Procedural Level Generation and Dynamic Path-finding A Combined Approach

Neil Notman - 40124066
Supervisor: Dr Benjamin Kenwright
Second Marker: Dr Neil Urquhart

Submitted in partial fulfilment of
the requirements of Edinburgh Napier University
for the Degree of
BSc Games Development (Hons)

School of Computing

March 17, 2016

Authorship Declaration

I, (Neil Notman), confirm that this dissertation and the work presented in it are my own achievement.

Where I have consulted the published work of others this is always clearly attributed;

Where I have quoted from the work of others the source is always given. With the exception of such quotations this dissertation is entirely my own work;

I have acknowledged all main sources of help;

If my research follows on from previous work or is part of a larger collaborative research project I have made clear exactly what was done by others and what I have contributed myself;

I have read and understand the penalties associated with Academic Misconduct.

I also confirm that I have obtained informed consent from all people I have involved in the work in this dissertation following the School's ethical guidelines.

Signed:

Date:

Matriculation no:

Data Protection Declaration

Under the 1998 Data Protection Act, The University cannot disclose your grade to an unauthorised person. However, other students benefit from studying dissertations that have their grades attached.

Please sign your name below one of the options below to state your preference.

The University may make this dissertation, with indicative grade, available to others.

The University may make this dissertation available to others, but the grade may not be disclosed.

The University may not make this dissertation available to others.

Abstract

As games evolve and expand with the times there is an increased demand placed upon the hardware of games consoles and personal computers in terms of storage space and memory usage. This project will analyse the approaches currently available and will examine the viability of developing a combined approach to level generation and path finding with the aim of producing a solution that will compete with current approaches in terms of the quality of the terrain generated tested through the results given by the path finding and the distances between nodes that will be placed in sane (flat, reachable) locations on the terrain this will also allow for testing the quality of the path finding approach by testing against different complexities of terrain and existing industry techniques.

Contents

List of Tables

List of Figures

Acknowledgements

1 introduction

This chapter will outline the basic topics relating to the project in terms of both level generation and path-finding then some of the optimisation techniques that may be used in the project will be discussed. Finally the main goals for the project will be analysed and the layout of the report from this point forward will be given.

1.1 Procedural Level Generation

Procedural level generation is the process of creating a level through an algorithm contained within a computer program instead of using software to create this content then loading it into the program this approach was mainly utilised in the early day of games development where storage space was a major concern.
The first game to contain procedurally generated levels was Richard Garriott's game Akalabeth: World of Doom which was published by California Pacific Computer Company in 1980 [Wikipedia, 2015].

This project is going to utilise procedural level generation to create a average sized game world by creating a height-map which is a grey-scale then reading the data from a height-map and using this data to transform a plane to create a large scale landscape in chunks.

1.2 Path-Finding

Path-Finding is an area of Artificial Intelligence that allows for definition what parts of a given level can be moved to by computer controlled players this is done by creating a set of nodes on walkable areas then linking these nodes together to form a path there are various algorithms that accomplish this task. A time-line of path-finding developments is given at figure: 4.
The project will attempt to use path-finding on the generated level in small chunks to allow for faster processing and access to more optimisation techniques.

1.3 Optimisation Techniques

There are many factors that can affect the run-time performance of computer programs and a similar amount of approaches that can be utilised to improve this some of the most prominent techniques in this area include use of multi-threading and parallelism frameworks such as OpenMP for use in single computer based solutions and MPI which allows for parallel implementation over a network.

There are other options that differ from a parallel approach that could be utilised these can include memory alignment of instructions or a graphics processing unit (GPU) based implementation.

1.4 Aims and Objectives

The main aim of this project is to examine and test the viability of combining level generation and path finding algorithms to create a double ended testing mechanism that will be achieved through the creation of an application using the chosen solution to allow a user to create various sizes of level. The application will then build the level in sections and perform path finding on each section and assess the quality of both the level and the path taking into account the complexity of a given level such as obstacles slopes and caves and then assessing how easy it is to build a path round that level this will give a score to both the technique of level generation and the path-finding technique used.

The objectives for this project is to find a method of combining level generation and path finding and gathering data from both the chosen solution and current practices to make an unbiased comparison between them and establish the viability of a double ended testing mechanism.

1.5 Report Layout

The report goes on to a review of literature sources associated with the project then technical information on the project will be supplied next there will be an analysis into the methodology used in implementation and finally the testing and results will be given alongside the conclusion of the project which will be based on the results of testing.

2 Literature-Review

2.1 Background and inspiration

The inspiration for this project was based around pushing performance and analysing both level generation and path-finding approaches to find a common area where these methods can be combined to improve run-time performance and to allow testing of the quality of both the terrain and path that is built in terms of complexity against the amount of nodes required to build a logical path and any deviations that are made to the path due to constraints posed by the terrain.

For a background of various path-finding techniques please refer to paper[Husdal, 2015]

2.2 Procedural Level generation Techniques

There are a variety of ways to generate either 2d or 3d geometry within a program this section will discuss the documented methods and compare them in terms of usability and performance.

The paper [Hendrikx et al., 2013] examines the different layers of a game that can be procedurally generated this includes a section on game space for levels or maps and defines these both abstract or concrete methods for generation of both indoor and outdoor environments as well the paper gives a taxonomy of commonly used procedural content generation techniques.

2.2.1 The Diamond-Square algorithm

This method of level generation was proposed by [Fournier et al., 1982] is based off fractal subdivision to generate randomised terrain based on two steps which are split into the diamond step and the square step hence the algorithms name. The diamond step of this algorithm uses the edge points of a square to generate a random value in the midpoint then the square step of this algorithm performs the same function however uses the edge points of the diamond made previously which gives a square as the result. The algorithm is recursive meaning that it performs multiple passes through each step with the generated surface gaining detail every pass however the increase in detail is due to an increase in geometry each pass this means the algorithm can be memory intensive due to the size of the array used to store the height values being a power of two + 1 this means for eight passes through the algorithm the array would need up to 256KB memory if storing floating point values [Miller, 1986].

2.2.2 The marching cubes algorithm

The marching cubes algorithm is another method that can be used to generate terrain it was first proposed in 1987 by [Lorensen and Cline, 1987]

2.2.3 Height-map generation and Rendering

A height map is a grey scale image that makes use of depth to create a section of terrain with areas where the terrain is raised being shown as brighter and lower areas shown as dark for most height maps digital noise is used to produce a suitable terrain for an example of the output of a height map please refer to figure 2.

2.2.4 Perlin Noise

To create the image 2 Perlin Noise was used this is a gradient based noise developed by Ken Perlin in 1983 [Perlin, 1985] to create procedural textures using the algorithm outlined below.

- Get an input point of the image
- Loop through the neighbours of this point
- Generate a pseudo-random gradient vector.

  - perform the following calculation

$$G \cdot (P - Q) \tag{1}$$

Figure 1: G= gradient vector P = input point Q = Neighbouring point

  This equation will give the value of P where G is equal to 0 at point Q.

- Finally you Interpolate between the the points down to your point, using an S-shaped cross-fade curve e.g($(eg : 3t^2 - 2t^3)$) this will give the weighting applied in each dimension.This step will require computing of the curve n times, followed by 2n-1 linear interpolations to get the final result.

2.3 Path-finding Techniques

This section will address some of the key path finding techniques that would address the problem posed within the project an analysis of past work done in this area will be given with the aim to analyse how this can be utilised and expanded within the context of the project.

2.3.1 Dijkstra's algorithm

The algorithm for path finding presented by Edsger Dijkstra in 1959 [Cormen et al.,] allows searching of a weighted graph to determine the shortest route through a set of nodes by working out the shortest path from one node to every other connected node in the graph this is known as a uniform cost search. The nodes can be used to represent space within either a two dimensional grid or a point in three dimensions. This method can be used within the project as the cost of travelling between nodes can be calculated as the euclidean distance between the nodes this will allow us to remain with vector calculations which would benefit a GPU based implementation. If we look at figure 4 it is shown there have been a number of developments to this method of path-finding the main adjustments that are beneficial to the project will be discussed below.

2.3.1.1 Developments to graph based path finding

There have been a multitude of developments to graph based path finding techniques these include the techniques mentioned in 4 the main developments and their benefits in the context of this project will be discussed.

The first development to graph based techniques that is interesting is the article by [Goodchild, 1977] this was where the traditional orthogonal approach to building paths was modified due to the fact that errors were produced with straight line and smoothly curved paths the solution was to look at using both the orthogonal and diagonal steps to reduce the errors produced such as deviation from the path and this also prevents the path from becoming too long.

Another development of interest is the creation of a spread based algorithm for path-finding that was first proposed by [Califano, 2000] the origin of the splash (structural pattern localization analysis by sequential histograms) algorithm was to identify patterns in amino acids and other uses within the field of computational biology relying on sparse pattern matching techniques however the splash is shown in [Califano, 2000] to be both memory and computationally efficient when compared with other path finding methods that utilise sparse pattern matching this is of value as it allows for a path that has patterns will be processed faster and using the splash algorithm could be beneficial due to the parallelizable nature of that this algorithm uses to search through data.

2.3.2 Path finding in games

Path finding is an area of artificial intelligence widely used in games to allow computer controlled characters to walk on the geometry that makes up a game world or level the paper by [Graham et al., 2003] describes the method for path finding on geometry by first building a navigational mesh which describes the parts of a given world or level can be traversed or using a node based waypoint system which is traditionally based on the visibility of one node to the next which allows for traversal of the geometry there is also a breakdown of path finding techniques into directed and undirected methods with directed methods such as the A* algorithm which combines the cost based searching of nodes or a navigational mesh with a heuristic search to the goal this allows for greater efficiency over Dijkstra's algorithm.

2.3.3 The A* algorithm

The A* or A star path-finding algorithm was first proposed in 1968 the paper [Hart et al., 1968] this algorithm for path-finding uses a heuristic based search to find the shortest route for a given weighted graph there are many benefits to using this algorithm these include storing previously any visited nodes that do not lead to the goal this helps to prevent backtracking and eliminate infeasible solutions leading to a reduction in the search space thus an increase in runtime performance although this comes with an increased memory overhead as a result of using two lists over one list to store the path.

2.3.4 Analysing the quality of path-finding techniques

The niche of this project is that there has been no previous attempt to quantify the quality of procedural level development techniques with regards to ease of path-finding as a factor for the assessment of quality and assess path-finding with regards to the complexity of a terrain that is generated this will be achieved through the development of a deep learning AI system that can learn from it's own examples other systems that utilise this method of learning such as the system developed by google(tm) in their deep mind department called alphago which was used to beat the current champion of the game GO[Silver et al., 2016] this was made possible by training the system with the moves of experts then allowing it to learn from playing against itself using two neural networks one to determine promising moves and the other searches ahead of the promising moves found this reduces the search space allowing this processing to take place as a brute force approach would fail due to the magnitude of moves possible in this project the AI system could be utilised to fine tune the metric for level complexity by learning how to score an environment and gradually increasing the consistency of results gathered.

2.3.5 Previous work in this field

There has been a previous honours project based on performance of GPU based path-finding. The thesis that was authored by [McMillan, 2014] contains work on a variety of path-finding techniques including Dijkstra, A* and diffusion methods with both a sequential and parallelised implementation the optimised versions of these path-finding approaches which may be utilised in cases of performance bottleneck as this project will not be specifically related to performance but rather the production of an elegant constrained path and suitable terrain.

There was also a project that proposed a metric for picking good procedurally generated terrain this was worked on by .... and looked at different methods for the creation of terrain

2.4 Optimisation techniques

To optimise the project a wide range of factors and approaches are being considered the purpose and uses of these will be briefly discussed below along with a discussion on how these methods could benefit this project.

2.4.1 Parallelism and parallel programming approaches

The first approach to address is that of how to get the most performance out of the development platform (PC) one of the main ways to optimise for this platform is the use of parallel programming this will be implemented through the OpenMP API [Dagum and Enon, 1998]which is built into Visual Studio this will allow for multi-threading of any code running on the central processing unit (CPU) through the use of pre-processor directives to specify sections of code to incorporate parallelism this approach should give a performance increase by utilising the multiple cores in the central processing unit over a sequential implementation that uses a single core.

Other parallel frameworks have been considered however are not at this point set to be implemented these are the MPI (Message Passing Interface)[Snir, 1998] which allows for distributed parallelism over a network this would be extremely beneficial to performance as the PC that the program is running on would have more resources available however this approach can have issues mainly due to the communication being network based meaning that bandwidth can affect the performance of the program to produce a performance decrease due to these factors it is unlikely that the project will utilise this technique.

2.4.2 Graphics Processing Unit (GPGPU)

GPGPU(General Purpose Graphics Processing Unit) programming allows the user to run non graphics based code on the GPU this is beneficial due to the amount of processing cores that the GPU contains over the central processing unit however there area number of drawbacks to this approach and it can only be used within certain problem area's for example GPGPU programming is a good fit to math based problems over large data sets in comparison to problems that branch through sections of code this is due to the way data is processed on the GPU this is done by streaming the data through the GPU cores in groups known as either wavefronts or wraps which are groups of threads within the GPU and the terminology for these differs depending on the technology used instead of the normal fetch execution cycle of the CPU where instructions and resources are fetched from memory before they are executed or loaded. To utilise the GPU for processing we must send the data from the CPU to the GPU then perform our operations and finally transfer the answer from the GPU back to the CPU this is one of the main drawbacks to GPGPU programming as the performance is determined not only by the hardware but also by the speed of data transfer between the CPU and GPU and the size that this data takes up in memory.

The use of GPGPU techniques for parallel programming were only recently applied to AI because of the branching nature of the problems in this area it has always been considered that the AI system discussed in 2.3.4 used a distributed system to utilise the hardware resources of multiple machines to carry out parts of the processing this is what allowed the massive amount of processing power required to allow the alphago system to process the best moves at every point of a match.

2.5 Commercial packages

There exist a number of commercial applications that fall within the context of this project most of these are contained within games engines which contain ways to generate levels and to build paths and evolutionary systems which are generally used to find the best solution can be found within lots of packages one of the most noteable being journey planning software such as that contained within Google Maps.

3 Technical Information

4 Implementation Methodology

This section will analyse how the different areas of the implementation are approached and seek to explain how these approaches work at a technical level.

For a basic outline of the expected execution of the project please refer to figure 6

4.1 Level Generation

To generate the three dimensional terrain used in the project a height-map was generated through the use of the Libnoise external library which is discussed below.

4.1.1 Libnoise external library

This external library generates different types of noise as shown by figure 2 this is achieved use of various modules that can produce different types of noise including perlin which is the basis for our output shown in 3 another feature of this library is that it allows the noise generated to be written out to various file types for the height-map in the project a Bitmap (BMP) graphical file type was produced which is then loaded into the project and stored in a data structure for manipulation and rendering.

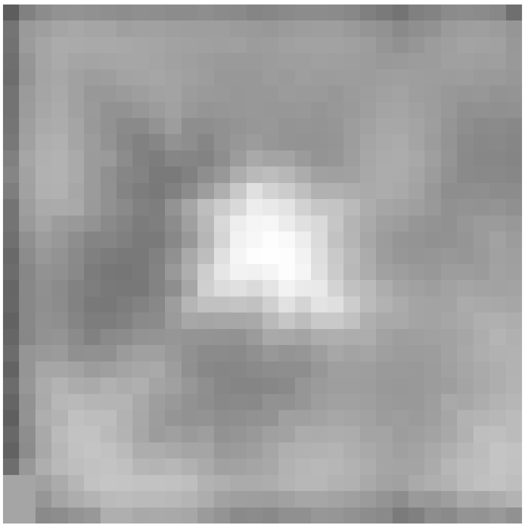

Figure 2: An image showing of one of the height-maps generated by noise

When this image is rendered to the screen it gives the following output:

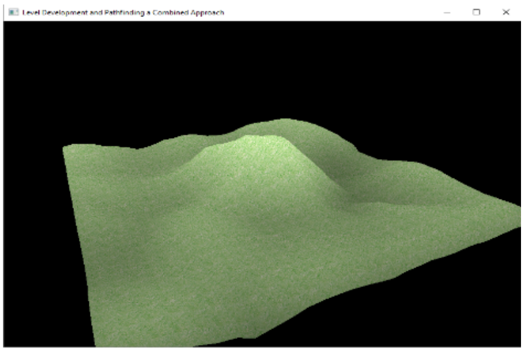

Figure 3: the output from rendering the heightmap

4.2 Additional Information / Knowledge Required

To implement the project a solid knowledge of a graphical programming language such as openGL or DirectX is required also perfomance based GPGPU programming API's such as OpenCL or Nvidia's CUDA could be useful for optimising the project.

5 Testing and results

Test Id	Test Name	Test Rationale	results	Retesting result (if modified)
1	Height-map generation and loading	This is the first chosen level generation technique used within the project	pass	n/a
2	Storage and rendering of height-map	This is necessary to test to allow continued use of the height-map technique	pass	n/a
3	Height-map performance	This is needed to establish the viability of the height-map method for the project	pass	n/a
4	Other level generation techniques	This could be required if the height-map approach has issues or is not viable due to low performance	pass	may be revisited as additional functionality
5	node placement and rendering	This will allow for visualisation of the data for testing and the nodes are required for the path finder	ongoing	n/a

6 Conclusions

7 Bibliography

References

[lev,] Dungeon delving: A short introduction to procedural level generation. https://www.youtube.com/watch?v=VGSHsh83_ns. Accessed: 2015-09-15. 26

[Califano, 2000] Califano, A. (2000). Splash: structural pattern localization analysis by sequential histograms. *Bioinformatics*, 16(4):341–357. 13, 14, 28

[Cormen et al.,] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. Dijkstra's algorithm. pages 595–599. 13, 28

[Dagum and Enon, 1998] Dagum, L. and Enon, R. (1998). Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55. 15

[Eastman, 1989] Eastman, J. R. (1989). Pushbroom algorithms for calculating distances in raster grids. In *Proc. Autocarto*, volume 9, pages 288–297. 28

[Fournier et al., 1982] Fournier, A., Fussell, D., and Carpenter, L. (1982). Computer rendering of stochastic models. *Communications of the ACM*, 25(6):371–384. 11

[Goodchild, 1977] Goodchild, M. (1977). An evaluation of lattice solutions to the problem of corridor location. *Environment and Planning A*, 9(7):727–738. 13, 28

[Graham et al., 2003] Graham, R., McCabe, H., and Sheridan, S. (2003). Pathfinding in computer games. *ITB Journal*, 8:57–81. 14

[Hart et al., 1968] Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107. 14

[Hendrikx et al., 2013] Hendrikx, M., Meijer, S., Van Der Velden, J., and Iosup, A. (2013). Procedural content generation for games: A survey. *ACM Trans. Multimedia Comput. Commun. Appl.*, 9(1):1:1–1:22. 11

[Huber and Church, 1985] Huber, D. L. and Church, R. L. (1985). Transmission corridor location modeling. *Journal of Transportation Engineering*, 111(2):114–130. 28

[Husdal, 2015] Husdal, J. (2015). Corridor analysis – a timeline of evolutionary development. *Unpublished coursework. University of Utah, USA.* 11, 28

[Lombard and Church, 1993] Lombard, K. and Church, R. (1993). The gateway shortest path problem: generating alternative routes for a corridor location problem. *Geographical systems*, 1(1):25–45. 28

[Lorensen and Cline, 1987] Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM. 12

[McIlhagga, 1997] McIlhagga, D. (1997). Optimal path delineation to multiple targets incorporating fixed cost distance. *Unpublished Thesis, Carleton University, Ottawa, ON.* 28

[McMillan, 2014] McMillan, C. ((2014)). Comparison of pathfinding algorithms using the gpgpu. *BEng (Hons) Games Development Dissertation). Edinburgh Napier University (Hart, E., Urquhart, N.* 15, 26

[Miller, 1986] Miller, G. S. P. (1986). The definition and rendering of terrain maps. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pages 39–48, New York, NY, USA. ACM. 11

[Perlin, 1985] Perlin, K. (1985). An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296. 12

[Shaker et al., 2015] Shaker, N., Togelius, J., and Nelson, M. J. (2015). *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer. 26

[Silver et al., 2016] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489. 15

[Snir, 1998] Snir, M. (1998). *MPI–the Complete Reference: The MPI core*, volume 1. MIT press. 16

[Sturtevant and Buro, 2005] Sturtevant, N. and Buro, M. (2005). Partial pathfinding using map abstraction and refinement. *AAAI*, 5:1392–1397. 26

[Wikipedia, 2015] Wikipedia (2015). Akalabeth: World of doom — wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Akalabeth:_World_of_Doom&oldid=670318824. [Online; accessed 13-October-2015]. 9

Appendices

A Project Overview

Keywords— Level-Generation,Path-Finding,Algorithm,Optimisation

A.1 Overview of Project Content and Milestones

This project will include research into current practices used in industry for path-finding and level generation with the emphasis being performance based compared against realism. The main milestone for this stage will be coming across a method to integrate both level generation and path-finding into an algorithm which will be carried forward to the development stage.

Development of an application which will utilise both level generation and path-finding either by the modification and optimisation of existing approaches or the creation of a new algorithm. The application will allow the user to create various sizes of level and the application will then give a visual representation of both the level and path that has been built.

The application will then be optimised and tested against some of the methods researched with the aim to show that a combined approach may achieve better results in a similar or faster run time with the paths created being checked for mistakes or inconsistencies based on various sanity checks (no path nodes on areas that should not be traversable such as extreme slopes or dips in the level).

Creation of a report that contains all project information and related work with a detailed analysis and testing of the approach used to provide an evaluation of the solution using performance figures to accurately examine the viability of this approach in comparison to the current industry standards and draw a conclusion over the project as a whole.

For a more detailed look at the time-line for this project please refer to Figure 5

A.2 The Main Deliverables

- A application that will allow the user to create various sizes of level and will give a visual output of the level and path built with a focus on the optimisation techniques used to integrate level generation and path finding.

- A detailed report containing an analysis on the approach taken and a comparison between the solution and other methods used currently using figures from both to evaluate the effectiveness and viability of the approach used.

- Test logs of the finished application and an industry standard practice to allow for an unbiased comparison to be made based on results obtained from these tests when documenting the project.

A.3 Target audience for the deliverables

The deliverables of this project may be of interest to people involved with both level generation and path finding in the games development industry. Researchers involved in this field may find the report and comparison of techniques used to be of interest. Other people may find this project helpful with regards to optimisation techniques. The main aim of the project is to show the performance of a combined approach to level generation and path-finding and an evaluation of the quality of paths built.

A.4 The work to be undertaken

- Research into current industry practices for level generation and path finding to allow for visualization of a solution to combine these algorithms.

- Development of the application and chosen approach with a focus on performance while maintaining a level of realism.

- Optimisation of the developed solution and gathering of test data from the solution and an industry approach.

- Creation of a report detailing the approach taken to implementing the project and providing an evaluation between the solution compared to industry practices.

There is a diagram showing expected steps of execution within the application shown at Figure 6

A.5 Knowledge and skills required

To be able to complete the project to a high standard I will first need to research the field's of both level generation and path finding as a solid knowledge will be needed to create a combined approach. In terms of development further research into optimisation techniques for C++ in Visual Studio will allow for a more polished and better performing solution also as a level of realism will be maintained experience with graphic programming would also benefit the project.

A.6 Information sources that provide a context for the project

The video available at [lev,] gives a few very basic examples of levels that can be generated and also details some methods to define areas within a level(slopes,walls,and others) this definition of areas within a level could be used for testing the path finding within the project and may give a basic concept on how to generate content within the level produced in relation with the project.

The paper[Sturtevant and Buro, 2005] analyses various path-finding algorithms however the purpose of their project was to look at an alternative approach to the A* path finding algorithm. Use of concepts such as map abstraction could be of interest to this project as it would allow a way to break the level into chunks which can then be processed individually by a path finding algorithm.

The textbook for level generation [Shaker et al., 2015] Provides extensive research into procedurally generated content in games from grid based levels to full landscapes there are multiple approaches detailed within the textbook that can provide a basis to generate a level within the project.

Previous honours projects to note

This paper[McMillan, 2014] analyses various path finding algorithms and discusses at optimisation options this will be the inspiration for optimising the project and will give a wide variety of path finding algorithms to research going forward.

For a timeline of research into path finding algorithms please refer to Figure 4

A.7 The importance of the project

This project is important as it will seek to integrate level generation and path finding which could allow for faster loading times in games while reducing the storage space taken by a game. When comparing the difference between developed solution and current industry practices - if the solution proves to be better in terms of performance and the quality of the path built is of a suitable standard for the complexity of the level that is generated.

A.8 The key challenges to be overcome

- Research of a variety of level generation and path-finding techniques to gain knowledge and propose a viable solution for development.

- Development of an application that will use the envisioned solution and give a visual output to the user of both level and path built.

- Research into a way of determining the quality of path finding and level generation techniques.

- Writing of a detailed report that will clearly explain the solution chosen and give a view into past work done within this field then the report will provide an unbiased comparison between the approach and current industry practices to evaluate the project in terms of the quality the path built when compared to the complexity of the level generated.

Creator	Method	Year
Dijkstra [Cormen et al.,]	Graph searching	1959
Goodchild [Goodchild, 1977]	Orthagonal and diagonal movement	1977
Huber and Church [Huber and Church, 1985]	analysis of neighboring cells	1985
Eastman [Eastman, 1989]	Pushbroom procedure	1989
Lombard and Church [Lombard and Church, 1993]	GSP(Gateway-shortest-path)	1993
Mcilhagga [McIlhagga, 1997]	Fixed cost distance	1997
Andrea Califano [Califano, 2000]	Splash algorithm	2000

Figure 4: A time-line of path finding algorithms [Husdal, 2015]

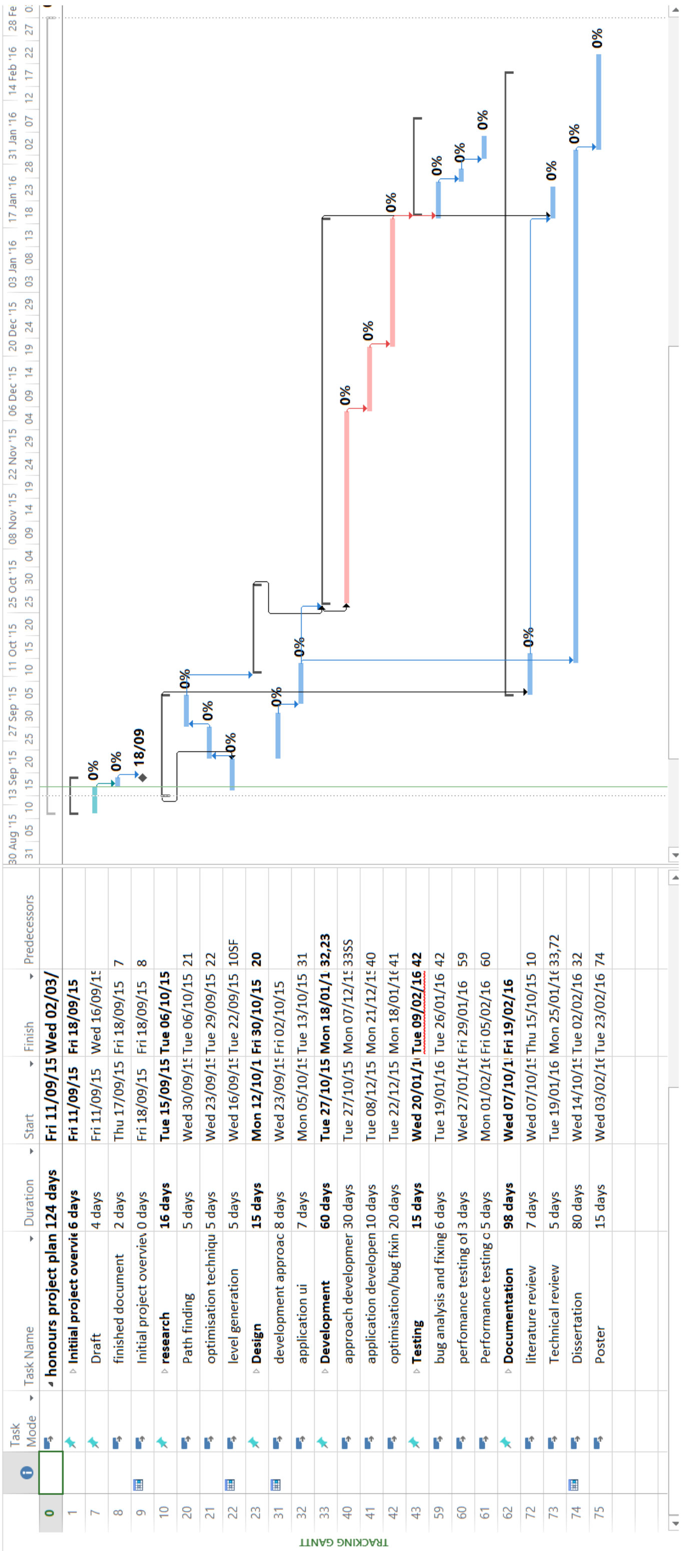

Figure 5: An image of the project gantt chart

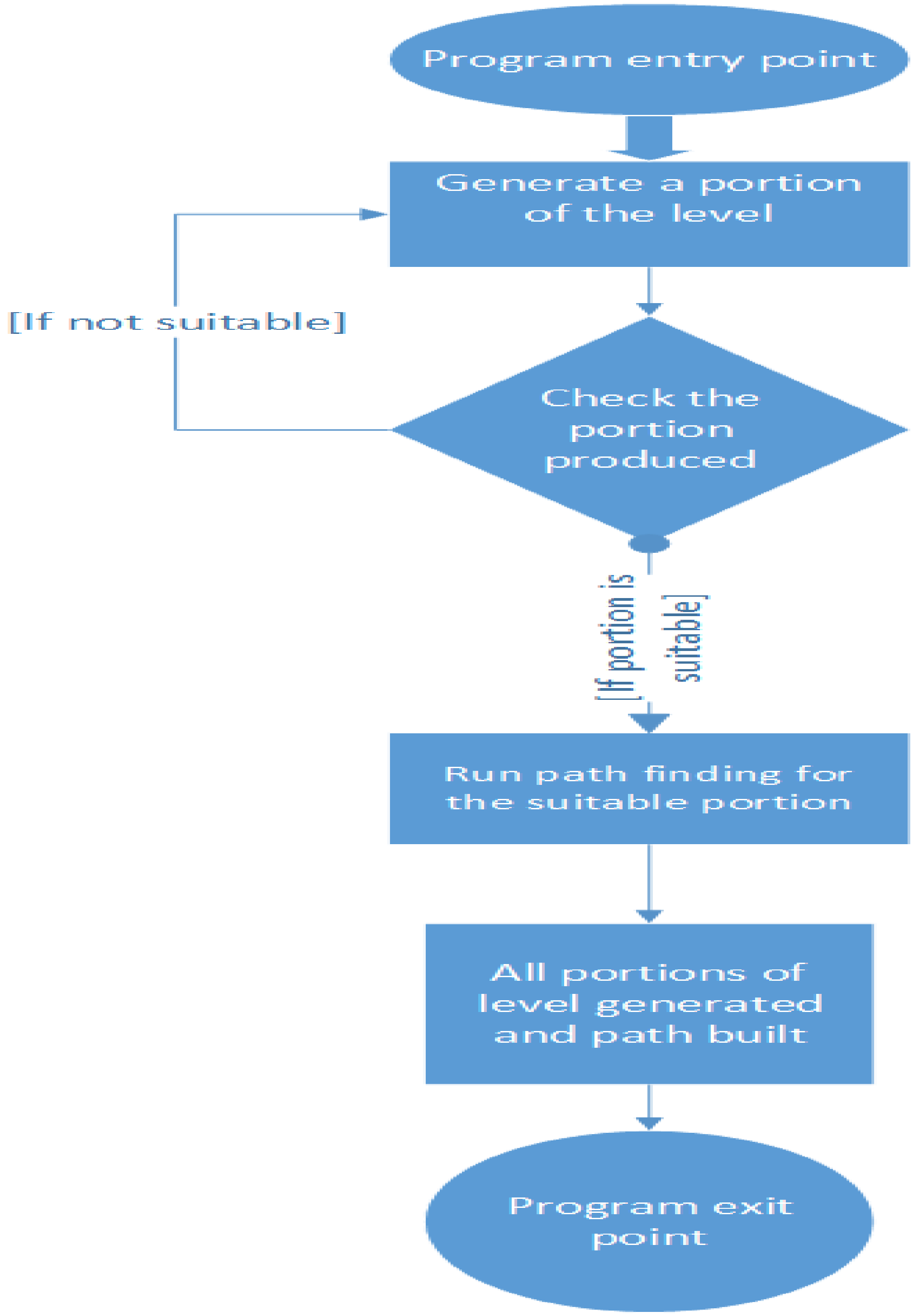

Figure 6: This shows how the program will execute

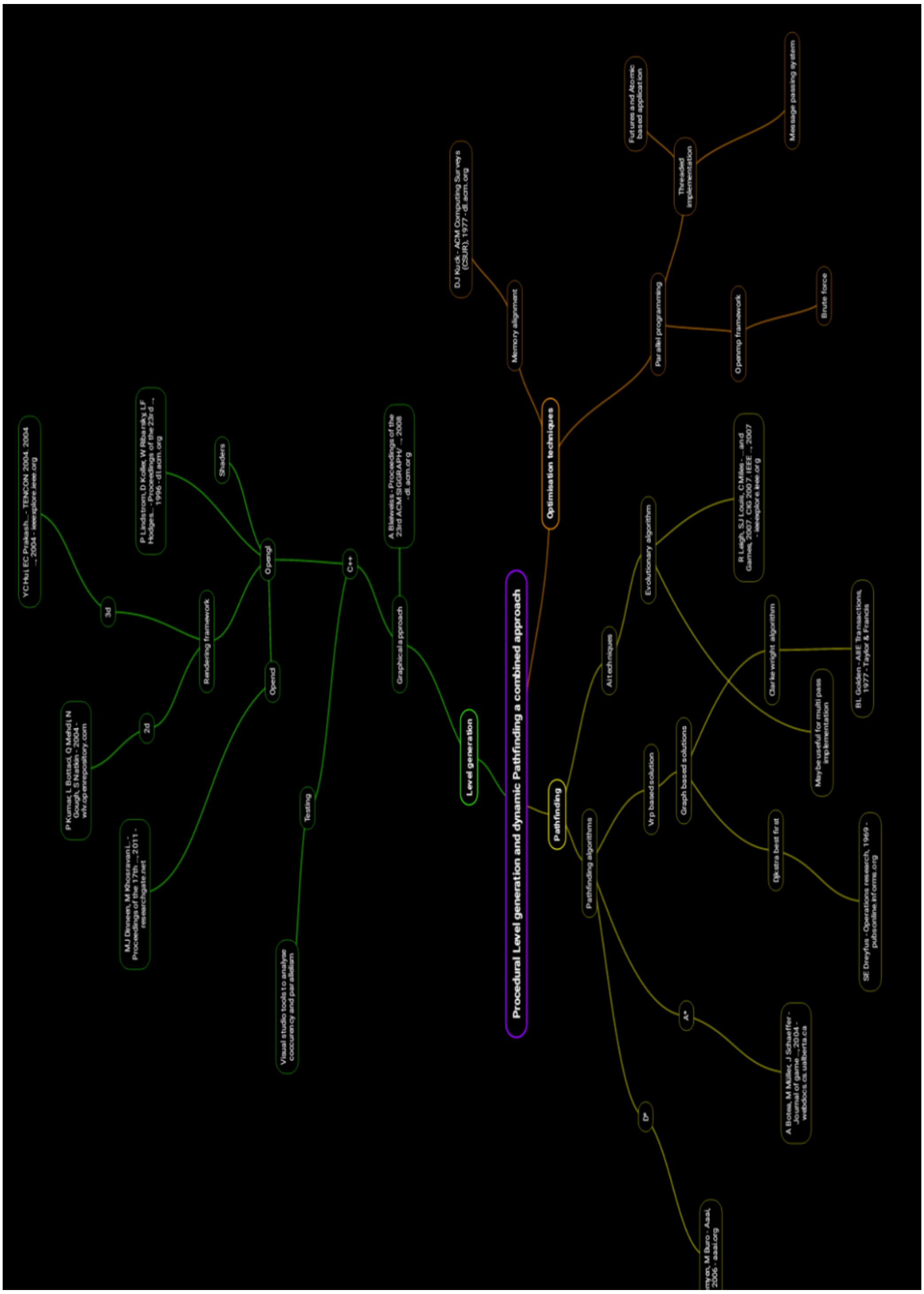

Figure 7: This is a mind map covering the scope of the project

B Diary Sheets and project management

B.1 Week 2 and before

This was the point where a supervisor for the project was established and formal meetings were begun from week 3

B.2 Meeting 1 22/9/15 Week 3

Objectives
update literature/background
Polish IPO for submission
Block diagram

Progress
First draft of IPO and Project plan
First meeting

Comments
On Track/looking good

B.3 Meeting 2 29/9/15 Week 4

Objectives
start drafting literature review
Mind map
Add keywords
setup version control with basic renderer
Comparison grid

Progress
IPO handed in
Collecting papers for research
Setup blog

Comments
On Track

B.4 Meeting 3 6/10/15 Week 5

Objectives
Thesis template

Add main bib file
Screen-shots from implementation
Add mind map
Write introduction
Comparison grid

Progress
Skeleton Framework
Continued Collecting papers for research
Added keywords to IPO

Comments
Starting to slow down

B.5 Meeting 4 13/10/15 Week 6

Objectives
Screenshots
Add mind map and comparison grid to thesis
Fix and complete level generation implementation
Draft lit review
Write introduction
Blog updates
Print and bring thesis to next meeting

Progress
Height-map generation code implemented
Continued Collecting papers for research
Drafted thesis and started introduction

Comments
Need to improve time management
On track-good work

B.6 Meeting 5/Group Thesis Review 20/10/15 Week 7

Feedback
Report looking good so far minor changes to be made to already completed sections.

Progress
Mind map finished and added to thesis.
Loading of height-map code fixed.

B.7 Meeting 6 27/10/15 Week 8

Meeting was missed however general discussion was had with supervisor during the week to state goals for this week and record progress informally.

B.8 Meeting 6 3/11/15 Week 9

Objectives
Update thesis
Look at commercial packages for comparison
Implementation work (Test cases)
Print and bring thesis to next meeting

Progress
Numerical experimentation for level generation
Started writing implementation methodology
almost finished lit review(Draft version)

Comments
Coursework week(busy)
Second marker meeting next week

B.9 Meeting 7 10/11/15 Week 10

Objectives
Update thesis
Fix rendering
Implementation work (Test cases)
Blog/Website updates Print and bring thesis to next meeting

Progress
Started to fix development issues(rendering)
Thesis Work
Re-done blog/website (more structured)

Comments
Review thesis progress next week
Week 10-Second marker meeting organisation

C Second Formal Review Output

D Appendix 4 and following