Nightmare Run

Video demo: https://youtu.be/amD1Az8Wog4

Gitlab repository: https://gitlab.doc.gold.ac.uk/smada001/creativeproject

By Sandor Madacsi 13/08/21

Summary

Brief

An action based 2d side scroller game project about an old man who lost his life Energy and fell into a nightmare. The main component of the game should be to combat different kinds of enemies and obstacles to gain more energy (score) and to get out of the nightmare reality by defeating the final enemy to get his diary back.

Mechanics & Plot

The game plays in a metropolis full of machines that set to exterminate any remaining life that isn't automated. The main hero finds himself in his worst nightmare, a city without life. He needs to fight through enemies, collect energy to escape the place.

As a traditional side scroller, the player advances from left to right fighting enemies. The player can shoot with infinite ammo. The core mechanics is to survive by the end of the level as much energy points as possible. Player is controlled by:

- either 'A' and 'D' or 'LEFT_ARROW' and 'RIGHT_ARROW' keys to move
- 'Space' key to jump
- 'Left mouse 'click to shoot

Audience

As we experienced many difficulties over last year it is extremely difficult to stay positive. Life can be overwhelming, and we can get lost in our heads very easily. The purpose of the game and plot is to show no matter how bad the situation is there is always a way forward.

The main audience is anyone who likes side scrollers and pixel art and who overthink stuff.

Background Research

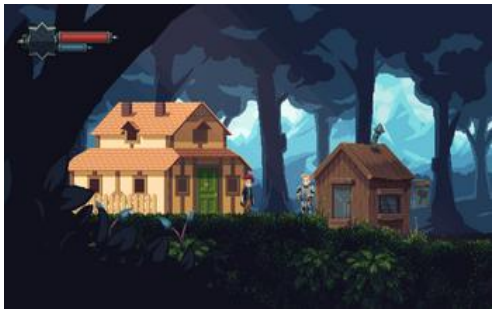
The first inspiration for the game was *Hotline Miami*. The art style, the color palette and the surreal snappy feel set the initial idea for a melancholic but fast paced platformer.

[9] IGN (2020).

For the game mechanics I relied on the basics that *Super Mario Bros.* set as guidelines. Patrolling enemies, collecting score jumping through difficult platforms. The clearly distinguishable object in Mario is a good example of asset design as each object has their own functionality either visually or mechanically if the player can interact with

[11] Gamespot(2021).

them. It is possible to achieve more with less: a simple game mechanics, like collecting coins can be more challenging than solving a difficult puzzle.



Adam C. Younis Who I referenced a few times was a huge inspiration as I follow his pixel art and game called *Insignia*. His stunning assets are truly incredible. As pixel art is limited in terms of linework and colours, he uses the limitations to create that much more powerful scenes.

[12] Uppon-hill.itch.io(2020).

The Story in *Lamentum* is something to get hooked on. The game is in a demo stage, and it is about to get released its environment and plot got me excited. I really enjoy interesting and out of the box stories.

[10] Kickstarter(2021).

Design

Art style

Pixel art was the primary style when it came to designing object sprites.

Key reasons to choose this art style:

- Limited color palette leaves less room for error when it comes of contrast in the scene
- Simple approach to body proportions let me to be more creative in designing assets
- Nostalgic feel is a bonus

Resolution

320 by 180 pixel was the best option as it looked reasonably good in full HD resolution.

Character

Initially I was too ambitious with the character design. I bumped into the first problem as the animation takes a lot of work even if a character has a size of 8 by 16 pixels. As animating the character took way more time and effort without any reward, the character became simpler, and the size became 8 by 8 pixels

Iterations:

First 

Second 

Third 

Final 

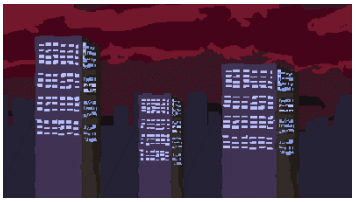
Run animation:

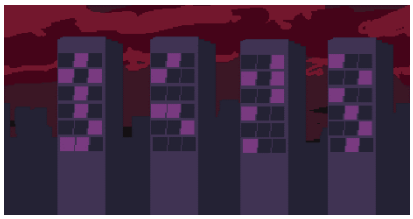
First iteration was too jagged and didn't feel right doing development. It was influenced by [16]AdamCYounis (2020).

Final iteration is less detailed therefore it looks smoother while running. It was influenced by

[18]Saultoons (2021).

<u>Environment</u>





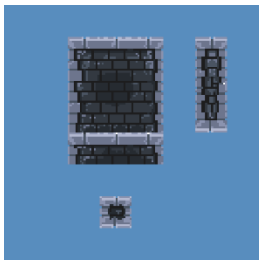
I wanted to go for a post-apocalyptic aesthetic. It should set the mood for the player that it's not a nice place to be in and danger is out there. As I never used to draw pixel art before it took some time to get a feel for the colors, but it turned out to be just enough to translate my vision.

Enemies

I tried to make the assets, cartoony but still have some personality that isn't too childish. Each enemy has their own characteristic both visually and mechanically. I tried to separate each object by color to indicate different functionality.

Collectables & Obstacles & Bullets

Tile set



Based on the very informative video from source [15] AdamCYounis (2020).

I wanted to go for a brick style tile set as it reflects the metropolis environment. I tinted the colour in Unity to fit into the scene.

Development

Tools

Unity as the game engine with C# for a few reasons:

- I already had experience with c ++ / object-oriented programming and C# is similar.
- Unity has many key features such as collision detection, physics, tagging that made the early prototyping phase way easier and I could focus on more creative aspects like the game design, character animation etc.
- Unity was compatible with JetBrains Rider (more details below)

Rider from JetBrains to write the scripts for the game. The editor has very intuitive error flags or even suggestions (like expensive method invocation that tells me if a method is computationally expensive to execute) which was very useful to refactoring code.

Aseprite to create the game assets. A well-versed pixel art editor which made the design and creation fun and compact in just one app. I used it for:

- Prototyping scenes, objects
- Character / level design
- Character animation
- Tile sets

Dev outline

The first outline that I planned out at the beginning of the project:

Stage 1

-Prototyping, setting up

-Initial assets: Character / Platform

-Basic game mechanics (moving, camera setup, jumping)

Stage 2

-Prototyping

-Second iteration of assets: Character, background, platforms, tiles, collectables, enemies

-Polish basic game mechanics

-Implement first iteration of advanced game mechanics: Change of character based on collectable

Stage 3

-Prototyping

-Third iteration of assets: Character animations

-Second iteration of advanced game mechanics

Stage 4

-Prototyping

-Last Iteration of Assets

-Third iteration of advanced game mechanics

Stage 5

-Polishing game

Stage 6

-Polishing game documentation

Stage 7

Last minute polish / submitting

Actual Project Outline

Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 7
Initial Player	Enemy	Restructure	Menus	Commenting	Constructing the documentation
Basic Game mechanics	Refined player movement	New assets	Shooting	Enemy inheritance fix	Tidy up project
-	New Jump	Separation of concerns	Enemy inheritance	3 types of enemies	Commenting everything
-	Collectables	Tilling level building	Level design	Shooting separation for player	Video of final run
-	-	Animation	-	-	-

Data structure

One of the primary goals in this project was to improve my data organization skills and file naming. I tried to be consistent throughout the development The assets are organized in categories that are easily distinguishable.

Main categories:

- Animations: player animations

- Objects: all prefabs

- Palettes: tile set palettes to draw the level with

- Scenes: level and menus

- Scripts: all game logic

- Sprites: images for all objects

- Tiles: tile objects for the palette

Subcategories:

<u>The prefabs and Sprites are stored / divided into:</u>

-Background

-Collectables

-Enemy

-Player

<u>Scripts are stored:</u>

-Collectables: collectables

-Enemy: enemy

-Other items: different ammos, hp bar, spike pit

-Player

-Systems: background and camera controls / Menus

Scripts / Mechanics

Scripting was the most difficult part of the project. As I haven't used Unity before or even C# I spent a huge chunk of my time restructuring my code. One of the main challenges were to understand how different objects or their components talk to each other via different scripts. It might be still insufficient, but I improved my code over time, and it works.

Player

As I only have one player it made sense to divide all actions into different chunks:

PlayerMove : responsible for moving the character on the x axis by the movement speed and flipping the body accordingly.

PlayerHealth: sends data to the health bar and is manipulated if enemy or obstacle damages the player. If player dies it triggers the game over scene.

PlayerFire: instantiates bullet (Ammo) objects to shoot at the enemies

PlayerJump: manipulates the body accordingly. It is triggered if the player is grounded. It has a simple and a more advanced jump algorithm for a smoother experience.

PlayerScore: updates the UI with the current energy score when player collides with the energy collectable.

Enemy

It took more adjustments / restructuring than I expected to create multiple enemies. I made a base enemy Class that all enemies inherit from. Enemies generally patrol in an area, so it was evident that all enemies inherit that behavior. Only the health, damage and the style of attack differs.

Enemy (Base): It's in a form of an abstract class so it can not have an instance by itself it has to be used in a child class. This script has all the necessary variables and object that all enemies will use such as:

-Movement speed,

-Health

-Patrolling state Boolean

-Ground checker object

-Health bar object,

-Body collider

-Transform of player position

The script also takes care of movement, collision detection (if an obstacle is in front then the enemy should decide to turn. Damages player on collision

Patroller enemy

Uses all the mechanics that the base enemy class, sets up the hp bar for the patroller enemy object

Uses faster movement speed

Follower enemy

Uses all the mechanics that the base enemy class and sets up the hp bar. It also have an attack function. The overwritten base virtual update function is extended that the enemy can decide if the player is in attack range and able to follow and flip according to the position of the player.

Gunner enemy

Different attack function. Based on the player position the gunner is shooting in the player's direction. Uses the same ammo structure but with a separate bullet and object and script.

Ammos:

Ammo of the player

Ammo of the enemy

Ammo of the tower obstacle

Collectables:

Health collectable heals the player if it's health not full already.

Energy collectable rewards player with score.

Obstacles:

Water damages player on collision, but player can jump out of it.

Spike pit damages player on collision and throws up player up and right to indicate the damage

Gunner tower: A constant stream of lasers for the player to dodge.

Menu:

<u>Start menu:</u>

-Play Button: Starts Level

-Info Button: Displays instructions and plot

-Exit Button: Exit game

<u>Game over menu:</u>

-Start again: Starts Level

-Exit button: Exit game

Pause menu:

-Resume: stopping game with escape key

-Exit button: Exit game

-Restart Button: restart Level

End menu:

-Exit button: Exit to start menu

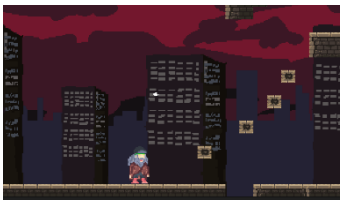
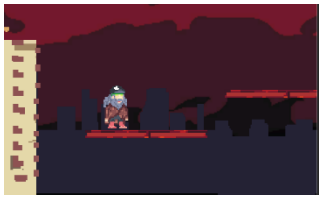
System:

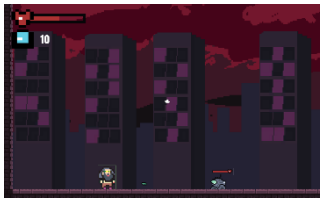
Camera system: Camera object is set on the player and following the player in a set area by a minimum and maximum value

Parallax: background animation that is based on the player object. It gives the illusion of a 2.5D environment

Project Progression



Debugging / Problem solving

Unity and Rider

I had a few difficulties using unity that became annoying habits:

-Going into play mode right after working on a script didn't always compile the code so even though I changed something the previously compiled version was playing. I became more aware of this problem as I spent more time in Unity and Rider and double checked if the code is compiled before I entered play mode.

-Switching between the arrow moving tool and the resizing tool was a problem all the time. I made too many errors by mistaking one of these tools with the other one.

-Having two many of the same prefabs or not have any at all. At the beginning when I slightly changed a prefab, I saved the new iteration as an original, so I kept piling up them rapidly or I forgot to make even one. Further down the line I learned to change the child prefabs by changing the original.

Parallax

Implementing parallax background scrolling had some weird dithering of the object while moving. Adjusting the values of the z axis to determine the far or near Clipping Plane made it smoother as I experimented.

Structure & Inheritance

The most difficult task was to understand how objects and their components interact or how scripts communicate with each other. As a starting point I made longer more dense scripts that include most of the functionalities of either the player or the enemy. As an example I had the Move(); function and it had the Jump(); inside it. Later when I wanted to use more complex logic it made sense to rethink the structure. The player then was divided based on actions. Firing, scoring, moving, health had their own functionality. As only one player object was used it didn't make sense to have a constructor or to have a base class that others inherit from.

When it comes to the enemy object, I had the same mentality at the beginning. A long script that had everything from moving – shooting. From source [13] Beaver Joe (2019). I restructured the enemy object and it made sense to have a base class that have all the essential variables and functions. I knew that I won't have too complex stuff as jumping so patrolling, moving could fit in a script without being too long. As the base class had virtual functions and cannot be used as a stand-alone game object, I just had to make 3 different enemy scripts that have their own flavour. This way it was way easier to keep track of each enemy's development progress. At the end I learned more about inheritance, virtual functions, the way to override them and most importantly to have the instance of the object working properly.

Patrolling

First iteration:

To make the enemy patrol I restricted its movement between two blocks of platform that are tagged as 'Obstacle'. The collider on the enemy would collide with the block and flip the sprite and change the movement speed negative. It worked as a basic patroller, but I couldn't extend the functionality of the enemy.

Second iteration:

Following the character in a certain radius seemed a good idea. Based on [8] Dev (2020). *I had a separate follow and attack radius. If the player entered the follow radius it checked if the player is front of or behind the enemy (playerPos > enemyPos) = Behind , (playerPos < enemyPos) =*

Front. After that the algorithm checked if it is attack range. If yes it would attack otherwise just follow. This version worked well but the code was too repetitive and confusing to work with. As I later realised, I needed a firing position to shoot at the player so by only flipping the sprite won't flip the firing position of the body. I was also restricted by the platforms as the body collider can only turn the enemy if its colliding with something.

Third iteration:

I started multiplying the speed by negative one to flip the movement on the x axis instead of assigning negative movement speed value as I did before. I also used localScale to flip the body on collision with a wall. I didn't like the fact that I am restricted between blocks to make the enemy patrol. Based on the resource [7] The Game Dev_ (2020). I started using floor checking objects to see if there is still ground at foot level. This way the enemy was able to patrol on a stand-alone platform without any restrictions. To have a shooting object to shoot from we needed to flip the body when it has collided with a wall. For these two colliders were on the enemy. One at foot level to stand on and one that extended on one side (the side where the enemy faces) that checks for obstacles. It had a funky bug because there is a small gap between these collides so I assume by colliding with the player can affect it to squash the colliders together and make the enemy to flip constantly. I haven't fixed this issue at the end because it might take a completely different approach to fix. Overall, this way the enemy can patrol and attack the player as well as detect walls.

Shooting

As I mentioned previously to shoot, I used firing position objects. It is a different question on how to direct the bullets. I instantiated the bullets when the left mouse click was pressed. Depending on if the player is behind or in front of the enemy, I rotated it by 180 degrees on the z axis.

As for the player I determined the rotation based of if the player was facing left or right. I had to fiddle a lot with the fire rate and the speed of the bullet to make it work consistently. I had one problem that is still in the game. If the player start shooting and moving at the same time it takes damage as the player runs into their own bullet. I tried to add the player velocity to the bullet velocity but that made all the already fired bullets jump up when the player jumps seconds later which is not ideal. As I think about the problem, I should have created multiple ammos that focus on one subject on collision and ignore everything else.

Diary Spawn

I wanted the final enemy to spawn the item that ends the game on collision. I was thinking about the problem a lot and I came up with a solution that isn't the most optional or elegant but at the end it worked.

Firstly, I created the script that will switch to the end scene on collision. The script is attached to the 'Diary' Object that the player will collide with. The diary object will be instantiated on the final enemy's position once the bullet object kills the enemy. So, in a nutshell I am calling the diary object in the bullet (Ammo) script rather then the enemy that the bullet kills which should be de logical way. The solution is a bit complicated as the ammo can also trigger the collision with the diary upon spawning. I haven't figured out why it happens, but if I stop shooting after the enemy dies and shoot

a bit later then the ammo will ignore the diary object. Only the player will be able to collide with it after.

Evaluation

The project turned out simpler than I imagined as I faced many difficulties during the development, I had to cut down the expectations. The game turned out snappy and fun to play but it needs more variations that might improve the experience.

I learned my way around Unity, and I am more confident using C#. I also have a better grasp on OOP and the uses of those object with each other. My data structure skills improved a lot but there is more to learn. As of time management I had more freedom because I developed this project alone. It was a 'double edged sword' as I wasn't depending on anyone therefore, I had more time to handle this project on my own and it led to slacking of more.

If I had any scripting difficulties I instantly focused on the structure as it was the key factor of making the components work with each other. I think it paid off in the long run as I wasn't building on badly structured codebase.

A good example is the enemy restructure. I started off creating the enemies with different scripts controlling different mechanics. It was the wrong approach as I wanted to use these scripts on different enemies. As I realized it won't work build a base enemy then by using inheritance, I was able to use the key features in all the enemies with a single script.

I relied on online resources to get inspiration and better understanding on the subject. I think in the future it will be much easier to start a game project from scratch. I think I made enough research to understand the bits that I implemented in the final game. I think it would be beneficial to focus on the key things in future project as I spent a significant amount of time researching aesthetic aspects that delayed the development time e.g.: Animating the player.

As I developed the project, I always tested the mechanics as I went. By adjusting variables in the inspector, Unity made it easy to test the game constantly.

After this project I am certain that I want to get myself into a new game development project as a hobby. I have a more well-rounded idea how I could add on top of this game or how I could reuse the bits I developed. Here I am listing all the mechanics and aspects that I would love to tackle:

- A dialog / inventory system to collect miscellaneous items along the way to enhance story telling

- Save system

- More obstacles and enemies

- Different levels

- Boss fights

- More detailed environment assets and animations

-Sound

Bibliography

[1] Lets Make A Game Together. (2017). *Episode 1-Project Set up – 2D Side Scroller (Super Platformer Bros)*, Unity Tutorial. [YouTube video]. Available at: https://www.youtube.com/watch?v=BdlL5bwbCiI [Accessed 10 June 2021].

[2] Lets Make A Game Together. (2017). *Episode 2-Camera & Death – 2D Side Scroller (Super Platformer Bros)*, Unity Tutorial. [YouTube video]. Available at: https://www.youtube.com/watch?v=y3-P2vktSrc [Accessed 15 June 2021].

[3] Lets Make A Game Together. (2017). *Episode 3-Basic Enemy – 2D Side Scroller (Super Platformer Bros)*, Unity Tutorial. [YouTube video]. Available at: https://www.youtube.com/watch?v=-mrGHaAdX8M [Accessed 15 June 2021].

[4] GucioDevs (2015). *Spikes – Part 14*, Unity 5 2D Platformer Tutorial. [YouTube video]. Available at: https://www.youtube.com/watch?v=YLflI90PA-c [Accessed 15 July 2021].

[5] Brackeys (2017). *START MENU In Unity*. [YouTube video]. Available at: https://www.youtube.com/watch?v=zc8ac_qUXQY [Accessed 4 Aug 2021].

[6] Brackeys (2017). *How to make a Health Bar In Unity!*. [YouTube video]. Available at: https://www.youtube.com/watch?v=BLfNP4Sc_iA [Accessed 4 Aug 2021].

[7] The Game Dev_ (2020). *Part 1, 2D enemy PATROL AI in Unity* [YouTube video]. Available at: https://www.youtube.com/watch?v=rn3tCuGM688 [Accessed 25 July 2021].

[8] Dev (2020). *Simple unity 2d platformer AI*. [online]. (Last updated 21 April 2020). Available at: https://dev.to/bounasrnour/simple-unity-2d-platformer-ai-1hp8 [Accessed 5 Aug 2021].

[9] IGN (2020). *Hotline Miami Review* [online]. (Last updated 27 October 2012). Available at: https://www.ign.com/articles/2012/10/27/hotline-miami-review [Accessed 20 Feb 2021].

[10] Kickstarter(2021). *Lamentum – Survival horror in an endless nightmare*[online]. (Last updated 22 July 2021). Available at: https://www.kickstarter.com/projects/828792638/lamentum-survival-horror-in-an-endless-nightmare [Accessed 22 July 2021].

[11] Gamespot(2021). *Classic NES Series: Super Mario Bros. Review* [online]. (Last updated 08 June 2004). Available at: https://www.gamespot.com/reviews/classic-nes-series-super-mario-bros-review/1900-6100213/ [Accessed 2 April 2021].

[12] Uppon-hill.itch.io(2020). *Insignia* [online]. (Last updated 14 February 2020). Available at: https://uppon-hill.itch.io/insignia/ [Accessed 12 May 2021].

[13] Beaver Joe (2019). *How to use INHERITANCE (C#) to create more characters' behaviors in Unity* [YouTube video]. Available at: https://www.youtube.com/watch?v=Vt_vyRpVeK0 [Accessed 15 July 2021].

[14] AdamCYounis (2021). *The Perfect Pixel Art Parallax Tutorial [and Unity script!]* [YouTube video]. Available at: https://www.youtube.com/watch?v=tMXgLBwtsvI [Accessed 16 June 2021].

[15] AdamCYounis (2020). *Tile Set Art, Pixel Art Class* [YouTube video]. Available at: https://www.youtube.com/watch?v=Q-vZ6ZvvSys [Accessed 15 July 2021].

[16] AdamCYounis (2020). *Run Animation Tutorial [Part 1] | First Pass, Pixel Art Class* [YouTube video]. Available at: https://www.youtube.com/watch?v=LPBvrdJ_1a8 [Accessed 16 June 2021].

[17] AdamCYounis (2020). *Constructing a Character, Pixel Art Class* [YouTube video]. Available at: https://www.youtube.com/watch?v=Iv4ZgH0bbSY [Accessed 10 May 2021].

[18] Saultoons (2021). *Easy Pixel Art Walk Cycle Tutorial* [YouTube video]. Available at: https://www.youtube.com/watch?v=7T6yOk5n-zk [Accessed 2 Aug 2021].

[19] Craft Games(2019). *Unity 2D Platformer Movement (Beginner Friendly Tutorial)* [online]. (Last updated 6 June 2019). Available at: https://craftgames.co/unity-2d-platformer-movement/ [Accessed 24 July 2021].

[20] Brackeys (2017). *PAUSE MENU In Unity*. [YouTube video]. Available at: https://www.youtube.com/watch?v=JivuXdrIHK0&t=422s [Accessed 7 Aug 2021].