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

Madeleine Kay

***Abstract*—This paper will look at the use of path-finding and *Artificial Intelligence* (AI) visualisation in digital games. Specifically at the use of foregrounding AI and the visualisation of path-finding. While previous papers have researched AI visualisation and how to achieve it there did not appear to be many papers on the field of visualising AI in video 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.**

Analysis of the results showed that...

I. Introduction

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

This paper will look at visualising the tree produced by RRT and the process taken to produce it. The visualisation will be around an enemy NPC. This will allow the players to see where the enemy NPC is going and what their line of sight is.

Implementation of the visualisation will be in a level of a 3D game made in Unity 5.6[1]. Logging tools in the playtesting software will the amount of time the players spends in a level and what percent of the level they explore. Analysis of this data will then determine whether the visualisation has had any effect on how participants explore.

Previous papers have researched visualising *Artificial Intelligence* (AI) and foregrounding AI. However, there is little on what effect this has on how the players explore the game. The research question proposed in this project is: how does visualising RRT pathfinding in an NPC affect how a player explores a game level?

II. Related Work

As the focus of this paper is on path-finding that was the first area researched. The methods looked at were A* path-finding in section II-A and RRT path-finding in section II-B. After that Foregrounding and Visualising AI was researched in section II-D. While there is research on AI visualisation there was a lack of it for digital games and what research was on games often focused on game development and level design as can be seen in section II-D.

This literature review found 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*.

```
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].

Algfoor *et al* [2] surveyed numerous papers on pathfinding. Their focus was 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 surveyed many grid types and gave the advantages of each.

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.

While their focus is on Android games Firmansyah *et al* [6] compared A* with Theta*. They found that performed

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

similarly time wise. However, A* produced a path with fewer nodes expanded and Theta* produced 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. When used in this project the algorithms will only be run once however as A* in 2 dimensions had a 100 percent success rate this should not be an issue.

B. RRT and Path-finding

RRTs are 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.

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

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.

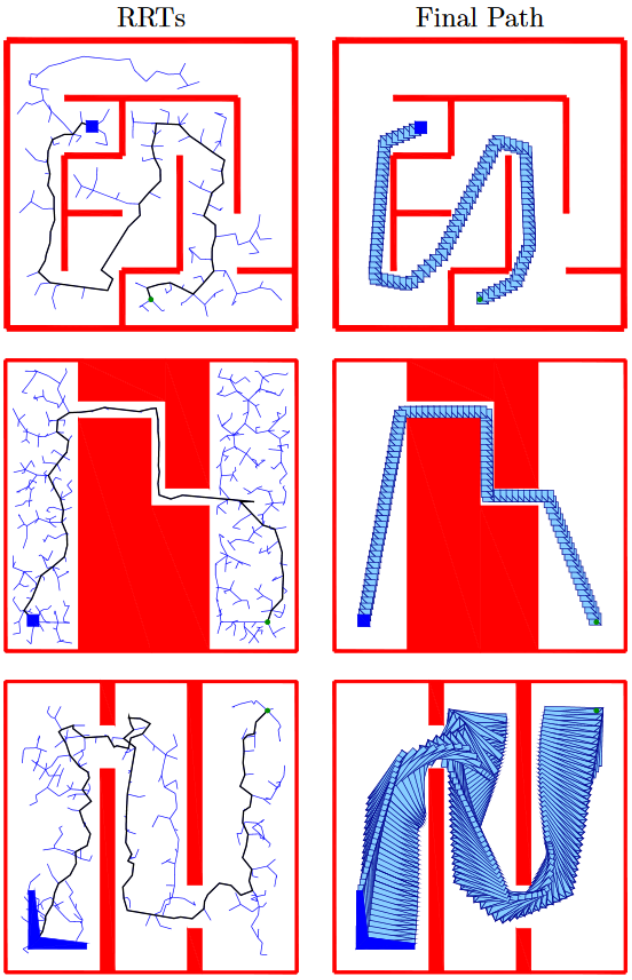

Fig. 2. Kuffner and LaValle's RRT path planner building a tree and plotting a path through it [1].

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]. 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. Path-finding

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

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

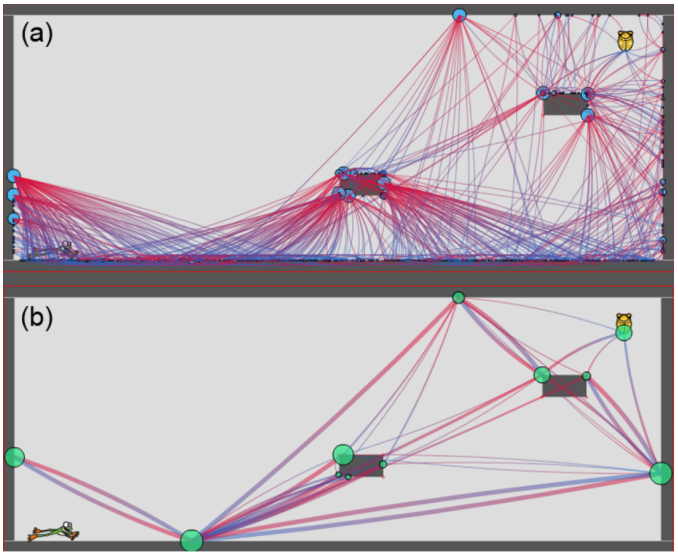

Fig. 3. Bauer *et al*'s[13] graph-based representation of RRT with and without clustering.

the tool once it has been run on a basic level design. The first image, image a, may be difficult to read. So, they make use of Dongen's method for graph clustering to make the output more legible as shown in the second image, image b [13], [15].

The focus of this project is on a similar type of visualisation but running in the game instead of during the game development. Therefore, there is a need for a similar technique to cluster the output of RRT. This should make the visualisation more useful as it should aid the player in a way that they can look at the visualisation and interpret what the NPC is going to do.

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 will use 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 will then be 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.

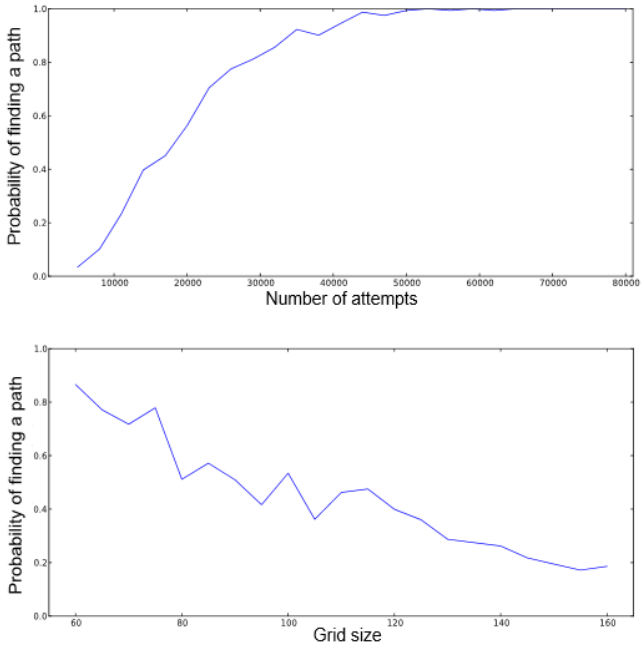

Fig. 4. Performance analysis of Tremblay *et al*'s [17] RRT when running on a Metal Gear Solid level [18].

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]

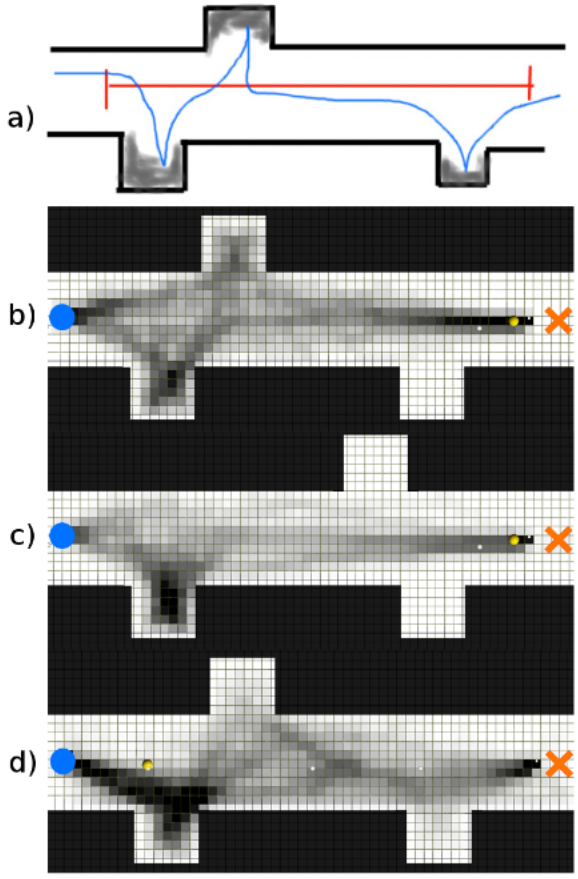

Fig. 5. Heat maps of a single level where the design has been changed each time [17].

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

Section II-D will look 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. 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]. Isla used Occupancy Maps to show where the enemy AI thinks the player currently is. 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* pathfinding 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 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 will inform them of the enemy’s position.

This paper will use the “AI as a villain” pattern as each enemy NPC will have their path-finding visualised around them. The players will have to consider this when exploring a level so they do not get attacked by the enemy. The use of this pattern will also aim to have an NPC that creates an experience rather than one that will always find the player, similar to Alien Isolation [23], [22]. While the player will 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. The game uses Occupancy Maps to display where the enemy thinks the player could be on the map. 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].

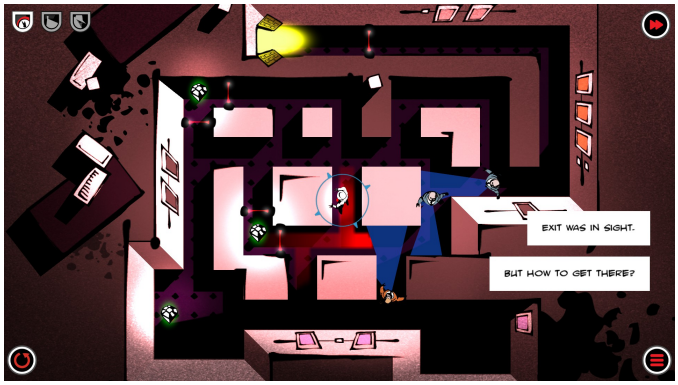

Fig. 6. A screen-shot from Third Eye Crime [20].

This pattern is relevant to this project as the enemy NPC will 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 did 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 will 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 will observe 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 did 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 is what the enemy NPC will be 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 did not look at player exploration the game they used involved a map where participants had to explore a maze. Participants which 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 proposed research question is: how does visualising pathfinding in an NPC affect how players explore a game level? The hypothesis drawn from this question are listed in subsection III-A.

An A priori power analysis was performed to determine how many participants were required for an effect size of 0.4. 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.

A large majority of the players were students on game development-related courses, this gave the issue that they treated the playtest as an evaluation of the game itself and gave feedback on that, not the visualisation. Another potential issue with this is that many students were familiar with Unity and therefore may have been able to guess what the visualisations showed. Another further issue is that some participants may have user tested a variation of the game before and therefore known parts of the level layouts which could influence how much they explored.

Update when playtesting complete.

The rest of the playtests were done online through Reddit and Twitter. The subreddits it was posted on were: /r/SampleSize, /r/UniUk, r/PlayMyGame and r/PlayTesters. These subreddits all relate to gaining participants and playtesters. This could have affected the results as only a certain type of person would go online looking for surveys and playtests to complete. Again some of the subreddits listed were related to game development and therefore the feedback given was related to the state of the game, not the pathfinding.

A. Hypothesis:

Table I shows the hypothesis being tested in this experiment. The methodology that will be used to test these hypotheses will be playtesting and a questionnaire. This will require human participants for both parts. Ethics approval for this

TABLE I
HYPOTHESIS

	Hypothesis	Null Hypothesis
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.
2	Visualising RRT path-finding has an effect on the percent of the level the participant explores.	Visualising RRT path-finding has no effect on the percent of the level the participant explores.
3	Visualising Unity navmeshes has an effect on the percent of the level the participant explores.	Visualising Unity navmeshes has no effect on the percent of the level the participant explores
4	Visualising RRT has an effect on the length of time the participant spends in the level.	Visualising RRT has no effect on the length of time the participant spends in the level.
5	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.
6	In comparison to RRT visualisation visualising Unity navmeshes has an effect on the percent of the level the participant explores.	In comparison to RRT visualisation visualising Unity navmeshes has no effect on the percent of the level the participant explores.
7	In comparison to RRT visualisation visualising Unity navmeshes has an effect on length of time the participant spends in the level.	In comparison to RRT visualisation visualising Unity navmeshes has an effect on length of time the participant spends in the level.
8	The visualisation of Unity navmeshes is comprehensible to players.	The visualisation of Unity navmeshes is not comprehensible to players.
9	The visualisation of RRT path-finding is comprehensible to players.	The visualisation of RRT path-finding is not comprehensible to players.
10	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 to Unity navmesh visualisation to players.

methodology was gained from Falmouth Universitys Research Ethics Board.

The game the pathfinding variations will be tested in is 'Gates of Amenti' a 3D Metroidvania game which has a 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.

Reference about Metroidvanias?

Players will be given one of the game variations to play and then asked to complete a questionnaire on their experience. The game will also log the player's position at regular intervals to calculate what percent of the level they explore. How long a participant spends exploring the level will also be logged.

B. Play-testing

More details on game used. Justify GoA use

1) Play-test Variations: 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.

TABLE II
PLAY-TEST VARIATIONS

	RRT	Unity Navmesh
Hidden Path-finding	Variation 1	Variation 3
Visualised Path-finding	Variation 2	Variation 4

During the playtests players were randomly assigned one of the four finding 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 was calculated by recording how many doors the player walked 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 R[2].

Detail stats tests done.

After completing a playtest 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 playtesting, participants were also asked to fill out a questionnaire on the game online using Google Forms. Nordin *et al* [28] say that questionnaires are vital for understanding how players feel when playing digital games [28], [29]. They can also help give uniformity and consistency to the data gathered from participants [29].

Questionnaires are also beneficial as they can prompt players to give answers they may not have given spontaneously. The two options here are to either create a questionnaire specifically for the experiment or to use an existing one. There are a wide variety of questionnaires that already exist to measure players experiences in play testing [28], [30]. 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 [28], [30], [31], [32].

However, Nordin *et al* [28] 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.

[2]R Available: https://www.r-project.org/

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 potential issue with the questionnaire is self-report bias. Some of the hypothesis 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.

FIND REF

Most of the hypotheses required quantitative data from the playtests. However, some required qualitative data so a questionnaire was 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 [33] was looked at, this taxonomy appears to be widely used but is not validated. Instead Yee's player taxonomy Yee's questionnaire for player taxonomies will be used in combination with questions based on the hypotheses that require qualitative data.

New stuff: Looked at Bartle not validated [33] used [31], [32]

IV. Software Development

The software used to develop the playtesting software was Unity 5.6.3f1[3] and Visual Studio Community 2017 15.5.4[4].

The analysis was carried out in R version 3.4 in R Studio.

The base game 'Gates of Amenti' was developed with a group using Agile. The game was still in development while being adapted to be used for playtesting and was therefore unfinished, this is a potential issue looked at in section VII. The additions for use in the playtests for this paper used the prototyping life cycle method [34]. The basic versions of the data collection such as the timer and a script to generate the RRT were added early on and then analysed and revised until it met the requirements for the playtesting.

A. Software Testing

To test the playtesting software was working as required a simple bot was developed. All the bot needed to do was move forward to test whether the doors registered when walked though and then whether the timer and data export worked.

1) Doors: The first bot was to test whether the doors recorded whether the player had moved through them. Whether a door has been walked through it used to determine what percent of the level the player explored. The test was a bot moving through the door and checking that the GameState class recorded it.

2) Timer and Data Export: The second bot was 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 .

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

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

```
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.

	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 was whether the path-finding visualisation worked as intended. This did not require a bot it can be seen 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.

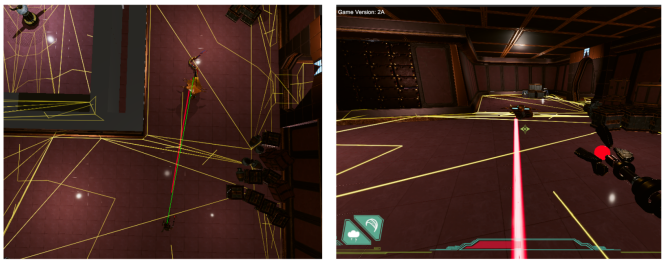

Fig. 9. The navmesh visualisations shown both in editor and in scene.

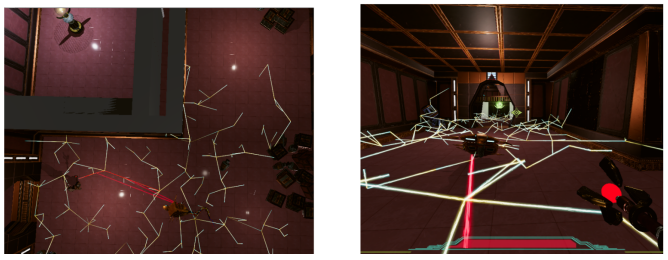

Fig. 10. The RRT visualisations shown both in editor and in scene.

V. Data Collection and Removal

All the data used either came from the game's exported data or from the questionnaire. When the player dies in-game, the game resets so a number of results had 0 percent explored a low play times. As these were not proper playtests I removed them from the results set being analysed. Many players also died within the first few minutes of playing, giving a playtest with a low time and exploration amount, these results were also removed for the analysis.

VI. Analysis and Interpretation

Table III shows the correlations and P-Values calculated from the results using R. In a majority of cases the null hypothesis was proven. However, two cases had a P-Value of less than 0.04 and are therefore statistically significant.

The first case was a small positive correlation of 0.2807 between the percent of the level explored when RRT pathfinding is visualised. This suggests that players explored more in the RRT variation with visualisation. However, there was no correlation between RRT visualisation and the time spent in the level...

Some participants said that they thought the RRT was leading somewhere and followed it through the level/went to it when they saw it

However

The second case was a negative correlation of 0.4894 between the percent explored and the NavMesh pathfinding being visualised. This suggests that player explored less when the NavMesh was visualised.

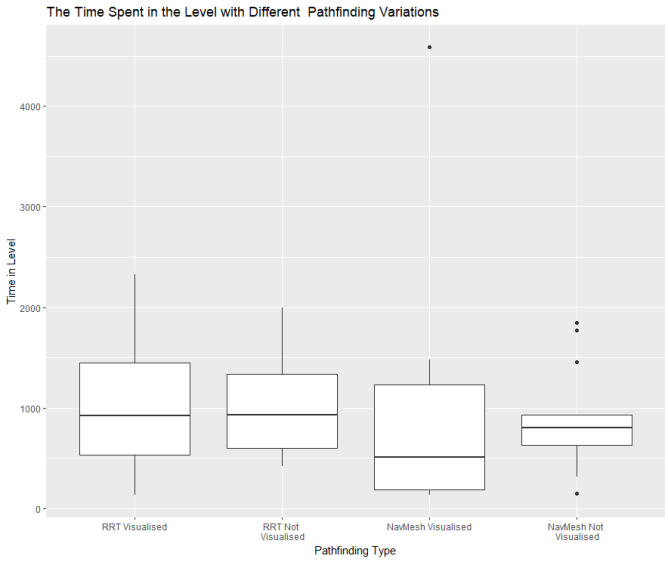

Fig. 11. Box and Whisker plot of the time spent in the level with the 4 game variations.

A. Null Hypotheses

Section on nulls

VII. Potential Issues and Future Work

Finish off

While the methodology of the study ..... there were many areas that could be improved in future work to improve the quality of the data collected. This section will identify a number of weaknesses in this study and how they could be improved on in future work.

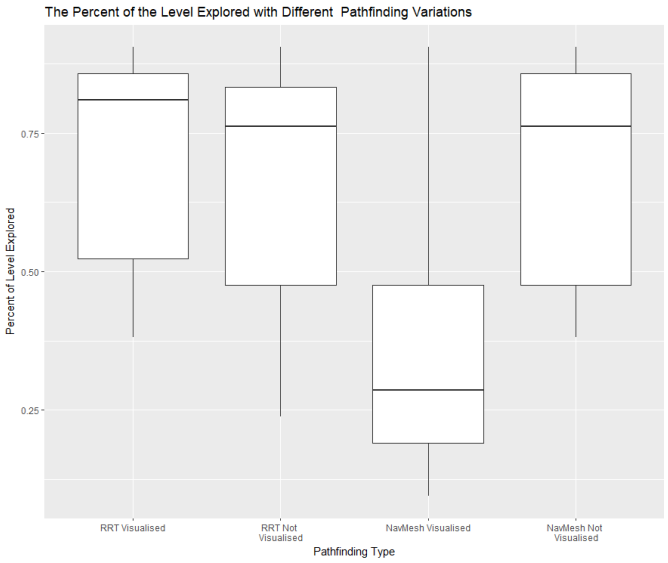

Fig. 12. Box and Whisker plot of the percent of the level explored with the 4 game variations.

A. Playtesting Game

A first potential issue with the playtesting is that the game was not designed specifically for 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 playtesting or altered more to suit the study.

B. Participants

Another issue with using the game 'Gates of Amenti' is that many of the play tests used game development students who had playtested other levels of the game or seen the game trailer during the university show and tell day. This meant that they could have been looking for things they had seen in the trailer as opposed to exploring or being affected by the visualisation.

In future work the player would also be selected randomly and from a larger pool of people. In this study, a majority of players were students studying a game development course.

C. Performance Issues

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

Get machine specs?

TABLE III
MY CAPTION

Independent Variable	Dependent Variable	Correlation	T-Test - P Values
Visualised Pathfinding	Percent Explored	-0.1766742	0.2009
RRT Used	Percent Explored	0.1697141	0.4229
Visualised RRT	Percent Explored	0.2807048	0.04425
Visualised NavMesh	Percent Explored	-0.4893995	0.01176
Visualised Pathfinding	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 Pathfinding	Understandable Visuals	0.1627247	0.2731
RRT Used	Understandable Visuals	0.1910156	0.3341
Pathfinding Visualised	Logical Enemy Movement	-0.2728879	0.06224
RRT Used	Logical Enemy Movement	-0.01419318	0.9234

D. Visualisation

The visualisation itself was also an issue as many players did not know what it was or they assumed it was part of the games aesthetic and ignored it.

As players did not understand the visualisation this also suggests the questionnaire should have explained it to some extent to get better answers. This also combined with the issue of self-report bias many players did not know

RRT needed to be better performance wise so it could be bigger and then have clustering to make more usable

E. Collected Data

While the data collected from the game did work it could be refined to get more data that would allow for better analysis. For example, currently the game only records how many doors the player walks through, an improvement on this would be to record what door they walk through to create heat maps and perform more statistical analysis to see whether players explore differently in the different game variations. * Self Report Bias

VIII. CONCLUSION

NEEDS UPDATING BASED OFF DATA

In conclusion, while the use of RRT is more frequent in robotics than digital games its use appears feasible in this project. Its past use in game design tools shows that clustering can make the output understandable to the user. However, the need of optimisations such as RRT* or the use of A* may arise to produce a shorter path or to find any path.

Visualisation and foregrounding of AI have been successfully used before suggesting that it can be used here. Finally, previous studies have looked at way-finding and player exploration. These suggest that environmental factors in digital environments do have some effect on player exploration.

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] 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.
[29] 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.
[30] 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.
[31] N. Yee, "Motivations of play in online games," *CyberPsychology and Behavior*, vol. 9, no. 6, pp. 772–775, 2006.
[32] 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.
[33] R. Bartle, "Hearts, clubs, diamonds, spades: Players who suit muds," *Journal of MUD research*, vol. 1, no. 1, p. 19, 1996.
[34] 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.