Nathan V (100886412) Programmer

Rish V (100827691) Programmer

---------------------------------------------------------------------

Version 2.0

Nathan’s Roles & Responsibilities:

Programmer & was put in charge of the Performance Profiling & Optimization Design Patterns. (Implemented, Scriptable Objects and Inventory System)

Rish’s Roles & Responsibilities:

Programmer & In charge of the DLL/Plugin and Game Design Improvements (Implemented/Fixed, Health Bar, Add FMOD to project for simple sound management)

Percentage of Contributions:

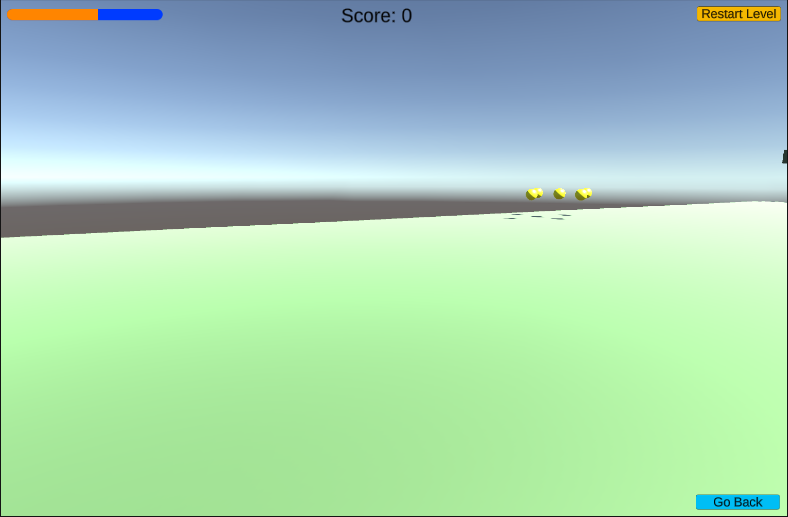
Nathan V. = 50%

Rish V. = 50%

Interactive media scenario information:

The media scenario that we have developed for this assignment is to show off the possibilities of movement-based ideas within Unity. So, we chosen to make a parkour movement-based platformer using each of the deliverables we are being asked to implement. Additionally in Version 2.0, a health bar and inventory help increase the scope of the project with starting points up adding ideals about enemies and upgrades or items to help increase movement speed or jump height.

Previous Game Design Pattern Improvements/Implementation from Part 1

One of the improvements made to the original deliverables was fixing/adding functionality to the health bar such as taking damage and plus recovering health (Apart of the optimization design patterns i.e. Inventory). The Reason why we fixed it was, in the future in any of us wanted to build off this project and wanted to add obstacles or enemies it would be easy to implement the damage to the player.

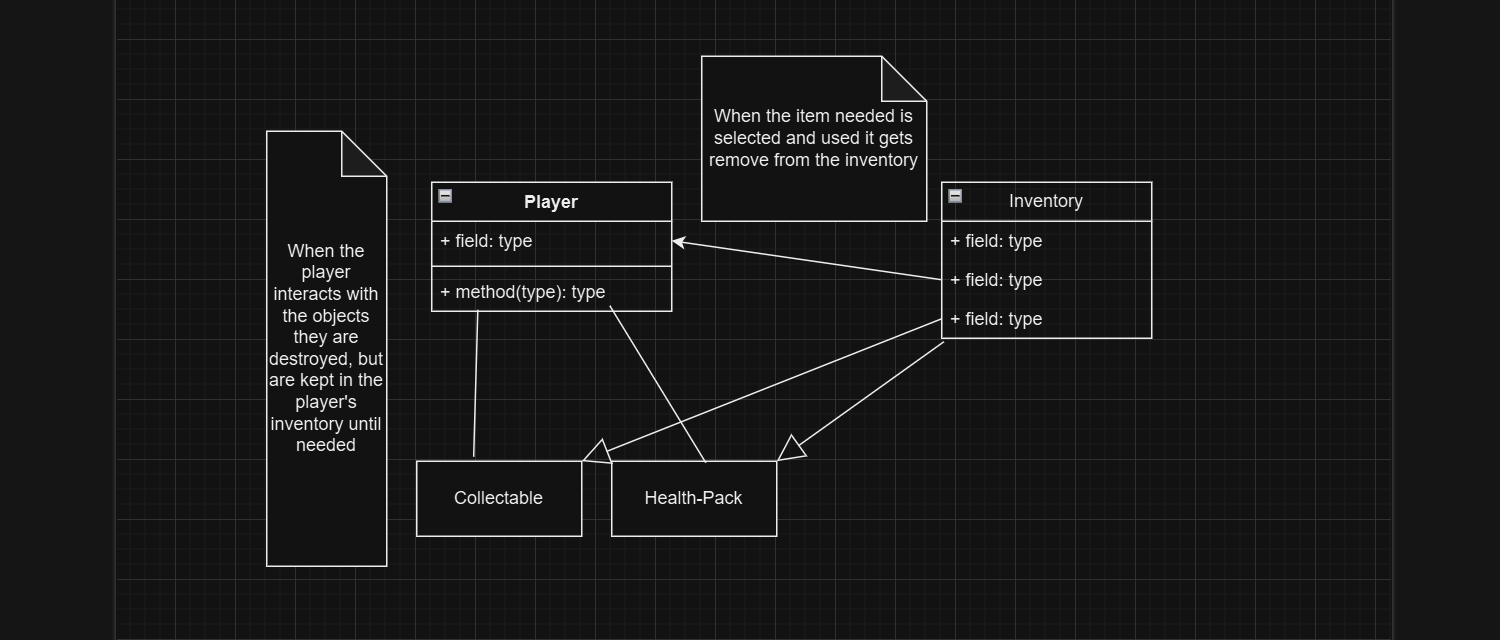
Optimization Design Patterns:

Scriptable Object

A screenshot of a computer screen

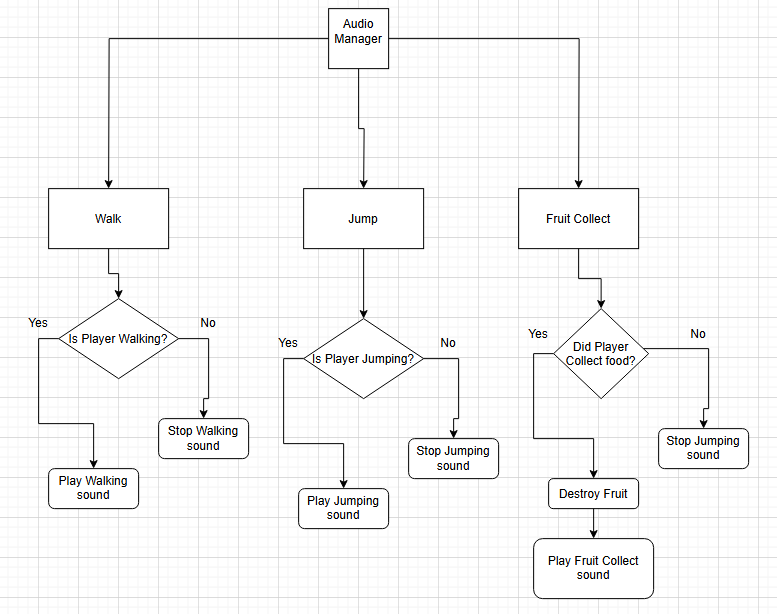
Description automatically generatedThis was best implementation for this project because, there’s so many different values such as ground drag, jump height, walking speed, etc. That it clogs up the scripts. So, by having the Scriptable objects it helps group them up for easier management and additionally when it comes to implementing new variables or options. Overall, it just helps with memory and not having messing code with too many unneeded lines of code.

A screen shot of a computer

Description automatically generatedObject Pooling/Inventory

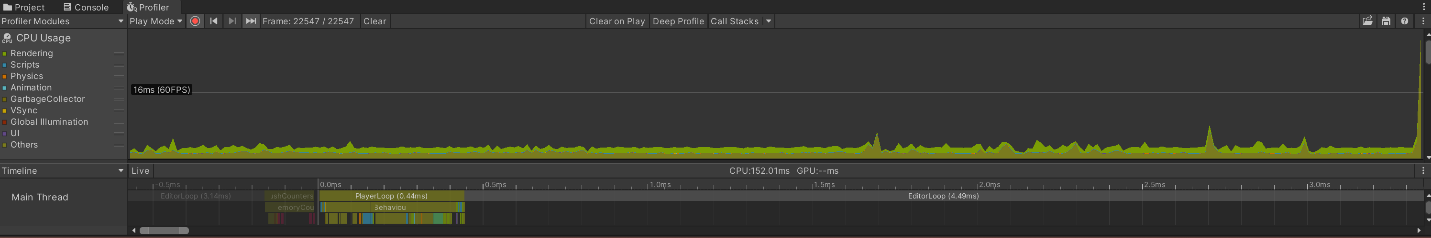
Object Pooling was used to make a sudo-inventory system. The way it works in the game is the player can/has items that the player can use such as a health pack, which heals the player and the normal pick up which the console lists out how many the player has pick up. It’s a very basic implementation that can be change to sort certain needs or update it for more items.

DDL/Plugin:



I feel like FMOD is a crucial plug-in for Unity Game Developers to know. This Plug in stores all of your audio files neatly and can be used to modify sounds as well. FMOD also makes the accessibility more easier for developers to add their sounds in the necessary areas of their game.

For this project we depend on sound effects to let the players know about their actions accordingly. It's important to organize the sound effects appropriately to ensure the project files are ordered neatly

Performance Profiling

The Profiler we used was the basic one built into Unity, and we used Static batching on certain objects such as the collectibles and other such items as to have them not take to much memory to render or load in.

---------------------------------------------------------------------

Version 1.0

Nathan’s Roles/Responsibilities:

Programmer & worked on the singleton and observer parts of the assignment (Implemented the Sound manager, UI manager, collectables and health bar)

Rish’s Roles/Responsibilities:

Programmer & worked on command design pattern and factory pattern of the assignment (Implemented the restart button and collectable drop in function)

Percentage of Contribution:

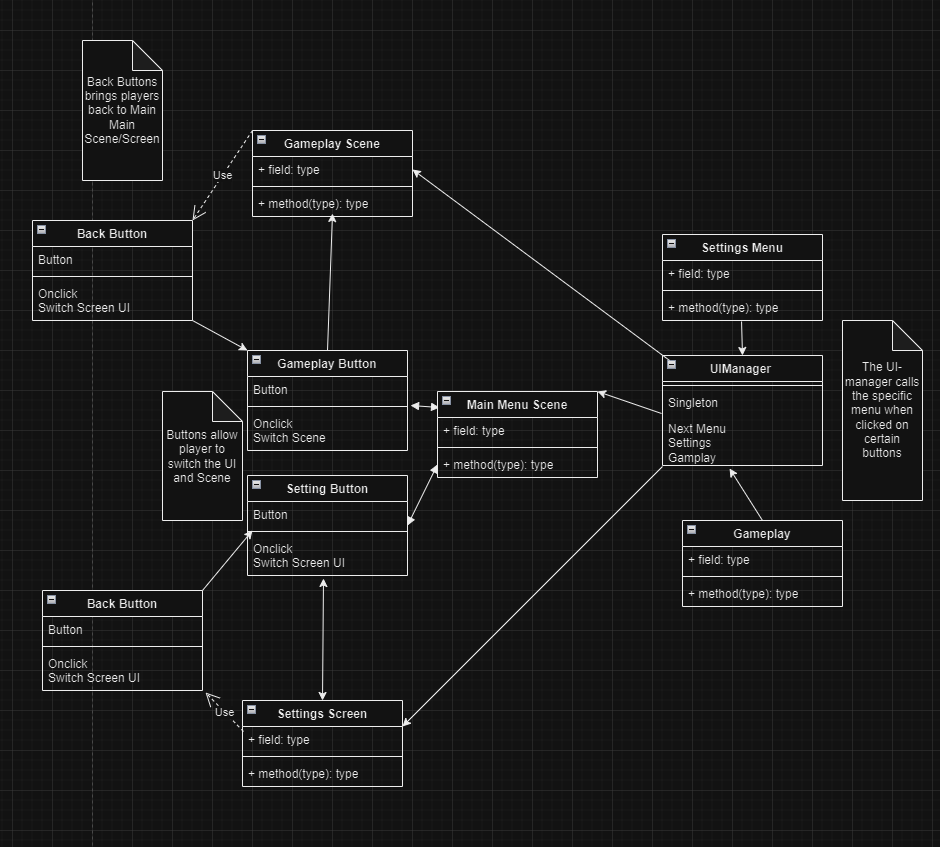
Nathan V. = 50%

Rish V. = 50%

Interactive media scenario information:

The media scenario that we have developed for this assignment is to show off the possibilities of movement-based ideas within Unity. So, we chosen to make a parkour movement-based platformer using each of the deliverables we are being asked to implement. But from the player’s perspective they are asked to learn about the various ways of movement that can be created within a game engine such as Unity. Some of ways the player can learn about the movement from the deliverables is the collectables to see how UI interacts with all the other components. Also, the Sound manager to be able to list to the difference of certain actions such as jumping or walking.

Singleton Implementation/Explanation

UI Manager

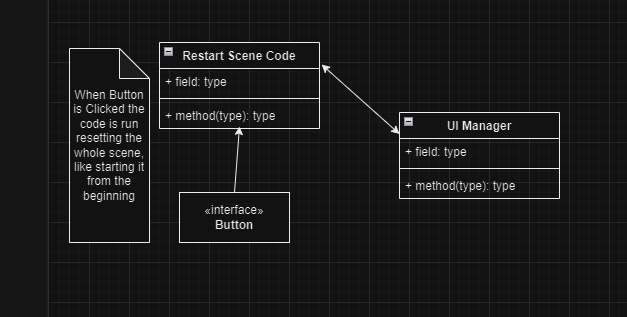
The way the UI manager was implemented was to have a canvas object where are the menu objects are and have then disabled so that when the player starts the game it starts on the main menu. Though the code for the Main menu takes the canvas menus and holds in a list so that when certain buttons are pressed the scripts runs and outputs the correct menu. This helps the experience by making it so the legend on the side bar in unity isn’t clogged or overwhelming as well simplifies the process for the code to figure out what menu needs to be loading and to remember if the player wants to go back to the previous screen.

A black screen with white text

Description automatically generatedSound Manager

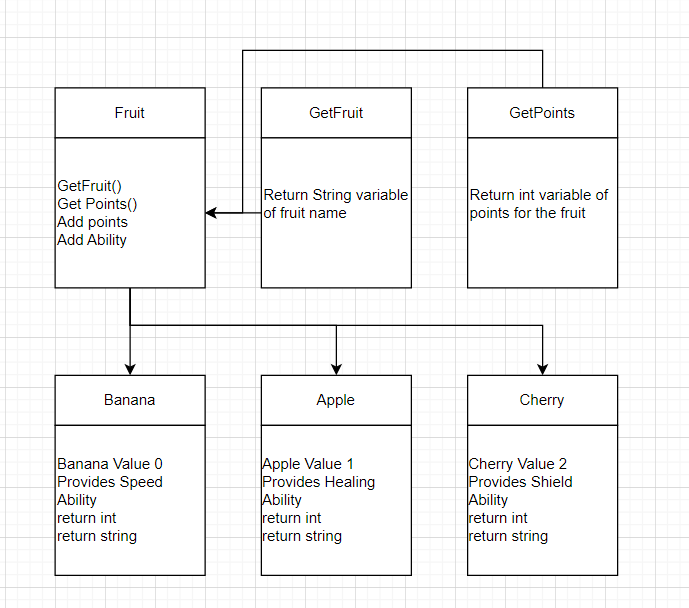
The way the sound manager was implemented has the script with Enum-list that has the specific sounds listed in order, where you need to place them as such when attaching them to the sound manger empty game object. So, the way it works is the specific sound is played when the player performs a certain action or presses a certain key (i.e. space is space it plays a jump sound effect) This helps the player during the experience by giving a sound effect so that juices up the experience and lets the player know what action was performed.

Command Design Pattern Implementation/Explanation

Restart Button

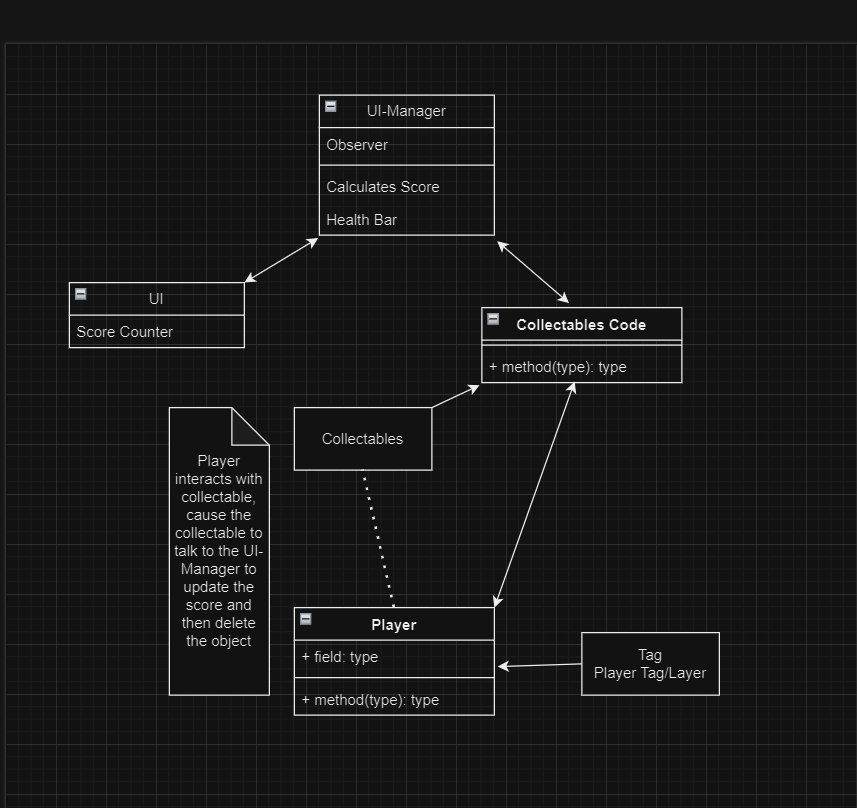
The way the button was implemented was again using the UI manager first and placing a button for the user to press to reset the scene from the beginning. But moreover, a simple script is attached to empty game object so that when the player clicks the button the script is read and then in turn restarts the scene. The whole experience benefits from this implementation because say if the player were to get stuck or frustrated from a section this button would allow the player to restart and try again from the beginning.

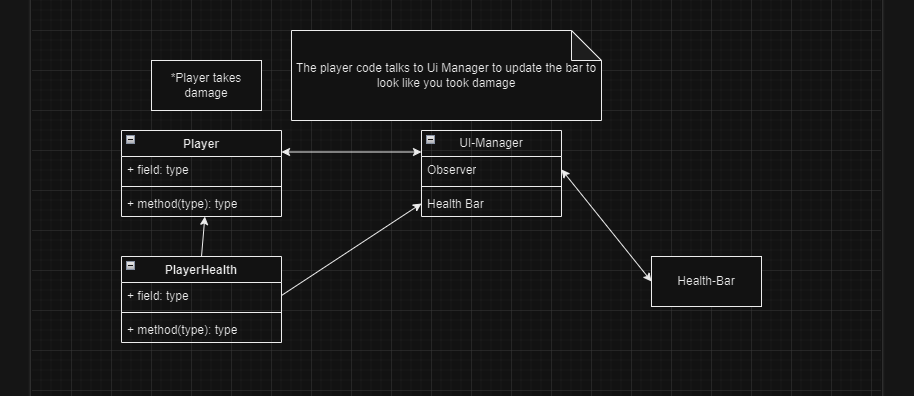
Factory Pattern Implementation/Explanation

The Factory Command pattern is used with the fruits system. Each Fruit provides a specific boost and points for the game, depending on which fruit is taken by the player. For example, If the player selects a banana, they get a 3 point bonus and more speed.

Observer Implementation/Explanation

Collectables

The way the collectables were implemented was by having a canvas with new text part that when the game ran would load with the score text and collectables in the game have a script attached to the game object that when collected by the player would add to the score UI text. The reason was added and how it benefits the whole experience a reason for players to jump around to find things to collect to make the players explore around the map and test their video game movement skills.

Health Bar

The health bar was implemented by again UI canvas object to show the health bar model and scripts attached to both the player and Manager. The player script lets us choose the amount of health the player has, while the manager script takes that number and then presents in the health bar. This helps the experience, by showing players they need to be carful when exploring as they could be enemies or obstacles that could damage the player and possibly kill them.

References and Help:

“FMOD.” *Fmod.Com*, www.fmod.com/. Accessed 26 Nov. 2024.

“FIRST PERSON MOVEMENT in 10 MINUTES - Unity Tutorial.” *YouTube*, YouTube, [www.youtube.com/watch?v=f473C43s8nE](http://www.youtube.com/watch?v=f473C43s8nE). Accessed 24 Oct. 2024.

“PLEASE Use a Unity SOUND MANAGER! - Full Tutorial.” *YouTube*, YouTube, [www.youtube.com/watch?v=g5WT91Sn3hg](http://www.youtube.com/watch?v=g5WT91Sn3hg). Accessed 24 Oct. 2024.

“How to Add Footsteps Sounds in Unity.” *YouTube*, YouTube, [www.youtube.com/watch?v=uNYF1gsvD1A](http://www.youtube.com/watch?v=uNYF1gsvD1A). Accessed 24 Oct. 2024.

“How to Make a UI Management System in Unity.” *YouTube*, YouTube, [www.youtube.com/watch?app=desktop&v=rdXC2om16lo](http://www.youtube.com/watch?app=desktop&v=rdXC2om16lo). Accessed 24 Oct. 2024.

“How to CHANGE SCENE in Unity When Pressing a Button.” *YouTube*, YouTube, [www.youtube.com/watch?v=dZ8I73erDSk](http://www.youtube.com/watch?v=dZ8I73erDSk). Accessed 24 Oct. 2024.

“Unity - How to Reset / Reload Scene with C# (Restart Game Button Click, 2021) | Easy Tutorial.” *YouTube*, YouTube, [www.youtube.com/watch?v=TVSLCZWYL\_E](http://www.youtube.com/watch?v=TVSLCZWYL_E). Accessed 24 Oct. 2024.

“Unity 2019.3 - Unlocking the Mouse Cursor.” *YouTube*, YouTube, [www.youtube.com/watch?v=StA6vtCZyI0](http://www.youtube.com/watch?v=StA6vtCZyI0). Accessed 24 Oct. 2024.

“Factory Method in C#.” *Refactoring.Guru*, refactoring.guru/design-patterns/factory-method/csharp/example. Accessed 24 Oct. 2024.

Sound References:

Sound Effect from <a href="https://pixabay.com/sound-effects/?utm\_source=link-attribution&utm\_medium=referral&utm\_campaign=music&utm\_content=38778">Pixabay</a>

Sound Effect from <a href="https://pixabay.com/?utm\_source=link-attribution&utm\_medium=referral&utm\_campaign=music&utm\_content=30946">Pixabay</a>

Sound Effect from <a href="https://pixabay.com/sound-effects/?utm\_source=link-attribution&utm\_medium=referral&utm\_campaign=music&utm\_content=6136">Pixabay</a>