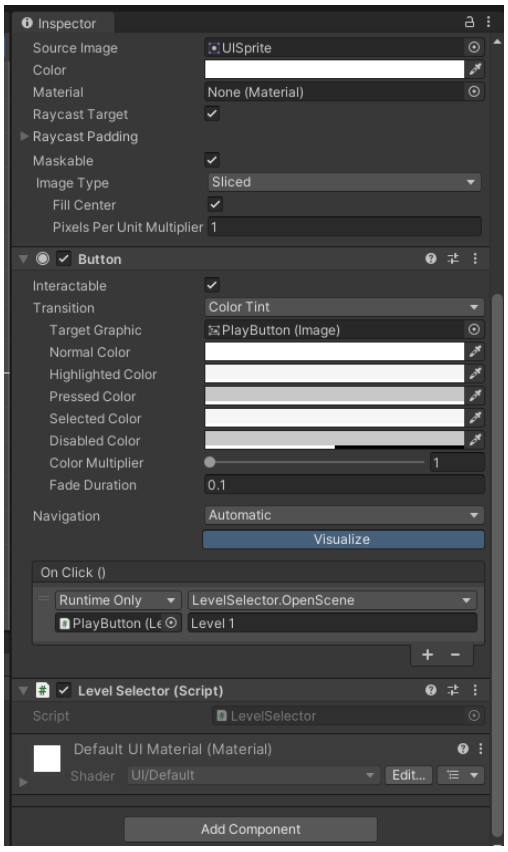
Sheng's Unity Programming Learning Journal

2024-10-15

I created my first dev log file! This is where I will talk about what I've learnt and how I dealt with any issues I have faced throughout the development of my prototype along with the tutorials I have made. To make this straightforward I will be using screenshots from my prototype to make the 4 tutorials.

2024-10-29

I am developing a simple main menu on unity and so far I managed to get the buttons to work! However there are issues which I faced throughout the development of my prototype. Turns out that I couldn't switch scenes when I press the play button on the main menu. Only to find out that I forgot to drag the level selector script in the inspector for the button.



03/12/2024

This is the day where I will implement a simple movement script for 2D! Thanks to Paul, who left tutorials on simple movement scripts I managed to improve one of my old movement scripts which I implemented a few years ago.

My old code, which did not use delta time, caused the player movement speed to be inconsistent which also made my game run less smoothly.

```
// Update is called once per frame
Unity Message | 0 references
void Update()
{
    if(Input.GetKey(KeyCode.A))
    {
        transform.position += new Vector3(-1, 0, 0) * speed * Time.deltaTime;
    }
    if(Input.GetKey(KeyCode.D))
    {
        transform.position += new Vector3(1, 0, 0) * speed * Time.deltaTime;
    }
    if(Input.GetKey(KeyCode.W))
    {
        transform.position += new Vector3(0, 1, 0) * speed * Time.deltaTime;
    }
    if(Input.GetKey(KeyCode.S))
    {
        transform.position += new Vector3(0, -1, 0) * speed * Time.deltaTime;
    }
}
```

New Movement Script which uses delta time (consistent)

```
if (Input.GetKey(KeyCode.A))
{
    transform.position += new Vector3(-1, 0, 0) * playerSpeed;
}
if (Input.GetKey(KeyCode.S))
{
    transform.position += new Vector3(0, -1, 0) * playerSpeed;
}
if (Input.GetKey(KeyCode.W))
{
    transform.position += new Vector3(0, 1, 0) * playerSpeed;
}
if(Input.GetKey(KeyCode.D))
{
    transform.position += new Vector3(1, 0, 0) * playerSpeed;
}
```

Old Movement script with no delta time (Inconsistent, and makes the game almost unplayable)

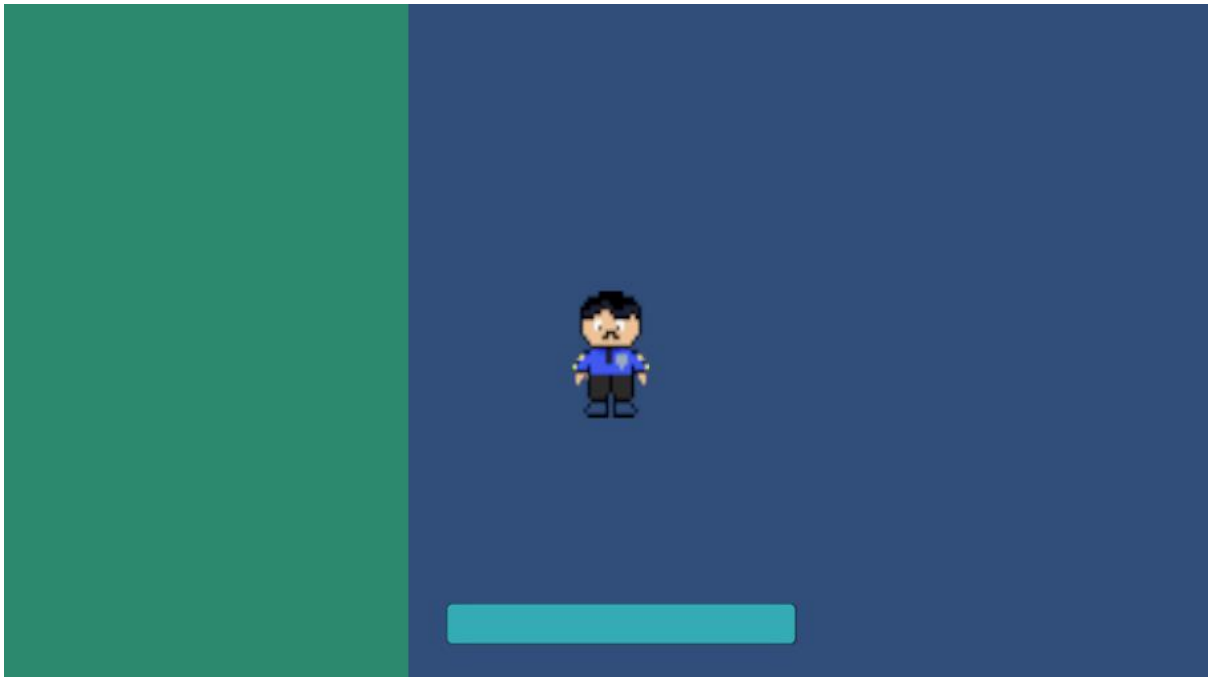
22/12/2024

Today is the day where I finally work on my player's health bar tutorial along with the prototype. It is my first time learning how to implement a player health system for a game!

So far everything is going well since I managed to assign the player's default health, and the max health the player can have in the game, however there is one issue: The player only takes damage once when they collide with the damage object.

Therefore I decided to upgrade the health system to make sure that the player takes damage over time when they stay inside the damage object (The poisonous gas).

The new updated player health system will also include: A health bar UI!!!! Now you can finally see how much health you have during your playthrough!



For the object that will damage the player, there will be poisonous gas that is there for the player's inconvenience and the moment the player collides with the gas the game will detect the player walking in and out of the gas,

therefore the longer the player stays in the poisonous gas the more health they will lose.

```
if (isPlayercollide && player != null)
{
    PlayerHealth playerHealth = player.GetComponent<PlayerHealth>();
    if (playerHealth != null)
    {
        playerHealth.TakeDamage(damage * Time.deltaTime);
    }
}
```

The image above is the updated damage script where the game will detect whether the object is touching the player. The use of delta time allows the player to take damage in consistent ticks. Without the use of delta time, will cause the player to take a lot of damage over a short amount of time to the point it makes the level impossible for the player to beat.

My game prototype is designed for the player to strategise and plan out their routes, giving them enough time to navigate their way around a simple maze on the search for assets that need to be recovered.

07/01/2025

Today, I will be working on a simple script where the player can carry a barrel of uranium which will be salvaged for resources once the player makes it out of the building alive!

I found some old code from one of my old projects and this will be used to implement a simple game function which allows the player to carry an object. However there is one small issue... The code allows almost anything to be parented with the barrel. Therefore the goal of the day is to find a way to make sure the script triggers when the player collides with the barrel and nothing else.

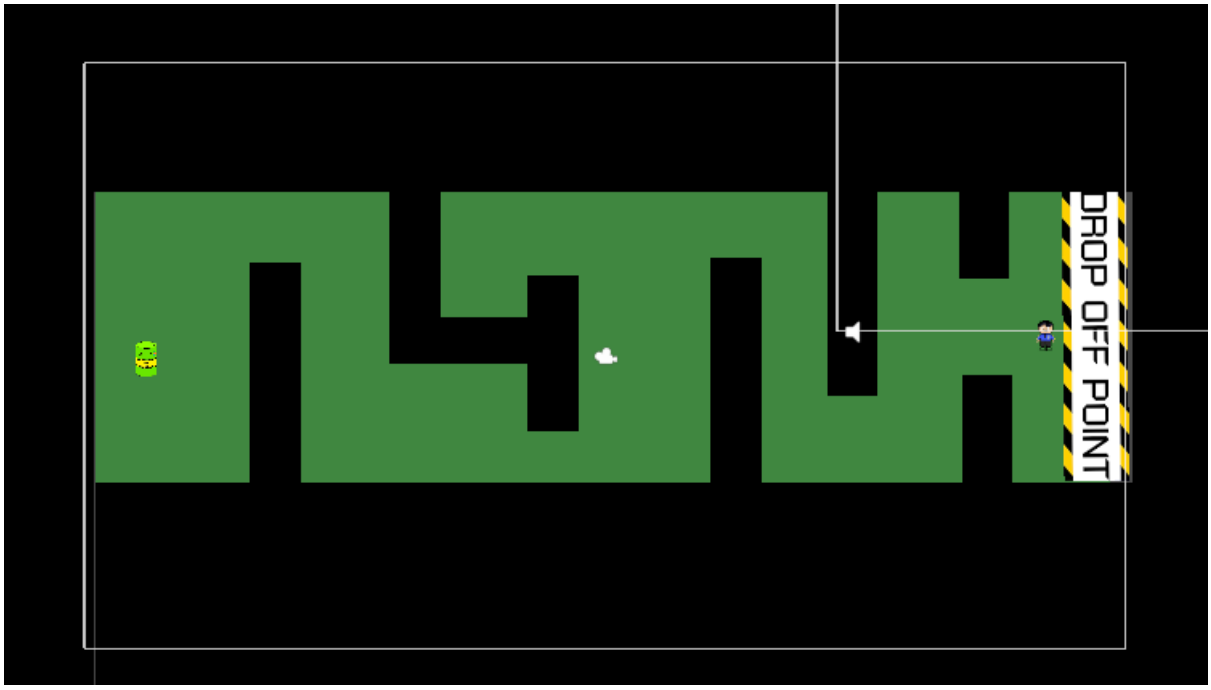
Turns out the most simple solution is to just add a “CompareTag” condition where the command triggers when the player themselves collide with the barrel.

```
Unity Script | 0 references
public class UraniumBarrel : MonoBehaviour
{
    Unity Message | 0 references
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.CompareTag("Player"))
        {
            transform.SetParent(collision.transform);
        }

    }
}
```

10/1/2025

This will be my final progress update for my game. So far I couldn't get my wall to work so I decided to go for another solution: I will be killing the player the moment they make contact with the black border in my level.



The image above is the level design. I will be making the player carry the barrel to the drop off and their goal is to navigate their way around the maze while their health is drained by the poison gas.

```
1 reference
public void TakeDamage (float amount)
{
    health -= amount;

    if (health <= 0)
    {
        SceneManager.LoadScene("Lose Screen");
    }
    UpdateHealthBar();
}
```

I also recently implemented a win and lose condition for the player. This will send them to a lose screen where they can try again or return to the main menu.

For my final tutorial, I will be talking about Win and lose conditions. This is one of the most important parts of a game. That is if your game has an ending…

It will be a simple tutorial so I won't be explaining much.

Small update: I also improved the hit box of the barrel since it will contribute to the player's hitbox the moment it is parented.

For the win and lose tutorial I have covered how to create a win/lose screen, implement buttons that let the players retry the level, and a button that returns the player to the main menu.

I then talked about the Lose condition by referring to my health tutorial (tutorial 3) where the moment the player's health goes to 0 the player will be sent to the lose screen.

For the win condition, I decided to cover a simple win condition where the player must reach a finish line to complete the level. This includes: Covering collisions for the finish line, adding a condition that sends the players to the win screen when they reach the finish line, and a brief explanation and reminder for the developers in the event they forget something when they are developing a simple game.

I will be leaving my GitHub repository link to my source files for my game prototype here to end this off.

GitHub repository link for Nuclear Lockup 2025 edition:
https://github.com/aria184/NuclearLockup2025_1.git