“Before Dark” Progress Report

Published by: React Gaming

4/3/2020

**Preface**

After coming up with a main concept, a game design doc, and a few assets, we pitched our concept to Dr. Ye in front of the class. We shared our team composition, our vision for the game, and an initial look at our company website. Our vision was met with praise and optimism from our dear evaluator. With bolstered enthusiasm, React Gaming set out to fabricate our game, “Before Dark.”

**Updated Team Structure**

Our first meeting after the presentation involved reevaluating what everyone on the team would be doing moving forth. Unfortunately, Jordan Meese had dropped the class, so we had to divide up the work he was supposed to do among us 5. The updated responsibilities are as follows:

* Michael Gee: group leader; manages the group; makes sure tasks and goals are completed on time; schedules the group meetings; programs player and character movement; updates the OPPM
* Leopold Frilot: programmer and documenter; tracks everything the group does; manages the tech behind the group including GitHub and Discord; programs game logistics such as saving and scene management
* Bram Metz: environmental artist; designs the levels and general feel of the game as well as the entities that inhabit it such as enemies, loot, and structures
* Courtland Crouchet: programmer and web designer; manages everything about the website; designs most of the mechanics and second-to-second gameplay that the player will be experiencing
* Angel Martinez: character artist; creates character models and animations to go with them; implements these into the game; went MIA

**Software architecture of the game**

At present, the current state of the game only has one level to explore. This level is intended to be a forest area that exists behind the playground area. Once the player starts the game, they will be able to move, jump, and attack freely. Currently the only goal is to get to the old man before the timer runs out.

A screenshot of a computer

Description automatically generated

**Implemented Features In-Depth**

Test Pablo/PlayerController(script)

One of the first features that we implemented was a temporary animated 3D model, “Test Pablo,” with a player movement script which allows the player to move the character via the keyboard’s WASD keys. This was implemented with the help of online videos based on 3D movement. It begins with a simple capsule object which the programmer can use either the rigid body or character controller for movement, in this case the character controller was used for more realistic movement. The script was then set up to move based on basic WASD inputs. A jump movement event with the press of space bar was also added to the script. Public variables in this script include movement speed, jump force, gravity scale, a controller for the 3D capsule to bind to the object, vector3 movement direction, rotation, an animator variable, and a transform pivot. Each of these public variables plays very important role in the movement of the player. For starters the moveSpeed variable controls the movement speed from the inspector allowing the programmer to control how fast they wish the model to move. The jump force is how high the player can jump into the air while the gravity scale controls how fast they come back to the ground. The controller is attached to the 3D model which is how the object accesses the script. The vector3 movement direction enables to player to move in different directions since this is a 3D game there must be movement in the x, y, and z directions. The animator is later used when the 3D model is imported into the game. This allows the programmer to place the model at the same location as the capsule and use it as a collider hiding it from the player and animating the model on top of it. The pivot object is utilized by a camera controller script.

CameraFollow(script)/CameraCollision(script)

The camera controller script allows the player to turn using the mouse and view the world around them. This is done my taking the main camera and adding several public variables dealing with offset values to control the distance of the camera, a target position to locate the main character, a pivot which allows the main camera to move with the player, rotation speed, and min and max view angle which determines how far up and down the camera is allowed to go. This script was then attached to the player movement script previously described so the two can work in conjunction with one another. A camera collision script was also implemented to go along with the movement and main camera. This handles events where a player may bump into a 3d object in the world around them or try to look through walls or the ground. The main purpose of this camera collision script is to prevent these previous mentioned activities from happening. If a player bumps into the wall the camera will gracefully move towards the main character zooming in closer, improving the visual quality of the game. At this time, a temporary 3D model is imported into the game and is utilizing these three scripts to make sure everything works properly, and no bugs arise in the final implementation of “Before Dark”. Once the main character model is made, and rigged, it will then be imported into the game and new animations, like the one already being used, will be assigned.

GameLoop(script)

GameLoop is the main driving force of the game along with the PlayerController script. It holds the essential variables of the current game state. It also initializes Pablo on level load to the correct location. It currently has three main functions: Saving/Loading, updating the time each frame, and updating the health. These are all public functions and are referenced/manipulated by other scripts.

PlayerData(script)/SaveManager(script)

PlayerData is a program that does not run at runtime. When SaveState is called from the GameLoop script, the PlayerData constructor is called from within the SaveManager script. This constructor saves various important data such as Pablo’s current health into the PlayerData variables. The variables are then sent to SaveManager to be converted into binary and saved to the hard drive in a file. When the LoadState function is called from within GameLoop, the binary file is opened and interpreted by SaveManager and the contents are copied into PlayerData for GameLoop to access and pick what it needs to override its own variables to reach the saved state as close as possible. Currently the variables saved are current level, current health, time left, current max health, and current position.

UIUpdater(script)

UIUpdater calls the GameLoop public variables health and time and changes the UI text accordingly.

EnemySpawner(script)

EnemySpawner will constantly check to see how many enemies are spawned in the level and make sure to spawn enemies until the number matches the predetermined max spawn. The spawn will be on a timer. So multiple don’t get spawned at once.

LightAdjuster(script)

LightAdjuster is a script that calls the current time from GameLoop and adjusts the rotation of the light source accordingly. There are two main variables, startingSunRotation and lightSections. startingSunRotation is a variable that controls the initial point of the sun in degrees. 90 represents 12:00 and 180 represents sundown. lightSections is a variable that controls how smooth the sun falling is. The higher the lightSections, the smoother the sun will drop.

SceneSwitch(script)

SceneSwitch is a crucial script that allows the game to move from level to level. It is completely dormant until another script calls one of its methods. These methods include LoadParticularScene, LoadNextLevel, GetCurrSceneIndex, LoadIntro, LoadSettingsScreen, LoadLoseScreen, and Quit. All of these are self-explanatory and simple but helps that they are all located in the same script.

GoblinController(script)

GoblinController is the script that handles all the goblin’s behavior. As our first enemy the goblin has many features that later enemies will have, although it is simpler and therefore a better task to start with. Currently the goblin has a health pool that is depleted when attacked by the player, and a simple aggressive AI is being worked on right now that can go on to be applied to other enemies.

Friendly Characters

The “Friendly Characters” GameObject houses the friendly characters from each level. This currently is only Test Pablo, Pablo, Homeless Man, and Older Brother.

* Test Pablo: Test Pablo is a temporary complete asset with animations that we are using to test scripts while Angel works on the actual Pablo asset.
* Pablo: Currently just a yellow capsule with a blue top hat until Angel creates the actual asset
* Older Brother: Currently just a light-green capsule until Angel creates the actual asset
* Homeless Man: Currently just a red capsule until Angel creates the actual asset. Currently touching it wins the game.

Enemies

The “Enemies” GameObject will house the enemies that get created by the EnemySpawner script. Currently we only have the Goblin created, as that is the enemy of the Forest level.

* Goblin: The goblin was imported from the Unity Assets store since we couldn’t create a model. It is rigged with animations and skeleton. We’re working on being able to get it moving without falling.

UI

The UI is currently very bare bones. It shows the mechanics instructions, health, and time left in the day. Health and Time are updated by the UIUpdater script.

SFX

There are two sound effects in use. One is an ambient forest sound. According to the recorder, it is a heavy birdy forest ambiance in Koliba, Slovakia in the Spring. The other is background music created by PandaTooth on YouTube. It is a medley of piano covers of “Zelda: Breath of the Wild” music. Both soundtracks are royalty-free.

Level Design

* Level 1: Bram’s input here

Website

Courtland’s input here

**Actual Member Contributions (as of 4/3/20)**

Michael Gee

* Arranged all group meetings
* Imported temporary 3D model to test movement and animation
* Initialized the progress report document
* Implemented the following scripts:
  + Camera follows the player
  + Camera detects collisions
  + Player movement controller
  + Animation manager
* Summary:

During each phase of the project, I made sure goals were met in a timely fashion, and team members were contributing to the overall success of the project. Utilizing the OPPM sheet to make sure we are on task. Unfortunately, we lost some group members, with one dropping the class, and another who went MIA for ~5 weeks. This resulted in a 6-man team turning into a 4-man team, which made the overall workload of the project much more strenuous that it should have been. Each team member had to pick up extra work with other classes during a pandemic. Although the game is not exactly how we described, as it would be in the original document, we still managed to pull though, and made a decent demo. Along with group management, I also implemented the player controller script, which allows the main player to move, camera controller script, and camera collision script, as well as helping with documentation and other areas of the game when a bug may arise or if anything else needed work.

Leopold Frilot

* Set up the forest scene, “level one” to act as a template scene with template UI, Light, Level structure, and other necessary components
* Created and managed the company GitHub and Discord server
* Implemented the following scripts:
  + A core game loop script
  + UI updater
  + Saving/loading
  + Sunlight adjuster based on in-game time
  + Scene switcher
* Summary:

For this project, I mainly focused on setting the group up for success. Discord and GitHub have proved invaluable for group projects in the past and they continue to prove themselves incredibly useful. After gathering everyone together in the discord, I constructed a comprehensive guide to setting up GitHub for easy collaboration. Unfortunately when using GitHub with Unity, a “.gitignore” file is essential, which led to some stressful nights of testing just to get that to work.

Once inside Unity, my goal was to use the 3 years of mistakes and experience with Unity to set the Unity environment up so that it stays organized and flexible. This included setting up an initial file structure that would house our assets. After initializing a scene to be our first level, I created empty GameObjects of the main elements that we would need in each level. All actual GameObjects would be children of these empty GameObjects. This allows for better organization and easier manipulation through scripts. I then made sure almost all assets that we incorporate in the level are prefabbed to allow for flexibility down the road. The scripts that I chose to implement were all logistical and background controllers/manipulators. They are outlined above.

Bram Metz

* Summary:

Courtland Crouchet

* Website design and combat mechanics
* Implemented the following scripts:
  + Goblin Controller which detects Pablo’s attacks
  + Pablo attack animations
  + Pablo attack colliders
  + Item search script
* Summary:

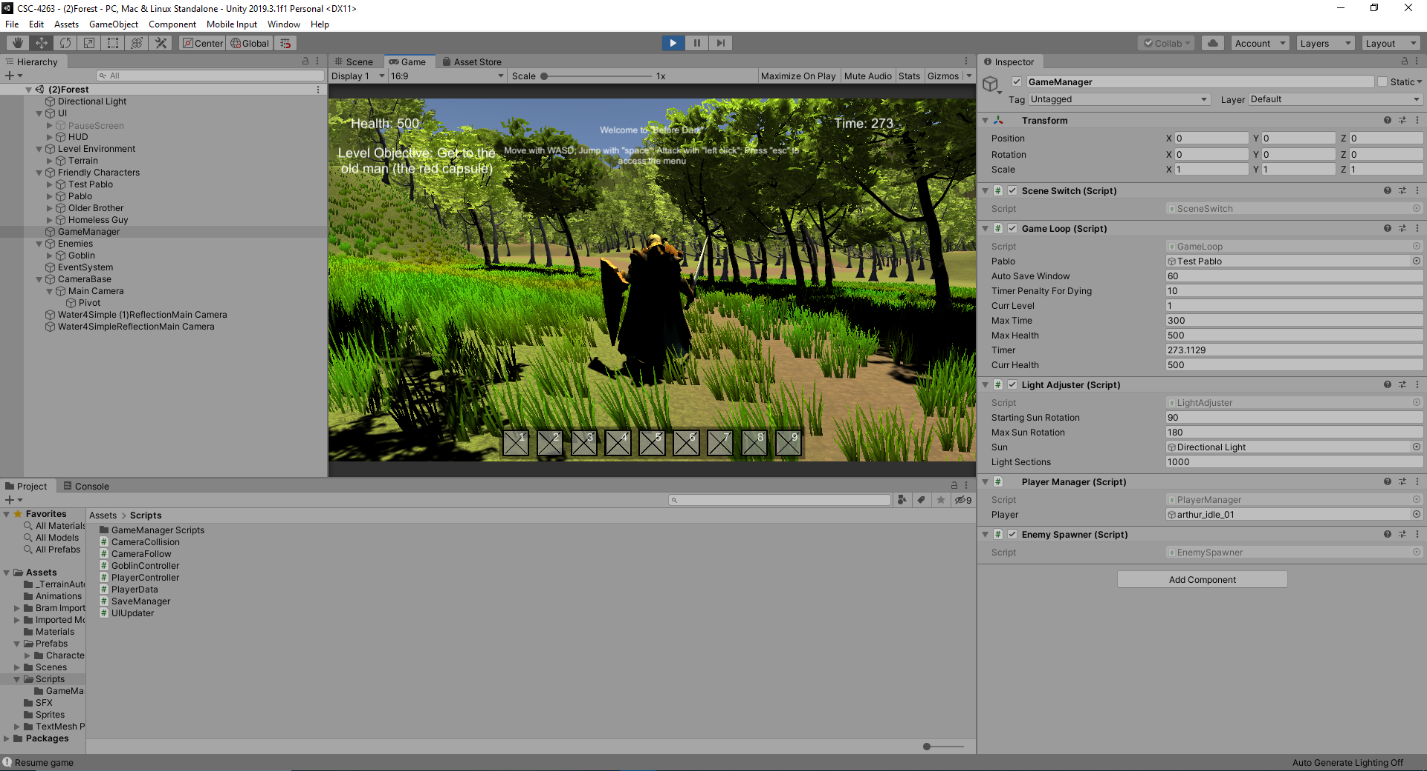
I made the website from scratch using Twitter’s Bootstrap library and hosted it using Google’s Firebase. I also implemented the starting combat mechanics for Pablo and our first enemy, a goblin.

Angel Martinez

* Nothing yet

**Future-plans**

Future-plans include expanding the game into more than three levels, adding new bosses, items, and quest that Pablo can go on. Some examples of more levels include a cave like dungeon, Pablo encounters in the woods, which he can enter with the appropriate item such as a flashlight. Another level could include a city / grocery store, which Pablo travels too after making it back home and his mother realizes a grocery was forgotten. Pablo then becomes lost in the store and must find his way out before his mom leaves him there and travels back home for dinner. The neighborhood, level 3, can also be expanded in size and Pablo can encounter his neighborhood friends where new side quest are located and he must help his friends complete their quest before finishing the final quest, making it back home before the streetlights turn on. His friend’s quest can include sub-side-quest which each reward the player with and item, Pablo can then combine these items into a powerful weapon of his imagination, allowing him to enter new areas of the game, and defeat bosses and enemies previously unbeatable. Some new bosses include a vampire in the cave level, Pablo’s teacher from school at the grocery market, and another secrete identity boss that Pablo must face once completing his friends side quest to reclaim their hang out.

Discussions of a hard mode will be implemented which players can beat on a shorter time crunch. When the player beats hard mode, they will unlock access to special developer weapons which will have different effects on enemies. An example of this would be a weapon that allows Pablo to one shot enemies to make replaying the game a fun experience for those who have already beat it. Other future-plans include a potential sequel to the game. Where Pablo goes on another adventure, except instead of recovering his mother’s groceries he now has a new task to complete, new areas to explore, and new imaginative monsters to defeat. The sequel can happen outside of Pablo’s home neighborhood, with many different possibilities for quest and items to unlock. This may include the city Pablo lives outside of, which he travels to by car or train. New mechanics can be added. Also, in the following sequels to “Before Dark” Pablo grows in age and realizes it wasn’t just his imagination he was facing but possess a unique power that ordinary humans do not have. He then uses this power to combat more dangerous foes who come to destroy his neighborhood, and areas he grew up in. This allows implementation of supernatural abilities Pablo can obtain such as flight, invisibility, spell casting, and other abilities humans do not possess.