XV: XSCAPE VELOCITY

BUCKINGHAMSHIRE
NEW UNIVERSITY

EST. 1891

```python
def Student():
    Luke_Brighton = 22331280
    if Luke_Brighton == 22331820:
        print("SubmitWork(COM4008)")

Student()
```

Table of Contents

Team: Brayden Smith 22326622.
Luke Brighton 22331820.

https://github.com/LuxZen-Tao/BNU-COMP4008-CW1.git

Introduction

I have always believed that games are far more than just a form of entertainment. They are powerful tools for learning, outlets for creativity, and spaces where art, culture, and human connection can come together. In today's world, games have evolved into a form of art and far faster than photography did, and so now embraced as a dynamic medium for storytelling, cognitive stimulation, and social interaction my journey into game development is driven by the desire to create experiences that go beyond traditional gaming. With the potential to inspire and engage people on a deeper level. Through my games, I hope to share ideas, express creativity, and offer something that can help others grow, learn, and find a safe space to be vulnerable and truly allow themselves to feel.
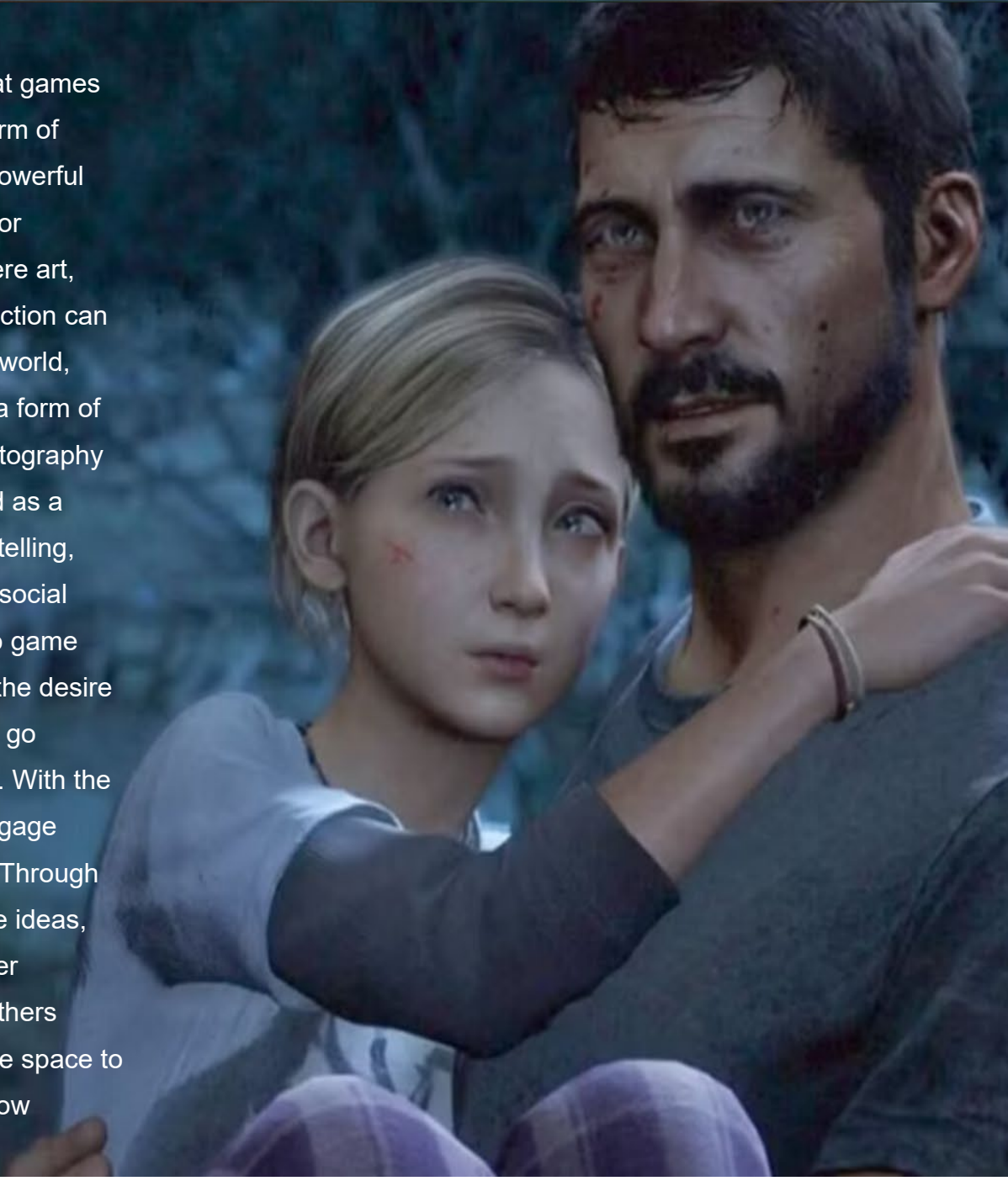
My First Experience with Unity

My initial experience with Unity was both exciting and somewhat overwhelming, reminiscent of when I first started learning Python. At first, the interface felt complex and like stepping into a maze blindfolded, but, as I began to explore, I started recognising patterns and tools that made the process more manageable. It didn't take long for my mindset to shift from "How will I ever learn this?" to imagining the possibilities of what I could create once I did.

Navigating the scene editor turned out to be simpler than expected. Unity's tools and user-friendly interface tutorials guided me through each step. Moving blocks and controlling the robot were surprisingly engaging tasks, and as I grew more familiar with the system, I began experimenting with additional features. These smaller tasks helped me solidify a small amount of knowledge and skill to expand.
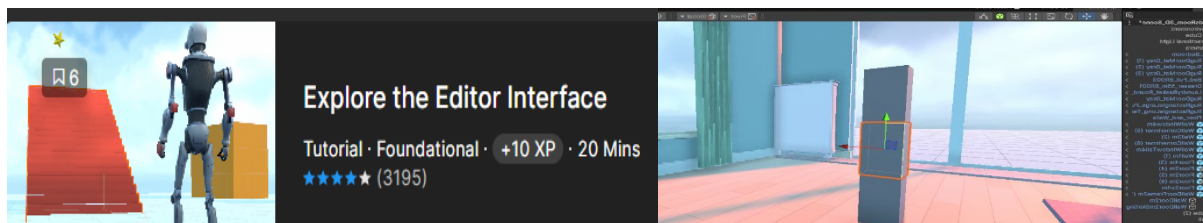
As I progressed through the tutorials, I became increasingly confident using the editor's movement, interface, and components. I started to experiment with the knowledge I had acquired, testing what Unity and I could achieve together. Each experiment contributed significantly to my understanding and skill development. I began intuitively building a level, researching C# scripting, and designing game mechanics around what was achievable at this stage. Before long, I had moved beyond the tutorials, and there was no turning back.
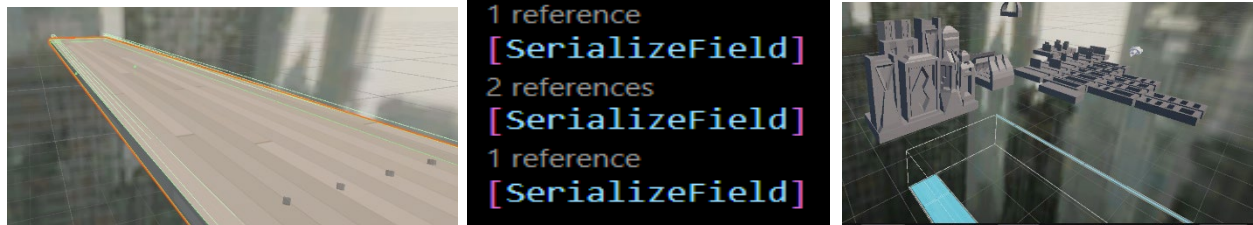

Exploring Core Mechanics for *Xscape Velocity*

Focusing on the core gameplay mechanics of *Xscape Velocity*, I concentrated on character movement using C# scripting, ensuring the controls were smooth and responsive. I utilised the Input.GetAxisRaw() method to achieve sharper movement based on my research. Additionally, I developed a functional perspective camera rotation system to dynamically follow the player while maintaining clear visibility of the action, an essential feature of the gameplay.

Environmental interaction became another key focus. I experimented with mechanics such as object detection, triggers, and interactive NavMeshes to achieve the desired behaviours. This turned out to be more complex than anticipated, but I was determined to persevere. Initially, I attempted shrinking colliders, testing static and kinematic settings, and adjusting gravity. Even placing tiles to appear connected did not return the results I hoped for.
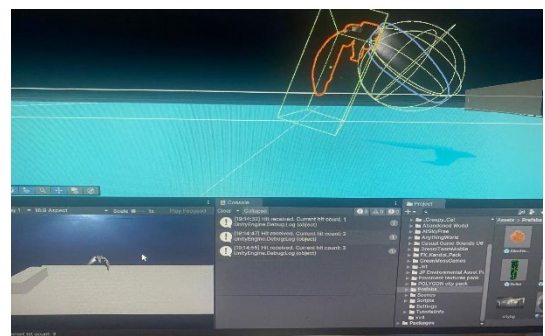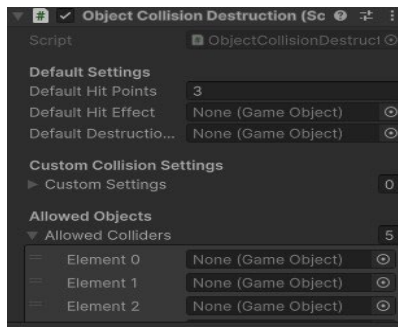
Through research on platforms like GitHub and the Unity scripting API manual, I discovered that Unity allows for almost limitless scripting possibilities, including colliders and other components. This revelation opened new doors. For instance, I scripted colliders to interact only with predefined game objects as a temporary solution. While not ideal in the long term, these early developments provided invaluable insights into Unity's functionality and laid the groundwork for shaping *Xscape Velocity*'s gameplay.

After resolving the world mechanics, I began exploring Unity's animation system at a basic level. Using the animation tab and keyframe bar, I created actions such as moving walls and spinning items. I even experimented with animating the player craft to emit an exhaust blast. Although I didn't delve too deeply into this area, it gave me a good understanding of how the animation system could enhance the game.

Developing the Combat System

Creating the combat system proved to be a fluctuating process, working seamlessly one moment and breaking entirely the next. I quickly realised how sensitive features could be when scripting everything from scratch. With my limited knowledge of C#, I reminded myself not to fall into the Dunning-Kruger effect—I don't know enough to fully grasp how much I still don't know.

One success was making smaller enemies spawn from a main enemy and follow the player. This worked well during initial tests, but in the current build, it faltered. The issue may have been due to interference between the NavMesh and the follow-object script or errors in the Instantiate code. Regardless, game development takes time, and I recognised the need to prioritise other elements.
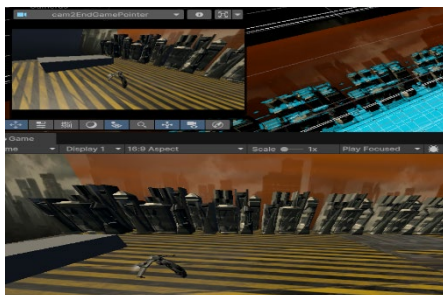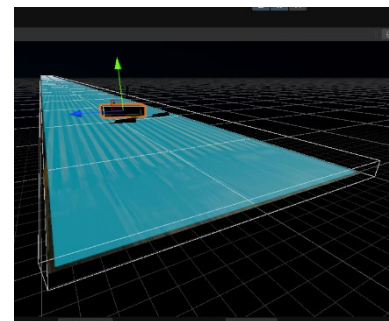
Although ambitious for someone with no prior knowledge of programming or Unity, I'm proud of the progress made. The current build includes basic player and enemy projectile firing that needs fixing, as well as stage destruction. While the combat system and certain features, such as power-ups, remain incomplete, I hope their documentation demonstrates their inclusion during development.

Advanced Features and Customisations
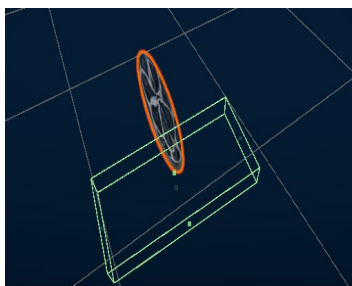
As the game evolved, I explored additional mechanics such as item pickups, enemy interactions, and combat enhancements. This required learning more about Unity's built-in components, Rigidbody physics, and implementing interactive elements like bullets colliding with enemies and obstacles, bullets ricocheting, and enemies firing projectiles. Scripting these interactions presented numerous challenges. For instance, layering components and scripts sometimes cause conflicts, akin to asking the computer to go north and south simultaneously.

Resolving these issues involved countless hours of testing, undoing lines of code, and identifying which components worked well together.

To address issues such as enemies being destroyed immediately after spawning, I scripted hit points for individual enemies and environmental elements. This added depth, as hazards could now vary in difficulty. Some of my most significant discoveries occurred through experimentation,a process that often revealed solutions I hadn't initially considered. For example, learning about procedural generation from a peer inspired new ideas for future projects.

While item pickups were partially implemented, integrating these scripts with my existing code was far from straightforward. After multiple failed attempts, including rewriting the scripts, I decided to move on due to time constraints. Similarly, combat mechanics required more effort than expected. Creating a bullet prefab involved configuring components like mass and gravity, instantiating objects correctly, and scripting collision behaviours, all while avoiding friendly fire. Unfortunately, some of these features were not polished enough for inclusion in the final build, but they represented valuable learning experiences.
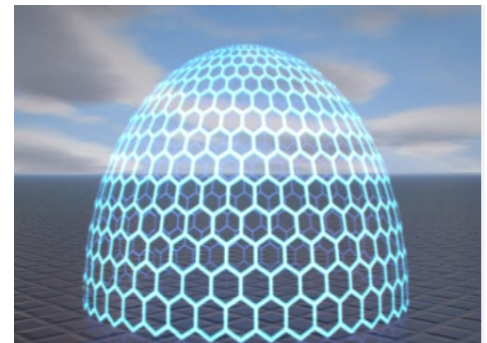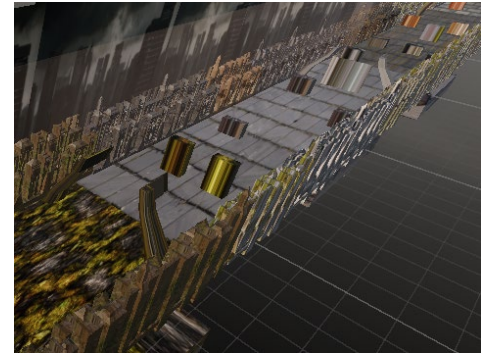

Testing, Optimisation, and Polishing

In the final stages, my focus shifted to refining the game's performance and addressing bugs and glitches—of which there were many. This involved optimising scripts, ensuring smooth gameplay outside the Unity environment, and polishing the visual and audio elements. Unfortunately, I discovered Unity's Profiler and performance testing tools too late, having diverged from the tutorial path earlier than I should have. My enthusiasm to start building was both my driving force and my folly.
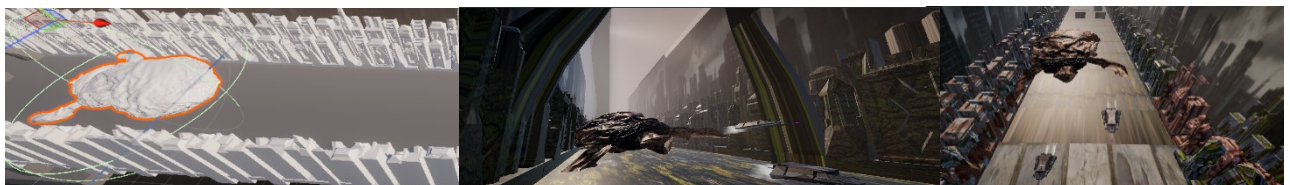
The process involved endless testing—adding elements, tweaking scripts, and troubleshooting issues. For instance, a late-stage bug caused the player object to get stuck on an invisible wall. Despite trying various fixes—from resizing and rescripting to removing meshes entirely—the issue persisted. I suspect an early component or script conflicted with recent additions, but resolving this exceeded my current knowledge and available time. One could say that this process tested me in a way, and I found the not so fun and 'glamorous' side of digital creation. Lots of small tweaks and attention to detail is required, even for a small project, and if I want smooth stability and working implementation, I need to be ready for hours of meticulous work; but like most things that are difficult, it will be worthwhile.
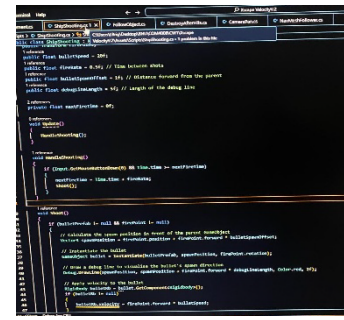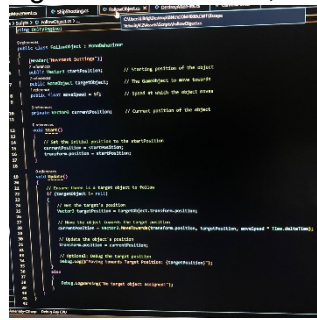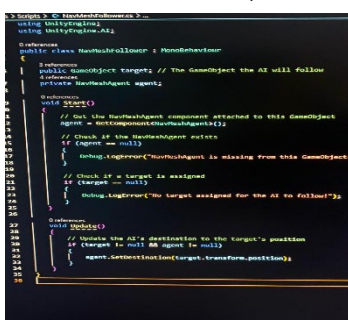
Reflections and Next Steps

This assignment has provided me with what I feel is a wealth of knowledge relative to when I started. I have gained some understanding of programming languages such as Python, developed a minor grasp of C#, and significantly improved my skills in Unity. From basic game mechanics to advanced system design, each challenge has deepened my understanding of game development and computer programming.

I have written a full conclusion and Problem and solution documents and added it to the repository, but for a quick summary I would say, looking ahead, I plan to continue refining my skills with Unity and coding languages as I continue aspiring to create immersive worlds, memorable characters, and emotion evoking stories, with a little food for thought on the side. Though this project was ambitious, it has strengthened my resolve and reaffirmed my belief, that, 'it's not what you do…but why you do.'

Luke Brighton
22331820