Tic-Tac-Toe

You will create a program to play tic-tac-toe. While doing this you should have the following in your program:

- A function
- If/Else statement
- events
- World and character methods

To set up the board, you can use squares (in shapes). You will use these since there are some issues in creating a game with the traditional “X” and “O” values. Instead you will use red and blue values for the two players. Your program will need to do the following:

- Change the color of a square when it is clicked. You need to make sure it has not been played before and no one has won. If this happens, tell the player.
- After you do a move, check if the player won the game. If someone wins, then put a notice on the screen (see 3D text).
- You need to make sure you automatically alternate the color of who is playing.

Here are a few suggestions to help you solve this problem:

- You can create an invisible square that is set to the color of the current player. By changing it appropriately, you can tell who is supposed to play.
- You need to decide all the cases of how someone can win. Try to come up with a way to make it so the cases are related. What this means is you don’t need to repeat similar code for similar types of checks. By using parameters with a functions (or method), you can save some work.
- It may be difficult to get the squares and messages to look flat. Try running the method “turn to face” during setup so they are permanently looking at the camera.
- All the squares are the same. Once you get one the right size, etc., you can repeat it for the other 8 squares.

Break up into two groups and design solutions for the problem. Do a storyboard and specify what methods and functions you will have. When you finish, have the groups compare their designs to see if you can come up with one nice design.

The next step is to code it up and try to make it work. Try to work in smaller groups (pairs if possible) for this step. If you get stuck, ask help from others.

Before you start, you can see the solution I coded up to see what it is likely to look like.
Exercise 2: Boolean conditions

Play a game using the decks of yellow and blue cards. Each blue card has an English sentence that is either true or false. Each yellow card has one of three logical connectors: not, and, or. The game is played as follows:

- Each player starts with 6 cards, 3 yellow and 3 blue.
- When it’s your turn, if you can make a logical formula that evaluates to true using at least 4 of your cards, put down the formula and draw new cards. Otherwise, trade in one of your cards for a card of the same color.
- The first person to put down at least 12 cards wins.

You can group cards to show which one applies first. Place a space to indicate groups. By default, logical operators are done from left to right.

If you don’t know if a statement is true or false, you can ask someone to find out.
Creating a clock

In this programming exercise you will create something like a clock. It will display all the times from 4:00 to 6:00 in five minute increments. Thus, it will show:

4:00 then 4:05, then 4:10, …, 4:55, 5:00, 5:05, 5:55, 6:00

To do this think about:

• loops with the complex version
• 3D text to show the clock
  o look dragging in text to change its display
  o look at the World string functions. They allow you to change a number to a string and join strings together

Note that the solution will likely have times like: 5.0:10.0. Alice shows one digit past the decimal point by default.

Work in pairs to design a solution and then code it up.