Face Mesh Filter Camera and detection system in AI technology

Instructor: Dr. Ata Jahangir Moshayedi

Name: Nafiz Md Imtiaz Uddin
Id no: 2520190011
Course: Development of Mobile Application

What is Camera Filtering in the app inventor in the AI?

AI cameras are simply cameras that use AI programs to wisely deal with images and videos. Computational photography is usually the core of an AI(artificial intelligence).

AI systems have been able to accomplish useful tasks like classifying images or understanding spoken language. AI computations can be performed on the smartphones and tablets available to students.

MIT App Inventor | App Inventor Resources | Artificial Intelligence with MIT A

https://appinventor.mit.edu/explore/ai-with-mit-app-inventor

MIT APP INVENTOR

Create Apps! | About | Educators | News | Resources | Blogs | Give | ENHANCED BY Goo

Artificial Intelligence with MIT App Inventor

AI WITH APP INVENTOR

Artificial Intelligence (AI) has been part of computing since the 1950s. But it's only been since 2000 that AI systems have been able to accomplish useful tasks like classifying images or understanding spoken language. And only very recently has Machine Learning advanced to a point such that significant AI computations can be performed on the smartphones and tablets available to students.

MIT is building tools into App Inventor that will enable even beginning students to create original AI applications that would have been advanced research a decade ago. This creates new opportunities for students to explore the possibilities of AI and empowers students as creators of the digital future.

AI with MIT App Inventor includes tutorial lessons as well as suggestions for student explorations and project work. Each unit also includes supplementary teaching materials: lesson plans, slides, unit outlines, assessments and alignment to the Computer Science Teachers of America (CSTA) K12 Computing Standards.

As with all MIT App Inventor efforts, the emphasis is on active constructionist learning where students create projects and programs that instantiate their ideas.

https://appinventor.mit.edu/explore/ai-with-mit-app-inventor

MIT App Inventor | Facemesh Filter Camera

appinventor.mit.edu/explore/resources/ai/facemesh

MIT APP INVENTOR | Create Apps! | About | Educators | News | Resources | Blogs | Give | ENHANCED BY Goo

Facemesh Filter Camera

Facemesh Filter Camera

Difficulty: intermediate
Lesson Type: tutorial
Subject: computer science

Grade Level:
- 6-8
- 9-12

Resource URL: Facemesh Camera Filter

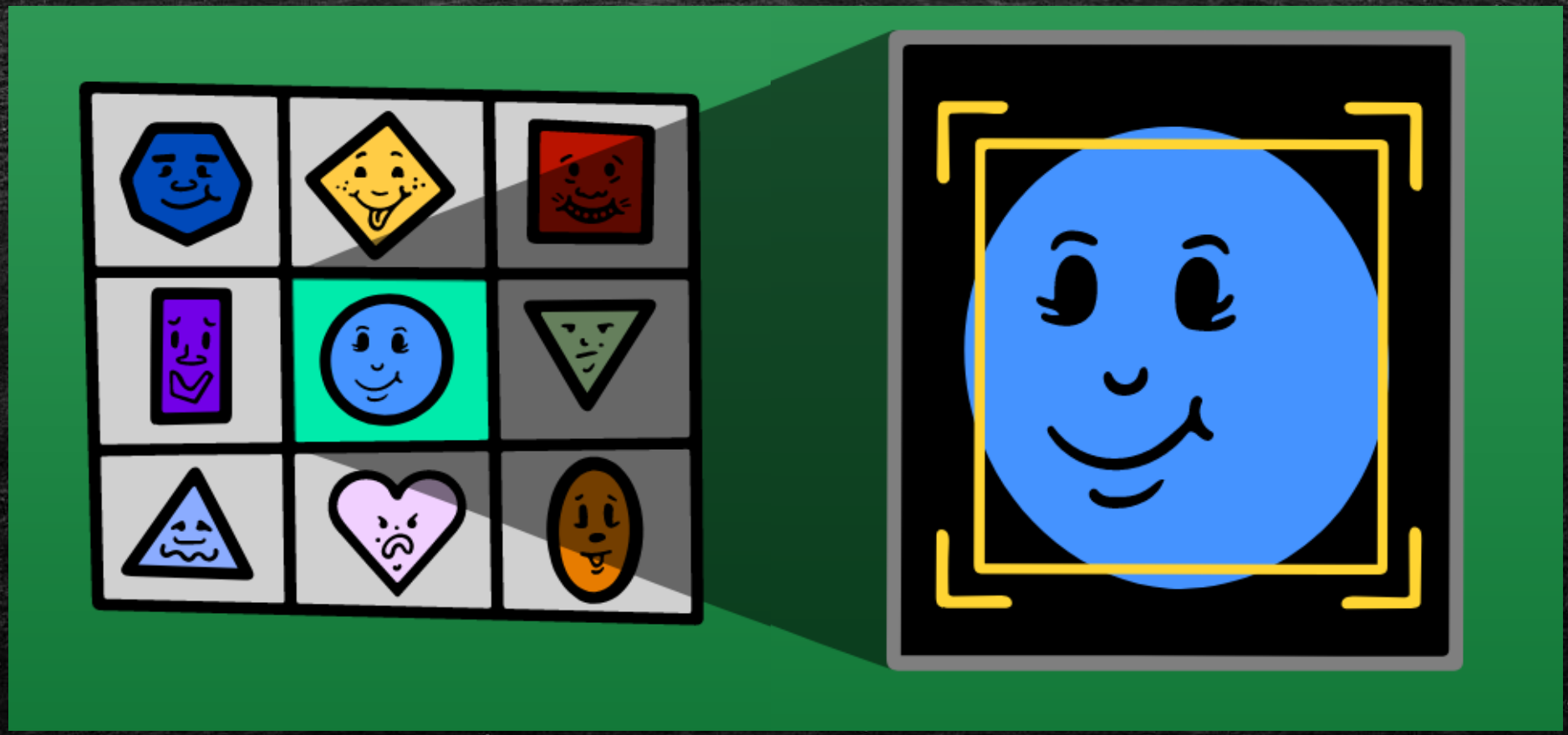
Have you taken photos with facial filters? Instagram and Snapchat facial filters have taken the internet by storm, but do you know how these filters work? Would you like to make your own facial filters? Our friends at YR Media have this excellent interactive article about facial recognition. In this tutorial, we will be using a similar but different technology - facial landmark detection. You are challenged to create a filter camera using a new AI technology called Facemesh.

https://appinventor.mit.edu/explore/resources/ai/facemesh

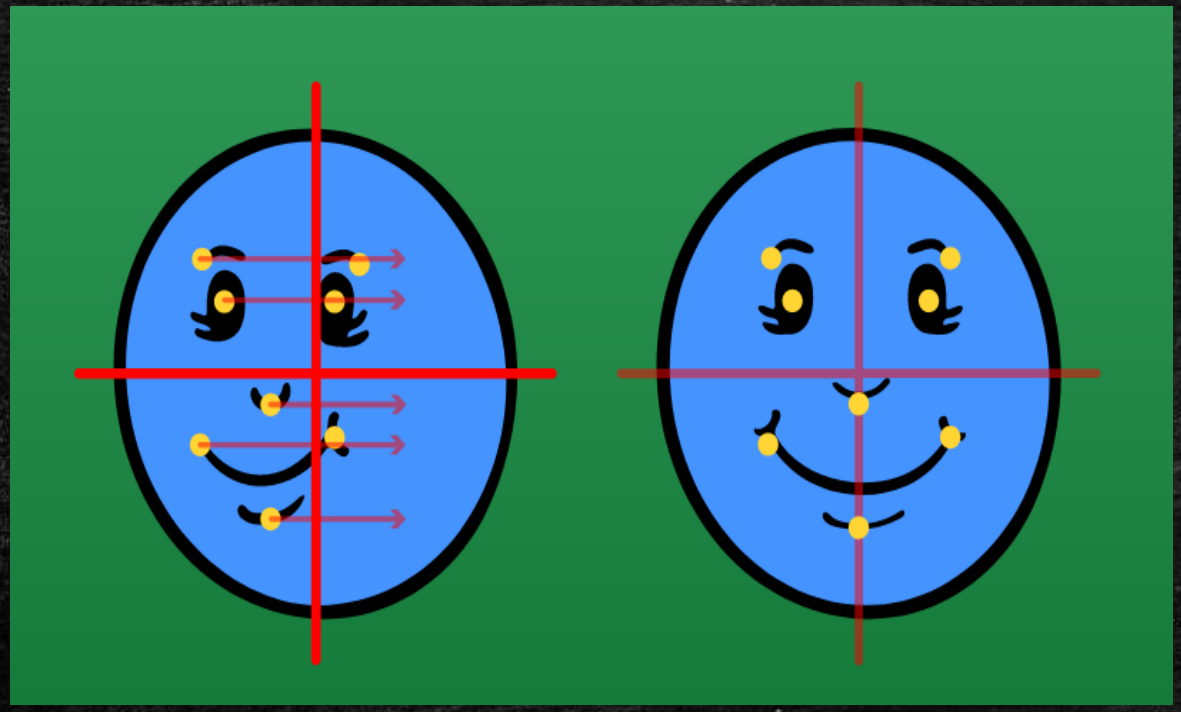
Prerequisite of learning AI technology

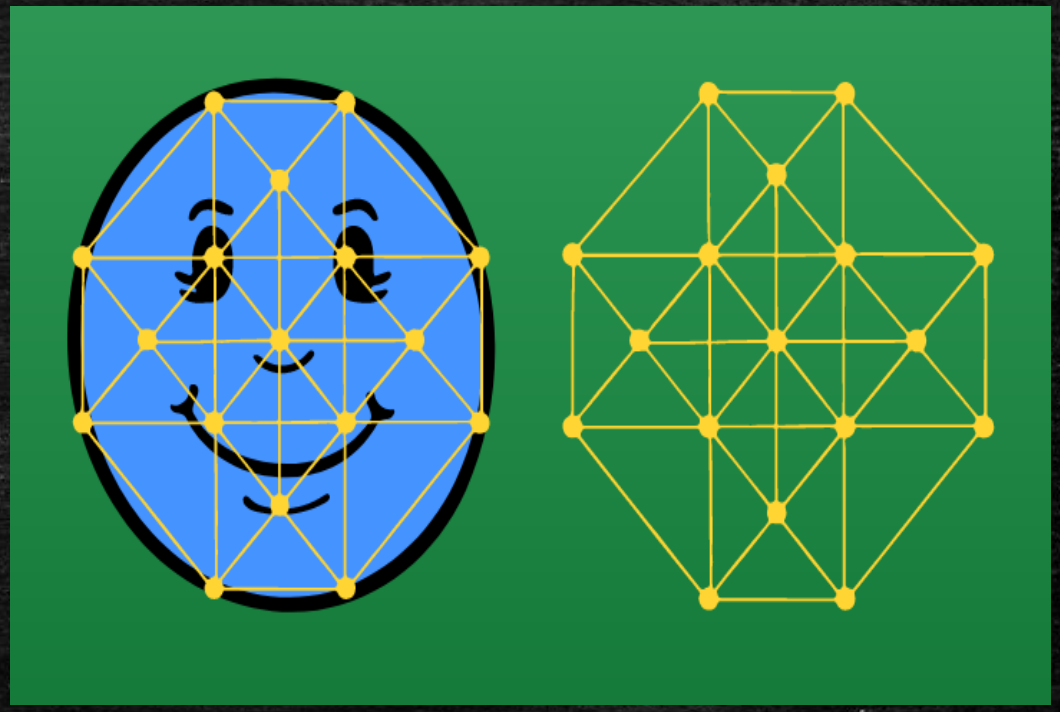
- Know good math and physics
- Good programming language knowledge
- Technique of logic and algorithm
- Graphical modeling
- Cognitive science theory
- Automation, Robotic system
- Good Analytical Skills

How Facial Recognition Works

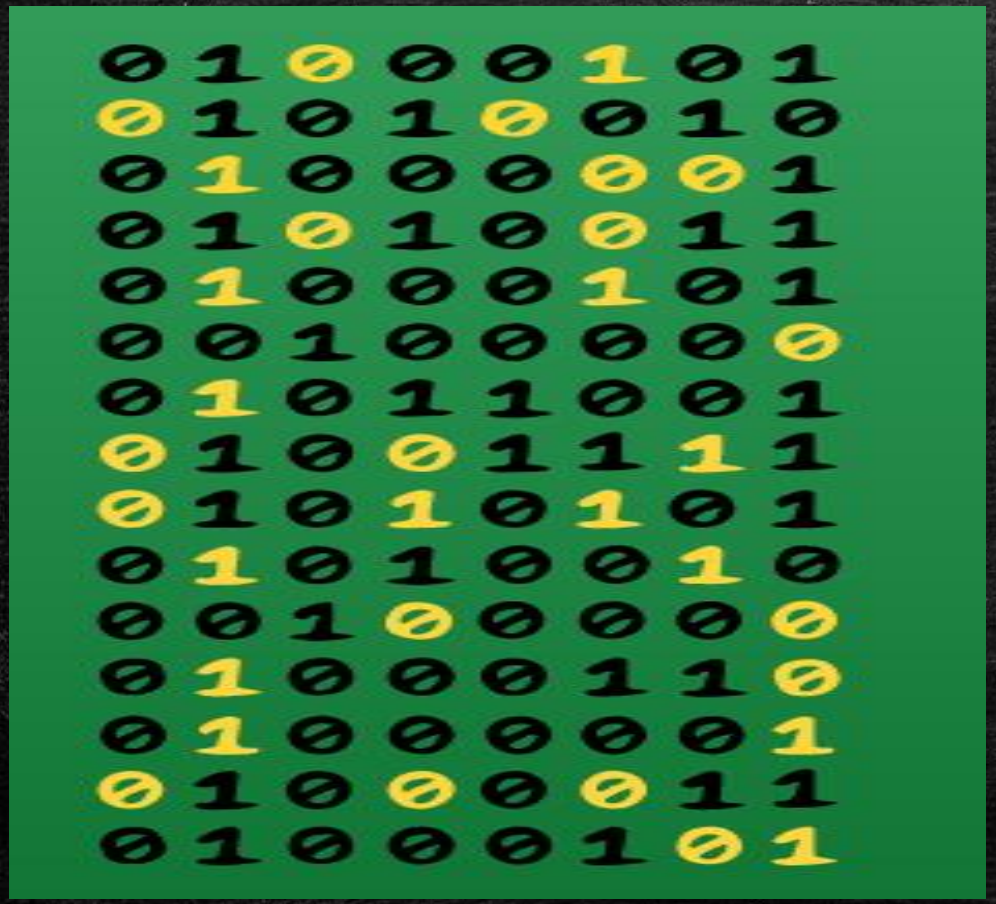


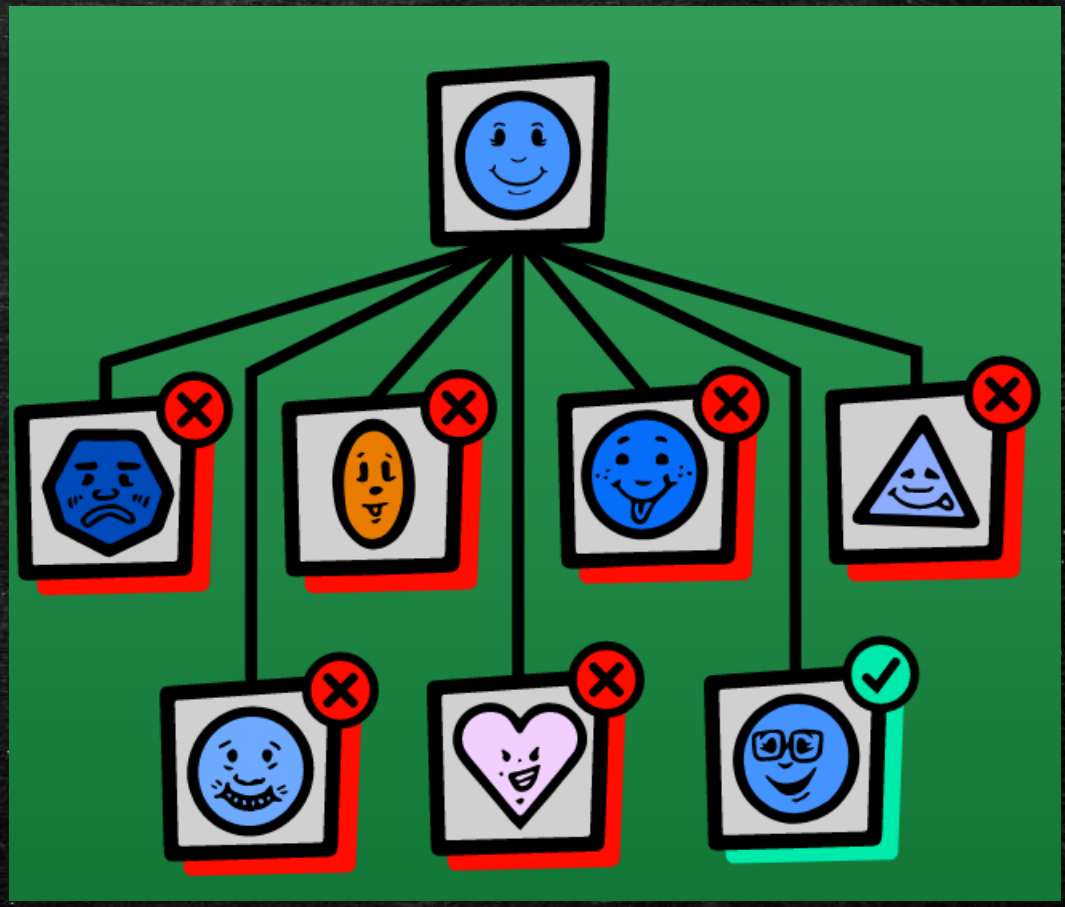
How Facial Recognition Works



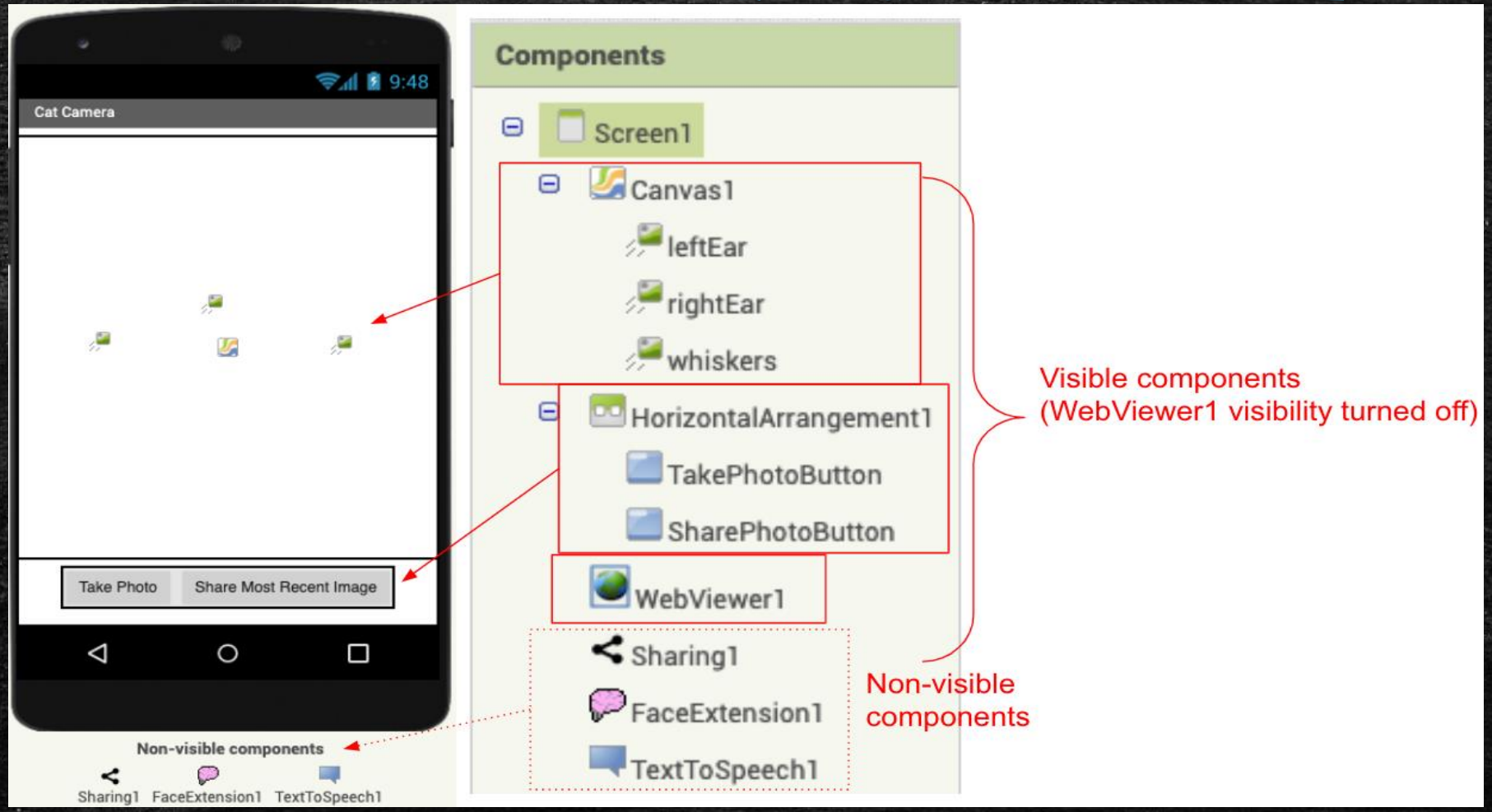


How Facial Recognition Works

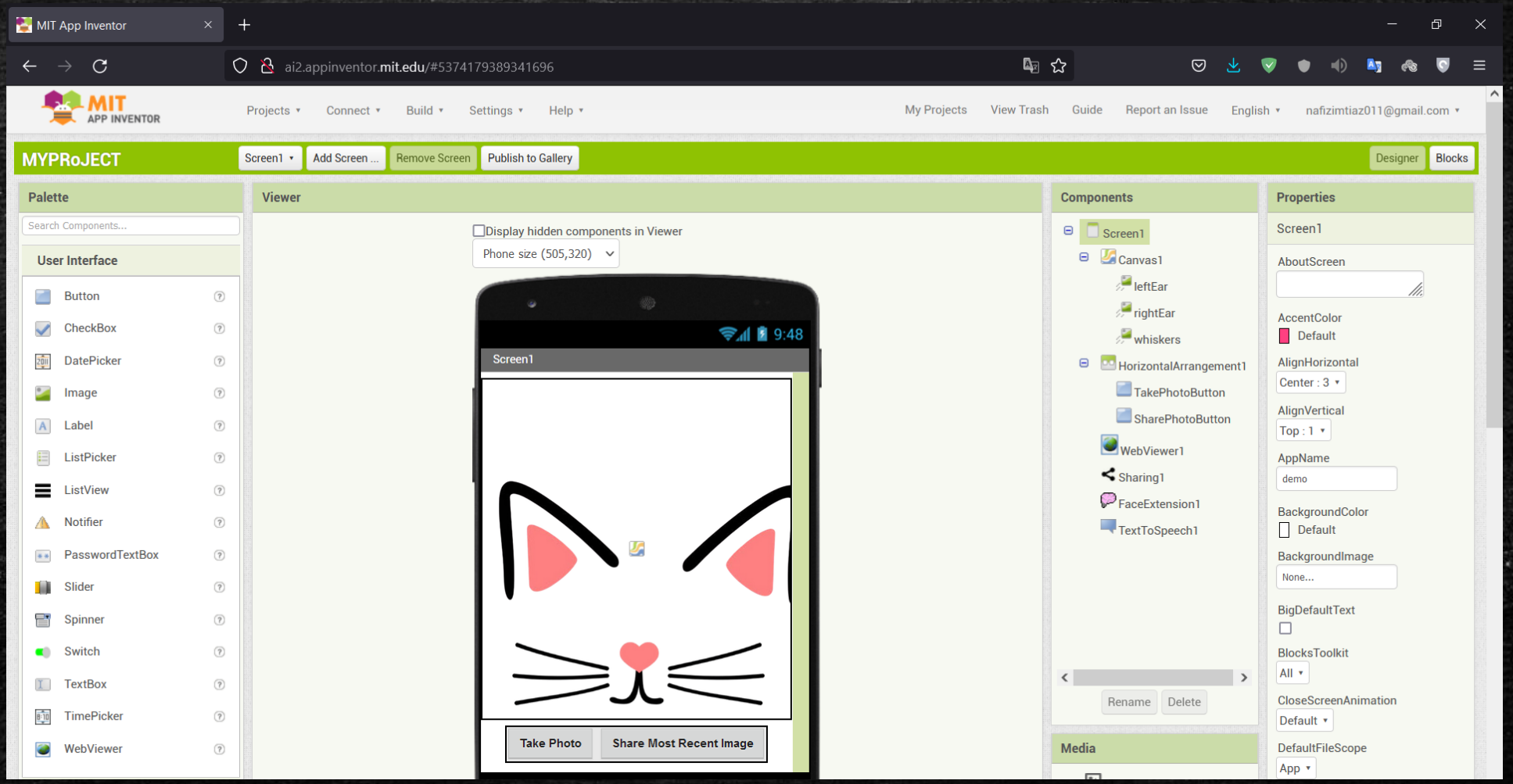




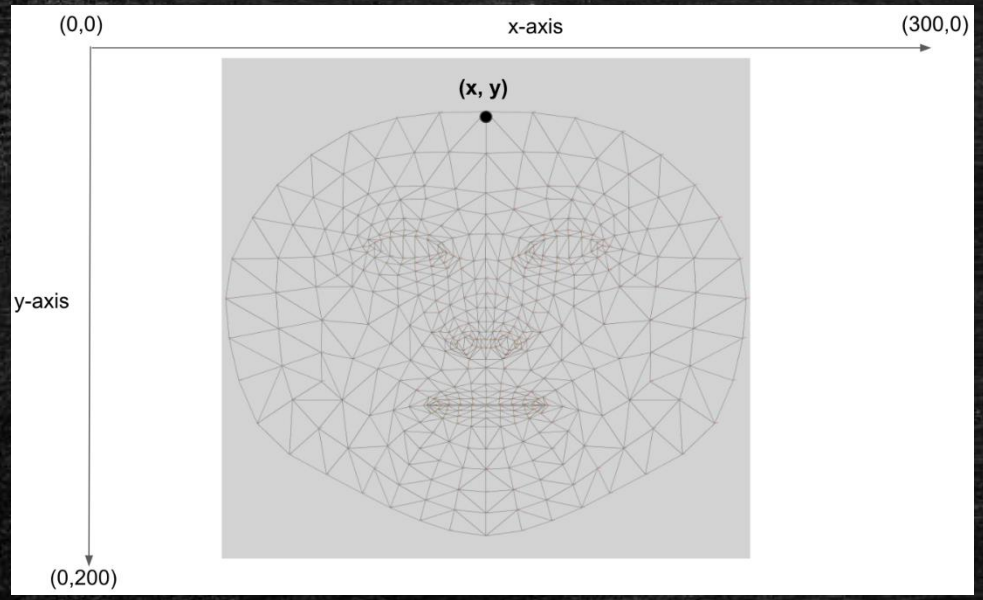
Designer Section of Face Mesh Camera



Screenshot of Facemesh filtering Designer section



Face Point Format



- Each key point is returned as a list of two elements representing the x and y-coordinates. For example, the key point "forehead" will be a list of 2 elements:
- **[forehead x-coord, forehead y-coord]**
- When Facemesh is unable to track the entirety of a face, it will return an empty list so the filter will not work; make sure the face is within the camera frame!

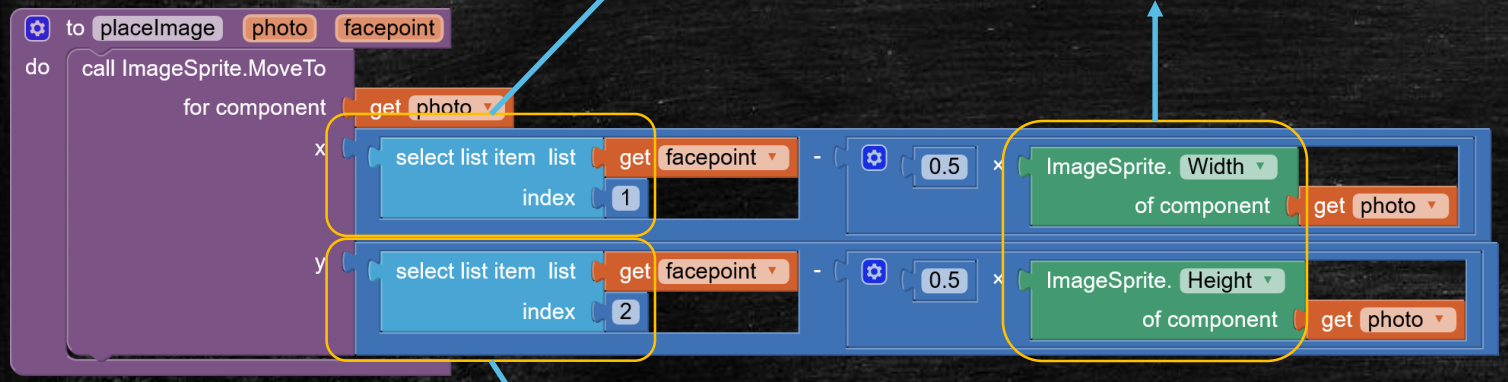
Take 2 variables and TakePhotoButton

- Here we have two variables: *countingphoto* simply counts how many photos have been taken, and *recentphoto* stores the file name of the most recently taken photo.
- When we click the **TakePhotoButton**, the *recentphoto* file name is updated to be the current image, and the *countingphoto* increases by 1. The first photo we take will be called 'IMG1.jpg' and the second photo will be called 'IMG2.jpg', etc. These files are all saved to our device. Depending on our device, we can find the photos wherever files are saved. For example, on a Google Pixel, the photos are saved to the "Files" app.
- The TextToSpeech means that a robot will say "Okay done" out loud when the photo is taken.
- When we click the **SharePhotoButton**, we can share the most recent photo! We'll be able to share it using any app installed on our device that shares images, such as Google Drive, Dropbox, etc.

Using Function: To take procedure

To get x_f, use the following block that gets the item at index 1 from the facePoint list.

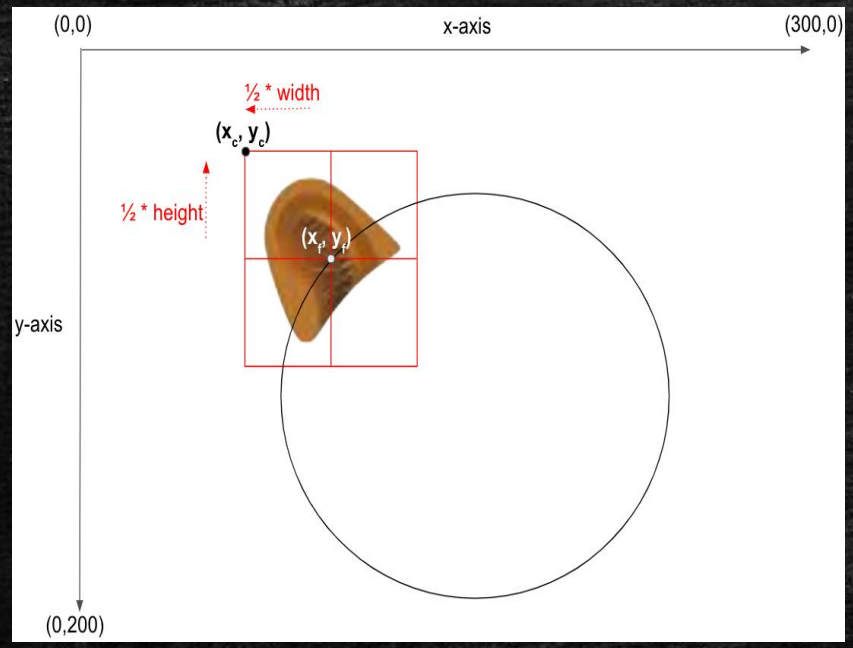
Width and Height section



To get y_f, use the following block that gets the item at index 2 from the facePoint list.

- Now we need to write some code to place the three ImageSprites (leftEar, rightEar, and whiskers) on the corresponding points on the face.
- placeImage procedure places the center of img, the **ImageSprite**, on the facePoint, the key point tracked by Facemesh.

Placing the ears and whiskers



- By calling ImageSprite.MoveTo (x, y), we move the ImageSprite's **top-left corner** to (x, y).
- However, if we want the **center** of img to be placed on the facePoint (x_f, y_f), we have to do some simple math to get the coordinates of its top-left corner, which we will call (x_c, y_c).
- Note that in this x-y coordinate system of the **Canvas**, (0, 0) is the top left corner. The difference from the normal Cartesian coordinate system is that y increases downwards.
- In the scenario above, we want to place the left ear centered on the left forehead facePoint (x_f, y_f) returned by Facemesh.
- To get the adjusted horizontal placement
  - **$x_c = x_f$ - 0.5 * image width**
- To get the adjusted vertical placement
  - **$y_c = y_f$ - 0.5 * image height**
- When we call ImageSprite.MoveTo (x_c, y_c), the image is centered on (x_f, y_f), which are the coordinates of the facePoint.

Face Updated

- When the **FaceExtension** detects a face, it triggers the following FaceExtension1.FaceUpdated event. This event handler's code has also been created for us.

Moving

- The way a basic filter works has two parts.

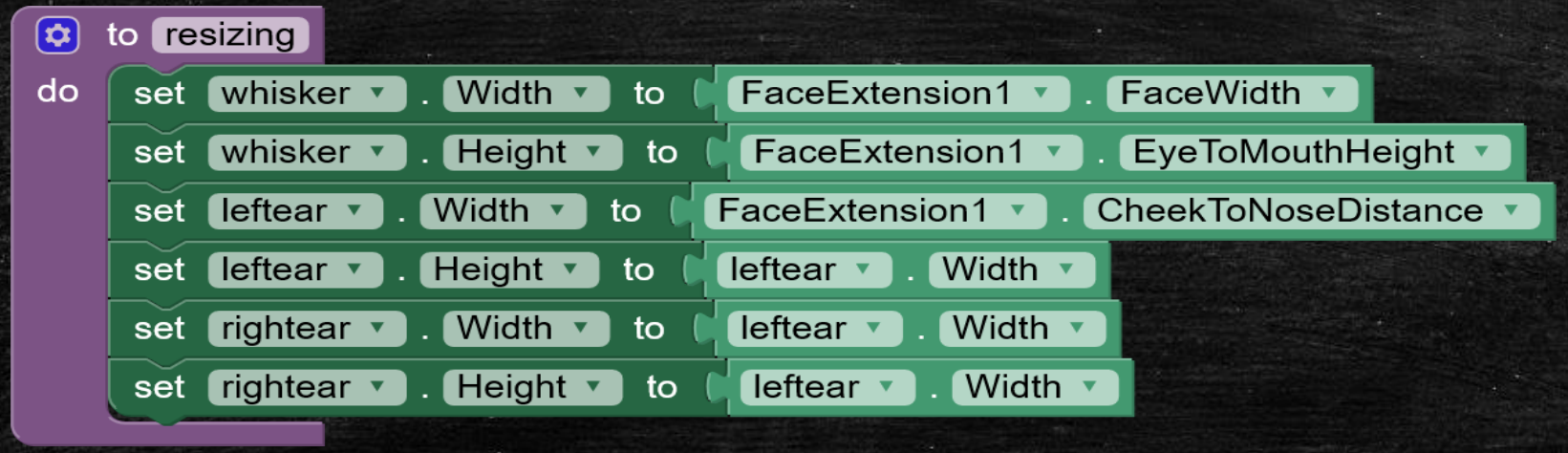
1. moving: when the face moves, the images move along with the face.
2. resizing: when the face becomes bigger or smaller, the images are resized accordingly.

- Call placeImage to move **leftEar** to the left-side of the forehead (FaceExtension.LeftForehead).
- Call placeImage to move **rightEar** to the right-side of the forehead (FaceExtension.RightForehead).
- Call placeImage to move **whiskers** to the top of the mouth (FaceExtension.MouthTop).
- Remember, placeImage takes in two arguments:

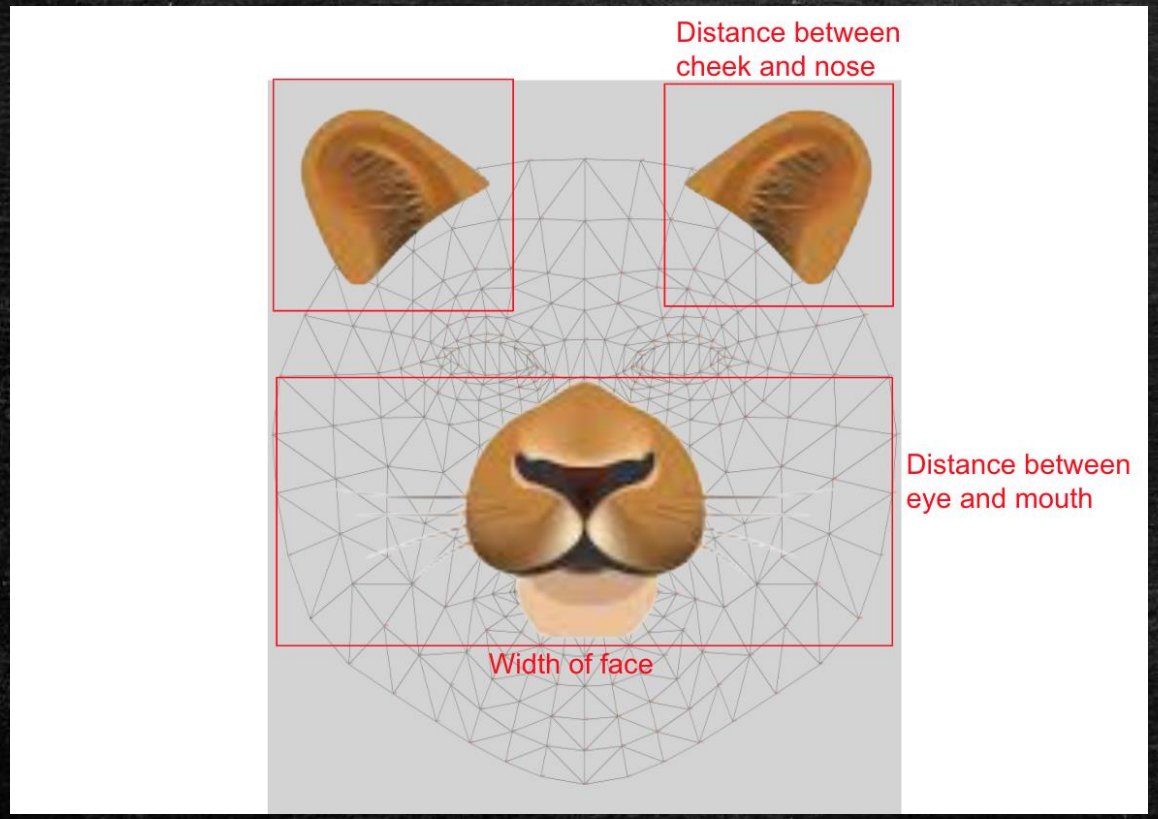
1. the **ImageSprite** object
2. a face key point detected by **FaceExtension**.

Resizing

- When we define the width or height of the image, we're defining the width or height of the image *bounding box* (highlighted in red).
- All widths and heights are arbitrarily chosen to fit what we might expect; we can change them as we like.

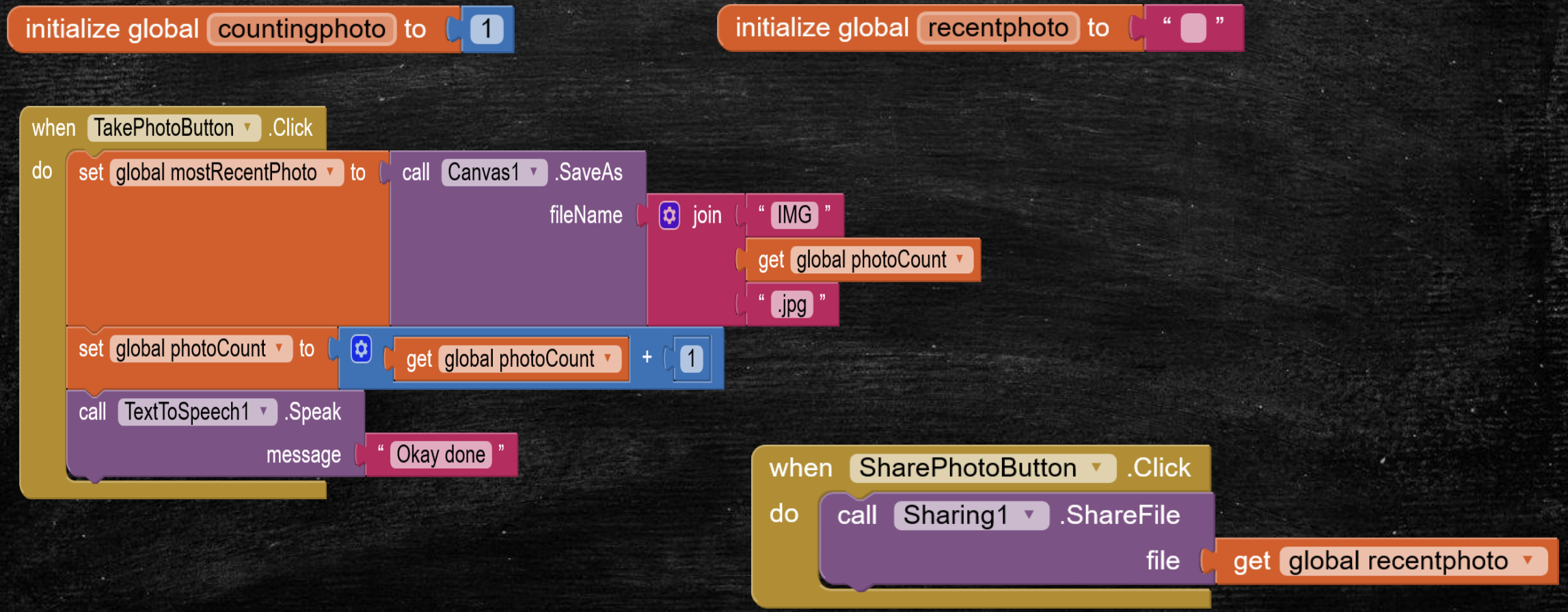


Whiskers and Ears setting

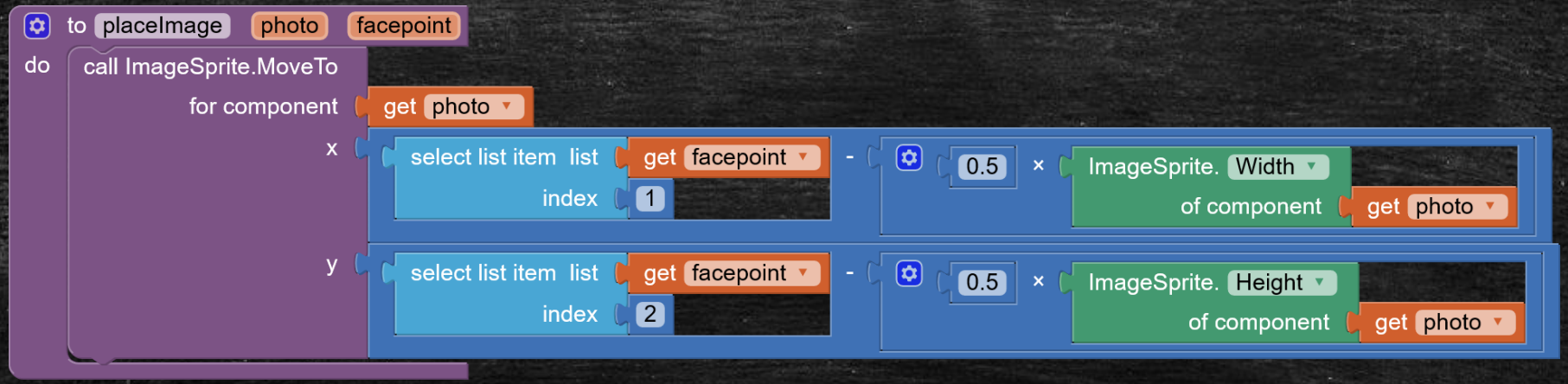


- **<u>Whiskers:</u>**
- Width is set to the width of the face.
- Height is the distance between the eye and the mouth.
- **<u>Ears:</u>**
- Width is set to the distance between the cheek and the nose.
- These have square-shaped bounding boxes, so we can set the width and height to be the same

Block design



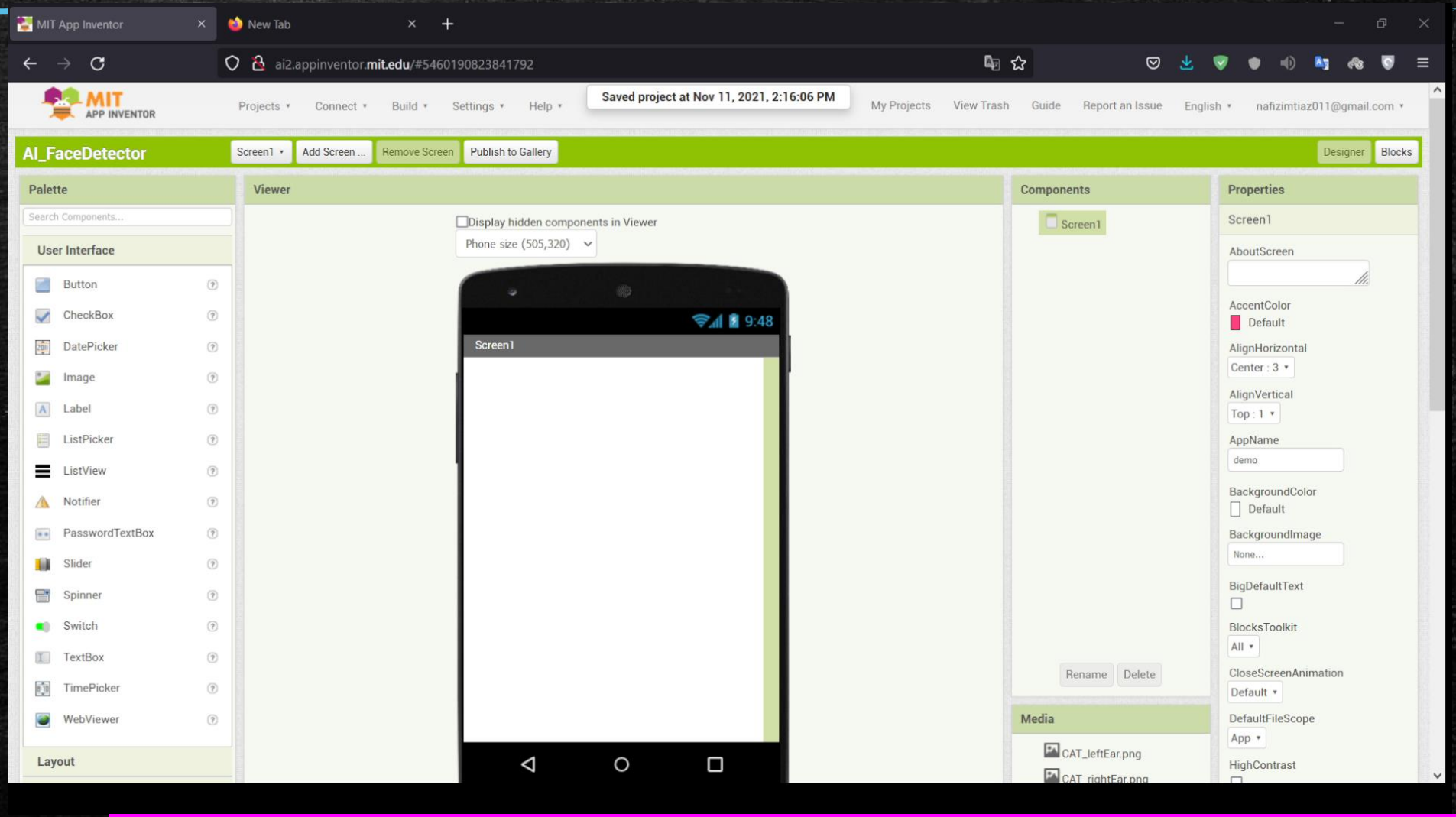
Block design



Block design

Facemesh Filter Camera design and block

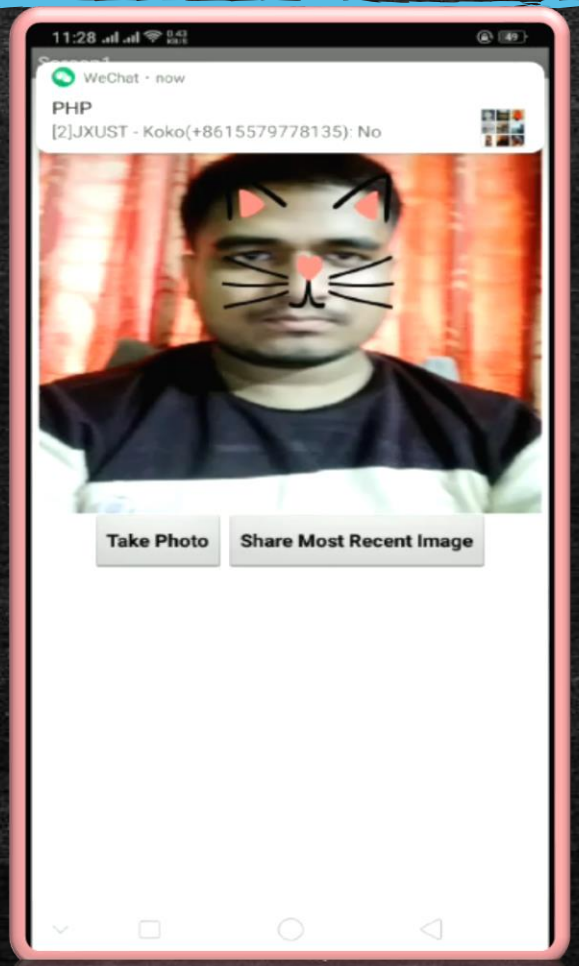
Clip



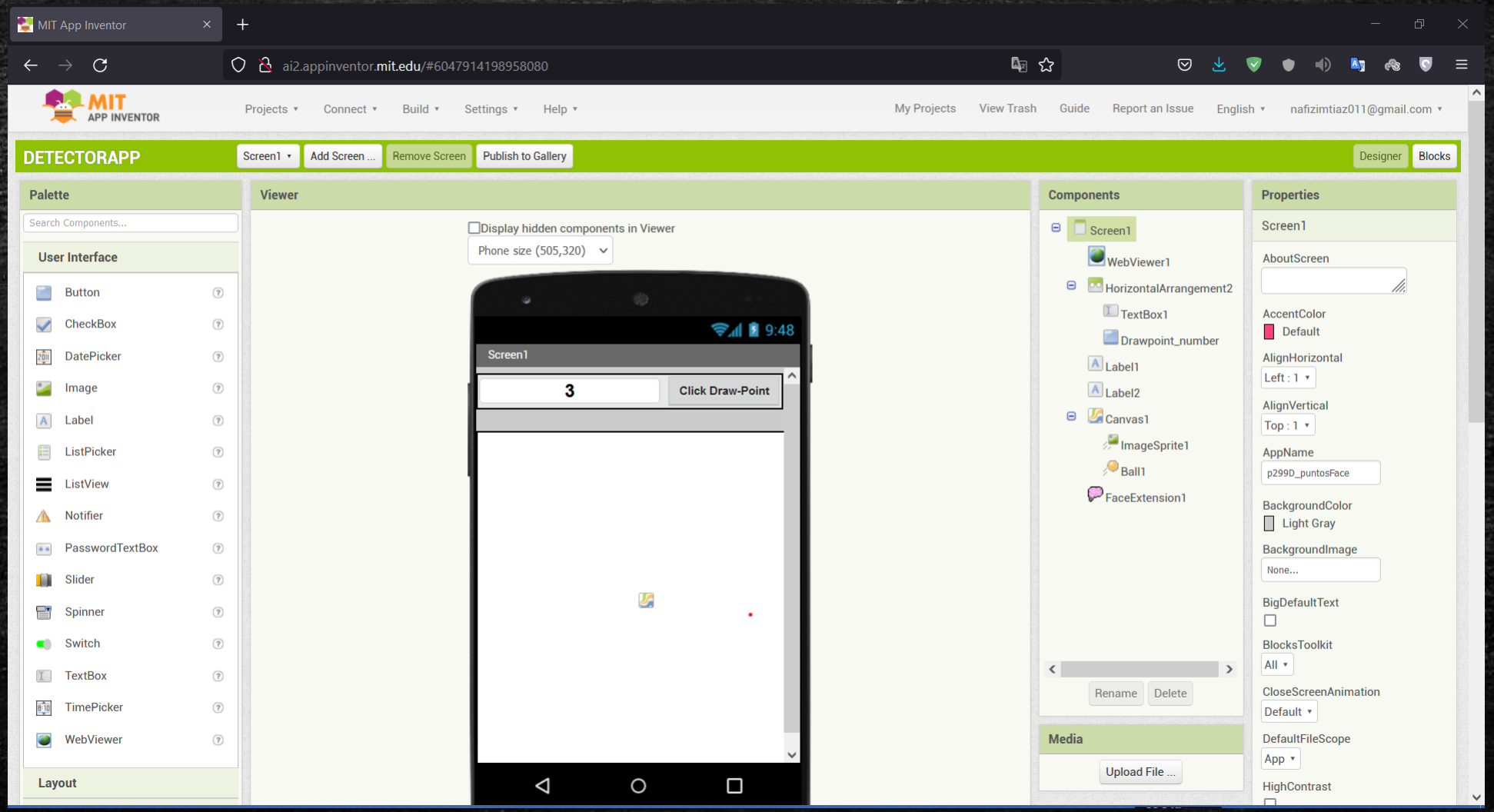
https://mit-cml.github.io/extensions/data/extensions/edu.mit.appinventor.ai.facemesh.aix

Testing Mobile

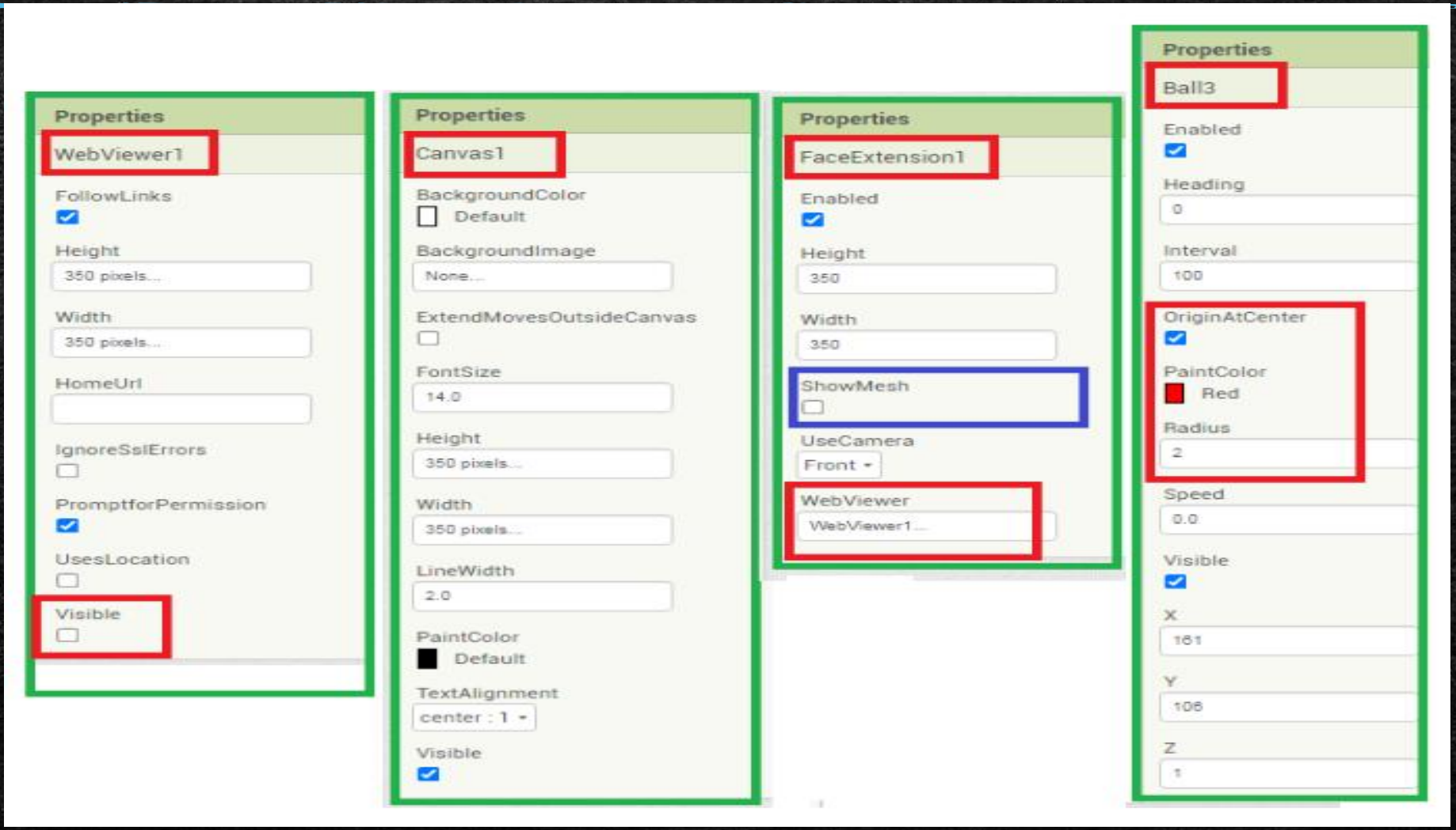
Check that our app can track a face and have the ears and whiskers correctly positioned on the face.



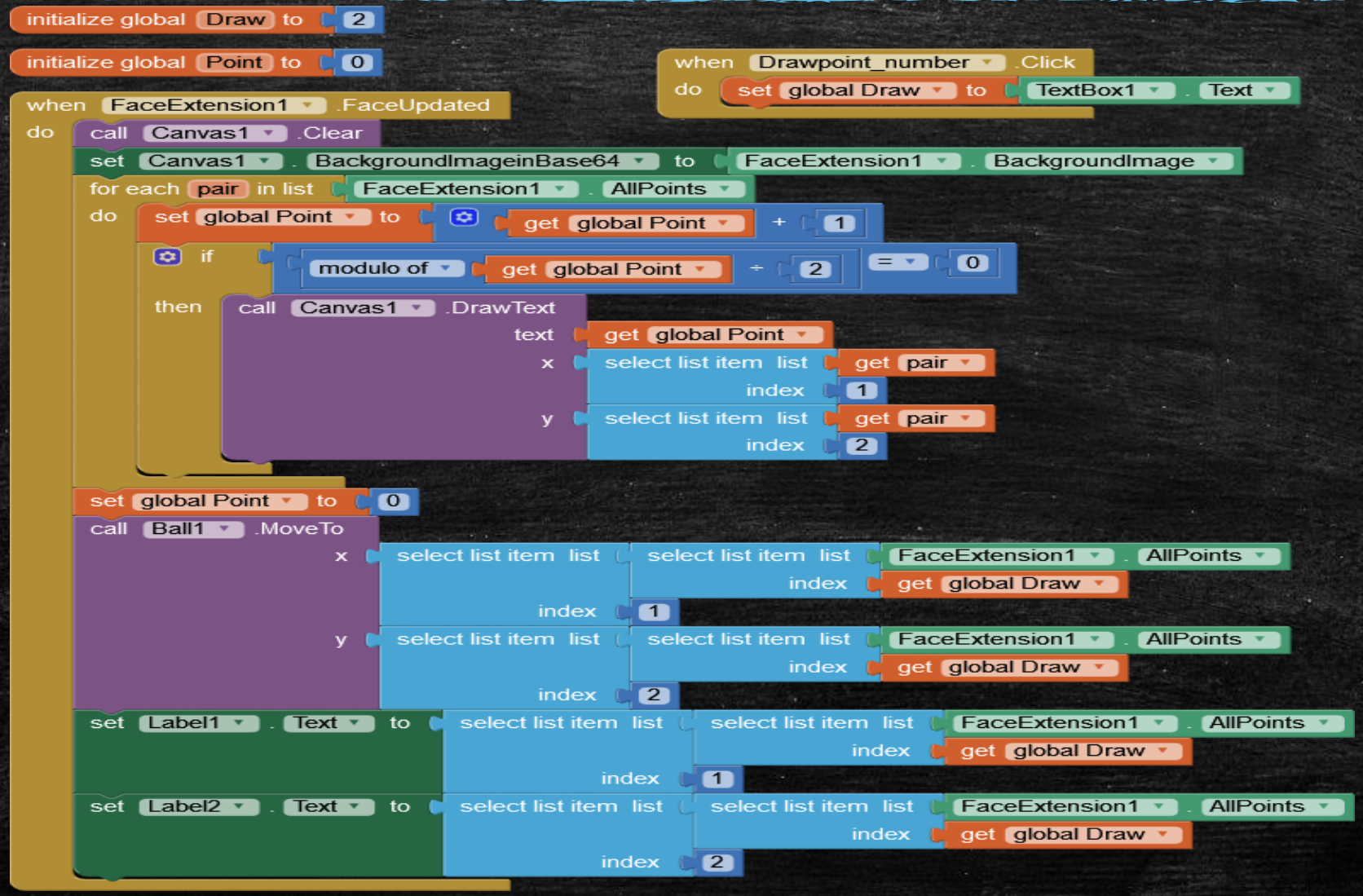
Another one Screenshot of designer section



Designer Section of Face detection



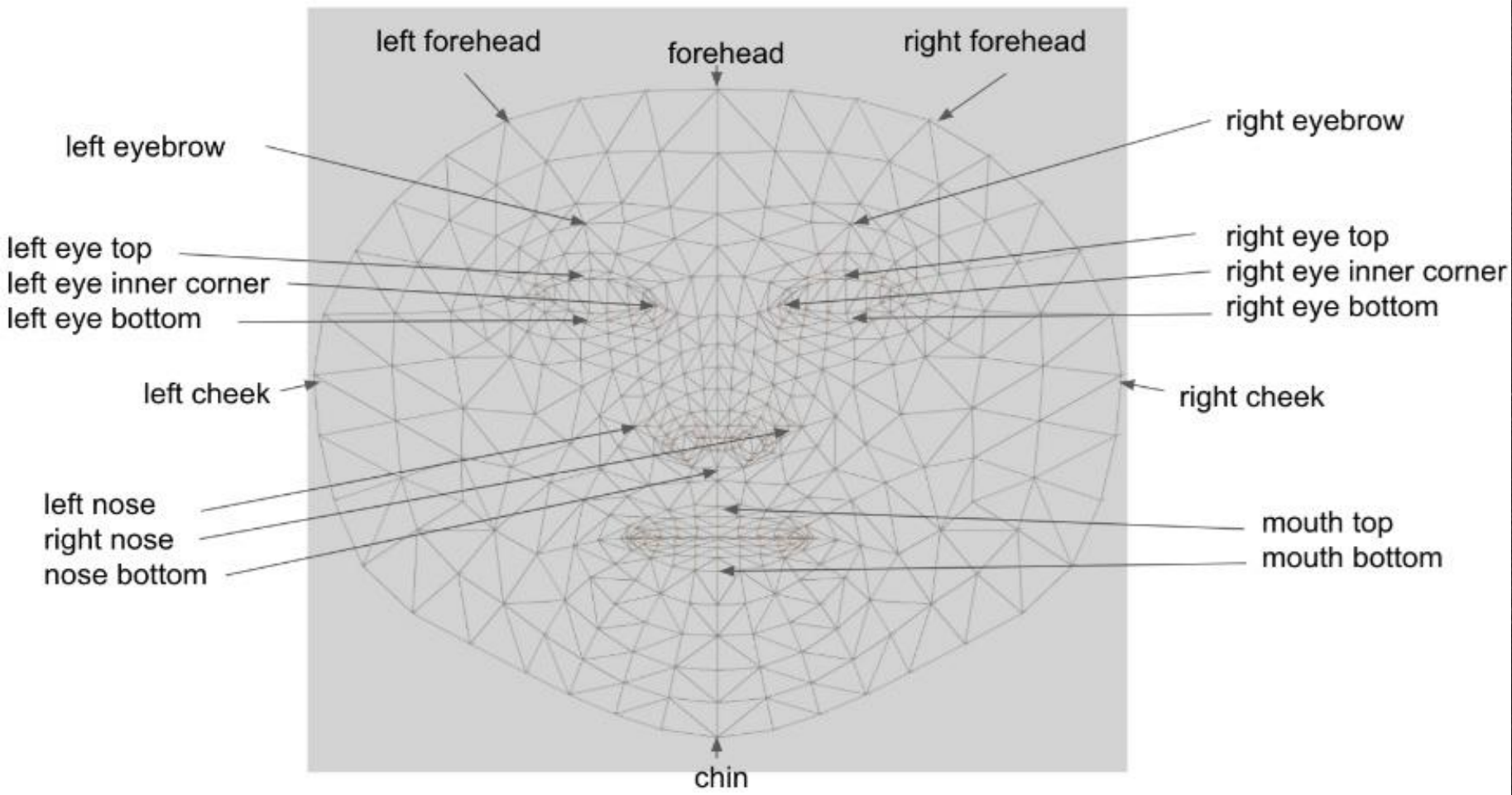
Block section



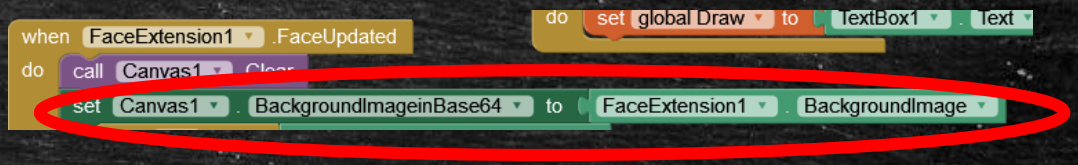
About this Face detection

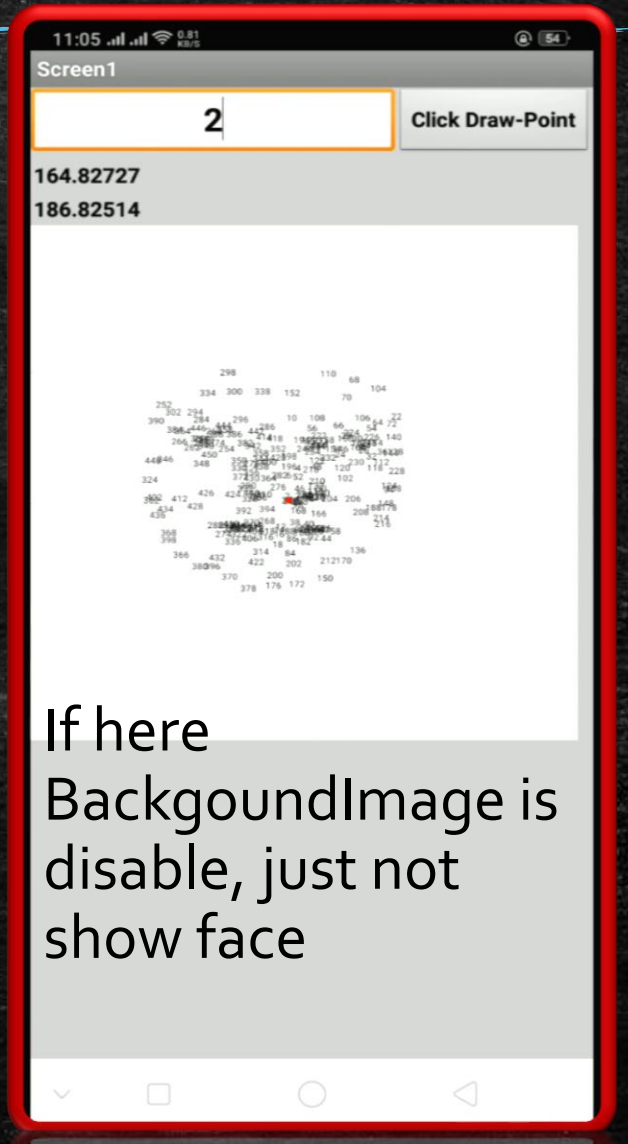
- We direct the front camera of our versatile towards us, we notice our face.
- The augmentation takes 450 focuses from our face. Lattice.
- If we mark the Show Mesh Property we will see the situation of each point all over.
- They look tiny and extremely near one another.
- So we will make an application to see the quantity of each point in a more clear manner.
- I have just put the even focuses, since altogether there are 450 and if we put every one of them they will blend in with one another.
- As we move, these focuses will change their position x, y
- Each point has two directions, in a table: 1 - > x 2 - > y
- A few focuses are found straightforwardly like Chin, Forehead, LeftEyeBottom etc.

About this Face detection



Testing Mobile





Nafiz Md Imtiaz Uddin
江西理工大学
Computer Science and Technology(2019-2023)
15679763939@163.com