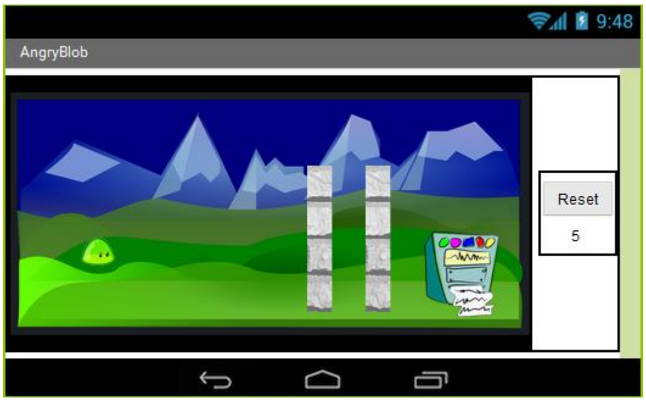
InfoSphere goes Android

Angry Blob

Great that you chose **AngryBlob**! AngryBlob is a fun game where you have to destroy the **super computer** with the help of the "**Blob**" . This work sheet helps you to create an App, which...

... makes a disappear on your screen or rather your **playing field**.
... allows the player to throw the by pressing, dragging and releasing it towards an object (e.g. a super computer).
... finally destroys the or its defence, if they are hit.



Level of difficulty: ★★★★

Create a project

In order to get started with the *MIT App Inventor* connect with its website and **start** a **new project**.

1.) Create your own project and name it appropriately.
2.) Connect the App Inventor with your cell phone.
3.) Start working within the **Designer**.

The structure of your app

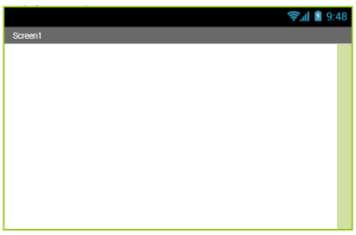
At first think about how your app should look like. Your app needs space for the following elements (which you will add to your app one by one):

- a ***playing field*** (area where moves)
- a ***reset button*** (restarts the game)
- a ***life running display*** (bar which reduces the 's lives after each unsuccessful try).

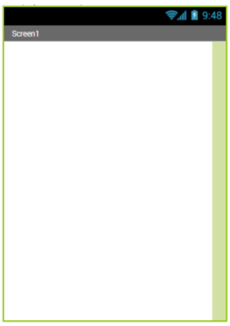
There are many things that have been done. If you want to fit all of the above into your app change the screen orientation from ***Unspecified*** to ***Landscape***.

> ***Landscape*** is defined as horizontal; therefore it's a specific direction. If you want it to be vertical change it to ***Portrait***.

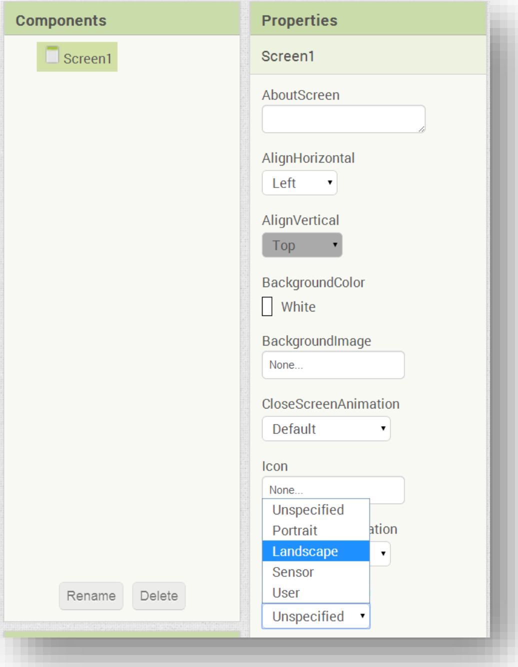
1.) Select *Screen1* in the **Components**.
2.) Look for *ScreenOrientation* in the **Properties**.
3.) Change it from *Unspecified* to *Landscape*.



Landscape



Portrait



Components
Screen1
Rename Delete

Properties
Screen1
AboutScreen
AlignHorizontal
Left
AlignVertical
Top
BackgroundColor
White
BackgroundImage
None...
CloseScreenAnimation
Default
Icon
None...
Unspecified
Portrait
Landscape
Sensor
User
ation
Unspecified

The components of your app

Therefore 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***
  - You will find the **Canvas** in the *Palette* within *Drawing and Animation*.

> A **canvas** is defined as an area on which objects like your 🟢 can move.
> For the ***reset button*** you need a ***button*** with the text „Reset“.
> For the ***life bar*** you need a ***label*** with the text „5“, if your 🟢 should have a total of 5 lives (you are free to decide how many lives your 🟢 should have).

1.) Think about where you want to place your elements.
2.) Choose the *ScreenArrangements* you'll need for your **layout**.
3.) Add all components (*Canvas*, *Button* and *Label*) to your app.

The first test

After that you should have positioned all the necessary components in appropriate spots then you should test how it's displayed on your smartphone.

1.) If it isn't already running, start the app on your smartphone.
2.) Do you like the setup? If you do just continue; otherwise change it until you like it.

The app stays active on your phone. 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 appropriately names. This way you will have no problem to distinguish between them in the **Blocks-Editor**.

A background for your playing field

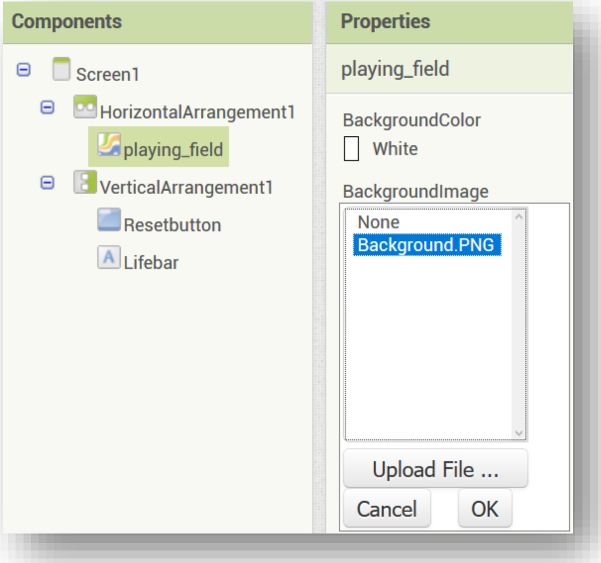
In the following add an area to your playing field to make the ● feel right at home. Upload one (out of two possible) backgrounds from this folder:

Desktop / InfoSphere goes Android / AngryBlob

After that you have to assign the background to your canvas:

1.) Select the *canvas* in Components.

2.) In the Properties click on *BackgroundImage* and assign the Background.PNG to the canvas.



Components

- Screen1
  - HorizontalArrangement1
    - playing_field
  - VerticalArrangement1
    - Resetbutton
    - Lifebar

Properties

playing_field

BackgroundColor
□ White

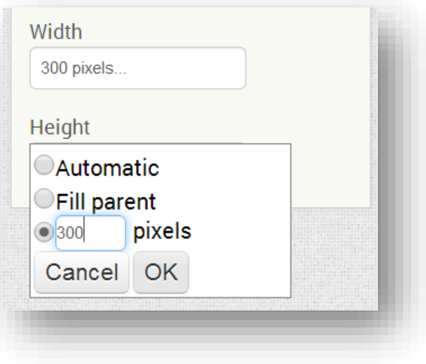
BackgroundImage
None
Background.PNG

Be careful! Sometimes you'll need to adjust the size of your elements by hand.

1.) The *canvas* is still selected.

2.) In *Properties* look for *Width* and *Height* and change both to *300 pixels*.

3.) Test it and change the number of pixels -if necessary- until you like it.



Width

300 pixels...

Height

○ Automatic
○ Fill parent
● 300 pixels

The Blob

In addition to that, we want to add the 🟢 to the playing field.

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

An ***ImageSprite*** is a special kind of image with the unique ability to move on a canvas.

The 🟢 is in default position on the playing field. In **Properties** you can get it’s ***X-*** and ***Y-*** coordinates and also change it.

1.) Select the Blob in Components.

2.) Change the X- and Y-coordinates in the Properties and find a nice starting point for the Blob.

The supercomputer and the walls

Besides the you'll need some other **ImageSprites**:

- an *ImageSprite* for the Supercomputer and
- **two** *ImageSprites* for the walls.

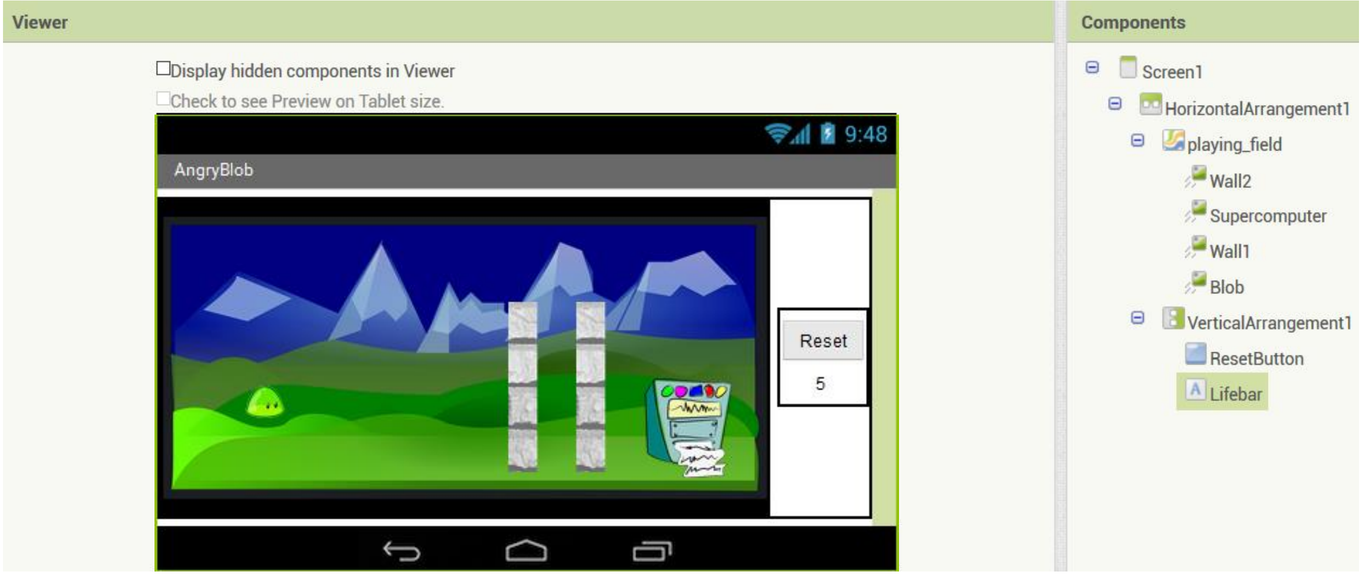
    - **Hint**: If you want to rotate the walls, you have to change the value of "**Heading**" in the Properties. However the result of the rotation is only visible on your smartphone. (Try changing the value to 45 and 90.)

1.) Add the ImageSprites to a reasonable position just as you did in the last section.

2.) Make sure to check the layout on your smartphone.

The intermediate result

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



Viewer

□Display hidden components in Viewer

□Check to see Preview on Tablet size.

Components

- Screen1
  - HorizontalArrangement1
    - playing_field
      - Wall2
      - Supercomputer
      - Wall1
      - Blob
    - VerticalArrangement1
      - ResetButton
      - Lifebar

That was quite a task so far ☺ .
Keep it going!

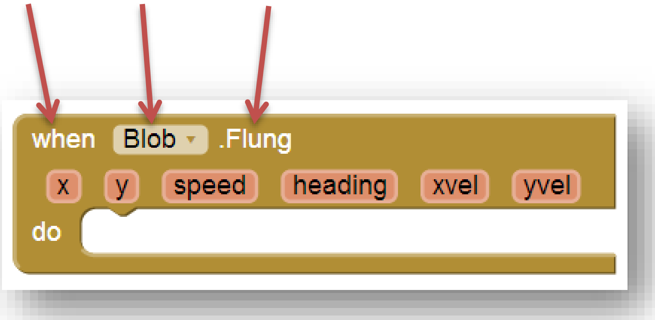
The movement of the Blob

At first the should start flying by flinging it.

1.) Change into the **Blocks Editor**.

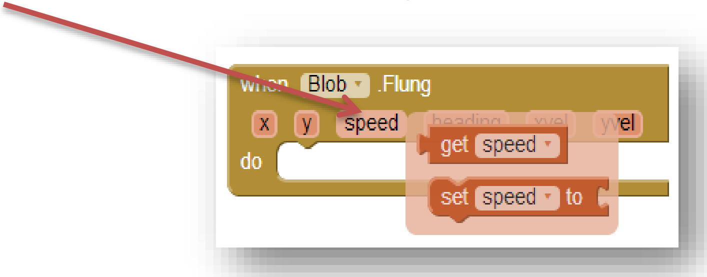
2.) Select ***Blob*** and choose the block *when Blob Flung*.

When the is **flung**, something can be **done**.



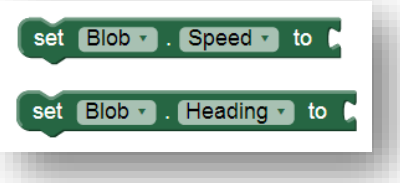
```
when Blob ▾ .Flung
  x  y  speed  heading  xvel  yvel
do
```

With this function you can also set variables which are needed for example to set **heading** or **speed**. You can select them by moving your mouse to the names of the variables.



```
when Blob ▾ .Flung
  x  y  speed  heading  xvel  yvel
do
get speed ▾
set speed ▾ to
```

3.) Assign new values to **speed** and **heading** of the Blob now:



```
set Blob ▾ . Speed ▾ to
set Blob ▾ . Heading ▾ to
```

4.) Combine the blocks on your own and ask your instructor if you need any help.

5.) Try it out! ☺

Evaluation after each try

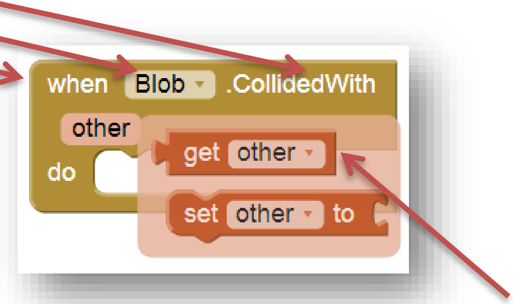
Your turn is over, when the

- collides with one of the ***walls***,
- hits the or
- touches the borders of the ***playing field***.

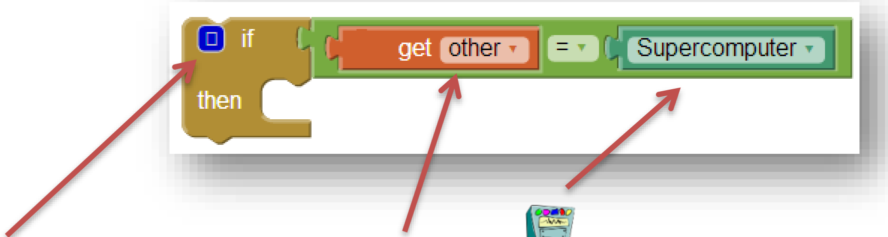
After each throw the app has to check for each of these possibilities. It also has to decide if the game can be continued or if it is over.

Implement these possibilities in the following segments:
For the first two cases (**Wall, Supercomputer**), you have to use this function:

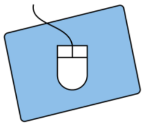
when ***Blob.CollidedWith***:
When the ***Blob*** has ***collided with..*** , something happens.



Additionally to that you will get the new variable ***other*** which stores what the has hit during its collusion. Therefore ***other*** can either be a ***wall*** or the ***supercomputer***.
In the next step you should check which of the three events from the bullet points above did happen. To check if the was hit use the following:



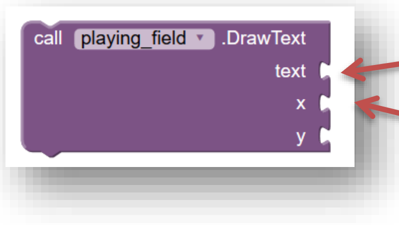
If the component that was hit (***other***) is the , then something can be done.



1.) Create the if-then-construction for the cases and *walls*.
2.) If you need help, don't hesitate to ask someone.

The super computer was hit

If the was hit you have to tell the player that he won. You can use this construction to display a text on the playing field:


call playing_field ▾ .DrawText
text
x
y

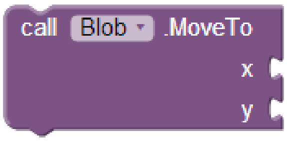
Put the text you want as an output here.

Connect numbers to specify position of textoutput on the playing field.

A wall was hit

If a wall got hit move the to its starting position:

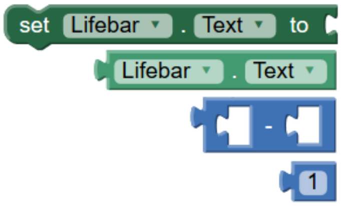
➔ use this block:


call Blob ▾ .MoveTo
x
y

Consider also the following:

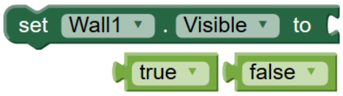
- **player's lives should be decreased by 1**

  ➔ needed blocks:


set Lifebar ▾ . Text ▾ to
Lifebar ▾ . Text ▾
-
1

- wall should be **destroyed**

  ➔ therefore make it invisible:


set Wall1 ▾ . Visible ▾ to
true ▾
false ▾

- check if the player has lost (life = 0)
  - *if* the life equals 0 *then* let a text pop up to tell the player that he has lost and the game is over.

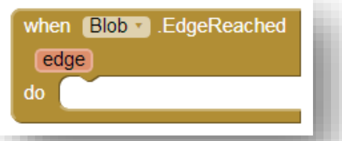
1.) If you haven't already done that yet: Implement the collisions for the and the walls.

2.) Test your app.

Very good! This was a lot of work. You're almost there ;)

The edges were reached

To finalize the programming of the 🟢 movement your last step is to set up a function which checks whether the 🟢 has reached one of the edges or not.



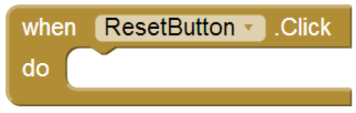
when Blob .EdgeReached
edge
do

If an edge is reached you have to:

- move the 🟢 to its **starting position**
- **decrease the player's lives** by 1
- check whether the player has **enough lives left**

The Reset-button

Finally make sure that your game ***can be reset*** by adding a ***reset button*** to your app. *When* the reset button is clicked *then* all basic settings should be restored.



when ResetButton .Click
do

<u>Basic settings</u>

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

1) The life bar has a basic setting.
2) The position of the 🟢 has to be ***changed to its starting point***.
3) Destroyed components (walls and the super computer) have to be made ***visible again***.

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

Congratulations ☺

You have successfully programmed an Angry-Blob-Game. On the next page you can find tips, tricks and suggestions on how you can extent your game even further. If you want to continue with another game talk to the instructors.

Extension

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

More Walls

Implement more walls at different locations in your playing field to make your game more challenging.

Moving Walls

If you have mastered the basic setup make the walls move. Consider in which area the walls should move and at which point they should change their directions.
Example: the walls move towards the edges of the screen and change their movement direction by 180° if they reach an edge.

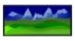
Life total

Increase the life total, if it is too difficult to destroy the with your current amount.

Extra Lives

Add a second to your playing field which grants you an extra life if you hit it.

List of references:

- Source: pixabay.com, Autor: OpenClipartVectors (CC0)
- Source: pixabay.com, Autor: geralt (CC0)
 *- Source: pixabay.com, Autor: OpenClipartVectors (CC0)*
- Source: openclipart.org, Autor: mi_brami (Unlimited Commercial Use)
- Source: InfoSphere
any other graphics are screenshots of the App Inventor (http://appinventor.mit.edu/explore/)