Karlsruhe Institute of Technology

FAKULTÄT FÜR ELEKTROTECHNIK UND
INFORMATIONSTECHNIK

A Guide for future VR development at the IPE with the Unreal Engine

Internship report
of

Jonas Teufel

am Institut für Prozessdatenverarbeitung
und Elektronik

Supervisor: Andreas Kopmann

Bearbeitungszeit: 01.04.2015 – 30.09.2015

Contents

List of Figures

1. Introduction

1.1 The premise

This document aims to create an entry point to unreal engine VR development. Hereby the focus is not necessarily on creating a full blown game mechanic, but rather on projects, which would be both achievable and beneficial in a scientific context.
Although this document aims to create sort of a general overview, it is being written specifically for the *Institute of data processing*(IPE) at the KIT. The IPE has already conducted multiple VR projects. I will use them as an example to explain, what purpose a VR project could have in the scientific context:

- One project used the CAD models for the *KATRIN*[1] experiment and imported those as models into a VR setting, which can be used present the experiment to a broader public, which is not able to visit the restricted area for example.
- Part of the research at the KIT campus north is the recording of x ray tomographies. Using this method researchers were able to create detailed 3D models of insects within fossils. There has been an effort to import these insect models into a VR setting. The capabilities of the unreal engine could help to animate the movement and behaviour of these long extinct species for example. An audience would be able to experience these animals first hand.

1.2 document overview

This document aims to achieve several things:

entry point This document should provide a basic overview of the first steps to be taken, when starting VR development with the unreal engine, such as the installation of required programs and the setup of the needed hardware.

choices The success of a VR project is significantly determined, by the planning process or the general "vision" for the final project. There are some game-specific and also VR-specific design choices, that have to be well thought about when determining the initial goal of the project.

tutorial I will attempt to provide tutorial-like sections for chosen topics, which will be helpful for developing a first application scaffold.

ressources The unreal engine is too big of a topic to be explained well in a single document. Thus, this document will not provide a "real" tutorial. Instead I will reference some literature and other resources, that have done a good job of explaining certain aspects of the topic.

[1] `https://www.katrin.kit.edu/index.php`

2. First steps

2.1 Installing the development environment

2.1.1 Unreal engine requirements

The *minimal requirements*[1] for installing the unreal engine are quite low, as can be seen in figure 2.1 [Unr]. These requirements make it seem as if development with the unreal engine was actually possible on such a machine.
I have tried to install the development environment on a computer, which just barely satisfied these requirements. From the experience I can say, that while the program actually starts, it is completely unusable. The graphics card and the CPU will be absolutely overwhelmed.

Recommended Hardware

Operating System	Windows 10 64-bit
Processor	Quad-core Intel or AMD, 2.5 GHz or faster
Memory	8 GB RAM
Video Card/DirectX Version	DirectX 11 or DirectX 12 compatible graphics card

Figure 2.1: Minimal system requirements for the unreal engine

On the far bottom of the same requirements page, epic games lists the specs of the PC's, which themselves use for their game development, as shown in figure 2.2.
These should be considered as the *actual* minimal requirements for a smooth development experience.

- Windows 10 64-bit
- 64 GB RAM
- 256 GB SSD (OS Drive)
- 2 TB SSD (Data Drive)
- NVIDIA GeForce GTX 970
- Xoreax Incredibuild (Dev Tools Package)
- Six-Core Xeon E5-2643 @ 3.4GHz

Figure 2.2: Reasonable system requirements for the unreal engine

In the end, a machine with the above or comparable hardware specifications *should* be used to guarantee a lag free experience of the unreal editor software.

[1] `https://docs.unrealengine.com/en-US/GettingStarted/RecommendedSpecifications/index.html`

2.2 Installing the unreal engine

The installation of the unreal engine is managed by installing the *Epic Games Launcher*. This launcher is used to manage and launch games, developed by Epic Games. But it can also be used to download the Unreal Engine from. Using the Epic Games Launcher provides the benefit, that it is easily possible to download multiple different versions of the engine. All these versions can be installed and managed on the same computer at once.
The installation of the unreal engine is rather well documented on the following web page:

`https://docs.unrealengine.com/en-US/GettingStarted/Installation/index.html`

2.3 Additional programs

When creating a game, a developer has to deal with many different things:

- Creating the 3D models for the characters and the environment
- Creating surface material textures for the 3D models
- Creating animation sequences

Just to name a few.
While all these steps can be done within the Unreal Editor itself, it is strongly discouraged to do so. All of these processes have their own, specialized application programs. These programs provide a superior environment with better functionality.
The problem is, that most of these programs require a license fee.

2.4 Installing Blender

Blender[Fou] is an open-source program, which is mainly used for 3D modeling and VFX animations sequences. But it can also be used for the other operations. Even though for some steps Blender is also not the best tool to be used, it is still better than using the built in functionality of the Unreal Engine.

Blender is installed by downloading an installer from the following page:

`https://www.blender.org/download/`

Running the installer program should install the blender program, which will be executable through a desktop icon.

3. Design Choices

3.1 The scope of the project

3.1.1 basic considerations

Before starting to work on a project, several things have to be outlined at first. One of the major questions leading up to a VR project will be "What will it have to accomplish?" or "Why are we doing this?". A likely answer for the case of a VR project in a scientific context would be to showcase some sort of scientific project or scientific paper to a broader public. The VR space is used as a medium to create a superficial understanding within those people, which cannot grasp the concrete details.
The next question should be about the scope of the project "What specific aspects of our work would we like to showcase?". This includes the task of figuring out, which features the VR presentation has to have, but more importantly which features it *does not* have to implement. It is important to understand, that it is a *very hard and time consuming* task to design and implement a *good* VR experience. Thats why a reasonably step during the development is to explicitly think about, where complexity can be reduced as much as possible. This goes for all aspects of a game. Possible approaches would be to think about at which points it would not hurt the aesthetics to use easier textures or how to strategically limit the playable area.

3.1.2 example 1

To further explain these considerations, I will give a fictional example:
Biologists have discovered a fascinating new method of transportation within an exotic tadpole species. Since their movement cannot accurately be described with images alone, they want to create a VR game to showcase their findings to the public. More concretely they define the goal of the VR presentation as *Explaining the steps involved in the movement of the tadpoles.*
During an initial brainstorming it is quickly clear, that the setting of the game should be underwater, where the user can observe the tadpoles natural motion in 3D. A lot of ideas are gathered, for example the following ones:

- The playable area could be designed like the bottom of the ocean, or the inside if a pond or aquarium
- There could be many tadpoles everywhere, swimming in different directions
- A user interface could be used to switch between an external view of the tadpoles to an internal one, with all the muscles.
- The user will be sitting in a submarine capsule which can be controlled to swim within the world

While all these ideas provide great inspiration and would undoubtably make a nice VR experience, it is important to keep in mind, that usually a lot of people invest a lot of time into developing nice-looking, smooth game experiences. Since creating a presentation for the public usually is a rather low priority within a scientific project, the amount of time and man power, which can be spent on the VR presentation is very limited.
Due to the lack of time the team stripped down the features of the final brainstorming to this final description of the game: The player will find itself standing in a bright white room. The player can navigate the room by walking into all directions. In the center of the room there is a circular platform. In front of the platform is a single button menu, which can be used with the motion controllers. The buttons are "spawn tadpole" and "description". Upon pressing the "spawn tadpole" button a tadpole will appear floating stationary above the platform, performing its signature motion. Upon pressing description an audio description of the tadpoles movement will start to play. The player can investigate the motion from all different sides, by moving around the platform, but he cannot leave the room.
This is a good example of tightening the scope of a project to save in production effort. A lot of features were sacrificed for the sake of simplicity, yet the original goal is still achieved.

3.2 VR specific design choices

When embracing a VR project there are some elements of game design especially relevant to the platform being virtual reality specifically.
In the following sections I will be summarizing key aspects, which are listed in the following book:

Unreal for Mobile and Standalone VR[Hil19]

For further details on the topics, read the corresponding chapters of the book.

3.2.1 Why VR?

This is usually the first and most important question that has to be asked before starting a VR project: *Do we actually need VR for this.* We will look at the example of creating a virtual reality showcase of some scientific experiment or paper.
The aim of such a project is of course to create a visually appealing presentation for an audience. But there are multiple different ways of presenting a complex topic to someone: Beginning with just static illustrations on a placard, a narrated slideshow, a simple animation without interactions... And all of these can probably be realized with less effort, than a VR experience would require. Every medium has its unique advantages. In that sense, it would only be worth the additional effort if the given presentation would benefit greatly from the advantages of VR. These advantages are:

- An unrivaled sense of 3 dimensional *depth* and *size*
- Interactivity

So the questions leading up to whether or not VR is to be used as the medium of presentation are:

- Does our project need to display size and depth differences in 3 dimensions?
- Does our project benefit from incorporating interactivity

If the scope for example is to illustrate a (mostly) 2 dimensional process without any kind of user interaction, VR might not be the right choice.
Going even further VR might even be a very *wrong* choice in some situations: Consider the

following example: A team has approximately X hours to spent on creating a presentation of their topic. During this time they could either create an above average presentation with traditional media such as placards and animations *or* a below average VR presentation. Chances are high, that the audience will be put off by the fact that the VR is "bad" quality. On the other hand, the audience might have enjoyed the good traditional media, even though it is *"just"* 2 dimensional.

Knowing this, the question *Why VR?* provides us with good hints of how to create a successful VR experience: Designing a simple, yet highly interactive application structure for the user. As well as creating an environment which makes use of close range *and* medium range objects to nicely incorporate a feeling of 3 dimensional scale and depth.

3.2.2 The method of locomotion

The method of locomotion has been one of the most important aspects of VR game development.
In a traditional game, the user will usually play a character, whose direct translatory movement can be controlled using analog control sticks on a game controller. Simply adapting this method for VR games has led to many users experiencing so called *motion sickness*, or more generally *simulation sickness*. This is most likely due to the fact, that the brain is experiencing movement through the visual stimulus, which it cannot confirm with its sense of acceleration of balance of the bodily stimulus. These different stimuli cause the user to experience a feeling of disorientation and general sickness.

This circumstance has led to the development of alternative locomotion mechanics: Among the most popular ones are *teleportation* and *rail based locomotion*.
With teleportation the user will utilize the motion controllers to point to a specific place in the world. After pressing an additional button, the vision of the player is quickly faded out into darkness and then faded in again, with the new position being the one previously pointed at.
Rail based locomotion still uses direct translatory movement of the player position, but with an important change in game design: The player is confined in a vessel such as a wagon or an aircraft, which is moving be itself. This type of motion is easier for the brain, as it can relate to real world experiences such as using a car or train, to explain the differences in stimuli.

That all being said, there is strong evidence, that direct character motion using analog sticks might still be a valid possibility. Studies could show, that even though the majority of people first exposed to a VR environment experience simulation sickness, it fades away after regular exposure. There is only a small fraction of the population, whose brain cannot adapt to the experience even after a long exposure time.

But since one cannot expect the audience to have accustomed to the motion sickness yet, it might still be a good idea to implement the teleportation option for movement.

4. Ressources

4.1 chapter overview

in this section I will introduce some helpful resources in the form of books and websites, which will be helpful for the development of VR project with the unreal engine.

Unreal for Mobile and Standalone VR [Hil19] The focus of this book is specifically on development for mobile VR hardware, although all the topics described are also applicable for *tethered VR platforms*[1].
This book includes general advice for the setup of a development pipeline, tools and design choices. Furthermore it includes two in depth tutorials: One for developing an interactive product showcase and another for developing a full game with all essential game mechanics.

4.2 Book *Unreal for Mobile and Standalone VR*

This book can be acquired for free from the KIT library[Web19], when accessed from the KIT wifi or a VPN, using the following URL:

`https://link.springer.com/book/10.1007%2F978-1-4842-4360-2`

Chapter 3: Before you start

The third chapter from the book was the basis for the chapter 3 about design choices in this document. Everything mentioned and a lot more is described here in detail.
Furthermore the third chapter includes some nice tips on *low hanging fruits*[2] of every VR project.

Chapter 4: Unreal for VR

The main focus of this chapter is on creating a VR project for the *Oculus Go* platform specifically. The basic tutorial for setting up simple motion controller locomotion blueprints is generally applicable nonetheless.

chapter 6: Creating an Interactive VR Presentation

This chapter is a full on tutorial on how to create a VR product presentation. Since a product presentation has a lot in common with a presentation for a science project, this chapter is the most relevant to get started with a project.
The chapter roughly contains the following content:

[1] Tethered VR platforms are those platforms, which need mounted basestations to track the position of the user. These platforms have the advantage of offering a 6 DOF (degree of freedom) VR experience in contrast to the 3 DOF of mobile VR platforms

[2] the term describes aspects of a project, which can improve the project a lot and are easy to achieve

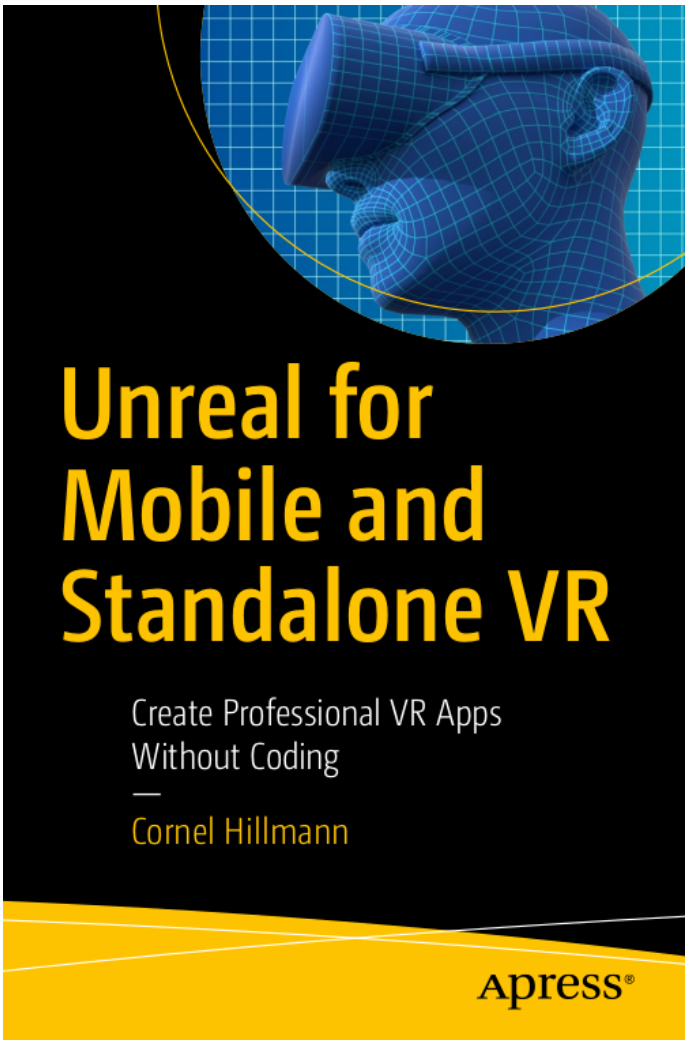

Figure 4.1: cover of the book "Unreal for Mobile and Standalone VR"

- Pipeline for creating the scene aesthetics using Unreal and Blender
- lighting the scene
- basic environment interactions with floating UI buttons and sequencer animations

5. Tutorials

5.1 Using the Sequencer for Basic Animations

Basic Introduction

Since Version 4.22 the Unreal Engine has been extended with the *Sequencer* Functionality. Within the editor, objects are being represented by their position, which is their X, Y and Z coordinates. Additionally to their position objects also have *properties*. You can think of a light source for example, one of its properties would be the intensity of the light.
With the sequencer tool you can create new "sequence" objects. With these sequences you can define specific sequences of position and property changes for objects within the game world. What would be an example for this? Imagine a game setting, where there is a closed door in a hallway. As the player walks down the hallway, he will trigger a new Sequence called "DoorSequence" for example. Within this sequence, you have previously defined, that the door will rotate outwards from its angles by 90 degrees, within a time frame of 1 second. Upon triggering the sequence the player will witness the animation of the door opening automatically, until it does not move anymore after 1 second and then remain open.
To summerize: The sequencer tool can be used for simple, world-bound animations. Why only simple animations? The sequencer system is relatively limited to manipulating "only" the position and other properties of *whole* objects. Thus it is not able to for example deform whole objects. But when animating a characters facial expressions for example, what you want to do is to make a series of deformations of the actual model as time progresses. To make these kind of *more dynamic* animations a different functionality of the Unreal Engine has to be used.

Basic concepts of sequences

The most important concept for working with animations is that of a *keyframe*. Overall you can compare the sequence object with a video: In the end the animation itself is nothing more than a series of still images played rapidly in series. Each one of these still images within a video is called *frame*. As such the different moments during a sequence are also called frame.
Now imagine an example: You simply want to move a red cube from left to right. So what you would do is create a first keyframe at the 0 seconds mark within the sequence, with the cube still being on the left. Then you would create another keyframe at 1 second, with the cube being on the right, where you want it to be at the end. Than you could already play the sequence and see an animation. Now you could ask: "How come, that I see the whole animation? I havent yet defined all the positions of the cube *in between* the start and the end!". This is why sequences are so convenient: You only need to set the *defining moments* or *keyframes* of an animation. The program will *interpolate* all the frames in between those to moments to create a smooth animation.

So in the end working with the sequencer really just comes down to setting the appropriate keyframes and letting the program do the rest

Setting up a Sequence

Now we are getting to the practical part. First, a quick summary of what has to be done for setting up a new sequence in the Unreal Engine:

- Setting

Changelog

Version 0

0.0.0 - 18.11.2019

- Created the git repository
- Copied the template for the KIT bachelors degree to be used as the basis of this document
- Added a first version of the "premise" chapter and "document overview"
- Added chapter about the system requirements of the unreal engine.

0.0.1 - 19.11.2019

- Added the chapter about installing unreal engine
- Added the chapter about the other application

Possible improvements in the future:

- Extend the other application chapter with the list of all the applications that was given in the book

0.0.2 - 21.11.2019

- Added a draft of the chapter about design choices. The chapter
  - Section about the scope of the project: Rather make the scope smaller, because a VR game is a lot of work
  - Section about "Why VR?" why should you even make a VR project. Could traditional media do the same?
  - Section about the method of locomotion in VR. Why teleportation might be a good idea.
- Began working on the "ressources" chapter, which is supposed to list a few helpful ressources (books mainly) in the pursuit of making a VR representation

0.0.3 - 22.11.2019

- Wrote the chapter about the book "Unreal for mobile and standalone VR"
- removed the appendix section for now. Since I dont need that yet.
- removed the inclusion of the preamble.tex
- removed the inclusion of the list of tables for now
- Added preliminary sections for the "tutorials" section.

0.0.4 - 26.11.2019

- Added all the missing citations
- Decided, that for the first version there will only be a sequencer section in the tutorials chapter
- Started to write the sequencer tutorial

Bibliography

[Fou] Foundation, Blender: *Blender.org - Home of the Blender project - Free and Open 3D Creation Software.*

[Hil19] Hillmann, Cornel: *Unreal for Mobile and Standalone VR: Create Professional VR Apps Without Coding / by Cornel Hillmann*. 2019, ISBN 978-1-4842-4360-2.

[Unr] *Hardware and Software Specifications*. `https://docs.unrealengine.com/en-US/GettingStarted/RecommendedSpecifications/index.html`.

[Web19] Webredaktion der KIT-Bibliothek: *KIT-Bibliothek*. `https://www.bibliothek.kit.edu/cms/index.php`, September 2019.