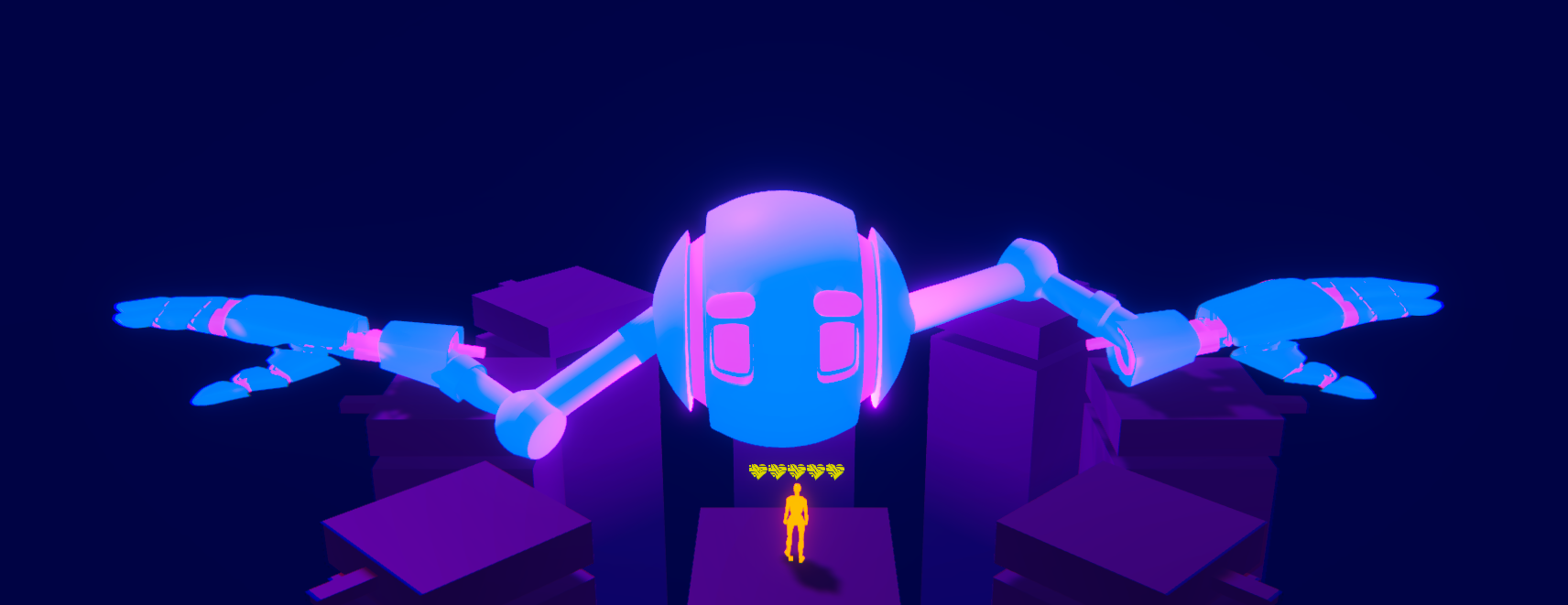
304CR Games and AI - Coursework 2 Report

Chris Bodsworth – Student ID 8522104

Video Link: https://youtu.be/umpEPirOnto

GitHub Build + Code: https://github.coventry.ac.uk/bodswor2/AICW2-Build-And-Code.git



-Intro

For my Coursework 2 AI project, I have created a rhythm boss fight platformer demo called “Neon Slap”.

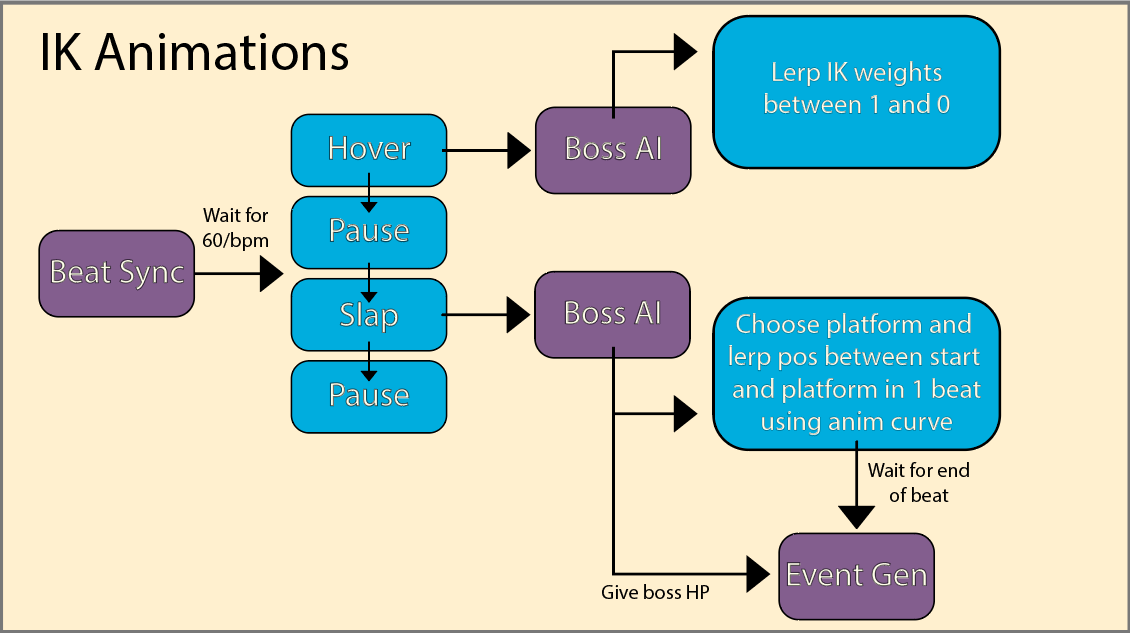
In this game, a colossal robot boss is in the middle of a set of platforms and the player platforms around it in order to avoid its attacks. The player can then summon bombs to damage and finally defeat the robot. The boss uses two distinct AI systems to defeat the player.

The first kind of AI that is introduced is an IK animation system. Using this IK system and a rhythm synchroniser, the boss selects a platform to attack and slaps it on the beat of the music. The boss, having two arms, does this for each of them.

The second system that is present is a fuzzy logic event generator. This system takes the boss’ HP into account and each event’s probability; the generator uses a list of fuzzy logic scriptable objects to equate whether or not each event should happen.

Throughout this report, I will be explaining both AI systems in a more in-depth manner, I will then highlight some key advantages and drawbacks of the way in which these AI systems are implemented, before finishing this report on a general reflection of the project.

-IK Procedural Animation



Let's start by outlining and explaining how the IK animation system functions in the context of this game.

To control the IKs of the boss’ rig, Unity’s Animation Rigging system has been implemented. This package allows the developer to control and manipulate a rig’s bones within the Unity Game Engine. When developing the boss’ procedural animation, the developer has to computationally decide where those bones should go and how the IK system behaves and reacts to what is going on in the game.

To keep the rigging simplistic, I sourced some models from Sketchfab and rigged simple arms in 3DS Max. The boss is simply a head with arms, so the rigging was very straightforward.

The head uses a Multi-Aim constraint to always look at the player. Each arms uses a Two Bone IK Constraint from its Upper Arm to its Wrist; each arm has a target that controls the placement of its hands. These targets are what we can move in code to control where the hands move, the rest of the arm follows because of IK.

The boss AI has two stages, these being hover and slap; the “BeatSync” synchroniser progresses through these stages, as well as pause stages, at every beat of the music and calls the appropriate functions in the AI script.

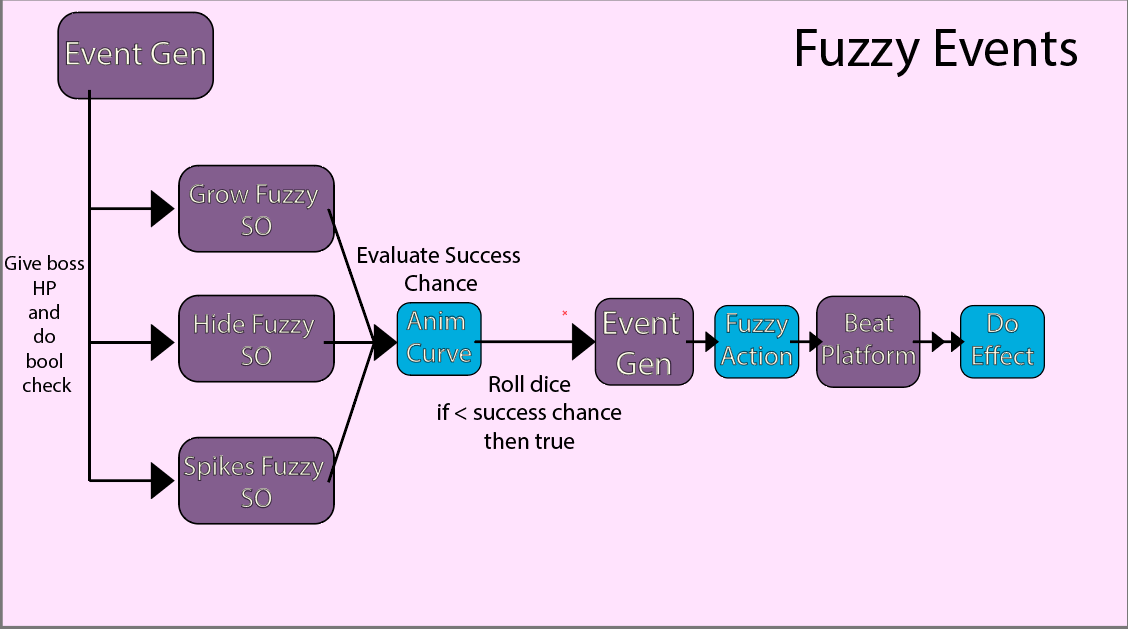
In hover mode, the boss simply lerps the IK weight of each of its arms from 1 to 0. By doing this, both hands will return to the boss’ Idle animation (waving its hands up in the sky). Lerping the weight in this way allows for smooth transition of the hand IK target from the platform position to the idle animation position over the time span of one beat.

Following this, the boss will enter slap mode. When slapping, the boss will choose platforms for each of its arms and will then lerp each hand’s IK target to the position of the chosen platform. Each hand has a “PlatformZone” which consists of a trigger collider that tracks what platforms are within its collider and makes a list of available platforms that the hand can slap. The hand then randomly choses a target platform from its zone’s list. By using these trigger zones, it ensures that the hand will only slap platforms it is near to and helps add logic to the game.

After having chosen a target platform, each IK target starts by lerping its weight from 0 back to 1 and then lerps it’s position between where it currently is in the Idle animation and the position of the target platform. When lerping its position, each hand uses an animation curve to control its speed. Controlling the speed allows for more realism, as the arms start by slowly approaching the platform before speeding up as they approach their targets. This allows for a greater impact on the platform.

Once the hands have finished their slap, they activate particle effects on the given platform before calling the “EventDo” function of the event generator. This leads to fuzzy events, which will be discussed in the next section of this report.

-Fuzzy Logic Event Generator



In this next step of the report, now that we have finished talking about the IK system, we will continue progressing through the pipeline of boss behaviours and will outline how fuzzy logic is used in the project. For the purpose of this project, we are defining Fuzzy Logic as the process of using probability curves to add randomness to the likelihood of an event happening, instead of using absolute values or pure conditional if statements.

In the game demo, Fuzzy Logic is used for a Fuzzy Event Generator that can do random effects on the game’s platforms based on animation curves and the boss’ HP. Our Fuzzy Event Generator has a list of fuzzy event scriptable objects that represent different effects that can happen after the boss has slapped a platform.

Each of these scriptable objects has an animation curve, a Fuzzy Action enum, and an isRandom Boolean. The animation curve will be used to dictate how likely an event is to happen based on the current HP of the boss. The Fuzzy Action defines what happens if the event happens successfully, and the isRandom Boolean defines whether the event should happen on the target platform provided by the boss AI or another random platform.

When the slap phase ends, this being when the hand has made contact with the platform, the boss AI calls the Event Generator’s “EventDo” function. This function causes the Event Generator to go through its list of Fuzzy Events and does boolean checks with each event’s “TryAction” function.

The “TryAction” function takes the boss’ HP as its parameter - which is supplied by the Boss AI script - then uses the scriptable object’s animation curve to evaluate a success chance value, this being a float between zero and one.

After evaluating the success chance for the given scriptable object, each object does a “dice roll” and gets a random float, also between zero and one. The success chance and dice roll values are then compared. If the dice result is smaller than the success chance, the event is counted as being successful and returns a true boolean to the event generator. If it is greater than the success chance, it returns a false boolean.

The event generator does this boolean check for every fuzzy event in its list. If a boolean returns as false from an event, the generator skips it. If, however, it returns as true, the generator will make the event happen.

For every true boolean, the boss checks if the scriptable object is random. If it is, then it will randomly choose a platform to affect out of all available platforms in the scene. Otherwise, it uses the one supplied by the boss AI. For the context of our game demo, all of our scriptable objects’ “isRandom” booleans have been set to false; this is so that the player can predict that only platforms targeted by the boss will have something happen to them.

Once it knows which platform to affect, the generator sets the target platform’s “currentEffect” enum and starts the correct coroutine with a set timer value in seconds as parameter. For this demo, the timer has been set to five seconds.

The target platform then changes the desired values, this being scale, presence of spikes, and visibility of the platform in its Update method. The started coroutine then waits the desired time before resetting the platform to an unaffected state.

With scriptable objects, the fuzzy event generator has been made in a way so that the number of events can be easily added to, and new effects can be easily written and set up. For the context of this demo, three events have been set up, these being as follows: growing a platform; making a platform disappear; or activating a platform’s spikes, which, when stepped on, make the player lose health. Each of these events last five seconds.

In the following section of this report, now that we have explained both AI methods in detail, we will discuss the main advantages and drawbacks of how they were implemented in Neon Slap.

-Advantages and Drawbacks

In this section of the report, I am going to be going over some of the main drawbacks and main advantages of the AI models introduced in this game.

Let's talk about the advantages and drawbacks in the context of the IK animation system. First of all, being able to control and manipulate the IK positions within the Unity Game Engine is a convenient solution compared to creating different animations for every platform.

Procedural animations via IK allow us to adapt the movement of the boss’ arms in a way that allows for greater scalability; the boss’ animations are independent from the number of platforms. In this way, we are able to develop more complex levels with a different quantity of platforms, and the developed system is able to adapt without any major code changes. All we would have to do is increase the quantity of platforms that the boss knows of. So the main advantage of using IK animation in this way is scalability and ease of use.

Now, as for the drawbacks of the IK system, the main one would be that, because of how we have set it up, we lose a lot of character and precise control of the boss’ animation. As it is not handcrafted and is instead made procedurally, we would lose a lot of personality and natural feel if we wanted to apply it to a particularly goofy character, for example. However, this is at least partially solvable as the Unity’s Animation Rigging system allows for the use of base animations that are only modified and guided by IK instead of replaced.

For the Fuzzy Logic system, there are also advantages and drawbacks for us to discuss. The main advantage of this system is that it is fully customisable by the developer; there are no set values or thresholds for an event to take place, and the animation curves are completely editable. As the animation curves represent a vast array of points and values, they allow for a wide range of flexibility in setting the requested event parameters.

Fuzzy logic also adds a lot of replayability and unpredictability to the game. By always having at least a very small chance that events may happen - even when the boss is at full health - the boss may do some effects much earlier or later than expected. This allows for every playthrough to feel different and limits the feel of monotony or predictability. In the context of a difficult boss fight, having variety in what may happen is essential. If the player is expected to play through the same fight countless time until success, it is paramount that systems like the Fuzzy Event Generator are implemented to keep the player engaged. So the main advantages of fuzzy logic in the context of Neon Slap are customisability for the developer and variety for the player.

There are still drawbacks to such a system. We have limited the amount of excessive randomness in our game, but it would be very easy to see how complete randomness and fuzzification of boss behaviour could dampen player experience. In the context of a boss fight, we want to add randomness to keep things interesting; however, we need to do it in a balanced way so that the player does not feel cheated or frustrated. Main rules and patterns still need to be closely followed so that the player has the possibility to learn and become better at telegraphing the boss behaviour.

Generally speaking, with all AI systems, balance is key. All systems in a game must exist with the aim of increasing player enjoyment and experience. Whether through Fuzzy Logic, Procedural Animation, or other AI elements, like State Machines, they must all be applied in a cohesive and balanced manner so that the player has an engaging player experience in a state of flow without getting too frustrated or ever feeling cheated. In short, AI must work in cohesion to enhance player experience.

-Reflection:

As we come to the end of this project, in this last section, we will reflect on how the project went, what went well, what could be added, and also, generally, what future works would need to be implemented in order to take this project to commercial release.

There are various elements of “Neon Slap” that we have not discussed throughout this report. The main undiscussed extra functionalities are how the player summons bombs, how the non-AI game logic works, as well as various other small game elements.

In this report, we focused on the AI systems that are in play, discussed in what ways these systems were advantageous to the project, as well as in what way they could, when unmanaged, just as easily become detriments to the game as a whole.

If we were to expand the project, there are several elements that we would consider adding. The first of these would be more phases for the boss; we have created the boss’ behaviour in such a way so that it is easily expandable, modifiable and can easily accept new moves.

When we had previously planned the implementation of this game, the boss was planned to not only be able to slap but also punch and shoot at the player. There were plans for more complex behaviours to increase the player’s level of challenge. To enable all of this, a more developed finite state machine would have to be implemented, and this was deemed unnecessary for the scope of this demo.

As a whole, I am very satisfied with how this project went and the end product produced. I believe that this would be a good portfolio piece, and the experience of creating a game development video was very valuable.

Below are references for assets and resources used, a link to the YouTube video where I showcase and explain the project is also included with this coursework submission.

Thank you for reading my report. I look forward to seeing how this project develops in the future.

Bibliography:

Robot head model:  
Robin Butler on Sketchfab.com | <https://skfb.ly/6vqC7>

Robot hand model:  
Sean Nicolas on Sketchfab.com | <https://skfb.ly/6XLBS>

Background music:   
ENDGAME by Alex-Productions | <https://www.youtube.com/channel/UCx0_M61F81Nfb-BRXE-SeVA>

Boss shield icon:sbed on game-icons.net | <https://game-icons.net/1x1/sbed/shield.html>

Platform grow icon:  
Delapouite on game-icons.net | <https://game-icons.net/1x1/delapouite/expand.html>

Platform hide icon:  
Delapouite on game-icons.net | <https://game-icons.net/1x1/delapouite/invisible.html>

Platform spike icon:  
Delapouite on game-icons.net | <https://game-icons.net/1x1/delapouite/spiked-wall.html>

Player heart icon:  
Lorc on game-icons.net | <https://game-icons.net/1x1/lorc/techno-heart.html>