The goal of this lab is to get started with using the more advanced aspects of graphics modules that allow you to create interactive applications.

There is nothing to submit.

There are 2 options for interactive graphics that are available on the lab machines, Pygame, and the Zelle graphics you used in your early drawing assignment. Both of these programs are generally easy to install and run, have good documentation, and have been used extensively by students in the past which means they are well tested and likely known by the lab assistants, myself, Mike Tie, etc. Therefore I very strongly encourage you to use one of these if you choose to use graphics in your final project. That said, you may request to use something else in your final project proposal, just talk to me first!

In general, Zelle graphics is simpler to learn to use and works well for turn-based games or other projects that draw relatively basic scenes and use only very simple animations. Many grid-based games work well using Zelle graphics, such as tic-tac-toe as a basic example. Pygame has a steeper learning curve, but allows for more complex/fast animation if that is what you want. Pretty much any action-type game with fast animation and frequent user input will require pygame. You might also want pygame for something like a simulation of the solar system where the animations are simple but require many objects to be moving on the screen at the same time.

Provided here are example “games” in each of the graphics options. Your task is to go through each and understand the basics of how things work in that graphics system. A guideline for each is below. If you get through the basic functionality checks of Zelle and think you might be interested in using pygame, skip the more advanced Zelle and try the basics of pygame. Otherwise if you want to just focus on Zelle that’s fine too.

1. Zelle graphics
- requirements to use: only graphics.py module in same folder as your program
- documentation: textbook, docstrings/comments within the code in graphics.py
- example game: mouseandkeys.py

To start, open mouseandkeys.py and look at the top of the file for the list of all of the interactive things you can do. Run the program and try each thing, make sure you understand what it does in the simulation, then look at the code to start to understand how each action is coded.
Check that you understand basic functionality:
- change the location that individual characters appear when you type them to the bottom left corner instead of the upper left. Make them appear red instead of black.
- change so that the small circles that appear when you click move up the screen instead of down.
Check that you understand more advanced functionality:
- fix “reset” so clears text that appears when you press “q” (right now it starts over except for that text stays if it was there)
- display time counter that shows number of seconds since started simulation, changes

value only once per second
- change so that when small circles reach edge of screen (top or bottom), they don't go off of the screen, instead they turn around and go back in the other direction.

2. Pygame
- requirements to use: use lab computer or install Pygame (see Mike Tie for help installing on your own computer)
- documentation: https://www.pygame.org/docs/
- example game: targetPractice.py

To start run the game and follow the instructions on the first screen to understand how the actions work. You should move the cross-hairs around (hold down the left mouse button while moving the mouse), click on boxes and see them change color, test what happens if you miss on 3 clicks. Then look through the code and see if you can understand where things are being done in the program itself.
Check that you understand functionality:
- change the boxes to always start colored red and change to blue when clicked.
- change so only 1 box moving around
- change so that when successfully click on a box it speeds up a little as well as changes color.
- currently "score" on endscreen is always just "000". Change to track how many times successfully click in box, and display that number as the player's score on the endscreen.
- add score display that's always showing along with the number of lives, update score each time box is clicked.