SKETCH & GUESS

In this unit you will make a drawing game where one person draws and others guess what is being drawn

THE APP CHALLENGE

Part 1: Create an app with multiple screens

❏ Screen1 will have 2 buttons - one for the Sketcher and one for the Guesser. An example of what Screen1 should look like is on the page 2.

❏ A second screen, SketcherScreen will contain a Canvas for the Sketcher to draw on. An example of what SketcherScreen should look like is on page 3.

❏ If the user clicks on the "I want to draw" button, SketcherScreen should open.

❏ If the user clicks on the "Back" button in the SketcherScreen, it should close that screen to return to Screen1.

Part 2: Add a drawing feature to SketcherScreen

❏ When the "New Picture" button is clicked, a random item is chosen from a list, and displayed in **DrawingLabel**, to tell the user what to draw. The list should contain at least 12 items to make the game interesting.

❏ When the user drags their finger on the **Canvas**, a black line is drawn where their finger drags. Complete the diagrams on pages 4,5 before coding the drawing.

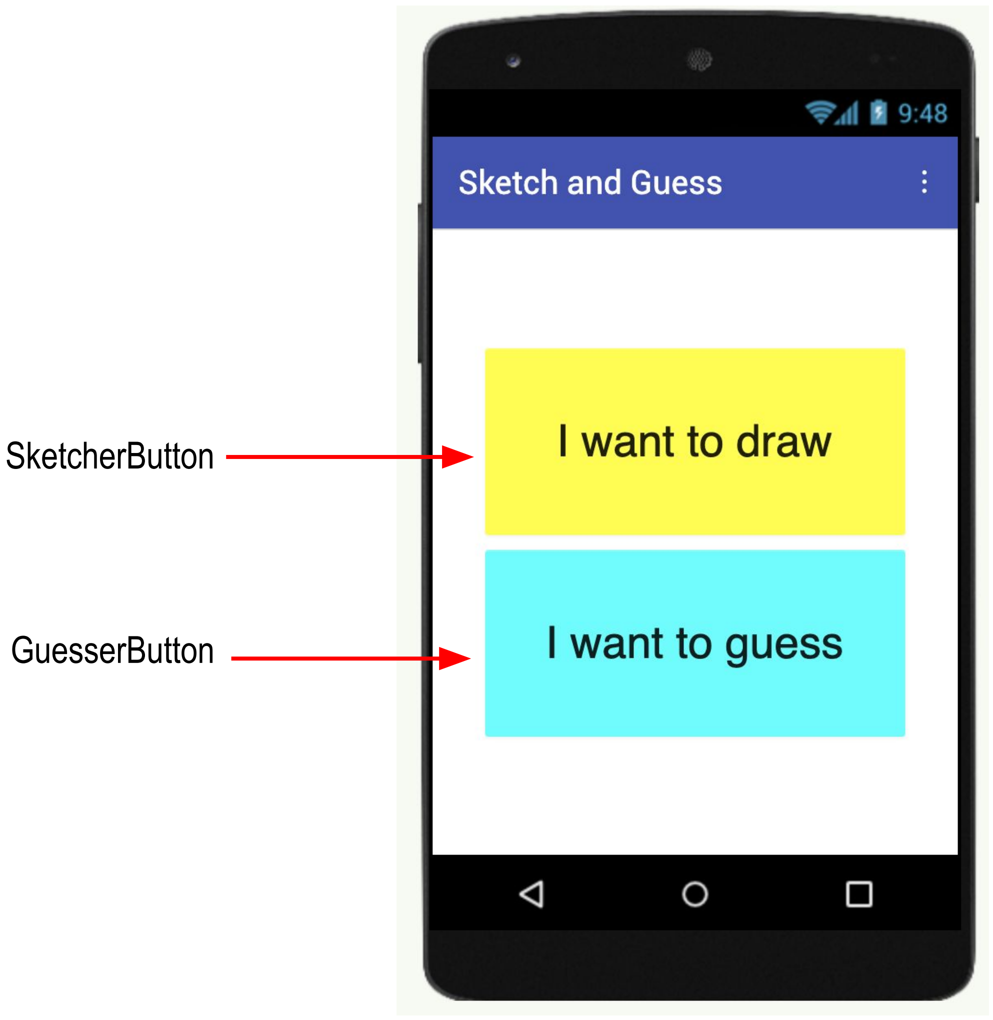
❏ When the user clicks the **ClearButton**, the **Canvas** clears.

When adding many new features to a game, it is always a good idea to break it down into parts. Get one part working before moving on.

SKETCH & GUESS

This is an example layout for Screen1.

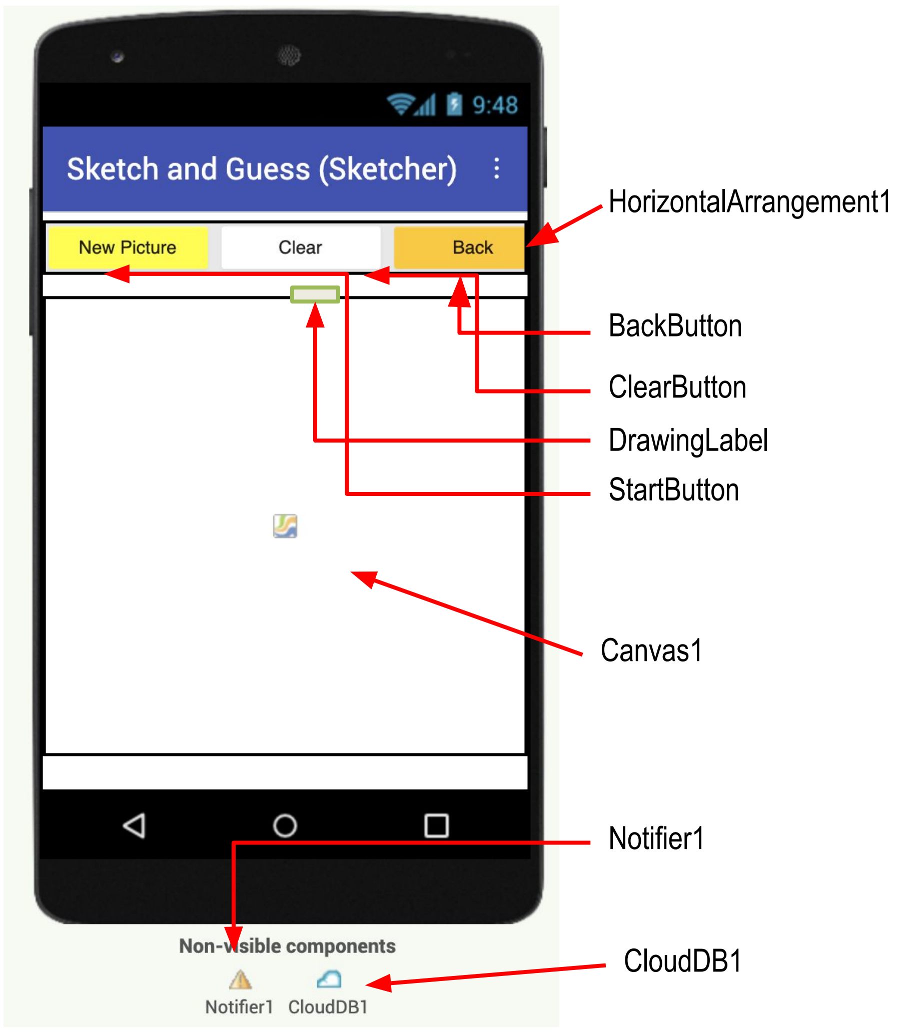
SCREEN1



SKETCHERSCREEN

When the user chooses to be the Sketcher, the app will open the SketcherScreen.

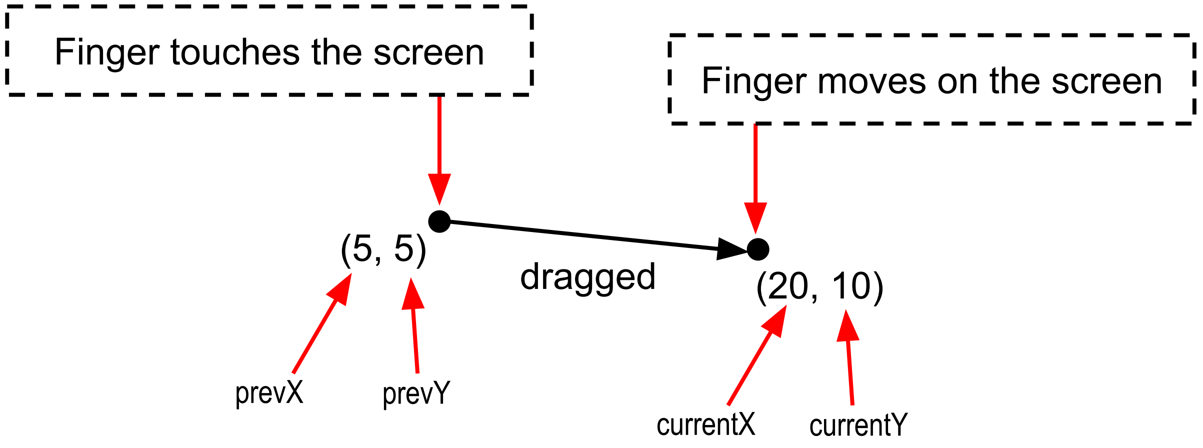
Below is a possible layout for the SketcherScreen.



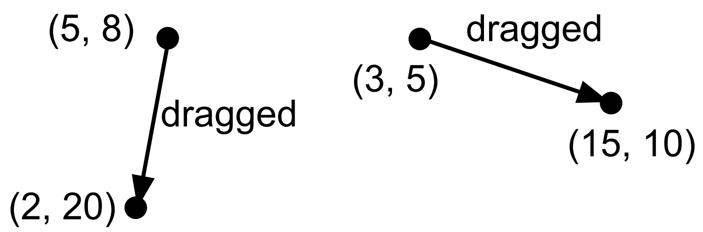
HOW TO DRAW A LINE IN THE APP

To draw something on the Canvas, you need to use the **when Canvas.Dragged** block and the **call Canvas1.DrawLine** blocks.

The diagram below shows how to use coordinates to draw a line in the app. A line is drawn by joining two points. Using **Canvas1.DrawLine**, you need to specify the position of the start point (x1, y1) and the position of the end point (x2, y2).



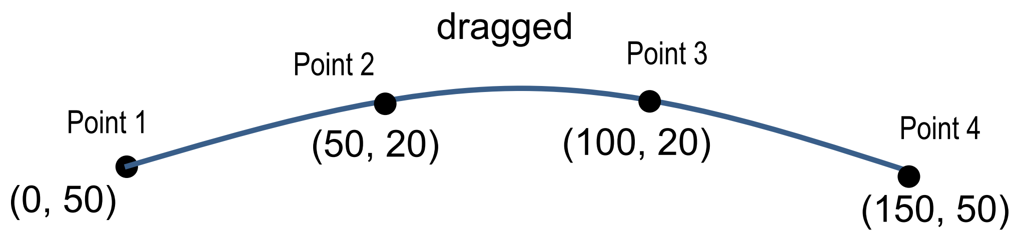
If you want to draw a line in the app, you need to use **prevX**, **prevY** and **currentX**, **currentY**. Below please work with your partner to fill in the blanks.



prevX		
prevY		
currentX		
currentY		

HOW TO DRAW A CURVE IN THE APP

A line is formed by connecting many dots. Each of the dots is represented by its coordinate (x,y).



If you want to draw a curve in the app, you need to draw many lines, each with its own **prevX**, **prevY** and **currentX**, **currentY**. Below please work with your partner to fill in the blanks.

	segment 1 (Point 1-Point 2)	segment 2 (Point 2-Point 3)	segment 3 (Point 3-Point 4)
prevX			
prevY			
currentX			
currentY			

THE APP CHALLENGE (continued)

Part 3: Send drawing data to CloudDB so you can see drawings across devices

❏ Add the GuesserScreen, where the Guesser can see what is being drawn by the Sketcher. See the following page for a sample GuesserScreen layout.

❏ When the user clicks on the “I want to guess” button in Screen1, open the GuesserScreen.

❏ When the user clicks on the “Back” button in the GuesserScreen, close that screen to return to Screen1.

❏ In SketcherScreen, when the user drags their finger across the screen, in addition to drawing, the drawing information (**prevX, prevY, currentX, currentY**) is stored in **CloudDB**. HINT: Store in a list so it can be stored using a single tag.

❏ When a user clicks the **ClearButton** in SketcherScreen, communicate that action to **CloudDB** as well.

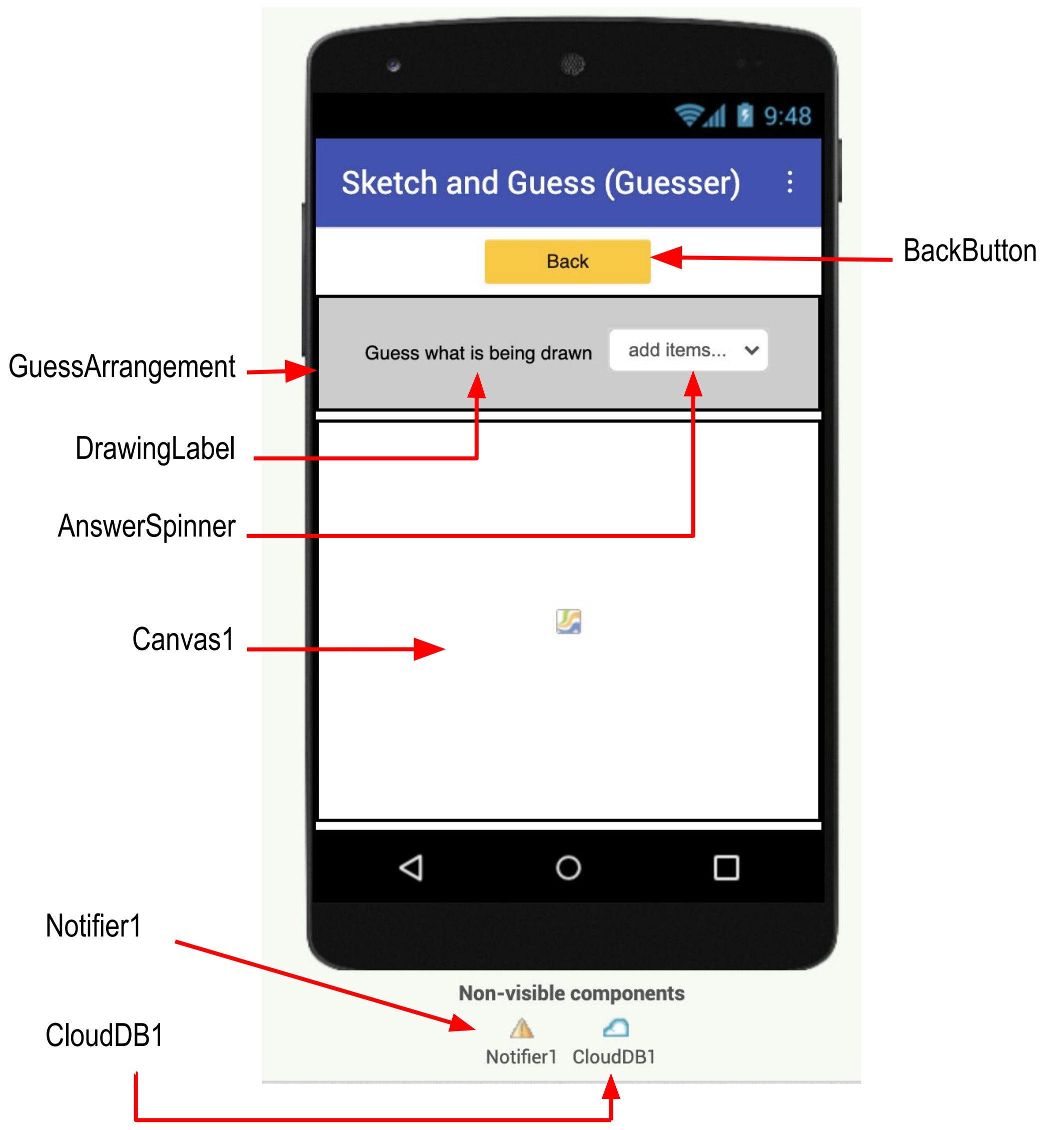
❏ In GuesserScreen, when new drawing data is received from **CloudDB**, use the drawing data received to draw on the **Canvas**.

❏ In GuesserScreen, when information about the clearing of the Canvas is received from **CloudDB**, clear the **Canvas**.

GUESSERSCREEN

Part 3: Add GuesserScreen

GuesserScreen should look something like the image below.



THE APP CHALLENGE (continued)

Part 4: Add guessing functionality to the app

❑ In SketcherScreen, when the Sketcher starts drawing, store the **currentDrawing** information in **CloudDB**.

❑ When a Guesser receives **currentDrawing** information from **CloudDB**, store that in a variable for later answer checking.

❑ In GuesserScreen, display the list of possible drawing objects in **AnswerSpinner**, which should match the list in SketcherScreen from which a random item to draw is displayed.

❑ When a Guesser makes a guess by selecting from the **AnswerSpinner**, notify them as to whether they are correct or incorrect.

❑ When the Guesser chooses a correct guess, store that information in CloudDB.

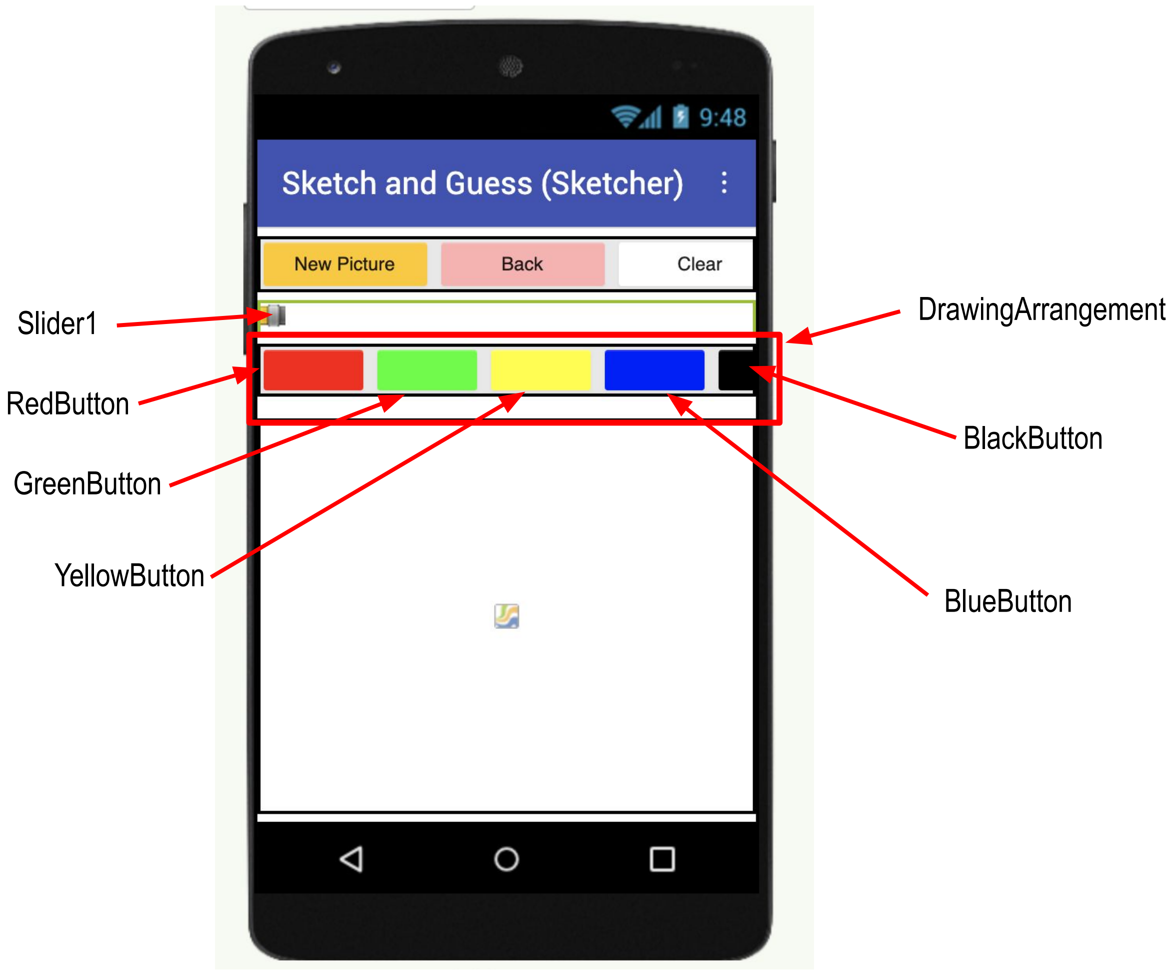
❑ When the Sketcher and Guesser receives information from CloudDB that the Guesser has guessed correctly, display that message with the Notifier.

THE APP CHALLENGE (continued)

Final Challenge

- ❏ Add color buttons to the SketcherScreen so the Sketcher can change the color of the pen.
- ❏ Add a **Slider** component to SketcherScreen, so the Sketcher can change the LineWidth of the pen when drawing. Make the minimum width 1 and the maximum width 10.
- ❏ Store the color and pen line width in CloudDB from SketcherScreen along with the start and end points of the line.
- ❏ In GuesserScreen, when the drawing data is received from CloudDB, use the additional color and line width information to draw the line using the correct color and line width.

Below is an example layout for the added Slider and color Buttons.



Choose Ways to Extend Your App

Here are a few features you could add if you want to expand your app

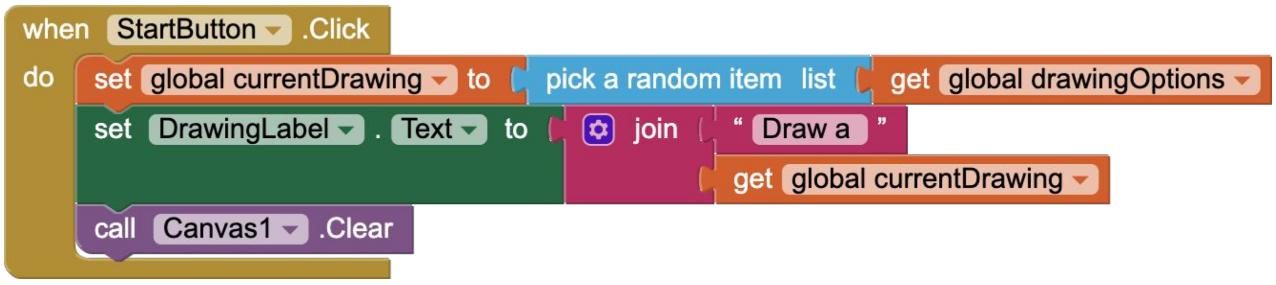
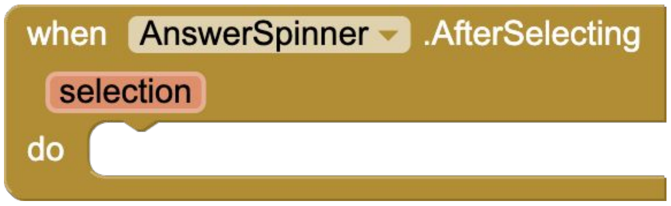
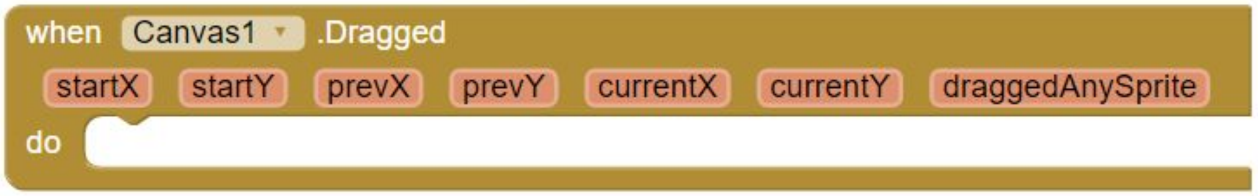
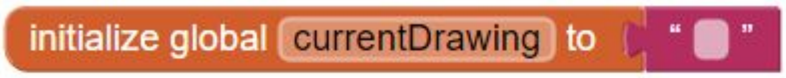
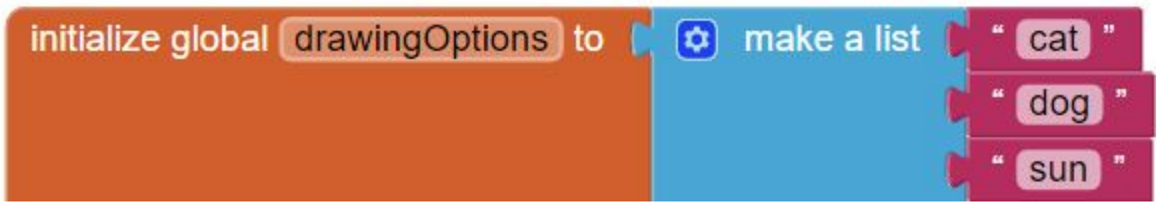
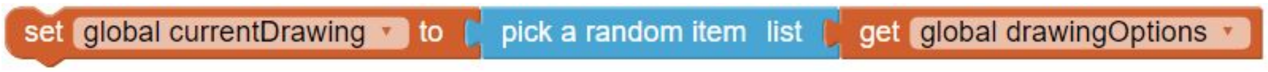
Add TextToSpeech to speak what is to be drawn

Send users back to Screen1 on a correct guess to restart the game

Keep score! Each player can keep track of their correct guesses!

What other ideas do you have?

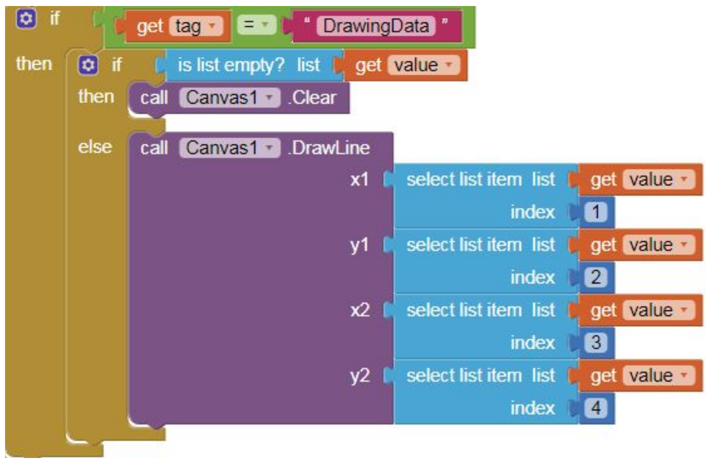
COMPUTATIONAL THINKING CONCEPTS

Sketch And Guess
1. Sequences  when StartButton .Click do set global currentDrawing to pick a random item list get global drawingOptions set DrawingLabel . Text to join " Draw a " get global currentDrawing call Canvas1 .Clear
2. Events  when AnswerSpinner .AfterSelecting selection do  when Canvas1 .Dragged startX startY prevX prevY currentX currentY draggedAnySprite do
3. Naming/Variables  initialize global currentDrawing to " "
4. Manipulation of data and elementary data structures  initialize global drawingOptions to make a list " cat " " dog " " sun "  set global currentDrawing to pick a random item list get global drawingOptions

COMPUTATIONAL THINKING CONCEPTS

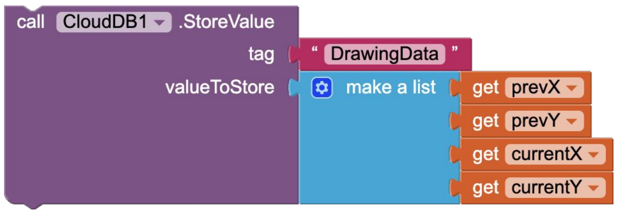
Sketch And Guess Part 3

2. Conditionals

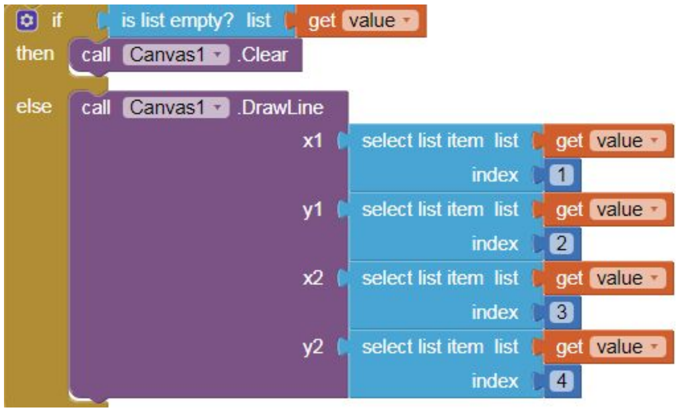


```
if get tag = "DrawingData"
then
    if is list empty? list get value
    then
        call Canvas1.Clear
    else
        call Canvas1.DrawLine
            x1  select list item list get value
                                 index 1
            y1  select list item list get value
                                 index 2
            x2  select list item list get value
                                 index 3
            y2  select list item list get value
                                 index 4
```

4. Manipulation of data and elementary data structures



```
call CloudDB1.StoreValue
    tag "DrawingData"
    valueToStore make a list get prevX
                             get prevY
                             get currentX
                             get currentY
```



```
if is list empty? list get value
then
    call Canvas1.Clear
else
    call Canvas1.DrawLine
        x1  select list item list get value
                             index 1
        y1  select list item list get value
                             index 2
        x2  select list item list get value
                             index 3
        y2  select list item list get value
                             index 4
```