Nicholas Bowman, Sonja Johnson-Yu, Kylie Jue **Section #5**
CS106AP **July 24, 2019**

Section #5: Solutions

1. Grid Drawing

```
#!/usr/bin/env python3

import sys
import tkinter


def make_canvas(width, height):
    (provided)
    top = tkinter.Tk()
    top.minsize(width=width + 10, height=height + 10)

    canvas = tkinter.Canvas(top, width=width, height=height)
    canvas.pack()
    canvas.xview_scroll(6, 'units')  # hack so (0, 0) works correctly
    canvas.yview_scroll(6, 'units')

    # draw blue boundaries - sides
    canvas.create_line(0, 0, 0, height - 1, fill='blue')
    canvas.create_line(width - 1, 0, width - 1, height - 1, fill='blue')
    # top to bottom
    canvas.create_line(0, 0, width - 1, 0, fill='blue')
    canvas.create_line(0, height - 1, width - 1, height - 1, fill='blue')
    return canvas


def draw_grid(width, height, n):
    canvas = make_canvas(width, height)

    # Vertical lines at 1/n, 2/n, .. n-1/n
    for i in range(1, n):
        x = int(width * i/n)
        canvas.create_line(x, 0, x, height - 1, fill='red')

    # b. horizontal lines + text
    for i in range(1, n):
       y = int(height * i/n)
       canvas.create_line(0, y, width - 1, y, fill='red')
       canvas.create_text(3, y, text=str(y), anchor=tkinter.SW)


# Default canvas size
```

```
WIDTH = 500
HEIGHT = 300

def main():
    args = sys.argv[1:]

    # args:
    # n               # draws 500x300 with this n
    if len(args) == 1:
        draw_grid(WIDTH, HEIGHT, int(args[0]))
        tkinter.mainloop()

    # args:
    # width height n  # draws custom size grid with n
    if len(args) == 3:
        draw_grid(int(args[0]), int(args[1]), int(args[2]))
        tkinter.mainloop()


if __name__ == '__main__':
    main()
```

2. Random Circles

```
from campy.graphics.gwindow import GWindow
from campy.graphics.gobjects import GOval
import random

WIDTH = 800
HEIGHT = 600
RADIUS_MIN = 30
RADIUS_MAX = 100
N_CIRCLES_MAX = 10
COLORS = ['RED', 'ORANGE', 'YELLOW', 'GREEN', 'BLUE']

def make_one_circle(window):
    radius = random.randint(RADIUS_MIN, RADIUS_MAX)
    x0 = random.randint(0, WIDTH  - 2 * radius)
    y0 = random.randint(0, HEIGHT - 2 * radius)
    circle = GOval(width=2 * radius, height=2 * radius, x=x0, y=y0)
    color = random.choice(COLORS)
    circle.filled = True
    circle.fill_color = color
    window.add(circle)

def make_all_circles(window):
    n_circles = random.randrange(0, N_CIRCLES_MAX) + 1
```

```
    for i in range(n_circles):
        make_one_circle(window)


def main():
    window = GWindow(width=WIDTH, height=HEIGHT, title="Random Circles")
    make_all_circles(window)


if __name__ == "__main__":
    main()
```

3. Robot Face

There are many ways to decompose this problem. This is just one of them!

```
from campy.graphics.gwindow import GWindow
from campy.graphics.gobjects import GOval, GRect

WIDTH = 600
HEIGHT = 400

HEAD_WIDTH = 150
HEAD_HEIGHT = 250
EYE_RADIUS = 20
MOUTH_WIDTH = 100
MOUTH_HEIGHT = 30


def main():
      window = GWindow(width=WIDTH, height=HEIGHT, title="Robot Head")
      midpoint_x = window.width / 2
      midpoint_y = window.height / 2
      add_head(window, midpoint_x - HEAD_WIDTH / 2, midpoint_y -
HEAD_HEIGHT / 2)
      add_eye(window, midpoint_x - HEAD_WIDTH / 4, midpoint_y -
HEAD_HEIGHT / 4)
      add_eye(window, midpoint_x + HEAD_WIDTH / 4, midpoint_y -
HEAD_HEIGHT / 4)
      add_mouth(window, midpoint_x - MOUTH_WIDTH / 2, midpoint_y +
HEAD_HEIGHT / 4)

def add_head(window, x, y):
      draw_rectangle(window, x, y, HEAD_WIDTH, HEAD_HEIGHT, "gray")

def add_eye(window, center_x, center_y):
      draw_circle(window, center_x, center_y, EYE_RADIUS, "yellow")

def add_mouth(window, x, y):
```

```
      draw_rectangle(window, x, y, MOUTH_WIDTH, MOUTH_HEIGHT, "white")

def draw_rectangle(window, x, y, width, height, color):
      rect = GRect(width=width, height=height, x=x, y=y)
      rect.filled = True
      rect.fill_color = color
      window.add(rect)

def draw_circle(window, center_x, center_y, radius, color):
      x = center_x - radius
      y = center_y - radius
      circle = GOval(width=2*radius, height=2*radius, x=x, y=y)
      circle.filled = True
      circle.fill_color = color
      window.add(circle)


if __name__ == '__main__':
      main()
```

4. Sunset

```
from campy.graphics.gwindow import GWindow
from campy.graphics.gobjects import GOval, GRect
from campy.gui.events.timer import pause


WIDTH = 600
HEIGHT = 400

SUN_DIAMETER = 75
HORIZON_HEIGHT = 100
SUNSET_VELOCITY = 1.0
PAUSE_TIME = 40 # MS pause b/w frames


def make_sun(window):
    """
    Creates and returns an oval representing the sun.
    """
    # Center the GOval in the window.
     sun = GOval(width=SUN_DIAMETER, height=SUN_DIAMETER, x=(window.width
- SUN_DIAMETER) / 2, y=(window.height - SUN_DIAMETER) / 2.0)
    sun.filled = True
    sun.fill_color = "yellow"
    return sun
```

```
def make_horizon(window):
    """
    Creates and returns a rectangle representing the horizon.
    """
    # horizon should horizontally fill window and should have
    # height of HORIZON_HEIGHT. It will be aligned to the bottom
    # of the window.
     result  =    GRect(width=window.width,  height=HORIZON_HEIGHT,  x=0,
y=(window.height - HORIZON_HEIGHT))
    result.filled = True
    result.fill_color = "green"
    return result


def perform_sunset(window, sun):
    """
    Simulates a sunset
    """
    # Keep moving the sun downward until it has set.
    while not has_sun_set(window, sun):
        sun.move(0, SUNSET_VELOCITY)
        pause(PAUSE_TIME)


def has_sun_set(window, sun):
    """
    Given the sun, determine whether or not it has set.
    """
    # The sun has set as soon as its top is below the horizon.
    return sun.y > window.height - HORIZON_HEIGHT


def main():
    window = GWindow(height=HEIGHT, width=WIDTH, title="Sunset")
    sun = make_sun(window)
    horizon = make_horizon(window)
    window.add(sun)  # add sun then horizon so sun can set behind it
    window.add(horizon)
    perform_sunset(window, sun)


if __name__ == '__main__':
    main()
```

5. Bonus: Pynary Bomb

The correct command to defuse the bomb is:

```
$ python3 pynary_bomb.py 3 4 9
```

If you didn't work through this yourself, try tracing through the problem now and verifying that this sequence of numbers would in fact defuse the bomb!