SVE에서 절대위치를 같는 오브젝트에 변형의 적용

정동현[O] 이창섭 Larry F. Hodge[†] 송창근
한림대학교 컴퓨터공학과
Dept. of Computer Science, Univ. of North Carolina at charlotte[†]
{dhjeong[O], cslee, cgsong}@hallym.ac.kr, lfhodges@ucc.edu[†]

Applying Transformation to the Absolutely Positioned Object in SVE

Dong Hyun Jeong[O] Chang Sub Lee Larry F. Hodge[†] Chang Geun Song
Dept. of Computer Science, Univ. of North Carolina at charlotte [†]
Dept. of Computer Engineering, Hallym University

Abstract

The Simple Virtual Environment (SVE) library is the application programming interface for creating the virtual environment easily. Even though it has a lot of efficient features, applying transformation to absolutely positioned objects is difficult. In this paper, we designed a simple method with which it is possible to rotate the absolutely positioned object freely. To test the method, we designed a simple virtual environment. The environment is designed with people, street and building objects. To reduce time consumption of displaying high-detailed people models, we only used texture-based objects. The person objects are modeled as quadric-shape and textured with front-view images of people. To make more realistic environment, objects always change its orientation themselves following to the user location. The result and the testing environment will be demonstrated.

1. Introduction

SVE library is one of the most broadly used virtual environment libraries. The first version of SVE originally designed at Georgia Tech. in 1995. With the libraries, a lot of research results have been published. While using the SVE library, we usually load object files to the virtual environment. Loading object files is one of the priori works. After loading objects, we can position those objects to somewhere in the environment. But positioning objects is not the easiest easy way as we wish. The reason of that is most of objects have their own pivot and center position. In the case of using the absolutely position object designed by using 3rd party modeler, it has its own position information. If we load the object to the environment, the object will be somewhere with its own position information. But the problem is that there is no method for gathering the position information in the SVE. Therefore, if we add transformation methods to the object, the object will be transformed depending on not the SVE structure information (SVE_object) but its original location information.

To solve the problem, we designed a simple method. The method references the SVE object structure and the object position information. To test the designed method, we used several virtual environments. In this paper, we defined some definitions. If the object's pivot value is different from the world origin position, we call the object as the absolutely positioned object otherwise the relatively positioned object.

This work was supported by the Korea Science and Engineering Foundation (KOSEF)

In the following section, we review previous related researches. In later section, we will briefly explain and show the designed method and the virtual environment. Our conclusion will be followed.

2. Previous work

A lot of researchers have designed several virtual environment toolkits or libraries. Most of them have their own efficient and effective features for designing the virtual environment. In this paper, we focused on using the SVE library. The SVE library designed for the purpose of designing the virtual environment easily [1]. Also many people use the library for making the virtual environment or evaluating some experiments. Melder and at al presented some methods for figuring out the contact friction in situations where an object is being both translated and rotated [2]. Ware and Rose researched translational positioning in virtual environment using real-hands [3].

3. The designed method

Most of objects have their own pivot and center position. To model 3D objects, people use 3rd party object modeling software such as 3DS-MAX, MAYA and so on. After generating the objects, the SVE library loads the objects to make virtual environments. To maintain the position and orientation, the SVE keeps the information of the object to the structure called SVE_object. In the SVE library, when a new object is loaded to the virtual environment, the SVE_object structure is set with default values. Therefore, if the pivot or center position of object

is different, it will make some problems while applying some transformation to objects. The reason of the problem is that the SVE cannot know that the object has different position.

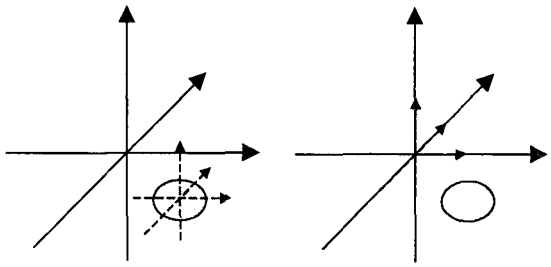


Figure 1. Correct (left) and Incorrect Positioning (right) in the SVE library.

Figure 1 shows object center position depending on the world coordinate system. If the object is positioned in absolute location, there will be some mismatch like the right image in Figure 1. Even though the object is positioned in somewhere of the environment, the SVE_object only has the same orientation information with world coordinate system. If we apply the transformation to the absolutely positioned object (right image in Figure 1), the object will be transformed differently. To make working correctly, the object's center position must be the same with world origin position.

As mentioned above, absolutely or relatively positioned objects have different object center position. If the object has the relative position, it probably works properly because the object center position is the same with the position information of the SVE_object.

Table 1. Wavefront .obj file description & top view in 3DS-MAX.

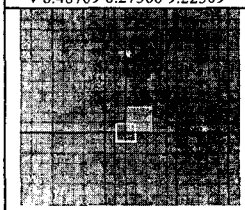
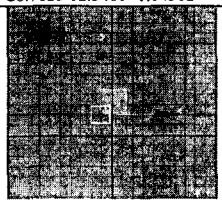
Relatively positioned object	Absolutely positioned object
# object Box01	# object Box01
g Box01	g Box01
v -8.48709 -6.27306 -9.22509	v 11.8081 -1.40328e-006 -20.2952
v -8.48709 -6.27306 9.22509	v 11.8081 -5.96798e-007 -1.84502
v 8.48709 -6.27306 -9.22509	v 28.7823 -2.14525e-006 -20.2952
v 8.48709 -6.27306 9.22509	v 28.7823 -1.33876e-006 -1.84502
v -8.48709 6.27306 -9.22509	v 11.8081 12.5461 -20.2952
v -8.48709 6.27306 9.22509	v 11.8081 12.5461 -1.84502
v 8.48709 6.27306 -9.22509	v 28.7823 12.5461 -20.2952
v 8.48709 6.27306 9.22509	v 28.7823 12.5461 -1.84502
 Top view	 Top view

Table 1 shows files descriptions of the absolutely and relatively positioned objects. The difference of two objects is that they are positioned in different location of the environment because they have their own position.

The pivot information can be easily found while we are using the 3rd party object modeling tools. The pivot has an object coordinate system. Therefore, if we apply some transformation to the object, the object will be transformed depending on the pivot coordinate. Initially the pivot position is the same with object center position when it is generated. But the pivot position can be changed. Adapting some transformation to the object which has modified pivot position is quite useful for generating free-style models. But in the SVE system, the pivot and object center position must be the same coordinate information because there is no known. The main reason of the mismatch is that the SVE supported file format (.obj file) does not have any pivot information. Whenever an object is loaded to the environment, SVE always sets the pivot information with default values even if is has to have different values. So, applying transformations can make different results. The position (pivot) information has a form of 4x4 matrix.

Table 2. The initial 4x4 matrix form in the SVE_object.

Relatively positioned object	Absolutely positioned object
$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \alpha & \beta & \gamma & 1 \end{bmatrix}$

Table 2 shows that the initial matrix (left matrix form) when the object is loaded to the virtual environment. In the case of using the absolutely positioned object, the matrix has to have some values such as α, β, and γ because they indicate the initial position of the object. But in the SVE library, the three variables have the value of zero as a default value. Therefore, the absolutely positioned object is working like the relatively positioned object even if the initial position is different. Our designed method is used for compensating the difference values.

The method is simple. First, it creates an empty object and positioned the object to the exactly same center position of the absolutely positioned object. And then apply some transformation to the empty object. After applying, we can extract the changed matrix from the empty object and again apply the values to the original object.

To show the efficiency, we designed the virtual environment adapting the method. The environment is designed with people, street and building objects. People objects are positioned absolutely. And to reduce time consumption of displaying high-detailed people models, we only used texture-based objects. The person objects are modeled as quadric-shape and textured with front-view images of people. By adapting the method, people objects always change their directions toward the user whenever user moves in the virtual environment.

Table 3. A simple example of applying rotation to the absolutely positioned object.

```
float getDegrees(float &dCosTheta, float dSinTheta, bool bAdding)
{ // change radians to degrees
   float bMomentvalue = 0;
   if (bAdding == true)
```

```
// additional moment value for showing front view of model to the user.
   bMomentvalue = 90;

   if (dSinTheta > 0)
      return (float)(-1. * fabs(acos(dCosTheta)*RADS2DEGS) - bMomentvalue);
   else if (dSinTheta < 0)
      return (float)(1. * fabs(acos(dCosTheta)*RADS2DEGS) + bMomentvalue);
   else
return 0;
}

void rotateObject(char *ObjName, float &rotationValue, bool bAdding)
{// apply rotation depending on the empty object
   SVE_object obj, Posobj;
   SVE_point centerPos;
   M_matrix directionMatrix;

   obj = SVE_findWorldObject(ObjName);
   Posobj = SVE_createEmptyObject("Ghost01");   // create empty object
   SVE_getObjectCenter(obj, centerPos); // get object center position
   // move empty object to real object position
   SVE_moveObject(Posobj, centerPos);
   // move direction depending on the user
   SVE_lookAt(Posobj, user, TRUE);
   SVE_getWorldMatrix(Posobj, directionMatrix);

   // calculate theta value (degree)
   float MovingDirection = (float) getDegrees(directionMatrix[0][0],
directionMatrix[0][2], bAdding);

   // rotate back if there is previous rotation applied
   SVE_rotateWRT(obj, Posobj, (float)-1*rotationValue, 'y');
   // rotate object based on degrees
   SVE_rotateWRT(obj, Posobj, MovingDirection, 'y');
   rotationValue = MovingDirection;        // save applied value temporary
   SVE_deleteObject(Posobj, NULL); // delete empty object
   }
```

Table 3 shows that the absolutely positioned object always changes its orientation via y-axis and always looking at the user object.

The rotateObject function has three parameters such as object name, rotation value and additional flag. The rotation value is used for changing the original position or direction of the absolutely positioned object because whenever rotation function (SVE_rotateWRT) is used, viewing matrix value will be accumulated; therefore if we want to apply new rotation value acquired from the transformation matrix, it has be changed to the initial position. Otherwise the object will be rotated continuously because of the accumulated value. Therefore before applying the newly measured value, the initial position and orientation of the absolutely positioned object have to be retrieved. To get the rotation degree and position values, we used the transformation matrix.

In our example, we only focused on using the y-axis rotation. Therefore x and z-axis rotations are not concerned. The matrix representation for y-axis rotation is :

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

From the cosine and sine functions, we can extract rotation value (theta). With the value, we can apply transformation to the object easily. Also there is an important reminder that the additional moment value is added to the theta value because we do not know whether the pivot z-axis is perpendicular to user or not. The moment value must be indicated while adapting those functions manually.

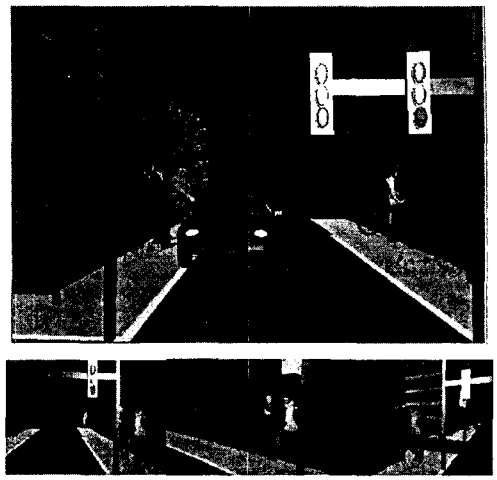


Figure 2. Examples of the virtual environment.

In figure 2, there are people objects shown on the screen. The objects are all absolutely positioned objects. And the objects are changing its direction following the user position correctly.

4. Conclusions

The SVE library is one of the most useful virtual environment libraries. Even though the SVE library has a lot of pre-defined functions, features and hard-ward support, it has a small problem while applying transformations to the absolutely positioned object. In this paper, we designed a simple method for applying transformation to the absolutely positioned object. The idea is a simple and efficient. It uses the empty object before applying the transformation to the object. To apply the method, we designed the virtual environment adapting the method.

For the future work, it is recommended modifying or adding the designed method to the SVE library. Therefore people can easily apply transformation to the object.

5. References

[1] G. Drew Kessler, Doug A. Bowman, and Larry F. Hodges, "The Simple Virtual Environment Library: an Extensible Framework for Building VE Applications", PRESENCE, MIT Press, 2000

[2] N. Melder, W. S. Harwin and P. M. Sharkey, "Translation and Rotation of Multi-Point Contacted Virtual Objects", Proceedings of Eurohaptics Conference pp 218-227, 2003.

[3] Colin Ware and Jeff Rose, "Rotating Virtual Objects with Real Handles", In ACM Transactions on Computer Human Interaction(TOCHI), June, Vol. 6, No. 2, pp. 162-180, 1999.

[4] Colin Ware and Roland Arsenault, "Frames of Reference in Virtual Object Rotation", Proceedings of the 1st Symposium on Applied perception in graphics and visualization, pp. 135-141, 2004.