15-112, Week 5, Simon

Today's lab is one big project: build **Simon**, the classic 1978 memory game. Work with a partner, and see how far you can get during lab. Every milestone below builds on the previous one, so a working (if ugly) game at each step beats a beautiful half-game.

Your instructor will demo the game live. You don't need to perfectly mimic the look, and you should feel free to improve on it. (But don't make it look worse...)

Setup: use the same `cmu_graphics` setup as last week. Make a `lab05` folder with the `cmu_graphics` folder inside it, create a file called `simon.py`, and build from there. Use `onAppStart`, `redrawAll`, `onMousePress`, `onKeyPress`, and `onStep` (set `app.stepsPerSecond` to control how often `onStep` fires).

Developer notes

General features of the game:

- The game screen consists of 4 colored squares and an area to display the player's answers. (Mouse clicks in the answer display area do nothing.)
- The pattern flashes, then the timer activates, decrementing in 0.5 second intervals.
- Mouse presses only register *after* the pattern has been flashed.
- The user enters colors by clicking the squares.
- The game continues, round after round, until the user makes a mistake.
- After game over, the game can be restarted with `r`.

How the countdown timer works:

- The timer delay for the game is 100 ms (that is, `app.stepsPerSecond = 10`).
- The timer decrements by 0.5 after every five timer fires (`onStep` calls).

How the flashing works:

- Each color is "flashed" for 0.5 seconds, and there are 0.5 seconds in between colors.

Hint: with `onStep` running every 100 ms, think about how many steps 0.5 seconds is, and about what state you need (a step counter, an index into the pattern, a "flashing or not" flag) to know what to draw on each step.

Suggested milestones

You don't have to follow these, but if you're not sure where to start, this order works well. Test at the end of each one before moving on.

1. **Game screen layout.** Draw the 4 colored squares and the answer display area. Make sure that nothing is hardcoded to a particular window size.
2. **Pattern generation and flashing.** Generate a random pattern, then flash it one color at a time. Input must be ignored while the pattern is flashing.
3. **Player input.** After the pattern finishes, accept mouse clicks on the squares. Show each answer in the answer display area as it's entered.
4. **Checking answers.** Detect mistakes and go to game over. If the player completes the pattern correctly, go to the next round.
5. **Countdown timer.** After the pattern flashes, start the timer and decrement it by 0.5 every five `onStep` calls. End the game when it hits 0.
6. **Game over and restart.** Show a game over screen, and restart the game when the user presses `r`.