<https://www.youtube.com/watch?v=PBYKqvDK8d8> – Music

**Will be using custom annotations (Skip icons)…**

**My own level design, ledge climbing, gameplay something of my own in the background ---**

* Ludum Dare
* FOV
* New story
* Ledge Climber
* Concerns
* Level Design
* Quad Trees

Hello YouTube, my name is Ajay Venkat and you are watching part 3 of the development log series where I am building and Open World game using Unity and hopefully teaching you some stuff along the way. Today I will be updating you on the current progress of the ledge climber system, the FOV on my NPC’s and how I have been personally growing since the last time I have made a video, this video is going to be feature packed.

Before we begin I want to take a quick moment to thank every who has subscribed, we just reached 500 subscribers which means so much to me. If you haven’t subscribed already, you should do so now to keep up with some awesome and original content. Also make sure to keep up with my development blog at codewithajay.com where I post detailed and regular updates on systems that I am creating and without any further ado let’s get right into the video.

Firstly I would like to reveal some more information about the overall story of the game, I did touch base on this in part 1 of this series, so if you haven’t seen that make sure you do. Firstly something you should know about the world that we are set in, is that these creatures are not new to humans – they have been living alongside each other for many centuries. But unfortunately humans decided that they had more rights to the planet than these creatures and decided to abuse, torture and sell them for money… One that fell victim to this pain is our antagonist of the story, she was brought up in slavery only to watch her family being killed by humans, with all her rage against humanity she escaped and started a small uprising. Seeing that one had the courage to go up against these humans, the creatures finally began to rebel against humans, which grew more and more and eventually turned into a full-scale war.

The story is set during the final stages of the war, where the creatures are on the verge of defeating the humans and the protagonist must find a way to save all, but I will not reveal any more of the story than I already have… I will now tell you about this month and everything that I have done since the last video.

Let’s start on July 9th, this was my birthday and I just turned 16 and started the first NPC mechanic of the game, which is Field of View for the game. Field of view can be viewed as quite a simple mechanic until you go into the depths of how to make it fun and interactive, so I decided to do a bit of research into how I can make the field of view interesting and this is what I did.

Traditionally how my FOV algorithm worked was, it would check if the player is in the view radius then check if the player was in the view angle and if all above is true then it sent a ray towards the center of the player and if it hit the player then we decided that the NPC could see the player, but honestly this is very boring and unrealistic. What is there was a cube that blocked the center of the player’s body, then that would mean the NPC, technically couldn’t see the player. So what I did was took the player capsule and added 4 relative points onto the capsule. One which referenced, the head, the feet, the core and the neck. Now that everytime the NPC has the player in the view angle, instead of sending one ray only to the center of the player, it sends 4 rays to each reference point.

Now no matter, what part of the body is showing, the NPC can detect the player – but this also gives us extra control. Before the visibility was either 1 or 0, true of false but now we can say – if 2 of the 4 points are found set visibility to 0.5 or if all 4 points are found set visibility to 1 and this just gives us so much more control when it comes to the behaviour editor, so once I finished and polished up that system I jumped right into the task I have been waiting to complete for a long time… The ledge climber system.

Now I made a promise to myself before beginning this, and that promise was – I would not focus on animations, or realistic movement from ledge to ledge. The goal before the 11th of August was to get a functioning editor script where I could go through and tag ledges on objects, allow the player to connect to ledges and move across them and traverse the scene and I can proudly say that, that goal was accomplished. Now let me get into the details of how this system works, if any of you want to replicate this.

Currently I am at the finishing stages of the system, I have managed to create an editor script which can take any mesh and detect it’s vertices and allow me to click and tag any of them as ledge points. A ledge is built out of specific ledge points that connect with each other, the player is able to traverse along ledge points, but if they want to connect from one ledge to another, there has to be a pre-defined connection created inside the editor and I am proud to say I have made this entire process quite easy. If a point has multiple connections it can go to, it looks at the input and automatically decides which connection is best, based on this input. The input vector is normalised, then the direction between the connections are normalised and the closest connection direction to the input vector is selected.

So all together, I can tag ledge points, create connections, have my player stick to ledges and traverse along them and finally jump between ledges. Let me break down the system a little bit. –

So what is a ledge point, well a ledge point is just a vertex index of a given mesh, using the vertex index assigned to the ledge point, I can get the location of the vertex no matter how the object is transformed or changed which allows for maximum scalability. The ledge point also stores a reference to the ledge connections it has and it’s parent ledge. The parent ledge is composed of an array of ledge points and can either be a looped ledge or a non-looped ledge, the difference being object that have a connection present between the last and the first point – this is important while processing the movement between ledge points.

So how exactly does the player latch onto a ledge, well when the player jumps I use the velocity of the player to detect where a ledge could be and send a ray towards the object. If the object by chance has a ledge then the closest ledge towards the player is calculated. Then further calculations are done to find the two closest ledge points in that ledge, then to create a seamless latching onto the ledge I automatically calculate around 10 temporary points in-between the two ledge points, allowing the player to latch onto any of them. You can see this demonstrated in the background, but the blue cubes that appear as the player moves.

In essence this is how the entire system works, but creating this is only half the challenge – I still have to tackle animations, IK and procedural movement and I will tell you now, it will be a massive challenge – that’s why I have put it off so that I can get some more core systems complete, to get me closer towards the vertical slice that I am expecting. For those who are not familiar with vertical slices, it is just a sample of your game, where you can show case your mechanics and make sure the concept of your game is fun and all the mechanics and features work well together. It gives you a chance to test your game and even play around with a bit of design, but I still have a way to go until that.

Over the last month I have also been doing a lot of learning on how I can prepare myself for the future of this game, and one big thing is optimising Unity for open worlds, because strictly speaking – Unity wasn’t built for massive open worlds out of the box, so we have to be smart about how we go about optimising it. I was researching about the Quad Tree structure and how I can implement this for the level design – The quad tree is a space oriented way of storing data and it’s very efficient and I believe a lot of collision systems are built using it, for optimisation. So what I think I can do, correct me in the comments if I am wrong – but I can create the game level is massive chunks and then allow a Quad Tree to segment these chunks based on the game object density so that I can know which parts are more populated and I can quickly find out where and what can be un-loaded and loaded back into the scene really quickly. This is just purely conceptual so far, I have had other ideas for optimisation but this is looking pretty promising at the moment.

Unfortunately it may still take a little while before I get to world optimisation, but I have been doing a lot of learning about level design and conveying mood, story and emotion through the level design. I have primarily been finding ways to link the current story with the level design and I am in the midst of creating the map of the game but I don’t want to show it, until I have completed it fully, but in general that’s pretty much a big chunk of what I have been doing for the past month, I apologise for the long upload duration, but I did want to finish the ledge climber system to show some progress, and I am glad to say that the project is going on track and on schedule so far. I did manage during this month to complete a quick Ludum Dare, which was super fun – you can find the link to the game, down below.

In the next episode I should have gotten further into the NPC mechanics and hopefully could have started the world optimiser, so if you are willing to see that make sure to smash that subscribe button and join this little community of ours. If you did enjoy yourself here or learnt something – show some support by hitting that like button. I thank you all again for 500 subscribers, and until next time – stay awesome.