Make This

Chapter 10 – Making a Notepad

Having an app that functions as a notepad is very useful either for writing notes or just doodling. The Canvas component of App Inventor allows you to add the ability to draw on the screen. In this tutorial, you'll construct an app that allows you to draw lines, draw dots, erase the screen, and even switch your drawing colors.

This tutorial teaches the following skills:

- **Using the Canvas Component**
- **Using Global Variables**

Note: Before attempting this exercise, complete the Chapter 2 and 3 exercises to familiarize yourself with the App Inventor interface, getting your Android device connected, and starting a new project.

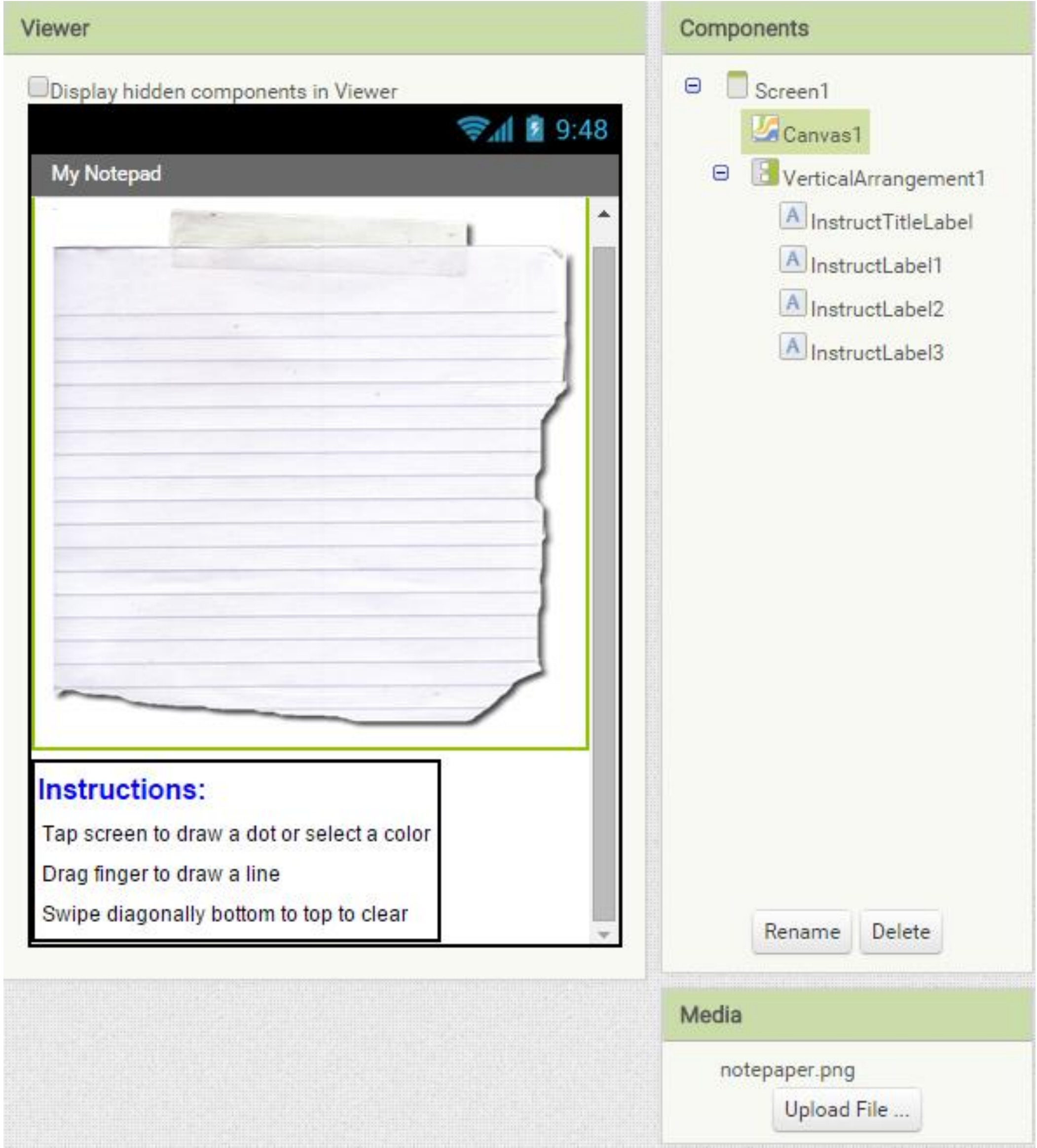
Building the Notepad App

1. Navigate to http://appinventor.mit.edu/explore/. If necessary, sign in with your Google Account.
2. Start a new project named *NotepadApp*. Change the **Title** of **Screen1** to *My Notepad*.
3. Connect App Inventor to your Android device.

Part 1 – Constructing the Interface

The interface allows the user to tap the screen to draw a dot. Tapping the screen again will change the color. The user can drag their finger across the screen to draw a line. To clear the screen, the user makes a quick swipe diagonally (from the bottom to the top).

Your layout for your interface should look like this:



Instructions:

Tap screen to draw a dot or select a color

Drag finger to draw a line

Swipe diagonally bottom to top to clear

Build the interface shown above by dragging out the components shown in the following table:

Component	Palette Group	Component Name	Function
Canvas	Drawing and Animation	Canvas1	Provides a rectangular area on which the user can draw
VerticalArrangement	Layout	VerticalArrangement1	Area to line up the instruction labels
Label	User Interface	InstructTitleLabel	Contains instruction header
Label	User Interface	InstructLabel1	Contains first line of app instructions
Label	User Interface	InstructLabel2	Contains second line of app instructions
Label	User Interface	InstructLabel3	Contains third line of app

			instructions

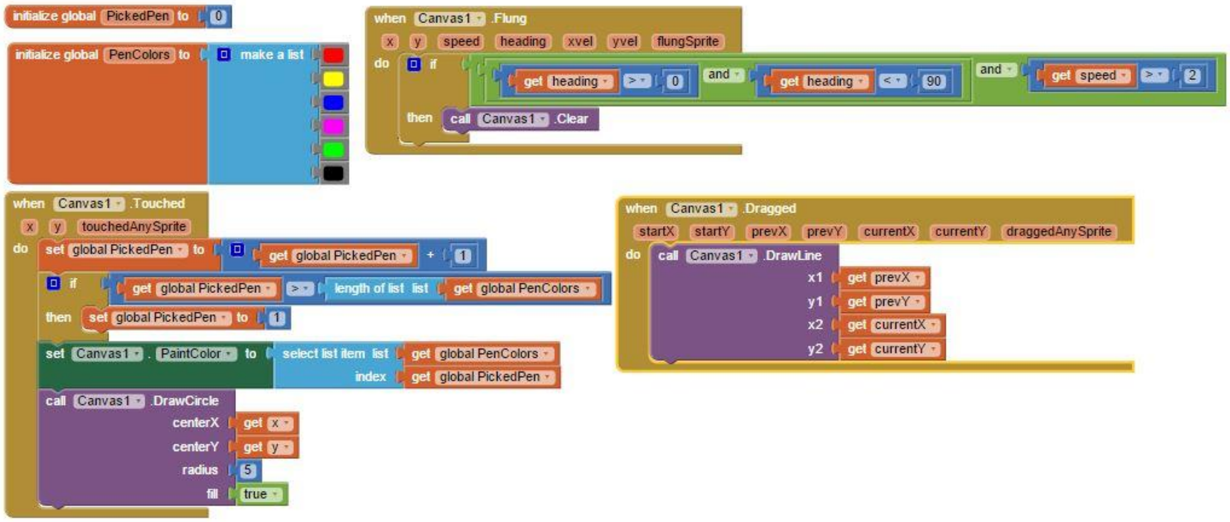
Set the properties of each component in the following ways:

- Change the **BackgroundImage** of **Canvas1** to *notepaper.png* (upload the file provided) and the **Height** to *320 pixels*. **Note:** You may need to change the Height to a different value depending upon the size of your device's display.
- Change the **Text** for **InstructTitleLabel** to *Instructions:*. Change the **FontSize** to *18* and the **TextColor** to *Blue*.
- Change the **Text** for **InstructLabel1** to *Tap screen to draw a dot or select a color*.
- Change the **Text** of **InstructLabel2** to *Drag finger to draw a line*.
- Change the **Text** of **InstructLabel3** to *Swipe diagonally bottom to top to clear*.

Part 2 – Program Functionality of the App

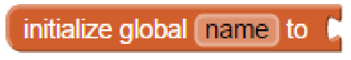
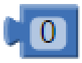
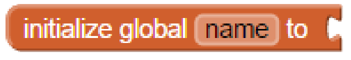
1. Change to **Blocks** view.

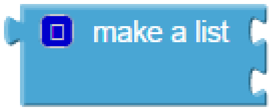
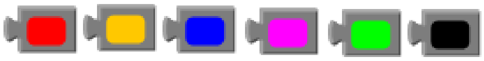
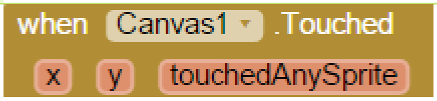
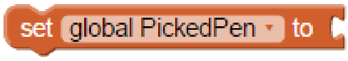
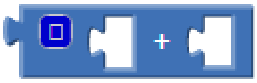
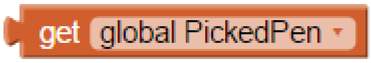
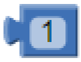
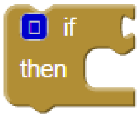
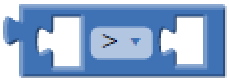
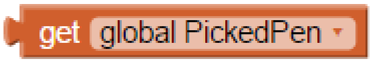
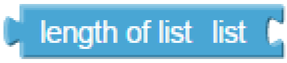
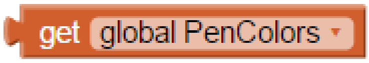
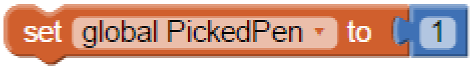
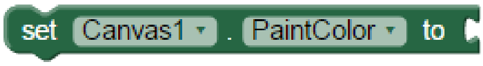
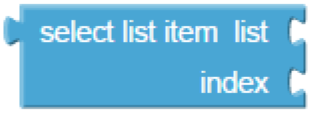
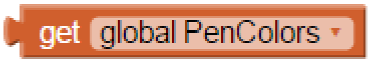
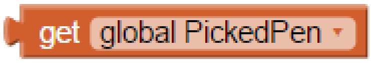
The blocks for this app are as follows:

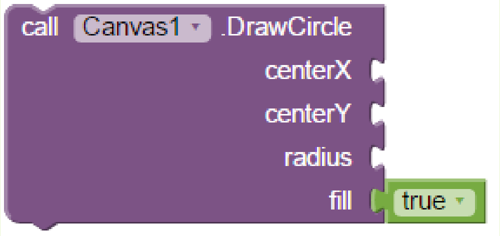
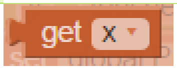
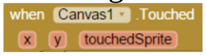
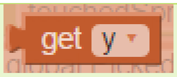
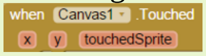
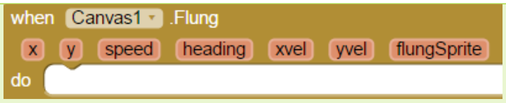
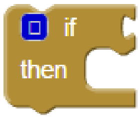
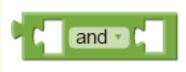
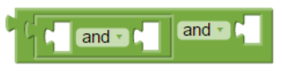
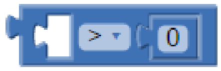
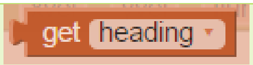
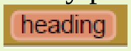
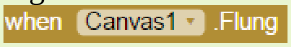


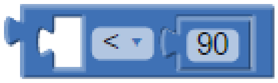
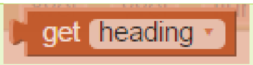
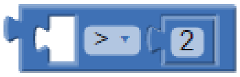
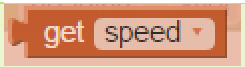
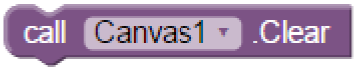
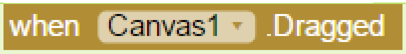
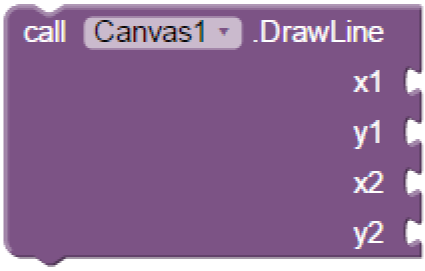
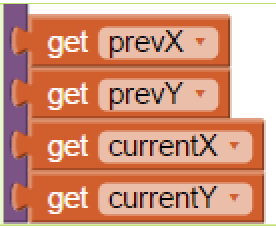
```
initialize global PickedPen to 0

initialize global PenColors to make a list 🟥
                                           🟨
                                           🟦
                                           🟪
                                           🟩
                                           ⬛

when Canvas1 .Flung
  x  y  speed  heading  xvel  yvel  flungSprite
do  if  get heading > 0  and  get heading < 90  and  get speed > 2
    then  call Canvas1 .Clear

when Canvas1 .Touched
  x  y  touchedAnySprite
do  set global PickedPen to  get global PickedPen + 1
    if  get global PickedPen > length of list list  get global PenColors
    then  set global PickedPen to 1
    set Canvas1 . PaintColor to  select list item  list   get global PenColors
                                                   index  get global PickedPen
    call Canvas1 .DrawCircle
                     centerX  get x
                     centerY  get y
                     radius   5
                     fill     true

when Canvas1 .Dragged
  startX  startY  prevX  prevY  currentX  currentY  draggedAnySprite
do  call Canvas1 .DrawLine
                  x1  get prevX
                  y1  get prevY
                  x2  get currentX
                  y2  get currentY
```

2. Now drag out the blocks (as indicated in the table below) and arrange them as shown above.

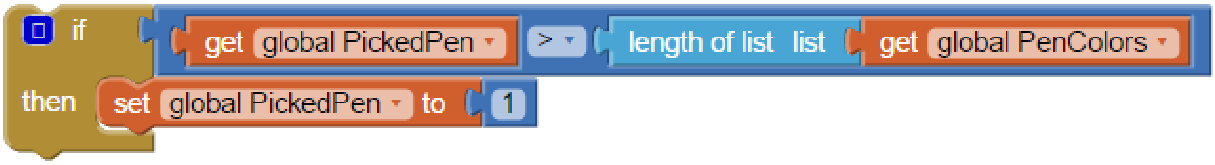
Block	Drawer	Function
 initialize global name to	Variables – Change name to PickedPen	Global variable for value of pen color (index) in the list
 0	Math	Initial value of PickedPen variable
 initialize global name to	Variables – Change name to PenColors	Global variable for color of pen

	Lists (use the mutator button □ to add more items to the list)	Creates a list with any number of items
	Colors	Color blocks used to define the pen colors in the list
	Canvas1	If canvas is touched and the user immediately lifts their finger, this block returns the (x,y) coordinates for the spot touched.
	Variables	Sets the value of the PickedPen variable
	Math	Returns the sum of two values. Used to increment the value of the PickedPen variable.
	Variables	Returns the current value of the PickedPen variable
	Math	Defines a value of one.
	Control	If a value is true, then execute some blocks
	Math	Returns true is the first value is greater than the second value
	Variables	Returns the current value of the PickedPen variable
	Lists	Counts the number of items in the specified list
	Variables	Returns the value of the PenColors variable (which is a list)
	Variables, Math	Sets the value of the PickedPen variable to one
	Canvas1	Sets the drawing color within the Canvas
	Lists	Returns an item in a list based on its position (index) in the list
	Variables	Returns the value of the PenColors variable (which is a list)
	Variables	Returns the current value of the PickedPen variable used

		to indicate the position (index) in the list
 call Canvas1 .DrawCircle centerX centerY radius fill true	Canvas1	Draws a circle at the given coordinates (x,y), with a defined radius (r), on the canvas.
	Obtained by pointing at the argument in the  block	x coordinate value of where the Canvas was touched
	Obtained by pointing at the argument in the  block	y coordinate value of where the Canvas was touched
	Math	Value of five - defines radius of circle to be drawn
 when Canvas1 .Flung x y speed heading xvel yvel flungSprite do	Canvas1	When a quick swipe (flung) gesture is made on the canvas, returns the (x,y) coordinates of the start of the swipe, the speed of the swipe and the heading (direction) of the swipe (from 0 to 360 degrees)
	Control	If a value is true, then execute some blocks
 x 2	Logic – Nest one inside the other  which means all three statements must be true	Returns value of true if all inputs are true
	Math	Checks if a value is greater than zero
	Obtained by pointing to the  argument in the  block	Returns the value of the direction of the swipe gesture in degrees

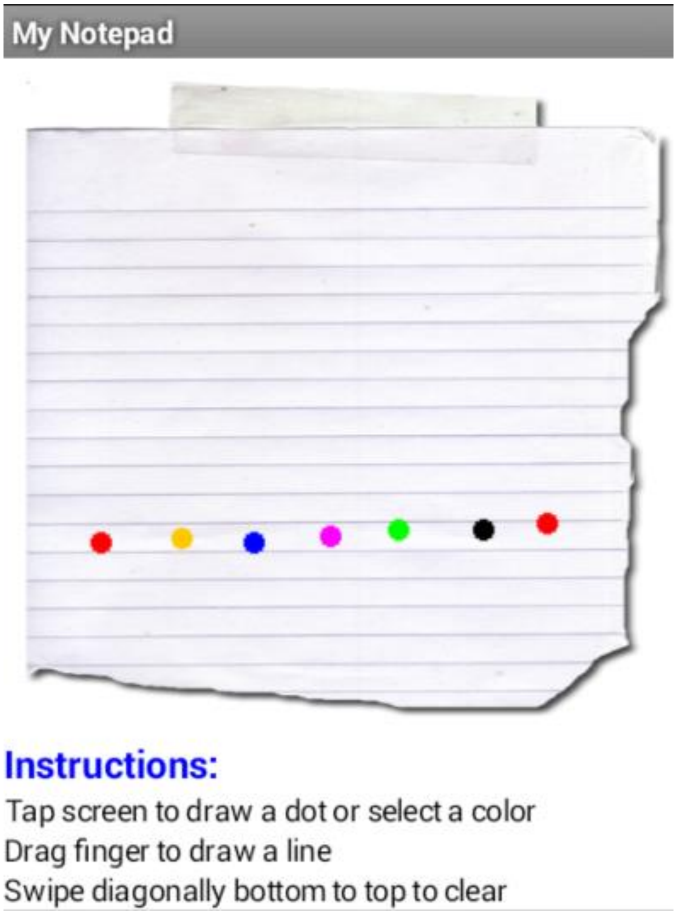
 < 90	Math	Checks if a value is less than ninety
 get heading	Obtained by pointing to the heading argument in the when Canvas1 .Flung block	Returns the value of the direction of the swipe gesture in pixels per millisecond
 > 2	Math	Checks if a value is greater than two
 get speed	Obtained by pointing to the speed argument in the when Canvas1 .Flung block Text	Returns the value of the speed of the swipe gesture in degrees
 call Canvas1 .Clear	Canvas1	Clears everything drawn on the Canvas
 when Canvas1 .Dragged	Canvas1	When a user touches the screen and drags their finger to another point, this block keeps track of the starting and ending coordinates
 call Canvas1 .DrawLine x1 y1 x2 y2	Canvas1	Draws a line on the Canvas between the given points. The line drawn follows the path of the user's finger.
 get prevX get prevY get currentX get currentY	Dragged out from the appropriate arguments in the when Canvas1 .Dragged block	Return the values of the starting and stopping points of the drag

Now test out your app touching and releasing the screen 6 times. This should draw six dots of different colors (starting with red) on the Canvas. A seventh touch will draw a red dot again as the



```
if   get global PickedPen > length of list list get global PenColors
then set global PickedPen to 1
```

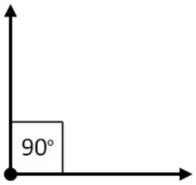
block sets the PickedPen variable back to 1 (the first position in the list) when the PickedPen variable exceeds the number of items in the list. After touching your screen 7 times, it should resemble this:



My Notepad

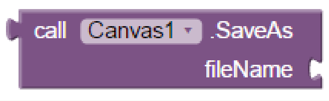
Instructions:

Tap screen to draw a dot or select a color
Drag finger to draw a line
Swipe diagonally bottom to top to clear

Now drag your finger to draw a circle on the screen. Notice the color of the circle will be the same as the last dot you put on the screen. Now swipe quickly from bottom left to the top right at an angle of between zero and 90 degrees. 

All drawing from your Canvas will be erased. **Note**: Make sure to swipe on the drawing canvas, not some other part of the screen.

Extensions to This Project

1. Touching the screen to change drawing colors is somewhat cumbersome. Modify your app (perhaps using a list picker component) that makes it easier for the user to choose a drawing color.
2. Modify the app so that the user can save what they have drawn to their device. Hint: Investigate the  block.

```
call Canvas1 .SaveAs
fileName
```

Resources

- AI2 Drawing and Animation Components: Canvas
- AI2 Variables
- User Guide for App Inventor 2
- Guide to Understanding Blocks

MIT App Inventor is a blocks-based programming tool that allows everyone, even novices, to start programming and build fully functional apps for Android devices. Google's Mark Friedman and MIT Professor Hal Abelson co-led the development of App Inventor while Hal was on sabbatical at Google. App Inventor runs as a Web service administered by staff at MIT's Center for Mobile Learning - a collaboration of MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL) and the MIT Media Lab. MIT App Inventor supports a worldwide community of nearly 3 million users representing 195 countries worldwide. App Inventor is an open-source tool that seeks to make both programming and app creation accessible to a wide range of audiences. App Inventor is the property of the Massachusetts Institute of Technology (MIT) and the work licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License. For more information on App Inventor, go to MIT App Inventor About Us page.