Introduction

Welcome to CS 195n! In this assignment you’ll be creating the basic framework of the game engine you will be developing for the rest of the semester. You’ll also create a simple Tic-Tac-Toe game on top of your game engine. All assignments in this course will follow a similar format of requiring both engine features and a game built on the engine, so they all have two sets of requirements: one for the engine, and one for the game. Your handin must meet both sets of requirements to receive credit, and you must keep your engine code separate from your game code (as noted in the Global Requirements).

To get the support code for this assignment (and for all future assignments), copy the contents of the starter directory in /course/cs195n/support/ to your project directory for Tic. This should add a cs195n package to your src directory, which will contain the four support code classes.

Design Check

- How will you define an application?
- How will you define a screen?
- What plans for a UI kit do you have (bounding boxes, aspect ratios)?

Engine Requirements

- **Basic Framework.** The engine must separate an application from screens, and support an application with multiple screens.

- **Drawing.** The engine must draw the current screen on every “draw” event (originating from the support code). Each screen must be able to define the way in which it is drawn independently from other screens.

- **Mouse events.** The engine must be able to process mouse events (originating from the support code) and allow each screen to define how they are handled.

- **Keyboard events.** The engine must be able to process keyboard events (originating from the support code) and allow each screen to define how they are handled.

- **Screen updates.** The current screen must update itself on every “tick” event (originating from the support code).

- **Resizing.** The engine must handle resize events, and allow the current screen to update any relevant internal state when the draw area is resized. It must also ensure that size information is preserved when the current screen changes – if the current screen changes after a resize event, the new current screen must be aware of the correct window size as well.
Game Requirements

Your game must implement the rules of Tic-Tac-Toe: two players, X and O, take turns marking squares on a 3x3 grid with their respective symbols. If a player succeeds in placing three symbols in a horizontal, vertical, or diagonal row, that player wins. If all the squares are filled without either player completing a row, the game is a draw. Both players can be human players; you do not need an AI opponent for this assignment.

In addition, your game must:

- Put a time limit on each player’s turn. If the player does not make a move when the time expires, it becomes the other player’s turn and the timer resets.
- Show the countdown timer in milliseconds on the game screen.
- Have at least two screens: an in-game screen and another screen, such as a title or win/lose screen.
- Make use of mouse events somewhere.
- Make use of keyboard events somewhere.
- Clearly display which player’s turn it is.
- At the end of the game, effectively communicate which player won, or if it was a draw.
- Display the state of the game on a square board that scales with window size. The board must remain square at all times, no matter the window’s aspect ratio.
- Have UI elements that fill as much space as they can while keeping the board square. This means you cannot draw UI elements at a fixed size.
- It must be possible to start a new game without restarting the program.

Keep in mind that there are also global requirements that apply to every checkpoint. Be sure to confirm that you meet these as well!

Suggested Extras

If you meet the requirements listed above, you will receive credit for the assignment. However, if you have extra time and wish to make your game more interesting, you might want to implement some of these extra features. Note that you cannot receive extra credit for going beyond the assignment’s requirements because assignments do not get scores or letter grades, but the students who playtest your game will certainly appreciate if you make it more fun to play.

- Let the user pick a larger-size board, such as 4x4, 5x5, or even NxN.
- Add a simple AI player and the option to play against the computer. You could even support multiple difficulties! If you do this, remember that the option to play with two human players must still work as in the original requirements.
Handing In

Hand in the entire directory tree for your project, including both your engine and game code. You must also include a README file that describes how to verify each requirement, and an INSTRUCTIONS file that describes how to play your game, as specified in the Global Requirements. If you’re using Eclipse, execute `cs195n_handin tic` from the root directory of your Eclipse project (something like `/path/to/a/workspace/tic`). If you’re not using Eclipse, the directory you hand in must include an ant build script that compiles your code into an executable jar.