Work Day 1: 11/1/23

Although I will not be able to sit down and code the game for a few days, I decided to think up a basic overview for the game, especially what is doable and what is not doable. At this stage, nothing is concrete, and there is a high probability that I will overscope and cut features when I realize that the implementation will take too much time.

Basic Idea: Some sort of vaguely RPG combat game where the player earns upgrades by playing. I was thinking of a survival game with a shop where the player fights an endless horde of enemies, but a (very) simple roguelite could work too.

Aesthetics:

- Sword-and-Sorcery Fantasy: The player is immersed in the role of a fantasy adventurer with an array of magical powers and cool weapons (assuming I can find good sprites).
    - Success: Variety of abilities and weapon upgrades, build variety, interest in experimentation
    - Failure: Use of only one strategy, abilities are too similar/useless/boring, players disinterested in available abilities/upgrades, melee attack spam
- Skillful Play: The player is invested in playing well and overcoming challenges.
    - Success: Progress is correlated with skill, focus on timing, focus on managing resources, decision making is important, player feels some sense of challenge
    - Failure: Progress and character strength tied primarily to RNG, gameplay too easy, gameplay too difficult, "mindless" gameplay, "unfair" enemy attacks

Core Gameplay:

- Typical combat-oriented RPG gameplay loop. The player kills enemies/monsters of some type, which drop gold and potentially (lesser) upgrades. The player can occasionally access shops, where they spend the gold in exchange for upgrades and better abilities. This supports the "sword-and-sorcery" aesthetic by giving access to cool weapons and magic as the player progresses, as well as setting up a "dungeon crawl" type of setting where the player is an adventurer seeking to hone their skills by killing monsters. This also supports the goal of an emphasis on skillful play, as the player is rewarded with more gold, progress, and a stronger character for their mechanical skills and build choice.

Notes:

- Player has two basic actions besides movement - casting a spell, and swinging a sword
- Access to a shop at various points in the game
- Player can get straight upgrades (more HP, mana, dmg) or new abilities (different spells, different swords)

- Game is probably a roguelite - predefined start and shop rooms, and then combat rooms randomly chosen from room presets with randomized enemies
- "Difficulty modifier" scales with progress from the start (number of rooms cleared) - higher difficulty means more enemies and harder enemies
    - Threshold values for when higher tiers of enemy begin to spawn
- Defensive moves - dash, block, parry?
- If I have time - player can be offered sidegrades (ex. drop melee dmg for increased spell dmg)
- No healing besides from room drops/shops
- I probably don't have time to make a boss enemy, so gameplay is endless until the player dies and success is measured by number of rooms cleared

Work Day 2: 11/5/23

I finally was able to start work on this project, as I had assignments for two other CS classes. The goal for the day is the implementation of basic functionality, as well as some planning for more complex functionalities.

Implemented:

- Player movement (directional, rotation)
- Player dashing
- Basic sword and spell functionality (swinging a sword, shooting projectiles)
- Abstract class for spell scripts

Planning Stats and Abilities:

- Stats:
    - Health (HP)
        - Health restores should be uncommon, while health upgrades are rare
    - Mana (MP)
        - Mana restores should be common, while mana upgrades are rare
    - Melee Damage Modifier/Multiplier
    - Spell Damage Modifier/Multiplier
- Abilities:
    - Need a superclass for abilities, which will be components or child objects of the Player
    - Player class view of abilities/spells is generalized - just call activate() on the component/object when a key is pressed
    - Give the player access to two spells at a time?
    - For spells, might have to delay destroying the projectile for AoE type spells...
    - Example Spells (probably not implementing *all* of these):
        - Magic Bolt - basic starting spell, shoots out one small projectile
        - Energy Blast - shoots out one larger, more damaging projectile

        - Triple Blast - shoots out three larger projectiles with small/medium damage
        - Fireball - shoots out a large projectile that does medium damage, but does DoT damage on enemies
        - Freezing Flare - shoots out a small projectile that does medium damage, but slows affected enemies
        - Lightning Strike - shoots out a fast small projectile that does heavy damage
        - Magic Beam - shoots out a continuous beam that does small amounts of damage
        - Force Shield - summons a shield in front of the player that absorbs or deflects incoming attacks
        - Force Surge - rushes the player a long distance forward and does small amounts of damage to enemies in the way

Work Day 3: 11/9/23

Well, it's been a while.

Implemented:

- The Enemy abstract class
- Functionality for a working Mage enemy, including the script and prefab
    - The Mage tracks the Player and periodically shoots spells at the Player
- Spell functionality on the Player and Enemy:
    - Taking damage on hit
    - Applying a damage over time effect
    - Applying a temporary slow or stun effect
- A rudimentary, placeholder UI that shows the Player's HP

Changes:

- Design Changes:
    - Lightning Strike should stun for a short period instead of being heavily damaging
- Code Changes:
    - Modified class inheritance hierarchy:
        - Player and Enemy now inherit from superclass Entity so they can share methods pertaining to spell effects
        - Genericized spells that involve shooting a projectile into the ProjectileSpell class
        - Changed the Projectile class, which determines the behavior of the actual projectile, into an abstract class
        - Genericized energy spell projectiles (spells that only do damage and have no special effects) into the EnergyProjectile class

Work Day 4: 11/10/23

Implemented:

- Added player stat functionality
  - Casting spells drains mana, and dashing drains stamina
  - Mana and stamina are displayed in the UI
  - The player regenerates stamina over time
  - Player now has a "gold" stat
- Added spell choice functionality for player
  - Added code that waits for player input of affected spell slot when acquiring/upgrading spells
    - Player chooses a slot to overwrite when acquiring a spell with all slots full
    - Player chooses a slot to upgrade when getting an upgrade where the player has multiple spells
  - Added UI text that instructs the player to choose a slot when appropriate
- Added code to the Player class that allows for alteration of stats and changing the player's sword/spells (helpful for powerups)
- Added some enemy functionality
  - Enemies die when they hit 0 HP
  - Added the Mercenary enemy and Warrior enemy script for melee enemies
  - Added a delayed sword swing for enemies so the player has warning before they swing
  - Enemies have a gold reward and give the player gold upon death
- Added some spells
  - Fireball, Frostflare
- Added pickups
  - Health and mana pickups restore an amount of HP/MP
- Added powerups and functionality for buying powerups
  - Powerups are activated when the player is in range (of a trigger) and presses F
    - The powerup object disappears after being picked up
    - Activation only occurs if the player has enough gold to buy the powerup
  - Player can get and spend gold
  - Ability type powerups give the player a spell or sword
  - Upgrade type powerups give the player a stat boost
- Added powerups for Fireball and Frostflare

Bugs:

- Spells created by ability powerups have a non-zero z-rotation for some reason.

Work Day 5: 11/11/23

Implemented:

- Fixed the aforementioned spell projectile rotation bug (mostly)

- Added the Energy Blast spell and associated powerup
- Added the Longsword
- Added the tier 2 enemies
    - Mage, Warrior
- Added text to powerups to show the gold cost
- Added rooms/scenes
    - Start room
    - 3 combat rooms
    - Shop room
- Added doors that enable the player to leave the current scene
    - Doors are locked by the number of enemies in the current room - all enemies need to be dead
- Added a MapManager that handles random(ish) generation of rooms
    - MapManager keeps track of current room, how many rooms have been cleared, how many enemies are in the room
    - Randomly chooses the next room (with some parameters)
        - Guaranteed shop for every 10 rooms
    - Loads the next room
    - Initializes the next room by placing the Player at a spawn point and spawning enemies/powerups
- Implemented random generation of enemies
    - Enemies spawn at random points
    - Added "SpawnZone" objects that provide legal bounds for enemy spawns
    - Type of enemy spawned depends on number of rooms cleared (partially implemented)
- Implemented random generation of powerups
    - Only happens in shops
    - Powerups spawn at predetermined locations marked by SpawnZone objects
- Enemies now have a chance to drop (health/mana) pickups
- Player can move between scenes and actually progress and play the game

Changes:

- Buffed the player's stamina regen rate
- Cut the non-projectile planned spells (Magic Beam, Force Shield, Force Surge)

Work Day 6: 11/12/23

Implemented:

- Added final round of powerups
    - Added some spells and appropriate powerups - Lightning Strike, Triple Blast

    - Added Greatsword and sword powerups
    - Added stat upgrades - Health, Stamina, Mana, Sword Damage, Spell Damage
- Added final round of enemies
    - Added tier 3 enemies - Archmage, Champion
- Fully implemented difficulty scaling for enemy and powerup spawns
- Fully implemented player death (can no longer have negative HP...)
- Added an end room/scene and configured room generation so the player reaches the end after 30 rooms
- Finally added solid bounds to rooms/scenes
- Finalized UI elements
    - Score is now displayed in the UI
    - Added text for Victory and Death screens
    - Added text for status notifications to entities (signals when an entity is stunned, burning, etc)
- Implemented audio
    - Added Audio script
    - Added audio assets
    - Added appropriate calls to Audio methods throughout code

Changes:

- Object UI moved from object World Canvas to main Canvas
- Powerup text now also displays the powerup's name
- UI stat display now shows the current and maximum values for that stat
    - (Ex. Health: 93 / 100)
- Game is no longer endless and has a stopping point
- Pickup drop probabilities changed such that mana pickups are more likely to spawn than health pickups

Potential Issues:

- Game too hard (or I'm just bad at my own game)
- Needs more balancing - enemy difficulty seems to scale faster than player power
- Shop may spawn too early when player doesn't have money, denying them powerups

Work Day 7: 11/13/23

Implemented:

- Finalized level design - added walls and appropriate spawn zones to combat rooms

Changes:

- Rotated the Triangle sprite used for Mages so that enemies face the right way when casting

- Rebalanced some things
    - Lowered HP of enemies to make them less tanky
    - Buffed gold rewards for low tier enemies, nerfed rewards for high tier enemies
    - Buffed/nerfed stats on spells and weapons

Postmortem

Initially, my goals were to create an engaging RPG/roguelike game themed around a spell-and-sorcery fantasy. My intention was to provide the player with a wide array of gear, abilities, and upgrades, such that the player can create a build as they play through the game. As well, I aimed for multiple playstyles and builds to be viable, encouraging experimentation and multiple playthroughs. I also wanted to reward skillful gameplay over button mashing, and I wanted the game to be challenging, but not crushing or impossible.

By the end, I still retained some of the above goals, but I felt as if I had to compromise on some of those goals in order to focus on the main goal of finishing on time and delivering a workable product. My goals were balancing the game (to some extent) to create an experience that isn't annoyingly unfair, and hammering out bugs to make sure the player doesn't get stuck. As well, I ended up with an additional goal of choosing audio that matched the magical aesthetic I wanted such that the player is immersed in their role of spellsword.

I accomplished my intended goals - to some extent. I did end up with a good amount of spells and swords, and I would say that there is some build variety and space for decision making; for example, the player might choose powerful but costly spells, or stick with lesser spells that are more economical to use. However, I ended up cutting some of my planned spells, and thus cut a playstyle that would have provided more variety - a melee bruiser mage that uses Force Shield and Force Surge to block attacks and close gaps. Therefore, a lot of builds became some variation of "kite and shoot enemies, then finish them in melee." As for my goal of rewarding skill, there is some element of skill involved in the game, especially with dashing, but the game can sometimes feel unfair, especially when RNG denies the player powerups. Balancing the game was difficult, especially with a time crunch.

Besides goals, one thing that went right about the project was that I genuinely enjoyed myself. I had fun playing the game myself, even if I realized that making a skill-based game was dumb because I had no skill. It was satisfying to see my ideas come to life over time as I added more features. However, one thing that went wrong was that the game and its code became somewhat complex, and thus bugs could become difficult to track. As well, making changes caused downstream effects that forced me to investigate and make appropriate changes in multiple places. I had multiple instances of a minor change breaking my prefabs or other elements of my game.

I wish I knew just how little time I actually had to work on the game versus the scale of what I wanted. Making certain mechanics work cost more time than I anticipated. As well, I wish I knew more about quaternions (particularly Slerp) before I started, because I spent too much time coding my own poorly-implemented version of Slerp to rotate a sword, when I could have just used a Quaternion method. I also wish I knew more about Unity UI elements and how to attach UI to objects. In general, there were a lot of Unity-specific things that I had to learn on the fly from the documentation that slowed me down, but the flipside of that is that I learned plenty of new Unity things. I learned how to use coroutines, which was immensely helpful, as well as how to manage, load, and populate scenes. This project expanded my knowledge of Unity classes and methods.