7️⃣

Lecture-7. PyGame

Lecture 7: Pygame

1. Getting Started

1.1 Introduction to Pygame

Pygame is a Python library used to create **games and multimedia applications**. It provides functionality for rendering graphics, handling input events, playing sound, and more.

1.2 Installing Pygame

To install Pygame, use the following command:

```
pip install pygame
```

To check if it is installed correctly, open Python and run:

```
import pygame
print(pygxame.__version__)
```

1.3 Creating a Basic Pygame Window

Every Pygame application follows a basic structure:

- Initialize Pygame
- Set up the game window
- Create a game loop that runs continuously

```
import pygame

pygame.init()

WIDTH, HEIGHT = 800, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Pygame Window")

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

    screen.fill((0, 0, 0))  # Clear screen with black color
    pygame.display.flip()  # Update the display

pygame.quit()
```

2. Working with Images

2.1 Loading and Displaying Images

Pygame allows loading and displaying images in various formats (PNG, JPG, BMP, etc.).

```
image = pygame.image.load("image.png")  # Load image
screen.blit(image, (100, 100))  # Draw image at (100, 100)
```

2.2 Scaling and Transforming Images

You can resize and rotate images:

```
scaled_image = pygame.transform.scale(image, (200, 150))  # Resize to 200x
150 pixels
rotated_image = pygame.transform.rotate(image, 45)  # Rotate by 45 degrees
```

2.3 Transparency in Images

To support transparency (alpha channel):

```
image = pygame.image.load("image.png").convert_alpha()
```

3. Music and Sound Effects

3.1 Loading and Playing Sounds

Pygame supports both **music** and **sound effects**.

```
pygame.mixer.init()
music = pygame.mixer.Sound("sound.wav")
music.play()
```

3.2 Playing Background Music

Use `pygame.mixer.music` for longer music tracks:

```
pygame.mixer.music.load("background.mp3")
pygame.mixer.music.play(-1)  # Loop indefinitely
```

3.3 Controlling Audio

```
pygame.mixer.music.pause()  # Pause music
pygame.mixer.music.unpause()  # Resume music
pygame.mixer.music.stop()  # Stop music
```

4. Geometric Drawing

4.1 Drawing Shapes in Pygame

Pygame allows drawing basic shapes using `pygame.draw`.

```
# Draw a red rectangle
pygame.draw.rect(screen, (255, 0, 0), (50, 50, 200, 100))

# Draw a blue circle
pygame.draw.circle(screen, (0, 0, 255), (300, 300), 50)

# Draw a green line
pygame.draw.line(screen, (0, 255, 0), (400, 100), (600, 400), 5)
```

To **realize geometric drawing** in **Pygame**, you can use built-in drawing functions like:

- `pygame.draw.line()` – Draw a line
- `pygame.draw.rect()` – Draw a rectangle
- `pygame.draw.circle()` – Draw a circle
- `pygame.draw.ellipse()` – Draw an ellipse
- `pygame.draw.polygon()` – Draw a polygon
- `pygame.draw.arc()` – Draw an arc

4.2 Drawing with Transparency

```
surface = pygame.Surface((200, 100), pygame.SRCALPHA)
pygame.draw.rect(surface, (255, 0, 0, 128), (0, 0, 200, 100))  # 50% transpare
```

```
nt red rectangle
screen.blit(surface, (100, 100))
```

5. Timer

5.1 Using Pygame's Built-in Timer

Timers control events and regulate game speed.

5.1.1 Controlling the Frame Rate

```
clock = pygame.time.Clock()
while running:
    clock.tick(60)  # Limits the loop to 60 FPS
```

In **Pygame**, `clock = pygame.time.Clock()` is used to **control the frame rate** of the game loop. This prevents the game from running too fast and ensures smooth animations.

How `pygame.time.Clock()` Works in This Example

1. `clock = pygame.time.Clock()`
   - Creates a clock object to control time.
2. `clock.tick(30)`
   - Limits the game loop to **30 frames per second (FPS)**.
   - Without this, the loop would run as fast as the computer allows, making movement unpredictable.
3. **Smooth movement**
   - The `tick(30)` ensures that movement is consistent, regardless of the computer's processing speed.

What Happens If You Don't Use `Clock.tick(FPS)`?

- The game **might run too fast**, depending on CPU speed.

- Animations could **become choppy** on slower devices.
- Movement speeds may **vary on different computers**.

5.1.2 Using Timer Events

Pygame allows setting up recurring events.

```
TIMER_EVENT = pygame.USEREVENT + 1
pygame.time.set_timer(TIMER_EVENT, 1000)  # Trigger event every 1000ms (1 second)
```

In the game loop:

```
for event in pygame.event.get():
    if event.type == TIMER_EVENT:
        print("Timer event triggered!")
```

Common Pygame Events

1. `pygame.QUIT` – When the user clicks the close button of the window.
2. `pygame.KEYDOWN` – When a key is pressed.
3. `pygame.KEYUP` – When a key is released.
4. `pygame.MOUSEBUTTONDOWN` – When a mouse button is pressed.
5. `pygame.MOUSEBUTTONUP` – When a mouse button is released.
6. `pygame.MOUSEMOTION` – When the mouse moves.
7. `pygame.VIDEORESIZE` – When the window is resized.
8. `pygame.WINDOWFOCUSGAINED` / `pygame.WINDOWFOCUSLOST` – When the window gains or loses focus.

List of Common `event.key` Values

```
pygame.K_UP       # Up arrow key
pygame.K_DOWN     # Down arrow key
```

```
pygame.K_LEFT      # Left arrow key
pygame.K_RIGHT     # Right arrow key
...
pygame.K_a   # 'A' key
pygame.K_b   # 'B' key
pygame.K_c   # 'C' key
...
pygame.K_0   # '0' key
pygame.K_1   # '1' key
pygame.K_2   # '2' key
...
pygame.K_RETURN   # Enter key
pygame.K_ESCAPE   # Escape key
pygame.K_BACKSPACE  # Backspace key
pygame.K_TAB      # Tab key
pygame.K_SPACE    # Spacebar
pygame.K_DELETE   # Delete key
pygame.K_HOME     # Home key
pygame.K_END      # End key
pygame.K_PAGEUP   # Page Up key
pygame.K_PAGEDOWN # Page Down key
pygame.K_INSERT   # Insert key
```

Conclusion

This lecture covered Pygame basics, including setting up a game window, working with images, adding sound, drawing shapes, and handling timers. With these fundamentals, you can start developing your own games and interactive applications!