# A NEW APPROACH FOR BETTER LOAD BALANCING OF VISIBILITY DETECTION AND TARGET ACQUISITION CALCULATIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ANIL YİĞİT FİLİZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JULY 2010

Approval of the thesis:

**A NEW APPROACH FOR BETTER LOAD BALANCING OF VISIBILITY DETECTION AND TARGET ACQUISITION CALCULATIONS**

submitted by **ANIL YİĞİT FİLİZ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering** _____

Asst. Prof. Dr. Tolga Can
Supervisor, **Computer Engineering Department, METU** _____

**Examining Committee Members:**

Prof. Dr. İsmail Hakkı Toroslu
Computer Engineering Department, METU _____

Asst. Prof. Dr. Tolga Can
Computer Engineering Department, METU _____

Assoc. Prof. Veysi İşler
Computer Engineering Department, METU _____

Asst. Prof. Sinan Kalkan
Computer Engineering Department, METU _____

Dr. Çağatay Ündeğer
SimBT Inc. _____

**Date:** _____

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Name, Last Name: ANIL YİĞİT FİLİZ

Signature :

# ABSTRACT

A NEW APPROACH FOR BETTER LOAD BALANCING OF VISIBILITY DETECTION AND TARGET ACQUISITION CALCULATIONS

Filiz, Anıl Yiğit

M.S., Department of Computer Engineering

Supervisor : Asst. Prof. Dr. Tolga Can

July 2010, 40 pages

Calculating visual perception of entities in simulations requires complex intersection tests between the line of sight and the virtual world. In this study, we focus on outdoor environments which consist of a terrain and various objects located on terrain. Using hardware capabilities of graphics cards, such as occlusion queries, provides a fast method for implementing these tests. In this thesis, we introduce an approach for better load balancing of visibility detection and target acquisition calculations by the use of occlusion queries. Our results show that, the proposed approach is 1.5 to 2 times more efficient than the existing algorithms on the average.

Keywords: Occlusion Query, Parallel, Line of Sight, Visual Perception

# ÖZ

GÖRÜLEBİLİRLİK VE HEDEF TESPİTİ HESAPLAMALARININ YÜK DENGELEMESİNİN DAHA İYİ YAPILABİLMESİ İÇİN YENİ BİR YAKLAŞIM

Filiz, Anıl Yiğit
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Yard. Doç. Dr. Tolga Can

Temmuz 2010, 40 sayfa

Simülasyonlarda varlıkların görsel algılarının hesaplanması sırasında varlığın görüş hattı ile sanal dünya arasında karmaşık kesişim testleri yapılmaktadır. Bu çalışmada arazi ve arazi üzerinde bulunan çeşitli nesnelerden oluşan dış ortamlar ele alınmıştır. Ekran kartlarının örtme sorgusu gibi donanım destekli yetenekleri sayesinde bu testler hızlı bir şekilde gerçekleştirilebilmektedir. Bu tezde örtme sorguları kullanılarak görülebilirlik ve hedef tespiti hesaplamalarının yük dengelemesinin daha iyi yapılabilmesi için bir yaklaşım sunulmuştur. Sonuçlarımız önerilen yaklaşımın mevcut algoritmalara kıyasla ortalamada 1.5 ile 2 kat arasında daha hızlı olduğunu göstermektedir.

Anahtar Kelimeler: Örtme Sorgusu, Paralel, Görüş Hattı, Görsel Algı

# ACKNOWLEDGMENTS

I am deeply grateful to my supervisor, Asst. Prof. Dr. Tolga Can, for his encouragement, inspiration and guidance. Because of his support and outstanding personality, I will always remember the preparation of this thesis as a fun and rewarding experience.

I would like to thank all of those who supported me in any respect during the completion of the project.

Most importantly, I would like to thank my family, Aynur Filiz, Mümin Filiz, H. Günseli Filiz-Demirkol, and Başak Duduhacıoğlu. Without their endless love and support, this thesis would not be possible.

# TABLE OF CONTENTS

# LIST OF TABLES

TABLES

# LIST OF FIGURES

FIGURES

# CHAPTER 1

# Introduction

With the rapid growth of computer systems, modeling of real world problems by the use of computers has become possible which opened a world of opportunities. One such opportunity is the use of computer simulations that try to simulate an abstract model of a system. This abstract model can vary from simple mathematical formulas to real-time full scale combat simulations. The more detail the model possesses, the more computing power is required to achieve acceptable results in the simulations. With the growing trend to use computer simulations, numerous highly detailed models were proposed which raised the performance requirement bar to a level hardly reached even with today's high-tech computing hardware. This progression led to a search of optimization opportunities in the proposed methods to ease the hardware's burden on the calculations made by the simulation. This thesis aims to optimize the performance of visual perception models which are generally found on high-detailed combat simulations.

## 1.1 Problem Definition

In military combat simulations, the soldiers, vehicles and other elements are generally called simulation entities and the aim is to simulate these entities and other environmental objects in the system as realistic as possible to develop military tactics or to provide hands-on combat training for the users of the simulation. The non-user controlled entities are simulated by the system by the use of models inspired by real-life situations. Visual perception models used in simulations generally aim to calculate the information that the entity gathers from the surrounding environment by the use of its visual perception systems.

One of the most important information gathered from the visual systems of an entity is the visibility information of the other entities in the simulation. In order to calculate this information a check must made to see if there exists an unblocked LOS between the entities. The LOS check is the most resource consuming operation of visual perception calculations. The LOS between the entities can be blocked either by the terrain or by other objects, so if no optimizations are performed, all of these entities in the simulation must be checked when visibility is being calculated. When the terrain is huge and the entities have complex representations, this test becomes a bottleneck in visual perception calculations. A variety of methods are proposed to optimize these calculations. Some of these methods propose making use of the hardware visibility queries, which are GPU implemented routines featured in recent graphics cards. Usage of this method increases the performance of the algorithm by making some calculations on the GPU and introducing a new bottleneck, scalability. Since the visibility queries are hardware implemented routines, two different visibility queries can not be run in parallel. When the entity count is large, a lot of visibility queries must be made. If the GPU can not handle the requests in time, the calculation of these queries begins to take a lot of time and block the execution of other steps of the algorithm. Also, when the terrain data is huge, the LOS intersection calculations with the terrain require time consuming I/O operations as the data is being cached into main memory which slows down the whole visual perception calculation.

Another important information gathered from the visual system is the acquisition level of the target which specifies the gathered target details by perceiving the target visually. The visual perception models combine the visibility information with the target acquisition information to specify the identification level of the viewed target. The target acquisition process is a computation heavy process and existing visibility optimizations do not introduce any solutions to the handling of the acquisition calculations. As a result, the target acquisition calculations are generally made after the visibility calculations which also introduces a utilization problem since no additional computations are made during this stage.

To sum up, existing methods to optimize visual perception calculations fail to utilize the system resources properly when the data to be processed is huge and when a complete visual perception model is necessary. In this thesis, we attack this problem and propose a novel method that can utilize system resources more efficiently.

## 1.2 Related Work

The first visual perception models used in games and simulations consisted of simple line-of-sight queries. One of the most classic and simple line-of-sight algorithms is the Bresenham algorithm which dates back to 1965 [4] and works on regular grid-post terrain representations. Originally an algorithm to draw lines, it has been in use in many areas where iterating over a grid along a line is necessary. According to Duvenhage [6], this algorithm was later extended by the works of Bangay [2] and Boyer [3] which proposed more efficient ways to iterate on the terrain. Works of Duvenhage add further information to the terrain data by storing the terrain as a kd-tree and adding min/max values to the tree nodes which are used to further improve the performance of the LOS algorithm.

In 1997, Hewlett-Packard offered the occlusion test extension [9] to be added in the OpenGL 1.2 specification which enabled fast querying of drawn primitives that pass the depth test. Salomon [16] and Verdesca [18] proposed to use the occlusion queries as a preliminary step to avoid unnecessary calculations which enabled the workload of the LOS algorithm to be shared among the GPU and CPU.

The TIN terrain representation was proposed by Peucker [15] in 1980 which led LOS algorithms to be separated as regular terrain data based algorithms and irregular terrain data based algorithms. The irregular terrain data based algorithms became popular for performing better when optimized meshes were used for the terrain. TIN algorithms and usage examples are mentioned in more detail in Chapter 2.

Some of the mentioned methods include the effects of other objects to the LOS tests. Since some objects must be represented separately from the terrain, the intersection of these objects with the LOS ray is generally handled separately. The OneSAF LOS explained in the Verdesca [18] paper checks intersection with the objects in the scene at each iteration of the LOS ray on the terrain. The proposed approach in this thesis completely separates the object LOS intersection tests and the terrain LOS tests which is described in Chapter 3.

An important military topic, target acquisition models are generally used as the last step in the visual perception calculations to find whether the entity is able to identify a target. Koopman's work [12] dating back to 1956 provides a basis for current research in the field. The DYNTACS curve fit model developed later in 1960 uses experimental data to calculate the

acquisition of objects in different conditions. The NVEOL Target Acquisition Model [17] developed by the U.S. Army's Night Vision and Electro-Optical Laboratories tries to model the sensor's detection capabilities directly. This model is presented in more detail in Chapter 2.

## 1.3 The Proposed Approach and Contributions

In this thesis, an optimized visual perception algorithm is introduced to divide the calculation steps among available resources and utilize most of the computation power. Previous methods propose optimizations to terrain line-of-sight queries but fail to achieve scalable architectures when GPUs are involved. Using occlusion queries as a way to avoid unnecessary line-of-sight calculations (as the previous work by Salomon [16] and Verdesca [18] suggests) fails to use the GPU for a constant and scalable speed-up. For example, in the event that the GPU culling phase of the Salomon algorithm runs slower than the actual CPU calculation, the GPU provides no speed-up in the overall calculation. And also, when the environment contains a lot of occluding non-terrain objects, the performance of the algorithm drops to that of the object LOS tests since no GPU calculations are made in parallel when object LOS tests are calculated.

Govindaraju [7] proposes to use multiple GPUs to reduce parallelization problems on occlusion query based culling algorithms. This thesis proposes another approach. By completely separating the terrain LOS, object LOS, and the acquisition steps, CPU, GPU, and I/O operations are utilized better. Unlike previous research which use occlusion queries to speed-up the terrain LOS tests, the proposed method uses occlusion queries to fully calculate the object LOS tests. Also, a number of checkpoints are added to each step to make sure the algorithm runs at the speed of the fastest calculated step. With this approach, the algorithm is expected to be running at the speed of the fastest component in the system; thus, reducing scalability problems associated with CPU-GPU hybrid approaches which requires GPU calculations to be calculated faster than CPU calculations to introduce any speed-up. The proposed approach is unique in the sense that it offers a solution that mentions all three steps commonly found in visual perception calculations and it calculates all of the steps in parallel without blocking any system resources. In the event that any of these steps complete and report that the target is out of LOS, an early out mechanism is introduced to cancel other steps; thus, reducing

unnecessary calculations. Our results show that the proposed approach is 1.5 to 2 times more efficient than the existing algorithms on the average.

At the end of Chapter 3, detailed test results of the proposed approach are given. Test scenarios for real world situations with different complexities are prepared and the performance of popular visual perception algorithms in these scenarios are also given for side-by-side comparison of different approaches.

# CHAPTER 2

# Background

This chapter contains brief overviews of the topics necessary in understanding the proposed visual perception model. The visual perception models are generally used in combat simulations. First, introductory information on combat simulations and common practices in combat simulation modeling will be given. Second, the algorithms that are used in the proposed visual perception model are presented in more detail.

## 2.1 Overview of Combat Simulations

A combat simulation is a type of simulation where the elements commonly found in combat are represented as entities, and the state changes of these entities are simulated through the use of computer systems. Each combat entity has different characteristics; so, a different model comprising the behavior of each entity is used in the simulation. Hartman and Parry [8] classify these models in several categories. The models used in combat simulations are classified as dynamic models as opposed to static models since the state changes are modeled as a function of time. The simulation software generally consists of a simulation loop which updates the states of the entities dynamically as time progresses. Since the state of entities depend on states of other entities, the simulation works as a linear equation system in which one of the variables is time. The output of the simulation, which is the state of this system of equations for a given time, is either displayed to the user or saved to some storage for later use, depending on the usage area of the simulation software. The usage areas of combat simulations are discussed below.

### 2.1.1 Usage Areas

Combat simulations are used in many areas where usage of software to simulate the combat environment can be more efficient, accurate or economic than actually constructing the desired combat environment. The scope and the contents of different types of combat simulations used in different areas vary greatly in their choices of data representation, level of detail, determinism, modeling of time, etc.

The simulations can be used to analyze the efficiency of a developed military hardware or the capabilities of existing task forces. Analysis based combat simulations have high level of detail since computation time is not the primary concern. Providing such detail requires detailed simulation models focusing on realistic representation of the combat elements. These models usually contain probabilistic values and additional effort is put into modeling of the precision of these values. The distribution of these values must be carefully selected to provide usable analysis data.

Combat simulations can also be used to train commanders for practice of developed tactics or handling the stress of real world battle scenarios. Such human operated simulations when coupled with specialized hardware are called simulators. Simulators generally contain less detailed models to be able to provide a real-time simulation environment to the human operator. Achieving real-time performance is one of the most important goals of the simulator designs.

As Hartman and Parry suggests [8] there are many other usage areas of combat simulations such as hardware acquisition or force structuring which impose different requirements on different aspects of the simulation system. The design of the models used in the simulations are heavily dependent on the requirements imposed by the usage areas which generally introduce some drawbacks to the system. This makes it impossible to design a combat simulation model which can satisfy the requirements of all usage areas. So, specialized models must be used to answer the needs of specialized areas in combat simulation.

### 2.1.2 Combat Simulation Structure

The structure of a combat simulation is heavily dependent on the requirements of the simulation domain but a general model can be given to summarize commonly followed guidelines in simulation modeling. Some steps generally found in combat simulations are given below, which are generally executed in the given order:

1. Perception
2. Decision Making
3. Acting
4. Publishing

Since a combat simulation system can be described as a function which takes current state of the simulation entities as input and calculates and sets the states of these entities in the following steps, the general model is expected to have stages to calculate and set the states. The general model given above contains the ”Perception” and ”Decision Making” stages for calculating the states of the entities for the given time and the ”Acting” stage for setting the states of the entities as required by the model of the entities. ”Publishing” stage applies to networked simulations which utilize two or more computers to distribute the calculation load among a cluster of computers. The ”Publishing” state can also be used to store or display the current state of the entities.

In the ”Perception” stage, the sensory inputs of the entities are calculated. These inputs can be gathered from the communication devices the entity possesses or the visual or auditory perception devices or organs of the entities. The output of this stage provides information for the ”Decision Making” stage. The ”Decision Making” stage takes these inputs and the world state and uses these information to generate the actions of the entities. These actions can be calculated for each individual entity or a team of entities depending on the detail or command level required. If some or all of the entities are human controlled entities the ”Decision Making” stage is replaced by the inputs taken from the graphical user interface of the simulation for those entities. The ”Acting” stage, which is the next stage in the computations, takes the decisions made in the previous stage and makes the required actions. In this stage

all of the effected states of the entities are set and the current time of the simulation can be incremented to the next time step. The ”Publishing” stage distributes these changed states to the other computers in the simulation network. In this stage, High Level Architecture (HLA), which is a specialized framework specifically designed for distributing simulation state, can be used. Use of such software also enables different simulations to communicate and work together forming a larger simulation [13].

All of the mentioned stages depend on the previous stage to finish, to be able to start its calculations. This limits the usage of parallelism in the simulation flow. Entity level parallelism can be achieved by computing these stages in a pipelined fashion if the calculation of the states of one entity does not depend on other entities. In an interactive combat environment, this expectation does not hold; so, some synchronization points for iterative calculations are necessary. One example to such a synchronization point could be the collision detection mechanisms that can be used to model realistic physics of the entities. In collision detection algorithms, all the states of the entities must be final to be able to determine the impacts of collisions between the entities.

The proposed method in this thesis is designed in a way to effectively utilize entity level parallelism in the simulation flow. The visual perception of one entity is aimed to be calculated as soon as possible and the calculations of the entities are done sequentially. As opposed to calculating the perception of all the entities at once, which does not allow the entities to be pipelined in the simulation flow, our approach allows the entity to move on to the next stage in the pipeline.

### 2.1.3 Representation of Simulation Data

The simulation data for a combat simulation consists of various types of world state information. Some examples to this type of information commonly found in combat simulations are ground elevation data, raster images of terrain, entity model data, entity visual representations, look-up tables, etc. A brief description of the representation of such information is necessary in understanding the covered visual perception concepts.

In combat simulations, one of the most important types of information is the terrain data. The terrain data consists of different layers of information representing the ground surface

topography. This type of terrain data in the computer simulation domain is called a ”Digital Elevation Model”, a term used by Miller and Laflamme in 1958 [14]. The DEM can be represented either using regular grids or TINs.

A very important data format standard for regular grid elevation data is the DTED format developed by National Geospatial-Intelligence Agency [1]. In this format, the elevation data is kept as a matrix of terrain height values. The resolution of this matrix depends on the level of the DTED map used. DTED introduces pre-determined resolution levels ranging from level 0 to level 2. Level 0 is the lowest resolution possible in which the distance between the data nodes is 30 arc seconds. In level 2, this distance becomes 1 arc seconds which provides a very high resolution elevation map that can be used in high detailed simulations.

Another representation of terrain data is the TIN format developed by Peucker [15]. In TIN formatted data, the terrain is represented as triangles of varying sizes and shapes. Each triangle side is shared with another triangle and this neighborhood information is kept in addition to the points of the triangle which makes it easy and fast to iterate over the triangular height field.

The objects in a simulation may have complex models for visualization. These models can sometimes be used for other calculations such as visual perception. Using these models such purposes increases the representational correctness of the simulation but raises performance issues since the visualization data may contain too much unnecessary detail. The proposed visual perception model uses high-detailed visualization data of the entities while calculating the occlusion of the view. There are too many formats for the representation of visual data and there are no specific requirements imposed by the proposed method so the visual representation data format will not be covered in much detail. The visualization data used in the test scenarios uses triangles to represent the surface of the models. The points of the triangles are stored in an array and the ids of the associated points of the triangles are stored separately to increase vertex reuse and decrease communication overhead when transferring this data to the GPU unit for the occlusion query calculations. Other information specific to visualization such as normals or textures are not stored in the models since visualization of the simulation field is beyond the scope of this work.

## 2.2 Perception Modeling

Perception modeling is the modeling of the process of using sensory information to detect and identify the objects in the simulation environment. In military applications and research, this process is generally referred to as ”target acquisition”. In the context of this thesis, the term ”target acquisition” will be used to refer to the identification of targets.

Perception does not necessarily mean visual identification. The perception from other types of devices such as radars or thermal sensors can be modeled with the proposed method with some modifications. The details of the necessary modifications are discussed in the target acquisition algorithms section. In order to perceive a target visually, it must be unoccluded in the path of the sensor. The algorithms for checking this occlusion are discussed in the visibility algorithms section.

### 2.2.1 Visibility Algorithms

Visibility algorithms cover the occlusion tests between the entity's line of sight and the occluding objects. Line of sight is a term commonly used in wargames or role playing games referring to the area an entity is looking at. Generally, the output of a visibility test is a true or false value meaning if the line of sight to another entity is occluded by occluding objects in the area or not. These occluding objects can be categorized in three categories;

1. Terrain
2. Stationary Entities
3. Moving Entities

The occlusion of the terrain is very important in the simulations of ground entities. Especially when the earth is not represented as a flat earth, the ground limits the visibility field of objects to a great extent. In aerial simulations, the terrain occlusion is important in the identification of ground targets since they can be hidden behind terrain details such as hills and mountains.

The height of the ground is generally measured relative to the sea-level. In order to represent the world in a non-flat shape, world representation models are used. There are various shape

models for the modeling of the earth. The flat earth model assumes that the world is a flat plane and uses the measured height values without change. Spherical earth model assumes that the world is a perfect sphere and maps the measured values to points on the sphere. There are also other complex world representation models such as the ellipsoid model [10] which is beyond the scope of this work since the spherical model provides a fast and reasonable approximation of the earth for the targeted combat simulation applications.

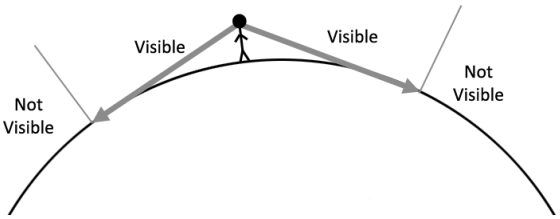



Figure 2.1: Visibility Field in the Spherical Earth Model

If the simulation area is not very small, the flat earth representation becomes very error prone. For example, in a flat terrain, the visibility field of a human entity 1.80 meters in height is infinite. But if spherical earth model is used and the radius of the earth is assumed to be 6371 kilometers, the maximum range of the visibility field becomes approximately 4.8 kilometers. Usage of the spherical model requires additional calculations to be made in line of sight tests since the visibility line moves away from the earth as it moves away from the source when the spherical model is used. 2.1 shows an example visibility field in the spherical earth model.

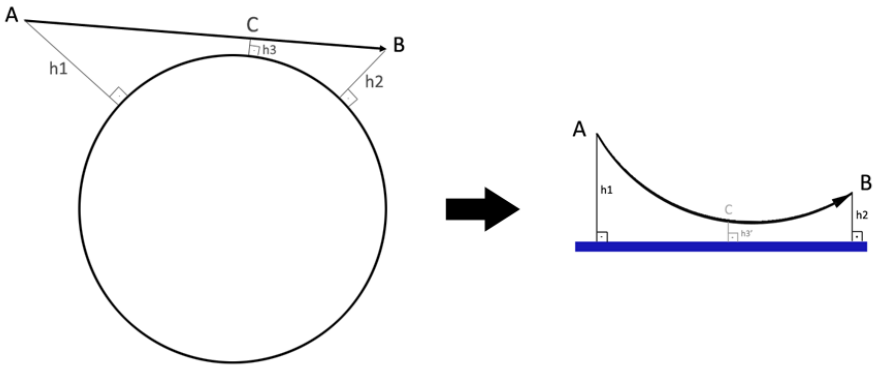



Figure 2.2: Modification of the LOS Ray in the Spherical Earth Model

One approach to use the spherical earth representation in terrain line of sight tests is to modify and raise the height of the LOS ray depending on the distance from the source point to calculate the effect of the curvature of earth at the current point as the LOS ray progresses. This approach is illustrated in 2.2. If geographical coordinates are used, the calculation of the distance becomes complex. The geographic coordinates can be converted to earth-centric coordinates to simplify the distance calculations. When the geographic coordinates are mapped to the earth centric coordinates, the distance can be found simply as the three dimensional distance between the source and the current point.

The algorithm for the progression of the LOS ray on terrain height-field depends on the representation of the terrain height-field. Generally used data types for this purpose are the DTED and TIN formats.

Iteration over DTED formatted data is calculated using grid iteration methods like Bresenham's line drawing algorithm proposed in 1965 [4]. Since DTED formatted data consists of regularly spaced grid nodes, the starting node can be found using the starting coordinate and the extents of the terrain height field. The LOS ray moves along the height field by moving from a grid node to one of its neighbor nodes. There are 4 possible directions the LOS ray can move to but since the LOS ray can not move backward, only two directions need to be checked. Bresenham's algorithm finds which direction the line will move to by using simple comparison operations. At each grid node the height of the LOS ray at the current point is checked against the height of the current grid node to see if the ray hits the ground. The height of the LOS ray at the current point can be found by using the current distance, total distance and total height difference. The height of the grid node at the current point can be found using one of various methods described below.

The non-interpolated method assumes that the height values at a grid node is uniform and is equal to the height value of the south-west pole point. This approach is very fast, but it provides very inaccurate results for large grid nodes. The quad interpolation method uses all four of the height values at the pole points to interpolate the height at any point in the node. 2.3 shows the interpolation of height values. This model is used in the DYNTACS simulation software and it provides a very accurate representation of the terrain field at the price of computation time.

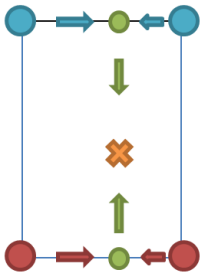


Figure 2.3: Quad Interpolation of the Height Values in DTED Formatted Terrains

When TIN formatted data is used, the ray moves along the triangles that represent the terrain. The starting triangle is found by checking each triangle against the starting point. As the ray moves, at each triangle along the path of the ray, a check is made to find which side of the triangle the LOS ray will move to and the calculations continue with that triangle. When the exit point from a triangle is found the height at that point is found using linear interpolation between the two points of the line that the exit point is on. Since the triangle represents a plane, the height at the entry point and the height at the exit point represents the surface of the plane. The height of the LOS ray at the entry and exit points are found and then these values are checked against the surface heights at those points to find the exact point of intersection if it exists.

### 2.2.2 Acquisition Algorithms

Acquisition is the process of calculating the level of identification attained by sensors. The output of this stage is the acquisition level, which categorizes the amount of information gathered. The contents of this categorization depends on the system that is modeled, such as eyes or binoculars for soldiers or thermal sensors for vehicles, and the level of detail required by the other systems of the simulation. Some general models were proposed outlining several stages of identification as discussed in ”Simulating the Target Acquisition Process” chapter of the book ”High Resolution Combat Modeling” [8].

1. Cuing Information

2. Detection

3. Classification

4. Recognition

5. Identification

The levels provided above represent increasing amount of information gathered from the environment about the target. The categorization of the amount of information is useful in the military applications since there are well defined responses to each level of identification. For example a tank may choose to engage a target if it is identified. But if only cuing information is available, some scout units may be sent to the destination to gather more information.

The early acquisition models focus on using experimental data to find detection probabilities. One such model is the glimpse model where the glimpse count of a target from a sensor is determined. This number is compared against experimental glimpse counts obtained using that sensor to find a detection probability.

An acquisition algorithm which have influenced many other acquisition models is a model developed by U.S. Army Night Vision and Electro-Optical Laboratories (NVEOL) and is able to produce probabilities for all of the mentioned identification levels. In contrary to previous acquisition models, the NVEOL model tries to model the acquisition process physically. This approach allows the use of different sensors and targets without altering the algorithm.

In the NVEOL model, if the observer's field of search covers the area of the target, the acquisition process is started. Until it reaches the observer, the light reflected from the target passes through the atmosphere and is attenuated by atmospheric conditions. This light creates an image in the observer's sensor. If the observer can distinguish the target from the background in this image, the target can be identified. The ability of the observer to distinguish a target from the background is modeled by the use of experimental data. Johnson proposed to use resolution cycles of an image, which are highly contrasted lines, in object discrimination studies. He suggested some fixed cycle counts necessary to achieve 50% probability of recognition in different acquisition levels. 2.1 shows the necessary cycle counts for desired acquisition levels.

The method proposed by Johnson calculates a target's resolution cycles using a function of range and target signature. The signature is a value based on the characteristics of the viewing

Table 2.1: N50 Values for Different Acquisition Levels

| Acquisition Level | N50 Value |
|---|---|
| Detection | 1.0 |
| Classification | 2.0 |
| Recognition | 3.0 |
| Identification | 6.4 |

sensor. If a visual sensor is used, the signature is the contrast difference between the background and the target. If a thermal device is used, the signature is the heat difference between the environment and the target. After the resolution cycle frequency is calculated, this value is used to find the sensor system resolution cycles in the produced image. With N50 value taken as the mean value of the cumulative distribution function, the probability to identify the target at the corresponding acquisition step given infinite time is found. This probability value can be adjusted using the time spent identifying the target to compute the actual probability value.

# CHAPTER 3

# The Proposed Method

The approach proposed in this thesis applies to analysis simulations where representational accuracy is required. It assumes that the terrain data is represented as regular grid data and the objects are represented as polygon meshes. Also, unlike existing LOS optimization approaches, this approach involves the calculation of the acquisition of the objects in LOS. The reason for such a domain choice is that the existing methods for optimization of LOS algorithms generally apply to specific sections of the visual perception calculations. However, when these methods are put into practice, acquisition information is required and the calculations for the acquisition step makes it hard to satisfy the performance requirements of a real-time simulation. The proposed algorithm offers a general solution to the perception calculations of a simulation system and optimizes the case where a complete visual perception model is necessary, the data to be processed is huge, and the time requirements are strict.

Existing methods for visual perception poorly utilize the processing power of the system where the simulation is run. This observation led to the construction of a method aimed to solve the utilization problem. In order to achieve this, we manipulate the order of the basic constructs of the visual perception algorithm. These basic constructs are;

1. View Culling
2. Terrain LOS
3. Object LOS
4. Acquisition

View culling, in the context of visual perception algorithms, is the step where the targets

outside of the viewing frustum of the viewer are marked as not in LOS. This step was not included in the descriptions in previous chapters since this calculation is generally embedded into the terrain LOS or object LOS steps. In this section, the view culling step is considered as one of the main steps of the calculation to better pinpoint the performance effectiveness of the proposed method.

To exemplify the thought process in the development of the method, different orderings of these four constructs used for LOS optimizations are considered.

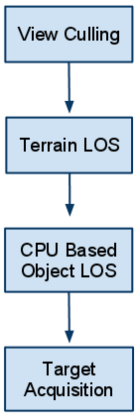



Figure 3.1: Sequential Ordering

The most basic ordering of the perception steps is the most straightforward solution to the problem (3.1). This ordering is taken as the base solution since it is the simplest. This ordering calculates the view culling, terrain LOS, object LOS and acquisition steps sequentially. All of the steps are calculated using the CPU. The only optimization in this approach is to avoid the next steps when a step reports that the target is not in LOS. When compared to the object LOS and acquisition steps, the terrain LOS step is generally the most time and resource consuming step of the algorithm. Because of this, one can observe in this model that the optimization to avoid the next steps does not provide as much gain as the next ordering which switches the places of the object LOS and terrain LOS steps by utilizing the GPU.

The given ordering shown in 3.2 uses occlusion queries to calculate the object LOS. When occlusion queries are used, the view culling step is embedded in the object LOS step since the occlusion queries use the viewing frustum of the viewer to calculate object occlusions. View culling is expected to cull most of the calculations, so the order of the terrain LOS and object

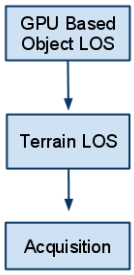



Figure 3.2: Reverse Sequential Ordering with GPU Based Object LOS

LOS are switched to be able to perform view culling as early as possible.

The reverse sequential ordering culls most of the terrain LOS and acquisition calculations but it does not effectively utilize processing power since the CPU remains idle while calculating the object LOS, and the GPU remains idle while calculating the other steps. A different approach proposed by Salomon [16] and Verdesca [18] utilizes the GPU and the CPU at the same time by using the occlusion queries to cull the terrain LOS calculations. It performs a quick occlusion query test on the pre-calculated depth buffer data. Their method handles the object LOS step by checking intersection with the objects on the terrain at each iteration of the ray, so the object LOS can be thought of as a sequentially calculated step. Acquisition step is not mentioned in this method. Since the scope of this thesis contains the acquisition step, it is added at the end of the calculations as a sequentially calculated step which leads to the ordering given below in 3.3.

The ordering which uses terrain culling with sequential object and acquisition tests culls most of the calculations since the terrain culling step is calculated very efficiently. In this model, as an optimization, the object and acquisition tests can be run in parallel with the terrain LOS and terrain culling steps.

In the model shown in 3.4, occlusion queries are used for terrain culling, so it is not possible to use occlusion query based optimizations for the object LOS calculations since a single GPU can not run two different queries in parallel on hardware. Also, the object, acquisition, and terrain tests are calculated on the CPU, which means that the workload is not shared equally among hardware components. The approach proposed in this thesis does not use the terrain culling step as an optimization and calculates the object LOS on the GPU; so, all of

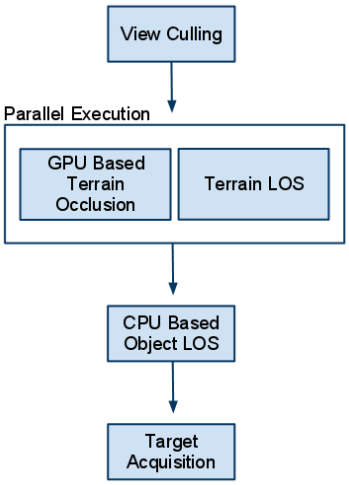



Figure 3.3: Terrain Culling with Sequential Object and Acquisition Tests

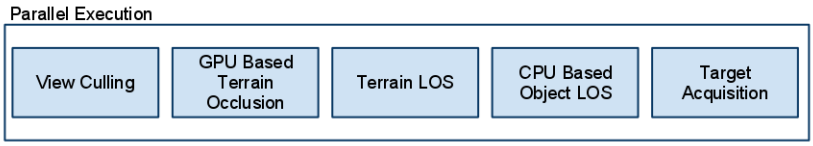



Figure 3.4: Terrain Culling with Parallel Object Acquisition Tests

the perception steps can be run in parallel while effectively utilizing hardware (3.5).

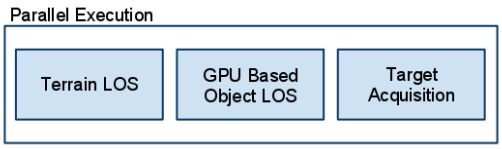



Figure 3.5: Proposed Approach

The approach proposed in this thesis calculates the perception much faster than the existing models. It does not use a view culling phase, which is already embedded in the occlusion query based object LOS step. The embedded view culling culls away most of the calculations in parallel while also calculating the object LOS step. This approach combines the effectiveness of the culling step proposed by Salomon [16] with the faster object LOS step, which greatly reduces the calculation time. One other source of performance increase is due

to better hardware utilization, which is achieved by running three different steps utilizing different hardware components in parallel without any sequential calculations. Also, early out mechanisms are added to the perception steps to further increase the performance of the perception calculation. The algorithms used in the mentioned steps and the introduced early out mechanisms are described in further detail in the next sections.

## 3.1 Algorithm Steps

Proposed approach combines a few different optimizations and aims for full utilization of different hardware components. Most importantly, the terrain LOS and the object LOS steps are separated since they mainly utilize other hardware components. All of the visual perception steps are run in parallel for each entity with visual perception capabilities. When all of the three steps are finished for an entity, the algorithm moves on to the next entity. Each state checks the other states for completion. When a step finishes its calculation and reports that the object is not in LOS, the other steps are expected to abort their calculations. The details of this approach are discussed in the ”Early Out Mechanisms” section.

### 3.1.1 Terrain LOS

In the terrain LOS step one can observe that if the data is huge and the computations for the terrain intersection tests are not complex, the main bottleneck in the terrain LOS calculation becomes the file I/O operations, since in an optimized scenario, the iteration between the nodes can be calculated in a few CPU cycles but the terrain cache is generally unable to hold all of the requested area because of limitations in the hardware (e.g. In 32-bit systems a maximum of 4 GBs can be loaded on main memory). In order to load the requested area, the cache must free some other data which will probably be loaded again when a similar LOS operation is requested. This repeated deleting and loading of terrain data increases the calculation time from a few cycles to milliseconds, because of repeated I/O requests. The caching algorithms can be optimized to an extent, but no proposed solution can be expected to remove the I/O operations completely. So if an optimized terrain LOS algorithm is used, the terrain LOS step is expected to mainly utilize the disk and use the other system resources minimally. This information shows that it could be possible to make other calculations as the

terrain LOS is being processed. If the terrain LOS and object LOS are not separated, this I/O bound operation will block the execution of object LOS as it is generally processed along the iteration of the LOS ray.

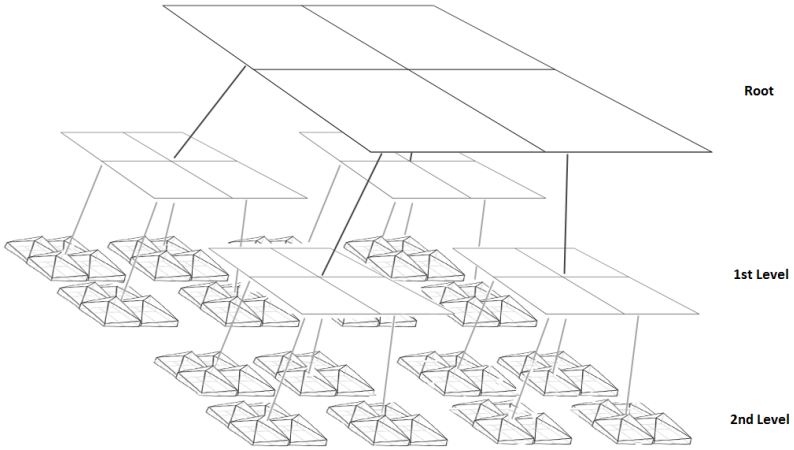



Figure 3.6: Multi Resolution Terrain

A modified version of the terrain representation model by Duvenhage [6] is used for the terrain height data, which is a tree of regular grids that represents a multi resolution terrain hierarchy (3.6). Each node of the regular grid is either a leaf node or a regular grid itself. Each node stores the minimum and maximum height values existing in that node hierarchy. The leaf nodes store a regular grid of height values. This multi resolution terrain hierarchy is selected for its ability to represent DTED formatted height maps which contains different levels of detail in different areas. Also, the stored min/max values optimize the terrain LOS calculations to a great extent especially in special cases such as aerial simulations or spherical earth representations. The terrain LOS calculation algorithm is given as:

**if** $startHeight > maxHeight \wedge endHeight > maxHeight$ **then**
  return $InLOS$
**else if** $startHeight < minHeight \wedge endHeight < minHeight$ **then**
  $losPoint \leftarrow start$
  return $NotInLOS$
**else if** $nodeType = LeafNode$ **then**
  **repeat**

```
        calculate intersection point with next data
        curPoint ← intersectionPoint
        interpolate the height value at curPoint using quad interpolation
        if heightValue < curHeight then
          losPoint ← curPoint
          return NotInLOS
        end if
      until curPoint = end
    else
      prevPoint ← start
      repeat
        calculate intersection point with next node
        curPoint ← intersectionPoint
        recurse using prevPoint and curPoint
        prevPoint ← curPoint
      until curPoint = end
    end if
    return InLOS
```

### 3.1.2 Object LOS

The object LOS step is generally not used, or overlooked for its simplicity in previous research in the field. It focuses on ray/triangle intersections if polygon mesh data is used. Even if objects are represented as bounding volumes, these volumes can be represented as triangle meshes. The ray/triangle intersection tests are good candidates for GPU based solutions. The proposed approach uses occlusion queries for object line of sight tests. It directly separates the terrain representation from the object representation which allows this step to be calculated on another thread of execution. Also this separation also allows the moving objects to be used in LOS calculations where other approaches do not allow this. Occlusion queries are used in such a way that the viewer is placed at the beginning point of the LOS ray looking at the direction of the ray and the scene is rendered from the viewer's view point using occluding objects to fill the depth buffer and the queries are made to check if the query objects are occluded by the other objects. This calculation is done purely on the GPU so the other system

resources are unaffected from this step. The steps of the proposed object LOS calculation are given as:

**for all** *object* in *occludingObjects* **do**
  render object
**end for**
**for all** *object* in *queriedObjects* **do**
  begin occlusion query for object
  render object
  end occlusion query for object
**end for**
**for all** *object* in *queriedObjects* **do**
  get occlusion query result for object
  **if** *renderedPixelCount* > 0 **then**
    $inLos[object] \leftarrow false$
  **else**
    $inLos[object] \leftarrow true$
  **end if**
**end for**

### 3.1.3 Acquisition

The acquisition step is also covered by the proposed approach since the LOS calculations are generally used for calculating visual sensory input. In other scenarios where LOS is only used for checking visibility between points, this step can be omitted easily since the other steps have no dependencies on the acquisition step. The acquisition step in the proposed method uses the MRC-MRT (Minimum Resolvable Contrast - Minimum Resolvable Temperature) based perception model described in the book by Dudzik [5] which is heavily influenced by the NVEOL target acquisition algorithm described in the ”Background” chapter of this thesis. This model uses the MRC and MRT values described in the NVEOL model to calculate the resolvable cycle counts on the target. This value is then used to calculate an acquisition probability for each acquisition stage.

## 3.2 Early Out Mechanisms

When a visual perception step reports that the target is not in LOS, the remaining calculations become obsolete since there is no need to calculate additional information about objects which are not in LOS. When the calculation steps are run sequentially, simple checks can be made not to calculate the remaining steps but when the calculations are run in parallel, a method must be introduced to stop the parallel calculations involving the viewer and the target. In order to stop the parallel calculations, some check-points are added to each step in order to allow the LOS operation to be aborted in case a step reports no visibility between objects.

The check-point for the terrain LOS is checked between the leaf node boundaries. Before the LOS ray iterates to the next node, the abort flag is checked to see if the ray should continue. If the abort flag is set, the calculation is aborted and the next node is not fetched. Also if the terrain LOS ray intersects with the surface the abort flag is set to notify other steps of the algorithm. For the object LOS calculations, the abort flag is checked before rendering the target entity and before each query is sent to the GPU. Also, if a query results as ”not seen”, the abort flag for the corresponding entity is set to abort the other calculations. The acquisition step does not have a main loop so the abort flag is checked before various points of the algorithm which are assumed to be calculation heavy. The main concern in determining these checkpoints is that before a heavy calculation independent of the previous and next calculations, the abort flag should be checked to be able to cull this heavy calculations as early as possible. The determined checkpoints for the acquisition step can be given as:

calculate target's contrast with background
**if** $queryStatus = NotInLOS$ **then**
  return $Bypassed$
**end if**
calculate target's bar frequency
**if** $queryStatus = NotInLOS$ **then**
  return $Bypassed$
**end if**
calculate target's resolvable bar count
**for all** *value* in $N50Table$ **do**
  **if** $queryStatus = NotInLOS$ **then**

```
        return Bypassed
    end if
    check if acquired in the current state
    if acquired then
        return InLOS
    end if
end for
return NotInLOS
```

# CHAPTER 4

# Experimental Results

The results of the performance tests conducted to measure the effectiveness of the proposed method are presented in this chapter. Several models based on existing methods are constructed for comparison. In the following sections, the tested models, scenarios, and the achieved results are described.

## 4.1 Test Models

The models used for testing are the previously described models which use different orderings of the visual perception steps. These models are;

1. Base Model
2. Reversed Base Model with GPU Object Tests
3. Terrain Culling with Sequential CPU Object Tests and Acquisition
4. Parallel Execution of Terrain Culling, CPU Object Tests and Acquisition
5. Proposed Method without Early Out Mechanisms
6. Proposed Method

All of these models use the ”Terrain LOS” algorithm described in the previous section, so the effectiveness of the described min/max tree optimization or TIN data representation is not measured since it is beyond the scope of this thesis. All of the models use the same target acquisition algorithm which is described in the ”Target Acquisition” section. Since the

parallelization methods for occlusion query based visual perception algorithms are studied, some of the models used for testing use occlusion query based object LOS algorithms while some models use CPU based object LOS algorithms. The object LOS algorithm used in the second, fifth and sixth models are described in the ”Object LOS” section in the ”Proposed Method” chapter. The first, third and the fourth models use the algorithm given below as:

```
for all target in queriedTargets do
    construct a ray starting at viewer and ending at target
    for all occluder in occluders do
        for all triangle in occluderTriangles do
            check intersection between ray and triangle
            if intersected then
                return NotInLOS
            end if
        end for
    end for
end for
return InLOS
```

In the models which use the CPU based object LOS algorithm, view culling is performed. Without the use of view culling, these models compare very poorly against the occlusion query based object LOS algorithms, which embed the view culling step in its execution steps. The used view culling algorithm checks the angle and distance between the targets and the viewer to see if they are outside the viewing frustum of the viewer.

As mentioned previously, the tested models mainly differ in the ordering of the perception steps. The base model calculates the view culling, terrain LOS, CPU based object LOS and target acquisition steps sequentially. The reverse base model calculates the GPU based object LOS, terrain LOS and target acquisition steps sequentially. The third model is the implementation of the optimization proposed by Salomon [16] with target acquisition and view culling steps added sequentially. The fourth model uses the same algorithm with parallel view culling and target acquisition steps. The fifth model is the proposed model without the mentioned early out mechanisms. The sixth model is the proposed model with early out mechanisms.

## 4.2 Test Scenarios

The scenarios used in testing are created with the aim of simulating the usage of the visual perception algorithms in combat simulation environments. In combat simulation environments, the perceiving entities and the perceived targets are the combat units. The battlefield also generally contains some occluding, but not perceiving objects like trees or buildings which will be called ”obstacles”. The combat units used in the test scenarios consists of soldiers, vehicles, and aircrafts. The height, width and depth of these units are modeled similar to corresponding military units. Also, these units are modeled as units in motion since the scanning motion and the change of the viewing frustum affect the performance of the perception algorithm.

The test scenarios are categorized according to the count of the entities used and the size of the battlefield terrain. The unit counts used are multiples of the unit counts in the first scenario, which contains 125 soldiers, 25 vehicles and 5 aircrafts. The test scenarios are;

1. Scenario 1 (155 Units, 125 Obstacles, 225 $km^2$ Terrain)
2. Scenario 2 (310 Units, 250 Obstacles, 900 $km^2$ Terrain)
3. Scenario 3 (620 Units, 500 Obstacles, 3600 $km^2$ Terrain)
4. Scenario 4 (1040 Units, 1000 Obstacles, 14400 $km^2$ Terrain)

The terrain for the scenarios were created using randomized fractals which start from the four corners of the square shaped terrain and calculate the mid-point by adding a random height value $x$ to the average of the corner node heights. After the height of the mid-point is calculated the heights of the mid-points of the edges are calculated using the average of the two corner nodes on each edge and again adding a random height value $x$. The algorithm divides the square into four equally big squares and recurses on these squares using half the random height value $x$. Using this method, a very realistic terrain can be created. A screen shot of one the randomly created terrains used in testing can be seen in 4.1.

In the scenarios 3 and 4, the terrain data becomes huge ($> 512MBs$) so the maximum amount of system memory that the terrain uses is limited. The terrain cache used for scenario 3 is limited with 96 MBs of memory and the terrain cache for scenario 4 is limited with 384 MBs of memory. After these limits are passed, the cache deletes some terrain nodes from the main

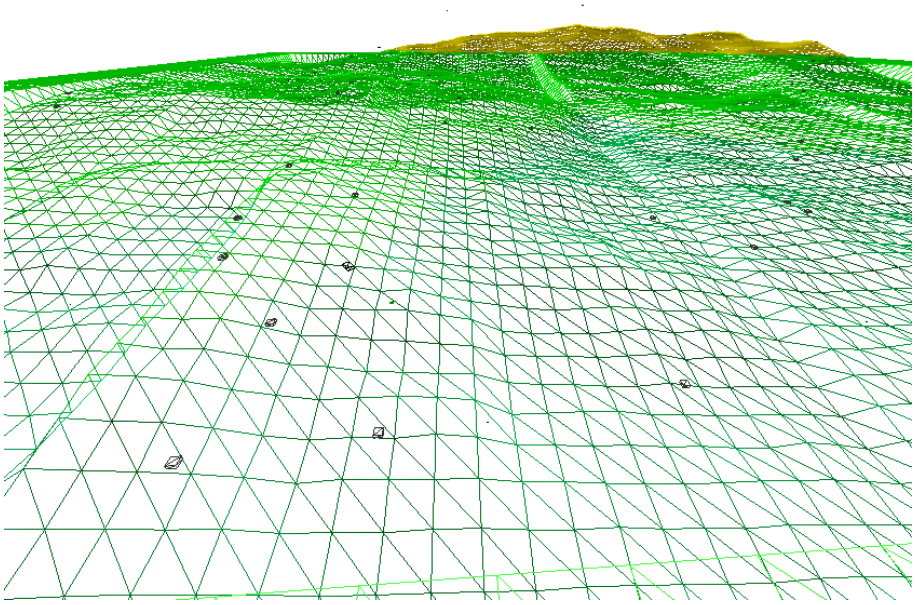


Figure 4.1: Terrain Created Randomly Using Fractals

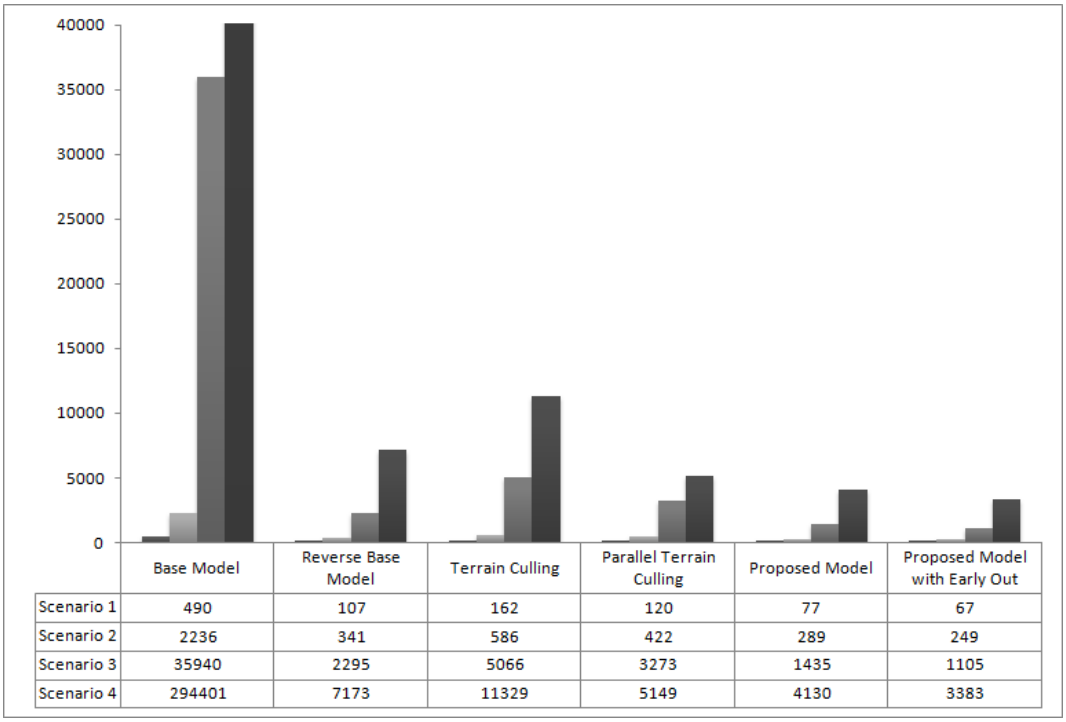
memory to free up space before loading a requested node. The effect of this behavior can be observed in the test results for scenarios 3 and 4.

The target acquisition calculation requires inputs about the state of the sun and moon to calculate the visibility of the targets under environmental lighting conditions. A noon environment where the entities can be perceived easily was chosen for the target acquisition since combat simulations generally use such environments. Also, in lowly lit environments, the target acquisition step is observed to be marking most of the targets as ”Not In LOS”, so that the calculations of the other steps are bypassed by the use of early out mechanisms. By using noon environment, the outputs of the test results becomes less dependent on the target acquisition step.

## 4.3 Test Results

The system used in testing is;

- Intel Core 2 Quad 2.4 GHz




| | Base Model | Reverse Base Model | Terrain Culling | Parallel Terrain Culling | Proposed Model | Proposed Model with Early Out |
|---|---|---|---|---|---|---|
| Scenario 1 | 490 | 107 | 162 | 120 | 77 | 67 |
| Scenario 2 | 2236 | 341 | 586 | 422 | 289 | 249 |
| Scenario 3 | 35940 | 2295 | 5066 | 3273 | 1435 | 1105 |
| Scenario 4 | 294401 | 7173 | 11329 | 5149 | 4130 | 3383 |

Figure 4.2: Comparison of Iteration Times of All Scenarios

- 4 GB Physical Memory
- NVIDIA GeForce 9600 GT

The simulation is run on the given system using a combination of 6 methods and 4 scenarios for a total of 24 test cases. These test cases are run for 10 minutes and various statistics are gathered from these test runs. In these test runs, the simulation uses fixed time steps. For each time step, the perceiving entities, which are soldiers, vehicles and aircrafts, calculate the LOS and acquisition against each other perceiving entity. For example, for Scenario 4, which contains 1040 perceiving units, a total of 1081600 perception states are determined. The time spent in calculating the perception of a single time step of simulation is called the ”Perception Iteration Time”. In 4.2, the comparison of the iteration times of the 6 models are given for all scenarios [1].

In the given chart, the iteration times are measured in milliseconds. The given chart shows that for all of the scenarios, the proposed method performs faster than the existing methods.

[1] In 4.2, the values above 40000ms are displayed as 40000ms because of lack of space.

The average speedup with respect to the other models is given in 4.3.

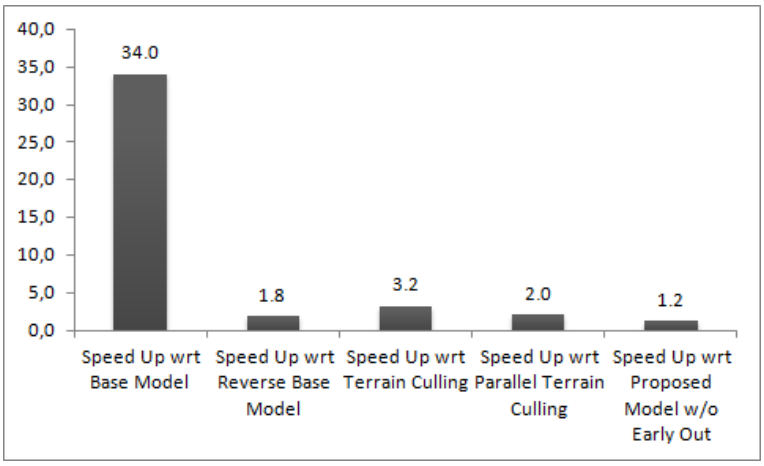
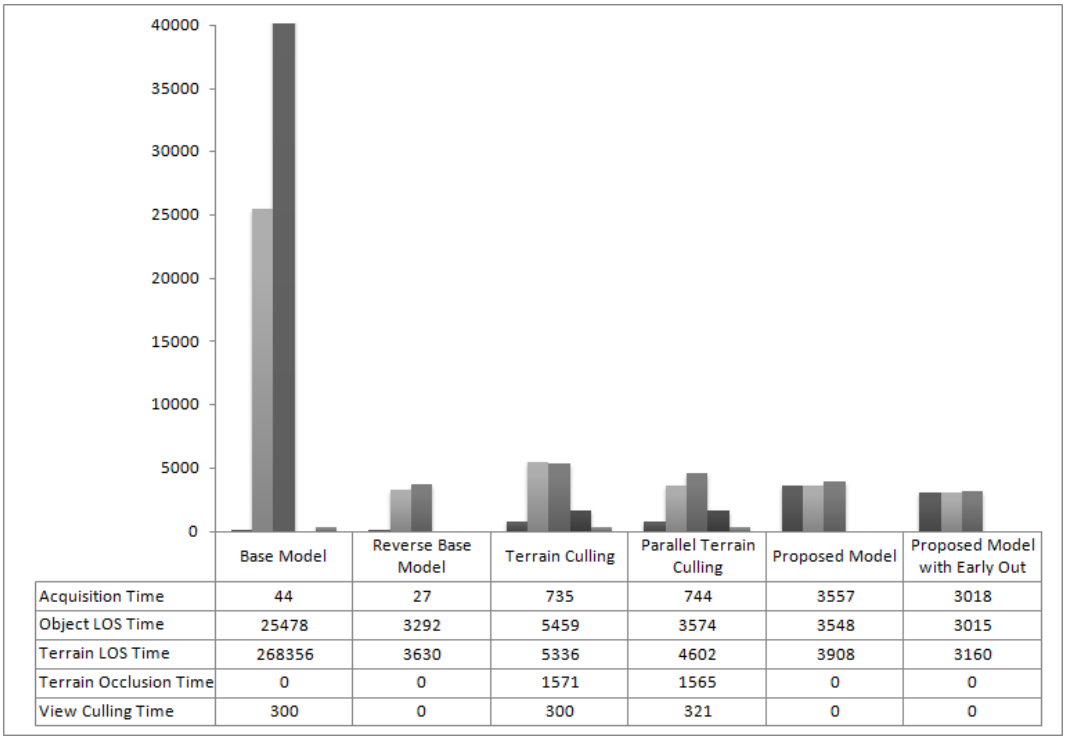



Figure 4.3: Average Speed Up of the Proposed Model

In 4.3, the huge speed up with respect to the base model is caused by the unculled terrain calculations in the base model. Especially in the scenarios 3 and 4, because of repeated usage of I/O operations, the terrain LOS calculations create a performance bottleneck. It can also be seen that the models which use GPU based object LOS easily outperform models that use CPU based object LOS. The results of other statistics such as algorithm step times shows the reason for this. In 4.4 [2] the time spent in calculating various algorithm steps at each iteration of Scenario 4 is given.

[2] In 4.4, the values above 40000ms are displayed as 40000ms because of lack of space.



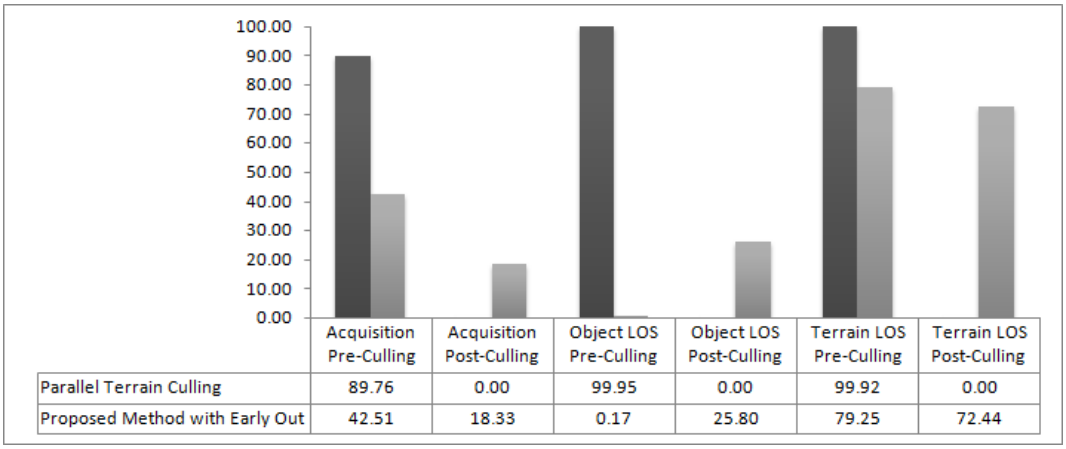

| | Base Model | Reverse Base Model | Terrain Culling | Parallel Terrain Culling | Proposed Model | Proposed Model with Early Out |
|---|---|---|---|---|---|---|
| Acquisition Time | 44 | 27 | 735 | 744 | 3557 | 3018 |
| Object LOS Time | 25478 | 3292 | 5459 | 3574 | 3548 | 3015 |
| Terrain LOS Time | 268356 | 3630 | 5336 | 4602 | 3908 | 3160 |
| Terrain Occlusion Time | 0 | 0 | 1571 | 1565 | 0 | 0 |
| View Culling Time | 300 | 0 | 300 | 321 | 0 | 0 |

Figure 4.4: Comparison of Algorithm Step Calculation Times for Scenario 4

In 4.4, it can be seen that when occlusion queries are used for object LOS both the terrain LOS and object LOS steps execute faster. The occlusion query based object LOS step executes faster than the CPU based object LOS step because of the algorithmic complexity of the CPU test. When the algorithms of the GPU based object LOS and the CPU based object LOS are compared, it can be seen that while the CPU based object LOS does $O(n^3)$ calculations, the GPU based object LOS calculates the outcome of object occlusion in $O(n^2)$ calculations. Also, because the GPU calculations involve the view culling step, these models calculate the terrain LOS step slightly faster.

Another important information gained from 4.4 is the utilization of hardware components in different models. We can derive from the given figure that, the main source of performance increase of the proposed method lies in its ability to divide the calculation equally among different hardware. In the proposed method, the time it takes to calculate the terrain LOS, object LOS and acquisition is almost the same, while the existing methods can not reach the same utilization, even when the algorithm steps are run in parallel. This is achieved by proper usage of hardware utilization by assigning different tasks to different hardware

components and by the usage of early out mechanisms to stop unnecessary calculations and thus, completing each step at the speed of the fastest running step. It can be seen from the previous figure ( 4.4) that the usage of early out mechanisms speeds up the proposed model by a factor of 1.2x. This performance is gained by checking the completion of other steps while calculating a perception step. The percentages of calculations culled in Scenario 4 using the parallel terrain culling model and the proposed model are given in 4.5.



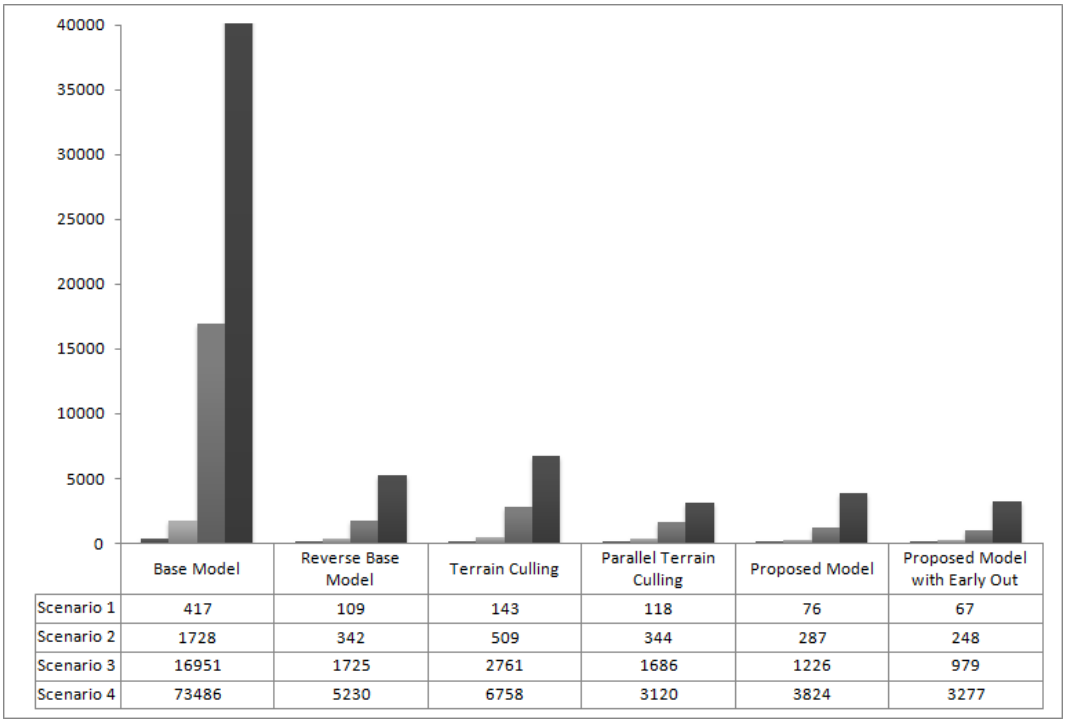

| | Acquisition Pre-Culling | Acquisition Post-Culling | Object LOS Pre-Culling | Object LOS Post-Culling | Terrain LOS Pre-Culling | Terrain LOS Post-Culling |
|---|---|---|---|---|---|---|
| Parallel Terrain Culling | 89.76 | 0.00 | 99.95 | 0.00 | 99.92 | 0.00 |
| Proposed Method with Early Out | 42.51 | 18.33 | 0.17 | 25.80 | 79.25 | 72.44 |

Figure 4.5: Percentages of Culled Calculations

In the 4.5, the rows named as ”Pre-Culling” are the calculations culled before the algorithm step begins. This culling step is a straightforward approach and it is used in all of the tested models. The rows named as ”Post-Culling” refer to the early out mechanism which culls the calculations after the calculation starts. In the parallel terrain culling model, most of the calculations are already culled before they begin. This is achieved by the usage of the view culling and the terrain culling steps. The proposed model, additionally, culls the calculations on the fly. The effect of the view culling step embedded in the occlusion query based object LOS step can be observed in this chart. The object LOS is calculated faster than the terrain LOS because of the I/O operations that the terrain LOS requires, so most of the object LOS calculations are not culled at the start. While the object LOS is being calculated, the terrain LOS calculations are mostly post-culled. After most of the object LOS calculations are completed, the terrain LOS calculations become pre-culled. During this time, the terrain calculations that finish before the object calculations stop them. Acquisition, like the object LOS step, is calculated faster than terrain LOS and interacts with the terrain LOS step similar

to the object LOS step.

The given test results are based on tests calculated using flat earth model. The effects of using flat earth representation and spherical earth representation models are described in the ”Background” chapter. In 4.6, the results of the same test cases, run using the spherical earth representation is given.




| | Base Model | Reverse Base Model | Terrain Culling | Parallel Terrain Culling | Proposed Model | Proposed Model with Early Out |
|---|---|---|---|---|---|---|
| Scenario 1 | 417 | 109 | 143 | 118 | 76 | 67 |
| Scenario 2 | 1728 | 342 | 509 | 344 | 287 | 248 |
| Scenario 3 | 16951 | 1725 | 2761 | 1686 | 1226 | 979 |
| Scenario 4 | 73486 | 5230 | 6758 | 3120 | 3824 | 3277 |

Figure 4.6: Comparison of Iteration Times of All Scenarios using Spherical Earth Representation

When spherical earth representation is used, the terrain LOS calculations are done much faster since more terrain calculations are culled by the min/max tree optimization used for the terrain. This is caused by the bending of the LOS ray toward the center of the earth. Since the LOS ray moves closer to the earth surface in spherical earth model, more points begin to pass the minimum height of the min/max tree and more calculations are culled. The effect of faster terrain calculations reduce the equally distributed utilization effect of the proposed method, thus producing the given speedups in 4.7.

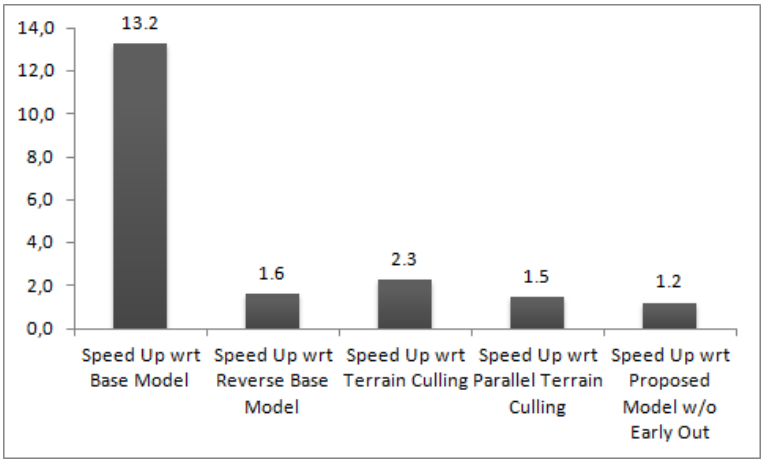



Figure 4.7: Average Speed Up of the Proposed Model using Spherical Earth Representation

# CHAPTER 5

# Conclusion

In today's simulation systems, the increase in hardware capabilities leads to more detailed and more resource consuming simulations. Compared to the systems of the past, today's simulation software processes so much detail that, the models developed for simulations systems of the past need to be constantly updated and optimized by the use of new technologies to catch up with the current requirements. The visual perception algorithms are a perfect example to this with many technologies being used in optimizations to enhance the performance and increase the level of detail achieved.

With the current developments in multi-core architectures, the scalability has become one of the most important topics in software and systems development. Even fairly new technologies, such as hardware implemented occlusion queries begin to lose popularity against newer technologies such as CUDA, because of scalability issues. In this thesis, the parallelization problems incurred by the use of occlusion queries are taken into consideration and a novel approach in hardware utilization is proposed to optimize the visual perception models using occlusion queries.

The proposed method uses the occlusion queries to calculate the occlusion of objects in the system. Such a usage also provides view culling, so that the terrain LOS and acquisition steps executed in parallel perform faster. Also, object LOS tests are calculated faster on graphics hardware by the use of hardware accelerated occlusion query facilities. This speedup in the object LOS step effects the terrain LOS calculations by the usage of pre-culling and post-culling steps to end the calculations of the other steps when a perception step reports that the object is not in LOS. With the addition of checkpoints to various perception steps to end the calculations early, the proposed method is tested to be about 2 times faster than

straightforward parallel execution of existing methods. If spherical earth representation model is used, our method performs 1.5 times faster than the compared methods.

As the proposed method was tested, the GPU based object LOS and CPU based object LOS algorithms were observed to be producing slightly different results, especially when calculating the LOS of distant objects. This difference is caused by the discretization of the environment in the rendering stage of the occlusion queries. When a distant object is rendered, it can produce an image too small to fill a pixel; so, it can be falsely reported as not in LOS. Since the acquisition process also categorizes distant objects as unseen objects, the error caused by the usage of occlusion queries was ignored in the scope of this thesis. Future work in the area includes the investigation of correctness of the occlusion query results and determining the exact error rates caused by this discretization. Other future work includes the usage of newer technologies such as CUDA in the place of occlusion queries and determining the effect of such usage to the performance and correctness of the visual perception algorithms.

# REFERENCES


[1] National Geospatial-Intelligence Agency. Digital terrain elevation data (dted) specification. http://www.nga.mil/ast/fm/acq/89020B.pdf Last Accessed Date: 10/06/2010, 1996.

[2] S. Bangay. Parallel implementation of a virtual reality system on a transputer architecture. Master's thesis, Rhodes University, Grahamstown, South-Africa, 1993.

[3] V. Boyer and J.J. Bourdin. Auto-adaptive step straight-line algorithm. *IEEE Computer Graphics and Applications*, 20(5):67–69, September / October 2000.

[4] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4:25–30, March 1965.

[5] M. C. Dudzik and J. D. Howe. *The Infrared and Electro-Optical Systems Handbook*, volume 4, chapter Electro-Optical Imaging System Performance Prediction. 1993.

[6] B. Duvenhage. Using an implicit min/max kd-tree for doing efficient terrain line of sight calculations. In *Proceedings of the 6th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*. ACM New York, NY, USA, 2009.

[7] N. Govindaraju, A. Sud, S.E. Yoon, and D. Manocha. Parallel occlusion culling for interactive walkthroughs using multiple gpus. In *Proc. IEEE Workshop on Commodity Based Visualization Clusters*. Citeseer, 2002.

[8] J.K. Hartman, S. Parry, and W.J. Caldwell. 1992.

[9] Hewlett-Packard. Opengl occlusion test extension preliminary. http://www.opengl.org/registry/specs/HP/occlusion_test.txt Last Accessed Date: 28/05/2010, 1997.

[10] B. Hofmann-Wellenhof and H. Moritz. *Physical Geodesy*. Springer Verlag, 2005.

[11] J. Johnson. Analysis of image forming systems. In *Proceedings of Image Intensifier Symposium*, pages 249–273, 1958.

[12] B.O. Koopman and Columbia Univ. New York. Search and screening oeg report no. 56. Technical report, Operations Evaluation Group, Navy Deptartment, 1946.

[13] F. Kuhl, R. Weatherly, and J. Dahmann. *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*. Prentice Hall PTR Upper Saddle River, NJ, USA, 1999.

[14] C. Miller and R. Laflamme. The digital elevation model-theory and application. *Photogrammetric Engineering*, 24:433–442, 1958.

[15] T.K. Peucker, R.J. Fowler, J.J. Little, and D.M. Mark. The triangulated irregular network. *American Congress on Surveying and Mapping*, 1980.

[16] B. Salomon, N. Govindaraju, A. Sud, R. Gayle, M. Lin, and D. Manocha. Accelerating line of sight computation using graphics processing units. In *Proceedings of the Army Science Conference (24th)*, December 2004.

[17] F. Shields and W. Lawson. *E-O System Evaluation at NVEOL.* PhD thesis, Night Vision and Electro-Optics Laboratory, Visionics Division, 1980.

[18] M. Verdesca, J. Munro, M. Hoffman, M. Bauer, and D. Manocha. Using graphics processor units to accelerate onesaf: A case study in technology transition. *Journal of Defense Modeling and Simulation*, 3(3):177, 2006.