Design and Implementation of Virtual Reality Contents based on Dynamic Event

Dongik Lee, Giyeol Baek, Yangwon Lim and Hankyu Lim(corresponding author)

***Abstract*—**Recently, thanks to the development of ICTs, hardware items such as controllers and displays, software for processing of the hardware items and content have been diversely developed in virtual reality content markets. In the market, several controller and display are being developed. Kinect and Leap Motion Controller are examples of motion recognition controllers. Also head mount display(HMD) is used as display unit. In addition, virtual reality content related technologies utilizing those technologies are also required for processing of interactions with users and for processing of events. The usage of combination of peripheral devices is not usual in virtual reality contents. Therefore, in the present study, virtual reality content that can process dynamic events utilizing Kinects which are motion recognition controllers that have been the mostly actively utilized recently and Oculus Rifts which are HMDs was designed and implemented.

***Keywords*—** VR contents; Leap Motion Controller; Dynamic Event; HMD; oculus lift; kinect

I. INTRODUCTION

RECENTLY, thanks to the development of information and communications technologies, virtual reality systems have been developed rapidly and studies of motion recognition and interactive controllers that induce users to be immersed as if they actually move and behave in virtual reality have been becoming active. Motion recognition controllers and relevant controllers are being developed and sold through many home console game devices and personal computers (PCs) [1]. Among them, Kinect supports both Xbox and PCs and is utilized in diverse virtual reality areas such as medicine and education as well as the game area. In addition, these controllers are also utilized in diverse virtual reality areas such as medical service and education. However, they have not been much welcomed in the virtual reality content industry recently because they have problems such as the issue of securing spaces for using Kinects at home, difficulties in combined use of Kinects and other controllers, low recognition rates and speed, input delays, and operation performance poorer than that of existing input devices. In addition, the position of Kinects in the virtual reality content industry is gradually weakening in that there is no appropriate killer content. HMD(Head Mounted Display) had many problems that made the universalization of them difficult at the beginning such as inconvenience in wearing due to the size and weight of the virtual reality equipment, low resolution of the display, and high prices. However, thanks to the recent development of displays and performance improvement, diverse HMDs have been developed such as the HMD devices of Sony, Oculus Rifts of Oculus, and Gear VR of Samsung and VR related markets have been also growing steadily. However, they are not being actively developed yet in other than the game area owing to the low recognition rate and level of awareness. In the case of HMDs, at an early stage, wearing them was uncomfortable because of their size and weight and they had many problems such as low resolution and expensive price of displays, but thanks to the development of displays and improvement in their performance, diverse HMDs such as Sony's HMD, Oculus' Oculus Lift, and Samsung's Gear VR have been developed and the growth of the relevant market has been continuing [2]. As such, not only hardware technologies such as Kinect and HMD but also software technologies have been designed and developed to be suitable for the development and utilization of virtual reality content. With these technologies, input methods were changed from those using keyboards and mouses changed into those using touch screens and have been recently developed into those using motion recognition technology. Instead of the two-dimensional input methods using keyboards, mouses, etc. for existing two-dimensional content, virtual reality content requires the processing of users' diverse dynamic movements such as the movements of hands, fingers, and bodies. Content for interactions with users has become necessary for input processing of these dynamic movements of users, that is, dynamic events.

This paper designed and embodied dynamic event driven virtual reality contents using together Kinect, a motion recognition controller, and Oculus Lift, an HMD. The contents were to embody prototypes which moved characters utilizing Kinect and made display with Oculus Lift using C++ and DirectX11. Through this, developing the virtual reality game and contents proposed in the paper is expected to be possible, virtual reality content proposed in the present paper for responses to users' diverse methods can be designed and developed.

This work was supported by a grant from 2015 Research Funds of Andong National University.

II. THE RELATED RESEARCH

A. Kinect

When the motion of the user is recognized using Kinect, the location information values of the user's skeleton may be

derived. Information on the skeleton is composed of a total of 20 joints including humans' hands, feet, and head [3]. We utilized location information values of these joints and used them in order to move characters.

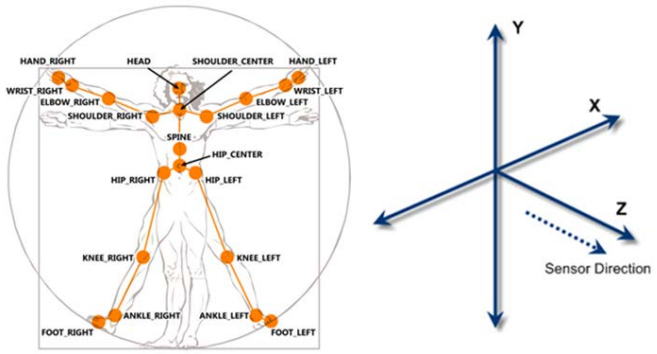


Fig. 1. Skeleton and Joint of Kinect

B. Oculus Lift

Another function of the Oculus Lift is head tracking, which is a function of applying changes in field of vision within the game and contents by sensing the head's rotation and forward and backward and up and down movements by the Oculus Lift.

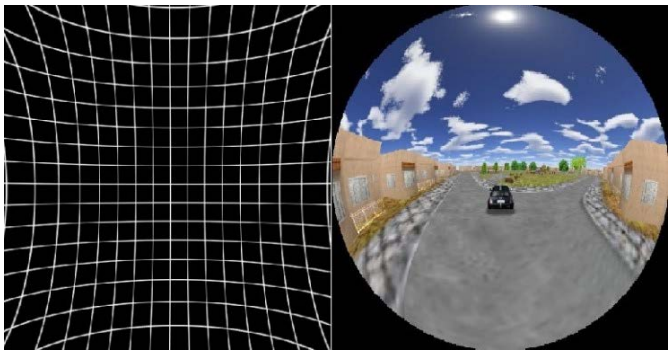


Fig. 2. Fish Eye Rendering

C. Interactivity in VR

Interactivity is related to two-way communications between the system and users or between users and is provided so that desired purposes can be quickly and easily accomplished. In particular, interactivity between the system and users in virtual reality spaces should be measured based on the reactivity of users' dynamic behaviors. Whereas reactivity was measured based on inputs using keyboards and mouses in the past, reactivity can be detected based on users' physical movements for interaction in virtual reality spaces. Since virtual reality spaces are implemented with three-dimensional objects, performing given works in virtual spaces using the existing input methods is difficult due to the limited or fixed viewpoints. Interactions can be made smoothly only when there is an interface that dynamically creates viewpoints suitable for three-dimensional spaces to solve this problem. In the present study, event processing in three-dimensional spaces was designed so that users' dynamic events can be processed in such virtual reality spaces based on interactivity between the system and users.

III. Function and Implementation

A. Function Analysis

The functions of the program are divided into three in order to perceive a user's motions through this Kinect and utilize the perceived motions as data to move virtual reality characters. First, Kinect has a function of perceiving a user's motions and manipulating characters of the game and is divided into head tracking of Oculus Lift, display, and virtual reality contents. The figure 3 below is the perspective plan of virtual reality contents utilizing the Oculus Lift and Kinect.

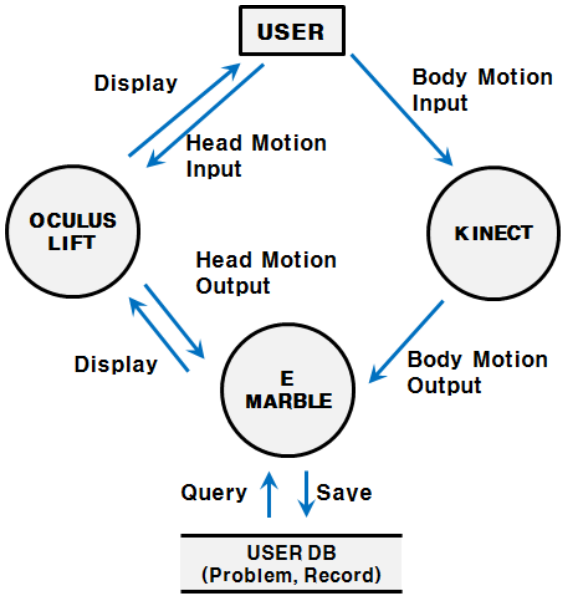


Fig. 3. DFD Structure_perspective plan

In the perspective plan of Figure 3, E Marble is the name of contents this paper designed and embodied. It is a main function of the game which is to bring information on the skeleton of the Kinect, manipulate characters and the virtual reality contents, bring information on head tracking from the Oculus Lift, calculate View Matrix and Projection Matrix, and perform rendering them to the Oculus Lift. Owing to the characteristics of the controller and the display, each function has high degree of mutual coupling.

B. Kinect Implementation

In order to move characters in the same way as the movements of the user, information on skeleton joints read from the Kinect should be applied to the character mesh's bones by calculating rotational matrix of each joint based on the information on the skeleton's joints. The Figure 4 below represents composition to correspond the character's bones with the Kinect's joints, and Kinect Tutorial and Avaterring Sample Source were referred to in order to derive the formula to calculate each joint's rotational matrix with information on

joints.

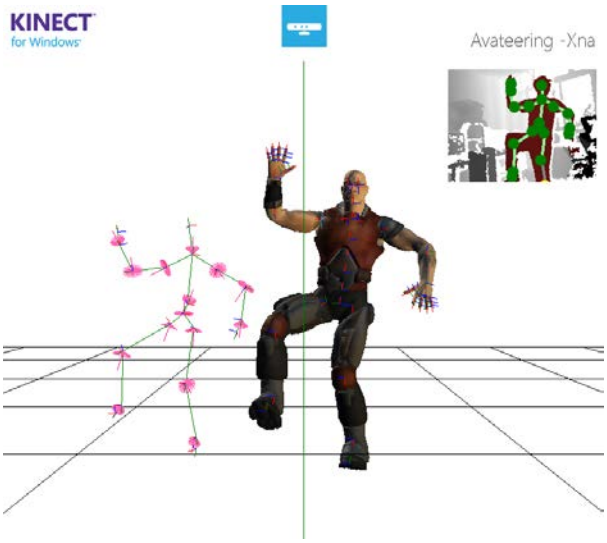


Fig. 4. Screen of Avateering C# Sample
(Left) Skeleton & Joint, (Middle) Character, (Right) Player

The Figure 5 below represents Kinect Skeleton Joint and Implementation Character Bone. Bone matrix calculated in such a way may be applied to character mesh through Skinning Animations.

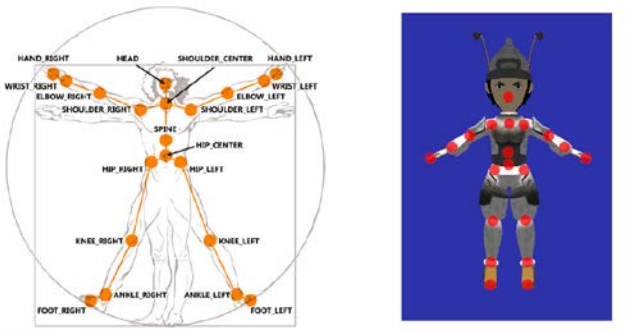


Fig. 5. Kinect Skeleton Joint and Implementation Character Bone

The following source codes are some of C# Sample codes converted into the form of C++, DirectX11. The XNAMath library was used as a math library. Since refactoring was conducted for implementation, the source codes were implemented in the same structure as that of C# Sample although there are some differences.

```
void SetJointTransformation(
const NUI_SKELETON_BONE_ORIENTATION& bone,
const NUI_SKELETON_DATA& skeleton,
XMMATRIX& bindRoot)
{
if (bone.startJoint ==
NUI_SKELETON_POSITION_HIP_CENTER &&
bone.endJoint ==
NUI_SKELETON_POSITION_HIP_CENTER)
{
bindRoot._41=0.f; bindRoot._42=0.f; bindRoot._43=0.f;
XMMATRIX invBindRoot = bindRoot;
XMVECTOR matInvDeter;
invBindRoot =
XMMatrixInverse(&matInvDeter, invBindRoot);
XMMATRIX hipOrientation = ConvertMatrix(
bone.hierarchicalRotation.rotationMatrix);

// find center bone, (parameter is bone's name)
MAP_TRANS_ITER iter =
m_mapBoneTransforms.find("Bip01 Pelvis");
// get center bone's transforms
XMMATRIX pelvis = *iter->second.matTransforms;

pelvis._41=0.f; pelvis._42=0.f; pelvis._43=0.f;
XMMATRIX invPelvis;
invPelvis = XMMatrixInverse(&matInvDeter, pelvis);

XMMATRIX combined =
(invBindRoot * hipOrientation) * invPelvis;

ReplaceBoneMatrix(
NUI_SKELETON_POSITION_HIP_CENTER,
combined, true);
}
else if (bone.endJoint ==
NUI_SKELETON_POSITION_SHOULDER_CENTER)
{ ... }
else if ( ... ) // Else Selection Position
{ ... }                    // Next...
}
```

Fig. 6. (Source Code 1) C++ SetJointTransformation()

C. Skinning Animations

The skinning animations process is necessary for adopting the calculated bone matrix to the character mesh. The Linear Blend Skinning(LBS), Spherical Blend Skinning, Dual Quaternion Skinning are examples for processing. We implemented the Skinning Animation by LBS.

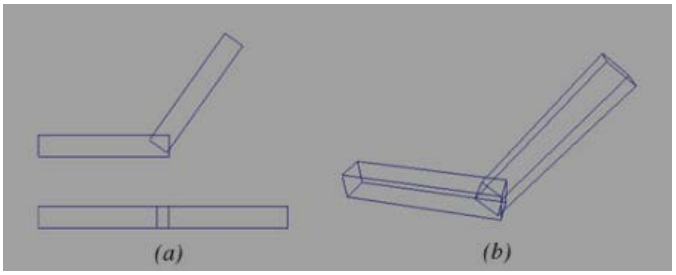


Fig. 7. Unskinning Mesh

Figure 7 is the unskinning mesh, and figure 8 is the example of the skinning mesh. The joint with the unskinning mesh is not natural, but the joint with the skinning mesh is very natural [6].

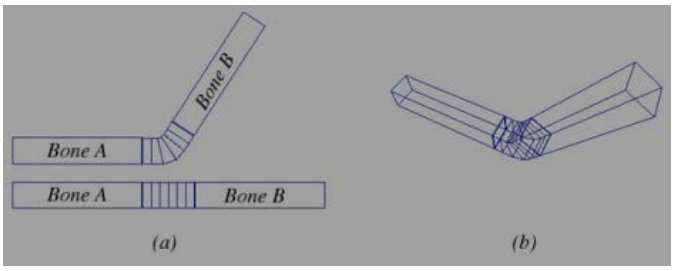


Fig. 8. Skinning Mesh & Vertex Blending

The below expression(1) is summation of each factors, the sum of w should be 1.

$$v' = w_0 v F_0 + w_1 v F_1 + w_2 v F_2 + w_3 v F_3 \quad (1)$$

v : Mesh Vertex

v' : Mixed Vertex

w : Bone Weight

F : Bone Matrix

The figure 9 is shader source of expression.

```
VS_OUTPUT output;

for (int i = 0; i < 4; ++i) {
   Output.Position +=
      mul(Input.Position,BoneMatrix[Input.BlendIndices[i]]) *
      Input.BlendWieght[i];
   Output.Normal +=
      mul(Input.Normal,g_BoneMatrix[Input.BlendIndices[i]]) *
      Input.BlendWieght[i];
}
```

Fig. 9. (Source Code 1) C++ Shader Source

D. Event Handling of 3 Dimension Space

The virtual reality contents embodied in this paper enabled virtual reality experience by using Kinect and Oculus Lift, but it is impossible to employ a mouse or keyboard for virtual reality contents manipulation. Therefore, manipulation should be made using Kinect as well. Such event was processed by setting a collider on the hands in the character's mesh and composing an object for manipulation and then detecting mutual collision.

cv: Point of surface between OBB and ball center(vector)

ov: OBB center(vector)

oa: OBB axis(vector)

ol: OBB axis length(vector)

sv: Ball center(vector)

sr: Ball radius(scalar)

$dist$: Distance between ball and OBB center(scalar)

$$dist = sv - ov$$

$$cv = ov$$

$$cv = cv + (dist \cdot oa_0) * dist$$

$$= cv + (dist \cdot oa_1) * dist$$

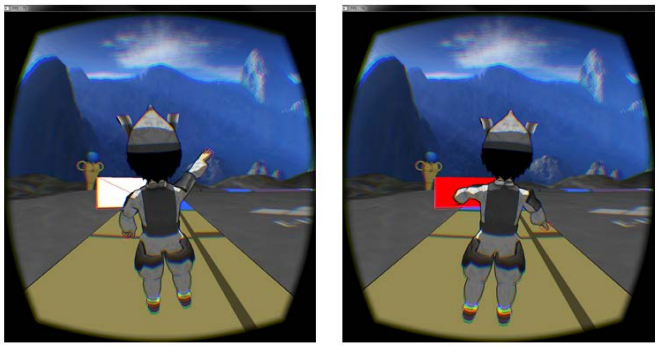
$$= cv + (dist \cdot oa_2) * dist$$

$$-ol_n \leqq (dist \cdot oa_n) * dist \leqq ol_n$$

$length = \|cv - sv\|$: $cv - sv$ (vector length)

$length \leqq sr * sr$: OBB and ball clash

UI Event : Click the button if collision ball in the character hand and UI crash during given time.



Before collision — After collision – Button Click

Fig. 10. UI Event Handling

Figure 10 above shows establishment of OBB(Oriented Bounding Box) aimed at event processing to the plan for UI output and event processing by testing collision with the sphere in the hands of the characters. The relevant character moves to the movement of a user and it was embodied for the user to click the button by extending the hand in order to trigger an event.

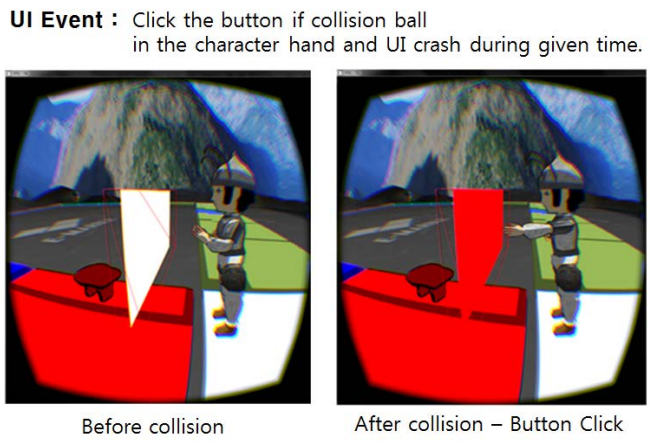


Fig. 11. UI Event Handling2 : View from Side

IV. Conclusion

The UI Event processing method based on users' dynamic inputs proposed in the present paper is advantageous in that it can allow users' free behaviors even without any separate controller other than Kinects and provide event processing for the users' free behaviors in virtual reality spaces. However, the proposed method has disadvantages such as problems in the accuracy of motion recognition using Kinect and in securing spaces and limitations in movement distances resulting from the utilization of Oculus Rifts that prevent free utilization of Kinects. In addition, it also has another disadvantage in that it may cause situations where it is crashed into invisible objects. Since Kinects still has another disadvantage of being unable to process users' turning, that is, posterior views, the movement of virtual reality is limited. The dynamic event processing based virtual reality content proposed in the present paper was implemented using Kinect v1. Kinect version v2 has been released now that provides functions as a more accurate motion recognition controller with improved recognition rates, resolution, and motion recognition performance. Controllers that can be used well in combination with Kinects include Razer Hydra which is a remote controller type motion recognition controller, biometric controller Myo, virtuix Omni that senses the movements of the lower body, and Emotiv Epoc that is a controller using brain waves and smartphones are judged to become good controllers if Gyro sensor and touch functions are utilized.

At present, HMD and motion recognition controller which can control user's dynamic input in virtual reality space have grown enough to produce games and contents, but still they have problems of high prices, low performance, and lack of key contents. Nonetheless, research and development on the area of virtual reality is continuously being made and performance of peripheral devices is being improved day by day and therefore they are good enough to be established as future new contents. In addition to the combination of Kinect and Oculus Lift proposed in this paper, utilization of other controllers will be possible to produce various virtual reality games and contents.

References

[1] Daegeun Kwon, Hyeongeun Shin, Yongje Kim, Jinyeong Jung, Seola Kim, Suncheon Jang, Daejun Lim, Yangwon Lim and Hankyu Lim, "*Action simulation game design using the leap motion controller*", Proceeding of ICMEM 2014, Turkey, 2014, pp.381-391.

[2] Davis, Simon, Keith Nesbitt, and Eugene Nalivaiko. "*Comparing the onset of cybersickness using the Oculus Rift and two virtual roller coasters.*" Proceedings of the 11th Australasian Conference on Interactive Entertainment (IE 2015). Vol. 27. 2015, pp.30-33.

[3] Guo, B., Sun, J., Wei, Y. C., Meekhof, C., amd Leyvand, T. "*Kinect identity: Technology and experience*", Computer, (4), 2011, pp. 94-96.

[4] *Oculus Lift*, https://en.wikipedia.org/wiki/Oculus_Rift

[5] *Avateering C# Sample*, https://msdn.microsoft.com/en-us/library/jj1 3104 1.aspx

[6] S.B. Park, B.W. Ryu, "*A study on effectivemess affecting on-line game in term of interactivity*" e-business research 7.5, 2006, pp.83-107.

[7] T.H. Kim, J. A. Park, "*Interface based on two hands space interaction for object making in 3 dimension Virtual environment*" HCI 2014, 2014, pp.1069-1072.

[8] Luna, Frank. "*Skinned mesh character animation with direct3d 9.0 c.*" MSDN Magazine 6, 2004, pp.18-26.

Dongik Lee and Giyeol Baek are students of Multimedia Engineering Department of Andong National University, Korea.

Yangwon Lim is a full time lecturer of Multimedia Engineering Department of Andong National University, Korea.

Hankyu Lim received the B.S. degree in Electronics Engineering from the Kyungbook National University in 1981. He received the M.S. degree in Computer Engineering from the Yonsei University in 1984. He received the PH. D. degree in Computer Engineering from the Sung Kyun Kwan University in 1997. He is a professor of Andong National University.