BitWith Bite

PYGAME MASTERY WORKSHEET

Optimization & Publishing

Module 19 of 20 · Level ★★★ · Time: 35 min

SCORE ___ / 20

Name _____ Class _____ Date _____

AFTER THIS WORKSHEET YOU CAN

✓ Apply key blitting optimizations ✓ Profile before optimizing ✓ Package a game with PyInstaller ✓ Publish a game to itch.io

QUICK RECAP

convert()/convert_alpha() is the cheapest speed win in Pygame. Profile with cProfile before guessing where slowdowns are. PyInstaller (--onefile --windowed) packages a script into a standalone executable, and itch.io is the standard free platform for publishing indie Pygame games.

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

1 What is the single biggest, cheapest blitting speedup?

- ◯ A. Reducing screen resolution
- ◯ B. Calling .convert() or .convert_alpha() after loading images
- ◯ C. Removing all sound
- ◯ D. Using fewer colors

2 Why profile before optimizing?

- ◯ A. Profiling is required by Pygame
- ◯ B. To measure where the actual slowdown is instead of guessing
- ◯ C. It makes the game faster automatically
- ◯ D. It compresses images

3 What does --onefile do in a PyInstaller command?

- ◯ A. Compiles only one function
- ◯ B. Bundles everything into a single executable file
- ◯ C. Deletes extra files
- ◯ D. Runs the game once

4 What is itch.io commonly used for?

- ◯ A. Writing Python code online
- ◯ B. Publishing and hosting indie/hobby games for free
- ◯ C. Compiling C++
- ◯ D. Managing databases

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

5 The --_____ PyInstaller flag suppresses the console window for GUI apps.

6 Python's built-in profiling module is called c_____.

7 Write the full PyInstaller command to package game.py as a single windowed executable.

8 Why does using pygame.sprite.Group's built-in draw()/update() often outperform hand-written loops?

9 Name two things you should prepare (besides the executable) before publishing a game to itch.io.

Section C · Challenge • CHALLENGE 5

10 Your game runs at 15 FPS instead of 60. Before touching any code, describe the exact first step you should take to find out WHY, and why skipping straight to "optimizing" without this step is risky.

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 windowed 6 cProfile 7 = pyinstaller --onefile --windowed game.py 8 = Group methods are written and tested by Pygame's maintainers to iterate sprites efficiently and consistently, reducing bugs and often avoiding redundant work compared to ad-hoc manual loops 9 = a game description/store page with screenshots or a trailer, and a README or set of controls/instructions for players | 10 = run a profiler (e.g. cProfile) first to see exactly which function(s) are consuming the most time; optimizing blindly risks spending effort rewriting code that isn't actually the bottleneck while the real slow function goes untouched