Kids Code

Setting Up PyGame

1. First we need to install the pygame library. Open up a **terminal** and type the following:

```
pip install pygame
```

2. Next, we have to create a new python file. Use the **cd** command in the terminal to navigate to the Desktop, then create a file with the **touch** command. Double click the file icon that should appear on your desktop.

```
cd ~/Desktop/
touch game.py
```

3. Enter the following text into the file and save it.

```
import pygame

pygame.init()
```

4. Lastly, we have to run the code. Using the terminal while still in the Desktop directory, run this command:

```
python game.py
```

Rotate Text!

Here's an example program showing you how to **rotate text** with pygame.

```
import pygame

# Initialize the game engine
pygame.init()

# Define some colors
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
BLUE = (0, 0, 255)
GREEN = (0, 255, 0)
RED = (255, 0, 0)

PI = 3.141592653

# Set the height and width of the screen
size = (400, 500)
screen = pygame.display.set_mode(size)

pygame.display.set_caption("Rotate Text")

# Loop until the user clicks the close button.
```

```
done = False
clock = pygame.time.Clock()

text_rotate_degrees = 0

# Loop as long as done == False
while not done:
    for event in pygame.event.get():  # User did something
        if event.type == pygame.QUIT:  # If user clicked close
            done = True  # Flag that we are done so we exit this loop

    # All drawing code happens after the for loop and but
    # inside the main while not done loop.

    # Clear the screen and set the screen background
    screen.fill(WHITE)

    # Draw some borders
    pygame.draw.line(screen, BLACK, [100,50], [200, 50])
    pygame.draw.line(screen, BLACK, [100,50], [100, 150])

    # Select the font to use, size, bold, italics
    font = pygame.font.SysFont('Calibri', 25, True, False)

    # Sideways text
    text = font.render("Sideways text", True, BLACK)
    text = pygame.transform.rotate(text, 90)
    screen.blit(text, [0, 0])

    # Sideways text
    text = font.render("Upside down text", True, BLACK)
    text = pygame.transform.rotate(text, 180)
    screen.blit(text, [30, 0])

    # Flipped text
    text = font.render("Flipped text", True, BLACK)
    text = pygame.transform.flip(text, False, True)
    screen.blit(text, [30, 20])

    # Animated rotation
    text = font.render("Rotating text", True, BLACK)
    text = pygame.transform.rotate(text, text_rotate_degrees)
    text_rotate_degrees += 1
    screen.blit(text, [100, 50])

    # Go ahead and update the screen with what we've drawn.
    # This MUST happen after all the other drawing commands.
    pygame.display.flip()

    # This limits the while loop to a max of 60 times per second.
    # Leave this out and we will use all CPU we can.
    clock.tick(60)

# Be IDLE friendly
pygame.quit()
```

Bouncing Balls!

He's some source code to make a ball slowly bounce around the screen. Can you speed up the ball? Can you make it bounce faster? Can you add another ball?

```
import pygame
import random

# Define some colors
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)

SCREEN_WIDTH = 700
SCREEN_HEIGHT = 500
BALL_SIZE = 25


class Ball:
    """
    Class to keep track of a ball's location and vector.
    """
    def __init__(self):
        self.x = 0
        self.y = 0
        self.change_x = 0
        self.change_y = 0


def make_ball():
    """
    Function to make a new, random ball.
    """
    ball = Ball()
    # Starting position of the ball.
    # Take into account the ball size so we don't spawn on the edge.
    ball.x = random.randrange(BALL_SIZE, SCREEN_WIDTH - BALL_SIZE)
    ball.y = random.randrange(BALL_SIZE, SCREEN_HEIGHT - BALL_SIZE)

    # Speed and direction of rectangle
    ball.change_x = random.randrange(-2, 3)
    ball.change_y = random.randrange(-2, 3)

    return ball


def main():
    """
    This is our main program.
    """
    pygame.init()

    # Set the height and width of the screen
    size = [SCREEN_WIDTH, SCREEN_HEIGHT]
    screen = pygame.display.set_mode(size)

    pygame.display.set_caption("Bouncing Balls")

    # Loop until the user clicks the close button.
    done = False

    # Used to manage how fast the screen updates
    clock = pygame.time.Clock()

    ball_list = []

    ball = make_ball()
    ball_list.append(ball)

    # -------- Main Program Loop -----------
```

```
    while not done:
        # --- Event Processing
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                done = True
            elif event.type == pygame.KEYDOWN:
                # Space bar! Spawn a new ball.
                if event.key == pygame.K_SPACE:
                    ball = make_ball()
                    ball_list.append(ball)

        # --- Logic
        for ball in ball_list:
            # Move the ball's center
            ball.x += ball.change_x
            ball.y += ball.change_y

            # Bounce the ball if needed
            if ball.y > SCREEN_HEIGHT - BALL_SIZE or ball.y < BALL_SIZE:
                ball.change_y *= -1
            if ball.x > SCREEN_WIDTH - BALL_SIZE or ball.x < BALL_SIZE:
                ball.change_x *= -1

        # --- Drawing
        # Set the screen background
        screen.fill(BLACK)

        # Draw the balls
        for ball in ball_list:
            pygame.draw.circle(screen, WHITE, [ball.x, ball.y], BALL_SIZE)

        # --- Wrap-up
        # Limit to 60 frames per second
        clock.tick(60)

        # Go ahead and update the screen with what we've drawn.
        pygame.display.flip()

    # Close everything down
    pygame.quit()

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

Snake!

Here's the source code for a basic game of snake, but it's not quite finished. Think you can finish it?

```
import pygame

# --- Globals ---
# Colors
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)

# Set the width and height of each snake segment
segment_width = 15
segment_height = 15
# Margin between each segment
segment_margin = 3
```

```
# Set initial speed
x_change = segment_width + segment_margin
y_change = 0


class Segment(pygame.sprite.Sprite):
    """ Class to represent one segment of the snake. """
    # -- Methods
    # Constructor function
    def __init__(self, x, y):
        # Call the parent's constructor
        super().__init__()

        # Set height, width
        self.image = pygame.Surface([segment_width, segment_height])
        self.image.fill(WHITE)

        # Make our top-left corner the passed-in location.
        self.rect = self.image.get_rect()
        self.rect.x = x
        self.rect.y = y

# Call this function so the Pygame library can initialize itself
pygame.init()

# Create an 800x600 sized screen
screen = pygame.display.set_mode([800, 600])

# Set the title of the window
pygame.display.set_caption('Snake Example')

allspriteslist = pygame.sprite.Group()

# Create an initial snake
snake_segments = []
for i in range(15):
    x = 250 - (segment_width + segment_margin) * i
    y = 30
    segment = Segment(x, y)
    snake_segments.append(segment)
    allspriteslist.add(segment)


clock = pygame.time.Clock()
done = False

while not done:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True

        # Set the speed based on the key pressed
        # We want the speed to be enough that we move a full
        # segment, plus the margin.
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                x_change = (segment_width + segment_margin) * -1
                y_change = 0
            if event.key == pygame.K_RIGHT:
                x_change = (segment_width + segment_margin)
                y_change = 0
            if event.key == pygame.K_UP:
                x_change = 0
```

```
                y_change = (segment_height + segment_margin) * -1
            if event.key == pygame.K_DOWN:
                x_change = 0
                y_change = (segment_height + segment_margin)

    # Get rid of last segment of the snake
    # .pop() command removes last item in list
    old_segment = snake_segments.pop()
    allspriteslist.remove(old_segment)

    # Figure out where new segment will be
    x = snake_segments[0].rect.x + x_change
    y = snake_segments[0].rect.y + y_change
    segment = Segment(x, y)

    # Insert new segment into the list
    snake_segments.insert(0, segment)
    allspriteslist.add(segment)

    # -- Draw everything
    # Clear screen
    screen.fill(BLACK)

    allspriteslist.draw(screen)

    # Flip screen
    pygame.display.flip()

    # Pause
    clock.tick(5)

pygame.quit()
```

Author: Alex Recker
Created: 2019-05-16 Thu 15:44
Validate