Game Proposal: GunCat

CPSC 427 – Video Game Programming

Team: Channel 5

Anthony Hayek, 56488752

Ayden Kinchla, 62172168

Diana Dou, 84855964

Kurtis Ho, 70904560

Raizen Banzon, 62408000

Sally An, 30397012

Story:

GunCat begins with you bursting out of a government facility in the middle of a major city center. Radio chatter alerts you that you have been in a coma for the last several months, and that you are to be feared. You're the sole survivor of an experimental super soldier program, and when you awaken, you are surrounded by dead clones. You are also a cat. The city is under attack by monsters and the military has begun pulling back to initiate Protocol: Scorched Earth, an emergency protocol to completely destroy the city. As such, it is up to you, GunCat, to fight off the monsters with your own four paws while there's still a city left to save.

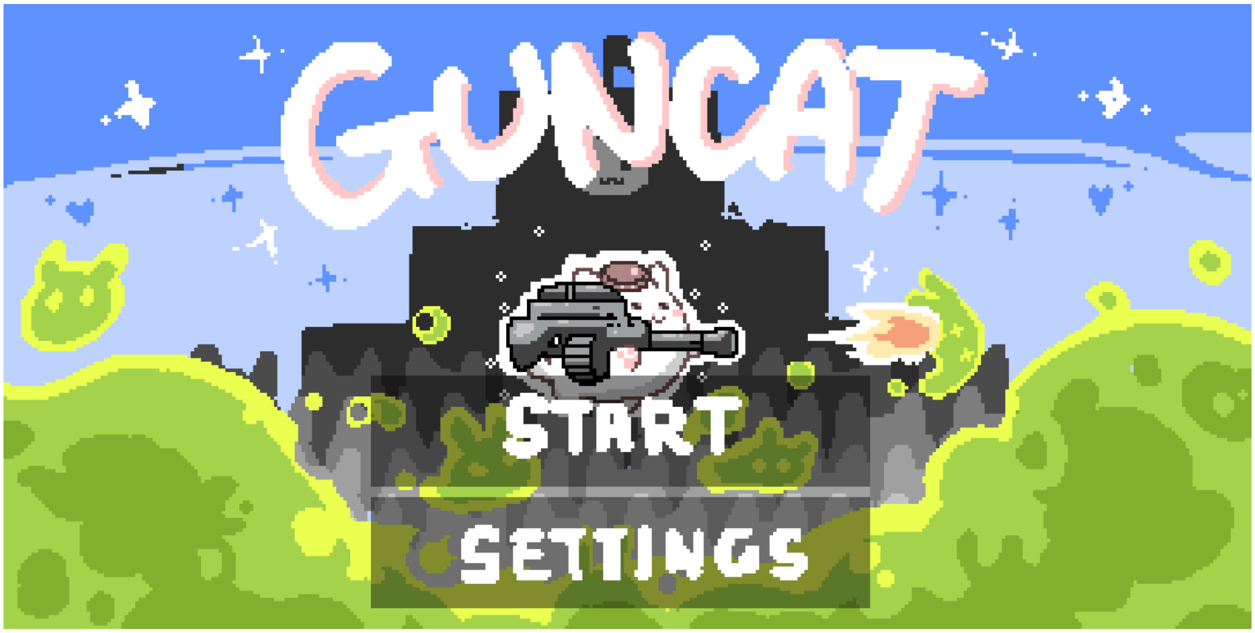
You will be working your way through the city, level by level, concluding each by taking out a powerful monster that causes the rest around it to scatter. Levels will include combat encounters, traversal sections, and secrets. Collectables include new weapons, new outfits for your cat, and reports left behind during the military evacuation that shed some light on The Military, the monsters, and your own dark past. The tone is absurd but the story is played completely straight.

Levels will be different parts of the city, an apartment building, a mall, a park, and you can navigate between them from a level select screen in the main menu. Your nature as a "super soldier" is apparent in your physical capabilities, with you regularly using the recoil from your weapons to rocket quickly throughout the levels. You are fast, you are durable, and you are fierce.

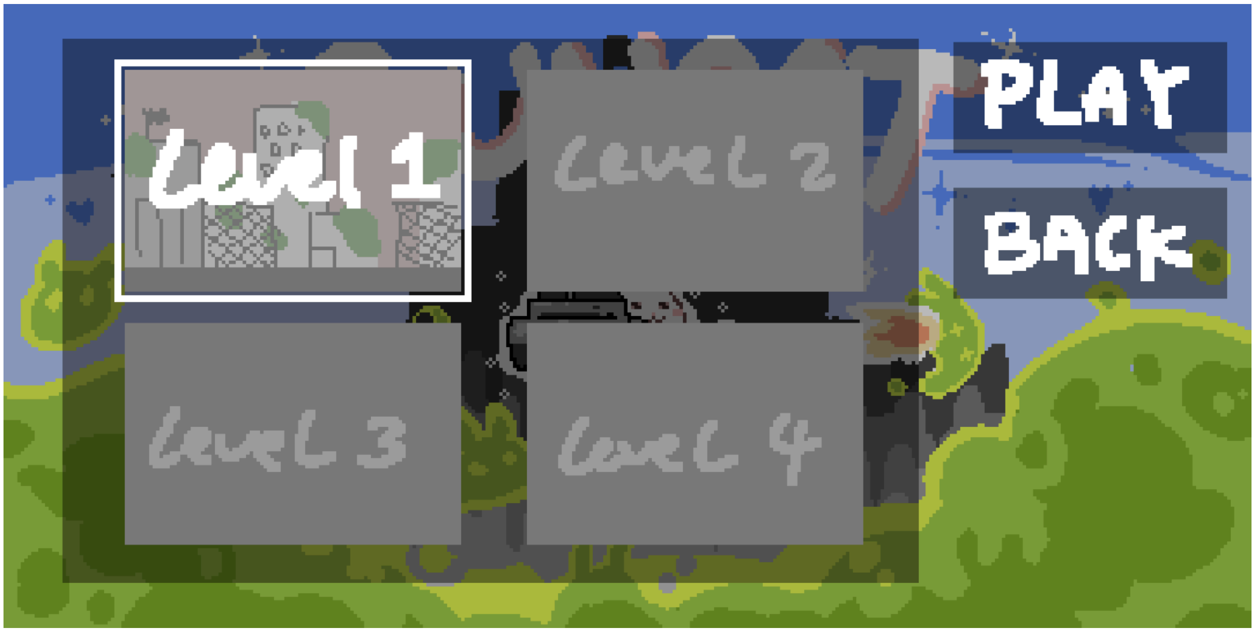
While we'd love to include multiple levels, we'll start with the initial city centre level and introduce new ones if we have the time. Collectables are also an optional feature and we'll focus on making one gun that feels good and is effective before introducing more weapons.

Scenes:

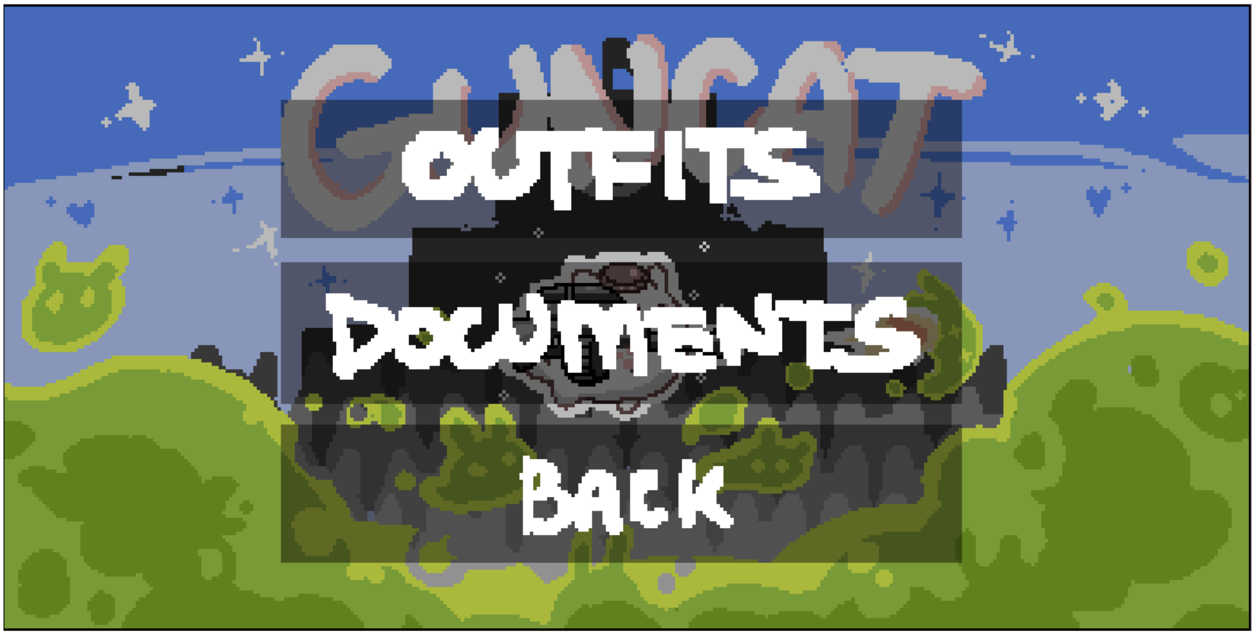
Main menu:

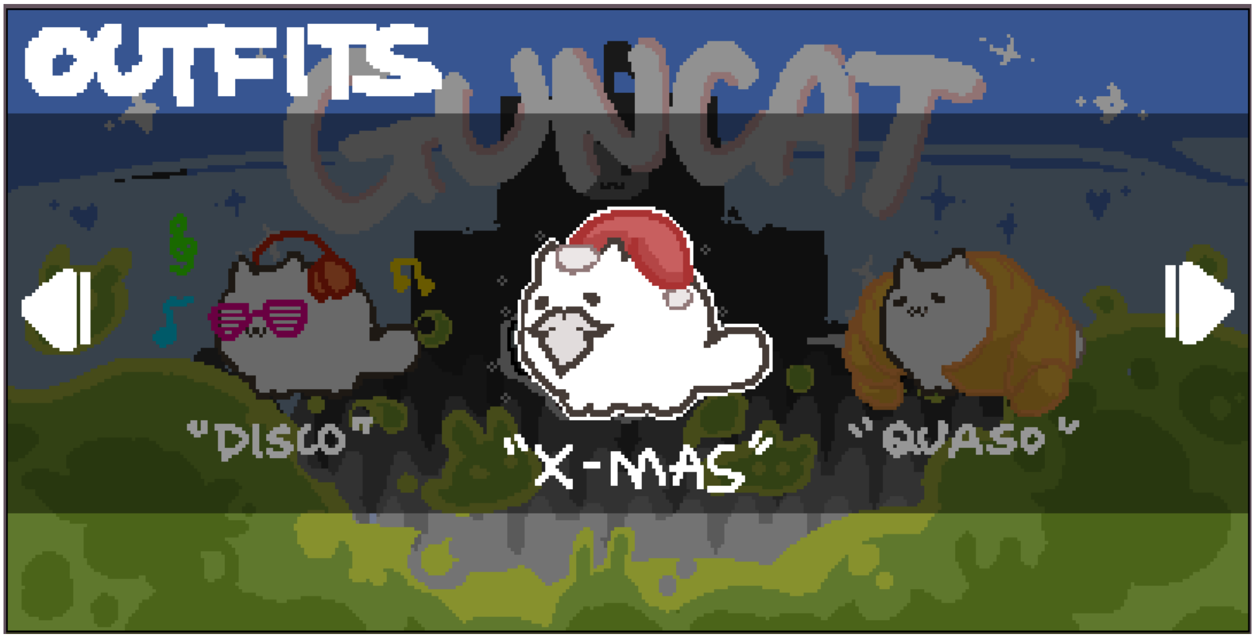


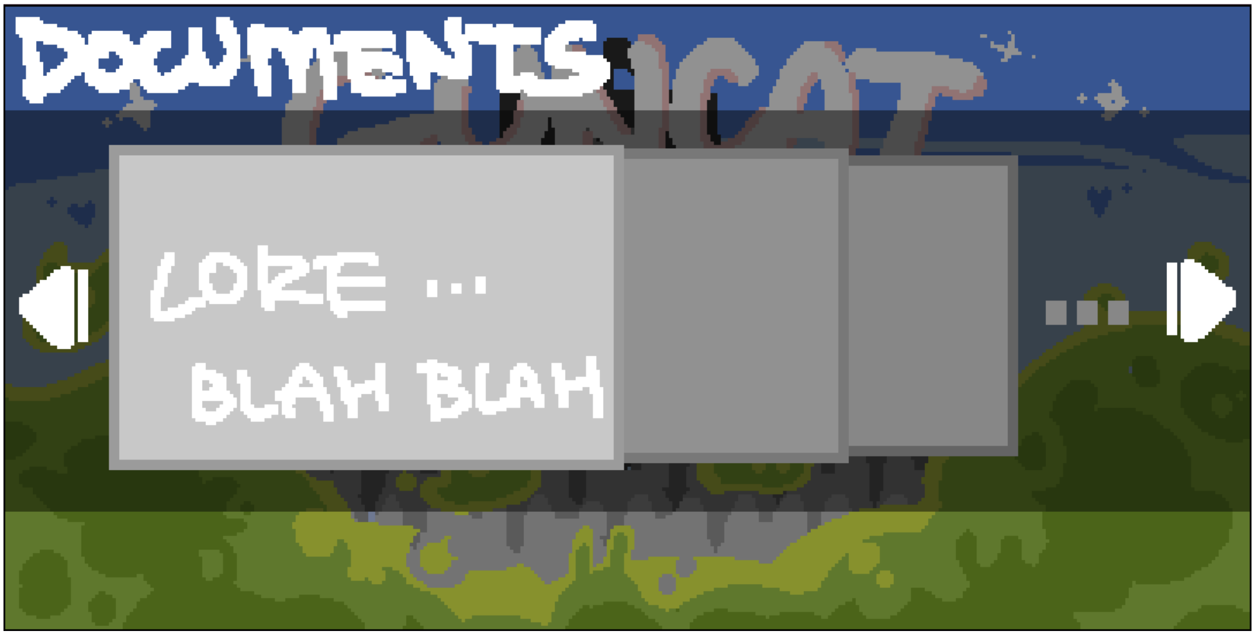
Click on 'Start' will take the player to the 'Level' Select screen



The Settings button leads to 'Outfits' and 'Documents' where the player can select outfits and review discovered codex logs. These are collected during gameplay as the player discovers secret locations where the items are hidden.



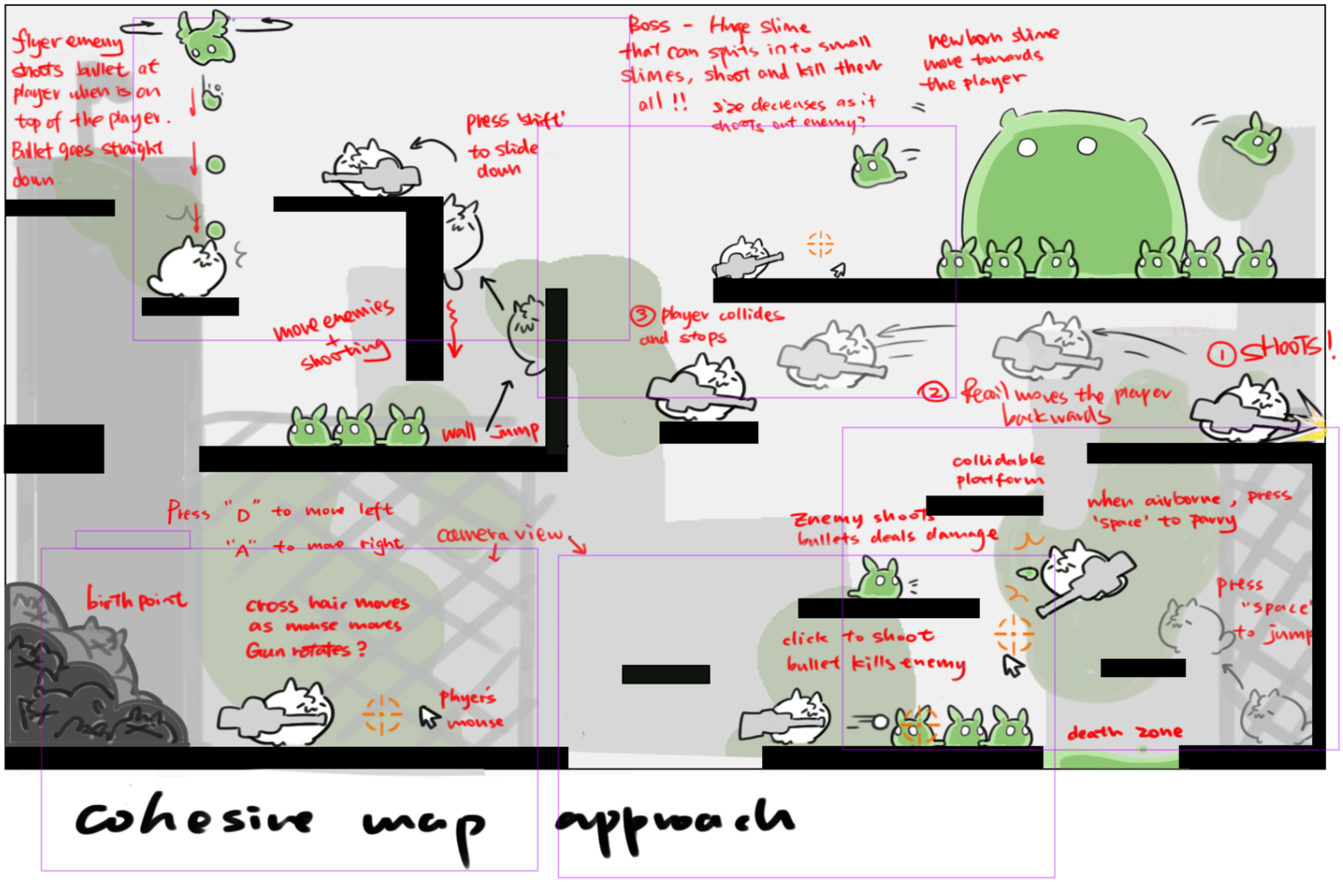




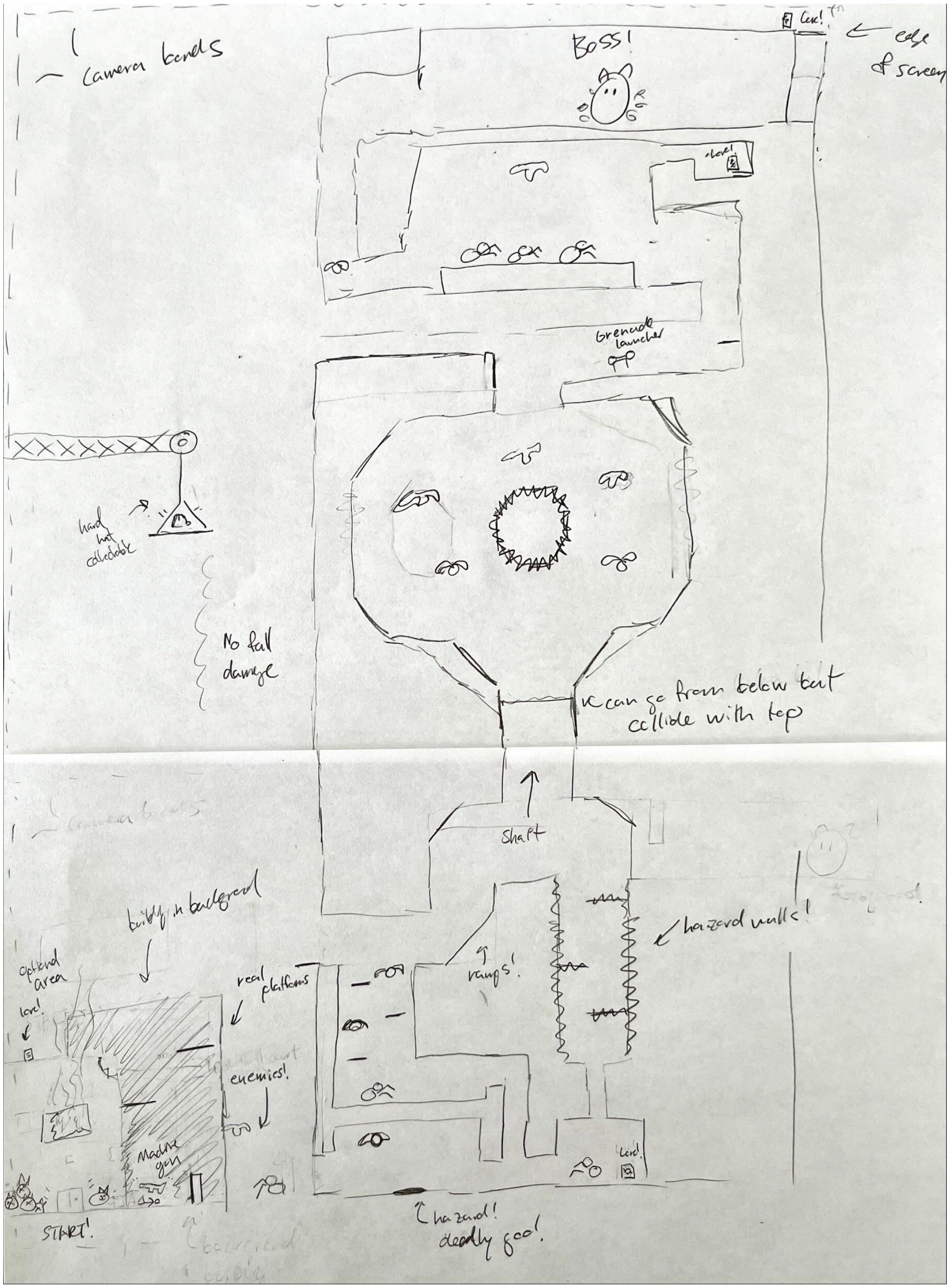
Cohesive Map Approach:

Camera always focuses on the player as they move throughout a large map, maintaining focus on the player's position (near the center of the screen) while rendering the corresponding section of the map. (Purple rectangle indicates the camera view)

- Map layout: The map is designed as a single, cohesive space. The entire map exists within one continuous level.
- Cam Transition: The camera stays centered on the player at all times, no loading or transitions when the player moves from one part of the map to another.


flyer emeny shoots bullet at player when is on top of the player. Bullet goes straight down
Boss - Huge slime that can spits into small slimes, shoot and kill them all !!
size decreases as it shoots out enemy?
newborn slime move towards the player
press 'shift' to slide down
more enemies + shooting
wall jump
③ player collides and stops
② recoil moves the player backwards
① SHOOTS!
collidable platform
when airborne, press 'space' to parry
Enemy shoots bullets deals damage
Press "D" to move left
"A" to move right
camera view
birth point
cross hair moves as mouse moves
Gun rotates?
player's mouse
click to shoot
bullet kills enemy
press "space" to jump
death zone
cohesive map approach

Disclaimer: above is a sample scene, not the actual level design for the game play.



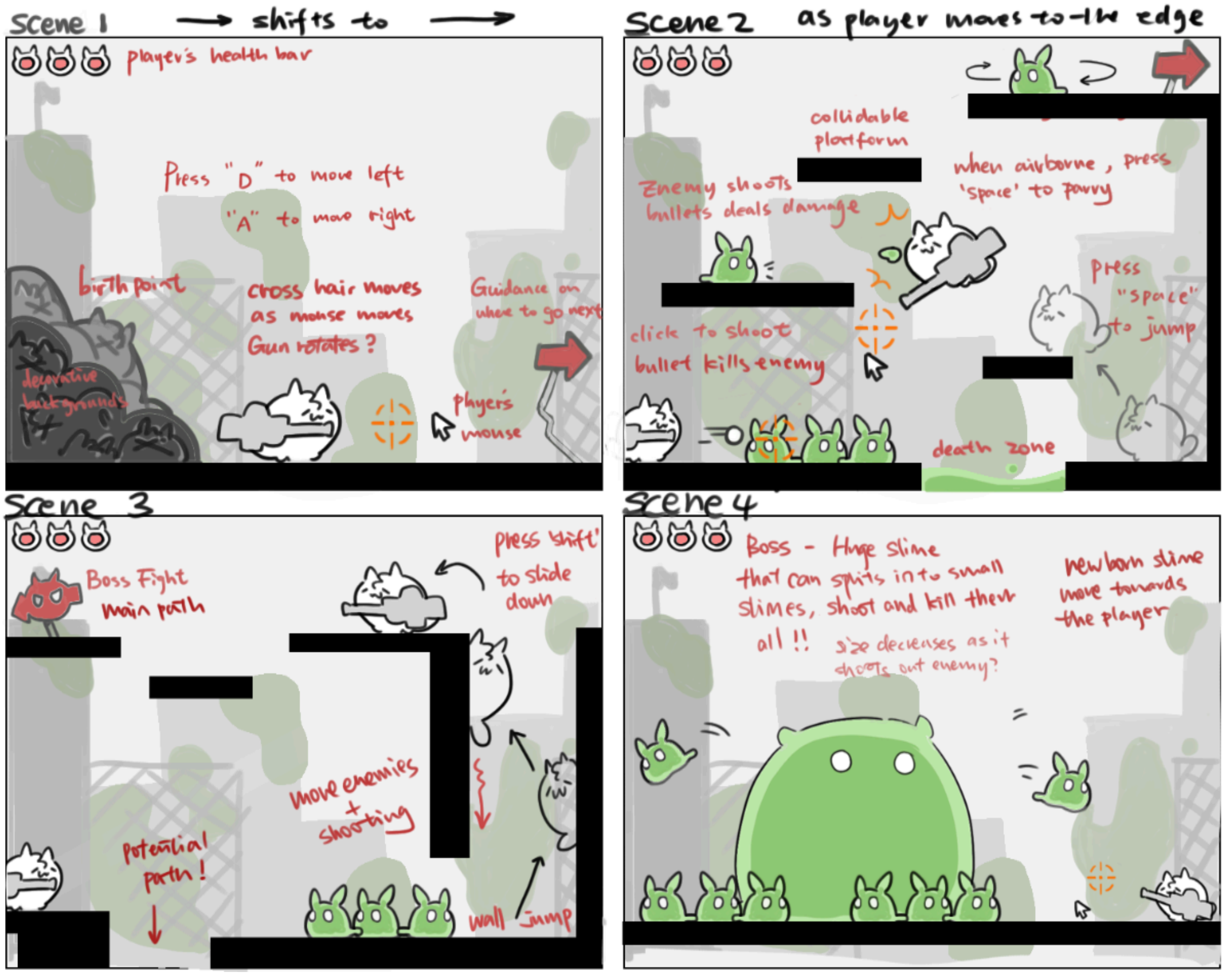
Here is a rough draft of the first level. It will take the player up a skyscraper without loading screens. At the top they will fight a boss.

Single Area Approach (Plan B):

If we run into issues with the cohesive map approach, whether due to tracking enemy state or with the camera following the player, an alternative would be splitting the map up into individual areas that are connected to each other and don't have a dynamic camera.

The entire map is divided into 4(could be less or more) areas. The camera will remain fixed on a single area at a time and will transition between areas when the player reaches the edge.

- Map layout: Camera starts in the bottom-left area,displaying that specific portion of the map.
- Cam Transition: As the player moves to the edge of the current area(e.g. touching the right or top boundary) the camera will shift to the corresponding area



Disclaimer: above is a sample scene, not the actual level design for the game play.

Technical Elements:

Rendering:

- **Camera focuses on the player always**. Except for special cases when player is at the corner of the canvas, the camera will a) continue to focus on the player, and showing the area that is outside the map; b) clip at the edge of the map's edge, and the camera will not be focused on the player as the player moves in that specific area.
- Parallax scrolling backgrounds.

Shaders:

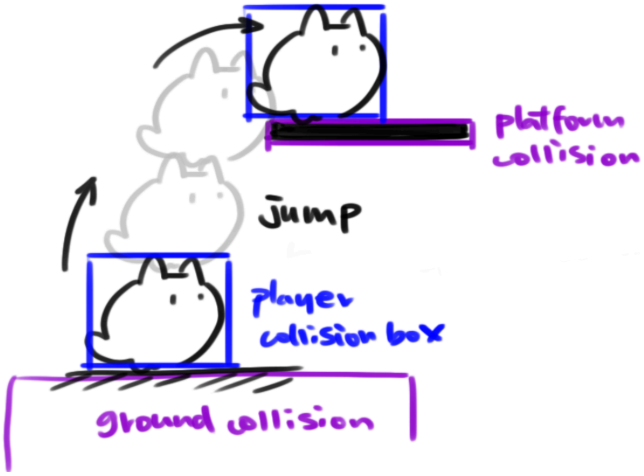
- Vertex Shaders
- Fragment Shaders
- Geometry Shaders

Geometric/sprite/other assets:

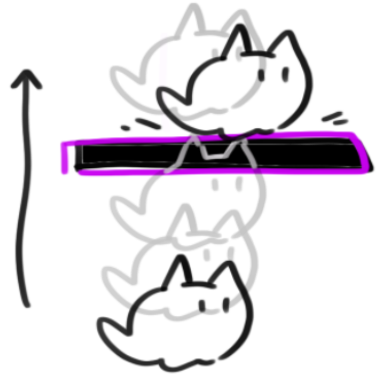
- Floors, walls, bullets and other collidable platforms can have simple geometrics.
- The player, enemies, and weapons can have sprites.
- OpenGL rendering Sprites: LearnOpenGL - Rendering Sprites
- Aseprite or CC0 sprite sheets as assets.

2D geometry manipulation:

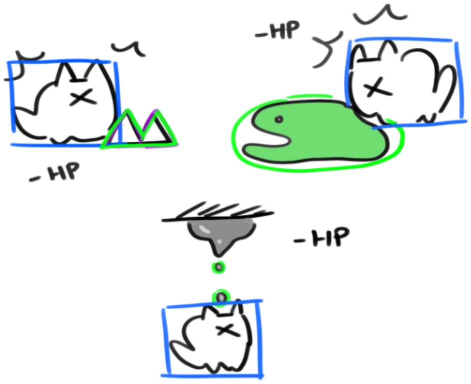
- Player Transformation:
    - The player movements and jumps involve vertical, horizontal or diagonal translations of the player's position.
- Player Collision:
    - PE collides platforms, walls, and floors. Collision is detected when hitting an enemy but the player only takes damage and is slowed, but can still move through the enemy. When jumping or firing the gun, we need to detect and resolve collisions between characters and collidable objects to properly launch, slow down, speedup or stop.

- 

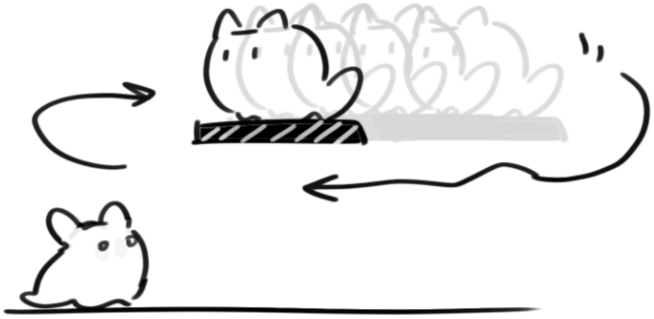
- Some platforms will be able to be passed through via the bottom but collide from the top.

- 

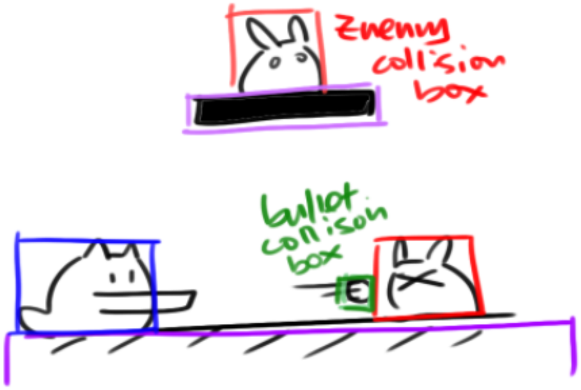
- There will also be hazards such as hot metal, slime pits, and spikes that damage the player and knock them back when collided with. They will be static.

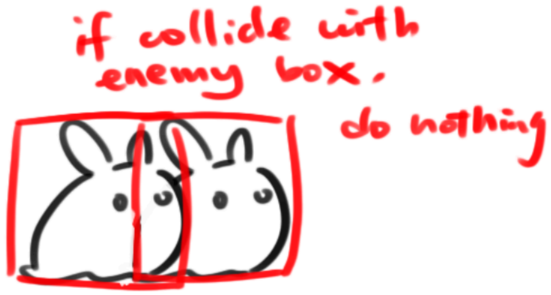
- 

- Moving platforms will be treated as colliding entities that have a set speed but are not affected by gravity. They will move back and forth between two set coordinates and are impossible to destroy.



- Enemy Collision:
    - Enemies must collide with the player, platform and player bullets, it’s not necessary to collide in between enemy entities. Player takes damage upon colliding with enemies, and the enemy takes damage upon colliding with the player's bullet.



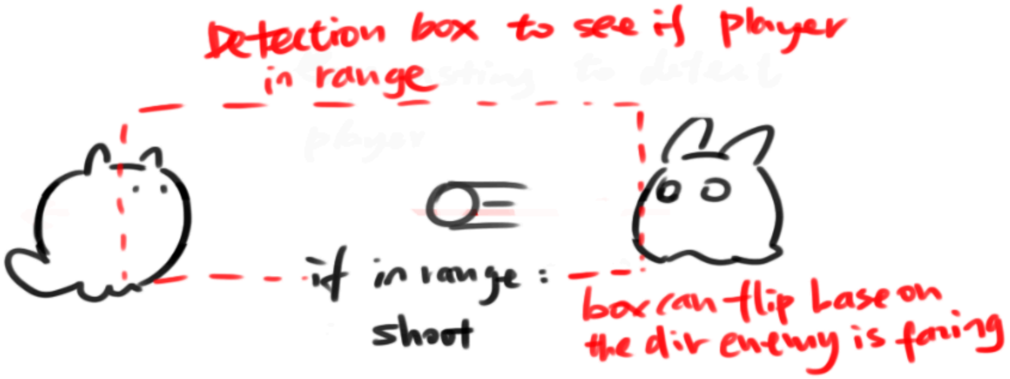


        -

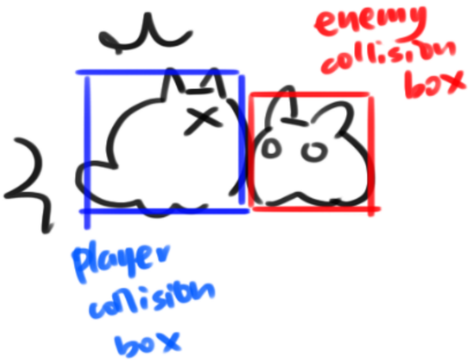
- Player Bullet Collision:
    - Bullets fired by the player will collide with any collidable objects in the environment. Bullet could deflect, cause damage, or just be destroyed if not colliding with anything after 5 seconds(it will eventually fly out of the camera view and disappear).

AI - enemy entity behavior:

- Implementing enemy behaviors: state machine, goal-oriented ai
    - Player proximity detection system to trigger changing states / actions.
        - Advanced feature: ray casting to implement Field of Vision detection in enemy entities instead of just player distance measurement
    - attacks types: collisions, spitting slime balls


Detection box to see if player in range
if in range : shoot
box can flip base on the dir enemy is facing

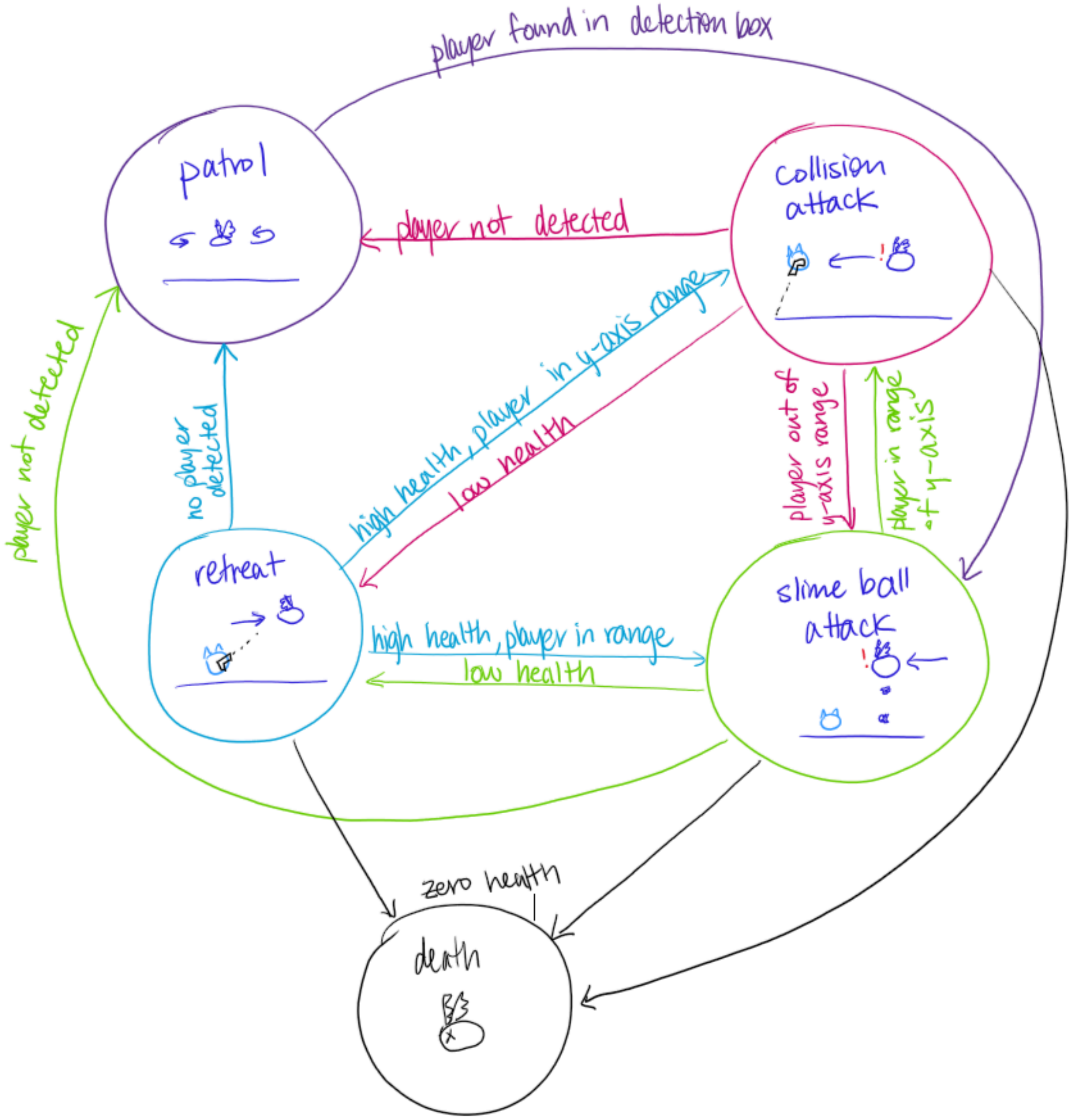
        -

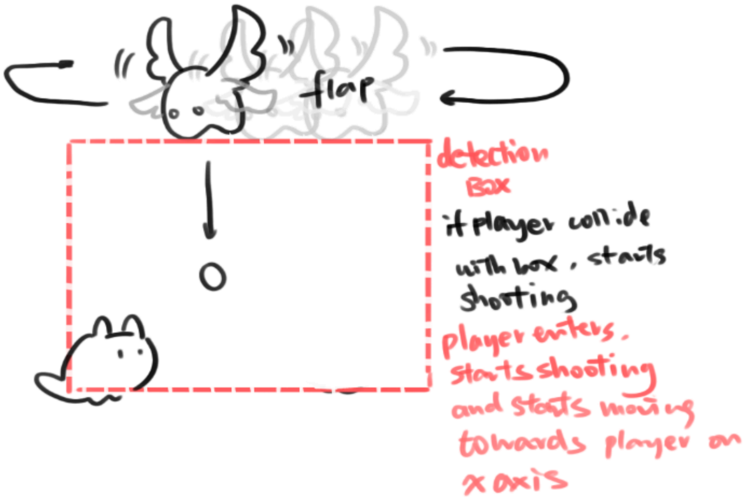


-

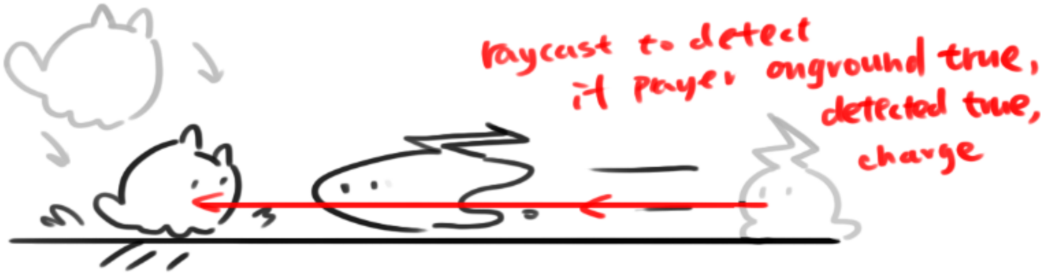
- Enemy types: static number per level/map (except at boss scene)
    - Flyer enemies:
        - Slowly flying around in bounded area creating obstacles/collisions and attacking player with slow moving slime balls.
        - Forces player to master dodging and parrying controls.
        - Player detection to slowly fly towards the player (on fixed y-axis); otherwise random movements.
        - Finite state machine:
            - Patrol: random movements flying around patrol-space
                - conditions: no player detected in patrol-space
            - Slime ball attack: spits slime balls at player (balls shoot down x-axis); random flying movement outside of collision range
                - condition: player detected in patrol-space, but too far from collision range
            - Collision attack: quickly approaches and collides with player
                - conditions: enemy health is high, player is within collision range (within the y-axis range)
            - Retreat: moves farthest away from last location damage was taken and from player, slowly regains health
                - conditions: enemy health is low

Flyer Enemy Behavior: finite state machine

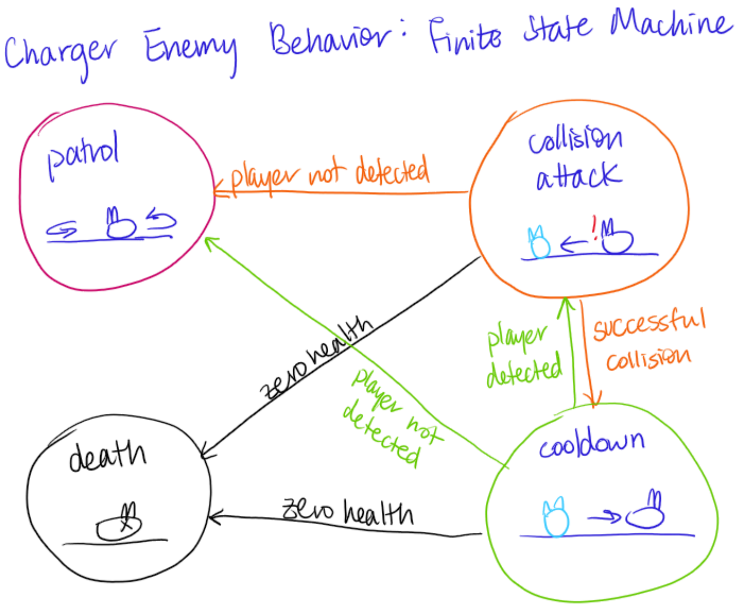

player found in detection box
patrol
collision attack
player not detected
high health, player in y-axis range
low health
player out of y-axis range
player in range of y-axis
no player detected
player not detected
retreat
slime ball attack
high health, player in range
low health
zero health
death


flap
detection Box
if player collide with box, starts shooting
player enters, starts shooting and starts moving towards player on x axis

- Charger enemies (if time permits!):
    - Charges towards the player if they touch the ground where the enemy entity is patrolling.
    - Forces player to stay in the air using gun; discourages simple walking.
    - Damages player health with collisions.
    - Fast, armored with strong health, bullets do very little damage, except weak point on top.
    - finite state machine:
        - patrol: periodic movement back and forth in patrol space
            - conditions: no player detected in patrol space
        - collision attack: charges towards player
            - conditions: player detected in patrol space
        - cooldown: moves away from player for set cooldown time
            - conditions: successfully collided with player

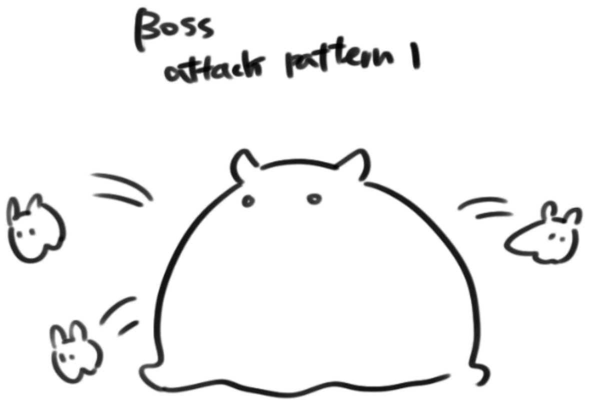


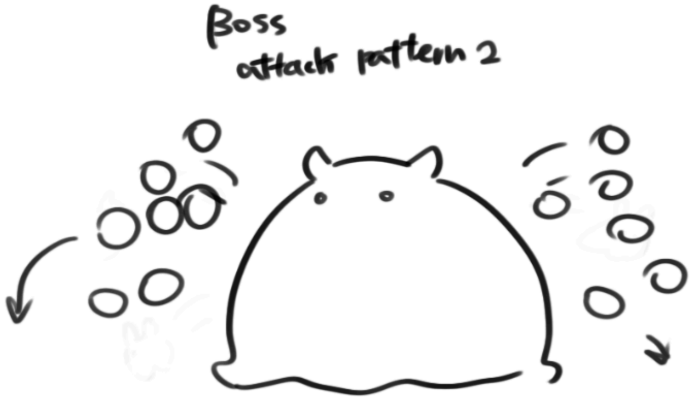
    -

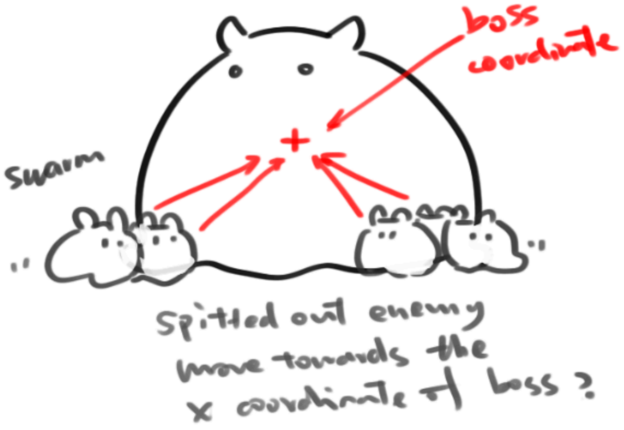


-

- Boss enemy:
    - Fire slow moving slime balls as well as huge gouts of slime.
    - Large-scale flyer or charger slime that also spawns slimes.
        - Spawning slimes causes enemy swarming around boss.
    - Goal-oriented AI: GOAP or BDI (plan B is a decision tree)
        - actions:
            - spawn slimes for attack or defense
            - spew gouts of slimeballs at player
            - move away from player
            - approach and collide with player
        - goals:
            - defense when health levels are low
            - attack player



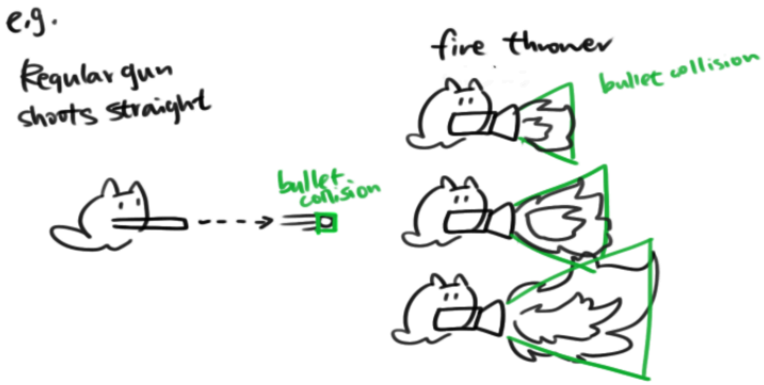




Physics:

- Physics will be handled by manipulation of position and speed components for different entities.

- Gravity will be a component applied to certain entities, things that have a gravity component will be constantly decrementing their speed component unless they're at an arbitrary "terminal falling speed"
- Bullet projectiles will usually just travel in a straight line, but other weapons could have their own physics and interact with yours in different ways.



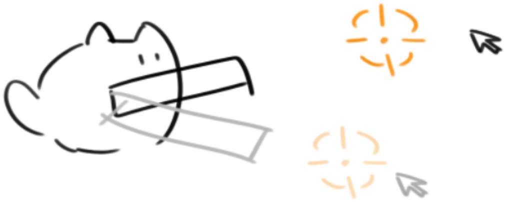
-

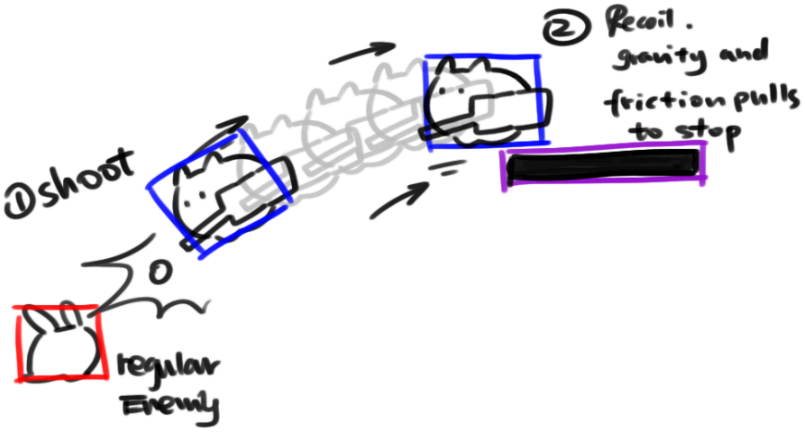
Weapon/bullets:

- Our starting weapon will be a fully automatic assault rifle.
    - Ammunition is infinite
    - Magazine capacity will be limited
    - Rate of fire will be around ~300 rpm
    - Rifle ammo
        - Bullets will travel straight and without any bullet drop off (see physics)
        - Bullets will continue straight until colliding with an object or until a certain time has passed (see 2D geometry manipulation)

Advanced Technical Elements:

- Gun aiming
    - The gun will move along according to the cursor **(Plan B: we can simply have the cursor act as the crosshairs without needing to render a separate UI element. Wouldn't look as clean though.)**

- 

- Shooting a gun applies recoil in the opposite vector of where the mouse is aiming. Each weapon will have a set "recoil magnitude" in its weapon component that will be used to calculate the vector components that are then applied to the player entity.
    - When the player fires their weapon, the coordinates of the player entity and the crosshair will be received. Once received, the game will calculate the relative vector between those two points. Once received, we will convert it into a unit vector (so that the distance of the crosshair does not produce more recoil) and multiply it by the weapon's "recoil magnitude". We will then reverse these vector components and simply add them to the player's speed component, launching them around. **(this feature is core to the gameplay and can not be replaced)**
    - 

- Particle systems. Can be used on enemies, or smoke effect when player shoots **(Plan B: The game will operate as normal without this feature, but the extra feedback on firing a gun will add a strong aesthetic component.)**

- AI - boss enemy behavior: goal-oriented behavior using Goal-Oriented Action Planning GOAP or Belief-Desire-Intention (BDI) **(Plan B: Using a decision tree where enemy behavior is solely based on hard-coded conditionals in the tree, which is more predictable but easier to debug.)**

- AI - enemy entity behavior: swarming behavior (apply boids behavior to flyer enemy types at boss scene) **(Plan B: We can instead just spawn normal flyer enemies that don't engage in swarming behavior. The fight will be less interesting however.)**
- AI - enemy entity behavior: Field of View detection of the player using raycast **(Plan B: We can instead have enemies locked to a random patrol route with player location detection regardless of view-blocking entities, however, reacting to the player will make them more interesting and dangerous. Feasiblity depends on how consuming it is to track all enemy entity states in game.)**
- Secondary weapon: Grenade launcher that player **(Plan B: We can keep the basic machine gun, but being able to swap to a second weapon with a different traversal tool could drastically expand the complexity of the gameplay. i.e. choices per second for the player)**
    - Player can fire a grenade that has a timer, and then jump in the vicinity of the grenade to get launched in the opposite direction based on where they are located relative to the grenade.
    - Magazine capacity would be much smaller compared to the assault rifle.
    - Physics for this will be much more advanced compared to the assault rifle.

Devices:

The two input devices we plan on supporting are keyboard and mouse.

Keyboard:

‘A’ key will be used to walk left.

‘D’ key will be used to walk right.

Space bar will be used to jump.

‘R’ key will be used for reloading

Swap weapons 'E'

‘Space bar’ while already in the air for parrying

- Successfully parrying a weapon or enemy in the air should automatically reload player's gun.

Mouse/trackpad:

The mouse will be used to aim the weapon in the direction you intend to fire.

Left click will be used to fire the weapon.

Tools:

YouTube

Google

Stack Overflow

Physics Library (TBD)

Trello for managing tasks

Github for version control

Aseprite for asset creations

Team management:

We will list tasks within GitHub issues as well as a Trello kanban board. Team members will take tasks and implement them. We are roughly assigned to a general area based on personal interests but we can ask for help and help other people's areas if there is a difficult task or bottleneck.

Each member will be responsible for getting their area done. If it's too much work, they should ask for help. If someone doesn't have much work or it's easy, they should offer to help other people. Try to start work as early as possible so that you can give a good estimate for how long it will take to complete!

Assigned areas/roles:

Anthony Hayek - Guns and Bullets, Weapon Design

Ayden Kinchla - Level Design / Rendering

Diana Dou - UI/Rendering + Assets

Kurtis Ho - Gameplay systems (character movement)

Raizen Banzon - Physics

Sally An - AI and enemies

Development Plan:

Milestone 1: Skeletal Game (due Oct 6)

(Since key elements of the game need to be implemented first, members will be working outside of their assigned roles to get the basic functionalities implemented. Some members will be working in pairs due to the complexity of the task and their technical strengths.)

Week 1 (tasks completed by Oct 2)

Character Movement:

- A & D move player entity (PE) left and right respectively - **Kurtis**
- Space causes PE to jump up in the air (add y speed component, preserves x movement momentum) - **Kurtis/Anthony**
- Character falls down due to gravity - **Raizen**

Enemies / AI:

- Flyer enemy entity: able to move in space - **Sally**
    - player detection
    - entity stays within bounded patrol area
    - For this milestone, flyer entity will have random flight patterns
- When player touches enemy, player dies - **Sally**
- Player collides with floor and walls - **Kurtis/Anthony**

Rendering:

- Render Player entity- **Diana/Ayden**
- Create and render wall entities that will bound the playable area and be textured - **Diana/Ayden**

Week 2 (tasks completed by Oct 6)

Physics:

    - Shooting a gun applies recoil in the opposite vector of where the mouse is aiming, add set magnitude stored in each weapon component to x and y speed components depending on vector created from player entity to crosshairs - **Raizen/Kurtis**

Weapons:

    - Bullets travel in a straight line until they hit a wall, enemy, or persist for a certain amount of seconds (bullet component, contains speed, damage, and lifetime) (some guns shoot things that aren't bullets, can be affected by gravity, flamethrowers, etc.) - **Raizen/Anthony**
    - Implement aiming at the mouse, get relative vector for physics system based on Player Entity position and crosshair position, multiply unit vector by recoil magnitude and reverse (see physics) **- Anthony/Sally**
    - When bullet touches enemy, they die **- Sally**

Graphics / Rendering:

- Camera follows PE as they move through level - **Diana**/**Ayden**
    - PE is always centered in screen unless reaching end of map (camera stops at end of map, but PE can reach wall)
- Crosshair entity tracks and moves with mouse - **Diana**/**Ayden**
    - Crosshair maintains a set distance from player entity when entity moves.

Milestone 2: Minimal Playability

Week 1

Graphics / Rendering:

- Create sprites
- Level Design
- Design & implement basic level

Enemies / AI:

- enemy entity shoots slime blobs (similar mechanics as bullets, but no recoil)
- implement charger enemy entity (enemy is ultimately optional)
    - enemy quickly moves toward player, only moving on x-axis
    - builds on player detection system
- implement flyer enemy behavior finite state machine

UI:

- main menu screen
- UI displays player health
- UI displays current gun ammo count

Week 2

Character movement:

- wall jumping (if colliding with wall and pressing "a" or "d" into it, slow fall speed. when pressing jump, leap off it at a 45 degree angle).

Physics:

- friction when touching floor (if colliding with floor, if x speed vector is +, subtract until 0. if x speed vector is - (going backwards) add until 0)

Weapons:

- weapons have max clip value and current clip value, counts down when fired
- if clip value is set to 0 after firing, automatically reload
- reload time is in weapon component, after set time in seconds set current clip to max clip (decreasing value applies to this value every screen, can make it bigger or slower depending on power ups, when it reaches 0 reload gun and set reload time back to max) the reload timer will reset if you swap weapons mid reload.
- bullets damage enemy entities they collide with

Graphics / Rendering:

- Animate sprites

Enemies / AI:

- invincibility frames when player touching enemy / slime ball
- add health component to entity and player (increment and decrement health according to type of damage received)
    - make sure enemy or player is able to die at zero health

Milestone 3: Playability

Week 1

Graphics / Rendering:

- plan A: parallax background
    - plan B: static background

Level design:

- hazards, Some ideas: slime pits, fire, laser pointers, fire hydrant spraying water, spikes, etc.…
- Place enemies in appropriate locations

UI:

- pause menu (go back to main menu / level select, also retry level)
- UI displays reloading time

Enemies / AI:

- Implement boss enemy entity: large-scale enemy entity
    - high health levels
    - high collision damage on player
    - player detection used to track, charge, and spit at player
    - goal oriented AI (plan B is decision tree) - begin with only attack goal

Week 2

Character movement:

- Parry, start counting down invulnerability timer (in frames?) if player entity collides with bullet entity and invulnerable is true, take no damage and flip all speed components of bullet.

Physics:

- Bouncing, wall entities have bounce component? (on collision, systems check this bounce variable and uses it as proportion of speed vector to return to colliding entity in transformed direction (the bounce)).
- Entities that can bounce will have a bounceable/movable component so that not all entities will bounce on a wall.

Enemies / AI:

- Implement boss enemy defensive behavior: at certain health level threshold, boss begins to focus on slime spawning for defense, and moves away from player.

Level design:

- Have a single level with two different routes to fight the boss.

Milestone 4: Final Game

Week 1

- credits
- lore section in pause menu
- level select screen
- enemy swarm behavior (at boss scene)
- select outfit menu from main menu

Week 2

- add collectable outfits, swap out PE with different one, can select from main menu
- add collectable lore with lore page you can read through
- make a second level?
- add multiple guns with different components and bullet behavior, some might create different entities that do dif things (e.g grenade launcher)