Procedural world generation and the challenges of rendering in non-Euclidean spaces

Aleksy Bałaziński Karol Denst

October 19, 2023

Contents

1 Abstract

Procedural generation, as a method used in creating video games, has experienced a significant increase in popularity in recent years. It has been employed extensively in acclaimed titles such as *Minecraft* and *No Man's Sky* to create virtually infinite worlds that the player is free to interact with. On the other hand, a recent game *Hyperbolica* shed light onto a novel idea of making the virtual worlds even more interesting by setting them in non-Euclidean spaces. The objective of this thesis is to create a small video game incorporating both of the aforementioned concepts.

1.1 History of changes

Date	Author	Description	Version
18.10.2023	Aleksy Bałaziński Karol Denst	Initial description of the project	1.0

2 Vocabulary

Hyperbolic geometry – geometry obtained by replacing Euclid's parallel postulate with the following axiom: "For any given line R and point P not on R, in the plane containing both line R and point P there are at least two distinct lines through P that do not intersect R."
Elliptic geometry – geometry obtained by replacing Euclid's parallel postulate with the following axiom: "Through any point in the plane, there exist no lines parallel to a given line."
Procedural world generation – a technique that allows for creating content (commonly terrain) in a video game, animated movie etc. in an automated manner, that is using an algorithm rather than using manually crafted assets.
NPC – non-player character.
FPS – frames per second.
GB – gigabyte.
RAM – random access memory.

3 Specification

3.1 Executive summary

The video game is written in C#. The two main libraries used in the development are OpenTK (used for rendering) and BepuPhysics (as the physics engine). Models used in the game were created in Blender. Techniques of rendering in non-Euclidean spaces are based on a paper by Szirmay and Kalos. Terrain generation was implemented using the marching cubes algorithm, where the underlying scalar field was generated using Perlin noise.

3.2 Functional requirements

As a player I want to	So that	Flags
Start a game in Euclidean space	I can explore Euclidean spaces	Must
Start a game in hyperbolic space	I can explore hyperbolic spaces	Must
Start a game in spherical space	I can explore spherical space	Must
Have gravity in the game	I have Earth-like experience	Must
Can collide with other objects in the game	I can interact with other objects	Must
Be able to move around	I can explore the world	Must
Be able to jump	I can jump	Must
Be able to shoot projectiles	I can see how things move in non-Euclidean spaces	Must
Have a limited number of projectiles in the inventory	The game is more realistic	Must
Be able to interact with NPCs	So that there is more to do in the game	Should
Have NPCs move around	The world looks more rich	Must
Edit the terrain	I can view any structure in different spaces	Must
See the marker of the terrain editor	I can decide where the terrain modification is going to take place	Must
Have different terrain generated every time I load a game	I can explore different terrains	Must
Have light sources that illuminate other objects	I can see what lighting looks like in different spaces	Must
See a night/day cycle	I can compare the world during the day and the night	Should
Have a vehicle I can enter	Move around faster	Must
Have a vehicle with suspension springs	I can drive around comfortably	Must
Have a vehicle that has two top velocities	I can drive faster if I get bored	Must
Have models for the player and NPCs	Characters look good	Must
Have model for the vehicle	Vehicle looks good	Must
Have an inventory	I can manage different items	Must
Have objects cast shadows	I can explore what shadows look like in different spaces	Should
Have a way to save/load a game	Be able to store interesting worlds	Must
Be able to save a game	I can save interesting worlds	Must
Be able to load a game	I can revisit interesting worlds	Must
Have a CLI	I can manage my saves and other game options	Must
Have a `help` option in the CLI	I know what commands are available and what they do	Must
Have GUI	I can manage my saves easily	Should

Table 1: Player functional requirements

3.3 Non-functional requirements

Requirements area	Requirement #	Description
Functionality	1	The application should offer world generation in 3 different spaces as well as means to explore them.
Usability	2	The application should have a menu and gameplay controls.
Reliability	3	The application should run without crashing or memory leaks for extended periods of time.
Performance	4	The application should run at 60 FPS with a resolution of 1920x1080 on a computer with at least Intel Core i7-1260P and 16 GB of RAM.
Sustainability	5	The game should save logs and world data to make solving any potential bugs easier.

Table 2: Project non-functional requirements

4 Project schedule

Task	Begin Date	End Date
Implement a basic OpenTK application Implement terrain generation Implement lighting Implement mining/building Implement collision physics Add gravity Add a drivable car Implement loading and displaying models Implement model animation Implement shooting projectiles Implement ray casting Add NPCs Implement inventory system Implement a simple CLI Implement displaying hitboxes Implement load and save system for the game	01/10/2023	19/10/2023
Add better models for NPCs, vehicles and the player	19/10/2023	02/11/2023
Add shadows	19/10/2023	09/11/2023
Add day/night cycle	02/11/2023	16/11/2023
Add a menu	17/11/2023	01/12/2023
Implement hyperbolic space Implement spherical space Add different terrains to the generation	01/10/2023	07/12/2023

Table 3: Project schedule

5 Risk analysis

SWOT	Threats	Opportunities
Internal	• Not enough time to implement all features • Bad design decisions hindering further development • Failure to implement rendering in non-Euclidean spaces	• Project developed with best practices in mind, resulting in easily extensible code. • Project employs the techniques of object-oriented programming to make it easier to manage state by exposing simple, well-defined interfaces. • Project is well documented, making it easier to understand the code base.
External	• External libraries don't support highly specific features • Poor documentation (or lack thereof) for external libraries	• Sparking interest in non-Euclidean geometries • Proposed methods of rendering non-Euclidean geometries could be used in other projects

6 References