discover the basics of the Pygame library

Part 2

What’s the goal of our meeting?

Let’s discuss Part 1

What we have learned in Part 1:

- Create a display surface
- Draw figures on the display surface
- Set the colors
- Discrete keyboard movements and continuous movements
- Restrict our display surface
- Blitting images
- Add background music

Lets download Part 1 from GitHub
Please scan QR or use a link:

https://github.com/halloweex/learn-pygame

Use command line :

```
→  ~ git clone https://github.com/halloweex/learn-pygame.git
```

Plan for today

What we have to learn:

- How to define fonts
- Blitting text on the screen
- Add sounds effects
- Collision detection
- Create our very first game

Blitting text: define fonts

Step 1:

```
# Available system fonts
fonts = pygame.font.get_fonts()
for font in fonts:
    print(font)
```

Step 2:

```
# Define fonts
system_font = pygame.font.SysFont('calibri', 64)
custom_font = pygame.font.Font('RedBlock.ttf', 32)
```

Resource to download fonts:
fontspace.com/

Blitting text:
Blit the text on the screen

Step 1:

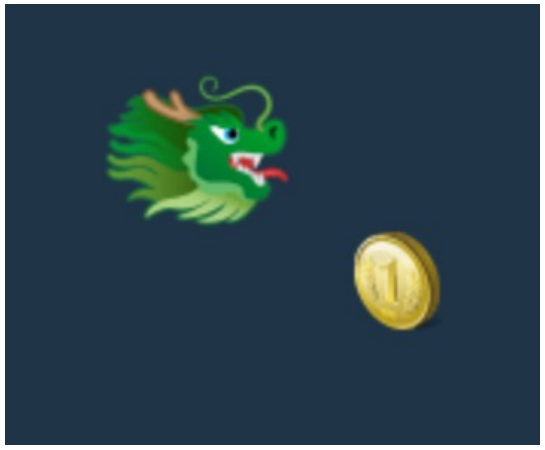
```
# Define text
system_text = system_font.render("Dragon Rule!", True, GREEN, DARKGREEN)
system_text_rect = system_text.get_rect()
system_text_rect.center = (WINDOW_WIDTH//2, WINDOW_HEIGTH//2)
```

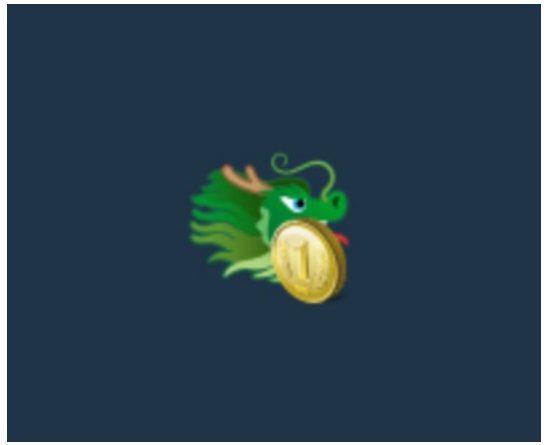
Step 2:

```
# The main game loop
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Blit the text at the given cordinates to our display
    display_surface.blit(system_text, system_text_rect)
    display_surface.blit(custom_text, custom_text_rect)

    # Update display
    pygame.display.update()

# End the game
pygame.quit()
```

Collision Detection

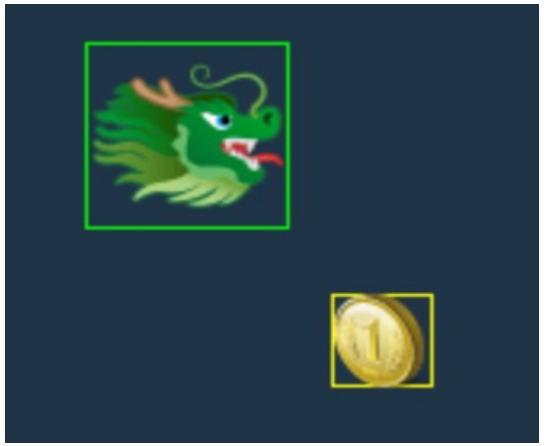
Let’s detect when the objects are touching each other:

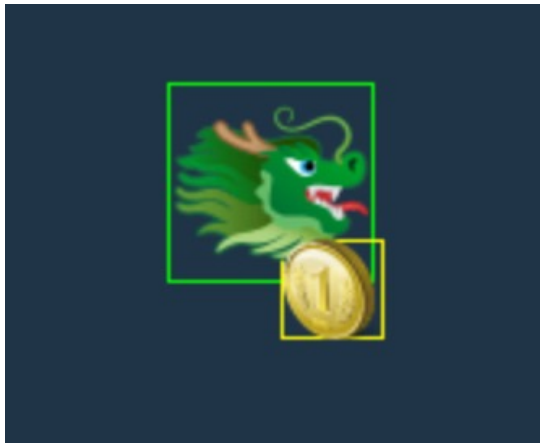




Small challenge for you:

Let’s draw rectangles to show the rect frame of our objects





Collision Detection:

Step 0: Lets draw rectangles to show the rect frame of our objects

```
# Draw rectangles to represent the rect's of each object
pygame.draw.rect(display_surface, (0, 255, 0), dragon_image_rect, 1)
pygame.draw.rect(display_surface, (255, 255, 0), coin_rect, 1)
```

Step 1: detect when the objects to collide each other and make some actions with them

```
#Check for collision between two rects
if dragon_right_rect.colliderect(coin_rect):
    print('HIT')
    coin_rect.x = random.randint(0, WINDOW_WIDTH - 32)
    coin_rect.y = random.randint(0, WINDOW_HEIGTH - 32)
```

Add sound effects and music: define sounds effects, set volume

Step 1:

```
# Define sound
sound_1 = pygame.mixer.Sound('sound_1.wav')
sound_2 = pygame.mixer.Sound('sound_2.wav')
```

Step 2:

```
# play the sound effects
sound_1.play()
pygame.time.delay(2000)
sound_2.play()
```

```
# Change the volume of a sound effect
sound_2.set_volume(.1)
sound_2.play()
```

Resource to download sound effects:
leshylabs.com

Feed the Dragon Setup 1:

1. Create a core
2. Set FPS and clock
3. Set game values

One small task for you!

Please create a core for our game:

1. Import necessary libraries
2. Create a display surface
3. Create the main game loop

1.The core of all Pygame projects:

```python
import pygame

#INITIALIZE pygame
pygame.init()
WINDOW_WIDTH = 600
WINDOW_HEIGHT = 300
display_surface = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))
pygame.display.set_caption("Hello Pygame!")

#The main game loop
running = True
while running:
    #Loop through a list of Event objects that have occured
    for event in pygame.event.get():
        print(event)
        if event.type == pygame.QUIT:
            running = False

    #End the game
pygame.quit()
```

2. Set FPS and clock:

Step 1:

```
# Set FPS and clock
FPS = 60
clock = pygame.time.Clock()
```

Step 2:

```
clock.tick(FPS)
```

3. Set game values

```
# Set game values
PLAYER_VELOCITY = 5
PLAYER_STARTING_LIVES = 5
COIN_STARTING_VELOCITY = 5
COIN_ACCELERATION = .5
BUFFER_DISTANCE = 100

score = 0
player_lives = PLAYER_STARTING_LIVES
coin_velocity = COIN_STARTING_VELOCITY
```

*Buffer distance its a kind of time delay. To keep our coin out of the screen.

```
coin_img = pygame.image.load("img/coin.png")
coin_rect = coin_img.get_rect()
coin_rect.x = (WINDOW_WIDTH + BUFFER_DISTANCE)
coin_rect.y = random.randint(64, WINDOW_HEIGTH - 32)
```

Feed the Dragon Setup 2 add the assets :

1. Set sounds and music
2. Define text
3. Set images

1. Set sounds and music

```
# Set sounds and music
coin_sound = pygame.mixer.Sound('sounds/coin_sound.wav')
miss_sound = pygame.mixer.Sound('sounds/miss_sound.wav')
miss_sound.set_volume(.1)
pygame.mixer.music.load('sounds/ftd_background_music.wav')
```

2. Set fonts and define text

Step 1:

```
# Set fonts
font = pygame.font.Font('AttackGraffiti.ttf', 32)
```

Step 2:

```
# Define text
score_text = font.render("Score: " + str(score), True, GREEN, DARKGREEN)
score_rect = score_text.get_rect()
score_rect.topleft = (10, 10)

title_text = font.render("Feed the dragon!", True, GREEN)
title_rect = title_text.get_rect()
title_rect.centerx = (WINDOW_WIDTH // 2)
title_rect.y = (10)

lives_text = font.render("Lives: " + str(player_lives), True, GREEN, DARKGREEN)
lives_rect = lives_text.get_rect()
lives_rect.topright = (WINDOW_WIDTH - 10, 10)
```

3. Set images

```
# Set images
player_image = pygame.image.load("img/dragon_right.png")
player_rect = player_image.get_rect()
player_rect.topleft = (10, WINDOW_HEIGTH // 2)

coin_img = pygame.image.load("img/coin.png")
coin_rect = coin_img.get_rect()
coin_rect.x = (WINDOW_WIDTH + BUFFER_DISTANCE)
coin_rect.y = random.randint(64, WINDOW_HEIGTH - 32)
```

Gameplay of our game inside of the main loop

1. Check if the user wants to move
2. Move the coin
3. Check for collisions (increasing speed every 2 times when we get a collision)
4. Game over condition

1. Check if user wants to move:

2. Move the coin:

3. Check for collisions (increasing speed every 2 times when we get collision)

One more task

Let’s make an incrementing score and decrementing lives

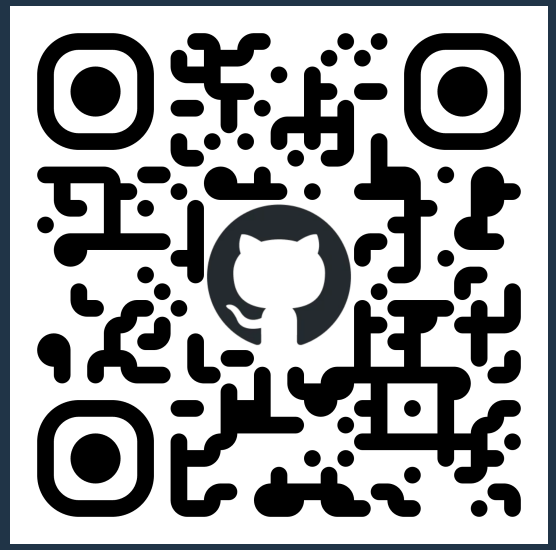
4. Game over condition:

```
#Pause the game until reset the game
pygame.mixer.music.stop()
is_paused = True
while is_paused:
    for event in pygame.event.get():
        # Quit
        if event.type == pygame.QUIT:
            is_paused = False
            running = False
        # Play again
        if event.type == pygame.KEYDOWN:
            score = 0
            pygame.mixer.music.play(-1, 0.0)
            player_lives = PLAYER_STARTING_LIVES
            player_rect.y = WINDOW_HEIGTH//2
            coin_velocity = COIN_STARTING_VELOCITY
            is_paused = False
```

Congratulations, we reached the goal!

Thank you!
Let’s keep in touch

https://www.linkedin.com/in/halloweex



https://github.com/halloweex