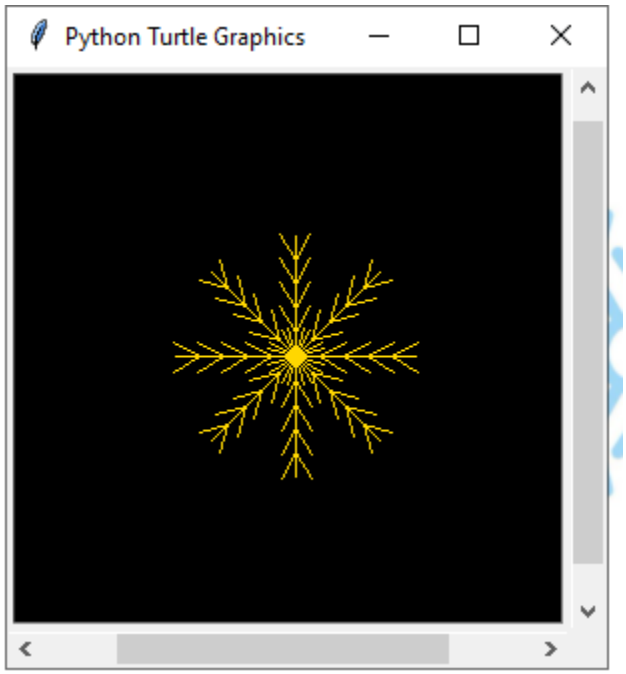
Erica Hung, Hairong Li, Gabe Butler
ATLS 1300

Final Project Pseudocode (Generative Christmas Art)

Snowflake -- Hairong

Snowflake Updates:

- **Screenshot of one random snowflake**



- **Code so far**
  - https://drive.google.com/file/d/1h1sqGybWefHdiUfizQNGG2zzUjcvNVHd/view?usp=sharing
  - **Note: ran into attribute error in SnowFlake class(commented out): “** 'Turtle' object has no attribute 'Turtle' **” or “**'Turtle' object has no attribute 'onscreenclick' **”**
  - Try to panel onclick to generate the new snowflake but not sure how it work yet because of the error
  - Haven't set up the Manager class yet

Class UMLs:

Class Name	**SnowFlakes**	**Manager**
Attributes	Color(“gold”) Size Angle Radius Position	Background color(“black”) Panel Draw
Method	snowflake branch	run() setup()

Libraries: turtle, random

Problems/Strategies:

- Randomization
- Branch number, angle, size

Tasks:

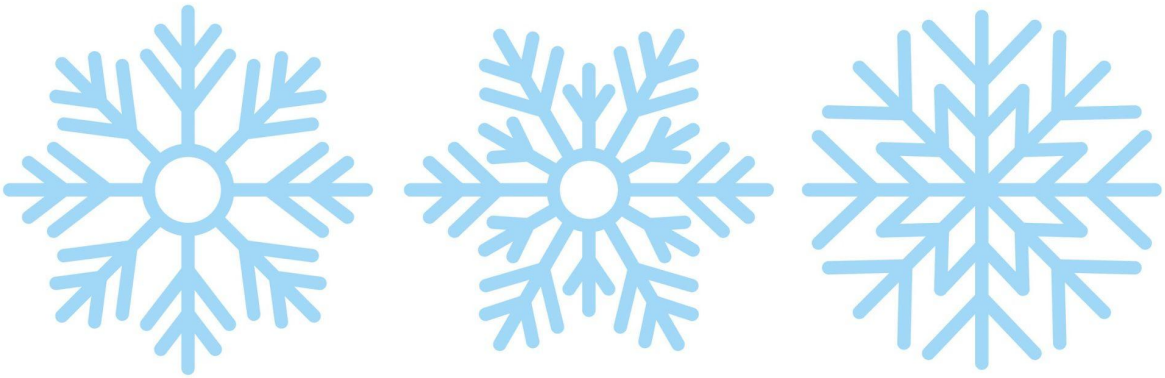
- Create SnowFlake class
    - Set the random number of sizes and radii of the snowflake
    - Draw from random choices
    - Onclick to generates new pattern and clear last pattern
- Manager class
    - Set up the panel
    - Run the animation

Steps:

- Import turtle libraries: turtle, random
- Define variables
    - branchNum
    - branchSize
    - branchGap
    - branchAngle
- Create the SnowFlake class
    - Set up location
    - Hide turtle
    - Random choice to select patel numbers
    - angle = random.choice(angles)

        - num = int(360/angle)
        - Create methods for the snowflakes pattern
            - Onclick to change the pattern
- Create the Manager class
    - setup()
        - Set up panel size and panel color
            - W = 600
            - H = 600
            - panel.bgcolor(“black”)
    - run()
        - Run the entire code and animation
- # leave the window open until click to close
    - panel.exitonclick()

Example Diagram: There will be one snowflake generated and drawed by the turtle on the screen every time the user clicks the screen, not image.



Christmas Tree -- Gabe

Class UMLs:

Class Names	**Stump(Turtle.turtle)**	**Tree(Turtle.turtle)**	**Star(Turtle.turtle)**	**Manager**
Attributes	Color (black) Position	Color (black) Position	Color (gold) Position	size() bgcolor("white") turtle.colormode(255) turtle.tracer(0)
Methods	draw()	mosaic() triangle() circle() ~~Binary~~	basic() bubbles() teardrop() ~~Spiral~~	run() setup()

edit: binary tree and spiral star are removed (the look did not go with the overall style)

Libraries: turtle, random, math

Problems/Strategies:

- Significant randomization

Tasks:

1. Create Stump (class)
    a. Create Draw (method)
        i. Creates black rectangle
2. Create Tree (class)
    a. Create tree shapes (methods)
        i. Mosaic
            1. Greek style mosaic lines in triangle shape
        ii. Triangle
            1. Basic black triangle
        iii. Circle
            1. Hoops (ovals) that get smaller in triangle shape
        ~~iv. Binary~~
            ~~1. Word art in shape of triangle (holiday phrases in binary)~~
3. Create Star (class)
    a. Create star shapes (methods)
        i. Basic

1. Regular filled star

ii. Bubbles

1. Star made of circles

iii. Teardrop

1. Star made of 5 teardrops

~~iv. Spiral~~

~~1. Spiral with star shape cut out~~

4. Generate various trees

Steps:

1. Import turtle, random, math
2. Set up panel size and panel color
    a. W = 600
    b. H = 600
    c. panel.bgcolor("white")
3. Create list of tree locations
4. class Stump(turtle.Turtle)
    a. Go to each tree location
    b. Draw 25x50 rectangle
5. class Tree(turtle.Turtle)
    a. Go to each tree location
    b. Randomly pick one of the methods
        i. Mosaic
            1. Draw line of length 100
            2. Turn left 90 degrees
            3. Forward 1
            4. Turn left 90 degrees
            5. Draw line of length 99
            6. …
            7. Continue until line length is 0
        ii. Triangle
            1. Draw triangle (of size 3?)
        iii. Circle
            1. Parametric equation of oval
            2. Draw oval
            3. Slightly decrease oval size
            4. Continue until oval size is zero
        ~~iv. Binary~~
            ~~1. Make list of holiday phrases ("Happy Holidays", "Season's Greetings") and convert to binary~~

~~2. Write out one of the phrases (randomly chosen) in a triangle shape (or write out normally and cut out triangle shape~~

6. Class Star(turtle.Turtle)
    a. Go to each tree location
    b. Randomly pick on of the methods
        i. Basic
            1. Draw simple star
        ii. Bubbles
            1. For loop
                a. Draw 3 circles, decreasing in radius (like a snowman)
                b. Go back to center
                c. Rotate to draw the next arm of the star
        iii. Teardrop
            1. Draw teardrop shape
            2. Rotate
            3. Draw next teardrop shape
        ~~iv. Spiral~~
            ~~1. Draw spiral shape~~
            ~~2. Cut out star shape~~
7. Draw ~16 random trees with a stump and a random star

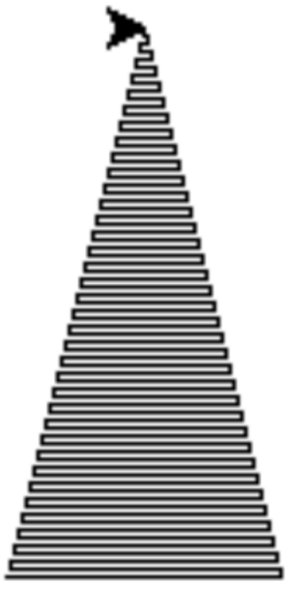
Example (will be turned into method)

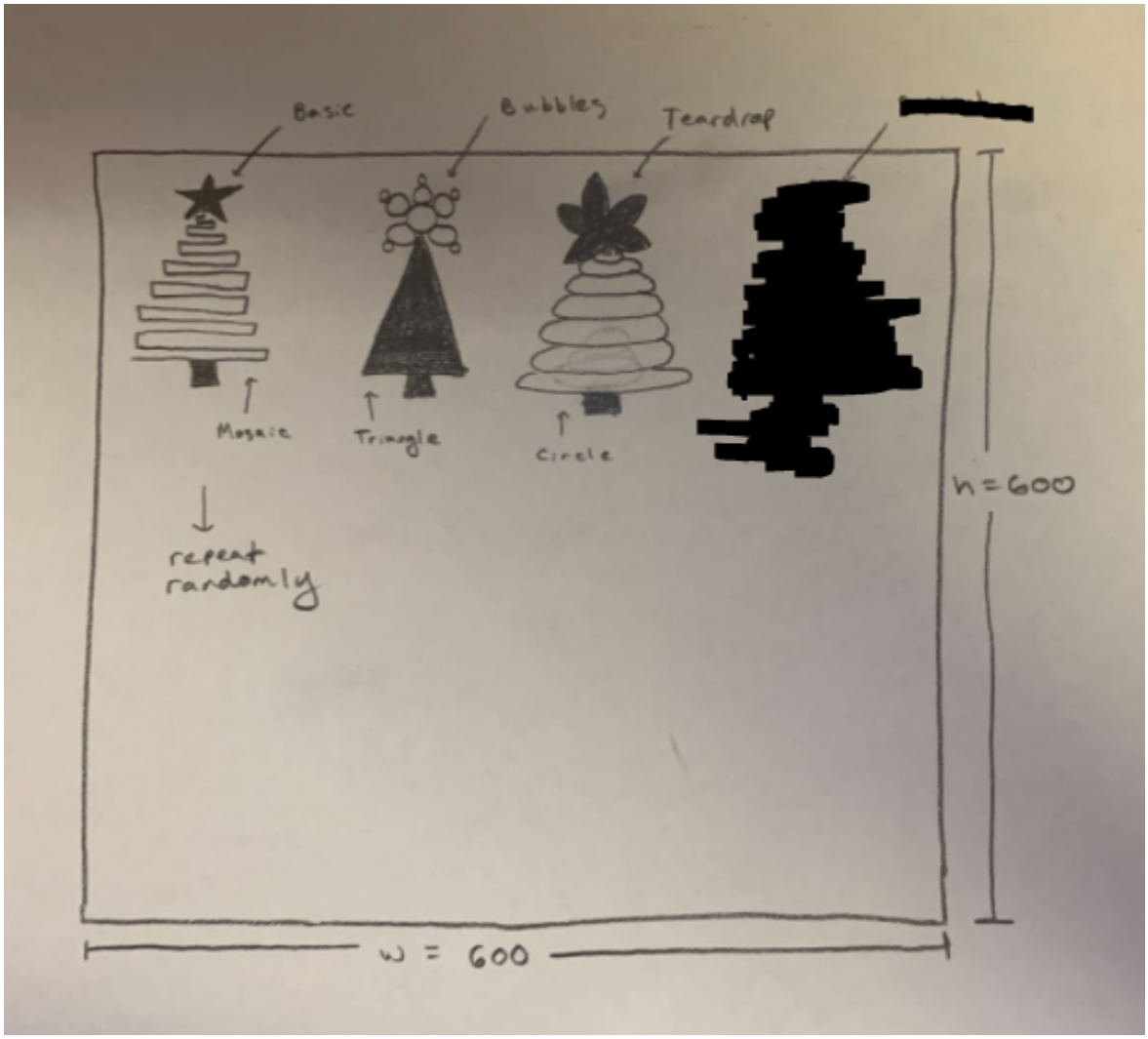
Brief Example of what *Mosaic* would look like

```
import turtle
import math
import random

count = 70
turtle.speed(0)

for i in range(35):
    turtle.forward(count)
    turtle.left(90)
    turtle.forward(2)
    turtle.left(90)
    count -= 1
    turtle.forward(count)
    turtle.right(90)
    turtle.forward(2)
    turtle.right(90)
    count -=1
```




Basic
Bubbles
Teardrop
[redacted]
Mosaic
Triangle
Circle
repeat randomly
h = 600
w = 600

Christmas Ornaments -- Erica

Class UMLs:

Class Names	**Lights**	**Ornaments**	**Manager**	**Patterns**
Attributes	Color (white) Shape Position	Color (white) Shape Position	Panel (black) Click interaction	Color (gold) Lines Dots Position
Methods	changeLightColor()	changeOrnColor()	setup() run() turtle.onclick()	changePattern()

Libraries: turtle, random

Problems/Strategies:

- Randomization
- User interaction
- Placement of ornaments, lights, and patterns

Tasks:

1. Create the lights, ornaments, patterns, and manager classes
2. Add a user interaction (clicking)
3. Create a position list for lights, ornaments, patterns

Steps:

1. Import turtle, random
2. Create the manager class
    a. Set up the panel
        i. Set up panel size and panel color
            1. W = 600
            2. H = 600
            3. panel.bgcolor(“black”)
    b. Set start color and end color
        i. Start_color = “black”
        ii. End_color = “white”
    c. Run the animation
    d. Write the text

        i. “Happy Holidays! Click the screen!”
    e. Set up click interaction
        i. Allows user to click the panel to change the background color
            1. Panel color inverts (black & white)
    f. Create ornament positions list
    g. Create the lights positions list
    h. Create the string positions list
    i. Create the patterns positions list
        i. Create a positions list for dots pattern
3. Create the patterns class
    a. Set turtle color to “gold”
    b. Find coordinate range to place pattern in
    c. Turtle go to min (x,y), move forward to max (x,y) (using the patterns position list)
        i. Create lines throughout the ornament
            1. **Example:**
                a. turtle.goto(0,10), turtle.fd(50)
                b. turtle.goto(0,20), turtle.fd(50)
                c. turtle.goto(0,30), turtle.fd(50)
    d. Make turtle circle shape and stamp to create a dots pattern
        i. Use the dots positions list
4. Create the ornaments class
    a. Send turtle to ornament positions using the ornaments positions list
    b. Create ornament body:
        i. turtle.circle(60)
    c. Create the top part of ornament:
        i. For i in range(4):
            1. turtle.fd(30)
            2. turtle.rt(90)
    d. Add changeOrnColor()
5. Create the lights class
    a. Create light body:
        i. curve()
        ii. turtle.left(100)
        iii. curve()
        iv. turtle.fd(120)
        v. turtle.goto(lightPosition[x])
    b. Create the string for light to hang from:
        i. turtle.goto(stringPosition[x])
    c. Add changeLightColor()