Game Proposal: Bermuda

CPSC 427 – Video Game Programming

Team: Tony and the ABCDs

Tony Zhang - 42560326

Andy Li - 65134645

Bob Pham - 44606424

Brandon Yih - 43117480

Caeleb Koharjo - 18539551

David Sopheap - 24296634

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.

x = Some Unknown Number To Be Determined

Background Story:

An expedition team never returned from exploring an unknown area at the bottom of the ocean. You are dispatched to go investigate. While exploring in your submarine, you're attacked by a mysterious creature and the submarine is torn apart. You find yourself surrounded by the debris of the expedition team's submarines and equipment trailing off towards a cave-like entrance
All you have left is an oxygen tank, an atmospheric diving suit, and a harpoon gun to explore through these caves and figure out what happened.

Loading Into The Game:

The start of the game will display a comic-like frame (or frames) depicting the background story mentioned above. In the frame, the mysterious creature will appear as a silhouette. The screen will then go black and an oxygen tank will appear on screen. The oxygen tank will fill up to act as a "game loading" bar (See Figure 2 in Scenes). The screen will then change to place the player in the first room of the game.

The Player and HUD:

Any input controls mentioned here will be summarized in the Devices section.

The player will be dressed in a diving suit and will be equipped with the following:

- Personal Oxygen Tank (See Figures 8, 31 in Scenes)
  - Equipped to the back of the player
  - Oxygen tank depletes slowly over time
  - If oxygen level reaches 0, the player will die
  - Oxygen will be the basis of most mechanics in the game
- Harpoon Gun (See Figure 12 in Scenes)
  - Held by the player
  - Aimed using mouse cursor (will have a max distance range from player)
  - Fired using mouse click
  - Deals damage and returns to player by rope on collision or reaching target
  - Firing the harpoon will use a fixed low amount of oxygen

The player will move using the following:

- W A S D Keys (See Figure 9 in Scenes)
  - Used to move around
  - Does not use any extra oxygen
- Shift Key (See Figure 13 in Scenes)
  - Holding shift while moving will allow the player “Dash” (move at x% velocity)
  - Will use oxygen by increasing oxygen depletion rate by x% while shift is held

The player HUD will display the following (See Figure 3 in Scenes):

- Oxygen tank meter will be displayed on the left side
  - Meter represents oxygen remaining
  - Low oxygen will change the colour of the meter and change the music
- Pause Button
  - The pause button will use a fixed low amount of oxygen
  - The player will see a message warning them that pausing just used oxygen
  - Buttons to resume or exit the game

- Net Inventory
  - Displays number of nets in inventory
  - Press 1 Key to load net to fire on next mouse click
  - Press 1 Key again to unload net before firing
  - Stuns an enemy if hit (See Figure 16 in Scenes)
  - Firing consumes one net and uses a fixed low amount of oxygen
- Torpedo Inventory
  - Displays number of torpedoes in inventory
  - Press 2 Key to load torpedo to fire on next mouse click
  - Press 2 Key again to unload torpedo before firing
  - Does circular area damage at target location (See Figure 18 in Scenes)
  - Firing consumes one torpedo and uses a fixed low amount of oxygen
- Concussive Inventory
  - Displays number of concussive bombs in inventory
  - Press 3 Key to load concussive to fire on next mouse click
  - Press 3 Key again to unload net before firing
  - Pushes enemies in a cone backwards a short distance (See Figure 17 in Scenes)
  - Firing consumes one concussive and uses a fixed low amount of oxygen
- Pistol Shrimp Inventory
  - Displays number of pistol shrimp charges in inventory
  - Press 4 Key to load pistol shrimp charge to fire on next mouse click
  - Press 4 Key again to unload net before firing
  - Kill enemies / high damage to (mini) bosses in a cone (See Figure 19 in Scenes)
  - Firing consumes one pistol shrimp and uses a fixed high amount of oxygen

Map Generation:

Upon the start of each playthrough, a map is created. Each map consists of 15 rooms, each connected via hallways between them (See Figure 4 in Scenes). Different sections of the map will bring its own distinct atmosphere, introducing different types of objectives or enemies that prevent the player from unlocking the next room (See Figure 5 in Scenes). Note that this figure only shows a high-level map for design purposes; in the real game, the player will not have access to any sort of map.

Note all algorithms mentioned are tentative for now.

- ➢ Algorithm: the map may initially begin as a square of non-navigable ‘wall’ tiles, with each tile reflecting a certain pixel distance to be rendered. Rooms and hallways can then be generated as a random graph of nodes and vertices, with each room and hallway ‘hollowing-out,’ i.e removing the wall tiles of the initial square with a combined random graph generator and random room generator, mentioned in Room Generation.

Room Generation:

Each room that the player goes through will either have a completely fixed design or have aspects that are randomly generated (See Figure 6, 7 in Scenes).

The following is a list of aspects that might be randomized in a given room:

- ❖ The physical traversable space of the room (the shape of the room)
  - ➢ Algorithm: Each room maps onto a node generated by the algorithm that generates the randomized map. In the simplest form, a room’s boundaries can then be generated by generating a navigable square with *x* tile sides. For non-square rooms, we might further randomize the appearance of the room by generating more random, fully non-navigable squares that touch the sides of the overall generated square recursively to effectively add more walls to the initial square, which makes the room more natural and diverse.
- ❖ Enemies
  - ➢ Spawn point location (if any)
  - ➢ Number of enemies
  - ➢ Types of enemies
  - ➢ Algorithm: A number, x, is chosen in a range depending on the room’s difficulty, and then we randomly, without bias, sample x tiles from all the tiles in the room, and place 1 enemy on each tile we sample. Sampling bias may be introduced for the types of enemies, with later rooms having more dangerous foes, or even a predetermined mixture of foes.
- ❖ Hallways connecting rooms
  - ➢ Locked by objective
  - ➢ Algorithm: Each hallway maps onto a vertex generated by the algorithm that generates the randomized overall map. These can be generated as a random

m*n tile rectangle, with the same detailing algorithm we used for room shape (or a tweaked version, or none), if we want.

- ❖ Oxygen Tanks (Map Object - Not to be Confused with Personal Oxygen Tank)
  - ➢ Location
  - ➢ Algorithm: These could be generated identically to the algorithm described in Enemies, or, if we want a harder game, could be sampled from all navigable tiles on the map instead of on a per-room basis.
- ❖ Breakable Map Objects
  - ➢ Location
  - ➢ Size
  - ➢ Breakable Crates - chances to contain each consumable (Ex. net)
  - ➢ Algorithm: See the algorithm mentioned for Oxygen Tanks.
- ❖ Objective Related Assets (Ex. Keys to Open Doors)
  - ➢ Location
  - ➢ Algorithm: For the first few milestones, we will keep it simple and make it so that all objectives open doors for the room they are in; for future milestones, we could look into, for example, placing objectives to different doors in different rooms randomly, and then using a graph-traversal algorithm to ensure that the map is beatable.

    For example, we could generate a random graph, with random rooms (nodes), doors (rooms), and objectives (specially designated nodes). We could then pair each door = {d_1, d_2, … d_n} and objective {k_1, k_2, k_3}, and traverse the graph using graph traversal.

    When we move to a new room, we add all k_n inside to our traversal's 'inventory'.

    When we move to a new vertex, v, we check if we have the key k_n that opens the door d_n at that vertex, v. If not, we come back later, if so, we 'open' it and keep going.

    At the end, if the algorithm marks every node, the map is beatable. If not, we throw it away and start over again.

- ❖ Geysers
  - ➢ Location
  - ➢ Algorithm: See the algorithm mentioned for Oxygen Tanks (but these will likely be very rare).

Fixed design rooms, on the other hand, are merely a select few rooms that we will design by hand; that is, the listed elements above are not randomized but purposefully placed by us, such as the room that contains the final boss.

Oxygen:

The oxygen mechanic is the main gimmick of our game. The player starts with a finite, but refillable amount, and when it runs out, they instantly die. Outside of basic movement, nearly every other interaction costs a fixed set of oxygen.

Costs are too prone to changes to be named specifically here and will be tweaked and refined as we playtest our game. Rough descriptions of costs for specific actions are named in their corresponding section under Player and HUD or Room Gameplay.

Oxygen depletes over time or decreases a fixed amount from enemy hits. Additionally, the following player actions cost oxygen:

1. Dashing - increasing movement speed (See Player and HUD, Figure 13)
2. Firing the harpoon gun or a consumable from the player inventory (See Player and HUD)
3. Pausing (See Player and HUD)
4. Opening doors that require oxygen to unlock (See Room Gameplay, Figure 24)
5. Igniting glass lamp box (See Room Gameplay, Figure 23)

The player can also refill oxygen at two places, whose placement details are in the above section on Room Generation and purpose details are in the Room Gameplay section below.

1. Oxygen Tank (Map Object) - refill oxygen a substantial amount (See Room Gameplay)
2. Geyser - infinitely refills your oxygen, but are therefore rarer (See Room Gameplay, Figure 10)

Room Gameplay:

The player will have to play through multiple rooms in order to reach The Final Boss Room. This game is designed to accommodate different playstyles, where deciding how to use oxygen in various ways influences the gameplay. The player may, for example, fire at an enemy in order to

more safely move through the room and access a required item that unlocks a connecting door elsewhere on the map. However, a player does not have to defeat enemies to progress, unless a miniboss is guarding a door. Therefore another way to get to the required item in this example is using the dashing mechanic to maneuver around enemies that are swarming an area (See Figure 28 in Scenes). This moment-to-moment choice between speed and safety, combat or maneuvering, is the reason managing the player's oxygen is so important. Another choice to be made is whether the entire map is worth exploring. There will be multiple paths that lead to The Final Boss Room, but whether the player decides they want to unlock all the doors and explore every room before moving ahead is a personal choice.

There will be multiple interactable parts to each room including:

- Breakable Map Objects
  - Breakable crates with chance for consumables inside (See Figure 20 in Scenes)
  - Breakable walls to hide behind or break to make a path (See Figure 21 in Scenes)
  - Oxygen tank to do one of the following: (See Figure 10 in Scenes)
    - Stand next to it to refill fixed substantial amount of oxygen
    - Shoot it if oxygen was not taken and cause damaging area explosion (See Figure 11)
- Unbreakable Map Objects
  - Geyser that refills oxygen indefinitely when standing next to it
  - Stone block can be pushed (See Figure 22 in Scenes)
  - Glass lamp box can be pushed and ignited by fixed low amount of oxygen (See Figure 23 in Scenes)
  - Pushable object can only move up, down, left, and right (See Figure 22 in Scenes)
- Enemies
  - Hinder progress (See Figures 37-39)
  - Players can shoot enemies and they die if their health reduces to 0. No visible bar will be shown to the player.
  - Enemies hitting the player will cause oxygen meter to decrease a fixed amount

- Doors
  - Will lead to another room
  - May be locked by an objective (See Figures 24-27 in Scenes)
    - Key(s) can be collected that open doors
    - Mini bosses may be required to kill to open doors
    - Pressure plate(s) may need to be interacted with to open doors, guarded by clusters of enemies (can be weighed down by pushable objects)
    - Some doors may require fixed oxygen costs to open

Note that the item a player interacts with to open a door may not necessarily correspond to the door(s) of the room that it was found in, although which door it opens will be visually broadcast to the player to avoid confusion.

The Final Boss Room Gameplay (See Figure 29 in Scenes*)*:

Once the player has chosen to enter the boss room, the screen fades to black. After this, a comic-like frame (or frames) depicting the player preparing themselves to go in, and how their adventure thus far has culminated to this. The player enters the room, and the silhouette from the opening scene appears, with the expedition team ensnared in the background. The silhouette has a brief dialogue taunting the player. The silhouette comes into colour to reveal the final boss and the battle begins.

The player must first engage with 3 waves of enemies of increasing difficulty, filled with enemies found throughout the rest of the game. The player then fights the final boss, who is the toughest enemy within the game. The boss summons void tentacles, shoots energy projectiles, and attacks you physically with its tentacles, as pictured.

After the boss is defeated, the game is completed.

Game Over:

If the player has no more oxygen remaining, the player will suffocate and pass out on the seafloor. The music will change and the screen will fade to black before presenting the Game Over screen (See Figure 35 in Scenes). It will be possible for the player to use up the rest of the remaining oxygen from their own action input so the player has to keep an eye on their oxygen level at all times.

Completing The Game:

Upon completing the game, the music will change and the screen will fade to black before presenting the Game Completed screen (See Figure 36 in Scenes).

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 gameplay elements you are planning to copy.

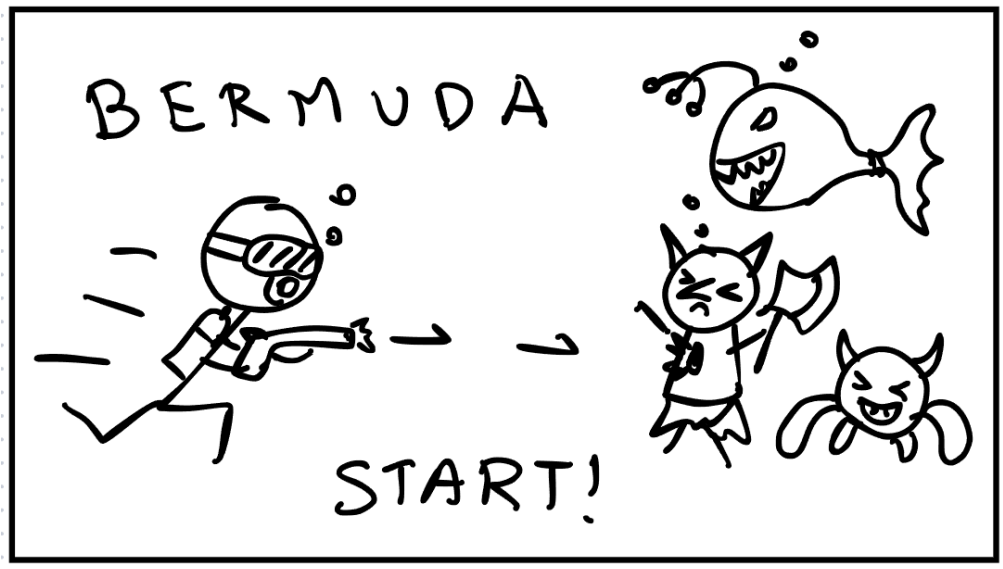


Figure 1. Starting screen for game

Figure 2. Oxygen tank filling up as a loading screen

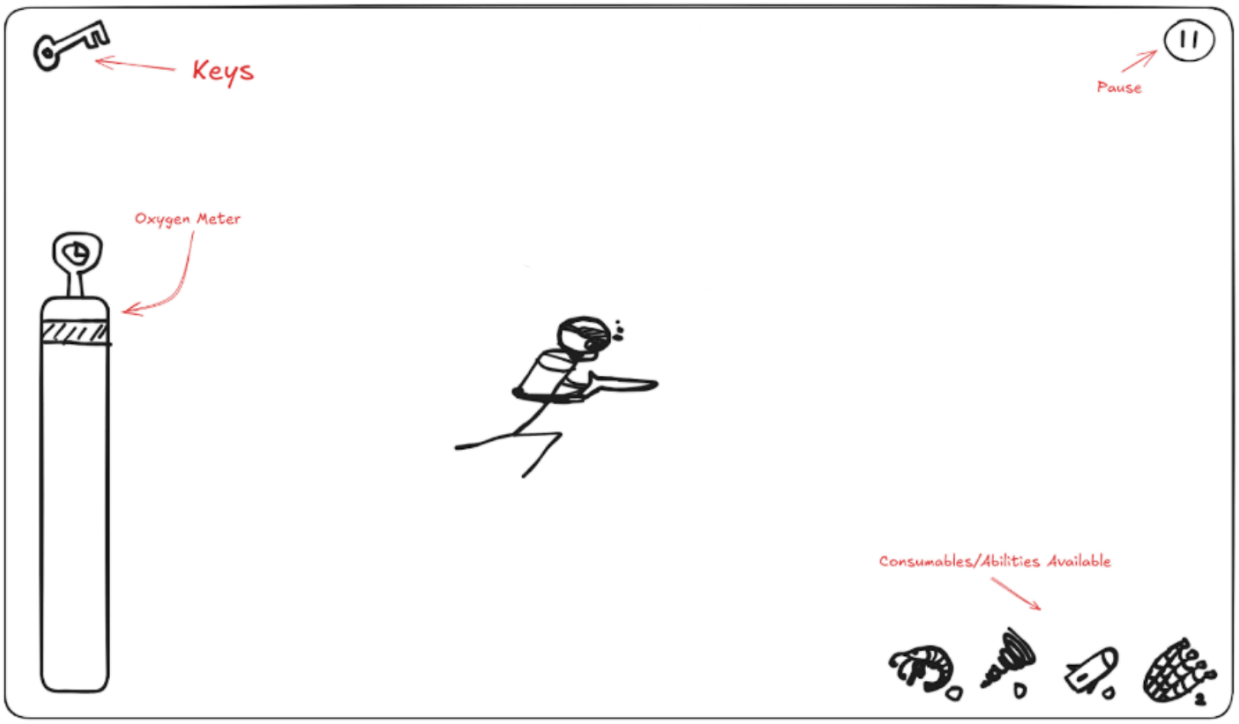


Figure 3. Player spawning into start of the game (with HUD)

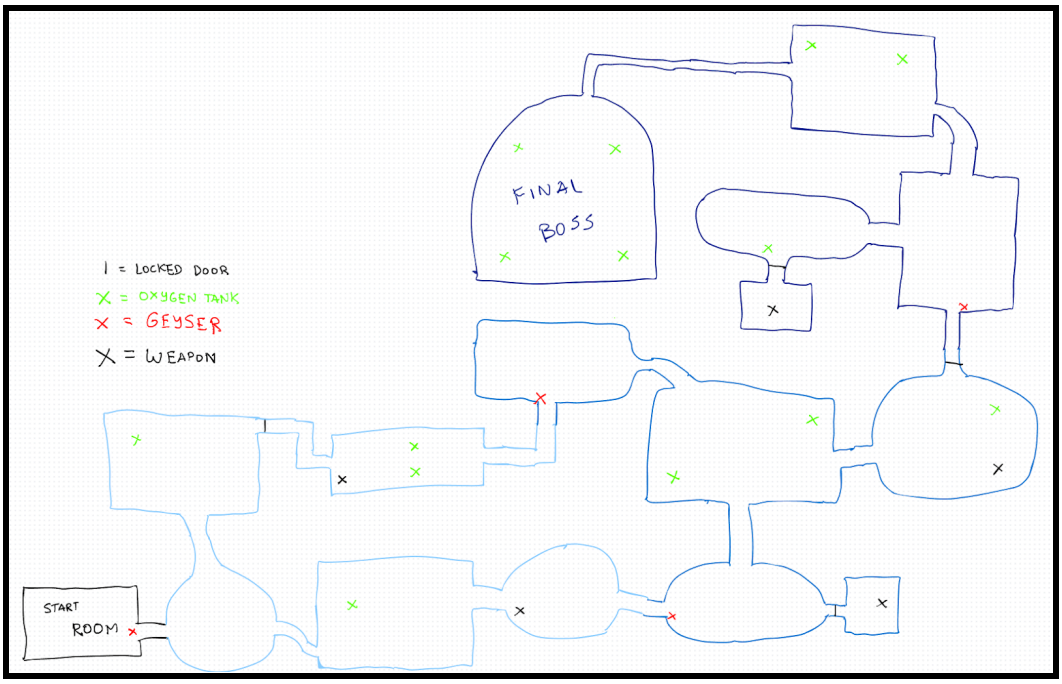


Figure 4. Player will traverse a 15-room generated map

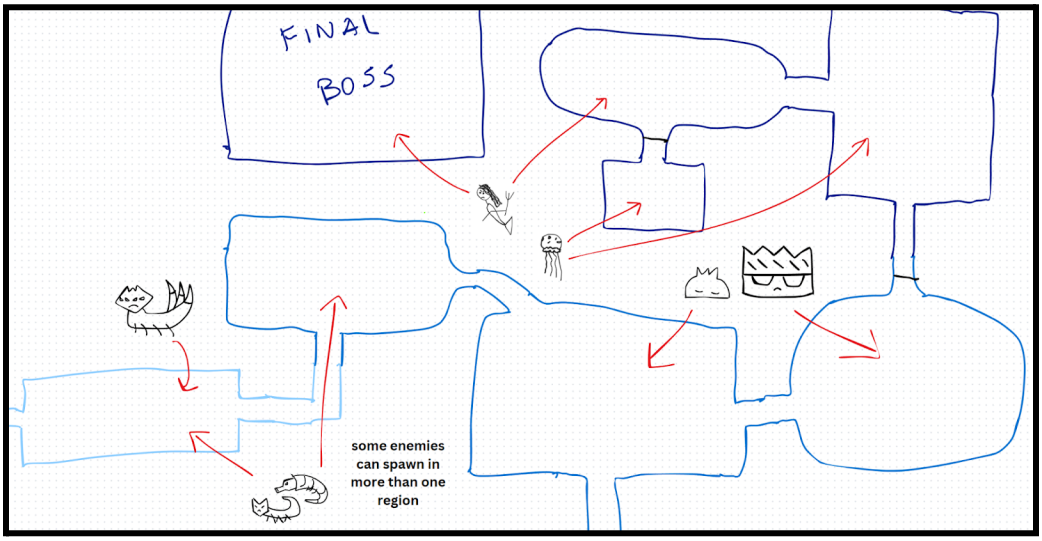
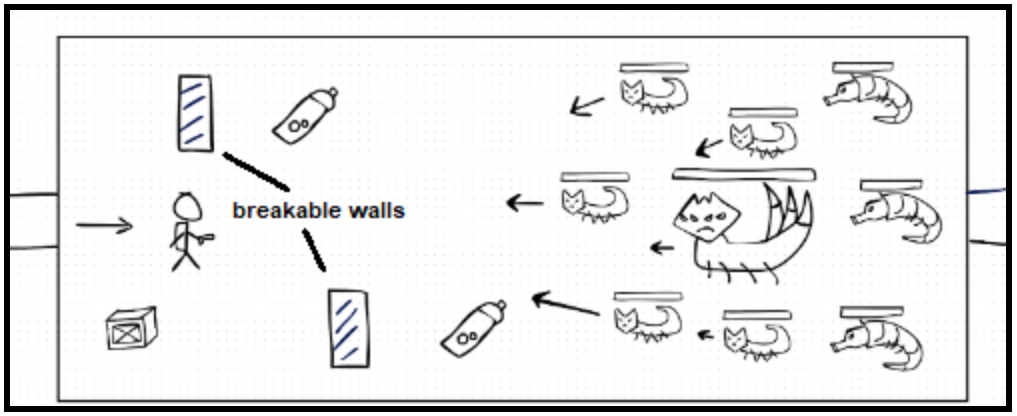


Figure 5. Color-coded regions on Figure 4 will have different types of enemies



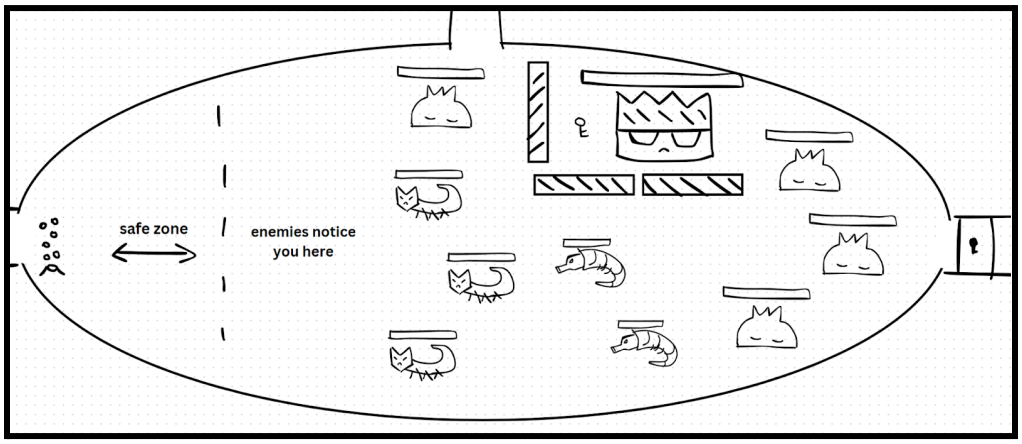


Figure 6 and 7. Example rooms with randomly generated entities

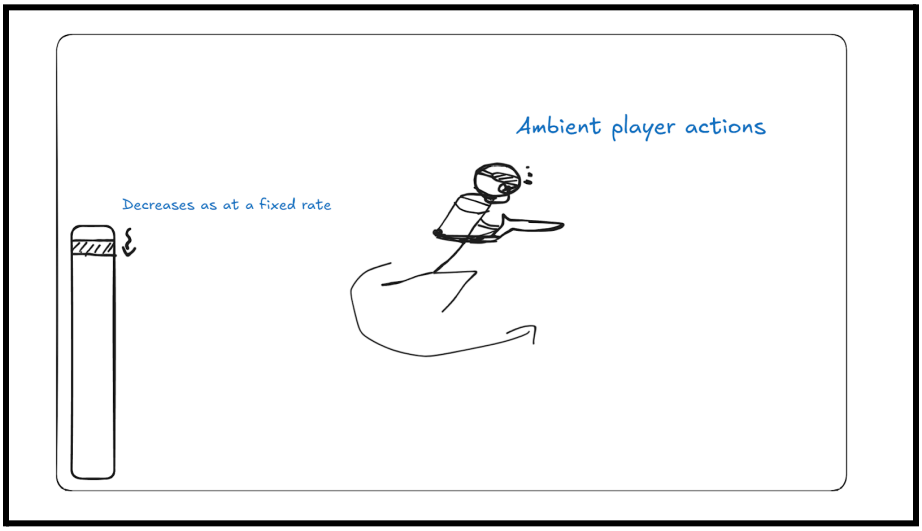


Figure 8. Oxygen depletes slowly over time

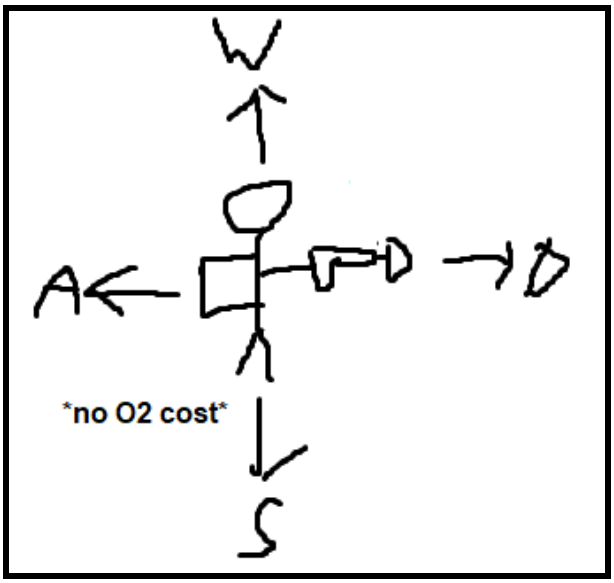


Figure 9. WASD to move, no extra oxygen cost for simple movement

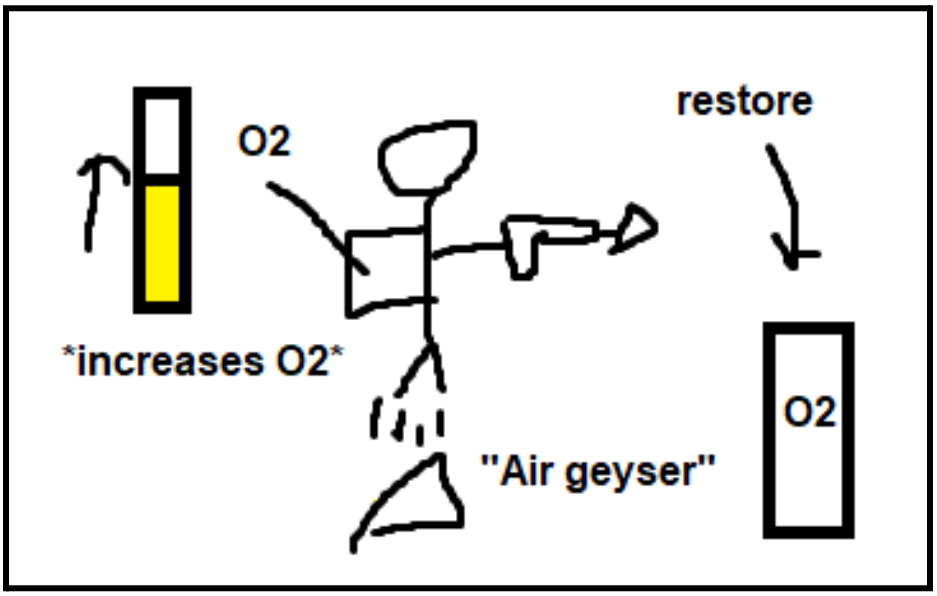


Figure 10. Player can increase oxygen level using air tanks or air geyser supply

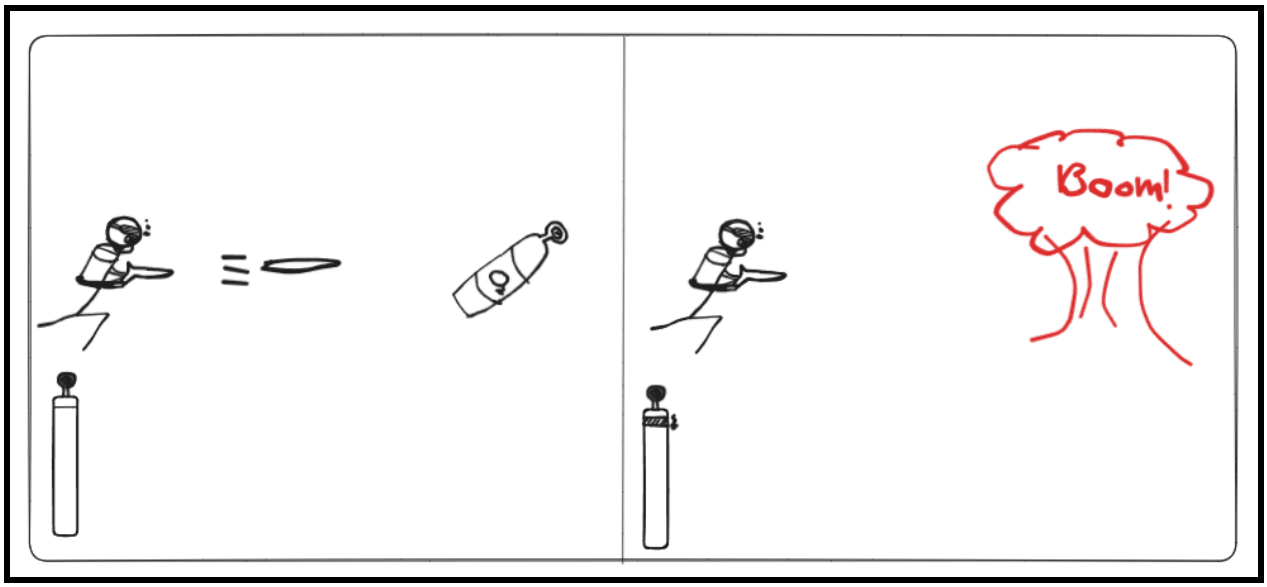


Figure 11. Oxygen Tanks (Map Object) can also be shot to cause area explosion

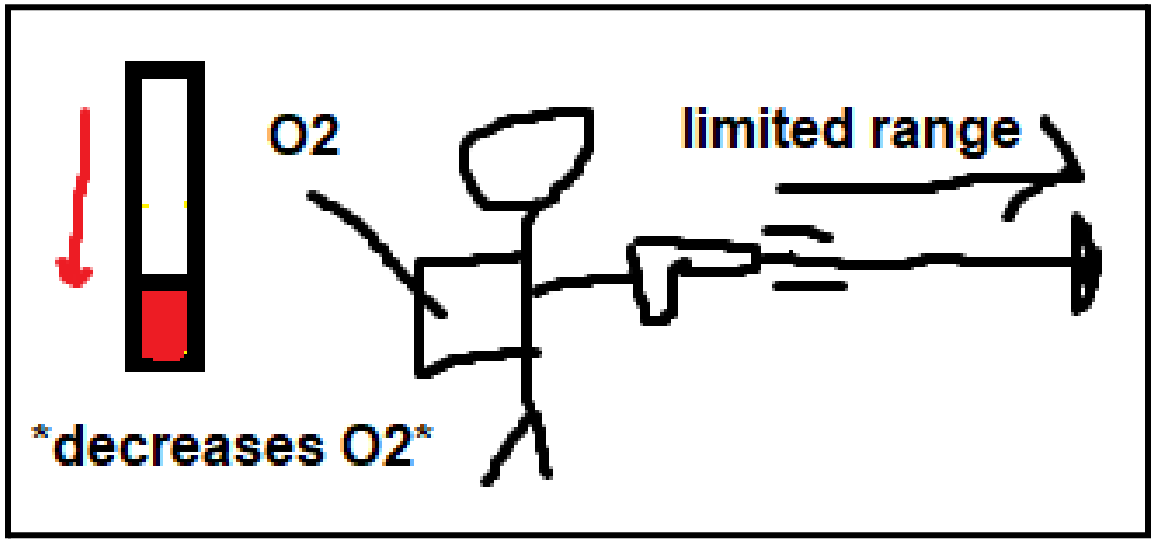


Figure 12. Firing harpoon requires some oxygen (air propelled)

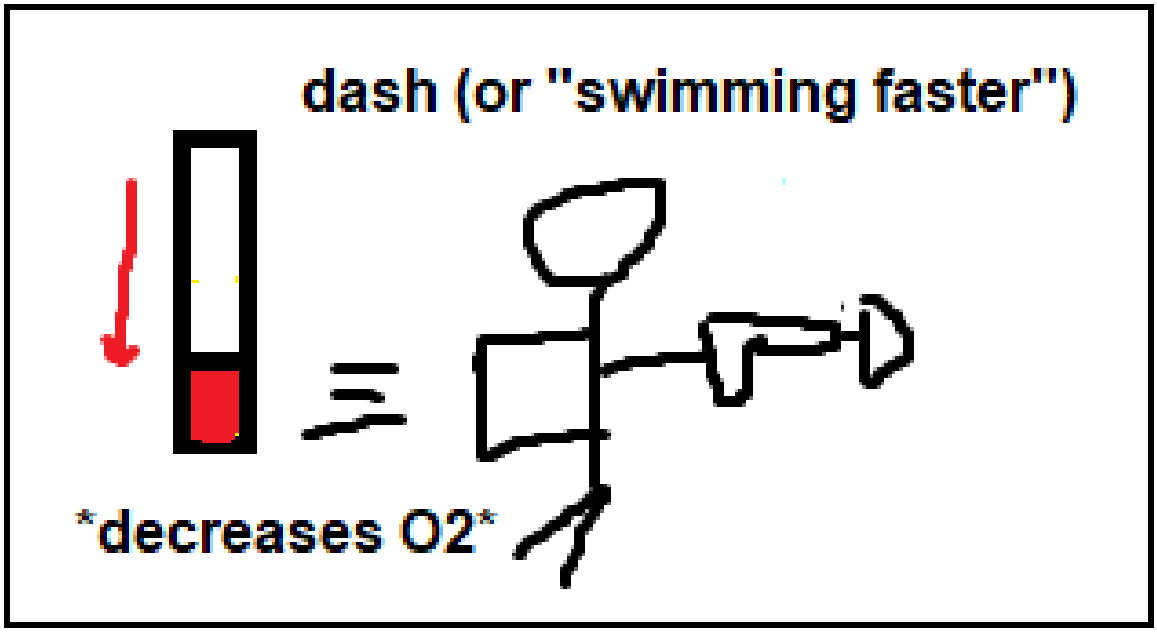


Figure 13. Player can use oxygen to propel forward faster

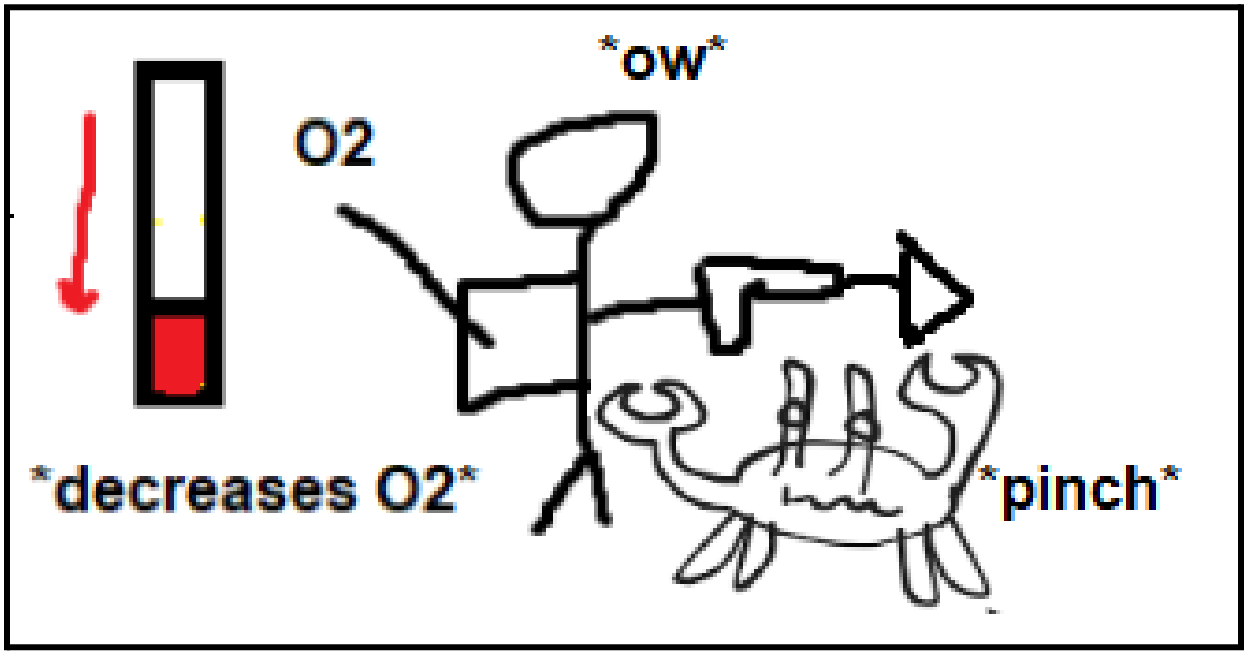


Figure 14. Fixed amount of oxygen depletes when hit by an enemy

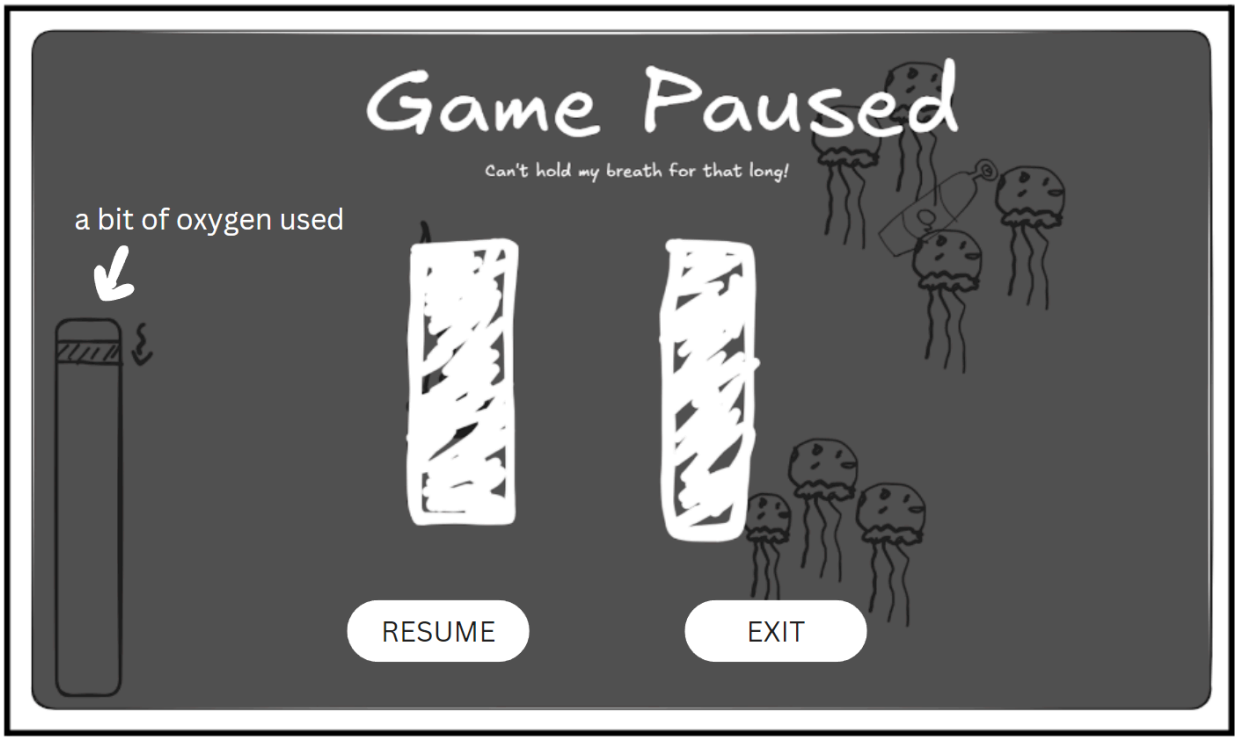


Figure 15. Pause screen; pausing will use some oxygen

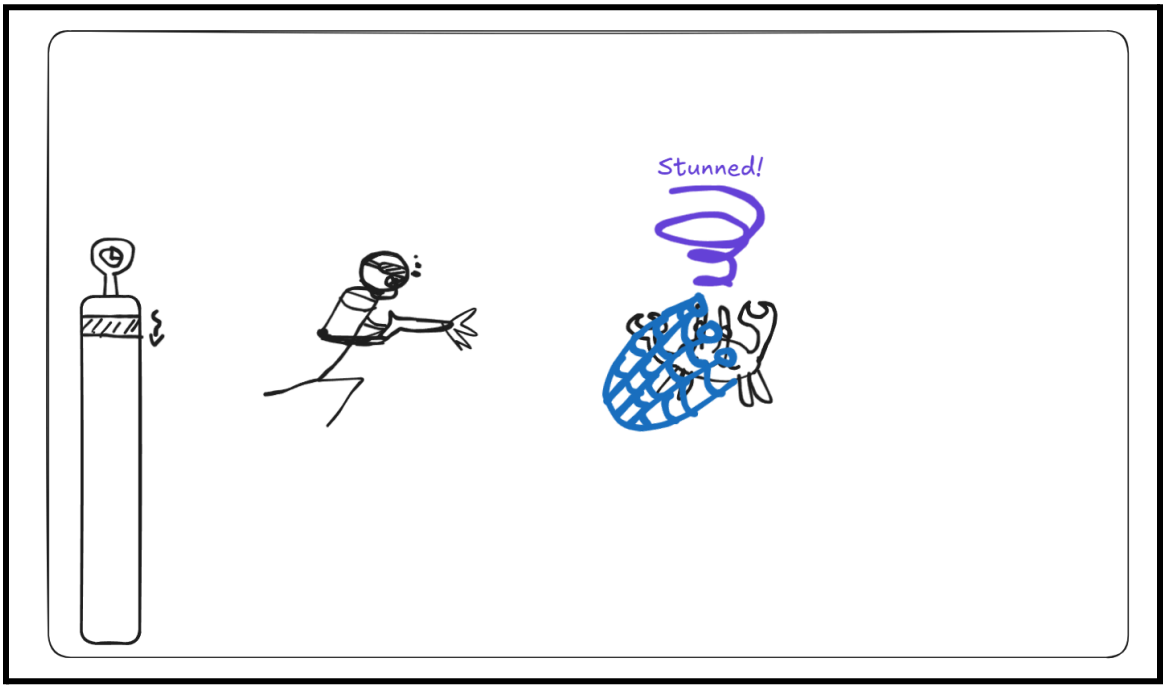


Figure 16. Consumable: Net that can be used to stun an enemy, uses some oxygen

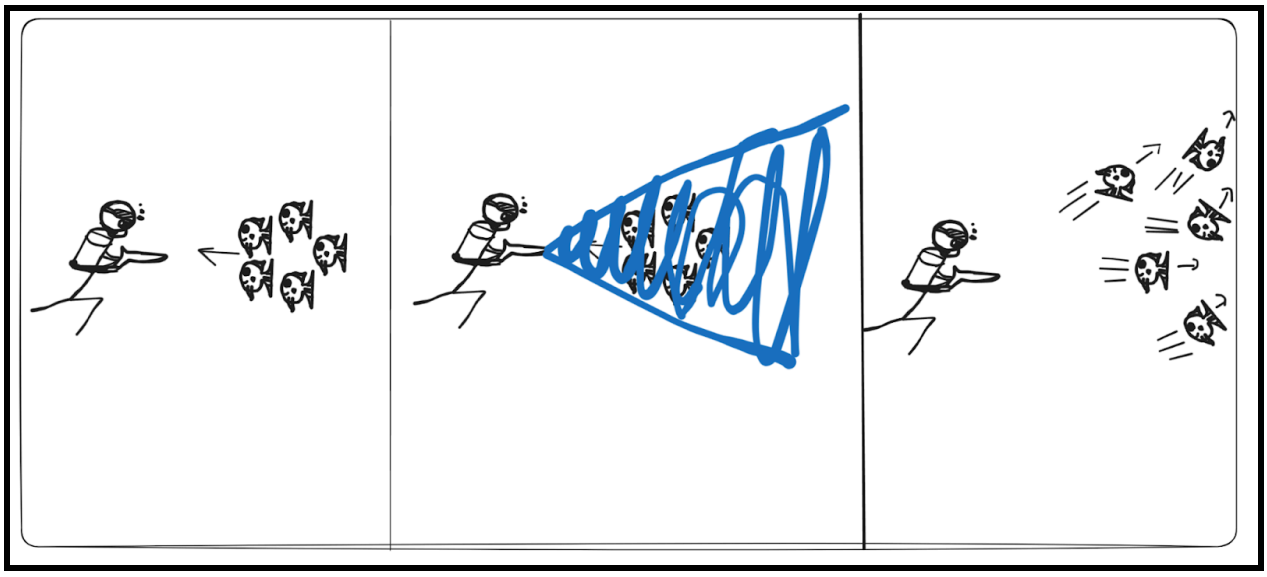


Figure 17. Consumable: Concussive that can be used to push enemies in a cone back

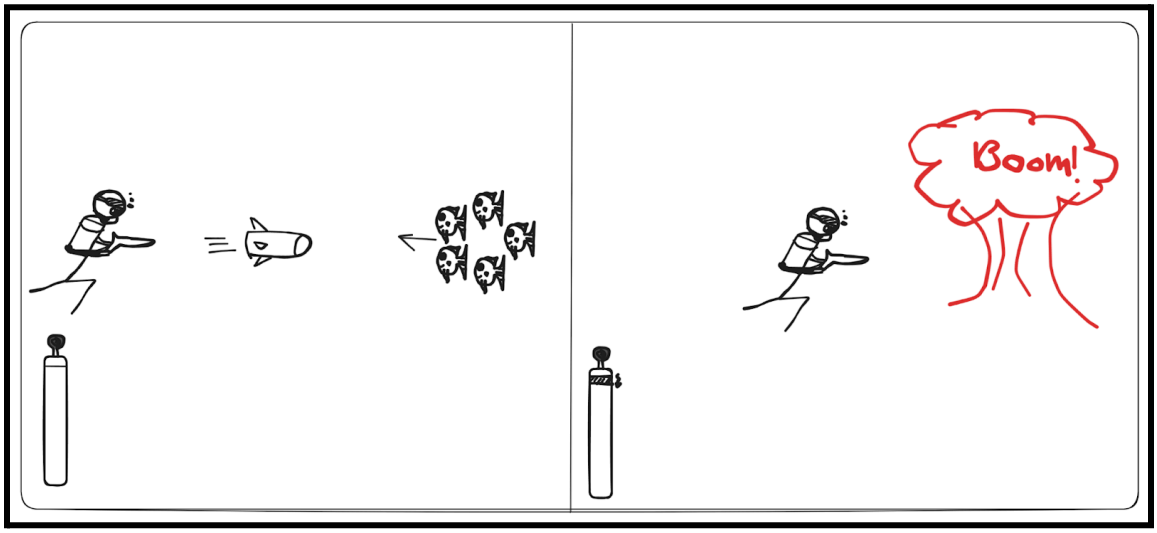


Figure 18. Consumable: Torpedo that can be used to do damage in an area, requires low amount of oxygen

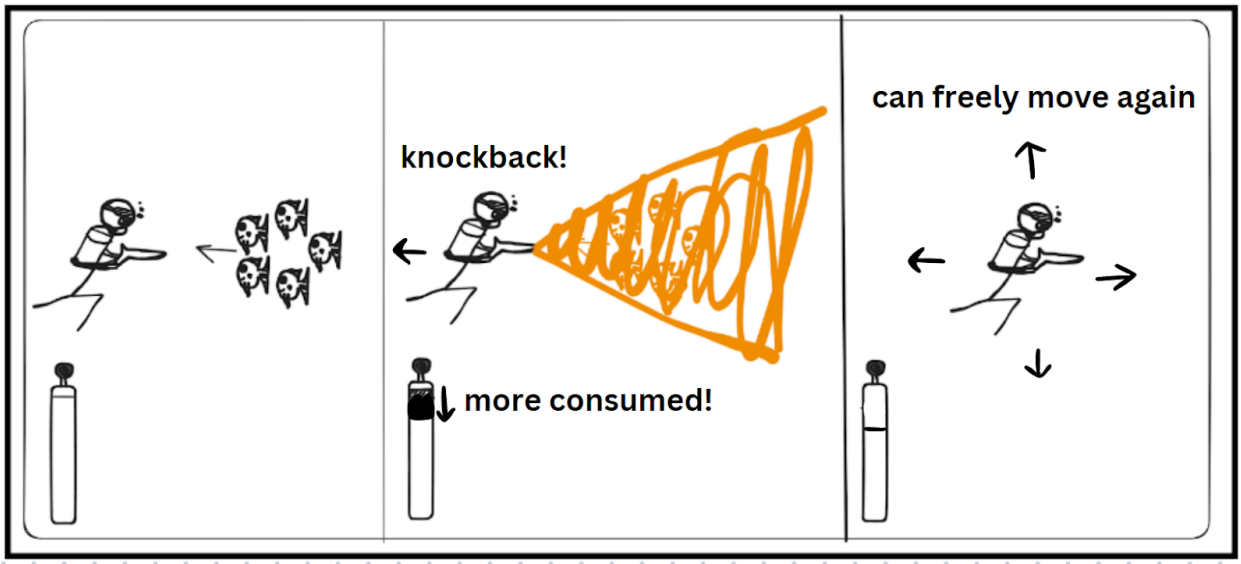


Figure 19. Pistol shrimp clears cone area of basic enemies or deals high damage to bosses, has knockback recoil and consumes high amount of oxygen

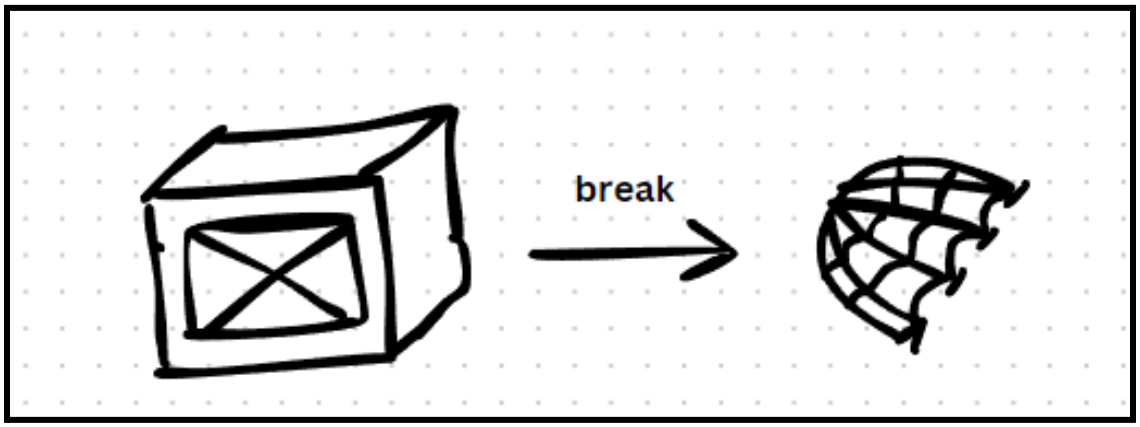


Figure 20. Breakable crates can contain usable items

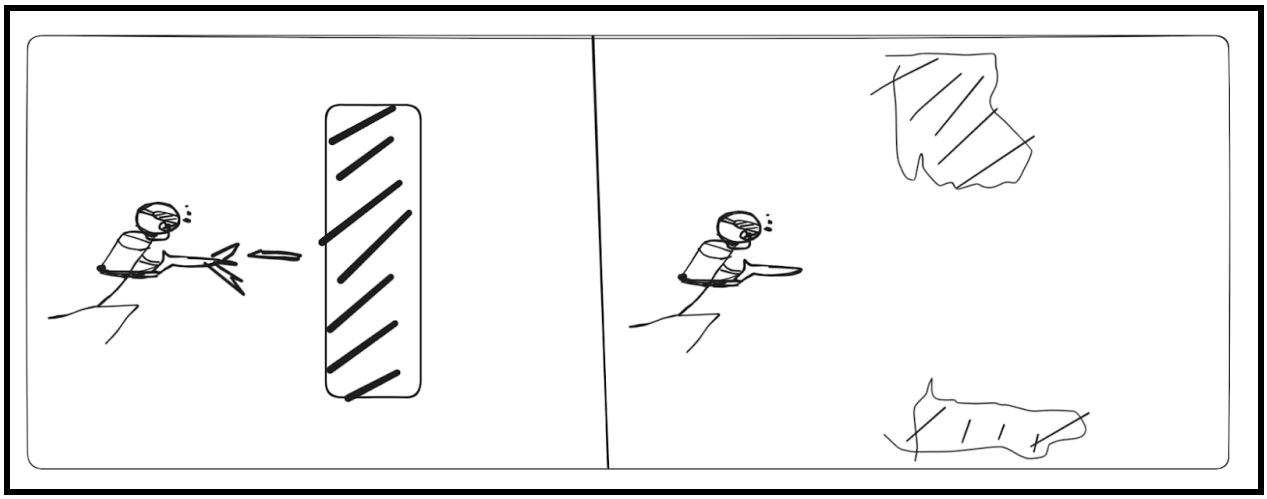


Figure 21. Shooting breakable walls destroys them

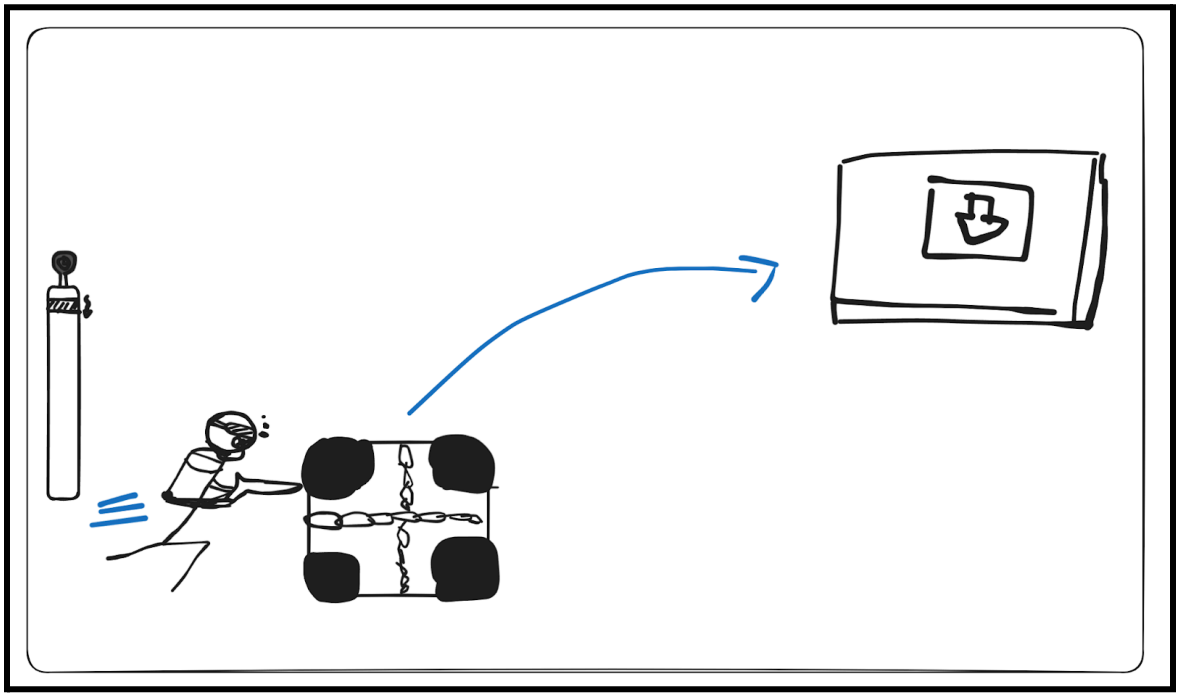


Figure 22. Unbreakable stone block can be pushed onto pressure plates

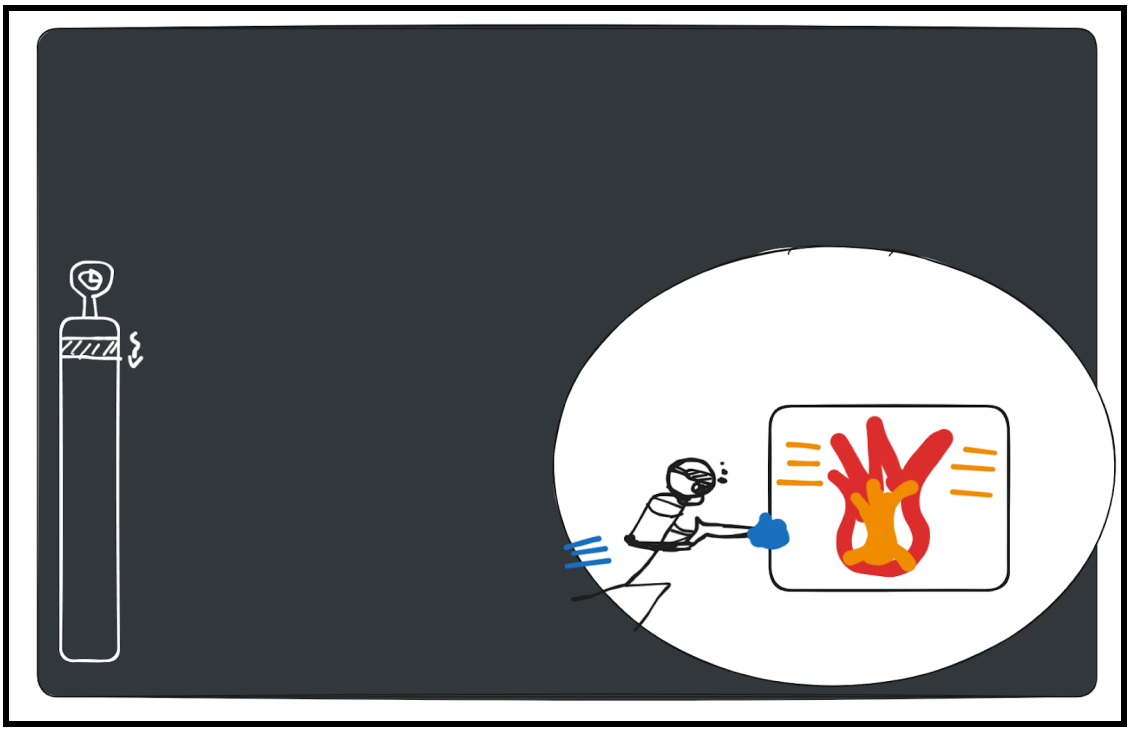


Figure 23. Dark room with unbreakable lamp box; ignited using oxygen and pushed around

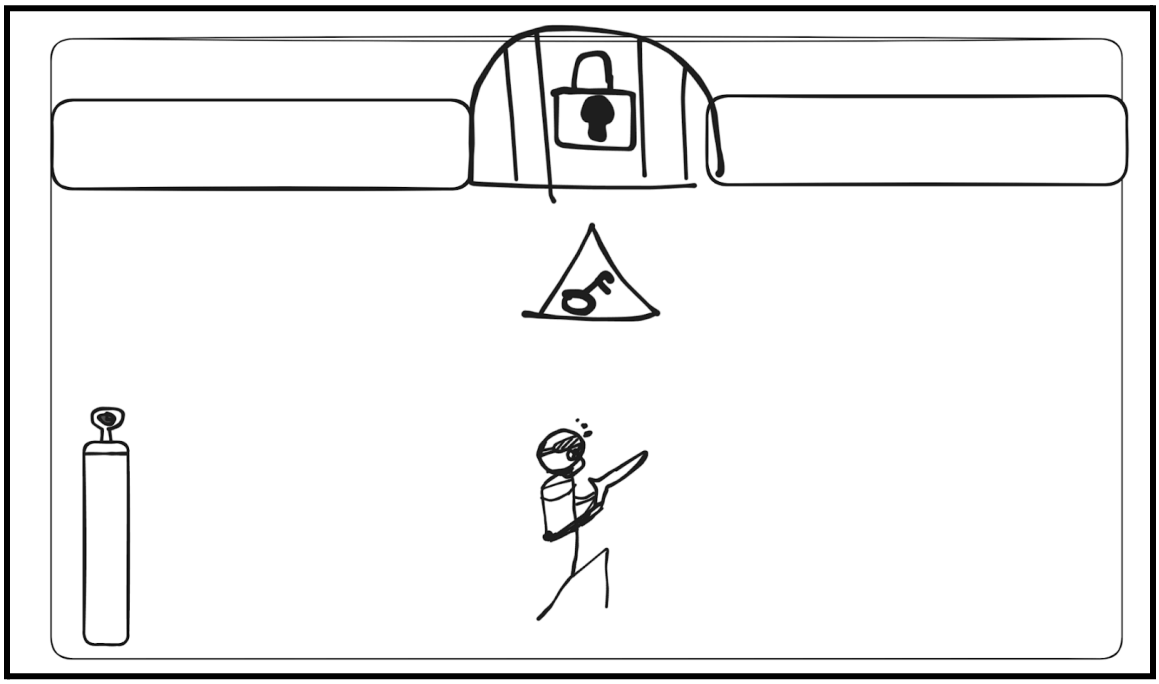


Figure 24. Room with key objective to open door

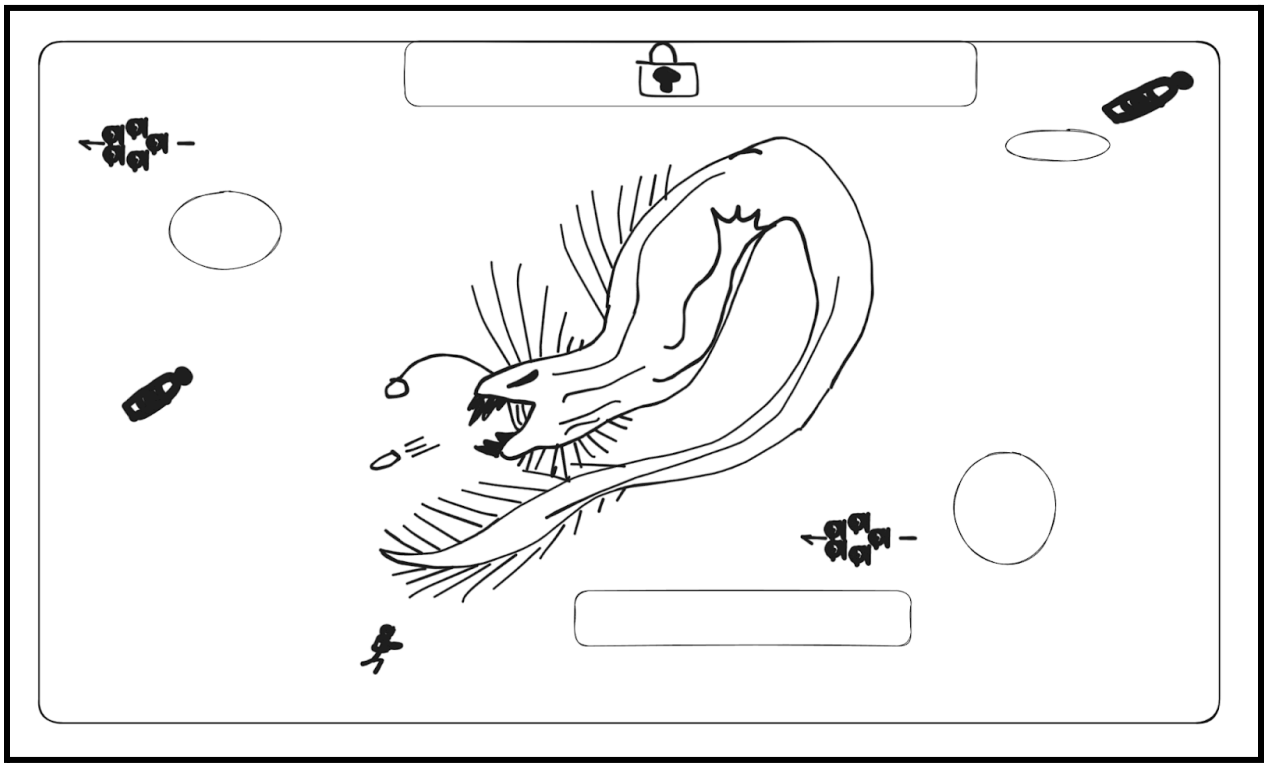


Figure 25 Room with mini boss objective to open door

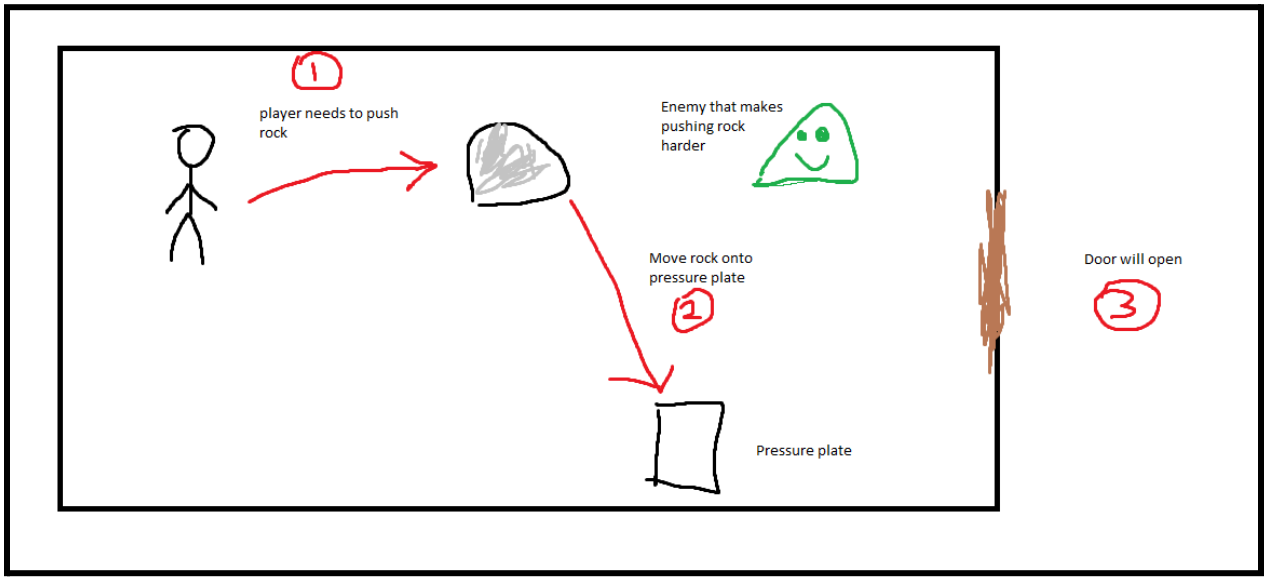


Figure 26. Room with pressure plate objective to open door

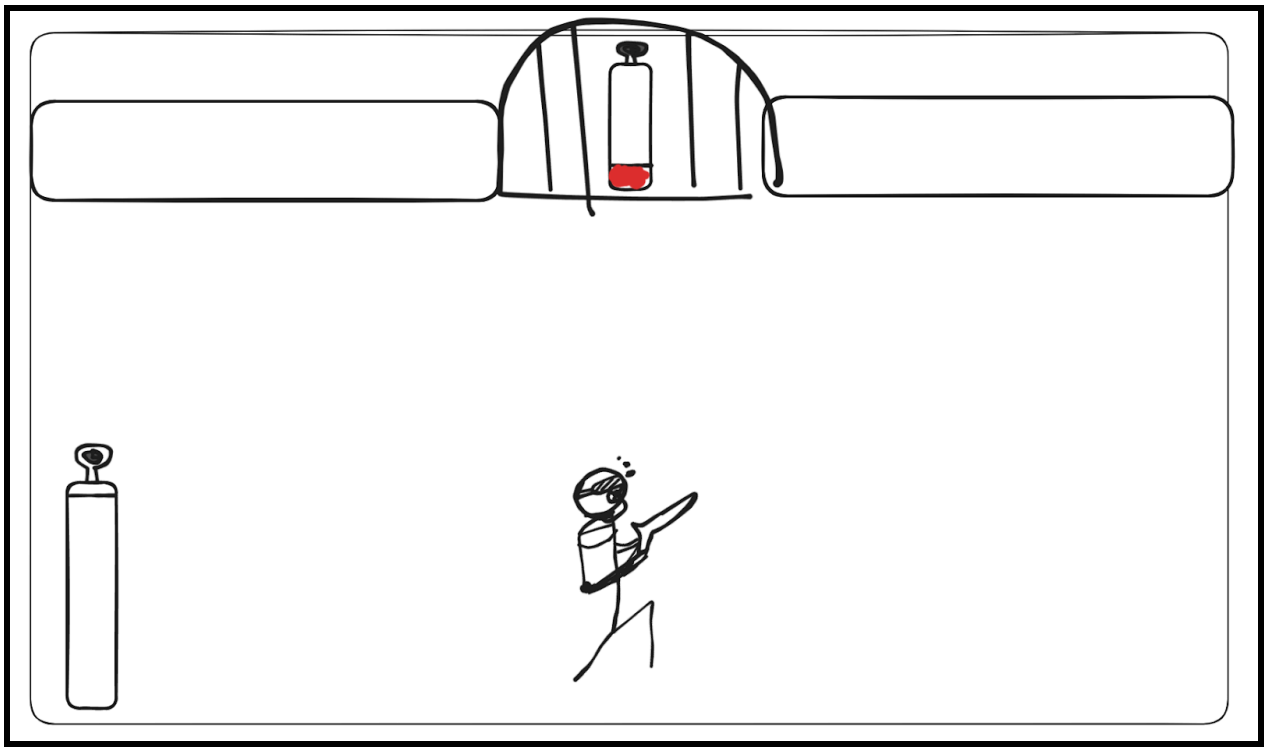


Figure 27. Room with door that requires oxygen to open

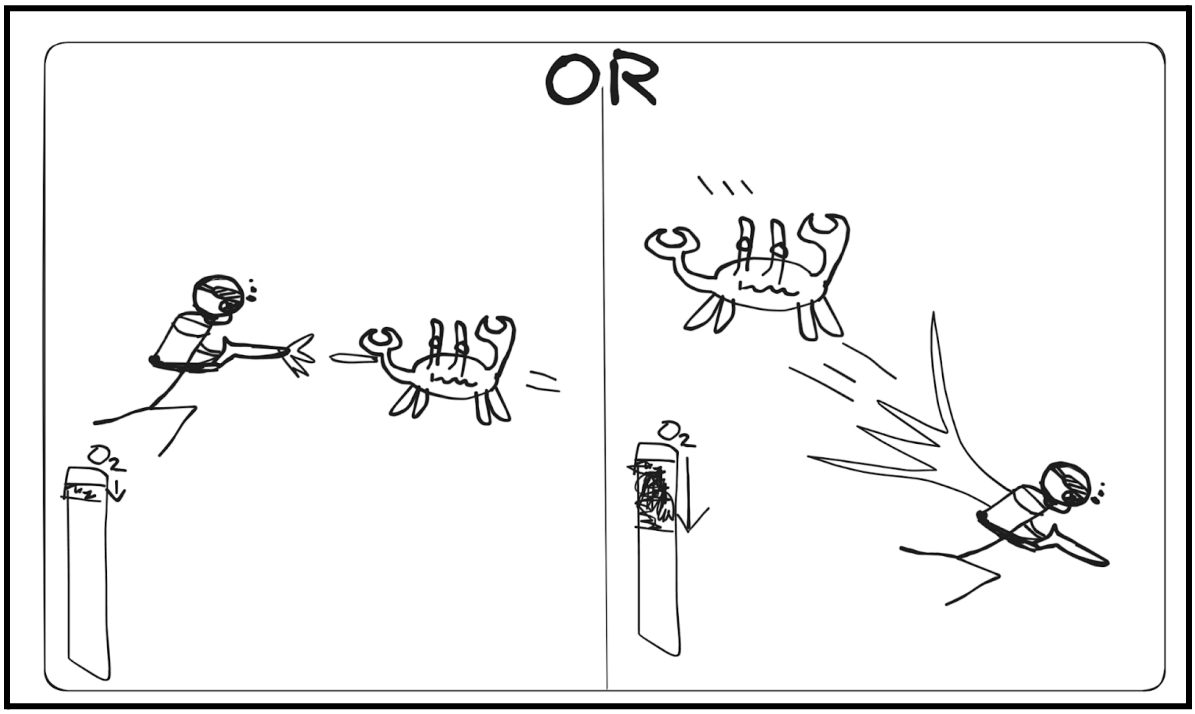


Figure 28. Player completing objective fighting enemy vs. dashing around enemy

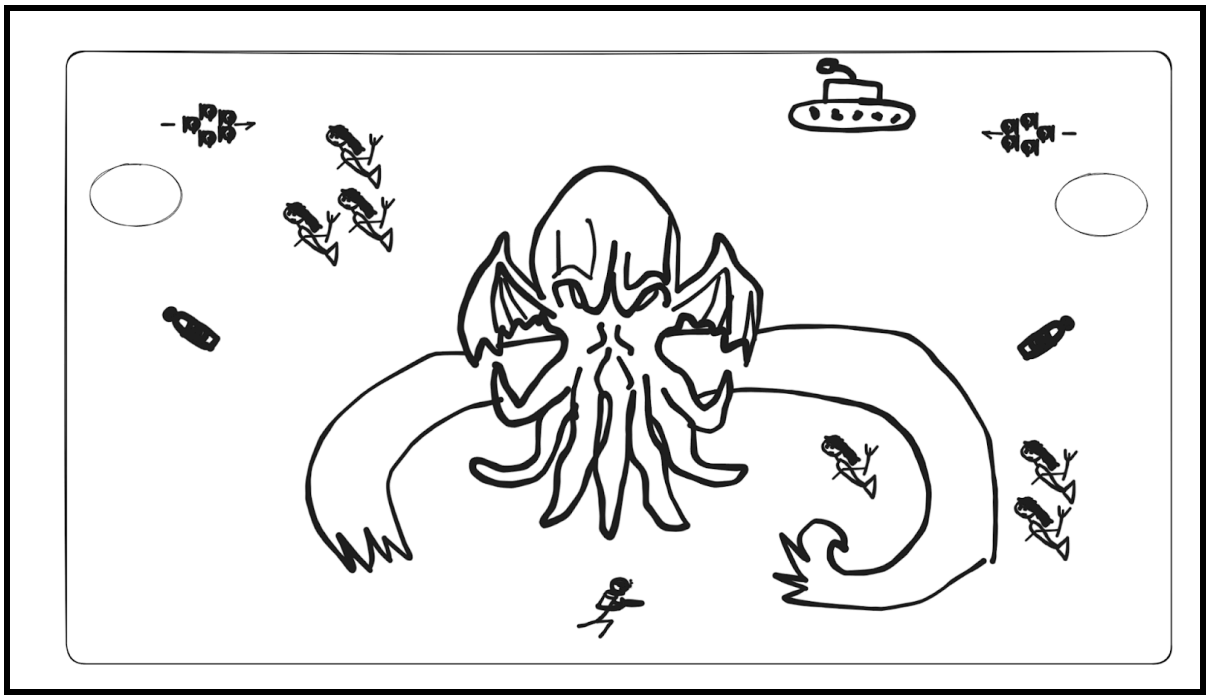


Figure 29. Boss room and final boss (silhouette from opening scene)

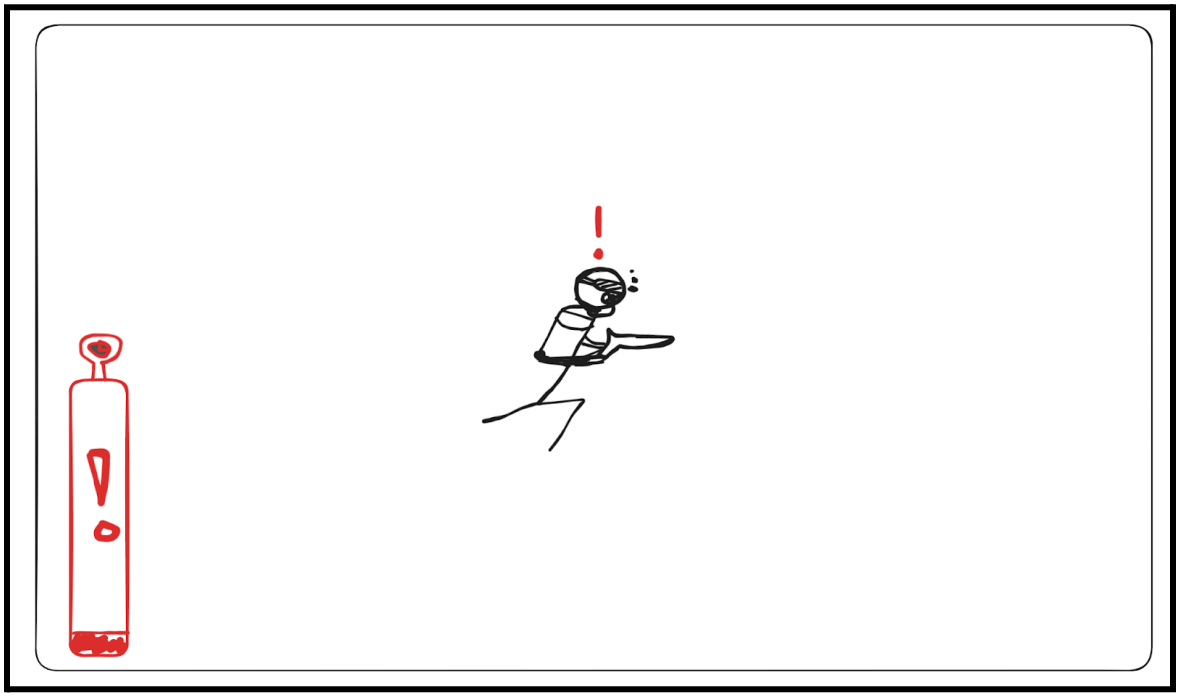


Figure 30. Oxygen meter will change colour and music will change when oxygen is low

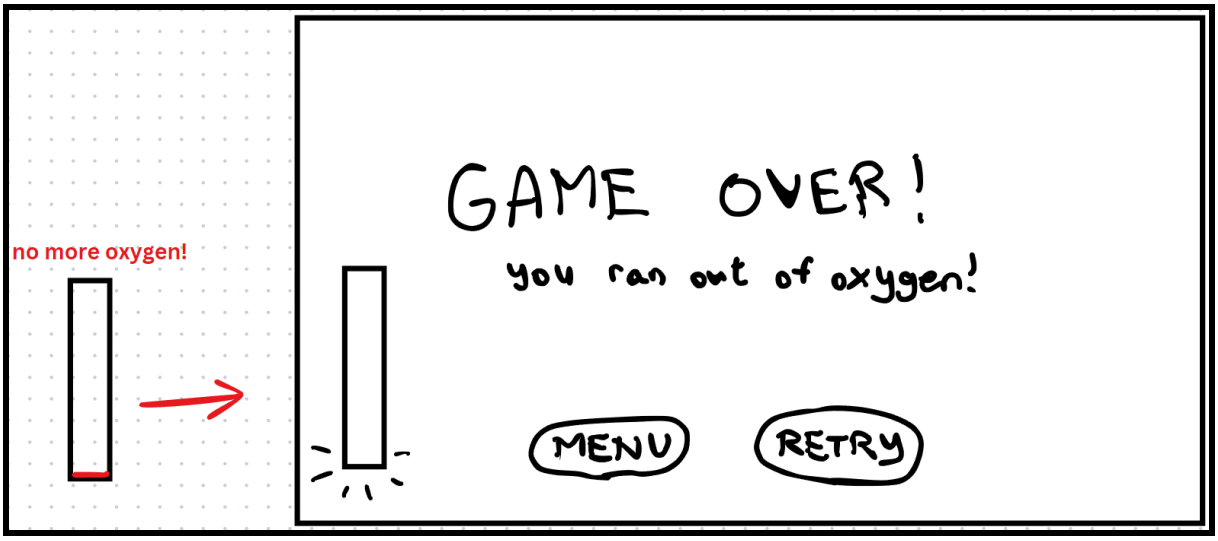


Figure 31. Player will die when O_2 level runs out

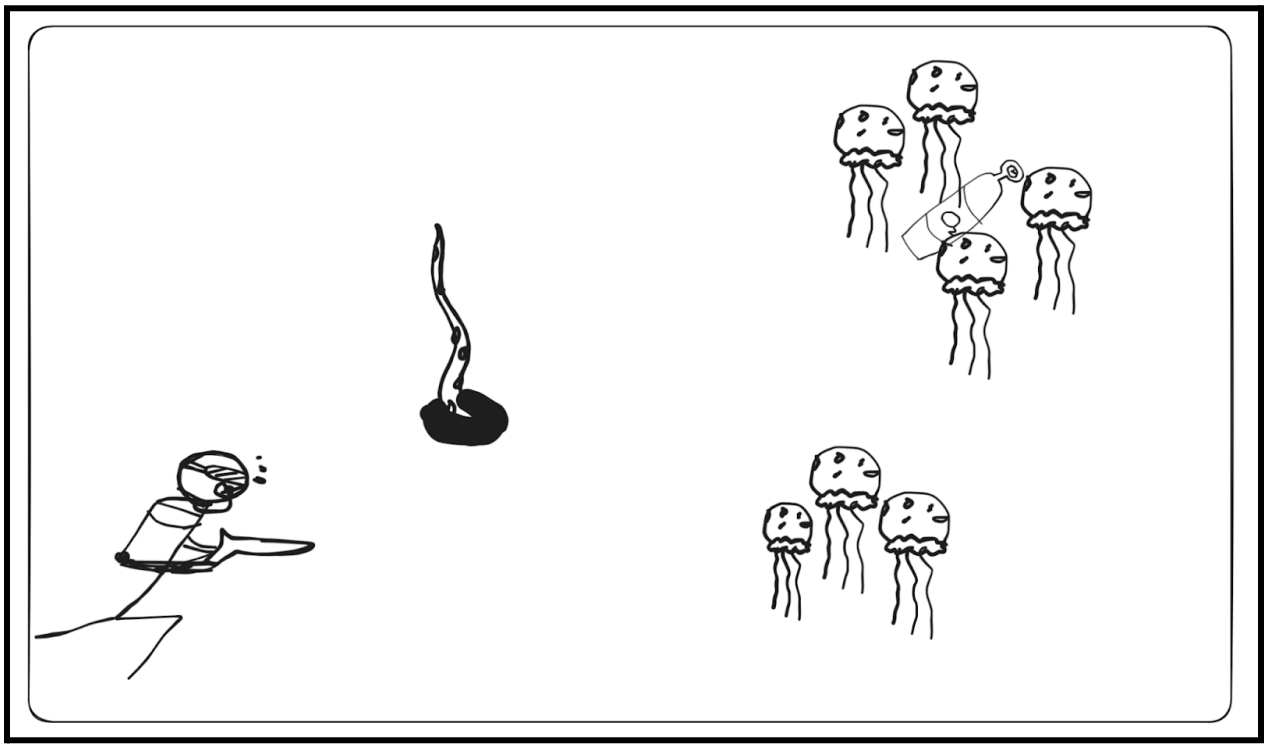


Figure 32. Stationary enemies that the player can attack or avoid

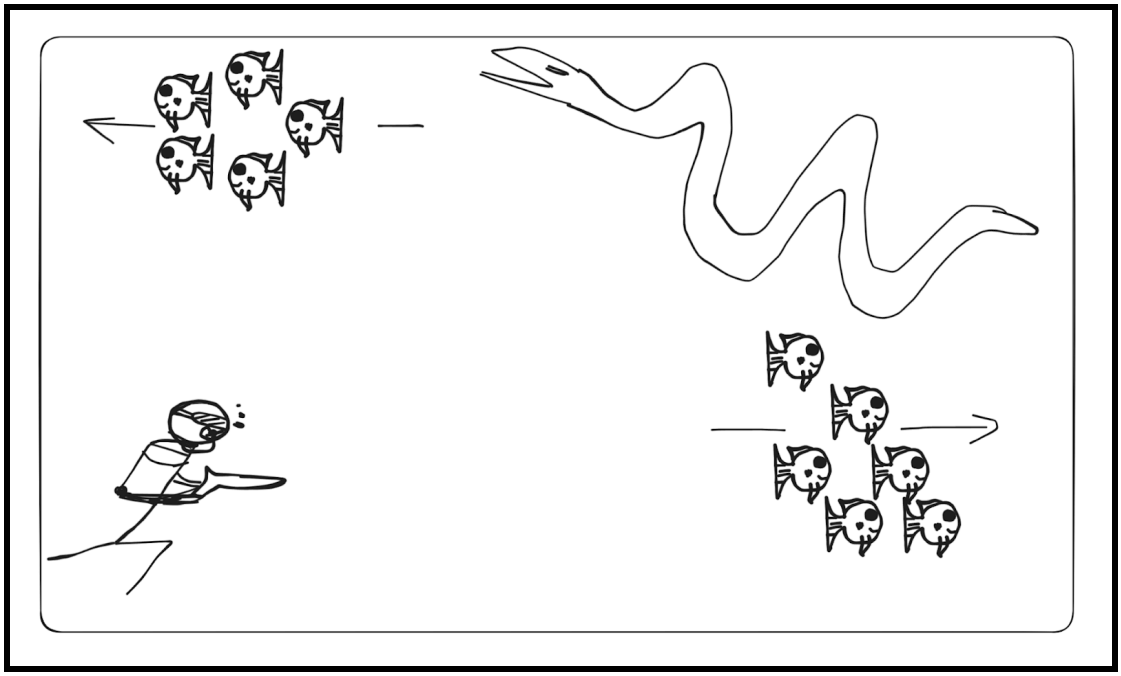


Figure 33. Enemies move around the map that the player can engage with or wait to pass

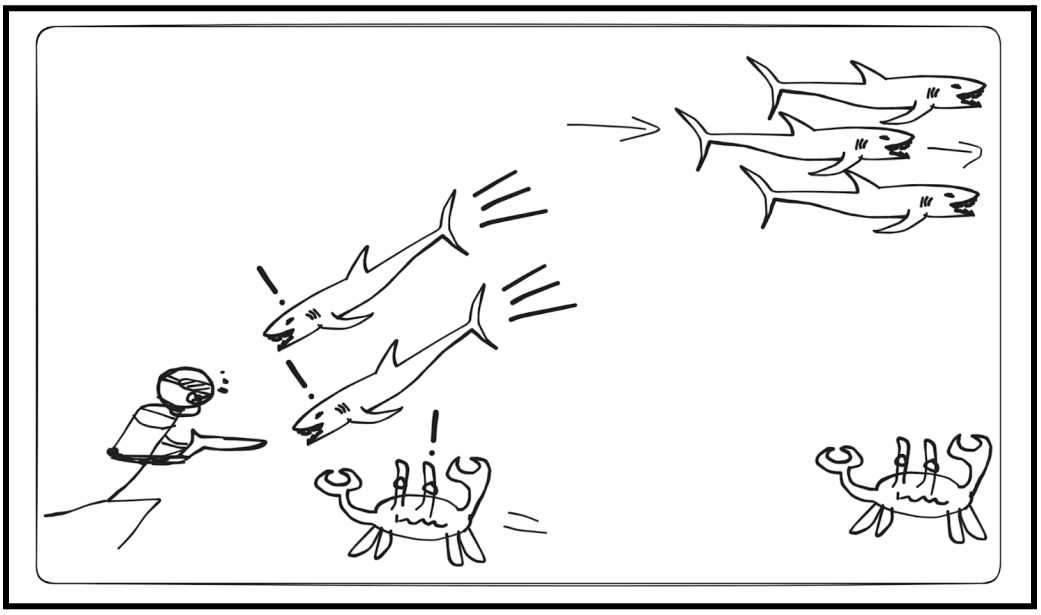


Figure 34. Enemies that will pursue the player once nearby

Figure 35. Game over screen

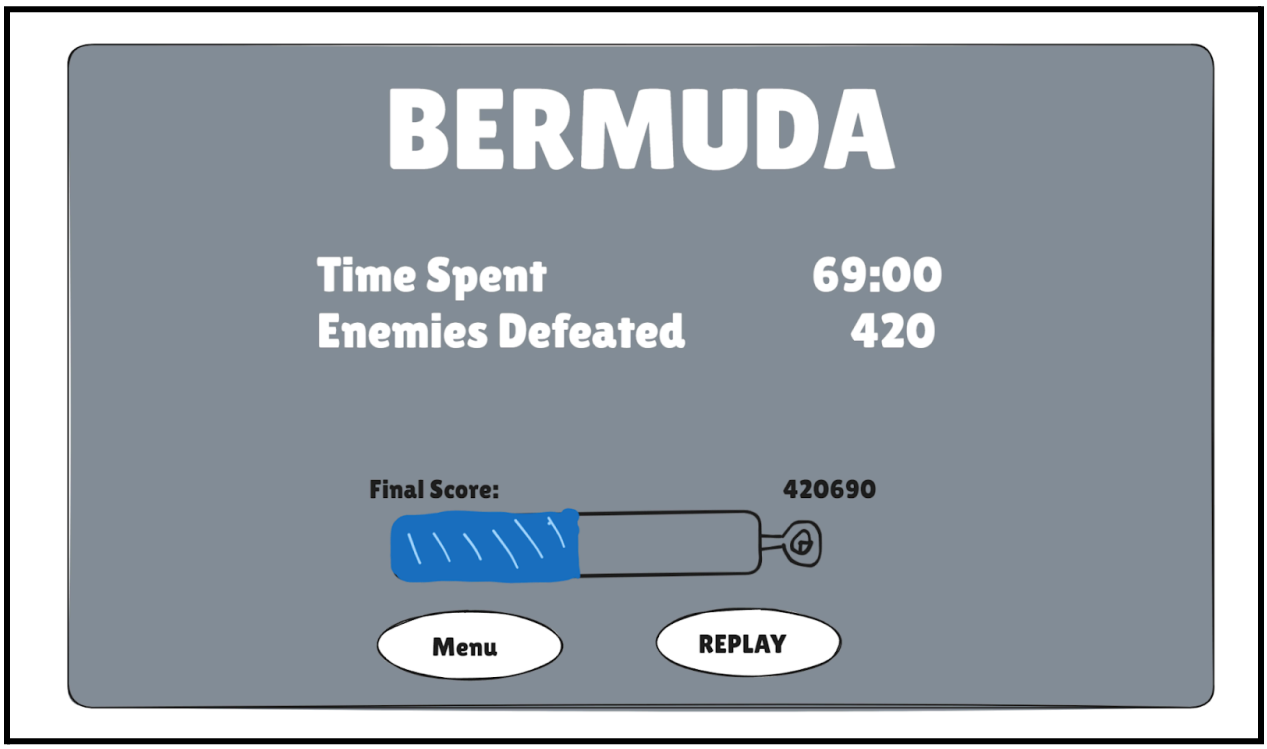


Figure 36. Game completed screen with enemies defeated and time spent

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:

(Where we get assets is in the assets section)
Shoot, damage, break, and loot effects

- Shooting - This will be spawning one of our drawn projectiles (harpoon, net, etc.)
    - These projectiles when shot will move across the screen in the direction they were shot at.
- Damage - This will be rendering a solid red flash on the player or enemy when they are taking damage (projectile collision or enemy/player collision).
- Break - This will be rendering cracks on the boxes/walls that are breakable when taking damage.
- Loot - loot (keys, nets, etc) will have shiny yellow particles emitting out of it to indicate reward.
- Brightness - certain areas will be cast in darkness, where any loot or terrain features will be unseen until a light is used to illuminate it.

HUD

- Oxygen tank image on the left-hand side with a meter representing the amount of oxygen the user has.
    - The meter will increase and decrease based on the amount of oxygen the player has.
- Pause button on the top right of the screen
- Bottom right of the screen, in a row, we have four spaced out inventory item icons with a number representing their count. Items with a count of 0 will not be displayed
    - Net
    - Torpedo
    - Concuss
    - Pistol shrimp
- Top left of screen, a key icon will appear if the player is holding a key.

Character Sprites

- Player & Enemy sprite (Where we get the sprites is mentioned in asset section)
    - Idle sprite for when they are idle.
    - Swimming sprite for when they are moving.

Map

- Map will be rendered as tiles in the background.
    - Floors will be rendered with floor tile texture.
    - Walls will be rendered with wall tile texture.
    - Pressure plates will be rendered with a unique floor tile texture.

Environment

- Box sprites and door sprites will be rendered on top of the map tiles.
    - Boxes have cracking keyframes when taking damage.
    - Boxes after taking enough damage will break into multiple parts and then disappear.
- Loot will be rendered when a box breaks.
    - Loot means things like keys, nets, torpedoes, etc.
- Moveable rocks will be rendered on the map.
    - When the player pushes a rock, the rock will be rendered moving in that direction.

Assets:

Sprites

- We'll create pixel sprites for enemies, loot, doors, boxes, walls, rocks, and the player. If there is a lack of time, we will find free non-copyrighted assets.
    - Places we'll get free assets from:
        - https://kenney.nl/
        - https://itch.io/game-assets/free

Audio

- We'll create sound effects for shooting, death, monsters, water, breaking box, and breathing.

- We'll create music for ambience in rooms and main menu.
- If we run into time crunches when creating sound effects and music, we will source them from https://freesound.org/

Level background

- We'll draw our maps and rooms using tiles.

2D Geometry Manipulation:

Collisions

Collisions will be defined as when the player's center comes within a defined distance with another entity/environment feature.

- We will have collision detection for players, environment, enemies, and projectiles.
    - Player -> Enemy collision
        - Players will take damage on colliding with an enemy
    - Player -> Environment collision
        - Players will be blocked by walls and boxes.
        - Players will be able to move rocks by colliding with them, pushing the rock in the same direction the player is moving.
    - Projectile -> Player collision
        - Players will take damage if hit by a projectile
    - Projectile -> Enemy collision
        - Enemy will take damage if hit by a projectile
    - Projectile -> Environment collision
        - Projectile will be blocked by walls and rocks.
        - Boxes will take damage if hit by projectile.
    - Enemy -> Environment collision
        - Enemy will be blocked by walls and boxes.

Transform

- Upon being broken by the player, boxes will transform into loot.
- Upon being interacted by a player holding a key, doors will open.

Destroy

- Boxes can be destroyed by the player
- Certain walls can be destroyed by the player

Gameplay:

Player

- WASD to move
- Left click to shoot
- Cursor to aim
- Shift + movement key to dash.
    - This will have a cooldown and will consume oxygen
- E: Interact with environment
    - Player opens door when holding key
- ‘1’ Key: Load Net
- ‘2’ Key: Load Torpedo
- ‘3’ Key: Load Concussive Weapon
- ‘4’ Key: Load Pistol Shrimp

Character-World

- Puzzles that obstruct the player
    - Find the key for a locked door
    - Pressure plate to open door
        - Player can move rocks to the pressure plate
- Interactable objects
    - Player can break boxes to gain loot
    - Player can move rock to pressure plate
    - Player can interact with locked door to open the door when they have a key
- Enemies that obstruct and damage the player
    - Regular enemies
        - Enemies that attack by running into the player.
        - Ranged enemies that shoot projectiles
        - Area of Effect enemies
    - Mini Bosses

        - Same as regular enemies, but stronger
    - Bosses
        - Larger sized
        - Will have a unique behavior (see Final Boss section)

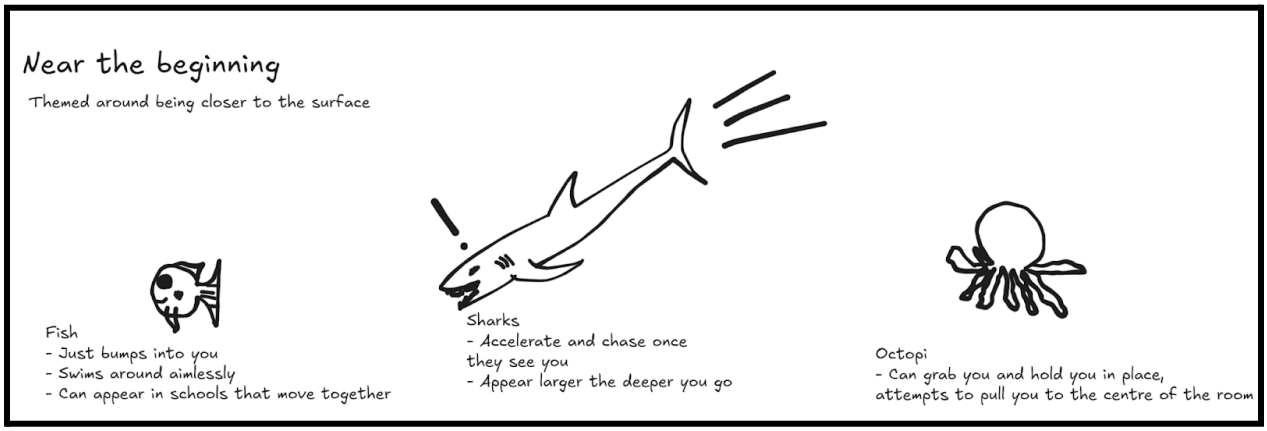


Figure 37. Enemies that will appear near the beginning of the game

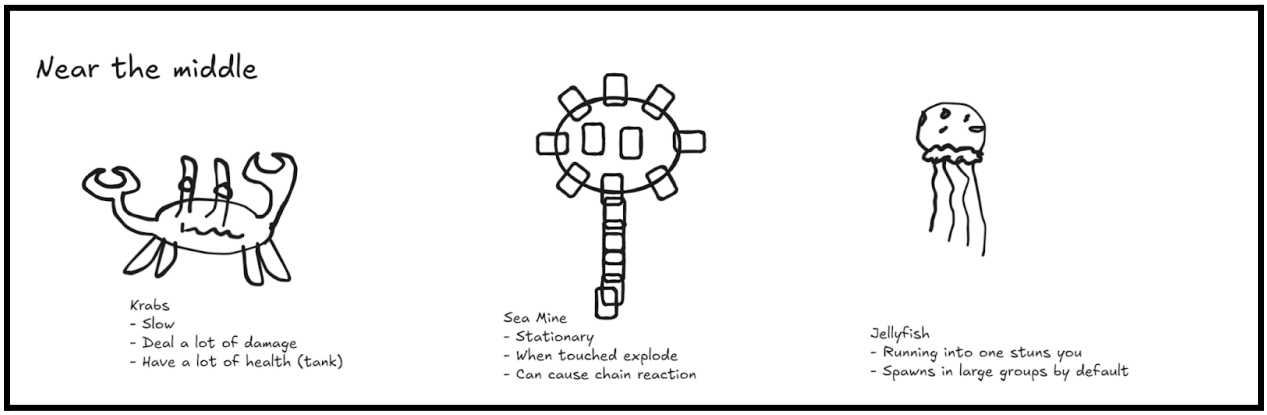


Figure 38. Enemies that will appear near the middle of the game

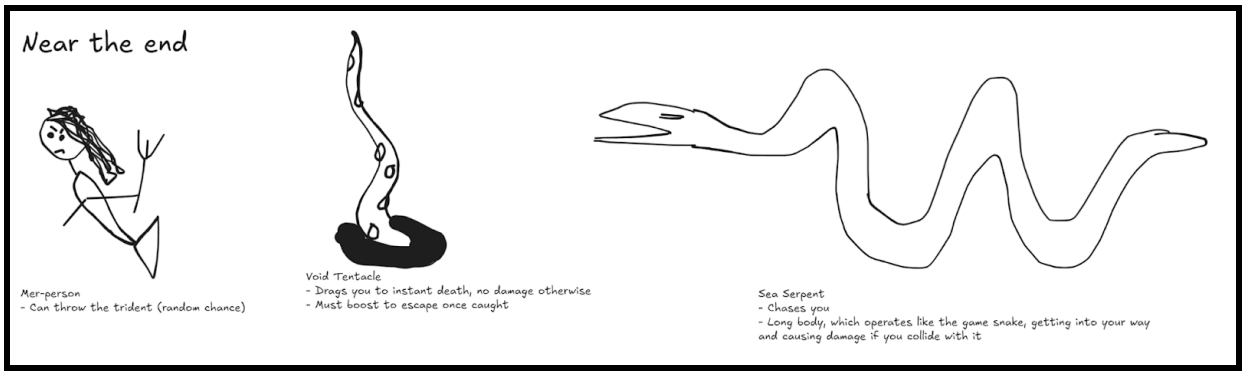


Figure 39. Enemies that will appear near the middle of the game

AI:

Enemy

- Detection behavior
    - During this behavior, the enemy will sit idle and occasionally walk back and forth (if walking into a wall, they will be blocked).
    - The enemies will have a detection radius surrounding them, if the player enters that detection radius, the enemy will move to chasing behavior or targeting behavior based on whether they are a melee or ranged type.
- Chasing behavior
    - We will have the enemy chase the player by moving towards the player's position.
    - If the player leaves the enemy's detection radius, they will return to the detection behavior.
- Dodging behavior
    - Stronger enemies and mini-bosses will have a dodge radius. If the player's projectile enters this dodge radius, we will have a random number generator that generates a random integer from [1,3]. If the RNG hits 3, the enemy will attempt to dodge the projectile by moving in the opposite direction of where the projectile enters the dodge radius.
- Targeting behavior
    - Ranged enemies will shoot a projectile at the player's position when the player enters their detection radius.
    - When players exit their detection radius, they return to the detection behavior

- Tentacle behavior
    - During this behavior, when the player collides with an enemy (comes within a certain range), the player's position will be moved towards a defined position relative to the enemy. For example, the player can be dragged towards the base of the tentacle enemy sprite.
- Snake behavior
    - Enemies defined with snake behavior will be defined as multiple entities, named the head and the tail. The head is the primary entity that dictates the behavior of the entity and its movement. The tail entities will follow the movements of the head entity with some delay.
        - This is attempting to emulate the behavior of the snake in the game snake
- Swarming Behavior
    - Enemies can be spawned in groups, and will behave as a collective. This will be accomplished through BOIDs.

Loot Generation

- When a box breaks, we will have a random number generator that will generate a number from [1,5] and have one of our current 5 loot spawn based on the number generated.

Physics:

Player movement

- WASD controls will add directional acceleration to the character. After the controls are released, the player's acceleration gradually slows to 0.
- SHIFT + WASD will dash which will basically add a high initial directional velocity and acceleration but only on the first key press.

Overlapping entities

- Entities will be able to occupy the same position

Projectile kinematics

- Player harpoon will have a high initial velocity until it slows to a stop.
- Enemy trident will have constant velocity until it hits a wall where it will stop.

Sound:

We will play sound effects we created or obtained from freesounds.org during the following moments:

Sound effects:

- Player actions
    - Shooting
    - Moving
    - Dashing
    - Picking up loot
- Enemy actions
    - Chasing
    - Shooting
    - Growling
- Player status
    - Low oxygen
    - Death
    - Gain oxygen
- Environment
    - Opening doors
    - Breaking boxes
    - Geyser bubbles

Music

- Environment-based music
    - Easy difficulty rooms
    - Medium difficulty rooms
    - Hard difficulty rooms
- Enemy-based music
    - Boss music, which takes higher precedence over environment-based music
- Main menu music

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.

If time issues occur, we'd like to cut down on the diversity of the constituent elements rather than axing them entirely.

In other words, if we cannot have all our enemy types, we will scale it down to just a handful of the most unique ones. If we cannot implement all the abilities, we will scale it down to just the pistol-shrimp or the net, etc.

However, we want at least *some* approximation of all the systems we named in Story to be in our game, especially for critical sections like Oxygen. We believe in our systems and want to provide the player with as many unique things to manage as possible, but time may disagree.

More specifically, here are the more advanced features we want to prioritize, and our plan B if time is an issue:

1. Randomly generated rooms according to the algorithm mentioned in Room Generation, perhaps tweaked to provide tight oxygen and challenging gameplay; fewer randomized aspects as mentioned in its section may be randomized as an alternative.
2. Enemies; while we want smart and unique enemy interactions as mentioned above in that section, we may end up making enemies dumbly target the player's coordinates or cut them entirely if we are really strapped for time. For example, we can not implement swarming behavior and snaking.
3. Artistic cohesion; this is difficult to concretely state, but we want the game to look good, and this will likely decrease if time left decreases. This may include points like deprioritizing complex lighting or shading algorithms, or spending less time looking for cohesive art.

Devices:

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

We plan to support a mouse and keyboard control scheme. The actions mentioned here are also described in more detail under Player and HUD.

- Left Click: Shoot Harpoon Gun or Loaded Consumable
- Mouse - Aim
- WASD: Basic Movement
- Shift + WASD: “Dash” Movement
- E: Action (Ex. Opening Doors, Ignite Lamp)
- ‘1’ Key: Load Net
- ‘2’ Key: Load Torpedo
- ‘3’ Key: Load Concussive Weapon
- ‘4’ Key: Load Pistol Shrimp

Tools:

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

https://www.pixilart.com/draw

Visual assets:

- https://kenney.nl/
- https://itch.io/game-assets/free

Sound assets: https://freesound.org/

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 plan on operating in sprints, with the first half of the sprint dedicated to implementing the various tasks and stories that we agreed to complete during that sprint. The second half of the sprint will be dedicated to helping team members finish their tasks, in addition to verifying

functionality, integration, and requirements. In case of extra bandwidth, new tasks can be brought in if and only if they can be completed within the sprint.

Team members will meet twice a week, tentatively after classes, to discuss what progress they have made in their tasks, and challenges that they may be encountering, and discuss work to be done in future sprints.

Team members will be splitting the work equally, however members will be given partial ownership over a certain feature. Owners will not be the sole developer on a feature, but will allow us to always have a local expert on an aspect of the game, and have someone push for features to get over the finish line. Owners will have the final say over implementation details in the case that there are disputes within the group, and be required to review code changes affecting their area.

- Tony: Movement, Player Controls
- Andy: Oxygen, Asset creation
- Bob: Enemies, Enemy AI, Build
- Brandon: Rendering
- Caeleb: Player / Entity environment interactions, collisions
- David: Random Map Generation

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

- Set up a game template with a consistent directory
- Set up the entity registry, and the component containers

Week 2

- Two rendered, navigable, connected, prebuilt rooms
- Basic Player Controls (WASD)
- 1 rendered jellyfish enemy (described in Figure 38 in Technical Elements) inside the second room
    - Sprite will just be a rectangle for this milestone
- *Creative 1 Basic - Camera control*
    - Rendered character that can move and the camera is centered around
- *Creative 2 Basic - Basic physics*
    - Implement ability to fire a kinematic harpoon projectile (also an entity) in a straight line
- Oxygen system - decreases based on movement / attacks / being attacked
    - Colour changes when oxygen is low (done by smooth interpolation)
- Make spreadsheet of known bugs
- Aforementioned entities will also have collision detection and placeholder, simple sprites.
- If time permits, implement the character dash with all the mechanics pertaining to it

Plan B

- Reduce to one rendered room
- Push dash mechanic to next milestone

Task Assignment:

- Tony: Movement, Player Controls, Basic Physics (Oct. 4)
- Andy: Oxygen (October 2nd), Help with rendering
- Bob: ECS Framework (October 1st), Enemies (October 4th)
- Brandon: Basic Rendering and Interpolate (October 4th), Full Rendering (October 5th)
- Caeleb: Player / Entity environment interactions, collisions (October 4th)
- David: Room Framework (October 1st), Pre-Built Rooms Ready to be Rendered (October 3rd)

Exact Plan B and Task Assignments for future Milestones not determined this far in advance

General Plan B options discussed in Advanced Technical Elements

General Responsibilities discussed in Team Management

Milestone 2: Minimal Playability

Week 1

- Simple enemies have been implemented, such as the fish, sea-mine, jelly-fish, shark
    - Enemies will take damage from player projectiles
    - *Creative 3 Basic - Simple Path Finding*
        - Enemy pathfinding AI will be implemented using BFS to find the closest path to reach player
    - Player will lose oxygen from collision with enemy or collision with enemy's projectile
- Mini Boss (Bound to a room)
    - Mini Boss AI will be implemented to actively fight the player in the room
- The game should support at least one objective for rooms, likely keys, since it is the simplest.
- The game supports an FPS counter on the top right, as well as very simple basic instructions that can be brought up on demand upon a keyboard press.
- Implemented pause button and screen
- Address bugs and determine any that can be fixed this milestone

Week 2

- *Creative 4 Basic - Basic integrated assets*
    - The currently implemented enemy and characters have fully implemented sprites and animations
- Collisions between the enemies and players are implemented via meshes.
- The map generation is more complex and supports a variable number of rooms, as well a few aspects within them as mentioned in its appropriate section are randomized, namely enemy locations, key locations, geysers, drops, and doors between the map.
- Another objective is added.
- The test plan is updated with all newly found defects.
- A gameplay video is recorded.
- Optimized performance to maintain a playable 2-minute demo.
- Update spreadsheet of known bugs

Milestone 3: Playability

Week 1

- All aspects (behavior, appearance, AI) of more complex enemies should be implemented.
    - Enemies with advanced abilities
    - Swarming behavior around
- All aspects of the final boss are implemented.
- Most other objectives for rooms have been implemented.
- Start and end game screens implemented.
- *Creative 5 Basic - Audio Feedback*
    - Background music (normal, low oxygen)
    - Sound effect for movement, shooting the harpoon, and being hit by enemies
- *Creative 6 Basic - Story Elements*
    - The game's opening crawl has the story comic-styled cutscene.
- Darkness mechanic and puzzles added
- Address bugs and determine any that can be fixed this milestone

Week 2

- *Creative 7 Advanced - Physics-Based Animation*
    - Implement momentum-based movement for player, projectiles
- Random map generation now supports the new enemies, new objectives, and pickups, Random rooms should also now scale in complexity as rooms get later, and objectives should now appear in random rooms, not just for the doors that the current room opens.
- The test plan is updated with all newly found defects.
- Video report detailing all game features and bugs as per the rubric is recorded.
- Update spreadsheet of known bugs.

Milestone 4: Final Game

Week 1

- Final Balance Changes to enemies, consumables
- All enemies are implemented.
    - Advanced pathfinding behavior
- Advanced decision making for final boss
- Random map generation is now fully balanced from last sprint.
- Fix all remaining critical bugs remaining from the last sprints bug reports.
- Ensure enough content to last a 10 minute non-repetitive game.
- Implemented some aspect of a tutorial system, such that the game's goals are obvious to the player.
- Integrate third party external libraries and tools.
- Address and resolve all remaining bugs

Week 2

- *Creative 8 Basic - Balance Changes*
    - Check that weapon and consumables damage, oxygen capacity, and enemy strength are such that the game is challenging with an engaging pace
- *Creative 9 Advanced - Particle systems*
    - Particle emission system for glowing loot
    - Geyser bubbles
- The game robustly handles any user input; it should be impossible to crash the game by users just inputting random keys.
- Video report detailing all game features and bugs as per the rubric is recorded