REWIND

TIME GOOBERZ 

The Team

Sam Beckley	Business Lead
Bailey Parker	Tech Lead
David Samson	Sound Design & Programmer
Pratyush Trivedi	Graphic Design & Programmer
Harry Cohen	Level Design & Programmer

The Story

Codename ‘The Janitor’ receives mission to clean the APL

Mission Fails when an explosion drops Janitor 100 floors below

Abandoned Elevators are out of power

Escape is possible only by locating batteries

Gameplay

Bullet Hell style twin-stick shooter

Distinct Levels with clear objectives

Secret Rooms to encourage exploration

Core Mechanic: Rewind

Player ‘Ghost’ created every time player leaves a room

Revisiting Rooms causes player ‘ghosts’ to spawn

Ghost Bullets are deadly

Weekly Process

Small Outside of Class Meetings to pair program

Merge ONLY During Class more easily resolve conflicts

Discuss Direction & Assign Features before leaving class

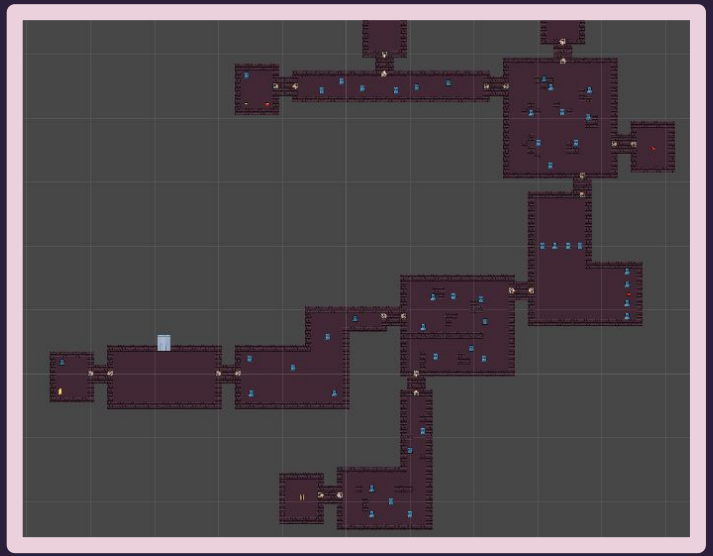
Process Lessons Learned

Fewer Full Team Meetings instead small pair programming meetings

Never Merge Code Outside of Class to limit potential damage

Playtest and Report Bugs Thursday to identify any critical bugs

Level Design



Scenes to facilitate editing all rooms in a level

Level Design

One Prefab per room for *time travel magic*

Level Design Lessons Learned

Playtest Informed Design using team members, friends and innocents

Playtest with Fresh Testers to help control difficulty

Hidden Rooms to encourage exploration and increase replayability

Rewind Focused each room is designed around core mechanic

Internal Design

Enemy Loaders to avoid the prefab of prefabs problem

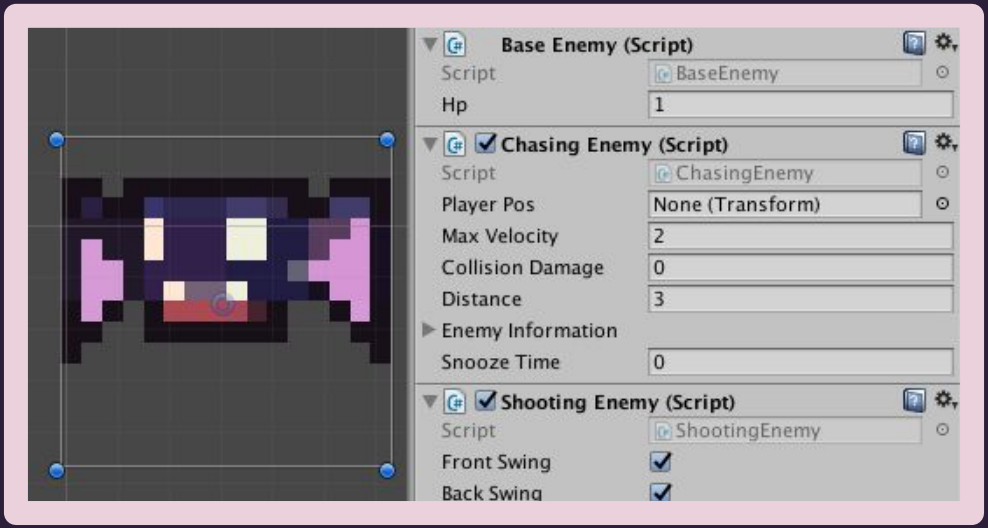
Unity Physics to facilitate faster iterations

Grouped Assets and Scripts for easier feature development

Modular Transition API plugs nicely into custom room management and interstitials

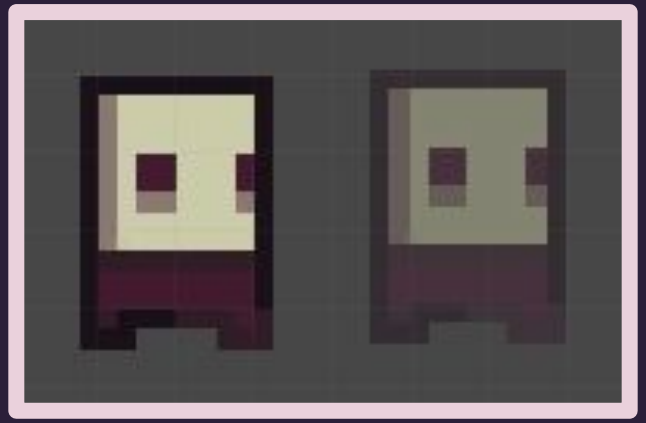
Overloaded Animation Controllers to standardize animations and maximize code reusability

Composition Pattern



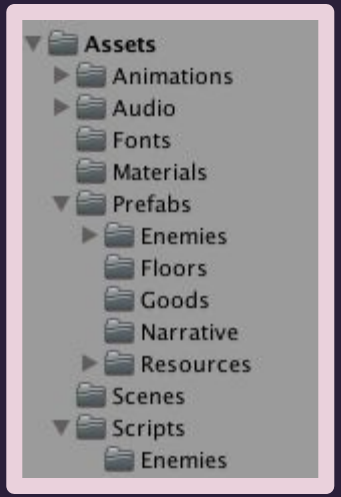
Composition with UI tunable parameters to maximize code reuse

Command Pattern



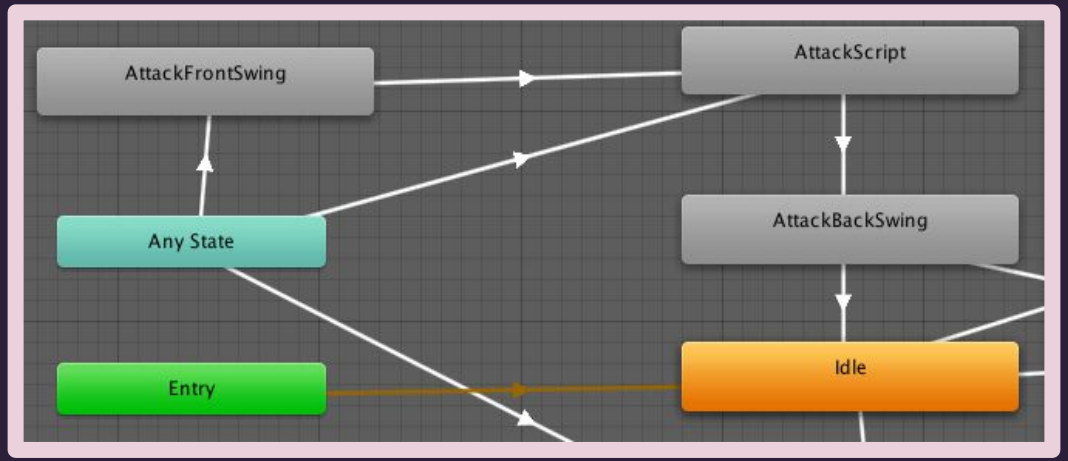
Command Pattern allows Player and Ghost to share code

Asset Structure



Recommended Unity Structure because if it ain’t broke don’t fix it

Animation Controller



Animation State Machine used by both code and animations to track entity state and trigger behaviors at appropriate times

Unity Lessons Learned

Go with the Flow doing things the Unity way is ultimately easier

Research Unity has a tool that does that, don’t build your own

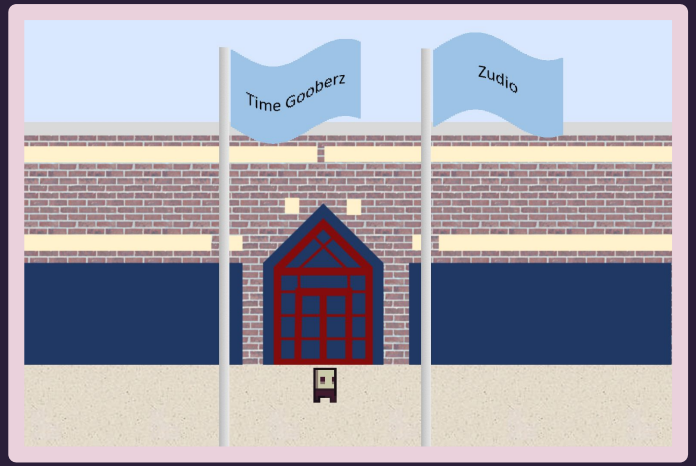
Control Who Touches What Prefabs Git and Prefabs don’t play well

Narrative Architecture

Scenes Handled Programmatically instead of as animations

Some examples:

- Waving Flags in Prologue
- Parallax while Falling
- Entering/Exiting Elevators



Sound Design

Custom Music by David

- Main Theme
- Level Theme
- Boss Theme

Sound Effect from open sources

- Player shooting
- Enemy shooting
- Enemy/player hits
- Enemy deaths
- Pick up items
- Unlock doors
- Regain Health
- Elevator sounds

Final Result

We made a fun game, should have focused on our core mechanic more

Game Design Lessons Learned

Cut Early to remove need to support vaporware features

Realistic Goals focus mental effort where it is needed

Playtest Constantly to inform current and future design

MVP ASAP find the fun and prune distractions quickly

Bug Fix Quickly to avoid bugs becoming features

Gold Demo

Questions?