Establishment of virtual trauma soldier model

Junjie Zhu*, Dongfeng Liu

Guangdong University of Technology, Guangzhou, China

*2112203082@mail2.gdut.edu.cn, liudf@gdut.edu.cn

Abstract. With the development of modern military science and technology, modern war has entered the age of information warfare. Medical service exercise simulation battlefield rescue that is to use computer simulation battlefield rescue scene has been an important way to improve the rescue ability of ambulance personnel. At present, there are many medical service exercise simulation battlefield built by different software. The contribution of this paper is that we will restore the injured condition of soldiers quickly on the battlefield of modern war in virtual system and build the virtual trauma soldier model. This paper mainly realized various blood stains in soldiers' severed limbs, gunshot wounds, burns, knife wounds and various combat wounds, and explained the principle of realization, and showed their effects in Unity3D.

Keywords: several injured conditions, virtual trauma soldier model, medical service exercise simulation battlefield

1 Introduction

Military medical [1] support, crucial for maintaining combat effectiveness, relies heavily on medical personnel. Given the volatile international landscape with recurring regional conflicts, preparedness for military engagements remains paramount. Modern warfare necessitates diverse medical responses [2][3] due to varied injuries, including multiple sites, extensive trauma, and complex post-injury complications, posing significant challenges in casualty care. Enhancing frontline medical capabilities through simulated battlefield rescue exercises is pivotal. Utilizing computer simulations offers advantages over traditional methods, notably cost and resource efficiency [4], while providing realistic scenarios unaffected by physical constraints, thus optimizing training efficacy. Simulated casualties enable meticulous assessment and treatment, enhancing first responders' skills and ultimately improving survival rates [5]. Given the pivotal role of frontline rescuers, comprehensive battlefield medical training is imperative for all soldiers, aiming to enhance casualty survival and minimize fatalities during search and rescue operations, offering a scientific approach for medical protocol development [6][7][8][9].

In this paper, through the Unity3D engine model and C# programming, we simulate the injury process of soldiers through the two links of injury state and injury type, collect relevant data information according to the structure of the injury model, and

Y. Feng et al. (eds.), *Proceedings of the 4th International Conference on Internet, Education and Information Technology (IEIT 2024)*, Atlantis Highlights in Social Sciences, Education and Humanities 26,
https://doi.org/10.2991/978-94-6463-574-4_18

establish the corresponding injury animation and model. In the virtual system, we can see how the injury occurs, what happens after the injury, and clearly distinguish the type of injury. This paper focuses on amputations, gunshot wounds, burns, knife wounds, and various blood stains in various combat injuries of soldiers, and shows how these affect the soldier model in Unity3D.

2 Parameters of Soldier Model and Injury Types Theory

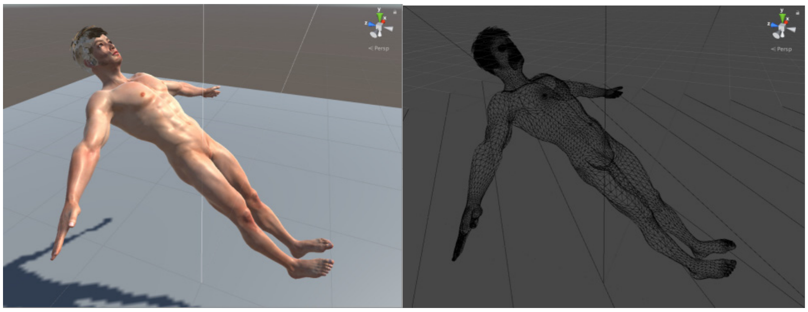

Fig. 1. Normal character model display(left). Shading Mode: Wireframe in Scene view

For the virtual soldier model, which is normally presented in our view as shown in Fig. 1, the concept we need to understand is the SkinnedMesh of the soldier model in Unity3D, as the operations in this paper are basically performed on the mesh of the model. Fig. 1 left shows the skin rendered on the frame based on Figure 1 right. In Unity3D, understanding the concept of SkinnedMesh for soldier models is crucial. SkinnedMeshes are meshes where vertices are associated with bones, mimicking real-life skeletal structures. Each vertex's position change is influenced by one or more bones with weighted configurations. Like real muscles and bones, multiple bones affect each muscle, with some muscles influenced by only one bone. This process of connecting vertices to bones is called skinning.

Various effects related to war injuries are achieved in a simulated battlefield environment through mesh processing. These effects include amputations caused by shells or other weapons, burns caused by artillery fire, gunshot wounds, stab wounds caused by close-range combat, and blood distribution. In order to enhance the realism of these war injuries, this paper also uses PS software to process textures and apply them to the clothes of the character model. This allows for the representation of both body and clothing damage. In conclusion, these techniques are considered suitable for high precision and multi-material models that provide realistic depictions of war-related injuries.

3 Injury Types Implementation Scheme

In the second part of this paper, an overview of the various damage types was given. In this section, we describe in detail the model mesh vertex separation techniques and texture properties as well as texture blending techniques. These techniques are implemented using C# programming algorithms.

3.1 Model Mesh Vertex Separation Techniques

This technique is primarily used to realize cutting amputation injuries in soldier limb injuries, i.e. dynamic limb separation displays. This type of injury is characterized by its smooth and neat cross-section, so the injury is characteristically easier to realize. Next, we talk about the model mesh vertex separation technique.

The technique is divided into two steps: the display of limb separation and the mending of the empty part of the torso and the amputated limb.

To display limb separation, we exploit the triangular nature of models, where triangles are formed by connecting vertices. By selecting specific positions within the entire triangle set of the original soldier model, we partition it into two sets: severed limbs and residual parts, assigning different materials. This process visually separates and displays the limbs of the original model. Illustrated in Fig. 2 using a cube example, a cube comprises eight vertices and six faces, each face composed of two triangles. Dividing the square entails partitioning the original vertices and triangles into two groups, with the shape of each group determined by its respective vertices and triangles.

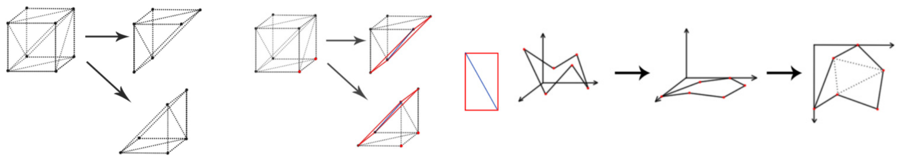

Fig. 2. The left Figure shows an example of virtual cube model segmentation. Maximum section loopback (red line), red dots are the marked vertices of the separated sections. The right Figure shows Triangle Generation: Ear Clipping Algorithm

The stitching of the hollow section of the torso and the severed limb: this part consists of two parts: the search for the maximum ring of the section and the complement of the triangular surface according to the circle of the search. Then talk about these two parts:

1. Search for the maximum cross-section loop: we first locate intersection lines within the models representing the severed limbs and the torso, as depicted in Fig. 3 by the red loops formed after separating the cube into two prisms. Intersection lines are formed by overlapping edges of two triangles composing the model. Vertices of triangles on either side of these intersection lines are recorded differently. The vertices belonging to the part to be separated are marked with red dots in Fig. 3. Traversing the model to identify and connect these segments yields the largest cross-section loop. However, for complex models like the human body, a challenge arises due to numerous unrelated yet duplicated vertices, potentially leading to errors in finding the largest loop and resulting in strange deformations. This issue is

addressed by traversing the model to identify and correct or remove these duplicate vertices.

2. Make a triangular surface according to the maximum loopback: Generating a triangular surface from the largest cross-section loop involves solving a triangulation problem. Various algorithms exist for this purpose, such as Delaunay triangulation, point-by-point insertion (Lawson algorithm), Bowyer-Watson algorithm, and Ear Clipping algorithm. Due to the potential increase in memory consumption caused by inserting more vertices, this paper opts for the Ear Clipping algorithm, which is fast and does not generate additional vertices, thus reducing computational overhead. This algorithm achieves triangulation by iteratively removing "ears" from a polygon until only triangles remain. The approach in this paper involves connecting adjacent vertices in the largest loop obtained from the cross-section to form triangles. It's crucial to ensure no holes are present in the formed triangles during this process. As these triangulation algorithms operate in a 2D plane, the spatial vertices of the soldier model are mapped onto a 2D plane before triangulation, ensuring the correct process, as depicted in Fig. 2(right).

After processing with grid vertex separation technology, most of the content of amputation was realized, as shown in Fig. 3 From the Figure you can clearly see the shape of several SkinMesh parts of the body and the severed limbs after the Figure of the model treated after the person suffered a cut amputation injury. The image can also clearly generate a plane of many triangles on the section of the severed limb. From the image, you can also see that the sections of several SkinMesh sections after disconnecting have been filled with triangles without holes.

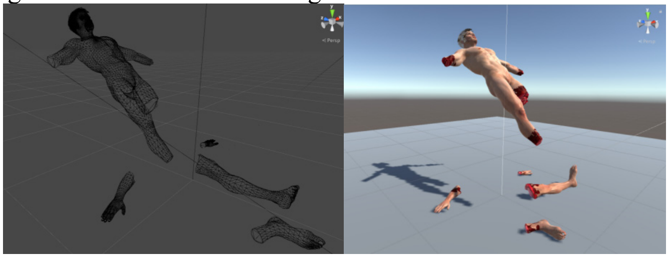

Fig. 3. Triangular surface modelling diagram of cutting amputation injuries(left). Final rendering of a model of a cutting amputation injury (right)

3.2 Texture Properties and Texture Blending Techniques

In this technique, we discuss the concept of textures, which define the appearance of objects in Unity3D, allowing us to visualize what models look like, while meshes describe their shape. Textures, as standard bitmap images applied to mesh surfaces, provide most of the intricate details and are integral to materials. Materials and meshes are complementary in modeling; neither can construct a model fully without the other. The right side of Fig. 1 depicts a mesh, while the left side illustrates a human model with appropriate materials applied to the mesh. In Fig. 3(left), vertex separation techniques are applied to the model, separating multiple meshes from a single pair of meshes to generate severed limbs. In Fig. 3 (right), after limb cutting, correct materials are assigned to each SubMesh. Additionally, bloodstain effects are added to the skin near the severed limbs, enhancing realism on top of the correct materials.

The next step is texture blending technology, which relies on the properties of textures. In many game scenarios, such as FPS games, adding details like bullet marks on walls or cuts on trees enhances realism. We decided to utilize this technology to simulate various wounds on characters and achieve clothing damage. The principle involves obtaining UV coordinates of a point on the soldier model and its corresponding texture maps, then processing pixels in the texture map region to replace original pixels with those representing different types of wounds. The result merges two images into one, achieving the desired effect.

In Unity3D, colors are described using RGBA format, where each pixel's color is represented by a four-dimensional vector (r, g, b, a). UV coordinates, on the other hand, are two-dimensional coordinates indicating the position of a pixel on a texture map. The RGB components (r, g, b) define a color in the RGB color space, while the alpha component (a) defines transparency: a value of 1 represents full opacity, while 0 represents full transparency. It's essential to set all injury and human body images to RGBA 32-bit format upon importing into Unity3D.

In our texture blending technique, we utilize the color properties of the (r, g, b) components to simulate various wounds on the human skin. UV coordinates are used to locate the position of skin wounds, ranging from (0,0) to (1,1), representing coordinates from the bottom left corner to the top right corner of the skin map. However, the resolution of injury textures used in the project varies based on the size of the wound. Therefore, UV coordinates need to be mapped to the actual coordinates of the texture.

Mapping the texture involves multiplying the x value of UV coordinates by the width of the texture resolution and the y value by the height. This mapping allows us to blend textures at the required positions. Subsequently, the pixel values of the blended skin image at blending positions are replaced with the corresponding pixel color values from the injured texture. The process is shown in Fig. 3. The body effect after texture blending is shown in the image to the right of Fig. 3.

When the Main Texture is manipulated, it alters the corresponding material of the shader, resulting in changes to the model's appearance. The application of Fig. 4 (left) is illustrated in Fig. 4 (right). However, a problem arises because our texture blending technique replaces pixel values of blended skin images with those from the injured texture, causing unnatural transitions at the edges of textures. To address this issue, we refine and optimize the algorithm by performing linear interpolation between the original skin pixel values and those from the injured texture along the edges. This enhancement resolves issues of prominent visual artifacts occurring when textures overlap.

Additionally, the transparency of clothing is manipulated using the alpha component (a) to achieve the effect of damage (Fig. 5). This involves processing the transparency of clothing, building upon the previous algorithm that dealt with RGB values. While RGB values were addressed earlier, here we focus on manipulating the alpha values. Since this part utilizes grayscale images, unlike the RGBA color images used for blood effects, it's necessary to grayscale and normalize some color images using Photoshop, or create and process grayscale images to meet the requirements.

For grayscale images, where RGB values are equal, grayscale value represents pixel intensity, normalized to range 0 to 1. In RGBA, 'a' component defines transparency (a=1 fully opaque, a=0 fully transparent). Manipulating 'a' value based on grayscale pixel intensity achieves clothing damage effect: black (fully transparent), white (fully opaque), and gray (partial transparency for edges). Adjusting grayscale image size controls damage range. These are partial effects; more variations shown later.

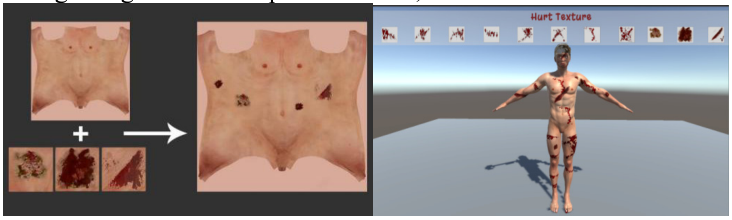

Fig. 4. Material map of texture fusion technique (left). Rendering of the model with texture fusion technique (right)

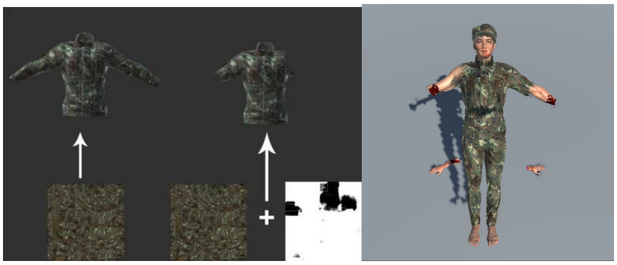

Fig. 5. The left picture shows the principle of clothing damage effect, and the right picture shows the effect of clothing damage combined with soldier injury.

The results of the soldier battle trauma modelling are shown in Fig. 6. As shown in the Figure below, the soldier model suffered a variety of injuries on his body, including amputation, gunshot wounds, burns, stab wounds and the distribution of various forms of blood mentioned at the beginning of the article.

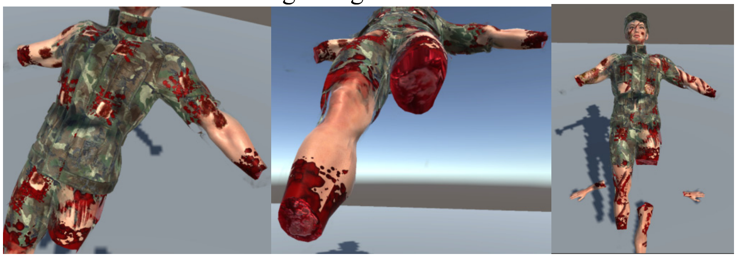

Fig. 6. Virtual Soldier Model War Damage Effect.

4 Conclusion

In this paper, we have achieved various battle injuries including amputation, gunshot wounds, burns, knife wounds and various blood stains in the soldier model through the joint effect of several algorithmic techniques of c# programming, and demonstrated the effects of these dynamic injuries on the soldier model in Unity3D. In the next work, we will continue to learn, research and innovate to add more other injuries, such as penetration injuries, impact injuries, extrusion injuries, etc., so that the virtual soldier battle trauma model can more comprehensively simulate the battlefield injuries to soldiers caused by enemy weapons directly in the battle, as well as the injuries to soldiers caused by the combat operations or the war environment. At present, the domestic battlefield simulation and rescue system is still in the exploration and research stage, and the soldier injury model is far from the foreign development level. This paper aims at dynamic modelling of soldier injury in battlefield simulation rescue system, and the related research content has strong practical value.

References

1. Xu, S.J., Liu, M.C., Hu, S.B., Fen, J.L., & Liu, X.W. (2022). Application and analysis of practical training of medical service based on AR, VR and MR Technology. South China Journal of Defense Medicine, 36(10), 805-808.
2. Bilal's, W. (2019). Virtual Trauma and Simulation: Cybernetic Performance in. Stories in Post-Human Cultures, 207.
3. Wang, J.R. (2018). Thinking on the construction of combat trauma rescue system to cope with the change of combat form and injury condition under the new military reform. Journal of management.
4. Chen, Q., Wang, S.W. (2011). Application of Unity-based virtual reality technology in education. Software guide(12), 76-78.
5. Xue, Q., Cao, B. W., Li, A. L., Yu, P. G., & Yao, Y. J. (2012). Research of Application of Bounded Rationality on Virtual Soldier Modeling. Applied Mechanics and Materials, 198, 746-750.
6. Wu, J., Li, Y., Liu, Q., Su, G., & Liu, K. (2017, June). Research on Application of Unity3D in Virtual Battlefield Environment. In 2017 2nd International Conference on Control, Automation and Artificial Intelligence (CAAI 2017) (pp. 465-468).
7. Gao, Z.H., Ou, Y.J., Chen, R.F., Zhu, M., Liu, P., Yang, Y. (2022). Investigation and assessment analysis of combat injury treatment training for Navy grassroots military doctors. Journal of naval medicine, 43(09), 918-920.
8. XSteven, L., Hauw, J. K., Keane, M. B., & Gunawan, A. A. S. (2023). Empowering military in tactical and warfare area with virtual reality technology: a systematic literature review. Procedia Computer Science, 227, 892-901.1.
9. Wang, D., Cao, S., Liu, X., Tang, T., Liu, H., Ran, L., ... & Niu, J. (2021). The virtual infantry soldier: integrating physical and cognitive digital human simulation in a street battle scenario. The Journal of Defense Modeling and Simulation, 18(4), 395-406.