Game Proposal: Eviction of the Damned

CPSC 427 – Video Game Programming

Team: Team Elephant

Luke Joe 69301273

Logan Keener 67439018

Wendy Greening 78836632

Aayush Behl 26071365

Derrick Cheng 94068467

Story and Gameplay:

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.

The game is set in an apartment building where the landlord is a black magic practitioner and demons possess all the tenants. The player is a tenant in the same apartment building who is back from a month-long vacation, only to find his neighbors possessed by demons. The player's goal is to exorcise all the tenants by defeating them in battle, making their way to the landlord, the story's main villain. The game is a top-down dungeon crawler where each floor is a dungeon and contains a possessed tenant as a boss.

Each dungeon will feature one main boss, the tenant of that apartment unit, and several smaller enemies that will appear before. The player needs to first clear out the apartment of the smaller enemies. The last enemy defeated drops a key to the boss's room (key drop feature to be implemented if time permits, else the door will just unlock). Once the player enters this room, the boss fight begins. The dungeons also consist of other elements like obstacles (furniture, terrain that slows the player down) and items to restore health. The final dungeon of the game will be the landlord's office and the final boss will be the landlord.

The enemies can do melee attacks, ranged attacks, or both. The player can only use melee attacks using the currently equipped weapon. Players have a variety of melee weapons to choose from, each varying in speed, damage, and range. Players gain access to more weapons as they clear more dungeons. All of the picked-up weapons can be accessed later using a weapon wheel. The player can move around the map, attack, and do an invincible dash action - a short burst of movement during which the player will not take damage. A stamina mechanism will ensure that players cannot overuse the dash feature.

The main challenge for the player is to dodge as many of the enemy attacks as possible while attacking the enemy at the same time. The goal for the player is to damage the enemy enough to deplete their health bar, causing a successful exorcism. Once a tenant is exorcised, the

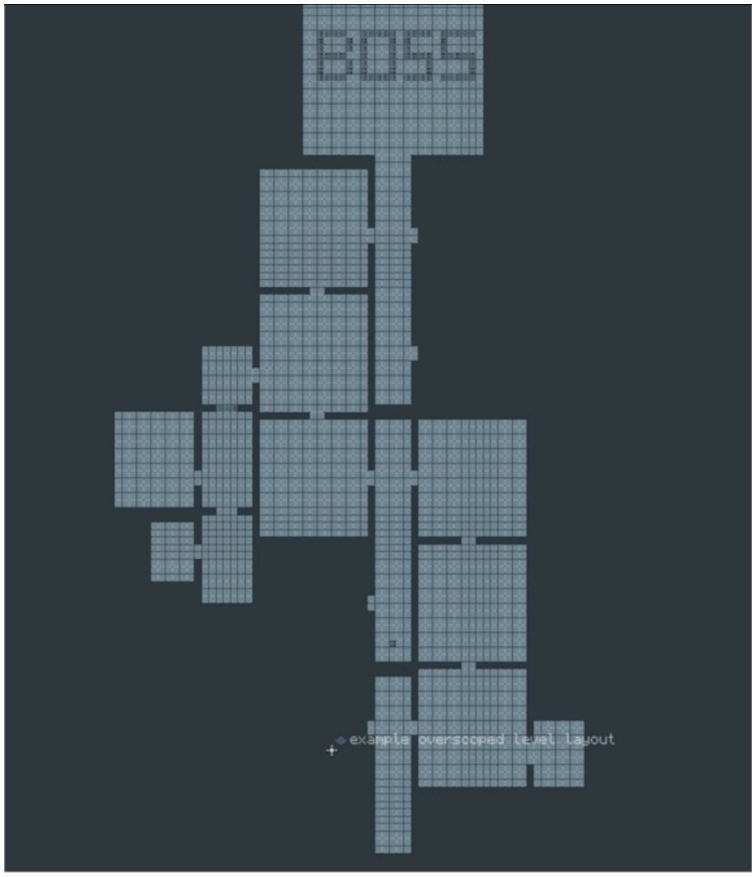
player can talk to the tenant, who as a token of thanks, gives the player items like keys to the landlord's office, new weapons, and items that give players permanent buffs (for example +5% chance of a critical hit).

Scenes:

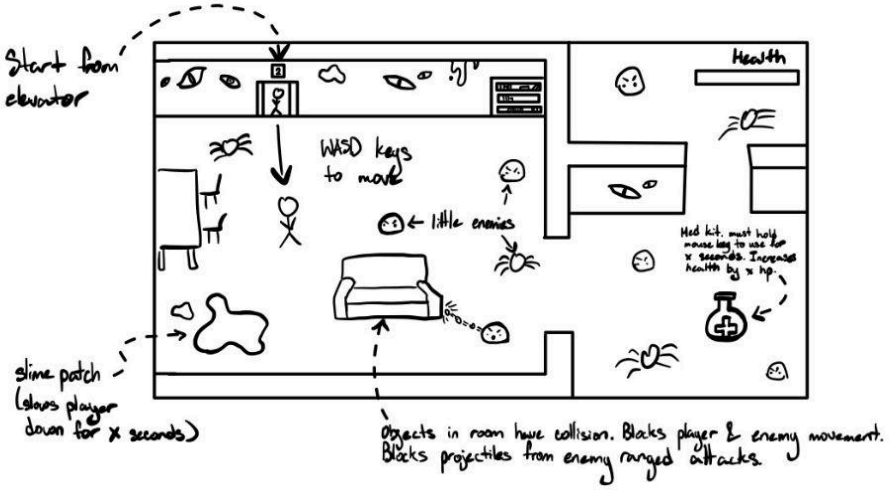
Produce basic, yet descriptive, sketches of the major game states (screens or scenes). These should be consistent with the game design elements, and help you assess the amount of work to be done. These should clearly show how players will interact with the game and what the outcomes of their interactions 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.

Start Screen

Example Layout Of All The Rooms In A Level

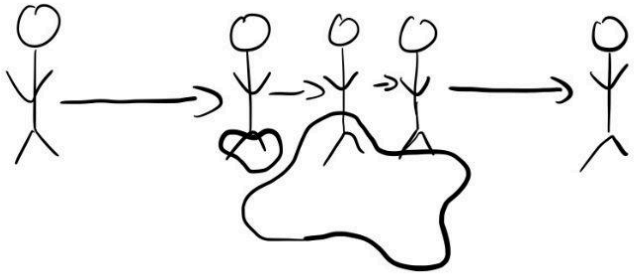


Example Level Screen (Before Exorcism)



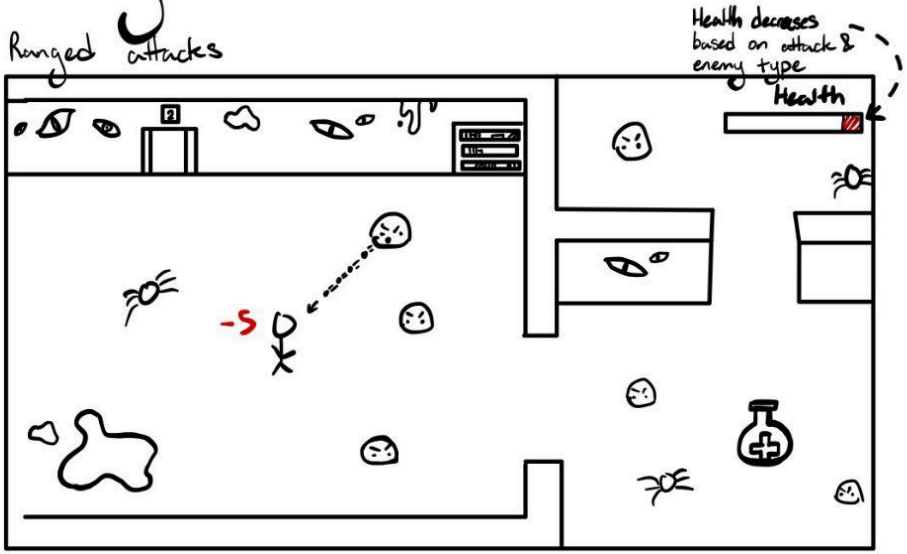
Traps

Example: sticky slime patch - slows speed while on it

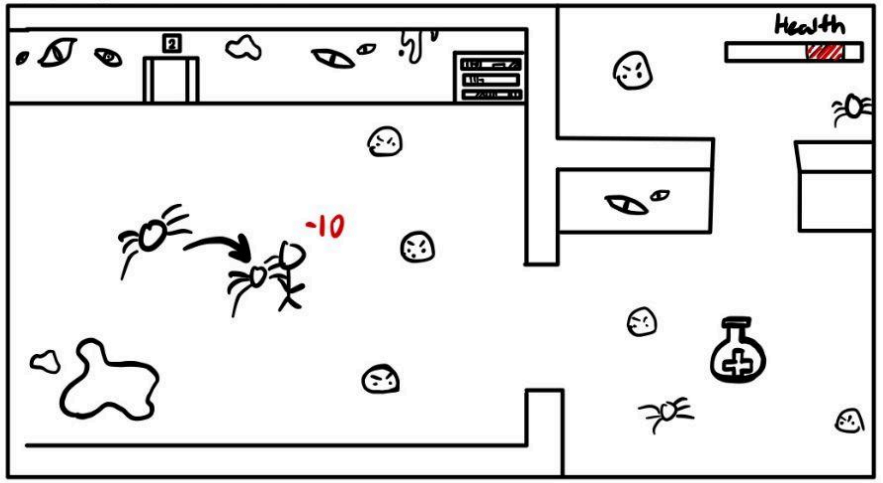


Enemy Actions

Ranged attacks

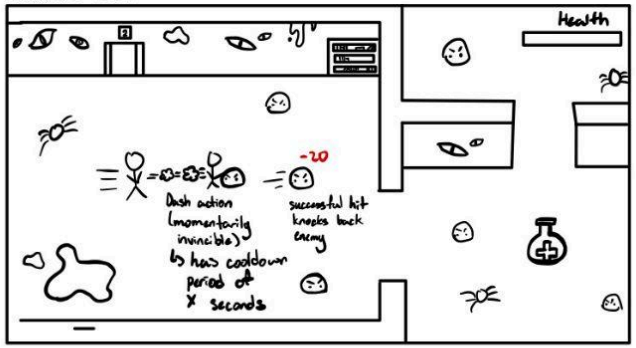


On Contact Attacks

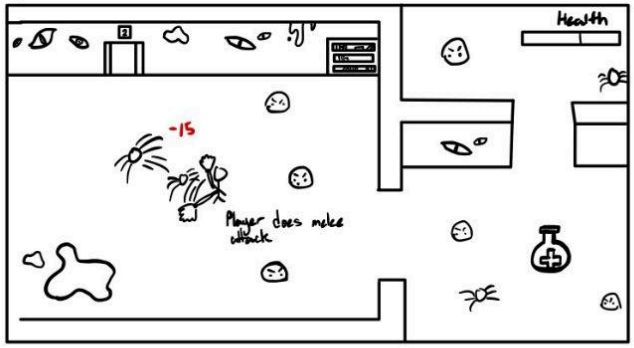


Player Actions

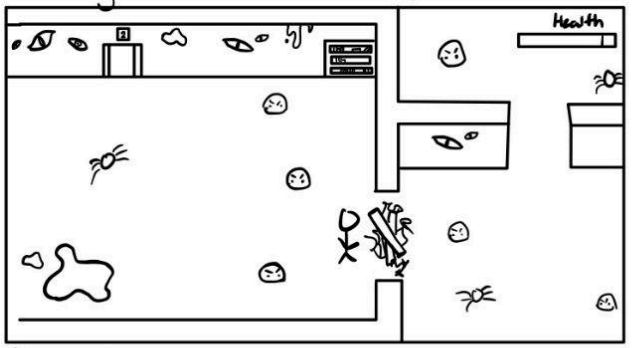
Invincible Dash



Attack

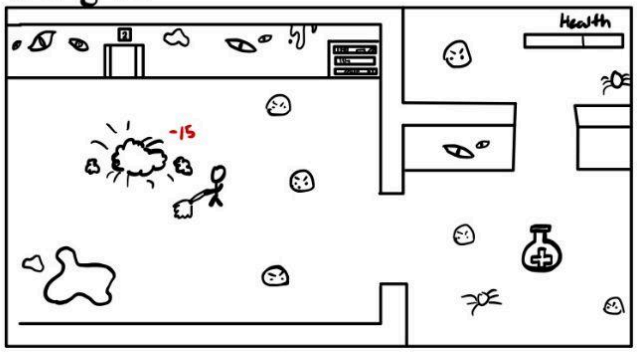


Removing Obstacles - fallen debris (Advanced Technical Element)



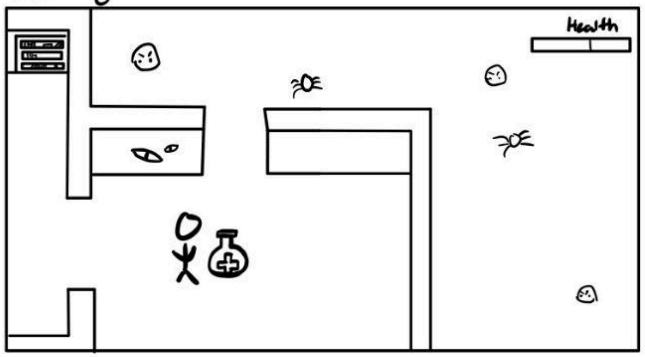
Requires holding click/button mashing on object to move.
Cannot attack enemies while removing obstacles.
Cannot go into next room until obstacle is removed.

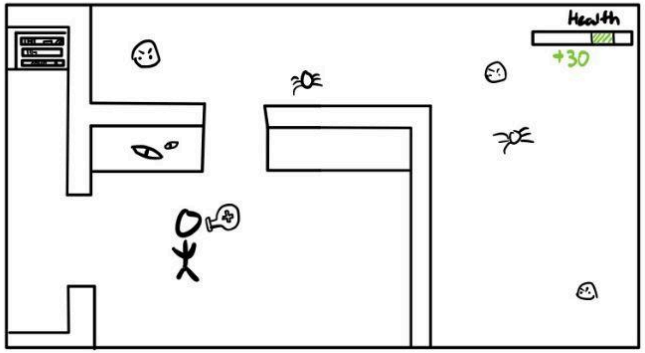
Enemy Killed



When enemy's health reaches zero.
Killing the last little enemy on this level automatically unlocks room to fight the possessed tenant.

Healing





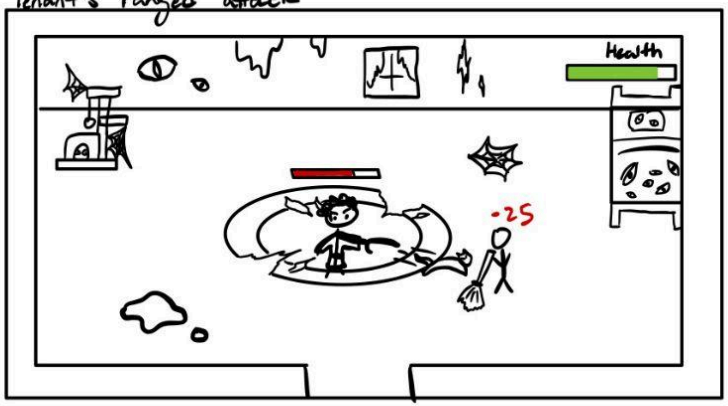
Hold mouse click on location to use single-use healing object.
Increases health by x amount

Fighting Possessed Tenant (main enemy of level/floor)

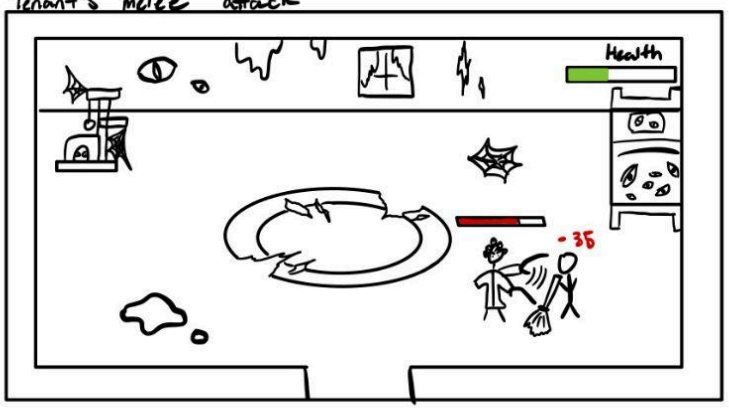
- Occurs in last room on level. Last enemy killed gives access to this room.

Possessed Cat Lady Example

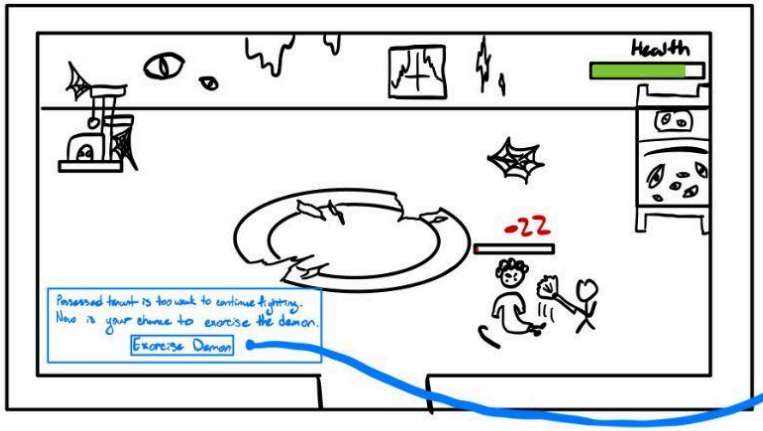
Tenant's ranged attack



Tenant's melee attack

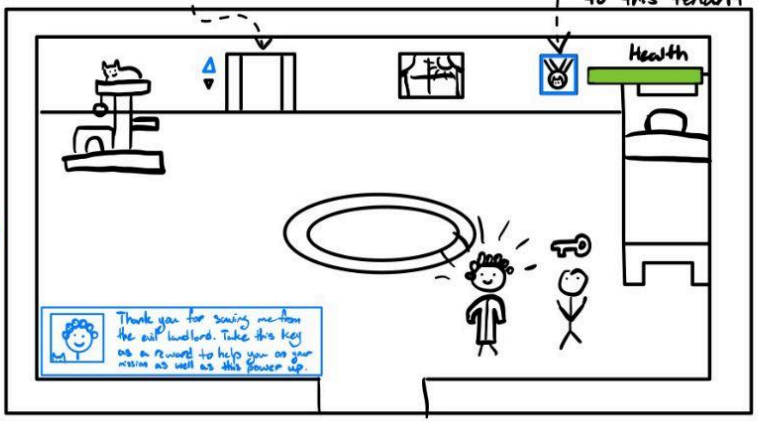


Defeating/Exorcising Possessed Tenant



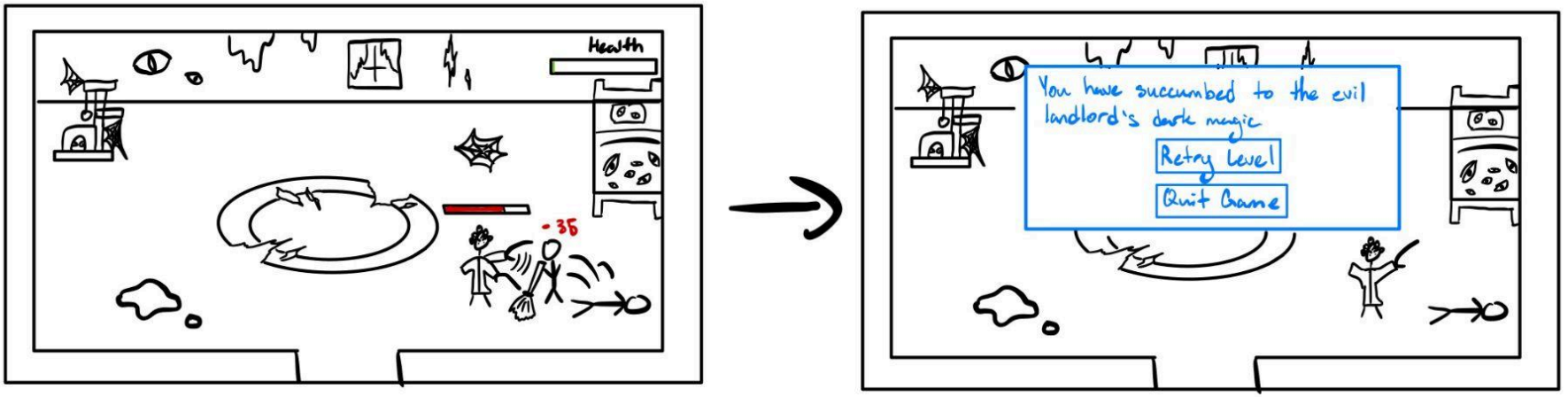
Elevator to next level appears

- Gained medallion with new buff unique to this tenant

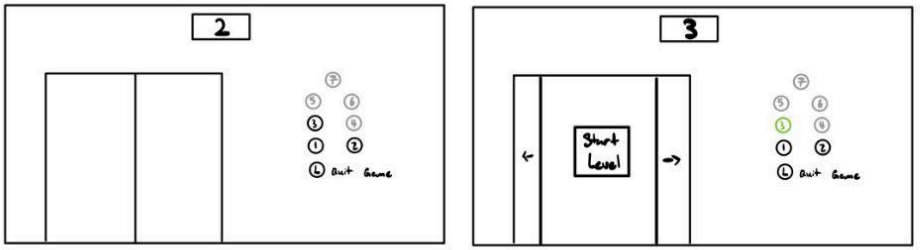


Room appearance returns back to normal

If Player Dies



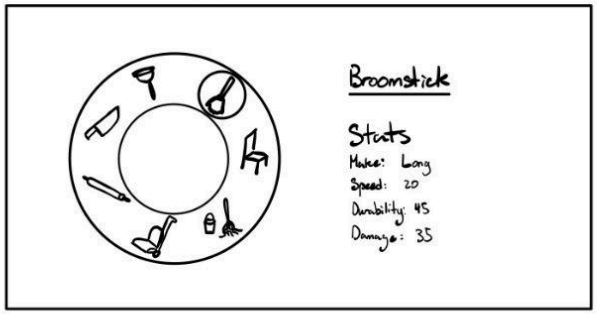
Elevator (Menu)



Inventory (Weapons)

open by clicking 'E' key

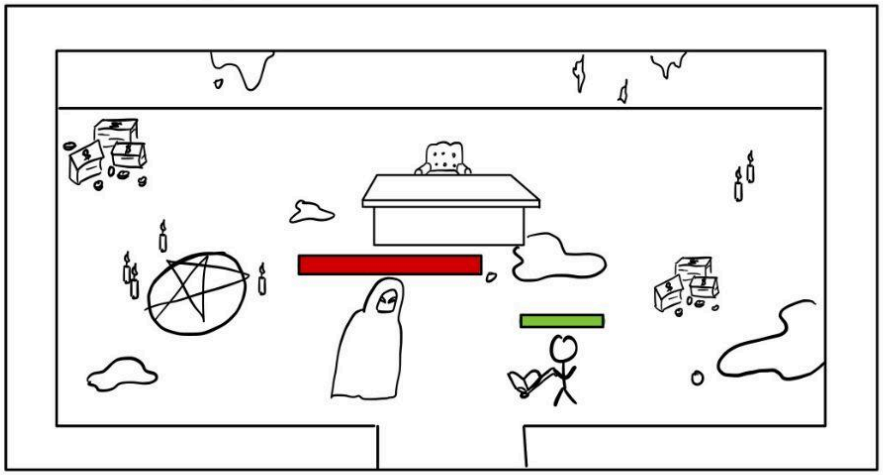
scroll to toggle item



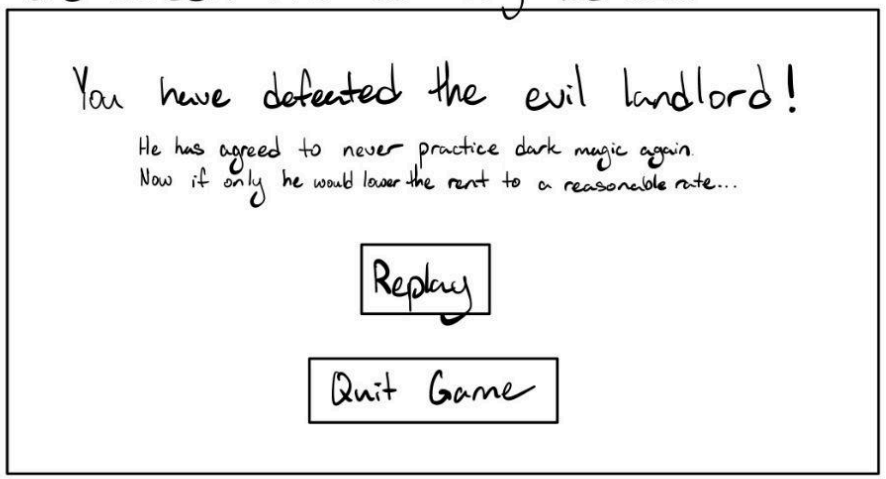
Boss Battle (Last Floor)

- Mechanics are similar to fighting a possessed tenant but the boss (evil landlord) has much more health.

Boss battle room



End screen after defeating the boss



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.

1. Rendering: The game is a top-down 2D dungeon crawler, which means top-down pixel art will have to be rendered. Rendered elements will include dynamic elements like the player, enemies, projectiles, etc., and static elements like the background, obstacles in the dungeon, etc.
2. Geometric/sprite/other assets: The game will use 2D sprites for in-game elements like the player, tenants, weapons, etc. Each dungeon will also have its own theme tiles for the walls, floor, and obstacles.
3. 2D geometry manipulation: The game requires collision detection to stop entities from running through walls/obstacles, and to collect items when they walk through them. Collision detection is also required for combat logic, to determine whether the player or an enemy has taken damage.
4. Gameplay logic/AI: The player will be able to do regular movement, dash movement, and regular melee attacks. The enemies can do both melee and/or ranged attacks, depending on the enemy - some enemies will inflict contact-based damage. If an enemy's attacks collide with the player, or vice versa, the attacked party loses a certain amount of health. If an entity runs out of health, they are rendered inactive (game over in case of the player, enemy exorcised in case of an enemy). The enemy AI will include pathfinding that allows the enemies to make the best possible attack in the best possible direction to damage the player. Once defeated, the player can pick up any dropped items from the enemy, which will need to be saved in the player state.
5. Physics: The key drop animation (if implemented), player and enemy attacks, and ranged attack paths will be controlled by gameplay physics.

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.

We would like to include a light source that creates shadows, which would be a more advanced rendering technique. It would look interesting to make the lights flicker like they're broken and it would add to the "creepy" look of the game that we're going for. The alternative if we are not able to accomplish this would just be to not have it and the game would still look okay.

Another element we would like to add is a button-mashing element to lift fallen objects off the ground - as the player presses faster/more, the push gains momentum and is lifted into the upright position. An alternative would be to scrap the fallen object obstacles altogether or have it be just a button that you need to hold down and we make an animation that makes it look like the object is being pushed up very fast after the button has been held down for some amount of seconds, so there would be no physics component required.

A third idea we had was (if we decide to give tenants special attacks) having a tenant whose special attack is summoning a swarm of fruit flies that can chase the player around, that can be animated using a BOIDS swarm simulation. This might be hard to implement, so we may not include it.

Devices:

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

We are planning to support a mouse and keyboard setup. Player movement will be mapped to WASD. Movement in the direct cardinal directions will be allowed based on the key pressed, as well as diagonal movement if two input directions are pressed at once. A to be determined number of keys near the wasd keys will be added to allow for usable abilities and items. The left mouse button will be used to attack and select things in the in-game ui. The right mouse button will be used for a dash that provides invulnerability. Mouse movement will be used for aiming and camera movement. There will be a key setup for character info, and allow the player to manage their gear and abilities. The space key will be used to interact with objects such as hazards or buffs. Finally, the escape key will open an options menu for in-game settings.

Tools:

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

https://www.raylib.com/examples.html

SDL2

GLM

Freetype

RecastNavigation

We plan on using SDL2 as a library to render 2d images, provide access to audio, and capture input from the keyboard and mouse. We will also be using GLM as a mathematics library to handle physics simulation, movement, and ray tracing for interactions between the player and enemy AI. Some additional libraries that could be used include Freetype to render fonts for the UI as well as RecastNavigation to handle navigation meshes and pathfinding for the enemy AI.

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.

We are planning to use an agile structure for our team management, with weekly meetings reflecting on what has been done and how it compares to our goals set in the development schedule. Our current goal for dividing responsibilities is this:

- Player movement/attack and controls: Logan, Aayush
- Enemy design + ai: Logan, Derrick, Aayush
- Combat system (weapons, stats, weapon storage, etc): Wendy, Derrick
- Art: Wendy, Logan, Luke
- Rendering: Derrick, Luke
- Level design + scene transitions + “random” generation of enemies and objects in level: Wendy, Aayush
- Sound: Luke

We will try to comment and document our processes as much as possible, so that if one team member needs to have their workload reduced or wants to switch assignments with somebody else, that change will be as seamless as possible. Every week, we will assign new tasks according to everyone’s capacity for the upcoming week, and adjust the schedule/scope as needed. There are some things we haven’t decided in the development plan (ex: how many different weapons we will include in the game) because that’s something we want to decide on after we’ve seen a few iterations of the game and have a more solid idea of how it will fit into the larger final product.

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

Creating basic Entities/Components: enemies, players, objects, etc.

- Player entity
- Level enemy type 1: slow speed, contact-based damage
- Level enemy type 2: high speed, contact-based damage
- (May scrap later adjusting for scope) Level enemy type 3: slow speed projectile shooter
- Wall, ground, furniture

Player movement: basic up, down, left, right controls + camera following player around

- If time allows: add dash movement for player (requires also adding stamina to stats)

Player stats: health bar and speed

Character art: idle + movement animations, dash animations if we do that

Week 2

Design level 1 - crazy cat lady tenant

- Level 1 will include collisions w/ walls and furniture
- We will use the same map for all levels unless there is time to design more, variance will be ensured by semi-random generation of enemies and objects

- In skeletal game: add a hardcoded trigger to reveal the area that will be the mini-boss fight room + hardcoded locations for objects and enemies

Background art: walls, furniture, ground tiles

- If this doesn't get finished by the end of the week, that's okay - we would like to have it finished by the end of M2 though

Rendering / Lighting: making sure that objects and their collisions are all rendered properly

Milestone 2: Minimal Playability

Week 1

Enemy art: movement animation, attack animations for tenant mini-bosses

- Work on this throughout week 1 and week 2. First get the idle position for every enemy and then work on movement (only a few frames desired for each animation)
- Art for "possessed" tenant can just overlap original art - add something like red eyes and dark particles that hover around them

Enemy AI: basic patrol/chase for level enemies, shortest pathfinding logic for tenant mini-bosses

Interactable objects:

- Single-use health objects around the apartment (have a set amount of time it takes to heal with them, during which you cannot move or attack)
- Nice to have: fallen objects that require button mashing reflected in-game by the player gaining momentum then (if enough button mashing is completed) moves object completely - if it is not completed the object falls/slides back into its previous position
- May also add treasure chests that have new weapons or power ups
- *Note: we can also consider adding these during M3 or M4

Combat system + HP bar (finish by the end of week 2)

- Short melee and long melee weapons for player
- Projectile shooting abilities for some tenants (and possibly one level enemy)
- Before implementing weapon inventory: just use numbers on keyboard
- Stats for each melee weapon: speed, durability, damage amount

Week 2

Object art: health objects, treasure chests if added, fallen objects

Start screen, end screen, pause menu, elevator between levels

Establish a basic combat AI for tenant mini-bosses and landlord final boss

If time/capacity: add a special ability/attack unique to each tenant (or however many we would like - all would be best for consistency)

- Would really like to have a tenant that can summon a swarm that attacks the player

Nice to have: particle-scattering effect after killing level enemies

UI and FPS counter

Help/Tutorial for controls

Milestone 3: Playability

Week 1

Finalize # of levels and design for each one - finish level design by the end of week 2

- Most assets will most likely be very similar for the same objects in diff levels (same structure/shape, maybe different color scheme or details)

Add key that appears after last level enemy is killed to collect and use on tenant's bedroom door

- Can be scrapped if there is not time, before this we can just have the basic toggle locked/unlocked and alert player when it unlocks

Decide on amount of weapons we want to include, + bonuses that the player will gain after each level

- add circular scrolling weapon holster that pauses the game when you want to change equipped weapon

Week 2

Fine tuning stats for each enemy and weapon

Fine tuning "random" generation for enemies and objects in levels

Audio system - add background music

Milestone 4: Final Game

Week 1 + 2

Finish any art that is not finished

Add advanced technical components if possible: button mashing that accelerates "push" movement on fallen objects, rendering "shadows", particle scattering effect

Add extra visual effects + SFX to select movements and actions