

**Coursework 1**

**Game Artificial Intelligence**

|  |  |
| --- | --- |
| **Module Name:** | Game Artificial Intelligence |
| **Module Code:** | M3I622941 |
| **Module Leader:** | Hamid Homatash |
|  |  |
| **Student ID:** | S1717551 |
| **Student Name:** | Dawid Piotr Kubiak |

**By submitting this assignment, I agree to following statement:**

“Except where stated explicitly, all work in this report, is my own original work and has not been submitted elsewhere in the fulfilment of the requirement of this or any other award”

<https://github.com/dejwkubikson/FPS-Complex-AI>

**Human-like Complex AI**

Contents

[Project Proposal 3](#_Toc27684165)

[What are human like actions? 4](#_Toc27684166)

[How will this be achieved? 4](#_Toc27684167)

[Environment 4](#_Toc27684168)

[Investigation and Research 4](#_Toc27684169)

[Killzone’s waypoints 4](#_Toc27684170)

[Call of Duty’s AI’s desire to kill 5](#_Toc27684171)

[Emotional AI characters as the AI’s next step 5](#_Toc27684172)

[Pac-Man’s randomness 5](#_Toc27684173)

[Experiments 6](#_Toc27684174)

[Implementation 6](#_Toc27684175)

[AgentScript 6](#_Toc27684176)

[CoverFinderScript 7](#_Toc27684177)

[MovementScript 9](#_Toc27684178)

[EmotionScript 9](#_Toc27684179)

[DecisionMakingScript 10](#_Toc27684180)

[Occurred issues 13](#_Toc27684181)

[Evaluation 20% 14](#_Toc27684182)

[Memory usage 14](#_Toc27684183)

[CPU usage 15](#_Toc27684184)

[Testing scripts with Stopwatch 15](#_Toc27684185)

[Overall Performance 16](#_Toc27684186)

[Reflection 16](#_Toc27684187)

[Overall performance 16](#_Toc27684188)

[How did the project go? 17](#_Toc27684189)

[New possibilities 17](#_Toc27684190)

[A journey I enjoyed 17](#_Toc27684191)

[References 17](#_Toc27684192)

# Project Proposal

Artificial Intelligence is becoming popular to a greater extent thanks to the technical advancement. People hear more often about this term and are getting generally more interested in it. Artificial Intelligence is a theory and development of computer systems that are able to perform tasks that normally involve human intelligence, such as decision making, learning or planning. Due to its possibilities, Artificial Intelligence is also used in games, performing tasks related to steering behaviours, pathfinding or the mentioned before, decision making.

The project is all about creating Artificial Intelligence that takes actions as close to human-like as possible. I hope to achieve a behaviour that when the player faces the AI agent he will get a feeling like he is facing another person.

### What are human like actions?

To start with, when the AI will be in a group, it will try and move towards the player, cover by cover, thinking as one and working in a group rather than individually by supplying cover fire towards the player when one of the agents is changing its cover, by trying to flank the player and by spreading around the location to make it harder for the player to track the agents, control them and react to their actions. These types of behaviours are similar to how people try to work as a team to reach a certain goal. Moreover, a random factor will be added to keep the AI similar to what people do as although we plan our actions, we do not always abide by them due to various circumstances, sometimes it’s an impulse, sometimes the emotions take over, sometimes we come up with a better idea. The change of decisions is usually based on how we currently feel, if we are in danger or if we are in the winning position, therefore, three basic emotions will be implemented for the agents – fear, confidence and adrenaline.

### How will this be achieved?

How to make the AI act as close to human alike as possible? Using pattern recognition, planning and emotions. In order to obtain such impression for the player, that he is facing another person, multiple AI systems will be implemented. Hierarchical Finite-System Machine will be the base of agent’s actions for decision making, which is going to be built upon the environment and player’s actions. Moreover, integrating steering behaviours, some randomness and emotional factors will help agent pick the most appropriate (or emotional), human like decision. Implemented emotions’ ‘levels’ will change and impact the agents’ actions and plans depending on what is currently happening around them. Due to the fact that fear and confidence are the opposites of each other, one will rise while the second will decrease which will impact the overall performance of the agents, for example, high fear will result in inaccuracy, less position changes and desire to attack the player. On the other hand, high confidence will make the agents more eager to attack, change covers and so on. These two emotions, will have an impact on the whole group, therefore, high fear level on one of the agents will slightly effect the others, the more agents are in fear, the bigger overall effect. Adrenaline will affect the speed that agents take the actions, however, not always as planned – this is the emotional factor that will impact the randomness. Adrenaline level will be triggered by injuries and fear.

### Environment

This complex AI will be made for a Frist Person Shooter style game, with the player facing several enemies. The AI will be checked and tested with different number of opponents to notice the changes in behaviour in different situations. The test scene will be on a platform with obstacles so that both, the player and agents, can have multiple objects to hide behind, pick different paths for their goal and react to each other’s actions on a big scale. The agents’ mission is to kill the player, therefore, the player’s objective is to survive.

# Investigation and Research

### Killzone’s waypoints

Arjen Beij, one of the AI designers for Killzone, during European Game Developers Conference in London, said that they have focused more on individual AI soldiers rather than on coordinating various squads. Killzone’s AI is based on using the terrain and objects so that the agents can cover and protect themselves. Beji said that there are four main emphases[1]:  
- picking tactically advantageous positions,   
- evaluating the position   
- tactically moving to and from position to position,  
- suppression fire  
The agents’ main point of focus is defense as they are mostly looking for covers. The AI uses waypoints that are placed around 2 game meters away from one another and the AI can evaluate his position, then, the agents are encourage to search out for positions that guarantee cover. For each waypoint, the AI soldier checks if the player does or does not have a cover, this is “programmed in a lookup table, which acts like an old pen-and-paper board game.” This is much more efficient than ray casting from each point. The 4000 waypoints per map require only about 64 kilobytes. [1] The AI does not reassess its position on the move, however, in cases when the player changes his position, the agent will abort his decision and recalculate it – this is an example of an Alarm Mechanism that is used in Hierarchical Finite-System Machine. Killzone’s AI uses around 12% of CPU power, which is a great result considering that it was designed for Playstation 2. Therefore, using the waypoints with a lookup table is really worth considering for cover finding, especially when you compare the power it used to today’s CPUs. Nonetheless, my overall thought is that it would be much better if the AI was tweaked a bit to be more aggressive as its defensive tactic might not be so engaging for the player.

### Call of Duty’s AI’s desire to kill

Due to the fact that the game will be a First Person Shooter type, I have had a deeper look into this year’s (2019) Call of Duty Modern Warfare enemy behaviour. When playing in a Realistic mode the AI is very aggressive, most of the time targeting the player and although some behaviours are definitely scripted, the overall feeling is very pleasant for the player. For example, when clearing a house, the agents sometimes jumped out of the rooms, sometimes hid in a corner of a room or just waited with the gun pointed at the doors and started shooting as soon as they opened. The only thing that’s missing is that the enemies were taking actions individually rather than as a group. How does the above translate to my desired AI? The enemies’ behaviour when clearing the house seemed random, yet, their actions were invoked by what player did. This is of course AI cheating as it knows exactly what the player is doing and where he is, however, there was no feel to it thanks to the randomness factor. Moreover, we can assign emotions to them, as for example, when a woman tried to pick up a gun, once the player pointed his gun at her and aimed, she stopped reaching for it – this would refer to the ‘fear’ emotion in this project. It is worth noting that there were other factors that impacted player’s feelings and overall experience like shouting and screaming which built up emotions but focusing only on the AI part it still definitely gave a great impression.

### Emotional AI characters as the AI’s next step

Damian Isla and Bruce Blumberg mention in their New Challenges for Character-Based AI for Games report that character-based AI will be the AI’s next step. Character-based AI is a system that seeks to simulate high-level logical human thinking [2]. Moreover, they have mentioned that one of the components of character-based AI is Emotion Modelling. They elaborate that emotions do not only show off on our face, physical motion or stance, but that they influence how we make, think and plan our own decisions. They state that virtual characters should use show off their emotions to full extent – not just physically, but psychologically as well – which would then impact on how the agent behaves. The authors gave an example of ‘frustration’ as AI’s emotion, that it could be used as a signal that “this strategy is not working, try another” [2], ‘urgency’ and ‘desperation’ could have an impact on the character being in rush which would result in how quickly he makes the decisions.

### Pac-Man’s randomness

Michael Mateas in his “Expressive AI: Games and Artificial Intelligence” article after explaining that AI system should construct intelligent behaviour - behaviour that the player can assume that it has been composed by an intelligence with its own desires and that reacts to player’s action [3] – gives details about the interview with Toru Iwatani, Pac-Man’s designer. In the interview, Iwatani mentions that two of the four ghosts moved more randomly to give a more natural way of getting closer to Pac Man. As he adds, if the player would be constantly under attack, he would become discouraged, therefore, he has developed a wave-patterned attack. Michael Mataes tries to evaluate the ghosts actions a bit more. He spots out that each of the four ghosts actually followed a different strategy when pursuing Pac Man. [6] Thanks to their unpredictability of behaviours that interprets player’s actions, Michael considers it as one of the fundamental goals of successful AI design.

### Experiments

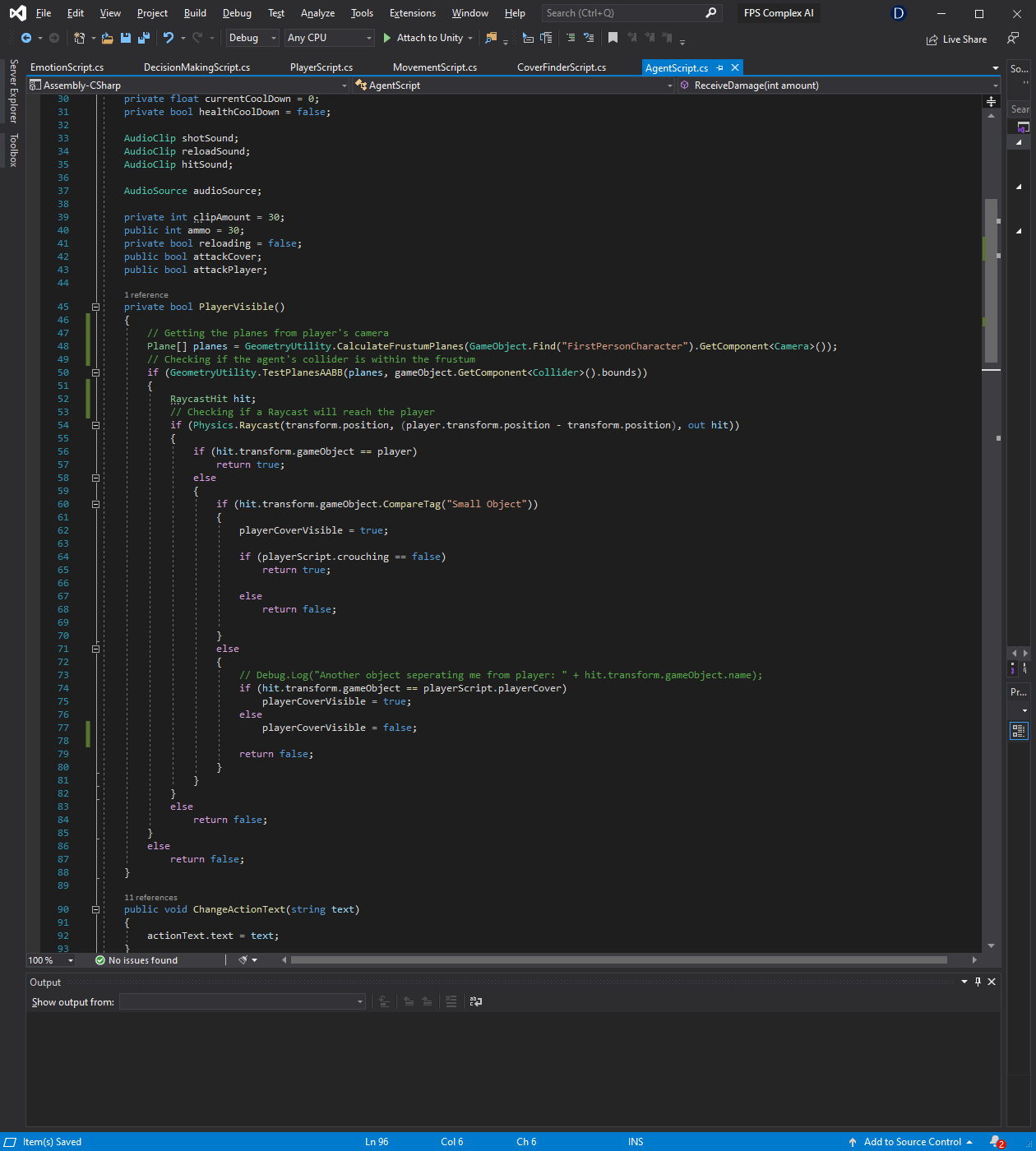
Before creating the whole environment, I have tested each part I wanted to take from the above mentioned examples. From Killzone’s AI I have managed to use three of the main emphases described by Beji. I have successfully tested the agents to pick tactically advantageous positions (for example when trying to flank the player), trying to tactically move to achieve their objective (moving from cover to cover) and even made the AI characters provide covering fire when one of the agents was currently on the move to his next cover. Furthermore, throughout the tests, I tried to achieve Call of Duty’s desire to kill by making the agents more eager to move to covers that make the agents closer to the player. Apart from that, each time they spot the player they will shoot without hesitating. From Damian Isla’s and Bruce Blumberg’s report I have played around with the emotions and tested how they impact on agent’s performance and the decision they make. Lastly, after adding decision trees that are based on randomness, I have spotted that the agents seemed more natural and realistic as they were much more unpredictable. Once all of the separate parts were fully working they were joined to form one, big environment.

# Implementation

As mentioned in the previous chapter, the scripts needed to be connected to form one big environment. This was achieved by making one of the scripts the main point of action. The agent is controlled by five different scripts that are correlated. Main operations are done in the DecisionMakingScript which has all agent’s scripts assigned and uses the player’s script as well to determine what to do. However, before analyzing how the script works, all the individual components need to be described.

### AgentScript

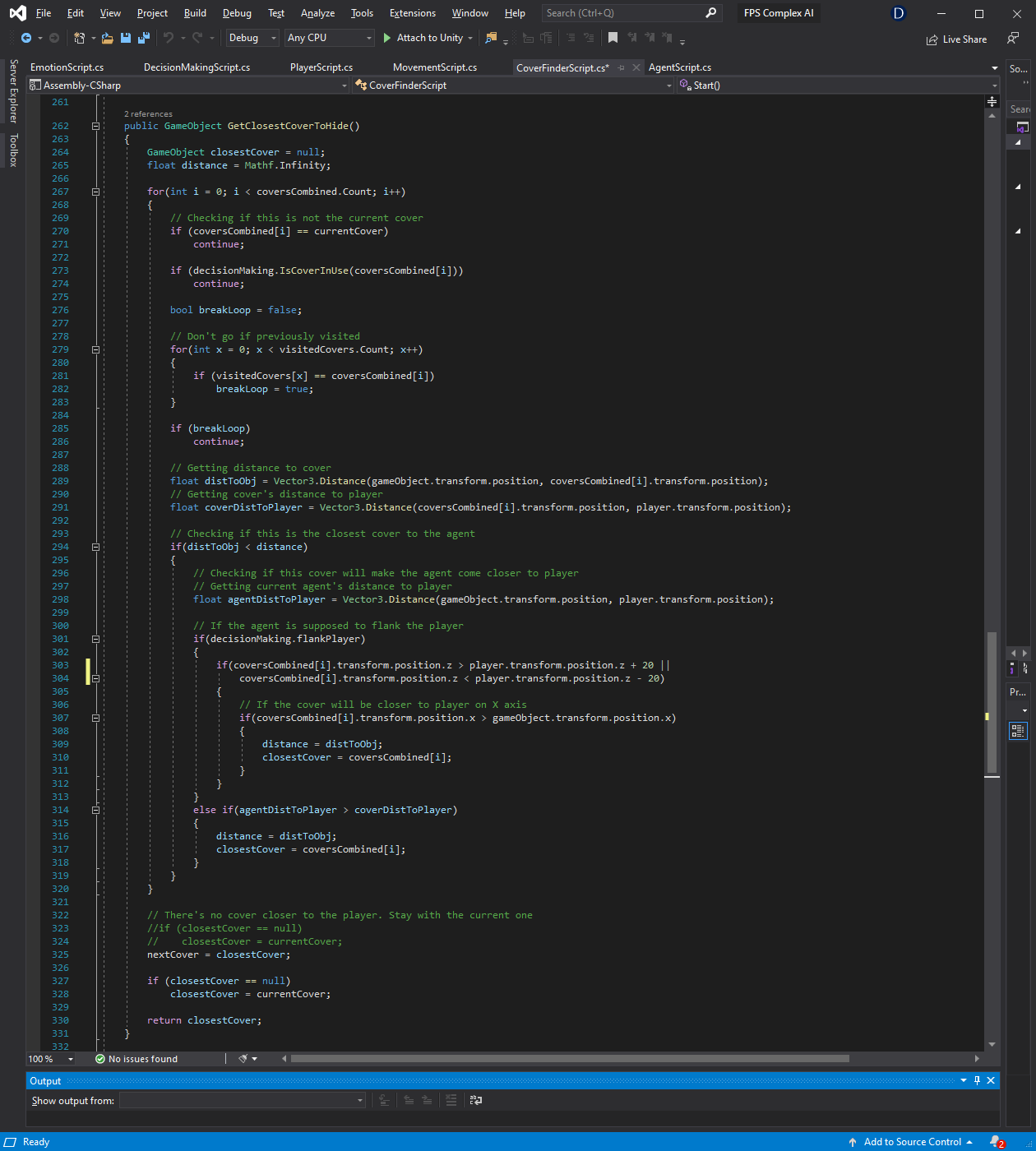
AgentScript script is in charge of agent’s actions – shooting, reloading, playing sound clips, receiving damage and even dying. Apart from that, the script plays a vital role as it checks if the player is visible using camera’s planes. This is crucial to the DecisionMakingScript as it is an ‘alarm’ which indicates that the agent might be in danger.



The function is ran every frame and returns a Boolean value. It starts by taking the planes that form the player’s camera view frustum and calculates whether the agent’s collider is within the bounds of the camera’s view. If it is, a ray cast is set from the agent to the player and if the player is hit with the ray cast, the function will return a true value, moreover, the function checks if the player’s cover is visible which would allow the agent to attack it.

### CoverFinderScript

CoverFinderScript is responsible mainly for finding appropriate covers but also for dealing with any collisions. The collisions determine if the agent has reached the cover he chose or if there is an object on the way that he is supposed to avoid. Avoiding the objects is also done within this script as it is closely related to moving to cover. Avoiding objects is based on randomness and some basic logic. If there is no object next to the item that the agent needs to avoid he can choose whether he wants to avoid the object from the left, right or by jumping on it (only if the object is small), however, if there is an item on a certain side, the agent will go the opposite way.



The GetClosestCoverToHide() function returns a game object. In order to get the closest cover a for loop iterates through a list holding all the available covers. In the for loop, an if statement compares whether current cover is not the one iterating over now, and then if the cover is not in use by any other agent which solves the problem of having all the agents behind one cover. After that, due to the fact that the intention was to have agents rather aggressive and eager to get the player, a for loop checks if the cover was not visited before, the agent’s are supposed to constantly try to move to towards the player. Later on, the function compares the agent’s distance to the cover and if it is lower than the one registered before, depending if the agent is supposed to flank the player or not, the closest cover is assigned. A possibility of the map running out of available covers has been taken into account, therefore, if the closest cover at the end of the function is still null, it means that the agent has visited all the covers or that the current cover is the only one separating the agent from the player. The agent will stay behind that cover and attack the player.

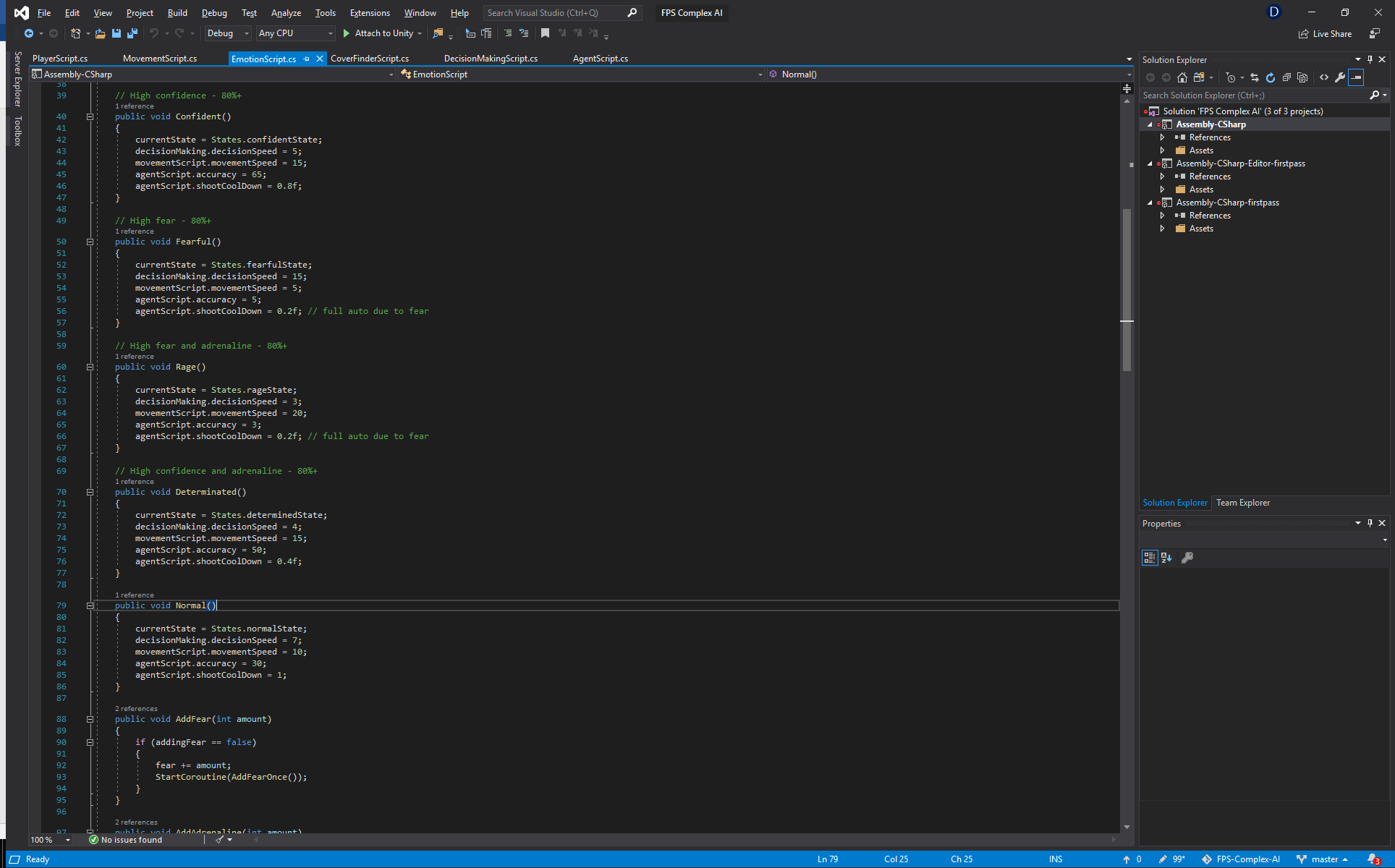
### MovementScript

Another script that plays a vital role is the MovementScript. It controls the agent’s movement. The agent has a path to follow that has specified Vector3 points list held in the above script. When the player reaches the point it is simply removed from the list. Furthermore, the script controls agent’s crouching - which allows the agent to fully hide behind small objects – and lastly, controls the jumping action that is possible to perform on small objects. The ability to jump gives another way of avoiding small objects.

### EmotionScript

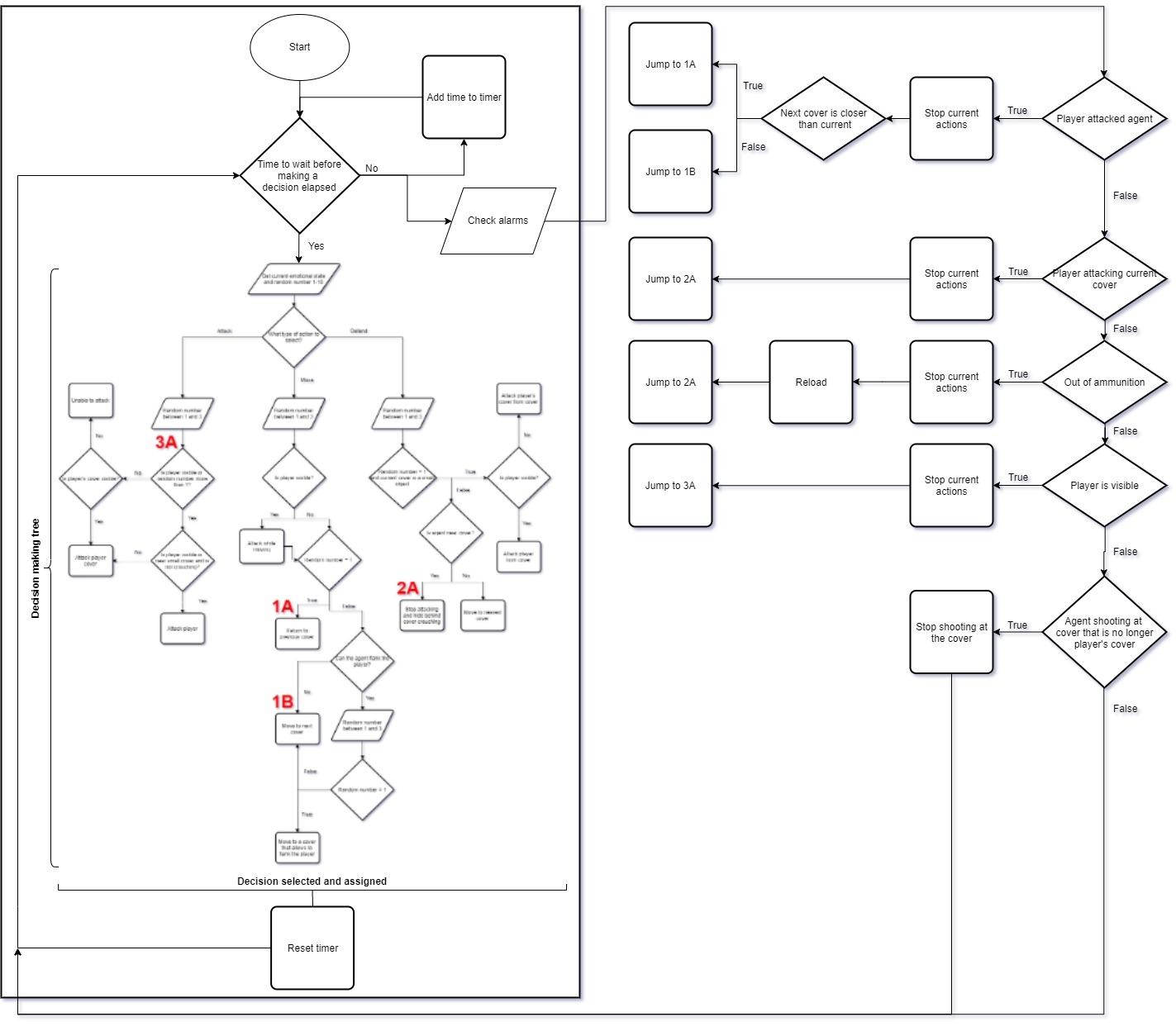
Last but not least, an EmotionScript script is interacting with the DecisionMakingScript. It is responsible for showing agent’s current emotional state. There are three emotions implemented. The agent can be fearful, confident or ‘on’ adrenaline, although fear and confidence are the opposites, the adrenaline plays a role as an additional factor to determine the agent’s state. The emotion needs to reach 80% to have an effect on the agent, fear rises when the agent is receiving damage or when his cover is under attack. Confidence slowly grows when the agent is not under attack and pushes up when he successfully hits the player. Adrenaline builds up when agent’s cover is under attack or when the player is currently targeting him and shooting into his direction. For example, when the agent has high confidence, his emotional state is confident which will result in high accuracy, movement and decision making speed, whereas with additional adrenaline, he will be determined, which will result quicker decision making and higher shooting rate with just slightly less accuracy than when confident. On the other hand, combining fear and adrenaline will result in agent changing his decisions often, low shooting accuracy and fast shooting rate to achieve an effect of shooting fully automatic.

Each frame, the EmotionScript checks if the agent is currently in fear, confident or has high adrenaline level. Depending on his emotion levels, he then can be in one of the following emotional states: confident, fearful, raging or determined. As described before, each emotional state results in different movement speed, shooting rate, accuracy or how quickly the agent will make new decisions. Furthermore, agent’s emotional state is used in the DecisionMakingScript in order to make the agent base his decisions depending on his emotions – just like a human being.

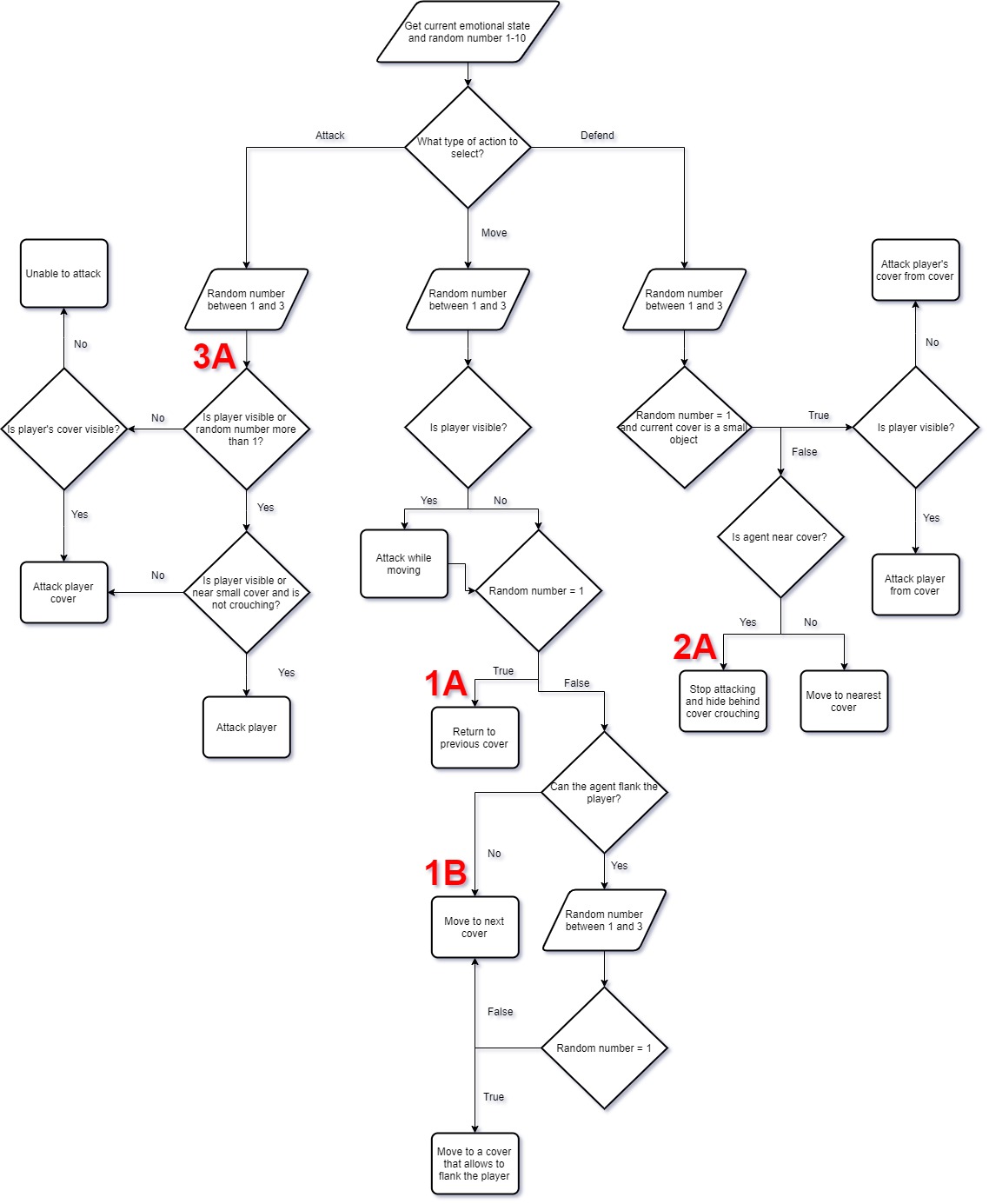


### DecisionMakingScript

Taking all of the above mentioned scripts into consideration, DecisionMakingScript decides what actions should the agent perform. However, apart from retrieving all the details, there are two decision trees implemented that give randomness to the game. When the script does not have to deal with the alarms - when agent is under attack, when agent’s cover is under attack, when the agent runs out of ammunition or when he is visible by the player - the first decision tree takes into account the current emotional state of the agent and depending on the state, the agent will have more or less desire to either attack, move or defend. The second decision tree is more about how to perform the selected action – for example, if the agent chose to move, the second tree would determine if he should move to the next cover, or return to the previous one with the odds higher to move to the next cover to force the agent in a natural way to move closer to the player.



How the decision making mechanism works.

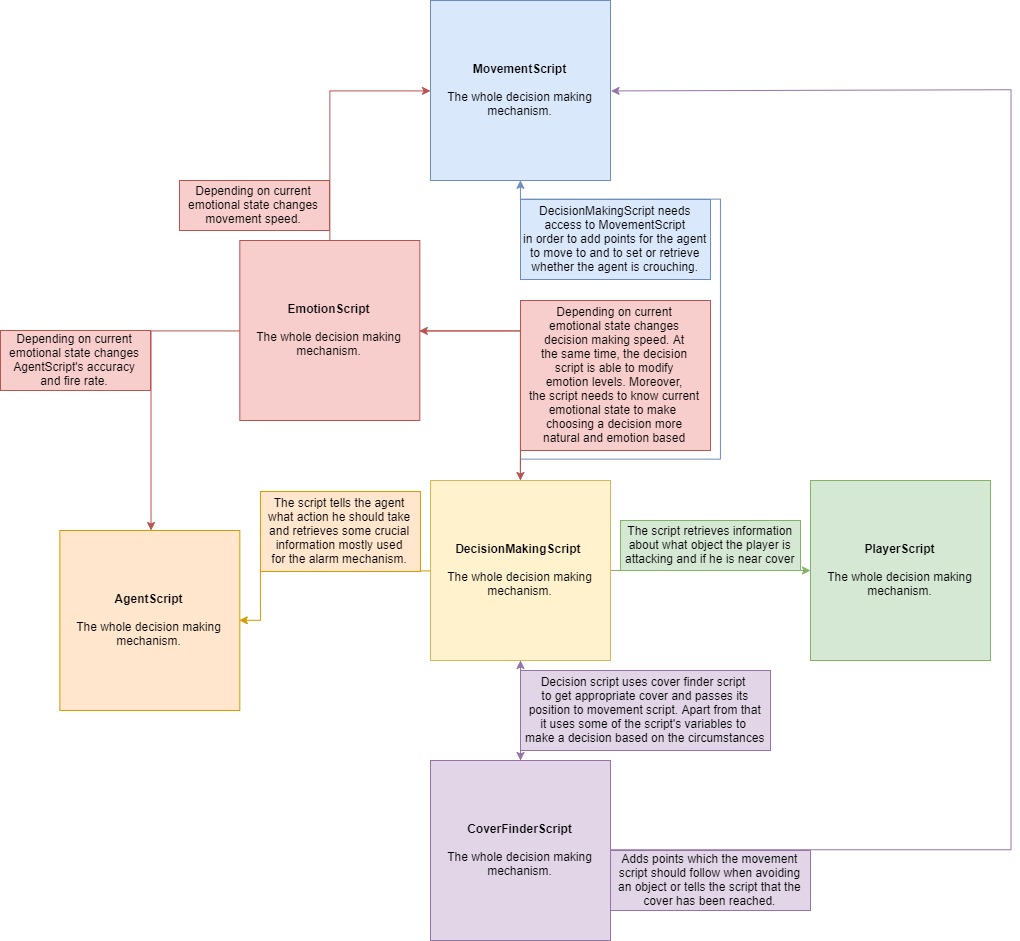


Closer look at the tree inside the machine.

How does the diagram translate to AI agents acting human-like? First of all, the FSM gets agent’s current emotional state, depending on that state there are different odds on what type of action he will chose – this is considered as the first decision tree. After that, due to the fact that each action can be done in multiple ways, a second decision tree (more of a branch) has been implemented, however, this time it’s purely based on randomness (with just double checking if the action can be performed). After a decision has been made, the agent will try to fulfill it and wait until the next decision time comes up. However, in some circumstances the decision can be stopped and an ‘alarm’ mechanism will take over. The alarm means that the agent is in danger or vulnerable and has to stop his current actions and needs to counter the alarm. Alarm takes the agent straight to a picked decision – as long as it is possible to obtain it without checking if it is feasible- and the agent will then process the selected activity. If no alarms occur the agent would just proceed with the decision making.

### Occurred issues

The main issue was how to make all these separate scripts work together seamlessly. I had to draw all the scripts in a diagram and think about how to connect them to form one big environment. I had to consider what information the scripts are going to need to make the AI feel natural and how they are going to gather them. I came up with the following:



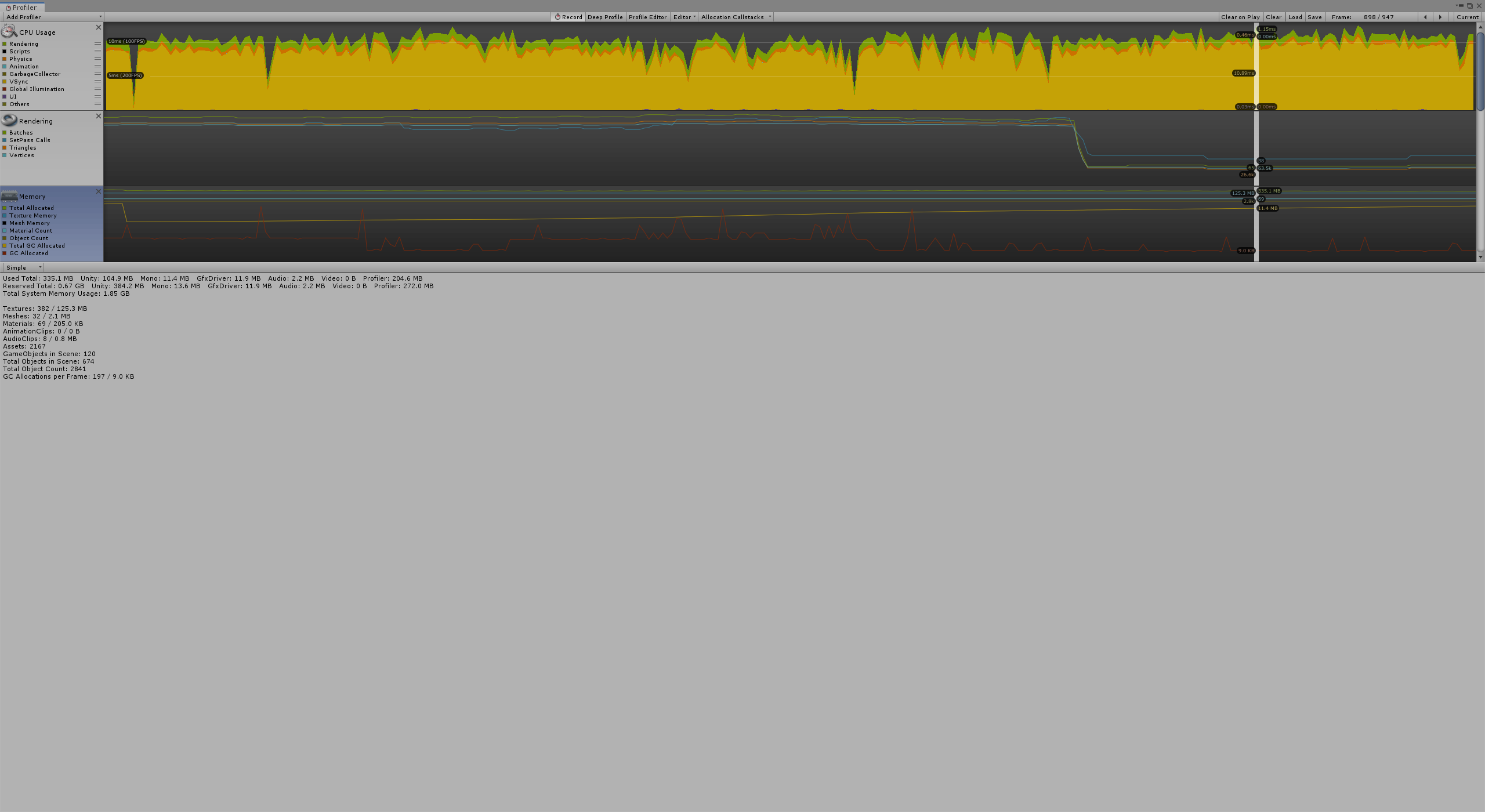
Another issue I had to face was the agents moving from cover to cover. At first, the agent kept moving between same covers (as they were closest to each other), therefore, I had to make the agent look for covers that are nearest to him but will make him move closer to the player as well. After that, I had to prevent the agents from colliding with objects on their way, in order to do that, an avoidance system has been implemented. To bypass an object, the agent can either move left, right or if the object is small, jump on it – moving left or right is randomly selected as long as there is no item on the side of the obstacle to avoid, if there is, the agent takes the free way. This still has not been fully resolved and in some cases the agents struggle to avoid an object. A better way would be to use pathfinding. This would definitely help the agents find the right way to their destination point and make the game feel more natural and AI agents more intelligent.

Videos of AI agents making decisions:

https://drive.google.com/file/d/1Qlfs2TLwiYnGJKFou2dpbKT\_W30htHyF/view?usp=sharing

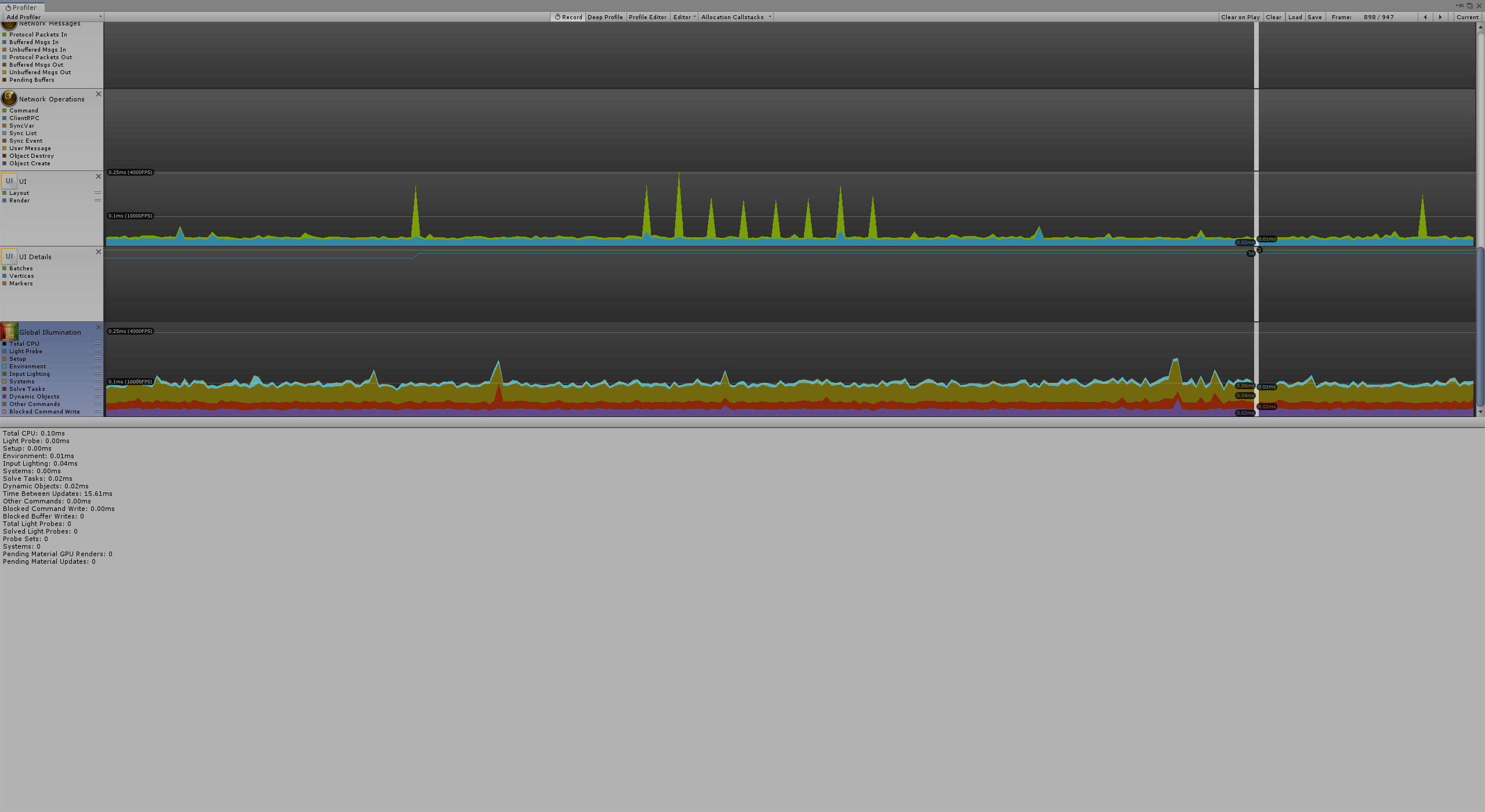
# Evaluation 20%

### Memory usage



It is worth noting that the above memory usage shown in the profiler shows all of the memory usage in the Editor, and not the application’s release platform. According to Unity’s documentation, these numbers are generally larger than when running on release version due to Unity Editor using specific objects to take up the memory, and the Editor window using extra memory for itself. The total memory used is 335.1 MB with Unity taking only around one third of it. Total reserved memory is 0.67GB for the purpose of not asking the operation system for memory too often. Garbage Collection allocations per frame is 197 giving 9 KB and it runs around every 10 seconds. I have tested the project for a longer period and have not spotted any memory leaks.

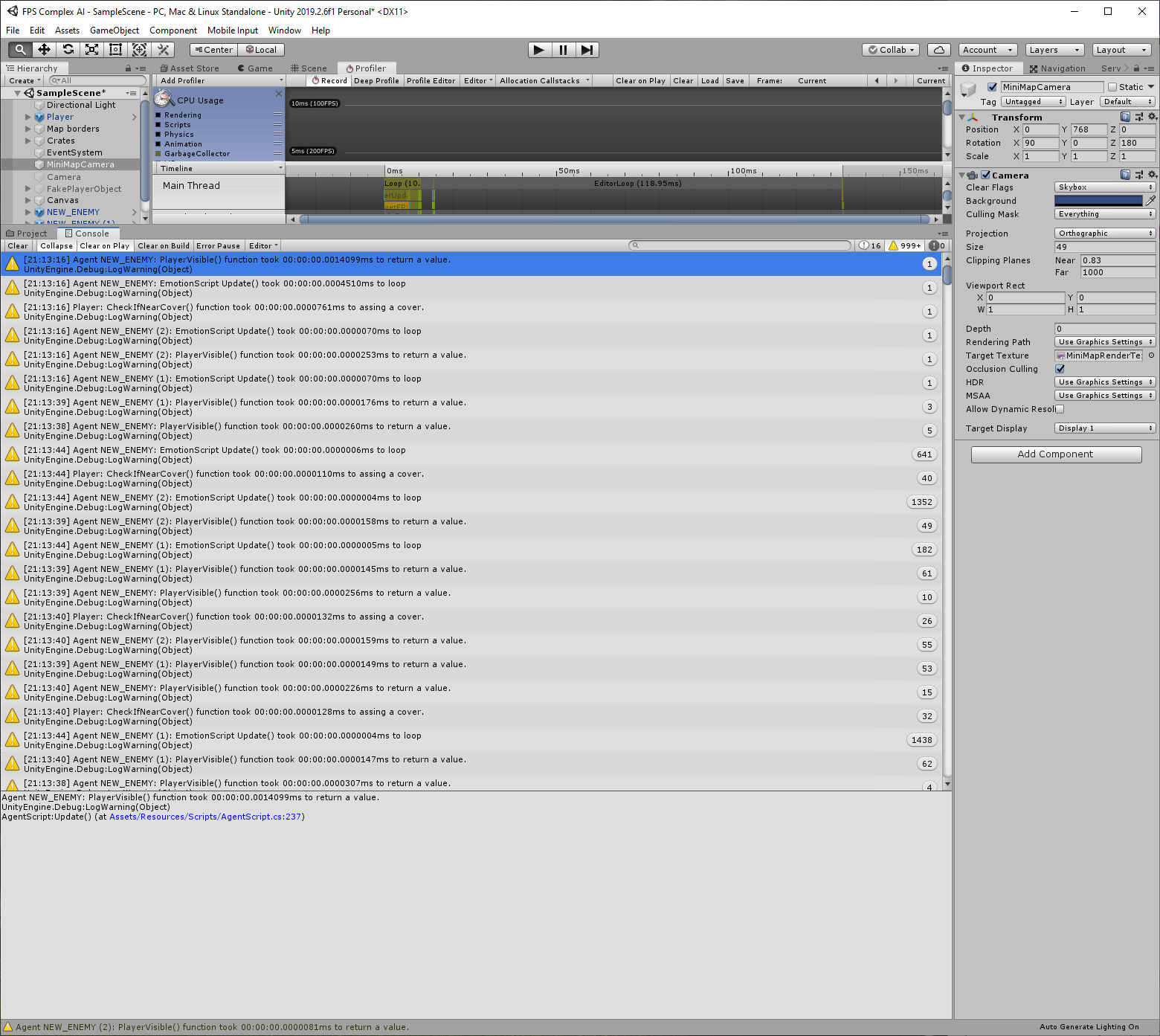
### CPU usage



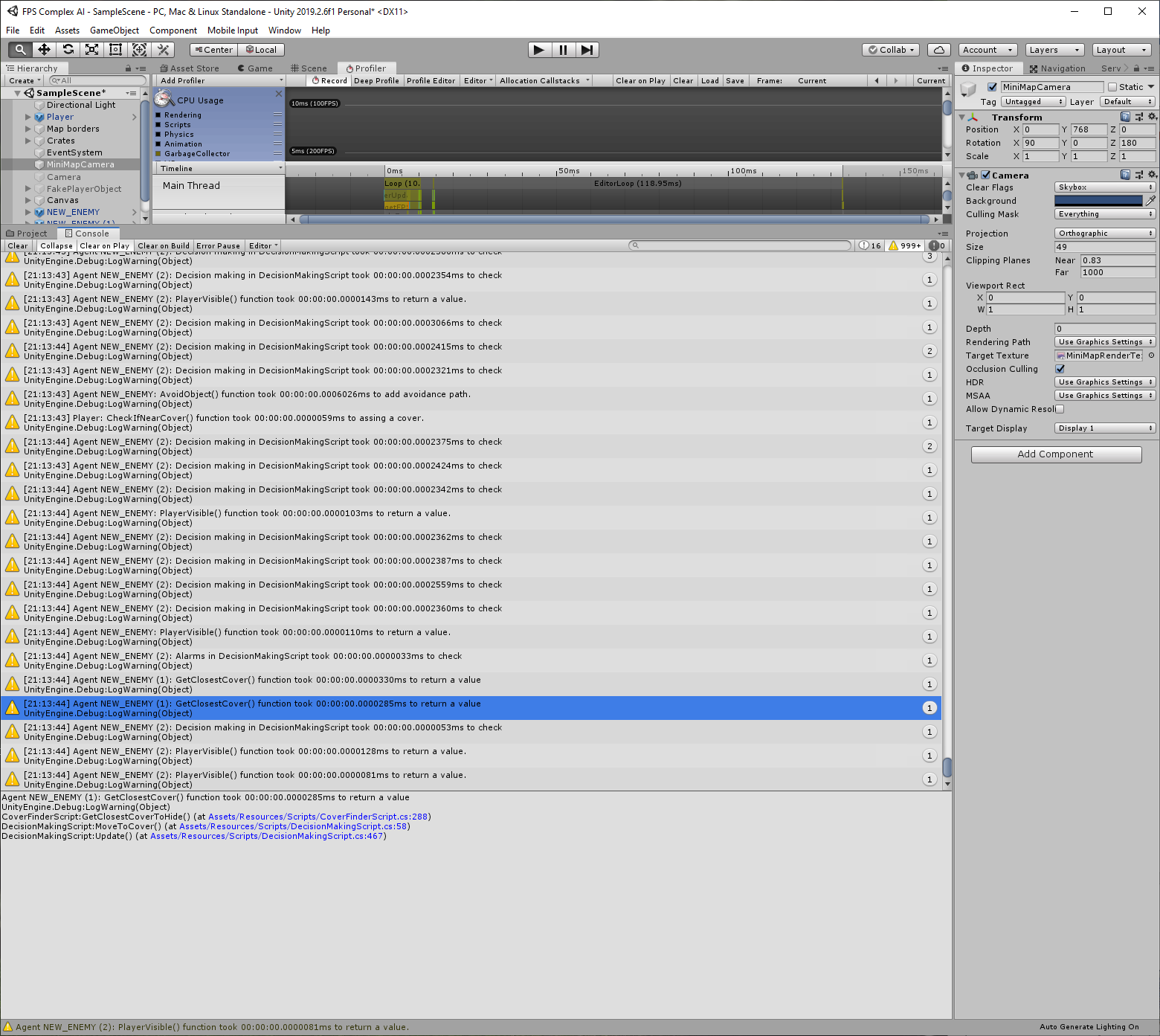
The Global Illumination shows statistics and how much CPU time is consumed by the real-time GI subsystem across all worker threads.

### Testing scripts with Stopwatch

Main, expected to be heavy in calculations, function from each script have been tested using the C#’s Stopwatch class. The AgentScript’s PlayerVisible() function takes around 1.5 milliseconds to return a Boolean value if the player is visible by the agent or not.



In CoverFinderScript I have tested two function, the first one, GetClosestCover() that returns the closest cover to the agent takes approximately 0.03 milliseconds. The second one, AvoidObject() that adds a pathway to help the agent avoid an object, took 0.6 milliseconds.



PlayerScript that is attached to the player, has a CheckIfNearCover() function that checks if the player is currently near a cover. This takes 0.01 milliseconds to assign a value to a Boolean variable and a GameObject.

In EmotionScript, the full Update() loop takes the same amount of time as the player’s CheckIfNearCover() function to determine in which emotional state the agent is.

In the main script, DecisionMakingScript, that combines all of the above, dealing with ‘alarms’ takes only 0.003 milliseconds and making a decision – based on two trees - around 0.02 milliseconds.

### Overall Performance



The average performance is on an acceptable level bearing in mind that the scene does not contain any high poly objects or advanced physics. I must admit I was expecting it to be less efficient and that the refresh rate would be lower and that using the watches in the script would produce much higher timings. I am happy that the rendered frames per second seem to be quite steady and there are no sudden drops in the performance. Looking at the graph the refresh rate is around 100 frames per second. With V-Sync disabled (in the profiler) average rendered frames per second stay at around 1000 which means that running the scripts is not very complicated for the computer. Furthermore, the CPU Profiler did not detect any issues to warn about which would mean that the project is designed properly. Last but not least, considering the timings of crucial and, as I initially thought, heavy functions I am generally pleased regarding the performance.

# Reflection

I was always interested in Artificial Intelligence and eagerly read articles about it, no matter if the article was about how AI is used in Business Intelligence, Face Recognition, Data Mining or games. I enjoyed discovering and learning about Artificial Intelligence as whole. I am really pleased that I had the opportunity to discover it myself during this project, all the more that I already wanted to do an AI project for a while. Although I am mostly interested in Deep Learning and watching how the machine solves puzzles or plays games and becomes better and better with each attempt, I undertook a simpler type of AI – Hierarchical Finite State Machine. However, I definitely intend to learn about it more and will experiment with different AI techniques.

### Overall performance

I think my overall performance could be rated as just above standard. I have tried all my best to create something I have imagined and thought that would be great to have in a game that I could actually play. I think I might have wanted to do too much and had to simplify some things – for example, in my project proposal I wanted to use pathfinding along the decision making system, however, due to lack of time to make it as good as I wish, I had to use something that I do not think I could name as pathfinding. Nonetheless, as I have mentioned, although I could not use all the techniques that I expected to use at the beginning of starting the project, I am glad how the final version of the project operates.

### How did the project go?

The main part I was struggling with was making the agents move to covers while avoiding any object they might have encountered. This has not been fully resolved until now but considering that the project was mostly focused on how the agents make their decisions, I think it is okay to turn a blind eye on that. On the other hand, I think that the decision system is scripted pretty well and that I have taken into account any situation that might occur and neutralized any unexpected results (as long as it is not something that is related to movement). I believe that the decision mechanism is working as I intended at the start of the project and that it could actually give the player experience of facing a human-like, decision making AI. Furthermore, I think I might have started the project wrongly. I started by designing each script separately whereas I believe I should have been developing the scripts synchronously. I have spotted this half way through the project, when it was already slightly too late. I think I have lost a lot of time due to the way I began the project. Creating the scripts one by one made me end up with a lot of similar variables – they were made for the same purpose – that I had to erase when combining the scripts. On the other hand, if I had started designing the scripts from scratch at the same time, I assume it would be easier for me to link them all together at the end. This was certainly a lesson for me and the next time I take on an AI project I will start by designing the whole environment at the very beginning.

### New possibilities

I have definitely learned new things throughout creating the project. Apart from learning how to design the whole AI system before even starting the project, one of the main skills I have gained is to understand how the AI works and how to look at it from a different perspective – from the point of creation. Furthermore, I enjoyed the fact that before implementing anything, I had to think about thousands of possible outcomes and how to make them work the way I want. I have enjoyed running the same scenario multiple times and not knowing why something is not working properly or at least how I expected it to. This was a challenge I really enjoyed. There are a lot of possibilities within the Artificial Intelligence study, and from what I have learned, I assume that AI is not yet at its best and it will shock us all.

### A journey I enjoyed

First of all, I enjoyed the fact that I have done and learned something completely new. I have never investigated AI and thanks to the project I had to read and go through articles regarding AI and AI used in games. Thanks to that, I deepened my knowledge not only in programing, but also in a totally new field. I got the idea of how AI works, how to implement and how to expect it to work. Moreover, I have learned how much of AI there is in games that I would have never thought of. This helped me look on the games that I play from a very different angle. I will definitely try to work more with AI and create some new projects with different AI techniques.

# References

[1] Hachman, M. (2005). How AI Works In FPS Games - ExtremeTech. [online] ExtremeTech. Available at: https://www.extremetech.com/computing/75786-how-ai-works-in-fps-games [Accessed 11 Nov. 2019].

[2] Isla, D. and Blumberg, B. (n.d.). New Challenges for Character-Based AI for Games. [online] Available at: https://www.aaai.org/Papers/Symposia/Spring/2002/SS-02-01/SS02-01-009.pdf [Accessed 18 Nov. 2019].

[3] Mateas, M. (n.d.). Expressive AI: Games and Artificial Intelligence. [online] Available at: http://steel.lcc.gatech.edu/~mateas/publications/MateasDIGRA2003.pdf.

[Accessed 26 Nov. 2019]

Other readings and videos:

1. Herve-Victor Fomen (2018). Can artificial intelligence show real humanlike emotions? [online] Leyton Insights. Available at: https://www.leyton.com/blog/?p=3193-can-artificial-intelligence-become-emotional [Accessed 2 Dec. 2019].

2. Dill, K. (n.d.). What Is Game AI? [online] Available at: http://www.gameaipro.com/GameAIPro/GameAIPro\_Chapter01\_What\_is\_Game\_AI.pdf [Accessed 6 Dec. 2019].

3. Frontiers, A. (2018). Artificial Intelligence In Game Design. [online] Medium. Available at: https://medium.com/aifrontiers/an-overview-of-artificial-intelligence-for-video-games-f491229c0e7d. [Accessed 6 Dec. 2019]

4. Two Minute Papers (2019). OpenAI Plays Hide and Seek…and Breaks The Game! YouTube. Available at: https://www.youtube.com/watch?v=Lu56xVlZ40M [Accessed 11 Nov. 2019].

5. Arzt, S. (2019). AI Learns to Park - Deep Reinforcement Learning. YouTube. Available at: https://www.youtube.com/watch?v=VMp6pq6\_QjI [Accessed 11 Nov. 2019].

‌6. What Makes Good AI? | Game Maker’s Toolkit. (2017). YouTube. Available at: https://www.youtube.com/watch?v=9bbhJi0NBkk [Accessed 11 Nov. 2019].

Used models in project:  
Crate:  
<https://free3d.com/3d-model/wooden-crate-153647.html>

Soldiers:  
<https://free3d.com/3d-model/free-low-poly-rigs-soldier-73507.html>  
<https://free3d.com/3d-model/free-low-poly-soldier-28299.html>

Sounds:  
Gun shot:  
<https://www.youtube.com/watch?v=gFGVCCg9Y44>  
Grunt:  
<https://www.youtube.com/watch?v=5RW4Hfz-MJg>