



Table of Contents

What You're Building	2
Getting Started	3
Introduction	4
Iteration or how we will get to Finished App.	4
Iteration 1 Create still image of our Game	5
Getting Ready	5
Set up the Components	6
Bus stop reached !	10
Iteration 2 Add Moving Ship	11
Moving the Pirate	11
Bus stop reached !	12
Iteration 3 Add Moving Coins	12
Moving the Coins	12
Bus stop reached !	14
Iteration 4 Add collect Coins	15
Detecting Collisions	15
Bus stop reached !	18
Iteration 5 Add Levels	18
New Level	18
Bus stop reached !	19
Iteration 6 Add Time limit and Game Over	20
Starting Game	20
Tip	20
Game Over	21
Reset Button	21
Bus stop reached !	22
Iteration 7 Add Variations - over to you !	22
Variations	22
I see a mistake !	23
Appendix 1.	24





Variables	24
Built-in blocks	24
Procedure Blocks	27
Designer	28
Source	29
Assets	29

What You're Building





Figure 1. The Get the Gold app

By building the [Get The Gold App](#) you will get practice with;

- setting visibility
- using Clock components
- Timers
- detecting collisions
- Orientation sensor

You'll program an application that has a pirate ship whose goal is to collect all the gold on the screen.

Getting Started





Connect to the App Inventor web site and start a new project. Name it **GetTheGold**, and also set the screen's **Title** to "GetTheGold".

Introduction

This tutorial introduces the following skills, useful for future game development:

- Using the Clock component
- Using Clock.Timer to move sprites
- Using Orientation Sensor to move a sprite
- Using collision detection
- Using Slider Component
- Setting an Icon for the Application

Iteration or how we will get to Finished App.

Now we are not going to go immediately create the finished App. We are going to use a process called 'iteration' to start with a basic App and then gradually iterate to our finished app.

It's like getting on a bus. There are bus stops along the way to our finish destination, so 'all aboard' and off we go !



CoderDojo Castleknock



Figure 2. Iteration is like a bus stop.

So the iterations, or bus stops we will go through are;

1. Create still image of our Game
2. Add Moving Ship
3. Add Moving Coins
4. Add collect Coins
5. Add Levels
6. Add Time limit and Game Over
7. Add Variations - over to you !

At each step, we should have a working app before we continue to the next step, we call this iterative development.

Iteration 1 Create still image of our Game

Getting Ready



CoderDojo Castleknock

For this game, you will have two types of image [sprites](#). Click below to download the image file for your sprites.



Figure 3. ImageSprites used in Get the Gold app

Set up the Components

Use the component designer to create the interface for **GetTheGold**. When you finish, it should look something like the Figure 4 below (more detailed instructions below the figure).

Because we are using iteration, our first stop is to Design the look and feel of our app. We will do this in the Designer editor and not need to create any blocks of code for our first stop !

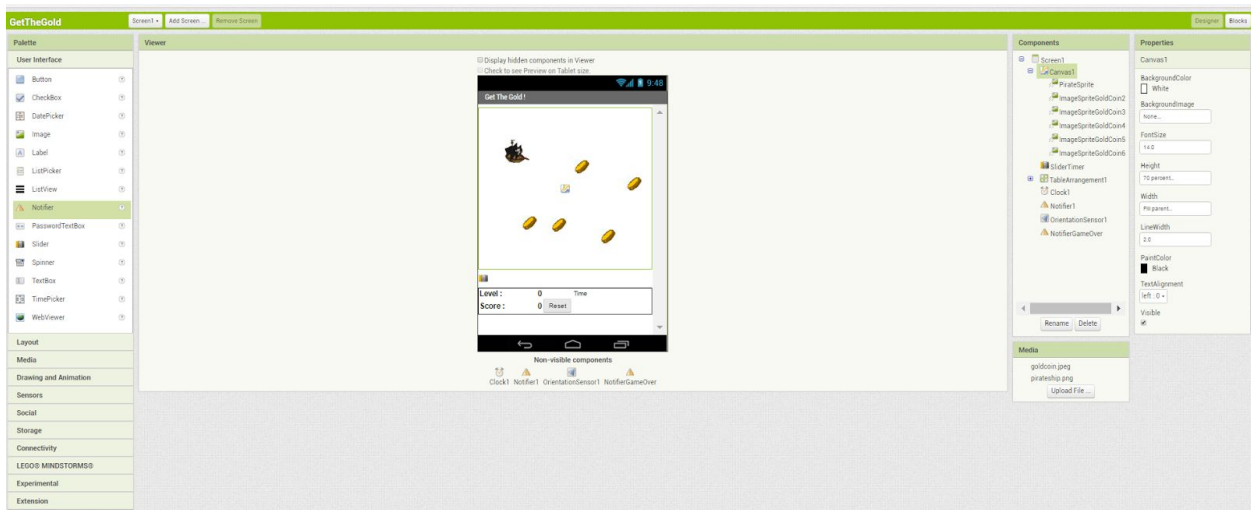


Figure 4. Component designer layout used in Get the Gold app



There are some components in the table, so we will look at them separately;

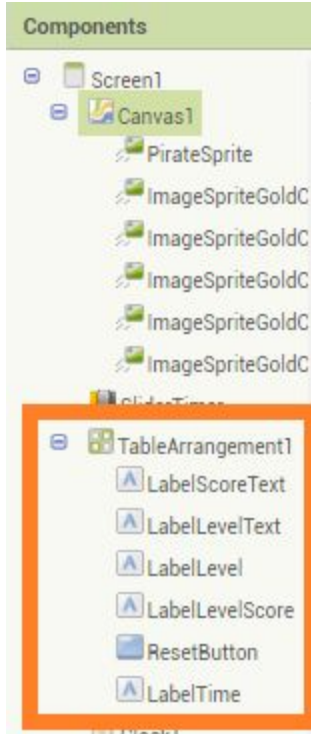


Figure 5. Component designer layout for TableArrangement used in Get the Gold app

To create this interface, put the following components into the Designer by dragging them from the Component Palette into the Viewer.

<i>Component Type</i>	<i>Palette Group</i>	<i>What you will name it</i>	<i>Purpose</i>
<i>Slider</i>	<i>UserInterface</i>	<i>Slider1</i>	<i>Slider represents time left</i>
<i>TableArrangement</i>	<i>Layout</i>	<i>TableArrangement1</i>	<i>Holder for all the button, Score and Level so they all go on the table layout</i>
<i>Label</i>	<i>UserInterface</i>	<i>LabelTime</i>	<i>Say that our slider represents time left.</i>
<i>Label</i>	<i>UserInterface</i>	<i>LabelLevelScore</i>	<i>Say what the game score is.</i>
<i>Label</i>	<i>UserInterface</i>	<i>LabelLevel</i>	<i>Say what level of the game we are on.</i>



<i>Label</i>	<i>UserInterface</i>	<i>LabelLevelText</i>	<i>Say "Level :"</i>
<i>Label</i>	<i>UserInterface</i>	<i>LabelScoreText</i>	<i>Say "Score :"</i>
<i>Button</i>	<i>Basic</i>	<i>ResetButton</i>	<i>To reset the game so the player can play again.</i>
<i>Canvas</i>	<i>Drawing and Animation</i>	<i>Canvas1</i>	<i>The background that we will be putting our imagesprites on.</i>
<i>ImageSprite</i>	<i>Drawing and Animation</i>	<i>PirateSprite</i>	<i>The Pirate Ship in our game.</i>
<i>ImageSprite</i>	<i>Drawing and Animation</i>	<i>ImageSpriteGoldCoin2</i>	<i>One of the gold coins in the game.</i>
<i>ImageSprite</i>	<i>Drawing and Animation</i>	<i>ImageSpriteGoldCoin3</i>	<i>One of the gold coins in the game.</i>
<i>ImageSprite</i>	<i>Drawing and Animation</i>	<i>ImageSpriteGoldCoin4</i>	<i>One of the gold coins in the game.</i>
<i>ImageSprite</i>	<i>Drawing and Animation</i>	<i>ImageSpriteGoldCoin5</i>	<i>One of the gold coins in the game.</i>
<i>ImageSprite</i>	<i>Drawing and Animation</i>	<i>ImageSpriteGoldCoin6</i>	<i>One of the gold coins in the game.</i>

Table 1. All of the visible components for the Get the Gold app

There are two main types of components in an app: visible and non-visible.

The app’s visible components are the ones you can see when the app is launched—things like buttons, text boxes, and labels. These are often referred to as the app’s user interface.

Non-visible components are those you can’t see, so they’re not part of the user interface. Instead, they provide access to the built-in functionality of the



device; for example, the *Texting* component sends and processes SMS texts, the *LocationSensor* component determines the device’s location, and the *TextToSpeech* component talks. The non-visible components are the technology within the device—little people that do jobs for your app.

<i>Component Type</i>	<i>Palette Group</i>	<i>What you will name it</i>	<i>Purpose</i>
<i>Clock</i>	<i>Sensors</i>	<i>Clock1</i>	<i>We use the Clock for its Timer method to move the coins.</i>
<i>Notifier</i>	<i>UserInterface</i>	<i>Notifier1</i>	<i>Tell player we have finished a level.</i>
<i>Sensor</i>	<i>OrientationSensor</i>	<i>OrientationSensor1</i>	<i>Find out what way we are pointing our phone.</i>
<i>Notifier</i>	<i>UserInterface</i>	<i>NotifierGameOver</i>	<i>Tell player game is over.</i>

Table 2. All of the non-visible components for the Get the Gold app

Set the properties of the components as described below:

<i>Component Type</i>	<i>Action</i>	<i>Another Action</i>
<i>ResetButton</i>	Change Text property to "Reset".	<i>No other changes required</i>
<i>PirateSprite</i>	Change Speed property to 6.	Upload the pirateship image and set Picture property to pirateship.
<i>ImageSpriteGoldCoin(2,3,4,5,6)</i>	Upload the goldcoin image and set Picture property to goldcoin.	<i>No other changes required.</i>
<i>Clock</i>	Change TimerInterval property to 2000	<i>No other changes required.</i>
<i>Screen1</i>	Change icon to pirateship.png	<i>Set ScreenOrientation to Fixed.</i>



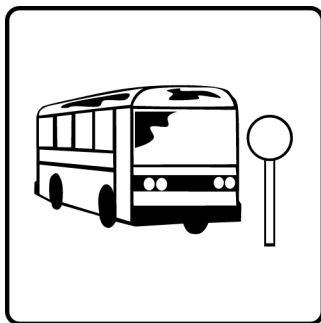
Screen1	Set Title to "Get the Gold!"	<i>No other changes required.</i>
Canvas1	<i>Set Height to 70%</i>	<i>No other changes required.</i>
SliderTimer	<i>Set MaxValue to 30</i>	<i>Set MinValue to 0</i>
SliderTimer	<i>Set ThumbPosition to 30</i>	<i>No other changes required.</i>
TableArrangement1	<i>Set Rows to 2</i>	<i>Set Columns to 6</i>
LabelScoreText	<i>Set FontSize to 16</i>	<i>Set Text to "Score:"</i>
LabelLevelText	<i>Set FontSize to 16</i>	<i>Set Text to "Level:"</i>
LabelLevel	<i>Set FontSize to 16</i>	<i>Set Text to 0</i>
LabelLevelScore	<i>Set FontSize to 16</i>	<i>Set Text to 0</i>
LabelTime	<i>Set Text to "Time"</i>	<i>No other changes required.</i>

Table 3. All of properties for components for the Get the Gold app

Bus stop reached !

We have finished our first iteration (reached our first bus stop). At this point we should have an App which looks like our finished game, but nothing happens yet

!



In the next iterations we will add the 'smarts', so when we do something in the game, something happens.



Iteration 2 Add Moving Ship

Moving the Pirate

To move the PirateSprite, we want the user to be able to tilt the phone and thus move the sprite in the direction that they choose. To do this, we will use the `OrientationSensor.OrientationChanged` event handler like in Figure 6.

You may notice that `OrientationSensor.OrientationChanged` provides 3 attributes: azimuth, pitch and roll.

If you want to know more about this, look at this [tutorial](#) about this and other sensors. We are only interested in something that the OrientationSensor calculates for us, the angle the phone points to.

We want to reassign PirateSprite's current heading to the heading given to us from `OrientationSensor.OrientationChanged`. This means that the user can now control the direction of the pirate ship by tilting the phone.

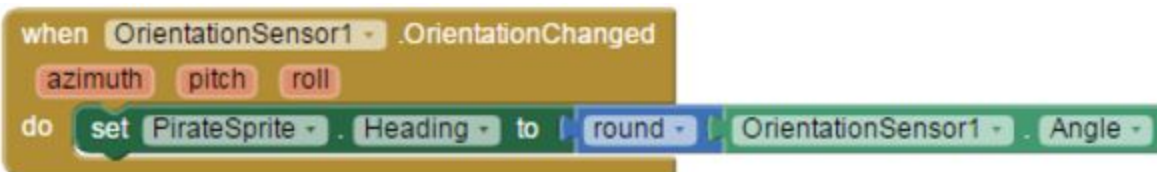


Figure 6. Orientation Component used in Get the Gold app

To prevent the pirate from moving off the screen, we will also use `PirateSprite.Bounce` when an edge is reached like in Figure 7.



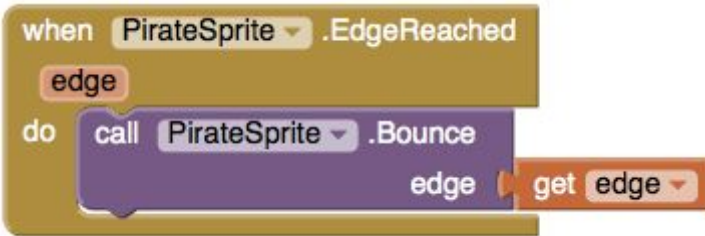
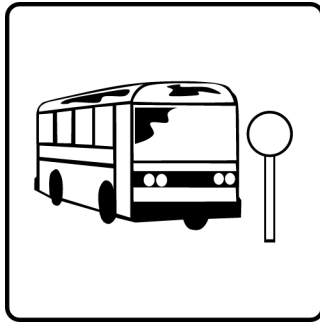


Figure 7. *PirateSprite.EdgeReached*.

Bus stop reached !



We have finished our second iteration (reached our second bus stop).

So now we should have a App which looks like our finished game (from the first iteration), and we can tilt and the ship follows our tilting but bounces off the edges of the screen so it does not disappear !

Iteration 3 Add Moving Coins

Moving the Coins

We want the coins to move to random positions on the screen. We will use `Clock1.Timer` and the ImageSprite's `MoveTo` method to do this.

When the `Clock1.Timer` goes off, we want all of our gold coin ImageSprites to move to a new random location on the Canvas. We will do this by using the `Sprite.MoveTo` block.

If you want to know about the Timer, see the [Bottle Flip tutorial](#).

`MoveTo` takes in two arguments: the x and y coordinates on the canvas of the new position we want the sprite to move to. We want the Sprite to move to a new *random* location so we will use the `random integer` block found in



CoderDojo Castleknock

the Math box. Since we want each Gold ImageSprite to move to a new location, we repeat this process for each sprite's MoveTo function.

For ImageSprite2, we want x to be a random integer from 0 to $Canvas1.Width - ImageSprite2.Width$ and y to be a random integer from 0 to $Canvas1.Height - ImageSprite2.Height$. This is to be repeated for all the Gold Image Sprites.

Remember that sprites are measured at the upper left corner as (0,0) so if we don't want them to go off the screen, we need to take the sprite's height/width into account when setting the range for our random numbers.

We will do this by setting up our blocks as in the top orange rectangle in the image below:



```

when Clock1.Timer
do
  call ImageSpriteGoldCoin2.MoveTo
  x random integer from 0 to Canvas1.Width - ImageSpriteGoldCoin2.Width
  y random integer from 0 to Canvas1.Height - ImageSpriteGoldCoin2.Height

  call ImageSpriteGoldCoin3.MoveTo
  x random integer from 0 to Canvas1.Width - ImageSpriteGoldCoin3.Width
  y random integer from 0 to Canvas1.Height - ImageSpriteGoldCoin3.Height

  call ImageSpriteGoldCoin4.MoveTo
  x random integer from 0 to Canvas1.Width - ImageSpriteGoldCoin4.Width
  y random integer from 0 to Canvas1.Height - ImageSpriteGoldCoin4.Height

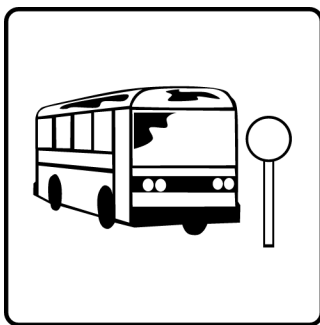
  call ImageSpriteGoldCoin5.MoveTo
  x random integer from 0 to Canvas1.Width - ImageSpriteGoldCoin5.Width
  y random integer from 0 to Canvas1.Height - ImageSpriteGoldCoin5.Height

  call ImageSpriteGoldCoin6.MoveTo
  x random integer from 0 to Canvas1.Width - ImageSpriteGoldCoin6.Width
  y random integer from 0 to Canvas1.Height - ImageSpriteGoldCoin6.Height

  set global Time to get global Time - 1
  set SliderTimer.ThumbPosition to get global Time
  if get global Time = 0
  then
    call NotifierGameOver.ShowChooseDialog
    message Game Over
    title Get the Gold
    button1Text ok
    button2Text
    cancelable true
  
```

Figure 8. Clock1.Timer

Bus stop reached !



We have finished our third iteration (reached our third bus stop).

So now we should have a App which;

- looks like our finished game (from the first iteration).

- we can tilt and the ship follows (from the second iteration).
And keeps moving coins to random locations (from this iteration).

Iteration 4 Add collect Coins

Detecting Collisions

App Inventor detects collisions by checking for an intersection between the bounding rectangles of each ImageSprite. We call this rectangle-based collision detection. As you can see in the image below, sprites with circular or polygon shape will appear to collide because of the rectangular bounds around them when they might not actually be colliding.

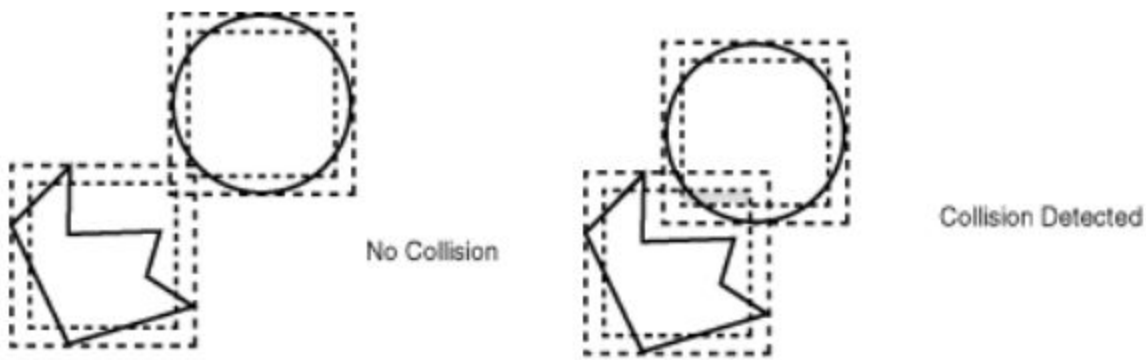


Figure 9. Collision detection in App Inventor.

We can use the `PirateSprite.CollidedWith` event handler to detect whenever the pirate ship collides with another sprite or gold coin. You may notice that `PirateSprite.CollidedWith` takes in an argument. This argument is the object that PirateSprite just collided with. We will be testing inside the handler for which object so the name of this argument is not significant. You can name it other.

Whenever the pirate collides with a gold coin, we want the coin to disappear. We can do this by setting the coin's visibility to false. To find



which coin the pirate collided with, we will use the `PirateSprite.CollidingWith`.

We can use `PirateSprite.CollidingWith` to take in a component (each of the gold coin sprites) to detect which sprite was hit. This is a component block and NOT a text block with the words ImageSprite inside. The component block can be found in the drawer for each component. If a sprite was hit, we will;

- set its visibility to false
- increase the score

The above is done in the top orange box in Figure 10.



Figure 10. When `PirateSprite.CollidedWith`

Tip.

We can ignore the yellow box for now, we will come to that in the next iteration, all levels.

We create a global variable called score to keep track of the score. From the Built-in drawer, drag out the **initialize global(name)** from variables and change name to score and set it to 0 like figure 11.



CoderDojo Castleknock

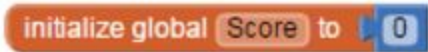


Figure 11. Creating Global variable Score.

If we use code in a number of places it's a good sign that we should create a procedure and call the procedure each time we want to run the code.

So we create the procedure **AddScore** by dragged out from the procedure Drawer an empty to procedure do block like figure 12.



Figure 12. Creating Procedure

We then change the text procedure to AddScore, and add code so it looks like figure 13.

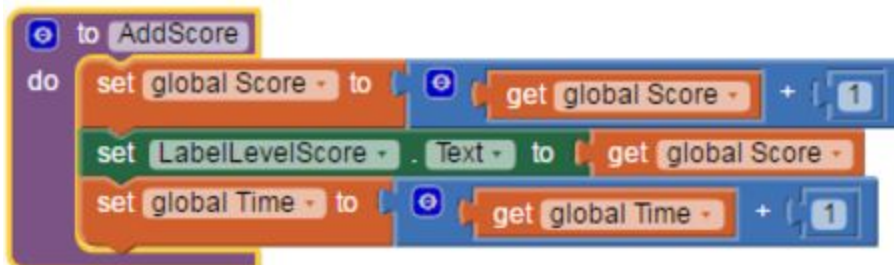


Figure 13. AddScore Procedure

- To add 1 to the score
- Update the LabelLevelScore with the new score
- Give the player additional time as a reward for getting the coin (ssh, thats a secret !)

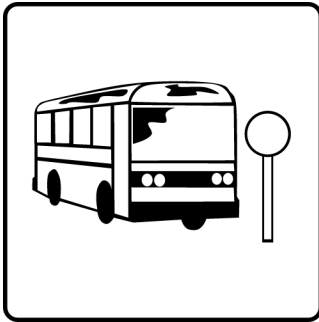
Once we have created the AddScore procedure, we can drag a block like figure 14 by dragged out from the procedure Drawer and adding it to every time we get a coin like in Figure 10.





Figure 14. Calling Add Score procedure

Bus stop reached !



We have finished our fourth iteration (reached our fourth bus stop - nearly there !).

So now we should have a App which;

- looks like our finished game (from the first iteration).
 - we can tilt and the ship follows (from the second iteration).
 - Keeps moving coins to random locations (from third iteration).
- And detects if the ship collects a coins (from this iteration).

Iteration 5 Add Levels

New Level

From *Figure 10*. When *PirateSprite.CollidedWith* looking at the yellow box which we will look at in *Figure 15* below.



Figure 15. Checking for end of level.

We create a global variable called level to keep track of the level. From the Built-in drawer, drag out the **initialize global(name)** from variables and change name to score and set it to 0 like figure 16.



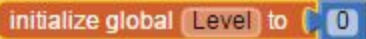
A Scratch code block with a yellow background and a blue tab. The text reads "initialize global Level to" followed by a blue input field containing the number "0".

Figure 16. Creating Global variable Score.

We check if all the coins are visible and if they are not, we tell the play using the notifier that the level is completed. When the player clicks “ok”, the notifier.

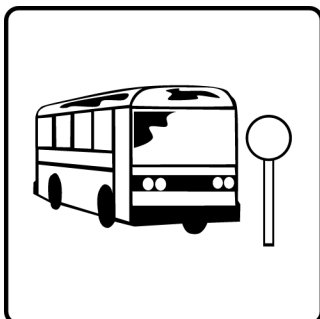


Figure 17. Notifier1.AfterChoosing when player completes a level.

The Notifier1.AfterChoosing event, i.e. Figure 17 is called. This;

- Resets all ImageSpriteGoldCoin{2-6} visible property to true.
- Adds 1 to a variable called Level.
- Set the Text property of LabelLevel from the variable Level

Bus stop reached !



We have finished our fifth iteration (reached our fifth bus stop).

So now we should have a App which;

- looks like our finished game (from the first iteration).

CoderDojo Castleknock

- we can tilt and the ship follows (from the second iteration).
- Keeps moving coins to random locations (from third iteration).
- Detects if the ship collects a coins (from the fourth iteration).

And introduce Levels (in this iteration)

Iteration 6 Add Time limit and Game Over

Starting Game

When we start Get the Gold, we use the **When Screen1.Initialize** like figure 18.

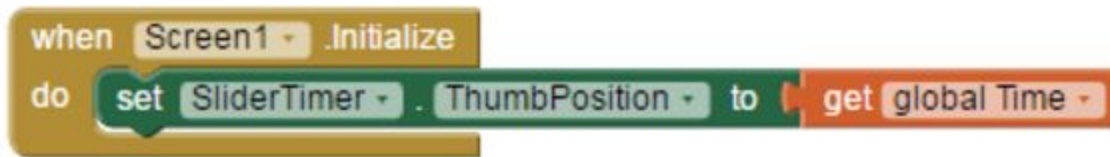


Figure 18. When Screen1.Iniatialize used in Get the Gold app

We set the slider to show the time left by setting it to the Global time.

Tip

When we use numbers to set variables, it's a good idea to create variables for the static variables so it's easier to change later by simply changing the variables. An example of this is in Figure 19.

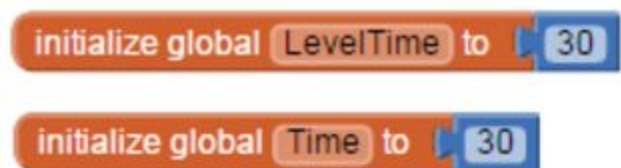


Figure 19 Using static variables



CoderDojo Castleknock

Here to set the Time to complete the Game as LevelTime to 30. If we want to modify this later, we can set the Variable Time to LevelTime when we initialize the application like figure 20.



Figure 20 Example of Setting how much time we have for Get the Gold.

Game Over

If we have no more time we say tell the player using the NotifierGameOver notifier and then close the application by dragging the close application block from Built-in Control menu on the blocks tab.

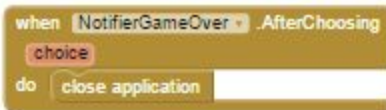


Figure 21. NotifierGameOver.AfterChoosing

Reset Button

After the user hits all of the gold sprites with the pirate ship, none of them will be visible. The reset button should set all of the gold sprites' visibility to true like in Figure 22.



Figure 22. ResetButton.Click



Bus stop reached !

We have finished our sixth iteration (reached our sixth bus stop).

So now we should have a App which;

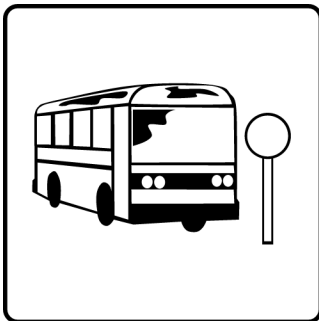
- looks like our finished game (from the first iteration).
- we can tilt and the ship follows (from the second iteration).
- Keeps moving coins to random locations (from third iteration).
- Detects if the ship collects a coins (from the fourth iteration).
- Introduce Levels (from the fifth iteration)

And keeps track of a time limit for the game and has a game over.

You can get off the Bus now or look at the variations in the next iteration !

Iteration 7 Add Variations - over to you !

Variations



Once you get this program running, you may want to do the following additional features to extend it. For example,

- Change the speed of the ship or gold coins
- Add an enemy sprite that when collided with, causes your pirate to lose speed
- How do you reduce time to complete game ?
- How do you give a time bonus for collecting a coin ?
- How do you change time between coin position changes ?
- Oops there is a mistake in the code, when we click the reset button, we do not reset the time, how do we fix this ?
- Add sound effects for
 - Getting a coin

- Finishing a level



Ps

We created this tutorial standing on the shoulders of giants. We want to acknowledge the original Get the Gold at <http://appinventor.mit.edu/explore/ai2/get-gold.html>.

I see a mistake !

If you see a mistake, email coderdojocastleknock@gmail.com so we can fix this tutorial.



Appendix 1.

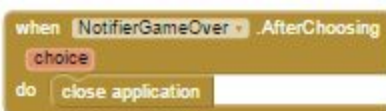
Variables

- score
- level
- level_time
- time



Built-in blocks

- When Screen1 Initialize
- When Clock1.Timer
- When Notifier1.AfterChoosing
- When.ResetButton.Click
- When OrientationSensor1.OrientationChanged
- PirateShip.CollidedWith
- When NotifierGameOver.AfterChoosing



CoderDojo Castleknock

```
when Clock1.Timer
do
  call ImageSpriteGoldCoin2.MoveTo
  x random integer from 0 to Canvas1.Width - ImageSpriteGoldCoin2.Width
  y random integer from 0 to Canvas1.Height - ImageSpriteGoldCoin2.Height
  call ImageSpriteGoldCoin3.MoveTo
  x random integer from 0 to Canvas1.Width - ImageSpriteGoldCoin3.Width
  y random integer from 0 to Canvas1.Height - ImageSpriteGoldCoin3.Height
  call ImageSpriteGoldCoin4.MoveTo
  x random integer from 0 to Canvas1.Width - ImageSpriteGoldCoin4.Width
  y random integer from 0 to Canvas1.Height - ImageSpriteGoldCoin4.Height
  call ImageSpriteGoldCoin5.MoveTo
  x random integer from 0 to Canvas1.Width - ImageSpriteGoldCoin5.Width
  y random integer from 0 to Canvas1.Height - ImageSpriteGoldCoin5.Height
  call ImageSpriteGoldCoin6.MoveTo
  x random integer from 0 to Canvas1.Width - ImageSpriteGoldCoin6.Width
  y random integer from 0 to Canvas1.Height - ImageSpriteGoldCoin6.Height
  set global Time to get global Time + 1
  set SliderTimer.ThumbPosition to get global Time
  if get global Time = 0
  then
    call NotifierGameOver.ShowChooseDialog
    message Game Over
    title Get the Gold
    button1Text ok
    button2Text
    cancelable true
```

CoderDojo Castleknock

```
when ResetButton . Click
do
  set ImageSpriteGoldCoin2 . Visible to true
  set ImageSpriteGoldCoin3 . Visible to true
  set ImageSpriteGoldCoin4 . Visible to true
  set ImageSpriteGoldCoin5 . Visible to true
  set ImageSpriteGoldCoin6 . Visible to true
```

```
when Notifier1 . AfterChoosing
  choice
do
  set global Level to (get global Level + 1)
  set LabelLevel . Text to (get global Level)
  set ImageSpriteGoldCoin6 . Visible to true
  set ImageSpriteGoldCoin5 . Visible to true
  set ImageSpriteGoldCoin4 . Visible to true
  set ImageSpriteGoldCoin3 . Visible to true
  set ImageSpriteGoldCoin2 . Visible to true
```

```
when OrientationSensor1 . OrientationChanged
  azimuth pitch roll
do
  set PirateSprite . Heading to (round (OrientationSensor1 . Angle))
```

```
when PirateSprite . EdgeReached
  edge
do
  call PirateSprite . Bounce
  edge (get edge)
```

```
when Screen1 . Initialize
do
  set SliderTimer . ThumbPosition to (get global Time)
```



```

when PirateSprite CollidedWith
  other
do
  call PirateSprite CollidingWith other ImageSpriteGoldCoin2
then
  call AddScore
  set ImageSpriteGoldCoin2 Visible to false
do
  call PirateSprite CollidingWith other ImageSpriteGoldCoin3
then
  call AddScore
  set ImageSpriteGoldCoin3 Visible to false
do
  call PirateSprite CollidingWith other ImageSpriteGoldCoin4
then
  call AddScore
  set ImageSpriteGoldCoin4 Visible to false
do
  call PirateSprite CollidingWith other ImageSpriteGoldCoin5
then
  call AddScore
  set ImageSpriteGoldCoin5 Visible to false
do
  call PirateSprite CollidingWith other ImageSpriteGoldCoin6
then
  call AddScore
  set ImageSpriteGoldCoin6 Visible to false
do
  call PirateSprite CollidingWith other ImageSpriteGoldCoin7
then
  call AddScore
  set ImageSpriteGoldCoin7 Visible to false
do
  call PirateSprite CollidingWith other ImageSpriteGoldCoin8
then
  call AddScore
  set ImageSpriteGoldCoin8 Visible to false
do
  not ImageSpriteGoldCoin2 Visible and not ImageSpriteGoldCoin3 Visible and not ImageSpriteGoldCoin4 Visible and not ImageSpriteGoldCoin5 Visible and not ImageSpriteGoldCoin6 Visible and not ImageSpriteGoldCoin7 Visible and not ImageSpriteGoldCoin8 Visible
then
  call (Notifier) ShowChooseDialog
  message Next Level
  title Get the Gold
  button1Text GO
  button2Text
  button3Text
  cancelable false
  
```

Procedure Blocks

- To AddScore
- To ResetTime

```

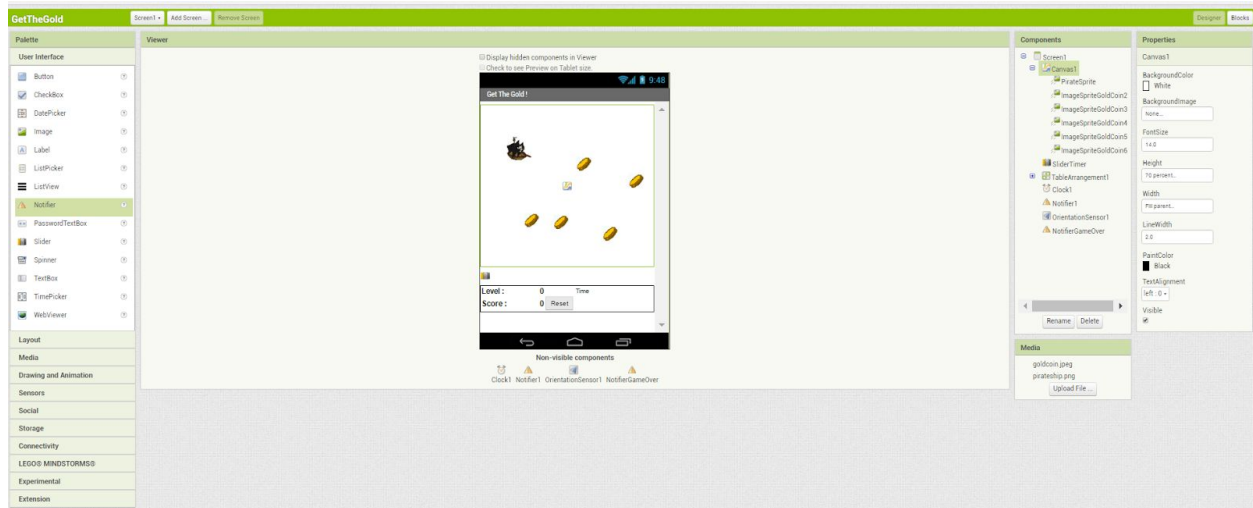
to AddScore
do
  set global Score to get global Score + 1
  set LabelLevelScore Text to get global Score
  set global Time to get global Time + 1
end

to ResetTime
do
  set SliderTimer MaxValue to get global LevelTime
  set SliderTimer MinValue to 0
end
  
```

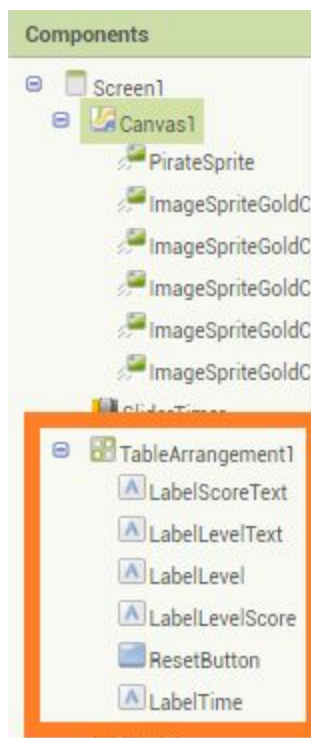


CoderDojo Castleknock

Designer



There are some components in the table, so we will look at them separately;





Source

[Get the Gold](#)

Download this to your computer and then upload it to your app inventor account. To do this ;

- Click on the bottle flip app
- When you run this, the source code shown downloads to your laptop where it can be then uploaded to you App Inventor tab via;

Projects -> Import Project (.aia) from my computer...

Assets

Assets are what we sometimes call sounds and images used in a game.

([Sample sound and images](#))

