Lights Out

By Psyactive Studios

"What is seen through the eyes of an adult, is easily misunderstood in the mind of a child."

Technical Design Document

Devlyn : Lead Programmer : Coordination, Overseeing, Core Development
Corey : Semi-Lead Programmer : Maintenance, Refining
Devran : Programmer : Organizing, Drafting

Table of Contents

Game Concept

Lights Out is a horror based game that utilizes a player controller to navigate rooms, solve puzzles using an interactable system and reach the end goal all the while keeping their stats in check. The game is singleplayer based and it gives the player itself the abilities to walk, interact and toggle their light as they take control of their environment to best suit the end-goal.

Technical Goals

The technical aspects of ***Lights Out*** will be in line with how the predetermined designed levels will interact with both the player and the lighting systems being used in tandem with one-another. The player will need to be able to interact with the environment set out for them, as well as solve puzzles, shake their portable light object and open / close (toggle) doors.

The supposed “Monsters in the dark” aren’t being programmed with complex AI terminology, rather it’ll use simple AI programming to function properly.

The lighting system being utilized for the game will need to be a relatively close to “real-time” system that can be used to difference between light and darkness to determine if the player is in one or the other, this will likely be achieved through detection layers that can determine if the player is actually in the dark or not without needing complex mathematical equations to determine light levels.

Technical Goal 1 - Tilemap Generation

Responsibility: Devlyn
Tech Goal: To achieve a fully functioning Tilemap Generator system that allows for easy creation of levels / maps for the project. This allows the designers to draw the maps rather than individually placing each tile on the map.

Technical Goal 2 - User Interactivity

Responsibility: Corey
Tech Goal: To implement and unify all User Interfaces from a programming and technical stand-point, as well as implementing and refining core functionality such as the protagonist’s “Glow Toy” (Portable Light Source) and player controls.

Technical Goal 3 - Non-Gameplay Essentials

Responsibility: Devran
Tech Goal: To develop all systems based around the core elements that will tie the game’s loose ends in development together, such as implementing the Fireplace time-flow, Audio / Sound Queues and any other non-gameplay essentials that pop-up during development.

Technical Risks

The technical risks following this particular project are not directly crucial to the project's success, as these risks are minor in terms of effectiveness. However it is still worth listing them in case they do happen in the future.

Technical Risk 1 - Broken Scripts

A major concern with the technicalities behind the project would be how scripted are handled, if any of the more crucial scripts (such as GameManager.cs) were to fail, there'd be a lot of scripts that rely on those systems that would also fail, causing a domino chain of broken scripting and making it quite clear how broken the game is.

To account for this; Put more focus into ironing out these systems as much as possible while reducing the margin for error, mostly by trimming out unused or unnecessary definitions, functions / methods and so on. That way when a script reads a Manager script there will be little to no margin for error that causes the script to break due to the Manager rather than the scripting within the object itself.

Features / Mechanics / Tasks

Feature / Mechanic	Who's Responsible	Scheduled Date
~~Clock and~~ Fire	Devran	Beta - 11/11/2023
Fear Mechanic	Devlyn	Alpha - 8/11/2023

Player Control	Devlyn	Alpha - 2/11/2023
Settings	Corey, Devlyn	Alpha - 1/11/2023
Main Menu UI	Corey	Alpha - 10/11/2023
Interaction System	Devlyn	Beta - 15/11/2023
Scene Manager / Game Manager	Corey, Devlyn	Beta - 16/11/2023
Interactables	Devran	Beta - 17/11/2023
Fail State System	Corey	Beta - 22/11/2023
Spawn Generation	Devlyn	Beta - 23/11/2023
Glow Toy	Corey	Alpha - 10/11/2023
~~Chase Scene~~	~~Devlyn~~	~~Beta - 24/11/2023~~

Deliverables

Deliverable	Who's Responsible	Who's the Owner
~~Itch.io Release Installer~~	~~Corey~~	~~Producer~~
Itch.io Release .ZIP	Devran	Producer

System Requirements

The game utilizes real-time lighting systems within a 3D space and relies heavily on shadow and core lighting, perhaps a heavier system requirement spec-sheet is required.

Minimum Requirements

Requires a 64-bit processor and operating system.

- ***Operating System:*** Windows 7, Windows 8, Windows 10, Windows 11
- ***Processor (CPU):*** Intel Core i3-4340 / AMD FX-6300
- ***Memory:*** 4GB Memory
- ***Graphics:*** NVIDIA GTX 660 2GB / AMD Radeon HD 7850 2GB
- ***DirectX:*** DirectX 10, DirectX 11
- ***Storage:*** 5GB Available Space

Recommended Requirements

Requires a 64-bit processor and operating system.

- ***Operating System:*** Windows 10, Windows 11
- ***Processor (CPU):*** Intel Core i5-12400F / AMD Ryzen 3 2200G
- ***Memory:*** 6GB Memory
- ***Graphics:*** NVIDIA GTX 1050 TI 4GB / AMD RX 580
- ***DirectX:*** DirectX 10, DirectX 11
- ***Storage:*** 5GB Available Space

Target Device - Personal Computer (PC)

Refer to ***System Requirements*** for a full list of minimum and recommended requirements.

Third Party Tools

The following third party tools will be utilized to both program and develop the game itself, including the game's framework, objects, etc.

Programs	Use-Case
Unity Engine (Version: 2021.3.13f1)	Used to develop the game in its entirety so as to make it easier to both collaboratively work with team members on the project with all skill-sets whilst also enabling easier porting of game systems.
Visual Studio 2022	Programming IDE for the game's C# scripts for utilization within the Unity Engine itself.
~~Fl Studio~~	~~Audio producing tool to create music and sound effects for the game.~~
~~Inno Setup~~	~~Installer to create an easier method for setting up and running the project.~~
Autodesk Maya	3D Program to create the game's models and animations to be then imported directly into the Unity Engine.
Adobe Photoshop	Art / Photo Editing software used to produce the game's 2D art elements such as UI and Character Art.

File Formats

File formats used for models, textures, sounds and any other assets.

Type	Accepted Formats
Models	.fbx
Image	.png, .svg
Scripts	.cs, .json, .ini

Coding Conventions

Programming Language: *C#* + *Unity Engine APIs*

```
using System;

namespace Unity {

    /// <summary>
    /// Basic Object
    /// </summary>
     public class Object : MonoBehaviour
     {
        #region Properties

        // Public variables and properties must start with an uppercase

        public int Getter { get => getter; }
        public int Setter { set { setter = value; } }
        public int Property { get { return property; } set { property = value; } }

        #endregion

        #region Variables

        // Variables must start with a lowercase

        private int getter;
        private int setter;
        private int property;

        #endregion

        #region Inspector

        // SerializeField must be inline with the variable it's serializing

        [SerializeField] private int serializedField;

        #endregion

        #region Internal

        private void Update()
        {

        }

        /// <summary>
        /// This is a private internal method
        /// </summary>
        /// <param name="foo"></param>
        /// <param name="bar"></param>
```

```
        private void MyPrivateFunc(int foo, string bar)
        {
            return;
        }

        #endregion

        #region Methods

        /// <summary>
        /// This is a public method
        /// </summary>
        /// <param name="baz"></param>
        /// <param name="qux"></param>
        public void MyPublicFunc(int baz, string qux)
        {
            return;
        }

        /// <summary>
        /// A basic method
        /// </summary>
        public void BasicFunction()
        {

            int _scopedVariable = 0; // Variables defined in function must be prefixed with a underscore and start at a lowercase

            if (_scopedVariable != 0) // Use guard clauses to eliminate if statement nesting
                return;

            //
            // - If indentation from functions initial indentation is above 3,
            //   most likely it is a key indicator that the code you are writing needs to be
            //   extracted into a separate function.
            // - Try to avoid while true loops as much as possible.
            // - Use a Foreach loop if accessing data in an array frequently and not modifying
            //   the array itself.
            //

            return;
        }
        #endregion
    }
}
```

Source Control

Github -> used to store the project

Github Desktop / Sourcetree -> used to commit/clone/push/merge to Github (can use MSVC)

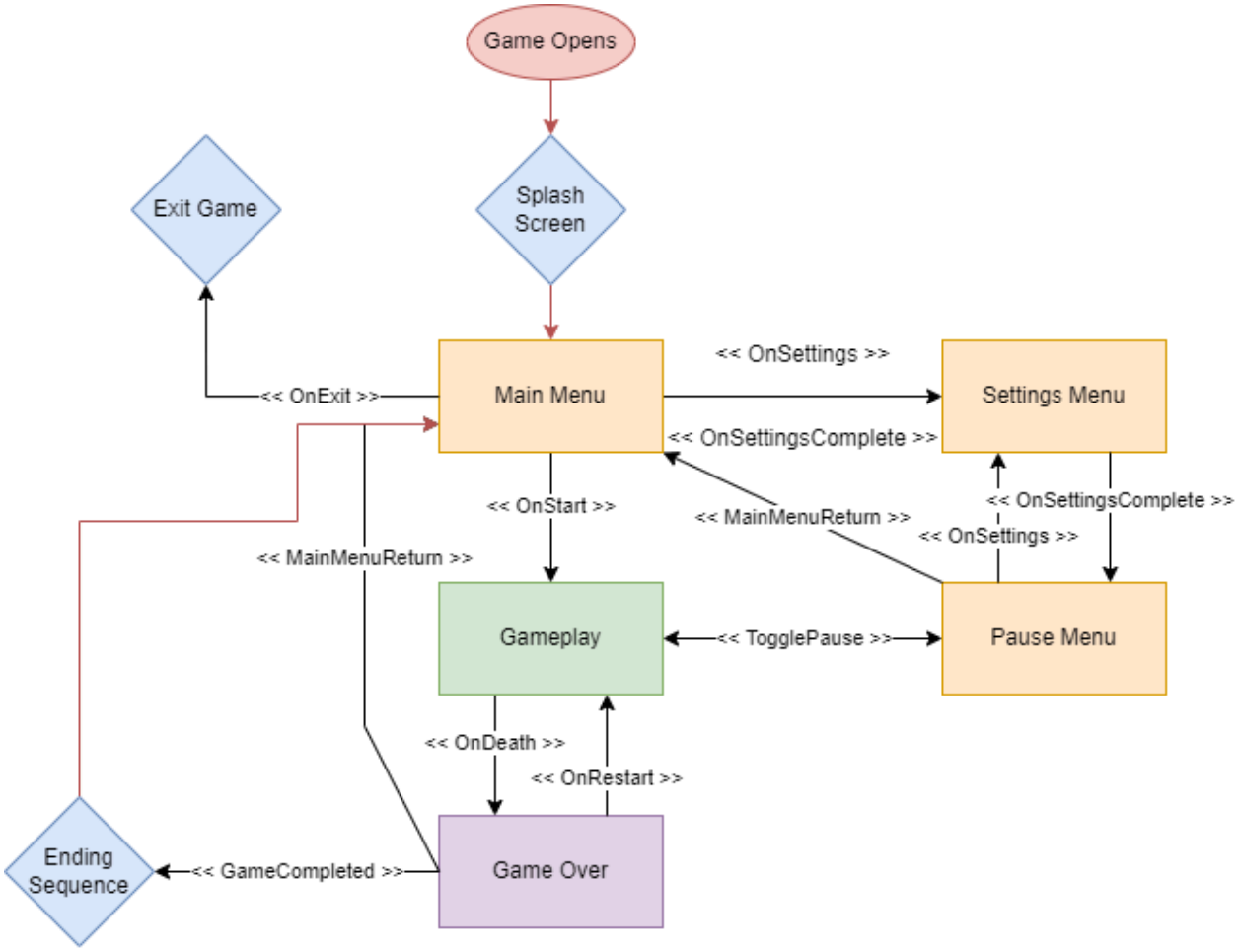
Game Flow

Scenes	Who's Responsible	What is does
Menus - Logo Splash Screen	Corey	Introduces the team who developed the game before transitioning into

		the Title Screen, lasts around 3 - 5 seconds.
Menus - Main Menu	Devran	Enables the player to navigate between menus, change settings, view credits, exit the game or start the game all in one scene.
Game - Introduction	Corey	Debriefs the game's narrative and provides context as to why we follow the protagonist through the game.
Game - Start	Corey	With your protagonist spawned within the game world, the game will start counting time and progressing forward.
Game - Puzzles	Devlyn	To get to the end-game, the player will need to complete a randomly generated

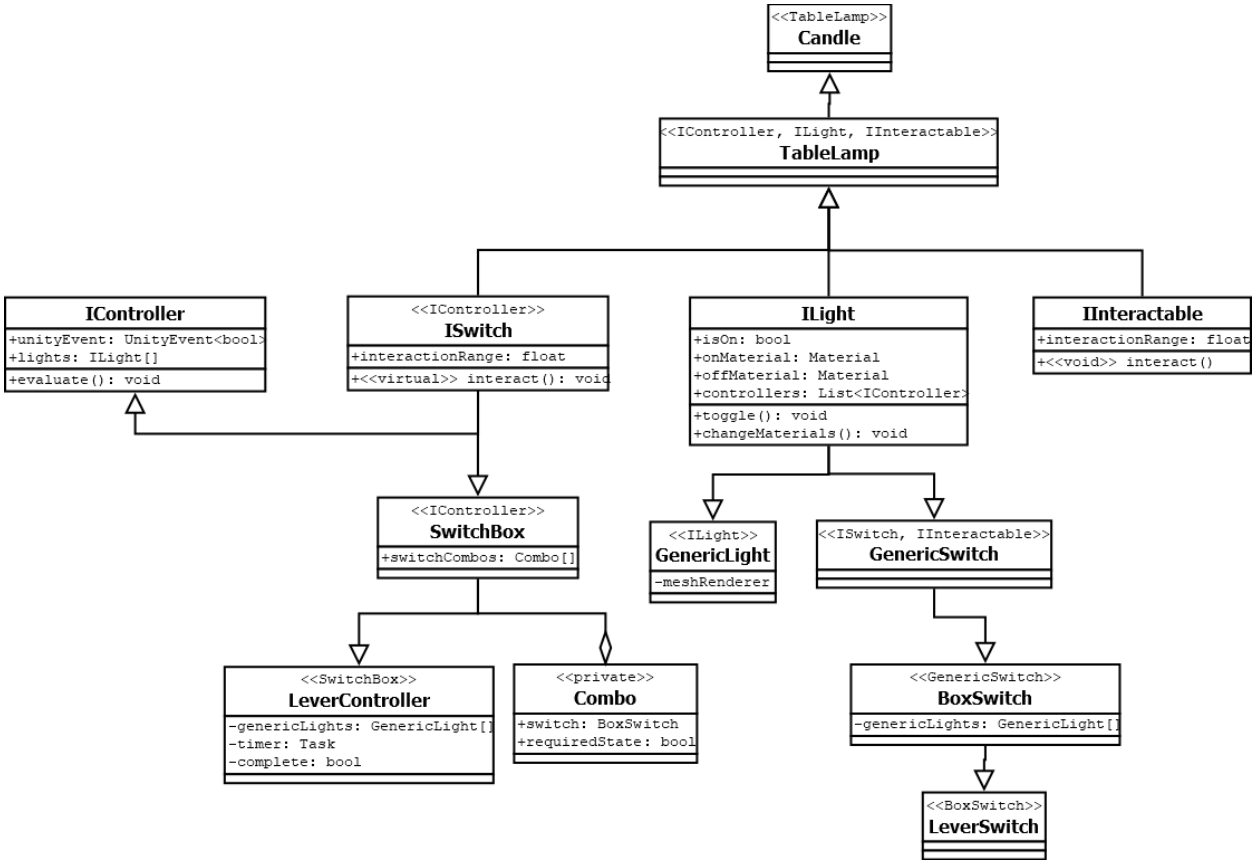
		pre-set of puzzles throughout the house.
Game - Game Over State	Corey / Devlyn	Upon your fear meter reaching its limit, everything around the protagonist becomes pitch black, the protagonist stares at the camera as the shadow hands grasp them firmly, shaken with the darker ambience getting louder until everything cuts off to black, a menu is then shown with you being tasked to either restart or quit.
~~Game - End-Game Climax~~	~~Devlyn~~	~~After completing the base game's puzzles and doing everything else between, the end-game climax will start, the chase scene where the protagonist has to~~

		~~b-line it for their room through an extended and collapsing hallway, reminiscent of a dream or nightmare. This will lock all states out and force the player to move forward or else a game-over state is initiated.~~
Game - Ending	Corey / Devlyn	Contains the player's overall game-stats and rank for completion.



Game Objects and Scripts

To develop the game's objects and interactivity, we've decided it'd be best to have base versions of objects that are inherited by more specific variants of objects in an inheritance chain. That way we can save needing to re-define variables and have all object logic be synced across with each-other, the class diagram below shows how inheritance controllers are handled with this programming mindset. Lights and Switches being the most common form of objects within this project, both utilize this method of inheritance to create compatibility with multiple types of objects that perform similar actions.



Gameplay Systems

Gameplay System 1 - Movement

Who's Responsible: Devlyn

Description: The player's velocity is instantly set to a direction and speed. Direction depends on the direction of the input. Based on the input it gives motion and direction to the protagonist.

Scripts: PlayerController.cs, InputManager.cs

Gameplay System 2 - Interaction

Who's Responsible: Devlyn

Description: To indulge and make an impact on the world around you through interaction. Mostly to change the state of the light, being turned on and turned off.
Scripts: GenericSwitch.cs, GameManager.cs, InputManager.cs

Gameplay System 3 - Glow Toy

Who's Responsible: Corey
Description: To be a source of light and to provide a comforting feeling for the fear meter. The light source the player has to maintain shaking the toy for an amount of time to brighten up, and will slowly dim.
Scripts: GlowToy.cs, InputManager.cs, PlayerData.cs

Gameplay System 4 - Light List (UI)

Who's Responsible: Corey
Description: A notepad-like list of remaining lights and what rooms are completed.intended as an indicator to how many lights are left and where they are.
Scripts: LightCountLabel.cs, LevelController.cs

Input System(s)

Target Platform	Input System	Who is responsible
Personal Computer (PC)	Mouse + Keyboard	Devlyn / Corey

User Interface

Add user interface designs here as well as descriptions for what they do, how they function and how our programming will affect these titular options and so on.

Splash Screen

When loading up the game. Made by the party involved with the making of the game. The player is introduced to a brief display of the logo. ultimately having no impact on the core game. Then after is the introduction of the main menu.

Main Menu

After the splash screen is an interactive design of the central navigation component, utilizing a SceneManager to alternate between setting to setting. Here are key aspects that define the UI of a main menu:

Options Setting

A place of preferences, or choices that users can customize to tailor their experience or control the aspects of Audio and Visuals.

Audio

For the most part it consists of sliders for FX volume, Music, and Master volume be a combination of the two, provided next to the sliders is a value box that could be scripted to show the numerical percentage and have the value boxes display correspond with sliders input, thus having effect on the volume output.

Alternative (if no to having input based value box next to sliders)

For the most part it consists of sliders for SFX volume, Music, and Master volume be a combination of the two. The Audios output could be scripted to correspond with sliders input, thus having effect on the volume and sharing the same principle behind all three options.

Visual

A brightness slider that is inspired by minecraft's brightness visibility in caves, where even at max, you can't see anything. Along with a single button for Full Screen and a Resolution to have a different fit to the screen.

Credits

The purpose being to shed light and give **credit** to those involved with the development of this project. Also having no impact on the core game.

Quit

The "Quit" button is the most useless when it comes to actually playing the game. Although it's kind to add it to the main menu, it stops the game's executable.

Start Game

A "Start Game" button in the menu serves as a UI element that initiates the gameplay loop. Clicking or selecting this button transitions from main menu to the actual gameplay.

Gameplay Loop

The game is singleplayer based and it gives the player itself the abilities to walk, interact and toggle their light as they take control of their environment. The player will need pre-selected keybinds to be able to interact with the environment set out for them, as well as solve puzzles, shake their portable light object and open / close (toggle) doors. The gameplay is also embedded with the major UI such as:

The H.U.D

a visual representation of a list of the remaining lights, that counts down the number of lights “on” as other lights are being turned off. As well as a meter bar that executes the lose condition when it reaches to full.

Pause Menu

Consists of the following:

Options

Transitions back to the “Option Settings” earlier mentioned.

Resume

Ultimately hides the pause menu, giving the appearance of continuing the game.

Main Menu

Returning to the main menu previously mentioned. Progress will be lost.

Game Over Display

The “Game Over” display has been made into a total result that will be shown at the end of the gameplay just before returning to Main Menu. It’s the sum of the time taken between each traversable and playable wing of the manor then adding them up for the total amount of time, and some math to give it a ranking system of some kind.

Gameplay UML Diagram