Setting Up a Broadcast Ready Virtual Studio with Position, Focus and Zoom Tracking in the Real-time Engine Unity3D

Adams Jente
AP University College
2000 Antwerpen, Belgium
jente.adams@gmail.com

Bujoková Pavlína
Tomas Bata University in Zlín
760 01 Zlín, Czeck Republic
pavlina.bujok@gmail.com

Valéry Dorian
École Nationale d'Ingénieurs de Tarbes
65 000 Tarbes, France
d.valery12@gmail.com

Amouzegh David
Université de Valenciennes
59313 Valenciennes, France
david.amouzegh@wanadoo.fr

Kastel Thiemo
St. Pölten University of Applied Sciences
3100 St. Pölten, Austria
thiemo.kastel@fhstp.ac.at

Kovaleva Polina
UAS Ansbach
91522 Ansbach, Germany
kpolina007@gmail.com

Saup Sonja
UAS Ansbach
91522 Ansbach, Germany
kishaja1337@gmail.com

Sanmartín Marco Laura
Universitat Politécnica de Catalunya EPSEVG
08800 Vilanova i la Geltrú, Spain
lau_sanmar4@hotmail.com

Trim Robert
Utah Valley University
Orem, UT 84058, USA
trimro@uvu.edu

2016 International Conference on Culture & Computer Science (ICCCS), Namibia University of Science and Technology (NUST), Windhoek, Namibia
ISBN 978-99916-903-0-8.

Abstract
Virtual Reality is becoming more and more popular among all areas in multimedia. Using Virtual Reality in live production can make a significant difference in the content that can be created. There are two primary approaches to create a Virtual Reality environment: static and dynamic. In the dynamic system the camera can change position, zoom and focus with the virtual environment being synchronised with the camera and set to create a seamless visual world. The authors explored the possibility of making an inexpensive dynamic virtual system based on the game engine Unity3D which is the first one of its kind. In the project the authors were able to: track the zoom and position of the camera, chroma key the live set and connect this data to the virtual environment within Unity3D in real time. Through team effort, including project management, marketing strategies, technical setup, and creative input a live show was produced with our dynamic system.

Author Keywords
Virtual Studio; Tracking System; Unity3D; Real-time; Broadcast; TV Live Show; Focus & Zoom Tracking

ACM Classification Keywords
I.3 [COMPUTER GRAPHICS]: Three-Dimensional Graphics and Realism; I.3.7 [Three-Dimensional Graphics and Realism]: Virtual Reality

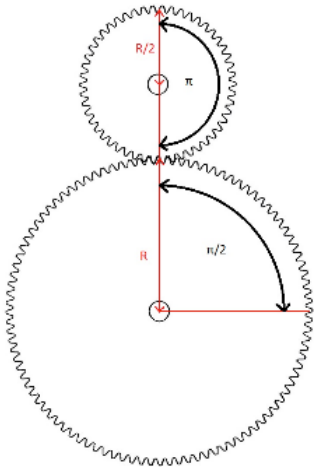

Figure 1: Alignment of the gear on the camera lens (below) and the gear on the sensor (above)

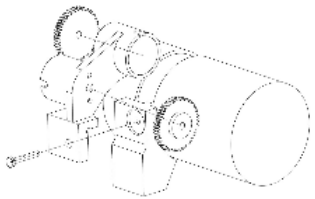

Figure 2: Sensor mounting frame position on the camera lens

1. Introduction

Virtual Reality is one of the fastest growing areas reaching a broad spectrum of users from various fields, through the implementation of this technology. This paper introduces a unique approach to the application of a dynamic virtual studio system based on the Unity3D game engine. The project focus is to prove the working connection between a physical and a virtual camera, and assure its performance for live production. Available resources for this specific project were financial resources and technical equipment from the St. Pölten University of Applied Sciences, which are tracking system, PC-Workstations, HD-Cameras and a HD-Video Studio environment as well as the workforce of seven students. The project client is the university itself.

The research was organised in the context of the European Project Semester (EPS) hosted by the St. Pölten University of Applied Sciences, Austria. The project team consists of seven students from different countries (France, Germany, Czech Republic, Spain, and Belgium) and a team of supervisors. This paper discusses the general approach of virtual studios for live broadcasts and explains the resulting solution for the HD-Videostudio of the St. Pölten UAS.

2. Technical approach and setup

2.1 Lookout on the market

Why exactly is a game engine a suitable software for dynamic live broadcasts? The software for live dynamic virtual studios require a substantial financial investment. Therefore a more financially friendly solution to this is the use of a game engine. Games and videos have a lot in common, both show the story in moving pictures and both are made of a certain amount of frames per second.

One of the goals of the project was to act within a small budget. A free and well documented game engine is preferable. Unity3D does fit these needs and is one of the most versatile and popular engines out there. It is easily expandable with plugins and it supports three main programming languages; with the two most popular being C# and JavaScript.

2.2 Sensor Mounting Frame

In order to connect the pictures of the virtual and physical cameras dynamically two major things have to be tracked on the physical camera to reproduce the changes in Unity3D; the position of the camera and the the zoom and focus. When the operator shifts these values on the physical camera these changes will be applied to the virtual background. The connections between the virtual and physical camera are:

- Sensors tracking zoom and focus.
- IR cameras tracking the rigidbodies.

Physical data communication to the Unity3D engine is provided through Wi-Fi and the Ethernet connections, between the cameras and these devices.

- 2.2.1 Tracking the zoom and focus

In order to track the zoom and focus two sensors are required. Sensors used in the project are both rotary encoders. These sensors are mounted to the camera equipped with gears that mesh with the gear on the lens.

The first step in creating the mounting frame was to take measurements of the lens, the camera and the surrounding assets. With this information the design of the mounting piece for the sensors was created.

The main function of the sensors is to precisely track the rotation of the rings on the lens. Therefore the gears mounted on the sensors needed to be as small as possible, so the

range of use of the sensors would be bigger. The bigger the range of use, the better the resolution of the tracking.

The diameter of the gears is half of the diameter of the lens. So when the lens turns 90º, which is the maximum rotational range, the sensors will turn for 180º. So the final ratio is 2:1 (Fig. 1). In the final setup the sensors track the zoom and focus and output precise data (Fig. 2).

- 2.2.2 Tracking the position and rotation

The camera position and rotation is tracked by four IR cameras which track the changes in the placement of rigidbodies. The rigidbodies have four reflecting spheres and have to be attached to the camera. To have the best position of the rigidbody and allow easy attachment on the camera, a slider was deemed as the best solution. The slider (Fig. 3) is mounted on the highest position, on the handle, in order to make the rigidbody visible for all IR cameras.

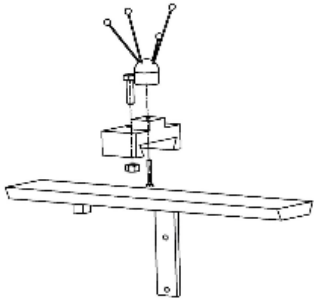

Figure 3: Placement of the rigidbody on its slider

- 2.2.3 Manufacturing

All the pieces were designed with a 3D CAD software and were produced with 3D printers (XYZ Da Vinci 1.0 Junior and Witbox 2). The used material is PLA. 3D printers allow to produce fast prototypes; it is useful for testing purposes.

2.3 Unity3D

The system uses Unity3D v4.6 pro or newer to display the virtual background and to calculate the position, rotation, zoom and focus of two virtual cameras. Said system is using the data of the position tracker and rotary encoders that track the position, pan, tilt, pedestal, zoom, and focus of the corresponding physical cameras. Unity3D is a game engine and is primarily used by indie game developers.

- 2.3.1 Position tracking

The IOtracker system with four infrared cameras and rigidbodies is used for position tracking. The rigidbodies reflect the infrared light emitted by the infrared cameras, and the software calculates the position, pan, tilt, pedestal and dutch tilt of the rigidbodies. Due to the unique shape of the rigidbodies, the system can recognize them. The position is calculated by the IOtracker software and sent in an OSC packet over Ethernet. The OSC packet contains nine fields for: frame number, rigidbody, the xyz coordinates, yaw, pitch, roll and W, a value that is in relation to the yaw, pitch and roll.

In Unity3D a C# script listens for the OSC packet of the tracking system. Another C# script triggers an event every time an OSC packet is received. The data from the OSC packet is used to align the position, pan, tilt, pedestal and dutch tilt of the virtual camera to the physical camera. The C# scripts work with two threads for the position tracking which are two processes which are executed simultaneously. Unity3D changes the position only each frame because of its inner workings.

The distance has to be synchronised, so that the virtual camera moves inside the scene relatively to the distance of the physical camera. As every scene does not have the same dimensions, the dimensions provided by the tracking system should be divided by a value that makes the movement of the physical and the virtual camera match.

- 2.3.2 Zoom and focus tracking

The zoom and focus are tracked by RE22 rotary encoders, two encoders for each camera. The encoders are connected to an MSX-E1701 interface which reads the sensors and transmits the data over Ethernet using the SOAP protocol. Unity3D is using a C# script to send a SOAP packet encapsulated in a http packet to the interface and listens for the response. The script loops twice for each camera to track the zoom and focus steps taken by the physical camera, and changes the zoom and focus for its corresponding

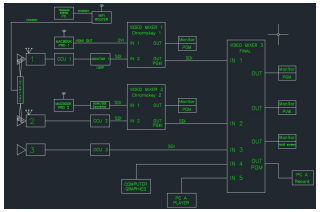

Figure 4: Studio setup including all components

Figure 5: Dimensions of virtual studio and set

virtual camera accordingly. The zoom is changed by moving the camera forward and the focus is changed by the depth of field value.

The script moves the virtual camera forward in little steps, which have to be executed multiple times to match the zoom distance in the corresponding cameras. The focus depends on the lens of the camera and the lens has to be measured before the synchronisation process.

2.4 Production

In order to easily integrate the created electronic and mechanical technologies the control room required a unique setup. Key to the project success, with the programming part done, was to figure out different ways to set up the control room in order to avoid signal delay between the cameras.

The data was distributed between two computers and transmitted to Unity3D. Once the data was collected, the background scene had to be adapted to the position, rotation and the field of view of the studio camera. A big challenge was the correct lighting of the green box due to the different props. It was important as the zoom was providing close-ups. A Unity3D plugin allowed the use of the SDI output in the PC, simplifying the workflow. Chroma keying was processed in an external PC with four SDI inputs and was then sent back through two SDI outputs to the video mixer.
Once all the signals were synchronized it was important to route the cameras to their dedicated backgrounds and setup the outputs to provide a single picture in the main video mixer. An external video mixer was used to chroma key the picture of the third camera, which was the wide shot static camera (Fig. 4).

The final setup in the video mixer included the three combined signals on input 1, 2 and 3, belly bindings (fill and key) on input 5 and 6 and a PC with a video player on input 4. This setup allowed a simple workflow with the possibility to change signals (Fig.4).

2.5 Graphics

To visualize the effectiveness and correct functionality of the Unity3D script and the tracking system as well as to test the performance in a live broadcast, a virtual background was created. For the creation of the environment 3D model the animation software Cinema 4D R17 was used.

- 2.5.1 Creating the set

The virtual background was created according to the theme of the test live show, which was "Lost in Space". As usually, television live sets are created in a simplistic fashion, this project required a unique and uncommon shape.

It was important to construct the set within the same dimensions as the provided studio space. The green box is roughly 8m wide, 6m long and 7m high (Fig. 5). The constructed studio set has a larger length for easier camera movement and a smaller height for the feeling of an enclosed room. After the model was finished, textures fitting to the theme of the live show were created. This texture required a metallic material with no reflections and shadows as the virtual studio cannot display these attributes of the physical set. A matte and rough surface fits these conditions. To break up the shadows and reflections even more, a bump map was added. Afterwards the textures were lit out with consideration of the physical set, were then rendered and connected to their corresponding polygon meshes (objects). It was of great significance to have the light temperature, intensity, range and diffusion match to the possibilities of the physical set while being authentic within the theme of the show.

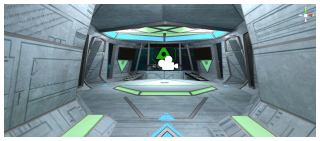

Figure 6: Virtual environment in Unity 3D

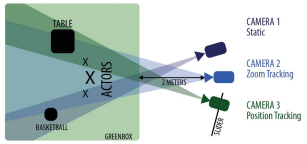

Figure 7: Positioning of the cameras and props on set

Figure 8: Shot from recorded live show

- 2.5.2 Further work with Unity3D
During the baking process a normal map, light map and base colour were exported as 5000 x 5000px. The resolution was high so the details of the map display well. Within Unity3D these three image files (TIFF) were applied to the "Bumped Diffuse Shader".

It is important that the pivot point positions and rotations, and the UVW-coordinates are imported correctly as well. The FBX-file type keeps such information in addition to the hierarchy of the objects (Fig 6).
Lastly, a virtual camera was created with the field of view matching the corresponding physical camera.

2.6 Proof of performance
In order to showcase the functionality of the different components under realistic circumstances a demonstration was necessary. Therefore, a live show was deemed the best option. In preparation, a TV show script was written, intro videos, short 3D animations and live graphics with the VizRT software were created as well as actors were casted. The shoot took place in the TV Studio of the University Of Applied Sciences, St. Pölten (Fig. 7).

The specially designed sensor mounting frames, rigidbodies and the interface for the zoom tracking were mounted to the physical cameras.
As the position of the camera varied a lot due to deviations of the tracking system and the position of the rigidbodies, the field of view (FOV) was adjusted manually for each camera, depending on its position and angle. As a result of that the FOV on each of the three cameras was slightly different. For the live show, video textures were added to the modeled screens of the studio. These were connected to a short C# script for display during the live show.

As described above (Chapter 2.4) the signals from each camera are synchronised through various computers and video mixers to create the chroma key. Then these signals are combined with the virtual background and sent to the final video mixer, ready for broadcast.

3. Results and discussion
Recording the first live show was a successful test (Fig. 8) with results which are discussed in the following paragraphs. These results can be applied in future broadcasts and other studio productions.

3.1 Sensor Mounting Frame
The preparation process for creating the sensor mounting frame was compounded by the fact that the exact measures of the lens were not known and were measured manually. The manufacturer could not supply the exact amount of teeth on the gear either. Initially, in the mounting frame two spaces for sensors were designed, each for zoom and focus. The recorded show only utilised the zoom sensor; therefore the holder is still applicable for future recordings and expandable for the focus sensor. The mounting frame was specially designed for the cameras of the UAS St. Pölten. Yet as long as the camera lens and remote control for the zoom and focus are similar, this frame can be used.

3.2 Unity3D script
During the recording the zoom worked best with a steady slow speed, faster zooming disconnected the virtual background from the physical. To fix this issue, the zoom script can either be adjusted to be faster or fill in the missing steps automatically.
Due to malfunctioning sensor cables, limited resources and time there was a problem tracking the zoom on multiple cameras with the interface. A circuit, using a microcontroller and a Wi-Fi module which sends the data from the sensor to Unity3D, can be used in addition to the interface.

Furthermore, not all infrared (IR) cameras had the mounted rigidbodies in their field of view. Due to that the the pan and tilt were not tracked correctly. This was likely caused by the small tracked space and the amount of equipment and operators on set. Therefore only one camera had its position tracked.For better tracking a wider tracked range with extra IR cameras is necessary.
In addition to that a gimbal lock occurred during the rotation of the cameras so the script was adjusted to use quaternions. Due to the limited functionality of Unity3D with quaternions and the engine translating Euler angles to quaternions internally, Euler angles were chosen at first.Due to the limited time and resources there the development of the focus tracking was deemed unnecessary for the first prototype. In order to synchronise the focus of the physical and virtual camera a calculation heavy process was inevitable, so the hardware requirements were very high in order to achieve realistic results.

3.3 Studio setup
Unity3D does not initially support HD-SDI while the video studio is not equipped to use other means of interfaces to transfer data to the video mixer. During the first tests an external video mixer was connected to the video mixer of the studio with a DVI interface; this resulted in synchronisation problems. An HD-SDI plugin for Unity3D was more desirable to solve the synchronisation problems.

3.4 Virtual background
The lighting in the green box was carefully created to eliminate shadows in order to achieve the best results in the chroma key. The virtual set has numerous light sources to match the lighting in the green box and the virtual set.
In order to make the zoom possible in Unity3D the position and angle of the virtual camera was changed which led to a slight deviation of the field of view between the virtual and physical cameras. An adjusted script solved this problem. The virtual set was at times unsteady due to the sensitivity of the tracking system; this problem can be resolved by adding more cameras to the tracking system itself.

4. Conclusion
A major goal of this project was to keep the financial expenses low. The university provided the project with the interface for receiving position data, three out of the four used sensors and a position tracking system. Within our limited time frame we were able to keep the costs at approximately 500 euro. The created dynamic virtual studio system was used for a live show to showcase its abilities. During this live show this system proved to be capable of being used for this purpose. There were minor problems during the live show due to the limited time and resources but it was possible to create a presentable looking live show. Most of these problems had a workaround which made it possible to create a presentable looking live show with one mobile camera, a static camera with the ability to zoom and another static camera. The project could have been extended to use smart glasses with a Unity3D program. The utilisation of smart glasses can help the camera operators by visualising the virtual background. This is indeed a task for further project groups.

The authors laid out a first working prototype for the UAS St. Pölten and are thrilled that the further development, enhancement and usage of this system will be continued by the professors and students on campus.

Acknowledgments
We would like to thank: Christoph Thalinger for his expertise in the video studio and his feedback, Markus Wagner for his knowledge in Unity3D and Matthias Husinsky who was always prepared to help us in different areas.