**Software Documentation**

**Poly Boom**

Garrett Hoepf

Dr. Blasi

CST-460

4/28/2024

**Table of Contents:**

**1 - Introduction**

* 1. Project Overview
  2. GDM Summary
  3. Technologies Overview

**2 – Planning and Milestones**

2.1 Concept and Scope

2.2 Milestones

**3 – Development**

3.1 MVP systems

3.2 3D art and assets

3.3 Animation

3.4 Audio

3.5 Code / Scripts

**4 – Version Control**

4.1 GitHub Repository

4.2 Local Project Backup

**5 – Future of the Project**

5.1 Immediate future

5.2 Cut Features to be added in the future.

**6 – Conclusion**

6.1 Learning outcomes

6.2 Final thoughts

* 1. **Project Overview**

Poly boom is a wave survival shooter built in the Unity game Engine. Game has an emphasis on interactive physics while the player fends off waves of enemies.

Key Features:

* First person perspective
* Low poly art style
* Physics objects
* Destructible buildings (only 1 implemented by Capstone submission date)
  1. **GDM Summary**

Game Objective:

* Survive.

Lose Condition:

* Player health reaches zero.

Items / Tools:

* Gun
* Rocket Launcher

Art Style:

* Low Poly

Audio:

* Music Synthwave tracks
* SFX audio realistic.
* Gunshots
* Rocket Launcher
* Explosion
* Melee impact

Platform:

* PC

Developers:

* Garrett Hoepf, Sole Developer
  1. **Technologies Summary**

**Unity Game Engine** – Development Environment

**Blender** – Open-source 3D modeling software. Used for models, animations, and texture materials.

**Audacity –** Audio editor for SFX

* 1. **Concept and Scope**

Inspired by simple FPS shooters like Call of Duty Zombies and Killing Floor. Conceptualization centered around adding something new to that already done formula. Physics was the decided mechanic. Gameplay Feedback is an important aspect of wave survival shooters as the core gameplay is relatively simple, physics was meant to add to this while gunplay took the forefront. Scope was originally meant to include progression system, but due to severe setbacks. Progression had to be dropped for the sake of refining other elements of the game before project submission. Conceptualization also considered skillset. I have all the skills to make the game I originally intended. I have experience with coding, Unity Engine as a whole, 3D modelling, animation, and audio editing. I wanted a project that would incorporate all of these skills in the final product. Ultimately, the project did do this, but the timeframe prevented me from bringing the project up to the quality I know I’m capable of.

* 1. **Milestones**
* Concept/Scope submission
* Testing Environment
* FPS Controller
* Enemy Behavior
* Shooting Mechanics
* 3D environment models/Level design
* Starter weapon animation
* Explosive weapon projectile and impulse mechanics
* 3D environment fractured models
* Explosion and fractured model interaction
* Explosion and Physics objects interaction
* Explosive weapon animation and FPS mechanics
* Enemy animation and attack behavior
* Audio design
* Menu Scenes
* UI
  1. **MVP Systems**
* MVP submitted 2/19/2024.
* Player controller was functional.
* FPS mechanics were crude but functional.
* Wave spawner was implemented.
* Enemy behavior was implemented.
* Level was built and concise, albeit small and none interactable.
  1. **3D art and assets**
* **Acquired Assets**
  + Level surface plane built with modular streets and sidewalks, acquired asset.
  + Parking Garage
  + Vehicles
  + Park Foliage and Trees
  + Zombie enemy model
  + Blood particle system
  + Explosive and smoke particle system
  + Zombie model
  + DISCLAIMER: all acquired assets were acquired legally and for fair use by the policy of the Unity Asset Marketplace.
* **Original Assets**
  + Weapons built, rigged, textured, and animated.
  + Player arms built, rigged, textured, and animated.
  + Apartment complex and small pillared building
  + Fractured environment models
  + Lamp and light asset
  + Miscellaneous small environment assets
  + Muzzle flash assets made by me.
  1. **Animation**
* AK styled weapon, all original animations
  + Fire
  + Reload
  + Idle
* Rocket Launcher, all original animations
  + Fire
  + Reload
  + Loaded Idle
  + Empty Idle
* Player arms, all original animations
  + All AK animations
  + All Rocket Launcher animations
* Zombie enemy, acquired animations.
  + Running
  + Attacking
  1. **Audio**
* All audio bases acquired; edits made to various clips.
  + Gun shot: pitch altered.
  + Rocket Launcher shot: repurposed and edited audio from jackhammer.
  + Zombie hit effect: unaltered.
  + AK reload: clip sped up and cut to match.
  + RPG reload: cut to match.
  + Explosion: unaltered base, logarithmic falloff altered.
  1. **Code/Scripts**
* **All Scripts will be included with documentation separate from this document.**
* Death\_menu.cs: Handles logic for restarting level or returning to menu upon player death.
* Enemy.cs: Handles enemy health and death behavior, as well as instantiating ragdolls.
* Enemy\_AI.cs: Pathfinding, animation control, and attacking behavior for enemies.
* Fire\_Rocket.cs: Instantiates rocket projectile with impulse force.
* FPS Controller.cs: Handles player movement
* GunSystem.cs: Modular script that can be applied to different raycasting weapons to alter properties within Unity inspector. Also allows player to damage enemies with weapon.
* Main\_Menu.cs: Handles all logic in the main menu.
* Model Destruction Swap.cs: Handles interaction between the players Rocket Launcher and destructible environments.
* Particle\_Kill.cs: Short script to control lifetime of a particle instance
* Pause\_Menu.cs: Handles all pause menu logic during gameplay.
* PlayerHealth.cs: Manages player’s health, ability to take damage, and calls post processing effects.
* Rocket\_Explosion.cs: Plays particle system associated with explosion and creates omnidirectional impulse force to simulate an explosion, also deals damage to enemies.
* Vignette\_Take\_Damage.cs: Controls post processing for damage.
* Volume\_Manager.cs: Controls master volume via slider.
* Wave\_Spawner.cs: Logic of enemy spawns, able to loop through multiple spawn points scattered throughout map and count total enemies spawned, and also when enemies are eliminated.
* Weapon\_Switching.cs: Allows player to switch between the weapons available to them with the scroll wheel or number keys.
  1. **GitHub Repository**
* [**https://github.com/CircularBox/Poly-Boom**](https://github.com/CircularBox/Poly-Boom)
* Contains most recent files for project.
* Also has 2 functioning build versions, more recent version are too big to store in a repo.
  1. **Local Project Backup**
* Project has backup on local machine, regularly updated to keep up with current version of the project.
* Located on separate storage device in the event a storage devices fails containing the project files

**5.1 Immediate Future**

* Development of the project will cease for the upcoming future.
* Project has seen constant development over the course of the past 6 weeks. Typically ranging from 25 to 30 hours per week of active development time.
* Development has continued at the detriment of my physical health in an effort to provide a presentable project by the deadline. Sleepless nights have become a regular over the past months.
* The project will be resumed at a later date, exact date has not been decided.

**5.2 Cut Features to be Added.**

* Weapon selection, a larger roster of weapons was originally intended. That’s why the gun system script is so customizable.
* Performance rewards, originally intended to be tied to weapon selection. Good performance would allow the player to unlock more weapons.
* Gibs system, an additional layer of player feedback, as the player deals damage the enemies would show visible damage, including being able to knock of parts of the enemies’ body.

**6.1 Learning Outcomes**

* Project planning: multiple weeks of planning were done before the project went into active development. A GDD was made, timeline was set (and changed), and deadlines were placed.
* Scope management: The scope of the game had to be revised significantly in the midst of development. Heeding the original scope of the project would’ve resulted in an unpayable mess of systems come the deadline.
* Version Control: The project was backed up in an online repository and a local backup to ensure project integrity should there be a critical failure.
* 3D Art: Was able to enhance my relatively amateur 3D skills in animation and rigging. As well as adapting models to better suit there intended functions in game.
* Unity engine: Dove into almost all aspects of Unity’s development environment, giving me valuable experience and problem solving within the confines of the tools available.

**6.2 Final Thoughts**

I am not satisfied with how the project turned out as of now. I know I can do better but given the severe time constraints I was working with I think I delivered the best I could. Given that I had only about 6 weeks of full steam ahead development time, all of which still had to fit within my already busy schedule. The project I’m presenting at the conclusion of this class is more of a proof of concept in my mind, I wouldn’t even consider presenting this as a ready for sale product. This project is not to be abandoned simply because the class is over. The project I set out to create is one that I really want to produce because it is a game that I know I would enjoy playing. The game is simply too shallow right now to fill that purpose though. I will be taking a break from development on the project due to how much it has negatively affected my physical health in the recent weeks. I’ve had 1 or 2 sleepless nights per week the past 2 months simply trying to manage and develop the project. Once I have fully recovered mentally, I will resume the project at my own pace, one that won’t deteriorate my wellbeing. Until then, Poly-Boom has ceased active development.