ÇANKAYA UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

Project Report
Version 1

CENG 408
Innovative System Design and Development II

Non-Euclidean Game Engine

Ömer Buğra İnce
201611030
Barış Mert
201611041
Ozan Bayraktar
201711004

Advisor: *Dr. Faris Serdar Taşel*

Table of Contents

Abstract

Nowadays, most game engines and for this reason most of the games use Euclidean geometry as a base for their development. Euclidean games as we describe them do not bend space and time. As these games continue to get developed, non-Euclidean games also started to get popular because things you can do with non-Euclidean spaces are more mind blowing and confusing for us to perceive, therefore this creates a great field to discover both for developers and players. Our objective is to create a non-Euclidean game engine that will contain non-Euclidean features to help game developers create their own games using these features as they see fit. In this report we will detail the geometry and methods we will use.

Keywords: Non-Euclidean geometry, Game engine, Euclidean geometry

1. Introduction

Game engines are created and used to develop new games or remake existing ones; they are IDEs for games. They make complex game development processes simpler by providing tools to make normally big tasks done easily, while the game engine does all the work in the back. To put it another way, game engines are frameworks designed specifically to be used in development and creation of video games.[1] Unity, Unreal Engine, Source and many more examples can be given for game engines that are used today.

Since the creation, or even idea, of the first video game, developers have mainly used Euclidean geometry to represent their games. Because, Euclidean geometry is what we use in tasks we perform in our daily lives. From our way of perceiving the world to the way we represent the fiction world we use in our games. [2] Euclidean geometry that most games use is geometric representation of axioms and theorems on the plane and solid figures employed by the Greek mathematician Euclid.[3]

Video games always try to introduce new concepts to draw people into their game's "alternative reality".[2] As game developers were looking for these new concepts to introduce, some of them were interested in the idea of non-Euclidean geometry, which is literally any geometry that is not the same as Euclidean geometry. [4] Thus our objective is to give game developers some tools they can use to develop this type of games.

This report goes through related works that got developed before and lists software and libraries that can be used to both develop a game engine and implement non-Euclidean features.

1.1 Problem Statement

Game Engines don’t come packed with non-Euclidean functionalities. With this project, we aim to implement basic non-Euclidean functionality to our project and make it easy to access.

1.2 Background or Related Work

Not many works are done in this area, but we listed and written about them in our literature search.

1.3 Solution Statement

After our researches we understood that developing an addon for a popular game engine is better for accessibility. So, we choose to use Unity as our base application, which is one of the most used and most popular game engines.

1.4 Contribution

We researched old and upcoming non-Euclidean games and which game engines they used to choose our base application. Also, we researched how they implemented non-Euclidean functionality using these engines.

2. Literature Search

2.1 Similar and Related Works

There are many works that make use of non-Euclidean geometry. Some of them is listed in the following sections.

2.1.1 Hyperbolica

Hyperbolica is a game with a non-Euclidean curved space where you journey through bizarre landscapes, solve puzzles, navigate labyrinths and do much more while also being challenged to perceive the non-Euclidean space.[5] At least this is how the game developer of this game explains his work. It is a game in development but the developer of the game shares development logs in video format in his Youtube channel. [6]

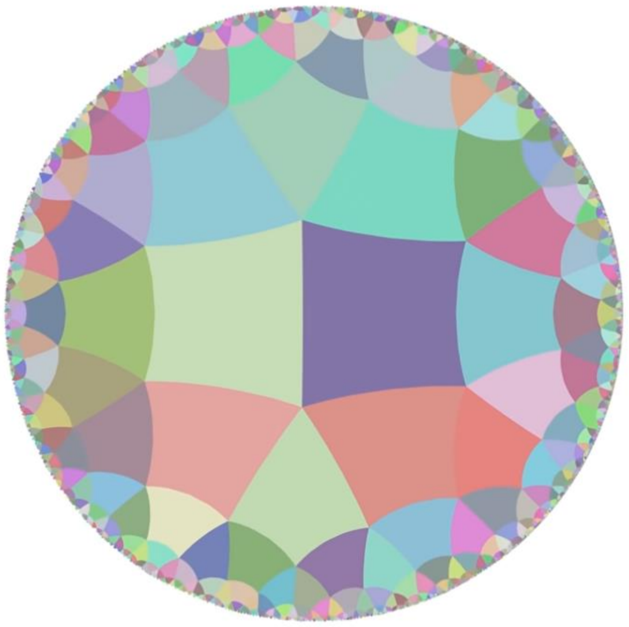

Figure 1 - Hyperbolica's Tilemap

The game uses Unity Game Engine and creates a hyperbolic geometry by making square per vertex number five instead of four (shown in Figure 1) and by making use of visual tricks and changes on perspective locations of game objects. It is a complex method and is detailed in a development log developer recorded and published on Youtube. [7] He also has other development log videos where he goes into detail of Non-Euclidean geometry and a video where they show a demo of their non-Euclidean game engine. [8] [9]

2.1.2 Antichamber

Game devs describe this work as being a mind-bending psychological exploration game in a Escher-like world where hallways wrap around upon each other, spaces reconfigure themselves and accomplishing what you think is impossible, maybe is the only way forward.[10] This game uses Unreal Engine 3 as their game engine. Antichamber uses many camera tricks and uses perspective manipulation to make some mind-blowing puzzles to solve. The game makes use of something called a stencil buffer which is an extra data buffer found on modern graphics hardware.[11] There are videos that go into detail about how Antichamber makes use of it. [12]

2.1.3 Superliminal

Superliminal is a puzzle game based on the perspective and optical illusions. What you do with an object in the game may cause unexpected results. Players need to change their perspective and think outside the box to solve puzzles and progress through the game. Game scales objects you interact with according to angular diameter and lets you solve puzzles by scaling objects to different sizes.

2.2 Software and Libraries

Many different software and libraries can be used to make a non-Euclidean game engine possible. We listed the primary options in the following sections.

2.2.1 Unity

Unity is the world's leading application for creating and developing real-time 3D content, which may be a game or an animation. It also provides a wide range of tools to make these content and lets users publish them to a wide range of devices. [14] Thus unity is one of the best game engines to make a game project in and Unity uses C# as its scripting language which is a common and easy to use programming language. It does not provide tools for making a game in non-Euclidean geometry and our task will be enhancing it with prefabs, scripts and even add-ons if needed to make a game engine for non-Euclidean projects using tricks on perception.

2.2.2 OpenGL

OpenGL is one of the first frameworks that come into mind for developing graphic applications. Since its first release in 1992, it has become the industry's most widely used and supported 2D and 3D graphics application programming interface. [15] OpenGL let us write a game engine from scratch giving us the ability to make non-Euclidean rendering the default option however it lacks the uncountable number of tools (i.e. objects, materials, collisions which will be needed to implemented from scratch) Unity provides us with, in exchange for more customization.

2.3 Conclusion

Using non-Euclidean geometry is a new concept that game developers make use of to expand concepts of game development. In our report we analyzed games done in non-Euclidean geometry and features they used to provide us with ways we can implement in our engine. Report also goes through software and libraries one can use to code a non-Euclidean game engine.

3. Summary

3.1 Summary of Conceptual Solution

Solution to the problem was to create prefabs and pre-written scripts that adds non-Euclidean functionality, ready to use with a simple drag and drop operation in the game development environment.

3.2 Technology Used

We used C# language and Unity Engine to create our project.

4. Software Requirements Specification

4.1 Introduction

4.1.1 Purpose

The purpose of this document is to explain the Non-Euclidean game engine. The goal of this game engine is to create games by using non-Euclidean geometric elements by going beyond the boundaries of current reality. This document contains detailed information about the project's requirements, reflecting the defined restrictions and recommended software functionality.

4.1.2 Scope of project

Most games today are made with game engines created based on known physics rules. For this reason, games are designed and created in accordance with Euclidean geometry. Whenever we want to make a project that is not suitable for Euclidean geometry, we have to design the mechanics of this game from the beginning. The main purpose of our project is to design a game engine ready to be used in the creation of games based on non-Euclidean geometry mechanics.

Most mechanics available are actually about smoothly moving the player from one location to another, directing the player to another location as opposed to the one they think they will reach, or altering the surrounding items or some material about the user indiscriminately during this smooth teleport to create an illusion.

While working on this project, it will be more beneficial for us to use the Unity Game Engine and C# language, which we are familiar with in this sector, and we think that it can be more efficient while getting help about the problems we face in the rest of the project.

4.2 Overall Description

4.2.1 Product Perspective

Non-Euclidean Game Engine is for game developers who are making games about Non-Euclidean geometry. These games generally use portals and camera tricks to change perspective. Game engine has 2 parts which are portals and camera tricks.

4.2.2 Development Methodology

For developing the project, we meet twice a week in the beginning and end of the week and we divide project parts among project members.

4.2.3 User Characteristics

4.2.3.1 Game Developer

- User could be anybody but preferably a game designer or game developer.
- User must be able to read and understand English language due to engine being English
- User must have knowledge of game making or game developing and read the guide provided beforehand.

4.3 Requirements Specification

4.3.1 External Interface Requirements

4.3.1.1 User Interfaces

Unity's own user interface will be used to control this project.

4.3.1.2 Hardware Interfaces

There are no external hardware interfaces.

4.3.1.3 Software Interfaces

Product requires Unity 2019.1 to work.

4.3.1.4 Communications Interfaces

There are no external communications interface requirements.

4.3.2 Functional Requirements

4.3.2.1 Teleport_Object Function

Objects in the game world will be teleported when they collapse with the Portal object.

Process:

1. Object collapses with the portal
2. If portal has a connected portal, object is teleported to that portal,
3. if portal doesn't have a connected portal but has coordinates for teleportation, object is teleported to that coordinates,
4. if the portal doesn't have both, the object doesn't get teleported.

4.3.2.2 Warp_Objects Function

Objects in the Non-Euclidean Area will be warped differently depending on the type of the Non-Euclidean Area, if the Non-Euclidean Area is turned on.

Process:

1. Game Developer needs to define an object as a player and/or camera.
2. If Player/Camera is collapsing with the Non-Euclidean Area;
3. System will check if the Non-Euclidean Area is turned on,
4. If it is turned on, objects will be warped according to type of the non-euclidean area,
5. if it is turned off or a player/camera is not identified, nothing will happen.
6. When the Player/Camera leaves the Non-Euclidean Area, the area will return to normal.

4.3.2.3 Change_Camera Function

Game Developers may want to use their own camera to change perception of the Non-Euclidean Area so depending on the type of the Non-Euclidean Area, the system will change the camera defined to the one the user wants.

Process:

1. Game Developer needs to define an object as a player and/or camera.
2. Game Developer needs to define a camera that will be changed with the current one.
3. When Player/Camera collapses with the Non-Euclidean Area, the camera will be changed with the one the Game Developer has chosen.
4. When Player/Camera leaves the Non-Euclidean Area, the camera will be changed with the default one.

4.3.2.4 Basic Game Engine Functions

Other than our project's functionality, a basic game also needs to contain some functionality. These are;

- Gravity,
- Object Collision,
- Character movement,
- Character-Object Interaction,
  and many more individual functionalities based on the needs of the game Game Developer plans to develop.

4.3.3 Performance Requirement

Performance is highly relevant to the project the user is working on, scaling with its size. But as a minimum, Unity 2019.1 requires:

- Windows 7, 8, 10, 64-bit operating system,
- graphics card with DX10+ capabilities,
- and an appropriate amount of space in the disk.

4.3.4 Design Constraints

Software will be designed as a set of non-Euclidean assets to help developers build non-Euclidean game projects. As the plot or gameplay of their project may require specific elements that we can't predict, this project will only cover common elements that can be used in a variety of non-Euclidean game projects.

4.3.5 Software System Attributes

4.3.5.1 Portability

Non-Euclidean Game Engine is a set of assets for Unity thus it can be deployed to any Unity instance in any computer with ease.

4.3.5.2 Performance

Performance scales with the needs of the user, the assets only requires Unity's own system requirements to work.

4.3.5.3 Usability

Assets are a common part of Unity's own usage, thus this product will be easy to use with basic knowledge about Unity itself.

4.3.5.4 Availability

Assets will work on Unity 2019+ and all operating systems that Unity 2019+ works on.

5. Software Design Description

5.1 Introduction

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

5.1.1 Purpose

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

5.1.2 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.

5.1.3 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.

5.2 Architecture Approach

5.2.1 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.

5.2.2 Software to Use

This project will be a set of assets for Unity Game Engine. 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.

5.2.3 Class Diagram

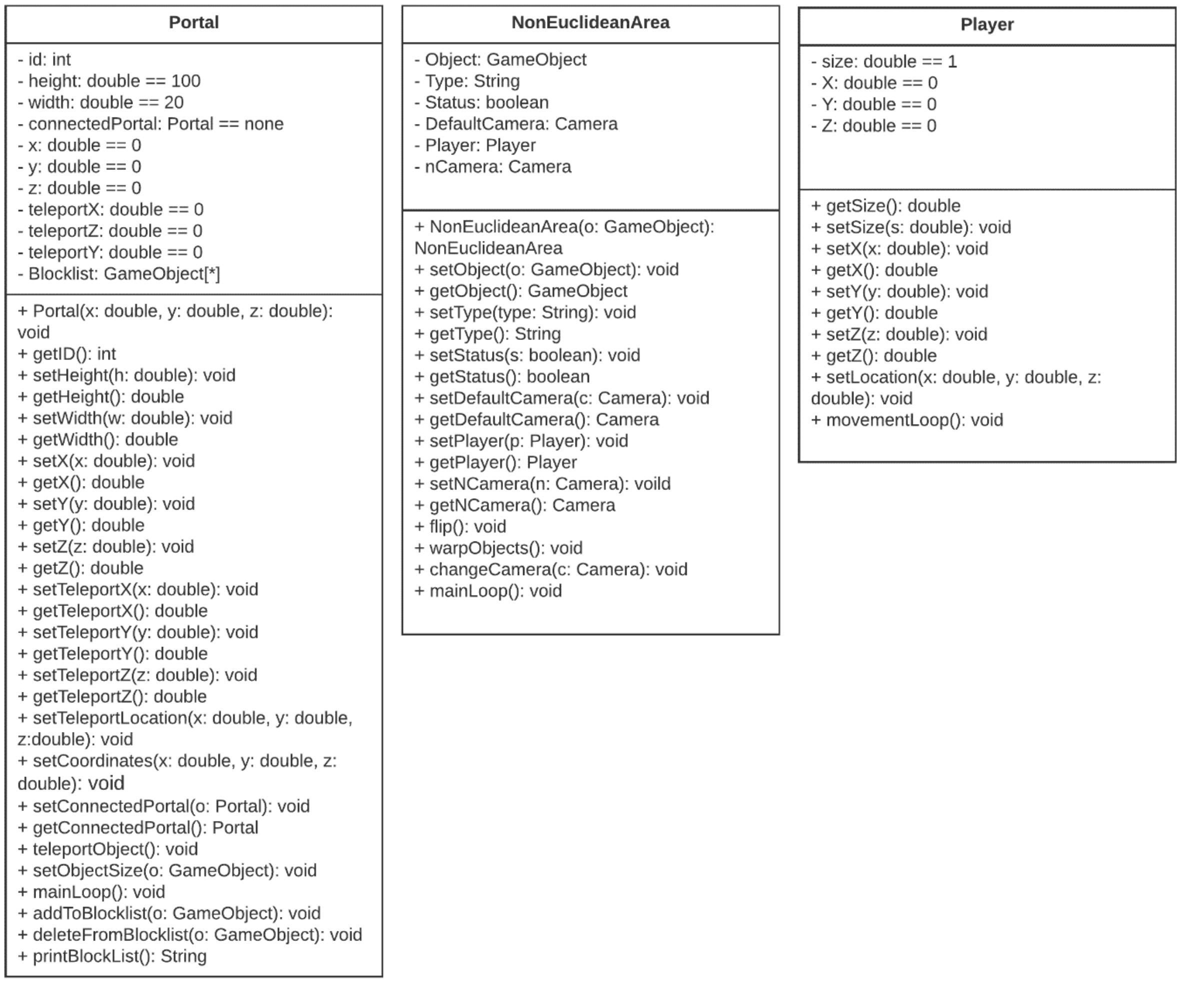
Portal
- id: int
- height: double == 100
- width: double == 20
- connectedPortal: Portal == none
- x: double == 0
- y: double == 0
- z: double == 0
- teleportX: double == 0
- teleportZ: double == 0
- teleportY: double == 0
- Blocklist: GameObject[*]
+ Portal(x: double, y: double, z: double): void
+ getID(): int
+ setHeight(h: double): void
+ getHeight(): double
+ setWidth(w: double): void
+ getWidth(): double
+ setX(x: double): void
+ getX(): double
+ setY(y: double): void
+ getY(): double
+ setZ(z: double): void
+ getZ(): double
+ setTeleportX(x: double): void
+ getTeleportX(): double
+ setTeleportY(y: double): void
+ getTeleportY(): double
+ setTeleportZ(z: double): void
+ getTeleportZ(): double
+ setTeleportLocation(x: double, y: double, z:double): void
+ setCoordinates(x: double, y: double, z: double): void
+ setConnectedPortal(o: Portal): void
+ getConnectedPortal(): Portal
+ teleportObject(): void
+ setObjectSize(o: GameObject): void
+ mainLoop(): void
+ addToBlocklist(o: GameObject): void
+ deleteFromBlocklist(o: GameObject): void
+ printBlockList(): String
NonEuclideanArea
- Object: GameObject
- Type: String
- Status: boolean
- DefaultCamera: Camera
- Player: Player
- nCamera: Camera
+ NonEuclideanArea(o: GameObject): NonEuclideanArea
+ setObject(o: GameObject): void
+ getObject(): GameObject
+ setType(type: String): void
+ getType(): String
+ setStatus(s: boolean): void
+ getStatus(): boolean
+ setDefaultCamera(c: Camera): void
+ getDefaultCamera(): Camera
+ setPlayer(p: Player): void
+ getPlayer(): Player
+ setNCamera(n: Camera): voild
+ getNCamera(): Camera
+ flip(): void
+ warpObjects(): void
+ changeCamera(c: Camera): void
+ mainLoop(): void
Player
- size: double == 1
- X: double == 0
- Y: double == 0
- Z: double == 0
+ getSize(): double
+ setSize(s: double): void
+ setX(x: double): void
+ getX(): double
+ setY(y: double): void
+ getY(): double
+ setZ(z: double): void
+ getZ(): double
+ setLocation(x: double, y: double, z: double): void
+ movementLoop(): void

Figure-2: Class Diagram

5.2.3.1 Class Diagram Description

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.

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.

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.

5.2.4 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 project.

5.3 Flowchart

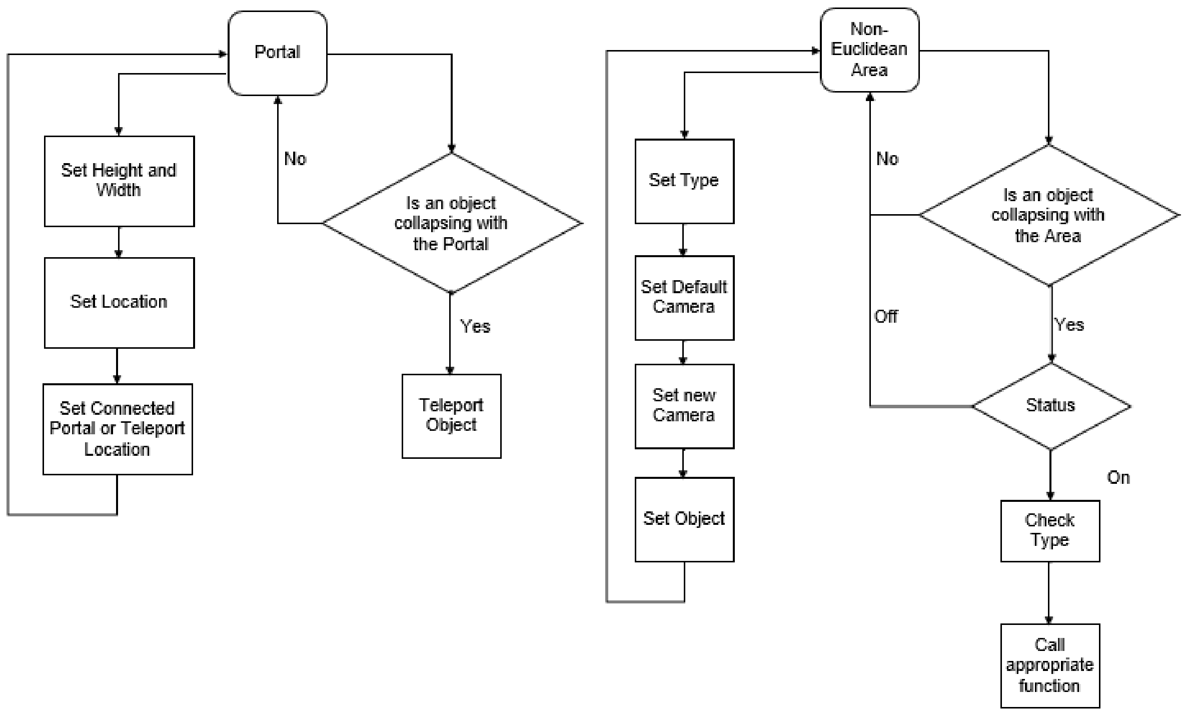

Figure-3: Flowchart

Figure-2 shows usage of assets as a flowchart and how it should work.

6. Test Plan

6.1 INTRODUCTION

6.1.1 Version Control

Version No	Description of Changes	Date
1.0	First Version	April 16, 2021

6.1.2 Overview

Assets for Non-Euclidean Game Engine will be tested to see if they are running as planned and written in SRS and SDD.

6.1.3 Scope

This document includes test plan, test design specifications and detailed test cases that will be used in testing non-Euclidean Game Engine.

6.2 FEATURES TO BE TESTED

This section lists and gives a brief description of all the major features to be tested. For each major feature there will be a Test Design Specification added at the end of this document.

6.2.1 Player Movement (PM)

Player movement is the basic concept of controlling an object to traverse game world. Player movement will be tested to see if there are any unintended controls to fix them.

6.2.2 Environment Interaction (EI)

Environment interaction testing will be done to see if player can interact with objects in the game world to move/pick up objects, open/close doors, pulling levers etc.

6.2.3 Portal (PT)

Portal teleports players from one place to another with or without them knowing it while also shrinking/expanding them if wanted. Portal testing will be done to see if portal can seamlessly teleport players, shrinks/expands as intended etc.

6.3 ITEM PASS/FAIL CRITERIA

6.3.1 Exit Criteria

- 100% of the test cases are executed
- 95% of the test cases passed
- All High and Medium Priority test cases passed

6.4 REFERENCES

[1] Non-Euclidean Game Engine - Software Requirements Specification, April 9, 2021
[2] Non-Euclidean Game Engine - Software Design Description, April 9, 2021

6.5 TEST DESIGN SPECIFICATIONS

6.5.1 Player Movement (PM)

6.5.1.1 Sub features to be tested

6.5.1.2.1 Movement (PM.M)

Player can move around the scene by walking/sprinting.

6.5.1.2.2 Jumping (PM.J)

Player can jump.

Camera Control (PM.CC)

Player can turn and look around the scene.

6.5.1.2 Test Cases

Here list all the related test cases for this feature

TC ID	Requirements	Priority	Scenario Description
PM.M.01	3.2.4	H	Press “w” button to move forward.
PM.M.02	3.2.4	H	Press “a” button to move left.
PM.M.03	3.2.4	H	Press “d” button to move right.

PM.M.04	3.2.4	H	Press “s” button to move backwards.
PM.M.05	3.2.4	H	Press “shift” and “w” buttons to sprint.
PM.J.01	3.2.4	H	Press “space” button to jump.
PM.CC.01	3.2.4	H	Move mouse to turn around.

6.5.2 Environment Interaction (EI)

6.5.2.1 Sub features to be tested

6.5.2.1.1 Interact (EI.I)

Use a button to interact with object to perform specified action. Open/close for doors, on/off for buttons etc.

6.5.2.1.2 Pick Up (EI.PU)

Pick up an object in the game world and move it.

6.5.2.2 Test Cases

Here list all the related test cases for this feature

TC ID	Requirements	Priority	Scenario Description
EI.I.01	3.2.4	H	Press “E” button to interact with object.
EI.PU.01	3.2.4	H	Press “F” button to pick up the object.
EI.PU.02	3.2.4	H	Move forward after picking up the object.
EI.PU.03	3.2.4	H	Press “F” while holding the object to drop the object.

6.5.3 Portal (PT)

6.5.3.1 Sub features to be tested

6.5.3.1.1 Seamless Portal (PT.SP)
Portal can replicate its connected portal’s view.

6.5.3.1.2 Teleport (PT.T)
Portal can teleport player to its determined destination.

6.5.3.1.3 Shrink/Expand (PT.SE)
Portal can shrink/expand player if wanted.

6.5.3.2 Test Cases
Here list all the related test cases for this feature

TC ID	Requirements	Priority	Scenario Description
PT.SP.01	3.2.1	H	Place portal and look through it.
PT.T.01	3.2.1	H	Place portals and go through them as player.
PT.SE.01	3.2.1	H	Place two portals with different sizes and go back and forth.

6.6 Detailed Test Cases

6.6.1 PM.M.01

TC_ID	PM.M.01
Purpose	Player should move forward.
Requirements	3.2.4
Priority	High
Estimated Time Needed	10 seconds
Dependency	-
Setup	Start game environment.
Procedure	[A01] Press “w”.
	[V01] Observe that player moves forward.
Cleanup	-

6.6.2 PM.M.02

TC_ID	PM.M.02
Purpose	Player should move left.
Requirements	3.2.4
Priority	High
Estimated Time Needed	10 seconds
Dependency	-
Setup	Start game environment.
Procedure	[A01] Press “a”.
	[V01] Observe that player moves left.
Cleanup	-

6.6.3 PM.M.03

TC_ID	PM.M.03
Purpose	Player should move right.
Requirements	3.2.4
Priority	High
Estimated Time Needed	10 seconds
Dependency	-
Setup	Start game environment.
Procedure	[A01] Press “d”.
	[V01] Observe that player moves right.
Cleanup	-

6.6.4 PM.M.04

TC_ID	PM.M.04
Purpose	Player should move backwards.
Requirements	3.2.4
Priority	High
Estimated Time Needed	10 seconds
Dependency	-
Setup	Start game environment.
Procedure	[A01] Press “d”.
	[V01] Observe that player moves backwards.
Cleanup	-

6.6.5 PM.M.05

TC_ID	PM.M.05
Purpose	Player should sprint.
Requirements	3.2.4
Priority	High
Estimated Time Needed	10 seconds
Dependency	PM.M.01
Setup	Start game environment.
Procedure	[A01] Press “w” and “shift” at the same time.
	[V01] Observe that player sprints forward.
Cleanup	-

6.6.6 PM.J.01

TC_ID	PM.J.01
Purpose	Player should jump.
Requirements	3.2.4
Priority	High
Estimated Time Needed	10 seconds
Dependency	
Setup	Start game environment.
Procedure	[A01] Press “space”.
	[V01] Observe that player jumps.
Cleanup	-

6.6.7 PM.CC.01

TC_ID	PM.CC.01
Purpose	Player should turn around.
Requirements	3.2.4
Priority	High
Estimated Time Needed	10 seconds
Dependency	
Setup	Start game environment.
Procedure	[A01] Move mouse in different directions.
	[V01] Observe players view area changes.
Cleanup	-

6.6.8 EI.I.01

TC_ID	EI.I.01
Purpose	Player should interact with the object.
Requirements	3.2.4
Priority	High
Estimated Time Needed	1 minute
Dependency	
Setup	Add an interactable object to game world and start game environment.
Procedure	[A01] Press “E”.
	[V01] Observe that player interacts with the object in the intended way.
Cleanup	-

6.6.9 EI.PU.01

TC_ID	EI.PU.01
Purpose	Player should pick up the object.
Requirements	3.2.4
Priority	High
Estimated Time Needed	3 Minute
Dependency	
Setup	Add a pickable object to game world and start game environment.
Procedure	[A01] Press “F”.
	[V01] Observe that player picks up the object.
Cleanup	-

6.6.10 EI.PU.02

TC_ID	EI.PU.02
Purpose	Player should move the object.
Requirements	3.2.4
Priority	High
Estimated Time Needed	3 minutes
Dependency	PM.M.01
Setup	Add a pickable object to game world, start game environment and pick it up
Procedure	[A01] Press “w”.
	[V01] Observe that player moves the object.
Cleanup	-

6.6.11 EI.PU.03

TC_ID	EI.PU.03
Purpose	Player should pick up the object.
Requirements	3.2.4
Priority	High
Estimated Time Needed	3 minutes
Dependency	-
Setup	Add a pickable object to game world, start game environment and pick it up.
Procedure	[A01] Press “F”.
	[V01] Observe that player drops the object.
Cleanup	-

6.6.12 PT.SP.01

TC_ID	PT.SP.01
Purpose	Player should see through the portal.
Requirements	3.2.1
Priority	High
Estimated Time Needed	5 minutes
Dependency	-
Setup	-
Procedure	[A01] Place two portals at different places.
	[A02] Connect them from inspector.
	[A03] Start game environment.
	[A04] Look through portal.

	[V01] Observe that player can see the view of the connected portal.
Cleanup	-

6.6.13 PT.T.01

TC_ID	PT.T.01
Purpose	Player should teleport back and forth through portals.
Requirements	3.2.1
Priority	High
Estimated Time Needed	5 minutes
Dependency	PM.M
Setup	-
Procedure	[A01] Place two portals at different places.
	[A02] Connect them from inspector.
	[A03] Start game environment.
	[A04] Go through portal.
	[V01] Observe that player teleports to the connected portal.
Cleanup	-

6.6.14 PT.SE.01

TC_ID	PT.SE.01
Purpose	Player should shrink or expand according the size differences of connected portals.
Requirements	3.2.1
Priority	High

Estimated Time Needed	5 minutes
Dependency	PM.M, PT.T.01
Setup	-
Procedure	[A01] Place two portals at different places with different sizes.
	[A02] Connect them from inspector.
	[A03] Start game environment.
	[A04] Go through the smaller portal.
	[A04] Go back through the bigger portal.
	[V01] Observe that when player goes through the smaller portal it expands and when it goes through the bigger portal it shrinks.
Cleanup	-

7. Test Results

7.1 Individual Test Results

TC ID	Priority	Results	Explanation
PM.M.01	H	Pass	
PM.M.02	H	Pass	
PM.M.03	H	Pass	
PM.M.04	H	Pass	
PM.M.05	H	Pass	
PM.J.01	H	Pass	
PM.CC.01	H	Pass	
EI.I.01	H	Fail	Will be depreciated.
EI.PU.01	H	Pass	
EI.PU.02	H	Pass	
EI.PU.03	H	Pass	
PT.SP.01	H	Pass	
PT.T.01	H	Pass	
PT.SE.01	H	Pass	

7.2 Summary of Test Results

Out of 14 tests 1 failed, which will be depreciated and will not be used.

7.3 Exit Criteria

Exit criteria has been made

Criteria	**Met or Not**
100% of the test cases are executed	Y
95% of the test cases passed	Y
All High and Medium Priority test cases passed	Y

8. Conclusion

We managed and developed a small-scale project. During the development and management of this project, we learned how to communicate with team members better and how to use Unity & GitHub more in-depth. Our aim was to develop a set of pre-written code and objects that can be used in development non-Euclidean games with ease, we accomplished what we aimed for and it may be further developed to be better and more inclusive of different functionality as it only covers mainstream part of non-Euclidean games and geometry.

Acknowledgement

We are grateful to our advisor Faris Serdar Taşel for guiding us through this project, answering our questions and helping us understand parts we don’t know about. We are also grateful to other researches and game developers who shared their researches and experiences on the matter.

References

[1] Studytonight.com. What Is A Game Engine? | Studytonight. [online] Available at: https://www.studytonight.com/3d-game-engineering-with-unity/game-engine [Accessed 6 November 2020].

[2] Guimarães, F. and Mello, V., 2015. Geometry Independent Game Encapsulation For Non-Euclidean Geometries. [ebook] Available at: https://www.visgraf.impa.br/Data/RefBib/PS_PDF/sib2015filipe/wip-filipe.pdf [Accessed 6 November 2020].

[3] Encyclopedia Britannica. Euclidean Geometry. [online] Available at: https://www.britannica.com/science/Euclidean-geometry [Accessed 6 November 2020].

[4] Encyclopedia Britannica. Non-Euclidean Geometry. [online] Available at: https://www.britannica.com/science/non-Euclidean-geometry [Accessed 6 November 2020].

[5] Steam. Hyperbolica. [online] Available at: https://store.steampowered.com/app/1256230/Hyperbolica/ [Accessed 6 November 2020].

[6] Youtube. Codeparade. [online] Available at: https://www.youtube.com/channel/UCrv269YwJzuZL3dH5PCgxUw [Accessed 6 November 2020].

[7] Youtube. 2020. Rendering Hyperbolic Spaces - Hyperbolica Devlog #3. [online] Available at: https://www.youtube.com/watch?v=pXWRYpdYc7Q [Accessed 6 November 2020].

[8] Youtube. 2020. Non-Euclidean Geometry Explained - Hyperbolica Devlog #1. [online] Available at: https://www.youtube.com/watch?v=zQo_S3yNa2w [Accessed 6 November 2020].

[9] Youtube. 2018. Non-Euclidean Worlds Engine. [online] Available at: https://www.youtube.com/watch?v=kEB11PQ9Eo8 [Accessed 6 November 2020].

[10] Steam. Antichamber. [online] Available at: https://store.steampowered.com/app/219890/Antichamber/ [Accessed 6 November 2020].

[11] Wikipedia. Stencil Buffer. [online] Available at: https://en.wikipedia.org/wiki/Stencil_buffer#:~:text=A%20stencil%20buffer%20is%20an,RAM%20of%20the%20graphics%20hardware [Accessed 6 November 2020].

[12] Youtube. 2020. How do non-euclidean games work? | Bitwise. [online] Available at: https://youtu.be/lFEIUcXCEvI?t=540 [Accessed 6 November 2020].

[13] Steam. Superliminal. [online] Available at: https://store.steampowered.com/app/1049410/Superliminal/ [Accessed 6 November 2020].

[14] Unity. Unity Platform | Unity. [online] Available at: https://unity.com/products/unity-platform [Accessed 6 November 2020].

[15] OpenGL. 2020. Opengl Overview. [online] Available at: https://www.opengl.org/about/ [Accessed 6 November 2020].