Programming resources and Project example

Fermilab - TARGET 2018
Week 4

Notes about the projects

OK to use existing libraries

OK to use existing programs

1. You have to report what you use
2. You have to understand what you use
3. You have to add something significant

Some links to get help

http://anandology.com/python-practice-book/object_oriented_programming.html There are some mistakes but has good ideas

http://python-textbok.readthedocs.io/en/1.0/index.html

http://www.python-course.eu/index.php (http://www.python-course.eu/deep_copy.php)

https://www.gitbook.com/book/swaroopch/byte-of-python/details

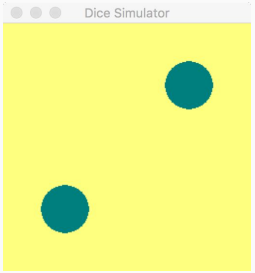
https://www.tutorialspoint.com/python/index.htm

Tutorial and Reference - https://docs.python.org/3/

help() - within the interpreter

An example program: Dice simulation

1. Project goal: A graphical dice simulator, possibly a rolling dice
2. State of the art: What is around
3. Adjust the aim given the time constraints (only 2D images)
4. Code and test



State of the art

Text simulators, very simple:

http://www.pythonforbeginners.com/code-snippets-source-code/game-rolling-the-dice
https://www.youtube.com/watch?v=b6U3rw-cH6A
https://www.youtube.com/watch?v=GhygOO_CSRY

Ideas with graphic:

https://github.com/alekhrycaiko/graphic_dice_simulator_pygame/blob/master/main.py
http://pygame.org/project-Pygame+Dice-1287-.html
http://python3.codes/a-graphical-dice-simulator/

A complete program in another language:

http://a.teall.info/dice/
https://github.com/emanchado/3d-die-roller
http://www.teall.info/2014/01/online-3d-dice-roller.html
http://www.virtualdiceroll.com/
http://dice.virtuworld.net/

Code 1/3 (Initialization)

```
#!/usr/local/bin/pyhon3
'''
    Using DiceSimulator.py for the rendering
    http://python3.codes/a-graphical-dice-simulator/

    And pygame/examples/scaletest.py for the conrol loop
    https://www.pygame.org/

    Author: Marco Mambelli

'''

import pygame
import random
import time


# Constants
DSIZE = 256                     # Size of window/dice
spsz = DSIZE//10                # size of spots
m = int(DSIZE/2)                # mid-point of dice (or die?)
l=t=int(DSIZE/4)                # location of left and top spots
r=b=DSIZE-l                     # location of right and bottom spots
rolling = 12                    # times that dice rolls before stopping
DEFAULT_DICECOL = (255,255,127)         # die colour
DEFAULT_SPOCOL = (0,127,127)            # spot colour

DEBUG = True
```

Code 2/3 (Dice class)

```
class Dice:
    IMPOSSIBLE_TRANSITIONS = [ 0, 6, 5, 4, 3, 2, 1 ] # 0 added for completion n+new_n==7
    def __init__(self, display, n=1, diecol=DEFAULT_DICECOL, spotcol=DEFAULT_SPOCOL):
        self.d = display
        self.diecol = diecol
        self.spotcol = spotcol
        self.n = n

    def paint(self, n):
        """ Change the value of dice and paint it"""
        self.n = n
        self.d.fill(self.diecol)                    # clear previous spots
        if n % 2 == 1:
            pygame.draw.circle(self.d,self.spotcol,(m,m),spsz)  # middle spot
        if n > 1:   # n==2 or n==3 or n==4 or n==5 or n==6:
            pygame.draw.circle(self.d,self.spotcol,(l,b),spsz)  # left bottom
            pygame.draw.circle(self.d,self.spotcol,(r,t),spsz)  # right top
        if n > 3:   # n==4 or n==5 or n==6:
            pygame.draw.circle(self.d,self.spotcol,(l,t),spsz)  # left top
            pygame.draw.circle(self.d,self.spotcol,(r,b),spsz)  # right bottom
        if n == 6:
            pygame.draw.circle(self.d,self.spotcol,(m,b),spsz)  # middle bottom
            pygame.draw.circle(self.d,self.spotcol,(m,t),spsz)  # middle top
        pygame.display.flip()
        # Need to peek for events, otherwise the display is not updated/flipped
        pygame.event.peek()

    # … continue on next column
```

```
    # … continue here, class Dice

    def roll(self, roll_max=20):
        """Roll the dice (multiple changes)"""
        if DEBUG:
            print("Rolling:")
        for i in range(int(random.random() * roll_max)):
            n = random.randint(1,6)
            if DEBUG:
                print(" - roll %s: %s" % (i, n))
            if self.n + n == 7:  # impossible transition
                continue
            self.paint(n)
            time.sleep(0.2)
```

Code 3/3 (main loop - 3 columns)

```
def main():
    """show dice - main loop
    """
    # pygame.init()
    # initialize display
    pygame.display.init()
    d = pygame.display.set_mode((DSIZE, DSIZE))
    d.fill(DEFAULT_DICECOL)
    pygame.display.set_caption("Dice Simulator")
    # Add text
    pygame.font.init()
    myfont = pygame.font.SysFont("monospace", 15)
    label = myfont.render("Press UP to roll!", 1, DEFAULT_SPOCOL)
    label2 = myfont.render("and ESC o Quit", 1, DEFAULT_SPOCOL)
    d.blit(label, (20, 20))
    d.blit(label2, (20, 40))
    # turn off the mouse pointer
    # pygame.mouse.set_visible(0)
    # Get the dice
    dice = Dice(d)
    # main loop
    bRunning = True
    bUp = False
    bDown = False
    bLeft = False
    bRight = False
    mouseDown = False
    dice_face = None


# … continue on next column
```

```
    # … continue here 1
    while(bRunning):
        pygame.display.flip()
        for event in pygame.event.get():
            new_event = True
            if DEBUG:
                print ("Event loop: %s (%s)" % (event.type, event))
                if event.type == pygame.KEYDOWN or event.type == pygame.KEYUP:
                    print(" key event: %s" % event.key)
            if event.type == pygame.QUIT or (event.type == pygame.KEYDOWN and event.key == pygame.K_ESCAPE):
                bRunning = False
            if event.type == pygame.KEYDOWN:
                dice_face = None
                # Check for numbers 1-6
                if event.unicode == '1':   dice_face = 1
                if event.unicode == '2':   dice_face = 2
                if event.unicode == '3':   dice_face = 3
                if event.unicode == '4':   dice_face = 4
                if event.unicode == '5':   dice_face = 5
                if event.unicode == '6':   dice_face = 6
                # Check for cursors
                if event.key == pygame.K_UP:    bUp = True
                if event.key == pygame.K_DOWN:  bDown = True
                if event.key == pygame.K_LEFT:  bLeft = True
                if event.key == pygame.K_RIGHT: bRight = True
            if event.type == pygame.KEYUP:
                if event.key == pygame.K_UP:    bUp = False
                if event.key == pygame.K_DOWN:  bDown = False
                if event.key == pygame.K_LEFT:  bLeft = False
                if event.key == pygame.K_RIGHT: bRight = False
            if event.type == pygame.MOUSEBUTTONDOWN:
                # pos, button
                dice_face = None
                mouseDown = True
            if event.type == pygame.MOUSEBUTTONUP:
                # pos, button
                mouseDown = False

        # … continue on next column
```

```
    # … continue here 2

        if DEBUG and new_event:
            new_event = False
            print ("After loop: %s, %s, %s" %
(dice_face, bUp, mouseDown))

        if dice_face:
            dice.paint(dice_face)
            dice_face = None
        elif bUp or mouseDown:
            dice.roll()

    # Invoke main loop if script
    if __name__ == "__main__":
        main()
```

Some ideas for projects

http://www.math.stonybrook.edu/~scott/papers/MSTP/crypto/crypto.html

https://target2018.onlineth.com/

Use the data provided by Andrew (NOVA detector and file transfer)

Online code highlighting: https://tohtml.com/python/