Code Library · Stage 7

Six tiny programs about counting, scoring, and painting text on the screen. Copy, paste, tweak.

First, a tiny bit of background

Two ideas drive this lesson:

- **A score is just a variable**. It starts at 0 and grows with score += 1 when something good happens. (That's short for score = score + 1.)
- **pygame.font** lets us paint that number on the screen. Three steps: pick a font (once), render the text into an image (every frame), then blit the image at an (x, y) position. Without these, pygame can't write a single letter on the screen.

```
# the heart of Stage 7
score = 0                                          # a number that changes
font  = pygame.font.SysFont("Arial", 32)           # pick a font ONCE

# ...inside the game loop, every frame:
img = font.render("Score: " + str(score), True, WHITE)
screen.blit(img, (10, 10))                         # paint it top-left
```

Watch out for one classic beginner error: "Score: " + score will crash because Python can't add a word to a number. Always wrap the number with str() first: "Score: " + str(score).

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 number, a colour, or a font size - see what happens.

Game 1 · Click Counter

What it does

The simplest pygame text program ever. Click anywhere and a counter on screen goes up by one. No game, no enemies - just text + a variable. This is the building block for every score in every game you'll ever build.

```
import pygame

pygame.init()
screen = pygame.display.set_mode((600, 400))
pygame.display.set_caption("Click Counter")

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

# 1. make a font ONCE near the top
font = pygame.font.SysFont("Arial", 48)

clicks = 0   # the number we'll show on screen

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        if event.type == pygame.MOUSEBUTTONDOWN:
            clicks += 1

    screen.fill(SPACE)

    # 2. render the text into an image
    img = font.render("Clicks: " + str(clicks), True, WHITE)

    # 3. blit (paste) the image onto the screen
    screen.blit(img, (180, 170))

    pygame.display.flip()

pygame.quit()
```

Try these one by one

Count 1 Use the SPACE key instead of mouse clicks.

```
if event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE: clicks += 1
```

Count 2 Make the number GIANT - change SysFont size from 48 to 120.

```
font = pygame.font.SysFont("Arial", 120)
```

Count 3 Make it count DOWN from 100 instead of up from 0.

```
clicks = 100
# inside the event: clicks -= 1
```

Game 2 · Score on Hit (a real shooter)

What it does

A complete mini-shooter - player at the bottom, 5 falling enemies, bullets, and a SCORE in the top-left that goes up every time you hit an alien. This is basically your Galaxy Defender, scored.

```
import pygame
import random

pygame.init()
WIDTH, HEIGHT = 800, 500
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Score on Hit")
clock = pygame.time.Clock()

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

player = pygame.Rect(370, 420, 60, 40)
bullets = []
enemies = [pygame.Rect(random.randint(0, WIDTH - 50),
                       random.randint(-300, 0), 50, 50) for _ in range(5)]

score = 0
font = pygame.font.SysFont("Arial", 32)

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:
            bullets.append(pygame.Rect(player.x + 27, player.y, 6, 15))

    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 - 60: player.x += 5

    for b in bullets[:]:
        b.y -= 10
        if b.y < 0: bullets.remove(b)

    for e in enemies:
        e.y += 2
        if e.y > HEIGHT:
            e.y, e.x = random.randint(-200, -50), random.randint(0, WIDTH - 50)

    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   # the magic line

    screen.fill(SPACE)
    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)
    screen.blit(font.render("Score: " + str(score), True, WHITE), (10, 10))
    pygame.display.flip()
    clock.tick(60)
```

Try these one by one

Hit 1 Bigger rewards: change score += 1 to score += 10.

Hit 2 Move the score to the top-right corner.

```
screen.blit(img, (WIDTH - 180, 10))
```

Hit 3 Change the score colour to YELLOW.

```
img = font.render("Score: " + str(score), True, YELLOW)
```

Game 3 · Big Bold Number

What it does

One huge yellow digit in the middle of the screen. Press UP to add 1, DOWN to subtract 1. Teaches centring with get_rect(center=...) - a tidy way to place text exactly where you want.

```
import pygame

pygame.init()
screen = pygame.display.set_mode((600, 400))
pygame.display.set_caption("Big Bold Number")

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

# size 200! a single giant digit
big = pygame.font.SysFont("Arial", 200, bold=True)

number = 7   # change this to anything!

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:   number += 1
            if event.key == pygame.K_DOWN: number -= 1

    screen.fill(SPACE)
    img = big.render(str(number), True, YELLOW)
    rect = img.get_rect(center=(300, 200))   # centre it
    screen.blit(img, rect)
    pygame.display.flip()

pygame.quit()
```

Try these one by one

Big 1 Use the LEFT and RIGHT arrows instead of UP and DOWN.

```
if event.key == pygame.K_LEFT:  number -= 1
if event.key == pygame.K_RIGHT: number += 1
```

Big 2 Change the colour every time the number changes - try RED for odd, GREEN for even.

```
colour = (255, 80, 80) if number % 2 else (60, 220, 100)
```

Big 3 Add a word: render "Level " + str(number) instead of just the number.

```
img = big.render("Level " + str(number), True, YELLOW)
```

Game 4 · Two-Line Scoreboard

What it does

A big yellow SCORE on top with a small white instruction beneath it. Two different font sizes side by side. Click anywhere to earn 5 points.

```
import pygame

pygame.init()
screen = pygame.display.set_mode((600, 400))
pygame.display.set_caption("Two-Line Scoreboard")

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

# two fonts of different sizes
big   = pygame.font.SysFont("Arial", 56, bold=True)
small = pygame.font.SysFont("Arial", 20)

score = 0

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        if event.type == pygame.MOUSEBUTTONDOWN:
            score += 5    # +5 per click

    screen.fill(SPACE)

    # line 1: big yellow score
    line1 = big.render("SCORE: " + str(score), True, YELLOW)
    screen.blit(line1, (60, 130))

    # line 2: small white hint underneath
    line2 = small.render("Click anywhere to earn 5 points", True, WHITE)
    screen.blit(line2, (60, 200))

    pygame.display.flip()

pygame.quit()
```

Try these one by one

Two 1 Add a THIRD line for the current level.

```
line3 = small.render("Level: 1", True, WHITE)
screen.blit(line3, (60, 230))
```

Two 2 Change the small font to italic.

```
small = pygame.font.SysFont("Arial", 20, italic=True)
```

Two 3 Centre both lines horizontally using rect.get_rect(center=(300, ...)).

Game 5 · Colour-Changing Score

What it does

The score's COLOUR changes as the number grows: grey, then green, then orange, then red. Uses a tiny helper function colour_for(score) - a sneak peek of Lesson 9 (Helpers).

```
import pygame

pygame.init()
screen = pygame.display.set_mode((600, 400))
pygame.display.set_caption("Colour-Changing Score")

SPACE = (0, 0, 50)
GREY  = (180, 180, 180)
GREEN = (60, 220, 100)
ORANGE = (240, 140, 50)
RED   = (240, 60, 60)

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

score = 0

def colour_for(s):
    if s < 5:   return GREY
    if s < 15:  return GREEN
    if s < 30:  return ORANGE
    return RED      # 30+ is HOT!

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        if event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE:
            score += 1

    screen.fill(SPACE)
    img = font.render("Score: " + str(score), True, colour_for(score))
    screen.blit(img, (130, 160))
    pygame.display.flip()

pygame.quit()
```

Try these one by one

Hot 1 Add a fifth tier: at 50+ make it MAGENTA.

```
if s < 50: return RED
return (255, 0, 200)
```

Hot 2 Move the thresholds: green starts at 20 instead of 5.

```
if s < 20: return GREY
if s < 35: return GREEN
```

Hot 3 Change the score by 5 per press instead of 1.

```
score += 5
```

Game 6 · High Score Tracker

What it does

Two numbers: current score (white) and HIGH score (yellow). Press SPACE to earn a point. Press R to reset the current score - but the high stays. This is how every arcade game ever made works.

```
import pygame

pygame.init()
screen = pygame.display.set_mode((600, 400))
pygame.display.set_caption("High Score Tracker - press SPACE to score, R to reset")

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

font = pygame.font.SysFont("Arial", 36)

score = 0
high  = 0

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_SPACE:
                score += 1
                if score > high:        # update the high score on the fly
                    high = score
            if event.key == pygame.K_r:
                score = 0               # reset score, KEEP high

    screen.fill(SPACE)
    screen.blit(font.render("Score: " + str(score), True, WHITE),  (40, 100))
    screen.blit(font.render("HIGH:  " + str(high),  True, YELLOW), (40, 160))
    pygame.display.flip()

pygame.quit()
```

Try these one by one

High 1 Make the high score persist in a third "all-time" variable that NEVER resets, even on R.

High 2 Add a celebration: print("NEW HIGH!") whenever score breaks the high.

```
if score > high:
    high = score
    print("NEW HIGH!")
```

High 3 Display the difference: "You need 5 more to beat the high".

```
diff = high - score
screen.blit(font.render("Need: " + str(diff), True, WHITE), (40, 220))
```

Where this takes us next

Your game now has STAKES. Players want to push the score higher. But what happens when they win? When they lose? Right now the game just keeps going forever. Lesson 8 introduces **game states** - a Start screen, a Playing screen, and a Game Over screen - so the game has a beginning, a middle, and an end.

Challenge: take Game 2 (Score on Hit), add a HIGH variable like Game 6, and make the high reset only when you press R. That's a full retro arcade in 80 lines!