



BitWithBite

PYGAME MASTERY WORKSHEET
Particle Effects

Chapter: Pygame Mastery · Level ★★☆☆ · Time: 30 min

SCORE

--- / 20

Name _____ Class _____ Date _____

AFTER THIS WORKSHEET YOU CAN

- ✓ Build a Particle with position, velocity and lifetime
- ✓ Manage bursts with a ParticleSystem class
- ✓ Configure fire, smoke, rain, snow & sparks
- ✓ Explain why object pooling helps at scale

QUICK RECAP

Smoke, fire, rain, snow, explosions and magic effects all come from the same underlying idea: spawn a burst of tiny short-lived objects that move, fade, and disappear. This module builds one reusable ParticleSystem that powers every effect by changing only its emit parameters.



Section A · Concept Check

• INTERMEDIATE

4 × 1 = 4

1 What core piece of state lets a particle know when to disappear?

- ☐ A. Its color
- ☐ B. A lifetime value compared against its accumulated age
- ☐ C. Its radius
- ☐ D. The screen resolution

3 How does ParticleSystem.update() remove dead particles?

- ☐ A. Pygame removes them automatically
- ☐ B. A list comprehension keeps only particles where is_dead() is False
- ☐ C. They are never removed
- ☐ D. The screen is cleared

2 What mainly distinguishes "fire" particles from "smoke" particles?

- ☐ A. They need completely separate classes
- ☐ B. Different emit parameters using the same Particle/ParticleSystem classes
- ☐ C. Fire cannot use gravity
- ☐ D. Smoke needs a different engine

4 What does self.vel *= 0.98 do each frame?

- ☐ A. Instantly stops the particle
- ☐ B. Applies slight drag, gradually slowing velocity
- ☐ C. Doubles the particle's speed
- ☐ D. Changes the particle's color



Section B · Problem Solving

• INTERMEDIATE

2 + 3×3 = 11

5 A single particle needs position, velocity and a countdown that determines when it dies.

6

The ParticleSystem's methods spawn a burst of particles configured for one specific effect.

7 Name the two reusable classes that power every particle effect in this module.

8 Give 3 emit parameters that change between the "fire" and "snow" effects.

9 Why is rebuilding self.particles with a list comprehension every frame fine for hundreds of particles but potentially wasteful for thousands?



Section C · Challenge

• CHALLENGE

5

10 A boss explosion needs 5,000 spark particles and the game starts dropping frames. Using two techniques from this module, explain how you would fix it.

Reflection — the most useful thing I learned: _____

A ___/4

B ___/11

C ___/5

Total ___/20

Teacher's Signature

Parent's Signature

⌘ ANSWER KEY — FOLD OR CUT BEFORE HANDING OUT

1-B 2-B 3-B 4-B | 5 lifetime (compared to age) 6 emit (e.g. emit_explosion) 7 = Particle and ParticleSystem 8 = e.g. velocity/speed, color, lifetime (also radius) 9 = Rebuilding a list each frame allocates new memory; at hundreds of particles the cost is trivial, but at thousands the constant allocation/garbage collection becomes a real performance hit | 10 = Use an object pool that reuses dead Particle instances instead of allocating new ones every frame, and/or cap the spawn count for very large bursts