DIVERSITY
by EPITECH

ARCOBRA

MY FIRST AR FILTER

ARCOBRA

For this activity you will use the **Android Studio** interface with the **Java** language.

If you want to install the application that you are going to make on your own **Android** phone, get a Cobra to help you do it!

Feel free to ask the *Cobra's* present for any other help during the activity.

Introduction

You are an independent developer and you want to release your own *Snapchat/Instagram* filter thanks to Google's new development kit: **ARCore**!
So you will be guided through this activity to use *3D objects* while tracking a face through the camera.

Installation (to be done by a Cobra)

Android Studio

Go to `https://developer.android.com/studio` and download Android Studio then unarchive the download.
Move the `android-studio` folder to the root (`/home/[user_name]`)
Open a terminal in the folder, run: `./bin/studio.sh` and install Android Studio

Google ARCore for Emulator

Go to `https://github.com/google-ar/arcore-android-sdk/releases` and download `Google_Play_Services_for_AR_X.XX.X_x86_for_emulator.apk`.

ARCobra Project

Go to `https://github.com/nicklamyeeman/ARCobra`, unarchive the download and open the project in Android Studio.

Modify the file `Graddle Scripts > local.properties` if Android Studio does not offer it to relocate the Android SDK (`/home/[user_name]/Android/Sdk`)

Install the Android Studio emulator

In ***Android Studio > Tools > Device Manager*** create a new device.

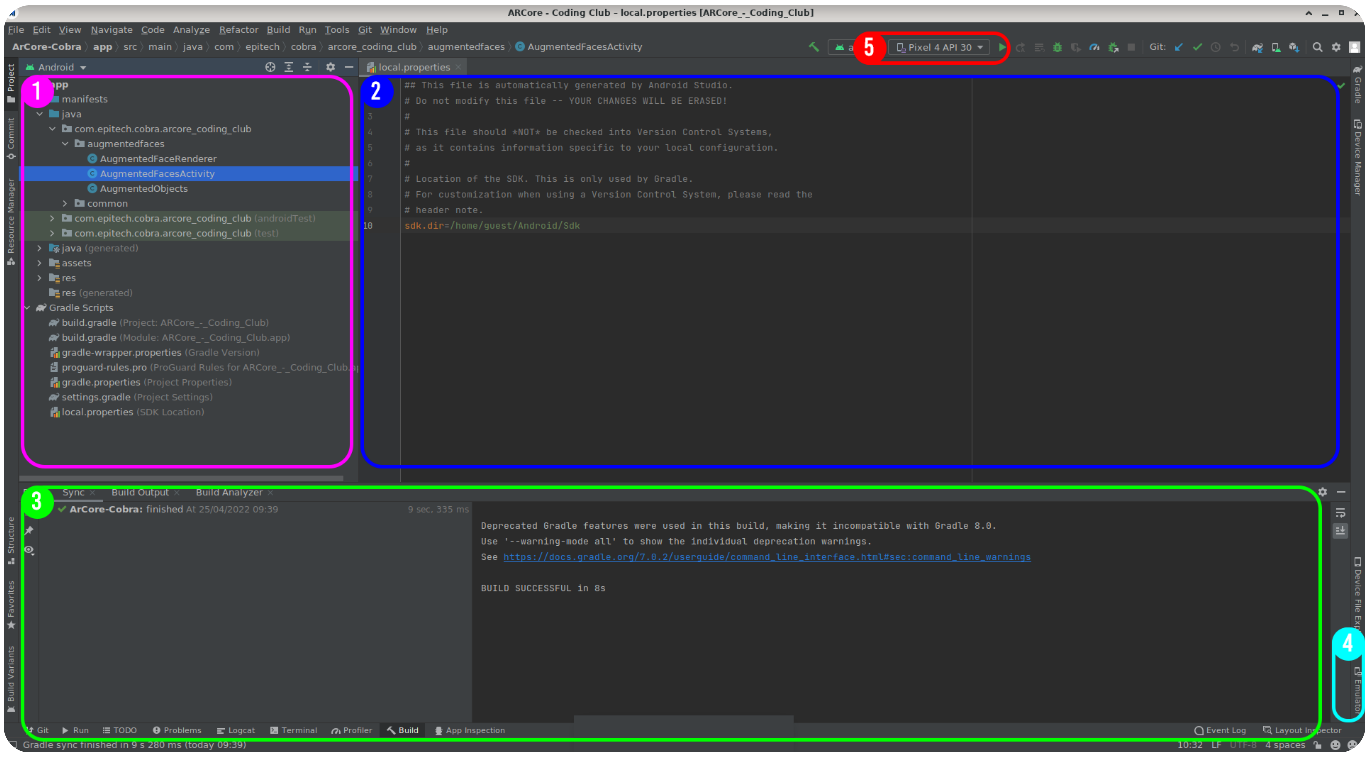
API Level 24 minimum. Default: Pixel 4 API 30
Warning: When selecting a system image, make sure you go to the x86 tab
Display the advanced settings and select `webcam0` for the front camera

Launch the emulation and *drag & drop* the apk on the emulator to install Google Services for AR

Android Studio

The interface

Welcome to **Android Studio**, a working environment for all Android developers.
Here is a short description of the interface:



```
## This file is automatically generated by Android Studio.
# Do not modify this file -- YOUR CHANGES WILL BE ERASED!
#
# This file should *NOT* be checked into Version Control Systems,
# as it contains information specific to your local configuration.
#
# Location of the SDK. This is only used by Gradle.
# For customization when using a Version Control System, please read the
# header note.
sdk.dir=/home/guest/Android/Sdk
```

```
ArCore-Cobra: finished At 25/04/2022 09:39    9 sec, 335 ms

Deprecated Gradle features were used in this build, making it incompatible with Gradle 8.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/7.0.2/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 8s
```

1. This is what we call the ***tree*** of the project, it is where all the files useful for the creation of the Android application are located.
2. Here you will find the ***code editor***, this is where you will write code.
3. Here you will find various useful windows, such as the ***Build*** window, which allows you to follow the compilation process, or the ***Logcat*** window, which allows you to see the developers' messages while the application is active.

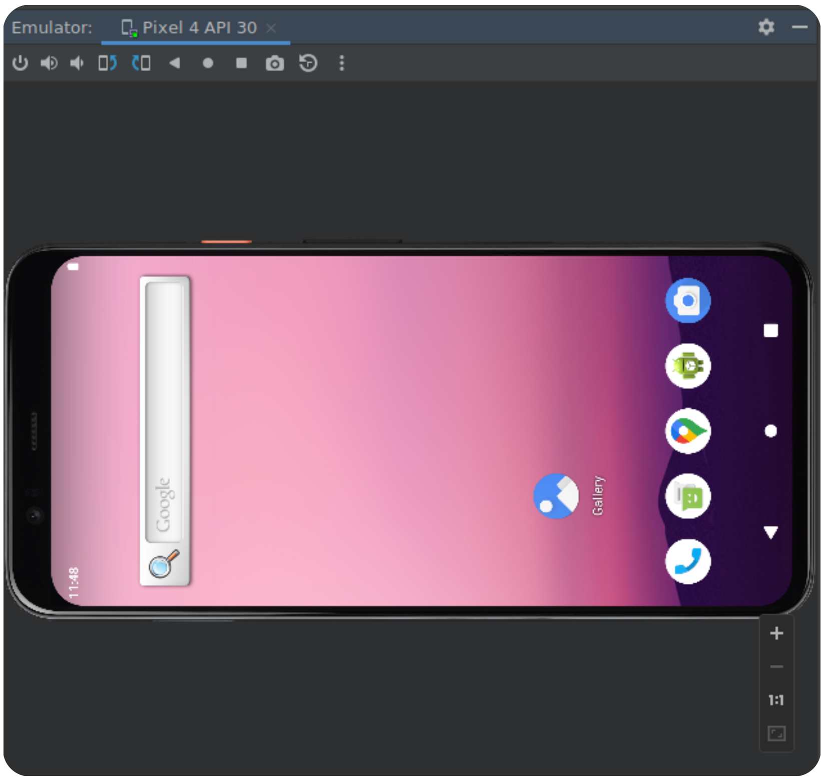
Compilation is the transformation of a human readable code into a computer readable application.

4. This little button opens the ***Android emulator*** window.

5. And this one allows you to ***launch the application*** on the emulator (Pixel 4 API 30) or on your smartphone if you have requested it

The emulator

If you launch the emulator on Android Studio, make sure to orient it in landscape as below.



Step 1: Create a place to store augmented reality objects

For all the steps, you will have to modify only one file: **AugmentedFacesActivity**.

All steps are marked in this file, just find the comment blocks corresponding to the step

Comments and comment blocks are lines that are not read by the program.
Comment blocks start with :

```
/**
 *   STEP 1 : [...]
 */
```

and end with :

```
/**
 *
 */
```

This ***first step*** serves as a very quick introduction to Java.

The file you are modifying is called a ***Class***.

```
public class AugmentedFacesActivity [...]
```

This ***class*** allows to display the content of the application. To do this it needs resources that can be stored in *variables* called ***attributes***.

```
    [...]
    private GLSurfaceView surfaceView;
    private boolean installRequested;
    private Session session;
    private final SnackbarHelper messageSnackbarHelper = new SnackbarHelper();
    [...]
```

So we will create our own attribute to put all the AR objects in it.
Let's base it on the ***SnackbarHelper*** attribute:

```
/**     options        class          attribute name          initialization    **/
    private final SnackbarHelper messageSnackbarHelper = new SnackbarHelper();
```

Taking the example of the initialization of the attribute above, we will create our attribute with the following information:

- ✓ options : `private final`
- ✓ class : `AugmentedObjects`

```
private final AugmentedObjects augmentedObjects = new AugmentedObjects();
```

Step 2: Create the objects in augmented reality

Now that we have created a space to store our objects, we will ***create*** them.

In step 1 you gave a **name** to your attribute, this is very important because it is this name that we will use throughout the program.

```
                                  /** attribute **/
private final AugmentedObjects augmentedObjects = new AugmentedObjects();
                                    /** name **/
```

With this name you can access the ***methods*** of the class `AugmentedObjects`

A **method** is an instruction that performs a specific action.
For example, in video games, the `Hero` classes usually have methods for moving or fighting.

Here the ***methods*** will be written as follows:

```
attribute_name.method_name(option1, option2, ...);
```

Here is the information needed to create your objects:

- ✓ attribute name: `augmentedObjects` *(depending on how you called it at the very beginning)*
- ✓ method name : `addObject()`
- ✓ options :
  - `context` : *the context of the application. Here it will always be* `this`
  - `name` : *the name of the object you want. Example :* `"nez"`
  - `objName` : *the path of the object. Example :* `"models/objects/nose.obj"`
  - `materialName` : *the path of the material. Example :* `"models/materials/nose_fur.png"`
  - `area` : *where to place the object on the face*

You will manipulate objects and textures already designed for the activity.
In the interface tree, you will find the objects in the folder
- ***assets > models > objects***
and textures in the folder
- ***assets > models > materials***

There are only 3 possible places on the face:

```
/** FOREHEAD LEFT **/
AugmentedObjects.FACE_AREA.FOREHEAD_LEFT

/** FOREHEAD RIGHT **/
AugmentedObjects.FACE_AREA.FOREHEAD_RIGHT

/** CENTER **/
AugmentedObjects.FACE_AREA.CENTER
```

```
/** Example creation of an object on the nose **/
    augmentedObjects.addObject(
        this,
        "nose",
        "models/objects/nose.obj",
        "models/materials/nose_fur.png",
        AugmentedObjects.FACE_AREA.CENTER
    );
```

BONUS : Change the face mask

This step is optional

You may have noticed the “BONUS” comment block

```
/**
 *    BONUS : Change the face mask
 */
```

The line of code just below allows to put a mask on the face.
By default, the texture is transparent: `models/meshes/empty.png`

It’s up to you to try to modify this texture!

Step 3: Display objects in real time

If you run the application now in your emulator or on your Android device, you will not have **any** objects displayed.

If you have done the bonus step, you should see the change

Before displaying each of the *1 by 1* objects there are **two** lines to add at the beginning and at the end:

```
augmentedObjects.updateObjectsMatrix(face);

// [...] The display of your objects

augmentedObjects.drawObjects(viewMatrix, projectionMatrix, colorCorrectionRgba,
    DEFAULT_COLOR);
```

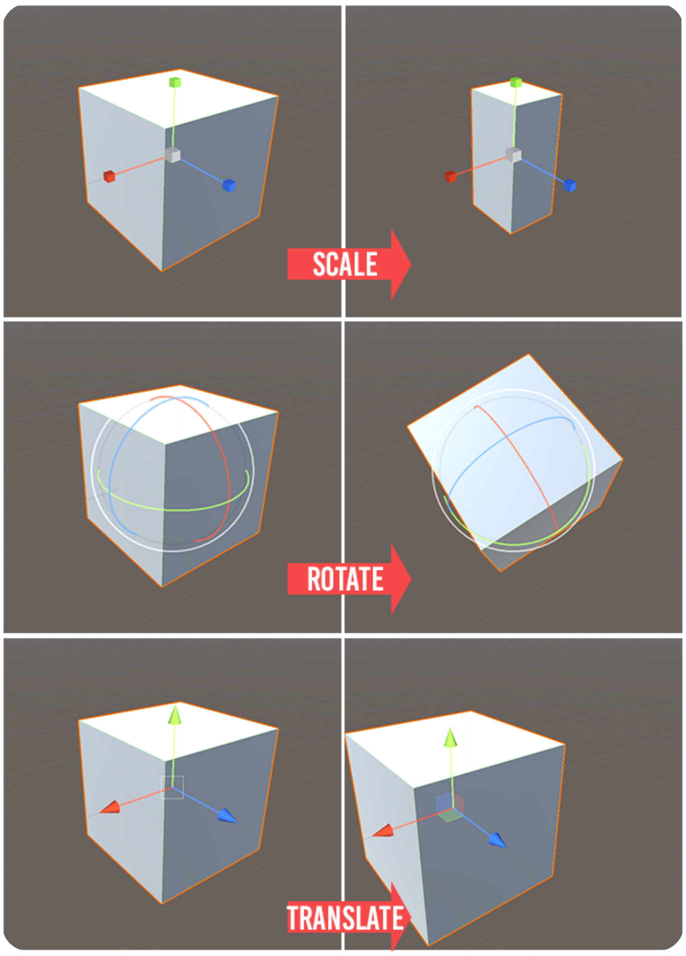
These two lines are already written but are in comment. Just remove the «**//**» at the beginning of the line so that the program can read the line of code.

Transformations of 3D objects

Before getting to the heart of the matter, let's look at a few concepts.

Each 3D object is defined in space by what is called a **matrix**.
This **matrix** contains all the necessary information about an object (size, position, angle, etc...)

The **matrices** can thus be **transformed** according to ***3 main categories*** :
- **Scale** : transformation of the size of the object
- **Rotation** : transformation of the orientation of the object in space
- **Translation** : transformation of the position of the object in space



These transformations are defined in a 3D space, therefore with values on the 3 axes of space: `x`, `y`, `z`

To display our objects it is **obligatory** to transform them:

✓ attribute name: `augmentedObjects` *(depending on how you called it at the very beginning)*

✓ method name : `updateObject()`

✓ options :

- `name` : *the name of the object you gave previously. Example :* `"nose"`
- `scaleFactor` : *the scale values of the object. Example :* `new float[]{ 0.5f, 2.0f, -2.0f }`
- `rotateFactor` : *the rotation values of the object. Example :* `new float[]{ 45.0f, 1.0f, 0.0f, 0.0f }`
- `translateFactor` : *the translation values of the project. Example :* `new float[]{ 0.5f, 0.0f, 0.8f }`

To ***scale*** an object you have to give the values in **x**, **y** and **z**.
These 3 values are used as *multipliers* on the respective axes.
For example, if you want to double the size of the object only on the x axis:
`new float[]{ 2.0f, 1.0f, 1.0f }`
Or if you want to divide the object by 10 (because it is 10 times too big):
`new float[]{ 0.1f, 0.1f, 0.1f }`

A negative value will reverse the object, it will be like looking in a mirror

To ***rotate*** an object, you have to give **the angle** of rotation and the values in **x**, **y** and **z**
For example, if you want to rotate the project by 45° on the Y axis:
`new float[]{ 45.0f, 0.0f, 1.0f, 0.0f }`
Or if you want to turn the object 90° on two axes:
`new float[]{ 90.0f, 1.0f, 0.0f, 1.0f }`

The values in x, y and z are always ***0.0f*** or ***1.0f***

To ***translate*** an object, you must give the values in **x**, **y** and **z**
For example, if you want to move the object up by 10 units:
`new float[]{ 0.0f, -10.0f, 0.0f }`
Or if you want to move the object to the bottom right while moving it back:
`new float[]{ -10.0f, 5.0f, -4.0f }`

The values are displacement, not position in space. If you move an object with `new float[]{ 0.0f, 0.0f, 0.0f }` it will not be at the center of the screen but will move by 0 on the X axis, 0 on the Y axis and 0 on the Z axis. Basically *it won't move*.

If you don't want to change the object, just set the default values.

The default options are :
- ***scaleFactor*** : `new float[]{1.0f, 1.0f, 1.0f}`
- ***rotateFactor*** : `new float[]{0.0f, 1.0f, 0.0f, 0.0f}`
- ***translateFactor*** : `new float[]{0.0f, 0.0f, 0.0f}`

```
augmentedObjects.updateObject(
    "nez",
    new float[]{1.0f, 1.0f, 1.0f},
    new float[]{0.0f, 1.0f, 0.0f, 0.0f},
    new float[]{0.0f, 0.0f, 0.0f}
);
```

If you want to use the default values you can also use `null`

```
augmentedObjects.updateObject(
    "nez",
    null,
    null,
    null
);
```

Step 4: It's your turn!

Now that you have all the cards in hand, your mission is to display :

- ✓ A fox left ear (`forehead_left.obj` + `ear_fur.png`)
- ✓ A right ear (`forehead_right.obj` + `ear_fur.png`)
- ✓ A nose (`nose.obj` + `nose_fur.png`)

With default values.

To go further

If you want to play with the remaining objects, try doing :
- CODING : *forehead left*
- CLUB : *forehead right*
- EPITECH : *center*

To put 3D texts in the right direction, manipulate the scaling, rotate and translate values multiple times.
To flip a 3D text for example (as in a mirror), just scale negatively on the y axis.

Feel free to try everything and have fun with the objects and textures that exist

If during your various experiments your 3D objects are not displayed **more**, try to re***Build*** the project by pressing the small green hammer next to the name of the device on which you are testing (the emulator or your phone)

There are several ***methods*** (in addition to the method `updateObject(...)`) that you can use to add transformations on your objects.

```
augmentedObjects.scaleObject(name, scaleFactor);
augmentedObjects.rotateObject(name, rotateFactor);
augmentedObjects.translateObject(name, translateFactor);
augmentedObjects.scaleRotateObject(name, scaleFactor, rotateFactor);
augmentedObjects.scaleTranslateObject(name, scaleFactor, translateFactor);
augmentedObjects.rotateTranslateObject(name, rotateFactor, translateFactor);
```

v 0.1.0

COBRA

CODING CLUB

DIVERSITY

by EPITECH