How Does Visualising Path-finding in an NPC Affect How Players Explore a Game Level?

Madeleine Kay

***Abstract*—This paper looks at the use of path-finding and *Artificial Intelligence* (AI) visualisation in digital games. Specifically at the foregrounding of AI and the visualisation of path-finding. While previous papers research AI visualisation and how to achieve it there does not appear to be many papers in the field of visualising AI in digital games. There has been research into the visualisation of path-finding. However, the focus was most commonly on the use of visualisation in game design. This paper looks at visualising various methods of path-finding on enemy NPCs in a 3D Metroidvania game developed in Unity. The results show a number of cases where player exploration and player deaths are affected by either a change in path-finding method or whether the path-finding is visualised. This suggests that these factors affect player exploration and could be taken into account in game design.**

I. Introduction

THIS project looks at visualising the path-finding of an enemy *Non Player Character* (NPC) using *Rapidly-exploring Random Tree* (RRT) path-finding and Unity's *Navigation Meshes* (NavMeshes). Figure 2 shows an example of RRT path-finding. Here the RRT explores an area and then draws a path along the tree between the start and the goal nodes [1].

This paper looks at visualising the tree produced by RRT, the mesh produced by Unity NavMeshes and the process taken to produce each. The visualisations are around an enemy NPC. This allows the players to see where the enemy NPC is going and where it could go. Implementation of the visualisation was developed in a level of a 3D game made in Unity 5.6[1]. Logging tools in the play-testing software export the amount of time the players spent in a level and what percent of the level they explored. Analysis of this data found a number of cases where the different path-finding methods and visualisations affected the percent of the level explored and the number of player deaths but no cases where the time spent in the level was affected. These are analysed in more detail in Section VI.

As Section II shows, previous papers explore visualising *Artificial Intelligence* (AI), foregrounding AI and visualising path-finding. However, there is little on using this in digital games beyond game design. The potential impact of the results of this study could be in game design as the use of different path-finding techniques here had an effect on how players explore. Game designers could take this into account in games ...use this when designing games

The results of this study also found that the visualisation of path-finding It could also be used in a game beyond design as the visuals a

[1]Unity 5.6 Available: https://unity3d.com/get-unity/download/archive

The research question that will be addressed in this project is: how does visualising path-finding in an NPC affect how a player explores a game level?

A. Road-map

Section II is a review of the literature on existing relevant work. The areas reviewed are A* path-finding, RRT path-finding, path-finding, foregrounding AI and exploring game environments. Section III details the methodology used in the experiments in this study. Section IV details the development of the software used for the experiments in this study. Section V details the data collected and how it was filtered and section VI looks at the analysis of this data and what the results imply. Section VII details the potential issues and ways to address them in future work. Section VIII looks at future research that could be done on this subject.

II. Related Work

The focus of this paper is on path-finding so that is the main focus of the literature review. The path-finding methods looked at are A* path-finding in Section II-A and RRT path-finding in Section II-B. Section II-C looks at applications of path-finding in games. Section II-D looks at foregrounding and visualising AI. While there is research on AI visualisation there is a lack of it for digital games and what research is on games often focuses on game development and level design as can be seen in Section II-D. This literature review shows that while there are many papers on AI visualisation there are few on AI visualisation in digital games and many of the ones on games are on visualising AI to aid development, not for the end user.

A. A* Path-finding

A* path-finding is widely used in both robotics and digital games [2]. In games, A* appears to be the most commonly used path-finding algorithm [2].

Hart *et al* [3] first proposed A* path-finding in 1968 as an improvement on Dijkstra's algorithm. A* aims to expand the fewest nodes possible to minimise the cost of the path where the cost is the distance between the start and goal nodes. Figure 1 shows pseudo-code for implementing A*.

Algfoor *et al* [2] survey numerous papers on path-finding. Their focus is on the use of different grid shapes in path-finding and the numerous algorithms available [2]. The most popular being the A* algorithm for use in both digital games and robotics. They survey many grid types and gave the advantages of each.

```
1  Put node_start in the OPEN list with f(node_start) = h(node_start) (initialization)
2  while the OPEN list is not empty {
3    Take from the open list the node node_current with the lowest
4         f(node_current) = g(node_current) + h(node_current)
5    if node_current is node_goal we have found the solution; break
6    Generate each state node_successor that come after node_current
7    for each node_successor of node_current {
8      Set successor_current_cost = g(node_current) + w(node_current, node_successor)
9      if node_successor is in the OPEN list {
10       if g(node_successor) ≤ successor_current_cost continue (to line 20)
11     } else if node_successor is in the CLOSED list {
12       if g(node_successor) ≤ successor_current_cost continue (to line 20)
13       Move node_successor from the CLOSED list to the OPEN list
14     } else {
15       Add node_successor to the OPEN list
16       Set h(node_successor) to be the heuristic distance to node_goal
17    }
18     Set g(node_successor) = successor_current_cost
19     Set the parent of node_successor to node_current
20   }
21   Add node_current to the CLOSED list
22 }
23 if(node_current != node_goal) exit with error (the OPEN list is empty)
```

Fig. 1. Pseudo-code for A* path-finding [3] [4].

Nash *et al* [5] say that A* cannot always find the true shortest path as it is limited to the grid. The shortest path can be found A* with post-smoothing paths or by using A* variants such as Theta* [5], [6]. Theta* expands on A* as it allows for all edges and angles in the grid to be used. Therefore, a path with a more optimal distance can be found.

Firmansyah *et al* [6] compare A* with Theta*. They found that they performed similarly time wise. However, A* produces a path with fewer nodes expanded and Theta* produces a shorter path.

Hu *et al* [7] propose an implementation of A* path-finding in the Unity engine, the engine used in this project. While their implementation is in an older version Unity the implementation in Unity 5.6 should still be similar. A further paper on path-finding is Wang and Lu's [8] paper which looks at path-finding in a 3-Dimensional environment. While again they were using A* they look at using A* in 3D and suggest using nodes instead of a grid.

Tremblay *et al* [9] look at the use of path-finding algorithms in level design. They compared 2-Dimensional A*, 3-Dimensional A*, RRT using A* for motion planning, RRT using *Monte Carlo Tree Search* (MCTS) and MCTS alone. 2-Dimensional A* consistently gave the fastest result with the highest success rate. As this was for game level design instead of gameplay they ran the algorithms 1000 times on high specification computers.

For this project, the algorithm will only be run once as it is being used at runtime in the game instead of in the game design stage that Tremblay *et al* [9] tested in. As A* has a 100 percent chance of finding the shortest path this should not cause an issue.

B. RRT and Path-finding

Rapidly Exploring Random Trees (RRT) is a search method used more in robotics than in digital games [10], [1]. Kuffner and LaValle [1] first proposed RRT in 2000. They intended to produce a random algorithm more efficient than the other search algorithms available at the time. Figure 2 shows Kuffner and LaValle's [1] RRT Path Planner. Path Planner is a variant of RRT which has the intended use of finding paths from the generated tree.

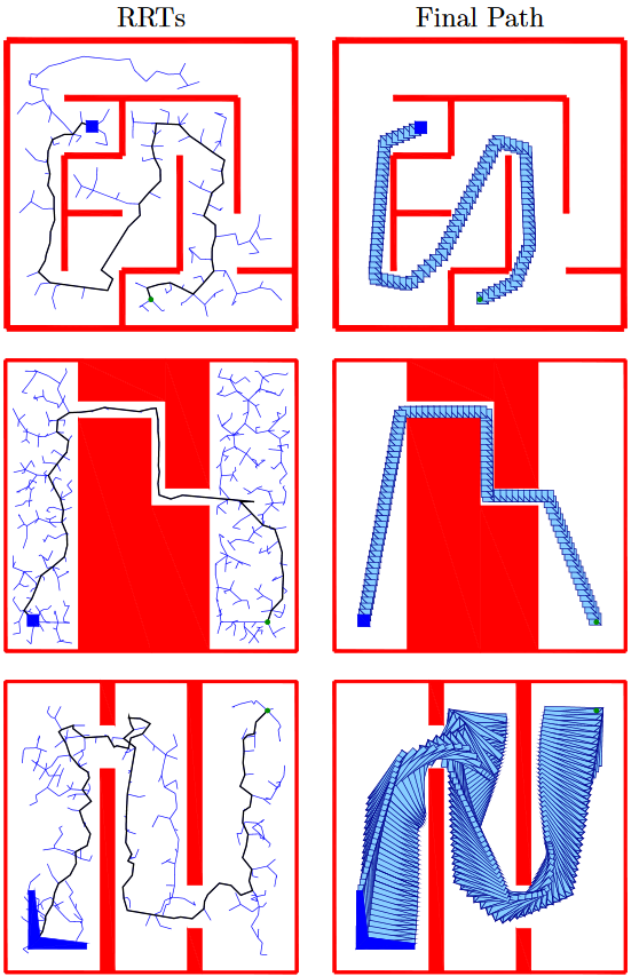

Fig. 2. Kuffner and LaValle's RRT path planner building a tree and plotting a path through it [1].

The process for RRT involves the random placement of nodes. A parent is then selected by finding the closest pre-existing node [1]. Algorithm 1 shows the pseudo-code for the RRT algorithm.

RRTs goal is to find a path between two points with no collisions. The path found may not be the optimal path though [1], [11]. Karaman and Sertac [12] say that the chance of RRT finding an optimal path is very unlikely [12], [9]. Where A* is guaranteed to find the shortest path as seen in Section II-A, RRT is unlikely to find the shortest path and may not even find a path at all. Karaman *et al* propose a variant of RRT called RRT*. RRT* starts the same as RRT, however, when a new node has to choose a parent node instead of selecting the nearest node it evaluates the cost of the nodes in regards to reaching its goal. Each iteration re-evaluates the parent nodes to reduce the cost. Rewiring of the tree happens when a lower cost path is found.

C. Applications of Path-finding

Bauer and Popovic [13] use RRT for level design in digital games. Like other papers mentioned in Section II-D, they

Algorithm 1 RRT Pseudo Code

BUILD_RRT($q$init)
T.init($q.init$)
for $k = 1$ to K **do**
  $q$rand $\leftarrow RANDOM\_CONFIG()$
  EXTEND (T, $q$rand)
end for

EXTEND(T, q)
$q$near $\leftarrow$ NEAREST_NEIGHBOR(q, T);
if NEW_CONFIG($q, qnear, qnew$) **then**
  T.add_vertex($q$new);
  T.add_edge(qnear, qnew);
  if $q$new = q **then**
    Return $Reached$;
  else
    Return $Advanced$;
  end if
  Return $Trapped$
end if

visualise the data to aid users [13], [14]. This data visualisation is for use in game development to aid game developers or to analyse procedurally generated levels.

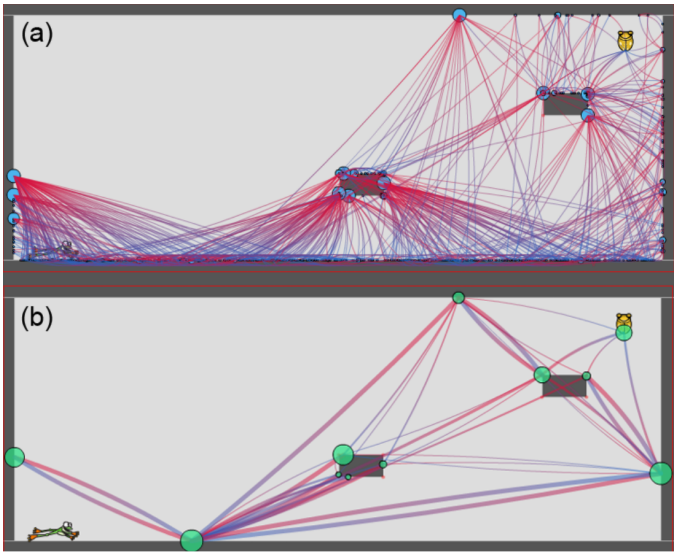

Fig. 3. Bauer *et al*'s[13] graph-based representation of RRT with and without clustering.

Their focus is on level design, not game-play. They propose a tool that analyses a level generated by PCG or a level designer. They then use RRT to calculate possible routes the player could take when playing. Figure 3 shows the output of the tool once it has been run on a basic level design. Figure 3(a) may be difficult to for designers to interpret. So, they use van Dongen's method [15] for graph clustering to make the output more legible as shown in Figure 3(b) [13].

Mendona *et al* [16] look at path-finding both in robotics and digital games. Their focus is on stealth path-finding in games and applying that to robotics. Like RRT, the methods they propose do not necessarily find the shortest path [12], [16]. Instead, they try to find the path where the agent spends most of the time in cover. They generate custom *navigation meshes* (navmeshes). Then they assign a weight to each polygon in the navmesh depending on how close it is to being behind cover.

As Mendona *et al* [16] use path-finding to find a stealth orientated path. The path with the optimal distance may not always be the path with the lowest cost in relation to the AI agent being in cover. Therefore, a requirement might not always be the shortest path.

Tremblay *et al* [17], like Bauer and Popovic [13], also use RRT visualisations to aid level design. They use RRT to visualise possible moves the player could make. Then they use clustering to make the results less cumbersome to the user [17]. Similar to Mendona *et al* [16], they focus on designing stealth games and finding stealth orientated paths in the game levels [17]. Figure 5 shows the basic level design they used with three areas for the AI agent to take cover. This allows level designers to see where players are likely to go and adjust the level design accordingly [17]. The use of RRT, in this case, is because it is flexible and inexpensive. Also, its random nature allows for the mimicking of a wider range of player behaviours [17].

This project uses RRT but not for reflecting player behaviour. Instead, its use is for creating a visualisation that fits with the game and that may be interesting for the player to interact with. A path that is interesting to play with is more important in this project than a path with an optimal distance. There is then a comparison between a visualisation of the Unity navmeshes against the RRT visualisation.

While the use of RRT to find a stealth orientated paths is different to its use in this project a potential problem is that Tremblay *et al*'s results showed that the chances of their RRT implementation finding a path decreased as the grid size increased. It also decreased as the number of attempts decreased as shown in Figure 4. This suggests that a potential issue with RRT is that it may not always find a path.

Tremblay *et al* [9] again look at the use of search algorithms for use in game development. This work builds on their 2013 paper on RRT in stealth games. As previously mentioned in Section II-A they look at the use of A*, MCTS and RRT to visualise player behaviour in platform games similar to Bauer [9] [13]. RRT is limited to drawing straight lines between the nodes, as straight lines are not always possible in RRT Tremblay *et al* [9] also use either A* or MCTS to find a path to the node and ensure that it is reachable. While A* is consistently fast with high success rates, RRT has varying results. RRT using MCTS for motion planning varied between 35 and 84 percent success rates and had the longest run times. In comparison, RRT with A* motion planning consistently had a success rate of over 60 percent. However, the run times for this combination varied. Both of Tremblay *et al*'s [9], [17] papers showed that a potential issue with the use of RRT is that it may not find a path. Both papers ran RRT combinations over 1000 times. In comparison, this paper runs RRT at runtime and

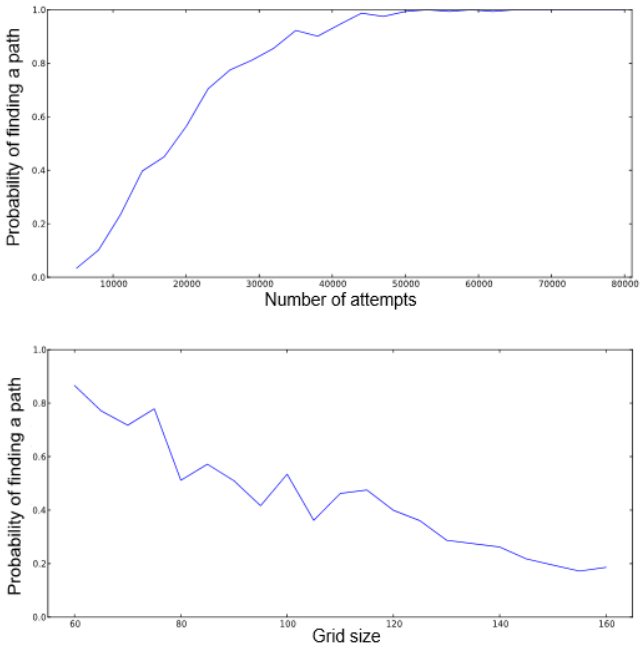

Fig. 4. Performance analysis of Tremblay *et al*'s [17] RRT when running on a Metal Gear Solid level [18].

only runs the algorithm once unless an RRT variant such as RRT* is used. If a path is not found a tree is still produced and visualised so may still influence players.

Section II-D looks at Third Eye Crime in more detail for its use of AI visualisation. However, it also uses visualisation of enemy paths as an important mechanic [19], [20]. Isla [19] uses Occupancy Maps to show where the enemy NPC thinks the player could be, as can be seen in Figure 6. Occupancy or Influence Maps do not produce a path instead they show the probability of the player being in different locations across the map [19], [21]. The enemy then moves to investigate that area reducing the probability of the player being there. Similarly, Miles and Louis [21] also used influence maps. While their example is specific to *Real Time Strategy* (RTS) games, like Isla they used Occupancy Maps. They used them as a base for A* path-finding instead of A* using the map itself for path-finding.

D. Foregrounding and Visualising AI

Most modern digital games make use of AI. However, it is rarely foregrounded or visualised in those games. Treanor *et al* [22] say that often the design of AI in games is to fit the game and complement gameplay. These AI are supporting the gameplay rather than being central to it.

Treanor *et al* [22] surveyed many games that foreground or visualise AI in different ways. From this, they propose a series of design patterns for foregrounding AI in digital games. The two design patterns relevant to this project are "AI as a Villain" and "AI is Visualised". They describe the first pattern as having the AI try to not outright defeat the player. Instead, it is designed to create an experience like in the game Alien Isolation [22], [23]. In Alien Isolation the enemy AI hunts the

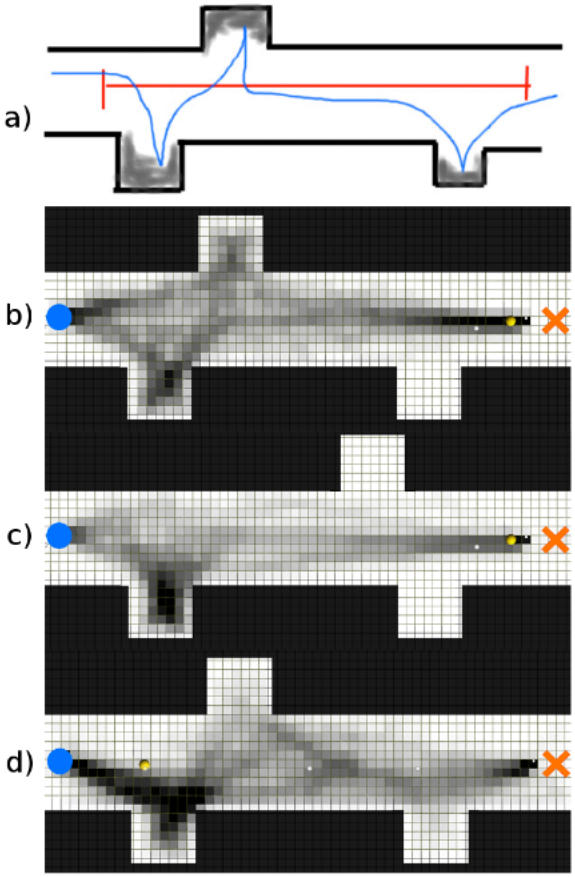

Fig. 5. Heat maps of a single level where the design has been changed each time [17].

player. This is foregrounding as the player must observe the AI and learn how to avoid it. There is also some visualisation as the player has a scanner that informs them of the enemy's position.

This paper uses the "AI as a villain" pattern as each enemy NPC has their path-finding visualised around them. The players can consider this when exploring a level so they do not get attacked by the enemy. The use of this pattern also aims to have an NPC that creates an experience rather than one that always finds the player, similar to Alien Isolation [23], [22]. While the player likely does not want to be caught by the enemy NPC they may want it to chase them so they can learn its patterns or lead it away from other enemies to make it easier to attack.

The second relevant design pattern is "AI is Visualised". This is where there is a visual representation of the AI's state or decision making in the game [22]. Most games hide this from the player but this design pattern visualises it making it mechanic. The example given by Treanor *et al* [22] is the game Third Eye Crime. Third Eye Crime is a game that follows the "AI is Visualised" design pattern [19], [20]. The game uses probabilistic object tracking through Occupancy Maps. As the enemy moves around the map it removes areas where the player is not from the Occupancy Map [19]. Generally, stealth games involve avoiding enemies. This design encourages the

player to trigger the mechanic, allowing them to use the visualisation to mislead and avoid the enemy [19], [20].

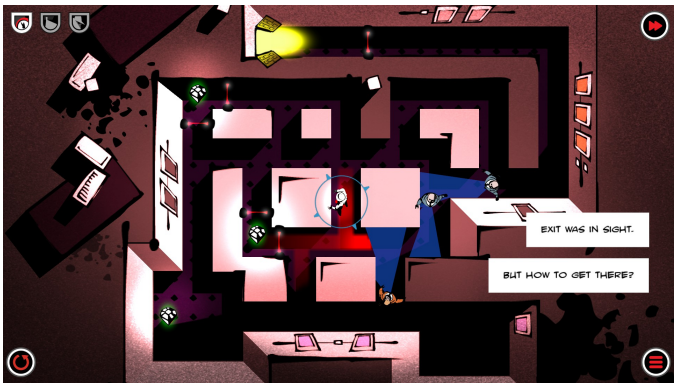

Fig. 6. A screen-shot from Third Eye Crime [20].

This pattern is relevant to this project as the enemy NPCs have RRT path-finding visualised around it allowing the player to see where the enemy is searching and decide how to overcome or outsmart it.

While Haworth *et al* [14] do not visualise an AI decision-making process they do visualise the possible decisions available to the player in a game on a tree structure. They research visualising decision trees in a game to see what effect it had on gameplay and the analytical reasoning of children. Their study does not come to any definite conclusions. However, their results suggest that the trees aided players as in the later levels of the game the children without the visualised decision tree struggled to beat the game. However, they noted this could also be due to unbalanced difficulty in the later levels. This makes the usefulness of the visualised tree questionable in this example.

A potential issue with this study is that Haworth *et al* [14] only tested the tree on a simple 2-Dimensional maze game for school children. Of these children, only a few had experienced playing digital games before. This could mean that the data is not relevant for 3D games or games available to buy on the market. In contrast, Isla's [19] visualised Occupancy Maps are in the game Third Eye Crime which is available for purchase on Steam, IOS and Android [19], [20].

Like Haworth *et al* [14], Bauer *et al* [13] also research visualising tree structures [13]. However, they used RRT which is an AI technique.

Another use of visualisation, mentioned in earlier sections, is in game design. Often to check player behaviour in player testing or to aid the design of levels [24], [13], [17], [9].

E. Exploring Game Environments

One method of guiding players through games is to use way-finding. Way-finding in games is often architectural differences or visual cues in the environment that guide the player to an area of interest [25], [26]. Way-finding cues are often subtle cues in the environment such as ivy growing up a wall to suggest the player can climb there.

The intention of visualising path-finding in this project is not to guide the players. However, like Si *et al* [25] it observes how players navigate and explore levels and whether, like the presence of way-finding cues, it affects player behaviour. Moura and Bartram [27] investigate the effects of different way-finding cues on players. They looked at methods used in AAA games and mimicked them in their own game. Their results show that the absence of way-finding cues was obvious to players. In contrast, the version with way-finding cues does not have enough cues to sufficiently guide the player. These results suggest that way-finding cues alone may not be enough to guide the player. They concluded that there is a need for more research as the results were inconclusive.

While this project focuses on enemy NPC path-finding this could be another interesting application of path-finding and visualisation. The path-finding could remain hidden from the player, as it normally is in digital games, but way-finding cues could be placed based on the path. This could then subtly guide the player through the game.

Si *et al* [25] investigated how players explore virtual environments. While their experiments were specific to Real Time Strategy (RTS) games the results may apply to other game types. Si *et al* [25] say that three common types of spatial exploration are; environment mapping, bonus item collecting and location/landmark discovery. The relevant exploration type for this project is spatial mapping. Firstly, as it logs what the enemy NPC is doing. Secondly, as it is the player behaviour that is being measured by the logging tool in the software.

A paper looked at in Section II-D was Haworth *et al*'s [14] study. While they do not look at player exploration the game they used involved a map where participants had to explore a maze. Participants who had the decision tree visualisation found it easier to navigate the maze. While Moura and Bartram [27] suggested way-finding alone was not enough to guide a player this suggests that visualisation could aid the exploration process [14].

III. Methodology

The research question addressed in this project is: how does visualising path-finding in an NPC affect how players explore a game level? The hypotheses drawn from this question are listed in Section III-A.

An a-priori power analysis was performed to determine how many participants were required for an effect size of 0.4. 0.4 was chosen as this percentage would have a more significant impact on gameplay. An effect size of 0.3 would have also found a large enough difference to impact gameplay. However, a power analysis for an effect size of 0.3 found that 90 participants would have been required, which was not feasible in the given time frame of this study.

The results showed that a minimum of 52 participants was required. As not all datasets were complete further participants were tested resulting in 62 sets of results. The play-testing was completed by participants both online and in person to reach the required sample size. Both participant types were given the same instructions and questionnaire.

Of the play-tests done in person, a large majority of the players were students studying game development-related courses, the potential issues relating to this are discussed in Section VII.

TABLE I
SUBREDDITS USED FOR PLAY-TESTING

Subreddit	Subject
r/SampleSize	Finding participants
r/UniUK	Thread on finding participants
r/gamedev	Game Development
r/Unity3D	Game Development
r/playmygame	Finding game play-testers
r/playtesters	Finding game play-testers

The rest of the play-tests were done online through Reddit. Many of the subreddits it was posted on were related to gaining participants and play-testers. Where the in-person play-testing has the issue of the players all being students, the online players could have a similar problem as the players are all users of the subreddits listed above that are looking for surveys to complete. Some of the subreddits listed were also related to game development and therefore the feedback given was related to the state of the game, not the path-finding.

While 21 play-tests were completed online 10 of them have incomplete or invalid data. Therefore, those participants results were filtered out of the final dataset and more tests were done in person to reach the sample size. The filtration of the data is discussed in more detail in Section V.

TABLE II
PLAY-TEST VARIATIONS

Play-test Variations	RRT Used	Path-finding Visualised
Variation 1	X	X
Variation 2	X	
Variation 3		X
Variation 4		

A. Hypotheses:

Table III shows the hypotheses that are tested in this experiment. The methodologies that are used to test these hypotheses are play-testing and a questionnaire. As this required human participants for both parts ethics approval for this methodology was gained from Falmouth Universitys Research Ethics Board.

The game the path-finding variations have been tested in is 'Gates of Amenti' a 3D Metroidvania game which has been designed with a focus on exploration. A Metroidvania game is an action-adventure game with a focus on exploration to discover new areas and power-ups [28].

Players were given one of the game variations to play and then asked to complete a questionnaire on their experience.

B. Play-testing

There are four variations of the game that can be seen in Table II. A random number generator was used to randomly select a game version for each participant.

During the play-tests players were assigned one of the four variations of the game. The players were asked to play that version for as long as they wanted. While playing, the software logged the time the player spent in the level per life and what percent to the level they explored. The percent explored is calculated by recording how many doors the player walks through as the game requires players to unlock doors to progress. This data is then exported into a .CSV. The .CSV is then analysed using the R statistical package[2] in R Studio[3]. The primary statistical tests used were T-Tests alongside correlations.

After completing a play-test of the game participants were asked to complete a questionnaire on their thoughts on parts of the game and Yee's taxonomy to determine their player type.

C. Questionnaire

Alongside play-testing, participants were also asked to fill out a questionnaire on the game online using Google Forms. Nordin *et al* [29] say that questionnaires are vital for understanding how players feel when playing digital games [29], [30]. They can also help give uniformity and consistency to the data gathered from participants [30].

Questionnaires are also beneficial as they can prompt players to give answers they may not have given spontaneously. For this study either a new questionnaire could be made from the relevant hypotheses or an existing one could have been used. There are a wide variety of questionnaires that already exist to measure players experiences in play testing [29], [31]. These come with the benefit of being more likely to be thoroughly tested such as the *Immersive Experience Questionnaire* (IEQ) which uses Likert scale responses or Yee's taxonomy [29], [31], [32], [33].

However, Nordin *et al* [29] say there are many issues with using pre-existing questionnaires. Firstly, many of these questionnaires are not readily available. Another issue is that not all questionnaires have been thoroughly checked for validity. There is also the potential issue of a researcher not fully understanding the questions put forward by another researcher or they may not understand what data that question is intended to get.

Another issue with the use of questionnaires is self-report bias. Some of the hypotheses being tested relate to the player being confused or not understanding the visualisations. Players are unlikely to admit to not understanding the game making the data gained invalid [34].

Most of the hypotheses in this paper require quantitative data from the play-tests. However, some require qualitative data so a questionnaire is necessary. In addition to the hypotheses based questions, there were also questions to determine the participants player type, as some players are more interested in exploration in games than others and that could influence the results. Firstly Bartle's Taxonomy [35] was looked at, this taxonomy appears to be widely used but is not validated. Yee's Taxonomy similarly uses a questionnaire to determine a person's player type but is also been validated [33], [32]. In this paper, Yee's questionnaire for player taxonomies is used in combination with questions based on the hypotheses that require qualitative data.

[2]R Available at: https://www.r-project.org/
[3]R Studio Available at: https://www.rstudio.com/products/rstudio/download/

TABLE III
HYPOTHESES

	Hypothesis	**Null Hypothesis**	**Data Collection Source**
1	Visualising path-finding has an effect on the percent of the level the participant explores.	Visualising path-finding has no effect on the percent of the level the participant explores.	Game Data
2	In comparison to RRT visualisation, Unity NavMeshes visualisation has an effect on the percent of the level the participant explores.	In comparison to RRT visualisation, Unity NavMeshes visualisation has no effect on the percent of the level the participant explores.	Game Data
3	Using RRT path-finding has an effect on the percent of the level the participant explores.	Using RRT path-finding has no effect on the percent of the level the participant explores.	Game Data
4	Using NavMesh path-finding has an effect on the percent of the level the participant explores.	Using NavMesh path-finding has no effect on the percent of the level the participant explores.	Game Data
5	Visualising path-finding has an effect on the length of time the participant spends in the level.	Visualising path-finding has no effect on the length of time the participant spends in the level.	Game Data
6	Using RRT path-finding finding has an effect on the length of time the participant spends in the level.	Using RRT path-finding finding has no effect on the length of time the participant spends in the level.	Game Data
7	Visualising Unity NavMeshes has an effect on the length of time the participant spends in the level.	Visualising Unity NavMeshes has no effect on the length of time the participant spends in the level.	Game Data
8	Visualising RRT path-finding has an effect on the length of time the participant spends in the level.	Visualising RRT path-finding has no effect on the length of time the participant spends in the level.	Game Data
9	Visualising path-finding has an effect on the number of times the player died.	Visualising path-finding has no effect on the number of times the player died.	Game Data
10	Using RRT path-finding has an effect on the number of times the player died.	Using RRT path-finding has no effect on the number of times the player died.	Game Data
11	The visualisation of Unity NavMeshes is comprehensible to players.	The visualisation of Unity navmeshes is not comprehensible to players.	Questionnaire Data
12	The visualisation of RRT path-finding is comprehensible to players.	The visualisation of RRT path-finding is not comprehensible to players.	Questionnaire Data
13	The comprehensibility of the visualisation of RRT path-finding is no different to Unity NavMesh visualisation to players.	The comprehensibility of the visualisation of RRT path-finding is different from Unity NavMesh visualisation to players.	Questionnaire Data

IV. SOFTWARE DEVELOPMENT

The engine used to develop the play-testing software was Unity 5.6.3f1[3] and Visual Studio Community 2017 15.5.4[4]. The analysis was carried out and the graphs were produced in R version 3.4 in R Studio.

The base game 'Gates of Amenti' was developed with the student-led games studio Bears are OP[5]. The game was developed over the course of seven months using Agile and Scrum project management techniques. The game was still in development while being adapted for use in the play-testing so was unfinished. This is a potential issue looked at in Section VII.

Agile was not used while developing the play-testing variation of the game as it was being developed by one programmer as opposed to the multi-disciplinary team developing the main game. Daily stand-ups were not necessary and a majority of the required features from the game had already been developed. For the development of the play-testing software, the Prototyping lifecycle method was used [36]. Prototyping was used as a majority of the game itself had already been developed, only a few new features required implementation. These features were; RRT path-finding, path-finding visualisation, data recording and data exporting. Basic versions of these features were added to the game and then improved on until the game was suitable for playtesting. In this case, suitable means that all the features worked and players could play through the level with minimal issues.

The first feature implemented was the RRT path-finding. This was developed in a separate scene in the game at first. The first iteration of the RRT built a tree in 2 dimensions in a cube. This was to test the implementation of RRT before trying it in a more complex environment. In the game, the basic data recording functionality was added early on but the ability to export it and view it as a part of the UI was added near the end of the development cycle. The first iteration of the NavMesh visualisation was implemented using Unity's low-level graphics library, GL[5]. This was only used for testing as it could only be observed using the in-game camera, it was not visible in the scene view. This was a test to see how the NavMesh could look. This was refactored to use Unity's line renderers to draw the polygons of the NavMesh, as can be seen in Figure 10. The RRT was then visualised in a similar manner using line renderers to draw the branches between the

[3]Unity 5.6 Available: https://unity3d.com/get-unity/download/archive

[4]Visual Studio Community 2017 Available: https://www.visualstudio.com/downloads/

[5]Gates of Amenti: https://bearsareop.itch.io/

[5]Unity GL Library: https://docs.unity3d.com/ScriptReference/GL.html

nodes.

The quality of the software is judged on its functionality and whether it is a playable game. For functionality, the features it was judged on were whether the data could be exported in a form that it could be analysed in, as the game exported the time, exploration percent and game version to a .CSV file this requirement was met. The other two main requirements were that the software could generate an RRT and that both RRT and NavMesh path-finding could be visualised. While both of these were functional they could have been of a higher quality which is discussed in more detail in Section VII.

A. Software Testing

To test that the play-testing software was functioning as required a simple bot was developed. All the bot needed to do was move forward to test whether the doors registered when walked through and then whether the timer and data export worked.

1) Doors: The first bot is used to test whether the doors record whether the player had moved through them. Whether a door has been walked through is used to determine what percent of the level the player explored. Figure 7 shows an example of the code used in the bot to disable the door trigger collider after use. The test for the doors is a bot moving through the door and checking that the GameState class recorded it.

```
18      private void Start()
19      {
20          doorCollider = this.GetComponent<Collider>();
21      }
22
23      void OnTriggerEnter(Collider collision)
24      {
25          if (collision.gameObject.name == "Player")
26          {
27              doorWalkedThrough = true;
28              doorCollider.enabled = false;
29          }
30      }
```

Fig. 7. A segment of the DoorState class that updates the door when the player walks through.

2) Timer and Data Export: The second test is to check whether the software could successfully export the necessary data to a .CSV file. This bot is the same as the door testing bot. Figure 8 shows an example of the data exported from the game.

	A	B	C	D	E
1	Player number	Pathfinding Visualised	RRT Used	Time in Level	Percent Explored
2	0	True	True	98.78061	0.3809524

Fig. 8. An example of the data exported from the game.

3) Visualisations: The final test is whether the path-finding visualisation works as intended. This does not require a bot it can be tested by observing the scene in Unity. Figure 9 shows the visualisation of the Unity NavMesh working in the engine. Figure 10 shows the visualisation of the RRT path-finding working in engine.

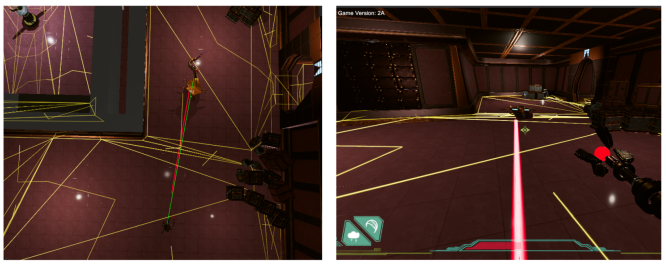

Fig. 9. The navmesh visualisations shown both in editor and in scene.

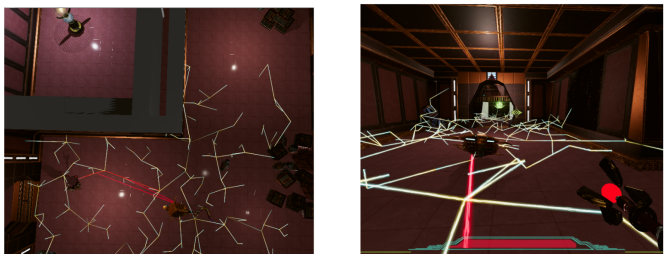

Fig. 10. The RRT visualisations shown both in editor and in scene.

V. Data Collection and Filtering

All the data used either came from the data exported from the game or from the questionnaire. There are 107 datasets from play-testing 62 participants. However, as not all participants gave complete datasets 10 datasets have been removed leaving 97 datasets from 52 participants.

When the player dies in-game the game resets, this lead to a number of results having 0 percent explored and low play times. As these were not proper play-tests the 11 instances of this were removed from the dataset leaving 86 datasets. Many players also died within the first few minutes of playing, giving a play-test with a low time and exploration amount, the 34 instances of this were also removed from the dataset. Removing these left 52 datasets; one per player. There is a potential issue here where the player may not explore and area again if they have already explored it before dying. However, data from play-throughs where the player died in under 5 minutes would likely have had a more significant effect.

VI. Analysis and Interpretation

The data analysis was conducted using R in R Studio. The results from this can be seen in Table IV. Figure 11 shows an example of the code running in R, it shows a correlation and a T-Test that have been run.

Table IV shows the correlations and P-Values from the T-Test results calculated using R. In a majority of cases there was insufficient evidence to reject the null hypothesis. However, there are some cases that have a P-Value of less than 0.05 and are thus statistically significant.

A. RRT and the Percent of the Level Explored

The first significant correlation is a small positive correlation of 0.2807 between the percent of the level explored and the

TABLE IV
T-TEST AND CORRELATION RESULTS

Independent Variable	Dependent Variable	Correlation	T-Test - P Values
Visualising Path-finding	Percent Explored	-0.1766742	0.2009
Visualised Path-Finding Methods	Percent Explored	0.5500408	0.003752
RRT Used	Percent Explored	0.2807048	0.04425
Non-Visualised RRT Use	Percent Explored	0.2807048	0.04425
Visualised RRT	Percent Explored	0.1697141	0.4229
Visualised NavMesh	Percent Explored	-0.4893995	0.01176
Visualised Path-finding	Time	0.0002296794	0.9987
RRT Used	Time	0.06559258	0.6445
Visualised RRT	Time	-0.0153404	0.9384
Visualised NavMesh	Time	0.001491802	0.9943
Visualised Path-finding	Understandable Visuals	0.1627247	0.2731
RRT Used	Understandable Visuals	0.1910156	0.3341
Path-finding Visualised	Logical Enemy Movement	-0.2728879	0.06224
RRT Used	Logical Enemy Movement	-0.01419318	0.9234
RRT Used	Deaths	-0.2656382	0.05917
Visualised Path-finding	Deaths	0.318452	0.01833

```
> cor(PF_VisData$Percent_Explored,PF_VisData$RRT_Used)
[1] 0.5500408
> t.test(PF_VisData$Percent_Explored~PF_VisData$RRT_Used)

        Welch Two Sample t-test

data:  PF_VisData$Percent_Explored by PF_VisData$RRT_Used
t = -3.2666, df = 20.61, p-value = 0.003752
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.4918121 -0.1089206
sample estimates:
mean in group FALSE  mean in group TRUE
         0.4139194           0.7142858
```

Fig. 11. An example of the R code used for calculating T-Tests and Correlations.

use of RRT path-finding. This suggests that players explore more in the RRT variations.

The graph in Figure 12 shows that the percent explored between the two RRT variations are both close, with the 'not visualised' variation being slightly lower.

There is no significant correlation between the RRT being visualised and the percent which would suggest that visualising the RRT does not affect the percent of the level explored. However, when just looking at visualised path-finding there is another significant correlation between the type of visualised path-finding used and the percent of the level explored. There is a strong positive correlation of 0.55 which again suggests that players explore more when RRT is used in comparison to NavMeshes. Another RRT related significant correlation is between non-visualised RRT and the percent explored which has a positive correlation of 0.281.

There is no significant correlation between RRT use and the time spent in the level. This suggests while players explore more when the RRT is used they do not spend more time in the level.

Some participants commented that they thought the RRT

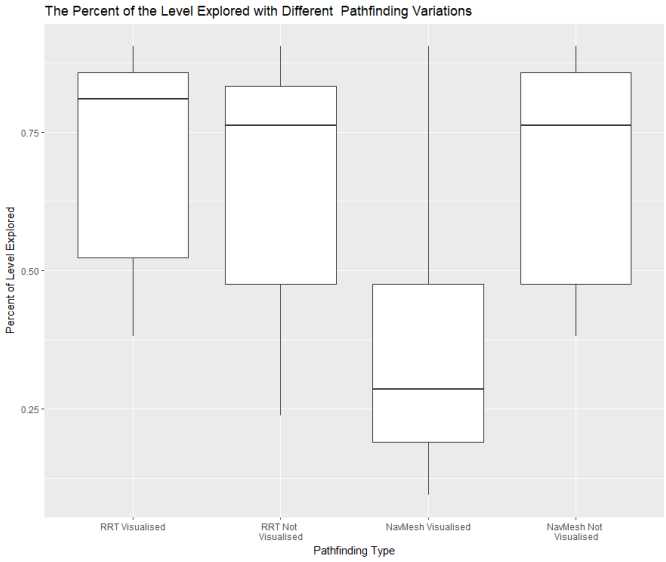

Fig. 12. Box and Whisker plot of the percent of the level explored with the 4 game variations.

visualisation was leading somewhere and followed it through the level or went to it when they saw it. However, this only accounts for when the RRT is visualised. Another possibility is that the RRT was spawned in a square and often did not fill a room which may encourage players to move around the edge of the room rather than down the middle. This may have made it easier to find hidden doors increasing their exploration percentage.

Another reason for this could be that in the play-testing software each RRT was limited to a single room as enemies could not follow players out of rooms this may have allowed them to avoid combat, or make combat faster, allowing them

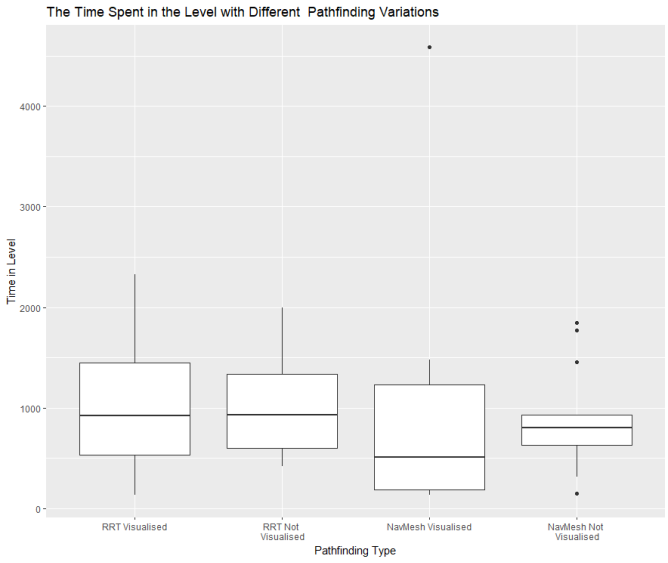

Fig. 13. Box and Whisker plot of the time spent in the level with the 4 game variations.

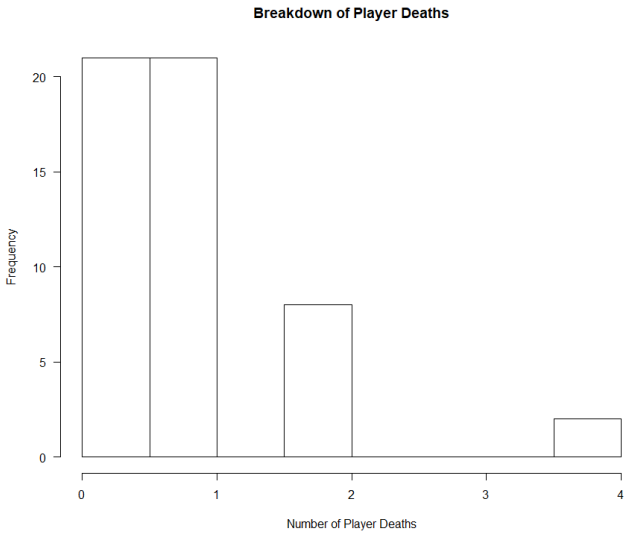

Fig. 14. Histogram of the frequency of the number of player deaths.

to explore more.

B. NavMesh Visualisation and the Percent of the Level Explored

In contrast to RRT having a positive correlation with the percent of the level explored, the visualisation of NavMeshes has a negative correlation of -0.4894 with the percent explored. This suggests that players explored less when the NavMesh was visualised. As many play-testers were game development students using Unity they may have known that the visualisation was of the NavMesh. This may have helped them find a quicker route through the level or lead them to avoid rooms with enemies in. Another reason could be that seeing the pathfinding showed them the possible pathways aiding them in finding a more direct path through the level

C. Visualised Path-Finding and Player Deaths

Another significant correlation is the positive correlation between the visualised path-finding and the number of player deaths. This means that players with the variations with visualised path-finding died more than players without it. One reason for this could be the player being distracted by the visualisation and therefore did not focus on enemies.

In the RRT variation, the tree visuals are only in areas where there are enemies, as some players claim to move to the RRT whenever they see it they could have been more likely to find enemies than players without the visualisation.

As can be seen in the graph in Figure 14 an equal number of players died 0 or 1 time. The one death could also be due to the player not knowing how to play the game which may have skewed the results. A smaller number of players died 2 or 4 times.

D. Questionnaire Data

There is no correlation or statistical significance between the questionnaire data and the exported game data. This could potentially be due to poorly worded questions or players not understanding what the visualisation, as mentioned in more detail in Section VII. Many players claimed to understand the path-finding visualisation but, their comments after playing suggested that they do not as many believed it was an aesthetic choice.

Apart from the cases previously mentioned, there was insufficient evidence to reject the rest of the null hypothesis. This suggests that neither of the visualisations were more understandable than one another to the player. Also, neither the change in path-finding method nor the visualisation affected the time spent in the level even when the percent explored increased.

Figure 13 shows that the times for all four variations were close. However, unlike figure 12 there are a number of outliers on the NavMesh variations. Firstly, on visualised NavMeshes, there was an outlier with over 4000 seconds spent in the level where a majority of the other play-tests for all four variations were under 2000. Finding this outlier in the play-test data showed that this player has a very high play time with a low exploration percent. As this was a remote player this suggests that the player just launched the level and left the game running without playing. Though the player did die so they may have made some attempt to play.

VII. Potential Issues and Future Work

While the methodology of the study was appropriate in this case, there were many areas that could be improved in future work to improve the quality of the data collected. This section identifies a number of weaknesses in this study and how they could be addressed in future work.

A. Play-testing Game

One issue with the play-testing software is that the game was not specifically designed for use in this paper. While the game is a Metroidvania game with a focus on exploration

a few participants noted the enemies in the game were very aggressive and a majority of players died at least once which could have affected their desire to explore. In a future study, a game would either be designed specifically for the play-testing or altered more to suit the study.

B. Participants

Another issue with using the game 'Gates of Amenti' is that many of the play-testers were game development students who may have play-tested other levels of the game or seen the game trailer during the university show and tell day. This means that they could have been looking for things they have previously seen in the trailer as opposed to exploring or being affected by the visualisation. As a majority of them were students studying a game development course they play-tested the game looking for bugs and issues instead of just playing the game which again may have affected the results.

In future work the player would be selected randomly and from a larger pool of people.

C. Performance Issues

The RRT variations of the game were more resource intensive to run than the NavMesh versions. This lead to a frame rate drop at the start of the game in the RRT variations. Many of the play-tests used Falmouth Games Academy's computers which were capable of running both versions well. However, some play-tests were done remotely and therefore the computer specification and performance of the game are unknown. In future studies, this would be combated by either doing all the play-tests in one location ensuring everyone uses similar computers or to expand on the games data collection to include the frame rate or the questionnaire to include player's computer specification.

D. Visualisation

The visualisation itself is also potentially an issue as many players did not know what it was or they assumed it was part of the game's aesthetic and ignored it. However, the visualisation could still affect the player even if they do not understand it. Many participants suggested they do not understand the RRT visualisation but the results still showed their exploration was affected.

As many players did not understand the visualisation it suggests that the questionnaire should have explained it to some extent to get better answers. This also combined with the issue of self-report bias which is looked at in Section VII-E.

While the RRT visualisation worked as intended, in future work an area to explore would be increasing the RRT size. In this experiment, the RRT was resource intensive so in future work the RRT could be optimised more and then the RRT could be increased in size and clustering could be used to filter the tree.

E. Collected Data

The data collected from the game was usable but it could be improved to get more data that would allow for more in-depth analysis. For example, currently the game only records how many doors the player walks through, an improvement on this would be to record what doors they walked through to create heat maps and perform more statistical analysis to see whether players explore differently in the different game variations or if the players died in similar locations.

Another issue with the collected data is that the questionnaire relied on players being honest. There could have been self-report bias where players gave different answers. For example, players were asked if they understood the visualisation and they are unlikely to admit to not understanding. This could be addressed in future work by relying more on in-game data and wording the questions differently.

F. Random Nature of RRT

As RRT randomly selects where to place each node each player would have a different RRT. This affects the enemy movement as it leads them to move in different ways. Also, some areas had clutter objects such as crates in them, if the path went through the crates they could be knocked over causing the enemy to have to move a different way and affecting how the player can move.

While A* always find the shortest path between two points it is very unlikely that players and enemies will be in the exact same spot and thus unlikely that players will get the same path from A*. As both versions have the potential to give different paths each time this is not currently a major issue.

VIII. Future Work

The most unexpected significant result found in Section VI is the NavMesh visualisation decreasing the percent of the level explored. Future research on this subject could research that in more detail to find a more definite reason for why that happened.

Also, as Section VII mentions, there were issues with the game used in this study, as future research would require a different or more complete game it could also give interesting data to look at pathfinding in more complex environments and other game genres such as stealth games.

Another area that could be explored in future work is the RRT visualisation and variations of it. As the use of RRT in this study had numerous small RRTs with less than 500 nodes each, another area to research could be comparing having an RRT per room to an RRT per level or varying the number of nodes in the RRT. Also for larger RRT variations clustering methods such as the one mentioned in Section II-C that Bauer and Popovic use could be used.

IX. Conclusion

While the use of RRT is more frequent in robotics than digital games its use was feasible in this project. Its past use in game design tools and the results gathered here show that it can be used in games. Previous studies have looked at way-finding and player exploration and suggest that environmental factors in digital environments do have some effect on player exploration. Similarly, this study found player exploration changed between games variations.

This study found a number of significant results which suggest that the pathfinding method and visualisation do affect player exploration in a game level. Like many of the papers reviewed in Section II this could be applied in game design. Level designers could take it into account what type of path-finding will give the result they want as different results may be wanted for different genres or even different sections of the game.

In this study visualising the path-finding to the player often resulted in the player being unsure of what the visualisation was so more research may be required to use the visualisations in video games rather than just designing them.

REFERENCES

[1] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, 2000.

[2] Z. A. Algfoor, M. S. Sunar, and H. Kolivand, "A comprehensive study on pathfinding techniques for robotics and video games," *Int. J. Comput. Games Technol.*, vol. 2015, pp. 7:7–7:7, Jan. 2015.

[3] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, 1968.

[4] L. Alsed, "A* algorithm pseudocode," [Online]. Available: http://mat.uab.cat/ alseda/MasterOpt/AStar-Algorithm.pdf, [Accessed: 11-Nov-2017].

[5] A. Nash, K. Daniel, S. Koenig, and A. Feiner, "Theta*: Any-angle path planning on grids," in *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2*, ser. AAAI'07. AAAI Press, 2007, pp. 1177–1183.

[6] E. R. Firmansyah, S. U. Masruroh, and F. Fahrianto, "Comparative analysis of a* and basic theta* algorithm in android-based pathfinding games," in *2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M)*, 2016.

[7] J. Hu, W. gen Wan, and X. Yu, "A pathfinding algorithm in real-time strategy game based on Unity3D," in *2012 International Conference on Audio, Language and Image Processing*, July 2012, pp. 1159–1162.

[8] M. Wang and H. Lu, "Research on algorithm of intelligent 3d path finding in game development," in *Industrial Control and Electronics Engineering (ICICEE), 2012 International Conference on*. IEEE, 2012, pp. 1738–1742.

[9] J. Tremblay, A. Borodovski, and C. Verbrugge, "I can jump! exploring search algorithms for simulating platformer players," in *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2014.

[10] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Dept., Iowa State University, Tech. Rep., 1998.

[11] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 1478–1483.

[12] S. Karaman, "Incremental sampling-based algorithms for optimal motion planning," *Robotics Science and Systems VI*, vol. 104, 2010.

[13] A. W. Bauer and Z. Popovic, "RRT-based game level analysis, visualization, and visual refinement." in *AIIDE*, 2012.

[14] R. Haworth, S. S. T. Bostani, and K. Sedig, "Visualizing decision trees in games to support children's analytic reasoning: Any negative effects on gameplay?" *Int. J. Comput. Games Technol.*, vol. 2010, pp. 3:1–3:11, Jan. 2010.

[15] S. M. Van Dongen, "Graph clustering by flow simulation," Ph.D. dissertation, Utrecht University, 2001.

[16] M. R. F. Mendona, H. S. Bernardino, and R. F. Neto, "Stealthy path planning using navigation meshes," in *2015 Brazilian Conference on Intelligent Systems (BRACIS)*, Nov 2015, pp. 31–36.

[17] J. Tremblay, P. A. Torres, N. Rikovitch, and C. Verbrugge, "An exploration tool for predicting stealthy behaviour," *IDP*, vol. 13, 2013.

[18] "Metal Gear Solid," Konami, 1998.

[19] D. Isla, "Third Eye Crime: building a stealth game around occupancy maps," in *Proceedings of the Ninth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, ser. AIIDE'13. AAAI Press, 2014, pp. 206–206.

[20] "Third Eye Crime," Steam, Moonshot Games, 2014.

[21] C. Miles and S. J. Louis, "Towards the co-evolution of influence map tree based strategy game players," in *2006 IEEE Symposium on Computational Intelligence and Games*, May 2006, pp. 75–82.

[22] M. Treanor, A. Zook, M. P. Eladhari, J. Togelius, G. Smith, M. Cook, T. Thompson, B. Magerko, J. Levine, and A. Smith, "AI-based game design patterns," in *In Proceedings of the 10 International Conference on Foundations of Digital Games, FDG 2015. Society for the Advancement of the Science of Digital Games*, 2015, pp. 5–6.

[23] "Alien Isolation," Steam, Creative Assembly, 2014.

[24] M. J. Nelson, "Game metrics without players: Strategies for understanding game artifacts," in *Proceedings of the 19th AIIDE Conference on Artificial Intelligence in the Game Design Process*, ser. AIIDE'11-19. AAAI Press, 2011, pp. 14–18.

[25] C. Si, Y. Pisan, C. T. Tan, and S. Shen, "An initial understanding of how game users explore virtual environments," *Entertainment Computing*, vol. 19, pp. 13–27, 2017.

[26] F. Bacim, A. Trombetta, R. Rieder, and M. Pinho, "Poster: Evaluation of wayfinding aid techniques in multi-level virtual environments," in *2008 IEEE Symposium on 3D User Interfaces*, March 2008, pp. 143–144.

[27] D. Moura and L. Bartram, "Investigating players' responses to wayfinding cues in 3d video games," in *Proceedings of the extended abstracts of the 32nd annual ACM conference on Human factors in computing systems*. ACM, 2014, pp. 1513–1518.

[28] C. Nutt, "The undying allure of the metroidvania," [Online]. Available: http://www.gamasutra.com/view/news/236410, [Accessed: 01-May-2018].

[29] A. I. Nordin, A. Denisova, and P. Cairns, "Too many questionnaires: measuring player experience whilst playing digital games," in *Seventh York Doctoral Symposium on Computer Science & Electronics*, vol. 69, 2014.

[30] A. Denisova, A. I. Nordin, and P. Cairns, "The convergence of player experience questionnaires," in *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*, ser. CHI PLAY '16. New York, NY, USA: ACM, 2016, pp. 33–37.

[31] C. Jennett, A. L. Cox, P. Cairns, S. Dhoparee, A. Epps, T. Tijs, and A. Walton, "Measuring and defining the experience of immersion in games," *Int. J. Hum.-Comput. Stud.*, vol. 66, no. 9, pp. 641–661, Sep. 2008.

[32] N. Yee, "Motivations of play in online games," *CyberPsychology and Behavior*, vol. 9, no. 6, pp. 772–775, 2006.

[33] N. Yee, N. Ducheneaut, and L. Nelson, "Online gaming motivations scale: Development and validation," in *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 2012, pp. 2803–2806.

[34] S. I. Donaldson and E. J. Grant-Vallone, "Understanding self-report bias in organizational behavior research," *Journal of business and Psychology*, vol. 17, no. 2, pp. 245–260, 2002.

[35] R. Bartle, "Hearts, clubs, diamonds, spades: Players who suit muds," *Journal of MUD research*, vol. 1, no. 1, p. 19, 1996.

[36] P. Isaias and T. Issa, "Introduction to information systems models and methodologies," in *High Level Models and Methodologies for Information Systems*. Springer, 2015, pp. 1–19.

APPENDIX A
ACKNOWLEDGEMENTS

I would like to thank my project supervisor Dr. Ed Powley whose guidance and assistance were invaluable while working this project. I would also like to thank Bears are OP for allowing me to use their game in my playtesting and all the playtesters who took part in the study. Finally, I would like to thank the third year BSc students whose peer reviews and support I greatly appreciated throughout this project.

APPENDIX B
FIRST APPENDIX: REFLECTIVE REPORT

While I am pleased with the paper I produced, there are many parts of this paper, the software and the software development process that I could have improved on. While I was slightly behind schedule on the written part of my dissertation I believe my main weaknesses during this project lay in the software and playtesting side of the project.

A. Play Testing

The first issues were with play-testing, I underestimated how long this would take and left it too late. This lead to me struggling to get enough participants and resorting to getting more online. Online participants were not ideal as I had no control over what computer they used or whether the data they gave was legitimate. One way to address this in the future would be to run a larger pilot study. In this paper, the pilot study consisted of 4 players, one for each variation. These all took around 20 minutes however in the actual study player took up to an hour.

Another issue was that I found it hard to get people to participate in the study, this was partially due to me leaving the play-testing too late so many students were working on assignments and not willing to do the play-test. Again this could be solved by starting the playtesting earlier or designing a shorter game so participants would not have to devote so much time to the play-test.

B. Data Collection

The data collection using Google Forms and exporting from Unity worked well, the data was gathered and easy to analyse. The main issue was that the two datasets were separate so when analysing the data they had to be combined by matching up the player numbers on each set which was time-consuming. In the future, I would want to combine the two. One way to do this could be making a Unity scene for the questionnaire so all the data is exported from the game into a single .CSV. This removes the need for Google forms and matching player numbers but would mean that playtesting would need to be done in person as otherwise the file produced by the game would need to be emailed to me and there is a chance that the participants could tamper with the data or not send it at all.

As well as improving data collection I would also want to improve what data is collected. In this study, I ran a small pilot study of four participants to see how long play tests take and to see what data I would get. I ran brief statistical tests on this data but not enough. In the future, I would want to do a larger pilot study and run more tests to see if the data exported from the game is enough or useful. For example in this study the game exports what percent of the doors have been walked through, now I have more data it is obvious additional data such as what doors they walked through and when would have been useful.

C. Play Testing Game

One decision I had to make was what game I used for play-testing, my two options were to make a basic Unity game myself or use my COMP340 group game. In the time frame I had I did not think I could make a game that would be engaging enough to investigate player exploration so I choose to use the group game.

A benefit of using the group game is that it gave me more time to work on other aspects of the projects such as the RRT implementation and writing the paper it. The main issue with using the group game is that it was an unfinished game, it had white boxed areas and missing features such as the end of level boss and saving, which meant players had to restart after dying. In this case with the given time frame, I think using the group game was the best option due to the time it saved. However, if I were to research this area again a game designed for the study would be more appropriate. This would allow for more experimentation with the path-finding visualisation and how it is used in a level and would allow for a complete game level to be made as the game would be planned around the study rather than another module's assignment deadlines.

D. Conclusion

In conclusion, given the time frame, available resources and this being my first time attempt at primary research the end result was satisfactory. However, as listed above there are many things I learned through doing this project that would improve any future research I do in this area.