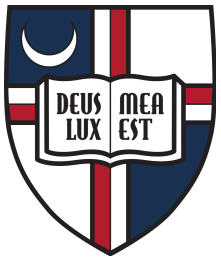
The Catholic University of America



CSC 322 Intro to Computer Graphics
Motion Parallax Landscape using OpenGL

Trung Le

Advisor: Dr. Hieu Bui

Washington D.C, November 23, 2020

Table of contents

I. Introduction:

The purpose of project is to build a 2D landscape screen via programming. The viewpoint can be manipulated by the cursor position on the screen. Components of the landscape include the sun, 2 layers of mountain, birds, the foreground with a grass field, and a tree. I used Python and OpenGL for the programming language and programming library to implement the landscape. The result is that the viewpoint can change when moving the cursor around the screen and the closer object should move faster than the further.

II. Working environment and install requirements:

- Python 3.8.2.
- OpenGL (GLUT)

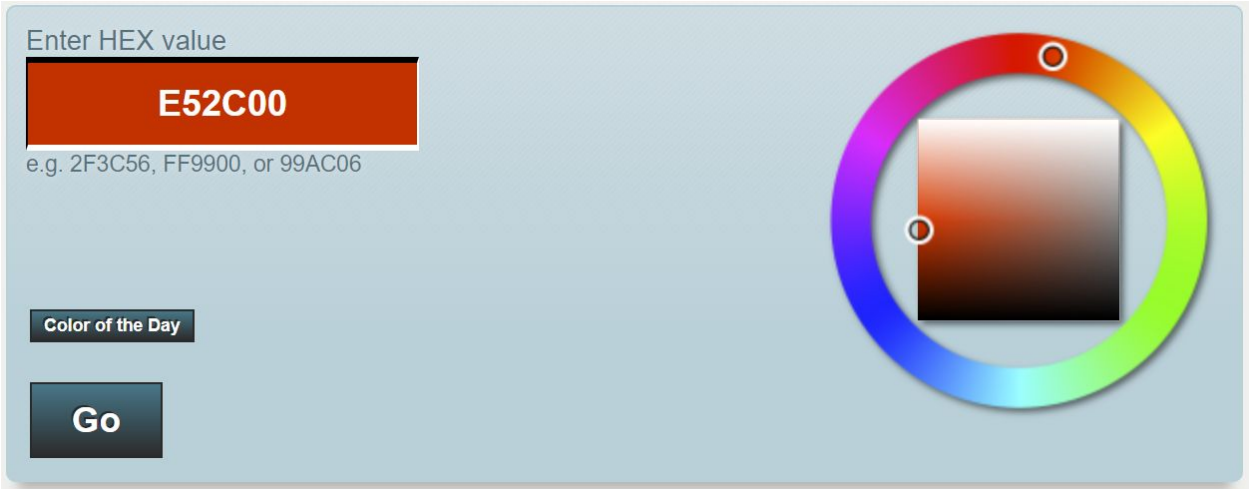
The newest PyOpenGL no longer includes GLUT so that it'll be an import error when we install a normal version using 'pip install PyOpenGL'. The solution for GLUT is using a python extension package of PyOpenGL (install the unofficial pyopengl here NOT the accelerate version). Choose the package match your system (python version, 32 or 64-bit). Remember to uninstall all of the package relevant to OpenGL before installation.

Link for wheel file of extension package: https://www.lfd.uci.edu/~gohlke/pythonlibs/#pyopengl

III. Methods:

OpenGL library provides various support for 2D geometry drawing, from filled and unfilled figure to color and width drawing lines. We should draw in the order from back to front to avoid unexpected figure overlapping. Therefore, step by step, we draw blue sky, the sun, first further mountain, two closer mountain, grass, blades, tree, and then 5 flying birds.

For the RBG color parameter applied to each object, we can use some online tools on the internet so that we can find the appropriate color and its RGB value on each channel.
The tool I used for color value tunning: http://www.perbang.dk/rgb/



This tool allows us to transfer from color to RGB hex value and vice versa. The maximum value for each channel is 255 (8 bits for each channel, 24 bits in total).

1. Initial global variable:

First of all, we need to initialize the global variables.

```
# init global variables
dimension = 800
width = dimension
height = dimension
layer0 = [0,0]
layer1 = [0,0]
layer2 = [0,0]
layer3 = [0,0]
color_1 = (random.random(), random.random(), random.random())
color_2 = (random.random(), random.random(), random.random())
color_3 = (random.random(), random.random(), random.random())
bird_move = 0
```

The canvas that the landscape has a square size of 800 defined by *dimension* variable. Layer 0,1,2,3 store posion of cursor on screen with different scale factor. Layer 0 is used for the farest of objects like the sun and birds. Otherwise, layer 3 using to manipulate the foreground which is the closest one. We will discuss the scale factor of each layer in later section. Moreover, we also need to

initialize random color for 3 mountains as we don't want the mountain color change everywhen the programming interpretation invokes the loop function.

2. **Geometry function definition:**

Starting from primitive geometries such as triangles, rectangles, and ellipses.

```
def drawTriangle(x, y, width, height):
    glBegin(GL_TRIANGLES)
    glVertex2f(x-width/2, y)
    glVertex2f(x+width/2, y)
    glVertex2f(x, y + height)
    glEnd()
```

```
def drawRectangle(x, y, width, height):
    glBegin(GL_QUADS)
    glVertex2f(x, y)  # bottom left point
    glVertex2f(x + width, y)  # bottom right point
    glVertex2f(x + width, y + height)  # top right point
    glVertex2f(x, y + height)  # top left point
    glEnd()  # done drawing
```

```
def drawEllipse(x, y, radiusX, radiusY):
    glLineWidth(1)
    glBegin(GL_LINE_LOOP)
    for i in range(360):
        rad = math.pi*i/180
        glVertex2f(x + math.cos(rad)*radiusX, y + math.sin(rad)*radiusY)
    glEnd()
```

3. **Implement compounded object:**

Then implement compounded component: blue sky, the sun, mountains, foreground, and birds. The dimension of components is manually tunning so that the scene is in harmony.

a) **Blue sky:**

The blue sky is the simplest component in the landscape. We just need to draw a blue rectangle on the origin position with the same width and height of the screen.

```
def blueSky():
    # draw bluesky background
    glColor3f(135/255, 206/255, 235/255)
    drawRectangle(0 ,0 , width, height)
```

b) **Sun:**

The sun is a circle, a special shape of an ellipse that has the same horizontal and vertical radius. So that we only need to provide a radius parameter to draw a circle. RGB base color for yellow is Red and Green, no blue, hence, the value for RGB color parameter is (max, max, min)

```
def sun():
    x= (layer0[0] + 0.9*width)
    y= (layer0[1]+0.8*height)
    radius = 30
    glColor3f(1.0, 1.0, 0.0)
    drawFillEllipse(x, y, radius, radius)
    glColor3f(0.0, 0.0, 0.0)
    drawEllipse(x, y, radius, radius)
```

c) Mountain:

There are 2 layers for mountain, one further and two closer ones. The further mountain should be higher and the two left. The color for 3 mountain is set with 3 initial color global variables. Layer value is also added to the x and y position parameter.

```
def mountain():
    # draw further moutain - first layer
    glColor3f(*color_1)  # set random color 1
    drawTriangle(layer1[0]+width/2, layer1[1]+0, width*0.7, height*0.65)

    # draw 2 closer moutain - second layer
    glColor3f(*color_2)  # set random color 2
    drawTriangle(layer2[0]+width*0.2, layer2[1]+0, width*1, height*0.55)

    glColor3f(*color_3)  # set random color 3
    drawTriangle(layer2[0]+width*0.8, layer2[1]+0, width*1, height*0.55)
```

d) Foreground:

There are 3 main components on the foreground, blades, grass and tree.

Firstly, come up with blades drawing. Blades merely a green rectangle whose the height change when the cursor move up and down.

```
def blades():
    glColor3f(140/255, 251/255, 35/255)
    drawRectangle(0 ,0 , width, layer3[1]+height*0.17)
```

Secondly, the grass is a combination of multiple green rectangles. Distance between grass is equal to its width .

```
def grass():
    glColor3f(140/255, 251/255, 35/255)
    for i in range(-100, 200):
        drawRectangle(layer3[0]+ i*width/100,layer3[1]+height*0.17,width/200,height/20)
```

e) Birds:

A bird is drawed by 2 black folded lines. We need a thicker line so that the width of drawing line is set to 3. Using for loop 5 times to draw 5 bird with different position. Layer 0 (apply to the sun) is added to the flock of birds.

```
def birds():
    glColor3f(0/255, 0/255, 0/255)
    glLineWidth(3)
    global bird_move
    for i in range(5):
        glBegin(GL_LINE_STRIP)
        glVertex2f(layer0[0]+width*0.0+i*width*0.06+bird_move*width*0.01, layer0[1]+height*0.7-i*height*0.025)
        glVertex2f(layer0[0]+width*0.03+i*width*0.06+bird_move*width*0.01, layer0[1]+height*0.69-i*height*0.025)
        glVertex2f(layer0[0]+width*0.06+i*width*0.06+bird_move*width*0.01, layer0[1]+height*0.7-i*height*0.025)
        glEnd()
```

4. Mouse handler function:

This is a passive callback function invoked everywhen the system catches a cursor movement alteration. Once Glut pass to this function 2 geometric position x and y, we use it to update the horizontal and vertical value for each layer. The first element of the layer array is used to store the horizontal position and the left for the vertical position.

```
def motion_func(x, y):
    layer0[0] = x/50
    layer0[1] = y/50

    layer1[0] = x/16
    layer1[1] = y/16

    layer2[0] = x/12
    layer2[1] = y/12

    layer3[0] = x/5
    layer3[1] = y/5
```

5. Timer handler function:

glutTimerFunc is used to set up a timer for update the horizontal location of birds every period of time. We need to pass a specified number of milliseconds to trigger the callback timer function. Here I pass 50 mean the *update()* function would be invoked every 50 ms.

```
glutTimerFunc(50, update, 0)
```

```
def update(value):
    global bird_move
    bird_move+=1
    if bird_move > 150:
        bird_move = -50
```

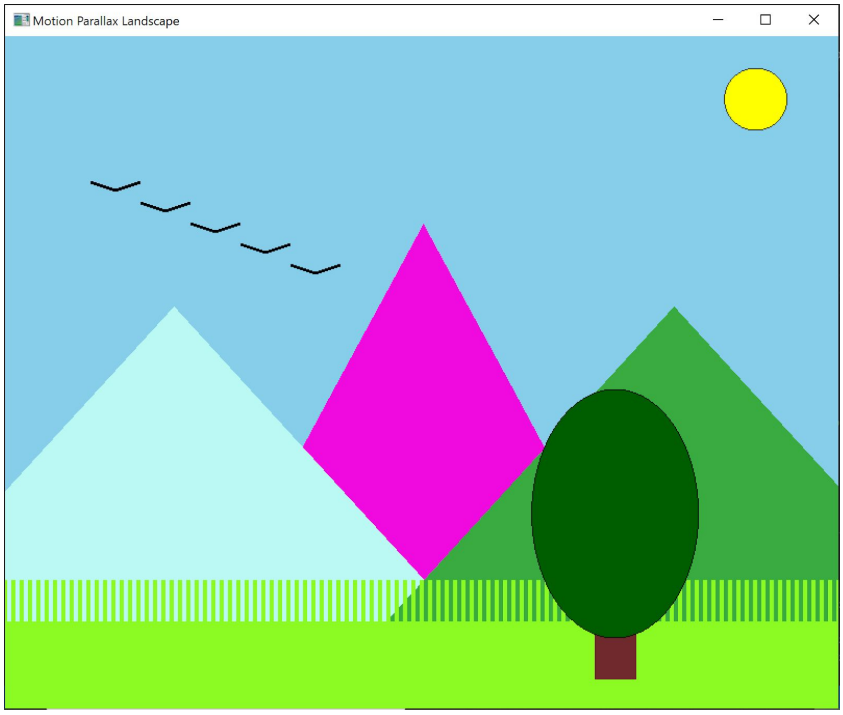
6. Combine all together:

```
def landscape():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)  # clear the screen
    glLoadIdentity()  # reset position
    refresh2d(width, height)

    blueSky()
    sun(30)
    mountain()
    Foreground()
    birds()

    glutSwapBuffers()
```

IV. Result:



V. Conclusion:

This project help me have experience on implementing a 2D scene via programming. Tkinter and OpenGL are both efficient library to draw 2D screen on PC. However, Tkinter is a robust tool specifically for create a Graphical User Interface. Therefore, OpenGL is a appreciated library for this particular project.

Github link for grading purpose:

References:

https://github.com/tinmarr/Motion-Parallax-Nifty-Problem/blob/master/sketch.js#L9

http://nifty.stanford.edu/2019/dicken-motion-parallax/specification.html