Digital Green

Feel nature with one click

2016-1-PT01-KA219-022939

Training Activity

9th to 13th January, 2017

AGRUPAMENTO DE ESCOLAS DR. SERAFIM LEITE

Erasmus+

What is App Inventor?.....1

***Exercise 1*- Let´s do some magic!**.....2

***Exercise 2*- Catch the WIND!**.....9

***Exercise 3 Part I* EnergyQuiz**.....15

***Exercise 3 Part II* – Energy Quiz**.....21

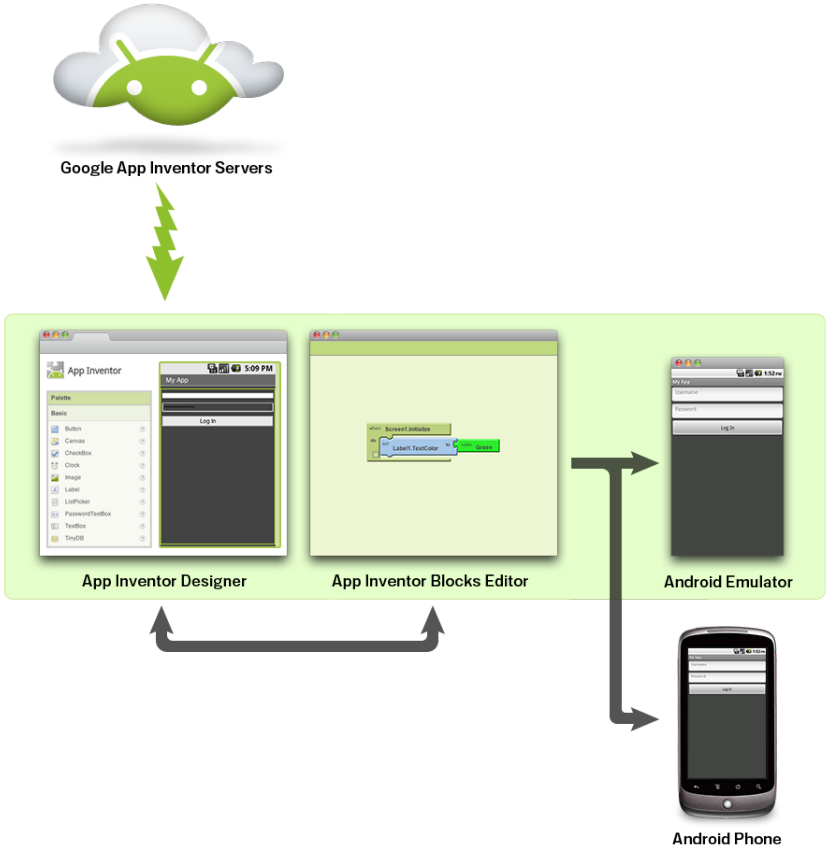
***Practice* - Types of energy**.....25

References.....26

What is App Inventor?

App Inventor for Android is an open-source web application originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT).

It allows newcomers to computer programming to create software applications for the Android operating system (OS). It uses a graphical interface which allows users to drag-and-drop visual objects to create an application that can run on Android devices.

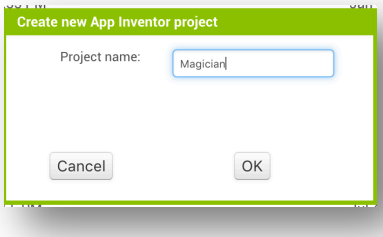


Exercise 1- Let´s do some magic!

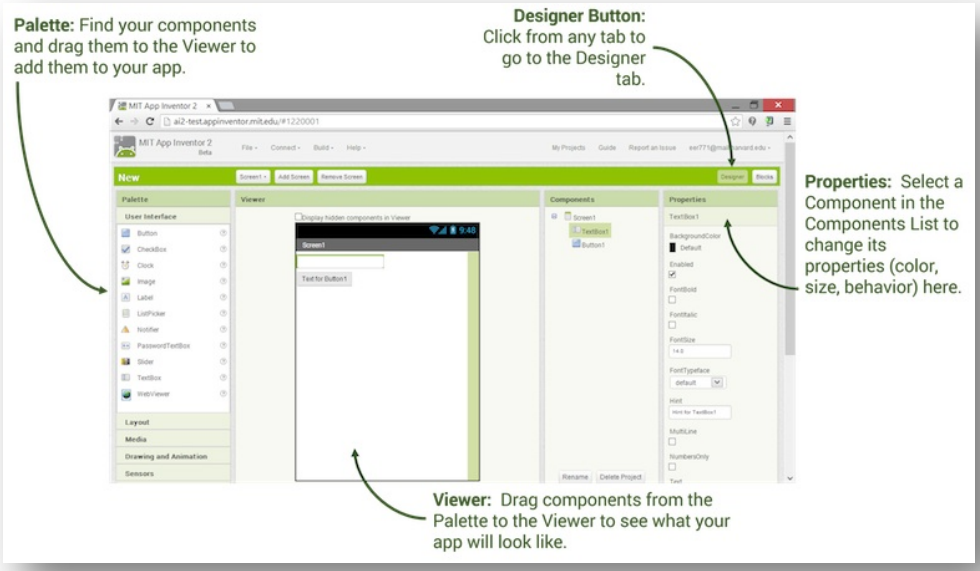
This will be your first app: "Let´s do some magic!"

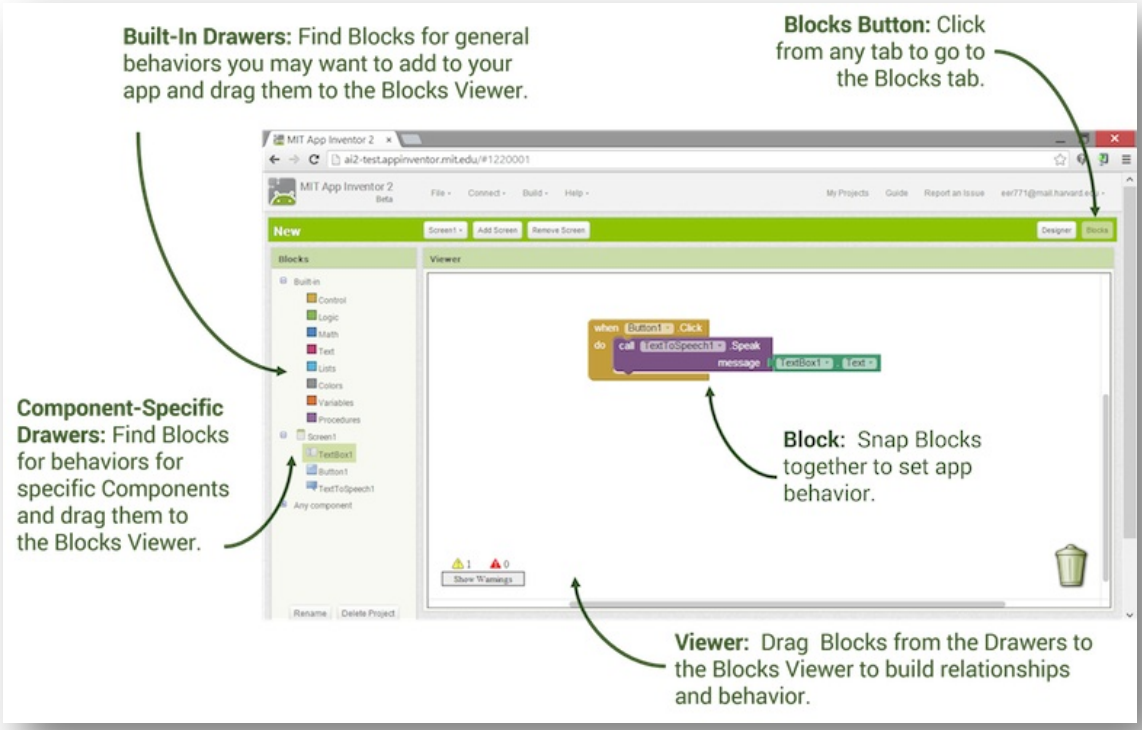
You can begin programming with App Inventor by opening a browser to **ai2.appinventor.mit.edu**.

1- Create a new Project:



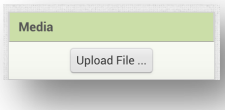
2- Know the environment



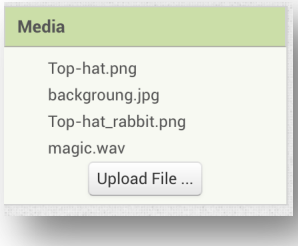


3- Download the folder “1stExercise” from Google Drive to your computer. You will need some images and sounds!

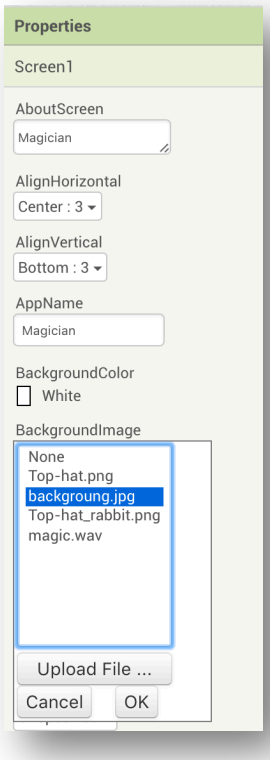
4- Upload all the media elements to you project



Now this area should look like this:

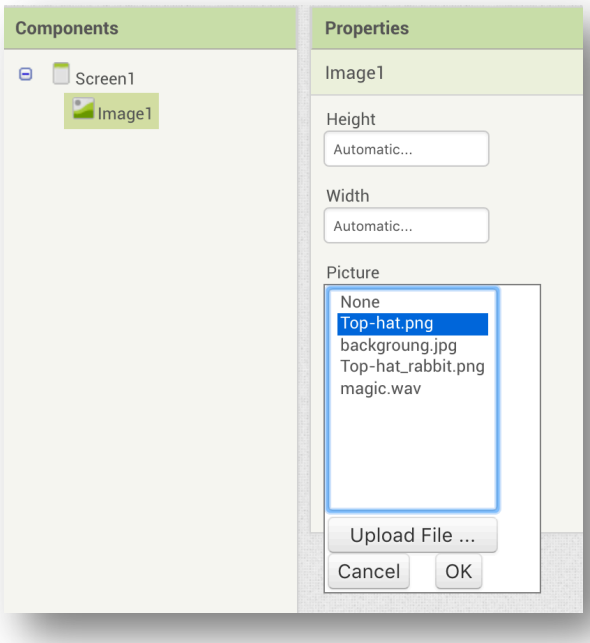


5- Change properties of the screen as showed:

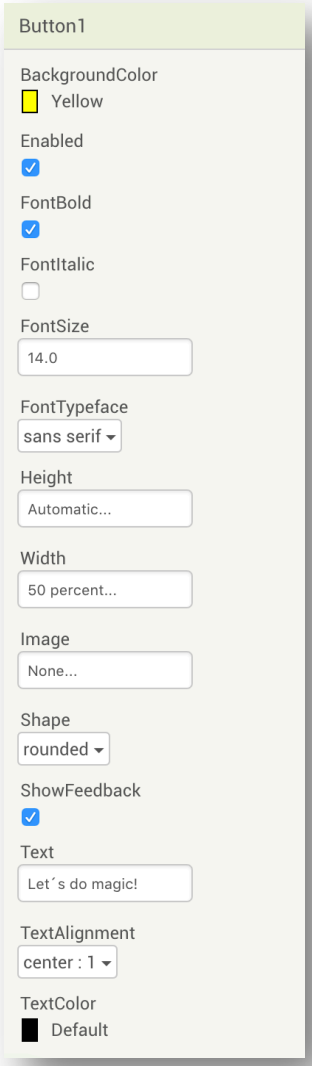


6- Now let's add the first component – an image.

Go to the Palette, open the User Interface drawer if it is not open, click Image, and drag it to the Viewer. You only will see a small square. Look at the Properties box on the right side of the Designer. It shows the properties of the image. Change it according to the following image:

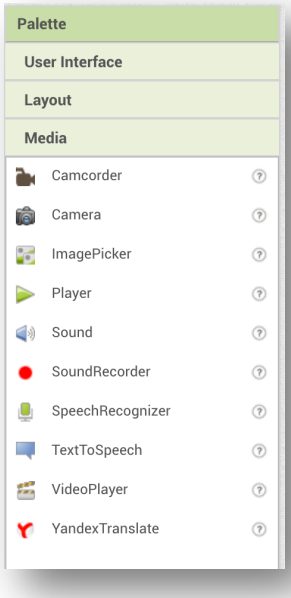


7- Add a button below the hat and change properties:



8- Adding the magic sound

In your app, you want to take a rabbit from the hat and at the same time a sound should be played. For this, you'll need to add the magic sound. Go to the Palette at the left of the Designer window and click the header marked Media to expand the Media section. Drag out a Sound component and place it in the Viewer. No matter where you drop it, it will appear in the area at the bottom of the Viewer marked "Non-visible components." Non-visible components are objects that do things for the app but don't appear in the visual user interface.



9- Click Sound1 to show its properties.

Click the Source property and then go through the steps to upload and choose the magic.wav file you downloaded earlier. Now, the Designer should appear as shown:

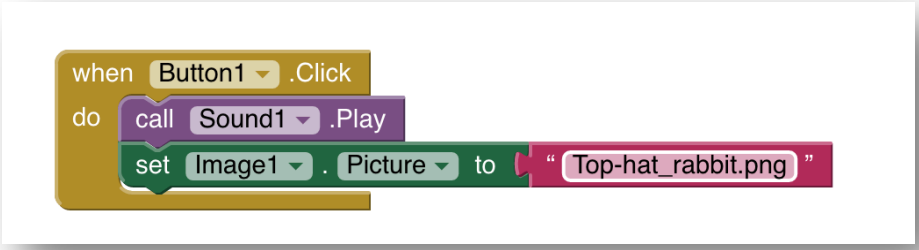
10- Adding Behaviours to the Components

You've just added Button, image, and Sound components as the building blocks for your first app. Now, let's make the Magic! You do this with the Blocks Editor. In the top right of the Component Designer, click "Blocks." Look at the Blocks Editor window. This is where you instruct the components what to do and when to do it.
You're going to direct the button to do two different things:
- play a sound when the user taps it.
- Change the image of the top hat.

At the top left of the window, beneath the Blocks header, you'll see a column that includes a Built-in drawer and a drawer for each component you created in the Designer: Button1, Image1, Screen1, and Sound1. When you click a drawer, you get a bunch of options (blocks) for that component. Click the drawer for Button1. The drawer opens, showing a selection of blocks that you can use to build the button's behaviour, starting with Button1.
Click the block labeled Button1.Click and drag it into the workspace. You'll notice that the word "when" is included on the Button1.Click block.

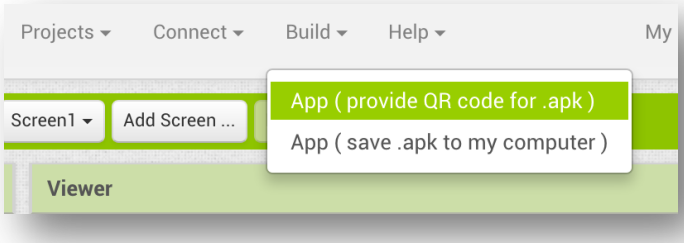
Blocks including the word "when" are called event handlers; they specify what components should do when some particular event happens. In this case, the event we're interested in happens when the app user click the button.

Next, you'll add some blocks to program what will happen in response to that event:



11- Testing your application

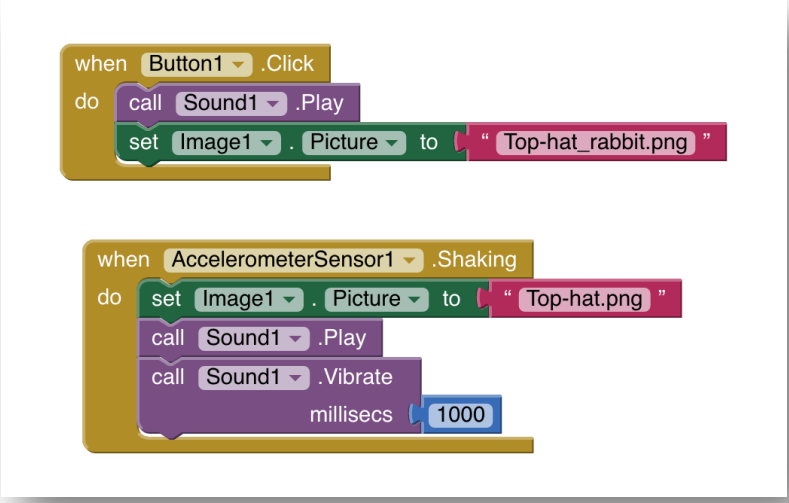
First you need to build the APK file so that you can execute it in an android device:



After a while you will see the QR Code! Install on your android device and have fun!

12- Now let's improve our App:
- If the device is shacked the rabbit will disappear;
- At the same same it should vibrate for 1 second and play the magic sound.

To do this you need to work with sensors, go to Designer, in the Palette components list, expand the Sensors area and drag out an AccelerometerSensor. As with any non-visible component, no matter where you place it in the Viewer, it will move to the "Non-visible components" section at the bottom of the Viewer.
You'll want to treat someone shaking the device as a different, separate event from the button click. This means that you need a new event handler. Go to the Blocks Editor. There should be a new drawer for AccelerometerSensor1. Open it and drag out the AccelerometerSensor1.Shaking block. It should be the second block in the list.



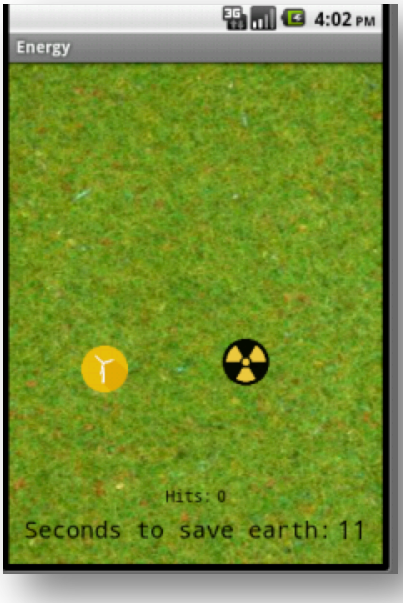
Task 1

Every time you shake the device the sound is played. Use variables to solve the problem!

Exercise 2- Catch the WIND!

This example shows you how to create a variation of MoleMash, a game inspired by the arcade classic Whac-A-Mole™ , in which mechanical critters pop out of holes, and players score points when they successfully whack them with a mallet.

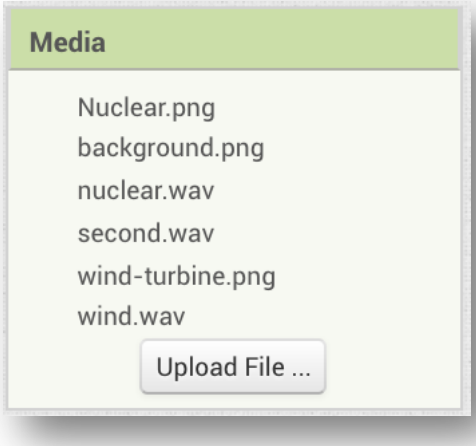
The theme of this game will be Energy – wind versus nuclear.



1- Download all the files from Google Drive (Exercise2)

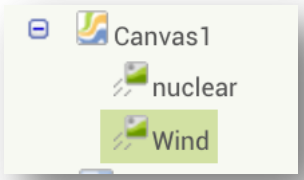
2- Create a new project “Energy”

3- Import all the media files



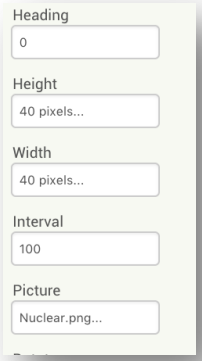
4- Create a new Canvas (Palette>Drawing and Animation). Change the height and width to 75%.

5- Inside the Canvas create two imagesprites (Palette>Drawing and Animation) and rename them:

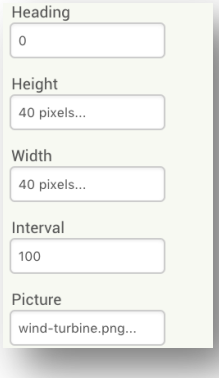


6- Change the both imagesprites properties:

nuclear

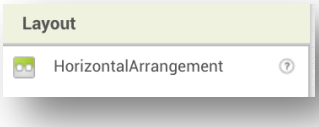


Wind



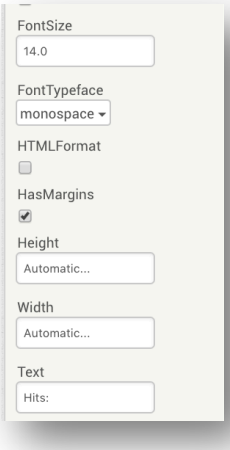
7- After the Canvas1 add a new button and rename it as “Start”.

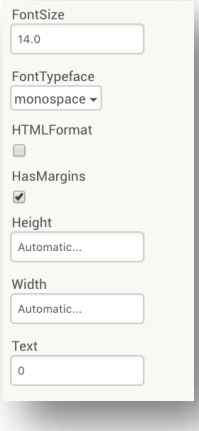
8- After the button insert a layout element:



9- Add two labels inside the HorizontalArragement and rename them as “labelhits” and “hits”

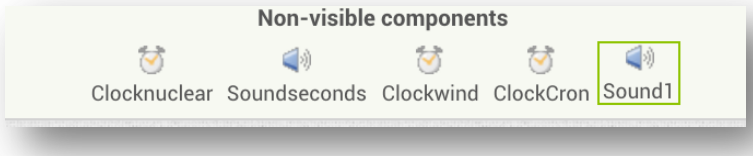
10- Change the properties of the two labels (*labelhits and hits)*:





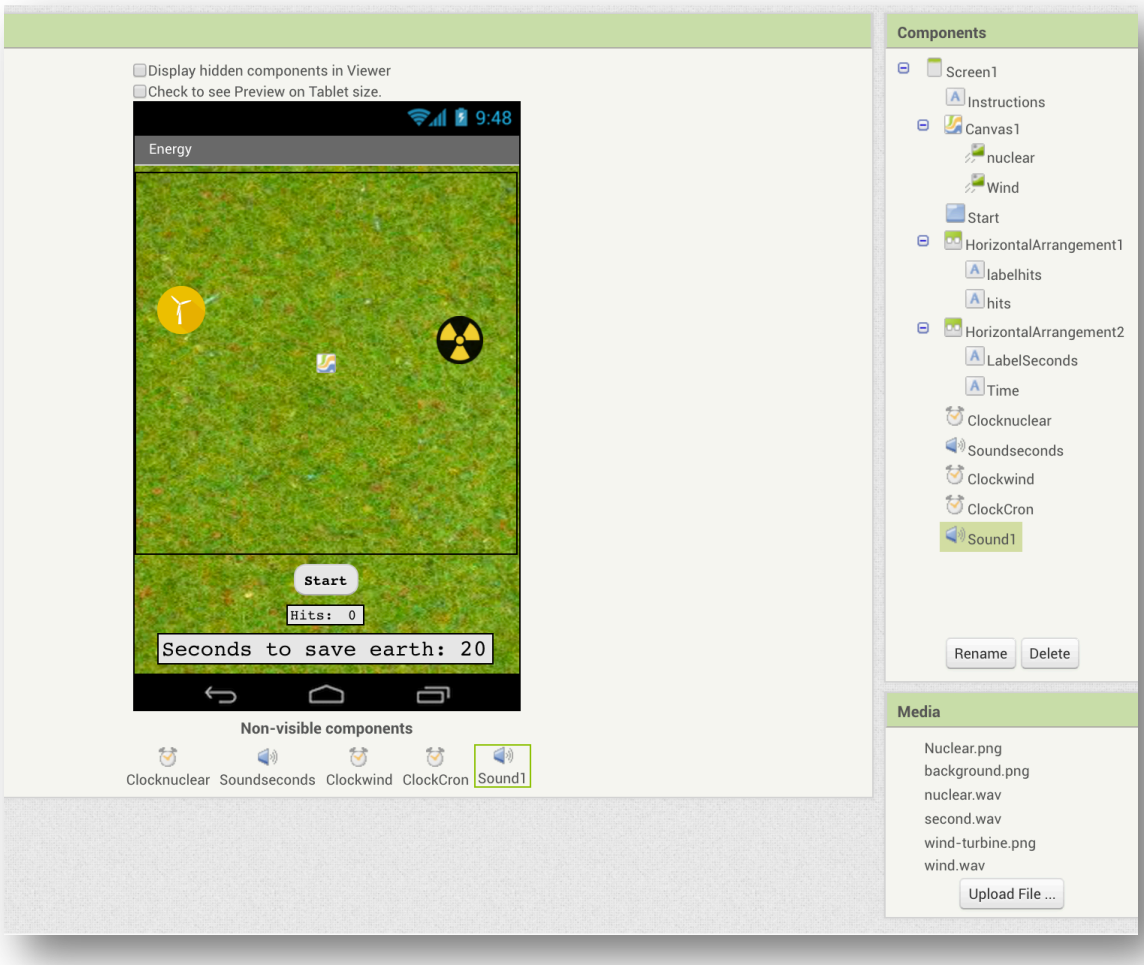
11- Bellow this HorizontalArragement create a new one, and inside it create two more labels. Rename them as “LabelSeconds” and “Time”. Change the text properties to “Seconds to save Earth” and “20”.

12- Add the following components and rename them (you can find the clock in palette> sensors).



Change the Soundseconds>Source to “second.wav” and all the clocks should be disabled.

13- Now, your screen should now look like something like:



Let's try to make sense!

Purpose of the game: Hit the wind energy image and avoid the nuclear energy

Rules:

The game will start when the button "start" is pressed

The images move randomly inside the canvas

+1 point if you hit on the nuclear energy image and play a sound (nuclear.wav)

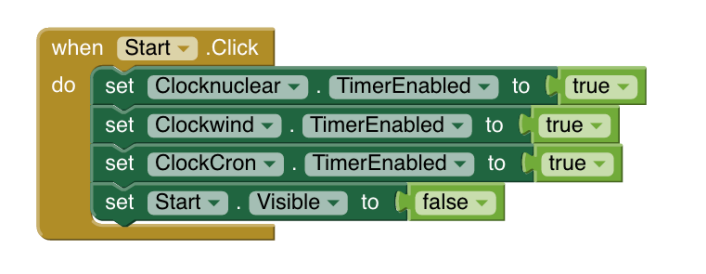
-1 point if you hit on the wind energy image and play a sound (wind-turbine.wav)

After 20 seconds the game will end

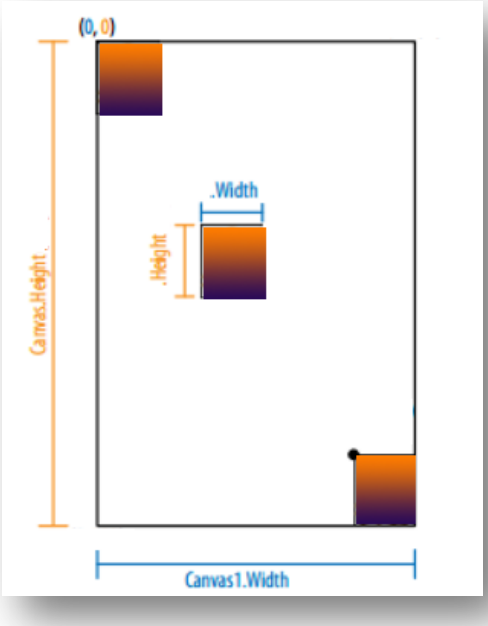
For each second should be played a sound (seconds.wav)

Implementing the logic – change to blocks view.

The game will start when the button "start" is pressed
The timers added before control the game. So the first thing you must do is enable them.

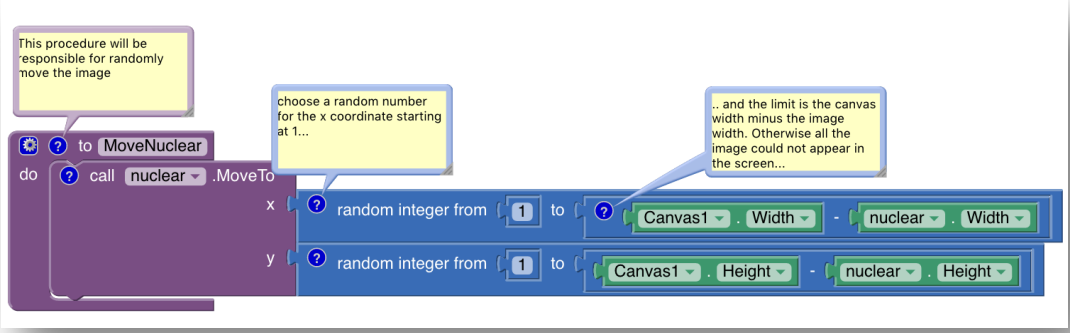


The images move randomly inside the canvas
To understand how to move the nuclear image for example, we need to look at how android graphics work. The canvas (and the screen) can be thought of as a grid with x (horizontal) and y
(vertical) coordinates, where the (x, y) coordinates of the upper-left corner are (0, 0). The x coordinate increases as you move to the right, and the y coordinate increases as
you move down. The x and y properties of an ImageSprite indicate where its upper-left corner is positioned.

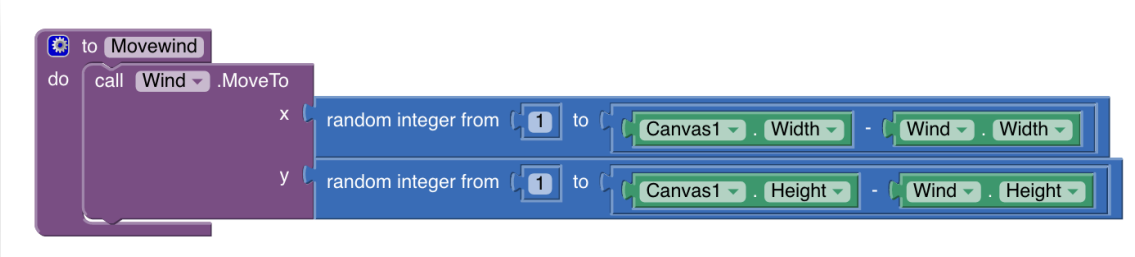


To determine the maximum available x and y values so that image fits on the screen, we need to make use of the Width and Height properties.

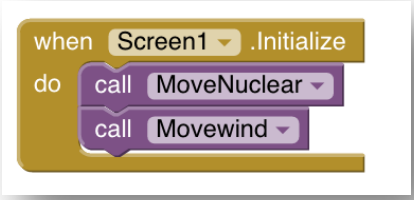
The next image shows the procedure you will create, annotated with descriptive comments (which you can optionally add to your procedure).



Do the same procedure to move the wind image:

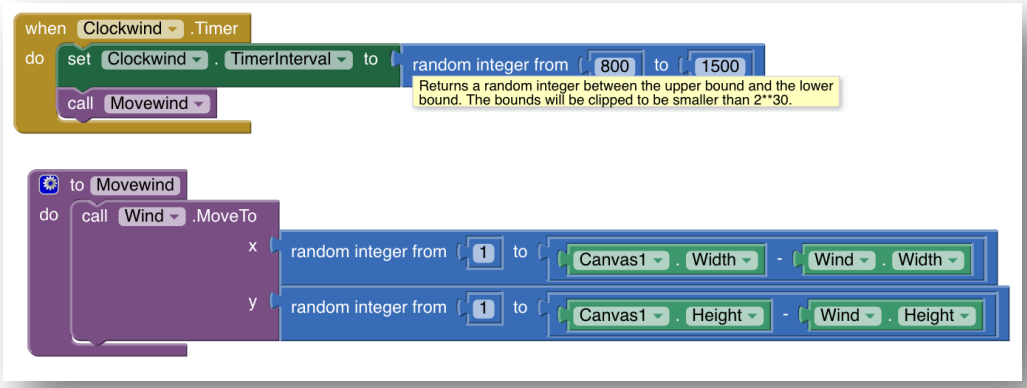


Now that you've written the Move procedure, let's make use of it. Because it's so common for programmers to want something to happen when an app starts, there's a block for that very purpose: Screen1.Initialize. All we have to do is call the procedure!



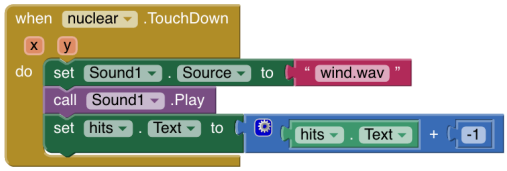
Making the image moves randomly will require the Clock component. We left the Timer Interval property for Clock1 at its default value of 1 000 (milliseconds), or 1 second. That means that every second, whatever is specified in a Clock1.Timer block will take place. Here's how to set that up:

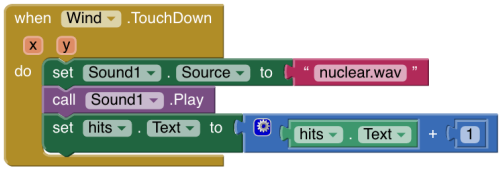
You have deal with the nuclear image. Now its the same thing for the wind image:



Test the App!

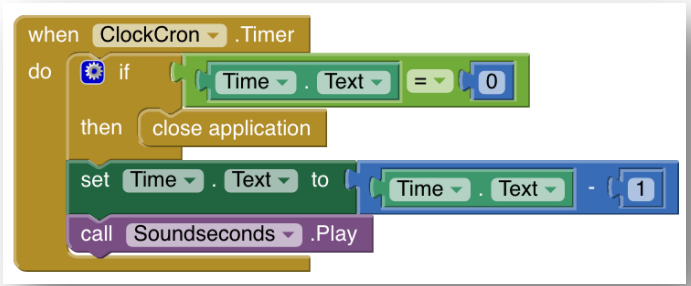
+1 point if you hit on the nuclear energy image and play a sound (nuclear.wav)





Test the App!

After 20 seconds the game will end
For each second should be played a sound (seconds.wav)



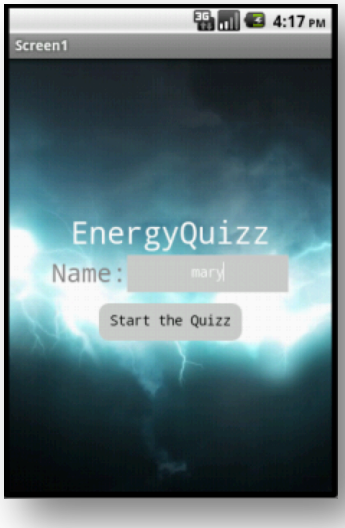
Test the App!

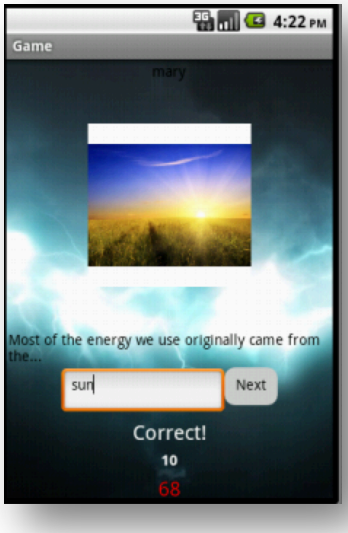
Task 2

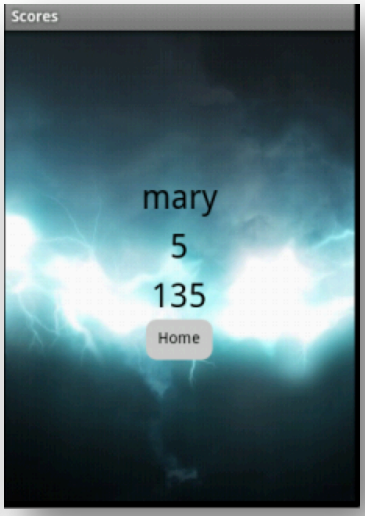
This game ends abruptly! How can you solve this? Tip: Add another screen…

Exercise 3 Part I EnergyQuiz

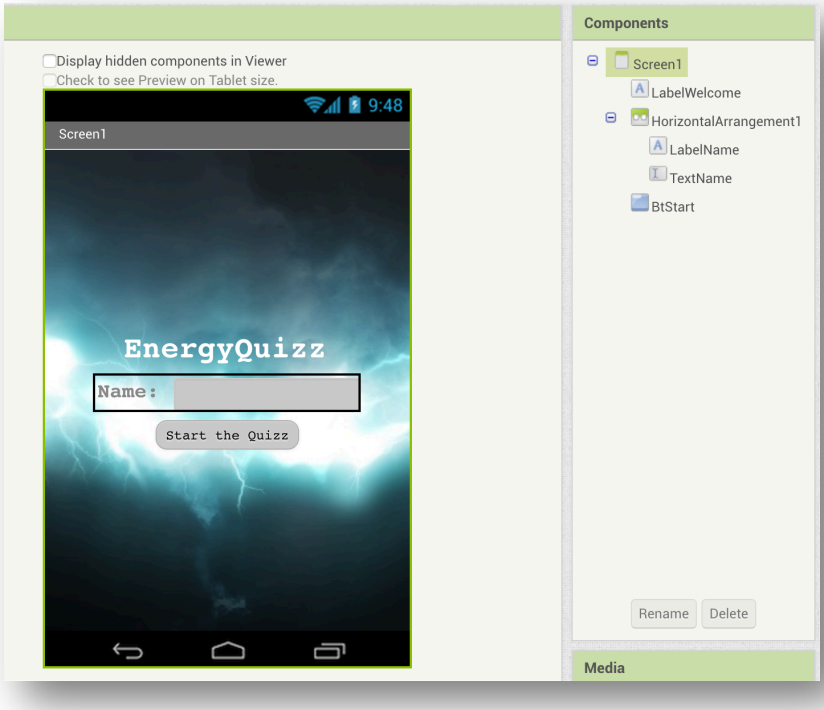
The Energy Quiz app has a simple interface for displaying the questions and allowing the user to answer:





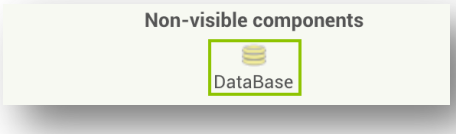


1. Download all the media files from Google Drive
2. Working with screens
   2.1. Most of the apps have a screen intro. All the App inventor starts on screen1 – this will be the intro. Add the components as showed in the image:

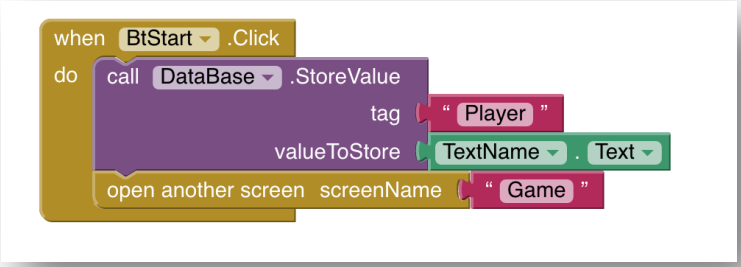


2.2. Create another screen and return to the first Screen (Screen1)

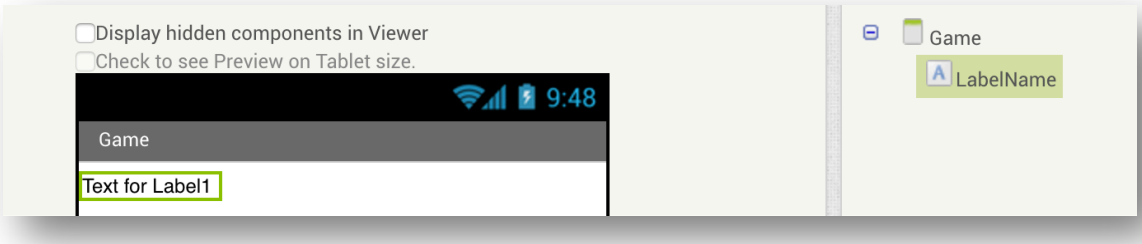
2.3. All the information will be loss when you change screen. Since we want to save the player name, it should be save. We will use a database component: TinyDB. Drag this component and rename it as “Database”:



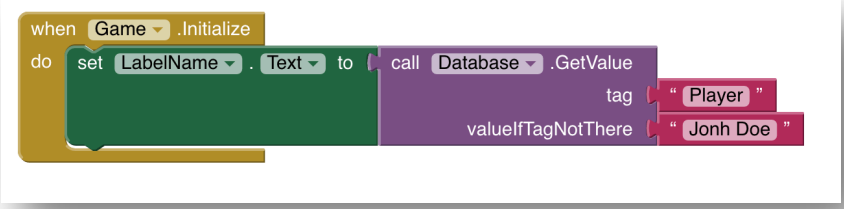
Now you must go to the blocks view and give the correct instructions.



2.4. Go to Game screen, add a label and rename as “PlayerName”



We want to see on this label the name that the player previously inserted and stored in the database:
So, when the screen initializes something must happen:



2.5. Don't forget to create a block to make possible go to the screen Game!

Test the App!

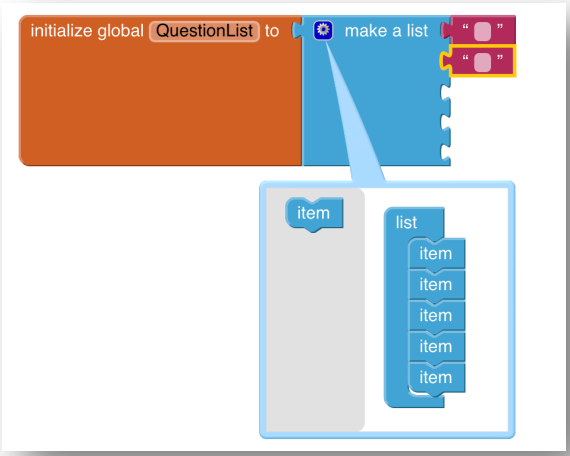
3. On screen Game, build the components from the snapshot:

You'll need to program the following behaviours:

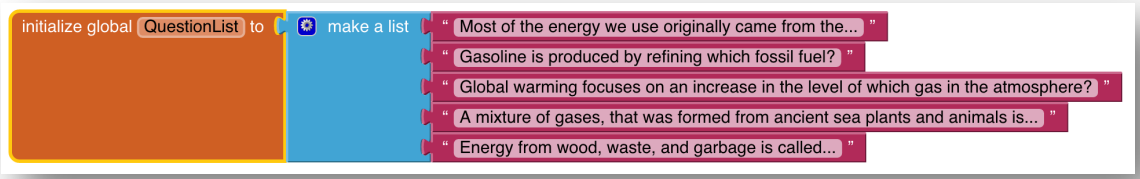
a) When the screen starts, one random question appears, including its corresponding image.
b) When the user clicks the Next, another question appears (after submit an answer!)
c) When the user answers a question, the app will report whether it is correct.

You'll code these behaviors one by one, testing as you go.

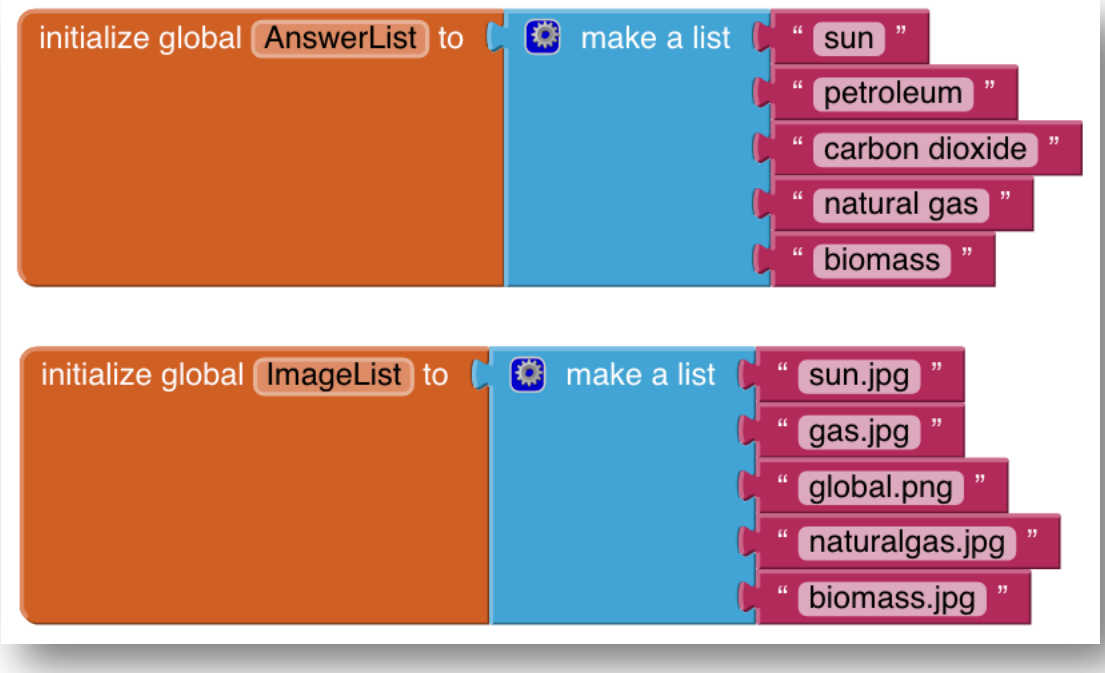
To begin, define one list variables based on the items QuestionList to hold the list of questions:



At the end you shoul have the QuestionList as this:

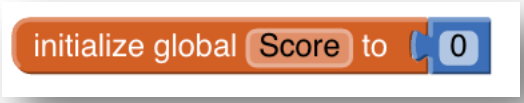


Now, you need to define an AnswerList and an ImageList. Make sure that the answer and the image is in the same position. You can duplicate the QuestionList block and change it:



The app needs to keep track of the current question as the user clicks the NextButton to proceed through the quiz. You'll define a variable named currentQuestionIndex for this, and the variable will serve as the index into both the QuestionList, AnswerList and ImageList.

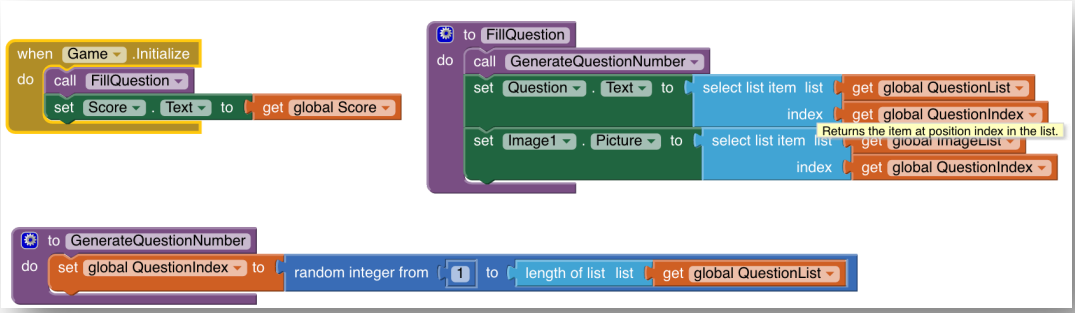
You'll define another variable named Scores to set and update the scores. At the begging of the game this variable will have 0.



We can define some rules:

correct answer +10 points
correct answer -10 points

When the screen "Game" is opened one random question should be presented. Let's see the code:

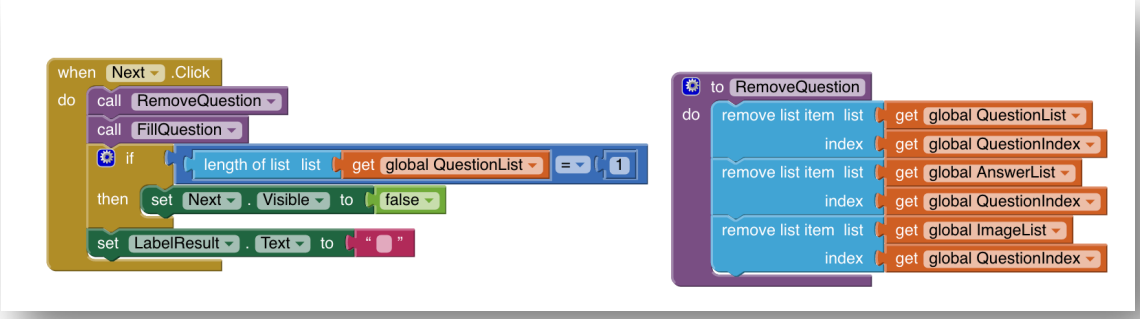


The principals to wrote this code are:

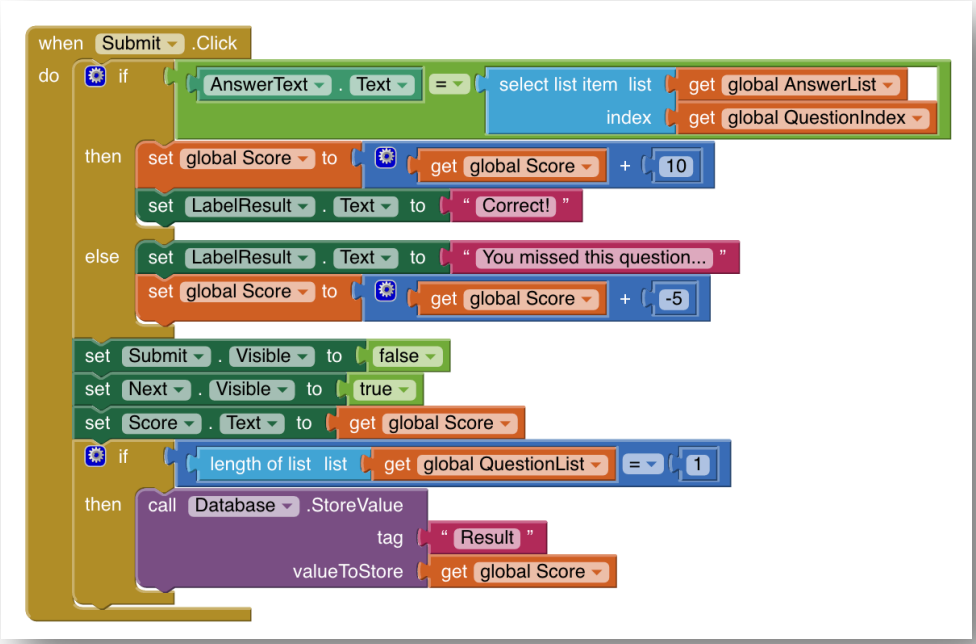
a) The variable globalQuestionIndex has a randon value tht varies from 1 to the total of elements in the list;
b) When the game starts, the procedure "Fill Question" will be executed and shows the question and the image that are in the position QuestionIndex.

However, the code will not work because the same QuestionIndex can be generated…

We must make sure the once one QuestionIndex in generated the correspondent question will be deleted.
We can do this when the user goes to the next question:



Now its time to check if the answer is the correct one and generate the next question:



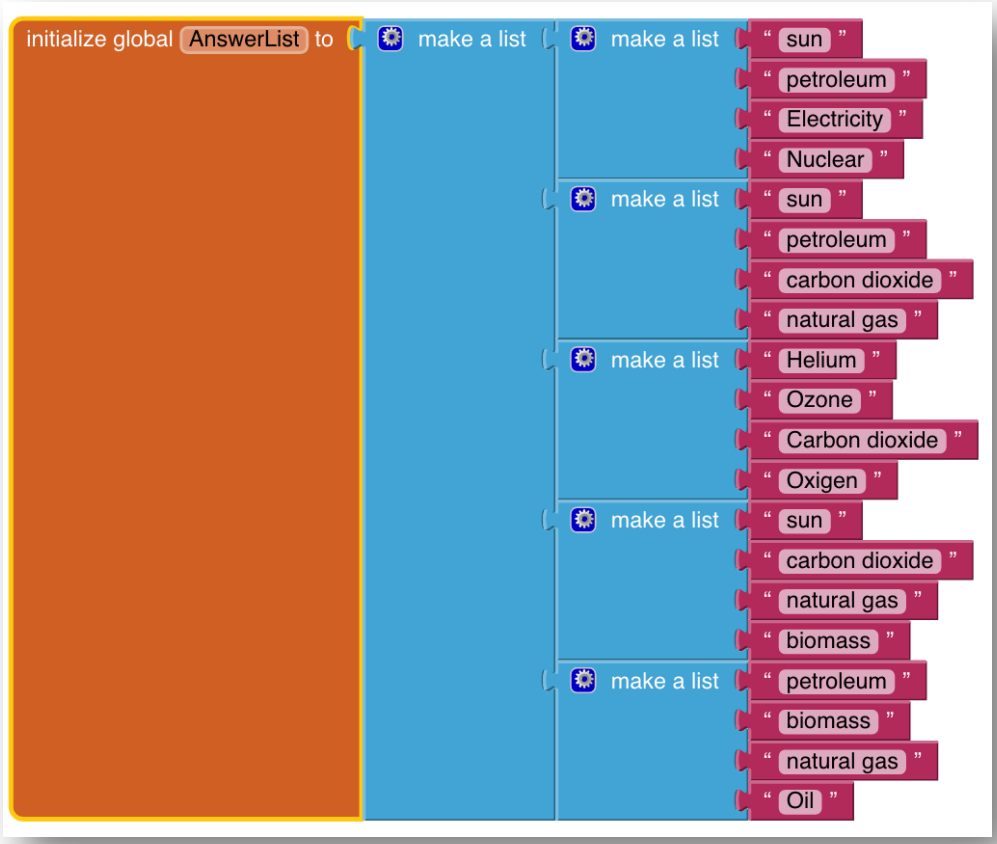
Test the App!

Task 3

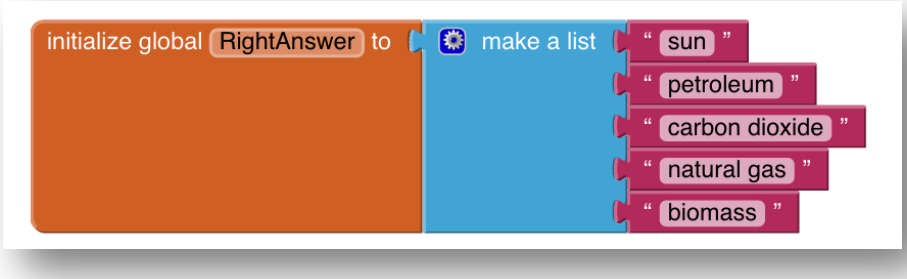
1- In this app the user can answer the same question several times. Make the changes so the user can answer only one time to each question.
Tip: Should both buttons (Submit and Next appear at the same time?)
2- Add 5 more questions to the quiz. Make sure the images have 300x300px.
3- Add sounds effects. https://freesound.org/browse/tags/sound-effects/
4- Add a chronometer to this app.
5- Save time into the database
6- Create the Screen "Scores" with the name, score and time.

Exercise 3 Part II – Energy Quiz

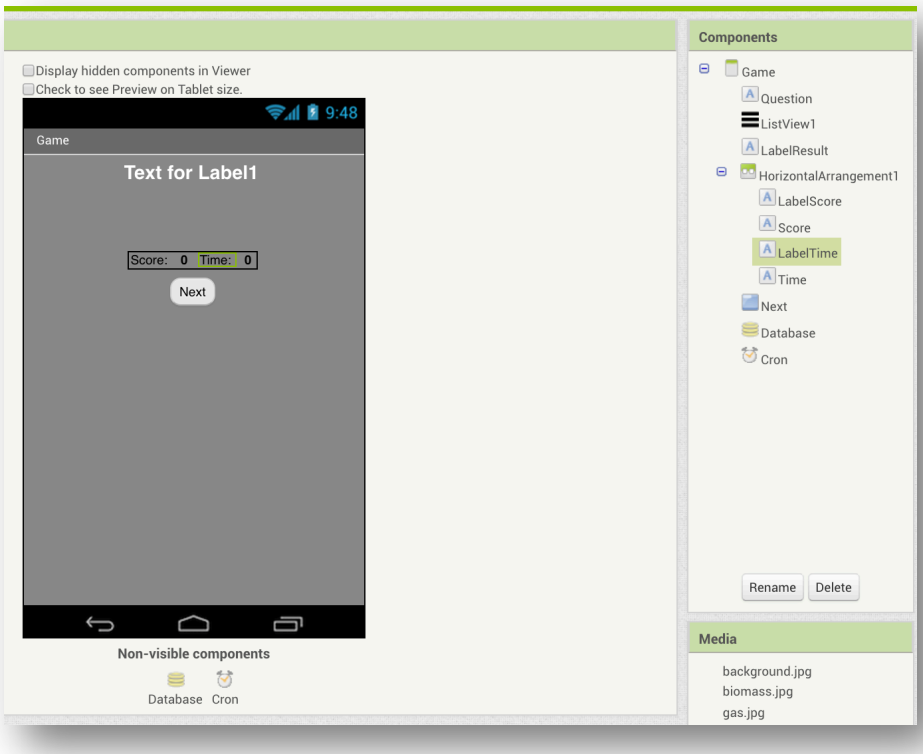
As you probably already realised, the previous quiz has some problems. The answer provided by the user must be exactly the same that is in the list in order to be correct – otherwise is incorrect. One possible solution is to make multi choice questions.
You can open the previous project and save as NewEnergyQuiz. There are a few changes to make!
First we must change the AnswersList, making a new list containing the options for each answer:



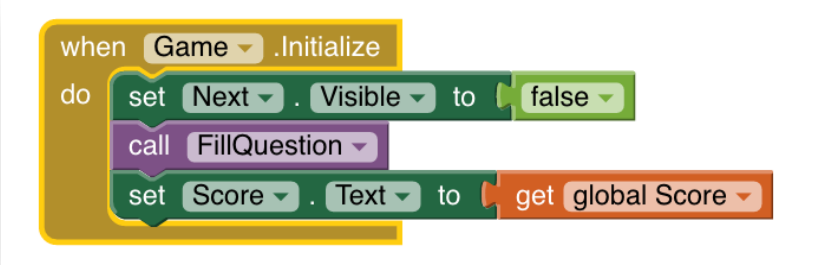
Now we have to create a new List, RightAnswer, where are the correct answers to the questions:



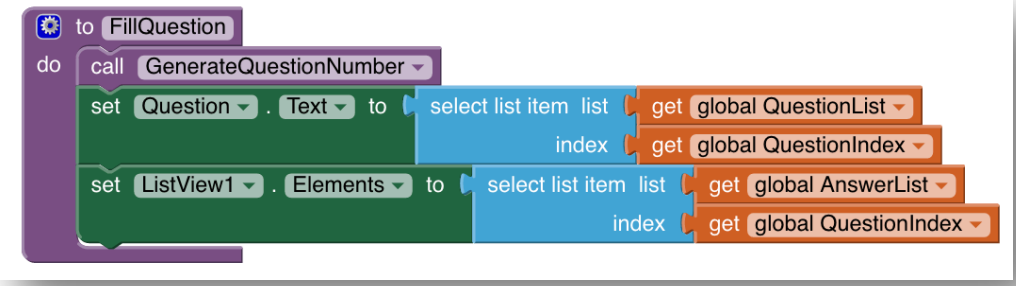
Change the Screen Game by adding/removing the components to look like this:



When the game starts (Game.Initialize) the actions to do are quite the same as the previous example:

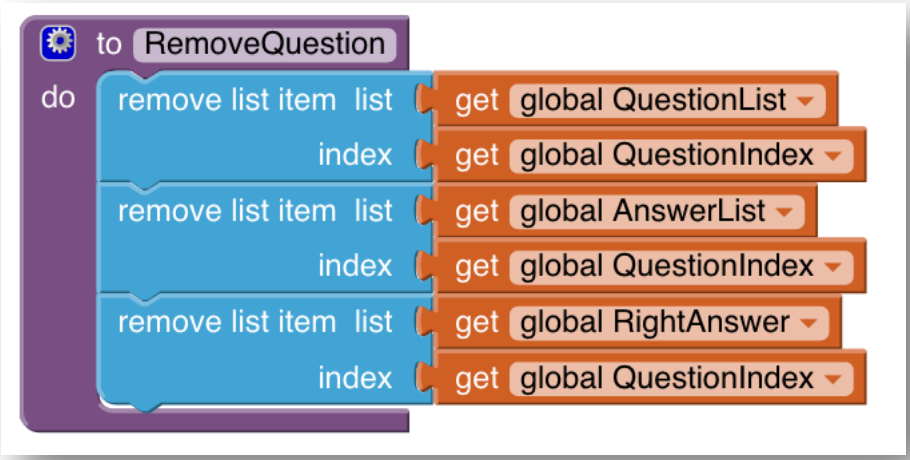


The procedure FillQuestion is very different:

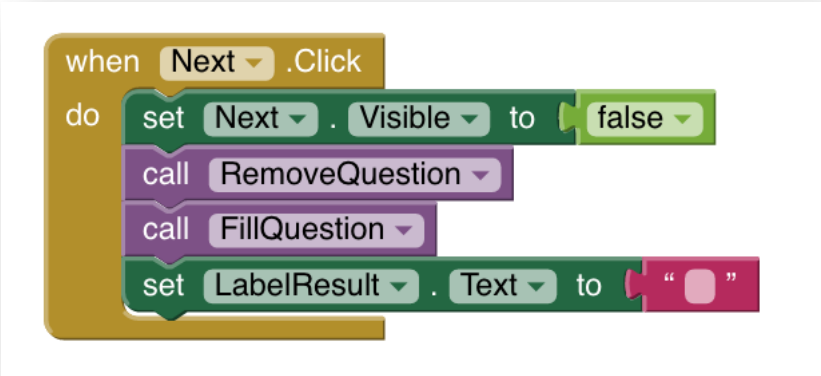


Each item in the variable answerList is itself a list containing four items. If you select an item from answerChoices , the result is a list.
The ListView component allow us to define the element of the list as a list.

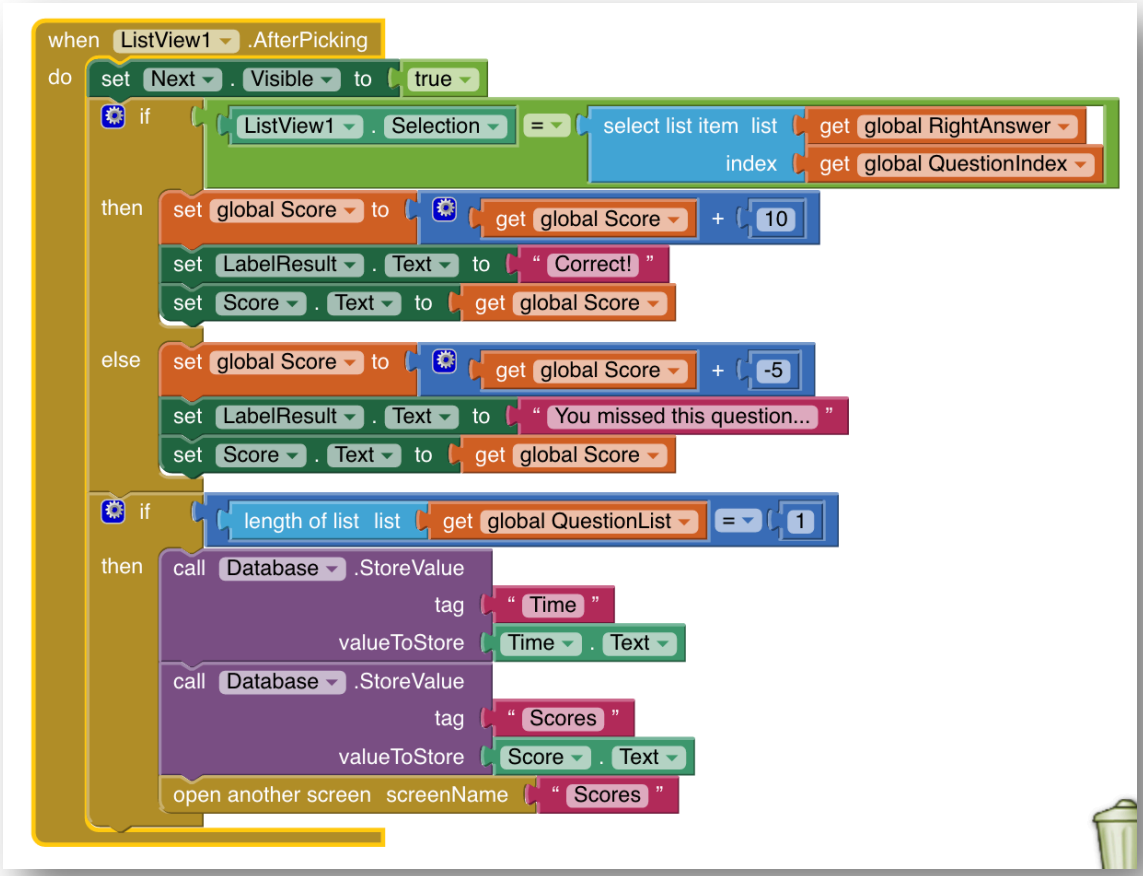
The procedure RemoveQuestion very similar to the other example:



The actions that should be executed when the button Next is clicked are also very similar to the previous example.



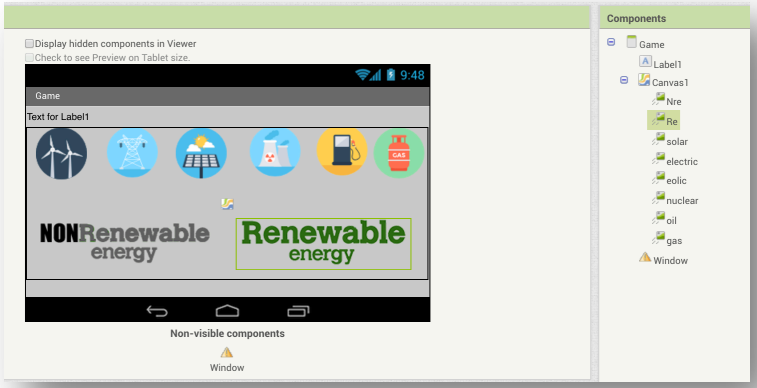
When the user select an item from the list corresponds to the event AfterPicking. Let´s see the code:


when ListView1 .AfterPicking
do set Next . Visible to true
if ListView1 . Selection = select list item list get global RightAnswer
index get global QuestionIndex
then set global Score to get global Score + 10
set LabelResult . Text to " Correct! "
set Score . Text to get global Score
else set global Score to get global Score + -5
set LabelResult . Text to " You missed this question... "
set Score . Text to get global Score
if length of list list get global QuestionList = 1
then call Database .StoreValue
tag " Time "
valueToStore Time . Text
call Database .StoreValue
tag " Scores "
valueToStore Score . Text
open another screen screenName " Scores "

Test the App!

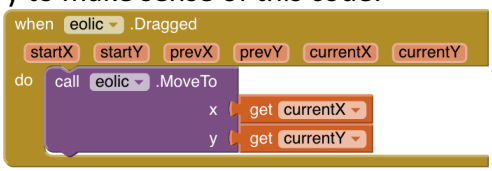
Practice - Types of energy

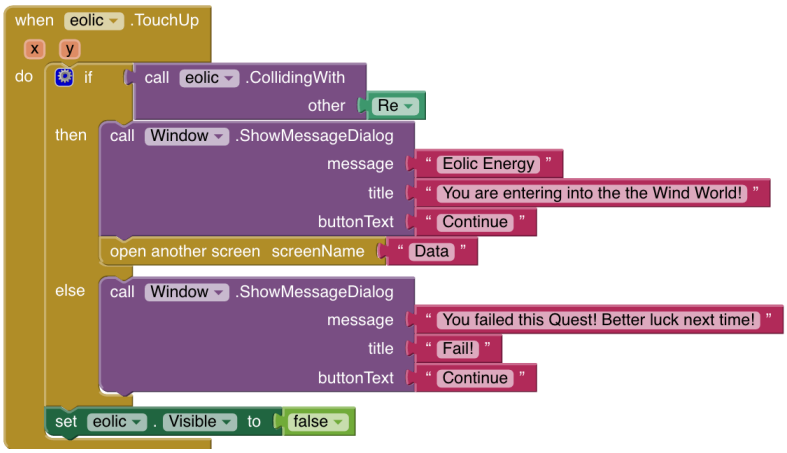
You want to create an app which purpose is to educate the player about types of energy. The game consists on dividing 6 icons that represent 6 different types of energy. From there it´s very simple: if you get the answer right, lets say you drag the solar panel to "renewable energy", you´ll be redirected to a quiz or other activity about that type of energy, and be awarded points based on your performance. If you fail, the icon you dragged disappears and you lose your chance to win points. Time is also a factor in this app.



Some considerations:

a) Try to make sense of this code:





b) You will need at least 3 screens: intro, game (above), data (quiz) and scores;
c) The media files are in Google Drive.
d) Don't forget to use sound effects!
e) You can make your own rules!
f) Above all, you will need your imagination - and of course what you learned from the previous exercises!

References

Hsu, Y. C., Rice, K., & Dawley, L. (2012). Empowering educators with Google's Android App Inventor: An online workshop in mobile app design. *British Journal of Educational Technology*, *43*(1), E1-E5.

http://explore.appinventor.mit.edu/ai2/get-gold

http://puravidaapps.com/snippets.php