International Journal of Smart Home
Vol. 9, No. 2 (2015), pp. 193-206
http://dx.doi.org/10.14257/ijsh.2015.9.2.18

X3D Nodes for Representing and Rendering Real Characters in 3D Virtual Environments

In-Kwon Kim[1], Sin-Ae Kwon[1], Mihye Kim[2], Kwan-Hee Yoo[*,1]

[1]Department of Digital Informatics and Convergence and Dept. of Software Engineering, Chungbuk National University
52 Naesudongro, Heungukgu, Cheongju, Chungbuk, South Korea
khyoo@chungbuk.ac.kr
[2]Department of Computer Science Education, Catholic University of Daegu
mihyekim@cu.ac.kr

Abstract

This paper proposes and defines four eXtensible 3-dimensional (X3D) nodes to render real-world characters such as human being and animals in 3D virtual environments, which conform to the X3D standard format for representing and rendering real-world characters in virtual spaces, aiming to make them available as an extension of the X3D core nodes. Several examples are implemented to evaluate the effectiveness of the proposed X3D nodes. Implemented results demonstrate the feasibility of the proposed X3D nodes as an extension of the X3D core components for rendering real characters in 3D virtual spaces.

Keywords: *Real character representation, X3D nodes, 3D virtual space*

1. Introduction

Following rapid advances in three-dimensional (3D) technologies, including virtual reality (VR) and augmented reality (AR), application services in many areas are attempting to deliver enhanced content. Applications such as sports broadcasts, weather forecasts, movies, games, and shopping are already utilizing these 3D technologies to increase user satisfaction by achieving improved immersion through more realistic visualization. Educational learning systems are also shifting into a new paradigm, with various forms of educational content exploiting these 3D technologies to improve student immersion and interaction. Such multi-purpose learning systems include experiential and situated virtual learning systems using 2D or 3D virtual spaces and multimedia content, as well as VR and/or AR [1-5]. In such systems, students and teachers usually participate in and interact with the learning systems through virtual characters (i.e., avatars) serving as representatives of themselves in the virtual space. Learners and teachers in real-world spaces are modeled as avatars in 3D virtual spaces, and users interact with the systems through the avatars. As a result, these types of learning systems are limited as regards improvement of user immersion and interaction because users need to control and animate virtual characters through user-steered interfaces.

To improve these areas of user interaction, a method was proposed to represent and render real-world characters in 3D virtual spaces based on mixed reality (MR), where both real learners and teachers are embedded; that is, the method uses real-world characters in a virtual space instead of virtual characters to improve learner immersion and interaction [1], [2]. In this paper, we extend this by defining and developing extensible 3D (X3D) nodes for the implementation of this method, with the

* A Corresponding author

aim of achieving an implementation that conforms to the X3D architecture [6] for implementing real character rendering in virtual spaces. In addition, we plan to make them available as an extension of the X3D core standard nodes supported by the Web3D Consortium [6].

The remainder of this paper is structured as follows. Section 2 reviews the technologies on which this study was based, including MR and X3D. Section 3 provides an outline of the method for representing and rendering real-world characters in 3D virtual spaces proposed previously [1], [2], and presents the four X3D nodes that were developed for the implementation of the method. Section 4 gives several examples of the use of the proposed X3D nodes to evaluate their effectiveness. Section 5 summarizes the paper and outlines directions for future research.

2. Theoretical Background

2.1. Mixed Reality

Mixed reality (MR) has been the subject of growing research interest in a wide range of fields as an enabling technology for next-generation IT applications. MR is a more general term than AR [7], which includes VR and allows real-time mixture between computer-generated digital content (in the forms of 3D computer graphics, images, or virtual objects) and the real-world. The goal is to give a more detailed meaning to the objects in the real world and also to support enhanced user immersion via improved realism [8–11]. Users can interact with a mixed virtual environment created by seamlessly integrating virtual and real worlds in real-time. Figure 1 shows a simplified MR representation as a virtuality continuum [12]. In this schematic, AR is a synthetic environment where some virtual objects are overlapped on a real-world environment, while an augmented virtuality (AV) environment is where some real objects are embedded in a virtual environment.

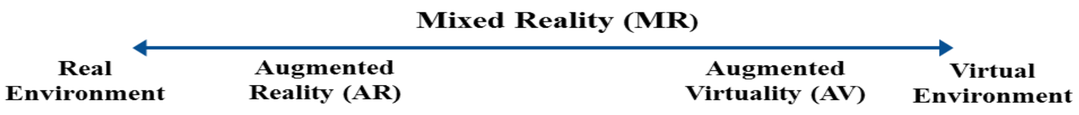

Figure 1. MR representation as a "Virtuality Continuum"

Using MR technologies, it is possible to provide multi-sensory 3D data and improved operability via tangible user interfaces (TUIs) [13, 14]. There are also several reports demonstrating that MR technologies can facilitate or enhance a number of learning methods, including active learning, constructive learning, intentional learning, practical learning, and cooperative learning [15]. Intuitive and experiential learning through strengthening presence and sensual immersions, the possibility of practical and constructive learning by combining real and virtual worlds, intentional and active learning through manipulating real objects, and the possibility of face-to-face collaborative learning through encoding real-world objects may all be facilitated using MR technologies in educational learning systems [14, 16].

2.2. X3D Standards

X3D is a modeling language and runtime architecture for defining interactive web-based 3D graphics and multimedia content [6]. It is also a free open-source software standard that uses the XML-based file format ratified by the International for Standardization and the International Electrotechnical Commission (ISO/IEC 19775) for the "storage, retrieval, and playback of real-time graphic content embedded in applications" [17] as the subsequent version of the Virtual Reality Modeling

Language (VRML). X3D provides data-encoding formats and offers advanced application programming interfaces (APIs). Advantages over other graphics standards include XML and HTML-integrated, componentized, profiled, and well-specified functionalities with a rich variety of componentized features [17], [18].

The basic unit of the X3D runtime environment is the scene graph, which contains all of the objects in the X3D system as well as the relationships among them. The scene graph is represented as a hierarchy with ordered components for functionality specifying relationships between objects. Here, a component is a collection of grouping nodes (i.e., a set of functionality consisting of several related objects). A node is the fundamental component of a scene graph. In X3D, the term *node* is usually synonymous with *object* [6], [18]. More precisely, an object is an instance of a node.

X3D has evolutionary, extensible, and embedding characteristics with a wide variety of componentized functionalities for texturing, modeling, rendering, and user interactivity. It supports spatialized audio and video, interchangeable humanoid animation, physical simulation, navigation, user interaction, 3D graphics, sound, scripting, and real-time communication that enable the implementation of enhanced VR- or MR-based systems. Other supporting features of X3D include computer-aided design (CAD), 2D graphics, programmable shaders with multi-stage texturing, lighting, geospatial positioning, polygonal and parametric geometry, pixel and vertex shaders, and hierarchical transformations. Furthermore, it supports user-defined objects, broadcast, layering, and networking [6], [17–19]. More detailed functionalities of X3D can be found in the X3D specifications supported by the Web3D Consortium [6].

3. X3D Nodes for Embedding Real Characters in 3D Virtual Spaces

3.1. Overview of Real Character Representation and Rendering into Virtual Spaces

A method for embedding and rendering real human characters in virtual spaces was proposed previously [1], [2]. Figure 2 shows the overall configuration of this method.

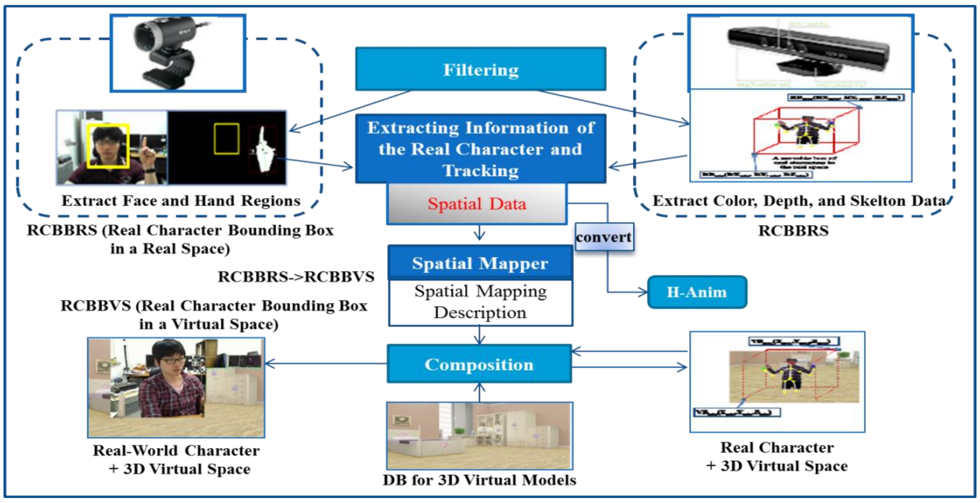
Filtering
Extracting Information of the Real Character and Tracking
Spatial Data
convert
Spatial Mapper
Spatial Mapping Description
H-Anim
Composition
Extract Face and Hand Regions
Extract Color, Depth, and Skelton Data
RCBBRS (Real Character Bounding Box in a Real Space)
RCBBRS
RCBBRS->RCBBVS
RCBBVS (Real Character Bounding Box in a Virtual Space)
Real-World Character + 3D Virtual Space
DB for 3D Virtual Models
Real Character + 3D Virtual Space

Figure 2. Overall Configuration of the Previously Proposed Method [2]

To encode a real-world human character into a virtual space, first, a sequence of two images containing the information of a real-world character is obtained from two input devices: one from a 2D camera such as a webcam, internet protocol camera, or digital camera, and the other from a depth camera such as GameCube or Kinect. A webcam was used as the 2D camera and the Kinect as the depth camera. The Kinect is a motion-sensing input device developed by Microsoft, who provides a software development kit (SDK) designed for Windows-based PCs [20]. The two images are filtered to suppress image noise, and the facial and hand regions of the real-character are extracted from the 2D image, while the color, depth, and skeleton information of the character are extracted from the Kinect data.

A movable bounding box is constructed on both images by calculating the depth information of the real character in the real-world space to define the movement of the character in the virtual space. A moveable bounding box is also formulated in a virtual space to which a real character of a real space is embedded. In this paper, the bounding box of a movable space of a real character (object) in a real space is referred to as a "real character bounding box in a real space" (RCBBRS), and a moveable bounding box of a real character in a virtual space is referred to as a "real character bounding box in a virtual space" (RCBBVS). Here, an RCBBRS is mapped to an RCBBVS, and the real character is composited with a virtual object through texture mapping and rendered in the RCBBVS, resulting in a new 3D virtual space where the real character is presented. The detailed processes and algorithms for obtaining the depth information of real characters in real space, constructing movable bounding boxes, and rendering real characters in 3D virtual spaces can be found in previous studies [1, 2].

3.2. X3D Nodes

We defined and developed four X3D nodes (RCSensingDeviceNode, RCBBRSNode, RCBBVSNode, and RCSpatialMapper) based on the X3D standard format for representing and rendering real-world characters in 3D virtual models, as shown in Table 1.

Table 1. Four X3D Nodes Defined in this Paper

RCSensingDeviceNode{			**RCBBRSNode** {		
SFString[in]	id	""	SFString[in]	id	""
SFString[in]	description	""	SFString[in]	description	""
SFString[in]	type	camera	SFVector[in, out]	startpoint	0 0 8000
SFFloat[in]	fov	45.0	SFVector[in, out]	endpoint	320 240 30000
SFInt[in]	framerate	20	RCSensingDeviceNode[in]	sensingDevice	""
SFImage[out]	image	0 0 0	}		
MFString[in]	jointtype	""			
MFVector[out]	values	0 0 0			
SFBool[in]	usedChromakeying	false }			
RCBBVSNode {			**RCSpatialMapper** {		
SFString[in]	id	""	SFString[in]	id	""
SFString[in]	description	""	SFString[in]	description	""
SFVector[in, out]	startpoint	0 0 1	SFVector[in, out]	realspace	""
SFVector[in, out]	endpoint	1 1 1	SFVector[in, out]	virtualspace	""
SFString[in]	virtualspace	null	SFString[in]	direction	0 0 1
SFVector[in, out]	vsStartpoint	0 0 0	SFVector[in, out]	scale	1 1 1
SFVector[in, out]	vsEndpoint	1 1 1	SFVector[in, out]	up	0 1 0
}			SFString[in]	vcObject	null }

3.2.1. RCSensingDeviceNode: This node sets the sensing information of the input devices. Users can set the properties (features) of their input device, including the type, angle, and frame rate to obtain real-world objects (in this case human beings) from a camera. It has the following properties:

• *type*. This property indicates the type of the input camera. We specify two types of cameras: 2D (designated "camera") and depth cameras ("depthcamera").

• *fov*. This field specifies the viewing angle of the input camera with a default value of 45°.

• *framerate*. This is indicates the number of frames transmitted per second.

• *image*. This designates the image of a real character captured from a camera.

• *jointtype*. This field specifies joint object names on human figures for animating the skeleton. We defined 23 joints of human figures based on humanoid animation (H-Anim) joints [21] and Kinect 20 joints [20] as shown in Table 2. H-Anim is "an abstract representation for modeling three dimensional human figures" [22] to create and animate 3D human figures and to track the motions of real-world human beings in virtual environments [22]. This is the international standard (ISO/IEC 19774) prepared by a number of working groups, including ISO/IEC JTC 1, SC 24, and the H-Anim Working Group of the Web3D Consortium. Microsoft supports the Kinect SDK to provide the tools and drivers required for Kinect-enabled applications for Microsoft Windows [20]. Using the Microsoft SDK Skeleton, users can easily extract the depth information of the skeleton of a real object and perform joint tracking without calibration, which provides several advantages over other open-source tools. The Microsoft SDK enables tracking of 20 joints as shown in Table 2. Figure 3 shows the mapping between the joint object names of the Kinect and those of H-Anim. The joint type *all* selects all of the joint names we defined.

• *values*. This field indicates the joint position coordinates of the joint names specified in the field *jointtype*, and is defined as a vector data type. For example, when the field *jointtype* is specified as "skullbase, l_hand," this is expressed as [skullbase_x, skullbase_y, skullbase_z] [l_hand_x, l_hand_y, l_hand_z].

• *usedChromakeying*. This property designates whether or not the background of a photographed real image is removed. When its value is "true," the background color of a real image is removed using chroma keying techniques [23], as shown on the left side of Figure 4. When the value is "false," a real image is used as it was captured without removing the background, as shown in the right side of Figure 4.

Table 2. Twenty-three Joint Object Names for the Property *Jointtype*

The Kinect	H-Anim	jointtype	The Kinect	H-Anim	jointtype
HEAD	skullbase	skullbase	KNEE_RIGHT	r_knee	r_knee
SHOULDER_CENTER	vc4	c_shoulder	KNEE_LEFT	l_knee	l_knee
SHOULDER_RIGHT	r_shoulder	r_shoulder	ANKLE_RIGHT	r_ankle	r_ankle
SHOULDER_LEFT	l_shoulder	l_shoulder	ANKEL_LEFT	l_ankle	l_ankle
ELBOW_RIGHT	r_elbow	r_elbow	-	r_sternoclavicular	r_sternoclavicular
ELBOW_LEFT	l_elbow	l_elbow	-	l_sternoclavicular	l_sternoclavicular
WRIST_RIGHT	r_wrist	r_wrist	HAND_RIGHT	-	r_hand
WRIST_LEFT	l_wrist	l_wrist	HAND_LEFT	-	l_hand
SPINE	vl1	SPINE	FOOT_RIGHT	-	r_foot
HIP_CNETER	HumanoidRoot	c_hip	FOOT_LEFT	-	l_foot
HIP_RIGHT	r_hip	r_hip	-	-	all
HIP_LEFT	l_hip	l_hip	-	-	-

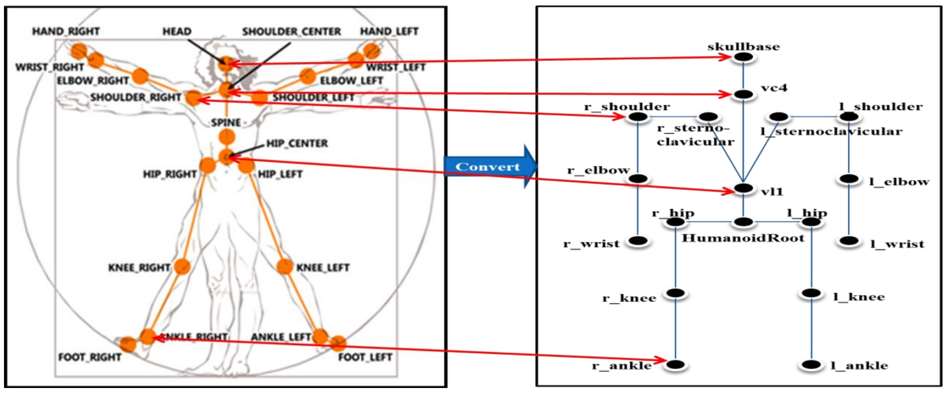

Figure 3. Mapping the Joint Object Names of the Kinect and H-Anim

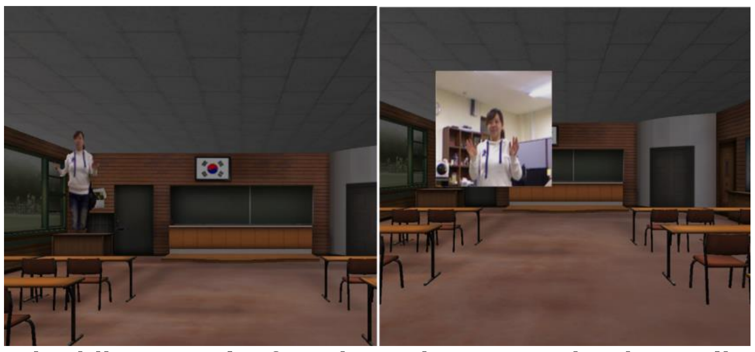

Figure 4. Embedding Results for when Chroma Keying is Applied to a Real Image (left), or Not Applied (right)

3.2.2. RCBBRSNode: This node generates a movable bounding box of the real character in real spaces. A movable bounding box has two corner points: a start point, which is the minimum volume of the bounding box, and an end point, which is the maximum volume of the bounding box, as shown in Figure 5. The description of its properties is as follows.

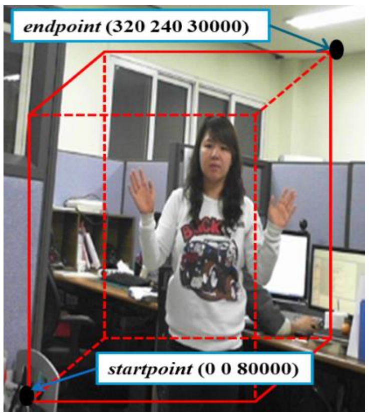

Figure 5. Example of Setting up a Movable Bounding Box of a Real Character in a Real-world Space

• *startpoint*. This field specifies the minimum Cartesian coordinate of the movable bounding box of a real character in a real-world space. It is defined using Cartesian coordinates and its default value is (0, 0, 8000). This *startpoint* will be mapped to that of the movable bounding box of the character in a virtual space.

• *endpoint*. This field indicates the maximum coordinate of the movable bounding box of a real character in a real space. Its default value is (320, 240, 30000). In the same way as *startpoint*, this *endpoint* is mapped to that of the movable bounding box of a real character in a virtual space. The range of values of the X, Y, and Z coordinates are 0–640, 0–480, and 6000–40000, respectively.

• *sensingDevice*. This specifies the input sensing device, *id*, to obtain and load the settings of the *id*.

3.2.3. RCBBVSNode: This node generates a movable bounding box of a real character in the 3D virtual space. A bounding box has two corner points: a *startpoint* and an *endpoint,* as for RCBBRSNode. The volume of RCBBVS is generated in accordance with the ratio of two points in RCBBRS. The *startpoint* and *endpoint* of RCBBRS is mapped to those of RCBBVS. Figure 6 shows a movable bounding box of a real character in a virtual space.

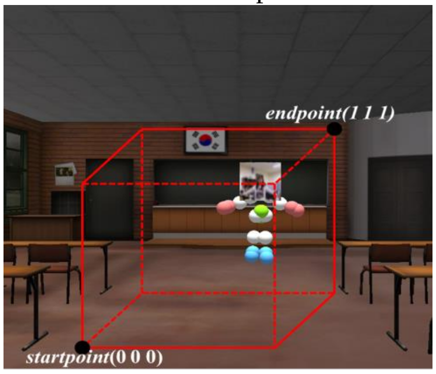

Figure 6. A Movable Bounding Box of a Real Character in a Virtual Space

• *startpoint*. This field specifies the minimum coordinate of the movable bounding box of a real character in a virtual space. The *startpoint* in RCBBRS is mapped to this *startpoint*. The position of a movable bounding box in a virtual space (i.e., RCBBVS) is relative to that of RCBBRS; that is, the *startpoint* of RCBBVS is generated in accordance with the ratio of two points in RCBBRS. Consequently, it will be (0 0 0) when the *startpoint* of RCBBRS has the default value (0, 0, 8000).

• *endpoint*. This designates the maximum coordinate of the movable bounding box of a real character in a virtual space. The *endpoint* of RCBBRS is mapped to this *endpoint*.

• *virtualspace*. This indicates the file name of a 3D virtual model (i.e., scene file) to load as a virtual space from the database for 3D virtual models. The supported file formats are .3ds, .fbx, and .obj.

• *vsStartPoint*. This specifies the minimum value of the range of a movable space of a bounding box in a virtual space. This is represented as an (X, Y, Z) coordinate calculated using the following formula:

(1) $vsStartPoint = (startpoint.x*(x2-x1)+x1,\ startpoint.y*(y2-y1)+y1,\ startpoint.z*(z2-z1)+z1)$

• *vsEndPoint*. This field indicates the maximum range of a movable space of a bounding box in a virtual space. This is obtained from:

(2) $vsEndPoint = (endpoint.x*(x2-x1)+x1,\ endpoint.y*(y2-y1)+y1,\ endpoint.z*(z2-z1)+z1)$

where the minimum value of a real character is (x1, y1, z1) and the maximum is (x2, y2, z2).

Figure 7 shows an example of the coordinates of the *startpoint*, *endpoint*, *vsStartPoint*, and *vsStartPoint* where the minimum value (x1, y1, z1) of a real character in a real-world space is (-230, -150, -350) and the maximum value (x2, y2, z2) is (400, 300, 450).

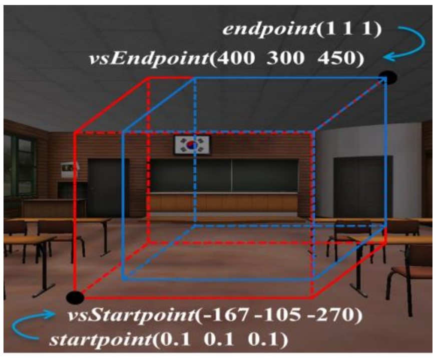

Figure 7. Range of a Movable Space of a Bounding Box in a Virtual Space

3.2.4. RCSpatialMapper: This node renders real characters in 3D virtual models. If an RCBBRS is mapped to an RCBBVS, the real-world character in the RCBBRS can be embedded in the RCBBVS of the virtual space, while tracking a sequence of the continuous movement of the real character in the virtual space in real-time. The real character is represented in the RCBBVS of the virtual space by texture mapping with a virtual object (*vcObject*). The direction and scale of the texture-mapped real character are changeable and a virtual object can also be replaced with other virtual objects.

• *realspace*. This field specifies the *id* of an RCBBRSNode to be loaded as the bounding box of a real character in a real-world space.

• *virtualspace*. This field specifies the *id* of an RCBBVSNode to be loaded as the bounding box of a real character in a virtual space.

• *direction*. This field indicates the direction of a real character in a virtual space. It is represented by Cartesian coordinates with a default value of (0 0 1). With the default value, the real character is facing forward as shown on the left side of Figure 8. If we change this from (0 0 1) to (1 0 1), the direction of the real character is changed as shown on the right side of Figure 8.

• *scale*. This field indicates the scaling of a real character when mapped to the virtual space. It is represented by Cartesian coordinates and has the default value (1 1 1). Figure 9 illustrates a change in the scale of a real character from (1 1 1) to (2 1 1).

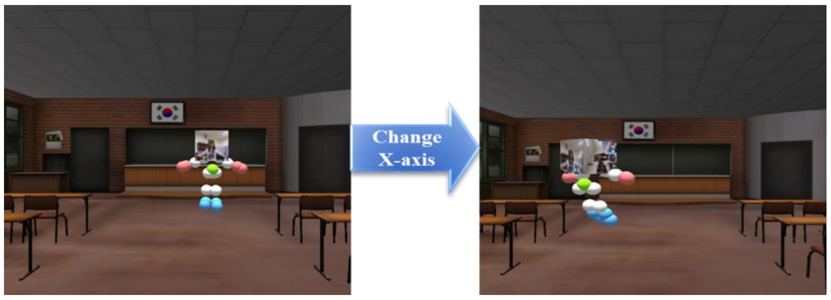

Figure 8. Embedding Examples with Direction (0 0 1) (left) and (1 0 1) (right)

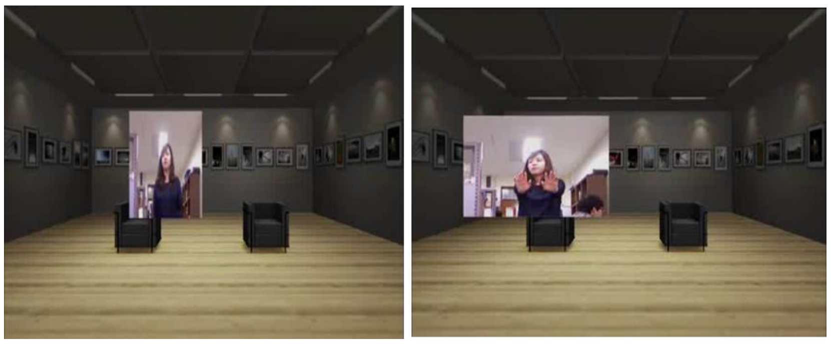
Figure 9. Embedding Examples with Scale (1 1 1) (left) and (2 1 1) (right)

• *up*. This specifies the direction of the head of a real character. When it is (0 1 0), the head direction of a real character is on top.

• *vcObject*. This field indicates a virtual object (*vcObject*) to be added to the real character image through texture mapping. In this paper, we present two different types of virtual objects: person.3ds as shown in the left side of Figure 10 and plane.3ds, shown on the right side. Users can define their own 3D models using the .3ds, .fbx, and .obj file formats.

Figure 11 shows rendering examples of a real character in a 3D virtual model through the bounding box mapping between RCBBRS and RCBBVS, and texture mapping between the real character and the virtual object. The texture-mapped real character will move in the virtual space, tracking the movement of the real character in the real space in real-time.

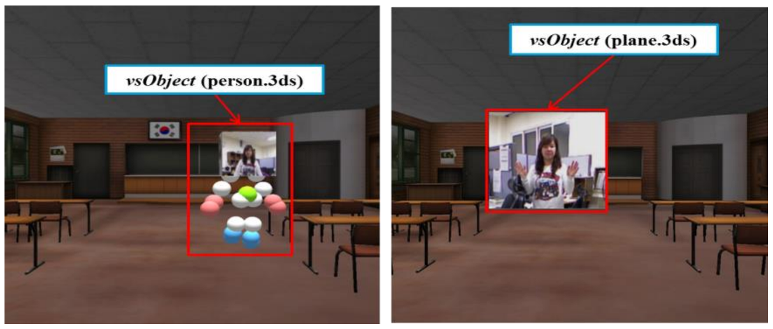

Figure 10. Texture Mapping Examples with Two Virtual Objects

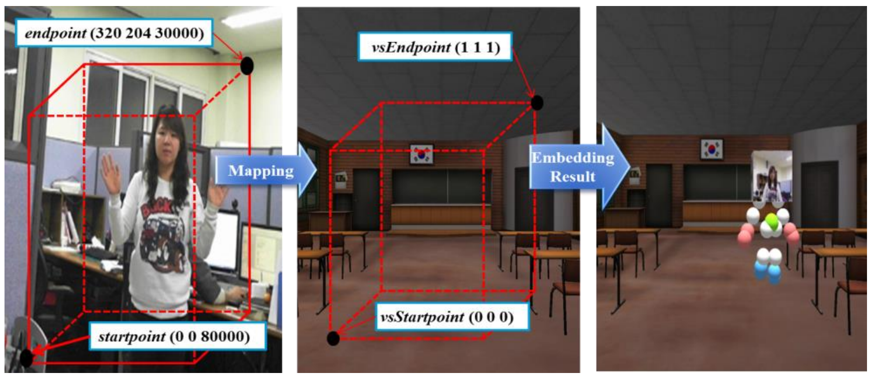

Figure 11. Rendering Results of a Real Charter in a 3D Virtual Space

4. Experimental Results

A number of experiments rendering real characters in a virtual space using the proposed X3D nodes were carried out to evaluate the effectiveness of the system. The examples were developed using Unity 3.415t, Microsoft Visual Studio 2008 and 2010 on Windows 7 using an Intel Core i5-2500 3.30 GHz Quad-Core processor and an Nvidia GeForce GTX 550 Ti graphics card.

Table 3 shows the X3D source code for representing a real-world character captured through a 2D camera (i.e., a webcam) using the four X3D nodes defined in Section 3. First, a continuous sequence of movements of the real human being is captured using the webcam (cam0) as defined in the node RCSensingDeviceNode in Table 3. The real character is encoded without removing the background (i.e., *usedChromakeying* = false). Second, as indicated in the node RCSpatialMapper, the *startpoint* and *endpoint* of RCBBRS (bbrs1) is mapped to those of RCBBVS (bbvs1), and then the real character is embedded in the RCBBVS of the virtual space, classroom.3ds, through texture mapping with the virtual object plane.3ds. The properties of the RCBBRS and RCBBVS are specified in the nodes RCBBRSNode and RCBBVSNode, respectively. Through these mapping processes, a real character in a real-world space is rendered into a virtual space, simulating the movement of the real-world character in real-time. Figure 12 shows the rendering results using the X3D source code shown in Table 3 with a number of screenshots captured from a sequence of the continuous movements of the real character in the virtual space classroom.3ds.

Table 3. X3D Source Code for Rendering a Real Character Captured using a 2D Camera

```
<RCSensingDeviceNode id = “cam0” type = “camera” fov = “50” framerate = “30” usedChroma-
keying = “false” jointtype = “all”> </RCSensingDeviceNode>
<RCBBRSNode id = “bbrs1” description = “a movable space of a real character in a real space”
    startpoint = “0 0 8000” endpoint = “640 480 30000” sensingDevice = “cam0”> </RCBBRSNode>
<RCBBVSNode id = “bbvs1” description = “a movable space of a real character in a virtual space”
    startpoint = “0 0 0” endpoint = “1 1 1” virtualspace = “classroom.3ds”>  </RCBBVSNode>
<RCSpatialMapper id = “rcsp1” realspace  = “bbrs1” virtualspace = “bbvs1” direction = “0 0 1”
    scale = “1 1 1” vcObject = plane.3ds”> </RCSpatialMapper>
```

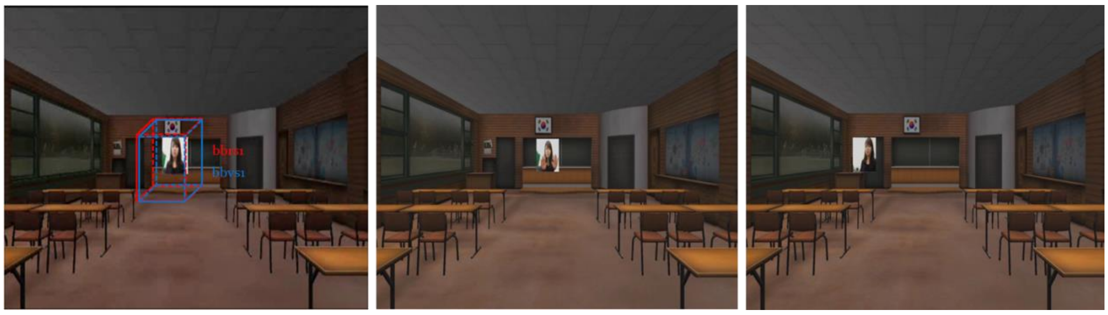

Figure 12. Rendering Results for the X3D Codes in Table 3

Table 4 shows the X3D source code for embedding a real-world character captured using a depth camera (i.e., the Kinect). In this example, the real character data from the depth camera is embedded in the RCBBVS of the virtual space artRoom.3ds through texture mapping with the virtual object person.3ds. Figure 13 shows the rendering results with a number of screenshots captured from a sequence of the continuous movement of the texture mapped real character in the virtual space artRoom.3ds.

Table 4. X3D Source Code for Representing a Real Character Captured using a Depth Camera

```
<RCSensingDeviceNode id = "depthcam1" type = "depthcamera" fov = "50" framerate = "30"
   usedChromakeying = "false" jointtype = "all" </RCSensingDeviceNode>
<RCBBRSNode id = "bbrs2" description = "a movable space of a real character in a real space"
   startpoint = "0 0 8000" endpoint = "640 480 30000" sensingDevice = "depthcam1">
   </RCBBRSNode>
<RCBBVSNode id = "bbvs2" description = "a movable space of a real character in a virtual space"
   startpoint = "0 0 0" endpoint = "1 1 1" virtualspace = "artRoom.3ds"> </RCBBVSNode>
<RCSpatialMapper id = "rcsp2" realspace = "bbrs2" virtualspace = "bbvs2" direction = "0 0 1"
   scale = "1 1 1" vcObject = "person.3ds"> </RCSpatialMapper>
```

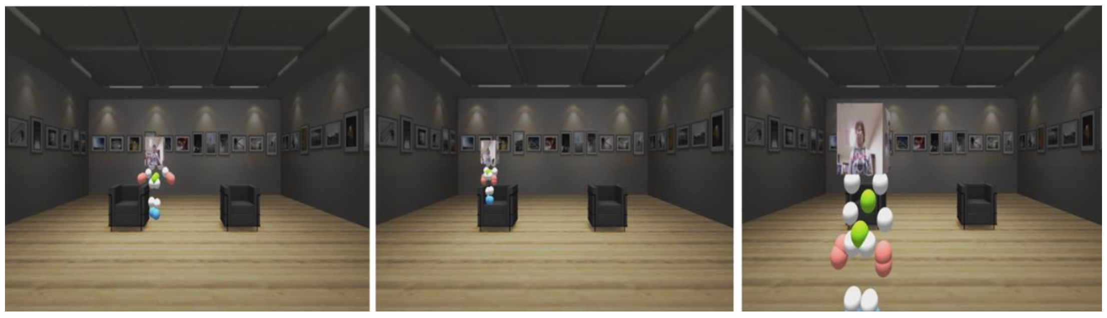

Figure 13. Rendering Results for the X3D Codes in Table 4

Figure 14 shows the embedding results corresponding to the X3D source code shown in Table 3. Note that the *startpoint* of RCBBRSNode was modified from (0 0 8000) to (0 0 0) and the virtual object was changed from plane.3ds to person.3ds; in addition, the *direction* and *scale* of the real character were changed from (0 0 1) to (-1 0 1), and from (1 1 1) to (1.5 1 1), respectively.

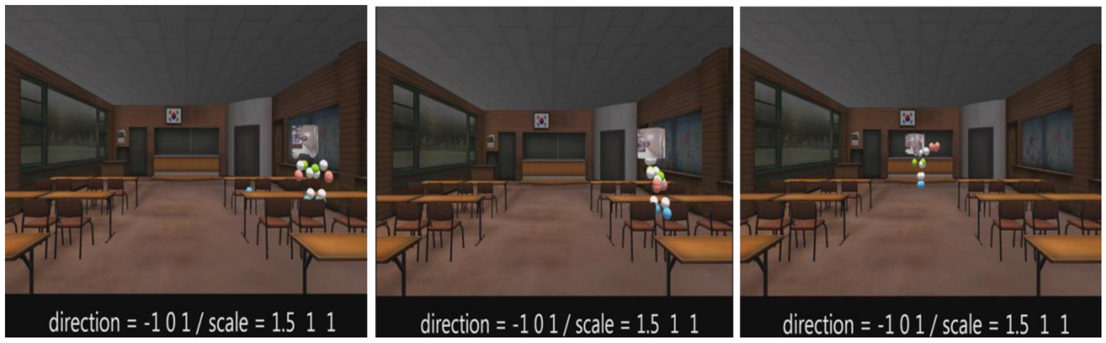
direction = -1 0 1 / scale = 1.5 1 1
direction = -1 0 1 / scale = 1.5 1 1
direction = -1 0 1 / scale = 1.5 1 1

Figure 14. Embedding Results of the X3D Source Code Shown in Table 3 where the Start Point, Direction, and Scale of the Virtual Object were Changed

Figure 15 shows the rendered results of embedding of a real character in a virtual space, while tracking continuous movement. Here, the background color of the photographed real image was removed using chroma keying (i.e., *usedChromakeing* = true) and the *startpoint* and *endpoint* of the RCBBVS in the node RCBBVSNode was changed from (0 0 0) to (0 0 0.2) and (1 1 1) to (1 1 0.8), respectively.

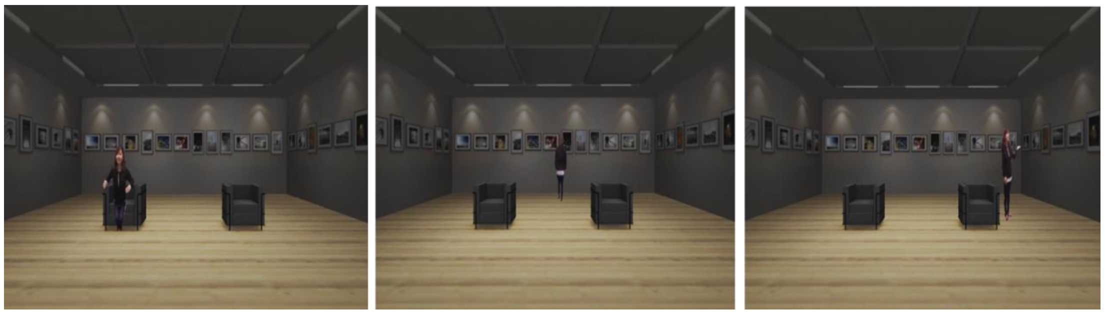

Figure 15. Embedding Results for when the Start Point, Direction, and Scale of the Virtual Object were Changed

5. Conclusion

This paper has described four X3D nodes for representing and rendering real characters in 3D virtual models in real-time. The objective of this paper was to develop a standardized implementation that conforms to the X3D format for real character rendering in virtual spaces as an extension of the X3D core nodes [6]. To evaluate the effectiveness of these X3D nodes, several application examples were investigated. The results show that the settings and features we defined in the four X3D nodes were sufficient to obtain sensing information of an input device, generating a movable bounding box of the real character both in real space and in the virtual space, and to synthesize and render the real character in the virtual model. This demonstrates the feasibility of the four X3D nodes for rendering real characters in 3D virtual spaces.

Further development of more realistic 3D virtual models is required to implement more immersive virtual e-learning environments. The development of more advanced techniques for recognizing gestures of real characters is also required. In addition, the development of techniques that enable the augmentation of virtual spaces based on these gestures would be a valuable enhancement of current VR and MR environments.

Acknowledgements

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (NIPA-2014-H0301-14-1022) supervised by the NIPA (National IT Industry Promotion Agency), and by the Ministry of Education(MOE) and National Research Foundation of Korea(NRF) through the Human Resource Training Projection for Regional Innovation.

References

[1] J. S. Jeong, J. A. Park, S. A. Kwon, C. Park, N. Baek and K. H. Yoo, “A Reference Model for Representing Real Characters into the 3D Virtual Space”, Information Science and Tehcnology, vol. 1, no. 1, **(2011)**.

[2] J. S. Jeong, J. A. Park, S. A. Kwon, C. Park, N. Baek, R. H. Jang and K. H. Yoo, “An Embedding Method for Real Characters into the 3D Virtual Space”, International Journal of Software Engineering and Its Applications, vol. 7, no. 1, **(2013)**, pp. 69-78.

[3] J. S. Jeong, C. Park, J. J. Han, M. S. Im, R. H. Jang, M. Kim and K. H. Yoo, “Development of a 3D Virtual Studio System for Experiential Learning”, CCIS, vol. 195, **(2011)**, pp. 78-87.

[4] M. Kim, J. S. Jeong, C. Park, R.H. Jang and K. H. Yoo, “A Situated Experimental Learning System Based on a Real-Time 3D Virtual Studio”, LNAI, vol. 7457, **(2012)**, pp. 364-371.

[5] S. Stieglitz, C. Lattermann and M. Kallischnigg, “Experiential Learning in Virtual Worlds – A Case Study for Entrepreneurial Training”, Proceedings of the 16th Americas Conference on Information Systems, **(2010)**; Lima, Peru.

[6] Web3D Consortium, http://www.web3d.org/, http://www.web3d.org/realtime-3d/specification/all, http://www.web3d.org/files/specifications/19775-1/V3.3/index.html **(2013)**.
[7] P. Milgram and S. Zhai, "Applications of augmented reality for human-robot communications", Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, **(1993)**; Yokohama, Japan.
[8] R. Silva, J. C. Oliveira and G. A. Giraldi, "Introduction to Augmented Reality", Technical Report, LNCC, **(2003)**.
[9] B. Parhizkar, A. A. M. Modwahi, A. H. Lashkari, M. M. Bartaripou and H. R. Babae, "A Survey on Web-based AR Applications. International Journal of Computer Sicence, vol. 8, no. 1, **(2011)**, pp. 471-479.
[10] R. T. Azuma, "A survey of augmented reality", Presence: Teleoperators and Virtual Environment, vol. 6, no. 4, **(1997)**, pp. 355-385.
[11] R. T. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier and B. MacIntyre, "Recent Advances in Augmented Reality", IEEE Computer Graphics and Application, vol. 25, no. 6, **(2001)**, pp. 34-47.
[12] P. Milgram and F. A. Kishino, "Taxonomy of mixed reality visual display", IEICE Transactions on Information and Systems (Special Issue on Networked Reality), E77-D (12), **(1994)**, pp. 1321-1329.
[13] H. Ishii and B. Ullmer, "Tangible bits: Towards seamless interfaces between people, bits and atoms", Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems, **(1997)**; Atlanta, GA, USA.
[14] B. K. Kye, J. H. Kim and J. H. Ryu, "Pedagogical Understanding of Augmented Reality", KERIS Issue Report RM, Korea Education and Research Information Service, **(2007)**.
[15] B. E. Shelton, "How augmented reality helps students learn dynamic spatial relationships", Doctoral Dissertation, University of Washington, **(2003)**.
[16] J. H. Ryu, I. H. Jo, H. O. Heo, J. H. Kim, B. K. Kye and B. S. Go, "The Next Generation of Learning Model for Augmented Reality Enhanced in Tangible Interface", Research Report CR, Korea Education and Research Information Service, **(2007)**.
[17] J. J. Hong, M. Kim and K. H. Yoo, "Development of a 3D Digital Textbook Using X3D", LNEE, vol. 214, **(2012)**, pp. 341-351.
[18] Web3D Consortium, What is X3D?, http://www.web3d.org/realtime-3d/x3d/what-x3d, **(2013)**.
[19] X3D, Wikipedia, http://en.wikipedia.org/wiki/X3D, **(2013)**.
[20] Microsoft Corporation, Kinect for Windows & Kinect for Windows SDK Beta 1 Programming Guide Version, http://www.microsoft.com/en-us/kinectforwindows/, Kinect for Windows SDK, http://msdn.microsoft.com/en-us/library/hh855347.aspx, **(2013)**.
[21] Web3D Consortium, Humanoid Animation (H-Anim), http://www.web3d.org/realtime-3d/working-groups/h-anim, **(2013)**.
[22] Web3D Consortium, Humanoid Animation (H-Anim): Foreword, Introduction. http://www.web3d.org/x3d/specifications/ISO-IEC-19774-HumanoidAnimation/, **(2013)**.
[23] Chroma Keying, http://www.mediacollege.com/glossary/c/chroma-key.html, **(2013)**.

Authors

In-Kwon Kim. He received his B.S. in the Department of Computer Education at Chungbuk National University, Korea in 2013, and is a M.S. student in Department of Digital Informatics and Convergence at Chungbuk National University, Korea from 2014. Her research interests include computer graphics, e-learning, 3D character animation, and multimedia.

Sin-Ae Kwon. She received her M.S. in Computer Education Division, at Chungbuk National University, Korea in 2013. Her research interests include computer graphics, e-learning, 3D character animation, and multimedia.

Mihye Kim. She received her Ph.D. degree in Computer Science and Engineering from New South Wales University, Sydney, Australia in 2003. She is currently an Associate Professor in the Department of Computer Science Education at Catholic University of Daegu, South Korea. Her research interests include knowledge management and retrieval, computer science education, digital textbooks, and cloud computing.

Kwan-Hee Yoo. He is a professor in the Department of Software Engineering at Chungbuk National University, Korea. He received his B.S. in Computer Science from Chonbuk National University in 1985, and his M.S. and Ph.D. in computer science from KAIST (Korea Advanced Institute of Science and Technology) in 1988 and 1995, respectively. His research interests include computer graphics, 3D character animation, and dental/medical applications.