Ambrosia

Game Design Document

Created by Tyler Hendrickson

Table of Contents

High Concept

Ambrosia is a first-person wave survival game with an emphasis on cooperative play. Your group of 1-3 players have crash landed on a planet full of ravenous beasts. Your main objective is to survive the various waves of enemies while also defending and collecting parts for the spacecraft.

Game Design

Story

A small group of highly-advanced food people (known as Grubbies) have set off on an excursion into space to find more materials to bring back to their home planet. Several weeks into their adventure, however, the spacecraft began to experience problems with the thrusters. As the power to their thrusters gradually diminished, the Grubbies quickly found themselves being pulled in by the gravitational force of a nearby, uncharted planet and inevitably crashing down on the surface.

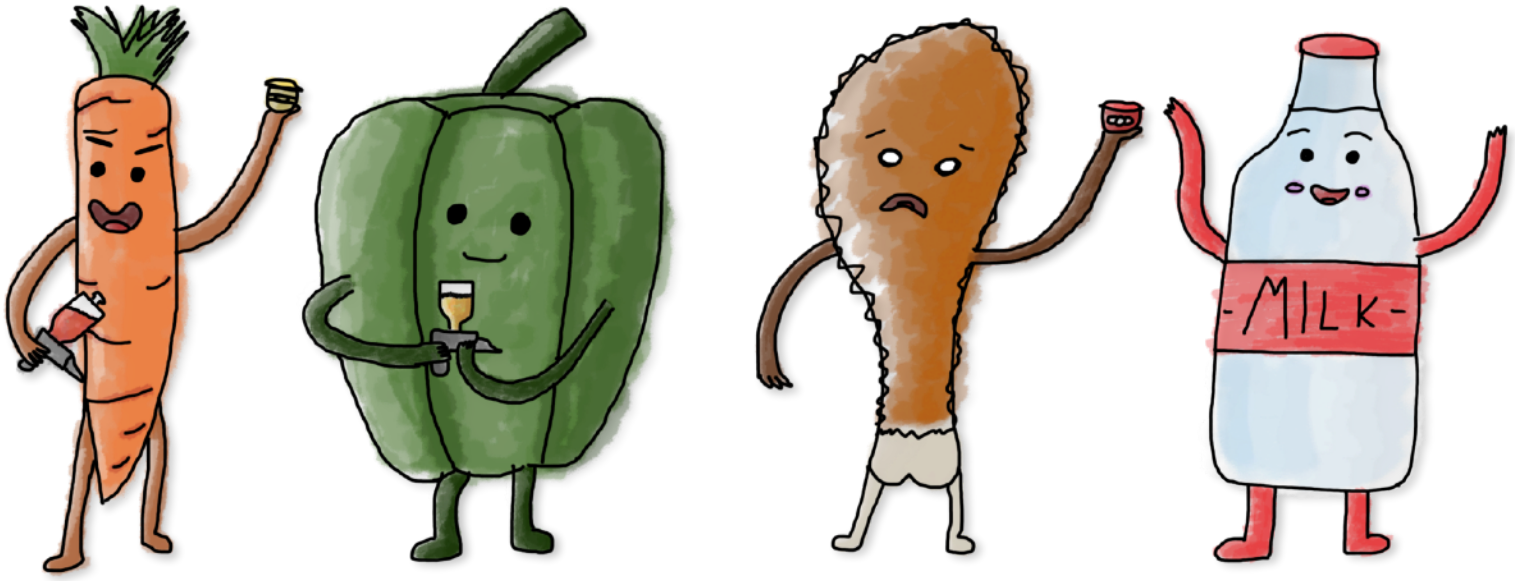
The Grubbies soon discovered that they were not alone on this planet. Inhabited by this planet appeared to be a species of ravenous, sickly beasts that would devour anything within their sense of scent. Realizing the threat these beasts imposed, the Grubbies quickly began to arm themselves and prepare for the fight ahead.

Characters

Grubbies

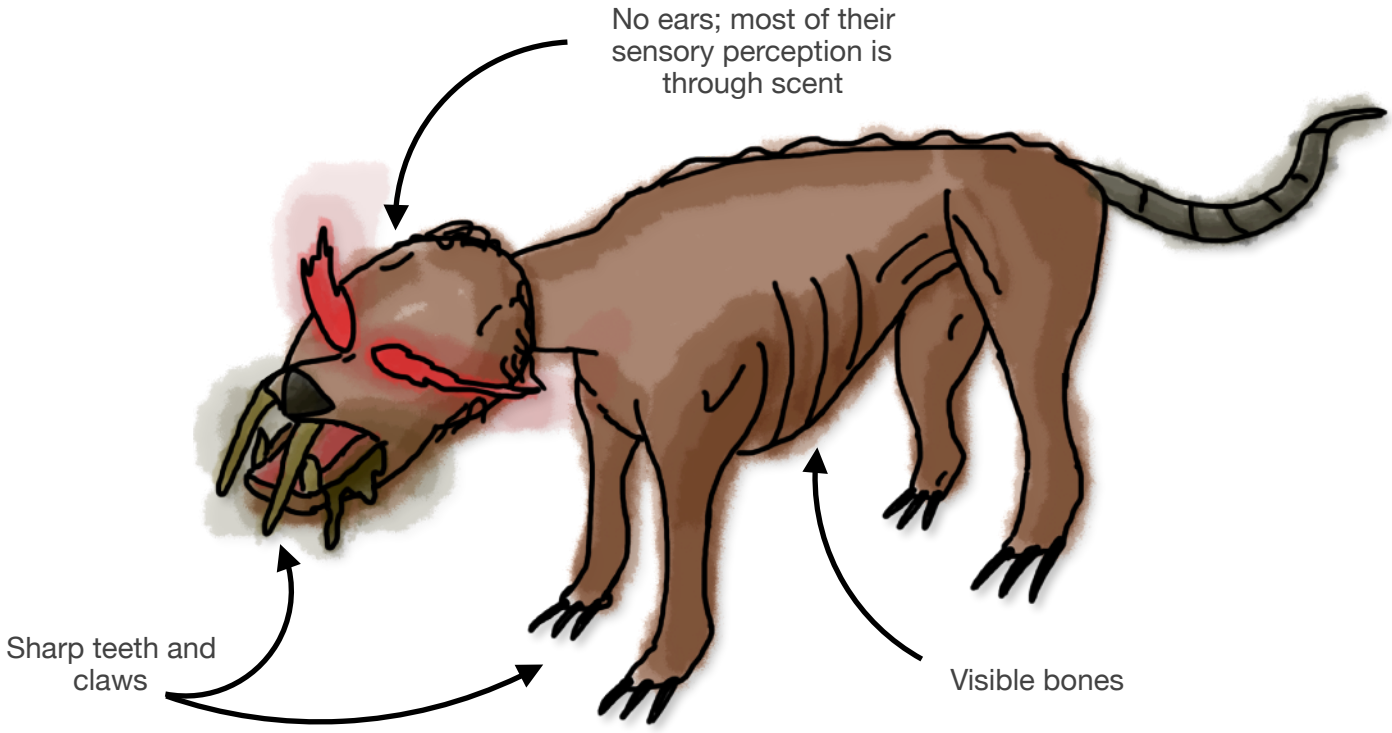
The playable characters in Ambrosia are a species of food-like creatures known as Grubbies. Grubbies are peaceful, docile creatures with an extreme intellect. They are often known to spend their time exploring the cosmos in search of new resources for their extravagant food dishes.

Despite their cute and friendly nature, they are not defenseless. Each Grubby utilizes the latest in culinary tech to blast their enemies into oblivion. While not the most athletic of creatures, they possess the ability to run, jump, and throw items as far as their noodle-like arms will allow. Their ability to operate as a unit, each bringing their own set of skills to the table, has allowed them to triumph over their foes time and time again.



The Tasteless

Upon landing on the uncharted planet, the Grubbies discover they are not alone—the planet is inhabited by a species of beast they affectionately dub “The Tasteless”. The Tasteless appear to be a species on the brink of extinction as their food supply is nearly non-existent. Each creature exhibits features of extreme malnutrition as well as razor-sharp claws and teeth.



Environment

The Environment in Ambrosia takes place on the home world of The Tasteless. Their planet is characterized by being extremely barren, rocky, and even volcanic. Despite this, there is a considerable amount of variety on the map itself. This is to help liven the visual tone a bit as well as give some spatial reference to the players in the game (one can call out “enemies on the shoreline” to teammates).

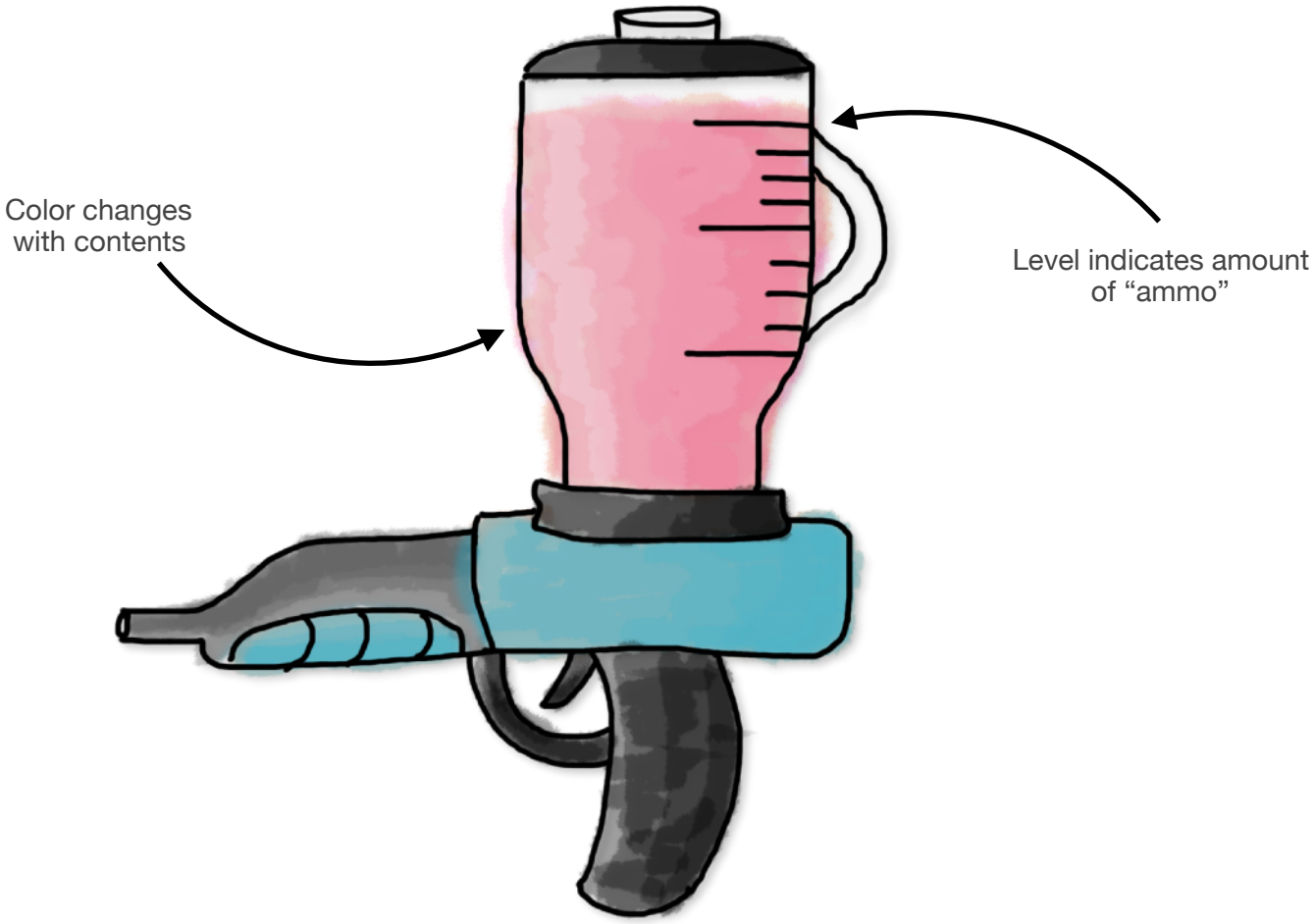
EARLY ENVIRONMENT SKETCH

Weapons

Main Weapon

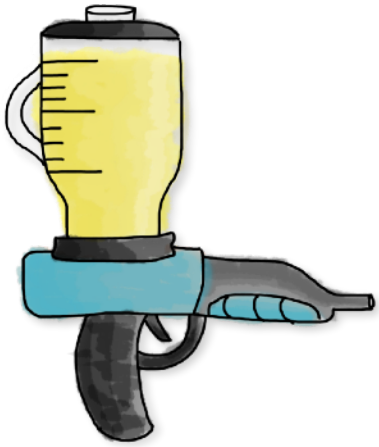
Blender Blaster

In Ambrosia, your primary weapon is the Blender Blaster. This weapon features a high-powered blender mounted upon a blaster that shoots the pressurized contents at the target. Rather than having conventional ammo, the source of power for the blaster are the contents inside the blender. The level of the contents indicate how much “juice” you have left and will slowly recharge when not in use.



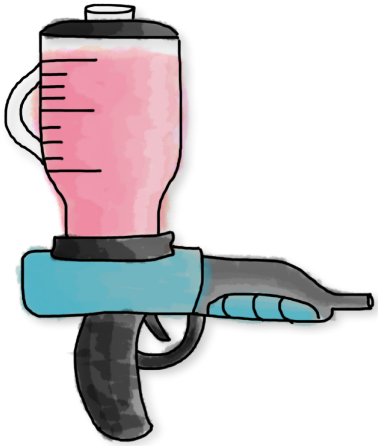
Properties

The Blender Blaster's properties are derived from the contents in the blender. Different contents will alter the aspects of the weapon (range, strength, regeneration rate, etc.). There are three different options for the Blender Blaster:



LEMONADE

Longest range ammunition. Low damage output, but doesn't deplete as quickly.



SMOOTHIE

Medium range ammunition. Strikes a good balance between damage and rate of depletion.

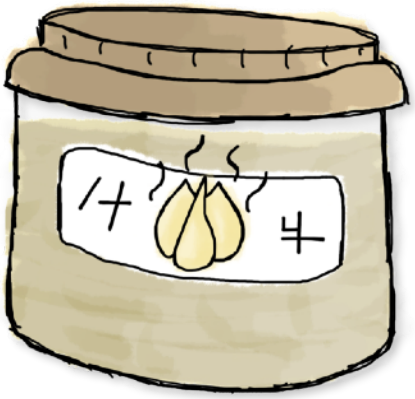
VEGETABLE JUICE

Short-range ammunition. Very high damage output, but also depletes very quickly.

Sub Weapon

Om Nom Bomb

As your sub weapon, the Om Nom Bomb acts like your typical grenade. When thrown, the Om Nom Bomb will detonate after a certain delay. Upon detonation, each Om Nom Bomb will perform a different effect (these are detailed below). Each player may only choose one type, so it's important to consider how each will benefit your team.



GARLIC DIP

Disrupts the senses of enemies due to its strong smell. All enemies will be drawn to the smell for a certain duration.

SPICY SALSA

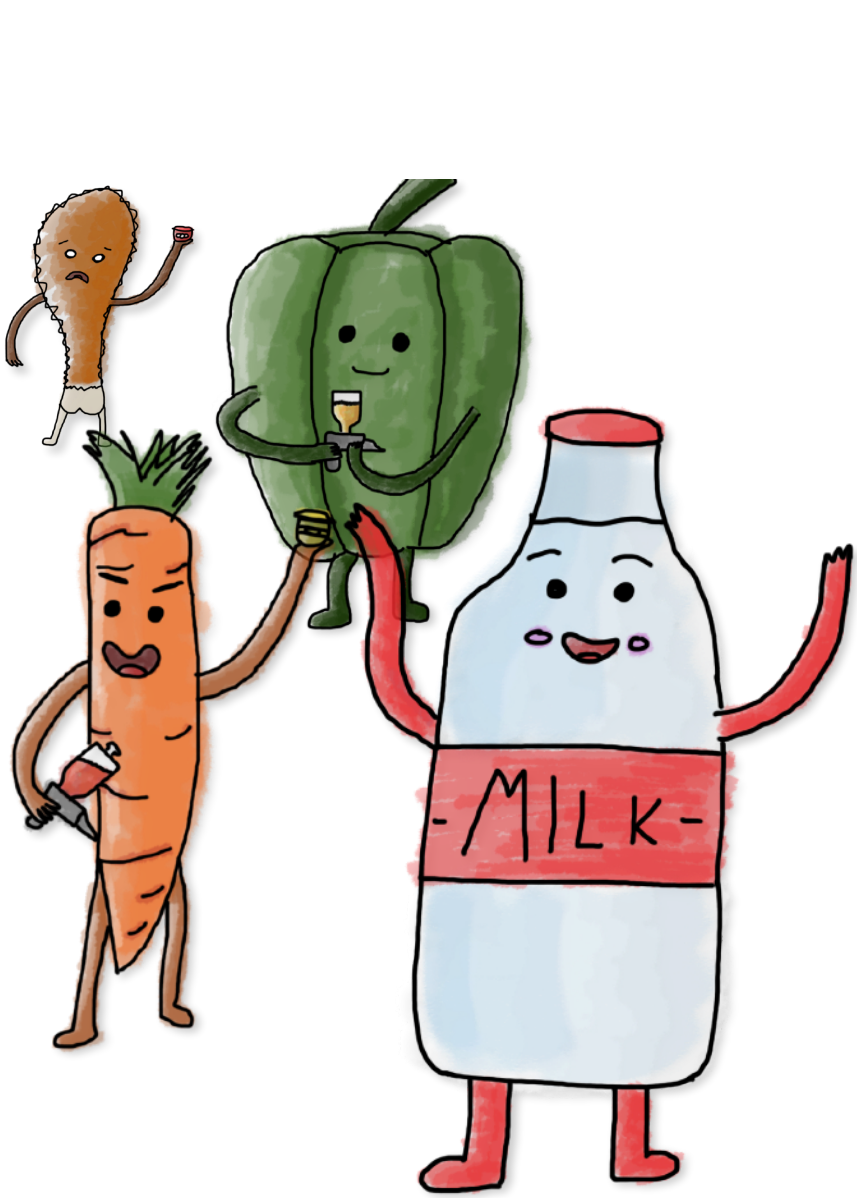
Detonates in a fiery explosion damaging nearby enemies and leaving a burn effect that deals damage over a period of time.

PURE HONEY

Creates a sticky area on the ground that will slow enemy movement and temporarily reduce their attack strength.

Art

The general art style of the game will focus greatly on **contrast**. The playable Grubbies, their weapons, and spaceship will all be inspired by food items and be very colorful, cartoony, and generally friendly looking. The enemies and their planet, however, will be menacing, sickly, muted in color, and more realistic.



Sound & Music

Sound

Sound effects will be relatively simple by design. These will focus more on giving the player **auditory clues** that enrich the gameplay experience.

Examples include:

- Audible “hit” sound to let you know that your weapon has hit an enemy
- Click or grinding sound to indicate no ammo or bombs
- Low grunting or growling sound to alert the player to the location of an enemy
- Crunching sound to let you know when you are taking damage
- Synth sounds when a player returns a ship part
- Explosive sounds for Om Nom Bombs
- Warning alert when spacecraft is under attack

A future version of Ambrosia may also be mixed in **5.1 surround sound**. With the proper equipment/device, this would allow for the player to have full spatial audio to help them locate things in their surroundings.

Music

The music in Ambrosia is still being decided upon, but it is anticipated that it will likely take upon an **ambient sound** rather than a catchy soundtrack. This will help to give the environment some atmosphere.

Additionally, Ambrosia may make use of **layered music tracks** so that the soundtrack adapts to the current game conditions. For example, when the player is in combat, the soundtrack will become more intense and taper off a bit when they are further from the action.

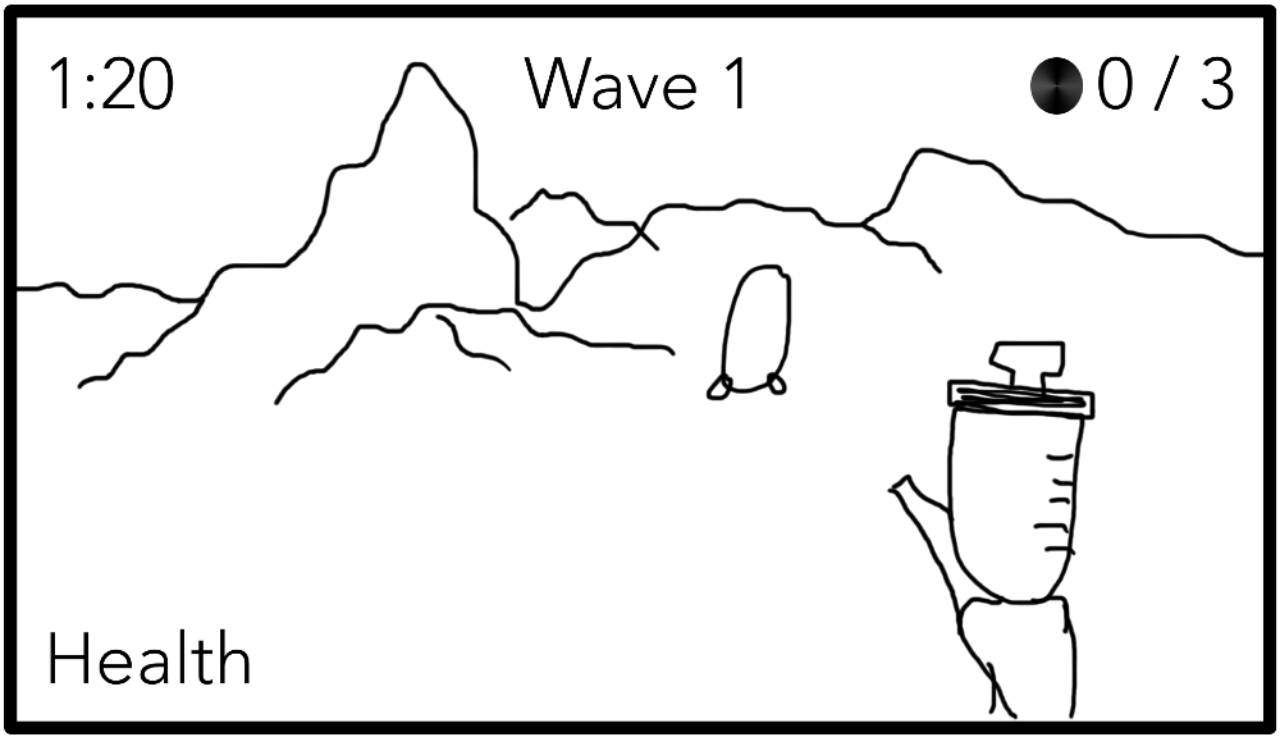
User Interface

Heads-Up Display (HUD)

Ambrosia's in-game HUD is anticipated to be relatively simple, but give players quick access to all the info they need.

The major components are:

- **Time remaining** - Time left until the current wave is over. Displayed in the top left.
- **Wave number** - The number of wave you are currently fighting. Displayed in the top center.
- **Ship parts** - The number of ship parts you have returned out of the total required to advance the wave. Displayed in the top right.
- **Player health** - The current health of your player. Displayed in a number in the bottom left. Damage progression will also be displayed using a visual border around the screen.
- **Current ammo** - The current level of ammunition for your weapon. Displayed using the in-game blaster model in the bottom right.

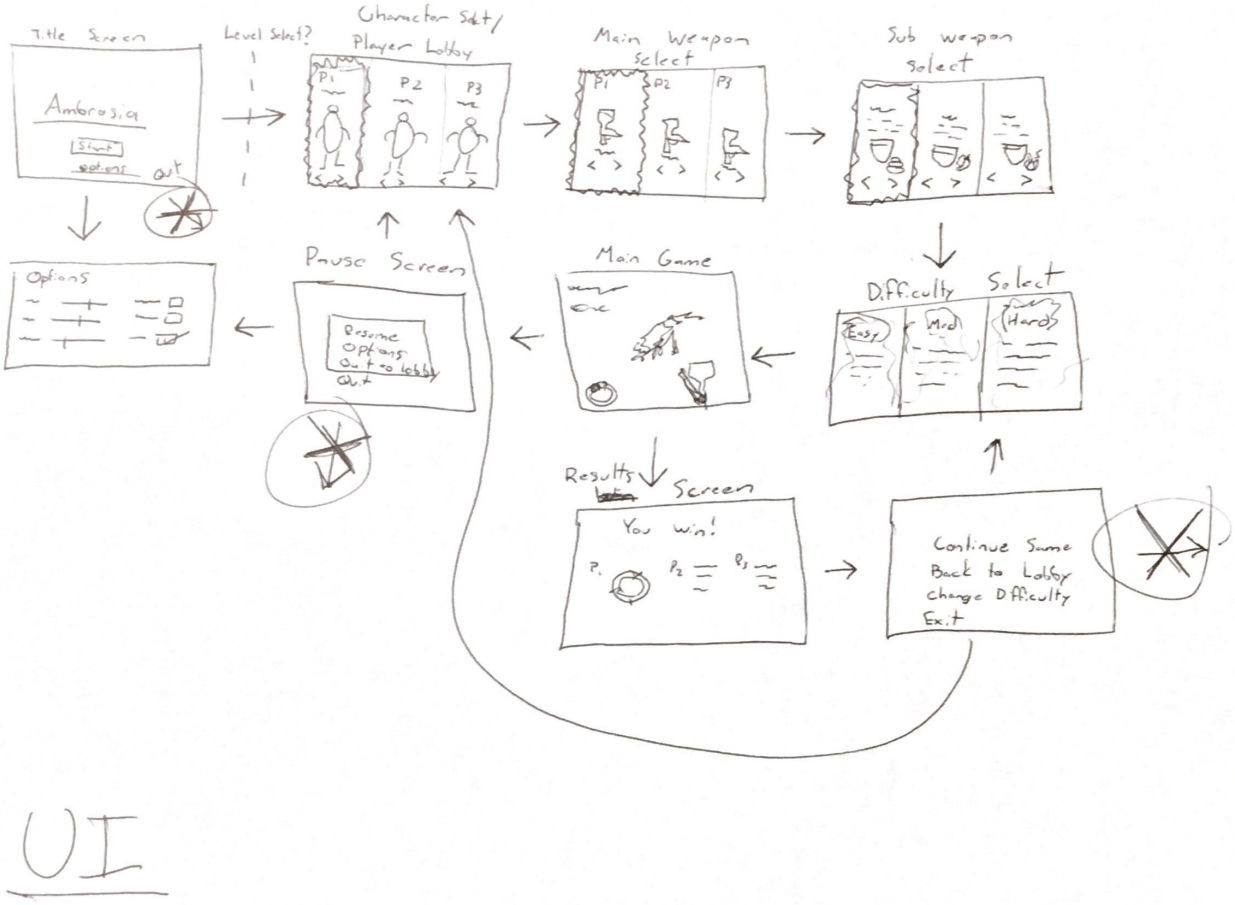


ROUGH MOCKUP OF IN-GAME HUD

• Menus

It is anticipated that Ambrosia will have the following different menus:

- **Title Screen**
- **Game Options**
- **Character Select / Player Lobby**
- **Main Weapon Select**
- **Sub Weapon Select**
- **Difficulty Select**
- **Main Game** (In-Game HUD)
- **Pause Screen**
- **Results Screen**
- **Results Screen** (With Options)



PROGRESSION BETWEEN DIFFERENT MENUS

Gameplay

Game Controls

Keyboard and Mouse

Look/Aim	Mouse/Trackpad Movement
Move	WASD or Arrow Keys
Sprint	Shift
Shoot	Left Mouse
Throw Grenade	E
Menu	Esc

Controller

To be determined…

Number of Players

To be determined…

Victory Conditions

Victory

To successfully complete a wave in Ambrosia there are a few conditions that need to occur:

- At least one player must survive to the end of the current wave
- The number of required ship parts have been deposited in the spacecraft
- The spacecraft has not been destroyed by enemies

Failure

If any of the following conditions are met, the wave (and game) will result in a loss:

- All players have been defeated
- Spacecraft has been destroyed
- Required ship parts were not collected

Software Design

Game Objects

Player

The player game object is the primary one you will deal with. This object and attached functionality will deal primarily with your movement, jumping, aiming, shooting, throwing bombs, and collecting items.

Scripts
- Player
- PlayerController

Enemy

The enemy game object is another one you will deal with the most. This object and attached functionality will deal with determining closest location to attack, attacking objects, and basic map traversal.

Scripts
- Enemy
- EnemyController

Ship

The spacecraft is a stationary game object. This object deals primarily with collecting ship parts from the player. It also has health attributes and can be attacked by enemies

Scripts
- Ship
- ShipController

Collectable

The collectables are just what they sound like. Their main functionality is allowing the player to pick them up and disappearing when that occurs. They also will do a check to see if the player is already holding a collectable

Scripts
- Collectable

Bullet

The bullet is a game object that is created as the player holds down the fire button. Multiples of these are spawned and each is able to determine whether it hits and enemy and does damage.

Scripts

- Bullet

Bomb

Like the bullet, the bomb is a game object that is spawned as the player inputs the command to throw it. After a delay, it detonates and hurts the enemies within range. It does not explode upon impact.

Scripts

- Bomb

Development Process

Overview

The development of Ambrosia occurred over the space of one semester and was broken down into four milestones. Despite working alone on this project, various tools were used to organize the process and to keep track of version history. Various development activities continued after the semester ended to further refine and polish the game for showcasing within a portfolio.

Tools

Perhaps the most important organizational tool used during the development process was the use of Trello. Trello allowed me to track all of the user stories and operate in an Agile-like manner.

In addition to this, both Bitbucket and Unity's built-in collaboration function were used for version control (redundancy is good, right?). This gave me the option to roll back changes if something were to go awry.

User Stories

Milestone 1

Milestone 1 was a sort of "test" in getting a player to move around in an environment. Basic elements were added to make it a game where tasks were required to win.

Milestone 2

Milestone 2 was the first basis of an actual game. User stories included:

- Creating a GameManager to manage the state of the game.
- Creating attributes for enemies, and player.
- Implementing enemies with basic AI behavior using a Nav Mesh Agent.
- Implemented shooting mechanic.
- Finalized 1st person view.

Milestone 3

The highlight of this milestone was implementing the wave mechanic. Other additions focused on adding lingering features from Milestone 2 and improving the environment. User stories include:

- Player is now only able to thrown one Om Nom Bomb at a time with a cool down between uses. This prevents the spamming of the bombs which would make the game too easy.

- Various missing reference errors were fixed regarding objects that had been destroyed (checking for null before accessing).
- Blender Blaster has a charge that discharges when firing and recharges when not in use.
- Spacecraft is now able to be damaged and destroyed by enemies, resulting in a loss.
- Enemies will stop moving when in range of a target.
- Multiple ammo types were created for the Blender Blaster in code. Inaccessible for now.
- Health counter was transitioned to a visual border to represent damage. This keeps the players focus on the action.
- Various code refinements and optimizations.
- Finished creating basic environment details.
- Implemented wave mechanic. Three waves of enemies spawn and there is a transitionary period between each wave.
- When the ship is under attack by enemies, a message is displayed to alert the player.
- Separated UI behavior from the GameManager.
- Implemented a UI bar to indicate the ship's current health.
- Designed HUD element to show the cool down of the Om Nom Bomb.

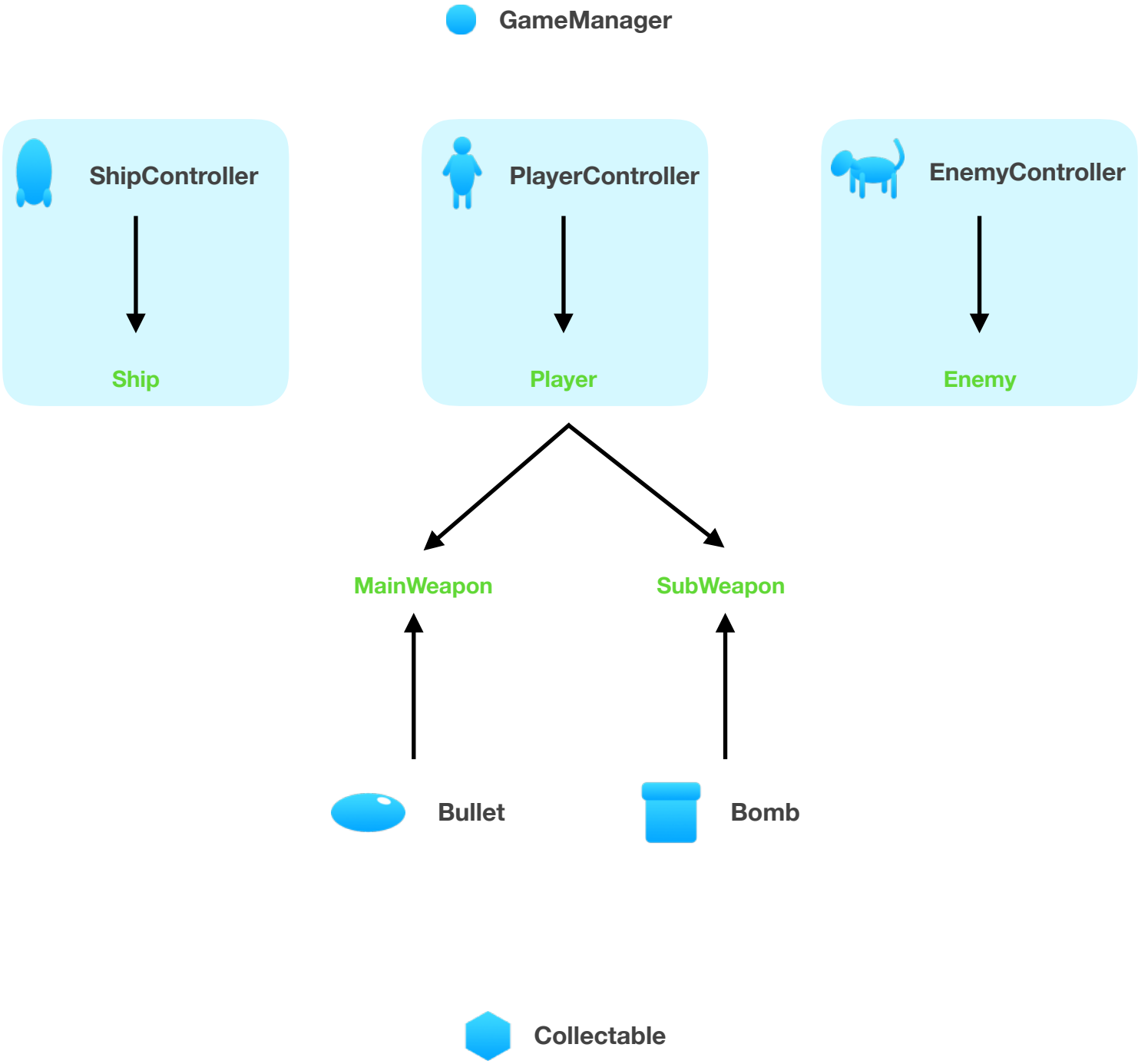
Milestone 4

This milestone was concerned with wrapping up all missing features and functionality and further adding polish to the game. User stories from this milestone are as follows:

- Implemented a HUD bar element indicating current level of ammo.
- Implemented ability to drown and included a status message explaining what happened if this does occur.
- Added invisible boundaries around the game world to prevent players from falling off the level.
- Instead of ship parts scattering the map, a spawning mechanic was introduced to spawn a ship part randomly on the map at specific intervals. More ship parts will spawn than are actually needed to advance the wave.
- Added much more detail to the model for the Blender Blaster.
- Fleshed out the environment with various rock formations.
- Completed rigging and animating the enemy models. Enemies will now walk when moving and change to an attacking animation when attacking the player or the ship.

UML Diagrams

Abstract UML Diagram



Playtesting

Feedback

Playtesting was performed based on a build from Milestone 2. Despite it being relatively simple at this point, a good amount of feedback was positive. However, there were multiple sentiments echoed across playtesters.

Playtesters commonly mentioned the following things:

- The game is too hard
- The map is too large and not interesting
- Couldn't tell when an enemy was attacking the ship
- No way to tell what the ships health is
- Hard to pick up ship parts (collider too small)
- Throwing bombs didn't seem to work
- Shooting randomly cut out
- Models are basic and lacking animations
- No sound effects or music

While most of these were things I was aware of, there were a few that I wasn't anticipating. For example, the collider being too small on the ship parts was something that I hadn't even thought of. I was able to address most of these in Milestone 3, but there are a few more that will require some extra time.

Changes

Difficulty

Because multiple playtesters mentioned that the game was too difficult, multiple changes were made to the game more accessible. These will be further refined as more feedback is given.

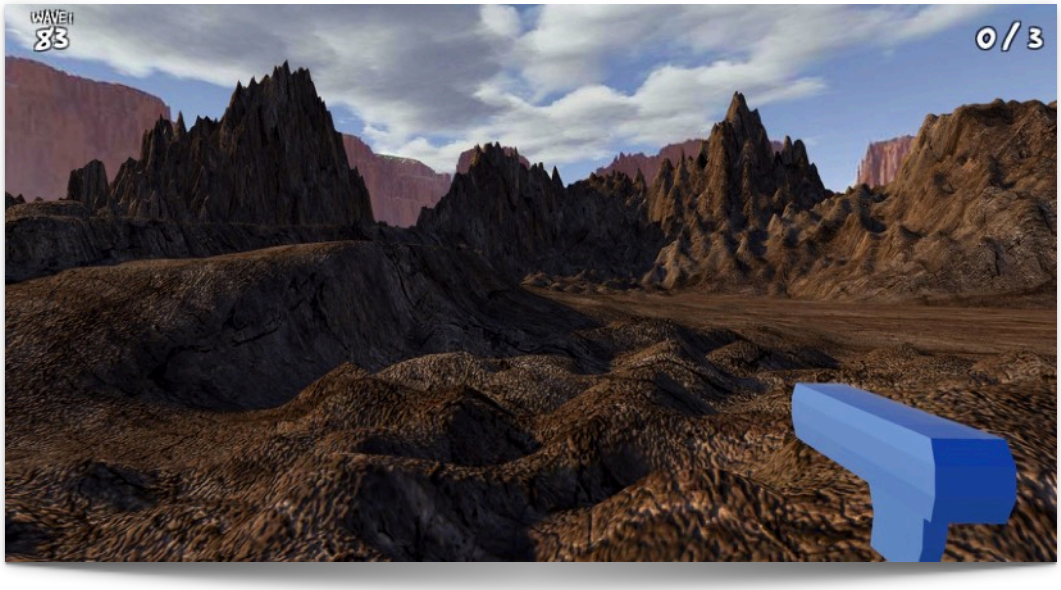
SHIP HEALTH	500	→	**1000**
ENEMY ATTACK	20	→	**15**
SECONDS PER WAVE	60	→	**90**
SPAWN DELAY	10	→	**15**
PARTS REQUIRED	3	→	**2**

Health Indicator

Because the health number in the previous build of Ambrosia was often hard for playtesters to see, health was transitioned to be represented by a frame around the screen instead. As you take more damage, the frame becomes darker and more apparent. This is similar to other shooter games and lets you keep your focus on the action.

Environment

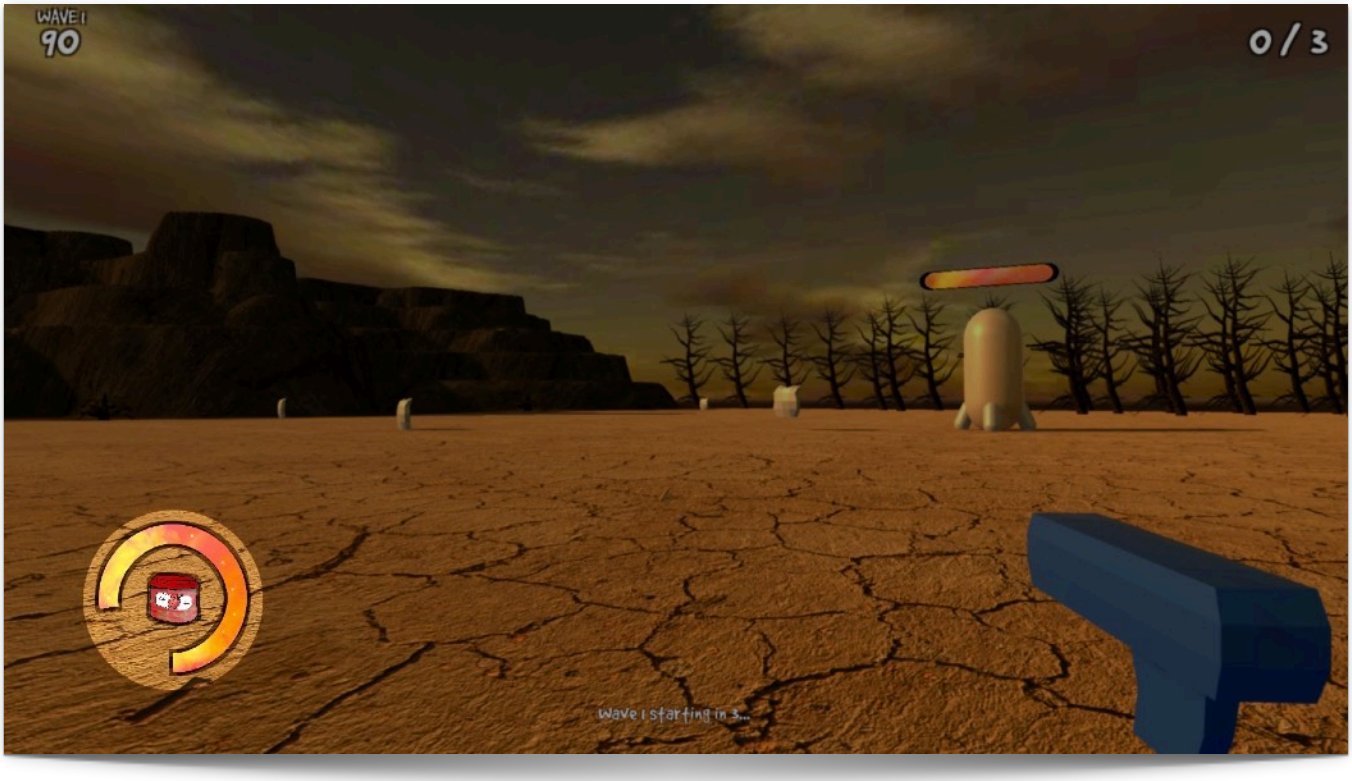
Another common complaint was that the environment was too basic. Playtesters described the world as “monotone” and “lacking in variety”. As such, the environment was completely overhauled with a greater variety of assets and terrain. Additionally, the map size was greatly reduced.



Bomb Recharge

In the build of the game provided to playtesters, you could only throw one Om Nom Bomb at a time and had to wait certain amount of time before you could throw another. This confused playtesters as there was no indication of how long you had to wait or when another bomb was available to be thrown. Some playtesters even thought that the bombs weren't working.

Due to this, a radial recharging indicator was added in the bottom left corner to show when you could throw a bomb again and what Om Nom Bomb you have equipped.



Ship Health and Alert

Playtesters explained that it was hard for them to tell when the ship was being attacked when they were out collecting ship parts or not looking at it. Additionally, it was impossible to see the current health of the ship, so it would often be destroyed to the surprise of the playtester.

In response, two features were included:

- A **health bar** above the ship showing the current health of the ship. This bar will orient itself to the player no matter which way they are facing.
- An **alert message** that displays whenever an enemy is attacking the ship.