Game Proposal: Stranded

CPSC 427 - Video Game Programming

Team Members:

Henry Yang 96756903

Tyler Won 21693981

Charles Plotado 71890545

Justin Li 24368623

Venus Lee 49830474

Cam Mayhew 72999170

Story:

Define the genre of the game, the theme, and the most basic game mechanics. Write down the background story; list major levels, game rules, and player goals. Based on these definitions, give a detailed design of a minimum scene of the game (e.g. a "screenshot" of the first level or map) with information about possible player operations, possible interactions with the scene and the UI, one example enemy, and other items you think necessary. You can combine the description with the concept art visuals (the next item). The level of details should be enough for the reader to play the simplest version of the game in one's mind.

The story begins with the player crash landing on an alien planet. During the descent, important parts of the ship are jettisoned throughout the planet. The player has the technical expertise to fix the ship, but it cannot do so unless all the spaceship parts are found. The goal is to find the parts, fix the ship, then take off. However, the planet is experiencing an atmospheric phenomenon that blocks the local sun's light. The player must meet their nutritional requirements and fend off whatever unknown threats waiting in the dark.

The genre of the game is a top-down 2D survival game. There is a constant fog-of-war surrounding the player, severely limiting their vision. Enemies are scattered throughout the map, hiding and waiting for the unsuspecting player. If the player is unfortunate enough to alert an enemy, they will not stop hunting the player until one of them dies.

The player is far from defenseless. They start with a basic ranged weapon which is sufficient for weaker enemies. As they explore further, the player will have the chance to loot more advanced weapons from the planet. The origins of these weapons are not known, but such gifts are not to be questioned. Thus, the player can scour the land without fear in search of their precious spaceship parts.

Time is of the essence. The player's ship has an emergency food supply, but it is limited. Finding all the spaceship parts before starvation is a must. Only a limited supply of food can be carried

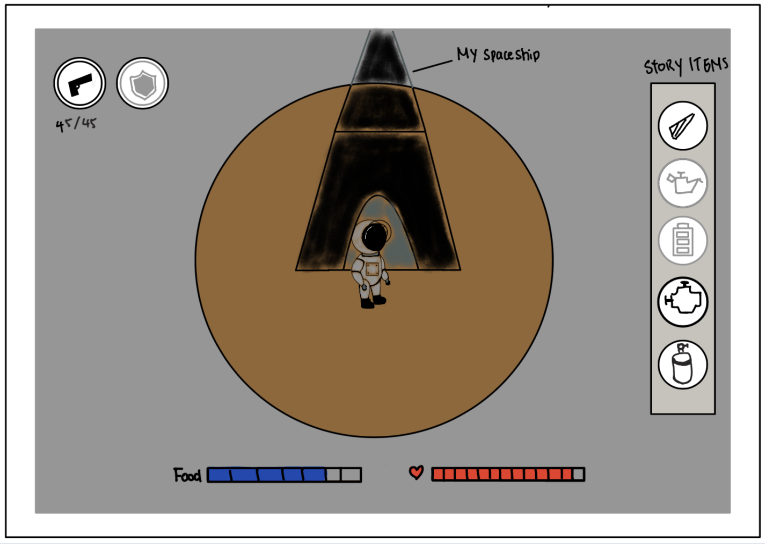
by the player when venturing outside the spaceship, so be aware of food consumtion! To restock on food, the player must return to the spaceship.

The following are some concept screens for the game.

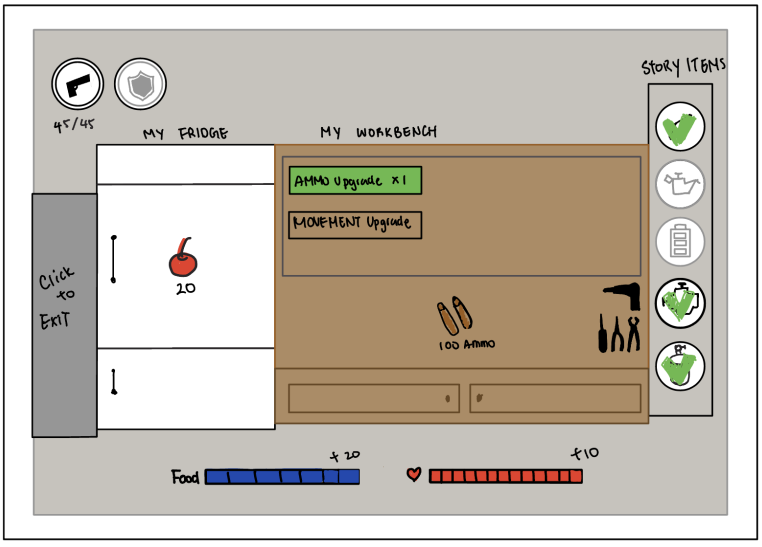
Concepts:

Produce basic, yet descriptive sketches of the major game states (screens). These should be consistent with the game design elements and help you assess the amount of work to be done.

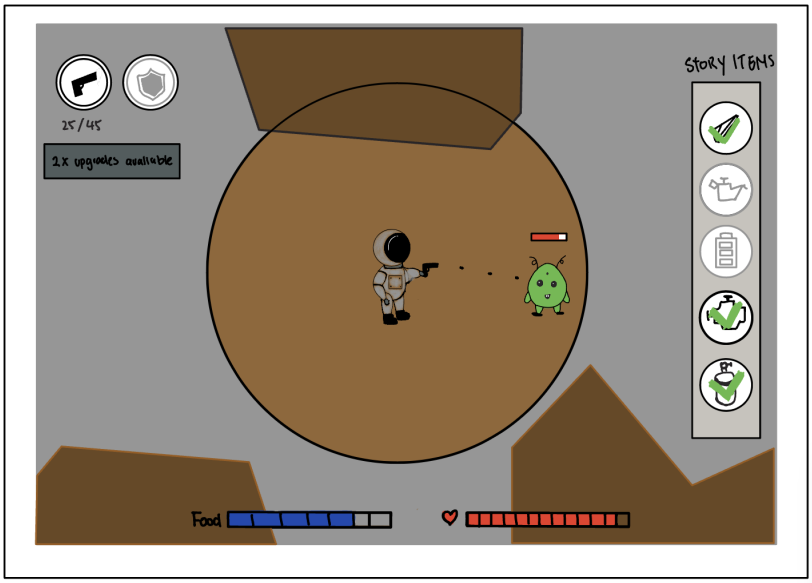
Game start: View when the game starts. The player starts in the middle of the map which is where the spaceship will be.



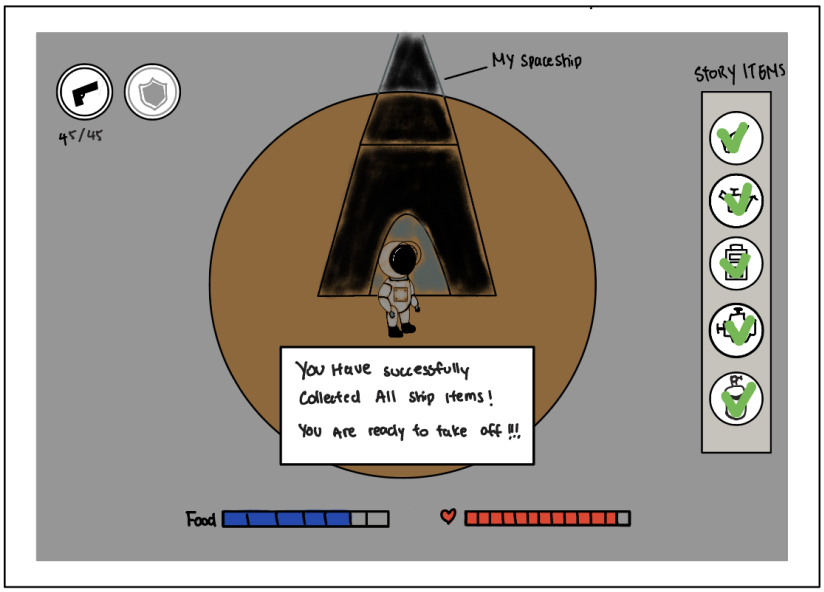
Spaceship interior: View when the player enters their spaceship. Inside the spaceship, there is a display of the amount of food, ammo, and upgrades the player has. The workbench box is interactable to allow players for upgrades. The player's health, food, and ammo is replenished upon entering the spaceship.



Exploration: View when the player is exploring the world. The player's vision is obscure by a fog of war. Various UI elements indicate the player's health, food, items, etc.



Victory: View when the player beats the game.



Technical Elements:

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

Rendering:
Graphics for character, background, loot, UI element. Basically everything on screen are done through rendering and updated at each game loop

In terms of detail, perhaps splitting into 3 layers in the order of rendering

- First render for the background layer which consists of non-interactable background terrain textures.
- Next is Entity layers like player and other intractable entities including mobs, loots, projectiles
- Last render the top layer which contains UI element like setting button, health bar, quest bar, weapon/armor display, base direction arrow, etc

2D geometry manipulation: 2D geometry manipulation such as vector transformation will be used in many parts of the game

- player, mob, project movement.
- Collision detection with bounding box for player, mobs, terrain collider, attacks(projectile).
- Assets creation
- Animation for player, monster, items loot on the ground etc.

Assets:
We will use simple shapes in the early stage of the development to represent necessary entities (players, mobs, etc) and switch on to self drawn sprites or free to use assets available online in the later stage of development. In the future, we may audio feedback and/or music to complement the user experience.

Gameplay logic/AI:
The number is subjected to change depending on progress of development. We plan to have 5 sets of monsters to cover 5 different zones across the map. The zone's boundary is not shown explicitly, however the player will realize it through change in monster's appearance and respected attack pattern.

Each kind of monster in a set has a different combination of attributes like movement speed/attack damage/detect range/health. One quest item to be spawned randomly within each zone. Players will be notified on screen when a quest item is within close distance. Monsters AI with different behaviors are programmed to follow fixed instructions or rules.

Physics:
Basic physics like velocity will be included in the game. This will be used for every moving entity, namely player controls and projectiles. Physics will be simulated per-frame and values are modified independent of frame rate by compensating for frametime.

Collisions are important since combat is a feature and we don't want the player to phase through walls. Since our game deals with fast-moving projectiles in weaponry, a sweep-based continuous approach to collision will be implemented to ensure consistency. This will iron out bugs detecting collisions involving high-velocity objects and/or high frametime conditions that are a problem in discrete solutions.

If time allows, a particle system may be added to enhance visual fidelity and user experience.

Advanced Technical Elements:

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

- Fog of war - Critical impact.
    - Can not be missed since this is a big core mechanic for our game
- Advanced pathfinding algorithm for monster - Low impact.
    - Missing this means monsters may not pick the most efficient route. Missing this would not affect the core gameplay.
- Particle system for better aesthetic on projectile: Low/Moderate impact.
    - Projectiles are not as flashy, hence more pressure on making a detailed sprite or other easier-to-implement alternatives like sound effects.
- Intro/ending cutscene - Low impact.
    - As an alternative we can just apply a black sprite to cover the whole screen and change the transparency of such a sprite to reveal the game screen. The ending cutscene is to display a sprite with text like “you’ve escaped!” when the player beats the game.

Devices:

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

We will be using both the keyboard and mouse as input for our game. Our primary controls will be the standard W,A,S,D keys - W to go forward, A to go left, S to go down and D to go right. Additionally, since we will have ranged combat in our game, we have decided to use the left mouse click button to fire our weapon. The left click button will also be used in game to pick up items, interact with the environment (for example inside the ship to complete a weapon/character upgrade).

Tools:

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

SFML (Simple and Fast Multimedia Library) for graphics, audio, and input handling
Undecided library that makes coroutines and/or async/await easier to work with.

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.

General workflow:

- Team members are split into pairs to work on assigned features
- A branch should be created for each feature
- Work for the feature should be done on the associated branch

- When a feature is complete, a pull request should be made. The pull request must be reviewed by at least one other team member before being merged into main

Project management:

- Use Github project task-board to keep track of and assign tasks
- Tasks can be in one of 4 states:
    - Todo: Work on the task has yet to be started
    - In progress: The task is currently being worked on
    - In review: Task is awaiting review (i.e. open pull request)
    - Complete: The task is complete, changes have been merged to main
- Tasks will be assigned an appropriate due date to ensure they are completed on time
- All tasks pertaining to a milestone should be completed at least one day before the milestone due date to allow for features to be integrated

Weekly meetings:

- Weekly meetings have been set for Wednesday @ 1:15pm and Saturday @ 10:30am

Development Plan:

Provide a list of tasks that your team will work on for each of the milestones. 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). These should be consistent with the course milestones but can pre-empt those.

Milestone 1

Create the user-controlled character

- Character should appear as a simple shape (ex. a square) on the screen
- User should be able to control the movement of the character with the keyboard (WASD). Camera should follow the character's movement (basic creative component 1)
- Character should exhibit correct collision detection and avoidance with game boundaries, mobs, etc.
- **Character should not be able to pick-up items or attack**
- Character's vision should be obscured by fog of war

Create the spaceship

- Spaceship should appear as a simple shape (ex. a rectangle) on the screen
- Spaceship should be located at the center of the map
- User should not be able to to interact with the spaceship

Create the map

- **Map should be completed up to "zone 1"**

Create some items

- **"zone 1" character weapon and defense items should be implemented**

- At least one character upgrade, food, ammo, and story item should be implemented
- Some items should be spawned in random locations on the map on game start
- All items should appear as simple shapes, but different items should be distinguishable from one another

Create some mobs

- **“zone 1” mobs should be implemented**
- **“zone 1” mobs should be spawned in random locations on the map on game start**
- Mobs should move but not attack. Mobs should track the character with a simple path finding algorithm (basic creative component 2)
- All mobs should appear as simple shapes, but different mobs should be distinguishable from one another

Milestone 2

Improve the user-controlled character

- The user should be able to pick-up items with the keyboard (ex. press E) and aim and attack with the mouse (mouse cursor indicates direction to attack, left-click to attack)
- The character should have its own designated sprite with animations for picking up items and attacking

Improve the spaceship

- Character health, food, and ammo should automatically regenerate when the character interacts with the spaceship (i.e. user presses a designated key on their keyboard when close to the spaceship)
- The spaceship should have its own designated sprite

Create the UI

- UI elements for character health and food, character gear (i.e. weapon and armor), story items should appear on the screen
- A basic tutorial should occur on game start

Expand the map

- Map should be completed up to “zone 2”

Add and improve items

- “zone 2” character weapon and defense items should be implemented
- More character upgrade, food, ammo, and story items should be implemented
- Items should still spawn in completely random locations on the map on game start
- Some items should be updated to have their own designated sprite with animations

Add and improve mobs

- “zone 2” mobs should be implemented
- Mobs should move and attack. Mobs should track the character with an advanced path finding algorithm (advanced creative component)
- Mobs should still spawn in completely random locations on the map on game start
- Some mobs should be updated to have their own designated sprite with attack and movement animations

Milestone 3

Improve the spaceship

- User should be able to enter the spaceship on interact (i.e. after they press a designated key on their keyboard)
- Character health, food, and ammo regeneration should occur when the character enters the spaceship
- Story items should automatically be turned in when the character enters the spaceship
- User should be presented with a static screen when inside the spaceship. The screen should display the amount of food and ammo stored in the ship. There should also be a workbench that the user can interact with to apply character upgrades they picked up while exploring

Expand the map

- Map should be completed up to "zone 3"

Add and improve items

- "zone 3" character weapon and defense items should be implemented
- More character upgrade, food, ammo, and story items should be implemented
- Items should be spawned pseudo-randomly
    - Character weapon and defense items should only spawn in their appropriate zones (i.e. "zone 1" weapon and defense items should only spawn in "zone 1", "zone 2" weapon and defense items should only spawn in "zone 2", etc.)
    - At most one story item should be spawned in a zone
- Some items should be updated to have their own designated sprite with animations (basic creative component 2)

Add and improve mobs

- "zone 3" mobs should be implemented
- Mobs should be spawned pseudo-randomly
    - Mobs should only spawn in their appropriate zones (i.e. "zone 1" mobs only spawn in "zone 1", "zone 2" mobs only spawn in "zone 2", etc.)
- Some mobs should be updated to have their own designated sprite with attack and movement animations (basic creative component 2)

General improvements

- Game reloadability (basic creative component 1)
- Precise collisions (advanced created component)
- Optimize memory management

Post-milestone 3 Writeup:

Plan changes:

basic creative component with audio feedback.

Our game development mostly aligned with the proposal except with a few missing below:

Quest item does not get turned into the spaceship. Game does not have an ending state/screen.

No workbench to interact with in the spaceship, food and ammo replenish automatically, upgrades apply automatically for now.

Milestone 4

Finish the map

- Map should be completed up to “zone 5”, the last zone

Improve UI/UX

- Add game start screen/cutscene/text narrative (basic creative component 2)
- Add victory/defeat screen/cutscene/text narrative (basic creative component 2)
- Tutorial should be refined so that no verbal explanation is required to play the game

Add and improve items

- “zone 4” and “zone 5” character weapon and defense items should be implemented
- All character upgrade, food, ammo, and story items should be implemented
- All items should have their own designated sprite with animations
- Particle effects should be added to some items (ex. projectiles shot from weapons) (advanced creative component)

Add and improve mobs

- “zone 4” and “zone 5” mobs should be implemented
- All mobs should have their own designated sprite with movement and attack animations

General improvements

- Gameplay balancing (basic creative component 1)
- Sound effects should be added for items, mobs, etc. (basic creative component 3)
- Game optimization
- Bug fixes