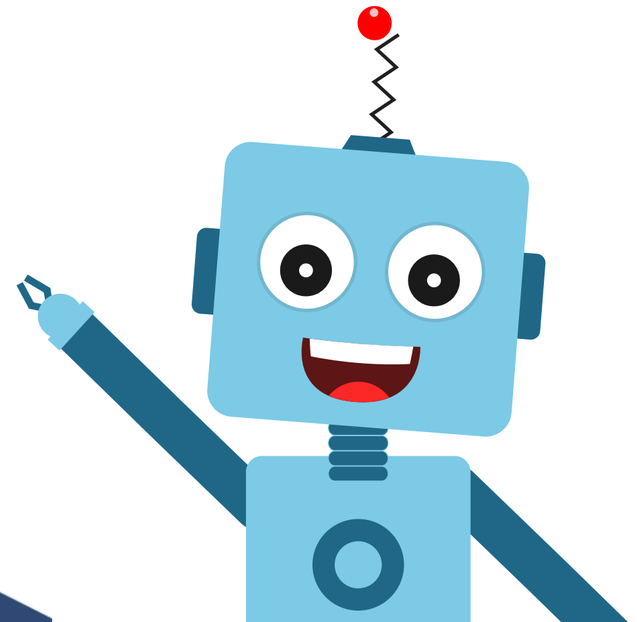
POWERED BY STEMpedia

Face Emotion Detection

Session 15



Topics covered

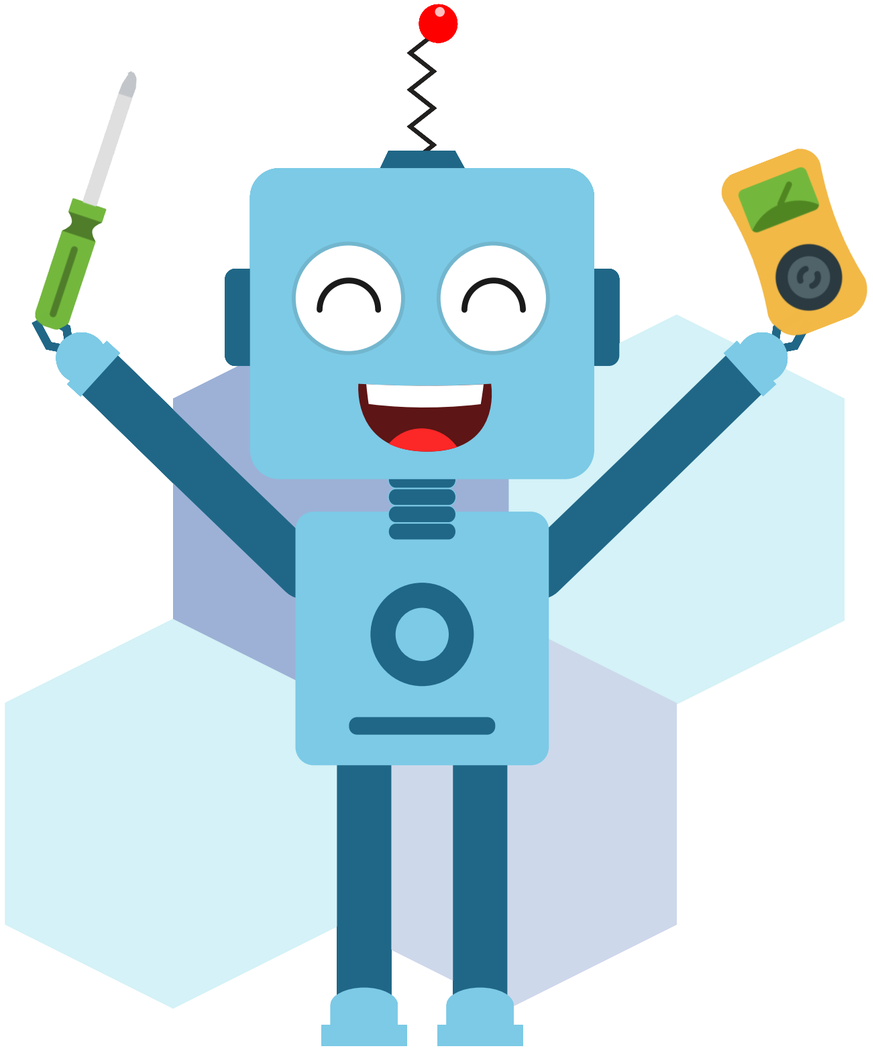
POWERED BY STEMpedia

1. Activity :
    1) Face Expression Detector
    2) Mimic Face With Quarky

ACTIVITY

POWERED BY STEMpedia

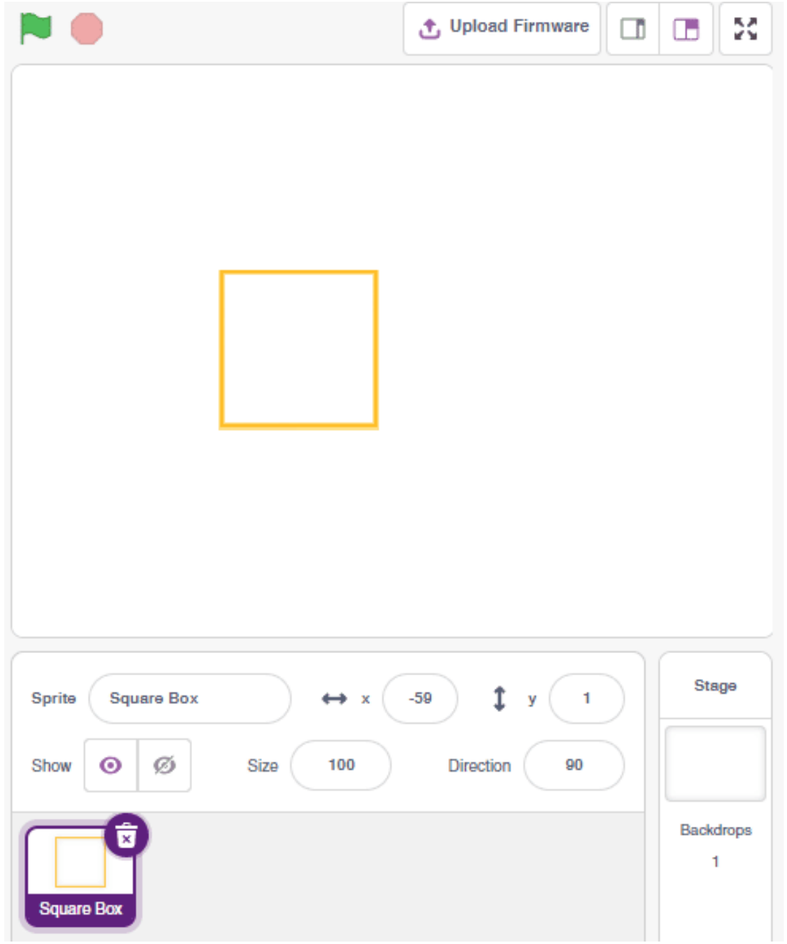
Face Expression Detector



Setting Up the Stage

POWERED BY STEMpedia

- Open PictoBlox and create a New File.
- Select the coding environment as Python Coding.
- Add a new Square Box sprite and delete the Tobi Sprite.



Let's code

POWERED BY STEMpedia

- Inside the script, the sprite object is already initiated by default. We need to initiate two more objects; Facedetection and Quarky.
- The code then looks as follows:

```
sprite = Sprite('Square Box')
fd = FaceDetection()
quarky = Quarky()
```

- Now, we turn 'on' the video on the stage from the device camera and keep the transparency to 0.
- We use the video() function of face detection object for this.

```
# Turn the video ON with 0% transparency
fd.video("ON", 0)
```

Recognize Image

POWERED BY STEMpedia

- We will continue with the same script.

- In order to analyse image, that is being captured from the camera and shown on the image we use an analysestage() function.

- We want this function to run continously, for this we put the above function inside a while loop that runs forever.

```
# Keep analyzing the camera forever
while 1:
    fd.analysecamera()
```

Display Emotions and Tracking

POWERED BY STEMpedia

- Now to **display emotions on the stage**, that the camera detects from the camera we use an **expression()** function and add it as an argument inside the **say()** function.

```
sprite.say(fd.expression())
```

- Now **to track the face on stage**, we need to get three attributes of the detected face: **x-coordinate**, **y-coordinate** and **width**. For this we use **x()**, **y()** and **width()** functions from face detection class.
- We also need to set the attributes of square sprite according to the above detected values. For this we pass the above values to the respective sprite functions: **setx()**, **sety()** and **setsize()** as shown below:

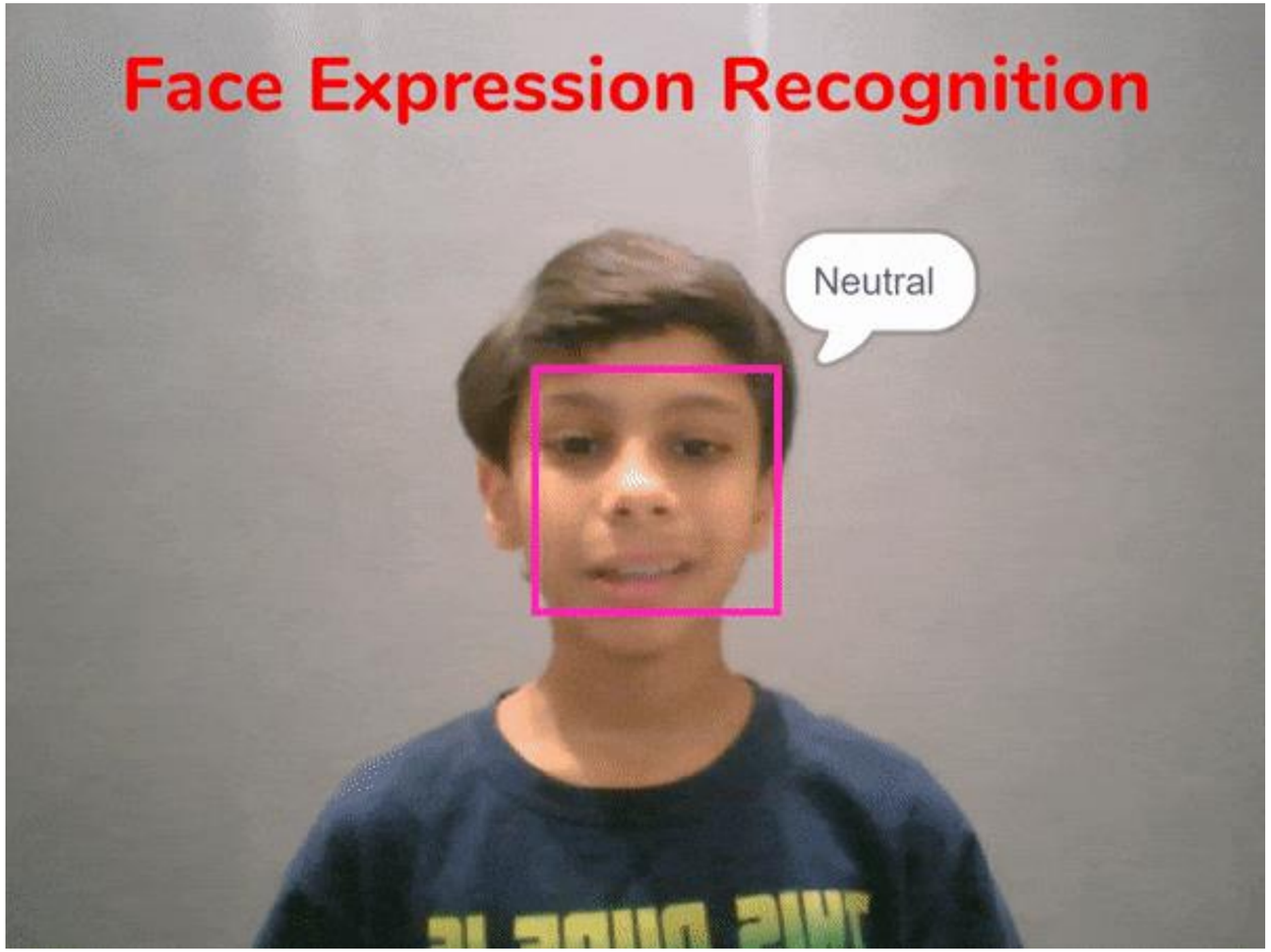
```
sprite.setx(fd.x(1))
sprite.sety(fd.y(1))
sprite.setsize(fd.width(1))
```

Final Code

POWERED BY STEMpedia

```
sprite = Sprite('Square Box')
fd = FaceDetection()
quarky = Quarky()
# Turn the video ON with 0% transparency
fd.video("ON", 0)


while 1:       # Run this script forever
  # Analyse image from camera
  fd.analysecamera()
  sprite.say(fd.expression())
   # Say the face expressions
  sprite.setx(fd.x(1))
  sprite.sety(fd.y(1))
  sprite.setsize(fd.width(1))
```

Final Output

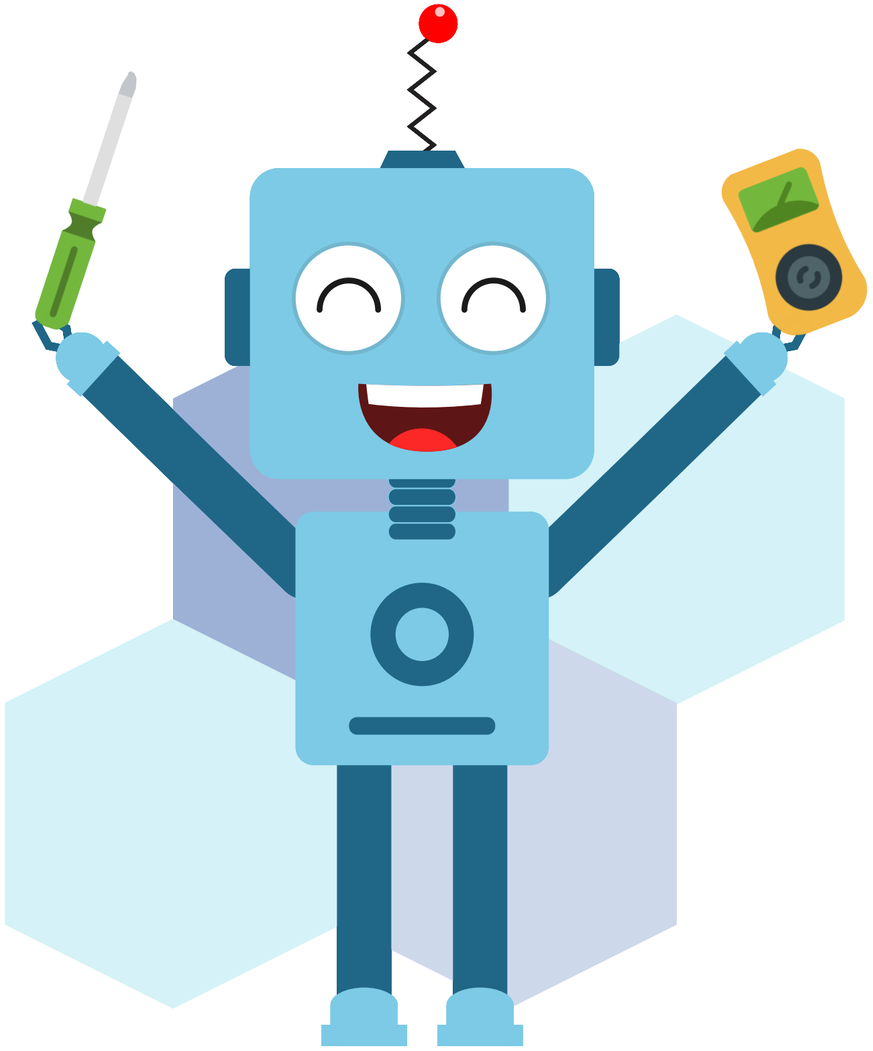


ACTIVITY

POWERED BY STEMpedia

Mimic Face with Quarky

In this activity, we're going to write a script that will detect our facial expressions, these facial expressions will then be mimicked by the Quarky robot.



Initiating objects

POWERED BY STEMpedia

- Open **PictoBlox** and create a **New File**.
- Select the coding environment as **Python Coding**.
- Add a new **Square Box** sprite and delete the **Tobi** Sprite.
- Inside the script, the sprite object is already initiated by default. We need to initiate two more objects; **Facedetection** and **Quarky**.
- The code then looks as follows:

```
sprite = Sprite('Square Box’)

fd = FaceDetection()
quarky = Quarky()
```

Turning on video on Stage

POWERED BY STEMpedia

- Now, we **turn 'ON' the video on the stage** from the device camera and keep the transparency to **0**. We use the **video()** function of face detection object for this.

```
# Turn the video ON with 0% transparency
fd.video("ON", 0)
```

- We also need to **display a bounding box around the face** that will be detected from camera on the stage, for this, we can use the **enablebox()** function.

```
fd.enablebox()
```

Analyzing Face Expressions and Mimicking with Quarky

POWERED BY STEMpedia

- In order to analyse image, that is being captured from the camera and shown on the image we use an **analysestage()** function.
- We want this function to **run continously**, for this we put the above function inside a while loop that runs forever.

```
# Run this script forever
while 1:
    fd.analysecamera()
```

- To display emotions on the stage, that the camera detects from the camera, we use the **expression()** function of Face Detection class and add it as an argument inside the say() function of Sprite class.

```
sprite.say(fd.expression())
```

Analysing Face Expressions and Mimicing with Quarky

POWERED BY STEMpedia

- Now, we want to check, **whether the face detected is showing happy emotion or not?**
- For this we use the **isexpression()** function of **Face Detection** class.
- We want Quarky to show/mimic the happy emotion too, for this we use **showemotion()** function of **Quarky** class.

```
# if face expression is happy
if fd.isexpression(1, "happy"):
  # show happy emotion on Quarky
   quarky.showemotion("happy")
```

Analysing Face Expressions and Mimicing with Quarky

POWERED BY STEMpedia

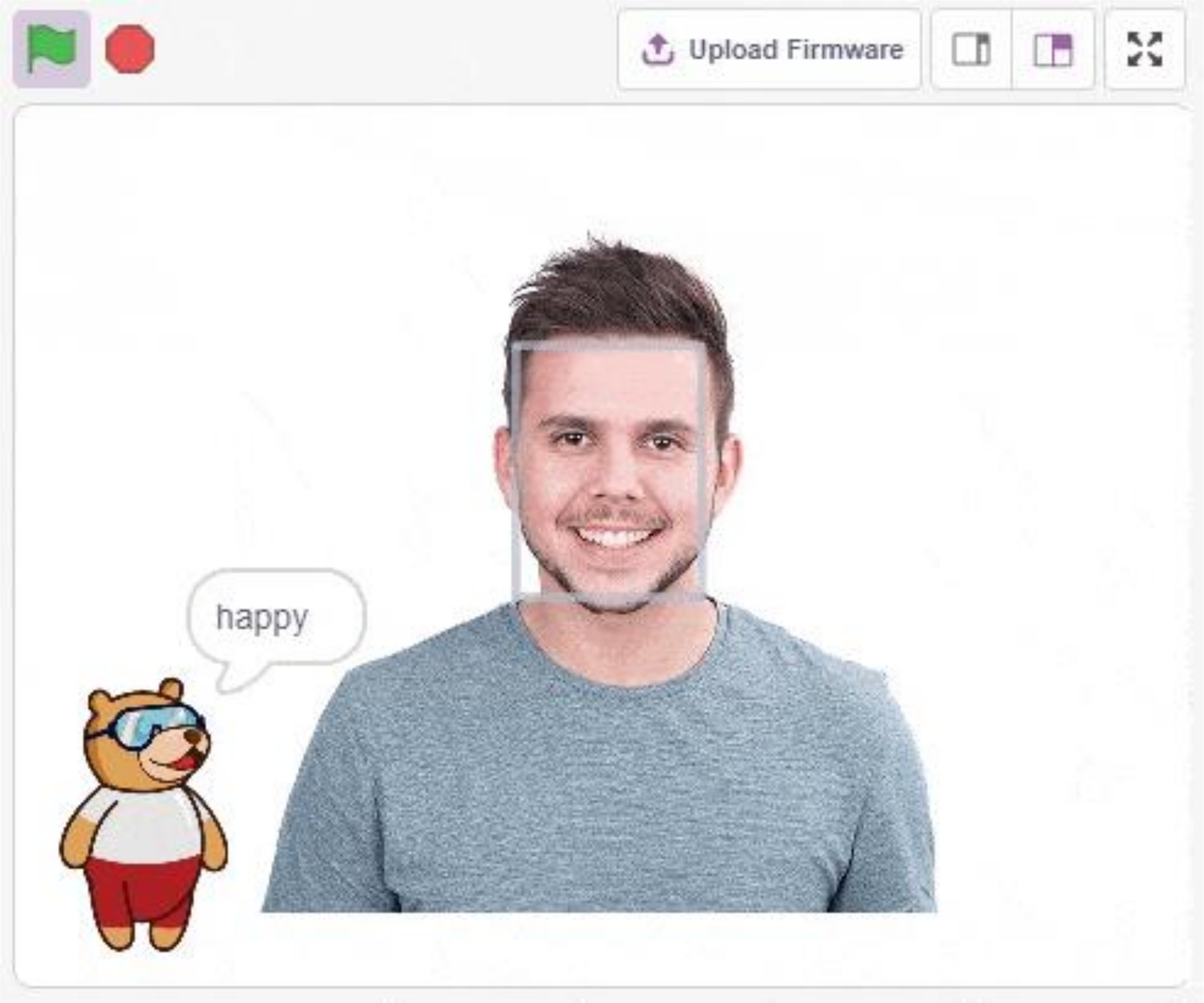
- Similarly we mimic other emotions on Quarky like sad, surprise and angry.

```
# if face expression is happy
  if fd.isexpression(1, "happy"):
    quarky.showemotion("happy"# show happy emotion on Quarky

  if fd.isexpression(1, 'sad'):    # sad emotion
    quarky.showemotion("crying")

  if fd.isexpression(1, 'surprise'):  # surprise emotion
    quarky.showemotion('surprise')
```

Final Code

```
sprite = Sprite('Tobi’)
fd = FaceDetection()
quarky = Quarky() # Turn the video ON with 0% transparency
fd.video("ON", 0)
fd.enablebox() # Run this script forever


while 1:
    fd.analysecamera() # Analyse image from camera
    sprite.say(fd.expression()) # Say the face expressions
    if fd.isexpression(1, "happy"): # show happy emotion on Quarky
     quarky.showemotion("happy")
    if fd.isexpression(1, 'sad'): # sad emotion
     quarky.showemotion("crying")
    if fd.isexpression(1, 'surprise'): # surprise emotion
     quarky.showemotion('surprise’)
    if fd.isexpression(1, 'angry'): # angry emotion
     quarky.showemotion('angry')
```

Final Output



ACTIVITY

POWERED BY
STEMpedia