Splat_II

Date: April.14/26

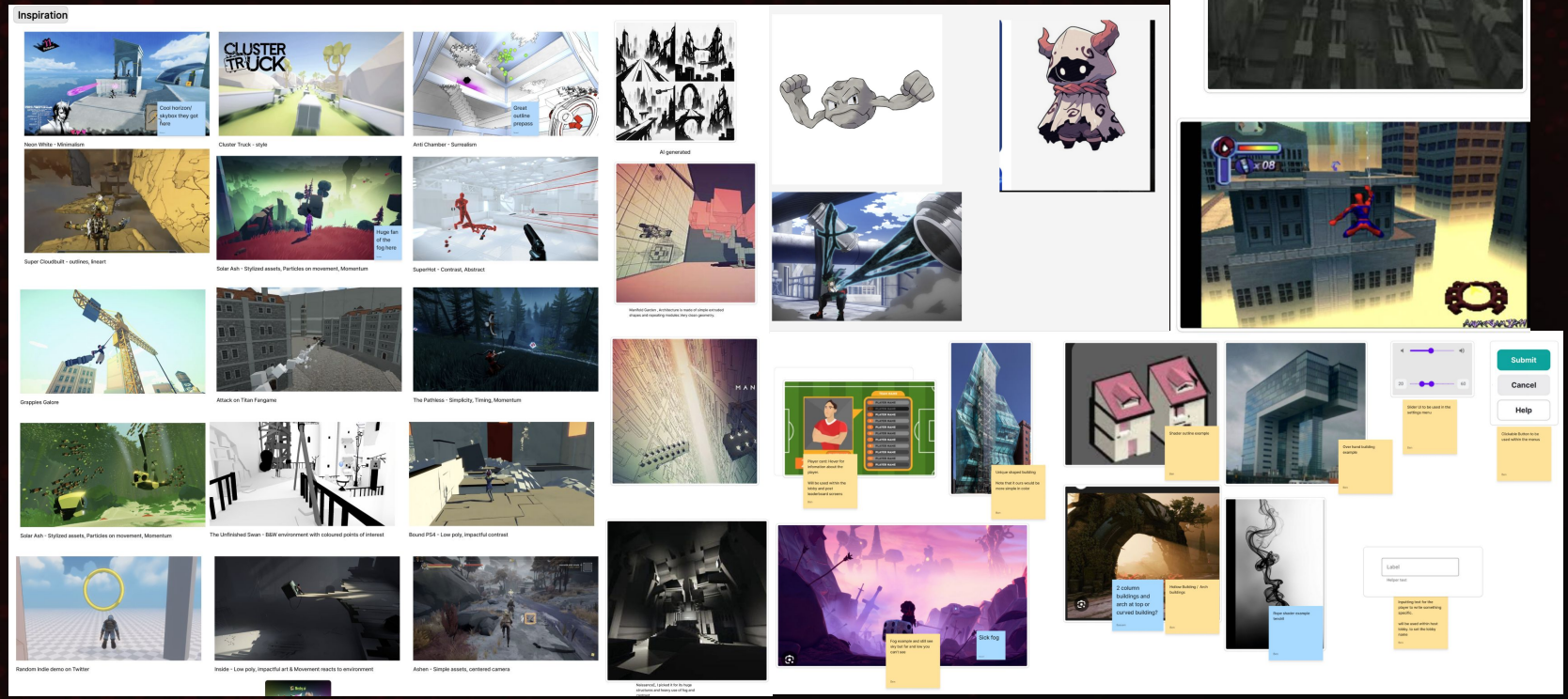
Elevator Pitch

Splat_II is a fast-paced racing game where players use grappling hooks to swing through a procedurally generated city, competing to reach the finish line in the shortest time possible.

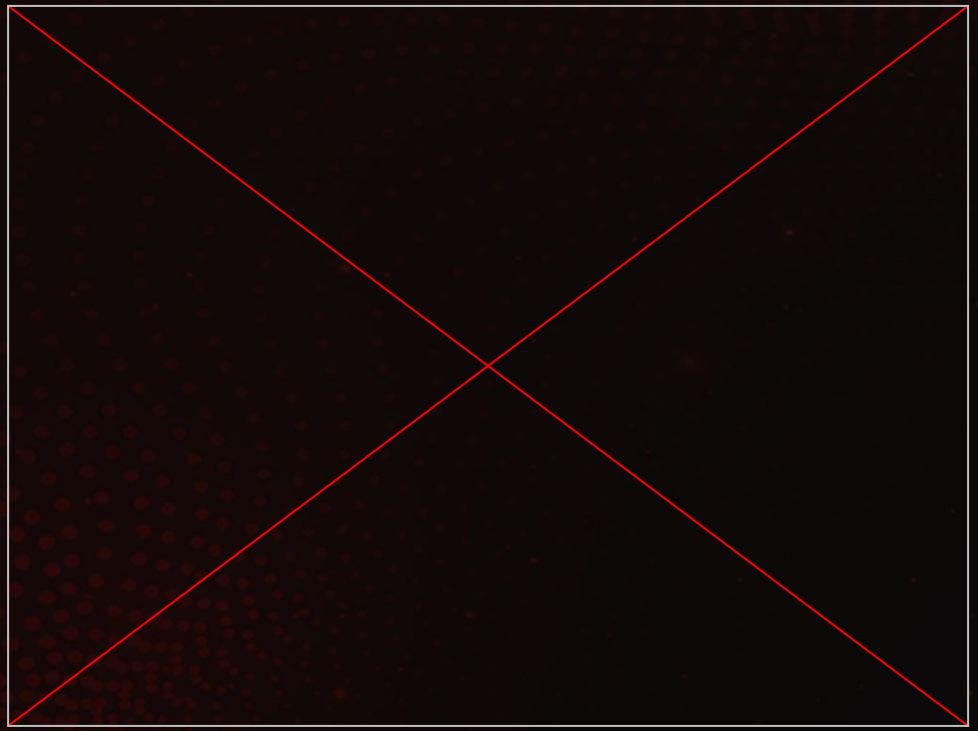
Motivation

- Focus on momentum-based gameplay
- Encourage skill mastery through practice
- Add risk/Reward decisions to movement

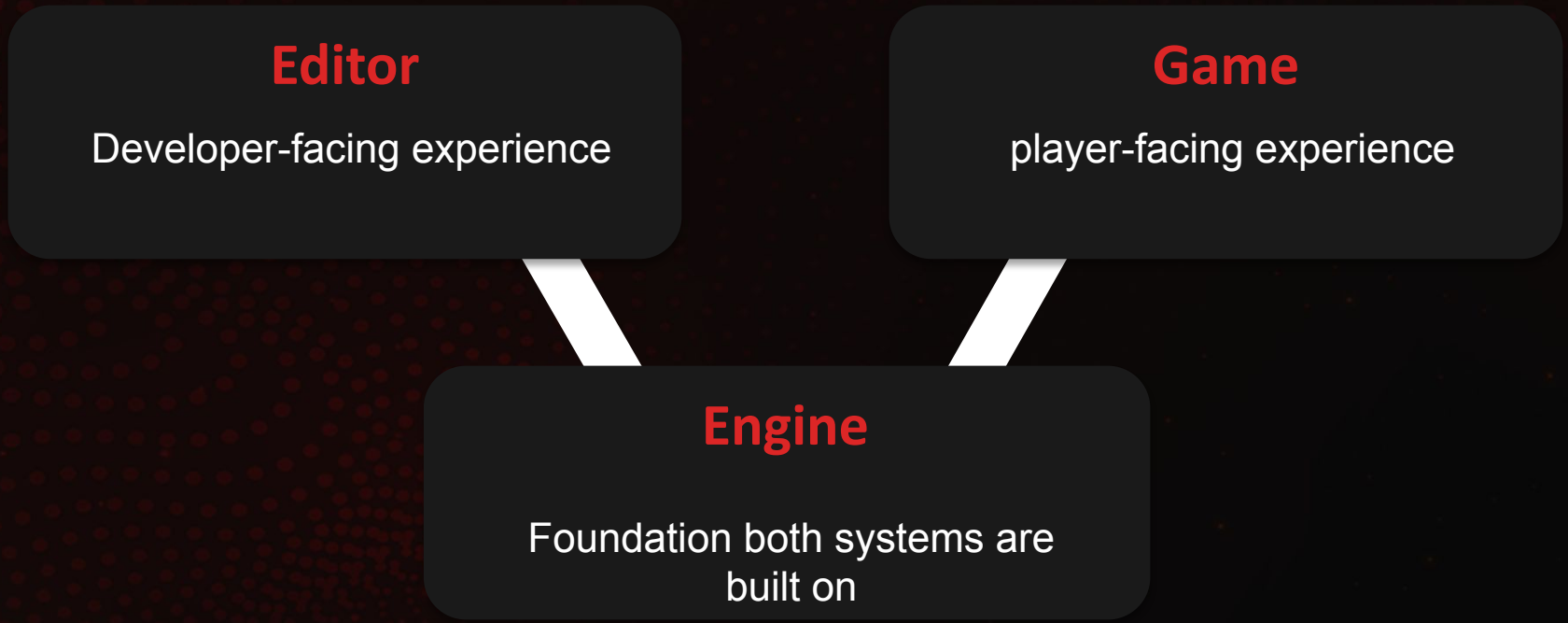
Design Decisions



GamePlay Visuals



Overview of systems



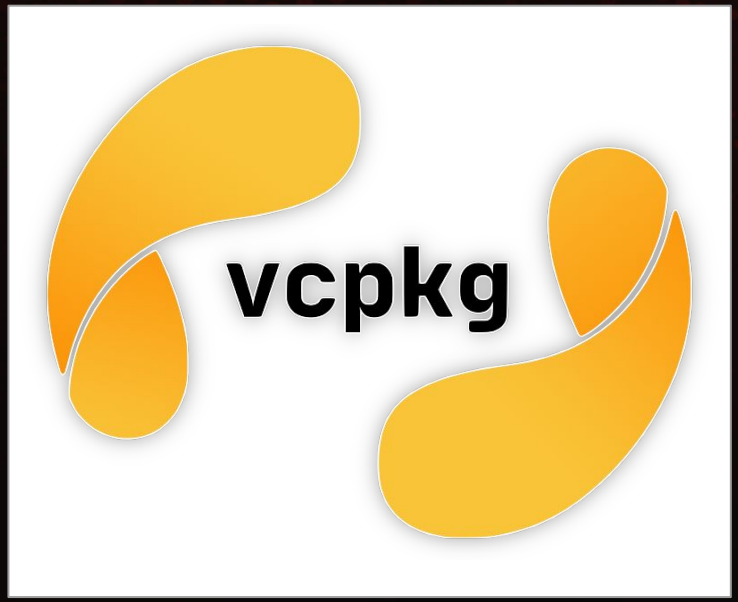
VCPKG

Dependencies

Easily incorporates depdicings during Development

Setup Vcpkg

Ensured that developers had VCPKG working on their machine



GameObjects & Components

Rendering System

- Previously all GameObjects had a render method
- Switched to using Render Components
- RenderComponent submits Render Command
- Render Command defines what to render
- Renderer Processes commands to draw objects

Build Types

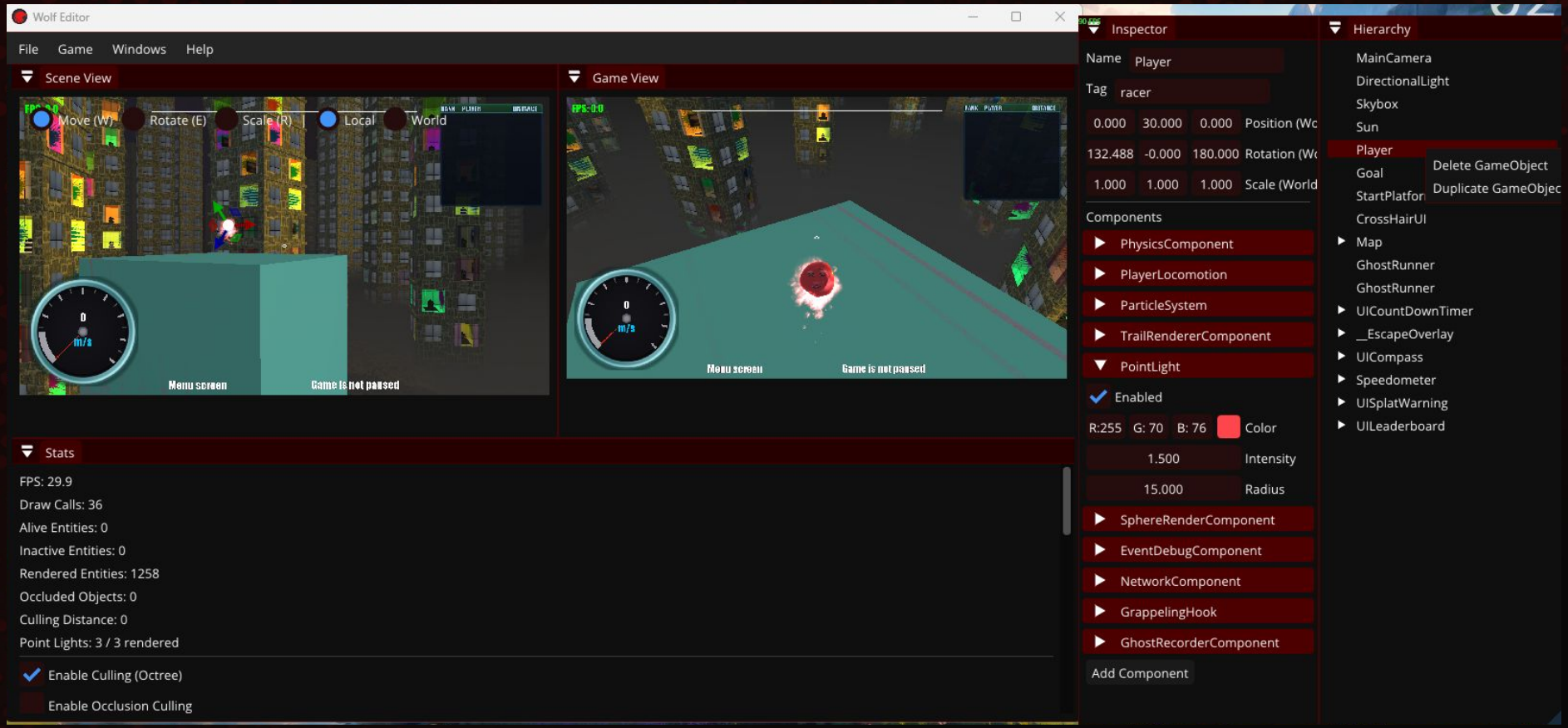
- Game Build and Editor Build
- Game Build handles core gameplay
  - Map loading and game loop
  - Optimized performance
- Editor Build used for development
  - Live editing of game data and scenes
  - Allows playing inside editor
  - Slight performance overhead due to debug systems

Editor

- Built Custom editor for faster development
- Live editing of gameObjects and component data
- Save/Load system for persistent changes
- Developer tools: Create, Delete, Duplicate (Objects)
- Undo/Redo Support

Picture Next Slide

Editor


Wolf Editor
File
Game
Windows
Help
Scene View
Move (W)
Rotate (E)
Scale (R)
Local
World
Menu screen
Game is not paused
Game View
FPS: 0.0
Menu screen
Game is not paused
Stats
FPS: 29.9
Draw Calls: 36
Alive Entities: 0
Inactive Entities: 0
Rendered Entities: 1258
Occluded Objects: 0
Culling Distance: 0
Point Lights: 3 / 3 rendered
Enable Culling (Octree)
Enable Occlusion Culling
Inspector
Name Player
Tag racer
0.000 30.000 0.000 Position (Wc
132.488 -0.000 180.000 Rotation (W
1.000 1.000 1.000 Scale (World
Components
PhysicsComponent
PlayerLocomotion
ParticleSystem
TrailRendererComponent
PointLight
Enabled
R:255 G: 70 B: 76 Color
1.500 Intensity
15.000 Radius
SphereRenderComponent
EventDebugComponent
NetworkComponent
GrappelingHook
GhostRecorderComponent
Add Component
Hierarchy
MainCamera
DirectionalLight
Skybox
Sun
Player
Goal
StartPlatfor
CrossHairUI
Map
GhostRunner
GhostRunner
UICountDownTimer
__EscapeOverlay
UICompass
Speedometer
UISplatWarning
UILeaderboard
Delete GameObject
Duplicate GameObjec

Engine Singleton

- Provides global access point to core systems
- Stores all major managers (Audio, Render, Scene, etc)
- Ensures a single shared engine instance
- Simplicis system access across the project
- Acts as central hub for engine architecture

Settings Manager

- Built settings manager for persistent player data
- Reads/Writes settings to external files
- Applies changes during runtime
- Supports save/load across sessions
- Example: Player customization (color, keybinds)

Written File

```
  invertMouseY: false
  sens: 0.481250018
gameplay:
  grappleAutoPull: false
audio:
  masterVolume: 0.0718749985
rendering:
  cullingDistance: 800
  showFPS: true
  resolutionScale: 1
graphics:
  screenShakeEnabled: true
  shadowQuality: Low
  mapPreset: Day
player:
  name: Player
  color:
    - 1
    - 0.341367841
    - 0.0114070177
```

Cameras

- Initial camera became too large and hard to manage
- Split into Base Camera Component for shared functionality
- Created specialized camera types for different modes
  - Editor Camera
  - Ghost Camera
  - Race Camera
- Each camera handles its own behavior and logic
- Improved modularity and maintainability

Game State Machine

- Scene Switching logic became increasingly complex and bloated
- Introduced GameStateMancine to manage game flow
- Uses Push/Pop stack-based state system
- Each state controls its own logic and behaviour
- Example States
  - Main Menu: Handles UI input and navigation
  - Race State: Manages gameplay flow and race completion
- Improved separation of game logic and scene management

MainMenuState

RaceState

SteamManager

- Central entry point to all steam features
- Stores access to steam-related managers
- Handles integration with Steam API
- Provides a single access point across the project

Steam Lobby Manager

- Handles creation and joining of steam Lobbies
- Uses Steam’s built in lobby system
- Manages player sessions and lobby data
- Supports the Networking system

Steam Leaderboard Manager

- Interfaces with steam leaderboard system
- Steam supports ~10,000 leaderboards
- Each map had its own leaderboard entry
- Separate debug and release leaderboards
- Automatically creates leaderboards for new maps

Dev Leaderboard

scene1.yaml_Dev Ascending - 0
Hide scores

Rank	Player	Score	Details
1	Brizly	14	-
2	falconcharge	14	-
3	benmartin2003	16	-

Release Leaderboard

scene1.yaml_2026 Ascending - 0
Hide scores

Rank	Player	Score	Details
1	falconcharge	11,037	-
2	AlienNug	23,986	-
3	Brizly	25,036	-
4	KludgyCartoon79	28,111	-
5	EvilSlayer	31,590	-
6	benmartin2003	32,454	-
7	colby3025	49,716	-

Steam Presence Manager

- Handles player status / Presence updates
- Syncs Steam friends list with current gameplay state
- Updates activity based on current game mode
  - Menu
  - Lobby
  - In Race
- Supports joining sessions even when the game is not running (Steam invite / join support)

Steam Upload Process

- Uses Steam’s built in upload system (steamworks)
- Requires developer login through steam tools
- Upload builds directly to Steam depots
- Upload Workflow
  - Built files locally
  - Upload using Steam publishing tool
  - Changes are version controlled (similar to git merge flow)
- Depots
  - Separate Build configuration per platform
  - Example: Windows and Mac builds contain different files
  - Ensures correct version is delivered for OS

Grappling Hook

- Replaced basic debug line grapple with an object
- Implemented as a segmented cylinder
- Segments allow for dynamic movement and flexibility
- Customer shader using sum of sines
- Creates a more responsive feel

Slowed video of Grappling hook

UI

- UIManager
  - Holds all UI objects
- UISlider
- UI Compass
- Fixed UIButton Accuracy
- UILabel

UI Label

UI Slider

UI Compass

Overview

World Generation

Terrain noise, biomes, erosion, streaming, LOD, building placement, curves, instancing

Player Systems

Entire Locomotion, dual grapple, Aim assist, coyote time, crosshair, speed boosts

Visual Effects

Volumetric fog, trail renderer, black hole shader, particle system, occlusion culling

Rendering Work

Fog integration, GPU instancing, Transparency, performance overhaul, 7 shader pairs written

Game Systems

Ghost replay, Cutscene manager, Tutorial, Settings, 3 gameplay gates

Engine Work

Frustum culling, Octree, Bounding volumes, Occlusion, Background loader, Debug renderer

Physics

Jolt config, heightfield terrain, collision layers, sphere colliders, constraints

Tools

3D debug renderer, bounding volume visualizer, stats window, Tracy profiling, editor fixes

Previous Year

Particle system, Text rendering, Culling system, Debug renderer.

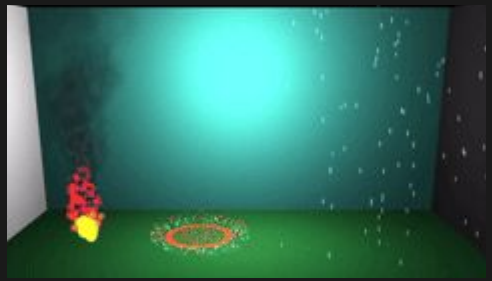
Full performance overhaul across the entire engine

Previous Work

Clean code from last year. Good practices. Plugged in day one.

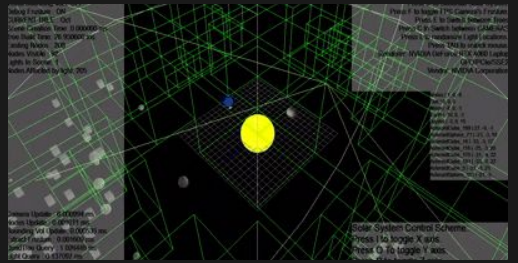
Particle System

5 emitter shapes,
Affector pipeline,
YAML serialization,
CPU depth sorting
for transparency



Culling System

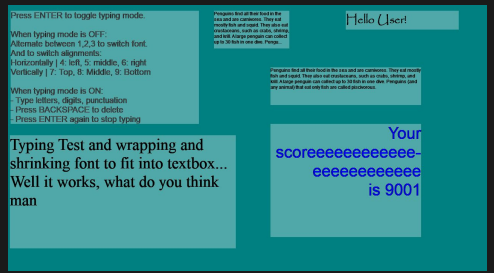
Frustum plane,
AABB tests,
Octree spatial
partitioning



Text Rendering

Font atlas,
text boxes,
screen space scaling.

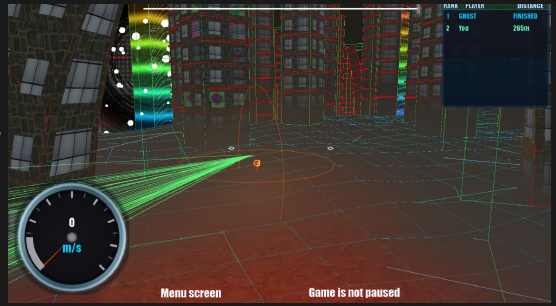
Fixed a window
resize bug later



Debug Renderer

Immediate shapes,
Named layers

Most important
debugging tool



Player Movement

PlayerLocomotion: the single biggest component in the entire project.

Camera

Third person Camera with speed-reactive Effects

Speed Boosts

Collision-triggered gates for momentum increase

Physics Layers

Separating players, buildings, and terrain

Dual Grappling Hooks

PlayerLocomotion: the single biggest component in the entire project.

Aim Assist

Raycast targeting with 40 samples per frame
Coyote Time: 0.3s forgiveness for near-misses

Constraints

Rope physics spring-dynamic constraints

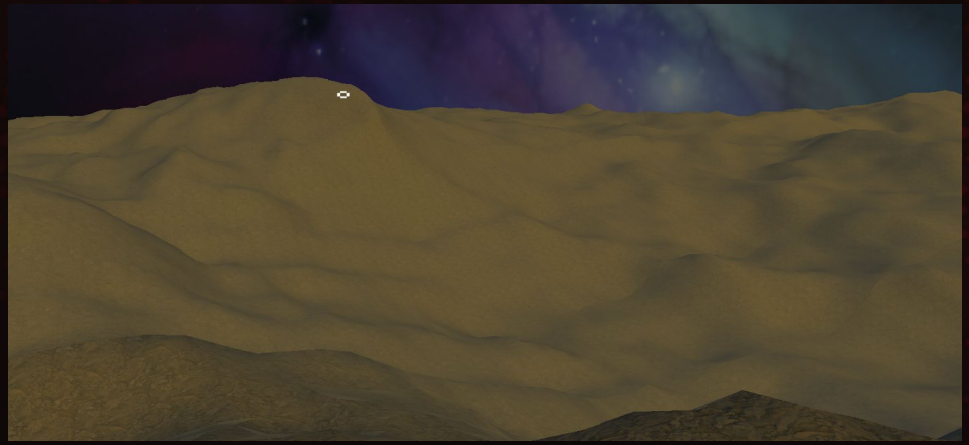
Stuck in Buildings

Raycast from camera -> avoid being inside buildings

A float typically uses 4 bytes.

Terrain System

Full pipeline: generation, streaming, rendering, physics. All procedural.

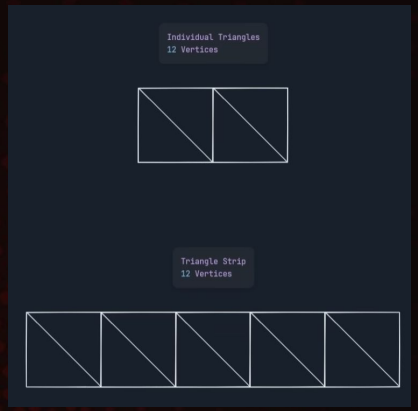


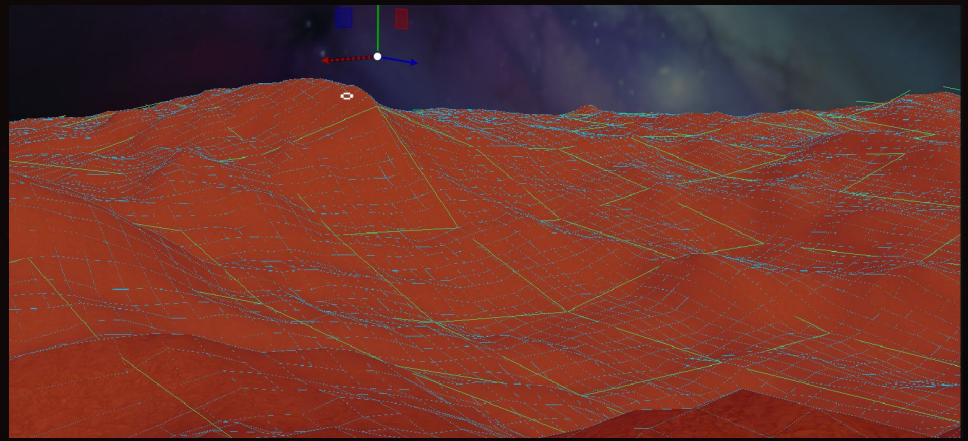
Noise

Layered Perlin noise with basic biome blending & thermal erosion

Streaming

Chunks stream on worker threads with LRU atlas cache





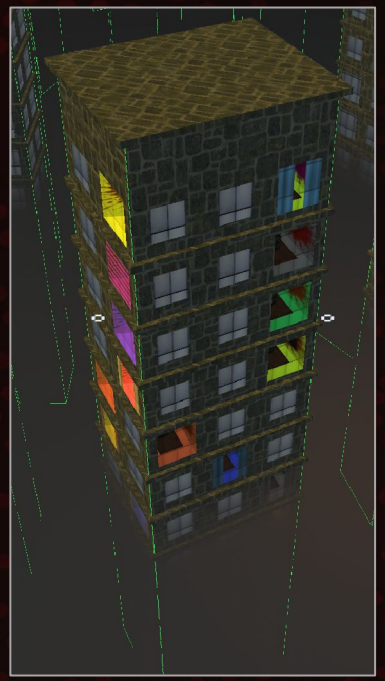
Challenge

Chunks unloading while still referenced caused crashes

Vercidium - Youtube

Buildings: How They Are Made

Flat Top



Two types

GPU-Instanced: hundreds in 1 call

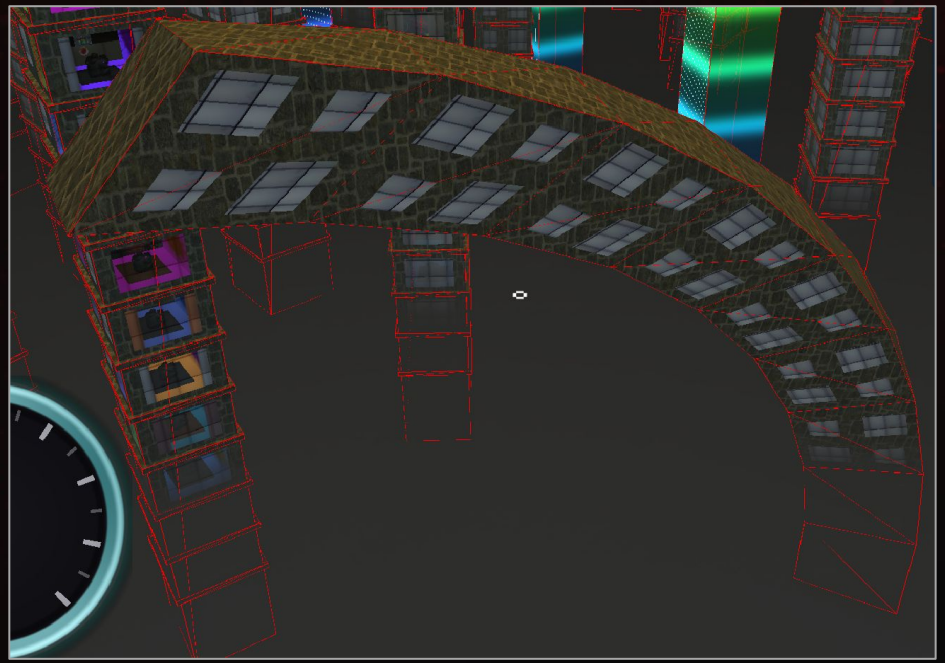
Curved Buildings

Visually bend on GPU
Physics collider bends on CPU

Matching CPU to GPU

Multiple rewrites

Curved / Arched



Buildings: How They Are Placed

Every building generated at runtime. Not one placed by hand.

Rule-Based Procedural City

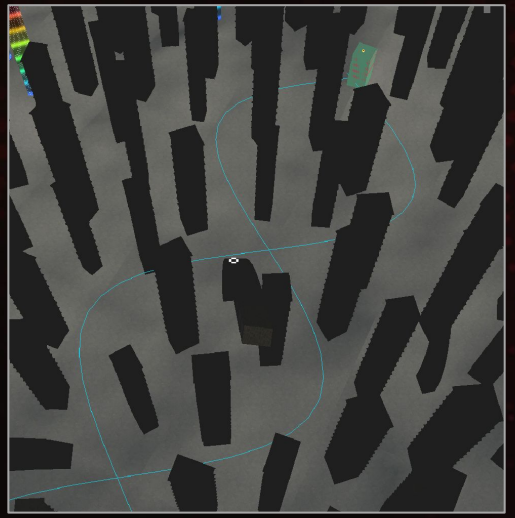
Noise masking and jitter for downtown-like density
Height scaling for visual progression

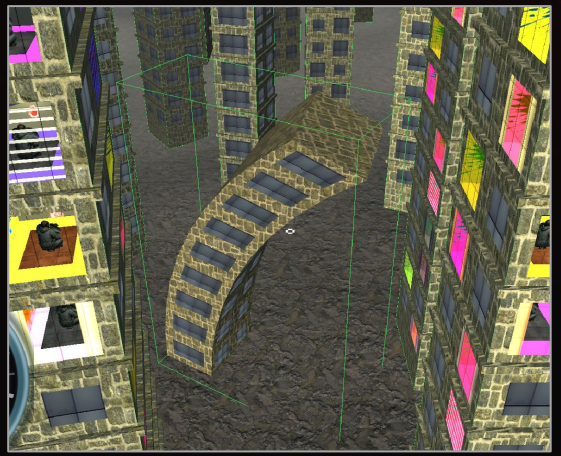
Route System

Ensuring doable map every generation, used for cutscene

Trial & Error

Every placement rule was added because of a bug





Volumetric Fog

Four major iterations.

Technique

Screen-space ray marching with scattering

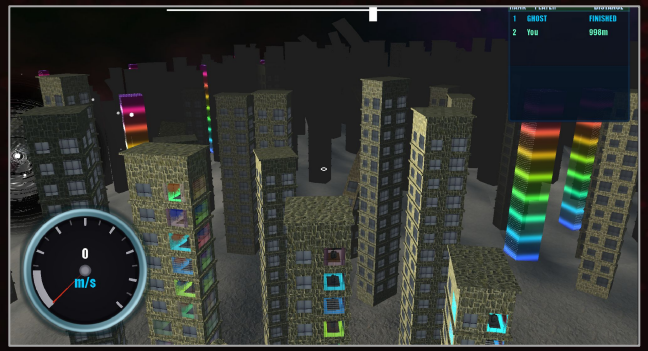
Half Resolution, Smoothing and height offset

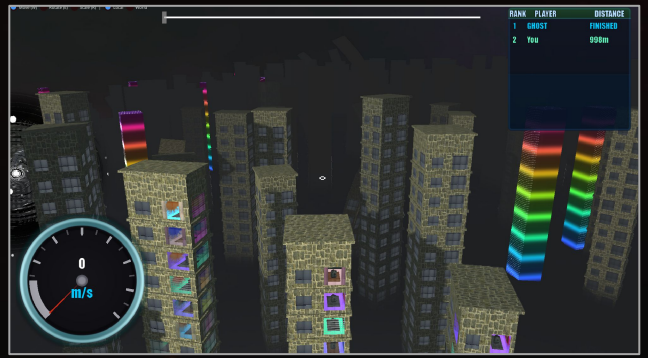
Renditions

Basic -> Layered -> Thin Film -> Ray Marched

Learnt

Half-res = good, Iterating on these renditions was necessary to reach the final result





Trail Renderer

Researched, built from scratch, optimized.

Technique

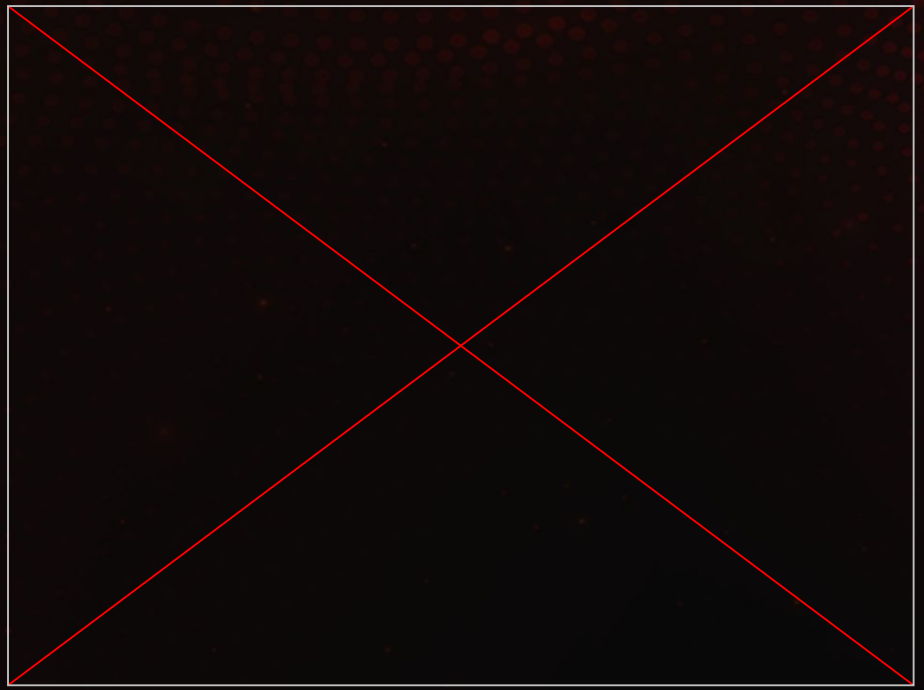
Ring Buffer with custom shader between player world-space points

Problem

Covered screen, felt flashy, Iterated

Game Changing

Added late, gave the game a fresh feel



Black Hole Shader

Replaced the basic goal with a fully procedural effect. No textures, all math.

Technique

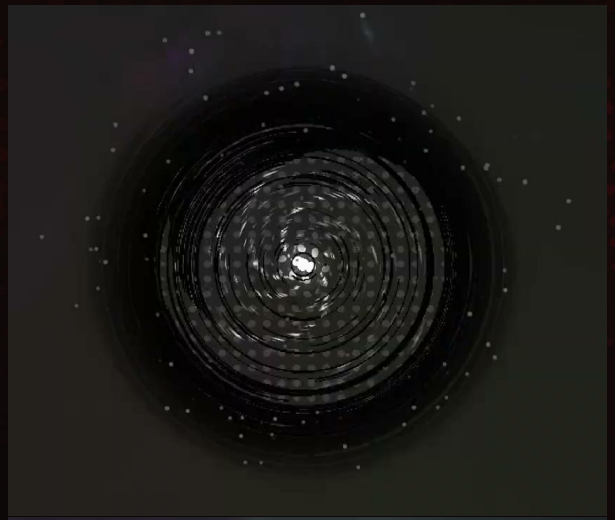
Spiral arms, Halftone, Distortion

Visual Gameplay Cue

Interacting with environment, Particles pulled in

Difficulty

Shader toy for hours



Ghost Replay

Race against yourself. Binary recording and playback.

How

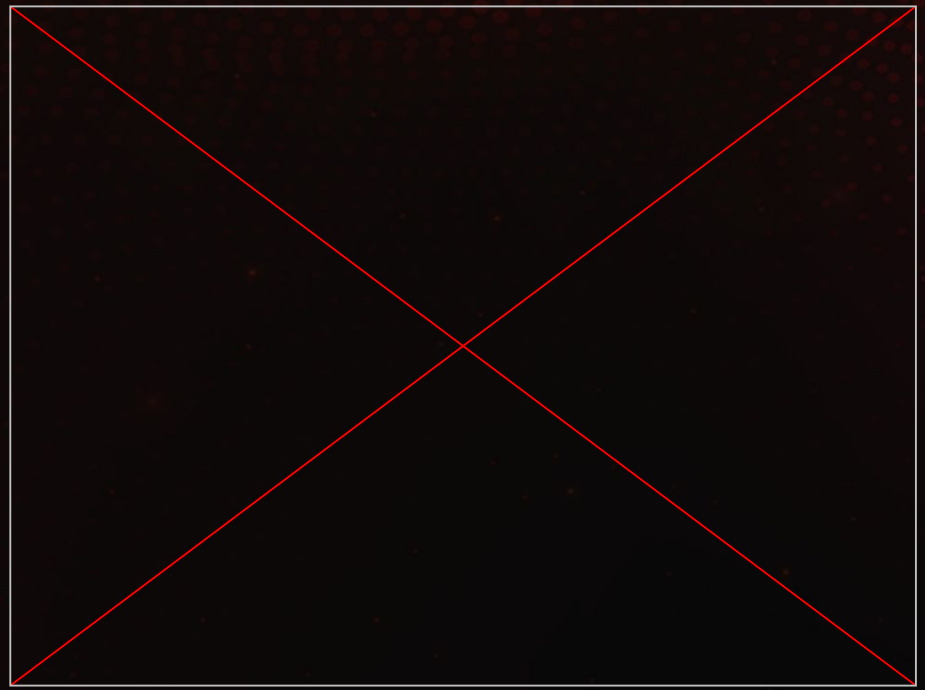
Binary Ghost recording at 30 fps | 56 bytes/frame

Issues

Ghost drifted out of sync; race clocks & cutscene addition

Performance

Binary is way faster than YAML or JSON, saved straight to disk



Cutscene System

For cinematic sequences.

Queue-Based Camera

7 command types for the camera to follow

Issue

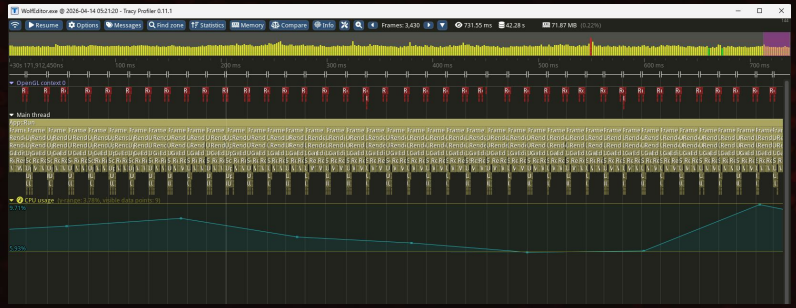
Camera was inconsistent following a path
Fixed using splines and curves

Arc-length parameterization

Constant speed regardless of how sharp the path curves

Optimization Flow

Profiled with Tracy.



Optimization

Fixed building additions not instancing

YAML save file sizes decreased

Refined culling logic for long distance maps

Chunked buildings and culled larger amounts (smaller = more CPU usage)

Profiling Pipeline

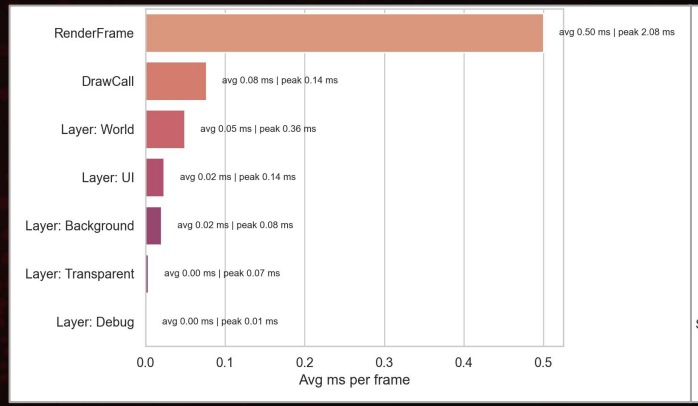
Saved tracy exported to CSV

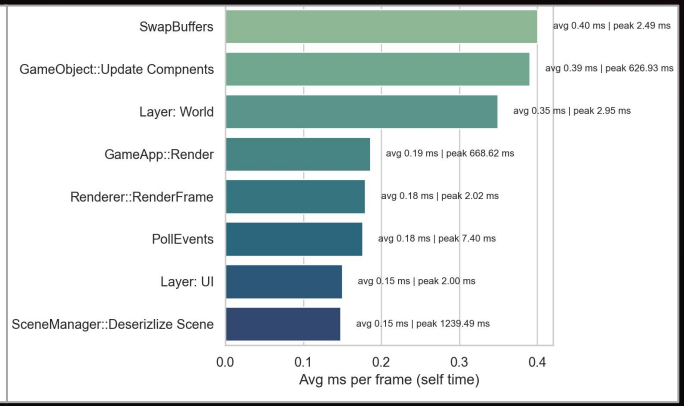
Categorizes every zone by subsystem (renderer, physics, etc)

Generates charts

Background Loader

Eliminated frame spikes but player spawns before other systems intialized, constant crashes - removed





Thanks

esgo,
epik