Extension On Navigation Meshes

Bachelor Thesis
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

Author : Sabin Gurung
Director : Nuria Pelechano
Index
# Table of Content

1 Introduction 10
   1.1 Project Formulation 12
      1.1.1 Main objectives 12
      1.1.2 Secondary objectives 13
   1.2 Personal motivation 13
   1.3 Project Structure 13

2 NEOGEN 3D 18
   2.1 Overview of NEOGEN 3D 18
   2.2 Subprocesses of NEOGEN 3D 19
      2.2.1 Terrain generation 19
      2.2.2 Slope constraint 19
      2.2.3 Height constraint 21
      2.2.4 Obstacle boundary detection 23
      2.2.5 Notch Detection 25
      2.2.6 Notch Elimination 25
      2.2.7 Cell portal graph creation 27

3 Complex and flexible pathfinding 31
   3.1 Off-Mesh Portal 32
      3.1.1 Handling staircases 33
      3.1.2 Handling jumps 35
   3.2 Additional information on nodes and edges 36
   3.3 NeoPathFinder (Program Implementation) 37
      3.3.1 Algorithms 38
      3.3.2 Core data structures 38
      3.3.3 NEOGEN adapter 39
   3.4 A* specialization 40
      3.4.1 Heuristic function 40
      3.4.2 Movement cost 41
      3.4.3 Successor generation 41

4 Limitations of NEOGEN 3D 46
4.1 Methodology to detect the limitations 46
4.2 Debugger Application 46
4.3 Detected limitations of NEOGEN 3D 46
  4.3.1 Non manifold mesh 46
  4.3.2 Height Constraint 50

5 Project Management 57
  5.1 Planning 57
    5.1.1 Program Duration 57
    5.1.2 Tasks
      Initial analysis and design 57
      Understand the existing code 58
      Setting up the environment 58
      Designing a solution 59
      Program Implementation 59
      Experimentation and Result 59
      Project management course 60
    5.1.3 Time Allocation For Tasks 60
    5.1.4 Problems and Alternatives 61
  5.2 Development Tools 61
  5.3 Budget 63
    5.3.1 Hardware cost 63
    5.3.2 Software cost 63
    5.3.3 Indirect Cost 63
    5.3.4 Staff Cost 64
    5.4.5 Additional Cost 64
    5.4.5 Total Cost 65
  5.4 Sustainability Analysis 65
    5.4.1 Economic 65
    5.4.2 Environmental 66
    5.4.3 Social 66

6 Conclusion 71

References 73
1 Introduction
1 Introduction

Path planning is the field of Artificial Intelligence (AI) whose objective is to study and research algorithms that plan the optimal route to traverse from one point to another in a given environment. They are primarily utilized in the field of robotics and simulation programs such as video games. Even though robots are used in the real world and virtual characters are present in games, they both use similar real time path planning techniques. In the case of video games, environments have different set of restrictions and rules. For example, being able to pass through portals or jump 5m high to reach the end goal. Whereas in the real physical world, different rules and restrictions are present which are more restrictive. Virtual characters present in virtual environment must be able to navigate the environments without violating the restrictions of the environment.

Path planning problems can be separated into two parts:
- Representation of the environment.
- Algorithms to find solution.

Graph theory is a simple yet very powerful concept. A graph is a data structure consisting of nodes and edges connecting the nodes. They are versatile data structures that are able to represent many real life problems, states or environments in a simple and easy to process manner. Many path finding algorithms that have been researched are based on graphs. The algorithms are optimal and fast enough to be able to be used in real time application. They are general, so they can be tailored to any situation. A-Star (A*) path finding is the widely used algorithm to resolve path finding problems in video games.

In a virtual environment such as video games, autonomous characters are agents that utilize path planning algorithms to navigate from location to location in a convincing manner. In order to achieve this
they use a combination of global and local navigation techniques. Global navigation consists of using algorithms to plan the path. While traversing the planned path, local navigation is utilized in order to produce behaviors such as steering away to avoid obstacles or other agents.

The widely used way to represent the environments in video games is Navigation Meshes (Navmesh). Navigation meshes are collections of 2D convex polygons. Each polygon defines an area in an environment that is traversable by agents. In this project our objective is to improve and extend navigation meshes in order to achieve more complex and advanced navigation features.
1.1 Project Formulation

1.1.1 Main objectives

Navigation meshes are important data structures which are used in path planning algorithms. The main reason is because they are very efficient and flexible.

In real life applications, for a given environment its navigation mesh is generally created manually. They may be automatically generated partially but majority of the work is done manually. Automatic generation of navigation meshes aims to generate navigation meshes for a given environment with little to no manual intervention.

A set of algorithms that generate near optimal navigation meshes for an input environment have been studied in the paper *A Framework for Navigation of Autonomous Characters in Complex Virtual Environments* by Ramon Olivia and Dr. Nuria Pelechano. NEOGEN (Near-Optimal Generator of navigation meshes) is a framework written by the authors that includes the algorithms introduced in the paper. The framework is written in C++.

The output navigation meshes of NEOGEN can be used to plan simple paths (walk from point A to B if there exist sequences of convex polygons that the character can walk through). But a path can be complex (may include swinging from a rope, jumping, etc). **The primary objective of this project is to find ways how the navigation mesh generated by NEOGEN-3D so that it can be used for more advanced and complex pathfinding**. For example climbing stairs, jumping from platform to platform, etc.
1.1.2 Secondary objectives

This project also pursues the following secondary objective of detecting limitations of the current implementation of NEOGEN-3D and propose solutions to overcome those limitations.

1.2 Personal motivation

I have always been obsessed with video games with all my life. I preferred this source of entertainment than ones provided by movies and books. I find it much more engaging.

Only recently I have started seeing video games in a different way other than just a source of entertainment. The underlying techniques utilized to render such artistic graphics, the precise calculation and real life theories applied to simulate realistic physics and AI techniques to simulate complex behaviors is what I like to understand and research.

In this project I would like to utilize the fundamentals that I learned throughout my informatics degree and my ability as a programmer to extend the path planning techniques to allow complex navigation in specific situations.

1.3 Project Structure

The documentation of the project is separated into 6 chapters. In Chapter 2 provides an overview of NEOGEN 3D framework. The details of the implementation of NEOGEN 3D Framework will also be discussed as this knowledge is required to carry out the project as well as following the subsequent chapters. Then chapter 3 will explain how NEOGEN 3D can be extended to use advanced and complex pathfinding. This chapter will also introduce concepts of off mesh linking and attaching additional information to nodes and edges of the
graph. In addition, how these concepts can be implemented into a C++ framework will also be explained. **After that in chapter 4 we will discuss about the limitations of current version of NEOGEN 3D.** Solutions to overcome the detected limitations of NEOGEN 3D is also included. **Subsequently chapter 5 will contain the details of project management.** This chapter will explain the task distribution, development tools, budget analysis and sustainability analysis of the project. **Finally chapter 6 will provide conclusion for the project.**
2 NEOGEN 3D
2 NEOGEN 3D

As the project consists of extending an existing code, understanding the existing code of the framework and the concepts and algorithms utilized in the framework is an important task. This chapter provides an overview of the NEOGEN 3D framework and the concepts introduced in the paper *A Framework for Navigation of Autonomous Characters in Complex Virtual Environments* by Ramon Olivia and Dr. Nuria Pelechano, as this knowledge is required to follow incoming chapters.

2.1 Overview of NEOGEN 3D

As mentioned in chapter 1, NEOGEN 3D is a framework written in C++ for automatic generation of navigation meshes. For any arbitrary 3D scene as an input, an output of near optimal navigation mesh of the scene is generated.
2.2 Subprocesses of NEOGEN 3D

The algorithms as a whole contains several individual steps.

- Terrain generation
- Slope Constraint
- Height Constraint
- Obstacle boundaries
- Notch detection
- Notch elimination
- Cell Portal graph

fig. sub steps of navmesh generation in NEOGEN 3D

2.2.1 Terrain generation

Based on the modeling software used, 3D objects are saved in different formats (obj, 3Ds, blen, etc). Through different libraries and third party resources, files of these format can be parsed and universally used way to represent 3D object, vertex and index arrays can be easily obtained. The vertex and index arrays are used to initialize a special data structure in NEOGEN call NeoTerrain.

2.2.2 Slope constraint

Before generation of navigation meshes, it is important to filter out faces that are not traversable. One of the ways to achieve this is through slope constraint.
For a given slope constraint $Sc$, if the slope of a face is greater than $Sc$, then the face is considered an obstacle, if not then the face is considered a walkable face. Walkable face is a face that is traversable while obstacle face is a face that is not traversable.

fig. computing the slope of the face by finding the angle between the normal of the face and the up vector $(0, 1, 0)$.

fig. (left to right, top to bottom) (i) Original mesh with face normal indicated by light blue line (ii) output when all faces satisfy slope constraint (iii) output when some faces do not satisfy slope constraint. (blue faces are walkable, red faces are obstacle)
2.2.3 Height constraint

Even though a face is computed as walkable by the previous process (slope constraint), if we take the height of the agent into account, it is not guaranteed that all the region of the face will be traversable. While taking height of an agent into account, a region is traversable by the agent if the vertical distance between the region and all the objects right above the region is greater than the height of the agent.

fig. (from left to right) (i) all of the face is traversable by the agent (ii) portion of the face is traversable by the agent. (agent is the green box, blue shows region traversable by the agent and red shows the region not traversable)

In the figures above we illustrate how height constraint affects the walkable region. When there exists objects with vertical distance to the floor less than the height of the agent, only the portion right below the object is not traversable whereas the rest of the floor is traversable.

In this process of the nav mesh generation, for each walkable face, initially objects right above it with vertical distance less than the height of the agent is detected. From the detected objects only regions violating the height constraint is detected. Subsequently, the detected regions of the detected objects are projected vertically to the walkable face. Finally the projected region is clipped out from the walkable face.
fig. (from left to right) (i) initial scene with blue denoting walkable region (ii) detection of objects that violate height constraint (iii) detection of regions violating height constraint (iv) projecting the region to the walkable face, (v) clipping out the projected region from the walkable face.
2.2.4 Obstacle boundary detection

The next step after height constraint is obstacle boundary detection. Obstacle boundary is a polygon where all the edges of the polygon are obstacle edges. An edge is an obstacle if it is defined as an obstacle or if it is one of the edges of an obstacle face or the edge has only one incident face.

- There are two types of obstacle boundaries,
- outermost boundary of the objects
- boundary representing the holes inside a boundary

These two types of obstacle boundaries are distinguished using the orientation of the boundary.
The obstacle boundary is computed by first finding an obstacle edge and finding the subsequent obstacle edges until you create a complete polygon. Finding the subsequent edge is achieved by finding the next edge of the current face, including it in the obstacle boundary if the edge is an obstacle, if not, traverse the edge to the next face and repeat the process.
2.2.5 Notch Detection

Notch is a vertex of a polygon containing interior angle greater than 180 degrees. From the obstacle boundaries detected in the previous step all the notches are detected.

fig. notches (red squares) of the obstacle boundaries.

2.2.6 Notch Elimination

The detected obstacle boundaries, which can be abstracted to polygon with holes needs to be converted to navigation meshes. Therefore it needs to be split to a set of convex polygons. This is achieved by adding portals on notches so that the resulting angles are less than 180 degrees.

fig. generating navigation mesh by adding a portal (blue) on notch (red) resulting in two convex angles.
The main algorithm consists of iteratively transforming all the notches into convex vertices. For each notch an **Area of Interest (AOI)** is defined. AOI for a notch $v_i$ is the area between the lines generated by extending $v_{i-1}v_i$ and $v_{i+1}v_i$.

It is proven if a line $v_{ip}$ is created where $p$ is a point anywhere in the AOI of $v_i$, the line $v_{ip}$ will split the initially notch $v_i$ into two convex vertices.

Given we have a notch with interior angle $180 - a > 180$, which can be split into two new vertices with interior angle $b$ and $c$. The portal needs to be created in AOI so that $180 - a < b < 180$.

Firstly, let us prove that if a portal is created outside of AOI, their will still remain one notch. If $b > 180$, then $b$ is a notch. If $b < 180 - a$,

\[
360 = a + b + c \\
360 - a - c = b \\
360 - a - c < 180 - a \\
c > 180 (c \text{ is a notch})
\]

Secondly, let us prove that if a portal is created inside of AOI, both interior angles $b$ and $c$ will be convex. If $b < 180$, $b$ is a convex angle by
definition. So we just need to prove if \( b > 180 - a \), then the other interior vertex \( c \) will also be convex.

\[
360 = a + b + c \\
360 - a - c = b \\
360 - a - c > 180 - a \\
c < 180 \quad (c \text{ is convex})
\]

The algorithm creates a portal in the AOI region and in different ways depending on the composition of the vertex, edges and portals inside this region. The algorithm also eliminates any detected redundant portals.

### 2.2.7 Cell portal graph creation

After generating convex polygons by eliminating notches, a Cell Portal Graph (CPG) is generated. CPG is a graph structure where the cell represents convex polygon and portal represents the edge connecting the convex polygon with each other.

fig. complete navmesh generation process. (from top to bottom) (i) initial mesh (ii) slope constraints (iii) height constraint (iv) boundary detection (v) notch detection (vi) CPG generation.
3 Complex and flexible pathfinding
3 Complex and flexible pathfinding

The Cell Portal Graph (CPG) automatically generated by NEOGEN 3D for any arbitrary 3D scene at present only supports planning very simple paths. The computed path are limited to traversing from one cell to another cell if and only if the cells are connected by portals. As a result, even if an agent is able to jump, fly or if there exist certain cells that has the ability to teleport agents to another cell, currently such variables are not taken into account.

![Diagram of CPG with portals](image)

fig. A path where an agent is able to jump is not achievable with the CPG generated from NEOGEN 3D.

In this chapter we will investigate some concepts that enable us to extend NEOGEN 3D framework so that it can be used for more advanced path planning. In section 3.1 we will introduce a concept of Off Mesh Portal and how it can be used on NEOGEN 3D. Subsequently, section 3.2 will introduce the idea of storing additional information to cells and portals. Section 3.3 explains the code implementation of these concepts in to work with NEOGEN 3D.
3.1 Off-Mesh Portal

Off-mesh portal refers to a portal that connects two cells even if the cells are not connected physically. It is a concept that is used in popular video game engines such as Unity 3D, Unreal Engines, etc.

Off-mesh portal enables us to create links between any arbitrary cells allowing a path to exist even if they are not connected physically. The main advantage of off-mesh portal is the flexibility as just adding a different type of edge between two nodes allows us to compute every possible path while using the same graph algorithms such as A star, Breadth First Search (BFS), etc.

One of the disadvantages of off mesh portal is the difficulty to automate its generation. The generation of on-mesh portal is deterministic as only one case determines its creation: cells must be physically connected. The generation of off mesh portal is non deterministic as there can be several cases: jump distance, step height, teleport range, etc. As a result it hard to automate generation of off-mesh link.
One of the ways to automate the generation of off-mesh portals is to turn the process into a deterministic one. This is done by preemptively finalizing a finite number of cases for the generation.

### 3.1.1 Handling staircases

One of the limitations of current version of NEOGEN 3D is that it did not handle stair cases.

![Staircase](image1.png)

**(a)** a staircase during applying slope constraint. Blue represents walkable area and red represents obstacle faces. **(b)** the resulting navigation mesh generated.

As seen in the figure, the navigation mesh generated for a staircase does not contain portals connecting the walkable regions as they are not physically connected. As a result each walkable region of the staircase is not reachable from the next and previous region.

Off-mesh portals can be utilized to connect walkable regions the walkable regions.
For the automatic generation of off mesh portals for handling staircase, the algorithms will be based on a parameter step height which represents max limit of the vertical distance between any two reachable region.

An alternative solution to off mesh portals for handling staircase would be transforming the steps into connected slanted surface. The disadvantage of this alternative is the loss of accuracy as we are assuming and reconstructing the scene. In addition, this lack of specific information about the surface where the character is walking introduces a lot of artifacts during animation.
3.1.2 Handling jumps

Handling jump between cells is also resolved by off mesh portals. For the automatic generation, the parameter max jump distance will be taken into account. For each cell, if there exists another cell in distance less than max jump, then an off mesh portal between them will be created.
fig. an off mesh link between c1 and c2 is created if the distance \( d(c1, c2) \) is less than max jump distance.

### 3.2 Additional information on nodes and edges

In section 3.1, we introduced the concept of off mesh link that enabled a cell to be reachable from another even if they are not physically connected. As a result, the path planning is able to compute more complex paths. At the moment, only one path can be computed as there is a lack of information required to properly traverse the path by agent. While traversing a computed path, an agent must take appropriate action such as swimming if they are on water, changing the velocity while climbing uphill, etc.

![Diagram](image.png)

fig. c1 is region representing water, c2 is represents land, green line represents the computed path from A to B.
In the figure, while an agent traverses a computed path, as long as it is traversing c1, the agent must perform a swim action. And when it is traversing c2, the agent must perform walk action.

This can be solved by attaching additional information to the cells and portals. So that, when an agent traverses through the path, it will process the attached information and perform the necessary actions.

3.3 NeoPathFinder (Program Implementation)

In this section we will describe the coding implementation of the concepts introduced in the previous section as well as how they interact with current version of NEOGEN 3D framework.

The NeoPathFinder framework can be separated into 3 blocks:
- Algorithms
- Core data structures
- NEOGEN adapter
3.3.1 Algorithms

This block contains classes related to graph search algorithms. There is a general implementation of the A* algorithm. Sub-components of A* such as computing heuristic, movement cost, etc, are implemented as interface so that the algorithm can be extended for any specific needs.

![UML diagram of algorithm blocks](image)

fig. uml diagram of algorithm blocks. (member functions and variables are omitted)

3.3.2 Core data structures

This block contains the core data structures. We have nodes, edges and graphs.
3.3.3 NEOGEN adapter

This code block glues the pathfinding to the existing NEOGEN 3D framework. It consists of a specific implementation of interface and classes introduced in previous blocks and adapts them to work with NEOGEN 3D.
**NeoNavMeshNode**: Represents a cell of the navigation mesh generated by NEOGEN 3D.

**NeoNavMeshEdge**: Connects two nodes in a graph making one node reachable from the other. Has an enumeration determining its type: OFF_MESH and ON_MESH). Also represents portal of the navigation mesh generated by NEOGEN 3D.

**NeoNavMeshGraph**: Graph structure consisting of NeoNavMeshNode and NeoNavMeshEdge connecting then.

**NeoNavMeshInitializer**: Takes output of NEOGEN 3D and converts it to NeoNavMeshGraph which is later used in pathfinding. Also responsible for generating off link edges.

**NeoPathFindAlgs**: Implements the interfaces of the algorithm block.

**NeoAgent**: Represent the agent that traverses the path. Contains properties that permits and restricts navigation of certain NeoNavMeshNode and NeoNavMeshEdge.

**NeoPathFindEngine**: Responsible for handling pathfinding request for an agent and returning solution path if there exists one. Also responsible for adding additional information to the computed path.

### 3.4 A* specialization

The path planning algorithm utilized in NeoPathFinder is A-star. A* is an efficient general algorithm for pathfinding in graph data structures. In this section we cover how we specialize the general special algorithm to fit our needs.

#### 3.4.1 Heuristic function

The heuristic function of a given cell is the distance between the centre of current cell and the centre of goal cell.
fig. yellow cell is the current cell, green is the goal cell, the length of the line represents the heuristic value of the yellow cell.

### 3.4.2 Movement cost

The movement cost from one cell to its neighbour is the distance between their centres (similar to heuristic function).

### 3.4.3 Successor generation

For a given cell, all the other cells that are reachable will be the successors. The cells may be connected by either off mesh or on mesh portals.
fig. (from left to right) (i) original mesh (ii) generated navmesh (iii) generated off mesh links (blue lines) (iv) computed path while moving from white cell to green cell.
4 Limitations of NEOGEN 3D
4 Limitations of NEOGEN 3D

4.1 Methodology to detect the limitations

The methodology used for the detection of limitations of the current version of NEOGEN consists of the following steps.
- Detect errors with an arbitrary 3D map.
- Detect on which sub-step of the algorithm the error is occurring.
- Hypothesize the issue for the problem.
- Confirm the hypothesis with simpler map.

4.2 Debugger Application

A graphical application needed to be developed to facilitate the debugging process. The application must consist of the following features:
- load and visualize arbitrary 3D objects.
- separately execute different steps of navigation generation process and visualize results in each step.
- a console feature to quickly execute basic instructions such as printing vertex and face info.

4.3 Detected limitations of NEOGEN 3D

In this section the detected bugs and limitation of NEOGEN are documented.

4.3.1 Non manifold mesh

NEOGEN is not capable of handling non manifold mesh. A non manifold mesh is a mesh containing either non manifold edges or a non manifold vertices. A non manifold edge is an edge connecting two faces with normals facing opposite direction or an edge connecting more than two faces. A non manifold vertex is vertex whose incident vertices do not
form a closed or open fan. A non manifold mesh is a common issue in algorithms and programs that deal with meshes. Often algorithms and programs only work with manifold meshes. For example 3D printing requires the mesh to be manifold, etc.

![fig. simple examples of non manifold geometry](image)

The specific step of NEOGEN that is not able to handle non manifold mesh is the detection of obstacle boundary. As mentioned earlier obstacle boundary is computed by first finding an obstacle edge and finding the subsequent obstacle edges until you create a complete polygon. Finding the subsequent edge is achieved by finding the next edge of the current face, including it in the obstacle boundary if the edge is an obstacle, if not, traverse the edge to the next face and repeat the process. For the process of detection of obstacle boundary to terminate, it is required to reach the initial edge of the obstacle boundary to complete the polygon. For a non manifold mesh we are not able to reach the termination condition of obstacle boundary detection process.
As seen in the figure the boundary detection process will never terminate as the process never reaches the starting edge. The process keeps on repeatedly switching between the last and second last iteration as shown in the figure. As the result NEOGEN 3D crashes when handling non manifold meshes.

The ideal scenario would be to have all input 3D meshes being manifold meshes. Which results in NEOGEN 3D having this prerequisite. As mentioned earlier, often algorithms and concepts regarding meshes have prerequisite of input being a manifold mesh. However, in practice it is hard to maintain this prerequisite as it is challenging for 3D modellers and designers to design complex 3D scenes while maintaining the manifold property.
One possible solution to this limitation will be to preprocess the input mesh to transform it into a manifold mesh if it is a non manifold mesh automatically. Then run the NEOGEN 3D framework as usual with the preprocessed mesh. The advantage of this solution is that there is no need to modify the existing NEOGEN 3D algorithm.

There have been studies regarding repairing a mesh to turn it into a non manifold. Most of the algorithms often result in the output manifold mesh having some approximation.

A simpler way to transform a non manifold mesh to a manifold mesh is simply duplicating non manifold vertices and edges. As a consequence the faces incident to the corresponding manifold vertex or edge will get detached.

As the incident faces of non manifold edges and vertices are detached the resulting navigation mesh will contain disconnected cells. There will exist cells that are next to each other but not connected by a portal. As a result some cells will not be reachable to other cells even if they are close. This issue could be fixed by post processing the resulting navigation meshes by introducing portals to connect nearby cells.
The disadvantage of this solution is the increment in the complexity of the mesh as we are replicating non manifold edges and vertices which will result in the mesh having higher total number of edges and vertices.

### 4.3.2 Height Constraint

Another limitation of NEOGEN 3D is applying height constraint only guarantees correct results if there are no intersection among the 3D objects.

![Fig. (i) scene without intersection (ii) scene with intersection](image)

Representation of 3D objects in computers are limited to faces, edges and vertices. Therefore 3D objects are always hollow. As a result it is difficult to differentiate between a hollow and solid object in a computer graphics model. For example, a room can be considered a hollow object, meaning agent can traverse inside the room, whereas a rock is a solid object so agents should not be able to traverse inside them.

If an obstacle object, a solid cube is intersected with the floor only the upper faces of the cube will be taken into account while applying height constraint. As a result if the vertical distance between the upper faces of the cube and floor is greater than the height constraint the portion of the floor intersecting with the cube will not be clipped out.
As shown in the figure above, in case (i), the solid cube violates the height constraint as a result the respective portion of the floor has been clipped out. However in case (ii), the upper face does not violate the height constraint. Therefore the portion of the floor is not clipped out suggesting that portion of the floor inside the solid cube is traversable which should not be the case.
fig. limitation of height constraint on a 3D scene with cube and plane

One possible solution would be to recreate the walkable faces by introducing obstacle edges. This is done to limit access to the portion of walkable region that is inside another solid object from walkable region that is outside other solid object.

fig. addition of obstacle edges on the walkable floor. (blue denotes walkable region, red denotes obstacle region, green line are obstacle edges introduces in the walkable region)
fig. (from left to right) (i) obstacle boundary (ii) resulting navigation mesh (yellow lines are the portals connecting the cells)

As shown in the figure above, introducing obstacle edges results in obstacle boundaries containing them. Therefore, when the navigation mesh is generated the portion of the floor inside the cube is not reachable from the portion outside the floor as we introduced the obstacle edges that are not traversable. One artifact of this solution is that there will exist a walkable region inside the cube. However this region will not be accessible from the outside region.
5 Project Management
5 Project Management

In this chapter we will include the details of the management of the project.

5.1 Planning

Due to the limited time period for the completion of project, it is very important to utilize time as efficiently in order to complete the project in due time and to produce a project of professional quality. In order to do so, some basic guidelines that needs to be followed are,

- Smart planning of the project before execution.
- Subdivision of tasks into clear subtask along with the estimated time necessary to complete each task.
- Having a clear vision of the goal objective at all times.
- Knowing the state of the project at all times and being in control of the project.

5.1.1 Program Duration

The approximate duration of the project is of 4 months, from March 1 to July 1. Project consists of researching a solution, implementation of the solution and testing it.

5.1.2 Tasks

In this section we will divide the project into smaller tasks so that they can be accomplished one by one in a logical manner. In the following we list out all the estimated tasks and its details.

Initial analysis and design

During the initial analysis of the project we will understand the concept and the scope of the project. We research about the various aspects of the project and think about the possible ways we can carry
out the project. The technologies that will be utilized in the project will be decided. The architecture of the solution will then be designed so that all the smaller components of the project can be connected. The initial documentation of the project will be prepared. The course Gestió De Projectes(GEP) will provide a guideline for preparing the initial documentation. The initial design of the project may not be accurate because based on the status of the project and time limit we may include additional features which may increase the scope of the project during execution of the project.

Understand the existing code

Since the project consists of extending the feature of an existing code base it is necessary to understand it first. In this task we go through and experiment with the code to get a better understanding of it. Additional resources includes the original paper of the author, A Framework for Navigation of Autonomous Characters in Complex Virtual Environments which details the algorithm that is implemented in the code. Possible complications for this task are limited resources, only the code base and the original paper, no technical documentation of the code itself and no other sources exist. But regular meetings and discussion with the tutor will help overcome this complication.

Setting up the environment

In this section we will set up the environment of the project. We will gather all the technologies that were decided in the initial design and bring it all together. The necessary code section will be extracted from the code base we are extending (exclude things such as sample examples). Since we will be using multiple PCs for the project (desktop PC at home and laptop while in college) the necessary programs will be installed in all the machines and check that its working correctly. A git repository of the initial empty project will be made so that it is easier to work from multiple machines.
Designing a solution

During this phase, we design theoretical solutions and implement very quick and rough prototypes. The solutions will be discussed with the tutor. We may iterate this task more than once (but not a lot). This subtask may be revisited during the implementation process if we detect minor improvements that can be done.

Program Implementation

This section includes writing implementation of the solutions proposed and sample application to demonstrate the solution. Possible complications may be time taken to learn the technologies that have been decided to be included in the project. This task will be further subdivided into several subtasks. The application can be divided into two parts, 1) the core application and 2) User interface.

- [Core Application] This section implementing the core algorithm and functionality proposed in the solution. The implementation must be tested. The code must seamlessly integrate with initial code that we are extending.
- [User Interface] User interface will be added so that the program is more accessible and easier to use. Designing a beautiful user interface is a time consuming task. Since the user interface is not the main objective of the project, the design of the interface will be more functional and less aesthetic.

Experimentation and Result

It is very important to analyze the performance of the implementation. This provides us with useful information such as its drawback and strength. Utilization of this implementation in real life simulation programs is the end goal. The viability of the solution in such applications will be deduced from the result. The specific scenarios where this implementation is viable will be discussed.
Project management course

GEP course provides knowledge required to carry out a project successfully. All students working on a Bachelor's Degree Final Project are required to take this course. A project report and presentation will be required to be submitted to complete the course.

5.1.3 Time Allocation For Tasks

The estimated time necessary to accomplish each task is provided in the table below,

<table>
<thead>
<tr>
<th>Task</th>
<th>Time (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial analysis and design</td>
<td>20</td>
</tr>
<tr>
<td>Existing code analysis</td>
<td>30</td>
</tr>
<tr>
<td>Setting up the environment</td>
<td>20</td>
</tr>
<tr>
<td>Designing solution</td>
<td>70</td>
</tr>
<tr>
<td>Program Implementation (Core Application)</td>
<td>140</td>
</tr>
<tr>
<td>Program Implementation (User Interface)</td>
<td>90</td>
</tr>
<tr>
<td>Experimentation and results</td>
<td>50</td>
</tr>
<tr>
<td>GEP Course</td>
<td>75</td>
</tr>
<tr>
<td>Project report</td>
<td>30</td>
</tr>
<tr>
<td>Project presentation</td>
<td>10</td>
</tr>
<tr>
<td>Total time</td>
<td>535</td>
</tr>
</tbody>
</table>
5.1.4 Problems and Alternatives

5.2 Development Tools

In this section we explain the tools used throughout the project.

**Microsoft Visual Studio 2017 Community Edition**:

The programming language used for the project is C++ as the NEOGEN 3D framework was implemented in C++. As the operating system we were working the project was on windows, Microsoft Visual Studio 2017 was the evident choice because it is the most full featured c++ IDE for windows operating system. We opted for the community edition as it is free of cost.

Alternative : Jetbrains CLion, Eclipse CDT, etc.

**Irrlicht Rendering Engine**:

For rendering the 3D models and navigation meshes we opted to use irrlicht framework. Irrlicht is an open source C++ 3D game development framework. The strong points were it was easier to get started with and is quite lightweight in comparison to other C++ 3D rendering engines. One disadvantage of irrlicht was lack of community support in comparison to something like Ogre3D. Ogre3D is an alternative 3D rendering engine with enormous community support. This is disadvantage is covered by the fact that irrlicht is much simpler to use.

Alternatives : Ogre3D, Panda3D, etc

**wxWidgets**:

wxWidgets is an open source C++ graphic user interface (GUI) library. User interfaces will enable us to write tools and programs that are more interactive which will result in an increase in productivity of the project. We opted wxWidgets over Qt. Qt may be the most popular GUI framework but is bloated with features that we will never use in our project.
Alternatives: Windows MFC, Qt, Fltk

**wxFormBuilder**: Writing user interface code manually is challenging. wxFormBuilder allows us to rapidly create and design user interfaces and automatically generate the corresponding code segments.

Alternatives: wxGlade

**Blender**: Blender is an open source multimedia software with features encompassing 3D modelling and game engine. Blender will be used in our project for modelling maps to test on NEOGEN Framework. An important feature for blender is writing python scripts to manipulate the meshes as blender provides and api for mesh operations in python.

Alternatives: Autodesk Maya

**Git**: Git is an open source software that provide version control feature. Git allows us to have several versions of the code and simplifies the management of different versions. Source code can also be uploaded to cloud using git. As a result it is easier to work from different computer system and git will automatically sync the project in all systems.

Alternative: SVN
5.3 Budget

This section provides and analysis of budget and expenses of the project.

5.3.1 Hardware cost

Primary hardware tools required to complete the project is a laptop computer.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Cost (euro)</th>
<th>Life (years)</th>
<th>Amortized Cost (euro)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lenovo G50-80</td>
<td>650</td>
<td>4</td>
<td>162.5</td>
</tr>
<tr>
<td>Total</td>
<td>650</td>
<td></td>
<td>162.5</td>
</tr>
</tbody>
</table>

5.3.2 Software cost

Majority of the software utilized in the project are open source. Microsoft visual studio is a commercial software, but the one we are using is community edition which is free to use for non-commercial development. Their zero cost is one of the primary reasons for its utilization of these softwares. Even though many open source projects have a bad reputation for low quality and lack of support and update, the reliability of the software we have included is deduced from its long and excellent track record and its utilization by a large community including commercial companies.

5.3.3 Indirect Cost

The primary indirect cost for the project includes electricity and internet. It is assumed that the electricity resources will be used throughout the project duration whereas internet resource will be used for only 40 percent of the total project duration. Assuming that the
energy consumed by the laptop is 80 Watts, the following costs can be deduced. For transportation Carnet 18 Jove will suffice. The cards provides metro and bus service for duration of 3 months for a cost of 105 Euro. The work is done remotely so there is no necessity for renting a locale to carry the project.

<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
<th>Approximate Cost (Euro)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electricity</td>
<td>0.1255 Euro/kWh</td>
<td>6.6</td>
</tr>
<tr>
<td>Transport</td>
<td>35 Euro/month</td>
<td>140</td>
</tr>
<tr>
<td>Internet</td>
<td>45.0 Euro/month</td>
<td>13.12</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>159.72</td>
</tr>
</tbody>
</table>

5.3.4 Staff Cost

The primary human resources are: 1) Project Manager, 2) Engineer, 3) Developer.

<table>
<thead>
<tr>
<th>Role</th>
<th>Salary(euro/hour)</th>
<th>Time(hour)</th>
<th>Cost(euro)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Manager</td>
<td>50</td>
<td>115</td>
<td>5750.0</td>
</tr>
<tr>
<td>Engineer</td>
<td>40</td>
<td>140</td>
<td>5600.0</td>
</tr>
<tr>
<td>Developer</td>
<td>35</td>
<td>280</td>
<td>9800.0</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>535</td>
<td>24150.0</td>
</tr>
</tbody>
</table>

Based on the responsibility of each role and the scale at which the role affect the completion of the project, the above salaries have been assumed.

5.4.5 Additional Cost

An additional cost of 8 percent will be added to overcome unforeseen circumstances.
### 5.4.5 Total Cost

<table>
<thead>
<tr>
<th>Resource</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>162.5</td>
</tr>
<tr>
<td>Software</td>
<td>0.0</td>
</tr>
<tr>
<td>Indirect cost</td>
<td>159.72</td>
</tr>
<tr>
<td>Staff</td>
<td>24150.0</td>
</tr>
<tr>
<td>Subtotal</td>
<td>24472.2</td>
</tr>
<tr>
<td>Additional Cost (8%)</td>
<td>1957.7</td>
</tr>
<tr>
<td>Total</td>
<td>26429.9</td>
</tr>
</tbody>
</table>

### 5.4 Sustainability Analysis

#### 5.4.1 Economic

The major percent of our total project cost is human resources. There will not be any upgrade and updates on the hardware and software resources during the project. Hardware breakdown will be easily covered by the additional cost that we added to the project.

The techniques provided in other state of art projects are primarily utilized in video games. People working in these industries have two options,

- To understand the techniques and knowledge provided by the state of art projects and re-implement it. The cost for this option is low to almost none.
- Utilize other piece of software (Game engines) that provides the techniques mentioned in the projects in an easy to use manner. The cost for this option is pretty high and depends on the quality of
the software. Some popular softwares are Unreal Engine, Unity3D, etc.

Our project provides the details of the technique and also provides a base code implementation of the techniques which other developers can include in their projects. Therefore our project provides both options to other developers.

5.4.2 Environmental

The project does not imply any major effects to the environment. The environmental resources utilized in the project are minimum and are only indirectly affected.

The impact will be further minimized through the process of recycling and reusing. The primary tool of the project, the laptop will be sent to the recycling center when it bypasses its useful lifetime.

This project along with other state of art projects which focuses on AI and simulation techniques primarily focus on affecting and influencing the software side, especially video games. These type of projects provide much less impact on the environment than projects that aim to provide solutions to some physical problems. For example, projects involving robotics and machinery that have the task to solve a problem that exists in the real world will have a much greater impact. Therefore, our project does not improve the environmental impact when compared to other similar projects because this is not our direct goal.

5.4.3 Social

The project will have a reasonable impact on the social dimension. The people primarily affected are video games enthusiasts including
developers as well as players. Other researchers specializing in the fields of AI and simulation are also affected.

Even if the project does not see any direct usage, it is absolutely necessary for this project to exist. For problems that requires such experimental and situational solutions, every solution will provide something for the future. The video game industry has seen some of the most impressive growth in recent years and will continue to grow in the future. This project will directly or indirectly affect in that growth.
6 Conclusion


6 Conclusion

To summarize our objective of the project was to extend NEOGEN 3D framework to permit advanced and complex pathfinding and detecting limitation of NEOGEN 3D as well and proposed solution for the limitations. We introduced the concept of off mesh links which allowed permitting paths even when the walkable region of the mesh were not connected physically. As a result permitting complex paths including jumps, stairs, etc. In addition storing of information on nodes and edges of the graph permitted different behaviour of agent as it traverses the computed path. As for limitations of NEOGEN 3D, firstly we have non manifold meshes which is resolved by transforming them into manifold meshes. Secondly, incorrect height constraint given intersection among the objects of the 3D scene.

There is room for further studies over this topic. Such as implementing the concepts to a popular platform such as Unity3D. And also providing much better solution for the limitations of NEOGEN 3D.

As a consequence of this project, I myself learnt to use valuable tools such as blender and python for 3D modelling and mesh manipulation. I also improved in user interface programming. And most importantly, I gained knowledge on how video games and simulation programs utilize navigation meshes to create advanced path planning systems. To conclude, I personally feel that this project has been a very good experience and am content to have worked on it.
References


https://upcommons.upc.edu/bitstream/handle/2117/106282/TROM1de1.pdf


https://pdfs.semanticscholar.org/2101/e4a5d9a5822f769eb83be10d3bc894dd8bef.pdf

Mat Buckland. Programming game AI by example. Cp5 The secret life of graphs, Cp8 Practical Path planning.
http://index-of.co.uk/Game-Development/Programming/Programming%20Game%20AI%20by%20Example.pdf

Amits A* pages
http://theory.stanford.edu/~amitp/GameProgramming/

Recast Navigation
https://github.com/memononen/recastnavigation/

Electricity price Spain
https://tarifaluzhora.es/

TMB Prices

Unity3D Game Engine
https://unity.com/

Unreal Game Engine

Git
https://github.com/
https://git-scm.com/

Irrlicht
http://irrlicht.sourceforge.net/

Wxwidgets
https://www.wxwidgets.org/

Blender
https://www.blender.org/