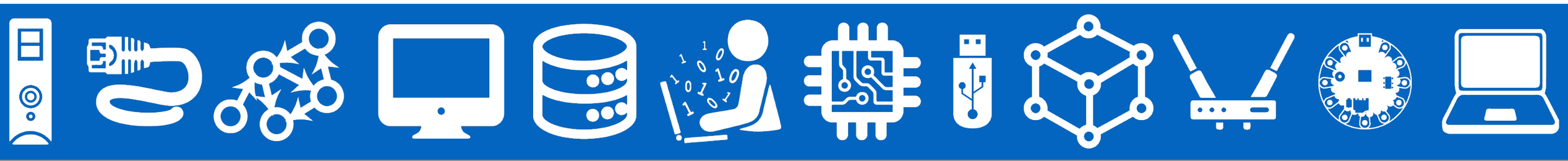
CS134:
Tic Tac Toe 3



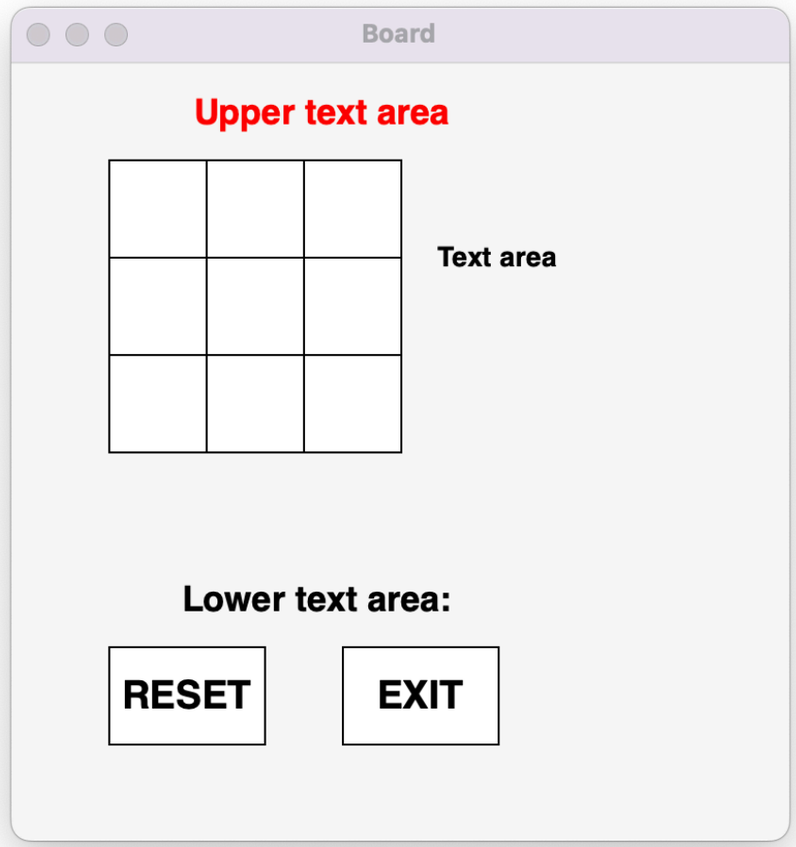
Announcements & Logistics

- **HW 9** due Monday @ 10 pm
- **Lab 9 Boggle**: two-week lab
  - **Part 1** due next Wed/Thur 10 pm
  - **Part 2** due following week
  - Both parts have a **prelab** due at the beginning of lab
  - Can solve jointly with partner/ or individually and then discuss
  - Have it ready **on a sheet of paper at the start of lab**

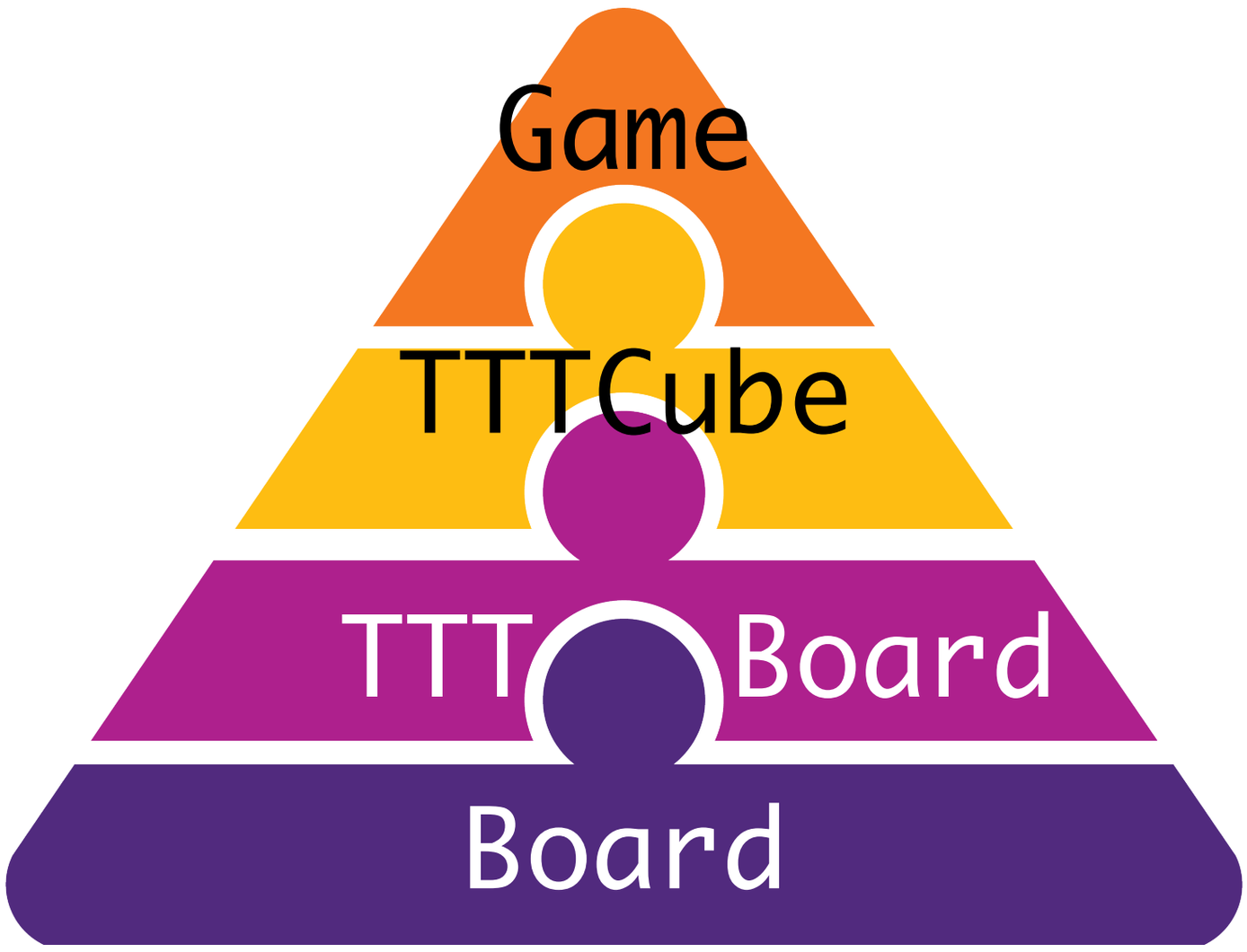
Do You Have Any Questions?

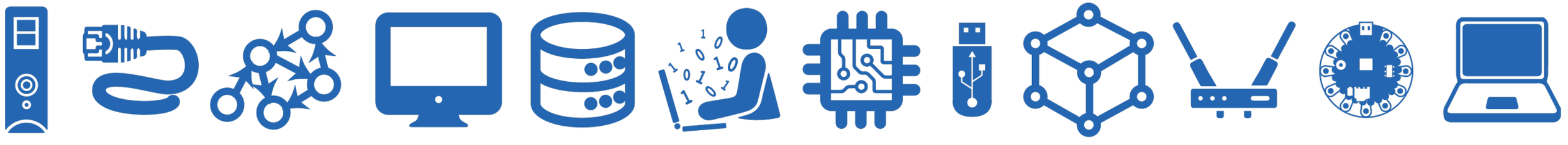
Where We Are

- Designed classes to represent a TTTBoard and TTTCube
- Before that, we designed a graphical `Board` class to display a board
- **Today** we will bring these together:
  - Design a graphical tic-tac-toe game
- **Next time**:
  - Finish up TTTGame class
  - Discuss differences between TTT and Boggle



TTT Game Logic





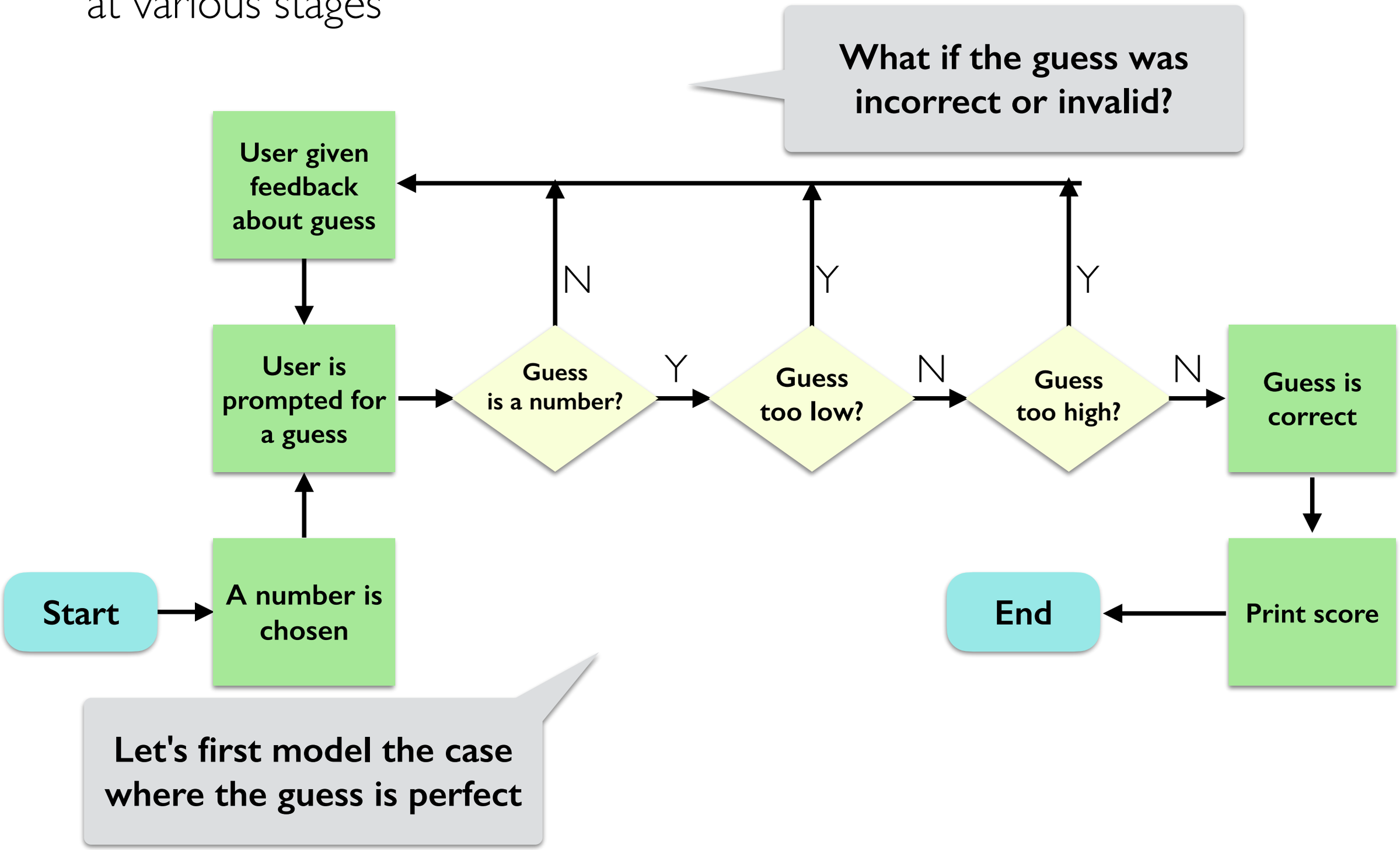
Other Games?

- Have we implemented any other games this semester?
  - Actually, yes! The "number guessing game"
- General idea:
  - A number is chosen (either at random or by the game runner)
  - A player repeatedly guesses numbers until the chosen number is guessed
    - Hints of "higher" or "lower" are given after each incorrect guess
  - The player's score is the number of guesses needed to identify the number

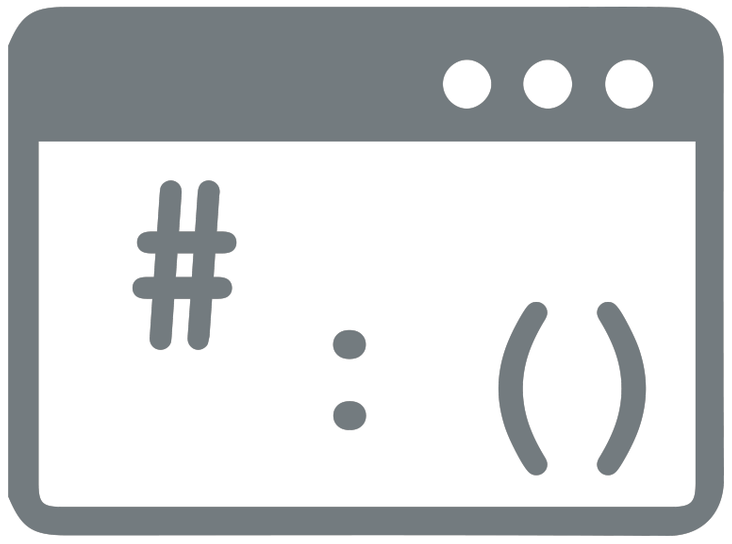
Let's describe the gameplay logic using a flow chart

Guessing Game Logic

- Let's create a flowchart to help us think through the state of the game at various stages

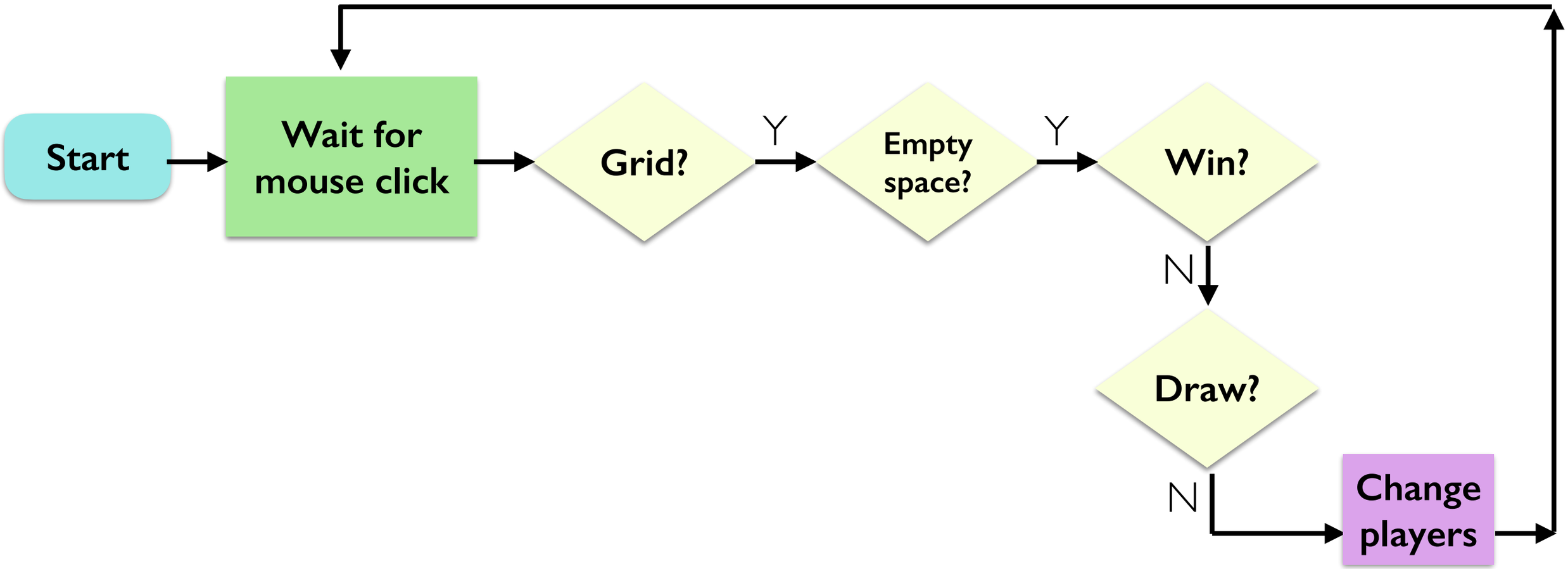


Guessing Game: Code in Class



Finally…TTT Game Logic

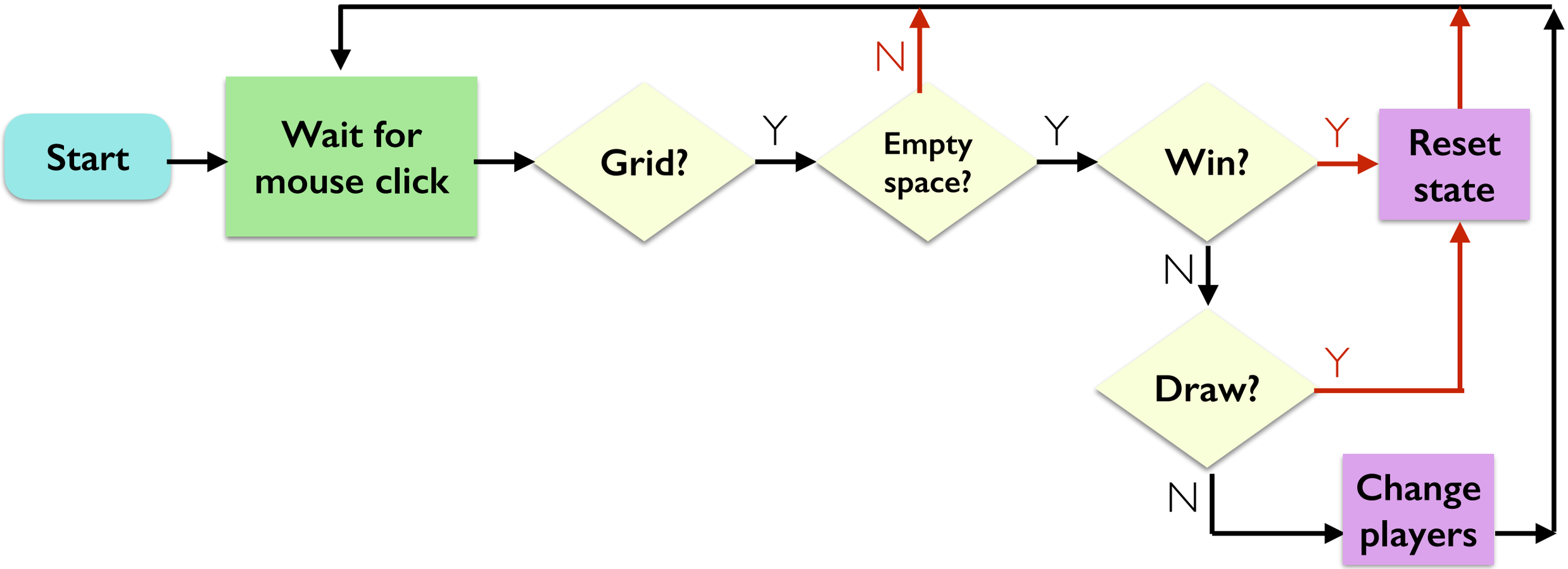
- Let's create a TTT flowchart to help us think through the state of the game at various stages



Let's think about the "common" case: a valid move in the middle of the game

Finally…TTT Game Logic

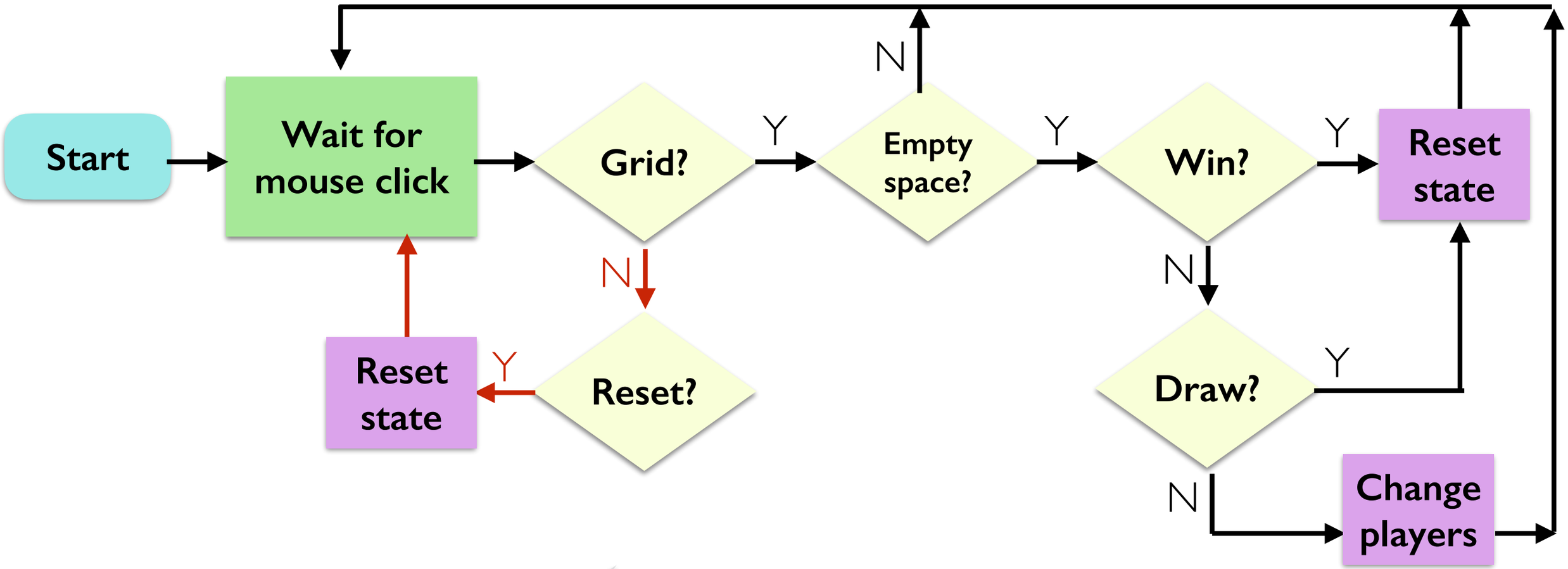
- Let's create a TTT flowchart to help us think through the state of the game at various stages



Now let's consider the case of a win, draw, or invalid move

Finally…TTT Game Logic

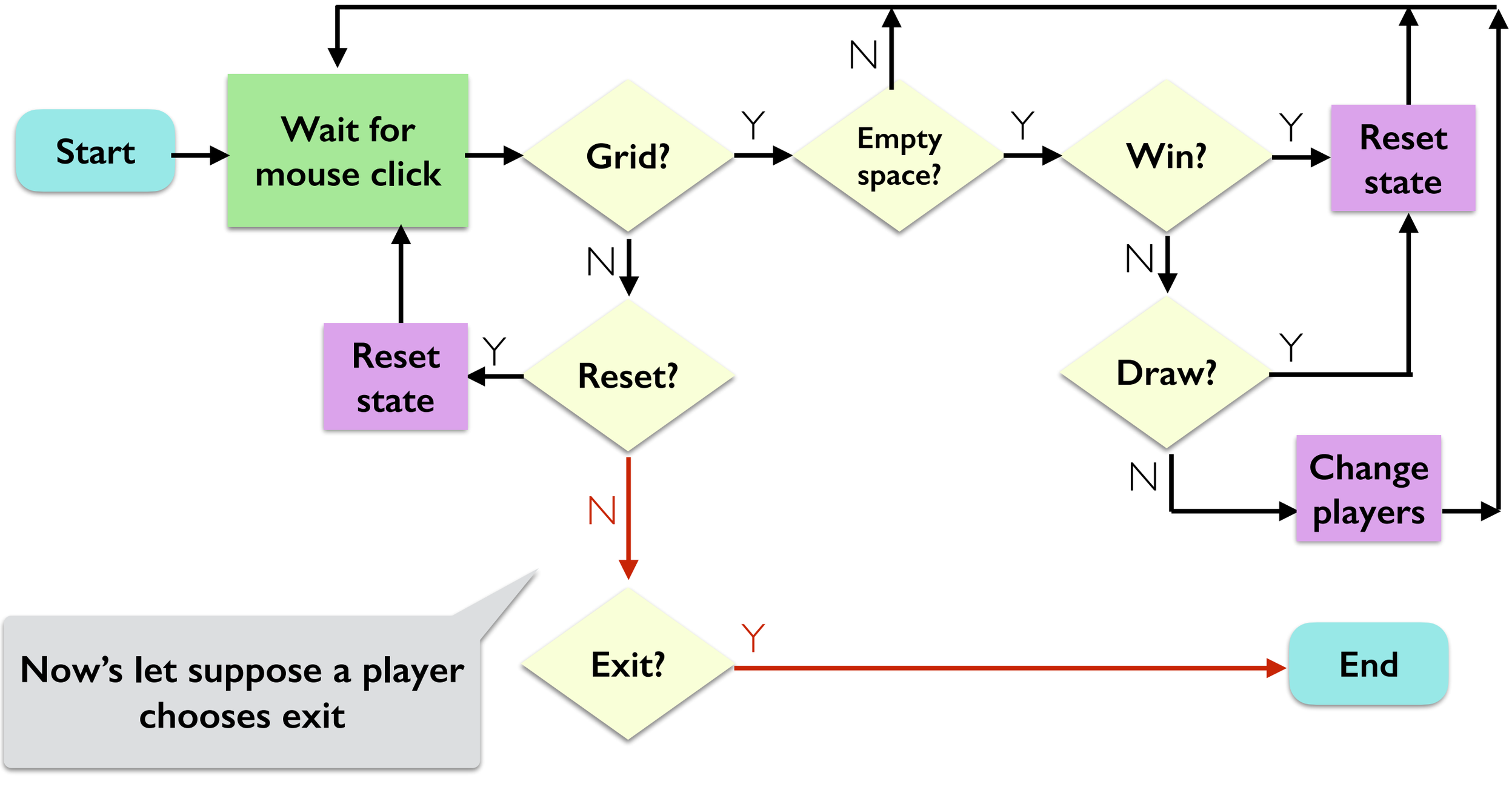
- Let's create a TTT flowchart to help us think through the state of the game at various stages



Now's let suppose a player chooses reset

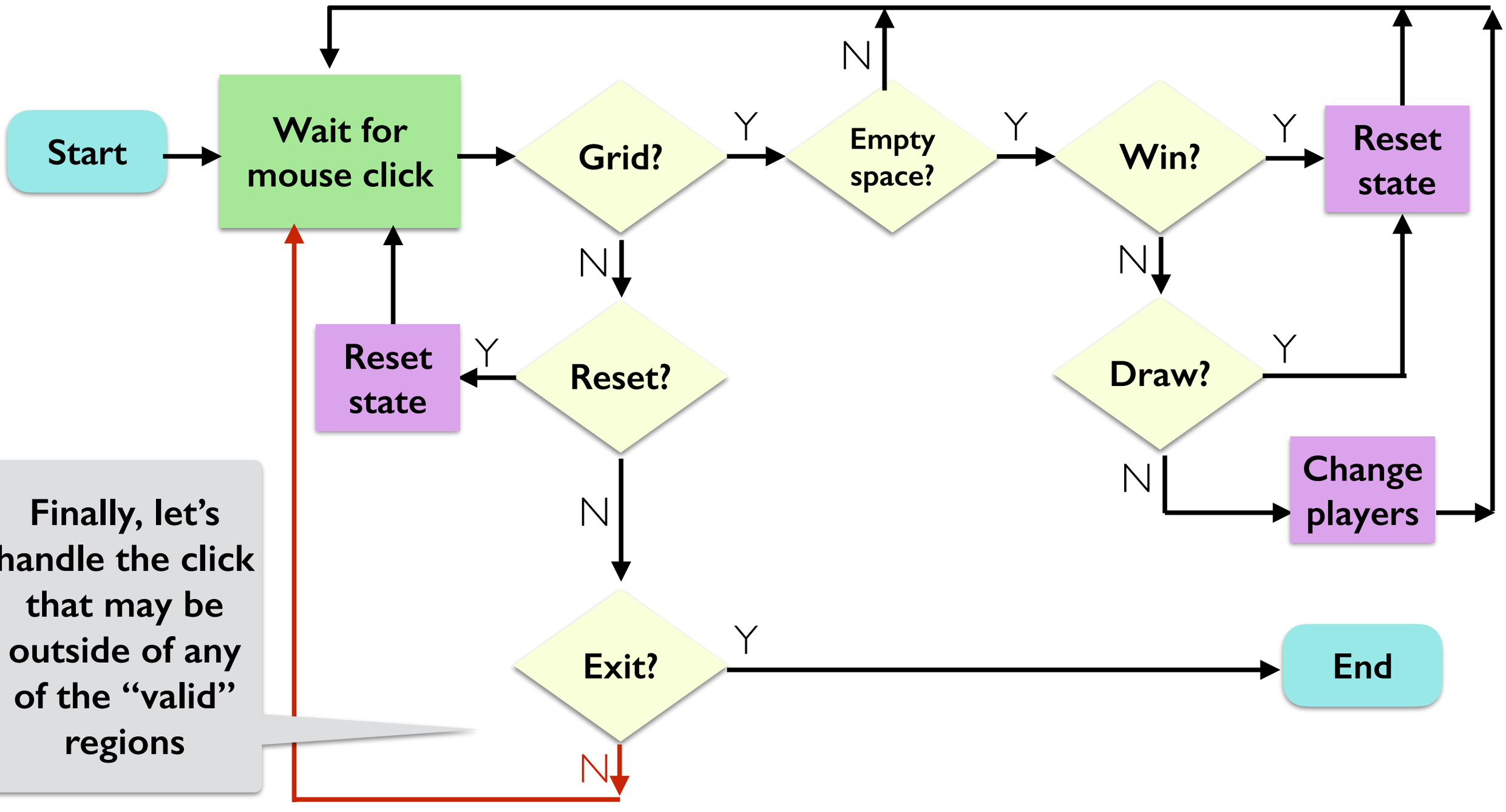
Finally…TTT Game Logic

- Let's create a TTT flowchart to help us think through the state of the game at various stages



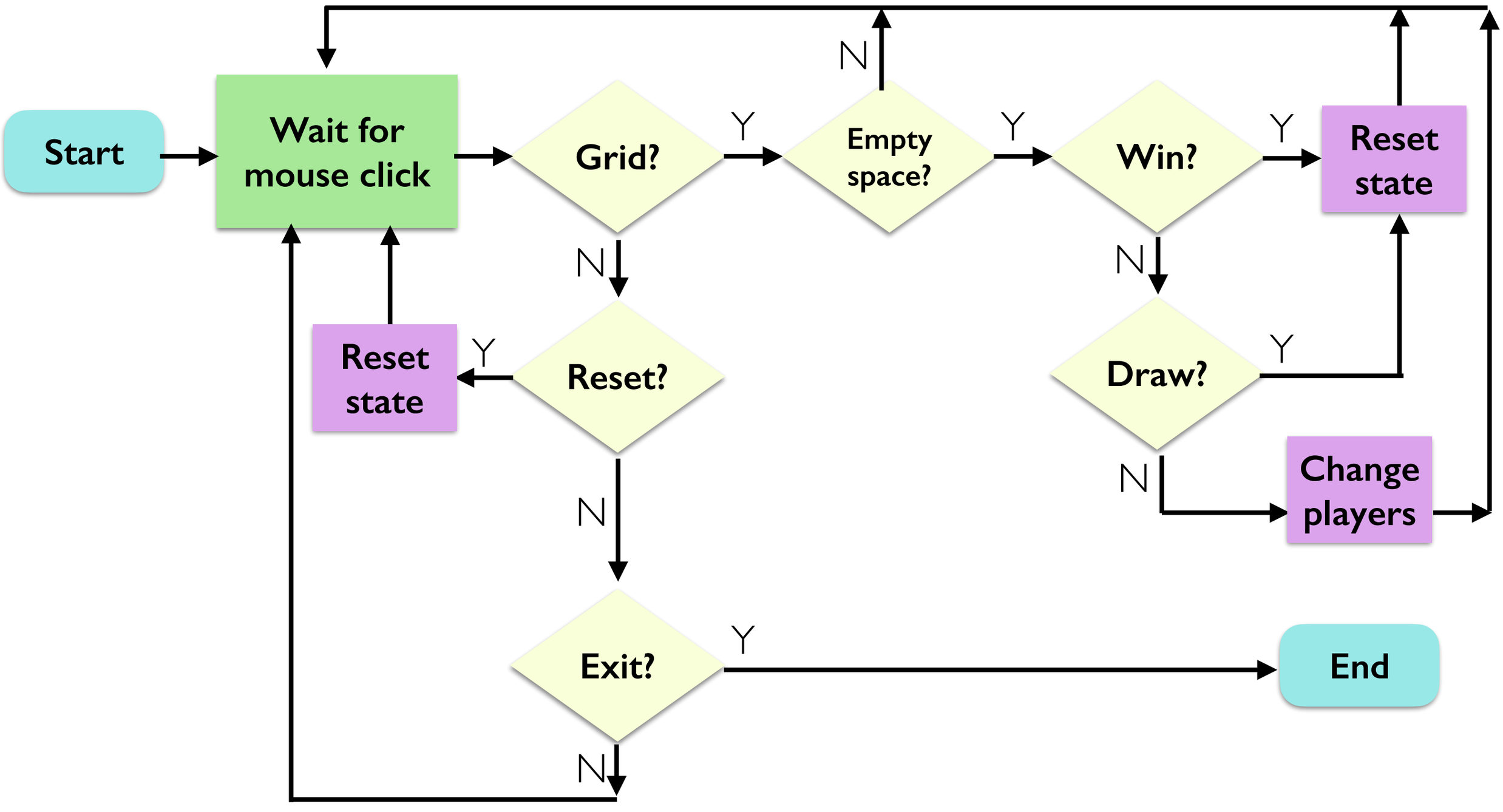
Finally…TTT Game Logic

- Let's create a TTT flowchart to help us think through the state of the game at various stages



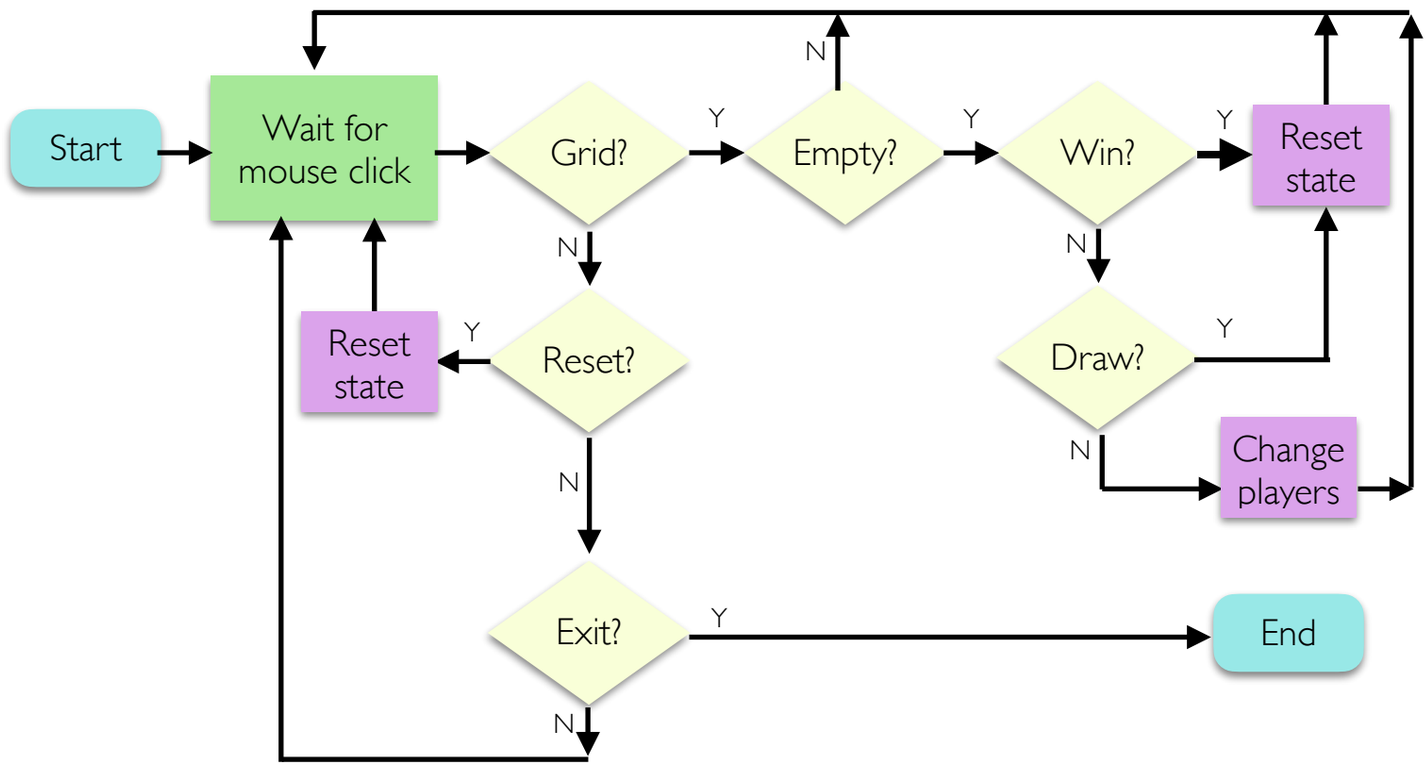
Finally…TTT Game Logic

- Let's create a TTT flowchart to help us think through the state of the game at various stages



Translating our Logic to Code

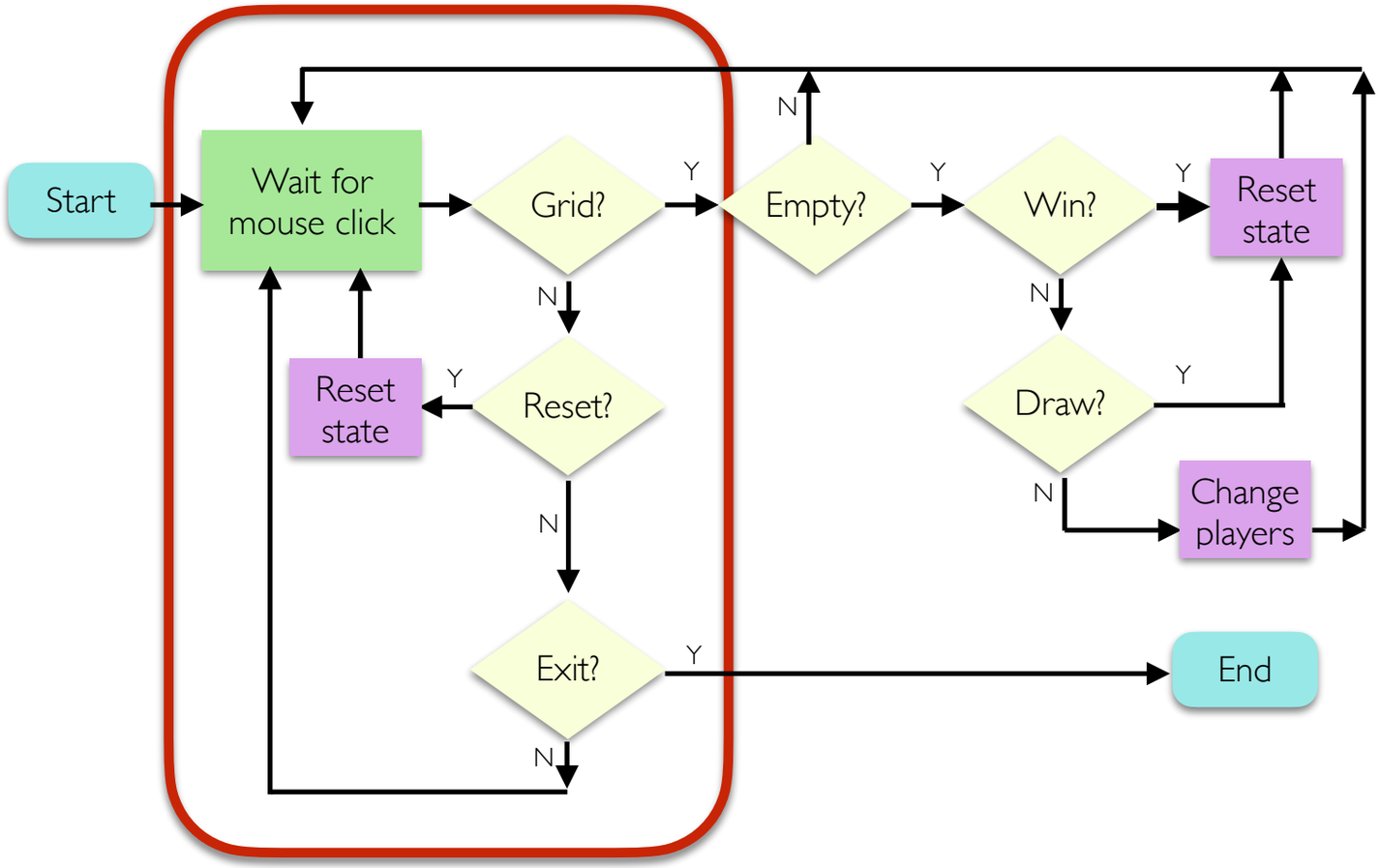
- Let's think about `__init__`:
  - What do we need?
    - a `board`, player, and maybe `num_moves` (to detect draws easily)



Translating our Logic to Code

- Now let's write a method for handling a single mouse click (point)
- The game continues (waits for more clicks) if this method returns True
- If this method returns False, game ends

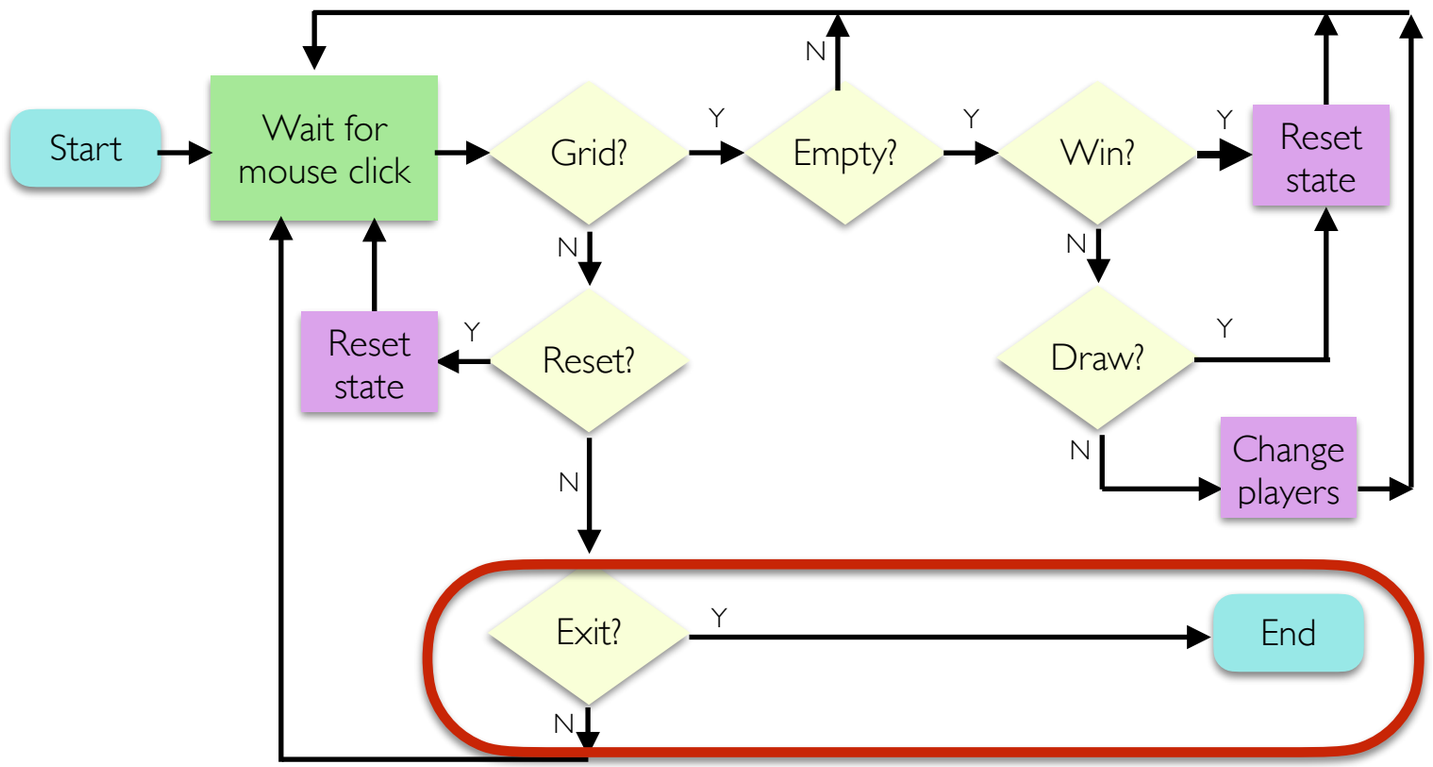
```
def do_one_click(self, point):

  # step 1: check for exit button
  if self._board.in_exit(point):
    # TODO

  # step 2: check for reset button
  elif self._board.in_reset(point):
    # TODO


  # step 3: check if click on the grid
  elif self._board.in_grid(point):
    # TODO


  # keep going!
  return True
```



Translating our Logic to Code

- Let's handle the "exit" button first (since it's the easiest)

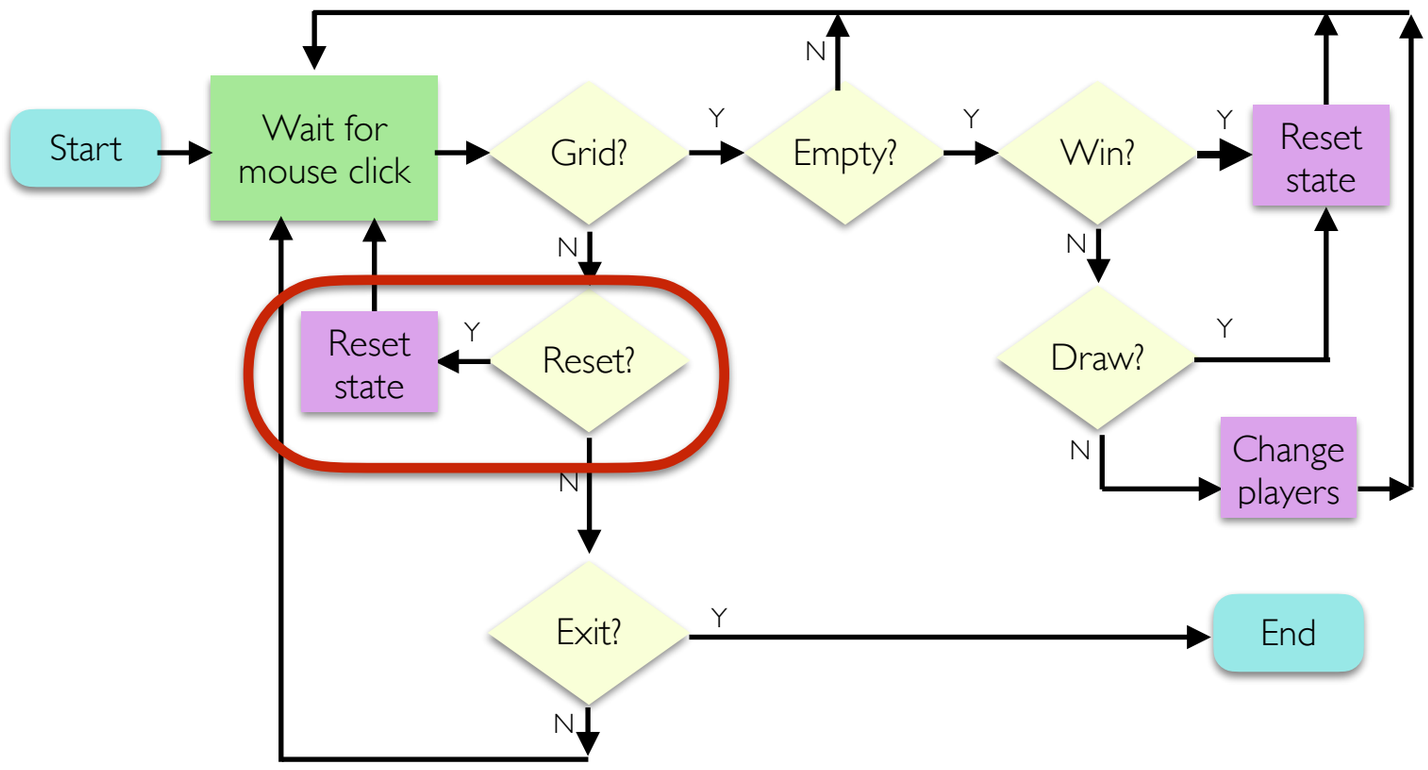
```
if self._board.in_exit(point):
    print("Exiting...")
    # game over
    return False
```



Translating our Logic to Code

- Now let's handle reset

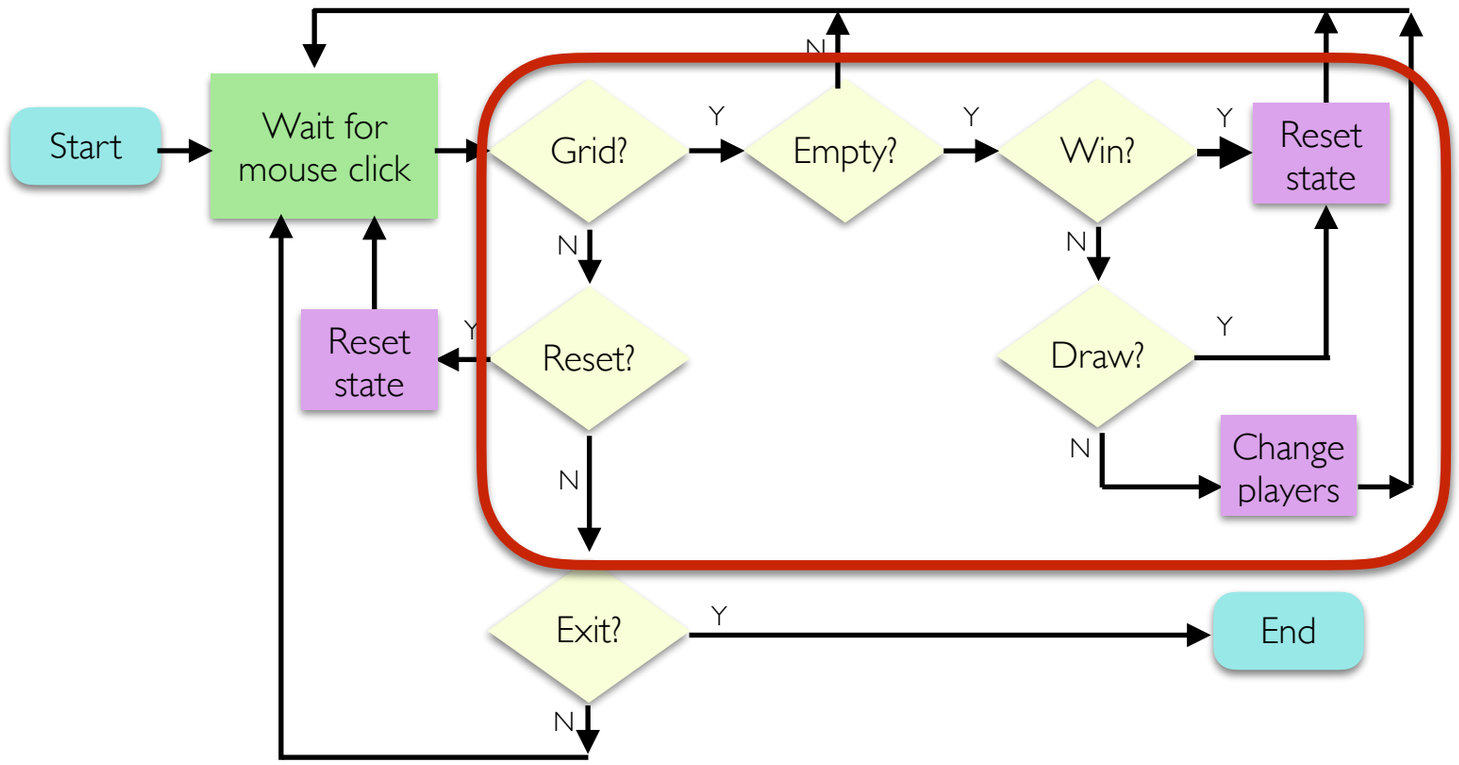
```
elif self._board.in_reset(point):
    print("Reset button clicked")
    self._board.reset()
    self._board.set_string_to_upper_text("")
    self._num_moves = 0
    self._player = "X"
```



Translating our Logic to Code

- Finally, let's handle a "normal" move. Start by getting point and TTTCube

```
elif self._board.in_grid(point):

    # get the cube at the point the user clicked
    tcube = self._board.get_ttt_cube_at_point(point)
```



Translating our Logic to Code

- The rest of our code checks for a valid move, a win, a draw, and updates state accordingly
- At the end, if the move was valid, we swap players

```
elif self._board.in_grid(point):

    # get the cube at the point the user clicked
    tcube = self._board.get_ttt_cube_at_point(point)

    # make sure this square is vacant
    if tcube.get_letter() == "":
        tcube.set_letter(self._player)
        tcube.place_cube(self._board)

        # valid move, so increment num_moves
        self._num_moves += 1

        # check for win or draw
        win_flag = self._board.check_for_win(self._player)
        if win_flag:
            self._board.set_string_to_upper_text(self._player + " WINS!")
        elif self._num_moves == self._board.get_rows()
                                            * self._board.get_cols():
            self._board.set_string_to_upper_text("DRAW!")
        # not a win or draw, swap players
        else:
            # toggle player!
            self._player = "O" if self._player == "X" else "X"

# keep going!
return True
```

TTT Summary

- Basic strategy
  - `Board`: start general, don't think about game specific details
  - `TTTBoard`: extend generic board with TTT specific features
    - Inherit everything, update attributes/methods as needed
  - `TTTCube` isolate functionality of a single TTT cube on board
    - Think about what features are necessary/helpful in other classes
  - `TTTGame`: think through logic conceptually before writing any code
    - Translate logic into code carefully, testing along the way

Boggle Strategies

- At a high level, Tic Tac Toe and Boggle have a lot in common, but the game state of Boggle is more complicated
- In Lab 9 you should follow a similar strategy to what we did with TTT
- ***Don't forget the bigger picture as you implement individual methods***
- Think holistically about how the objects/classes work together
- Isolate functionality and test often (use `__str__` to print values as needed)
- **Discuss logic with partner/instructor before writing any code**
- Worry about common cases first, but don't forget the "edge" cases
- Come see instructors/TAs for clarification

GOOD LUCK and HAVE FUN!

The end!