Seekers Test Plan for Milestone 2

CPSC 427 – Video Game Programming

# Team: Los Pollos Hermanos Gaming (LPHG)

Samuel Yiu (32494940)

Ahmed Al Busaidy (29200722)

Jakob Khalil (86176674)

Amir Goodarzvard Chegini (56851199)

Brendon Son (87271490)

## 1. Game AI and Decision Trees

### 1.1. Enemy State Behavior

* Verify enemies transition between states (idle, patrol, chase, attack)
* Test enemy detection radius and response to player
* Verify enemies coordinate in groups
* Test path: src/systems/AISystem.hpp

### 1.2. Combat Behavior

* Test enemy attack patterns and ranges
* Verify enemy weapon usage (melee vs ranged)
* Check enemy dodge and positioning logic
* Test victory dance animation on player death

## 2. 3D Animation System

### 2.1. Character Animations

* Test skeletal animation system
* Verify smooth transitions between animation states:
  + Idle
  + Walking/Running
  + Attack (standing and running)
  + Dodge roll
  + Death
  + Stagger
  + Victory dance
* Test path: src/renderer/AnimatedModel.hpp, src/renderer/Animation.hpp

### 2.2. Weapon Animations

* Verify weapon attachment to character models
* Test weapon switching animations
* Check projectile animations
* Test path: src/renderer/Joint.hpp

## 3. Asset Integration

### 3.1. 3D Models

* Verify loading of character models:
  + Hero character
  + Warrior enemies
  + Archer enemies
  + Zombie enemies
* Check environmental models:
  + Trees
  + Walls
  + Terrain
* Test path: src/models/

### 3.2. Audio Assets

* Test combat sound effects:
  + Sword slashes
  + Bow shots
  + Footsteps
  + Dodge sounds
* Verify background music
* Test path: src/audio/

## 4. Mesh-based Collision Detection

### 4.1. Entity Collisions

* Test player-enemy collisions
* Verify projectile-entity collisions
* Check weapon hit detection
* Test path: src/systems/CollisionSystem.hpp

### 4.2. Environmental Collisions

* Test wall collisions
* Verify tree collision boundaries
* Check collision with map boundaries
* Test dodge roll collision handling

## 5. Tutorial System

### 5.1. Control Instructions

* Verify WASD movement tutorial
* Test mouse aim tutorial
* Check attack instruction display
* Verify dodge roll tutorial
* Test path: src/systems/TutorialSystem.hpp

### 5.2. Tutorial Flow

* Test tutorial trigger conditions
* Verify tutorial progression
* Check tutorial UI visibility
* Test tutorial completion

## 6. Performance Features

### 6.1. FPS Display

* Verify FPS counter visibility
* Check FPS counter accuracy
* Test FPS stability under load
* Test path: src/app/Application.hpp

## 7. Procedural Generation

### 7.1. Map Generation

* Test dungeon room generation
* Verify corridor connections
* Check enemy placement
* Test object distribution
* Test path: src/systems/ProceduralGenerationSystem.hpp

### 7.2. Environment Variety

* Verify room size variation
* Test different layout configurations
* Check decoration placement
* Verify spawn point placement

## 8. Graphics and Rendering

### 8.1. 3D Camera System

* Test camera movement and rotation
* Verify perspective projection
* Check camera collision handling
* Test path: src/renderer/Camera.hpp

### 8.2. Lighting System

* Test Blinn-Phong shading
* Verify normal mapping
* Check dynamic shadows
* Test path: src/shaders/AnimatedBlinnPhong.fs.glsl

## 9. Stability Testing

### 9.1. Performance

* Test frame rate stability during:
  + Heavy combat
  + Multiple enemies
  + Large rooms
  + Particle effects

### 9.2. Memory Management

* Monitor memory usage during extended play
* Test for memory leaks
* Verify resource cleanup

## Test Execution Notes:

### 1. Environment Setup

* Clean build directory
* Fresh game instance
* Consistent hardware environment

### 2. Test Documentation

* Record all test results
* Document any bugs found
* Note performance metrics
* Track crash reports

### 3. Success Criteria

* Stable 60 FPS gameplay
* No critical bugs
* Smooth animation transitions
* Responsive controls
* Consistent collision detection