Jennifer Guajardo

ENGL 3500

Prof. Pasanek & Bennett

5/6/2024

**Game Title: “Walking in the Dark”**

**Final Project Reflection: Designing a Game Without a Fail State**

**Introduction**

Failure in traditional video games cannot be permanent. Otherwise, the medium would be flooded with dysfunctional products. You could perish as many times in *Super Mario Bros.* or *Elden Ring* as much as you want, but your character would always come back. There’s almost always some way to reset the core game, even if consequences can carry over between playthroughs. On the opposite end, Mario only needs to defeat Bowser once to be considered victorious, regardless of how many wins the latter will have under his belt. Therefore, one could argue that the player’s victory is statistically inevitable so long as they keep playing. This got me thinking of games that allow room for self-imposed challenges. For example, speed running or no-damage runs, which are both possible in the games I mentioned, are only relevant because players can take the traditional mode of victory for granted. While it’s tempting to make an impossible game to expose this imbalance, I feel that it would be more enjoyable, both as a developer and player, to explore the multiple degrees of success that this phenomenon can create. “Walking in the Dark” is my attempt to create a game that accelerates the player’s journey to mastery, and I will go over both my designs and thought processes that led to the final build.

**Theme: Self-Driven Self-Improvement**

“Walking in the Dark” is a first-person platformer without a fail state. That said, the rules of the game must be made clear. The player is judged by how fast they can complete a course that’s shrouded in darkness, and each level is designed to be finished in under a minute after they’ve honed their skills. The only thing at the player’s disposal is their ability to walk and jump, their flashlight, and their capacity to memorize terrain. The flashlight will increase the amount of time passed, so one second will be judged as two seconds, a minute will be judged as two minutes, and so on. At the end of every course, one cube will appear for every in-game “minute” that has passed, up to four. If no cubes appear in the room at the end of the course, that means the player finished the level in under a “minute” with few flashlight inputs. With or without cubes, reaching the end still counts as a win, but having no cubes is this game’s equivalent to a “perfect score.” Through this ruleset, it’s impossible to truly lose. Through negation, a player will always win, but the form of their victory will change depending on their reflexes, quick decision-making, and terrain memorization.

I’m a fan of self-driven “striving play,” even if I enjoy games where every aspect of failure is absurdly punishing. For me, there is nothing more enjoyable than repeatedly fighting a hard boss for hours, even if they don’t go down. With the invention of health bars, timers, and other symbols of progress, I can see visible manifestations of my improvements. “Walking in the Dark” integrates both the cubes and timer as tangible proof of the player’s skill, letting them keep track of their performance over multiple sessions. The short play time allows for the quick, artful expression of the player's mechanical knowledge without requiring a massive time commitment. Any action from the player is unprompted until the last course, where they’re encouraged to improve their time. The player is given the chance to improve at their own pace, creating an empowering personal experience.

**Mechanics & Coding**

The core mechanics were simultaneously the most integral and difficult things to complete for the final build. I sought to expand on the functionality of week six’s homework assignment, trying to give the first-person experience its distinct platforming flavor. While being able to do a third-person perspective in a 3D environment became part of my skillset later, I started only knowing how to intuit a game from a first-person perspective. Essentially, if the player’s node could rotate, that would mean that the camera, by being a child node, would rotate with them. Of course, I must mention that I sought tutorials to help make the process work (See Appendix). However, I did make heavy adjustments to the physics and camera to both debug and improve movement.

The flashlight and timer gave me some trouble. Previously, I dubbed the game as a “true walking simulator" due to the heavy attention I gave to navigation and possible interactions. The original plan for one large level had to be scrapped due to my inability to get a handle on Godot’s larger lighting systems. Until experimentation, I didn’t realize that node visibility, as it appeared in the rendered scenes, could be altered in GDScript. Thanks to that, I made a simple conditional that would read whenever the flashlight was in its “visible” state, enabling that double-incrementation of time I mentioned before. As a hobby, I used to work a lot with RPGMaker MV, a JavaScript-based game engine. While I still respect the system and its coding language, I cannot express how relieved I was when I discovered that variables in GDScript can update every frame (60 per second) without the performance issues seen in the other engine. If the timer couldn’t judge the player at the end of every level, then there’d be no feedback on how the player performed. During testing, the game couldn’t be full screen due to the console being the only way to see how much time passed, leading to the awkward, partial window view seen in the final build.

The input mapping was based on the principles of the “CharacterBody2D” node taught in class. Aside from those aspects of the control scheme, I didn’t stray too far from the traditional layout of WASD to move and the spacebar being used to jump. By doing this, I ensured that players wouldn’t be alienated. People could easily just pick up the game and start playing it like any other first-person shooter. I assigned the flashlight functionality to the left mouse button, figuring that, with the said button being so ubiquitous in computer interactions, one might even turn it on by accident. For an added layer of functionality and accessibility, a player can walk forward by right-clicking, too. This makes the game more friendly for those trying to optimize, as anyone going forward can just focus their left hand on strafing and jumping. In addition, I mapped “R” to put the player back to the start of the game just in case they got stuck. The player must hold it down for ten seconds before it’s activated.

Most of the code is dedicated to the script attached to the player’s node. Not only was this necessary for movement, but it helped me repurpose the same functions repeatedly. The player is identical in all scenes, and the rings at the end of each level, using “Area3D,” read how much the timer increments, enabling the player to teleport to the different cube-having areas. I was honestly surprised that Godot didn’t make any sort of fuss because I kept having multiple rings, from different scenes, link to the same functions in the main script. Overall, its length was finished at approximately 200 lines long (counting spaces and comments). I am skeptical about whether this would be good practice in an actual development environment, considering the number of dependencies, but this implementation allowed for the final build to be cohesive with its programming.

**Level Design**

My idea for Course One’s layout translated well into the final game. It’s a simple S-shaped path with three necessary jumps. The first two are minor speed bumps, and the last one is a pit that requires the player to backtrack to the beginning. I constructed it like this because I wanted the player to understand their limitations from the get-go. The entire environment looks homogenous without the flashlight, giving the illusion of a straight path. If the player were to blindly go forward, they’d be treated to an almost immediate fall, emphasizing the importance of keeping track of your elevation. At the same time, the straightforward layout of the simple geometry communicates to the player that they don’t have to rely on the flashlight 100% of the time.

With the second course, I decided to play with the players’ expectations, literally dropping them into a pointless maze where the solution was right behind them. Had they not learned beforehand how geometry blends in appearance without the flashlight, they would know at this moment. With the goal ring obscured, the player must learn how to orient themselves within the 3D space. As the level goes on, the player will encounter semi-transparent “glass” walls (I adjusted the meshes’ transparency setting). They can notice this by seeing how light interacts with their surfaces differently. Engaging with this material is necessary to complete the level. Both the slope and the fake-out paths use it, whether for climbing or acting as walls. The addition of the rotating walls, which use the same script as the speedbumps, also takes advantage of the transparency effect, shoving any inattentive players back into the maze while never obscuring the goal ring to the point of disorientation.

Course Three is an extension of the glass effect from Course Two. The rooms are built with some of the meshes not aligning with their respective static bodies. I tried to pull off an effect where the player appears trapped in a room, but there’s a mirror that reveals where one could move. Granted, I couldn’t get this working for a second-player model, but I was at least able to give an affordance to the player, showing a simulated reflection of where important geometry is passable. My personal favorite, although surprisingly difficult in its execution, was the secret chute right after a speed bump in the third room. I had to manipulate way too many different plain meshes to get that working. The climax of the level is the goal ring’s reflection at the end. The “hitbox” of the ring is located under a hard-to-notice roof. Players need to investigate as far into the mirror as possible to figure this out. I even expanded the hitbox of the invisible goal ring just so the player would spend less time searching for it.

Course Four started with the question: What would happen if I removed the camera as a child of the player’s character body? While the execution was still clunky (the controls inherently made it disorienting to play test), I was able to create a convincing third-person experience. The level design featured heavy use of perspective, cylindrical objects, and transparent blocks to make it easier to judge where the light lands, letting the player understand their jump direction at any given time. Like any good gimmick level, I didn’t want to make this one too long to not overstay its welcome.

Course Five’s layout was a result of figuring out a series of bugs from the rotation script and my implementation of it. While I got it working for the bumpers, there were significant times when I couldn't get the geometry to rotate without it warping. I admit that I’m still not certain as to how I fixed it, but I believe it was due to outsourcing the action of rotation to parent nodes. This experience helped me improve the design of the other courses, particularly the second. There are three instances where I intended for the player to jump on rotating platforms. The first is a low-stakes example where it’s not required, but the player can observe that it's possible to stand on. Because of this, the player will know they can be pushed in all directions by geometry. Thus, the second is an elevator-like platform that allows the player to take advantage of that knowledge. The third builds on the principles of the second. Being right before the goal, the lack of stable ground was used to raise the stakes, forcing the player to be decisive with the timing of their movement.

Course Six is the simplest one out of all of them, containing a message from me about the nature of the game's mechanics and progression. The construction of the sign was simple: Two plain meshes, and a .PNG file imposed on the one in front of the player. As stated before, I withheld any direct explanation until the end to encourage replay value. If the player was guided to this at the start, they’d try to grind on a single course repeatedly before moving on to the next. I like the idea of the player at least trying out the different courses before trying to perfect them. There’s joy in both finishing and mastering. However, trying to do both at once invites tedium. Going in with and without foreknowledge invariably warps the experience of the player; they discover layouts that they cannot forget.

Despite being the player’s first impression, the hub area was one of the last things to be worked on. Having the elements of previous stages available, I created one big room sampling all of them. The ring that leads the player to the first level requires them to understand how to move. Meanwhile, the second ring is closer, save for the transparent wall between them. The third ring is just narrowly in sight, encouraging the player to hug the wall to see which is passthrough, foreshadowing the mechanic of the level. The fourth and fifth rings both have aspects of geometry from their respective areas, and the sixth is physically at the highest point the player could feasibly reach. With the final areas being accessible only by stairs, the player is almost certain to approach them later. Thus, the player is almost certain to go through the levels in the order I intended.

**Concluding Thoughts:**

My top priority was to create a game where a player couldn’t lose, and I succeeded. In the process, I cultivated an experience meant to be replayed, guiding the player to perfection without being demanding of them. There are several things that I wish I could've included, such as secrets that reference other aspects of the class, but I feel keeping the visuals minimal helped keep the aesthetic experience clean. Ultimately, the narrative created by "Walking in the Dark" is both personal and self-driven, subtly celebrating all the player's wins and improvements, however incremental.

**Appendix:**

* Tutorials:
  + First-Person Perspective: <https://www.youtube.com/watch?v=xIKErMgJ1Yk>
  + Help With Scene Changes: <https://youtu.be/XHbrKdsZrxY?si=wYUMdFqHqOhZpikg&t=101>
  + Fullscreen/Windowed Mode Help: <https://youtu.be/tau1IpRBLds?si=IjpowjafyGb7ZQpz>
  + Sound Effect Implementation: <https://www.youtube.com/watch?v=h3_1dfPHXDg>
* SFX & Music:
  + UNIVERSFIELD made all music and sound effects: <https://pixabay.com/users/universfield-28281460/>
    - The file names in the project are unaltered from their source.
    - Ambience: <https://pixabay.com/sound-effects/pleasant-atmosphere-153275/>
    - Click Noise: <https://pixabay.com/sound-effects/click-button-140881/>
  + **NOTE:** Some are used, and some are unused. In the original prototype, alongside the flashlight’s clicking noise, there used to be an ominous song every time it was turned on. Along with that, there was a drone in the background, which made me briefly consider making “Walking in the Dark” a horror game. However, I couldn’t come up with a good threat to the player.
* Controls:
  + WASD to move.
  + Mouse to move camera.
  + Left-Click to use flashlight.
  + Right-Click to move forward.
  + Hold R to reset level.
  + Exit Window: Alt-Tab
* One of the early prototypes will also be published on GitHub for the sake of comparison. Link: <https://github.com/jenniferGuajardo/Test-Time>