



Assignment of bachelor's thesis

Title:	Tools for users embedding to video
Student:	Bogdan Putintsev
Supervisor:	Ing. Jan Buriánek
Study program:	Informatics
Branch / specialization:	Web and Software Engineering, specialization Computer Graphics
Department:	Department of Software Engineering
Validity:	until the end of summer semester 2022/2023

Instructions

Moderní interaktivní show vyžadují zapojení co největší možnosti zapojení uživatelů do děje. Jedna z metod je vložení uživatelů pro připravených videí, tzv. video-embedding nebo též augmentation. Práce se zaměřuje na vytvoření nástrojů pro vložení uživatelů do předem připraveného show (video nebo real-time zobrazení).

- 1) Prostudujte existující show využívající vkládání uživatelů do videí či performancí.
 - 2) Popište architekturu systému pro vkládání uživatelů do show a videí.
 - 3) Vytvořte vlastní návrh nástrojů, které umožní vložení uživatele a jeho parametrů do připraveného show.
 - 4) Proveďte reálnou implementaci nástrojů a demonstруйте jejich funkčnost na jednoduchém show využívající 'video-embedding'.
 - 5) Práci zpracujte včetně obrazových příloh a komentovaného zdrojového kódu.
 - 6) Implementaci vhodný způsobem otestujte.
-



**FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE**

Bachelor's thesis

Tools for users embedding to video

Bogdan Putintsev

Department of Software Engineering
Supervisor: Ing. Jan Burianek

May 13, 2021

Acknowledgements

I would like to express my thanks to my supervisor Ing. Jan Buriánek for his advice and support in the creation of this thesis. I would also like to thank my family, my girlfriend and my best friends for helping me through the whole study.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46 (6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the “Work”), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on May 13, 2021

.....

Czech Technical University in Prague

Faculty of Information Technology

© 2021 Bogdan Putintsev. All rights reserved.

This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).

Citation of this thesis

Putintsev, Bogdan. *Tools for users embedding to video*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

Abstrakt

Moderní interaktivní show vyžadují co největší možnosti zapojení uživatelů do děje. Jedna z metod je vložení uživatelů pro připravených videí, tzv. video-embedding nebo též augmentation. Práce se zaměřuje na vytvoření nástrojů pro vložení uživatelů do předem připraveného show (video nebo real-time zobrazení). Výstupem je vytvořený prototyp automatizované aplikaci pro vytvoření videa za pomoci herního engine Unreal Engine a technologie Chroma-key.

Klíčová slova Unreal Engine, Automatizace, Filmová produkce, Rozšířena realita, Chromakey

Abstract

Modern interactive shows require user involvement in the process. One of the solutions for users embedding to prepared videos is augmentation. This thesis is focusing on creating a tool for video or real-time augmentation to pre-prepared show. The outcome of this thesis is a built prototype of an automated application for shooting films with the usage of Unreal Engine and Chromakey technology.

Keywords Unreal Engine, Automation, Film production, Augmented reality, Chromakey

Contents

1	Introduction	1
1.1	Task Definition	1
1.2	Challenges	2
1.3	Goals	3
2	Analysis	5
2.1	Reference Projects	5
2.1.1	Holo - holograms in augmented reality	5
2.1.2	Timberland Augmented Reality Campaign	5
2.1.3	MANNER - Interactive Digital Signage	7
2.1.4	Virtual Movies - Augmented reality	7
2.1.5	Lynx Excite Angel Ambush London	7
2.1.6	Skin & Bones	8
2.1.7	Aircards - Augmented reality business cards	9
2.2	Prerequisites	10
2.3	Actors and Roles	11
2.3.1	Human	11
2.3.2	Filmmaker	11
2.4	Initial Architecture	11
2.4.1	View	13
2.4.2	Model	15
2.4.3	Controller	15
2.5	Technologies	16
2.5.1	Chromakey	16
2.5.2	Alpha compositing	16
2.5.3	Alpha-channel	17
2.5.4	RGBA	18
2.5.5	The algebra of compositing	18
2.5.6	Game Engines	20

2.5.7	Visual Scripting	22
2.5.8	Image Processing	23
2.5.9	OpenCV	24
2.5.10	NumPy	24
2.5.11	Rendering	25
2.5.12	Postprocessing	26
2.6	Development Stack	27
2.6.1	Game Engine	27
2.6.2	Programming language	27
2.6.3	Video Recording	27
2.6.3.1	Built-in tools	28
2.6.3.2	NewTek NDI	28
2.6.3.3	Sequencer Scripting plugin	29
2.6.3.4	FrameCapture plugin	29
2.6.3.5	OpenCV	29
3	Realization	31
3.1	Graphical User Interface	31
3.1.1	Main Menu	31
3.1.2	Guidance	32
3.1.3	Recording	33
3.1.4	Actual Shot	33
3.1.5	Final Screen	34
3.2	Plot File Format	35
3.2.1	Film Structure	36
3.2.2	Shot Structure	36
3.2.3	Body Position Type Enum	37
4	Prototype Demonstration	39
4.1	Concept Idea	39
4.2	Assets Import	39
4.3	Film plot declaration	40
4.4	Creation of the Level Sequences array	42
5	Conclusion	43
5.1	Room for future improvement	43
	Bibliography	45
6	Contents of enclosed CD	49

List of Figures

2.1	Holo - augmented reality application	6
2.2	Virtual Fitting Room by LemonOrange	6
2.3	Manner - Advertising campaign	7
2.4	Virtual Movies in a mall	8
2.5	Lynx Excite Angel Ambush London	8
2.6	Skin & Bones	9
2.7	Aircards application	9
2.8	Requirements Diagram	10
2.9	Actors and Roles	11
2.10	Requirements Diagram - part 1	12
2.11	Requirements Diagram - part 2	13
2.12	MVC - dependencies	14
2.13	Chromakey	16
2.14	Alpha compositing	17
2.15	RGBA	18
2.16	Compositing Example in Unreal Engine	21
2.17	Unreal Engine	21
2.18	Blueprint Example in Unreal Engine	23
2.19	OpenCV	24
3.1	Main Menu	32
3.2	Guidance	32
3.3	Recording Menu	33
3.4	Shot Review	34
3.5	Final Screen	34
3.6	Example of a JSON file	35
4.1	Realistic Forest Pack in the Marketplace	40
4.2	Infinity Blade: Fire Lands in the Marketplace	40
4.3	Main menu after customization	41

4.4	Shot declaration in JSON file	41
4.5	Level Sequence file for the first shot	42

Introduction

Every year filmmaking process is changing. Computer graphics starts looking natural and indistinguishable from the real world. Technologies in studios 50 years before and now have distinctly changed. And even modern techniques such as chromakey that is well-known for everyone starts outliving themselves.

Nowadays CGI is available not only for huge studios with a multi-million dollar budget. There are lots of open-source software that make indie film production possible and saves money for essential expenses. But still, it requires corresponding skills and not user-friendly for beginners.

There is a tendency to simplifying a movie creation for personal purposes. The application marketplace is full of tools for photo/video editing with prepared presets or assets created for public usage. Here comes the idea, where presets can be a pre-prepared film without any actors and anybody can be starred in the film.

This thesis focuses on creating an automated system, that can be used in public spaces, museums, and galleries. The prototype will be able to create a real-time augmented reality movie. The actor can see help guidance on a screen during recording. He would see body position in a shot or a hint of what he needs to do. The application can handle such things as chroma-key processing, and help guides on a screen. The actor can see himself in real-time as preview results in a low resolution and ask for the final result that will be rendered in a high-quality resolution. Everyone can also create custom plots with assets.

1.1 Task Definition

This system is based on a movie script that will be converted into the queue, where each element stands for the certain actor's instructions and contains information about scenes, camera's position, and guidance instructions.

The input of this system is pre-prepared scenes (video sequences) with the assets and movie script in a suitable format (JSON). The user's input

(personal data) contains information on a photo, video, and text formats that will be provided in a runtime. The output is a movie that was automatically created and contains all the assets on a scene and personal data.

The whole movie is created with a certain amount of shots, where each shot is a new element in the queue. One single shot consists of closeup, background, and a layer with user data and generates a set of frames. A movie script consists of:

1. Technique information
2. Declaration of all assets
3. Guidance information
4. Periphery communication (camera, sound)

At the beginning of the process, the user will choose a nickname. The system will assign a unique ID and hash with the filename of the future movie. Right after that application will guide an actor to required action (answering the questions what to do, where and when) and request personal data (photo, video, or an interactive quiz).

When all the shots are done, the system will show a preview of the final video and ask the user, whether he wants the whole version. In case he'll take it, a user will type his email address with a link containing the status of his request, the approximate time when it will be rendered and link to download the video.

1.2 Challenges

The first problem includes reading all the instructions set as a file or through the protocol. It needs to be read and controlled if the input is correct.

The data set has to be converted into a queue. Each element contains information about augmentation details.

The system requires an interface that can check if all the periphery works correctly, which means the cameras can send an image flow, and the tracker reacts on camera transition, and if not - help to figure this out.

There must be a hint on a screen, that will tell the actor what to do. This help guide will be supporting the actor through the whole process of making a film. It contains information about the actor's text in a plot, body in a shot, head position on a screen, specific actions, etc.

After recording one shot, users will see the preliminary result in a media player. The actor approves if he likes the current shot or repeats it.

In the end, it needs to calculate the final video with the whole augmentation in maximum quality and show it on the user's screen.

1.3 Goals

There are different goals, which will be archived. The solution will be described in the next section.

1. Choose tools for the creation of the system.
2. Find a solution for video capturing, so the application will be able to record the data flow from the camera and save it to the repository
3. Design suitable GUI
4. Creating a suitable format for reading the plot information about the film
5. Implement processing of this format
6. Implement transition between scenes
7. Media player realization
8. Implement the offline rendering, the system can concatenate the shots and combine personal data with computer graphics

The last goal is to create an autonomous prototype of a system that will be able to interpret the plot and execute all the instructions which were declared before.

The final part of this thesis consists of testing the prototype in a real studio with prepared materials.

Analysis

2.1 Reference Projects

At the beginning of the work process, it was necessary to find existing solution, so there will be a basis for all the requirements and specificity. There are lots of similar projects that are used from a game development to a usual shopping sphere. Choosing was based on a technologies that were used such as augmented reality and computer vision.

2.1.1 Holo - holograms in augmented reality

Holo is an application that has similar principles of work. It allows recording a video with 3D objects using augmented reality. In the beginning, you can choose and download the required assets, calibrate your position, and after that make a short film with the characters you've chosen. Assets are not predefined like in our application but could be set in runtime. [1]

2.1.2 Timberland Augmented Reality Campaign

Timberland used the technology of augmented reality, a virtual fitting room, where visitors are able to choose clothes at will. First of all, they had to put their head in a special position, so the camera can take a customer's picture. After that clients could use a virtual cursor just moving their hand and this way try clothes on themselves. Thanks to Kinect technology, devices can analyze a person's movement and gesture. In the end, favorite outfits were saved and sent by email, so the customer can buy it online or just show the employees so it can make things faster.

One of the main principles of gamification is to give a choice, industries use it all over the world to increase income and influx of customers. This example of such projects with gesture recognition can be useful in a shopping industry. [2]

2. ANALYSIS



Figure 2.1: Holo - augmented reality application [1]



Figure 2.2: Virtual Fitting Room by LemonOrange [2]



Figure 2.3: Manner - Advertising campaign [3]

2.1.3 MANNER - Interactive Digital Signage

The Manner is a line of Austrian confectionery by Josef Manner Comp AG. They made an advertising campaign on a Wien railway station. It is a simple game, where everyone can catch waffles on the head, so the players should have balanced to not drop it all. The winner gets a sheet, which he can change with real waffles for free. Signage can recognize the human body and work with this shape, so the head acts as a collision. This cheap and simple project is remarkable for all passers-by and that meets the main purpose for all the advertisers. Similar campaigns were created by Red Bull and Pago-Granini companies in the same railway station [3][4][5]

2.1.4 Virtual Movies - Augmented reality

LemonOragne company created a small real-time movie in a mall, where all the children could be a part of it. Here is an example of using alpha compositing, where assets were pre-rendered and overlaid with usual video streaming. [6]

2.1.5 Lynx Excite Angel Ambush London

Another one example of using augmented reality in real-time was in a railway station of Victoria in London, where people standing on a special sign could see themselves on a screen. After a few seconds, they could see an angel falling from

2. ANALYSIS



Figure 2.4: Virtual Movies in a mall [6]



Figure 2.5: Lynx Excite Angel Ambush London [7]

the sky. This animation was pre-rendered and the final image was composed using alpha-blending. [7]

2.1.6 Skin & Bones

Skin & Bones is one of the examples, how museums are using augmented reality. This app brings the animal's skeleton and put a 3D model of that animal on that place in real-time. [8]



Figure 2.6: Skin Bones [8]

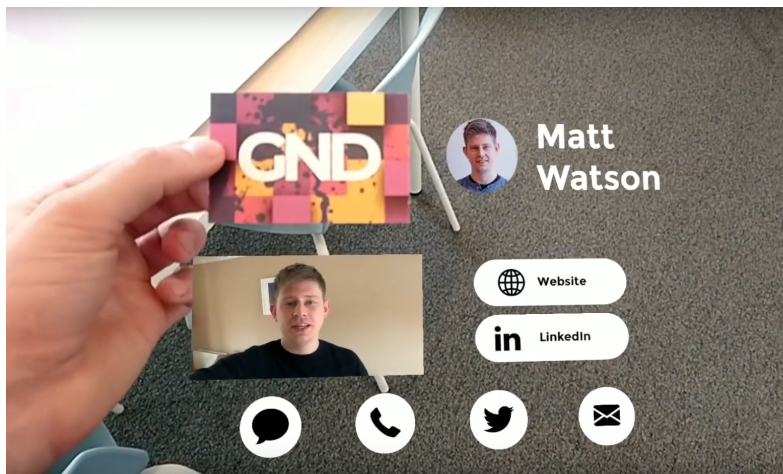


Figure 2.7: Aircards application [9]

2.1.7 Aircards - Augmented reality business cards

The Aircards is a new way to expand the ordinary business card into a new informative way. Near the card, you can put any additional information you want such as links to the website, email, social media, or your own company video. It gives a new life to give contact information.[9]

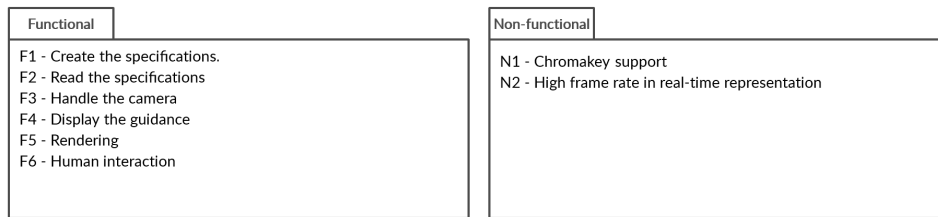


Figure 2.8: Requirements Diagram

2.2 Prerequisites

In this section we can deduce these functional prerequisites:

1. **F1 - Create the specifications.** The system must be able to create and read a file format with regard to the plot to handle all the assets and instructions including VFX, guide helper, etc.
2. **F2 - Read the specifications.** Read all the necessary files, assets, and information for further work.
3. **F3 - Handle the camera.** Assess the camera.
4. **F4 - Capture video.** Convert analog video signal and save it to the storage.
5. **F5 - Display the guidance.** Display tutorial and guide to the actor helping him through the whole film recording.
6. **F7 - Rendering.** Ability to concatenate shots, render the whole video and combine background and foreground.
7. **F8 - Human interaction.** The actor can take part in the process, watch the video, approve or deny the transition to the next shot.

To non-functional prerequisites can be included:

1. **N1 - Chromakey support.** Background can be replaced using chromakey technology, if it's specified in a plot.
2. **N2 - High frame rate in real-time capturing.** The application must be capable of recording the video in acceptable resolution with a frame rate at least 20 FPS in average.



Figure 2.9: Actors and Roles

2.3 Actors and Roles

This section describes actors and external systems that will be used.

2.3.1 Human

Primary actor. Will be filmed and interacting with the interface. He can start the process of recording the film and ask for a final video in a full resolution afterwards.

- start the process of making the film
- ask for the final video in a full resolution

2.3.2 Filmmaker

A user who can create or edit screenplay file, upload custom assets and specify such things as:

- assets in each shot
- the guidance instructions for the primary actor

2.4 Initial Architecture

This section describes an initial architecture for the implementation of all the requirements in section. It's separated into three components the so-called Model-view-controller.

In the MVC paradigm the user input, the modeling of the external world, and the visual feedback to the user are explicitly separated and handled by three types of object, each specialized for its task. The view manages the

2. ANALYSIS

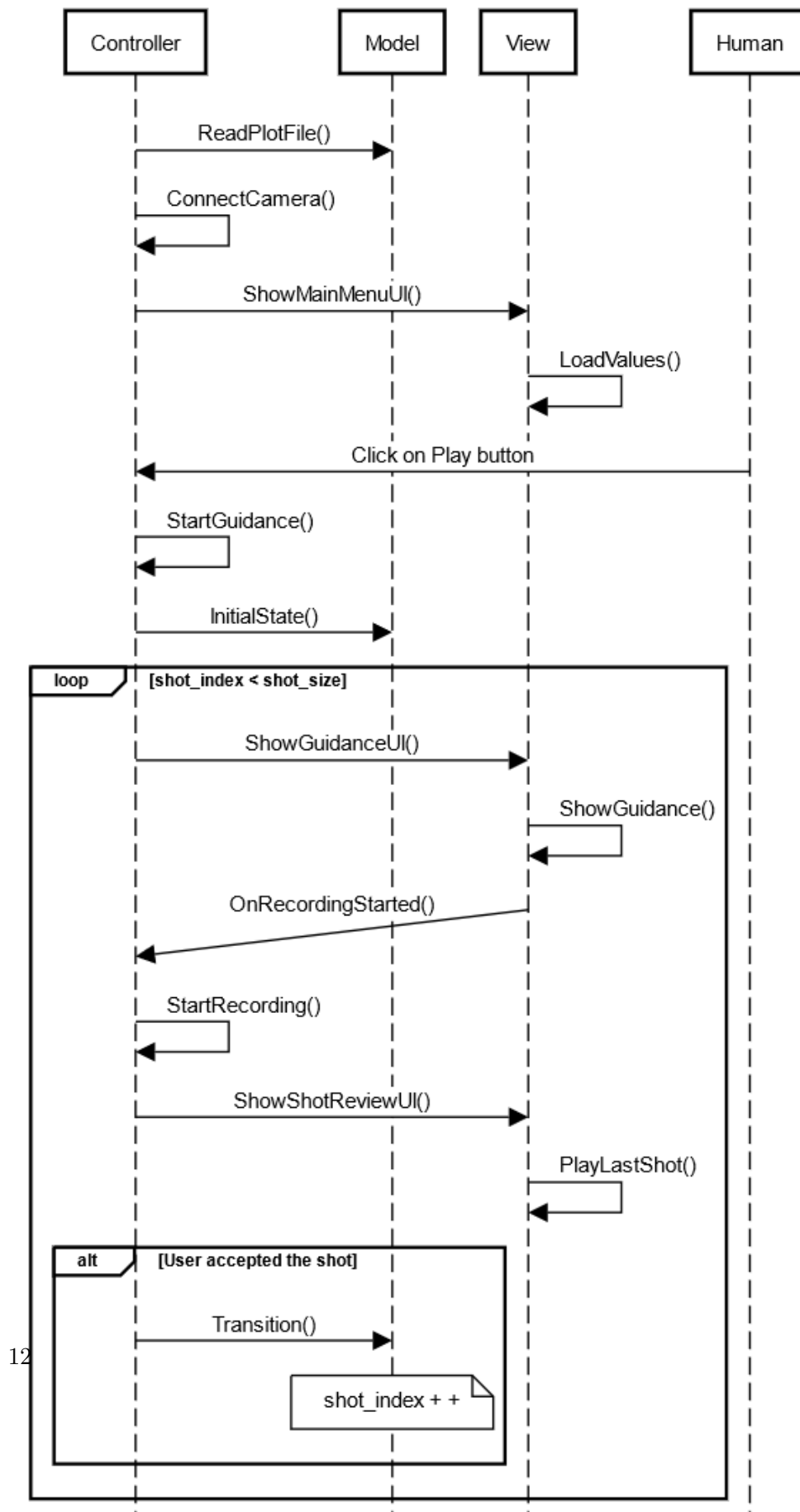


Figure 2.10: Requirements Diagram - part 1

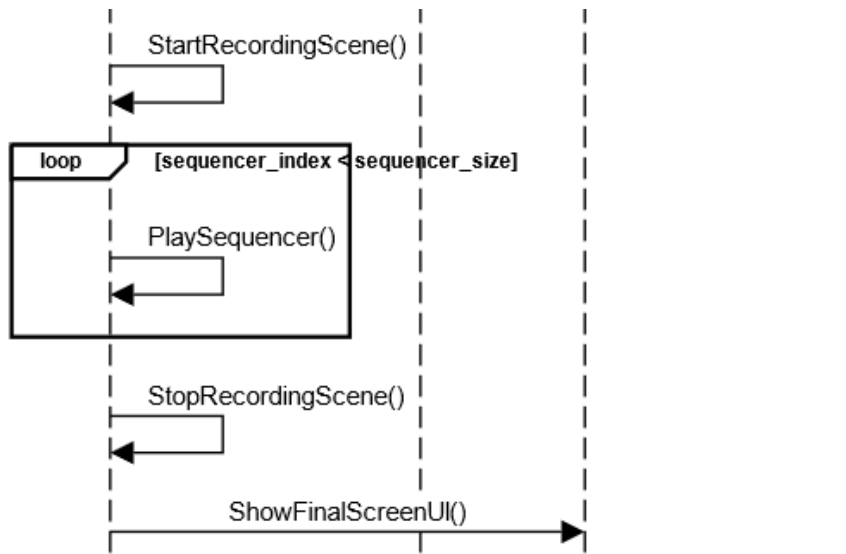


Figure 2.11: Requirements Diagram - part 2

graphical and/or textual output to the portion of the bitmapped display that is allocated to its application. The controller interprets the mouse and keyboard inputs from the user, commanding the model and/or the view to change as appropriate. Finally, the model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller). [10]

2.4.1 View

View component is responsible for handling the display of the user interface. It contains methods for opening widgets, style declaration, and guidance display.

- **View Actor** - stands for communication and switching between widgets
- **Main Menu UI** - a widget will load the style of the interface from the model and shows the main menu
- **Guidance UI** - loads guidance information from the model, converts it into the image and shows the guidance to a user
- **Recording UI** - indicates that recording has started
- **Shot Review UI** - graphical representation of media player, communicates with the model, shows the result of a shot or the whole film

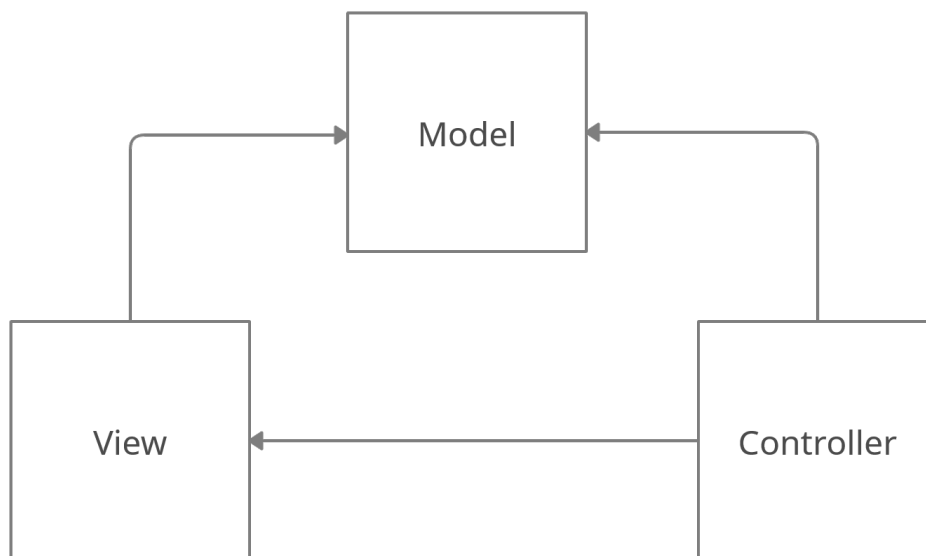


Figure 2.12: MVC - dependencies

2.4.2 Model

In the final implementation, the model component consists of one class - Model Actor. It performs the following functions:

- read the data from JSON file
- validate input from the file
- keep data in a storage
- keep the array of all the shots
- implements methods for transition to the next shot
- pass information to classes from View component

2.4.3 Controller

The Controller Actor class handles events, accepts input, and calls required methods for View or Model components. Also it can handle:

- contains methods for video capturing and rendering
- calling external scripts
- open level sequences files



Figure 2.13: Chromakey [14]

2.5 Technologies

2.5.1 Chromakey

Chromakey is a technology of layering, or compositing two or more images or frames in one composition, color electronic rear-projection used on television, and in modern digital film technology. Using chromakey, you can separate the background layer and replace it with an image or video.

An image can be assumed to be a composite of the foreground and the background. The foreground and the background of each pixel are linearly combined in terms of this pixel's foreground opacity (called alpha). Image matting is the process of estimating the foreground, the background and the alpha for each pixel [11].

Image matting is an important technique in image and video editing, which was originally developed for film and video production [12]. The mixing of several pictures—the elements—to form a single resulting picture—the composite—is a very general notion. Here we shall limit the discussion to a special type of composite frequently made in film and television, the matte shot. This consists of at least two elements, one or more foreground objects each shot against a special backing color—typically a bright blue or green and a background. We shall limit ourselves to the case of one foreground element for ease of presentation [13].

2.5.2 Alpha compositing

Alpha compositing is a process of combining two images, where one of them is a background. In other words, it stands for layering different images and

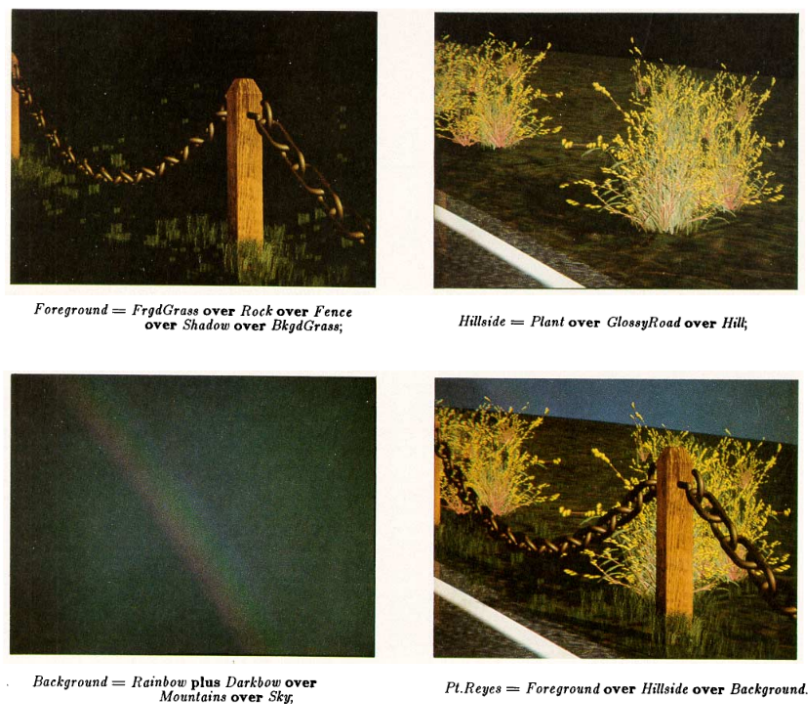


Figure 1

Figure 2.14: Alpha compositing [15]

merging them into one using a transparency feature.

Separating the image into elements which can be independently rendered saves enormous time. Each element has an associated matte, coverage information which designates the shape of the element. The compositing of those elements makes use of the mattes to accumulate the final image [15].

2.5.3 Alpha-channel

The standard RGB channel contains only information about the color, however, it is not enough for blending purposes. So it requires information about the mixing factor to understand the transparency regarding other images. To handle this process image requires transparency, it can be realized with alpha-channel. This concept consists of additional pixel data, a special value from 0 to 1, where 0 means that pixel is fully transparent, and the value of 1 means that the value is fully occluding [15]. Alpha-channel is required. It gives shape and transparency to a single image.



Figure 2.15: RGBA [16]

2.5.4 RGBA

RGBA is a color space model with supplemented the fourth alpha channel. The last channel indicates how transparent is a certain pixel.

2.5.5 The algebra of compositing

When blending pictures we don't have information about overlapping. All we know is color information and alpha-channel for each pixel. So we need to know how to interplay with these values [15].





So let α_A and α_B be the values representing the transparency of two objects A and B . Each object has only $(1 - \alpha_A)$ and $(1 - \alpha_B)$ of their background visible, so that the background shows only $(1 - \alpha_A)(1 - \alpha_B)$ of the pixel. The








value of $\alpha_A(1 - \alpha_A)$ defines object A on the foreground of object B. The value of $\alpha_B(1 - \alpha_A)$ defines the opposite.

The result is summarized in the following table:

Description	Area
$\neg A \cap \neg B$	$(1 - \alpha_A)(1 - \alpha_B)$
$\neg A \cap B$	$\alpha_B(1 - \alpha_A)$
$A \cap \neg B$	$\alpha_A(1 - \alpha_B)$
$A \cap B$	$\alpha_A\alpha_B$

Now when we separated the image into four disjunctive parts - 0, A, B, AB, we can define operations to determine their intersection

Operation	F_a	F_b	Diagram
clear	0	0	clear
A	1	0	
B	0	1	
A over B	1	$(1 - \alpha_A)$	
B over A	$(1 - \alpha_B)$	1	

A in B	α_B	0	
B in A	0	α_A	
A out B	$(1 - \alpha_B)$	0	
B out A	0	$(1 - \alpha_A)$	
A atop B	α_B	$(1 - \alpha_A)$	
B atop A	$(1 - \alpha_B)$	α_A	
A xor B	$(1 - \alpha_B)$	$(1 - \alpha_A)$	

2.5.6 Game Engines

A game engine, also known as a game framework or game architecture, is a software designed for creating video games. The core includes such components as physics technologies, rendering for 2D and 3D graphics, collision detection, particle system, sound processing. It's been used for many different purposes from architecture visualization to general game creation. Nowadays there are 3 leading companies in a game engine area - Unity, Unreal Engine, and Unigine. All of them have pros and cons. Choosing the software proceeded from price offers, plugins support, a barrier of entry, and actual knowledge and possibilities.

Unreal Engine is one of the leading 3D engines initially based on creating state-of-art game development but now is applicable in architecture, broadcast, live events, films television, and so on. Written in C++ it differs in its portability and supports a lot of platforms from Windows, Linux, macOS to Xbox One, and Nintendo. Unreal Engine was the final choice because it's open-sourced, code is available on GitHub, it has a huge community, and it's free to use. [17]



Figure 2.16: Compositing Example in Unreal Engine



Figure 2.17: Unreal Engine [17]

2.5.7 Visual Scripting

A visual programming language (VPL) is a programming language where the programmer constructs a program by graphically arranging graphical and textual objects – note that the spatial arrangement of these objects typically have a semantic meaning. A visual programming environment (VPE) is an interface which allows the programmer to manipulate the graphical and textual objects. VPEs have been shown to be an ideal platform for non-expert programmers to produce algorithmic solutions [18].

The visual programming (VP) community has produced many ideas in the past decade with the goal of ameliorating the difficulties of programming. These ideas, in turn, have raised questions about the cognitive processes required in programming and are of particular interest in the context of empirical studies. For example, many visual programming languages (VPLs) have been designed to reduce the mental effort of programming. The design decisions for the VPLs are, currently, the result of (educated) guesses about how to facilitate the programmer’s cognitive processes, rather than the result of established theory [19].

In Unreal Engine VPL is called *Blueprints*. The Blueprint Visual Scripting system in Unreal Engine is a complete scripting system based on the concept of using a node-based interface to create gameplay elements from within Unreal Editor. As with many common scripting languages, it is used to define object-oriented (OO) classes or objects in the engine [20].

A *Graph* is a network of nodes that can be connected to one another in order to define the flow of execution for the network. Graphs are the basis for implementing functionality in Blueprints. Each Blueprint can contain one or more graphs, depending on the type of Blueprint, that define the implementation of a particular aspect of the Blueprint. Each graph within a Blueprint can also contain sub-graphs, which are essentially collections of nodes collapsed into their own separate graph, mainly for organizational purposes [20]. Unreal Engine distinguishes graph type:

- *Event Graphs* - contains a node graph that uses events and function calls to perform actions in response to gameplay events associated with the Blueprint.
- *Construction Scripts* are useful for Blueprint Class initialization, as they run right after the Components list is set up for the Blueprint Class.
- *Functions* are node graphs belonging to a particular Blueprint that can be executed, or called, from another graph within the Blueprint.
- *Macros* are the same as collapsed graphs of nodes. They have an entry point and exit point designated by tunnel nodes. Each tunnel can have any number of execution or data pins which are visible on the macro node when used in other Blueprints and graphs.

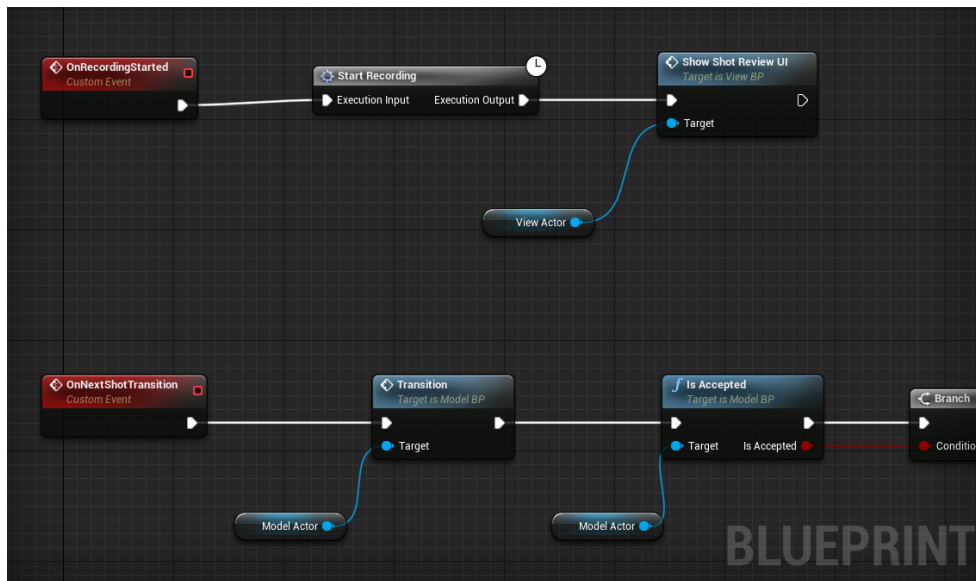


Figure 2.18: Blueprint Example in Unreal Engine

2.5.8 Image Processing

Digital image processing is a scientific discipline that processes digital images in a camera flow with algorithms. Algorithms could be different, such as segmentation, edge detection, shape detection, etc. It's widely used in computer vision, robotics, or medical diagnostic.

Digital image processing is used in two types of applications. The first deals with those applications in which the images are processed for direct use by a human investigator. Examples include processing modules for image enhancement, restoration, and compression. In the second type of application, image processing finds its utility as a front-end processing module of a complete machine vision system. In this case, the overall aim is not simply image-to-image transformation, but rather an interpretation of information embedded in the images [21].

A digital image is represented as a discrete two-dimensional array of numbers. Each element in the array is known as a pixel (for picture element). These pixels are assigned values that correspond to the relative brightness of the tiny portions of the image that they depict. These values are known as gray levels. Digital image processing deals with the systematic manipulation of the pixel gray levels and their distribution [21]. For gray tints, there is only one channel, but for RGB or HSV color models, there are three channels per pixel.

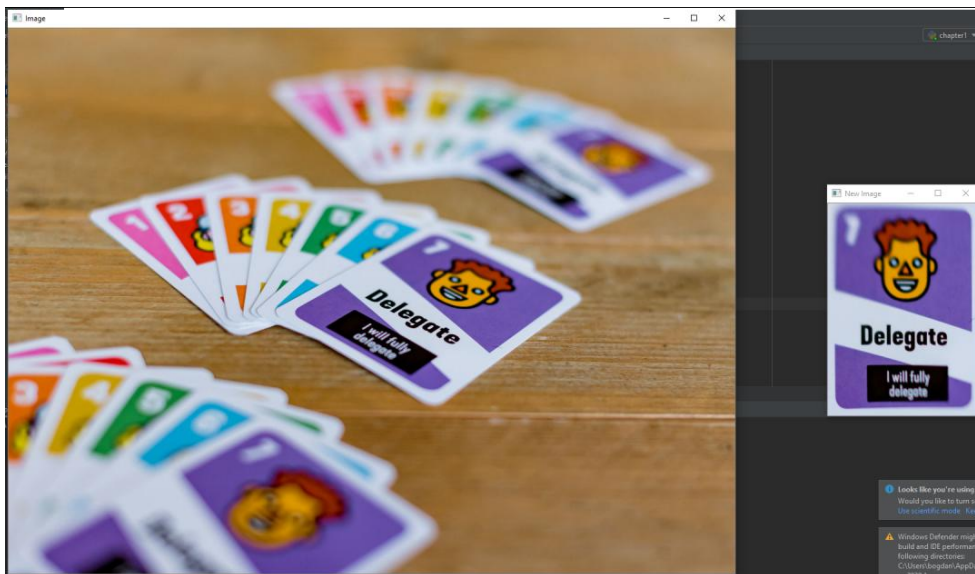


Figure 2.19: Example of usage OpenCV library

2.5.9 OpenCV

OpenCV (Open Source Computer Vision Library) is a library that includes more of 2500 optimized computer vision and machine learning algorithms, which can be used for face recognition, identify objects, classify human actions, track camera movement, etc. It supports a lot of programming languages such as C++, Python, and Java and different platforms like Windows, Linux, Android, and macOS. It's free for use under a BSD license [22].

OpenCV also supports video capturing and video recording. The library includes all necessary codecs for several video formats. To capture a video, you need to create a *VideoCapture* object, after that it's possible to receive camera data flow or iterate through the prepared video file. For recording, there is a *VideoWriter* class. It takes a filename, codec, number of frames per second, and resolution as input parameters.

2.5.10 NumPy

NumPy is an open source project aiming to enable numerical computing with Python. It was created in 2005, building on the early work of the Numeric and Numarray libraries [23].

It's very suitable for working with large, multi-dimensional arrays and matrices. Useful for scientific researches and in computer science. In this thesis adding the library is excessively important due to the image representation. Each frame in a video is represented as a matrix, that is where Numpy becomes helpful.

2.5.11 Rendering

One of the primary goals of the whole thesis is to receive the final video. First of all, all shots have to be concatenated. Second of all, composure has pre-rendered background and foreground with an actor.

Rendering is the process of generating an image from a model of a virtual scene. We describe the geometry, viewpoint, textures, lighting, and shading information. It's commonly used in architecture, animation movies, game development, physics simulation, and visualization. Modern technique stands for imitation of the real world considering the computation possibilities and aims at the basic purpose of the project. It's not necessary to copy the real world with redundant details. Simplify the environment and simulate the view using the physics and geometry data is reasonable.

There are two categories of rendering: pre-rendering and real-time rendering:

- *Pre-rendering* (also known as offline rendering) is a process in which assets were previously prepared on different equipment and may be outsourced by the developer to an outside company. Usually, it's computationally intensive process, where time to process one frame varies depending on computing power. Typically used for movie or cartoon creation.
- *Real-time rendering* (also known as online rendering) is producing and analyzing images in real-time. It's used in all kind of interactive 3D graphics or image analysis. Typical example of usage is video games, where next frame is unknown and depends on player's input and in this case it needs to be rendered rapidly.

There are several models to describe the rendered image. These features are required to simulate different surfaces efficiently and affect the color of certain pixels.

- *Shading* - how the color and brightness of a surface varies with lighting
- *Texture mapping* - method for defining color information according to the certain position of 3D model
- *Bump-mapping* - method of simulating the geometry on surfaces using the texture
- *Shadows* - effect of blocking a light source by an object
- *Reflection* - simulation the mirror or glossy reflection
- *Caustics* - reflection of light off a shiny object

2.5.12 Postprocessing

The term of post-processing is used in video industry to improve the quality of image processing using several methods. Unlike rendering it's used to add some extra details and make the picture look relatively better.

- *Color correction* (including *Color adjustments* and *Color Curves*) lets you change the brightness, tone, contrast, exposure, temperature and adjust specific ranges in hue, saturation, or luminosity
- *Depth of Field* defines the focal length, background area that stays blurred and foreground in focus
- *Anti-aliasing* effect softens the edges in image
- *Ambient Occlusion* effect darkens the areas that are not subjected to ambient lighting
- *Bloom* makes bright areas glow
- *Motion Blur* blurs the image in the direction of camera's movement

2.6 Development Stack

This section focuses on a tool declaration for the implementation of all tasks of the thesis. Each section is devoted to a certain type of software, library, or method of achieving the goal. Some of the instruments were chosen because of the intention to learn and explore new technologies but in most cases, it was guided by objective reasons.

2.6.1 Game Engine

Unreal Engine was chosen because it's one of the most powerful, popular game engines with potential. It's challenging to learn all the required aspects of this program, but it allows us to create AAA realistic scenes, which is a common requirement in film making. Also, it contains lots of tools for film production, such as Sequencer, Composure that helps in animations, rendering, and post-processing the whole film.

2.6.2 Programming language

Even though the primary programming language for game development in Unreal Engine is C++, I decided not to use this in production. First of all, a built-in library in a game engine is massive. It requires a deep understanding and time to understand all these concepts. Second of all, the project depends on third-party libraries (such as OpenCV). After an enormous exploration, I couldn't find any sensible solution. It took time to solve problems with linking and broke the whole project several times. One of the benefits of Unreal Engine is a plugin for Python language support. The solution was to use Python scripts while working with third-party libraries and the Unreal Engine's Blueprints for the rest. The Blueprints was the most straightforward way to manipulate with a game engine. It's helping with debugging and intuitive for learning.

2.6.3 Video Recording

The next challenging goal was to be able to record the screen. The video of each shot must be saved in data storage before the rendering process. There are several possible solutions for this goal:

- Built-in tools
- NewTek NDI
- Plugins for UE
- OpenCV

2.6.3.1 Built-in tools

Unreal Engine becomes well-liked in film productions, not only in indie companies but in huge like Disney Films. They use the engine because now it's the only way to archive photo-realistic render in real-time.

The Epic Games tries to support it and develops tools for filmmaking. The Sequencer allows video editing, animation, and working with cameras. The Composure plugin makes it possible to create layers, such as background, media, and foreground, setup chromakey.

Take Recorder enables recording the actor's animation in real-time or from motion capture linked to characters in the level. All recordings can be added to the Sequencer to create the whole film. Unfortunately, the Take Recorder doesn't work with a Media Player and doesn't allow recording the video from a camera. That's the reason, why these tools have to be declined.

2.6.3.2 NewTek NDI

NDI® is a free protocol for video over IP, enabling better video for everyone. NDI software is in the hands of millions of customers worldwide, creating an interconnected community of storytellers. NDI allows individuals and organizations to access the benefits of IP-based, software-defined visual storytelling for a fraction of the cost of other complimentary IP protocols [24].

Package for Windows includes:

- NDI Studio Monitor
- NDI Screen Capture
- NDI Webcam Input
- NDI VLC Plugin

The company also provides an NDI Software Development Kit that provides the tools and resources developers and manufacturers need for integration.

NDI is compatible with platforms of Adobe, OBS, Apple, Android, and even Unreal Engine. It makes live streaming possible, and the company shows great potential.

On the other hand, it has significant disadvantages for this thesis. NDI works with technology video over IP. All the data have to be delivered through the protocol, and it always impacts delay. Video has to be recorded in a maximum quality depending on a computer's potential. Moreover, it requires the presence of the server or a data center, where the camera will be separated.

And last but not least, the recording has to be controlled remotely. It requires development with an SDK, which is a challenging task. There are wrappers for various programming languages that use NDI SDK, but nobody can guarantee, they are working correctly.

2.6.3.3 Sequencer Scripting plugin

While doing the rendering task, it was necessary to find a solution for a high-quality rendering. A sequencer has its rendering function, but it works manually, which doesn't comply with requirements. There is a built-in plugin that opens access to sequencer API. There is almost no information about this tool on the internet, but developers included an example script in a game engine. It worked well for the first time but started to crush the game engine in practice without any reason.

2.6.3.4 FrameCapture plugin

FrameCapture is a C++ based library in Unreal Engine that lets you take a snapshot of your game viewport at the desired resolution, and record your in-game screen. It allows you to adjust the camera's depth of field settings to create beautiful photographic images. There is an option to display the UI screen in a snapshot or video output. Enabling the UI screen will capture the entire game window and hence will also display contents that are not directly rendered by your game viewport. Also, it allows you to convert your game screen into texture and display in UMG image format, and load image files from your hard disc as texture [25].

Installation of this plugin is trivial. First, it needs to spawn the actor and after that, there are available functions for taking a snapshot or for the video rendering. It all can be used in blueprints.

This plugin was included throughout almost the whole development process, but unfortunately, it's unstable and takes a lot of CPU memory. Each shot has its duration, and the final video has a low framerate, so each shot is considerably shorter than it has to be. In some cases, it writes broken videos and it has a bad impact on a workflow.

2.6.3.5 OpenCV

OpenCV is not primarily used for these purposes, but the library contains tools for video recording. To create a video, you need to create a *VideoCapture* object. Constructor requires a file name, codec for a video format, number of frames per second, and resolution. Then in each iteration of the loop, add a frame to an object. To save the ready file you need to use method *release*, which gathers it into a file.

Unreal Engine supports Python scripts. There is an addon that has to be enabled. Then a node *Execute Python Script* will be available in the blueprints.

One of a disadvantages of this method is that a game engine executes all the scripts in the thread. The screen pauses during the execution and waits until the end. For recording, it's not a big problem, because an actor should

2. ANALYSIS

never look into a camera. Plus, it increases the quality of a result, because all the resources aim to a recording process.

Online rendering is responsible for recording the user input and save it into storage. The user will see a preview after each shot and rate it.

The research contained testing lots of possible solutions. After comparing all the pros and cons, I decided to use the FrameCapture plugin for rendering the final video and custom script with OpenCV for an actor recording.

Realization

This chapter focuses on the realization of the prototype. Each section contains a description of the goal, related problems, and the final solution.

3.1 Graphical User Interface

For the implementation prototype, it was necessary to create a graphical user interface (GUI) that will be intuitively understandable and personalized (which means it has to be customizable for each creator).

All the 2D graphic elements are implemented in UE with User Interfaces (UIs) and Heads-up displays (HUDs). It's a way of providing information and interaction. The HUDs are usually non-interactive, while UIs refer to menus and other interactive elements.

These can be created via so-called Widgets. Each widget contains a design and graph tabs. Design mode provides tools for creating a layout, inputs, graphical elements, and a panel for positioning. Graph mode represents a blueprint related to a widget. Each widget contains a constructor, functions, macros, and each element can be represented as a variable.

UIs can also handle events (such as pressing the button or a checkbox). So practically, each widget has its presenter and a controller. But for architecture simplifying in the thesis, they are only presenters and calls a global controller actor to notify about user interaction.

3.1.1 Main Menu

It is the first widget in the whole workflow. It shows the film's name, its description, some background image, and a button that starts the film recording. This menu is customizable. Background image, blur size, and the accent color is described in a plot file. In the future, it can be more personalized, and some features can be included.

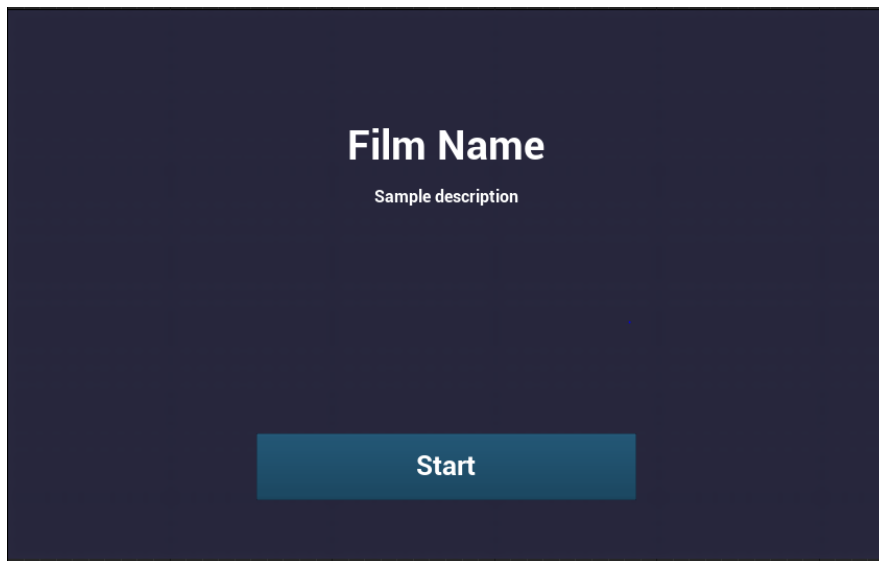


Figure 3.1: Main Menu

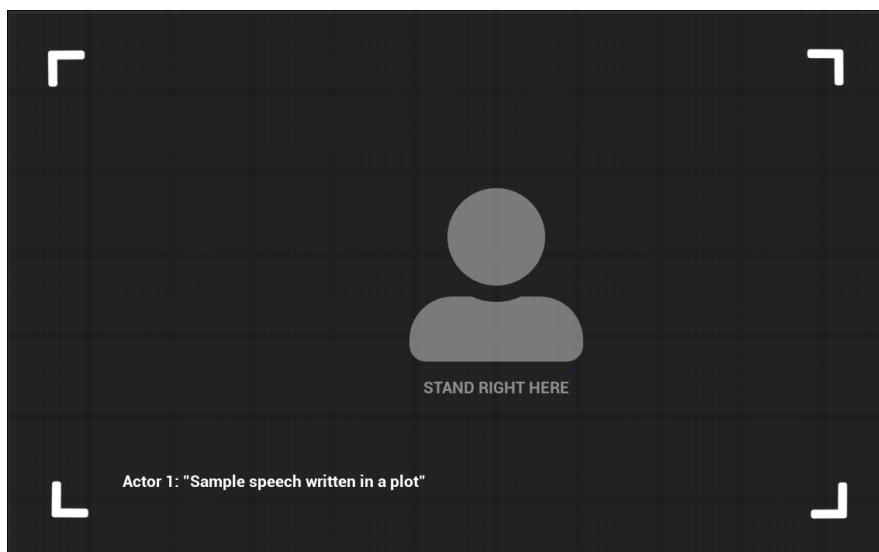


Figure 3.2: Guidance Menu

3.1.2 Guidance

Guidance is one of the most essential widgets. It receives the data from a model actor about the current scene. This information contains hints to the actor what is going on in this shot.

There is always a text that the actor needs to say or to do. The creator will be able to specify where to place the actor on a screen. It is shown with a

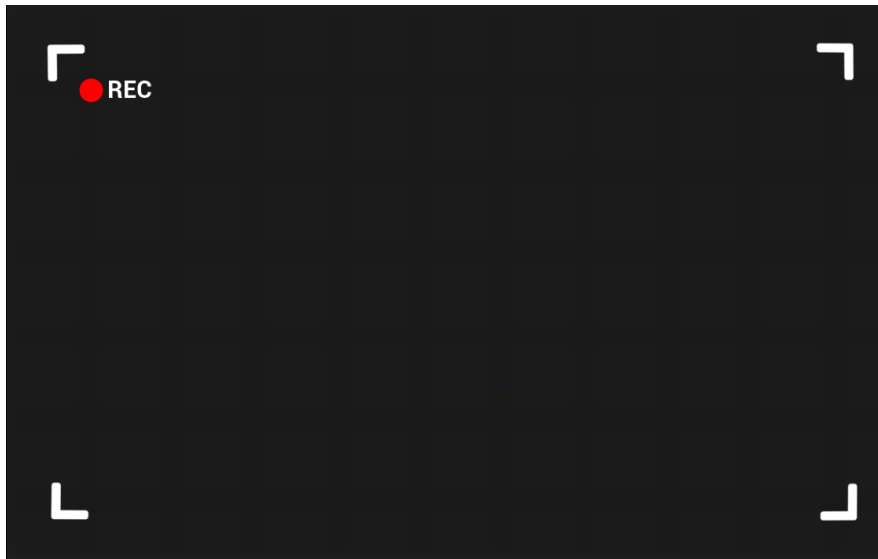


Figure 3.3: Recording Menu

specific icon. Images can be included in the project and are also customizable. On top of it is a number of the shots left until the end of the film.

There is a font pre-installed in a project for more customization. It's called Font Awesome, the internet's most popular icon toolkit that has been redesigned and built from scratch. Usage is simple - it requires copying the element from the official site and paste it as a text field in a widget.

3.1.3 Recording

The recording window indicates that the actor is being filmed. It has only a blurred background and the frame of a camera. After the timer finishes, it disappears.

3.1.4 Actual Shot

This widget is responsible for showing the actual shot that has been taken. It opens a Media Player with the last recorded video. There are buttons to manipulate a video, such as play, pause, or start playing from the beginning. There are also two valuable buttons - thumb up (like) and thumb down (dislike). The actor will be able to see the video and rate it. If the current shot satisfies him, he approves it, and the program goes to the next shot. Otherwise, it repeats the same scene.

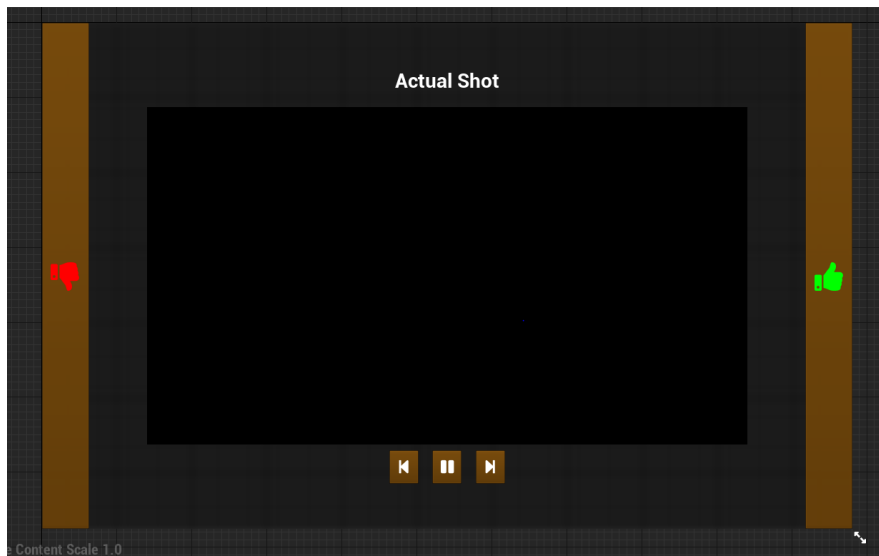


Figure 3.4: Shot Review Menu

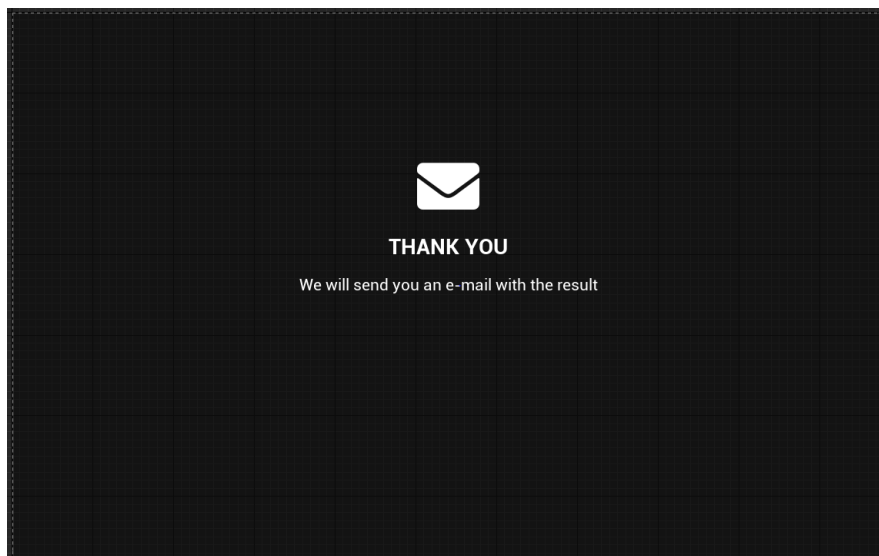


Figure 3.5: Final Screen Menu

3.1.5 Final Screen

This window indicates that rendering is completed. Sending an email is not implemented, but it shows the principle of work. Widget stays for five seconds, and after, goes to the main menu.


```
1  [
2  {
3    "Name": "awesome_film",
4    "film_name": "My awesome film",
5    "film_description": "Cool awesome film full of action...",
6    "working_directory": "D:\\Film\\",
7    "script_directory": "D:\\Scripts\\",
8    "delay": 5,
9    "background_color":
10   {
11     "B": 30,
12     "G": 30,
13     "R": 30,
14     "A": 255
15   },
16   "text_color":
17   {
18     "B": 255,
19     "G": 255,
20     "R": 255,
21     "A": 255
22   },
23   "accent_color":
24   {
25     "B": 81,
26     "G": 89,
27     "R": 245,
28     "A": 255
29   },
30   "blur_size": 4,
31   "camera_name": "Trust Webcam",
32   "track_index": 1,
33   "shots_list": [
34     {
35       "duration": 5,
36       "msg_info": "Greet someone and say \\\"Hi!\\\"",
```

Figure 3.6: Example of a JSON file

3.2 Plot File Format

This section describes the principles of the future file format. One of the goals of this thesis is to create a customizable application that can handle augmented reality. Filmmaking has an enormous application area. It requires designing a structure that can describe the movie, plot, and all technical details.

It has to be capable of:

1. Add basic film info such as name, description, and technique definition
2. Define the design: color palettes, background, blur, roundness
3. Determine camera connection, keep the IP address or name of the camera

3. REALIZATION

4. Describe all the shots in the scene and guidance data

Also, it has to be portable and available on different platforms. JSON format has been used because of this. Almost all high-level programming languages have methods for encryption and decryption - it makes serialization handy.

3.2.1 Film Structure

1. `Name` - unique key for the film
2. `film_name` - name of the film that will be displayed
3. `film_description` - description of the film
4. `working_directory` - a path to a folder where all assets are placed, and where all shots will be saved
5. `script_directory` - a path to a folder with all the necessary python scripts
6. `delay` - time in seconds of guidance screen before recording
7. `background_color` - background color with an alpha channel
8. `text_color` - font color
9. `accent_color` - main color of all the buttons and elements
10. `blur_size` - an integer value of blur strength
11. `camera_name` - a name or IP address of the camera which is used in a game engine (if the value is empty, it will choose the first available camera)
12. `track_index` - track of your camera (usually 0 or 1)
13. `shots_list` - array list of all the shots

3.2.2 Shot Structure

1. `duration` - time in seconds of current shot
2. `msg_info` - bottom message to an actor (usually a speech or an instruction)
3. `body_position_is_required` - boolean value if the icon is required
4. `body_position_x` - float value from 0 to 1 of the x position (ignored if `body_position_is_required` is false)

5. `body_position_y` - float value from 0 to 1 of the y position (ignored if `body_position_is_required` is false)
6. `body_position_type` - type of the icon (see 5.2.3)

3.2.3 Body Position Type Enum

The project has included font for custom icons, so-called FontAwesome. That allows to expand this structure and add new elements. All icons are placed in a GuidanceUI under the HeadPositionBox and disabled by default. Depending on a structure value icon can be changed.

1. `FULLBODY` - long shot
2. `HALFBODY` - medium shot
3. `WINK` - custom icon with smile

Prototype Demonstration

This chapter is focused on creating an example of usage. It contains sections about the basic idea, film plot customization, assets import, and creation level sequence files.

This example doesn't show the maximum capability of the application. Unfortunately, it was infeasible to visit a film studio with a chromakey and high-quality lighting because of the quarantine limitations. That's why the output video has a lower performance than expected.

4.1 Concept Idea

One of the applied examples is to create a short video postcard of scenery if you can't visit your favorite country. It consists of two shots:

1. big forest scene where the camera slightly moves from the actor high to the sky. The actor has to pretend he's moving and looking up.
2. lava cave. An actor is shown from his back. The camera moves from the media shot closer to him.

4.2 Assets Import

The next step was to find suitable assets for the concept idea. I looked up in the Unreal Engine's Marketplace. Several criteria have been considered:

- assets have to be free
- it's better to be Epic Games content
- files couldn't take up too much space
- they have to be compatible with UE 4.25.4

4. PROTOTYPE DEMONSTRATION

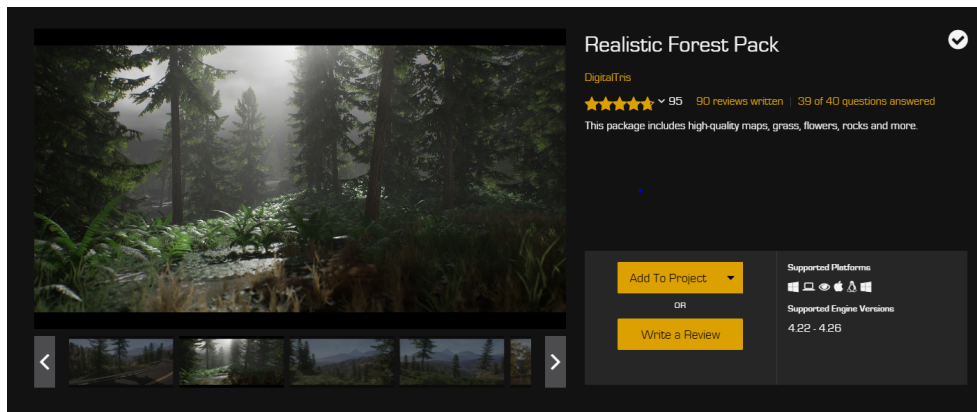


Figure 4.1: Realistic Forest Pack in the Marketplace

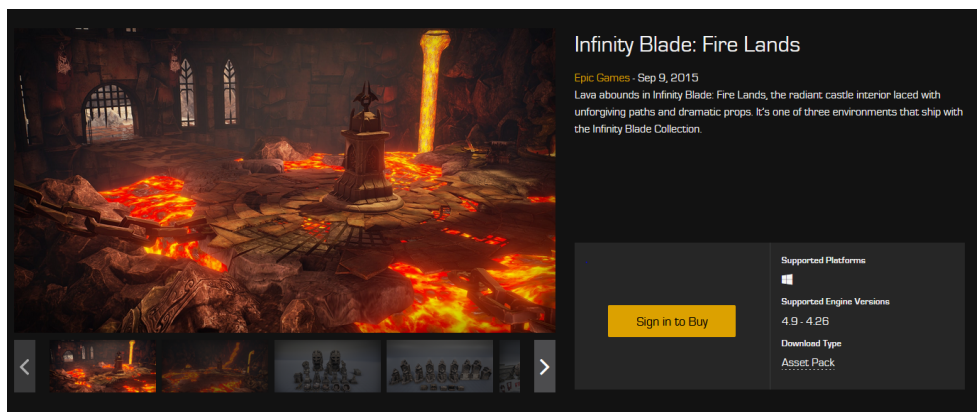


Figure 4.2: Infinity Blade: Fire Lands in the Marketplace

For the first shot was used package *Realistic Forest Pack* by *DigitalTris*. This package includes high-quality demo maps, grass, flowers, rocks, and more [26].

The second shot is *Infinity Blade: Fire Lands* by *Unreal Engine*. It's one of the most used packages for demos.

4.3 Film plot declaration

The next step was to configure the JSON file where all the guidance instructions will be consistent. I changed the name of the film, added a new description, so it's different from the default version. Then I changed the user interface color palette and made the accent color red. After that, all guidance instructions were changed in a `shot_list` array. Message info now tells the actor what to do in the shot, and user will see it on a screen.

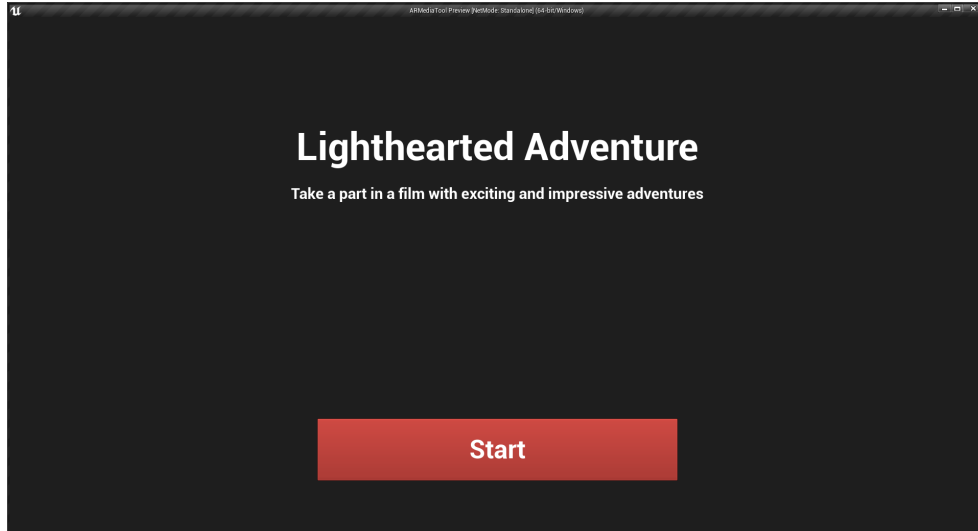


Figure 4.3: Main menu after customization

```
30     "blur_size": 4,  
31     "camera_name": "Trust Webcam",  
32     "track_index": 1,  
33     "shots_list": [  
34         {  
35             "duration": 5,  
36             "msg_info": "Pretend like you're moving forward and look up into the sky",  
37             "body_position_is_required": true,  
38             "body_position_x": 0.64999997615814209,  
39             "body_position_y": 0.5,  
40             "body_position_type": "FULLBODY"  
41         },  
42         {  
43             "duration": 9,  
44             "msg_info": "Stand from your back in the center, you are looking at the lava cave.",  
45             "body_position_is_required": true,  
46             "body_position_x": 0.5,  
47             "body_position_y": 0.5,  
48             "body_position_type": "HALFBODY"  
49         }  
50     ]  
51 }  
52 ]
```

Figure 4.4: Shot declaration in JSON file

4. PROTOTYPE DEMONSTRATION

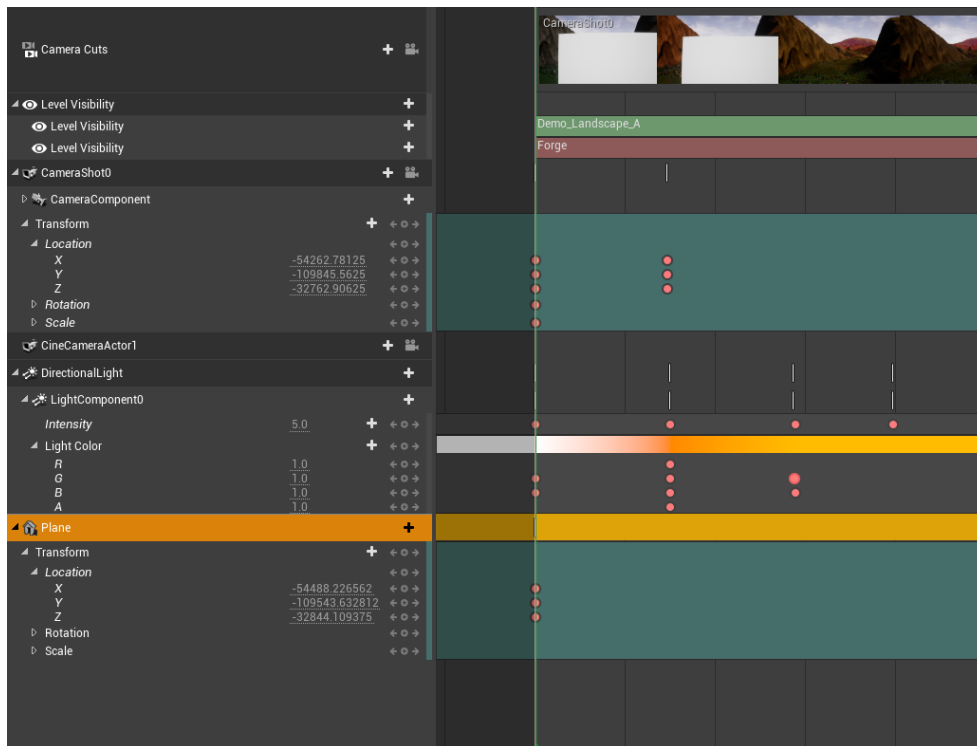


Figure 4.5: Level Sequence file for the first shot

4.4 Creation of the Level Sequences array

The final part was to create pre-prepared scenes and animations. Two Level Sequences were added to a project with a specific name. The name of all the sequencer files must start with a "Scene_", and after goes the index of shot, starting from zero.

Camera Cuts track specifies which camera(s) is used in a shot. The camera is in the right place, and there is a plane in its field of view with a texture of final video.

Actors are animated using keying technology. Components are specified for each frame. Position, rotation, scale, color, intensity - everything can be animated.

Because all the assets in two shots are in the two maps, it has to be included in the sequence. For this, in the *Level* window, there were added two new maps. In the Sequencer, there is a *Level Visibility* track that contains information about maps that are enabled and disabled.

Conclusion

This thesis investigated different techniques in virtual production, live recording, streaming, and rendering processes in Unreal Engine. There were given and tested various techniques. Although the completed analysis showed that there is a lack of suitable solutions for automated film production in Unreal Engine, I managed to create the functioning prototype with the usage of third-party libraries combining with the game engine tools for film-making.

The outcome shows that all principal goals are succeeded:

1. the application can record the data flow from the camera and save it to the repository
2. the suitable GUI has been successfully designed
3. there is a customizable plot film layout containing information about shots
4. there is a queue of all the shots. The application can load them from a file
5. the model implements a transition between scenes
6. user can see the result in a media player and decide whether he likes it or not
7. the system can record the final video and save it into the storage

5.1 Room for future improvement

There is still a gap for improvement. For instance, a framerate for a final recording is low and depends on system requirements. It's the most challenging problem with which I couldn't cope.

However, there are some promising ideas for the future growth of this application:

5. CONCLUSION

- implementation of a recognition system for detection of some problem during the recording (the actor is not on a scene or not doing what he need to do, stands too close or too far, doesn't say anything)
- web integration to unite people from various location
- work with sound
- more customizable guidance with transparent images and more elements
- more interaction with a user (shot depends on user's choice)
- sending files to a external servers or through the email

Bibliography

- [1] 8I LTD *Holo - Holograms for Videos in Augmented Reality*. [software], 2017-09-19, [cit. 2020-12-05]. Available from: <https://play.google.com/store/apps/details?id=com.eighti.holo.android>
- [2] LEMON&ORANGE *Timberland Augmented Reality Campaign*. [online], 2014-09-01 [cit. 2020-12-05]. Available from: <https://www.youtube.com/watch?v=5TZmQPdhpak>
- [3] ZIICON *MANNER - Human Motion Tracking and Gesture Recognition - Interactive Digital Signage*. [online], 2015-05-22, [cit. 2020-12-05]. Available from: <https://www.youtube.com/watch?v=za-pZl1BQjg>
- [4] ZIICON *PAGO-GRANINI - Human Motion Tracking and Gesture Recognition - Interactive Digital Signage*. [online], 2015-06-29, [cit. 2020-12-05]. Available from: https://www.youtube.com/watch?v=pQIirdYa_rS
- [5] ZIICON *RED BULL ZERO: Human Motion Tracking and Gesture Recognition - Interactive Digital Signage*. [online], 2013-05-22, [cit. 2020-12-05]. Available from: <https://www.youtube.com/watch?v=CO2-QbEA0CQ>
- [6] LEMON&ORANGE *Virtual Movies - Augmented Reality*. [online], 2016-11-27, [cit. 2020-12-05]. Available from: <https://www.youtube.com/watch?v=sg5YvW1M4eY>
- [7] IMAJINPERCEPTIONSAV *Lynx Excite Angel Ambush London Victoria.flv*. [online], 2011-05-06, [cit. 2020-12-05]. Available from: <https://www.youtube.com/watch?v=I4tvqalREho>
- [8] SMITHSONIAN'S NATIONAL MUSEUM OF NATURAL HISTORY *Skin & Bones promotional video*. [online], 2015-01-26, [cit. 2020-12-05]. Available from: <https://www.youtube.com/watch?v=7agVb4IG16M>

BIBLIOGRAPHY

- [9] AIRCARDS *Aircards - Augmented Reality Business Cards*. [online], 2019-03-21, [cit. 2020-12-05]. Available from: <https://www.youtube.com/watch?v=xBYDmauqdaw>
- [10] STEVE BURBECK, PH.D. *How to use Model-View-Controller Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC)*, 1987, [cit. 2021-04-02]
- [11] Berman, A. Dadourian, and P. Vlahos. *Method for removing from an image the background surrounding a selected object*. US Patent, 6(134):346, 2000, [cit. 2021-03-04]. Available from: <https://dl.acm.org/doi/pdf/10.1145/1291233.1291302>
- [12] R. Fielding. *Techniques of Special Effects of Cinematography*. Focal Press, 1985, [cit. 2021-03-04].
- [13] BEYER, W. *Traveling Matte Photography and the Blue Screen System*. American Cinematographer, May 1964, p. 266. The second of a four-part series, [cit. 2021-03-04]. Available from: <https://dl.acm.org/doi/pdf/10.1145/237170.237263>
- [14] CINE 24 VFX *chroma key after effects - Perfect green screen in 5 minutes*. [online], 2018-02-11, [cit. 2020-12-05]. Available from: https://www.youtube.com/watch?v=VA2yv_vv_JLIE
- [15] THOMAS PORTER, TOM DUFF *Compositing Digital Images*. Computer Graphics Volume 18, Number 3, July 1984, [cit. 2020-12-05]. Available from: <http://graphics.pixar.com/library/Compositing/paper.pdf>
- [16] Spitzak *The RGBA sample image composited atop checkerboard, as most browsers do not do this themselves any more*. [online], 2018-04-25, [cit. 2020-12-05]. Available from: https://upload.wikimedia.org/wikipedia/commons/0/0b/RGBA_comp.png
- [17] EPIC GAMES *Unreal Engine*. [software], May 1998, [cit. 2020-12-05]. Available from: <https://www.unrealengine.com/>
- [18] MWAWI F MSISKA, LYNETTE VAN ZIJL *From visual scripting to Lua*. Association for Computing Machinery, October 2012, [cit. 2021-03-04]. Available from: <https://dl.acm.org/doi/pdf/10.1145/2389836.2389848>
- [19] K. N. WHITLEY, ALAN F. BLACKWELL *Visual programming: the outlook from academia and industry*. Association for Computing Machinery, 1997, [cit. 2021-03-04], ISBN: 978-0-89791-992-0.
- [20] EPIC GAMES *Blueprint Visual Scripting*. August 2015, [cit. 2021-03-04]. Available from: <https://docs.unrealengine.com/en-US/ProgrammingAndScripting/Blueprints/index.html>

- [21] MOHAN MANUBHAI TRIVEDI *Image processing*. Encyclopedia of Computer Science, January 2003, [cit. 2021-04-03]. Available from: <https://dl.acm.org/doi/pdf/10.5555/1074100.1074470>
- [22] OPENCV *OpenCV*. [software], June 2000, [cit. 2021-04-03]. Available from: <https://opencv.org/about/>
- [23] NUMPY *NumPy*. [software], 1995, [cit. 2021-04-03]. Available from: <https://numpy.org/about/>
- [24] Vizrt Group *Newtek NDI*. [software], September 2015, [cit. 2021-04-05]. Available from: <https://www.vizrtgroup.com/about/>
- [25] Sameek Kundu *FrameCapture - Screen Capture and Video Recorder*. [software], August 2018, [cit. 2021-04-05]. Available from: <https://www.unrealengine.com/marketplace/en-US/product/framecapture-screen-capture-and-video-recorder>
- [26] DigitalTris *Realistic Forest Pack*. [software], 4 January 2019, [cit. 2021-04-05]. Available from: <https://www.unrealengine.com/marketplace/en-US/product/realistic-forest-pack>

Contents of enclosed CD

readme.txt	the file with CD contents description
demo	folder that contains prototype demonstration
├─ FilmBasics.json	example of film plot file
├─ result.avi	video example of demonstration
├─ Recordings	directory with the recordings videos
src	the directory of source codes
├─ thesis	the directory of \LaTeX source codes of the thesis
├─ scripts	a directory containing python scripts for recording
├─ ARMediaTool.zip	ZIP archive with the source of the project
text	the thesis text directory
├─ thesis.pdf	the thesis text in PDF format