Code Library · Stage 10

Six final programs about the GAME LOOP - the 60-times-per-second heartbeat of every video game. The last one is the complete Galaxy Defender, top to bottom.

First, a tiny bit of background

Every video game is built on the same simple idea: a **while running:** loop that repeats 60 times every second. Each time around, it does six things in order - read events, read held keys, update positions, check collisions, draw everything, and flip the screen.

```
# the heart of Stage 10
running = True
while running:
    # 1. EVENTS - quit / key presses
    # 2. KEYS   - which keys are held
    # 3. UPDATE - move bullets, enemies, etc
    # 4. COLLIDE - did anything hit anything?
    # 5. DRAW   - paint background + everything
    pygame.display.flip()    # 6a. show the new frame
    clock.tick(60)           # 6b. wait so we run at 60 FPS
```

Two lines do the real magic at the end: **pygame.display.flip()** reveals the new frame, and **clock.tick(60)** waits just long enough so the loop runs at 60 frames per second on any computer.

How to use this handout

Each program 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 one number at a time - that's how every coder learns. The last program is the COMPLETE Galaxy Defender. Run that and you have a real game.

Game 1 · Heartbeat (smallest possible loop)

What it does

The smallest pygame loop that can exist. Opens a space-blue window and keeps it open until you click X. No moving things, no input - just the heartbeat. Every other game adds work between the open and the flip.

```
import pygame

pygame.init()
screen = pygame.display.set_mode((600, 400))
pygame.display.set_caption("Heartbeat - the smallest game loop")
clock = pygame.time.Clock()

# the smallest possible "loop". Nothing happens - the window just stays open.
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    screen.fill((0, 0, 50))    # space-blue background
    pygame.display.flip()      # show the new frame
    clock.tick(60)             # wait so we run at 60 FPS

pygame.quit()
```

Try these one by one

Beat 1 Change the background colour to red (200, 30, 30).

```
screen.fill((200, 30, 30))
```

Beat 2 Try clock.tick(10) - the program will respond to clicks sluggishly because the loop only runs 10x/sec.

Beat 3 Print "tick!" every frame and watch how fast it scrolls in the console.

```
print("tick!")
```

Game 2 · One Square Animation

What it does

Game 1 + a yellow square that slides across the screen. The secret to ALL motion in games: change a number (x += 3), redraw, repeat. There is no other trick.

```
import pygame

pygame.init()
WIDTH, HEIGHT = 600, 400
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("One Square Animation")
clock = pygame.time.Clock()

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

# the secret of all motion: change a number, redraw, repeat
x = 0

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

    x += 3                              # update the number
    if x > WIDTH: x = -50               # loop back to the left

    screen.fill(SPACE)
    pygame.draw.rect(screen, YELLOW, (x, 180, 50, 50))
    pygame.display.flip()
    clock.tick(60)

pygame.quit()
```

Try these one by one

Anim 1 Speed it up: x += 12.

Anim 2 Make it bounce instead of teleporting. (Hint: dx = 3 then dx = -dx at the edges.)

```
if x <= 0 or x >= WIDTH - 50: dx = -dx
x += dx
```

Anim 3 Make a second square that moves DOWN as the first moves RIGHT.

Game 3 · The Six Phases

What it does

A real game loop with all six phases clearly labelled in comments. Move a green square with LEFT/RIGHT. Use this as your template for any game from now on - just fill the phases with your own logic.

```
import pygame

pygame.init()
WIDTH, HEIGHT = 600, 400
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("The Six Phases")
clock = pygame.time.Clock()

WHITE = (255, 255, 255)
GREEN = (0, 220, 100)
SPACE = (0, 0, 50)
font  = pygame.font.SysFont("Arial", 22)

player = pygame.Rect(280, 320, 40, 40)

running = True
while running:
    # ■■ PHASE 1: EVENTS ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # ■■ PHASE 2: KEYS ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
    keys = pygame.key.get_pressed()
    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

    # ■■ PHASE 3: UPDATE ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
    # (nothing else to update in this tiny demo)

    # ■■ PHASE 4: COLLISIONS ■■■■■■■■■■■■■■■■■■■■■■■■■■■■
    # (nothing to collide here)

    # ■■ PHASE 5: DRAW ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
    screen.fill(SPACE)
    pygame.draw.rect(screen, GREEN, player)
    screen.blit(font.render("Use LEFT / RIGHT", True, WHITE), (10, 10))

    # ■■ PHASE 6: FLIP + TICK ■■■■■■■■■■■■■■■■■■■■■■■■■■■
    pygame.display.flip()
    clock.tick(60)

pygame.quit()
```

Try these one by one

Six 1 Add UP/DOWN movement in PHASE 2.

```
if keys[pygame.K_UP]   and player.y > 0:           player.y -= 5
if keys[pygame.K_DOWN] and player.y < HEIGHT - 40:  player.y += 5
```

Six 2 Add an enemy and check collision in PHASE 4. Print "hit!" when they touch.

Six 3 Change the FPS to 30 in PHASE 6. Spot the slower feel.

```
clock.tick(30)
```

Game 4 · FPS Demo

What it does

Press UP/DOWN to change the frames-per-second live. Watch the yellow square move slowly at FPS 10 and almost teleport at FPS 240. Now you FEEL what clock.tick really does.

```
import pygame

pygame.init()
WIDTH, HEIGHT = 600, 400
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("FPS Demo - press UP/DOWN to change FPS")
clock = pygame.time.Clock()

YELLOW = (255, 215, 0)
WHITE  = (255, 255, 255)
SPACE  = (0, 0, 50)
font   = pygame.font.SysFont("Arial", 32)

x = 0
fps = 60

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP:   fps = min(fps + 10, 240)
            if event.key == pygame.K_DOWN: fps = max(fps - 10, 5)

    x = (x + 3) % WIDTH

    screen.fill(SPACE)
    pygame.draw.rect(screen, YELLOW, (x, 180, 50, 50))
    screen.blit(font.render("FPS: " + str(fps), True, WHITE), (10, 10))

    pygame.display.flip()
    clock.tick(fps)             # the speed changes live!

pygame.quit()
```

Try these one by one

FPS 1 Make the slowest FPS just 1 - watch the program crawl.

```
fps = max(fps - 10, 1)
```

FPS 2 Show the square's x position on screen too.

```
screen.blit(font.render("x: " + str(x), True, WHITE), (10, 50))
```

FPS 3 Take screenshots at 10 FPS and 120 FPS - notice how blurry slow looks vs how crisp fast looks.

Game 5 · Two Loops, One Window

What it does

Some games use multiple loops - one for the menu, one for the game. We wrap each in its own function. menu_loop() runs until you press SPACE, then play_loop() runs forever. Same pygame window the whole time.

```
import pygame

pygame.init()
WIDTH, HEIGHT = 600, 400
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Two Loops, One Window")
clock = pygame.time.Clock()

YELLOW = (255, 215, 0)
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)

# loop A - the menu
def menu_loop():
    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:
                return  # leave the menu, go to play
        screen.fill(SPACE)
        screen.blit(big.render("MY GAME", True, YELLOW),               (170, 130))
        screen.blit(small.render("Press SPACE to start", True, WHITE),  (185, 220))
        pygame.display.flip()
        clock.tick(60)

# loop B - the actual game
def play_loop():
    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()
        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
        screen.fill(SPACE)
        pygame.draw.rect(screen, GREEN, player)
        pygame.display.flip()
        clock.tick(60)

menu_loop()
play_loop()
```

Try these one by one

- **Two 1** Add a third loop for a Game Over screen, called after play_loop() returns.
- **Two 2** Make menu_loop() also exit on the Q key.

```
if event.key == pygame.K_q: pygame.quit(); exit()
```

Two 3 Show your name on the menu instead of "MY GAME".

Game 6 · The COMPLETE Galaxy Defender

All ten stages, top to bottom. This is the final game - menu, scoring, three lives, game over, restart, helper functions, and a 60-FPS game loop with every phase labelled. **Read it like a story.** Every line maps to a stage you built.

Part 1 of 2 - setup, variables, helpers:

```
# ■■■ COMPLETE GALAXY DEFENDER (Part 1 of 2) ■■■■■■■■■■
# Stages 1-9 - setup, helpers, all variables in place.

import pygame
import random

pygame.init()
WIDTH, HEIGHT = 800, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Galaxy Defender")
clock = pygame.time.Clock()

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

player = pygame.Rect(370, 520, 60, 40)
player_speed = 5

bullets = []
bullet_speed = 10

enemies = []
enemy_speed = 2
for i in range(5):
    enemies.append(pygame.Rect(random.randint(0, WIDTH - 50),
                               random.randint(-300, 0), 50, 50))

score, lives = 0, 3
font     = pygame.font.SysFont("Arial", 32)
big_font = pygame.font.SysFont("Arial", 72)

state = "MENU"

def draw_text(text, x, y, color=WHITE, big=False):
    f = big_font if big else font
    screen.blit(f.render(text, True, color), (x, y))

def reset_game():
    global score, lives, bullets
    score, lives, bullets = 0, 3, []
    player.x = 370
    for e in enemies:
        e.y = random.randint(-300, -50)
        e.x = random.randint(0, WIDTH - 50)

# ■■ the GAME LOOP continues on the next page... ■■
```

Game 6 · The COMPLETE Galaxy Defender (continued)

Part 2 of 2 - the game loop with all six phases:

```
# ■■■ COMPLETE GALAXY DEFENDER (Part 2 of 2) ■■■■■■■■■■
# Stage 10 - the game loop. All six phases visible.

running = True
while running:
    # PHASE 1: EVENTS
    for event in pygame.event.get():
        if event.type == pygame.QUIT: running = False
        if event.type == pygame.KEYDOWN:
            if state == "MENU" and event.key == pygame.K_SPACE:
                state = "PLAYING"
            elif state == "PLAYING" and event.key == pygame.K_SPACE:
                bullets.append(pygame.Rect(player.x + 27, player.y, 6, 15))
            elif state == "GAME_OVER" and event.key == pygame.K_r:
                reset_game(); state = "PLAYING"

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

    if state == "MENU":
        draw_text("GALAXY DEFENDER", WIDTH//2 - 260, 220, color=YELLOW, big=True)
        draw_text("Press SPACE to start", WIDTH//2 - 120, 320)
    elif state == "PLAYING":
        # PHASE 2: KEYS
        if keys[pygame.K_LEFT]  and player.x > 0:           player.x -= player_speed
        if keys[pygame.K_RIGHT] and player.x < WIDTH - 60:  player.x += player_speed
        # PHASE 3: UPDATE
        for b in bullets[:]:
            b.y -= bullet_speed
            if b.y < 0: bullets.remove(b)
        for e in enemies:
            e.y += enemy_speed
            if e.y > HEIGHT:
                e.y, e.x = random.randint(-200,-50), random.randint(0, WIDTH-50)
                lives -= 1
                if lives <= 0: state = "GAME_OVER"
        # PHASE 4: COLLISIONS
        for b in bullets[:]:
            for e in enemies:
                if b.colliderect(e):
                    if b in bullets: bullets.remove(b)
                    e.y, e.x = random.randint(-200,-50), random.randint(0, WIDTH-50)
                    score += 1
        # PHASE 5: DRAW
        pygame.draw.rect(screen, GREEN, player)
        for b in bullets: pygame.draw.rect(screen, YELLOW, b)
        for e in enemies: pygame.draw.rect(screen, RED, e)
        draw_text("Score: " + str(score), 10, 10)
        draw_text("Lives: " + str(lives), WIDTH - 130, 10)
    elif state == "GAME_OVER":
        draw_text("GAME OVER", WIDTH//2 - 180, 200, color=RED, big=True)
        draw_text("Final score: " + str(score), WIDTH//2 - 110, 300)
        draw_text("Press R to play again", WIDTH//2 - 130, 350, color=YELLOW)

    # PHASE 6: FLIP + TICK
    pygame.display.flip()
    clock.tick(60)

pygame.quit()
```

Try these one by one

Final 1 Add a WIN state when score >= 30. Show "YOU SAVED THE GALAXY!" in green.

Final 2 Make enemies speed up every 10 points. (Hint: if score % 10 == 0: enemy_speed += 1)

Final 3 Change EVERY colour to your own palette. Title too. Make it look like YOUR game.

Final 4 Take a screenshot of your finished, themed game and share it. You earned it.

Where to go next

You finished. Galaxy Defender is built. The 10-stage rocket framework you learned here - **Import** → **Setup** → **Body** → **End** - is the SAME framework that powers every 2D game ever written. Pick a new idea (a maze, a racer, a platformer, a top-down adventure) and try to build it from scratch using the same 10-stage outline.

If you want to keep going: try downloading real pygame on your computer to escape Trinket's limits, then look at libraries like **pygame-zero** (simpler) or **arcade** (modern). Or jump into a sister language like JavaScript and build games for the web with **p5.js**. Same framework, new tools. Have fun. ■