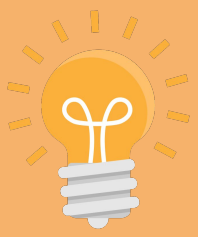


Unit 6: Graphics

1. What do I want my life to be like in 10 years from now.?
    a. In 10 years from now, I see myself finishing up with my doctorate and taking my certification exams to become a medical physicist. I would probably be working on my dissertation to finish doctoral school.
2. What am I willing to do to get there?
    a. To get there, I will apply for scholarships so I will have financial help along with maintaining excellent grades in school.
3. What colleges offer the type of education I need?
    a. The colleges that offer the type of education I need are Yale University and Hofstra University. I've decided to go with Yale because Yale has the programs I need all the way to a doctorate degree and residential program. Hofstra has the same but I plan to go away for college and Yale holds a high prestige in a person's accolades.
4. Look at the Education ROI site
    a.
5. Determine what colleges and careers are for you?
    a. The career I've chosen is medical physicist and it requires a Bachelor's in a physics-related topic which I've chosen Applied Physics to be that topic. A Master's Degree and Doctoral Degree is needed in Medical Physics after that.
6. How much have I save to go to college?
    a. I don't know how much I would have to save because I plan on being financially funded to go to college.
7. How much would I have to borrow to afford the college of my choice?
    a.

I don't know how much I would have to save because I plan on being financially funded to go to college. Same thing applies for

Unit 5 Reflection

to divider

What I Learned

1. EarSketch Basics:
    a. How to set up and use the EarSketch platform
    b. Basic programming concepts applied to music creation
    c. Using functions like setTempo(), fitMedia(), and setEffect()
2. Music Theory and Composition:
    a. Concepts of rhythm, tempo, pitch, and musical form
    b. Creating different sections (e.g., A-B-A form) in a composition
    c. Understanding copyright and fair use in music
3. Advanced Programming Techniques:
    a. Creating custom functions for more efficient code
    b. Using loops and string operations in music programming
    c. Implementing effects and envelopes to enhance sound
4. Sound Manipulation:
    a. Using makeBeat() for creating rhythmic patterns
    b. Applying various audio effects and understanding their parameters
    c. Recording and uploading custom sounds

Challenges

- Debugging: Learning to identify and fix syntax, runtime, and logic errors in code.
- Balancing creativity with technical requirements: Combining musical ideas with programming constraints.
- Understanding complex concepts: Grasping more advanced topics like envelopes, filtering, and pitch manipulation.

Questions

How can I further optimize my code for more complex musical compositions?

When will learn len and break so we can use it in our assignments?

6.1: 12/3/2024

⬅️ BACK to divider

- Simplegui is the module that allows you to execute code with color and visualizations
- GUI stands for graphical user interface
- Simplegui.create_frame is used to create a frame variable
- **Draw Handler Function:**
- def draw_handler(canvas):
- backg = "RGB(" + str(r) + "," + str(g) + "," + str(b) + ")"
- frame.set_canvas_background(backg)
- **Creating and Running the Frame:**
- frame = simplegui.create_frame("Colors", 500, 500)
- frame.set_draw_handler(draw_handler)
- frame.start()
- **RGB Colors:**
  - Colors are represented in RGB format, where each component (Red, Green, Blue) ranges from 0 to 255.
  - Example RGB values:
  - Red: (255, 0, 0)
  - Green: (0, 255, 0)
  - Blue: (0, 0, 255)
  - White: (255, 255, 255)
  - Black: (0, 0, 0)
- **Binary Representation:**
  - Each color component uses an 8-bit binary representation:
  - Maximum value (255): 11111111
  - Minimum value (0): 00000000
- **Hexadecimal Color Format:**
  - Colors can also be represented in hexadecimal format (e.g., #45ABFF).
  - Each pair of hex digits represents the intensity of Red, Green, and Blue.
- **Using Color Names:**
  - SimpleGUI allows using common HTML color names like "Red", "Blue", "Green", etc.
- RGB(red,green,blue)

6.2: 12/3/2024

to divider

- The following code demonstrates how to use SimpleGUI to draw text with varying colors:

```
import simplegui
def draw_handler(canvas):
    r = 0
    g = 0
    b = 0
      for r in range(256):
        color = "rgb(" + str(r) + ", " + str(g) + ", " + str(b) + ")"
        canvas.draw_text("Python", (50, 50 + r), r / 2 + 10, color)
frame = simplegui.create_frame("Python", 500, 400)
frame.set_draw_handler(draw_handler)
frame.start()
```

- Draw_handler function^**:**
- The draw_handler(canvas) function is defined to handle drawing operations on the canvas.
- It uses a loop to change the color and position of the text "Python" as it is drawn multiple times on the canvas.
- **Canvas and Coordinate System:**
    - The canvas starts at coordinate (0, 0) in the upper left corner,
    - As the x-coordinate increases, it moves right; as the y-coordinate increases, it moves down.
- **Drawing Text:**
    - The canvas.draw_text() method draws graphical text on the canvas.
    - Parameters include:
    - Text string: "Python"
    - Starting coordinates: (50, 50 + r)
    - Font size: r / 2 + 10
    - Color: color
- **Looping Effects:**
- The loop iterates over a range of values (0 to 255), creating a gradient effect where the text color transitions from black to red and increases in size.
- **Continuous Drawing:**
- The draw handler function runs in a loop until stopped, continuously updating what is displayed on the screen based on the defined drawing logic.
-

6.3: 12/3/2024

to divider

- In SimpleGUI, the coordinate system has its origin at the top-left corner of the canvas.
- X values increase as you move right, and Y values increase as you move down.
- This differs from traditional coordinates where the origin is typically at the center, and Y values decrease as you move down.
- **Advantages of This System:**
  - No need to deal with negative numbers.
  - Only integer values are used for coordinates; decimal values (e.g., (2.5, 3)) are invalid.
- **Creating a Frame:**
  - The simplegui.create_frame() command initializes a frame on the screen with three parameters:
    - Title (string)
    - Width (integer)
    - Height (integer)
- frame = simplegui.create_frame("Dots", 400, 400)
- **Defining Functions:**
- Use the def keyword to define functions in Python. Choose descriptive names for clarity.
- **Drawing Points on Canvas:**
  - To draw points, use the draw_point() method. It requires two arguments:
    - The point's (x, y) coordinates
    - The color of the point
- import simplegui
- def draw_handler(canvas):
- canvas.draw_point([100, 100], "Red")
- frame = simplegui.create_frame("Dots", 400, 400)
- frame.set_canvas_background("White")
- frame.set_draw_handler(draw_handler)
- frame.start()
-
- **Visualizing Multiple Points:**
- Drawing one point may not be noticeable; however, drawing multiple points can create shapes and patterns on the canvas.

6.4: 12/3/2024

to divider

```
import simplegui
import random
def draw_handler(canvas):
    # Drawing lines
    canvas.draw_line([100, 200], [200, 0], 5, "black")
    canvas.draw_line((200, 0), (300, 100), 5, "red")
    canvas.draw_line([300, 100], [100, 200], 5, "green")
        # Drawing a polygon
    canvas.draw_polygon([(100, 200), (200, 0), (300, 100)], 2, "blue", "yellow")
frame = simplegui.create_frame("Lines", 400, 400)
frame.set_canvas_background("White")
frame.set_draw_handler(draw_handler)
frame.start()
```

- **Drawing Lines**
  - Function: canvas.draw_line()
  - Parameters:
  - point1: Starting point as a coordinate pair (x, y).
  - point2: Ending point as a coordinate pair (x, y).
  - line_width: Width of the line (integer).
  - line_color: Color of the line (string).
- Example: canvas.draw_line([100, 200], [200, 0], 5, "black")
- **Notes:**
- coordinate pairs use either parentheses (x, y) or square brackets [x, y].
- Lines drawn later in the draw function will overlap earlier drawings, similar to drawing on paper.
- Drawing Polygons
- Function: canvas.draw_polygon()
- Parameters:
- point_list: A list of points that define the vertices of the polygon.
- line_width: Width of the lines connecting the points (integer).
- line_color: Color of the polygon's outline (string).
- fill_color: Color to fill the inside of the polygon (string).
- Example:
- canvas.draw_polygon([(100, 200), (200, 0), (300, 100)], 2, "blue", "yellow")
- Notes:
- The function connects the specified points in the order given and can fill the interior with a specified color

6.6 - 6.8: 12/3/2024

to divider

canvas.draw_circle((x,y), circle_radius, line_thickness, line_color, fill_color)

This command takes four parameters, plus one optional parameter. The first is the location of the center of the circle. We also include parameters defining the radius of the circle, the thickness of the line, and the color of the line. The fifth, optional, parameter is the fill color, which lets us color the interior of the circle with a color of our choosing.

*There is no built in command to create part of a circle, such as a semicircle or an arc. Instead, the next best option is to cover up the part of the circle we don't want by adding another shape or line.

You can create new shapes by covering up parts of shapes with other functions.
Example: cover a circle with a polygon to make a semicircle/smile

Scope: the lifetime of the variable in a program, including both duration and location
A global variable makes the x value on the outside and inside of the function be treated as the same.

```
x = 5
def draw(canvas):
    global x
    x = x + 5
    canvas.draw_circle((x, 200), 50, 1, color, color)
```

This code created a moving circle animation

Unit 6 Assignment Explanation

to divider

Frame count variable:
Goes up by 1 every time the draw_scene function is called. Its the backbone of the animation

Sun:
sun_y = 100 + 20 * abs((frame_count % 120) - 60) / 60
canvas.draw_circle((50, sun_y), 30, 2, sun_yellow, sun_yellow)

- frame_count % 120 creates a cycle of 0 to 119
- Subtracting 60 shifts this to -60 to 59
- abs() makes this 60 to 0 to 60 (**60 fps**)
- Dividing by 60 scales this to a range of 0 to 1
- Multiplying by 20 makes the range 0 to 20
- Adding 100 positions the sun's center between y=100 and y=120

Clouds:

```
for i in range(2):
    cloud_x = (frame_count + i * 200) % (600 + 100) - 50
    canvas.draw_circle((cloud_x, 80), 20, 1, cloud_white, cloud_white)
    canvas.draw_circle((cloud_x + 20, 70), 20, 1, cloud_white,
cloud_white)
        canvas.draw_circle((cloud_x + 40, 80), 20, 1, cloud_white,
cloud_white)
```

- Two sets of clouds are created (i = 0 and 1)
- cloud_x calculates the x-position of each cloud set:
  - frame_count moves the clouds to the left
  - i * 200 offsets the second cloud set
  - % (600 + 100) wraps the clouds around when they go off-screen
  - - 50 starts the clouds slightly off-screen to the left
- Three circles are drawn for each cloud, slightly offset to create a cloud shape

Unit 6 Assignment Explanation

⬅ BACK to divider

Birds:

```
for k in range(5):
    bird_x = (frame_count * 2 + k * 50) % 600
    bird_y = 150 + 10 * abs(((frame_count + k * 20) % 80) - 40) / 40
    canvas.draw_line((bird_x - 10, bird_y), (bird_x, bird_y - 5), 2, "black")
    canvas.draw_line((bird_x, bird_y - 5), (bird_x + 10, bird_y), 2, "black")
```

- Five birds are created (k = 0 to 4)
- bird_x calculates the horizontal position:
  - frame_count * 2 moves the birds to the right
  - k * 50 spaces out the birds
  - % 600 wraps the birds around the screen
- bird_y calculates the vertical position:
  - Similar to the sun, but with a cycle of 80 frames
  - k * 20 offsets each bird's vertical position
- Two lines are drawn to create a simple "V" shape for each bird (how my dad draws birds)

To-do:

- ❑ Finish wonderland animation
- ❑ Submit notebook 6
- ❑ Resubmit storyboard
- ❑ Complete activity guides in CFM

Name: Micah Kennedy

Block: 1-2 Date: 12/17/2024

Unit 6: Test Review

1. Define the following:
   a. global variable one copy of the variable that is used throughout the whole program
   b. range function create a list of functions
   c. for loop a loop used for counting

2. When do you use a for loop instead of a while loop?
   -when there is a definite starting and ending point
   -you know how many times you want the loop to run

3. Write a line of code that will draw a solid box from point (108, 67) to (56, 178).

```
canvas.draw_polygon([(108,67),(108,178),(56,178),(56,67)],5,"purple","orange")
```

4. Write a line of code that will draw a green circle at the point (78, 119) with a radius of 170.

```
canvas.draw_circle((78,119),170,5,"Green")
```

5. For a horizontal line the y-axis values are the same.

6. Write a FOR loop that will draw 3 circles with a radius of 20 exactly 50 points apart in a vertical line. The first point should be (100,100).

```
for x in range (100,350,50):
    canvas.draw_circle((100,x),20,5,"green")
```

7. For the following circle statement match each letter with what it does.

```
canvas.draw_circle ((500, 300), A, B, C, D)
```

   a. fill color D
   b. line color C
   c. radius A
   d. line weight B

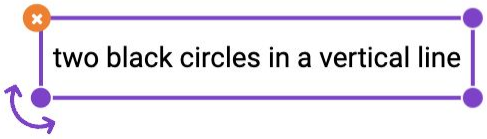
11. Rewrite the following while loop as a for loop:

```
c = 10
while (c < 20):
   c = c + 4
   print (c),
```

```
for x in range (14,24,4):
  print(x)
```

12. What does the following code draw?

```
canvas.draw_circle ((100, 75), 50, 10, "Black")
canvas.draw_circle ((100, 175), 50, 10, "Black")
```



13. What does the following code draw?

```
canvas.draw_polygon([(350, 200), (200, 450), (50, 350)], 5, "Black", "Gray")
```