# Object-Oriented Design and Development of a Simulator Toolkit for 3D Graphics Teaching in Computer Graphics

Khaikhae Chulajata and Chesada Kaewwit

***Abstract*—Computer Graphics (CG) is a fundamental course to further study the related subjects such as computer animation, computer vision, computer games, and image processing. It is very difficult to shortly understand CG if students have no experience in theory of CG and basic mathematics. In consequent, we develop a simulator toolkit for teaching three-dimensional (3D) Graphics in order to assist students in understanding with 3D Graphics and using OpenGL. Moreover, this research focuses on using Object-Oriented environment to design and implement the simulator. From an assessment of students' satisfaction and using the simulator, this simulator is rated in the range of good and excellent for all aspects.**

***Index Terms*—Computer graphics, object-oriented, opengl, simulator.**

## I. Introduction

Computer Graphics (CG) is a fundamental course to further study the involved subjects such as computer animation, computer vision, computer games, and image processing. One of the most principle contents of this subject is three-dimensional (3D) Graphics because it is applicable to graphical usage, game development and multimedia application. In the CG course, 3D Graphics consist of transformation, lighting and coloring, and projections. These concepts are difficult to shortly understand due to complexity of the Graphics parameter setting and basic mathematics knowledge for transformation such as linear algebra, matrix, vector and so on. Moreover, most students have no an experience in 3D Graphics knowledge and using Graphics Application Programming Interface (API).

There are many researches which focus on how to the students easily understand the CG concepts especially 3D Graphics and use the Graphics API. They used a variety of teaching techniques to achieve the goal.

M. B. Gousie [1] stated that the concepts of CG are able to easily understand by implement many CG projects using an appropriate API such as OpenGL, Open Inventor or Java3D. The goals of the projects were to enhance understanding of CG principles and force students to design and code projects in a different programming language. However, students who interested in computer graphics need more skills in programming to implement the 3D graphics program and familiarize the 3D setting commands of API. In additional, From R. D. Necaise's study [2], an interactive graphics is one of the techniques supporting the process of learning and teaching in CG. Rance's research proposed the Graphix Windowing Toolkit as an alternative to the use GLUT which is the basic tools for creating a simple user interactive OpenGL based programs. The use of this toolkit has expended the interests and imagination of the students in his introductory CG course and has resulted in a number of highly successful programming projects. But the Rance's research did not implement the 3D transformation, projections, lighting and shading. In addition to, the research of G. Shultz [3] proposed a higher-level pedagogically Graphics API, called Simple Library. It is a simple Graphics API in order to facilitate program coding in 3D Graphics. However, the students need to code the Graphics program to understand 3D Graphics concepts. On the other hand, this research did not include the material color setting in 3D Graphics which is an essential property of them. Moreover, G. Shanshan's study [4] presented a new CG teaching model, namely all theories or algorithms of CG are included in a synthetic computer scenes case. Implementation steps for case teaching is introduced in detail, and then the multimedia teaching based on the new teaching model is researched. It has been proved that the efficiency and quality of teaching have been improved in practice.

From the conclusion of these researches it was revealed that the use of simulator for learning CG is a necessary to understand the basic concepts and their operations of 3D Graphics such as translation, rotation, lighting, coloring, and so on.

For this reason, this research focuses on using Object-Oriented environment to design and implement the simulator to assist students in understanding with 3D Graphics and using OpenGL. Object-Oriented environment is used for this research since it is easy to extend the objects [5] and their properties to conform to advance CG content.

Furthermore, the simulator is reinforcement of mathematics knowledge for studying in CG. It will be advantageous for both computer science and computer animation students who early study in CG course. Therefore, we divided the development of the simulator toolkit into 2 parts: object oriented design and implement. Then, we took the simulator to our CG class to evaluate it.

## II. Object Oriented Design and Implementation

The simulator is developed for the teaching assistance of CG course by using Microsoft visual C++ and graphics API such as OpenGL, GLUT and AntTweakBar. The simulator aims to assist students in understanding the fundamental concepts of CG such as setting color of material and lighting

Manuscript received December 10, 2012; revised March 12, 2013.

The authors are with the School of Science, University of the Thai Chamber of Commerce (e-mail: khaikhae_chu@utcc.ac.th).

of object, creating the viewport and understanding 3D transformations. Initially, we designed use case for serving the purpose of the simulator consisting of eight use cases as shown in Fig. 1. These use cases include:

- Create Viewport for creating viewport.
- Delete Viewport for deleting viewport.
- Create Object for creating object.
- Delete Object for deleting Object.
- Set Object Property for setting properties of an object such as Translation, Scale, Rotation and color.
- Create Light for creating a source of light.
- Delete Light for deleting a source of light.
- Set Light Property for setting properties of light such as Position, Ambient, Diffuse and Specular.

Then, we designed classes as shown in Fig. 2 for representing things in the real world as shown in Fig. 3. The problem domain classes include CGS Viewport, CGS World, CGS Camara, CGS Object and CGS Light. CGS World consists of CGs Object and CGS Light. The class CGS World represents the real world consisting of an object and light. The object and light are represented by the class CGS Object and CGS Light respectively. The class CGS Viewport represents a viewport, which is a region of a screen used to display object captured by a camera. And the camera is represented by the class CGS Camera. The object type of the simulator can be a teapot, polygon or sphere.

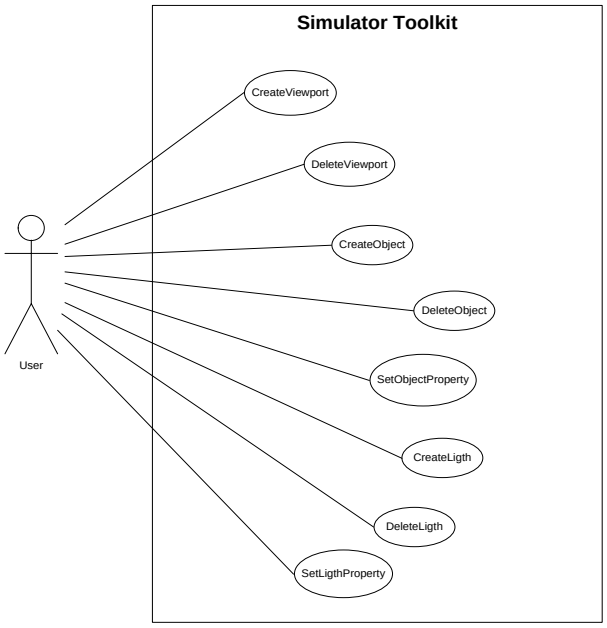



Fig. 1. Use case diagram.

Next, the class diagram can be converted into code to implement class of the simulator. The example of the implementation to the class "CGS Object" is shown in Fig. 4.

The result of design and implementation is the simulator that is comprised of 2 windows. The first window is the control center and the other window is the main window shown in Fig. 5.

From Fig. 5 a), the control center which is the main controller of the simulator is separated into 4 parts as follows: The first part is Viewport Management. It controls and manages the viewport of simulator such as create, delete and reset the viewport. The second part is World and Camera. The part is able to create and delete both objects and cameras on the viewport. The third part is Light Sources. Its functions are creating, deleting and setting the lighting source. Next, the final part, called Objects, is the created and deleted 3D objects.

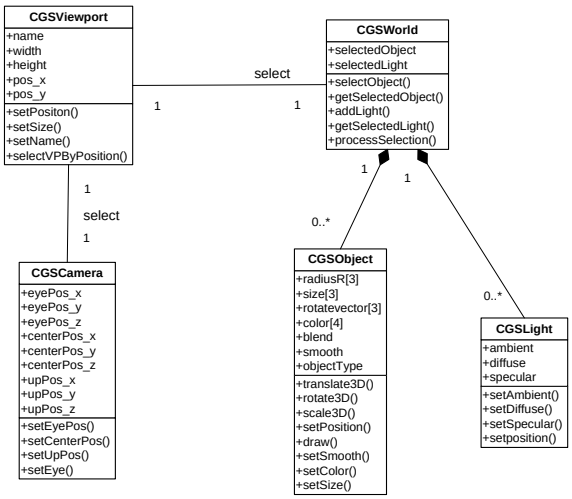



Fig. 2. Class diagram.

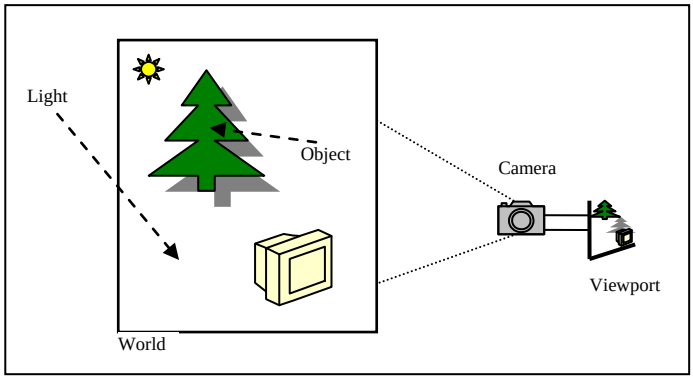


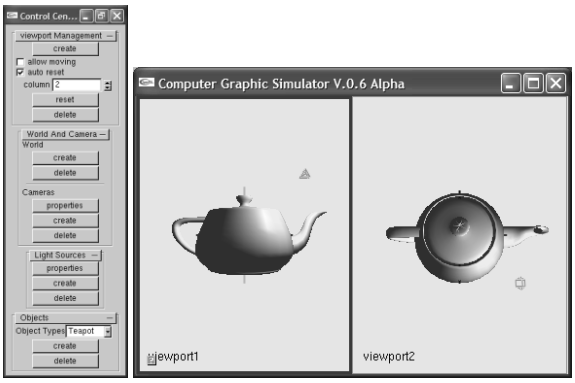

Fig. 3. Example of things in the real world.

From Fig. 5 b), the window displays a selected object on the viewport. The window can be added another viewport to show different view of the object.

```
class CGSObject
{
public:
   float posVector[3];
   float radiusR[3];
   float size[3];
   float rotateVector[3];
   float matrix[16];
   float color[4];
   bool blend;
   bool smooth;
   float speed;
   int objectID;
   string objectType;
public:
   CGSObject();
   void SetID(int id);
   void SetPosition(float x,float y,float z);
   void SetRadius(float rx,float ry,float rz);
   void SetSize(float szx,float szy,float szz);
   void SetSpeed(float sp);
   void setRotate(float rx,float ry,float rz);
   int getID();
   float* getPosition();
```

```
float* getRadius();
float* getSize();
float getSpeed();
float* getRotate();
void Translate3D(float x,float y,float z);
void Rotated3D(float radians,float x,float y, float z);
void Scaled3D(float x,float y,float z);
void Translate3D();
void Rotated3D();
void Scaled3D();
void setColor4f(float red,float green,float blue,float alpha);
setColor4v(const float *color);
void setBlend(bool state);
void setSmooth(bool state);
const float* getColor4v();
float getBlend();
float getSmooth();
void addTransformMatrix(const float *matrix);
void clearMatrix();
void dotMatrix(float *m2);
void Render();
void Draw();
void drawLines();
string getObjectType();
};
```

Fig. 4. An example of class CGS object.




a) b)

Fig. 5 a) The control center and b) the main window.

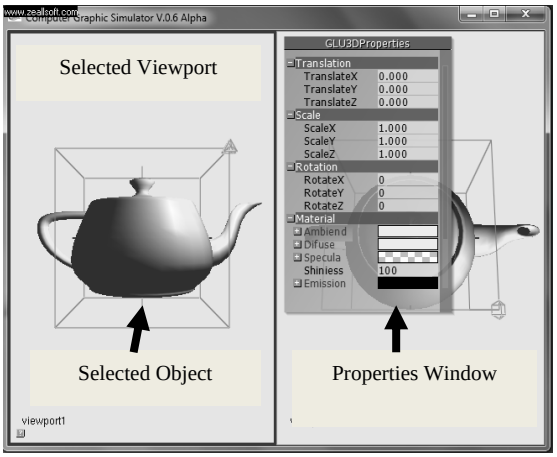



Fig. 6. Example of modification of the object's properties.

In case the students want to change object's properties, they must select a viewport and object, and then right click on the object, the properties window will be displayed as shown in Fig. 5. The students can adjust properties of the selected object such as translation, scale, rotation and material. Interactions between objects related to this use case are shown in Fig. 6.

The sequence diagram shown in Fig. 7 can be described as follows:

- Two methods, select VPBy Position (x, y) and process Selection (x, y), in Viewport class are called by a student.
- Viewport class interacts with World class via calling selected Object (object ID) method.
- The get Select Object method in World class is invoked by selecting one of the shown objects on screen. A frame will appear after the selection.
- The student is able to set several properties such as rotation, color, scale and position settings of the selected object via Rotate3D (x, y, z), Set Color (red, green, blue) Scale3D (x, y, z), and Set Position (x, y, z), respectively.

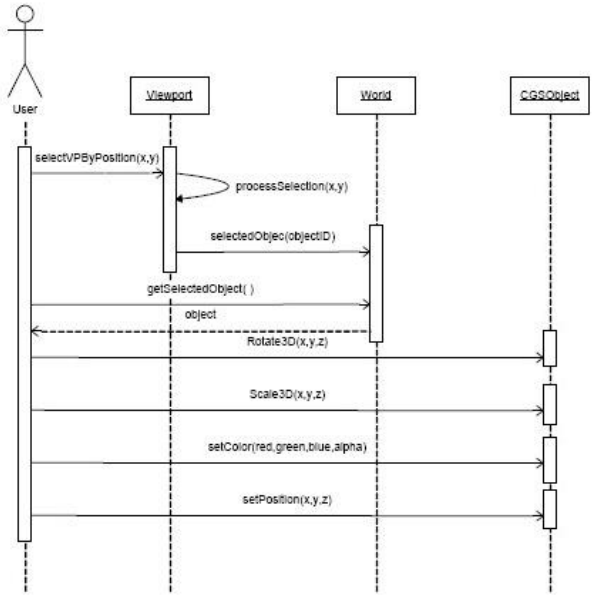



Fig. 7. Sequence diagram for use case set object property.

## III. Assessment of the Satisfaction of Users

There are two groups of the evaluation. For the first time, 57 students enrolled in CG course in the second semester of 2009 school year. Then, 42 students who have studied in CG course opened in the first semester of 2010 school year. The total number of student in this evaluation is 99 persons. The criteria used to evaluate users' satisfaction include ease of the Toolkit using and understanding of 3D Graphics. The measure scale ranges from 1 to 5 (Very poor to Excellent), shown in Table I.

TABLE I: Satisfaction of the 3D Graphics Toolkit

| Assessment Topic | Average Satisfaction Level |
|---|---|
| Ease of using | Good |
| Ease of understanding | Excellent |

The study discovered that more than 80% of students understood 3D Graphics concepts and OpenGL in CG course. Besides, the simulator supports ease of using with Good satisfaction level.

## IV. Conclusion

According to the study results, we found that the simulator

toolkit is important tool for the 3D graphics study. It can be applied some theories of CG to practice and to reduce time in learning and teaching 3D Graphics API. In this paper, we develop the simulator toolkit for teaching 3D Graphics and using OpenGL. The simulator is developed by using Microsoft Visual C++, OpenGL, GLUT, and AnTweakBar. Then, we use Object-Oriented environment to design and implement the simulator because it is easy to extend the objects and their properties to the simulator.

## REFERENCES


[1] M. B. Gousie, "Teaching computer graphics in a small department," *Journal of Computing Sciences in Colleges*, vol. 15, pp. 194-202, May 2000.

[2] R. D. Necaisei, "Interactive graphics using OpenGL and the Graphix Windowing Toolkit," *Journal of Computing Sciences in Colleges*, vol. 22, pp. 220-202, December 2006.

[3] G. Shultz, "Integrating 3D graphics into early CS Courses," *Journal of Computing Sciences in Colleges*, vol. 21, pp. 169-178, February 2006.

[4] G. Shanshan, Z. Caiming, Z. Rui, and C. Jing, "Design and Implementation of Case in CG Case Teaching," in *Proc. IEEE Symp. IT in Medicine and Education (ITME 2008)*, IEEE Press, Dec. 2008, pp. 332-335, doi: 10.1109/ITME.2008.4743882.

[5] E. Youdon, "Object-Oriented Systems Design: an Integrated Approach," *Upper Saddle River*, NJ: Prentice-Hall International, 1994.


**Khaikhae Chulajata** received her M.Sc. in Applied Statistics, majoring in Computer Science, from the National Institute of Development Administration (NIDA). She is currently a lecturer in the School of Science, University of the Thai Chamber of Commerce. Her research interests are in system analysis and design, database management, and software development.

**Chesada Kaewwit** received his M.Sc. in Applied Statistics, majoring in Computer Science, from the National Institute of Development Administration (NIDA). He is currently a lecturer in the School of Science, University of the Thai Chamber of Commerce. His research interests are in computer graphics, data communication and networking, and Internet technology.