Fast Forward

[Project: Bad Signal]

Lake Washington Institute of Tech - Digital Gaming and Interactive Media Bachelors Class of 2025

Revision: 1.3.2

GDD Written by: Alex ‘Krab’ Krabach
Assisted by: Maddie Richardson

Overview

- **Theme / Setting / Genre**
  - 3D Boomer Shooter
  - Retro vaporwave
  - Action/FPS

- **Core Gameplay Mechanics**
  - High-speed movement-based shooter
  - Time-limited levels encouraging fast completion
  - Emphasis on player flow and momentum

- **Targeted platforms**
  - PC - Mouse/Keyboard

- **Team Composition**
  - Bachelors Class
    - Programmers
      - Shain Allen (Lead)
      - Josh Abens
      - Ben O’Connor
      - Maverick Wilkinson
    - Artists
      - Alesha Baughman (Lead)
      - Ethan Berning (Co-Lead)
      - Ashe Krzykwa
      - Erin Hilliker
      - Ethan Berger
      - Ita Cummins
      - Maddie Richardson
      - Tyler Charnholm
      - Will Collins
    - Miscellaneous
      - Alex ‘Krab’ Krabach (Producer)
      - Quin Jesus (Sound Designer)
      - Todd Baagason (Advisor)

➢ Tools

- Unity
- Perforce
- Confluence
- Jira
- Visual Studio
- Maya
- Substance Painter
- Photoshop
- Blender

➢ The Elevator Pitch

Fast Forward is a high-energy, vaporwave-styled boomer shooter where style meets speed. Dash, blast, and flow your way through 80's themed digital ruins as you fight off enemies with seconds to spare.

➢ Project Description (Brief)

Fast Forward is a stylish, fast-paced shooter inspired by Neon White, Overwatch, and classic speedrunning games. Set in a nostalgic digital world with an 80s flair, players will blast through time-limited levels using speed, style, and precision.

➢ Project Description (Detailed)

Fast Forward aims to capture the thrill of speedrunning by building levels that reward precision, quick reflexes, and momentum-based traversal. Players will utilize advanced movement mechanics and a gun with alternative fire modes to collect additional time for completing the level. Gameplay emphasizes chain reactions and alternate paths for high replayability.

➢ Design Principles

- ➢ **Speed is a reward.** Players who master movement should feel fast, fluid, and in control. They should feel unbound by the limitations of traditional FPS maps.
- ➢ **Nostalgic Style.** Visuals should be readable and striking without overwhelming the player, cogniscent of the feel of the 80s Miami feel with a unique twist.
- ➢ **Failure teaches, fast.** Losses are brief, resets are immediate, and players always want to try again.

➢ Player Experience Goals

- ➢ **Feel powerful, not invincible.** Success comes from skill and flow, not brute force.
- ➢ **Stay in motion.** The player should almost always be moving forward — stopping feels wrong.

- **Know when you're good.** Level completion times, style bonuses, and alternate paths reinforce player mastery.

Level Design Philosophy

- Layouts are crafted with forward momentum in mind — players should rarely need to backtrack.
- Difficulty curves with player proficiency; early level teach by doing, while later in the level challenge through layered mechanics.

Story and Gameplay

➢ **Story (Brief)**

In an alternate 1980s, VR became mainstream thanks to a revolutionary console called the VisualGit. Over time, virtual features became embedded in everyday devices through a universal Compatibility Cord™, giving rise to entire digital ecosystems.

But as these systems expanded, so did the threat of malware. You play as Dizzy, a virtual repair specialist who enters corrupted devices to eliminate viruses before they wipe out their internal worlds. Her shop, Fast Forward, is the last line of defense between glitch-ridden collapse and recovery — but she has to work fast.

➢ **Story (Detailed)**

The VisualGit, developed by tech company Greeble in the 1980s, brought VR into homes decades before our timeline. Its signature feature, VisualHome, allowed players to earn and display digital furniture in fully customizable virtual spaces. The idea spread rapidly across devices, aided by the introduction of the Compatibility Cord™, which enabled users to transfer content between systems.

Over the years, VisualHome evolved into a cultural and economic phenomenon — one that also attracted widespread hacking. Users began unknowingly downloading malware disguised as in-game assets, leading to data corruption and complete system failure.

Now, with the release of the VisualU and its open marketplace, the problem has reached critical mass. Enter Rachel Scott, daughter of the VisualGit's creator and known online as Dizzy. Drawing on her father’s tech and her own expertise, she launches Fast Forward, a rogue repair service that digitizes her into infected systems. There, she battles aggressive viruses in real time to save users’ data before it’s lost forever.

➢ **Gameplay (Brief)**

A movement-heavy FPS where players complete levels by eliminating enemies and reaching the exit as fast as possible.

➢ **Gameplay (Detailed)**

Fast Forward blends combat and a movement playground to create exhilarating combat. Every level presents multiple paths to reward exploration and mastery. Players are equipped with a ranged weapon for precise shooting and for crowd control. Enemies and environmental traps encourage rapid thinking and quick traversal. The faster and cleaner the run, the better the score.

Assets Needed

- **2D**
  - Textures
    - (3) Environment Textures
    - Character Texture
    - Enemy Textures
    - Trap Textures
    - UI
    - Checkpoint Texture
  - Heightmap data (if applicable)
    - 25 x 14 grid
    - 32 x 32 per cell
    - 2 rows for HUD
    - Screen size: 800 x 448
    - Play size: 800 x 384

- **Sound**
  - Player SFX
    - Footsteps
      - Remember to find sounds for different surfaces.
      - Jumping
    - Wall Run
      - Feedback received - needs to sound less like a jetpack, more windy.
    - Slide
  - TV Enemy SFX
    - TV Enemy Ambient
      - Static, distorted old TV broadcasts.
    - TV Enemy Movement
      - Heavy plastic box, machinery clunking.
    - TV Enemy Attack
    - TV Enemy Take Damage
      - Heavy impact on plastic, metal, glass.
    - TV Enemy Defeat/
      - Glass breaking.
  - Boombox Enemy SFX
    - Boombox Enemy Ambient
      - Static, distorted old radio broadcast. Close to what the TV will sound like.
    - Boombox Enemy Movement
      - Heavy plastic box, machinery clunking.

        - Boombox Enemy Attack
        - Boombox Enemy Take Damage
            - Heavy impact on plastic and metal.
        - Boombox Enemy Defeat
            - Plastic, metal breaking.
    - Statue Enemy SFX
        - Statue Enemy Ambient
        - Statue Enemy Movement
            - Stone moving around.
        - Statue Enemy Attack
        - Statue Enemy Take Damage
            - Heavy impact on stone, concrete, etc.
        - Statue Enemy Defeat
            - Stone crumbling/falling apart.
    - Gun SFX
        - Gun Shooting
    - Saw SFX
        - Saw Arm
            - Use RTPC to slow down game audio just a bit to coincide with the saw mechanic.
        - Saw Use
            - Think DJ scratching on a turntable mixed with an electric saw. /
    - Music
        - Main Menu Music
            - Cyberpunk, Vaporwave vibes.
        - Level Music
            - Same with Main Menu Music.
    - UI
        - Menu Button Select
        - Menu Button Press
    - Misc.
        - Level Ambient Sounds
            - Machine humming.

- **Code**
    - Character Scripts (Player Controller)
    - Object Scripts (Runs in the background)
        - Checkpoints
        - Environmental Obstacles
    - Enemy Scripts

Animation

- Environment Animations
    - Checkpoint
- Character Animations
    - Player
        - Walking
        - Jumping
        - Shooting
        - Sliding
        - Reloading
        - Color transitions
- Enemy Animations
    - All required elements necessary for the enemy, such as running, jumping, teleporting, burrowing, shooting, clinging, etc.

Schedule

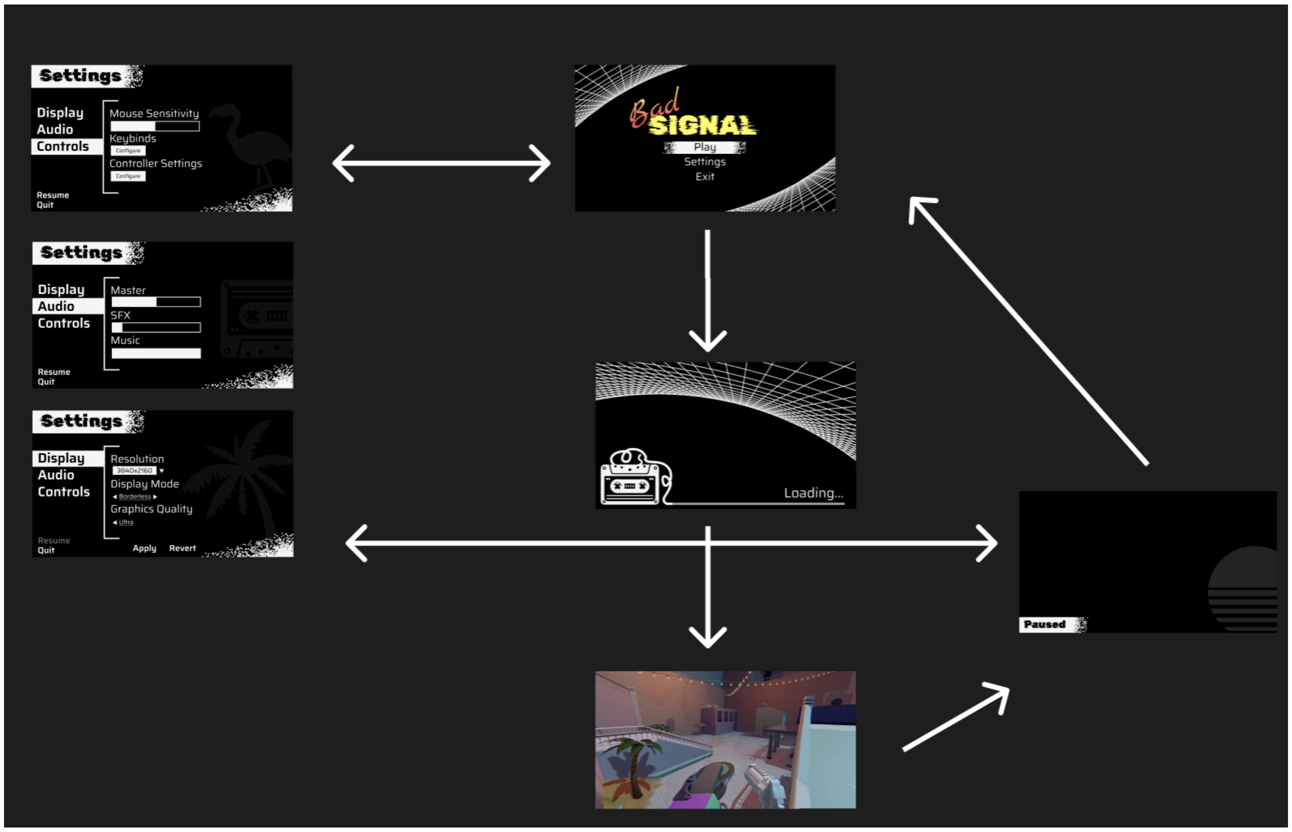
- **Preproduction**
  - Week 1 - 3
    - Create (3) reference boards and begin drafting concepts
    - Build prototype player controller
    - Research required structures and tools (such as state machines and Aseprite)
    - Assign Lead roles and artist and programmer tasks
    - Ensure Confluence and Perforce connectivity and usage

- **Production**
  - Week 4 - 6
    - Build player controller
    - Build enemy scripts
    - Draw and animate
      - Player character sprites
      - Enemy sprites
      - Environment tile sheets
    - Implement sound and UI
    - Build a test level

- **Postproduction**
  - Week 7 - 10
    - Extensive playtesting
    - Finalize sound and UI Implementation
    - UI, camera scale, and player controller fully tuned
    - Everyone has (1) level to submit made by themselves

UI and Style Guide

➢ **UI Flow and Mockups**



➢ **Style Guide Link**

➢ [Bad Signal - Style Guide - Google Docs]()

- Created by Alesha Baughman and Ethan Berning

Post Mortem

- **Post Mortem Summary**