Shadow Image and Special Effects Implementation Techniques for Virtual Shadow Puppet Play

TAN KIAN LAM, ABDULLAH ZAWAWI TALIB
School of Computer Sciences
Universiti Sains Malaysia
11800 USM Pulau Pinang
MALAYSIA
andrewtankianlam@gmail.com, azht@cs.usm.my

Abstract: - This paper describes the shadow puppet play which is a modelling technique using OpenGL in order to produce realtime shadow image. Besides, the novelty of this paper is to improve the interaction of shadow puppet play by applying both the texture mapping and the blending technique. The integration of these methods gives the special lighting effects that are vital for the quality improvement of the traditional shadow puppet play. The outcomes of this research has been rated successful based on the result of the experiments that measures the satisfaction of the respondents.

Key-Words: - Traditional Shadow Puppet Play, Virtual Shadow Puppet Play, Texture Mapping, Blending, Accumulation Buffer, Geometric Transformation

1 Introduction and Related Work

In Malaysia and many parts of Asia, traditional shadow play or Wayang Kulit is one of the well-known traditional storytelling methods. There are several types of Wayang Kulit Theatre throughout Southeast Asia (see Figure 1). Wayang Siam also known as Wayang Kulit Kelantan is the most widely performed in Malaysia and it originates from Kelantan. Besides showing the puppet, its musical orchestra that consists of up to a maximum of 10 players play various instruments such as drums, serunai and gongs [1]. Si Galigi Me Wayang [2], Nang [3], Shadow play of China [4], and Turkish shadow play [5] are some of the shadow puppet plays around the world.

Figure 1. The Wayang Kulit Theatre

Chee and Talib [6] have proposed a framework for virtual storytelling using traditional shadow puppet play. The framework includes a study on the mapping of traditional shadow puppet play into a virtual storyteller and a proposed method of rendering of shadow images for computerized traditional shadow puppet show. Rahman [7] has attempted Wayang "Virtual" which is an experimental attempt at combining traditional shadow puppet play together with three-dimensional model using the "IRIS Showcase" Silicon Graphics software. Zhu et al. [8] have proposed a set of techniques for the simulation of Chinese Shadow Play. In their work, digital photographs with a high resolution of the puppet are taken and these images are then processed digitally for the purpose of modelling the puppet. They have used a rendering technique based on photon mapping method to make the shadow effect. Using photon mapping, rendering time is much faster compared to other methods such as radiosity. Hsu et al. [9] have presented the design of an animation system that can generate the motions of a puppet in a shadow play automatically.

Tan et al. [10] have introduced a method of modeling to model shadow puppet play using sophisticated computer graphics techniques available in OpenGL in order to allow interactive play in real-time environment as well as producing realistic animation. This paper complements Tan et al. [10] by providing details of the implementation techniques for shadow imaging and special effects for the virtual shadow puppet play.

2 Implementation Approach and Methods

As shown in Figure 2, shadow image is required in order to ensure the right appearance of the puppet in the shadow play environment. Special effect is the adjustment of the brightness of the light and blurring

that are being changed in rapid motion according to the necessities of the effects. In this research, an attempt is made to make the puppet which is a plane (2D) behave as though they are in various environment with lighting technique and accumulation buffer effects. In order to generate the shadow image, texture mapping method is applied to improve the puppet image. At the same time, blending technique is also applied to produce shadow effect. The combination of the puppet image and the shadow effect generated plays a crucial role in the improvement of the shadow image of the traditional shadow puppet play. Special effects which consist of background lighting and blurring are generated using the lighting technique and accumulation buffer respectively with the aim of generating the right environment for virtual shadow puppet play.

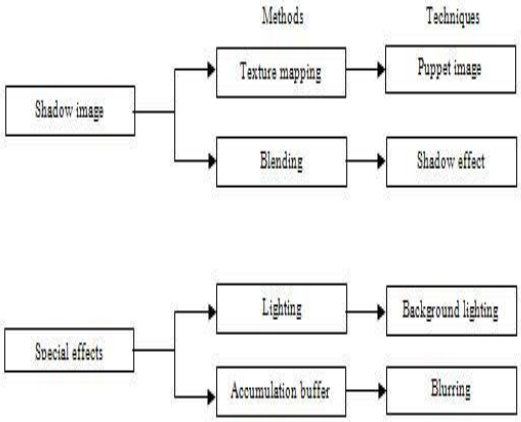

Figure 2. The Overall Implementation Approach

3.1 Methods for Shadow Image

3.1.1 Texture Mapping

Texture mapping allows us to glue an image to any polygon. Texture mapping works only in RGBA mode. A texture is usually thought as being two-dimensional like most of the images, but it also can be one dimensional. The data describing a texture may consist of one, two, three, or four elements per pixel, which represents anything from a modulation constant to an (R, G, B, A) quadruple. Texturing is enabled or disabled by using glEnable() or glDisable() with the symbolic constant GL_TEXTURE_1D or GL_TEXTURE_2D for one– or two– dimensional texturing [10][11]. Texture mapping is one of the initial steps before applying blending technique in order to allow an interactive and realtime playing for the puppet images in virtual shadow puppet play.

3.1.2 Blending Technique

Blending technique is a function that combines colour values from a source and a destination. The final effect is that parts of the scene appear as translucent. When blending is enabled, the alpha value is often used to combine the colour value of the fragment being processed with the pixel, which is already stored in the frame buffer. The blending effects occur after the scene has been rasterised and converted to fragments. With blending, the user can use alpha blending to create a translucent fragment that lets some of the previously stored colour value "show through". During blending, the colour values of the incoming fragment (the source) are combined with the colour values of the corresponding currently stored pixel (the destination) in a two-stage process. This is analogous to real shadow puppet play as the screen acts as a translucent material that allows the shadow image of the puppet to be seen by the audience in interactive and realtime penvironment [10][12].

3.2 Methods for Special Effects

3.2.1 Lighting Technique

Macromedia Flash MX can be used to create several themes to create virtual lighting effect for virtual shadow puppet play. Various tools such as Color Panel can be used to paint out the themes [10]. These themes are generated as the background lighting for virtual shadow puppet play.

3.2.2 Accumulation Buffer

Accumulation buffer can be used for motion blur, scene antialiasing and simulating photographic depth of field to produce the blurring effects in virtual shadow puppet play. Typically, a series of images is generated in one of the standard colour buffers, and these are accumulated. Then, it writes into the accumulation buffer. When the accumulation is finished, the result is copied back into a colour buffer for viewing. The accumulation buffer can be used to approximate what we would see in a photograph where items are more and more blurred as their distance from a plane of perfect focus increases. It is not an exact simulation of the effects produced in a camera, but the result looks similar to what a camera would produce [10][13].

4 Implementation

4.1 Shadow Imaging Techniques

The purpose of using texture mapping is to import the image and glue the image to the polygon that we have created. Figure 3 provides the pseudo code for texture mapping that consists of two parts. The first part from lines 2 to 25 loads the bitmaps and converts the bitmap

into texture. The second part from lines 26 to 31 creates a rectangle and applies the texture mapping on the rectangle. GLAUX library needs to be included (see line 1) because this is the header file for the GLAUX library that contains the particular function AUX_RBGImage().

```
1       #include <gl/glaux.h> // header file for GLAUX library
        .
        .
2   int LoadBodyTexture()   //Load Bitmaps and Convert To Textures
3       {
4       int Status = FALSE;   //Status Indicator
5       AUX_RGBImageRec *TextureImage[1];   //Create Storage Space for the
texture
6       memset(TextureImage,0,sizeof(void*)*1); // Set the Pointer to NULL
7
8       //Load the Bitmap, Check for Errors, If Bitmap's Not found Quit
9       if(TextureImage[0]=LoadBMP("body.bmp"))
10      {
11      status=TRUE;    //Set the Status to True
12      glGenTextures(1, &texture[0]);    //Create the Texture
13
14      //Typical Texture Generation Using Data From the Bitmap
15      glBindTexture(GL_TEXTURE_2D, texture[0]);
16      glTexImage2D(GL_TEXTURE_2D, 0, 3, TextureImage[0]->sizeX,
17        TextureImage[0]->sizeY, 0, GL_RGB, GL_UNSIGNED_BYTE,
18        TextureImage[0]->data);
19      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
20      GL_LINEAR);
21      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
22      GL_LINEAR);
23      }
        .
        .
24      free(TextureImage[0]->data);   //Free the texture image memory
25      free(TextureImage[0]);   //Free the image structure
        }
        .
26  glBegin(GL_QUADS);
27      glTexCoord2f(0, 0); glVertex3f(0, 0, -1);
28      glTexCoord2f(1, 0); glVertex3f(1, 0, -1);
29      glTexCoord2f(1, 1); glVertex3f(1, 1, -1);
30      glTexCoord2f(0, 1); glVertex3f(0, 1, -1);
31  glEnd();
        .
32  glEnable(GL_TEXTURE_2D);   //Enable texture mapping
```

Figure 3. Coding for Texture Mapping

The function glTexParameteri (lines 19 to 22) is used to specify the minification and magnification filtering methods. The first parameter which is GL_TEXTURE_2D sets the texture in two-dimensional. The second parameter, which is GL_TEXTURE_MIN_FILTER is used whenever the texture mapping of a pixel is being done to an area greater than one texture element. The third parameter, which is GL_LINEAR is in used because this parameter provides a faster method than others such as GL_NEAREST_MIPMAP_LINEAR. In line 21, the second parameter which is GL_TEXTURE_MAX_FILTER is used because the texture mapping of the pixel is being done to an area less than or equal to one texture element.

Lines 26 to 31 create the rectangle and perform texture mapping on the rectangle. The purpose of using glTexCoord is to specify the texture coordinates in one, two, three, or four dimensions. glTexCoord1 sets the current texture coordinates to (s, 0, 0, 1); a call to glTexCoord1 sets them to (s, t, 0, 1). Similary, glTexCoord3 specifies the texture coordinates as (s, t, r, 1), and glTexCoord4 defines all four components explicitly as (s, t, r, q). The current texture coordinates are part of the data that is associated with each vertex and with the current raster position. Initially, the values for s, t, r, and q are (0, 0, 0, 1).

The purpose of using blending techniques is to create shadow image that has the same effects as the shadow puppet play image of the traditional shadow puppet play. Blending can provide fast generation of the image to allow fast real time interaction. Blending is initially disabled. So, in order to enable blending function, we need to use glEnable(GL_BLEND). glBlendFunc() that defines the operation of blending when it is enabled and it controls the effect of blending. The first parameter for glBlendFunc, which is GL_SRC_ALPHA means the source colour is multiplied by source alpha. The second parameter, which is GL_ONE_MINUS_SRC_COLOR means the destination colour is multiplied by (1, 1, 1, 1 – source colour). Transparency is best implemented by using blend function (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA) with primitives sorted from furthest to nearest. Transparency calculation does not require the presence of alpha.

The incoming (source) alpha is correctly thought of as a material opacity, ranging from 1.0 which represents complete opacity, to 0.0 which represents complete transparency.

After blending is enabled, the incoming primitive colour is blended with the colour that is already stored in the frame buffer. glBlendFunc() defines the operation of blending when it is enabled and it controls the effect of blending. The first function parameter specifies which of the nine methods is used to scale the source colour components. The second function parameter specifies which of the nine methods is used to scale the destination colour components.

Figure 4 shows the shadow image of a puppet after applying the above implementation of texture mapping and blending techniques.

Figure 4. Shadow Image of a Puppet

4.2 Special Effects

4.2.1 Lighting Technique

This section describes the techniques used to create the lighting effects by using Macromedia Flash MX. If we use the lighting functions in OpenGL, we can see the boundary for the rectangle that we have created as shown in Figure 5, because the light rays from light source illuminate on the vertices for the rectangle that we have created. As a result, we have decided to use Macromedia Flash MX to create the lighting effects by including a background lighting created using the software.

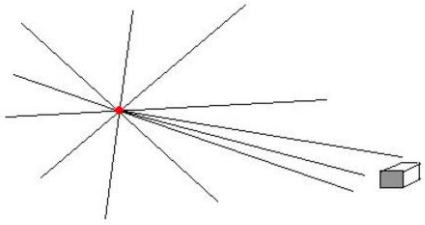

Figure 5. The Effect of Using OpenGL Lighting Functions

Figures 6 to 8 are the themes that we have created namely darkest theme, brightest lighting theme and moderate lighting theme respectively.

Accumulation buffer is simply an extra image buffer that is used to accumulate composite images. Figure 9 provides the code that implements the blurring effect on shadow puppet play image using the accumulation buffer.

Figure 6. Darkest Theme

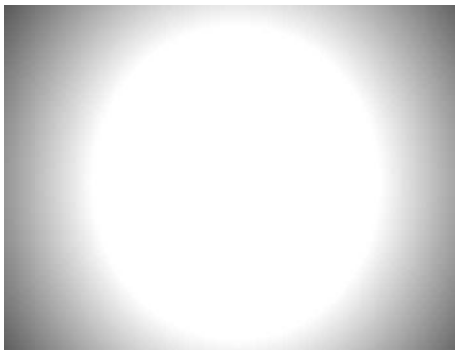

Figure 7. Brightest Lighting Theme

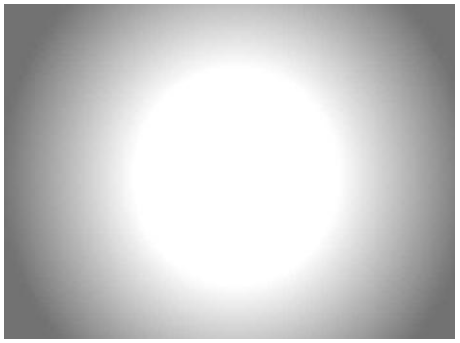

Figure 8. Moderate Lighting Theme

Lines 1 to 11(Figure 9) form the declaration for the jitter point. The purpose of using the jitter point is to move the camera 8 times. It will accumulate each of the frames 1/8 in a buffer. Then, only the buffer is displayed and we can only see a blurred vision.

Lines 12 to 25 and lines 26 to 34 define two routines for jittering namely accPerspective() and accFrustum(). The routine accPerspective() is used in place of gluPerspective(), and the first parameters of both routines are the same. To jitter the viewing frustum for depth of field, the x and y jitter values are passed to the fifth and sixth parameter of accPerspective().The last three parameters are used for depth-of-field effects. The amount of blur is determined by multiplying the x and y jitter values (seventh and eighth parameters of accPerspective()) by a constant. To turn off the antialiasing, we set the fifth and sixth parameters to accPerspective() to 0.0.

```
1   jitter_point j8[] =
2   {
3   {-0.334818, 0.435331},
4   {0.286438, -0.393495},
5   {0.459462, 0.141540},
6   {-0.414498, -0.192829},
7   {-0.183790, 0.082102},
8   {-0.079263, -0.317383},
9   {0.102254, 0.299133},
10  {0.164216, -0.054399}
11  };
    .
    .
12  void accFrustum(GLdouble left, GLdouble right, GLdouble
    bottom,GLdouble top, GLdouble near1, GLdouble far1, GLdouble pixdx,
    GLdouble pixdy, GLdouble eyedx, GLdouble eyedy, GLdouble focus)
13  {
14  GLdouble xwsize, ywsize;
15  GLdouble dx, dy;
16  Glint viewport[4];
17  glGetIntegerv (GL_VIEWPORT, viewport);
18  xwsize = right – left;
19  ywsize = top – bottom;
20  dx = -(pixdx*xwsize/(GLdouble) viewport[2] + eyedx*near1/focus);
21  dy = -(pixdy*ywsize/(GLdouble) viewport[3] + eyedy*near1/focus);
22  glMatrixMode(GL_MODELVIEW);
23  glLoadIdentity();
24  glTranslatef(-eyedx, -eyedy, 0.0);
25  }
26  void accPerspective(GLdouble fovy, GLdouble aspect, GLdouble near1,
    GLdouble far1, GLdouble pixdx, GLdouble pixdy, GLdouble eyedx,
    GLdouble eyedy, GLdouble focus)
27  {
28  GLdouble fov2, left, right, bottom, top;
29  fov2 = ((fovy*3.1418)/180.0)/2.0;
30  top = near1 /(cos(fov2)/sin(fov2));
31  bottom = -top;
32  right = top*aspect;
33  left = -right;
34  accFrustum(left, right, bottom, top, near1, far1, pixdx, pixdy, eyedx, eyedy, f
    ocus);
    .
35  glAccum(GL_ACCUM, 0.125);
    .
    .
36  glAccum(GL_RETURN, 1.0);
```

Figure 9. Coding for Accumulation Buffer

The scene is drawn eight times, each with a slightly jittered viewing volume, by calling accPerspective().

Figure 10 shows an image produced with blurring effect by using the depth of field effect and Figure 11 shows the same puppet without any bluring effect.

Figure 10. Blurring Effect using Depth of Field

Figure 11. Without Blurring Effect

5 Evaluation and Discussion

A survey together with a set of experiments are carried out to determine the acceptability and the quality of the techniques and methods derived in this research. Table 1 shows the result of the survey that evaluates the puppet's appearance, interactive and realtime playing, lighting and blurring effects.

From the table, 95% of the respondents have given an average rating on the first question which is on the effect of the puppet's appearance compared to the actual shadow puppet. In the second question, which is in interactive realtime playing of the virtual shadow puppet play, majority of the respondents (65%) have given the rating of 6 on the interactive and real time playing of the virtual shadow puppet play. Most of the respondents (80%) gave the highest scale (7 out of 7) on the lighting and blurring effects of the virtual shadow puppet play compared to the traditional shadow puppet play.

Questions / Rating Scale	1) Puppet's Appearance	2) Interactive and Real Time Playing	3) Lighting and Blurring Effects
1	0	0	0
2	0	0	0
3	0	0	0
4	2	0	0
5	10	2	1
6	7	13	3
7	1	5	16
Total	20	20	20

Table 1. Result from Respondents

From the result of the evaluation, some useful and effective feedbacks are obtained. First of all, many of the respondents think that the virtual puppet modelling is rather compatible to the traditional ones. The crafting of the outline of the puppet has properly shadowed the real puppet. The respondents also commented that the boundary of the rectangle can be seen when the moderate lighting theme is used.

On the lighting effect, many respondents showed their agreements and they were fascinated by the differences that can be made to the brightness. This is due to one distinct similarity that can be found between the theme for the traditional shadow puppet play and our theme in virtual shadow puppet play. Some respondents suggested further enhancement of the effects using the fade in and fade out techniques. For example, fade in technique is required when a puppet has to be removed and brought in respectively . In addition, the respondents are satisfied in witnessing the extra and special blurring effects.

6 Conclusion and Future Work

As compared to the previous research of virtual shadow puppet play, this research has provided alternative solutions in visual simulation and animation of shadow puppet play. Firstly, the application of texture mapping and blending methods have produced the shadow puppet play realistically and efficiently, and allowed interactive realtime playing of the virtual shadow puppet. Secondly, special effects are applied and hence, the background of the shadow puppet is well-suited to the flow of the story. The lighting has brought life to the virtual shadow puppet play as well as displaying the play in real-time surroundings.

In order to further improve this research, accurate vertices with the vertices of the puppet should be created. Moreover, the display screen can be enhanced by using the fading effects. Lastly, consideration of physics factor should be taken care in order to produce a more natural animation.

References:

[1] Matusky P., *Malaysian Shadow Play and Music: Continuity of an Oral Tradition*: Oxford University Press, 1993

[2] Edward V. N., *Javanese Wayang Kulit: An Introduction*: Oxford University Press, 1980

[3] Salij H. J., *Shadow Play and other Stories*: Heinemann Singapore, 1982

[4] Liu J., *Chinese Shadow Puppet Plays*: Morning Glory Publishers, 1998

[5] David C., *Shadow Puppets & Shadow Play*: Crowood Press, 2008

[6] Chee J., Talib A. Z., A Framework for Virtual Storytelling Using Traditional Shadow Play, *Proc. International Conference on Computing and Informatics (ICOCI 06)*, (CD Proc.), 2006, pp. 1-6

[7] Rahman K. A., (1999) Wayang "Virtual" Integration of Computer Media in Traditional Wayang Kulit (Shadow Play) Performance, [Online]. [Accessed 9th Feb 2008]. Available from World Wide Web: http://www.itaucultural.org.br/invencao/papers/Rahman.htm

[8] Zhu Y. B., Lee C. J., Shen I. F., Ma K. L., Stompel A., A New Form of Traditional Art: Visual Simulation of Chinese Shadow Play, *International Conference on Computer Graphics and Interactive Techniques Sketches and Applications*, 2003, pp. 1-1

[9] Hsu W. S., Ye T., Planning Character Motions for Shadow Play Animations, *Proc. of International Conference on Computer Animation and Social Agents (CASA'05)*, 2005, pp. 184-190

[10] Tan K. L., Talib A. Z., Osman M. A. Real-Time Simulation and Interactive Animation of Shadow Play Puppets Using OpenGL *International Journal of IJCIE* Vol. 4, 2010, pp. 1-8

[11] Mason W., Jackie N., Dave S., *OpenGL Programming Guide*: Addison Wesley Developers Press, 1997.

[12] Donald H., Pauline B., *Computer Graphics with OpenGL*: Pearson Prentice Hall, 2004.

[13] Richard S. W., Michael S., *OpenGL Super Bible Second Edition*: Waite Group Press, 2001.