Visualizing Procedurally Generated Roller Coasters

Ege Guney
[1] Department of Computer Science, McGill University

April 4th 2024

This Project transforms the procedurally generated roller coaster track obtained from roller coaster tycoon 2 game (RCT2), and transforms it into a roller coaster track in unity that is similar to the RCT2 version. By enabling a direct transformation of the track, this project aims to provide realistic simulation experience, while obtaining potentially impactful data from the ride and user.
Keywords: Roller Coasters, Unity, Simulation, Procedural Generation, Visualization

1 Introduction

Roller Coasters both in real life and in games have been used as entertainment tools for centuries. There are various tools to analyze or modify the current virtual roller coasters existing in games or simulations, however in this project we procedurally generate and simulate these roller coasters in Unity.
Goals of procedurally generating roller coasters will help us gain more precise information about various roller coasters, including but not limited to the user experience, limits of accelerations and environmental aspects. By providing an immersive ride experience, this project allows users to act as if they are actually riding a roller coaster.
Roller coasters are created with a specific input format obtained from the game Roller Coaster Tycoon 2. By taking advantage of open source unity packages, this process converts a text file input to a functioning roller coaster simulation with real time visual data retrievers. This project allows potential researchers to gain insight on the general behavior of humans on roller coasters and on which parts of the roller coasters are giving the users the most ”excitement” or anticipation.

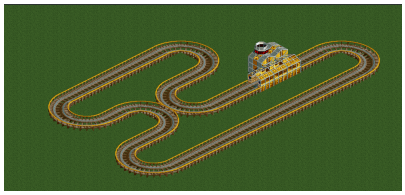

Figure 1: RCT2 Curves Track

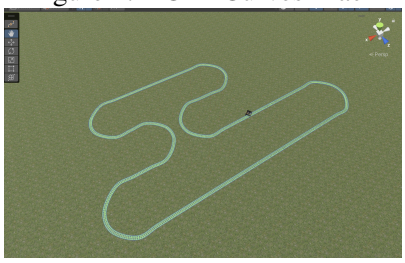

Figure 2: Unity Curves Track

2 Purpose

The purpose of this research project is to explore the utilization of Roller Coaster Tycoon 2 (RCT2) for simulating and visualizing roller coasters inside an ’open’ environment. Roller Coaster Tycoon 2 is a popular simulation video game that allows players to design and manage amusement parks, including the construction of intricate roller coaster rides. By leveraging the roller coaster formats of RCT2, this project aims to develop a platform that enables users to simulate and visualize roller coasters in a more accessible and interactive manner.

The primary goal of this project is to enhance accessibility and ease of use for individuals interested in analyzing and modifying roller coaster designs. By providing a platform that utilizes RCT2 input formats, users will be able to directly import roller coaster data from the game into our simulation environment.

Furthermore, the project aims to offer various tools for analyzing user experience factors of roller coaster rides. Through visualization techniques, users will be able to observe the simulated rides from different

perspectives, gaining insights into the behavior of the coaster under different conditions.

Overall, the ultimate goal of this research project is making it possible for various research practices or experiments to be performed both on and around the roller coasters. These could include testing different parameters on humans and/or human-like behaviours in unity with different output data retrieval methods.

3 Methodology

Methodology consists of several parts, starting from the input file format. Then report includes information about the data and the shape of the splines, turns and loops, support structures, physics on the cart and the track, and output formats.

3.1 Track Creation

Track was created using Unity 2022.3 with Splines package installed. Input file is parsed to a class called Track within the TrackUnit class. This class has the following values as fields.

```
trackType
coordinates
adjustedCoordinates
direction
trackClassification
trackSlope
trackBank
speed
lateralGForce
verticalGForce
```

3.1.1 Input Format

Track Type, Coordinates, Track Station End, Track Slope Start, Track Slope End, Track Bank Start, Track Bank End Speed (Lateral G force) (Vertical G Force)

This data was processed and utilized to instantiate track units within the Unity environment, ensuring alignment with realism criteria.

Adjustments to coordinate values were made to enhance realism within the Unity environment. The adjustment process involved modifying coordinate values based on specific modifiers to ensure accurate representation within the simulated environment.

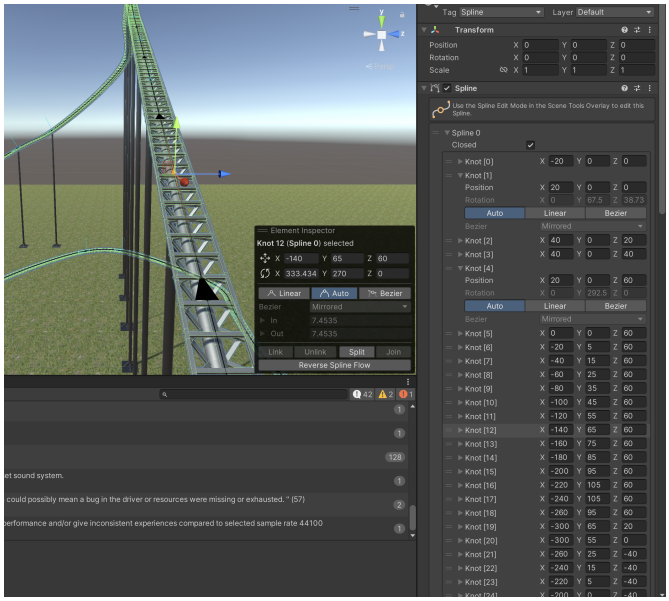

Figure 3: Spline Editor Structure

```
private int xModifier;
private float yModifier;
private int zModifier;

void adjustCoordinateValuesV2(...)
{
    xModifier = 20;
    yModifier = 5f;
    zModifier= 20;
    TrackStructure tr;
    ...
}
```

3.1.2 Splines

After creating Adjusted coordinate values from input, the algorithm creates a Bezier knot from this coordinate (Provided in the Splines Package in Unity).
The provided track unit is then turned into a Track Structure object, indicating the index and whether or not the unit will be used in the created track. This object is then stored in a list that will then be used to create and instantiate the track.
While creating the spline from the coordinates, the connections between each bezier knot are calculated automatically using an automated connection method provided in the splines package. Other aspects like rotation and banks of the track are adjusted while creating the knots.

3.1.3 Spline Data

Spline Track holds the data for location of each ”adjusted” coordinate and the speed value related to that coordinate in it’s data storage provided by the package. Later, after this data is created, other files/scripts can

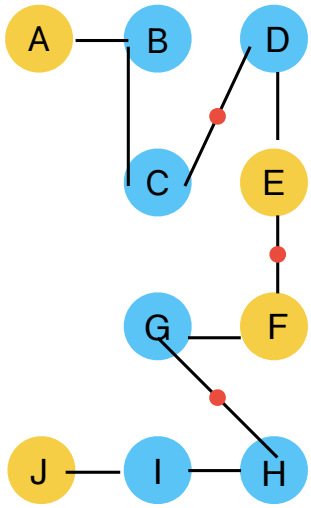

Figure 4: *Turn Structure on input file. Input File has them in the alphabetical order. Adjusted format only keeps track of A,E,F,J..*

access this data without the need to access any file or other script.

3.1.4 Track Instantiation

Track instantiation is done after creating the spline. Spline instantiation script is run internally after creating everything related to the track as a spline. Spline instantiation algorithm fills the spline with small track unit objects. After this process, track units can be baked manually to access them as game objects.

3.1.5 Turns & Loops

The variability in input file formats necessitated employing different algorithms to handle various types of turns. Challenges arose particularly due to the discrepancy between the coordinate space used in RollerCoaster Tycoon 2 (RCT2) and the float values required for precise trajectory representation. To address this, adaptive strategies were employed based on the characteristics of each turn type.
Another way of fixing the curves was to use the red highlighted middle points in Figure 4. However, this format proved to be inefficient and harder to generalize.
Temporary values were generated to keep track of the start and approximate end points of turns, enabling the creation of smooth and accurate curves within the Unity environment.

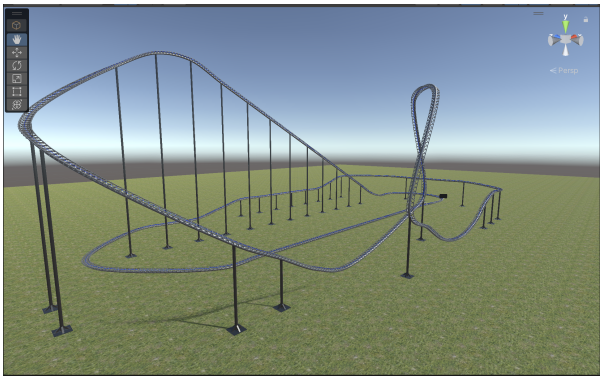

Figure 5: TrackPicture

```
if(tIn.TrackType.Contains("5")){
    turnParser = 7;
}
if(tIn.TrackType.Contains("3")){
    turnParser = 4;
}
```

The above method creates a temporary value that keeps the start and the approximate end of the turn and uses those values to make the curve smooth and accurate.

3.1.6 Support

Support structures were generated post-track creation to enhance visual fidelity reminiscent of real-world roller coasters. Each adjusted coordinate, serving as a "knot" in spline terminology, was assessed for support placement, subject to height constraints and absence of neighboring "knots" directly below.

3.2 Simulation

3.2.1 Physics

The physics of the roller coaster cart were simulated utilizing the spline package within Unity. A floating-point value "t" was evaluated to determine the cart's position along the spline. This value ranged from 0 to 1, mapping to specific positions along the track. Manipulating the rate of change of "t," coupled with camera adjustments, facilitated an approximate simulation of roller coaster physics.

3.2.2 Cart

The roller coaster cart was constructed using basic Unity objects supplemented with appropriate assets for seating. The camera was positioned at head level to provide an immersive experience.

3.3 Data Retrieval

Data retrieval was accomplished through the instantiation of track instances using a spline instantiating

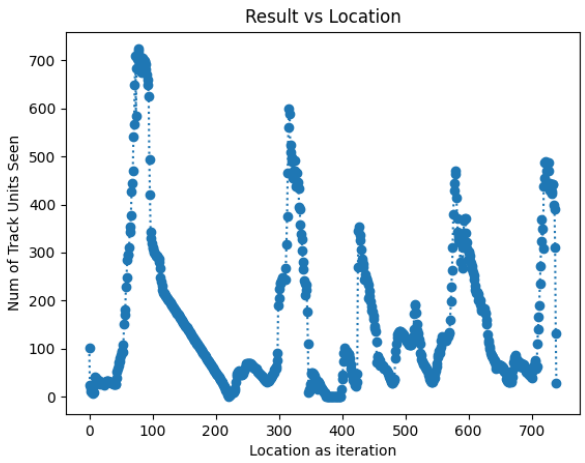

Figure 6: Resulting data from Figure 5

script, converting each small track unit into Unity game objects. In real-time, the algorithm computed the number of visible track units from the camera's perspective, logging the data in a specific format.
The output format comprised unit locations and corresponding perceived units from the camera's viewpoint, facilitating comprehensive data retrieval for analysis and visualization.
The format of the output is:
Unit location: (x,y,x); Result: (x1,y1,z1);(x2,y2,z2);...
Where the result is the list of different units that are perceived from the camera.

4 Results and Discussion

Overall, the implemented methodology demonstrated robustness and effectiveness in creating, simulating, and analyzing roller coaster tracks within the Unity environment. The project's outcomes lay a solid foundation for further research and development in the field of virtual roller coaster design and simulation.

4.1 Track Anticipation Data

The data derived from track anticipation in this project serves as a valuable resource for conducting experiments aimed at evaluating the effectiveness of visual elements and visibility dynamics within a roller coaster experience. These experiments can encompass a diverse array of environments, ranging from immersive virtual reality (VR) simulations to mobile platforms and in-person amusement park settings. By utilizing the outcomes of such experiments, researchers and industry professionals can gain insights into the characteristics of different roller coaster tracks and their effects on humans. Moreover, the findings obtained through these experiments hold the potential to inform and refine ongoing roller coaster designs. For example, the anticipation data obtained from this project can help the researches gain insight on which parts of the track can be modified.

4.2 Improvements & Future Work

There are several parts of the project waiting to be improved. These parts are highlighted and are providing various ways of improvement.

4.2.1 Assets

Cart and Track objects can be improved to provide more realistic roller coaster features. Support assets could be connected to create a trestle-like structure, again providing a more visually realistic roller coaster experience.

4.2.2 Simulation

Acceleration and deceleration could be made with more calculations, providing more accurate physics. Lateral and Horizontal G-forces could be added to camera movement, enhancing reality aspects.

4.2.3 Environment

Other Objects could be added to the environment, that would make the environment similar to RCT2. By adding various assets, researchers could experiment with different visual effects of the effects on the ride experience.

4.3 Versions

There exists two versions of input data and two versions of simulations within the project. First version is created with a directional data that is given in the input based on the track unit. This input format later changed, providing a better turn and loop unit information.
Version 1 can be reached with commenting out all the V2 code chunks and removing the comment from V1 methods. However, to fully create the ride experience, the user would need to add a spline package script called SplineAnimation.cs to the cart and attach it to the current spline(track).

5 Conclusions

By providing essential tools to create, change and experience the roller coaster rides in an open and experimental environment, this project helps open a new research path in roller coaster technology and experiments. Utilization of procedural roller coasters can and potentially will change the understanding of certain aspects of these rides. The simulation created in unity,

can be used to improve both in game and real life roller coasters. In conclusion, the project aims to create a roller coaster experience with unpredictable and procedural input, while providing an experimental, editable and open environment.

6 Declarations

6.1 Open Source Project

Link: `https://github.com/Ege-Guney/RollerCoaster-Simulator`

6.2 Acknowledgements

Packages & Assets Used:

- Unity Splines package
- 70s Car Seat by Tokyo Liu

6.3 Supervisors

- Prof. Clark Verbrugge
- Jonathan Campbell

7 References

Jonathan Campbell, Clark Verbrugge. ”Procedural generation of rollercoasters.” IEEE Conference on Games (CoG 2023).