Human Computer Interaction (CSE 4015)
(SCOPE)

GAME CONTROL USING CUSTOM HAND GESTURES
J COMPONENT REPORT

Submitted by

ADITYA ROHILLA (18BCE0929)
KHWAAB THAREJA (18BCE0930)
D.V.R ADITYA (18BCE0949)

Submitted to

Prof. SHASHANK MOULI SATAPATHY

IN

B.Tech. Computer Science Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

INDEX

ABSTRACT

Gesture recognition is a type of perceptual computing user interface that allows computers to capture and interpret human gestures as commands. Since the gaming industry is on a rise in India in the form of esports, this project on its own can improve the current status of motion gestures in gaming. This Project aims to use gesture technology to play Snake Game (or any similar game) by detecting gestures to control the movement of the snake on the screen. It has been implemented in Python using the OpenCV library. Although for the demo purposes we built a 2-D Snake Game and the implementation is shown in the video link provided at the end of the project. It can be used to play any game or control any application using gestures only. The gestures replace the actions of the keyboard and mouse . The novelty of this project is that there is no project where people can define their own custom gestures to play any game. It improves the user experience aspect by letting the users play simple games in a fun and intuitive manner. There are many ways to achieve this like using AR and VR but those options are not often affordable by the general public. Our project just uses the webcam which makes it far more feasible. The target audiences for this work are pre-teens and teenagers who like to explore technology and play different games.

INTRODUCTION

The general definition of gesture recognition is the ability of a computer to understand gestures and execute commands based on those gestures. In the world of gesture recognition, a gesture is defined as any physical movement, large or small, that can be interpreted by a motion sensor. The gestures replace the actions of the keyboard and mouse .

OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage, then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source BSD license. OpenCV also supports the deep-learning frameworks TensorFlow, Torch/PyTorch and Caffe.

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.

The PyAutoGUI library provides cross-platform support for managing mouse and keyboard It lets your Python scripts control the mouse and keyboard to automate interactions with other applications to enable automation of tasks. These scripts can control other applications by sending them virtual keystrokes and mouse clicks, just as if you were sitting at your computer and interacting with the applications yourself. The pyautogui library works on Windows, macOS, and Linux, and runs on Python 2 and 3. A tool like this has many applications, a few of which include taking screenshots, automating GUI testing , automating tasks that can only be done with a GUI, etc.

Related work

[P1] Image processing algorithms for gesture recognition using MATLAB

The paper talks about the gesture recognition processes in which the gestures or postures of human body parts are identified and are used to control computers and other electronic appliances. The most contributing explanation behind the developing gesture recognition is that they can make a basic communication way among humans and computers called HCI. This paper reasons out the ways to identify hand postures and establish a man-machine interaction. The color image is converted into a binary image and preprocessed and the number of fingers is counted using the scanning method in MATLAB.

[P2] Hand gesture recognition system using image processing

The paper refers to gestures as a form of non verbal and non vocal communication and tells how bodily actions can be used to communicate particular messages in speech. These types of gestures may involve various body parts and expressive displays of these parts to enhance the accuracy. With such gestures, individuals may even be able to portray a variety of feelings and thoughts. Such with the help of image recognition techniques can be useful for detection of messages by a computer and will establish an interaction between computer and humans.

[P3] Detection of moving objects through color thresholding

The paper uses intensity of image to differentiate the regions between the background and the objects. These techniques are based on segmentation and image processing area. If the contrast has a high intensity, the threshold based on the concept of segmentation observes a high accuracy in classification of pixels. Further, the paper talks about the concepts to identify objects by means of their color (thresholding), this technique was implemented in the development of a game program. Going forward, different colours might have different detection ranges however red, yellow and green colours were found to provide better results in the process of object detection.

[P4] A hidden Markov model based dynamic hand gesture recognition system using OpenCV

In this paper, it proposes a novel and quicker framework for dynamic hand gesture recognition by utilizing OpenCV. Many hand signal acknowledgment techniques utilizing visual investigation have been proposed. In the paper, the hidden Markov model (HMM) is proposed for hand signal recognition. The entire framework is divided into three detection and tracking, feature extraction and training and recognition. the use of OpenCV's inbuilt functions, the system is easy to develop, its recognition rate is quite fast and so the system can be practically used for real-time applications.

[P5] Solving The Classic Snake Game Using AI

The paper represents a survey of a variety of papers for the snake game in order to compare its traits and key elements. The paper talks about the AI bot, and corresponding algorithms for the enhanced computer playing skills and techniques. Players can follow the simultaneously running AI bot to play the game effectively. It includes three searching algorithms related to artificial intelligence, Best First Search, A* Search and improved A* Search with forward checking, and two baseline methods Random Move and Almighty Move.

[P6] Single Object Trackers in OpenCV: A Benchmark

The paper talks about the use of object tracking with the help of computer vision and depicts its applications in human-computer interaction, video surveillance, medical treatments, robotics, smart cars, etc. The paper provides insights into how to choose an object tracking method from the widespread OpenCV library. We provide benchmarking results on the OTB-100 dataset by evaluating the eight trackers from the OpenCV library.

[P7]: Combining Hand Detection and Gesture Recognition Algorithms for Minimizing Computational Cost

The paper proposes a solution for the problem that convolution layers of neural networks take a significant part of computer resources even if the target object is absent in the frame. It provides a solution with a combined

hand gesture recognition system that uses a hand detector to detect hand in the frame and then switches to gesture classifier if hand was detected.

[P8]: Hand motion based mouse cursor control using image processing technique

This paper proposes to develop a method and model that identifies hand motions which can be used as an input command to work together with the workstation system. It created a method to recognize the motions and built-in-task for every motion. The technique mainly focused on the use of webcams which is then used to develop a virtual Person Workstation Interaction Device.

[P9]: Gesture Detection in Digital Image Processing based on the Use of Convolutional Neural Networks

The paper talks about the findings obtained from tests of an image detection system utilizing the YOLO network. The main objective of the paper was to display the development of a system offering real-time gesture detection from camera feed images. Potential applications of the postulated gesture recognition system include toy control systems, photography, and broadly defined office assistance.

[P10]: Hand Gesture Recognition for Non-Contact Control of a Technical System

The article proposes a method and algorithm for determining the hand gesture on a video sequence from only one optical camera. The approach based on the analysis of cluster characteristics allows implementing the embedded algorithm on real-time ARM computers. To increase the effectiveness of the system, the contributors proposed their own clustering algorithm. And also, an example of applying the implementation algorithm for a gesture of squeezing the palm of a hand into a fist followed by unclenching of a fist.

Applications

Gesture recognition is the mathematical interpretation of a human motion by a computing device. Gesture recognition, along with facial recognition, voice recognition, eye tracking, and lip movement recognition are components of what developers refer to as a perceptual user interface (PUI). The goal of PUI is to enhance the efficiency and ease of use for the underlying logical design of a stored program, a design discipline known as usability. In personal computing, gestures are most often used for input commands. Recognizing gestures as input allows computers to be more accessible for the physically-impaired and makes interaction more natural in a gaming or 3-D virtual reality environment. Hand and body gestures can be amplified by a controller that contains accelerometers and gyroscopes to sense tilting, rotation, and acceleration of movement -- or the computing device can be outfitted with a camera so that software in the device can recognize and interpret specific gestures. A wave of the hand, for instance, might terminate the program. In addition to the technical challenges of implementing gesture recognition, there are also social challenges. Gestures must be simple, intuitive, and universally acceptable. The study of gestures and other nonverbal types of communication is known as kinesics.

The major application areas of gesture recognition in the current scenario are:

Automotive sector - Gesture recognition technology is widely expected to be the next generation in-car user interface. Gesture recognition determines whether the driver has performed recognizable hand or finger gestures within an allotted space without contacting a touchscreen. For example, an approaching hand can activate the in-car infotainment system, or in a more sophisticated system, the driver can touch the steering wheel then tilt his head left or right to turn the volume of the stereo up or down. A camera placed in the steering wheel or on the dashboard is programmed to watch for certain gestures. When it sees them, it sends a signal to the processor that handles the connected infotainment hardware. The data is analyzed to determine what the driver is doing, ascertain which display controls the driver wants to adjust and then activate the appropriate features.

Consumer electronics sector - The major application of Gesture Recognition in consumer electronics includes smartphones, tablets, laptops, PCs, televisions, set-top boxes, and others. Gesture recognition in smart television enables the viewers to control their smart TVs with finger and hand gestures. A Smart TV recognition engine tries to match the user's hand gestures to a predefined set of gestures. This operation can be performed locally by the TV. The gesture recognition software can either directly be integrated on a smart TV platform or it can be used on an accessory camera. Gesture recognition for PCs and desktops has been one of the trending aspects in the modern world of gadgetry. Human gesture recognition is one of the newest ways to input information or control a computer. This gesture recognition technology enables the user to naturally and intuitively interact with the computer making the PC viewing and usage experience more interactive and realistic.

Gaming sector - The growing gaming industry is also likely to be a major driver for the global gesture recognition market over the forecast period. While the use of gesture recognition technology in gaming applications is relatively new, its potential is immense, as gesture recognition technology forms a natural fit for the gaming industry, as it provides users with a more personal and interactive way to control the game. The use of gesture recognition technology has been highly popular in the gaming industry, which is likely to be a major driver for the global gesture recognition market over the forecast period.

Defence - In spite of the fact that there are numerous robots controlled using commands from user or self-controlled that uses GPS and sensors, the requirement for gesture-controlled robots is on the ascent for military purposes, which is called Unmanned ground vehicles (UGVs). These robots are utilized to increase the warrior's capacity in open territory. In the last few years, tremendous research is going on in various parts of the world to develop robots for military purposes. This inspiration helped us fabricate a prototype gesture-controlled robot (called UGV) to embrace missions like border patrol, reconnaissance, and in dynamic battle both as a standalone unit (automatic) and as well as in coordination with human soldiers (manual)

Smart Homes - Take an example of a smart house where a hand recognition software allows us to track a hand's 3D position, rotation and gesture. This

allows smart house owners to operate the lights without ever touching the switch, simply by waving their hand in front of it to activate the lights.

WORKING METHODOLOGY

In this Project, We built a Classic 2-D Snake Game named 'SNAKE FUN' by using the PyGame library in Python. It is a free and open-source Python programming language library used for making a multimedia application like games. To create the snake, We first initialized a few color variables in order to color the snake, food, screen, etc. In this snake game, if the player eats the apple the size of the snake will increase if it hits the boundaries or the snake collides with his own body, the game is over then he loses and the final score will be displayed along with the leaderboard showing top 10 scores.

Function Associated gestures are those gestures that use the natural action of the arm/hand/other body parts to associate or provide a cognitive link to the function being controlled. We used these Function Associated gestures for giving the input through the webcam and directing the snake accordingly.
In our case first, any green object with the largest area is detected and a circumcircle is generated with it's centroid.This is to negate the chances of detecting multiple green objects in the frame.Then the centroid is tracked for motion by subtracting from previous frames and the direction is determined.

Then finally, we integrate both these prospects by assigning a specific key for a gesture to bind them.We can use any green color object to direct the snake.
We can also define any RGB color to be detected instead of green but, green is recommended as it's the least likely color to be present in the background.
We used python libraries such as NumPy, imutils, PyAutoGUI, and OpenCV for the adding function associated gestures to our snake game.

INDIVIDUAL CONTRIBUTION

The individual contributions planned for the project are as follows:

Contributions by D VENKATA RAJESWARA ADITYA:

Created the snakegame.py file and added all features like leaderboard, game logic, GUI etc

Contributions by ADITYA ROHILLA:

Created the Gesture Control script for processing the color and identify the object based on image processing techniques

Contributions by KHWAAB THAREJA:

Integrated both gesture control and game aspect of the project using PyAutoGUI and binding the keyboard keys with the gesture defined.

SOFTWARE REQUIREMENTS

Although the Exe file of our project comes with all requirements as a package, if one wants to explore our project via an editor, the following modules are required to run:

numpy==1.15.2
imutils==0.5.1
PyAutoGUI==0.9.38
opencv_python==3.4.3.18
pygame==1.9.4

These can be installed using “pip” commands.

HARDWARE REQUIREMENTS

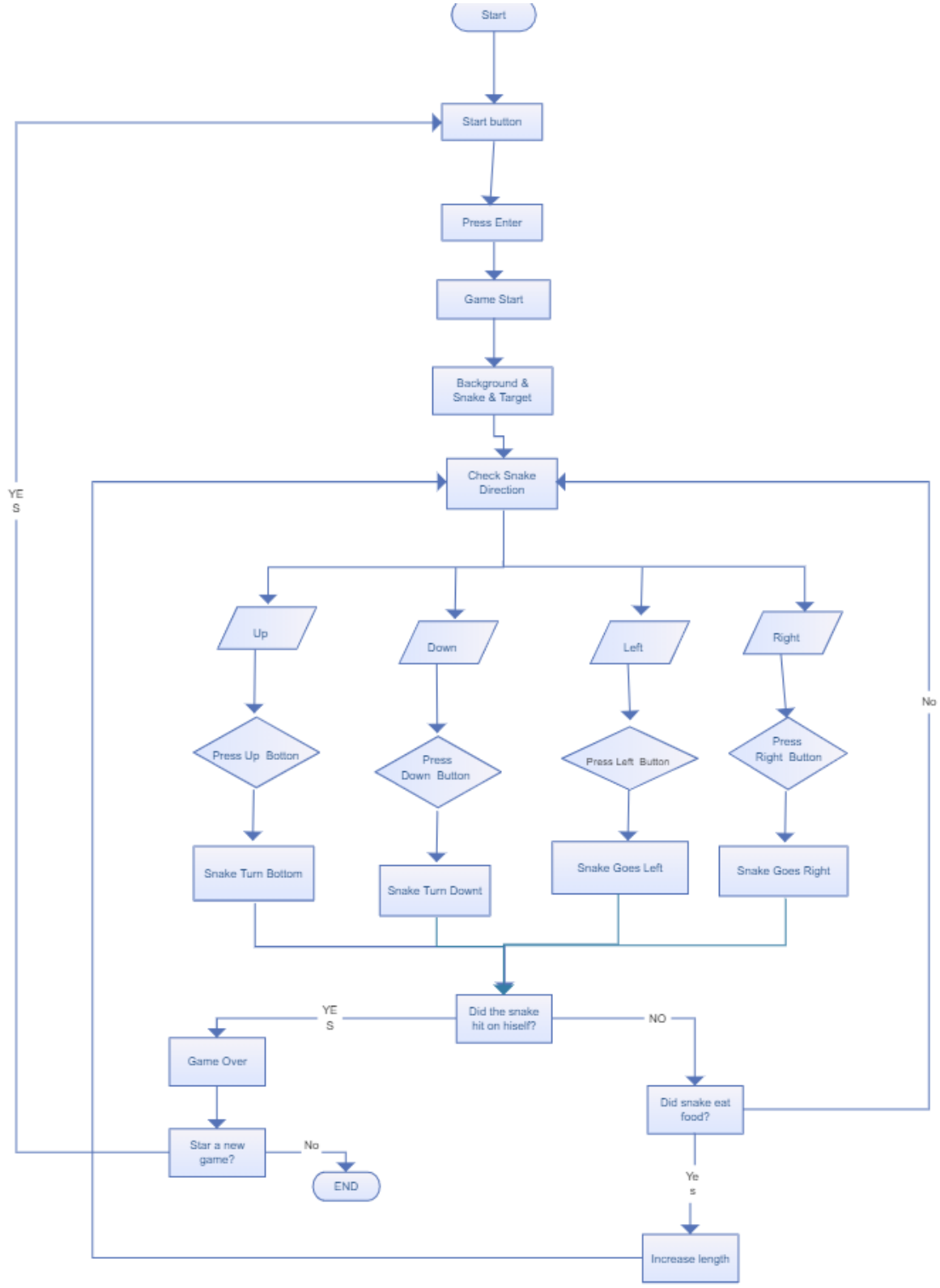
Best part about this project is that there are no major hardware requirements no run this program. Only the following are required:

Any PC with:
-Windows XP or greater
-intel core duos processor or higher
- 2 GB RAM

A decent Web camera for gesture control part

Any green object as the controller

PROPOSED SYSTEM PROCESS FLOW



SAMPLE CODE

GAME CODE:

SnakeGame.py

```python
import random
import pygame
import sys
from pygame.locals import *
from settingsSnakeFun import *
from Leaderboard import *

def main():
    global CLOCK, SCREEN, FONT

    pygame.init()
    CLOCK = pygame.time.Clock()
    SCREEN = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
    FONT = pygame.font.Font('freesansbold.ttf', 18)
    pygame.display.set_caption('Snake Game')

    showStartScreen()
    while True:
        pygame.mixer.music.play(-1,0.0)
        runGame()
        pygame.mixer.music.stop()
        showGameOverScreen()

def runGame():

    startx = random.randint(5, CELLWIDTH - 6)
```

```python
def runGame():

    startx = random.randint(5, CELLWIDTH - 6)
    starty = random.randint(5, CELLHEIGHT - 6)
    global wormCoords
    wormCoords = [{'x' : startx, 'y' : starty}, {'x': startx - 1, 'y':starty}, {'x':startx - 2, 'y':starty}]
    direction = RIGHT

    apple = getRandomLocation()

    while True:
        for event in pygame.event.get():
            if event.type == QUIT:
                terminate()
            elif event.type == KEYDOWN:
                if (event.key == K_LEFT or event.key == K_a) and direction != RIGHT:
                    direction = LEFT
                elif (event.key == K_RIGHT or event.key == K_d) and direction != LEFT:
                    direction = RIGHT
                elif (event.key == K_UP or event.key == K_w) and direction != DOWN:
                    direction = UP
                elif (event.key == K_DOWN or event.key == K_s) and direction != UP:
                    direction = DOWN
                elif event.key == K_ESCAPE:
                    terminate()
```

```
    if wormCoords[HEAD]['x'] == -1 or wormCoords[HEAD]['x'] == CELLWIDTH or wormCoords[HEAD]['y'] == -1 or wormC
        return

    for wormBody in wormCoords[1:]:
        if wormBody['x'] == wormCoords[HEAD]['x'] and wormBody['y'] == wormCoords[HEAD]['y']:
            return

    if wormCoords[HEAD]['x'] == apple['x'] and wormCoords[HEAD]['y'] == apple['y']:
        APPLEEATSOUND.play()
        apple = getRandomLocation()
    else:
        del wormCoords[-1]


    if direction == UP:
        newHead = {'x': wormCoords[HEAD]['x'], 'y': wormCoords[HEAD]['y'] - 1}
    elif direction == DOWN:
        newHead = {'x': wormCoords[HEAD]['x'], 'y': wormCoords[HEAD]['y'] + 1}
    elif direction == RIGHT:
        newHead = {'x': wormCoords[HEAD]['x'] + 1, 'y': wormCoords[HEAD]['y']}
    elif direction == LEFT:
        newHead = {'x': wormCoords[HEAD]['x'] - 1, 'y': wormCoords[HEAD]['y']}
    wormCoords.insert(0, newHead)


    SCREEN.fill(BGCOLOR)
    drawGrid()
    drawWorm(wormCoords)
```

```
def getTotalScore():
    return ((len(wormCoords) - 3) * 1000)

def drawPressKeyMsg():
    pressKeyText = FONT.render('Press A Key To Play', True, YELLOW)
    pressKeyRect = pressKeyText.get_rect()
    pressKeyRect.center = (WINDOWWIDTH - 200, WINDOWHEIGHT - 100)
    SCREEN.blit(pressKeyText, pressKeyRect)

def drawSettingsMsg():
    SCREEN.blit(SETTINGSBUTTON, (WINDOWWIDTH - SETTINGSBUTTON.get_width(), WINDOWHEIGHT - SETTINGSBUTTON.get_height()

def checkForKeyPress():
    if len(pygame.event.get(QUIT)) > 0:
        terminate()

    keyUpEvents = pygame.event.get(KEYUP)
    if len(keyUpEvents) == 0:
        return None
    if keyUpEvents[0].key == K_ESCAPE:
        terminate()
    return keyUpEvents[0].key

def showStartScreen():
    titlefont = pygame.font.Font('freesansbold.ttf', 100)
    titleText = titlefont.render('SNAKE FUN', True, DARKGREEN)
    while True:
        SCREEN.fill(BGCOLOR)
```

```
def terminate():
    pygame.quit()
    sys.exit()

def getRandomLocation():
    return {'x': random.randint(0, CELLWIDTH - 1), 'y': random.randint(0, CELLHEIGHT - 1)}

def showGameOverScreen():
    gameOverFont = pygame.font.Font('freesansbold.ttf', 40)
    gameOverText = gameOverFont.render('Game Over', True, WHITE)
    gameOverRect = gameOverText.get_rect()
    totalscoreFont = pygame.font.Font('freesansbold.ttf', 20)
    totalscoreText = totalscoreFont.render('Total Score: %s' % (getTotalScore()), True, WHITE)
    #updating score in leaderboard
    name=r"NOV"
    scoree= getTotalScore()
    print(scoree)
    update_leaderboards(name,scoree)
    totalscoreRect = totalscoreText.get_rect()
    totalscoreRect.midtop = (WINDOWWIDTH/2, 110)
    gameOverRect.midtop = (WINDOWWIDTH/2, 30)
    SCREEN.fill(BGCOLOR)
    SCREEN.blit(gameOverText, gameOverRect)
    SCREEN.blit(totalscoreText, totalscoreRect)
    j=200
    with open("Leaderboards.txt") as f:
        for line in f:
            boardFont = pygame.font.Font('freesansbold.ttf', 20)
```

```
            j=j+20
    drawPressKeyMsg()
    pygame.display.update()
    pygame.time.wait(1000)
    checkForKeyPress()

    while True:
        if checkForKeyPress():
            pygame.event.get()
            return

ef drawScore(score):
    scoreText = FONT.render('Score: %s' % (score), True, WHITE)
    scoreRect = scoreText.get_rect()
    scoreRect.center = (WINDOWWIDTH - 100, 30)
    SCREEN.blit(scoreText, scoreRect)

ef drawWorm(wormCoords):
    x = wormCoords[HEAD]['x'] * CELLSIZE
    y = wormCoords[HEAD]['y'] * CELLSIZE
    wormHeadRect = pygame.Rect(x, y, CELLSIZE, CELLSIZE)
    pygame.draw.rect(SCREEN, YELLOW, wormHeadRect)

    for coord in wormCoords[1:]:
        x = coord['x'] * CELLSIZE
        y = coord['y'] * CELLSIZE
        wormSegmentRect = pygame.Rect(x, y, CELLSIZE, CELLSIZE)
        pygame.draw.rect(SCREEN, GREEN, wormSegmentRect)

ef drawApple(coord):
    x = coord['x'] * CELLSIZE
```

```
def drawWorm(wormCoords):
    x = wormCoords[HEAD]['x'] * CELLSIZE
    y = wormCoords[HEAD]['y'] * CELLSIZE
    wormHeadRect = pygame.Rect(x, y, CELLSIZE, CELLSIZE)
    pygame.draw.rect(SCREEN, YELLOW, wormHeadRect)

    for coord in wormCoords[1:]:
        x = coord['x'] * CELLSIZE
        y = coord['y'] * CELLSIZE
        wormSe draw: MissingModule ct(x, y, CELLSIZE, CELLSIZE)
        pygame.draw.rect(SCREEN, GREEN, wormSegmentRect)

def drawApple(coord):
    x = coord['x'] * CELLSIZE
    y = coord['y'] * CELLSIZE
    appleRect = pygame.Rect(x, y, CELLSIZE, CELLSIZE)
    pygame.draw.rect(SCREEN, RED, appleRect)

def drawGrid():
    for x in range(0, WINDOWWIDTH, CELLSIZE):
        pygame.draw.line(SCREEN, DARKGRAY, (x, 0), (x, WINDOWHEIGHT))
    for y in range(0, WINDOWHEIGHT, CELLSIZE):
        pygame.draw.line(SCREEN, DARKGRAY, (0, y), (WINDOWWIDTH, y))

if __name__ == '__main__':
    main()
```

LEADERBOARD CODE

```
    elif direction == 'North':
        if last_pressed != 'up':
            last_pressed = 'up'
            pyautogui.press('up')
            print("Up Pressed")
            #pyautogui.PAUSE = 2

    elif direction == 'South':
        if last_pressed != 'down':
            pyautogui.press('down')
            last_pressed = 'down'
            print("Down Pressed")
            #pyautogui.PAUSE = 2
    cv2.putText(frame, direction, (20,40), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 3)
    cv2.imshow('Game Control Window', frame)
    key = cv2.waitKey(1) & 0xFF
    counter += 1
    if (flag == 0):

        pyautogui.click(int(width/2), int(height/2))
        flag = 1


    if(key == ord('q')):
        break

video_capture.release()
cv2.destroyAllWindows()
```

```
        # are no scores lower than yours
        if cur_index is None:

            # last_place essentially gets the number of entries thus far
            last_place = int(lb_lines[-1].split()[0].strip(')'))
            entry = "{}) {}\t{}\n".format((last_place+1), new_name, new_score)
            lb_lines.append(entry)
        else: # You've found a score you've beaten
            entry = "{}) {}\t{}\n".format(cur_place, new_name, new_score)
            lb_lines.insert(cur_index, entry)

            lb_lines_cp = list(lb_lines) # Make a copy for iterating over
            for line in lb_lines_cp[cur_index+1:]:
                if len(line.split()) < 2: continue
                position, entry_info = line.split(')', 1)
                new_entry_info = str(int(position)+1) + ')' + entry_info
                lb_lines[lb_lines.index(line)] = new_entry_info

    with open(lb_file, 'w') as lb_file_o:
        lb_file_o.writelines(lb_lines)


if __name__ == '__main__':
    name, score = extract_log_info()
    update_leaderboards(name, score)
```

GESTURE CODE:

game-control-using-object-tracking.py > ...

```
import cv2
import imutils
import numpy as np
from collections import deque
import time
import pyautogui

greenLower = (29, 86, 6)
greenUpper = (64, 255, 255)

buffer = 20
flag = 0
pts = deque(maxlen = buffer)
counter = 0
(dX, dY) = (0, 0)
direction = ''
last_pressed = ''


video_capture = cv2.VideoCapture(0)
time.sleep(2)
width,height = pyautogui.size()
while True:



    ret, frame = video_capture.read()
    frame = cv2.flip(frame,1)
    frame = imutils.resize(frame, width = 600)
```

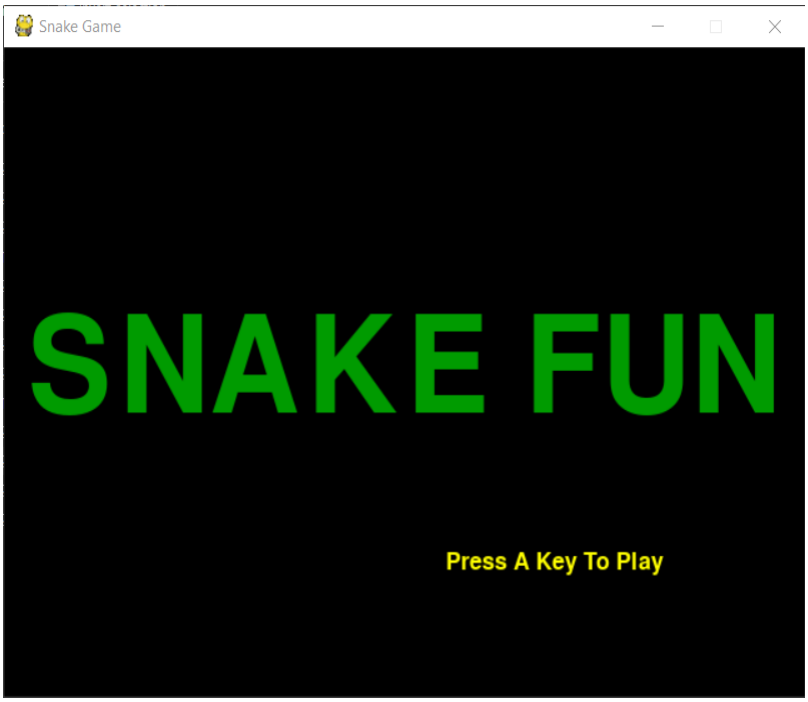
```
ret, frame = video_capture.read()
frame = cv2.flip(frame,1)
frame = imutils.resize(frame, width = 600)
blurred_frame = cv2.GaussianBlur(frame, (11,11), 0)
hsv_converted_frame = cv2.cvtColor(blurred_frame, cv2.COLOR_BGR2HSV)

mask = cv2.inRange(hsv_converted_frame, greenLower, greenUpper)

mask = cv2.erode(mask, None, iterations = 2)
mask = cv2.dilate(mask, None, iterations = 2)

cnts,_ = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
center = None


if(len(cnts) > 0):
    c = max(cnts, key = cv2.contourArea)
    ((x, y), radius) = cv2.minEnclosingCircle(c)

    M = cv2.moments(c)
    center = (int(M['m10'] / M['m00']), int(M['m01'] / M['m00']))


    if radius > 10:

        cv2.circle(frame, (int(x), int(y)), int(radius), (0,255,255), 2)
        cv2.circle(frame, center, 5, (0,255,255), -1)

        pts.appendleft(center)
```

```
    for i in np.arange(1, len(pts)):

        if(pts[i-1] == None or pts[i] == None):
            continue

        if counter >= 10 and i == 1 and pts[-10] is not None:

            dX = pts[-10][0] - pts[i][0]
            dY = pts[-10][1] - pts[i][1]
            (dirX, dirY) = ('', '')

            if np.abs(dX) > 50:
                dirX = 'West' if np.sign(dX) == 1 else 'East'

            if np.abs(dY) > 50:
                dirY = 'North' if np.sign(dY) == 1 else 'South'


            direction = dirX if dirX != '' else dirY


        thickness = int(np.sqrt(buffer / float(i + 1)) * 2.5)
        cv2.line(frame, pts[i - 1], pts[i], (0, 0, 255), thickness)


    if direction == 'East':
        if last_pressed != 'right':
            pyautogui.press('right')
            last_pressed = 'right'
            print("Right Pressed")
            #pyautogui.PAUSE = 2
```

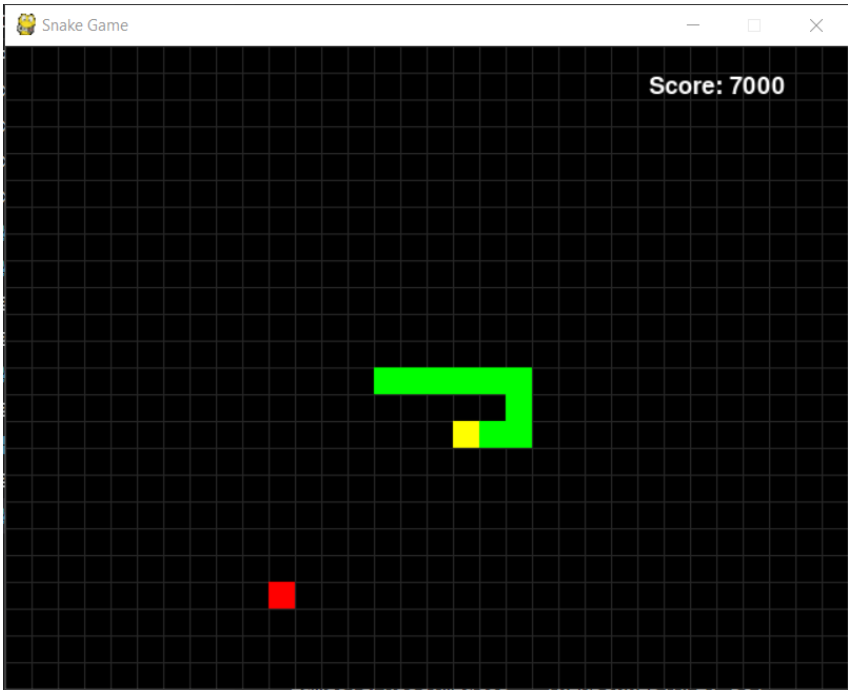
```
if direction == 'East':
    if last_pressed != 'right':
        pyautogui.press('right')
        last_pressed = 'right'
        print("Right Pressed")
        #pyautogui.PAUSE = 2

elif direction == 'West':
    if last_pressed != 'left':
        pyautogui.press('left')
        last_pressed = 'left'
        print("Left Pressed")
        #pyautogui.PAUSE = 2

elif direction == 'North':
    if last_pressed != 'up':
        last_pressed = 'up'
        pyautogui.press('up')
        print("Up Pressed")
        #pyautogui.PAUSE = 2

elif direction == 'South':
    if last_pressed != 'down':
        pyautogui.press('down')
        last_pressed = 'down'
        print("Down Pressed")
        #pyautogui.PAUSE = 2
cv2.putText(frame, direction, (20,40), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 3)
```

```
    elif direction == 'North':
        if last_pressed != 'up':
            last_pressed = 'up'
            pyautogui.press('up')
            print("Up Pressed")
            #pyautogui.PAUSE = 2

    elif direction == 'South':
        if last_pressed != 'down':
            pyautogui.press('down')
            last_pressed = 'down'
            print("Down Pressed")
            #pyautogui.PAUSE = 2
    cv2.putText(frame, direction, (20,40), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 3)
    cv2.imshow('Game Control Window', frame)
    key = cv2.waitKey(1) & 0xFF
    counter += 1
    if (flag == 0):

        pyautogui.click(int(width/2), int(height/2))
        flag = 1


    if(key == ord('q')):
        break

video_capture.release()
cv2.destroyAllWindows()
```

IMPLEMENTATION RESULTS



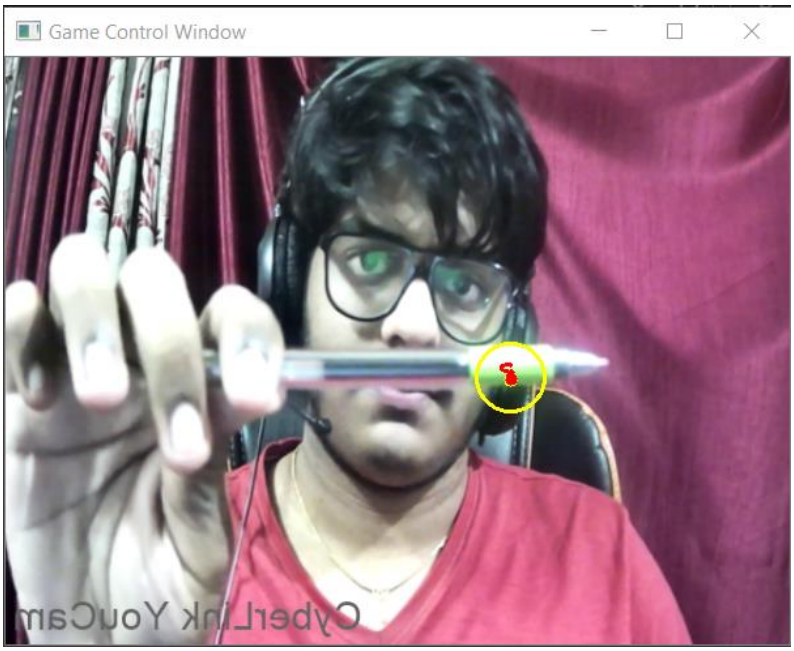
SNAKE GAME MAIN MENU SCREEN



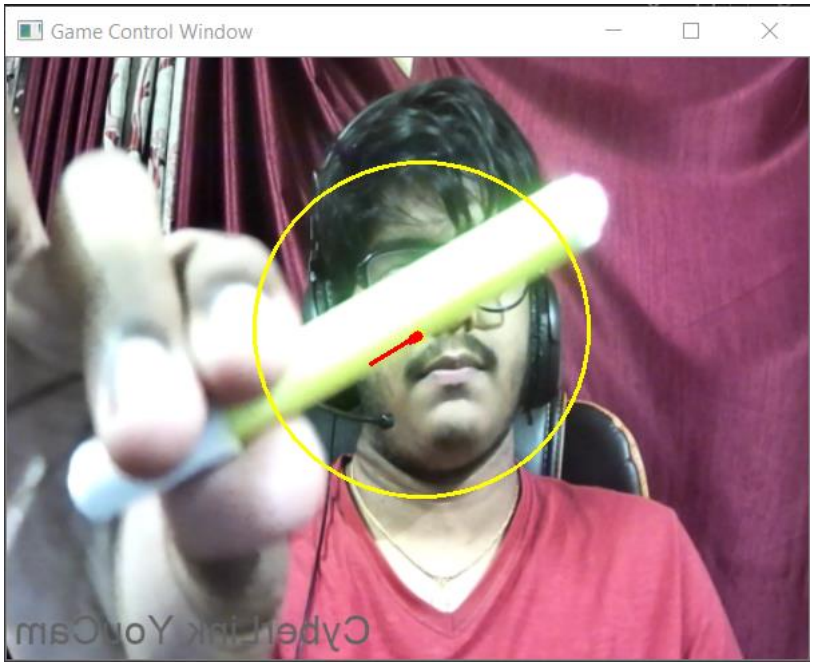
GAME PLAYING

FINAL SCORE AND LEADERBOARD

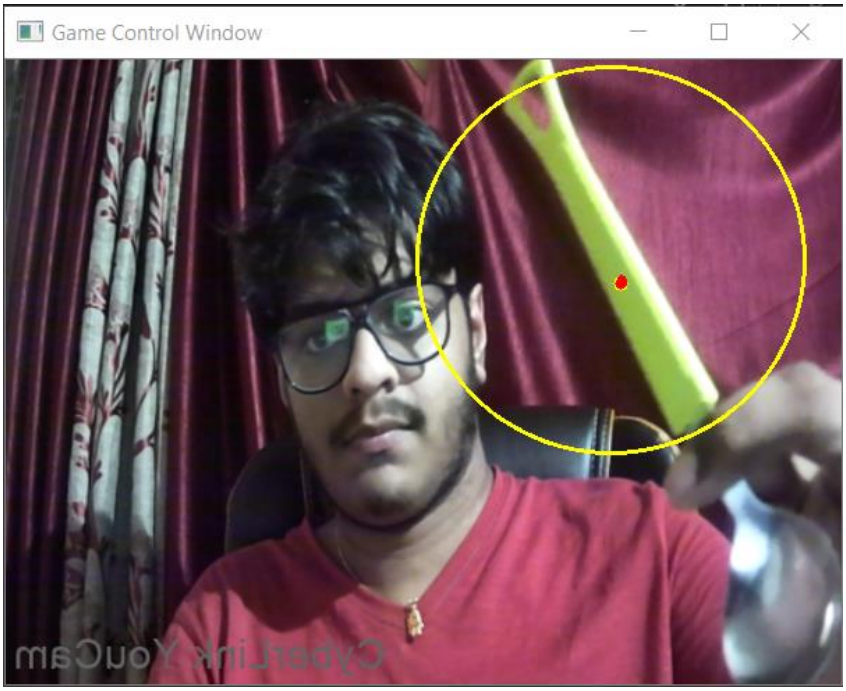
WEBCAM INTERFACE FOR CONTROLLING THE GAME AND SHOWING OBJECT DETECTION



FIRST POSITIVE DETECTION



SECOND POSITIVE DETECTION



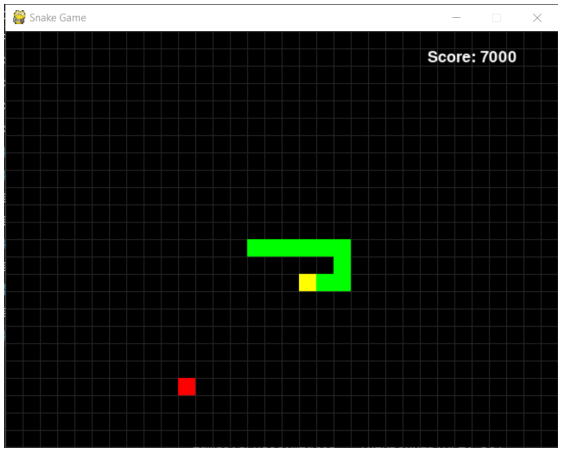
THIRD POSITIVE DETECTION

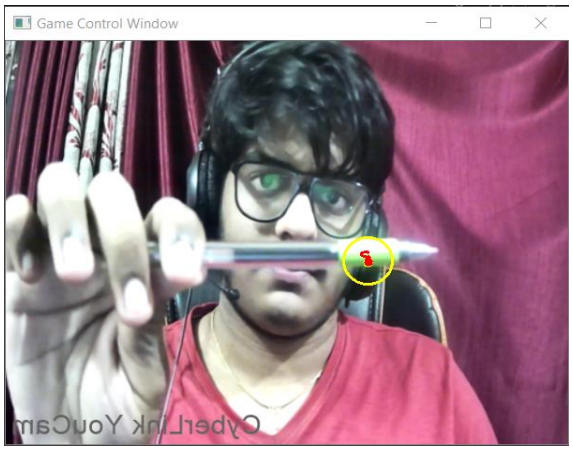
Above three pictures depict the efficiency of our gesture control script. The program can detect any green object in its vicinity no matter what shape or size. We can also choose the color to recognize.

INTERFACE VALIDATION USING HEURISTICS

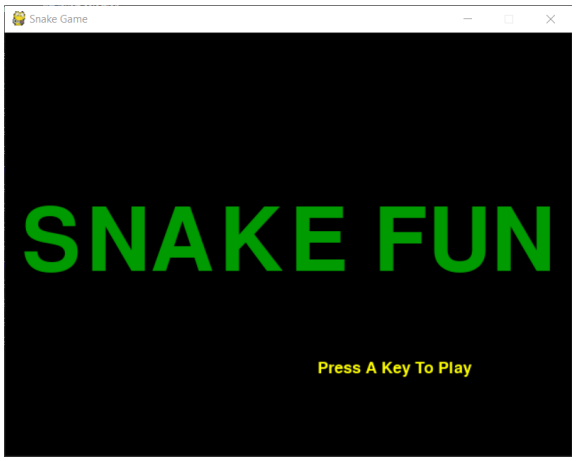
Heuristic evaluation is a process where experts use rules of thumb to measure the usability of user interfaces in independent walkthroughs and report issues. Evaluators use established heuristics like Nielsen-Molich and reveal insights that can help design teams enhance product usability from early in development. As experts, they go through a checklist of criteria to find flaws which design teams overlooked. While developing the application we tried following the heuristics stated by Neilsen in the following ways :

1. **VISIBILITY OF SYSTEM STATUS :** The system should always keep users informed about what is going on, through appropriate feedback within a reasonable time. For example, In our Snake Game, the Interface always keeps the player informed of what is going on by continuously showing the score he scored, and an increase in snake's length depicts that the snake has eaten an apple. For the Webcam Interface, the user has been informed which object is being detected for controlling the snake by showing the green circle around it along with a red dot for showing the centroid of the circle.



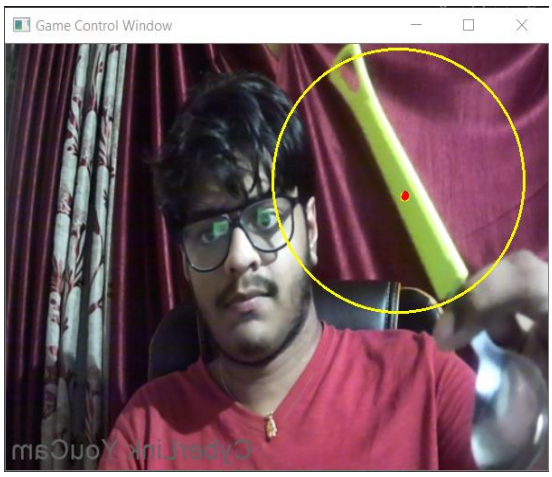


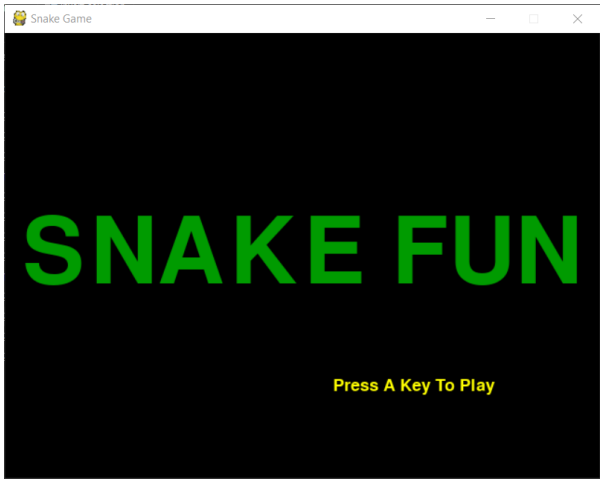
2. **MATCH BETWEEN SYSTEM AND THE REAL WORLD :** The System should speak the users' language, with words, phrases, and concepts familiar to the user, rather than system-oriented terms. Information should appear in a natural and logical order.order. For For example, the score or the leaderboard in our game application must be able to communicate with the user using a language that can be understandable to them.



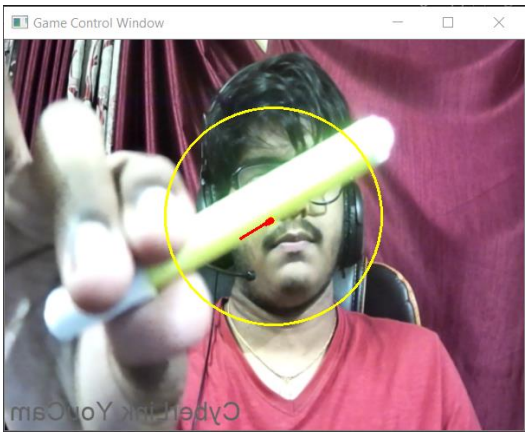
3. **USER CONTROL AND FREEDOM :** Users often choose system functions by mistake and will need a clearly marked emergency exit to leave the unwanted state without having to go through an extended dialogue. For example, In our Game Application's main menu the user is free to proceed and play the game by pressing A. Similarly after getting the leaderboard user can again play the game by pressing the same key 'A' and if the user doesn't want to play again he can simply close the Game application from the cross button present in the Top Right corner.

4. **CONSISTENCY AND STANDARDS :** Users should not have to wonder whether different words, situations, or actions mean the same thing. For example, playing the Game key is 'A' across both the main menu and leaderboard menu. Whenever we show the green circle around the object its in same format on both the pages

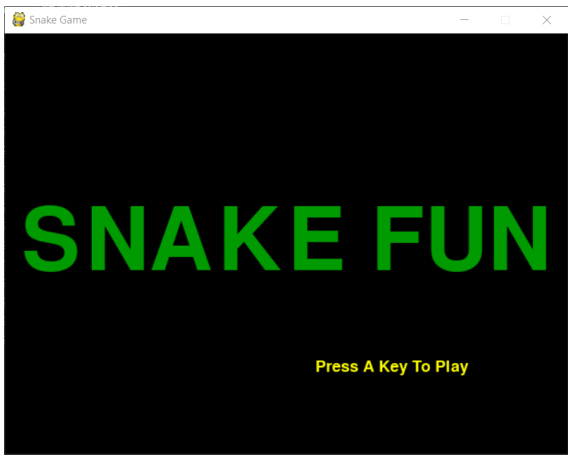




5. **ERROR PREVENTION :** Even better than good error messages is a careful design which prevents a problem from occurring in the first place. For example, Gamers, just like interface users, can get distracted and make errors. As designers, we must help players prevent unconscious errors. In our Webcam interface when the green colour object is not detected by the webcam it won't show the yellow circle.



6. **RECOGNITION RATHER THAN RECALL :** Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. For example, in the game application window, The user needs not to remember which key is needed to play the game. It's clearly visible that the 'A' key is mentioned whenever there is an option to press it.



7. **FLEXIBILITY AND EFFICIENCY OF USE :** The Interface should be flexible transforming itself between a novice user and an advanced user. For example, if a novice user is using it he can just simply follow the demonstrated steps and play the game. If an advanced user is using he can go through the source file and can change the color of the object to be detected or can add any audio sounds to the game.

8. **AESTHETIC AND MINIMALIST DESIGN :** Dialogues should not contain information which is irrelevant or rarely needed. For example, In our game application window, there are no unnecessary dialogues are there only necessary things like score, the leaderboard has been displayed.

9. **HELP USERS RECOGNIZE, DIAGNOSE AND RECOVER FROM ERRORS:** Error messages should be expressed in plain language, precisely indicate the problem, and constructively suggest a solution. For example, In the game application and webcam interface there is no such error message.

10. **HELP AND DOCUMENTATION :** A great user interface lets the user navigate through its features without any documentation or training. But if there is any user who could not make it out, adequate help should be provided within the product. For example, In our Project user can go through the source codes so that he can add some additional features to it like a game audio track, the color of the object which is to be detected for playing the game.

CONCLUSION AND FUTURE SCOPE

This project may be initially made in python using free modules like OpenCV and pyautogui. Once the project is functional, future business aspects include making the project into a user-friendly software with proper UI where people can define their gestures and key bindings with few button clicks. Since the gaming industry is on a rise in India in the form of esports, this project on its own can improve the current status of motion gestures in gaming. For e.g., if we are to consider a PS4 console, there is a separate stick for detecting motion sensors but with this technology or program, the motion in the gaming will be detected without such support of the separate sticks. This project improves the user experience aspect by letting the users play simple games in a fun and intuitive manner, thus enhancing the gaming experience of the user and making even the simplest of games more interactive. There are many ways to achieve this like using AR and VR but those options are not often affordable by the general public. Our project just uses the webcam which makes it far more feasible.

CITATION TABLE

S.NO	PAPER NO.	CITATION
1	[P1-2014]	Chowdary, P. Raghu Veera, M. Nagendra Babu, Thadigotla Venkata Subbareddy, Bommepalli Madhava Reddy, and V. Elamaran. "Image processing algorithms for gesture recognition using MATLAB." In *2014 IEEE international conference on advanced communications, control and computing technologies*, pp. 1511-1514. IEEE, 2014.
2	[P1-2016]	More, Sagar P., and Abdul Sattar. "Hand gesture recognition system using image processing." In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pp. 671-675. IEEE, 2016.
3	[P2-2017]	Barba-Guamán, Luis, Carlos Calderon-Cordova, and Pablo Alejandro Quezada-Sarmiento. "Detection of moving objects through color thresholding." In *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1-6. IEEE, 2017.
4	[P4-2013]	Shrivastava, Rajat. "A hidden Markov model based dynamic hand gesture recognition system using OpenCV." In *2013 3rd IEEE International Advance Computing Conference (IACC)*, pp. 947-950. IEEE, 2013.
5	[P5-2019]	Sharma, Shubham, Saurabh Mishra, Nachiket Deodhar, Akshay Katageri, and Parth Sagar. "Solving The Classic Snake Game Using AI." In *2019 IEEE Pune Section International Conference (PuneCon)*, pp. 1-4. IEEE, 2019.
6	[P6-2020]	Brdjanin, Adnan, Nadja Dardagan, Dzemil Dzigal, and Amila Akagic. "Single Object Trackers in OpenCV: A Benchmark." In *2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, pp. 1-6. IEEE, 2020.
7	[P7-2020]	Golovanov, Roman, Dmitry Vorotnev, and Darina Kalina. "Combining Hand Detection and Gesture Recognition Algorithms for Minimizing Computational Cost." In *2020 22th International Conference on Digital Signal Processing and its Applications (DSPA)*, pp. 1-4. IEEE, 2020.
8	[P8-2020]	Balamurugan, P., J. Santhosh, and G. Arulkumaran. "HAND MOTION BASED MOUSE CURSOR CONTROL USING IMAGE PROCESSING TECHNIQUE." *Journal of Critical Reviews* 7, no. 4 (2020): 181-185.
9	[P9-2020]	Golec, Pawel, Wieslawa Gryncewicz, Krzysztof Hauke, Marcin Hernes, and Artur Rot. "Gesture Detection in Digital Image Processing based on the Use of Convolutional Neuronal Networks." In *2020 10th International Conference on Advanced Computer Information Technologies (ACIT)*, pp. 430-435. IEEE, 2020.
10	[P10-2020]	Muratov, Yevgeniy, Mikhael Nikiforov, and Olga Melnik. "Hand Gesture Recognition for Non-Contact Control of a Technical System." In *2020 International Russian Automation Conference (RusAutoCon)*, pp. 1107-1111. IEEE, 2020.

REFERENCES

[P1]
https://ieeexplore.ieee.org/document/7019356

[P2]
https://ieeexplore.ieee.org/document/7754766?anchor=authors

[P3]
https://ieeexplore.ieee.org/abstract/document/7975755

[P4]
https://ieeexplore.ieee.org/abstract/document/6514354

[P5]
http://ieeexplore.ieee.org.egateway.vit.ac.in/document/9105796?arnumber=9105796

[P6]
http://ieeexplore.ieee.org.egateway.vit.ac.in/document/9194647?arnumber=9194647

[P7]
http://ieeexplore.ieee.org.egateway.vit.ac.in/document/9213273?arnumber=9213273

[P8]
https://www.scopus.com.egateway.vit.ac.in/record/display.uri?eid=2-s2.0-85082338076&origin=inward&txGid=b43ae80269743a630ac6dba5767883b3

[P9]
http://ieeexplore.ieee.org.egateway.vit.ac.in/document/9208860?arnumber=9208860

[P10]
http://ieeexplore.ieee.org.egateway.vit.ac.in/document/9208182?arnumber=9208182

https://stackoverflow.com/questions/48648597/how-to-use-batch-file-to-run-multiple-python-scripts-simultaneously

https://www.101computing.net/wp/wp-content/uploads/leaderboard.txt

https://stackoverflow.com/questions/40704323/valueerror-expected-2-got-1-python

https://stackoverflow.com/questions/42014195/rendering-text-with-multiple-lines-in-pygame

https://stackoverflow.com/questions/16726354/saving-the-highscore-for-a-python-game

APPENDIX

1. Link To PPT: https://drive.google.com/drive/folders/1rd-ulqfoWsUoLC2vpNiVzYOQQI2SnM7w?usp=sharing

2. Link To Pre-Recorded Demonstration Video:

https://drive.google.com/drive/folders/1rYE03i008R3q4X3DsGk14j7E0Cl6F1U8?usp=sharing

3. Link To Access Source Code File And Files Containing Steps to Execute the Project

https://drive.google.com/drive/folders/1Nzh03p1YHABeRZq_oenB1KAiOaKC2rSo?usp=sharing