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

1507866

***Abstract*—This paper will look at path-finding and *Artificial Intelligence* (AI) in digital games. Specifically at the use of foregrounding AI and the visualisation of path-finding. Previous papers have researched AI visualisation in games and how to achieve it. There has also been research into the visualisation of path-finding. However, the focus was most commonly on the use of visualisation in game design.**

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 [2]. Logging tools in the play testing software will record 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 affect on how participants explore.

Previous papers have researched visualising *Artifical 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 path finding in a NPC affect how a player explores a game level?

II. Related Work

A. A* Pathfinding

A* path-finding is widely used in both robotics and digital games [3]. In games A* appears to be the most commonly used path-finding algorithm [3].

Hart *et al* [4] 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* [3] surveyed numerous papers on path finding. Their focus was on the use of different grid shapes in path-finding and the numerous algorithms available [3]. 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.

```
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 [4] [5].

Nash *et al* [6] say that A* can not 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* [6], [7]. 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* [7] compared A* with Theta*. They found that performed similarly time wise. However, A* produced a path with less nodes expanded and Theta* produced a shorter path.

Hu *et al* [8]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 [9] 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* [10] 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 game play 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 Pathfinding

RRTs are a search method used more in robotics than in digital games [11], [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.

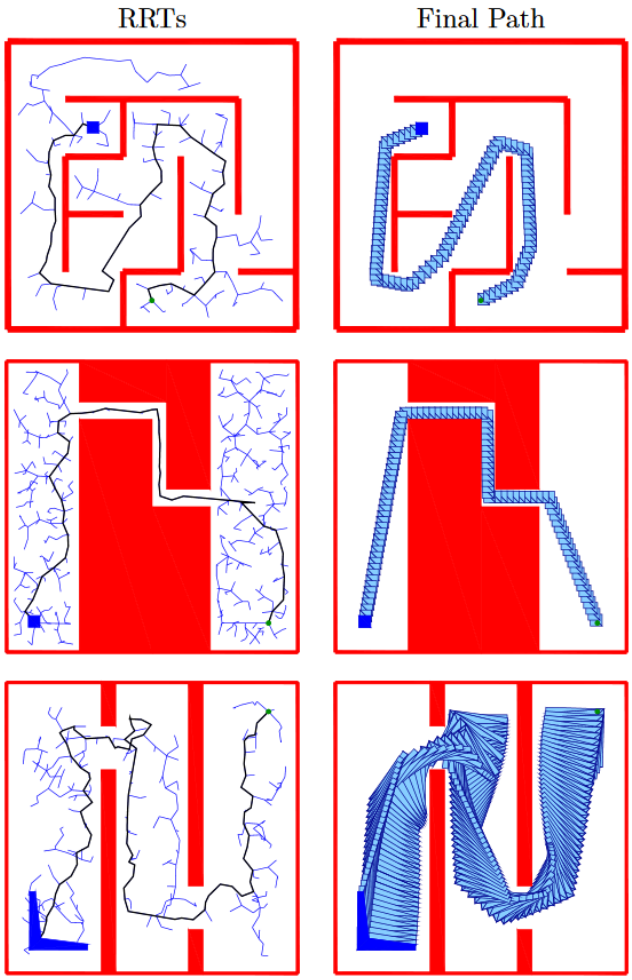

Fig. 2. Kuffner and LaValle's RRT path planner building a tree and plotting a path through it [1].

The process for RRT involves the random placement of nodes. A parent is then selected by finding the closest pre-existing node [1]. Figure 3 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], [12]. Karaman and Sertac [13] say that the chance of RRT finding an optimal path is very unlikely [13], [10]. 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.

```
BUILD_RRT(q_init)
1  T.init(q_init);
2  for k = 1 to K do
3      q_rand ← RANDOM_CONFIG();
4      EXTEND(T, q_rand);
5  Return T
```

```
EXTEND(T, q)
1  q_near ← NEAREST_NEIGHBOR(q, T);
2  if NEW_CONFIG(q, q_near, q_new) then
3      T.add_vertex(q_new);
4      T.add_edge(q_near, q_new);
5      if q_new = q then
6          Return Reached;
7      else
8          Return Advanced;
9  Return Trapped;
```

Fig. 3. Kuffner and LaValle's RRT algorithm pseudo-code for RRT construction [1].

C. Path-finding

Bauer and Popovic [14] use RRT for level design in digital games. Like other papers mentioned in the Foregrounding and Visualising AI section they visualise the data to aid users [14], [15]. This data visualisation is for use in game development to aid game developers or to analyse procedurally generated levels.

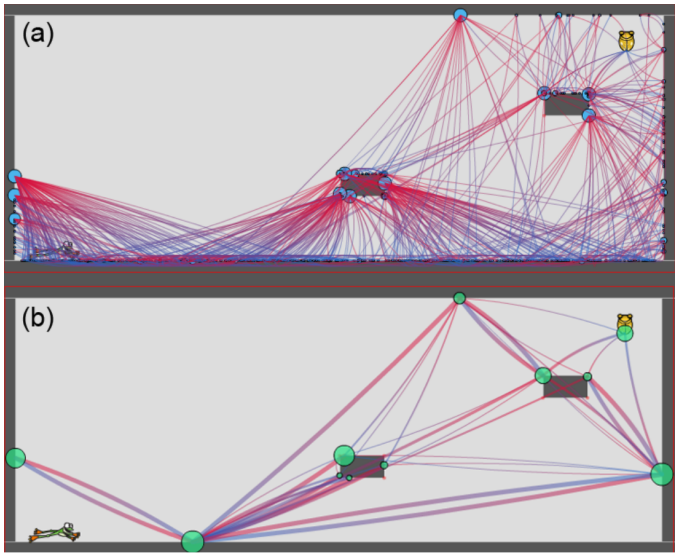

Fig. 4. Bauer *et al*'s[14] graph-based representation of RRT with and without clustering.

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 4 shows the output of 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 [14], [16].

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* [17] 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 [13], [17]. 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* [17] 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* [18], like Bauer and Popovic [14], 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 [18]. Similar to Mendona *et al* [17], they focus on designing stealth games and finding stealth orientated paths in the game levels [18]. Figure 6 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 [18]. 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 [18].

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 maybe 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 5. This suggests that a potential issue with RRT is that it may not always find a path.

Tremblay *et al* [10] 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 the A* Path-finding section they look at the use of A*, MCTS and RRT to visualise player behaviour in platform games similar to Bauer [10] [14]. RRT is limited

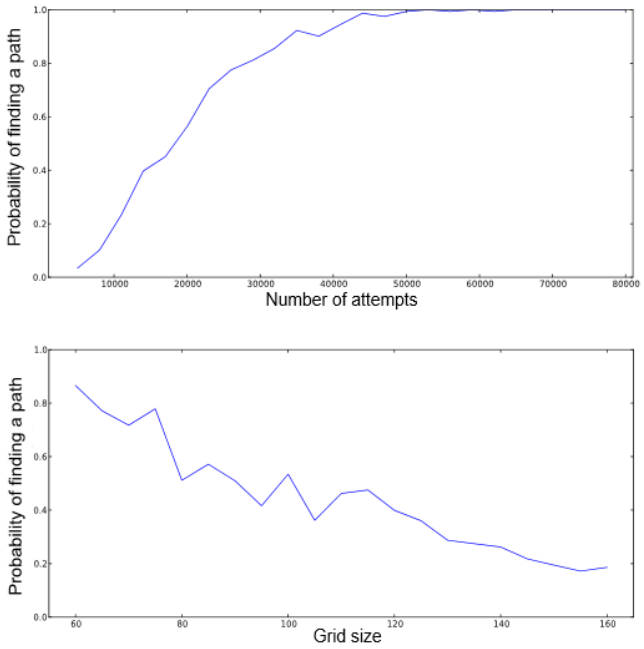

Fig. 5. Performance analysis of Tremblay *et al*'s [18] RRT when running on a Metal Gear Solid level [19].

to drawing straight lines between the nodes, as straight lines are not always possible in RRT Tremblay *et al* [10] 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 over 60 percent. However, the run times for this combination varied. Both of Tremblay *et al*'s [10], [18] 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 run time 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.

The Foregrounding and Visualising AI section 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 [20], [21]. Isla [20] 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 [20], [22]. 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 [22] 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.

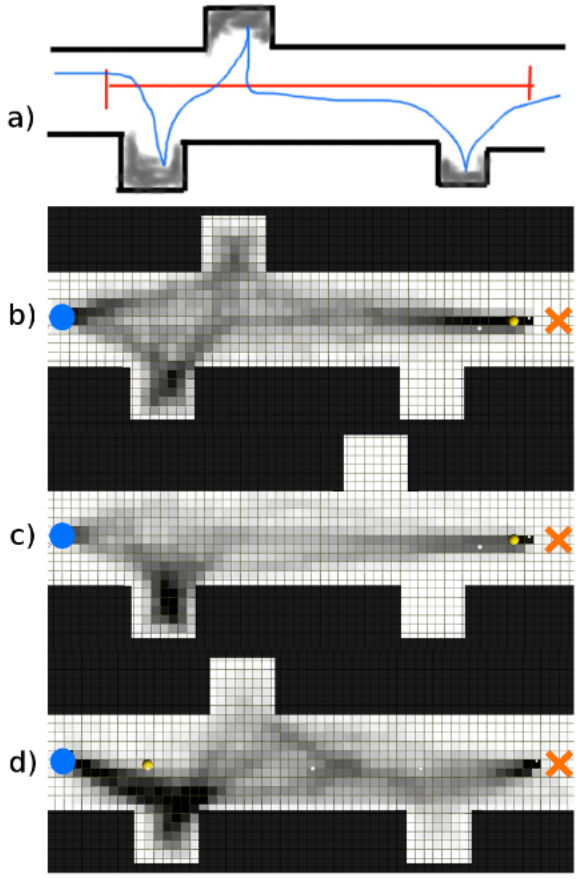

Fig. 6. Heat maps of a single level where the design has been changed each time [18].

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* [23] say that often the design of AI in games is to fit the game and complement game play. These AI are supporting the game play rather than being central to it.

Treanor *et al* [23] 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 [23], [24]. 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 a NPC that creates an experience rather than one that will always find the player, similar to Alien Isolation [24], [23]. 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 [23]. Most games hide this from the player but this design pattern visualises it making it mechanic. The example given by Treanor *et al* [23] is the game Third Eye Crime. Third Eye Crime is a game that follows the "AI is Visualised" design pattern [20], [21]. The game uses probabilistic object tracking through Occupancy Maps. The game uses Occupancy Maps to display where the enemy thinks the player could be in the map. As the enemy moves around the map it removes areas where the player is not from the Occupancy Map [20]. 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 [20], [21].

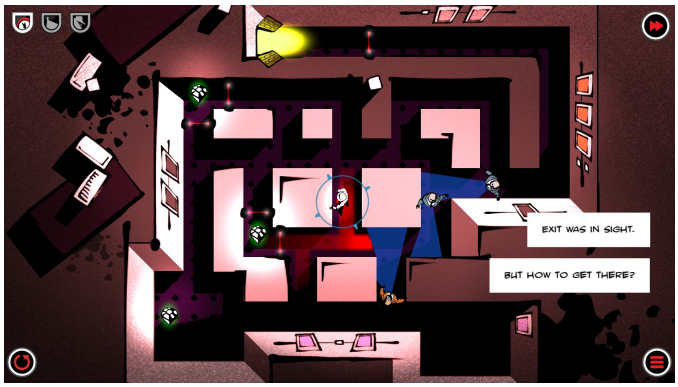

Fig. 7. A screen-shot from Third Eye Crime [21].

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* [15] 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 game play 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* [15] 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 [20] visualised Occupancy Maps are in the game Third Eye Crime which is available for purchase on Steam, IOS and Android [20], [21].

Like Haworth *et al* [15], Bauer *et al* [14] also research visualising tree structures [14]. 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 [25], [14], [18], [10].

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 [26], [27]. 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* [26] 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 [28] 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 off the path. This could then subtly guide the player through the game.

Si *et al* [26] 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* [26] 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 the Foregrounding and Visualising AI section was Haworth *et al*'s [15] 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 [28] suggested way-finding alone was not enough to guide a player this suggests that visualisation could aid the exploration process [15].

III. Methodology

The methodology that will be used to test the given hypotheses will be play testing and a questionnaire. This will require human participants for both parts.

A. Hypothesis:

Null Hypothesis: Visualising RRT path-finding has no effect on the percent of the level the participant explores.
Hypothesis 1: Visualising RRT significantly affects the percent of the level the participant explores.
Hypothesis 2: Visualising RRT significantly affects the length of time the participant spends in the level.
Hypothesis 3: Visualising Unity navmeshes affects the percent of the level the participant explores compared to RRT.
Hypothesis 4: Visualising Unity navmeshes are not comprehensible to players.
Hypothesis 5: Visualising RRT is comprehensible to players.

The game the enemy NPC will be tested in is a 3D metroidvania game which has a focus on exploration. A metroidvania game is an action-adventure game with a focus on exploration to discover new areas and power ups. 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. Playtest Variations

There will be multiple variations of the game to test the different hypothesis.

1) Control: Unity Navmesh: The first variation will be the control version of the game. This version will have no visualisation on the enemy NPC's path-finding. The enemies in the game will use the default Unity navmeshes.

2) Unity Navmesh Visualised: The second variation will use Unity's built in navmeshes and navmesh agents for path-finding. This will be visualised on the floor of the level around the enemy NPC.

3) RRT Path-finding Visualised: The third variation will have visualised path-finding. The path-finding will use RRT and the player will be able to see the tree around the enemy NPC.

4) RRT path-finding: The final version will also use RRT path-finding but there will no visualisation around the enemy NPC.

A-B testing will be used on participants. A-B testing is where each participant will be assigned a version of the game to play [29]. In this case to prevent them from guessing what the goal of the play test is.

The software will log the player's position in the level throughout their play test. This data will be exported in a .CSV every second. Varying the export rate is unlikely to provide data of interest as the focus is on time spent in a level and how much was explored.

The exported data will then be analysed using R [30]. Wallner says heat maps are useful as they are easy to generate and easy to discern patterns from [31]. Tremblay *et al* [10] also used heat maps on their RRT visualisations. They used them as part of their tool for level designers to use to see where players could go and the effects of them altering the level design which can be seen in Figure 6. Here they will

only be used for analysis and to assist in finding noticeable patterns.

Other statistical analysis in R will be generated to support the heat maps. T-Tests will be performed to compare the data on the null hypothesis and the sets of data on hypotheses 1 to 3.

After completing a play test of the game participants will be asked to complete a questionnaire on their experience.

C. Questionnaire

Alongside play testing participants will also be asked to fill out a questionnaire on the game online using Google Forms. Nordin *et al* [32] say that questionnaires are vital for understanding how players feel when playing digital games [32], [33]. They can also help give uniformity and consistency to the data gathered from participants [33].

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 [32], [34]. 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 [32], [34].

However, Nordin *et al* [32] say there are many issues with using 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.

Most of the hypotheses here can be tested using the quantitative from the play tests. However, hypotheses 4 and 5 require qualitative data so a questionnaire will be necessary. The IEQ tests for player immersion but has questions related to interacting with the game world and causes of frustration in it that may relate to the visualisation of path finding. Therefore, relevant questions from the IEQ will be used as well as additional questions based on hypotheses 4 and 5 of this paper.

IV. Conclusion

In conclusion, while the use of RRTs 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 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] Unity Technologies, "Unity," [Online]. Available: https://unity3d.com/, [Accessed: 11-Nov-2017].

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

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

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

[15] 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.

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

[17] 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.

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

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

[20] 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.

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

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.
[27] 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.
[28] 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.
[29] P. Hynninen and M. Kauppinen, "A/b testing: A promising tool for customer value evaluation," in *2014 IEEE 1st International Workshop on Requirements Engineering and Testing (RET)*, Aug 2014, pp. 16–17.
[30] R Core Team, "The r project," [Online]. Available: https://www.r-project.org/, [Accessed: 11-Nov-2017].
[31] G. Wallner and S. Kriglstein, "An introduction to gameplay data visualization," in *Game Research Methods*, P. Lankoski and S. Björk, Eds. Pittsburgh, PA, USA: ETC Press, 2015, pp. 231–250.
[32] 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.
[33] 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.
[34] 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.