Evaluation of Procedaully Generatated Escort Mission Maps

Adam Blance - 40161070

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

School of Computing

April 8, 2018

Authorship Declaration

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

Contents

List of Tables

List of Figures

Acknowledgements

Chapter 1

Introduction

From the beginning of the project the objective has been to develop a mathematical equation to evaluate escort mission maps. Then to procedurally generate maps that can successfully follow the evaluation technique created. The dissertation produced underneath documents the work that was involved in achieving this.

1.0.1 Motivation

With video games being one of the largest markets in the world right now, gaming companies must be developing and releasing their games as quickly as possible. If there is any conceivable way for them speed up releasing their games, they should take these approaches.

One of the biggest genres of games currently is team based shooters. With one of the larger games in the genre being Overwatch with currently over 35 million players worldwide. . This genre of game has a very different development cycle to other games relying on continual updates adding new maps and characters to keep players interested and continually playing. To release new maps and characters on a regular basis can be difficult especially if the game has a large competitive scene because the new feature must be fully tested before released.

With testing new features in games becoming the most time consuming part of the development cycle, companies must look into other ways to streamline the testing of games. In this project one of the possible ways to decrease the time taken to test maps is explored. If companies can use a mathematical formula to test the fairness of maps, rather than the more conventional user based testing which can take up to several weeks, would allow for companies to produce more content in a quicker time frame, generating a greater profit and allowing for better content to be produced in the future.

1.0.2 Aims and Objectives

There are two main aims of the project, the first is to create, implement and evaluate an algorithm that assesses escort mission maps once this has been complete, the next aim is to plan and develop a piece of software that can successfully follow the algorithm produced. The objectives below have been selected in order to accomplish these aims.

- Research relevant topics and papers.
- Create or use an existing framework that can be used to develop the map.
- Design an algorithm taking into account size, path of the payload and fairness.
- Develop a programme that can procedurally generate escort mission maps.

1.0.3 Scope

Deliverable

The items to be delivered from this project are: An algorithm that can evaluate any escort mission map and inform the user of the strengths and weakness of the map; A C++ program that can procedurally generate maps that can be tested by the algorithm; the results and analysis of the surveys that examine the credibility of the evaluation algorithm.

Boundaries and Constraints

When testing a map there are a large amount variables to be evaluated, as time is one of the limiting factors in this project, it was decided that 3 major factors would be tested. They are:

- The size of the map.
- The quality of the object that will be escorted's (payload) path.
- The overall balance\fairness of the map

1.0.4 Chapter Outlines

- Background - Discusses the research made into the topic.
- Methodology – States the methods used throughout the implementation process.
- Results – Displays the results from the tests carried out.
- Conclusion – Summarises results, assess s

Chapter 2

Background

In this section, an investigation into escort mission maps and how to evaluate them is presented. Requirements and challenges are presented, and in particular video game maps are examined. Procedural generation one of the main factors of this project is also examined in depth. Finally path-finding and other evaluating techniques is presented, focusing on how to evaluate maps in video games.

2.0.1 Co-operative First Person Shooters

First Person Shooters are a large video game genre that have dominated the market for over 20 years with one of the first major titles being Quake in 1996 and other notable series including Call of Duty, Battlefield and Halo, with the objective of the game usually being to get the most eliminations. A subsection of this genre is the co-operative first person shooter. This type of game heavily relies on teamwork, with games being won by holding onto an objective or pushing a payload ————————————————————————-

Escort Missions

An escort mission is a game-mode in cooperative first person shooter such as Overwatch and Loadout. In this mode there are two teams, the attacking and the defending team. The two teams have different aims in order to win. The attacking team attempts to escort the payload across the map to an objective while the defending team tries to stop them. It is difficult to pin down the first game to have escort missions. One of the more notable was in 2007 with Team Fortress 2. While it was not originally included in the game it was added in the first free update. It is important to note that the original Team Fortress had a game mode called Escort but this has many differences with its sequel, such as the payload being a controllable player and there being three teams. Other games that use this game mode include Global Agenda and Wolfenstein: Enemy Territory.

2.0.2 Maps

In video games, maps are the term used to describe the area that the player can move around in. Different games will have different types of maps depending on how the developer wants the player to play the game. For instance an RPG like Skyrim or The Witcher will usually have a large map for the player to explore the world and find their own path, while other faster paced first person shooters like the Call of Duty franchise might rely on a more linear map to construct the most enjoyable experience for their players. It is very important for a programmer to constantly be planning and evaluating the map they are creating as this will directly affect how enjoyable the game they created is.

2.0.3 Escort Mission Maps

Escort mission maps are the play areas that escort missions are performed in. Between each game these maps can vary greatly so for this project, the definition of an escort mission map must be stated. The maps must have an attacking and defending spawn area. This is the place where the teams start at the beginning of the match and will return too whenever they die. The map must also have an unobstructed route that runs between the spawns, which the payload will be moved through. This route will be generated using path-finding algorithm which is discussed below.

2.0.4 Procedural Generation

The term procedural generation is used in computing to describe a way of creating data algorithmically. It allows for ————————————

Procedurally Generated Maps

It is very common in gaming to use procedural generation, as it allows companies to take short-cuts when creating the game, and allows them to focus on more time consuming aspects of the game such as; graphics and game-play. Procedural generation has been used in many video games for a variety of things. This includes weapon generation in the Borderlands series and Spore which uses procedural generation to create animals and evolve them. However the most common use of procedural generation is to create the map. This has been used in many notable games including, Minecraft , No Man's Sky and Dwarf Fortress. These create huge worlds for the player to roam around in and can allow the map to go on forever. Procedural generation will be used in the project on a much smaller scale then most games to generate the map but it will be used in conjunction with other techniques such as path-finding.

Path-Finding

Many games today have implemented an advanced artificial intelligence for a variety of reasons, an important aspect of this path-finding. This is used for a variety of reasons such as enemy's movement and being used for troop movements in real time strategy games. Path-finding is way for a computer to calculate the shortest route between two points. It does this by searching a graph by starting at one vertex and evaluating the neighbouring nodes. The most common of the path-finding algorithms is Dijkstra's algorithm.

Dijkstra's Algorithm

Published by Edsger W. Dijkstra in 1959, Dijkstra's algorithm is a way to calculate the shortest path between nodes on a graph. Solving the single-source shortest path problem. Nodes on a graph can represent either a 2-dimensional grid or points in a 3-D space. The most common example of a real life application of Dijkstra's algorithm is being used in geographical maps to plot the fastest route between two cities, where each city is a node and the roads are associated with a weight.
The algorithm works by initially marking the distance from the starting point to ever other vertex on the graph.

A Star Path-finding (A*)

The A* algorithm was first presented in 1968 by Peter E. Hart as an improvement on Nils Nilsson's A1 algorithm. A1 was designed to be a heuristic approach to increase the speed of Dijkstra's algorithm invented back in 1964. A* is a search based algorithm that attempts to calculate the most optimal route to a specified goal by travelling through a weighted graph.
The algorithm works by searching the nodes surrounding the current node using heuristic that estimate the distance from the end node. A* contains two lists which are used to sort the nodes, open and closed. The open list contains all the nodes yet to be evaluated, while the closed list contains the nodes that have been assessed. Each node is given their own cost which is used to find the best path with the smaller cost the better. The score is calculated using the formula:

$$f(n) = g(n) + h(n)$$

Where G is cost of travelling to the node and the h cost also known as the Heuristic cost is the cost to travel to the goal. This can be calculated in several ways and determines the efficiency and effectiveness of the algorithm. The F cost is the total cost of adding the two values together.
The total cost is used by A* to guide the algorithm towards the goal. At each iteration of the algorithm the node with the lowest total cost is removed from the open list and

its neighbours are evaluated. This involves searching the closed list to see if the neighbour has already been evaluated and checking if the node cannot be passed. If either of these cases are true then ignore the node, otherwise the node is checked to see if it exists in the open list, if it isn't its F cost is calculated and the node is added to the open list. If the node already exists in the open list then its G cost is compared with the current lowest G cost, if the new G cost is lower than this is the better path. This process progresses until the goal has been evaluated and added to the open list or all nodes have been evaluated and the open list is empty, if this happens the goal cannot be reached from the start point.

2.0.5 Evaluating Maps

2.0.6 Related Work

Chapter 3

Methodology

3.0.1 Gradient Noise

Gradient noise is a form of noise that is used to procedurally generate a texture

Perlin Noise

Perlin Noise was created

Chapter 4

Implementation

4.0.1 Theme

Chapter 5

Evaluation

Evaluation rubbish goes here

Chapter 6

Conclusion

Conclusion rubbish goes here

Bibliography

Appendices

A Initial Project Overview Document

B Week 9 Interim Report

C Diary Sheets

D Timeline