Hello, World!

Harry Smith

Learning Objectives

- Write a program that draws a specified image
- Understand the model of drawing used in PennDraw
- Judge how lines of code will affect the program output

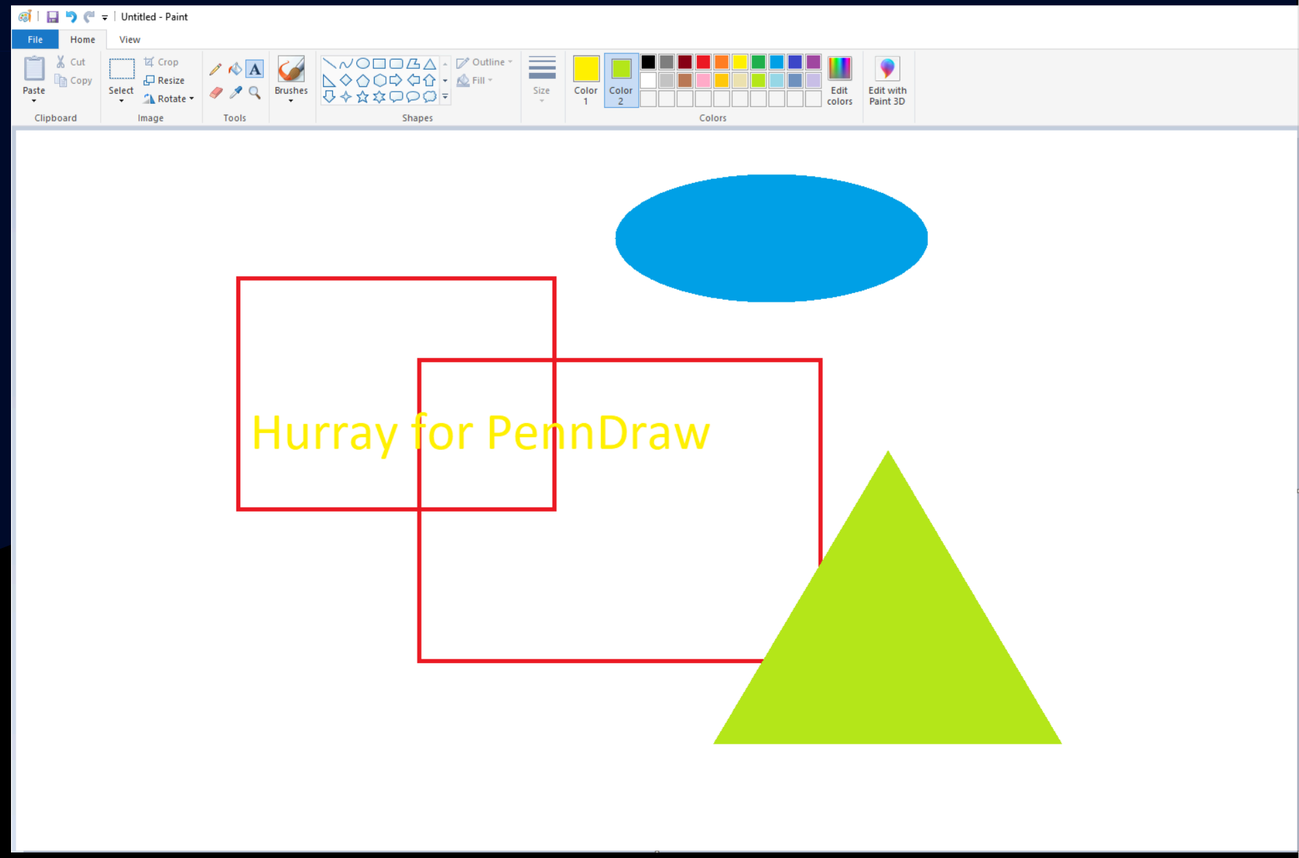
What is PennDraw?

PennDraw

- The name of a group of related drawing tools available for you to use.
  - Adapted from a library called “StdDraw” if you see that anywhere
- Any time we need to draw to the computer’s screen in CIS 1100, we’ll use PennDraw.
- You can access a full listing of PennDraw’s features
  on the page for PennDraw on the course website

PennDraw: the Model

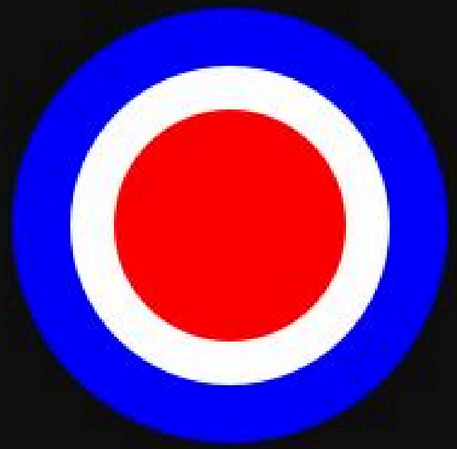
- Works like a programmable Microsoft Paint
- Uses a set canvas
- Has an imaginary “pen”
  - The pen has a color setting
    and a weight setting.
- Draw shapes
  - Rectangles, ellipses, arbitrary polygons
- Draw text



A PennDraw Program

`order_demo.py`:

```
import penndraw as pd
pd.set_canvas_size(400, 400)
pd.clear(15, 15, 15)

pd.set_pen_color(pd.BLUE)
pd.filled_circle(0.5, 0.5, 0.15)

pd.set_pen_color(pd.WHITE)
pd.filled_circle(0.5, 0.5, 0.11)

pd.set_pen_color(pd.RED)
pd.filled_circle(0.5, 0.5, 0.08)
pd.run()
```



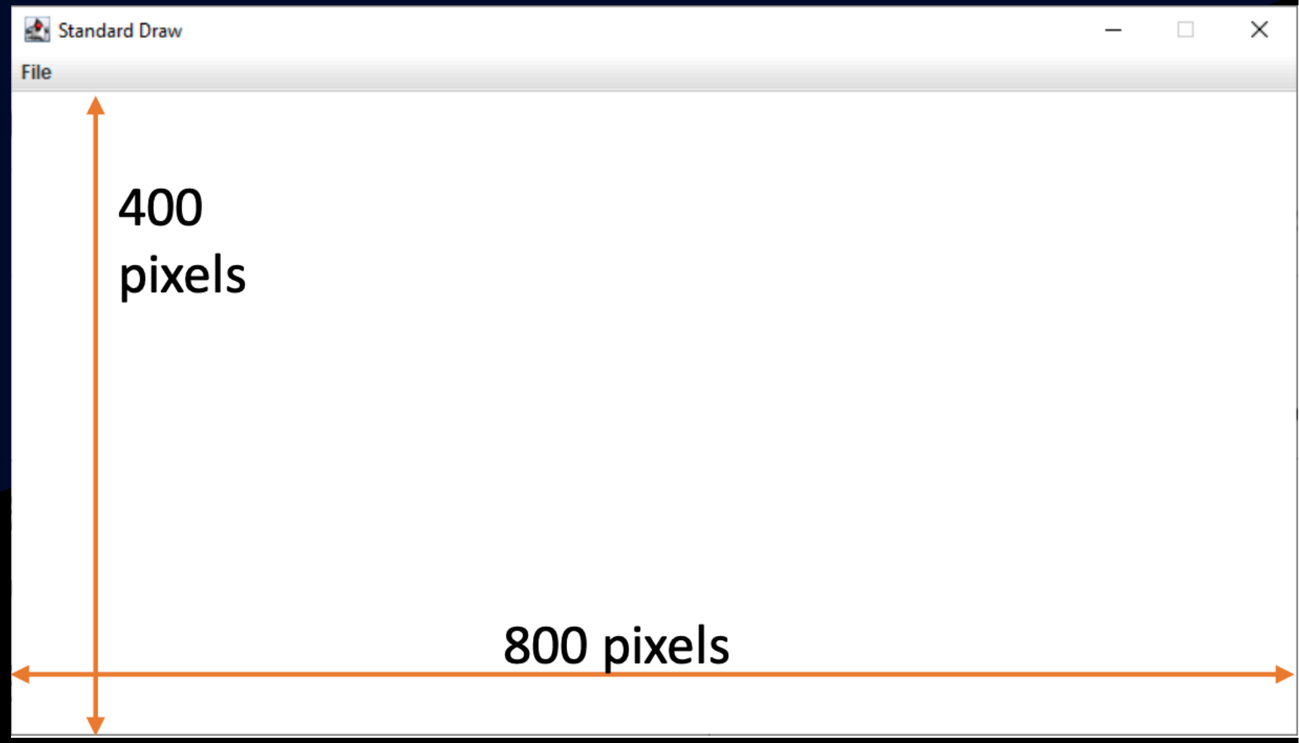
The PennDraw Model

The Canvas

The *canvas* refers to the window of space on which PennDraw can do its drawing.

It has a width and a height,
both defined in pixels.

- We usually express the size of a canvas like "*width* by *height*"
- Width is the *x dimension*
- Height is the *y dimension*

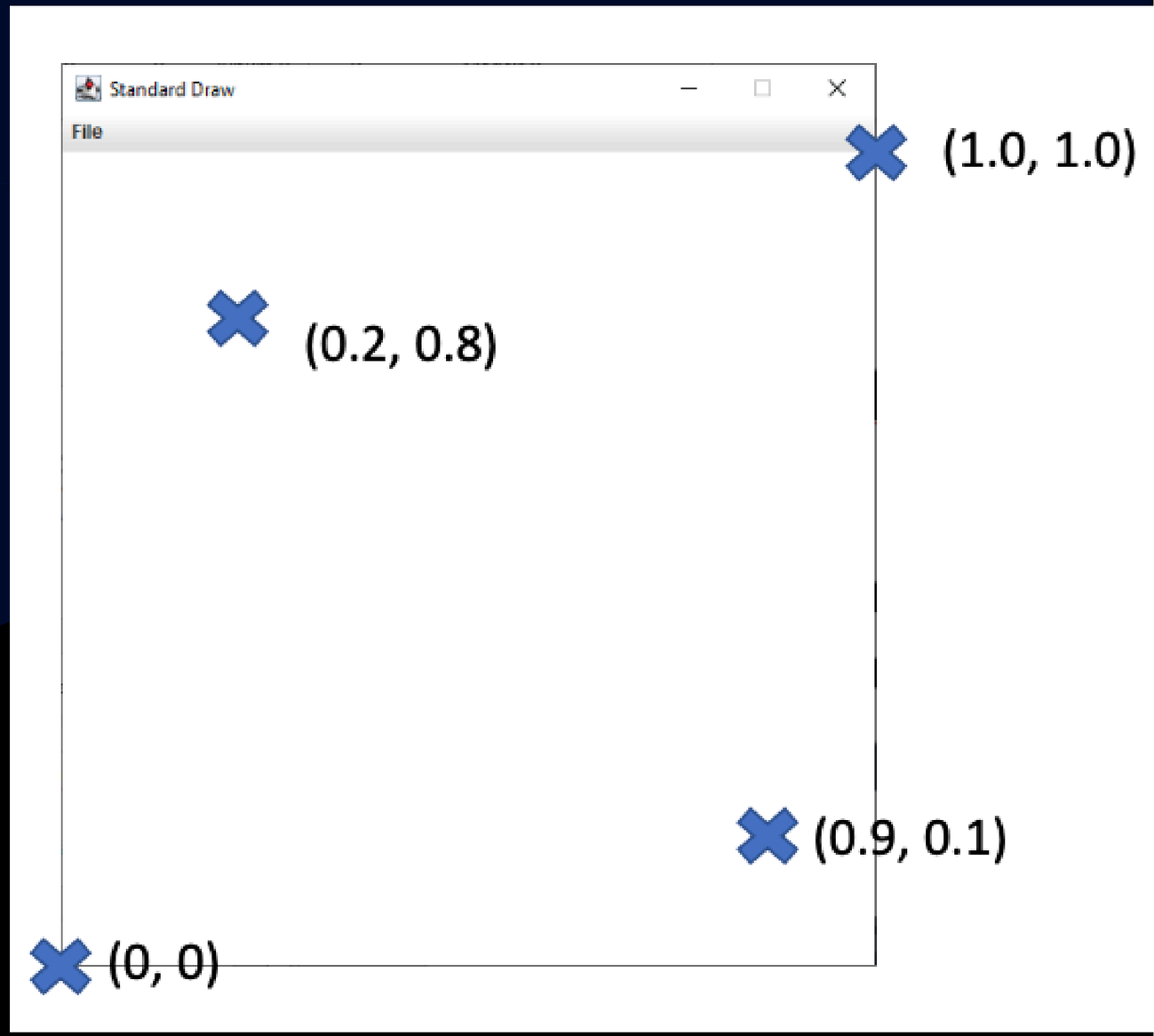


The Coordinate System

Canvas positions are accessed using coordinates.

By default, coordinates range from 0 to 1 in both the x dimension and the y dimension.

- The coordinate (0,0) refers to the bottom left position of the canvas.
- Coordinate (1,1) is found at the top right of the canvas.



The Pen

PennDraw works in a model where the programmer (you!)
gives a series of instructions, one by one, to a computer

Some instructions are responsible for changing how shapes will be drawn

- “changing the settings of the pen”, e.g. radius and color

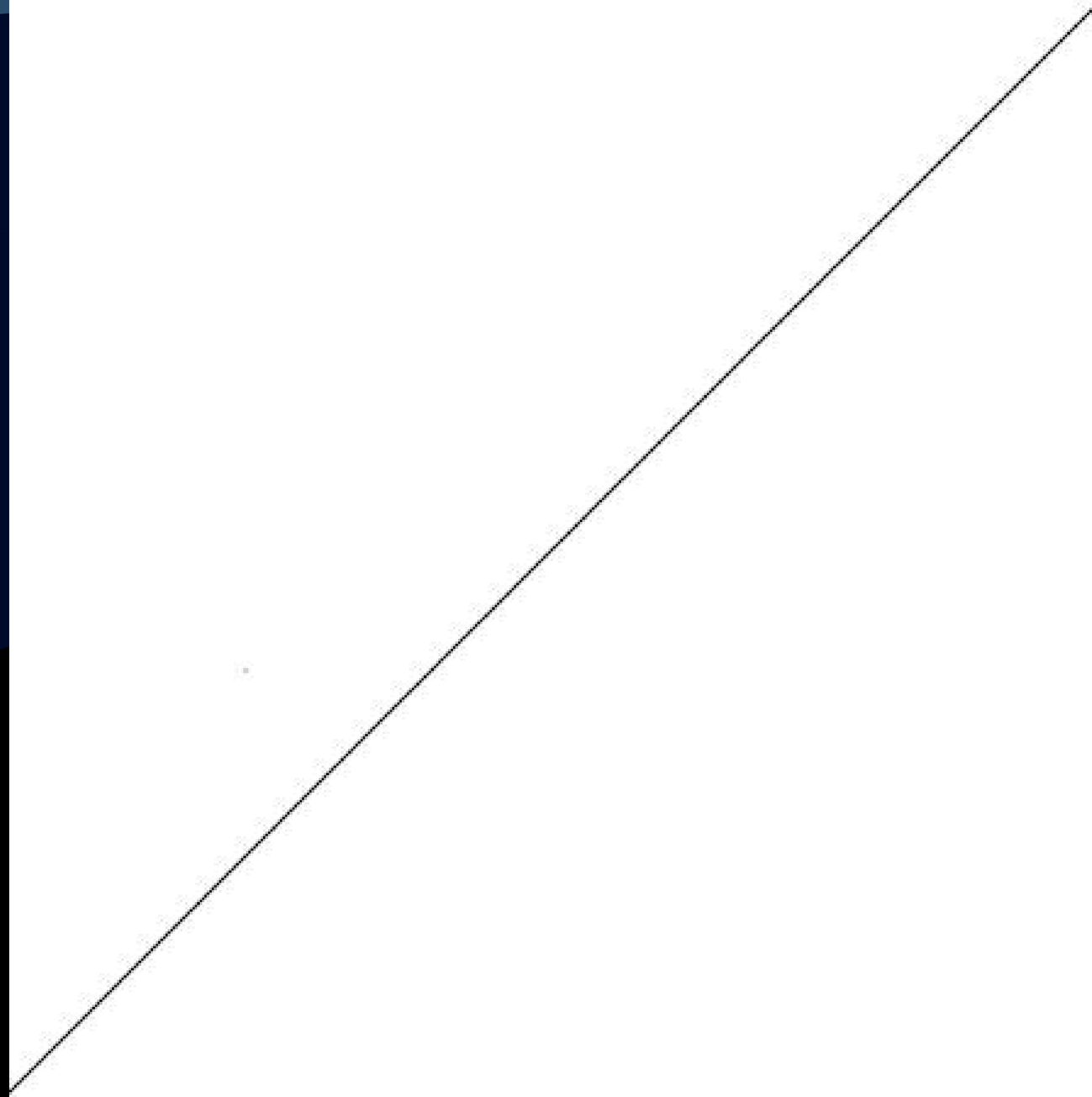
The instructions change the pen settings until the next time the settings are explicitly modified.

Changing Pen Settings

Radius

Whenever we ask PennDraw to draw e.g. a point or line on the screen, these marks will appear with a certain thickness determined by the current setting for the radius of the pen.

Pictured: a point and a line drawn with a default radius setting of 0.002

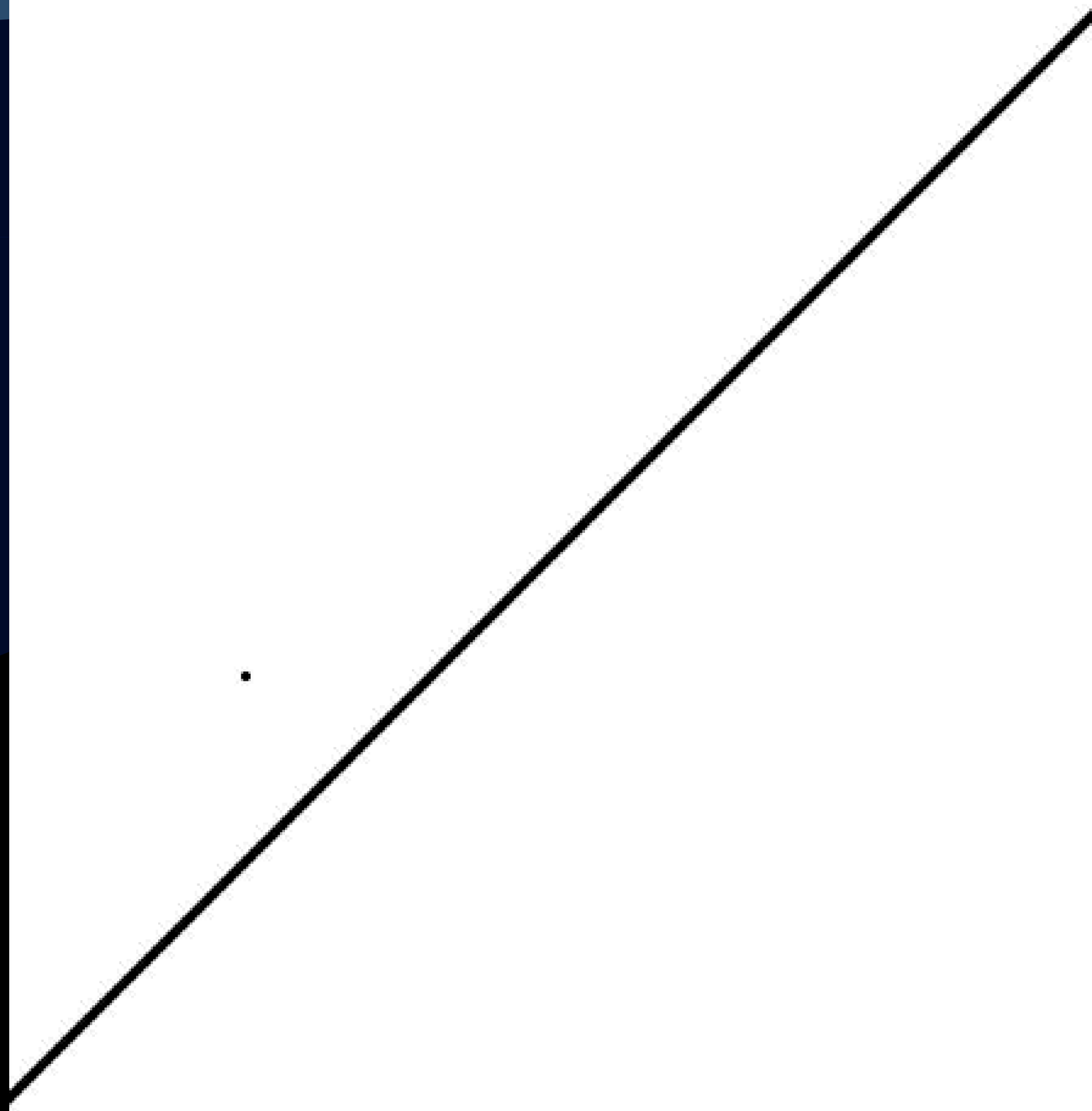


Radius

On right is the same drawing with the pen radius set to 0.008 four times the default setting.

To change the pen radius:

```
pd.set_pen_radius(0.008)
```



Color

Two ways to set the pen color.

First: referring to some of them by name

```
pd.set_pen_color(pd.BLUE);
pd.set_pen_color(pd.MAGENTA);
```

Color

Two ways to set the pen color

Second: specifying the red, green, and blue values of the color as integers from 0-255 each

```
# "pure red"
pd.set_pen_color(255, 0, 0)

# "twilight lavender"
pd.set_pen_color(138, 73, 107)
```

"pure red" and "twilight lavender"

Running PennDraw Programs

Importing PennDraw

- PennDraw is the name of a set of tools that are not built into Python directly.
  - Must tell Python where to go get the PennDraw tools and what they will be called in your program
  - At the start of your PennDraw programs:

```
import penndraw as pd
```

Running PennDraw

- Nothing will appear in your output unless you add `pd.run()` to the end of your program.
  - On its own, this statement doesn't draw or do anything
  - If you don't see anything being drawn, double check that you have `pd.run()` at the very end.

Stopping PennDraw Programs

- For `hello_world.py`, we had a simple loop: *edit, run, repeat*
  - The program was finished running once it ran out of instructions to execute
- For programs that use PennDraw, the program will continue to run so that you can see the drawing you made!
- Before re-running, you need to stop the program execution one of two ways:
  i. Close the drawing window
  ii. Press Control-C on your keyboard in the terminal

Demo: Running (and Stopping) a Program

Let's type, compile, and execute the following program:

```
import penndraw as pd
pd.filled_square(0.5, 0.5, 0.25)
pd.run()
```

You should see a black square in the center of the screen!

Demo: Running (and Stopping) a Program

Let's make a change to the program:

```
import penndraw as pd
pd.set_pen_color(pd.GREEN)
pd.filled_square(0.5, 0.5, 0.25)
pd.run()
```

Then, we need to **stop the execution** of the program, recompile, and run the program again.

You should see a **green** square in the center of the screen!

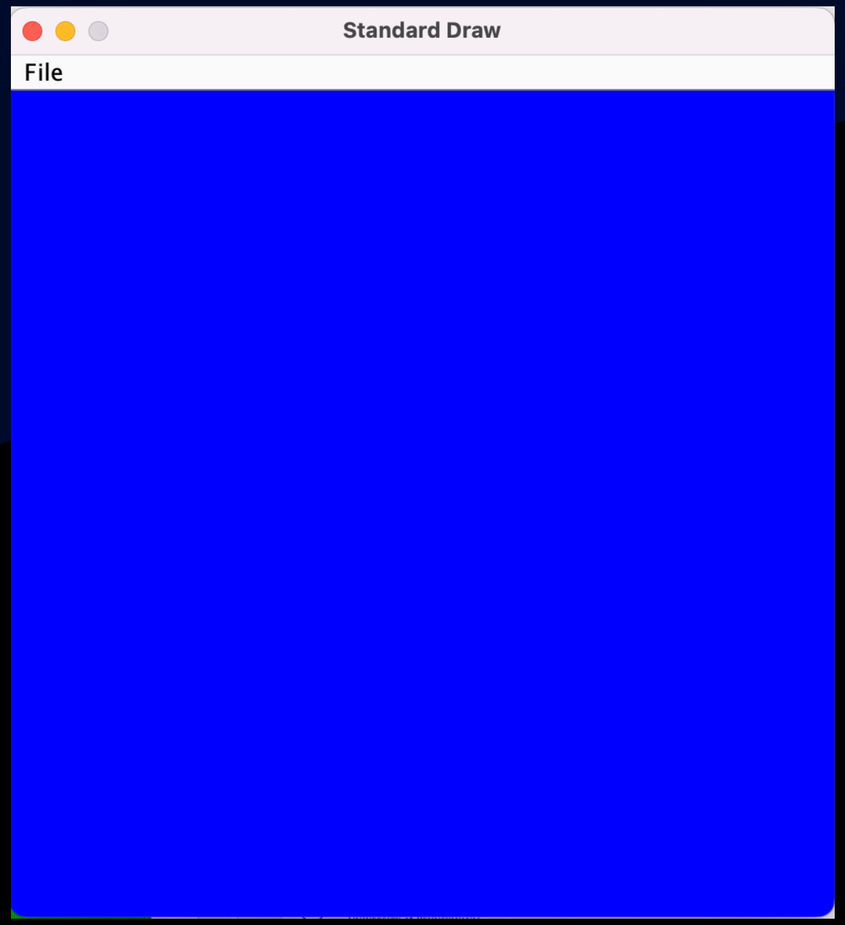
my_house.py

Inventory

- `pd.set_canvas_size(width, height)`
  - Sets the canvas to a certain `width` & `height` in pixels
- `pd.clear(color)`
  - Clears the screen and colors the background in the provided `color`.
  - `color` can be provided by name or by passing in three integers
    - e.g. `pd.BLUE` or `(0, 0, 255)`
- `pd.set_pen_color(color)`
  - Sets the `color` that future shapes will be drawn in
  - Ditto bullet two from `pd.clear()`

```python
import penndraw as pd

pd.set_canvas_size(500, 500)
pd.run()
```

```python
import penndraw as pd

pd.set_canvas_size(500, 500)

# draw a blue background
pd.clear(pd.BLUE)
pd.run()
```

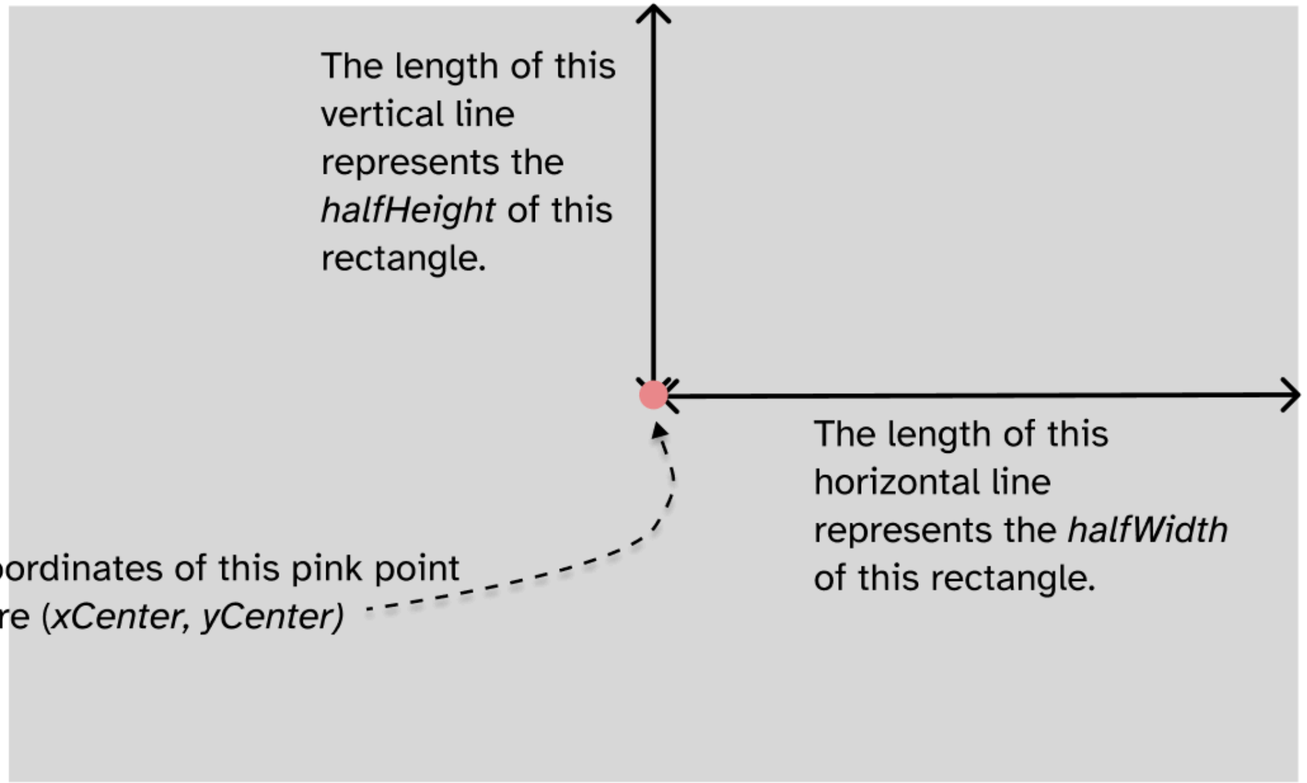


`pd.filled_rectangle(...)`

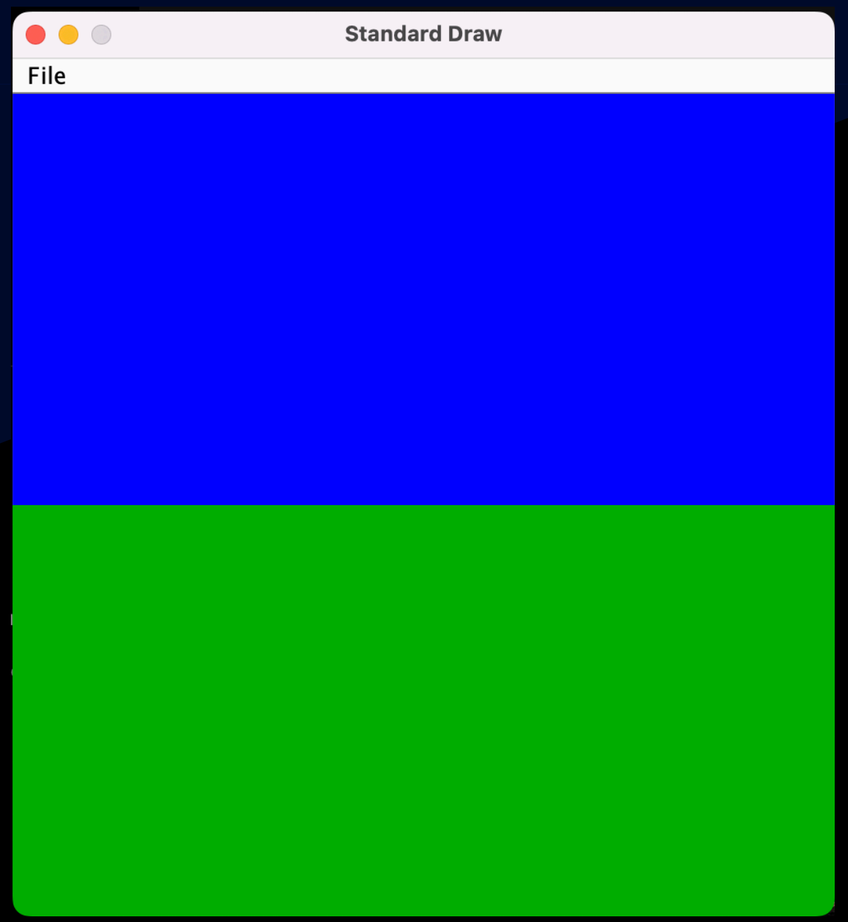
This function takes four arguments:

- `x_center`: the x coordinate of the center of the rectangle
- `y_center`: the y coordinate of the center of the rectangle
- `half_width`: the horizontal distance between the side of the rectangle and its center
- `half_height`: the vertical distance between the top of the rectangle and its center

This larger rectangle represents the canvas.



```python
import penndraw as pd

pd.set_canvas_size(500, 500)

# draw a blue background
pd.clear(pd.BLUE)

# draw a green field
pd.set_pen_color(0, 170, 0)
pd.filled_rectangle(0.5, 0.25, 0.5, 0.25)
pd.run()
```



`pd.filled_polygon(...)`

This function takes n pairs of (x, y) coordinates to draw an n-gon with vertices at the specified coordinates.

- For a triangle (3-gon), provide three coordinate pairs (six numbers)
- For a hexagon (6-gon), provide six coordinate pairs (twelve numbers)

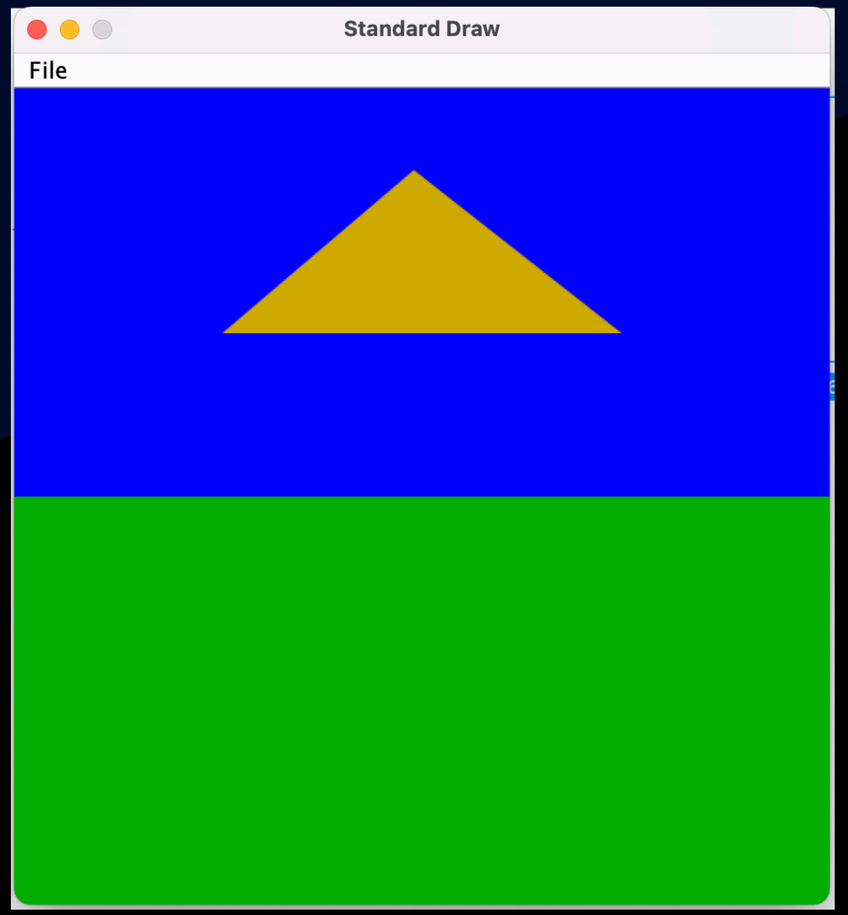
Drawing a Roof

`pd.filled_polygon(0.255, 0.70, 0.745, 0.70, 0.49, 0.90);`

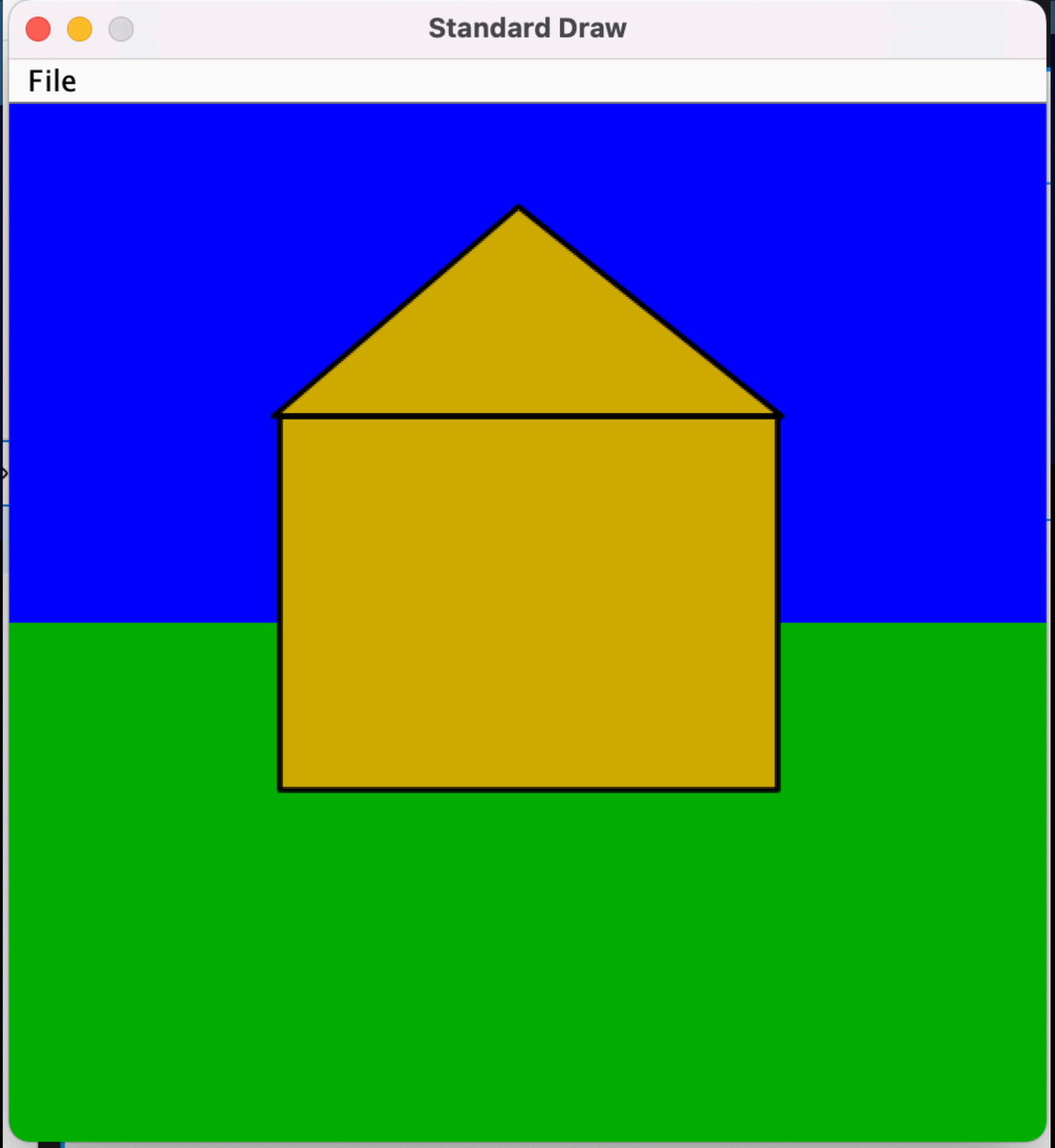
draws a polygon with vertices at coordinates:

- `(0.255, 0.70)`,
- `(0.745, 0.70)`,
- and `(0.49, 0.90)`

```python
import penndraw as pd

pd.set_canvas_size(500, 500)

# draw a blue background
pd.clear(pd.BLUE)

# draw a green field
pd.set_pen_color(0, 170, 0)
pd.filled_rectangle(0.5, 0.25, 0.5, 0.25)

# change the pen color to a shade of yellow
pd.set_pen_color(200, 170, 0)

# draw a filled triangle (roof)
pd.filled_polygon(0.255, 0.70, 0.745, 0.70, 0.49, 0.90)
pd.run()
```



```
import penndraw as pd

pd.set_canvas_size(500, 500)

# draw a blue background
pd.clear(pd.BLUE)

# draw a green field
pd.set_pen_color(0, 170, 0)
pd.filled_rectangle(0.5, 0.25, 0.5, 0.25)

# change the pen color to a shade of yellow
pd.set_pen_color(200, 170, 0)

# draw a filled triangle (roof)
pd.filled_polygon(0.255, 0.70, 0.745, 0.70, 0.49, 0.90)

# draw the house
pd.filled_rectangle(0.5, 0.52, 0.24, 0.18)
pd.run()
```


Standard Draw
File

```python
import penndraw as pd

pd.set_canvas_size(500, 500)

# draw a blue background
pd.clear(pd.BLUE)

# draw a green field
pd.set_pen_color(0, 170, 0)
pd.filled_rectangle(0.5, 0.25, 0.5, 0.25)

# change the pen color to a shade of yellow
pd.set_pen_color(200, 170, 0)

# draw a filled triangle (roof)
pd.filled_polygon(0.255, 0.70, 0.745, 0.70, 0.49, 0.90)

# draw the house
pd.filled_rectangle(0.5, 0.52, 0.24, 0.18)

pd.set_pen_radius(0.005)  # thicken the pen for outline drawing
pd.set_pen_color(pd.BLACK)  # make the pen black

# draw the roof, house, and door outlines with non-filled rectangles
pd.polygon(0.255, 0.70, 0.745, 0.70, 0.49, 0.90)  # roof
pd.rectangle(0.5, 0.52, 0.24, 0.18)  # house
pd.rectangle(0.596, 0.44, 0.08, 0.1)  # door

pd.point(0.54, 0.44)  # draw a doorknob
pd.run()
```