Software Design Description
for
Non-Euclidean Game Engine

Prepared By: Ömer Buğra İNCE

Barış MERT

Ozan BAYRAKTAR

Advisor: Dr. Faris Serdar TAŞEL

Table of Contents

1. Introduction

Software Design Description, is a document to detail implementation and design of the project in question and detailing it with visuals.

1.1 Purpose

Purpose of this document is to present the design and implementation of the Non-Euclidean Game Engine. Non-Euclidean game engine is a tool to help developers develop Non-Euclidean games.

1.2 Scope

This document details this project's design. It will have two major features. One of them is the customizable portal which can be used by user to perform tasks that will make the player feel like they are in a Non-Euclidean environment and second one is a Non-Euclidean Area which will warp objects and other props to make it look like a Non-Euclidean space.

1.3 Definitions, Acronyms, Abbreviations

Non-Euclidean: Elements that deny Euclidean Geometry principles
Portal: A gate for changing position in space-time plane.
Unity: A modern game engine with its own editor.
GameObject: Main object class for game objects in Unity.
Camera: Camera class for game camera in Unity.

2. Architecture Approach

2.1 Design Approach

We will use the spiral development approach because we will work in an area, we are not very accustomed to and by using this methodology we can add in new features and take out or change features that can't be implemented in the way we planned beforehand.

2.2 Software to Use

This project will be an add-on to Unity Game Engine and will work hand to hand with it. Code part will be written in C++ and C#. To write our code in, we will use different text editors and IDEs that will not be a part of our final product.

2.3 Class Diagram

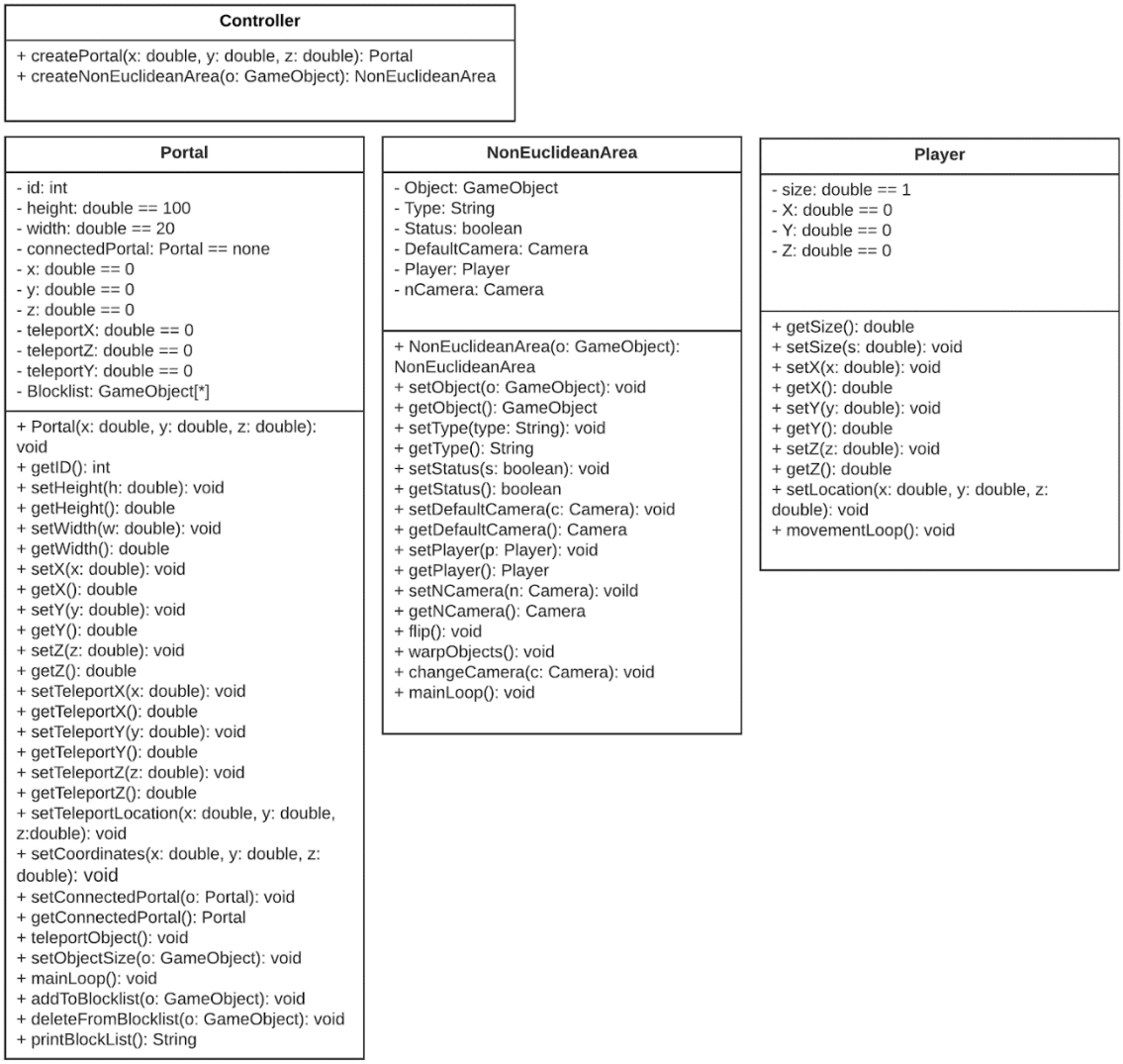

Figure-1: Class Diagram

2.3.1 Class Diagram Description

2.3.1.1 Controller

Controller will be interacted by the addon interface to create objects.
Functions:

- **createPortal:** createPortal function will be used by the addon interface to create a portal by calling the overloaded constructor of Portal class, which will require x, y, z coordinates.
- **createNonEuclideanArea:** This function will be used by the addon interface to create a Non-Euclidean Area by calling overloaded constructor of the NonEuclideanArea class, which requires a GameObject to put NonEuclideanArea attribute on.

2.3.1.2 Portal

Portal will have another portal that it is connected to or coordinates of a point to teleport a player it collapsed with. If both attributes are filled, the connected portal will have the priority.
Functions:

- **Set/Get Methods:** Set/Get methods are default set/get methods for private variables which will be used by Unity's "Inspector" interface to change attributes of the object.
- **setTeleportLocation:** This function is for changing the teleport location of the portal and for ease of use to not set each coordinate one by one.
- **setCoordinates:** This function is for changing coordinates of the portal by inputting all values in once and to not set each of them one by one.
- **teleportObject:** This function is called when an object collapses with the portal and will check if the Portal in question has a Connected Portal or a Teleport Location saved as a variable. If it has both of them Connected Portal will be prioritized.
- **setObjectSize:** Multiply object's size attribute depending on the difference between connected portals.
- **mainLoop:** Loop to check if Portal collapses with a GameObject that is not in the Blocklist and call appropriate functions.
- **addToBlocklist:** Blocklist is for objects that Game Developer doesn't want to teleport and with this function Game Developer can add objects to Blocklist.
- **deleteFromBlocklist:** For deleting objects from blocklist.
- **printBlocklist:** Prints an organized list of items in the Blocklist.

2.3.1.3 Non-Euclidean Area

Non-Euclidean Area will be an attribute that can be mounted on to GameObjects, if the GameObject has this attribute. When a player is collapsing with this area by being in it, depending on the type of the area, the player camera will be changed by another camera that has different attributes as to perception or game objects will be warped (also depends on the type of the area).

Functions:

- **Set/Get Methods:** Set/Get methods are default set/get methods for private variables which will be used by Unity's "Inspector" interface to change attributes of the object.
- **warpObjects:** Warps objects in the Non-Euclidean Area depending on the Type attribute.
- **changeCamera:** Changes DefaultCamera with inputted Camera.
- **mainLoop:** Loop to check if an object collapses with the Non-Euclidean area and calls appropriate functions depending on the situation.

2.3.1.4 Player

Player object will be a controllable object that can be moved by the player.

Functions:

- **Set/Get Methods:** Set/Get methods are default set/get methods for private variables which will be used by Unity's "Inspector" interface to change attributes of the object.
- **setLocation:** Sets user location for ease of use to not input each coordinate one by one.
- **movementLoop:** Moves users depending on the key pressed while in the game playing environment.

2.4 Unity Functionality

Unity supplies us with a lot of functionality that will be used in games Game Developers want to develop. And provide means to implement Basic Game Functionality mentioned in Software Specification Requirements. Unity does this by providing pre-made collision detectors implemented into the editor and/or pre-made scripts that can be found on the Unity Store. Since these can be provided by other means and our project's main focus is to help Game Developers with Non-Euclidean functionality, we will not implement these into our addon.

3. Flowchart

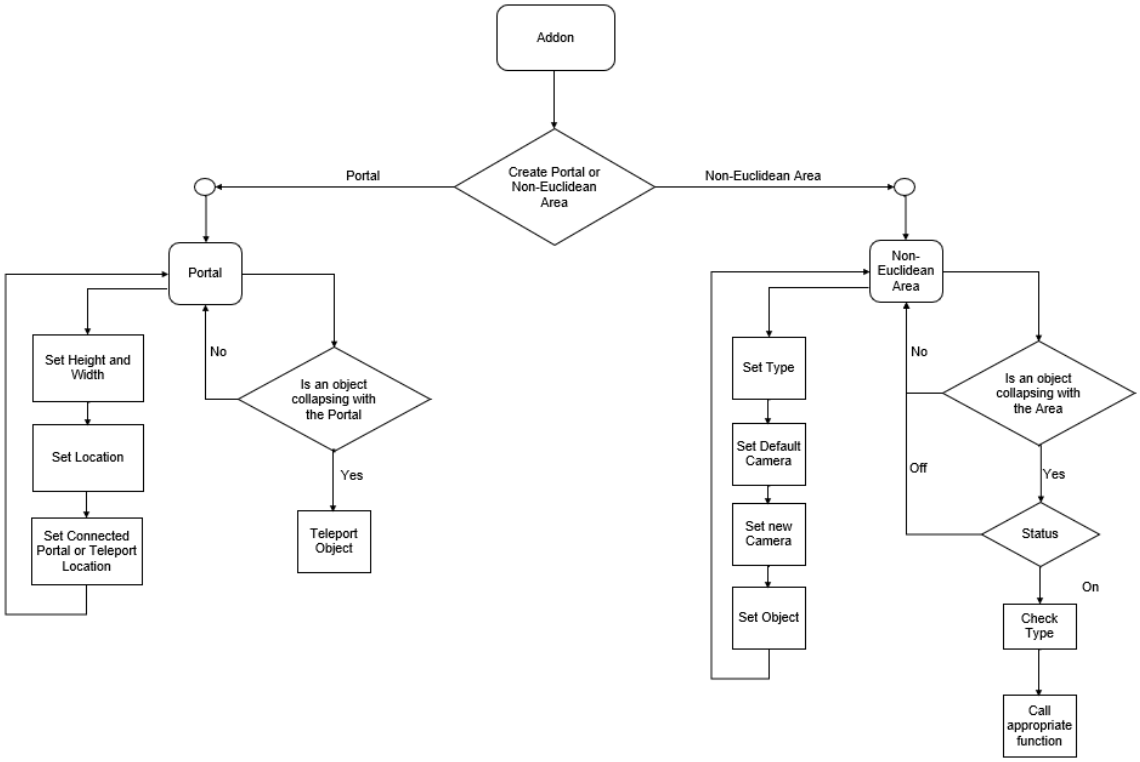

Figure-2: Flowchart

Figure-2 shows add-on's flowchart of operations and how it should work.

4. Add-on Interface

Non-Euclidean Game Engine addon will have its own interface docked into the Unity application and will have 2 buttons that will be used to create a portal and create a Non-Euclidean Area. Clicking either one will spawn its respective object in the game world. After that, created objects will be controllable and modifiable by Unity's own user interface.