Aalborg University Copenhagen

Semester: Master study 4th semester

Title:

DMX in Virtual Production for Film and Stage Performance

Aalborg University Copenhagen

Frederikskaj 12,

DK-2450 Copenhagen SV

Semester Coordinator:

Secretary:

Project Period:
February 2021 – June 2021

Semester Theme:

Master Thesis

Supervisor(s):

Henrik Schønau Fog

Project group no.:
n/a

Members:
Igor Akoulinine

Abstract:

Real time virtual production is a state of the art method to capture the final pixels of combined live action and visual effects in camera in real time. Gaming engine technology allows instant interaction between real and virtual worlds during production. However, control of the on set lighting is achieved by DMX protocol unlike the virtual lighting which is controlled by game engine lighting tools. Author explores the advantages and ease of use of DMX as a common lighting control method. This report will reflect on analysis of the realities and challenges of lighting in virtual production to design and implement system addressing to unify lighting control. This system also lets users choose either to control entire lighting from the gaming engine or external lighting console. The number of experts then were surveyed to make a conclusion on the potential role of DMX protocol in virtual production pipelines. The response from the experts indicated strong interests in this method of lighting control. However some of the respondents did not find it convenient enough. Guidelines for further research of the method and development of the system are suggested.

DMX in Virtual Production for Film and Stage Performance

Igor Akoulinine

Supervisor

Henrik Schønau Fog

AALBORG UNIVERSITY

Master of Science in Medialogy

Department of Architecture, Design, and Media Technology

Aalborg University Copenhagen

Table of contents

Figures and Tables

Attached files

Audio\ExpertInterview.wav

DMXtoolsSourceCodeAndPackage\

Images\Artifacts\AnalogConsole.jpg

Images\Artifacts\AnalogDimmer.jpg

Images\Artifacts\Automata.png

Images\Artifacts\Color probe design.png

Images\Artifacts\cornelius.png

Images\Artifacts\corridor.png

Images\Artifacts\stage.png

Images\Artifacts\System in action.png

Images\Artifacts\The stage.png

Images\Diagrams\Art-net.jpg

Images\Diagrams\Class Diagram (6).png

Images\Diagrams\DMX (1).png

Images\Diagrams\Project schedule.png

Images\Diagrams\RS485 Topology.png

Images\Diagrams\SMILE (2).png

Images\Diagrams\TAM (1).png

Images\Diagrams\UseCaseDiagram (1).png

Images\Diagrams\UseCaseDiagram (2).png

Images\Diagrams\UseCaseDiagram.png

Video\DMXtoolsDemo.mp4

Video\LightingDemoGroup606.mp4

Video\VideoAbstract.mov

Glossary

DMX - digital multiplex. The industry's standard protocol to control lighting fixtures for architectural and entertainment application
Fixture, head, lamp - DMX or light device.
Universe - DMX network that operates 512 channels.
Console, mixer, controller - lighting console.
Cue - the levels of all the channels saved into preset.
Cue stack - collection of cues**.**
Orthographic projection - a method of projection in which an object is depicted or a surface mapped using parallel lines to project its shape onto a plane.
Programmer - memory section of the lighting console that holds the channels levels information to be saved to the cue
HDPR - High Definition Rendering Pipeline, Unity's pipeline for rendering highly realistic environments.
AV - audio/video
Dimmer - device to control the intensity of the incandescent light sources.
EEvee - real time render plugin for Blender.
CCT - Correlated Color Temperature
HSI - color model which represents every color with three components: hue , saturation and intensity.
RGB - color model which represents every color with three components red, green, blue.
RDM - Remote Device Management is a protocol enhancement to DMX.

Abstract

Real time virtual production is a state of the art method to capture the final pixels of combined live action and visual effects in camera in real time. Gaming engine technology allows instant interaction between real and virtual worlds during production. However, control of the on set lighting is achieved by DMX protocol unlike the virtual lighting which is controlled by game engine lighting tools. Author explores the advantages and ease of use of DMX as a common lighting control method. This report will reflect on analysis of the realities and challenges of lighting in virtual production to design and implement system addressing to unify lighting control. This system also lets users choose either to control entire lighting from the gaming engine or external lighting console. The number of experts then were surveyed to make a conclusion on the potential role of DMX protocol in virtual production pipelines. The response from the experts indicated strong interests in this method of lighting control. However some of the respondents did not find it convenient enough. Guidelines for further research of the method and development of the system are suggested.

Preface

This report is the reflection of the Master Thesis in Mediology research and development project. The system which is developed during the project is experimental and might suffer from instability.

Report outline.

This report is structured in the following way: First I will introduce you to the topic, myself, project structure and also will supply you with valuable historic background information on the topic. Then I will analyse the current workflows of real time cinematography and possible use cases for the future system, make a set of requirements and prepare some tests. Afterwards I will take you on the journey of design and implementation. When the system is ready we will test it. Finally I will present you with findings and conclusions.

GitHub repository.

In addition to the materials attached to this report latest versions of the source code and the Unity package file could be found at:

https://github.com/igolinin/DMXtools

Introduction

Figure 1 Typical real time virtual film production set up. Unnamed Netflix production(Nishida, 2021)

In recent years convergence of technology made it possible to capture final pixel visuals in camera in the realtime. The hardware, gaming engines, camera tracking / motion control, computer graphics imagery not only developed but also got packaged into user friendly interfaces.

Now film directors can tell the stories that were not feasible to make the films of some years ago.

Big productions use large LED volumes for the visual effects(VFX) like for example Disney's Mandalorian(Disneyplus, 2021) paving the road for others. Before, the filmmakers were adding VFX to the filmed live action and now they are placing the live action in the middle of the VFX.

Broadcast productions has already adapted different variations of real time virtual productions including in-camera LED walls and blended in rendered back or fore grounds.

Real time gaming engines brought more to the cinematography than just render. They also have physics and interactivity. The behavior of the virtual environment and the object

within it could be scripted to follow the story or to be interacted with by actors or the crew in real time(Visual Effects Society, 2021).
With the acceptance of the real time technology production pipelines have changed drastically. Now what used to be “pre” and “post” simply became “production”. Many technological processes of the production which were sequenced can be executed in co-called asynchronous manner - without one waiting for another to be completed. This makes real time cinematography to be centered around the collaboration between different departments, teams and individual talents.
In order for all parts to collaborate efficiently new tools and techniques already have been and to be created. All the techniques that cinematographers gained over the years of filming reality also should be kept.
One of the most important elements of the cinematography is lighting. New methods of filming bring new challenges with the lighting control. Now it is not only on-set lighting which should be considered but also virtual lighting rendered by the real time engine. Being able to unify the source and method of lighting control is an important condition for successful collaboration.
Unity3D(Unity Technologies, 2020) is one of the gaming engines which is extensively used for virtual productions for films. Titles like Jungle Book(IMDB, 2016), Lion King(IMDB, 2019), Blade Runner 2049(IMDB, 2017) were made with Unity.
Unity3D is not only used by the productions with Hollywood budgets. Small and medium size video production studios are using it as well. User friendliness, high availability of the “how to” information and flexible licensing makes it accessible by almost everyone.
The latest **High Definition Render Pipeline**(HDRP) brings photorealistic rendering and advanced lighting control within the virtual space. However there is a gap that doesn't let users control the real lighting devices from the engine - either manually or programmatically. On the other hand virtual lighting could not be controlled from the legacy lighting control consoles which are used to control the on-set lighting.
This project aims to unify the control of virtual and real lighting on virtual production volumes. In addition I will explore the possibility of application of the same method in stage performance acts which could also use the power of real time engines to achieve great visuals during live shows.

Motivation

First, Let me introduce myself and explain how many of the topics covered in this report are related to my previous occupations and interests.
I was born in Leningrad, USSR and already during school days(1989) learned my first programming language(Fortran). After school I entered Physics of Metals faculty of Saint Petersburg Technical University with specialization in Laser and Plasma Methods of Material processing. However due to the USSR collapse and personal reasons I dropped out of university and began following my other passion - music. As a double bass player I have participated in many successful international bands, collaborated with many famous artists in a number of projects including music soundtracks for play movies. I migrated to Denmark in 1996.

In 2002-2004 I studied computer science at Niels Brock Business College and when in 2005 I decided to stop my artistic career as a musician, I entered a period of freelancing in coding and lighting, both as technician and as light designer.
From 2010 to 2017 full time employed by Comtech(doesn't exist any longer, now part of Nordic Rentals) one of Europe's leading rental and production companies specializing mainly in lighting equipment and outdoor stages.
Bachelor in Software Development from Copenhagen School of Design(KEA) was achieved in 2019.
In autumn 2020 as part of my internship I passed a game development course at DADIU - Danish Film Institute guided game development school. That gave me a great experience of scripting in Unity.
All listed above facts explain my constant interest in making technical solutions within the realm of art. This time I will exploit my skills and knowledge in lighting, programming and computer graphics.

Medialogy and this project.

I often get asked by different people what I'm studying. What is medialogy? I think I found the answer which explains it to most people.
Medialogy is scientific study which covers the intersection of computer and cognitive science.
In this project I'm going to employ software development techniques to help light operators to interact with the real time gaming engine and physical lights to achieve a preferred in-camera look. To design the system I will use my previous knowledge and will collect secondary data on cinematography, human perception, computer graphics and computer-human interactions.

Methodology

This report will cover the applied research on the lighting control which could be divided in two stages. Both stages are of exploratory type and the first one will prepare the ground for the second one.

- Stage one, secondary research of the lighting control and virtual productions based on the published books, articles, interviews and my personal experience. This research will help me to shape the application which in my opinion will help to resolve the problem.
- Stage two, primary research based on the experience of usage of the developed application including expert interviews, the survey and observations of users. Qualitative data received in this stage will then be analysed to form a conclusion on the current situation and guidelines for further research.

Project Management.

This project followed the iterative nature of the agile methodology(Alliance, 2015) and was not initially planned step by step. Instead it was divided into 4 sprint periods Figure 2. The length of each of them was approximately 1 month. The first sprint was mainly coding the proof of concept of the future application. That was done to make technical limitations more visible. Followed by the sprint when no coding was done but the achievements of the previous sprints were analysed along with the background information and current situation in the realm. The sprint three was divided half for coding and half for testing. The final sprint was dedicated to the findings analysis and writing of this report.

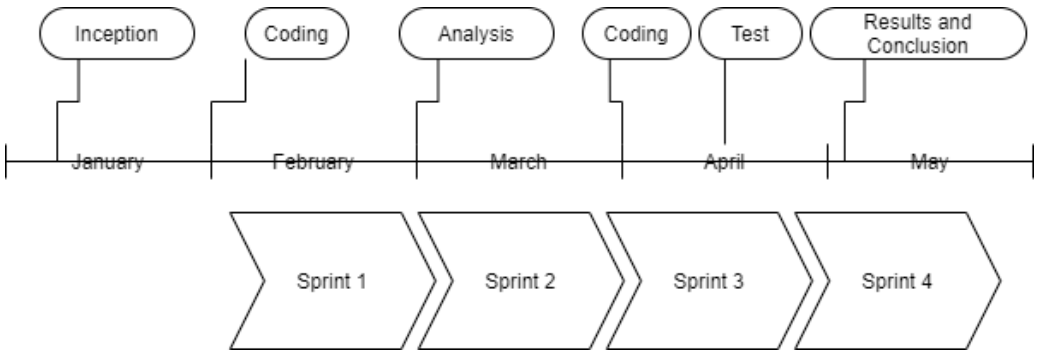


Figure 2. Project's schedule.

Tools.

To achieve this project's goals a number of free helping tools were used. I would like to credit all of them here.

- Images and diagrams were made with **Diagrams.net**(Diagrams.net, 2021) and Paint3D(Paint3D, 2018)
- Text of the report and other text documents was written in **Google Docs**("Google Docs," 2021)
- Internal document references were made with Google Docs plug-in **Cross Reference** (Cross Reference, 2011)
- Bibliography was collected and organized by **MyBib.com**(MyBib, 2021)
- The application was developed upon **Unity3D** platform(Unity Technologies, 2020)
- Versions of the application were controlled by **Git**("Git," 2021)
- The code repository was hosted by **GitHub**(GitHub, 2021)
- The code was edited with **Visual Studio Code** ("Visual Studio Code," 2016)

Background.

In this section I will provide the reader with the essential background information on the topics of lighting and cinematography. Since the beginning of cinematography, which is basically recording the light on the film, these two technologies developed simultaneously, but sometimes one had to wait for another. Very often light that worked on the theater stage could not be used in film and sometimes it was the other way around. I will start with light because people learned to control it long before they learned to record it. Then I will line up the technological achievements in order of their appearance.

Light field

Ancient philosophers (Gallardo, 2000 p4) had different theories on the nature of light and vision. Some thought that light was emitted from the eyes and the others that objects were emitting it.

Skipping the theories evolution over time we can define light as electromagnetic radiation that could be perceived by the human eye. That places light waves in between 400 nm and 700 nm of wavelength. Waves shorter(infrared) and longer (ultraviolet) cannot be seen by the human eye.

The light is emitted by light sources which can vary in intensity(radiosity) and wavelength(color). When it hits the object its further behavior is defined by the physical(optical) properties of the material of that object. Light can fully or partially reflect or bounce back, retract (bend), be absorbed, polarize (selectively penetrate) or disperse(prism effect)(Gershun, 1939).

When light due to the material's properties cannot penetrate through the object it throws the shadow - area on the surface with lower illumination.

Size of the light source results in the shadow's boundary quality - either it is sharp or spread out. The light that creates sharp shadows is called **hard light** and the one that creates shadows with washed-off boundaries is called **soft light**.

The combination of lights and shadows perceived by our eyes forms the image that we recognize as our environment.

Cameras can record the image in a similar manner, but even though they are getting better and better every year they still cannot achieve light sensitivity and flexibility of the human eye.

To improve the qualities of the recorded image and also to communicate to the viewer with the meaningful visual message number of the lighting techniques used by the filmmakers.

Lighting control

From very early days of civilization people used lighting and shadows to play a role in their religious, cultural and artistic acts and performances. In the following subchapters we will follow the evolution of the lighting control.

Before electricity

The first attempts to control and take advantage of the light were placing the objects that were supposed to be lit, in the rays of the natural lighting such as sunlight and fire. F.

Penzel(Penzel, 1978) in his book “Theatre lighting before electricity” demonstrates the evolution of the lighting methods from ancient theaters which were built the way that the stage is lit by the sunlight during the performance hours. Bonfires and torches were also widely used for all kinds of religious and cultural ceremonies. Later, oil lamps and deep candles were used for many centuries. French chemist Lavoisier(Brown, 2018) in 1781 suggested attaching movable reflectors to the oil lamps to achieve directional lighting. From around 1815 gas lanterns began to be used in the european theaters. Figure 3 and Figure 4 show the gas era lighting control system and light fixture(limelight). It is important to mention that all the fire related methods of illumination were very unsafe and often caused fire accidents.

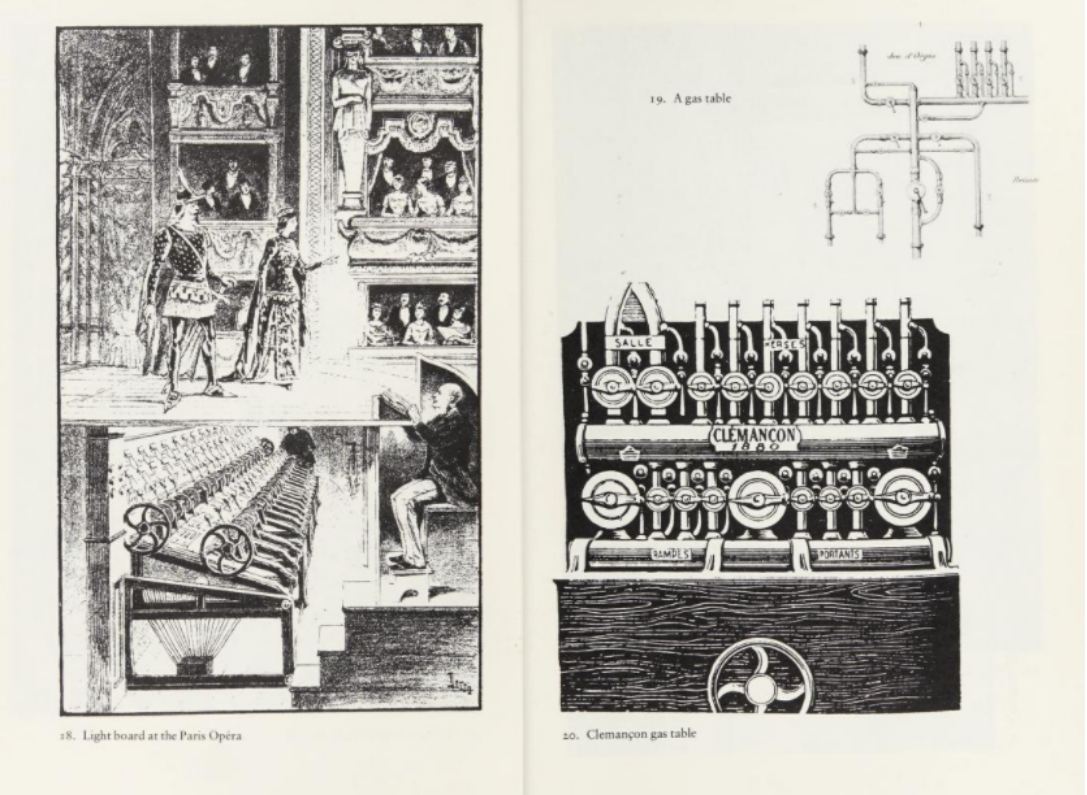


Figure 3 (Penzel, 1978) Gas lighting control systems.

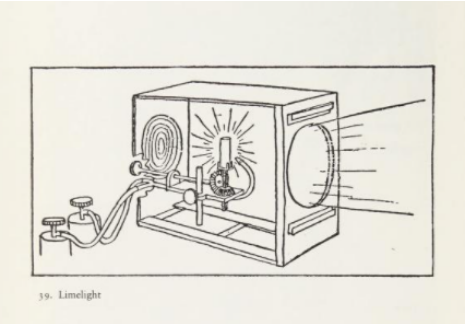


Figure 4 Limelight (Penzel, 1978)

Electric lamps

The electric carbon arc lamp(Whelan, 2016) was first invented around 1800 by Sir Humphry Davy in England and by Vasily V. Petrov in Russia. Practical usage of these lamps began later in the 19th century. Spot projectors based on arc technology could be controlled by shutters.

Carbon arc lamps were later replaced by incandescent lamps when those became brighter and improved their at first short life span. Those properties have been achieved by replacing carbon filament by metal(eventually tungsten) one and use of inert gas inside the bulb instead of vacuum. The other big advantage of incandescent lamps was the ability to turn them on and off and dim them up and down.

Dimmers

The device to control the intensity of the light fixture is called a dimmer.(McCandless, 1958) First commercial dimmers appeared in the 60-70's of the 19th century. They were placed backstage or under the stage and could not be operated remotely. Operators should have to be informed of events on the stage to perform changes of the lighting. Later audio intercom lines were used for this purpose.

Until the mid 20th century all the dimmers were of the variable resistor type and basically burned off the excess of the power when dimming. Figure 5 shows the leaflet for the theater switchboard that was deployed in one of the famous New York theaters in1920's. We can see that those days the dimmers and control surface were actually one unit.

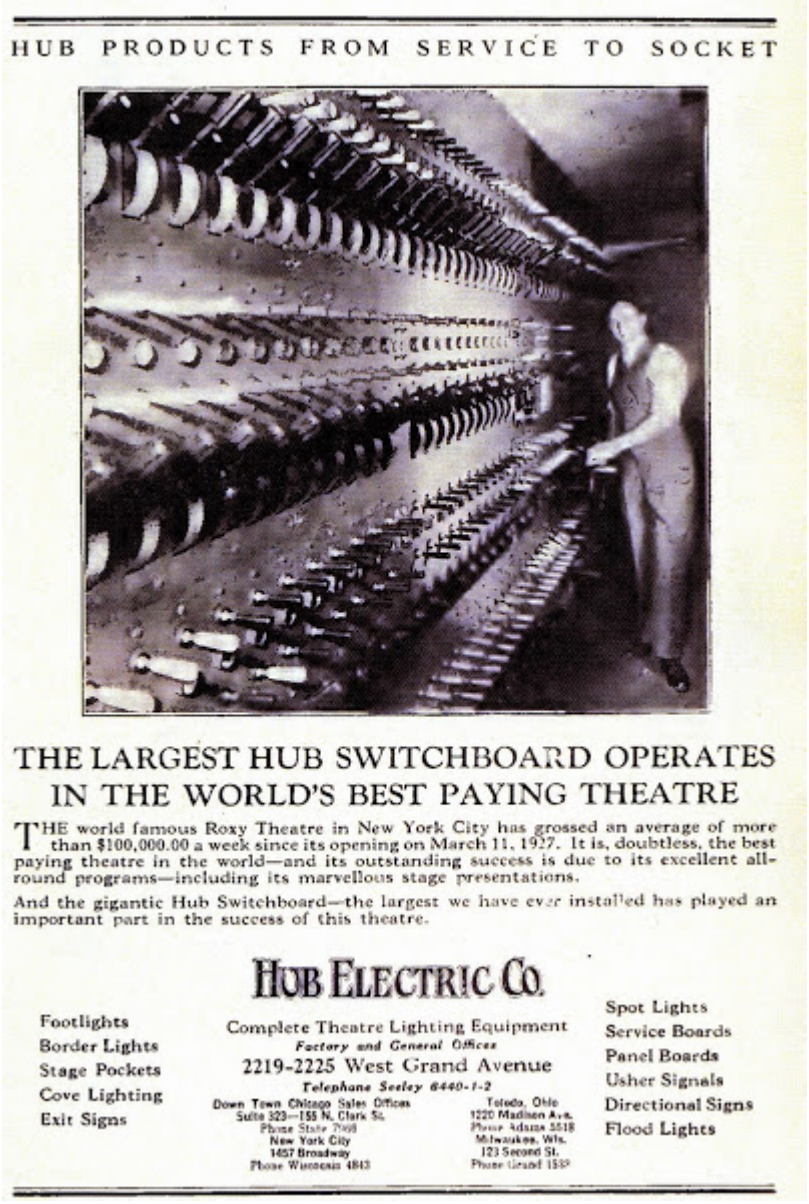


Figure 5 (Foreman & Motter, 2014)

Situation has changed after the first remotely controlled dimmers became mass produced in the 1950th. First, magnetic amplifier dimmer and then silicon-controlled rectifier (SCR) dimmers . It allowed the control console to be moved to "front of house"(FOH) - a publicly occupied area of the venue(McCandless, 1958).

Analog 0-10v

The 0-10V control system was widely used in the days before digital control took over. The method is very simple: changes in control circuit between 0 and 10 volts DC reflect in

proportional changes in output of the dimmer. So 5V in the control circuit would set the dimmer to output 50% of its full voltage capacity.
Biggest disadvantage of this setup was a requirement to use at least one cable to control one dimmer. That made it necessary to use expensive multiplex cables. Digital era opened new horizons for multichannel control.

DMX
DMX stands for Digital Multiplex and now is industry's standard protocol to control lighting fixtures for architectural and entertainment applications. Protocol(USITT, 2018) was created by the United States Institute for Theatre Technology(USITT) in 1986 and was updated by 1990 revision. DMX's purpose was to create the communication standard for interoperability between fixtures and lighting controllers designed and produced by different manufacturers. DMX inherited its physical layer from the electrical interface based on the RS-485 specification.

RS485
RS-485(sometimes referred as TIA-485 or EIA-485 was made to allow communication of the electrical devices in the electrically noisy environment over high length cables. RS-485 along with RS232, RS422, RS423 belongs to the family of serial communication methods for computers and devices. As you can see in the table 1 it beats all the sibling methods by number of the drivers and receivers, max cable length and the transfer rate.

	RS232	RS423	RS422	RS485
Differential	no	no	yes	yes
Max number of drivers Max number of receivers	1 1	1 10	1 10	32 32
Modes of operation	half duplex full duplex	half duplex	half duplex	half duplex
Network topology	point-to-point	multidrop	multidrop	multipoint
Max distance (acc. standard)	15 m	1200 m	1200 m	1200 m
Max speed at 12 m Max speed at 1200 m	20 kbs (1 kbs)	100 kbs 1 kbs	10 Mbs 100 kbs	35 Mbs 100 kbs
Max slew rate	30 V/µs	adjustable	n/a	n/a
Receiver input resistance	3..7 kΩ	≧ 4 kΩ	≧ 4 kΩ	≧ 12 kΩ
Driver load impedance	3..7 kΩ	≧ 450 Ω	100 Ω	54 Ω
Receiver input sensitivity	±3 V	±200 mV	±200 mV	±200 mV
Receiver input range	±15 V	±12 V	±10 V	–7..12 V

Max driver output voltage	±25 V	±6 V	±6 V	–7..12 V
Min driver output voltage (with load)	±5 V	±3.6 V	±2.0 V	±1.5 V

Table 1. Comparison of RS-family serial methods. (Lammert Bies © 1997-2019, 2019)

The main reason for this is that RS-485 uses differential signals sent over a twisted pair of cables. RS-485 is also half duplex meaning that data can only be transferred in one direction at a time. It is not a problem for DMX but becomes a challenge for bi-directional RDM.

Network topology of RS485(Figure 6) is the long line bus with multiple drop-outs for the devices. Terminating resistors are used one on each end of the line to eliminate reflections.

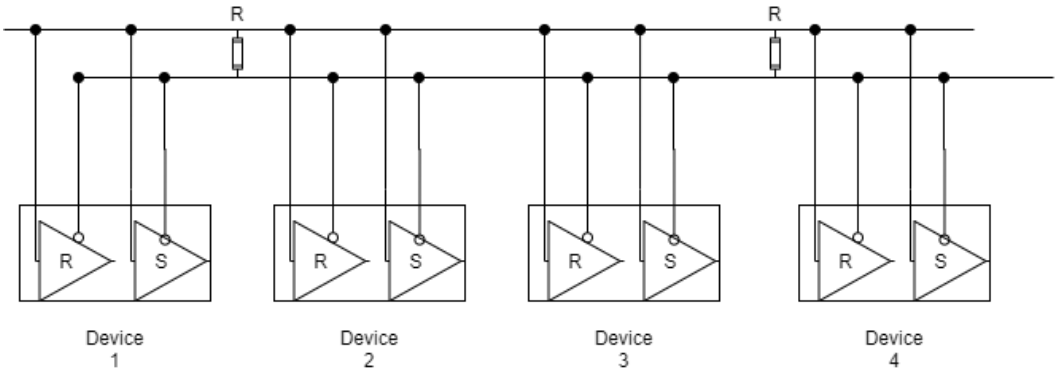


Figure 6. Network topology of RS485 img:I.Akoulinine

DMX512 Data Protocol.

USITT(USITT, 2018) says that data packets consisting of slot 0(START CODE) and upto 512 data slots should be sent in the serial and asynchronous format. Reset sequence appoints the beginning of each packet: BREAK, MARK AFTER BREAK, START CODE(slot 0). The vocabulary I'm using here is taken from the specification, but sometimes the slot is referred to as frame. Some other sources call the packet a frame. We will stick to specification notation.

There are two types of the START CODE: NULL START CODE and ALTERNATE START CODE.

The latter is any non-NULL value of the slot 0. Some values are reserved for future protocol expansion and other needs. The rest of course can be used by developers if needed.

NULL START CODE shows that all the following slots are data slots with values from 0 to 255 decimal. Protocol requires to send at least one data packet with NULL START CODE a second. Timing diagram from the specification paper (USITT, 2018) sets maximum refresh rate for 513 slots as 44Hz. DMX512 data link originating from a single DMX512 source is called DMX universe.

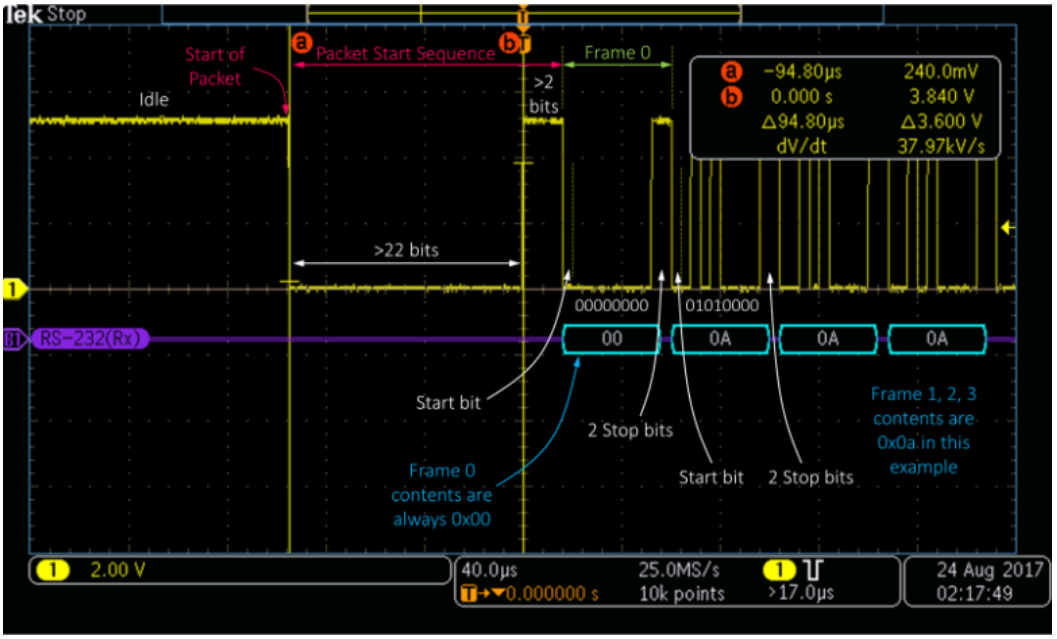


Figure 7. DMX frame.(“DMX Explained; DMX512 and RS-485 Protocol Detail for Lighting Applications,” 2017)

Originally NULL START CODE was also called dimmer code and was reserved for dimmer class data only but now it is used for all types of devices.
DMX512 requires no parity between devices. Each device should know its address and number of channels it uses data from.
Figure 8 shows how DMX works in practice. DMX controller sends DMX data packets upto 44 times a second. Packet is a byte array of length upto 512 variables. Each of them can have value from 0 to 255. Device only reads the amount of bytes it uses starting from its address index. Device executes appropriate function according to the value of the byte with specified index.

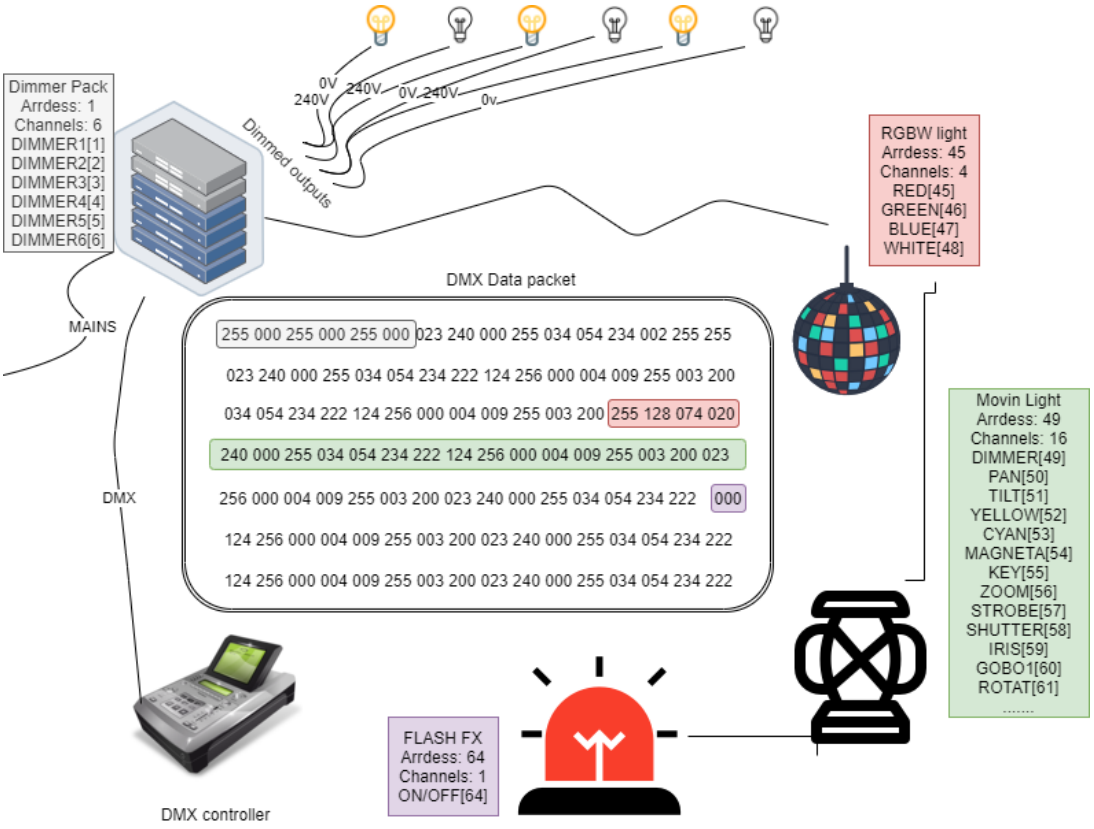


Figure 8 This is how DMX works in practice. img:I.Akoulinine

Lighting console

As was already mentioned earlier, the first consoles were the surfaces of the dimmer and switch units. Operators would interact with the handles and dials of mechanical, back then, switches and dimmers.

First remote consoles appeared in the 1950s(Holloway, 2010 p.4) and were the fader boards 1:1 mapped to the dimmer channels. At the same time consoles began to be equipped with crossfading functionality. Figure 9 demonstrates the method of crossfading two presets A and B. The operator could set intensity levels of in this case 24 dimmers independently for each of two presets and crossfade between them smoothly by gradually and simultaneously increasing one and decreasing the other.

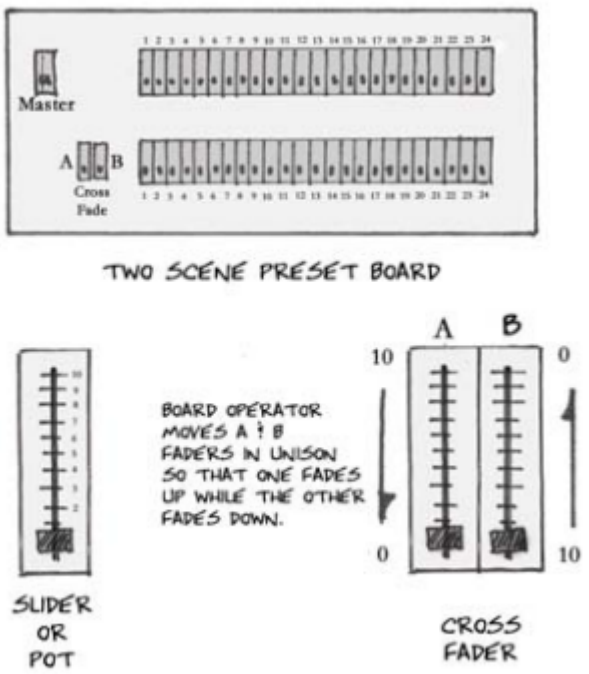


Figure 9 Crossfading technique demonstrated. src: (Holloway, 2010)

The preset of the channels intensity values is commonly known as **cue**. Later, computer and microcontroller based consoles could store cues in memory and play them back in order(cue list) or individually. Each cue can have its own delay time as well as fade in and out times.

The effects variety which could be programmed on the computer based console was getting more advanced but connection between console and the dimmers was still analog 0-10V which we covered above.

Soon after DMX became a standard in 1986 first consoles and dimmers and shortly other devices became DMX driven.

Multichannel devices such as moving lights, strobes and color scrollers could not be conveniently controlled by the fader board. To fix the issue soft patching was introduced(Holloway, 2010). Soft patching is a method of mapping device's channel functions (pan, tilt, red, green, blue…) to the appropriate DMX channel. In this way an operator can select a fixture or group of fixtures and manipulate values of named properties - not the numbered channels.

In the beginning of the DMX era 512 channels seemed to be a large number. With the growing popularity of multi channel devices 512 channels began to look very humble. Some manufacturers to increase precision of their fixtures started to use the 16bit approach. The idea is to use two neighbor channel's 8 bit values and combine them together to extend the number of steps available to go from 0 to 100 percent of total attribute value. That improved accuracy, but made one universe of DMX512 too small to

accommodate all the fixtures for the large shows. To address the problem console makers introduced products that can output 2 and more universes.

Art-net and ACN

As technology developed further and the use of the ethernet network became standard for many artistic applications to carry media information, the DMX's universe data structure was embraced by TCP/IP based sACN and Art-net protocols.

Architecture for Control Networks (ACN) is a collection of protocols made for the entertainment industry and maintained by the Entertainment Services and Technology Association since 2006. Capacity of sACN is over 64000 DMX universes.

Art-net was made even earlier, in 1998 by the company called Artistic License. It has almost a half of sACN capacity but even 32000 is more than enough for any lighting application. I will cover Art-net in detail in the chapters on design and implementation. Figure 10 shows the topology of the Art-net network.

Most professional lighting consoles of medium and large size support sACN and Art-net. There is a large variety of DMX distribution nodes and a growing number of lighting fixtures.

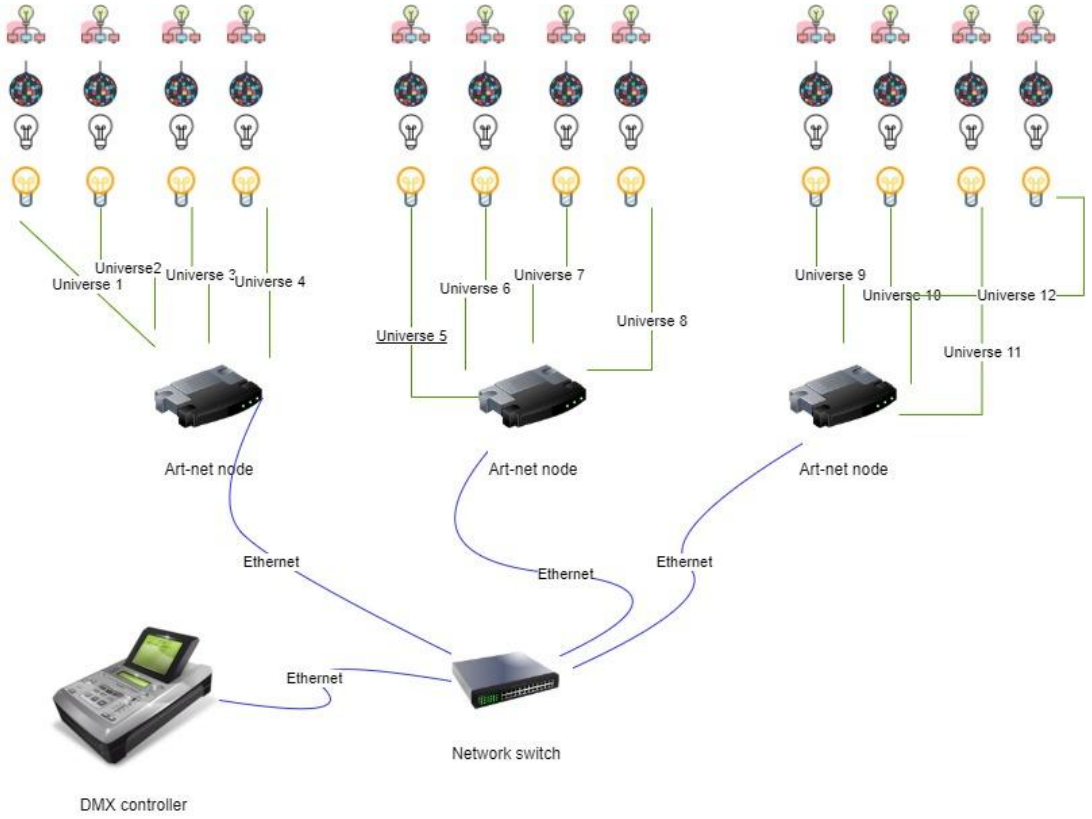


Figure 10. Art-net topology. img:I.Akoulinine

Lighting for Cinematography

Very poor exposure properties of the first film could not let cinematographers use any artificial lighting sources. The sunlight was the only option back then. That is why Black Maria(Figure 11) - first in history film production studio built by Thomas Edison in 1893, had roof that could be opened and was placed on the revolving platform to follow the sun.
Through over 100 years of history of filmmaking lighting technology overcame many technical challenges.
Indecent lamps in the beginning were too dull to be used for cinema lighting therefore it was common to use mercury vapor tubes and carbon arc lamps. Before 1927 due to none-sensitivity of the film emulsion to the red light, tungsten bulbs with the large red component could not be used(Brown, 2018 p4). After it was resolved tungsten and carbon arc light sources were competing but often co-working in different filming sets. A real breakout for tungsten bulbs in brightness was the invention of halogen-tungsten lamps in the early 1960s. At about the same time a new type of source - HMI(Hydrargyrum medium-arc iodide) also known as metal-halide, was developed for television. They had pleasant spectrum balanced daylight output but at first could not be used for filmmaking due to the flicker caused by AC current. Legacy carbon arc lamps used DC instead. The issue was promptly resolved by engineers.
Fluorescent tubes with high CRI(color rendering index) are often used to save watts of power.
LED panels became very popular in the past decade because of the compact size, powerful output and full color control.

Figure 11 Black Maria studio 1893 (Holz, 2014)

Virtual lighting

Virtual or 3D lighting(Gallardo, 2000) is a technique of placing and controlling the virtual light sources rendered by 3D rendering engines or real-time gaming engines. These engines render virtual 3D space to create 2D representation. Lee Lanier(Lanier, 2018,p16) places virtual lighting in the same evolutionary line with lighting techniques of the arts.
Visually virtual light is rendered to look exactly the same as real light would look on the photograph or video. When controlled over script or software tools virtual light retains all the properties of the real light but also adds many others which do not exist in reality. For example you can light an object and it won't cast a shadow or you can cast a shadow but make the object invisible. You can do all kinds of magic tricks in the 3D world.
We will continue the discussion on similarities and the differences of the real and virtual lighting in later chapters.

Visual Effects

Visual effects or VFX is the term that is used to cover a wide range of techniques to combine or augment live action footage with the artificially created or manipulated imagery. First VFX were used in filmmaking from very early days. They were very primitive but yet efficient to impress the audience. Among those there are many that are still in use as for example:

- Stop trick(one of Tik-tok's favorites)
- Combination print
- Hand painting
- Timelaps
- Multiple exposure

Later with the development of the technology a great technique called rotoscoping was added to the list. Puppets and scaled models of all kinds are also widely used.

Computer Generated Imagery in Cinematography

From the 1970's computer generated images and animations era began. A decade later it matured enough to be applicable in cinematography. The Stained Glass Knight character from S. Spielberg's Young Sherlock Holmes from 1985 is considered as the first CGI made character. Later in the 1990's Terminator 2 and Jurasissic park took computer graphics in cinematography to the next level. In the last decades there have been many researchers that propose use of artificial intelligence algorithms such as Generative Adversarial Networks to generate imagery(Karras, Laine, & Aila, 2019).

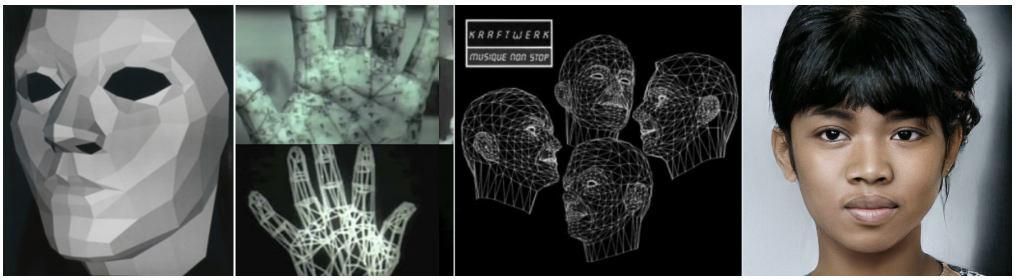


Figure 12. From left to right: Henri Gouraud's face 1971, A ComputerAnimated Hand by Edwin Catmull 1972, Rebecca Allen's Kraftwerk's video 1983, StyleGAN synthesised face (Karras, Laine, & Aila, 2019).

Nowadays CGI is used in cinematography in combination with great new techniques including character rigging, motion capture and chroma keying. LED walls in the real time virtual production are also displaying CGI rendered or manipulated images. We will return to this topic multiple times in the later chapters.

Real time cinematography.

Incredible achievements of computer graphics in cinematography could, until very recent times, only be used in pre or post production. The reason for this is that 3D modeling applications could not render into viewport in real time. They could produce video clips and were great for cartoons or to be used in post production. The help came from the game industry. Game engines were made to render 3D space in real time while being interacted with. That opened the door for real time final pixels in camera video productions. In this setup the game engine renders the virtual environment which is displayed on the LED wall positioned upstage. The actors and the practical set are positioned in front of it and the camera is capturing everything in real time. In addition real camera movements are followed by virtual camera movements within the game engine. Disney series Mandalorian(Disneyplus, 2021) used this technology from already season 1(Figure 13).

Figure 13. Image from the Mandalorian LED volume(Disneyplus, 2020).

Many believe that the future belongs to real time virtual production filmmaking. Many filmmakers say that this way of filming lets the crew members of different roles collaborate in the real time. More on real time cinematography challenges will be discussed later in this report.

Initial Problem Statement

All the information on historical development of the filmmaking and lighting technology along with computer generated graphics signals near or already achieved convergence of them. This convergence demands new tools and pipelines to maximize the efficiency of the real time collaboration. Since out of all the elements of filmmaking we have the focus on lighting we can formulate our initial question as:
How can we unify the way we control on-set(real) and the virtual lights during real time virtual video production?

Analysis

In the chapters of this section of the report I will share my findings on the previous work done in the direction of unifying the way lights are controlled in virtual and physical space. Then weighing the cons and pros I will analyse the workflow of mixed video production and overview possible use cases. Based on that I will create the set of the requirements for the possible solution and finally formulate the question for my exploratory research.

Previous work

Despite the young age of the virtual cinematographic productions there are many tools developed to make the life of the filmmakers easier. Some of these tools were developed addressing specific problems and scenarios and some were actually made for the sibling game industry and then adapted by techs to be used on film shooting sets. In fact right now real time virtual cinematography mainly became available because of the game engines. Rendering engines which were used for animated films and VFX could not do the job in real time. Now some of the 3D modeling and rendering engines have expanded their functionality, mainly in the form of extensions or add-ons. For example Blender recently introduced their users to **EEvee** (Eevee — Blender Manual, 2021) while Maya's users are arguing("r/Maya - Will we get real time rendering like Eevee in Maya 2020?," 2020) if it should have a similar solution. However here I will limit the scope of previous work review to the lighting control in the real time gaming engines.

Art-net and DMX for Unity

A prompt web search helped me to find a number of attempts to integrate DMX to Unity. Surprisingly there were no ready made solutions for sale in the Unity Asset Store. Here I would like to give my credits away to the people that made this wonderful open source project which helped me to get started with my own. I will tell you in brief about each of them.

DMX implementation for USB DMX interface made by GitHub user (davivid, 2016) was not quite finished but after minimal amount of work helped me to make a hole through to be able to send legal DMX data over serial port. It also had a simple editor window application enabling users to control individual channels.

On the art-net side a project made by GitHub user (sugi-cho, 2018) introduced me to the art-net library made in C# which could be found at("CodePlex Archive," 2021). The same repository also contained examples of the devices which gave me great inspiration and guidelines for creation of mine.

Lighting in Virtual Video Production

Virtual production types.

Unreal Engine(“Virtual Production Hub - Unreal Engine,” 2019) in its The Virtual Production Field Guide identifies four different types of virtual film production.

- **Visualization** is usually used in previs. Previs is a concept in filmmaking that helps to visualize the scene before filming. Storyboards could also fall under this term but I’m talking about computer generated visualization. Some of the visualizations are very close to the final content of the scene and sometimes the difference is only in the quality and the resolution of the assets. Often previs includes animation and lighting. According to the article(FAILS, 2020) in the VFX Voice magazine more and more previs studios are going the real-time way and using game engines for the previs instead of using 3D modeling software as they did it before. This fact lets us presume that already on that early stage of the production DMX could be incorporated in the scene and would make lighting inventory and control more organized. In addition the same settings can be used later during shooting.

Figure 14 VFX studio DNEG has launched its own virtual production and previs outfit(FAILS, 2020)

- **Performance capture** is probably the only type of virtual production that we should not be concerned with. It is also referred to as motion capture and is the process of recording motion information of talant's body, face or hands. This information could be then applied to the 3d models. Since there is no lighting information involved, this type of production would not interest us with one rare exception. This project by Diego Prilusky(TED, 2020) is about volumetric video capture which is a mix of performance capture and video shooting. There hundreds of cameras and lights are deployed on set, so I believe DMX could be a good option to control these fixtures.

Figure 15 Diego Prilusky's volumetric video capture volume. (TED, 2020)

- **Hybrid Green Screen Virtual Production** is taking its roots from very early days with the only difference that the screen was black. Later they used the bluescreen and a similar sodium vapor process(Figure 16) also known as yellowscreen. In fact, which color to use for the screen doesn't make much difference and the choice of green or blue colors is only because they are furthest away from the skin color.

  In the beginning mainly the live footage was used to fill the areas of the screen with the background but with the development of CGI the background was rendered by computers. This technology could also be used to remove an actor or his body parts from the shot to replace it with CGI rendered animation. There are many different ways the technology could be used. Sometimes it is also referred to as chromakeing.

Figure 16 The sodium vapor process is used in production of Mary Poppins 1964, (Bedard, 2020)

Now hybrid virtual production variations are also going realtime and they require very delicate lighting of both the real foreground and virtual background. One of the most known lighting issues during this type of production is shadows thrown by actors on the screen can make it hard to remove the chromakey. On the other hand reflection from the screen can spill background color on the actor.

- **Full LED Wall productions** became possible due to improving LED panels specs and their falling prices. We can say that this is the product of merging chroma keying with a back projection screen which was also used by filmmakers for decades. There are different configurations of these setups. From 75 meters in diameter 270 degrees curved screen gigantic volume of Mandalorian to 65” 4K LED screens positioned outside of the train windows(Figure 17).

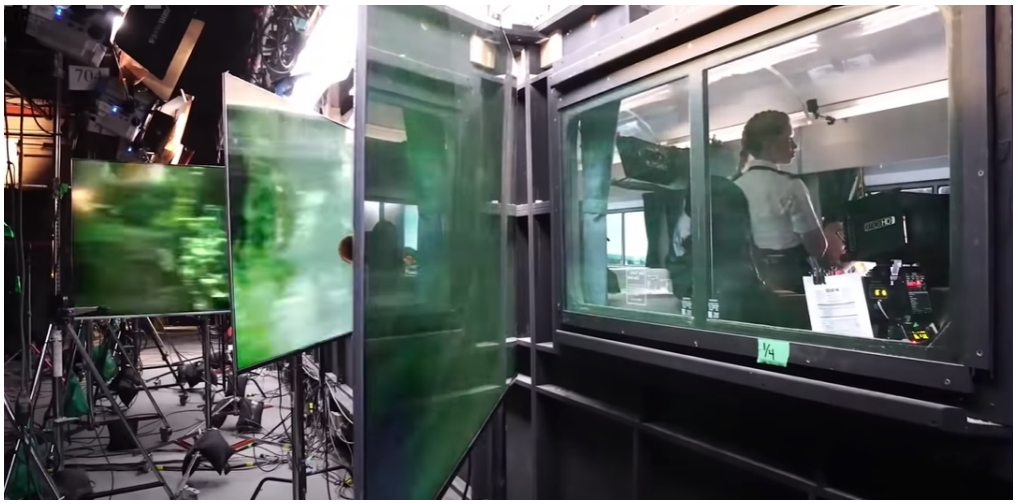


Figure 17 Train set made with 40 of 4K monitors - one outside of each train window. Note many light projectors panels in order to produce hard light. Stargate studio. Src(Unreal Engine, 2020b)

With the only exception of Performance capture real time lighting control of either virtual or real lighting must be applied.

LED screen lighting.

On some of the large and medium LED volume productions filmmakers often use the light from the screen itself. There are many variations of this technique. The screen produces soft light, the tone of which could be easier matched with the background tone compared to for example HMI lamps. The entire volume is a large softbox.
Often they use lightcards - light emitting objects they place in the scene where the camera cannot see them. Sometimes they use square blocks of the LED panels with the lightcards displayed on them and hang them where they are needed.
On the other hand sometimes the LED panel light is not hard enough like on the Figure 17. DMX control of color and some of the other attributes of lightcards could also be an option.

Camera, photography and lighting.

Let us talk here about the quality of the lighting in cinematography. What do lighting crew and photography directors(PD) want to achieve? According to the (Brown.B, 2018, p4,p14) in each frame of the video the filmmaker wants to focus our attention on one or another event or subject and tools like lights, lens and film help them to deliver the right message to the viewer.
There is one problem to achieve this and it is the fact that the human eye doesn't capture visual information the same way film or digital image sensors do. The Human Eye is an incredible instrument. I'm not going to cover the physiology of it or not even the dynamic range compared to the film or sensor.

Let's try to do a little experiment instead. Look at the corner of your room which is perhaps not very well illuminated. May be under your desk or table. Can you see all the details? Move your head a little to change the angle of view. Did it help to collect more information? Turn on the light or use the torch of your phone. Better. Come closer. Did you get all that you wanted to see?
If you film it the way your eye captures it in the beginning and play it back to someone on the TV screen or mobile phone, the viewer will not be able to do all these helping manipulations. If she moves the head, turning the light, moving closer to the screen it in most cases won't give any extra information.
Above example shows that lighting and the composition of the frame should be done the way that it delivers most information to the viewer, but of course in accordance with the narrative.
Narrative, the story which is to be told to the viewers should be the most important here and should drive the technology and not the other way around when the story is the slave of the technology or to the particular set or volume. Good example of this technological story dependency is Disney's Mandalorian where it seems that many narrative lines are there to use their gigantic LED volume. In my opinion the story suffers from the monotony of the desert scenes.

Properties of the good cinematographic lighting

There are many brilliant works on lighting technology in cinematography and on TV(Brown.B, 2018)(Jackman, 2020). If we want to extract the essential properties of good lighting they could be grouped to the following categories.

- Mood and tone
    - Environment(indoor or outdoor),
    - Light sources(natural or artificial, electric or fire)
    - Time of the day
    - Aesthetic stylization
    - Light direction(fx. backlight)

- Exposure
    - Visual balance(enough light but not to much)
    - Contrast

- Depth and separation
    - Layers with different illumination
    - Separation of the actors from background
    - Texture(comes from the shadows)

There are a couple of things to avoid when lighting for cinematography.

- Avoid front light, the one which has an axis close to the axis of your lens that makes all the textures flat. Please look at all the photographs you made with the flash.

- Avoid white clothing and white walls. White surfaces give a lot of uncontrollable reflections

Use cases.

Exploration of the virtual production types and properties of quality film lightin let us create a set of the use case scenarios which will be the solid ground for us to build future system requirements, functional and non functional. We can also set our imagination loose and add some use cases that are not related to cinematography or stage performance. There is probably a need for coordinated control of lighting in virtual and real space on music and theater stages and in museums or educational institutions. Further in this chapter I will list all the use cases found with descriptions and the diagrams.

1. Larger virtual video production(Figure 18). In this scenario we assume that the team has already physical lights and the lighting console on set and either LED or green screen is present. Rendering is happening in real time. Rendered material could be displayed on the LED or virtual monitor to be composited with the foreground image.

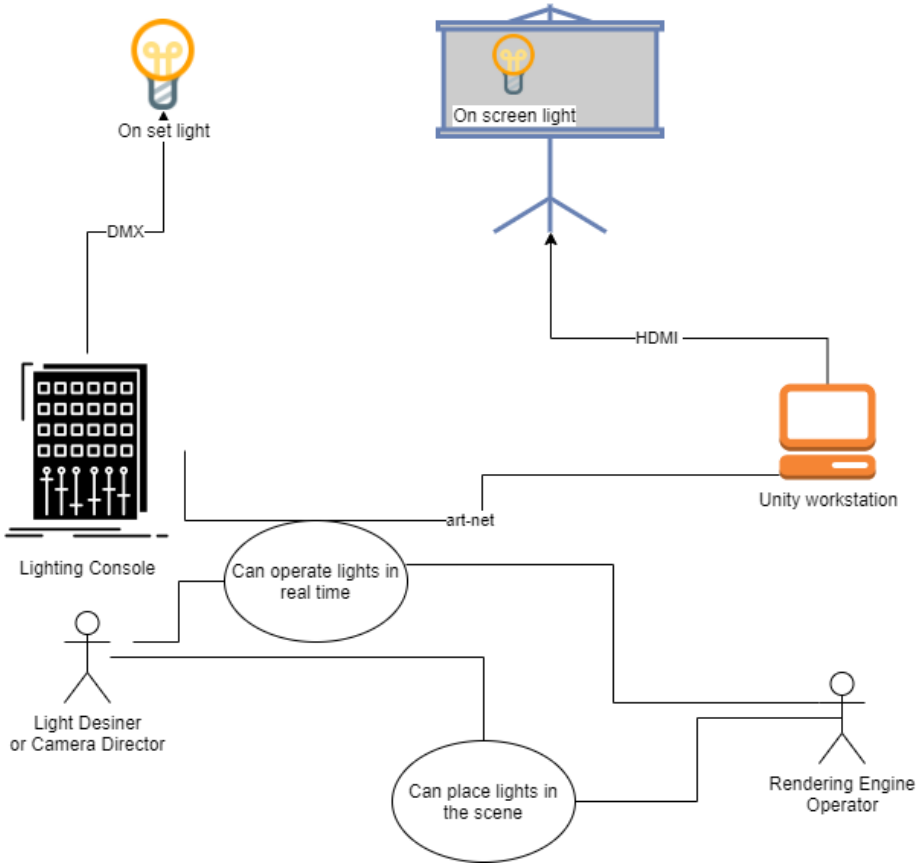


Figure 18. Larger virtual video production.

Lighting console is present and the network-connected to the Unity workstation. Practical lights are connected to the console either by DMX or art-net

2. Small studio or VP enthusiast production(Figure 19). This use case covers smaller filming crews with low budgets. For these teams controlling the real on-set DMX lights is rather a hard task if they don't have a DMX console on site. Also in some situations they would like to match the color in the scene with the color of the light that falls on the actor or elements of the practical set. For this reason translating the color in the virtual scene to DMX values and sending it to the real lights would be a very useful scenario. The lack of the console to save light presets also known as cues can be compensated with the feature of saving the presets to be played back later. In this way a little extension for Unity can replace the real console and save some budget resources which could be used elsewhere in production.

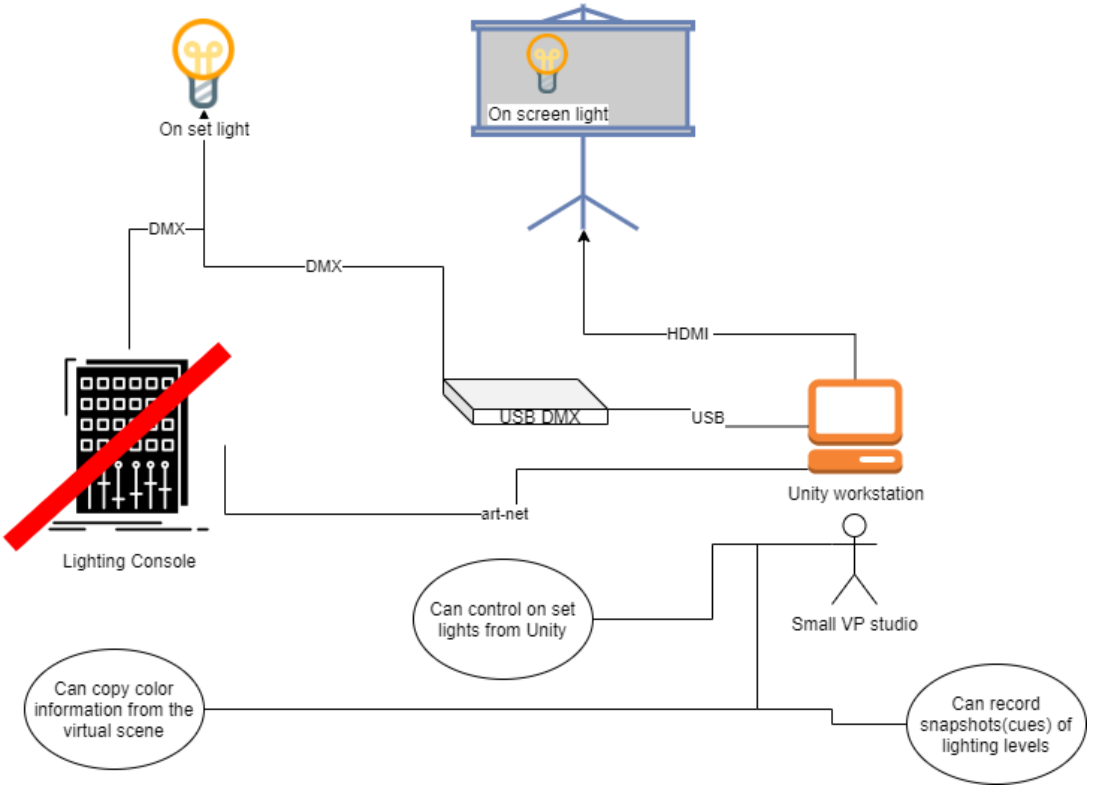


Figure 19 Small studio or VP enthusiast production.

3. Live show(Figure 20). This set up could be used in live music shows or theater plays. Projection on the backdrop of the stage or LED Wall positioned upstage is not exactly new but before the image or video used for this purpose was pre made. In the last decades media servers, DMX or MIDI controlled, were used for this. Now as everything is going real time rendered we can deploy Unity or its likes to create the image. Here only your imagination is the limit for all the different combinations of the lights on the stage and the environments, lights and VFX on the screen. The biggest advantage here of the real time compared to the pre recorded content is that things often do not go exactly after the timeline. This type of lighting during the light shows, when the operator doesn't know what is coming next is called busking and is a very common technique in the lighting of

the live shows.

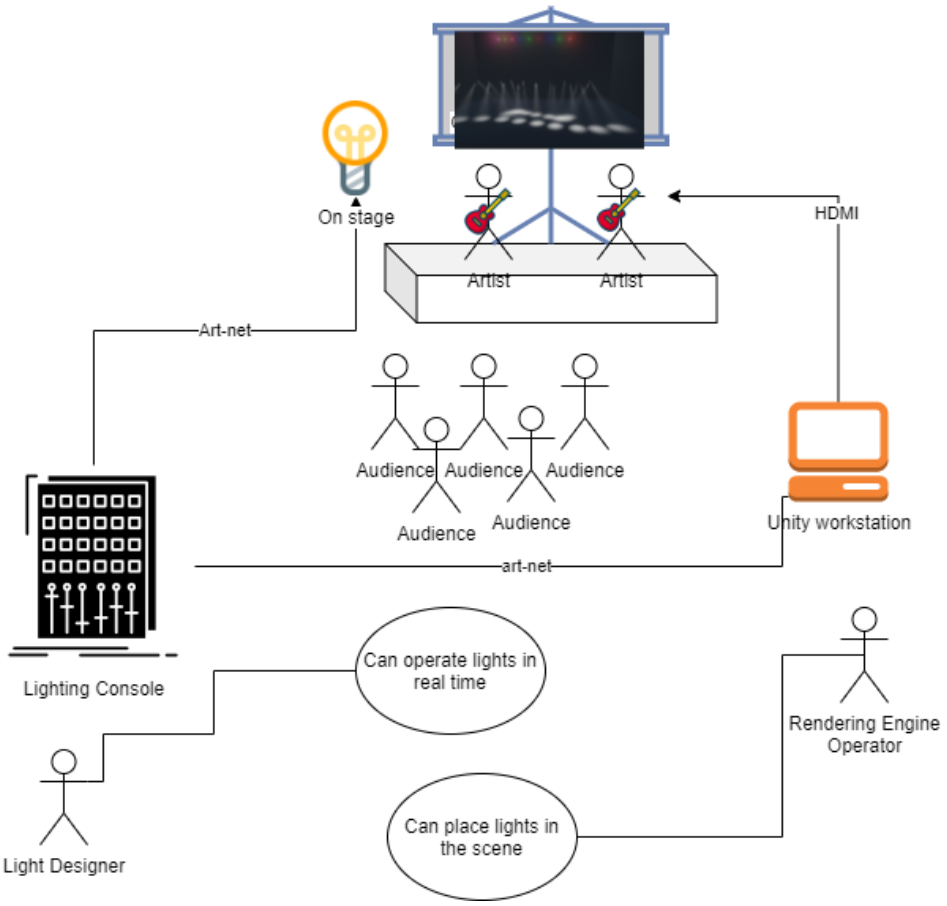


Figure 20 Live show.

4. Museum installations. Real time gaming engines are used more and more often in museums. Large and small LED screens and projection screens display historic and fictional scenes. For better immersion effect physical lights must interact with the events in the virtual space. Considering the range of DMX fixtures available on the market, control of these lights DMX control will be the best choice. Of Course there are more and more new light sources on the market which could be controlled by wireless protocols like WiFi and bluetooth based ZigBee and EnOcean but the range supported by these protocols devices is very little compared to DMX.

Requirements engineering.

Based on the above use cases, a set of requirements for the system that addresses to cover blank space in control of the lighting during virtual production could be generated. Some of these requirements are derived from more than one use case and vice versa some of these use cases generate more than one requirement. In my opinion this fits with the spirit of the matter of cinematography in which any means are good for the perfect, in the sense

of the storytelling, shot. The same here could be applied to the non cinematographic use cases.

Software requirements.

Nonfunctional

1. System should contain of
    a. Unity editor console application
    b. Unity editor preset recorder application
    c. Prefabs of virtual lighting fixtures
    d. Prefabs of twinned real fixture personality

2. Compatibility
    a. Software and OS
        i. System should be able to run on Unity.
        ii. System should be able to integrate into the HDRP project.
    b. Hardware
        i. System should support USB DMX device

Functional

1. System should hold the DMX data for a number of the universes.
    a. Amount of universes should be either pre-defined or user defined.
    b. DMX data should be accessible by virtual lights and sent to real DMX fixtures.
    c. DMX data for all universes should be accessible for reading and writing by public API method

2. Each system's universe should be able
    a. To be set as dmx out univers
        i. This universe's data should then be able to be sent over a USB DMX device to the DMX enabled light fixtures.

    b. To be set as an art-net device to be used to receive DMX data via art-net protocol from an external DMX controller.

3. System should be able to control each of 512 channels of each universe individually.
4. System should be able to do full black out(no output)
5. DMX fixtures
    a. Should be packed as prefabs
    b. Should unpack to break reference to the prefab object.
    c. Should contain light component
    d. Should derive from the same base class to achieve unified discoverability
    e. Should have access to read DMX data of the assigned channels

    f. Should have public interface to change universe and address values
    g. RGBW lights
        i. Should change RGBW color components according to the DMX value of the appropriate channels.
        ii. Should be a point light

    h. Moving Lights
        i. Should change RGBW color components according to the DMX value of the appropriate channels.
        ii. Should be a spot light
        iii. Should update pan and tilt orientation of the light beam

6. Twinned real fixtures
    a. Should be packed as prefabs
    b. Should unpack to break reference to the prefab object.
    c. Should contain light component
    d. Should derive from the same base class to achieve unified discoverability
    e. Should have access to read DMX data of the assigned channels
    f. Should have public interface to change universe and address values
7. Device patch and discovery
    a. System should find and list all the DMX fixtures placed in the scene.
    b. Users should be able to change the DMX address of the fixture from the list.
    c. Users should be able to select one or more DMX fixtures.
    d. Users should be able to select all DMX fixtures.
    e. Users should be able to reset selection of all DMX fixtures.
8. Fixture control
    a. System should list the controls for each attribute of the selected DMX fixtures
    b. Users should be able to change DMX values of all the channels of all the selected fixtures by the attribute slider.
    c. Deselected fixtures should remain their DMX values.
9. Cue recording
    a. System should be able to record the snapshot of all DMX data at the time.
    b. Users should be able to save the snapshots(cues) and name them
    c. Users should be able to save the collection of the snapshots per scene.
    d. Users should be able to play back the cue.
10. Color probe
    a. System should be able to detect the color of the light falling at the at particular position
        i. System should be able to place the probe in particular position in the scene(in front of the camera)
    b. System should send color information to update the DMX data object.

c. System should send color information to update the real DMX fixture connected to the DMX-out line.
d. Users should be able to correct the color information.

Final Problem Definition

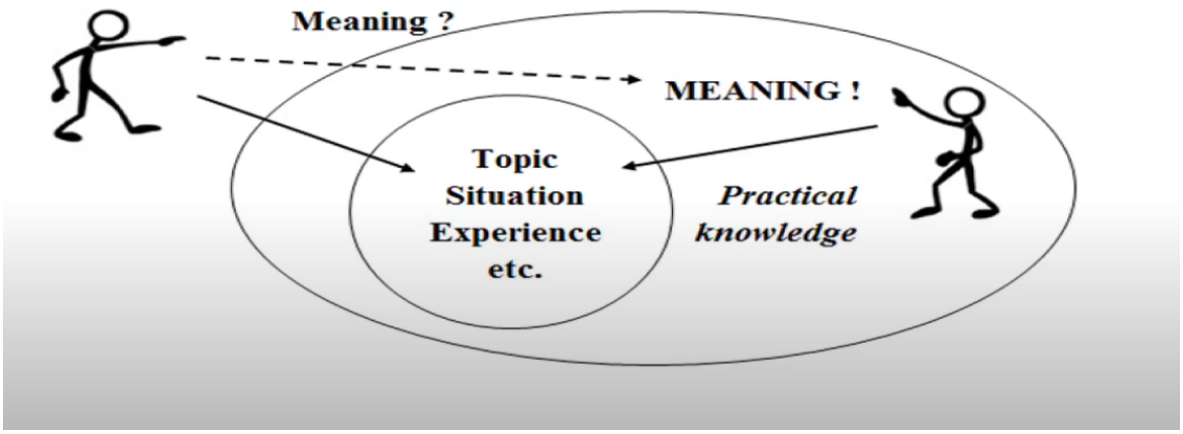
The system that will be built according to the above requirements will help us to answer the question of this research project. In such a system we can test all our assumptions about the convenience of the unification of the virtual and real lighting control by using DMX protocol.
Opinion of the domain experts and their reflections on the experience of using the proposed system will help us to answer our predictive question.
To what extent could DMX be used in lighting control of virtual productions for film and stage performance?

Methodology.

To explore the applicability of DMX in the virtual production the number of methods to be used to collect the feedback information from experts and potential users. The reason for the decision to use more than one method is not just a flair to try it all. My research is touching subjects that are emerging and developing at the moment. Yes, there is nothing new in controlling the lights with DMX - the technology was there for 35 years now. Yse, realtime rendering(gaming) engines were also there for a couple of decades. Virtual productions are also getting more and more as a common way of making films. I presume that it will be very difficult to find experts equally profound in both real time engines and DMX lighting. Therefore I chose to tackle it in three parallel directions in the hope that even minimal results from each of them will deliver some valuable information or innovative ideas.

Interview

To compose my interview I chose to use the approach called Problem Centered Interview(PCI)(Döringer, 2021)(Figure 21). This approach is often used in social science to interview the expert. It is based on the researchers prior knowledge and made in the form of dialog and almost a discussion. Must haves for this method are follow up questions. This method fits in the current situation because the author can be considered to be a domain expert to a certain degree.



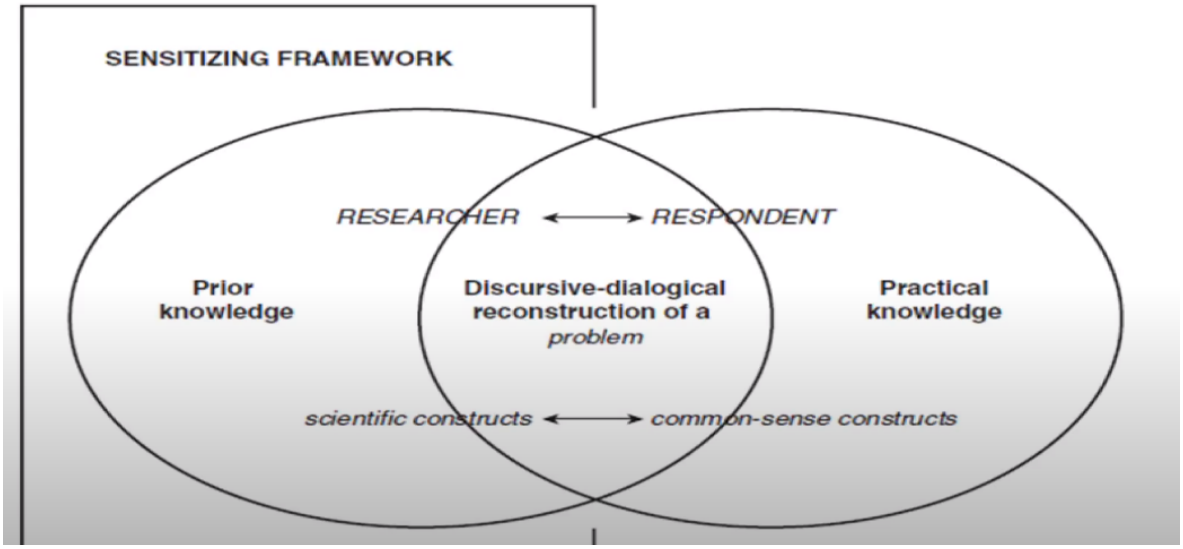


Figure 21 Problem Centered Interview(Döringer, 2021)

The questions of the interview were lined up as follows.

1. What is your background on real time rendering engines?
2. What is your background on virtual lighting?
3. What is your background on practical(real) lighting?
4. What is your background on DMX control?
5. What are the day-to-day lighting tasks that you need to complete?
6. Can you describe how you accomplish this task with your existing solutions?
7. What are the general disadvantages of your existing solution and how do you solve them?
    a. What is lacking today in how you manage this process?
    b. What are your current unmet needs?
8. Conversely, what are the general advantages of your existing solution?
    a. What are the technical advantages of existing solutions?

9. Why/How did you originally pick your existing solution?

10. Would you prefer to control real lights from Unity or a standalone console?
11. Would you prefer to control virtual lights from Unity or an external console?
12. What do you think is the future of lighting control of mixed video production? (mix between virtual and real sets)
13. Do you think DMX has a future in the virtual space of mixed video production?
14. If you had to buy such a solution what would be consideration points?
    a. Price
    b. Integration with other systems
    c. Ease to use
    d. Unique tech features
    e. Support
15. Follow ups.

Even though PCI recommends having topics instead of directly formulated questions I decided to keep them this way to discipline the researcher and avoid falling into the chaos.

Survey

As mentioned above, lack of the experts regarding the entire realm of my research topic made me search for opinions outside of my circle. The community of Virtual filmmakers on Facebook would be a great audience to get an opinion from. That would be a great stimulus for me to develop application to the production or near production level of readiness.

When my application was ready for usability test I have published a post with the short description of my application and links to my repository on GitHub, Google Forms questionnaire, and

I have embedded the button with the annoying red font pressing on which would open the browser window with my survey.

Figure 22. Survey button.

The button is never displayed again after it has been pressed. According to my installation manual CueEditor window functionality is displayed towards the end and therefore users would have already tried the console and have an opinion to answer the questions.

Questionnaire structure.
The Google Forms published questionnaire was divided in two sections: one two collect opinions regarding my research question and the other one two collect usability and quality of the software.
First of all I would like to emphasize that I was not interested to know or consider any of the respondents demographic data. Therefore there are no questions about age, nationality or education.
Instead, for me it was very important to know the background people were coming from. Precisely if they had background in rendering engines, virtual lighting, practical lighting and DMX control.
Then there is a group of questions targeting lighting control placement rendering engine or external console.
Then there were two questions which support or disprove some of my design decisions.

- Would you like to have an art-net out option?
- Would you like to have the DMX-in option?

Then there were two questions which support or disprove the presumed use cases.
Section two consisted of a pretty standard set of software evaluation questions.

- Ease of installation
- Ease of use
- Ease to learn how to use
- Hardware compatibility
- Operating system compatibility
- Ability to integrate with other apps
- Consistency with Interface
- Overall reliability
- Overall performance
- Software intuitivity
- How likely are you to recommend our software to others?

In the end there will be questions that both would help me to improve the software and show new view angles on the problem.

Observations

Here I must not only include the observations of people using the software but also people showing their interest on the topic when I demonstrate this software without their direct participation.
Here is an example case.

Half way through development I was contacted by a group of students who's project supervisor was also Henrik Schønau Fog. Their topic was "Lean-Back Machina: Attention-Based Skippable Segments in Interactive Cinema"(Rasmussen, Persson, Raursø, & Petersen, 2021) . They needed rapid light change in the scene on both sides of the LED screen.
Sounds like the right task for my application. We agreed to schedule the demo session where I would demonstrate the program in action and teach them on DMX. They did not have any prior knowledge of DMX. The entire session was recorded on video.
Observation of the session during playback helped me to discover new design ideas for my application but also to see where DMX comes hard to grasp for a novice.

Design

Design Goals

When approaching the design phase of any software project it is important to define a set of design goals that will make design decisions easier in the future. Process of software development is full of tradeoffs and these goals will play the role of road signs when you need to choose between, for example, look and performance.

- Usability - should be easy to use
- Productivity - should help to optimize pipelines
- Scalable UI - should fit into Unity editor windows layout
- Reliability - should run always
- Compatibility - both hardware and software.
- Fit for purpose - should get the best from lighting consoles and Unity editor
- Convenience - reduce repetitive actions by automation.
- Maintainability - should be easy to update
- Customer experience - this one is always very important for professional tools
- Low coupling - elements within the system should not be tightly coupled to each other, but also the system could be placed or removed from any project.

Data model

DMX data by itself has a very simple structure. As was already mentioned earlier, the DMX universe is just an array of bytes with the length of 512 elements. That means that the entire DMX data construction is an array of arrays of bytes or simply two dimensional arrays of bytes. So in pseudocode we can declare DMX data as

```
dmxData byte[number of universes][number of channels]
```

A dictionary could also be used but it cannot be serialized by Unity(Cooper, 2012).
I have chosen to lock the number of the universes to 8. It will satisfy most of the set-up's needs. In the future this number could be made user selected.
DMX data should mainly reside in the memory as it is dynamic state data and a matter of constant change. That is similar to the programmer data in the lighting console. **Programmer** is basically direct DMX output on all the universes. When the user saves it, it becomes the **cue -** recorded preset of the output.
To save the cue I use another data structure called Cue which aggregates the DMX data with the user chosen name.
Collection of cues form the cue stack. Cue stacks could be saved on disk and belong to the particular scene.
Method of updating main DMX data should also be the choice of the user - either it comes from the console itself, remote network controller or cue editor.

Class structure

From an object oriented point of view systems architecture follows both mediator and observer patterns. On the Figure 23 you can find ArtNetData class(green outline) being aggregated in almost every entity on the diagram.
Console, however, doesn't send the DMX values directly to the fixtures, but updates the main data object instead like in the mediator. The fixtures are all subscribed to the changes of the main DMX data exactly like in the observer pattern.
We can also see inheritance as all the different DMX devices derive from DMXFixture class - it really helps to discover them all in the scene.

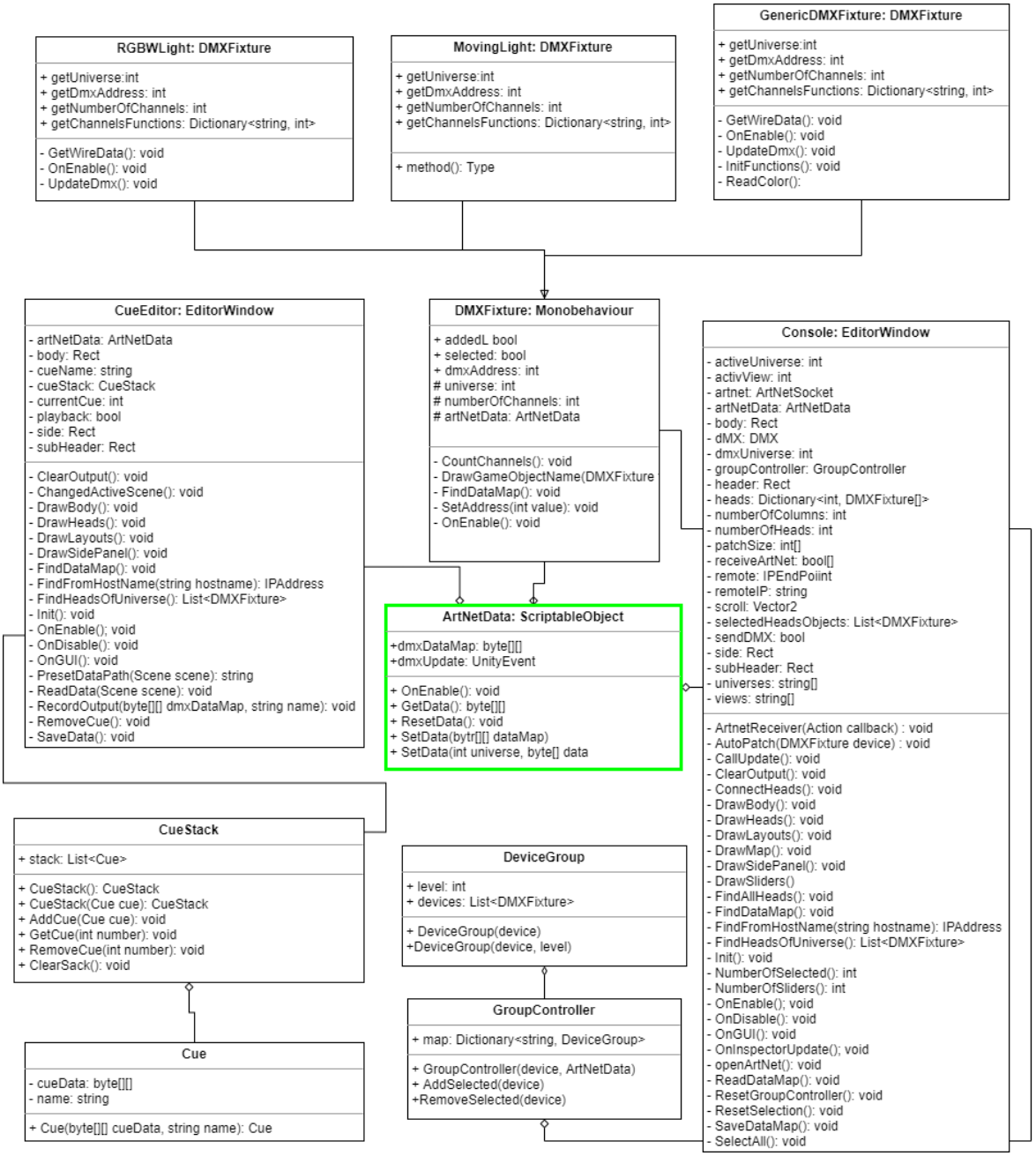


Figure 23. Entire system's class diagram.

UI design.

When designing the user interface I had to keep in mind that Unity doesn't have all the GUI elements that are available for Windows or HTML. For example the dropdown box is missing so I had to replace it with Select Grid which is a bar of buttons and takes much of the screen space.

Therefore I had to use breadcrumbs UI pattern instead of the dropdown menus which I planned in the beginning. There were some other challenges like missing toggle group. Unity also has another system of GUI for editor windows which is called UIElementst, but to me it looked to comlexed.

I have used a hierarchical design approach. In the first row there are buttons of general purpose.

They affect the entire system. Below, there is a universe switch row. Under there is the row of universe options like DMX out universe and receive art-net. It is hidden when the 'All' universe button is selected, elements which are single universe specific are hidden.

Below there are two view select buttons 'Heads' and 'Channels'. If channels selected when the universe is selected list of sliders, one for each DMX channel is displayed for.

If universe is enabled as art-net in universe - grid of labels displayed with current values of the DMX channels instead of the sliders.

If the user selects the 'heads' view all the heads which belong to the selected universe are displayed. Each head is selectable.

If the user selects one or more heads, the side panel will display the list of the sliders for all the attributes selected fixtures have.

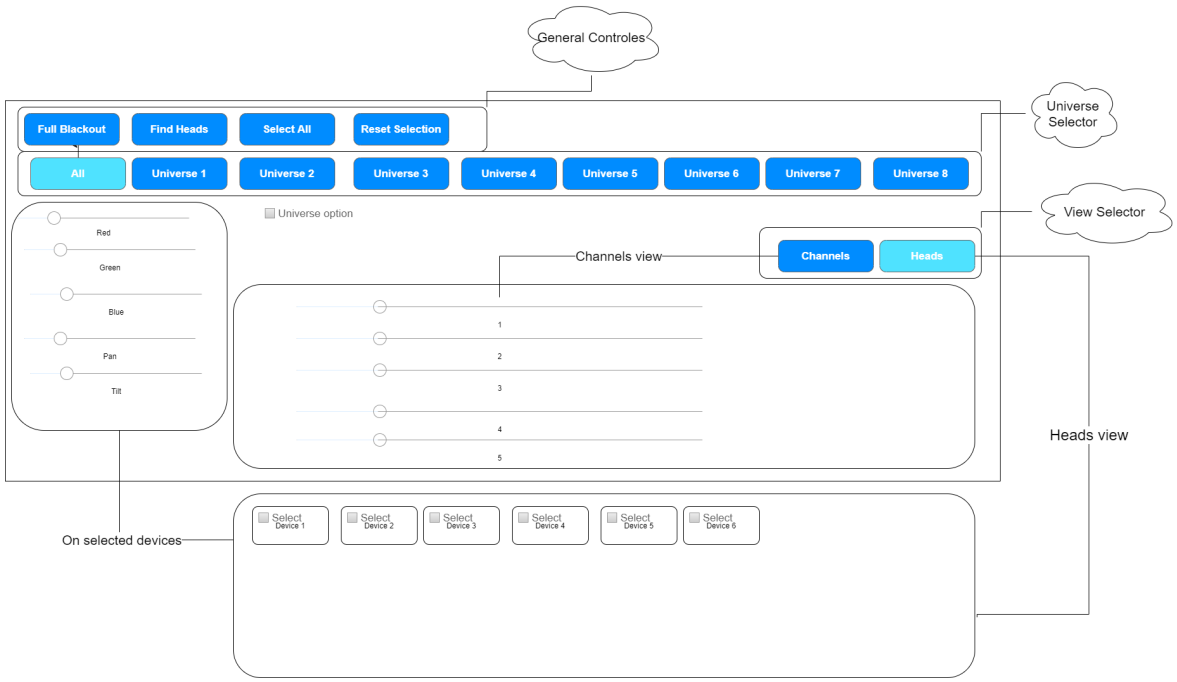


Figure 24 Mockup of the console's UI.

Color probe.

When this project just got initiated one of the first feature requests for the possible solution I received from this project's supervisor was matching the real DMX lights color to the light color in the virtual scene. At first I was not sure how to measure the color of virtual light falling on the object in the environment.

During my literature research on the methods of illumination measurements I came across an interesting one called **mean room surface exitance** (MRSE)(Cuttle, 2015).

The same author also introduces the **cubic illumination** concept, which is the spatial illumination distribution over a point and characterized by the illumination value of each cube's facets.

This method is made for real interior illumination and addressed to help interior and light designers(Duff, Antonutto, & Torres, 2015).

I can employ this method in the virtual environment by using virtual cameras to capture the surface and then analyse the color information. Figure 25 demonstrates the conceptual design. More on color probe in the implementation chapter.

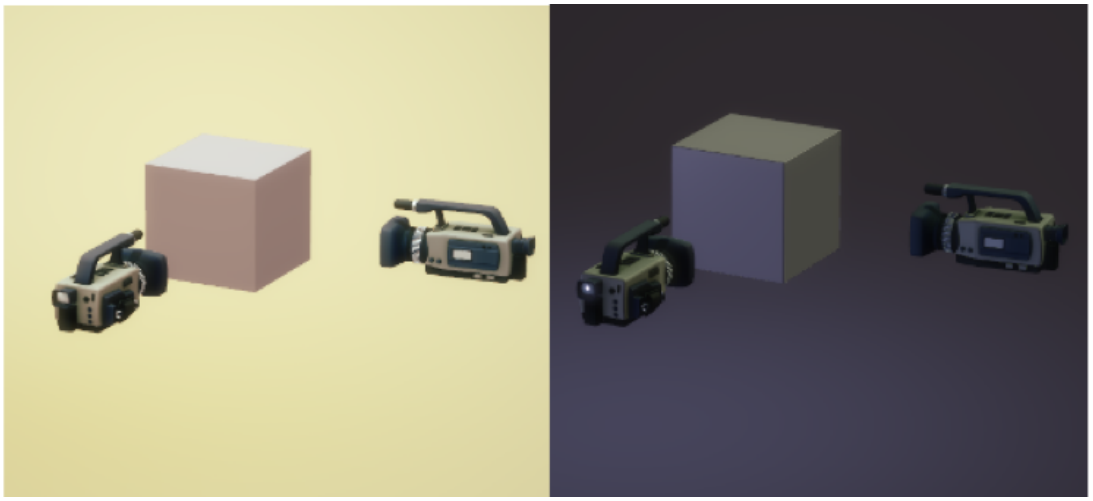


Figure 25. Using cubic illumination concept to capture color information.

Implementation

This chapter will reflect on the implementation phase of the application development. I will go through all the challenges and solutions. I will also discuss possible alternative solutions and why I did not use them. Readers who are not interested in knowledge of the coding details can skip this chapter. All the source code fragments in this chapter are in C# and are using Unity3d scripting API (Unity Technologies, 2020). I will start with general issues and then take a detailed look at the elements.

General Unity notes.

Unity is the gaming engine. Its original purpose is to design and build the games. Nowadays when the power of CPUs and GPUs grew enormously over the years and computer graphics technology can deliver real time photorealistic rendering on the computers which many consumers can afford, the line between games and cinematography is fading. However, even Unity is trying to chase Unreal Engine in the race for filmmakers favor and publishing a package for TV and Film, it still is only a gaming engine so far. One of the features that Unity is missing is what UE has already built in as nDisplay - multiple display cluster management(Unreal Engine, 2018).
There are other components which are being added to the newest cinema package for Unity which were there for a long time in the UE.
The funny difference between two competing gaming engines in terms of cinematography is the fact that the general name(and class) for any object in the scene in UE is Actor and in Unity is GameObject.
Unity has two modes it operates in: Edit and Play. Users edit the project in the Edit mode and can make all the changes they want in the scene. If they want to try the project in action they can enter the Play mode simply by hitting the play button - project then runs without being built. When the user exits the Play mode all the changes made during Play mode are gone.
On the other hand when Unity transits from Edit to Play Mode unity performs two actions:

- Reloads Domain - reset your scripting state, so the project starts with the fresh one.
- Reloads the scene - destroys all the GameObjects.

Not to mention that it destroys everything but it also takes a very long time for Unity to perform these two operations.
Luckily enough Unity introduced a new option in the project setting under Player menu section

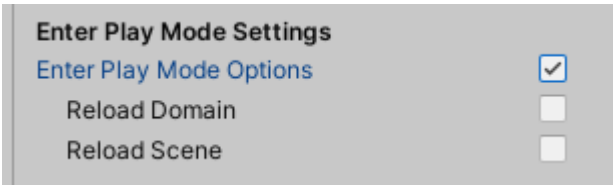


Figure 26 Disabling the reload in the project settings.(Unity Technologies, 2020b)

The effect of this settings could be seen on the diagram(Figure 26) below.

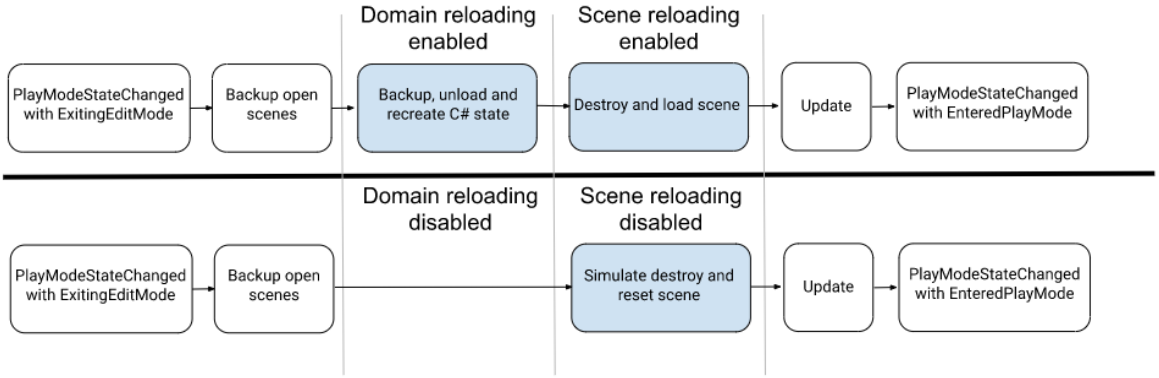


Figure 27 What happens when reloading is disabled. (Unity Technologies, 2020b)

These modifications to the project helped to achieve more control over the data and references. In addition all the classes that were intended to run in the editor mode were equipped with [ExecuteAlways] attribute.

Editor Window

When I was starting to implement the window for my console I had a choice of two systems that Unity has for the editor window programming. They have legacy OnGui() which is just a rendering member function of the EditorWindow class. Recently they introduced newer UIElements as a three layer compound system. It also has a class that inherits from the EditorWindow class plus UXML file for markup and USS file for style. Unity tried to copy common web apps architecture - backend, frontend and style. I found that UI elements would take longer to learn and master and went the OnGui() pass.
I've made the console window as the central class of my system, it displays all the systems controls and settings and aggregates communication and data objects. It also has some of the control functionality but big portions of it delegated to other classes.
It has a dynamic UI and hides the irrelevant elements from the user. It also hides the controls if the user chooses to pass control to the externa console.
Unity allows developers to add the menu items to the editor's menu so the attribute MenuItem over the Init() function.

ArtNetConsole.cs

```
[MenuItem("Window/Art-net/Console")]
static void Init()
{
    //...
```

```
}
```

So the above code results in the new item added to the menu in order to open the console window.

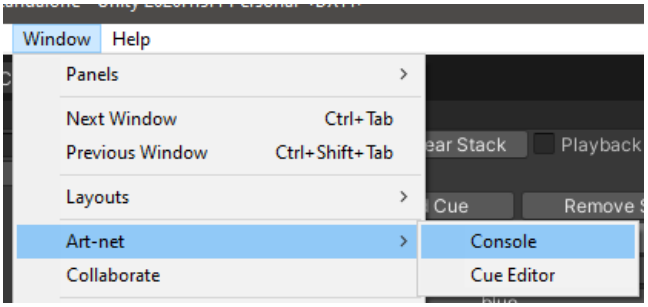


Figure 28 New menu item in Unity editor.

The script inside the `Init()` function makes sure that there is only one window of this type in the editor.

ArtNetConsole.cs

```
ArtNetConsole window =
(ArtNetConsole)EditorWindow.GetWindow(typeof(ArtNetConsole));
window.titleContent.text = "ArtNet Console";
window.Show();
```

`OnGui()` function which is responsible for rendering the window delegates to the elements render functions.

ArtNetConsole.cs

```
void OnGUI()
{
   DrawLayouts();
   DrawHeader();
   DrawSidePanel();
   DrawBody();
}
```

In the above code the `DrawLayouts()` is the one which holds the proportions of the window zones which are rendered by `DrawHeader()`, `DrawSidePanel()`, `DrawBody()`.

ArtNetConsole.cs

```
void DrawLayouts()
 {
      header.x = 0;
     header.y = 0;
     header.width = Screen.width;
     header.height = 65;

      side.x = 0;
      side.y = header.height;
      side.width = 400;
      side.height = Screen.height - header.height;

      body.x = side.width;
      body.y = header.height;
      body.width = Screen.width - side.width;
      body.height = Screen.height - header.height - 20;

      subHeader.x = 0;
      subHeader.y = 0;
      subHeader.width = Screen.width;
      subHeader.height = 20;
 }
```

You can see that in this way we can control the layout by relative and absolute values. Basically this function configures the x, y, width and height of 4 private rectangle objects.

ArtNetConsole.cs

```
Rect header;
Rect side;
Rect body;
Rect subHeader;
```

Each of these rectangles becomes an argument in the GUILayout.BeginArea(Rect area) function.

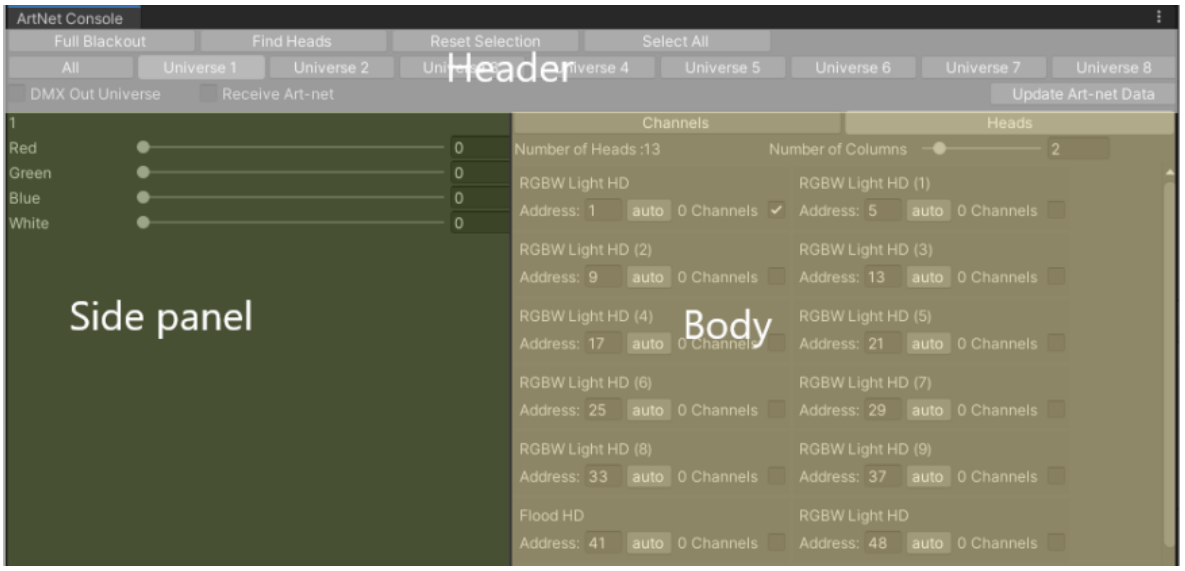


Figure 29. Window divided into layout areas.

That was my implementation of the layout as you can see that was not very complicated compared to implementing UXML layer. However it is not as convenient with the styles and you need to change the style of every component individually.
Each area component contains a number of nested elements - buttons, toggles, sliders and text or integer fields. Unity has two classes for the control elements GUILayout and EditorGUILayout.
The first one was made for in-game layout before Unity introduced Canvas API. The other one is actually made for the purpose I'm using - editor windows. They both work in the editor window and I'm using both, because some of the elements exist only in one and not in the other.
There is no event handlers attached to the GUI or EditorGUI elements so for example

ArtNetConsole.cs
activeUniverse = GUILayout.SelectionGrid(activeUniverse, universes, 9);

Results on the screen in the row of buttons to switch between universes.

Figure 30. GUILayout.SelectionGrid rendered.

This function returns the integer value of the index of pressed button every time OnGUI() repaints.
The same situation with buttons and toggles with the only difference that they return boolean values.

ArtNetConsole.cs

```
if (GUILayout.Button("Find Heads", GUILayout.Width(150)))
{
    heads = FindAllHeads();
}
```

In a similar manner we can chain conditions to hide or display toggles in accordance with the user's choice.

ArtNetConsole.cs

```
bool setAsDmx = EditorGUILayout.ToggleLeft("DMX Out Universe", dmxUniverse
== activeUniverse, GUILayout.Width(150));
if (setAsDmx)
{
        dmxUniverse = activeUniverse;
        sendDMX = EditorGUILayout.ToggleLeft("Serial DMX", sendDMX);
        if (sendDMX && dMX == null)
        {
           dMX = new DMX(dmxUniverse - 1);
        }
        if (!sendDMX && dMX != null)
        {
            dMX.Quit();
            dMX = null;
         }
 }
```

You can see how enabling one option opens up rendering of the other. In this case if we choose to set this universe as DMX out univers, then you can enable DMX serial communication with the DMX fixture connected to the DMX line of the USB device.
Similar but not exactly the same situation is with sliders: they display the value of the variable they are representing in the window. We should always check if the return value is equal to the input value and if it is not we need to assign the new value to the variable slider represent. Good example is DMX channel sliders.

ArtNetConsole.cs

```
int i = EditorGUILayout.IntSlider((int)artNetData.dmxDataMap[activeUniverse - 1][f],
0, 255, GUILayout.Width(300));
if (i != artNetData.dmxDataMap[activeUniverse - 1][f])
{
       artNetData.dmxDataMap[activeUniverse - 1][f] = (byte)i;
 }
```

Scriptable Object

Unity promotes the ScriptableObject class as being a perfect data container. It derives from the same base object as MonoBehaviour class but it does not have some of the event functions for example Update(). Scriptable object is aiming to save the memory as all the objects which are referencing it are accessing the same copy.

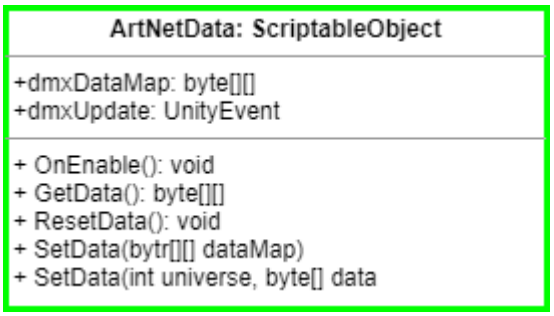


Figure 31. ScriptableObject ArtNetData. Class diagram.

On the Figure 31 you can see class diagram of the ArtNetData class which derives from ScriptableObject and is the data container for my DMX universes.

As I already have mentioned in the Design chapter that I decided to use a 2 dimensional array instead of a dictionary for reasons of the built in serialization mechanism of Unity. I was wrong and it didn't work. I could not find the answer why. Perhaps because of the second dimension. My attempts to save the object to disk as an asset with the data and then retrieve it again and restore the data did not succeed.

Every time I opened the application again the ArtNetData instance was null and I had to re-initialize it again. Adding the System.Serializable attribute did not help either.

ArtNetData.cs

```
[CreateAssetMenu(menuName = "ArtNet/DMX data map")]
[System.Serializable]
public class ArtNetData : ScriptableObject
```

However, in this case the fact that the data could not be saved to disk was not critical. For me it was good enough that data was in order after I canceled the domain reload, so I could retain the DMX values when the Play button was pressed. For stability reasons I have included a check of ArtNetData actually exists in most critical places.

As you can also see on the Figure 31 there is one more public member to this class and it's of type UnityEvent. Unity has a built in event system and they promote it often together with ScriptableObject for decoupling the components in the scene. Here I must explain my intuition about the use of the ScriptableObject and UnityEvent in my system.

The idea was to use it as a wire used in the real DMX; all the lamps were "connected" to it by aggregating it.

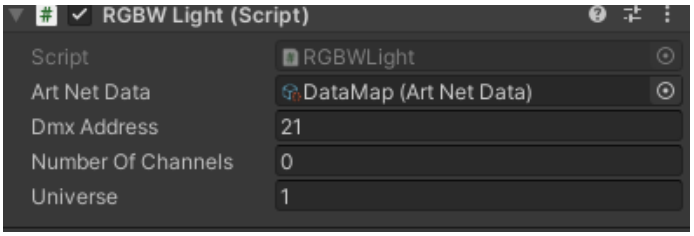


Figure 32. ScriptableObject aggregated by the light.

The event called dmxEvent would be raised every time there is something to update; it helps to trigger functionality in the Edit mode when Update() function is not called.

Art-net

To implement art-net functionality I have used some of the code from(sugi-cho, 2018). As was already mentioned, the ArtNetSocket object is aggregated in the console. After its Open() function is called it creates the socket and starts listening on the specified local IP.

ArtNetConsole.cs

```
void OpenArtNet()
{
      if (artnet != null)
         artnet.Close();
      artnet = new ArtNetSocket();
      artnet.Open(IPAddress.Any, null);
      ArtnetReceiver(CallUpdate);
}
```

The default port for art-net is 6454. After the socket is open, artnet is listening for incoming packets and if it receives valid DMX packets it raises the calls to the delegate function.

ArtNetConsole.cs

```
void ArtnetReceiver(Action callback)
{
    artnet.NewPacket += (object sender,
ArtNet.Sockets.NewPacketEventArgs<ArtNet.Packets.ArtNetPacket> e) =>
      {
         if (e.Packet.OpCode == ArtNet.Enums.ArtNetOpCodes.Dmx)
         {
               var packet = e.Packet as ArtNetDmxPacket;
               var universe = packet.Universe;
               if (receiveArtNet[universe])
```

```
            {
                artNetData.SetData(universe, packet.DmxData);
            }
        }
         callback();
    };
}
```

This code is using the principles of functional programming when the delegate(sometimes called **callback**) function is used as an argument of the caller function. You can also see another example of functional programming in use of the co-called **lambda** function with the event as an argument.

DMX USB Pro

As mentioned earlier I've used initial code from this repository(davivid, 2016). However I have upgraded it for my needs. For example I have changed the constructor to be aware of the universe DMX-out assigned to.

DMX.cs

```
public DMX(int universe)
{
        serialPorts = GetPortNames();
        this.universe = universe;
        OpenSerialPort();
        FindDataMap();
        initTXBuffer();
        dmxThread = new Thread(ThreadedIO);
        dmxThread.Start();
        updateDMX = true;
}
```

That helped me get DMX out data directly from my ScriptableObject and by this let me implement art-net in and DMX out to work simultaneously on the given universe.

DMX.cs

```
private void ThreadedIO()
{
    while (true)
    {
        if (updateDMX)
        {
            Buffer.BlockCopy(artNetData.dmxDataMap[universe], 0, TxBuffer,
DMX_INDEX_OFFSET, N_DMX_CHANNELS);
```

```
                if (serialPort != null && serialPort.IsOpen)
                {
                    serialPort.Write(TxBuffer, 0, TX_BUFFER_LENGTH);
                };
                Thread.Sleep(200);
            {
    }
}
```

Lights and Discoverability.

Now, let's talk about light - both virtual and real DMX fixtures. Figure 33 demonstrates how all the DMX controlled lights derive from the base class DMXfixture.

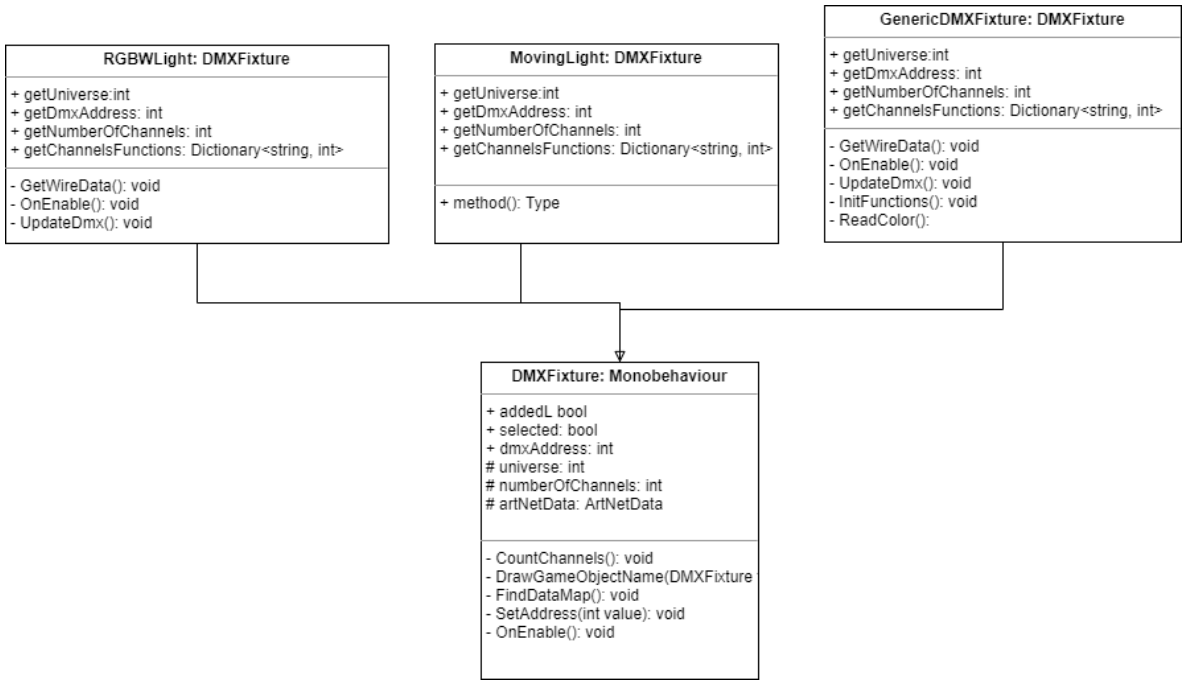


Figure 33 Inheritance of all the lights from DMXFixture class.

I have developed three types of lights. Two of them RGBW light and Moving light are virtual lights and have Unity light component attached to the transform. RGBW is set up as point light and receives 4 DMX channels for changing RGBW values. The moving light is a 6-channel fixture and on top of RGBW can move within Pan and Tilt planes.
The last DMXfixture child is a non-virtual light; it is a placeholder for the real DMX fixture connected to the line. Users can assign a DMX address to this one which would be the DMX address of the external fixture. The universe number should be of the DMX out universe. Users can edit the list of the attribute names to be displayed in the console.

All of the DMX devices have ArtNetData ScriptableObject aggregated and therefore have direct access to the data.They just use the values of the channels that this fixture's address is set to.
The discoverability of the fixtures is done by Unity's FindObjectOfType() function which finds all the objects derived from DMXFixture.

ArtNetConsole.cs

```
private Dictionary<int, DMXFixture[]> FindAllHeads()
{
        DMXFixture[] heads = GameObject.FindObjectsOfType<DMXFixture>();
        for (int i = 0; i < heads.Length; i++)
        {
           heads[i].FindDataMap();
        }
        Dictionary<int, DMXFixture[]> sortedHeads = new Dictionary<int,
DMXFixture[]>();
        patchSize = new int[numberOfUniverses + 1];
        for (int i = 0; i < heads.Length; i++)
        {
           patchSize[heads[i].getUniverse]++;
           patchSize[0]++;
        }
        sortedHeads[0] = heads;
        for (int i = 1; i <= numberOfUniverses; i++)
        {
           var tempHeads = heads.Where(h => h.getUniverse == i).ToList();
           var ordered = tempHeads.OrderBy(x => x.getDmxAddress);
           sortedHeads[i] = ordered.ToArray();
        }
        return sortedHeads;
}
```

The operation that is triggered by 'Find heads' button is resulting into all the heads that were found in the scene are displayed in the body area of the console(Figure 34).

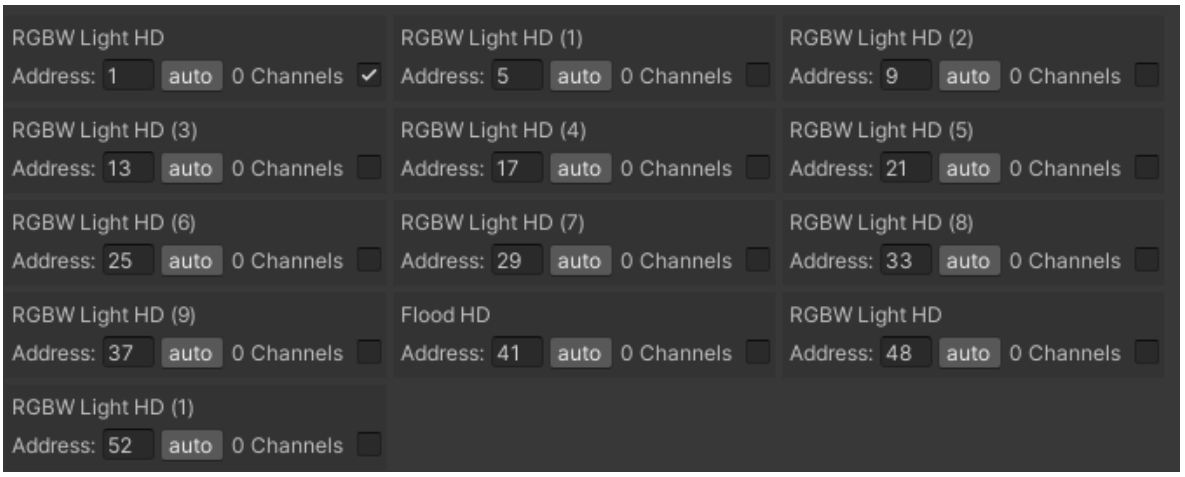


Figure 34 Fixtures selection pane.

Fixture then could be selected for control. The DMX address of the fixture could be changed in the console. Pressing the 'auto' button will automatically assign the first available address in the universe.
Before I move on to the fixture control I must say that all the fixtures are packed as prefabs and placing them in the scene is just dragging them from the project window to the hierarchy. The script unpacks the prefabs right away. Otherwise after entering and then exiting Play mode all the updated values like address and universe are reseted to the values of the prefab.

Group Controller

Group controller is a little construct with big responsibilities. On the Figure 35 you can see the class diagram of this very important component of my system.

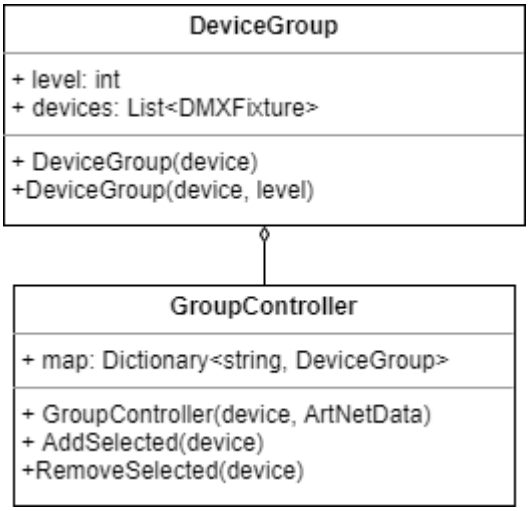


Figure 35. Group controller class view.

Its functionality helps users to manipulate not with the numbered channels but with the attributes names. For example if the user selects the fixtur in the fixtures pane the side panel of the console will display the list of sliders - one for each attribute the fixture has. So if for example we select 4-channel RGBW fixture we will get sliders as on the Figure 36.

Figure 36 Side sliders for RGBW fixture.

If we move the Red slider the light will start increasing the value of the red channel and light will shine red. If you select one more light - the newly selected light will also turn red. If you move Green slider color will change on both of them. Lets in addition to this two select 6-channels Moving Light and the side panel will display more sliders as on the Figure 37

Figure 37. Moving light sliders.

If you move the color sliders - it will change the color on all: RGBW and moving lights. If you move the pan or tilt slider, the beam of the moving light is going to move. In the other words, sliders will affect the attribute of the light if it has it.
This is the most essential function of any lighting console. It translates the numbers into attributes names for the user to control the light. Please note that this feature usually is not available on the budget consoles which do not support the soft patch of the multichannel fixtures.

Cue Editor

Now we can control the color, positions and whatever other attributes the fixtures can have. We can adjust the look so we are happy with it and want to save it to be recalled later. This preset as was already mentioned earlier in lighting vocabulary is called **cue**. To achieve this functionality I have developed another editor window called Cue Editor. Figure 38 shows both: class structure of the component and the way it looks in Unity editor.
Many features of this window are similar to the ones of the console. It has its own menu entry and executes always: in Edit and Play modes. It is also connected to the ArtNetData object.

It can save cues - named snap shots of all the art-net data the system holds. The collection of cues is usually called the cue stack . In the large lighting consoles you can play the cues in a large number of different ways. Users can play them in sequence, set fade-in and fade-out time, cross fade or combine them(Chamsys, 2014). My humble application can only record them and play back one by one. All the cue stacks can be edited. The cues can be removed. Stack could be cleared and saved to disk. Right now only one stack per scene is supported and the user cannot edit the name of it - it gets the scene's name.

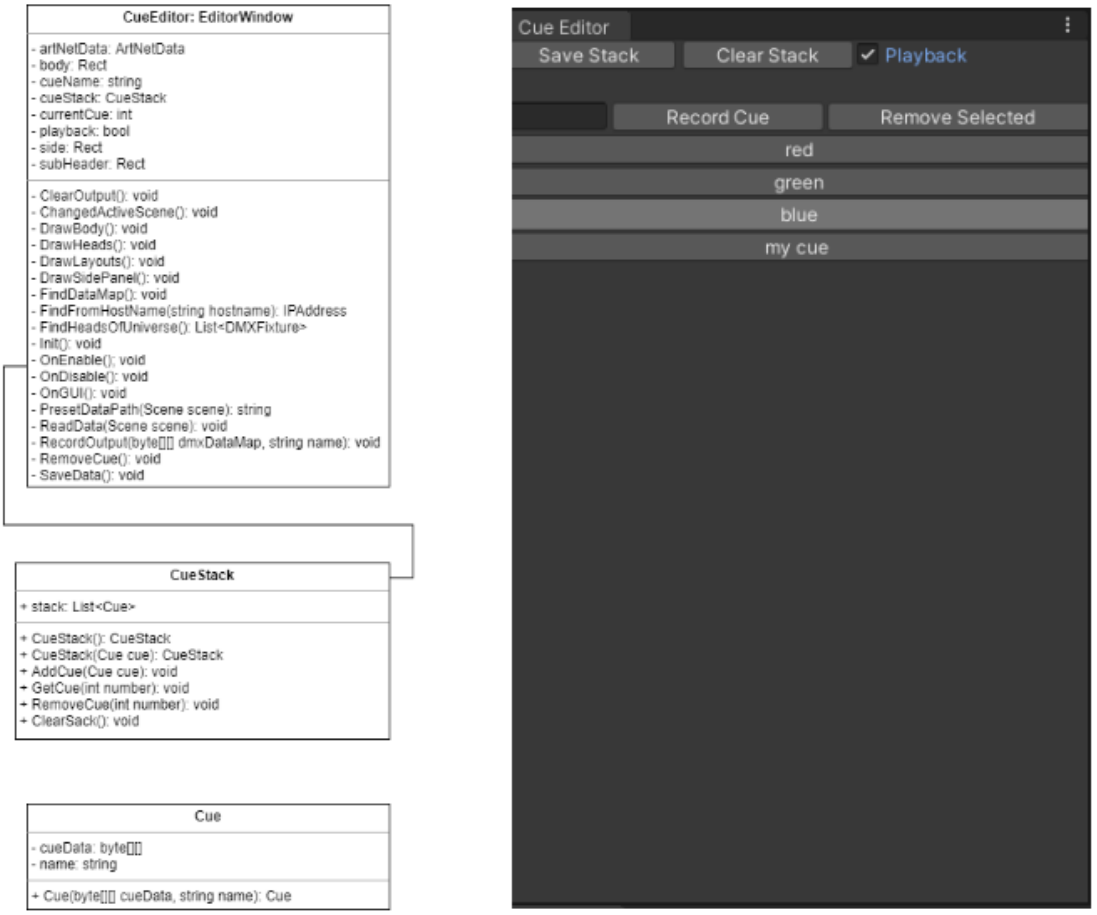


Figure 38. Class diagram and the screen capture.

As you can see on the Figure 38 the CueStack is the dictionary and saving it to disk was rather challenging and could not be achieved by Unity built in serialization. Instead I've used the("Json.NET - Newtonsoft," 2021) library to achieve the serialization of my stack. The serialization was pretty much straight forward and could be achieved by code.

CueEditor.cs
string json = JsonConvert.SerializeObject(cueStack, Formatting.Indented); string path = PresetPath(scene);

```
System.IO.File.WriteAllText(path, json);
```

Reading data back from disk and deserialization was not as smooth and I had to copy data byte by byte in the constructor of the Cue.

CueStack.cs

```
public Cue(byte[][] cueData, string name)
{
      this.name = name;
        this.cueData = new byte[8][];
        for (int i = 0; i < 8; i++)
        {
              this.cueData[i] = new byte[512];
              for(int j = 0; j < 512; j++)
              {
                 this.cueData[i][j] = (byte)cueData[i][j];
              }
         }
 }
```

Color Probe

The color probe is the tool to extract the light information from the point in the virtual scene. As I already mentioned in the design overview to achieve this I am using cubic illuminance concept(Cuttle, 2015). I have modified it to the requirements of my application.

Imagine that you have an actor standing in front of the LED wall that you want to illuminate with. The LED wall displays a virtual environment and you want to match the colors. I propose to use a reduced version of the cubic illuminance measuring method. Since we are only interested in the information of light falling on the facets of the cube that could be seen in the camera and we usually have two lighting points that would affect the lights - we are left with two facets. And due to the fact that angle is not always 90 degrees we can just use two plane surfaces to capture the light. Figure 39 shows the shape of the probe, angle could be adjusted.

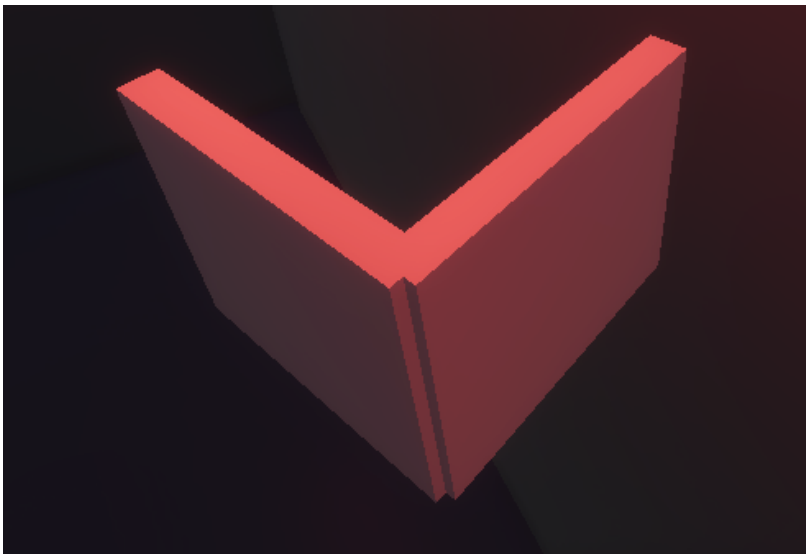


Figure 39. Color probe in the scene.

There is a camera attached in front of each facet. The camera is set for **orthographic projection**, it only renders the TransparentFX layer which the probe facets assign to. At the same time I exclude this layer from the culling mask of the Main camera.
The probe cameras render into render texture - a special type of texture that could be created in run time. The idea of the method was adapted from the youtube video(Bospear Programming, 2018).
The render texture is then referenced by a placeholder object which I have made for the real DMX lights. As you can see on the Figure 40 shows how render texture is used by the fixture.

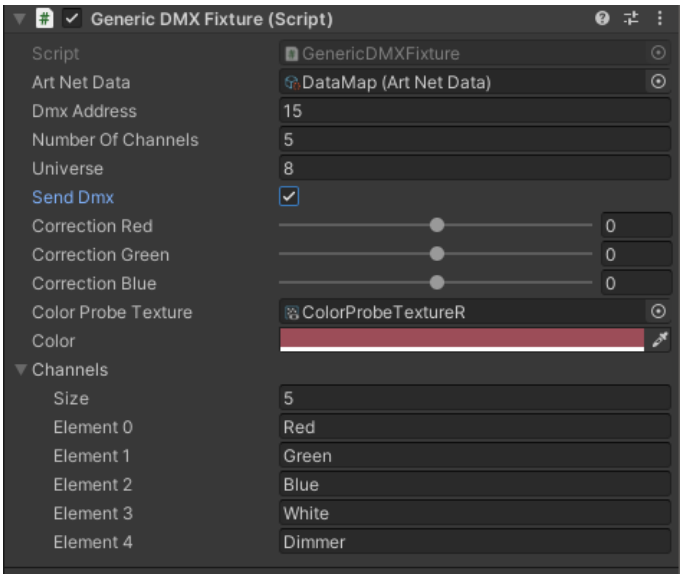


Figure 40. Inspector of the DMX fixture placeholder.

Then we copy the render texture to the 2Dtexture which we can retrieve RGB values from.

GenericDMXFixture.cs

```
RenderTexture tempTexture = RenderTexture.GetTemporary(colorProbeTexture.width,
colorProbeTexture.height, 24);
Graphics.Blit(colorProbeTexture, tempTexture);
RenderTexture previous = RenderTexture.active;
RenderTexture.active = tempTexture;

Texture2D tempTexture2D = new Texture2D(colorProbeTexture.width,
colorProbeTexture.height);
tempTexture2D.ReadPixels(new Rect(0,0, tempTexture.width, tempTexture.height),
0,0);
tempTexture2D.Apply();

RenderTexture.active = previous;
RenderTexture.ReleaseTemporary(tempTexture);
Color32[] colors = tempTexture2D.GetPixels32();
color = colors[0];
```

This color information is then set to appropriate channels of the ArtNetData and after is transmitted to the real DMX fixture.
There is a possibility to correct each color channel's information before it is sent to the DMX device.

GenericDMXFixture.cs

```
byte correctColor(byte component, int bias)
{
    return (byte)Mathf.Clamp(component+bias, 0f, 255f);
}
```

The sliders on the Figure 40 can help user to add or subtract the DMX value.
I have added an automatic positioning feature to the probe. If you place the probe in the scene it will automatically find the Main Camera and will place itself in front of it. Offset distance to the camera could be adjusted.

Testing

This chapter will explain how the testing procedures were coordinated and executed. I will start with technical testing of my system and after moving to the observations, survey and the interview which would help me to answer the question of my research. Not everything went down as it was planned. The detailed information - below.

Testing Software and Equipment.

Figure 41. Chamsys on PC and ArtNetominator.

In very early stages of the development I have added the serial DMX and art-net functionality to the project. To test art-net connectivity I have used Chamsys On PC("ChamSys MagicQ Downloads," 2020) and ArtNetominator(The ArtNetominator, 2019). The first is the computer based simulator of Chamsys's large hardware consoles. The software comes almost without any limitations compared to the commercial products. The limitations are the number of output universes and not being able to use USB DMX devices.

The second tool mentioned is the art-net data monitor which listens on all possible network interfaces on the port 6454.

To test DMX serial communication I have used USB DMX Pro MkII (Enntec, 2021) . It is a very advanced device, with much richer functionality than I needed for my experiments.

On the fixture side I used:

- NanLite MixPanel 60 Bicolor Hard and Soft CCT and RGBWW Light Panel("NanLite MixPanel 60 RGBW LED Panel," 2019)
- Eurolite LED THA-60PC Theater-Spot(Eurolite, 2017)

Figure 42 Mixpanel, Eurolite, USB DMX pro("NanLite MixPanel 60 RGBW LED Panel," 2019)(Eurolite, 2017)(Enntec, 2021)

Testing Set Up

All the testing sessions were situated at the SMILE lab ("Samsung Media Innovation Lab (SMILE)," 2017) at Aalborg University campus in Copenhagen, Denmark. In the lab there was a large LED wall consisting of 9(3 x 3) pieces of 55" Samsung LED screens. Figure 43 demonstrates testing set-up layout of the equipment. As you can see that it is convenient enough even for one person to operate the test. The actor who performed in all the tests was SMILE lab's mascot Cornelius.

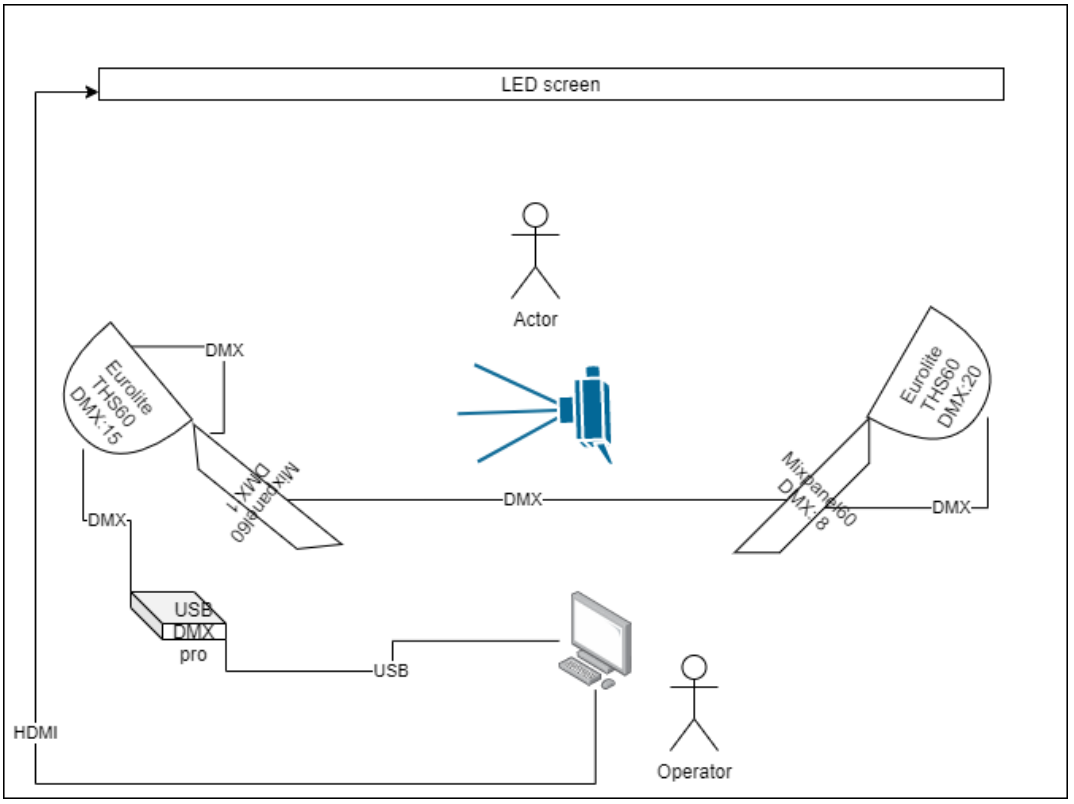


Figure 43 Test Set-up in the SMILE lab at AAU.

Testing Environments.

To run the testing of my system number of the testing virtual environments was constructed. Those environments did not have any specific quality or resolution requirements. I will show two environment examples here and will tell you what they were aimed to test

The Stage

This scene is like a playground for my system Figure 44. It is a large box made out of 5 planes(one side is open). The walls are made out of HDPR default material.

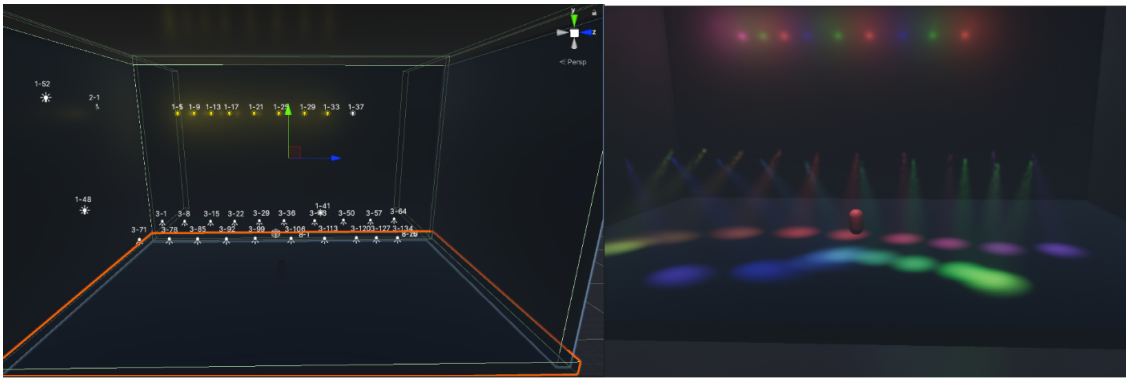


Figure 44 The stage scene view(left) and camera view(right).

The stage contains a large number of the RGBW and Moving lights. Main purpose to test art-net control of the virtual lights with shows made on the external lighting console. The same stage was used to test the color probe in different sceneries. One of them was flying through the dancing light beams.

The Corridor

This environment is built upon the assets which were taken from Unity's demo game project(Unity-Technologies, 2021). I've joined together some parts of the spaceship to make a little space station. This corridor is one of the parts of it. I have placed 30 RGBW lights assigned DMX addresses to all of them and made a number of presets. On Figure 45 you can see that all the lamps are set to different colors.

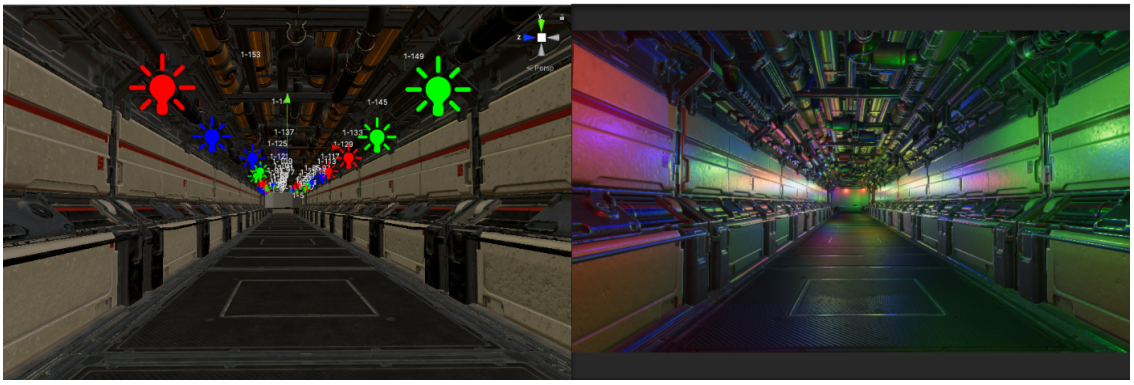


Figure 45. The corridor has 30 RGBW lights. Scene view(left) and main camera view(right)

I have also tested my system in a variety of different environments including Unity's default HDRP project demo scene. By doing this I wanted to confirm that my system can be integrated into any project.

Color Probe

To test the performance of the color probe tool following test conditions were raised. The environments mentioned in the previous chapter were equipped with the probe. To achieve this the probe was added to the scene and two render textures - one for each facet of the probe were referenced by two MixPanel60 placeholders Figure 40.
Two move the camera through the environment a simple FlyCamera.cs script was attached to the main camera. That allowed the user to move the camera by standard for gaming WASD keys.

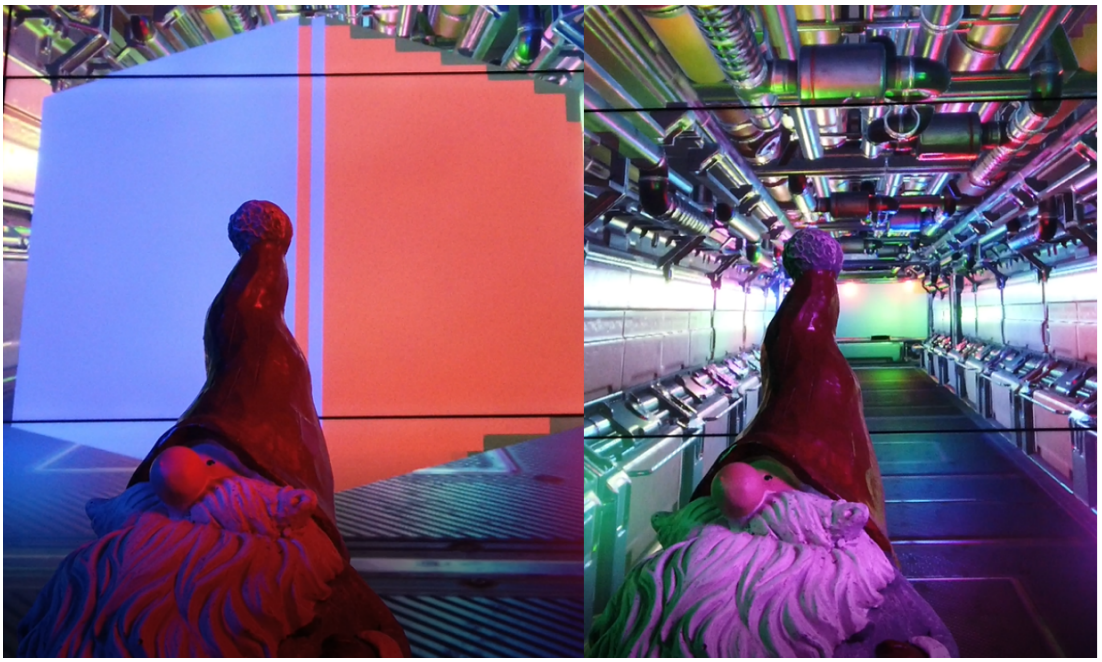


Figure 46. Color probe visible(left) and invisible (right)

The topology of the set was similar to the the one on Figure 43, but only MixPanels60s were used. The test was made in Unity's Play mode. For the first run, the camera's culling mask was set to render all layers which made the probe visible. The camera then was moved by the operator across the environment. The point of the test was to compare the color of the probe's facets with the light output of the DMX controlled fixture.
The second run was made in the same conditions but with the culling mask being set to exclude the TransparentFX layer. This made the color probe invisible.
The testing procedure was repeated multiple times in the corridor and on the stage and was filmed for future analysis.

Observations.

Demo session.

Near the end of sprint 2 the group of undergraduate students from AAU contacted me. They were working on a real time virtual production project. The topic of their research was interactive storytelling. As part of the experiment they had to shoot a remake of one of the scenes from the sci-fi film Ex Machina(Garland, 2015). The scene was situated in the room with mainly white walls and white furniture. The room is lit by white lighting. There are two actors having a conversation in the room. In the half way through the episode the lighting changes to red. The episode was planned to be filmed in the SMILE lab where actors would perform in front of the LED screen. The on set lighting consisted of the DMX fixtures mentioned above. Some of the group members had some photography and lighting experience and also good knowledge of CGI in general and Unity in particular. They have also developed a camera tracking system based on the HTC Vive products(Vive, 2021). However they did not have any knowledge of DMX control and asked me to help them with the task.
We agreed to have a demo session where I could demonstrate to them my system and we can talk about different scenarios and use cases. The session was recorded. Later playback and analysis helped me to improve the system and make it more convenient to use for novice DMX users. Figure 47 demonstrates the session after which the group took a timeout to make a decision on how to organize their pipeline.

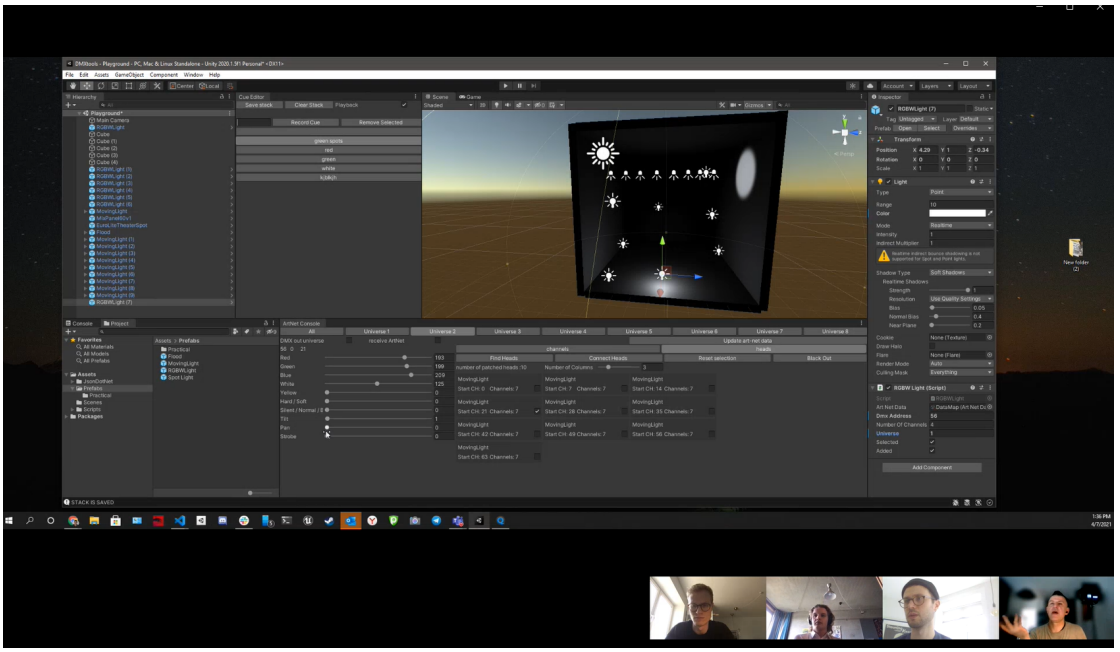


Figure 47 Live demo session.

Deployment assistance.

After a short period of time the group returned to me and asked to assist them to integrate my tools to their project. Appointment was made for the live session in the lab. After a discussion which resulted in the number of possible solutions, the group finally accepted the idea of using my system.

Notes taken during the session helped further improvements and gain better understanding of why DMX is sometimes hard to get a grasp of for a novice.

The group achieved the successful filming of their episode to make it work for them they had to do some pragmatic modifications. The results could be seen on the Figure 48.

Figure 48 Scene from the episode filmed featuring the DMX system.

Survey.

As was already mentioned above, a survey link was attached to the button on the window of Cue Editor. At the moment of writing I have received 7 qualified answers. That means that participants have already had some knowledge of either Unity or DMX control.

Interview.

To capture the sense of importance of the topic and to approve my design assumptions I have appointed an interview with Sebastian Bülow. Sebastian is one of Denmark's most innovative visual technical artists. He made a large number of live visual performances alone with different music bands. He is an expert in the history of cinematography and teaches the AV production course at Aalborg University. Not to mention that he designs all the pipelines for his shows himself.
Since we both have knowledge and experience in the AV and lighting and both got very excited about the matter. The interview's structure happened to be slightly different from how it was planned, but essential information and strong interest in the technology was captured. The interview was audio recorded.

Findings

The observations of the process of deploying and using the system by the peer group has shown scepticism and confusion towards DMX in the beginning and total acceptance at

the end. The feedback I've received from the group made me modify the system. The multiple iterations of explaining the technology helped me to make it more intuitive.
The expert interview did not answer the question of research directly but inspired the researcher and the respondent to further investigations on the topic. Among interesting ideas was video mapping and use of DMX for virtual scenography or scene reload. Another interesting idea was two extend the beams of the real light on to virtual space to create even more depth.
The survey questions answered by experts indicated strong interest in the technology, but kept the ones which were not familiar with the DMX in advance confused.
The software was mainly evaluated as stable and compatible, but not intuitive and easy to use by users with no DMX experience .
There is evidence of the correlation of proficiency in either DMX or Unity and opinion on ease to use the opposite.
Survey also registered a high level of support for all proposed use cases. Two features which were down prioritised by the developer art-net out and DMX in were voted as important. One user stated that art-net out is must have for him because all his setup is art-net.
Color probe was pointed out by one user as a progress in the right direction.

Discussion

DMX for UE4

Half way through my project I have found the information that starting from version 4.25 DMX package became available for Unreal Engine users("DMX Overview," 2021). According to the documentation and online video demo (Unreal Engine, 2020) the main use case that they were aiming to support is virtual life performances like the one Epic had in April 2020. The live performance of hip-hop artist Travis Scott(Ali-A, 2020) was "attended" by 12 million Fornite players.
DMX packages could be added to any project and consist of

- Protocol itself including Art-net and sACN, but no USB DMX support
- DMX fixtures package
- DMX engine
- Pixel mapping engine

The implementation is feature-rich and therefore seems to be difficult to grasp for the user that is new to DMX. Before I came across this solution I had already decided to do my implementation in Unity which I am much more familiar with. I have not studied UE's implementation in detail to minimize the bias in hope to find an even better solution. However by watching the tutorial videos I have noticed one issue that I have resolved in my solution. You can only see output of the lamps during Play mode and it makes it very difficult to position and focus the lights. You have to run Play mode every time you want to check how it works and all the changes made during Play mode are not saved so

sometimes the user has to switch multiple times from Edit to Play mode just to find the right position, color or intensity. I think it is not a very big problem if you want to set up a virtual live show, but for cinematography, very precise lighting is critical. It makes a big difference to be able to view the light in Edit mode.

Personal evaluation.

Here, I would like to make a personal statement about my system and DMX on Unity in general. I think that due to the low number of respondents with skills and experience in both lighting and real time engines, I, who spend years in both of the technologys separately and months working to join them together, have the right to be heard.
I think that overall system reliability is high. It works in different scenarios and is quite robust. Can run in Edit and Play mode. For most of the scenarios I would use the system and it would save me time to set up, but also simplify the control. Even with cue editors being so primitive it is possible to change the look of the stage instantly.
I think Unity should make their own DMX engine to be the part of the platform and make every light or even object to be able to assign different properties as DMX channels sourcing the data from the common DMX data object similar to how it is done in my implementation. The part of the DMX which usually causes confusion among novices is **addressing**. Perhaps there are better ways to hide this complexity behind some easy UI or API.

Conclusion

DMX is a good candidate to be used as a method of the control of not only lights but other objects within both; virtual and physical space. Asychronical nature of the DMX makes possible instant state changes. High availability of the DMX equipment for both; control and output, and high number of trained lighting personnel would convey the acceptance of DMX within real time engines. DMX has shown itself as a reliable interaction bridge between legacy lighting equipment and modern real time game engines. Future work is necessary in research as well as in the development of even more convenient tools.

Future Work

Research

To conduct further research on possible acceptance of DMX in real time productions the Technology Acceptance Model (TAM) could be used. TAM is an information systems theory that models how users come to accept and use a technology(Davis, 1989). I would suggest augmenting the general TAM with some external factors leading to the two main constructs which would indicate the intention to use DMX(Chuttur, 2009). Figure 49 shows how the results and quality of the application's use would be correlated to the perceived usefulness and years of previous experience would correlate to ease of use.

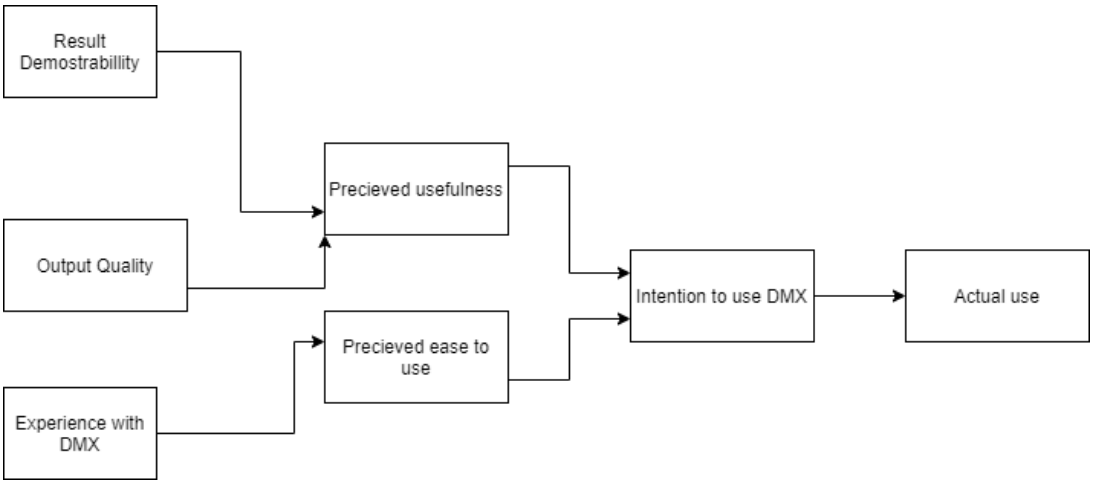


Figure 49 TAM. proposed model.

If applied to a larger audience this model could be used in comparative research of DMX against the non-DMX solution in completing the same tasks. In this study result and the output of two solutions could be compared. Years of experience with DMX could be the determining factor for perceived ease of use.

Development

There are many improvements that could be made to the system. Some of them were known beforehand and were not implemented due to lower priority compared to the others. Some were discovered during usability testing.

- Art-net output.
- Colors to DMX with the color picker
- Extended use of other color and tone notations than RGB(CCT, HSI).
- Tooltips could be added to all the controls.

Another interesting idea would be to create a light version of the DMX framework in the form of a prefab to control 1-4 DMX fixtures with an easy set up scenario. Then this light version could be compared with the current full featured system in completion of the same tasks.

Acknowledgments

I would like to express my special thanks to this project's supervisor Henrik Schønau Fog, great A/V expert Sebastian Bülow and everyone who gave me valuable feedback.
Above all I would like to thank my family for great help and support during this project and entire Master studies.

References

Ali-A. (2020). Fortnite Travis Scott *LIVE* FULL CONCERT Event! (HD) [YouTube Video]. Retrieved from https://www.youtube.com/watch?v=r5Q4FSdjZC4

Alliance, A. (2015, June 29). What is Agile Software Development? Retrieved May 17, 2021, from Agile Alliance | website: https://www.agilealliance.org/agile101/

basstronix. (2020, June 11). basstronix/UnityArtNet. Retrieved January 22, 2021, from GitHub website: https://github.com/basstronix/UnityArtNet

Bedard, M. (2020, September 20). Rotoscoping: The Perfect Marriage of Live-Action with Animation. Retrieved May 6, 2021, from StudioBinder website: https://www.studiobinder.com/blog/what-is-rotoscope-animation-definition/

Bospear Programming. (2018). [Unity] Get Light Intensity On Player [YouTube Video]. Retrieved from https://www.youtube.com/watch?v=NYysvuyivc4

Brown, B. (2018). *Motion picture and video lighting*. Routledge.

Chamsys. (2014). Chapter 15. Cue Stacks. Retrieved May 14, 2021, from Chamsys.co.uk website: https://secure.chamsys.co.uk/help/documentation/magicq/cue-stacks.html

ChamSys MagicQ Downloads. (2020). Retrieved May 14, 2021, from ChamSys website: https://chamsyslighting.com/pages/magicq-downloads

Chuttur, M. Y. (2009). Overview of the technology acceptance model: Origins, developments and future directions. *Working Papers on Information Systems*, *9*(37), 9–37.

CodePlex Archive. (2021). Retrieved May 3, 2021, from CodePlex Archive website: https://archive.codeplex.com/?p=acn

Cooper, T. (2012, October 25). Unity Serialization - Unity Technologies Blog. Retrieved March 1, 2021, from Unity Technologies Blog website: https://blogs.unity3d.com/2012/10/25/unity-serialization/?_ga=2.193225901.1400471804.1612794762-572226612.1606124811

Cross Reference. (2011). Cross Reference - Google Workspace Marketplace. Retrieved May 20, 2021, from Google.com website: https://workspace.google.com/u/0/marketplace/app/cross_reference/269114033347?hl=en&pann=docs_addon_widget

Cuttle, C. (2015). *Lighting design: a perception based approach*. Routledge.

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 319–340.

davivid. (2016, January 30). davivid/Unity-DMX. Retrieved May 3, 2021, from GitHub website: https://github.com/davivid/Unity-DMX

Diagrams.net. (2021). Diagram Software and Flowchart Maker. Retrieved May 17, 2021, from Diagrams.net website: https://www.diagrams.net/

Disneyplus. (2021). Mandalorian, Technology S:1, E: 4. Retrieved January 11, 2021, from Disneyplus.com website: https://www.disneyplus.com/en-gb/video/8112cabc-5ed9-4176-b0f8-b4baa0ab15f1

DMX Explained; DMX512 and RS-485 Protocol Detail for Lighting Applications. (2017, August 24). Retrieved January 18, 2021, from Element14.com website: https://www.element14.com/community/groups/open-source-hardware/blog/2017/08/24/dmx-explained-dmx512-and-rs-485-protocol-detail-for-lighting-applications

DMX Overview. (2021). Retrieved May 3, 2021, from Unrealengine.com website: https://docs.unrealengine.com/en-US/WorkingWithMedia/DMX/Overview/index.html

Döringer, S. (2021). “The problem centred expert interview”. Combining qualitative interviewing approaches for investigating implicit expert knowledge. *Null*, *24*(3), 265–278. https://doi.org/10.1080/13645579.2020.1766777

Duff, J., Antonutto, G., & Torres, S. (2015). On the calculation and measurement of mean room surface exitance. *Lighting Research & Technology*, *48*(3), 384–388. https://doi.org/10.1177/1477153515593579

Eevee — Blender Manual. (2021). Retrieved May 3, 2021, from Blender.org website: https://docs.blender.org/manual/en/latest/render/eevee/index.html

Enttec. (2021). DMX USB Pro interface Mk2, the new industry standard | ENTTEC. Retrieved May 14, 2021, from ENTTEC website: https://www.enttec.com/product/controls/dmx-usb-interfaces/dmx-usb-pro-interface/

Eurolite. (2017). LED THA-60PC Theater-Spot bk. Retrieved May 14, 2021, from Musikhaus Thomann website: https://www.thomann.de/gb/eurolite_led_tha_60pc_theater_spot_bk.htm

FAILS, I. (2020, April). How Previs Has Gone Real-Time. Retrieved May 4, 2021, from VFX Voice Magazine website: https://www.vfxvoice.com/how-previs-has-gone-real-time/

Foreman, R., & Motter, G. (2014). PART I: HUB SWITCHBOARD OPERATING MANUAL. Retrieved March 10, 2021, from Blogspot.com website: http://backstagefox1929.blogspot.com/2014/03/part-i-hub-switchboard-operating-manual.html

Gallardo, A. (2000). *3D lighting: history, concepts, and techniques*.

Garland, A. (2015, January 21). Ex Machina. Retrieved May 18, 2021, from IMDb website: https://www.imdb.com/title/tt0470752/?ref_=fn_al_tt_1

Gershun, A. (1939). The light field. *Journal of Mathematics and Physics*, *18*(14), 51–151.

Git. (2021). Retrieved May 17, 2021, from Git-scm.com website: https://git-scm.com/

GitHub. (2021). Build software better, together. Retrieved May 17, 2021, from GitHub website: https://github.com/

Google Docs. (2021). Retrieved May 20, 2021, from Google.com website: https://docs.google.com

Holloway, J. (2010). *Illustrated theatre production guide*. Taylor & Francis.

Holz, L. (2014). Jersey Indie. Retrieved March 17, 2021, from Jersey Indie website: https://www.jerseyindie.com/the-black-maria-film-festival/

igolinin. (2021, April 29). igolinin/DMXtools. Retrieved May 26, 2021, from GitHub website: https://github.com/igolinin/DMXtools

IMDB. (2016, April 7). The Jungle Book. Retrieved May 21, 2021, from IMDb website: https://www.imdb.com/title/tt3040964/

IMDB. (2017, October 4). Blade Runner 2049. Retrieved May 21, 2021, from IMDb website: https://www.imdb.com/title/tt1856101/?ref_=nv_sr_srsg_0

IMDB. (2019, July 12). The Lion King. Retrieved May 21, 2021, from IMDb website: https://www.imdb.com/title/tt6105098/?ref_=nv_sr_srsg_0

Jackman, J. (2020). *Lighting for digital video and television*. Routledge.

Json.NET - Newtonsoft. (2021). Retrieved May 14, 2021, from Newtonsoft.com website: https://www.newtonsoft.com/json

Karras, T., Laine, S., & Aila, T. (2019). A stylebased generator architecture for generative adversarial networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4401–4410.

Lammert Bies © 1997-2019. (2019). RS485, specifications and in depth tutorial - Lammert Bies. Retrieved March 11, 2021, from Lammert Bies website: https://www.lammertbies.nl/comm/info/rs-485

Lanier, L. (2018). *Aesthetic 3D Lighting: History, Theory, and Application*. Taylor & Francis.

McCandless, S. R. (1958). *A method of lighting the stage*. Theatre Arts Books.

MyBib. (2021). MyBib Citation Manager. Retrieved May 20, 2021, from MyBib website: https://www.mybib.com

NanLite MixPanel 60 RGBWW LED Panel. (2019). Retrieved May 14, 2021, from Kamerahuset.dk website: https://www.kamerahuset.dk/nanlite-mixpanel-60-rgbww-led-panel

Nishida, S. (2021, April 15). Netflix released, “DX at the shooting site” that is no longer on location ... Next-generation studio infiltration. Retrieved May 10, 2021, from Businessinsider.jp website: https://www.businessinsider.jp/post-233046

Paint3D. (2018). Get Paint 3D - Microsoft Store. Retrieved May 17, 2021, from Microsoft Store website: https://www.microsoft.com/en-us/p/paint-3d/9nblggh5fv99#activetab=pivot:overviewtab

Penzel, F. (1978). *Theatre lighting before electricity*. Wesleyan university press.

r/Maya - Will we get real time rendering like Eevee in Maya 2020? (2020). Retrieved May 3, 2021, from reddit website:

https://www.reddit.com/r/Maya/comments/d2rnhq/will_we_get_real_time_rendering_like_eevee_in/

Rasmussen, M. E., Persson, M. K., Raursø, N. E., & Petersen, T. A. (2021). Lean-Back Machina: Attention-Based Skippable Segments in Interactive Cinema.

RocketJump Film School. (2016). Lighting 101: Quality of Light [YouTube Video]. Retrieved from https://www.youtube.com/watch?v=Jw066PBZe60&list=PLw_JAmvzR_MBd6rUWTpVma4demi1pnS9S&index=5

Samsung Media Innovation Lab (SMILE). (2017, May 3). Retrieved May 14, 2021, from Cph.aau.dk website: https://www.en.cph.aau.dk/collaboration/students/labs-campus/labs/samsung-media-innovation-lab--smile-.cid314851

Stagecraft - Electrification | Britannica. (2021). In *Encyclopædia Britannica*. Retrieved from https://www.britannica.com/art/stagecraft/Electrification#ref1004215

sugi-cho. (2018, November 13). sugi-cho/ArtNet.Unity. Retrieved March 1, 2021, from GitHub website: https://github.com/sugi-cho/ArtNet.Unity

TED. (2020). How volumetric video brings a new dimension to filmmaking | Diego Prilusky [YouTube Video]. Retrieved from https://www.youtube.com/watch?v=iwUkbi4_wWo&t=569s

The ArtNetominator. (2019). The ArtNetominator - Free ArtNet DMX Monitoring and Troubleshooting Software. Retrieved February 10, 2021, from Lightjams.com website: https://www.lightjams.com/artnetominator/

Unity Technologies. (2020a). Unity - Manual: Configurable Enter Play Mode. Retrieved May 12, 2021, from Unity3d.com website: https://docs.unity3d.com/Manual/ConfigurableEnterPlayMode.html

Unity Technologies. (2020b). Unity - Scripting API: Retrieved May 12, 2021, from Unity3d.com website: https://docs.unity3d.com/ScriptReference/

Unity Technologies. (2020c). Unity - Unity. Retrieved May 17, 2021, from Unity website: https://unity.com/

Unity-Technologies. (2021a, March 2). Unity-Technologies/SpaceshipDemo. Retrieved May 15, 2021, from GitHub website: https://github.com/Unity-Technologies/SpaceshipDemo

Unity-Technologies. (2021b, March 2). Unity-Technologies/SpaceshipDemo. Retrieved May 15, 2021, from GitHub website: https://github.com/Unity-Technologies/SpaceshipDemo

Unreal Engine. (2018). Rendering to Multiple Displays with nDisplay. Retrieved May 12, 2021, from Unrealengine.com website: https://docs.unrealengine.com/en-US/WorkingWithMedia/nDisplay/index.html

Unreal Engine. (2019). Virtual Production Hub - Unreal Engine. Retrieved January 14, 2021, from Unreal Engine website: https://www.unrealengine.com/en-US/virtual-production?sessionInvalidated=true

Unreal Engine. (2020a). Control Lighting With DMX In 4.26 | Inside Unreal [YouTube Video]. Retrieved from https://www.youtube.com/watch?v=-oIePxpasXk

Unreal Engine. (2020b). Virtual Production: The Transition to Real-Time Filmmaking | The Pulse | Unreal Engine [YouTube Video]. Retrieved from https://www.youtube.com/watch?v=p9XUd4TQl2Y

USITT. (2018). *Entertainment Technology-USITT DMX512-A Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories*. Retrieved from website: https://tsp.esta.org/tsp/documents/docs/ANSI-ESTA_E1-11_2008R2018.pdf

Visual Effects Society. (2021, May 20). Intersection Between Camera, VFX & Lighting for Virtual Production. Retrieved May 22, 2021, from Vimeo website: https://vimeo.com/483297338?utm_campaign=5370367&utm_source=affiliate&utm_channel=affiliate&cjevent=b303ef85b22311eb814933300a180510&clickid=b303ef85b22311eb814933300a180510

Visual Studio Code. (2016, April 14). Retrieved May 17, 2021, from Visualstudio.com website: https://code.visualstudio.com/

Vive. (2021). VIVE European Union | Discover Virtual Reality Beyond Imagination. Retrieved May 18, 2021, from Vive.com website: https://www.vive.com/eu/

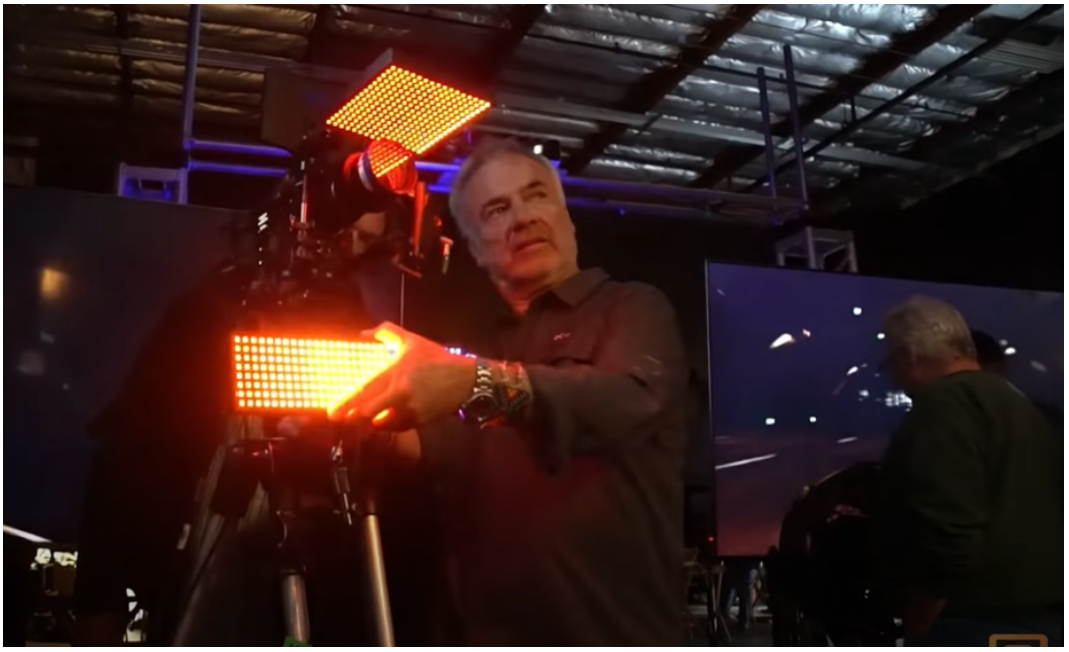
Whelan, M. (2016). Arc Lamps - How They Work & History. Retrieved March 9, 2021, from Edisontechcenter.org website: http://edisontechcenter.org/ArcLamps.html

Appendix.

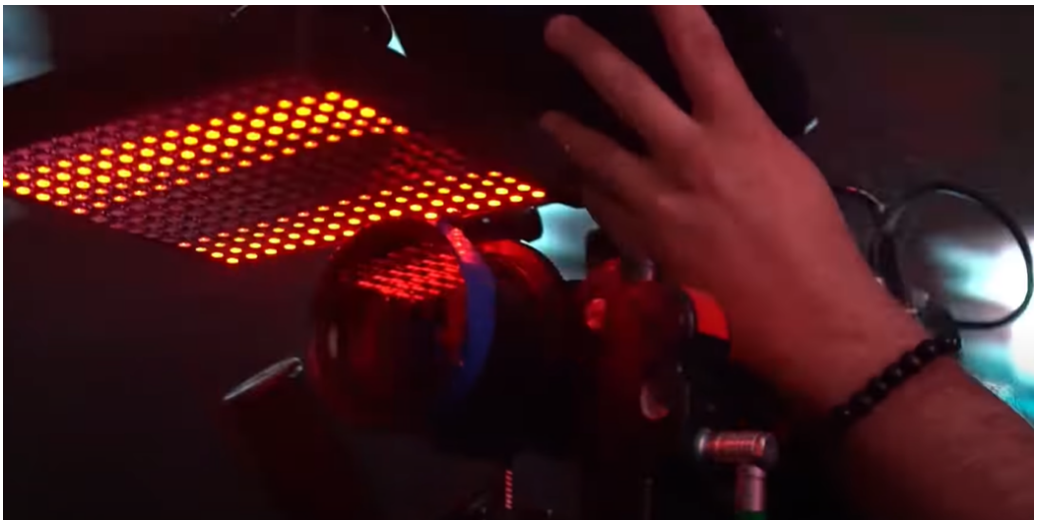
Random Figures.

Here I would like to present some interesting images from the virtual productions. of them don’t need any comments.

(Unreal Engine, 2020b)



(Unreal Engine, 2020b)



Note LED lights positioned right above the lens. Stargate studio. (Unreal Engine, 2020b)

Diego Prilusky’s volumetric video, hundreds of cameras that capture light and motion from every angle.

Unnamed Netflix production, Japan(Nishida, 2021)

Unnamed Netflix production, Japan(Nishida, 2021)

Unnamed Netflix production, Japan(Nishida, 2021)

Unnamed Netflix production, Japan(Nishida, 2021)

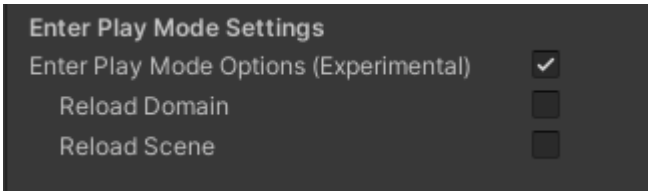
“A night view seen through a champagne glass. It's hard to make such a video by "synthesis", but it's easy with virtual production.”(Nishida, 2021)

Unnamed Netflix production, Japan(Nishida, 2021)

Manual

Installation

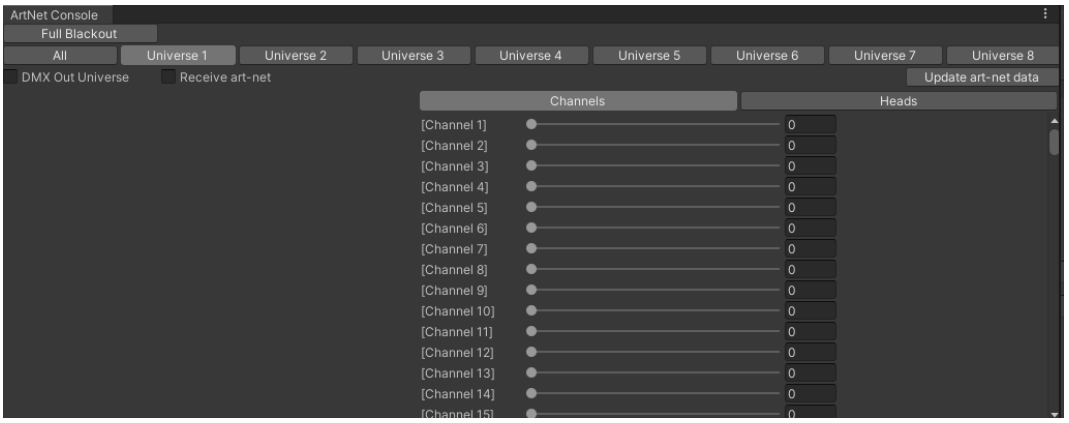
1. Download the package from https://github.com/igolinin/DMXtools/blob/main/DMXtools/DMXtools.unitypackage
2. Open Unity HDRP project (or create a new one).
3. Select **Menu/Assets/Import Package/Custom Package** and browse to DMXtools.unitypackage. Click “Import”.
4. Navigate to **Menu/Edit/Project Settings/Player** and set
    a. Api Compatibility Level: .Net 4.x,
5. Navigate to **Menu/Edit/Project Settings/Editor** and set
    a. Enter Play Mode Settings: true
    b. Reload Domain: false
    c. Reload Scene: false



Operations

!!! In case of any errors which might come after transitions between play and edit mode please, use the “Find Heads” button. It fixes the connections and rebuilds data objects if missing>

1. Navigate to Menu/Window/Art-net/Console this will open console window

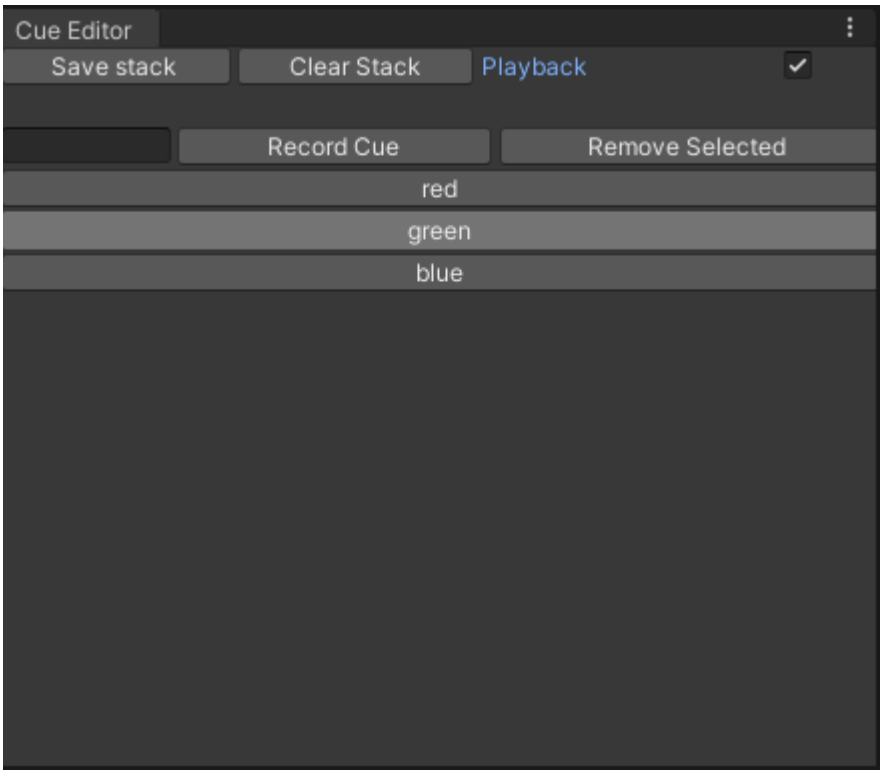


2. User can select one of 8 built-in art-net universes or choose all(heads only)
3. The selected universe can be marked as “DMX Out Universe” to send data of this universe to the real DMX fixtures connected via DMX usb dongle(only Enttec usb pro MKii tested, but should also work with open DMX dongle as well).

a. When the universe is enabled as “DMX out universe”, the option “Serial DMX” becomes visible.
b. USB DMX devices must be connected to the PC before “serial DMX” is enabled.
c. Only one physical DMX universe is supported.

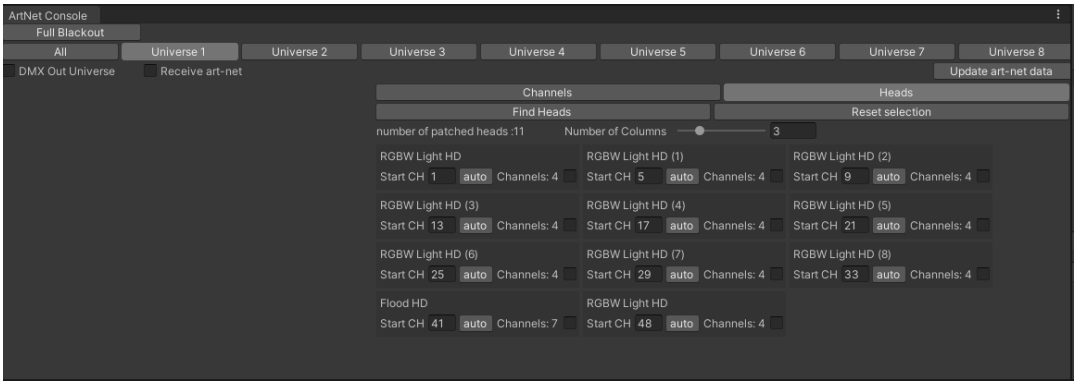
4. To get familiar with systems features please open demo scene “Playground”
5. Number of virtual DMX fixtures are already placed in the scene. By selecting the universe and choosing the heads view user can
    a. “Find heads”, which detects all the universe’s DMX fixtures that are present in the scene. The same button can be used in case of the error which might occur when exiting play mode.
    b. “Reset Selection” resets the selection of the heads.

6. When one or more heads are selected, the side panel displays sliders for the fixture’s common attributes.
7. Each universe including DMX out universe can be set to “Receive Art-net” in this case art-net data is received from the external art-net controller. This feature only works real time in the play mode, but pressing “Update Art-net Data” can update virtual fixtures output after the latest received art-net data while in edit mode.
8. Cue - the snapshot of all the DMX data values for all universes can be saved with a basic cue editor.
9. Navigate to Menu/Window/Art-net/Cue Editor.

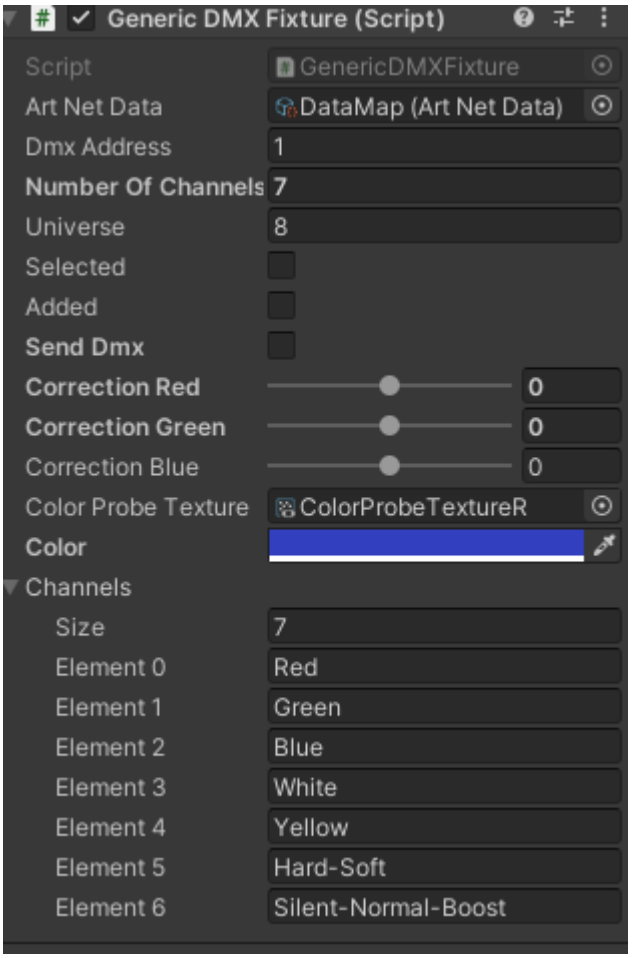


10. To save the cue simply enter the cue name and click the “Record Cue” button.

11. To remove the cue click “Remove Selected”
12. To save the entire cue stack click “Save Stack”. The stack is saved on a per scene basis.
13. To play back the cues enable the “Playback” option. It is recommended to reset selection and black out on the console before beginning the playback.
14. To add more fixtures to the scene drag and drop prefabs from Assets/Prefabs/Lights/HDRP.
    a. Set universe and address numbers in the inspector.
    b. Set address number in console.
    c. Click ‘auto’ to set the first available(not used) DMX address.
    d. Press “Find heads” if there are any errors.

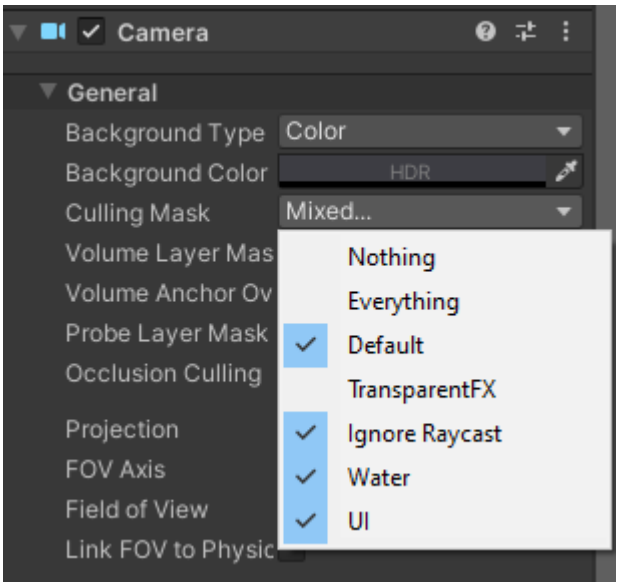


15. To control practical DMX fixtures via USB dongle, the placeholder object must be placed in the scene. You can find prefabs of these objects in Assets/Prefabs/Lights/Practical.
    a. Drag and drop GenericDMXFixture to the scene
    b. Chose number of channels in Channels/Size
    c. List all the channel attribute names under “Element1, ... Element N”
    d. Select Universe and DMX address.
    e. Select the color probe render texture if you want this fixture to follow the color of the probe.
    f. Send DMX must be enabled to control the fixture by probe’s values.
    g. Move sliders up and down to make color correction.
        i. You can add or subtract 0-255 units to/from any color component. Clamped result will still be within DMX valid values(0-255)



Color probe(experimental).

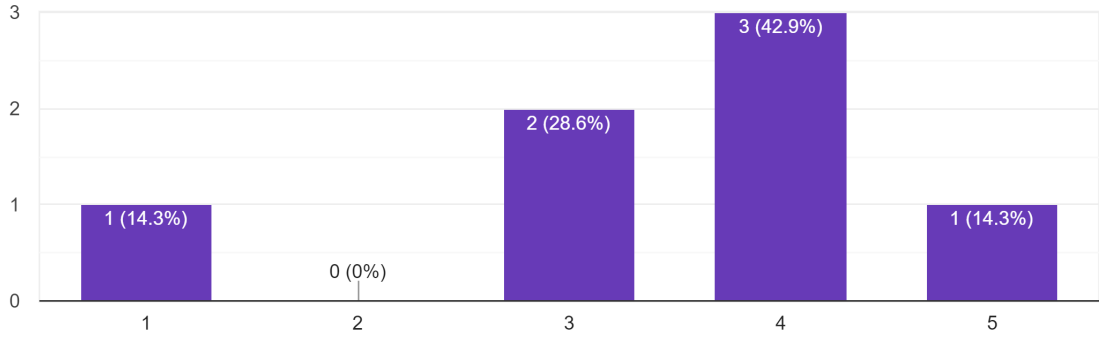
21. To place the color probe in the scene drag and drop ColorProbeSet prefab from Assets/Prefabs/Tools/ColorProbSet in the scene it will position itself in front of the camera. Offset could be changed in the inspector.
22. Please exclude the TransparentFX layer from the culling mask of your Main Camera to make the color probe invisible.


Camera
General
Background Type
Color
Background Color
HDR
Culling Mask
Mixed...
Volume Layer Mas
Volume Anchor Ov
Probe Layer Mask
Occlusion Culling
Projection
FOV Axis
Field of View
Link FOV to Physic
Nothing
Everything
Default
TransparentFX
Ignore Raycast
Water
UI

Survey results

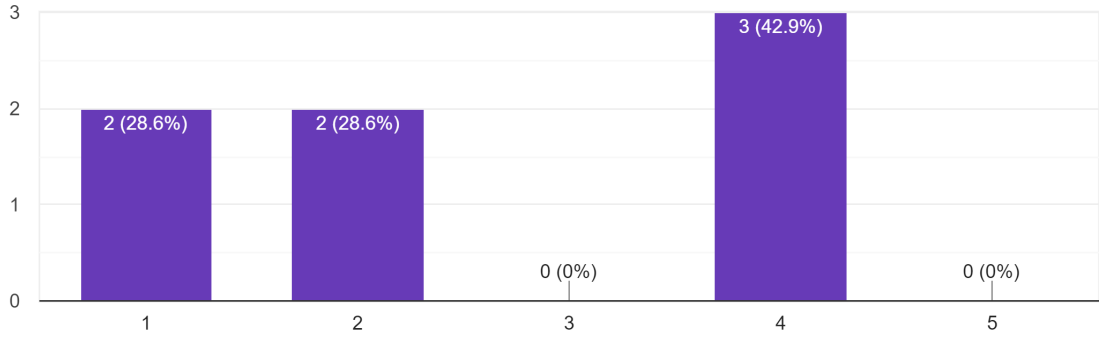
What is your expertise level in real time rendering engines?

7 responses



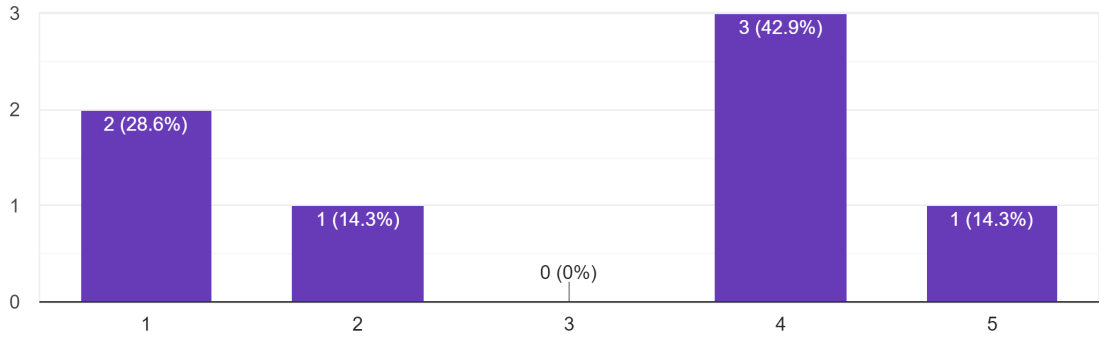
What is your expertise level in virtual lighting?

7 responses



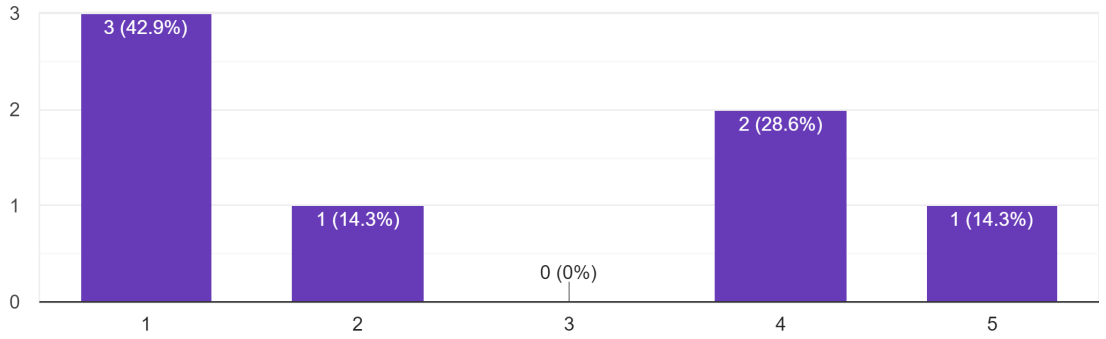
What is your expertise level in practical(real) lighting?

7 responses



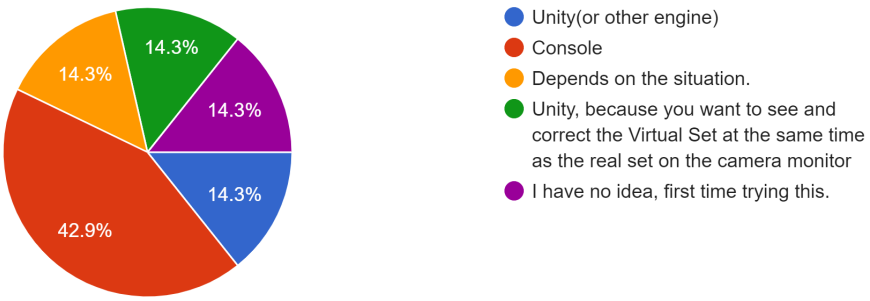
What is your expertise level in DMX control?

7 responses



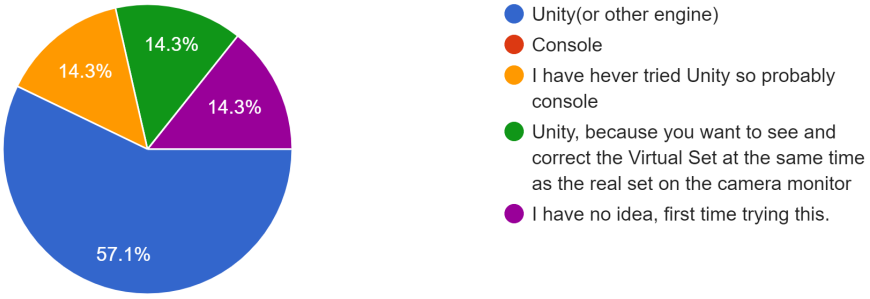
If you have to control on set (real) lights during VP session would you prefer to control them from Unity or a standalone (physical) console?

7 responses



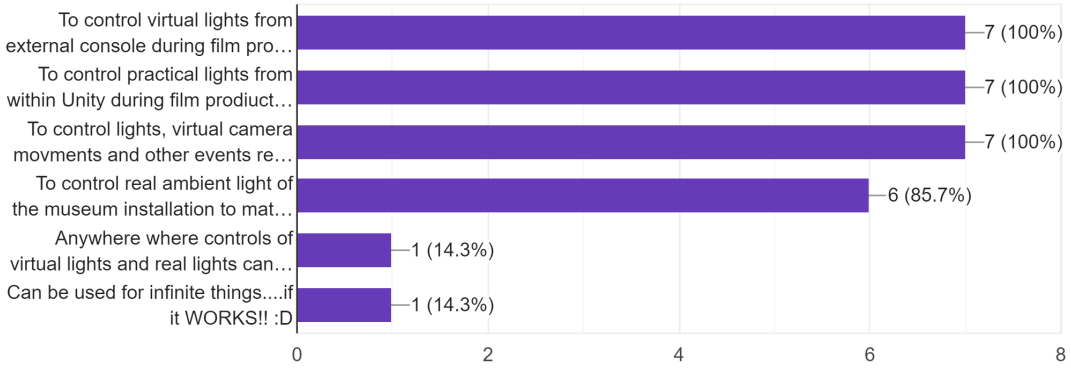
If you have to control virtual lights during a Virtual Production session would you prefer to control them from Unity or an external (physical) console?

7 responses



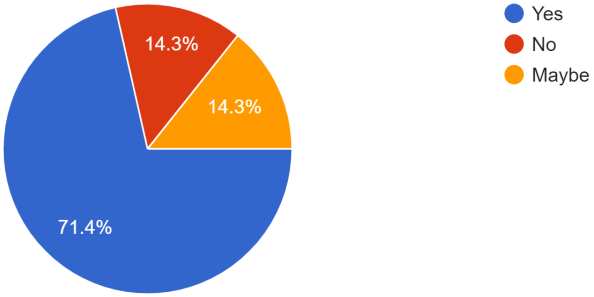
In which of these scenarios do you think DMX tools for Unity can be used?

7 responses



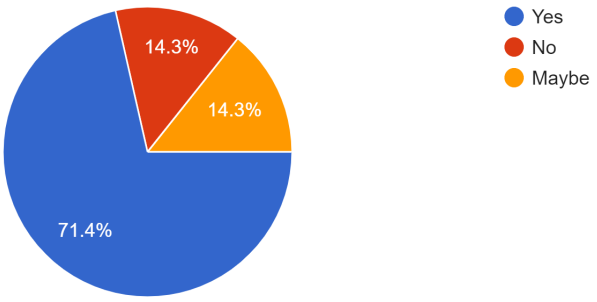
Would you like to have art-net out option.

7 responses



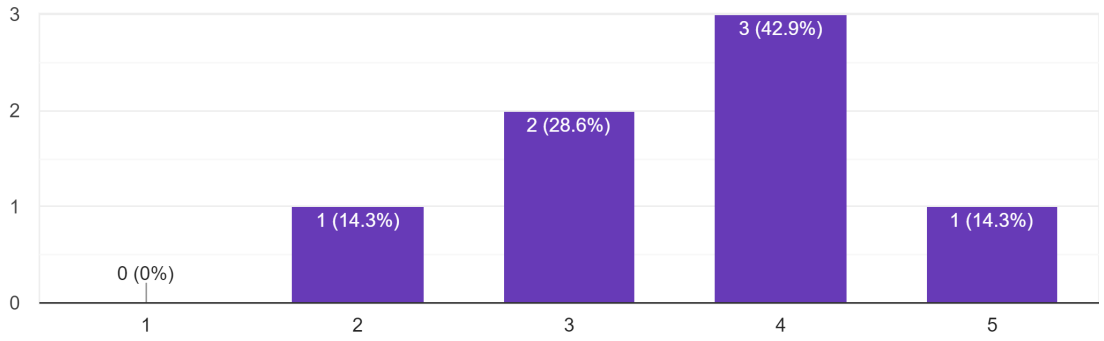
Would you like to have DMX in option.

7 responses



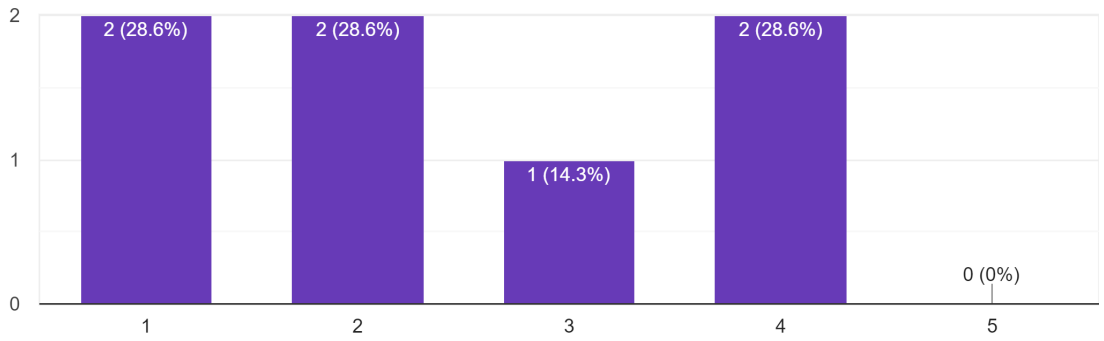
Ease of installation

7 responses



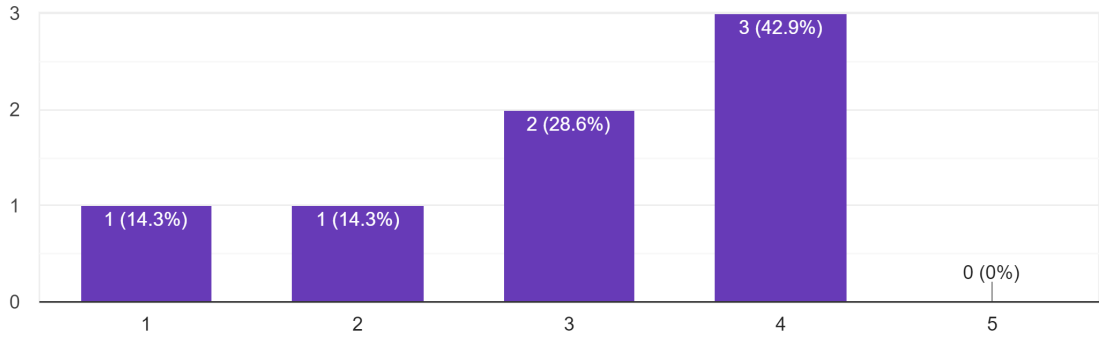
Ease of use

7 responses



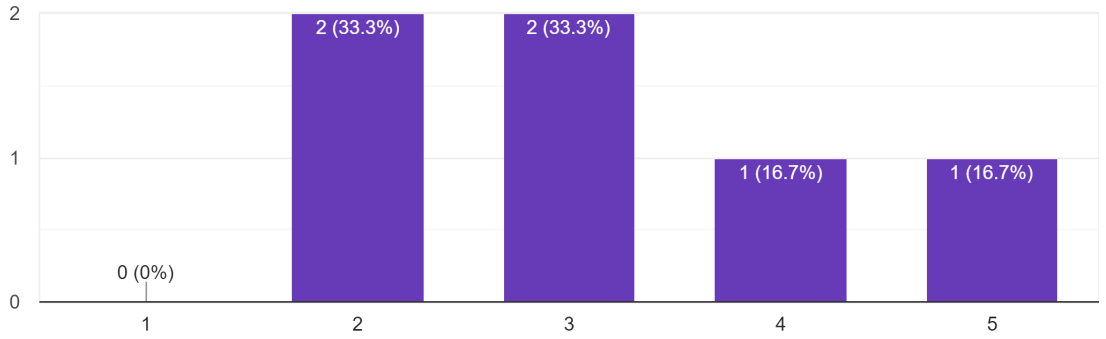
Ease to learn how to use

7 responses



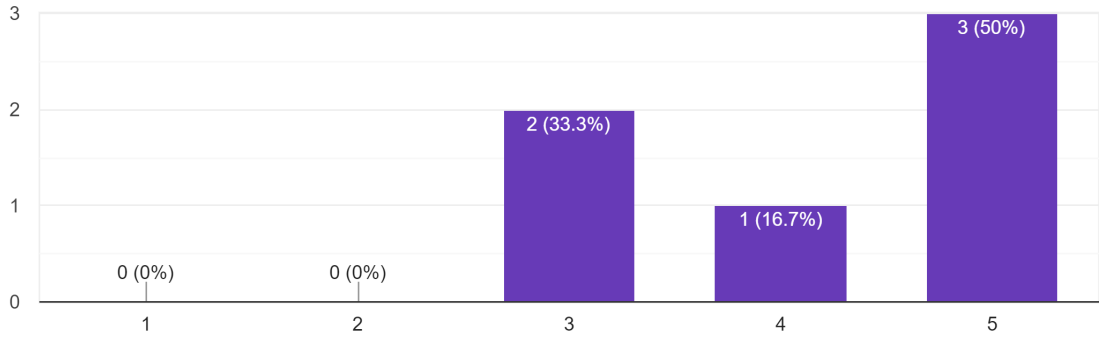
Hardware compatibility

6 responses



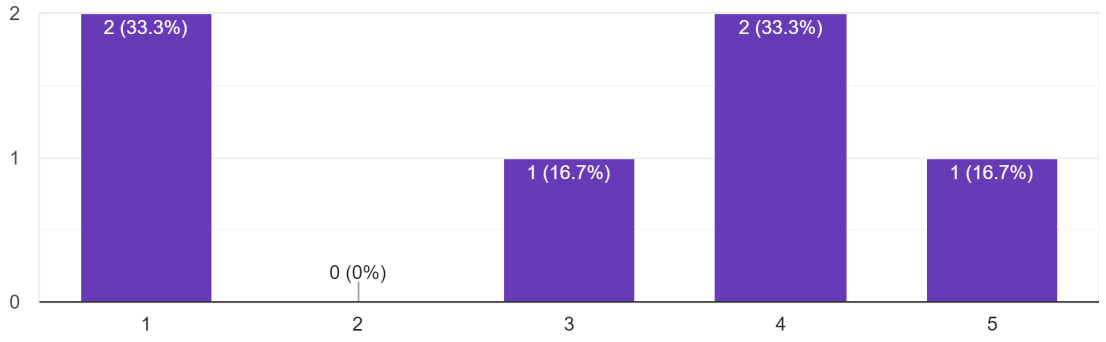
Operating system compatibility

6 responses



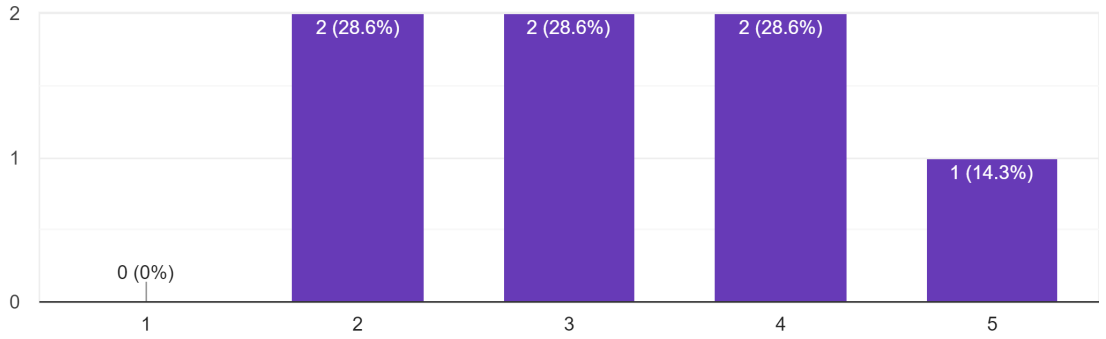
Ability to integrate with other apps

6 responses



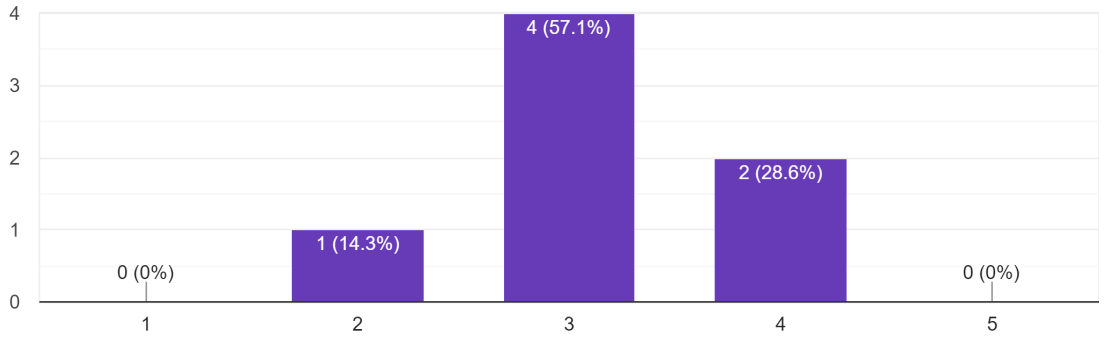
Consistency with Interface

7 responses



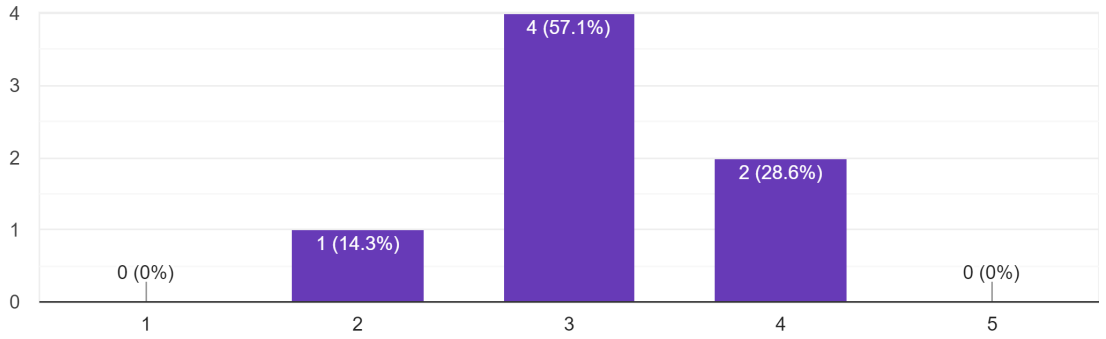
Overall reliability

7 responses



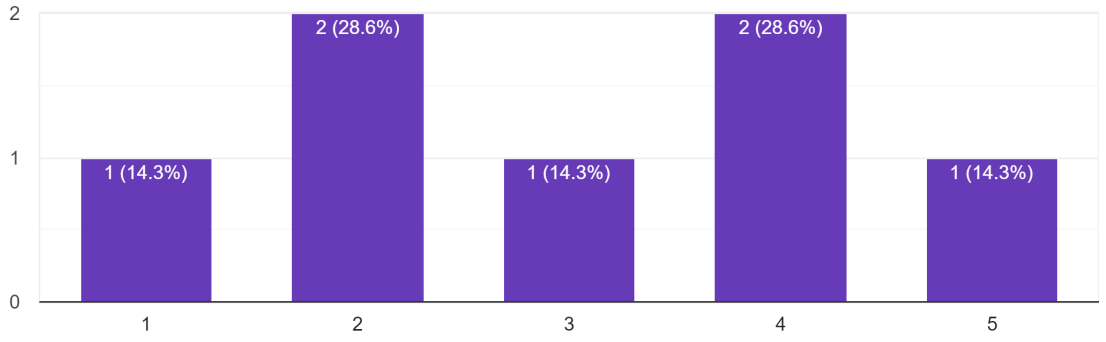
Overall performance

7 responses



Software intuitivity

7 responses



How likely are you to recommend our software to others?

7 responses