

ArtInMotion

Content

1.	ArtInMotion user view				
2.	Des	igning 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	ΔII Ł	nlocks	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.

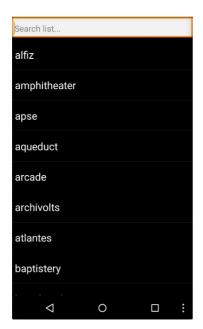


Picture 1: Screen 1 - Splash screen



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).

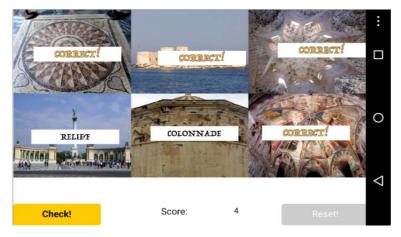


Picture 3: Screen 3 - ListView



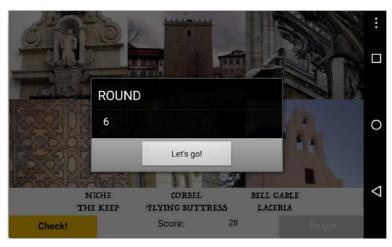
On the other hand, if the user chooses Art and play, positioning the phone horizontally, you get into a quiz game, where, in each round (from six), six pictures are shown and you are asked to drag and drop the right term onto the correspondent picture. Then, with all the labels

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

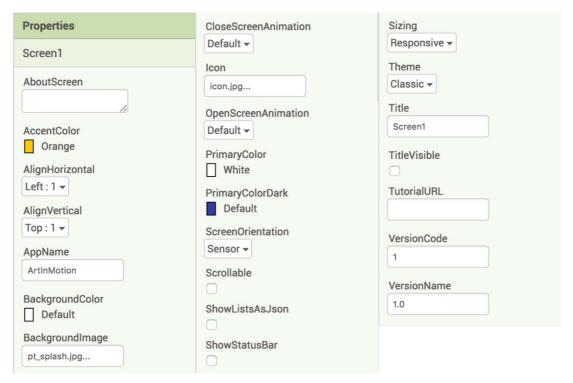
2. Designing and blocking the screens

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:

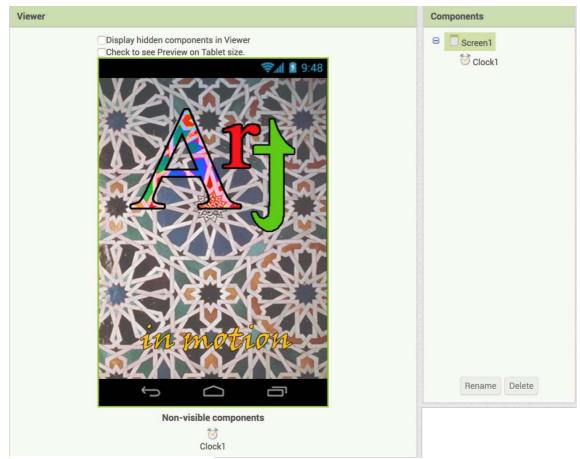


Picture 8: Screen 1 - Properties

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.

```
when Clock1 . TimerEnabled to false open another screen screenName . Screen2 .

when Screen1 .Initialize do set Clock1 . TimerEnabled to true when Screen1 .ScreenOrientationChanged do if Screen1 . Width Screen1 . Height then set Screen1 . BackgroundImage to is splash.jpg else set Screen1 . BackgroundImage to pt_splash.jpg .
```

Picture 10: Screen 1 - Blocks

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

2.2. Screen 2

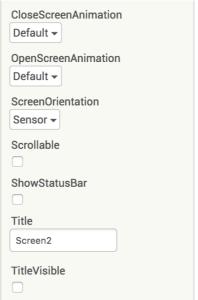
A. Designer

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:

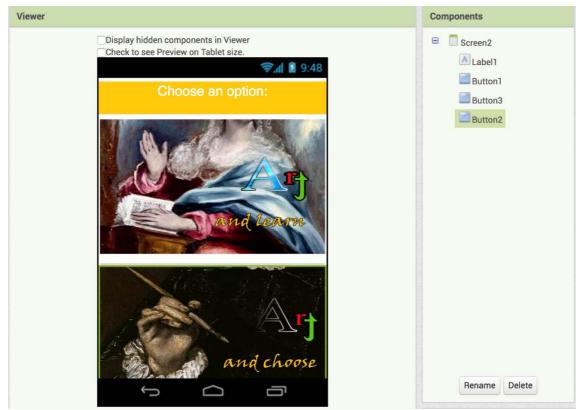
Properties							
Screen2							
AboutScreen							
AlignHorizontal Center: 3 -							
AlignVertical Center : 2 ▼							
BackgroundColor Default							
BackgroundImage None							

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 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).

```
when Button1 ▼ .Click
do open another screen screenName
when Button2 - . Click
do open another screen screenName ( "Screen4"
when Screen2 .BackPressed
do ( Button3 ▼ . Height ▼ ( 0
   set Button2 ▼ . HeightPercent ▼ to (
        set Label1 		. HeightPercent 		 to 			 10
        set Button3 ▼ . HeightPercent ▼ to 0
        set Button2 . Height to 0
        set Label1 . Height to 0
        when Button3 

■ .Click
do close application
```

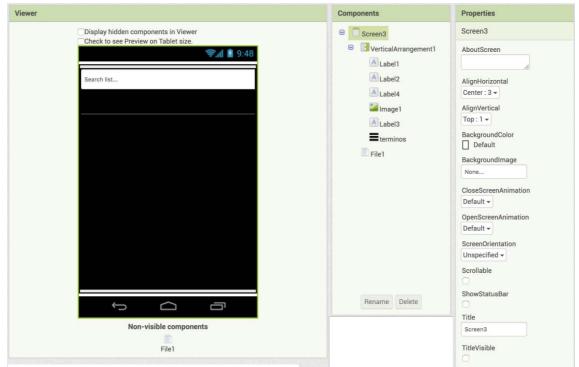
Picture 17: Screen 2 Blocks

2.3. <u>Screen 3</u>

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.

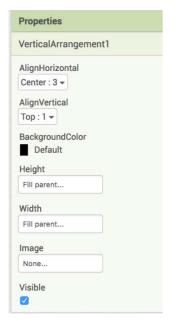


Picture 18: Screen 3 Design, components and properties

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



Label1 properties



Picture 22: Screen 3-Label2 properties



Picture 23: Screen 3-Label3 properties



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:



```
en File1 .GotTex
 set global glosario o to list from csv table text get text
      ch item in list get global glosario
      add items to list list get global terminos_auxiliar -
                                 lect list item list split text
                                                                                             .0.
                                                                         index 2
                                                              · 8 ·
                                                1
      add items to list list get global descripciones_auxiliar
                                 ect list item list | split - text
                                              2
      add items to list list
                              get global legend_auxiliar
                                                      item list split text
                                                             3
                                      start 1
                                                                                        .8.
                                                                        3
 set terminos . Elements to get global terminos_auxiliar .
n Screen3 - In
                         //glossary.csv
```

Picture 26: Screen 3- reading and storing info from .csv file

When screen 3 is initialized, reading from glossary.csv file is done (previously uploaded). As we have completed .csv file in order to have a row per term, we charge whole file in a global list variable called *global glosario*, where we will have a row from the file in each element in the list. Then, we have to read element by element from the initial list *global glosario* to fill each element of a different list just to separate term names from descriptions and legends in three lists.

As we get information from file formatted like this:

We have to split each row from file to take out parenthesis and semicolons and we use auxiliary list elements as *global terminos_auxiliar*, *global descripciones_auxiliar* and *global legend_auxiliar* for storing terms, descriptions and legends separately.

At the end we assign to elements of list *terminos* all the term names stored in *terminos_auxiliar* to show the list view.

Next thing to do is handling even after picking one element from *terminos* list:

```
when terminos AfterPicking

do set Screen3 . Scrollable to true . set terminos . Visible to false . set Label3 . Visible to true . set Label3 . Visible to true . set Label4 . Text to select list item list get global descripciones auxiliar . set Label3 . Picture to gioin replace all text terminos . Selection . set Label3 . Text to select list item list get global legend auxiliar . set Label3 . Text to select list item list get global legend auxiliar . set Label3 . Text to select list item list get global legend auxiliar . set Label3 . Text to select list item list get global legend auxiliar . set Label3 . Text to select list item list index terminos . SelectionIndex . set Image1 . Visible to true .
```

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).

```
when Screen3 BackPressed

do if terminos Visible false then set Screen3 Scrollable to false set Label1 Visible to false set Label2 Visible to false set Label4 Visible to false set Label4 Visible to false set terminos HeightPercent to 100 set terminos Selection to 0 set terminos Visible to true else open another screen screenName Screen2
```

Picture 28: Screen 3- Back button event handler

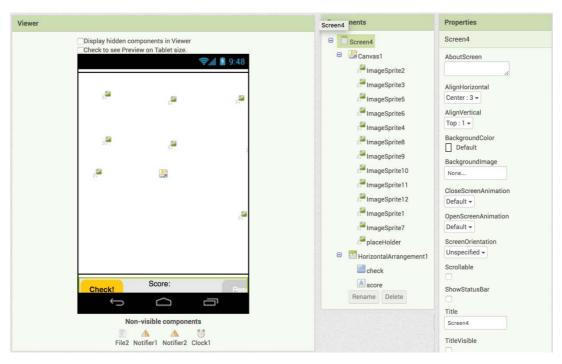
2.4. <u>Screen 4</u>

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.

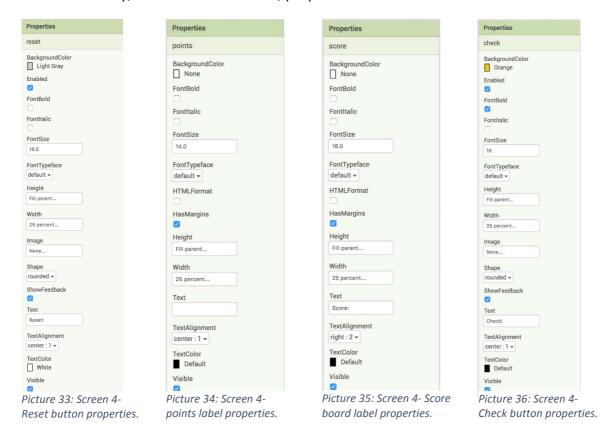


Picture 31: Screen 4- Label imageSprite properties.



Picture 32: Screen 4- Auxiliary imageSprite properties.

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



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:

```
initialize global Spri_12_ori_X to 0
                                                                 initialize global Spri_8_ori_Y to 0
 initialize global round to
                                                                 initialize global Spri_11_ori_X to 0
 initialize global terminos to ( create empty list
                                                                 initialize global Spri_8_ori_X to 0
 initialize global glosario to ( create empty list
                                                                 initialize global Spri_10_ori_Y to 0
 initialize global terminos_auxiliar to ( create empty list
                                                                 initialize global Spri_7_ori_Y to 0
 initialize global Spri_12_ori_Y to 0
                                                                 initialize global Spri_10_ori_X to 0
 initialize global Spri_9_ori_Y to 0
                                                                  nitialize global Spri_7_ori_X to 0
 initialize global Xglobal to 0
                                                                   nitialize global currentISP to 🕻 " 🔲 "
 initialize global Spri_9_ori_X to 0
                                                                  nitialize global positions to 🕻 🔯 create empty list
initialize global Spri_11_ori_Y to 0
                                                                 initialize global anch_pant to 0
Picture 37: Screen 4-
                                                                 initialize global Xlabels to 0
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:

```
Screen4 .Inii
call File2 .ReadFrom
                    //glossary.csv
        Screen4 Width Screen4 Height
then 🔯 if
                 Screen4 . Width . / Screen4 . Height
          set ImageSprite1 . Width . to
                                         Canvas1 - . Width - / (3.4)
             ImageSprite1 - . Width - to
                                         Canvas1 Width / 3
     set global Xglobal - to
                             Canvas1 - Width - / 2
                                                                 ImageSprite1 - Width - × 3 / 2
     call resize_imgs -
     call lista_imgs =
     call resize_labels -
     call ShuffleList
                     get global positions -
     call move_imgs -
     call reZ_sprites >
     set points . Text to 0
     set Label1 . Visible to false
     set Screen4 . ScreenOrientation to landscape
     set check . Visible to true
     set reset . Visible to true
     set points . Visible to true
     set score . Visible to true
     set Canvas1 . Visible to false
     set Label1 . Visible to true
     set points . Visible to false
     set reset . Visible to
     set check . Visible to false
     set score . Visible to false
       global anch_pant - to
                               Canvas1 - Height - + (
                                                           Screen4 - . Width - × 0.05
```

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_mages_and_labels* is called to initiate each round (here the first one), we will explain it later.

```
when File2 ...GotText

text

do set global glosario to list from csv table text get text

for each item in list get global glosario do add items to list list get global terminos auxiliar replace all text select list item list split text select list item list split text get item at "()"

index 2

at ";"

call ShuffleList positions get global terminos auxiliar call ini_images_and_labels visit from csv table text get text select list item list split text get item at "()"

index 2

at ";"
```

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).

```
to resize_imgs
ImageSprite1 ▼ . Width ▼
                                              4
   set ImageSprite1 ▼ . X ▼ to get global Xglobal ▼
    0
    set [ImageSprite2 → ]. Width → to [ImageSprite1 → ]. Width →
    ImageSprite1 - Height -
   set ImageSprite1 		□ . Y 		□ to
                             0
   set [ImageSprite3 - ]. Width - to [ImageSprite1 - ]. Width -
    set ImageSprite3 → . Height → to ImageSprite1 → . Height →
    set ImageSprite3 ▼ . Y ▼ to □ 0
   set ImageSprite3 ▼ . X ▼ to
                                  ImageSprite1 - . Width - × 2
                                                                     🛨 🥊 get (global Xglobal 🔻
    set ImageSprite4 ▼ . Width ▼ to ImageSprite1 ▼ . Width ▼
   set [ImageSprite4 → . Height → to | ImageSprite1 → . Height →
    set ImageSprite4 ▼ . X ▼ to get global Xglobal ▼
   set ImageSprite4 ▼ . Y ▼ to ImageSprite1 ▼ . Height ▼
    set ImageSprite5 ▼ . Width ▼ to ImageSprite1 ▼ . Width ▼
   set [mageSprite5 ▼]. Height ▼ to [ ImageSprite1 ▼]. Height ▼
   set [mageSprite5 ▼ . X ▼ to [ to [ ImageSprite1 ▼ . Width ▼ + Figet global Xglobal ▼
    set ImageSprite5 ▼ . Y ▼ to I ImageSprite1 ▼ . Height ▼
   set [mageSprite6 - ]. Width - to [mageSprite1 - ]. Width -
   set [mageSprite6 ▼ . Height ▼ to [mageSprite1 ▼ . Height ▼
   set [mageSprite6 → . X → to [ ② | ② | [mageSprite1 → . Width → × [ 2 ] + | get global Xglobal →
   set ImageSprite6 → . Y → to ImageSprite1 → . Height →
```

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).

```
to resize_labels
set [ImageSprite7 ▼ . Width ▼ to
                                                                                                                    0.75 × ImageSprite1 • Width •
   ② if Canvas1 ▼ . Width ▼ >▼ get global anch_pant ▼
    then set global Xlabels to
                                                                                                                      else set global Xlabels v to
                                                                                                                   get global anch_pant / 2 - 1 (0 | ImageSprite7 v). Width v x (3 ) / 2 | + get global Xglobal v
  set [mageSprite7 - X - to | get global Xlabels - set [mageSprite7 - Y - to | 0 | [mageSprite1 - Height - x | 2 |
         et global Spri_7_ori_X ▼ to 【 ImageSprite7 ▼ . X ▼
        et global Spri_7_ori_Y v to I ImageSprite7 v . Y v et ImageSprite8 v . Width v to ImageSprite7 v . Width v
  set [mageSprite8 - . X - to 1 0 | [mageSprite7 - . Width - + | get global Xlabels - set [mageSprite8 - . Y - to 1 0 | [mageSprite1 - . Height - x 2]
  set [mageSprite8 ] . Height | to [ ImageSprite7 ] . Height |
      set global Spri_8_ori_X > to | ImageSprite8 > . X > set global Spri_8_ori_Y > to | ImageSprite8 > . Y >
   set [mageSprite9 → . X → to to to to mageSprite7 → . Width → × 2 + get global Xlabels →
  set ImageSprite9 v . Y v to ImageSprite1 v . Height v × 2
 set [mageSprite9 v . Width v to [mageSprite7 v . Width v set [mageSprite9 v . Height v to [mageSprite7 v . Height v set global Spri_9_ori_X v to [mageSprite9 v . X v set global Spri_9_ori_X v to [mageSprite9 v . X v set global Spri_9_ori_X v to [mageSprite9 v . X v set [mageSprite10 v . Width v to [mageSprite7 v . Width v to [mageSprite
   set [ImageSprite10 → . [Height → to | ImageSprite7 → . [Height →
  set global Spri_10_ori_X > to | ImageSprite10 > . X > set global Spri_10_ori_Y > to | ImageSprite10 > . Y > set ImageSprite11 > . Width > to | ImageSprite7 > . Width >
  set [mageSprite11 v. X v to | 0 | mageSprite7 v. Width v | eqt global Alaucia set [mageSprite11 v. Y v to | 0 | mageSprite1 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | + | mageSprite7 v. Height v | 2 | 
   set [ImageSprite11 ▼ . [Height ▼ to [ImageSprite7 ▼ . [Height ▼
  set global Spri_11_ori_Y v to | ImageSprite11 v . Y v set ImageSprite12 v . Width v to | ImageSprite7 v . Width v
   set [ImageSprite12 ▼ . Height ▼ to [ImageSprite7 ▼ . Height ▼
   set [ImageSprite12 ▼ . X ▼ to ]
                                                                                                    ImageSprite7 . Width × × 2 + get global Xlabels v
   set [mageSprite12 ▼ . Y ▼ to | ② | ② | [mageSprite1 ▼ . Height ▼ | ▼ | 2 | + | [mageSprite7 ▼ . Height ▼
     set_global Spri_12_ori_X ▼ to ( ImageSprite12 ▼ . X ▼
      set global Spri_12_ori_Y ▼ to [ ImageSprite12 ▼ . Y ▼
```

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).

```
to ShuffleList positions

do for each I from 1

to length of list list get positions by 1

do initialize local replace to get i + random integer from 0 to length of list list get positions - get i replace get replace get replace get replace get replace
```

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).

```
to Swap positions index replace

do initialize local posValues to select list item list get positions

index get index

replacement select list item list get positions

index get replace

replace list item list get positions

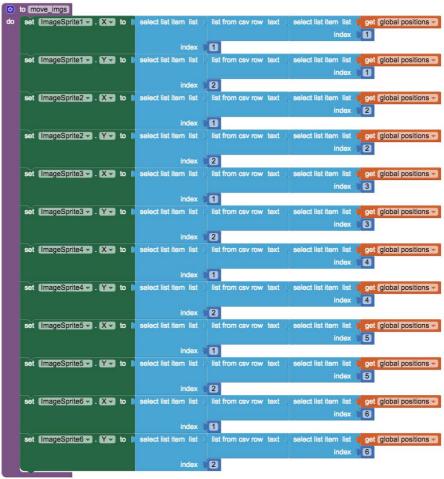
index get replace

replace ment get posvalues

replacement get posValues
```

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).

```
to reZ_sprites

do set ImageSprite7 . Enabled to true v

set ImageSprite8 . Enabled to true v

set ImageSprite8 . Enabled to true v

set ImageSprite9 . Enabled to true v

set ImageSprite9 . Enabled to true v

set ImageSprite10 . Enabled to true v

set ImageSprite10 . Enabled to true v

set ImageSprite11 . Enabled to true v

set ImageSprite12 . Enabled to true v

set ImageSprite12 . Enabled to true v

set ImageSprite12 . Enabled to true v
```

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).

```
Screen4 Height V Screen4 Width V 1.6
    set [mageSprite1 → . Width → to get global anch_pant → / 3.2]
     set [ImageSprite1 ▼ . Width ▼ to
                         get global anch_pant - / 1.85
                                                                   ImageSprite1 . Width . × 3 / 2
call resize_imgs >
                get global positions -
   move_imgs ~
set Screen4 - . ScreenOrientation - to | landscape "
   Label1 ▼ . Visible ▼ to false ▼
set check . Visible to true
      et 🔻 . (Visib
                le to true
set points . Visible to true
   Canvas1 ▼ . Visible ▼ to true ▼
 set points . Text to 0
                                           Drag these 6 art terms and drop them on the righ...
                                          get global round -
                                 ArtInMotion 
                                " [Let's go!] "
```

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:

```
o ini_images_and_labels
      get global round = 1
             Screen4 . Width . Screen4 . Height
                                                   Drag these 6 art terms and drop them on the righ...
                                         * ArtInMotion **
                                         Let's go!
     set ImageSprite1 . Picture . to
     set [mageSprite2 . Picture to
                                            .jpg
     set ImageSprite3 . Picture to
     set ImageSprite4 - . Picture - to
     set ImageSprite5 . Picture to I
                                                              global terminos_auxiliar >
     set ImageSprite6 . Picture to
     set [mageSprite7 ] . Picture | to [
     set ImageSprite8 . Picture to
     set ImageSprite9 . Picture to
                                   ioi 🗯
     set [ImageSprite10 ▼ . Picture ▼ to
     set ImageSprite11 . Picture to
                                                    You have finished with this score:
                                                   points . Text
                                                    get global round ≥ 0 and get global round ✓ 10
                                                   <br>Are you asleep?, It seems you have been petr... 
                                                         get global round > 2 10 and >
                                                         <br>So so, have you try visiting Italy!
                                                              get global round → ≥ ▼ 20 and ▼ get global round ▼ < ▼ 30
                                                         se Se Se Ser>Wow!, You may end up being Michelangelo afte...
                                   Finish!
                                   Start again
                                  Exit
```

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:

```
when ImageSprite11 Dragged
 startX startY
                prevX prevY
                              currentY currentY
               get global currentISP v = v | ImageSprite11 v
          call ImageSprite11 . MoveTo
                                         get currentX
                                                              ImageSprite11 *
                                                                              Width • //
                                         get currentY
                                                              ImageSprite11 Height
when ImageSprite11 TouchDown
do set global currentISP v to ImageSprite11 v
    call reZ_sprites *
    set ImageSprite11
when ImageSprite11 TouchUp
x y
do set global currentISP to placeHolder v
```

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).

```
to check corrects
  o if call [mageSprite7 . CollidingWith
                                  other | ImageSprite1
   then il ImageSprite7 Picture * # correct_label.jpg
        then set [mageSprite7...] . Picture ... to ... correct_label.jpg ...
              set ImageSprite7 Enabled to false
              set points . Text . to points . Text . 1
   o if call [mageSprite8] .CollidingWith
                                 other I ImageSprite2
   then (a) if ImageSprite8 • Picture • # correct label jpg
         then set (mageSprite8 . Picture . to correct label jpg
              set ImageSprite8 . Enabled . to false .
              set points... Text... to O points... Text...
   i call ImageSprite9 • CollidingWith
                                 other ImageSprite3 *
   then (c) if ImageSprite9 • Picture • # correct label jpg
        then set [mageSprite9 . Picture . to correct label.jpg ...
              set [mageSprite9 v]. Enabled v to false v
              set points . Text to points . Text .
   o if call [mageSprite10 CollidingWith]
                                  other ImageSprite4
   then (a) if ImageSprite10 • Picture • # Correct label pg
        then set [mageSprite10 * ] Picture * to ( correct_label.jpg) *
              set (ImageSprite10 ... Enabled ... to () false ...
              set points . Text . to points . Text . 1
   o if call [mageSprite11 - CollidingWith
                                 other ImageSprite5
   then if ImageSprite11 Picture correct_label.jpg
        then set ImageSprite11 . Picture to correct label.jpg
              set (ImageSprite11st) Enabled to ( false *)
              set points . Text to points . Text .
   o if call ImageSprite12 . CollidingWith
                                  other ImageSprite6 •
   then O if ImageSprite12 Picture  # correct_label.jpg *
         then set [mageSprite12 . Picture to to correct label pg *
              set [mageSprite12 ... Enabled ... to ] false ...
              set points. Text. to points. Text.
   set Clock1 ... TimerEnabled ... to true ...
```

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 <code>ini_images_and_labels</code> procedure, showing the next round message and changing all images and labels to the six next ones.

```
when Clock1 . Timer
   set Clock1 . TimerEnabled to false
    set ImageSprite7 . X . to get global Spri_7_ori_X
    set [mageSprite7]. Y to [
                                 get global Spri_7_ori_Y
                                 to true *
    set ImageSprite7
                      Enabled *
    set ImageSprite8 . X . to get global Spri_8_ori_X .
    set ImageSprite8 . Y . to
                                 get global Spri_8_ori_Y
                      . Enabled v to true v
    set ImageSprite8 *
    set ImageSprite9 . X v to
                                 get global Spri 9_ori X
    set [mageSprite9 v . Y v to ]
                                 get global Spri_9_ori_Y
    set ImageSprite9 . Enabled to true .
                                 get global Spri_10_ori_X
    set [mageSprite10 v . X v to [
    set [mageSprite10 * . Y * to ]
                                  get global Spri_10_ori_Y
    set ImageSprite10 . Enabled to fitrue .
    set ImageSprite11 . X . to get global Spri_11_ori_X .
    set ImageSprite11 Y to
                                  get global Spri_11_ori_Y
    set ImageSprite11 . Enabled to
                                       true *
    set [mageSprite12 * ]. X * to get global Spri 12 on X
    set ImageSprite12 . Y . to get global Spri_12 ori_Y .
    set ImageSprite12 . Enabled to true
    set global round 1 to 2
                              get global round
   call ini_images_and_labels *
```

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).

```
to reset
do set global round to 1
    set ImageSprite7 • . X • to get global Spri_7_ori_X • set ImageSprite7 • . Y • to get global Spri_7_ori_Y •
     set [mageSprite7 . Enabled to [true v]
     set ImageSprite8 • . X • to get global Spri_8_ori_X • set ImageSprite8 • . Y • to get global Spri_8_ori_Y •
     set ImageSprite8 . Enabled to true
     set ImageSprite9 • . X • to | get global Spri_9 ori_X • set ImageSprite9 • . Y • to | get global Spri_9 ori_Y •
     set [mageSprite9 . Enabled . to [true .
     set ImageSprite10 . X . to get global Spri_10_ori_X .
     set [mageSprite10 * . Y * to get global Spri_10_ori_Y *
     set [mageSprite10 * ]. Enabled * to t true *
     set [mageSprite11 * . X * to [get global Spri_11_ori_X *
     set ImageSprite11 . Y to get global Spri 11 or Y
     set [mageSprite11 * ]. Enabled * to [true *]
     set [ImageSprite12 * ]. X * to get global Spri_12_ori_X *
     set [mageSprite12 v . Y v to | get global Spri 12 or Y v
     set [mageSprite12 v ]. Enabled v to [true v
     set points . Text to 0
    call ini_images_and_labels *
```

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

3. All blocks

Screen1

```
when Clock1 * . Timer
do set Clock1 * . TimerEnabled * to false * open another screen screenName * Screen2 *

when Screen1 * Initialize
do set Clock1 * . TimerEnabled * to true *

when Screen1 * . Screen1 * . Width * < * Screen1 * . Height * then set Screen1 * . BackgroundImage * to * pt_splash.jpg * else set Screen1 * . BackgroundImage * to * pt_splash.jpg *
```

Screen2

```
when Button1 .Click
do open another screen screenName [ "Screen3"
when Button2 .Click
when Screen2 .BackPressed
do if Button3 . Height . # 10
   then set Button1 . HeightPercent to 45
        set Button2 . HeightPercent v to 45
        set Label1 . HeightPercent to 10
        set Button3 . HeightPercent to 0
   else set Button1 . Height to 0
        set Button2 v . Height v to 0
        set Label1 . Height to 0
        set Button3 v . HeightPercent v to 30
when Button3 .Click
do close application
```

```
Screen3
           vhen File1 .GotText
       do set global glosario v to list from csv table text get text v
                        for each litem in list leget global glosario vodo add items to list list get global terminos_auxiliar vodo.
                                                                                                                  item select list item list split text select list item list split text get (item at ( ) at ( 
                                           index 1

o add items to list list get global descripciones auxiliar vitem select list item list split vitex get item vat
                                                                                                                                                                                                                                                                                                                                                    index [2]
                                              o add items to list list get global legend auxiliar ritem segment text select list item list split text get (item rat | " get (item rat | 
                                                                                                                                                                                                                                                              index (3)
                                                                                                                                                                                       start (1
                                                                                                                                                                                 length ( select list item list split text get fitem at index 3
                    set terminos . Elements . to get global terminos auxiliar .
       do call File1 .ReadFrom
                    fileName ( " (//glossary.csv "
    when Screen3 • BackPressed

do if terminos • Visible • = • (false • )
                     then set Screen3 v . Scrollable v to false v set Label1 v . Visible v to false v set Label2 v . Visible v to false v set Label3 v . Visible v to false v set Label4 v . Visible v to false v set Label4 v . Visible v to false v set Image1 v . Visible v to false v set Image1 v . Visible v to false v set terminos v . Selection v to 100 set terminos v . Selection v to 100 set terminos v . Visible v to true v
                        else open another screen screenName * Screen2 *
   when terminos * AfterPicking
do set Screen3 * . Scrollable * to true *
set terminos * . Visible * to false *
set Label1 * . Visible * to true *
set Label2 * . Visible * to true *
set Label3 * . Visible * to true *
set Label3 * . Visible * to true *
set Label4 * . Visible * to true *
                      set Label1 • . Text • to pupcase • terminos • . Selection • select Label2 • . Text • to select list item list pupcase • get global descripciones auxiliar • index terminos • . SelectionIndex •
                        set [mage1 v ]. Picture v to ( join [
                                                                                                                                                                                                           replace all text ( terminos v . Selection v
                                                                                                                                                                                                              segment ( * 🗎 " replacement ( * 🚍 "
                      set Label3 * . Text * to * select list item list | get global legend_auxiliar * index | terminos * . SelectionIndex *
                      set [mage1 v]. Visible v to [true v]
  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 Cocreate empty list
```

Screen 4

```
In section occurs acres to
                                                                                                                                                                     a married actions have a
PROPERTY AND ADDRESS.
                                                                                                                                                                                                                   THE RESIDENCE OF
                                                                                                                                                                                             PERSONAL PROPERTY AND 
                                                                                                                                                                     I SHEED OFFICE AND A
                                                                                                                                                                 THE RESERVE NAME OF REPORT OF
    THE RESERVED IN COLUMN
```

```
THE PERSON NAMED IN
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             Committee of the last of the l
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             PRODUCE OF THE PARTY OF
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              DESCRIPTION OF THE OWNER, THE OWN
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             DESCRIPTION OF THE PERSON NAMED IN COLUMN 1 IN COLUMN 
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      STREET, STREET, SQUARE, SQUARE
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   Color Street Street
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  COURS COR LA
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              ACCORDING TO THE OWNER, WHEN THE PARTY OF TH
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  STREET, STREET, ST
                PRODUCE HER
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        STREET, SQUARE, SQUARE
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        COMP COM - COMP COM
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        STATE OF STREET
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   ACTORNAL METER - 14 . HEFTER METERS
```