A Novel Level-of-Detail Technique for Virtual City Environments: Design and Evaluation

Ankit Singh

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science & Applications

April 26nd, 2012
Blacksburg, Virginia

Keywords: Virtual City Environments, X3D, CityGML, 3D Blacksburg, LOD techqniues

A Novel Level-of-Detail Technique for Virtual City Environments: Design and Evaluation

Ankit Singh

ABSTRACT

Virtual City Environments (VCEs) and Mirror Worlds can be a useful resource for communities such as the local government, researchers and the general public to collaborate on tasks like town planning, threat assessment, commerce and research. There are open standards like Extensible 3D (X3D, which represents 3D graphics) and CityGML (a Geography Markup Language to manage 3D building data). These standards are royalty-free and used to create, manage, share and portray such environments. However, there are critical challenges to delivering such complex and detailed Mirror Worlds in real-time.

In this work, we focus on runtime data structures and performance for Level-of-Detail (LOD) management and real-time portrayal. We begin with a VCE defined in existing semantic models such as the CityGML specification. We implement and evaluate a novel X3D-based Level-of-Detail technique called *ProxyPrismLOD*, which leverages the CityGML standard of a 4-step LOD hierarchy. For switching between different models of the same object at near ranges, our LOD technique uses a custom shape we call a *ProxyPrism* to optimally encapsulate irregularly and asymmetrically shaped building models.

First, we ran a user study to understand the visual dynamics of range-based LOD switching. Specifically, we evaluated several scaling factors for an exponential range cutoff function. The function is based on the model's size as well as the environment density. In this experiment, participants rated "visual granularity" and "distraction" levels of the LOD technique over two Software Field-of-View (sFOV) conditions. A scaling factor of $\boldsymbol{\beta = 3}$ was determined. Second, we ran a series of simulations to study the performance benefits of *ProxyPrismLOD* technique over the basic range-based LOD. We observed performance benefits up to $\mathbf{7.46\%}$ in terms of overall Frames-per-Seconds (FPS) on the models we tested.

Acknowledgments

I am thankful to my advisor, Dr. Nicholas F. Polys whose encouragement, support and guidance from the beginning helped me develop an understanding of the subject. Without his cheerful and supportive advising this thesis would not have been possible. I would also like to thank Peter M. Sforza and folks at CGIT for giving valuable resources and input on my thesis and 3D Blacksburg project. I thank Dr. Chris L. North for his support and advice during my studies here at Virginia Tech and for serving on my committee.

I thank Department of Computer Science at Virginia Tech and the staff of the graduate program office for being accommodating to my queries.

My parents deserve special mention for their support and sacrifices which helped me continue my studies.

Contents

List of Abbreviations

FPS	Frames-per-Second
LOD	Level-of-Detail
OGC	Open Geospatial Consortium
sFOV	Software Field-of-View
VCE	Virtual City Environment
VE	Virtual Environment

List of Figures

List of Tables

List of Algorithms

Chapter 1

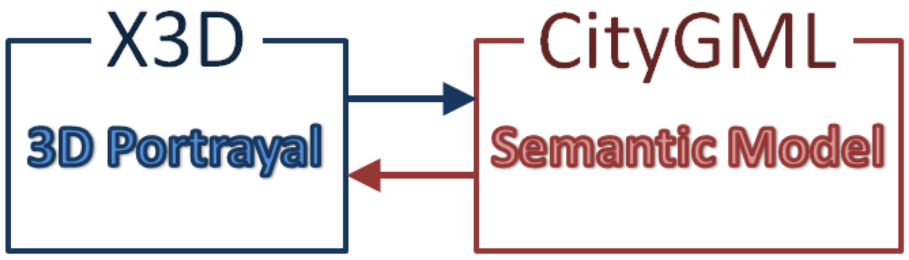
Introduction

We have seen many different approaches to managing Level-of-Detail (LOD) in applications which use computer graphics. These techniques are often application specific and are made to specifically work by optimizing the performance and quality tradeoffs of the application itself. Even though these techniques use ”traditional” LOD approaches [1], they try to use algorithms which are specifically optimized for the application in use. For example, a video game might use ”tessellation” [2] as a method to optimize the performance of the 3D graphics. Whereas a map application [3, 4] will use a tile based LOD management for 2D graphics at a longer range and then switch to range based LOD management at closer range for 3D building models. This shows that in order to get the most optimized performance many different approaches have to work together.

Also, in communities like Web3D [5] and OGC (Open Geospatial Consortium) [6], there has been a continuous development of open-standards like X3D [7, 8], CityGML [9, 10, 11] etc. X3D serves as the open-standard for representing 3D scenes and objects, CityGML represents 3D data for the buildings, and is oriented to managing building models and urban objects. X3D standard serves as a 3D portrayal platform, and requires standards like CityGML which store the semantics of the 3D objects in order to better manage these environments (see Figure 1.1).

We propose a novel LOD technique for Virtual City Environments (VCEs), which is designed and implemented in X3D and optimized specifically for VCEs to provide faster and more specific LOD management: by incorporating the semantic definitions and structure of 3D buildings in the LOD technique. We have used the CityGML standard as the basis for describing 3D building models in different LOD and use switching technique

Figure 1.1: X3D and CityGML Interoperability.



specifically tailored for VCEs. For future discussions, we will refer the technique implemented and discussed here in this work as *ProxyPrismLOD* technique.

1.1 Contributions

There are many different LOD techniques in use by different applications. Many standards which support 3D graphics implement simple LOD techniques to improve performance of rendering engines. For example, SEDRIS standard [12] implements different LOD techniques based on Distance, Spatial Resolution, Map Scale and Volume whereas, X3D implements simple Tile-based and Range-based LOD techniques. In this work we try to evaluate a novel LOD technique which is designed specifically for VCEs. The main contributions of this work are listed below.

- Implement a novel LOD technique, *ProxyPrismLOD* which uses simple and easy-to-use definitions to effectively manage highly detailed 3D data as well as optimize the performance in rendering engines.
- Evaluate the benefits in terms of performance of using a *ProxyPrism* shape for LOD switching between highest detailed LOD building models.
- Devise a switching function for implementing switching across all LOD models. This function takes into account the size of the building model and the size of the geographical coverage.

- Evaluate the effect of Software Field-of-View (sFOV) [13] on the switching function used in LOD technique and empirically evaluate an optimal value in terms of ”visual continuity” and ”distraction” levels.
- Using the collected data, we revisited the *ProxyPrismLOD* and added the ability to automatically calculate LOD cutoffs depending upon size and priority levels. This updated LOD prototype will help make the process of managing the environments easier with minimal involvement from the designers.

In subsequent chapters we present a detailed evaluation of the performance and quality tradeoffs of our technique and the background and motivation behind the features implemented. Organization of report is as follows:

Background
Discuss existing LOD techniques and open standards and how they can be tied together for better LOD management.

Challenges & Research Focus
Identify specific challenges faced in existing LOD techniques in terms of performance and LOD management.

***ProxyPrismLOD* Technique: Design & Implementation**
Discuss in detail the features and implementation of *ProxyPrismLOD* technique.

Virtual City Generator
Discuss the workings of virtual city generating script which helped in evaluating our *ProxyPrismLOD* technique.

***ProxyPrismLOD* Technique: Evaluation**
Discussion of the user study for evaluating the effect of sFOV and switching distance on ”visual continuity” and ”distraction” levels. This chapter also discusses the performance evaluation of *ProxyPrism* shape.

Discussions & Future Work
Discuss our findings and revisit the ProxyPrismLOD technique incorporating improvements in terms of LOD management. Also discuss future work in terms of integration of *ProxyPrismLOD* technique in 3D Blacksburg project.

Chapter 2

Background

In this chapter we discuss in detail the background behind *ProxyPrismLOD* technique. We will discuss ”traditional” LOD techniques in use [1], various LOD selection factors and analyze these methods in terms of complexity of implementation and computational complexity at run-time. We will also discuss open-standards like X3D and CityGML, and how they play an important role in our *ProxyPrismLOD* technique. Our main goal is to deploy our LOD technique in an ongoing project called 3D Blacksburg [14], which is a Mirror World [15, 16] of Blacksburg for use in many communities for collaboration. Finally, a brief background on the project will be given.

2.1 LOD Management

This section describes the fundamental LOD techniques most commonly used in practice. We study the LOD techniques and selection factors in terms of user-involvement, complexity of implementation and computational load at run-time. We also analyze the various tradeoffs associated with these techniques such as ease-of-use, visual granularity and implementation details etc. Figures 2.3 and 2.4 shows the classification of techniques and selection factors and how they rate on the scale of implementation complexity and computational load.

2.1.1 LOD Techniques

Here we discuss the ”traditional” LOD techniques which are used in many applications today.

1. **Discrete LOD**
   This technique was proposed by [17], and this approach uses different versions of the same 3D object with varying LOD. These different versions are rendered based depending upon many factors, one of which is distance from the user. For example, observing a distant object, the user will not be able to see the quality of the model, so a less detailed model would be rendered, see Figure 2.1 ([18]). However, observing an object very closely, the user can make out the details in its texture and quality, hence a highly detailed model would be rendered.

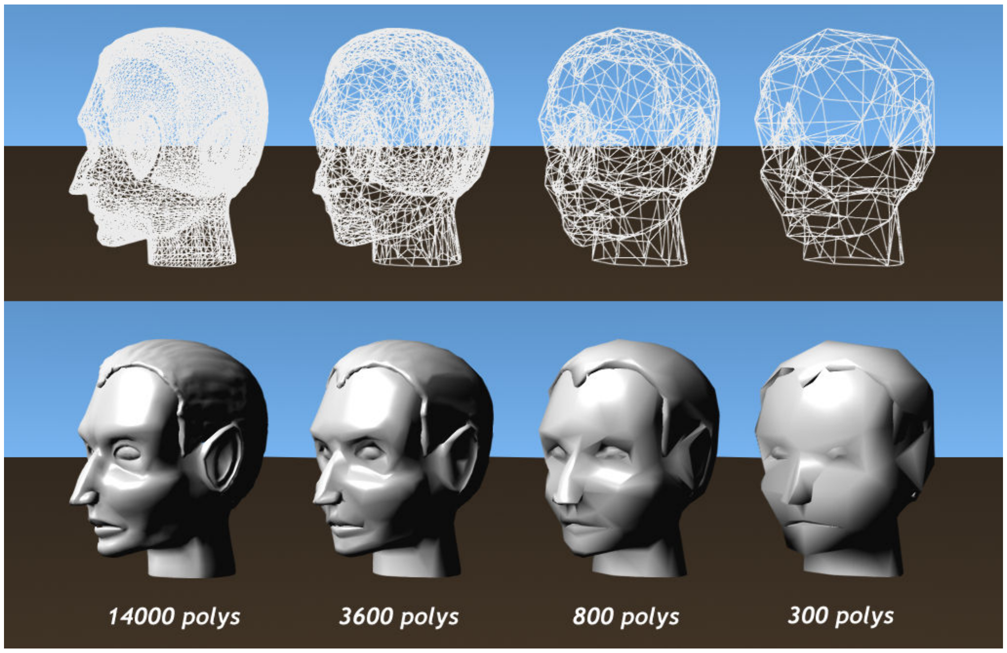
   This approach is very straightforward and simple to use, as it contains a fixed number of discrete levels and the LOD technique only has to choose the most appropriate level depending upon the selection factor. However, the process of making discrete LODs from a reference model can be done in different ways:

   (a) The user creates these levels manually, while creating the base model. This approach is the easiest to implement as the responsibility of model creation is on the user, and the LOD techniques just has to reference the appropriate model.

   (b) The LOD technique preprocesses the models to create appropriate discrete level models. This requires complex algorithms which can work with different building models without any intervention from the user. This approach is complex to implement and is also computationally more intensive.

2. **Continuous LOD**
   Instead of creating the discrete LOD at the preprocessing stage or have different models of the same building, this techniques turns the model into a data structure which can divide the model structure and create a continuous stream of LOD [19] depending upon the complexity of each sub-structure, see Figure 2.2 ([1]). Depending upon the complexity of the approach used, there are a few ways this technique can be implemented:

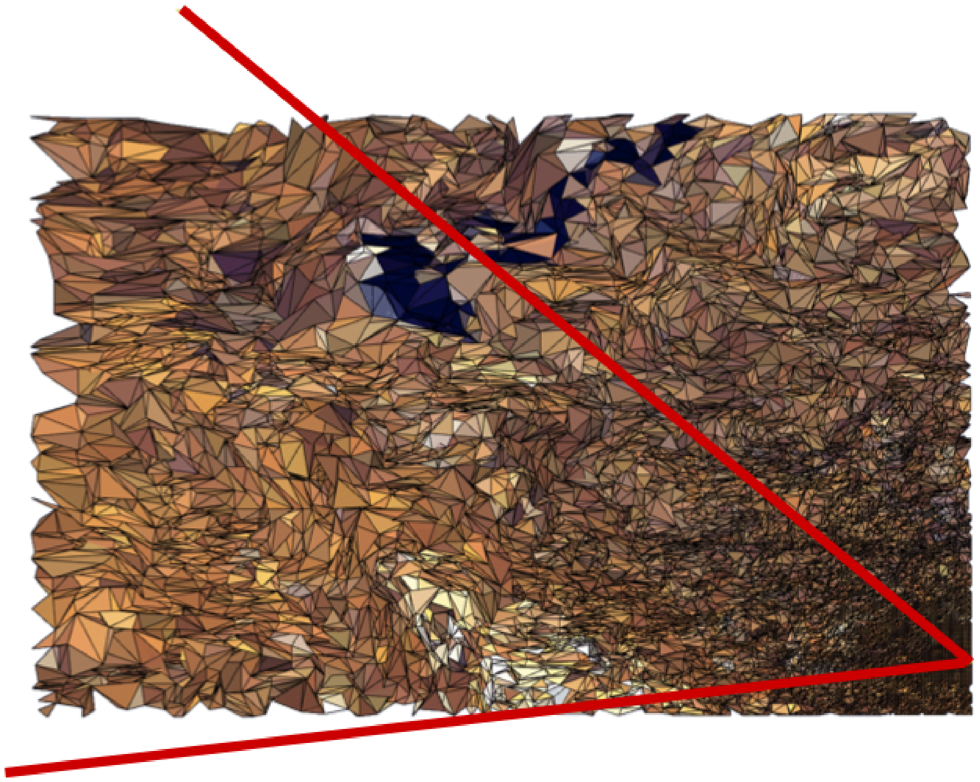
Figure 2.1: Discrete LOD: difference in polygon count across all levels.



(a) One way is to create a model into a stream of continuous LOD. It is somewhat similar to Discrete LOD, but instead of having a fixed number of levels, this framework offers many levels occurring at very short intervals. During run-time the system fetches the appropriate LOD from the data structure. Because of having numerous levels in a short interval space, this system is capable of delivering much better visual granularity at run-time. LOD delivery occurs by supplying the base model and progressively refining the model as requested by the software.

(b) Another way used today, is in the form of tessellation [2]. In this method each surface is divided into many sub-surfaces and depending upon the quality of LOD needed these sub-surfaces are refined to further levels. This approach is computationally more intensive.

Figure 2.2: Continuous LOD: decrease in polygon count as we move away from the user.



2.1.2 Tradeoffs

It is important to study the tradeoffs involved in these techniques because the application requirements can drive the choice of LOD technique (see Figure 2.3).

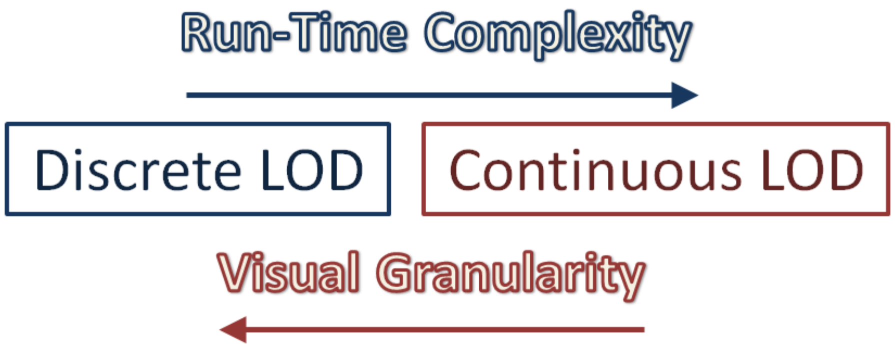
1. **Discrete LOD**
   This technique requires the least amount of run-time processing and most of the work is done in the pre-processing stage. However, this approach does not provide a very good visual granularity and the LOD selection factor has to be chosen carefully to compensate for that. Also, this requires a lot of user input in defining the levels, which can be avoided if it is integrated into design cycle of the models [20].

2. **Continuous OD**
   This framework requires extra preprocessing to create data structures which will deliver appropriate LOD, as well as for evaluation, simplification and refinement of the data model at run-time. Also, in some approaches like tessellation, run-time

complexity and computational load also increases. Hence, the cost of processing and client memory use makes this technique less adopted, unless a better granularity than discrete LOD is needed.

Due to the problems discussed above and difficulty of implementation, Discrete LOD is the most commonly LOD technique used in practice. In this work we use a combination of Discrete LOD and an optimized custom shape called *ProxyPrism* in our LOD technique. We will show that this provides a easy-to-use and yet efficient LOD delivery mechanism for VCEs.

Figure 2.3: LOD Techniques: tradeoffs.



2.1.3 LOD Selection Factors

We discussed the commonly used LOD techniques in the last section and characterized them in terms of their implementation complexity and run-time computational load. All the techniques discussed above still require an algorithm, based on which it chooses to switch between different LOD models. Depending upon the choice of these metrics the ”visual perception” of the LOD switching can be reduced, and on the same hand these parameters can significantly contribute to the run-time computational load. We will discuss these selection factors, (see Figure 2.4) in detail and their tradeoffs.

1. **Distance**
   Range-based metric is the easiest way to switch between different LODs [21, 22, 23]. Each LOD had a predefined ”cutoff” distance which indicates the distance below

which the model in rendered, and switches to a less detailed model if the user is at a larger distance than the cutoff. This is also the least computational heavy LOD selection factor, with only a simple distance comparison of each object per frame required.

This framework is implemented by storing different LOD models in a data structure along with a LOD threshold associated with it. X3D implements a range-based LOD node which is discussed later on in this chapter.

2. **Model Size**
Sometimes, the size of the model also plays an important role in LOD switching. By size, we mean the real dimensions relative to other models in the environment. For example, the size of a house will be comparatively much larger to the size of a lamp post and it would be unwise to treat them equally in terms of LOD switching. LOD switching cutoffs should directly proportional to their size to provide a better LOD switching of the environment. This selection factor is meant to be used in addition to the pure range-based approach [24].

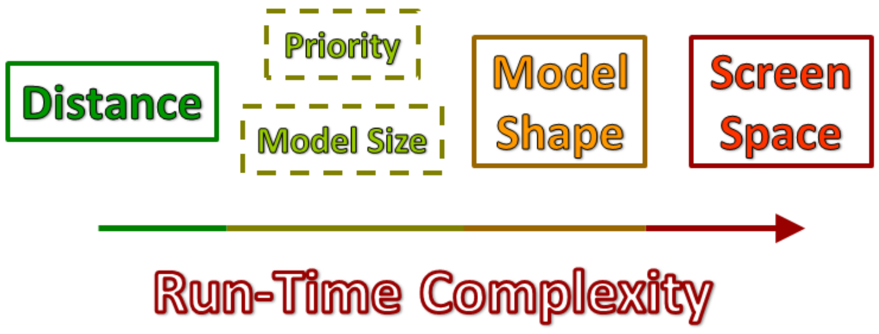
3. **Model Shape**
Another factor which should be considered is the model shape [25]. Many models are not regular in shape and using a symmetric selection factor like distance (which is equivalent to a bounding sphere) does not optimally envelope the building model. Using a bounding box is an improvement over range-based approach, but still does not work well with odd shaped or asymmetric buildings. Using a custom *ProxyPrism* shape to encapsulate a building model addresses this but it adds run-time computational load as the position of the user must be tested to be interior to the *ProxyPrism* shape.

4. **Screen Space**
Another approach used is the size of the object in terms of percentage of the screen occupied (pixels occupied by the object relative to the total screen resolution). It extends the idea of range-based switching by actually measuring the amount of space acquired by a particular object on the screen, rather than measuring the user's position from an arbitrary point. This results in a more generic and accurate method for LOD switching compared to simple range-based technique. However, this technique is quite computational heavy, as it requires a number of vertices of the object be projected on the screen space [26]. For example even a simple box will require

projecting 8 different vertices on the screen and calculating the resulting area, which takes far more time compared to simple user-model distance calculation.

5. **Priority**
Sometimes there are many objects being rendered in the environment which are critical for the perception of the environment. For instance, taking a virtual city tour of Paris will be greatly affected if the Eiffel Tower is reduced to a cylinder at a large distance [27]. Therefore some objects, such as landmarks in a VE carry a higher priority level so that they are degraded the least across distances, thus preserving the perception of the environment.

Figure 2.4: LOD Selection Factors: run-time complexity.



In our approach we have used the Discrete LOD framework and range-based selection factor for simplicity and to keep the task of LOD switching computationally less heavy. For switching between the most complex models, we use a custom shape called *ProxyPrism* which optimizes the spatial bounds of the model so that unnecessary switching is avoided.

2.2 X3D Open Standard

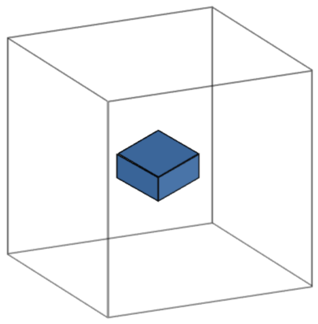
X3D is a royalty free XML-based file format used to represent 3D scenes and objects [7, 8]. For the purpose of development, X3D supports a lot of features for authoring environments which visualize large datasets like protein structures, 3D medical scans [28], VCEs which depict real cities [29] and tools for architectural studies [30]. Also it has: an extensive support for performance data collection, JavaScript for development and support on many platforms brought out by X3D rendering engines like Instantreality engine [31, 32]. X3D was used as the platform for developing the *ProxyPrismLOD* technique.

X3D does have support for LOD switching. These techniques are simple and used to optimize performance of the renderer:

1. **Bounding Box**
   Many engines use bounding boxes for cutting rendering operations. All X3D GroupingNode types include fields for defining bounding box center and size. These cubical boxes are used to help give the renderer a volume for rendering the object when inside the bounding box limits (see Figure 2.5a). They are used with two field definitions, the center which is a SFVec3f and gives the center of the bounding box in the local geometry and the size which gives the size or extent of the bounding box for x, y and z axis respectively (see Figure 2.5b).

Figure 2.5: Bounding Box

(a) Bounding Box limits (not visible by default)



(b) BoundedNode X3D Declaration

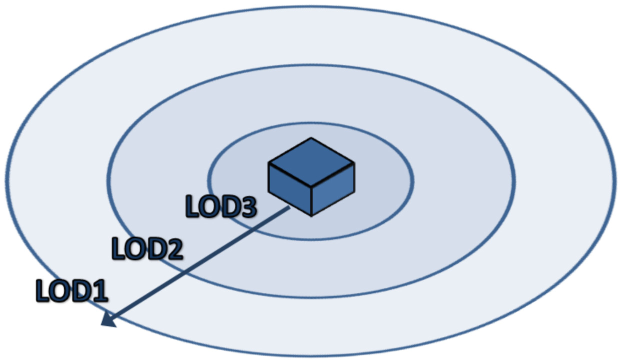
```
<BoundedNode bboxCenter='0 0 0' bboxSize='10 10 10' >

    <Inline url='...' >
    <Inline url='...' >
   ...

< /BoundedNode >
```

2. **Range LOD**
   The LOD node uses range-based approach for switching between different objects

in X3D (see Figure 2.6a). This grouping node is generally meant to manage loading and switching of models. The users can specify the distances in the ”range” which defines multiple distances at which the 3D model will switch. Each distance in the range field corresponds to subsequent definitions of nodes later on. Figure 2.6b describes an example declaration of Range LOD. It contains multiple values in the range field and is used to define the distances at switch the subsequent model definitions will be switched.

Figure 2.6: X3D Range LOD Node

(a) Different range values for different LOD models



(b) LOD X3D Declaration

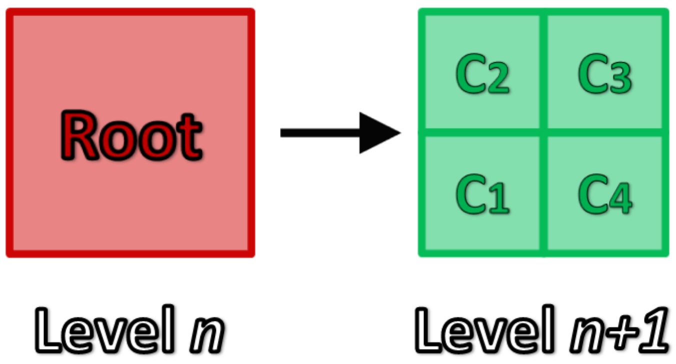
```
<LOD range='10 50 150' center='0 0 0' >
    <Inline url='LOD_3.x3d' >
    <Inline url='LOD_2.x3d' >
    <Inline url='LOD_1.x3d' >
< /LOD >
```

3. **GeoLOD**
   It is implemented as an X3D node in the geospatial component which helps recursively define deeper levels of terrain tiles originating from a parent GeoLOD node. Each tile in this type of reference refers to another set of 4 child tiles as its nodes, (see Figure 2.7a) hence this is commonly referred to as a quad-tree structure. The current node contains nodes for defining the current tile properties such as material, texture and corresponding terrain elevation grid. Other fields include the center of the current tile, range for switching to next level and URL definitions for child nodes (see Figure 2.7b).

These techniques the existing methods in X3D to use LOD for optimizing the rendering engine’s performance, however they are very generic in nature. *ProxyPrismLOD* technique is designed specifically to be used in VCEs and tries to optimize the performance even further. Detailed implementation will be discussed later on in this report.

Figure 2.7: X3D GeoLOD Node

(a) Quad-Tree Hierarchy



(b) GeoLOD X3D Declaration

```
<GeoLOD
      range='10'
      geoSystem="GD','WE"
      center='0 0 0'
      rootUrl='root.x3d'
      child1Url='c1.x3d'
      child2Url='c2.x3d'
      child3Url='c3.x3d'
      child4Url='c4.x3d'
      / >
```

2.3 CityGML Standard

CityGML is a standard widely used for storage and reuse of virtual city models and other objects [9, 10, 11, 33]. It is an official OGC standard and can be used free of charge. It is used to represent urban city objects along with the 3D data it contains. Therefore, not only it serves as a tool to store 3D data but also to store semantics associated with the data, such as roof, doors, windows, and other objects. It is often used to manage GIS data of building models and other urban objects.

In the context of the portrayal of LOD techniques, we have seen that existing techniques do not have any general guidelines on how to adapt them for different environments, which is left at the discretion of the author. For example, the number of levels for LOD switching, distance measurements are left for the user to decide. For VCEs and mirror worlds using CityGML definitions, this behavior is standardized and therefore needs just a few definitions along with some control over the characteristics of LOD technique.

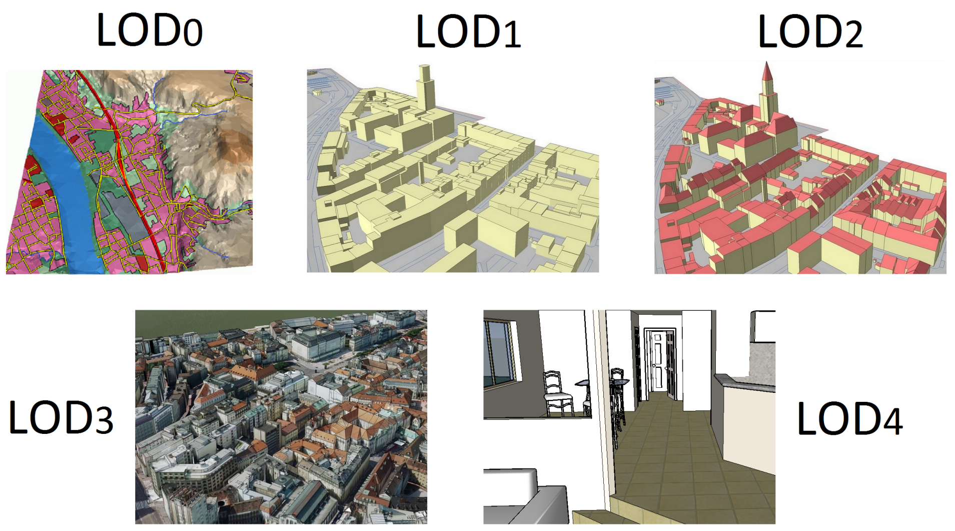
CityGML defines 5 levels for building models, namely LOD0 through LOD4 [34] (see Figure 2.8). With each level, the complexity of the model increases:

- **LOD0**
  The lowest level, LOD0, is nothing but a terrain model for representing a given region, which may be covered with aerial imagery or a map. LOD0 is not directly

associated only with a certain building or an object but comprises of all objects over an area.

- **LOD1**
  LOD1 comprises of the simplest 3D block model of a building without any roof or exterior definition. It may be defined as a simple extrusion covering the building borders.
- **LOD2**
  The next level, LOD2 is more detailed than LOD1 in terms of proper roof definitions as well as differentiated surfaces on the exterior of the building.
- **LOD3**
  This level provides detailed exterior model of the building. If photo textured files are available, they can be mapped to the structure in this level.
- **LOD4**
  This is the highest detailed model, which adds interiors to the building like rooms, doors, stairs and furniture etc. Also, there is a significant jump is performance load between LOD4 and LOD3 when rendering these buildings.

Figure 2.8: Levels of Abstraction for Urban 3D Objects in CityGML.



CityGML has a well-defined structure to define and manage building models of a particular building. This helps designers to create and have many different versions of buildings while trying to make one detailed model. [20] talks about ways to progressively refine building models starting from lowest levels. This will help in the creating multiple LOD for a VCE without putting extra effort. Also, since CityGML characterizes the different levels of a building complexity, the number of levels to be used is no longer ambiguous and standardized. Furthermore it will also become much easier to maintain a constantly updating VE, where the authors can just update the files used, wherever they are stored in the repository. Due to the features and advantages described above, the use of CityGML for managing building complexity will be ideal in implementing a LOD technique specifically for VCEs.

2.4 3D Blacksburg

This is a project undertaken to create a geo-referenced mirror world for the Virginia Tech Campus [16] and the Town of Blacksburg [14]. Aim of the project is to create a high-quality VE of the Blacksburg region. Multi-disciplinary researchers, experts and students from Virginia Tech and local governments are helping in developing the infrastructure for maintaining an interactive 3D virtual city model. This infrastructure will appeal to different user groups:

- **Public**
  Availability of Virtual City Models will help in tourism, commerce and social ventures.

- **Town Management**
  Activities like emergency management, threat assessments and town planning will benefit from such an infrastructure as it will streamline the interaction between different agencies to plan their activity and also in sharing of resources.

- **Academic Community**
  Such an infrastructure will also help in new research for many disciplines.

As the 3D Blacksburg progresses, we anticipate several challenges:

- **Development**
  3D Blacksburg has started by creating a 3D database of relatively low quality buildings (mostly LOD1 and LOD2). But, in future we expect the quality/fidelity of these

models to improve. Hence, we have to take into account the future improvements of building models while designing our LOD technique.

- **Maintenance**
  Also with time, the size and complexity of the building models in 3D Blacksburg database will increase and we would need techniques which will effectively manage these huge databases.

- **Visualization**
  Another challenge is to create tools which can be used by the end user effectively to visualize these databases. This will lead to collaboration between different users as well as different agencies.

The goal of the *ProxyPrismLOD* technique is to be deployed in the 3D Blacksburg infrastructure for better management of large-scale VCEs using CityGML definitions and give a performance optimized experience to the end users.

Chapter 3

Challenges & Research Focus

As discussed in previous chapter, there are different kinds of LOD techniques and they are used depending upon the complexity of environment as well as the application requirements. In this chapter, we will discuss the motivation and the challenges behind developing a new and novel LOD technique for use in VCEs.

The motivation behind our *ProxyPrismLOD* technique is directly reflected by the challenges faced in the existing techniques implemented in X3D to deliver LOD in 3D environments. These techniques use simple range and bounding-box based approaches to improve rendering performance of the VEs. These approaches are simple and easy to use, but have shortcomings in terms of performance and implementation that can be improved.

Apart from the existing techniques implemented in X3D, *ProxyPrismLOD* technique uses CityGML definitions of building objects as more than just mere graphical 3D models. It tries to define every aspect of a building structure so that it can be better managed and developed upon in GIS. It defines four levels in a building from which a different LOD model can be characterized. We will discuss how the CityGML fits in the implementation of *ProxyPrismLOD* technique later on in this chapter.

Also, the *ProxyPrismLOD* technique is intended to be used in the 3D Blacksburg project, which is a geo-referenced Mirror World based on Virginia Tech campus [16] and the town of Blacksburg. The technique will help manage the 3D models of buildings in Blacksburg and along with improvements in rendering performance.

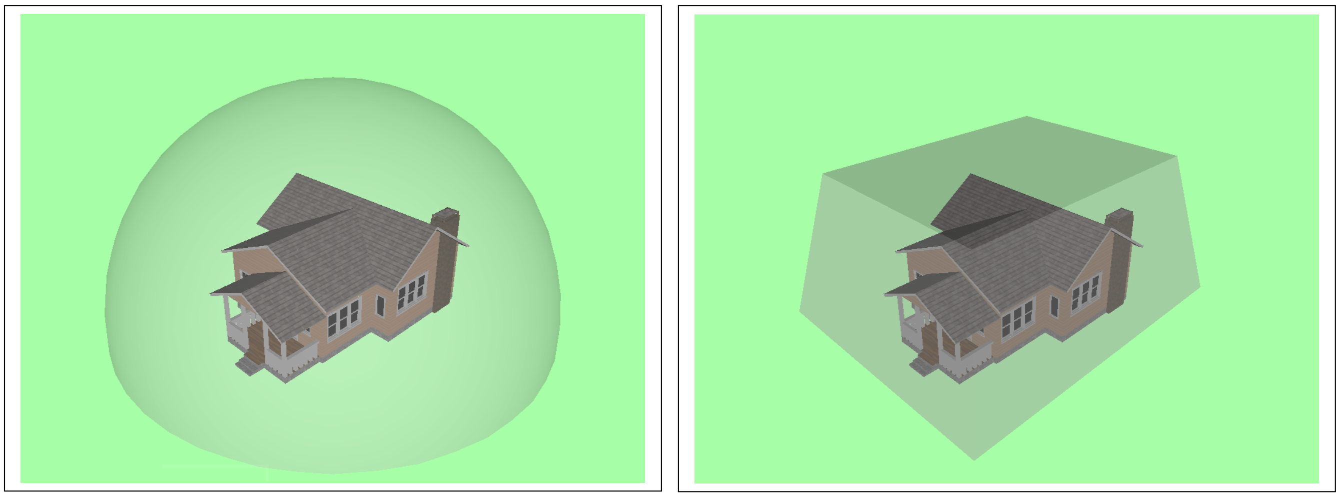
3.1 *ProxyPrism* Shape

There are many cases where existing X3D LOD node is not fully optimized for performance. A range-based metric for switching LOD4 buildings will not cover the boundary of a building optimally (see Figure 3.1a). It will cover the largest area required to encapsulate the whole model. This is not a problem when the switching threshold is far away because the visual overage of the building is almost constant at a larger distance. Whereas, when the user is close to the building even a slightest change in viewing angle can greatly affect the visual coverage.

LOD4 describes buildings with interior details and is visible at a constant distance around the periphery of the building and not from the center of the building (see Figure 3.1b). Hence, optimally a proxy shape which covers the buildings around its edges will be most optimized in terms of performance. Also this shape cannot be a simple cuboid, as buildings can also have asymmetric structure. To optimally envelope such a building multiple faces would be required. This shape will be more flexible is accommodating many all non-symmetrical oddly shaped buildings and block structures.

Figure 3.1: Volume coverage in range-based and *ProxyPrism*-based LOD switching.

(a) Bounding Sphere (b) *ProxyPrism*



3.2 Focus of *ProxyPrismLOD* Technique

The extensibility of X3D provides a powerful means to implement and test new, custom node functionality. There are numerous additional LOD techniques from other standards and the literature that may be of interest to test. For our design of *ProxyPrismLOD* technique, computational and perceptual impact of LOD technique is being tested. This technique is evaluated and analyzed in order to demonstrate the advantages associated with it. The detailed implementation of *ProxyPrismLOD* technique is explained in the next chapter.

- **Techniques to manage large-scale VCEs (3D Blacksburg)**
  We use CityGML definitions to manage the 3D data of the building models. Since CityGML is used for GIS data management, we want to integrate the sematic definitions from CityGML in our technique.
- **New LOD definition to optimally use CityGML model definitions**
  - Radial distance approach is used for observing a model at a larger distance.
  - Proxy shape called *ProxyPrism* used for switching between LOD3 and LOD4.

Next chapter discusses the *ProxyPrismLOD* technique and its features in detail.

Chapter 4

ProxyPrismLOD Technique: Design & Implementation

This chapter will discuss the features and implementation of *ProxyPrismLOD* switching technique. Our technique is directly motivated from the CityGML standard as well as a need for an easy way to deploy LOD switching in X3D based VCEs.

As discussed earlier the CityGML standard describes four levels of building models in its features with varying detail and corresponding performance load on the render process. It can range from a few vertices in LOD1 without any textures to around 100k vertices in LOD4 with several megabytes of texture memory. This range in memory requirements makes it more important to design a switching technique which maintains the high quality of such a VE as well as guarantees smooth performance.

While it is important to have a switching technique which could use all 4 LOD models and efficiently switch between them, it is also important to consider the ease of deployment of such a technique. For better management of a large VE, the author should be able to use a simple interface which encapsulates the implementation of the LOD technique. This interface should have the necessary parameters which will be used to define the working of the LOD technique. Also, a platform which supports development, and tools for performance analysis will help effectively analyze the LOD technique.

4.1 X3D Prototype

X3D Prototype node can be used to develop custom, first-class nodes with the encapsulated functionality built in. We have implemented our *ProxyPrismLOD* technique using a X3D Prototype. The inner workings of the Prototype can be defined and implemented in a separate file and just needs to be referenced when instantiating that node. This gives the author a lot of freedom while designing a custom node with existing X3D nodes and functionality. Also, X3D Prototype gives a lot of advantages in terms of functionality and flexibility:

- Implementing this LOD switching technique using X3D Prototype provides a clean interface for defining the building models, switching parameters like distance, rotations and coordinates.
- Support for JavaScript allows us to effectively implement our LOD switching algorithms as well as carry out complex 3-dimensional mathematical calculations during run-time. Use of JavaScript also helps in automated performance data collection, which in turns helps us measure the effectiveness of various methods over others.
- Since the whole functionality of the X3D Prototype is hidden inside a separate namespace, possibly a file, it makes debugging a lot more convenient.

Subsequent sections will discuss in-depth the implementation and features of our *ProxyPrismLOD* technique.

4.2 Features

Before diving down into the implementation of the LOD technique and its explanation, it will be beneficial to understand the motivation behind the features implemented.

These features make our technique prototype novel in many aspects, therefore it is important to discuss them in detail in this section:

4 Levels

This feature is directly motivated from CityGML standard, which defines 4 levels of abstraction for 3D urban objects (see Figure 2.8). LOD1 denotes the block models without any roof or exterior definition. LOD2 contains block models with proper exterior definition of features like extrusions and roof. LOD3 defines a detailed exterior model with possible photo textured building surfaces. Finally, LOD4 which is the most detailed model also contains interior details like rooms, doors, stairs and furniture. Though this abstraction was created for better management and generation of building models in GIS, it is also ideal for use in a LOD technique implementation. CityGML is an open OGC standard and can be used free of charge. Currently, there are many implementations of virtual city databases which use CityGML. Hence, using an already available open standard will be helpful for testing and the resulting analysis will be useful for the community already using CityGML.

Ease-of-Use

As discussed earlier in this chapter, it is quite important for the LOD switching technique be easily deployable for the authors. As the goal of this work is to be used in future implementations of 3D Blacksburg, complexity of deploying this technique will play a major role. X3D Prototype provides a clean interface for developing any custom functionality by providing only the interface with a file reference, where its inner workings are defined. This aspect gives flexibility in comparing different techniques by simply changing the file reference. For example. the authors can test new switching techniques by modifying the existing Prototype files and do not have to worry about generating the whole VE as long as the prototype definition has the same interface.

ProxyPrism

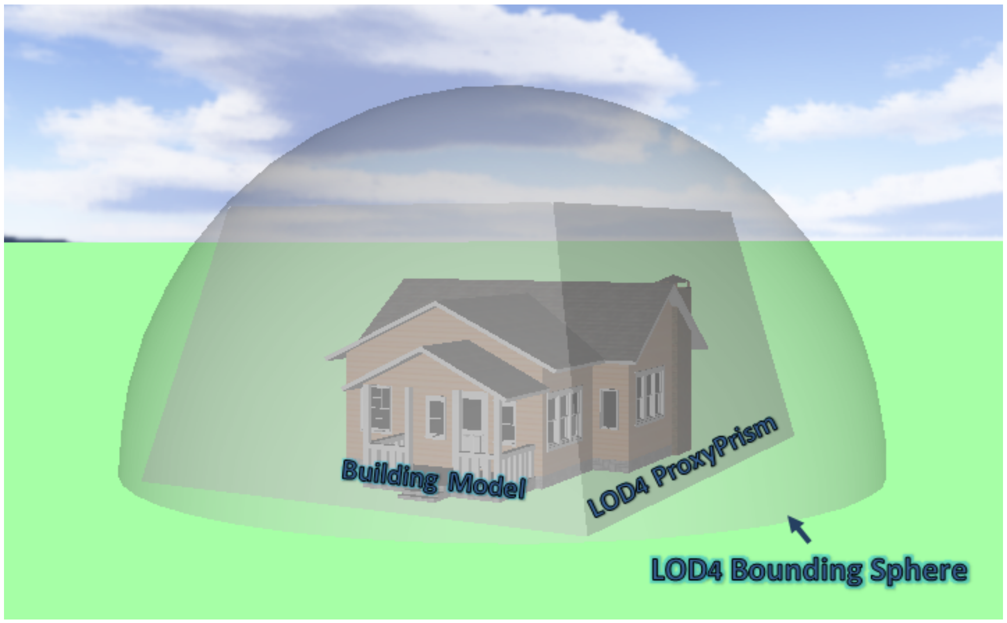
Our *ProxyPrismLOD* technique is essentially based on a range-based function for deciding when to switch. However, as discussed earlier, a purely range-based switching algorithm will have to cover more volume in order to bound the LOD4 model when compared to a box based LOD (see Figure 4.1). LOD4 models are essentially LOD3 models with interiors. Therefore, they are most important when covering the region inside the building, rather than outside. Therefore LOD4 should only be visible covering area just outside the model borders or edges, so that it only switches from LOD3 to LOD 4 when the user is very close to the building. Since LOD4 models contain interior details, there is a big jump in terms of number of vertices and texture memory used when switching from LOD3 to LOD4. Using an optimized shape will be less taxing to the render process when switching from LOD3 to LOD4 and will improve the performance. However, to calculate the position of a point inside a *ProxyPrism*, it will require more computations (perform user-interior-test with respect to *ProxyPrism*) compared to a simple range-based calculation. Therefore, we need to study the tradeoffs in this approach.

Optimizations

As discussed earlier, using a *ProxyPrism* minimizes the LOD4 region, but it also makes the switching calculations much more intensive. To minimize those calculations we need to optimize it as much as we can:

- Instead of calculating whether the point is inside the *ProxyPrism* or not after the user comes inside LOD3 region, another distance value is defined for LOD4 switching. This value is essentially a sphere which bounds the *ProxyPrism* completely as shown in Figure 4.1. Therefore, the calculations for finding the point inside the Proxy Prism can be further reduced to only when inside that region.

Figure 4.1: Comparison between Proxy Prism and Bounding Sphere.



- Another optimization which could be useful is when we start calculating the position of the point inside the *ProxyPrism*. Since it is required that user position be inside all six planes of the *ProxyPrism*, we can break the calculations whenever a false value is encountered, thus eliminating the need to calculate based on remaining planes.

Keeping these features in mind, our *ProxyPrismLOD* technique is implemented in X3D Prototype node. Next section will discuss the implementation of above mentioned features in detail.

4.3 Implementation Details

Now we will discuss the interface and the switching algorithm for *ProxyPrismLOD* technique and illustrate the workings with an example.

4.3.1 Prototype Declaration

Figure 4.3 shows the ProtoDeclare field with all the required fields for adding a building to be used with our *ProxyPrismLOD* technique. It is a very clean interface and contains only essential fields which will be used when adding a building model to this LOD technique. Description of the fields is as follows:

LOD Models

There are 4 separate fields for providing the location of the appropriate model file (see Figure 4.3a). In the Prototype file, these models are referenced as separate *Inline* files. This makes it easier to manage these models, when user can just update the file path inside an existing VE or overwrite with an updated model file.

Position

This field denotes the center of the building model in the VE and is represented by a *SFVec3f* (see Figure 4.3b). If a Transform is used before using the Prototype Instance for a particular model, the value of position field in Transform needs to be entered here. Entering the position is important as the LOD switching calculations occur in the Prototype declaration file, which is unaware of the absolute position of the building in the parent VE.

LOD Cutoffs

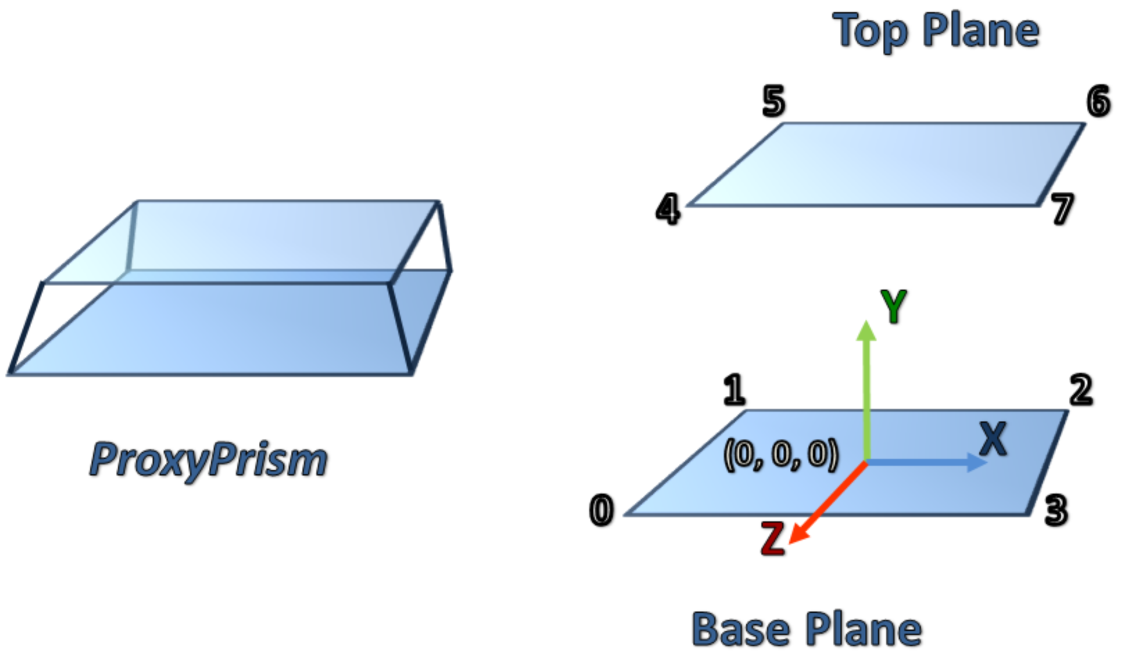
These fields, as the name suggests give the absolute limit or the extent of that particular LOD model (see Figure 4.3c). These are represented by *SFInt32* and are restricted by their order of importance, i.e. LOD1_cutoff>LOD2_cutoff>LOD3_cutoff.

This is similar to the range field in the LOD technique currently used in X3D. Authors can predefine these LOD cutoffs according to their application. However, an optimal cutoff range-based on ″visual continuity″ and ″distraction″ to the viewer needs to be studied. A user study was performed to study this aspect and is discussed later in 6.

ProxyPrism

This field denotes the coordinates of the *ProxyPrism* the user wants to define. It is represented by *MFVec3f* which is an array of *SFVec3f* as shown in Figure 4.3d. The ordering of the coordinates is explained in Figure 4.2. Starting from the lower plane the coordinates are ordered in clockwise direction continued to the upper plane. This 3D shape is the most important part of the algorithm as using this makes the calculations considerably more complex when compared to a simple range-based switching technique. The use of top and base plane in the ordering example in the Figure 4.2b, is used to simplify defining the *ProxyPrism* by making it easier to visualize.

Figure 4.2: *ProxyPrism* Coordinate Ordering.



Rotation

This field is defined to make the task of properly reporting the *ProxyPrism* coordinates easier and is represented by *SFInt32* (see Figure 4.3e), which essentially is the angle in degrees. If the building model is not aligned with any of the axis, users will have to find the exact coordinates of each of the vertices of *ProxyPrism*. However, using this field they can also define the coordinates by aligning the model to either X or Z axis and specify in the rotation field how much the model needs to be rotated along Y axis.

Figure 4.3: *ProxyPrismLOD* X3D Prototype Declaration.

(a) LOD model files (b) model position (c) LOD cutoffs (d) *ProxyPrism* (e) model rotation

```
<ExternProtoDeclare name='LOD_Switch' url='protos/LOD_Switch.x3d'>
    <field accessType='inputOutput' name='LOD1' type='MFNode'/>
    <field accessType='inputOutput' name='LOD2' type='MFNode'/>
    <field accessType='inputOutput' name='LOD3' type='MFNode'/>
    <field accessType='inputOutput' name='LOD4' type='MFNode'/>
    <field accessType='initializeOnly' name='position' type='SFVec3f'/>
    <field accessType='initializeOnly' name='LOD1_cutoff' type='SFInt32'/>
    <field accessType='initializeOnly' name='LOD2_cutoff' type='SFInt32'/>
    <field accessType='initializeOnly' name='LOD3_cutoff' type='SFInt32'/>
    <field accessType='initializeOnly' name='proxy_prism' type='MFVec3f'/>
    <field accessType='initializeOnly' name='rotation' type='SFInt32'/>
</ExternProtoDeclare>
```

4.4 LOD Switching Algorithm

This section will cover the inner workings of the LOD switching algorithm. Aspects ranging from initialization components, per-frame LOD switching algorithm to *ProxyPrism* calculations are discussed in this section.

Initialization Components

Algorithm 1 shows the initialization components of our *ProxyPrismLOD* technique. These components are used throughout the main switching algorithm and initiating them in the beginning reduces unnecessary calculations later on. A modified vector is created with the *ProxyPrism* coordinates for applying the affine transform to get the corrected coordinates from the "Rotation" field in Prototype declaration. At the same time the bounding vertex is also calculated from *ProxyPrism* vector (farthest point from the center of *ProxyPrism*. This bounding vertex which is named "LOD4_cutoff" will help in the optimization discussed earlier and is used in the main switching algorithm.

Algorithm 1 Initialize Components and Variables.

```
Require: LOD1_cutoff $>$ LOD2_cutoff $\&$ LOD2_cutoff $>$ LOD3_cutoff
Require: $sizeof$(proxy_prism) $= 8$
  LOD4_cutoff $= 0$
  center $=$ $centerof$(proxy_prism)
  for $index = 0$ to $sizeof$(proxy_prism) $- 1$ do
      corrected_prism[$index$] $=$ $affine$(proxy_prism[$index$],rotation)
      if $distance$(proxy_prism[$index$],center) >LOD4_cutoff then
          LOD4_cutoff $=$ $distance$(proxy_prism[$index$],center)
      end if
  end for
```

Real-Time Switching Algorithm

Algorithm 2 shows the algorithm used to switch LOD buildings in real-time, when it is provided with a change in user's position. The algorithm does a straightforward switching for LOD1, LOD2 and LOD3 as it uses the LOD cutoffs for triggering a LOD Switch node (triggers anything inside it ON/OFF). This LOD Switch node encapsulates the actual LOD building models defined by *Inline* in the Prototype. For triggering LOD4 models, the algorithm first uses the "LOD4_cutoff" to reduce calculations. If the user's position is found inside "LOD4_cutoff" bounds, then it triggers the function which calculates whether the user's position is inside the *ProxyPrism* or not. Depending upon the value returned by the function, the algorithm switches the LOD4 model.

Algorithm 2 LOD Switching.

```
if $userPositionChanged() = $ true then
    user_distance = $distance($user_position,position$)$
    if user_distance $> 0$ $\&$ user_distance <LOD4_cutoff then
        if $userWithinProxyPrism($user_position$) =$ true then
            $enable($LOD4$)$
        else
            $enable($LOD3$)$
        end if
    else if user_distance >LOD4_cutoff $\&$ user_distance <LOD3_cutoff then
        $enable($LOD3$)$
    else if user_distance >LOD3_cutoff $\&$ user_distance <LOD2_cutoff then
        $enable($LOD2$)$
    else if user_distance >LOD2_cutoff $\&$ user_distance <LOD1_cutoff then
        $enable($LOD1$)$
    else if user_distance >LOD1_cutoff then
        $disable()$
    end if
end if
```

Function userWithinProxyPrism(user_position)

As the function name suggests, this function calculates whether the user's position is inside the *ProxyPrism* or not (see Algorithm 3). It works on the principle of relative position of a point with respect to a plane. It starts by dividing the *ProxyPrism* into six planes and calculating the normal of those planes by choosing 2 vectors from every plane. In our current implementation we have assumed that these 4 points are nearly co-planar. It now calculates the angle the normal makes with the user's position vector with respect to the plane. According to the ordering of the coordinates and the angle calculated, it estimates whether the user's position is inside the *ProxyPrism* with respect to that particular plane. If the result is true, it continues with other remaining planes, otherwise it terminates the function and return false, thus avoiding unnecessary calculations.

Algorithm 3 For $userWithinProxyPrism$(user_position).

```
  prism_planes = $makePlanes$(proxy_prism)
Require: $sizeof$(prism_planes) = 6
  for $plane\_index = 0$ to $sizeof$(prism_planes) $- 1$ do
      normal = $findNormal$(prism_plane[$plane\_index$])
      user_vector = $getVector$(prism_plane[$plane\_index$],user_position)
      if $sameDirection$(normal,user_vector) $\neq$ true then
          return false
      end if
  end for
  return true
```

4.5 Example

This section shows an example use of the LOD Switching Prototype with the field declarations. Figure 4.4 shows the example declaration of a sample building using the X3D Prototype. Figure 4.5 is a screenshot of the X3D file, which is using Prototype in action. The distance spheres and the *ProxyPrism* are shown for reference only and do not appear in the actual VE. Figure 4.6 shows the 4 different LOD models used in this example.

Figure 4.4: Example using sample building models.

(a) LOD model files (b) model position (c) LOD cutoffs (d) Proxy Prism (e) model rotation

```
<Transform translation='0 0 0'>

    <ProtoInstance name='LOD_Switch'>

        <fieldValue name='LOD1'>
            <Inline url='inlines/LOD1.x3d'>
        </fieldValue>
        <fieldValue name='LOD2'>
            <Inline url='inlines/LOD2.x3d'>
        </fieldValue>
        <fieldValue name='LOD3'>
            <Inline url='inlines/LOD3.x3d'>
        </fieldValue>
        <fieldValue name='LOD4'>
            <Inline url='inlines/LOD4.x3d'>
        </fieldValue>
        <fieldValue name='position' value='0 0 0>
        <fieldValue name='LOD1_cutoff' value='825>
        <fieldValue name='LOD2_cutoff' value='500>
        <fieldValue name='LOD3_cutoff' value='320>
        <fieldValue name='proxy_prism' value='12.5 0 -12.5 -12.5 0 12.5 12.5 0 12.5
                                   12.5 0 -12.5 -10 15 -10 -10 15 10 10 15 10 10 15 -10>
        <fieldValue name='rotation' value='90>

    </ProtoInstance>

</Transform>
```

Figure 4.5: LOD Switching Bounds.

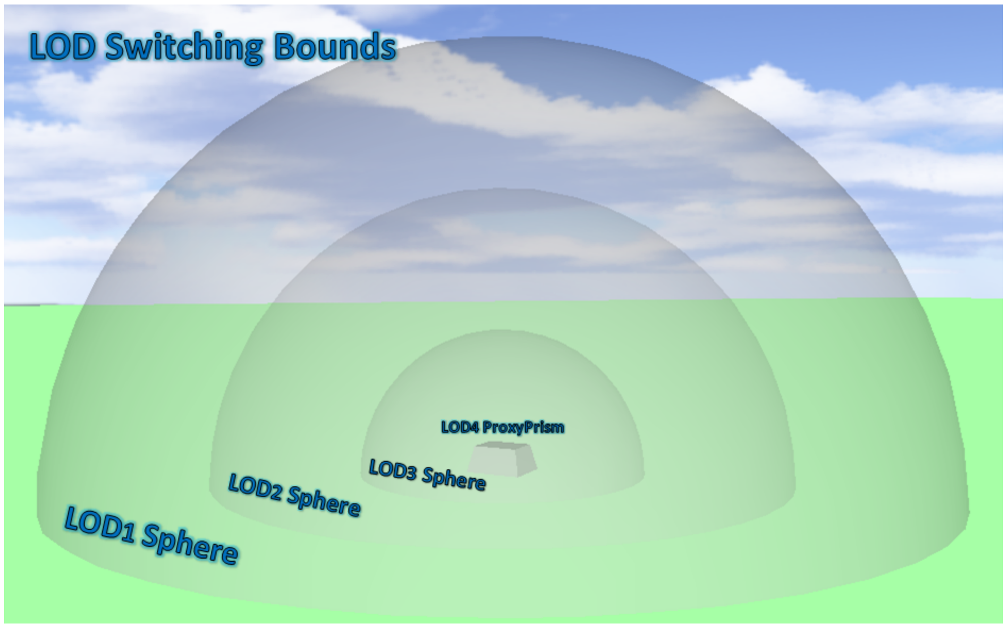
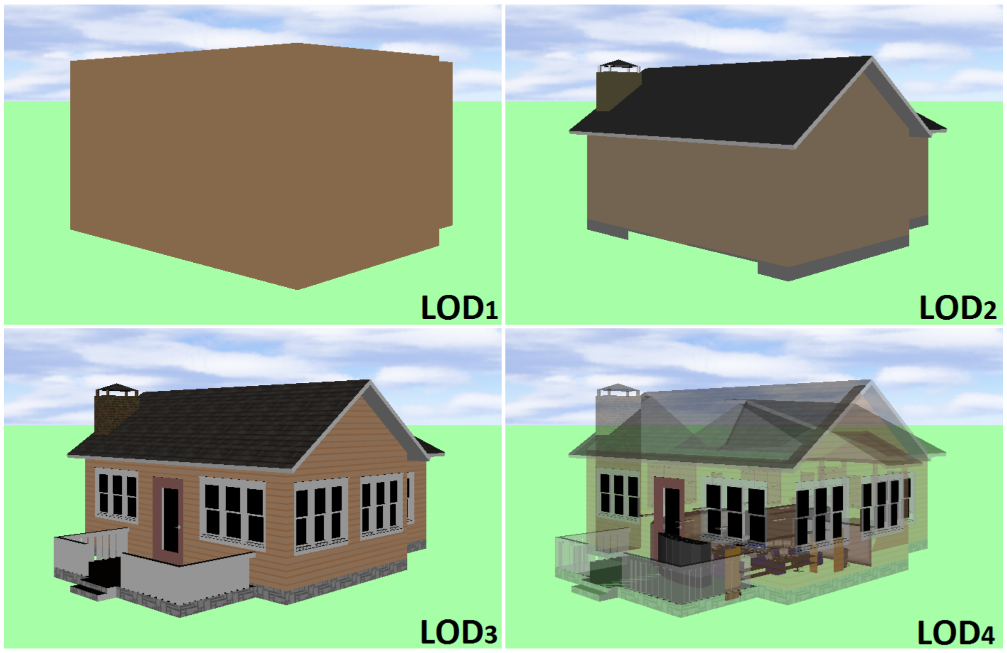


Figure 4.6: LOD Building Models.



Chapter 5

Virtual City Generator

In the previous chapter, the implementation details of the LOD switching technique were discussed. It covered the key aspects and features of the switching technique, an in-depth implementation, and an example on how to use the Prototype in an X3D environment. Since this technique is motivated from VCEs and the problems faced in terms of performance and managing large scale high quality environments, it is fitting to use a VCE in our performance test and analysis of the switching technique.

Using already existing VCEs may give an advantage in terms of future deployments, but the results will not be generic in nature. Therefore, it will be more suitable to generate VCEs using a random script, which generates different environments every time the script is run. This gives many advantages over fixed environments:

- The results obtained with randomly generated environments are more generic in nature, and gives a lot of control on the generation of the environment.
- While running user studies, running many runs on the same VE results in the user getting familiar with the environments, which can affect the results and data collection depending upon what is being studied. Randomly generated environments can remove this problem if the user sees a different environment every time.
- In order to obtain performance results for *ProxyPrismLOD* switching technique, randomly generated environments give a uniform distribution of buildings across the environments and hence the results are more credible. It will be also very easy to generate a large number of environments to obtain more data points.

Apart from generating the VEs randomly, the animated tours were also generated using a random script in order to scale our studies to a large number of data points for performance testing and also user studies.

5.1 Model Set

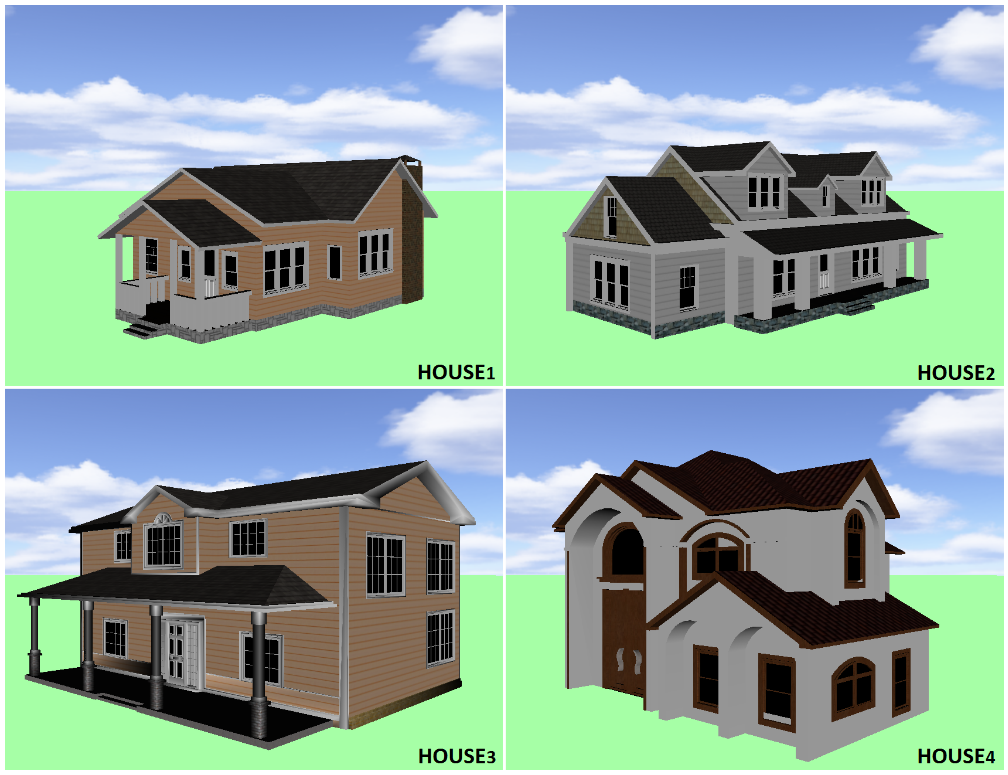
For the VCE, a set of four building models were used with all 4 LOD levels as defined in CityGML. These building models were obtained from Google 3D Warehouse [35] as LOD4 models and then down converted to LOD1, LOD2 and LOD3. In this set, two buildings are smaller houses with single floorplan, whereas the remaining 2 are multistoried houses (see Figure 5.1):

1. **LOD1**
   This model contains the block structure of the house.
2. **LOD2**
   This is the detailed roof structure along with material definition for walls and roofs.
3. **LOD3**
   This model includes detailed exterior structure containing, windows, doors with photo textures for all exterior features.
4. **LOD4**
   This extends the same LOD3 model and adds interior structures like furniture stairs etc.

5.2 Script Generators

This section will describe the essential features of the VE generator and animated path generator. For the purpose of generating a randomized X3D file for testing a PHP script was used. For both the virtual city and the animated path, a random function will return the type of building to be added and the next waypoint in the path respectively. PHP offers a lot of functionality in terms of ease to use syntax, good documentation availability and is helpful in outputing text files (in our case X3D files).

Figure 5.1: Building Models used in the user study.



5.2.1 Virtual City Generator

This generator script uses the predefined X3D models from the above mentioned set to generate a virtual city grid while placing random building models at every grid point. Every aspect of the virtual city can be customized in the script. For example, number of rows, columns in the city, width and height, types of building, maximum number of building of a particular type etc. As the script starts generating the city grid, it randomly selects a building from the building models set (defined in LOD switching Prototype with all 4 LOD levels) and places at every point in the grid. The number of buildings which can be selected can vary. In order to get consistent results, a maximum limit to a particular type of building was added. So the author can control the maximum number of each type of building, but the buildings will be placed randomly in the city. The algorithm is described in detail here in Algorithm 4.

Algorithm 4 Virtual City Generator Script.

```
for $row\_index = 0$ to no_of_rows$-1$ do
    for $column\_index = 0$ to no_of_columns$-1$ do
        pos_x = $column\_index \times$ (road_width+building_width)
        pos_z = $row\_index \times$ (road_width+building_length)
        building_position = $position$(pos_x, 0,pos_z)
        building_type = $getRandom$(0,no_of_buildings$-1$)
        while $limitExceeded$(building_type) = true do
            building_type = $getRandom$(0,no_of_buildings$-1$)
        end while
        $insertBuilding$(building_type,building_position)
    end for
end for
```

5.2.2 Animated Path Generator

Once the virtual city is generated, script generates a corresponding animated path through the virtual city using Position and Orientation Interpolators nodes in X3D. It starts by selecting a fixed number (directly proportional to the total number of buildings with variable multiplier α) of buildings as waypoints, ensuring that there is no direct path between consecutive waypoints. This is done so that in the end we will have a fixed number of waypoints in the animated path. As it starts adding waypoints it updates the Position and Orientation Interpolators for the path and also adds a path into the building whenever it reaches a waypoint (see Figure 5.2b). When using the script, the animated path length will vary every time the script is used. Therefore, to ensure a constant speed of the animation the animation time is kept directly proportional to the path length (with a variable multiplier β). Algorithm 5 shows a detailed description of the animated path generator.

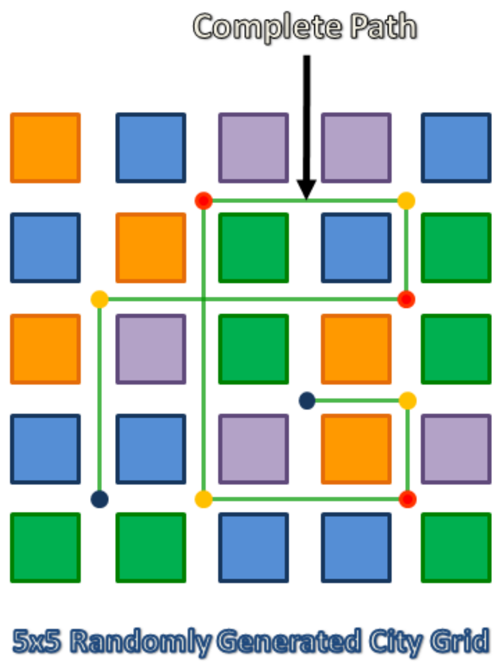
Figure 5.2 shows an example of a virtual city generated by the script along with the animated path.

Algorithm 5 Animated Path Generator Script.

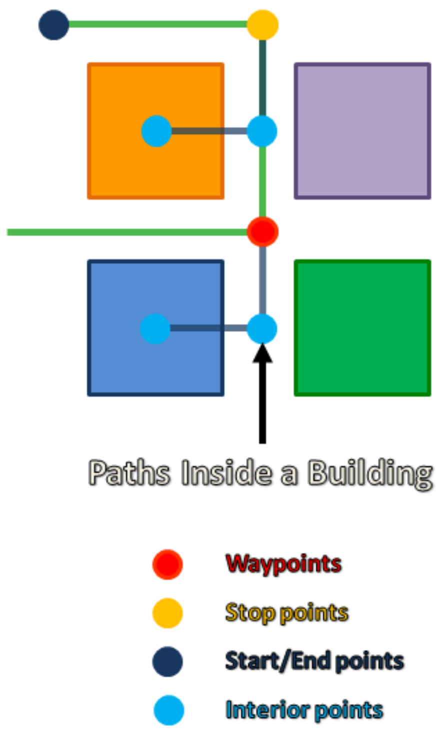
```
no_of_waypoints = $\alpha \times$no_of_rows$\times$no_of_columns
for $waypoint\_index = 0$ to no_of_waypoints$-1$ do
    $row\_index = getRandom(0,$no_of_rows$)$
    $column\_index = getRandom(0,$no_of_columns$)$
    waypoints$[waypoint\_index] = waypoint(row\_index, column\_index)$
    while $directPath($waypoints$, waypoint\_index - 1, waypoint\_index) =$ true do
        $row\_index = getRandom(0,$no_of_rows$)$
        $column\_index = getRandom(0,$no_of_columns$)$
        waypoints$[waypoint\_index] = waypoint(row\_index, column\_index)$
    end while
    $update($position_interpolator, waypoints$[waypoint\_index])$
    $update($orientation_interpolator, waypoints$[waypoint\_index])$
    $update($total_distance$)$
end for
animation_time = $\beta \times$total_distance
```

Figure 5.2: Example environment generated by PHP script.

(a) Virtual City & animated path generated



(b) Path inside a building



Chapter 6

ProxyPrismLOD Technique: Evaluation

In this chapter we will evaluate our *ProxyPrismLOD* technique. In Chapter 4 we have discussed the various features of our technique and how they will help in improving performance and manageability. In this chapter we are trying to empirically evaluate the benefits we have discussed so far. Our evaluation consists of a user study followed by a simulation based performance analysis. We use the data obtained from the user study to control certain parameters in our simulations, which will be discussed later on in this chapter.

The user study is focused on determining an optimal LOD cutoff function which will be used to calculate the different LOD cutoffs. Also, it investigates the effects of sFOV on the user′s perception of ″visual continuity″ and ″distraction″ levels. Using the parameters obtained for the LOD cutoff function obtained from the user study, we ran simulations to test the performance benefits of using ProxyPrism shape over a range-based switching. For both these experiments we used the *ProxyPrismLOD* prototype developed in Chapter 4 for LOD management and switching, and the virtual city generator script discussed in Chapter 5 for generating different environments and animated paths.

6.1 User Study

In this section we will discuss the objectives of our user study, formulation of the LOD cutoff function based on various parameters, the experimental setup followed by the subjective analysis of the feedback.

6.1.1 Objectives

One of the most important factors in designing a discrete LOD technique is to find a balance between the ″visual granularity″ and performance. In choosing the LOD cutoffs, various factors come into place such as the building size, relative spatial coverage etc. Based on the various factors we are trying to find to subjectively evaluate the discrete LOD switching in VCEs:

1. Find the optimal LOD cutoff distances for LOD cutoffs while studying the effect of LOD switching on ″Visual granularity″ and ″distraction″ levels.
2. Study the effect of sFOV on the same ″visual granularity″ and ″distraction″ levels as well as the user′s preference for them.

In order to study the LOD switching, we need to devise a function which can be applied to models of different sizes and spatial coverage and that can calculate the LOD cutoffs for all different levels. We will discuss the switching function in detail in the next section.

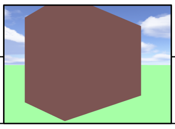
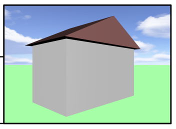
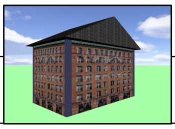
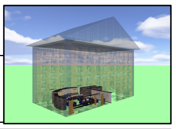
6.1.2 LOD Cutoff Function

Before discussing our LOD cutoff function, we will like to talk about the parameters we think contribute to the scale and separation of various LOD cutoffs at different levels:

Scale of LOD Cutoffs

As described earlier, we are using LOD cutoffs to switch between four differently detailed LOD levels of the same building. Also from CityGML definitions, the complexity and details increases exponentially with each LOD. Table 6.1 shows the characterization of a sample building in all LOD levels. We observe here that number of triangles and texture memory increases with each level. Although, this example does not reflect every possible building model made using CityGML definitions but it does reflect on how the scale of complexity increases with each level. This scale also reflects on how we should design the separation between different LOD models. Figure 6.1 shows a comparison between a linear scale and an exponential scale along with the complexity characteristics from Table 6.1. Due to better mapping of complexity with an exponential parameter, we choose to use exponential scale for LOD separation in our LOD cutoff function.

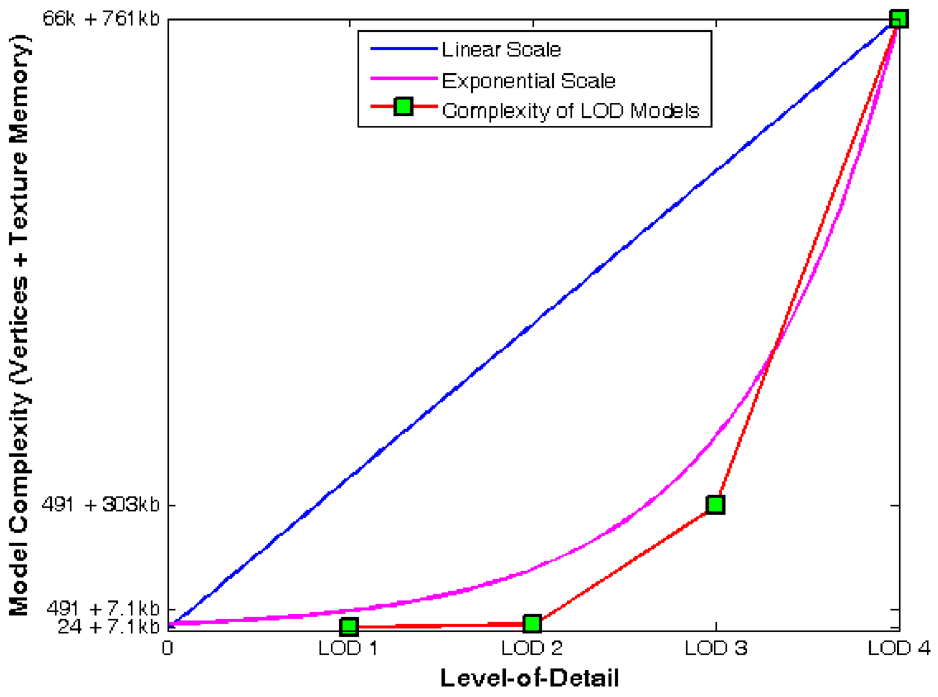
Table 6.1: Characterization of a sample building in all Levels-of-Detail.

LOD$_1$		**LOD$_2$**	
Triangles:	12	Triangles:	377
Vertices Transformed:	24	Vertices Transformed:	491
Textures:	1	Textures:	1
Texture Memory (bytes):	7100	Texture Memory (bytes):	7100
LOD$_3$		**LOD$_4$**	
Triangles:	377	Triangles:	52224
Vertices Transformed:	491	Vertices Transformed:	66594
Textures:	3	Textures:	11
Texture Memory (bytes):	303275	Texture Memory (bytes):	761003

Model Size

Another factor which contributes how the LOD cutoffs should be scaled is the model size. For example, a building model is much larger in size compared to a small house, and hence should have a larger coverage area for higher detailed LOD.

Figure 6.1: Linear and exponential scale with respect to complexity of LOD models.



Larger the model, larger the distance for switching when compared to a smaller model.

Plot Size

While model size is an important factor, it only considers each object individually. It does not into take into account the density of the VCE (no. of distinct 3D models in a given block size). Plot size is defined as the 2D coverage of a particular model, which includes all the area surrounding the building model associated with the model. For example, cities like New York are very dense in terms of buildings, whereas a town like Blacksburg is relatively less dense. Due to the density, building models in Blacksburg can be seen from a larger distance when compared to New York. Hence, a VCE based on Blacksburg should have a larger scale in terms of LOD cutoff compared to New York.

These parameters contribute to our LOD cutoff function, which is then used to calculate the different cutoff distances for a particular building model. The LOD cutoff function is described below:

$$\boldsymbol{Y} = \boldsymbol{\alpha} \times \boldsymbol{\beta} \times \boldsymbol{\gamma}^{\boldsymbol{X}} \qquad \{\forall \boldsymbol{X} \in \mathbb{R} : \mathbf{0} \leq \boldsymbol{X} \leq \mathbf{1}\} \tag{6.1}$$

where,

$$\begin{aligned} \boldsymbol{\alpha} &= \text{Longest Diagonal of Building Model,} \\ \boldsymbol{\beta} &= \text{Scaling Factor,} \\ \boldsymbol{\gamma} &= \text{Plot Radius} \end{aligned}$$

$\boldsymbol{\alpha}$ denotes the length of the longest diagonal of the building model with respect to ground plane and is directly proportional to the model size. $\boldsymbol{\beta}$ is the scaling factor and $\boldsymbol{\gamma}$ is the radius of the plot size. In order to calculate the LOD cutoffs using this function, $\boldsymbol{Y}$ is calculated for $\mathbf{0.33, 0.67}$ & $\mathbf{1}$ for LOD3, LOD2 and LOD1 cutoffs respectively (see Figure 6.2).

6.1.3 Experimental Setup

6.1.3.1 Variables

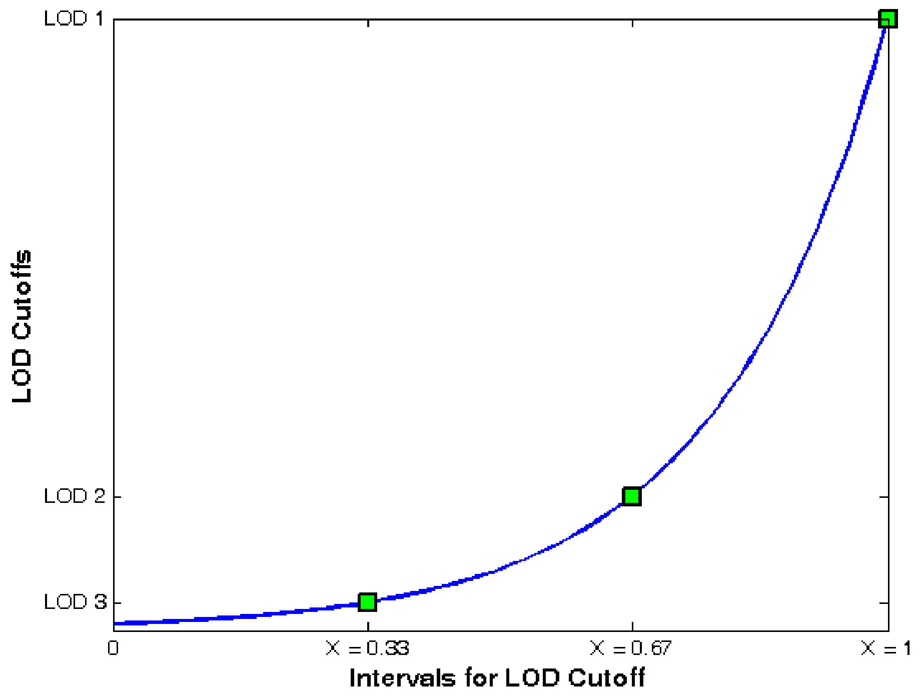
For the user study we varied the independent variables of:

1. Effect of $\boldsymbol{\beta}$, which is the scaling factor, and
2. sFOV **(LOW = 45°, HIGH = 75°)**,

on the dependent variables measured by user rating ($\mathbf{1} \rightarrow \mathbf{7}$) on the "visual granularity" and "distraction" observed.

Table 6.2 shows different variables tested in the user study for each participant and Figure 6.3 shows the range of $\boldsymbol{\beta}$ across **Run 1** $\rightarrow$ **Run 4**.

Figure 6.2: LOD cutoff function.



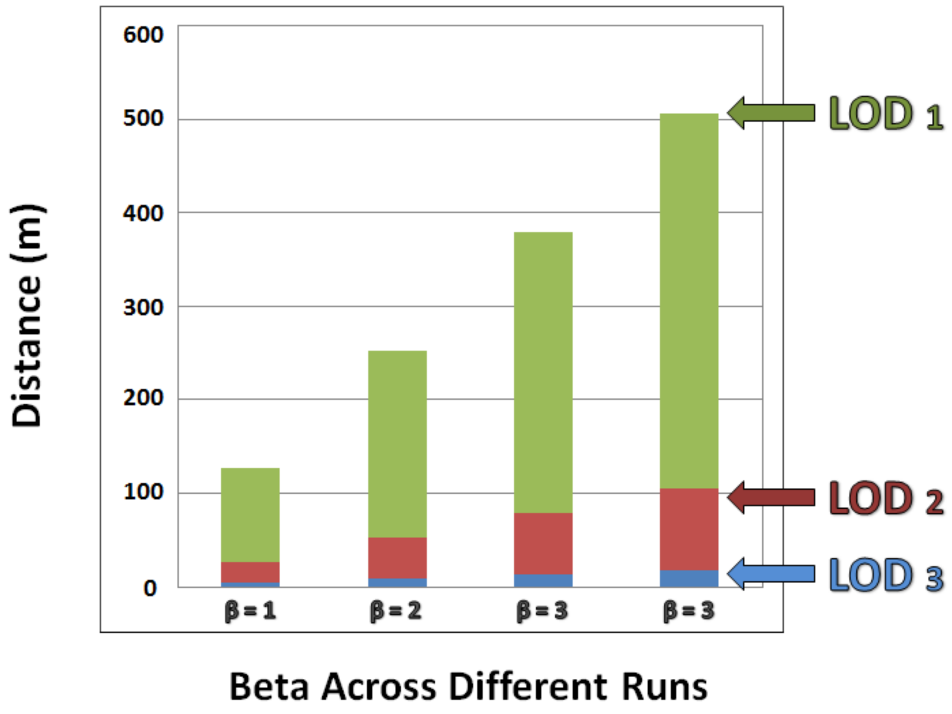
6.1.3.2 Stimuli

For every participant, 8 different VCEs of size $\mathbf{20} \times \mathbf{20}$ were generated along with an animated path using the virtual city generator script and the model set (see Chapter 5), but without the paths going inside the building models (Figure 6.5 shows the overview of the generated VE). The runs are in ascending order of β (ranging from $\mathbf{1} \rightarrow \mathbf{4}$ with increments of 1) for each sFOV condition **(LOW** $= \mathbf{45°}$**, HIGH** $= \mathbf{75°}$**)** (Figure 6.4 shows a screenshot of a sample run in both sFOV conditions). β is the scaling factor which

Table 6.2: Variables tested for each participant.

	$\beta = 1$	$\beta = 2$	$\beta = 3$	$\beta = 4$
LOW sFOV	✓	✓	✓	✓
HIGH sFOV	✓	✓	✓	✓

Figure 6.3: Range of β in meters across Run 1 $\rightarrow$ Run 4.



contributed to the LOD cutoffs.

In order to ensure that there is no bias because of the ordering of the run (ascending or descending order of β and sFOV), in the actual study we randomized the order of the sFOV conditions as well as the ascending or descending order of β. This ensures a uniform distribution across all participants.

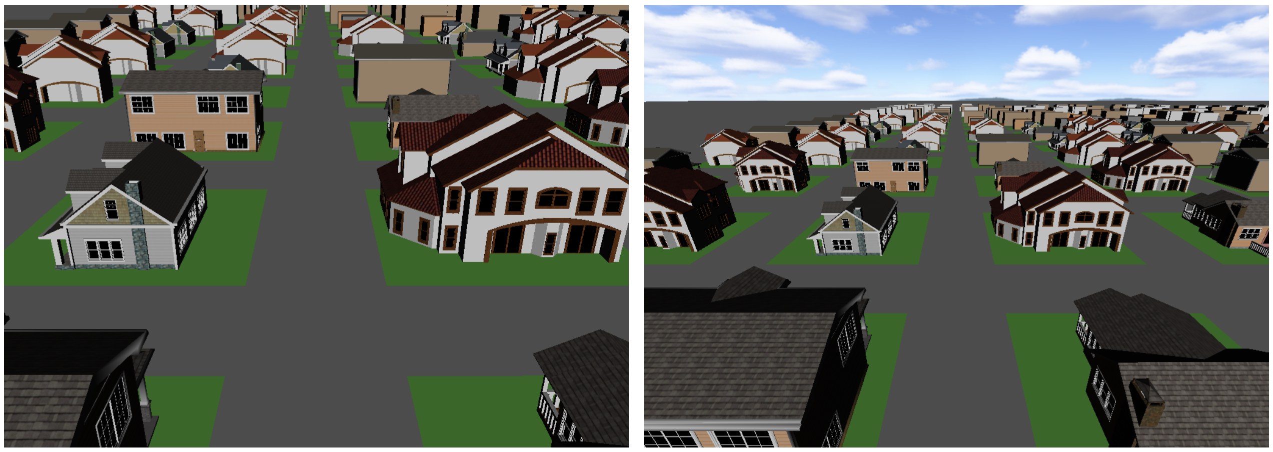
6.1.3.3 Measures

Before the study, each participant was asked to fill out a pre-study demographic questionnaire. The participants were then asked to observe every animated path on a 65" HDTV with a physical FOV of $45°$ and after every run they were asked to rate on a scale of $\mathbf{1} \rightarrow \mathbf{7}$ based on three questions:

1. **How natural and smooth was the rendering of this virtual environment?**
   This question asks how smooth the animated path in terms of FPS (frames per second). This is question is added so that people know the difference between the

Figure 6.4: Comparision between LOW and HIGH sFOV.

(a) LOW sFOV (b) HIGH sFOV



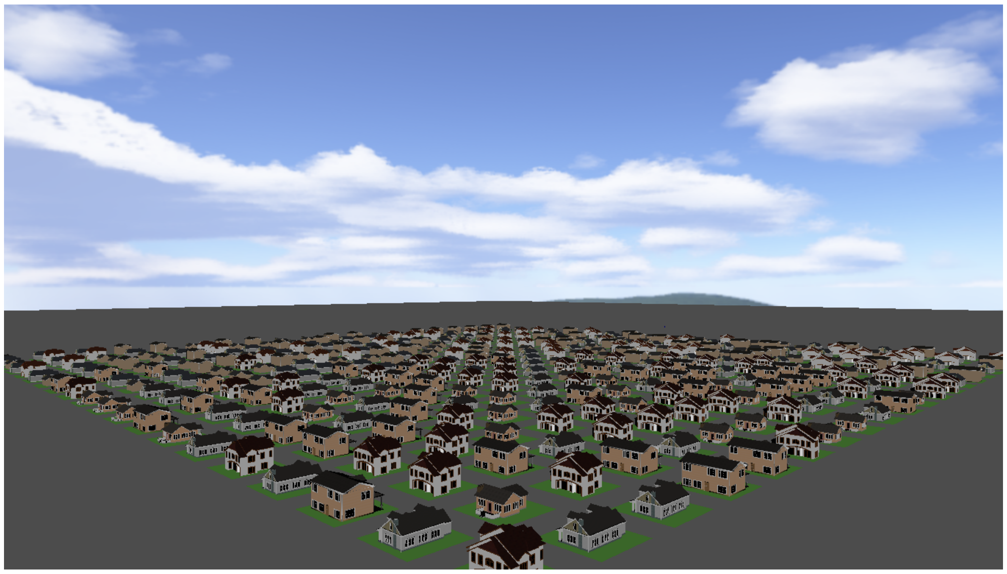
general engine's performance and the LOD technique performance in terms of "visual granularity" and "distraction" levels. We are not going to analyze the results of this question. We include this question to help users distinguish between the "visual granularity" and "distraction" experienced, in contrast with the FPS of the trial.

2. **Please indicate how much popping of building models did you observe?**
   This question directly relates how much switching of LOD models did the user observes. User will observe the changing of building models from a maximum in $\boldsymbol{\beta} = \mathbf{1}$ to minimum in $\boldsymbol{\beta} = 4$.

3. **How distracting was the switching of building models in this run?**
   This question is an extension of the previous question where the user is asked to rate the distraction level of the switching LOD models. Sometimes even though the users observe the switching of LOD models, they might not find it very distracting because the switching is occurring in periphery rather than their focus area.

After the study is completed, each participant fills out a post-study questionnaire, where we ask their personal preference on the sFOV choice and their justification. Refer to Appendix C, D and E for the questionnaires.

Figure 6.5: Overview of a sample VCE used in the User Study.



6.1.4 Analysis

The study was conducted for $\mathbf{12}$ participants in the age group $\mathbf{19-43}$, of which $\mathbf{10}$ were graduate students at Virginia Tech. In their preference of sFOV, $\mathbf{66}\%$ preferred using high sFOV because of the greater viewing angle.

We performed one-way **ANOVA** analysis with $\boldsymbol{\alpha} = \mathbf{0.05}$ on the data collected for questions 2 and 3 with both sFOV conditions. We observed significant difference for all data samples with $\boldsymbol{\alpha} < \mathbf{0.0001}$. In order to determine significant differences between all pairs of $\boldsymbol{\beta}$, we performed Means Comparisons using **Tukey-Kramer HSD** (see Table 6.3). For all pairs except **Run 3** and **Run 4** ($\boldsymbol{\beta} = \mathbf{3}, \mathbf{4}$) of **Switching Observed High, Distraction Level Low** and **Distraction Level High**, we observed significant differences for both conditions of sFOV. Also ratings for question 3 is lowest for **Run 3** and **Run 4**, hence distraction levels are minimal in **Run 3** and **Run 4**.

As the switching distances or the LOD cutoffs are scaled with $\boldsymbol{\beta}$, we observed that the rendering is the smoothest for $\boldsymbol{\beta} = \mathbf{1}$ and most discontinuous for $\boldsymbol{\beta} = \mathbf{4}$ (question 1) for both sFOV conditions. Therefore, from the standpoint of the performance, $\boldsymbol{\beta}$ with the smallest value is preferable. However from the standpoint of the "distraction" from LOD switching a higher $\boldsymbol{\beta}$ is preferable. The results show that there was no significant difference for runs with $\boldsymbol{\beta} = \mathbf{3}, \mathbf{4}$ in terms of "distraction" levels. Therefore, a $\boldsymbol{\beta} = \mathbf{3}$ will be most optimal in terms of both performance and "distraction" levels. Also, apart

Table 6.3: Results of Means Comparisons using Tukey-Kramer HSD.

	Run 1 - Run 2	**Run 2 - Run 3**	**Run 3 - Run 4**
Switching Observed LOW	$\alpha < 0.0001$	$\alpha = 0.0024$	$\alpha = 0.0076$
Switching Observed HIGH	$\alpha < 0.0001$	$\alpha = 0.0102$	$\boldsymbol{\alpha = 0.0708}$
Distraction Level LOW	$\alpha < 0.0001$	$\alpha = 0.0025$	$\boldsymbol{\alpha = 0.1195}$
Distraction Level HIGH	$\alpha < 0.0001$	$\alpha = 0.0267$	$\boldsymbol{\alpha = 0.1042}$

from the preference of sFOV from the participants, we did not observe any significant difference in our scores for question 2 and 3.

For our next experiment, where we run simulations to measure the performance benefits of using *ProxyPrism* shape, we will use $\boldsymbol{\beta = 3}$ for the LOD cutoffs.

6.2 Performance Simulations

In the previous experiment we ran a user study to determine the optimal scaling factor for LOD switching. In this section, we describe an experiment by simulation to determine the performance impacts of using a *ProxyPrism* shape versus a range-based test for switching models at close range.

6.2.1 Objectives

We were able to identify an optimal scaling factor $\boldsymbol{\beta = 3}$, based on ”visual granularity” and ”distraction” levels. Here, we devise an experiment to focus on:

1. Analyzing the performance benefits of using *ProxyPrism* shape over range-based approach.
2. Studying the effect of sFOV on LOD switching on both *ProxyPrismLOD* technique and range-based LOD technique.

We intend to compare our *ProxyPrismLOD* technique to a purely range-based LOD technique. While this is the default function of X3D's LOD node, we copied our Prototype implementation of the *ProxyPrismLOD* and simply removed the shape-based calculations and tests from the internal scipt node.

6.2.2 LOD Techniques

In an effort to compare the *ProxyPrism* shape benefits, we develop two LOD switching techniques which differ only where we are measuring the performance benefits, i.e. the switching between LOD3 and LOD4 building models:

***ProxyPrismLOD* Technique**
: As described in Chapter 4, this technique uses 4 LOD models defined by CityGML standard and uses range-based LOD switching for LOD1, LOD2 and LOD3. For switching between LOD3 and LOD4 we use a custom shape called *ProxyPrism*, which is essentially a shape comprised of **8** vertices and accommodates many oddly shaped and asymmetric building models.

Range-Based Technique
: This technique is developed to effectively compare the performance benefits of the *ProxyPrism* shape. This technique is implemented exactly the same way the *ProxyPrismLOD* technique, except the switching between LOD3 and LOD4 is also based on range-based metric.

6.2.3 Experimental Setup

We performed simulations using the virtual city generator script (see Chapter 5, where the animated paths also go inside building models. We tested two different LOD techniques, namely *ProxyPrismLOD* and range-based with two sFOV conditions **(LOW = 45°, HIGH = 75°)**. We generated 10 VCEs of size $\mathbf{20 \times 20}$ using the building model characterized in Table 6.1. Table 6.4 shows the different variables tested in this simulation experiment.

Table 6.4: Variables tested for the simulation experiment.

	ProxyPrismLOD	**Range-Based LOD**
LOW sFOV	✓	✓
HIGH sFOV	✓	✓

6.2.4 Analysis

We perform this experiment on a set of 10 animated paths in the VE. Each path is tested over the four conditions mentioned in Table 6.4 with the same animated path and the same VCE. At the end of each simulation run, the browser outputs an array of all FPS samples collected during the entire run. Since the animated path is randomly generated, the total distance covered varies across the 10 runs. In order for valid comparison, the camera speed is kept to a constant **20 meters/second**. The browser collects $\approx$ **30 samples/second** and on an average each animated path lasts around $\approx$ **70 seconds**. This gives us $\approx$ **2100 samples** for every run.

We aggregated the samples according to the two independent variables: LOD technique and sFOV (see Table 6.5). Therefore we have **20** observations in each group. When subjected to **T-Tests**, we have that there was significant difference in both LOD techniques and sFOV groups. A significant difference between **HIGH** and **LOW** sFOV confirms our intuition that wider sFOV incurs a higher rendering load ($\alpha < 0.0001$). Also, a significant difference between *ProxyPrismLOD* and range-based LOD demonstrates that our technique significantly outperforms the range-based LOD technique ($\alpha < 0.0001$).

After analyzing the data collected on the FPS values, we calculated the performance improvements in each simulation across all conditions. The percentage benefit is calculated using the formula:

$$\%\textbf{benefit} = \frac{\boldsymbol{FPS_{PP} - FPD_{DB}}}{\boldsymbol{FPS_{DB}}} \times \mathbf{100} \tag{6.2}$$

where,

$$\begin{aligned} \boldsymbol{FPS_{DB}} &= \text{Average FPS for Range-Based LOD Technique,} \\ \boldsymbol{FPS_{PP}} &= \text{Average FPS for } \textit{ProxyPrismLOD} \text{ Technique} \end{aligned}$$

Table 6.5: FPS analysis for all 10 simulation environments.

	ProxyPrismLOD		**Range-Based LOD**	
	Low sFOV	**High sFOV**	**Low sFOV**	**High sFOV**
Run 1	32.731	31.363	32.726	29.317
Run 2	32.187	30.143	29.973	29.260
Run 3	32.265	29.657	32.160	27.968
Run 4	32.312	29.910	31.540	29.743
Run 5	32.068	30.191	31.745	29.101
Run 6	31.802	30.014	30.832	27.928
Run 7	31.419	31.119	30.741	30.037
Run 8	31.823	28.938	30.226	28.815
Run 9	31.758	30.331	31.432	28.626
Run 10	30.844	30.044	30.611	29.494
Average	**31.921**	**30.171**	**31.199**	**29.029**
Std. Dev.	**0.525**	**0.686**	**0.873**	**0.701**

Table 6.6 shows the average, maximum, minimum and standard deviation of the performance benefits under each condition listed in Table 6.4. *ProxyPrismLOD* technique always outperforms range-based LOD technique. The performance benefits range from $\mathbf{0.01}\% \rightarrow \mathbf{7.38}\%$ in low sFOV condition and $\mathbf{0.42}\% \rightarrow \mathbf{7.46}\%$ in high sFOV condition.

Table 6.6: Performance Benefits of using *ProxyPrismLOD* technique.

	Average	**Maximum**	**Minimum**	**Std. Dev.**
LOW sFOV	2.36%	7.38%	0.01%	2.36%
HIGH sFOV	3.96%	7.46%	0.42%	2.56%

Although the performance benefit may seem small in this experiment, there are several factors behind it:

1. The difference in FPS values will only occur when the animated path goes in the region between LOD4 cutoff and *ProxyPrism* shape. Therefore considering the complete animated path, only a small fraction of that distance is actually in the region between the LOD4 cutoff and *ProxyPrism*.

2. The models used in our simulation experiment are relatively simple; larger and more complex models will reflect even better performance benefits.

6.3 Conclusion

In this chapter we discussed two different evaluation experiments we conducted on our *ProxyPrismLOD* technique. The first experiment was in the form of user study to evaluate an optimal scaling factor for the LOD cutoff function. After analyzing the data, we concluded that $\boldsymbol{\beta = 3}$ is the most optimal scaling factor in terms of ”visual granularity” and ”distraction” levels. The second experiment was based on simulations to measure the performance benefits of using *ProxyPrism* Shape. We showed that *ProxyPrismLOD* technique outperforms range-based technique under all conditions.

In the next chapter we will discuss our findings in greater detail and how we can incorporate the results in our *ProxyPrismLOD* technique to make it easier to use with improved functionality.

Chapter 7

Discussion

In this report, we discussed the design and evaluation of our *ProxyPrismLOD* technique. We started by describing the problems faced in existing LOD techniques and the need for a custom LOD technique designed specifically for VCEs. We began from the CityGML data model and used the X3D standard for the portrayal implementation of our LOD technique.

For testing purposes, we also developed a script-based virtual city generator (see Chapter 5), which can generate VCEs of different configurations and also add random animated paths as tour guides. Using these generated VCEs we are able to test:

1. Optimal switching distances, or LOD cutoffs in terms of "visual granularity" and "distraction" levels and their effect on sFOV. We found that $\boldsymbol{\beta = 3}$ in the LOD cutoff function (see Equation 6.1) is the optimal scaling factor for LOD cutoffs.
2. We also evaluated performance benefits of using *ProxyPrism* shape when compared to a range-based sphere, and found performance improvements upto $\mathbf{7.46}\%$. These improvements are measured in terms of average FPS values (see Equation 6.2).

Using the results from our evaluation we propose improvements on the *ProxyPrismLOD* technique. These improvements would make this technique easier to use with minimal setup and offer more features. In the next section we revisit *ProxyPrismLOD* technique discussing our improvements.

7.1 Revisit *ProxyPrismLOD* Technique

From our evaluation we were able to identify features which would make *ProxyPrismLOD prototype* easier to use:

Automatic LOD Cutoff Calculation

Using the LOD cutoff function, it is possible to automatically calculate the LOD switching distances. β is the scaling factor which determines the scale of the building model. Therefore using the cutoff equation and the variable parameter β, the LOD cutoffs can be easily calculated.

Priority Level

We talked about β being the variable parameter, which can determine the scale of the LOD cutoff distances. We can use the ”type of building” as the parameter which determines the scaling factor β. For example, a house will have a lower β than an office building. Using this definition we can also define the landmarks which will be visible at a much larger distance at higher LOD.

Figure 7.1 shows the improved interface of our *ProxyPrismLOD* technique. The field **model_type** (see Figure 7.1c) is used for determining β, and can be set by the author. Using this β the LOD cutoff function automatically calculates the LOD cutoff distances.

Figure 7.1: Improved *ProxyPrismLOD* X3D Prototype Declaration.

(a) LOD model files (b) model position (c) model type (d) *ProxyPrism* (e) model rotation

```
<ExternProtoDeclare name='LOD_Switch' url='protos/LOD_Switch.x3d'>
    <field accessType='inputOutput' name='LOD1' type='MFNode'/>
    <field accessType='inputOutput' name='LOD2' type='MFNode'/>
    <field accessType='inputOutput' name='LOD3' type='MFNode'/>
    <field accessType='inputOutput' name='LOD4' type='MFNode'/>
    <field accessType='initializeOnly' name='position' type='SFVec3f'/>
    <field accessType='initializeOnly' name='model_type' type='SFString'/>
    <field accessType='initializeOnly' name='proxy_prism' type='MFVec3f'/>
    <field accessType='initializeOnly' name='rotation' type='SFInt32'/>
</ExternProtoDeclare>
```

7.2 Future Work

We have developed our *ProxyPrismLOD* technique to be used in 3D Blacksburg project. Its features will help us manage large number of LOD models and improve performance for LOD4 models. Our *ProxyPrismLOD* technique serves as a LOD management utility for the 3D Blacksburg environment and also provides performance benefits over existing range-based LOD techniques.

However, several future improvements can be made on the existing technique:

1. **Larger *ProxyPrism***
   Currently, we are using an 8 coordinate *ProxyPrism* to simplify the process of defining the *ProxyPrism*. However, we can extend the number of coordinates by letting the user define the *ProxyPrism* in terms of planes. So ideally the author can define as many 4 coordinate planes they want for the *ProxyPrism*. We would have to test the computational load of a complex *ProxyPrism* in terms of performance benefits over range-based technique.

2. **Automatic Generation of *ProxyPrism***
   Using the LOD1 model from the CityGML definitions, the LOD technique could estimate a *ProxyPrism* shape which encapsulates the LOD1 model. This will require a pre-processing stage where the *ProxyPrism* would be estimated.

3. **Including *ProxyPrism* in CityGML**
   Another possible way to make the *ProxyPrism* easier to use is by defining the *ProxyPrism* along with LOD1, LOD2, LOD3 and LOD4 models in the 3D Database itself. This way the user would only reference the building model in the LOD technique and all other necessary fields would be queried from the database.

In an experiment run by OGC, called 3D Portrayal Interoperability Experiment (3DPIE), the Web3D service was tested for 3D databases from Paris, Berlin, Mainz and Blacksburg. The 3D data was delivered in X3D and CityGML format for different client applications. As the future milestone in 3D Blacksburg project, techniques to deliver the 3D data along with the *ProxyPrismLOD* technique would be studied. Even though the *ProxyPrism* technique has been developed using the X3D specifications, studying the effect of using the LOD technique on different clients, such as Instantreality [31] and Bitmanagement [36] would help see the effect of the client application on the performance benefits.

Bibliography

[1] D. Luebke, B. Watson, J. D. Cohen, M. Reddy, and A. Varshney, *Level of Detail for 3D Graphics*. New York, NY, USA: Elsevier Science Inc., 2002.

[2] M. Doggett and J. Hirche, "Adaptive view dependent tessellation of displacement maps," in *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, ser. HWWS '00. New York, NY, USA: ACM, 2000, pp. 59–66.

[3] "Google earth," http://earth.google.com/.

[4] "Google maps," http://maps.google.com/.

[5] "Web3d consortium — open standards for real-time 3d communication," http://www.web3d.org/.

[6] "Open geospatial consortium — ogc ®," http://www.opengeospatial.org/.

[7] D. Brutzman, *X3D: Extensible 3D Graphics for Web Authors*. San Diego, CA: Elsevier, 2007.

[8] "X3d for developers," http://www.web3d.org/x3d/.

[9] T. H. Kolbe, "Representing and exchanging 3d city models with citygml," in *3D Geo-Information Sciences*, ser. Lecture Notes in Geoinformation and Cartography, J. Lee and S. Zlatanova, Eds. Springer Berlin Heidelberg, 2009, pp. 15–31.

[10] A. Stadler, C. Nagel, G. Knig, and T. H. Kolbe, "Making interoperability persistent: A 3d geo database based on citygml," in *3D Geo-Information Sciences*, ser. Lecture Notes in Geoinformation and Cartography, J. Lee and S. Zlatanova, Eds. Springer Berlin Heidelberg, 2009, pp. 175–192.

[11] T. H. Kolbe and G. Grger, "Towards unified 3d-citymodels," in *In: Proc. of. ISPRS Commission IV Joint Workshop on Challenges in Geospatial Analysis, Integration and Visualization II, September 8 - 9*, 2003.

[12] P. Birkel, "Sedris data coding standard," in *In Proceedings of the Spring Simulation Interoperability Workshop*, 1999.

[13] D. A. Bowman, E. Kruijff, J. J. LaViola, and I. Poupyrev, "3d user interfaces," 2005.

[14] "3d blacksburg collaborative," http://www.3dblacksburg.org/.

[15] D. Gelernter, *Mirror worlds or the day software puts the universe in a shoebox: how will it happen and what it will mean*. New York, NY, USA: Oxford University Press, Inc., 1991.

[16] D. Tilden, A. Singh, N. F. Polys, and P. Sforza, "Multimedia mashups for mirror worlds," in *Proceedings of the 16th International Conference on 3D Web Technology*, ser. Web3D '11. New York, NY, USA: ACM, 2011, pp. 155–164.

[17] J. H. Clark, "Hierarchical geometric models for visible surface algorithms," *Communication of the ACM*, no. 19, p. 554, 1976.

[18] "Level of detail," http://glasnost.itcarlow.ie/ powerk/GeneralGraphicsNotes/levelofdetail/Levelofdetail.htm.

[19] H. Hoppe, "Progressive meshes," in *Computer Graphics (SIGGRAPH96 Proceedings), 99108*, 1996.

[20] J. Döllner and H. Buchholz, "Continuous level-of-detail modeling of buildings in 3d city models," in *Proceedings of the 13th annual ACM international workshop on Geographic information systems*, ser. GIS '05. New York, NY, USA: ACM, 2005, pp. 173–181.

[21] J. Yan, "Advances in computer-generated imagery for flight simulation," August 1985.

[22] M. Cosman and R. Schumacker, "System strategies to optimize cig image content," in *In Proceedings of the Image II Conference*, June 10-12 1981.

[23] B. Schachter, "Computer image generation for flight simulation," *Computer Graphics and Applications, IEEE*, vol. 1, no. 4, pp. 29 –68, oct. 1981.

[24] J. Wernecke, *The Inventor Mentor: Programming Object-Oriented 3d Graphics with Open Inventor, Release 2*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1993.

[25] D. Luebke and C. Erikson, "View-dependent simplification of arbitrary polygonal environments," in *In SIGGRAPH 1997*, 1997, pp. 199–208.

[26] D. Koller, P. Lindstrom, W. Ribarsky, L. F. Hodges, N. Faust, and G. Turner, "Virtual gis: A real-time 3d geographic information system," in *Proceedings of the 6th conference on Visualization '95*, ser. VIS '95. Washington, DC, USA: IEEE Computer Society, 1995, pp. 94–.

[27] T. A. Funkhouser and C. H. Séquin, "Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments," in *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '93. New York, NY, USA: ACM, 1993, pp. 247–254.

[28] N. W. John, "The impact of web3d technologies on medical education and training," *Computers & Education*, no. 49, 2007.

[29] J. Dllner, T. H. Kolbe, F. Liecke, T. Sgouros, and K. Teichmann, "The virtual 3d city model of berlin-managing, integrating and communicating complex urban information," 2008.

[30] R. Dachselt, M. Hinz, and K. Meissner, "Contigra: an xml-based architecture for component-oriented 3d applications," in *Proceedings of the seventh international conference on 3D Web technology*, ser. Web3D '02. New York, NY, USA: ACM, 2002, pp. 155–163.

[31] D. Fellner, J. Behr, and U. Bockholt, "Instantreality - a framework for industrial augmented and virtual reality applications," 2009.

[32] "Instantreality," http://www.instantreality.org/.

[33] "City geography markup language (citygml) encoding standard," http://www.citygml.org/.

[34] "Citygml lod models," http://www.eurosense.com/.

[35] "Google 3d warehouse," http://sketchup.google.com/3dwarehouse/.

[36] “Bitmanagement - interactive web3d graphics,” http://www.bitmanagement.de/.

Appendix A

IRB Approval

VirginiaTech

Office of Research Compliance
Institutional Review Board
2000 Kraft Drive, Suite 2000 (0497)
Blacksburg, Virginia 24060
540/231-4606 Fax 540/231-0959
e-mail irb@vt.edu
Website: www.irb.vt.edu

MEMORANDUM

DATE: March 7, 2012

TO: Nicholas Polys, Ankit Singh

FROM: Virginia Tech Institutional Review Board (FWA00000572, expires May 31, 2014)

PROTOCOL TITLE: Study of Range-Based LOD Switching Techniques with Respect to Visual Granularity

IRB NUMBER: 12-217

Effective March 7, 2012, the Virginia Tech IRB Administrator, Carmen T. Green, approved the new protocol for the above-mentioned research protocol.

This approval provides permission to begin the human subject activities outlined in the IRB-approved protocol and supporting documents.

Plans to deviate from the approved protocol and/or supporting documents must be submitted to the IRB as an amendment request and approved by the IRB prior to the implementation of any changes, regardless of how minor, except where necessary to eliminate apparent immediate hazards to the subjects. Report promptly to the IRB any injuries or other unanticipated or adverse events involving risks or harms to human research subjects or others.

All investigators (listed above) are required to comply with the researcher requirements outlined at http://www.irb.vt.edu/pages/responsibilities.htm (please review before the commencement of your research).

PROTOCOL INFORMATION:
Approved as: **Exempt, under 45 CFR 46.101(b) category(ies) 2**
Protocol Approval Date: **3/7/2012**
Protocol Expiration Date: **NA**
Continuing Review Due Date*: **NA**
*Date a Continuing Review application is due to the IRB office if human subject activities covered under this protocol, including data analysis, are to continue beyond the Protocol Expiration Date.

FEDERALLY FUNDED RESEARCH REQUIREMENTS:
Per federally regulations, 45 CFR 46.103(f), the IRB is required to compare all federally funded grant proposals / work statements to the IRB protocol(s) which cover the human research activities included in the proposal / work statement before funds are released. Note that this requirement does not apply to Exempt and Interim IRB protocols, or grants for which VT is not the primary awardee.

The table on the following page indicates whether grant proposals are related to this IRB protocol, and which of the listed proposals, if any, have been compared to this IRB protocol, if required.

Date*	OSP Number	Sponsor	Grant Comparison Conducted?

*Date this proposal number was compared, assessed as not requiring comparison, or comparison information was revised.

If this IRB protocol is to cover any other grant proposals, please contact the IRB office (irbadmin@vt.edu) immediately.

cc: File

Appendix B

Informed Consent Form

Informed Consent for Participant of Investigative Project

Virginia Polytechnic Institute and State University

Title of Project:	Study of Range-based LOD switching techniques with respect to Visual Continuity
Principal Investigator:	Dr. Nicholas F. Polys
Co-Investigators:	Ankit Singh

I. THE PURPOSE OF THIS RESEARCH/PROJECT

You are invited to participate in a study of a new Level-of-Detail (LOD) switching technique in a Virtual City Environment. This research studies the effect of "Switching Distance" and "Software Field-of-View (sFOV)" on the visual granularity of an animated path in a virtual city environment. This study involves experimentation for the purpose of evaluating an optimal switching distance which would be used in different applications.

II. PROCEDURES

You will be asked to observe a set of animated paths in a randomly generated virtual city environment which would run on a HDTV display. Before starting, you will be familiarized with the appearance of all different LOD levels of the building models that will appear in the animated path by running a sample path. Please note that you do **not** need to remember their appearance or the path, this is done only to make you comfortable with the environment you will observe. The animated path will take you through a fixed number of buildings (waypoints). A LOD switching technique would be used to switch between different models of the same building (4 levels). You will see this switching happening in real-time on the animated tour. Our goal is to identify an optimal condition for switching in terms of "distraction" and "visual continuity" of the animated tour. We are **not** evaluating the how detailed the generated virtual environments seems, but how natural the switching of models seems to you. All information that you help us attain will remain anonymous. You may be asked questions during and after the evaluation in order to clarify our understanding of your evaluation.

You will also be asked to fill out questionnaires related to your background with such applications, and to also collect your feedback.

The session will last around 30 minutes. You may also terminate your participation at any time, for any reason.

You will be given full instructions and any clarifications you might have before we start our experiment. If anything is unclear, be sure to ask us questions.

III. RISKS

The proposed experiments are straightforward test of observing an animated path on a HDTV display. There is no physical strain and the only foreseeable physical risks are slight eye strain. There are no known mental risks.

If you experience any eye strain or dizziness during a session, then between tasks step away from the HDTV display to take a rest break. The experimenter will explain when you can take such rest breaks. If you decide you cannot continue, you will be allowed to leave with no penalty.

IV. BENEFITS OF THIS PROJECT

Your participation will help test out a new switching technique for virtual city environments. Your feedback would be valuable in establishing the validity and tradeoffs of the switching technique in terms of "naturalness", "visual granularity" and performance. These insights will help incorporate these techniques in Virtual City implementation.

You are requested to refrain from discussing the evaluation with other people who might be in the candidate pool from which other participants might be drawn.

V. EXTENT OF ANONYMITY AND CONFIDENTIALITY

The results of this study will be kept strictly confidential. Your written consent is required for the researchers to release any data identified with you as an individual to anyone other than personnel working on the project. The information you provide will have your name removed and only a subject number will identify you during analyses and any written reports of the research.

VI. COMPENSATION

Your participation is voluntary and unpaid.

VII. FREEDOM TO WITHDRAW

You are free to withdraw from this study <u>at any time</u> for any reason.

VIII. APPROVAL OF RESEARCH

This research has been approved, as required, by the Institutional Review Board for projects involving human subjects at Virginia Polytechnic Institute and State University, and by the Department of Computer Science.

IX. SUBJECT'S RESPONSIBILITIES AND PERMISSION

I voluntarily agree to participate in this study, and I know of no reason I cannot participate. I have read and understand the informed consent and conditions of this project. I have had all my questions answered. I hereby acknowledge the above and give my voluntary consent for participation in this project. If I participate, I may withdraw at any time without penalty. I agree to abide by the rules of this project

________________________________ Signature	____________ Date
________________________________ Name (please print)	________________________ Contact: phone or address or
	________________________ Email address (OPTIONAL)

Should I have any questions about this research or its conduct, I may contact:

Investigator:	Dr. Nicholas F. Polys	Phone (540) 231-0968
	Director of Visual Computing, Virginia Tech Research Computing	
	Email: npolys@vt.edu	
Review Board:	David M. Moore	Phone (540) 231-0968
	Office of Research Compliance,	
	2000 Kraft Drive, Suite 2000, Blacksburg, VA 24060	

cc: the participant, Dr. Nicholas F. Polys

Appendix C

Pre-Study Questionnaire

User Pre-Study Questionnaire

Please help us to categorize our user population by completing the following items.

Gender (circle one): Male Female

Age: _____________

Do you wear glasses or contact lenses (circle one)?

No Glasses Contact Lenses

Occupation (if student, indicate graduate or undergraduate):

____

Major / Area of specialization (if student): _________________________________

Have you used applications which use Virtual Environments before (e.g. Google Earth, Second Life, FPS games etc.)?

a. Never
b. Have used a few times
c. Regularly use them

Please specify: ____

Have you ever used a virtual reality (VR) system other than a game on a console or with mouse and keyboard? If so, please describe it (what type of display was used, what kind of application (e.g. game, architectural walk-through) was running and the context of use (user study, research, entertainment).

____

____

____

Appendix D

Animated Path Questionnaire

Animated Tour Questionnaire

Run: _____ Participant: _____

Rate on a scale of 1-7

How natural and smooth was the rendering of this virtual environment?

Discontinuous	**1**	**2**	**3**	**4**	**5**	**6**	**7**	Smooth

Please indicate how much popping of building models did you observe?

Negligible	**1**	**2**	**3**	**4**	**5**	**6**	**7**	High

How distracting was the switching of building models in this run?

No Distraction	**1**	**2**	**3**	**4**	**5**	**6**	**7**	Quite Distracting

If you experienced any glitches, problems or challenges during this run, please describe:

____

____

____

Appendix E

Post-Study Questionnaire

User Post-Study Questionnaire

Which Field of View would you prefer when using a Virtual City Environment?

___ Low software FOV
___ High Software FOV

Please explain why?

Please use the following space to leave any additional comments, ideas or problems you might had.