UNIVERSITY OF CALIFORNIA

Los Angeles

Cave and City: A Procedural Reconstruction of the Urban Topography of Magnesia on the Maeander

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Architecture

by

Marie Saldaña

2015

ABSTRACT OF THE DISSERTATION

Cave and City: A Procedural Reconstruction of the Urban Topography

of Magnesia on the Maeander

by

Marie Saldaña

Doctor of Philosophy in Architecture

University of California, Los Angeles, 2015

Professor Diane Favro, Chair

Caves and cities are normally found at the opposite ends of the architectural spectrum. The former is typically perceived as a primitive, undesigned object, part of nature; the latter is held as the height of civilized planning and rationality. At the Greco-Roman city of Magnesia on the Maeander in western Turkey, however, these two typologies were intimately bound together in unexpected ways. Magnesia on the Maeander was the site of the ancient cult of a cave-dwelling Apollo at Hylai, whose devotees were sometimes imbued with a supernatural strength that enabled them to jump from high cliffs and rush through the mountains carrying entire trees that they had torn up by the roots. City and cave appear to have been linked through a complex

dynamic of movement in which the frenzied tree-carriers ended up depositing their burdens at a city-center sanctuary of Dionysos. This movement provides an alternative way of understanding the layout of the city, and augments the scant archaeological remains of the street grid. Drawing on textual and empirical evidence, this study seeks to understand if the city grid, its main thoroughfares, and the orientation of its buildings were influenced by the desire to connect to important topographical features such as the cave of Apollo.

The reconstitution of these dynamics into a synthetic urban topography requires the collation of incomplete evidence from a variety of different sources and time periods, including historical accounts, numismatics, inscriptions found on the site, architectural remains, geographical surveys, and the preservation of toponyms in local memory. This project makes use of procedural modeling, an emerging 3D mapping technique that facilitates the generation of hypothetical three-dimensional visualizations based on geographically located data to explore a series of focused questions about the impact of cultic practice and landscape on the urban layout of Magnesia on the Maeander. The first set of questions is spatio-empirical in nature, and concerns the reconstruction of Magnesia's city plan. The second seeks to elucidate a 'topography of ritual' at Magnesia and explore the ways in which this network of natural places and religious sites coexisted with, and undermined, the spatial system constituted by the city grid. The idea of the cave and its role in Greek architecture is the third object of study, as represented at Magnesia by the sanctuary of Apollo. Finally, these questions form a case study for a critique of the value and potential of the procedural modeling methodology. As a whole, the project aims to contribute to knowledge of the landscape and built environment of Asia Minor in the archaic through Roman periods, as well as entering into discourses of theory and praxis in the Digital Humanities.

The dissertation of Marie Saldaña is approved.

Sylvia Lavin

Michael Osman

Chris Johanson

Diane Favro, Committee Chair

University of California, Los Angeles

2015

DEDICATION

To my parents, and Ashley

TABLE OF CONTENTS

LIST OF FIGURES

All figures and illustrations are by the author unless otherwise noted.

ACKNOWLEDGEMENTS

My thanks go first and foremost to my committee, especially my dissertation advisor, Diane Favro, whose patience and guidance have been a resource of sanity for the last five years. This dissertation would not have been accomplished without Chris Johanson, who first introduced me to procedural modeling and whose mentorship has been a consistent asset in my journey. I extend warm thanks to Sharon Gerstel and the department of Art History at UCLA for invaluable support in the form of a Dixon Fellowship as well as the opportunity to explore fieldwork and intellectual horizons beyond my discipline. I am very grateful to Orhan Bingöl for his generous spirit of collaboration on the Magnesia excavations and surveys. Without his cooperation I would not have had the opportunity to visit and study such a fascinating site. Pascal Mueller and Matthias Buehler patiently fielded many questions from me in my efforts to learn CityEngine. Finally, I cannot forget the kindness and hospitality of the Kardeş family during my stay in Turkey.

Marie Saldaña received her bachelor's degree in Humanities from the University of Southern California in 2001. She earned master's degrees in Archaeology from Durham University (2002), and in Architecture from the University of California, Los Angeles (2010).

Chapter 1: Introduction

Caves and cities are normally found at the opposite ends of the architectural spectrum. The former is typically perceived as a primitive, undesigned object, part of nature; the latter is held as the height of civilized planning and rationality. At the Greco-Roman city of Magnesia on the Maeander in western Turkey, however, these two typologies were intimately bound together in unexpected ways. Magnesia was founded in the archaic period by settlers originally from Magnesia in Thessaly, who arrived in Asia Minor by way of Crete. In the early 4th century BC the city was transferred from an undetermined location near the Maeander River to the site its ruins occupy today, nestled between the foothills of Mt.Thorax (modern Gümüşdağ) and a bend in the Lethaios River (modern Gümüşçay). The site is near the village of Tekinköy, on the road between Ortaklar and Söke. Magnesia is most famous for its temple of Artemis Leukophryne, which is believed to be the work of the architect Hermogenes, as its plan accords with the description given in Vitruvius of the innovative *eustyle* (Vitr.3.3.6-9). A few other major buildings such as a theater, stadium, bath and gymnasium are still visible today, but the street grid remains buried under many meters of alluvium deposit from the annual flooding of the Lethaios.

Also little-understood are Magnesia's intriguing ritual practices, especially those of the ancient cult of a cave-dwelling Apollo at Hylai. Apollo's followers, known as *dendrophoroi* after representations on Magnesian coins,[1] are described in passage of Pausanias (10.32.6) from the second century AD:

[1] See section 3.1 below.

> There is also near Magnesia on the river Lethaios a place called Aulai,[2] where there is a cave sacred to Apollo, not very remarkable for its size, but the image of Apollo is very old indeed, and bestows strength equal to any task. The men sacred to the god leap down from sheer precipices and high rocks, and uprooting trees of exceeding height walk with their burdens down the narrowest of paths.

City and cave appear to have been linked through a complex dynamic of movement in which the frenzied tree-carriers ended up depositing their burdens at the city-center sanctuary of Dionysos. This movement provides an alternative way of understanding the layout of the city, and augments the scant archaeological remains of the street grid. Drawing on textual and empirical evidence, this study seeks to reconstruct the city plan, and understand if its main thoroughfares and the orientation of its buildings were influenced by the landscape and its network of natural places of religious significance.

The reconstitution of these dynamics into a synthetic urban topography requires the collation of incomplete evidence from a variety of different sources and time periods, including historical accounts, numismatics, inscriptions found on the site, architectural remains, geographical surveys, and the preservation of toponyms in local memory. Central to this project is the use of procedural modeling, an emerging 3D mapping technique that facilitates the generation of hypothetical three-dimensional visualizations based on geographically located data. 3D procedural models were instrumental to this work's research goals, which track the spatial presence of cultic practice and landscape on the urban layout of Magnesia on the Maeander, from the archaic period through the Roman era (roughly 800 BC – 400 AD).

[2] Originally translations of Pausanias read 'Hylai', 'Aulai' is a revision of Wilamowitz. The emendation, discussed in section 3.1 below, is critical in historical interpretations that connect the *dendrophoroi* to the cave cult of Apollo.

1.1 Disciplinary Framework

As an architectural historian with a background in design, my approach will differ from that which might be taken by a classicist or archaeologist. While I embrace the contributions of these fields and draw on their literature, I will be more concerned with reading the landscape, texts, and built environment of Magnesia with an eye to spatial relationships and practices. I attempt to elucidate the kinds of design thinking that may have been at play in the city's development over time. Likewise, my use of drawings, diagrams, and 3D models is intended not merely as illustration of my arguments but a way of thinking through problems visually and spatially.

While the use of 3D models to aid the study of past environments is not new, my contribution argues for procedural modeling as a methodology that has the potential not only to document, but to deepen and augment, the thinking processes of modeling. The emphasis throughout is on hypothesis testing and experimentation. This places my work in a growing line of research within the Digital Humanities that foregrounds the heuristic process of modeling. Willard McCarty has said that the difference between a model and an idea or other static representation is that that model can be manipulated: "computational models, however finely perfected, are better understood as *temporary states of coming to know* rather than fixed structures of knowledge (McCarty, 2004, p. 257). Yet if a model is constantly in flux and can never be considered truly finished, it must paradoxically reveal its source data and structures as discrete, finite entities. There is a recognized need for greater transparency in the sources and decisions and data that go into the making of 3D historical visualizations (Bentkowska-Kafel and Denard, 2012). Procedural methodology is proposed here as a means towards an open-ended, yet fully documented, modeling process.

1.2 Research Themes

I target several areas which have been underexplored in the literature to date. My research objectives and intended contributions are organized around three broad themes: Hellenistic city planning; the topography of ritual; and the idea of the cave in Greek architecture. A fourth goal of the project is to provide a case-study and critique of my methodology, procedural modeling for holistic urban reconstruction.

1.2.1 The City Plan

The first research theme is spatio-empirical in nature, and concerns the reconstruction of the city plan of Magnesia on the Maeander. The urban plan is a matter of speculation, since archaeology has only revealed the barest fragments of a grid. It was assumed by Magnesia's original excavator, Carl Humann, that Magnesia was planned on a grid according to Hippodamian principles. This conjecture has been echoed by the few other scholars who address the topic. The presupposition of a Hippodamian design strategy implies a proportional scheme in which whole numbers, expressed in the Greek foot, were used. While such designs have been demonstrated at several of Magnesia's neighbors, such as Priene, Miletus, and Ephesus, the dimensions of Magnesia's city blocks have never been convincingly established.

The reconstruction of the grid is important for several reasons. First, it knits together the urban fabric by expressing the routes via which traffic was carried through the city, connecting architectural nodes physically, visually, and kinetically. Pending the availability of further archaeological evidence, the identification of probable street grids and armatures may aid in the planning of future surveys and excavations. Most importantly for the current investigation, it

forms the basis for locating features of Magnesia's ritual environment that are known only through textual evidence and would otherwise remain inaccessible to spatial analysis.

1.2.2 The Topography of Ritual

The important role played by rural religious shrines in the development of Greek communities and social practices has been well established by scholars in diverse fields including anthropology, classics, and archaeology (Nilsson, 1961; De Polignac, 1995; Raja and Rüpke, 2015). At the site of Magnesia, the foundation of the city was predated by two shrines which were associated with prominent features of the landscape: that of Artemis Leukophryne, which was situated on the banks of a thermal lake and oriented toward the triangular peak of Mt.Thorax; and of Apollo at Hylai, in which an ancient statue of the god was housed within a small cave. As the city grew, new cults, notably that of Dionysos, were linked to the earlier ones by means of ritual movement and symbolic gesture. This 'polytheistic network', underpinned by the aspects of the natural landscape to which its important nodes are aligned, is spatially counterpoised to the arithmetically-derived Hippodamian grid system. The two are simultaneously enmeshed and conflicted. Movement and visibility between sites was constrained by the access provided by streets and architecture, although priorities of equitable land division and civic cohesion were in many ways at odds with a pre-existing polytheistic landscape oriented to features of the natural terrain. Because of the multiplicity of modes of reading the city and its site, the reconstruction of Magnesia's plan and the mapping of ritual and movement in relation to the city's topography are mutually dependent exercises.

The physical presence of the cults of Apollo and Dionysos at Magnesia has not yet been substantively detected in the landscape, although suggestive clues abound in inscriptions, ancient sources, and representations on coins. The curious phenomenon of the *dendrophoroi* has inspired

many commentaries and theories. One argument which I seek to evaluate with the spatial evidence provided by the reconstruction model links the *dendrophoroi* with the cult of Dionysos (Reinach, 1890; Kern, 1895; Robert, 1977; Henrichs, 1978; Detienne, 2002). An inscription (Kern, 1900, no.215) that describes an epiphany in which an image of Dionysos was discovered in a plane tree rent by lightning also records the names of three *thiasoi* that were founded in his honor. The names are descriptive and suggest links with Magnesia's urban topography. The mapping of the locations of the *thiasoi* and a spatial-architectural investigation of their relation with the site of the cave and the city plan, has never been undertaken, and forms part of the agenda for this study. There is much to be learned in the articulation of probable solutions. I discuss the ways in which the location of the cult centers might have influenced the urban layout, how ritual and movement between these cult centers were shaped by the city and its architecture, and how this may have changed over time. This discussion adds to the body of literature which addresses the interconnectedness of religion, landscape, and architecture in the ancient world.

1.2.3 The Cave

The third theme of this dissertation frames the cave of Apollo at Hylai within a broader perspective that encompasses the cave as object and idea in the Greek world. Caves were a pervasive and important part of the Greek landscape, and recent studies (Weinberg 1986; Ustinova, 2009) have surveyed the actual caves which can be identified from literature and archaeology. However, the cave is insufficiently understood as an urban and architectural *topos* in Hellenistic-Roman cities. This study undertakes to elucidate this construction through the examination of the specific case of the sanctuary of Apollo at Hylai.

With the notable exception of the Minotaur's Labyrinth at Knossos, the cave in ancient Greece, mythological or otherwise, was seldom explicitly architectural. The infiltration of the

cave into designed space was subtle and diverse. Clues to the operational nature of the cave can be found in the literature of the period. Florence Weinberg identifies three main categories of cave-topos found in the writers of Antiquity: the *locus amoenus*,[3] the monster's lair, and the sacred cave (Weinberg, 1986, p.5). In all three, she argues, the cave is a symbol of unformed, unrefined material being, which must be overcome in order for the intellect to transcend man's animal nature. This progression, from the darkness of the cave to the light of knowledge, has its most famous example in Plato's Allegory of the Cave (Plat.Rep.7.514a.2-517a.7). While this model appears time and again in theoretical discourses of the cave, in practice and in popular conception the Greeks viewed caves, along with other significant features of the landscape, as imbued with supernatural agency. By housing personifications of chthonic forces (for example, the caves of Calypso, the Cyclops, and the Nymphs in Homer's *Odyssey*) they were themselves an extension of that force, occupying an ambiguous territory between symbolic expression and autonomous power. Greek architecture did not interfere directly with that power – the cave is sometimes manifested as a highly abstracted reference, such as in the adyton of a temple; and at other times left natural and unchanged, connected with architecture by proximity or (as I argue is the case at Magnesia) by movement and ritualistic associations. Viewing the city of Magnesia vis-a-vis its nearby cave sanctuary will help to resituate the cave (physically and conceptually) in the urban fabric of Greek cities, and contribute to studies of landscape in ancient Greece & Rome.

[3] i.e. "pleasant place", which is alternately construed as a shelter, or a trap.

1.2.4 Procedural Holistic Urban Modeling

The three areas of historical inquiry introduced above also serve as a case study for the digital research methodology this project undertakes: procedural modeling for holistic urban reconstruction. The process of modeling an ancient city in 3D involves a good amount of interpolation, especially in cases such as Magnesia where firm archaeological evidence is scarce. Holistic urban modeling is predicated on the assumption that informed hypotheses are worth making in order to create digital simulations that aid the study of urban experience. Provided the sources for the model can be effectively managed and documented, there are substantive benefits to be derived from the attempt to give spatial presence to a wide range of aspects of an urban environment, including terrain, infrastructure, major monuments, and housing. In procedural modeling, each factor is linked with its context within the city as a whole, resulting in a productive laboratory for testing hypotheses of urban spatiality and change over time.

Procedural modeling entails the use of computer scripts, written out in text format, to generate parametric 3D models based on geographically-located data. Using the 'Roman City Ruleset', a suite of procedural rules I wrote to describe Greek and Roman building typologies, I generated a 3D urban model for each phase of the city's existence. This project is intended to serve as both a repository for the city's architectural data as well as a testbed for a speculative reconstruction of the urban plan, with the ultimate goal of analyzing how landscape and ritual played a role in shaping the urban fabric.

I treat the city of Magnesia on the Maeander as a case study to determine whether procedural modeling gives us a way to "phase" each possible element against the others, helping to identify alternatives which have a higher degree of probability. Procedural models can contribute to a holistic understanding of the city in relationship to its topography, ritual, and

broader context. The use of 3D modeling to fill in the street network with houses and other urban architecture aids in elucidating the visuality of the city and its important architectural monuments. The project also serves as the basis for a critique for the viability of procedural modeling as a methodology for archaeological reconstruction and digital humanities, and evaluates the usefulness of the Roman City Ruleset as a groundwork for future research. I have included the abridged text of some of the rules themselves in Appendix B. Although the complete, functional rules will be released in an updated version with the digital publication of the project, I have included them here in support of my argument that the code functions as a document of the decisions that went into making the model, and as a demonstration of the depth of the project. The attributes specific to Magnesia that were used in the model can be found in Appendix A.

1.3 Structure

The structure of the chapters follows a straightforward logic intended to bind together the three research questions and methodological critique in a coherent presentation. Chapter 2 provides the background of the topic at hand, the history and topography of Magnesia on the Maeander. Chapter 3 presents the 'problems', that is, the unresolved questions surrounding the cults of Apollo and Dionysos, as well as clues to how the cave may been conceptually embodied in the city's architecture. Chapter 4 discusses the methodology of procedural modeling and its implications for humanistic research. Chapter 5 presents the results of the 3D investigation, considering in turn the city plan, the landscape of ritual, and the cave's presence. Finally, an evaluation of the outcomes is offered, with a critique of the methodology and proposals for continuation of the research.

Chapter 2: Historical Topography of Magnesia

Because archaeological exploration at Magnesia has historically been centered on the agora and sanctuary of Artemis Leukophryne, architectural knowledge of the rest of the urban fabric, as well as the many other shrines attested by inscriptions and fragments, remains fragmented. At best, attempts at reconstruction have merely superimposed a Hippodamian-type grid over the area enclosed by the city walls, with little attempt to understand its relationship with the terrain, older settlement patterns, or diverse ritual spaces.[4] This chapter recounts the empirical and historical evidence necessary to map this interpenetration of architectural space, natural space, and ritual space.

Fig. 1 The triangular peak of Thorax and the pediment of the temple of Artemis. The pediment was reconstructed to the south of the actual temple site.

[4] See Bingöl, 2007 (pp.134-135).

2.1 Archaeology at Magnesia

Interest in finding and documenting Magnesia was largely inspired by the fame of the Artemis temple and precinct. British geographer William John Hamilton was the first to correctly identify the site around 1800.[5] Fellow Englishman and antiquarian William Martin Leake published an account of the determination of Magnesia's location, describing how it was to be sought on the road between Ephesus and Tralles beneath a mountain called Thorax, as described by Strabo (XIV.1.39; Leake 1824, p.243). After Hamilton, the French were the next Europeans to study Magnesia. Drawings were made in 1817 – 1821 by Jean-Nicolas Huyot, the site was visited in 1838 by Desiré-Raoul Rochette, and eventually an excavation of the temple was attempted in 1842 – 1843 by a team led by Charles Texier along with the architect Jean Jacques Clerget. The drawings by Clerget, as well as the other early documentation which show the location of structures such as the theater, odeion, and city gates, which have since all but disappeared, remain largely unpublished and are held in the Bibliothèque National in Paris. Since 1985 excavations of the site have been ongoing and conducted by Ankara University under the directorship of Orhan Bingöl.[6] The state of Magnesia's excavated urban fabric is partial, although much remains to be explored 4-5m below the surface of the farmland. The ruins are listed and described in detail by Bingöl (2007), but a brief summary here will serve to introduce the site as it developed over time, based on the remains that exist today (Fig. 2).

[5] This attribution is upheld by Leake (1924), Rennel (1831), and Rayet (1845).

[6] Professor Bingöl and his team have generously shared their drawings and excavation reports as part of a collaboration between Ankara University and UCLA's Experiential Technologies Center. The project, entitled "Digital Anatolia", encompasses endeavors of which this dissertation is but a small part, and includes an aerial digital survey of the site and interactive web presentation of 3D models and other documentation.

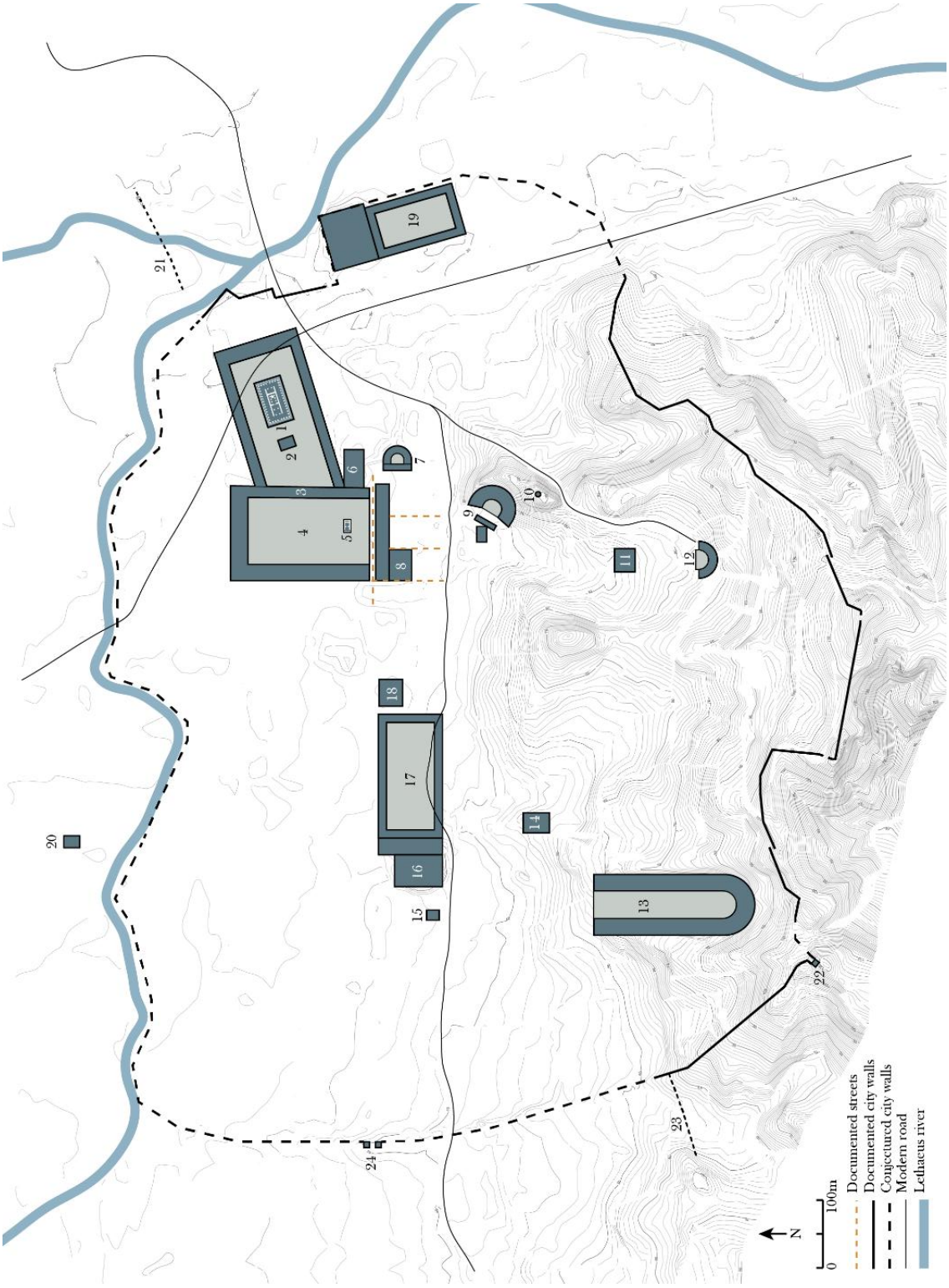

Fig. 2 Actual state plan of Magnesia on the Maeander. (1)Temple of Artemis Leukophryne (2)Altar of Artemis (3)Propylon (4)Agora (5)Temple of Zeus Sosipolis (6)Market basilica (7)Odeion (8)Prytaneion (9)Theater (10)Temple of Athena (11)Dioscuri (12)Theatron (13)Stadium (14)Serapeion (15)Temple of Dionysos (16) Gymnasium baths (17) Gymnasium palaestra (18) Large roman building (19) Lethaios gymnasium (20)Roman temple (21) Remains of bridge (22) Tower (23) Aqueduct

2.2 The Archaic Period (800 BC – 479 BC)

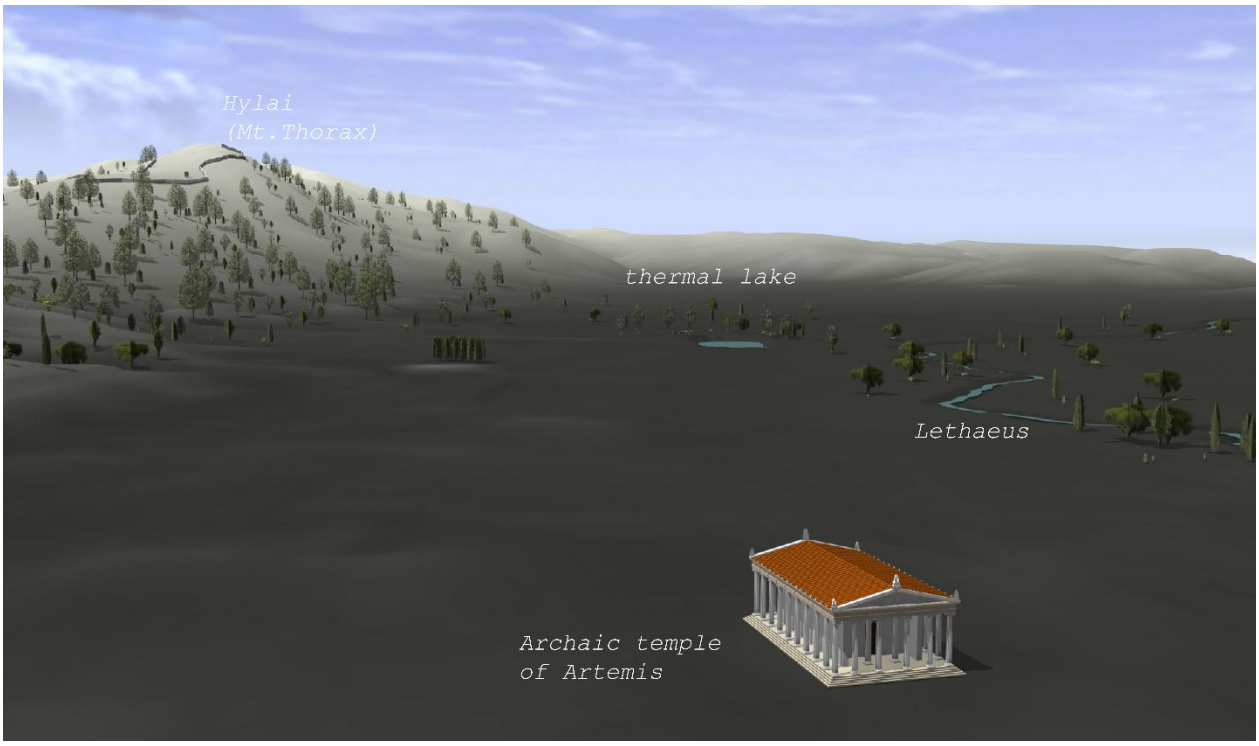

Fig. 3 3D procedural model of Magnesia in the archaic period.[7] The walled area on the peak is the site which may be identified with Hylai.

The city of Magnesia on the Maeander was originally located not at its present site, but approximately three miles away near the Maeander River. According to Diodorus Siculus (14.36), Magnesia's original site was 120 stades (15km) from Ephesus, a distance which corresponds with the confluence of the Maeander and one of its tributaries, the Lethaios. In archaic times this location would have been close to the gulf of Miletus, and on the Maeander's alluvium-rich northern bank.[8] Bingöl (2007, p.25) argues that the old city of Magnesia had been linked with the sanctuary of Artemis Leukophrene via a sacred path long before the famous

[7] This image, and all other images of 3D models, are original models by the author developed for hypothesis testing of the arguments presented in this dissertation.

[8] P. Thonemann's 2011 geographical history of the Maeander valley explains the geological conditions which led people to settle on the north bank of the Maeander from antiquity to the present day. The south bank of the Maeander does not drain directly into the valley and therefore lacks the rich alluvium deposits of the northern foothills. It is also prone to heavy flooding (Thonemann 2011, 10-14).

temple by Hermogenes was built, in which case the original city would have been connected to the present site through topographical memory.

The archaic temple of Artemis was, according to Xenophon, near "a lake of more than a stadium in length, with a sandy bottom and an unfailing supply of drinkable, warm water" (Xen. Hell. 3.2.19). Artemis was a goddess often associated with marshy places (Scully, 1979).[9] At Magnesia she may be an incarnation of the pre-Hellenic mother goddess (Cybele/Dindymene) to whom Themistocles built a temple and dedicated his daughter as priestess when he was ruler of Magnesia in the mid-5th century BC (Plut.Them.30.3). The lake would probably have been located in the marshy area of the plain near where the north-west corner of the city wall would later run. It may have been fed by a thermal spring that still exists in the village of Yeniköy (Bingöl, 2007, p.24). High ground water continues to flood the site seasonally. A settlement, perhaps called Leukophrys, may have occupied the higher ground towards the foothills, bordered in the west by the Lethaios. The archaic temple was hexastyle in plan, and made of local limestone. Column drums of the Ionic order, with Ephesus-type bases, were found in the foundations of the Hellenistic temple, and Humann suggested that wall remains below the eastern colonnade of the Hellenistic temple formed the foundation of the Archaic one (Humann 1904, pp.46-49). This would indicate that the two had the same orientation towards Thorax.

According to the founding myth of the city (Kern 1900, no.17)[10] the Magnesians, who had settled in Crete after leaving their original homeland in Thessaly, saw white crows flying overhead. Thinking this was a sign that they should return to their native land, they consulted the oracle at Delphi. The oracle told them that instead they were to settle in a different country, and

[9] The Artemision at Ephesus also lies on swampy ground.
[10] One of the most complete such inscriptions that survive. An English translation is available in Bingöl 2007, 16.

that they would be led there by a man called Leukippos ("white horse"). Leukippos led them to a place called Mandrolytia, where the king's daughter, Leukophryene, fell in love with him and betrayed her father's city by opening the gates to Leukippos, thus allowing him to establish the archaic city of Magnesia on the Maeander. The recurrence of the term "white" -- in the color of the crows, the names of the hero and heroine of the founding myth, the name of the shrine of Artemis -- has been linked by Bingöl to the presence of silvery-white colored magnesium stone in the mountains near the city, although the name "magnesium" refers to the region in Thessaly, and was not associated with the metal until the eighteenth century. In antiquity the "stone of Magnesia" mentioned by Euripides and Plato refers either to magnetic iron-stone or to silver. [11] These references can be more confidently traced to Magnesia in Caria.[12] If the "stone of Magnesia" was originally silver this would be borne out by the presence of modern silver mines in Mt. Thorax, which is called Gümüşdağ (silver mountain) in Turkish, while the Lethaios River is known as Gümüşçay (silver stream), and a nearby village is called Gümüşköy (silver village). In any case, Mt.Thorax and the sanctuary of Artemis, both seem to have been tied together in associations of whiteness and luminosity since the earliest settlements in the region.

[11] "Looking at men's opinions, he attracts and then releases their belief like Magnesian stone". (Euripedes,Oeneus, 567); "[A] divine power, which moves you like that stone which Euripedes named a magnet, but most people call "Heraclea stone".

[12] This mainly rests on the "Heraclea stone" being linked with the presence of places called Heraclea near Magnesia. It may simply mean "Hercules' stone", i.e. magnet stone rather than silver (see note 15).

2.3 The Classical Period (480 BC – 322 BC)

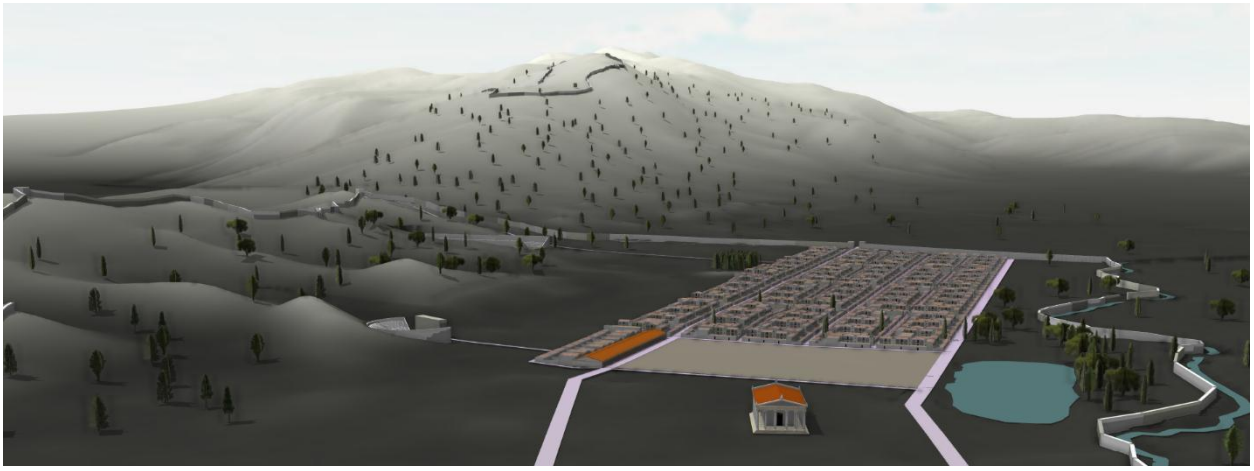

Fig. 4 3D procedural city model of Magnesia in the Classical period. At this time the city blocks had been laid out, but the stoas surrounding the agora and temple of Artemis had not yet been built.

As the mouth of the river gradually silted up the Maeander valley, the ancient maritime cities on the gulf of Miletus were abandoned. Around the same period, the need for defenses became pressing, and the original Magnesia did not possess any defensive walls or topography. Diodorus (16.36) recounts that Thibron, a Spartan commander who had come to Magnesia via Ephesus to fight against the Persians in 400 BC, was concerned that the unfortified location of the city would lead the Persians to recapture it after his departure. He therefore temporarily moved the population to the foothills of Thorax. Bingöl (2007, p.30) identifies this location with the remains of an ancient settlement called "Büyük Manastır" (large monastery) by local inhabitants and "Leukophrys" by scholars (Fig.5).[13]

[13] Following Alfred Phillipson (1936), Bingöl calls this place Leukophrys after the passage in Xenophon (3.2.19), though the toponym associated with Xenophon's description, unless it is meant to cover a very general area including the foothills of Thorax, seems rather to describe the location of the Artemision. Philippson's map delineates the borders of the site he calls Leukophrys. I could find no other map locating the site, though Humann (1904, Blatt I) does show a "watchtower" in this location (Fig.11). Bingöl's mentions of the "Büyük Manastır" (2007, p. 30) certainly correspond with this site, though he does not indicate its precise whereabouts. According to H. Lohmann (2006), Philippson "identified a fortified ancient settlement with orthogonal street layout" at this location, although I have not been able to verify this.

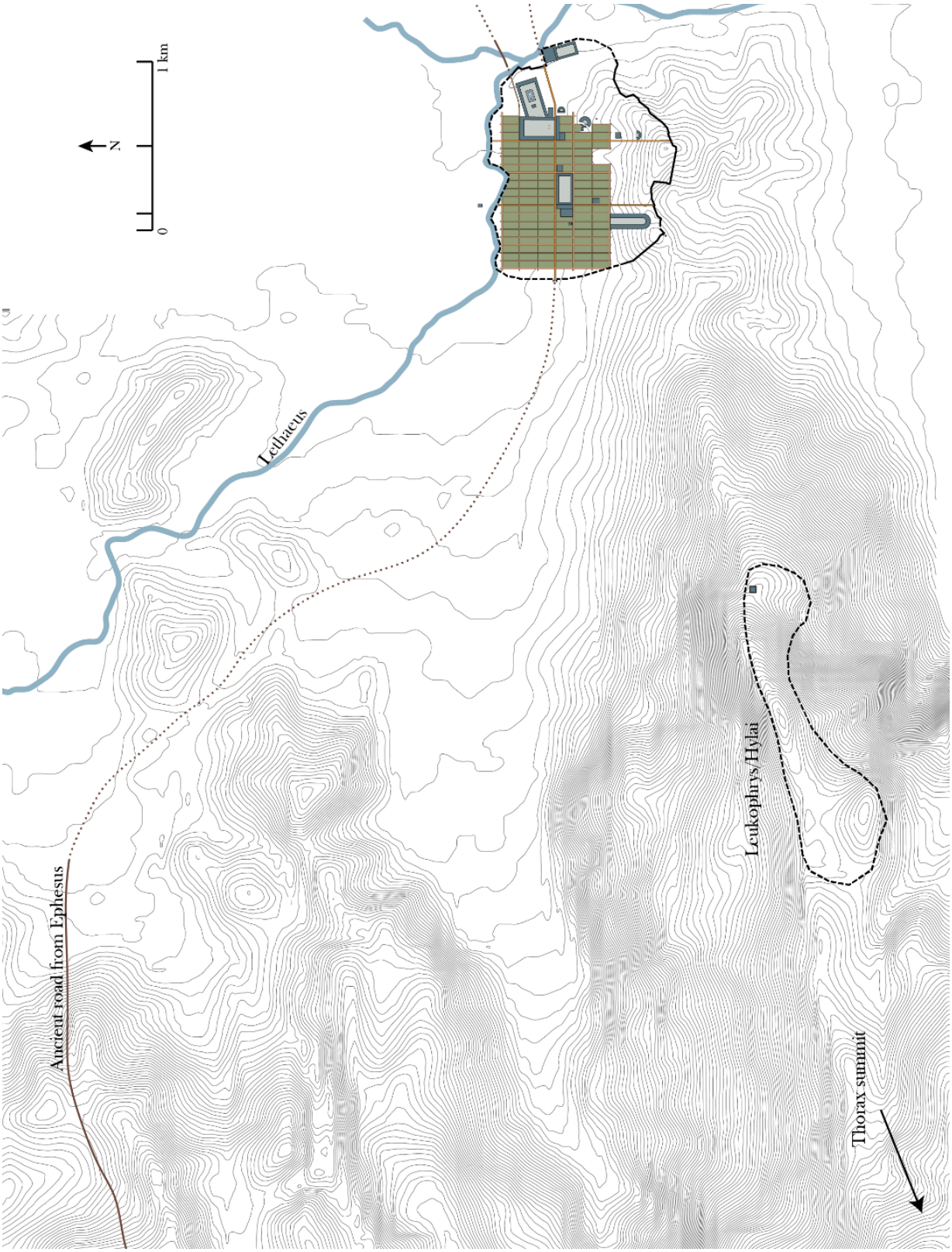

Fig. 5 Plan of Magnesia and environs. Road and site of Leukophrys/Hylai after Philippson (1936).

It was below this mountain fortress that the Magnesians founded their new city sometime after the Peace of Antialkidas in 386 BC (Bingöl 2007, 31).[14] At this time, around 60 years after Hippodamus planned the new city of Piraeus, it is likely that a grid would have been laid out and the basic dimensions of its city blocks devised. The agora, theater, and stadium must have existed at the founding of the city, but their form in this period is unknown, obscured by later renovations. Most probably at this point, the agora was simply an open space carved out from the city grid and based on the dimensions of the insulae, perhaps flanked by freestanding stoas or buildings.

2.4 The Hellenistic Period (323 BC – 145 BC)

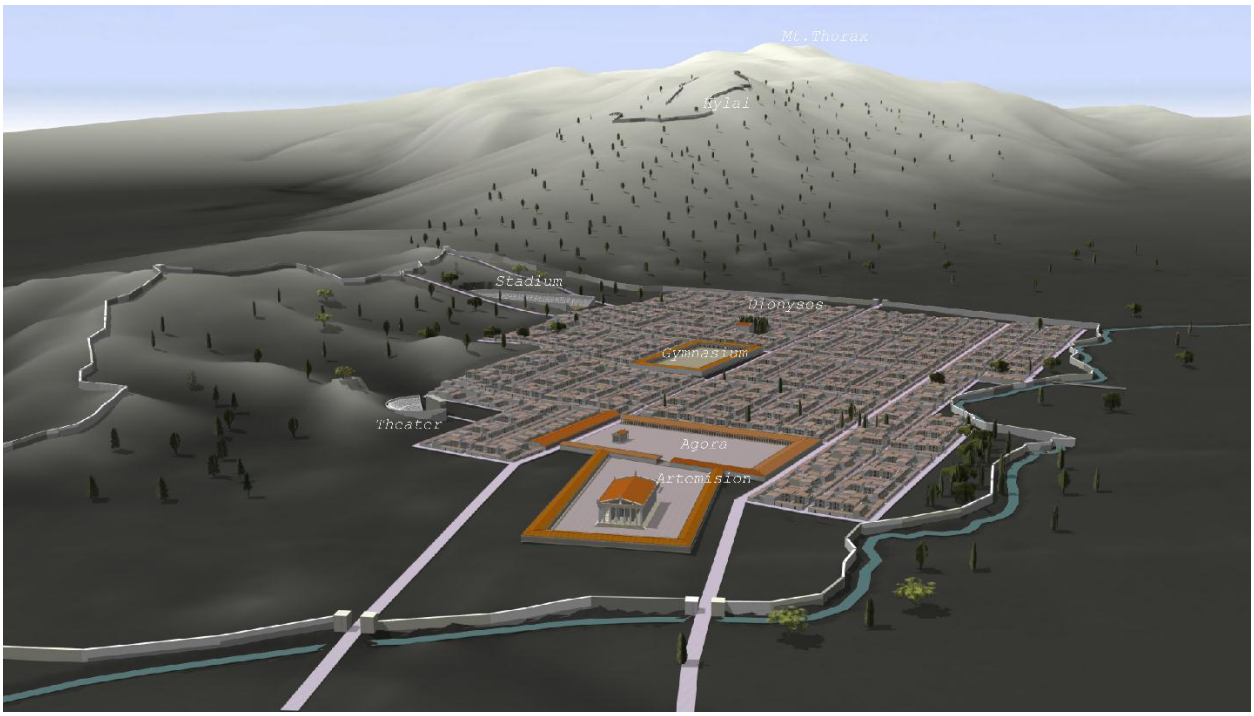

Fig. 6 3D procedural city model of Magnesia in the Hellenistic period. The city grid has expanded and the stoas in the city center have been constructed. A grove of trees marks the site of the temple of Dionysos.

[14] See also Strabo (XIV.1.58)

In the second half of the third century BC Magnesia saw a period of increased architectural development that left its mark on the major monuments of the city. Following an epiphany of Artemis in 220 BC that gave rise to the institution of the Leukophryena games (Kern 1900, No.16),[15] the city invested in renovating its public buildings. It is to this phase that the construction of the pseudodipteral temple of Artemis, attributed to Hermogenes, belongs. In addition to the temple, the stoas surrounding the sanctuary, the altar of Artemis, the stoas of the Agora, and the temple of Zeus Sosipolis (another possible work of Hermogenes), all belong to this period (Humann 1904, p.22).

Fig. 7 The Artemision in 2010. View towards the south.

[15] After the epiphany the Magnesians consulted the oracle of Apollo at Delphi, who instructed them to institute an *agon* (athletic competition). However, it was fourteen years before the plan was realized. The responses from the cities of Greece who responded to Magnesia's invitation to the games were displayed on the walls of the agora. See Bingöl (2007, pp.65-67); and Thonemann (2007).

The Artemision sanctuary (3rd -- 2nd c. BC), Magnesia's most extensively excavated area, is visible from the highway.[16] The orientation of the temple of Artemis was clearly important enough to be preserved and even emphasized by its enclosure. Ritually, the Magnesians remained oriented towards Thorax even as their city plan took a more pragmatic shape from the cardinal directions.

Fig. 8 3D model of the Artemision

The Agora at this time took on the form of the 'Ionian type' clearly exemplified by the agoras at Miletus and Priene. This comprised a U-shaped stoa bordered at the south end by the main road of the city, with another stoa with shops behind closing off the U-shape to form a complete rectangle. In the Hellenistic period, the road was probably left open to traffic, gates only being interposed in the Roman period (Humann 1904, pp.109-110). The agora, although partially excavated by Humann in the late nineteenth century, is now mostly underground again, barring recent excavation work on the east stoa that has resulted in the discovery of a

[16] The Artemision was fortified by thick stone masonry walls in the Byzantine period, by which time the population of the city had shrunk dramatically. These walls, which make use of spolia from the earlier Hellenistic city walls, are still standing in large stretches and are a prominent part of the ruins today. Though they hinder the excavation of earlier material, they cannot be removed without damaging the structures.

cryptoporticus.[17] Within the agora stood the Hellenistic temple of Zeus Sosipolis, conjectured also to be the work of Hermogenes, since it typifies the *eustyle* described by Vitruvius (3.3.6).

Fig. 9 Procedural 3D model of the agora and temple of Zeus Sosipolis in the Hellenistic period. View from the west entrance to the Agora. The position of the temple of Zeus obscures the larger temple of Artemis behind it.

If the chronology given in a 2nd c AD inscription recording the foundation of the cult of Dionysos can be substantiated, then a temple of Dionysos was also built around this time.[18] A temple of Serapis is also presumed to date to the Hellenistic period based on its early 2nd-century dedicatory inscription (Kern, 1900, no.99). The round temple of Athena on the hill above the theater was also added in the Hellenistic period, as was a temple dedicated to the Dioskouroi (Humann, 1904, p.27).

[17] See p.43 below for further discussion of the cryptoporticus.
[18] See further discussion on the founding of Dionysos' cult in section 2.3.2 below.

2.5 The Roman Period

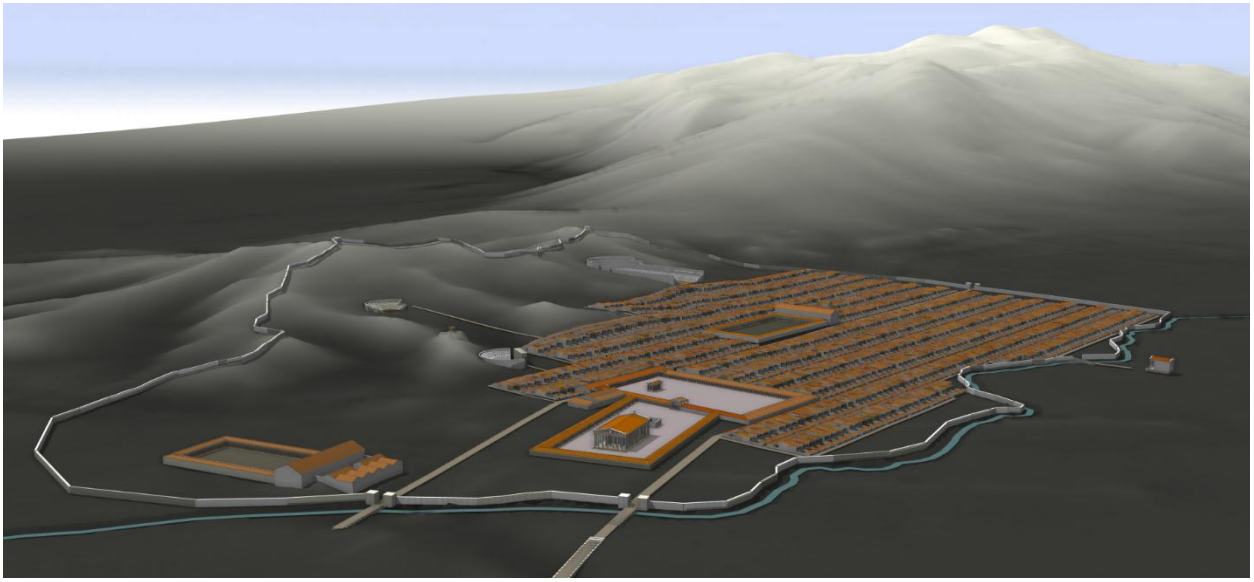

Fig. 10 3D procedural model of Magnesia in the Roman period. The two large bath/gymnasium complexes are now visible.

Several large public buildings were added or restored during the Roman period. In the 1st century AD, the propylon between the agora and Artemision was built along with the pavers of the "assembly area" immediately in front of the propylon. Adjacent to the agora a market basilica dating from the 2nd century AD has also been recently excavated and documented. South of the Artemision, a late Roman house with a hypocaust system has been uncovered, although in the same vicinity a Roman odeion, recorded by Humann, has been completely obliterated. A small, unfinished theater of the 1st century AD, unknown to early excavators, was uncovered by the Turkish team in the foothills to the south of the Agora. The unusual layout of this building, in which the seats for the audience had to be reached via the orchestra, is conjectured to have been intended for religious use (Bingöl, 2007, pp.157-158). Construction work on the small theater was probably discontinued due to a landslide.

Further west along the main route of the city lies the Roman gymnasium with its associated baths and palaestra. The apodyterion of the gymnasium is one of the best-preserved

buildings in Magnesia, although it has not been fully excavated. A second gymnasium/bath complex, also from the Roman period, lies in the eastern part of the city next to the Lethaios River. The structure is aligned to the south-west, like the Artemision, in contrast to the rest of the city grid, which is oriented to the cardinal directions.[19]

Fig. 11 Magnesia's stadium in 2014

One of the most impressive buildings of Magnesia, the stadium (Fig.11), was covered by hillside in Humann's time. Recent excavations have removed much of the earth on the east side of the stadium, revealing a large structure containing numerous inscriptions. The stadium was remodeled in white marble during Roman times, and had an imposing arcaded sphendone and starting block. Another building from the Roman era is a small unexcavated temple that stood on a high podium on the northern side of the Lethaios. The podium contains a vaulted substructure 2.5m high, according to the drawing published by Humann (1904, p.31).

[19] Bingöl (2007, 168) considers this simply a response to the course of the Lethaios, though the state of this area in the original plan of the city, perhaps influenced by a prior settlement in the area of the sancturary of Artemis, should also be considered (see Ch.5.1 below).

Fig. 12 Magnesia's "tumulus" hill, seen from the orchestra of the small theater

2.6 Summary

The layout of the city remained essentially the same from the city's founding to the Roman Imperial age, but the Roman enlargement of the gymnasia and addition of baths, market basilica, and odeion, as well as the marble renovation of the stadium, suggest a prosperous population eager to be entertained and to impress visitors from other cities. By contrast, almost all of the city's known religious architecture, including the temples of Artemis, Zeus, Athena, Dionysos, Serapis, and the Dioskouroi, appear to have been built in the Hellenistic period, or at any rate retained their Greek aspect. Perhaps this indicates that Magnesia's religious culture remained Greek and saw little need to change, even as its citizens acquired Roman recreational habits. This pattern seems to have held true for Magnesia's civic architecture and identity as well. Unlike the city center at Aphrodisias, which became increasingly monumentalized under Roman rule (Ratté, 2002), changes to the agora and Artemision at Magnesia were relatively small in scale: the

addition of the propylon connecting the two spaces, the addition of gateways to the agora entrance, a library and latrine behind the stoas of the precinct of Artemis. These rather small changes had great impact, however – they served to articulate the transitions between public areas: between street and square and sanctuary, in the manner of Roman armatures elsewhere in the empire (MacDonald, 1988). Thus while Magnesia retained its Hellenistic orthogonal plan and civic architecture, it did acquire some aspects of the "passage architecture" that gave Roman cities their characteristic spatial choreography.

It was within this context that the rituals linking the city of Magnesia to the cave of Apollo were defined. The 3D model, as will be demonstrated in Chapter 5, allows the full impact of small-scale changes to be assessed. A relatively minor intervention, such as the propylon, would barely register on a two-dimensional plan, where its footprint looks much the same as the adjacent stoa. However, realized in three dimensions, the impact of the propylon marking a monumental transition truly comes to the fore. The effect of Hellenistic planning also becomes evident. The 3D model demonstrates how a visual sequencing was achieved by the strategic placement of the Zeus temple in the Agora. As a visitor arrived at the agora's west entrance, the spectacular temple of Artemis was fully concealed behind the much smaller temple of Zeus (Fig.9). It was only on approach to the temple on axis that the space unfolded fully.[20] The spatially-oriented, visual approach afforded by the holistic urban model allows these instances of design thinking to become perceptible and linked with the broader context of the city's development.

[20] The location of the entrance to the Artemision in the Hellenistic period is not clear, although it presumably was at the center of the agora's east stoa where the Roman propylon was later interposed.

Chapter 3: The Cave, the Tree, and the City

At the time of its re-founding on the slopes of Mt.Thorax in the early 4th century BC, the city of Magnesia would have encountered a rural landscape that was already ripe with meaning. M. Nilsson emphasized that in the popular imagination, the gods were already present in nature, before planners and architects superimposed their rationalized vision of urban life:

> Anyone who wishes to understand the religion of antiquity should have before him a living picture of the ancient landscape as it is represented … in Strabo's description of the lowland at the mouth of the river Alpheus. "The whole tract," Strabo [(VIII.3.86)] says, "is full of shrines to Artemis, Aphrodite, and the nymphs, in flowery groves, due mainly to the abundance of water; there are numerous hermae on the roads and shrines of Poseidon on the headlands by the sea." One could hardly have taken a step out of doors without meeting a little shrine, a sacred enclosure, an image, a sacred stone, or a sacred tree. Nymphs lived in every cave and fountain. This was the most persistent, though not the highest, form of Greek religion. (Nilsson 1961, pp.17-18)

Such practices illustrate the concept of "lived religion" as applied to the ancient world (McGuire 2008; Raja and Rüpke, 2015, pp. 3-4). The formal sanctuaries of institutionalized religion were but one aspect of the space of ritual in cities like Magnesia. The complex polytheistic underpinnings of the region were not completely subsumed within the axial grid that was imposed when the city was founded in the Classical period. In fact, these older relationships – between diverse gods, their shrines, and the landscape – may help us to form a more complete understanding of Magnesia's urban layout. Because archaeological remains of the city plan are scant, the topographical data contained in inscriptions describing cultic practices in the vicinity provide an alternative means of mapping the city. M. Detienne (2002, 148) argues that this 'grass roots' approach, starting from "the objects, the acts, the particular situations presented by the primary data, using these as so many reagents to see what aspect of this particular divine power comes to the surface in a given configuration" can help avoid the trap of over-generalizing the

symbolic interpretations that the presence of a temple devoted to a given deity might evoke. Allowing "an experimental dimension somewhat analogous to qualitative analysis in chemistry" to guide such investigations leads to the identification of "a range of possibilities, without which polytheism remains opaque, a dead system (*ibid*.)." It is precisely such a range of possibilities for the spatial and architectural configuration of Magnesia that the present work seeks to elucidate from the textual, archaeological, and topographical data.

I will focus my discussion on the evidence concerning the lesser-known rituals surrounding Apollo of the cave at Hylai and Dionysos "of the plane tree". These two cults, particularly when considered in conjunction with one another, will enrich the experimental reconstruction of Magnesia's urban topography which is the purpose of this study. Movement is a critical factor here, located through multi-modal analysis of the rituals that connected the two religious centers, that brings the reconstruction into the third (spatial) and fourth (temporal) dimensions. Significantly, this was movement that crossed the boundaries of the city walls, constituting a multivalent perspective that encompassed the city and its architecture both from within and without. This chapter works from the precepts set out by Nilsson and Detienne to uncover the overlapping polytheistic dynamics that are reflected in Magnesia's urban development.

3.1 The Cave of Apollo at Hylai

From the earliest time of its formation, Magnesia was associated with a cave sanctuary dedicated to Apollo. It is possible to reconstruct the outlines of this cult from several pieces of textual and numismatic evidence. The first indication we have for the cave-dwelling of Apollo comes from the *Letter to Gadatas* written by the Persian king Darius sometime during his reign (522-586

BC) in which the latter reprimands Gadatas for exacting tribute from the "sacred gardeners of Apollo":

> The king of kings, Darius, son of Hystaspes, to his servant Gadatas speaks as follows: I understand that you do not obey every point of my instructions…. Since you choose to disregard my desires as regards the gods, I shall cause you to experience, if you do not change, my wrath excited by an injury. The sacred gardeners of Apollo have been subjected by you to tribute and required to work profane land; that is to disregard the sentiments of my ancestors toward the god who said to the Persians ……[21]

The letter is known from a second-century AD Greek inscription, now located in the Louvre, found on a marble stele in villager's garden in present-day Germencik, which lies on the road between Magnesia and Aydin (ancient Tralles) about 6 km east of the city of Magnesia. The text is generally accepted as a second century A.D. Greek "re-publication" of an actual Persian-era letter, although some (Hansen, 1896) doubt its authenticity due to the apparent irony in Darius' professed reverence for Apollo – during the reign of Cyrus, the Persians had burnt the great temple of Apollo at Didyma to the ground. However, others (Briant, 2002) argue that the tone of the letter is consistent with Persian epistolary style from the period. Whatever Darius' motivations for protecting the gardeners of Apollo, the letter seems to have been re-aired as a confirmation of the special privileges held by them, which would have been further reinforced by the great antiquity of their sanctuary and its status.

At the time of Darius, Magnesia would have been situated at its original location somewhere in the Maeander valley. We can therefore draw the conclusion that a cult of Apollo in the vicinity of the current site pre-dated the city itself. After the new city had been established, evidence for a cult of Apollo involving "*dendrophoroi*" – men who carry trees – appears on

[21] English translation from Briant (2002). Original published in Cousin and Deschamps (1889).

Roman coins of Magnesia of the 3rd century AD (Robert, 1977, p.78) (Fig.13). The cult is attested as late as the end of the 5th century AD by a reference to *Appollonos Aulai* in the life of Isidore by Damascius. If all of these documents do in fact refer to the same cave cult, then it would have existed for at least 1,000 years in the vicinity of Magnesia.

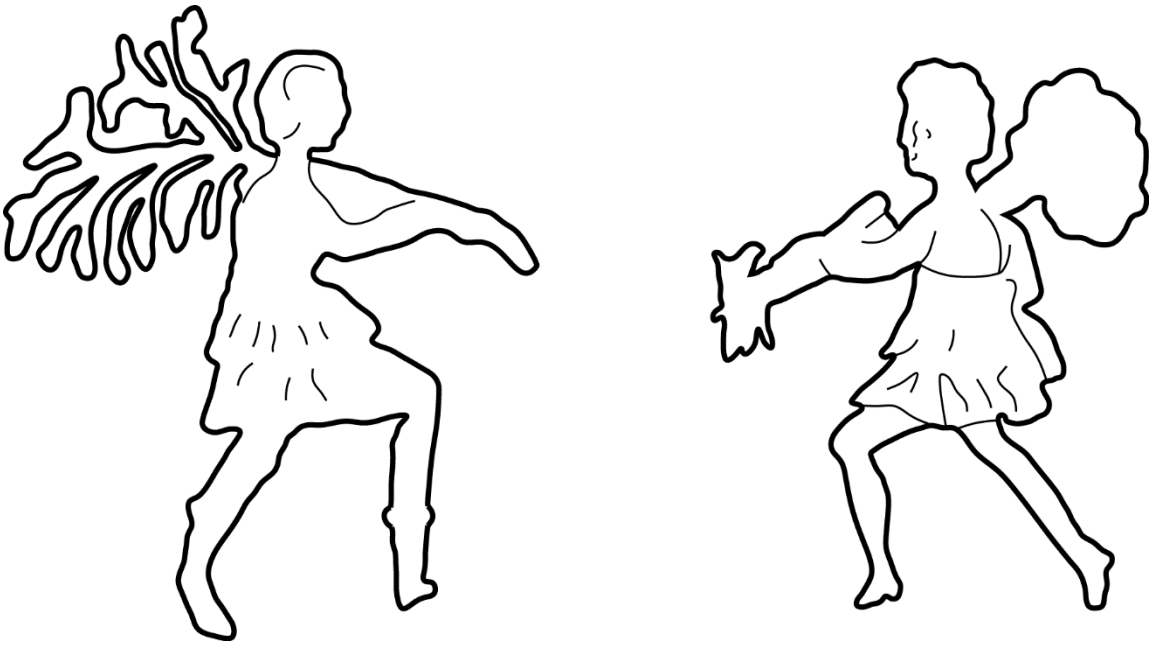

Fig. 13 Images of dendrophoroi from the coins of Magnesia. Left: Gordian III (225 – 244 AD); Right: Otacilia (244 – 249 AD). After Robert (1977, Fig.15, p.78).

The location of the cave, however, remains a mystery. Texier (1849, 90) was confident it could be discovered in the hills to the north of Magnesia, near the village of Gümüş:

> Following the slope of the mountain towards the village of Gumuch, halfway up one can see a large cave open to the south. I have not found in the these ruins any inscription that could tell me the ancient name of the village; but this cave is, I think, sufficient evidence to recognize the village of Hylae, which, according to Pausanias, was Magnesia's neighbor, and distinguished by a cave dedicated to Apollo in which it preserved a very ancient statue of the god.[22]

[22] This passage, as well as all the following quotations from Texier, Rayet, Kern, and Robert, are my translation.

Rayet (1877, 132) confirmed that Texier's conjecture was "probably accurate", while conceding that "I haven't seen it myself, and I haven't learned that there existed in the environs of Magnesia any other remarkable cave". Kern (1895, 93 n.1) dismissed this conjecture, asserting that while there are actually three cavities above the village of Gümüş, "those caves are nothing more than quarries". Moreover, he had "searched for this cave very often, but always in vain".

The toponym for the site of Apollo's cave was identified as Hylai in Pausanias; an alternate transliteration was proposed by Wilamowitz (1900, p.572, n.3) which changed the name to Aulai in order to bring it in correspondence with Magnesian coins of the 1st-3rd c AD which bear the epithet "Apollo Aulaites" and depict the figure of Apollo Chitharoedus (Schulz 1975, p.39). Robert (1977, p.83) supports Wilamowitz' revision, and 'Aulai' has since been widely adopted in translations of Pausanias. However, Ustinova (2009, p.120) points out that Hylai could derive from Hyllouala in Caria, where there was a sanctuary and oracle of Apollo. Whether or not these were one and the same, the indigenous Carian etymology of Hyllouala would support the conjecture that the cave-dwelling Apollo was a Hellenized incarnation of an older god. Ustinova assumes that the sanctuary would have functioned, like many dedicated to Apollo, as a *manteon* or oracular site. While there is no evidence in the texts that suggests the "gardeners of Apollo" did anything except enact a form of *oribasie,* the exalted frenzy that led them to uproot and carry whole trees, it should be noted that most natural or architectural caves dedicated to Apollo did have some oracular function (Ustinova, 2009, pp.109-155). Ustinova further proposes that Hylai could also be identified by a place called Hieracome, mentioned by Livy in the time of Augustus (38.13):

> [From Magnesia] after crossing the Maeander they marched to Hiera Come. Here there was a noble temple to Apollo and an oracular shrine; it is said that the priests delivered the responses in smooth and graceful verses.

Apart from being mentioned after Magnesia, there is nothing in this passage which corresponds with our picture of Apollo and the *dendrophoroi*, nor does Hieracome bear much etymological resemblance to Hylai. By all other accounts, Apollo lived in a small cave, and there is no mention in Pausanias of a "noble temple", or an oracle. Furthermore, the behavior of the *dendrophoroi* doesn't harmonize with the delivery of "smooth and graceful verses". The geographical detail of Hieracome being reached after crossing the Maeander also doesn't agree with the other evidence. If Apollo's cave was in the hills near the south bank of the Maeander, it would be at a distance of at least 8 km from the city, rather far for the *dendrophoroi* to carry their trees (not to mention necessitating a river crossing as well),[23] whereas their presence on the coins of Magnesia suggests that they were associated with the city's identity and probably geographically proximate as well. Furthermore, in the last known account of the sanctuary at Aulai/Hylai,[24] in the life of Isidore by Damascius, two late-5th century philosophers make a pilgrimage to the sanctuary from their home in Aphrodisias. The narrative describes them swimming across the Maeander to reach the sanctuary and almost drowning.[25] Since Aphrodisias is located south of the Maeander, this would seem to indicate that the Hylai was north of the river, in other words where the city of Magnesia was located.

Despite the difficulties in pinpointing the exact location, there are certain conclusions that can be drawn with confidence. One is that a sanctuary dedicated to Apollo existed in a cave outside the city in the hills or mountains, not only because this is geographically where caves are

[23] By way of comparison, the Nyseans carried a live bull 2km to the Plutonium at Acharaca (Strabo 14.1.44), but this route was relatively level throughout. Apart from the greater distance, the south bank of the Maeander represents a conceptual difference – it was an unsettled area far from the well-travelled route that connected Ephesus, Magnesia and Tralles, whereas Acharaca and Nysa were linked by topography and roads.
[24] In this text, *Apollonos Aulai* is named, instead of Hylai. Henceforth in the text it is assumed that the two are synonymous and 'Hylai' will be used for the sake of simplicity.
[25] Transcribed in Photius, extract 116.

found but also because of the "sheer precipices" and "narrow paths" described by Pausanias. It is clear that the cave was in a wooded, rural area that was nevertheless proximate to the city of Magnesia, since the *dendrophoroi* were uprooting fully-grown trees, yet are depicted on many of the coins of the city, thus placing them firmly within the imagery associated with the city's identity. It is likely that Hylai was located on the slopes of Mount Thorax instead of the hills to the north (above the village of Gümüş) because of the higher and wilder elevations of Thorax. There is archaeological evidence in support of this theory as well. A statue of Apollo, clad in the long Chitharoedus garment in which he is depicted on the coins bearing the epithet *Apollonios Aulaites*, was found in the hills southwest of Magnesia, near Argavlı village, in 1995 (Bingöl 2007, 179). The proximity of the location of the find with the nearby "Büyük Manastır" site, where Thibron is presumed to have brought the Magnesians to safety before relocating the city to the plain, is further evidence in support of the conjecture that the fortified hilltop site was known as Hylai.[26]

The great antiquity of the sanctuary, established by the "Letter to Gadatas", supports the theory that the name was derived from an indigenous Carian toponym. The site of the cult and its members enjoyed special privileges that were seemingly preserved until late antiquity, as indicated by the extracts from the Life of Isidore. The pre-Hellenic roots of the cult, perhaps, account for the eccentric behavior of the *dendrophoroi* and their deviation from the normal settings and practices of a sanctuary dedicated to Apollo, in which the priests did not practice divination, but instead engaged in rites more commonly associated with Cybele or Dionysos. Because the city was surrounded by defensive walls since Hellenistic times, the *dendrophoroi*

[26] Rather than Leukophrys, as conjectured by Phillipson (1936). See note 11 above.

would have to enter the city via one of the gates, thus connecting their ritual with the city layout and street grid. The specific nature of this alignment will be closely examined in Chapter 5.2, which presents of the results of investigations that use 3D models to reconstruct the ways in which the cave of Apollo might be traced in the urban scheme of Magnesia through its possible connection with a group of sanctuaries dedicated to Dionysos.

3.2 Cult of Dionysos

Like Apollo at Hylai, we find Dionysos behaving rather out-of-character in Magnesia. Although not explicitly connected with Apollo Aulaites in the epigraphical record, the cult of Dionysos occupies an intriguingly complementary position when considered in relationship to the urban topography. An inscription found in Magnesia in 1890 records a singular event that would have occurred in the early 3rd century BC (Kern, 1900, no.215):[27] One day a large plane tree standing in the city center was torn apart by a violent wind, revealing an image of the young Dionysus. Upon consulting the oracle of Apollo at Delphi as to the meaning of this occurrence, the Magnesians were told to bring back with them three Theban maenads who would instruct them in the ways of Dionysus and his rites, for they had forgotten the god, who was already present when they built their city.

This inscription comprises two marble pieces: a larger stele that recounts the oracle, and a smaller square piece containing a dedicatory inscription. The two pieces were not found together, being scattered in various parts of the village of Tekke, but Humann and Kern assert that they have it on good evidence that both pieces were originally found at a site just to the west of the City Gymnasium "at a point on the field where the debris clearly indicates that an antique

[27] See also Reinach (1890) and Kern (1895). These discussions are updated by Henrichs (1978) and Graf (2004).

building formerly stood there" (Kern, 1895, p.85).[28] This location, now completely covered, is also the site where in Humann and Kern's time another inscription could still be observed in situ, and which they called the "Mysteninschrift" (Kern, 1900, no.117). This inscription testifies to the practice of Dionysian mysteries and records the sums of money donated by worshipers. On the basis of these three inscriptions, Kern and Humann posit the existence of a temple of Dionysos at this location, a few steps west of the City Gymnasium. This location, in the heart of the city, would accord with the account of inscription no.215, which describes the plane tree where the epiphany of Dionysos happened as πλατάνου κατὰ τὴν πόλιν.

The inscription contains a prose postscript (Kern, 1900, no.215a) which reads:

> "In accordance with the oracle, and through the agency of the envoys, three maenads were brought from Thebes: Kosko, Baubo, and Thettale. And Kosko organized the thiasus named after the plane tree, Baubo the thiasus outside the city, and Thettale the thiasus named after Kataibates. After their death they were buried by the Magnesians, and Kosko lies buried in the area called Hillock of Kosko, Baubo in the area called Tabarnis, and Thettale near the theater."[29]

The genuineness and dating of the inscription is crucial if we are to take it as a guide to the actual topography of Magnesia. Though controversial, the oracular origin of the Dionysian cult is attested by multiple chronological details, and thus the inscription provides valuable information on the urban layout of Magnesia. Henrichs (1978, p.130) provides a convincing analysis of the text and concludes that "we may now proceed on the assumption that the city of Magnesia, at the urging of the Delphic oracle, actually imported three maenads from Thebes sometime between 278 and c.250 BC, and that these maenads died in Magnesia and were buried there at public expense, probably before 207/06 B.C."

[28] See Fig.2:2
[29] Translated by Henrichs (1978).

The inscription names the three thiasoi (groups of devotees) led by the Theban maenads, along with their burial places, but finding the places they refer to is no simple matter.[30] The maenad called Kosko led a thiasos called *Platanistinoi*, which seems to be a reference to the plane tree where the epiphany of Dionysos occurred. If, following Humann and Kern, we accept that the supposed Temple of Dionysos was constructed on the site where the plane tree originally stood, just to the west of the Gymnasium, this may be the location of Kosko's thiasos. She is described as being buried at *Koskobounos*, which suggests it was on a hill. This toponym raises the question of whether the maenads' burial places are to be sought near the location of their respective thiasoi, since the terrain around the putative temple of Dionysos is relatively flat. On the other hand, the plane trees invoked by the name of Kosko's thiasos need not have been on the site of the epiphany, but may have simply referred to a grove of plane trees elsewhere where the group of devotees gathered to pay homage to the original appearance of Dionysos in Magnesia.

Furthermore, the name of the second thiasos places it in contradistinction to the city and therefore may indicate that the other thiasoi were indeed located, by contrast, within the urban environment. The second maenad, Baubo, led a *thiasos pro poleos* (outside the city walls) and was buried at a place called Tabarnis. Henrichs (1978, 134) suggests that Baubo's thiasos might have been the most likely of the three to host actual maenadic rituals, since it was located at a proper distance from the city. Dionysiac rites were traditionally associated with natural scenery such as woods and hills. The site associated with the toponym Tabarnis remains unlocated, although Kern assumes that Tabarnis must be "a place outside the city" (Kern, 1895). In fact,

[30] A distinction should be made between the names of the thiasoi, which describe groups of people and the names of the maenads' burial sites, which refer to places. However, if the maenads were buried in separate, significant locations it seems likely they may have practiced in different neighborhoods as well. The names of their thiasoi may well be clues, as the spatial designation *pro poleos*, for example, appears to indicate.

Tabarnis was mentioned in another Magnesian inscription, which describes it as being the location of a spring from which water was diverted to the city (Kern 1900, no. 251). This suggests at least two possible locations: an aqueduct descending from the hills met the city walls in the south-west (Humann, 1904, p.29). This interpretation would seem to indicate that Tabarnis was a village within Magnesia territory, situated in the foothills of Mt.Thorax. Alternatively, Humann's plan locates a spring above the Theatron (Fig.2:12), directly in line with a pump room and fountain in the southwest corner of the Agora that was excavated by the German team. This spring is within the city walls, but would have been outside the city grid proper, as the streets and residential blocks probably did not extend into the foothills (Bingöl 2007, p.128). On the other hand, Reinach (1890) argues that the words *pro poleos* (which he interprets as "before the city") must be taken to indicate a location of a sanctuary at the gates of the city, while Henrichs (1978, p.130) deems it "very likely" that Tabarnis echoes the Latin *taberna*, suggesting an Imperial, rather than Hellenistic, date for the toponym.[31] According to Henrichs, this does not rule out the actual existence of such a place, as the author of the inscriptions, Apollonios Mokolles, may have simply been describing the location of the ancient maenadic tombs using the name he was familiar with from his own era. If Tabarnis is indeed derived from *taberna*, this points to a location within the city where shops would be found, rather than outside of it. However, the combined evidence of the inscription that identifies Tabarnis with a spring that supplied the city with water, together with the designation of Baubo's thiasos *pro poleos*, seems to indicate that Tabarnis would most likely be found in the southern foothills above the city.

[31] Thonemann (2011, p.257) however, considers Tabarnis to be an "indigenous, or at least very ancient Greek" name.

The etymology of the toponym describing the location of the thiasos of the third maenad Thettale, *kataivatai*, is as suggestive as it is controversial. The early commentators were drawn to make the connection with *katabasis* and descent to the underworld: Reinach said that "the thiasos of *katavatai* awakens the idea of descent and particularly of the descent into hell". Reinach evokes Strabo's mention of several charonion, or caves which functioned as portals to the underworld, in the vicinity of the Maeander valley (Strabo 12.8.17). After pointing out that the name of the Lethaus River evokes Lethe, one of the five rivers of Hades, he continues: "perhaps, however, *katavatai* means simply an area of Magnesia which sloped steeply towards the river Lethaios" (Reinach 1890, p.360). Kern (1895, p.93) also argued for a meaning of 'descent', citing the *thasis ton Kataivaton,* an underground crypt where the Eleusinian mysteries were held. However, instead of transposing this on the sloping banks of Lethaios, he prefers the steep mountainsides of Thorax. Again he makes the association with a cave, but this time rather than the *charonion* of Strabo he mentions the fabled cave of Apollo at Hylai, and reiterates this connection in the catalog of inscriptions published in 1900 (p.139, note to line 36). Albert Henrichs (1978, p.133) instead takes *kataivatai* as a reference to Zeus Kataibates, or Zeus who descends in lightning: "they would have met at a location where the lightning of Zeus had struck, and was therefore taboo except for religious use".[32] He also makes the interesting point that *kataivatai* is a masculine form; suggesting that Thettale's thiasos may have included men: an unorthodox configuration. It is tempting to speculate that the men in this third thiasos were the *dendrophoroi* of Apollo, and that *katavatai* is indeed a reference to descent into a cave, or alternatively, from the heights of Aulai.

[32] See also Maass (1891,186)

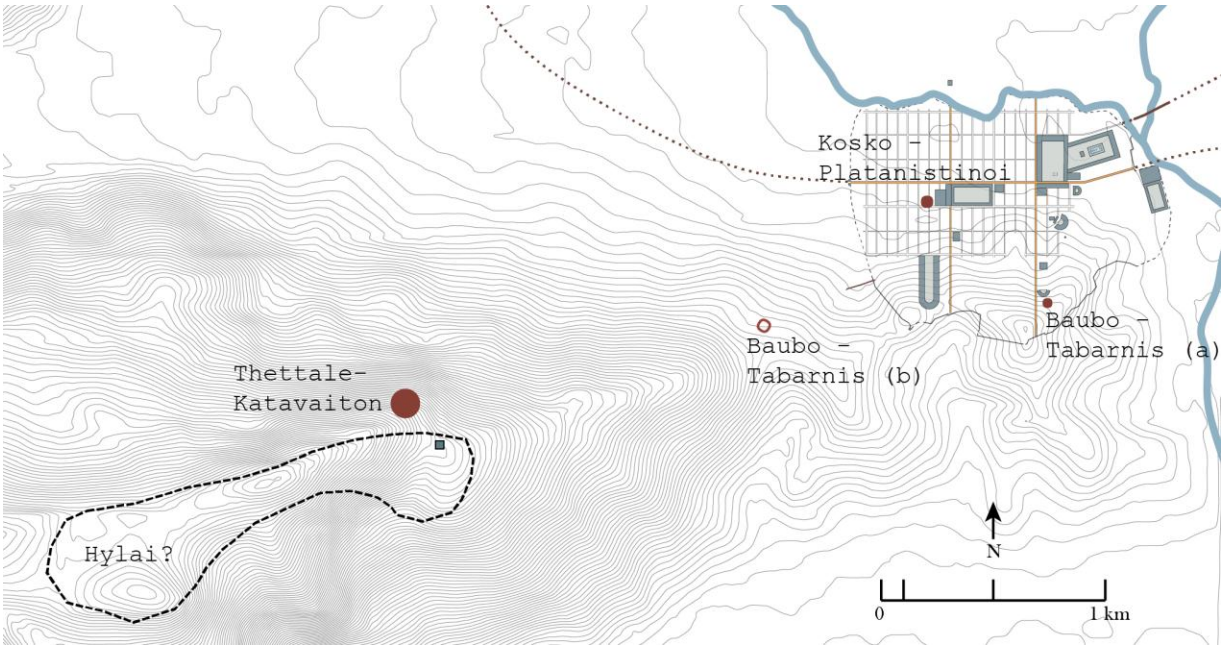

Fig. 14 Possible locations of the three thiasoi

3.3 Connection between Apollo of the Cave and Dionysos in the City

By the late 19th century, Reinach had first noticed the connection between the Dionysian thiasos of *katavaiton* and the idea of "descent" (as into a cave), and Kern had explicitly linked this to the cave of Apollo at Hylai, The name *dendrophoros* had been used in numismatic catalogs to identify the men carrying trees pictures in numerous Magnesian coins (Schulz, 1975, p.39). However, this appellation was understood as a reference to the followers of Cybele, and not Apollo, since the cult of Cybele was known to use tree branches in their ceremonies. A direct link between Apollo and Dionysos at Magnesia, however, remained elusive. In 1895, numismatist Imhoof-Blumer published a Magnesian coin of the mid-2nd c. AD which shows a child Dionysos (a type of representation of the god frequently encountered at Magnesia) seated in a shrine between two columns, and figure dancing before him, carrying a tree (Schulz, 1975, p.76 no.189). The two columns seem to indicate the existence of a temple of Dionysos in Magnesia which housed the image of the god, just as Humann had suggested. Imhoof-Blumer

identified the dancing figure as the *dendrophoros*, citing the passage in Pausanias that describes the men who "uprooting trees of exceeding height walk with their burdens down the narrowest of paths", and links it with the inscription of Dionysos' appearance in the plane tree. He proposed that "the *dendrophoroi* who bring their burden to the sanctuary of Dionysos indicate a ritual in honor of Dionysos Dendrites, the god of vegetation" (Imhoof-Blumer, 1895, pp.285-286).

In 1977, a comprehensive study of the connection between Apollo in the cave and Dionysos in the city was offered by Louis Robert. His review of the numismatic evidence for representations of Apollo and Dionysos in Magnesia (Robert 1977, p.84) led him to conclude:

> It seems reasonable to me to assume that those possessed by Apollo, drawing their extraordinary powers from the statue of Apollo at Aulai, doubtless began their exploits of running through the mountains and uprooting of trees at the sacred wood of the god, and wandered uninterrupted with their charge through the mountains and ravines of the surrounding area (including Mount Thorax) to arrive in the city at the sanctuary of Dionysos and deposit before the god the uprooted tree. It was necessary that this tree, uprooted at Apollo's cave in the forests of Aulai, arrived somewhere. It cannot be brought back to the sanctuary from whence it had been removed, since the "obstacle course" exhausted those who had brought it far away. One cannot abandon it in a corner.

He notes that the cults of both Dionysos and Apollo in Magnesia are united by their connection with trees: the plane tree in which the image of Dionysos was found, and the trees carried by the followers of Apollo. Since the cave at Aulai predated Dionysos' epiphany, perhaps the ritual of the tree-carrying *dendrophoroi* was a response to this event, a re-enactment of the appearance of Dionysos in seasonal celebrations.

Certainly the specific connection between Aulai and Dionysos' temple can only be as old as the temple itself. Before that, we must still account for the cave and its strange gardeners. Though not unprecedented, the cave is an unusual home for Apollo, who is more commonly

encountered as the god of architecture and cities.[33] Apollo Pythios played a role in the founding of Magnesia through his oracle at Delphi. In fact, the dynamic between Apollo and Dionysos at Magnesia flouts most of the modern stereotypical characteristics of both gods, although these stereotypes (such as that Apollo represents reason, order, intellect, etc.) may be mostly modern inventions. Marcel Detienne (1986; 2002) sees the persistent Apollonian-Dionysian duality in modern thought as the result of the influence of Friedrich Nietzsche's *Birth of Tragedy*. This work, Detienne argues, eclipsed the complex relationship that existed between Apollo and Dionysos in many local variations throughout the Greek world (Detienne 2002, p.148). He proposes instead that the Apollonian-Dionysian dynamic be treated as an example of a polytheistic system "primarily constituted by the *relations* between gods", characterized by "practices and their variety…concrete configurations…, the acts, objects and situations that contextualize the relations between divinities". Such a configuration may be sought in the spatial and architectural (or natural) context in which the ritual of the *dendrophoroi* was played out at Magnesia. It is by teasing out these concrete, empirical localities, architectonic settings, and movement between them, that we may begin to understand the relationship of Apollo and Dionysos as it existed at a specific place and time.

In view of this, the fact that Apollo is to be found in a cave at some distance from the city, is worth examining more closely. One strain of Apollo's character is thought to have its origins as an indigenous god of Asia Minor, who originated in Lycia and was Hellenized by around 800 BC (Weinberg 1986, p.134). As such he originally was much more 'unstable' and chthonic in character; only after the progressive taming of his nature towards the qualities more

[33] According to Callimachus (Hymn to Apollo, verses 55-64) Apollo "delights in the founding of cities" and was therefore called *Archegetes* (leader and protector of colonies).

familiar from the Nietzchiean perspective does he seem to represent the diagrammatic opposite of Dionysos (Weinberg, 1986, p.305). In fact, the distinction between Apollo and Dionysos is often blurred or even nonexistent, particularly in the sense in which both gods functioned as "gods of inspiration"; whether towards prophecy or frenzy or a combination or the two. At Delphi, the center of the Greek world (and a place which held a particular importance for the Magnesians, who frequently consulted the oracle there), the site is shared by Apollo and Dionysos. There is evidence that before Apollo and his cult arrived at Delphi, the place was sacred to Dionysos (Weinberg 1986, p.304); even after Apollo's accession, Dionysos remained the sole occupant for the three months of the year that Apollo was said to be visiting the Hyperboreans in the north.[34] Significantly, the Delphic oracle may have drawn its power from a cave, although ancient accounts and modern archaeology differ as to whether this cave actually existed and what function it served in the sanctuary.[35] Strabo (9.3.5), although apparently without having witnessed it himself, recounts: "They say that the prophetic chamber is a cave, hollow in depth, with a rather narrow mouth, from which arises the breath of inspiration." However, modern archaeologists have not found evidence for a geological formation corresponding to this description. Whatever the form of the cave, the existence of this configuration at Delphi, a place of manifest importance for the Magnesians, may be the template for the recurrence of the Apollo – Dionysos – cave triad in Ionia.

3.4 Chthonic and Underground Cults at Magnesia

Spatially, the cave is as elusive at Magnesia as its idea is evocative. Still, the prevalence of underground spaces associated with caves or cave-like functions in Greece allows us to entertain

[34] On origins and sources for the story of Apollo and the Hyperboreans, see Fontenrose (1959,pp.382 – 383; note 25 p.382)
[35] Ustinova (2009, 121-155) gives an extended account of the controversy surrounding the Delphic adyton.

the possibility that some resonance of the cave of Apollo at Hylai might have found its way into the architecture of Magnesia. Caves dedicated to Apollo were relatively rare, though not without precedent. Delphi is the most famous example, and in Asia Minor the temple of Apollo at Hierapolis and the temple of Apollo at Claros both contain underground spaces beneath the cella where prophecy was performed. The capacity of caves to cause the phenomenon of *enthousiasmos* – inspiration through possession by the gods -- was well-established in antiquity. Strabo (14.1.1, 12.8.17) refers to three such caves in the Maeander Valley: one of which, the Aornum, was at a place called Thymbria halfway between Magnesia and Myus. This cave, along with that of Acharaca outside Nysa, and the cave under the temple of Apollo at Hierapolis, was one of a type that seems to have been filled with noxious vapors that produced mild hallucinations in those who entered. All three of these caves were considered Charonia, or gates to the underworld, and indeed the cave at Acharaca was associated with the temple of Pluto and Kore. However, the consciousness-altering properties of each cave served a different function: at Acharaca the cave had healing properties, while at Hierapolis the vapors may have served an oracular purpose (Ustinova 2002, 284; Ustinova, 2009, 273). At Aezani in Phrygia, the cave sanctuary was associated with an indigenous mother goddess who was Hellenized as Cybele.

Turkish archaeologists have recently discovered that the east stoa of Magnesia's agora, which contains the propylon to the Artemision sanctuary, originally possessed two levels, the lower one being an underground cryptoporticus (Bingöl 2007, pp.105-109). The agora is heavily impacted by the high ground water that regularly floods the area, and it is hard to imagine why this underground structure would have been built if not for some important ritual purpose. This conjecture is supported by the discovery of frescoes on the cryptoporticus wall depicting a female figure resembling Artemis standing in a chariot (*ibid*.). Cryptoportici are known to have

had cave-like functions in some cases, notably for incubation and healing as at the Asklepion at Pergamon. A similar case is found in the "den" of a group of priest-healers known as *pholarchs* at Elea (Ustinova 2009, pp. 191-209; Rickert, 2014, pp.477-479).[36] This cult, of which the philosopher and poet Parmenides was a member, was dedicated to Apollo.[37] In an interesting parallel to the activities that may have occurred at Hylai, the priests of Apollo at Elea (called Hyele in Herodotus) went into incubation not to heal themselves, but to receive divine wisdom. If these two examples are any clue, we may imagine that the cryptoporticus at Magnesia functioned as a place of restricted access where priests or other functionaries withdrew from the crowds of the Agora into a dark, silent cave-like space where they could prepare themselves mentally to officiate in rituals.

Nearby in the agora, Zeus was worshiped in his chthonic guise 'Sosipolis' and honored with the sacrifice of a bull as befits an underworld deity (Kern 1900, no.98).[38] The orientation of the temple of Zeus Sosipolis, which faces west, may be evidence of the architectural emphasis of his chthonic nature (Bingöl 2007, 112). Zeus is one of two prominent deities at Magnesia who were said to have been born, or raised, in a cave.[39] The other is Dionysos, whose characteristic representation on Roman coins of Magnesia as a child seated on a mystical *cista mystica*, a basket containing a serpent used in ritual, also connotes the underworld (Robert, 1997, Fig.10,

[36] The term *pholeos* was also used by Strabo in his description of the cave of Archaraca near Nysa in Caria, which also had healing properties (Strabo, 14.1.44; Ustinova, 2009, p.198). The connection between the two practices is supported by the fact that Elea was a colony of the Phocaeans, who fled Caria in the Persian era (Herodotus 1.167).
[37] Parmenides' famous poem, "On Being", essentially enacts the reverse of Plato's allegory, in that it describes a *katabasis*, or descent to wisdom and truth (Rickert, 2014).
[38] Zeus Sosipolis was considered a chthonic god (Long, 1987, pp.248-9). The procession of youths bearing the sacrificial bull is reminiscent of the procession from to the Plutonium at Acharaca (Strabo 14.1.44)
[39] Zeus was hidden in a cave under Mt.Ida on Crete as a baby, to protect him from his father Cronos, who believed he was destined to be killed by his son. Caves sacred to Zeus were common Greece (Weinberg 1986, pp.115-118). It is toward the cave of Zeus on Crete that the group of philosophers is walking as they describe the ideal city of Magnesia in Plato's *Laws.*

p.67; Schulz, 1975, p.38).[40] It is suggestive of a particular attunement to the functions of the cave that both of these gods were represented at Magnesia in a guise which emphasized these chthonic aspects of their nature. The origins of Zeus and Dionysos recall the cave as a nurturing place, which provided protection and seclusion. The presence of the cryptoporticus in the Agora may evoke the healing aspects of the cave. These benevolent properties are balanced by the cave's function as a gate to the underworld, as evidenced by the cycle of death and rebirth symbolized in the sacrifice of the bull to Zeus Sosipolis.

Today, the underworld still has a presence at Magnesia, albeit a less allusive one. Bingöl (2007, p.163, p.178) recounts that underground passages persist in the imagination of the local population, although this is certainly due to the fact that half-buried structural vaults have the air of secret tunnels. One of these is the vaulted substructure of the Roman temple north of the Lethaios, which according to local folklore is said to be the start of an underground passage leading to the gymnasium. This explanation is probably more colorful than is warranted. The dimensions of the vaulted space correspond closely to the space supporting the podium of the temple of Augustus at Antioch in Pisidia. Mitchell and Waelkens (1998, pp.119-120, p.158) convincingly demonstrate that the vault under the temple at Pisidian Antioch, like many other temples dedicated to the imperial cult,[41] was a byproduct of the desire for an elevated podium, that perhaps served as a treasury or other cellar, instead of a cult room intended to emulate a cave, as is the case with the Temple of Zeus at Aezani.

[40] The child Dionysos was raised by Nymphs in a cave in the idyllic peninsula of Nysa (Weinberg 1986, pp.120-124).

[41] Cf. the Maison Carrée at Nimes or the temple of Rome and Augustus at Caesarea Maritima (Amy, R. and Gros, P. (1979). *La Maison Carrée de Nîmes*. Éditions du Centre national de la recherche scientifique).

3.5 Summary

The cults dedicated to Apollo at Hylai and Dionysos in the city interacted with the urban fabric of Magnesia on the Maeander through the dynamic of movement. This movement passed from the mountain slopes of Thorax, a sacred natural space which long functioned as a topographic anchor in the memory of Magnesians, through the city walls and, the processional route taking its shape from the planned grid of the streets, interacted with the architectural space of the city. The *dendrophoroi*, in bearing their fully-grown trees, intermingled the natural space of which their cave was the spatial apex, with the rationally-planned built environment of the Greek polis. In order to more fully understand how this movement and ritual functioned as "operative polytheism" (to paraphrase Detienne), I elaborate this performative dynamic within an architectural reading of the city. Specifically, an experimental method applied through the use of 3D models will shed light on how the architecture and planning of Magnesia responded to this ritualistic context. The following chapter details the technical and epistemological implications of procedural modeling, the methodology undertaken in this endeavor.

Chapter 4: Methodology: Procedural Modeling for Holistic Urban Reconstruction

In order to read the disparate evidence for Apollo and Dionysos in a spatial context, it is necessary to expand the analytical toolset. Scholars of the ancient world have long been preoccupied with the quest to interpret the many-layered picture that results from combining evidence from ancient texts, empirical archaeological data, and geographical landscape surveys. Today, the ever-increasing amount of data available to researchers demands that methodologies adapt as well. The use of 3D digital technology aids the task of archaeological reconstruction in many ways. Crucially, digital tools provide a means of aligning and comparing discrete data sets which juxtapose visual material alongside geographic, textual, architectural, and quantitative information. Furthermore, the methodology I present here, procedural modeling, allows for the documentation of the decision making process and use of source data in the making of the Magnesia model. This aids in making clear when known or interpolated factors were used in the modeling of hypothetical scenarios, such as Magnesia's conjectural urban plan. Finally, the resulting three-dimensional models allowed for the holistic analysis of the city as a complex phenomenon involving spatial, material, and cultural determinants.

The reconstruction of Magnesia used in this study makes use of a suite of procedural rules which generate 3D models of Hellenistic and Roman architecture and urban environments from a variety of periods and contexts. The term 'rules' in procedural modeling refers to the computer code that generates a 3D model. Unlike traditional 3D modeling software, in which users directly manipulate polygons to simulate form, procedural modeling entails the use of computer scripting languages in the textual semantic description of a building that then generates a polygonal

model. This represents not only a technical, but also an epistemological difference, as the choice of modeling method can influence not merely the cost or aesthetic outcome of a project, but also how information is selected, processed, and indeed what is considered to be information instead of noise. Procedural modeling provides a framework for each stage of the transmutation of data in the modeling process to be rigorously thought out and documented, allowing 3D models to move beyond visualization to become robust research tools.

4.1 Previous Work in 3D Architectural Reconstruction

3D architectural reconstructions may be created in the service of a wide range of research agendas. The technique used to create a model should therefore be matched with the motivation for making it. While I argue that it represents a paradigm shift in 3D modeling, procedural modeling is certainly not the only valid approach to creating an architectural reconstruction model. Other, more widely used methods may sometimes be quicker, easier, and more appropriate to the task at hand. In view of this caveat, some discussion of other modeling techniques is necessary in order to make clear when procedural modeling provides a distinct advantage and when it does not.

4.1.1 Non-Procedural Modeling Techniques

'Traditional' modeling software is based on either polygon mesh or NURBS modeling.[42] Polygon mesh modeling is probably the most common form of 3D software and is represented by such popular software such as 3ds Max and SketchUp.[43] Polygon modeling derives 3D form from primitive geometric forms which are scaled, rotated, and transformed as necessary (Foley et

[42] NURBS stands for "non-uniform rational B-spline"
[43] 3ds Max is an Autodesk product, the industry standard for rendering and animation: http://www.autodesk.com/products/3ds-max/overview. SketchUp, formerly a Google product, is now produced by Trimble, and is considered a "user-friendly" 3D modeling package: http://www.sketchup.com/.

al., 1993). In this it is similar to procedural modeling, except that the polygon mesh modeler manipulates the objects directly in a visual interface, pointing and clicking to modify geometry. This type of GUI is intuitive and fast, and easily learned. However, unlike procedural modeling, it does not require that the modeler "spell out" in textual form the decisions which are taking place, so the record of the modeling process, along with the opportunity to attach scholarly evidence to the interpretative model, is more likely to be lost, unless the modeler takes care to document their choices.

NURBS modeling, a feature of software packages like Maya and Rhino[44], is similar to polygon modeling in this way. However, NURBS modeling uses flexible splines rather than polygons for the creation of geometry, which allows for the realistic rendering of organic forms and curved surfaces (Piegl, 1991; Rogers, 2000). Analogous to sculpture, NURBS modeling is even more intuitive than polygon modeling, and therefore also carries the risk, when used for research models, of some scholarly rigor being lost in the process. However, it does some things well that procedural models do extremely poorly, namely the representation of curved and organic forms. NURBS modeling software Rhino and Maya have increasingly incorporated 'parametric' features into their packages. The terms 'parametric' and 'procedural' are sometimes used interchangeably, but in practice they represent quite different concepts. Generally speaking, 'parametric' signifies any technique which operates through the use of parameters (Monedero 2000). But 'parametricism' has taken on a specific meaning in the context of 3D modeling for architectural design (Burry 2003).[45] Within the realm of architectural reconstruction, parametric

[44] Autodesk Maya is a powerful but complex software which is good at simulating physical dynamics, such as liquids, fire, and air: http://www.autodesk.com/products/maya/overview. McNeel Rhinoceros is a simpler NURBS modeler optimized for the jewelry and industrial design fields, though also used widely in architecture: https://www.rhino3d.com/.

[45] With a capital 'P', 'Parametricism' most likely refers to the controversial manifesto by Patrik Schumacher (2008).

building components can be scripted in Maya, as they were for a 3D model of the Suleymaniye Mosque in Istanbul (Chevrier et al, 2009). 'Procedural' modeling, on the other hand, connotes the use of parameters within a rule-based modeling approach that goes beyond a means of manipulating 3D data, to imply a logic of form. The term 'procedural' itself reflects a focus on the process or recipe for a given set of outcomes.

Parametric modeling invites comparison with another 3D technique widely used in the architectural world: Building Information Modeling (BIM).[46] Procedural modeling and BIM modeling are alike with regard to the emphasis on parametric management of data, yet they differ in significant ways. Like procedural and parametric models, BIM modeling software such as ArchiCAD, Revit, and Vectorworks[47] create models via attributes and parameters rather than the visual manipulation of points, as in polygon mesh or NURBS modeling. However, BIM models make use of industry-specific libraries of parametric components (windows, doors, columns, slabs)[48] and are intended primarily as a means for streamlining communication between architects and their contractors and aiding the production of construction documents and cost estimates. Because they are capable of containing a great deal of data with the model, BIM models have begun to be adopted in the cultural heritage field, for documenting vernacular architecture (Fai et al., 2013) or the preservation or refurbishment of historical buildings (Del Giudice et al., 2013). However, the contemporary orientation of their parametric libraries renders

[46] A useful summary of BIM can be found at http://www.graphisoft.com/archicad/open_bim/about_bim/ and http://www.autodesk.com/solutions/building-information-modeling/overview. See also Azhar (2011).
[47] Graphisoft ArchiCAD: http://www.graphisoft.com/archicad/; Autodesk Revit: http://www.autodesk.com/products/revit-family/overview; Nemetschek Vectorworks: http://www.vectorworks.net/.
[48] Often off-the-shelf products ready for purchase.

BIM software packages rather limited when it comes to reconstructing architecture from fragmentary archaeological remains or images (Boeykins et al., 2012).

Another method that is becoming common in 3D archaeological documentation and cultural heritage is photogrammetry, which can reproduce any historical element with photorealistic accuracy.[49] Using various processes, an individual object such as a work of sculpture, a column, or even an entire building may be captured using laser scanners, photographs, or structure from motion (SfM) in order to achieve an extremely point-dense, accurately photo-textured model (Böhler et al., 2004; Kadobayashi et al., 2004). The software which create these models come in both open-source and commercial varieties, and include popular packages such as Photoscan and 123D Catch[50]. Likewise the equipment they require can range from a simple phone camera to an expensive laser scanner. Photogrammetry is well-suited for the documentation of artifacts, as it can be used on-site as the basis for extremely accurate measurements and line drawings. For reconstruction models which rely on scant archaeological remains, however, the method is less useful.

4.1.2 Background of Procedural Modeling

In such cases, empirical evidence must be supplemented with architectural knowledge. Uniquely among computer modeling techniques, procedural modeling taps into a lineage in architectural theory dating back to antiquity. This line of thought, which was articulated by Vitruvius in the first century BC and later taken up variously in the 16th, 18th, and 20th centuries ,[51] seeks to

[49] Many examples of projects that incorporate these techniques can be found in the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences: http://www.isprs.org/publications/archives.aspx.
[50] Agisoft Photoscan: http://www.agisoft.com/; Autodesk 123D Catch (free app) http://www.123dapp.com/catch.
[51] An early proponent of procedural methods for architectural design was Rodrigo Gil de Hoñatón (c.1500 -1577), who devised structural and proportional rules for the design of Gothic churches. In the eighteenth century, Enlightenment theorists Marc-Antoine Laugier and Quatremère de Quincy posited an origin for architecture in the

elucidate and systematize an underlying logic of architecture. The fundamental concept of this view could be summed up by what Plato called 'that of dividing things again by classes, where the natural joints are, and not trying to break any part, after the manner of a bad carver' (Plato, *Phaedrus*, 265e). The reading of distinctions between parts, and the syntax of their joints, naturally lead to an association between the logic of architectural systems and linguistic grammars. Indeed, procedural modeling can sometimes seem like a game in which large masses must be broken up into ever smaller ones until the full resolution is reached, taking care not to carve in a manner that will break any of the parts.

Procedural modeling has its roots in computer graphics techniques, such as shape grammars and Lindenmayer systems or L-systems (Prusinkiewicz and Lindenmayer, 1996), which aim to describe things efficiently in order to visualize them accurately. An efficient description operates with the minimum number of general rules that result in the widest range of variable and valid outputs. The conceptual core of procedural modeling in architecture is the work on shape grammars by Stiny and Gips (1972), who were interested in developing a computational basis for design. The computational bias inherent in procedural logic was later taken up by computer scientists looking for new techniques for graphics rendering. Similarly to L-systems, which are mathematical models for creating organic self-similar forms, procedural grammars offered an efficient way to generate multiple differentiated objects with a minimal number of rules. The procedural grammar used in this project, CGA Shape Grammars, was

classical vocabulary of ancient Greece, precisely because of the (procedural) linguistic analogy that underpinned its system of interrelated parts, and because they saw language as a prerequisite to culture and civilization. In the twentieth century, architects saw the similarities between procedural logic and computational processes as potential for new design methods (see work by Stiny, Gips, and Mitchell, discussed below).

developed at ETH Zurich Computer Vision Laboratory and commercialized as CityEngine.[52] Most current work on procedural modeling occurs within the field of computer graphics (Schinko et al., 2015), with applications in the urban planning, gaming, and entertainment industries. In recent years, archaeology and cultural heritage projects, such as the significant test cases built around ancient Rome and Pompeii,[53] have also begun to explore the use of procedural modeling for the reconstruction of ancient sites (Haegler, et al., 2009).

Classical architecture and its rigorous system of orders and proportions is, indeed, the architectural style most commonly cited by proponents of the mathematical logic of architecture. The adaptability of classical architecture to grammatical description has been exploited since antiquity when the 'rules' of its orders were codified by Vitruvius in his *Ten Books on Architecture*, even if such systematic regularity rarely occurs unadulterated in practice.[54] A treatise by William J. Mitchell (1990) devoted to the subject of the logic of architecture relied heavily on classical and neoclassical examples to formulate a system for design. Many Greek and Roman cities, moreover, were built around a template that arranged a core set of civic buildings on a grid-based infrastructure.[55] This modular, systematic approach to city planning

[52] A precedent for my current project is Pascal Mueller's 2010 PhD dissertation which used classical temples as a case study for demonstrating the potential of CGA shape grammar, the procedural language that eventually became the core of ESRI CityEngine. Mueller's (unpublished) dissertation and Parthenon rule, which is distributed as an example with CityEngine software, were indispensable in my efforts to master CGA shape grammar. Mueller, as one of the principal authors of the CGA shape grammar language, was a co-founder of ETH spin-off company Procedural, which first released CityEngine. However, his work was oriented to the field of computer science and his study of temples focused on peripteral temples of the Doric order. The rules I present here are my own work, as a full restructuring and rewriting of the code, with the addition of much new material, was necessary to implement a wider agenda geared toward a humanities audience. For an overview of the architectural application of CGA shape grammar, see Mueller et al., 2006.

[53] See the Rome Reborn Project, http://romereborn.frischerconsulting.com/. The procedural aspects of this project were published in Dylla et al., 2010. On Procedural Pompeii, see http://www.esri.com/software/cityengine/resources/casestudies/procedural-pompeii.

[54] See Wilson Jones (2009, *passim*) on the problems of applying Vitruvian theory in the field.

[55] Though much research has been done on this topic, I will mention here two important works: On Greek grid-planned cities and housing, see Hoepfner and Schwandner (1986); On Roman architectural urbanism, see MacDonald (1988).

resonates with the computational mindset of the digital age. In the design world, contemporary architect Rem Koolhaas ran a studio at Harvard investigating what he called the "Roman Operating System" (Koolhaas et al., 2000), and meanwhile the Roman City has been readily adopted by simulation games such as Minecraft and CivCity Rome.

4.1.3 Choosing a 3D modeling technique

The purpose of this section has been to show where procedural modeling fits within the context of other 3D modeling methods for architectural reconstruction, all of which are valid approaches with strengths in different areas.[56] I chose to use procedural modeling for the modeling of Magnesia for several reasons: First, I wanted to explore the potential of script-based models to preserve decision-making processes and metadata. The 'rules' that create the models, therefore, were of more service to my research goals than the virtual reality aspect of modeling, and therefore the level of abstraction inherent in the procedural technique was an acceptable cost. The steep learning curve and time investment of learning the scripting language was mitigated by the longer-term goal of my work: to accumulate a robust library of procedural rules that can be expanded and amended by others to facilitate future research.[57] Most importantly, my research questions called for a method that would allow the iterative generation of many different models while preserving the decision structure driving the process. The first question, regarding the city plan, relied in large part on the malleability of the interactive street grid in the procedural model to allow the testing of many hypothetical reconstructions. Second, the analysis of the interface of

[56] Most 3D projects, including my own, involve a combination of modeling softwares and approaches. In the case of my project, it was unrealistic to procedurally model sculptural architectural elements (such as a Corinthian column capital), and so these were modeled in other software and imported as assets. The procedural rule controls which asset is selected and where it is placed (for example, to suit attributes such as level of detail or column order).

[57] The procedural rules described here were developed by the author over a period of five years. It should be noted that my approach pushes the limits of the CGA shape grammar and therefore required a period of time to master the scripting language; this may be prohibitive for many humanists. It is, however, possible to use procedural modeling in a way which makes use of simpler rules but derives many of the same benefits of GIS-based 3D analysis.

ritual and terrain was greatly aided by the ability to quickly represent conjectural models for buildings that had great impact on the urban environment but whose spatial presence had not been considered fully due to limited evidence. Finally, though it is not included in the procedural vocabulary, the very resistance of the cave to this type of modeling underscored its significant absence from the built form of Magnesia. This insight ties into my broader argument about the way in which the cave, exemplified by the home of Apollo at Hylai, resisted architecturalization in the Greek world.

4.1.4 Criteria for Analysis

Of course, for the purposes of scholarship, 3D models are only as good as the value they add to research. The usefulness of 3D models for humanistic research has been thoroughly considered elsewhere (Favro, 2006; Frischer and Dakouri-Hild, 2008; Johanson, 2009),[58] therefore I will simply reiterate the basic criteria against which all 3D methodologies must be judged:

1. The model provides insights that would not likely have arisen without it.
2. The model provides demonstrable evidence either in support or in refutation of a hypothesis.
3. The model effectively connects data of different types in a way which makes it easier to interrogate and analyze the data.
4. The process of making the model has aided the understanding of the material.

If one or more of the criteria is fulfilled, the model has made a contribution to the research. This is in distinction to models which merely illustrate the arguments which were arrived at

[58] See also the principles for cultural heritage visualization set out in the London Charter (2009): http://www.londoncharter.org/.

independently of the modeling process. I evaluate these criteria further with regard to my research on Magnesia in Chapter 5.4 below.

4.2 Creating the Roman City Ruleset

The impetus for creating the 'Roman City Ruleset',[59] a library of procedural rules that generate the essential building typologies for modeling Greek and Roman cities, emerged gradually from a series of projects. Each of these investigates a different research question, but all require a comprehensive city model that could incorporate a large amount of data and yet be readily adaptable to representing different time periods, scenarios, or alternate reconstructions. In the course of realizing these projects, a workflow emerged which forms the basis of the Roman City Ruleset. The central challenge of this workflow was to integrate empirical data with procedural methods. Applying a generalized description for a Roman temple to an actual, excavated temple site in the Roman Forum, for example, tended to show the many ways in which such generalizations fall short. Therefore, the procedural rules rarely existed in a static state for long and were constantly re-written as the need for new parameters arose. This process of writing and re-writing procedural rules led to the discovery of elements which could be unexpectedly linked together and therefore systematized. As will be shown below, this became a knowledge-producing exercise in itself that informs, and is informed by, the research process, helping us create structural hypotheses to fill in the gaps left by incomplete remains, while allowing for the singularity of features and contexts and also the generation of plausible alternatives. I consider this workflow an "integrated" approach to procedural modeling because it aims at a holistic depiction of architectural data, incorporating geographic databases, published documentation, 3D

[59] The 'procedural rules' mentioned and cited below all belong to the 'Roman City Ruleset', were written for Esri CityEngine, and are the work of the author. See Appendix B below.

models, semantic descriptive rules, and interactive displays. This approach is intended to be a method for elucidating the logic of architecture as well as an efficient means of creating fully realized data models of ancient cities. In the sections that follow, I will describe the steps that make up this workflow and how the procedural rules were written.

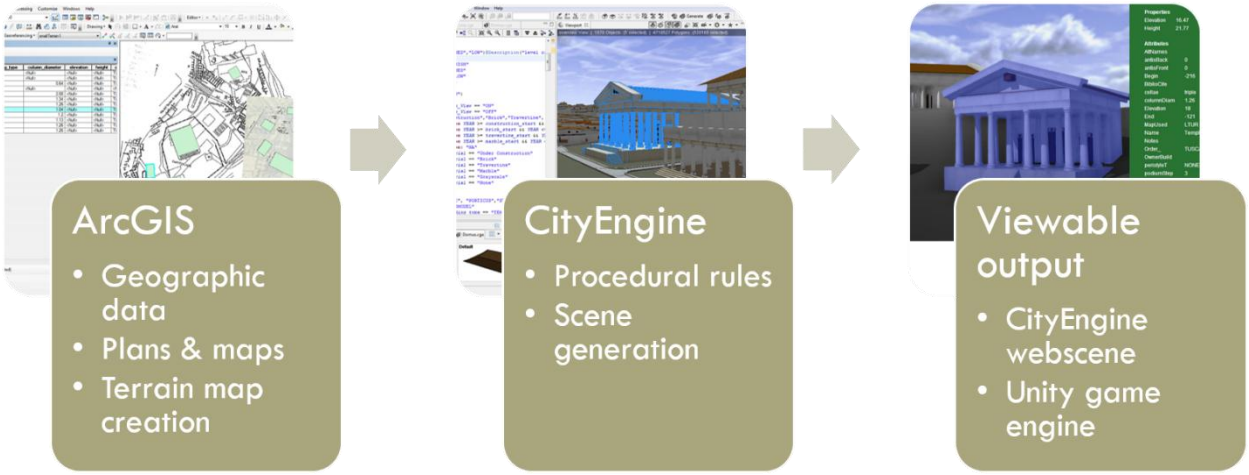

Fig. 15 Workflow overview

4.2.1 Reconstructing Terrain

The workflow begins with an ArcGIS map of the site which aggregates and georeferences all relevant published plans and survey data. A site model of the terrain is produced using contour lines or elevation points using ArcGIS Spatial Analyst's Topo to Raster tool, which interpolates a hydrologically correct digital elevation model (DEM) from the data.[60] Reconstructing and modeling ancient topography is as much of an interpretative process as the modeling of ancient buildings. When it is available, archaeological data that shows the elevation of a building's foundation can aid in the reconstruction of the ancient terrain level. However, this is rare in

[60] An overview of this tool, with references of technical papers, can be found at: http://desktop.arcgis.com/en/desktop/latest/tools/spatial-analyst-toolbox/h-how-topo-to-raster-works.htm#GUID-989894B7-1C35-4B46-8142-ACF3BFA6553C.

Magnesia, where most of the city lies several meters below the modern topography, and is obscured by erosion, building accretion, and other factors. Geophysical surveys of the modern landscape, particularly a 2003 survey by the Turkish government, provided the elevation points and contour lines which were used to reconstruct the macro-features of Magnesia's terrain, such as the location of steep slopes, ravines, and flat areas. For the larger area including Mt.Thorax for which no maps were available, this data was combined with contours derived from a terrain model 'grabbed' via SketchUp's Google Earth interface.[61] The two datasets were combined in ArcMap to create the overall terrain model.

Ideally, the data would be modified to reflect the accretion of silt in the plain which covers most of Magnesia's urban structures.[62] The Artemision, Stadium and Theatron have been mostly excavated to the depth of their ancient ground level. Yet, for the rest of the city, there was simply not enough information to determine the amount by which the terrain should be adjusted. Because silt accretion mostly affects the flat area of the plain and therefore would alter the terrain in degree rather than profile of slope, it was decided to use the modern terrain elevation as the basis for the present reconstruction.[63] The area of Magnesia that the city grid would have covered is fairly flat, with a slope affecting only the southernmost blocks. It is presumed that the grid did not extend into the foothills (Humann, 1904, p. 21; Bingöl, 2007, p. 128). Consequently, the alignment of the grid with the terrain does not present any particular challenges, as it does at more steeply-sloping sites such as Sagalassos, Priene, and Pergamon.

[61] More information on this procedure can be found at; http://help.sketchup.com/en/article/95069.
[62] A good example of this process is Romano, et. al., "Making the Map": (http://digitalaugustanrome.org/volumes/read/making-the-map) from the Digital Augustan Rome project. This article demonstrates the challenges of modeling historical terrain even for a site as well-documented as ancient Rome.
[63] Should additional data become available in the future, the 3D scene can be updated using the current workflow.

Also conjectured is the course of the Lethaios River, which certainly has changed in the course of millennia. Indeed, the outline shown in Humann's drawings from 1904 does not correspond exactly to the winding of the modern riverbed. In this case, as with the terrain model, I have followed contemporary data, since the shape of the river in Humann's day was still far removed from its course in antiquity, and determining the precise route at that time is beyond this scope of this study.

4.2.2 Geospatial Data

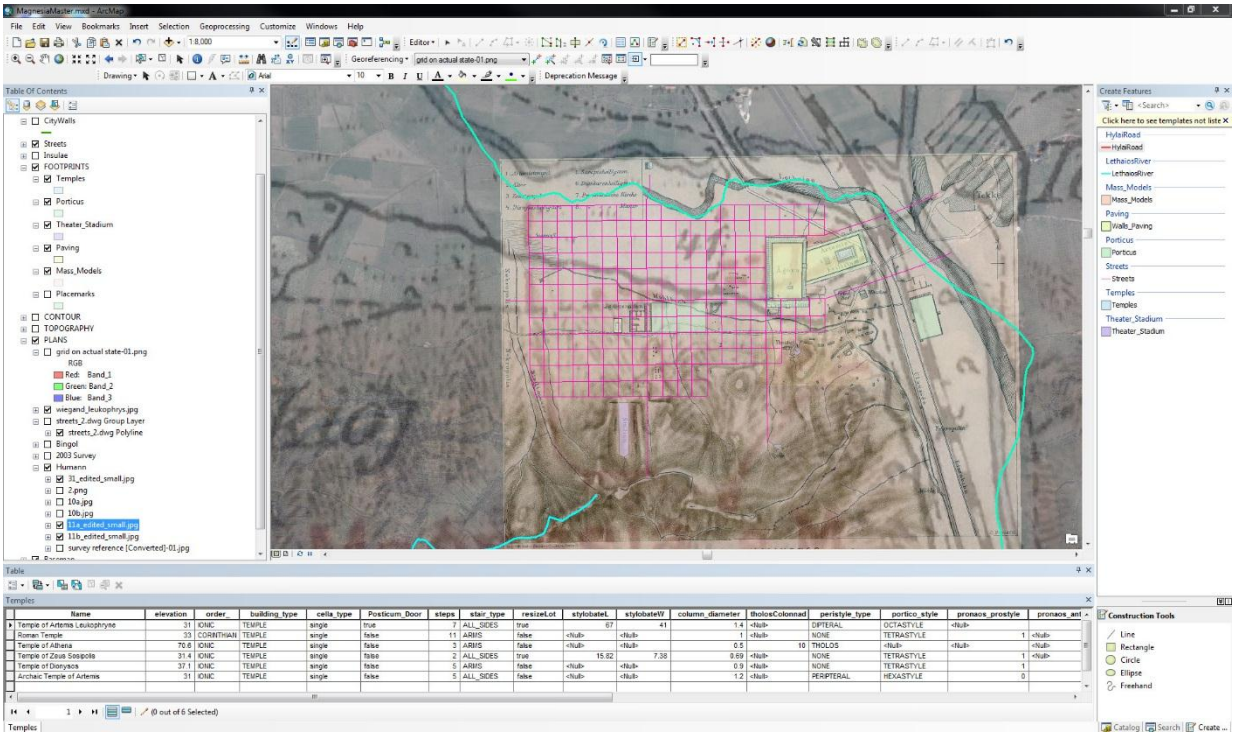

Fig. 16 Screenshot from the Magnesia GIS file.

When setting up a digital map, a geodatabase is created which will be the container for all of the model's metadata. Building footprints are drawn as a polygon layer in the geodatabase and fields are added that correspond to the attributes I have written into the 'Roman City Ruleset' and which will be displayed with the final model. In addition, fields can be created in the geodatabase that record the citation for each attribute's source, indicate whether the value is a

guess or an estimate, or provide further comments on the decision-making process associated with the model. The geodatabase is then imported to CityEngine, the procedural rules are applied to the building footprints, and the model is generated and finally exported to a web-based viewer.

4.2.3 Encoding Architecture

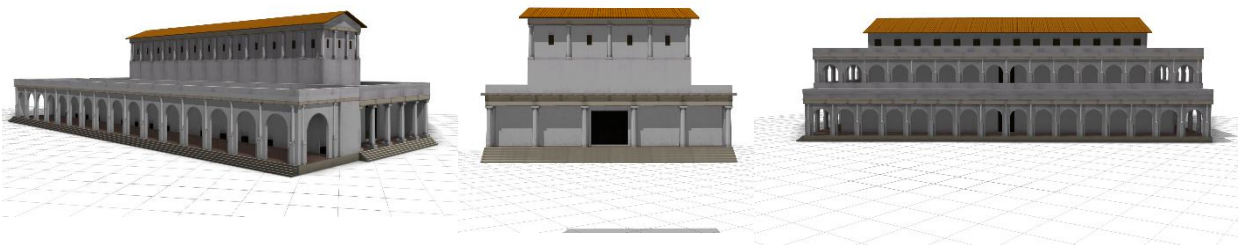

Fig. 17 Models generated from the 'basilica' and 'porticus' rules

So far, the process seems fairly automatic. However, this belies the thought and scholarship that must go into the authorship of the rules, which I consider to be the heart of the procedural methodology. Procedural rules are only 'automated' on the graphics end – the modeling still has to be done from scratch, albeit in a textual rather than visual interface. This means that procedural rules may be as general, or as specific, as the author writes them, just as a prose description of a building may be a generic gloss, or an in-depth study of a single structure. Which approach to take is entirely at the discretion of the modeler, and there are valid reasons to choose both.[64] When care is taken to discern evidence from conjecture, the writing and use of procedural rules need not be an attempt to impose a generalized, unified theory on diverse instances. Instead, the differentiation to be found in the built environment will serve to enrich and diversify the rule. The advantage of writing procedural rules is, then, the ability to express

[64] For example, a generic rule for a massing model when scant evidence is available, or an architecturally complex rule to compare dimensioning schemes on two similar buildings.

concisely and quantitatively the degree of known differentiation for a given variable – not only in semantic form, but in visual, three dimensional form as well.

Although it aids in the efficiency of scene generation, there are some drawbacks to the way in which shape-grammar based procedural languages enforce a hierarchical structure upon the rules. Because a change applied to a parent shape affects all of its child shapes, one must deconstruct a building's design in a very top-down way in order to model it. However, this may not be the way in which the building was conceived by its architects or builders. Furthermore, it is often the case that a historical building is the accretion of many layers of additions and alterations over time, in which the pieces are unrelated and stuck together and not reflective of a unified top-down schema that mirrors some Platonic abstract ideal. In a practical sense, too, shape grammars are limited by their language. For example, CGA shape grammar does not have a vocabulary to describe curved or radial geometry, and therefore building types such as theaters, stadia, and *tholos* temples can only be 'hacked' in terms of polygon splits. My rules for a theater or stadium, for example, would seem to have been a simple exercise in symmetrical, radial geometry. However, the procedural grammar was not well-equipped to describe such geometry, which made the writing of this rule a rather tortuous process.[65]

[65] Because of the lack of radial functions, the segments in the sphendone (rounded part) of the stadium or theater have to be manually drawn in ArcGIS (currently an imperfect process as well) and each face separately named numerically.

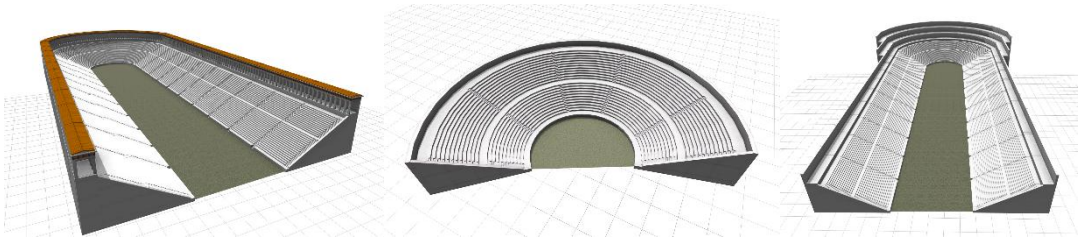

Fig. 18 Models generated by the Stadium/Theater rule

3.2.3.1 Temple Rule

In order to write effective procedural rules it is necessary to work backward to some extent, taking into consideration the rule's intended purpose and use. For example, a defined set of building layouts with well-documented proportional systems gave the 'Temple' rule a set of constraints that determined its structure (Wilson Jones, 2009). In this case, the crucial parameters – for example plan type, column order and column diameter – are often available from archaeology. Therefore, this rule operates *conditionally*, using many if/then clauses, for example:

```
case peripteral || closedAlae:
      offset(-column_diameter/2,inside)
      s(scope.sx-n*2+column_diameter,'1,'1)
      center(xz)
      Cella1(n)
```

This translates to a phrase that might sound like: "If the peristyle is peripteral or has closed alae, the cella will have a space the width of one intercolumniation on either side." This statement might be expanded or modified by additional clauses to express changes in this particular proportional rule over the course of antiquity.

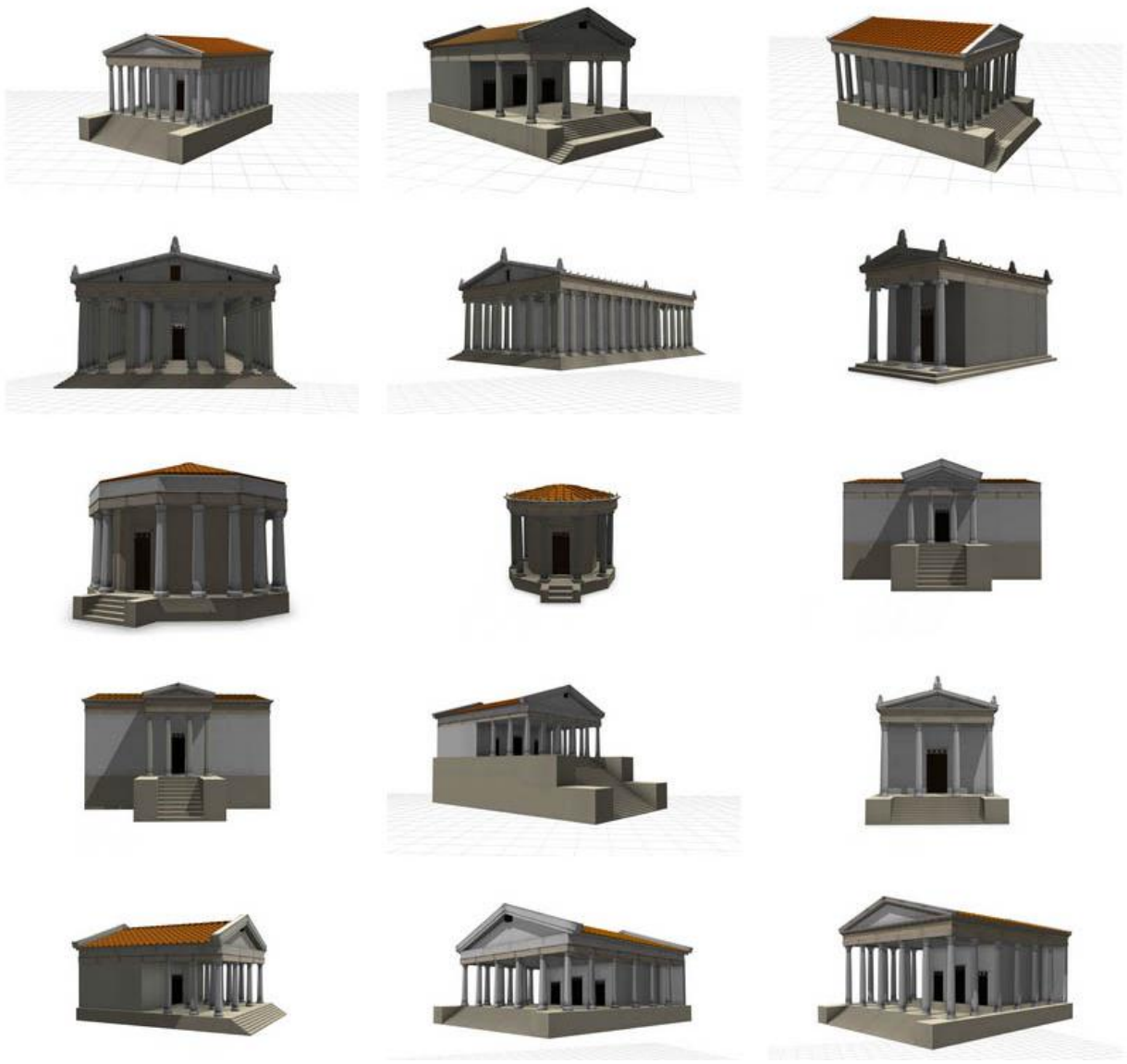

Fig. 19 Variations generated by the Temple rule.

A procedural rule that uses CGA shape grammar works from the most undifferentiated level of massing and breaks that volume down into ever-smaller parts which give the model its detail. In the Temple rule, the first steps establish the correct width, length, and elevation of the stylobate. Then, the parameters set by the GIS-imported object attributes tell the rule what kind of steps to give the podium. Next the cella and peristyle are extruded as basic masses. Depending on what plan type and column order have been specified in the parameters, these masses are then

split further according to the proportional system that has been defined in the attributes. Finally, non-procedural assets are inserted into the subdivided geometry to form columns and column capitals, roof tiles and antefixes.[66] The temple chooses between three versions of columns assets, each with a different resolution, depending on the level of detail selected in the procedural rule. This greatly simplifies the process of exporting models for different outputs, such as high-resolution rendering versus low-polygon models for interactive online models.

3.2.3.2 Domus Rule

While it re-uses pieces of the same code that makes up the Temple rule (e.g. colonnades, textures, colors, and assets), the 'Domus' rule, which I wrote for the Roman City Ruleset to describe Roman houses, operates slightly differently. It can either be built upon specific building footprints from the geodatabase, or tied to a dynamic street network that updates automatically as the streets are changed. This rule was meant to generate city infill for a much larger area than is actually known through archaeology, and not much evidence exists about the actual appearance of ordinary houses from the period in question, although general principles were derived from analogous evidence from other sites. Therefore, this rule operates primarily in a *stochastic* mode, generating random variations to mimic the variegated texture of an urban fabric. A typical clause would be:

```
case backyard == false:
split(z){~1*rand(.8,.9): Mass |'rand(.01,.05): NIL}
```

("for any given row of houses with no backyard, make the depth of each individual house a random value between 80% -90% of the maximum depth") The goal here is not a detailed single

[66] I have chosen to import these detailed elements as non-procedural assets due to their complex, idiosyncratic, or curvilinear geometry.

building, but that the overall massing be of the appropriate scale, level of density, and form to provide a plausible holistic view of the city. Of course, if a researcher possessed much data for houses, the rule could be written to reflect that. The beauty of procedural modeling is that is malleable to the researcher's source material and aims.

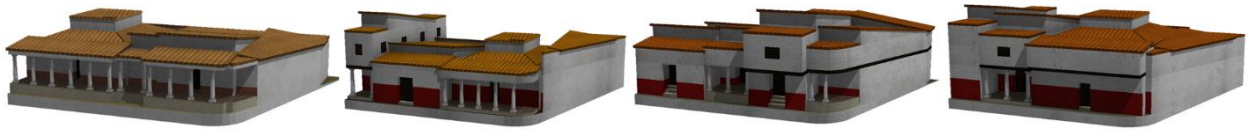

Fig. 20 Variations generated by the Domus rule

The Domus rule begins with lots that are generated from a street network. This street network might be imported as line shapefiles from GIS, or drawn by hand. Each time a street is moved or altered, the blocks and lots that adjoin it are parametrically altered as well. The initial steps of the Domus rule compensate for any slope that might be found in the underlying terrain and by constructing a level foundation for the lot. The lot is then subdivided, randomly, into houses of varying widths within a margin that is determined by a factor set in the attributes. Each house is set back from the road to a slightly different degree, in order to provide realistic variation in the streetscape. Finally the undifferentiated mass of each house is extruded and given a random seed variable that it will carry through all the subsequent steps of the procedural rule. This randomized variable will be used at different stages to ensure that each house is unique and slightly different from its neighbors. Each distinctive feature of the domus is slightly varied each time the seed is updated: the color of the walls and roof, the height of the floors, the number of floors, whether or not the house has an atrium, front porch, or back yard, the width and number of windows and doors. Some attributes can be determined manually if a specific house type is desired. For example, choosing `attr houseType = "SHOPS"` will generate the façade of each

house as a shop with a wider opening and a counter. An extended portico can be created in front of a row of houses by choosing `attr porch = "FRONT PORCH",` whereas if no porch is desired, `attr porch = "NONE"` can be chosen. For a realistic and randomly varied assortment, `attr porch = "MIXED"` would be the default setting.

3.2.3.3 Integrated Structure of the Rules

The suite of rules was designed to minimize duplication of code, therefore certain functions, such as colors, textures, and common architectural elements such as colonnades, entablatures, and roofs, were separated into modules which are referenced by each collating rule, such as a building type. A maximum number of variations can thus be generated with minimal duplication of code. Modules can be referenced by many different building types, for better consistency and efficiency. The suite of rules thus acts like a library of modular building blocks which can be combined to form different typologies, which in turn make up a city.

3.2.3.4 Attributes and Parameters

Attributes and parameters reflect one of the most malleable and interesting aspects of the procedural modeling process, because it is through these that variation and specificity are introduced. Attributes are the descriptors, named by the modeler, that inform a given model.[67] Some examples for the Temple rule define elevation, column order, and door width. Parameters are the values which fulfill the attribute, for example:

```
attr elevation = 10
attr order_ = "IONIC"
attr door_width = 1.3
```

[67] For more on how attributes operate within the CityEngine software, see the manual at http://cehelp.esri.com/help/index.jsp?topic=/com.procedural.cityengine.help/html/manual/toc.html.

As in the Temple and Hellenistic Houses rules described above, attributes may be hard-coded, stochastic, conditional, or probabilistic. Attributes can take their values from objects (usually from the geodatabase attribute tables), or they can be "mapped" onto the scene using a graphical map layer.

4.2.4 Use and Documentation of Sources

Each of the rules of the Roman City Ruleset was written around a 'default' set of proportional relationships that aims to appropriately accommodate actual data inputs. The rules were designed to model buildings that are no longer extant or have limited evidence, so the formal schema were derived from many sources, including comparable buildings, historical documents, and contemporary analysis. A simple example of the incorporation of primary and secondary sources into a 'default' rule, and its application to a specific reconstruction, is the Monumental Arch rule. Mark Wilson Jones (2000, 58) outlined 22 'propositions' concerning the principal proportions of the Arch of Constantine, many of which take as a common module the column height, which also equals one-third of the overall length of the building. Wilson Jones argues that the proportions of this Late Antique monument (c. 315 AD) were based on the Arch of Septimius Severus, built a century earlier, and that the use of simple arithmetic relationships based on squares and triangles (*ad quadratum* and *ad triangulatum*) in both monuments was typical of Roman elevation design in the Imperial period (Wilson Jones, 2000, pp. 60-61 *passim*). The detail he provides as well as the rationale of the 'representativeness' of this proportional scheme make it a good starting point for a procedural rule. But because the arches I will model are more likely to be known merely by the dimensions of their foundations rather than the height of their columns, I have re-configured Wilson Jones' primary dimension (2000, p.59, Fig. 13) as a factor of the overall length. This allows the modules to be expressed as:

```
const M     = length/3
const m     = M*.375
const p     = m*.0933
```

If one simply wanted to model the Arch of Constantine, these terms would be sufficient. But the design of monumental arch facades was incredibly diverse and inventive, and the rule must be robust enough able to accommodate a variety of different inputs. The Fornix Fabianus, for example, was a single arch, built in the Roman Forum in 121 BC, and survives only in fragments (Richardson, 1992, p.154). In order for the procedural rule to be useful in representing the Fornix Fabianus, Wilson Jones' proportional scheme was combined with the evidence provided by an extant arch closer to the Fornix Fabianus in form and date: the Augustan Arch at Susa, from the late 1st century BC.

Fig. 21 Triple and single arch generated from the Monumental Arch rule.

Using the rubric of Wilson Jones' modules, I found that the proportions of the arch at Susa could be expressed in the same terms with an additional expression as well as the variable attribute arch_type, thus:

```
@Group("General",2)@Range("SINGLE","TRIPLE")
attr arch_type          = "SINGLE"
const single      = arch_type == "SINGLE"
const triple      = arch_type == "TRIPLE"

const M     = case triple: length/3
else:   length/2
const m     = M*.375
const p     = m*.0933
const k     = p*4.75
```

The writing of procedural rules is essentially this process repeated over and over, as each application of the rule will be based on a different type of input and require a re-examination of the relevance of the source material.

Tracking the source for each expression in the rule is therefore essential. A simple working solution is to 'comment' the appropriate line of code with a citation and/or notes:

```
const M = length/3 // cf. Wilson Jones (2000, p.59, fig.13)
```

While 'constants' such as this one are internal to the rule, the 'attributes' (such as 'arch_type') are modifiable in each instance. Therefore, the source of the variable attributes attached to an individual model must also be documented. This can be done at the level of the geodatabase in the form of a bibliographic note that specifies how attributes were derived from sources. Any attributes not cited in this note are understood to be default values, and for these the annotated code provides the citation, as demonstrated above. The best way to present such annotation as part of the final model is a challenge on which will be tackled in the next phase of the digital project. It is hoped that eventually a more streamlined workflow for presenting annotations of the code and geodatabase will be developed, perhaps though a Python script or other customized

interface, since this functionality is not provided in the software. For the present, we shall have to be content that the relevant information is being preserved.

4.3 Application to Research Questions

In order to more effectively demonstrate the research potential of procedural modeling and the Roman City Ruleset, I will briefly introduce two prior projects centered on the city of Rome before addressing aspects of modeling specific to Magnesia on the Maeander. The two Roman projects are a methodological 'lineage' for Magnesia, and they also illustrate the range of outputs that are possible to achieve with the same core workflow.

4.3.1 Augustan Rome

Fig. 22 Screenshot from the Augustan Rome project

The Augustan Rome project is an example of how procedural modeling can be incorporated into the workflow of a diverse range of outputs. In this case, the research question concerned

how the use of building materials in Rome changed before and during the reign of Augustus.[68] In order to visualize this, the dates for each building phase and its associated material were first entered into the geodatabase.[69] I then wrote a clause into a "master rule" for the scene that determined the year to be visualized and changed the color and form of each building accordingly. Here, we chose to avoid realistic colors and textures in order to diagrammatically visualize change over time, and to highlight the visual impact of certain building materials, such as brick, travertine, and marble. The output platform in this case is the CityEngine web viewer, which has the advantage of using webGL to operate seamlessly in browsers, while preserving all the metadata from the original geodatabase. A user can compare different time periods side-by-side with a slider to change between views, turn layers on and off, or query the metadata using a search filter. For example, the search term "Augustus" returns all the buildings either named after or donated by Augustus, with the rest of the city greyed out. In order to show the impact of a flood on the city, a layer showing the extent of the flood level can be toggled on and off.[70]

In the Augustan Rome project, the advantage of procedural modeling lay in the relative ease with which we were able to turn a large database of the buildings and topography of Augustan Rome into a comprehensive city model, with all the metadata attached and visible in the final 3D product.[71] One challenge we faced was making the 3D content simple enough to keep the file size small for ease of downloading and streaming. Therefore much of the detail we were capable

[68] This project was led by Diane Favro at UCLA's Experiential Technologies Center. Project website: http://etc.ucla.edu/projects/augustan-rome.
[69] For this project, much of the data was drawn from Lothar Haselberger's *Mapping Augustan Rome* (2002).
[70] The 3D terrain model follows the reconstructed contour map of the Digital Augustan Rome project (see http://digitalaugustanrome.org/volumes/read/making-the-map). A plane at 15 meters above sea level was intersected with this terrain, and the areas where this plane appears above the terrain designates the flood extent.
[71] The current version of the Augustan Rome model, the result of several iterations, was the product of several months' discontinuous labor but was built on the procedural rules of the Roman City Ruleset, which had been in development for several years.

of generating had to be sacrificed. Procedural modeling helps simplify the process somewhat by allowing for level of detail to be built into the procedural rules by the author. Therefore, the same model may be easily re-purposed for a high-resolution detailed rendering, or simplified if the research goal does not require realism.

4.3.2 RomeLab

Fig. 23 Screenshot from RomeLab, in Unity game engine

The challenge of dynamically streaming procedural content in a gaming context was taken up in another project, RomeLab,[72] which also makes use of the Roman City Ruleset. In contrast to the diagrammatic rendering of the previous example, here the procedurally-generated models were exported to the Unity Game Engine,[73] which allowed the creation of a web-based, multiplayer game environment where up to 30 avatars can 'walk' through the space at one time, interact with objects, or even fly above the city streets.[74] One of the many research objectives the model served was the investigation of the spatial impact of temporary and permanent structures

[72] RomeLab (http://romelab.etc.ucla.edu/) is led by Chris Johanson at UCLA's Experiential Technologies Center.
[73] Unity Game Engine: http://unity3d.com/.
[74] The code that powers the multiplayer server was developed specifically for the Humanities Virtual World Consortium, and deployed in RomeLab.

on performance and spectacle in the Roman Forum. Once again, one of the principal advantages of procedural modeling in this case was its ability to easily rapid-prototype alternative reconstructions. In one phase of the project, six different 'scenes' representing different building phases were presented side-by-side (Saldaña and Johanson, 2013). The first-person avatar perspective provided a different, very instructive viewpoint from which to judge the impact that results from even a slight alteration to terrain elevation or building proportions. Factors such as these are sometimes extremely difficult to appreciate in a standard birds-eye view of a 3D model, let alone in a two-dimensional drawing.

4.3.3 Modeling Magnesia

Fig. 24 Procedural model of the temple of Artemis Leukophryne

Much of the Roman City Ruleset was originally written for the city of Rome in the 2nd c BC – 1st c AD and in order to model Magnesia had to be expanded to adapt to Greek models which extend back to the 5th c BC. The Magnesian case study, while falling within the architectural vocabulary of classicism, belongs to a different culture and environment, and chronologically

precedes the Roman exempla. Therefore, nothing from the ruleset was assumed, and all definitions were examined for their appropriateness to the present context. Some of the rules, such as the Temple rule, had already been developed with Hellenistic precedents in mind and needed little modification since they were already quite comprehensive. Therefore, it was a simple matter to conjecturally represent the temple of Dionysos as an Ionic building, based on prototypes from Pergamon and Teos.[75] The Roman temple was based on the similar footprint of the temple dedicated to Augustus in Antioch in Pisidia (Fig.25). The Theater/Stadium rule was custom-written around the Magnesian examples, as was the altar of Artemis. Some general rules such as bridges and streets were easily integrated, while in other cases, for example the city walls, the rule was written as simply as possible to avoid an excess of detail which would not have contributed to the purposes of this study. The baths and apodyteria are represented as generic mass models, since we only have enough evidence to determine an approximate scale. For the temples of the Dioscuri and Sarapis, of which nothing is known except their locations and dedicatory inscriptions, simple massing models are used as well. No matter the complexity, each rule was built around temporally-inflected attributes in order to visualize the development of the city over time. Four versions of the city model were created, representing the Archaic, Classical, Hellenistic, and Roman periods (Figs. 3,4,6,10).

[75] The choice of plan for the Magnesia conjecture follows Pergamon (prostyle) rather than Teos (peripteral) mainly due to the probable small size of the temple. Procedural modeling allows the building footprint size to be easily changed, as well as the parameters affecting plan and column diameter. Thus while the plan of the temple is hypothetical, a range of appropriate guesses can be generated using the model.

Fig. 25 Conjectured procedural representation of the Roman temple north of the Lethaios

The Hellenistic era houses were modeled based on the type of Priene as represented by Hoepfner and Schwandner (1987, p.171, Fig. 172). As such they show very little variation in plan or elevation, and since we have no evidence of Hellenistic houses from Magnesia and do not know how they were aligned to major streets, I have chosen to leave the visualization of the housing fairly generic and to vary the orientation of the housing within individual blocks somewhat more than is allowed in Hoepfner and Schwandner's reconstructions. In this version, orientation is procedurally controlled, with house lots being automatically aligned according the width of adjacent streets (Fig.26b).

Fig. 26 a. Insula model with regular orientation (left); b. with mixed/random orientation (right).

The case study of Magnesia, the focus of this dissertation, tests the efficacy of the Roman City Ruleset in modeling a complete city in multiple temporal variations from start to finish. It serves as a trial of the ruleset's adaptability to different eras and contexts within the classical world. The model's usefulness for the purposes of this study will be an indication of the viability of the procedural methodology and holistic urban models for future research. Far from being a modeling exercise which exists solely within the confines of a single project, the procedural rules that drive the reconstruction of Magnesia are designed to form the basis of a continually growing and evolving library of architectural knowledge for Greco-Roman world.

4.4 Summary

My aim in this chapter has to been to introduce procedural modeling as a powerful new methodology that has yet been underexploited by the Digital Humanities, perhaps because the technical focus of much of the literature has obscured its potential for directly addressing humanistic concerns. Procedural modeling allows the investigator to approach visual and 3D content through a rigorously syntactic and process-oriented framework, and preserves the hierarchy of decisions that result in a visual interpretation of archaeological evidence. Eventually, I plan to make the Roman City Ruleset available to others who are interested in adapting, expanding, and making use of it for their own research. I am also interested in working with large datasets to comparatively model multiple cities. Another possible direction for future work is suggested by recent developments in computer vision work on procedural modeling, which attempt to reverse-engineer procedural rules from existing buildings (Vanegas et al, 2012; Mathias et al, 2012; Thallera et al, 2013). This is an approach which could hold potential for architectural historians, perhaps as a method of 'distant reading' or 'thick reading' large corpora of buildings and comparing the findings against theoretically-driven procedural rules. Although

the use of 3D models in support of scholarly arguments is still in the early stages, procedural models are extremely information-rich and the ways in which they can be used to aid research are just beginning to be explored.

Chapter 5: Towards an Experimental 3D Reconstruction of the City

The city model of Magnesia was put to use in service of three research questions. The first concerns the layout of the city itself. The procedural methodology allowed for the fine-tuning of the street grid as it underwent changes over time and was informed by the mandates of different hypothetical scenarios. Some of these scenarios relate to the second area of investigation, in which various possibilities for the locations of the cults of Apollo and Dionysos within the city of Magnesia and its environs are tested and the various networks formed by these locations are holistically analyzed, in order to give spatial presence to performative aspects of Magnesia's terrain. The model thus functions multimodally, dealing both directly and indirectly with evidence, providing insight not merely through positivistic cartographic representation but also through inference, so that elements whose form is indistinct or unknown are given as much weight as those which are more defined. This forms a critical part of the third research question, in which the spatial tracking of ritual and movement allows Apollo's cave to emerge as an operative presence within the dynamic of the city. In the final section of this chapter the methodology itself comes under scrutiny and the outcomes of the project as a whole are evaluated.

Fig. 27 Plan view of the 3D procedural model (proposed layout for the Roman period shown)

5.1 (Re-)Drawing the City Grid

Drawing was an important preliminary step in the process of modeling Magnesia in three dimensions. Before the procedural rules could be brought into play, the dimensions of the city blocks and the outline of the grid had to be established. This was most effectively accomplished by many CAD drawings and sketches which tried to fit different proportional schemes to the actual state plan, taking note of where corners of major buildings indicated a street might have existed. This process and its outcomes are detailed below.

5.1.1 Problems of the evidence

The biggest challenge in the reconstruction of Magnesia's city plan is the fact that no pan-urban street patterns have been identified. The only streets and blocks of which we have a record were documented by Humann in the course of his excavations of the agora in the late 19th century

(Humann 1904, Blatt II). There is even less information available for the form of Magnesia's houses; the only building excavated within the blocks south of the agora was a peristyle structure (Fig.12:8) Humann proposed as Magnesia's *prytaneion*, the seat of the municipal government (Humann 1904, p.112). Published plans are problematic, as the reproductions are of varying size, detail, and quality; therefore it is difficult to obtain precise measurements.

As the first step in the modeling process, I scanned and georeferenced the published plans by Humann and Bingöl,[76] but a degree of accuracy was necessarily sacrificed to the resolution of the scan and other inconsistencies.[77] One of the enlightening, yet sometimes frustrating outcomes of this step in the georeferencing process is that it compels the researcher to acknowledge variations in the data and make critical choices about which source to follow, documenting decisions along the way. In this case, I have followed satellite imagery wherever possible to locate remains; with regard to what is no longer visible Humann's map (published in Kern 1900, frontispiece) is most complete and accurate. This disclosure made, the closer study of the available evidence and comparanda that follows does allow us to develop theories about the city plan.

5.1.2 Previous Interpretations

The basic outlines of Magnesia's general layout are fairly clear even from the limited evidence. Magnesia was founded in the classical period, when urban planning was becoming highly

[76] Plans of Magnesia in other publications invariably reproduce those of Humann, Bingöl's plans also rely heavily on the German drawings. The streets excavated by Humann, being presently underground, were not part of the digital file that was available to me.
[77] Similar problems were recounted by Scherrer (2001, p.80), who made a digital reconstruction of the grid at Ephesus.

regularized and theoretical.[78] Newly-founded cities in this period followed the innovations of Hippodamos of Miletos, using regular numbers in the dimensions of insulae for ease of planning and divisibility (Hoepfner and Schwandner 1986, Abb.251 p.252; Scherrer 2001, 86). The presence at Magnesia of the "Ionian" type agora also found at Miletus and Priene suggests that Magnesia's insulae were based on the Hippodamian model. Humann (1904, p.21) observed that the streets probably intersected each other according to a grid system as at Priene, but stopped short of providing precise dimensions of the blocks or streets apart from the graphical representation found in his plans. These drawings, showing three streets running up to the south stoa of the agora, are the sole remaining suggestion of what the street grid may have looked like. R. Martin (1974, p.114, p.123) noted the contrast between the orientation of the Artemision sanctuary and the new city plan as the opposition of old and new tendencies; he also observed that the agora comprised exactly six insulae and intersected one of the main streets. Martin gives the dimensions of Magnesia's insula as 98.5 m x 42.5 m, or 300 feet by 130 feet, assuming a foot of 0.328 m. The actual size of the Greek foot differed considerably over time and space,[79] but it seems odd that Magnesia would use such a long 'Doric' foot in planning their city, while Priene, which was founded nearby around the same time and under similar circumstances, used the shorter and more familiar Ionic foot of 0.295 m (Thonemann 2011, p.244). A length of 300 feet might make sense as half a *stadion*, but the long foot would give us a *stadion* of 196.8 m, longer than Magnesia's actual stadium, the length of which was given by Clerget as 185.9 m (Humann, 1904, p.29; Bingöl, 2007, p.172). The stadium was likely to have been planned at the origin of

[78] The date of Magnesia's founding is dated by Bingöl (2007, 31) to 386 BC or later. By comparison, the plan for Piraeus, the first 'Hippodamian' city, is dated to 451 BC. On the theoretical nature of Hippodamian cities, see Hoepfner and Schwandner (1986).
[79] Common lengths are between 0.295-0.297 for the Attic or Ionic foot, to 0.326 – 0.328 for the Doric foot (Boyd and Jameson, 1981, p. 332).

the city, and while the dimension used to determine its length might have resulted in its being shorter than Clerget's measurement, it could hardly have been larger.[80] Bingöl (2007, p.133) for his part, also divides the agora into six insulae and offers measurements of the blocks as 96.35 m by 41.66 m, but does not explain how this works out in feet. Indeed, these figures do not seem to correspond to any likely dimensional scheme.

5.1.3 Proposed Dimensions of the Insulae

Martin's and Bingöl's proposals for Magnesia's design appear to be correct in the supposition that the city plan was based on the agora, and that the agora itself, excluding the southern stoa, represents six city blocks. The problems and inconsistencies in their interpretations arise from the manner in which the subsequent grids are drawn. Both take the insula, not including the streets, as the basic building block of the city layout. This is common in cities founded in the archaic and early Classical period such as Naxos and Himera where long strips of insulae determined the pattern and streets were narrow and of secondary importance (Boyd and Jameson 1981, p.340).[81] After the 5th century BC with the advent of Hippodamian-type town planning however, planners began to contend with the overall layout of a town. This meant the blocks became subordinate to the streets in shaping the city and less likely to serve as the basic unit of measurement. Vitruvius tells us that surveyors would lay out the main lines of the streets and public areas first, subdividing large areas into smaller ones (Vitr. 1.5.ff.). Examples can be seen at Rhodes (Kondis 1958; Wycherly 1964); Halieis (Boyd and Jameson, 1981), and Ephesus

[80] I.e. the existing stadium could be larger than the original one, if for example the struture was enlarged in the Roman period, or the precise positions of the start and finish line (the datum for the measurement) is uncertain.
[81] See also Komboti, Olinthos, Kassope. The 'strip' (*per strigas*) type cities were analyzed by Hoepfner and Schwandner (1986, 250), who differentiate the longer blocks theoretically as well as formally from the later Hippodamian type, calling them *streifenstädten* ("strip cities"). The blocks were long and of variable length, rather than adhering to a proportional system. R. Martin associates this type with agriculturally-oriented communities where equitable distribution of arable land was of primary importance (Martin, 1973, pp.97-107).

(Scherrer, 2001). This approach preserved overall harmony at the expense of small variations in the dimensions of the insulae.[82]

More importantly, though, when the grid includes the area occupied by streets at Magnesia it is possible to find a much more convincing dimensional scheme. Keeping the division of the agora in six blocks (but not leaving gaps for streets), the pattern that emerges is of a module measuring 106.92 m long by 47.55 m wide, or 360 feet by 160 feet, using a foot of 0.297. This unit is much more similar to the 0.295 Ionian foot found at Priene. The block is of the same 9:4 proportions proposed by Martin, but the scheme makes more sense. The length of 160 feet, common in Hippodamian insulae, is also found at Priene, Ephesus, and Piraeus.[83] The longer dimension of 360 feet meshes is suggestive of the units of 36 or 50 *plethra* (100 feet) used in quadrature for the division of rural land (Heimberg, 1984). Thonemann (2011, pp. 243-244) has found evidence for the use of quadrature in inscriptions that record the sale of 50-*schoinoi* farm plots on the civic territory of Magnesia at the turn of the third century BC (Kern 1900, no.8).[84]

Moreover, this dimensional scheme also corresponds with the dimensions of the Artemision (Fig.28). Turning the block on axis with the Artemision, we find that its width (47.5m) delineates the width of the temple at the base of its steps, while the interior of the sanctuary is equal to four blocks. The stoas of the agora and Artemision, as well as the temple of Artemis, all date to Magnesia's "building boom" of the mid-late third century BC, so it is not

[82] Hoepfer and Schwandner (1986) prioritize the individual house lot as the primary unit upon which town plans were based. While this seems more true of the 'strip' type cities of the archaic and early Classical era, the strictness inherent in this granular approach does not seem to do justice to the flexibility of the overall vision that certainly guided many later cities.
[83] Priene: see Hoepfner and Schwandner (1986, pp.150-153); Ephesus: Scherrer (2001, p.86); Piraeus: Hoepfner and Schwandner (1986, pp.13-15, Fig.10).
[84] Thonemann (*ibid.*) finds similar evidence at Priene.

surprising to find the same proportional scheme underlying the entire complex. What is less clear is whether the dimensions of the older temple of Artemis and its precinct had an influence on the derivation of the 360 x 160 foot module in the original planning of the city.

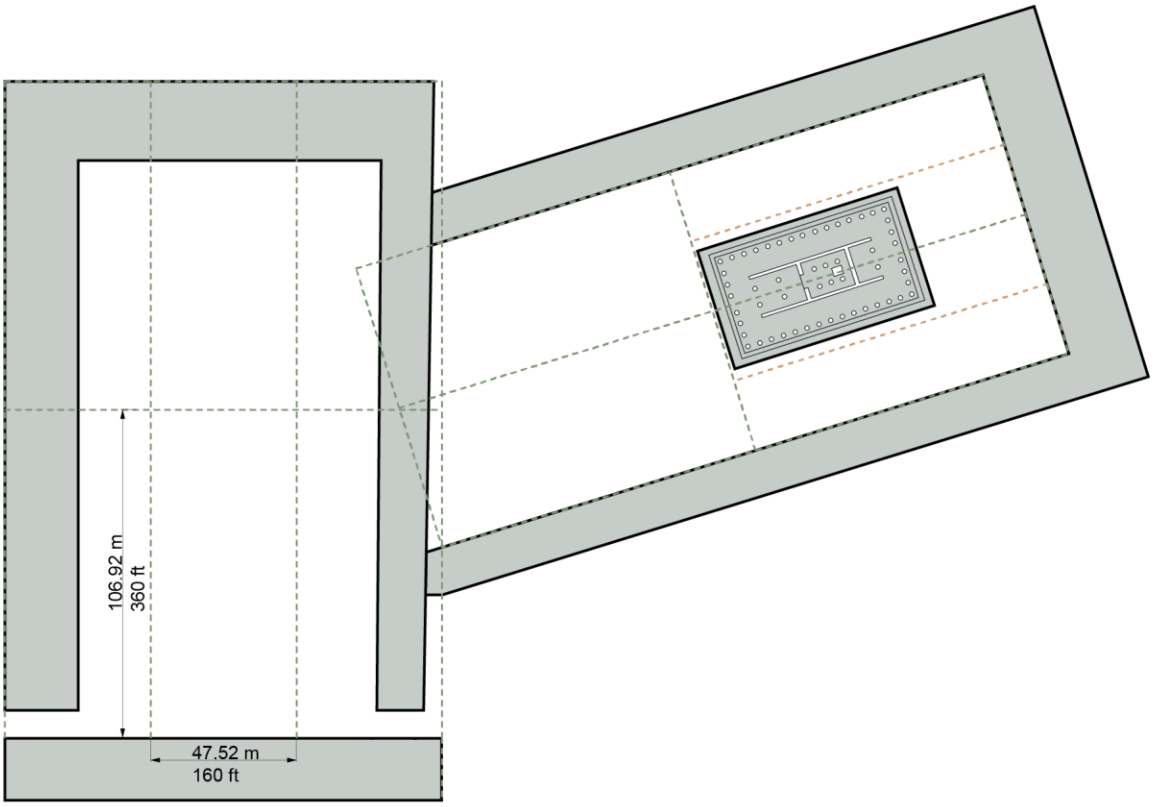

Fig. 28 Derivation of the insula dimensions. The agora comprises 6 insulae while 4 insulae, turned on axis with the temple of Artemis, delimit the area of the Artemision enclosure.

Arrayed over the city area, the 360 x 160 ft. block fits remarkably well with the actual state plan (Fig.29). Even if they were used to apportion lots of rural land, quadrature-derived squares seem absent from the plan of the city itself. However, the area between the agora and the west wall of the city can been seen as divided up into four equal rectangular quadrants (Fig.29). Such an arrangement lends itself to division into twelve sections of 8 blocks each. Inscriptions attest that the population of Magnesia was divided into twelve *phylae* or tribes, each named after

a god (Kern 1900, p.212 sv. *Phyllen*; Szántó, 1906, p.275), in emulation of Plato's ideal city, also called Magnesia (Plat.Laws 5.745d-e), though it is probable that the cult of the twelve gods was introduced after the original planning of the city (Long, 1987, p. 222).[85] It is possible, however, that the planners of Magnesia may have been especially motivated to create a geometrically satisfying, even philosophically meaningful layout because of the approach over the hills that would have afforded a 'bird's-eye' view of the city to travelers arriving from Ephesus.

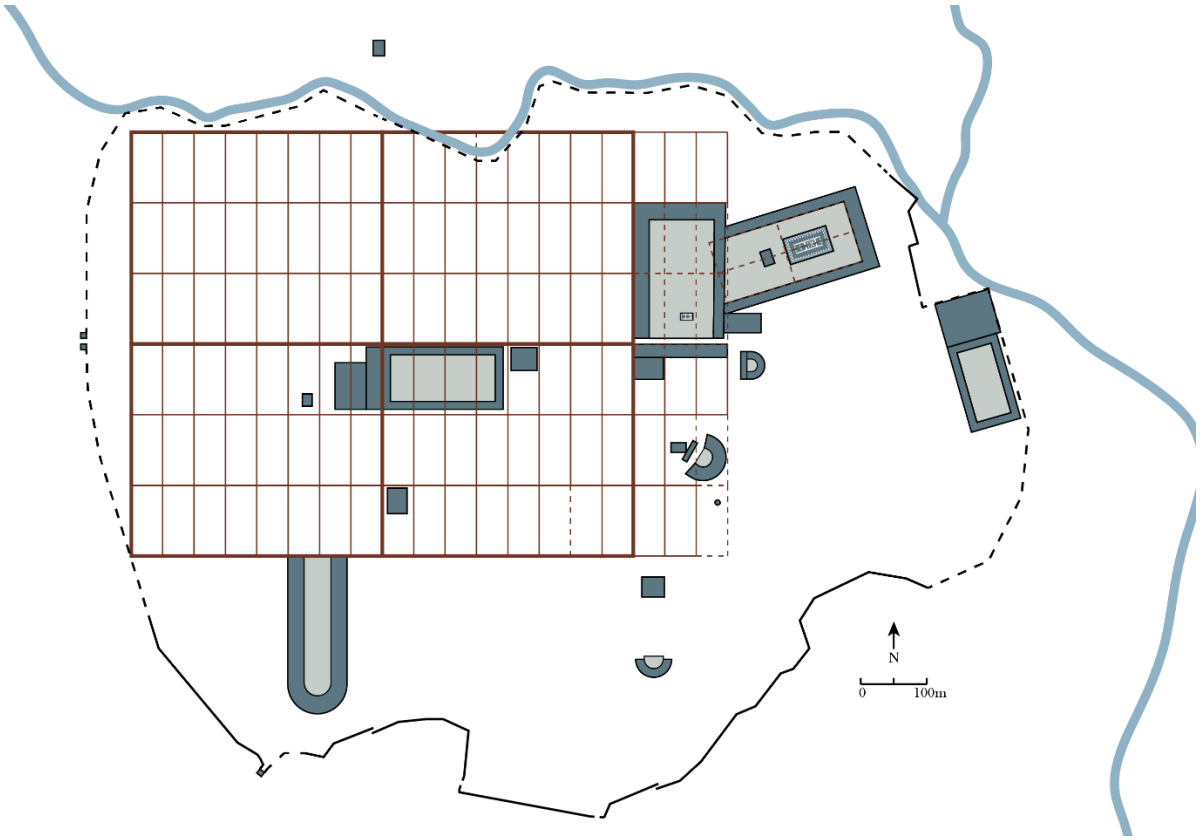

Fig. 29 Proposed grid overlaid on actual state plan

[85] An inscription from 196 BC (Kern, 1900, no.98) describes a procession in honor of the twelve gods to a temporary altar set up in the agora, in association with the sacrifice of a bull to Zeus Sosipolis (Long 1987, pp. 248 – 251).

5.1.4 Division of the Insulae into House Lots

Due to the lack of excavated houses at Magnesia, the manner in which insulae were divided into houses is a matter of pure speculation. Detailed studies of other cities, however – particularly Priene – allow us to venture a guess as the approximate shape and size of Magnesia's typical residences. Older Greek cities were typically planned in strips made up of two rows of approx. 50 ft. sided square house lots. This meant that widths of 100 ft. (described as *plethra*) or 120 ft. (*schoinos*) were common insula dimensions (Boyd and Jameson, 1981, p.335). In Hippodamian-type systems, however, house lots become more rectangular and insulae wider (Hoepfner and Schwandner, 1986, pp.258-9). Magnesia's 160 ft. wide block suggests that it was divided into four lots across, in the manner of Piraeus, Priene, and Abdera (*Ibid.*, Abb.255). After subtracting for the streets around 30 ft. (8.91 m) from the short edges and 20 ft. (5.94 m) from the long sides, and supposing a 10 ft.(2.97 m) alley bisecting the block, the two smaller blocks would measure 160 feet by 140 feet. This in turn could be subdivided into eight 80 ft. by 35 ft. blocks, similarly to Abdera (70 ft. by 35 ft.) or Priene (80 ft. by 30 ft.). The remaining area could be split into 18 lots of approximately 247 m² (Fig.30).[86] This hypothetical division would allow Magnesia's houses to be oriented towards the major east – west roads, corresponding to the flow of traffic through the city. It would also allow for favorable southern sun exposure in the courtyards, as well as accommodating the slope of the land which descended from the foothills of Thorax in the south, to the banks of the Lethaios in the north. This is the pattern observed at Priene. Based on this analog, we may presume a three-part division of the houses' rectangular shape into a front storeroom structure, a courtyard in the middle, and dwelling at the back of the lot (*Ibid.*, pp.169-176; pp.264-267). However neatly we may conjecturally divide up a typical

[86] By comparison, Priene had lots of ca. 207 m²; Hierapolis ca.208 m²; Miletus ca. 260 m², and Piraeus ca. 242 m².

block, the measurements of individual houses would vary according to the width of the adjacent streets, the more important dimension being the larger 360 ft. by 160 ft. module.

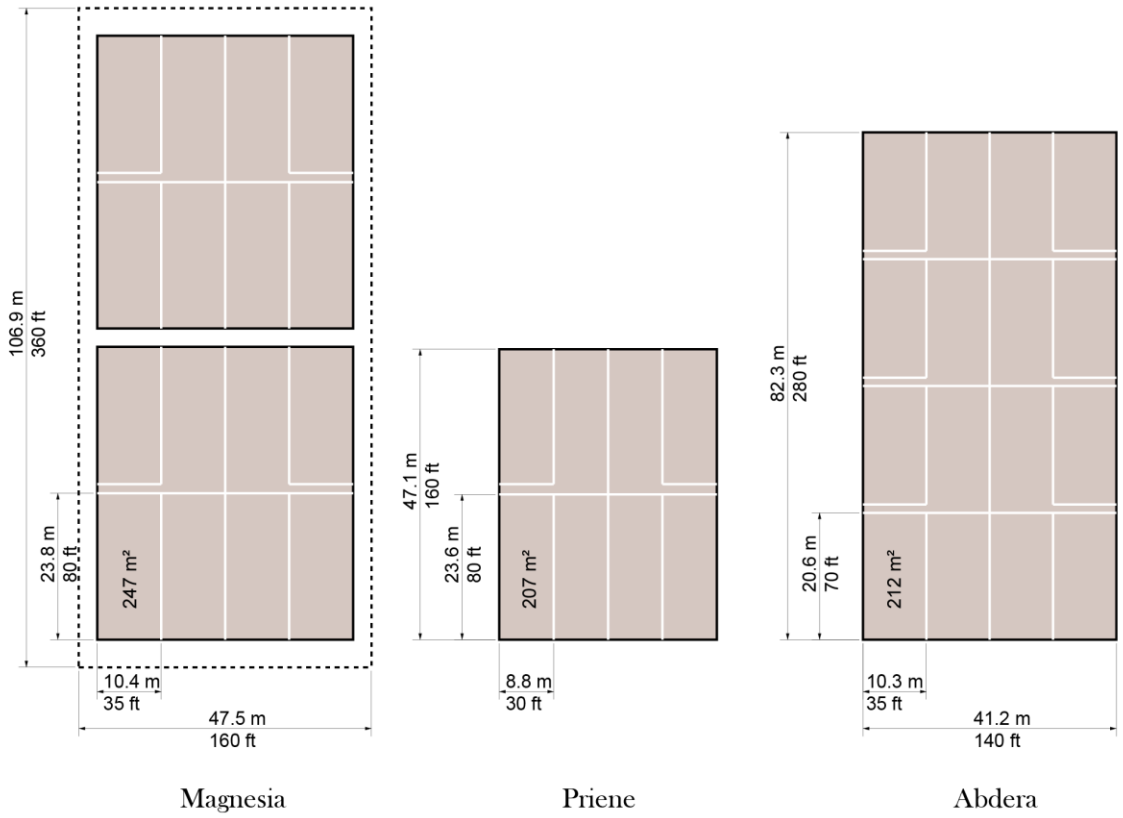

Fig. 30 Proposed division of Magnesia's insulae with comparisons (Priene and Abdera after Hoepfner and Schwandner, 1986, p. 259, Abb. 255)

5.1.5 Extent of the City, Number of Blocks and Population

According to a referendum of the second century BC (Bingöl, 2007, p. 138) the number of citizens participating was 4,678. If these constituted one-fourth of the overall population (the rest being women, children, slaves, and non-citizens), and there were approximately 10 inhabitants in each dwelling (Billows, 2005, p.201), then the overall population of Magnesia at this time was around 18,712 and would require 1871 houses on 104 city blocks. Magnesia's grid of four rectangular quadrants consists of 96 blocks, perhaps indicating that by the time of the

referendum, some 150 years since the city's founding, the city had grown beyond its originally designed capacity (Bingöl, 2007, p.138). Another possibility is that the referendum included citizens who lived within Magnesia's civic territory on family farmsteads or in villages. . If the city had been intended for 10,000 - 12,000 people, as Bingöl (*ibid.*) suggests, then 96 insulae would have been more than sufficient. After dividing these 96 blocks into 12 sections, each *phyle* would have comprised approx. 1,440 people, probably supplemented by the population residing in the local countryside.

An overall conclusion suggested by the results is that Magnesia was founded, as were many Hippodamian cities (Kostof, 1991, p.105), with an eye to future expansion. Space was left between the original four quadrants of the city's grid and its walls, and the eventual extents of the insulae are not known. There was enough space within the city walls to add additional insulae on both the eastern and western edges of the grid. In Hellenistic times Magnesian rural real estate had become exceptionally expensive, perhaps due to the cultivation of cash crops such as vines, olives and figs (Thonemann, 2011, p.246). For most of antiquity, then, Magnesia was a mid-sized city with prosperous agricultural holdings, larger than mountainous Priene but smaller than the major port city of Ephesus.

5.1.6 The Major Streets

A key characteristic of a city's main thoroughfare, both in Hellenistic and in Roman times, was that it passed from gate to gate and intersected the main public space of the city, the agora (MacDonald, 1988, p.3; Parrish, 2001, p.11). Clerget documented a gate in Magnesia's western wall that had all but disappeared by Humann's time. However, we know from Clerget's drawings (reproduced in Humann, 1904, p.19, Fig. 4) that the gate's flanking towers were 8.6m square, and the opening in between is roughly the same dimensions. Therefore, we may presume a width

of 30 feet (8.9m) for this street. Magnesia was on the road which connected Ephesus and Tralles, and the route which linked the western gate with the Agora must have been the main thoroughfare of the city. Bingöl (2007, p.136) indicates that the road south of the gymnasium was the city's main axis, citing the fact that this is still the major route through the site today, albeit 6m above its level in antiquity. However, this is probably a misinterpretation, since the road south of the gymnasium does not pass through the agora. Bingöl appears to interpret a break in the line of the city walls on Humann's plan (Kern 1900, frontispiece) as indicating the location of a gate. Humann, however (1904, p.20) states that the location of the west gate could not be ascertained by his survey. Furthermore, the foothills of Thorax rise rather sharply to the south of this street, whereas the street passing through the agora is located more centrally in the plain and therefore we may assume within the grid, allowing space for the major buildings, shops, and colonnades that must have flanked it on both sides. Finally, the road south of the agora abuts the Lethaios gymnasium, and it is unlikely that this large building would have been placed directly blocking the major city gate that must have existed at its eastern end.

The northernmost limit of the bath/gymnasium complex, of Roman date, is slightly offset from the street width marked by the gate to the Agora, suggesting that the intervening space could have accommodated a colonnaded walkway in Roman times.[87] Once it passed through the agora, this road must have either turned to the north-east towards Tralles or to the south-east towards Priene. In both cases the road would bypass the Lethaios gymnasium and exit the city

[87] A short stretch of columns west of the Agora has been identified by Bingöl (2007, 188) as a colonnaded street; however this is more likely to have formed part of a portico or forecourt; the location of the 'street' does not coincide with a major thoroughfare; it is rather narrow (5m) and its length only runs about 30m before turning a corner at a right angled. Its other end abuts the massive, arched corner of a Byzantine building of uncertain purpose. For this reason, we might conjecture that the colonnade was erected as a fountain or an extension to this building, and is not one of the major routes of the city.

through a gate in the wall, connecting with a bridge crossing the Lethaios. In support of this theory, we may cite the evidence found by Bingöl (2007, p.129) for two ancient bridges over the Lethaios, visible both north and south of the modern bridge (Fig. 31). The road between Tralles and Priene, passing through the Argavlı strait along the banks of the Lethaios, ran from north to south on the eastern side of Magnesia. The two bridges (and presumably gates) on this side of the city, therefore, would allow access in both directions.

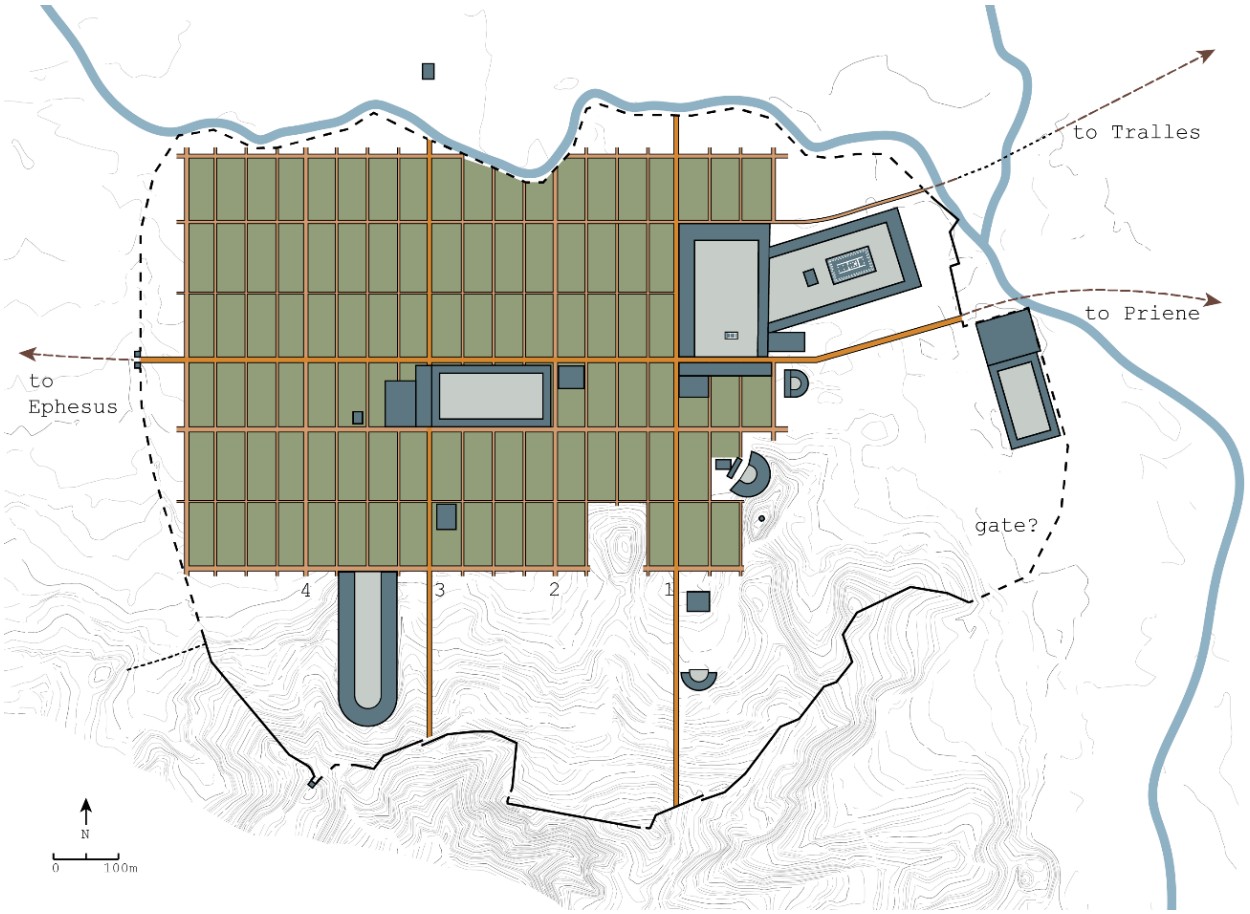

Fig. 31 Insulae and streets. Major routes highlighted.

The topography of Magnesia, bounded as it is to the south by mountains and to the north by a river, as well as its position on an east-west route, suggest that there was probably not an important *cardo* or north-south axis. The streets oriented in the north-south direction must have

been narrower and of less importance to the city plan than the east-west avenues, though the positioning of major buildings seems to indicate larger north-south streets every four city blocks (Fig. 31). The first of these is the street that ran past the west wall of the agora, and presumably continued into the hills to reach the theatron (Fig. 31:1). Another major street (Fig. 31:2) would pass between the east wall of the gymnasium palaestra and the corner of a large Roman building described by Bingol (2007, p.188) as "roughly square in plan with the remains of vaulted foundations at its centre" (Fig. 2:18).[88] Perhaps the most important north-south street is that which bisects the city grid, drawing a line from a door in the city wall in the southern foothills all the way across the Lethaios where it aligns with the Roman temple on the north bank (Fig. 31:3). Finally, according to this scheme a major street would align with the tower that sits above the stadium at the southwest corner of the city wall, flanking the western edge of the stadium (Fig.31:4).

5.2 Modeling the Space of Ritual

Although these basic outlines can be posited with confidence, our knowledge remains incomplete, and our propositions await archaeological verification. However, the results allow us to piece together a comprehensive albeit conjectural picture of Magnesia's urban development, and it is possible to begin to investigate ritual movement in the city and how it changed with the urban fabric over time. In Chapter 3 the historical evidence for cults of Apollo at Hylai and Dionysos of the plane tree was explored. The section that follows maps this evidence onto the spatial terrain provided by the 3D simulation.

[88] I was not able to find any additional information about this building, neither its date, plan, nor extent, and therefore have only indicated it generically in the plan.

5.2.1 The Path of the *Dendrophoroi*

The cave of Apollo at Hylai, if located in the vicinity of the site in the foothills of Thorax that directly overlooks Magnesia, would have been connected most directly with the city by way of a road that extended more or less directly down the ridge, traversing the city gate at the point directly above the stadium where the remains of a tower stand. In fact, a modern road (which ultimately diverts towards the village of Argavlı in the south rather than towards the ancient city) still follows a similar route today (Fig.35). The prominence of this slope as seen from the city suggests that a procession traveling down this road from the mountain may even have been visible from the city below, as the model seems to indicate (Fig.32).[89]

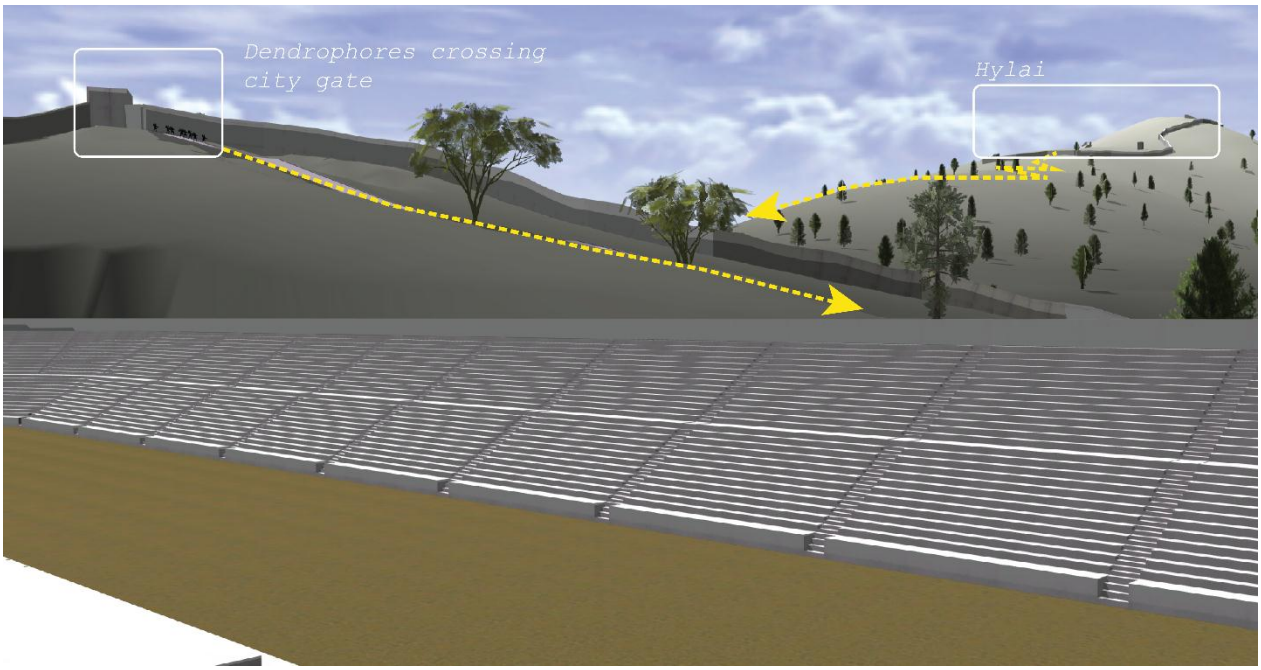

Fig. 32 View from the stadium of Hylai and the proposed route of the dendrophoroi. See Fig.35 (route a).

After passing the city gate at the tower, the *dendrophoroi* could have continued their descent down one of the hills that flank the stadium, which would have provided convenient seating for

[89] On the ancient road, see Keil, (1908, pp.166-7) and Thonemann (2011, pp. 103 – 104). The road is shown in the map by Phillipson (1936)

spectators. Perhaps they proceeded through the stadium itself (Fig.35, route b), though further excavation of the upper tiers of the sphendone would have to verify the feasibility of this. However they reached the foot of the stadium hill, from there it is less than 300 meters to the site of the presumed temple of Dionysos. If the procession took the road east of the stadium (Fig.35, route c), it may have traveled a road that passes by the conjectured site of the temple of Sarapis, and is on axis with both the city gymnasium and the temple to the north of the Lethaios. Possibly this area was less densely built up than the neighborhoods surrounding the city center, and thus more accommodating the transport of large trees and large crowds of spectators. As late as the mid-third century, if the inscription of the founding of the Dionysian cult is accurate, the site of the Dionysos temple was occupied by a grove of plane trees, suggesting that it was located in a quarter of the city that was undeveloped, or had ample room for gardens and orchards.

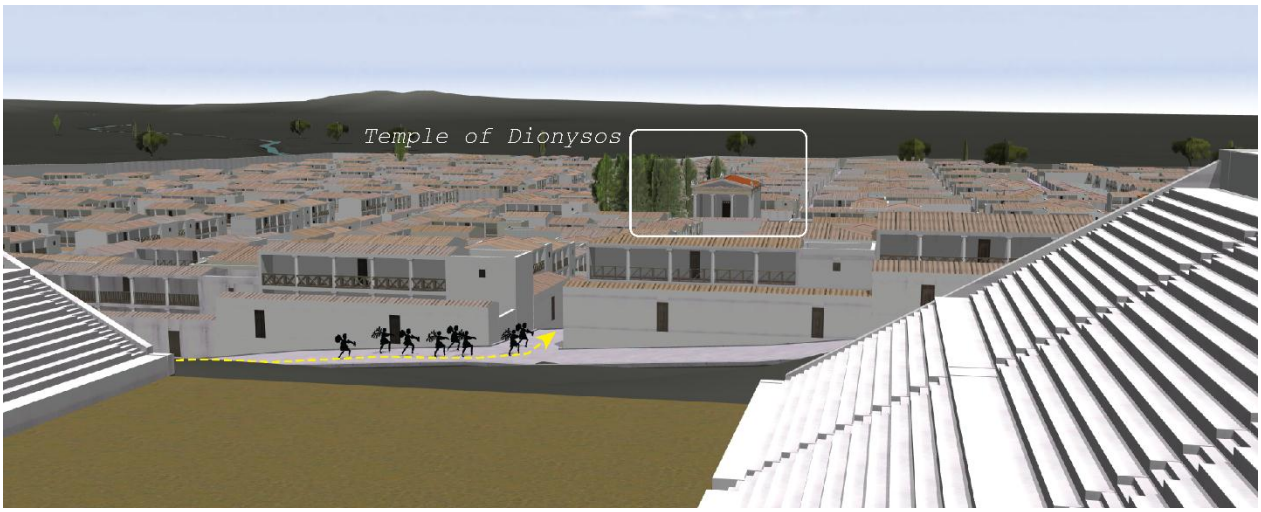

Fig. 33 View from the stadium of dendrophoroi approaching the temple of Dionysos. See Fig.35 (route a).

Fig. 34 View from the pronaos of the Dionysos temple of the approaching dendrophoroi. See Fig.35 (route a).

The position on a major east-west route of the finds that Humann identified as Dionysus' temple seems to indicate that the temple would have been sited on the street, oriented to the south, with a clear view of the tower and gate in the city wall beyond the stadium whence the *dendrophoroi* may have emerged during their descent into the city (Fig.34). As is the case of the alignment of the Artemis temple, here the question is not of human visibility as much as symbolic orientation in space. If the Dionysus temple did face south, it did so in distinction to Magnesia's other major temples (e.g. Zeus and Artemis), which faced west or south-west. The north-south alignment of the Dionyos temple would underscore the temple's urban embeddedness with the street grid, its character of being *kata ten polin*, "of the city".

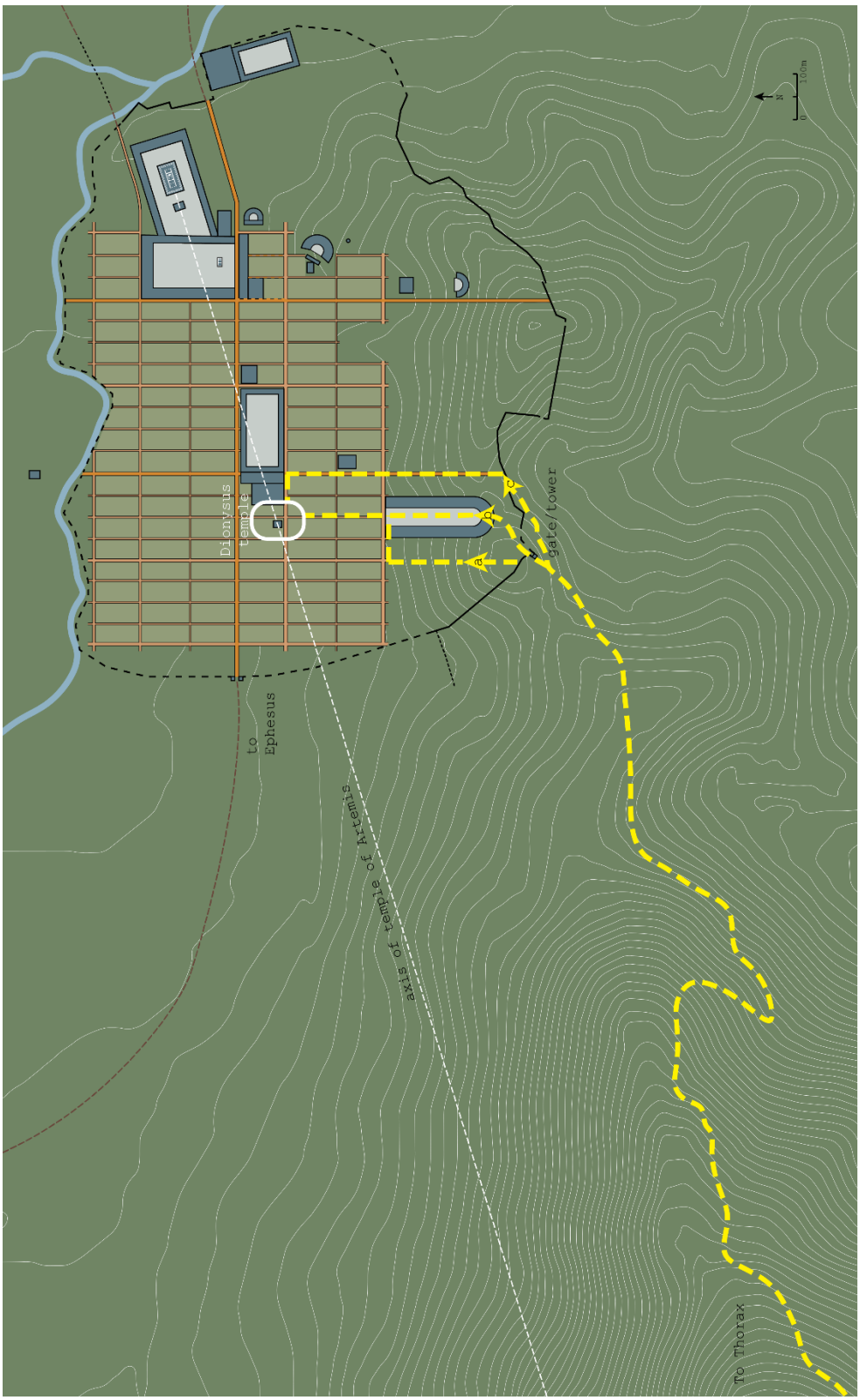

Fig. 35 Possible paths of the dendrophoroi into the city

5.2.2 Subverting the Grid: the Polytheistic Network

All of this activity occurs in the south-east quarter of the city, staying clear of the civic center represented by the agora, prytaneion, market basilica, and odeon, as well as the sanctuary dedicated to Artemis Leukophryne. In fact, when spatially mapped onto the topography and architecture of the city, the Apollonian-Dionysian duality seems transformed into a triad. The position of the temple of Dionysos acts as a mediator or buffer between the two facing poles of Apollo and Artemis – the siblings distinguished here by their diametrical opposition: one high, one low; one dwelling within a cave surrounded forest, one residing in a magnificent temple surrounded by stoas; one outside the city's walls, the other within its heart. From this perspective, the two seem to face each other, and between them lies the city of Magnesia, watched over by both gods. The 3D model shows that from the pronaos of the temple of Artemis, looking directly forward, one would have had a clear view of the site in the foothills of Thorax perhaps called Hylai, even with the altar of Artemis interposed (Fig.36). A line directly perpendicular to the front of the temple would bypass the site of 'Hylai' to the north, and in fact also misses the actual summit of Thorax (Fig.35). The orientation of the temple of Artemis towards the triangular peak where Hylai sits is therefore an illusion, yet visually the heights of Hylai dominate the skyline as seen from Magnesia (Fig.1), and it was probably perceived as part of the mountain, a visual stand-in for the actual, hidden peak. Though the cave is hidden, the mountain is visually present from every point in the city. The prominent visual framing of city as seen from the hilltop perch of Hylai completes the enfolding of the city within the purview of its protective deities (Fig.38).

Fig. 36 View of Thorax/Hylai from the pronaos of the Hellenistic temple of Artemis Leukophryne. Altar in middleground.

In the center of the city's grid, Dionysos stands between the two siblings. The axis of the temple of Artemis, although bypassing Thorax/Hylai, precisely intersects the location proposed by Humann for the Dionysos temple (Fig.35). Whether or not this was intentional, it does indicate that the spatial function of the temple should be viewed with consideration of its centrality within the city. The location of the thiasoi may give us further clues to Dionysos' role in the city's sacred polarities. The thiasos of Kosko, *platanistinoi*, as suggested above (Ch.3) might be sought at the location of the temple itself, where the plane tree of the original epiphany once stood. The second maenad, Baubo, is perhaps to be sought near the theatron, where the spring which sits above the fountain in the agora might indicate the location of Tabarnis. If the third thiasos of the maenad Thettale, the *kataivatai*, is in fact associated with the decent into a cave such as that of Apollo, then this might have been located in the village of Hylai. Together these form an elongated triangle that frames the stadium, and points to but does not penetrate the civic/religious core of the agora/Artemision (Fig.37). From this perspective, the dispersal of the cult of Dionysos in three thiasoi seems a strategic deployment. A triangular zone is created,

which ‘contains’ the wild and irrational behavior of the *dendrophoroi* so that it cannot disturb the rest of the city. The stadium, at the center of this triangle, may have been a focus of the activity.

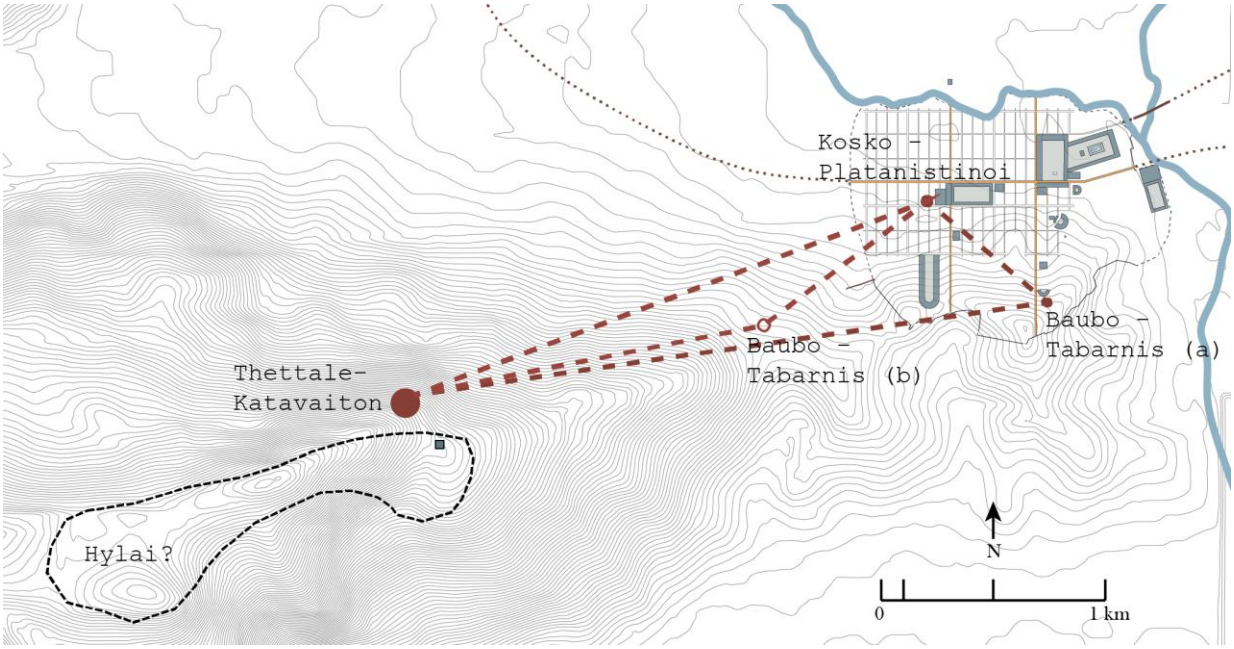

Fig. 37 Network of Dionysian thiasoi

5.3 Locating the Cave

A central goal of this study has been to locate the cave of Apollo at Hylai both physically and conceptually within Magnesia’s urban topography. Yet the fact that the actual cave has never been found may be beside the point in some ways. Caves are by nature hidden things, not visible monuments but secret places buried within the earth. Very few of the caves mentioned in ancient sources have been located, unless they were situated beneath or adjacent to a monumental building such as a temple that attracted archaeological attention. Given that the cave for the Greeks was a natural cave, usually associated with springs of water and the bees which were considered to be sacred to nymphs, the indigenous traditions of rock-cut architecture the first Greek settlers encountered in Anatolia, notably in Lycia and Phrygia, probably had little influence on the Greek idea of “caveness”. Anatolian rock-carvers created meticulously

articulated facades, but the interiors of these monuments, mostly intended as tombs, were dark and small. By contrast, while Greek architecture has sometimes been characterized by its externality (Zevi, 1957, pp.76-78), the cave for them was all about interiority, descending into darkness, going within.[90] It thus remained distinctly non-architectural, though never completely disassociated from architecture and civilization, but rather in dialogue with it.

Based on the combined testimony of the accounts of Pausanias and Damascius,[91] and the representation of *dendrophoroi* on the city's coins, we may safely assume that citizens of Magnesia would have been aware of the famous sanctuary of Apollo at Hylai, and that some individuals made pilgrimages there. Yet most residents probably did not know its exact location. The idea of the cave was in many ways more important than the physical presence of the cave itself. In relation to the city, it was a place both underground and high up, correlating to both sides of Apollo's nature. Spectators watching the approach of the *dendrophoroi* may have associated the ritual as much with Dionysos as with Apollo, because of the culmination of the journey at Dionysos' temple, and because the behavior of the *dendrophoroi* corresponds more to typical Dionysian ritual. The cave of Hylai, then, both in terms of its physical location and its associated practices, is hidden. It is unlikely that the place was on a regular pilgrimage route, because in contrast with most caves associated with Apollo, Hylai does not seem to have been an oracular site. The Magnesians were devoted to Pythian Apollo, traveling repeatedly all the way to Delphi to consult the oracle there. The existence of a local competitor would seem to be in conflict with this tradition. Nor does Hylai appear to have been a *charonium*, in the manner of Acharaca, Thymbria, or Hierapolis. Magnesia's necropolis lies in the plain well below the

[90] The Romans, who like the Anatolians came from a landscape of soft tufa and also had a tradition of rock-carving dating back to the Etruscans, combined aspects of interior space and exterior monumentality in their cave-spaces.
[91] See discussion in Chapter 3.1 above.

heights of Thorax, suggesting that the "steep precipices" and forests with large trees where the gardeners of Apollo dwelt were free of funerary associations. A possible connection with these caves may be that the geological activity that produced mephitic vapors in the *charonia* of the Maeander valley had an intoxicating effect that caused the "strength equal to any task" that inspired the *dendrophoroi*. However, in this case the devotees of Apollo were not inspired to prophecy, or healing as happened in the rite at the Plutonium of Acharaca. In the reverse of the procession of the Nyseans towards Acharaca, movement started at Hylai and was directed towards the city of Magnesia, with the ultimate goal of the temple of Dionysos. The cave of Apollo was therefore implicit in the three spatially dispersed outposts of this cult, the Dionysian *thiasoi*. Though visually and physically inaccessible, the cave was present in the city, permeating its rationalized grid by means of ritual movement and spectacle.

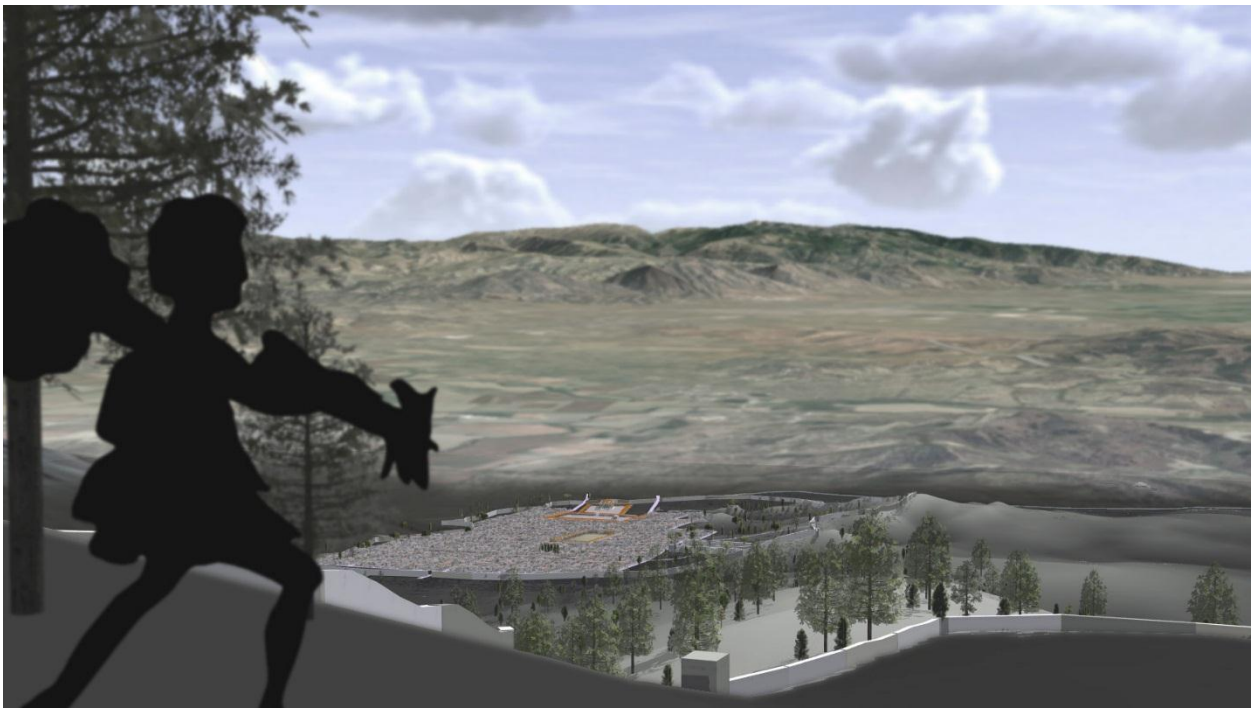

Fig. 38 View of the dendrophoroi towards the city from Hylai

It makes sense that Dionysos was the vehicle chosen for this infiltration by Apollo himself, if we recall the oracle that instituted the cult at Magnesia. Apollo had been at Hylai long before the founding of the city, yet he had no way of entering it, until Dionysos made his appearance and the maenads were installed. Pythian Apollo would naturally work in conjunction with Dionysos, with whom he shared the precinct of Delphi and the Corycian cave. The Magnesians, loyal to Apollo Pythios for his role in leading them to their new city, may have felt compelled to honor him by incorporating the ancient and strange cave of Hylai into their civic identity. This was not accomplished though architecture, since the cave was probably in too remote and precipitous a site to accommodate the building of a temple. Rather, they built a temple to Apollo's proxy Dionysos in the center of the city, at the spot which the epiphany helpfully indicated.

It is a unique solution but not entirely without context. Sacred caves abounded in Greece, including many that were associated with buildings and cities, but there was no single dominant architectural strategy for dealing with them. This was surely due in part to the inherent formlessness of caves themselves, which are part of nature and thus resisted humanizing classicism. Yet as Nilsson (1961) shows, the polytheistic meaning with which the natural world was imbued was not incompatible with design strategies meant to evoke a rational and democratic society. Rather, the two formed two overlapping matrices. As the experiments of the previous chapter show, the type of grid represented by Hippodamian planning is readily represented with line drawings, arithmetical calculations, and diagrams. The second type, however, more readily responds to the expanded analytical mode of 3D modeling. Reductive drawings showing axes and paths do not adequately visualize the polytheistic grid; this type of network was tied to the natural terrain and perceived from eye level as a human perceives the

landscape, a far cry from the totalizing bird's-eye view implied by the urban grid that is the purview of gods or those with god-like powers.

5.4 Conclusions

The goal of this project has been fourfold: First, to provide well-reasoned empirical propositions regarding the layout of the city of Magnesia at the time of its founding and through its development from the archaic to the Roman period. In addition to the reconstruction of the plan including streets and insulae, this involved an attempt to locate toponyms recorded in inscriptions and ancient historical sources that document religious activity in Magnesia. Second, this information was used to inform an investigation of the hypothesis that the interpenetration of ritual, architectural, and natural space constituted a set of overlapping 'polytheistic grids' that coexisted with the orthogonal grid of the city streets. Third, the role of the cave as represented by Apollo at Hylai was investigated. Last, a goal of this project has been to critique the digital tools and methodologies that were used, particularly GIS-based procedural modeling. The final paragraphs of this dissertation consist of an evaluation of this methodology and proposals for its future development.

In Chapter 4 above, arguments were presented for and against the use of procedural modeling. The methodology did prove advantageous in many ways: procedural modeling and use of the Roman City Ruleset made the holistic visualization of Magnesia's urban environment possible within the time frame and manpower resources of the project. It also aided the speculative modeling of many elements which were uncertain or hypothetical; the ability to generate many versions of the model underscored that it was to be viewed and treated as a research tool and not as an instance of 'scientific truth'. The goal of the model was to give spatial

presence to a range of possibilities, much as I argue the interaction of ritual and landscape must have taken shape within a range of configurations and not as a pre-designed, static diagram. Procedural modeling provided a framework for the incorporation of alternatives and the reasoning behind them. It also proved to be an effective data management system; attributes and variables stored in the geodatabase can be accessed and retrieved as the project develops. Urban landscapes are complex datasets, and GIS-based procedural models are an effective way of dealing with them.

The term ‘rules’ can be an unhelpful term for designating the scripts that generate procedural models. It tends to connote aspects of rigidity and positivism, understandably inspiring resistance from scholars who are less familiar with procedural modeling and raising concerns that uncertainty, context, and differentiation are being treated lightly. As I have argued above, this need not be the case, and I would prefer to term my digital work ‘hypotheses’ or ‘hypothetical definitions’ rather than ‘rules’, but the terminology is already in place. Further development of, and emphasis on, the epistemological implications of procedural modeling should help allay concerns of this sort. This means a more precise exposition of how the rules parse certainty vs. uncertainty, perhaps using the terms of the mathematical concept of fuzzy logic. A hurdle to be overcome here is the fact that although the procedural rules exhaustively document the decisions of the modeler with regard to uncertainty, the CGA shape grammar language in which it is couched prevents it from being used ‘raw’ in support of an argument. Until some means is devised to translate the code document to a medium (such as prose, diagrams, or other visualization) which is accessible to a scholarly audience, the whole point of procedural modeling, which is the documentation of process and sources, remains effectively moot.

Therefore, the question of interface design, which has not been raised in this project heretofore, must be addressed if procedural modeling is to be validated as a scholarly method. The writing of procedural rules is inherently an exercise which corresponds closely with the research, rather than presentation phase of a project. Yet the disciplinary imperatives of the Digital Humanities demand that final presentation in the case of 3D modeling must be addressed. A natural continuation for the work presented here would be to enter into the emerging practices and discourses surrounding digital publication and preservation of 3D content. This would encourage an even tighter weave between the digital content and the text, while ensuring that projects like this one have a chance of being vetted and cited by the academic community. The particular challenges and rewards of procedural modeling will only be acknowledged by scholars if the obstacles that stand between modeling and clear, well-designed digital publication is overcome. It is my hope that this research will become a test-case for developing standards for the publication of 3D arguments that will pave the way for Digital Humanists of the future.

Appendix A: Model Attributes

This section contains the attribute tables from the ArcGIS database that were used to generate the models. Attributes given in bold are documented by sources; the rest are conjectural. A review of the documentary evidence for each building is given in the footnotes. All other attributes and constants are default and/or internal to the rule and are documented in Appendix B.

Temples

Table 1 Temple attributes

	Temple of Artemis Leukophryne	**Roman Temple**	**Temple of Athena**	**Temple of Zeus Sosipolis**	**Temple of Dionysos**	**Archaic Temple of Artemis**
start_date	-203	150	-200	-200	-225	-500
end_date	300	300	300	300	300	-202
column_diameter	**1.4**	1	0.5	**0.69**	0.9	1.2
elevation	31	33	70.6	31.4	37.1	31
order_	**IONIC**	CORINTHIAN	IONIC	**IONIC**	IONIC	**IONIC**
steps	**7**	11	3	**2**	5	5
centerOpening_front	**1.33**					1.2
centerOpening_back	**1.33**					1.2
peristyle_type	**DIPTERAL**	NONE	**THOLOS**	**NONE**	NONE	PERIPTERAL
portico_style	**OCTASTYLE**	TETRASTYLE		**TETRASTYLE**	TETRASTYLE	**HEXASTYLE**
posticum_antis	**1**	0	0	**1**	0	0
pronaos_prostyle	**0**	1	0	**1**	1	0
pronaos_antis	**2**	0	0	**0**	1	1
pseudo	**TRUE**					
stair_type	**ALL_SIDES**	ARMS	ARMS	**ALL_SIDES**	ARMS	ALL_SIDES
stylobateL	**67**			**15.82**		
stylobateW	**41**			**7.38**		

Posticum_Door	**TRUE**					
pediment_windows	3	0	0	**0**	0	0

Porticus

Rule file: “Porticus.cga”

Table 2 Porticus attributes

Name	Artemision Stoa	Agora West Stoa	Agora NorthEast Stoa	Agora SouthEast Stoa	Propylon	Agora South Stoa
start_date	-200	-200	-200	-200	50	-200
end_date	300	300	300	300	300	300
building_type	STOA	STOA+SHOPS	STOA	STOA	PROPYLON	STOA+SHOPS
column_diameter	0.57	0.57	0.57	0.57	0.83	0.57
elevation	31	31	31	31	31	32
order_	DORIC	DORIC	DORIC	DORIC	IONIC	DORIC
steps	3	3	3	3	3	2
sides	6	5	0	0	0	0
stoaDepth	0	0	0	0	0	0
left	open	colonnade	open	open	open	wall
shapeType	LINEAR	LINEAR	SEGMENTED	SEGMENTED	SEGMENTED	SEGMENTED
right	open	open	open	colonnade	open	wall
segment			last	standalone	standalone	standalone
shopDepth	0	6	0	0	0	7.4
centerOpening_propylon	0	0	0	0	1.428	0
columnSpacing	2.5	2.5	2.5	2.5	2.5	2.5
propylon_style					HEXASTYLE	
back			wall	wall	colonnade	
backSteps					TRUE	

Table 3 Porticus attributes (continued)

Name	City Gymnasium Palaestra	Lethaios Gymnasium Palaestra	Agora West Gate	Agora East Gate	Classical Agora
start_date	200	200	-200	-200	-350
end_date	300	300	300	300	-201
building_type	PALAESTRA	PALAESTRA	PROPYLON	PROPYLON	STOA
column_diameter	0.57	0.57	0.57	0.57	0.57
elevation	36	29	31	31	31
order_	CORINTHIAN	CORINTHIAN	IONIC	IONIC	DORIC
steps	2	2	1	1	3
sides	0	0	0	0	6
stoaDepth	10	10	0	0	0
left			open	open	open
shapeType	OFFSET	OFFSET	SEGMENTED	SEGMENTED	LINEAR
right			open	open	open
segment			standalone	standalone	
shopDepth	0	0	0	0	0
centerOpening_propylon	0	0	1.2	1.2	0
columnSpacing	2.5	2.5	2	2	2.5
propylon_style			TETRASTYLE	TETRASTYLE	
back			colonnade	colonnade	wall
backSteps					

Mass Models

Table 4 Mass Models attributes

Name	start_date	end_date	building_type	elevation	height
Altar of Artemis Leukophryne	-250	300	ALTAR	31	5.27
Skene	-300	300	SKENE	36	15
City Gymnasium	200	300	APODYTERION	36	10.5
City Gymnasium	200	300	BATHS	36	10.5
Lethaios Gymnasium	200	300	APODYTERION	29	10.5
Lethaios Gymnasium	200	300	BATHS	29	10.5
Market Basilica	150	300	MARKET BASILICA	31	10
West Gate	-300	300	GATE	38	10
West Gate	-300	300	GATE	38	10
Tower	-300	300	TOWER	113	10
Southeast Gate	-300	300	GATE	26	10
Southeast Gate	-300	300	GATE	26	10
Northeast Gate	-300	300	GATE	28	10
Northeast Gate	-300	300	GATE	28	10
Tower	-300	300	TOWER	130	10
Southeast Bridge	-300	300	BRIDGE	22	5
Northeast Bridge A	-300	300	BRIDGE	24	5
Northeast Bridge B	-300	300	BRIDGE	24	6
North Bridge	-300	300	BRIDGE	27.61	5
Hylai/Leukophrys Tower	-600	300	TOWER	364	12

Appendix B: The Roman City Ruleset

This section contains excerpts from the code of the Roman City Ruleset. I have excluded those rules which were not used in the making of the Magnesia city model, as well as the less architecturally detailed rules, and the rules which manage textures and assets. In the interests of saving space and because the purpose of this section is primarily to demonstrate the logic of procedural model generation, I have not reproduced some of the attributes which control scene functions (such as level of detail, time periods, materials and colors). The body of each rule is complete.[92]

Temple

```
import col : "Colonnade.cga"
import tex : "TexturesAssets.cga"
import rf:   "Roof.cga"


#--------General

@Group("General",2)
attr generate_roof                    = true

@Group("General",2)@Range("TUSCAN","DORIC","IONIC","CORINTHIAN")
attr order_                                    = "DORIC"
       const tuscan                            = order_ == "TUSCAN"
       const doric                                    = order_ == "DORIC"
       const ionic                                    = order_ == "IONIC"
       const corinthian                        = order_ == "CORINTHIAN"

#--------Peristyle

@Group("Peristyle",3)@Range("NONE","PERIPTERAL","DIPTERAL","CLOSED_ALAE","T-SHAPE","THOLOS")
attr peristyle_type                   =       case tuscan: "NONE"
                                                      else: "PERIPTERAL"
       const noPeristyle                       = peristyle_type == "NONE"
       const peripteral                        = peristyle_type == "PERIPTERAL"
```

[92] If used in CityEngine, the missing attributes will result in error messages. The inclusion of procedural rules in this manuscript is meant as an argument for the code as a text to complement the argument presented in the chapters, and not as a fully functional and up-to-date release. The proper place for that is as part of the digital publication of this project, when I plan to make the complete rule files available for download. It should also be noted that the rules have not been fully annotated. That, too, is intended as part of the digital publication.

```
        const dipteral                      = peristyle_type == "DIPTERAL"
        const closedAlae                    = peristyle_type == "CLOSED_ALAE"
        const Tshape                        = peristyle_type == "T-SHAPE"
        const tholos                        = peristyle_type == "THOLOS"

@Group ("Peristyle",3)@Range("DISTYLE","TETRASTYLE","HEXASTYLE","OCTASTYLE","DECASTYLE")
attr portico_style                  =       case tuscan: "TETRASTYLE"
                                                         else: "HEXASTYLE"
        const distyle                       =       portico_style == "DISTYLE"
        const tetrastyle                    =       portico_style == "TETRASTYLE"
        const hexastyle                             =       portico_style == "HEXASTYLE"
        const octastyle                             =       portico_style == "OCTASTYLE"
        const decastyle                             =       portico_style == "DECASTYLE"

@Group ("Peristyle",3)
attr pronaos_prostyle         = 0

@Group ("Peristyle",3)
attr pronaos_antis            = 0

@Group ("Peristyle",3)
attr pronaos_prostyle_gap     = 0

@Group ("Peristyle",3)
attr posticum_prostyle        = 0

@Group ("Peristyle",3)
attr posticum_antis                 = 0

@Group ("Peristyle",3)
attr posticum_prostyle_gap    = 0

@Group ("Peristyle",3) @Range("true","false")
attr pseudo                                 = "false"

#--------Peristyle: Colonnade

@Group ("Peristyle","Colonnade",3)
attr column_diameter          =       1

@Group ("Peristyle","Colonnade",3)
attr centerOpening_front      =       1 #factor of columnSpacing

@Group ("Peristyle","Colonnade",3)
attr centerOpening_back               =       1

@Group ("Peristyle","Colonnade",3)
attr antisColumns_front               = true

@Group ("Peristyle","Colonnade",3)
attr antisColumns_back        = true

@Group ("Peristyle","Colonnade",3)@Range("5","10","20")
```

```
attr tholosColonnadeSpacing   = "10"

const pilasters                                =       case tuscan: false
                                                              else: false
const columnHeight                    =       col.columnHeight
const peristyleH                      =       columnHeight+entablatureH
const columnOffset                    =       (col.baseW*column_diameter-column_diameter)/2

#--------Entablature

const entablatureH                    =       column_diameter*col.entablatureH
const architraveD                     =       column_diameter
#--------Cella

@Group ("Cella",4)@Range("single","triple")
attr cella_type                               = "single"
       const oneCella                         = cella_type == "single"
       const threeCellae                      = cella_type == "triple"

@Group ("Cella",4)
attr cellaW                                   = 1.5

@Group ("Cella",4)@Range("true","false")
attr Posticum_Door                    = "false"

const mainDoorHeight          = columnHeight*.5
const Door_Height                     = mainDoorHeight*.8
const wallThickness           = column_diameter
const doorD                                   = 0.4
const doorWindowH                     = mainDoorHeight*0.15
const doorFrameW                      = column_diameter*.25
const doorFrameH                      = column_diameter*.5
const entranceFrameW          = doorFrameW+0.1
const entranceH                               =
Door_Height+doorWindowH+doorFrameH+entranceFrameW+0.5*step_height
const entranceW                               = 2*doorFrameW+2*entranceFrameW
const mainEntranceH                   =
mainDoorHeight+doorWindowH+doorFrameH+entranceFrameW+0.5*step_height
const mainEntranceW                   = 2*doorFrameW+2*entranceFrameW
const corniceProjection       = column_diameter *0.1
const cellaSetback                    = (col.baseW - column_diameter)/2

#--------Podium

@Group ("Podium",5)@Range("FRONT","FRONT_NARROW","ARMS","SIDE","SIDE+FRONT","ALL_SIDES")
attr stair_type                               =       case tuscan || corinthian: "ARMS"
                                                              else: "ALL_SIDES"
       const front                                    = stair_type == "FRONT"
       const frontNarrow                      = stair_type == "FRONT_NARROW"
       const arms                                     = stair_type == "ARMS"
       const side                                     = stair_type == "SIDE"
       const both                                     = stair_type == "SIDE+FRONT"
       const allSides                         = stair_type == "ALL_SIDES"
```

```
@Group ("Podium",5)
attr steps                                  = 5

@Group ("Podium",5)
attr resizeLot                        = false

@Group ("Podium",5)
attr stylobateL                             = 30

@Group ("Podium",5)
attr stylobateW                             = 20

@Group ("Podium",5)
attr extra_steps_down         = 0

@Group ("Podium",5)
attr step_height                      = 0.28

@Group ("Podium",5)
attr armW                                   = case tholos: column_diameter
                                                    else: column_diameter*3


@Group ("Podium",5)
attr armH                                   = steps*step_height

const frontSteps                      = steps/2
const baseH                                 = steps*step_height
const treadDepth                      = 0.4
const landingDepth                    = column_diameter*2

#-------Roof

@Group("Roof",6)
attr roof_angle                       =       case tuscan: 19
                                                      case ionic: 13.6
                                                            else: 15
@Group("Roof",6)
attr Troof_angle                      = case tuscan: 12
                                                            else: 15

@Group("Roof",6)@Range("true","false")
attr antefix                          = "false"

@Group("Roof",6)
attr pediment_windows         = 0


const roofBrickW                      = column_diameter
const roofBrickH                      = column_diameter*1.25
const roofW                                 = 1.1
```

```
#------Texture Size

const tile                                  = columnHeight


#------Index

firstColumn(side)                       = split.index == 0                        && comp.index == 0
               && comp.sel == side
lastColumn(side)                        = split.index == split.total-1 && comp.index == comp.total-
1 && comp.sel == side
boundaryColumnR                                 = lastColumn("left") ||  lastColumn("right")
boundaryColumnL                                 = firstColumn("left") || firstColumn("right")


////////////
///////START

Lot -->
case !noModel:
        t(0,elevation,0)
        print(scope.sx)
        print(scope.sz)
        Stylobate(scope.sx,scope.sz)
else: NIL

Stylobate(x,z) -->
        case resizeLot == true:
                innerRect
                s(stylobateW,0,stylobateL)
                center(xz)
                Stylobate1
        case tholos:
                Tholos
        else:
                innerRect
                s(x,0,z)
                center(xz)
                Stylobate1

Stylobate1 -->
        color(base)
        Temple
        t(0,steps*step_height,0)
        color(wall)
        Cella(scope.sx)

///////THOLOS

Tholos -->
s('1,baseH+peristyleH,'1)
i(tex.cylinderAsset)
```

```
split(y){baseH:color(base)comp(f){all:TholosBase(comp.index,comp.total)}
                |columnHeight:color(column)comp(f){all:TholosColonnade(comp.index,comp.total)}
                |entablatureH:
comp(f){side:setPivot(xyz,5)col.Entablature(scope.sx,column_diameter)}
                }


TholosBase(n,y) -->
        case n == 0:setPivot(xyz,5) FrontStairs(scope.sx)
        case n == y-1: [ tex.Block("wall",tile)]offset(-col.columnSpacing/2,
inside)extrude(peristyleH) comp(f){all:TholosCella(comp.index,comp.total)}
        else: tex.Block("wall", tile)


TholosColonnade(n,y) -->
        case n == 0: setPivot(xyz,5)split(x){~1:rotate(abs,pivot,0,-18,0)t(0,0,-
column_diameter/4)col.ColumnTile(1,column_diameter,split.total)
                                                        |~1:NIL}
        case n == y-1: t(0,0,entablatureH)[reverseNormals
tex.Block("wall",tile)]color(roof)roofHip(roof_angle)comp(f){top:
rf.TempleRoof(col.columnSpacing,roofBrickW,roofBrickH)}


        case n < 10 && tholosColonnadeSpacing == "20":
setPivot(xyz,5)split(x){~1:rotate(abs,pivot,0,-18,0)t(0,0,-
column_diameter/4)col.ColumnTile(1,column_diameter,split.total)

        |~1:col.ColumnTile(1,column_diameter,split.total)}
        case n < 10 && tholosColonnadeSpacing == "10": setPivot(xyz,5)rotate(abs,pivot,0,-
18,0)t(0,0,-column_diameter/4)col.ColumnTile(1,column_diameter,split.total)
        case n < 10 && tholosColonnadeSpacing == "5" && n%2 == 0:
setPivot(xyz,5)rotate(abs,pivot,0,-18,0)t(0,0,-
column_diameter/4)col.ColumnTile(1,column_diameter,split.total)
        else: NIL



TholosCella(n,y) -->
        case n <2: NIL
        case n == 2:split(x){~1:Wall
                                                |mainEntranceW+n*0.5:
split(y){mainEntranceH:Entrance| entranceFrameW : Cornice|~1:Wall}
                                                |~1: Wall}
        else: Wall

///////PODIUM


Temple -->
        case arms :
                s(scope.sx+column_diameter*2,'1,'1)
                center(x)
                Temple1


        case noPeristyle || Tshape:
                s(scope.sx+column_diameter,'1,scope.sz+column_diameter)
                center(xz)
                Temple1


        case peripteral || dipteral:
                offset(cellaSetback, inside)
```

```
                Temple1

        else:

                Temple1


Temple1 -->
        StairsAround(steps-1)
        extrude(steps*step_height+extra_steps_down*step_height)
        t(0,-extra_steps_down*step_height,0)
        Stairs

Stairs -->
        case allSides:
        NIL
        else:
        split(z) {~1: Podium | .001: FrontStairs(scope.sx)}

Podium -->
        tex.Block("wall", tile, tile)

StairsAround(n) -->
        case n >= 0 && allSides :
                [ t(0,step_height,0) StairsAround(n-1) ]
                offset(n*treadDepth, inside) extrude(step_height) tex.Block("wall",tile)
        else :
                NIL

FrontStairs(n) -->

        case front:
                Steps(steps+extra_steps_down)

        case frontNarrow:
                s (column_diameter*4,'1,'1)
                center(x)
                Steps(steps+extra_steps_down)

        case arms:
                split(x)        {armW:  s('1,'1,steps*treadDepth)
                                                i("builtin:cube")
                                                Arm
                                        | ~1:   Steps(steps+extra_steps_down)
                                        |armW:  s('1,'1,steps*treadDepth)
                                                i("builtin:cube")
                                                Arm}
        case side:
                i("builtin:cube")
                s(scope.sx-steps*treadDepth*2,'1,landingDepth)
                center(x)
                setPivot(xyz,2)
                comp(f)         {left: Steps(steps+extra_steps_down)
```

```
                                    | right: Steps(steps+extra_steps_down)
                                    | back: tex.Block("wall", tile)
                                    | top: tex.Block("wall", tile)}

        case both:
                split(x)        {armW /*~1*/ : s('1,'1,frontSteps*treadDepth)
                                                i("builtin:cube")
                                                comp(f) {all: tex.Block("wall", tile)}
                                    |~1 /*n-steps+frontSteps*treadDepth*2*/ :
        s('1,'1,landingDepth+frontSteps*treadDepth)

                i("builtin:cube")

                center(x)

                setPivot(xyz,2)

                comp(f) {left: split(x)       {scope.sx-frontSteps*treadDepth: split(y)
        {(steps+extra_steps_down)*step_height-
(frontSteps*step_height):Steps(steps+extra_steps_down-frontSteps)


                                            |~1: NIL}

                                                        |~1:NIL}

                                |right: split(x){~1:NIL

                                                        |scope.sx-frontSteps*treadDepth:
split(y)       {(steps+extra_steps_down)*step_height-
(frontSteps*step_height):Steps(steps+extra_steps_down-frontSteps)


                                            |~1: NIL}}

                                |back: split(y)        {~1:tex.Block("wall", tile)

                                                        |frontSteps*step_height:      t(0,0,-
landingDepth-frontSteps*treadDepth)


                set(trim.vertical,false)


                Steps(frontSteps)}

                                |top: split(y) {~1:t(0,0,-frontSteps*step_height)

                                                                tex.Block("wall", tile)

                                                        |frontSteps*treadDepth:NIL}

                                }
                                    |armW /*~1*/:s('1,'1,frontSteps*treadDepth)
                                            i("builtin:cube")
                                            tex.Block("wall", tile)
                                    }

        else: tex.Block("wall", tile)
```

```
Arm -->
        case scope.sy>armH+step_height:
        split(y){~armH:tex.Block("wall",tile)
                        |~armH:s('1,'1,(scope.sy/step_height)*treadDepth)
                                        tex.Block("wall",tile)
                        }
        else: tex.Block("wall",tile)

Steps(n) -->
         split(y)       {step_height: Step(n)
                                | ~1: Steps(n-1)}

Step(idx) -->
        s('1,'1,scope.sz+idx*treadDepth)
        i("builtin:cube")
        tex.Block("wall", tile)



///////CELLA

Cella(X) --> # gets column spacing taking into account the portico style and any irregularities
in the intercolumniation.
        case tetrastyle || distyle:
                CellaOffset((X-((X/3)*centerOpening_front))/2,scope.sx)
                Peristyle((X-((X/3)*centerOpening_front))/2)


        case hexastyle:
                CellaOffset((X-((X/5)*centerOpening_front))/4,scope.sx)
                Peristyle((X-((X/5)*centerOpening_front))/4)



        case octastyle :
                CellaOffset((X-((X/7)*centerOpening_front))/6,scope.sx)
                Peristyle((X-((X/7)*centerOpening_front))/6)


        case decastyle:
                CellaOffset((X-((X/9)*centerOpening_front))/8,scope.sx)
                Peristyle((X-((X/7)*centerOpening_front))/8)


        else:
                NIL

Peristyle(n) --> # tuscan style has a different column spacing on the sides
        case tuscan && noPeristyle: Peristyle(n,(scope.sz/2)/pronaos_prostyle+pronaos_antis)
        else: Peristyle(n,n)
```

```
CellaOffset(n,xDim) --> #finds the footprint of the cella walls and roof

        case peripteral || closedAlae:
                [t(0,peristyleH,0)Roof(n,scope.sz)]
                s(scope.sx-n*2+column_diameter,'1,'1)
                center(xz)
                Cella1(n,xDim)


        case dipteral:
                [t(0,peristyleH,0)Roof(n,scope.sz)]
                s(scope.sx-n*4+column_diameter,'1,'1)
                center(xz)
                Cella1(n,xDim)

        else:
                Cella1(n,xDim)
                t(0,peristyleH,0)
                Roof(n,scope.sz)

Cella1(n,xDim) -->

        case closedAlae :
                extrude(peristyleH)
                split(y){~1:Cella2(n,n,xDim)|entablatureH:
                                                                                comp(f){left:
Entablature(n)

        |right: Entablature(n)

        |bottom:offset(-architraveD/2,inside) InnerCeiling}}

        case peripteral: extrude(peristyleH)
                                        split(y){~1:Cella2(n,n,xDim)|entablatureH:
s('1,'1,scope.sz-n*2)

        center(z)

        comp(f){front:EntablatureFront(n)

                |left: Entablature(n)

                |right: Entablature(n)

                |back: Entablature(n)

                |bottom:offset(-architraveD/2,inside)InnerCeiling

                t(0,0,-entablatureH)reverseNormals InnerCeiling}}

        case noPeristyle && !tuscan || Tshape:

                extrude(peristyleH)
                split(y){~1:Cella2(n,n,xDim)|entablatureH: comp(f){side:NIL
```

```
        |bottom:offset(-architraveD/2,inside)InnerCeiling

        t(0,0,-entablatureH)reverseNormals InnerCeiling}}



         case noPeristyle && tuscan :

                extrude(peristyleH)

        split(y){~1:Cella2(n,(scope.sz/2)/pronaos_prostyle+pronaos_antis,xDim)|entablatureH:
comp(f){side:NIL

        |bottom:offset(-architraveD/2,inside)InnerCeiling

        t(0,0,-entablatureH)reverseNormals InnerCeiling}}



        case dipteral && pseudo == "true":
                extrude(peristyleH)

                split(y){~1:Cella2(n,n,xDim)|entablatureH:  s('1,'1,scope.sz-n*4)

        center(z)

        comp(f){front: s(xDim,'1,'1)center(x)EntablatureFront(n)

                |back: s(xDim,'1,'1)center(x)EntablatureFront(n)

                |left: Entablature(n)

                |right: Entablature(n)

                |bottom:offset(-architraveD/2,inside)InnerCeiling

        t(0,0,-entablatureH)reverseNormals InnerCeiling}}

        else:

                extrude(peristyleH)
                split(y){~1:Cella2(n,n,xDim)|entablatureH: NIL}

Cella2(n,o,xDim) -->
case dipteral && pseudo == "true" && pronaos_antis >=1 && posticum_antis == 0:
        split(z){~o:Cella3(n,o,split.index,split.total, pronaos_prostyle+pronaos_antis*2+1,
posticum_prostyle,xDim)
                |{~o:Cella3(n,o,split.index,split.total, pronaos_prostyle+pronaos_antis*2+1,
posticum_prostyle,xDim)}*
                |.01:Cella3(n,o,split.index,split.total, pronaos_prostyle+pronaos_antis*2+1,
posticum_prostyle,xDim)}

case dipteral && pseudo == "true" && posticum_antis >=1 && pronaos_antis == 0:
        split(z){~o:Cella3(n,o,split.index,split.total, pronaos_prostyle,
posticum_prostyle+posticum_antis*2+1,xDim)
```

```
                |{~o:Cella3(n,o,split.index,split.total, pronaos_prostyle,
posticum_prostyle+posticum_antis*2+1,xDim)}*
                |.01:Cella3(n,o,split.index,split.total, pronaos_prostyle,
posticum_prostyle+posticum_antis*2+1,xDim)}

case dipteral && pseudo == "true" && posticum_antis >=1 && pronaos_antis >= 1:
        split(z){~o:Cella3(n,o,split.index,split.total, pronaos_prostyle+pronaos_antis*2+1,
posticum_prostyle+posticum_antis*2+1,xDim)
                |{~o:Cella3(n,o,split.index,split.total, pronaos_prostyle+pronaos_antis*2+1,
posticum_prostyle+posticum_antis*2+1,xDim)}*
                |.01:Cella3(n,o,split.index,split.total, pronaos_prostyle+pronaos_antis*2+1,
posticum_prostyle+posticum_antis*2+1,xDim)}

else:
        split(z){~o:Cella3(n,o,split.index,split.total, pronaos_prostyle+pronaos_antis,
posticum_prostyle+posticum_antis,xDim)
                |{~o:Cella3(n,o,split.index,split.total, pronaos_prostyle+pronaos_antis,
posticum_prostyle+posticum_antis,xDim)}*
                |.01:Cella3(n,o,split.index,split.total, pronaos_prostyle+pronaos_antis,
posticum_prostyle+posticum_antis,xDim)}



Cella3(n,o,b,k,p,r,xDim)-->

#parameters:
#n = column spacing (front)
#o = column spacing (side)
#b = split.index
#k = split.total
#p = number of rows in front
#r = number of rows in back
#xDim = stylobate width

        case peripteral && b == 0
                || dipteral && b == 0

                :NIL
        case peripteral && b == k-1
                || dipteral && b == k-1

                :NIL
        case peripteral && b-r == 1
                || dipteral && b-r == 1

                :CellaWalls(n,o,k,p,r)
        case peripteral && b>= k-p-1 && b<k-1
                || dipteral && b>= k-p-1 && b<k-1
                || closedAlae && b>= k-p-1 && b<k-1
                :PorchRows(n,b,k,p,r,1,xDim)
        case peripteral && b-r<1
                || dipteral && b-r<1

                :PorchRows(n,b,k,p,r,2,xDim)
        case closedAlae && b == 0
                :CellaWalls(n,o,k,p,r)
        case noPeristyle && b-r ==0 || Tshape && b-r ==0
```

```
            :CellaWalls(n,o,k,p,r)
        case noPeristyle && b>=k-p && b<k-1 || Tshape && b>=k-p && b<k-1
            :t(0,0,column_diameter/2)
            PorchRows(n,b,k,p,r,1,xDim)
        case noPeristyle && b == k-1 || Tshape && b == k-1
            :PorchRows(n,b,k,p,r,1,xDim)
        case noPeristyle && b-r < 0 || Tshape && b-r < 0
            :PorchRows(n,b,k,p,r,2,xDim)

        else: NIL



CellaWalls(n,o,k,p,r) -->

        case closedAlae:
            s('1,'1,scope.sz*(k-2-p))
            comp(f) {front: CellaFront(n,o)
                    |left:  CellaSide(n)
                    |right: CellaSide(n)
                    |back:  CellaBack(n,o)}

        case peripteral || dipteral :
            s('1,'1,scope.sz*(k-3-p-r))

            comp(f) {front: CellaFront(n,o)
                    |left:  CellaSide(n)
                    |right: CellaSide(n)
                    |back:  CellaBack(n,o)}

        case Tshape:
            s('cellaW,scope.sy+(steps*step_height)+entablatureH,scope.sz*(k-1-p-r))
            center(x)
            t(0,-baseH,0)
            split(y){baseH: color(base) tex.Block("wall",tile)
                            |~1: comp(f){front:    CellaFront(n,o)
                                                   |left:  CellaSide(n)
                                                   |right: CellaSide(n)
                                                   |back:  CellaBack(n,o)}
                            |entablatureH: comp(f){side:Entablature(n)
                                                                  |top: TRoof(n)}
                            }


        else:
            s('1,'1,scope.sz*(k-1-p-r))
            comp(f) {front: CellaFront(n,o)
                    |left:  CellaSide(n)
                    |right: CellaSide(n)
                    |back:  CellaBack(n,o)}
```

```
CellaFront(n,o) -->
               case pronaos_antis >=1 && dipteral && pseudo == "true":
                      split(x){wallThickness: s('1,'1,pronaos_antis*o*2-column_diameter/2)
                                                                  i("builtin:cube")
                                                                  tex.Block("wall", tile)
                                    |~1: EntranceWall(n)
                                    |wallThickness: s('1,'1,pronaos_antis*o*2-
column_diameter/2)
                                                                  i("builtin:cube")
                                                                  tex.Block("wall", tile)}
               case pronaos_antis >=1 && dipteral && pseudo == "false"
               ||  pronaos_antis >=1 && !dipteral:
                      split(x){wallThickness: s('1,'1,pronaos_antis*o)
                                                                  i("builtin:cube")
                                                                  tex.Block("wall", tile)
                                    |~1: EntranceWall(n)
                                    |wallThickness: s('1,'1,pronaos_antis*o)
                                                                  i("builtin:cube")
                                                                  tex.Block("wall", tile)}
               else:
                      EntranceWall(n) Pilasters(n)

EntranceWall(n) -->

       case scope.sx > (mainEntranceW+n*.5) && oneCella || scope.sx <
mainEntranceW+n*.5+(entranceW+n*.4)*2 && threeCellae && scope.sx >= (mainEntranceW+n*.5)  :
               split(x){ ~1                  : Wall
                               | mainEntranceW+n*.5 : split(y){ mainEntranceH : Entrance |
entranceFrameW : Cornice| ~1 :Wall }
                               | ~1           : Wall }
       case scope.sx > mainEntranceW+n*.5+(entranceW+n*.4)*2 && threeCellae && hexastyle ||
tetrastyle || distyle:
               s('1,'1,'1) center(x)
               split(x){ ~1                                                  : Wall
                               | ~n           : split (x) {~1: Wall | entranceW+n*.4: split(y){
entranceH : Entrance| entranceFrameW : Cornice | ~1 : Wall } | ~1: Wall}
                               |~n*centerOpening_front: split(x){~1: Wall| mainEntranceW+n*.5:
split(y){ mainEntranceH : Entrance | entranceFrameW : Cornice| ~1 : Wall} | ~1: Wall}
                               | ~n           : split (x) {~1: Wall | entranceW+n*.4: split(y){
entranceH : Entrance| entranceFrameW : Cornice | ~1 : Wall } | ~1: Wall}
                               | ~1                                          : Wall }

       case scope.sx > mainEntranceW+n*.5+(entranceW+n*.4)*2 && threeCellae && octastyle ||
decastyle:
               s('1,'1,'1) center(x)
               split(x){ ~1                                                  : Wall
                               | ~n           : split (x) {~1: Wall | entranceW+n*.4: split(y){
entranceH : Entrance | entranceFrameW : Cornice| ~1 : Wall } | ~1: Wall}
                               |~n: Wall
                               |~n*centerOpening_front: split(x){~1: Wall| mainEntranceW+n*.5:
split(y){ mainEntranceH : Entrance | entranceFrameW : Cornice | ~1 : Wall} | ~1: Wall}
                               |~n: Wall
                               | ~n           : split (x) {~1: Wall | entranceW+n*.4: split(y){
entranceH : Entrance | entranceFrameW : Cornice | ~1 : Wall } | ~1: Wall}
                               | ~1                                          : Wall  }
```

```
        else:
                Wall

Entrance -->
        //t(0,0,- treadDepth *0.2)
        split(y){ step_height *0.5    : t(0,0,treadDepth*0.5) Wall( wallThickness+treadDepth)
                        | mainEntranceH                                                 : split(x){
entranceFrameW : Wall | ~1 : DoorMain | entranceFrameW : Wall }
                        }

DoorMain -->
        color(wood)
        t(0,0,-wallThickness*0.15)
        split(y){ ~1          : split(x){ doorFrameW : DoorFrame(wallThickness*0.8) | ~1 : Door |
doorFrameW : DoorFrame(wallThickness*0.8) }
                        | doorFrameH  : Wall(wallThickness*0.8) DoorCornice
                        | doorWindowH : DoorWindows }

Door -->
        s('1,'1,doorD)
        t(0,0,'-1)
        i(tex.doorAsset)
        tex.Block("wood")

DoorCornice -->
        s('1,'1,corniceProjection)
        i(tex.doorCorniceAsset)
        tex.Block("wood")

Cornice -->
        s('1,'1,corniceProjection)
        i(tex.doorCorniceAsset)
        tex.Block("block")

DoorWindows -->
        s('1,'1,doorD*0.3)
        t(0,0,'-1)
        split(x){ ~scope.sy :  i(tex.topdoorAsset)
                                                        tex.Block("wood")}*

CellaSide(n) -->
                Wall

CellaBack(n,o) -->
        case posticum_antis >=1 && Posticum_Door == "true" && !closedAlae && !dipteral
        || posticum_antis >=1 && Posticum_Door == "true" && !closedAlae && dipteral && pseudo ==
"false":
                split(x){wallThickness: extrude(posticum_antis*o)
                                                                tex.Block("wall", tile)
                                |~1: EntranceWall(n)
                                |wallThickness: extrude(posticum_antis*o)
```

```
                                                        tex.Block("wall", tile)}
        case posticum_antis >=1 && Posticum_Door == "true" && !closedAlae &&  dipteral && pseudo
== "true" :
                split(x){wallThickness: extrude(posticum_antis*o*2-column_diameter/2)
                                                        tex.Block("wall", tile)
                                |~1: EntranceWall(n)
                                |wallThickness: extrude(posticum_antis*o*2-column_diameter/2)
                                                        tex.Block("wall", tile)}

        case posticum_antis >=1 && Posticum_Door == "false" && !closedAlae && !dipteral
        || posticum_antis >=1 && Posticum_Door == "false" && !closedAlae && dipteral && pseudo ==
"false":
                split(x){wallThickness: extrude(posticum_antis*o)
                                                        tex.Block("wall", tile)
                                |~1: Wall
                                |wallThickness: extrude(posticum_antis*o)
                                                        tex.Block("wall", tile)}
                case posticum_antis >=1 && Posticum_Door == "false" && !closedAlae &&  dipteral &&
pseudo == "true":
                split(x){wallThickness: extrude(posticum_antis*o*2)
                                                        tex.Block("wall", tile)
                                |~1: Wall
                                |wallThickness: extrude(posticum_antis*o*2)
                                                        tex.Block("wall", tile)}
        case posticum_antis == 0 && Posticum_Door == "true":
                EntranceWall(n)
        case closedAlae:
                s(scope.sx+n*2-column_diameter,'1,'1) center(x)
                set(trim.vertical, false)
                Wall
        else:
                Wall


DoorFrame(thickness)-->
        s('1,'1,thickness)
        t(0,0,'-1)
        i("builtin:cube")
        tex.Block("wood")

Wall -->
        Wall(wallThickness)


Wall(thickness) -->
        s('1,'1,thickness) t(0,0,'-1) i("builtin:cube")
        tex.Block("wall", tile)




///////PERISTYLE

Peristyle(n,o) -->
```

```
        case dipteral && pseudo == "false":
                extrude(columnHeight+entablatureH)
                split(y){~1: Peristyle2(n) | entablatureH: InnerEntablature(n) comp(f){front:
EntablatureFront(n)
                                                        |left: Entablature(o)
                                                        |right:Entablature(o)
                                                        |back:EntablatureFront(n)
                                                        |bottom:offset(-architraveD/2-
n,inside)InnerCeiling
                                                        t(0,0,-entablatureH)reverseNormals
InnerCeiling}  }

        else:
                extrude(columnHeight+entablatureH)
                split(y){~1: Peristyle2(n) | entablatureH: comp(f){front: EntablatureFront(n)
                |left: Entablature(o)
                |right:Entablature(o)
                |back:EntablatureFront(n)
                |bottom:  Ceiling}}
Peristyle2(n) -->
        case closedAlae :
                comp(f) {front: col.FrontColonnade(n,centerOpening_front)
                                |left:  col.NoFirstLastFlush(n)
                                |right: col.FirstFlushNoLast(n)
                                }

        case peripteral || dipteral && pseudo == "true":
                comp(f) {front: col.FrontColonnade(n,centerOpening_front)
                                |left:  col.FirstLastFlush(n)
                                |right: col.FirstLastFlush(n)
                                |back:  col.FrontColonnade(n,centerOpening_back)
                                }

        case dipteral && pseudo == "false":
                comp(f) {front: col.FrontColonnade(n,centerOpening_front)    t(0,0,-
n+column_diameter-columnOffset*2) col.InnerFrontColonnade(n,centerOpening_front)
                                |left:  col.FirstLastFlush(n) InnerColonnade(n)
                                |right: col.FirstLastFlush(n) InnerColonnade(n)
                                |back:  col.FrontColonnade(n,centerOpening_back)    t(0,0,-
n+column_diameter-columnOffset*2) col.InnerFrontColonnade(n,centerOpening_back)
                                }


        else: NIL

InnerColonnade(n) -->
        t(0,0,-n+column_diameter-columnOffset*2)
```

```
        col.NoFirstNoLast(n)



InnerEntablature(n) -->
        s(scope.sx-n*2+column_diameter,'1,scope.sz-n*2+column_diameter)
        center(xz)
        comp(f){front: EntablatureFront(n)
                       |left: Entablature(n)
                       |right:Entablature(n)
                       |back: Entablature(n)}

#parameters:
#n = column spacing (front)
#o = column spacing (side)
#b = split.index
#k = split.total
#p = number of rows in front
#r = number of rows in back
#xDim = stylobate width

PorchRows(n,b,k,p,r,a,xDim) -->
        case dipteral && pseudo == "true" && a == 1 && pronaos_antis>=1 && b<k-p-
1+pronaos_antis*2 && b%2 == 0: AntisRow(n,a,xDim)
        case dipteral && pseudo == "true" && a == 1 && pronaos_antis>=1 && b<k-p-
1+pronaos_antis*2 && b%2 == 1: NIL
        case dipteral && pseudo == "true" && a == 1 && pronaos_antis>=1 && b== k-p-
1+pronaos_antis*2 : NIL
        case dipteral && pseudo == "true" && a == 2 && posticum_antis>=1 && b-
r+posticum_antis*2>=1 && b%2 == 0: AntisRow(n,a,xDim)
        case dipteral && pseudo == "true" && a == 2 && posticum_antis>=1 && b-
r+posticum_antis*2>=1 && b%2 == 1: NIL
        case dipteral && pseudo == "true" && a == 2 && posticum_antis>=1 && b-
r+posticum_antis*2<1 : NIL
        case dipteral && pseudo == "false" && a == 1 && pronaos_antis>=1 && b<k-p+pronaos_antis
                || !dipteral && a == 1 && pronaos_antis>=1 && b<k-p+pronaos_antis
                || Tshape  && a == 1 && pronaos_antis>=1 && b<k-p+pronaos_antis:
AntisRow(n,a,xDim)
        case dipteral && pseudo == "false" && a == 2 && posticum_antis>=1 && b-
r+posticum_antis>=0
               || !dipteral && a == 2 && posticum_antis>=1 && b-r+posticum_antis>=0
               || Tshape && a == 2 && posticum_antis>=1 && b-r+posticum_antis>=0:
AntisRow(n,a,xDim)
        else:
               ProstyleRow(n,a,b)

AntisRow(n,a,xDim) -->
        case  a == 1 && scope.sx>=n*2 && antisColumns_front == true
               ||a == 2 && scope.sx>=n*2 && antisColumns_back == true:
               s(xDim,'1,'1)
               center(x)
               col.AntisColonnade(n,centerOpening_front,a)
        else: NIL
```

```
ProstyleRow(n,a,b) -->
	case !noPeristyle && !Tshape:
		t(0,0,column_diameter/2+col.columnOffset)
		s(scope.sx+n*2-column_diameter,'1,'1)
		center(x)
		PorchColonnade(n,a)
	else:
		col.FirstLastFlushFront(n,centerOpening_front)



Pilasters(n) -->
	case pilasters == true:
		s(scope.sx+n*2-column_diameter,'1,'1)
		t(0,0,column_diameter/2)
		center(x)
		split(x){n:    NIL
			|~n: col.ColumnTile(split.index,column_diameter,split.total)
			|~1: NIL
			|n:    col.ColumnTile(split.index,column_diameter,split.total)
			|.01:  NIL}

	else: NIL

PorchColonnade(n,a) -->
	case a == 1 && pronaos_prostyle_gap == 0:
		col.FrontColonnade(n,centerOpening_front)

	case a == 2 && posticum_prostyle_gap == 0:
		col.FrontColonnade(n,centerOpening_front)

	case a == 1 && pronaos_prostyle_gap > 0:
		split(x){n:    NIL
			|{~n:  col.ColumnTile(split.index,column_diameter,split.total)}*
			|~n*centerOpening_front*pronaos_prostyle_gap*2: NIL
			|{~n:  col.ColumnTile(split.index,column_diameter,split.total)}*
			|.01:  NIL}

	case a == 2 && posticum_prostyle_gap > 0:
		split(x){n:    NIL
			|{~n:  col.ColumnTile(split.index,column_diameter,split.total)}*
			|~n*centerOpening_front*posticum_prostyle_gap*2: NIL
			|{~n:  col.ColumnTile(split.index,column_diameter,split.total)}*
			|.01:  NIL}


	else: NIL

///////ENTABLATURE, PEDIMENT & ROOF


Entablature(n) -->
```

```
        col.Entablature(n,column_diameter)


EntablatureFront(n) -->
        col.EntablatureFront(n,centerOpening_front,column_diameter)

Roof(n,zDim) -->

        case tuscan  && generate_roof == true && !Tshape:
                s('roofW,'1,'1)
                center(x)
                Roof1(n)

        case !tuscan &&  generate_roof == true && !Tshape:
                s(scope.sx+architraveD,'1,scope.sz+architraveD)
                center(xz)
                Roof1(n)

        case !tuscan &&  generate_roof == true && Tshape:
                s(scope.sx+architraveD,'1,scope.sz+architraveD)
                center(xz)
                Roof1(n)

        case tuscan &&  generate_roof == true && Tshape:
                s(scope.sx+architraveD,'1,((pronaos_prostyle+pronaos_antis)*n)+architraveD+((zDim-
(pronaos_prostyle+pronaos_antis)*n))/2)
                t(0,0,((zDim-(pronaos_prostyle+pronaos_antis)*n)/2)-architraveD/2)
                center(x)
                Roof1(n)

        else: NIL

TRoof(n) -->
        roofGable( Troof_angle ,0,0, true,1)
        comp(f)         {top: rf.Roof(roofBrickW,roofBrickH)
                               |vertical: tex.Block("wall",tile)
                               |bottom: Overhang}

        comp(e){ ridge: rf.Ridge(0,roofBrickW,roofBrickH)| hip:
rf.Ridge(roofBrickW*0.4,roofBrickW,roofBrickH) | valley:
rf.Ridge(roofBrickW*0.4,roofBrickW,roofBrickH) }

Roof1(n) -->

case Tshape:
        roofGable( roof_angle ,0,-col.geisonProjection-col.triglyphW/2, true,0)
        comp(f)         {top: rf.TempleRoof(n,roofBrickW,roofBrickH)
                               |front: rf.Pediment(n)
                               |bottom: Overhang}
        comp(e){ ridge: rf.Ridge(0,roofBrickW,roofBrickH)rf.TopAcroteria(scope.sx) Beam| hip:
rf.Ridge(roofBrickW*0.4,roofBrickW,roofBrickH) | valley:
rf.Ridge(roofBrickW*0.4,roofBrickW,roofBrickH) |eave: rf.SideAcroteria(scope.sx)}
```

```
else:
        roofGable( roof_angle ,0,-col.geisonProjection-col.triglyphW/2, true,0)
        comp(f)        {top: rf.TempleRoof(n,roofBrickW,roofBrickH)
                             |vertical: rf.Pediment(n)
                             |bottom: Overhang}
        comp(e){ ridge: rf.Ridge(0,roofBrickW,roofBrickH)rf.TopAcroteria(scope.sx) Beam| hip:
rf.Ridge(roofBrickW*0.4,roofBrickW,roofBrickH) | valley:
rf.Ridge(roofBrickW*0.4,roofBrickW,roofBrickH)|eave: rf.SideAcroteria(scope.sx) }

Beam -->
        case tuscan: rf.Beam
        else: NIL

Overhang -->
        case tuscan && generate_roof == true:
               split(y){rf.pedimentWidth-col.geisonProjection: NIL
                             |~1:    s('1,'1.02,.1)
                                          center(y)
                                          tex.Block("wall",tile)
                             |rf.pedimentWidth-col.geisonProjection: NIL}

        else: NIL

///////CEILING

Ceiling -->
        case tuscan && generate_roof ==true:  t(0,0,-entablatureH)tus.Ceiling.
        case doric && generate_roof ==true:          t(0,0,-entablatureH) tex.Block("wall", tile)
        case ionic && generate_roof ==true:   t(0,0,-entablatureH) tex.Block("wall", tile)
reverseNormals tex.Block("wall", tile)
        case corinthian && generate_roof ==true:t(0,0,-entablatureH) tex.Block("wall", tile)
reverseNormals tex.Block("wall", tile)
        else: NIL

InnerCeiling -->
        case generate_roof == true:
        tex.Block("wall",tile)
        else: NIL
```

Porticus

```
import col : "Colonnade.cga"
import dom : "Domus.cga" (houseType = "ROMAN SHOPS", baseHeight = 0, redWall_on = "false")

import rf:   "Roof.cga"
import tex : "TexturesAssets.cga"

#-------General

@Group("General",2)@Range("TUSCAN","DORIC","IONIC","CORINTHIAN")
attr order_                                     = "DORIC"
       const tuscan                             = order_ == "TUSCAN"
       const doric                                     = order_ == "DORIC"
       const ionic                                     = order_ == "IONIC"
       const corinthian                         = order_ == "CORINTHIAN"

@Group("General",2)@Range("STOA","PORTICUS","STOA+SHOPS","PORTICUS+SHOPS","SHOPS","PALAESTRA","PR
OPYLON","COLONNADE STREET", "COLONNADE STREET + SHOPS")
attr building_type                          = "PORTICUS"
       const shops                                     = building_type == "SHOPS" || building_type
== "PORTICUS+SHOPS" || building_type == "STOA+SHOPS" || building_type == "COLONNADE STREET +
SHOPS"
       const propylon                       = building_type == "PROPYLON"

@Group("General",2)@Range("COLONNADE","ARCADE")
attr colonnadeType                          =       "COLONNADE"

       const columns                                   = colonnadeType == "COLONNADE"
       const arches                                    = colonnadeType == "ARCADE"

@Group("General",2)@Range("LINEAR", "OFFSET", "SEGMENTED")
attr shapeType                                  =       "SEGMENTED"
       const linear                                    = shapeType == "LINEAR"
       const offset                                    = shapeType == "OFFSET"
       const single                                    = shapeType == "SEGMENTED"

@Group("General",2)
attr stoaDepth                                  = 10

@Group("General","end type",2)@Range("wall","colonnade","open")
attr left                                       = "open"

@Group("General","end type",2)@Range("wall","colonnade","open")
attr right                                      = "open"

@Group("General","back type",2)@Range("wall","colonnade","open")
attr back                                       = "open"

const stoaHeight                        =       columnHeight+entablatureH
const height                            =       baseHeight+columnHeight
```

```
#-------Base
@Group("Base",3)@Range("continuous steps", "spaced steps", "none")
attr stepType                          = "continuous steps"
       const continuous                = stepType == "continuous steps"
       const spaced                    = stepType == "spaced steps"
       const noStep                    = stepType == "none"

@Group("Base",3)
attr steps                                    = 5

@Group("Base",3)
attr extra_steps_down         = 0

@Group("Base",3)@Range("true","false")
attr backSteps                         = "false"

@Group("Base",3)@Range("true","false")
attr rightSteps                               = "false"

@Group("Base",3)@Range("true","false")
attr leftSteps                         = "false"

@Group("Base",3)
attr step_depth                        = .35

@Group("Base",3)
attr step_height                       = .2

@Group("Base",3)
attr ground_height                     = .5

@Group("Base",3)
attr extra_height                      = 0

@Group("Base",3)
attr stairW                                   = case columns: columnSpacing-column_diameter-
col.columnOffset*2
                                                        else: archWidth

const baseHeight                       = steps*step_height+extra_steps_down*step_height

#------Colonnade

@Group("Colonnade",4)
attr column_diameter                   =      case arches: archColumnDiameter else: 0.6

@Group("Colonnade",4)
attr columnSpacing                            =      case tuscan && columns:
       column_diameter*4
                                                                    case doric && columns:
       column_diameter*2.5
                                                                    case ionic && columns:
       column_diameter*5.5
```

```
                                                        case corinthian && columns:
column_diameter*9
                                                        case tuscan && arches:
        archColumnDiameter*4
                                                        case doric && arches:
        archColumnDiameter*2.5
                                                        case ionic && arches:
        archColumnDiameter*5.5
                                                        else:
                archColumnDiameter*9
@Group("Colonnade",4)
attr plinthH                            =       case arches: archWidth/2 else: 0

@Group("Colonnade","Propylon",4)
attr centerOpening_propylon             =       1 #factor of columnSpacing

@Group("Colonnade","Propylon",4)@Range("TETRASTYLE","HEXASTYLE")
attr propylon_style                     =       "TETRASTYLE"
        const tetrastyle                        =       propylon_style == "TETRASTYLE"
        const hexastyle                                 =       propylon_style == "HEXASTYLE"

const columnHeight                      =       case tuscan && columns:
column_diameter*7+plinthH
                                                        case doric && columns:
        column_diameter*5.5+plinthH
                                                        case ionic && columns:
        column_diameter*9+plinthH
                                                        case corinthian && columns:
column_diameter*10+plinthH
                                                        else: archH
const entablatureH                      =       case columns:
column_diameter*col.entablatureH
                                                        else:
archColumnDiameter*col.entablatureH

#-------Arcade

@Group("Arcade",5) @Range("Front Side", "Both Sides", "Off")
attr showOrders                                 = "Front Side"
        const singleSide                        = showOrders == "Front Side"
        const bothSides                                 = showOrders == "Both Sides"
        const ordersOff                                 = showOrders == "Off"
@Group("Arcade",5)
attr archWidth                          =       3

const archColumnDiameter                =       archWidth/10
const archH                                     =       (1.5*archWidth)+plinthH-archWidth/2
const orderW                            =       col.orderW

#-------Segmented Shape Type

@Group("Segment Type",6)@Range("start","middle","last","standalone")
attr segment                            = "standalone"
        const start                             = segment == "start"
        const middle                    = segment == "middle"
```

```
        const last                                      = segment == "last"
        const standalone                        = segment == "standalone"

#-------Linear Shape Type

@Group("Linear Type",7)
attr sides                              =  4


#-------Shops

@Group("Shops",8)@Range("true","false")
attr noPorticus                                 = "false"

@Group("Shops",8)
attr shopDepth                          = case !shops: 0 else: 4

#--------Walls

const wallThickness             = 1

#--------Roof

@Group("Roof",9)@Range("shed","gable","flat")
attr roofType                           = "gable"

@Group("Roof",9)
attr roof_angle                         = case roofType == "gable": 13.8 else: 10

const roofBrickW                        = columnSpacing *0.25
const roofBrickH                        = columnSpacing *0.35
const railingH                          = 1

#-------Texture Size

const tile                                      = columnHeight
const streetTexTile             = 5

//////////
//////START

Shops -->
case noPorticus == "true" && !noModel:
        t(0,elevation,0)
        extrude(height)
        split(y){baseHeight:color(base)t(0,-extra_steps_down*step_height,0)
                                        comp(f){top:color(floor)tex.Block("floor",.5)
                                                                |front: Steps
                                                                |left:Wall
                                                                |right:Wall
                                                                |back: Wall}
                        |~1:color(wall)comp(f){front:dom.LowerFacade(rint(rand(-.5,1.5)),"front")
                                                                |left:Wall
```

```
                                                        |right:Wall
                                                        |back:Wall
                                                        |top:Roof}
                    }
case noPorticus == "false" && !noModel:      Lot
else: NIL

Lot -->
//t(0,elevation,0)
//Lot.

///*
case offset && !noModel:
       t(0,elevation,0)
       innerRect
       [offset(-stoaDepth)t(0,ground_height,0) OffsetStoa]
       extrude(ground_height)color (ground) tex.Block("ground",5)
case single && !noModel:
       alignScopeToAxes(y)
       SegmentStoa(scope.sy)
case linear && !noModel:
       t(0,elevation,0)
       LinearStoa
else: NIL
//*/
//////////////////
///////OFFSET STOA

OffsetStoa  --> comp(f) { inside: NIL | border: OffsetStoa2 }

OffsetStoa2 -->
color(base)
extrude(baseHeight)
t(0,-extra_steps_down*step_height,0)
comp(f) {4: Steps
              |1: extrude(stoaHeight)
                     comp(f) {1: color (roof)Roof
                                   |4: split(y){col.columnHeight: OffsetColonnade
                                                        |~1:Entablature}
                                   |2: color(wall) OffsetWall
                                   |0: [t(0,0,-columnHeight)offset(-
column_diameter,inside)color(roof) tex.Block("roof",4,24)] reverseNormals color(floor)
tex.Block("floor",.5) }
              |2: Wall}

OffsetColonnade -->
case columns: color(column)col.NoFirstLastOnCorner(columnSpacing)
else: color(wall)col.OffsetArcade

OffsetWall -->
case shops: ShopWalls
else: Wall
```

```
//////////////////
///////LINEAR STOA

# n = comp.index
# y = comp.total
# sides = number of walls +1
# n == 0 floor
# n == 1 roof
# n == 2 last end colonnade (sides = number of exterior faces +1)
# n < sides = walls
# n == sides = start end colonnade
# n == sides+1 = first column side
# n == y-1            = last side

LinearStoa -->
color(base)
extrude(baseHeight)
t(0,-extra_steps_down*step_height,0)
comp(f){all: LinearBaseSide(comp.index, comp.total)}

LinearBaseSide(n,y) -->
case n == 1:                          extrude(stoaHeight) comp(f) {all:
color(wall)LinearSide(comp.index, comp.total)}
case n <= sides && n>1:       color(base) tex.Block("wall",tile)
case n > sides && n<y-1 :     color(base) Steps
case n == y-1:                        StepsEnd
else:                                       NIL

LinearSide(n,y) -->
case n == 0: reverseNormals color(floor) tex.Block("floor",.5) #Floor
                      reverseNormals t(0,0,-columnHeight)offset(-
column_diameter,inside)color(roof) tex.Block("roof",4,24) #Ceiling
               //     reverseNormals t(0,0,-entablatureH)color(floor) tex.Block("floor",.5)
#Flat Roof
case n == 1: Roof #Roof
case !shops && n>2 && n < sides: Walls #Back Walls
case shops  && n>2 && n < sides: split(y){~1:ShopWalls|entablatureH: Entablature}#Back Walls -
Shops
case n == sides && left == "colonnade" && columns:  split(y){~1: s(scope.sx-
shopDepth,'1,'1)t(shopDepth,0,0)col.FirstLastFlush(columnSpacing)|entablatureH: Entablature}
#start end colonnade (left)
case n == sides && left == "colonnade" && arches:  split(y){~1: s(scope.sx-
shopDepth,'1,'1)t(shopDepth,0,0)col.LinearArcade|entablatureH: Entablature} #start end arcade
(left)
case n == sides && left == "open": split(y){~1: NIL|entablatureH: Entablature} #start end
open(left)
case n == sides && left == "wall": split(y){~1: s(scope.sx-shopDepth,'1,'1)t(shopDepth,0,0)
Wall|entablatureH: Entablature} #start end wall (left)
case n == sides+1 && n < y-1 && columns && left != "colonnade": split(y){~1: color(column)
col.FirstFlushLastOnBreak(columnSpacing)|entablatureH: Entablature} #ColonnadeMiddle (first inner
side)
case n == sides+1 && n < y-1 && columns && left == "colonnade": split(y){~1: color(column)
col.NoFirstLastOnBreak(columnSpacing)|entablatureH: Entablature} #ColonnadeMiddle (first inner
side)
case n > sides+1 && n < y-1 && columns: split(y){~1: color(column)
col.NoFirstLastOnBreak(columnSpacing)|entablatureH: Entablature} #ColonnadeMiddle
```

```
case n == y-1 && columns && right == "colonnade": split(y){~1: color(column)
col.NoFirstNoLast(columnSpacing)|entablatureH: Entablature} #ColonnadeMiddle (last inner side)
case n == y-1 && columns && right != "colonnade": split(y){~1: color(column)
col.NoFirstLastFlush(columnSpacing)|entablatureH: Entablature} #ColonnadeMiddle (last inner side)
case n > sides && n <= y-1 && arches: split(y){~1: col.LinearArcade|entablatureH: Entablature}
#ArcadeMiddle
case n == 2 && right == "colonnade" && columns: split(y){~1: s(scope.sx-
shopDepth,'1,'1)col.FirstLastFlush(columnSpacing)|entablatureH: Entablature} #last end colonnade
(right)(columns)
case n == 2 && right == "colonnade" && arches: split(y){~1: s(scope.sx-
shopDepth,'1,'1)col.LinearArcadeEnd|entablatureH: Entablature} #last end arcade (right)(columns)
case n == 2 && right == "open": split(y){~1: NIL|entablatureH: Entablature} #last end open
(right)
case n == 2 && right == "wall": split(y){~1: s(scope.sx-shopDepth,'1,'1) Wall|entablatureH:
Entablature} #last end wall (right)
else: NIL


//////////////////
//////SINGLE SEGMENT

SegmentStoa(yDim) -->
case yDim > 0:
       extrude(world.y,50)
       split(y){yDim+extra_height:SegmentBase(yDim)|~1: NIL}
else:t(0,elevation,0)
       extrude(stoaHeight+baseHeight)
       t(0,-extra_steps_down*step_height,0)
       color(base)
       split(y){baseHeight: comp(f){front:StepsType(scope.sy/step_height)
                                                              |back:
BackStepsType(scope.sy/step_height)
                                                              |left:
LeftStepsType(scope.sy/step_height)

       |right:RightStepsType(scope.sy/step_height) }
                     |~1: SegmentSides}

SegmentBase(yDim) -->
comp(f){top:extrude(stoaHeight) SegmentSides
              |front: color(base)StepsType(scope.sy/step_height)| side:color(base) Wall}


SegmentSides -->
color(wall)
split(y){columnHeight:split(z){shopDepth:color(wall)comp(f){front:s('1,scope.sy+entablatureH-
.1,'1)dom.LowerFacade(rint(rand(-.5,1.5)),"front")

              |back:Wall

              |left: Wall

              |right:Wall

              |bottom: reverseNormals color(floor) tex.WholeBlock("floor",.5)}
                                                              |~1:comp(f)    {front: FrontSide
```

```
                                                                                |left:
        LeftSide
                                                                                |right:
RightSide
                                                                                |back:
        BackSide

        |bottom:reverseNormals color(floor) tex.WholeBlock("floor",.5)}
                                                                    }
                |~1:  comp(f){front:FrontEntablature(scope.sx)
                                |left: Entablature
                                |right: Entablature
                                |back:FrontEntablature(scope.sx)
                                |top:  Ceiling
                                Roof}
                }



FrontSide -->
        case start && left == "wall" && columns && !propylon:
col.NoFirstLastOnBreak(columnSpacing)
        case start && left != "wall" && columns && !propylon:
col.FirstFlushLastOnBreak(columnSpacing)
        case last && right != "wall" && columns && !propylon: col.NoFirstLastFlush(columnSpacing)
        case last && right == "wall" && columns && !propylon: col.NoFirstNoLast(columnSpacing)
        case standalone && right != "wall"    && left != "wall" && columns && !propylon:
        color(column) col.FirstLastFlush(columnSpacing)
        case standalone && right == "wall"    && left != "wall" && columns && !propylon:
        color(column) col.FirstFlushNoLast(columnSpacing)
        case standalone && right != "wall"    && left == "wall" && columns && !propylon:
        color(column) col.NoFirstLastFlush(columnSpacing)
        case standalone && right == "wall"    && left == "wall" && columns && !propylon:
        color(column) col.NoFirstNoLast(columnSpacing)
        case middle  && columns && !propylon:        color(column)
col.NoFirstLastOnBreak(columnSpacing)
        case arches && start : s(scope.sx-col.archColumnBaseW/2,'1,'1)
t(col.archColumnBaseW/2,0,0)col.ArcadeStart
        case arches && standalone: s(scope.sx-col.archColumnBaseW,'1,'1) center(x)
col.ArcadeStart
        case arches && !start || arches && !standalone:s(scope.sx-col.archColumnBaseW/2,'1,'1)
col.Arcade
        case propylon && columns: PropylonColonnade(scope.sx)
        else: NIL



LeftSide -->
        case start && left == "wall"||
                standalone && left == "wall": extrude(-wallThickness)Wall
        case start && left == "colonnade" && back != "wall"  && columns ||
                standalone && left == "colonnade" && back != "wall"  && columns:
col.FirstFlushNoLast(columnSpacing)
        case start && left == "colonnade" && back == "wall"  && columns||
                standalone && left == "colonnade" && back == "wall"  && columns:
col.NoFirstNoLast(columnSpacing)
        case arches && left == "colonnade" && back == "colonnade" :  s(scope.sx-
col.archColumnBaseW,'1,'1) center(x) col.ArcadeStart
```

```
        case arches && left == "colonnade" && back == "wall" :  s(scope.sx-wallThickness,'1,'1)
t(wallThickness-col.archColumnBaseW/2,0,0)col.ArcadeStart
        case arches && left == "colonnade" && back == "open" :  s(scope.sx-
col.archColumnBaseW/2,'1,'1) col.ArcadeStart
        else: NIL

RightSide -->
        case last && right == "wall"||
                standalone && right == "wall": extrude(-wallThickness)Wall
        case last && right == "colonnade" && back != "wall" && columns ||
                standalone && right == "colonnade" && back != "wall"  && columns:
col.NoFirstLastFlush(columnSpacing)
        case last && right == "colonnade" && back == "wall"  && columns||
                standalone && right == "colonnade" && back == "wall" && columns :
col.NoFirstNoLast(columnSpacing)
        case arches && right == "colonnade"  && back == "colonnade":  s(scope.sx-
col.archColumnBaseW,'1,'1) center(x)col.ArcadeStart
        case arches && right == "colonnade"  && back == "wall":  s(scope.sx-
wallThickness,'1,'1)t(col.archColumnBaseW/2,0,0)col.ArcadeStart
        case arches && right == "colonnade"  && back == "open":  s(scope.sx-
col.archColumnBaseW/2,'1,'1)t(col.archColumnBaseW/2,0,0)col.ArcadeStart
        else: NIL

BackSide -->
        case back == "colonnade" && start && left != "open" && columns && !propylon:
col.FirstOnBreakNoLast(columnSpacing)
        case back == "colonnade" && start && left == "open" && columns && !propylon:
col.FirstOnBreakLastFlush(columnSpacing)
        case back == "colonnade" && last && right != "open" && columns && !propylon:
col.NoFirstNoLast(columnSpacing)
        case back == "colonnade" && last && right == "open" && columns && !propylon:
col.FirstFlushNoLast(columnSpacing)
        case back == "colonnade" && standalone && right != "open"   && left != "open" && columns
&& !propylon:  color(column) col.NoFirstNoLast(columnSpacing)
        case back == "colonnade" && standalone && right == "open"   && left != "open" && columns
&& !propylon:  color(column) col.FirstFlushNoLast(columnSpacing)
        case back == "colonnade" && standalone && right != "open"   && left == "open" && columns
&& !propylon:  color(column) col.NoFirstLastFlush(columnSpacing)
        case back == "colonnade" && standalone && right == "open"   && left == "open" && columns
&& !propylon:  color(column) col.FirstLastFlush(columnSpacing)
        case back == "colonnade" && middle && columns && !propylon :       color(column)
col.FirstOnBreakNoLast(columnSpacing)
        case back == "wall" : extrude(-wallThickness)Wall
        case back == "colonnade" && arches && !last  && !standalone :  s(scope.sx-
col.archColumnBaseW/2,'1,'1) col.Arcade
        case back == "colonnade" && arches && last || back == "colonnade" && arches &&
standalone: s(scope.sx-col.archColumnBaseW/2,'1,'1) t(col.archColumnBaseW/2,0,0) col.ArcadeStart
        case back == "colonnade" && propylon && columns: PropylonColonnade(scope.sx)
        else: NIL

PropylonColonnade(X) -->
case tetrastyle:
        GetSpacing((X-((X/3)*centerOpening_propylon))/2)
case hexastyle:
        GetSpacing((X-((X/5)*centerOpening_propylon))/4)
else: GetSpacing(columnSpacing)
```

```
GetSpacing(n) -->
col.FirstLastFlushFront(n,centerOpening_propylon)

////////////
///////STEPS

BackStepsType(n) -->
case backSteps == "true":StepsType(n)
else: Wall

LeftStepsType(n) -->
case leftSteps == "true":StepsType(n)
else: Wall

RightStepsType(n) -->
case rightSteps == "true":StepsType(n)
else: Wall

StepsType(n) -->
case continuous: Steps(n)
case spaced && columns: Wall split(x){ column_diameter+col.columnOffset: NIL|stairW:Steps(n)|~1:
NIL}
case spaced && arches && start || spaced && arches && standalone: Wall split(x){ orderW:
NIL|stairW:Steps(n)|~1: NIL}
case spaced && arches && !start && !standalone : Wall split(x){ stairW:Steps(n)|~1: NIL}
else: Wall

Steps -->
split(y){step_height :  extrude((split.total - split.index) * step_depth) s(scope.sx+(split.total
- split.index) * step_depth,'1,'1) tex.Block("wall", tile)}*

StepsEnd -->
setPivot(xyz,2)
split(y){~step_height :  extrude((split.total - split.index) * step_depth) tex.Block("wall",
tile)}*

Steps(n) -->
alignScopeToAxes()
split(y)       {step_height: Step(n)
               | ~1: Steps(n-1)}

Step(idx) -->
extrude(idx*step_depth)tex.Block("street", streetTexTile)


////////////
///////WALLS

Walls -->
color(wall)
split(y){~1: s('1,'1, wallThickness)
                     t(0,0,-wallThickness)
                     set(trim.vertical, false)
                     i("builtin:cube")
```

```
                    tex.Block("wall", tile)
              |entablatureH:Entablature}

ShopWalls -->
color(wall)
s('1,'1,shopDepth)
t(0,0,-shopDepth)
set(trim.vertical,false)
i("builtin:cube")
comp(f){back:s('1,scope.sy+entablatureH,'1)dom.LowerFacade(rint(rand(-.5,1.5)),"front") |front:
Wall | left: Wall| right:  Wall}


Wall --> tex.Block("wall",tile) reverseNormals tex.Block("wall",tile)

//////////////////
///////ENTABLATURE

Entablature -->
case columns: col.Entablature(columnSpacing,column_diameter)
else: col.Entablature(columnSpacing,archColumnDiameter)

FrontEntablature(X) -->
case propylon && columns && tetrastyle:
       col.EntablatureFront(((X-
((X/3)*centerOpening_propylon))/2),centerOpening_propylon,column_diameter)
case propylon && columns && hexastyle:
       col.EntablatureFront(((X-
((X/5)*centerOpening_propylon))/4),centerOpening_propylon,column_diameter)
case !propylon && columns: col.Entablature(columnSpacing,column_diameter)
else:
       col.Entablature(columnSpacing,archColumnDiameter)
////////////
////////ROOF

Ceiling -->
case roofType == "flat":
       color(floor)tex.Block("floor",.5)#Flat Roof
       t(0,0,-column_diameter*col.friezeH) reverseNormals offset(-
column_diameter/2,inside)color(roof) tex.Block("roof",4,24)
else:
       t(0,0,-column_diameter*col.friezeH) reverseNormals offset(-
column_diameter/2,inside)color(roof) tex.Block("roof",4,24)

Roof -->
case tuscan && propylon:
              s('1.1,'1,'1)
              center(x)
              Roof1
else: Roof1

Roof1 -->
       case roofType == "gable" && !propylon:
              roofGable(roof_angle)
```

```
                comp(f){ top : rf.Roof(roofBrickW,roofBrickH)|vertical: tex.Block("wall", tile)}
                comp(e){ ridge: rf.Ridge(0,roofBrickW,roofBrickH) | hip:
rf.Ridge(roofBrickW*0.4,roofBrickW,roofBrickH) | valley:
rf.Ridge(roofBrickW*0.4,roofBrickW,roofBrickH)}
        case roofType == "gable" && propylon:
                roofGable( roof_angle ,0,-col.geisonProjection-col.triglyphW/2, true,0)
                comp(f){ top : rf.TempleRoof(columnSpacing,roofBrickW,roofBrickH)|vertical:
rf.Pediment(columnSpacing)|bottom: Overhang}
                comp(e){ ridge: rf.Ridge(0,roofBrickW,roofBrickH) | hip:
rf.Ridge(roofBrickW*0.4,roofBrickW,roofBrickH) | valley:
rf.Ridge(roofBrickW*0.4,roofBrickW,roofBrickH)}
        case roofType == "flat":extrude(railingH) comp(f){side:s('1,'1,wallThickness)t(0,0,-
wallThickness)i("builtin:cube")tex.Block("wall",tile)}
        else:
                roofShed(roof_angle)
                comp(f){ top : rf.Roof(roofBrickW,roofBrickH)|vertical: tex.Block("wall", tile)}
                comp(e){ ridge: rf.Ridge(0,roofBrickW,roofBrickH) | hip:
rf.Ridge(roofBrickW*0.4,roofBrickW,roofBrickH) | valley:
rf.Ridge(roofBrickW*0.4,roofBrickW,roofBrickH)}

Overhang -->
        case tuscan:
                split(y){rf.pedimentWidth-col.geisonProjection: NIL
                                |~1:    s('1,'1.02,.1)
                                                center(y)
                                                tex.Block("wall",tile)
                                |rf.pedimentWidth-col.geisonProjection: NIL}

        else: NIL
```

Theater/Stadium

```
import tex : "TexturesAssets.cga"
import col : "Colonnade.cga"
import rf: "Roof.cga"

#--------General

@Group ("General",2)@Range("THEATER","STADIUM")
attr building_type                                  = "STADIUM"
const stadium                           = building_type == "STADIUM"
const theater                           = building_type == "THEATER"

#--------Stadium

@Group ("Stadium",3)@Range("Arcaded","Flat","Colonnaded")
attr sphendoneType                      = "Flat"
@Group ("Stadium",3)
attr sphendoneWidth                     = 1
@Group ("Stadium",3)
attr kerkisWidth                        = 15

const sphendoneH                        = 1.75
const sphendoneH_D1                     = .55
const sphendone2H                       = 4.66
const sphendone2D                       = 1

#--------Theater
@Group ("Theater",4)@Range("5","7")
attr kerkides                           = "5"
const five_kerkides                     = kerkides == "5"
const seven_kerkides          = kerkides == "7"

#--------Rows
@Group ("Rows",5)
attr orchestraPodiumH         = seatHeight*3
@Group ("Rows",5)
attr diazomaWidth                       = seatDepth*2.5
@Group ("Rows",5)
attr tier1_rows                                 = 14
@Group ("Rows",5)
attr tier2_rows                                 = 8
@Group ("Rows",5)
attr seatDepth                          = .75
@Group ("Rows",5)
attr seatHeight                                 = .4
@Group ("Rows",5)
attr stairWidth                                 = .7
@Group ("Rows",5)
attr stepDepth                          = seatDepth/2

const totalSteps                        = tier1_rows+tier2_rows
```

```
const totalH                          = d1H+d2H
const totalD                          = (totalSteps*seatDepth)+(diazomaWidth*2.5)
const d1D                                     = tier1_rows*seatDepth+diazomaWidth
const d1H                                     = tier1_rows*seatHeight+orchestraPodiumH
const d2H                                     = tier2_rows*seatHeight
const d2D                                     = tier2_rows*seatDepth+diazomaWidth

#-------Colonnade
@Group ("Colonnade",5)
attr column_diameter          = 1

#--------Walls
@Group ("Walls",6)
attr angle                                    = 30
@Group ("Walls",6)
attr angle2                                   = 26.5

const wallThickness                   = .6
const tile                                    = 5

#---------Roof
const roofBrickW = col.columnSpacing *0.25
const roofBrickH = col.columnSpacing *0.35

###########__START__#############

#Lot should outline only the orchestra (the ground in the middle, not including the seats)

#Lot should have an odd number of faces around the curved side

#Numbering of faces starts counterclockwise (on the right) with the curved edge, at 0.



Lot -->
case !noModel:
       t(0,elevation,0)
       //Lot.
       ///*
       extrude(seatHeight*(tier1_rows+tier2_rows)+orchestraPodiumH+sphendoneH+sphendone2H)
       comp(f) { bottom:reverseNormals color(ground)tex.Block("ground",5) | front: NIL | top:
NIL |all = AllSides}
       //*/
else: NIL

AllSides -->
       comp(f) {all: Sides(comp.index, comp.total) print(comp.total)}


Sides(i, z) -->

## Theater - 5 kerkides
```

```
        case theater && five_kerkides && i == 0:
        EndStairsD2

        case theater && five_kerkides && i < rint(z/5*1):                                   Rows

        case theater && five_kerkides && i == rint(z/5*1):                                  Stairs

        case theater && five_kerkides && i > rint(z/5*1) && i< rint(z/5*2): Rows

        case theater && five_kerkides && i ==  rint(z/5*2):                                 Stairs

        case theater && five_kerkides && i > rint(z/5*2) && i< rint(z/5*3): Rows

        case theater && five_kerkides && i == rint(z/5*3):                                  Stairs

        case theater && five_kerkides && i > rint(z/5*3) && i< rint(z/5*4): Rows

        case theater && five_kerkides && i == rint(z/5*4):                                  Stairs

        case theater && five_kerkides && i == z-1:
        EndStairsLeftD2


## Theater - 7 kerkides
        case theater && seven_kerkides && i == 0 :
        EndStairs

        case theater && seven_kerkides  && i < rint(z/7*1):                                 EndRows

        case theater && seven_kerkides  && i == rint(z/7*1):
        EndStairsUpper

        case theater && seven_kerkides  && i > rint(z/7*1) && i < rint(z/7*2):     Rows

        case theater && seven_kerkides  && i == rint(z/7*2):
        Stairs

        case theater && seven_kerkides  && i > rint(z/7*2) && i < rint(z/7*3):     Rows

        case theater && seven_kerkides  && i == rint(z/7*3):
        Stairs

        case theater && seven_kerkides  && i > rint(z/7*3) && i <(z/7*4):          Rows

        case theater && seven_kerkides  && i == rint(z/7*4):
        Stairs

        case theater && seven_kerkides  && i > rint(z/7*4) && i < rint(z/7*5):     Rows

        case theater && seven_kerkides  && i == rint(z/7*5):
        Stairs

        case theater && seven_kerkides  && i > rint(z/7*5) && i < rint(z/7*6):     Rows
```

```
        case theater && seven_kerkides  && i == rint(z/7*6):
        EndStairsUpperLeft

        case theater && seven_kerkides  && i > rint(z/7*6) && i < z-1:                    EndRows

        case theater && seven_kerkides  && i == z-1:
        EndStairsLeft

## Stadium

        case stadium && i == 0:                                                          StadiumArm

        case stadium && i%4 == 0 && i != z-1:                                StadiumStairs

        case stadium && i%4 != 0 && i%2 == 0:                                UpperStairs

        case stadium && i == z-1:                                                StadiumArmLeft

        else:
        Rows


StadiumArm -->
        split(x){stairWidth: EndStairsD2Straight |~1: AllStraightRows|1: EndStairsD2NoWall}

StadiumArmLeft -->
        split(x){1: EndStairsLeftD2NoWall |~1: AllStraightRows|stairWidth:
EndStairsLeftStraightD2 }

AllStraightRows -->
split(x){kerkisWidth:StraightRows |stairWidth: StraightStairs|~1:split(x){~kerkisWidth:
StraightRows |~stairWidth: StraightStairs}*|kerkisWidth: StraightRows}



###########__ROWS__#############

Rows -->
case sphendoneType == "Flat" || sphendoneType == "Colonnaded":
        s('1,scope.sy+sphendone2H,'1)
        color(wall)
        split(y){orchestraPodiumH: reverseNormals Row(split.index,split.total)|{~seatHeight:
Row(split.index,split.total)}*|sphendoneH: SphendoneD2|sphendone2H: NIL|sphendone2H: NIL}
else:
        s('1,scope.sy+sphendone2H,'1)
        color(wall)
        split(y){orchestraPodiumH: reverseNormals Row(split.index,split.total)|{~seatHeight:
Row(split.index,split.total)}*|sphendoneH: SphendoneD2|sphendone2H:SphendoneD3|sphendone2H:
SphendoneD4}

Row(n,z) -->

        case n == 0:
        s('4,'1,diazomaWidth)
        center(x)
```

```
        t(0,0,seatDepth*n)
        Seat

        case n == tier1_rows-1:
        s('4,'1,diazomaWidth)
        center(x)
        t(0,0,seatDepth*n+seatDepth)
        Seat

        case n>= tier1_rows && n<z-4:
        s('4,'1,seatDepth)
        center(x)
        t(0,0,seatDepth*n+diazomaWidth)
        Seat

        case n == z-4:
        s('5,'1,seatDepth+diazomaWidth)
        center(x)
        t(0,0,seatDepth*n+diazomaWidth)
        Seat

        else:
        s('5,'1,seatDepth)
        center(x)
        t(0,0,seatDepth*n+seatDepth)
        Seat

###########__ROWS_STRAIGHT__#############

StraightRows -->
        color(wall)
        split(y){orchestraPodiumH: reverseNormals
StraightRow(split.index,split.total)|{~seatHeight:
StraightRow(split.index,split.total)}*|sphendoneH: SphendoneD2Straight|sphendone2H: NIL}

StraightRow(n,z) -->

        case n == 0:
        s('1,'1,diazomaWidth)
        center(x)
        t(0,0,seatDepth*n)
        Seat

        case n == tier1_rows-1:
        s('1,'1,diazomaWidth)
        center(x)
        t(0,0,seatDepth*n+seatDepth)
        Seat

        case n>= tier1_rows && n<z-3:
        s('1,'1,seatDepth)
        center(x)
        t(0,0,seatDepth*n+diazomaWidth)
```

```
        Seat

        case n == z-3:
        s('1,'1,seatDepth+diazomaWidth)
        center(x)
        t(0,0,seatDepth*n+diazomaWidth)
        Seat

        else:
        s('1,'1,seatDepth)
        center(x)
        t(0,0,seatDepth*n+seatDepth)
        Seat

###########__END_ROWS__#############

EndRows -->
        color(wall)
        split(y){orchestraPodiumH: reverseNormals EndRow(split.index)|{~seatHeight:
EndRow(split.index)}* |sphendoneH: SphendoneD1|sphendone2H: NIL}


EndRow(n) -->

        case n == 0:
        s('1.2,'1,diazomaWidth)
        center(x)
        t(0,0,seatDepth*n)
        Seat

        case n == tier1_rows-1:
        s('2,'1,diazomaWidth)
        center(x)
        t(0,0,seatDepth*n+seatDepth)
        Seat

        case n>= tier1_rows:
        NIL

        else:
        s('2,'1,seatDepth)
        center(x)
        t(0,0,seatDepth*n+seatDepth)
        Seat

###########__SEAT__#############

Seat -->
        i("builtin:cube")
        comp(f) {top: Block |back: Block |left: Block |right: Block}


SeatEnd -->
```

```
        i("builtin:cube")
        comp(f) {bottom: reverseNormals Block |front:reverseNormals Block |left: reverseNormals
Block |right: reverseNormals Block}


###########__STAIRS__#############

Stairs -->

        color(wall)
        split(y){orchestraPodiumH: reverseNormals Steps(split.index,split.total)|{~seatHeight:
Steps(split.index,split.total)}*|sphendoneH: SphendoneD2 |sphendone2H: NIL}



Steps(n,y) -->
        case n == 0:
        s('2,'1,diazomaWidth)
        center(x)
        t(0,0,seatDepth*n)
        Seat

        case n == tier1_rows-1:
        s('8,'1,diazomaWidth)
        center(x)
        t(0,0,seatDepth*n+seatDepth)
        Seat

        case n>= tier1_rows && n<y-3:
        s('8,'1,seatDepth)
        center(x)
        t(0,0,seatDepth*n+diazomaWidth)
        split(x){~1:s('1,'1, seatDepth)Seat |stairWidth: Step|~1: s('1,'1, seatDepth)Seat}

        case n == y-3:
        s('9,'1,seatDepth+diazomaWidth)
        center(x)
        t(0,0,seatDepth*n+diazomaWidth)
        split(x){~1:s('1,'1, seatDepth+diazomaWidth)Seat
                        |stairWidth: Step
t(0,0,seatDepth)s('1,'1,diazomaWidth)i("builtin:cube")comp(f){top: Block}
                        |~1: s('1,'1, seatDepth+diazomaWidth) Seat}

        else:
        s('8,'1,'1)
        center(x)
        t(0,0,seatDepth*n+seatDepth)
        split(x){~1:s('1,'1, seatDepth)Seat |stairWidth: Step|~1: s('1,'1, seatDepth)Seat}

###########__STADIUM_STAIRS__#############

StadiumStairs -->
case sphendoneType == "Flat" || sphendoneType == "Colonnaded":
        s('1,scope.sy+sphendone2H,'1)
```

```
        color(wall)
        split(y){orchestraPodiumH: reverseNormals
StadiumSteps(split.index,split.total)|{~seatHeight:
StadiumSteps(split.index,split.total)}*|sphendoneH: SphendoneD2|sphendone2H: NIL|sphendone2H: NIL
}


else:
        s('1,scope.sy+sphendone2H,'1)
        color(wall)
        split(y){orchestraPodiumH: reverseNormals
StadiumSteps(split.index,split.total)|{~seatHeight:
StadiumSteps(split.index,split.total)}*|sphendoneH: SphendoneD2|sphendone2H:
SphendoneD3|sphendone2H:SphendoneD4 }


StadiumSteps(n,y) -->
        case n == 0:
        s('2,'1,diazomaWidth)
        center(x)
        t(0,0,seatDepth*n)
        split(x){~1: Seat |stairWidth: split(y){~seatHeight:Step|~seatHeight:
t(0,0,seatDepth)Step}|~1: Seat}

        case n == tier1_rows-1:
        s('8,'1,diazomaWidth)
        center(x)
        t(0,0,seatDepth*n+seatDepth)
        Seat

        case n>= tier1_rows && n<y-4:
        s('8,'1,seatDepth)
        center(x)
        t(0,0,seatDepth*n+diazomaWidth)
        split(x){~1:s('1,'1, seatDepth)Seat |stairWidth: Step|~1: s('1,'1, seatDepth)Seat}

        case n == y-4:
        s('9,'1,seatDepth+diazomaWidth)
        center(x)
        t(0,0,seatDepth*n+diazomaWidth)
        split(x){~1:s('1,'1, seatDepth+diazomaWidth)Seat
                        |stairWidth: Step
t(0,0,seatDepth)s('1,'1,diazomaWidth)i("builtin:cube")comp(f){top: Block}
                        |~1: s('1,'1, seatDepth+diazomaWidth) Seat}

        else:
        s('8,'1,'1)
        center(x)
        t(0,0,seatDepth*n+seatDepth)
        split(x){~1:s('1,'1, seatDepth)Seat |stairWidth: Step|~1: s('1,'1, seatDepth)Seat}


###########__UPPER_STAIRS__#############

UpperStairs -->
case sphendoneType == "Flat" || sphendoneType == "Colonnaded":
        s('1,scope.sy+sphendone2H,'1)
```

```
	color(wall)
	split(y){orchestraPodiumH: reverseNormals
UpperSteps(split.index,split.total)|{~seatHeight:
UpperSteps(split.index,split.total)}*|sphendoneH: SphendoneD2|sphendone2H: NIL |sphendone2H: NIL
}
else:
	s('1,scope.sy+sphendone2H,'1)
	color(wall)
	split(y){orchestraPodiumH: reverseNormals
UpperSteps(split.index,split.total)|{~seatHeight:
UpperSteps(split.index,split.total)}*|sphendoneH: SphendoneD2|sphendone2H:
SphendoneD3|sphendone2H: SphendoneD4 }

UpperSteps(n,y) -->
	case n == 0:
	s('2,'1,diazomaWidth)
	center(x)
	t(0,0,seatDepth*n)
	Seat

	case n == tier1_rows-1:
	s('8,'1,diazomaWidth)
	center(x)
	t(0,0,seatDepth*n+seatDepth)
	Seat

	case n>= tier1_rows && n<y-4:
	s('8,'1,seatDepth)
	center(x)
	t(0,0,seatDepth*n+diazomaWidth)
	split(x){~1:s('1,'1, seatDepth)Seat |stairWidth: Step|~1: s('1,'1, seatDepth)Seat}

	case n == y-4:
	s('9,'1,seatDepth+diazomaWidth)
	center(x)
	t(0,0,seatDepth*n+diazomaWidth)
	split(x){~1:s('1,'1, seatDepth+diazomaWidth)Seat
		|stairWidth: Step
t(0,0,seatDepth)s('1,'1,diazomaWidth)i("builtin:cube")comp(f){top: Block}
		|~1: s('1,'1, seatDepth+diazomaWidth) Seat}

	else:
		s('5,'1,seatDepth)
	center(x)
	t(0,0,seatDepth*n+seatDepth)
	Seat

###########__STAIRS_STRAIGHT__#############

StraightStairs -->

	color(wall)
	split(y){orchestraPodiumH: reverseNormals
StraightSteps(split.index,split.total)|{~seatHeight:
StraightSteps(split.index,split.total)}*|sphendoneH: SphendoneD2Straight|sphendone2H: NIL }
```

```
StraightSteps(n,y) -->

        case n == 0:
        s('1,'1,diazomaWidth)
        center(x)
        t(0,0,seatDepth*n)
        split(y){~seatHeight:Step|~seatHeight: t(0,0,seatDepth)Step}

        case n == tier1_rows-1:
        s('1,'1,diazomaWidth)
        center(x)
        t(0,0,seatDepth*n+seatDepth)
        Seat

        case n>= tier1_rows && n<y-3:
        s('1,'1,seatDepth)
        center(x)
        t(0,0,seatDepth*n+diazomaWidth)
        Step

        case n == y-3:
        s('1,'1,seatDepth+diazomaWidth)
        center(x)
        t(0,0,seatDepth*n+diazomaWidth)
        Step t(0,0,seatDepth)s('1,'1,diazomaWidth)i("builtin:cube")comp(f){top: Block}



        else:
        s('1,'1,'1)
        center(x)
        t(0,0,seatDepth*n+seatDepth)
        Step

###########__END_STAIRS__#############

EndStairs -->
        color(wall)
        split(y){orchestraPodiumH: reverseNormals EndSteps(split.index)|{~seatHeight:
EndSteps(split.index)}*|sphendoneH: SphendoneD1End |sphendone2H: NIL}



EndSteps(n) -->
        case n == 0:
        s('4,'1,tier1_rows*seatDepth)
        t(0,0,seatDepth*n)
        i("builtin:cube")
        comp(f){left: split(x){diazomaWidth-stepDepth: extrude(wallThickness)Block

        |~1:s('1,scope.sy+seatHeight,'1)extrude(wallThickness)
                                                comp(f) {back:
        s(scope.sx+diazomaWidth,'1,'1)t(-diazomaWidth,0,0)AngledWall
```

```
                                                                    |
top:s(scope.sx+diazomaWidth,'1,'1) Block | left:Block
                                                                    | bottom: Block}}
                        |top: split(y) {~1: NIL| diazomaWidth: Block }
                        |back:Block}

        case n == tier1_rows-1:
        s('4,'1,diazomaWidth)
        t(0,0,seatDepth*n+seatDepth)
        Seat

        case n>= tier1_rows:
        NIL

        else:
        s('4,'1,seatDepth)
        t(0,0,seatDepth*n+seatDepth)
        split(x){stairWidth: Step|~1: s('1,'1, seatDepth)Seat}

###########__END_STAIRS_D2_#############

EndStairsD2 -->
        color(wall)
        split(y){orchestraPodiumH: reverseNormals EndStepsD2(split.index,
split.total)|{~seatHeight: EndStepsD2(split.index, split.total)}*|sphendoneH:
EndSphendoneD2|sphendone2H: NIL}



EndStepsD2(n,y) -->
        case n == 0:
        s('4,'1,d1D)
        t(0,0,seatDepth*n)
        i("builtin:cube")
        comp(f){left: split(x){diazomaWidth: extrude(wallThickness)Block
                                                    |~1:s(totalD-
diazomaWidth,scope.sy+seatHeight,'1)extrude(wallThickness)
                                                    comp(f) {back: AngledWall
                                                            | top: Block | left:Block
                                                            | bottom: Block}}
                        |top: split(y) {~1: NIL| diazomaWidth: Block }
                        |back:Block}

        case n == tier1_rows-1:
        s('4,'1,tier2_rows*seatDepth+diazomaWidth)
        t(0,0,seatDepth*n+seatDepth)
        i("builtin:cube")
        comp(f) {top: split(y){~1: NIL
                                                    |diazomaWidth:Block }
                        |back: Block}

        case n>= tier1_rows && n<y-3:
        s('4,'1,seatDepth)
        t(0,0,seatDepth*n+diazomaWidth)
        split(x){
```

```
                        stairWidth:Step
                        |~1: s('1,'1, seatDepth)Seat}

        case n == y-3:
                s('9,'1,seatDepth+diazomaWidth)
                t(0,0,seatDepth*n+diazomaWidth)
                split(x){
                                stairWidth: Step
t(0,0,seatDepth)s('1,'1,diazomaWidth)i("builtin:cube")comp(f){top: Block}
                                |~1: s('1,'1, seatDepth+diazomaWidth)Seat}

        else:
        s('4,'1,seatDepth)
        t(0,0,seatDepth*n+seatDepth)
        split(x){stairWidth: Step|~1: s('1,'1, seatDepth)Seat}

###########__END_STAIRS_D2_NO_WALL#############

EndStairsD2NoWall -->
case sphendoneType == "Flat" || sphendoneType == "Colonnaded":
        s('1,scope.sy+sphendone2H,'1)
        color(wall)
        split(y){orchestraPodiumH: reverseNormals EndStepsD2NoWall(split.index,
split.total)|{~seatHeight: EndStepsD2NoWall(split.index, split.total)}*|sphendoneH:
EndSphendoneD2|sphendone2H: NIL|sphendone2H: NIL}
else:
        s('1,scope.sy+sphendone2H,'1)
        color(wall)
        split(y){orchestraPodiumH: reverseNormals EndStepsD2NoWall(split.index,
split.total)|{~seatHeight: EndStepsD2NoWall(split.index, split.total)}*|sphendoneH:
EndSphendoneD2| sphendone2H: s(kerkisWidth/2,'1,'1) SphendoneD3|sphendone2H: SphendoneD4}

EndStepsD2NoWall(n,y) -->
        case n == 0:
        s('5,'1,d1D)
        t(0,0,seatDepth*n)
        split(y){~seatHeight:Step|~seatHeight: t(0,0,seatDepth)Step}

        case n == tier1_rows-1:
        s('5,'1,tier2_rows*seatDepth+diazomaWidth)
        t(0,0,seatDepth*n+seatDepth)
        i("builtin:cube")
        comp(f) {top: split(y){~1: NIL
                                                        |diazomaWidth:Block }
                        |back: Block}

        case n>= tier1_rows && n<y-4:
        s('5,'1,seatDepth)
        t(0,0,seatDepth*n+diazomaWidth)
        split(x){
                        stairWidth:Step
                        |~1: s('1,'1, seatDepth)Seat}

        case n == y-4:
```

```
                s('9,'1,seatDepth+diazomaWidth)
                t(0,0,seatDepth*n+diazomaWidth)
                split(x){
                                stairWidth: Step
t(0,0,seatDepth)s('1,'1,diazomaWidth)i("builtin:cube")comp(f){top: Block}
                                |~1: s('1,'1, seatDepth+diazomaWidth)Seat}

        else:
        s('5,'1,seatDepth)
        t(0,0,seatDepth*n+seatDepth)
        split(x){stairWidth: Step|~1: s('1,'1, seatDepth)Seat}

###########__END_STAIRS_D2_STRAIGHT#############

EndStairsD2Straight -->
        color(wall)
        split(y){orchestraPodiumH: reverseNormals EndStepsD2Straight(split.index,
split.total)|{~seatHeight: EndStepsD2Straight(split.index, split.total)}*|sphendoneH:
EndSphendoneD2Straight|sphendone2H: NIL}



EndStepsD2Straight(n,y) -->
        case n == 0:
        s('1,'1,d1D)
        t(0,0,seatDepth*n)
                i("builtin:cube")
        comp(f){left: split(x){diazomaWidth-stepDepth: extrude(wallThickness)Block
                                                        |~1:s(totalD-
diazomaWidth+stepDepth,scope.sy+seatHeight,'1)extrude(wallThickness)
                                                        comp(f) {back: AngledWall
                                                                | top: Block | left:Block
                                                                | bottom: Block}}
                        |top: split(y) {~1: NIL| diazomaWidth: Block }
                        |back:Block}



        case n == tier1_rows-1:
        s('1,'1,tier2_rows*seatDepth+diazomaWidth)
        t(0,0,seatDepth*n+seatDepth)
        i("builtin:cube")
        comp(f) {top: split(y){~1: NIL
                                                        |diazomaWidth:Block }
                        |back: Block}

        case n>= tier1_rows && n<y-3:
        s('1,'1,seatDepth)
        t(0,0,seatDepth*n+diazomaWidth)
        split(x){
                        stairWidth:Step
                        |~1: s('1,'1, seatDepth)Seat}

        case n == y-3:
                s('1,'1,seatDepth+diazomaWidth)
```

```
                t(0,0,seatDepth*n+diazomaWidth)
                split(x){
                                stairWidth: Step
t(0,0,seatDepth)s('1,'1,diazomaWidth)i("builtin:cube")comp(f){top: Block}
                                |~1: s('1,'1, seatDepth+diazomaWidth)Seat}

        else:
        s('1,'1,seatDepth)
        t(0,0,seatDepth*n+seatDepth)
        split(x){stairWidth: Step|~1: s('1,'1, seatDepth)Seat}

###########__END_STAIRS_LEFT_D2_#############

EndStairsLeftD2 -->
        color(wall)
        split(y){orchestraPodiumH: reverseNormals
mirrorScope(true,false,false)EndStepsLeftD2(split.index,
split.total)|{~seatHeight:mirrorScope(true,false,false) EndStepsLeftD2(split.index,
split.total)}*|sphendoneH:mirrorScope(true,false,false) EndSphendoneD2Left|sphendone2H: NIL}



EndStepsLeftD2(n,y) -->
        case n == 0:
        s('4,'1,d1D)
        t(0,0,seatDepth*n)
        i("builtin:cube")
        comp(f){right: reverseNormals split(x){diazomaWidth: extrude(wallThickness)Block
                                                        |~1:s(totalD-
diazomaWidth,scope.sy+seatHeight,'1)extrude(wallThickness)
                                                        comp(f) {front: AngledWallLeft
                                                                        | bottom: Block | left:Block
                                                                        | top: Block}}
                        |bottom: reverseNormals split(y) {~1: NIL| diazomaWidth: Block }
                        |front:reverseNormals Block}

        case n == tier1_rows-1:
        s('8,'1,tier2_rows*seatDepth+diazomaWidth)
        t(0,0,seatDepth*n+seatDepth)
        i("builtin:cube")
        comp(f) {bottom: reverseNormals split(y){~1: NIL
                                                        |diazomaWidth:Block }
                        |front: reverseNormals Block}

        case n>= tier1_rows && n<y-3:
        s('8,'1,seatDepth)
        t(0,0,seatDepth*n+diazomaWidth)
        split(x){
                        stairWidth:StepEnd
                        |~1: s('1,'1, seatDepth)SeatEnd}

        case n == y-3:
                s('9,'1,seatDepth+diazomaWidth)
                t(0,0,seatDepth*n+diazomaWidth)
                split(x){
```

```
                                stairWidth: StepEnd
t(0,0,seatDepth)s('1,'1,diazomaWidth)i("builtin:cube")comp(f){bottom: reverseNormals Block}
                                |~1: s('1,'1, seatDepth+diazomaWidth)SeatEnd}

        else:
        s('4,'1,seatDepth)
        t(0,0,seatDepth*n+seatDepth)
        split(x){stairWidth: StepEnd|~1: s('1,'1, seatDepth)SeatEnd}

###########__END_STAIRS_LEFT_D2_NO_WALL#############

EndStairsLeftD2NoWall -->
case sphendoneType == "Flat" || sphendoneType == "Colonnaded":
        s('1,scope.sy+sphendone2H,'1)
        color(wall)
        split(y){orchestraPodiumH: reverseNormals
mirrorScope(true,false,false)EndStepsLeftD2NoWall(split.index,
split.total)|{~seatHeight:mirrorScope(true,false,false) EndStepsLeftD2NoWall(split.index,
split.total)}*|sphendoneH:mirrorScope(true,false,false) EndSphendoneD2Left| sphendone2H: NIL|
sphendone2H: NIL}
else:
        s('1,scope.sy+sphendone2H,'1)
        color(wall)
        split(y){orchestraPodiumH: reverseNormals
mirrorScope(true,false,false)EndStepsLeftD2NoWall(split.index,
split.total)|{~seatHeight:mirrorScope(true,false,false) EndStepsLeftD2NoWall(split.index,
split.total)}*|sphendoneH:mirrorScope(true,false,false) EndSphendoneD2Left| sphendone2H:
s(kerkisWidth/2,'1,'1)t(-scope.sx,0,0)SphendoneD3|sphendone2H:SphendoneD4}

EndStepsLeftD2NoWall(n,y) -->
        case n == 0:
        s('4,'1,d1D)
        t(0,0,seatDepth*n)
        split(y){~seatHeight:StepEnd|~seatHeight: t(0,0,seatDepth)StepEnd}

        case n == tier1_rows-1:
        s('8,'1,tier2_rows*seatDepth+diazomaWidth)
        t(0,0,seatDepth*n+seatDepth)
        i("builtin:cube")
        comp(f) {bottom: reverseNormals split(y){~1: NIL
                                                |diazomaWidth:Block }
                |front: reverseNormals Block}

        case n>= tier1_rows && n<y-4:
        s('8,'1,seatDepth)
        t(0,0,seatDepth*n+diazomaWidth)
        split(x){
                stairWidth:StepEnd
                |~1: s('1,'1, seatDepth)SeatEnd}

        case n == y-4:
                s('9,'1,seatDepth+diazomaWidth)
                t(0,0,seatDepth*n+diazomaWidth)
                split(x){
```

```
                                stairWidth: StepEnd
t(0,0,seatDepth)s('1,'1,diazomaWidth)i("builtin:cube")comp(f){bottom: reverseNormals Block}
                                |~1: s('1,'1, seatDepth+diazomaWidth)SeatEnd}

        else:
        s('4,'1,seatDepth)
        t(0,0,seatDepth*n+seatDepth)
        split(x){stairWidth: StepEnd|~1: s('1,'1, seatDepth)SeatEnd}

###########__END_STAIRS_LEFT_D2_STRAIGHT#############

EndStairsLeftStraightD2 -->
        color(wall)
        split(y){orchestraPodiumH: reverseNormals
mirrorScope(true,false,false)EndStepsLeftStraightD2(split.index,
split.total)|{~seatHeight:mirrorScope(true,false,false) EndStepsLeftStraightD2(split.index,
split.total)}*|sphendoneH:mirrorScope(true,false,false) EndSphendoneLeftD2Straight|sphendone2H:
NIL}



EndStepsLeftStraightD2(n,y) -->
        case n == 0 :
        s('1,'1,d1D)
        t(0,0,seatDepth*n)
        i("builtin:cube")
        comp(f){right: reverseNormals split(x){diazomaWidth-stepDepth:
extrude(wallThickness)Block
                                                |~1:s(totalD-
diazomaWidth+stepDepth,scope.sy+seatHeight,'1)extrude(wallThickness)
                                                comp(f) {front: AngledWallLeft
                                                        | bottom: Block | left:Block
                                                        | top: Block}}
                        |bottom: reverseNormals split(y) {~1: NIL| diazomaWidth: Block }
                        |front:reverseNormals Block}



        case n == tier1_rows-1:
        s('1,'1,tier2_rows*seatDepth+diazomaWidth)
        t(0,0,seatDepth*n+seatDepth)
        i("builtin:cube")
        comp(f) {bottom: reverseNormals split(y){~1: NIL
                                                |diazomaWidth:Block }
                        |front: reverseNormals Block}

        case n>= tier1_rows && n<y-3:
        s('1,'1,seatDepth)
        t(0,0,seatDepth*n+diazomaWidth)
        split(x){
                        stairWidth:StepEnd
                        |~1: s('1,'1, seatDepth)SeatEnd}

        case n == y-3:
                s('1,'1,seatDepth+diazomaWidth)
                t(0,0,seatDepth*n+diazomaWidth)
```

```
                split(x){
                                stairWidth: StepEnd
t(0,0,seatDepth)s('1,'1,diazomaWidth)i("builtin:cube")comp(f){bottom: reverseNormals Block}
                                |~1: s('1,'1, seatDepth+diazomaWidth)SeatEnd}

        else:
        s('1,'1,seatDepth)
        t(0,0,seatDepth*n+seatDepth)
        split(x){stairWidth: StepEnd|~1: s('1,'1, seatDepth)SeatEnd}



###########__END_STAIRS_LEFT__#############

EndStairsLeft -->
        color(wall)
        split(y){orchestraPodiumH: reverseNormals mirrorScope(true,false,false)
EndStepsLeft(split.index)|{~seatHeight:mirrorScope(true,false,false) EndStepsLeft (split.index)}*
|sphendoneH:mirrorScope(true,false,false)  SphendoneD1LeftEnd|sphendone2H: NIL}



EndStepsLeft(n) -->
        case n == 0:
        s('4,'1,tier1_rows*seatDepth)
        t(0,0,seatDepth*n)
        i("builtin:cube")
        comp(f){right: reverseNormals split(x){diazomaWidth-stepDepth:
extrude(wallThickness)Block

        |~1:s('1,scope.sy+seatHeight,'1)extrude(wallThickness)
                                                comp(f) {front:
        s(scope.sx+diazomaWidth,'1,'1) AngledWallLeft| top: s(scope.sx+diazomaWidth,'1,'1)t(-
diazomaWidth,0,0)Block | left:Block | bottom: Block}}
                        |bottom: reverseNormals split(y) {~1: NIL| diazomaWidth: Block }
                        |front:reverseNormals Block}

        case n == tier1_rows-1:
        s('4,'1,diazomaWidth)
        t(0,0,seatDepth*n+seatDepth)
        SeatEnd

        case n>= tier1_rows:
        NIL

        else:
        s('4,'1,seatDepth)
        t(0,0,seatDepth*n+seatDepth)
        split(x){ stairWidth: StepEnd| ~1: s('1,'1, seatDepth)SeatEnd}

###########__END_STAIRS_UPPER__#############

EndStairsUpper -->
        color(wall)
```

```
        split(y){orchestraPodiumH: reverseNormals
EndStepsUpper(split.index,split.total)|{~seatHeight:
EndStepsUpper(split.index,split.total)}*|sphendoneH: EndSphendoneD2|sphendone2H: NIL }

EndStepsUpper(n,y) -->

        case n == 0:
        s('2,'1,diazomaWidth)
        center(x)
        t(0,0,seatDepth*n)
        Seat

        case n == tier1_rows-1:
        s('8,'1,tier2_rows*seatDepth+diazomaWidth)
        center(x)
        t(0,0,seatDepth*n+seatDepth)
        i("builtin:cube")
        comp(f) {top: split(y){~1: split(x){~1: split(x){~1: split(y){~1: NIL|sphendoneWidth:
extrude(sphendoneH_D1)s('1,scope.sy+d1H,'1)t(0,-d1H+seatHeight,0)Block}

                        |wallThickness:  s('1,scope.sy+diazomaWidth*1.5,'1)t(0,-
diazomaWidth*1.5,sphendoneH_D1+seatHeight)UpperAngledWall }
                                                                        |stairWidth: NIL
                                                                        |~1: NIL}
                                                    |diazomaWidth:Block }
                        |back: Block}

        case n>= tier1_rows && n<y-3:
        s('8,'1,seatDepth)
        center(x)
        t(0,0,seatDepth*n+diazomaWidth)
        split(x){~1: NIL
                        |stairWidth:Step
                        |~1: s('1,'1, seatDepth)Seat}

        case n == y-3:
        s('9,'1,seatDepth+diazomaWidth)
        center(x)
        t(0,0,seatDepth*n+diazomaWidth)
        split(x){~1:NIL
                        |stairWidth: Step
t(0,0,seatDepth)s('1,'1,diazomaWidth)i("builtin:cube")comp(f){top: Block}
                        |~1: s('1,'1, seatDepth+diazomaWidth)Seat}

        else:
        s('8,'1,'1)
        center(x)
        t(0,0,seatDepth*n+seatDepth)
        split(x){~1:s('1,'1, seatDepth)Seat |stairWidth: Step|~1: s('1,'1, seatDepth)Seat}

############__END_STAIRS_UPPER_LEFT__##############

EndStairsUpperLeft -->
        color(wall)
```

```
        split(y){orchestraPodiumH: reverseNormals  mirrorScope(true,false,true)
EndStepsUpperLeft(split.index,split.total)|{~seatHeight: mirrorScope(true,false,true)
EndStepsUpperLeft(split.index,split.total)}*|sphendoneH:
mirrorScope(true,false,true)EndSphendoneD2Left |sphendone2H: NIL}



EndStepsUpperLeft(n,y) -->
        case n == 0:
        s('2,'1,diazomaWidth)
        center(x)
        t(0,0,seatDepth*n)
        SeatEnd

        case n == tier1_rows-1:
        s('8,'1,tier2_rows*seatDepth+diazomaWidth)
        t(0,0,seatDepth*n+seatDepth)
        center(x)
        i("builtin:cube")
        comp(f){bottom:reverseNormals split(y){~1: split(x){~1: split(x){~1: split(y){~1:
NIL|sphendoneWidth: extrude(sphendoneH_D1)s('1,scope.sy+d1H,'1)t(0,-d1H+seatHeight,0)Block}

                        |wallThickness:  s('1,scope.sy+diazomaWidth*1.5,'1)t(0,-diazomaWidth*1.5,-
sphendoneH_D1-seatHeight)UpperAngledWallLeft }
                                                                                |stairWidth: NIL
                                                                                |~1: NIL}

                                                                        |diazomaWidth:Block}
                        |front: reverseNormals Block}



        case n>= tier1_rows && n<y-3:
        s('8,'1,seatDepth)
        center(x)
        t(0,0,seatDepth*n+diazomaWidth)
        split(x){ ~1: NIL
                        |stairWidth: StepEnd
                        | ~1: s('1,'1, seatDepth)SeatEnd}

        case n == y-3:
        s('9,'1,seatDepth+diazomaWidth)
        center(x)
        t(0,0,seatDepth*n+diazomaWidth)
        split(x){~1:NIL
                        |stairWidth: StepEnd
t(0,0,seatDepth)s('1,'1,diazomaWidth)i("builtin:cube")comp(f){bottom:reverseNormals Block}
                        |~1: s('1,'1, seatDepth+diazomaWidth)SeatEnd}

        else:
        s('8,'1,'1)
        center(x)
        t(0,0,seatDepth*n+seatDepth)
        split(x){~1:s('1,'1, seatDepth)SeatEnd |stairWidth: StepEnd|~1: s('1,'1,
seatDepth)SeatEnd}

###########__STEP__#############
```

```
Step -->
	split(y){~1: SmallStep(split.index)|~1: SmallStep(split.index)}

SmallStep(n) -->
	s('1,'1, seatDepth)
	i("builtin:cube")
	t(0,0,n*stepDepth)
	comp(f) {top: s('1,'.5,'1)t(0,'1,0)Block |back: Block}

StepEnd -->
	split(y){~1: SmallStepEnd(split.index)|~1: SmallStepEnd(split.index)}

SmallStepEnd(n) -->
	s('1,'1, seatDepth)
	i("builtin:cube")
	t(0,0,n*stepDepth)
	comp(f) {bottom:reverseNormals s('1,'.5,'1)t(0,'1,0)Block |front:reverseNormals  Block}

###########__WALL__#############

AngledWall-->
	roofShed(angle,1) Block

AngledWallLeft -->
	 roofShed(angle,3) Block

UpperAngledWall -->
[roofShed(angle2,2)Block]
extrude(-d1H-seatHeight)Block

UpperAngledWallLeft -->
[roofShed(angle2,2)Block]
extrude(-d1H-seatHeight)Block

Block -->
	color(wall)
	tex.Block("wall",tile)

###########__SPHENDONE__#############

SphendoneD2 -->
	s('9,sphendoneH+totalH,sphendoneWidth)
	center(x)
	t(0,-totalH,totalD)
	i("builtin:cube")Sphendone

SphendoneD3 -->
	s('9,sphendoneH+totalH+sphendone2H,sphendoneWidth)
	center(x)
	t(0,-totalH-sphendoneH,totalD+sphendoneWidth)
	i("builtin:cube")Sphendone
```

```
SphendoneD4 -->
	s('16,sphendoneH+totalH+sphendone2H*2,sphendoneWidth)
	center(x)
	t(0,-totalH-sphendoneH-sphendone2H,totalD+sphendoneWidth*2)
	i("builtin:cube")Sphendone

SphendoneD2Straight -->
	case sphendoneType == "Flat" || sphendoneType == "Arcaded":
		s('1,sphendoneH+totalH,sphendoneWidth)
		t(0,-totalH,totalD)
		i("builtin:cube")Sphendone

	case sphendoneType == "Colonnaded" && scope.sx>stairWidth*1.1:
		s(scope.sx+stairWidth,sphendoneH+totalH,sphendoneWidth)
		t(0,-totalH,totalD)
		i("builtin:cube")
		Sphendone
	else:
		s(scope.sx+stairWidth,sphendoneH+totalH,sphendoneWidth)
		t(0,-totalH,totalD)
		i("builtin:cube")
		Block

EndSphendoneD2 -->
	s('9,sphendoneH+totalH,sphendoneWidth)
	t(-wallThickness,-totalH,totalD)
	i("builtin:cube")
	Sphendone

EndSphendoneD2Straight -->
	case sphendoneType == "Flat" || sphendoneType == "Arcaded":
		s(scope.sx+wallThickness,sphendoneH+totalH,sphendoneWidth)
		t(-wallThickness,-totalH,totalD)
		i("builtin:cube")
		Sphendone
	else:
		s(scope.sx+wallThickness,sphendoneH+totalH,sphendoneWidth)
		t(-wallThickness,-totalH,totalD)
		i("builtin:cube")
		comp(f){side:Block|top:ColonnadeEnd}

EndSphendoneLeftD2Straight -->
	case sphendoneType == "Flat" || sphendoneType == "Arcaded":
		s(scope.sx-wallThickness,sphendoneH+totalH,sphendoneWidth)
		t(wallThickness,-totalH,totalD)
		i("builtin:cube")
		comp(f) {side: reverseNormals Block| bottom: reverseNormals SphendoneTopLeft}
	else:
		s(scope.sx-wallThickness,sphendoneH+totalH,sphendoneWidth)
		t(wallThickness,-totalH,totalD)
		i("builtin:cube")
		comp(f) {side: reverseNormals Block| bottom: reverseNormals ColonnadeEndLeft}
```

```
EndSphendoneD2Left -->
        s('9,sphendoneH+totalH,sphendoneWidth)
        t(wallThickness,-totalH,totalD)
        i("builtin:cube")
        comp(f) {side: reverseNormals Block| bottom: reverseNormals SphendoneTopLeft}

SphendoneD1 -->
        s('9,sphendoneH_D1+d1H,sphendoneWidth)
        center(x)
        t(0,-totalH,d1D)
        i("builtin:cube")
        Sphendone

SphendoneD1End -->
        s('9,sphendoneH_D1+d1H,sphendoneWidth)
        t(-wallThickness,-totalH,d1D)
        i("builtin:cube")
        Sphendone

SphendoneD1LeftEnd -->
        s('9,sphendoneH_D1+d1H,sphendoneWidth)
        t(wallThickness,-totalH,d1D)
        i("builtin:cube")
        comp(f) {side: reverseNormals Block| bottom: reverseNormals SphendoneTopLeft}

Sphendone -->
        comp(f){side: Block |top: SphendoneTop}

SphendoneTop -->
        case sphendoneType == "Flat":
        Block

        case sphendoneType == "Colonnaded":
         Colonnade

        else:
        [Block]
        s('1, sphendone2D, sphendone2H)
        i("builtin:cube")
        Block

SphendoneTopLeft -->
        case sphendoneType == "Flat":
        Block

        case sphendoneType == "Colonnaded":
        ColonnadeLeft

        else:
        [Block]
        s('1, sphendone2D, sphendone2H)
        t(0,0,-sphendone2H)
        i("builtin:cube")
```

```
        Block

Colonnade -->
        extrude(col.colonnadeHeight)
        comp(f){top: Roof
                        |bottom: reverseNormals Block
                        |front: Wall
                        |back:split(y){col.columnHeight:
col.NoFirstLastOnBreak(col.columnSpacing)
                                                |~1:col.Entablature(col.columnSpacing,
column_diameter)}
                        }

ColonnadeLeft -->
        extrude(col.colonnadeHeight)
        comp(f){top: Roof
                        |bottom: reverseNormals Block
                        |back: Wall
                        |front:split(y){col.columnHeight:  col.LastOnBreakOnly(col.columnSpacing)
                                                |~1:col.Entablature(col.columnSpacing,
column_diameter)}
                        }

ColonnadeEnd -->
        extrude(col.colonnadeHeight)
        comp(f){top: Roof
                        |bottom: reverseNormals Block
                        |front: Wall
                        |back:split(y){col.columnHeight:  col.NoFirstLastFlush(col.columnSpacing)
                                                |~1:col.Entablature(col.columnSpacing,
column_diameter)
                                                }
                        |left:split(y){col.columnHeight: NIL
                                                |~1:col.Entablature(col.columnSpacing,
column_diameter)
                                                }
                        }

ColonnadeEndLeft -->
        s(scope.sx-stairWidth,'1,'1)
        extrude(col.colonnadeHeight)
        comp(f){top: Roof
                        |bottom: reverseNormals Block
                        |front: split(y){col.columnHeight:
col.NoFirstLastOnBreak(col.columnSpacing)
                                                |~1:col.Entablature(col.columnSpacing,
column_diameter)
                                                }
                        |back:Wall
                        |left:split(y){col.columnHeight: NIL
                                                |~1:col.Entablature(col.columnSpacing,
column_diameter)
                                                }
                        }
```

```
Roof -->
        roofGable(10,0,0,false,1)
        comp(f){side:Block
                        |top:color(roof)rf.Roof(roofBrickW,roofBrickH)
                        |bottom:Block}

Wall -->
        color(wall)
        extrude (-col.column_diameter)
        tex.Block("wall",col.columnHeight)
```

Hellenistic Houses

```
import col : "Colonnade.cga"
import tex : "TexturesAssets.cga"
import rf:   "Roof.cga"

#--------General

@Group("General",2)
attr generate_roof                    = true

@Group("General",2)@Range("TUSCAN","DORIC","IONIC","CORINTHIAN")
attr order_                                    = "DORIC"
       const tuscan                            = order_ == "TUSCAN"
       const doric                                    = order_ == "DORIC"
       const ionic                                    = order_ == "IONIC"
       const corinthian                        = order_ == "CORINTHIAN"
@Group("General",2)
attr height                           = 6

const wallThickness                   = .3
const windowW                         = .5
const windowH                         = .6
const doorW                                    = rand(.8,1)
const doorH                                    = rand(2,2.2)
#--------Colonnade
attr column_diameter          = (height-1)/12


////////////
///////START

Lot -->
alignScopeToAxes(y)
Lot(rand(.9,1.1))

Lot(n) -->
split(z){'.4*n:  Back(scope.sy,n,rand(1.9,3.1))
               |~1: Middle(scope.sy)
               |'.2*n: alignScopeToAxes(y) Front(scope.sy) }


Front(yDim) -->
extrude(world.y,100)
split(y){yDim:comp(f){side:WallSide}
               |height*.5:comp(f){front:EntranceWall
                                                      |side:WallSide
                                                      |top:Roof
                                                      }
               |~1: NIL
               }
```

```
Middle(yDim) -->
extrude(world.y,100)
split(y){yDim:comp(f){side:WallSide
                                    |top:split(x){wallThickness:
extrude(2)comp(f){all:WallSide}
                                                 |~1:
color(ground)tex.Block("ground_BW",1)
                                                 |wallThickness:
extrude(2)comp(f){all:WallSide}
                                                 }
                                    }
               |1: NIL
               }

Back(yDim,n,r) -->
extrude(world.y,100)
color(wall)
split(y){yDim:comp(f){top:color(floor)tex.WholeBlock("floor",.5)
                                                 |side:WallSide
                                                 }
                                    |height*n:split(x){'.75: split(y){'.5: comp(f){front:
Prostas(n,r) t(0,0,-2)EntranceWall

                       |side: set(trim.vertical,false)Wall

                       }

       |'.05: comp(f){top:color(floor)tex.WholeBlock("floor",.5)

                       |side:WallSide

                       |bottom: WallSide

                       }

       |'.4: comp(f){front:Prostas(n,r)Railing t(0,0,-2)EntranceWall

                       |side: set(trim.vertical,false)Wall

                       }

       |'.05: comp(f){side:WallSide

                              |top: Roof

                              |bottom: WallSide

                              }

       }
                                                 |~1: split(y){'.5:
comp(f){front:WindowWall(n)

               |side:WallSide

               }

       |'.4:comp(f){front:WindowWall(n)
```

```
        |side:WallSide

        |top: color(floor)tex.WholeBlock("floor",.5)

        }

        |'.1:comp(f){side:Wall}
                                                                                                    }

                                                                    }
                                        |~1: NIL
                                        }

Prostas(n,r) -->
color(column)
col.NoFirstNoLast(col.columnSpacing*3)//(scope.sx/r)

Wall -->
s('1,'1,wallThickness)
t(0,0,-wallThickness)
i("builtin:cube")
tex.Block("wall",col.columnHeight)

WallSide -->
tex.Block("wall",col.columnHeight)

WindowWall(n) -->
split(x){~1:WallSide
               |windowW*n:split(y){'.6:WallSide
                                        |windowH*n: color(wood)tex.WholeBlock("wood")
                                        |~1:WallSide
                                        }
               |~1:WallSide
               }

Railing -->
t(0,0,-column_diameter/2)
split(y){1:s(scope.sx-wallThickness*2,'1,.05)
                       center(xz)
                       color(wood)
                       split(x){~1:i(tex.railingAsset)Railing.}*
               |~1: NIL
               }

EntranceWall -->
split(x){~1:WallSide
               |doorW: split(y){doorH:color(wood)tex.WholeBlock("wood")
                                        |~1: WallSide
                                        }
               |~1:WallSide
               }

Roof -->
```

```
roofGable(12)
comp(f){top:color(roof)tex.Block("roof",4,24)
               |side: WallSide
               }
```

Shared Modules

Colonnade

```
import tex : "TexturesAssets.cga"


#-------General
@Group("General",0)@Range("HIGH","MED","LOW")
attr LOD                                        = "LOW"
       const highLOD                            = LOD == "HIGH"
       const medLOD                             = LOD == "MED"
       const lowLOD                             = LOD == "LOW"
@Group("General",0)@Range("TUSCAN","DORIC","IONIC","CORINTHIAN")
attr order_                                     = "DORIC"
       const tuscan                             = order_ == "TUSCAN"
       const doric                                     = order_ == "DORIC"
       const ionic                                     = order_ == "IONIC"
       const corinthian                         = order_ == "CORINTHIAN"
@Group("General",0)@Range("COLONNADE","ARCADE")
attr colonnadeType                          =       "COLONNADE"

       const columns                                = colonnadeType == "COLONNADE"
       const arches                                 = colonnadeType == "ARCADE"
@Group("General",0)
attr plinthH                                =       case arches: archWidth/2 else: 0
@Group("General",0)@Range("NONE","PERIPTERAL","DIPTERAL","CLOSED_ALAE","T-SHAPE","THOLOS")
attr peristyle_type                 =       case tuscan: "NONE"
                                                        else: "PERIPTERAL"
       const noPeristyle                    = peristyle_type == "NONE"
       const peripteral                     = peristyle_type == "PERIPTERAL"
       const dipteral                       = peristyle_type == "DIPTERAL"
       const closedAlae                     = peristyle_type == "CLOSED_ALAE"
       const Tshape                         = peristyle_type == "T-SHAPE"
       const tholos                         = peristyle_type == "THOLOS"
const tile                                      =       columnHeight
const colonnadeHeight               =       case columns:  columnHeight+entablatureH
                                                                else:
       archH+entablatureH

#-------Arcade
@Group("Arcade",1) @Range("Front Side", "Both Sides", "Off")
attr showOrders                                 = "Front Side"
       const singleSide                         = showOrders == "Front Side"
       const bothSides                                 = showOrders == "Both Sides"
       const ordersOff                                 = showOrders == "Off"
@Group("Arcade",1)
attr archWidth                              =       2

@Group("Arcade",1)
attr wallThickness                          =       .5
```

```
const archColumnDiameter              =       archWidth/10
const archColumnBaseW                 =       case arches && !ordersOff: archColumnDiameter*1.25
else: 0
const orderW                                  =       10*p
const baseDiff                                =       plinthH-archWidth/2
const m                                       =       (1.5*archWidth)/4
const p                                       =       (7*m)/75
const passageW                                =       m*2.06
const archH                                   =       1.5*archWidth

#------COLUMNS
@Group("Colonnade",2)
attr column_diameter                  =       0.6
@Group("Colonnade",2)
attr columnSpacing                            =       case tuscan:   column_diameter*4
                                                                     case doric:
        column_diameter*2.5
                                                                     case ionic:
        column_diameter*5.5
                                                                     else:
        column_diameter*9

const columnHeight                            =       case tuscan:   column_diameter*7+plinthH
                                                                     case doric:
        column_diameter*5.5+plinthH
                                                                     case ionic:
        column_diameter*9+plinthH
                                                                     else:
        column_diameter*10+plinthH

const columnOffset                            =       case columns:  (baseW*column_diameter-
column_diameter)/2
                                                                     else:
        0//archColumnBaseW/2
#(as factor of column diameter)
const baseH                                           =       case !doric:   0.5
                                                                             else:                  0
const baseW                                           =       case tuscan:   1.25
                                                                             case ionic:            1.375
                                                                             case corinthian:1.3
                                                                             else:                  1
const capitalH                                =       case tuscan:   0.5
                                                                             case doric:            0.45
                                                                             case ionic:            0.585
                                                                             else:                  1.1
const capitalW                                =       case tuscan:   1.3
                                                                             case doric:            1.1
                                                                             case ionic:            1.4
                                                                             else:                  1.3
const cornerCapitalW                  =       1.47
const cornerCapitalL                  =       1.46
const echinusPartsHeight              =       1
const abacusPartsHeight               =       1.15
const abacusOver                              =       capitalH*.34
```

```
#------ENTABLATURE (all attributes in this section are factors of column diameter)
const entablatureH                          =       case tuscan:    2.2148
                                                                    case doric:     1.7402
                                                                    else:
        architraveH+friezeH+friezeCymaH+dentilH+dentilCymaH+corniceH

#Architrave

const architraveH                           =       case tuscan:    0.9902
                                                                    case doric:             0.7309
                                                                    else:                   0.75
const architraveStepDepth           =       0.0375
const lowerFasciaH                          =       0.1607
const middleFasciaH                         =       0.2143
const upperFasciaH                          =       0.2678
const architraveCymaH               =       0.1071

#Frieze
const friezeH                               =       case tuscan:    1.2246
                                                                    case doric:             0.9348
                                                                    else:                   0.5625
const friezeCymaH                           =       0.0803

#Triglyph

const triglyphW                                     =       0.44
const triglyphD                                     =       triglyphW/9

#Dentil

const dentilH                               =       0.2143
const dentilD                               =       dentilH/2
const dentilW                               =       0.1071
const dentilSpacing                         =       0.0714
const dentilCymaH                           =       0.0357

#Cornice

const corniceH                              =       case tuscan: 1
                                                                    case doric: 0.3238
                                                                    case corinthian: .75
                                                                    else:0.2143


#Taenia

const taeniaH                               =       architraveH *0.1
const taeniaD                               =       .033

#Geison

const geisonProjection                      =       case corinthian:0
                                                                    else: 0.5
```

```
#------COLORS
@Group("Colors",9)
attr column                                     =       "#EEEEEE"
@Group("Colors",9)
attr architrave                                 =       "#E8DCC5"
@Group("Colors",9)
attr frieze                                     =       "#EBD8B5"
@Group("Colors",9)
attr metopes                                =       "#DB8C6C"
@Group("Colors",9)
attr triglyph                               =       "#576484"
@Group("Colors",9)
attr cornice                                =       "#C9C0AD"

const blue                                      =       "#0066FF"
const red                                       =       "#FF3300"
const green                                     =       "#33CC33"
const pink                                      =       "#FF6699"
const cyan                                      =       "#00FFFF"
const purple                                =       "#6600CC"
const orange                                =       "#FF9900"

///////SETUP  (should be done in importing rule)
Lot -->
case columns:
        extrude(colonnadeHeight)
        split(y){columnHeight:comp(f){front:FirstFlushNoLast(columnSpacing)|back:FirstFlushNoLast
(columnSpacing)|side:FirstFlushNoLast(columnSpacing)}
                        |entablatureH:comp(f){side:Entablature(columnSpacing,column_diameter)}
                        }
else:
        extrude(colonnadeHeight)
        split(y){~1: comp(f){front: Arcade |back: Arcade |left: Arcade|right: Arcade}
                        |entablatureH: comp(f){side:Entablature(columnSpacing,column_diameter)}
                        }

///////START

///////COLONNADE

FirstFlushNoLast(n) --> #first column flush, no last column)
        split(x){~n:t(column_diameter/2+columnOffset,0,0)ColumnTile(split.index,column_diameter,s
plit.total)
                        |{~n:ColumnTile(split.index,column_diameter,split.total)}*}

FirstFlushNoLast(d,n) --> #first column flush, no last column)-- with column diameter specified
        split(x){~n:t(d/2+ +columnOffset,0,0)ColumnTile(split.index,d,split.total)
                        |{~n:ColumnTile(split.index,d,split.total)}*}

NoFirstLastFlush(n) --> #no first column, last column flush
        split(x){~n: NIL
```

```
               |{~n:ColumnTile(split.index,column_diameter,split.total)}*
               |.01:t(-column_diameter/2-
columnOffset,0,0)ColumnTile(split.index,column_diameter,split.total)}

FirstLastFlush(n) --> #first and last columns flush
       split(x){~n:t(column_diameter/2+columnOffset,0,0)ColumnTile(split.index,column_diameter,s
plit.total)
                      |{~n:ColumnTile(split.index,column_diameter,split.total)}*
                      |.01:t(-column_diameter/2-
columnOffset,0,0)ColumnTile(split.index,column_diameter,split.total)}

FirstLastFlush(n,c) --> #first and last columns flush
       split(x){~n:t(column_diameter/2+columnOffset,0,0)ColumnTile(split.index,c,split.total)
                      |{~n:ColumnTile(split.index,c,split.total)}*
                      |.01:t(-column_diameter/2-
columnOffset,0,0)ColumnTile(split.index,c,split.total)}

FirstLastFlushFront(n,i) --> #first and last columns flush, with factor for center opening
       split(x){n:t(column_diameter/2,0,0)ColumnTile(split.index,column_diameter,split.total)
                      |{~n:ColumnTile(split.index,column_diameter,split.total)}*
                      |~n*i: ColumnTile(split.index,column_diameter,split.total)
                      |{~n:   ColumnTile(split.index,column_diameter,split.total)}*
                      |n:    ColumnTile(split.index,column_diameter,split.total)
                      |.01:t(-
column_diameter/2,0,0)ColumnTile(split.index,column_diameter,split.total)}

NoFirstLastOnBreak(n) --> #no first column, last column on break
       split(x){~n: NIL
                      |{~n:ColumnTile(split.index,column_diameter,split.total)}*
                      |.01:ColumnTile(split.index,column_diameter,split.total)}

NoFirstLastOnCorner(n) --> #no first column, last column on break
       split(x){~n: NIL
                      |{~n:ColumnTile(split.index,column_diameter,split.total)}*

       |.01:t(column_diameter/2,0,0)ColumnTile(split.index,column_diameter,split.total)}

FirstOnBreakNoLast(n) --> #first column on break, no last colum
       split(x){~n:
t(column_diameter/2+columnOffset,0,0)ColumnTile(split.index,column_diameter,split.total)
                      |{~n:ColumnTile(split.index,column_diameter,split.total)}*
                      }

FirstFlushLastOnBreak(n) --> #first column flush, last column on break
       split(x){~n:
t(column_diameter/2+columnOffset,0,0)ColumnTile(split.index,column_diameter,split.total)
                      |{~n:ColumnTile(split.index,column_diameter,split.total)}*
                      |.01:ColumnTile(split.index,column_diameter,split.total)}

FirstOnBreakLastFlush(n) --> #first column flush, last column on break
       split(x){~n: ColumnTile(split.index,column_diameter,split.total)
                      |{~n:ColumnTile(split.index,column_diameter,split.total)}*
                      |.01:t(-column_diameter/2-
columnOffset,0,0)ColumnTile(split.index,column_diameter,split.total)}
```

```
FirstLastOnBreak(n) -->
	split(x){~n:	ColumnTile(split.index,column_diameter,split.total)
			|{~n:	ColumnTile(split.index,column_diameter,split.total)}*
			|.01:	ColumnTile(split.index,column_diameter,split.total)}

NoFirstNoLast(n) --> #no first or last column
	split(x){~n: NIL
		|{~n:ColumnTile(split.index,column_diameter,split.total)}*}

LastOnBreakOnly(n) -->
		split(x){~1: NIL
			|.01:ColumnTile(split.index,column_diameter,split.total)}

FrontColonnade(n,i) -->
	split(x){n:	NIL
			|{~n:	ColumnTile(split.index,column_diameter,split.total)}*
			|~n*i: ColumnTile(split.index,column_diameter,split.total)
			|{~n:	ColumnTile(split.index,column_diameter,split.total)}*
			|n:	ColumnTile(split.index,column_diameter,split.total)
			|.01:	NIL}

InnerFrontColonnade(n,i) --> #no first or last column
	split(x){~n*2: NIL
			|{~n*.99:	ColumnTile(split.index,column_diameter,split.total)}*
			|~n*i: ColumnTile(split.index,column_diameter,split.total)
			|{~n:	ColumnTile(split.index,column_diameter,split.total)}*
			|n:	NIL
			|.01:	NIL}

AntisColonnade(n,i,a) -->
case a == 1: #front side
	split(x){n:	NIL
			|{~n:	NIL}*
			|~n*i: ColumnTile(split.index,column_diameter,split.total)
			|~n: ColumnTile(split.index,column_diameter,split.total)
			|{~n:	NIL}*
			|.01:	NIL}
else: #back side
	t (0,0,column_diameter+columnOffset)
		split(x){n:	NIL
			|{~n:	NIL}*
			|~n*i: ColumnTile(split.index,column_diameter,split.total)
			|~n: ColumnTile(split.index,column_diameter,split.total)
			|{~n:	NIL}*
			|.01:	NIL}

Propylon(n,i) -->
	FrontColonnade(n,i)
	split(x){~n:t(column_diameter/2+columnOffset,0,0)ColumnTile(split.index,column_diameter,s
plit.total)
			|~1:NIL
			|.01:t(-column_diameter/2-
columnOffset,0,0)ColumnTile(split.index,column_diameter,split.total)}
```

```
///////COLUMNS

ColumnTile(k,d,i) -->
               t(-d/2,0,-d-columnOffset)
               set(trim.vertical,false)
               s(d,'1,d)
               color(column)
               i("builtin:cube")
               Column(scope.sx,k,i)


Column(d,k,i) -->
       case tuscan && columns :
               split(y){plinthH:Plinth(d)
                              |d*baseH: Base(d)
                                     |~1: i(tex.smoothShaftAsset) tex.WholeBlock("tallBlock")
                              |d*capitalH: Capital(d,k,i)}
       case tuscan && arches :
               split(y){plinthH:Plinth(d)
                              |archColumnDiameter*baseH: Base(d)
                                     |~1: i(tex.smoothShaftAsset) tex.WholeBlock("tallBlock")
                              |d*capitalH: Capital(d,k,i)}
       case doric  :
               split(y){plinthH: Plinth(d)
                              |~1: i(tex.flutedShaftAsset)tex.WholeBlock("tallBlock")
                              |d*capitalH: Capital(d,k,i)}
       case ionic:
               split(y){plinthH:Plinth(d)
                              |d*baseH: Base(d)
                              |~1: s('1,scope.sy+d*capitalH*.34,'1) i(tex.flutedShaftAsset)
tex.WholeBlock("tallBlock")
                              |d*capitalH: Capital(d,k,i)}
       else  :
               split(y){plinthH:Plinth(d)
                              |d*baseH: Base(d)
                              |~1: i(tex.flutedShaftAsset)tex.WholeBlock("tallBlock")
                              |d*capitalH: Capital(d,k,i)}

Plinth(d)-->
       case doric:
               center(xz)
               tex.Block("tallBlock")
       else:
               s(d*baseW,'1,d*baseW)
               center(xz)
               tex.Block("tallBlock")

Base(d) -->
       case tuscan || corinthian:
               s(d*baseW,'1,d*baseW)
               center(xz)
               i(tex.baseAsset)
```

```
                tex.WholeBlock("block")
        else:
                s(d*baseW,'1,d*baseW)
                center(xz)
                split(y) {      d*baseH - d/3: i("builtin:cube") tex.WholeBlock("block")
                                | ~1: i(tex.baseAsset)tex.WholeBlock("block")}

Capital(d,k,i) -->
        case tuscan:
                i(tex.capitalAsset)
                tex.WholeBlock("block")
        case doric:
                s(d*capitalW,'1,d*capitalW) center(xz)
                split(y){ ~echinusPartsHeight: i(tex.echinusAsset) tex.WholeBlock("block")
                                | ~abacusPartsHeight : i(tex.abacusAsset) tex.Block("block") }
        case ionic && k == 0 && !tholos:
                s(d*cornerCapitalW,d*capitalH,d*cornerCapitalL) center(xz)
                i(tex.ionicCapitalCorner)
                mirror(true,false,false)
                tex.WholeBlock("block")
        case ionic && k == i-1 && !tholos:
                s(d*cornerCapitalW,d*capitalH,d*cornerCapitalL) center(xz)
                i(tex.ionicCapitalCorner)
                //mirror(true,false,false)
                tex.WholeBlock("block")
        else:
                s(d*capitalW,d*capitalH,d)
                center(xz)
                i(tex.capitalAsset)
                tex.WholeBlock("block")


////////ARCADE

ArcadeStart -->
case scope.sx>archWidth:
        extrude(-wallThickness)
        t(0,-archColumnBaseW/2,0)
        split(x){orderW: alignScopeToGeometry(yUp,4,world.lowest)Order
                        |~1:split(x){~passageW:alignScopeToGeometry(yUp,4,world.lowest)Passage
                                                |~orderW:
alignScopeToGeometry(yUp,4,world.lowest)Order}*}
else:
        extrude(-wallThickness)
        Wall

Arcade -->
case scope.sx>archWidth:
        extrude(-wallThickness)
        t(0,-archColumnBaseW/2,0)
        split(x){~1:split(x){~passageW:alignScopeToGeometry(yUp,4,world.lowest)Passage
                                                |~orderW:
alignScopeToGeometry(yUp,4,world.lowest)Order}*}
```

```
else:
        extrude(-wallThickness)
        Wall

LinearArcade -->
case scope.sx>archWidth:
        extrude(-wallThickness)
        s('1.1,'1,'1)
        center(x)
        t(0,-archColumnBaseW/2,0)
        trim()
        split(x){orderW: alignScopeToGeometry(yUp,3,world.lowest)Order

        |~1:split(x){~passageW:alignScopeToGeometry(yUp,3,world.lowest)Passage
                                                        |~orderW:
alignScopeToGeometry(yUp,3,world.lowest)Order}*}
else:
        extrude(-wallThickness)
        s('1.1,'1,'1)
        center(x)
        trim()
        Wall

LinearArcadeEnd -->
case scope.sx>archWidth:
        extrude(-wallThickness)
        s('1.1,'1,'1)
        center(x)
        t(0,-archColumnBaseW/2,0)
        trim()
        split(x){orderW: alignScopeToGeometry(yUp,4,world.lowest)Order

        |~1:split(x){~passageW:alignScopeToGeometry(yUp,4,world.lowest)Passage
                                                        |~orderW:
alignScopeToGeometry(yUp,4,world.lowest)Order}*}
else:
        extrude(-wallThickness)
        s('1.1,'1,'1)
        center(x)
        trim()
        Wall

OffsetArcade -->
case scope.sx>archWidth:
        set(trim.vertical,false)
        s(scope.sx+archColumnBaseW,'1,wallThickness)
        center(x)
        t(0,0,-wallThickness-archColumnBaseW/2)
        i("builin:cube")
        split(x){orderW: alignScopeToGeometry(yUp,1,world.lowest)Order

        |~1:split(x){~passageW:alignScopeToGeometry(yUp,1,world.lowest)Passage
                                                        |~orderW:
alignScopeToGeometry(yUp,1,world.lowest)Order}*}
```

```
else:
        set(trim.vertical,false)
        s('1,'1,wallThickness)
        t(0,0,-wallThickness)
        i("builin:cube")


Order -->
        case bothSides&& scope.sx>=archColumnBaseW : //&& geometry.isRectangular(40):
                set(trim.vertical,false)
                i("builtin:cube")
                comp(f){front:OrderFront| back: OrderBack |side: Wall}
        case singleSide && scope.sx>=archColumnBaseW :// && geometry.isRectangular(40):
                set(trim.vertical,false)
                i("builtin:cube")
                comp(f){front:OrderFront| back: Wall |side: Wall}
        else:   set(trim.vertical,false)i("builtin:cube")Wall

OrderFront -->
        alignScopeToAxes (y)
        split(x) {~1:Wall|archColumnBaseW:Wall ColumnMass|~1:Wall}
        print(archColumnBaseW)

OrderBack -->
        split(x) {~1:Wall|archColumnBaseW:Wall ColumnMass|~1:Wall}

ColumnMass -->
        extrude(archColumnBaseW)
        setPivot (yzx,3)
        s(archColumnDiameter,'1,archColumnDiameter)
        center(xz)
        i("builtin:cube")
        Column(scope.sx,1,1)

Passage -->
        split(y) {~archWidth: NIL |scope.sx/2: Arch|archWidth/4: Wall}

Arch -->
        set(trim.vertical,false)
        i (tex.arch)
        tex.WholeBlock("wall")

Wall -->

        tex.Block("wall",tile)

////////ENTABLATURE

Entablature(n,d) -->
case tuscan:
        t(0,0,-d*0.5)
        split(y){d*architraveH: Architraves(n,d)
                        |~1: Frieze(n,d)
```

```
                        }

case doric:
        //print(d*corniceH)
        t(0,0,-d*0.5)
        split(y){d*architraveH: Architraves(n,d)
                        |~1: Frieze(n,d)
                        |d*corniceH:Cornice(n,d)}
else:
        t(0,0,-d*0.5)
        split(y){d*architraveH: Architraves(n,d)
                        |~1: Frieze(n,d)
                        |d*dentilH: Dentils(d)
                        |d*dentilCymaH:Cyma(d)
                        |d*corniceH:Cornice(n,d)}

EntablatureFront(n,i,d) -->
case tuscan:
        t(0,0,-d*0.5)
        split(y){d*architraveH: ArchitravesFront(n,i,d)
                        |~1: Frieze(n,d)
                        }
case doric:
        t(0,0,-d*0.5)
        split(y){d*architraveH: ArchitravesFront(n,i,d)
                        |~1: Frieze(n,d)
                        |d*corniceH:Cornice(n,d)}
else:
        t(0,0,-d*0.5)
        split(y){d*architraveH: ArchitravesFront(n,i,d)
                        |d*friezeH+d*friezeCymaH: Frieze(n,d)
                        |d*dentilH: Dentils(d)
                        |d*dentilCymaH:Cyma(d)
                        |d*corniceH:Cornice(n,d)}

Architraves(n,d) -->
        split(x){  ~n : t('-1,0,0) s('2,'1,'1) Architrave(d)
                        | { ~ n : Architrave(d) }*
                        | ~ n   : s('2,'1,'1) Architrave(d) }


ArchitravesFront(n,i,d)-->
        split(x){n:    t('-1,0,0) s('2,'1,'1) Architrave(d)
                        |{~n:  Architrave(d)}*
                        |~n*i: Architrave(d)
                        |{~n:  Architrave(d)}*
                        |n:s('2,'1,'1) Architrave(d)}

Architrave(d) -->
        case tuscan || doric:
                color(architrave)
                t(0,0,-d*0.5)
                split(y){ ~1          : s('1,'1,d) i(tex.architraveAsset) tex.Block("block")
```

```
                              | d*taeniaH    : s('1,'1,d+d*taeniaD*1.5) i(tex.architraveAsset)
tex.Block("block") }
        else:
                color(architrave)
                t(0,0,-d*0.5)
                split(y){d*lowerFasciaH: Fascia(split.index,d)
                              |d*middleFasciaH: Fascia(split.index,d)
                              |d*upperFasciaH: Fascia(split.index,d)
                              |d*architraveCymaH: Fascia(split.index,d)}

Fascia(n,d) -->
        s('1,'1,n*d*architraveStepDepth+d)
        i(tex.architraveAsset)
        tex.Block("block")

Frieze(n,d) -->
        case tuscan:
                s(scope.sx+d,'1,d)
                center(xz)
                i(tex.friezeAsset)
                tex.Block("wall", tile)
        case doric:
                [s(scope.sx+d,'1,'1) center(x) Metopes(d)]
                t(0,0,d/2) Triglyphs(n,d)
        else:
                s(scope.sx+d+d*architraveStepDepth*3,'1,'1)
                center(x)
                s('1,'1,d+d*architraveStepDepth*3)
                center(z)
                color(frieze)
                i(tex.friezeAsset)
                tex.Block("wall", tile)

Metopes(d) -->
        s('1,'1,d+ d*triglyphW /15) center(z)
        i(tex.friezeAsset) color(metopes) tex.Block("wall", tile)

Triglyphs(n,d) -->
        color(triglyph)
        t(0,0,-0.007)
        split(x){ d*triglyphW : s('1,'1,d*triglyphD) i("builtin:cube") TriglyphWithGuttae(d)
                        | { ~ n : split(x){~1:NIL
                                                                        | d*triglyphW :
s('1,'1,d*triglyphD) i("builtin:cube") TriglyphWithGuttae(d)
                                                                | ~1:NIL
                                                                | d*triglyphW :
s('1,'1,d*triglyphD) i("builtin:cube") TriglyphWithGuttae(d)  } }* }


TriglyphWithGuttae(d)  -->

        case scope.sx < d*triglyphW*0.6:

                case split.index == 0: s(d*triglyphW,'1,'1) t('-0.5,0,0) Triglyph Guttae(d)
```

```
               else                 : t('1,0,0) s(d*triglyphW,'1,'1) t('-0.5,0,0) Triglyph
Guttae(d)
       case scope.sx < d*triglyphW*0.99:

               case split.index == 0: s(d*triglyphW,'1,'1) Triglyph Guttae(d)
               else                 : t('1,0,0) s(d*triglyphW,'1,'1) t('-1,0,0) Triglyph
Guttae(d)
       else:
               Triglyph Guttae(d)

Triglyph -->
       i(tex.triglyphAsset) tex.WholeBlock("block")

Guttae(d) -->
       case lowLOD: NIL
       else:
       t(0,-2*d*taeniaH,-0.007) s('1,d*taeniaH,d*taeniaD) i(tex.guttaeAsset)
tex.WholeBlock("block")



Dentils(d) -->
       s(scope.sx+d,'1,d)
       center(xz)
       i("builtin:cube")
       comp(f){front: split(x)       {{d*dentilSpacing: tex.Block("block")
                                                  |~d*dentilW: Dentil(d)}*
                                                  |d*dentilSpacing: tex.Block("block")}
                      |back:tex.WholeBlock("wall",tile)}

Dentil(d) -->
       case lowLOD:
               color(frieze)
               tex.WholeBlock("block")
       else:
               color(frieze)
               s('1,'1,d*dentilD)
               i("builtin:cube")
               tex.WholeBlock("block")

Cornice(n,d) -->
       s(scope.sx+d+ n *0.25- d*triglyphW ,'1,'1) center(x)
       Geisons(n,d)

Geisons(n,d) -->
       set(trim.vertical,false)
       split(x){   ~ n /4 : Mutule(n,d) t('-4,0,0) s('5,'1,'1) Geison(d)

                      | { ~ n   : split(x){ '0.25: Mutule(n,d) Geison(d)  }* }*
                      |   ~ n   : split(x){ {'0.25: Mutule(n,d) Geison(d)  }* | '0.25:
Mutule(n,d) s('5,'1,'1) Geison(d) } }

Geison(d)  -->
case corinthian:
       color(cornice)
```

```
        set(trim.vertical,true)
        t(0,0,d/2)
        s('1,'1, d/2)
        i(tex.corniceAsset)
        tex.Block("block")
else:
        color(cornice)
        set(trim.vertical,true)
        t(0,0,-d*geisonProjection)
        s('1,'1, d+d*geisonProjection )
        i(tex.corniceAsset)
        tex.Block("block")


Mutule(n,d) -->
        case lowLOD || !doric :
                NIL
        else:
                set(trim.vertical,true)
                color(triglyph)
                t(0,0,d/2+d*geisonProjection*0.2)
                s('d*triglyphW/( n /4),d*geisonProjection*0.25,d*geisonProjection*0.65) center(x)
                r(scopeCenter,8,0,0)
                i(tex.mutuleAsset)
                tex.WholeBlock("block")

Cyma(d) -->
        s('1.1,'1,d+d*taeniaD*2)
        center(x)
        t(0,0,-d/2-d*taeniaD)
        i("builtin:cube")
        tex.Block("block")
```

Roof

```
import tex: "TexturesAssets.cga" (LOD = LOD)
import col : "Colonnade.cga"

#-------General
@Group("General",0)@Range("HIGH","MED","LOW")
attr LOD                              = "LOW"
const highLOD                 = LOD == "HIGH"
const medLOD                  = LOD == "MED"
const lowLOD                  = LOD == "LOW"

@Group("General",0)@Range("TUSCAN","DORIC","IONIC","CORINTHIAN")
attr order_                                           = "DORIC"
       const tuscan                                   = order_ == "TUSCAN"
       const doric                                           = order_ == "DORIC"
       const ionic                                           = order_ == "IONIC"
       const corinthian                               = order_ == "CORINTHIAN"

const tile                                            =      columnHeight

#------Columns
@Group("Colonnade",2)
attr column_diameter                  =      1

const columnSpacing                           =      col.columnSpacing
const columnHeight                            =      col.columnHeight

#-------Roof
const roofThickness                           = 0.2

@Group("Roof")@Range("true","false")
attr antefix                          = "false"

@Group("Roof")
attr roof_angle                       =      case tuscan: 19
                                                      case ionic: 13.6
                                                             else: 15


const MidAcroH                        = column_diameter*2.5
const SideAcroH                               = column_diameter*1.75


#-------Pediment
@Group("Pediment")
attr pediment_windows                         =      0

const pedimentWidth                   =      column_diameter
#Sima
const simaLength                              =      pedimentWidth*1.1
const simaHeight                              =      case tuscan:
       pedimentWidth*0.5
```

```
                                                                case doric:
        pedimentWidth*0.27
                                                                else:
        pedimentWidth*0.5
const simaWidth                                 =       columnSpacing *0.5

#Geison
const geisonProjection                          =       column_diameter*col.geisonProjection
//column_diameter *0.5

#Cornice
const corniceH                                  =       column_diameter*col.corniceH

#-------COLORS
@Group("Colors",9)
attr wall                                       = "#EEEEEE"
@Group("Colors",9)
attr cornice                            = "#C9C0AD"
@Group("Colors",9)
attr roof                                       = "#A8763D"
@Group("Colors",9)
attr wood                                       = "#40331B"



////////////
///////START


TempleRoof(n,w,h) -->
        color(roof)
        [t(0,0.15,-0.05)Roof(w,h)]
        set(trim.vertical,false)
        split(y){ h*0.02: Antefixes(n,w,h) }


Roof(w,h) -->
        case highLOD :
                color(roof)
                set(trim.horizontal,true)
                split(y,noAdjust){ ~h : BottomBrickRow(w) }*
                split(y,noAdjust){ ~h*0.8 : TopBrickRow(w,h)
                                          | { ~h   : TopBrickRow(w,h) }*
                                          | ~h*1.2 : TopBrickRow(w,h) }
        else :
                color(roof)
                extrude(roofThickness)
                tex.Block("roof",4,24)

OverhangRoof(w,h) -->
        case highLOD :
                set(trim.horizontal,true)
                split(y,noAdjust){ ~h : BottomBrickRow(w) }*
```

```
                split(y,noAdjust){ ~h*0.8 : TopBrickRow(w,h)
                                          | { ~h   : TopBrickRow(w,h) }*
                                          | ~h*1.2 : TopBrickRow(w,h) }
        else :
                tex.Block("roof",4,24)


BottomBrickRow(w) -->
        s('1,'1.02,'1) r(scopeCenter,-3,0,0)
        split(x){ ~w : i(tex.roofBrickBottomAsset) BottomBrick. }*

TopBrickRow(w,h) -->
        t(0,0.05,w*0.1)
        split(x){ ~w*0.8   : NIL
                        | { ~w*0.4 : TopBrick(h)
                        |   ~w*0.6 : NIL }*
                        | ~w*0.4   : TopBrick(h)
                        | ~w*0.8   : NIL }

TopBrick(h) -->
        r(scopeCenter,-3,0,0) s('1,scope.sy+h*0.15,scope.sx*0.45)
        i(tex.roofBrickRound) TopBrick.

Ridge(offsetX,w,h) -->
        case highLOD :
                color(roof)
                t(offsetX,0,w*0.1)
                s(scope.sx-offsetX,w*0.5,w*0.2) center(y) i("builtin:cube")
                rotateScope(0,0,-90)
                split(y){ ~h : TopBrick(h) }*
        else:
                NIL

///////ANTEFIXES

Antefixes(n,w,h) -->
        case scope.sx < n *3:
                split(x){ ~ n : split(x){~1: Antefixa(w,h) | ~1: Antefixa(w,h) } }*
                t('1,0,0) s(w*0.4,'1,'1) Antefixa(w,h)
        else:
                split(x){   ~ n : split(x){~1: NIL      | ~1: Antefixa(w,h) }
                                | { ~ n : split(x){~1: Antefixa(w,h) | ~1: Antefixa(w,h) } }* }

Antefixa(w,h) -->
        case antefix == "true" :
                color(wall)
                s(w*0.4,h*0.2,'1) t('-0.5,0,0) i(tex.antefixAsset) Antefixa.
                tex.WholeBlock("block")
        else: NIL


TopAcroteria(w) -->
case antefix == "true" :
```

```
        i("builtin:cube")
        color(wall)
        s(simaLength/2,MidAcroH/2,MidAcroH)
        t('-1,0,0)
        center(y)
        setPivot(zxy,5)
        i(tex.acroterionAsset)
        [mirror(false,false,true)t(0,0,-w-geisonProjection) tex.WholeBlock("block")]
        tex.WholeBlock("block")


else: NIL

SideAcroteria(w) -->
case antefix == "true" :
        i("builtin:cube")
        color(wall)
        s(simaLength/2,SideAcroH/2,SideAcroH)
        t('-1,0,0)
        setPivot(zxy,5)
        i(tex.acroterionAsset)
        rotate(rel,pivot,0,0,-roof_angle)
        t('1,'.25,0)
        tex.WholeBlock("block")
        t(0,0,-w-geisonProjection)
        mirror(false,false,true)
        Anfefix.




else: NIL

///////PEDIMENT

Pediment(n) -->
        t(0,0,-pedimentWidth) s('0.995,'1,'1) center(x)
        [reverseNormals PedimentWindow]
        extrude(pedimentWidth-geisonProjection)
        alignScopeToAxes(y)
        comp(f){ top = comp(f){ 0: Simas(0) RankingGeisons(n,1) | 1: Simas(1) RankingGeisons(n,0)
}
                        //|back: reverseNormals tex.Block("wall", tile)
                        }


PedimentWindow -->
        case pediment_windows == 3:
        split(x){~1: PedimentWall|1: SmOpening|~1: PedimentWall|2.5: Opening|~1: PedimentWall|1:
SmOpening|~1: PedimentWall}

        else: PedimentWall
```

```
SmOpening -->
	split(y){1:split(x){.2:Frame|~1:NIL|.2:Frame}
			|.2: Cornice
			|~1: PedimentWall
			}


Opening -->
	split(y){2.75:split(x){.4:Frame|~1:NIL|.4:Frame}
			|.5: Cornice
			|~1: PedimentWall
			}

PedimentWall -->
	extrude(pedimentWidth-geisonProjection)
	tex.Block("wall", tile)

Cornice -->
	s('1,'1,'1.1)
	i(tex.doorCorniceAsset)
	tex.Block("block")

Frame -->
	extrude(pedimentWidth-geisonProjection)
	tex.Block("block")

Simas(side) -->
	case tuscan:
		t('-0.1*side,0,0) s('1.1,simaLength,corniceH) i("builtin:cube")
		tex.Block("block")
	else:
		t('-0.1*side,0,0) s('1.1,simaLength,simaHeight) i("builtin:cube")
		split(x){ ~simaWidth : i(tex.simaAsset) tex.Block("block")}*

RankingGeisons(n,side) -->
	case tuscan:
		t('0.03*side,0,0) s('0.97,simaLength,simaHeight)
		rotateScope(-90,180,0)
		s('1,simaHeight*0.9,column_diameter)
		t(0,'-1,0) i("builtin:cube")
		tex.Block("block")
	else:
		t('0.03*side,0,0) s('0.97,simaLength,simaHeight)
		rotateScope(-90,180,0)
		s('1,corniceH*0.9,1) t(0,'-1,0) i("builtin:cube")
		split(x){ ~ n/4.3: Geison }*


Geison -->
case corinthian:
	color(cornice)
	set(trim.vertical,true)
	t(0,0,column_diameter/2)
```

```
        s('1,'1, column_diameter/2)
        i(tex.corniceAsset)
        tex.Block("block")
else:
        color(cornice)
        set(trim.vertical,true)
        t(0,0,geisonProjection- column_diameter/2 ) s('1,'1, column_diameter )
        i(tex.geisonAsset)
        tex.Block("block")


Beam -->
        color(wood)
        i("builtin:cube")
        s('1,column_diameter,column_diameter*1.5)
        center(y)
        t(0,0,-column_diameter*1.5)
        tex.Block("wood")

Ceiling -->
        reverseNormals
        color(roof)
        split(y){pedimentWidth-geisonProjection*2: TiltRoof1 |~1: NIL | pedimentWidth-
geisonProjection*2:TiltRoof2}

TiltRoof1 -->
        set (trim.horizontal,true)
        roofShed (15)
        tex.Block("roof", 4,24)

TiltRoof2 -->
        set (trim.horizontal,true)
        roofShed (15,2)
        comp(f) {top:tex.Block ("roof",4,24)}
```

Bibliography

Akurgal, E. (1978). *Ancient Civilizations and Ruins of Turkey*. Haset Kitabevi.

Azhar, S. (2011). Building Information Modeling (BIM): Trends, Benefits, Risks, and Challenges for the AEC Industry. *Leadership and Management in Engineering* 11, no. 3, pp.241–52.

Bentkowska-Kafel, A., and Denard, H., eds. (2012). *Paradata and Transparency in Virtual Heritage*. AHRC ICT Methods Network Series, Digital Arts and Humanities. Ashgate.

Billows, R. (2005). Cities. In *A Companion to the Hellenistic World*, pp. 196–215. Blackwell Publishing Ltd.

Bingöl, O. (2007). *Magnesia Ad Maeandrum*. Homer Kitabevi.

Boeykens, S. et al. (2012). A Case Study of Using BIM in Historical Reconstruction. The Vinohrady Synagogue in Prague. https://lirias.kuleuven.be/handle/123456789/350340.

Böhler, W., and Marbs, A. (2004). 3D Scanning and Photogrammetry for Heritage Recording: A Comparison, *Proceedings of the 12th International Conference on Geoinformatics*, 291–98.

Briant, P. (2002*). From Cyrus to Alexander: A History of the Persian Empire*. Eisenbrauns.

Burry, M. (2003). Between Intuition and Process: Parametric Design and Rapid Protyping. In *Architecture in the Digital Age: Design and Manufacturing*, edited by Branko Kolarevic. Spon Press.

Chevrier, C., & Perrin, J.-P. (2009). Generation of Architectural Parametric Components. In *Joining Languages, Cultures and Visions, CAAD Futures* 2009, edited by T. Tidafi and T. Dorta, 119–32. Les Presses de l'Université de Montréal.

Cousin, G., & Deschamps, G. (1889). Lettre de Darius fils d'Hystaspes. *Bulletin de correspondance hellénique*, 13/1: 529–42.

Del Giudice, M. & Osello, A. (2013). BIM for Cultural Heritage. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-5/W2, pp.225–229.

Detienne, M. (1986). Apollo und Dionysos in der grieschischen Religion. *Die Restauration der Götter: Antike Religion und Neo-Paganismus*, pp. 124–31. Konigshausen + Neumann.

Detienne, M. (2001). Forgetting Delphi between Apollo and Dionysus. *Classical Philology*, 96/2: 147–58.

Dylla, K. et al. (2009). Rome reborn 2.0: a framework for virtual city reconstruction using procedural modeling techniques. *Proceedings of Computer Applications and Quantitative Methods in Archaeology (CAA)*.

Fai, S., Filippi, M. & Paliaga, S. (2013). Parametric Modelling (BIM) for the Documentation of Vernacular Construction Methods. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-5/W1, pp.115–120.

Favro, D. (2006). In the eye of the beholder: VR models and Academia. In L. Haselberger & J. Humphrey, eds. *Imaging Ancient Rome: documentation, visualization, imagination*. Journal of Roman Archaeology Supplementary Series. Portsmouth, R.I.: Journal of Roman Archaeology, pp. 321–334.

Foley, J. et al. (1993). *Introduction to Computer Graphics*. Addison-Wesley Professional.

Fontenrose, J. (1959). *Python: A Study of Delphic Myth and Its Origins*. University of California Press.

Frischer, B., and Dakouri-Hild, A., eds. (2008). *Beyond Illustration: 2D and 3D Digital Technologies as Tools for Discovery in Archaeology*. BAR International Series 1805. Oxford: Archaeopress.

Haegler, S., et al. (2009). Procedural Modeling for Digital Cultural Heritage. *EURASIP Journal on Image and Video Processing*.

Hamilton, W. J. (1842). *Researches in Asia Minor, Pontus and Armenia: With Some Account of Their* ... J. Murray.

Haselberger, L. (2002) *Mapping Augustan Rome*. Journal of Roman Archaeology.

Heimberg, U. (1985). Griechische Und Römische Landvermessung. *Diskussionen Zur Archdologischen Bauforschung* 4, pp. 277–96.

Henrichs, A. (1978). Greek Maenadism from Olympias to Messalina. In *Harvard Studies in Classical Philology*, 82: 121–60.

Hoepfner, W. and Schwandner, E-L. (1986). *Haus und Stadt im klassischen Griechenland*. Deutscher Kunstverlag.

Humann, C. (1904). *Magnesia am Maeander*. Berlin.

Imhoof-Blumer, F. (1895). Der dendrophoros auf Münzen von Magnesia in Ionien. *The Numismatic Chronicle*, 15: 284–6.

Johanson, C. (2009). Visualizing History: Modeling in the Eternal City. *Visual Resources: An International Journal of Documentation*, 25(4), pp.403–418.

Kadobayashi, R., et al. (2004). Comparison and Evaluation of Laser Scanning and Photogrammetry and Their Combined Use for Digital Recording of Cultural Heritage. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 35, no. 5, pp. 401–6.

Keil, J. (1908). Zur Topographie der ionischen Küste südlich von Ephesos. *JÖAI* 11 (Beibl.). pp.135-68.

Kern, O. (1895). Dionysos in Magnesia. *Beiträge zur Geschichte der griechischen Philosophie und Religion*, pp. 79–101. Berlin.

Kern, O. (1900). *Die Inschriften von Magnesia am Meander*. Berlin: W. Spemann.

Kondis, J. (1958). Zum Antiken Stadtbauplan von Rhodos. *AthMitt* 73, p.153.

Koolhaas, R. et al. (2000). *Mutations*. ACTAR.

Kostof, S. (1991). *The City Shaped: Urban Patterns and Meanings through History*. London: Thames and Hudson.

Leake, W. M. (1824). *Journal of a tour in Asia Minor: with comparative remarks on the ancient and modern geography of that country*. J. Murray.

Lohmann, H. (2006). Leucophrys. In Cancik, H. and Schneider, H., eds., *Brill's New Pauly*. Brill Online. <http://referenceworks.brillonline.com/entries/brill-s-new-pauly/leucophrys-e12223550>.

Long, C. R. (1987). *The Twelve Gods of Greece and Rome*. Brill Archive.

Maass, E. (1891). Theokrits Dionysos aus einer Inschrift Erläutert. *Hermes: Zeitschrift für klassische Philologie*, 178–90.

MacDonald, W.L. (1988*). The Architecture of the Roman Empire: An Urban Appraisal*. Yale University Press.

Martin, R. (1956). *L'urbanisme dans la Grèce antique*. A. & J. Picard.

Martin, R. (1973). Rapports Entre Les Structures Urbaines et Les Modes de Division et D'exploitation Du Territoire. *Civilisations et Sociétés* 33.

Mathias, M. et al. (2012). Automatic Architectural Style Recognition. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII-5/W16, pp.171–176.

McCarty, W. (2004). Modeling: A Study in Words and Meanings. In Schreibman, S., Siemens, R., and Unsworth, J., eds., *A Companion to Digital Humanities*. Blackwell.

McGuire, M.B. (2008). *Lived Religion: Faith and Practice in Everyday Life*. Oxford University Press.

Mitchell, W. J. (1990). *The Logic of Architecture: Design, Computations, and Cognition*. MIT Press.

Monedero, J. (2000) Parametric Design: A Review and Some Experiences. *Automation in Construction* 9, no. 4 (July): 369–77.

Mueller, P. et al. (2006). Procedural modeling of buildings. *ACM SIGGRAPH 2006 Papers*. Boston, Massachusetts: ACM, pp. 614–623.

Nilsson, M. P. (1961). *Greek Folk Religion*. Harper Torchbooks.

Parrish, D. (2001) Introduction: The Urban Plan and Its Constituent Elements. In *Urbanism in Western Asia Minor: New Studies on Aphrodisias, Ephesos, Hierapolis, Pergamon, Perge and Xanthos*, 8–41. Journal of Roman Archaeology Supplementary Series 45.

Philippson, A. (1936). *Milet III/V: Das Südliche Ionien*. De Gruyter.

Piegl, L. (1991) On NURBS: A Survey. *IEEE Computer Graphics and Applications* 11(01), pp.55-71.

Polignac, F. de. (1995). *Cults, Territory, and the Origins of the Greek City-State*. University of Chicago Press.

Prusinkiewicz, P, Lindenmayer A. et al. (1996). *The Algorithmic Beauty of Plants*. Springer.

Raja, R. and Rüpke. J. (2015). Archaeology of Religion, Material Religion, and the Ancient World. In Raja, R., and Rüpke, J., eds., *A Companion to the Archaeology of Religion in the Ancient World*, pp. 1–25. John Wiley & Sons, Ltd.

Raoul-Rochette, D. (1845). *Considérations archéologiques et architectoniques sur le temple de Diane Leucophryne récemment découvert à Magnésie du Méandre*.

Ratté, C. (2002). The Urban Development of Aphrodisias in the Late Hellenistic and Early Imperial Periods. *Patris Und Imperium*, pp. 5–32. Peeters.

Rayet, O., & Thomas, A. (1877). *Milet et le Golfe Latmique: Tralles, Magnésie du Méandre, Priene, Milet, Didymes, Heraclee du Latmos ; fouilles et explorations archéologiques ...*, Vols. 1-2. Paris.

Reinach, S. (1890). Oracle de la Pythie de Delphes adressé à la ville de Magnésie du Méandre. *Revue des études grecques* III, 349–61.

Rennell, J. (1831). *A treatise on the comparative geography of western Asia*. Rivington.

Richardson, L. (1992). A New Topographical Dictionary of Ancient Rome. JHU Press.

Rickert, T. (2014). Parmenides, Ontological Enaction, and the Prehistory of Rhetoric. *Philosophy and Rhetoric* 47, no. 4, pp. 472–93.

Robert, L. (1977). Documents d'Asie Mineure. *Bulletin de correspondance hellénique*, 101/1: 43–132.

Robert, L. (1978). Documents d'Asie Mineure. *Bulletin de correspondance hellénique*, 102/1: 395–543.

Rogers, D. (2000). *An Introduction to NURBS: With Historical Perspective*. Elsevier.

Saldaña, M., and Johanson, C. (2013). Procedural Modeling for Rapid-Prototyping of Multiple Building Phases. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XL–5/W1, pp.205–10.

Scherrer, P. (2001). The Historical Topography of Ephesos. In *Urbanism in Western Asia Minor: New Studies on Aphrodisias, Ephesos, Hierapolis, Pergamon, Perge and Xanthos*, 57–87. Journal of Roman Archaeology Supplementary Series 45.

Schinko, C., et al. (2015). Built by Algorithms - State of the Art Report on Procedural Modeling. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* XL–5/W4, pp.469–79.

Schulz, S. (1975). Die Münzprägung von Magnesia am Mäander in der römischen Kaiserzeit. Hildesheim ; New York: Olm.

Schumacher, P. (2008). *Parametricism as Style -- Parametricist Manifesto*. Available at: http://www.patrikschumacher.com/Texts/Parametricism%20as%20Style.htm [Accessed September 3, 2014].

Scully, V. (1979). *The Earth, the Temple, and the Gods: Greek Sacred Architecture*. Yale University Press.

Stiny, G. & Gips, J. (1972). Shape Grammars and the Generative Specification of Painting and Sculpture. In O. Petrocelli, ed. *The Best Computer Papers of 1971*. Philadelphia: Auerbach, pp. 125–135.

Szántó, E. (1906). *Ausgewählte Abhandlungen*. Mohr.

Texier, C. (1849*). Description de l'Asie Mineure: faite par ordre du gouvernement français en 1833 - 1937; beaux-arts, monuments historiques, plans et topographie des cités antiques*, Vol. 3. Paris.

Thallera, W. et al. (2013). Creating Procedural Window Building Blocks Using the Generative Fact Labeling Method. *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1.1, pp.235–242.

Thonemann, P. (2007). Magnesia and the Greeks of Asia (I. Magnesia 16.16). *Greek, Roman, and Byzantine Studies* 47, no. 2: 151–60.

Thonemann, P. (2011*). The Maeander Valley: A Historical Geography from Antiquity to Byzantium*. Cambridge University Press.

Ustinova, Y. (2009). *Caves and the Ancient Greek Mind: Descending Underground in the Search for Ultimate Truth*. Oxford University Press.

Vanegas, C. A. et al. (2012). Inverse Design of Urban Procedural Models. *ACM Trans. Graph.*, 31(6), pp.168:1–168:11.

Vitruvius Pollio. (1999). *Vitruvius: Ten Books on Architecture*. Cambridge University Press.

Vitruvius. (1931). *De Architectura*. (F. Granger, Tran.). Harvard University Press.

Weinberg, F. M. (1986). *The cave: the evolution of a metaphoric field from Homer to Ariosto*. Studies in the humanities. New York: P. Lang.

Wilamowitz, U. (1900). Compte Rendu Des Inscriptions de Magnésie. *Göttingische Gelehrte Anzeigen*.

Williamson, C. (2012). *City and Sanctuary in Hellenistic Asia Minor. Constructing Civic Identity in the Sacred Landscapes of Mylasa and Stratonikeia in Karia.* PhD Dissertation, University of Groningen.

Wilson Jones, M. (2000) Genesis and Mimesis: The Design of the Arch of Constantine in Rome. Journal of the Society of Architectural Historians 59, no. 1: 50–77.

Wilson Jones, M. (2009). Principles of Roman Architecture. Yale University Press.

Wycherley, R. E. (1942). The Ionian Agora. *The Journal of Hellenic Studies* 62, pp.21–32.

Wycherley, R. E. (1964). The City of Rhodes. *RIBA Journal* 71, no. 2, pp.71–73.

Zevi, B. (1957). *Architecture as Space*. M. Gendel, trans. New York.