Research Article

Three-Dimensional Animation Space Design Based on Virtual Reality

Rongxue Zhang

Shenyang Institute of Engineering, Shenyang 110000, Liaoning, China

Correspondence should be addressed to Rongxue Zhang; zhangrx@sie.edu.cn

Received 22 May 2022; Revised 25 June 2022; Accepted 28 June 2022; Published 1 August 2022

Academic Editor: Lianhui Li

Copyright © 2022 Rongxue Zhang. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

3D animation stereo space design is to process and analyze the collected target image information on the computer and finally obtain a 3D model that can represent the corresponding structure. As a branch of computer vision, 3D animation stereo space design is an important part of realizing the key technology of environment perception. What humans see in the three-dimensional world is a two-dimensional picture. Combined with the visual mechanism and the prior information of the object, it can realize the perception of the environment and promote the convenience of life and production. Games, movies, virtual humans, and any product that uses 3D technology require 3D engine support. The 3D engine encapsulates hardware operations and graphics algorithms and can manage a large number of texture and model resources to construct complex 3D scenes. In order to facilitate the research related to virtual human, in this paper, through the research and study of graphics-related mathematical knowledge and related technologies of engine architecture, an intelligent virtual animation generation engine based on DirectX12 is realized, and three-dimensional characters are controlled through related components, and the function of each module is tested and guaranteed. Test cases were designed to test the various functions of the engine and finally showed that the design goals in the demand analysis have been completed, and a lightweight three-dimensional animation three-dimensional space model has been realized.

1. Introduction

The emergence of digital technology has widened the expression space of animation art with its own advantages. It has brought new opportunities to animation art and scene design. According to the current research situation of scene design, in the conception method of animation scene design, people have formed thinking methods such as establishing the overall modeling consciousness, grasping the theme, determining the tone, and exploring unique and appropriate modeling forms. In terms of the expression of space in the scene, there are some specific methods for the elements and classification of animation scene and the methods of shaping the scene, such as using the sense of gravity, strengthening the depth of field, and using character scheduling. In the traditional scene design, there is also a certain theoretical basis for the light and shadow modeling and color creation in the scene. Light and shadow play an important role in animation film scene design, such as shaping space, creating atmosphere, and creating suspense. Nowadays, in the animation scene design, the design and research of furnishings and props are also an important aspect. In the film, they play an important role in explaining the character’s identity, shaping the psychological space, depicting the task character, and setting off the task emotion. With the rapid development of the animation industry, the term “scene design” is gradually known to people, but from the previous development, film and television animation scene design is often ignored in the animation industry and animation theory circle. From the perspective of professional training and theoretical research, it is basically in its infancy. For digital scene design, in terms of the research on the characteristics and performance methods of scene design, there is still room for further research. There are only a few books on scene design in the market, which often elaborate scene design from the perspective of two-dimensional animation...
or basic theory, without involving some characteristics of digital level. This paper attempts to build a lightweight 3D rendering platform that can quickly combine some intelligent algorithms with character control through the intelligent control framework to realize the interactive 3D animation space based on the natural user interaction interface [1–9].

2. Related Works

Since the 1970s, foreign countries have carried out a lot of research on graphics and engines. After half a century of development, a number of very excellent 3D software companies have been born. They include Discreet (later merged by Autodesk), Epic Games, Unity Technologies, Id Software, etc. These companies have played a great role in the development of computer graphics and the promotion of 3D engines and have produced rendering engines such as 3ds Max, Maya, Unreal, Unity, and Blender. These engines are widely used to this day. Later, more special 3D engines were born. Zbrush developed by Pixologic is a 3D engine for sculpting modeling. Houdini developed by Side Effects Software focuses on special effects design and terrain editing. Substance Painter and Substance Designer developed by Allegorithmic are mainly used for texture material design. Marvelous Designer developed by Foundry Video Effects Company realizes clothing tailoring and cloth simulation, and Daz 3D produced by Daz Production Company can edit high-quality character models. The current hot direction in the generation of 3D human animation is to reduce the cost and increase the restoration degree of animation. Abroad, Ryo et al. proposed a method to create a spatial 3D model using the Kinect sensor. The 3D information of the space is obtained from the Kinect sensor, and the 3D model is created by synthesizing the images and the obtained 3D information. Kunlin et al. parameterized the standard model according to anthropometric parameters. Then, the Kinect depth map of the human body model was optimized by processing and matching the point cloud data by using the PCL library. Quick integration with the PCL library yields a realistic human model. A new method for 3D human modeling based on a single Kinect was obtained by using an iterative closest point algorithm to register the captured upper human 3D point cloud data with standard reference human data. Zhang et al. proposed a highly personalized human modeling analysis method based on a single Kinect. First, a high-precision human positioning point cloud based on a single Kinect is obtained, and then, on the basis of ensuring the accuracy of the head of the point cloud, the point cloud information is preprocessed. Finally, by using hierarchical compactness to support radial basis functions (CS-RBFs), a 3D human body model is obtained by fitting the sampled point cloud to the existing human body. At present, there are few related papers on DirectX12 and Vulkan. Some articles have a clear description of the modules of the 3D engine. Compared with the above research, this research is more inclined to extend from the 3D animation design, from the basic data structure design to the implementation of related technologies [10–19].

3. Key Technologies of 3D Animation

Stereo Space

3.1. Scenario Management Technology. 3D characters and their environments can be collectively referred to as 3D scenes. The 3D engine needs to fully describe the scene and each object in it in order to integrate various model data and present them on the computer screen. According to different application requirements, the scene management technology in the engine can be divided into two categories:

1. Scene management based on space division: this technology includes a variety of space conversion and division algorithms, including quadtree, octree, and Bsp tree. The core is to establish a hierarchical structure according to the spatial distribution relationship of objects, and the nodes in the tree represent an area of space where objects are stored on leaf nodes. This type of scene management mainly uses the tree structure to quickly locate scene objects, optimize collision detection, and view frustum culling [20].

2. Scene management based on parent-child relationship: this management technology is also called “scene graph.” The core is to establish a hierarchical tree structure based on the parent-child affiliation of objects. The nodes in the tree are objects with local transformation attributes, and the world transformation is obtained by passing the local transformation between the parent and the child. This type of scene management mainly records the transformation transfer direction through a tree structure, so as to perform the overall transformation of multi-component objects. In the space partitioning technology, nonleaf nodes represent a space region and are mainly used for the efficiency optimization of internal computing.

In the scene graph, nodes represent objects with local transformation properties, which is convenient for batch manipulation of scene objects. After many 3D software tests, scene graph technology is very suitable for developers to create and design 3D scenes. Since this article focuses on the display of 3D characters and their environments, this article will focus on the technical details of the latter.

3.2. Character Animation Technology. 3D animation is one of the core technologies of virtual characters. Excellent animation performance can not only increase the authenticity of the characters but also enrich the interactivity of the characters. Animation technology has undergone a long history of evolution, but currently the most widely used animation technologies are mainly in the following two categories:

1. Vertex animation and vertex animation texture technology (VAT): vertex animation directly records the spatial position of each vertex of the model mesh, which can simulate very complex animation
scenarios. This animation technology is often used to simulate some very real object systems such as cloth and fluid. However, in the early stage, due to the large storage space required for vertex animation and the difficulty of mixing and reusing, it was rarely used in real-time systems. With the development of GPU hardware, the vertex animation texture (VAT) technology was born. For an animation with \( n \) vertices and a time length of \( f \), the coordinates of the vertices at each moment are stored in a texture of size \( n \times f \): animation texture sampling is performed in the vertex shader, and the pose transformation of the vertices is performed. The animation texture technology is often used for thousands of identical animated objects in the picture. It can greatly improve the efficiency of animation operation [21].

(2) Skeletal animation consists of two parts of data: the skinned mesh and the skeleton. The skinned mesh records the index of the bones affecting the vertices, and the skeleton records the parent-child relationship between the bones, as well as the local coordinate transformation curve of each bone. Through the local coordinate system transfer technology in the scene graph, the bone pose at a specific moment is calculated, and then the bone animation is mixed according to the bone index recorded by the vertices of the skinned mesh to obtain the final vertex coordinates, specifically as shown in Figure 1.

In the binding pose, set the palm bone world transformation matrix to WorldMat\(^\text{Bind}_\text{Bone}\) and the thumb mesh world transformation to WorldMat\(^\text{Bind}_\text{Finger}\): the offset matrix formula is as follows:

\[
\text{WorldMat}^{\text{Bind}_\text{Bone}} = \text{WorldMat}^{\text{Bind}_\text{Finger}} \text{WorldMat}^{\left(\text{Bone}_\text{Bone}\right)}
\]

(1)

The original coordinates of the mesh vertices are multiplied by the offset matrix to obtain the mesh vertex coordinates under the bone coordinate system and then multiplied by the skeletal animation matrix at a certain time to obtain the mesh vertex coordinates under the world coordinate system. The formula is as follows:

\[
\text{WorldMat}^{\text{Anim}_\text{Finger}} = \text{OffsetMat}^{\text{Finger}_\text{Bone}} \text{WorldMat}^{\text{Anim}_\text{Bone}}
\]

(2)

3.4. Interactive Technology. Interactive intelligent characters are often able to respond to external input, as shown in Figure 2.

The technology in which machines process external input and feedback response is called human-computer interaction. Human-computer interaction technology can be divided into the following three categories according to the development period:

(1) Input interaction based on command line interface (CLI): command line input interaction is the earliest human-computer interaction technology. The user inputs instructions through the keyboard, and the computer processes and feeds the results back.

(2) Input interaction based on graphical interface (GUI): GUI is the most common interaction mode today, and most 3D games and interactive videos use graphic interaction technology.

(3) The input interaction based on natural user (NUI) is more natural and humanized than the previous two interaction methods. The user can control or interact with the virtual character through body movements and voice information. Restricted by hardware devices, in this way, the interaction between virtual characters and real users is more natural and harmonious. as shown in Figure 2.

4. Design of 3D Animation Three-Dimensional Space Model Based on Virtual Reality

4.1. Component System Design. The core of the traditional component system is as follows:
(1) Each component inherits from a component base class and implements the Update function in the base class.

(2) There is a component list inside the entity class, which directly stores the base class component pointer. If the entity of the traditional component system is accessed through the System, the System will traverse the component list in the entity to determine whether it has all the required components, as shown in Figure 3.

In this working mode, the component data pointed to by the pointer is discrete. When the System operates a large number of entities, the Cache hit rate is not high, causing frequent page replacement and even page “jitter.” In order to store component data in contiguous space as much as possible, all components of all entities are placed in a contiguous memory block. In this version, the data between entities is discrete, which will reduce the Cache hit efficiency when operating a large number of entities. Since the separation of data and operation functions is achieved, the data should be stored continuously, as shown in Figure 3.

As shown in Figure 4, the same components of all entities are stored together, but, for example, Entity2 does not contain the HealthCmpt component, while Entity1 and Entity3 do not contain the AttackCmpt component. When all the entity components are stored together, there will be gaps, making the entity and the corresponding component data. Location relationships are not easy to manage. Therefore, a better method is to put the component data of entities with “similar components” in an Archetype, as shown in Figure 4. Entity is only used as an index pointing to the Archetype to indicate the location of the data contained in the entity in memory. There is a memory pool implementation in Archetype, and each block of memory is divided into a fixed size, which is convenient for memory alignment and improves the efficiency of continuous data reading, specifically as shown in Figure 5.

Component addition or deletion is an important function of ECS. The operation process design is shown in Figure 6.

4.2. Local Coordinate System Transfer Design. Many 3D animation engines, including 3 ds Max, Unity, and Unreal, build scenes through SceneGraph. SceneGraph is a multifork tree that records the parent-child relationship between scene nodes. In the engine, the transform of the parent node property changes that affect child nodes is called parent-child pose transfer. The effect of the translation operation on subsequent nodes is very intuitive. Only the translation transformation of the parent node is directly applied to the child nodes, as shown in Figure 7.

But when it comes to the rotation and scaling of the parent and child classes, the child class will appear in an unbelievable shape, as shown in Figure 8.

It can be seen that the scaling of the parent class is not directly used for the subclass objects but produces “nonequal
Figure 4: Contiguous storage of components.

Figure 5: Centralized storage of entity with the same components.

Figure 6: Component deletion process.
scaling with angles” according to the rotation of the subclass. Unequal scaling with an angle can be used as an example with a cube, as shown in the figure below. Double scaling with an angle of 45 degrees will result in what is shown in Figure 9.

Unequal scaling with angles can be split into two rotations and one scaling. For example, double scaling with a 45-degree angle is equivalent to the following: first rotate the coordinate system counterclockwise by 45 degrees to get the blue color shown in Figures 5–13, extend the new coordinate system to double the Y-axis direction, and then rotate the coordinate system 45 degrees clockwise to return to the original black coordinate system.

In order to store and load the scene, the basic data needs to be serialized and deserialized. Serialization and deserialization mean that all types of data are uniformly converted into binary and then stored in binary form or transmitted over the network. Relatively, when the file is read and the network is received, it is written according to the binary. The order of input is restored to the corresponding data structure.

A large amount of continuous data of the same type such as grids and animations can be read and written in binary using traditional data structures such as Vector. But sometimes it is necessary to record not only the value of the data, but also the dependencies between the data. The SceneGraph that represents the scene structure is this kind of data with dependencies, including the parent-child relationship between scene nodes and the mount relationship between entity and component. This kind of dependency obviously cannot be recorded with traditional data structures.

The data structure relationship designed in this paper is shown in Figure 10. This kind of inclusion relationship is very suitable for storing with JSON. JSON is a typical data structure that can be nested. It is very suitable for recording complex hierarchical data relationships. Many games in the market use JSON to store complex character attributes and level progress. Similarly, JSON can also be used to store the hierarchical relationship of SceneGraph. Another advantage of the JSON data structure is that it is convenient to add properties to JsonNode in the form of Key-Value, which is just suitable for storing resource indexes in material components and mesh components. There are many C++ JSON open-source libraries on GitHub. This article will use the cJSON open-source library to JSONize the scene structure. JSON data will change size according to the depth of SceneGraph, and it is also dynamic data, similar to Vector and String formats. When serializing JSON data, size is also required for reading and writing.

Through the serialization of static data and dynamic data, this study designs the following binary file structure. By storing and reading the binary file of this structure, the storage and restoration of 3D rendering scenes are basically realized, as shown in Figure 11.

4.3. Graphic Rendering Design

4.3.1. GBuffer Design. The first texture of the GBuffer can be used to record the normal information of the object. The normal may be the built-in normal of the object vertex, or the sampled normal calculated by the normal map and the surface tangent. The three-dimensional normal vector can be obtained through the text. The mentioned “normal compression formula” compresses the normal information into two dimensions. The second texture of the GBuffer can be used for the diffuse reflection color of the object. The diffuse reflection color will be obtained by sampling the diffuse reflection texture of the object material in the shader of the GBuffer. The first three channels of the four channels of RGBA are occupied, and the remaining one channel can be used to store other data. The third texture of GBuffer can be used to store special rendering information, such as Roughness (Roughness), Metalness (Metal), and masking (Ao) that may be statically baked for PBR material rendering. The last channel is used to store the RenderType, and the subsequent lighting processing stage will decide which lighting model to call for rendering based on the RenderType of the vertex. Since an object is basically not rendered by two different lighting models at the same time, the meaning of each channel will change with the RenderType. For example, when performing NPR rendering, it may be written into the MetalRoughAo texture, Emmisive, Shadow, equal coefficient. The sizes of the Gbuffer parts are as follows, totaling 80 bytes, as shown in Table 1.
Figure 9: Unequal scaling with angles.

Figure 10: Inclusion relationship of scene data.

Figure 11: Data structure of binary archive.
4.3.2. Construction of Deferred Rendering Pipeline. First, we set the rendering target as a screen-sized texture, judge the lighting rendering model to be called according to the A channel data RenderType stored in the GBuffer, read other required parameters from the GBuffer, and perform lighting calculations. Subsequent forward rendering image post-processing can be performed to process some objects with transparency, but obviously, transparent objects cannot be rendered between front objects, which is even the disadvantage of delayed rendering [22].

4.4. Animation System Design. VideoPose3d is an open-source video input 3D pose estimation library, which is different from using the depth camera Kinect for 3D pose estimation. The library only needs the video data provided by the nondepth camera and processes the input video through a neural network to obtain coordinate estimation of 3D joint points. This chapter attempts to combine this neural network with 3D character animation, so that 3D characters have the interactive ability of action simulation.

The VideoPose3d model was presented as a paper at Facebook’s 2019 Conference on Computer Vision in Pattern Recognition (CVPR). It has been open sourced on GitHub. Different from the traditional learning model based on joint point detection, this model uses the two-dimensional key points of the video frame as the input stream to perform time series convolution, so as to train to recognize the three-dimensional pose in the video, validated on the Human36m dataset with good results. Another reason for choosing VideoPose3d is because the model provides a semi-supervised learning method for unlabeled video, by predicting the 2D key points of the unlabeled video, then estimating the 3D pose, and projecting the 3D pose to the 2D pose in reverse by passing in the camera parameters.

5. Analysis of Experimental Results

5.1. Test Environment and Tools. This article uses Windows 10 as the test operating system and uses Microsoft’s C++ development environment Visual Studio 2019 for engine development. Vs 2019 provides a wealth of extension plug-ins, and you can download the HLSL syntax highlighting tool to improve the efficiency of Shader programming. In order to use the DirectX12 graphics development interface, the DirectX12 SDK needs to be installed, but most of the current Windows 10 SDK integrates the DirectX12 development environment. If the operating system includes it, you can write code directly without additional configuration. Part of the interface is written by Qt5.12.2. Qt’s .ui file needs to be precompiled by Qt’s built-in compiler, so the computer must install the corresponding version of Qt. The entire project is compiled and built through CMake, which is convenient to adapt to different versions of the Vs development environment.

5.2. 3D Animation Stereo Space Test. Take the FBX file below as a test example and refer to the running action around frame 180. Next, we will import the FBX into the engine for testing.
The running action of about 180 frames imports the FBX into the engine and sets the material to paste the texture on the model. The effect is as shown in Figure 13.

Comparing Figures 13 and 14, it is shown that the engine basically uses the material function to restore the character image. Next, adjust the time axis of the Cmpt_SkinRoot component of the color correction root node to around 180 frames, and the following effects can be obtained.

It can be seen from Figure 15 that the FBX animation data analysis of the animation system in this study shows that the transformation of the local coordinate system of the bones is normal, and the simultaneous animation effect in 3ds Max is restored. The end result is a demonstration of skeletal skinning techniques and a reproduction of the same animation system as 3dx Max’s internal poses.

6. Conclusion

The advent of the digital age has had varying degrees of impact on all aspects of our lives. It has also had a great impact on art and even animation scene design. From the many breathtaking special effects in TV films and digital 3D animations that rely entirely on digital media, the powerful influence of digital technology is vividly displayed. The core purpose of this paper is to build a 3D character display platform based on DirectX12. In order to realize the display of 3D characters, the basic data required for 3D character display is firstly displayed, and then the related programming of DirectX12 is carried out around the acquisition and conversion of these data. In the development of the engine, the functions and architectures of many existing engines and open-source engines were referenced to design the various modules of the engine.

Data Availability

The dataset can be accessed upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.
References

[1] M. Speicher, S. Cucerca, and A. Krüger, “VRShop,” Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, vol. 1, no. 3, pp. 1–31, 2017.

[2] T. Raffailiac and S. Huot, “Polyphony: programming interfaces and interactions with the Entity-component-system model,” in Proceedings of the ACM on Human-Computer Interaction, HAL, Valencia, Spain, June 2019.

[3] H. C. Park and N. Baek, “Design of selfengine: a lightweight game engine,” InformationScience and Applications, Springer, New York, NY, USA, 2019.

[4] A. Razzaq, Z. Wu, M. Zhou, S. Ali, and K. Iqbal, “Automatic conversion of human mesh into skeleton animation by using Kinect motion,” International Journal of Computer Theory and Engineering, vol. 7, no. 6, pp. 482–488, 2015.

[5] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black, “Keep it SMPL: automatic estimation of 3D human pose and shape from a single image,” in Proceedings of the European Conference Computer Vision – ECCV, pp. 561–578, Springer, Amsterdam, The Netherlands, October 2016.

[6] M. R. I. Hossain and J. J. Little, “Exploiting temporal information for 3d pose estimation,” in Proceedings of the European Conference on Computer Vision (ECCV), Springer, Germany, UK, August 2018.

[7] S. Park and N. Baek, “A shader-based ray tracing engine,” Applied Sciences, vol. 11, no. 7, p. 3264, 2021.

[8] R. A. Pavlík, Enabling Natural Interaction for Virtual Reality, Iowa State University, Ames, US, 2014.

[9] J. Tompson, R. Goroshin, A. Jain, Y. Lecun, and C. Bregler, “Efficient object localization using ConvolutionalNetworks,” in Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Boston, MA, USA, June 7-12 2015.

[10] F. Kistler, B. Endrass, I. Damian, C. T. Dang, and E. André, “Natural interaction with culturally adaptive virtual characters,” Journal on Multimodal User Interfaces, vol. 6, no. 1-2, pp. 39–47, 2012.

[11] P. Varcholík, Real-Time 3D Rendering with DirectX and HLSL, Addison-WesleyProfessional, Boston, MA, USA, 2014.

[12] W. Culbertson, 3ds Max Basics for Modeling Video game Assets:Volume 2: Model, Rig and Animate Characters for Export to Unity or Other Game Engines, CRCPress, Boca Raton, Fl, USA, 2021.

[13] C. Rudolph, A Framework for Example-Based Synthesis of Materials for Physically Based Rendering, Technische Universität chemintz, Chemnitz, Germany, 2019.

[14] T. Pfister, J. Charles, and A. Zisserman, “Flowing ConvNets for Human Pose Estimation in Videos,” Computer Vision and Pattern Recognition, vol. V2, Article ID 02897, 2016.

[15] D. Pavllo, C. Feichtenhofer, D. Grangier, and A. Michale, “3D human pose estimation in video withtemporal convolutions and semi-supervised training,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), CVF, Long Beach, CA, USA, June 2019.

[16] M. D. McIlroy, “Mass produced software components,” in Proceedings of the NATO Conference in Software Engineering, Scientific Research, Rome, Italy, October 1969.

[17] J. Gubata, Process Design Requirements -- Research Starters Business, vol. 16, no. 11-12, pp. 1–13, 2008.

[18] S. Corsaro and J. Clifford, Parrott. Hollywood 2D Digital Animation: The New Flash Production Revolution, Course Technology PTR, Boston United States, 2004.