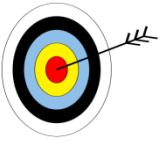



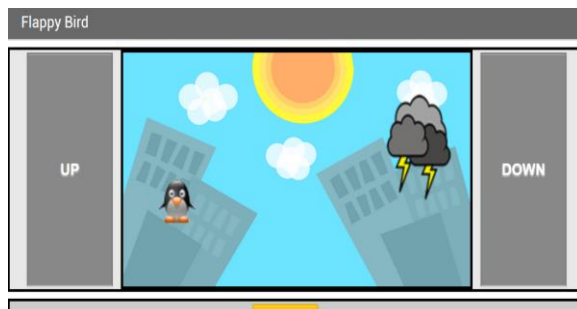


Flappy Bird



So you have decided on **FlappyBird**. FlappyBird is a fun game, where you have to help your **bird**  to dodge the **storm clouds**. This work sheet will help you create an App, which

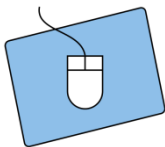
- ... let's you control a 
- ... generates **clouds** , which move towards you.
- ... counts up **points**, if you dodge the cloud.



Level of difficulty: 

Create a project


For every new app, you first have to create a project.



- 1.) Create your own project and give it an appropriate name.
- 2.) Connect the App Inventor with your cell phone, the same way you did for your first app.

The structure of your app

Now you have to think about what your app should look like. Your App needs space for these elements (which you will add to you App one by one):

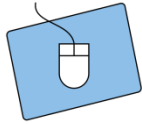
- a **playing field**, on which the  can move,
- a **reset button**, which restarts the game,
- an **UP-** and **DOWN-Button**, with which you can steer the bird.
- a **scoreboard**, which counts up your points.

That's quite a lot of stuff. If you want to be able to fit all of that, you should change the orientation of your screen from **Unspecified** to **Landscape**:

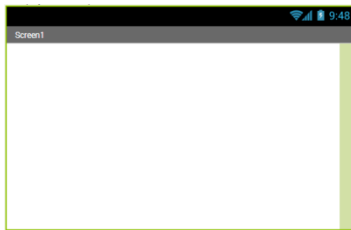
Flappy Bird



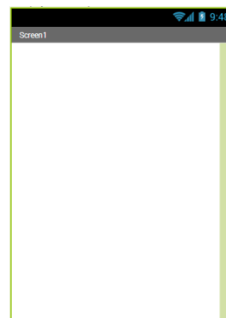
Landscape is a specific direction of your screen, which is horizontal. If you want a vertical display, choose **Portrait**.



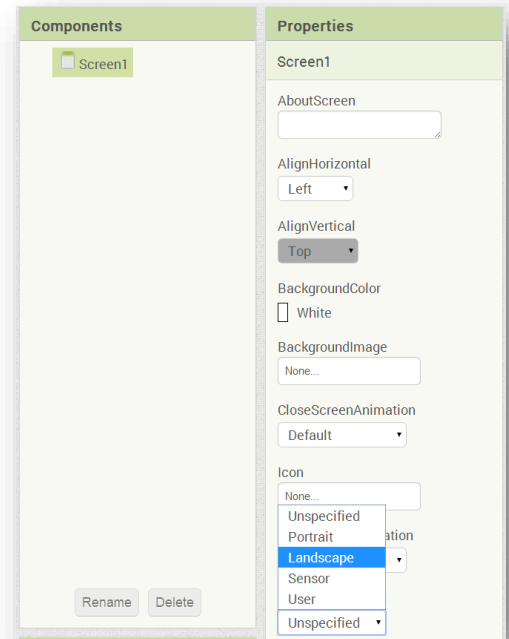
- 1.) Select *Screen1* in the **Components**.
- 2.) Look for *ScreenOrientation* in the **Properties**.
- 3.) Change its value from *Unspecified* to *Landscape*.



Landscape



Portrait



The components of your app


Now you can start adding elements to your app and rearrange them with the help of the **Screen Arrangements**.

- For the **playing field** you need a **Canvas**
 - o You can find the *Canvas* in the *Palette* under *Drawings and Animation*.

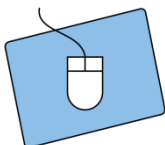
A canvas is a screen on which objects like your  can move.



For the **reset button** you need a **button** with the text „Reset“.

For the **movement** of the  you need two buttons. Label one of them „UP“, the other one „DOWN“.

For the **scoreboard** you need a label with the text „0“.

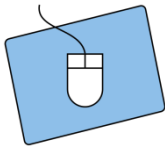


- 1.) Think about how you want to position your elements.
- 2.) Drag and drop the *ScreenArrangements* you need for your layout.
- 3.) Add the components (*Canvas*, *Button* and *Label*) to your app.

Flappy Bird

The first test

Now that you have positioned all components in appropriate spots, you can test how all of that looks on your smartphone.



- 1.) Start the App on your smartphone if you haven't done that already.
- 2.) Do you like the setup? If you do, just continue, otherwise modify it until you like it.



You can keep the App active on your device. The AppInventor updates any changes you make and displays them immediately, so that you don't always have to restart the App.

Don't forget to change the name ;)

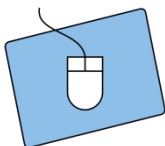
Now that all the important components of your App are in place it is time to give them fitting names, so you can distinguish them in the Blocks-Editor.

A background for your playing field

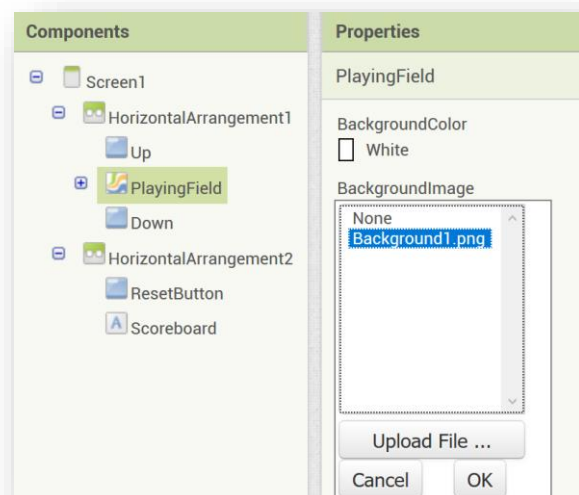
You should add a background to your playing field so that the 🐧 feels right at home. Upload one (out of two possible) backgrounds from this folder:

Desktop / InfoSphere goes Android / FlappyBird

Afterwards you have to assign the background to your canvas (e.g. PlayingField):

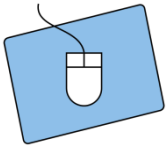


- 1.) Select the canvas under Components.
- 2.) In the Properties click on BackgroundImage and assign the background.png to the canvas.

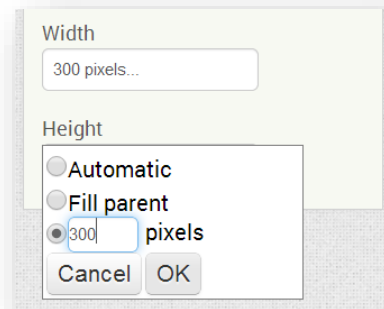


As you know, sometimes you need to adjust the size of your elements by hand.

Flappy Bird

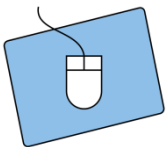



- 1.) The canvas is still selected.
- 2.) Under Properties look for Width and Height and change both to 300 pixels.
- 3.) Test it on your App and change the number of pixels, if necessary, until it looks perfect on your smartphone.



The Bird

Next, we want to add the  to the playing field.

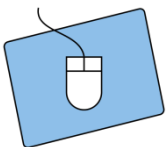



- 1.) Go to **Drawing and Animation** in the **Palette**.
- 2.) Drag an ImageSprite to the field.
- 3.) Change the name of the ImageSprite.
- 4.) Use the same folder as before and upload the  and assign it to the ImageSprite.



An ImageSprite is a special kind of image. Other than normal images, an ImageSprite can move on the canvas.

Your bird now has a default position on the playing field. Under properties you can get its position from the **X- and Y- coordinates** and you can set the position there as well.



- 1.) Select the bird under Components.
- 2.) Change the X- and Y-coordinates in the Properties and find a nice starting location for the .

Flappy Bird

A cloud

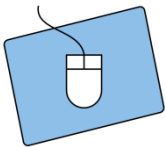
Next we need a . Follow the same steps as before to create a on the playing field. Don't forget the image and the position of the .

A timer for your clouds

The last element you need is a **Clock**. You need a clock, so that the can move around. You can find the clock in the palette **Sensors**.



A **Clock** is good for many different things. For instance you can set a **TimeInterval**. After a set period of time, a **TimeInterval** repeatedly triggers the function **Clock.Timer**.



- 1.) Drag a **clock** to your app.
- 2.) Select the **Timer** under *Components* and change the value of **TimeInterval** to 500 ms (500ms is equal to 0.5 seconds).

Properties

Clock1

TimerAlwaysFires



TimerEnabled

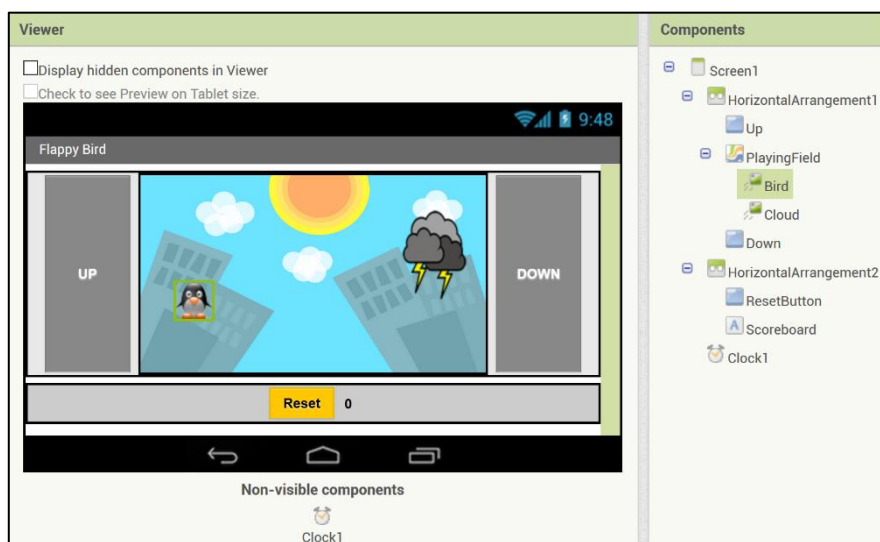


TimeInterval

500

The intermediate result

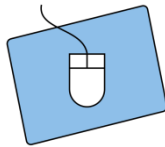
Your app should now look similar to this. If you still have any questions, don't hesitate to ask an instructor.



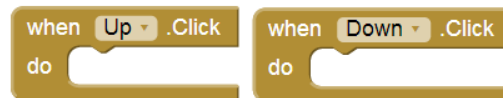
Flappy Bird




The movement of the bird

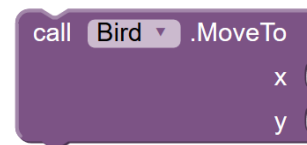
The bird can move up and down in this game. Therefore you have to include these two directions in your app.



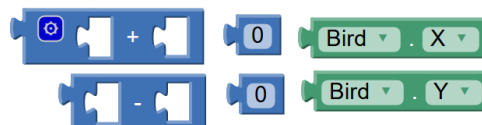
- 1.) Switch to the Blocks Editor.
- 2.) Select the **when xx.Click-Block** for the UP and DOWN button.



- 3.) If your  moves up and down, its **y-coordinate** is changed. The point of origin (0,0) of your coordinate system is in the top left corner of the canvas. That means:
 - a. If you move the  up, the y-coordinate has to **decrease**.
 - b. If you move the  down, the y-coordinate has to **increase**.
- 4.) Now you need this method:




- 5.) You now know everything you need to move the bird up and down. You also need these blocks:

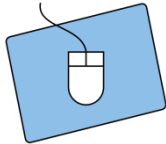


- 6.) Try to build in these blocks yourself. If you're having trouble, don't hesitate to ask an instructor.
- 7.) Don't forget to always test your app 😊

Flappy Bird


The movement of the cloud

For the movement of the clouds, you need a clock, or rather the **TimeInterval** that you've set up. Every time the timer is triggered, the  should move to the left.




- 1.) Switch to the Blocks Editor.
- 2.) Select the method **when Clock1.Timer:**

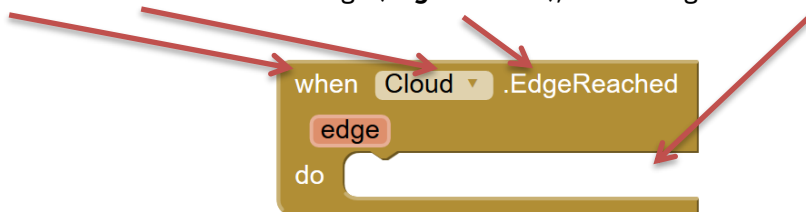


- 3.) Now it's similar to the movement of the bird:
 - a. The y-coordinate should stay the **same**.
 - b. The x-coordinate should **decrease**, so that the  moves to the left.
- 4.) Try to build in these blocks yourself. Again: Don't hesitate to ask an instructor if you need help.
- 5.) Hint: If your cloud moves too slowly, you have to change the **TimeInterval** in the Design-Editor.

Making new clouds appear



Every time the  hits the left border of the canvas, a new cloud should appear. You need these blocks for that:


When the cloud reaches the edge (EdgeReached), something can be done:



Drag this block to your **work space**.

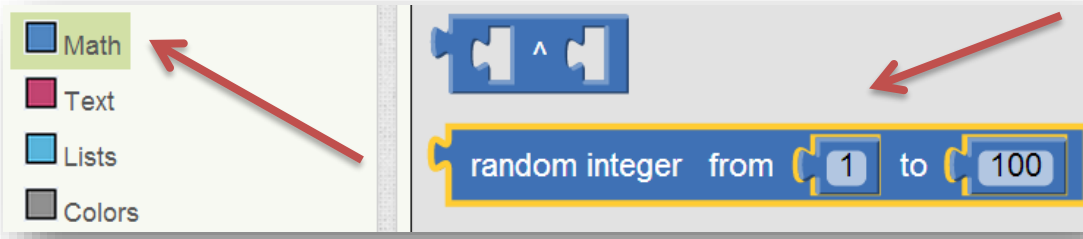
To have a **new cloud** appear, you will use a little trick:

- You just select a new position for the  at the right part of the screen.
- The player will think that the old cloud is gone and a new cloud has appeared.
- Don't forget that only the **y-coordinate** of the  should be random. That means that **x-coordinate** should always stay the same.

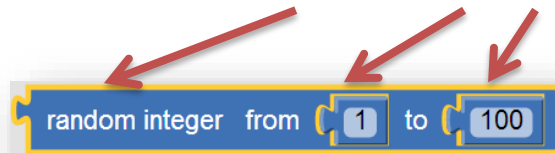
So the new **position** of the  should be **random**. The App Inventor can generate a **random number**, which you can use for your cloud.

You can find that method in the **BlocksEditor** under **Built-In-> Math**.

Flappy Bird



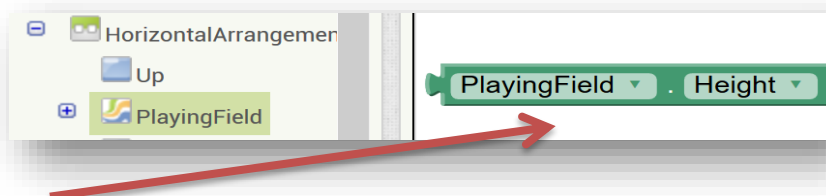
This method chooses a **random whole number** from **a** to **b**, in this case from 0 to 100.



The range of the random numbers:

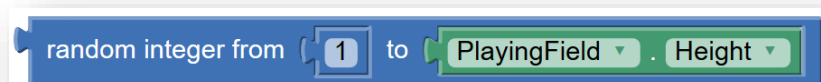
The range of these numbers should be **as wide as your field**.

You can get the height of the field like this:

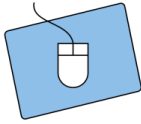


PlayingField.Height gives you the height (y-coordinate).

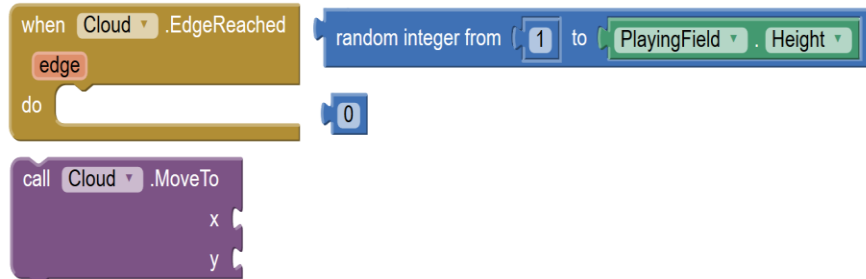
This is how the combined blocks should look like (for the height):



Flappy Bird



- 1.) Now come up with the random location of the cloud.
- 2.) Use these blocks:

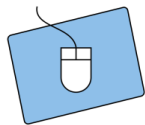


- 3.) Test your app.

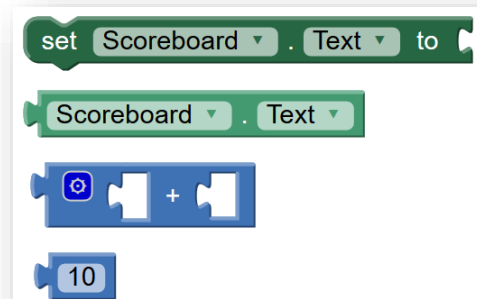
Awesome! We got a lot done. We're almost at the finishing line 😊

Count up the points

Every time a cloud reaches the left edge of the playing field, points should be added to the score board. You will now learn how to implement this function.

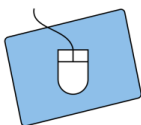


- 4.) Find these blocks:
- 5.) Combine them appropriately.
- 6.) Test your app.



The collision with the cloud

Now you should handle the collisions with the ⚡. If you hit a cloud, that's **game over**. These are the necessary steps:



- 7.) Look for the method **Bird.CollidedWith**, which will be triggered by the collision with the cloud.
- 8.) Add everything to this method that should happen in case of a collision:
 - The speed of the cloud should be set to 0.
 - A text should appear. For that, you can use the method **PlayingField.DrawText**.

Flappy Bird

The reset button

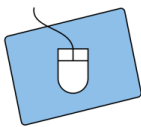
Lastly you have to make sure that the game **can be reset**. That's why you need a **reset button**. If the reset button is pressed, then all basic settings should be restored.



Basic settings

Now you have to consider what the basic settings are. This is what you know:

- 1) The scoreboard has a basic setting.
- 2) The bird can be moved back to its default position.
- 3) The cloud has a starting position.



- 1.) Think about what has to be reset.
- 2.) Implement your ideas with the help of the method **when ResetButton.Click**.



Congratulations 😊

You have successfully programmed the game “Flappy Bird”. On the next page you can find tips, tricks and suggestions on how you can expand your game even further. If you want to continue with your next game, talk to the instructors.

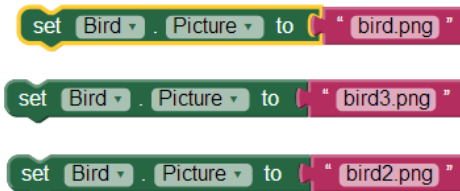
Flappy Bird

Expansions

Here you can find some ideas on how to develop your game further:

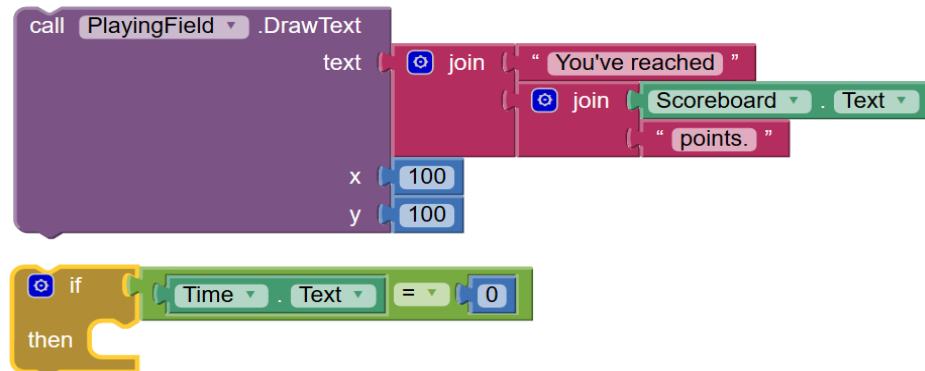
Additional bird and background images

You can set up buttons so that the player can choose between different images for the bird and the background. If the button is pressed, the picture should change:

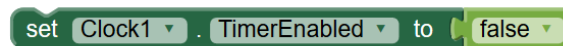
**An end to the game..**

... would be nice. You can create a new label which displays 100 at the start of the game. Every time the timer is triggered, reduce its value by 1.

- If the value reaches 0, you can reset the game or display a text which tells the player how many points he's got.



It's important to switch off the timer so that the cloud stops moving.



Hint: If you press the reset button, the timer has to be switched on again and the text on the playing field has to be removed. Browse through the methods of the playing field, and find out how you can clear it.

List of references:

- <https://pixabay.com/de/pinguin-linux-klein-baby-vogel-48559/>



- <https://pixabay.com/de/wolke-gewitter-blitz-wetter-regen-149323/>



- Source: InfoSphere

any other graphics are screenshots of the App Inventor (<http://appinventor.mit.edu/explore/>)