CT6GAMAI REPORT - 2533 words

[569 words]

Introduction

The first idea of the project was to implement a BeatEmUp game, mainly because is one of my favorite game genres, but also, the development process seemed fun. As time passed, more information about AI in games was acquired and the idea of doing a BeatEmUp game was a bit hard to achieve so the plan changed into making a battle simulator due to high capability of applying approximately the same amount of AI techniques as the previous idea and being more malleable in terms of ways to develop.

This project makes use of embedded Finite States Machines to store the behaviours of the agents and the transitions between each behaviour. A well known game franchise who its most major game components rely on finite state machine, based on "***Tommy Thompson***" in his article for *AI and Games* would be ***Batman Arkham*** (**6**) which uses the finite states machines to build the enemies character both in combat as well as in stealth segments.

Followed by Utility Theory, based on desirability for health on both agents where a formula is used to calculate this desire and based on the returned input, is making the agents deciding if they need health or not. There are obviously more ways of using utility theory in games for example "***Counter Strike 1.6***"[**10**] is using a desire to attack the player as well as desire for getting a weapon.

The next technique used is called Pathfinding which is calculating the shortest path from point A to point B. The way pathfinding works depends on the objective of every game, if the latter implies a game where it needs to find a path to all locations, then Dijkstra's algorithm might be the choice as it expands towards all the nodes in the grid. Another example would be in ***Civilizations***[**12**] moving through plains or desert might cost 1 move point but moving through forest or hills might cost 5 move points. The state of the art of this game's pathfinding system requires the usage of at least A star algorithm, as it uses a heuristic cost which makes the nodes to expand more towards the goal, not on the entire map.

In this project the steering behaviour used is path following which combined with Pathfinding enables the agent the capability to search and follow a path. This steering behaviour works based on waypoints set up along the path. There are also other types of steering behaviours, for example in ***God Of War III[15]***, the normal enemies are using a seek steering behaviour to receive a force that directs the agents towards the player position. Another example would be in “***Need For Speed Hot Pursuit***”[**16**], a pursuit steering behaviour is used to make the police car agent to go after the player.

Lastly, the project is making use of an eye perception which consists of a sphere projected around the agent within a radius. Another example of perception system is based on the smell and is present in “***Far Cry 5***”[**17**] more exactly this type of perception is using the breadcrumbs system and is applied on the dog to enable him with more abilities. Also another very useful perception system, especially for the first person shooter games such as “***Counter Strike Global Offensive*** ”[**18**] is called sound propagation, known as 3D pathfinding, where the agent can hear another player on the map.

Analysis [1459 words]

Finite State Machines

This project makes use of embedded Finite States Machines to store a collection of states for both agents. Tommy Thompson defines Finite States Machine in his article called:” ***AI & Games, Arkham Intelligence****”(***3**) as “*a model commonly used to simulate simple sequential logic*.” While the agent is in one state, he will continue on executing the same action until the circumstances are being changed.

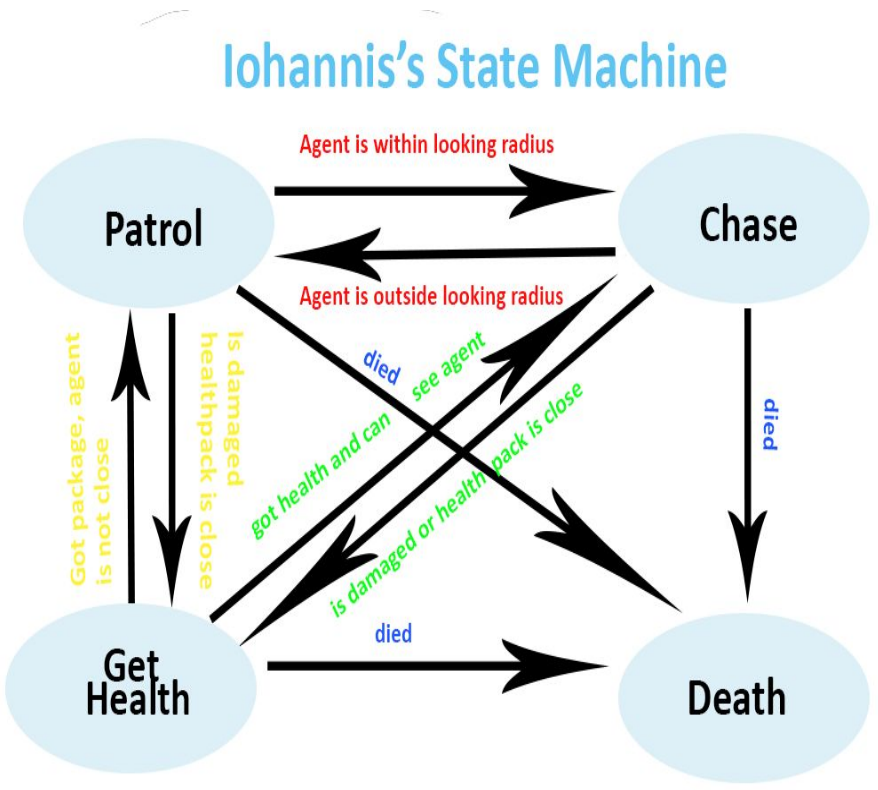

Iohannis's State Machine
Patrol
Chase
Get Health
Death
Agent is within looking radius
Agent is outside looking radius
died
see agent
got health and can
pack is close
is damaged or health
died
died
Got package, agent is not close
is damaged healthpack is close

Fig 1. Iohannis's Finite State Machines
Can be found in Report - Images[FSM]

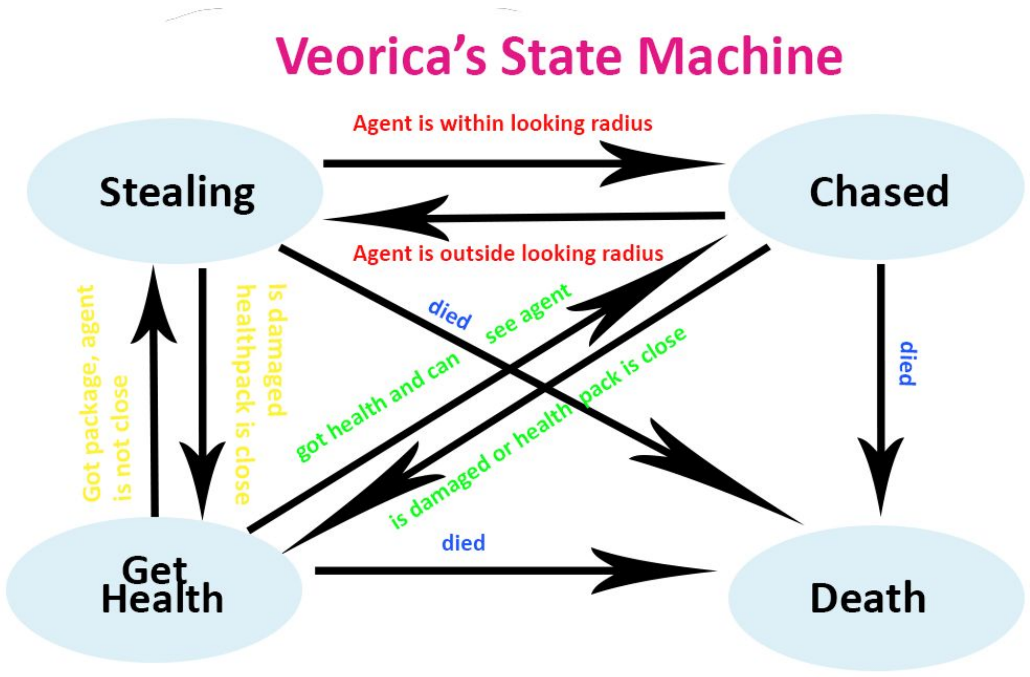


Fig 2. Veorica'sFinite State Machines - Can be found in Report - Images

Finite State Machines were used because by using states, the behaviour is split into chunks (states) making it easier to debug and develop. They do not require a lot of computational power.

In the book called ***Modeling Software with Finite State Machines A Practical Approach(1)***, one of the authors is mentioning: "*One of the most powerful models is a finite state machine (fsm), which is used to describe behavior: what to do in all imaginable situations.*".Is also making easier creating and controlling the execution flow of the game based on what *Fernando Bevilacqua* states in his article called: "***Finite-State Machines: Theory and Implementation****"(2)*: " *finite-state machine is a model used to represent and control execution flow. It is perfect for implementing AI in games, producing great results without a complex code"*

This project makes use of embedded finite state machine, this means that each state decides when is time to change the state, rather than the classic approach where a class named (Brain) is containing all the states and decides to which should change.
The reason for using this method is due to the minimum amount of states that the agents from this project detain and it also seemed easier to implement the state machine using this approach.

On the other hand, in games such as ***Horizon Zero Dawn*** [**19**] where an npc has many states, in this scenario the finite state machine becomes hard to manage and a better approach would be by using behaviour trees which based on ***Chris Simpson*** in his article called: "***Behavior trees for AI: How they work***"[20] for Gamasutra: Using this type of data structure enables the agents the ability to have two states running at the same time as well as having a much more complex and well organized behaviour flow.

Utility Theory-Desirability

In games, utility theory refers to how does an agent chooses an action from a list of strategies. Combined with finite state machines, refers to how does the agent transition between each state.

The agents decide on these transitions based on some input values deducted from the formula used to calculate the desirability.

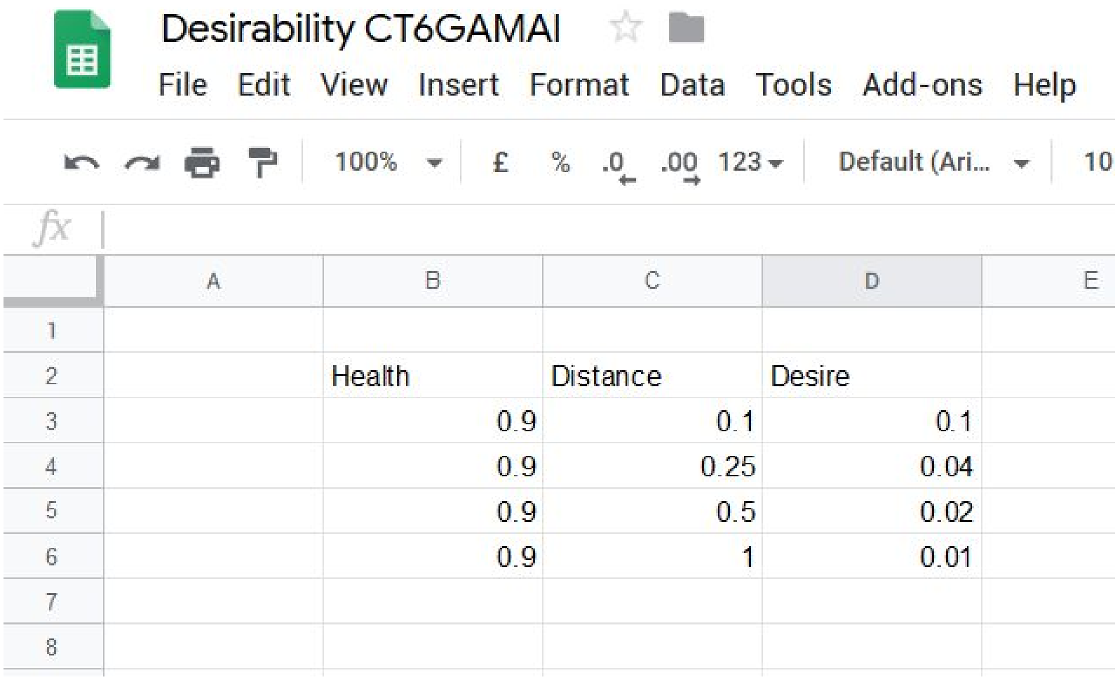
Based on the research deducted by ***Peter H. Farquhar*** called: ***Applications of Utility Theory in Artificial Intelligence Research,*** an early application of such technique was made by ***Arthur Samuel*** in 1959[**28**]: where he developed a checker-playing program which evaluate prospective moves based on the board position

This project makes use of desirability for getting health and is calculated every frame based on health status of the agent and on the distance from the agent to the health pack using this formula:

$$Desirability^{GetHealth} = k * (\frac{1 - HealthStatus}{DistanceToItem(HealthPack)})$$

Fig 3. Health desirability formula[**24**]

And the values were deducted using this sheet:



	A	B	C	D	E
1					
2		Health	Distance	Desire	
3		0.9	0.1	0.1	
4		0.9	0.25	0.04	
5		0.9	0.5	0.02	
6		0.9	1	0.01	
7					
8					

Fig 4. Desirability formula sheet [**25**] Can be found in Report - Images

As it can be seen, the desire is directly proportional with the current health level of the agent and is inversely proportional by the distance from a health pack.
This formula was used because it best suits the requirements of the artefact.

Pathfinding

A Star algorithm was chosen for Pathfinding in this project because is using the G cost(the cumulative cost from the source node), same as the" ***Disjkstra's Alghorithm***"[8], but in addition is using a special cost which makes it unique called heuristic cost(H cost), which is the estimated distance to the Target Node.
The formula used in this project to find this cost is known as "***Manhattan Distance***" **.** Manhattan Distance was used due to it's better adaptability to working with obstacles and also because is more efficient than the *Euclidean Distance* because it doesn't involve quadratic operations and lastly, it requires less computational calculation than *Dijkstra's algorithm* because is using the G cost and the H cost, where the other algorithm is using just the G cost.

On the other hand, in their research entitulated: "*Pathfinding in Games*", "*Adi Botea, Bruno Bouzy, Michael Buro, Christian Bauckhage,* and *Dana Nau*"[**11**] are stating that: "*The ALT admissible heuristic takes the maximum* ***maxl|d(n1, l)−d(n2, l)|*** *over a subset of landmarks*".Where d(n, l) represents the distance from the node n to the landmark l. Essentially, this formula calculates the cost over a subset of landmarks, making it a lot more efficient than the normal heuristic cost due to the low number of nodes generated. Less nodes implies better performance.

The nodes used in this project are represented in fig 6 using cubes. The red zone is marked by the obstacle nodes, while in the grey zone are the irrelevant nodes and lastly, with blue is marked the fastest path from those two points.z

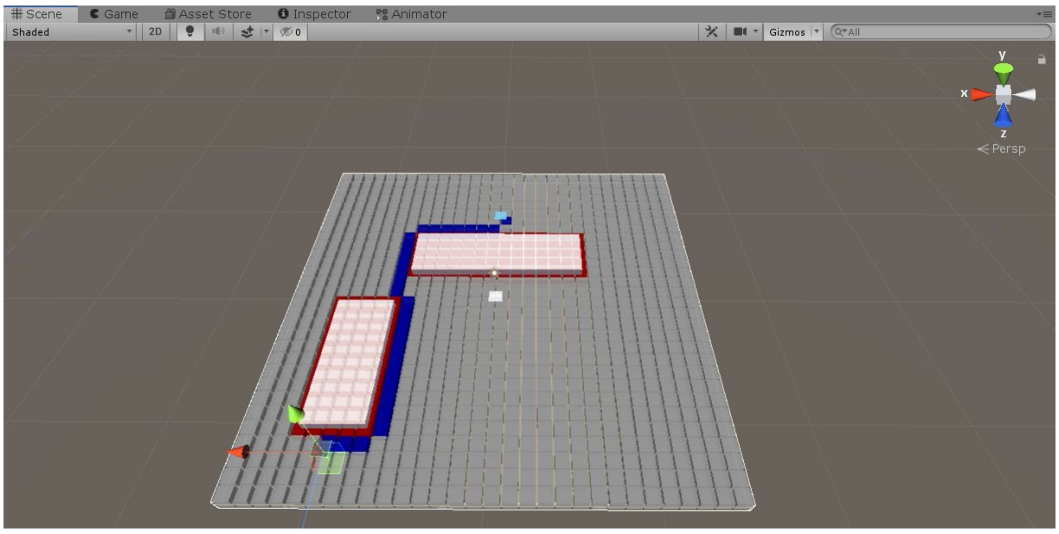


Fig.6. Pathfinding [**26**] Can be found in Report - Images

Steering Behaviours

Another technique present in this artefact is called Steering Behaviours. In this context, the term behaviour is defined by "***Craig W Reynolds"*** in his paper called: *"**Steering Behaviors For Autonomous Characters**"*[**13**] as: "*the improvisational and life-like actions of an autonomous character*". Autonomous character being defined by ***Matt Buckland*** as: "*An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future**"*** in his study called: "***Programming Game AI by Example***"[**14**]. All of this defining the steering behaviours as the ability of an autonomous character to improvise and make life-like actions over time.

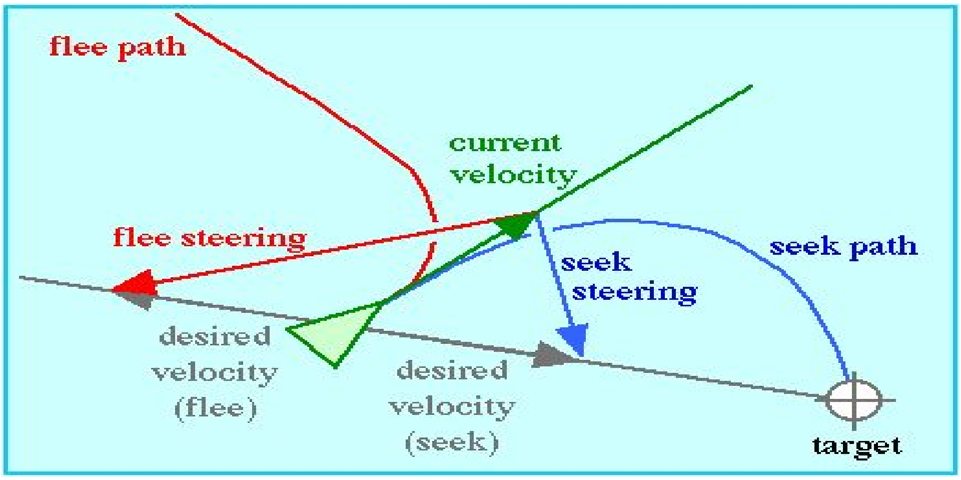


Fig. 7. Steering Behaviours [**27**]

The steering behaviour present in this project is path following. "***Fernando Bevilacqua***" in his article called: "***Understanding Steering Behaviors: Path Following***"[21] defining a path as: "*A path can be defined as a set of points (nodes) connected by lines. Even though curves can also be used to describe a path, points and lines are easier to handle and produce almost the same results.* ". In its most basic form, the behaviour of path following is simply following the points along the line. Depending on the type of behaviour that the developer wants to set to its npc, the agent can either loop on the path or to go back and forth(ping-pong).

This type of path following was used in this project because of the low requirements of the artefact and also to the rail based following system that wanted to be implemented.

To obtain a much smoother behaviour: "***Faris Janjos, Ron Reichart and Philipp Niermeyer***" state in their article for ***ScienceDirect*** called: "***Smooth Path-Generation Around Obstacles Using Quartic Splines and RRTs***"[**22**] that: "*A popular existing method for path planning applications of various vehicles is Dubins path*"[...] " *The algorithm computes the shortest curve between two points with pre-scribed initial and* *terminal tangents while satisfying a constraint on the maximal curvature of the path."* The problem with this algorithm is that the intersection of lines and arcs of circles produces a non-continuous curvature along the path and to fix this splines are being used.

Perception Systems

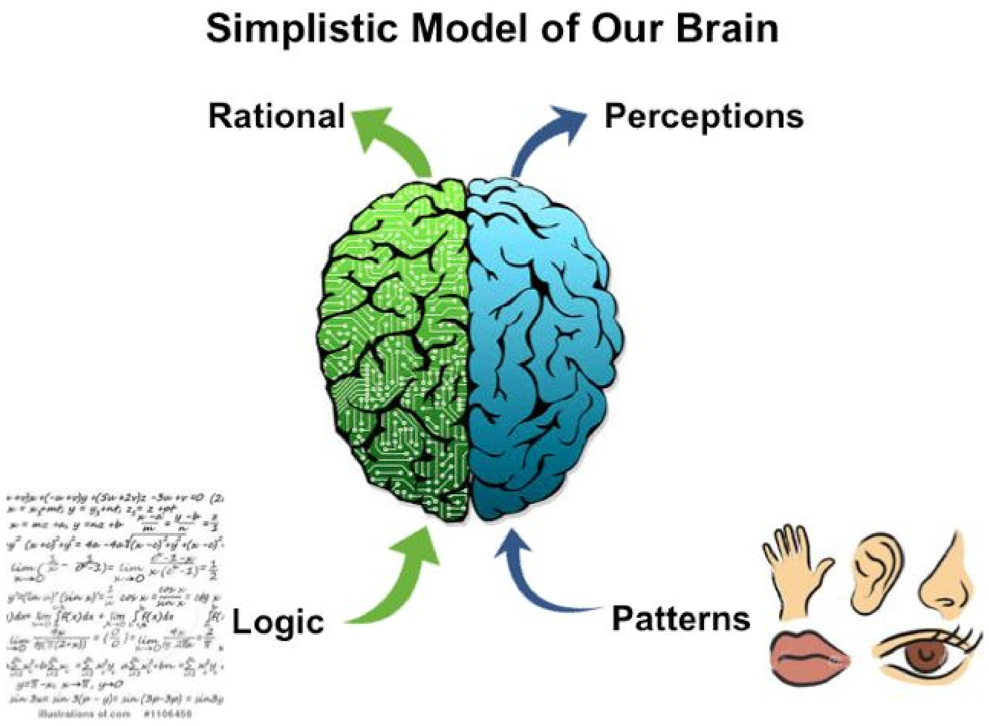


Fig.8. Model of the human brain[23]

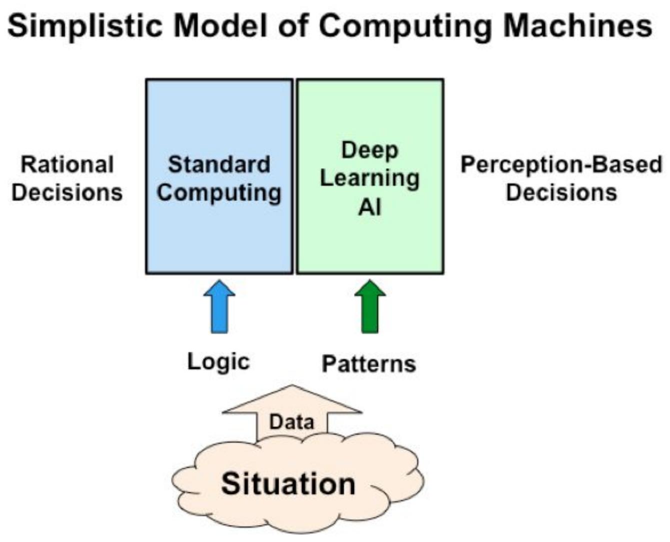


Fig.8. Model of the human brain[23]

When talking about perception systems in artificial intelligence, "***Sharad Gandhi***" is stating in his article called: "***AI is Essentially "Artificial Perception***"[**23**] that: "*The AI system decision-making is based on the perception of the input data patterns, behaving like the right side of the brain — specializing in perceiving patterns.*" He is succeeding in finding similarities between humans and machines.

In games, to improve the player experience, the developers are constructing agents with as much as human alike perception and behaviour as possible. Based on this, there are five possible types of sensory systems: vision, hearing, touching, smelling and tasting (which is not that much used)

In this artefact, there is just one type of these perceptions, which is the vision and it consists of a sphere with a setable radius, which checks for collision with the seeking target. This method of sight perception was used due to it's simple way of being developed which also matches the requirement of the artefact.

Another way to implement the sight would be by using line tracing, and based on the input received from the raycasted lines the agent is being able to see or not.

Reflection[505 words]

To summarize the analysation of the project, this entity is making use of embedded finite state machines to store typology behaviour, Grid based A star pathfinding combined with path following steering behaviours to successfully find and follow a path. Utility theory to get the desire of getting the health pack and sight perception system.

All of these are used in a 3D based Unity developed artefact which simulates a game of mouse and cat, where Iohannis (the blue character) is the cat who has to chase Veorica (the red character) ,which is going to be the mouse, on a grid base map in less than 90 seconds before stealing twenty coins (food).

The major problems that I encountered along the way with the techniques implementation were inside the steering behaviour for path following. The problem consisted in a bad usage of a while loop for a non enumerator type of function which caused crashing problems when playing. The next problem involving Path Following was linked with requesting following another path after the last one was completed. The issue here was the fact that I was resetting my direction vector in the moment where the target index, which is the index used to store the current waypoint, was higher than the capacity of the direction array. The quick fix was to set the index back to zero, not the direction.

Another big problem which generated a lot of frame dropping was linked to the way my perception system was working. More exactly, I was using the ***Vector3.Distance*** method which was using a lot of computational power due to usage of quadratic operations such as square rooting. The fix came from the ***Unity's documentation*** where it states that:"*Rather than compare the magnitude against a known distance, you can compare the squared magnitude against the squared distance*", and this fixed the problem. [**26**]

If I would have to start over again, I would definitely use the advice pointed out by ***Max Bessarabov*** in his article for ***Medium*** entitulated: "***Time Management for Developers: Control Your Life***"[**27**]. He talks about the importance of having a well structured plan and also is giving some insight tips on some very interesting applications.

As it can bee seen from my asset list (Fig 5) I didn't manage to make use of any sounds in this project which I think would give a better gameplay.

On reflection, I would love to have more time to finish the artefact as students from last year had double the time, this means that they had more time to learn and interact with the concept of ai in games but even so, the whole experience of creating something which on a very low scale is capable of doing actions same as a human, reminded me about why I went to this university and gave me a motivational boost truly needed and to enclose it all I think I did a decent job on implementing the required problem even though there is always room for improvements.

References

Wagner, F., Schmuki, R., Wagner, T. and Wolstenholme, P. (2006). *Modeling software with finite state machines*. Boca Raton, FL: CRC Press Taylor & Francis Group, p.78. (**1**)

Bevilacqua, F. (2013). *Finite-State Machines: Theory and Implementation*. [online] Game Development Envato Tuts+. Available at: https://gamedevelopment.tutsplus.com/tutorials/finite-state-machines-theory-and-implementation--gamedev-11867 [Accessed 9 Dec. 2019]. (**2**)

Thompson, T. (2019). Arkham Intelligence. [online] AI and Games. Available at: https://aiandgames.com/arkham-intelligence/ [Accessed 9 Dec. 2019]. (**3**)

Tong, F. (2019). Graph Theory and Deep Learning know-hows. [online] Medium. Available at: https://towardsdatascience.com/graph-theory-and-deep-learning-know-hows-6556b0e9891b [Accessed 9 Dec. 2019]. (**4**)

Joshi, V. (2017). Breaking Down Breadth-First Search. [online] Medium. Available at: https://medium.com/basecs/breaking-down-breadth-first-search-cebe696709d9 [Accessed 9 Dec. 2019]. (**5**)

(2015). Batman. Arkham knight. Burbank, CA :Warner Bros. Interactive Entertainment, Inc.(**6)**

Rockstargames.com. (2015). Grand Theft Auto V. [online] Available at: http://www.rockstargames.com/V/info [Accessed 21 Apr. 2015].(**7**)

Joshi, V. (2017). Finding The Shortest Path, With A Little Help From Dijkstra. [online] Medium. Available at: https://medium.com/basecs/finding-the-shortest-path-with-a-little-help-from-dijkstra-613149fbdc8e [Accessed 9 Dec. 2019].(**8**)

Funaki, S. (1980). Pac-Man. Japan: Namco. [**9**]

Counter-Strike 1.6. (1999). Valve. [**10**]

Botea, A., Bouzy, B., Buro, M., Bauckhage, C. and Nau, D. (2013). Pathfinding in Games. [online] Drops.dagstuhl.de. Available at: http://drops.dagstuhl.de/opus/volltexte/2013/4333/ [Accessed 12 Dec. 2019]. [**11**]

Meier, S. (2019). Civilization. MicroProse.[**12]**

Reynolds, C. (2006). Steering Behaviors For Autonomous Characters. [online] Red3d.com. Available at: http://www.red3d.com/cwr/steer/gdc99/ [Accessed 12 Dec. 2019].[**13**]

Buckland, M. (2010). Programming Game AI by Example. Burlington: Jones & Bartlett Learning, LLC. [**14]**

Filippov, V. (2010). *God Of War III*. Sony Computer Entertainment.[**15**]

Games, C. (2019). *Need For Speed: Hot Pursuit*. Electronic Arts. **[16]**

Montreal, U. (2019). Far Cry 5. Ubisoft. [**17**]

Morasky, M. (2019). Counter-Strike Global Offensive. Valve.[**18**]

Leeuw, M. (2019). Horizon Zero Dawn. Sony Interactive Entertainment. [**19**]

Simpson, C. (2014). *Behavior trees for AI: How they work*. [online] Gamasutra.com. Available at:

https://www.gamasutra.com/blogs/ChrisSimpson/20140717/221339/Behavior_trees_for_AI_How_they_work.php [Accessed 13 Dec. 2019]. [**20**]

Bevilacqua, F. (2013). *Understanding Steering Behaviors: Path Following*. [online] Game Development Envato Tuts+. Available at: https://gamedevelopment.tutsplus.com/tutorials/understanding-steering-behaviors-path-following--gamedev-8769 [Accessed 12 Dec. 2019].[**21**]

Janjos, F., Reichart, R. and Niermeye, P. (2016). SmoothPath-GenerationAroundObstaclesUsingQuarticSplinesandRRTs. [online] www.sciencedirect.com. Available at: https://www.sciencedirect.com/science/article/pii/S2405896317323157 [Accessed 12 Dec. 2019]. [22]

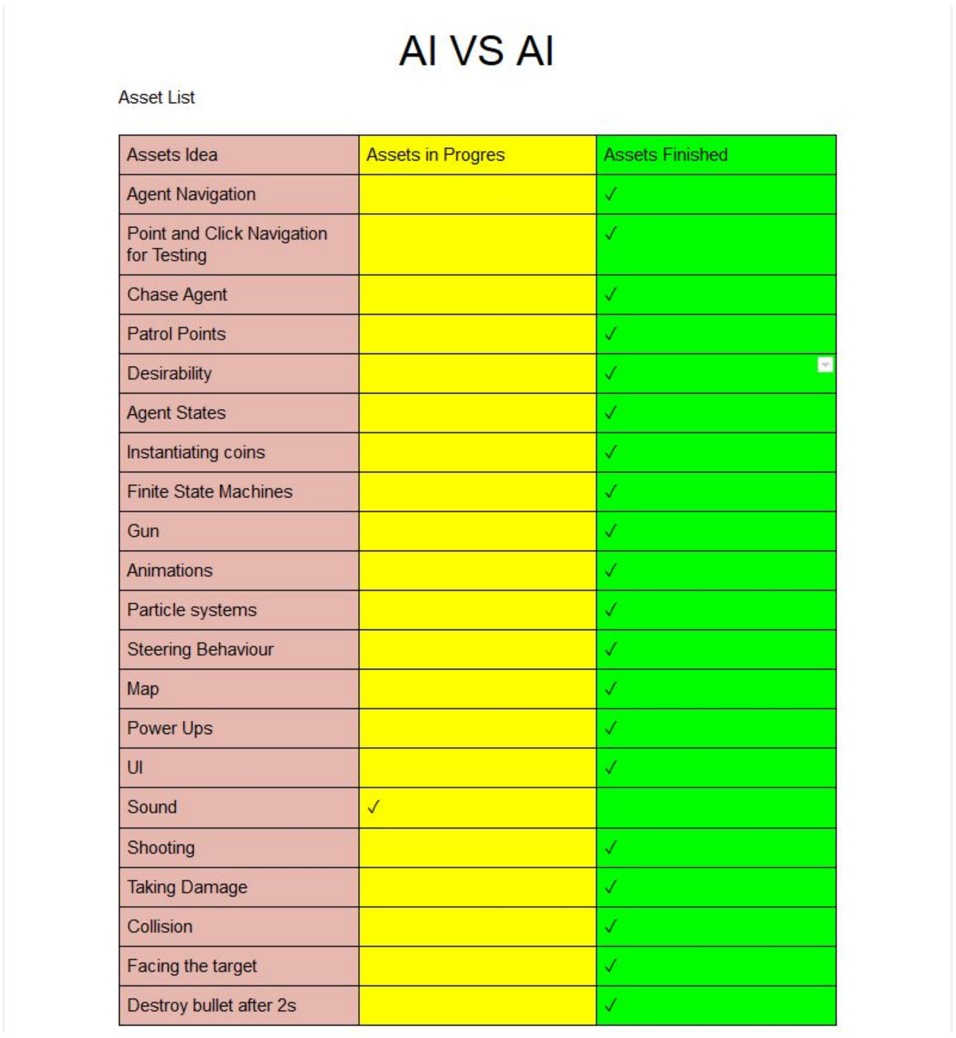
Gandhi, S. (2017). AI is Essentially “Artificial Perception”. [online] Medium. Available at: https://towardsdatascience.com/ai-is-essentially-artificial-perception-f69f0493613d [Accessed 13 Dec. 2019].[**23**]

Uddin, J. (2019). *Utility Theory / Desirability for AI Agents*. [online] Moodle.port.ac.uk. Available at: https://moodle.port.ac.uk/course/view.php?id=8456 [Accessed 13 Dec. 2019]. [**24**]

Uddin, J. (2019). *Steering Behaviours for Autonomous Characters*. [online] Moodle.port.ac.uk. Available at: https://moodle.port.ac.uk/course/view.php?id=8456 [Accessed 13 Dec. 2019]. [**27**]

Farquhar, P. (1990). *Applications of Utility Theory in Artificial Intelligence Research*. [online] link.springer. Available at: https://link.springer.com/chapter/10.1007%2F978-94-009-2203-7_10 [Accessed 13 Dec. 2019]. [**28**]

Appendix



AI VS AI

Asset List

Assets Idea	Assets in Progres	Assets Finished
Agent Navigation		✓
Point and Click Navigation for Testing		✓
Chase Agent		✓
Patrol Points		✓
Desirability		✓
Agent States		✓
Instantiating coins		✓
Finite State Machines		✓
Gun		✓
Animations		✓
Particle systems		✓
Steering Behaviour		✓
Map		✓
Power Ups		✓
UI		✓
Sound	✓	
Shooting		✓
Taking Damage		✓
Collision		✓
Facing the target		✓
Destroy bullet after 2s		✓

Fig 9. Asset List - Can be found in Report-Images