Worksheet 1 – Tutorial

An online version of these worksheets is available at:
http://ronanjsmith.com/ComputingInTheClassroom/worksheets.html.

In this tutorial, you will discover how to create simple apps using MIT App Inventor. Please carry out this work individually. You will create 3 apps:

- 'TalkToMe'
- 'BallBounce'
- 'DigitalDoodle'

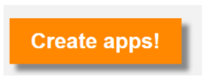
To carry out the tasks on this worksheet you will need to have a Google account. If you don't have one already, please create one at this URL: https://accounts.google.com/signup. It is up to you whether you use your own personal email address, your school email address or decide to create your own Gmail address.

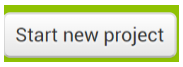
For each tutorial on this worksheet, you will be expected to start from the MIT App Inventor homepage, which you can find at http://appinventor.mit.edu/explore/.

1. TalkToMe

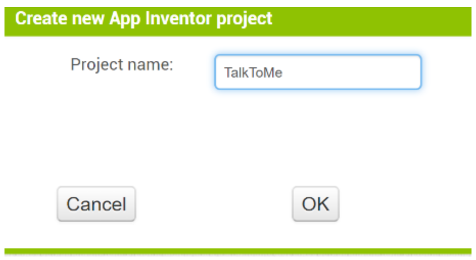
'TalkToMe' is an app that speaks a phrase out-loud to you when you press a button.

1. Click on the 'Create Apps' option in the top right-hand corner.
2. Click on the 'Start new project' button on the next screen.

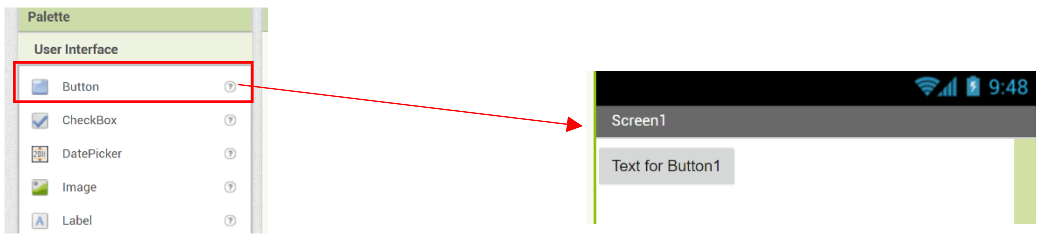




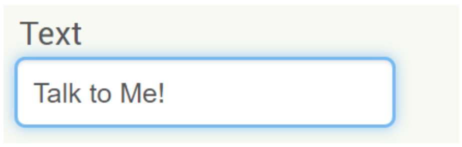
3. Name your project 'TalkToMe' and select OK. (No spaces are allowed.)



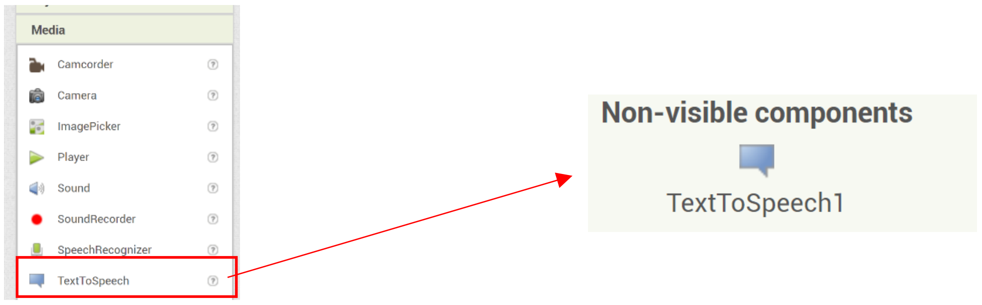
4. If you get a pop-up box, appearing, select the 'Do not show again' option and then click continue. If you don't see this box, skip to the next step.
5. You should now be in the 'Design View' or 'Designer' for your app. Here you can design the user interface (the look and feel of the screen) for the app by dragging and dropping different items from the palette at the left-hand side.
6. The first thing we want to do for the user interface is to add a button. This can be done by finding the 'Button' option in the palette, dragging it across and dropping it into the editor. You will see that a button appears with the text 'Text for button 1'. The button will also be added to the 'Components' list on the right-hand side.



If you click on the button, you will see the 'Properties' for the button appear down the right-hand side. Inside the properties for the button, scroll down to the 'Text' option and change it from 'Text for Button1 to 'Talk to Me!'.



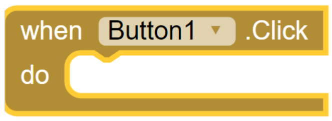
7. The final component we need to add to the user interface is called a 'TextToSpeech' component, and it can be found under 'Media' in the palette at the left-hand-side. Drag and drop this component into your app's screen and it will appear at the bottom under 'Non-visible components'.



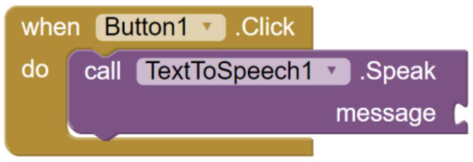
8. Now that we have our user interface ready, it's time to start programming. Select the 'Blocks' button in the top right corner to enter the blocks view.

9. The blocks view is where you write your program to control what the app should do. For the TalkToMe app, we want the phone to say something when we click the button. On the left-hand side of the screen you will see two types of block. These are built-in blocks, which are always available and component blocks, which are available for each of the components on the user interface.

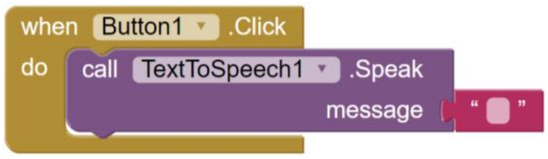
10. Find the components for your button, which should be called 'Button1' on the left-hand-side. Click that button and find the 'When Button1 Clicked' block. Drag it into the viewer in the middle of the screen.



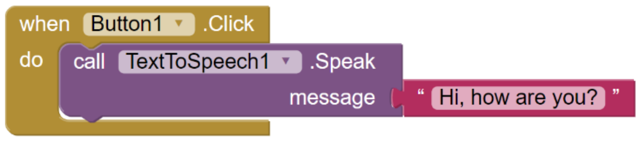
11. Next, find the 'TextToSpeech' component and find the purple block that says 'call TextToSpeech1.Speak'. Drag it and drop it inside the block that is already in the viewer. It should click into place like this:



12. Finally, we need to find the empty text block. This can be found inside the built-in 'Text' components. Drag and drop it onto the end of the block currently in the editor, so it looks like this:



13. Inside the speech marks you can type anything you like. Try typing something into it and pressing the button in the app on your phone/tablet. Your phone should speak the words in the text box. For example, if you wanted your phone to say "Hi, how are you?", you could do it like this:



Congratulations! You have now created your first Android app! Can you think of any extensions you could make to this app to make it better?

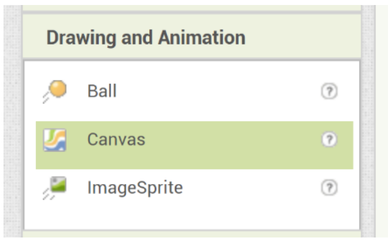
Note: You can run this app at home! To do this you will need to have the 'MIT AI2 Companion' app installed on your Android phone/tablet. Open the app and hit 'Scan QR code'. Then on App Inventor on the computer, go to your app and select 'Connect' (like you did for the USB connection before) then select 'AI Companion'.

Help sheets will be provided for you to run your apps in the school.

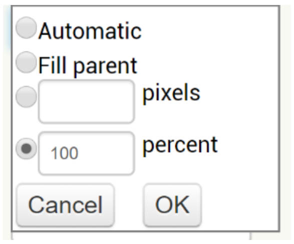
2. BallBounce

For the 'BallBounce' app, you will create an app that allows you to throw a ball around the screen.

1. We will now create a second app. If you are still in the 'TalkToMe' app window then select 'Projects'->'Start New Project'. Otherwise, go to 'Create Apps!'->'Start New Project'.
2. This time name the app 'BallBounce'. Remember no spaces are allowed.
3. This time, we will be using a 'Canvas' component, which can be found under the 'Drawing and Animation' title in the palette. Drag and drop this onto the screen.



4. We want this canvas to take up the whole screen. To do this we must select the canvas component on the screen, and find its width and height in the properties bar on the right-hand-side. Select 'percent' and enter the value '100' for both height and width. This will make the canvas fill the whole screen.

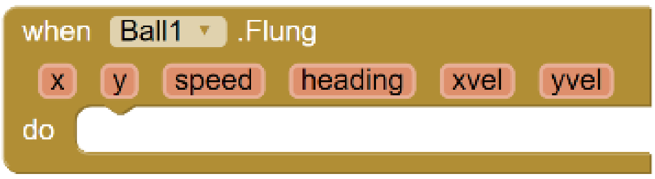


5. Also from the 'Drawing and Animation' section in the palette, you will see a 'Ball' component. Drag and drop this into the canvas, and to make it bigger, increase its radius inside the properties bar to 15.

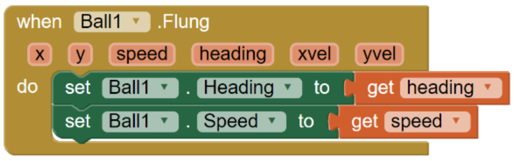
6. Now, select the 'Blocks' button in the top right corner to enter the blocks viewer, where you will create the program for your app.

7. You will notice that in the 'blocks' sidebar the built-in blocks are the same as last time, but you now also have blocks available for the canvas and the ball in your user interface.

8. Inside the 'Ball' component, find the 'when Ball1.Flung' block and drag it into the editor.

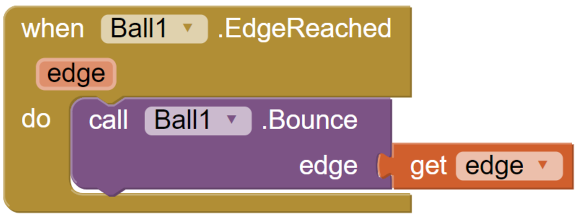


9. Also inside the blocks for the ball, we want to find the green blocks for ‘set Ball1.Speed to’ and ‘set Ball1.Heading to’. Drag them and drop them inside the block you have already placed in the editor.
10. The next step is to click on the word ‘speed’ inside the ‘when Ball1.Flung’ block inside the editor. Here, the word ‘speed’ is an example of a **parameter**, and so are ‘x’, ‘y’, ‘heading’, ‘xvel’ and ‘yvel’. When you click on ‘speed’, a new block should appear that says ‘get speed’. Drag and drop this to the end of the green block that sets the balls speed.
11. Do the same for heading. Your blocks should now look like this:



```
when Ball1 .Flung
  x  y  speed  heading  xvel  yvel
do  set Ball1 . Heading to  get heading
    set Ball1 . Speed to  get speed
```

12. If you are able to run this app on your phone, you should see that we now have an app which has a ball you can throw around by sliding your finger across it. There is one problem however, the ball flies off the edge of the screen and doesn’t come back!
13. Put the following blocks into your editor to make the ball bounce off the walls when it hits them:



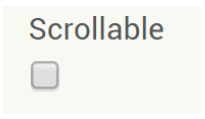
```
when Ball1 .EdgeReached
  edge
do  call Ball1 .Bounce
                 edge  get edge
```

You have now finished the BallBounce app, well done! Once again, try to think of more ways you could extend this app and make it better.

3. DigitalDoodle

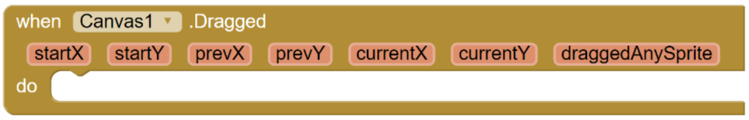
The ‘DigitalDoodle’ app will allow users to draw on the screen of their smartphone, as if they are drawing with a pencil.

1. Start a new project as you did before. This time call the project ‘DigitalDoodle’.
2. Once your app has loaded up, immediately check the ‘Scrollable’ property in the properties bar and make sure the box is **not ticked**.



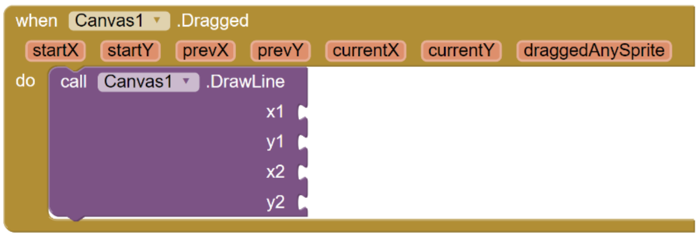
Scrollable
☐

3. Just like with the BallBounce app, we want to have a Canvas. Drag and drop the ‘Canvas’ component from the palette into the editor. Just like you did before, make sure the width and height properties for the canvas are set to 100%, so it takes up the whole screen.
4. Switch to the blocks editor, by selecting the ‘Blocks’ button at the top right.
5. We want to make it look like the screen is being drawn on when the user **drags** their finger across the canvas, so we need to use the ‘when Canvas1.Dragged’ block.



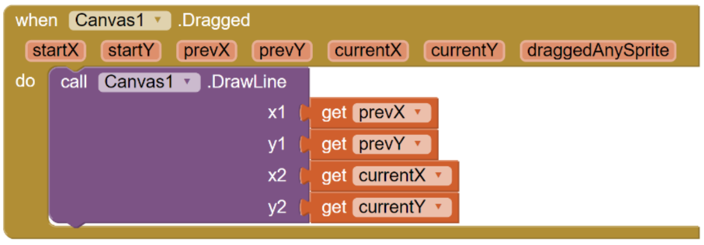
```
when Canvas1 .Dragged
 startX  startY  prevX  prevY  currentX  currentY  draggedAnySprite
do
```

6. If you look again in the blocks for the canvas component, you will find that there is a purple block called ‘Canvas1.DrawLine’. Put this block inside the ‘when Canvas1.Dragged’ block, making sure it clicks into place.



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

7. Finally, use the parameters in the top section of the ‘when Canvas1.Dragged’ block (for example ‘prevX’, ‘prevY’, ‘currentX’, ‘currentY’) to finish off your blocks like this:



```
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
```

You have now completed worksheet 1, well done!