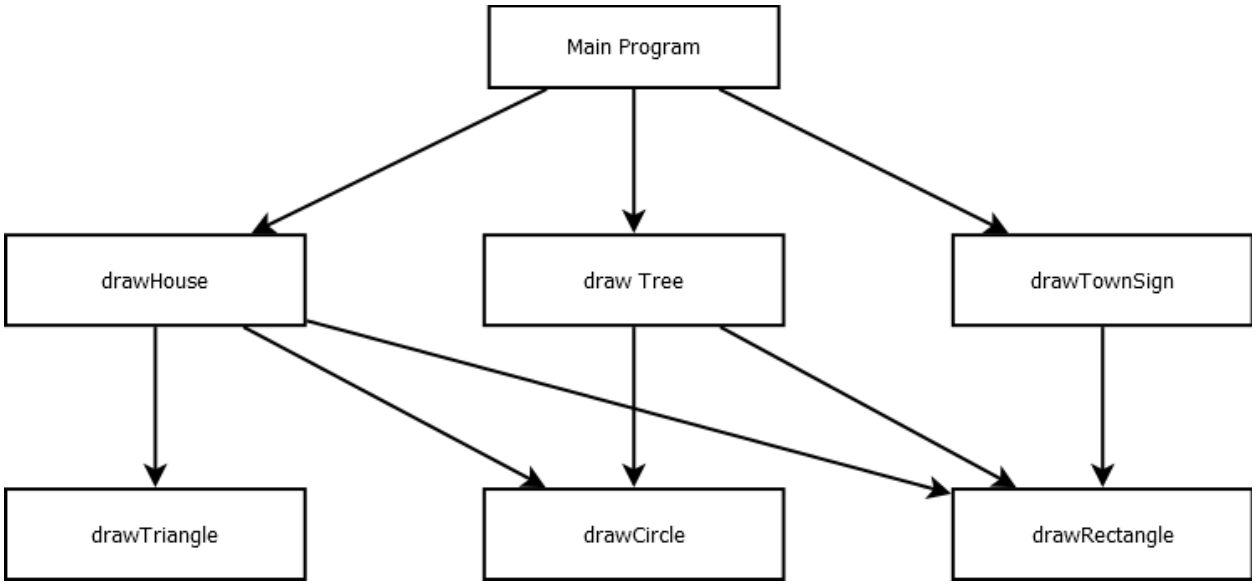
Neighborhood
Lab 8

Jonathan Finn, jpf212
Jackson de Gruiter , jtd431

"On my honor, as a Mississippi State University student, I have neither given nor received unauthorized assistance on this academic work."

CSE 1284 Introduction to Computer Programming
Lecture Section Time: MWF 10am
Instructor: Monika Jankun-Kelly
Lab Section: Friday
Lab Assistant: Tyler Narmore

Program Design



Functions:

drawTriangle()
    Draw a triangle depending on parameters

Parameters
---
x : int
    x coordinate of the anchorpoint
y : int
    y coordinate of the anchorpoint
tri_base : int
    the length of the triangle base
tri_height : int
    the height of the triangle
color : str
    the color of the triangle

Returns

None

##############################

drawRectangle()
    Draw a rectangle depending on parameters

Parameters
---
x : int
    x coordinate of the anchorpoint
y : int
    y coordinate of the anchorpoint
rec_width : int
    the length of the rectangle base
rect_height : int
    the height of the rectangle
color : str
    color of the rectangle

Returns

None

##############################

drawCircle()
    Draw a circle depending on parameters

Parameters
---
x : int
    x coordinate of the anchorpoint
y : int
    y coordinate of the anchorpoint
radius : int
    radius of the circle
color : str
    color of the circle

Returns
---
none

```
#############################

drawTree()
        Draw a tree depending on parameters

Parameters
----------
x : int
        x coordinate of the anchorpoint
y : int
        y coordinate of the anchorpoint
tree_height : int
        height of the tree
color : str
        color of the canopy of the tree

Returns
-------
None

#############################

drawHouse()
        Draw a house depending on parameters

Parameters
----------
x : int
        x coordinate of the anchorpoint
y : int
        y coordinate of the anchorpoint
house_width : int
        width of the house
house_height : int
        height of the house
primary_color : str
        color of the building
secondary_color : str
        color of the roof
```

Returns

None

##############################

drawTownSign
    Draw a sign with the town name depending on parameters

Parameters
---
x : int
    x coordinate of the anchorpoint
y : int
    y coordinate of the anchorpoint
text : str
    text to be printed on the sign

Returns

None

Start Drawing the Neighborhood
Read each file
read the first line
Add each line of the text file to text_array
    remove newline
    append the line to text_array
    read the next line
Look at each segment of text in the text array, and figure out what to do with it
    Make the string into an array of smaller strings
    Replace underscores with spaces
    If the first word is house, draw a house
    If the first word is tree, draw a tree
    If the first word is not one of the things above, print an error message
Draw the town sign last

Addition to File format:
We didn’t need to any any new information to the provided text file. The sign is the unique thing we added, but because the length of the sign is determined by the length of the provided neighborhood name, no new information is needed to draw it.

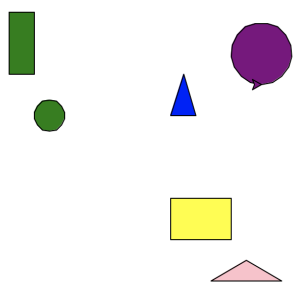
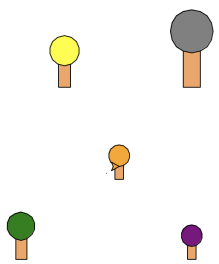
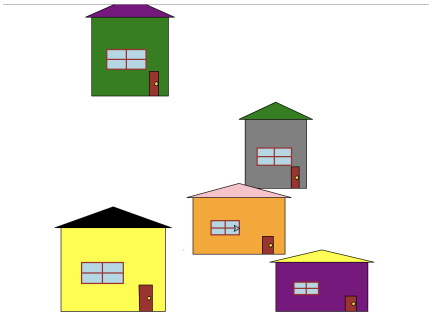
File format:
name of neighborhood
house x y width height wallColor roofColor
tree x y height leafColor

Testing

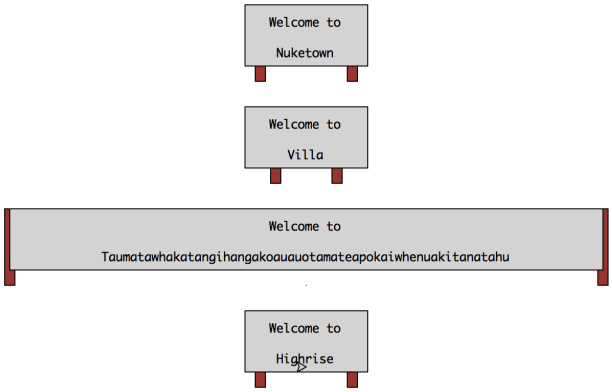
Testing Plan for the turtle graphics:

- We’re going test each function in various locations to make sure they works.
- Each function should be tested multiple times using different parameters each time before the final test.
- The final test should include all of the functions.

Turtle Graphics Screenshots:

Basic Shapes:	Trees:	Houses:
		

Town Sign:



Testing Plan for reading input from a text file:

- Try to print each line, to make sure the data we're getting is correct.
- Try to get each word in the string by itself, and make sure they can be used by the code.

Reading text file Screenshots:

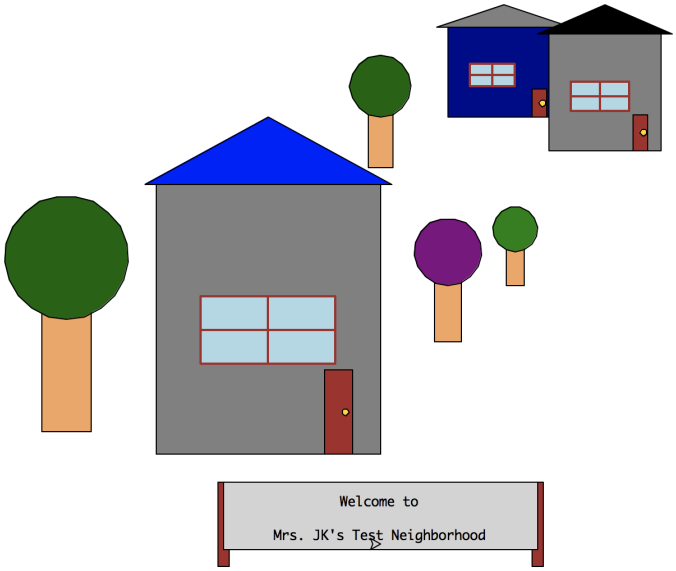
Printing Each line screenshot:

```
Mrs. JK's Test Neighborhood
house -200 -200 200 300 grey blue
house 60 100 100 100 dark_blue grey
house 150 70 100 130 grey black
tree -280 -180 220 dark_green
tree 0 55 110 dark_green
tree 60 -100 120 purple
tree 120 -50 80 green
```

Printing each word list screenshot:

```
['Mrs.', "JK's", 'Test', 'Neighborhood']
['house', '-200', '-200', '200', '300', 'grey', 'blue']
['house', '60', '100', '100', '100', 'dark blue', 'grey']
['house', '150', '70', '100', '130', 'grey', 'black']
['tree', '-280', '-180', '220', 'dark green']
['tree', '0', '55', '110', 'dark green']
['tree', '60', '-100', '120', 'purple']
['tree', '120', '-50', '80', 'green']
```

Final Screenshot



Analysis and Conclusions

Reflection Questions

1. List at least two things you did to learn about file IO. What did you find most helpful when studying file IO?

- We learned how to use the contents of a text file as parameters for our functions
- We learned how to string newline characters when reading input from a text file.

We looked at past in class examples and researched handling text files online. It was most helpful to review the in class examples so that we could understand how it works in context of a whole program

2. List at least two difficulties or errors you ran into while doing this lab. How did you overcome or fix them? If it was an error, give the error message.

- One difficulty we ran into was trying to make sure that the text on the sign was using a font that had the same number of pixels for each letter so that we could use a standard sizing for the sign. We tested a few fonts until we finally found one that worked for all different text input
- Another difficulty was making sure the tree trunk origin point as in the middle of the rectangle, but we solved that with trial and error

3. Describe two or three skills or concepts you learned or strengthened by doing this lab

- We strengthened our ability to effectively use functions in order to shorten the code
- We learned much more about how to handle file IO

References

functions:
Tony Gaddis book, chapter 5
Think Python Ch 3

files: opening, reading, and closing

Three ways to read a text file line by line in python

How to handle plain text files in python 3

Turtle:
Turtle Documentation

string manipulation:

break up a string/line into parts/tokens, split() function
Tony Gaddis book, Chapter 8.3, Page 512-513

replace part of a string, replace() function
Tony Gaddis book, Chapter 8.3, Page 507-508

Code Appendix

```
from turtle import *
import math
import time

setup(1000,1000)
goto(0,0)
penup()

##############################################
# drawTriangle()
#       Draw a triangle depending on parameters
#
# Parameters
# -----------
# x : int
#     x coordinate of the anchorpoint
# y : int
#     y coordinate of the anchorpoint
# tri_base : int
#     the length of the triangle base
# tri_height : int
#     the height of the triangle
# color : str
#     the color of the triangle
#
# Returns
# -------
# none

def drawTriangle(x, y, tri_base, tri_height, color):
    # Calculate all the measurements and angles needed to draw the
triangle
```

```
    side_length = math.sqrt((0.5*tri_base)**2 + tri_height**2)
    base_angle = math.degrees(math.atan(tri_height/(tri_base/2)))
    top_angle = 180 - (2 * base_angle)

    # Lift pen to prevent stray lines
    penup()

    # Go to some x and y coordinates
    goto(x, y)
    setheading(0)

    # Fill the triangle with some color
    fillcolor(color)
    begin_fill()
    pendown()

    # Draw the triangle
    forward(tri_base)
    left(180 - base_angle)
    forward(side_length)
    left(180 - top_angle)
    forward(side_length)

    # Stop filling and lift pen
    end_fill()
    penup()


##############################################
# drawRectangle()
#       Draw a rectangle depending on parameters
#
# Parameters
# ----------
# x : int
#     x coordinate of the anchorpoint
```

```
# y : int
#     y coordinate of the anchorpoint
# rec_width : int
#     the length of the rectangle base
# rect_height : int
#     the height of the rectangle
# color : str
#     color of the rectangle
#
# Returns
# -------
# none

def drawRectangle(x, y, rec_width, rect_height, color):


    # Lift pen to prevent stray lines
    penup()

    # Go to some x and y coordinates
    goto(x, y)
    setheading(0)

    # Set fill color, put pen back onto canvas
    pendown()
    fillcolor(color)
    begin_fill()

    # Draw the rectangle
    for side in range(2):
        forward(rec_width)
        left(90)
        forward(rect_height)
        left(90)

    # Stop filling and lift pen
```

```
    end_fill()
    penup()


##############################################
# drawCircle()

#       Draw a circle depending on parameters
#
# Parameters
# ----------
# x : int
#     x coordinate of the anchorpoint
# y : int
#     y coordinate of the anchorpoint
# radius : int
#     radius of the circle
# color : str
#     color of the circle
#
# Returns
# -------
# none

def drawCircle(x, y, radius, color):

    # Lift pen to prevent stray lines
    penup()

    # Go to some x and y coordinates
    goto(x, y)
    setheading(0)
    setpos(x, (y-radius))

    # Put pen down, then start filling
    pendown()
```

```
    fillcolor(color)
    begin_fill()

    # Draw the circle
    circle(radius)

    # Stop filling and lift pen
    end_fill()
    penup()


##############################################
# drawTree()
#       Draw a tree depending on parameters
#
# Parameters
# ----------
# x : int
#     x coordinate of the anchorpoint
# y : int
#     y coordinate of the anchorpoint
# tree_height : int
#     height of the tree
# color : str
#     color of the canopy of the tree
#
# Returns
# -------
# none

def drawTree(x, y, tree_height, color):

    # Draw the trunk of the tree, it will always be sandy brown
    drawRectangle((x-(tree_height * 0.1)), y, (tree_height * 0.2), tree_height *
0.5, "sandy brown")
```

```
    # Draw the leafy part of the tree, make it some color
    drawCircle(x, (y + (tree_height * 0.75) - 10), (tree_height * 0.25), color)


##############################################
# drawHouse()
#       Draw a house depending on parameters
#
# Parameters
# ----------
# x : int
#     x coordinate of the anchorpoint
# y : int
#     y coordinate of the anchorpoint
# house_width : int
#     width of the hosue
# house_height : int
#     height of the house
# primary_color : str
#     color of the building
# secondary_color : str
#     color of the roof
#
# Returns
# -------
# none

def drawHouse(x, y, house_width, house_height, primary_color,
secondary_color):

    # Define some variables that will be useful for house construction
    roof_height = house_height * 0.2
    building_height = house_height * 0.8

    # Draw the building
    drawRectangle(x, y, house_width, building_height, primary_color)
```

```
    # create the variables needed to draw the door
    door_width = house_width / 8
    door_height = house_height / 4

    # Draw the door relative to the height of the building, it will be brown
    door_x = x+(house_width * 0.75)
    door_y = y
    drawRectangle(door_x, door_y, door_width, door_height, "brown")

    # Draw the doorknob, it will be gold
    drawCircle(door_x + door_width*3/4, door_y + door_height/2, 3, "gold")

    # Draw roof, it will be the secondary color
    overhang = 20
    drawTriangle(x-(overhang / 2), (y + building_height), (house_width +
overhang), roof_height, secondary_color)

    # Set the pen color to brown, this will make the window frame look nice
    pencolor("brown")
    pensize(2)

    # Draw the window, fill with light blue
    window_size = building_height / 2
    window_x = (x+(house_width * 0.2))
    window_y = y + (building_height / 3)
    drawRectangle(window_x, window_y, window_size, window_size/2, "light
blue")

    # Divide the window into 4 sections
    goto(window_x + window_size/2, window_y)
    pendown()
    goto(window_x + window_size/2, window_y + window_size/2)
    penup()
    goto(window_x, window_y + window_size/4)
    pendown()
```

```
    goto(window_x + window_size, window_y + window_size/4)

    # Change the pencolor back to black, the size back to 1, then lift the pen
    pencolor("black")
    pensize(1)
    penup()


##############################################
# drawTownSign
#       Draw a sign with the town name depending on parameters
#
# Parameters
# ----------
# x : int
#     x coordinate of the anchorpoint
# y : int
#     y coordinate of the anchorpoint
# text : str
#     text to be printed on the sign
#
# Returns
# -------
# none

def drawTownSign(x, y, text):

    # Find out the length of the letters and set the sign length accordingly
    letters = len(list(text))
    sign_width = (letters*10)+10

    # Draw the rectangular feet of the sign
    drawRectangle((x+sign_width/2-5), y, 10, 75, "brown")
    drawRectangle((x-sign_width/2-5), y, 10, 75, "brown")
```

```
    # Make sure "Welcome to" fits inside the sign, if not, change the sign
width
    if letters < 10:
        sign_width = 120

    # Draw the sign that we will place the text on
    drawRectangle(x-(0.5*sign_width), y+15, sign_width, 60, "light grey")

    # Draw the text that tells the name of the neighborhood
    setpos(x, y+50)
    write("Welcome to", align="center", font=("Monaco", 12, "normal"))
    setpos(x, y+20)
    write(text, align="center", font=("Monaco", 12, "normal"))


##################################################################
#####################
# Start Drawing the Neighborhood


# Read each file
infile = open("input_file.txt")
text_array = []

# read the first line
line = infile.readline()

# Add each line of the text file to text_array
while line:

    # remove newline
    line = line.rstrip('\n')

    # append the line to text_array
    text_array.append(line)
```

```
    # read the next line
    line = infile.readline()


# Look at each segment of text in the text array, and figure out what to do
with it
for text in text_array:
    # print(text) # debugging

    # Make the string into an array of smaller strings
    text = text.split()

    # Replace underscores with spaces
    for i in range(len(text)):
        if "_" in text[i]:
            text[i] = text[i].replace("_", " ")

    # If the first word is house, draw a house
    if text[0] == "house":
        drawHouse(int(text[1]), int(text[2]), int(text[3]), int(text[4]), text[5],
text[6])

    # If the first word is tree, draw a tree
    elif text[0] == "tree":
        drawTree(int(text[1]), int(text[2]), int(text[3]), text[4])

    # If the first word is not one of the things above, print an error message
    else:
        print("ERROR, CANNOT DRAW: " + ""+ text[0] +"")


# Draw the town sign last
drawTownSign(0, -300, text_array[0])

input()
```