Lurking In The Dark

Made by David Taylor

Contents

[Lurking In The Dark 2](#_Toc30892091)

[Intro 2](#_Toc30892092)

[Discussion of the idea 2](#_Toc30892093)

[Design Plan 2](#_Toc30892094)

[Development Of the Game 3](#_Toc30892095)

[Monster 3](#_Toc30892096)

[Player 3](#_Toc30892097)

[Radar Effect 4](#_Toc30892098)

[GUI/UI 4](#_Toc30892099)

[Errors 5](#_Toc30892100)

[Major Issues 5](#_Toc30892101)

[Small Issues 5](#_Toc30892102)

[Reflective summary 6](#_Toc30892103)

[How did the project go? 6](#_Toc30892104)

[What Would I do differently? 6](#_Toc30892105)

[External items used 7](#_Toc30892106)

[Models 7](#_Toc30892107)

[Sound Effects 7](#_Toc30892108)

# Lurking In The Dark

## Intro

For the Project, I wanted to finally make a game that I designed when I just started making games in college. This game was called “The Cave” and the story was that a father and son would be camping out around a fire due to their car dying. The son would then be taken into a cave system by a monster and the father would enter the cave and search for the son. This would be a 3d survival game where you had to manage resources to survive.

I could have done this idea long ago, but I knew that I didn’t have enough time and knowledge to allow this project to fully succeed so I focused on learning games programming until I was confident in my skills.

The sumo digital search for a rising star was the perfect time to start on the project. It also reduced the time I needed to spend on creating a new game idea and planning.

## Discussion of the idea

Whilst looking back over my initial ideas for the game it would be too much work for the recommended amount of time for the project. Due to this, I decided to keep the generic theme so a cave, monster and a person going missing.

At this stage I came up with a few features I could implement and then researched them:

The first idea I had was Echolocation this would be a new way for the user to traverse the game world. This effect the only way I could find to create this effect would be through creating a shader. The unity shader system was completely new to me, so I had to spend time learning the system.

The Second Idea I had was Random Terrain Generation. This system uses 3d Perlin noise and marching cubes algorithm to create a smooth cave generation system. This would have increased the difficulty of the game as the map would be different every time so the notes would be in different locations.

The third feature wasn’t a feature just a goal to achieve a nice level of polish to the game. This was my third goal as I never have achieved a level of polish, I was proud of. Some ways I could make the game to feel polished is though detailed maps and animations.

## Design Plan

With the features that I wanted to include figured out, I laid out a plan. This plan was to split each feature into a separate version. Each version would be a working game that could be played and submitted. I planned out my project in this method due to only being able to work on the project over the Christmas and if anything happened like having to spend more time than expected on a feature it wouldn’t mess up my entire plan and I could just not add a feature.

Version one goals were a static map, Game Controller support, Sounds, Echo Location, Notes, Monster, Menu and Reset System. The version two goals were Random Map Generation and version three goals are Animations, Detailed levels, no known bugs and game difficulty levels.

My progress and completed features were tracked through the use of Trello. (trello.com)

## Development Of the Game

### Monster

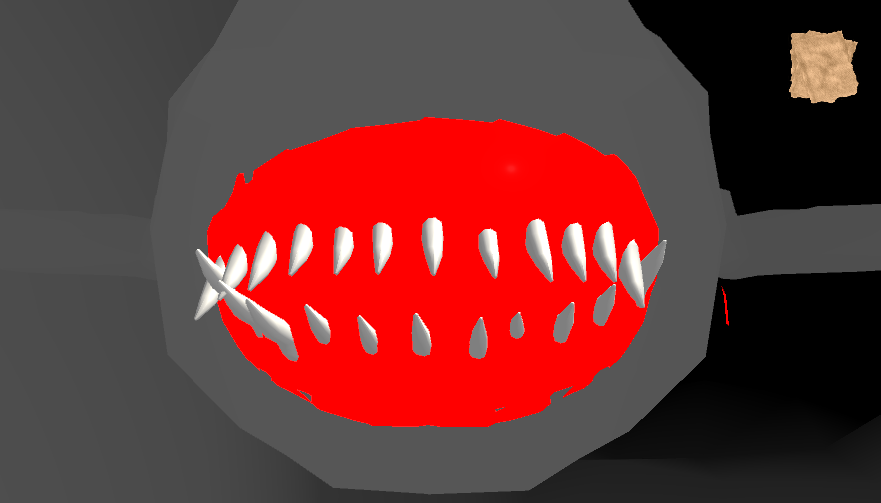
Within the monster, there are two Scripts Monster Moving and Look At Player.

The Monster Moving Script handles the movement of the monster. This script builds upon the provided Character Script and uses the DoMove(), DoGoTo() and GoTo() Functions to move the monster around the in-game world. I Have then added a Start(), Update(), GetPath(), FindClosest() and FindFurthest() functions. The movement system is based on nodes placed around the map at all intersections.

The script is set up so there are two states Active and idle. The active state finds the closest node to the player through the FindClosest() function. The node closest to the player and monster is passed into GetPath(). Within GetPath() it loops though all nodes looking for the lowest weighted nodes, this weighting is calculated by the distance from the final node and starting node. Once a node is selected to be the lowest weighting its added to a <list> named Route. Once the route is fully calculated then it is passed into Goto().

When the script is in the idle mode the FindClosest() and FindFurthest() functions are called this updates a bool within all of the nodes. Then if three seconds have passed without the monster moving the monster will choose a random location on the map to travel to. This is implemented to scare the player when the monster quickly dashes by and makes the monster feel less robotic.

The second script located within the monster is the Look At Player. This script handles Damaging the monster and the player death. The reason I have the two elements in one script is that they require similar variables to complete the tasks.



Player Death Animation

The second script is collision-based, so when you are within close proximity to the monster it checks if the player is holding their breath. If they are the monster ignores them but if there not it looks at the player and plays the death animations and sounds.

When the player is holding their breath the monster checks if the player has damaged it. If the player has damaged the monster it will play a wailing sound and one of its eyes goes blue representing one less health point. During this, it checks if the monster has any heath left if it doesn’t it plays the monsters death animation and sounds. This then resets the player back to the main menu after the sound has finished.

### Player

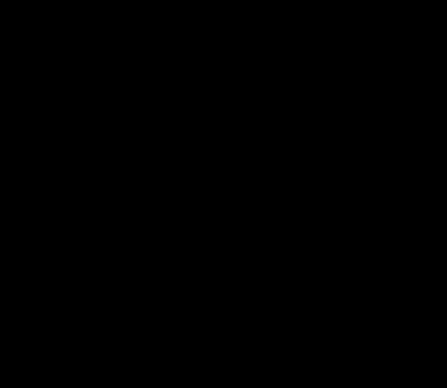
The Player has two scripts RadarEffect and FpsMovement. Within the RadarEffect the echolocation shader is controlled and fpsMovement is the movement script for the character.

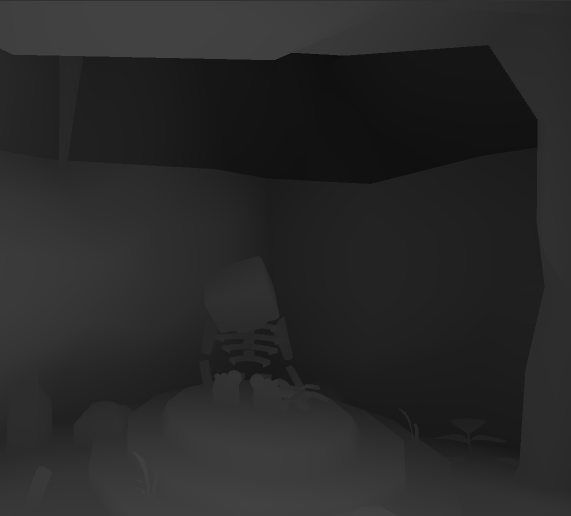
The RadarEffect Stores and Controls three radar pulses the monsters Radar, players Large pulse and the players small radar effect. Each of these effects needs a colour, width, fade, position and radius. The Radius variable is updated every frame to increase the distance from the origin point. The position variable is updated constantly for the small radar effect round the player and every activation for the large pulses.

The Fps Movement script has two main elements, the detection system to tell what type of input is being used and the movement function. The detection system loops through an array of connected controllers’ names and if any match up with the Nintendo Switch Pro Controller it checks if any of the buttons are pressed. Depending on the input type different values are passed into the movement function. The movement function handles all animations and sound effects used for movement. The inputs are read and converted to a Vector3 called moveDirection and that gets sent to the Character Controller to move the player.

### Radar Effect

The Radar Effect system includes three files two control scripts and the shader.

The control files are the RadarEffect script within player (look above for more information) and the RadarMenu this illuminates the menu, so the menu won’t be just pitch black (shown below).



Right: With RadarMenu Script

Left: Without Radar Menu

The Effect is created within the shader, how the effect is created is by calculating if the section of the game world represented by one pixel on the screen is within the radius of echolocation pulse. This calculation will give a range between 0-1 this value is then multiplied by the output colour. The output is then returned as that pixel’s colour. Before the colour is returned it checks what pulse’s value is the largest and returns that value.

### GUI/UI

The GUI/UI System is the same as the given project. All I did was to add a script Called MenuController This Handles the inputs for the controller and the mouse is handled directly through unity’s button On Click() events. The MenuController script has two main elements the same Detection System that the FpsMovement Script uses. When the Controller is active it changes an int representing the positions of the selector that appears on the buttons, this shows the user what they are selecting. All the code does is to check if the X/B button is Pressed and changes the selection respectively. Then when the A button is pressed it checks for the selection and runs the required section of code.



Choice Selector

### Errors

#### Major Issues

When creating the game, I had two major issues.

The first one was that it took longer than expected to learn and implement the echolocation using unity shaders. With the first version of the effect I wasn’t very happy with it, so I kept changing it until I was happy with it. I went through three different shaders versions one that allowed multiple pulses, one that had a small fade before the pulse and then the current effect. Theses changes happened over the entire development and since it was my core mechanic so I wanted it to be perfect. This and the next issue meant that I had to drop the level generation. This was difficult to decide as I really wanted to code the feature and play around with it but with the time I had left, I wouldn’t have been able to get it to a presentable state.

The second issue I had with development was that I was ill for a week over the Christmas break and I didn’t want to force myself to get up and work on the game. This caused me to fully ditch the idea I of random level generation.

#### Small Issues

Whilst making the game I still had some smaller errors one of these was based on my monster Moving script. This error occurred when I tried to implement a version of A\* pathfinding. My issue was that I kept accidentally creating infinite loops. This cause unity to freeze and I would have to force quit the program and restart. This error improved my debugging skills as I would have to run through the code line by line myself to figure out how the error was occurring.

Another smaller error I encountered was a glitch when swapping from controller to mouse/keyboard in the menu. The error occurred if you selected play with the mouse then used the controller to play the game. Next time you were in the menu it didn’t matter what you did it would just restart the game straight away. What I believe was happening was that unity was storing that initial mouse click and once the game started wouldn’t update that the mouse was cleared. To fix this I made the mouse click for play run through the controller menu script and only activate on mouseDown. This fixed the error but caused the controller and mouse menu systems to be connected.

## Reflective summary

### How did the project go?

I think that my game went really well, I went into this with one goal to scare one person and I was able to achieve that when showing the game to my friend. This project also increased my skills with unity before I hardly knew about shaders, controller support, project settings tab and Scripting Attributes (e.g. [Header(“”)])

The project was worth doing just to learn about unity’s scripting attributes as any new project I make can make full use of it and be fully adjustable from within the unity engine instead of having to step into visual studio to make small changes.

### What Would I do differently?

What I would do differently is take longer when making the game. This would allow me to get more feedback making sure the games feature’s made sense and that they are fun. Another thing I would do differently is to change my development plan from having versions with a new feature in each to something more standard.

Another major difference I would make if I was going to create this game again is to make the models myself this would improve my skill set as well as allowing for a consistent art style. Currently in the game the character model is very blocky, and the monster/caves are more realistic. This takes the player out of the experience when they look at the player's body then back up at the monster as the style just doesn’t match.

## External items used

### Models

Cave: <https://assetstore.unity.com/packages/3d/small-cave-kit-49372#description> by Wonder Games

Plants: <https://www.turbosquid.com/FullPreview/Index.cfm/ID/1271066> by Ditya\_Arya - Royalty Free License

Monster: <https://www.turbosquid.com/FullPreview/Index.cfm/ID/327786> by theQiwiMan - Royalty Free License

Skeleton: <https://www.turbosquid.com/FullPreview/Index.cfm/ID/620863> by seikto - Royalty Free License

Tooth: <https://www.turbosquid.com/FullPreview/Index.cfm/ID/543341> by beghelli - Royalty-Free

### Sound Effects

Player Death: <https://freesound.org/people/jorickhoofd/sounds/180351/> by jorickhoofd – Attribution License

Monster Death: <https://freesound.org/people/Jeffreys2/sounds/264038/> by Jeffreys2 – Creative Commons 0 License.

Footsteps: <https://freesound.org/people/Snudio_Records/sounds/267492/> by Snudio\_Records – Creative Commons 0 license

Cave Background Noise: <https://assetstore.unity.com/packages/audio/sound-fx/free-sound-effects-pack-155776> by Oliver Girardot

Monster base noise: <https://assetstore.unity.com/packages/audio/sound-fx/free-sound-effects-pack-155776> by Oliver Girardot

Holding breath in: <https://freesound.org/people/theshaggyfreak/sounds/316949/> by theshaggyfreak – Attribution Non-commercial License

Holding breath out : <https://freesound.org/people/ckvoiceover/sounds/401336/> by ckvoiceover – Attribution License.

Large Pulse Click : <https://freesound.org/people/Rvgerxini/sounds/475691/> by Rvgerxini – Creative Commons 0 License.