INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY BANGALORE

Assignment 1
2-D Rendering

Vaibhav Kansagara
January 27, 2019

1 PROBLEM STATEMENT

Create a 2D graphical application for rendering a touch-screen calculator.

1. The keys and output text box must be modeled using rectangles, as shown. Build a 2-dimensional geometric model for the keys and text box.
2. It is not necessary to write text using OpenGL, however when a key is pressed, its corresponding number or operator is outputted as text in the console. The box for a number key can be colored with a specific color (e.g. light gray), and operators can be colored in a different color.
3. Use keyboard keys, e.g. m or M to translate the calculator.
4. Use keyboard keys, e.g. r or R to rotate the calculator (this is different from rotating each key when pressing a key).
5. Use keyboard keys + and - to zoom in and out, respectively.
6. On clicking the number **0**, an overlay (a red rectangle) appears on the **0** key, and it rotates from -45 to 45, and back to 0.
7. Figure 1 shows the animation of a widget upon pressing the left mouse button **down** anywhere in its corresponding rectangular box. The red overlay box must be a scaled down duplicate of the selected key.

8. Generate a new action of pressing and dragging the right mouse button for moving a key using pickpoint action.

2 APPROACH

Explanation of global variables.
scale - amount by which the whole calculator is scaled by.
spin - amount by which the whole calculator is rotated by.
xpos and ypos - the current world coordinates of the cursor.
diffxpos and diffypos - the displacement of the pick-point rectangle.
left_press - whether the left mouse button is pressed or not.
right_press - whether the left mouse button is pressed or not.
press_and_release - whether the right mouse button was first pressed and then released.

Classes:
Point - stores the x,y,z coordinates of the point.
Color - stores the r,g,b values of the color
Rectangle - stores four points and color of the rectangle,text on the rectangle and two boolean values red_rotate and point_pick and left_spin of the rectangle which is the extra angle by which this rectangle has to rotate w.r.t. to it's centre.
Calculator - stores vector of rectangles.

1. Approach for task1: Generated the coordinates for all the rectangles and stores in appropriate class objects.Assigned the text to each rectangle from the hardcoded string array(details in the code).

2. Approach for task2: For this task glfwSetCursorPosCallback was used to get the screen coordinates of the cursor and then to convert this screen coordinates to world coordinates gluUnProject was used and after that a loop through all the rectangles was used to identify which rectangle was the cursor on.After which it's corresponding text was displayed.

3. Approach for task3: For this task glTranslatef(x, y, 0) was used and this was appiled to each rectangle of the Calculator object and hence all rectangles were displaced by equal amounts and thus rendering the calculator in the same shape but displaced position.Here x and y are the amount by which you want to translate the rectangle in x and y direction respectively.

4. Approach for task4: For this task glrotate(spin,0.0,0.0) was used and this was appiled to each rectangle of the Calculator object and hence all rectangles were rotated by equal amounts and thus rendering the calculator in the same shape but displaced position.Here spin is the amount by which you want to rotate the whole calculator.

5. Approach for task5: For this task glScalef(scale, scale, scale) was used and this was appiled to each rectangle of the Calculator object and hence all rectangles were scaled by equal amounts and thus rendering the calculator in the same shape.

6. Approach for task6 and task7: For this task the first thing is to determine which rectangle was left-clicked which as explained above is done by glfwSetCursorPosCallback and gluUnProject.After determining the rectangle it's boolean variable red_rotate is set to true,so in the coming iterations left_spin variable of that rectangle will change values form -45 to 45 and then back to 0 and disappear.

7. Approach for task8: For this task when the right mouse button is clicked,position of the cursor is saved and when released the displacement vector is calculated and this displacement vector is added to the position's of the rectangle which was clicked.Thus after this iteration the coordinates of the rectangle would change and the rectangle will appear in the new position.

3 How is rotating each key different from rotating entire calculator

Logic for rotating entire calculator : There are four commands which controls/govern the behaviour of the calculator:

```
glLoadIdentity()
glTranslate(x,y,0)
glScale(scale,scale,scale)
glRotate(spin,0.0f,0.0f,0.0f)
glpushmatrix()
//above matrix appiled to each rectangle in the calculator.
glpopmatrix()
```

Now every position vector will get multiplied by this matrix and hence will inhibit the effects of all four commands. The first command will load the identity matrix. The second command will shift the origin of the coordinate system to (x,y,0) point so if only these two commands were present,the calculator will translate to (x,y,0) relative to it's current position. Third command will scale the whole calculator and the Fourth command will rotate the whole calculator with the given spin.
Now the reason why whole calculator is moving because these commands are individually appiled to all rectangles. Once **glPushMatrix()** is called, copy of the current matrix(comprising the effects of four previous commands) will be loaded onto the stack and **glPopMatrix()** command will ensure that matrix is removed before we proceed to apply the same operation matrix to other rectangles.

Logic for rotating individual rectangle : Here the logic will be slightly different.The command stack will look like:

```
glLoadIdentity()
glTranslate(x,y,0)
```

```
glScale(scale,scale,scale)
glRotate(spin,0.0f,0.0f,0.0f)

glpushmatrix()

if(user has clicked this particular rectangle){
  glTranslatef(LeftBottom.getX(),LeftBottom.getY(),0.0f);    //1
  glRotatef(left_spin, 0.0f, 0.0f, 1.0f);                    //2
  glTranslatef(-LeftBottom.getX(),-LeftBottom.getY(),0.0f); //3
  glColor3f(1.0f,0.0f,0.0f);
}

glpopmatrix()
```

Now in this case the first four command will do the job as mentioned above but we want animation of the red-colored rectangle. For that to happen we need to identify which rectangle the user has clicked.To determine which button and where the user has clicked **glfwSetMouse-ButtonCallback,glfwGetCursorPos and gluUnProject** commands are used. Details can be found in the code. Once we determine which rectangle is clicked we will call 1,2,3 commands.

In opengl **glrotation()** command only performs rotation form the origin point. Hence we first need to shift the origin to the desired point of rotation. Command 1 does that. Command 2 performs the rotation. Here the rotation angle will be respect to the new coordinate system set forth by first four commands and not the absolute rotation angle.

Now we need to reverse the effects of changed coordinate system and we need to shift the origin to the point where it was before and hence the 3rd command does that.

4 How does point-picking work

Consider the case when we are not allowed to do pick-point action. In that case the code for translating,scaling and rotating the individual rectangle in the calculator would look like:

```
glLoadIdentity()
glTranslate(x,y,0)
glScale(scale,scale,scale)
glRotate(spin,0.0f,0.0f,0.0f)

glpushmatrix()
glBegin(GL_QUADS);
            glVertex3f(LeftBottom.getX(),LeftBottom.getY(),LeftBottom.getZ());
            glVertex3f(LeftTop.getX(),LeftTop.getY(),LeftTop.getZ());
            glVertex3f(RightTop.getX(),RightTop.getY(),RightTop.getZ());
            glVertex3f(RightBottom.getX(),RightBottom.getY(),RightBottom.getZ());
glEnd();
glpopmatrix()
```

Now consider the case when there is pick-point action.In my code when the user presses the right mouse button i store the coordinates and when user releases the right mouse button.Then i calculate the difference in the coordinates and add that amount of displacement to

the coordinates of the selected rectangle. Hence now the only difference is that the coordinates of the selected rectangle has changed. Everything else remains the same and the prior transformation would still be applied in the same way they were applied before.