

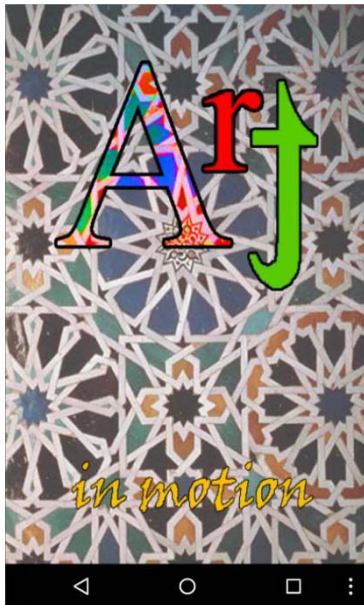
**ArtInMotion**

Content

- 1. ArtInMotion user view ..... 3
- 2. Designing and blocking the screens ..... 5
  - 2.1. Screen 1..... 5
  - 2.2. Screen 2..... 7
  - 2.3. Screen 3..... 9
  - 2.4. Screen 4..... 13
- 3. All blocks..... 28

# 1. ArtInMotion user view

ArtInMotion is an App developed to improve our knowledge about art and architecture. It has two different options that you can choose in first screen (picture 2), after splash screen (picture 1), shown by the details of two pictures by “El Greco”, Doménikos Theotokópoulos.

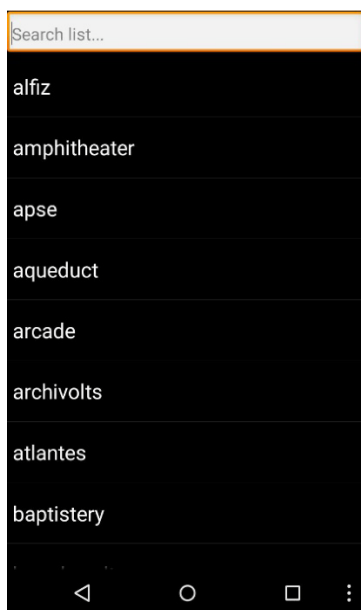


Picture 1: Screen 1 - Splash screen

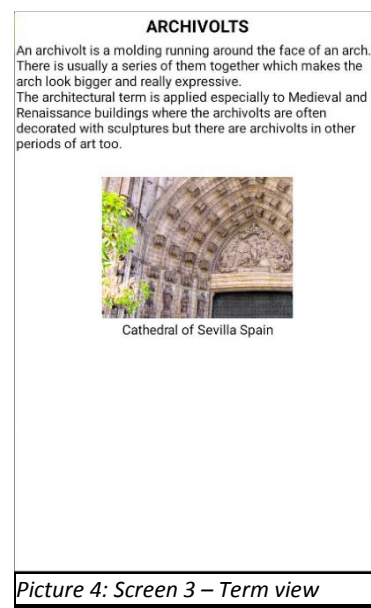


Picture 2: Screen 2

Choosing Art and Learn, you reach a list view of 36 different terms related to art (picture 3). Clicking each one of them you get an explanation about the term, references and a picture that has been taken by the Spanish team in the p@ls project (picture 4).



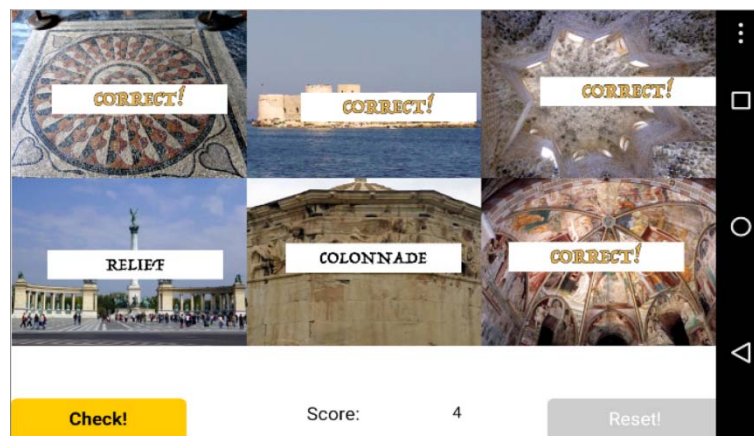
Picture 3: Screen 3 - ListView



Picture 4: Screen 3 – Term view

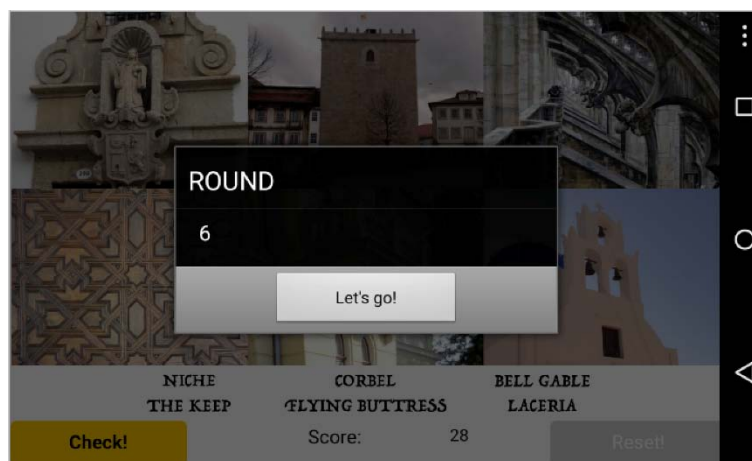
On the other hand, if the user chooses Art and play, positioning the phone horizontally, you get into a quiz game, where, in each round (from six), six pictures are shown and you are asked to drag and drop the right term onto the correspondent picture. Then, with all the labels

collocated, clicking in Check button it takes you to next round, checking how many terms you have placed in the correct picture and adding your points to the scoreboard (picture 5).



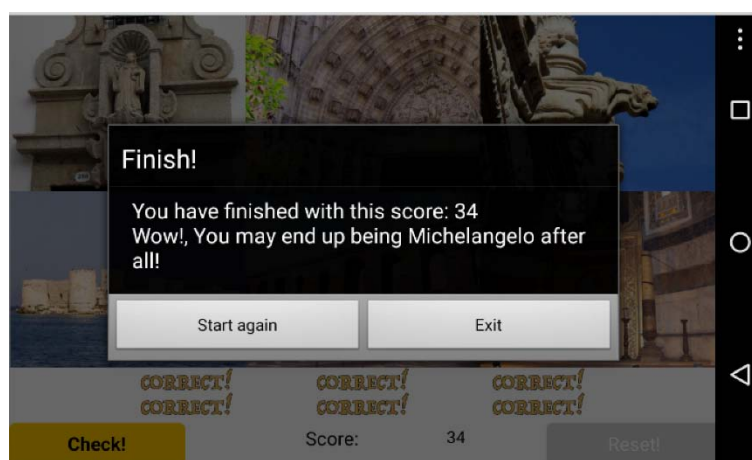
Picture 5: Screen 4 – Checking view

After thirty pictures, you are in the final round (picture 6):



Picture 6: Screen 4 – Final round view

And after that, the quiz is ended and the user gets the final score with a different motto depending on the score (picture 7).



Picture 7: Screen 4 – Finish view

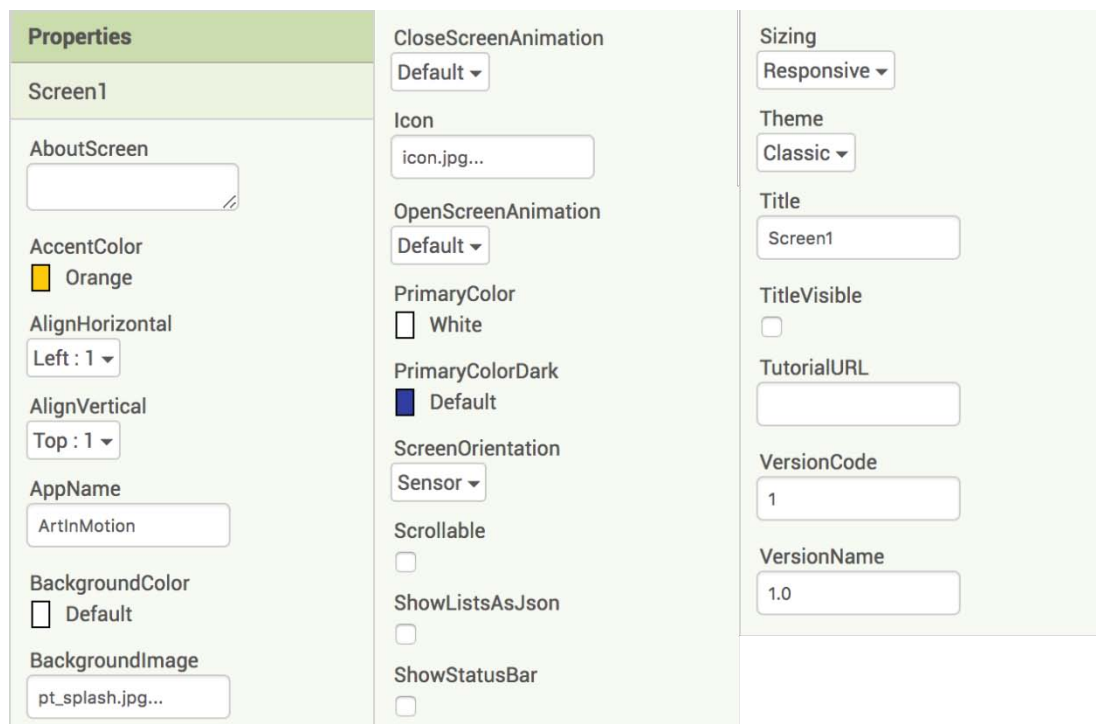
## 2. Designing and blocking the screens

### 2.1. Screen 1

#### A. Designer

This screen is the splash screen, which is shown for one second before entering in the first screen where user can interact. It only has a clock as a countdown, meanwhile a jpg picture related with project (a Mudejar skirting board from the Alcázar of Seville which has been touched up) is shown as a presentation.

Properties of this screen are:



Properties	Animation	Sizing/Theme
Screen1	CloseScreenAnimation Default	Sizing Responsive
AboutScreen	Icon icon.jpg...	Theme Classic
AccentColor Orange	OpenScreenAnimation Default	Title Screen1
AlignHorizontal Left : 1	PrimaryColor White	TitleVisible <input type="checkbox"/>
AlignVertical Top : 1	PrimaryColorDark Default	TutorialURL
AppName ArtInMotion	ScreenOrientation Sensor	VersionCode 1
BackgroundColor Default	Scrollable <input type="checkbox"/>	VersionName 1.0
BackgroundImage pt_splash.jpg...	ShowListsAsJson <input type="checkbox"/>	
	ShowStatusBar <input type="checkbox"/>	

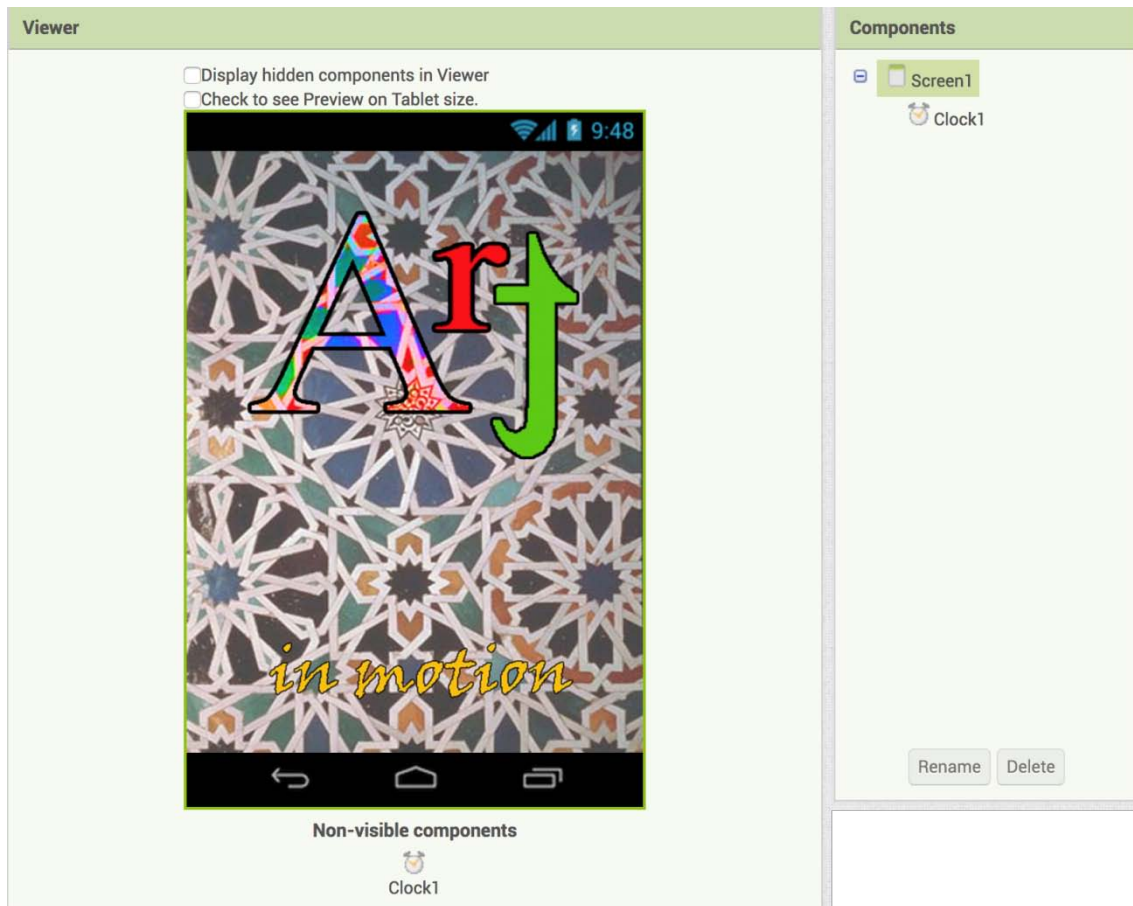
Picture 8: Screen 1 – Properties

As it can be seen, because this is the main screen, App name (ArtInMotion), icon (icon.jpg), screen orientation (sensor), sizing (responsive) and theme (classic) are defined here. In that way, medias which are uploaded are:

Icon.jpg as an icon square picture

pt\_splash.jpg as background of splash screen in portrait position

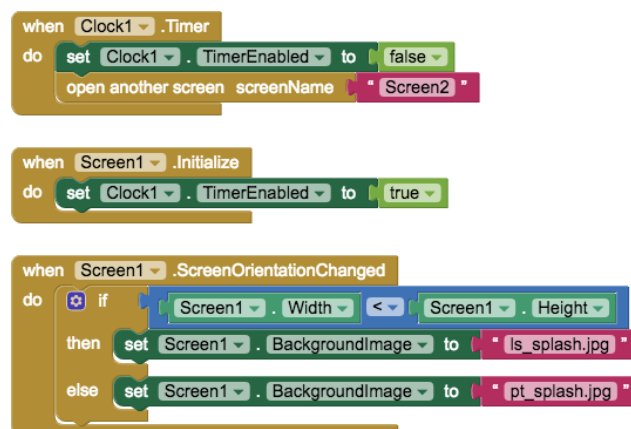
ls\_splash.jpg as background of splash screen in landscape position



Picture 9: Screen 1 – Design

## B. Blocks

As it is shown in the next picture, we just enable Clock1 timer when screen 1 is initialized, and also, if screen orientation is changed while splash screen is showing, background picture is changed (from pt\_splash.jpg to ls\_splash.jpg and vice versa) to adequate aspect ratio to the screen.



Picture 10: Screen 1 – Blocks

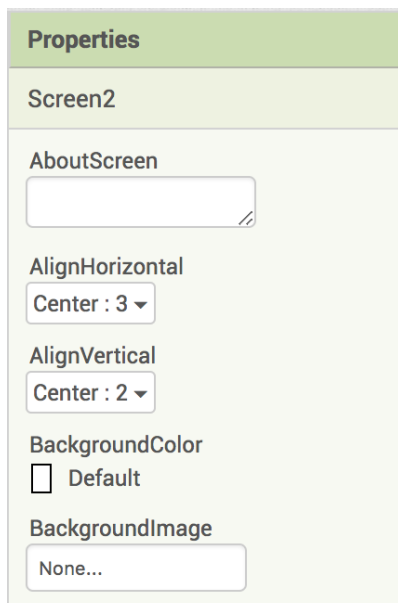
Finally, when the timer ends, screen 2 is opened.

## 2.2. Screen 2

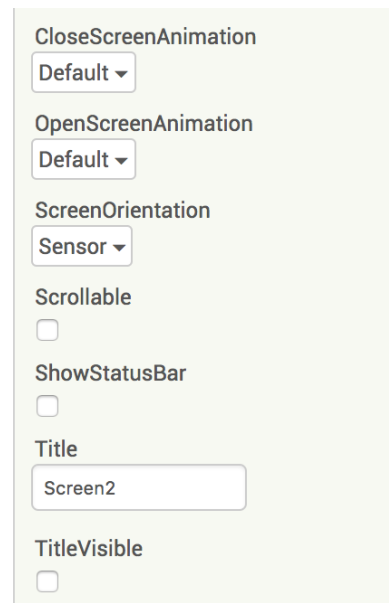
### A. Designer

In this screen options are displayed. The user can choose between two options, one of them takes the app to screen 3 where a list of terms, descriptions and pictures are displayed and the other to screen 4, which goes directly to the quiz. Also it has a handler if back button on the phone is pressed, that takes the user to a directly exit option.

Properties for this screen are:



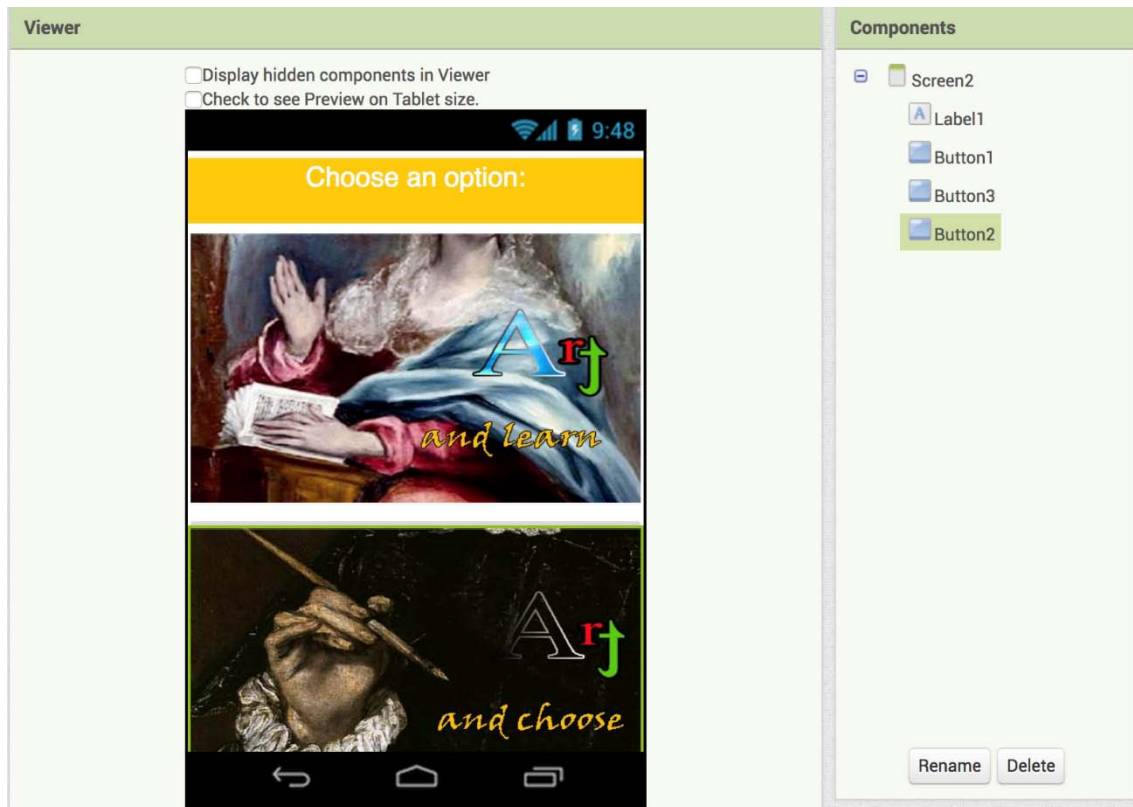
Picture 11: Screen 2 – Properties 1



Picture 12: Screen 2 – Properties 2

As it can be seen, we use here 4 components: label1 to display the explicatory sentence “Choose an option”, one button with learn.jpg background and other with choose.jpg background and another button for direct exit.





Picture 13: Screen 2 – Design

Properties for each button are:



Picture 14: Screen 2 Button 2



Picture 15: Screen 2 Button 3



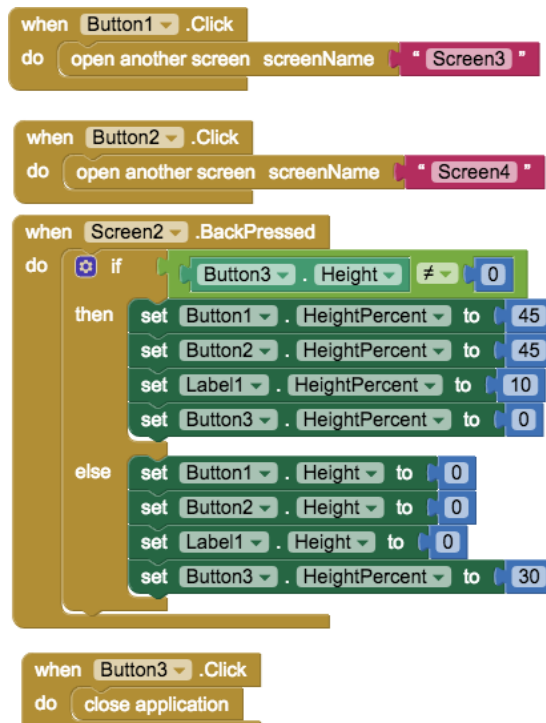
Picture 16: Screen 2 Button 1



## B. Blocks

This screen has a very simple logic, just two buttons to choose an option:

- Button 1 click event handler: Clicking Button1 Screen3 is opened and list of terms is displayed.
- Button 2 click event handler: Clicking button2 Screen4 is opened, the quiz is displayed.
- Phone Back Button pressed event handler: if the user press back button on the smart phone and it was the first time that it was pressed, we make button1, button2 and label1 disappear and we show button3, which if it now it is clicked, app is closed (with a Button 3 click event handler).



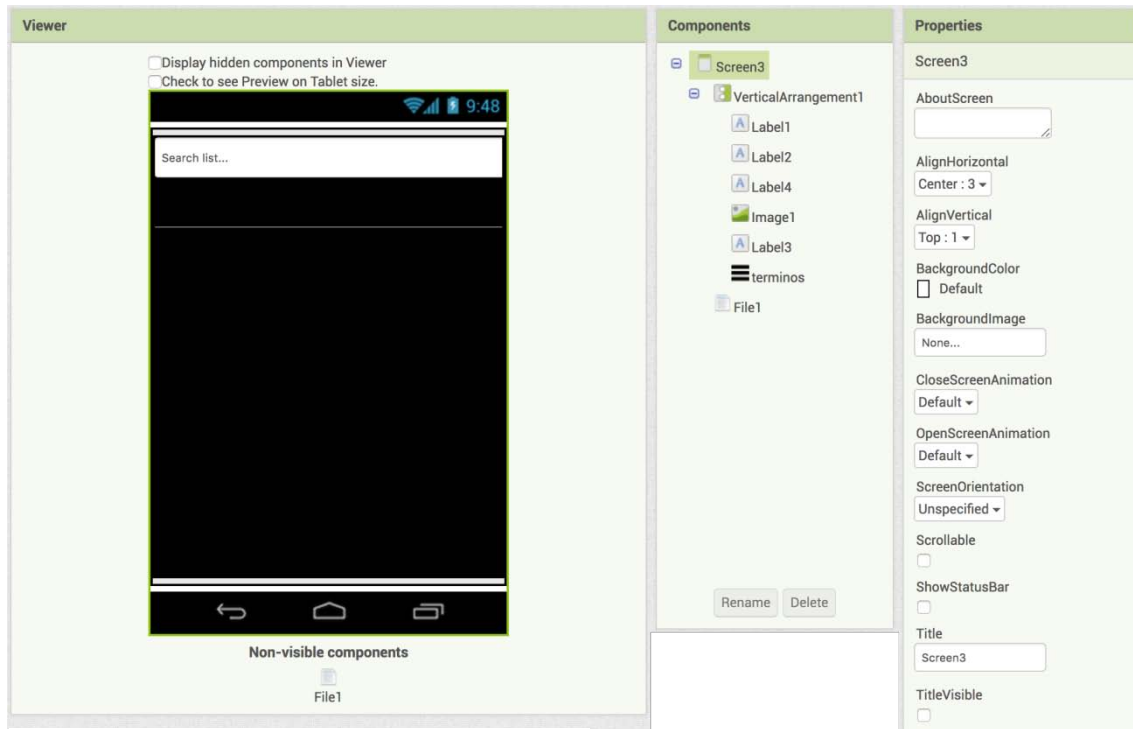
Picture 17: Screen 2 Blocks

## 2.3. Screen 3

### A. Designer

Now we are going to explain the two most complex screens. Firstly we would like to clarify that all data that is used, term, descriptions, pictures and legends are imported from a .csv file, that is to say, that they can be changed easily, in order to change terms, descriptions and pictures or simply increase elements and rounds. In this option, Art and Learn, a listview is shown with 36 terms, descriptions, pictures and legends imported from glossary.csv file.

Anyway media has to be uploaded, and in this case we need glossary.csv and all pictures, whose names are included in the .csv file.

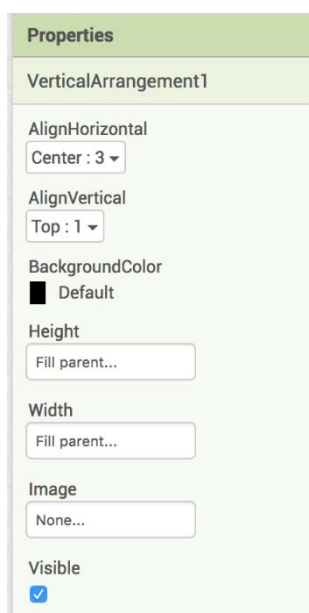


Picture 18: Screen 3 Design, components and properties

As it is shown, a vertical arrangement has been added, to display all components one above the next one.

The list view called *terminos* is displayed at the beginning, but the rest of components are hidden. When a term is selected, we hide the list view and show the other components to display labels of texts (term name, description and legend) and a picture which contains the correspondent image uploaded. If back button is pressed all elements are hidden but list view *terminos* is unhidden and shown again.

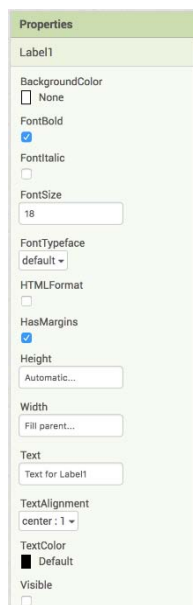
Properties of each components have been configured as shown in following pictures:



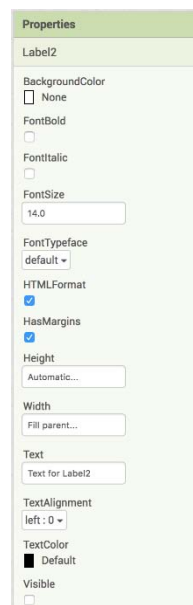
Picture 19: Screen 3- Vertical arrangement properties



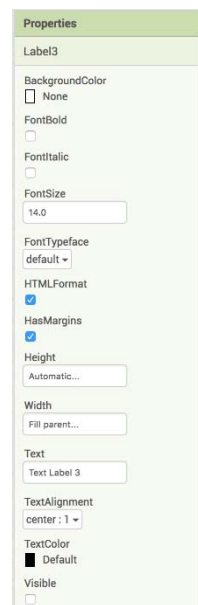
Picture 20: Screen 3-ListView properties



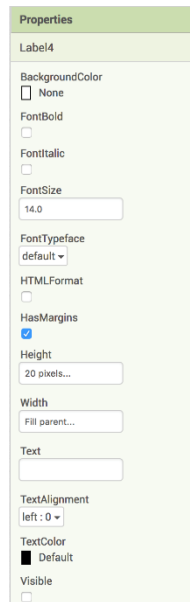
Picture 21: Screen 3-Label1 properties



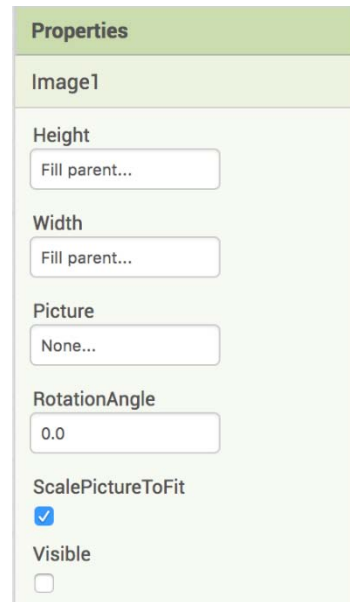
Picture 22: Screen 3-Label2 properties



Picture 23: Screen 3-Label3 properties



Picture 24: Screen 3-Label4 properties



Picture 25: Screen 3-Image1 properties

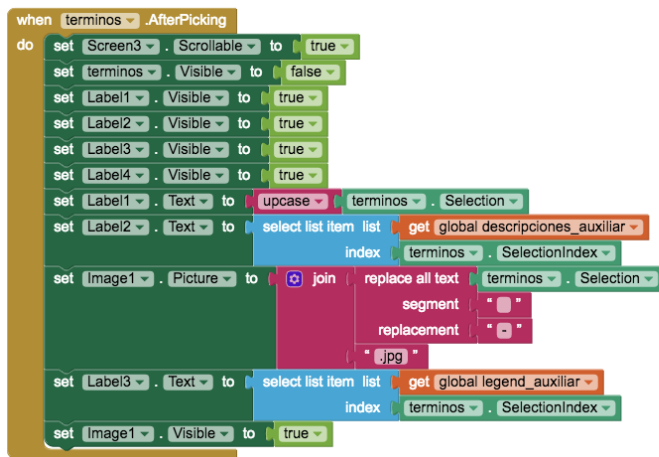
Label 1 is used for term tittle, label 2 for all the texts in the description, label 3 is used for the legends just under the picture and label 4 is a resource to have a blank space between text and picture.

## B. Blocks

In this case, blocks are also simple, but with the difference of getting data from the .csv file, as it is shown here:





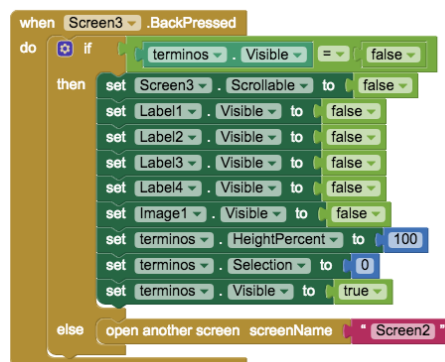


Picture 27: Screen 3- Showing term info after picking one from list

As we use the same screen that before, we have to hide the list component and unhide the others (all the labels and the picture). We also make screen 3 scrollable to make possible scroll down to read the whole text, be able to see the picture and legend.

At the end, we assign to each label the correspondent text from each auxiliary list, changing term text to uppercase and conforming the name of the picture adding .jpg at the end of term name.

Finally, we handle pressing back button in the phone, to take user to the list view again, just making visible *terminos* list and hiding all labels and picture. If back button is pressed when user already is in list view of this screen, app takes user to the previous screen (screen 2).



Picture 28: Screen 3- Back button event handler

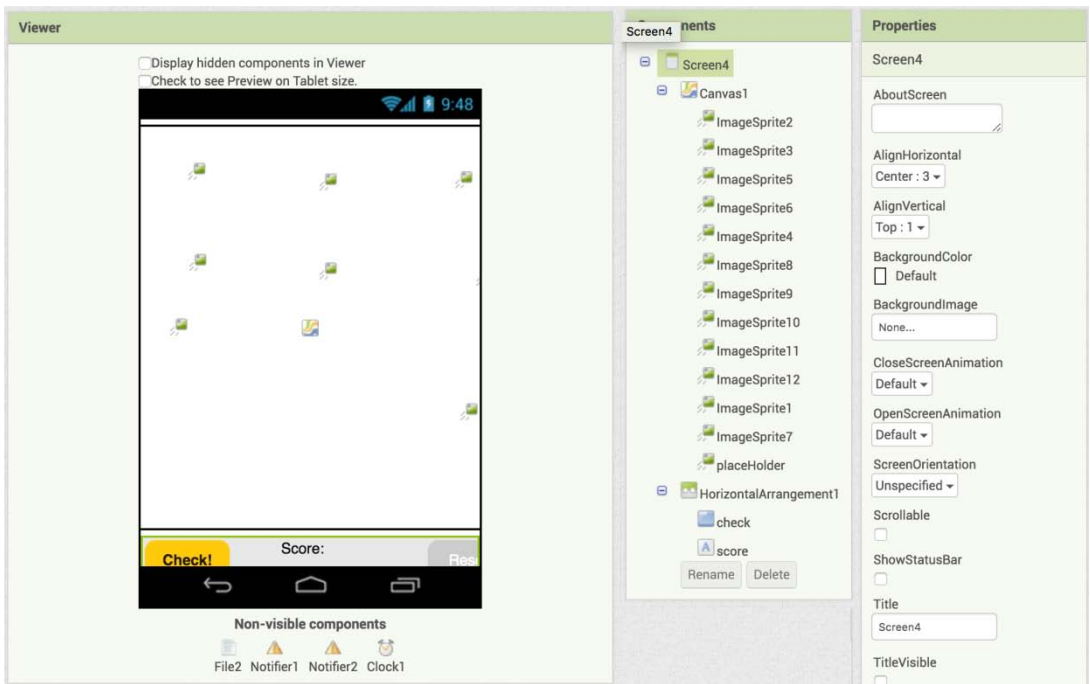
## 2.4. Screen 4

### A. Designer

This is definitely the most complex screen of the App. As it is shown in the next picture, a canvas is used to manage interactions between images sprites, occupying 90 percent of screen. The rest 10 percent is used, with a horizontal arrangement, to display the Check button, Reset button and the scoreboard.

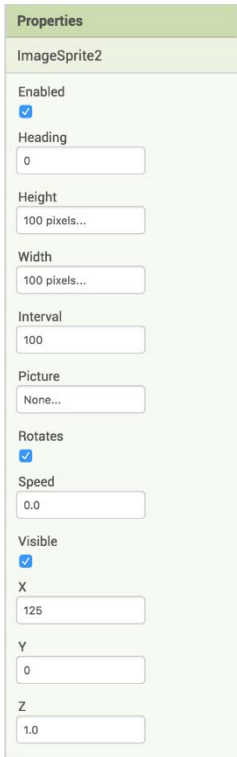
We also have included 4 non visible elements: the same .csv file as in screen 3, 2 notifications to show messages during the game, and a clock which makes a countdown while answers are being checked.

Screen design and components can be seen here:

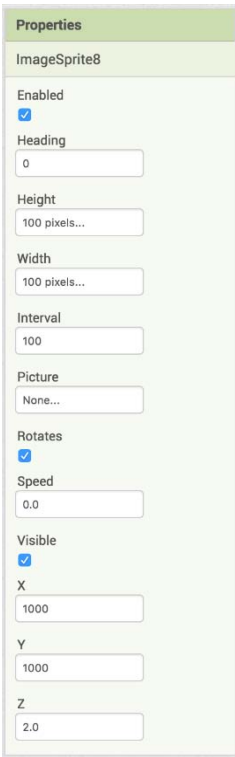


Picture 29: Screen 4- Design, components and screen properties.

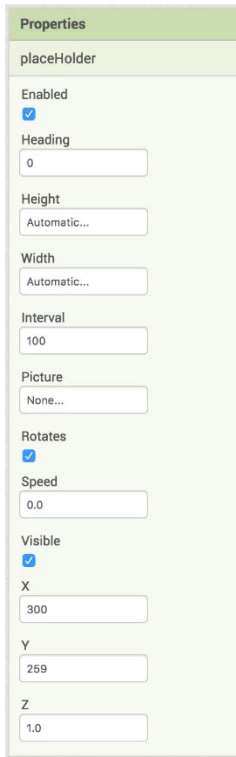
Six images sprites are used with term pictures as background. Other six images sprites are used with term labels as background, so events are detected when an image sprite is dragged and dropped onto other image sprite, colliding between them. An auxiliary image sprite, called placeholder, is included and used to avoid dragging two labels at the same time. Properties of the three kinds of images sprites are shown in the following pictures:



Picture 30: Screen 4- Picture imageSprite properties.



Picture 31: Screen 4- Label imageSprite properties.



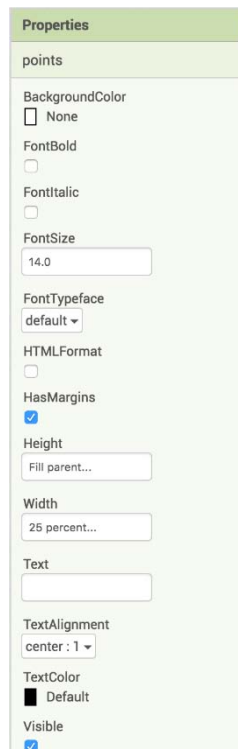
Picture 32: Screen 4- Auxiliary imageSprite properties.



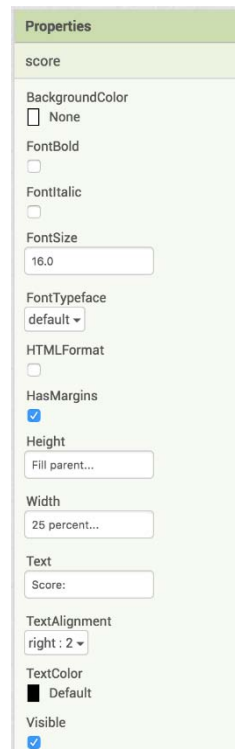
In the same way, for labels and buttons, properties are:



Picture 33: Screen 4-Reset button properties.



Picture 34: Screen 4-points label properties.



Picture 35: Screen 4- Score board label properties.



Picture 36: Screen 4-Check button properties.

These labels are used to count points and accumulate them when labels are placed onto the right picture. We have also included two buttons, one of them to let user check correct answers and pass to the next round (Check button). The other button is a reset, which can be clicked any moment during the game to reset the quiz to the round 1. There are six rounds. At the beginning of each one, a message telling that you are in that round is displayed. Also, at the end of the game (after round 6), a message is displayed with all the score reached and a motto related to art and correspondent to the score. In this moment, the message also gives user the option of starting the quiz again or exit the app.

## B. Blocks

As we told before, this is the most complex screen in our app.

At the beginning we start defining many variables that will help us to manage all the information and logic:



Picture 37: Screen 4-  
Variables defined

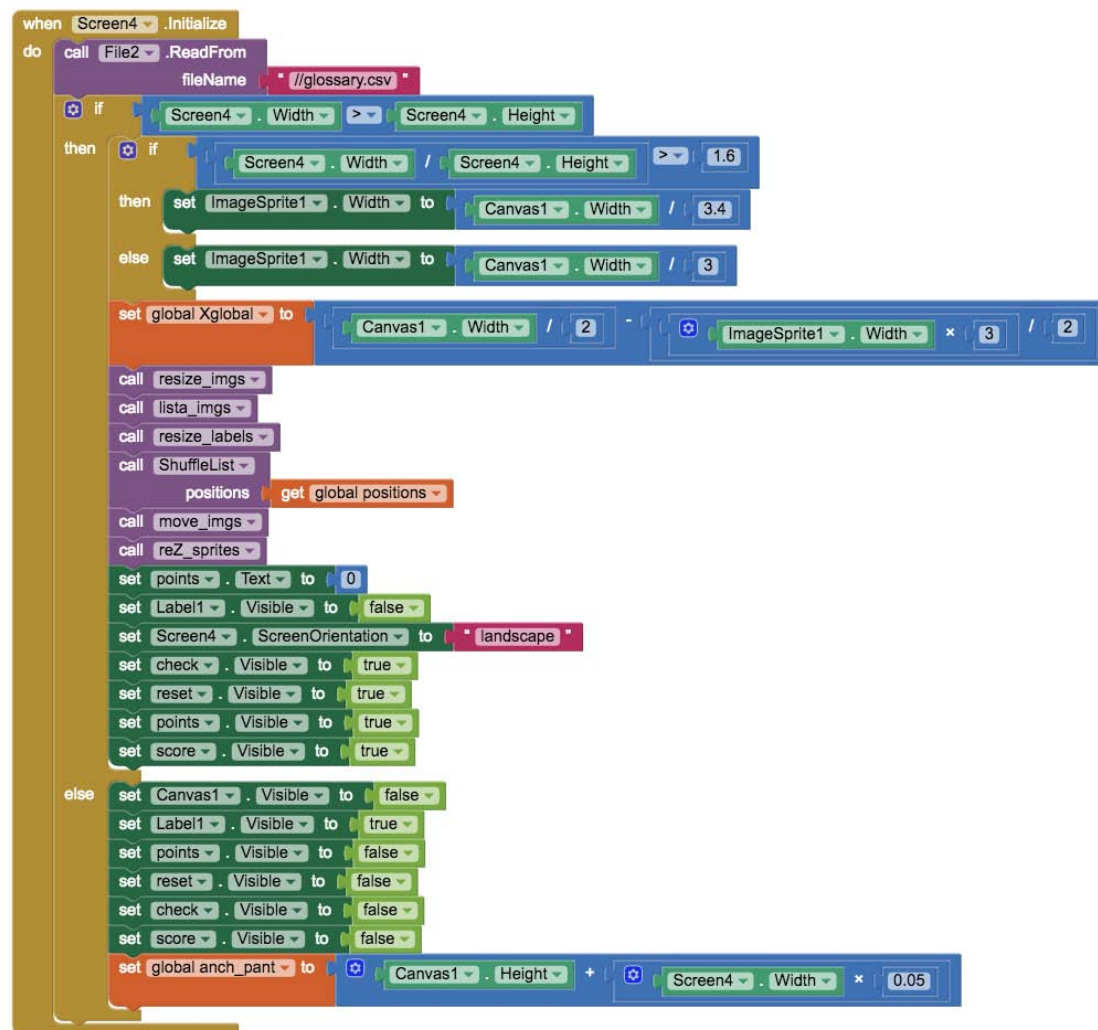
In general, all variables like *Srpi\_NN\_ori\_X* and *Srpi\_NN\_ori\_Y* are defined and used to store initial positions of label image sprites, in order to be able to take them to that position when reset button is pressed and when next round is started. Also, list called *positions*, is created to store all positions (coordinates x and y) of pictures and labels, depending on device screen size, in order to shuffle the picture positions, to avoid the game be so easy repeating always same solutions.

Lists called *terminos*, *glosario* and *terminos\_auxiliar* are used to store terms and pictures from .csv file.

A variable called *anch\_pant* is created to store device screen width. The app centres all the pictures in the device screen and gives a width and a height to each picture depending on this value, which is stored in *anch\_pant*.

Variable called *round* stores the current round between values 1 and 6.

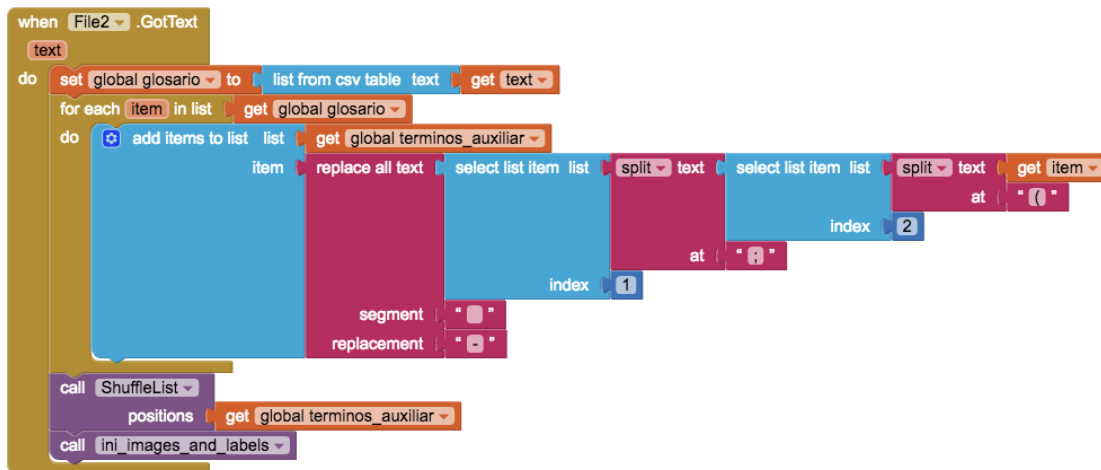
Now, we are going to explain initialization block, shown in picture 38:



Picture 38: Screen 4- Screen initialization

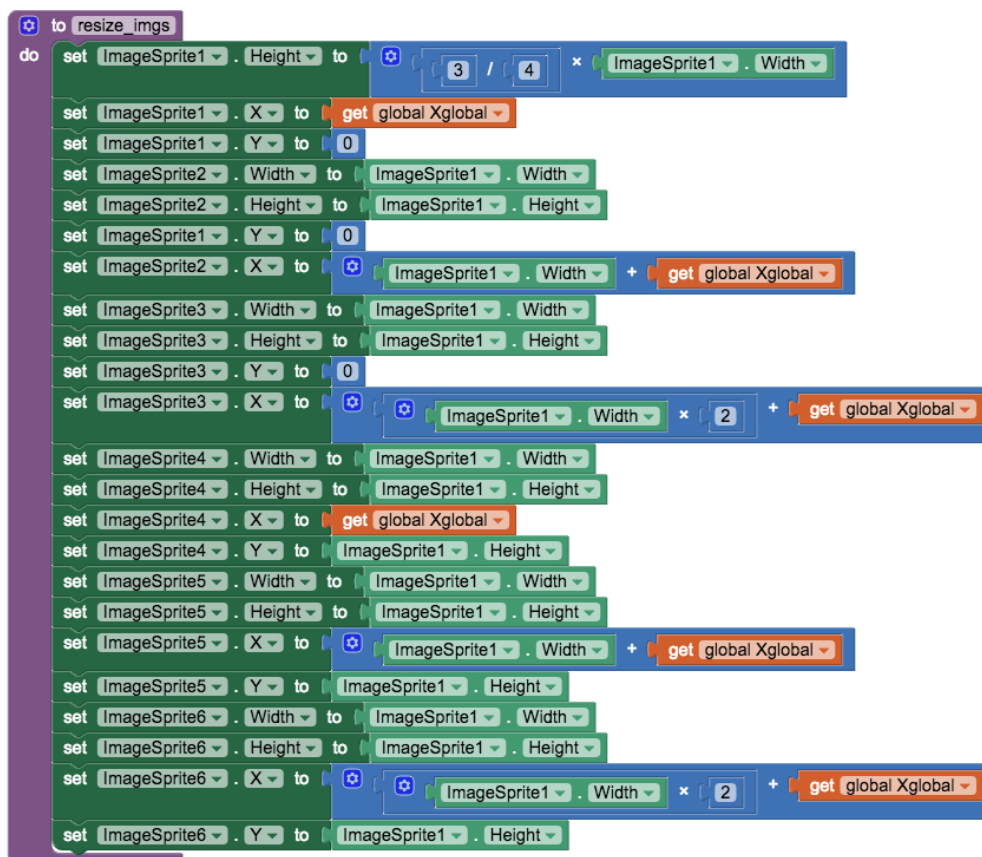
First thing that is done is to read .csv file from phone storing, as we did in screen 3. We use here the same file *glossary.csv*, and everything is done the same that before, storing all information in some lists in order to have the 36 terms and pictures in a list called *global glosario*. Then, as before, we split each row to eliminate parenthesis and semicolons, and the “clean” information to store it in a definitive list called *terminos\_auxiliar*. To avoid always playing the same game, because we always use the same .csv file, a shuffle function called *ShuffleList* is used. In that way, we shuffle every row coming from the file and every execution of our app is different.

Finally, *ini\_images\_and\_labels* is called to initiate each round (here the first one), we will explain it later.



Picture 39: Screen 4- getting data from .csv file

After reading data, we check if the device is in horizontal position (we just check if the screen width is bigger than the screen height or not), the only position in which the game can be played. If the device is in horizontal position, using screen width the app calculates the optimum width for all pictures and labels in order to occupy most of the screen. Just after that, X position of the first sprite is defined and stored in *Xglobal* variable. Now procedure *resize\_imgs* (picture 40) is called to set width, height and position in the screen of every picture image sprite, later the same is done with labels calling procedure *resize\_labels* (picture 42).



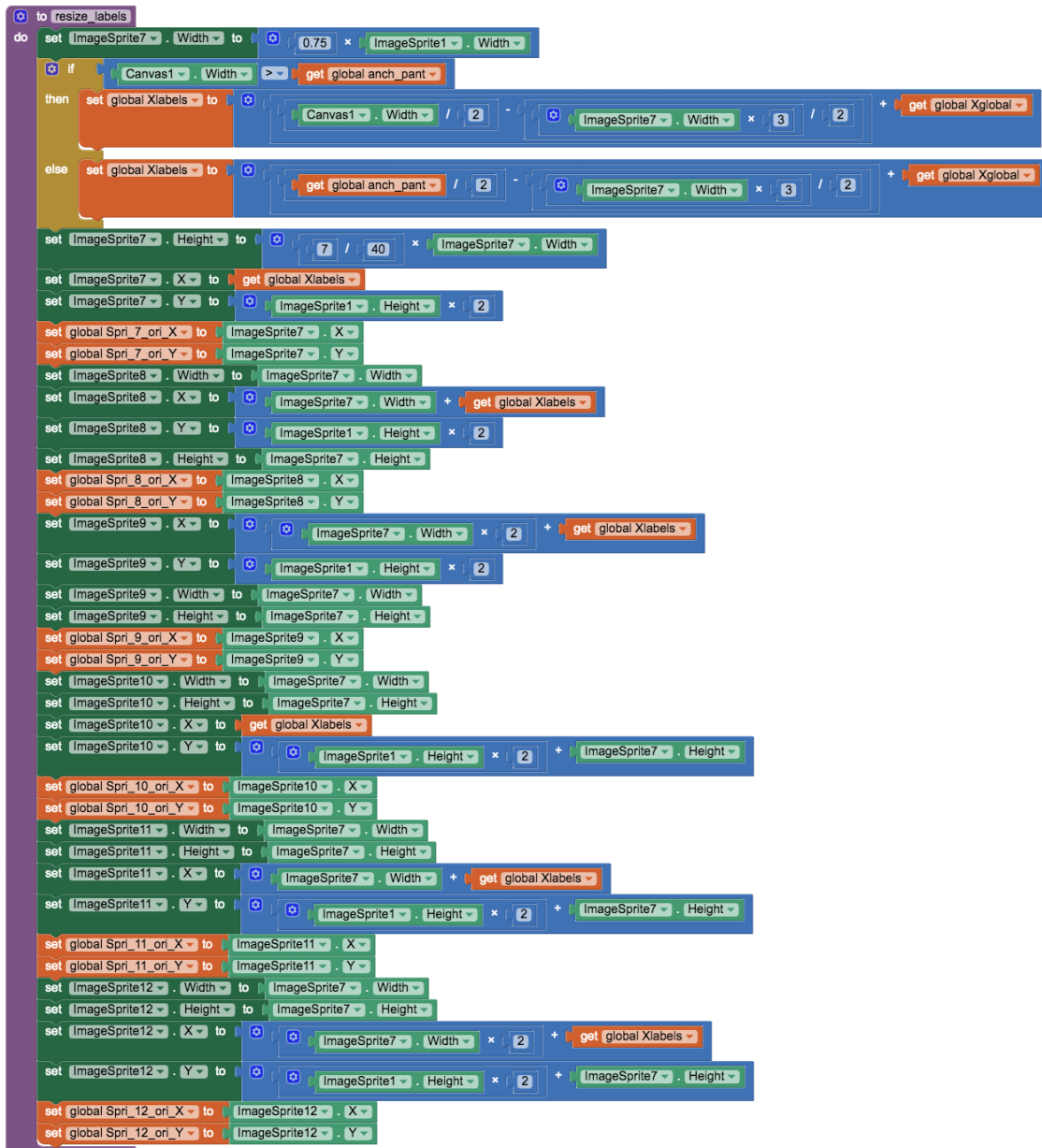
Picture 40: Screen 4- Resizing pictures for image sprites.

In this procedure, from variable Xglobal as initial X position, all positions for every picture image sprite is set, building a grid of 3x2 (picture 41).



Picture 41: Screen 4- pictures grid

Then, the same is done with labels, using X and Y position from the first picture image sprite. Because the labels are smaller, we reduce width and height from pictures and centre all the label group in screen (picture 42).



Picture 42: Screen 4- setting positions and sizes for labels.

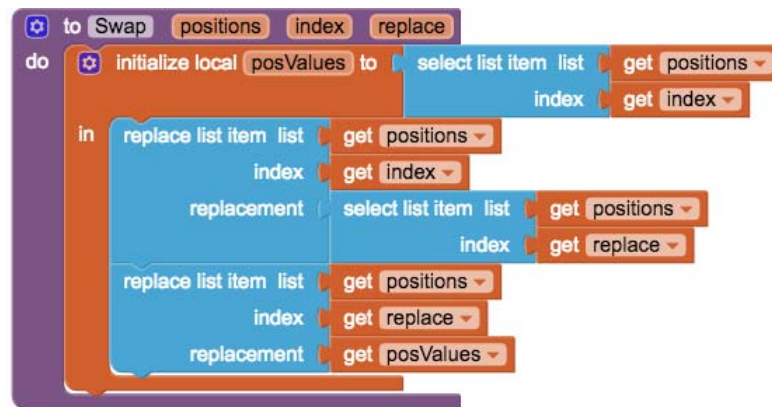
Now it is time to shuffle positions in order to move pictures for every time that a round is started. It is done with procedure called *ShuffleList*, using a random position each time (picture 43).



Picture 43: Screen 4- Shuffling positions for labels.

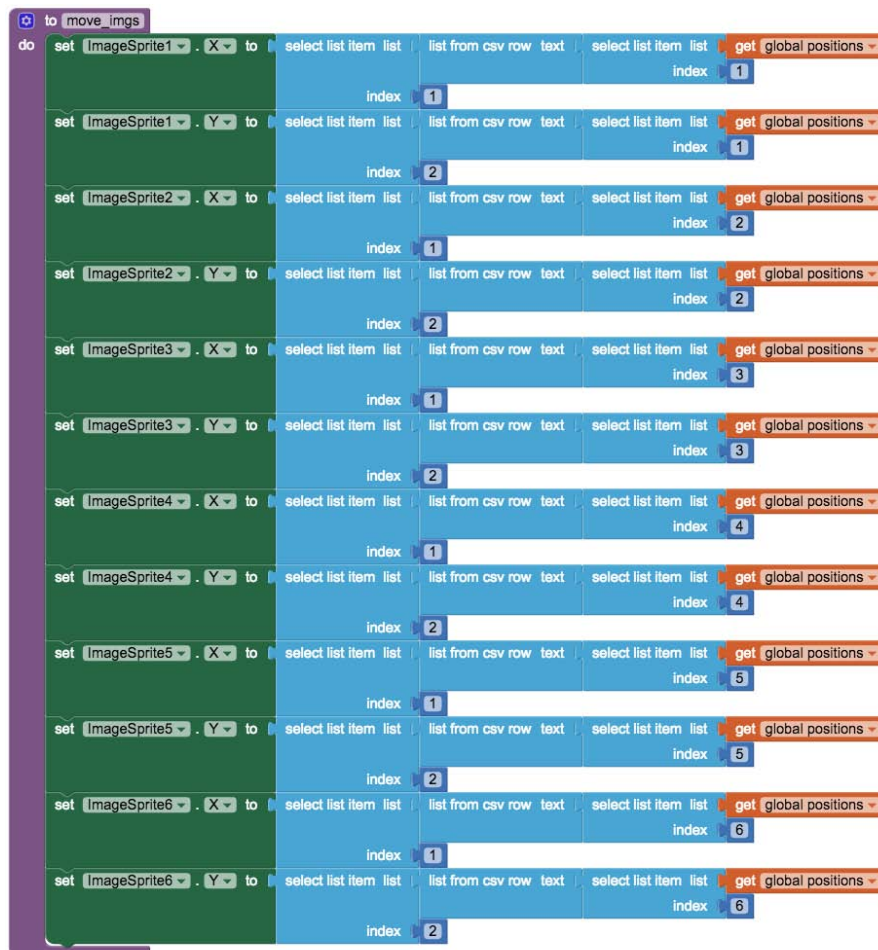


Inside *ShuffleList* procedure, *Swap* procedure is called, to make a replacement between elements in list using an auxiliary list (picture 44).



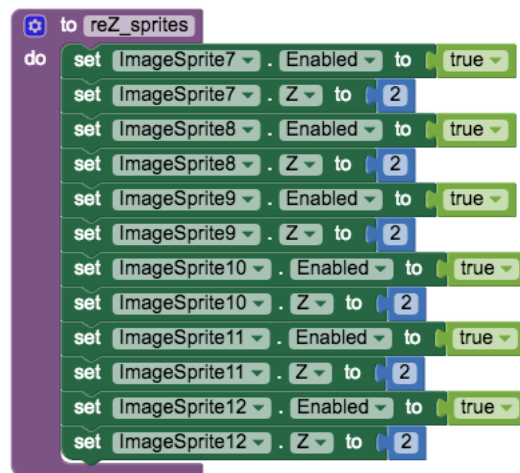
Picture 44: Screen 4- Swap procedure

After shuffling positions and getting all new positions in *global\_positions* variable, all picture positions are re-assigned (picture 45) using procedure *move\_imgs*.



Picture 45: Screen 4- Re-assigning picture positions

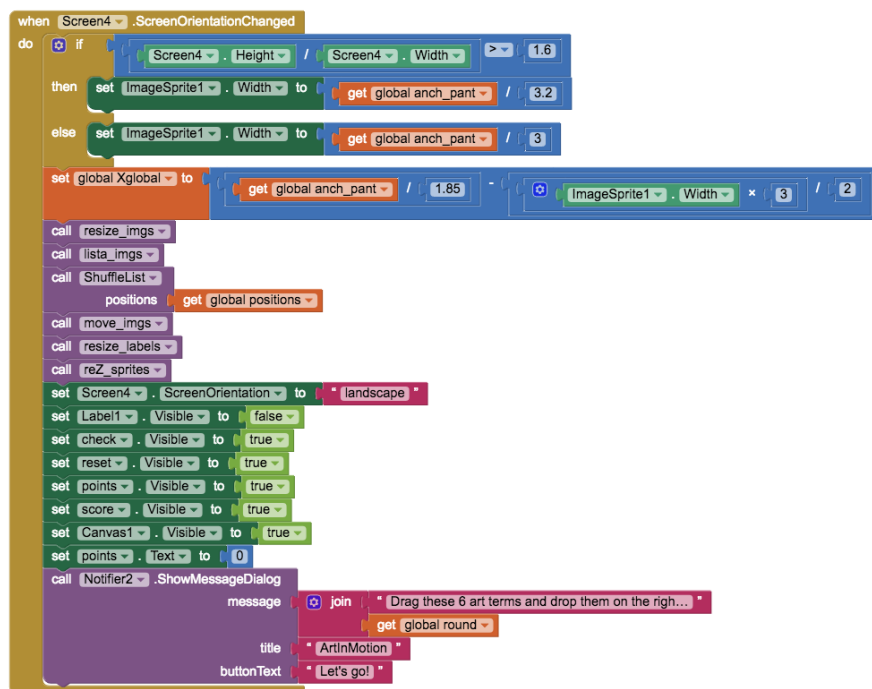
Then, just to make label dragger able, Z index of each label picture is increased to 2 (picture 46).



Picture 46: Screen 4- Changing Z index of every label

Now points are set to 0 and score and point label, check button and reset button are set to visible.

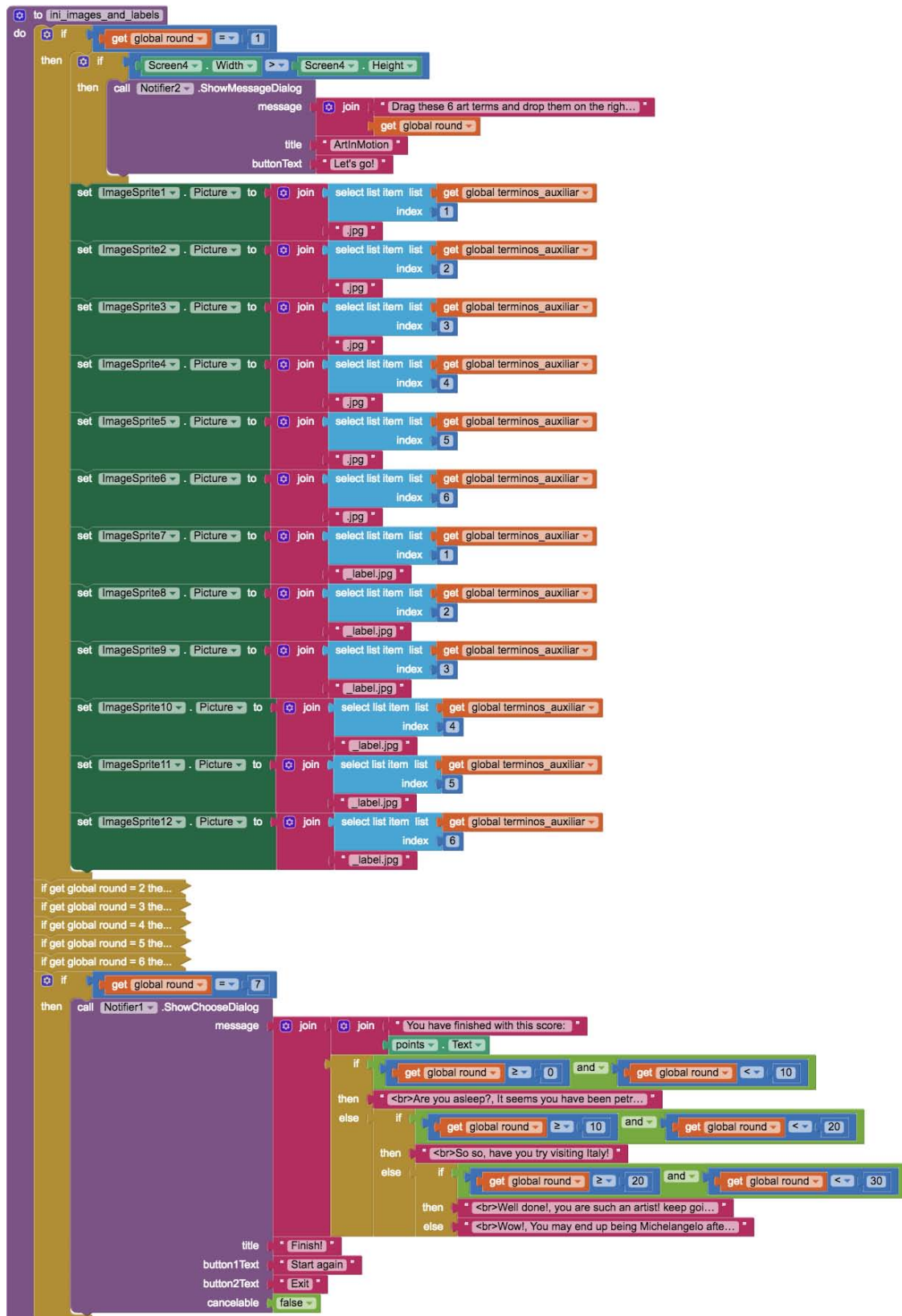
If the device was not set in horizontally position, everything is hidden but Label1, to show the message of: *To play this game, hold the mobile horizontally*, and the app waits to change position and to get the sensor event. When the position is changed, things are set in the same way as initialization (picture 47).



Picture 47: Screen 4- Handler of screen orientation change and initialization

Finally, a message is shown to explain game rules.

Now we are going to explain *ini\_images\_and\_labels* procedure:

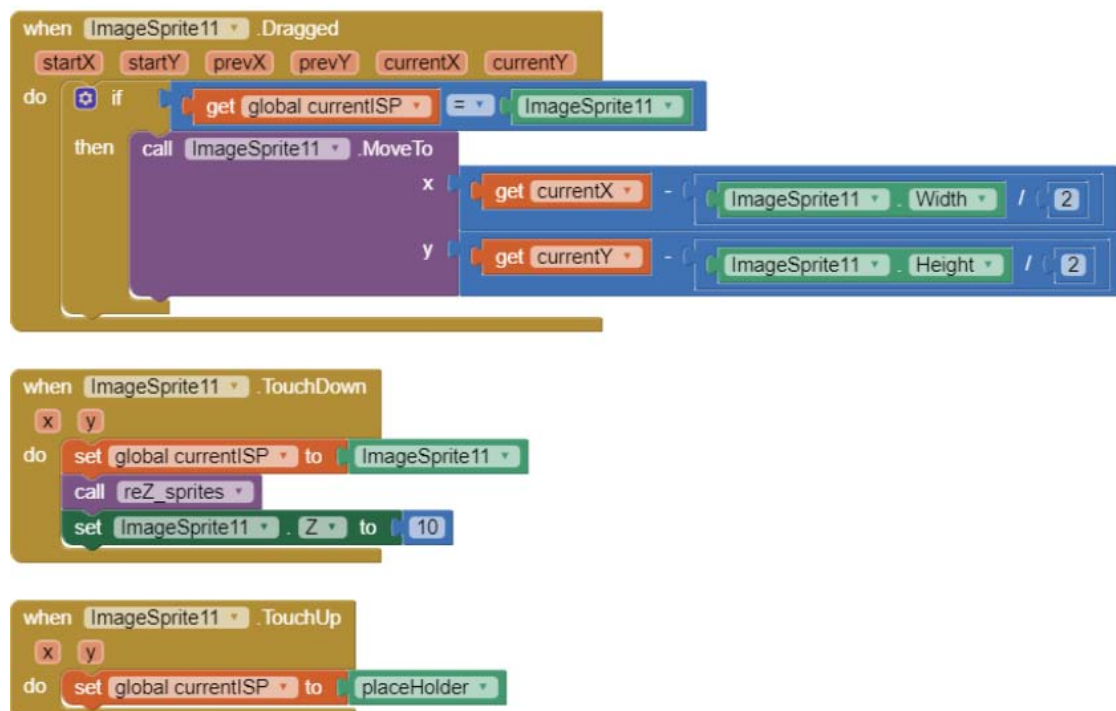


Picture 48: Screen 4- For each round, picture and labels are changed

Depending on *round* variable value, new sets of 6 pictures and 6 labels are charged from *terminos\_auxiliar* where all rows from .csv file are stored.

It is simple, every round, 6 pictures and labels are taken from the list, until last round, where depending on score, a message is displayed in different ranges.

It is time to tackle the core of the quiz, which it is not other than dragging and dropping events. As we explained before, to avoid dragging two image sprites at the same time, we use an auxiliary sprite called *placeholder*. As it is shown in picture 49, we have defined a touching up event, a dragging event and a touching down event for each label image sprite:

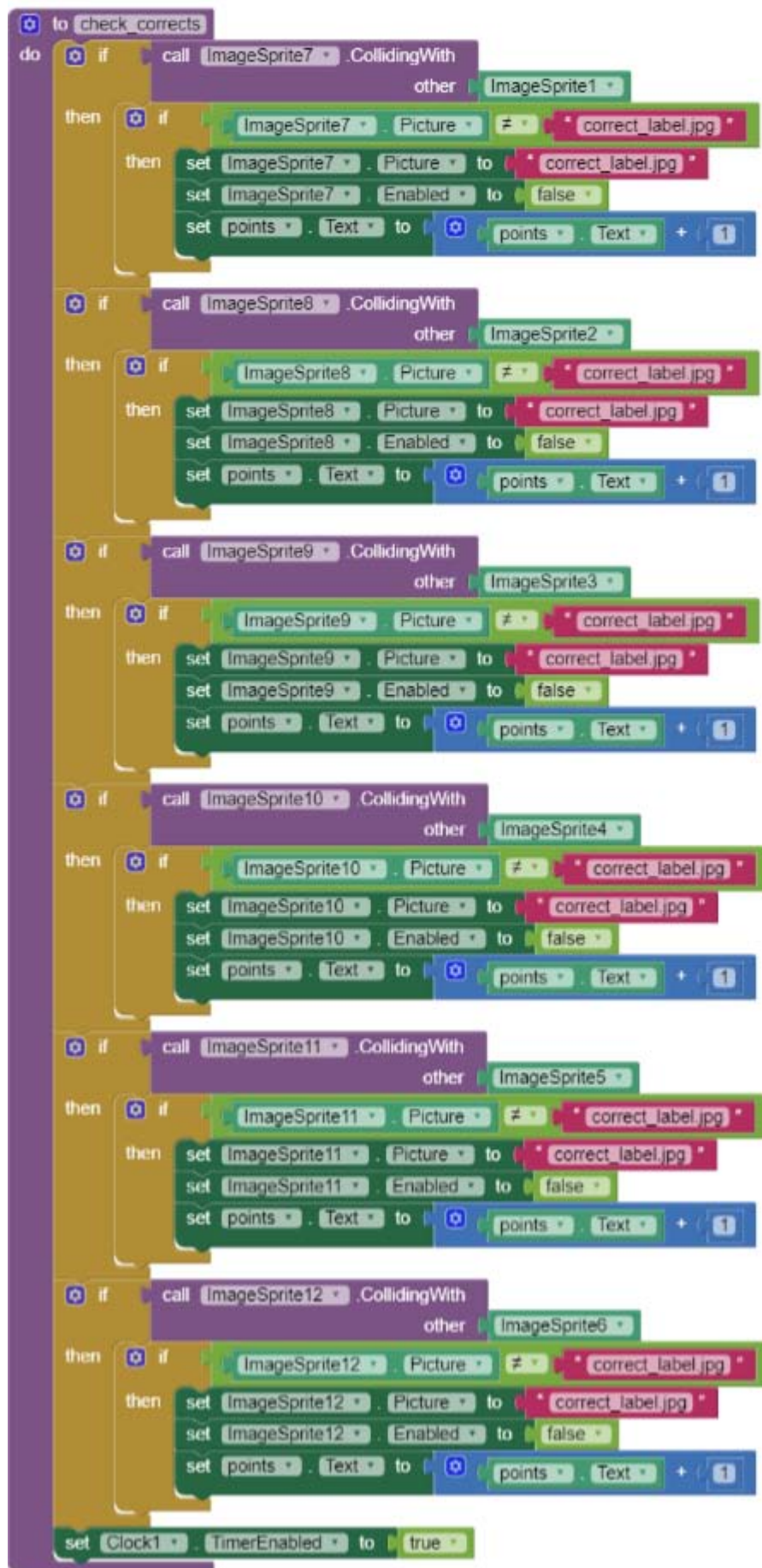


Picture 49: Screen 4- For each label, touching up, dragging and touching down events are defined.

In that way, when a label image sprite is touched down, we assign the *currentISP* to our label image sprite, in order to move just one label. If not, every time that the user touches another label image sprite dragging one; every sprite is added to the movement. While the image sprite is dragged, we change positions X and Y updating them to current position. When the user touches up the screen, we give back the *currentISP* to the auxiliary image sprite *placeholder*. We also set Z index of image sprites that has been dragged to 10, to ensure that it is above all elements on screen.

When the user has placed all the labels or every label that knows, it is the moment to check correct and the user has to press check button. When it is done, Check\_corrects procedure is called (picture 50).

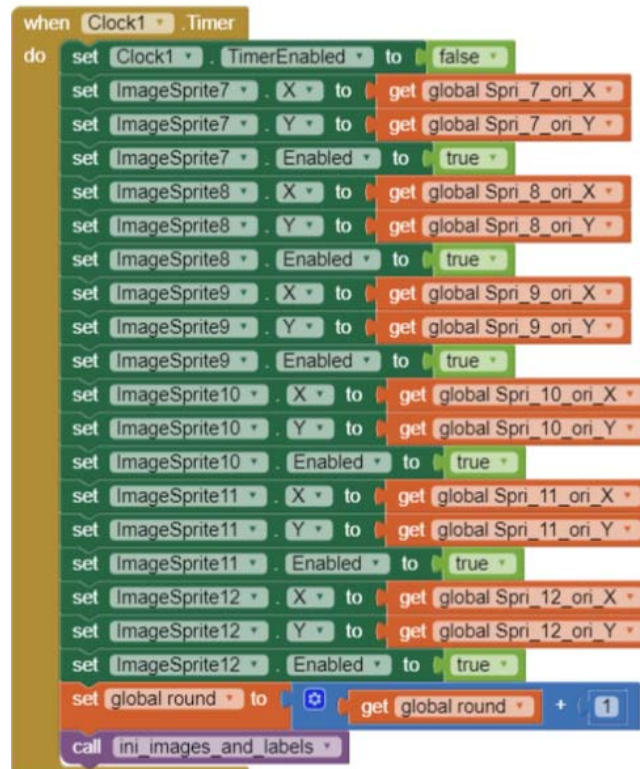




Picture 50: Screen 4- For each label, checking if it collides with the right picture

As we have defined, the answer is correct if label image sprite 7 is over picture image sprite 1, 8 on 2, 9 on 3, 10 on 4, 11 on 5 and 12 on 6. As we change backgrounds of every image sprite in every execution of app and in every round, this checking method can be fixed.

So, every time that `check_corrects` procedure is called, we check if these pairs of images sprites are colliding and if they are, we consider that the answer is correct and we show the Correct! label and we add one point to the scoreboard. While we are doing that, and to avoid users press again any button or image sprite, we enable `clock1` as a countdown of 1 second. After this second the app takes user to the next round, placing all labels to the origin position and calling again `ini_images_and_labels` procedure, showing the next round message and changing all images and labels to the six next ones.



Picture 51: Screen 4- After checking everything is taken to origin and app passes to the next round

Finally, we are going to explain `reset` procedure. This procedure is called when the user presses reset button, which can be pressed any time during the game. When it happens, the round is set to 1 (to the beginning) and every label image sprite is set to the original position in the grid. At the end, `ini_images_and_labels` procedure is called to start the game again (picture 52).

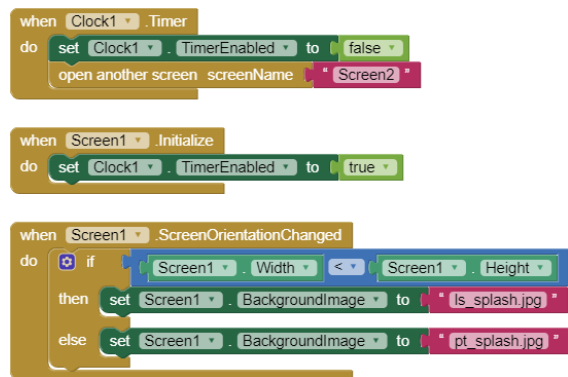




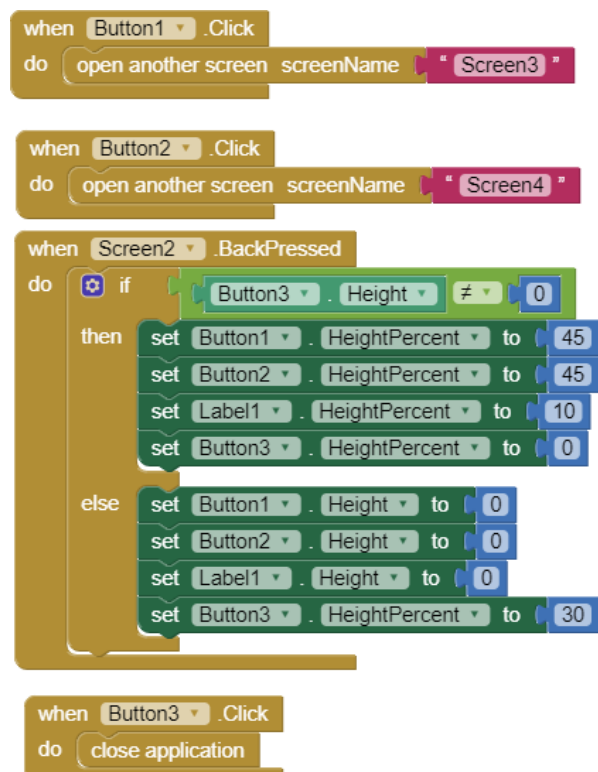
Picture 52: Screen 4- If reset button is pressed anytime during the game

## 3. All blocks

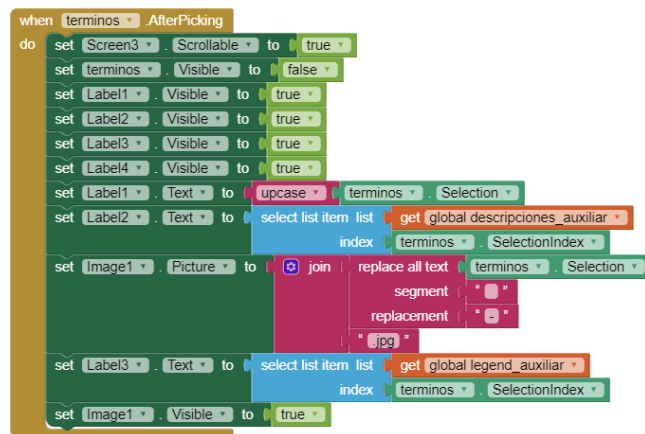
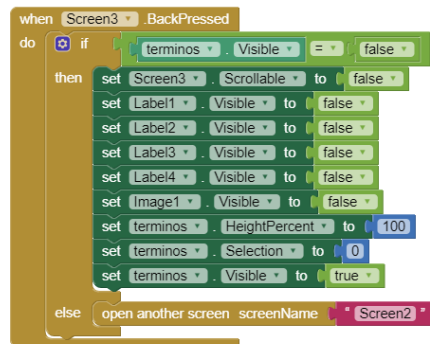
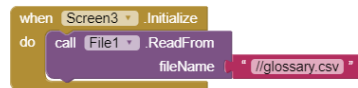
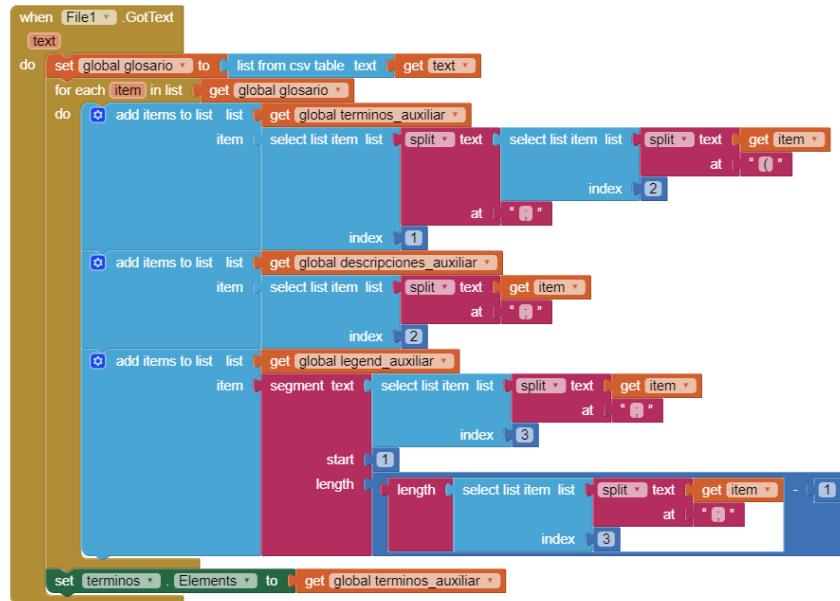
### Screen1



### Screen2



## Screen3



initialize global descripciones\_auxiliar to create empty list

initialize global terminos\_auxiliar to create empty list

initialize global legend\_auxiliar to create empty list

initialize global glosario to create empty list

initialize global terminos to create empty list

**Screen 4**



