AKDENİZ UNIVERSITY

FACULTY OF ENGINEERING
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

CSE 491
SENIOR DESIGN PROJECT
FINAL REPORT

Supervisor: Asst. Prof. MUSTAFA BERKAY YILMAZ

January 8, 2023

Selim Aybars Duran - 20190808019
Bengisu Şahin - 20180808052

Contents

1 Group Members and Roles

Selim Aybars Duran - Game Developer and 3D Artist
Bengisu Şahin - Game Developer and 2D Artist

2 Introduction

We are two students and independent game developers who studying computer science and engineering at Akdeniz University. For the Senior Design Project course, we will develop a computer game. This game is going to be a survival game based on a story. This game type is very improvable and liked by people. First of all, we need to determine what kind of game we want in our minds. We need to decide how the game will look (First Person, Third Person, Side, Top Down, etc.), what the story of the game will be and which game engine we will use. In addition, we need to know the language used by the engine. Besides the game engine, we will also need other programs to design models. Such as Autodesk 3dsmax, Blender, and Adobe programs.

2.1 What is the Independent Game Development?

Independent game development process made by one person or a small team, especially one without financial support from a publisher. Independent game developers are without any restrictions in terms of creativity, mostly in a home environment or in small offices, with their own means and ideas. The income of these small communities depends on the supporters they have, the advertising promotions they can do as their budget allows, and of course the quality of the games they make. There are indie games, which are short for independent games that have very high incomes and even crushed the sales figures of the games of some of the big companies. These are:

Minecraft – Mojang

Terraria – Re-Logic

Undertale - Toby Fox

Stardew Valley - ConcernedApe

Below, there are two figures that show the advantages and disadvantages of indie game development. [3] (Figure 1) and (Figure 2)

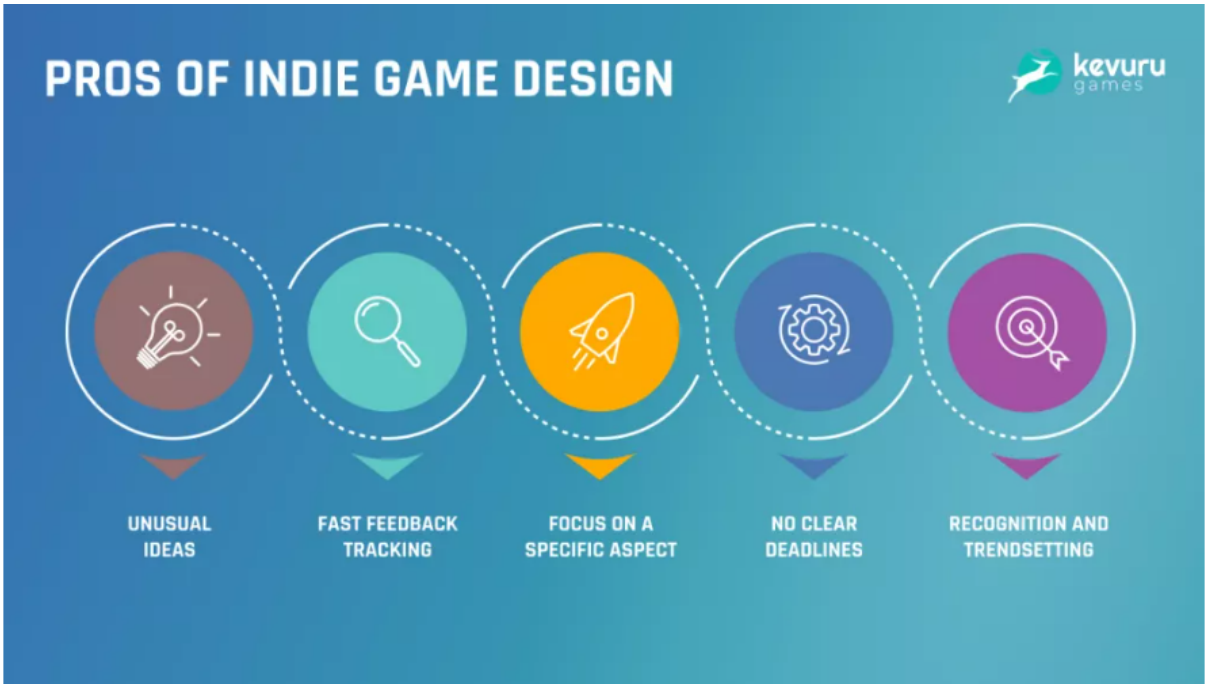

Figure 1: Advantages of Indie Game Design

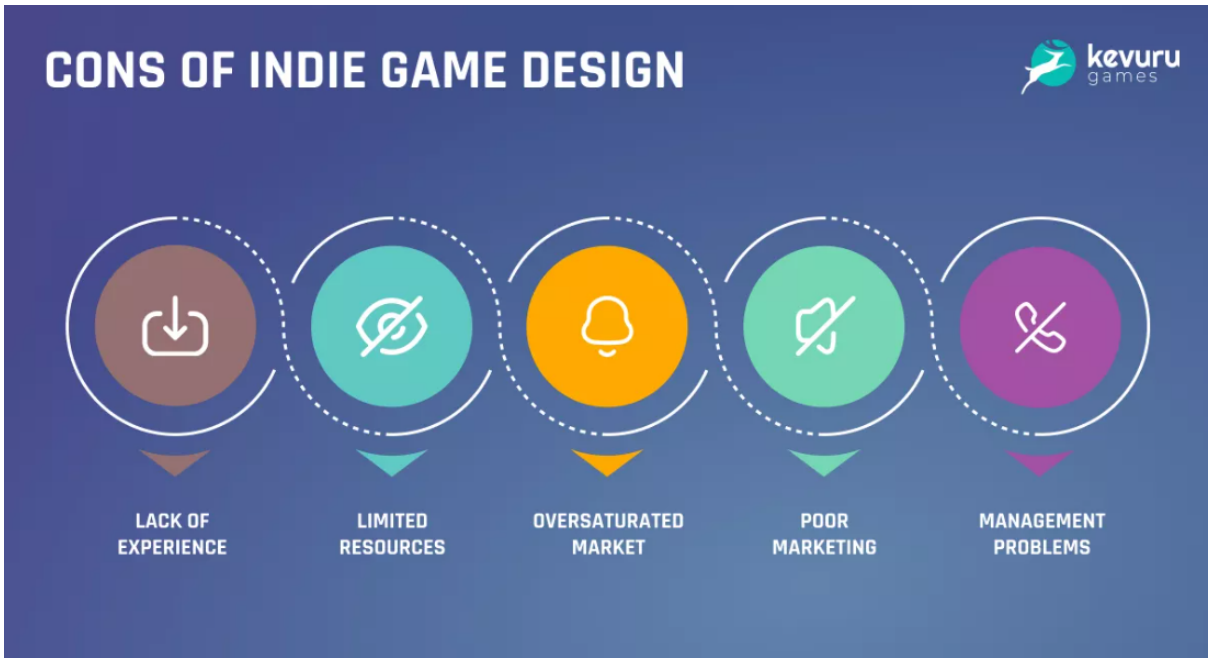

Figure 2: Disadvantages of Indie Game Design

3 Problem

The problem is the "Proper Game Engine". The game engine is the framework and ready-made skeleton used to create a game. Developers with the help of game engines; can easily create much faster and more advanced games for computers, consoles, and mobile devices. Also, we can describe the game engine as the heart of a game. In fact, we can summarize the game as the decorations made on this engine, the added story, sound, music, map, and overlays. The sine qua non of a game engine includes displaying the game map, models, and other graphic elements to the player, (collision detection: did you shoot someone, did you hit someone, where did you hit, how much health did you hit, etc.), things like playing audio and music files at the right time can be counted.

3.1 Definition of the Problem

The problem definition is "Why we don't build a game engine and how do we choose a proper game engine?". Building a game engine from scratch means hours of professional work, which must be done by teams and highly advanced people in software languages. Even the biggest game developers of today are aware of this situation, so instead of building their own game engines, they are taking the most important part of the workload off their shoulders by using the game engines that are already in place.

3.2 Difficulties

Difficulties of building a game engine:

1 - Rendering
2 - Math Calculations
3 - Input Taking
4 - Frame Timing Control
5 - System Initialization
6 - Takes Too Much Time

Difficulties of choosing a proper game engine:

1 - Does the game engine to be used charge a fee?
2 - What is the cross-platform support level for the platform(s) where the game will be released?
3 - Is the content of the documentation sufficient in terms of information?
4 - Which programming language is used for the game engine will use?
5 - What scripting language support is used if any?
6 - Is the interface of the game engine easy to use?
7 - How is the community support that the developer will interact with in terms of activity?

Difficulties of indie game development:

Graphics/Sound/Playability, including 2D and 3D, will vary depending on the size of the Indie Studio, the type of play it wants to do, and its budget. Since most indie studios have a limited budget, they try to do these things with the support of funding.

In one-man studios things are more difficult, the developer has to either make or buy the Graphics, Programming, Music, and Sound Effects itself. Stress and work tempo are very high when the developer does it himself/herself.

4 Comprehensive Survey

4.1 Survey About the Problem

Game engines are the heart of a game because you build the game on the story and characters you design and jump, run, and fly; It is the software that blends the data required for the game such as the light and shadow adjustment of the scenes, which allows physical movements such as explosions, collisions, gravity to be displayed on the screen, the sound of sounds, the editing of commands with codes, the use of AI (artificial intelligence) modules. The game engines actually provide the necessary optimization for the background game, and in some game types, creating a feeling for the player as if the character and object in the scene reflect this feeling to the player.

There are two game engines in competition among many game engines in the market. These are Unity and Unreal Engine.

4.1.1 Unity

Figure 3: Unity

It has a multi-platform capability and was created in Denmark by Unity Technologies. It produces 3D games and simulations more effectively than the available game engines. It is possible to create 2D games as well, although the quality is inferior to that of 3D game development. Only C# is used by Unity. The 2020 version now supports the Bolt plugin and does away with the Boo and UnityScript languages. Objects have pre-coded attributes that can be enabled by dragging and dropping them. You can purchase or download a variety of tools from Unity's huge asset store, including 2D and 3D model animation, scene, particle, lighting, shadow, ground, tree, rock, terrain, and sky-compatible add-ons. For the creation and development of games across several platforms, the Unity game engine may be preferred (PC, console, mobile device). Mobile platforms and VR (Virtual Reality) technology see the most success with it. The in-depth and instructive tutorials and courses can help you learn how to use the game engine in greater detail. Because of its simplicity compared to Unity's capability, it is preferred by software developers and is not difficult to use. Students who sign up for Pro accounts from Unity receive them for free. In the free version, there are no limitations and only the Unity game engine's logo is visible when the game is opened. Purchasing the Pro version is required if the yearly revenue is more than $200,000.[1] In terms of utilization, it is the game engine with the greatest community. When an issue arises, a solution is found promptly and without difficulty. Compared to other game engines, the documentation has more content. In comparison to Unreal Engine, Unity's graphics are inferior and it is unable to produce realistic-looking effects. Unlike previous game engines, this one integrates graphics and code, speeding up development and offering the developer a flexible workspace. There is a business license available. Small projects and the creation of mobile games are better suited for their use. It is not appropriate for the kind of massive projects we refer to as AAA games. The Unity game engine has an advantage over Unreal Engine when it comes to recruitment processes compared to Turkey.

4.1.2 Unreal Engine

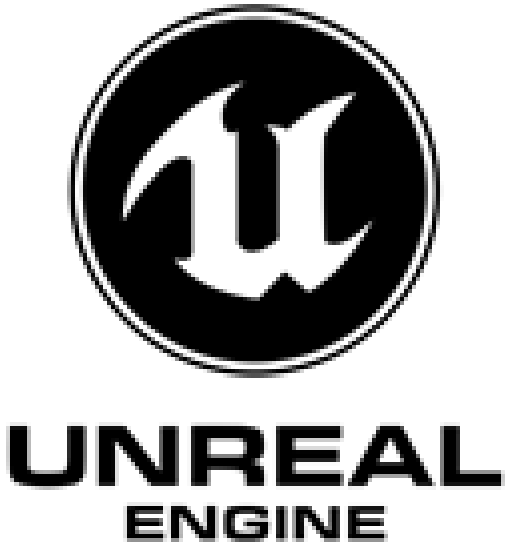

Figure 4: Unreal Engine

It is the engine that Epic Games created. Because it has also been employed in the film business, Unreal Engine is no longer merely a game engine. As a result, artists use it as an engine that is friendly to them. The language utilized is C++, and the drag-and-drop visual programming interface, which we refer to as Blueprint, is employed. Visual scripting can be used to create some games, however the Unreal Engine foundation is quite sophisticated and challenging to understand. Unreal Engine is slower and more challenging to use for game development than Unity. First-person shooters and third-person shooters are the game genres with which it has found success, and it is appropriate for creating 2D and 3D games. Compared to other game engines, it has the best graphics, making it possible to create AAA games, or what we refer to as massive and complicated games. As opposed to Unity, it has a smaller community. It supports multiple platforms. The license type is a commercial license. Unreal Engine 5 was unveiled in May 2020 and is anticipated to be released near the end of 2021. Currently there are 4 versions. As infrastructure, it supports VR (Virtual Reality) games like Unity. In accordance with Unity, it is the greatest engine that has succeeded in realistic visualization and has strong materials and animation features that can produce landscape and vegetation. It does not have as extensive documentation as Unity. Normally it's free, but if the fee earned exceeds 1,000,000 USD, a 5% royalty fee will be charged by Epic Games.

4.2 Common Approaches

Unity comes as a completely free game engine, but there are also paid and premium versions to choose from.

The games are for playing on various devices like iOS, Android, PC, and so on. Therefore, game development companies are required to develop various versions that users of any device can play without any glitches. The Unity 3D game engine allows game developers to make different versions for different platforms, while also being useful in quickly advertising the game title.

So with these common approaches, in this project, Unity will be our game engine.

4.3 What Other People Have Done?

4.3.1 Valheim

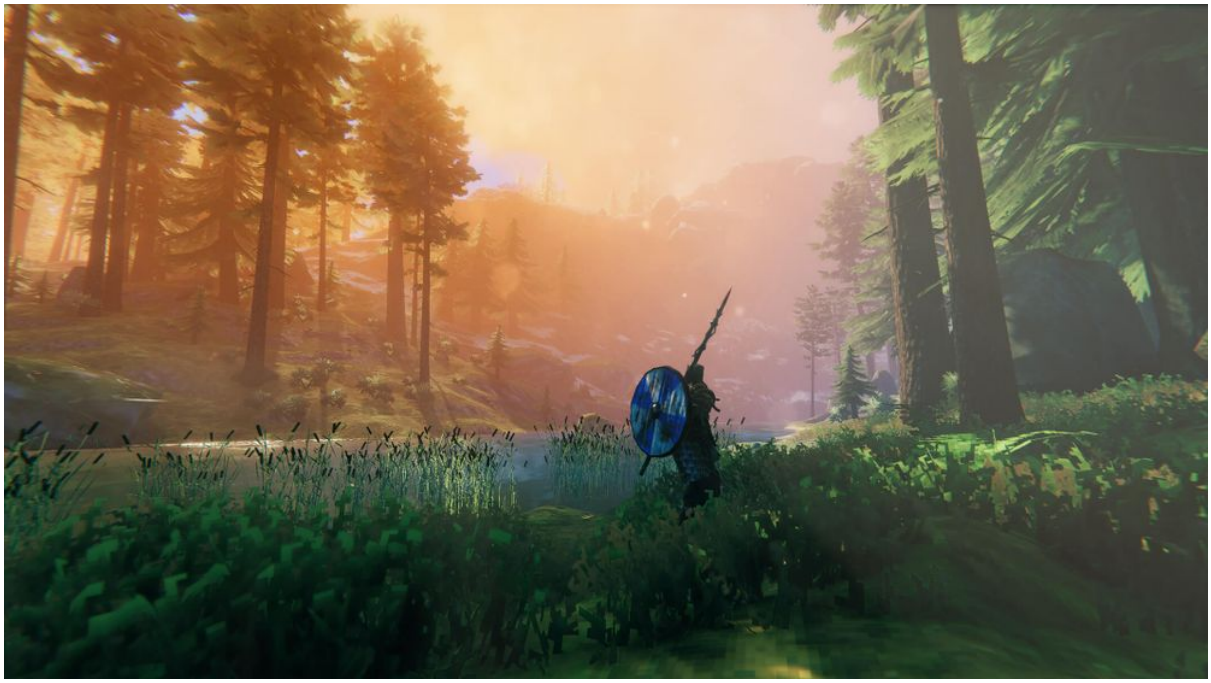

Figure 5: Valheim

Inspired by Viking culture, the survival game Valheim, which is still in early access, broke the record for the number of simultaneous players, despite being just announced, and became the second most played game in the world.

The Swedish developer Iron Gate Studio and publisher Coffee Stain Studios are teaming up to create the next survival and sandbox video game Valheim. For Linux and Windows, it was made available in early access, and it will be made available for Xbox One and Xbox Series X/S in 2023. A five-person team built on Richard Svensson's development work from a side project he worked on in his free time to create the game. It is made in Unity Game Engine.

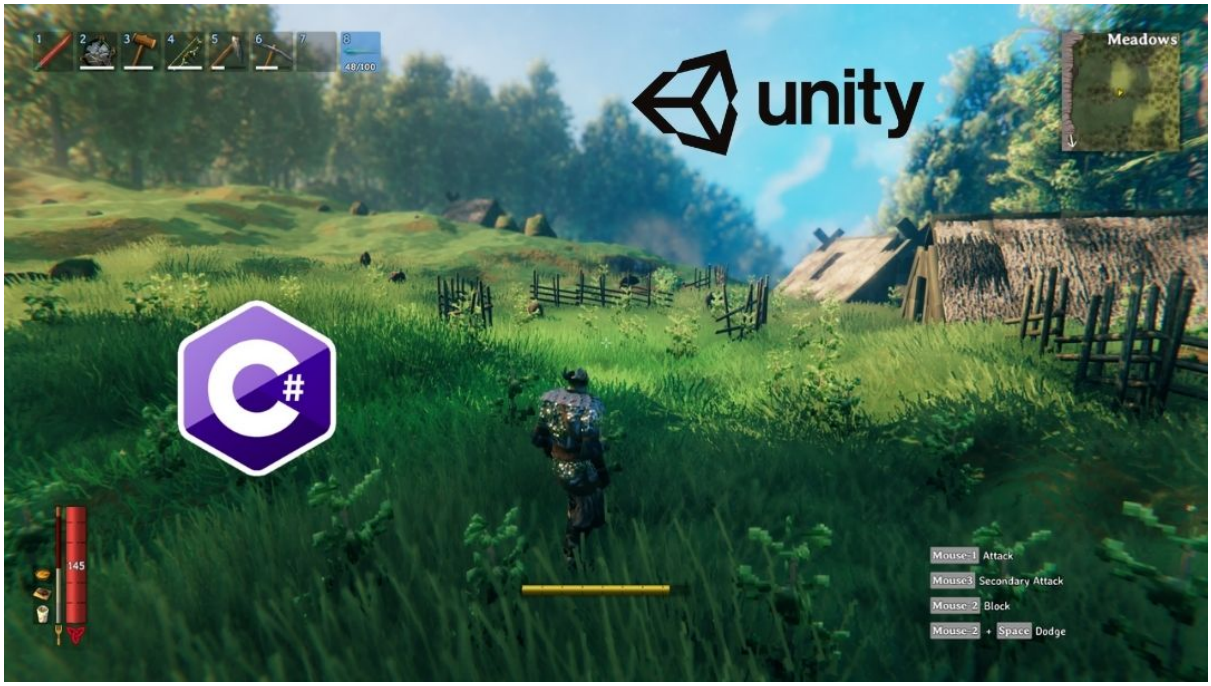

Figure 6: Valheim in Unity

[4]

4.3.2 Rust

Rust is a digitally distributed survival video game from Steam, developed by Facepunch Studios. It was released to Mac, Linux, and Microsoft Windows on December 11, 2013, and was in early access. It was back in early access on June 14, 2014. The game is made using the Unity 5 game engine.[5]

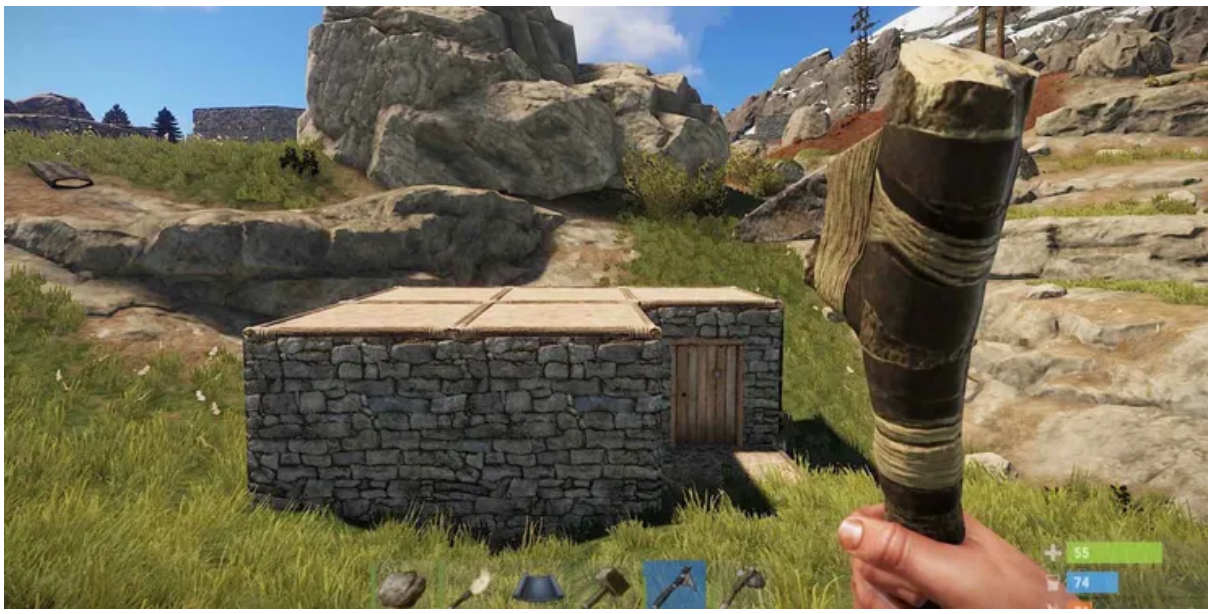

Figure 7: Rust

[6]

Figure 8: Rust
[7]

5 Method

Unity is a cross-platform game engine developed by Unity Technologies, which is primarily used to develop video games and simulations for computers, consoles, and mobile devices. It provides developers with a set of tools and frameworks to create 3D and 2D games, as well as simulations and other interactive content.

Some of the features of Unity include a visual editor, a physics engine, support for multiple platforms, networking support, and the ability to create custom tools and extensions. It also includes a large asset library of pre-made objects, materials, and other resources that can be used to build games and other interactive experiences.

Unity is used by a large community of developers and artists, and is a popular choice for creating games and interactive content for a wide range of platforms, including iOS, Android, PC, Mac, Xbox, and PlayStation.

Visual Editor: Unity provides a visual editor that allows developers to design levels, create animations, and script gameplay behavior using a drag-and-drop interface. The editor also includes a debugger and a profiler to help developers find and fix issues in their code.

Physics Engine: Unity includes a built-in physics engine that simulates realistic physics in games and other interactive content. This includes support for rigid body dynamics, collision detection, and soft body dynamics.

Multiplatform Support: Unity can build games and other interactive content for a wide range of platforms, including iOS, Android, PC, Mac, Xbox, and PlayStation. It also supports the creation of web-based content using HTML5 and WebGL.

Networking Support: Unity includes built-in support for networking, allowing developers to create multiplayer games and other interactive experiences that can be played online.

Asset Library: Unity includes a large asset library of pre-made objects, materials, and other resources that can be used to build games and other interactive experiences. These assets can be purchased from the Unity Asset Store, or created by developers using 3D modeling software.

Custom Tools and Extensions: Unity allows developers to create custom tools and extensions using the Unity Editor Scripting API. This can be used to automate common tasks, or to create entirely new functionality within the Unity editor.

Overall, Unity is a powerful and flexible game engine that is widely used by developers to create games and other interactive content for a variety of platforms.

5.1 Algorithms

5.1.1 A* Algorithm

The A* algorithm is a popular search algorithm that is used to find the shortest path between two points in a graph. It is commonly used in games and other applications that require pathfinding, such as GPS navigation systems or robot navigation.

In the context of Unity, the A* algorithm can be used to enable game objects such as characters or NPCs to navigate around obstacles in a game world. To use the A* algorithm in Unity, you will need to set up a navmesh (short for navigation mesh) that represents the walkable areas of the game world, and create an A* pathfinding agent that can move along the navmesh.

Here's a high-level overview of how the A* algorithm works: The algorithm starts at the starting point and adds it to a list of visited nodes called the "open set". The algorithm then examines the neighbors of the starting node and calculates a score for each neighbor based on its distance from the starting node and its estimated distance to the goal called the "heuristic". The algorithm selects the neighbor with the lowest score and adds it to the open set. The algorithm repeats this process for each node in the open set, until it reaches the goal node or determines that there is no path to the goal.

Scoring: As mentioned above, the A* algorithm assigns a score to each node based on its distance from the starting node and its estimated distance to the goal (called the "heuristic"). The total score for a node is the sum of its distance from the starting node and its estimated distance to the goal. The algorithm selects the node with the lowest score to expand next.

Heuristics: The heuristic is an estimate of the distance from a node to the goal. A good heuristic should never overestimate the distance to the goal, as this can lead to the algorithm finding suboptimal paths. One common heuristic is the Euclidean distance, which calculates the straight-line distance between two points. Other heuristics, such as the Manhattan distance (which calculates the distance between two points as the sum of the horizontal and vertical distances), can also be used.

Open and closed sets: As the algorithm expands nodes, it adds them to the open set (a list of nodes that have been visited but not yet fully expanded). When a node is fully expanded, it is removed from the open set and added to the closed set (a list of nodes that have been fully expanded). This helps the algorithm to avoid revisiting nodes and can improve its performance.

Path reconstruction: Once the algorithm has found the goal node, it can reconstruct the path by backtracking through the came from map (a dictionary that maps each node to its parent node). This allows the algorithm to trace the path from the start to the goal by following the chain of parent nodes.

There are many different ways to implement the A* algorithm in Unity, and there are also a number of third-party assets and plugins that can be used to add A* pathfinding to Unity projects.

To use the A* algorithm in Unity, need to set up a navmesh (short for navigation mesh) that represents the walkable areas of the game world, and create an A* pathfinding agent that can move along the navmesh.

Here are the basic steps for using the A* algorithm in Unity:

Set up a navmesh: The navmesh is a representation of the walkable areas of the game world. It can be created manually by placing NavMeshObstacle objects in the scene, or it can be generated automatically using the Unity Navigation system.

Create an A* pathfinding agent: The A* pathfinding agent is a script that can be attached to a game object (such as a character or NPC) to enable it to navigate around obstacles in the game world. The script should use the A* algorithm to calculate the shortest path between the agent's current position and the target destination, and then move the agent along the path.

Set up the A* algorithm: The A* algorithm will need to be implemented as a function in the pathfinding agent script. This function should take the start and goal positions as input, and return the shortest path between them as a list of nodes. The function should use the A* algorithm to calculate the path, using the navmesh to determine the walkable areas of the game world and the heuristic to estimate the distance to the goal.

Move the agent along the path: Once the A* algorithm has calculated the shortest path, the agent can be moved along the path using standard Unity movement techniques (such as transform.Translate or Rigidbody.MovePosition).

Step 1: Add the very first node to the OPEN list.

Step 2: Check if the OPEN list is empty or not; if it is, return failure and exit.

Step 3: If node n is the target node, return success and quit; otherwise, select the node from the OPEN list with the least value of the evaluation function (g+h).

Step 4: Expand node n' and create all of its successors, then place n in the closed list. For each successor n', determine whether n is already in the OPEN or CLOSED list; if not, compute the evaluation function for n' and enter it into the Open list.

Step 5: If node n is already in the OPEN or CLOSED state, it should be connected to the back pointer, which indicates the lowest g (n') value.

Step 6: Go back to Step 2.

Figure 9: Pseudocode for A* Algorithm

[2]

5.1.2 Sorting Algorithms for Unity Inventory System

Sorting algorithms are used to rearrange a list of items in a specific order such as ascending or descending order. There are many different sorting algorithms that can be used, and the best algorithm for a particular situation will depend on the specific requirements of the problem.

Here are a few sorting algorithms for a Unity inventory system:

- Bubble sort: Bubble sort is a simple sorting algorithm that compares adjacent elements in the list and swaps them if they are in the wrong order. It repeats this process until the list is sorted. Bubble sort is easy to understand and implement, but it is not very efficient for large lists.

- Insertion sort: Insertion sort is another simple sorting algorithm that builds the final sorted list one element at a time. It starts with the second element in the list and compares it to the previous element. If the previous element is larger, the two elements are swapped. The algorithm then moves on to the next element and repeats the process until the list is sorted. Insertion sort is slightly more efficient than bubble sort, but it is still not suitable for large lists.

- Merge sort: Merge sort is a divide-and-conquer sorting algorithm that works by dividing the list into smaller sublists, sorting the sublists, and then merging the sublists back together. Merge sort is more efficient than bubble sort and insertion sort for large lists, and it is a good choice for inventory systems that need to handle large amounts of data.

- Quick sort: Quick sort is another divide-and-conquer sorting algorithm that works by selecting a pivot element from the list and partitioning the list around the pivot. It is similar to merge sort in terms of performance, but it has a slightly different implementation.

Overall, the best sorting algorithm for a Unity inventory system will depend on the specific requirements of the system, such as the size of the inventory and the performance needs of the application.

The merge sort algorithm in a Unity inventory system:

```
using System.Collections.Generic;

public class Inventory
{
    private List<Item> items;

    // other inventory-related code here

    public void SortInventory(bool ascending)
    {
        items = MergeSort(items, ascending);
    }

    private List<Item> MergeSort(List<Item> list, bool ascending)
    {
        // base case: if the list has fewer than 2 elements, it is already sorted
        if (list.Count < 2)
        {
            return list;
        }

        // divide the list in half
        int mid = list.Count / 2;
        List<Item> left = list.GetRange(0, mid);
        List<Item> right = list.GetRange(mid, list.Count - mid);

        // sort the left and right halves recursively
        left = MergeSort(left, ascending);
        right = MergeSort(right, ascending);

        // merge the sorted halves back together
        return Merge(left, right, ascending);
    }
```

Figure 10: The Merge Sort Algorithm

```
private List<Item> Merge(List<Item> left, List<Item> right, bool ascending)
{
    // create a new list to hold the merged items
    List<Item> result = new List<Item>();

    // iterate through the left and right lists until one of them is empty
    while (left.Count > 0 && right.Count > 0)
    {
        // compare the first elements of each list
        if ((ascending && left[0].CompareTo(right[0]) < 0) ||
        (!ascending && left[0].CompareTo(right[0]) > 0))
        {
            // if the left element is smaller (or larger, if we are
            //sorting in descending order), add it to the result list
            //and remove it from the left list
            result.Add(left[0]);
            left.RemoveAt(0);
        }
        else
        {
            // if the right element is smaller (or larger,
            //if we are sorting in descending order),
            //add it to the result list and remove it from the right list
            result.Add(right[0]);
            right.RemoveAt(0);
        }
    }

    // add any remaining items from the left list
    while (left.Count > 0)
    {
        result.Add(left[0]);
        left.RemoveAt(0);
    }

    // add any remaining items from the right list
    while (right.Count > 0)
    {
        result.Add(right[0]);
        right.RemoveAt(0);
    }

    // return the merged list
    return result;
```

Figure 11: The Merge Sort Algorithm

This implementation of the merge sort algorithm takes a list of items and a boolean value indicating whether the list should be sorted in ascending or descending order. It uses the divide-and-conquer approach to recursively sort the list, and then merges the sorted sublists back together to create the final sorted list.

5.2 Designs

Game design is the process of creating the concept, mechanics, and rules of a game. It involves deciding on the theme, characters, storyline, and gameplay, as well as the goals, challenges, and rewards for the player.

There are many different approaches to game design, and the design process can vary depending on the type and scale of the game. Some game designers work on a small, independent team to create a simple game, while others work on larger projects with a more complex design process.

The design of a game typically starts with a concept or idea, which is then developed into a game design document (GDD). The GDD outlines the overall vision for the game, including the theme, characters, gameplay mechanics, and level design.

From there, game designers might create prototypes or mock-ups to test and refine the gameplay mechanics. They may also work with artists, programmers, and other game developers to create the final product.

Game design also involves playtesting and iteration to ensure that the game is fun, challenging, and balanced. This can involve gathering feedback from players and making changes to the game based on that feedback.

Overall, game design is a creative and collaborative process that involves creating engaging gameplay experiences for players.

Here is an example of a survival game design:

* Title: "Survive the Wilderness"

* Objective: The player must survive in a wilderness setting by finding shelter, gathering resources, and avoiding dangers.

- Gameplay: The player starts the game stranded in a wilderness with no supplies or shelter. They must explore their surroundings and gather resources, such as food, water, and materials to build a shelter. The player must also defend themselves against dangers in the wilderness, such as wild animals, natural disasters, and other hazards.

As the player progresses through the game, they must manage their resources carefully and make strategic decisions to stay alive. They can improve their shelter and acquire new skills and tools to better survive in the wilderness.

This is just one example of a survival game, but there are many different ways to design a game with a survival theme. The specific mechanics and challenges can vary depending on the desired gameplay experience and the resources available to the player.

6 Dataset

A data set may not be necessary for a survival game if the game has a simple design and mechanics that do not require a large amount of data to function. For example, a survival game that has a fixed set of resources and challenges, and does not involve procedurally generated environments or realistic AI behaviors, may not need a data set.

However, if the survival game has more complex mechanics or features that involve a large amount of data, such as procedurally generated environments or realistic AI behaviors, then a data set may be useful to support those features.

In the example above, the survival game "Survive the Wilderness" had a simple design with a fixed wilderness setting and a limited set of resources and challenges. In this case, a data set might not be necessary, as all of the necessary information could be included in the game itself.

As a result, our game does not need a dataset. Because we are independent game developers and our resources are limited.

7 Platform for Implementation

Unity is declared the platform for implementation because it is a popular choice for game development because it is cross-platform, easy to use, has a large asset library, and has a strong community of developers. It also has a number of powerful features that make it a flexible choice for creating a wide range of games and interactive content.

Unity has a large and active community of developers and artists who share tips, advice, and resources with each other. This can be a great resource for developers who are new to Unity, or who are looking for help with specific problems.

8 REFERENCES

References

[1] *Document From gameSparks*. 2022. URL: https://www.gamesparks.com/blog/unity-game-engine-review/.

[2] Mohammad Aakil Iqbal. *A* algorithm*. 2022. URL: https://www.ijraset.com/research-paper/design-and-implementation-of-pathfinding-algorithms-in-unity-3d.

[3] Victoria Mozolevska. *Indie Game Design*. URL: https://kevurugames.com/blog/indie-game-development-the-all-you-need-guide-to-revenues-most-profitable-genres-monetization-bonus-top-10-best-indie-games-2020/#:~:text=The%20word%20indie%20has%20nothing,financial%20support%20of%20large%20publishers..

[4] postpirates. *Valheim Unity*. 2022. URL: https://www.postpirates.com/2020/07/what-are-best-free-platforms-for-making.html.

[5] Rust. *Rust*. 2022. URL: https://tr.wikipedia.org/wiki/Rust_(video_oyunu).

[6] Rust. *rustImg*. 2022. URL: https://www.thegamer.com/unity-game-engine-great-games/.

[7] Rust. *rustImg2*. 2022. URL: https://blog.unity.com/community/porting-to-unity-5-the-untold-rust-journey.