Design Document for:

The Deep Dark Hedge

The Scariest of Maze Games

Written by Zach Ahearn

Friday, December 01, 202323

Part 1 : Construct 3 Considerations

Introduction

My game is ‘The Deep Dark Hedge’ is an angled top-down game about trying to escape a hedge maze with 3 different levels of difficulty. The player must avoid the enemies or be set back to their most recent checkpoint position. They can use a cross weapon which kills or deters enemies, depending on enemy type.

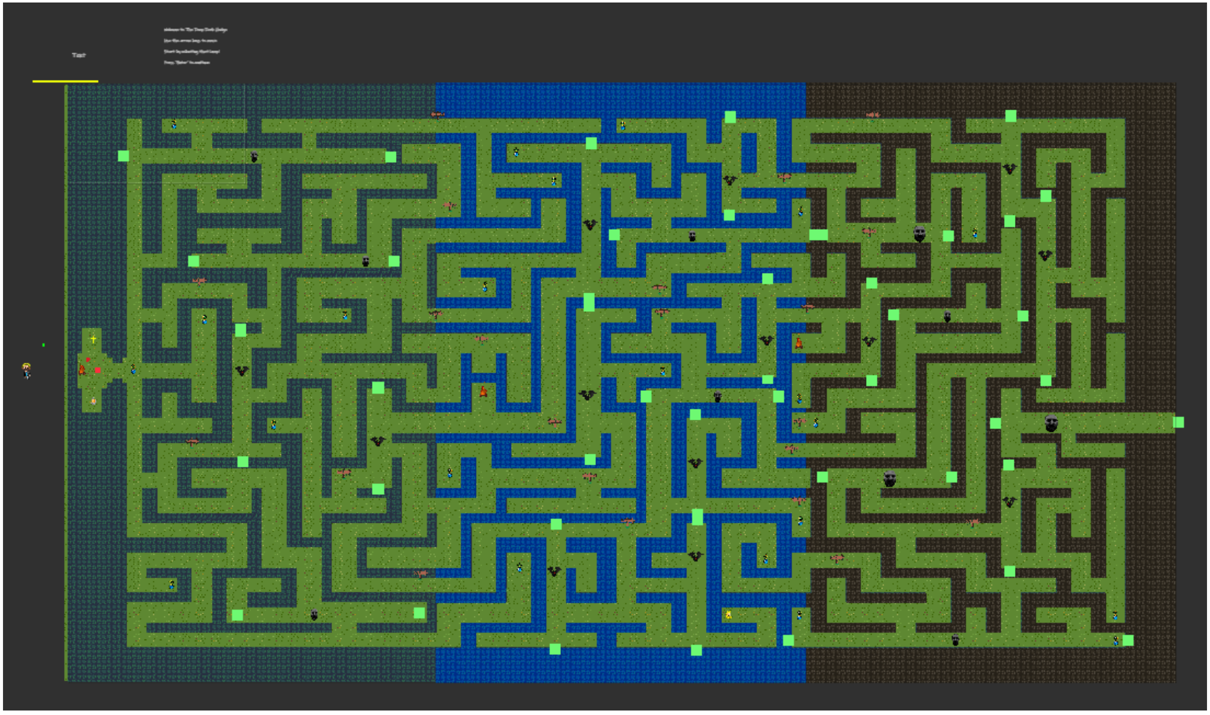


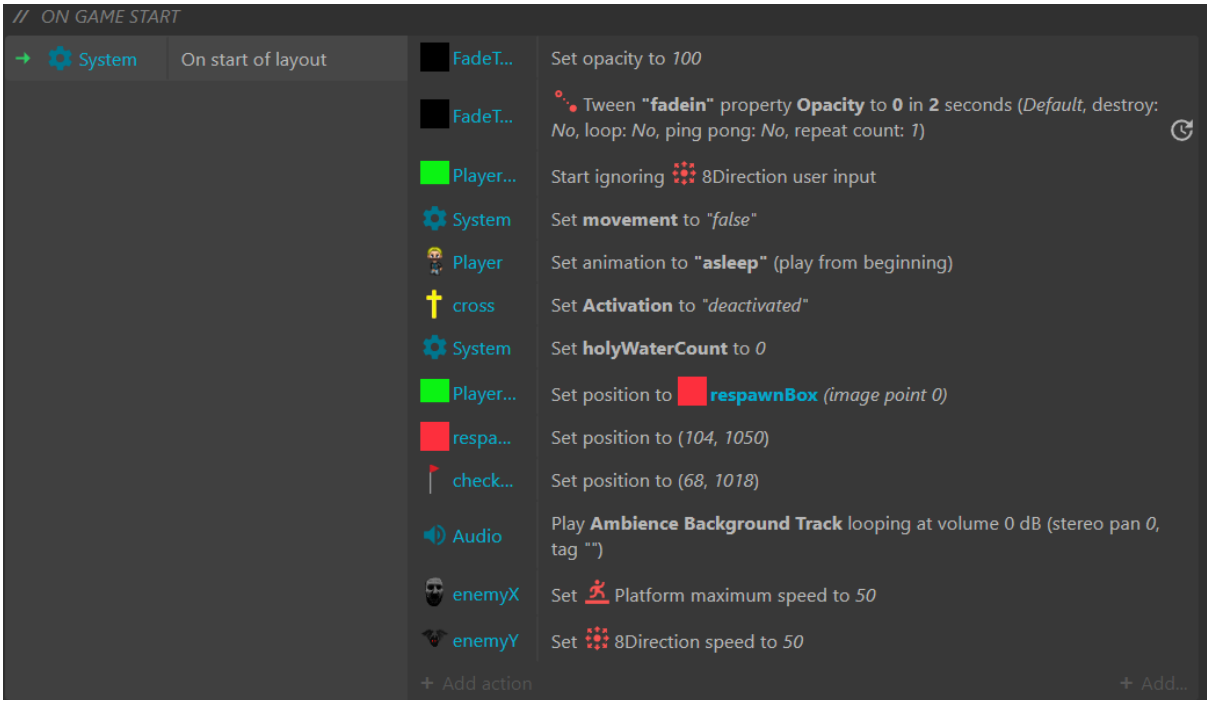
Figure 1 Full Layout

Layout

For the layout of my game, it consists of 6 layouts. The hedge layer is only used to draw the maze using a tilemap of hedge sprites. To draw the hedge itself, I found an image of a maze online and traced over it on a different temporary layer. The textures used to draw the maze change in each third of the layout. The first level is green, the second blue, and the last brown. My enemy layer holds my enemy sprites as well as the edge markers which the enemies use to change their patrol movement direction. The player layer holds most of my sprites such as the player sprite, lamp, cross, water bottles, campfire, etc. The darkness layer is just a giant, square, black sprite which I used to make the maze dark and requiring the lamp to see. It also holds separate sprites for the actual light that the lamp and cross emit. The hud layer is just used for text and the cross’s duration bar and ammo count. Finally, the fadeToBlack layer is another black sprite which I use to make the screen fade to black upon game start and player death. Figure 1 shows the full layout of the game.

Event Sheet

My game uses only one event sheet for everything. I mainly did this because the layout never changes, just its look depending how far right the player goes in the level.



// ON GAME START

System | On start of layout

FadeT... Set opacity to *100*

FadeT... Tween **"fadein"** property **Opacity** to **0** in **2** seconds (*Default*, destroy: *No*, loop: *No*, ping pong: *No*, repeat count: *1*)

Player... Start ignoring 8Direction user input

System Set **movement** to *"false"*

Player Set animation to **"asleep"** (play from beginning)

cross Set **Activation** to *"deactivated"*

System Set **holyWaterCount** to *0*

Player... Set position to **respawnBox** *(image point 0)*

respa... Set position to (*104*, *1050*)

check... Set position to (*68*, *1018*)

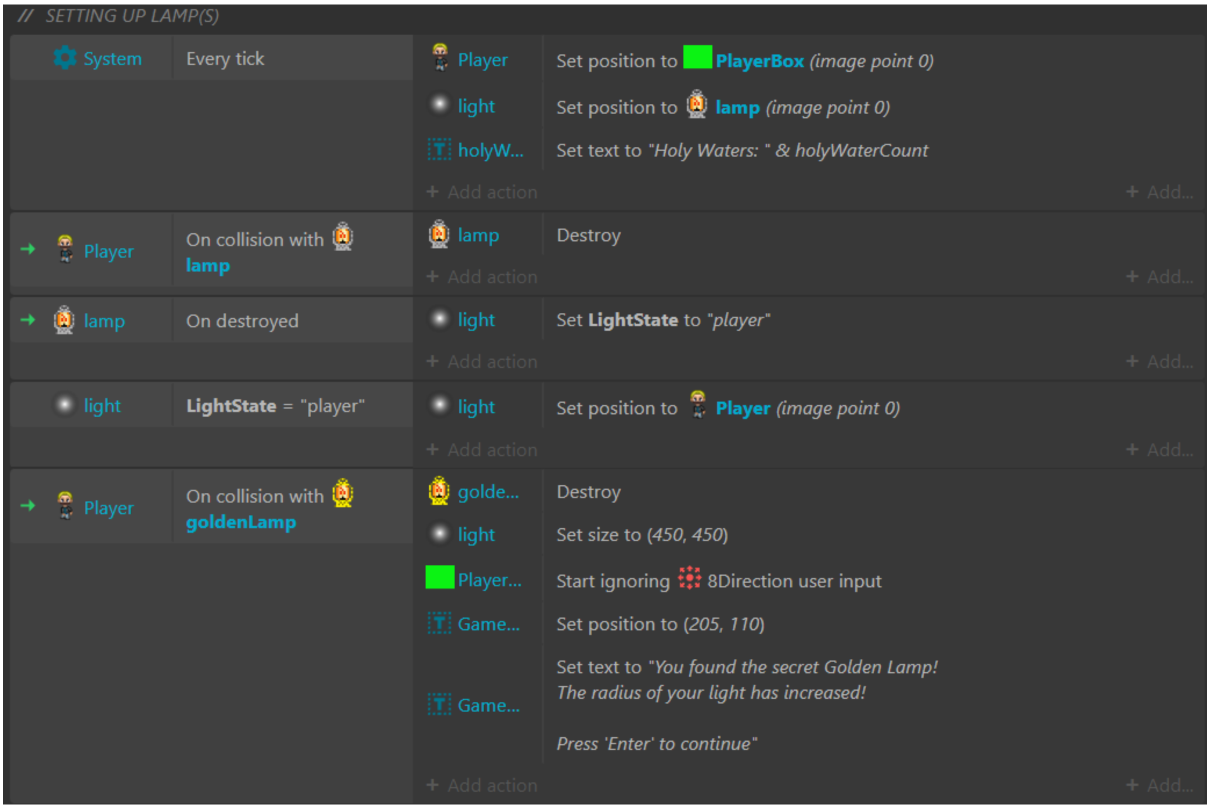
Audio Play **Ambience Background Track** looping at volume 0 dB (stereo pan *0*, tag "")

enemyX Set Platform maximum speed to *50*

enemyY Set 8Direction speed to *50*

Figure 2 Game start

My event sheet starts with a lot of actions being performed. The main point of this is to just reset variables to their original values if the player restarts the game after reaching the end of the maze.



// SETTING UP LAMP(S)

System | Every tick

Player Set position to **PlayerBox** *(image point 0)*

light Set position to **lamp** *(image point 0)*

holyW... Set text to *"Holy Waters: " & holyWaterCount*

Player | On collision with **lamp**

lamp Destroy

lamp | On destroyed

light Set **LightState** to *"player"*

light | **LightState** = "player"

light Set position to **Player** *(image point 0)*

Player | On collision with **goldenLamp**

golde... Destroy

light Set size to (*450*, *450*)

Player... Start ignoring 8Direction user input

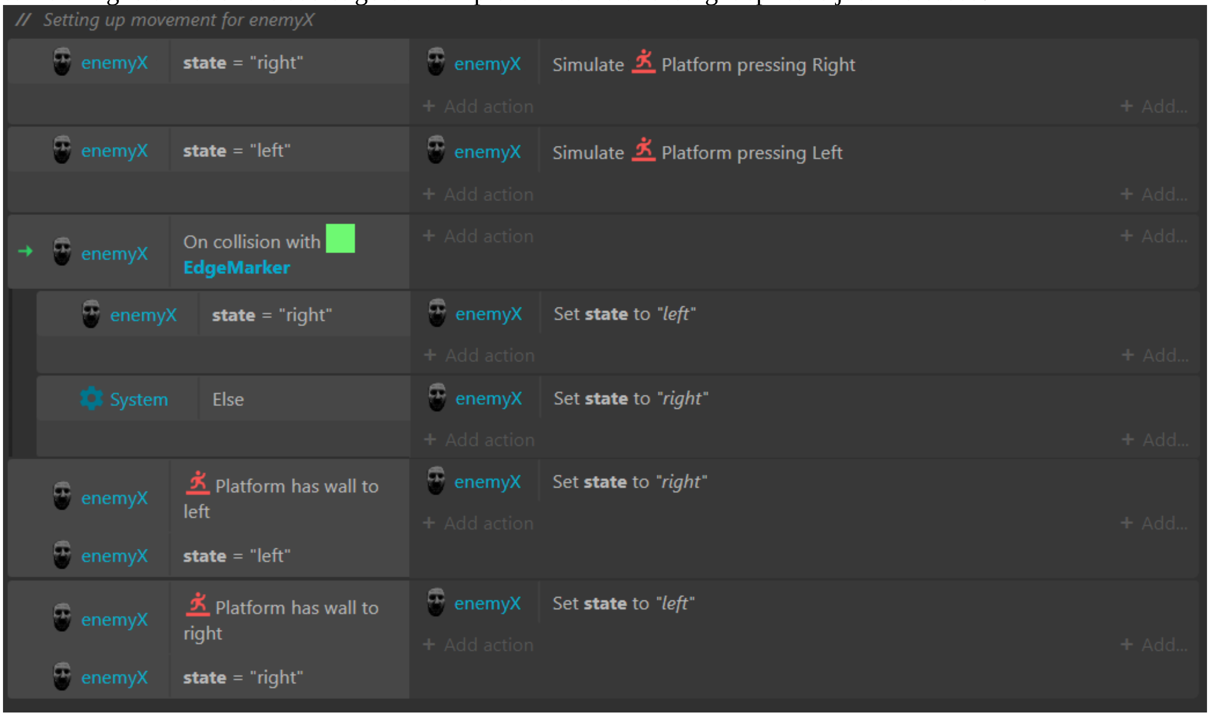
Game... Set position to (*205*, *110*)

Game... Set text to *"You found the secret Golden Lamp!*
The radius of your light has increased!

Press 'Enter' to continue"

Figure 3 Setting up the lamp(s)

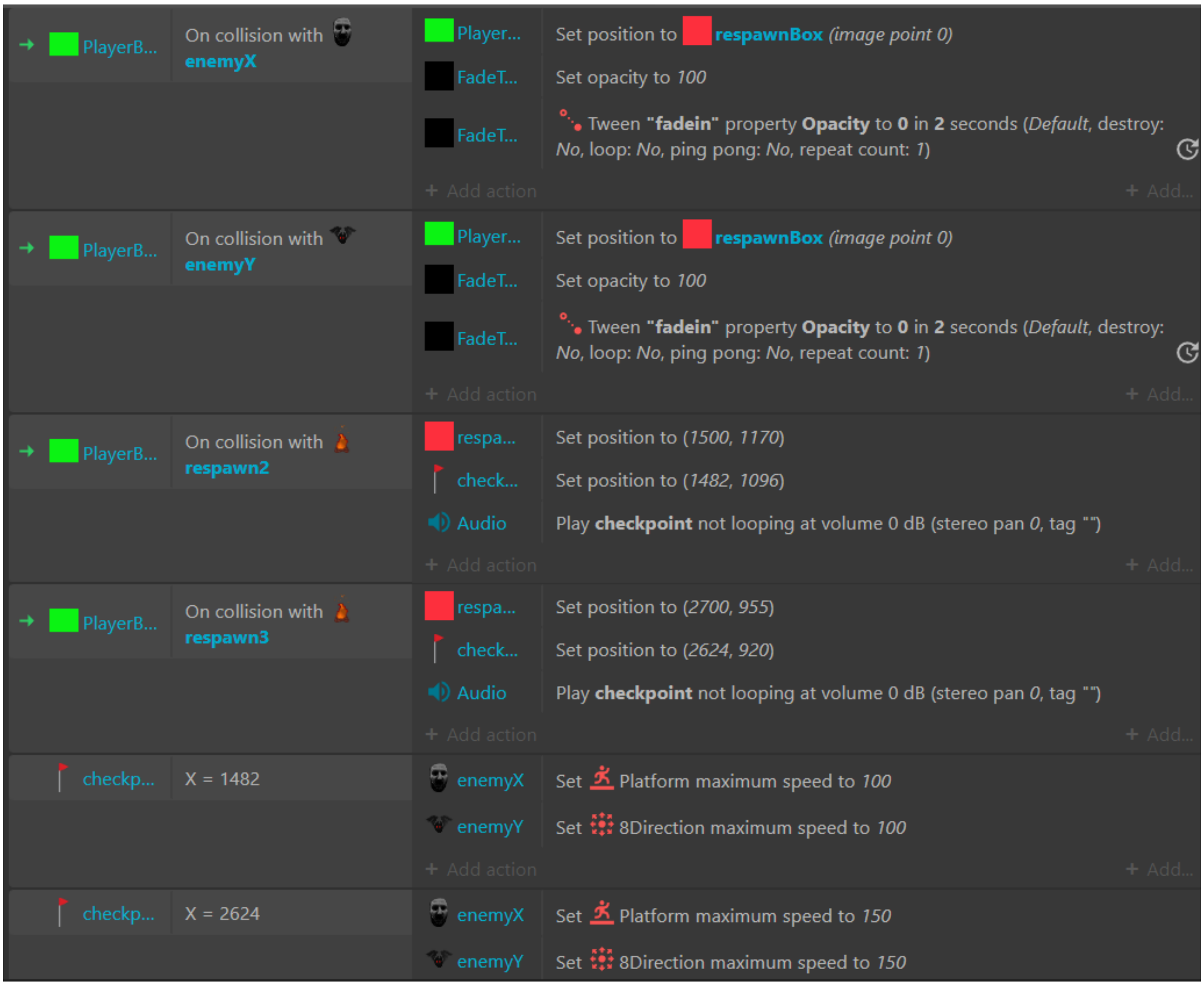
Figure 3 shows the next section of my event sheet which is about setting up the lamps in the game. When the player collides with the lamp, the lamp is destroyed and a light sprite is set to the position of the player. I did this using a string variable that when set to a certain value, executes the action. I did this a lot in my game as an alternative to using the condition of ‘on every tick’ as it basically works the same, but just more specific. The same thing is done for the secret golden lamp but the size of the light sprite is just increased.



```
// Setting up movement for enemyX

enemyX   state = "right"                    enemyX   Simulate Platform pressing Right

enemyX   state = "left"                     enemyX   Simulate Platform pressing Left

→ enemyX   On collision with 🟩 EdgeMarker
    enemyX   state = "right"                enemyX   Set state to "left"
    System   Else                           enemyX   Set state to "right"

enemyX   Platform has wall to left          enemyX   Set state to "right"
enemyX   state = "left"

enemyX   Platform has wall to right         enemyX   Set state to "left"
enemyX   state = "right"
```

Figure 4 Setting up enemy movement.

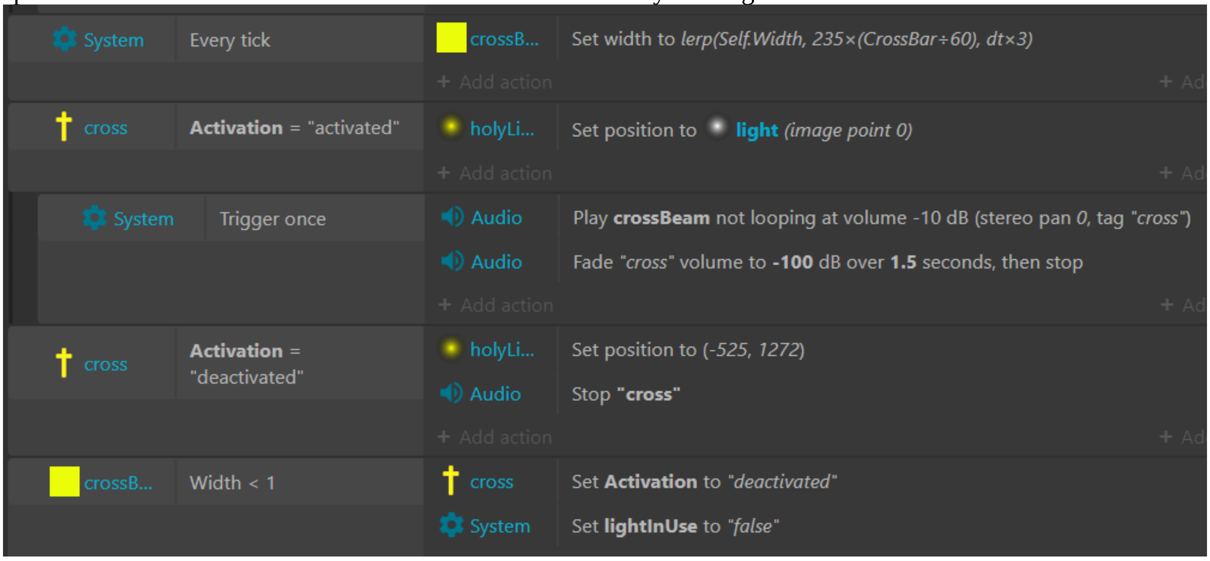
Next is my enemy movement. This is done with variables that tell the enemy what direction they should move. When they collide with an edge marker, their variable is essentially reversed, along with their movement. Figure 4 represents the events for enemyX, which is my enemy that only moves left and right. These same events are then repeated for my enemy, which moves exclusively up and down.



→ PlayerB...	On collision with **enemyX**	Player...	Set position to **respawnBox** *(image point 0)*
		FadeT...	Set opacity to *100*
		FadeT...	Tween **"fadein"** property **Opacity** to **0** in **2** seconds (*Default*, destroy: *No*, loop: *No*, ping pong: *No*, repeat count: *1*)
		+ Add action	+ Add...
→ PlayerB...	On collision with **enemyY**	Player...	Set position to **respawnBox** *(image point 0)*
		FadeT...	Set opacity to *100*
		FadeT...	Tween **"fadein"** property **Opacity** to **0** in **2** seconds (*Default*, destroy: *No*, loop: *No*, ping pong: *No*, repeat count: *1*)
		+ Add action	+ Add...
→ PlayerB...	On collision with **respawn2**	respa...	Set position to (*1500*, *1170*)
		check...	Set position to (*1482*, *1096*)
		Audio	Play **checkpoint** not looping at volume 0 dB (stereo pan *0*, tag "")
		+ Add action	+ Add...
→ PlayerB...	On collision with **respawn3**	respa...	Set position to (*2700*, *955*)
		check...	Set position to (*2624*, *920*)
		Audio	Play **checkpoint** not looping at volume 0 dB (stereo pan *0*, tag "")
		+ Add action	+ Add...
checkp...	X = 1482	enemyX	Set Platform maximum speed to *100*
		enemyY	Set 8Direction maximum speed to *100*
		+ Add action	+ Add...
checkp...	X = 2624	enemyX	Set Platform maximum speed to *150*
		enemyY	Set 8Direction maximum speed to *150*

Figure 5 Setting up death, respawn, and checkpoints.

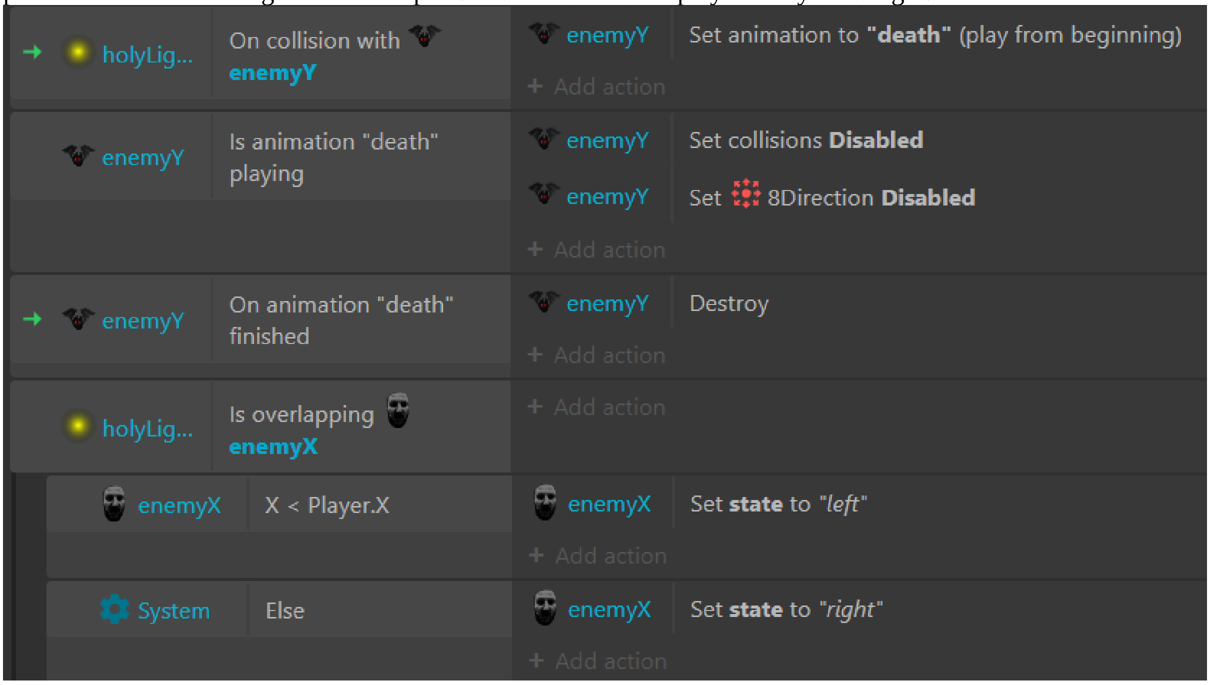
Next up are my events for when the player dies and where should they respawn. When the player collides with an enemy, the screen fades to black and they respawn at respawnBox. The position of the respawn box depends on if the player collides with a checkpoint campfire. Colliding with the campfire changes the position of the respawn sprite next to the campfire as well as the flag sprite. Depending on the position of the flag, the enemies' speed increases. This is how I handle the increase in difficulty of the game.



System	Every tick	crossB...	Set width to *lerp(Self.Width, 235×(CrossBar÷60), dt×3)*
		+ Add action	+ Ad
cross	**Activation** = "activated"	holyLi...	Set position to **light** *(image point 0)*
		+ Add action	+ Ad
System	Trigger once	Audio	Play **crossBeam** not looping at volume -10 dB (stereo pan *0*, tag "*cross*")
		Audio	Fade "*cross*" volume to **-100** dB over **1.5** seconds, then stop
		+ Add action	+ Ad
cross	**Activation** = "deactivated"	holyLi...	Set position to (*-525*, *1272*)
		Audio	Stop **"cross"**
		+ Add action	+ Ad
crossB...	Width < 1	cross	Set **Activation** to "*deactivated*"
		System	Set **lightInUse** to "*false*"

Figure 6 Holy Cross

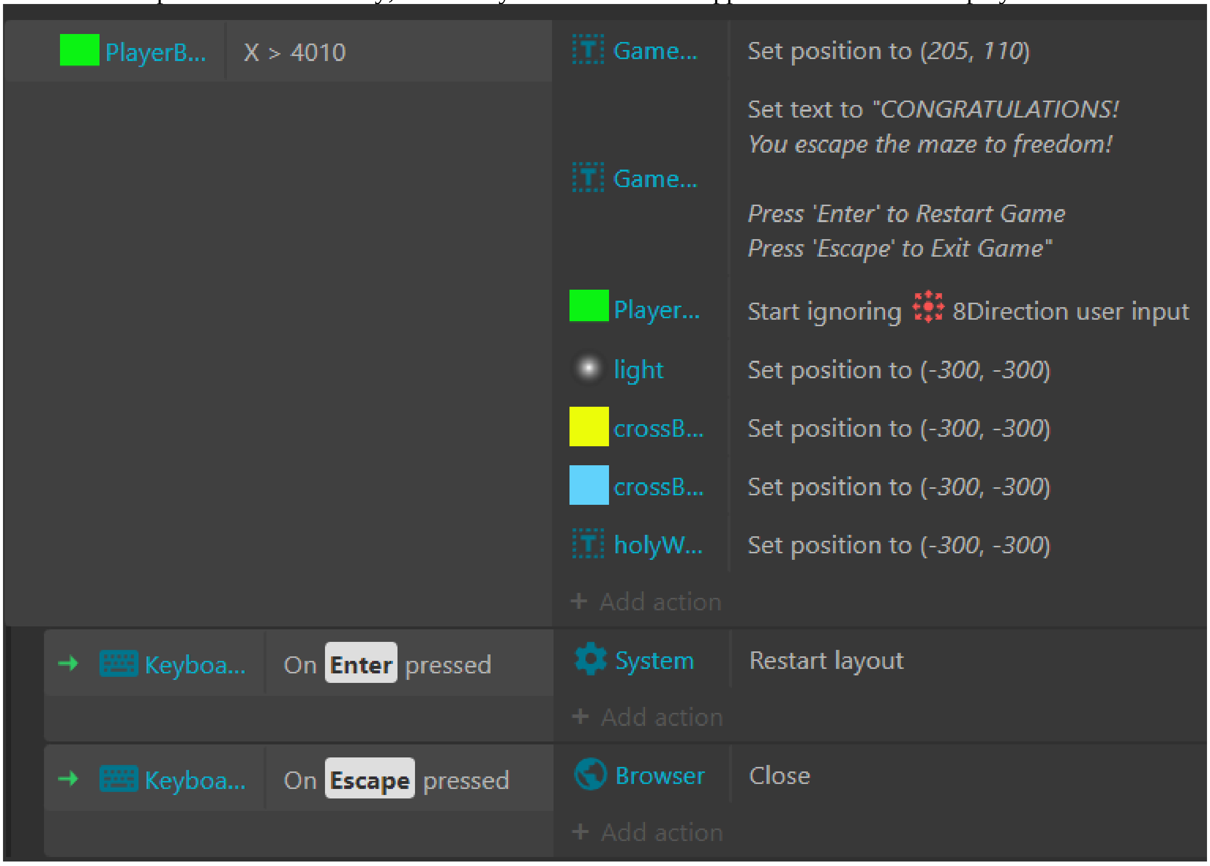
This next snippet of my events handles the holy cross's activation and duration. When activated, the yellow bar decreases. The main part I want to point out is at the top of figure 6 that uses 'lerp'. This makes the bar decreasing a smooth transition. When the cross is activated, a yellow version of the normal light is to the position of the normal light and overlaps it. This surrounds the player in a yellow light.



Object	Condition	Object	Action
→ holyLig...	On collision with enemyY	enemyY	Set animation to **"death"** (play from beginning)
			+ Add action
enemyY	Is animation "death" playing	enemyY enemyY	Set collisions **Disabled** Set 8Direction **Disabled**
			+ Add action
→ enemyY	On animation "death" finished	enemyY	Destroy
			+ Add action
holyLig...	Is overlapping **enemyX**		+ Add action
enemyX	X < Player.X	enemyX	Set **state** to *"left"*
			+ Add action
System	Else	enemyX	Set **state** to *"right"*
			+ Add action

Figure 7 holy light / enemy interaction

Figure 7 shows the enemies' interaction with the holy light. When enemyY collides with the light, they are killed, and their death animation plays. When enemyX collides with the light, depending on the player's position relative to the position of the enemy, the enemy will move in the opposite direction of the player.



Object	Condition	Object	Action
🟩 PlayerB...	X > 4010	Game...	Set position to (*205*, *110*)
		Game...	Set text to "*CONGRATULATIONS!* *You escape the maze to freedom!* *Press 'Enter' to Restart Game* *Press 'Escape' to Exit Game*"
		🟩 Player...	Start ignoring 8Direction user input
		light	Set position to (*-300*, *-300*)
		🟨 crossB...	Set position to (*-300*, *-300*)
		🟦 crossB...	Set position to (*-300*, *-300*)
		holyW...	Set position to (*-300*, *-300*)
			+ Add action
→ Keyboa...	On **Enter** pressed	System	Restart layout
			+ Add action
→ Keyboa...	On **Escape** pressed	Browser	Close
			+ Add action

Figure 8 Game End

Finally, the game ends if the player's x position is greater than 4010. The layout is 4000 pixels wide so this value is just 10 pixels outside the playable window. All sprites are moved off screen and the player is present with congratulation text. They are also presented with the choice of restarting the game with the enter key, which restarts the layout, or exiting the game with the exit key, which closes the game window.

Introduction Critical Analysis ad Conclusion

I found making this 2D game in Construct-3 to be fun and it is a good, simple engine for making games like this. One complaint I could make is that some specific things that I did in my game were difficult to figure out and I had to search through online forums to figure out how to do something by learning from what other people have done. My main example of this is my lerp code for a smooth decrease in the width of my duration bar for my cross. A similarity between construct-3 and java is how its if statement work. Events work the same as if-statements in java. Sub-events are the exact same as indented if-statements. Actions are just what happens inside of an if-statement. The obvious difference between Construct-3 and java is the Construct-3 is more user accessible and friendly. I believe someone could learn how to make a game in construct-3 much faster than in java.

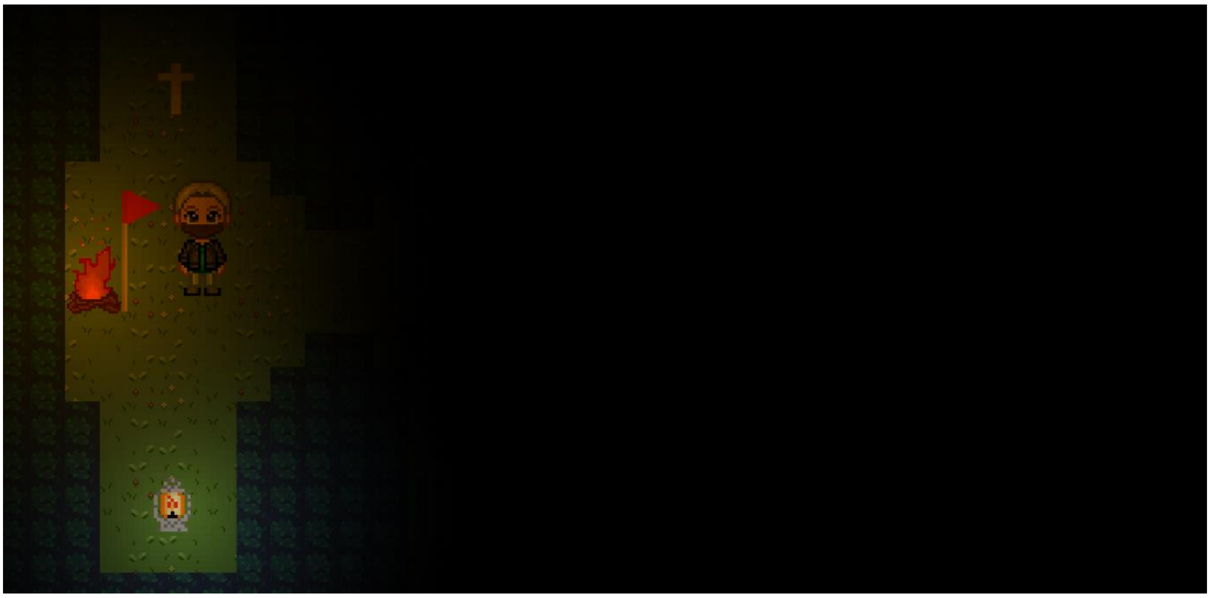


Figure 2 Game Start

Part 2 : Game Design

Common Questions

What is the game?

You wake up inside a dark hedge maze. Lost and with limited vision, you must traverse your way out of the maze, all while avoiding the monsters that roam among the hedges.

Why create this game?

I made this game because I really enjoy the horror genre and wanted my first game to fit into that section of gaming. Once I had the idea of a maze, the ideas proceeded to manifest from there.

Where does the game take place?

The game takes place in a large hedge maze which houses many enemies which roam through the tight corridors of the hedges.

What do I control?

You control the nameless main character, who you wake up as at the starting point of the maze and must navigate the maze and avoid its monsters.

What is the main focus?

The main focus of the game is to simply escape the maze by avoiding the enemies by either running away or hiding in small gaps in the hedges until the enemy bypasses you.

What's different?

One of the main differences with my game is that unlike the player normally obtaining a gun or some other firearm to defend against monsters, my game has a holy cross which emits a light to deter or destroy enemies.

Some Features

- The game will start off very dark until the player finds a lamp that will be near where they start. This lamp will emit a circle around the player which will light up the area.
- A Holy Cross will be found nearby the player at the start, but ammo will be limited, allowing to deter or kill monsters the player is particularly having trouble with.
- Monsters patrol the tight hallways of the maze with the player having to traverse around them. One way this can be done is by squeezing into a small space in the side of the hedge and wait for the monster to pass by before continuing.
- If the player is hit by a monster, they're killed and will restart at the last checkpoint.

Camera

Overview

The point of view is "angled top-down", like that of the game "Stardew Valley". It will always follow centered on the player.

Lighting

Overview

There is very limited lighting in the game, mainly from that of the player's lamp. There are also campfires in the maze which emit light, and act as respawn points for the player.

User Interface

Overview

The player's HUD is very limited for a more immersive experience.

User Interface Detail #1

A bar displays at the top of the screen which tracks how long the light from the Holy Cross will last until it is depleted. When the bar is empty, the holy light will disappear.

User Interface Detail #2

Next to the bar will be a counter. The counter tracks how many holy waters the player has, which is what the player uses as ammo.

Weapons

Overview

The player can only obtain one weapon in the form of a holy cross. When activated, a golden light appears around the player. If an enemy comes in contact with the light, they will be killed, or run away in fear, depending on the enemy type.

The World Layout

Overview

The world is split into three different sections.

World Layout Detail #1

The first 'level' of the world is an ordinary hedge maze with green leaves. Enemies are slow and less of a threat.

World Layout Detail #2

Level 2 turns the once green leaves to blue leaves. In this section enemies are a little denser and are moderately faster in their patrols.

World Layout Detail #3

Level 3 is the game at its toughest. There are many enemies, and the leaves are a sickly brown. Enemies in this level are at their fastest, so the player will have to be much more careful, especially when so close to escape.

Characters

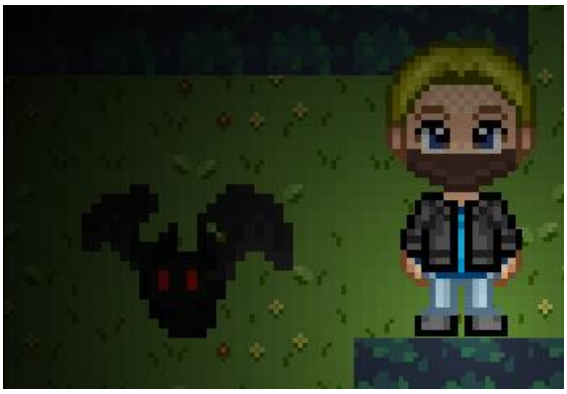


Figure 3 Player and Bat Enemy

Nameless Protagonist

The player controls a nameless man who wakes up in a maze without memory of how he got there… or even anything from before he got there…

Enemies

The maze is home to two different enemy types. Evil bats flying up and down the halls of the hedges. These foes cannot handle the light of the holy cross and will die on contact with it. Floating heads also levitate left and right through the maze. These enemies, while too powerful to be killed by the cross, will still run away in fear when the light comes in contact with their hellish skin.