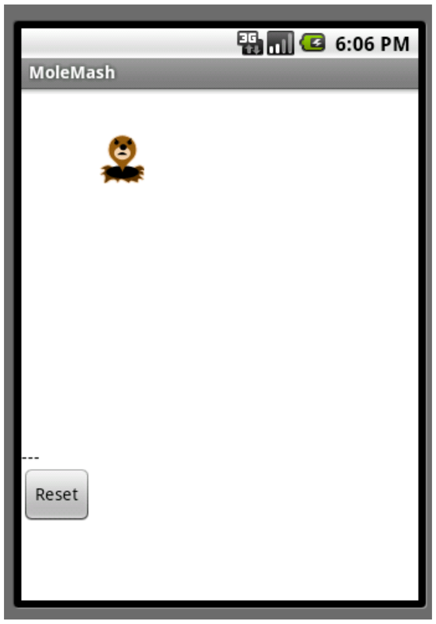
# MoleMash... Smash those Moles!

## Introduction

In this assignment you will create a simple game to play on your phone where the goal is to smash the mole before he disappears! Your finished product will look something like:



Note: The original tutorial can be found at http://appinventor.googlelabs.com/learn/tutorials/molemash/molemash.html, this is an extend version of this tutorial. There are some differences between this one and the original.

## The setup

Connect to the App Inventor web site and start a new project. Name it MoleMash, and also set the screen's title to MoleMash (in Properties on the right). Open the Blocks Editor and connect to the phone.

Download this picture of a mole and save it on your computer.

You'll design the game so that the mole moves once every second. If it is touched, the score increases by one, and the phone vibrates. Pressing ***reset*** resets the score to zero.

After completing this tutorial you will be more familiar with:

- images
- timers
- random numbers
- text

## The Components

To create this game we will need the following components. For each of the components below drag one from the Palette onto the Viewer.

- A Canvas, and then rename it MyCanvas. This is the area where the mole jumps around in.
- A Label, and then rename it ScoreLabel. This shows the players' current score.
- A Button, and then rename it ResetButton.
- A Clock, and rename it MoleTimer. This will make the mole move.
    - Note: the clock will appear in the non-visible components area, located below the screen with the canvas, label and button.
- A Sound, and rename it Noise. This is used to make the phone vibrate when the mole is smashed.
    - Note: the clock will appear in the non-visible components area with the Clock.

## Setup The Components

Now that all the components are added we need to adjust some of their settings so we can correctly use them in our game!

- First select MyCanvas, and changes its dimensions to 300 pixels wide and 300 pixels tall.
- Next select ScoreLabel and set the Text to 'Score: 0' (without the quotes)
- Next select ResetButton, and set the Text to 'Reset' (without the quotes)
- Finally select MoleTimer and set the TimeInterval to 1000 milliseconds. Also, make sure Enabled is checked.

## Adding the Mole

To add the moving mole we'll use a *sprite*.

Sprites are images that can move on the screen within a Canvas. Sprites can detect when they are touched.

Drag an ImageSprite component onto the Viewer. You'll find this component in the Animation category of the Palette. Place it within MyCanvas area. Rename it to Mole in the components bar.

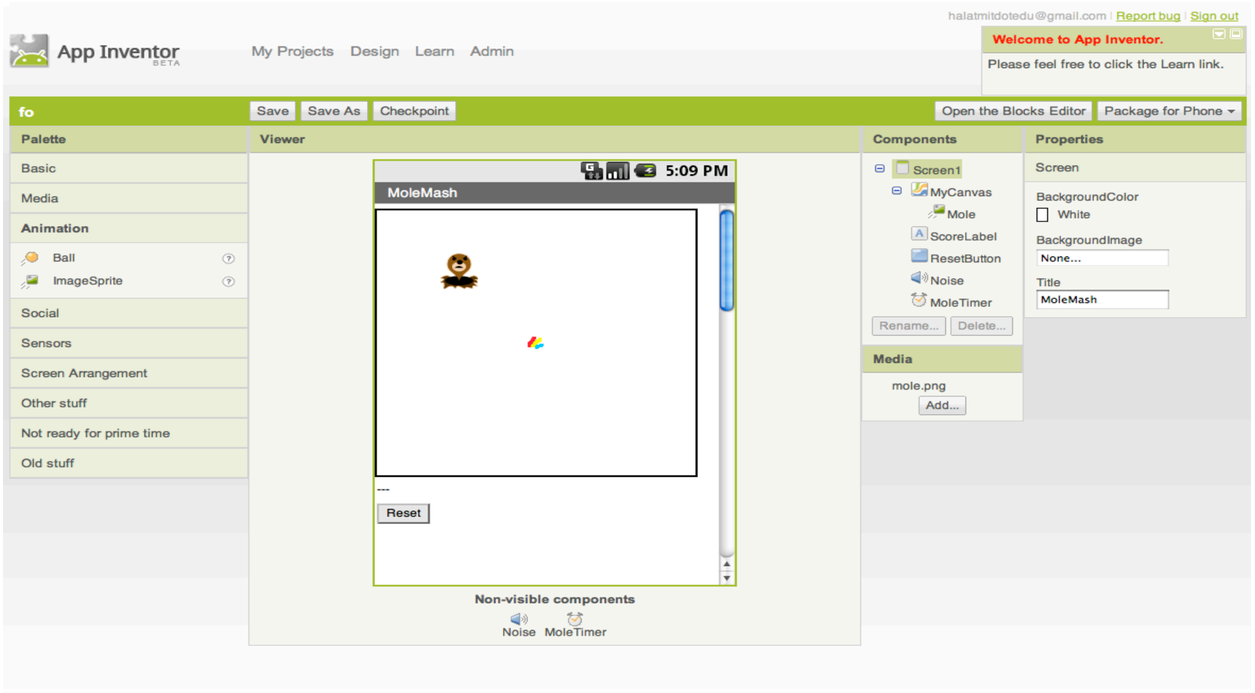
Set these properties for the Mole sprite:

- Picture: Use mole.png, which you downloaded to your computer at the beginning of this tutorial.
- Enabled: checked
- Visible: checked

For all the other properties leave them as default.

For the X and Y properties they will be automatically updated based on where you dragged the mole picture too. Try dragging the mole picture around and watch the coordinates change!

After all of these steps your screen should look like this:



BackgroundColor: White

BackgroundImage: None...

Title: MoleMash

## Component Behavior and Event Handlers

Now you'll specify the component behavior. This introduces some new App Inventor ideas. The first is the idea of a *procedure*.

A procedure is a sequence of statements that you can refer to all at once as single command. If you have a sequence that you need to use more than once in a program, you can define that as a procedure, and then you don't have to repeat the sequence each time you use it. Procedures in App Inventor can take arguments and return values. This tutorial covers only the simplest case: procedures that take no arguments and return no values.

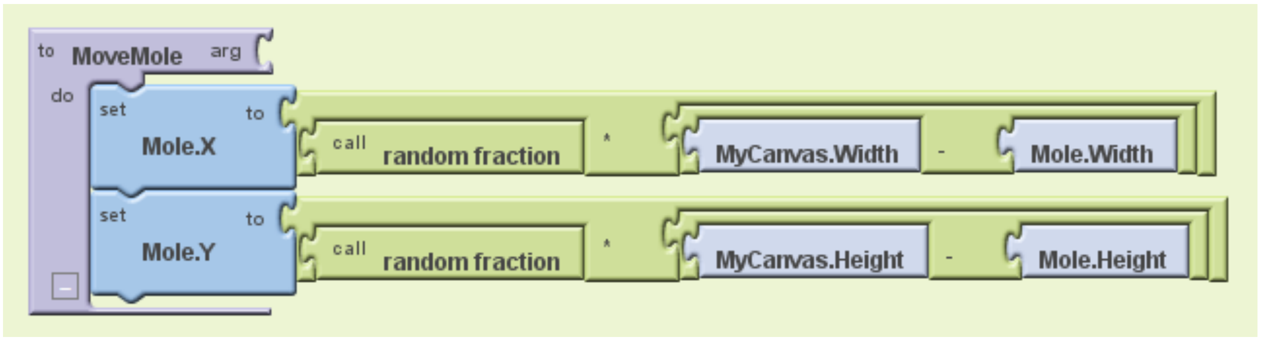
## Define Procedures

Define two procedures:

- MoveMole moves the Mole sprite to a new random position on the canvas.
- UpdateScore shows the score, by changing the text of the ScoreLabel

Start with MoveMole:

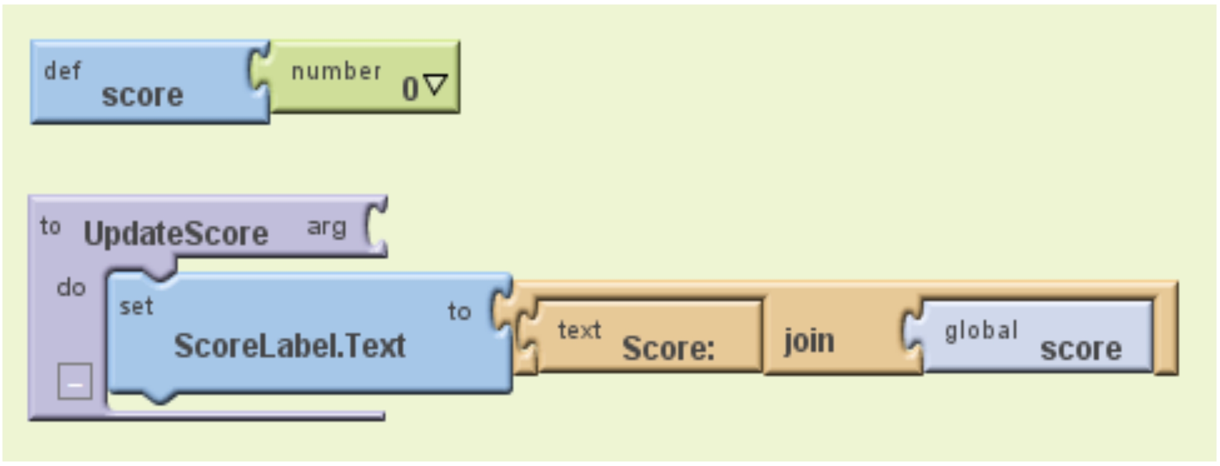
- In the Blocks Editor, under Built-In, open the Definition drawer. Drag out a to procedure block and change the label procedure to MoveMole.
- Note: There are two similar blocks: procedure and procedureWithResult. Here you should use procedure.
- The to MoveMole block has a slot labeled do. That's where you put the statements for the procedure. In this case there will be two statements: one to set the mole's *x* position and one to set its *y* position. In each case, you'll set the position to be a random fraction, between 0 and 1, of the difference between the size of the canvas and the size of the mole. You create that value using blocks for random-fraction and multiplication and subtraction. You can find these in the Mathdrawer.
- Build the MoveMole procedure. The completed definition should look like this:



- Leave the arg socket for MoveMole empty because MoveMole does not take any arguments. Observe how the blocks connect together: The first statement uses the Mole.X gets block to set mole's horizontal position. The value plugged into the block's socket is the result of multiplying:
    a. the result of the call random-fraction block, which a value between 0 and 1
    b. the result of subtracting the mole's width from the canvas's width.
- The vertical position is handled similarly.

With MoveMole done, the next step is to define a variable called score to hold the score (number of hits) and give it initial value 0. Also define a procedure UpdateScore that shows the score in ScoreLabel. The actual contents to be shown in ScoreLabel will be the text "Score: " joined to the value of the score.

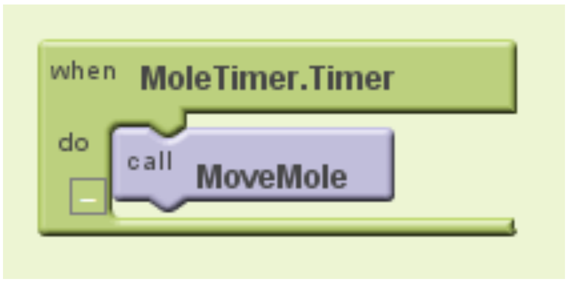
- To create the "Score: " part of the label, drag out a text block from the Text drawer. Change the block to read "Score: " rather than "text".
- Use a join block to attach this to a block that gives the value of the score variable. You can find the join block in the Text drawer.
- Here's how score and UpdateScore should look:



## Add a Timer

The next step is to make the mole keep moving. Here's where you'll use MoleTimer. components. Timer triggers repeatedly at a rate determined by the TimerInterval.

Set up MoleTimer to call MoveMole each time the timer fires, by building the event handler like this:



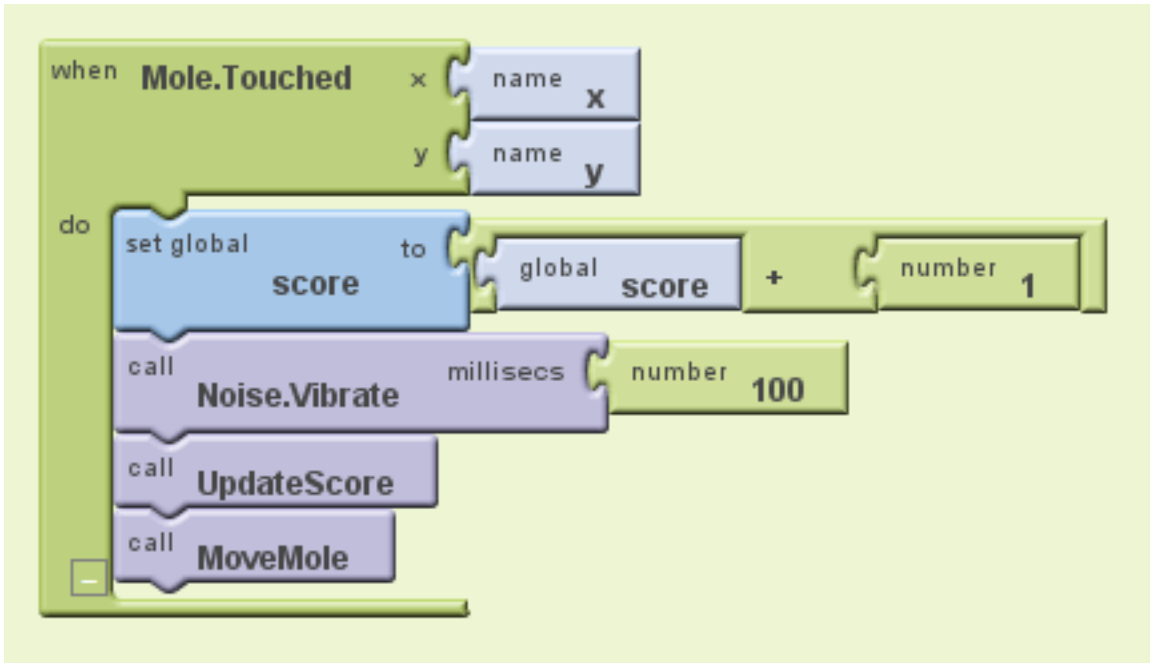
Notice how the mole starts jumping around on the phone as soon as you define the event handler. This is an example of how things in App Inventor start happening instantaneously, as soon as you define them.

## Add a Mole Touch Handler

The program should increment the score each time the mole is touched. Sprites, like canvases, respond to touch events. So create a touch event handler for Mole that:

1. Increments the score.
2. Calls UpdateScore to show the new score.
3. Makes the phone vibrate for 1/10 second (100 milliseconds).
4. Calls MoveMole so that the mole moves right away, rather than waiting for the timer.

Here's what this looks like in blocks. Go ahead and assemble the Mole.Touched blocks as shown.



Here's a tip: You can use typeblocking: typing to quickly create blocks.

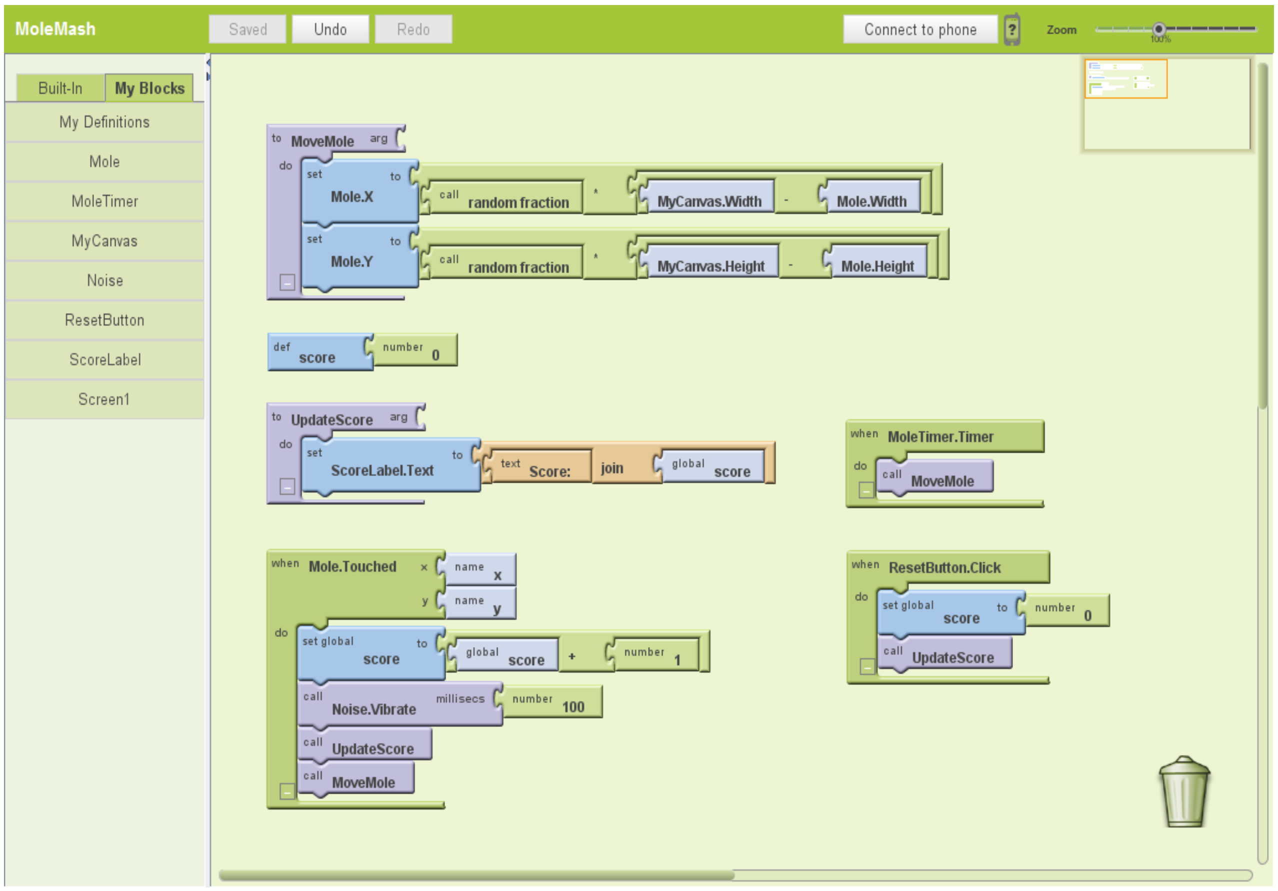
- To create a value block containing 100, just type 100 and press return.
- To create a MoveMole block, just type *MoveMole* and select the block you want from the list

## Reset the Score

One final detail is resetting the score. That's simply a matter of making the Reset button change the score to 0 and calling UpdateScore.

## Complete Program

Here's the complete MoleMash program:



```
to MoveMole arg
do  set Mole.X to  call random fraction * MyCanvas.Width - Mole.Width
    set Mole.Y to  call random fraction * MyCanvas.Height - Mole.Height

def score  number 0

to UpdateScore arg
do  set ScoreLabel.Text to  text Score:  join  global score

when MoleTimer.Timer
do  call MoveMole

when Mole.Touched  x name x
                   y name y
do  set global score to  global score + number 1
    call Noise.Vibrate millisecs  number 100
    call UpdateScore
    call MoveMole

when ResetButton.Click
do  set global score to  number 0
    call UpdateScore
```

## Extensions!

- How would you modify the game so that the mole moves more often?
- How would you modify the game to change the image to be that of a monkey?
- How would you modify the game to have more than one mole to smash?
- How would you modify the game so that if the somewhere that is not the mole is touched (the player misses the mole) one is subtracted from the score?