**Software Architecture Report**

Project: Elysian Solitude

Date: 01.2024

Version: commit 67e3cbf

1. What the purpose of the software project is
2. This may be a summary based on the planning documentation.

A fully working Massively Multiplayer Online Role-Playing Game (MMORPG) in Unity (Metin 2 style) . The game will have a few maps based on different themes where players can meet and level up by defeating different monsters and bosses. Every player will have their predefined abilities and can scale up their power by going on different adventures across the world.

1. Fulfilled capabilities – what can the project do at this point, and what is to be done until project fulfillment.   
   The whole idea of the game is travelling along all the maps, get all the items and finish your job to save the Empire. Along your way, you can do many different actions:  
   You can explore the design of different maps  
   You can battle with different mobs, correlated to the map theme  
   You can defeat bosses in the maps and gain different items  
   After killing a mob, you can use the items it drops to restore the health  
   You can use the items dropped by the bosses to teleport to other maps

1. Guides on how to:
2. Run the project locally.

To run the project locally you should run the .exe file (the builded version of the game), or open it in Unity app and run from a scene.

1. Build the project.

The project is already builded. Otherwise, you should download and install Unity and UnityHub, open the project and then click on run. To build it you should press the build button.

1. Deploy the project (either locally, or how is it hosted, where it is the case)

To deploy the project you should enter the site itch.io (or other similar site) and upload the project.

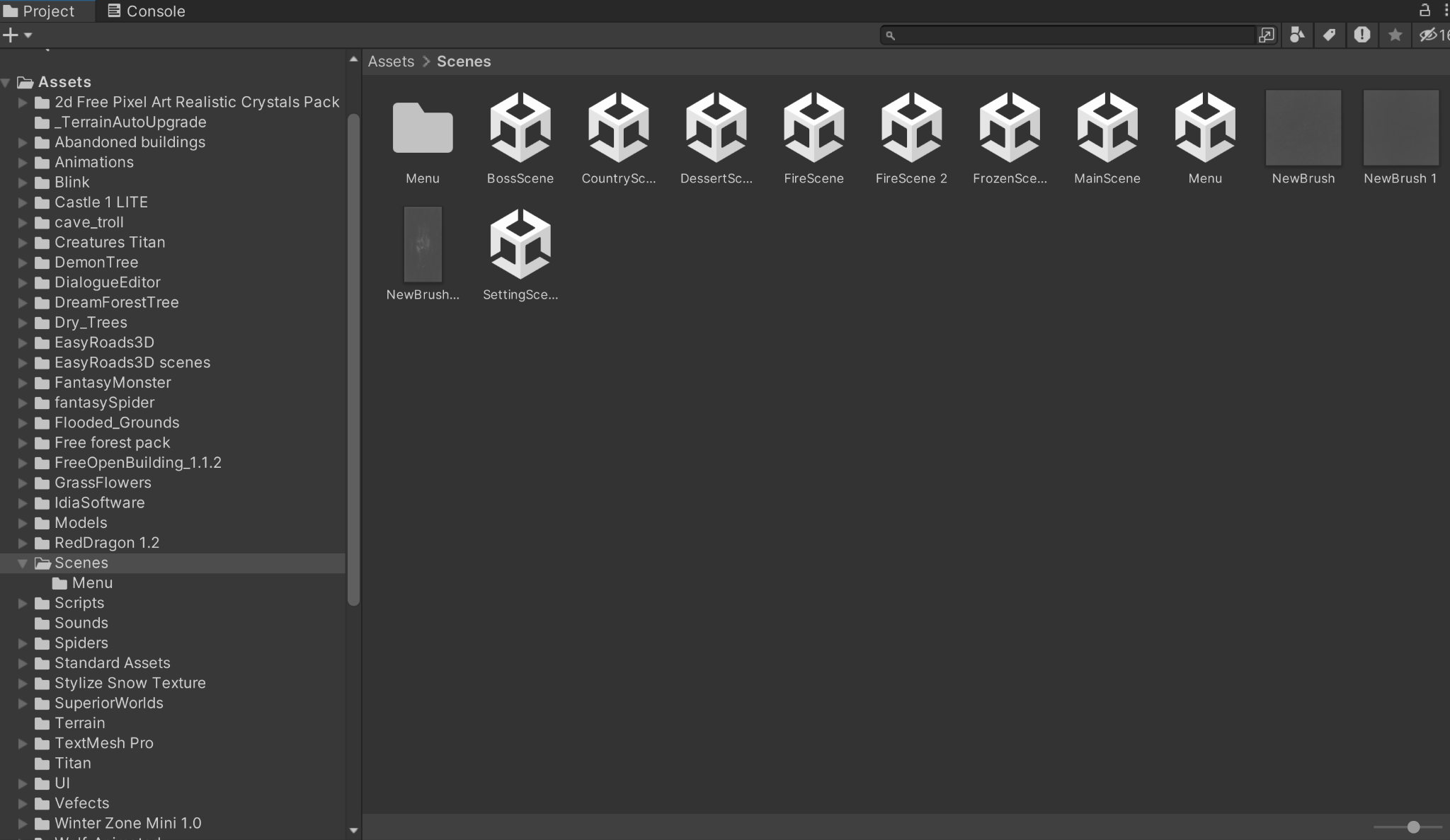
d. Contribution guide

1. Patterns used in your application

We used a few patterns like Singletone - MainMenu.

1. Application entry points
2. Data sources

The code and the packages we used is organized in the Assets folder of the project. There are all the documents of the application.



1. Data inputs

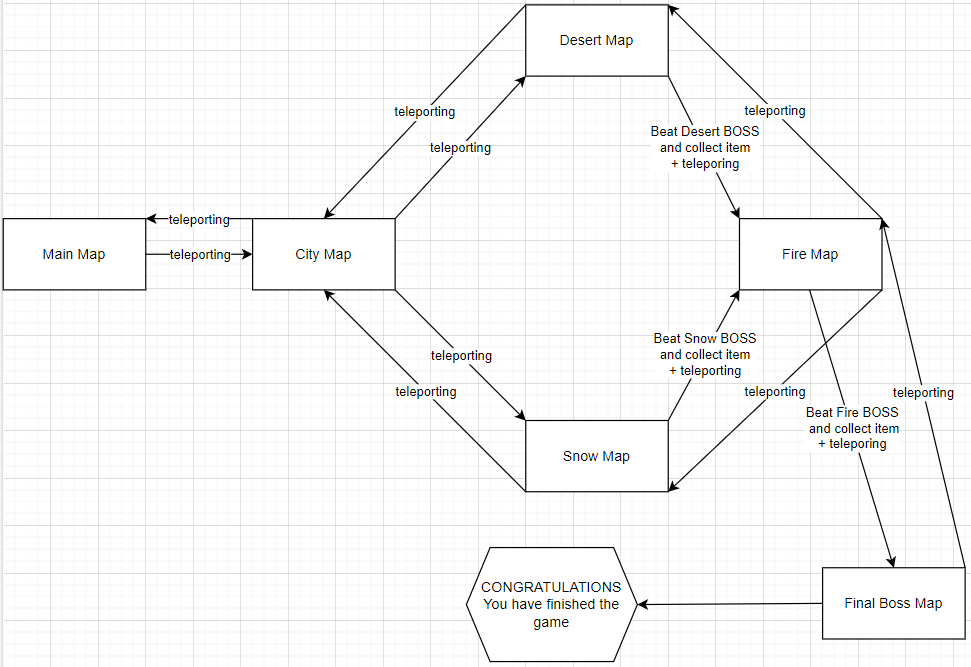
The inputs are mouse and keyboard events, witch controls the player and the action. With W,A,S,D you can walk on the map forward, to the left or right, or backward. With Q you can beat enemies, and the same you can do by using the 1,2,3 keys. The mouse is used to control the main menu.

1. Configuration files

The project contains basic configuration files for a Unity project.

1. High level diagrams of the architecture
2. User/data journeys

The users follow the next road map in their progress along the game. There are five maps where our hero should beat all the enemies to receive items in inventory. These items are valuable because it helps you travel between the maps and after the battle with the final boss, you receive a special one, that announce you about finishing all your long road and beating all the monsters. That comes with a special *Congratulations* message.



1. Most valuable output

The most valuable output of the game is the pleasure of playing it. You receive items by playing to keep it interesting, and each item is useful at one step.

1. Deployment plan
2. Where is the application deployed.

The application is deployed on itch.io and it is available on this link: <https://de3v.itch.io/eli>

b. How the CI/CD pipeline works.   
 We have 2 workflows running: Build workflow and Testing workflow.

The Build workflow is configured through the main.yml file and is triggered at every push and pull request. It is configured to build the game and to export it into a Win64x playable version. It is made up of 4 steps :

→ Checkout action is used to clone the repository into the runner's workspace

→ Caching action is used to cache the 'Library' directory and can speed up the build process by up to 50%

→ Unity Builder action is employed to build the Unity project using Unity credentials stored as secrets

→ Upload Artifact action is used to upload the build artifacts to github and provides a stable playable version of the game

The test workflow is configured through the test.yml file and is triggered at every push and pull request. It is configured to provide test results in play-mode, edit-mode and standalone-mode. It is made up of 4 steps:

→ The Checkout, Caching are very similar to the Build workflow steps.

→ Unity Test Runner action which is made to run tests in the 3 different modes.

→ Upload Artifact action is used to upload test results.

1. Description of the QA process
2. Test suites – what do they test

→ We basically did Manual Unit Testing where each of us tested key parts of their map including character movement, items dropped, mob interactions etc. If we look through many of the scripts we will surely find a lot of commented Debug.Log() lines of code.  
  
→ We also created the CI/CD pipeline for testing where we provide test results for every different mode in Unity.

1. External dependencies included in the project
2. APIs used

Our project does not integrate with external APIs.

b. Libraries

The game uses only Libraries offered by Unity (UnityEngine.UI , UnityEngine.Events etc.). There are some assets from the game that are imported from AssetStore.

1. How vulnerable is the project to dependency attacks (ex. [Dependency Confusion](https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610))

Dependency attacks, like Dependency Confusion, are a concern for any modern software project. We've tried to reduce this risk and we ensured that all the libraries and assets are sourced from official and verified repositories, and that means that our project contains only secure and autentic versions..