# Aerial Robotics Kharagpur Documentation Task 2

Agnibha Sinha *

***Abstract*— Face Detection Game documentation**

**This documentation is for the face detection game made using opencv c++. The program reads the live video from the webcam and detects the face. It correspondingly maps the face to a line in the game and uses it to make collisions with a ball moving in gravity free space. The game ends when the ball misses the line.**

## I. INTRODUCTION

The game can be broadly divided into 2 parts. Firstly we needed to detect the face from the live video from the webcam and secondly we needed to track the motion of the ball through its position and velocity. These 2 things were combined and mapped into a single output screen with the logic for the game to end and counting the number of collisions with the face. For the face detection, a few inbuilt functions have been used and for the tracking of the ball, the position has been updated according to the velocity and the velocity has been updated after every collision and all collisions have been assumed perfectly elastic. The initial velocity can be changed and this can be used to speed up or slow down the game. The whole game has been implemented using c++ and opencv only.

## II. PROBLEM STATEMENT

The problem statement was to make a game which we can play by moving our face in the video captured by the webcam. The objective of the game was to not let the moving ball collide with the floor and move our face to such a position that the ball would collide with the face. The ball would be moving in gravity free space and undergoing perfectly elastic collisions with the walls.

## III. INITIAL ATTEMPTS

Initially the face detection was tested using inbuilt functions. Then the tracking of the ball was tested separately using different initial velocities. Initially velocities were taken as integers but they could be taken as floating points also and mapped to integers only during plotting them on the output image which improves the accuracy of the ball tracking. The 2 parts were then merged together in the game.

## IV. FINAL APPROACH

**Ball Tracking:** Firstly we initialize the position and velocity to values of our choice. Next we add the velocities to the corresponding positions along the X and Y axis to get the new positions as we assume the time for one iteration to be unity.

$$xnew = x + velx$$
$$ynew = y + vely$$

If the new positions go out of bounds of the image, there is a collision with the walls and the corresponding velocity is reversed. For example,

$$if(xnew <= 0 || xnew > img.rows)$$
$$velx = velx * (-1);$$

Similar operations are done on position and velocity for y direction. The magnitude of the velocity does not change as the collisions are considered elastic. Therefore the initial velocity assigned becomes very important and decides the speed of the game.

The visual output is a solid filled circle drawn in red with the center as (x,y) which are the coordinates of the current position and radius as 3. Before drawing this, a black circle is drawn with the previous coordinates as center in each iteration. This ensures that the previously drawn circle is erased in a way as the background is also black. Therefore only the circle corresponding to the current coordinate is seen as output.

**Face Detection:**

The live video from the webcam has been taken using the VideoCapture function available in opencv. The process is operated in a loop and each frame from the video is processed and the corresponding output frame is displayed before going to the next frame.

The function used for face detection takes in the frame, cascade classifier and the scale as parameters. It locates the centre of the face and returns it. The prototype for the function is:

Point detectAndDraw(Mat& img, CascadeClassifier& cascade, double scale);

This function needs haarcascade xml files to work which have been included in the program. This function uses in built function “detectMultiScale” which helps detect the face.

The center of the face is detected and a blue circle is drawn around it. The coordinates of the centre are returned by the function.

**Mapping the detected face into the game:**

The function defined for face detection returns the center of the detected face. In this game only horizontal movement has been allowed and hence we are only interested in the x coordinate of the centre. 50 pixels on either side of the centre has been marked as the allowed region for collisions. Thus the center of the face detected is mapped into a straight horizontal line in green color whose midpoint is at the same x coordinate and has y coordinate at 400 and has a length of 100 pixels.

*Write anyone who might have helped you accomplish this eg any senior or someone

The y coordinate for the floor has been taken as 400. Thus the game ends immediately if the ball reaches y=400 and is not on the green line.

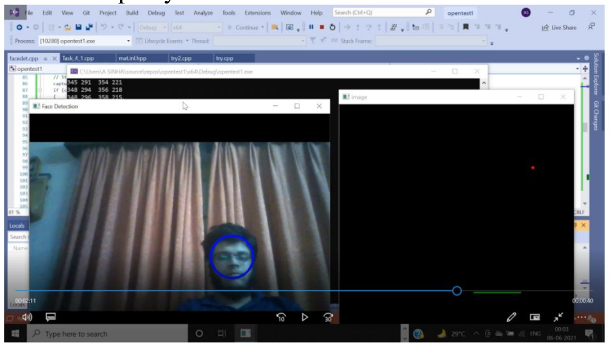
The collisions with the green line are also elastic so the velocity is updated in a similar way. In case a face is not detected, the centre returned is (0,0) in which we automatically set the midpoint of the green line to (250,400) which is the center of the screen on the line y=400.

**Playing the game:** The green line can be moved horizontally by moving the face with respect to the webcam. We try to move the line in a way such that the ball collides with it. In case the ball misses the line, the game ends. The speed of the game can be controlled by varying the initial velocity. The score is calculated based on the number of collisions with the green line. The number of lives can also be initialized before.

## V. Results and Observation

The game could be played but sometimes the face is not detected and there may be a lag. The speed of the game can be controlled but if it is too fast the face detection can lag a lot. To account for this the speed is recommended to be kept slow.

Here is a video of the running game and an image from the same: https://youtu.be/eS55A5xZ5T0



```
345 291  354 221
348 294  356 218
348 296  358 215
```

## Conclusion

It was a nice challenging task and I am happy that I was able to make a workable game in the end.

## References

[1] https://www.geeksforgeeks.org/opencv-c-program-face-detection/
[2] https://www.youtube.com/watch?v=2FYm3GOonhk&t=8525s