Adding VR to CARLA – Technical Guide

Introduction

As part of a project for the University of Applied Science Ruhr West, it was examined how the driving simulator CARLA [1] could be extended by a virtual reality mode.
The results and collected experiences are summarized in this document. Furthermore, a detailed step-by-step guide on how to set up and use this VR mode is provided.

This document refers to CARLA version 0.9.13, which was released in November 2021 [2].

It is important to mention in advance that CARLA is not designed to be used in VR, but is only intended as a simulation platform with desktop output.
The environments and objects are therefore not optimized purely for performance, which makes them difficult to use in VR. A few steps are taken in advance to increase the performance in VR a bit. Nevertheless, CARLA Simulator will not run completely smoothly in VR even with above-average hardware.

VR Mode – Preparation

Let's start with the preparation for including the VR mode. The whole process is done in the Unreal Editor. So first, CARLA should be started in the *x64 Native Tools Command Prompt for VS 2019* via the ***make launch*** command. This opens CARLA in the Unreal Editor and makes it available for further editing.

Environment

CARLA offers different environments and maps, which describe different scenarios and situations. In addition, CARLA distinguishes between normal and so-called HD maps, which are visually more appealing but require significantly more processing power for the GPU.

Therefore, it is recommended to switch to the map Town02 in the Content Browser of the Unreal Editor under *Content/Carla/Maps*, since it is the smallest and therefore most suitable for the VR mode (see Figure 1).

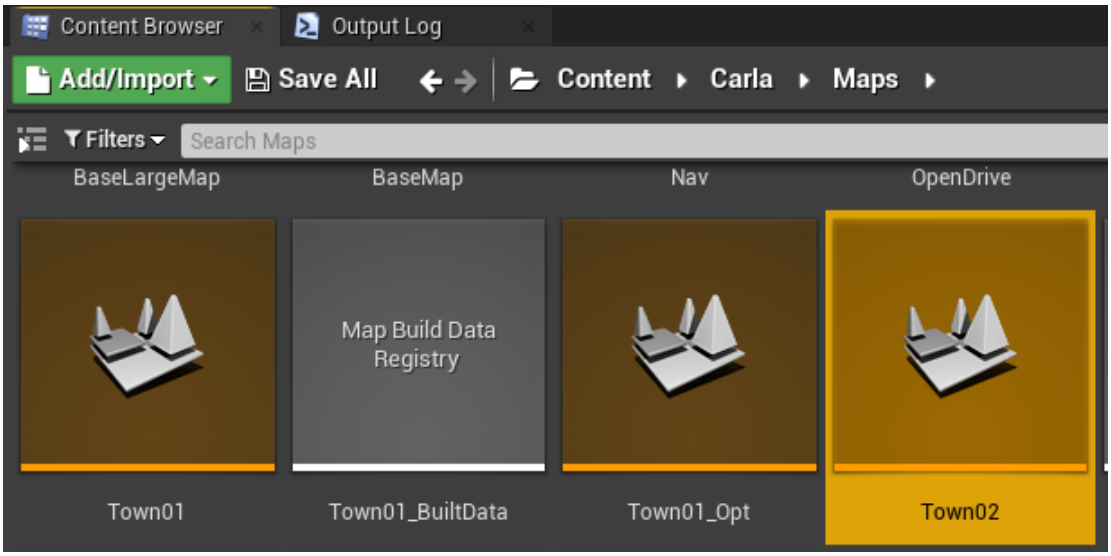

Figure 1 - Town02 listed in the Content Browser of Unreal Editor

To avoid having to change the map after each restart of CARLA, the default map can be adjusted in the settings. To do this, change the default maps to *Town02* in the Unreal Editor under *Edit/Project Settings* in the menu item *Project/Maps & Modes* (see Figure 2).

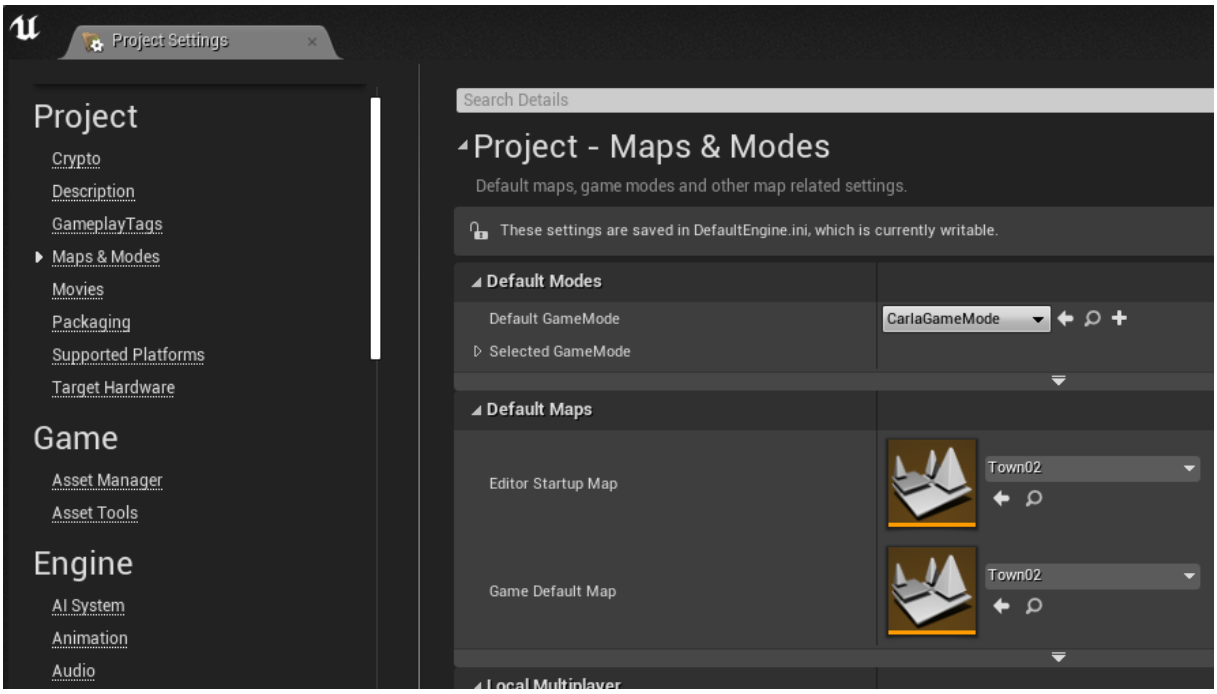

Figure 2 – Editing the default map in Project Settings

Another option to improve the performance in VR would be to choose one of CARLA's layered maps, marked by the suffix *_OPT,* where different parts of the environment can be toggled off.

SteamVR

To ensure compatibility with VR headsets from different manufacturers, *SteamVR* is used as an interface between Unreal and the HMD. To include *SteamVR* in the Unreal Editor, the *Plugins* option is selected from the menu bar under the *Settings* item (see Figure 3).

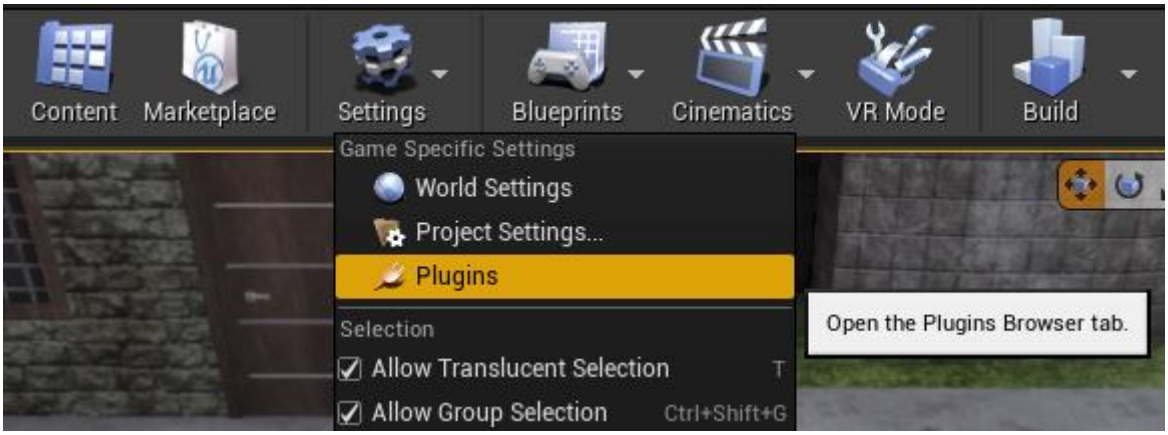

Figure 3 – Starting Plugin Browser

A new plugin browser will open. Using the search function or the *Virtual Reality* category on the left side, the *SteamVR* plugin can be searched for and integrated via the *Enabled* checkbox (see Figure 4).

Figure 4 – Enabling the SteamVR Plugin

Alternatively, Google VR, Oculus VR, or OpenXR are also available. From now on, *SteamVR* is automatically opened when the Unreal Editor is started and, if available, the connected VR headset is automatically detected.
It is recommended to close the Unreal Editor once after integrating the VR plugin and to restart it via the console with the command *make launch*!

Performance-Mode Unreal Editor

While running the VR mode, the Unreal Editor remains open and normally runs in the foreground as the main application. Nevertheless, it may happen that other applications move Unreal into the foreground.
In this case, the CPU load is heavily slowed down by Unreal Editor itself to conserve resources. However, since this has a strong impact on the performance of the CARLA simulator, it is recommended to disable this option.
To do so, uncheck the option ***Use Less CPU when in Background*** in the Unreal Editor under *Edit\Editor Preference* in the *Performance* menu (see Figure 5).

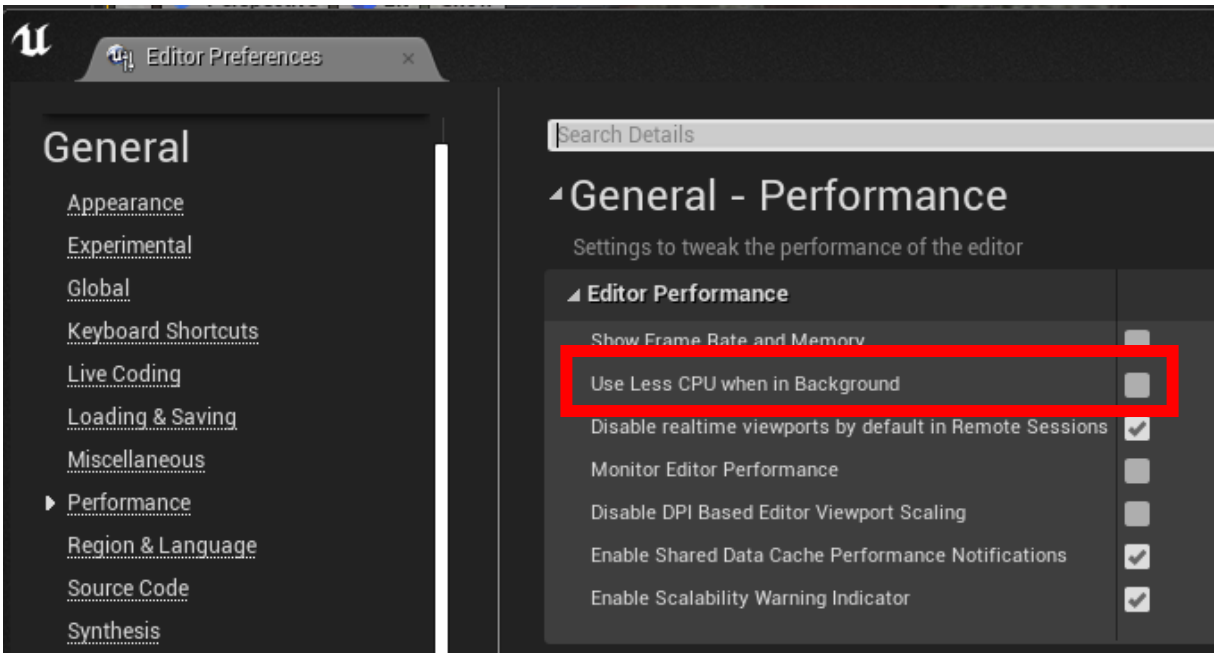

Figure 5 – Performance Option in the Editor Preferences

VR Mode – Vehicle and camera

Technically, all requirements for including a VR mode in Carla are now fulfilled. Now the vehicle and camera are placed in the map in the Unreal Editor, which will be used by the player when starting in VR mode.

Vehicle

CARLA offers an extensive selection of different vehicles, which have been modeled in quite a detailed way. Unfortunately, this does not apply to the interior of most vehicles, which is mostly rendered in a rudimentary and untextured way.

An exception is the *Mercedes CCC* vehicle, which has textures for seats, steering wheel, and dashboard. Therefore, it is one of the most suitable for being controlled from the first-person perspective in VR.

The vehicle model can be found in the Content Browser under *Content/Carla/Blueprints/Vehicles/MercedesCCC*. Since the blueprint of this vehicle will be modified in the following, it makes sense to create a copy of the vehicle files. To do this, right-click on the vehicle folder in the Content Browser and select the *Show in Explorer* option.

The file explorer opens, in which the entire vehicle folder can be copied and, for example, provided with the suffix *_VR*. The new folder also appears in the Unreal Editor. For a better assignment, the blueprint should also be renamed to the folder name (see Figure 6).

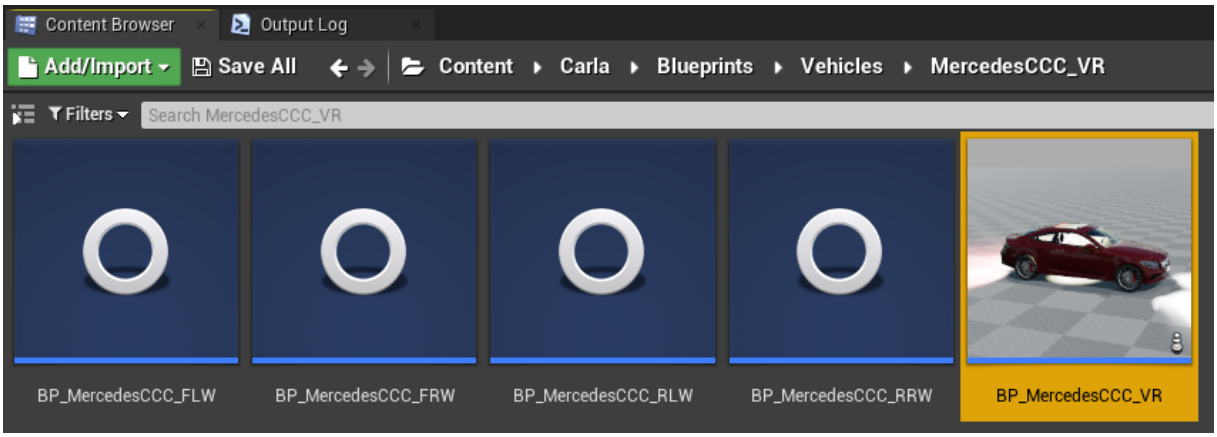

Figure 6 – Copying the vehicle folder

Afterward, the Blueprint can simply be placed in the environment by dragging and dropping and will now be listed in the upper right corner of the *World Outliner* window next to all other objects in the world.

To quickly find the vehicle again later, a folder named VR can be created in the *World Outliner* window, into which the vehicle (BP_MercedesCCC_VR) can also be moved using drag and drop (see Figure 7).

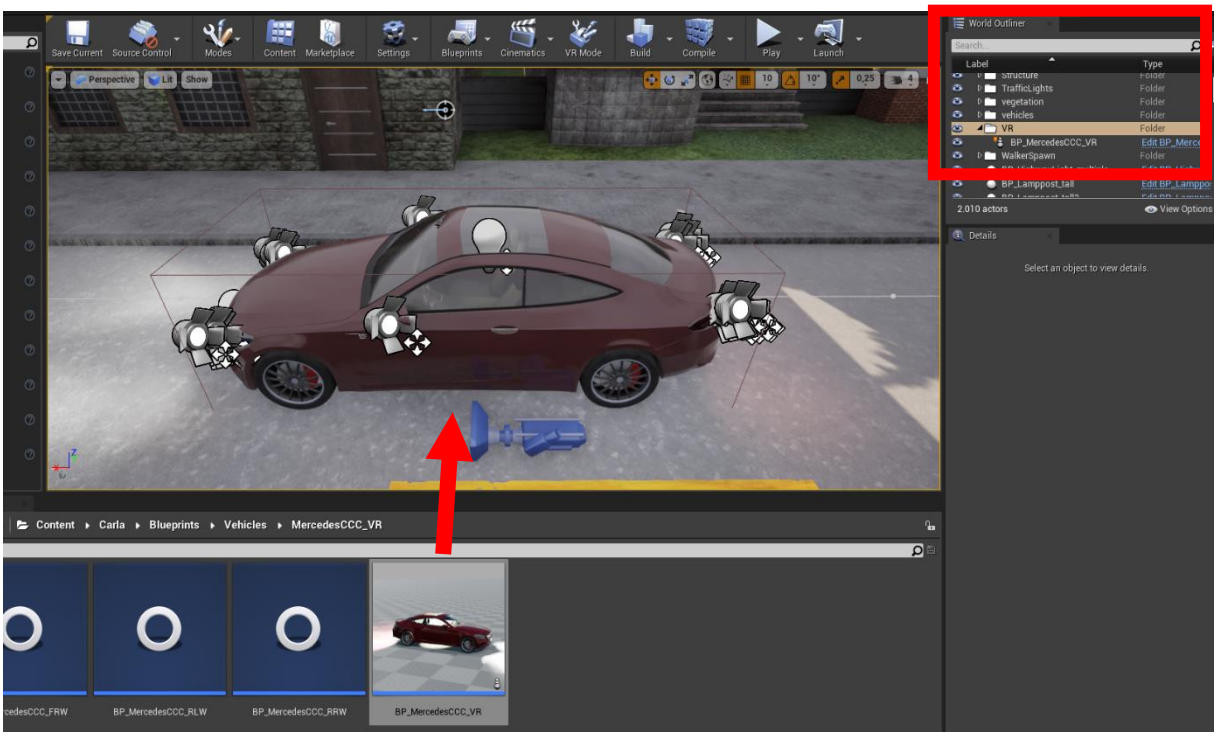

Figure 7 – Placing the vehicle inside the map

Controlling the vehicle

In order to allow the player to control the vehicle, it should now automatically be possessed as a pawn when the simulation is started.
To do this, the vehicle is selected in the game world or the *world outliner,* and the ***Auto Possess Player*** property is changed from *Disabled* to ***Player 0*** in the properties in the *Pawn* category (see Figure 8).

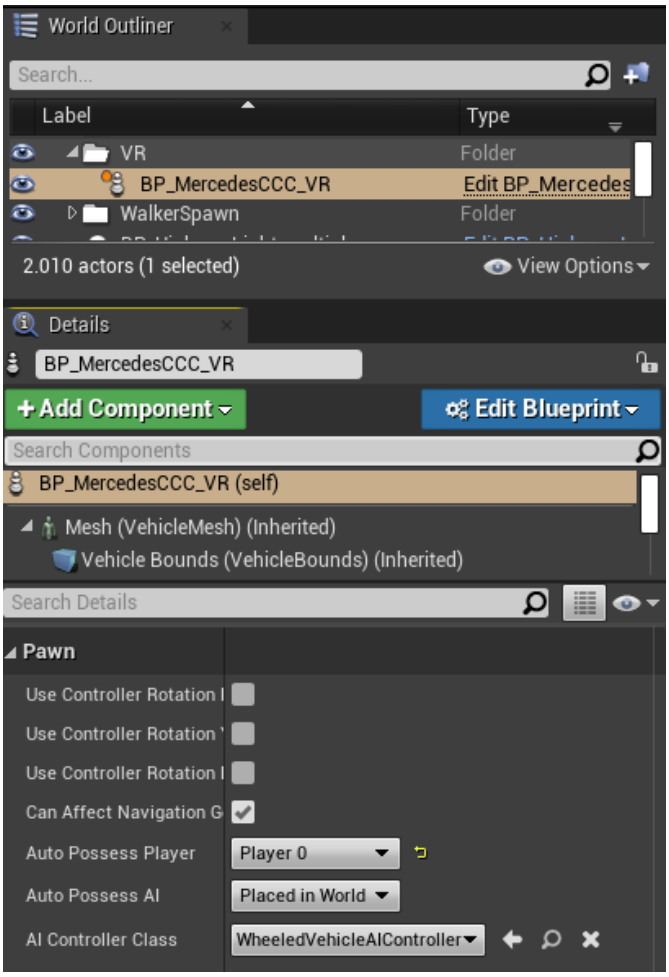

Figure 8 – Enabling Auto Possess for the player to automatically control the vehicle

By default, however, the vehicle does not respond to external input from the player. Manual control of the vehicle must therefore also be added.
First, the control commands must be linked to the input device. To do this, the ***Input*** item of the *Engine* category is selected via *Edit/Project Settings*.

Here, pre-defined and user-defined actions are linked to key combinations. Default functions are already created for the movement of a vehicle. These can be customized as desired depending on the preference of the player. A possible assignment to control the vehicle via keyboard can be seen in Figure 9.

Note: The integration of a gamepad is also possible. Unfortunately, other hardware such as a steering wheel or a joystick cannot be natively integrated. It might be possible that these input devices could emulate a gamepad, but this remains to be tested.

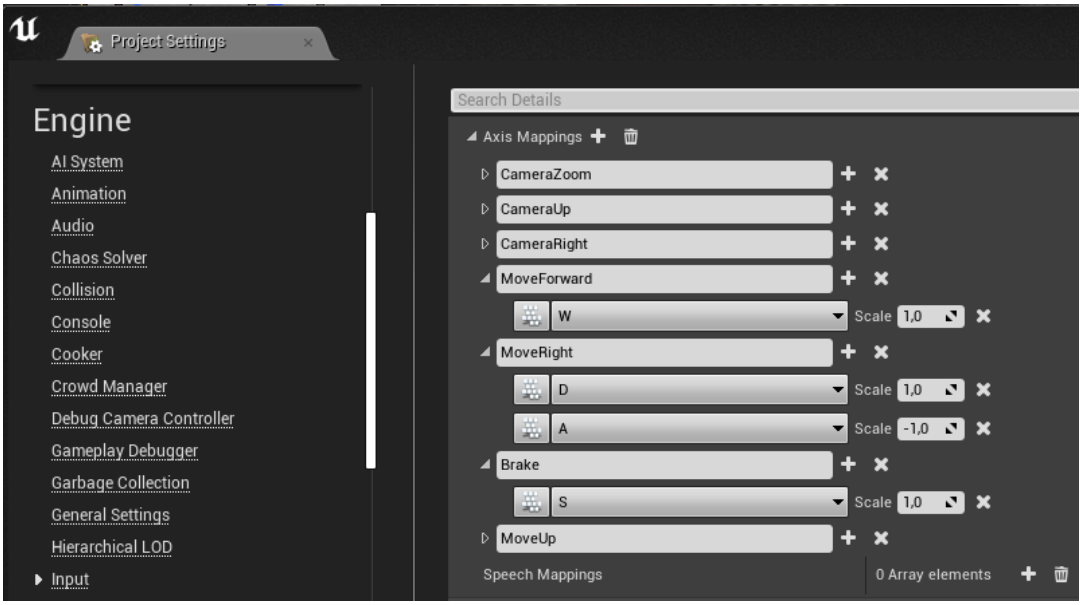

Figure 9 – Key binding for controlling the vehicle

In order for the vehicle to react to the actions linked to the key combinations, these must be integrated into the vehicle's blueprint. To do this, open the blueprint in the *Content Browser* by double-clicking on it.

In the *Event Graph*, the action events of the keys must now be linked to the input interfaces of the vehicle, as shown in Figure 10.
Right-click to search for the corresponding blocks using the search function and then add them to the *event graph*.

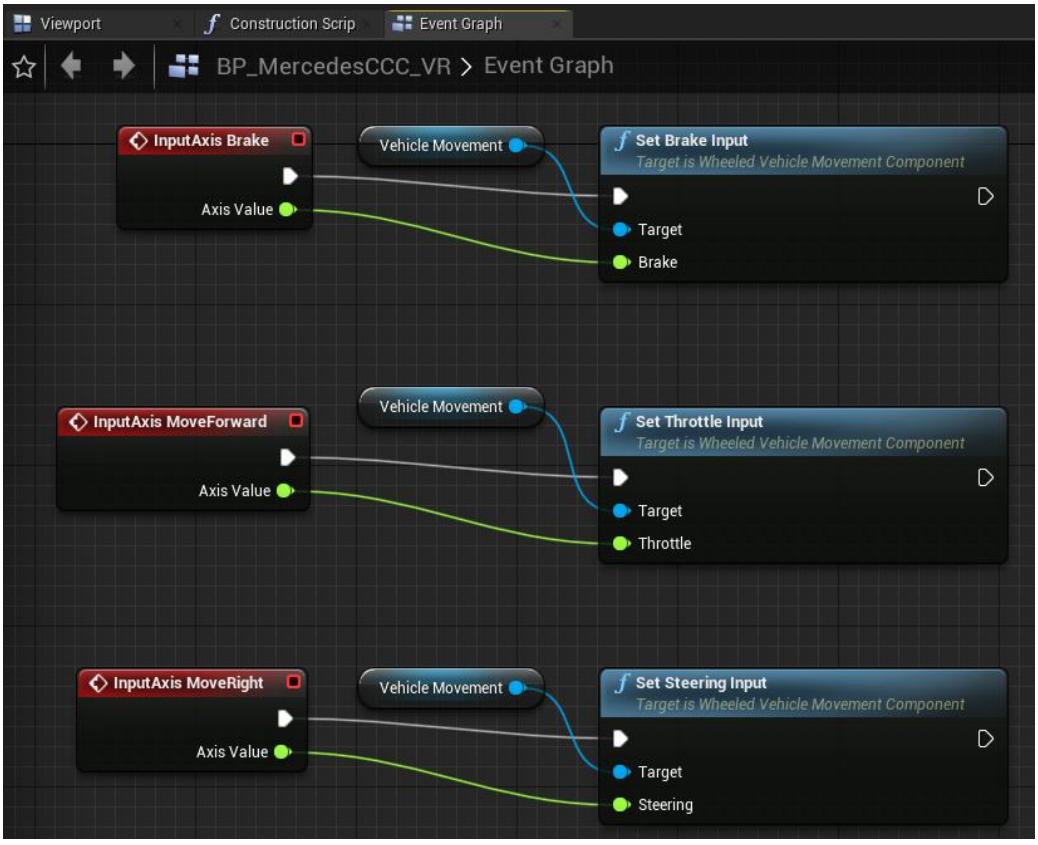

Figure 10 – Implementing car control inside the vehicles blueprint

It is recommended to search for the individual actions (*Brake, MoveForward, MoveRight*) and the individual inputs (*Brake, Throttle, Steering*).
It is important to make sure that the correct events and inputs are selected from a large number of options.

For the actions, it is important that the option of the ***Axis Events*** category is selected (see Figure 11).

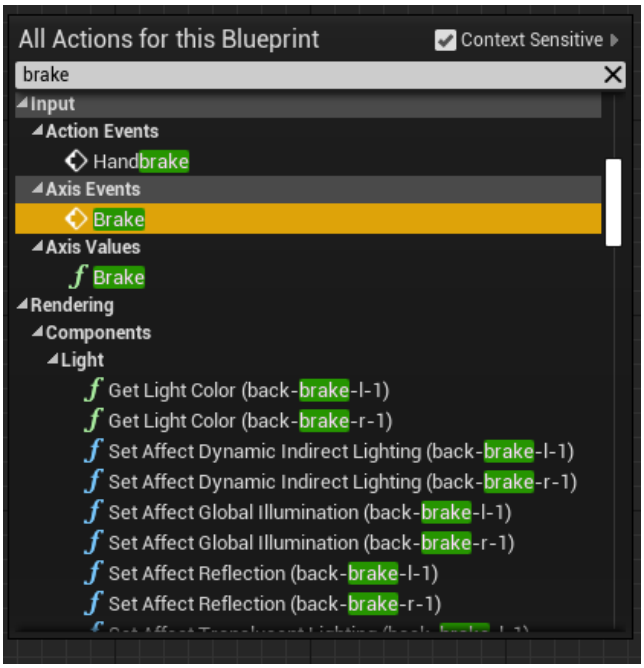

Figure 11 – Axis Event for the command "Brake"

The input values of the vehicle appear in two different categories. It is important to use the module of the ***Wheeled Vehicle Movement*** category (see Figure 12).

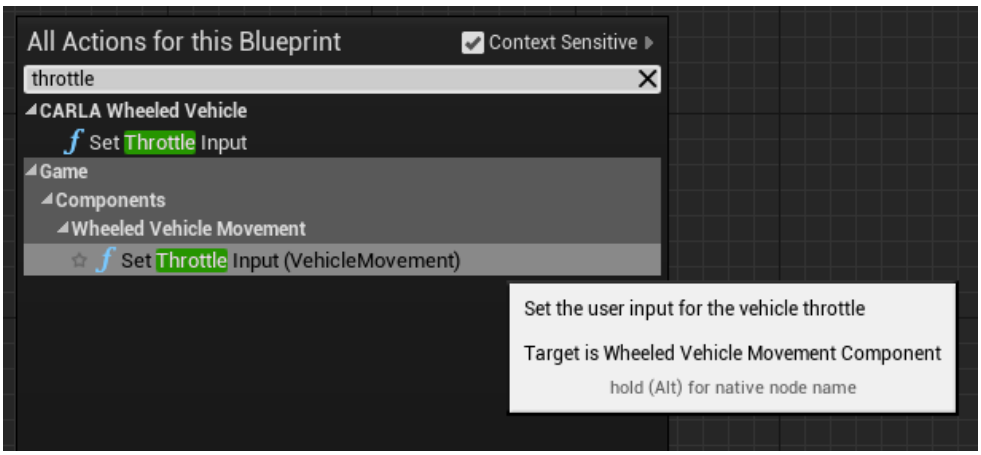

Figure 12 – Value Input from the category Wheeled Vehicle Movement

Camera

Now the camera needed for the VR mode is created and placed in the environment. The procedure here is similar to the normal standard procedure to integrate a VR camera into an Unreal project, see [3].

The difference is, however, that the camera is not defined as a new blueprint of its own, but, like the control described above, is integrated into the blueprint of the vehicle. Thus the camera is coupled to the vehicle and moves as a part of the vehicle through the game world.

To do this, the vehicle blueprint is opened in the *Content Browser* by double-clicking on it. Following the instructions in [3], a new ***scene*** with the name *CameraRoot* is added in the Components window, followed by a new ***camera*** with the name *VRCamera* (see Figure 13).

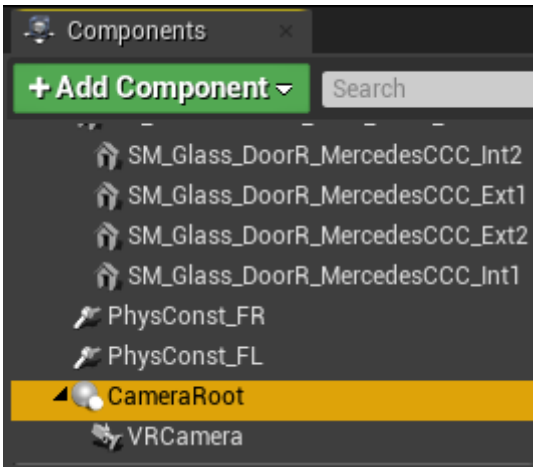

Figure 13 – Adding a VR camera to the vehicles blueprint

In the *EventGraph*, the block ***Set Tracking Origin*** is now added by right-clicking and connected to the already existing block Event *BeginPlay* (see Figure 14).
This way, the camera is synchronized and coupled with the position of our VR headset upon initialization.

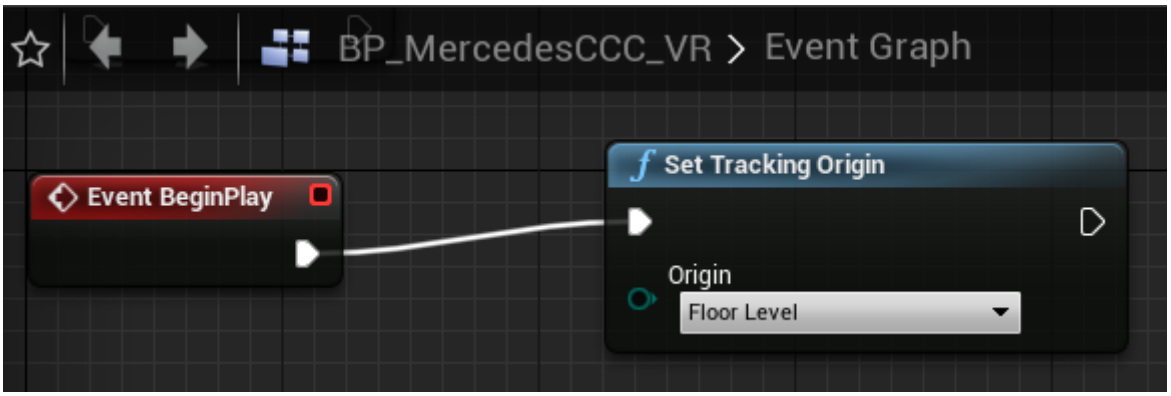

Figure 14 – Connecting HMD tracking with the camera

Finally, the position of the camera has to be adjusted. To do this, select the placed vehicle in the world or in the *World Outliner* window and select the ***VRCamera*** in the *Details* window.
The properties of the camera can now be edited, including the position (*Transform*) relative to the vehicle (see Figure 15).

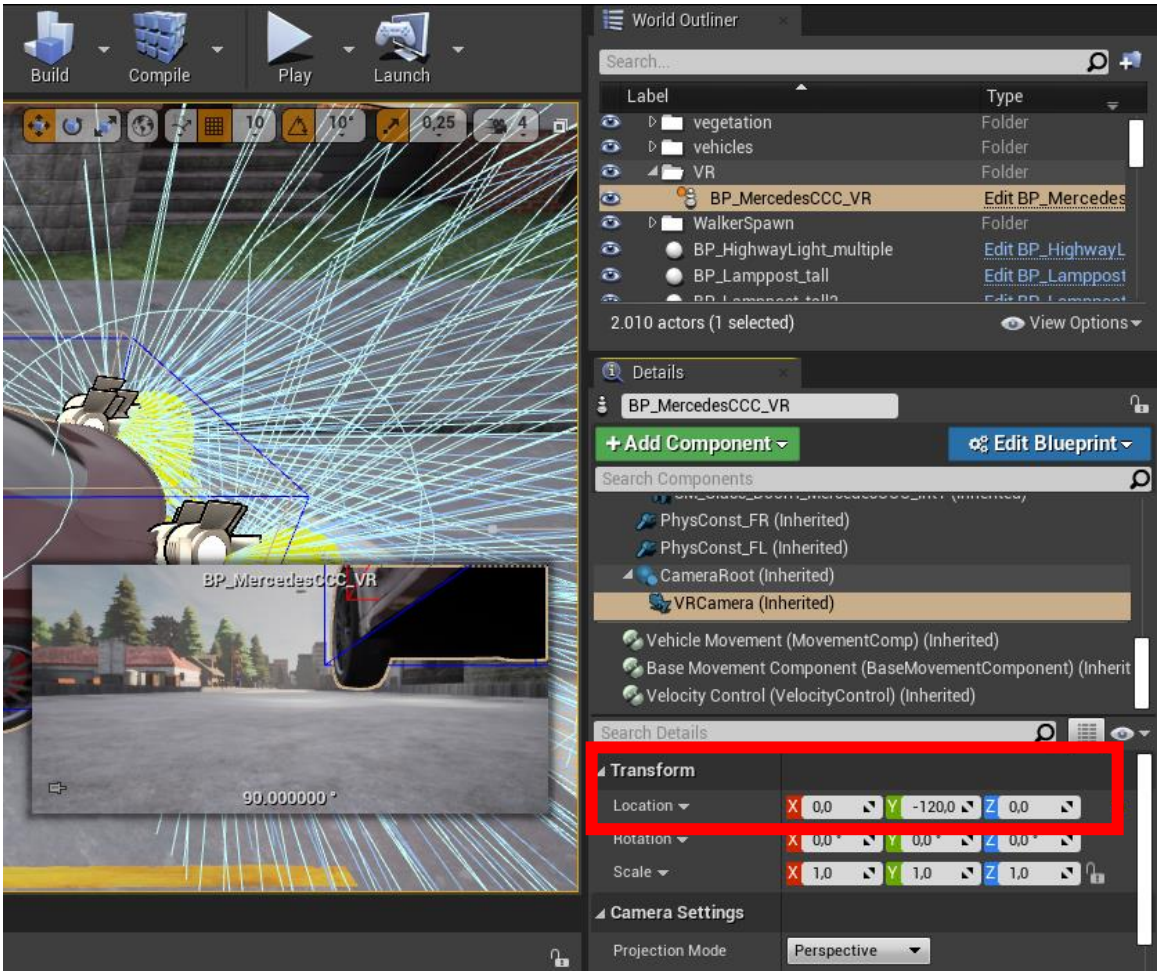

Figure 15 – Setting the camera position

Here it is important to note that the Z coordinate is set to the value 0 so that the height of the camera matches the height of the VR headset in the real world.
The X and Y coordinates must be set accordingly so that the desired position of the VR headset corresponds with the driver's seat position in the simulator.
Since this depends on the calibration of the VR headset and the position in the respective application, the correct position must be adjusted individually.

VR Modus – Play

If all steps have been executed as described, the integration of the VR components into the game world of CARLA Simulator is now completed.

The simulation can be started via the *Play* option in the upper menu bar.
By default, the *Selected Viewport* option is selected, which outputs the simulation to the connected desktop.

To use the VR headset as an output source, the ***VR Preview*** option must be selected via the drop-down menu next to the Play button (see Figure 16).

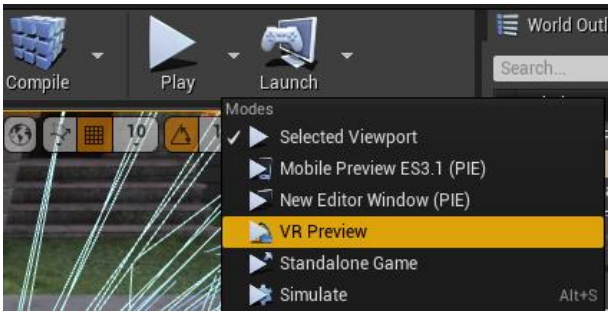

Figure 16 – Starting CARLA as VR Preview

Final Thoughts

As mentioned earlier, the performance of the CARLA simulator in VR is far from an optimal VR experience.

One way to improve performance could be to create a custom map that uses significantly fewer assets than the maps included in CARLA.

In addition, a revision of the lighting would be another factor that could help improve performance. The standard maps contain many individual light sources in the form of street lamps and traffic lights, which could be reduced in number or removed completely.

References

[1] “CARLA Website,” [Online]. Available: https://carla.org/. [Accessed 02 August 2022].

[2] “CARLA 0.9.13 Release,” [Online]. Available: https://carla.org/2021/11/16/release-0.9.13/. [Accessed 02 August 2022].

[3] “Set Up a Standing Camera for SteamVR,” [Online]. Available: https://docs.unrealengine.com/4.26/en-US/SharingAndReleasing/XRDevelopment/VR/SteamVR/HowTo/StandingCamera/. [Accessed 08 August 2022].