Code Library · Stage 8

Six tiny programs that flip the game between MENU, PLAYING, GAME OVER, PAUSE and WIN. Copy, paste, tweak.

First, a tiny bit of background

A **state** is just the current mood of the game. Same screen, same code - but it behaves differently depending on what state it's in. We store the mood in a plain text variable called state, then use **if / elif** inside the loop to run the right code for the right mood.

```
# the heart of Stage 8
state = "MENU"                  # MENU / PLAYING / GAME_OVER

while True:
    if   state == "MENU":       # title screen, wait for SPACE
        ...
    elif state == "PLAYING":    # normal game
        ...
    elif state == "GAME_OVER":  # results screen, wait for R
        ...
```

elif means "else if". Python checks each line in order and only runs the FIRST one that matches - so exactly one mood runs per frame, never two. To switch moods, you just change the variable: state = "PLAYING". Next frame, the loop sees the new value.

How to use this handout

Each game has three parts: **what it does**, **the code**, and **challenges**. Copy any code block into the Trinket editor on the lesson page, press the green ■ **Run** button, then change a key, a state name, or a colour - see what happens.

Game 1 · Two-State Toggle

What it does

The simplest possible state machine. Press SPACE and the screen flips between green ON and red OFF. No game logic - just a variable that switches between two values.

```
import pygame

pygame.init()
screen = pygame.display.set_mode((600, 400))
pygame.display.set_caption("Two-State Toggle - press SPACE")

GREEN = (60, 220, 100)
RED   = (240, 60, 60)
WHITE = (255, 255, 255)

font = pygame.font.SysFont("Arial", 56, bold=True)

# the simplest game-state idea: just two moods
state = "ON"

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT: pygame.quit(); exit()
        if event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE:
            # flip the mood
            state = "OFF" if state == "ON" else "ON"

    if state == "ON":
        screen.fill(GREEN)
        img = font.render("ON", True, WHITE)
    else:
        screen.fill(RED)
        img = font.render("OFF", True, WHITE)

    rect = img.get_rect(center=(300, 200))
    screen.blit(img, rect)
    pygame.display.flip()
```

Try these one by one

Toggle 1 Add a THIRD state. Cycle through ON, OFF, and MAYBE using a counter mod 3.

```
states = ["ON", "OFF", "MAYBE"]
i = (i + 1) % 3
```

Toggle 2 Use the M key instead of SPACE.

```
if event.key == pygame.K_m:
```

Toggle 3 Show today's date when state is ON.

```
import datetime
img = font.render(str(datetime.date.today()), True, WHITE)
```

Game 2 · Menu & Play

What it does

A title screen with "MY GAME" + "Press SPACE to start". Press SPACE and you enter PLAYING mode where you can move a yellow square left and right with the arrow keys. Two states, one variable.

```
import pygame

pygame.init()
WIDTH, HEIGHT = 600, 400
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Menu & Play")

WHITE  = (255, 255, 255)
YELLOW = (255, 215, 0)
SPACE  = (0, 0, 50)

big   = pygame.font.SysFont("Arial", 56, bold=True)
small = pygame.font.SysFont("Arial", 22)

state = "MENU"             # MENU or PLAYING
player = pygame.Rect(280, 320, 40, 40)

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT: pygame.quit(); exit()

    keys = pygame.key.get_pressed()
    screen.fill(SPACE)

    if state == "MENU":
        screen.blit(big.render("MY GAME", True, YELLOW),    (180, 130))
        screen.blit(small.render("Press SPACE to start", True, WHITE), (190, 220))
        if keys[pygame.K_SPACE]:
            state = "PLAYING"

    elif state == "PLAYING":
        if keys[pygame.K_LEFT]  and player.x > 0:               player.x -= 5
        if keys[pygame.K_RIGHT] and player.x < WIDTH - 40:      player.x += 5
        pygame.draw.rect(screen, YELLOW, player)

    pygame.display.flip()
```

Try these one by one

Menu 1 Change the title to your own game name.

```
screen.blit(big.render("COOL RUNNER", True, YELLOW), (180, 130))
```

Menu 2 Make the menu wait for the M key (not SPACE) to start.

```
if keys[pygame.K_m]: state = "PLAYING"
```

Menu 3 Add a subtitle - a tiny tagline beneath the title.

```
screen.blit(small.render("A space adventure", True, WHITE), (200, 195))
```

Game 3 · Game Over Screen

What it does

Player at the bottom, enemy falling from the top. Move with LEFT/RIGHT to dodge. The moment the enemy touches you, state flips to GAME_OVER and a big red message appears. Press R to reset.

```
import pygame

pygame.init()
WIDTH, HEIGHT = 600, 400
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Game Over Screen - press R to restart")

GREEN  = (60, 220, 100)
RED    = (240, 60, 60)
WHITE  = (255, 255, 255)
YELLOW = (255, 215, 0)
SPACE  = (0, 0, 50)

big   = pygame.font.SysFont("Arial", 56, bold=True)
small = pygame.font.SysFont("Arial", 22)

state = "PLAYING"
player = pygame.Rect(280, 320, 40, 40)
enemy  = pygame.Rect(280, 0, 40, 40)

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT: pygame.quit(); exit()

    keys = pygame.key.get_pressed()
    screen.fill(SPACE)

    if state == "PLAYING":
        if keys[pygame.K_LEFT]  and player.x > 0:           player.x -= 5
        if keys[pygame.K_RIGHT] and player.x < WIDTH - 40:  player.x += 5
        enemy.y += 4
        if enemy.y > HEIGHT:
            enemy.y, enemy.x = 0, 280
        if player.colliderect(enemy):
            state = "GAME_OVER"

        pygame.draw.rect(screen, GREEN, player)
        pygame.draw.rect(screen, RED, enemy)

    elif state == "GAME_OVER":
        screen.blit(big.render("GAME OVER", True, RED),               (130, 130))
        screen.blit(small.render("Press R to play again", True, WHITE), (170, 220))
        if keys[pygame.K_r]:
            enemy.y, enemy.x = 0, 280
            state = "PLAYING"

    pygame.display.flip()
```

Try these one by one

Over 1 Make GAME OVER happen after just 2 collisions instead of 1. (Hint: add a hits counter.)

```
hits = 0
# in collision: hits += 1
# if hits >= 2: state = "GAME_OVER"
```

Over 2 Show the player's final position on the Game Over screen.

```
screen.blit(small.render("Final x: " + str(player.x), True, WHITE), (200, 250))
```

Over 3 Use a different key (ENTER) to restart.

```
if keys[pygame.K_RETURN]:
```

Game 4 · Pause Button

What it does

A green square bounces left and right by itself. Press P to PAUSE - the square freezes and "PAUSED" appears in big letters. Press P again to RESUME. Two-way toggle inside a state machine.

```
import pygame

pygame.init()
WIDTH, HEIGHT = 600, 400
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Pause Button - press P to pause / unpause")
clock = pygame.time.Clock()

GREEN  = (60, 220, 100)
YELLOW = (255, 215, 0)
WHITE  = (255, 255, 255)
SPACE  = (0, 0, 50)

big = pygame.font.SysFont("Arial", 64, bold=True)

state = "PLAYING"
square = pygame.Rect(0, 180, 40, 40)
dx = 5

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT: pygame.quit(); exit()
        # toggle pause whenever P is pressed
        if event.type == pygame.KEYDOWN and event.key == pygame.K_p:
            state = "PAUSED" if state == "PLAYING" else "PLAYING"

    screen.fill(SPACE)

    if state == "PLAYING":
        square.x += dx
        if square.x <= 0 or square.x >= WIDTH - 40:
            dx = -dx
        pygame.draw.rect(screen, GREEN, square)

    elif state == "PAUSED":
        pygame.draw.rect(screen, GREEN, square)             # still drawn, just frozen
        img = big.render("PAUSED", True, YELLOW)
        screen.blit(img, img.get_rect(center=(WIDTH//2, HEIGHT//2)))

    pygame.display.flip()
    clock.tick(60)
```

Try these one by one

Pause 1 Darken the screen when paused (use a darker SPACE colour).

```
screen.fill((0, 0, 25))
```

Pause 2 Use ESC instead of P.

```
if event.key == pygame.K_ESCAPE:
```

Pause 3 Add a third state "OVER" that triggers if the square bounces 10 times.

Game 5 · Win Condition

What it does

A simple click-to-score game with a goal. Click 5 times to win. Reach the target and the screen flips to YOU WIN! Press R to play again. Teaches the pattern "check after every change → maybe switch state".

```
import pygame

pygame.init()
WIDTH, HEIGHT = 600, 400
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Win Condition - get to 5 clicks!")

GREEN  = (60, 220, 100)
WHITE  = (255, 255, 255)
SPACE  = (0, 0, 50)

big   = pygame.font.SysFont("Arial", 56, bold=True)
small = pygame.font.SysFont("Arial", 22)

state = "PLAYING"
score = 0
TARGET = 5

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT: pygame.quit(); exit()
        if event.type == pygame.MOUSEBUTTONDOWN and state == "PLAYING":
            score += 1
            if score >= TARGET:
                state = "WIN"
        if event.type == pygame.KEYDOWN and event.key == pygame.K_r and state == "WIN":
            score = 0
            state = "PLAYING"

    screen.fill(SPACE)

    if state == "PLAYING":
        msg = "Score: " + str(score) + " / " + str(TARGET)
        screen.blit(big.render(msg, True, WHITE), (90, 170))

    elif state == "WIN":
        screen.blit(big.render("YOU WIN!", True, GREEN),                (170, 130))
        screen.blit(small.render("Press R to play again", True, WHITE), (180, 220))

    pygame.display.flip()
```

Try these one by one

Win 1 Raise the target to 20.

```
TARGET = 20
```

Win 2 Show a different colour for each milestone (every 5 points).

```
colour = (60, 220, 100) if score < 5 else (255, 215, 0)
```

Win 3 Add a LOSE state that triggers if you take longer than 5 seconds. (Hint: pygame.time.get_ticks().)

Game 6 · Three Lives

What it does

Game 3 plus a lives counter. The first two hits just take a life - only the third hit triggers GAME OVER. Top-left of the screen shows Lives: 3 / 2 / 1 as you take damage. This is how almost every arcade game works.

```
import pygame

pygame.init()
WIDTH, HEIGHT = 600, 400
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Three Lives")

GREEN  = (60, 220, 100)
RED    = (240, 60, 60)
WHITE  = (255, 255, 255)
SPACE  = (0, 0, 50)

big   = pygame.font.SysFont("Arial", 56, bold=True)
small = pygame.font.SysFont("Arial", 22)

state = "PLAYING"
player = pygame.Rect(280, 320, 40, 40)
enemy  = pygame.Rect(280, 0, 40, 40)
lives  = 3

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT: pygame.quit(); exit()
        if event.type == pygame.KEYDOWN and event.key == pygame.K_r and state == "GAME_OVER":
            lives = 3
            state = "PLAYING"

    keys = pygame.key.get_pressed()
    screen.fill(SPACE)

    if state == "PLAYING":
        if keys[pygame.K_LEFT]  and player.x > 0:           player.x -= 5
        if keys[pygame.K_RIGHT] and player.x < WIDTH - 40:  player.x += 5
        enemy.y += 4
        if enemy.y > HEIGHT:
            enemy.y, enemy.x = 0, 280
        if player.colliderect(enemy):
            lives -= 1
            enemy.y, enemy.x = 0, 280
            if lives <= 0:
                state = "GAME_OVER"

        pygame.draw.rect(screen, GREEN, player)
        pygame.draw.rect(screen, RED, enemy)
        screen.blit(small.render("Lives: " + str(lives), True, WHITE), (10, 10))

    elif state == "GAME_OVER":
        screen.blit(big.render("GAME OVER", True, RED),                  (130, 130))
        screen.blit(small.render("Press R to play again", True, WHITE), (170, 220))

    pygame.display.flip()
```

Try these one by one

Lives 1 Start with 5 lives.

```
lives = 5
```

Lives 2 Add a heart symbol next to the count.

```
screen.blit(small.render("❤ " + str(lives), True, WHITE), (10, 10))
```

Lives 3 When lives drops to 1, make the screen flash red briefly.

```
if lives == 1: screen.fill((40, 0, 0))
```

Where this takes us next

You can now flip between game moods - menu, playing, paused, won, lost. Lesson 9 introduces **HELPER FUNCTIONS** - little named blocks of code like draw_text() and reset_game(). They turn your game from a long messy script into a tidy collection of small, readable pieces. (You actually used one already in Lesson 7's colour_for(score)!)

Challenge: combine Games 2, 3, and 6 into a single program with MENU, PLAYING, and GAME OVER, with 3 lives. That's basically the full Galaxy Defender state machine!