Intertwined

Development Info:

Name: Mauricio Mergal
Role: Programmer
Tools: Unreal 5, Github
Duration: 7 weeks
Team Size: 6

Game Info:

Genre: Puzzle, Escape Room
Platform: Windows
Status: Beta

About:

You are a doll, captured and about to be sacrificed. However, by the grace of a goddess whose name you do not know you have been granted the power to control time. Escape your prison and regain your freedom!

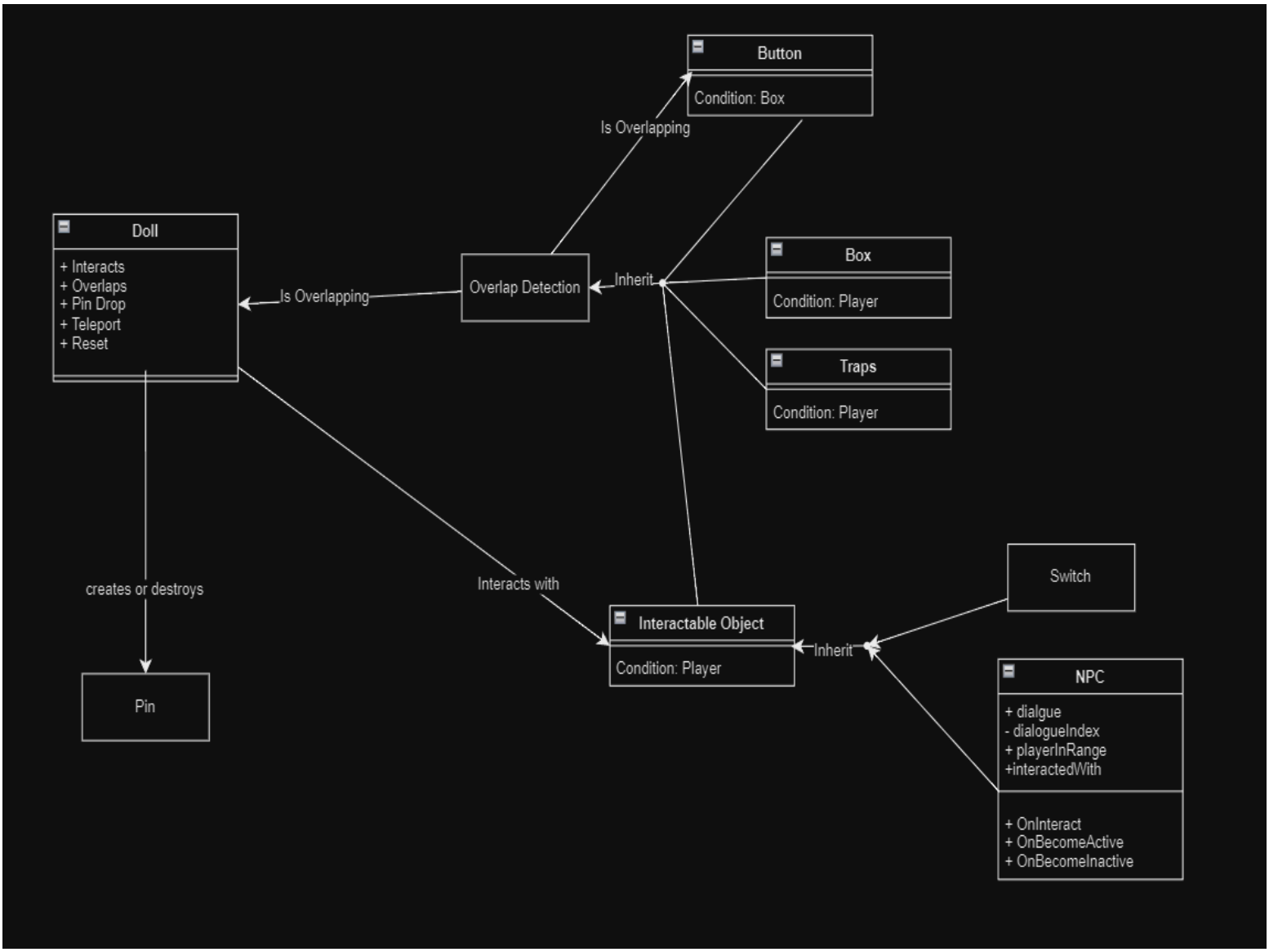
In this third-person puzzle-solving game you will be able to traverse rooms, interact with the environment, and teleport up to three locations established by you.

Contributions:

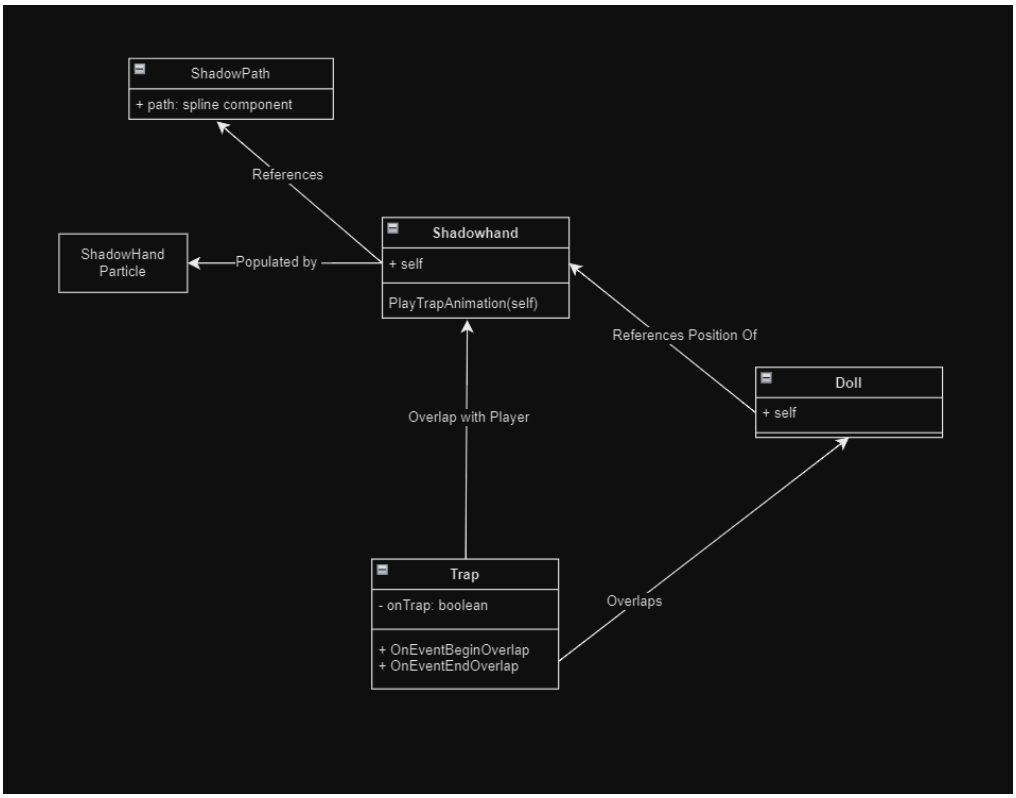
As part of this project, I prioritized a support oriented role to proactively fix bugs as they appear and enhance existing features, so the remaining members of the team could focus on their specialized tasks.

- Implemented the animations and textures for our player character and mapping them to inputs.
- Implemented our UI into our game world with full functionality.
- Implemented a resettable Trap object to add a layer of complexity to our puzzles
- Implemented NPC dialogue intractability with its respective UI.
- Worked on the overall improvement of mechanics and features based on the feedback gathered from players.
- Enhanced some of our current features by using Unreal's Niagara particle system to add visual appeal on top of the amazing models our team made. A total of 3 particle systems were made.
    - Two short animations were created using these systems.

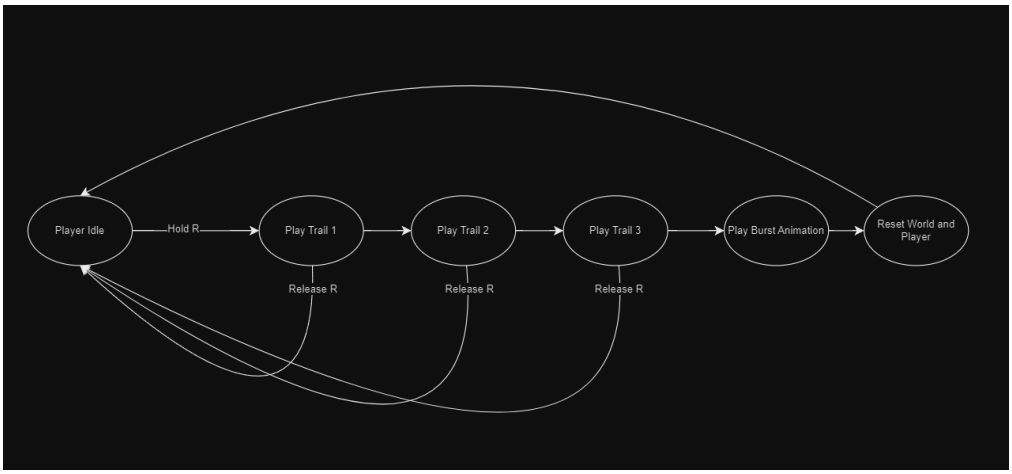
Code Architecture:



Overall Code Architecture



Detailed Look At Trap Implementation



Flowchart of Reset Animation

Version Control:

We used Github as our version control software.The original plan was to break down our individual tasks into their own independent branches and follow standard version control practice of pushing and pulling often. However, as none of us had ever used this software with Unreal before we ran into some issues before we understood how the two interacted.

- Blueprints and Github are scarcely compatible with one another. If two people work on a Blueprint then Github forces you to choose one version of the BP.
- Edits to the game world cause the same issues.
- Importing our custom materials and textures from branches caused merge conflicts that could not be resolved by Github.

Our solutions to these problems were to improve communication with what we were editing. Which also improved the tracking of each person's progress. With our new standards of communication built we were able to not only work from our branches and merge while avoiding overwritten code, but also whenever it was required (importing custom materials) working directly in our main branch.

Challenges:

- The learning curve for Unreal 5 was a steep one. This project was my first experience using this software along with their Blueprints system. The terminology, limitations and functionalities of this program were unknown to me and not knowing how they worked limited me in developing at the fullest from the beginning.
- While understanding how Github interacted with Unreal's systems, we had to completely change our version control standards. This made us create a lot of messy commits and forced many of us to rewrite code. For a period, we were unable to even maintain a standardization for our version control, which also added to our development time and tracking the progress of our tasks.
- All this resulted in the delayed development of our project, which affected us up until the final days of the project.

What I Learned:

- Unreal 5 documentation is extremely useful for efficiently developing in the application. However, sometimes their API's are out of date. That being said the community is very active so finding resources on what you are trying to do is easily accessible.
- Version control is everything. Understanding how the tools we are using interact with one another should be a priority at the beginning of every project. As it would have saved us a lot of wasted developer resources.
- Implementing the 80/20 rule has worked best for me particularly in this project due the delays we experienced. Using this rule we were able to create our features quickly, and once done with our respective tasks iterate through them to improve them to their

finalized version. Each iteration helped us guarantee that we had a deliverable product while also giving us the flexibility to evaluate which feature to prioritize moving forwards. Additionally, choosing to make components to make scalable and modular helped us develop alongside one another. This additional care allowed us to easily merge all our features together.

- Playtest as often as you can. Playtesters will find more bugs than you knew were possible to be in your code. Also, they will show you how a majority of people interact with the scene presented to them.