GAME INTERFACE USING

HAND GESTURE RECOGNITION

by

SRI HARSHAVARDHAN PALLA 20BCE1308
RUKKSANA A 20BCE1968
SIDDHARTH M 20BPS1007

A project report submitted to

Dr. Ganala Santoshi

in partial fulfilment of the requirements for the course of

CSE4015 –HUMAN COMPUTER AND INTERACTION

in

B. Tech. COMPUTER SCEINCE & ENGINEERING

Vandalur – Kelambakkam Road

Chennai – 600127

ABSTRACT

It is an undeniable fact that gaming is a very popular source of entertainment for kids and even adults. Video gaming works as a tool to attract players from variety of ages. So, to make it more modern in nature we will design the game in such a way that the player will have no need to interact with device or the computer in order to play it the game will be designed in such a way that the hand movements made by the player will be read and noted with respect to the interface. The computer will then scan the hand of the player and sense the movement that is made by him and processes the respective action which will in turn contribute to his progress in the game and will be completely devoid of keys or buttons.

This feature will ease the playing of the game and will make it more interesting and lively. This will avoid problems like sticky keys and other keyboard related issues. This project in a sense is more inclusive and user friendly as it can be easily played by differently able persons and people with numbness. Gesture recognition system that is used here consists of four major steps that is hand shape modelling, extracting hand related parameters like wrist or palm determination of finger and fingertips which will be followed by their classification.

INTRODUCTION

Computer technology has tremendously grown over the past decade and has become a necessary part of everyday live. The primary computer accessory for Human Computer Interaction (HCI) is the keyboard. The keyboard is not suitable for HCI in some real life situations, such as with Human Robot Interaction (HRI). The most natural and intuitive technique for HCI, that is a viable replacement for the computer keyboard is with the use of hand gestures.

Digital video games can be enjoyed more naturally and conveniently using hand gestures and machine learning in-corporate. The game can be controlled using specific hand gestures. This project aims to create a unique application that connects to the game. Input and interaction devices are an important part of every game for human interaction but in recent times it has been developed with various technologies to control computer-based systems. Interaction between humans comes from different modes like gestures, speech, text etc. All gesture

interaction is best for interacting with games. Furthermore, the approach of stochastic gradient descent hand gesture recognition system can efficiently track both static and dynamic hand gestures.

EXISTING SYSTEM

Today peoples are using high end gaming consoles for playing motion sensing games. The existing systems include popular gaming consoles like WII, Nintendo, PS1, PS2, PS3, and XBOX 360. Systems like WII or Play station Move which both use motion technologies. But these technologies did not have as much functionality that could be used whereas with this you track many different elements with no need for a hand held controller.

Limitations of existing:-

1. Console is required for playing games.
2. The existing system can only be used for playing games.
3. Light emitting sensors or joysticks are required in existing systems such as ps3 Nintendo, WII.
4. Most of the games are compatible only with particular gaming console manufacturers.
5. These motion sensing games are not compatible with PCs or laptops

PROBLEM STATEMENT

The computer will then scan the hand of the player and sense the movement that is made by him/her and processes the respective action which will in turn contribute to his progress in the game.

This feature will easen the playing of the game ,will make it more interesting and it also makes a person move which contributes in exercising.

This project in a sense is more inclusive and user friendly as it can be easily played by differently able persons and people with numbness. It promote controller free gaming as this user will not need have games controller like keyboard, mouse or joysticks. To offer experience of motion gaming for PC games without need of expensive gaming console and additional game controllers.

INTRODUCTION TO PROPOSED MODEL

The proposed model here deals with the motion sensing pertaining to the hand movements of the player and involves capturing of various types of hand movements and channeling them into various actions contained in the game.

Here if we look at the 3 games given we can observe that each of them works fundamentally through incorporating this particular idea to completely avoid the physical contact in entirety confining oneself to limited hand movements translating into correlated gaming advancements

PROPOSED SOLUTION

The proposed solution for all the 3 games has a common thread but varies in implementation of them.

The commonality involved in them is how with help of the Webcam the gaming proceeds with complete absence of physical contact and if we take different examples like battlecity, it is based on opening and closing of fist I.e it involves firing by ships if the fist is open and stoppage when it's closed.

In Dino run we have two quadrants where if the fist is open on top the player icon runs and in the lower quadrant the same gesture is reflected by the ducking of him

In Mario we have the different vertical quadrants and if the fist is opened in each of the left and right quadrant the playing icon runs towards each side correspondingly.

COMPONENTS OF THE MODEL

The components used in our model are:-

Hardware:-

The laptop or the PC in which the games are played should be equipped with a webcam to detect the hand movement.

Software:-

1. Pycharm ide
2. Python3
3. Opencv2
4. Tensorflow
5. Numpy
6. Pygame
7. Gym_extensions

DESCRIPTION OF COMPONENTS

Hardware:-

1. Webcam:- A webcam is a video camera which is designed to record or stream to a computer or computer network. They are primarily used in videotelephony, livestreaming and social media, and security. Webcams can be built-in computer hardware or peripheral devices, and are commonly connected to a device using USB or wireless protocols.

Software:-

1. Pycharm:- PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly

integrated to create a convenient environment for productive Python, web, and data science development.

2. Python3:- Python 3 is a newer version of the Python programming language which was released in December 2008. This version was mainly released to fix problems that exist in Python 2. The nature of these changes is such that Python 3 was incompatible with Python 2. It is backward incompatible.

3. Opencv2:- 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. The library is cross-platform and free for use under the open-source Apache 2 License.

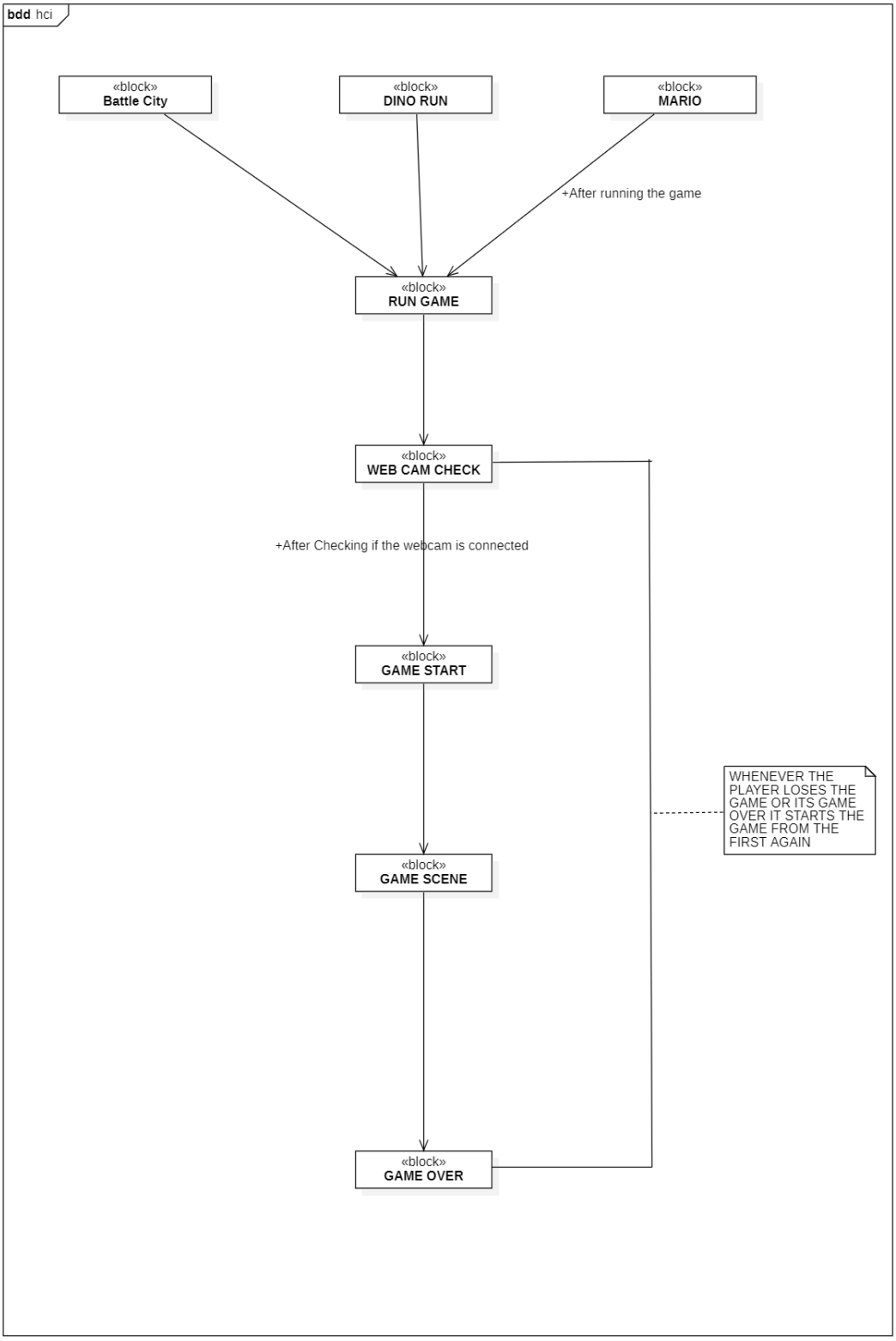
4. Tesorflow:- TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

5. Numpy:- NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

6. Pygame:- Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language.

7. Gym_extensions:- Gym is an open source Python library for developing and comparing reinforcement learning algorithms by providing a standard API to communicate between learning algorithms and environments, as well as a standard set of environments compliant with that API. Since its release, Gym's API has become the field standard for doing this.

BLOCK DIAGRAM



WORKING MECHANISM

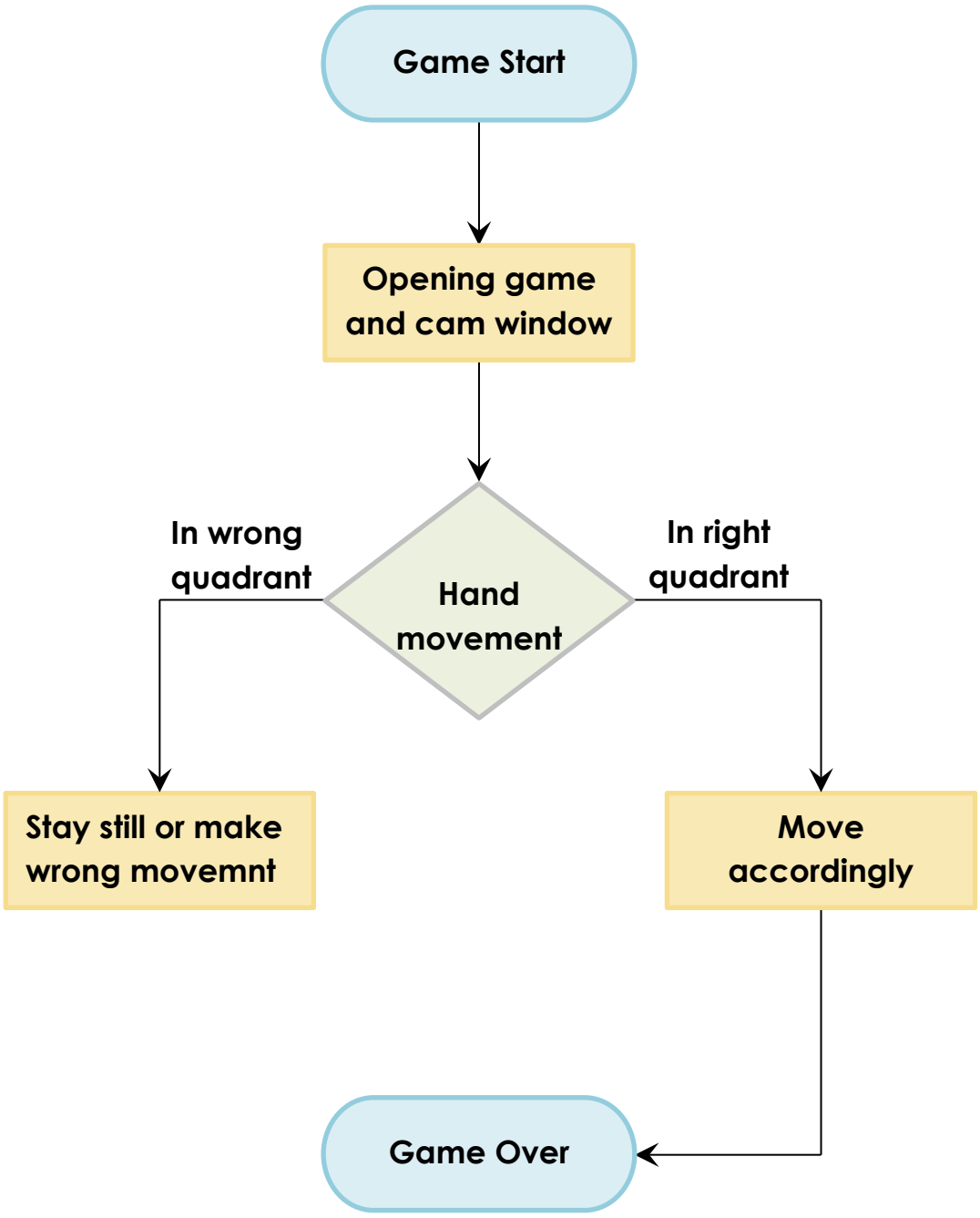
When the game is run using the pycharm terminal two new windows are opened one in which the game will be run and the other window to scan the hand movement of the player and take that hand movement as the input as the game input. The following are the different inputs for different games.

1. Battlecity:-The cam window screen for the battle city game is divided into five parts one circle in the middle and 4 triangles surrounding it the battle ship moves according to the position of the hand, if its placed in the upper triangle it moves up, lower triangle it moves and left and right if its place in the left and right triangle. The battle ship fires if the player hand is open and doesn't if the player wrist is closed.

2. Dino run:- The cam window is divided into 2 parts if the player hand is placed in the upper quadrant with is wrist open the dino jumps and if its in the lower quadrant with fist open it duck.

3. Mario:-The cam window is divided into three parts when hand is in the centre partition mario stands still and if the and is in the right quadrant with open fist he jumps to the right and same applies to the left quadrant.

FLOW DIAGRAM



CODE

Battlecity.py:-

```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
import cv2
import numpy as np
import multiprocessing as _mp
from src.utils import load_graph, detect_hands, predict, is_in_triangle
from src.battle_city_utils import battle_city
from src.config import RED, CYAN, YELLOW, BLUE, GREEN

##flags=tf.compat.v1.flags
tf.flags.DEFINE_integer("width", 640, "Screen width")
tf.flags.DEFINE_integer("height", 480, "Screen height")
tf.flags.DEFINE_float("threshold", 0.6, "Threshold for score")
tf.flags.DEFINE_float("alpha", 0.2, "Transparent level")
tf.flags.DEFINE_string("pre_trained_model_path", "src/pretrained_model.pb", "Path to pre-
trained model")

FLAGS = tf.flags.FLAGS


def main():
    graph, sess = load_graph(FLAGS.pre_trained_model_path)
    cap = cv2.VideoCapture(0)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, FLAGS.width)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, FLAGS.height)
    mp = _mp.get_context("spawn")
    v = mp.Value('i', 0)
    lock = mp.Lock()
    process = mp.Process(target=battle_city, args=(v, lock))
    process.start()
    x_center = int(FLAGS.width / 2)
    y_center = int(FLAGS.height / 2)
    radius = int(min(FLAGS.width, FLAGS.height) / 6)
    while True:
        key = cv2.waitKey(10)
        if key == ord("q"):
            break
        _, frame = cap.read()
        frame = cv2.flip(frame, 1)
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

```
        boxes, scores, classes = detect_hands(frame, graph, sess)
        frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
        results = predict(boxes, scores, classes, FLAGS.threshold, FLAGS.width,
FLAGS.height)
        if len(results) == 1:
            x_min, x_max, y_min, y_max, category = results[0]
            x = int((x_min + x_max) / 2)
            y = int((y_min + y_max) / 2)
            cv2.circle(frame, (x, y), 5, RED, -1)
            if category == "Closed" and np.linalg.norm((x - x_center, y - y_center)) <= radius:
                action = 0 # Stay
                text = "Stay"
            elif category == "Closed" and is_in_triangle((x, y), [(0, 0), (FLAGS.width, 0),
                                                    (x_center, y_center)]):
                action = 1  # Up
                text = "Up"
            elif category == "Closed" and is_in_triangle((x, y), [(0, FLAGS.height),
                                                    (FLAGS.width, FLAGS.height), (x_center,
y_center)]):
                action = 2  # Down
                text = "Down"
            elif category == "Closed" and is_in_triangle((x, y), [(0, 0),
                                                    (0, FLAGS.height),
                                                    (x_center, y_center)]):
                action = 3  # Left
                text = "Left"
            elif category == "Closed" and is_in_triangle((x, y), [(FLAGS.width, 0),
(FLAGS.width, FLAGS.height),
                                                    (x_center, y_center)]):
                action = 4  # Right
                text = "Right"
            elif category == "Open":
                action = 5  # Fire
                text = "Fire"
            else:
                action = 0
                text = "Stay"
            with lock:
                v.value = action
            cv2.putText(frame, "{}".format(text), (x_min, y_min - 5),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, GREEN, 2)

        overlay = frame.copy()
        cv2.drawContours(overlay, [np.array([(0, 0), (FLAGS.width, 0), (x_center, y_center)])],
0,
                         CYAN, -1)
        cv2.drawContours(overlay, [
```

```
            np.array([(0, FLAGS.height), (FLAGS.width, FLAGS.height), (x_center,
y_center)])], 0,
                        CYAN, -1)
        cv2.drawContours(overlay, [
            np.array([(0, 0), (0, FLAGS.height), (x_center, y_center)])], 0,
                        YELLOW, -1)
        cv2.drawContours(overlay, [np.array([(FLAGS.width, 0), (FLAGS.width,
FLAGS.height), (x_center, y_center)])], 0,
                        YELLOW, -1)
        cv2.circle(overlay, (x_center, y_center), radius, BLUE, -1)
        cv2.addWeighted(overlay, FLAGS.alpha, frame, 1 - FLAGS.alpha, 0, frame)

        cv2.imshow('Detection', frame)

    cap.release()
    cv2.destroyAllWindows()


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

Dino.py:-

```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
import cv2
import multiprocessing as _mp
from src.utils import load_graph, dinosaur, detect_hands, predict
from src.config import RED, GREEN, YELLOW


tf.flags.DEFINE_integer("width", 640, "Screen width")
tf.flags.DEFINE_integer("height", 480, "Screen height")
tf.flags.DEFINE_float("threshold", 0.6, "Threshold for score")
tf.flags.DEFINE_float("alpha", 0.3, "Transparent level")
tf.flags.DEFINE_string("pre_trained_model_path", "src/pretrained_model.pb", "Path to pre-
trained model")


FLAGS = tf.flags.FLAGS
```

```
def main():
    graph, sess = load_graph(FLAGS.pre_trained_model_path)
    cap = cv2.VideoCapture(0)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, FLAGS.width)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, FLAGS.height)
    mp = _mp.get_context("spawn")
    v = mp.Value('i', 0)
    lock = mp.Lock()
    process = mp.Process(target=dinosaur, args=(v, lock))
    process.start()
    while True:
        key = cv2.waitKey(10)
        if key == ord("q"):
            break
        _, frame = cap.read()
        frame = cv2.flip(frame, 1)
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        boxes, scores, classes = detect_hands(frame, graph, sess)
        frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
        results = predict(boxes, scores, classes, FLAGS.threshold, FLAGS.width,
FLAGS.height)

        if len(results) == 1:
            x_min, x_max, y_min, y_max, category = results[0]
            x = int((x_min + x_max) / 2)
            y = int((y_min + y_max) / 2)
            cv2.circle(frame, (x, y), 5, RED, -1)
            if category == "Closed":
                action = 0  # Do nothing
                text = "Run"
```

```
            elif category == "Open" and y < FLAGS.height/2:
                action = 1 # Jump
                text = "Jump"
            elif category == "Open" and y > FLAGS.height/2:
                action = 2
                text = "Duck"
            else:
                action = 0
                text = "Run"

            with lock:
                v.value = action
            cv2.putText(frame, "{}".format(text), (x_min, y_min - 5),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, GREEN, 2)
        overlay = frame.copy()
        cv2.rectangle(overlay, (0, 0), (FLAGS.width, int(FLAGS.height / 2)), YELLOW, -1)
        cv2.addWeighted(overlay, FLAGS.alpha, frame, 1 - FLAGS.alpha, 0, frame)
        cv2.imshow('Detection', frame)

    cap.release()
    cv2.destroyAllWindows()


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

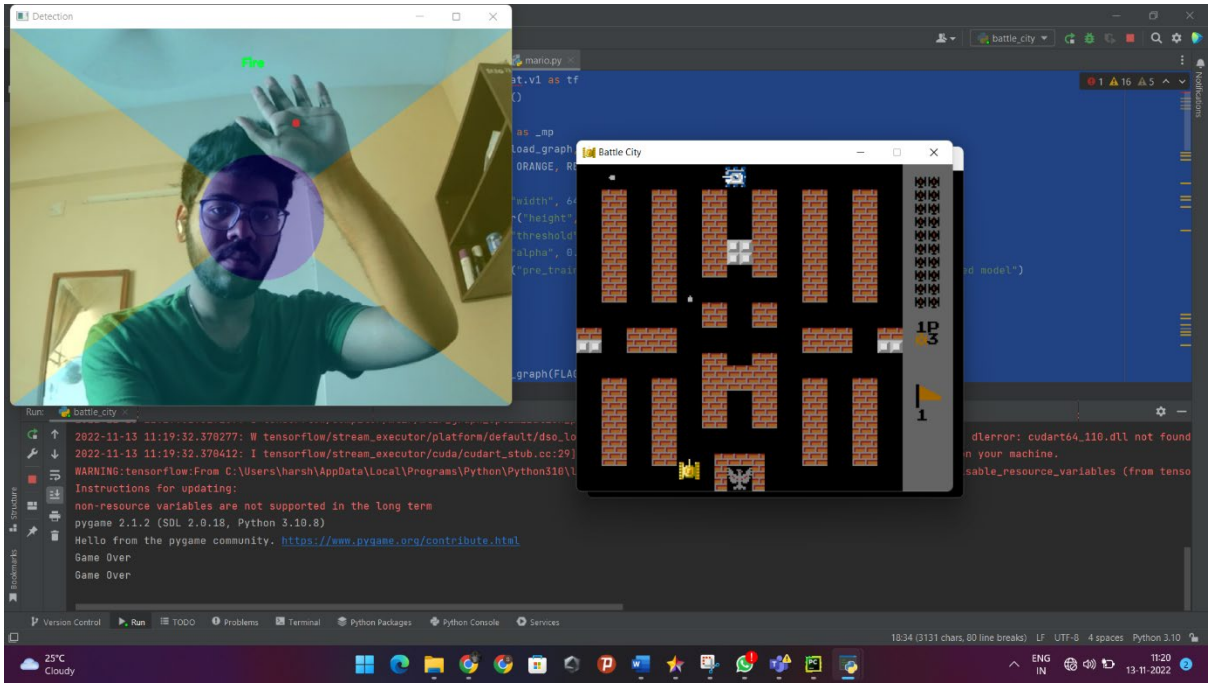
Mario.py:-

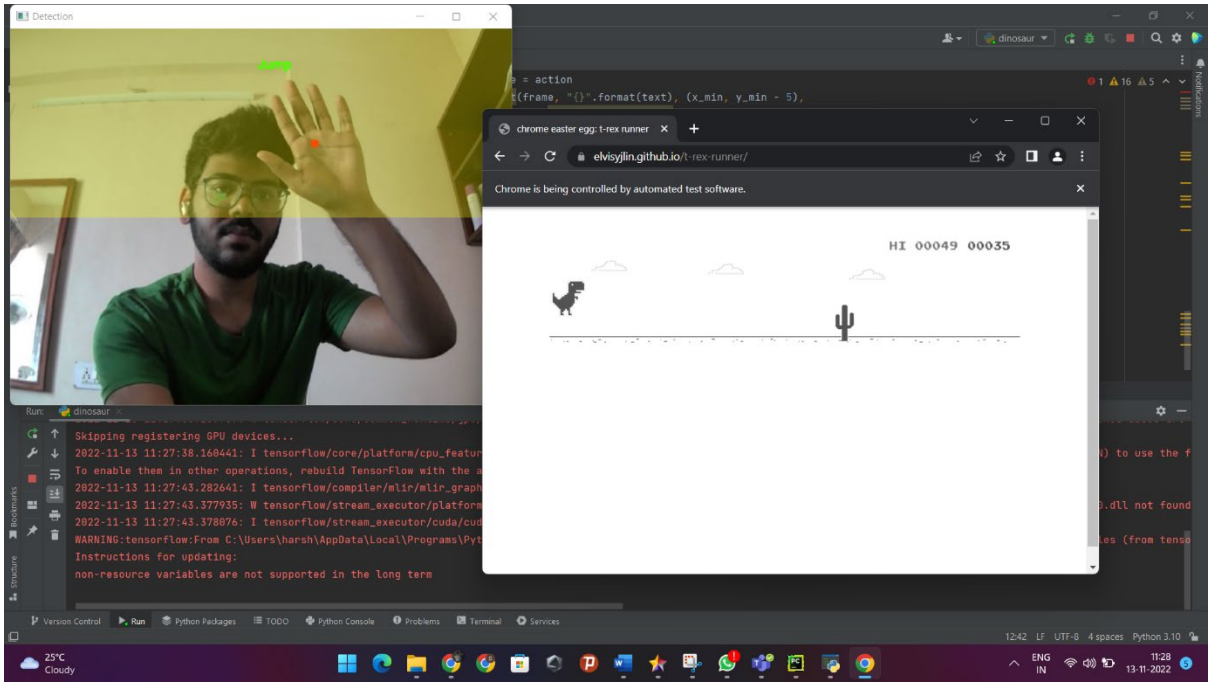
```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
import cv2
import multiprocessing as _mp
from src.utils import load_graph, mario, detect_hands, predict
from src.config import ORANGE, RED, GREEN

tf.flags.DEFINE_float("width", 640, "Screen width")
tf.flags.DEFINE_integer("height", 480, "Screen height")
tf.flags.DEFINE_float("threshold", 0.6, "Threshold for score")
tf.flags.DEFINE_float("alpha", 0.3, "Transparent level")
tf.flags.DEFINE_string("pre_trained_model_path", "src/pretrained_model.pb", "Path to pre-
trained model")

FLAGS = tf.flags.FLAGS


def main():
    graph, sess = load_graph(FLAGS.pre_trained_model_path)
    cap = cv2.VideoCapture(0)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, FLAGS.width)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, FLAGS.height)
    mp = _mp.get_context("spawn")
    v = mp.Value('i', 0)
    lock = mp.Lock()
    process = mp.Process(target=mario, args=(v, lock))
    process.start()
    while True:
        key = cv2.waitKey(10)
        if key == ord("q"):
```

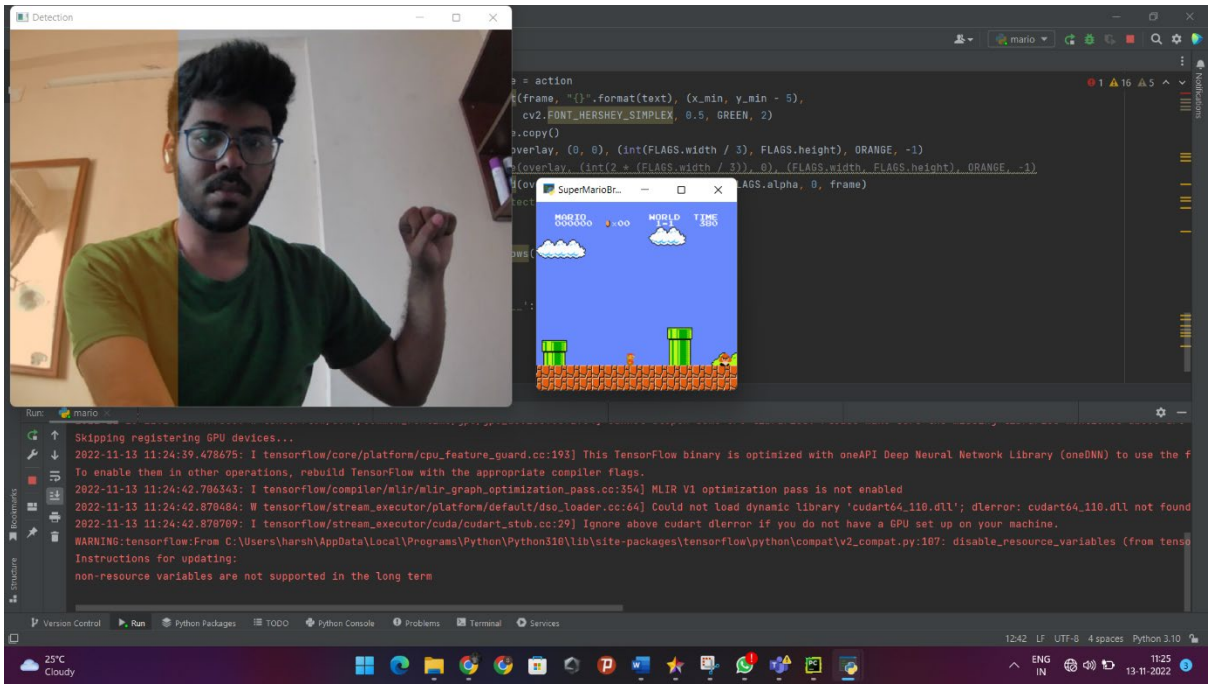
```
            break
        _, frame = cap.read()
        frame = cv2.flip(frame, 1)
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        boxes, scores, classes = detect_hands(frame, graph, sess)
        frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
        results = predict(boxes, scores, classes, FLAGS.threshold, FLAGS.width,
FLAGS.height)

        if len(results) == 1:
            x_min, x_max, y_min, y_max, category = results[0]
            x = int((x_min + x_max) / 2)
            y = int((y_min + y_max) / 2)
            cv2.circle(frame, (x, y), 5, RED, -1)

            if category == "Open" and x <= FLAGS.width / 3:
                action = 7  # Left jump
                text = "Jump left"
            elif category == "Closed" and x <= FLAGS.width / 3:
                action = 6  # Left
                text = "Run left"
            elif category == "Open" and FLAGS.width / 3 < x <= 2 * FLAGS.width / 3:
                action = 5  # Jump
                text = "Jump"
            elif category == "Closed" and FLAGS.width / 3 < x <= 2 * FLAGS.width / 3:
                action = 0  # Do nothing
                text = "Stay"
            elif category == "Open" and x > 2 * FLAGS.width / 3:
                action = 2  # Right jump
                text = "Jump right"
            elif category == "Closed" and x > 2 * FLAGS.width / 3:
                action = 1  # Right
```

```
            text = "Run right"
        else:
            action = 0
            text = "Stay"
        with lock:
            v.value = action
        cv2.putText(frame, "{}".format(text), (x_min, y_min - 5),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, GREEN, 2)
    overlay = frame.copy()
    cv2.rectangle(overlay, (0, 0), (int(FLAGS.width / 3), FLAGS.height), ORANGE, -1)
    ##cv2.rectangle(overlay, (int(2 * (FLAGS.width / 3)), 0), (FLAGS.width,
FLAGS.height), ORANGE, -1)
    cv2.addWeighted(overlay, FLAGS.alpha, frame, 1 - FLAGS.alpha, 0, frame)
    cv2.imshow('Detection', frame)

  cap.release()
  cv2.destroyAllWindows()


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

OUTPUT SCREENSHOTS







CONCLUSION

The proposed work will help to eliminate the traditionally completely. It only require web-camera to capture I/P image. This would lead to a new generation of human computer interaction in which no physical contact with device is needed. Anyone can use the system to operate the computer easily, by using gesture command. We have chosen to use the basic games like battlecity, mario and chrome dino in our project which are a great fun to play and adding these hand gestures to it will help to provide a better gameplay experience and also helps in providing some physical activity of the player.

REFERENCES

https://ieeexplore.ieee.org/document/5711226

https://www.geeksforgeeks.org/project-idea-games-/

https://www.researchgate.net/publication/342540793_HANDREHA_dynamic_hand_gesture_recognition_for_game-based_wrist_rehabilitation

https://www.researchgate.net/figure/game-controlling