**Game Instruction**

Move your face side-to-side to control the movement of the car character within the game.

Score more points by loading the smiley faces in your car trunk. (Just get the car in contact with the smiley face)

Remember, you have to avoid the bombs ( PS - things can get quite tricky)

Good luck!! Have FUN!!!

**How to run**

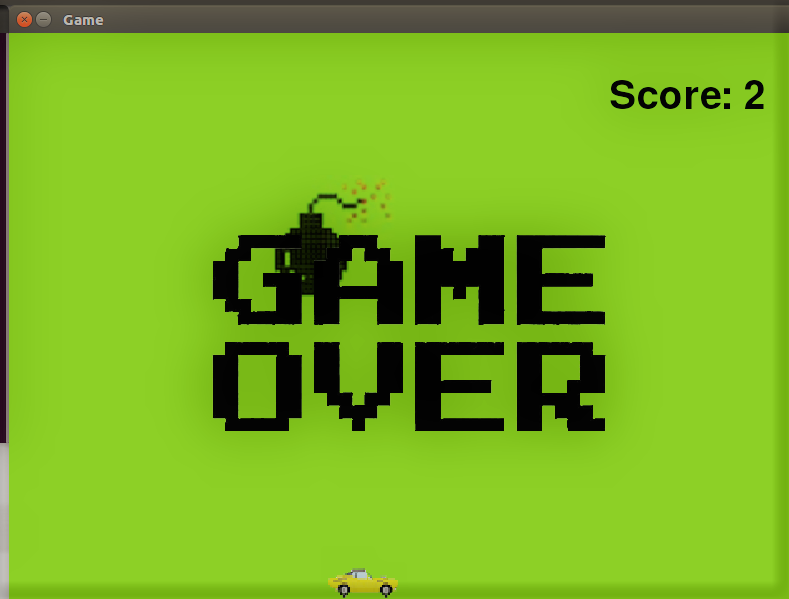
1. Install Python3
2. Install Pygame, Numpy and OpenCv
3. Run “Game.py” to play!

**MINI PROJECT WRITE-UP AND REFLECTION**

**Project Overview**

In Mini Project 4, we explore interactive programming, an instance of interactive computation which accepts input from the user as a program runs, and uses it as a tool in game creation. In this project, our team attempts to implement a game that takes user input from user facial-image data using facial recognition techniques. We created a game where a character, controlled by side-to-side movements of the face, has to get in contact with smiley faces for more points in score and dodge incoming bombs in order to remain in the game.

**Results**



We were able to program a game fully-controlled by side-to-side face movements. Using the CV2 package for facial detection, we were able to track the position of the philtrum and nose point. Using the pixel coordinates of the nose point as input, we were able to manipulate our car object along the x-axis making it resemble a facially manipulated game. The team decreased the sensitivity of the facial recognition to reduce the glitchy nature of the car.

We were also able to implement a score mechanism in place to introduce some sense of purpose in the game. Also, with the addition of bomb-art within the game, and necessary model changes, we were able to introduce a crash mechanism for ‘game over’ within the game. By moving the face side-to-side, the user is thus able to control a character (car art also moving side to side) and play our game without user input from the keyboard or mouse. The ‘glitchy’ motion of the bomb was also as a result of a change in the image being blitted onto the screen at a high frequency.

**Implementation**

Dodgeball game is composed of certain basic classes.

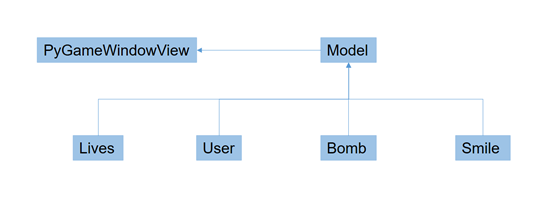
The Bomb class assigns attributes to falling bombs. It generates bomb in random position in its horizontal axis with increasing y coordinate of certain speed. The bomb falls in a constant speed from the top to the bottom of the screen if it does not collide with player controlled character.

The Smile class assigns attributes to falling smiley faces. It works in same way as Bomb class. The Lives class notes the lives a player has for one round of game.

The User class generates player controlled character. Facial recognition is used to grab x coordinate of player’s face through a pre-trained cascade file. One design decision we had here is to decrease the sensitivity of x-coordinate changed on the screen by filtering change of x-coordinate through a threshold value. Hence, the player controlled character can move more smoothly on the screen.

The Model class assembles Bomb, Smile, Lives and User. It builds two bombs and one smiley face and keeps tracking of their positions to check if a collision is happened between objects and player, then reflects the collision in certain attribute (lives and score).

The PygameWindowView draws all those objects created in our Model class.

**Reflection**

Initially, our group began designing our software with so many nested functions. One lesson that we learned is that once a code is written without a clear map of Object Oriented Programming relationships and detailed class structure, the code gets pretty messy and it is very time-consuming to rewrite all the code in OOP structure. While our code seemed to work on the surface, it didn’t incorporate Python’s many Object Oriented Programming features. As a result, our team had to redesign and refactor our existing code using Python’s classes, attributes and instances. Refactoring the code was definitely a difficult task.

Merge conflicts were sometimes frustrating, but we used them as an experience to learn more about GIT for version control and collaborative programming. We can definitely improve the conciseness and preciseness of our code. Through this project, we really got our hands on objected-oriented programming.