Postmortem Report

Mark Graven

Southern New Hampshire University

**Contents:**

**What went right --------------------------------------------------------------- [**[**Page 3**](#RTM)**]**

**Bug Testing / Reporting --------------------------------------------------------------- [**[**Page 5**](#Bugs)**]**

**Runtime Error Handling --------------------------------------------------------------- [**[**Page 5**](#Bugs)**]**

**Control Setup/Interface --------------------------------------------------------------- [**[**Page 6**](#Controls)**]**

**Image References --------------------------------------------------------------- [**[**Page 7**](#Images)**]**

**What went right**

1: **Simple but identifiable “Fuzzies”**

While the fuzzies aren’t exactly fuzzy, in the strictest sense, the two models chosen to represent fuzzies were custom made and are clearly identifiable to virtually all players. This becomes particularly evident when the player initiates the objective/quest at the farm and they are prompted to pick up X amount of Giraffe fuzzies, and Y amount of Pig fuzzies. With that information on the screen, the user can clearly identify which is which, and is able to make the decisions to chase the right fuzzy down.



They also have personality, albeit simple, that has them roam around within their paddock/fenced area, which adds to the randomness of where you’ll find each individual fuzzy. Even if they initially spawn near each other, each one will choose a random place within their fenced area, and it was uncommon that two were right by each other.

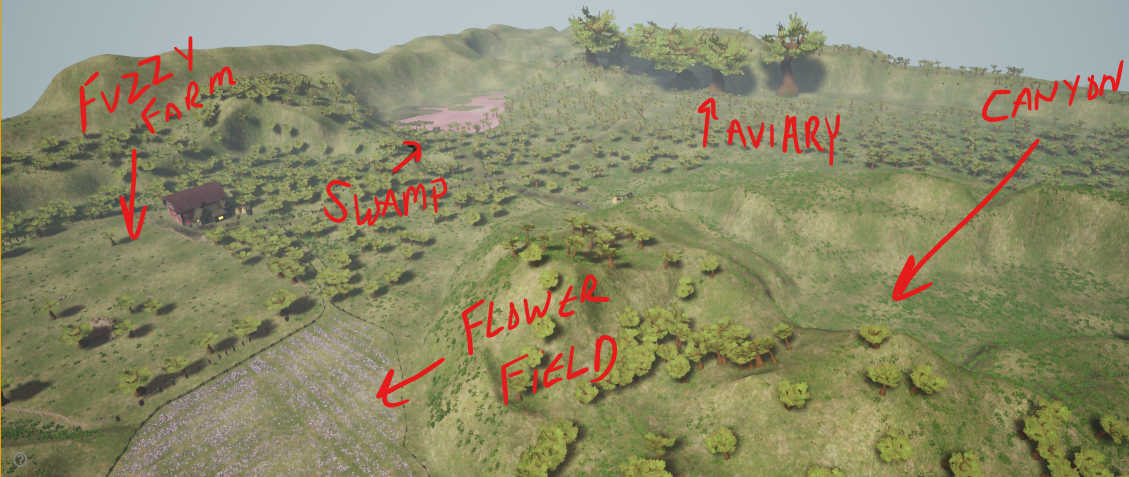
Also included was a random spawn feature that’s zones were used to ensure they wouldn’t spawn in/on trees, or high up on rocks, or outside of the fenced level. When you combined the random spawn and random roam, it truly made for a unique play experience every time.

**2: The Landscape**

The original concept of the gameplay field was simple enough and was done by creating a large enough navigable area for the player to chase down the fuzzies. Because (aesthetically) just having the single navigable field to explore is unpleasing, I added some rolling hills outside of the fenced playfield, as well as other mini bio zones like a swamp w/water, an aviary with huge trees, and a canyon area. All of this came together smoothly and was easy to accomplish using one asset pack from the store (Advanced Village Pack (AVP)) that kept the footprint of the project small and manageable.

I also created a simple but effective landscape texture so that most of the playfield was on top of a grassy surface. Adding in the various grass foliage from the AVP, with a dusting of trees, it made for a relatively pleasant combination of forests and fields. The swamp area utilized much of the same assets and was easy to create using a water texture that I had for a different project. I just had to apply that texture to a plane that was large enough to cover the swamp area I wanted it for and boom, instant swamp!

The aviary came together easily because I was able to use the same assets from the AVP which I scaled up and out to give an awe inspiring sensation to the habitat for the flying fuzzies (not implemented for this prototype). Additionally, the canyon was simple to create by using the basic landscaping tools provided within unreal.

Since the Slithering Swamp, Critter Canyon, and Forrest of Flight weren’t part of the requested prototype, I ensured that they were only basically built as to not expend undue time on them, and created invisible blocking volumes to prevent players from navigating into the incomplete areas. Additionally, since pathing / exploring to these areas can take a few minutes, I added in a simple teleport/transport mechanic to move players back to the starting/load in position if they were in face in the “out of bounds” region of the prototype. 

**3: Sounds**

The sounds for this came together very quickly and had I had a lot of fun with them. To start with the main sound, the music, I found an artist that has a ton of music that is free to use (with attribution) and he had this quite enjoyable piece that I used that fit the profile for a fun farm game. It is called Banjos Unite, by Alexander Nakarada used under CC 4.0 license (<https://www.serpentsoundstudios.com/royalty-free-music/comedy>). For the basics of the level (as the player runs around) the music is perfect and very peaceful/fun, but it was too slow for the timed event! A simple fix for it was to speed it up! I used the same WAV file and created a new cue with a modulator and pumped the pitch up (which naturally increases the music speed) and bam, a hasty frenzy collection music was born, and it worked beautifully. Not only did the music match the intent, but it worked perfectly for the desired length of time when the player would run out of time. The music ends just after the user is prompted with the menu options to quit or retry and I don’t think I could have planned that out any better if I had tried. Even better, when the song begins it has a distinct 3 beat drum intro that replaced the need for a SFX used to signal the beginning of the round. It was a great find that really helped to bring my project some fun life!

Regarding the SFX I really felt like keeping it minimal, while also adding to the feel. The first SFX that came together nicely was a simple one from freesound.org that is a low to high tone change that I used to signify the pickup of the fuzzies. Since the footprint of this particular file was very small, I used audacity to create two variations of the sound to use on the two different fuzzies. Though the two were similar, they were distinctly different, so players will know that they’re picking up different fuzzies just from sound alone.

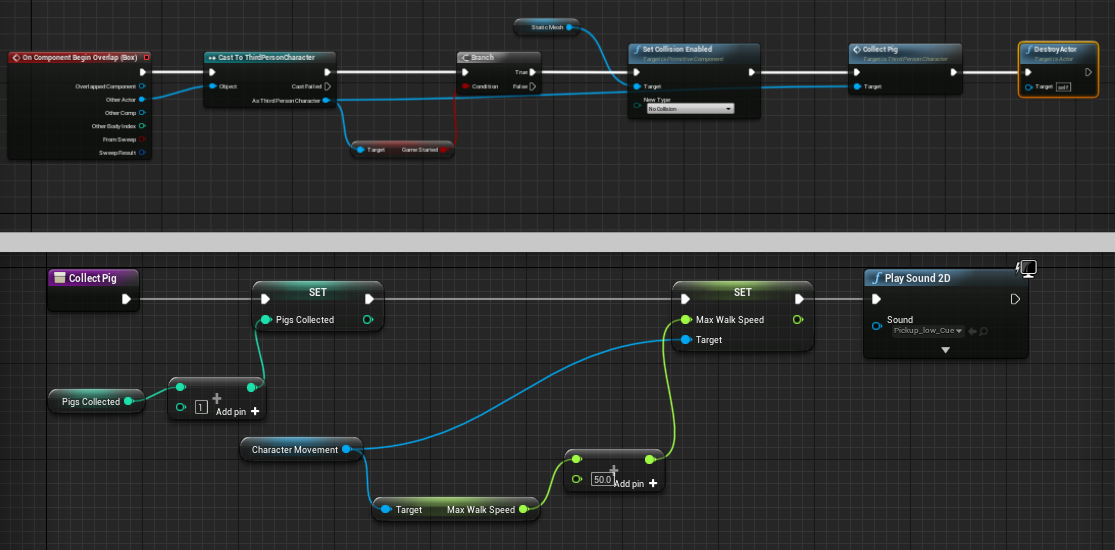
The next, and last, SFX that I added in was footsteps for the players character. A simple footstep timed well within the animation for running adds so much to the immersion the player will feel and is one of the first things I program in when I’m doing sound effects. Each footstep, no matter how fast the player is moving, will play the sound effect and is so simple to do it is essentially mandatory for me. I opted to not include a SFX for jumping as it isn’t an encouraged feature, more it was added to let the player feel more free in their controls/movement.

**4: Unreal Engine**

My proficiency in UE4 made this a very effective, and very fun, engine to use for this project. No direct coding was required as the entirety of the gameplay was handled utilizing the blueprints provided which allowed an exceptionally fast creation time for the game. Though it may have technically taken several weeks to create, that was because I opted to improve/change the game as the lessons progressed so that I could take from the lessons and add them in. I often found that what I had already programmed in was utilized in lessons learned later in the course. This was a pleasant and exceptionally welcome surprise as I was able to do things that I was familiar with, while also having the time to experiment and do things in ways I hadn’t done before, while not feeling any sensations of being overwhelmed with lack of time.

**5: Improved efficiency**

The GDD requirements made it so that I ensured I learned how to do things right, so that they were done correctly to meet the rubrics. With that in mind, the methods I used improved the efficiency of my game so that I wasn’t continuously re-programming over and over the same basic functions. The primary examples in this game was that I built the functions for the players score to increase inside of the player BP, and whenever the player overlapped with the fuzzy, it would call the function from within the players BP instead of externally pulling and modifying variables. This also allowed me to have more fun with what happens when the player collects an item. The following image shows the current state of the Pig and function built into the character BP named Collect Pig. Instead of rewriting the code from the function into the pig and every other variant of the fuzzy, I just had to call the function and modify which variable was being changed. I also added a movement speed enhancement on each fuzzy collected as well as their respective sound cue when they are picked up. If all of that was in the pig BP it would have cluttered it and made it harder to understand, and potentially impacted performance (not on this scale, but larger projects).



In hindsight this wasn’t needed for this particular project, but getting more comfortable with it and working within functions on the character blueprint has given me a stronger skillset when it comes to other, larger, projects, and has been an invaluable lesson from this project.

**What went wrong:**

**1: User Error**

The first thing I experienced difficulty with was my landscaping tools. I was suffering a problem where it seemed like only a single pixel was changing when I would attempt to sculpt my landscape, and I couldn’t find an answer anywhere through Unreal/Epic or any of my Google searches. I felt like I was at a loss and I was just about to submit a ticket to Unreal when I realized my error. I had my landscape gizmo size set to 1, which I thought was max (scale of 0-1), but in reality it was at it’s minimum, maximum size being over 8,000.

Luckily, this oversight was found early and was resolved with plenty of time to properly model my landscape, but it had potential to be devastating to my level design process and could have cost me not only time, but cost me the successes I saw in the creation of the levels.

**2: Bugs**

Early on we learned about destructible meshes, which I was excited for because it is something I enjoy working with in my projects. The physics within UE4 are great and provide a sense of realism when an item is destroyed and crumbles to the ground. However, some meshes caused the engine to bug and not correctly “explode” / shatter an object. Furthermore, once this bug occurred, all future meshes attempted to be shattered would revert to the first defective mesh, regardless if the mesh was in that project or not. Originally this occurred when I was working on the Space\_Race\_Prototype, but it caused issues with Fuzzy Farmer because the mesh that bugged in Space Race was causing all meshes in Fuzzy Farmer to exhibit the same mesh and error.

The problem was resolved with the launch of UE4 Version 4.24.2, but had already caused me to deviate from using destructible meshes (which aren’t a requisite of this particular game per GDD), but it was something that I was going to have some fun with originally. I feel like the delayed functionality of the destructible meshes prevented me from realizing an additional fun aspect in the game, but ultimately didn’t cause me any problems with the design.