Tour Into the Picture

Course: CS3241 – Computer Graphics
Students: Qin Yan, Wu Yinghui
Instructors: Dr. Tan Tiow Seng, Dr. Teh Hung Chuan
School of Computing
National University of Singapore
Lower Kent Ridge Road
Singapore 119260

Abstract

This assignment is to experience the interesting work of "Tour into the Picture" by Horry, Anjyko and Arai, SIGGRAPH 1997.

Introduction

This program accepts an input image together with coordinates (*ul*, *vt*), (*w*, *h*), and (*vx*, *vy*). The first two input coordinates defined the left boundary *ul*, top *vt*, width *w*, and height *h* of the back face of the environment in the image, while the last input coordinate defined the vanishing point of the viewing cube. The two major tasks accomplished by the program are: First, joining the vanishing point to the 4 corners of the back face, it partitions the input image into 5 parts: back, left, right, ceiling and floor. Second, it wraps the 5 parts to a cube of size 1x1x1—it recovers textures of the environment through bilinear interpolation, with the assumption that the back face has height 1 and the camera is at distance 1 from the back face. With these two tasks done, one can then move around in the created "environment" by re-mapping the recovered textures.

Implementation

There are nine tasks that needed to be implemented in the skeleton code provided.

1. **Display routine implementing**

Use `uDisplayList=glGenLists(no_of_lists)` to initialize the lists. Put texture mapping between `glNewList(list_no, GL_COMPILE)` and `glEndList()` to store the data in the list. `glCallList(list_no)` is called inside `display()`, which is the display callback.

2. **Easy navigation by mouse**

Besides the extended mouse callback, `mouse()`, a mouse motion callback, `motion()`, is added to deal with mouse movements while button is being pressed. Each action is assigned to a certain mouse event.

Right button is bound to show the system menus. This function is implement by using `glCreateMenu` and `glAttachMenu` in the `main` function as well as `initMenus` function.

Furthermore, when in photo display mode, user can choose vanishing point as well as the back plane by simple mouse clicks.

3. Zooming effect.

Zooming effect is implemented in both keyboard callback (Ctrl+Up / Ctrl+Down) and mouse motion callback (drag up/down under "Zoom" mode). After each action from keyboard or mouse, a new viewing frustum is built with modified values of *xMax*, *yMax* by calling `glFrustum(-xMax*FUDGE, xMax*FUDGE, -yMax*FUDGE, yMax*FUDGE, FUDGE, 10.0)`.

4. Point sampling

By executing point mapping, one point (*i, j*) on a certain wall will map to a point (*u, v*) on the original image. The values of *u* and *v*, which are got from the calculation, may not be integers. To get the color value of the fractional position, bilinear interpolation needs to be done. Bilinear interpolation is done inside the method `getPixel()`. Inside the method, u_0 and v_0 are used to indicate the integer value of *u* and *v*. The four point needed to be interpolated are (u_0, v_0), (u_0, v_0+1), (u_0+1, v_0) and (u_0+1, v_0+1). Get the color value *pix1, pix2, pix3* and *pix4* of these 4 points first. Linear-interpolate pix1 and pix2, pix3 and pix4 horizontally, get color1 and color2. Then do linear interpolation again on color1 and color2 vertically, get the color of the interpolated point. Note the case that one of more of u_0, v_0, u_0+1 or v_0+1 is out of bounds of texture maps should be taken into consideration.

5. Adding object.

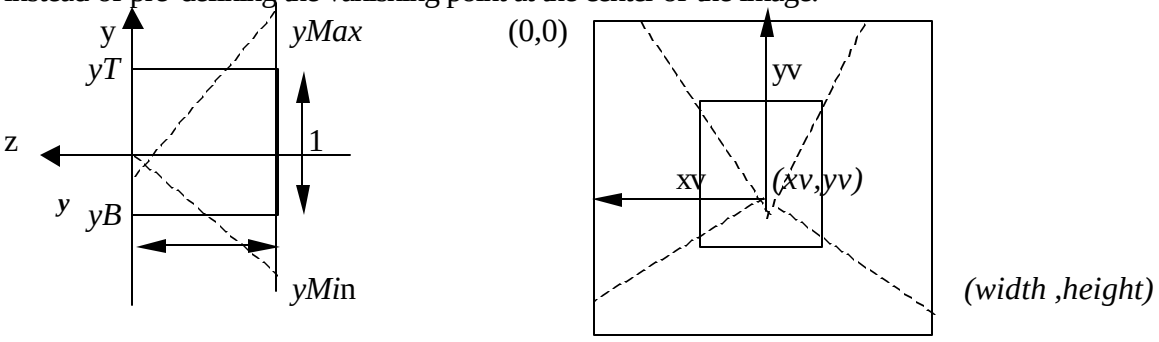
Seven relatively complex objects are predefined for user to insert into the scene. The implementations of these objects are from the sample code provided under the topic to lighting and materials. Basically, a library called GLM is included with the precompiled definitions of the seven objects so that it is easy to achieve higher performance for scene walk-through.

6. Display modes toggling

There are four display modes available: wire-frame, texture, polygon, and photo. A method `toggleMode()` is written to specify each display mode. Furthermore, references to this function is put in both keyboard and mouse callbacks so that user can choose the mode by either press 'T' in the keyboard or choose from the right-button menu.

7. Removing assumption(s)

The user is asked to define the vanishing point *(xv, yv)* as well as the back face instead of pre-defining the vanishing point at the center of the image.



$$a = \frac{yv - vT}{yv} = \frac{yT}{yMax} \qquad b = \frac{vB - yv}{height - yv} = -\frac{yB}{yMin}$$

$$yT\text{-}yB=1 \qquad \frac{yMax}{yMin} = \frac{yv}{height - yv} = \frac{1}{factor\_y}$$

Solving for *yT*, *yB*, *yMax* and *yMin*:

$$yT = \frac{a}{a + b \times factor\_y} \qquad yB = -\frac{b \times factor\_y}{a + b \times factor\_y}$$

$$yMax = \frac{1}{a + b \times factor\_y} \qquad yMin = yMax \times factor\_y$$

Apply the same functions on x-direction and calculate *xL*, *xR*, *xMax* and *xMin*. Use these values to do the point sampling. Also, to map *(x, y, z)* values in world coordinate to (*u*, *v*) in the original image needs to be calculated separately for $x>0$, $x<=0$ and so does *y*. Cause vanishing point is not necessary at the center of the image.

8. Lighting designing

Three types of light (point, spot and parallel) are predefined for the user to choose. Lights are defined inside `displayList()` using `glLightfv(source, parameter, pointer_to_array)` and `glLightf(source, parameter, value)` to set the parameters.

9. Key word setting to start animation

An method `anim()` is written to define the animation route. Inside this method, the cameral is set to move in x, y, -x, -y and z directions randomly. The speed in z direction is faster than that in x, y directions to ensure that the camera is going forward. Another method called `activateAnim()` is written active the animation by making using of `glutIdleFunc()` to set idle callback. If the camera reaches the back face, a `NULL` value will pass to the `glutIdleFunc()` to stop animation. Inside the keyboard callback method, a key 'a' or 'A' is attached to this event.

Conclusion

The nine requirements of this project are achieved by the method discussed above. Performance for the walk-through is satisfactory due to the various optimization techniques used, such as display lists, precompiled objects, etc.