Light Game

Work in Progress Name

Game Design Document

Revision 1.1

Project Description

The Elevator Pitch
Imagine yourself alone, in an unknown, dark place surrounded by unseen horrors with nothing but a flashlight. You have to escape, find light, but there are obstacles you will face and hopefully you're able to overcome these before you are killed by the creatures that surround you.

Game Engine
- Unity
- C#

Theme / Setting / Genre
- Dark, horror
- Puzzle platforming

Core Mechanics
- Flashlight, limited view
- Unlit puzzle solving
- Light-sensitive monsters

Target Platforms
- PC
- Mobile (if we can, because that's super cool)

Controls
- *PC*
  A and D to move left and right. Space to jump. Mouse to move flashlight.

- *Mobile*
  Press and hold left half of screen to move left, press and hold right half of screen to move right. To jump, move in one direction and tap other with other finger.

Project scope
Time scale: 2-3 months

Team
- Ramon Meza (ramon_meza.t22)
- Matt Metzger (mrace724)
- Dalton Curtin (dalton.curtin)
- Vi (huhwhathappened)
- Robin Bhattacharya (robin_bhattacharya)
- Joel Cruz (santana_andahalf)

Influences

- *Limbo (Game)*
  This dark platformer captures a creepy atmosphere that I believe would work perfect in our game. The overall environment art style is something we should seek to replicate, obviously with our own artistic twist.

- *Knock Knock (Game)*
  The perfect example of a 2D horror game. The art style is almost exactly what I envision. There's a particular scene in which you are traveling through a forest. This section of the game has great environment art. There are also hidden pictures in the environment, such as trees that look like hands grabbing the playing, or evil faces on the trees. This is great inspiration for our game.

- *SCP: Containment Breach (Game)*
  This game captures creatures and enemy AI perfectly. Each creature has it own unique features and back story that make them scary, if not visually then most definitely psychologically. SCP is a 3D maze horror game that scares the shit out of anyone who plays it and I think looking at it for inspiration for our creatures and monsters is smart.

- *Don't Starve (Game)*
  This game has exactly the style of art I want for our characters and creatures, if possible. Something concept looking, messy looking, but still beautiful.

Story and Gameplay

Story

Gameplay

Players will awaken in a dark house/warehouse, with the sound of monsters to the left. This will indicate to the player that they need to run away to the right. Here we have a simple puzzle to introduce the player to our collision detection/fluid animation system. We can have the player push open a door that is stuck, and thus will take a second before falling to the ground, allowing the player to run into a forest. Above the door on the outside is a lamp, which before off screen we will show the monster running through and dying. This will hopefully show our player that monsters are light-sensitive. Parallax scrolling will be a big environmental effect we will use for the forest as the player runs through. We can introduce the flashlight here, on the corpse of a dead unknown character, flickering. We will have the character automatically pick it up. As the player progresses they will push another obstacle, and thus putting their flashlight on their side.
The objective is to avoid using text tutorials, or any tutorial for that matter. There won't be any GUI if we design this correctly. I would love for the experience to be 100% fluid and natural. Mario never had text telling you what keys to press.

Core Mechanics Detailed

- *Flashlight, limited view*

  You will carry a flashlight throughout the game. Your flashlight is your key to survival and that needs to be emphasized at the beginning of the player's experience. We can achieve this sense of dependency by starting the player without it, surrounded by monsters.

  The flashlight will be controlled by the mouse on PC, or a simplified version for mobile will only be rotated as the player changes direction. To solve puzzles, the player will have to hang the flashlight from his/her belt, leaving it dangling from their side. The light beam will be made using 2DVLS for now.

  The light may be battery powered, losing energy overtime, causing it to flicker and eventually go out. At that time the player's vision will darker, leaving them with less of the world to see.

- *Unlit puzzle solving*

  The environment will affect what the puzzles are. For example, if a player is in a forest level, they may need to push a log to bridge a gap, or if they are in an abandoned warehouse, they may need to push a box to help them climb up into a vent.

One idea that may be cool to implement is having complex puzzles that may require the player to look around and backtrack before understanding how to solve it.

- *Light-sensitive monsters*

  Monsters will react to light, all light. If a player flashes his/her light at the monster, the monster will be stunned and burst into flames/smoke after an amount of time. Monster design will also reflect the idea that they always in the dark, lacking features like eyes or having more distinct features like ears. Monsters will travel faster if the player does not have their light on, as well.

Assets

Sprites

- Player (either spritesheet [harder] or limbs [easier])
- Flashlight
- Lanterns
- Streetlights
- Enemies (either spritesheet [harder] or limbs [easier])
- Corpses
- Door
- Wood panels
- Concrete walls
- Cracks
- Grass
- Grass blades
- Dirt
- Rocks
- Cement
- Trees (both dead and alive)
- Tree stumps
- Forest backdrop
- Flowers
- Batteries
- Fences (wooden, metal, electric, etc.)
- Mountains
- Particles (fire, smoke, fog, rain, snow)

Characters List

- Protagonist (male or female TBD)
- Enemies (will add later)

Levels

- Spooky Woods
- Abandoned Warehouse
- Creaking Wooden Houses

Sound

Ambient

- Rumbling sound
- Wind blowing
- Whispers
- Trees brushing together
- Crickets

Player Sounds

- Footsteps for various surfaces (wood, concrete, dirt, twigs, etc.)
- Light Breathing
- Heavy Breathing (after running or when monster gets to close)
- Heartbeat (when monster get's to close or when light goes out)

Monster/ Creature Sounds

- Footsteps for various surfaces (if time permits, otherwise one is fine)
- Various growls
- Various ad-libs
- Reverse talking
- Whispers
- “Wet” sounds (drooling if monster requires it, or just to make moving sound less human and scary)

Code

We will be using C# as our main programming language.

Character Scripts

- *PlayerMovement.cs*
  Left, right, and jumping.

- *FlashlightMovement.cs*
  Control the flashlight's rotation with the mouse.

- *FlashlightController.cs*
  Anything related specifically to the flashlight goes here. When the player puts away his flashlight, when the player is a certain distance from the light, what happens when the batteries run out.

- *PlayerCollisionDetection.cs*
  Detects when the player is close to an object and returns which object they are close to, preferably through raycasting, though triggers are easier. Performance is an issue, we will talk about this.

- *PlayerObjectInteraction.cs*
  Uses information from PlayerCollisionDetection.cs to figure out what needs to be done on a certain object.

- *PlayerAnimations.cs*
  Using information from the other scripts, this will communicate with the Animator component to provide as smooth as possible animations.

- *PlayerSounds.cs*
  This will be be the controller for all sounds that come from the player.

Enemy Scripts

- *EnemyMovement.cs*
  Left, right, and jumping.

- *EnemyCollisionDetection.cs*
  Detects when the enemy is close to an object and returns which object they are close to, preferably through raycasting, though triggers are easier. Performance is an issue.

- *EnemyObjectInteraction.cs*
  Uses information from EnemyCollisionDetection.cs to figure out what needs to be done on a certain object.

- *EnemyAnimations.cs*
  Using information from the other scripts, this will communicate with the Animator component to provide as smooth as possible animations.

- *EnemySounds.cs*
  This will be be the controller for all sounds that come from the enemy.

Ambient Scripts

- *CameraFollow.cs*
  This will specify an object the camera has to follow. This should be advanced enough to avoid clipping through the environment.

- *Parallax.cs*
  This will contain an array of layers that will parallax as the player moves.

- *EnvironmentSounds.cs*
  This will be be the controller for all sounds that come from the environment.

- *BackgroundMusic.cs*
  This controlls all background music. This will be hard, as it can be influenced by triggers and events.

Roles

- Ramon Meza
  FlashlightMovement.cs, FlashlightController.cs, PlayerSounds.cs

- Matt Metzger
  Github, CollisionDetection.cs, EnemyCollisionDetection.cs

- Dalton Curtin
  PlayerMovement.cs, PlayerObjectInteraction.cs

- Vi
  Setting up Animator, PlayerAnimationController.cs, EnemyAnimationController.cs

- Robin Bhattacharya
  Parallax.cs, CameraFollow.cs

- Joel Cruz
  EnvironmentSounds.cs, EnemyObjectInteraction.cs

Project Directory Structure

- LightGame
  - Assets
    - 2DVLS
    - Animations
      - Player
      - Enemy
        - Enemy1
        - Enemy2
        - etc.
      - Environment
    - Materials
      - Characters
      - Environment
    - Prefabs
    - Scenes
    - Scripts
      - Player
      - Enemy
      - Environment
    - Sprites
      - Characters
      - Level
      - Misc