1 Administrative Topics

- Show some proj 2 write-ups. They’re great!

2 Drawing a Night/Day Scene

Let’s draw a scene with ground, sky, and a house. We want daytime and nighttime versions of the scene. But it will be the same scene. So let’s use the same code, but use if-statements to draw objects different colors based on the time of day.

```python
import turtle
import shapes

# main code
turtle.tracer(False)
daytime = True

# sky
if daytime:
    shapes.rectangle(-300, 0, 600, 300, 'DeepSkyBlue1', True)
else:
    shapes.rectangle(-300, 0, 600, 300, 'blue4', True)

# ground
if daytime:
    shapes.rectangle(-300, -300, 600, 300, 'LawnGreen', True)
```
else:
    shapes.rectangle(-300,-300, 600, 300, 'DarkGreen', True)

turtle.ht()  # Make the turtle disappear!
turtle.update()  # Force turtle to draw everything you told it to draw
raw_input("Press Enter")

We can run this code in its current form to see a day scene (see Figure 1) or change one line:
daytime = False

and see a night scene (see Figure 2).

Figure 1: Day Background
Figure 2: Night Background
2.1 Adding the House

We want to draw a house. If it includes lots of shapes, and we think we might want to draw more than one house, we should place it in a function.

We are going to use lots of parameters for the house, because we want to allow the user to specify its overall size, its color, and the colors of the windows. The code for the house is

```python
# x, y is bottom left hand corner of house
def house(x, y, width_scale, height_scale, base_color, window_color):
    shapes.rectangle(x, y, width_scale, height_scale, base_color, True)
    shapes.triangle(x, y+height_scale, width_scale, 'black', True)
    for i in range(2):
        for j in range(2):
            shapes.rectangle(x+0.2*width_scale+0.4*i*width_scale,
                             y+0.2*height_scale+0.4*j*height_scale,
                             0.2*width_scale,
                             0.2*height_scale,
                             window_color, True)
```

Notice the nested for loops used for drawing the windows. This allows us to draw a 2D grid of windows. We have two columns and two rows. The position of each window is an offset from the bottom left corner of the bottom left window. The x-position is a function of \(i\), which varies in the outer loop. The y-position is a function of \(j\), which varies in the inner loop. We will talk more about loops on Friday.

Once we add the lines

```python
# house
if daytime:
    house(0, 0, 100, 200, 'DeepPink2', 'Medium Turquoise')
else:
    house(0, 0, 100, 200, 'DeepPink2', 'gold')
```

to the main code, we see the picture in Figure 3 when daytime is True and the picture in Figure 4 when daytime is False.
Figure 3: Day Scene
Figure 4: Night Scene
2.2 Redesigning the Code

Since the only difference between night and day is color, then let’s take advantage of that. Instead of having multiple if statements and instead of using nearly identical lines of code for the daytime shapes and the nighttime shapes, let’s introduce variables for the colors and use those. Let’s have one if-statement at the beginning of the code that stores colors for each shape, based on whether it is nighttime or daytime. Then, we use those variables when we actually draw the shapes.

This means we introduce `sky_color`, `ground_color`, and `window_color` in the main section of the code.

```python
# main code
turtle.tracer(False)
daytime = False

if daytime:
    sky_color = 'DeepSkyBlue1'
    ground_color = 'LawnGreen'
    window_color = 'Medium Turquoise'
else:
    sky_color = 'blue4'
    ground_color = 'DarkGreen'
    window_color = 'gold'

# sky and ground
shapes.rectangle(-300, 0, 600, 300, sky_color,True)
shapes.rectangle(-300,-300, 600, 300, ground_color,True)

# house
house(0, 0, 100, 200, 'DeepPink2',window_color)

turtle.ht()  # Make the turtle disappear!
turtle.update()  # Force turtle to draw everything you told it to draw
raw_input("Press Enter")
```

The results from calling this code are identical to those in Figures 3 and 4.
2.3 Making the scene movable and scalable

What if we want to draw this scene more than once? Say, a day version next to a night version? We should put its code into a function. We will call it `scene1`. The input to that function should include the position and a scale factor. In our case, we also want to include a parameter indicating whether it is day or night time.

The current scene is written with the “absolute” positions, i.e. we place the hours at (0,0), which means its bottom left corner is at the center of the window. We need to make these positions relative to a given location. In other words, all of our positions become offsets from the given scene position. These offsets will be the easiest to calculate if we mandate that the position of a scene is the position of the center of the scene.

We must also scale every shape. That means we must scale both the sizes (e.g. edge lengths) and any offsets.

For example

```python
shapes.rectangle(-300,0,600,300,sky_color,True)
```

becomes

```python
shapes.rectangle(x=scale*300,y=0, scale*600, scale*300, sky_color, True)
```

In a nutshell the steps are:

1. Create a function definition that includes parameters for the position and scaling.
2. Cut the code from the main program and paste it within the function (don’t forget to indent)
3. Go through the code, adding x and y to the positions and multiplying any offsets or lengths by the scale factor.

When I choose which code to copy from the main program, I choose all code that draws and none of the set-up or final code.

For this example, the result is:
# Stephanie Taylor
# simple_scene.py

import turtle
import shapes

# x, y is bottom left hand corner of house
def house(x, y, width_scale, height_scale, base_color, window_color):
    shapes.rectangle(x, y, width_scale, height_scale, base_color, True)
    shapes.triangle(x, y+height_scale, width_scale, 'black', True)
    for i in range(2):
        for j in range(2):
            shapes.rectangle(x+0.2*width_scale+0.4*i*width_scale ,
                             y+0.2*height_scale+0.4*j*height_scale ,
                             0.2*width_scale ,
                             0.2*height_scale ,
                             window_color, True)

# center of scene at x, y
def scene1(x, y, scale, daytime):
    if daytime:
        sky_color = 'DeepSkyBlue1'
        ground_color = 'LawnGreen'
        window_color = 'Medium Turquoise'
    else:
        sky_color = 'blue4'
        ground_color = 'DarkGreen'
        window_color = 'gold'

    shapes.rectangle(x-scale*300,y+0, scale*600,scale*300, sky_color , True)
    shapes.rectangle(x-scale*300,y-scale*300, scale*600,scale*300, ground_color , True)

    # house
    house(x+0,y+0,scale*100,scale*200, 'DeepPink2',window_color)

# main code
turtle.tracer(False)

scene1(0, 0, 1.0, True)

turtle.ht()  # Make the turtle disappear!
turtle.update()  # Force turtle to draw everything you told it to draw
raw_input("Press Enter")

We can now update the main code to draw night and day scenes next to each other:

```
scene1(-150, 0, 0.5, True)
scene1(150, 0, 0.5, False)
```

and the output is in Fig. 5.

Figure 5: Two scenes at once!