Game Proposal: Cleanse the Corruption

CPSC 427 – Video Game Programming

Team: Team 25

Munn Chai 56857261

Jason Xu 36612943

Kevin Guo 58081522

Kevin Shaw 25032863

Ziyu (Mark) Zhu 25611807

Story:

Briefly describe the overall game structure with a possible background story or motivation. Focus on the gameplay elements of the game over the background story.

Game Genres: Action, Rogue-like, Bullet Hell, 2D Top-down,

Story Description

The Mage Tower was once a beacon of hope and strength for humanity, a place where those who were blessed with mana honed their magic to protect the world. However, a strange corruption has since overtaken the tower, driving all but one mage into madness. The corrupted mages now attack anything that moves in a frenzied rage, spreading chaos and destruction. As the last sane mage, the protagonist must re-navigate the tower while fending off their former allies to uncover the source of corruption before it consumes the world.

The gameplay will feature fast-paced spellcasting, with the protagonist shooting spells at enemy mages while trying to dodge a barrage of different attacks. Players will be able to collect different relics which can upgrade their own spells to create different combinations of buffs and effects. Most enemy mages will shoot out ranged bullets, which the player must dodge or choose to block with certain defensive spells. Some elite enemies will also have melee movesets and cause the players to adapt to different combat styles and adjust their strategies. The final boss will be the Grand Magus, a strengthened amalgamation of different enemy mages the player faces throughout the game.

Overall Gameplay Structure

The player will start at the bottom of the Mage Tower, and explore each level to whatever degree they choose to. In each level, they will face one (or two?)

- Players begin at the Main Menu and splash screen, where they can start the game by clicking on the “play” button with their mouse.

- A Floor is generated with various rooms in a gridlike structure (see Scenes for example map generation) and populated with enemies, then the player is spawned onto it.

- The player has to navigate through rooms to find and then fight the boss before proceeding to the next Floor. While doing this, the player can collect spells and relics that strengthen the player’s abilities.
- The following rooms will be generated on floors:
    - **Spawn room:** this is the room which the player spawns in as they enter a floor. There are no enemies or treasure chests in this room. Only one spawn room can be generated per floor.
    - **Basic combat rooms**: these are filled with enemies, as well as a treasure chest which contains rewards (spells, relics). After each enemy is defeated, there is a random chance of a gold drop, the stronger the enemy, the better the reward and the better the probability of reward drop. 4-6 basic combat rooms can be generated per floor.
    - **Shop rooms:** these contain various items that you can purchase using gold. Items that you can purchase include:
        - Health potions that recover your health
        - Spells
            - Fire attribute
                - Shotgun cone shaped
                - Shorter range but higher damage
            - Ice attribute
                - Long range slow projectiles
                - Slows enemies
            - Thunder attribute
                - Piercing bullets – projectile pierce +2
                - Longer range
                - Lower rate of fire but higher damage
                - Very high projectile speed
        - Relics (see Gameplay Elements below for details on Relic system)
            - Relic of Strength
                - Damaging Ability: +1 damage
                - Movement Ability: +0.25 distance travelled
            - Relic of Time
                - Damaging Ability: spell cooldown -0.25 seconds (spells can have a minimum cooldown of 0.1 seconds)
                - Movement Ability: spell cooldown -0.25 seconds (spells can have a minimum cooldown of 1.0 seconds)
            - Relic of Quickness

            - Damaging Ability: Projectile speed +1 distance/second
            - Movement Ability:
        - Relic of Numbers
            - Damaging Ability: Projectile count +1
            - Movement Ability: Cast Count +1
        - Relic of hunting
            - Damaging Ability: Life Steal +10%
            - Movement Ability:
    - Random player upgrades:
        - Max health
        - Damage
        - Passive healing
        - Lifesteal
        - Movement speed
        - speed
- **Treasure rooms**: which just contain a treasure chest and no enemies. Only 2 treasure rooms can be generated per floor.
- **Boss rooms:** these have a single large boss enemy, which will have more health and stronger attacks than basic enemies. Killing this boss will reveal a ladder/staircase which leads up to the next floor. Only one boss room can be generated per floor.

- Finally, after clearing **three floors**, the player will enter a final boss floor, which will consist of a single room where the player will fight the final boss.

- After defeating the final boss, the player will reach the top of the tower, where they win the game, and credits will roll.

Gameplay Elements

Players will have 2 spell "slots", a spot for a damaging ability and a spot for a movement/utility ability. They will start with a basic spell for both slots, but will be able to swap new spells with their current ability. They can find new spells by opening treasure chests or purchasing them at shops. Players will also be able to find Relics to attach to spell slots, which will permanently increase the power of the spell currently inside said spell slot. When spells are swapped out, the buffs gained from their current Relics stay with the player and carry onto the new spell type.

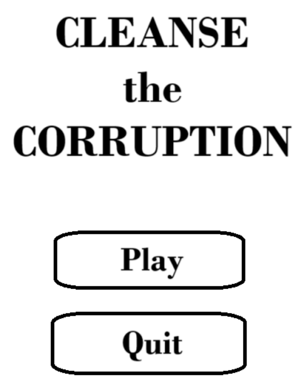
Basic enemies will also use the same spell slot and relic system for their attacks, but they will be assigned a certain spell and a random number of relics on that spell when they spawn, which will not change for the duration of their life.

Stronger enemies, such as bosses, will have more spell slots and more relics for those spell slots. However, if we find we have extra time to design, or decide it is necessary for more challenging gameplay, we may design unique attacks for bosses such as melee attacks, or large area of effect attacks.

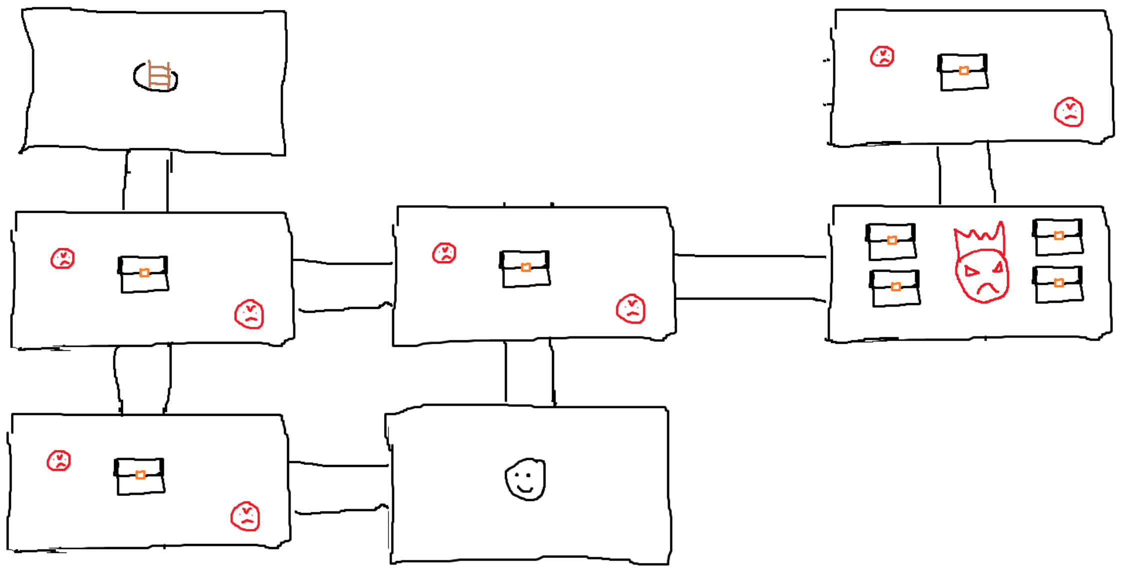
Scenes:

Produce basic, yet descriptive, sketches of the major game states (screens or scenes). These should be consistent with the game design elements, and help you assess the amount of work to be done. These should clearly show how players will interact with the game and what the outcomes of their interactions will be. For example, jumping onto platforms, shooting projectiles, enemy pathfinding or 'seeing' the player. This section is meant to demonstrate how the game will play and feeds into the technical and advanced technical element sections below. If taking inspiration from other games, you can include annotated screenshots that capture the game play elements you are planning to copy.

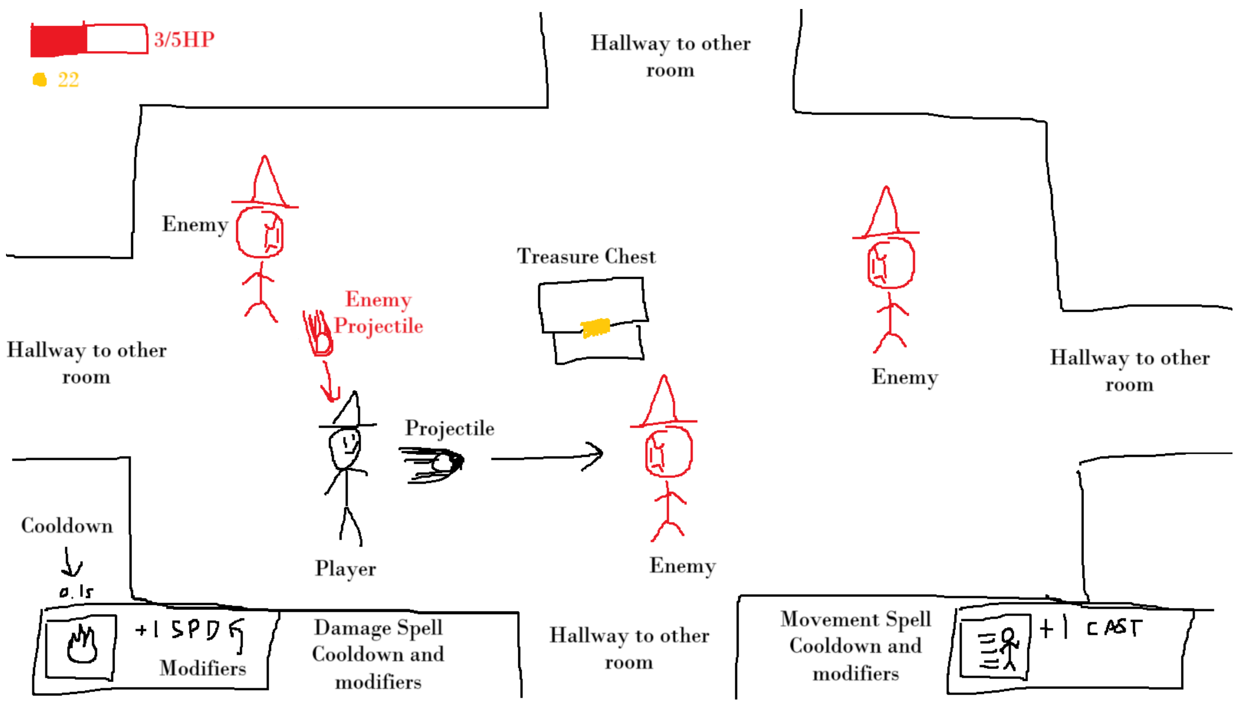
Main Menu

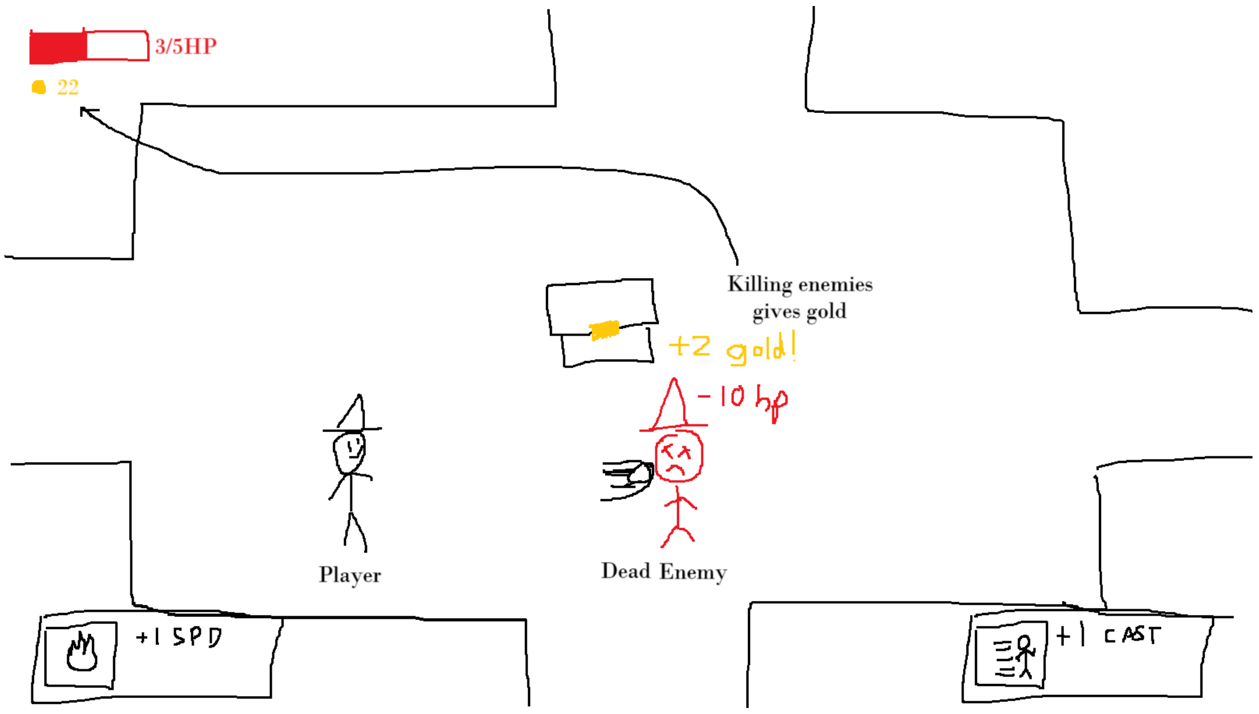


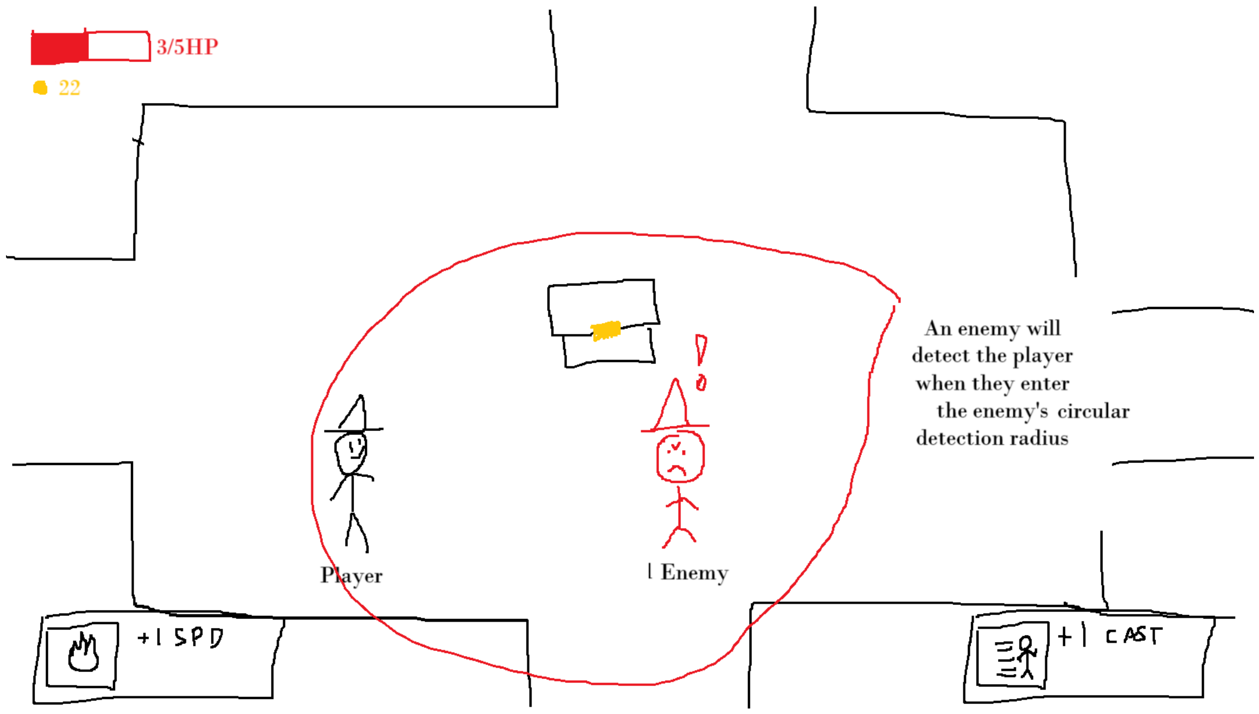
Example Room Generation



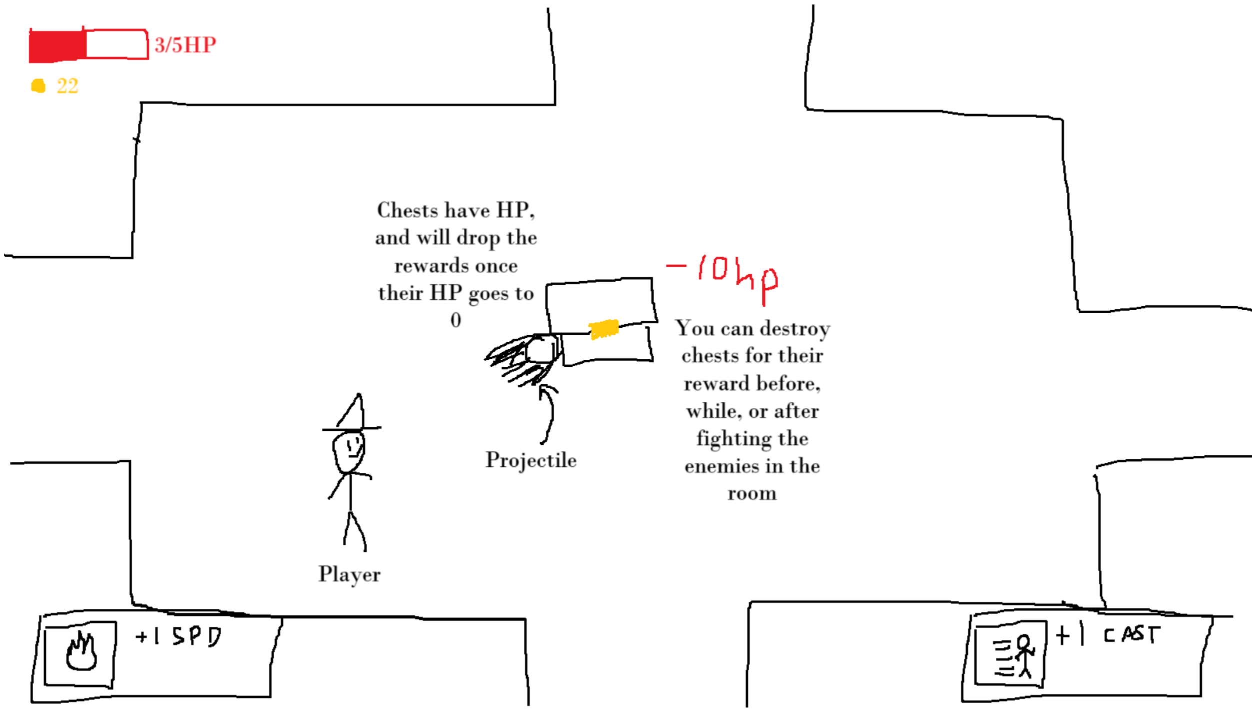
Combat Room


3/5HP
22
Hallway to other room
Enemy
Treasure Chest
Enemy Projectile
Enemy
Hallway to other room
Hallway to other room
Projectile
Player
Enemy
Cooldown
0.1s
+1 SPD
Modifiers
Damage Spell Cooldown and modifiers
Hallway to other room
Movement Spell Cooldown and modifiers
+1 CAST


3/5HP
22
Killing enemies gives gold
+2 gold!
-10 hp
Player
Dead Enemy
+1 SPD
+1 CAST



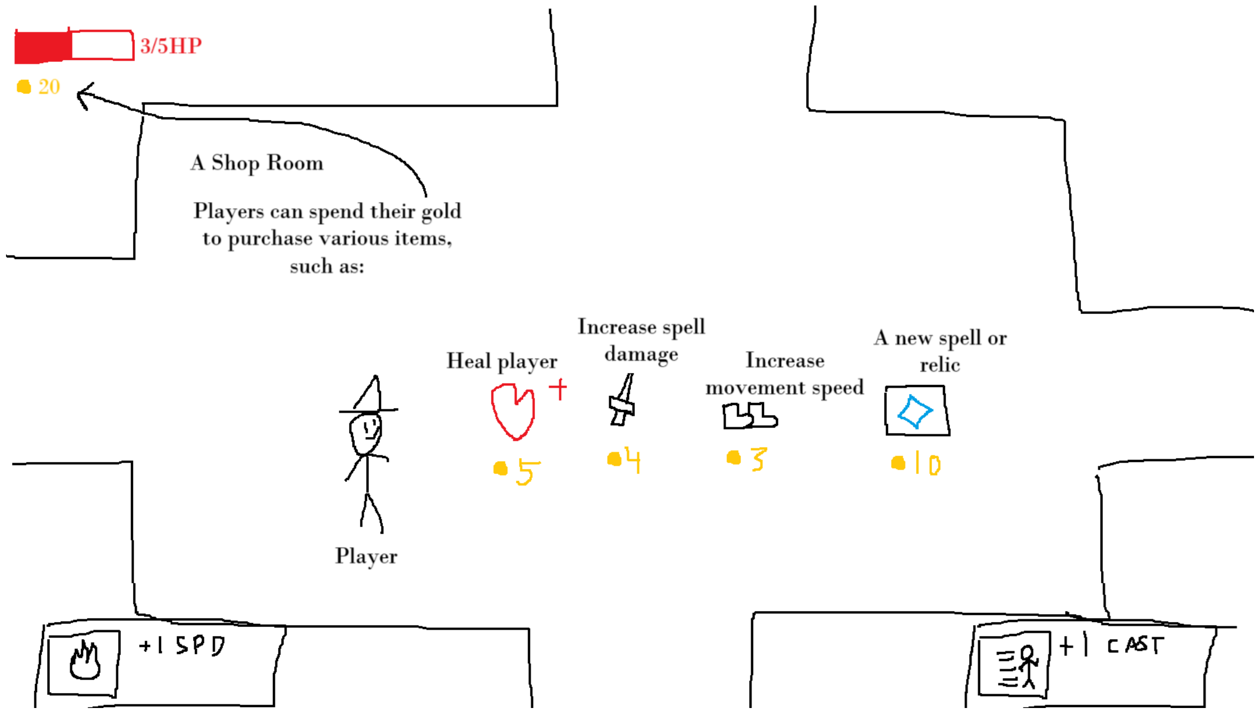
An enemy will detect the player when they enter the enemy's circular detection radius



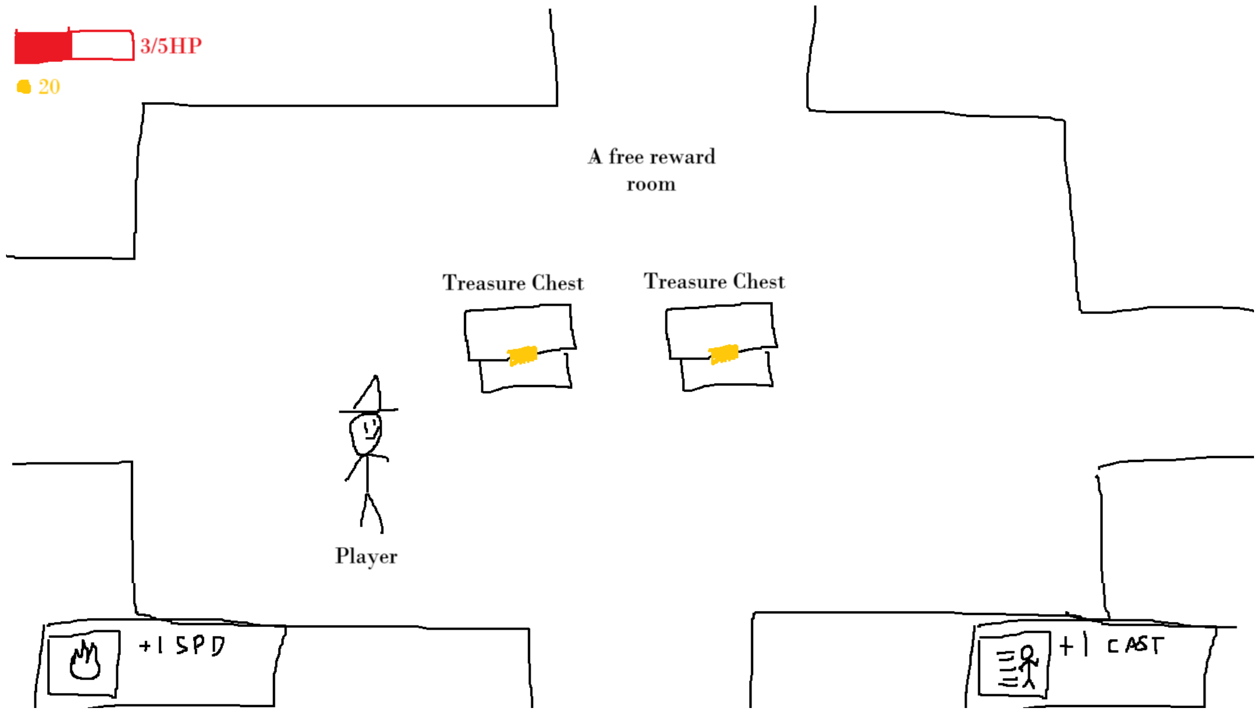
Chests have HP, and will drop the rewards once their HP goes to 0

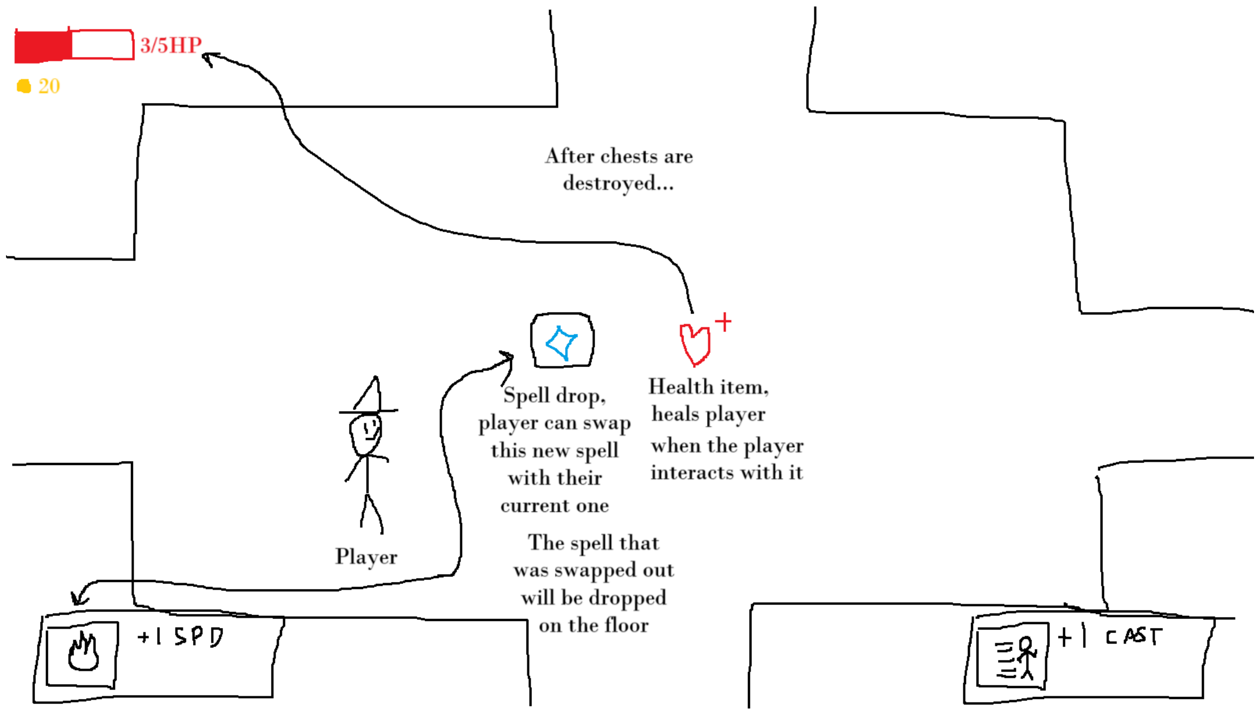
You can destroy chests for their reward before, while, or after fighting the enemies in the room

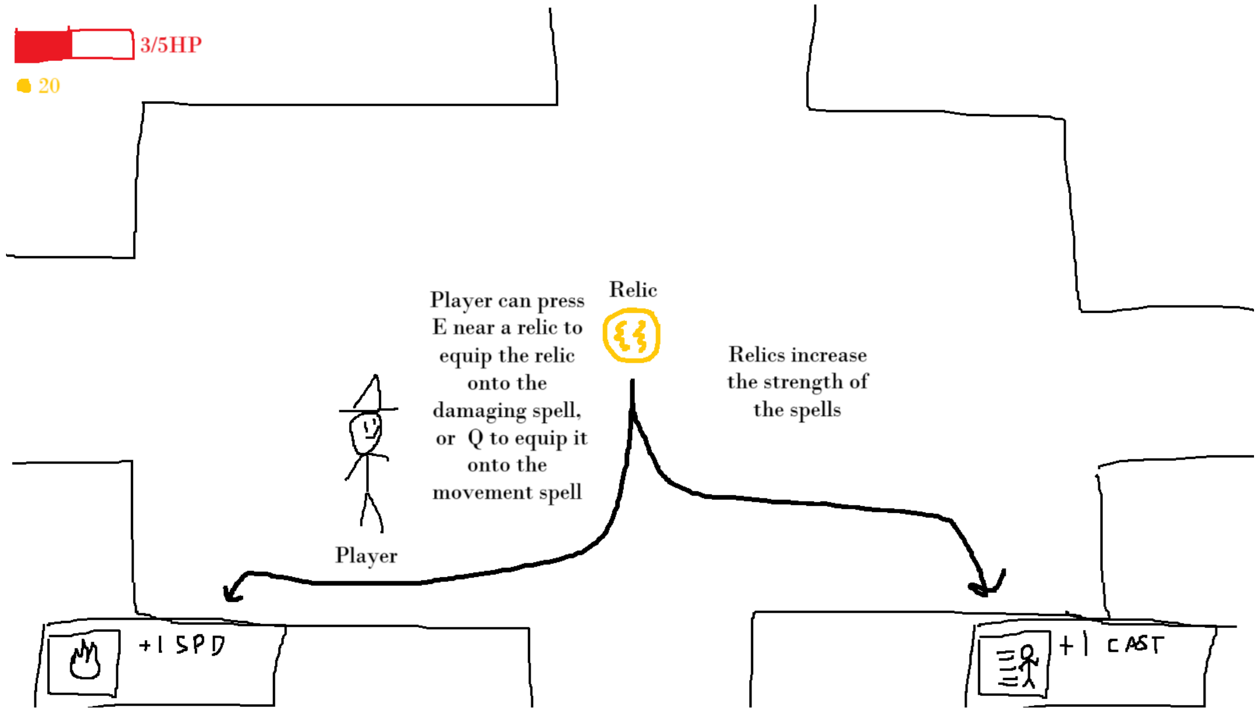
Shop Room



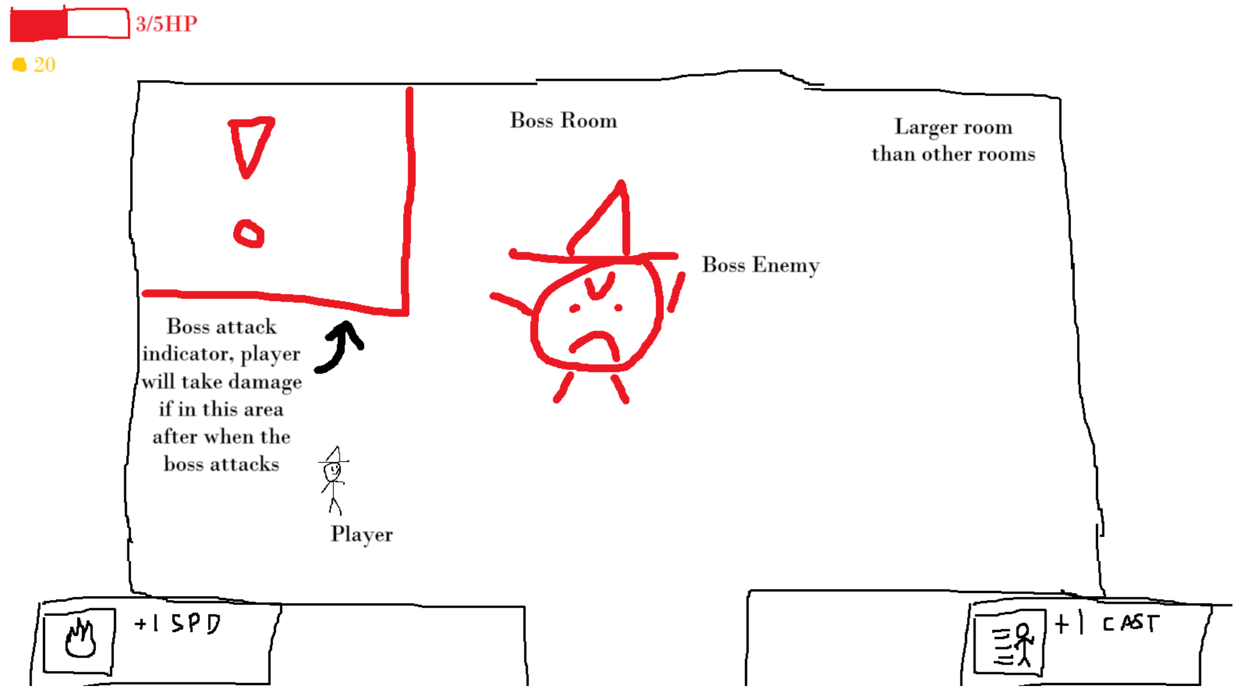
Free Reward Room

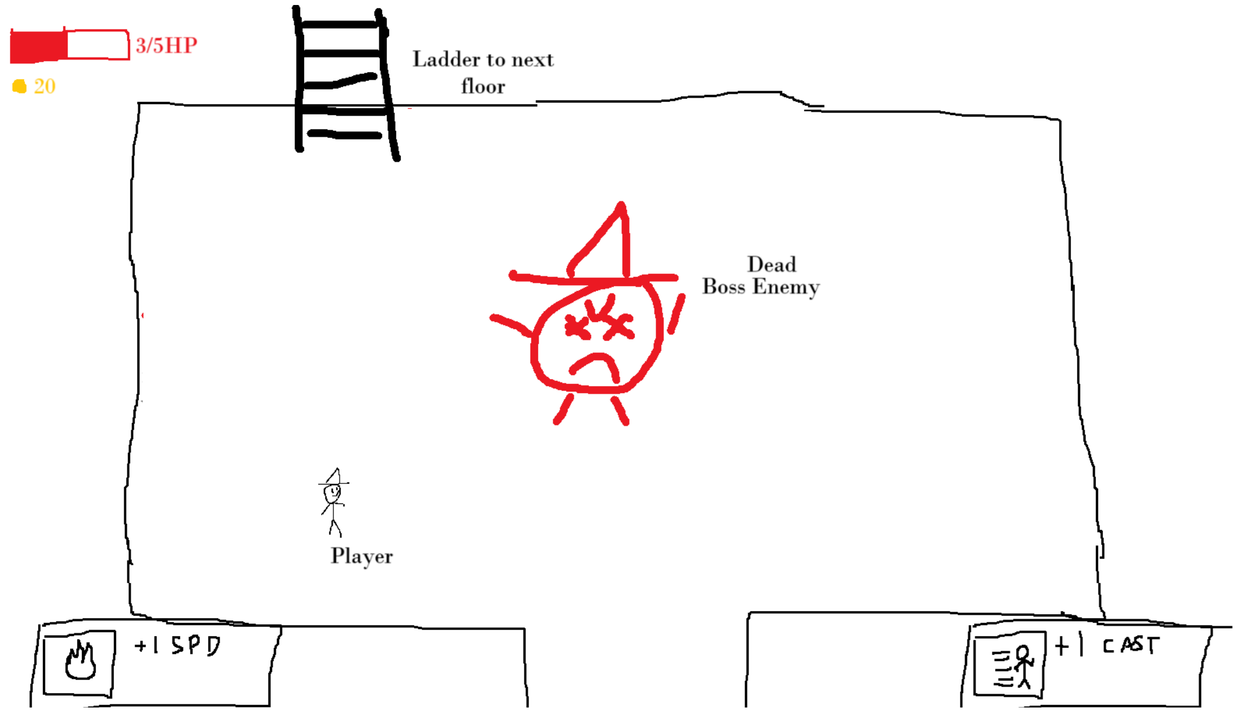



3/5HP
20
After chests are destroyed...
Spell drop, player can swap this new spell with their current one
Health item, heals player when the player interacts with it
Player
The spell that was swapped out will be dropped on the floor
+1 SPD
+1 CAST


3/5HP
20
Relic
Player can press E near a relic to equip the relic onto the damaging spell, or Q to equip it onto the movement spell
Relics increase the strength of the spells
Player
+1 SPD
+1 CAST

Boss Room


3/5HP
20
Boss Room
Larger room than other rooms
Boss Enemy
Boss attack indicator, player will take damage if in this area after when the boss attacks
Player
+1 SPD
+1 CAST


3/5HP
20
Ladder to next floor
Dead Boss Enemy
Player
+1 SPD
+1 CAST

Technical Elements:

Identify how the game satisfies the core technical requirements: rendering; geometric/sprite/other assets; 2D geometry manipulation (transformation, collisions, etc.); gameplay logic/AI, physics.

Rendering:

- **Simple Rendering Effects**
    - *Z-Sorting*: Entities with a sprite component will have a z-index, so we can draw sprites in a certain order (eg. draw the floor before the player).
    - *Y-Sorting*: Entities will be drawn from the top of the screen to the bottom, for example, an entity with position.y = 5 will be drawn before an entity with position.y = 0, so the entity at position.y 0 appears "on top" of the other entity.
    - *Camera Movement:* Our game will have a 2d Top-down camera that is locked on the player as they move.
- **Particle Systems**
    - We plan to add particles and more visual details for spells and enemies once our game is mostly functional.
    - Examples where we will use Particle Effects:
        - Chests glowing a faint golden colour
        - Spell shots emitting coloured particles depending on the spell

Lighting: We will not have any sort of dynamic lighting, all of it will be drawn into the textures themselves.

Particles / Decor / Various Non-functional Assets:

We plan on including particle effects for:

- Chests taking damage / glowing

Assets:

Sprites:

We will begin development using asset packs for placeholders, until later in development, where Munn will create new sprites or edit the existing sprites.

The game's art style will be in a pixel art style. The player's sprite will be a mage. Following the theme of the story, all enemy mage sprites will bear resemblance to the player's, but with less distinction, to signify that they are NPC/Enemies.
All projectiles will also have different sprites depending on their attributes.

- Bosses
- Floors and walls
- chests, spell icons, health pickups, shop items
- UI elements
    - player UI
    - Pause Menu
    - Options/Settings (if we have them)
    - Selecting which spell to put modifiers on
    - Title/Splash screen

Sound:

We'll need to source music and sound-effects for a number of things:

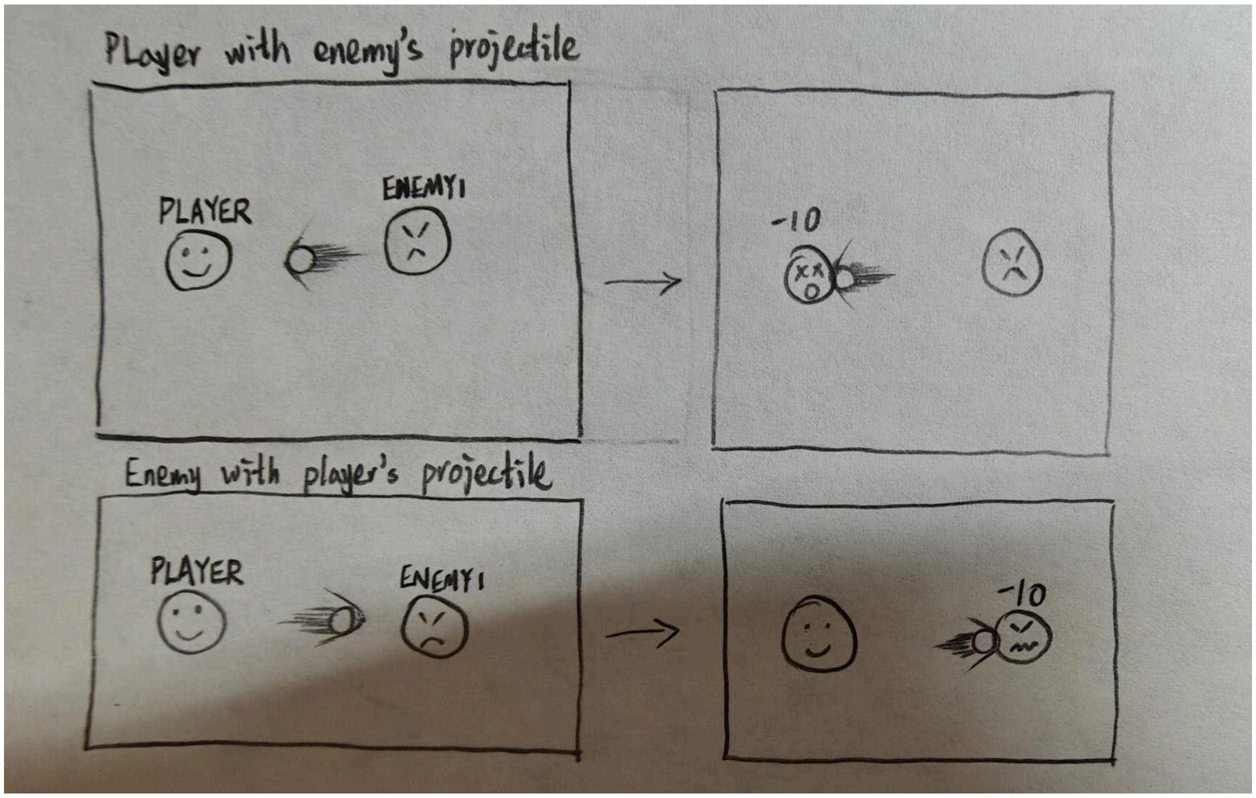
- Main Menu music
- Normal floor music
- Boss music
- Various Sound effects
    - Shooting a spell
    - Picking up a spell
    - Putting a relic on a spell
    - Getting injured
    - Using dodge/roll
    - Footsteps for movement
    - Pausing and unpausing the game
    - Going up/down a floor
    - purchasing an item
    - When seeing the Final Boss

2D Geometry Manipulation:

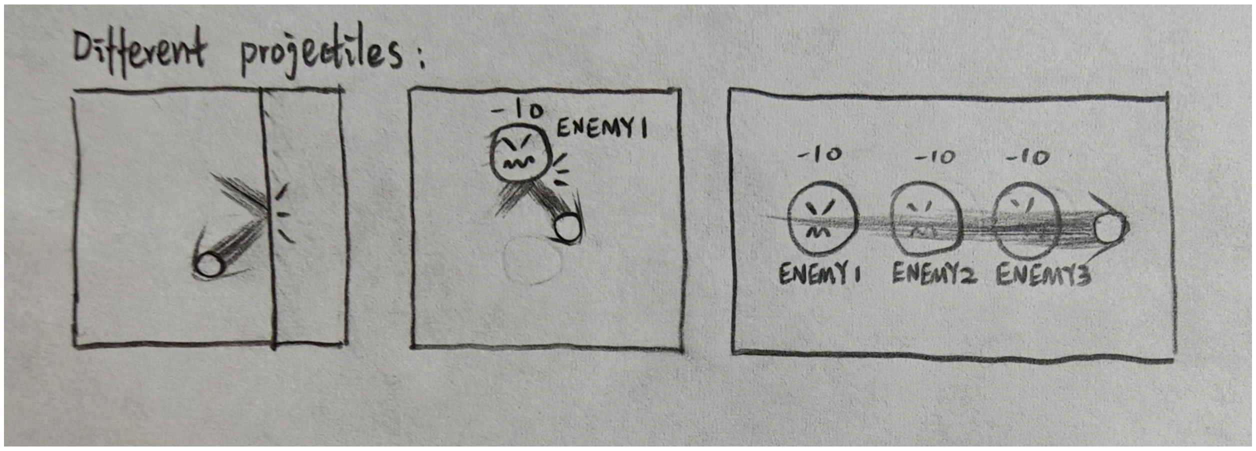
Physics and Simulation:

- Basic movement physics (eg. applying velocities to moving objects for basic movement and more complicated movement like knockback, dashes, etc.)
- Detecting collisions for the following interactions
    - Player with Enemy Projectiles
    - Player with walls/obstacles
    - Enemy with Player Projectiles
    - Hitscan attacks such as melee attacks, beams, etc.

An example of a player projectile:


Player with enemy's projectile
PLAYER
ENEMY1
-10
Enemy with player's projectile
PLAYER
ENEMY1
-10

Projectile Interactions:
Some projectiles will bounce when hitting walls, while some will ricochet when hitting enemies. Depending on the attribute some projectiles will also penetrate a set amount of enemies.



Gameplay Logic Elements:

- Start Menu
- Pause Button / Play Button
- Procedural Floor Generation
- Player Death upon health reaching zero
- Purchasing items in shop and losing the gold
- Subtracting health from enemies upon hitting them
- Interacting with spells
- Generate Spells and Relics with probabilities
- Elite upon death drops floor staircase which leads to new floor
- You move to new floor and game generates new floor upon entering staircase
- Keep track of stats and display either a “Victory” or “Defeat” screen
    - This screen should display player stats-- enemies killed, buffs taken, score, etc.

AI:

Enemy Behaviour:
Similar to a Bullet Hell game, enemies will track the position of the player. All enemies will aim to shoot towards the player accurately, with some enemy projectiles moving to track the player’s movement in real time.

Easy to implement enemy behaviours:

- circle the player at x radius, occasionally shoot spells at the player while moving
    - Might need the player moving around the whole map to find out all enemies.
- run directly at player until within X range, then stand still and occasionally shoot spells at the player(path finding algorithm to find shortest path between enemy and current position of the player considering presence of obstacles)
    - The player might not need to explore the map to kill all enemies since the enemies can find the player by itself.

More complicated enemy behaviours:

- Implement some pathfinding algorithm, follow path to player until within X range, occasionally shoot spell

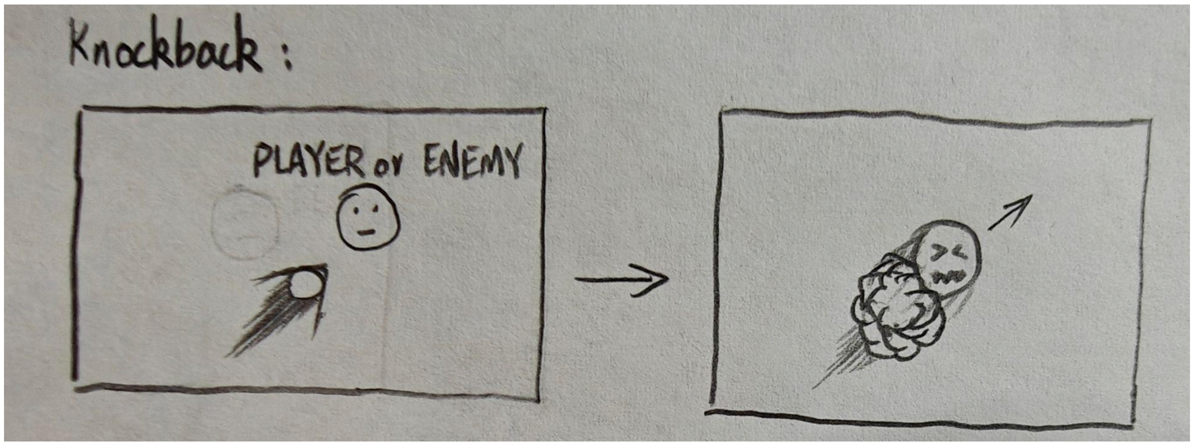
- Implement a state machine to dynamically switch between the above behaviours
- Implement some sort of “hivemind” for the enemies, so they all move/attack together
- observer pattern when an event occurs, all enemies react to it (ex. a very strong spell that stuns all enemies for 5 seconds).
- Boss fight (reach): the boss transforms into a new form upon some given event, such as when its health fall below some region etc. (decision tree)

Physics:

Movement:

Entities will move under a simple physics engine, where every moving entity has a velocity component and we move the entities by manipulating the velocity of the entity rather than directly adjusting the position. This allows us to implement the following interactions in the game:

- Player/enemies taking knockback when they are hit by a projectile
- Projectiles “bouncing” when they collide with a wall
- Acceleration and deceleration on players/enemies for smooth movement, as well as on projectiles



Advanced Technical Elements:

List the more advanced and additional technical elements you intend to include in the game prioritized on likelihood of inclusion. Describe the impact on the gameplay in the event of skipping each of the features and propose an alternative.

Procedurally Generated Floors (important)
A big part of our game consists in the “looting” aspect of it. Creating Procedurally generated

random floors allows us to scatter chests (loot) around the map such that the player never knows where things exactly will be.

If we were to forgo Procedurally Generated Floors, we'd likely use very ***simple room generation instead, where each floor would consist only of one handmade room.***

In this case, rather than having enemies spawn throughout the floor, we'd have enemies appear in waves. We would also have loot spawn periodically. This would drastically change the gameplay loop, as rather than exploring through a level to find the loot you need to strengthen your character (and sometimes maybe opting to rush through a floor in order to conserve health), you'll have to face EVERY enemy that would spawn.

As such, the player would have less choice about what risk to take, as they'd have to always take the same amount of risk.

Text Encounters Between Floors (less important)

We want to implement text encounters between floors. In these text encounters, we'll give the player:

- Some lore about what is happening in the Wizard Tower
- Dialogue options to choose from, all of which have some effect on either the players level of strength, or their future encounters.

After which, the player will proceed to the next floor.

This is completely optional, as it does not detract from the gameplay if it is not included.

Devices:

Explain which input devices you plan on supporting and how they map to in-game controls.

We will support the following mappings on mouse and keyboard:

- WASD to move up, left, down, and right, respectively
- Move mouse cursor to aim
- Left click to cast the damaging spell
- Shift to cast the movement spell
- E to interact
    - Purchasing items from the shop
    - Swapping spells with your current spell
    - Apply a relic to the damaging spell
- Q to apply relic to the movement Spell
- Esc to pause – Pause is a reach goal

Controller support is not essential to the core of our game. As such, we've decided to exclude it from our scope.

Tools:

Specify and motivate the libraries and tools that you plan on using except for C/C++ and OpenGL.

We will use SDL for audio, and GLFW for windowing.

Team management:

Identify how you will assign and track tasks and describe the internal deadlines and policies you will use to meet the goals of each milestone.

List of things to delegate: basically everything in the "Technical Elements" Section

Roles:

THINGS WE NEED TO COVER STILL

Munn:

- Rendering floors/walls/environment
- Collision and hitboxes
- UI elements for main menu and defeat/loss screens
- Asset creation
    - Sound effects:
        - Player/enemy movement
        - Spells being casted
        - Player taking damage
        - Enemies taking damage
    - Sprites: drawing pixel art sprites for player, enemies, floor/wall tiles, etc.

Jason:

- physics design:
    - physics of different projectiles/spells of player and different types of enemies
    - attribute and physics of player & different types of enemies

    - player/enemy movement
- player interaction
    - health logic of enemies/player
- Writing (story, lore, any text that shows up in the game)

Kevin Guo:

- Project Management – making sure we're meeting deadlines and requirements
- Gameplay Logic
    - Spell and Relic concept designs
    - attributes of all spells/attributes characters and scalability
- Rendering sprites onto the screen + camera
- Game state / scene manager

Kevin Shaw:

- Music
- Level Design and Procedural Generation of Levels
    - Shop implementation
    - Spawning Enemies In
- Gold and currency gained upon enemy death
- Spell/Relic Mechanics
- (I can also help with Github stuff. Like if there are weird PR conflicts I can help with that. This is just something additional and probably pretty small in addition to the other things I'll be doing.)

Mark:

- Enemy AI
    - Enemy Types
- Input Handling
    - Input from keyboard, mouse
- Rendering
    - Camera Movement: Camera locked on the player
    - Particle Systems

Internal Meetings:

We have two weekly meetings:

- Wednesdays in person after tutorial until roughly 6:30 - 7:00pm (Longer meeting)

    - Ensuring the code is merging and building well-- building every Wednesday together (Though of course we'll be building it constantly separately)

    - Anything people generally need help with? (For example: resolving git merge conflicts. Whatever comes up, we can delegate to a pair of individuals to resolve between themselves.)
    - Reflections on the week
    - Discuss coding challenges throughout the week
    - Gameplay direction reflection
    - Backlog assessment
    - Task delegation
- Saturday online at 8pm (Brief check-in)
    - Are there any problems with the codebase? Anything come up that we should be aware of?
    - Any current struggles? Any difficulties accomplishing tasks? Need clarification for anything?
    - Any merging questions or anything similar to that

Internal Github Policies

- *Branch Management Policies:*
    - we will have the Main Branch, which we will NOT push directly to
    - For every issue, whoever is working on it creates a separate branch off of main and works on the issue there.
        - this branch will also reference the issue that it is solving
    - Afterward, they submit a Pull Request into main to merge their separate branch into the main branch
    - After receiving one approval, then that branch will be allowed to be merged in
    - Immediately afterward, they will check out the updated main branch, pull the code off of the Repo, and attempt to build and run. It should work.

        - If it doesn't, either:
            - Rewind your changes
            - Fix the problem and submit a hotfix, pinging @everyone on Discord for merge support
            - Request help from another teammate using @everyone on Discord

- *Review Priority:*
    - We're agreeing to prioritize reviewing code over producing code, minimizing the period between PR Submission and merging
- *Issue Management:*
    - Tasks will be tracked using Github Issues, in the form of User Stories
    - Tasks will be tagged under their respective categories, using both Milestones (to indicate project milestone, as well as reach) and Labels (to indicate the type of role each one belongs to)

Code Style

We will use a linter to ensure our code is consistent in style. We will also create a code style document to keep naming conventions as consistent as possible.

Development Plan:

Provide a list of tasks that your team will work on for each of the weekly deadlines. Account for some testing time and potential delays, as well as describing alternative options (plan B). Include all the major features you plan on implementing (no code).

Milestone 1: Skeletal Game

Week 1

- Implement a basic ECS and update loop
- Implement input handling and simple player movement
- Simple rendering of Entities + camera movement:

        - Player
        - Placeholder enemies
        - Placeholder chests
        - placeholder floors/walls
    - Begin simple map Generation
        - Rooms will be rendered with placeholder sprites
        - Shops spawn, though have nothing in it
        - Chests spawn, and have an item randomly associated with
        - Elite rooms designated
    - Player can move around the screen

Week 2

- Complete simple map generation (as listed above)
- Simple enemy AI for one type of enemy
- Player can shoot a projectile
    - Projectile can collide with an enemy and trigger some sort of response (perhaps deleting the enemy) and remove itself
    - Projectile can collide with walls and remove itself
- Player can cast a movement spell which increases their velocity in a direction specified by their WASD inputs
- Record video of current game state, detailing each member's contribution and what we've developed so far.
- Manual gameplay testing

Milestone 2: Minimal Playability

Week 1

- Player can clear a floor by interacting with a floor exit
    - Possibly just one floor still
- Simple enemy design for multiple enemies
- Implement Spell/relic system
- Health logic, player/enemy death
    - Player death resets the game
- Map Generation generates:
    - Complete shop with purchasable items (note: not all of them will be functional yet):
        - Spells
        - Spell Upgrades
        - Healing
        - Stat buffs
    - Exit staircase that completes the floor
    - Chest that is destroyable to drop interactable spell

        - Elite Enemy Room (No elite enemy yet, just empty room)

- Simple assets
    - Replace placeholder sprites
    - Sound effects

Week 2

- Player can interact with spells, which will swap it out with the current spell
- Interactable objects:
    - Shop items
    - Spells
    - Relics
- Simple enemy AI for multiple enemy types
- Implement more damaging spells and movement spells
- Gold system, get gold by killing enemies
- Map generation generates:
    - Chest that is destroyable to drop interactable spell
    - Elite Enemy Room
- Simple assets
    - Replace placeholder sprites
    - Sound effects
- Manual gameplay testing

Milestone 3: Playability

Week 1

- When you clear a floor, a new floor is generated, and the player moves to the new floor
- Player must kill an Elite which drops a floor exit now
- Implement main menu and UI buttons
- Elite enemy AI designed
- Render animations for enemy, player, and projectile sprites
- Player death causes the scene to transition to a loss screen, which will then lead to the main menu again
- Start creating adding more complete assets for:
    - Sprites
    - Sound effects/music

Week 2

- Elites now spawn during the Map Generation
- Implement introductory and ending cutscenes
- More types of enemy AI design.
- Final Boss designed, complete with its own AI, with Final boss room

- Manual gameplay testing

Milestone 4: Final Game

Week 1

- Tutorial Floor
- Generate random aesthetic assets as part of map
- Debug all of the potential problems in the program.
- Possibly add text rooms in between floors to add narrative element and more player choice between dialogue options that impact the run
- Make sure all of the error handlers are robust and handles all known circumstances
- Particles for spell projectiles and interactable items

Week 2

- Fix any random bugs
- Create Game Development and Evolution Video
- rehearse to present.
- Manual gameplay testing