VIRTUAL REALITY AND VEHICLE DYNAMICS IN UNREAL ENGINE ENVIRONMENT

JAROSLAV MATEJ

Technical University in Zvolen

Department of Mechanics, Mechanical Engineering and Design, Zvolen, Slovak Republic

DOI:10.17973/MMSJ.2016_10_201688

e-mail: jaroslav.matej@tuzvo.sk

The paper is aimed on a vehicle dynamic simulation from scratch, using equations of motion solved by Runge-Kutta algorithm. The vehicle and a terrain interact in a game environment Unreal Engine, using line-trace functionality. All the scene objects are created in Creo Parametric, 3ds Max and Unreal Editor systems. The vehicle can be controlled by a keyboard and a racing steering wheel with pedals. The system allows free driving on the terrain in game-like style, recording and playback functions. All the computed parameters like vehicle's locations, rotations, forces, velocities and accelerations can be recorded and saved into file during free drive. Then the data can be visualized in playback mode, with displayed forces, in 2D space and 3D by means of virtual reality headset HTC Vive. Virtual reality allows the user a full spatial view on the vehicle's behavior.

KEYWORDS

vehicle dynamics, vehicle simulation, unreal engine, C++, virtual reality, virtual test facility

1 INTRODUCTION

The Unreal Engine (UE4) (Epic Games 2016) is an environment historically focused on game development. One of the most known game, created using this environment, is Unreal Tournament – one of the first 3D first person shooting games. In present the environment is freely available under a very friendly license, and it contains a development environment connected with Microsoft Visual Studio C++. We used Visual Studio to write the motion equations and all the necessary code to display and control the world (Fig. 1), and vehicle's objects using a steering wheel and a keyboard, and output the results.

The Unreal Engine is a game engine, which from point of view of the paper means that it is focused on real-time side of the simulation. In other words, if a load of the system is higher than actual system possibilities, the system will automatically decrease resolution or FPS (frames per second) rate to achieve real-time simulation. This can be a source of problems if a selected numerical integration method is used in order to get high precision data of vehicle's dynamical parameters.

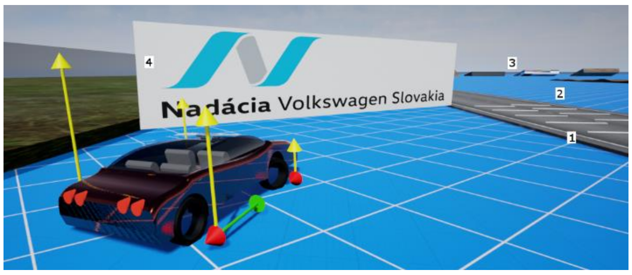

Figure 1. Virtual test facility: 1 – test road with obstacles, 2 – rough terrain, 3 – jump ramps, 4 – glass barrier and billboard

2 MATERIALS AND METHODS

A simulation environment we created consists of the terrain created by the means of Unreal Editor and imported objects from Creo Parametric 2.0. The vehicle consists of a chassis, wheels, back lights and glass cabin with seats. All the vehicle parts were created in Creo Parametric, exported as STEP and converted into FBX file format.

The Unreal Engine was used in order to visualize motion of the vehicle, and to create and use the terrain by a relatively simply way. Motion of the vehicle is described with equations of motion, based on theory by Zuvich (2008), Monster (2003), Vlk (2001), Tautkus (2011), Švígler (2013), Short (2004), and solved by Runge-Kutta 4th order (RK4) method in Tick (deltaTime) method. The deltaTime parameter is determined by the UE4 system, and is a function of system's performance and load. The parameter is used in RK4 method.

To get distances of the vehicle over the terrain, in locations of wheels (Fig. 2, AB), we used a line tracing method available in UE4 API and forum by Rama (2016).

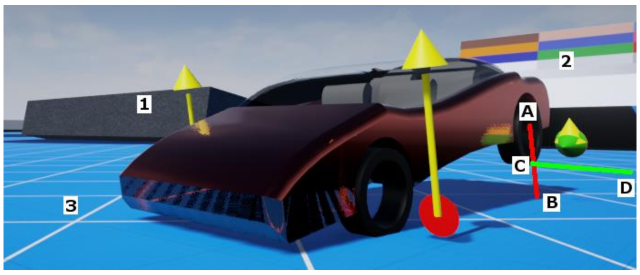

Figure 2. Contact with terrain after a jump, and line-tracing. 1, 2 – jump ramps, 3 – flat terrain, A – upper end of wheel spring-dumper unit, B – terrain, C – bottom of wheel, D – line tracing in horizontal direction. AB distance is used in RK4, CD – can be used in future to improve model's abilities.

These distances were used in the equations of motion of the vehicle. The main idea was to create the model as simple as possible, to decrease computer performance needs as much as possible, but on the other hand, we required an ability to visually evaluate a motion of the vehicle model. This is why we created a model that:

- allows motion of wheels in vertical direction only, considering the vehicle's frame,
- uses no tire and only one spring-dumper unit to simulate cushioning,
- supposes continuous contact of a wheel with the terrain (Fig.2), except the case when a spring's length should be higher than the free length of the spring,
- takes into the account only these forces (Fig. 3, 4): Fh – drive forces, Fb – brake forces, Da – aerodynamic drag force, G – gravity force, D' Alembert forces, Fp – spring forces, Fy – side forces on the wheels,
- has all the forces dependent on forces in the springs, which are, in this case, the same as normal forces,
- has the wheels which allow no rotation and the drive forces are applied directly on the wheels, without using an engine characteristic to simplify planned simulations and optimizations.

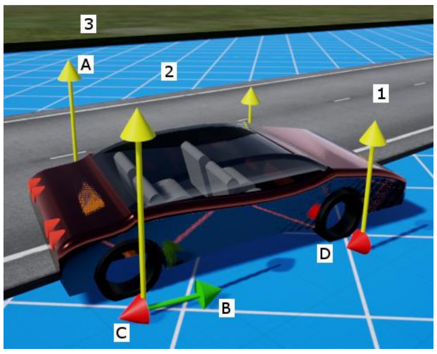

Figure 3. Visualized forces in Playback mode. 1 – test road, 2 – flat terrain, 3 – glass barrier around test facility, A – spring forces (Fp), B – drive forces (Fh), C – brake forces (Fb), D – rigid wheel.

We used following equations of motion, based on model on Fig. 4:

$$\sum F_x = 0: -G_x - D_a + (F_{hLF} - F_{bLF}).\cos\gamma_{LF} - F_{yLF}.\sin\gamma_{LF} + (F_{hRF} - F_{bRF}).\cos\gamma_{RF} - F_{yRF}.\sin\gamma_{RF} + (F_{hLR} - F_{bLR}).\cos\gamma_{LR} - F_{yLR}.\sin\gamma_{LR} + (F_{hRR} - F_{bRR}).\cos\gamma_{RR} - F_{yRR}.\sin\gamma_{RR} = m.a_x \qquad (1)$$

$$\sum F_y = 0: -G_y + (F_{hLF} - F_{bLF}).\sin\gamma_{LF} + F_{yLF}.\cos\gamma_{LF} + (F_{hRF} - F_{bRF}).\sin\gamma_{RF} + F_{yRF}.\cos\gamma_{RF} + (F_{hLR} - F_{bLR}).\sin\gamma_{LR} + F_{yLR}.\cos\gamma_{LR} + (F_{hRR} - F_{bRR}).\sin\gamma_{RR} + F_{yRR}.\cos\gamma_{RR} = m.a_y \qquad (2)$$

$$\sum F_z = 0: -G_z + F_{pLF} + F_{pRF} + F_{pLR} + F_{pRR} = m.a_z \qquad (3)$$

$$\sum M_x = 0: -b_L.(F_{pLF} + F_{pLR}) + b_p.(F_{pRF} + F_{pRR}) + (z_{offsetLF} + lSLF).(F_{hLF} - F_{bLF}).\sin\gamma_{LF} + F_{yLF}.(z_{offsetLF} + lSLF).\cos\gamma_{LF} + (z_{offsetLR} + lSLR).(F_{hLR} - F_{bLR}).\sin\gamma_{LR} + F_{yLR}.(z_{offsetLR} + lSLR).\cos\gamma_{LR} + (z_{offsetRF} + lSRF).(F_{hRF} - F_{bRF}).\sin\gamma_{RF} + F_{yRF}.(z_{offsetRF} + lSRF).\cos\gamma_{RF} + (z_{offsetRR} + lSRR).(F_{hRR} - F_{bRR}).\sin\gamma_{RR} + F_{yRR}.(z_{offsetRR} + lSRR).\cos\gamma_{RR} = -I_{xx}.\varepsilon_x \qquad (4)$$

$$\sum M_y = 0: b.(F_{pLF} + F_{pRF}) - c(F_{pLR} + F_{pRR}) + (z_{offsetLF} + lSLF).(F_{hLF} - F_{bLF}).\cos\gamma_{LF} - F_{yLF}.(z_{offsetLF} + lSLF).\sin(\gamma_{LF}) + (z_{offsetLR} + lSLR).(F_{hLR} - F_{bLR}).\cos\gamma_{LR} - F_{yLR}.(z_{offsetLR} + lSLR).\sin(\gamma_{LR}) + (z_{offsetRF} + lSRF).(F_{hRF} - F_{bRF}).\cos\gamma_{RF} - F_{yRF}.(z_{offsetRF} + lSRF).\sin(\gamma_{RF}) + (z_{offsetRR} + lSRR).(F_{hRR} - F_{bRR}).\cos\gamma_{RR} - F_{yRR}.(z_{offsetRR} + lSRR).\sin(\gamma_{RR}) + D_a.h_a = I_{yy}.\varepsilon_y \qquad (5)$$

$$\varepsilon_z = v_{x,local} * \sin(\gamma) / (c + b) \qquad (6)$$

Where:
Fp – spring force, Fh – drive force, Fb – brake force, Fy – side force, γ – steering wheel angle, b – distance between T (COG) and front axle in X axis, c - distance between T (COG) and rear axle in X axis, bL - distance between T (COG) and left wheels in Y axis, bP - distance between T (COG) and right wheels in Y axis, zoffset - distance between T (COG) and upper spring-damper mount point in Z axis, lS – instant spring length, Da - aerodynamic drag force, ε_z – angular acceleration around Z axis. All the relevant parameters have LF, RF, LR, RR suffix (Left Front, Right Front, Left Rear, Right Rear wheel).

The spring forces have been determined by equations:

$$F_{pLF} = v_{zLF}.k_{LF} + lTLF.c_{LF} \qquad (7)$$
$$F_{pRF} = v_{zRF}.k_{RF} + lTRF.c_{RF} \qquad (8)$$
$$F_{pLR} = v_{zLR}.k_{LR} + lTLR.c_{LR} \qquad (9)$$
$$F_{pRR} = v_{zRR}.k_{RR} + lTRR.c_{RR} \qquad (10)$$

Where:
v_z – velocities in Z axis, k – damping coefficients, lT – spring deformations, c – spring stiffness's

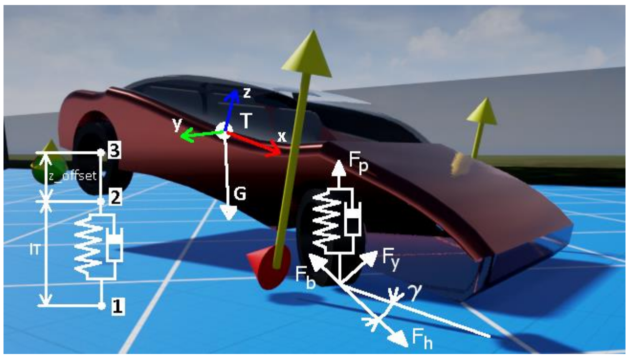

Figure 4. Mathematical model of the vehicle. 1 – center of a wheel, 2 – mounting point of a spring-damper unit, 3 – a point in COG plane XY, T – center of gravity (COG), Fp – spring force, Fb – brake force, Fh – drive force, Fy – side force, γ – steering wheel angle, G – gravity force

Velocities v_z at wheel locations were determined by equations:

$$v_{zLF} = b.\omega_y + b_L.\omega_x + v_{zT} \qquad (11)$$
$$v_{zRF} = b.\omega_y - b_P.\omega_x + v_{zT} \qquad (12)$$
$$v_{zLR} = -c.\omega_y - b_L.\omega_x + v_{zT} \qquad (13)$$
$$v_{zRR} = -c.\omega_y + b_P.\omega_x + v_{zT} \qquad (14)$$

Where:
ω_x – angular speed around x-axis (Fig.4), ω_y – angular speed around y-axis

The aerodynamic drag force Da has been determined by equation:

$$D_a = c_x \frac{1}{2}\rho.S_x.v_x^2 \qquad (15)$$

Where:
C_x – aerodynamic drag coefficient, ρ – air density, S_x – area of vehicle projection into a plane perpendicular on X axis, v_x – vehicle's velocity in X axis.

To run the simulations was used a powerful game computer with parameters listed in Tab. 1.

Table 1. Parameters of the computer we used

CPU	i7 – 4790 Haswell
RAM	16 GB DDR3
HDD 1	256 GB SSD
HDD 2	1000 GB
Video Card	NVidia GeForce GTX 970 4GB
Mainboard	ASUSTeK Maximus VII Hero
Steering Wheel	Thrustmaster Ferrari 458 Italy

3 RESULTS

The result of above described equations, implemented in UE4 C++ code is a game-like vehicle dynamics computer application that is not too much complicated and ready to be used in customized solutions. The vehicle's motion is controlled by the equations only, which can be verified by several ways. The application allows the user:

- Test the vehicle in free drive mode, by game-like way,

- Record the drive, including vehicle’s locations, rotations, forces, velocities, accelerations, and automatically output the result into file in CVS format, which can be easy imported into MS Excel,
- Evaluate the drive in Playback Mode, where recorded forces are displayed by arrows (see the previous figures),
- In Playback Mode the camera has geographic orientation (default view is North), and direction of view can be changed in real-time by steering wheel paddle shifters,
- Use an automatic speed control, based on simple regulator,
- Visual verification of the vehicle dynamic in virtual reality.

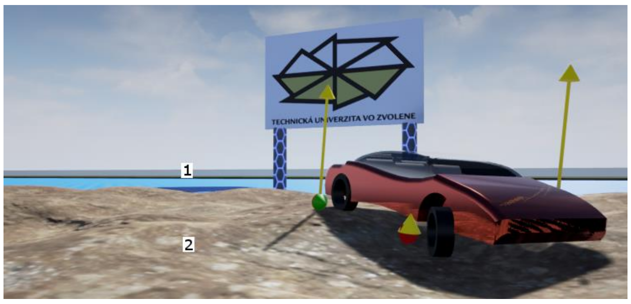

Figure 5. Rough terrain created in Creo Parametric. 1 – Flat terrain and barrier around the virtual test facility, 2 – the terrain.

The vehicle model is based on many parameters, which are independent on vehicle visualization, except the wheels locations, so they can be changed at any time, including real-time, during simulation. The terrain can be created and modified in Unreal Editor, 3ds Max or any CAD application (Fig. 5). It can be also bought at Unreal Market (Epic Games 2016).

4 DISCUSSION

In the Introduction we wrote about deltaTime parameter as input into the RK4 equations. A default value is 120 FPS (frames per second) which produces 1/120s deltaTime. This value seems to be acceptable, but if the value is decreased to 90 FPS, the vehicle seems to be a little unstable, which means that it sometimes produces a vibrations. However that is not a problem caused by low FPS only, but it can be also a result of the mathematical model, which is quite simple and containing low amount of damping elements. The result is that values of forces can change significantly in very short time. We did not solve this problem in detail, since the results seem to be acceptable for planned simulations and optimizations based on recorded data. Another authors and their results are, in general, scientifically (Chrono 2016, Švígler 2013) or game focused (Epic Games 2016, Monster 2003, Zuvich 2008). Equations of motion of game focused authors use higher separation of the vehicle’s longitudinal and lateral motions, simplifications, and results may not be accurate. The scientifically focused authors use more detailed models with high precision outputs, however the models have too much time consumption, unsuitable for real-time mode of the simulation. The presented model is placed between these limits.

It uses line trace functions by Rama (2016) that sense the terrain in vertical direction. It is used to compute the spring forces. However this functionality can be used also to sense the terrain in horizontal direction (Fig.2, CD). It could enable the vehicle to react on e.g. side slip and subsequent possible additional roll moment. This functionality we plan to implement in future.

The application is wrote in Visual Studio C++, which is powerful but it can be, sometimes, a little uncomfortable computer language. While other game engines (e.g. Unity3D) can use C# language, UE4 does not. A good new message is that Haxe language can also be used through Unreal.hx plugin. It is a free plugin for UE4 that enables developers to write code in Haxe, a high-level, type-safe language. The plugin compiles directly to C++ for high runtime performance, offering full access to the entire UE4 API and more. Haxe is an open source toolkit based on a modern, high level, strictly typed programming language, a cross-compiler, a complete cross-platform standard library and ways to access each platform's native capabilities.

5 CONCLUSIONS

The vehicle dynamical simulator was created in Unreal Engine game development environment. It is based strictly on equations of motion, and solved by Runge-Kutta 4th order algorithm in Tick() method of Unreal Engine. The vehicle is controlled by a keyboard and a racing steering wheel. The simulator has several modes, which allow it mainly to perform a free ride in game-like style, record such a ride, and output the ride into a file. All the virtual world’s parts can be created or modified in usual applications for 3D modelling, like Creo Parametric, 3ds Max, and also Unreal Editor.

The result is that there is a vehicle dynamics tool, which can be used in research and education of dynamics, and presentation of virtual reality. Thanks to UE4, the visual appearance of the application is very real and trustworthy.

ACKNOWLEDGEMENT

This research was supported by Volkswagen Slovakia Foundation (http://www.nadacia-volkswagen.sk).

Nadácia Volkswagen Slovakia

and Ministry of Education, Science, Research and Sport of the Slovak Republic (research project VEGA 1/0676/14).

REFERENCES

[Chrono 2016] Project Chrono. [cit. 9/5/2016] Available from: http://www.chronoengine.info

[Epic Games 2016] Unreal Engine. [cit. 5/20/2016] Available from: https://www.unrealengine.com/

[Epic Games 2016] Epic Games, Inc. Unreal Marketplace. [cit. 5/22/2016] Available from: https://www.unrealengine.com/marketplace

[Monster 2003] Monster, M. Car Physics for Games. 2003. [cit. 6/2/2016] Available from: http://www.asawicki.info/Mirror/Car%20Physics%20for%20Games/Car%20Physics%20for%20Games.html

[Rama 2016] Rama. Trace Functions. [cit. 5/22/2016] Available from: https://wiki.unrealengine.com/Trace_Functions

[Short 2004] Short, M. – Pont, M.J. – Huang, Q. Simulation of Vehicle Longitudinal Dynamics. Embedded Systems Laboratory, University of Leicester. 2004. [cit. 5/22/2016] Available from: http://www.le.ac.uk/eg/embedded/pdf/ESL04-01.pdf

[Svigler 2013] Svigler, J. Drive vehicles. The project CZ. 1.07/2.2.00/15.0383 Innovation of the field of Transport and handling equipment with a view to the needs of the labour market. UWB in Pilsen. 2013. (in Czech) [cit. 5/22/2016] Available from: http://www.kme.zcu.cz/download/predmety/468-mechanika-vozidel.pdf

[Tautkus 2011] Tautkus, A. Longitudinal and lateral dynamics. Powering the Future With Zero Emission and Human Powered Vehicles – Terrassa 2011 [cit. 5/22/2016] Available from:

http://www.ip-zev.gr/files/teaching/T3-4_Lateral_longitudinal_dynamics.pdf
[Vlk 2001] Vlk, F. Dynamics of motor vehicles. Brno: Publishing VLK, 2001. ISBN 80-238-5273-6 (in Czech)
[Zuvich 2008] Zuvich, T. Vehicle Dynamics for Racing Games. [cit. 6/2/2016] Available from: https://transporter-game.googlecode.com/files/VehicleDynamicsForRacingGames.pdf

CONTACTS

Ing. Jaroslav Matej, Ph.D.
Technical University in Zvolen
Faculty of Environmental and Manufacturing Technology
Department of Mechanics, Mechanical Engineering and Design
T. G. Masaryka 24, Zvolen, 960 53, Slovak Republic
Tel.: +421-45-5206555, e-mai: jaroslav.matej@tuzvo.sk
http://www.tuzvo.sk/matej