Code Library · Stage 2

Tiny pygame programs you can copy, paste, and run on the brand-new window you set up in Stage 2.

First, a tiny bit of background

In Stage 2 you opened a **window**. Pygame calls this window the **screen**, but you can also think of it as your **canvas** or **workspace** — a blank rectangle you can paint on.

Three things are worth knowing about the canvas before we draw on it:

1. **The coordinate grid.** Every pixel on the screen has an (x, y) address. (0, 0) is the **top-left** corner (not the bottom-left, like in maths). x grows to the right, y grows *downwards*. So (400, 300) is right in the middle of an 800x600 window.
2. **Colours are numbers.** A colour is a tuple of three numbers: *(red, green, blue)*, each 0 to 255. (0, 0, 0) is black. (255, 255, 255) is white. (255, 0, 0) is pure red. Mix the three to make any colour you want.
3. **The event loop.** Pygame programs sit inside a while running: loop. Each time around the loop, we check for events (mouse clicks, key presses, the close button), redraw everything, and flip the screen so the player sees it. Without the loop, the window would close instantly.

How to use this handout

Each game below has three parts: **what it does**, **the code**, and **challenges** for you to try. Copy any code block into the Trinket editor on the lesson page, then press the green ■ **Run** button. Change a number, change a colour, change a word - see what happens. That's how you learn.

Want to try your own idea? Ask an AI like Claude or ChatGPT for something like *'Write me a short pygame program that draws a smiley face'*, then paste the code into Trinket and click Run.

Game 1 · Hello Window

What it does

Opens a 600x400 window, paints it sky-blue, and keeps it open until you click the close button. The simplest possible pygame program - perfect for testing that your Trinket works.

```
import pygame

pygame.init()
screen = pygame.display.set_mode((600, 400))
pygame.display.set_caption("Hello Window")

running = True
while running:
    screen.fill((135, 206, 235))   # sky blue
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
    pygame.display.flip()

pygame.quit()
```

Try these one by one

Challenge 1 Change the title to your own name.

```
pygame.display.set_caption("My Awesome Game")
```

Challenge 2 Change the window size to 800x600.

```
screen = pygame.display.set_mode((800, 600))
```

Challenge 3 Paint it pink.

```
screen.fill((255, 192, 203))
```

Game 2 · Click the Balloon

What it does

A red balloon (a circle) appears in the middle of the sky. Click it - the window closes. That's the whole game! No score, no timer, no random colours, no moving balloon. Tiny but complete.

```
import pygame

pygame.init()
screen = pygame.display.set_mode((800, 600))
pygame.display.set_caption("Click the Balloon")

balloon_x = 400
balloon_y = 300
balloon_radius = 50

running = True
while running:
    screen.fill((135, 206, 235))                          # sky background
    pygame.draw.circle(screen, (255, 0, 0),
                       (balloon_x, balloon_y), balloon_radius)  # red balloon

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        if event.type == pygame.MOUSEBUTTONDOWN:
            mouse_x, mouse_y = pygame.mouse.get_pos()
            distance = ((mouse_x - balloon_x) ** 2
                        + (mouse_y - balloon_y) ** 2) ** 0.5
            if distance <= balloon_radius:
                print("You popped the balloon!")
                running = False

    pygame.display.flip()

pygame.quit()
```

What's new in this one

Only a few new ideas you haven't seen before:

- **Draw a circle:** `pygame.draw.circle(screen, colour, (x, y), radius)`
- **Detect a mouse click:** `if event.type == pygame.MOUSEBUTTONDOWN:`
- **Get the mouse position:** `mouse_x, mouse_y = pygame.mouse.get_pos()`
- **Check if the click hit the balloon:** compare the distance between the mouse and the balloon's centre to the radius.

Try these one by one

Balloon 1 Make the balloon blue.

```
(0, 0, 255)
```

Balloon 2 Move the balloon to the top-left corner.

```
balloon_x = 200
balloon_y = 150
```

Balloon 3 Make it bigger.

```
balloon_radius = 80
```

Balloon 4 Make it tiny.

```
balloon_radius = 20
```

Balloon 5 Change the sky. Try black, white, or grass-green.

```
screen.fill((0, 0, 0))
screen.fill((255, 255, 255))
screen.fill((0, 180, 0))
```

Balloon 6 Don't end the game when the balloon is clicked - just print a message.

```
# replace this line:
running = False
# with this:
print("Great job!")
```

Game 3 · Spotlight (circle follows your mouse)

What it does

A yellow circle follows your mouse around a dark window like a spotlight. Move your mouse — the spotlight chases it. This is the secret to ALL smooth movement in games: read the new position every frame, then redraw.

```
import pygame

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

running = True
while running:
    screen.fill((20, 20, 30))   # dark background

    # ask pygame: where is the mouse right now?
    x, y = pygame.mouse.get_pos()

    # draw a yellow circle there
    pygame.draw.circle(screen, (255, 230, 50), (x, y), 40)

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

    pygame.display.flip()

pygame.quit()
```

Try these one by one

Spotlight 1 Make the spotlight smaller.

```
pygame.draw.circle(screen, (255, 230, 50), (x, y), 15)
```

Spotlight 2 Make the spotlight green.

```
pygame.draw.circle(screen, (0, 255, 0), (x, y), 40)
```

Spotlight 3 Change the background to deep purple.

```
screen.fill((40, 0, 60))
```

Game 4 · Colour Switcher

What it does

Press the **SPACE** key to flip the background colour between blue, red, green, and yellow. Your first keyboard-controlled program!

```
import pygame

pygame.init()
screen = pygame.display.set_mode((600, 400))
pygame.display.set_caption("Colour Switcher - press SPACE")

colours = [(50, 100, 200), (220, 60, 60), (60, 180, 90), (240, 210, 70)]
i = 0

running = True
while running:
    screen.fill(colours[i])

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        if event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE:
            i = (i + 1) % len(colours)   # next colour, wrap around

    pygame.display.flip()

pygame.quit()
```

Try these one by one

Switcher 1 Add a fifth colour to the list. Try hot pink: (255, 105, 180).

```
colours = [(50, 100, 200), (220, 60, 60), (60, 180, 90),
           (240, 210, 70), (255, 105, 180)]
```

Switcher 2 Use the LEFT and RIGHT arrow keys instead of SPACE.

```
if event.key == pygame.K_LEFT:  i = (i - 1) % len(colours)
if event.key == pygame.K_RIGHT: i = (i + 1) % len(colours)
```

Game 5 · Smiley Face

What it does

Combines several shapes on one canvas: a yellow circle (the face), two black circles (the eyes), and an arc (the smile). Practise placing things with (x, y) coordinates.

```
import pygame

pygame.init()
screen = pygame.display.set_mode((500, 500))
pygame.display.set_caption("Smiley Face")

running = True
while running:
    screen.fill((30, 30, 40))                              # dark background

    pygame.draw.circle(screen, (255, 220, 0), (250, 250), 150)  # face
    pygame.draw.circle(screen, (0, 0, 0),     (200, 220), 18)   # left eye
    pygame.draw.circle(screen, (0, 0, 0),     (300, 220), 18)   # right eye

    # the smile (an arc)
    smile_rect = pygame.Rect(170, 230, 160, 100)
    pygame.draw.arc(screen, (0, 0, 0), smile_rect, 3.5, 6.0, 6)

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

    pygame.display.flip()

pygame.quit()
```

Try these one by one

- **Smiley 1** Move the eyes closer together by changing 200 and 300 to 220 and 280.
- **Smiley 2** Make the face purple: change (255, 220, 0) to (180, 50, 200).
- **Smiley 3** Add a tiny red circle for a nose at (250, 270), radius 8.

```
pygame.draw.circle(screen, (255, 0, 0), (250, 270), 8)
```

Where these games take us next

You've now used a real canvas, drawn shapes, detected mouse clicks, and responded to keys. Those are the building blocks of every pygame game. In Lesson 3 we'll give our colours friendly names like RED and YELLOW so we don't have to remember (255, 0, 0) every time.

Challenge: pick any game above, change three things, run it, see what happens. Then ask an AI for a new game idea that uses what you've learned!