IMGD 3000 Project 3

STUCK IN SPACE

Gabe Aponte & Henry Stadolnik

Design Document

NOTE: *Not much fundamentally changed from our initial plan to our final design. As such, this document is very similar to our original plan. However, the changes we did make are shown in red and anything that was removed from the game is struck out.*

Game Name: Stuck In Space

Team: The Visioneers
Gabe Aponte - gaaponte@wpi.edu
Henry Stadolnik - jhstadolnik@wpi.edu

Genre:
Wave-Based Arcade Survival Shoot-Em-Up

Game Description:
You are a pilot of a ship stranded in space, under attack from enemy aliens with no fuel left. Survive as long as you can against the horde using whatever ammo you can scrounge to power your arsenal of weapons!

Gameplay Summary:
The player is stuck in the center of the screen, unable to move. The player must aim their reticle at incoming enemies with the mouse and take them out before they can reach their ship. The player has several different weapons at their disposal, all of which use different types of ammunition. They must strategically switch between these to defeat their foes. In a pinch, the player can also overload their shields to take out all nearby enemies at the cost of some health.

Technical Features / Game Systems:
Procedural Wave Generation

- Randomly generate waves using “Difficulty Budget” system
  - Each enemy type has a difficulty value
  - The game chooses a pool of enemies to spawn each wave with a total difficulty value equal to the current Difficulty Budget value
  - ~~Waves may have a specific enemy type that appears in larger numbers~~
  - The Difficulty Budget value increases each wave
- The HUD displays the current wave number and messages for when waves start or end

Enemy Behavior

- Enemies have health that depletes when attacked, and get destroyed when they run out.

    - Most enemies have just a single hit point. “Tough” enemies have three.
- Enemy sprites will generally be 7x7 or smaller with 4-8 animation frames. ~~Some may have a separate left and right-facing version.~~
- There are several enemy types with distinct sprites
    - Basic enemy - Moves towards the player at constant speed
    - Tough enemy - Has more health than usual but moves slowly
    - Fast Enemy - Moves quickly towards player but is a larger target
    - Tricky enemy - Zig-zags erratically in towards the player
    - ~~Spiral enemy - Homes in on the player in a spiral motion~~
        - The dragonfly engine was unable to provide us with an easy way to make enemies move in a spiral pattern by orbiting the player with decreasing radius. For this reason, we cut the Spiral Enemy type from the game.
    - Swarm enemy - Spawns in a group of 5 enemies that all fly towards the player
    - Shooter Enemy - Approaches the player but then stops at range, periodically shooting destructible projectiles

Player Weapons

- Fire with left mouse click
- HUD displays the weapon list, ammo for each, and the currently-selected weapon
- Pickup ammo drops by shooting them
    - A weapon’s name and ammo count briefly change color when they get refilled by an ammo pickup
- Switch weapons with number keys
    - Weapons can also be cycled with the mouse wheel, the A and D keys, and the left and right arrow keys
    - We added these options to make weapon switching faster and give the player more options for how to control the game
- The player has several different types of weapons, each with different specializations and utility
    - Missile - A basic weapon with infinite ammo and a medium cooldown
    - Laser - A fast, narrow, linear shot that pierces through enemies.
        - Tough enemies are specifically weak to lasers, and a single hit with one will destroy them regardless of their current health.
    - Spread - Fire four bullets in a spread around the reticle
    - Bomb - Fire a detonatable bomb that explodes on impact
        - Detonatable bomb traits (Newly added)
            - The bomb was updated to be able to also explode on command in addition to exploding on impact. Attempting to fire the weapon while another bomb already exists will instead cause the current one to detonate.
            - The bomb blast radius was also increased to make this weapon more effective.
            - Tough enemies are specifically weak to bombs, and a single hit with one will destroy them regardless of their current health.

    - Plasma - Create a large, slow projectile that passes through enemies and does continual damage while in contact with them
    - Rapid - A fast-firing weapon with tiny projectiles

Ammo Pickups

- Enemies have a small chance to drop an ammo pickup for a random weapon on death
- Shooting an ammo pickup refills the ammo of the weapon indicated by the letter on the pickup
- Ammo Quota that increases based on the wave (Newly added)
    - There is a quota of ammo pickups that are required to drop in a given wave, which slowly increases at higher wave count the final enemies are guaranteed to drop enough to meet the required amount.
- Guarantees that every ammo type will drop (Newly added)
    - To ensure that players are equipped for every scenario, we implemented a system where the game tracks, for each weapon, how many ammo pickups have spawned since one of that type. If this counter crosses a certain threshold for a weapon, it is guaranteed that the next ammo pickup that spawns will give ammo for it.

Player Shields

- The HUD displays current shield capacity
- Collision with enemies / enemy weapons deplete shield
- Game over when shield is at zero percent and the player gets hit once

Shield Overload

- Activate with spacebar
- Costs a significant portion of shield integrity (15% of maximum)
    - Costs 15% of your maximum shield integrity, or all the rest of if if the player has shields at a lower percentage
- Blasts away all foes on-screen
- Meant to be a last-ditch save if you're about to get bombarded
- The visual effect for this changes the background color of the screen and all drawn characters

Leaderboard System

- Enter name after game ended (the system supports capital letters and spaces)
- Tracks the number of waves survived
- Displays only the top 15 scores (instead of 25)
    - If your current score is in the top 15, it will appear highlighted in yellow.
    - If you score lower than the 15th top score, your score will be displayed at the bottom of the screen in red.
- The game reads the leaderboard text file on launch to populate the leaderboard
- After each game, add the results to the leaderboard file
- The leaderboard can also be accessed from the main menu

Screen Shake (Newly added)

- Since our version of Dragonfly was not built to support screen shake directly, we implemented our own version by creating a ScreenShaker object that the camera follows. By extending the boundaries of the game slightly beyond the player view, we were able to create the effect of the screen shaking by having this object move rapidly and then reset whenever the player is hit.

Game Loop (Title -> Gameplay -> Results -> Repeat)

- The game cycles between several different game screens
  - Title/Menu
  - Gameplay
  - Results/Leaderboard
- Players can advance through the different screens using simple menu navigation
  - (choose options by pressing the indicated key)

Dragonfly Engine Changes / Additions (Newly added):

To support our game and make development easier, we added some new features to Henry's version of the Dragonfly Engine.

- Expanded the color palette to 34 colors
- Added a deleteObjectsOfType() method to the WorldManager
- Added setAnimationState() and setAnimationIndex() methods to Object, allowing direct control of an object's Animation without having to directly reference and replace it
- Added a new Solidness state: TANGIBLE, which can receive collisions but not initiate them and won't act as a wall. Using this state wherever possible can significantly reduce the number of times the engine checks objects for collisions each step.
- Added a new Object property: m_always_collide, a boolean which when true forces the engine to always check the object for collisions, even if it isn't moving. This is useful for stationary hazards or handling collisions with TANGIBLE objects.

Artistic Assets and Details:

Colors:

- Our version of Dragonfly supports 34 colors, of which we use about 20

Window:

- 120x36 characters, 1024x756 pixels
- Black background (changes to teal while using the shield overload ability)

Sprites and Visuals:

- All are custom-made. These ended up almost all being larger than originally anticipated.
  - Player ship (15x5 with 5 code-controlled animation frames; with two recolored copies for taking damage and losing shields)
  - Weapon projectiles
    - Missile (2x1 sprite without animation)
    - Laser (3 characters drawn in a line through code)
    - Spread (1x1 sprite with 4 animation frames)

        - Bomb (3x1 projectile sprite with 4 animation frames; 17x7 explosion sprite with 5 animation frames)
        - Plasma (11x5 sprite with 3 animation frames)
        - Rapid (1x1 sprite with 2 animation frames)
    - Enemy types
        - Basic (7x3 sprite with 8 animation frames)
        - Tough (8x3 sprite with 5 animation frames; with recolored copy for taking damage)
        - Fast (13x5 sprite with 5 animation frames)
        - Tricky (6x3 sprite with 6 animation frames)
        - Swarm (9x4 sprite with 8 animation frames)
        - Shooter (5x3 enemy sprite with 4 animation frames; 3x1 projectile sprite with 2 animation frames)
    - Ammo Pickups (5x3 object sprite with 6 animation frames and a weapon letter overlayed through code; 9x5 pickup effect sprite with 4 animation frames)
    - Death explosion (11x5 sprite with 7 animation frames)
    - HUD elements
        - Status readout (32x4 sprite without animation, with Shield Integrity and Wave Number overlayed through code)
        - Weapon readout (72x6 base sprite without animation; 7x1 selection sprite with 4 animation frames; weapon names and ammo counts overlayed through code)
    - Aiming Reticle (single character drawn through code)
    - Title Screen (73x22 sprite with 21 animation frames)
    - Game Over Screen (61x12 sprite with 25 animation frames)
    - Leaderboard (numerous lines of text drawn through code)
    - Twinkling stars (3 different 1x1 sprites with 1-4 animation frames)

Audio:

- Weapon sounds
    - 6 basic sounds (one per weapon)
    - Bomb explosion sound
    - Select weapon sound
    - Error sound (for when trying to fire a weapon without ammo)
- Ammo pickup sound
- Enemy sounds
    - Enemy death sound
    - Tough enemy hit sound
    - Enemy bullet fire sound
    - Enemy bullet counter sound (plays when destroying an enemy bullet)
- Player hit sound
- Game over sound
- Game start sound
- Menu select sound
- Menu confirm sound

- Shield overload sound
- Shield overload error sound (for when trying to overload shields without any left)
- ~~Background music (optional)~~
    - Deemed unnecessary, so it was removed

Implementation Plan:
We implemented everything ourselves and drew our own sprites. As necessary, we modified our version of the Dragonfly engine to expand its functionality to better support our project. We utilized freesound.org for most of the audio assets, and credited them in the AssetCredits.txt file. We created a few customized sounds using Reaper and Audacity. We utilized GitHub for source control and collaboration. While we did use the saucer-shoot code as a reference tool, we overhauled most traces of it and did not implement any outside source code to build this project.

Distribution of Work:
Henry
- Sprites / UI
- Procedural wave generation
- Weapon systems (HUD, ammo, attacks)
- Audio editing

Gabe
- Leaderboard system
- Player / Shields (health)
- Enemies (Base class and child classes)
- Audio implementations
- Screenshake

Schedule:
Monday 11/30/2020 - Submit Plan, Setup Engine, and Setup GitHub
Tuesday 12/1/2020 - The Basics
- Basic player class
- Basic enemy
- Basic weapon

Wednesday 12/2/2020 - Procedural Waves & Leaderboard tech
- Tech for procedural generation
- Tech for saving leaderboard as text file

Friday 12/4/2020 - All Weapons and Enemies
- ~~All weapons implemented~~
- ~~All enemies implemented~~
- Half of weapons implemented
- Half of enemies implemented
- Ammo pickups
- Art not final

Saturday 12/5/2020 - Alpha Due at 11:59 PM

- Rest of weapons implemented
- Rest of enemies implemented
- Full game loop (can replay without restarting program)
- Basic menu art
- Basic audio
- Basic sprites

Wednesday 12/9/2020 - Final Playable Due at 11:59 PM

- Polished art
- Rebalancing
- Bug fixing
- Full audio

Thursday 12/11/2020 - Promotional Material & Presentation

- Image
- Description
- Video
- Powerpoint / Demo (for in class)