DarkLight is a 3D horror game in which players explorer a dark and foreboding labyrinth while evading the sinister spirits that lurk within. The shining light that illuminates your surroundings is also a shroud that conceals looming phantoms.

My initial idea was to make a basic survival-horror shooter in the style of some of my favorite game series, such as Silent Hill and Resident Evil. I would simply place the player in a dimly-lit environment filled with beasts and/or demons (but not zombies of course). Then I'd give them a flashlight and a gun. However, I knew that that premise alone was not novel enough for a rapid prototype, so I thought of an interesting twist that lights would actually obscure some of the enemies instead of revealing them. This idea stuck with me because it spawned so many implications; I could create an interesting interplay between light and dark, making the flashlight/darkness create interesting decisions for the player rather than simply being a restriction on their vision.

I wanted players to be in a constant state of uncertainty. In the light, they would wonder if any hidden enemies were near, and in the dark they would worry about getting lost, missing items, or running into enemies that interact with light in the normal fashion. What's unknown is scary, and I intended have a consistent number of "unknown factors" over the course of a playthrough.

I intended for my gold spike to prove the viability of the core mechanic: the enemies that are hidden by light (which I referred to as "shades"), and the task of evading them while trying to navigate a complex level. I first attempted to implement the shades' unique lighting through a custom shader. This turned out to be a pretty big waste of time. It was unreasonable to untangle all of Unity's lighting system to interface it with the transparency of the shades' material. I decided instead to use an approximation of lighting through raycating from each light to three sample points on each shade. This created a very convincing effect. I did not want to design the final level before the mechanics of the game were somewhat finalized, so for the spike I laid out a grid of rooms and hallways blocked off with doors, and wrote a script to randomly open some of the doors when the game started. This created a level that was complex enough for playtesting. In terms of goals, I simply put a glowing blue orb for the player to find and bring to a glowing green orb. This was sufficient for gameplay, and strange enough to fit the intended tone of the game. The spike was pretty rough, but it did show that the game had potential to be both fun and scary. It also showed that the interaction with shades was engaging enough that I could leave out other enemies and combat for the purpose of the prototype.

The behavior of the shades was the main focus of development, and therefore went through many iterations. I knew I wanted these enemies to feel dynamic, so I had them patrol randomly around the map before they got within a range of the player and started following him. I also wanted to give the player opportunity to react to nearby enemies they have not noticed, so I designed a two-tier health system: when shades were in attack range, they would add to the player's "fear" value, and when this value reached a certain threshold he would lose health. Fear would decay over time, and was displayed to the player with a vignette effect. During testing, I noticed players could easily avoid shades by simply running away when the effect appeared. The health system was confusing as well. I also realized that players would never see enemies before they started tracking the player, so the patrolling was pointless. From these results I simplified the health system (hit enemy = take damage) and created an enemy manager that selectively spawns in enemies to follow the player. Further testing showed that players would often bump into enemies without ever seeing them, and this felt very unfair. I therefore added a screen grain and audio static effect that scales with how close the nearest enemy is. I also modified enemies to only start following the player once they are seen by the player.

Most of my feedback came from the in-class testing session. The main result was that players were initially confused by the game, and many gave up before fully figuring out the mechanics. This stemmed from a lack of guidance in terms of the player’s goals, as well as problems with the enemy mechanics, as mentioned earlier. A few players also expressed disappointment that, as a horror game, it had no sound yet. From this feedback, and some extra playtesting from friends, I made some changes. Changes I made to the shades have already been mentioned. I also added more guidance through an intro sequence meant to illustrate the shades’ interaction with light / the player’s view, as well as some text in the green orb room that tells the player they are searching for something (in other words, they aren’t *supposed* to know which ways to go). The change from a randomized and repetitive maze to a hand-made level also made navigation a lot easier, as the player could somewhat recognize places they had been to before. Finally, the addition of sounds effects transformed the atmosphere of the game, making it so much more immersive and fun to play, even though most of the sound effects do not actually provide gameplay information.

I’m happy to say that my prototype achieved all the thematic and gameplay goals I initially set for it. The tone of the game, expressed through the art, sound, and gameplay is very close to my initial vision. The experience of the game captures the potential of the design ideas about as well as any rapid prototype could have. I made some mistakes along the way, such as wasting a couple hours attempting to code shaders, omitting sound from the spike, and initially overcomplicating some systems, but I ironed out these problems with testing and iteration. In the future, I will use custom shaders only for graphical effects (not anything gameplay-centric), I will respect the importance of sound, and I will keep my initial implementations of systems relatively simple, only possibly adding complexity after testing.