Geometry Slash

Technical Design Document

Table of Contents

Project Overview

Game Concept

The player is a shape surronded by other hostile shapes where they have to survive and defeat enemies with the borders closing in. The game finishes with a boss battle.

Technical Goals

A level

- Player Input (Character Movement & Skills)
- Simple Enemy AI
- EXP System
- Basic Skill System
- 2D Collision Detection

- Game UI
- Player, Enemy, Attack Animations
- Closing in Border

B level

- Adaptive Enemy AI
- Audio pitch randomized
- Procedually generated map obstacles / hazards
- Skill tree system
- Power-Ups

C level

- Shadows
- Advanced Particle System
- Camera Effects / Camera Shaking

System Requirements

OS: Windows XP

Minimum hardware:

- Processor: 1.5GHz or faster
- Ram: 2gb or higher
- Graphics: 256mb video memory
- DirectX: 9.0 or later
- Storage: 200mb available space

Recommended hardware:

- 2.0 GHz dual-core or faster
- Ram: 4gb or higher
- Graphics: 512mb video memory
- DirectX: 9.0 or later
- Storage: 200mb available space

Technical Risks

- **Platform Compatibility:** There may be unexpected compatibility issues or performance bottlenecks on older or unusual hardware, especially with graphics drivers or input devices.
- **Performance Optimization**: Although 2D games are lightweight, inefficient code or unoptimized asset handling could result in slowdowns on low-end systems, especially when many objects or effects are on screen.
- **AI/Gameplay Logic Flaws:** Simplistic or buggy AI could make the game too easy, too hard, or unpredictable, making player experience not enjoyable.

Third Party Tools

• C++ Standard Template Library (STL)
• Simple DirectMedia Layer (SDL2) including SDL2_Image and SDL2_TTF APIs
• The OpenGL Extension Wrangler Library (GLEW)
• Dear ImGui
• FMOD Core (Low-Level) API
• Box2D

Logic and Technical Algorithms

Core Game Loop Logic:

- Initialize game state (player, enemies, map)
- Process player input (movement, shooting, skills)
- Update positions and states of all shapes
- Handle collisions (player/enemy, projectile/enemy, etc.)
- Spawn new enemies/waves as needed
- Check for win/lose conditions (player death, boss defeated)
- Render all entities and UI
- Repeat until game ends
- Update score

Key Algorithms:

- Collision Detection

- Enemy Spawning
- Enemy Behaviour
- Input Handling

Debug Features

- FPS Counter
- Log Output
- Player God Mode