Journal of Theoretical and Applied Information Technology
28th February 2019. Vol.97. No 4
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A STUDY ON REPRESENTING NATURAL SENSE OF ATTACK AND DESTROYING BODY-PART IN 3D GAME BASED ON UNREAL ENGINE 4

[1]CHANHYUN KIM, [2]YOUNGSIK KIM

[12]Dept. of Game and Multimedia Engineering, Korea Polytechnic University, Republic of Korea

E-mail: [1]acexr@naver.com, [2]kys@kpu.ac.kr (corresponding author)

ABSTRACT

The system component of the game called destruction has basically a feature that makes users feel more action and thrill. This feature can give players more realism and more immersion in the game. Especially, if we apply this system element to Virtual Reality (VR) technology which is emerging recently, we can simulate dangerous experiments that can not be done in reality in real time in other fields than the game. In this paper, we propose a mathematical method to produce a natural destructive fracture in First Person Shooter (FPS) games. Through the simulation using the Unreal Engine 4, the proposed method proved the efficiency of the physics and mathematical expressions derived from the colliding objects and the experiment in order to achieve optimal part destruction. By setting the frame rate of a 3D game based on Unreal Engine 4 between 30 and 60, we can simulate the partial destruction of a 3D game character while reproducing a natural sense of attack.

Keywords: *Natural Sense of Attack, Body-Part Destruction, 3D Game, Unreal Engine 4, Rendering Speed*

1. INTRODUCTION

As more and more people have enjoyed three dimensional (3D) games beyond 2D, demand has increased and this has led to an increase in the requirements of players or customers, which has enabled 3D game technology to develop more realistic graphics, It is now possible to immerse [1][3]. There is also a stage in the game immersion. Brown Cairns interviewed the gamers in three stages: 'captivity' - 'commitment' - 'full commitment'. Among them, users who feel 'commitment' and 'full commitment' feel strong emotions to the character who is moving and feel the game environment as real environment [4]. And in addition to the above, numerous games add "realistic hitting" to give players a fun and drive them to play the game for a long time [2]. To get a sense of hitting, many developers have used animations, sounds, and colorful effects. These effects have maximized human hearing, vision, and tactile sense in order to make sense of impact and induce action as a mediator to accept a sense of hitting [2].

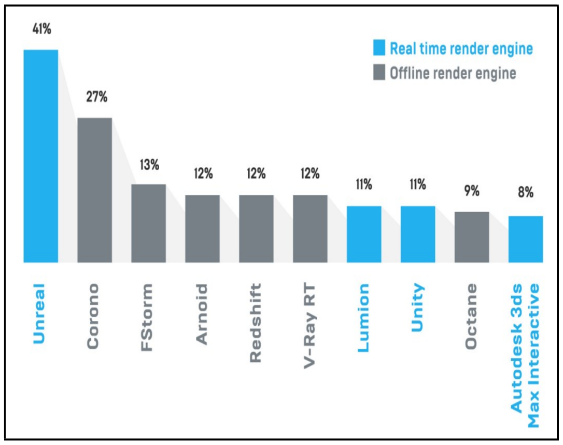
The natural repetition and local destruction of the subject in this paper is also closely related to the VR field. As the application of virtual reality technology becomes more widespread in the future, the abundant sensory abilities and 3D screen environment in the game domain make VR the ideal video game tool, and the speed of VR development in the game is faster than other fields [5][6][7].

In the game, the interaction between the player and others is an essential factor, and games based on VR or Mixed Reality(MR) using Head Mount Display(HMD) are no exception. For this, [6] implemented an MR System for the Behavior-like Interaction using an additional device to HMD that supports VR. [6] presents the theoretical content, the overall contents of the implemented MR System, the experiment, the evaluation, and the results.

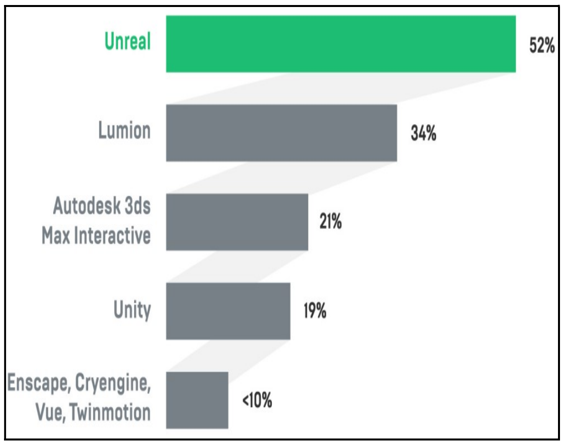
Moreover, it is more likely to be used in various industries, such as the use of simulation shooting equipment that replaces the actual shooting range, for military training, rather than limited to games. In fact, immersive education using HMD is used in various fields such as parachute training, familiarization of large traps, and infantry tactical training, and many IT defense companies are intensively developing immersive VR training system [8][9][10].

In a rapidly changing society where there is a great deal of available information and knowledge, adopting and applying the information at the right time and right place is needed to the main efficiency in both school and business settings.

Figure 1. Simulation of Hurricane Florence using Unreal Engine 4..



(a) Rendering engines



(b) Game engines

Figure 2. Usage Comparison in Game Engines.

Augmented Reality (AR) is one technology that dramatically shifts the location and timing of education and training. This literature review research describes AR, how it applies to education and training, and the potential impact on the future of education [9].

With Unreal Engine 4 which is chosen in this paper, there are more and more cases of simulating real situations, natural disasters, and experiencing specific situations. In September 2018, AFP released a 3D simulation showing the power of the hurricane Florence on the US East Coast as shown in Figure 1. The program used was the Unreal Engine 4 [12]. Unreal Engine 4 realistic results are being deployed in many industries [13].

According to a survey by Epic Games, the most popular real-time renderer for real-time producers was the Unreal Engine 4 as shown in Figure 2 (a). In addition, the actual Unreal Engine adoption rate has almost doubled over the past two years. According to a Computer Graphics Architect's questionnaire in 2016, Unreal Engine 4 showed the highest ranking when asked if he was studying a rendering engine for the future as shown in Figure 2 (b).

In this paper, Unreal Engine 4 basically provides the functions related to physics as a function so that users can calculate the value when they input a certain value and then show the result value on the game screen. Also, in this paper, we tried to produce the naturalness of the site destruction by using it. However, before using the formulas provided by the engine, we first used physical mathematical models related to the amount of impact of the object, and then used the formulas after identifying and proving that the derived formulas were the same as those provided by Unreal Engine 4. Experiments based on the ability to visualize the naturalness of the site destruction that is desired to be performed and then verifies the efficiency.

The study in [14] designs a third person action game using only DirectX library instead of the specialized techniques in game engines. By doing so, the game development costs will be minimized. The study in [14] also uses several basic algorithms in order to process the various events and to make the animation effects more efficiently managed in the graphic device.

As the Virtual Reality (VR) environment becomes popular, researches on how to feel force feedback under the VR environment have been actively studied [15]. In the use of such reaction force, hardware and software supporting reaction force are essential. In addition to the existing tabletop fixed force feedback game controllers, various haptic devices are required for the game player to obtain better haptic effects [15]. The study in [15] deals with custom haptic suits specially designed to give the body a feel. Also, the study in [15] presents a game interface for this, and proposes an effect generator that can edit and reproduce the reaction force that can give a sense of realism to game contents.

As one of elements to be able to endow more exciting and higher degree of completion for the game, the feeling of hit is realized by image, sound and body-sensing (vibration) effects [16]. When the feeling of hit is realized by game developer, most proper effects will be chosen with regard to genre, system and standpoint of world for the game [16]. In general, most of choices for the effects are performed by the experience of game developer or referring the other games [16]. The study in [16] introduces efficiency and important factors for the feeling of hit by analyzing the properties and degrees of feeling for all effects to represent the feeling of hit through experiments. For this, a software simulator was implemented to test all effects and therewith the final results are presented through questionnaires for the feeling of hit sent to gamers [16].

Most of the man-made systems can be modeled as a hybrid system which consists of both the high-level and the low-level component model [17]. High level model is responsible for decision-making and the low-level one takes control of the mechanical component parts [17]. Since the two models require different interpretation method according to their type, analysis of a hybrid system becomes a difficult job. For the Analysis of the high-level model, methods for discrete event system models such as finite state machine (FSM) can be used [17]. On the contrary, numerical analysis techniques are required for the low-level continuous-time system model. Since it becomes a difficult thing for a modeller specifies and develops both the two-level models altogether, the study in [17] proposed an efficient hybrid simulation method which employs a game physics engine that has been widely and successfully used in the area of game industry.

The study in [18] describes a simulation system that has been developed to model the deformation and fracture of solid objects in a real-time gaming context. Based around a corotational tetrahedral finite element method, this system has been constructed from components published in the graphics and computational physics literature [18]. The goal of [18] is to describe how these components can be combined to produce an engine that is robust to unpredictable user interactions, fast enough to model reasonable scenarios at real-time speeds, suitable for use in the design of a game level, and with appropriate controls allowing content creators to match artistic direction. Details concerning parallel implementation, solver design, rendering method, and other aspects of the simulation are elucidated with the intent of providing a guide to others wishing to implement similar systems. Examples from in-game scenes captured on the Xbox 360, PS3, and PC platforms are included [18].

The representative online game that combines the destruction of the region in Korea is 'Mabinogi Heroes' [19].

The main content of the game is to combat the destruction of the area before the boss, and when the user superimposes Damage of a specific area with a smash attack, a general attack or a special attack, the surrounding area is broken through the colorful animation to give the users a visual pleasure At the same time, the sense of realism and sense of action significantly increased.

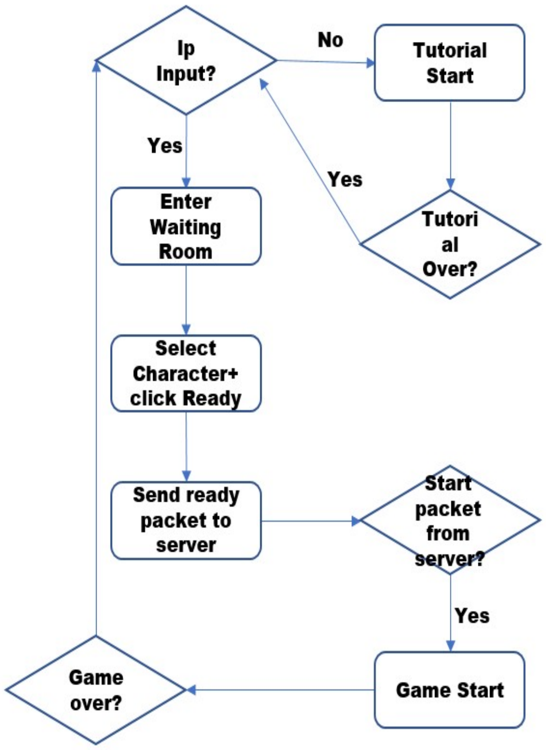
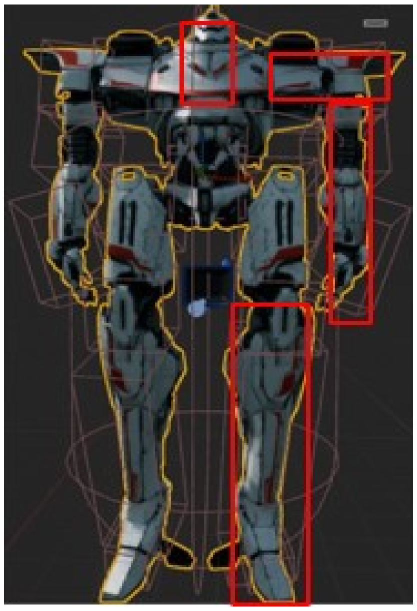
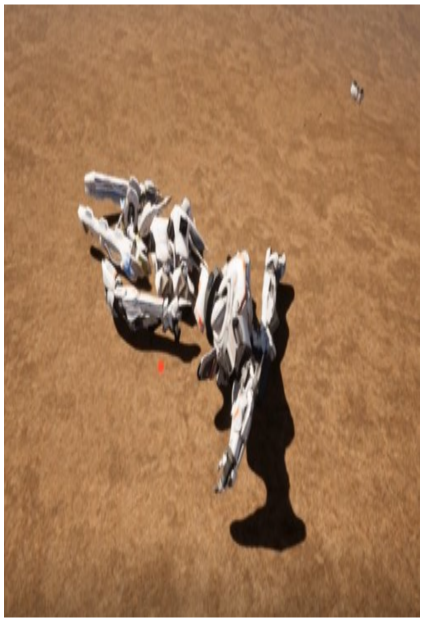


Figure 3. The Operational Flow of Game.

In this paper, we implement real-time Robotics First Person Shooter (FPS) 3D game which is linked with online server. When each of the joints of each robot hit the bullet by each part, it was calculated according to the velocity of the bullet and the mass of the joint, thereby naturally inducing the part to fall off and giving a sense of hitting to try to maximize the realism to the players.



(a) Physical Parts



(b) Breaking Parts

Figure 4. The Destroyed Parts.

2. Development of Robotics FPS 3D Game based on Unreal Engine 4

Figure 3 is a flow chart showing how the entire game works. The overall progression to the game is simple because it is the primary purpose of how to naturally immerse the players as their parts are destroyed.

```
typedef struct sc_packet_start
{
            unsigned char size;
            unsigned char type;
            short id;
            unsigned char Start;
};
```

```
typedef struct cs_packet_characterselect
{
            unsigned char size;
            unsigned char type;
            short id;
            unsigned char characterType;
};
```

Figure 5. The Packet Structure.

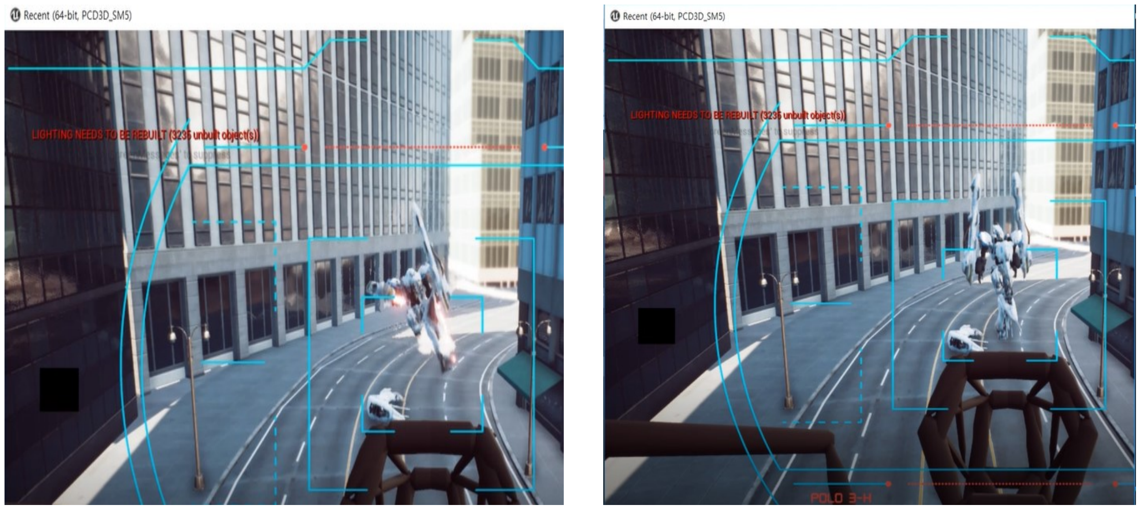
As shown in Figure 4 (a), each model has specified the area to be crushed. And, when the projectile comes flying and damages it, the damage is superimposed on the certain part, and the joint is destroyed later. Then complete decomposition is possible as in Figure 4 (b).

When the game starts, the server tells the red team and the blue team who play the game which character the enemy has chosen, and updates the coordinates and rotation values in real time. In addition, the server uses the vector <Bullet> data type to calculate the bullet type, coordinates, and rotation value of each player and sends the result to the clients.

Using these results, the clients accumulate the damage by calculating the location of the bullets and the shot area that they hit, and then when the damage is done, the shot area falls off.

The client sends and receives packet types that define protocols such as character movement, rotation, and bullets to and from the server. The IOCP server must check whether the packet information transmitted in real time is lost or not. In this case, the size variable is referenced in the structure defined above. After the server knows how many bytes the client will send, it must check whether the size of the transmitted packet matches the value sent from the variable size above. If it is confirmed that the size of prediction data reported by size is the same as the packet size to be processed, the data is transmitted to the relative clients after analyzing the information of the packets actually sent. In this case, after declaring the function Process Packet () function to analyze the data separately, it confirms and interprets what type of data is transmitted with the switch statement.

If the client satisfies the condition that the player is damaged after a certain damage, the part is destroyed and the part is partially destroyed, the client side that has been damaged first informs the server that he is dead, the game ends with dividing the winner and the loser, and sends it back to the game screen.


Recent (64-bit, PCD3D_SM5)
LIGHTING NEEDS TO BE REBUILT (3235 unbuilt object(s))
Recent (64-bit, PCD3D_SM5)
LIGHTING NEEDS TO BE REBUILT (3235 unbuilt object(s))

(a) Before Breaking One Leg *(b) After Breaking One Leg*

Figure 6. Game Screen Shots for a Broken Leg.

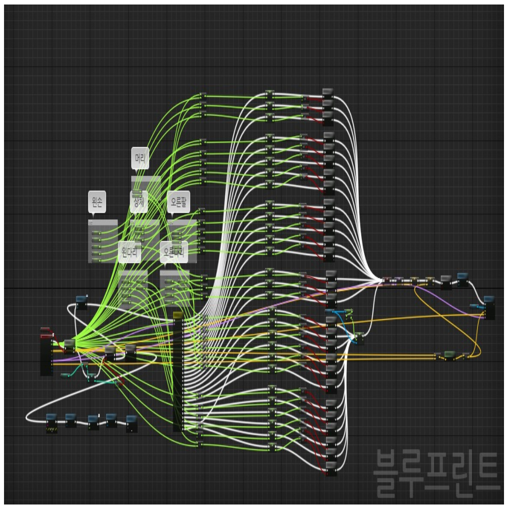
Table 1. Resources Used in Robotics FPS 3D Game

Models	No. of Triangles	Size by (X, Y, Z)
YellowCap	46,007	(416, 281, 131)
Zmech	47,412	(57, 91, 33)
WhiteCrow	58,439	(156, 235, 81)

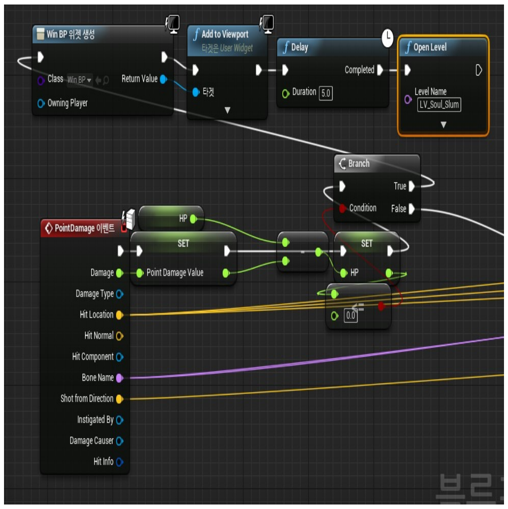
In Figure 6, you can see that the right leg of the American Robot player, which has been hit by the Korean robot player's shot, has broken down. It coordinates with the IOCP server to send and receive bullet coordinates, moving coordinates, and the right part in real time. Unlike general large-scale access MMORPG games, real-time FPS body-part destruction games developed in this paper do not implement dead reckoning because there is no large-scale access. When the state is updated, the client and the server periodically transmit the state variables to each other and implemented it.

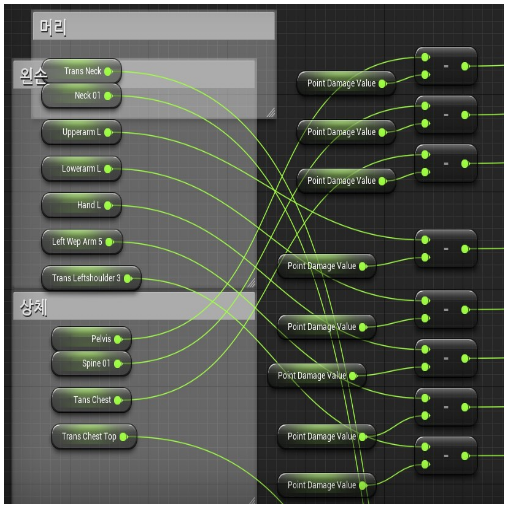
There are three models used in the robotics FPS 3D game created in this paper, and they use YellowCap, Zmech, and WhiteCrow models, respectively. The models were purchased from the Unreal Engine Asset Store for free and paid offerings. And resources and animations that are not provided are self-produced. Table 1 shows the resource size and the number of triangles of the three models purchased.

The size of each model is shown in Table 1 and is based on the right-hand rule (x, y, z).

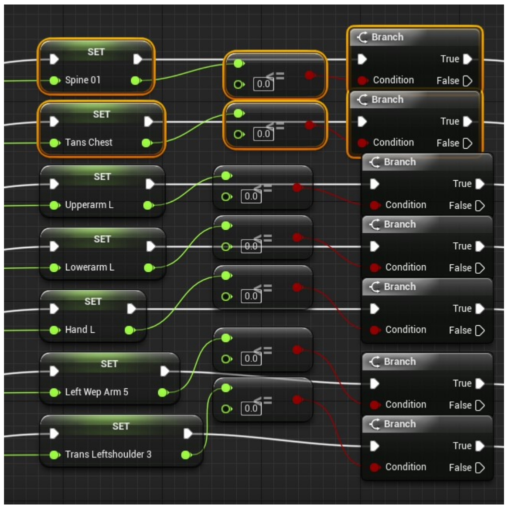


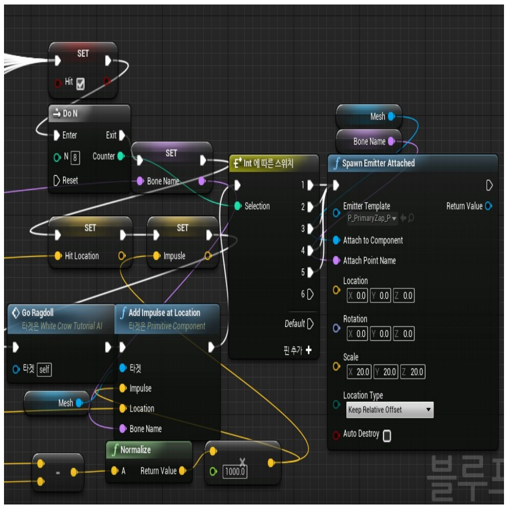
(a) Blue Print Pre-View





(b) Blue Print Process 1 (c) Blue Print Process 2





(d) Blue Print Process 3 (e) Blue Print Process 4

Figure 7. Blue Print Design of Unreal Engine 4 for Body-Part Destruction.

Table 2. Momentum of Bullets

Bullet Weight \ Bullet Speed	100m/s	200m/s	300m/s	400m/s	500m/s	600m/s
5kg	500kg m/s	1000kg m/s	1500kg m/s	2000kg m/s	2500kg m/s	3000kg m/s
10kg	1000kg m/s	2000kg m/s	3000kg m/s	4000kg m/s	5000kg m/s	6000kg m/s
15kg	1500kg m/s	3000kg m/s	4500kg m/s	6000kg m/s	7500kg m/s	9000kg m/s
20kg	2000kg m/s	4000kg m/s	6000kg m/s	8000kg m/s	10000kg m/s	12000kg m/s

3. DESIGN OF BODY-PART DESTRUCTION

Before describing the subclause on body-part destruction, we will describe Blue Print (BP) how we did body-part destruction inside the Unreal Engine4.

In this paper, Robotics 3D FPS game is designed to be destroyed in four stages. Specifically, there is a process of searching for a socket name for each part of an object and interlocking with the corresponding socket. At this time, since the model of the object used by the user does not have a socket or the name of the socket is different for each model, the developer must know and use it. As shown in Figure 7, the steps involved in body-part destruction are as follows.

① Damage occurs.

② Calling the socket at the point where the Damage is applied.

③ Change HP is HP - Damage of Point socket .

④ If the HP of the socket is <= 0, cut at the relevant site mesh.

In the process of Figure 7 (b), if there is a collision with an object having a damage through a collision check, it is a process of checking a collision point. Figure 7 (c) shows the socket name of the object (Korea, White, Yellow model) at the point of collision and connects to the socket of HP. Figure 7 (d) compares the values obtained after subtracting the Damage of the object which collided with the HP of the sockets of each visited region. In Figure 7 (e), when the HP of the socket is compared with 0 or less than 0, the socket is cut at the mesh.

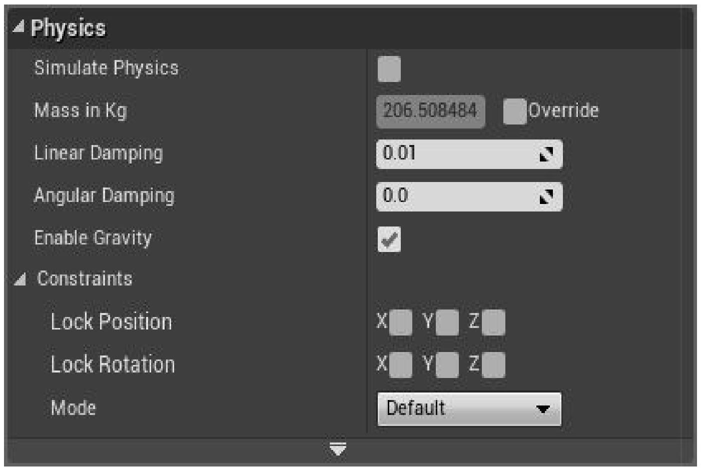
Generally, when the ballistic test is performed, the basic bullet speed is set at 200m/s, and the ballistic speed in Unreal is set at the same speed of 200m/s [8]. To carry out the experiment, the speed of the bullet trajectory of the Korean model was set to 200m/s, and the standard of the mass of each model was set to 50kg.

As shown in Table 2, the momentum of the bullet is obtained, and this momentum is used to substitute the impact amount related to the shot area again. The right part of the robot has a velocity value of 200m/s provided to the trajectory, and when the weight of each part of the robot is 50kg, the part to be shot is separated naturally.

The momentum is expressed as the product of the mass and velocity of the object in terms of the amount of motion of the object, such as Equation (1).

$$p = mv \tag{1}$$

The amount of impulse is expressed in terms of the amount of impact applied to the object when a certain force is applied to the object for a certain period of time, such as Equation (2).



Physics

Simulate Physics	
Mass in Kg	206.508484 Override
Linear Damping	0.01
Angular Damping	0.0
Enable Gravity	✓
Constraints	
Lock Position	X Y Z
Lock Rotation	X Y Z
Mode	Default

Figure 8. Physics Setting in Unreal Engine 4.

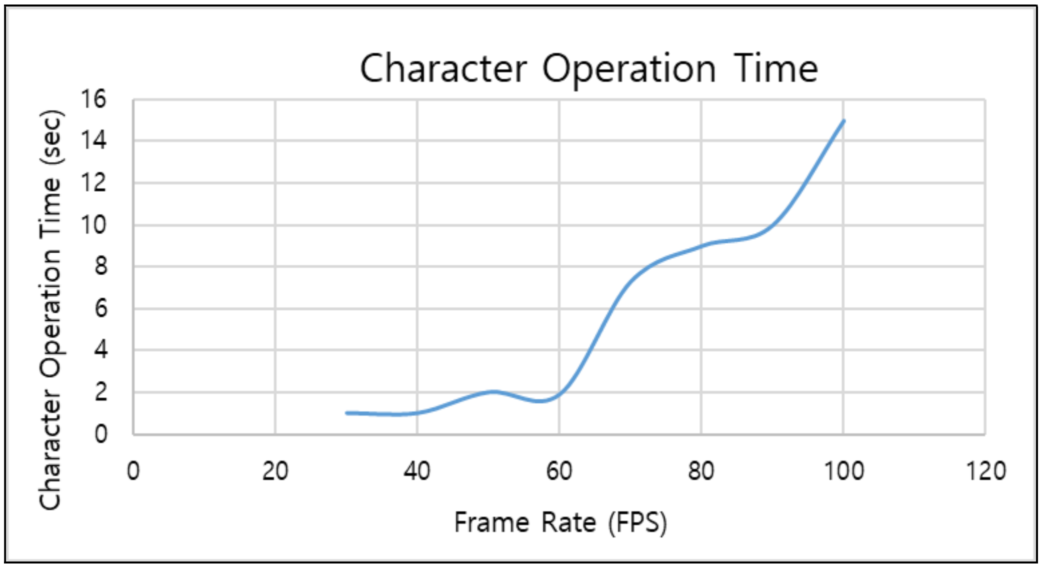


Figure 9. Character Operation Time according to Various Frame Rates.

$$I = F\Delta t \qquad (2)$$

The magnitude of the momentum can be expressed as the product of the mass and velocity of the object, and the direction of the momentum is the same as the direction of motion of the object. The magnitude of the impulse can be expressed as the product of the force acting on the object and the force acting on it, and the direction of the impulse is equal to the direction of the applied force. Equation (3) refers to the force that an object receives by impact. Impact forces can be obtained through changes in momentum over a very short period of time.

$$F = \frac{\Delta p}{\Delta t} \qquad (3)$$

The relation between momentum and impulse is equal to the amount of change in momentum as shown in Equation (4).

$$F = ma = \frac{m(v_2 - v_1)}{\Delta t} = \frac{(mv_2 - mv_1)}{\Delta t} \quad (4)$$
$$I = F\Delta t = mv_2 - mv_1 = \Delta p$$

That is, the amount of change in the amount of motion of the object is equal to the amount of impact received by the object. Using Equation (4), we experimented with increasing the mass of the robot parts while continuing to increase the speed of the bullet, and then derived the most natural results. In this paper, there are three kinds of bullets, and we derive the formula based on machine gun bullets.

4. PERFORMANCE EVALUATION

As shown in Figure 8, Physics in Unreal Engine 4 is a preference for physics simulation. By assigning the desired trajectory, velocity, and mass values to each property in the simulation, the Physics simulation results are displayed by the Unreal Engine4 internal process. You can use the Epic Games documents [11] to get a simulation result after adjusting and setting various states and situations.

Based on Unreal Engine 4, when it comes to Physics Bodies by creating Physics Assets and Skeletal Meshes, which automatically have the properties of physics by default [7]. When you work with the properties of Physics on Static Meshes or any other assets that we will attempt to simulate physics with, we will see a handful of different parameters that we can change in order to produce the desired effect under the Details panel [7] as shown in Table 3.

Unreal Engine 4 has an option to set the FPS automatically and is user adjustable. In Unreal Editor, enter the environment settings, enter [Engine] - [General Settings] - [FrameRate], and check the "Smooth Frame Rate", "Use Fixed Frame Rate", "Fixed Frame Rate", “Smoothed Frame Rate Range”, and “Min Desired Frame Rate”. Fixed Frame Rate allows you to fix the frame completely and play the game, or arbitrarily adjust the frame rate range to smoothly play to provide pleasant game play. We set the frame rate arbitrarily and measured the optimum frame rate when the game was executed in the computer specification of Intel i5-6500 CPU 3.20GHz, memory 8GB, 64 bit operating system, x64 based processor.

Table 3. Physics Properties [7].

Properties	Description
Simulate Physics	This parameter allows you to enable or simulate physics with the asset you have selected. When this option is unchecked, the asset will remain static, and once enabled, we can edit the Physics Body properties for additional customization.
Mass in Kg	This parameter determines the Mass of the selected asset using kilograms. This is important when you work with different sized physics objects and want them to react to forces appropriately.
Linear Damping	This property is used to simulate the different types of friction that can assist in the game world. This modifier adds a drag force to reduce linear movement, reducing the translation of the object.
Angular Damping	This property is a modifier of the drag force that is applied to the object in order to reduce angular movement, which means to reduce the rotation of the object.
Locked Axis	This parameter allows you to lock the physical movement of our object along a specified axis. We have the choice to lock the default axes as specified in Project Settings. We also have the choice to lock physical movement along the individual X, Y, and Z axes. We can have none of the axes either locked in the translation or rotation, or we can customize each axis.

In Figure 9, the frame rate is fixed at 100 and the game is played with IOCP server. I gradually decreased the frame rate by 10 and confirmed the result. The x-axis in Figure 8 is the frame rate (in Frames per Second (FPS)) and the y-axis is the time in seconds that the scene changes when the robot moves to the front. As a result, as shown in Figure 8, when the frame rate is 100, it took 15 seconds for the screen to be updated once moved. As the frame rate (FPS value) decreases, the speed of the screen update gradually increases. Especially, when the frame rate is set to 60 or more, the screen update time rapidly increases. Therefore, it is concluded that the optimal frame rate value is most suitable between 30 and 60. By setting the frame rate of a 3D game based on Unreal Engine 4 between 30 and 60, we can simulate partial destruction of a 3D game character while reproducing a natural sense of attack.

5. CONCLUSION

In this paper, we have examined the main expressions for realizing the natural part destruction in the 3D game based on Unreal Engine 4. We propose a mathematical method to produce natural destructive fracture in First Person Shooter (FPS) games. Through the simulation using the Unreal Engine 4, the proposed method proved the efficiency of the physics and mathematical expressions derived from the colliding objects and the experiment in order to achieve optimal part destruction. By setting the frame rate of a 3D game based on Unreal Engine 4 between 30 and 60, we can simulate partial destruction of a 3D game character while reproducing a natural sense of attack.

As the years go by, the technological power of science is rapidly developing. At the same time, the performance of computers is getting better and more and more technologies are being added to games. Representative technologies include VR and AR, and there are studies to give more realism and enjoyment to people by experiencing actual experience in virtual world while obscuring the distinction between real world and virtual world.

However, there are still many unresolved issues such as limited indoor space for playing VR games and motion blur due to VR viewing angle problems. Future work will be to demonstrate the effectiveness of VR in terms of viewing angle, game immersion and realism.

6. ACKNOWLEDGEMENT

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No. 2016-0-00204, Development of mobile GPU hardware for photo-realistic real time virtual reality).

REFERENCES:

[1] Ki-Yoon Kim, and Ju-Hwan Lee, “Differences of Fun and Immersion according to Game User Interfaces in the Virtual Space”, Journal of Digital Contents Society, Vol.18, No.8, pp.1489-1494, 2017.

[2] Jin-Seok Seo, Nam-Gyu Kim, “Empirical Analysis of the Feeling of Shooting in 2D Shooting Games”, Journal of the Korea Society of Computer and Information, Vol.15, No.2, pp.75-81, 2010.

[3] Hyeog-In Kwon, Hyun-Jung Rhee, Jin-Wan Park, “Exploring the Immersion Degree Difference Between 3D and 2D: Focus on Action-Adventure Game”, The Journal of the Korea Contents Association, Vol.11, No.1, pp.157-164, 2011.

[4] E. Brown and P. Cairns, “A Grounded Investigation of Immersion,” In: CHI'04 extended abstracts on Human factors in computing systems. ACM, pp.1297-1300, 2004.

[5] Gao LinChao and Choi Chul-Young, “Analysis of the Space Layout according to VR game Environment”, The Conference Proceeding of the Korea Contents Society, pp.421-422, 2018.

[6] Min Soo Choi and Jun Park, “The Mixed Reality System for the Behavior-like Interaction in VR Games”, Journal of Korean Society for Computer Game, Vol.30, No.4, pp.41-48, 2017.

[7] Katax Emperore, Devin Sherry "Unreal Engine Physics Essentials", Packt Publishing Ltd, 2015.

[8] Jong-Hyun Jo, Youn-Shin Lee,Hai-Lan Jin, "Numerical Simulation of Steel/Kevlar Hybrid Composite Helmet Subjected to Ballistic Impact", Transactions of the Korean Society of Mechanical Engineers A, Vol.36, No.12, pp.1569-1575, 2012.

[9] Kangdon Lee, "Augmented reality in education and training", TechTrends, Vol.56, No.2, pp.13-21, 2012.

[10] Mark A. Livingston, et al., "Military applications of augmented reality", Handbook of augmented reality, Springer, New York, NY, pp.671-706, 2011.

[11] URL, api.unrealengine.com/KOR/Engine/Physics/PhysicsBodies/Reference/index.html

[12] URL, news.naver.com/main/read.nhn?mode=LSD&mid=sec&oid=052&aid=0001193429&sid1=001

[13] URL, motorgraph.com/news/articleView.html?idxno=20882

[14] Sung-Ohk An, HeeBum Lee, Dong-Won Park, SooKyun Kim, Jinyoung Jung, “Design of 3D Action Game for PC Environment”, Journal of

the Korea Society of Computer and Information, Vol.19, No.4, pp.63-69, 2014.

[15] Heejung Bae, Byungtae Jang, “Haptic Suit Interface for Haptic Game”, The Conference Proceeding of Korea Multimedia Society, pp.499-502, 2003.

[16] Hyung-Je Cho, Sung-Jun Moon, “A Study on Enhancing Efficiency for Feeling-of-Hit in Games”, Journal of Korea Game Society, Vol.12, No.2, pp.3-14, 2012.

[17] Lee, Wan-Bok, and Seuc-Ho Ryu. "An Efficient Hybrid Simulation Methodology Using the Game Physics Engine." Journal of Digital Convergence, Vol.10, No.10, pp.539-544, 2012.

[18] Eric G. Parker and James F. O'Brien, "Real-time deformation and fracture in a game environment", Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. ACM, pp.165-175, 2009.

[20] Mabinogi Heroes Game, http://heroes.nexon.com/