Developing rope winder machine in Unreal Engine

A K M Bayazid

December 3, 2023

Abstract

The aim of this paper is the development and implementation of a rope winder machine using the Unreal Engine. This machine provides a realistic and immersive virtual environment for the optimization of the rope winding process. This project explores the rope-winding mechanism and virtual simulation technologies. Besides, physics simulations are implemented for the interaction between ropes and machine components.

1 Introduction

The Rope Winder Machine is a mechanical device that is used in various industries such as shipping, construction, and logistics for efficient rope handling. However, it is widely used in textile manufacturing. Implementing a rope winder machine virtually for the rope handling process can be a challenging task. In this paper, we will discuss the implementation process. The primary objective of this process is to design and develop a rope winder machine using the unreal engine that uses the same models as the rope winder in Figure(1) and has the same functionality. In order to do that we are to use C++ and Visual Studio to handle movement logic, and craft the various parts. This should then result in you only needing to move the knob in order to move the rest of the system, thus resulting in the ropes winding around each other.

Figure 1: A real life example of a rope winder crafted using 3d printed versions of the models we are to use

For our implementation, we will follow the structure of a real rope winder machine and design accordingly. After that, we will work with C++ code to move different parts. In addition, we will connect all the components properly and implement physics constraints for smooth connectivity and rotation. Besides, we will demonstrate rope using bones and physics constraints and rig it in Blender. After completing the implementation of the rope handling process, we will work with various shaders and Niagara FX.

2 Tools

In order to try and create as realistic of a rope winder as possible there are some tools we have to use in everything from creating the model inside the environment, scripting the behaviour of various parts, and designing the look.

2.1 Unreal engine

The Unreal Engine is a game engine created and maintained by Epic Games. It is a powerful engine with lots of industry use. However, it is widely used in game development where developers can build simulations, edit videos and sounds and render animations. We use Unreal Engine version 5.3.2 for our implementation.[1]

2.2 C++

C++ is one of the world's most popular object-oriented programming languages which gives a clear structure to programs and allows code to be reused. It can be found in today's operating systems, Graphical User Interfaces, and much more. It is the main coding language of the Unreal engine and by connecting Visual Studio to Unreal, we can create and edit properties directly [2]

2.3 Blueprints

The Blueprint Visual Scripting system in Unreal Engine that uses a node-based interface to create game-play elements. They are an important part of creation in Unreal, and can be used to handle various functions, like controlling the character. In many ways Blueprints and C++ are similar. They are both object oriented and both can achieve the same results when it comes to implementing functions. However with blueprints, the options are limited to what has been already, while C++ has more freedom. [3]

2.4 Blender

Blender is a free 3D modelling software that includes tools for producing 3D elements, rigging, sculpting, and animation. Blender is commonly used to create 3D objects, which are then loaded into Unreal Engine and utilized to create the game. [4] The most commonly used format for Blender is FBX.

2.5 Materials and Shaders

A material is anything that may be applied to an object or mesh to define its appearance. A material is used to set the value of a shader-available parameter, such as color, numeric values, and so on. [5] A shader is a small program that instructs the GPU on how to render an item on the screen and the computations that must be performed on the object. It deduces the visual of an object during the rendering of a scene. This can be anything from raw colours to lighting and shadows, to special effects, like having a texture disappear and reappear, change colour, etc. [5] However, a shader needs to be attached to a material. Besides, it calculates the necessary degrees of light, darkness, and colour during the shading process while producing a 3D scene. The way one makes shaders in the Unreal engine is similar to how one makes blueprints with nodes but a programming language is used to create shaders like HLSL and GLSL. Here, we will use High-Level Shader Language shaders and Niagara FX. HLSL is used as a way to reduce the amount of nodes needed to craft a Shader. By using a custom node and putting in the HLSL code. Most of these shaders then follow a UV map that dictates, how they are placed on a model.

3 Related works

While our task is to implement it in unreal based on a provided model, there are those that have analyzed the rope winding mechanism for other purposes, like in the research paper[6] . This paper explored the dynamic aspects of rope winding and unwinding using a motor, gear, and winch system. The authors focused on the Cable-Suspended Parallel Robot (CPR) structure known as the RSCPR(Rigid ropes S-type Cable-suspended Parallel Robot) system. In addition, they developed two software packages, JUMPWIND-OW and JUMPWIND-RSCPR, to analyze the non-linear dynamics of rope winding and unwinding. Here, the RSCPR system consists of three motor-winch subsystems each controlling a rope linked to a camera for 3D spatial movement. This paper also introduced the concept of "rope jumpy non-linear in one row radially multilayered winding (unwinding)", modelling the process kinematically and dynamically. They investigated the effect of the rope jumpy winding process on the response of the rest of the mechanism.

4 Methods

In order to craft the rope winder we can see from figure(1) there were a few steps we had to do at the beginning.We were provided the files for the models that we were to use, but had to set up everything ourselves. We started off by making a empty C++ project in unreal. We then created a bunch of C++ classes, that would represent the various meshes we were to use and made blueprints based on our C++ classes. This was done in order to make sure that we could more easily

alter and control our meshes. Then we made sure to have one C++ class representing the front, and one big one for all of them together, that way we could easily assemble the rope winder in the big class, but also test out various functionalities and interactions in the smaller ones. Do to the nature of Unreal and how things are set up, it is not completely possible to replicate the real life rope winder, with the simulated one. One of these major limitations is that while as you can see on figure(1), the rope winder does consist of the same main components, it uses nuts and bolts to keep things like the Knob, front and Annulus together. In order to make up for this we need to use physics constraints and constraint to lock the various components in place to simulate them being properly connected.

4.1 Knob and Annulus

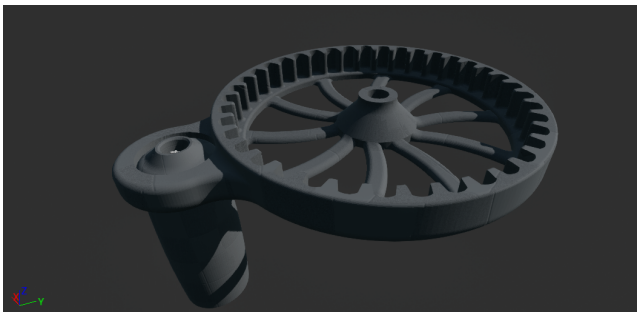

Figure 2: Annulus and Knob together

The way the knob ans annulus is set up is rather simple, the knob has been coded to rotate around the center of the blueprint, which in this case, is the center of the annulus. Then by using a physics constraint that has the annulus as the parent and the knob as a child, i have locked them together so the knob moving will also affect the annulus. This results in the combination looking like figure(2).

4.2 Gear Hook, and Anchor

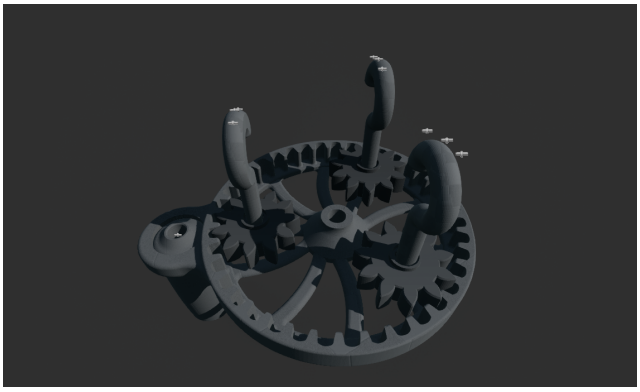

Figure 3: The model after we have added gear, hook and anchor to it

In order to simulate the fact that rotating the knob and annulus, would in turn rotate everything else, there were a few things that had to be done. We started by placing the gears inside the annulus. After that we made sure to lock movement and rotation, so it could only rotate in the correct direction. And in the end we activated physics. That way when the annulus would interact with them the gears would rotate alongside it. The hooks and anchors were set up in the same way, but were instead positioned in relation to the respective gears and each other, so that they could more smoothly operate with each other. The positions of all these objects were fine tuned to get the most optimal result. The end result can be seen on figure(3).

4.3 Rope

In order to make something that would be able to act an actual rope, what we needed to do was to take use of a skeleton mesh. A skeletal mesh is an asset that contains 2 parts. A visual part and a bone part. The visual part represents how it looks, in our case a long cylinder. And the bone part which can be seen in figure(4), dictates how it can move. This is

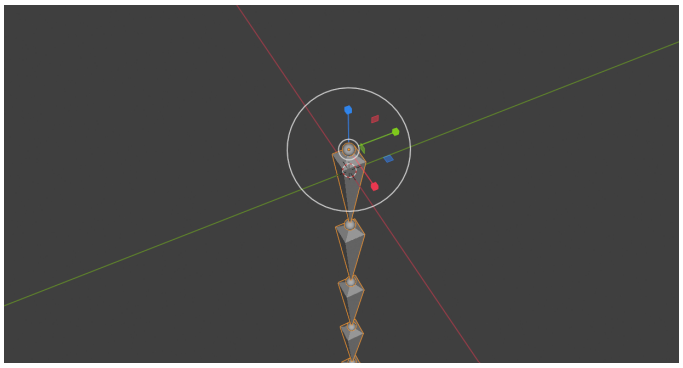

Figure 4: The bones inside the rope

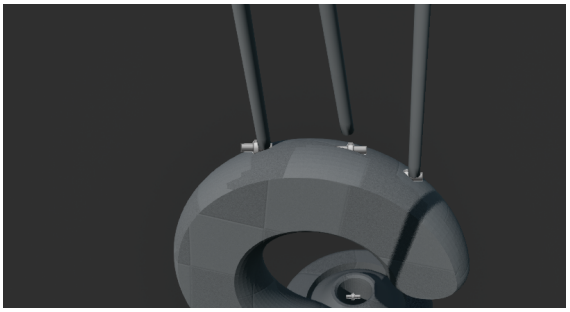

Figure 5: An image representing how the ropes are connected to the hooks with a physics constraint

what allows it to look and interact as a real rope. And by following the tutorial in the video[7] we ended up crafting the rope. One end of the rope was then connected using a physics constraint to the hook as seen in figure(5). And the other to one of the holes in the shuttle. The constraints where then adjusted so that when the hooks would spin the rope would follow along, but still be locked to that specific part of the shuttle. This was repeated until we had 3 ropes going on each of the 3 hooks.

4.4 Front and Shuttle

The front and the shuttle were the two most basic things to implement, the shuttle exists to make it so the ropes has a point to attach to, and the front is there to make it look like figure(1). Do to how the collision boxes are larger on the models, then the actual model would imply. Therefore we turned off the collision of the front.

4.5 Shading and animation

As far as shading is concerned we decided to make the ropes follow a RBG(Red, Green, Blue) theme and as such the code used to make something red, has a blue and green counterpart, with the same setup, but the color values changed. Like how the hooks and anchors have all been made using a basic color on them, this also applies to the design of the ropes, tho they use more intricate effects or shaders. The front was made orange in order to look like the front from figure(1). The gears were all made using following a video tutorial [8] to make the effect that the gears are black and have the complimenting colored hook emerging from the center of said shader. The shuttle was made using a modified version of the guide from following a video tutorial[9] combined with things we learned from following an online video tutorial[10], in order to make a ripple effect appear on it and to make the ripples periodically change color among red, blue and green and we made the base color of the so it looks like the real life counterpart. The knob was made using a modified version of the code from following a video tutorial [11], to give it a looping effect with a marble texture instead. The annulus shader was made by following a video tutorial [12] and was done to make it water ripple effect, to make it have a complementing color theme to the shuttle. The ropes connected were made invisible and instead have a Niagara particle effect following where the rope is supposed to be, that we made trough watching following a video tutorial[13]. Following tutorial from online guidance [14], the ropes were connected to the red hook, in order to have the ropes be different from the previous ones and to give them a glass like appearance. The final set of ropes we tried to make look like real ropes by using an altered version of what we learned from an online tutorial [15] and a free texture we got from a website [16].

4.6 Buttons

In order to make the buttons, we ended up following a video tutorial [17], and made some adjustments to how it works to make it work better for our project. There are 2 main components to this. The widget C++ class and blueprint, and the main "ropewindermechanism" C++ class and blueprint. The way it works is that the widget blueprint for the buttons has been connected to the ropewindermechanism blueprint, so that when the mechanism is in the level and you hit play, the buttons will appear on screen as can be seen in figure(6), in the top right corner, and by pressing 1 of them, the speed will be adjusted. Either increasing it, decreasing it, or reversing it. To make this happen, we needed to make it so each individual button has a name associated with it. These names corresponds with a "Onclicked" function in the C++ class, that will then activate a function the the main "ropewindermechanism" class. These functions are related to the text written on the buttons in figure(6). "Start" sets the speed to 40, "Increase" adds 10 speed, "Decrease" reduces it by 10, "Reverse" makes it go in the opposite direction in the same speed, and "Stop" reduces the speed to 0.

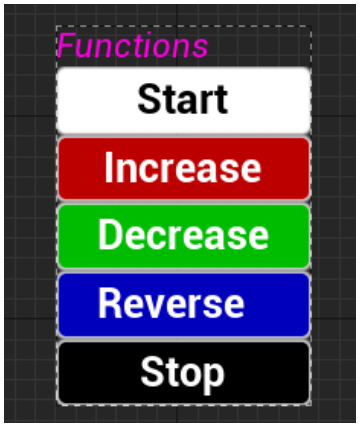

Figure 6: The setup of the buttons

4.7 Sound

To make our program realistic we recorded ourselves using the rope winder from figure(1), we then converted the sound file to a ".wav" format and set it up to be played when the start button is pressed. ‘

4.8 Environment

To make the environment we decided to just use some of the default cubes and cylinders, with basic shaders added to them, to make the area resemble a chair and desk from our classroom. The end result can be seen on figure (7)

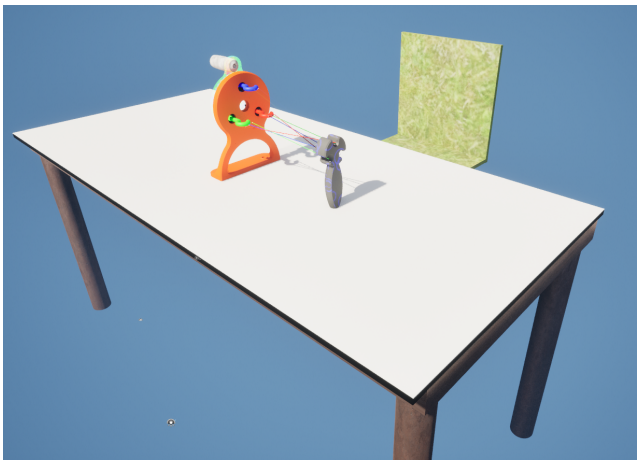

Figure 7: Rope winder standing on the desk we created

5 Results

As can be seen by comparing figure(8) and figure(9) the rope winder is functioning and once ran for a while the rope does twist around each other. And the buttons and sound are working properly once pressed.

6 Discussion

Tho figure(8) and figure(9) may give the appearance of it working properly, but there are some issues with how the program works. A lot of these issues stem from the fact that the models we were provided were never intended for unreal use, but was rather meant for 3D printing. Because of this, the collision of the shapes are off resulting in things like the gears and annulus, not properly colliding and as such the gears have less speed than they should. This results in issues, like how the green hook is no longer aligned with the other hooks in figure (9) compared to the others, unlike in figure(8). Another issue caused by this is that the models lack a proper UV map for textures, and we have to use an auto generated

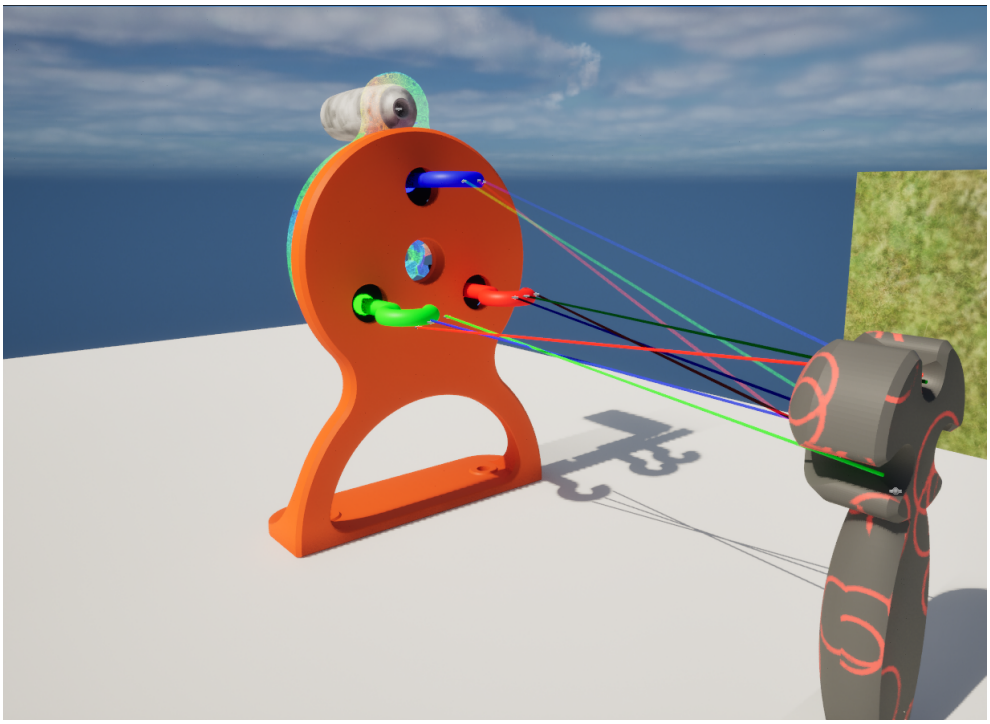

Figure 8: RopeWinder before we start

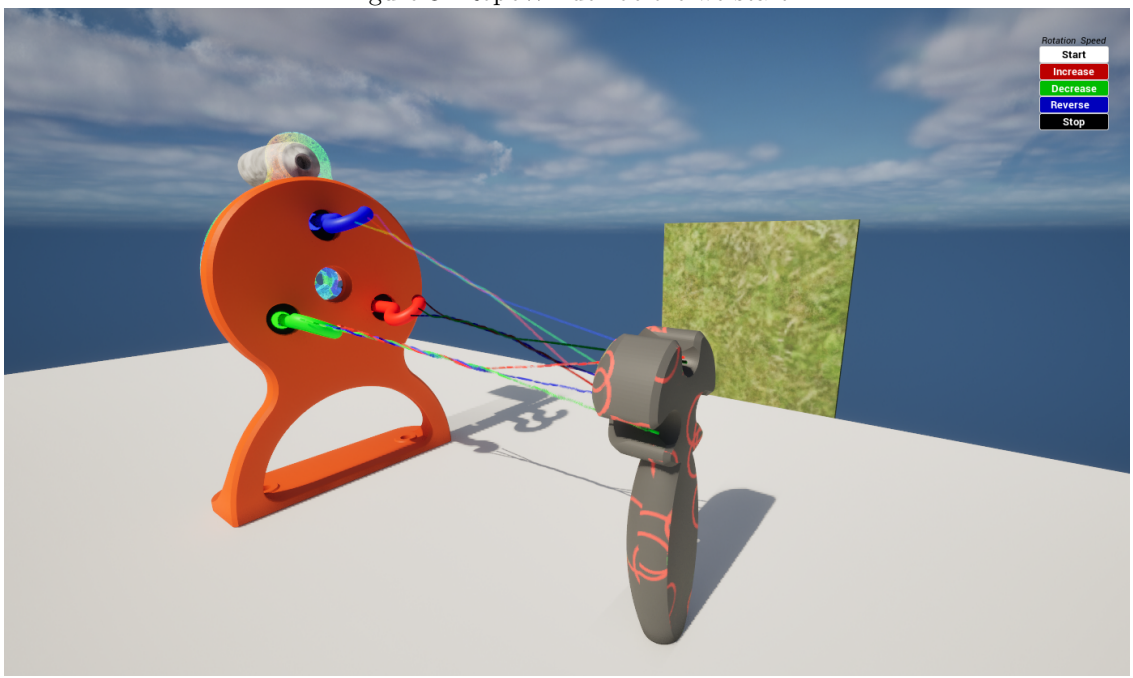

Figure 9: The RopeWinder after we have ran it for a while

one that is generated by Unreal. This makes it so, some of the textures aren't applied correctly. As can be seen when comparing figure(10) and figure(11), while it looks fine when using a shader that is just one color, once you start applying details the texture starts to break.

Figure 10: The front using a basic shader to make it look orange

Figure 11: The front using one of the default brick shaders that is a part of the starter content of a project

The best solution to both of these, would probably be to use blender in order to craft new and proper models for use. However, other methods exist, like trying to use Unreal's inbuilt functions to craft a new collision box, or having it in code, that force is transferred. However these solutions aren't as good, as having proper models. Another issue is with the rope physics. The physics of the ropes are not working properly, and as such i have had to turn of gravity on all of them, this also means everything starts to break apart once you have too much speed. These issues seem to stem from the parameters and length of the rope. As having a shorter rope, doesn't have as many issues. The solution for this would likely be to change the ropes in blender, or changing stuff like mass and scale in order to fine tune it into working better.

7 Conclusion

The implementation of the Rope Winder Machine Simulator project represents the designing and integration of advanced features. It offers a realistic and immersive winding process that can analyze and optimize the rope winding process allowing users to acquire hands-on experience in a virtual setting. The use of physics constraints provides accuracy and fidelity in displaying interactions between rope and machine components so that users can get real-life experience. In addition, C++ and Blueprint provide freedom to work independently and precisely that make the rope winder machine more realistic. However, the most challenging part of this task was constructing ropes and making them twisted. For this, we demonstrated bones and physics constraints. Additionally, we worked with rotation speed and added some buttons such as start, increase speed, decrease speed, reverse speed and stop. These buttons make this machine a real one. We also implemented various shaders so that the rope winder machine looks captivating and pragmatic. Not only that, but we also worked with HLSL and Niagara FX. Another fascinating option in this machine is the sound that we collected from a real rope winder machine. This provides a feel of realism. This demonstration exceeds our expectations regarding functionality, responsiveness, and user engagement. The intuitive user interface facilitates seamless control and monitoring of the rope winder machine and enhances user experiences in both the rope-handling process and the virtual simulations. Nevertheless, this project opens vast opportunities for further research and development. In the future, we can explore additional features and enhance realism (more sophisticated physics models). Additionally we would also make it so that the models are functioning correctly so that we can have proper textures and physics. Additionally, we will work on making coal in the shuttle and storing that coal. The Rope Winder Machine Simulator has become a successful convergence of theoretical understanding, engineering practical expertise, and virtual technologies.

References

[1] Adam Speight. What is unreal engine? epic games' creation tool explained, 1-10-2023. https://www.trustedreviews.com/explainer/what-is-unreal-engine-4342624.

[2] Epic Games. Programming with c++, 2 December 2023. https://www.trustedreviews.com/explainer/what-is-unreal-engine-4342624.

[3] Unreal. Introduction to blueprint visual scripting, 10-2023. https://docs.unrealengine.com/5.3/en-US/introduction-to-blueprints-visual-scripting-in-unreal-engine/.

[4] Charlie Fitzgerald. Unreal engine vs. blender - comparison guide, 2 December 2023. https://www.trustedreviews.com/explainer/what-is-unreal-engine-4342624.

[5] armDeveloper. What is a material and a shader?, 2 December 2023. https://developer.arm.com/documentation/102676/0100/What-is-a-material-and-a-shader.

[6] Ljubinko Kevac, Mirjana Filipovic, and Aleksandar Rakic. Dynamics of the process of the rope winding (unwinding) on the winch. *Applied Mathematical Modelling*, 48:821–843, 2017.

[7] Lusiogenic. Ue4 rope using bones and physics constraints in unreal engine 4 blender chain tutorial how to, 2 December 2023. [Video]https://www.youtube.com/watch?v=hEMcr98cDJE.

[8] Renderbucket. Ue5 tutorial - hlsl - introduction to raymarching, 2 December 2023. [Video]https://www.youtube.com/watch?v=dSY$_8Ii - HcIlist = PLoHLpVCC9RmMMmW5eP1aAyJrTjxd46rx_{index=6.}$

[9] Renderbucket. Unreal engine 5.3 - animated raindrop ripples in hlsl, 2 December 2023. [Video]https://www.youtube.com/watch?v=I4b1Jj$_V If8list = PLoHLpVCC9RmMMmW5eP1aAyJrTjxd46rx_{index=15.}$

[10] RenderBucket. Unreal engine 5 tutorial - technical shading - introduction to hlsl, 2 December 2023. [Video] https://www.youtube.com/watch?v=lsXB1PQdGx0list=PLoHLpVCC9RmMMmW5eP1aAyJrTjxd46rx.

[11] Tech Art Aid. Ue5: Burning wood material // unreal engine tutorial, 2 December 2023. [Video]https://www.youtube.com/watch?v=3pqqAWX7zlQ.

[12] Coreb Games. Unreal engine 5 tutorial: Creating a rainbow ripple effect with materials — ue5 guide, 2 December 2023. [Video] https://www.youtube.com/watch?v=uRk6qS7dnTY.

[13] Kat Sullivan. Unreal niagara 4: Skeletal meshes, animations, and particles (4.26), 2 December 2023. [Video] https://www.youtube.com/watch?v=ezqaH7nvG8o.

[14] Unreal Engine. Materials - tinted glass part 2 — tips tricks — unreal engine, 2 December 2023. [Video] https://www.youtube.com/watch?v=XRwFh6s5wqEt=1s.

[15] Ben Cloward. Fabric shaders - advanced materials - episode 10, 2 December 2023. [Video]https://www.youtube.com/watch?v=cTqQ4AYNTKMt=1s.

[16] Arroway Textures. fabric 021, 2 December 2023. https://www.arroway-textures.ch/products/fabric-021/?texture-tags=414texture-filter=.

[17] Institute of marxism leninism. How to create widget via c++ — game development — unreal engine 4 tutorial, 2 December 2023. [Video]https://www.youtube.com/watch?v=Ho5DcpfI3NMt=722s.