Great that you chose Asteroids! The goal of Asteroids is to move a spaceship through space while dodging small asteroids. This work sheet helps you to create an app, which will include...

- outer space with and ,
- the control for the ,
- implementation of the movement of the and
- counting of the points for the amount of time you’ll survive in space.

Level of difficulty: 🌟🌟🌟🌟

Create a project

In order to get started with MIT App Inventor connect with its website and create 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

Now 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 spaceships move)
- a reset button (restarts the game)
- a scoreboard (which counts up your points).

That’s quite a lot of stuff. If you want to fit all of the above into your app change the screen orientation from Unspecified to Landscape.
InfoSphere goes Android

Asteroids

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.

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 within Drawings 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 scoreboard you need a label with the text „0“.

1.) Think about how you want to position your elements.
2.) Choose the ScreenArrangements that you’ll need for your layout.
3.) Add the components (Canvas, Button and Label) to your app.
The first test

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

1.) Start the app on your smartphone, if it isn’t already running.
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 App Inventor 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. This way you will have no problem to distinguish between them in the Blocks-Editor.

A background for your playing field

Next create the outer space where your can move around. Upload one (out of two possible) backgrounds from this folder:

Desktop / InfoSphere goes Android / Asteroids

Afterwards 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.

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

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 it looks perfect on your smartphone.

The spaceship

Next, we want to add the spaceship 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.
4.) Use the same folder as before to upload the ImageSprite and assign it to the ImageSprite.

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

The spaceship is in default position on the playing field. In Properties you can get its X- and Y-coordinates and also change them.

1.) Select the spaceship in Components.
2.) Change the X- and Y-coordinates in the Properties and find a nice starting point for the spaceship.
The asteroids

Besides the asteroids you need some other ImageSprites to set the direction.

1. Add two ImageSprites for your asteroids.
2. Upload the image.
3. Assign the image to your ImageSprites and position them in a good spot.

A clock for the game

The last element you need to add is a Clock. You need a clock so that your ImageSprites can move around. You can find the clock in the Palette within the category Sensors.

A Clock is useful for many different things. For example to set a TimeInterval. A TimeInterval will repeatedly trigger the function Clock.Timer for a set period of time.

1. Drag a clock to the viewer.
2. In Components select clock and change its TimeInterval value in Properties to 1000ms (1000ms equals 1 second).
The intermediate result

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

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

The heading of the asteroids

An ImageSprite’s heading takes values between 0 and 359 degrees.
The picture to your right gives you an idea of how the angles translate into movement directions of the ImageSprite.

First the should move randomly on the playing field. Therefore two things need to be implemented:

1.) Randomly moving head constantly towards a new destination.

2.) touching the edges of the playing field, bounce off and keep on flying.
1.) Every ImageSprite has a preset heading. Look up the default settings of your ImageSprite in Properties.

2.) Change into Blocks-Editor.

3.) When the ImageSprite has reached an edge, it should bounce off of it and instantly change its direction. Select asteroid1 (example) and choose the block when asteroid1 edge reached.

   When Asteroid1 reached an edge, then something can be done.

   (Use this function for every asteroid.)

4.) For the heading you need these blocks:

5.) Combine both blocks and add a heading angle (0-359).

6.) Do the same for your second ImageSprite and test your app.

But something is still missing, for example that your ImageSprite won’t move correctly.
InfoSphere goes Android

Asteroids

The speed of the asteroids
So far you've only set the heading of your ◆, but their speed setting is still missing. To simplify matters set a default speed directly in Properties.

1.) Change the speed to a random value and test your app.
2.) Watch the movements of the ◆ and readjust their speed, if necessary.

The random direction of the asteroids
To get the ◆ to move randomly after they have reached the edges of the playing field, you need to set a random integer. Find the appropriate function in the Blocks-Editor in Built-in -> Math.

This function chooses an random integer from a to b, in this case between 1 and 100.

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

1.) Think about:
   a. Where you should implement the random numbers.
   b. What the range of these numbers should be.

2.) Try to come up with it yourself and frequently test your app.

If you're having difficulties feel free to ask your instructors!
InfoSphere goes Android

Asteroids

The control of the spaceship

The 🚀 can be controlled by touching the screen. Therefore two things are necessary:

1.) The touch of your finger or the coordinates of the touch must be recognized.
2.) The 🚀 must orientate itself towards these coordinates (Heading) and then move to their directions (Speed).

In order to get any touch on the playing field recognized use this function:

When the playing field has been touched up (contact is released), the function recognizes the x- and y-coordinates and then does something with them. To use the coordinates select them by moving your mouse to the X or Y and choose get.

Now you need the function point in direction to put in the coordinates. Find this function within your Spaceship block.

1.) Set the speed of your 🚀 in the Properties.
2.) Use the function above and change the heading of the 🚀 to the x- and y-coordinates of the touch.
3.) Test your app. If the steering is working everything is going as planned.

Awesome! You’ve got it. We’re almost at the finish line 😊
Increase the scoreboard/timer

To increment the time or points, you have to make sure that the label of the scoreboard is changed with the help of the function `Clock.Timer`. To increment the time increase the numerical value of the Label each time by 1.

1.) Use the blocks on the right.

2.) Make sure that the seconds increment correctly.

3.) Test your app.

Collision with an asteroid

Now let’s have a look at our collision with an asteroid. Consider the following:

1.) The app has to check whether a collision has happened or not.

2.) If a collision did happen the spaceship has to explode.

An ImageSprite can check whether it has collided with another ImageSprite or not.

When the spaceship collided with something then something can be done.

As soon as this happens the picture of the spaceship should change to an explosion and the speed of all ImageSprites should be set to 0.

1.) Upload one out of the explosions from the folder.

2.) Look for a function in the Blocks-Editor which can change a picture.

3.) Attach a specific text to this function. This text should have the name of your explosion (e.g. „explosion1.png”).

4.) Set the speed of all ImageSprites to 0.

5.) Test your app.
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 pressed then all basic settings should be restored.

```
when ResetButton.Click
  do
```

**Basic settings**

Now think about your basic settings. Is everything in order? This is what you know:

1) The scoreboard has a basic setting.
2) The has a default position.
3) The have default positions.

1.) Think about what has to be reset.
2.) Implement your ideas with the help of the when ResetButton.Click function.
3.) To change the positions of your ImageSprites use the MoveTo function.

Congratulations 😊

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

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

**Level of difficulty**
You can add multiple buttons to control the level of difficulty of your game. E.g. the higher the difficulty, the faster the ▲ or the slower the ➤.

**A third asteroid**
After a certain period of time, you can have a third ▲ appear. For this, look up the present time as shown in the value of your labels. As soon as they reach a certain value, you make a third ▲ appear or rather visible. That is, the third ▲ existed from the start, but was invisible.

**An end to the game...**
...would be nice. Therefore create a new label, which displays 100 at the start of the game. Every time the clock is triggered reduce its value by 1.
If the value reaches 0, either reset the game or display a text which tells the player how many points he's got.

Don’t forget to switch off the timer so that the spaceship and asteroids stop 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 functions of the playing field, and find out how to clear them.

**List of references:**
- https://pixabay.com/de/monde-zwei-planetengetriebe-295227/
- https://pixabay.com/de/sterne-raum-universum-431106/
- Source: InfoSphere

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