BitWith Bite

PYGAME MASTERY WORKSHEET

Tile Maps

Module 13 of 20 · Level ★★★ · Time: 40 min

SCORE ___ / 20

Name _____ Class _____ Date _____

AFTER THIS WORKSHEET YOU CAN

✓ Represent a level as a 2D grid ✓ Slice a tileset into individual tiles ✓ Render a tile grid with correct offsets ✓ Mark solid tiles for collision

QUICK RECAP

A tile map represents a level as a 2D grid of tile IDs. A tileset image is sliced into individual tiles, then rendered by looping row/column indices and blitting each tile at (col*TILE_SIZE, row*TILE_SIZE) — with a separate collision layer marking which tiles block movement.

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

1 How is a tile map level typically represented in code?

- ◯ A. A single image file
- ◯ B. A 2D list/array of tile ID integers
- ◯ C. A single Rect ◯ D. A sound file

2 What is the Tiled editor used for?

- ◯ A. Editing sound effects
- ◯ B. Visually building levels, exporting JSON/CSV your code loads
- ◯ C. Compiling Python ◯ D. Rendering fonts

3 What formula gives a tile's pixel position from its (row, col)?

- ◯ A. (row, col) ◯ B. (col*TILE_SIZE, row*TILE_SIZE)
- ◯ C. (row+col, row−col) ◯ D. Random

4 What is a collision layer?

- ◯ A. A visual-only decoration layer
- ◯ B. Data marking which tiles block movement
- ◯ C. A sound effect layer
- ◯ D. The background music track

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

5 Slicing a tileset image into individual tiles uses Surface._____() or a source rect blit.

6 Combining a tile map with a camera offset produces a _____ map.

7 Write the nested loop pattern (pseudocode is fine) to render every tile in a 2D grid called level_map.

8 Why keep collision data separate from the visual tile ID, rather than hardcoding "tile ID 3 is always solid" in the render loop?

9 What's one advantage of using the Tiled editor over hand-writing a 2D Python list for a large level?

Section C · Challenge • CHALLENGE 5

10 Your level is 200 tiles wide but the screen only shows 25 at a time. Explain how the camera offset from Module 15 combines with the tile rendering loop so only the visible tiles need to actually be considered (a basic optimization idea, not full code).

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 subsurface 6 scrolling 7 = for row, line in enumerate(level_map): for col, tile_id in enumerate(line): blit tile_images[tile_id] at (col*TILE, row*TILE) 8 = keeping a lookup table (a set of "solid" IDs, or a parallel collision grid) lets you redesign visuals or reuse a tile ID for multiple looks without rewriting collision logic scattered through the render code 9 = Tiled provides a visual, drag-and-drop interface for placing tiles, layers, and object markers — much faster and less error-prone than manually typing out large 2D arrays of numbers by hand | 10 = using the camera's offset, compute which column/row range is currently visible on screen (roughly camera.offset.x / TILE_SIZE to that plus screen_width / TILE_SIZE) and only loop over/render those tiles, skipping the rest of the 200-wide map entirely each frame