**HackBI VI App Inventor Teaching Script**

Downloads/Resources needed:

* App Inventor App on tablet or phone
* Laptop for accessing App Inventor Website

What is App Inventor?: is a visual programming environment that allows everyone to build fully functional apps for Android and iOS devices.

* Website link: <http://ai2.appinventor.mit.edu/> (Sign in with google)
* I will walk through of what App Inventor looks like:

1. Designer
2. Panels
3. Components
4. Properties
5. Screens
6. Builder
7. Talk about block coding
8. Built in blocks summary
9. Other Resources/ Videos with help
10. <https://appinventor.mit.edu/explore/content/basic.html>
11. <http://ai2.appinventor.mit.edu/reference/blocks/>
12. <https://appinventor.mit.edu/explore/ai2/tutorials>

Beginner/ demo app: Paint pot

Components:

* Canvas
* Buttons
* Horizontal arrangement
* Sound
* Camera

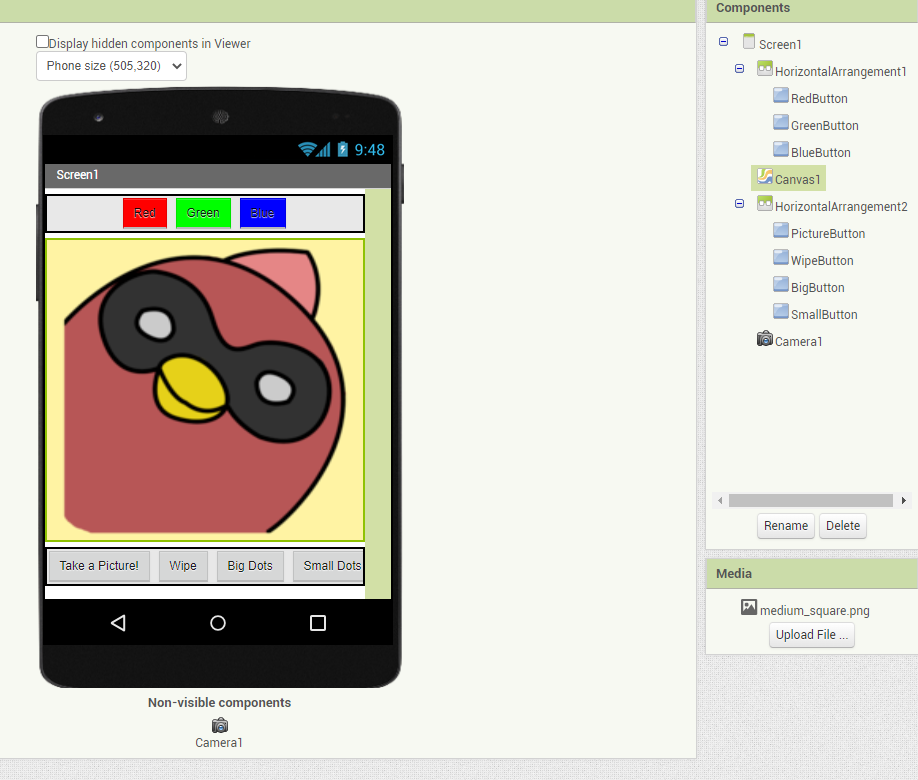
Step by step instructions to make paint pot:

1. Begin by making a new project by selecting the “projects” tab at the top left bar located near the MIT App Inventor logo in the top left. When selecting “projects” select “my projects”. Once you are on the “my projects” screen select the “start new project” underneath the MIT App Inventor logo. Name your project “PaintPot”.
2. Now you will see the main builder screen of app inventor, we will construct how we want the app to look.

* Go to the “Layout” drawer on the left palette and drag the Horizontal Arrangement onto the screen. Set its width property to fill parent.
* Then go to the “User Interface” drawer on the left palette and drag one button onto the horizontal arrangement showed on the screen. Name this button “RedButton” and under its properties make it red. Change the text property to “Red”
* Repeat two more time naming the other two “GreenButton”, “BlueButton”, set their text and colors. It should look like this below:
* Image
* Now go to the “Drawing and Animation” on the left palette and drag out the canvas component under the button arrangement from earlier.
* Under the Canvas’s properties set its width to fill parent and its height to 300 pixels.
* Now drag out another horizontal arrangement as you did earlier and place it beneath the canvas. Set its width to fill parent. Now drag out four buttons onto the arrangement and name them as:

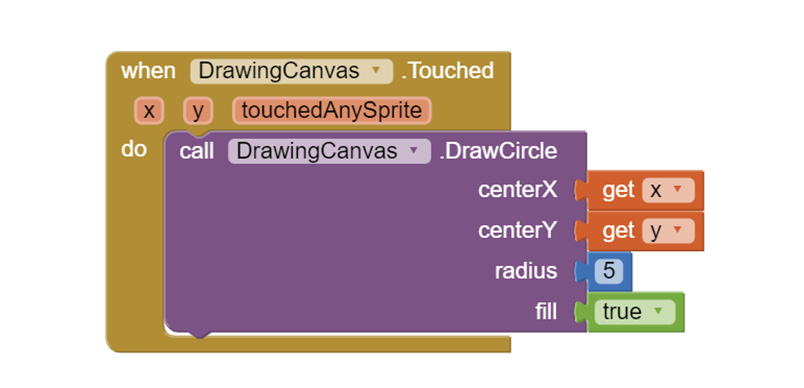
1. Take Picture
2. Rename the button “TakePictureButton”
3. Change text properity to “Take Picture”
4. Wipe
5. Rename the button “WipeButton”
6. Change text properity to “Wipe”
7. Big Dots
8. Rename the button “BigButton”
9. Change text properity to “Big Dots”
10. Small Dots
11. Rename the button “SmallButton”
12. Change text properity to “Small Dots”

* Finally, under the media drawer on the left palette drag out the “camera” component anywhere onto your screen. It does not matter since it is a non-visible component.
* Here is what it should look like:

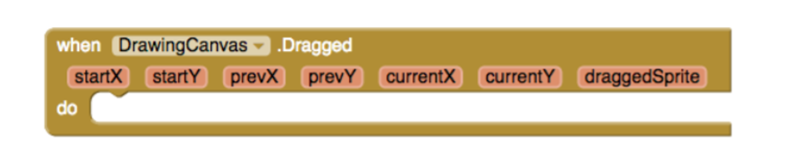


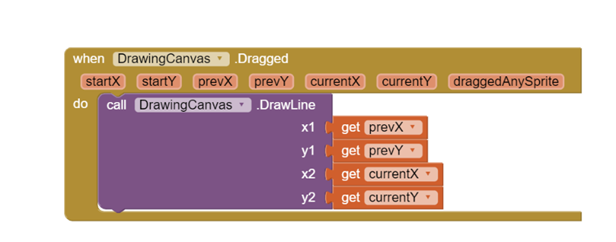
Now moving onto the block coding:

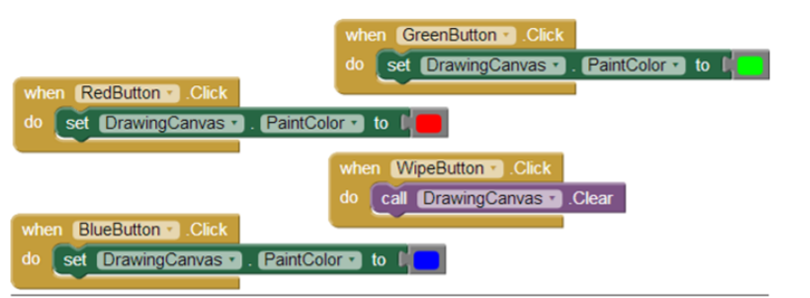
* Go to the block editor and we will start by doing the if the screen is touched then draw event
* In the Blocks Editor, select the drawer for the DrawingCanvas and then drag the DrawingCanvas.Touched block to the workspace
* From the DrawingCanvas drawer, drag out a DrawingCanvas.DrawCircle command and place it within the DrawingCanvas.Touched event handler
* On the right side of the DrawingCanvas.DrawCircle block, you’ll see three sockets for the arguments we need to plug in: x, y, and r. The x and y arguments specify the location where the circle should be drawn, and r determines the radius (or size) of the circle.
* Drag get blocks out for the x and y values and plug them into the sockets in the DrawingCanvas.DrawCircle block
* For now, set the radius to 5 using the math block
* Set the fill to true from the logic drawer



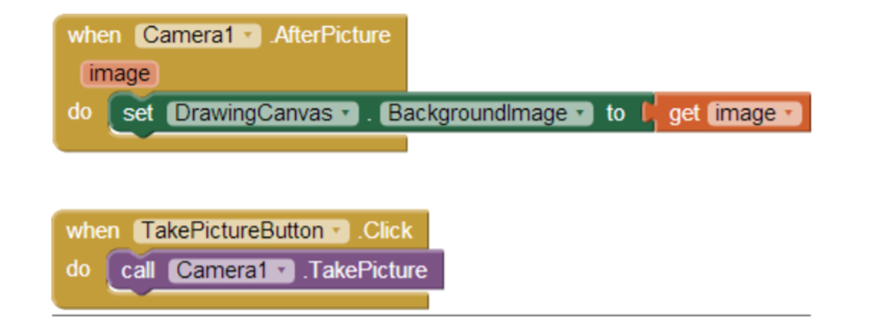
* Now We will make the event that draws the line
* From the DrawingCanvas drawer, drag the DrawingCanvas.Dragged block to the workspace.



* From the DrawingCanvas drawer, drag the DrawingCanvas.DrawLine block into the DrawingCanvas.Dragged block.
* Drag out get blocks for the arguments you need. A get prevX and get prevY should be plugged into the x1 and y1 sockets, respectively. A get currentX and get currentY should be plugged into the x2 and y2 sockets, respectively.  
  
* Now we will set up the colors to their buttons
* Open the drawer for RedButton and drag out the RedButton.Click block.
* Open the DrawingCanvas drawer. Drag out the set DrawingCanvas.PaintColor to block (you might need to scroll through the list of blocks in the drawer to find it) and place it in the “do” section of RedButton.Click.
* Open the Colors drawer and drag out the block for the color red and plug it into the set DrawingCanvas.PaintColor to block.
* Repeat those steps for green and blue buttons
* The final button to set up is WipeButton. Drag out a WipeButton.Click from the WipeButton drawer. From the DrawingCanvas drawer, drag out DrawingCanvas.Clear and place it in the WipeButton.Click block.



* The last button to code is the take picture button
* Open the TakePictureButton drawer and drag the TakePictureButton.Click event handler into the workspace.
* From Camera1, drag out Camera1.TakePicture and place it in the TakePictureButton.Click event handler.
* From Camera1, drag the Camera1.AfterPicture event handler into the workspace.
* From DrawingCanvas, drag the set DrawingCanvas.BackgroundImage to block and place it in the Camera1.AfterPicture event handler.
* Camera1.AfterPicture has an argument named image, which is the picture that was just taken. You can get a reference to it by using a get block from the Camera1.AfterPicture block, and then plug it into DrawingCanvas.BackgroundImage.



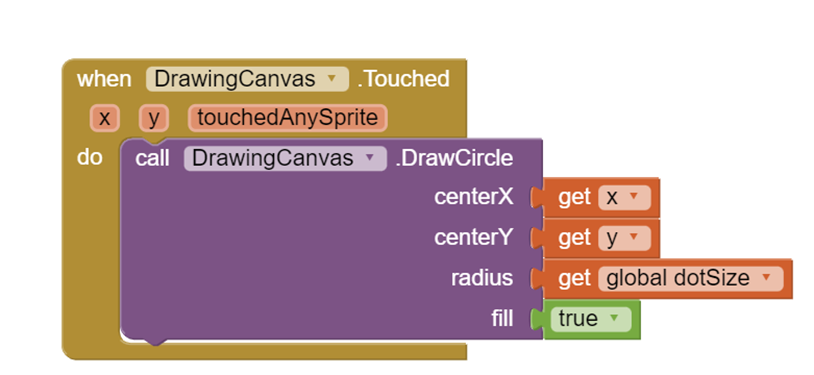
* Changing the Dot size:
* The first step is to define a variable called dotSize:

1. In the Blocks Editor, from the Variables drawer of the Built-in blocks, drag out an initialize global name to block. Within the initialize block, change the text “name” to “dotSize”.
2. Notice that the initialize global dotSize to block has an open socket. This is where you can specify the initial value for the variable, or the value to which it defaults when the app begins. (This is often referred to as “initializing a variable” in programming terms.) For this app, initialize the dotSize to 2 by creating a number 2 block (use the typeblocking feature: type a “2” in the Blocks Editor and then press Return) and then plugging it into initialize global dotSize to

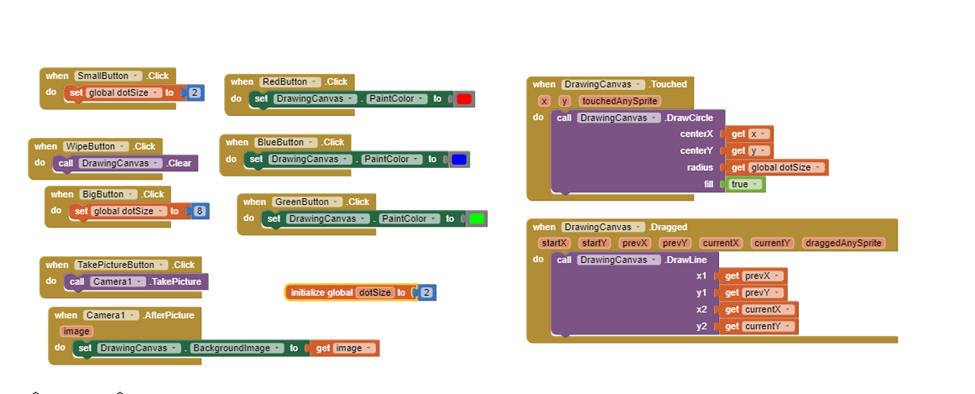


* Now we are going to go back to the argument of DrawingCanvas.DrawCircle in the DrawingCanvas.Touched event handler so that it uses the value of dotSize rather than a certain number.

1. Remove the number (5) from the radius spot and place the 5 into the trash in the bottom right corner.
2. Drag out a get block from the initilize global dotsize to block. You should see a get global dotSize block that provides the value of the variable. Drag it into the radius block.



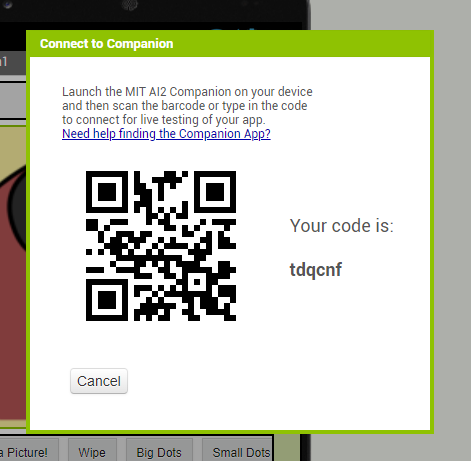
The Final app should look like this:





Testing the App:

* Now we will use the app we downloaded to test if our app works, first press the connect tab at the top near the projects tab. Then press AI Companion and it will generate a QR Code and another code.



* Now open your app on your device and either scan or enter the code to test your app. Make sure to test all your components to make sure it works.
* Once you have tested everything and fixed any errors go back to your website/project and select the build tab at the top. Then press android app (.apk) and it will generate a QR code for you to scan to get a download message for the app.
* Then follow the instructions to download the app onto your phone or tablet and now you have made an app!!!