International Journal of Internet, Broadcasting and Communication Vol.14 No.1 105-115 (2022)
http://dx.doi.org/10.7236/IJIBC.2022.14.1.105

IJIBC 22-1-13

Realistic Visualization of Car Configurator Based On Unreal Engine 4(UE4)

Yiming Zhong*, Tae Soo Yun**, Byung Chun Lee***

**Ph.D, Division of Digital Contents, Dongseo University, Korea*
***Professor, Division of Digital Contents, Dongseo University, Korea*
****Professor, Division of Digital Contents, Dongseo University, Korea*
mr.92est@gmail.com, tsyun@dongseo.ac.kr, myeyes71@gdsu.dongseo.ac.kr

Abstract

The platform for displaying cars has been changing with the times. From the popularity of paper media to the rise of computer graphics, the improvement of technology has brought more space and possibilities to the automotive industry. Yiming Zhong proposed the workflow of car configurator through Unreal Engine 4 to implement the basic functions of configuration in 2021, according to Yiming Zhong's final presentation, there is still room to improve the realism of graphics and functionality of the car configurator. Therefore, in this paper we propose to upgrade the car shaders and lighting environments according to the real-world physics and add multi-scenes switching function to car configurator. However the multi-scenes switching function also brings a large amount of data, which leads to the problem of display lag. At the end of the paper, we use the level of details(LOD) process to reduce the amount of data for real-time computing in Unreal Engine 4 and the increase of frames per second(FPS) values verifies the feasibility of our optimization solution.

Keywords: *Car configurator, Unreal Engine, Realistic, Multi-scenes switching, Optimization*

1. Introduction

In order to deliver more information to customers, the car display platform has a variety of forms. The most common form today is to offer car experience services through offline stores. The offline store will offer a variety of real cars on display so that consumers can get a face-to-face view of the car's appearance and configuration. The advantage of the offline store is that it allows consumers to have a visual experience about the car. However, the disadvantages of offline stores are also obvious: firstly, the operating costs are quite high, and secondly, consumers need to spend time and effort to reach the offline stores.

According to consumer demand for convenient experience, the online store has become another major form of car display. The rapid development of game engine in recent years has brought new possibilities for car display platforms. Unreal Engine 4(UE4) released the Product Configurator template in September 2019, which allows for product color and material changes [1]. The color and material of the product can be switched by this configurator, and this template provides an editable UI interface with the real-time rendering function of UE4, the graphics performance of this configurator is also realistic. Based on the workflow of the product configurator, Yiming Zhong proposed the workflow for implementing the car configurator in UE4. From the final presentation, there is still much room for completing the image realism and the enrichment of the

Manuscript Received: November. 25, 2021 / Revised: November. 28, 2021 / Accepted: December. 5, 2021
Corresponding Author: myeyes71@gdsu.dongseo.ac (Byung Chun Lee)
Tel: +82-51-320-1534, Fax: +82-51-320-2189
Professor, Division of Digital Contents, Dongseo University, Korea

functions. So we propose the improvement proposals to the realism of graphics and functionality of car configurator in this paper. In the graphics part, the car shaders in UE4 are upgraded with reference to real-world physics, and the Sun Position Calculator plugin is used to build lighting environment closer to the real world. In the function part, the multi-scenes switching function is implemented, the display scenes can be switched in real-time. Totally four scenes are created in the configurator: beach, desert, showroom, and bridge. and each scene contains an independent lighting environment, the lighting environment will also change when switching scenes. However, multi-scenes switching also brings more data volume, which can lead to display lag in the car configurator. For the lag problem we apply the level of detail (LOD) method to simplify the background model, and optimize the rendering settings in UE4. After applying the optimization solution, the car configurator shows a significant increase in frames per second (FPS) value, which also verifies the feasibility of the optimization solution and gives the car configurator a smooth and fluid experience while running multi-scenes switching function.

2. Related research

2.1 Development of the car configurator

The most common form of car configuration in existence is the brochure, which is simple and inexpensive to produce, but the presentation is not intuitive and the amount of information that can be conveyed in this way is very low due to the limitations of the paper medium. The second common approach is the online configurator, which is based on the same basic principles as the brochure, and it is often found on the car's official website, where the user can click on options to navigate through different car configurations and switch between specific viewing angles. Figure 1 shows the car configurator interface of Porsche's official website.

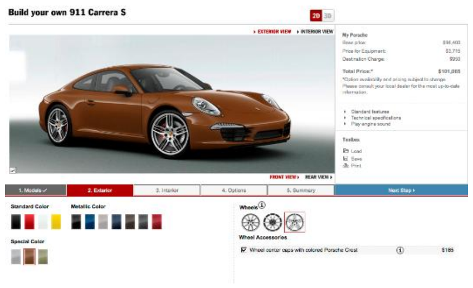


Figure 1. Porsche online car configurator

With the enhancement of the function of the game engines, car-related content developed through game engines has also become popular in the market. UE4 and car configurator related content first appeared in racing games. The game interface provides players with rich options for customization, and players can observe the change of car's appearance in real time. The car customization system in the racing game can be seen as the prototype of the car configurator. Figure 2 shows the configurator interface from UE4 racing game “Assetto Corsa” [2].

Figure 2. Car configurator in game “Assetto Corsa”

In 2014, Armin Pohl, CEO of visual effects company Mackevision, proposed the vision for the car configurator of the future. At the 2018 Geneva Motor Show, Ferrari announced a demo of a car configurator using a game engine, and with the development of game engine capabilities, the car configurator has achieved impressive improvements in terms of visual and functional [3]. In December 2020, Unity announced Unity Forma, an automotive configurator template developed using the Unity platform that supports importing 3D data for cars and customizing the configurator options to the customer's reality. Unity Forma is a car configurator template that supports importing car 3D data and customizing the configurator options according to the customer's reality. Figure 3 shows the service process of Unity Forma. However, Unity Forma a payment of $4,980 for access, a one-stop-shop solution for corporate users but too costly for individual users [4].

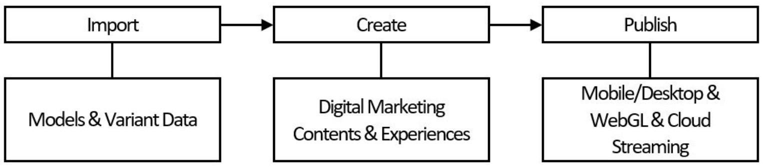

Import
Create
Publish
Models & Variant Data
Digital Marketing Contents & Experiences
Mobile/Desktop & WebGL & Cloud Streaming

Figure 3. The production process of Unity Forma

In April 2021, Epic Games partnered with Audi to launch a prototype of the car configurator, a template that demonstrates paint color changes, opening and closing doors and roof, changing wheel styles, changing the viewing angle [5]. The template is based on the same structure of the product configurator, all elements are controlled through Variant Manager.

2.2 Implementation of the car configurator in UE4

The production of car configurator can be divided into four steps, the first step is the 3D content production, this step includes the creation of 3D modeling and texture maps. The second step is Look-development in UE4, this step includes the import of 3D data and texture maps. editing shaders, lights, and render settings. The third step is the implementation of configurator, creating the car configuration switching function and user interface through Variant Manager in combination with blueprint. The fourth step is the final presentation, where the car configurator is packaged according to different purpose. Figure 4 shows the workflow of implementing the car configurator in UE4.

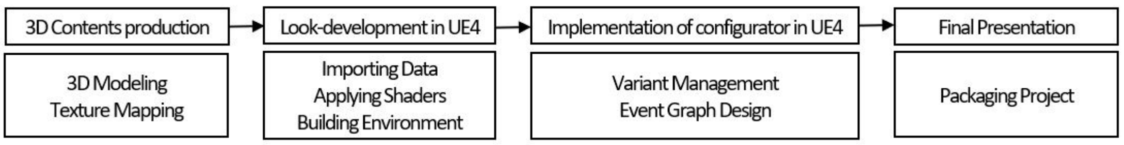


Figure 4. The workflow of implementing car configurator in UE4

In 3D content production part, the first step is creating 3D car model, using 3D software to create car model, and then output the model in FBX format. The second step is texture mapping, which gives the car a basic texture appearance. Look development in UE4 consists of three steps. The first step is the importing 3D models and maps to UE4. The second step is applying material shaders to 3D model. The third step is environment building, creating realistic scenes and lighting effects for the display of the car. Configurator implementation is the process of interacting with a car model through event graph, creating an interface that allows the user to control the car configurator by clicking on buttons. UE4 released the product configurator template in version 4.25. According to the information provided by UE4, the event graph is designed to use the Variant Manager plug-in to systematically classify customizable parts of the product and set each model or shader as a separate variant set in the category. For example, in the paint category, the paint options of red, yellow and blue are set respectively. The function can be understood as a switch that controls all the VariantSets. The function is connected to the VariantSet nodes. After nodes have been built, open the product configurator template, connect it to the built-in BP_configuration, the interface of the 3D car configurator can be activated by starting blueprint. The last stage is packaging. UE4 supports multi-platform applications. Depending on the final display platform, the car configurator can be converted to different formats [6]. Figure 5 shows the process of making a car configurator through the Variant Manager.

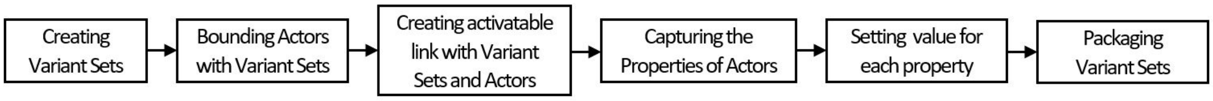

Creating Variant Sets
Bounding Actors with Variant Sets
Creating activatable link with Variant Sets and Actors
Capturing the Properties of Actors
Setting value for each property
Packaging Variant Sets

Figure 5. The process of Variant Manager in UE4

3. Realistic visualization of the car configurator in UE4

Based on the final presentation of Yiming Zhong's paper, in this chapter we propose the improvement process for enhancing the realism of the car configurator's graphics. The improvement process is divided into two sections, the first section is the improvements of the car shaders, and the second section is the improvements of the car display environment.

3.1 The improvements of shaders in UE4

This section contains the improvements of car paint shader and glass shader. These two materials arechosen because they both have highly reflective properties and complex layering structures, so they have more room for improvement and the coverage of these two materials on the car is also high, improving these two shaders has a very intuitive help to enhance the realism of graphics.

3.1.1 Car paint shader upgrades

Car paint surface is a complex structure, the first layer is an anti-corrosion material, the second layer is electro-coating, which allows the paint to adhere better to the surface of the vehicle. The third layer is the primer, which serves as a leveler, which is important since the cab often has marks and other forms of surface defect after being manufactured in the body shop. A smoother surface is created by leveling out these defects and therefore a better final product [7]. The fourth layer is the under coat, which gives the paint its basic color and metallic feel, according to the under-coat, which is often treated with aluminum flakes to give the paint a shiny visual effect in the light. It is usually sprayed on top of a colored base coat, clear coat is a glossy and transparent coating that forms the final interface with the environment [8]. Figure 6 shows the structure of car paint.

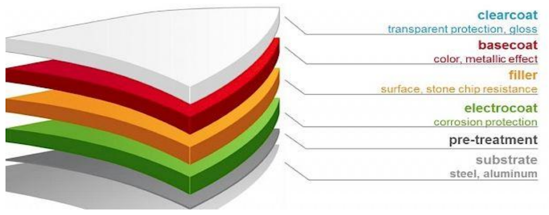
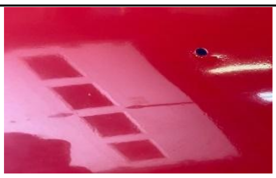
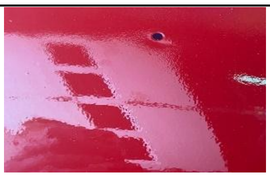


Figure 6. Layered structure of car paint

In the car configurator presented by Yiming Zhong, the texture maps are made by Substance Painter, with diffuse map controlling the base color, roughness map controlling the smoothness, metallic map controlling the metallic look and the normal map providing the bumpiness of the paint surface. This approach benefits from the ease of operation of Substance Painter and we can quickly achieve a good appearance performance. However, this approach does not match the real car paint structure. To get a more realistic car paint texture, we use the structure of real car paint as a reference and redesign the car paint shader in UE4.

First, we add a Constant3vector node to control the basic color of the car paint. Then we use the intensity value to control the metallic and roughness of the base paint. The clear coat is the outermost layer and it has an obvious effect on the final texture of the paint. We turn on the clear coat enable second normal option in the project setting and switch the shading mode of the material sphere to clear coat so that the clear coat function can be activated. In reality, the paint surface is not absolutely smooth, there are many tiny bumps on the clear coat, this phenomenon is known as orange peel. Table 1 shows the orange peel phenomenon on the car paint surface. We simulate the phenomenon of orange peel by adding a bump layer to the clear coat [9].

Table 1. The appearance of orange peel on paint

Phenomenon	Smooth	Orange peel
Appearance of car paint		

We add three new parameters: clear coat, flakes, and orange peel to the car paint shader. We also use intensity value to control the shader attributes. Compared with the previous way of describing materials through texture maps, our upgraded shader has more adjustable options, and the appearance of the shader can be adjusted according to different requirements. Table 2 shows the difference between the upgraded shader and the previous shader.

Table 2. Upgrade changes of car paint shader

Versions of shader	Previous version	Upgraded version	
Parameter Category	Base Color Metallic Roughness	Base Color Metallic Roughness	Clear Coat Flakes Orange peel
Screenshot in UE4			

3.1.2 Glass shader upgrades

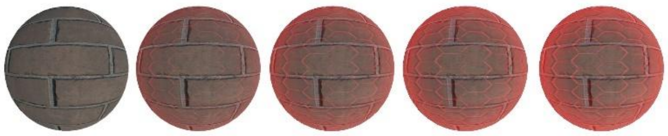
In reality, the structure of car glass can be divided into three parts: interior glass, blast-resistant laminate and exterior glass. Due to the existence of multi-layer structure, the reflection and refraction of car glass is also very complicated. To simulate the real texture of the glass, it is necessary to give the glass thickness in the 3D modeling part, and the key attribute of the simulated glass material is IOR (Index of Refraction), Index of Refraction is a value calculated from the ratio of the speed of light in a vacuum to that in a second medium of greater density. In order to help users simulate the texture of transparent objects, UE4 officially released the reference value of IOR [10]. As shown in Table 3, the IOR value of glass is defined as 1.52, which also provides an important reference for the simulation of glass material in this paper.

Table 3. IOR Value of common transparent objects

Material	IOR Value
Air	1.00
Water	1.33
Ice	1.31
Glass	1.52
Diamond	2.42

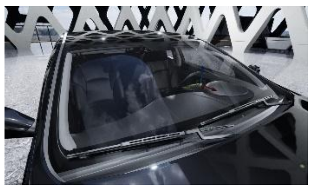
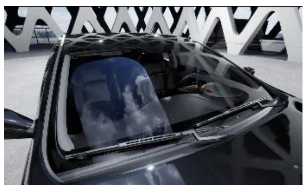
In the previous glass shader, three parameters including base color, roughness and opacity are used, ignoring the influence of IOR on the glass material. In this paper, we added the IOR attribute and the Fresnel attribute to improve the performance of the glass material. Fresnel is the effect of differing reflectance on a surface depending on viewing angle [11]. In UE4, the Fresnel material expression node calculates the falloff based on the dot product of the surface normal and the camera direction. Table 4 shows the appearance of Fresnel in UE4 based on different intensity value.

Table 4. Fresnel effect in UE4

Intensity Value	0	0.25	0.5	0.75	1
Fresnel Effect					

In reality, we also find that there is a color gradation phenomenon in car glass, which is generally expressed as dark corners around and lighter colors in the middle. The addition of Fresnel can simulate this phenomenon and further enhance the realism of the glass shader. Table 5 shows the upgrade changes of the glass shader and the final appearance in UE4.

Table 5. Upgrade changes of the glass shader

Versions of shader	Previous version	Upgraded version	
Parameter Category	Base Color Metallic Opacity	Base Color Metallic Roughness	Opacity IOR Fresnel
Screenshot in UE4			

3.2 The improvement of display environment

The improvement of the display environment is mainly divided into two sections: the modifications of lighting environments and the enrichment of scenes. In the first section, we mainly make improvements to the sunlight and the interior lighting environment of the car. To ensure the uniformity of the lighting effects inside and outside the car, we use the maximum and minimum exposure values in the post-process to control the light brightness of the car configurator to maintain an appropriate range to prevent over-darkness or over-exposure.

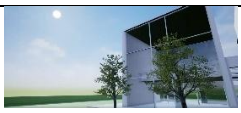
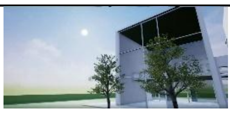
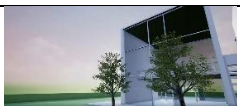
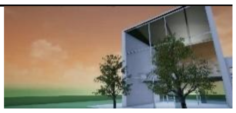
In the second section, we create four scenes: desert, beach, bridge and showroom to enrich the display scenes of the car. Each scene has a unique atmosphere that can bring a richer appearance experience. The addition of these four scenes also lay the foundation for the implementation of multi-scenes switching function in the fourth chapter.

3.2.1 Modifications of lighting environments

In the previous version of the car configurator, HDRI is used to control the global illumination of the scene. The advantage of HDRI is that the operation process is simple, and natural lighting effects can be obtained through HDRI map. However, the illumination information of HDRI can only be derived from the light source of the panoramic photo, and we cannot simulate the change of sunlight according to the time change in reality.

In this paper, we use Sun Position Calculator plugin instead of HDRI to simulate sunlight. Sun Position Calculator is a plug-in for simulating sunlight launched by UE4. Through this plug-in, we can define latitude, longitude, cardinal points, date and time of day. The sunlight changes according to these reference values [12]. This means that we can simulate the daylight at any place and at any time according to the actual situation. Table 6 shows the sunlight simulated by the Sun Position Calculator plugin under different time settings.

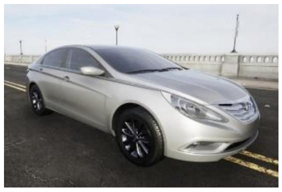
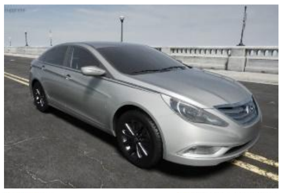
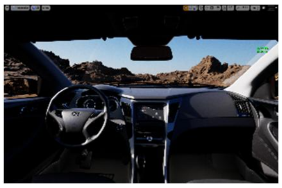
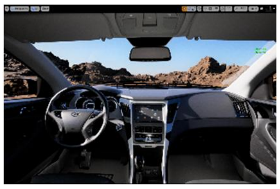
Table 6. The change of daylight caused by time settings in Sun Position Calculator

Time	3 pm	4 pm	5 pm	6 pm
Screenshot				

After improving the simulation of daylight, another problem is the lack of light inside the car. Due to the occlusion from the outside of the car, the light can only enter the car through the windows, which causes the shadows in the car. These shadows caused most of the interior to be invisible. In order to reduce the shadows while not destroying the light environment outside the car, we add two point lights inside the car to light up the environment and turn off the shadow properties of the two point lights to maintain the uniformity of the shadow direction.

At the end of this section, we define the minimum and maximum exposure values of the car configurator through post-process. To achieve this function, we first set the exposure mode to Auto Exposure Histogram in post-process. Auto Exposure is also called Eye Adaptation in UE4. This is to imitate the human eye adapting to different lighting conditions, in order to be able to see the surrounding situation clearly [13]. Therefore, the use of Auto Exposure Histogram is very suitable for us to solve the problem of dark interior in the car configurator. We set the value of Min EV100 to -10 and the value of Max EV100 to 20. When the brightness of the graphics of the car configurator is lower than the minimum value of the exposure, the Auto Exposure will increase the brightness of the graphics. In the case of overexposure, Auto Exposure will also control the brightness within the maximum range and automatically reduce the brightness of the graphics. Table 7 shows the improvement of the lighting environment and the appearance changes in UE4.

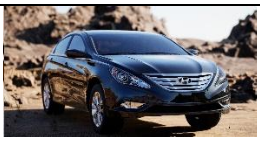
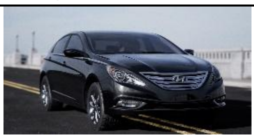
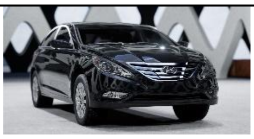
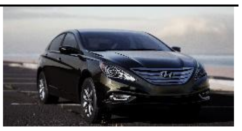
Table 7. Improvement of the lighting environment

Version of Lighting Environment	Previous Version	Upgraded Version
Daylight Source	HDRI	Sun Position Calculator
Exposure Mode	Manual Exposure	Auto Exposure Histogram
Exterior		
Interior		

3.2.2 Enrichment of display scenes

The display scene of the car is very important for showing the appearance of the car because the scene can provide the car with a series of information including light, reflection and atmosphere. In this paper, we have created four scenes which are desert, bridge, beach and showroom. There are two reasons for choosing these scenes. The first reason is that each scene has a completely different atmosphere, which can provide different styles of display performance for the car. The second reason is to prepare scene data in advance for the multi-scenes switching function in the next chapter. In the process of creating the scene, we use Quixel Megascans to build most of the scene models. Since the 3D assets of Quixel Megascans are all from photogrammetry, the realism of the model is very high, and due to the in-depth cooperation between UE4 and Quixel, 3D assets can be easily imported into UE4 through Megascan Plugin [14]. After completing the creation of the scene model, we also set different styles of lighting environment for each scene. Table 8 shows the final appearance of the four scenes.

Table 8. Final appearance of four scenes in UE4

Display scenes	Desert scene	Bridge scene	Showroom scene	Beach scene
Screenshot				

4. The implementation of multi-scenes switching function in car configurator

This section contains two sections, the section is the implementation of multi-scenes switching in UE4, showing the workflow of developing the scene switching function in car configurator. The second section is the increase in the amount of data due to the addition of multiple scenes, which leads to the appearance of display lag. To solve this problem, we propose optimization solutions and finally verified the feasibility of the solution by comparing the FTP values.

4.1 The workflow of implementing multi-scenes switching function in UE4

The first step is to package the four scenes we created earlier in UE4 through Level function. By default, UE4 has a present level that covers all elements such as models, lights, and blueprints. To pack four different scenes, we create four sub-levels, corresponding to desert, bridge, beach and showroom, and then import the model and lighting environment of each scene into the corresponding sub-level, so that we can control the switch of the scenes through the sub-levels.

To implementing multi-scenes switching function, we set up four sets in the Event Graph, each set corresponds to a sub-level. We can select a single sub-level through the Get Load Level node, this sub-level will become visible through the Loop Open Level Stream node, and other unselected sub-levels will continue to remain invisible through the Loop Unload Level Stream node. Figure 7 shows the workflow of the multi-scenes switching function in the Event Graph.

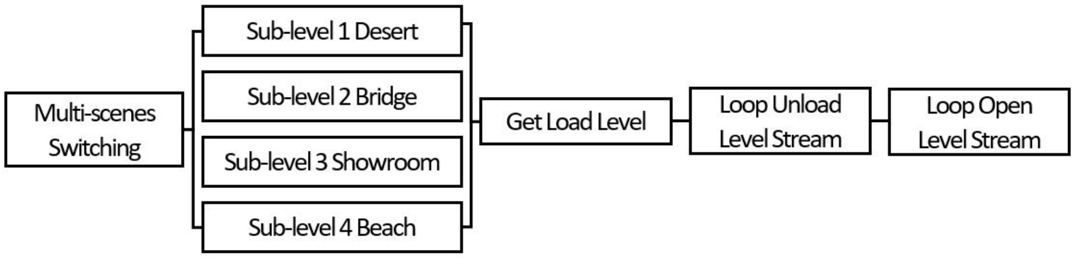


Figure 7. The workflow of the multi-scenes switching function in the Event Graph

After completing the Event Graph, we add a button to active the multi-scenes switching function in the car configurator's UI interface through UMG (Unreal Motion Graphic UI Designer). We set the trigger condition of the button to the mouse click action, and the event triggered by clicking the mouse is connected to the Event Graph of multi-scenes switching function, so that we can select different display scenes by clicking the switch button through the car configurator's UI interface.

4.2 Optimization of multi-scenes switching function

Although the multi-scenes switching function brings rich visual performance, it also leads to the appearance of display lag due to the increase of scenes. To solve the display lag problem, we propose an optimization solution for reducing the amount of computation. As mentioned above, we use many 3D assets from Megascans to build the scenes. Since most of the 3D assets of Megascans made by photogrammetry, this leads to a very high number of polycounts in every scene. We take a 3D dune model in desert scene as a test subject. The polycount of dune model is 36036 triangles. we reduce the polycount to 3603 triangles. The FPS (Frames Per Second) values are used to measure the changes in the smoothness of the display [15]. Due to the existence of texture maps, there is no obvious difference in appearance between different polycount dune models. Only in close-up shots we can see the difference in appearance between two models. But the increase of FPS value is obvious, the FPS value has increased from 24.52 to 30.14. The test result proves that reducing the model polycount has a direct effect on improving the smoothness of the display in UE4.

According to the results of the test, we reduce the polycount of all the background models in desert scene. The real-time FPS value of UE4 also verifies the effectiveness of LOD in improving the smoothness of the graphics. Table 9 shows the FPS value change of the desert scene after optimization.

Table 9. The change of FPS value after optimization

Version	Previous Version	Optimized Version
FPS Value	23.25	51.17
Polycounts	9,861,327 triangles	91,205 triangles
Screenshot		

5. Conclusion and future research

In this paper, we propose improvements to the realism of the car configurator graphics to add a new multi-scene switching function. In the part of improving the realism of graphics, we have improved the shader and display environment. In the upgrade of the shader, we have redesigned the car paint and glass shader. Compared with the previous version of the shaders, we have added more attributes to the new shaders. For example, we have added the three attributes of clear coat, flakes and orange peel to the car paint shader, which makes the structure of the car paint close to reality and also makes the reflection of the car paint surface more realistic. Similarly, we also add IOR and Fresnel attributes to the glass shader. IOR is used to define the refraction effect of glass, and Fresnel can simulate the color gradient of car glass.

In the improvement of the display environment, we replace HDRI with Sun Position Calculator Plugin, to simulate the sun's illumination at different times more realistically. To make the brightness of the lights inside and outside the car more harmonious, we use the maximum exposure value and minimum exposure value in Auto Exposure Histogram to control the brightness of the graphics, so that the graphics will not be overexposed or too dark. In the last part of this chapter, four display scenes are created. These scenes not only provide a rich display environment for the car, but also lay the foundation for the multi-scene switching function in the next chapter.

After completing the realism improvement of the graphics, we propose the workflow for implementing the multi-scenes switching function, which is not only a functional enrichment of the car configurator, but also an innovation of our paper. The multi-scene switching function developed by UE4 has not appeared in the market. Therefore, the workflow proposed in this paper has a very great reference value. We package the models and lighting environment of each scene through the Level function and control the switch of each scene by activating and deactivating the Level in the event graph. At the end, we connect the event graph to UMG (Unreal Motion Graphic UI Designer) to create a button for switching the environment in the UI interface of the car configurator.

Due to the addition of multiple scenes, the car configurator has a display lag problem. In order to ensure a smooth experience of multi-scenes switching, we propose an optimization solution by reducing the polycount of the environment models through the LOD (Level of details} method. We select a 3D dune model in the desert scene for comparison test, through test results we find that reducing the polycount can indeed increase the FPS (Frame Per Second) value in UE4, and the increasing of FPS value means the improvement of the smoothness of the graphics.

According to the research results of this paper, the research on the car configurator is mainly in the graphic and functional sections, and there is not much research on how to display the car configurator made by UE4 on other platforms. In future research, we will study the possibility of multi-platform display, package the car configurator through more media forms, and combine emerging technologies such as AR and MR to expand the application scenarios of the car configurator.

References

[1] Epic Games, "Product Configurator: How to customize and use the Product Configurator template", Unreal Engine Documentation, https://docs.unrealengine.com. 2019.

[2] Jimmy Thang, "Assetto Corsa Competizione leans into realism to create the ultimate racing sim", Unreal Engine Developer Interviews, https://unrealengine.com/, 2019.

[3] Volker Liedtke, "New Ferrari Car Configurator from Mackevision is based on Real-Time Game Engine", Linkedin, https://linkedin.com, 2018.

[4] Edward Martin, Michael Chun, Laurent Ruel, Randal Cumming, Oliver Schnabel, "Introducing Unity Forma: Reimagine marketing with real-time 3D", Unity Blog, https://blog.unity.com/manufacturing, 2020.

[5] Epic Games, "Automotive Configurator sample available now". Unreal Engine News, https://unrealengine.com/blog. 2021.

[6] Yiming Zhong, Tae Soo Yun, Byung Chun Lee, "The Implementation of Car Configurator Based on Unreal Engine 4". Korea Multimedia Society (KMMS) Summer Academic Conference, 2021.

[7] Kimio Toda, Abraham Salazar, Kozo Saito, "Automotive Painting Technology: A Monozukuri-Hitozukuri Perspective", pp.40-43, 2013.
DOI: https://doi.org/10.1007/978-94-007-5095-1

[8] Martin Rump, Gero Müller, Ralf Sarlette, Dirk Koch & Reinhard Klein. Photo-realistic Rendering of Metallic Car Paint from Image-Based Measurements. EUROGRAPHICS, pp.1-5, 2008.
DOI: https://doi.org/10.1111/j.1467-8659.2008.01150.x

[9] BESA LAB. Causes of the orange peel effect in paint. https://lab.bernardoecenarro.com. 2015.

[10] Epic Games, "Using Refraction Guide for using Refraction in your Materials", Unreal Engine Documentation, https://docs.unrealengine.com, 2020.

[11] Jeff Russell, "Basic Theory of Physically Based Rendering", Marmoset, https://marmoset.co/posts, 2015.

[12] CESIUM, "Using a Geospatially Accurate Sun San. CESIUM FOR UNREAL", https://cesium.com/learn/, 2016.

[13] John Hable, "How Epic Games is handling auto exposure in 4.25", Epic Games, https://docs.unrealengine.com, 2020.

[14] Lincoln Hughes, B. A. Baker, "Using Megascans in UE4 Environment Production", https://80.lv/articles/, 2018.

[15] Kayla Dube. "Understanding and Optimizing Video Game Frame Rates", Lifewire, https://www.lifewire.co m/optimizing-video-game-frame-rates-811784, 2020