NEWM-N320 Midterm Paper

Student: Jason Hampshire
Professor: Mathew Powers
Date: 3/29/2021

1. Narrative:
   As far as narratives go, I do not have a clear path for where I want to go with my level. I just started messing around with the different land tools and made a 2-tier pit that I thought looked cool. The first version that I turned in for "And so it begins….." mentioned making the end goal being probably escaping the pit, and that will still probably the case. Over time, the pit turned into a cave and went from a lighter atmosphere with water to a darker variant with lava. This shift in dichotomy compelled me to think of the level as a prison. At the moment, the very tentative narrative goes along the following points:
   - The cave where the level takes place is a high-security prison where criminals are thrown to never see the surface again.
   - The prison is named *Nec Evadere* which is Latin for "no escape."
   - The prison is said to exist on the verge between hell and earth. There are rumors that you can hear the screams of the damned if you get close enough to the bottom. Some even believe that the massive pool of lava at the bottom is actually just a portal to hell.
   - There are two tiers to the prison.
     - The top tier is for criminals that got sent to the pit for less serious crimes (the crime still has to be pretty dang serious to get sent here though)
     - The bottom tier, the one chilling right over the pit of lava at the bottom, is reserved for the truly evil of the world. Psychopaths that show no remorse for what they have done, the real Hannibal Lector type of characters. If you killed half a dozen people, you would not end up down here, but if you killed 60, tortured them and, I don't know, force-fed victims the bodies of earlier victims or something like that, then you'd end up down here.

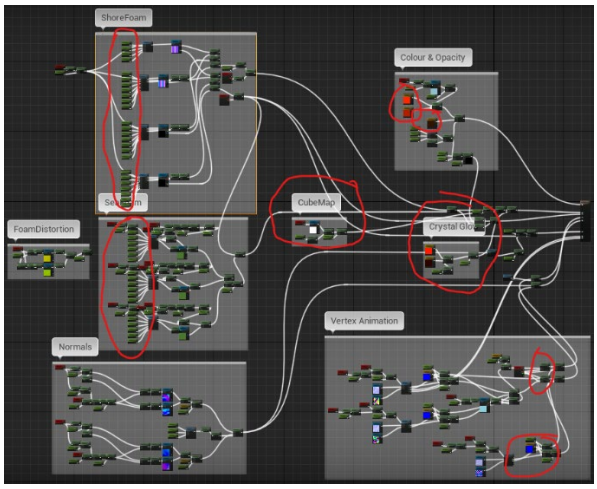
2. Situation/Objective:
   - The player's stake in this:
     - The year is in the early 1800s in Italy, located close to Naples. The saying "see Naples and die" is taken to a pretty literal extreme here instead of the artistic sense, as the prisoners that enter *Nec Evadere* are destined to never come out, living or dead.
     - You are a criminal, bad enough to get sent to *Nec Evadere* but not bad enough to be sent to the lower level. Your crime doesn't matter. The point is, you are not a very nice person and you deserve to be here.
     - You wake up in a cell on the upper tier. The door is locked tight, and the task will be to escape *Nec Evadere* .
   - I'm probably gonna add a twist at the end just for the fun of it where you think you escape, but in reality you are still stuck, or you die.

3. Indoor area:
    - The indoor area in my level is a bit complicated, as you could argue that my entire level is indoors.
    - The area that is indoors is an alcove along the wall made with a mixture of additive and subtractive BSP to make a room that looks like the inside of a square pyramid.
    - Currently, this area is planned to be one of many things:
        - The entrance/exit
        - A prison guard room of sorts
        - A storage room, possibly food storage
        - An "interrogation"(torture) room where prisoners are brought by the guards.
    - The room will be important to the flow of the narrative, I just don't know for sure how yet.

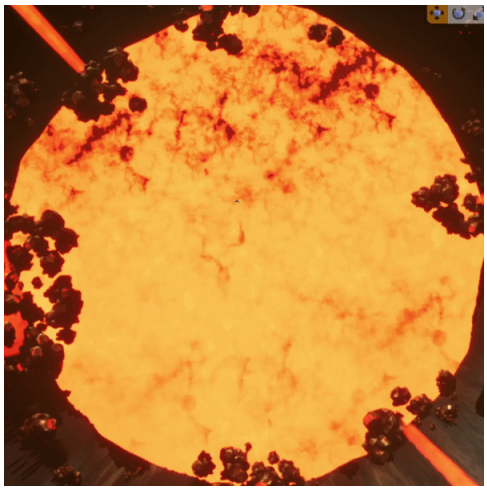
4. Outdoor area:
    - The entire rest of the level is "outdoors.
    - The prison's main room is the outdoor environment.
    - There will be cells evenly distributed along the walls of both tiers.
    - If you jump in the lava, you die.

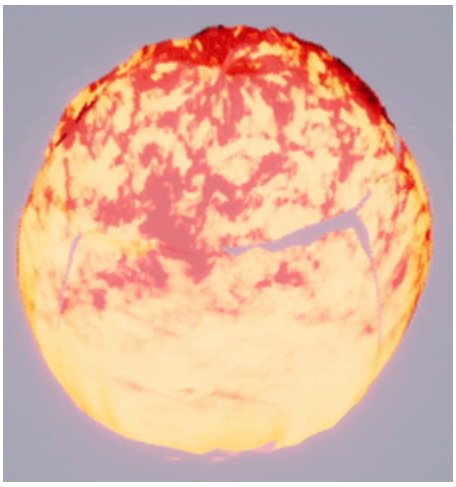
5. Environment:
    -
    - I made the environment a bit too large, taking up 17x17 landscape component squares.
    - The entire landscape is dominated by the basalt texture with bits of sandstone, slate, and other textures thrown in as well.
    - It is shaped like a two-tiered circle with sheer cliffs going from level to level.
    - The lava pit at the bottom is the lowest point the landscape sculpting tools will allow.
    - The upper tier is right about the height that typical levels are at when created.
    - The top is very close to the max landscape world height.
    - The ceiling of the cave has a massive, jagged crack that allows you to see the sky
    - There are 5 lava-falls that go from the top to the upper tier. There are 5 lava-falls that go from the upper tier to the lava pool at the bottom.
    - The upper tier is divided into 4 larger sections and 3 small sections by lava rivers.
    - The playable area will probably only happen in one of the larger sections since the world is so large.
    - Blocking and post process volumes around the lava.

6. Other Elements:
    - The lava material:
      When I tell you that this stuff was a pain in the butt to get working the way I wanted, I would be understating it. To start, I had to find a base. I ended up using the ocean material from the WaterMaterials pack and spent hours of trial and error to get everything to look the way I wanted.

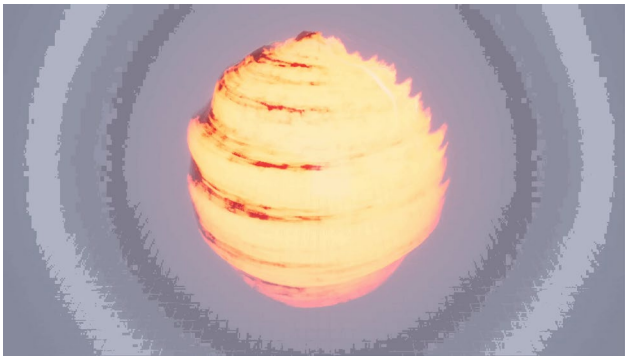
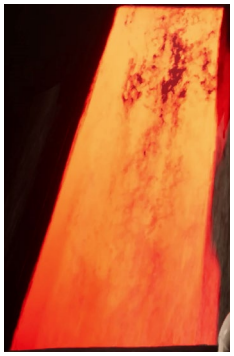


The circled areas on the blueprint are the ones I had to either edit or add myself. If you would like to see a clearer version to actually see what each node is, contact me and I can get you that. I turned the emissive layer from 10 to -40 to make the seafoam look black. I created the crystalGlow comment section and fed these colors into the emissive node as well to get the nice orange glow. I won't go further into the specifics, but eventually I was able to make the lava look as I wanted.



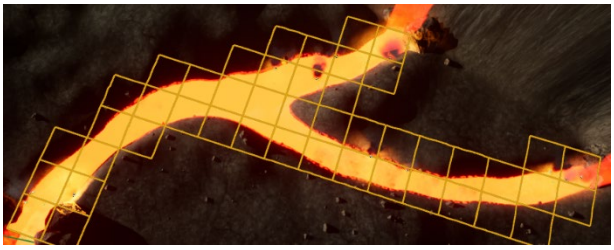
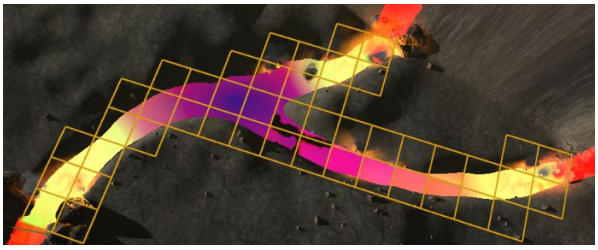


- Lava-falls:
  I again utilized the WaterMaterials asset pack as the base. I used the waterfall static mesh unaltered, but I had to do some major modifications to the material. The blueprint looks very similar to the blueprint for the ocean material I had already used so I was able to get the material and material instance figured out a lot quicker

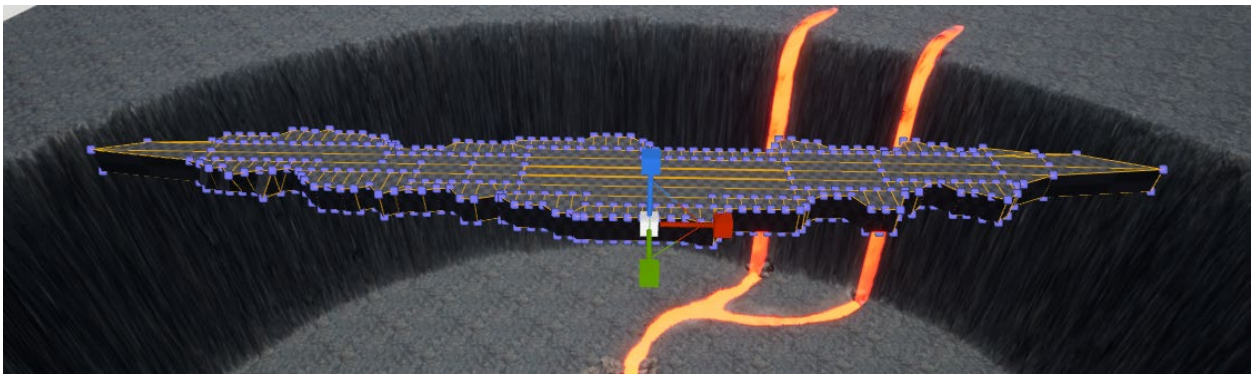



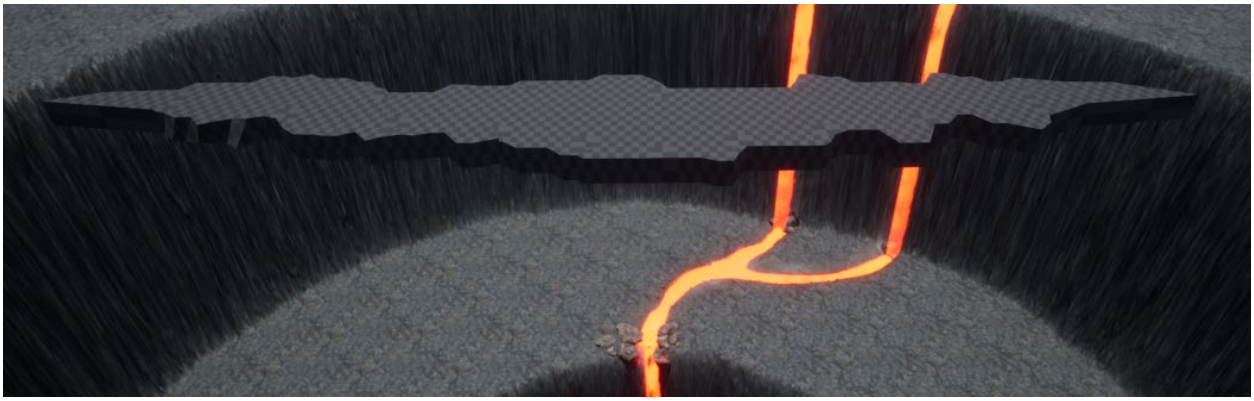

- Lava river:
  The lava was again a very large undertaking. Again, using WaterMaterials, I did quite a bit of research on how to best utilize their ocean static mesh for my level. I found out that the way the developer intended the mesh to be used was as a tile of many other of the same mesh. This resulted in me adding over 200 individual instances of this mesh. I also found some information about vector painting. Initially, I had no clue what this was, but after some more research, I figured it out. Basically, each RGBA channel can feed a value into a blueprint and you use the Mesh Paint mode to accomplish this. The material's vector painting aspects were as follows:
    - Red = Subtle Wave Normals
    - Green = Heavy Wave Intensity
    - Blue = Swap Edge Normals
    - Alpha = Vertex Movement Intensity

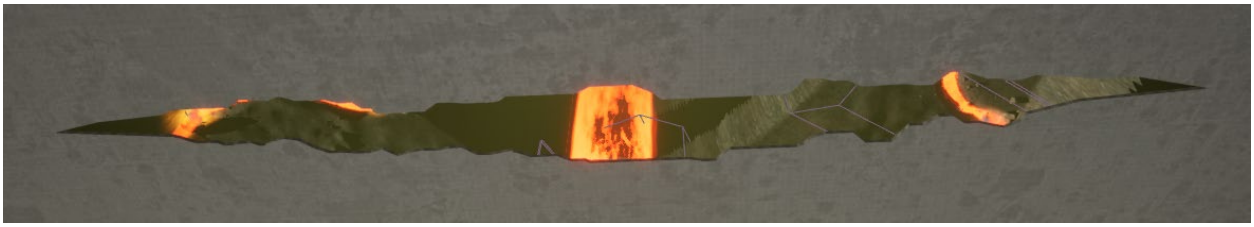
  Using these settings, I was able to dynamically edit the material's properties and make each instance of the mesh turn out the way I wanted. I ran into an issue with the UVs of the mesh/Material resulting in the waves going the wrong direction, but after some help from Emily, I was able to find a resolution to this issue.

- The crack in the ceiling:
  After already spending more time than I should have on a single aspect of this project, I decided it was a good idea to do something else that we haven't learned yet. I wanted a crack to be in the ceiling of the cavern, and to accomplish this, I wanted to use a subtractive BSP. Typically, BSPs are very limited on how you effect their geometry… until you start using Brush Editing mode. I will spare you the details, but after many more painstaking hours of extruding and editing vertices, I was able to create a disgusting asymmetric nightmare of a BSP that worked very well to create the subtractive BSP I wanted.

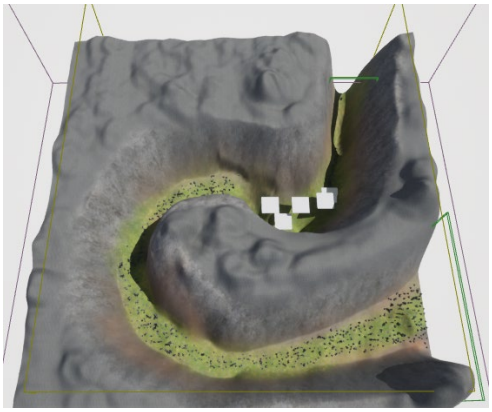






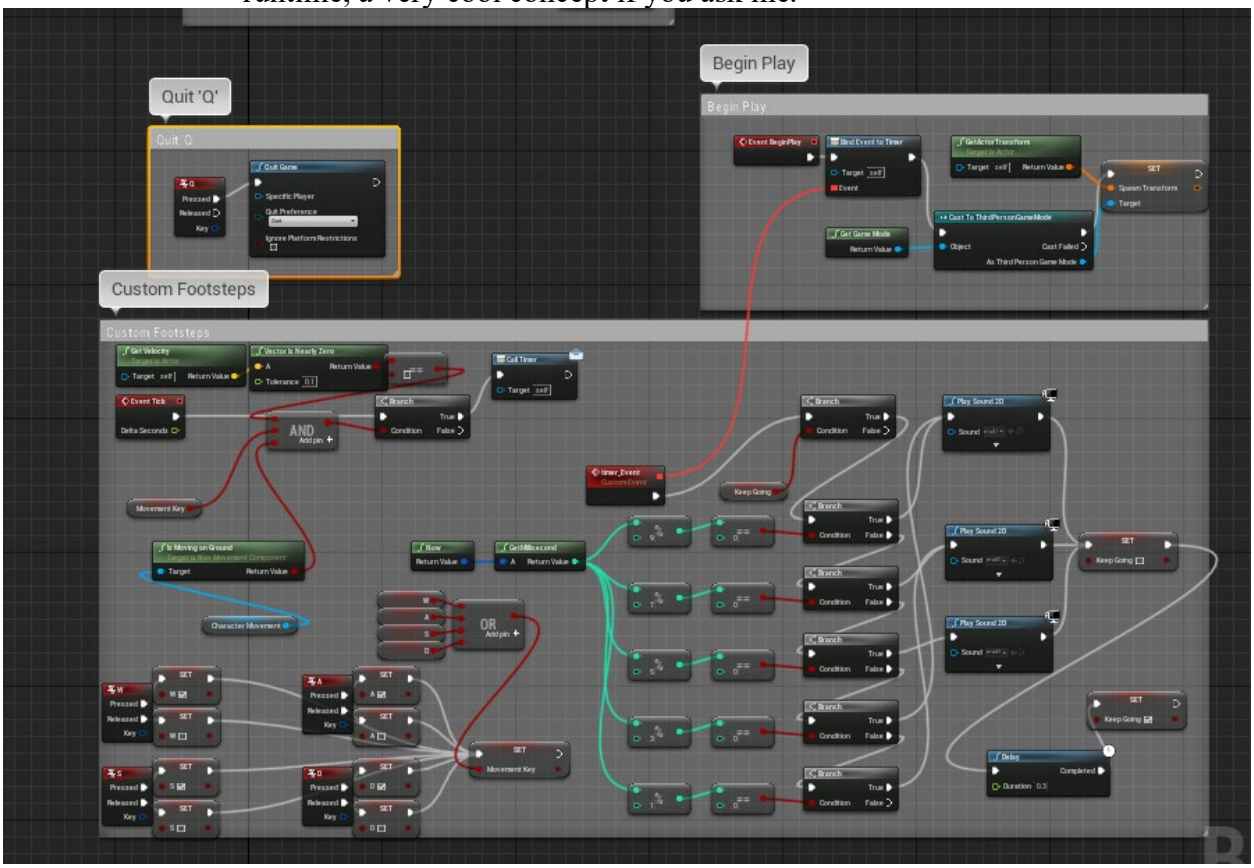
- Sounds:
  There is not a bunch to say about sounds in its own section other than that I added a few. I will discuss some in the Block-coding section instead of this one due to their interwoven nature. I added an ambient track that uses a sound cue to mixes some eerie SFX with a light lava SFX across the entire level. I then also added attenuated ambient lava SFX around both player accessible lava rivers and a cave sound cue to the inside portion of the level.

- Faux level:
  As a joke, I recreated the bad midterm level that was shown in class a few weeks back as the starting point for my project. I recreated it the best I could from memory, but I intentionally only spent less than ½ an hour working on it, just to give it that rushed, last minute feeling. I also removed the texture from the foliage to be as true to source material as possible. Since I did not want to be stuck in this area, I added a trigger volume behind the spawn point and at the end of the valley. This will be discussed further in the Block-coding section.



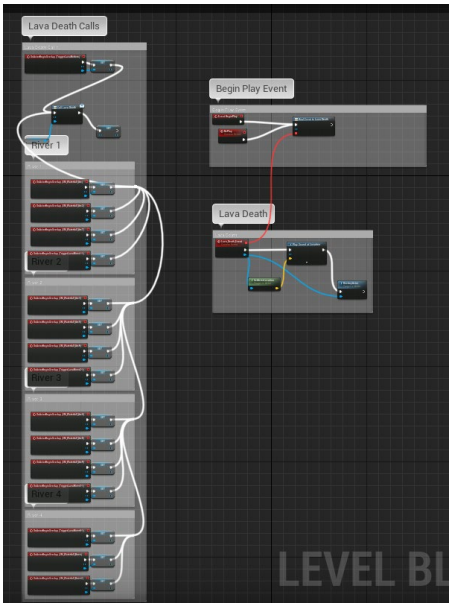
- Block-coding:
  Now comes to the part that I spent the most unnecessary time on. I spent days and days working just on block-coding, something that was in no way a requirement of the assignment.

    - Begin play event and event dispatchers:
      Most, if not all, of my block-code relied on these to function. Event dispatchers are similar to functions, at least in my mind, and while using them was not always the quickest solution, it will allow for them to be used by other sections of block-code in the future. The begin play event is called when the actor begins play; very self-explanatory. I would use this event to bind the event dispatchers to their corresponding event. The begin play event is also used to do anything that needs to be done right when the level starts. In my implementation, the assignment of event handlers seem quite unnecessary, but I can see how this would allow the same dispatcher to be re-assigned during runtime, a very cool concept if you ask me.



    - Footsteps:
      I spent probably 3-5 hours just on the footsteps, something that I am not proud of. Of course, I found out after I finished creating a miniature nightmare that you can just bind the SFX to the walking animation instead of all the stuff I did. A quick rundown of how this all works:

- When the level is started, the timer event dispatcher is bound to the timer event.
- Event Tick is called constantly, checking if the character is on the ground and if the character's velocity is not very close to zero.
- Separately, there is a Boolean value that is being used to see if any of the movement keys, WASD, are being pressed. If any are being pressed, the Boolean is set to true, otherwise, it is set to false.
- If the character has velocity, is moving on the ground, and one of the movement keys are being pressed, the timer event dispatcher is called and triggers timer_Event. I realize that the dispatcher could be cut out here, I just wanted to implement it this way now in case I want to bind the sound to animations.
- Timer_Event first checks to see if a Boolean called keepGoing (a homage to andy Harris if you know/have ever had him). This Boolean stops the event from triggering if the event has not finished a previous instance.
- The system time is then used in modulus division to choose which of three sounds will be played in an if…elseif…else structure.
- After the sound is played, the keepGoing Boolean is set to false and a 0.3 delay is started. After the delay, keepGoing is set to true. This keeps the footsteps semi-consistent with the animation.

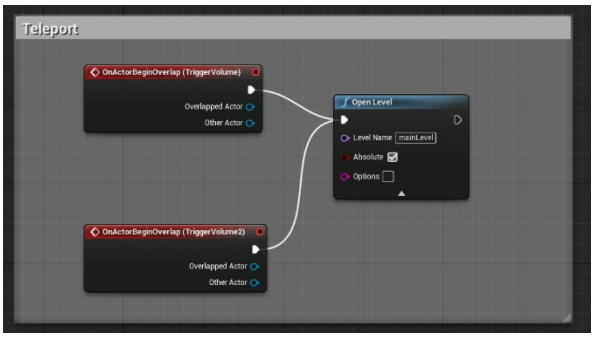


- Lava Death:
  Here is the block-code that I arguably spent the longest on and am still unhappy with how it turned out. Does it work? Yes. Does it bug out? No. Then what is the problem you might ask? It is implemented incorrectly. Yes it

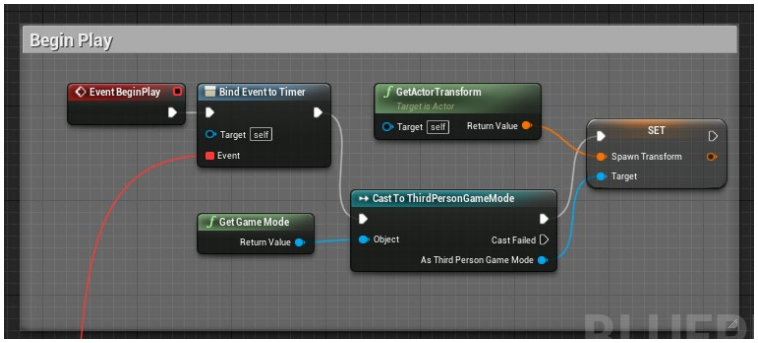
works without issue, but it's sloppy and done with a bypass instead of how I believe it should be done. The section that was not done the correct way will be bolded. Here is a rundown:

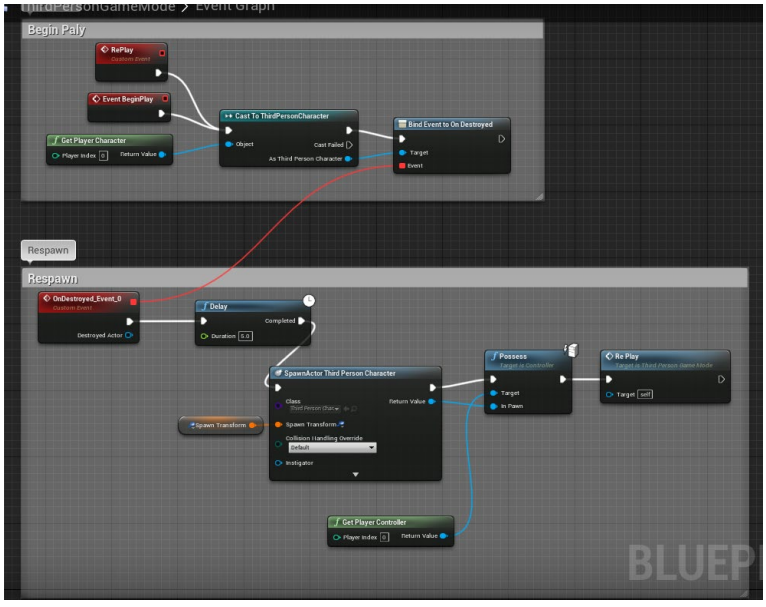
- On begin play, 'Lava_Death' event dispatcher is assigned to 'Lava_Death_Event.'
- Trigger volumes have been placed over the lava rivers and pit. Each of these volumes along with the collision of the lava-falls have an 'OnActorBeginOverlap' event tied to them, 16 in total.
- **Each event feeds into a 'Set' node that sets the actor variable 'actorToTarget' to Other Actor (which is the player characterin this case).**
- The 'Exec' output of all the nodes then feed into 'Lava_Death' event dispatcher call, **along with 'actorToTarget'.**
- 'Lava_Death_Event' activates, being passed the player character actor. The actor's position is retrieved.
- The 'Lava_Death' sound cue is played. This cue is a scream along with a lava splashing SFX.
- The player character actor is then destroyed.
- 'actorToTarget' is then set to NULL

The reason that using an actor variable is problematic is because it should not be necessary. What I was trying desperately to do is feed all of the 'Other Actor' nodes into the event dispatcher, just like all of the 'Exec' are. This would eliminate all of the 'Set' nodes and remove an unneeded variable from memory. When an attempt to feed all the 'Other actor' nodes into the dispatcher is made, it will only allow one. Every subsequent link attempt is met with 'replace existing input connections' message. Multiple actors CAN be fed into specific nodes, such as the 'Destroy actor' node. I believe this is because 'Destroy actor' does not produce an output from the actor input, only destroys it. The closest I got to making the blueprint function the way I want was to check to see if an actor was NULL or not, since only one event should actually have an actor output. This was unsuccessful, so I cut my loses and left the actor variable in.



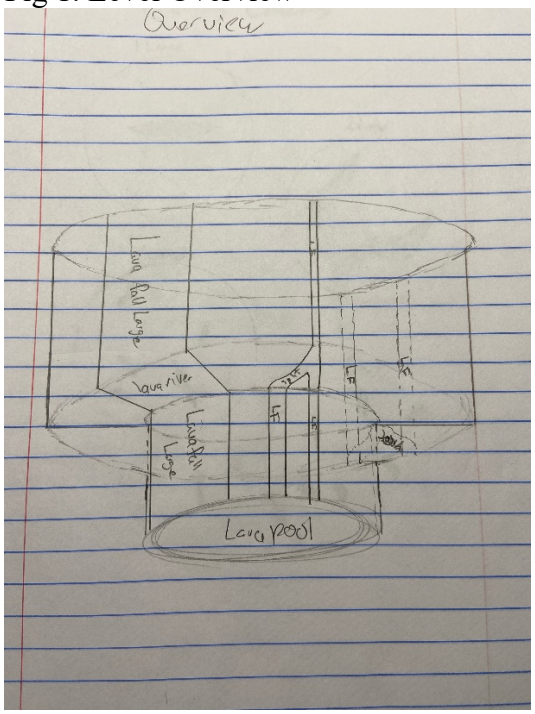
- Respawn:
  Respawn must work across two different blueprints, most of which is in the ‘ThirdPersonGameMode’ blueprint with a little in the ‘ThirdPersonCharacter’ blueprint. There are multiple ways of implementing this, some making the player lose collision, disappear, and teleport to the spawn, and others actually destroying the actor. I went with the destruction route. All the happens is that when the character is destroyed, a 5 second delay is started. After the delay, the player is brought back to their starting location. Here is a quick rundown:
  - When the level is started, the player is bound to an ‘OnDestroyed’ event in the ‘ThirdPersonGameMode’ blueprint.
  - At the same time in the ‘ThirdPersonCharacter’ blueprint, the position of the character (self) is set to a variable located in the ‘ThirdPersonGameMode’ blueprint.
  - When the character actor is destroyed, a 5 second delay is triggered.
  - After delay, a Third Person Character is spawned at the starting location that was saved from the ‘ThirdPersonCharacter’ blueprint
  - The player then possesses that character and the ‘mainLevel’ blueprint is recalled for security.





7. Sketches:
   All I have to saw as a preface is that I am not a drawer. I work on computers and very write anything by hand.

Fig 1. Level Overview



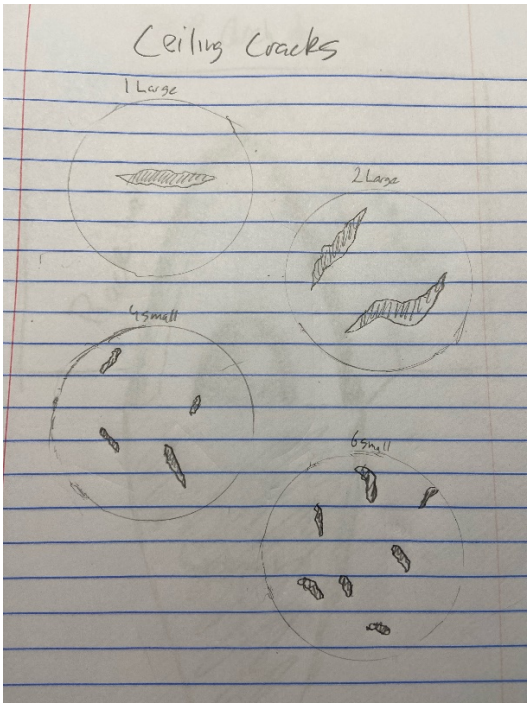


Fig 3. BSP Archway