VEHICLE DRIVING SIMULATOR WITH DATA ACQUISITION

David Michalík
Master Degree Programme (2nd year), FEEC BUT
E-mail: xmicha61@stud.feec.vutbr.cz

Supervised by: Miroslav Jirgl
E-mail: jirgl@feec.vutbr.cz

Abstract: This paper describes the creation of a vehicle driving simulator that collects and implements data acquired from a driver's inputs. These data are stored for future analysis of the driver and his style of driving. The simulator itself is realized through the Unreal Engine by Epic Games. The paper explains vital steps of the process such as importing vehicle models, creating armature for physics and the creation of the world itself, alongside the data overview of gathered information.

Keywords: vehicle driving, simulator, data acquisition, vehicle physics, game engine

1 INTRODUCTION

The main purpose of this project is to create a vehicle driving simulator which provides various ways of data acquisition. In order to do so, there are hardware components vital to the data acquisition, such as the steering wheel, pedals and a gear stick in order to simulate the conditions of a real car driving. In order for these hardware components to fulfil their purpose, a unique software application needs to be created. Although there are multiple game engines that offer a wide range of functions and graphical editors, for the creation of this simulator, the Unreal Engine developed by Epic Games is used, as it offers a user-friendly environment with multiple tools such as mesh, physics editors or blueprint programming (similar to Matlab - Simulink). Additionally, multiple assets are at the user's disposal in the engine's marketplace. In this case, an asset can be best described as an object, plugin or a tool that can be used with intended license, such as a car model.

2 REALIZATION OF THE SIMULATOR

In order to create the intended version of the vehicle driving simulator in the Unreal Engine, there are several key parts that need to be implemented:

- *Artificial world creation*
- *Importing vehicle model with armature*
- *Gameplay setup*
- *Data logging*

Perhaps the most important part of the process is creating the world itself, as the higher the quality of the world is, the higher the level of realism that player/driver experiences. With that in mind, a landscape is created with long distance objects around it that serve as simulated mountains around the environment. The goal is to create a sample map of a mountain city with several roads and buildings that will be sufficient for the showcase of the engine and its abilities. However, later on there is a need

to create a larger map that would represent a rural city area or even a city centre with functioning traffic and implemented driving rules. [1]

Another key part is the vehicle that a driver is going to operate in the world. First of all, a mesh is needed - a model of the vehicle created in a third party software such as Maya or Blender. In the simulator, a 3D open source model of a car is used with imported textures. The second step is to create an armature for the vehicle because the engine - more specifically Wheeled Vehicle component - needs to know what parts of the mesh need to have physics, collision and animation implemented. Such a thing can be done in a software like Blender, shown in Figure 1. There is a total of five bones (triangular objects in Figure 1) that complete the armature of the car, one for each of the wheels and one for the vehicle itself. Names of the bones are extremely crucial for the game engine, and so the parent core vehicle bone is called Root while others are named after the wheel they are assigned to: FL (Front Left), FR (Front Right), RL (Rear Left), and RR (Rear Right). [2]

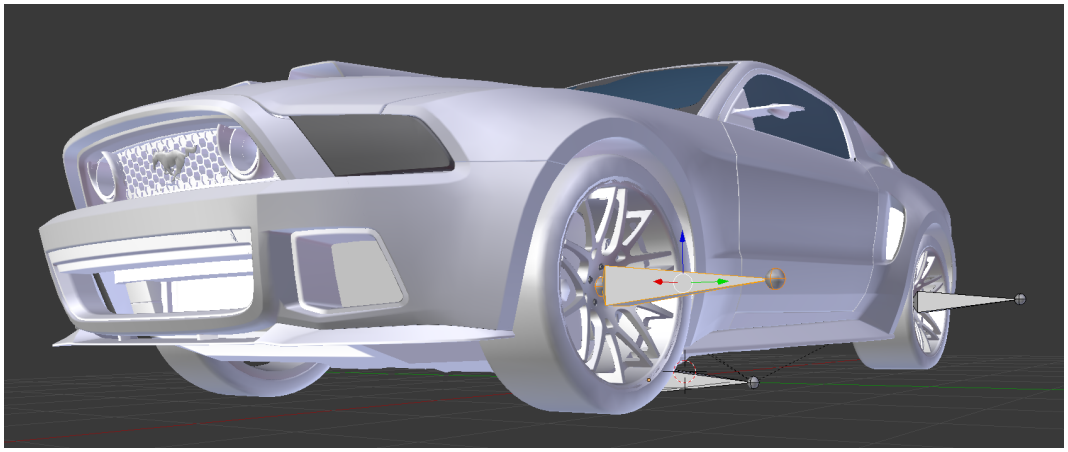


Figure 1: Model with implemented armature (bones)

When the mesh with implemented armature is completed, a so-called blueprint (a type of object in Unreal Engine) is created using the Wheeled Vehicle component. This component is based on NVIDIA's PhysX and controls the collision the vehicle has with other objects in the world so that the car fits on the ground and doesn't fall through. The bones that were created in the mesh help the physics asset handle the vehicle as a whole and each of the wheels separately. The animation asset on the other hand controls the movement of the vehicle components, which means that the wheels will turn and rotate based on a user's input. Wheel and tire configuration objects determine the behaviour of the wheels, for example the amount of rotation, friction or size of the collision on the wheels. [2]

In the same blueprint, a user's input is connected to steering and throttle output of the vehicle with the use of Unreal Engine's nodes. There are two ways to control the vehicle that the simulator offers; one with a standard W and S keys for throttle output and A and D keys for steering. The second way is by using the steering wheel and pedal controllers. The axis of the steering wheel is attached to the steering output and accelerator and break pedals are for throttle output.

When all of these steps are implemented - a functioning application is created, one where a user is free to drive around the landscape. Although there are currently few types of information that can be acquired about the driver, they are logged with the vehicle blueprint. Custom functions *Log Data On*

Screen and *Log Data To File* were created in C++ and inserted into the blueprint event handler to be used as nodes. The output of the *Log Data On Screen* function can be seen in the upper left corner in Figure 2 while the output of the *Log Data To File* function is saved as a .csv file.

Figure 2: First person view from the vehicle

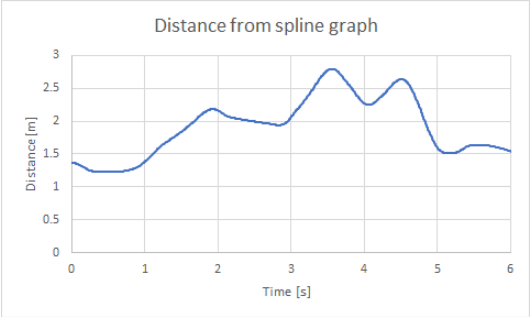
3 ACQUIRED DATA

In Table 1, an example of acquired data is presented from a single ride. As of now, the simulator logs velocity of the vehicle in kilometres per hour and the distance between the vehicle and a custom made spline component in the world. This data could be used in the future for measurement of the difference between a driver's path and an ideal path, presented by the spline component. There are multiple other things that can and should be monitored. Such as acceleration, steering wheel and pedals angle, the way of changing gears and others. These types of data will be included in the project in the future.

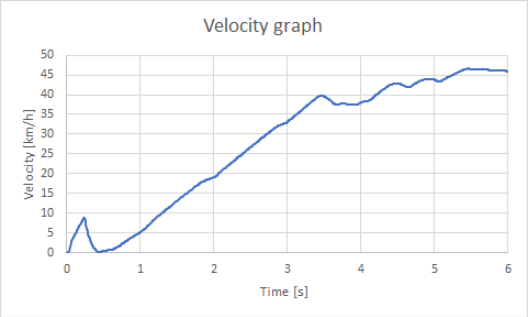
Time [s]	Distance from spline [m]	Velocity [km/h]
0.984	1.40	5.41
0.994	1.41	5.52
1.002	1.41	5.63
1.011	1.42	5.76
1.019	1.43	5.88

Table 1: Gathered data about the vehicle movement

Figures 3a and 3b shows graphs that are made from all the data gathered during the test ride. With the right form of data analysis, this data can be used to show statistics to a user about how safe his or her driving style is or how precise he or she handles the car. Additionally, Unreal Engine also offers a plugin that enables sending the data directly into the Matlab - Simulink, so that there can be two parallel programs running at the same time, providing real-time data analysis.



(a) Graph 1 - Distance from spline



(b) Graph 2 - Velocity

Figure 3: Graphs displaying acquired data from the test drive

4 CONCLUSION

It can be safely said that the simulator is still a work-in-progress and there are multiple ways to extend not only the amount and type of information about the driver, but also the increase of the level of realism so that the experience of a user increases in quality. For example, the interior of the car could be made more responsive for the driver and show information on the panel or turn the steering wheel according to a user's input. Traffic and its rules could also be implemented in the simulator world with a realistic model of a city with advanced graphics provided by Unreal Engine. There is a firm belief that in the future, this simulator could be expanded upon by anyone with game engine software knowledge, serving for various purposes and aims.

REFERENCES

[1] GREGORY, Jason. Game Engine Architecture, Second Edition. Second edition. A K Peters, 2014. ISBN 1466560010.

[2] TRISTEM, Ben, PATTUZZI, Sam. Unreal Engine C++ Developer: Learn C++ and Make Video Games. Udemy [online]. Updated 12/2018. Available at: https://www.udemy.com/unrealcourse [Accessed 10 Mar. 2019].

[3] Docs.unrealengine.com. (2019). Unreal Engine 4 Documentation. [online] Available at: https://docs.unrealengine.com/en-us/ [Accessed 10 Mar. 2019].