

# Virtual Production Implementation

## CMGT Graduation – Saxion

**Student:** Marco Nieuwenhuis 454721

**Bachelor:** Creative Media & Game Technologies

**Class:** EKT4c

**Company:** Tiny Giants  
Hoge Bothofstraat 39  
7511 ZA Enschede

**Company Tutor:** dhr. Arian Hohmann

**University:** Saxion University of Applied Sciences, Enschede  
M.H. Tromplaan 28  
7513 AB Enschede

**Project Tutor:** dhr. Taco van Loon

**Place, date:** Almelo, 11/05/2021

**Version:** 1

## Abstract

This paper written during the last semester of the CMGT course at Saxion focuses on the research and implementation of Virtual Production technology within the workflow of a 3D communication studio. Virtual Production (abbreviated as VP within this paper) is a new and upcoming technology that combines the physical and digital worlds by using camera's, trackers, and 3D software. The advantage of this technology compared to the industry standard workflow in which the greenscreen footage and 3D space are combined in post processing is, that VP allows the user to get direct feedback from the environment and see how things look and feel. This engagement allows for direct changes and feedback from the producer to get a better-looking image.

The project and its research are set up using different stages, like the Design Thinking Method, in which the received information will get verified using practical tests. With the results of those tests a prototype will be created on which future improvements will be made. The goal of this research is to acknowledge a basic understanding of the software and hardware used, in this case that is Unreal Engine 4, a camera and HTC Vive Virtual Reality headset. With this knowledge the foundations of a virtual production setup will be established which will serve as the building ground for the prototype and on which improvements will be made throughout the project.

## Contents

Abstract.....	2
Chapter 1 - Prologue .....	4
1.2 Saxion university & Tiny Giants.....	4
Chapter 2 – Reasoning and problem definition .....	5
2.1 Reason for the assignment .....	5
2.2 Problem statement .....	5
2.3 Problem definition .....	5
Chapter 3 - Main and sub questions .....	6
3.1 Main question .....	6
3. 2 Sub questions .....	6
Chapter 4 – Theoretical background .....	7
4.1 Sub question 1 .....	7
4.2 Sub question 2 .....	9
4.3 Sub question 3 .....	11
4.4 Sub question 4 .....	13
4.5 Sub question 5 .....	15
4.6 Sub question 6 .....	15
Chapter 5 - Results and conclusion .....	16
5.1 Prototype 1 – Tracked Camera Unreal Asset .....	16
5.2 Prototype 2 – Visualization Virtual Production.....	16
5.3 Prototype 3 – Hybrid Virtual Production.....	17
5.4 Conclusion .....	17
Chapter 6 – Recommendations and discussion .....	18
6.1 – Recommendations .....	18
6.2 – Discussion.....	19
Chapter 7 - Sources .....	20
Chapter 8 – Appendices .....	22

## Chapter 1 - Prologue

This research paper is set up within the framework of the graduation semester for Creative Media & Game Technologies at Saxion University of Applied Sciences in Enschede. This is in collaboration with Tiny Giants in Enschede.

### 1.2 Saxion university & Tiny Giants

In the last semester of CMGT studies at Saxion a research paper will be written in order for the student to graduate. This paper will focus on upcoming technology and growth of Virtual Production at an indie content creator scale. It includes a research on the subject, practical tests, and a prototype. Because this technology is new and upcoming, there are not that many reliable and established sources from the companies that helped create or contribute towards the technology. Because of this reason there have been made use of different sources in this research (YouTube video's, articles, etc.) to find the information needed and make the information given credible. The information found will be used in a practical test by the researcher in order to check if it is useable and contributes towards the goal of this research, or if the information/findings need some more research/tinkering for it to be useful.

The company at which this research will be done is Tiny Giants, a 3D Communication studio located at Hoge Bothofstraat 39, 7511 ZA in Enschede, the Netherlands. Tiny Giants was founded in 2016 by Ivan de Wolf, Bart Brinkman, Bas Helmig, and Arian Hohmann, a group of friends who have found this company during their studies. Now, after 4 years, international clients, and captivating content provided to corporates and organizations enhancing their stories, we have a team of experts with over 15 years of experience. You can find out more about Tiny Giants from Ivan in his video message to our audience - <https://www.tinygiants.nl/about-us>

With the research a prototype is created along with a manual for the company. The manual will include all the steps the company has to go through to setup a VP environment for their workflow along with an upgrade and troubleshooting part. The paper explains the choices that were made creating this manual and prototype based on the conducted research.

Finally, this paper will include several appendices, relevant images, notes, and information related towards the subject of Virtual Production and its growth. Whenever appendices are relevant to the research the text will refer to them.

## Chapter 2 – Reasoning and problem definition

### 2.1 Reason for the assignment

The reason for conducting this research is for the company to speed up their 3D workflow by being able to choose this technology as an optional step within their pipeline.

The researcher will look at different Virtual Production techniques to find and extract the most relevant and useful ones for the company. Based upon those results a prototype will be developed combined with a manual which describes the way to setup this prototype.

The result of this research can be beneficial for the company as it may save a lot of time in the production phase, as Virtual Production is a real-time based production, it will show the direct result with the benefit that changes can be made on the spot. This prevents the back and forth going between rendering out different environments to see how different changes may apply which in the end saves time.

### 2.2 Problem statement

To find and define the problem behind this research and the question of the client, a couple of questions will need to be answered. These questions provide more insight in the real problem, what the expected result of this research is, which parties are concerned/involved and which limitations the researcher and/or the company might stumble upon regarding the research.

#### *2.2.1 Description question of client.*

The question which the client formulated for this research is: What are the benefits of Virtual Production regarding to our current work and how can Virtual Production be implemented into our workflow if the benefits are in our favor to speed up our creative process?

#### *2.2.2 What products/services does the client need?*

The client needs the equipment and documentation of description and instruction on how they can perform a Virtual Production set which will accommodate their workflow.

### 2.3 Problem definition

Based upon the results mentioned in the problem statement the problem is defined as:

Since Virtual Production is a technology that is coming up quickly in the 3D and VFX industry, it is important for Tiny Giants to have a grip on this technology to discover its possibilities, and measure if its benefits can improve their workflow.

## Chapter 3 - Main and sub questions

To give the research a clear overview about which questions will be researched and how the desired answers will be achieved, a main question along with a couple of sub questions have been formulated. The answers to these questions will guide the research towards a clear end goal.

The main question for this research has been set up based upon the problem definition and a clear discussion with the client about their needs and interest. The sub questions are then formulated to divide the main question into smaller parts, each of them giving an answer to a part of the main question.

### 3.1 Main question

How to set up a virtual production set that can be used as a way to improve the workflow with the enhancement of operating a camera within 3D scenes and projects that Tiny Giants are working on?

### 3.2 Sub questions

- Which hardware and software are needed for the desired result?
- Which kinds of Virtual Productions setups are eligible to achieve the desired result?
- To what extent is the tracking hardware accurate compared to real-time movement as in movement and timing?
- What are the features that will enhance the operation of a camera in 3D space?
- How can this set-up be set up in an easy and quick way that makes it useable for users without much experience/knowledge?
- Which kinds of data can be exported from this set up?

## Chapter 4 – Theoretical background

In this chapter the researcher will answer the sub questions with the theory and tests relevant towards the question.

### Research method

To start of the research, an in-depth interview has been held with the client to find out what exactly they expect, their desired outcome and which subjects regarding virtual production they would like to receive more knowledge about.

This led to the creation of a main question along with a couple of sub-questions. To find all the answers to the sub-questions and other questions/problems that came to daylight during the research there has been made use of a qualitative, secondary research method which have been measured using experiments. The sources for these answers came from a wide range of content such as articles, video tutorials and explanations, documentation and conducting professionals.

To measure all the information found, a test setup has been made for the purpose of this research. This test setup consisted out of the following hardware:

- 2x3M Green screen,
- HTC Vive + Tracker 2.0 with two basestations covering a 2mx3m space\*,
- MSI Nightblade MI3 Desktop PC\*,
- Panasonic Lumix DMC-GX80 system camera with a 12-33mm lens,
- Capture card + Micro HDMI to HDMI cable

The objects marked with an asterisk (\*) are the only ones needed to conduct prototype 1 | Visualization.

### 4.1 Sub question 1: Which hardware and software is needed for the desired result?

To define which technology is needed the researcher must first need to set in stone which type of virtual production we are aiming for. After a discussion with the client, we have concluded that we are aiming for two different approaches which are: Visualization and Hybrid Virtual Production.

There are two types of Virtual Production which are applicable for this research, these are Visualization and Hybrid Virtual Production. Visualization is defined as prototype imagery created to convey the creative intent of a shot or sequence, while Hybrid virtual production is the use of camera tracking to composite green screen cinematography with CG elements (Kadner, 2019, p. 11-18).

To define which technology is needed to create these productions the researcher must investigate which kind of setup is applicable. According to Viehmann (2020) there are four types of setups: No sync (Testing VP Out), Partially Synced (Hyper Indie VP), Synced Greenscreen (Industry Standard VP) and Synced Volume (Filmmaking VP). Looking at the technology we can access and budget we are aiming for the Partially Synced set up. To define which technology is needed to create this set up we need to look at which hardware and software is needed.

#### *4.1.1 Software*

The software which is used to create a virtual production is Unreal Engine, this game engine is the base of every professional real time rendered environment. It has a lot of build-in and free to use plugins that enables the users to create a virtual production set without the need of having a lot of experience, it also comes with a huge library of free to use assets.

#### *4.1.1 Hardware*

The hardware part can be divided into four topics: Tracking systems, Camera, Computer hardware and background keying.

The tracking system can be upgraded from a simple VR setup to a professional tracking system, the professional tracking systems provide a much more stable and precise tracking solution but comes at the cost of a much higher budget. For this research we will focus on the HTC Vive + Tracker, this set up lets us track our camera movements into Unreal Engine.

#### *4.1.1 The camera*

There are a lot of different types of cameras we can use for virtual production ranging from webcams to high-end cinematography cameras like a Blackmagic camera. If it provides a decent quality of footage, it can be used to some extent. The camera we are using for this research is a mirrorless camera (Lumix GX80) which shows direct footage to the game engine using a capture card.

#### *4.1.1 The computer hardware*

The hardware requirements are not too high-end as Unreal Engine is a well optimized game engine so in reality almost any GPU that was released in the last 5 years is useable. Although the greatest advancement for virtual production recently has been the use of realtime ray tracings, which is only achievable with the latest RTX cards.

#### *4.1.1 Background keying*

In Hollywood sized productions a LED wall is used to visualize the 3D environment in real time, for this research we will use a green screen to separate the real footage from the background and composite it in our 3D environment using CG element layers.



## 4.2 Sub question 2: - Which kinds of Virtual Productions setups are eligible to achieve the desired result?

There are numerous different ways of virtual production that can be used to get different results as stated by (Kadner, 2019, p. 11-18). To find out which setup is the most eligible for the desired outcome of this research, multiple different setups have been researched and tested.

Viehmann (2020) stated in his article several levels of set ups along with its required hardware, the researcher has taken one of these setups, as mentioned earlier, as a baseline for which result will be aimed for.

In a video from Unreal Engine (2020) shows Matt Workman, founder of Cinematography Database, breaking down several different kinds of virtual production set ups, this video adds up along with the videos from Max Müller (2020), Richard Frantzén (2019) and Cinematography Database (2020) to quite a big extend of different types of setups which can be accomplished with low-budget consumer-based products.

Using the acknowledged information from these sources and the available technology two different setups have been chosen to be pursued: Visualization Virtual Production and Hybrid Virtual Production.

### *Visualization Virtual Production*

Visualization is the virtual production setup that most people are aware of and can be defined as prototype imagery created to communicate the creative intentions of a shot or sequence.

Visualization can take multiple forms such as pitchvis, previs, virtual scouting, techvis, stuntvis, and postvis. For this research Virtual Scouting and Techvis are the two forms which are appealing to dive further into.

#### *Virtual Scouting*

Virtual scouting presents a completely digital version of a location or a proposed set which crew members can interact with.

The VR part of Virtual scouting includes creating/modifying the environment, add cameras, plan out shots, create a set and even shooting an entire sequence without the need of any physical object/space. This allows filmmakers to focus on only the desirable regions of importance and leave out the ones that are less significant.

#### *Techvis*

Techvis is the combination of CG elements with practical equipment. This combination can be used to create shots as well as uniting pre-made footage with the CG assets. It is also the area where camera translation, placement, and lens choices can be justified, extenuating the risk of physically impossible virtual choices.

The reason for pursuing this type of virtual production is the purpose of using a tracker as a 3D camera that can be moved around in real-life to create cinematic shots within the 3D space, this setup is aimed at creating a viable replacement for manually animating cameras movements that can be done in real life.



Figure 1. Example of visualization virtual production for GoT (Source: Unrealengine.com)

### *Hybrid Virtual Production*

Hybrid virtual production is the use of camera tracking to composite green screen cinematography with CG elements. (Kadner, 2019)

This creation can be used as a live preview for the director, director of photography and the camera operator to be finalized in post or it can be used as the final in camera result that will be used. This case of virtual production is commonly seen in live broadcast but has been gaining a lot of attention and popularity for high-end production recently, like The Mandalorian for example.

The reason for pursuing this type of virtual production is to create a viable option for shooting or previsualizing a CG scene with real footage which outputs a live preview which can be adapted on the spot instead of going into a loop of reviewing the footage and recreating a shot or environment.

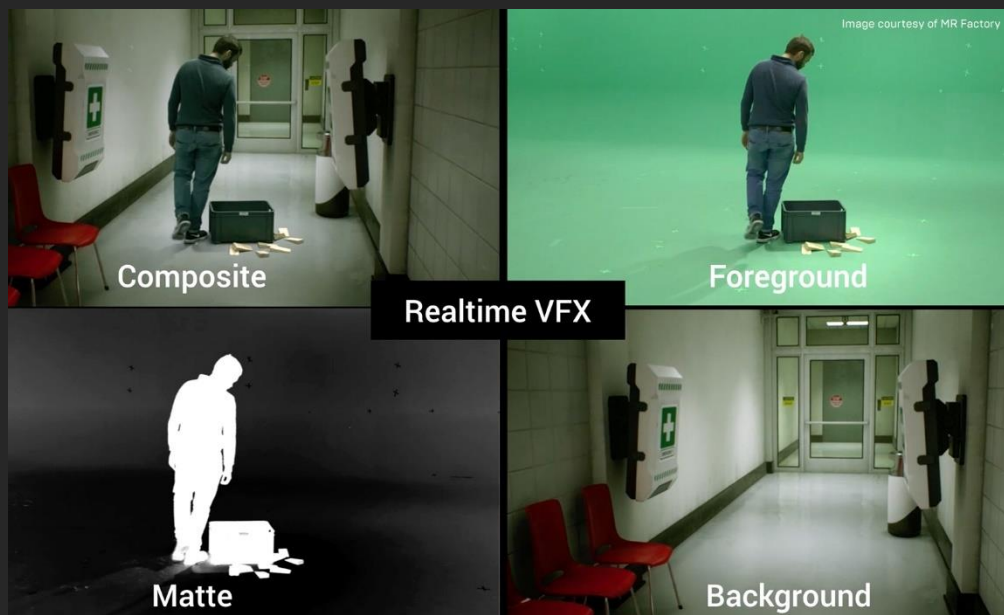


Figure 2. Example of hybrid virtual production with a green screen composition and a CG environment (Source: Unrealengine.com)

### 4.3 Sub question 3: To what extent is the tracking hardware accurate compared to real-time movement as in movement and timing?

To determine if the Vive is a viable option regarding this research, its accuracy has been researched and measured using an experiment. According to Borges, Symington, Coltin, Smith & Ventura (2018) is the HTC Vive an affordable solution for localization problems, in indoor and outdoor environments with great accuracy. Their research demonstrated that it has sub-millimetric precision when the tracker is static. But when in motion it could become very unstable ranging from millimetric up to metric due to rogue reflections perturbing the data or to a sudden loss of measurements.

In a research done by Niehorster, Li & Lappe (2017) concluded that While tracking is subjectively fast and supports good presence, the system end-to-end latency is low at 22 ms. They placed the Vive at each of the grid points (see Figure 1(a)) while the headset was facing either along the positive X axis, or along the negative X axis of the room. Figure 2 shows the mean recorded X-Z positions at each grid point. Most of the recorded X-Z positions seem to fall along a regular grid, both when the headset faced along the negative and the positive X axis of the room. Measures for some of the grid points were not available as tracking was lost there.

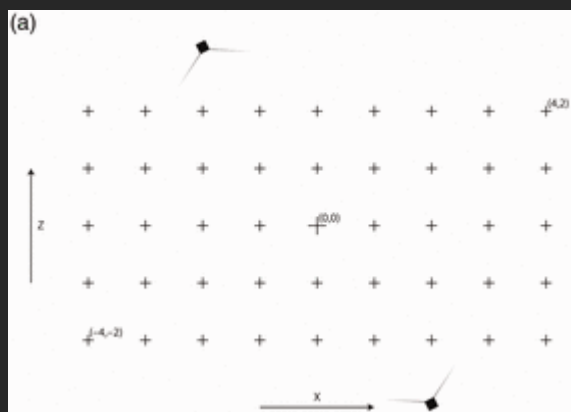


Figure 3. (a) Top view of the measurement setup. The crosses indicate the grid points that were drawn on the floor, and the black boxes indicate the lighthouses units.

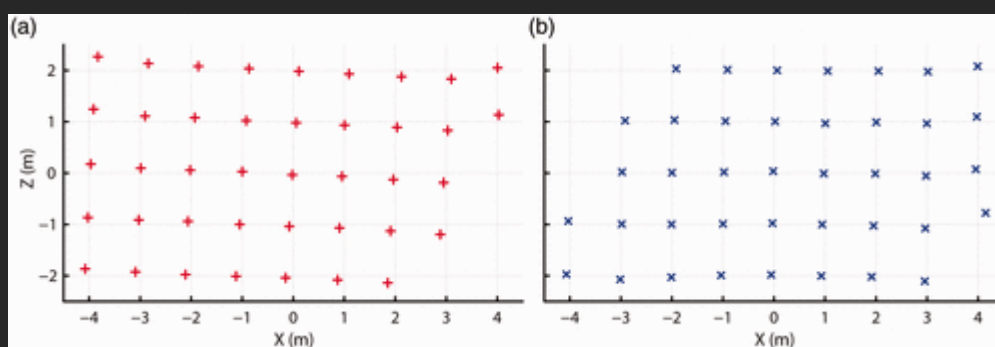


Figure 4. Recorded X-Z position (a) for when the headset faced along the positive X axis and (b) for when the headset faced along the negative X axis.

However, as Wiesing, Fink & Weidner (2020) stated in their research towards the response time of Unreal Engine and SteamVR, that it should be noted that the method heavily relies on game engines and VR runtime software. This statement is also applicable when looking at the accuracy of translation of positioning.

#### 4.3.1 Testing

To test out the accuracy of HTC Vive that is being used for this research, a small setup environment was created by using a small X and Y grid on the floor along with markers around 10 cm apart and a small area with 1cm apart together with a rotational platform. In this test the Vive Tracker 2.0 was used as the tracked device along with 2 basestations set up approximately 4.5 meters apart and facing each other.

To test out the accuracy the tracker was moved around on the grid and its movement got translated, recorded, and shown into Unreal Engine. To get the results of this test from the game engine, the precise changes in movement have been recorded into the console of Unreal Engine and written down. These results showed that the VR setup was indeed as accurate as described in other papers, up to an average of 2mm in this case. The precise rotational values could not be tested as Niehorster, Li & Lappe (2017) have shown that the Vive uses a reference plane for its measurements that is tilted with respect to the physical ground plane.

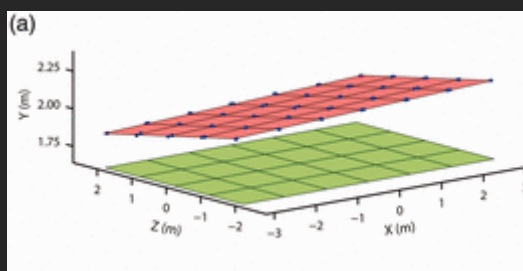


Figure 5. (a) The Vive's reference plane (red) fit to the 3D measurement locations (blue dots) reported by the Vive. The green plane indicates the plane formed by the physical measurement locations.

#### 4.4 Sub question 4: What are the features that will enhance the operation of a camera in 3D space?

When dealing with a real camera there are certain aspects and gadgets that make using the camera easier, such as a gimbal or tripods. However, these features can also be used to operate a 3D camera using a Vive Tracker.

Operating a camera in 3D space is not that difficult, an animator sets up a certain amount of keyframes, and the software does its work however, when it comes to tracking, creating changes in in animation or animating a 3D camera to make it look like it is being operated hand-held becomes difficult and very time consuming to achieve. This is where the tracked camera in a Virtual Production shines through, this set up allows to use to achieve all these things easily.

To create the tracked virtual cameras for the prototypes, a couple of guides have been followed on how to set these up. A video playlist from Wilson (2021) demonstrates how to set up a tracked virtual camera along with some tweaks and improvements, this guide has been used to create the first prototype (Ch. 5.1).

The blueprint for the camera includes functions attached to the camera which can be triggered using the Vive controller to enhance the operation of a camera such as adding a smoothness to the movement, cycle through lenses or zooming in or out. All within the push of a button.

##### 4.4.1 Testing and tweaking

To test out this blueprint, a small testing environment was set up together with an Unreal Scene. During the experiment, the researcher had to tweak the blueprint for Unreal to get the right Tracked Device ID (Figure 5). This device ID was linked to the sequence in which each HTC Vive accessories were activated which could change every time the headset was rebooted.

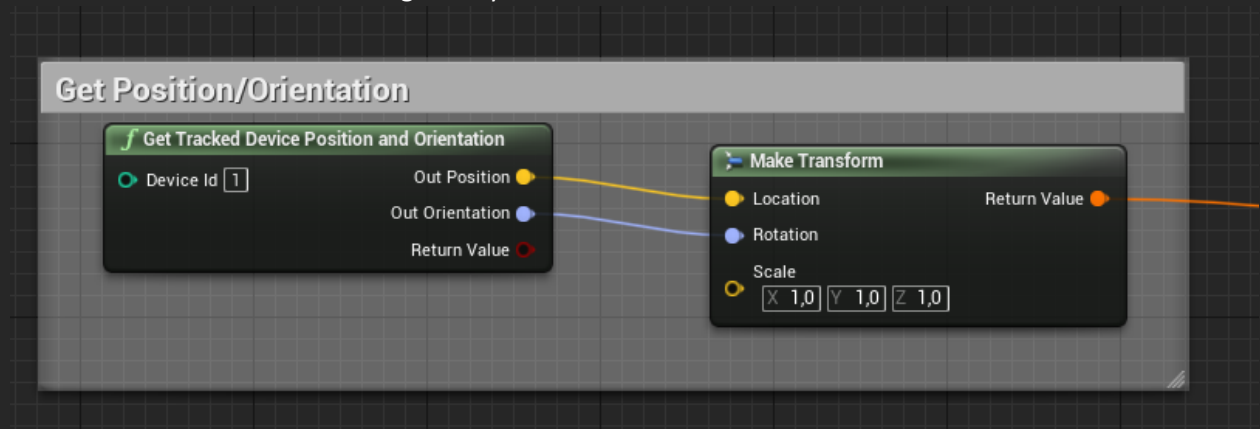


Figure 6. A part of the camera blueprint which shows the “Get the tracked device ID” (the Tracker) and translate its movement.



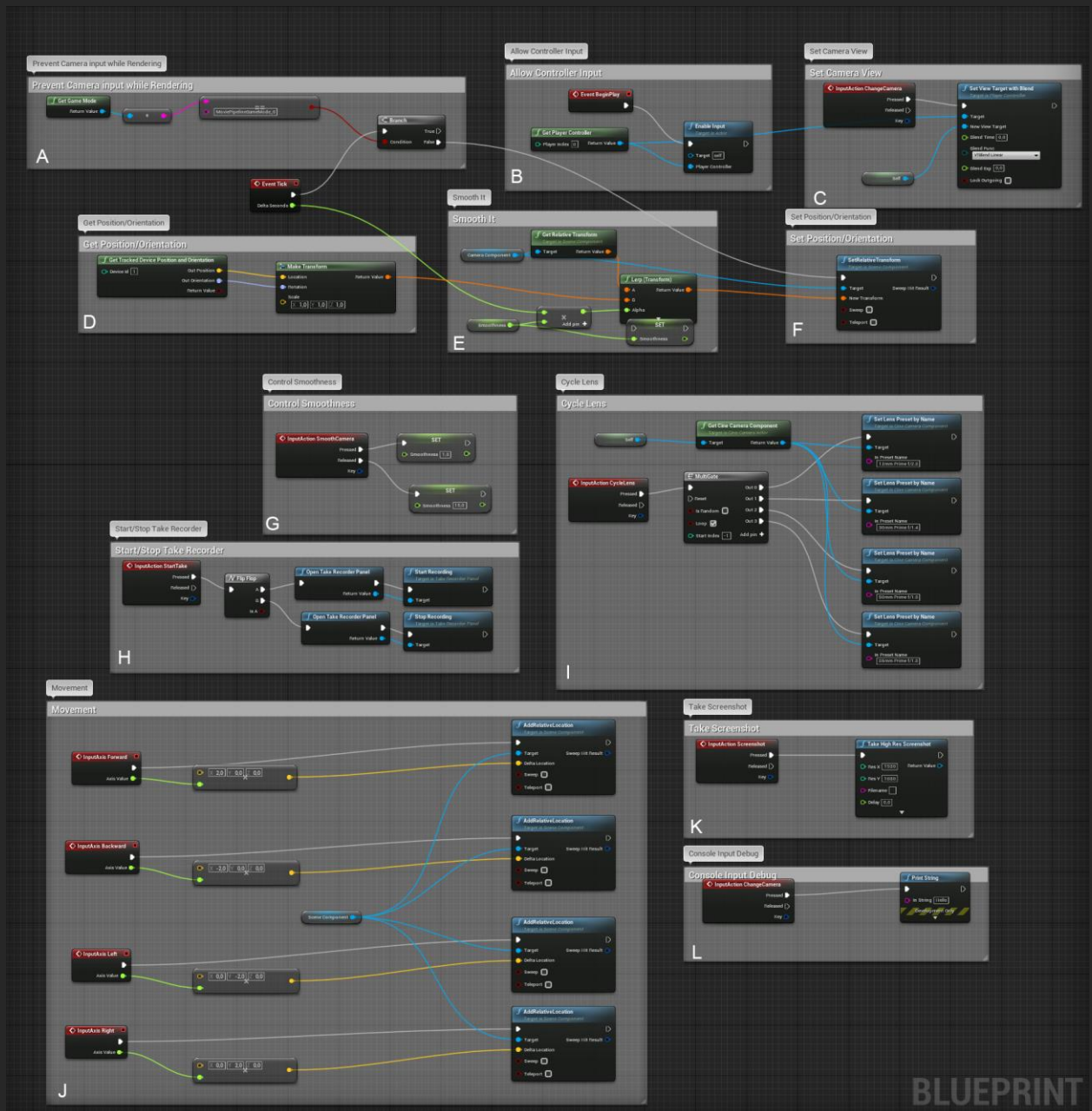


Figure 7. The whole blueprint from the tracked camera with all its additional functions separated into several boxes.

#### 4.5 Sub question 5: How can this setup be used in an easy and quick way that makes it useable for users without much experience/knowledge?

To make the findings and prototypes from this research useable for non-experienced users an investigation towards various ways to make a virtual production setup as easy as possible has been done.

The results of this investigation resulted in written guides (Ch. 5.2.1 & 5.3.1) for the client to use the set ups created by the researcher as intended. These guides will show the steps to get the prototypes working, how it works along with a troubleshooting part which can be used if certain aspects do not work as intended. These guides are written as comprehensible as possible, to achieve this the researcher utilized a couple of sources (Bloom, 2019) ("Creating a Guide", n.d.) ("Creating a 'How to' Guide", n.d.) describing the matters which make a guide good.

As for parts such as the virtual camera that is connected to the Vive Tracker, it consists out of an Unreal asset along with a custom build blueprint. To make this prototype easy to use the researcher has reduced and converted this whole prototype into one Unreal Asset which can be ex- and imported into any Unreal project.

#### 4.6 Sub question 6: Which kinds of data can be exported from this set up?

Unreal Engine allows its users to export a various amount of data ("Data Driven Gameplay Elements", n.d.), only a small part of this data is useful regarding the research. To funnel down the data needed the researcher looked at the different prototypes and which data needs to be extracted for them to function.

##### 4.6.1 FBX Export

For the second prototype (Ch. 5.2), the transferring of the camera footage made by the prototype towards the desired 3D software, the data from the translation of the camera in Unreal Engine is needed. For this data to be useable in all (standard) 3D programs it is best to export this camera as an FBX file. Unreal Engine includes a FBX exportation tool ("Exporting and Importing FBX files", n.d.) which makes exporting and saving this data simple.

##### 4.6.2 Camera info

Along with the translation of the camera mentioned before, the FBX exportation method also allows the transportation of different information about the camera such as lenses, focal point, zoom, etc. This information is useful when being used for prototype 2 (Ch. 5.2) where no real camera is being used.

##### 4.6.3 Footage

To render out the footage made with the composition tool and sequencer from Unreal Engine, the Movie Render Queue is the most viable option. ("Rendering Out Cinematic Movies", n.d.) These tools allow the user to set up their desired cinematic settings and sequence to batch render the footage in one go to be used as an outcome or be post-processed. ("How to Use the Movie Render Queue for High-Quality Renders", n.d.)

## Chapter 5 - Results and conclusion

The outcome from all the research that has been done led to the creation of three prototypes which will answer the main question: “How to set up a virtual production set that can be used as a way to improve the workflow with the enhancement of operating a camera within 3D scenes and projects that Tiny Giants are working on?”

These prototypes provide the client with various options to create a virtual production set that is focused on a specific need. An in-depth user manual has been written to guide the user with setting up all the prototypes and how to use them

### 5.1 Prototype 1 – Tracked Camera Unreal Asset

Prototype 1 is an Unreal Engine 4 Asset which can be imported into any Unreal 4 project and allows the user to use a tracked device to operate the camera attached in the asset. This asset is made of a Virtual Cinema Camera with a blueprint (script of node/code) attached, this blueprint enables the user to operate this camera with a tracked device such as an HTC Vive Tracker. Besides the tracking, this blueprint also contains numerous other features which will make operating the camera in 3D space much more user-friendly. These features are:

- Smoothing (Like a gimbal),
- Taking screenshots at a high resolution,
- Allowing the user to give input,
- Making big movements in the scene,
- Debugging the controller or blueprint,
- Start and stop recording the movement,
- Cycle through different lenses.

All these features are bound to a key on the HTC Vive controller but can manually be rebound to any key the user desires. The mapping file which can be imported into Unreal is provided along with the asset file. This prototype can act as a baseline on which any kind of virtual set can be built. The manual explains to the user what

### 5.2 Prototype 2 – Visualization Virtual Production

Prototype 2 uses the same asset and key bind mapping file from prototype 1, but this prototype is aimed towards guiding the user with using that asset to use for visualization virtual production. The goal of visualization virtual production, in regarding to the needs of the client, is to use the tracking system to create hand-held like shots in 3D space, then export the camera as a FBX to be later imported into the desired 3D software to be rendered out.

An example of this outcome can be found in appendix 2, chapter 8.

#### 5.2.1 Prototype 2 - Guide

This prototype is a written chapter in the user manual (Appendix 1) which will start with providing the user informational redirections to sources that will explain how to export a scene from any 3D software (For example cinema4D) and import this scene into Unreal Engine.

Once the scene is set up by the user, the guide will then explain in detail which steps the user can follow to set up Unreal Engine to start recording the movement of the camera and export this data as a FBX to be imported elsewhere.



### 5.3 Prototype 3 – Hybrid Virtual Production

Prototype 3 is made of two parts, an Unreal Engine 5 Asset which can be imported into any Unreal 5 project combined and a guide to setup a Hybrid virtual production set. This set allows the user to combine real footage with a CG environment in real-time using a compositing and tracking technique in Unreal.

#### 5.3.1 Prototype 3 – Unreal 5 Asset

The first part of this prototype is an Unreal Engine 5 asset, this asset is a pawn (equivalent to a controller) which has a blueprint attached to it like the first prototype. This blueprint provides the following features:

- Attaching a tracked device to control the pawn using a real-life controller,
- Enabling and starting live video feed when the user presses “Play”,
- Set the target framerate of the engine to match the framerate of the video feed,
- Allowing the user to give input,
- Attaching a Cine Camera Actor to the pawn to film the CG environment,
- Add a delay to the tracking to match the live video feed.

To setup all the inputs for these variables in an easy and quick way they have been made editable in the details panel of the asset, this makes it, so the user does not have to open the blueprint itself and find all the inputs manually.

#### 5.3.2 Prototype 3 - Guide

The second part of this prototype is a written guide (Appendix 1) which will explain the following processes:

- Importing and setting up the asset,
- Connecting the tracked devices into Unreal,
- Setting up the camera feed,
- Creating a composition,
- Chroma keying the greenscreen,
- Outputting/Rendering out the final composition.

The guide will explain every process in detail with the assumption that the user has no experience setting up a hybrid virtual production set in Unreal Engine 5.

### 5.4 Conclusion

While Virtual Production is becoming more essential and increasing in popularity, it is important for smaller companies to find out the potential benefits and different uses to know at an early state if this technology may advance their workflow.

By finding and measuring the information found regarding the problem statement and questions answering this statement, this research established that for the desires and needs of the client to be answered there are two virtual production methods applicable being: visualization virtual production and hybrid virtual production. With the right equipment that enables to track motion of a real-life object into a program at a low latency, both methods offer a solution to multiple scenarios that involve either hand-held camera movement, animating a camera or using a greenscreen that can save the client time in the creative process.

Transitioning to virtual production as a method for solving the stated scenarios will give the client an advantage in the long-term interest in filmmaking when virtual production eventually becomes an

industry standard technology for camera tracking and live compositing. In case of a future research into virtual production for this client, it should focus on cases where it could have been a solution to a problem that was being ran into.

## Chapter 6 – Recommendations and discussion

This research revealed that a good standard and quality of virtual production can be achieved using standard consumer-grade hardware and technology. On this basis there are a couple of recommendations regarding this research for the client which will allow them to enhance the process and upgrade the quality.

### 6.1 – Recommendations

As a recommendation for any kind of further research the client expects to do into virtual production, it is suggested to investigate cases they have experienced during their previous work in which virtual production might have solved a problem they encountered in post-production. For example, the wrong lighting or a camera feed that is unable to get tracked with just the footage.

The following recommendations are aimed towards the potential benefits from upgrading certain hardware aspects of a virtual production setup.

#### 6.1.1 - Tracking system upgrade

Using the HTC Vive, which is most used for indie-level virtual production, is a great way to get started due to its low costs, customizable features and it can be moved around easily. However, when the client decides they want a secure space within their studio to setup a virtual production studio, it is recommended to investigate more expensive and in-depth tracking systems which can be mounted into the roof or floor of the studio and provides a more stable setup such as the “Antilatency” system.

#### 6.1.2 - Capture card upgrade or Genlock

For the experiments of this research, a low-budget capture card was used. This resulted in a high delay in the live footage between the camera and the computer which eventually had to be fixed for the setup to work properly. Due to this conclusion, it is recommended for the client to investigate into PCI capture cards of a higher budget, from Elgato or Blackmagic for example, which can resolve this problem by having a much lower delay to an almost direct feed and allows for SDI input.

To completely resolve this problem the client might research the technique of Genlock which is a feature on certain cameras combined with a piece of equipment that synchronizes the video output of one source with another. This will make sure that both sources are played at the same time with no delay.

#### 6.1.3 - LED panels

A huge upgrade and recommendation for the client from the setup used in this research is, replacing the green screen with LED panels. A green screen was used to remove the background of the live footage to be replaced with a CG environment in this research and this method works well if the whole greenscreen is properly light up, and it is easy to set up and get going. But it comes with the problem of having to match the lighting and the reflections not correctly showing the CG environment, this is where LED panels which project the environment around the subject can solve a lot of these problems. One problem with LED panels currently is the pricing, having a LED wall is not something you get as a small studio so the recommendation for the client is to keep this option in the back of their mind, and following this technology which in the future will be more affordable for smaller studios.

## 6.2 – Discussion

Several things can be brought up for discussion from the research that has been carried out, underneath you can read them those points the reason on why something was missing in or holding back the research.

One of the bigger problems encountered was that this research was dependent of the available equipment that could be borrowed from Saxion's XR Lab and the budget of the researcher. This had impact on the outcome of certain tests and choices made throughout the research.

In terms of Unreal Engine experience, this research has the assumption of the user already having some knowledge of the software and only looks at the technique of setting up different virtual production sets in Unreal Engine but not at creating actual scenes or the cinematic part of Unreal Engine. This might be something the user still has to do after reading the research and manual.

As time goes on and programs get updated, so will Unreal Engine. This will result in the risk of the experiments and prototypes might not having similar results in newer versions of Unreal Engine as plugins/runtimes might get updated or changed in the future.

To give answer to how much time a virtual production might save compared to the “old-school method” is also nearly impossible to say, as this factor might be influenced by all other kinds of processes that have to be done which differs in every situation of creating a video.

## Chapter 7 - Sources

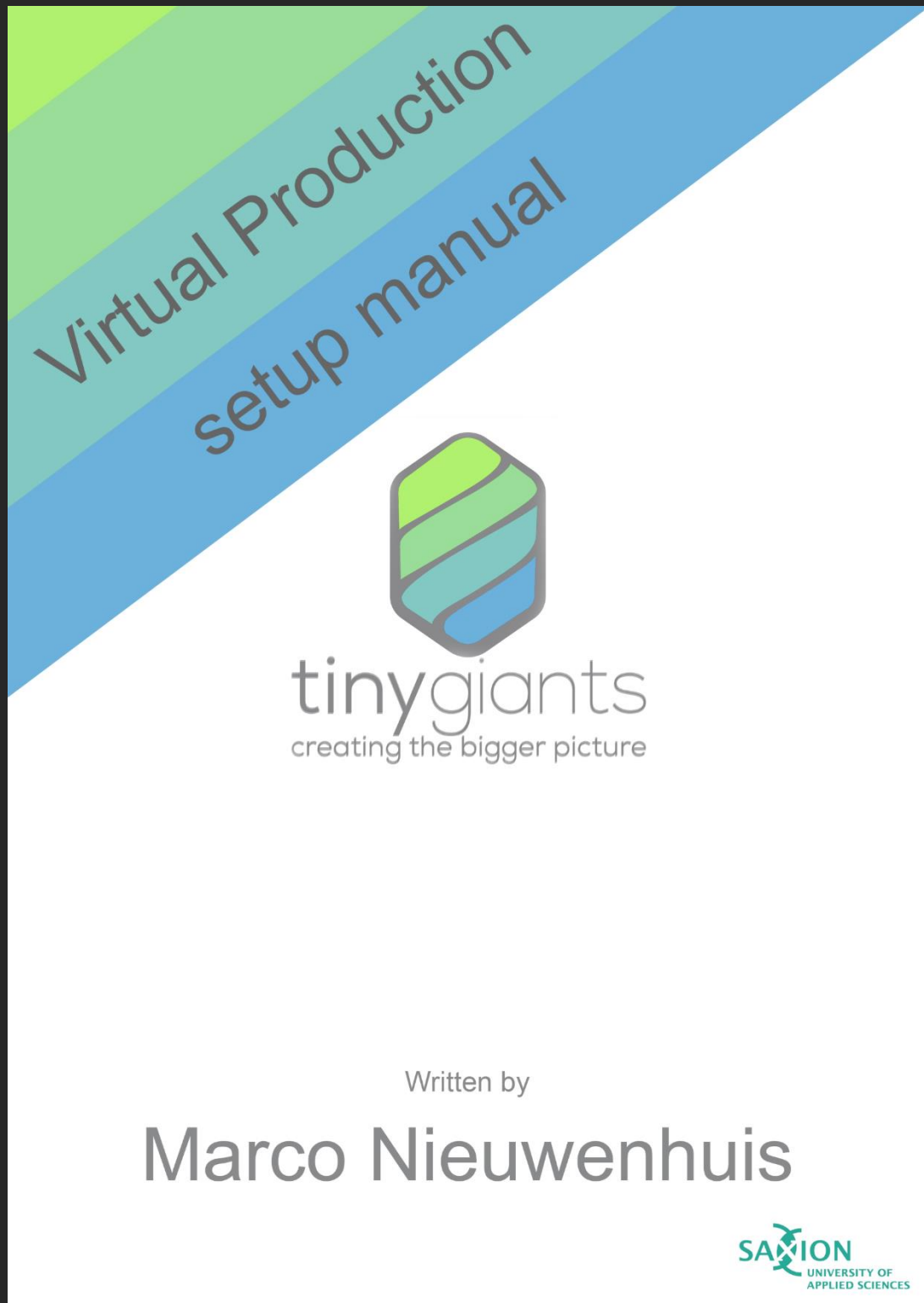
- Bloom, L. (2019). How to Write a Great User Manual in 12 Steps [Blog]. Retrieved from <https://www.dozuki.com/blog/2017/01/12/how-to-write-a-user-manual>
- Borges, M., Symington, A., Coltin, B., Smith, T., & Ventura, R. (2018). HTC Vive: Analysis and Accuracy Improvement. 2018 IEEE/RSJ International Conference On Intelligent Robots And Systems (IROS), 1(1). doi: 10.1109/iros.2018.8593707
- Cinematography Database. (2020, February 26). Indie Virtual Production is HERE! [Video]. YouTube. [https://youtu.be/C-M\\_GEKKmfU](https://youtu.be/C-M_GEKKmfU)
- Creating a Guide. Retrieved 11 May 2021, from <https://www.bath.ac.uk/guides/creating-a-guide/>
- Creating a 'How to' Guide. Retrieved 11 May 2021, from <https://www.bath.ac.uk/guides/creating-a-how-to-guide/>
- Data Driven Gameplay Elements. Retrieved 11 May 2021, from <https://docs.unrealengine.com/en-US/InteractiveExperiences/DataDriven/index.html>
- Exporting and Importing FBX files. Retrieved 11 May 2021, from <https://docs.unrealengine.com/en-US/AnimatingObjects/Sequencer/Workflow/ImportExport/index.html>
- How to Use the Movie Render Queue for High-Quality Renders. Retrieved 11 May 2021, from <https://docs.unrealengine.com/en-US/RenderingAndGraphics/RayTracing/MovieRenderQueue/index.html>
- Kadner, N. (2019). The Virtual Production Field Guide [eBook] (2nd ed., pp. 11-18). Retrieved from <https://www.unrealengine.com/vpfieldguide>
- Max Müller. (2020, September 10). Building a Virtual Production Studio at College with Unreal Engine 4 [Video]. YouTube. <https://www.youtube.com/watch?v=eYCnxCGatOo>
- Niehorster, D., Li, L., & Lappe, M. (2017). The Accuracy and Precision of Position and Orientation Tracking in the HTC Vive Virtual Reality System for Scientific Research - Diederick C. Niehorster, Li Li, Markus Lappe, 2017. Retrieved 11 May 2021, from <https://journals.sagepub.com/doi/10.1177/2041669517708205>
- Rendering Out Cinematic Movies. Retrieved 11 May 2021, from <https://docs.unrealengine.com/en-US/AnimatingObjects/Sequencer/Workflow/RenderAndExport/RenderMovies/index.html>
- Richard Frantzén. (2019, December 6). In depth Tutorial - Virtual Production with Unreal Engine and HTC Vive [Video]. YouTube. <https://www.youtube.com/watch?v=XWQZNw9iMNs>
- Unreal Engine. (2020, August 6). Work-From-Home Virtual Production | Unreal Fest Online 2020 [Video]. YouTube. [https://www.youtube.com/watch?v=-EGIAIjLF\\_M](https://www.youtube.com/watch?v=-EGIAIjLF_M)
- Viehmann, D. (2020). Virtual Production: Exactly Where to Start. Retrieved 11 May 2021, from <https://drewviehmann.medium.com/virtual-production-exactly-how-to-start-f73323c3b1d5>
- Wiesing, M., Fink, G., & Weidner, R. (2020). Accuracy and precision of stimulus timing and reaction times with Unreal Engine and SteamVR. Retrieved 11 May 2021, from <https://doi.org/10.1371/journal.pone.0231152>

Wilson, A. (2021). Before you continue to YouTube. Retrieved 11 May 2021, from <https://www.youtube.com/playlist?list=PLhmYb4VbSqh3UvGvGO3CD4wqqGI62FjaQ>

## Chapter 8 – Appendices

Appendix 1. Virtual Production Setup Manual written by Marco Nieuwenhuis for Tiny Giants

This manual guides the reader through the process of setting up each prototype described in chapter 5.



## Table of Contents

Intended use .....	25
Hardware and software requirements .....	26
Installation instructions .....	27
Plugins .....	27
Prototype 1 – Unreal Camera Asset.....	28
Programs: .....	28
Files: .....	28
How to use .....	28
How to import.....	28
Camera Blueprint .....	28
Controller Bindings .....	28
How to set up.....	28
Blueprint breakdown and controls .....	29
Console debugging [1] .....	29
Controller Input [2] .....	29
Change viewport view to camera [3] .....	29
Take a screenshot [4] .....	29
Prevent camera input while rendering [5] .....	29
Get position and orientation of the tracker [6] .....	29
Smoothing the camera movement [7] .....	29
Control the smoothness [8] .....	29
Translate the position and orientation to the virtual camera [9] .....	30
Start and stop the Take Recorder [10] .....	30
Cycle through lenses [11] .....	30
Move the Virtual Camera around [12] .....	30
Prototype 2 – Tracking and exporting as FBX .....	33
Programs: .....	33
Files: .....	33
How to use .....	33
Preparation .....	33
How to set up and use .....	34
Recording .....	34
Panels .....	34
Creating or selecting the right Animation Sequence to record to .....	34

Take recorder settings .....	35
Placing the Camera and entering play mode .....	36
Start and stop take .....	36
Check recorded take .....	36
Exporting .....	37
Prototype 3 – Green screen composition .....	38
Programs: .....	38
Files: .....	38
How to use .....	38
Preparation .....	38
How to setup .....	38
Setting up Live Link .....	39
Getting the camera live feed input .....	40
Calculating the delay .....	41
Creating the composition .....	43
Removing the green/blue screen .....	44
Adding the CineCamera to the composition .....	45
Creating composure material .....	46
Adding the composure material .....	46
Outputting the composition .....	46
Troubleshooting information .....	48



## Intended use

This manual will guide the user through the different kinds of setups depending on the intended use for the virtual production:

Prototype 1 is aimed towards using a tracker to control a camera in Unreal. This prototype includes an in-depth tracking system in its blueprint that includes several features as explained.

Prototype 2 is aimed towards the exportation of 3D camera footage as FBX to be intended to render in another 3D software such as Cinema4D or Blender.

Prototype 3 is aimed towards real-time compositing of greenscreen footage with a CG environment and exporting this footage.

**Only prototype 3 has been tested in Unreal Engine 5, so there is no guarantee for prototype 1 and 2 to work in Unreal Engine 5.**

In case the files are lost, you can download the files here:

[Google Drive](#)

[Mega](#)

## Hardware and software requirements

There requirements are based upon the minimum and recommended hardware specifications of Unreal Engine and HTC Vive:

<b>Processor</b>	Intel Core i5-4590/AMD FX 8350 equivalent or better	Intel Core i5-4590/AMD FX 8350 equivalent or better
<b>Memory</b>	8 GB RAM or more	8 GB RAM or more
<b>USB port</b>	2x USB 2.0 or newer	2x USB 2.0 or newer

Along with computer the following hardware components are required:

- A camera that can output live feed
- A cable to connect the camera to a PC
- A Virtual Reality headset/tracking system

The following programs are needed for everything to work:

Epic Games' Unreal Engine 4 or 5\* [Download page](#)

HTC Vive\*\* [Download page](#)

Steam with SteamVR [Steam download page](#) - [SteamVR download page](#)

\*Depending on the prototype

\*\*While running the HTC Vive setup make sure to select the correct headset that is being used in the setup.

## Installation instructions

### Plugins

For all the prototypes work correctly make sure to enable the following plugins in Unreal Engine 4:

1. Virtual Production
  - a. Composite Plane
  - b. Take Recorder
  - c. Timed Data Monitor
  - d. Virtual Production Utilities
  - e. VirtualCamera
2. Virtual Reality
  - a. Oculus VR
  - b. SteamVR
  - c. LiveLinkXR
3. Rendering
  - a. Image Plate
  - b. Media Compositing
  - c. Movie Render Queue
4. Media
  - a. Timecode Synchronizer
5. Editor
  - a. Media Player Editor
6. Compositing
  - a. Composure
7. Animation
  - a. Live Link

The plugin list can be found in the top bar under Edit -> Plugins.

## Prototype 1 – Unreal Camera Asset

Programs: Unreal Engine 4

Files: Prototype\_1\_Tracked\_Camera.uasset & Prototype\_1\_Controller\_Input.ini

### How to use

This asset contains a virtual camera for Unreal Engine with a blueprint attached to handle the tracking and controls of the camera as well as the .ini file for the hotkey bindings.

### How to import

#### *Camera Blueprint*

1. Navigate to the desired Unreal Project folder
  - a. Create an Unreal Project if none has been made yet
2. Copy the "Prototype\_1\_Tracked\_Camera.uasset" file and paste it into the project folder
3. The asset should now pop up in the Content Browser
4. Place the asset into the scene (at the desired location)

#### *Controller Bindings*

1. Navigate to the desired Unreal Project folder
2. Open under "Settings" the "Project settings"
3. Navigate to Engine > Input in the left tab
4. Click on "import" in the top-right corner
5. Navigate to the Prototype\_1\_Controller\_Input.ini file and import
  - a. If .ini file is lost, create the key bindings according to the screenshot (Figure 2)
  - b. Make sure all the names of the bindings match (Upper and lowercase sensitive)
6. (OPTIONAL) Change the key binds to another controller

### How to set up

For the tracking to work the user needs to manually set the Device ID. This ID should match the ID of the tracker or controller used as camera.

To edit this ID follow these steps:

1. Open the Prototype\_1\_Tracked\_Camera blueprint
2. Navigate to the "Event Graph" tab
3. Look for the "Get Tracked Device Position and Orientation" action (green) in the "Get Position/Orientation" box
4. Change "Device Id" from 0 to the desired ID\*
5. Press "Compile" in the top left and "Save" to save the blueprint

\*The correct device ID changes every time the headset gets restarted.

To find the correct ID the use the following trial and error method:

1. Increase the value of 0 by increments of 1 (0, 1, 2, 3, etc.)
2. Compile the blueprint
3. Press "Simulate" to simulate the blueprint

4. Head to the “**Viewport**” tab while simulating
5. Move the desired controller/tracker around (in real life)
6. Check if the camera in the viewport is moving according to the tracked device
  - a. If not: repeat steps
  - b. If it works: Stop simulating and save the blueprint

### Blueprint breakdown and controls

All the following sections are divided into boxes within the blueprint (See figure 1) to make it easier to locate.

#### *Console debugging [1]*

This box allows the user to debug an InputAction by printing a message to the console (or screen) when the desired InputAction gets triggered. Change “InputAction ChangeCamera” to the InputAction the user wants to debug

#### *Controller input [2]*

This box overwrites the default controller input to let the user use the controller from the Vive or keyboard to trigger actions.

#### *Change viewport view to camera [3]*

This box changes the current viewport and camera view from the static default camera to the tracked camera.

#### *Take a screenshot [4]*

This box (Take Screenshot) allows the user to take a screenshot of the current camera view. The user can input the desired resolution, filename, and delay into this action. (Default resolution = 1920x1080)

#### *Prevent camera input while rendering [5]*

This box (Prevent Camera input while Rendering) contains a Boolean which checks the current Game Mode the engine is set in. If the game mode is set to “MoviePipelineGameMode\_0” it will return true and prevent the tracker from influencing the virtual camera while rendering.

#### *Get position and orientation of the tracker [6]*

This box (Get Position/Orientation) receives the Id of the tracker/device used as camera and translates its position and orientation to a transformation. More info on this node can be found in “**How to set up**” section.

#### *Smoothing the camera movement [7]*

This box (Smooth It) uses a Lerp node which switches between A and B based upon the Alpha input (Like a 1 and 0 input). Whereas it switches between the two values set in “**Control the smoothness**” and adds these values to the translates values of the tracked camera.

#### *Control the smoothness [8]*

This box (Control Smoothness) contains an InputAction which upon pressing down or releasing the specified button will set the float value of “Smoothness” which is used in box [7] to a value of 1.5 and 15. These values can be changed depending on the smoothness that needs to be achieved

#### *Translate the position and orientation to the virtual camera [9]*

This box (Set Positions/Orientation) uses the values from box [6] and [7] to translate the movement of the camera to the target Camera Component (the camera in the scene). This function only works when the Boolean from box [5] returns a false output.

#### *Start and stop the Take Recorder [10]*

This box (Start/Stop Take Recorder) uses the unput from the InputAction StartTake to operate a Flip Flop node which switches back and forth between A and B. These functions then open the Take Recorder and either Start Recording or Stop Recording the movement and camera view.

#### *Cycle through lenses [11]*

This box (Cycle Lens) uses input from the InputAction CycleLens that triggers a MultiGate. This MultiGate loops through an output from 0 to 3 (Array). Each of these outputs trigger change the lens by using "Set Lens Preset by Name" of the target Cine Camera Component.

**Important:** When adding a new Lens Preset, make sure to type the exact same name of the preset in the node.

#### *Move the Virtual Camera around [12]*

This box (Movement) uses the four InputAxis assigned to scale vector A with B whereas vector B is the current position of the camera in the unreal scene (Where you placed it, not when tracking) and vector A is the desired amount you would like to move, this value can be changed towards your own likings. The "AddRelativeLocation" node then applies this movement to the camera.

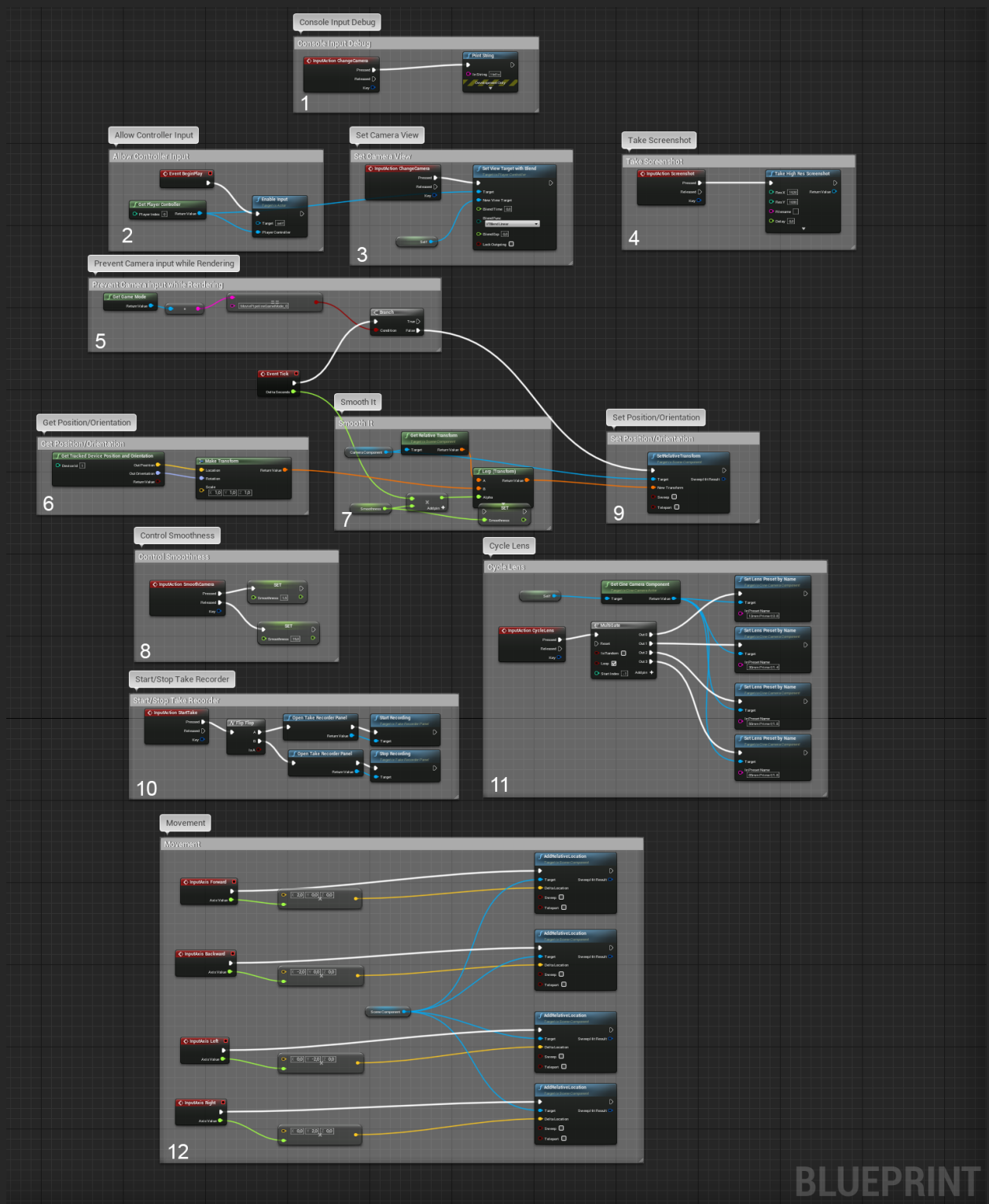


Figure 1. The layout of the blueprint

## Engine - Input






Input settings, including default input action and axis bindings.

 These settings are saved in DefaultInput.ini, which is currently writable.




### Bindings

Action and Axis Mappings provide a mechanism to conveniently map keys and axes to input behaviors by inserting a layer of indirection between the input and the behavior.

#### Action Mappings +

- ChangeCamera** + 
  -  1 Shift ☐ Ctrl ☐ Alt ☐ Cmd ☐ 
  -  Vive (R) Menu Shift ☐ Ctrl ☐ Alt ☐ Cmd ☐ 
- Screenshot** + 
  -  Vive (R) Grip Shift ☐ Ctrl ☐ Alt ☐ Cmd ☐ 
- CycleLens** + 
  -  Vive (R) Trackpad Shift ☐ Ctrl ☐ Alt ☐ Cmd ☐ 
- StartTake** + 
  -  Vive (R) System Shift ☐ Ctrl ☐ Alt ☐ Cmd ☐ 
- SmoothCamera** + 
  -  Vive (R) Trigger Shift ☐ Ctrl ☐ Alt ☐ Cmd ☐ 

#### Axis Mappings +

- Forward** + 
  -  Vive (R) Trackpad Up Touch Scale 1,0  
- Backward** + 
  -  Vive (R) Trackpad Down Touch Scale 1,0  
- Right** + 
  -  Vive (R) Trackpad Right Touch Scale 1,0  
- Left** + 
  -  Vive (R) Trackpad Left Touch Scale 1,0  

Speech Mappings

0 Array elements + 

Figure 2. Example of the default key bindings with their corresponding name provided



## Prototype 2 – Tracking and exporting as FBX

**Programs:** Unreal Engine 4, desired 3D software

**Files:** Prototype\_1\_Tracked\_Camera.uasset & Prototype\_1\_Controller\_Input.ini

### How to use

This prototype makes use of the asset from Prototype 1 to allow the user to film their scene in Unreal Engine using a tracker in real life and export the camera and its movement as an FBX to be used in the users' desired 3D program to render out.

**Important:** Follow the steps from Prototype 1 before using Prototype 2

### Preparation

To use this prototype, make sure the scene you would like to film in is ready to be sent to Unreal Engine from your 3D software. There are several ways to convert your scene to Unreal Engine depending on the software used, use one of the following guides or use a search engine to find a way to convert your scene.

Cinema4D to Unreal: <https://www.schoolofmotion.com/blog/how-to-export-from-cinema-4d-into-unreal-engine>

Documentation by Epic Games: <https://docs.unrealengine.com/4.26/en-US/WorkingWithContent/Importing/Datasmith/SoftwareInteropGuides/Cinema4D/>

Blender to Unreal: <https://epicgames.github.io/BlenderTools/send2ue/quickstart.html>

Addon download\*: <https://github.com/epicgames/blendertools>

Maya to Unreal: <https://autode.sk/3yDYzwl>

**Important:** Make sure in every 3D software to use the correct measurements!

\*To access the GitHub download page you will need to get access from Epic Games which can be acquired using the following link: <https://www.unrealengine.com/en-US/ue4-on-github>

## How to set up and use

After you have set up your scene in Unreal you have to import the camera asset and key bindings, instructions on how to import this asset can be found in “Prototype 1 – Unreal Camera Asset” (page 6).

Now you should have:

- Your scene in Unreal Engine,
- A tracked camera in the scene,
- The right key bindings imported

## Recording

### Panels

Before you press on Play or do any kind of recording, make sure to open the following windows in Unreal engine:

- Content Browser
- Take Recorder

Now follow the next steps to set up the recording mechanism in Unreal and start recording the camera movement.

### Creating or selecting the right Animation Sequence to record to

Unreal Engine needs to know which sequence it needs to play after the moment you press “Play” and which to record the translation of the camera to. To set up an Animation sequence follow these steps:

1. Go to “Cinematics” in the top bar
2. Add a Level Sequence
3. Place it in your desired folder and navigate to that folder
  - a. If you already have an animation sequence, navigate to that sequence in the content browser
4. Right click on the sequence
5. Open in Take Recorder (figure 3)

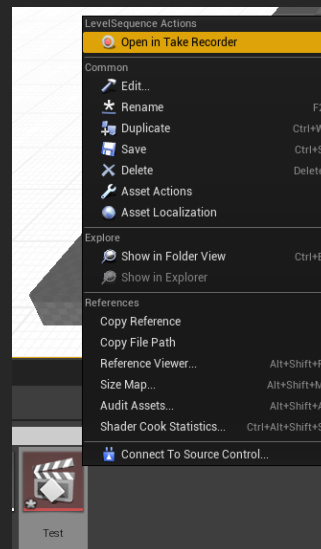


Figure 3. Example of Selecting an animation sequence and opening it in the Take Recorder

## Take recorder settings

After the right animation sequence has been selected, navigate to the Take Recorder panel, and follow these steps:

1. Add the tracked camera by selecting the camera in the viewport or world outliner
  - a. Make sure the Camera is highlighted
2. Go to the Take Recorder tab and press “+ Source”
3. Select “From Actor” and add “Prototype\_1\_Tracked\_Camera” (figure 4)
  - a. The name might differ if you have names your camera differently
  - b. **Optional:** If you select your camera in in the Take recorder tab, in its settings you can select the specific properties you would like to record from the camera
4. Open the Take Recorder settings (figure 4)
  - a. If Take Recorder Settings are not show, click on the icon in the top right of the tab with the 3 gears
5. Set the Save directory to the desired location within the project
6. Make sure to **uncheck** “Start at Current Time”. This makes it so the recording always start at frame 0.

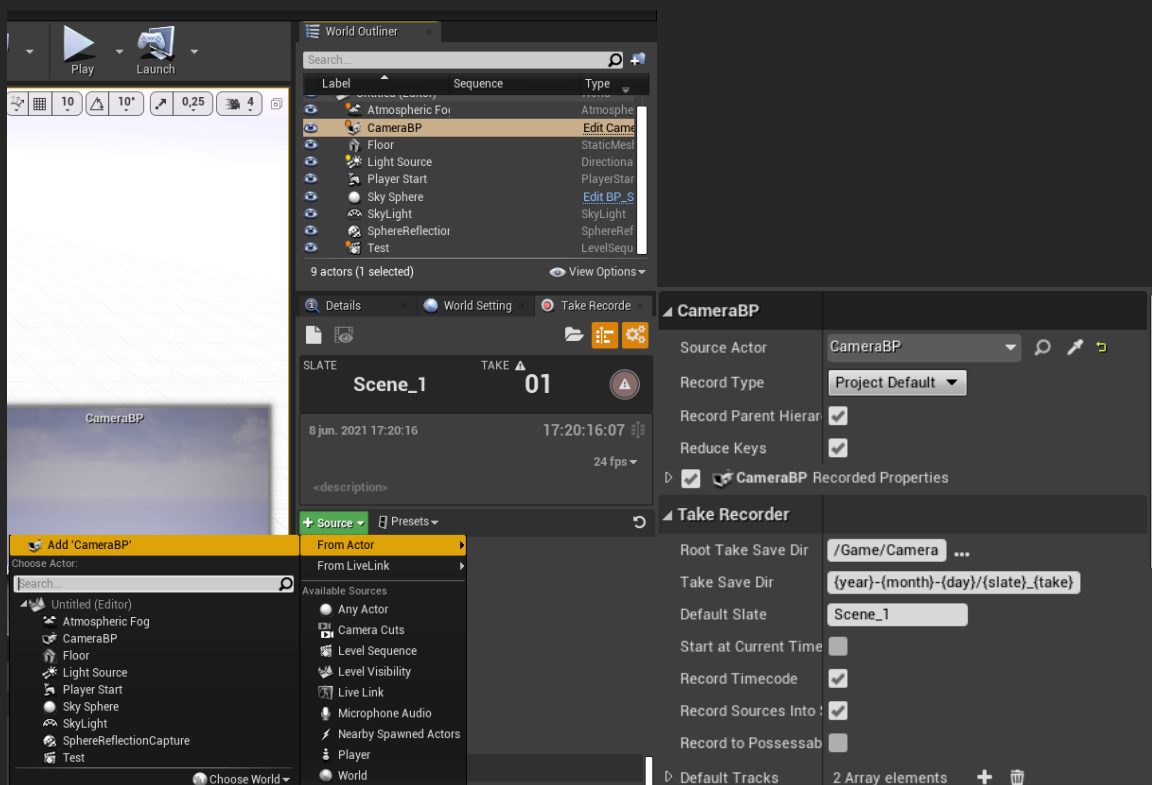


Figure 4. Example of adding the camera from the World Outliner to the Take Recorder (left) and the Take Recorder settings (right). The icon to open the settings can be found in above the Grey/Reddish round button with a triangle in it (left)

### *Placing the Camera and entering play mode*

Do this step after the Take Recorder has been set up with the right animation sequencer. With Take Recorder panel open do the following:

- Place the Prototype\_1\_Tracked\_Camera asset into the scene at the desired location
- Press "Play" in the top bar to play the scene
- Click on the viewport to allow for controller and keyboard input
- Press "1" on the keyboard or the Change View button to change the viewport view to the tracked camera

### *Camera settings and switching lenses*

If there are specific camera settings you would like to use, then click on the camera in the viewport before going into Play mode to record and change its settings in the details panel.

- To switch the lenses before or during filming **press** anywhere on the **touchpad**.

### *Moving around*

To move the camera around in the scene while recording (Not by using the tracked movement) use the **touchpad** to navigate around the scene.

### *Start and stop take*

Once you are set in place you can start the Take recorder using the key bind, pressing "2" on the keyboard. To do this manually you can press shift + F1 to take control of the mouse and manually press the record button. To stop the recording press the same button.

### *Check recorded take*

To check the animation after you are done creating all the shots follow these steps:

1. Navigate to the selected save folder that you designated in the Take recorder settings
  - a. The folder containing the animations should have the date of the recording  
(**Example:** 2021-06-07)
2. Inside you will find a Level Sequence and a folder with same name as the sequence but with \_Subscenes added (**Example:** Scene\_1\_01\_Subscenes)
3. Navigate to the correct slate and take subscenes folder
4. Open the Level sequence inside this folder
5. Play back the animations

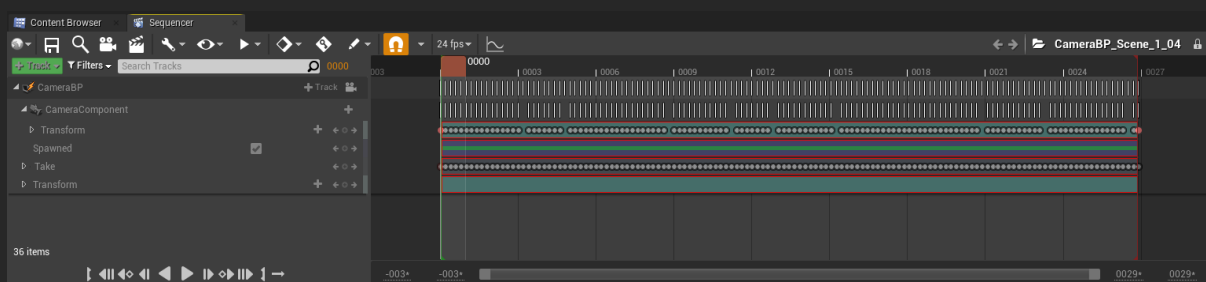


Figure 5. Example of a recorder Sequence containing the animations of the camera

## Exporting

To export the Camera movement as an FBX file follow these steps:

1. Select the sequencer track containing right camera animations
2. Right click on the camera in the sequencer (The parent of all the animations)
3. Select Export (figure 6)
4. Navigate to the desired save location and hit save
5. Choose the desired FBX Settings

After this you can import this FBX into your desired 3D software to be rendered out!

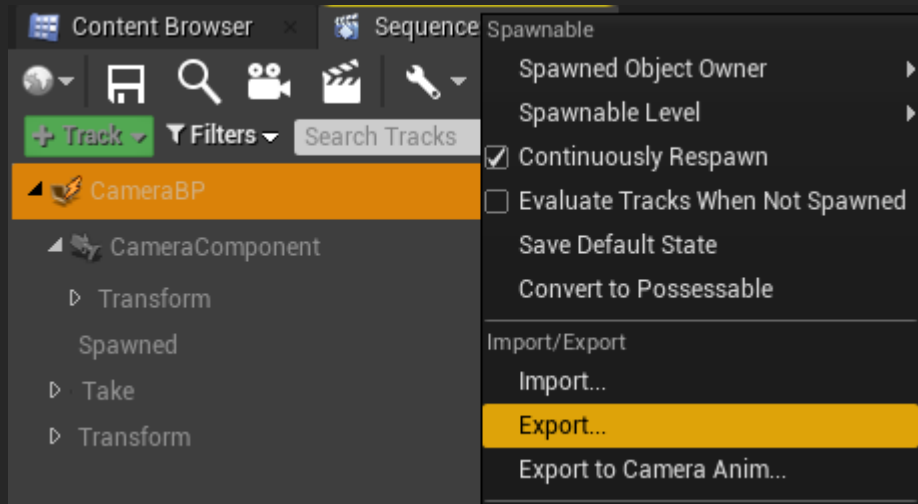


Figure 6. Example of exporting the Camera animations

## Prototype 3 – Green screen composition

Programs: Unreal Engine 5

Files: Prototype\_3\_Tracked\_Camera.uasset

### How to use

This prototype allows the user to composite real-life footage directly into a CG scene using a tracked camera and composition in Unreal Engine 5.

**Important:** This prototype has been setup using Unreal Engine 5, to make it work in Unreal Engine 4 use the asset of Prototype 1 and 2 and follow the compositing steps here.

### Preparation

Before working with this Prototype make sure to light your green screen correctly and connect your camera to the computer so it receives the footage.

Make sure that your Vive is correctly connected, and the tracker is connected as well

- To connect the tracker click on the three lines in the top left of steamVR
- Click on devices and choose “Pair Controller”
- Choose “I want to pair a different type of controller...”
- Select the HTC Vive Tracker and follow the instructions

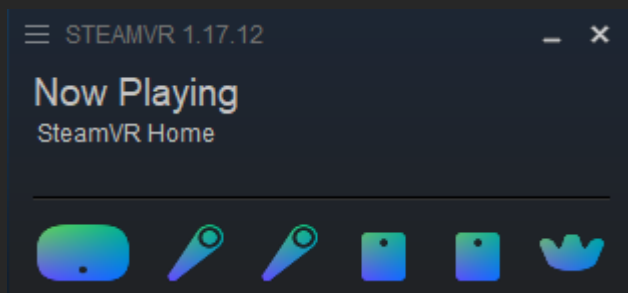


Figure 7. Example of a correctly connected HTC Vive and Tracker.

### How to setup

Launch Unreal Engine 5 and open the desired project or create a new one and make sure the correct plugins are enabled in Unreal 5, if not enable them and restart the engine

**Note:** The desired project can also be an Unreal 4 Project which will automatically be transferred

### Setting up Live Link

- Navigate to the Live Link tab in top left corner
  - o If it is not present, navigate to “Window”
  - o Go to Virtual Production and choose Live Link
- In the Live Link tab add a source by clicking the “+ Source” button
- Choose LiveLinkXR Source and tick all the boxes
- Proceed by clicking on “Add” (figure 8)

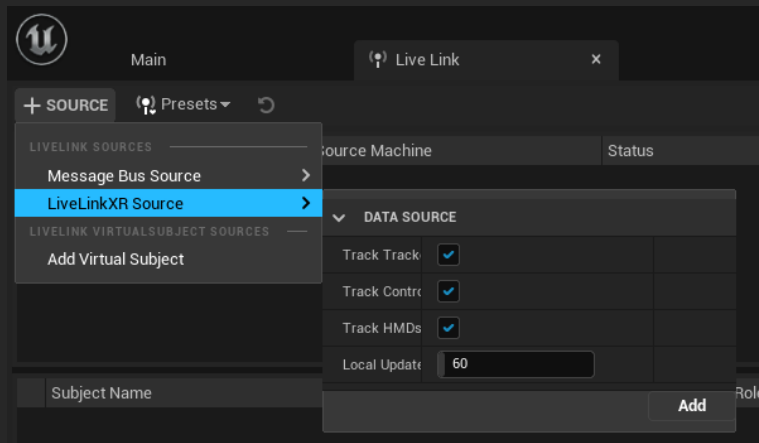


Figure 8. Adding the LiveLinkXR Source to the Live Link

- In the Live Link tab you should now see your HTC Vive setup (Starting with “SteamVR..”)
- Check in the bottom window if the tracker is recognized and working (figure 9)
  - o The dot at the end of the row shows if a device is active (green) or inactive (yellow)
  - o If the tracker is not showing up, remove the source, turn on the tracker and redo the steps

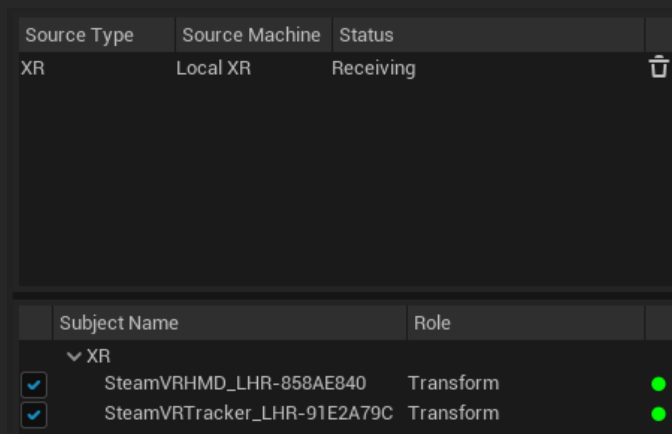


Figure 9. Example of a connected HTC Vive source and the visible Headset and Tracker

### Getting the camera live feed input

- Open the content drawer
- Right Click anywhere in the window
- Navigate to “Media” and create a “Media Player”
- Check the box to create a “Video Output MediaTexture Asset” and hit Ok (figure 10)

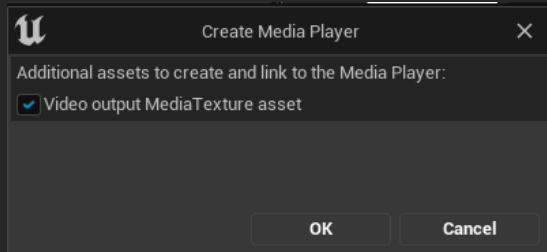


Figure 10. The pop window after choosing to create a Media Player

- Open The created media player
- Click on the folder in the top left
- Navigate to “Video” and select the video input you want to use (figure 11)

The feed from your camera should now be shown on the screen and a Media URL has been created (figure 12)

- Copy and save the Media URL to a clipboard

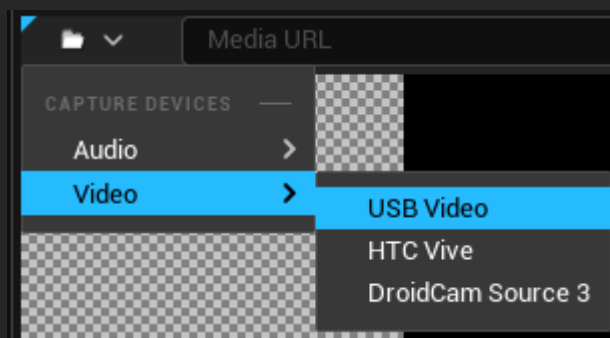


Figure 11. Example of choosing a Media capture device

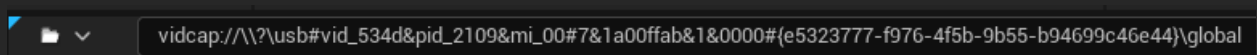


Figure 12. Example of a Media URL



Import the Prototype\_3\_Tracked\_Camera.uasset by copying it into the content folder of the project

- Navigate to Prototype\_3\_Tracked\_Camera in the content drawer
  - o This should be in the same folder you copied it into
- Drag the Prototype\_3\_Tracked\_Camera pawn into the scene at the desired location
- Click on the “+ Create” button in the viewport window
- Navigate to Cinematic and create a “Cine Camera Actor”
- Select Prototype\_3\_Tracked\_Camera in the scene or World Outliner tab
- Open details panel and under the Default Section change the following variables (figure 14):
  - o Camera = The added Cinema Camera which you would like to use in the scene
  - o Tracked Delay = The number of frames/ticks that is between the live footage and footage shown on screen
  - o Live Link Subject = The tracker/controller you like to use to track the camera with
  - o Media player = The media player you have created
  - o FPS = The FPS at which the engine should run at (This should be equal to the same FPS as your camera feed)
  - o Camera URL = Insert the earlier copied Media URL from the clipboard or Media Player

#### *Calculating the delay*

To find/calculate the delay between the camera footage and camera feed in Unreal you can follow these steps:

**Important:** This can be done on the same PC, but it is recommended to open Unreal Engine on another PC and match the footage of that Timecode with the one on the main PC

#### **Do this on both computers (if using two computers)**

- Navigate to “Edit” in the top left corner
- Open “Project Settings” and navigate to “General Settings” under ENGINE
- Scroll down to the “Timecode” section
- Set the “Generate Default Timecode Frame Rate” to the same framerate as your camera/project and close the project settings window
- Navigate to “Window” in the top left
- Under Virtual Production choose “Timecode Provider”
  - o Make sure the Timecode Provider is visible somewhere on your screen

#### **Do this on the main computer**

- Open your Media Player playing the camera feed
- Aim your Camera to the Timecode Provider on screen
- Hold down your mouse somewhere on the grey area (Marked red in figure 13 left) to freeze the footage and Timecode
- Calculate the difference between the Timecodes (Example in figure 13 =  $14 - 05 = 9$  frames)
- Check the delay by clicking on the white button next to the Timecode with an arrow pointing down
- Set the “Frame delay” to your calculations and check again (Figure 13 right)

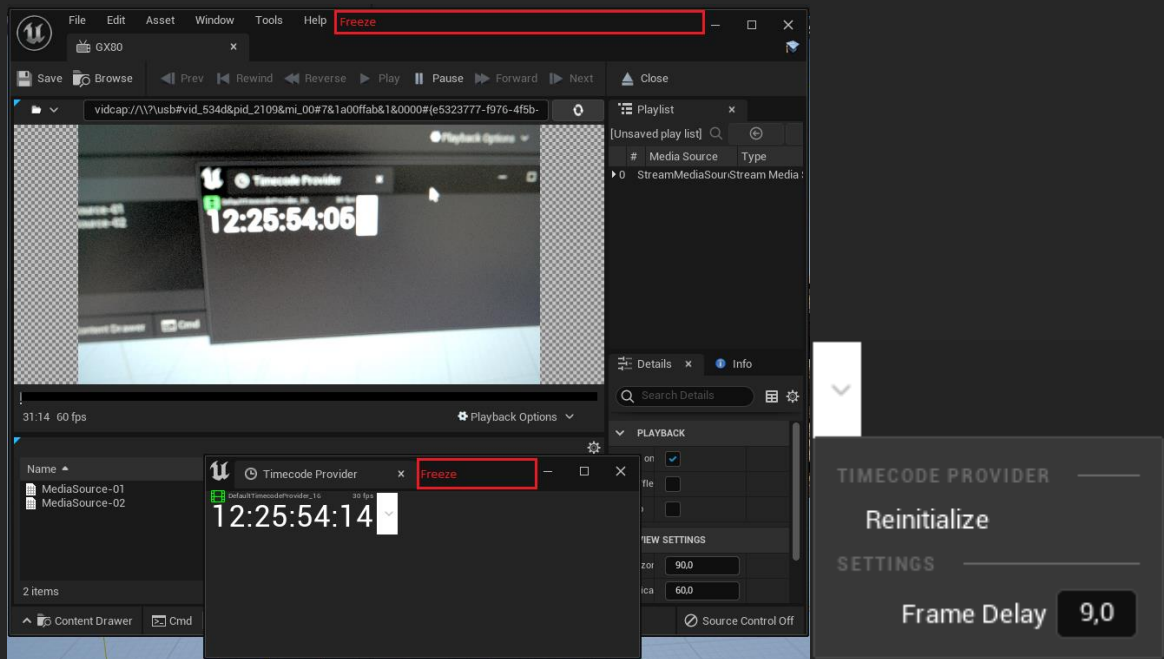


Figure 13. Example of timecode panels showing in the Engine and the marked area to freeze both frames (left) and the frame delay input to match the timecodes after calculation (right)

▼ DEFAULT		
Tracked Delay	<input type="text" value="0"/>	↩
Camera	<input type="text" value="None"/> 🔍 ↗	
Live Link Subject	<input type="text" value="None"/>	
Media Player	<input type="text" value="None"/> 🔍 ↗	
FPS	<input type="text" value="0,0"/>	
Camera URL	<input type="text"/>	

Figure 14a. The variables in the detail panel of the tracked camera asset

▼ DEFAULT		
Tracked Delay	<input type="text" value="11"/>	
Camera	<input type="text" value="CineCameraActor2"/> 🔍 ↗	↩
Live Link Subject	<input type="text" value="SteamVRTracker_LHR-91E2A79C"/>	↩
Media Player	<input type="text" value="GX80"/> 🔍 ↗	↩
FPS	<input type="text" value="30,0"/>	↩
Camera URL	<input type="text" value="vidcap://\\?\usb#vid_534d&amp;pid_2109&amp;mi_00#7&amp;1a00ffab&amp;1&amp;0000#{e5323777-f976-4f5b-9b55-b94699c46e44}\global"/>	↩

Figure 14b. Example of all the variables with their required input

### Creating the composition

- Open the Composure Composition window in the top left (Next to the World Outliner)
  - o If it is not there, navigate to "Window" in the top left
  - o Go to Virtual Production and choose Composure Composition
- Right click in the window and choose "Create new comp"
- Select "Empty Comp Shot" to create a new composure and name it however you like (figure 15)

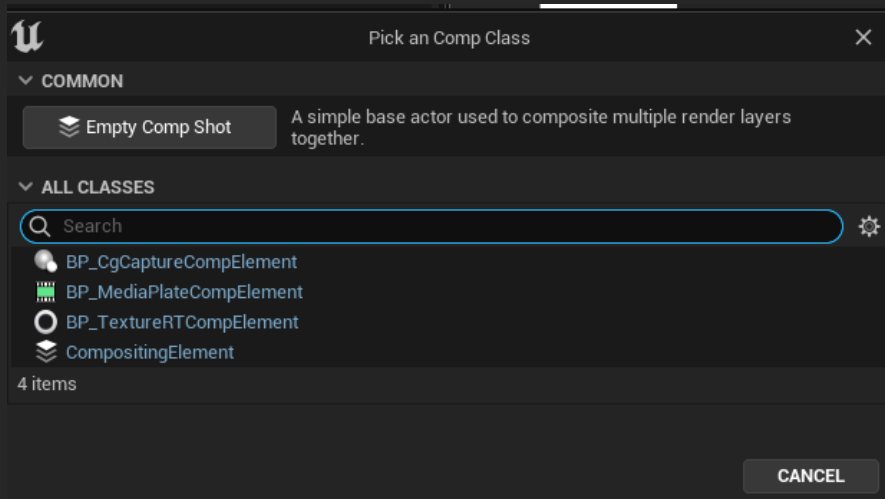


Figure 15. The pop-up window when creating a new composure

- Right click on the created comp and choose "Add a layer element"
- Add both a "Media plate" and "CG Layer"

Expand the created composure to reveal the Media Plate and CG Layer (figure 16)

- Select the Media plate
- In the Details panel navigate to Composure
- Under composure open the Inputs section
  - o Add an array element if there are 0 array elements
- Makes sure that the "MediaSource" OR "InputPass\_01" is a "Media Texture Composition Input"
  - o They are both the same but might have one of both names
- Open the MediaSource and apply the following changes:
  - o Enabled should be checked
  - o Media Source = The media texture you created with the media player
    - This source should have the same name as the media player

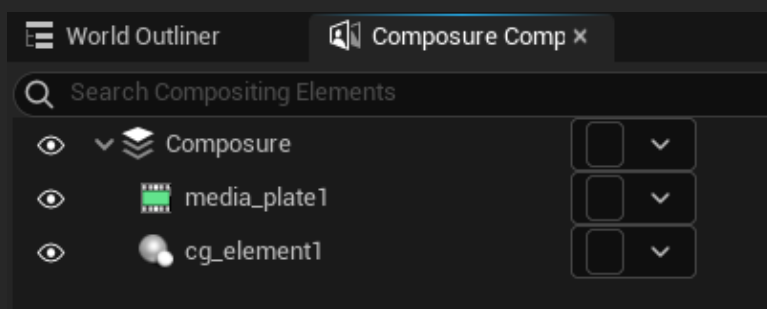


Figure 16. Example of a composure with a Media Plate and CG Layer

### Removing the green/blue screen

- Open the Media Plate layer element
- In the details panel navigate to “Transform/Compositing Passes”
- Expand “Transform Passes”
  - o There should be 3 array elements. If not, create the following 3 elements (figure 18):
    - Multi Pass Chroma Keyer
    - Multi Pass Despill
    - Compositing Element Material Pass
- Expand the Multi Pass Chroma Keyer (**Default name:** “Chroma Keying”) (figure 17)
  - o Set the Material to M\_SinglePassDiffColorKeyer
  - o Add an array element to the Key Colors (of none are present)
  - o In the Key Color Index click on the black color
  - o In the Color Picker window choose the green screen color that you are using

**Quick tip:** An easy way to get the right color is to open the color picker if the Key Colors Index and open the Media Player playing the video feed from the Content Drawer. Then in the color picker, choose the eye dropper and eye-drop the color of the green screen straight from the media player

- Expand the Multi Pass Despill (**Default name:** “Despill”)
  - o add an array element to the Key Colors (of none are present)
  - o In the Key Color Index choose the green screen color that your using
- Expand the Compositing Element Material Pass (**Default name:** “Erode”)
  - o Set the material to ErodeAlpha (If its not set)
  - o Play around in the material parameters to get the right Erode value, this might differ for everyone
    - default ErodeKernalSize = 0
    - Default ErodeNumSample = 8

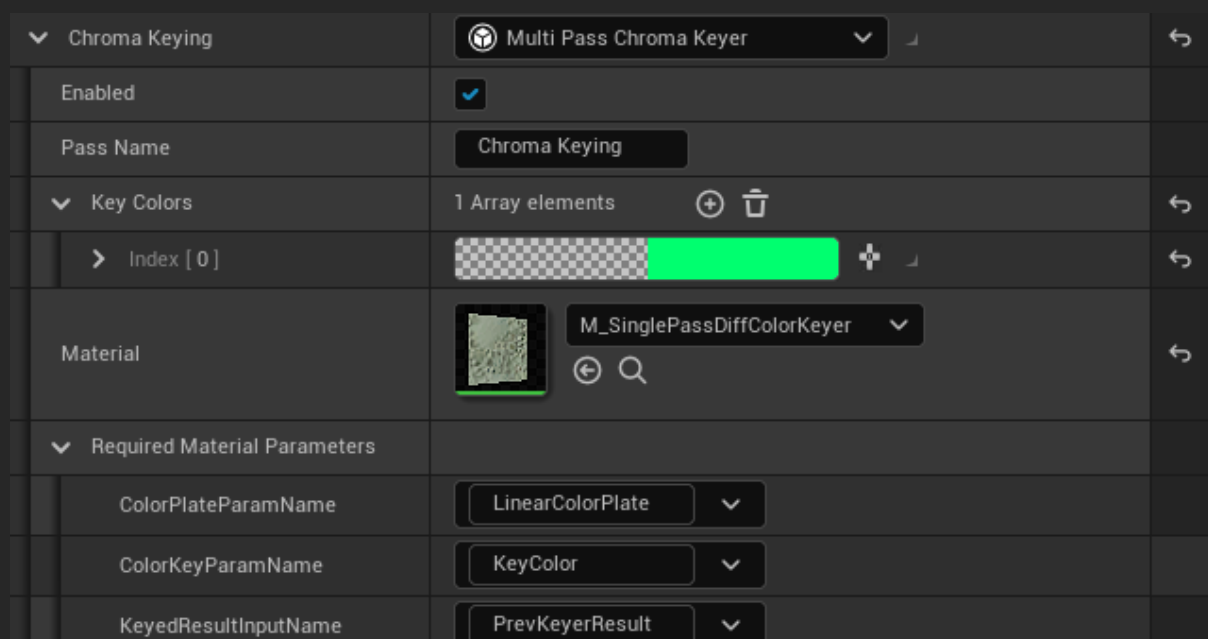


Figure 17. Example of a correctly set up Chroma Keying pass

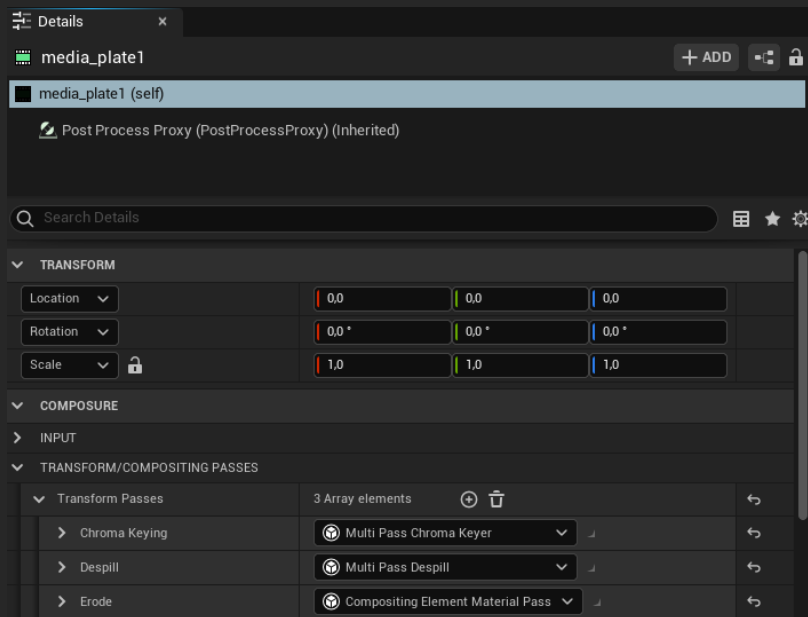


Figure 18. Example of a Media Plate containing three Transform Passes

#### *Adding the CineCamera to the composition*

- Open the CG element in the composure
- In the Details panel navigate to “Composure”
- Expand the “Input” section
- Set the “Camera source” to Override
- Set “Target Camera Actor” to the Cine camera actor you are controlling/using (figure 19)

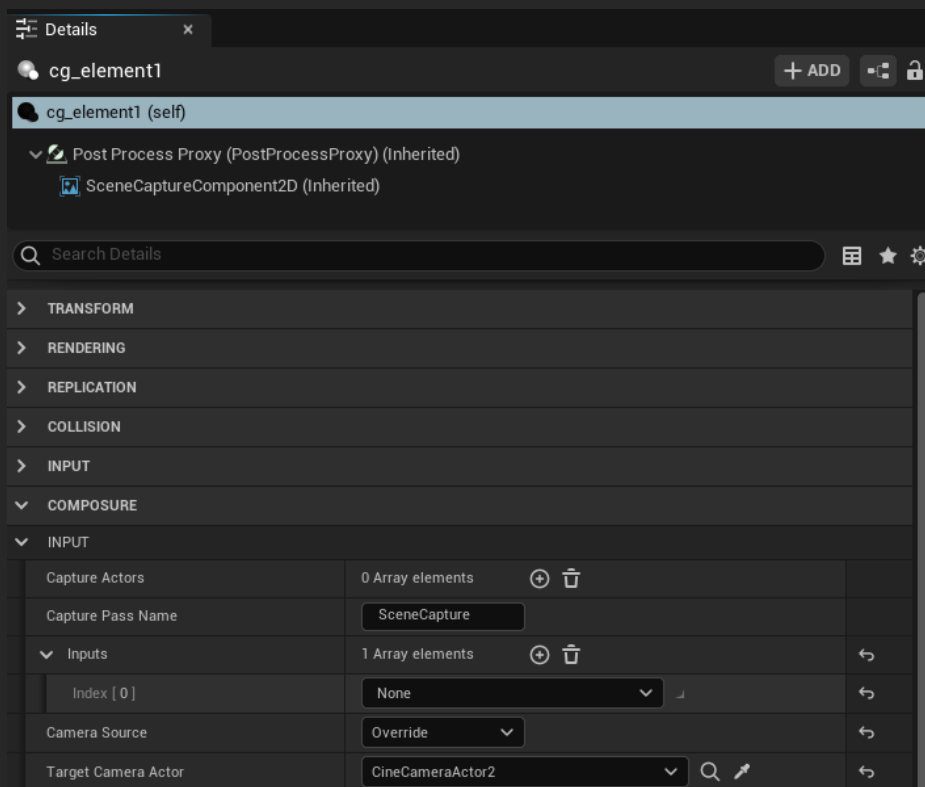


Figure 19. Example of a CG Layer (cg\_element1) with a Cine Camera Actor selected

### Creating composure material

- Open Content Browser
- Right click anywhere and create a new Material
  - o Name it something like “ComposureMaterial” to make it recognizable
- Open the created material
- Select the ComposureMaterial node (Main node)
- In the Details panel (Bottom Left) set Material Domain to “Post Process” (figure 20)
- In the Material Graph window right click anywhere and create two “TextureSampleParameter2D” nodes
- **Important:** Give these nodes the exact same name as the created layer elements (media\_plate1 and cg\_element1, in this case (Uppercase sensitive)
  - o Renaming them can be done under “Parameter Name” in their Details panel
- Create an “Over” node
- Put both the RGBA outputs of the Texture nodes into the Over node
  - o Make sure the Media Plate (A) goes Over the CG Element (B)
- Put the RGBA output of the Over node into the Composure material’s “Emissive Color”
- Apply and Save

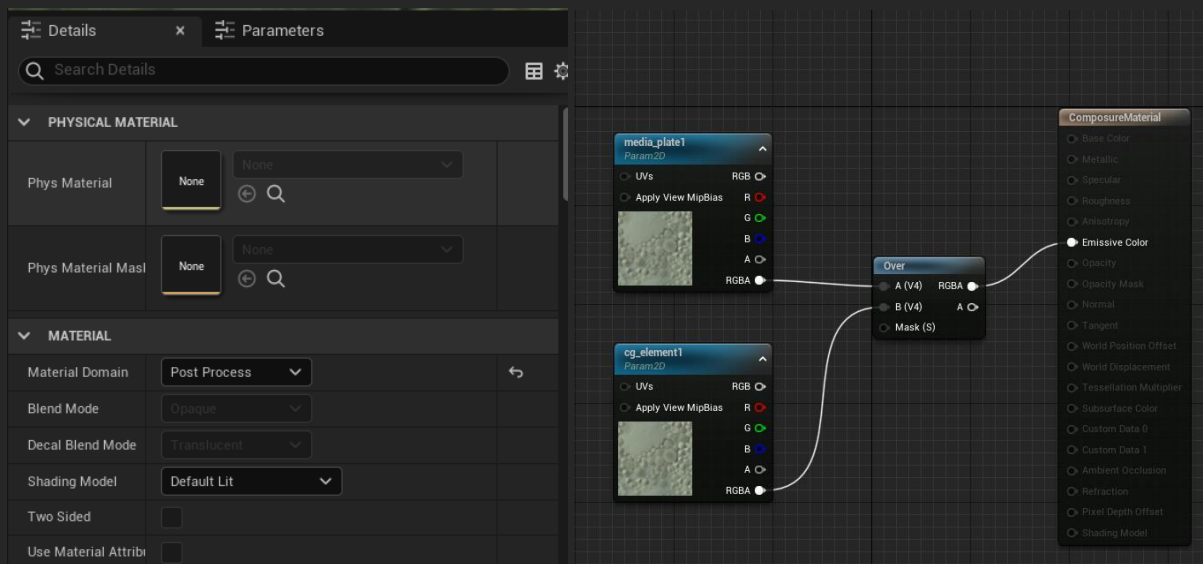


Figure 20. Example of a Composure Material setup where the main node’s Material Domain is set to “Post Process” (left) and the composure layers are combined using an Over node (right)

### Adding the composure material

- Open the Composition in the Composure Compositing panel
  - o Composition is the parent of the Media plate and CG Element
- Expand the “Composure” section in the Details panel and proceed to Expand the “Transform/Compositing Passes” section
- Add an Array Element to the Transform Passes (if none are present)
- Open the Array Element (**Default name:** TransformPass\_0)
  - o Make sure this Pass is set to “Compositing Element Material Pass”
- Set the Material to the Composure Material Created earlier (figure 21)

### Outputting the composition

- Navigate to the Output section in the Details Panel of the Composure
  - o Output can be found right under the “Transform/Compositing Passes” section

- Add an Array element to the “Outputs” (if none are present)
- Set the “OutPutPass\_0” to “Player Viewport Compositing Output”

To save/render the composition follow the next steps:

- Navigate to the Output Section of the composure
- Create a 2<sup>nd</sup> array element
- Set the second OutputPass (OutputPass\_1) to “EXRFile Compositing Output”
- Expand the second OutPutPass and set the following settings:
  - o Output directory = The folder to which it saves
  - o Filename = The naming of the files created
  - o FPS = The FPS of the outcome, should be same as the camera’s FPS
- Under “Render Resolution” the output resolution can be set. (figure 21)

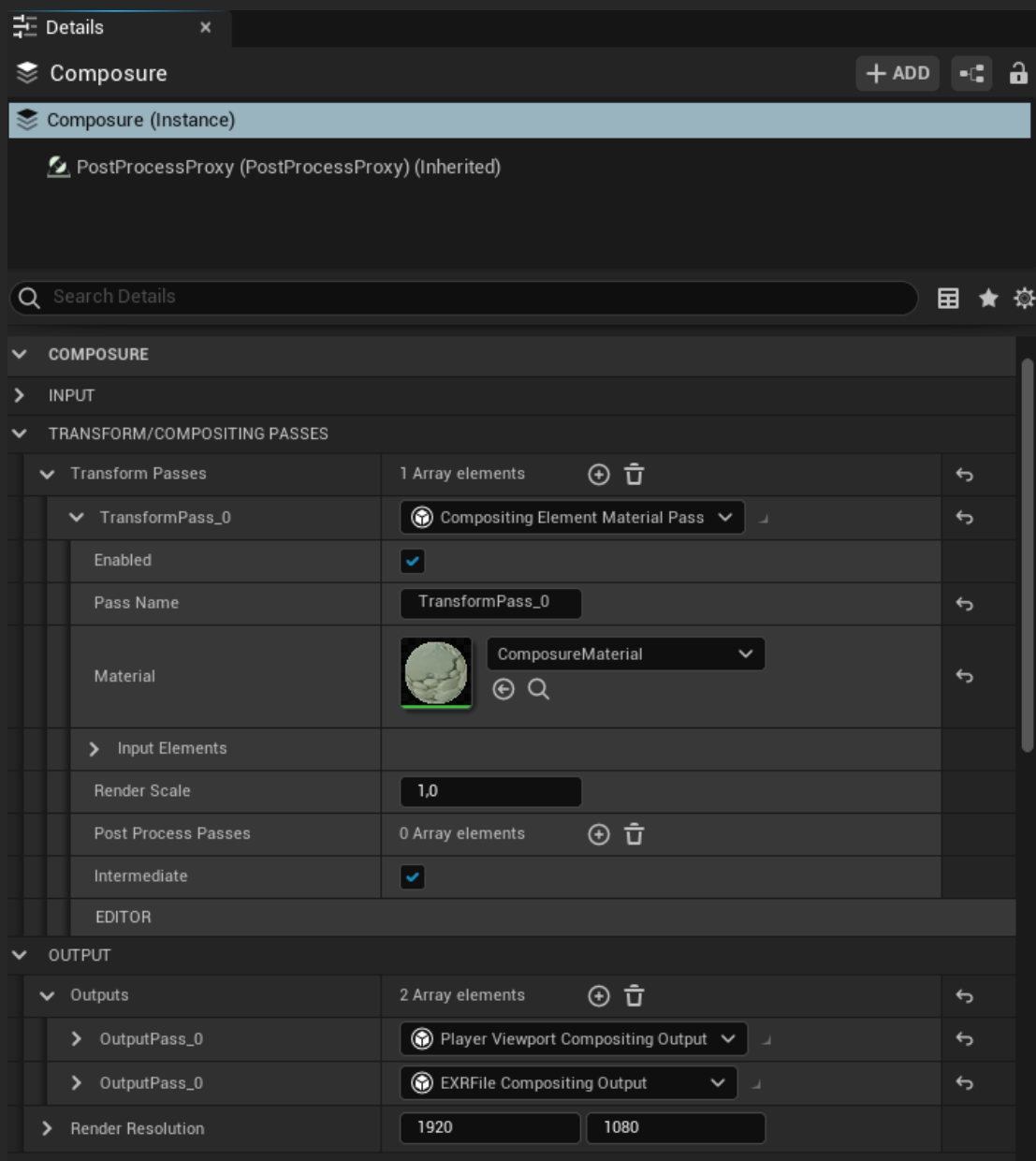


Figure 21. An example of a composure with its material set to the ComposureMaterial and the outputs set to viewport AND EXRFile

## Troubleshooting information

If you encounter any problems while setting up one of the prototypes it is recommended to troubleshoot the following:

- Check if you have the correct plugins enable
- Check if Unreal Engine, HTC Vive and SteamVR are up to date
- Check if Unreal has changed a certain documentation

Documentation for both Engines can be found underneath:

[Unreal Engine 4](#)

[Unreal Engine 5](#)



Appendix 2. A video showcasing the outcome render of using prototype 2:

<https://youtu.be/KtiMDk66Uzw>



Figure 8. The blender scene and the same scene imported into Unreal Engine 4 to film.

### Appendix 3. Camera setup iterations with the tracker

