BitWith Bite

PYGAME MASTERY WORKSHEET

Game Architecture

Module 18 of 20 · Level ★★★ · Time: 40 min

SCORE ___ / 20

Name _____ Class _____ Date _____

AFTER THIS WORKSHEET YOU CAN

✓ Design a Scene base class ✓ Switch scenes with a SceneManager ✓ Centralize asset loading

✓ Use a config module for constants

QUICK RECAP

A Scene base class (handle_events/update/draw) lets every screen — menu, gameplay, game over — follow the same shape, while a SceneManager holds the current scene and swaps it on request. Centralized asset loading and a config.py constants module keep the codebase organized as it grows.

Section A · Concept Check • ADVANCED 4 × 1 = 4

1 What three methods does a typical Scene base class define?

- ◯ A. init, load, save
- ◯ B. handle_events, update, draw
- ◯ C. play, pause, stop ◯ D. read, write, close

2 What does a SceneManager hold?

- ◯ A. Every asset in the game
- ◯ B. The currently active Scene, swapped on transition
- ◯ C. The player's save file ◯ D. The frame rate

3 Why centralize asset loading in one helper function?

- ◯ A. It's required by Pygame
- ◯ B. Avoids scattered, duplicated load calls and enables caching
- ◯ C. It makes images smaller ◯ D. It's not useful

4 What typically lives in a config.py module?

- ◯ A. Constants like SCREEN_WIDTH, FPS, colors
- ◯ B. The actual game loop ◯ C. Sprite images
- ◯ D. Sound files

Section B · Problem Solving • INTERMEDIATE 2 + 3×3 = 11

5 A Scene subclass overrides handle_events, update and _____.

6 Switching scenes is usually done by calling manager._____(new_scene).

7 Write the SceneManager.switch_to() method that replaces the current scene.

8 Why is scattering magic numbers (e.g. 800, 600, 60) across many files a maintenance problem?

9 What's the benefit of the game loop only ever calling current_scene.update()/draw() instead of a giant if/elif chain checking game state everywhere?

Section C · Challenge • CHALLENGE 5

10 Design the scene flow for a simple game: MenuScene → PlayScene → GameOverScene → back to MenuScene. Describe what triggers each transition.

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-A | 5 draw 6 switch_to 7 = def switch_to(self, new_scene): self.scene = new_scene 8 = if the screen size ever changes, every file with a hardcoded 800/600 must be found and updated individually, which is error-prone and easy to miss a spot 9 = the game loop code stays simple and identical regardless of how many scenes exist — it just calls whatever the current scene's own update/draw are, instead of growing a huge if/elif chain that must be edited every time a new state is added | 10 = MenuScene switches to PlayScene when the player clicks "Play"; PlayScene switches to GameOverScene when the player's health reaches 0 (or they complete the level); GameOverScene switches back to MenuScene when the player clicks "Return to Menu" (or presses a key)