ARTransform: Visualization of Three Dimensional Geometric Transformations in Augmented Reality Environment

Kah Pin Ng, Guat Yew Tan

School of Mathematical Sciences, Universiti Sains Malaysia, Penang, Malaysia

Abstract - *Three dimensional (3D) geometric transformations are basic and important skills when working in computer graphics related areas. Traditional ways in learning the skills include forcing the 3D results to be demonstrated on 2D output media thus challenge the imaginations of the viewers. This paper presents ARTransform – an augmented reality (AR) application for 3D geometric transformations analysis. ARTransform animates 3D geometric transformations in real environment, which enables the viewers to move around the transforming objects and make closer observations. Graphical user interface control panels are developed to provide user-specified geometric transformations and animations. ARTransform adopts the concept of keeping the learning curve gentle to enable the viewers to spend more time in transformation analysis, thus, user interaction is achieved by conventional input devices via keyboard and mouse. ARTransform is still at its infancy and much work remains to be done. User-test on the application will be carried out when more features are added to the application.*

Keywords: Augmented Reality, Geometric Transformation

1 Motivation and Related Work

Geometric transformation is an important component in computer graphics related areas, such as animation, computer vision, image processing, geometric modeling, pattern recognition, etc. It deals with changing the geometric descriptions of an object in terms of position, orientation or size, and adopts the nature of affine geometry where the object's properties remain unchanged after the transformation [1]. The traditional way to view the geometric transformation is paper-based, i.e. using pen and paper to draw and animate the transformed object with the imagination of the viewer. Nowadays, more common and sophisticated ways are using virtual reality (VR) and computer animation to create motion parallax especially while transforming a three-dimensional (3D) object, to achieve the depth illusion. Geometric transformation functions are the most basic functions available in almost all well-known graphics packages, e.g. Mathematica, MATLAB, AutoCAD, 3D Studio Max, Maya, Lightwave 3D, etc. The OpenGL tutorial developed by Nate Robin animates OpenGL's geometric transformation functions by rolling the mouse over the functions parameters on the screen to change the position or shape of the object's local axes [2]. However, examples mentioned above display their 3D outputs on a two-dimensional (2D) screen. One major limitation is that, only the front face of the object can be seen, the back of the object can be analyzed visually only by "turning" the object with a keyboard/mouse interactively. An alternative is to display the object in wireframe view on the computer screen, but the wireframe creates too many lines which will confuse the viewer eventually. In consideration of the drawbacks to represent 3D objects on 2D devices, researchers have been actively working towards a solution for 3D object representation in 3D environment, and this is where Augmented Reality (AR) becomes the overwhelming research focus.

The ability of AR to bring the 3D objects into real environment has made it a useful tool in spatial skill training [3]. Construct3D developed in Vienna University of Technology focuses on the education in mathematical and geometry construction process to high school students [4], University of Washington uses AR to teach their third year students earth-sun relationships and found significant overall improvements in students' understandings after the AR exercises [5].

In this paper, we propose an application called ARTransform to visualize the spatial geometric transformation by taking the advantage of viewer's mobility offered by AR to animate the object transformations in the real world. By moving freely in the room, the viewer can see and analyze step-by-step changes in orientation, shape, size and position of the transforming object. The project was inspired while teaching Computer Graphics course to our undergraduate students by using OpenGL's geometric transformation functions. Confusions were always caused by the 3D geometric transformations especially when involving a series of scaling and rotation. By displaying the animated transformation in AR, we believe that the viewer's mobility will help him/her achieve the best learning results.

2 Geometric Transformation Overview

In general, geometric transformation can be classified into two categories, viewing and modeling transformations which deal with changing of the camera/viewer's positions and changing of the object's position respectively [6]. The first version of our project focuses on modeling transformation. The modeling transformation can be expressed as

$$\mathbf{P}' = \mathbf{M}\cdot\mathbf{P}\ . \qquad (1)$$

where **P**' denotes the transformed coordinates after composite transformation **M** applied to the original coordinates, **P**. The composite transformation is formed by,

$$\mathbf{M} = \mathbf{M}_n\cdot\mathbf{M}_{n-1}\cdot\mathbf{M}_{n-2}\cdot \ldots \cdot\mathbf{M}_3\cdot\mathbf{M}_2\cdot\mathbf{M}_1\ . \qquad (2)$$

where $\mathbf{M}_k$, $\boldsymbol{k} \in \{ 1, 2, \ldots, n\}$, denotes individual geometric transformation. The transformation sequence is written in reverse order, i.e. from right to left, being $\mathbf{M}_1$ and $\mathbf{M}_n$ the first and the last transformations respectively, to be applied to an object. Thus, if an object is rotated $\theta°$ anti-clockwise about a space line parallel to the z-axis, with equations $x=x_r$ and $y=y_r$; then it is scaled to (s_x, s_y, s_z) of its original size, by referring to a fixed point (x_f, y_f, z_f), finally translated (t_x, t_y, t_z) units; the transformation sequence is

$$\mathbf{T}(t_x, t_y, t_z) \cdot \mathbf{T}(x_f, y_f, z_f) \cdot \mathbf{S}(s_x, s_y, s_z) \cdot \mathbf{T}(-x_f, -y_f, -z_f) \cdot \\ \mathbf{T}(x_r, y_r, 0) \cdot \mathbf{R}(0, 0, \theta°) \cdot \mathbf{T}(-x_r, -y_r, 0)\ . \qquad (3)$$

In OpenGL, the above transformation statements are written as,

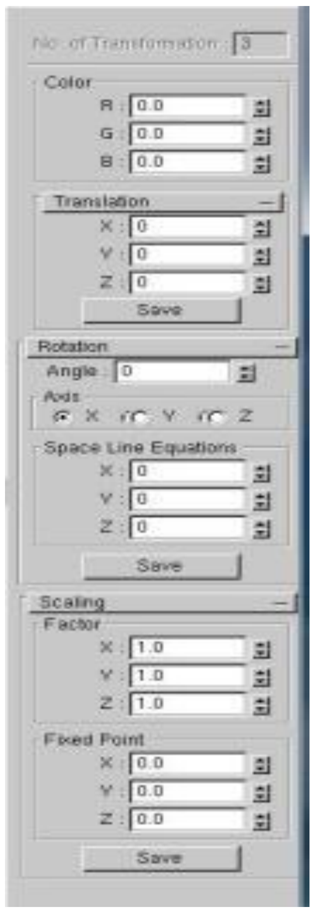
```
glTranslatef(tx, ty, tz);
glTranslatef(xf, yf, zf);
glScalef(sx, sy, sz);
glTranslatef(-xf, -yf, -zf);
glTranslatef(xr, yr, 0);
glRotatef(theta, 0, 0, 1);
glTranslatef(-xr, -yr, 0);
```

Note that the statements in OpenGL require the last transformation to be executed first. Traditionally, we learnt that when an object undergoes a series of geometric transformations, the coordinates of the object are changed depending on types of transformations applied to, and causing the changes in shape and size of the transformed object. This type of transformation is called fixed coordinate system transformation. However, anecdotal evidence showed that this concept is confusing while following the execution sequence by using OpenGL statements, as the intermediate transformed object obtained in OpenGL's execution sequence is the exact reverse of the fixed coordinate system transformation.

An alternate and better way to solve the conceptual confusion is to use local coordinate system transformation. In this case, the local coordinate system of the object is transformed while maintaining the coordinates of the object, instead of applying the transformation operations on the object coordinates as mentioned in fixed coordinate system transformation. The local coordinate system transformations work well by following the sequence of the transformation statements in OpenGL. Each intermediate transformed object is displayed based on the execution sequence as noted. Though the sequence of writing the transformation and the final result are the same, the concept is changed to view the transformation from a different perspective.

3 Application design

Conceptually, ARTransform is designed based on local coordinate system transformation to view in detail the geometric transformations step-by-step. It is built on ARToolkit for its freely available source codes, after analyzing the set of AR tools available [7]. The created AR objects and their transformations are displayed on the ARToolkit marker. Though a normal PC camera is sufficed, we opt to use the 3DVisor's Z800 Pro AR head mounted display to view the transformed objects, mainly to take advantage of the viewer's six degree of freedom offered by AR.



No. of Transformation : 3

Color
R : 0.0
G : 0.0
B : 0.0

Translation
X : 0
Y : 0
Z : 0
Save

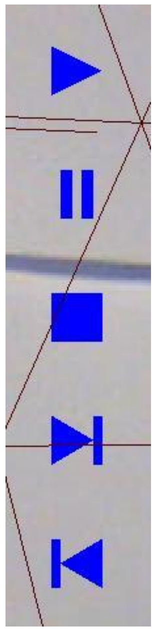
Rotation
Angle : 0
Axis
[✓ X] Y Z

Space Line Equations
X : 0
Y : 0
Z : 0
Save

Scaling
Factor
X : 1.0
Y : 1.0
Z : 1.0

Fixed Point
X : 0.0
Y : 0.0
Z : 0.0
Save

(a)



(b)

Fig. 1. Graphical user interface control panels: (a) TRANSCP; (b) ANIMCP.

The aim of ARTransform is to enable the viewer to be focus and grasp the concept of 3D geometric transformation easily and thus we keep manipulation of ARTransform simple during our design and development stages. Further, in order to minimize the abrupt change in user interaction habit, the viewer is guided to adapt to AR environment in stages by using the familiar classic input methods via keyboard and mouse in the first version of ARTransform. For logical input, we have developed two types of easy-to-understand graphical user interface control panels, i.e. transformation control panel (TRANSCP) and animation control panel (ANIMCP). TRANSCP contains options to enable the users to specify geometric transformation desired, ANIMCP contains buttons to animate the transformation edited in TRANSCP. Fig. 1 shows the TRANSCP for keyboard input and ANIMCP which can be displayed in the real environment for mouse input.

After transformation operations have been specified by the user in TRANSCP, each transformation is stored as a single entry in a table named `transData`. Space line rotation and fixed point scaling are decomposed into 3 basic geometric transformations, i.e. translation, rotation/scaling, and translation again; before they are stored as linked-lists of three entries in table `transData`. The desired color of the transformed object is also specified via TRANSCP. Fig. 2(a) shows the attributes for an entry of the table. In Fig. 2(b), index 0 shows a simple translation; index 1 shows decomposition of a composite transformation – fixed point scaling at (50.0, 0.0, –50.0), into three entries; index 2 shows decomposition of a rotation about a space line parallel to *z*-axis, with equation *x*=50 and *y*=50, into three other entries.

The initial version of ARTransform offers a set of basic functions on the ANIMCP to animate the transformations, including start, stop, pause, continue, forward, backward/undo and step-by-step view of the transformations. The design of the animation control buttons adopts the design concept of the buttons on a movie player. The viewer can analyze the same transformation over and over again until he/she has completely understood.

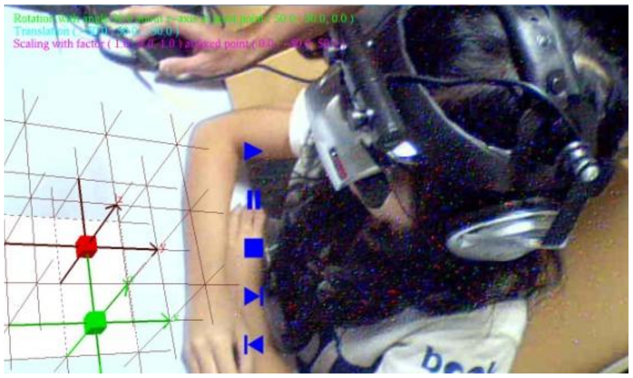


Fig. 3. A student with HMD is observing the transformations animations on ARTransform.

Fig. 3 shows a student with head mounted device observing the transformation animation. Each transformation to be observed is displayed on the top left-hand-corner position in the real environment.

4 Example

Based on the anecdotal evidence collected from the students' feedback in the Computer Graphics course, the main confusion in geometric transformation is caused by applying a series of scaling and rotation immediately one after another. In this section, we demonstrate an example involving rotation about a space line parallel to one of the coordinate axes, followed by fixed point scaling, which was mentioned in sections 2 and 3. The series of transformations are defined as: rotation of 90° anti-clockwise about the space line parallel to *z*-axis with equations *x*=50 and *y*=50; scaling of (1, 2, 1) by referring to a fixed point (50, 0, –50), and finally translation of (–50, –50, 50) units. Each transformation is given a different color to distinguish from the previous transformations.

The OpenGL statements in transformation sequence is given as follows,

```
glTranslatef (-50.0, -50.0, 50.0);
glTranslatef (50.0, 0.0, -50.0);
glScalef (1.0, 2.0, 1.0);
glTranslatef (-50.0, 0.0, 50.0);
glTranslatef (50.0, 50.0, 0.0);
glRotatef (90.0, 0.0, 0.0, 1.0);
glTranslatef (-50.0, -50.0, 0.0);
```

The output given by ARTransform application is displayed in Fig. 4. Fig. 4(a) – (d) show step-by-step transformations. Fig. 4(a) shows the original object in red. Fig. 4(b) shows the translation of (–50.0, –50.0, 50.0) units in each *x*-, *y*- and *z*-directions resulting in green object. Fig. 4(c) shows previously transformed green object is scaled to (1, 2, 1) of its original size at a fixed point (50, 0, –50) and resulting in yellow object. Fig. 4(d) shows previously transformed yellow object is rotated anti-clockwise about a space line parallel to *z*-axis with equations *x*=50 and *y*=50, resulting in magenta object. In this example, the original object in red is displayed as a consistent reference to the transformed objects. The *x*-, *y*- and *z*-grid lines of the original object are turned on, each grid space denotes 100 units and thus the transformed object's location and size can be analyzed when comparing with the original object.

5 Conclusions and Future Work

The hypothesis of seeing the 3D geometric transformations in the real environment can improve the viewer's understanding of geometric transformation was supported by the anecdotal evidence from our students' feedback at the end of the Computer Graphics course each year. In the feedback, the students expressed that the whiteboard teaching method of geometric transformation challenged their imagination and concentration, and that they were always lost in the 2D whiteboard space. Computer aided teaching in virtual reality is easier to understand compared to whiteboard, however, there was a strong urge to pull the object and its transformations out of the 2D display for closer analysis, especially when the original and the transformed objects were blocking each other.

At the moment, ARTransform is still at its infancy and we have not yet used it widely for students' learning purpose. Much works remain to be carried out. In our next version, we plan to include animated graphical explanations on 3D viewing attributes and characteristics, for example, eye/camera position, projection reference point, look-at point, perspective and orthographic transformations with frustum and parallelogram respectively. All animations will be carried out in real environment. The new version will also enhance the user interaction by adding a 3D mouse to interact directly with the AR object. For the new enhancements, the original aim of maintaining the learning curve gentle is kept in mind during the implementation.

6 Acknowledgement

This work is jointly sponsored by Universiti Sains Malaysia's Fellowship and Short Term Grant under reference number FPP 2010/0860 (P3427).

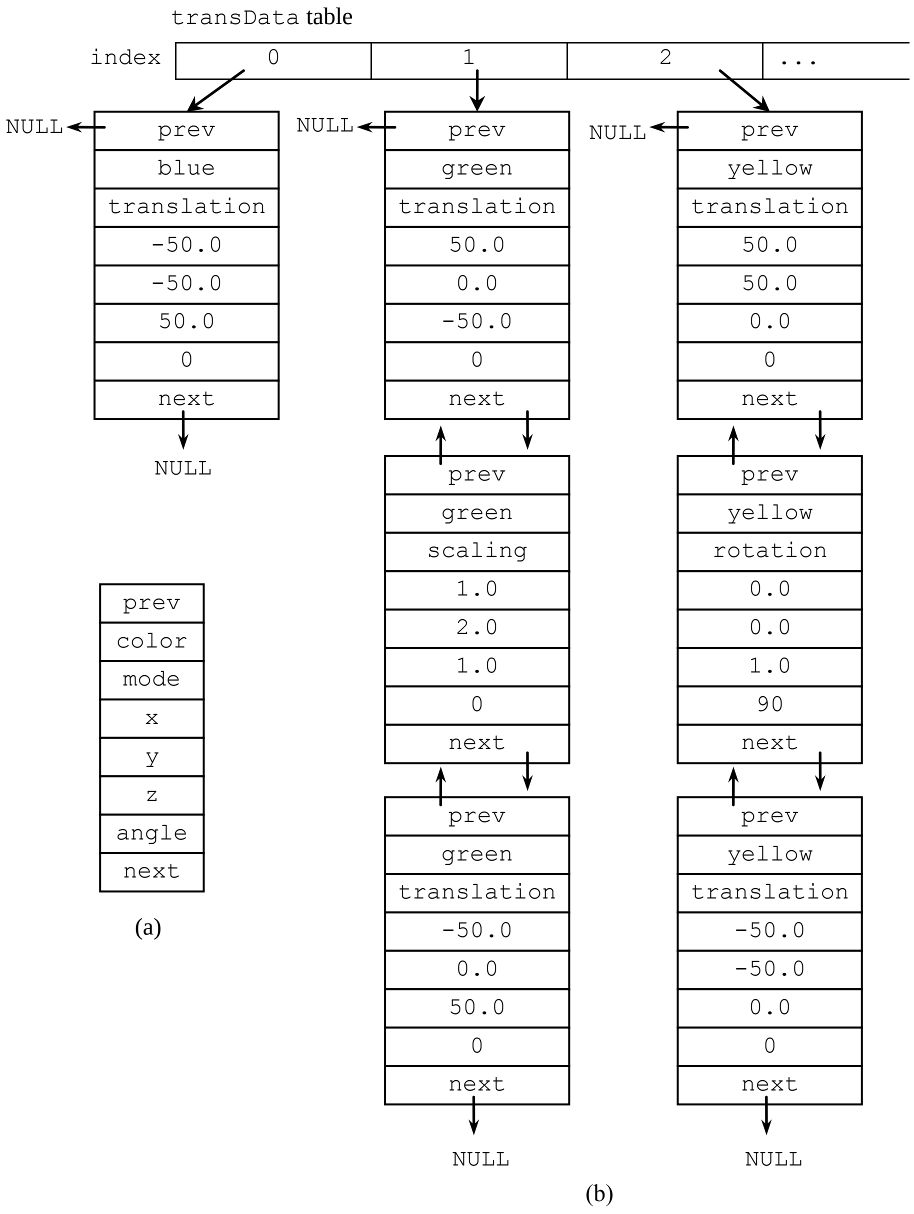
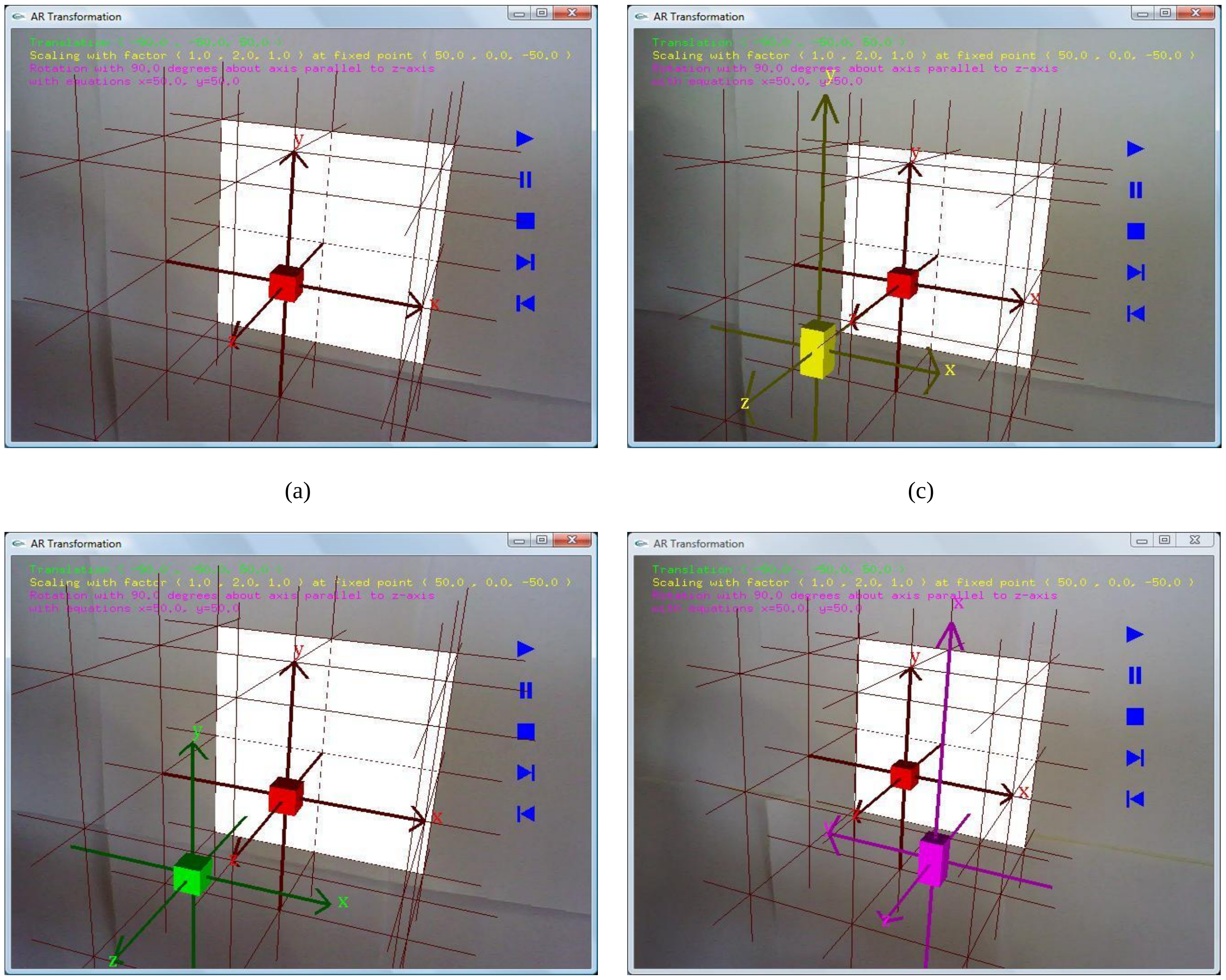


Fig. 2. Structures of `transData` Table: (a) Attributes of one entry in `transData`; (b) Index 0 shows simple transformation, indices 1 and 2 show decomposition of composite transformations to linked-lists.



(a)

```
Translation ( -50.0 , -50.0, 50.0 )
Scaling with factor ( 1.0 , 2.0, 1.0 ) at fixed point ( 50.0 , 0.0, -50.0 )
Rotation with 90.0 degrees about axis parallel to z-axis
with equations x=50.0, y=50.0
```

(c)

```
Translation ( -50.0 , -50.0, 50.0 )
Scaling with factor ( 1.0 , 2.0, 1.0 ) at fixed point ( 50.0 , 0.0, -50.0 )
Rotation with 90.0 degrees about axis parallel to z-axis
with equations x=50.0, y=50.0
```

(b)

```
Translation ( -50.0 , -50.0, 50.0 )
Scaling with factor ( 1.0 , 2.0, 1.0 ) at fixed point ( 50.0 , 0.0, -50.0 )
Rotation with 90.0 degrees about axis parallel to z-axis
with equations x=50.0, y=50.0
```

(d)

```
Translation ( -50.0 , -50.0, 50.0 )
Scaling with factor ( 1.0 , 2.0, 1.0 ) at fixed point ( 50.0 , 0.0, -50.0 )
Rotation with 90.0 degrees about axis parallel to z-axis
with equations x=50.0, y=50.0
```

Fig. 4. Step-by-step transformations: (a) Original object; (b) Translation of (–50, –50, 50); (c) Scaling of (1, 2, 1) at fixed point (50, 0, –50); (d) Rotation of 90° anti-clockwise about space parallel to *z*-axis with equations *x*=50 and *y*=50.

7 References

[1] Hearn, D., Baker, P.: Computer Graphics with OpenGL, 3rd Ed. Pearson Prentice Hall (2004)

[2] Nate Robin's OpenGL Tutor, http://www.xmission.com/~nate/tutors.html

[3] Dünser, A., Steinbügl, K., Kaufmann, H., Glück, J.: Virtual and augmented reality as spatial ability training tools. In: 7th ACM SIGCHI New Zealand chapter's international conference on Computer-human interaction: design centered HCI. ACM Press, New Zealand (2006)

[4] Kaufmann, H.: Construct3D: An Augmented Reality Application for Mathematics and Geometry Education. In: 10th International Conference on Multimedia, pp. 656 – 657. ACM Press, France (2002)

[5] Shelton, B. E., Hedley, N. R.: Using Augmented Reality for Teaching Earth-Sun Relationships to Undergraduate Geography Students. In: 1st IEEE international Augmented Reality Toolkit Workshop, Germany (2002)

[6] Shreiner, D., The Khronos OpenGL ARB Working Group: OpenGL Programming Guide 7th Ed.: The Official Guide to Learning OpenGL, Ver 3.0 and 3.1. Addison-Wesley Professional (2010)

[7] Ng, K.P., Tan, G.Y., Iman, L.Y.: Overview of Augmented Reality Tools. In: 18th National Symposium in Mathematical Science (SKSM), Malaysia (2010)