10 Pygame Projects

A learning path to become genuinely comfortable with pygame.
Ten games. Suggested order. Each one teaches one or two new ideas.

The path at a glance

Build these in order. By the time you reach Snake or Space Explorer, every skill in there will feel familiar because you'll have practised it in an earlier project. Don't skip ahead — each project intentionally introduces just one or two new ideas.

1	■	**Balloon Pop**	Mouse clicks · randomness · scoring
2	■	**Catch the Star**	Keyboard input · collision · score
3	■	**Rainbow Drawing**	Turtle · keys · random colours
4	■	**Dodge the Falling Blocks**	Animation · spawning · collision
5	■	**Guess the Colour**	random.choice · timers · key checks
6	■	**Colour Maze**	Turtle · obstacles · simple collide
7	■	**Simple Road Game**	Scrolling · rects · random obstacles
8	■	**Alien Shooter**	Bullets · multiple enemies · loop
9	■	**Snake**	Lists · growth · random food
10	■	**Space Explorer**	Animated bg · many objects · difficulty

Every project ships with: **a description**, **the skills you'll learn**, **starter ideas or code**, and **3-5 challenges** to make it yours. Project 1 (Balloon Pop) is fully written out as a worked example — paste it straight into your Trinket and start tweaking.

■ Project 1 · Balloon Pop

Click on the moving balloon before it gets away. Earn 1 point per pop.

What you'll build

A balloon (a coloured circle) appears at a random spot. Click it and you earn a point — the balloon jumps to a NEW random position with a NEW random colour. Keep going for as long as you can.

Skills you'll practise

`import` `Game window` `Draw circles` `Random numbers` `Mouse clicks` `Score` `Game loop`

The complete code

```
import pygame
import random

pygame.init()

WIDTH, HEIGHT = 800, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Balloon Pop")
font = pygame.font.SysFont(None, 40)

balloon_radius = 40
balloon_x = random.randint(balloon_radius, WIDTH - balloon_radius)
balloon_y = random.randint(balloon_radius, HEIGHT - balloon_radius)
balloon_colour = (random.randint(50, 255),
                  random.randint(50, 255),
                  random.randint(50, 255))
score = 0

running = True
while running:
    screen.fill((135, 206, 250))                              # sky-blue bg
    pygame.draw.circle(screen, balloon_colour,
                       (balloon_x, balloon_y), balloon_radius)
    score_text = font.render("Score: " + str(score), True, (0, 0, 0))
    screen.blit(score_text, (20, 20))

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        if event.type == pygame.MOUSEBUTTONDOWN:
            mx, my = pygame.mouse.get_pos()
            distance = ((mx - balloon_x) ** 2 + (my - balloon_y) ** 2) ** 0.5
            if distance <= balloon_radius:
                score += 1
                balloon_x = random.randint(balloon_radius, WIDTH - balloon_radius)
                balloon_y = random.randint(balloon_radius, HEIGHT - balloon_radius)
                balloon_colour = (random.randint(50, 255),
                                  random.randint(50, 255),
                                  random.randint(50, 255))

    pygame.display.flip()

pygame.quit()
```

How it works (the short version)

1. **import pygame, random** — bring in the libraries.
2. **pygame.init() + set_mode** — turn pygame on and open a window.
3. **balloon variables** — random x, random y, random RGB colour.
4. **while running:** — the heartbeat that repeats 60 times per second.
5. **screen.fill + draw.circle** — paint sky and balloon every frame.
6. **MOUSEBUTTONDOWN** — Python's word for "the player clicked".
7. **distance check** — if the click landed inside the balloon, score up + reset balloon.
8. **pygame.display.flip()** — reveal the new frame.

■ Try these one at a time

Pop 1 Make the balloon bigger.

```
balloon_radius = 70
```

Pop 2 Black background.

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

Pop 3 Always-red balloon.

```
balloon_colour = (255, 0, 0)
```

Pop 4 Make it tiny (much harder!).

```
balloon_radius = 10
```

Pop 5 Square instead of a circle - swap pygame.draw.circle for pygame.draw.rect.

Pop 6 Print 'Pop!' to the console every successful click.

```
print("Pop!")
```

Pop 7 Custom window title.

```
pygame.display.set_caption("Pop the Balloon!")
```

■ Stretch challenges (for when those feel easy)

- Add a 30-second timer and a final score screen.
- Add 3 lives - lose one every time you click but miss.
- Two balloons on screen at once.
- A high score that persists for the session.
- Random balloon SIZES (use random.randint for the radius too).
- A scenic background: sun, clouds, grass (use pygame.draw.circle / pygame.draw.rect).
- A pop sound effect (pygame.mixer.Sound + a free .wav from sfxr.me).

■ Project 2 · Catch the Star

Move a small square with the arrow keys; touch the gold star to score and respawn it somewhere new.

Skills you'll practise

Keyboard input **Collision (rect.colliderect)** **Score counter**

How to build it

Make a pygame.Rect for the player. Use pygame.key.get_pressed() for movement. Make a pygame.Rect for the star at a random position. If they collide, score += 1 and the star jumps to a new random spot.

■ Try these one at a time

Star 1 Use a bigger star so it's easier.

Star 2 Star spawns ONLY in the top half.

```
star.y = random.randint(0, HEIGHT//2)
```

Star 3 Add a TIMER counting down from 30 seconds.

■ Project 3 · Rainbow Drawing

Press the arrow keys to drive a turtle around a canvas. Each step paints in a fresh random colour.

Skills you'll practise

Turtle module **Keyboard listeners (turtle.onkey)** **Random colours**

How to build it

Use turtle.Screen() + turtle.Turtle(). Bind UP/LEFT/RIGHT to functions that call t.forward / t.left / t.right and change t.color to a random RGB triple each time.

■ Try these one at a time

- **Rain 1** Cycle through 7 set colours instead of random.
- **Rain 2** Press SPACE to clear the screen with t.clear().
- **Rain 3** Press +/- to make the pen size bigger or smaller.

■ Project 4 · Dodge the Falling Blocks

Slide a player along the bottom of the screen while coloured blocks rain down. Survive as long as you can.

Skills you'll practise

Animation (changing y each frame) **Random spawning** **Collision check + game over**

How to build it

Player = pygame.Rect at the bottom, moves LEFT/RIGHT. Blocks = a list of pygame.Rect, each with its own falling speed. When a block leaves the screen, recycle it back to the top with random x. If player.colliderect(block): game over.

■ Try these one at a time

- **Dodge 1** Speed up every 10 seconds.
- **Dodge 2** Random block COLOURS and SIZES.
- **Dodge 3** Add a score = number of seconds survived.

■ Project 5 · Guess the Colour

A big square cycles to a random colour every second. Press R, G, or B to guess — match its dominant colour to score.

Skills you'll practise

random.choice **pygame.time.get_ticks() for timers** **Multi-key input**

How to build it

Every 1000ms, pick a new (r, g, b). Player presses R / G / B; you compare which channel of the colour is highest and award a point if they match.

■ Try these one at a time

Guess 1 Add a YELLOW key (Y) for when red+green dominate.

Guess 2 Decrease the timer each round so the game speeds up.

Guess 3 Add lives: lose one when you guess wrong.

■ Project 6 · Colour Maze

Use the turtle to reach a goal while avoiding randomly placed coloured obstacle squares.

Skills you'll practise

turtle drawing **Manual coordinate checks** **Win/lose conditions**

How to build it

Use turtle.Screen(). Plot a goal somewhere on the canvas. Place ~10 random coloured squares as walls. The player turtle moves with arrow keys; check if turtle.distance(obstacle) < 30 to detect a hit.

■ Try these one at a time

Maze 1 Add a time limit before the maze auto-resets.

Maze 2 Random goal position every round.

Maze 3 Coloured obstacles do different things (red = lose, blue = teleport).

■ Project 7 · Simple Road Game

Drive a car left and right between lanes while randomly-coloured traffic flies down the road.

Skills you'll practise

Scrolling background **Multiple lanes** **Random obstacles**

How to build it

Player = a car rect at the bottom, only moves LEFT/RIGHT between 3 fixed lane positions. Traffic = a list of car rects falling at random speeds + colours. Background = two grey rectangles scrolling DOWN to fake forward motion.

■ Try these one at a time

- **Road 1** Add a YELLOW dashed line down the middle.
- **Road 2** Score = miles travelled (1 mile per second).
- **Road 3** Add a fuel gauge that depletes — collect green fuel pickups.

■ Project 8 · Alien Shooter

Galaxy Defender simplified: ship at the bottom, aliens drift down, press SPACE to fire.

Skills you'll practise

Bullets (list pattern) **Multiple enemies** **Game loop with collisions**

How to build it

You've literally built this in Lessons 1–10 of the course! Use the cumulative code as your starting point. Then theme it differently — change aliens to monsters, bullets to magic bolts, background to a forest.

■ Try these one at a time

Alien 1 Make each alien a different colour from a palette of 4.

Alien 2 Add a BOSS alien every 30 points - bigger + takes 3 hits to destroy.

Alien 3 Add power-ups that drop randomly and give 5 seconds of fast-fire.

■ Project 9 · Snake

The classic. A growing snake chases food. Don't crash into yourself.

Skills you'll practise

Lists (snake body segments) **Growing the list on food** **Random food positions**

How to build it

snake = a list of (x, y) tuples. Every frame, prepend a new head position and pop the tail (unless food was eaten - then keep the tail to grow). Game over if the head touches any other segment OR the wall.

■ Try these one at a time

- **Snake 1** Wrap the snake around the edges instead of game-over on wall hits.
- **Snake 2** Each food is a random colour and worth 1-3 points.
- **Snake 3** Speed up every 5 points eaten.

■ Project 10 · Space Explorer

Fly a spaceship through scrolling space, collecting random-coloured stars and dodging asteroids. Difficulty ramps up over time.

Skills you'll practise

Many moving objects **Animated parallax background** **Difficulty curve**

How to build it

Three lists: stars (collect them), asteroids (dodge them), starfield (background dots scrolling at 1-3 different speeds for depth). Every 30 seconds, spawn rate + speed both increase.

■ Try these one at a time

Explore 1 Add 2 background star layers at different speeds for parallax.

Explore 2 Random asteroid sizes - bigger ones hurt more.

Explore 3 Final boss every 100 stars: a huge asteroid you have to shoot 10 times.

■ Make every game your own

These tweaks work in EVERY project. Change ONE thing and re-run — that's the fastest way to learn what each line does.

Background colours to try

```
screen.fill((0, 0, 0))         # Black
screen.fill((135, 206, 235))   # Sky blue
screen.fill((34, 139, 34))     # Grass green
screen.fill((25, 25, 112))     # Midnight blue
screen.fill((255, 192, 203))   # Pink
```

Quick scenery (paste before your game objects)

```
pygame.draw.circle(screen, (255, 255,   0), (700,  80), 40)   # Sun
pygame.draw.rect  (screen, (  0, 180,   0), (0, 500, 800, 100))  # Grass
pygame.draw.circle(screen, (255, 255, 255), (100, 100), 25)    # Cloud
```

Use random to make every game feel different

```
colour = (random.randint(0, 255),
          random.randint(0, 255),
          random.randint(0, 255))
x      = random.randint(50, 750)
y      = random.randint(50, 550)
radius = random.randint(20, 60)
```

Where this takes you

Finish all 10 and you'll have practised every core skill behind 2D game development: drawing, animation, input, randomness, collisions, scoring, game states, lists, and the all-important game loop. That's enough to build almost any 2D game you can imagine — or to graduate to real pygame on your own laptop, pygame-zero, the arcade library, or even Unity / Godot.

One last thing: the first game you finish without copying any code is the moment you stop being a learner and start being a developer. Aim for that. ■