Untitled 2

Alternative Option: The "Engine vs. Game" Split

Instead of organizing by file type (putting all scripts in `scripts/` and all enums in `enums/`), I recommend organizing by dependency.

This structure highlights that you have built a Mini-Game Engine (your CG requirements) and a Game (the Claw Machine) that uses it.

📂 Recommended Structure

```
Claw-Machine-Arcade-Game/
├── assets/                    # Keep distinct types separated here
│   ├── sprites/
│   ├── textures/
│   └── fonts/
│
├── engine/                    # YOUR "CG LIBRARY" (The stuff you get graded on)
│   ├── __init__.py
│   ├── raster.py              # Bresenham, Polygon filling
│   ├── transformations.py     # Matrix math, Affine transforms
│   └── primitives.py          # Basic shapes (Circle, Rect)
│
├── game/                      # THE CLAW MACHINE LOGIC
│   ├── __init__.py
│   ├── entities/              # Specific game objects
│   │   ├── claw.py
│   │   └── prize.py
│   ├── scenes/                # Game states
│   │   ├── menu.py
│   │   └── play.py
│   └── constants.py           # Game-specific settings (colors, speeds)
│
├── main.py                    # The entry point (keep this at root!)
└── requirements.txt
```

Why this is better for *this* specific context:

1. The "Professor Friendly" Root: By keeping `main.py` at the very root (not inside `src/`), you guarantee that anyone can unzip your project and run `python main.py` without path errors.
2. Logical Separation:
   - **`engine/`**: Contains code that *could* be used in any game (math, line drawing). This uses zero game logic.
   - **`game/`**: Contains code specific to the Claw Machine (Claw physics, Prize logic). This imports from `engine`.

3. **Import Simplicity:**
    - From `main.py`, you just do: `from game.scenes.menu import MenuScene`.
    - Inside the game, you do: `from engine.transformations import translate`.

Concrete "Next Steps" for You

If you want to adopt this "Engine vs. Game" structure without breaking everything, here is the order of operations:

1. **Create the folders:** Make an `engine` folder and a `game` folder.
2. **Move the "Math":** Move `raster.py` and `transformations.py` into `engine/`.
3. **Move the "Logic":** Move `scenes/`, `entities/`, and `config.py` into `game/`.
4. **Fix Imports:**
    - Open your game files.
    - Change `import raster` to `from engine import raster`.
    - Change `import config` to `from game import config`.

Here are the specific import changes required to make the **Engine vs. Game** structure work.

This guide assumes you have moved `main.py` to the **root folder** (outside of `src`), created the `engine/` and `game/` folders, and added empty `__init__.py` files to every folder.

1. Changes in `main.py` (The Root Entry Point)

Because `main.py` is now at the very top, it acts as the controller that imports from both your "engine" and your "game" packages.

In `main.py`:

```
# --- BEFORE ---
# from scenes.menu_scene import MenuScene
# from constants import SCREEN_WIDTH, SCREEN_HEIGHT

# --- AFTER (The Fix) ---
import pygame
from game import config  # Replaces 'import constants'
from game.scenes.menu_scene import MenuScene
from game.scenes.claw_machine_scene import ClawMachineScene
from engine.game_loop import GameLoop  # If you moved game_loop to engine

def main():
    pygame.init()
    screen = pygame.display.set_mode((config.SCREEN_WIDTH, config.SCREEN_HEIGHT))
    # ... rest of your code
```

---

2. Changes in `game/entities/claw.py` (Entities)

Your game entities (like the Claw) need to use the math tools from the `engine` and the settings from the `game`.

In `game/entities/claw.py`:

```python
# --- BEFORE ---
# import transformations
# import raster
# from constants import CLAW_SPEED

# --- AFTER (The Fix) ---
from engine import transformations
from engine import raster
from game import config

class Claw:
    def update(self):
        # Example usage
        self.x += config.CLAW_SPEED

    def draw(self, screen):
        # Example usage of engine math
        transformations.translate(self.x, self.y)
```

---

3. Changes in `game/scenes/claw_machine_scene.py` (Scenes)

Scenes act as the "glue." They import Entities (from `game`) and draw them using Raster functions (from `engine`).

In `game/scenes/claw_machine_scene.py`:

```python
# --- BEFORE ---
# from entities.claw import Claw
# from entities.prize import Prize
# import raster

# --- AFTER (The Fix) ---
from game.entities.claw import Claw
from game.entities.prize import Prize
from engine import raster

class ClawMachineScene:
    def render(self, screen):
        # Now you can use the engine's rasterizer
        raster.draw_line(screen, ... )
```

4. Changes in `engine/raster.py` (Graphics Core)

This file should remain pure. It should rarely need to import anything from `game/`. It should only depend on other `engine` files or standard libraries (like `math` or `pygame`).

In `engine/raster.py`:

- Likely Change: None.
- Why: If `raster.py` previously imported `transformations`, and they are now neighbors in the `engine/` folder, you can keep `import transformations` or change it to `from . import transformations` (relative import) to be safe.

💡 Critical Requirement: `__init__.py`

For these imports (`from game.entities ...`) to work, you must ensure every folder has an empty file named `__init__.py`.

```
Claw-Machine-Arcade-Game/
├── engine/
│   ├── __init__.py  ⟵ DON'T FORGET THIS
│   └── ...
├── game/
│   ├── __init__.py  ⟵ THIS ONE TOO
│   ├── entities/
│   │   ├── __init__.py
│   │   └── ...
│   └── ...
```