**GAME-N-GEEK**

**INTERACTIVE GAMING PORTAL**

**PROJECT REPORT**

**Submitted by:**

| SI NO | REG.NO | NAME |
|---|---|---|
| 1 | 20BCE2515 | Parth Borana |
| **2** | 20BCE2537 | Vadarevu Sai Sampreet Khushal |
| **3** | 20BCE2772 | Saimrika Gupta |
| **4** | 20BCE2806 | Ashmit Gupta |

**PREPARED FOR:**

HUMAN COMPUTER INTERACTION (CSE 4015)

PROJECT COMPONENT

**UNDER THE GUIDANCE OF:**

Prof. Swarnalatha P.

Professor Grade 1

School of computer Science and Engineering

**PROBLEM STATEMENT/ ABSTRACT:**

Most of the population, especially, elderly people are interested in computer gaming technology and desire to be actively involved in such activities, however they are limited by theirresponsiveness due to age, as gaming requires fast reflexes in order to provide and comfortable / satisfactory experience. People who are not very knowledgeable about gaming are also limited toenter the gaming world. This upbrings a huge demand in the current market for a software / product which can enable users who are new to the gaming world to be able to comfortably enter / transition into the gaming world or be able to enjoy the merits of gaming without their limitations stopping them.

**AIM AND SCOPE:**

To provide gamers and people of interest in the gaming field with a unique and mesmerizingexperience which can bring a shift toward human movement computer gaming.

To provide a software / platform for real time visual interaction with the system to ensure thecomplete gaming experience.

To introduce people with limited knowledge of gaming to the gaming world and to ensure a comfortable experience as well as a smooth transition into the gaming technology.

## INTRODUCTION:

Game-N-Geek allows for interactive gaming experience which uses tracker to be able to interactin gaming environments using human movements.

Game-N-Geek appears to be a platform that utilizes tracking technology to allow for interactive gaming experiences that incorporate human movements. This technology can provide a more immersive and engaging gaming experience by allowing players to physically interact with the game environment using their body movements.
There are several types of tracking technology that can be used for this purpose, including motion capture systems, camera-based tracking, and wearable sensors. These technologies can detect and interpret a player's movements, allowing them to control their in-game avatar in real time.
Overall, the use of tracking technology in gaming has the potential to revolutionize the way we play and experience games. By incorporating human movements into the gameplay, we can create more immersive and interactive experiences that are both fun and engaging.

Game-N-Geek is a program that allows users to play any PC related video game using their hand signs and movements instead of using mouse and keyboard. It allows users to play games using hand signs.

Game-N-Geek allows for a smooth gaming experience without the inclusion of hands and can be used to interact within the gaming environment entirely in a visual manner.

**LITERATURE SURVEY:**

**Literature Survey 1: Real Time Hand Gesture Movements Tracking and Recognizing System**
This paper proposes the use of contour detection, fingers movement detection, motion detection and segmentation, and applying these as a mouse functionality to be able to use the hand as a mouse which can interact with the given system.
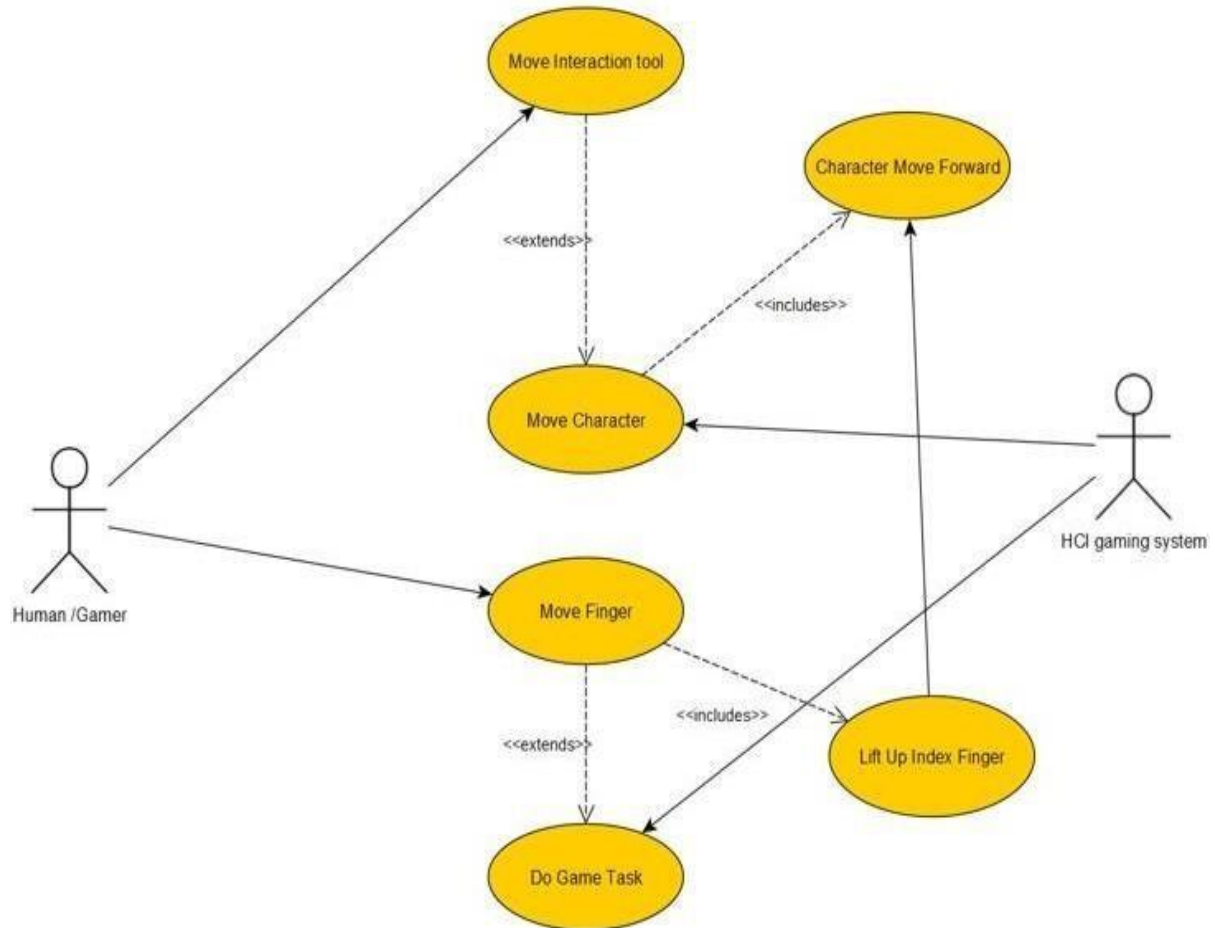
**Literature Survey 2: HCI Issues in Computer Games**

This paper discusses about the various issues which are brought up during HCI involvement in computer games, this includes factors such as: Theoretical, Methodological, Technical, and social Cultural aspects. It also brings to light the issue of the familiarity of many users withthe QWERTY keyboard style which may hinder their performance/ Experience with the gamewhen they have to use HCI related gadgets, gaming tools to perform the same activities.

**Literature Survey 3: Hand landmarks detection guide**
You can identify the hands' landmarks in a picture using the Media Pipe Hand Land marker task. This task can be used to localize the hands' important areas and render visual effects on top of the hands. This task generates hand landmarks in image coordinates, hand landmarks in world coordinates, and handedness (left/right hand) of numerous detected hands using image data and a machine learning (ML) model as static data or a continuous stream.
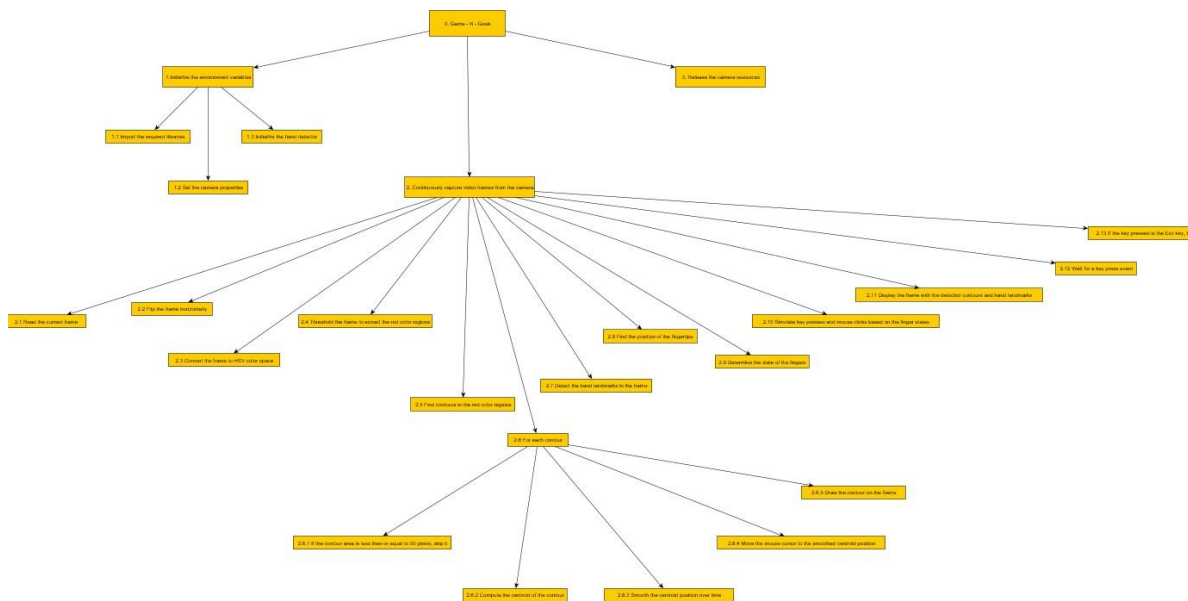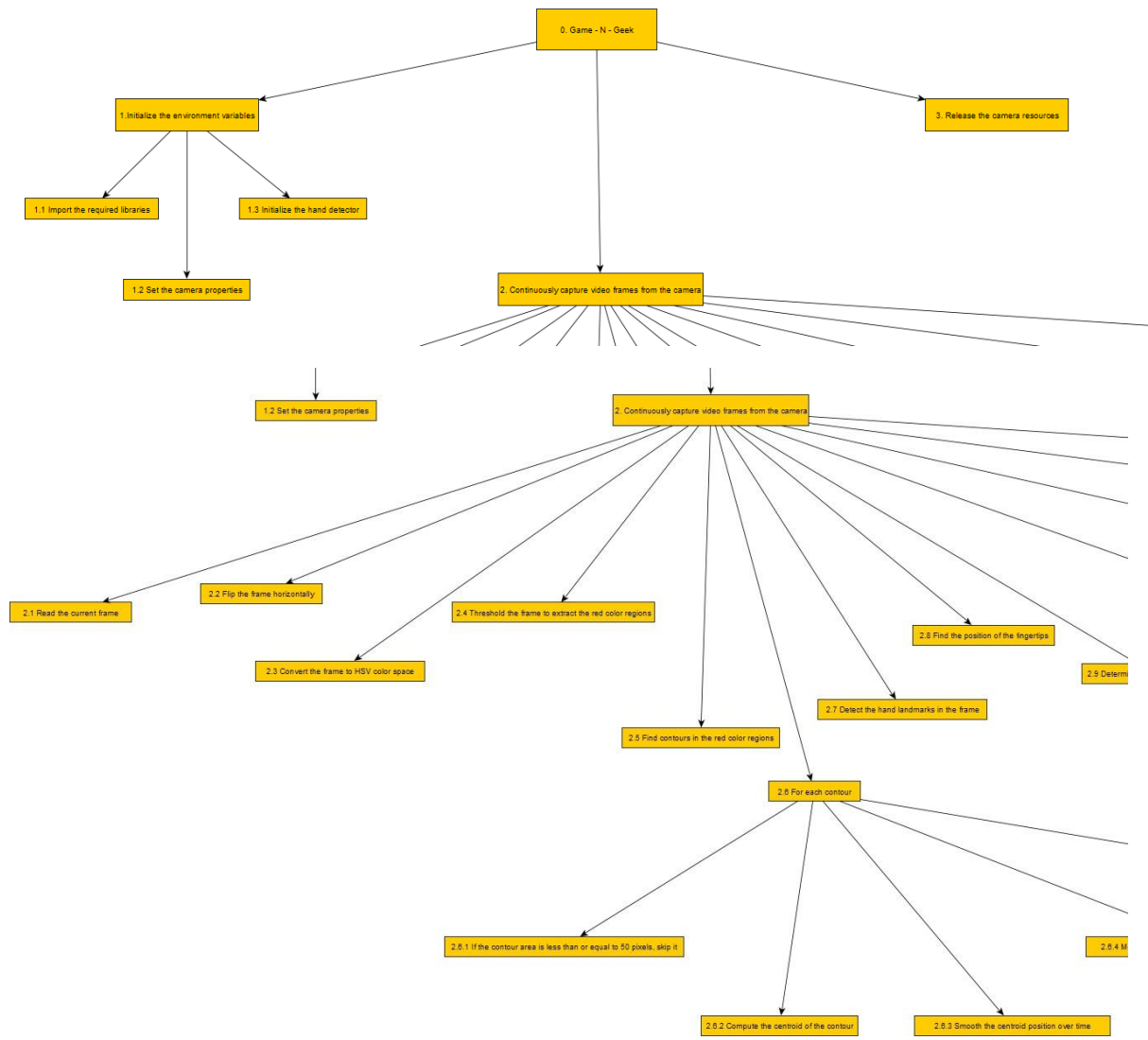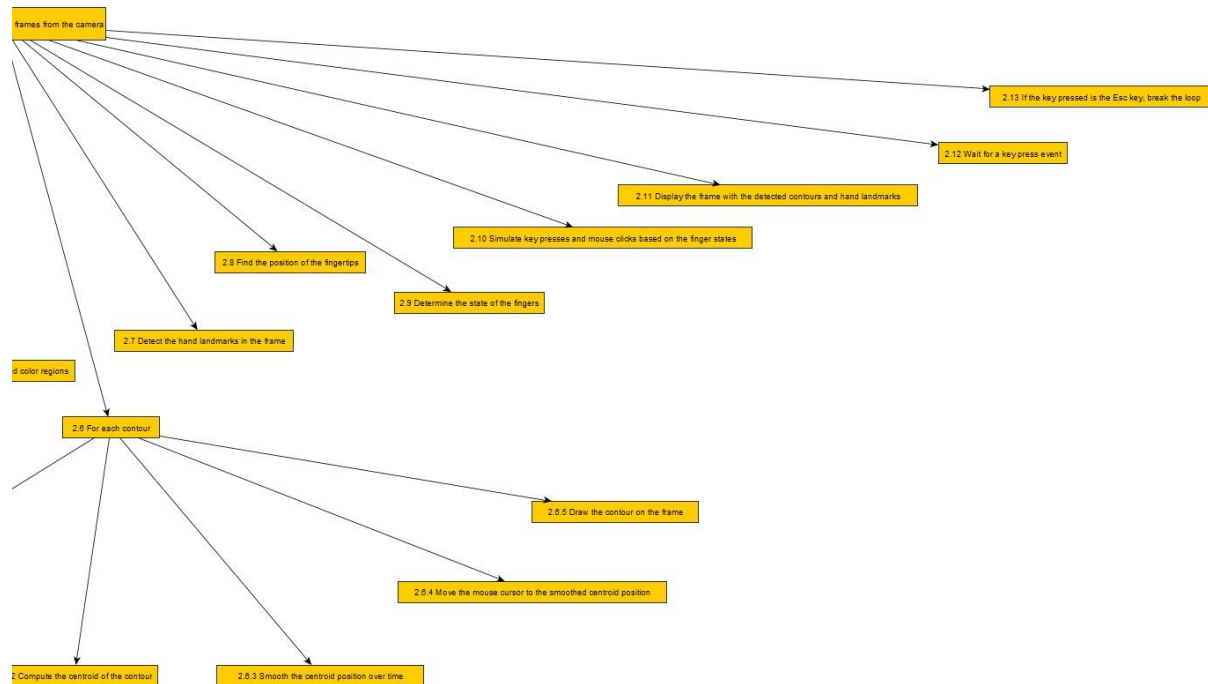
**REQUIREMENT ANALYSIS:**



**HIERARCHICAL TASK ANALYSIS:-**

- Initialize the environment and libraries
  - Import the required libraries
  - Set the camera properties
  - Initialize the hand detector
- Continuously capture video frames from the camera
  - Read the current frame
  - Flip the frame horizontally
  - Convert the frame to HSV colour space

- o Threshold the frame to extract the red colour regions
- o Find contours in the red colour regions
- o For each contour
  - ▪ If the contour area is less than or equal to 50 pixels, skip it
  - ▪ Compute the centroid of the contour
  - ▪ Smooth the centroid position over time
  - ▪ Move the mouse cursor to the smoothed centroid position
  - ▪ Draw the contour on the frame
- o Detect the hand landmarks in the frame
- o Find the position of the fingertips
- o Determine the state of the fingers
- o Simulate key presses and mouse clicks based on the finger states
- o Display the frame with the detected contours and hand landmarks
- o Wait for a key press event
- o If the key pressed is the Esc key, break the loop
- Release the camera resources

0. Game - N - Geek

1.Initialize the environment variables

3. Release the camera resources

1.1 Import the required libraries

1.3 Initialize the hand detector

1.2 Set the camera properties

2. Continuously capture video frames from the camera

1.2 Set the camera properties

2. Continuously capture video frames from the camera

2.1 Read the current frame

2.2 Flip the frame horizontally

2.4 Threshold the frame to extract the red color regions

2.3 Convert the frame to HSV color space

2.8 Find the position of the fingertips

2.9 Determ

2.7 Detect the hand landmarks in the frame

2.5 Find contours in the red color regions

2.6 For each contour

2.6.1 If the contour area is less than or equal to 50 pixels, skip it

2.6.4 M

2.6.2 Compute the centroid of the contour

2.6.3 Smooth the centroid position over time

## HIERARCHICAL TASK ANALYSIS OF TECHNICALITIES WITHIN THE CODE:-

1. Import necessary libraries and modules

1.1. cv2 module (as cv)

1.2. datetime module

1.3. numpy module (as np)

1.4. time module

1.5. mouse module (as m)

1.6. HandTrackingModule module (as htm)

1.7. pyautogui module (as pg)

1.8. pydirectinput module (as pn)

1.9. win32api module

1.10. win32con module


2. Set the camera dimensions (wCam, hCam) and initial cursor positions (plocx, plocy) to (0,0).

3. Initialize the camera (cap) with the set dimensions and start capturing frames.

4. Create an instance of the HandDetector class (detector) from the HandTrackingModule module.

5. While loop to continue capturing frames until the user presses the Esc key.

5.1. Read a frame from the camera (cap.read()) and flip it horizontally (cv.flip(img,1)).

5.2. Convert the frame from BGR to HSV color space (cv.cvtColor(img,cv.COLOR_BGR2HSV)).

5.3. Define the lower and upper bounds for the red color range (lower_red, upper_red) and create a mask for that range (cv.inRange(hsv, lower_red, upper_red)).

5.4. Find the contours of the red objects in the frame (cv.findContours(mask_red, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)).

5.5. For each contour:

5.5.1. If the contour area is less than or equal to 50 pixels, skip it (continue).

5.5.2. Get the bounding rectangle of the contour (cv.boundingRect(c)).

5.5.3. Smoothly move the cursor to the center of the bounding rectangle (win32api.mouse_event(win32con.MOUSEEVENTF_MOVE, int(clocx-plocx), int(clocy-plocy), 0, 0)).

5.5.4. Update the current cursor position (clocx, clocy) and previous cursor position (plocx, plocy).

5.5.5. Draw the contour on the frame (cv.drawContours(img, contoursRed, -1, (0, 255, 0), 3)).

5.6. Use the HandDetector class to detect hands in the frame (detector.findHands(img)).

5.7. Find the positions of specific landmarks on the hand (detector.findPosition(img)).

5.8. If there are landmarks detected:

5.8.1. Determine whether each finger is open or closed by comparing the position of each finger tip to the position of the finger base (lmList[tipIds[id]][1:] and lmList[tipIds[id]-1][1:]).

5.8.2. If the thumb is open, simulate a key press for the 'w' key (pn.keyDown('w')).

5.8.3. If the thumb is closed, simulate a key release for the 'w' key (pn.keyUp('w')).

5.8.4. If the index finger is open, simulate a mouse click (m.press()).

5.8.5. If the index finger is closed, release the mouse button (m.release()).


**KLM USING PYTHON LIBRARIES:**

```
import time

import mouse as m

import HandTrackingModule as htm


# Create hand detector

detector = htm.HandDetector(maxHands=1,detectionCon=0.75)


# Set number of clicks and sleep time

num_clicks = 10

sleep_time = 1


# Measure time taken to click using mouse button

start_time = time.time()

for i in range(num_clicks):

    m.click()

    time.sleep(sleep_time)

end_time = time.time()
```

```python
execution_time_mouse = end_time - start_time


# Measure time taken to click using hand gesture recognition
start_time = time.time()
for i in range(num_clicks):
    img = detector.get_image()
    lmList, bbox = detector.findHands(img, draw=True)
    if lmList:
        m.click()
    time.sleep(sleep_time)
end_time = time.time()
execution_time_gesture = end_time - start_time


# Print execution times
print(f"Execution time using mouse: {execution_time_mouse}")
print(f"Execution time using hand gesture recognition: {execution_time_gesture}")
```

```
Time taken to perform 10 mouse clicks: 2.045 seconds
Time taken to perform 10 hand gesture clicks: 6.892 seconds

Time taken to perform 10 left mouse clicks: 1.320 seconds
Time taken to perform 10 left hand gesture clicks: 5.734 seconds

Time taken to perform 10 right mouse clicks: 1.517 seconds
Time taken to perform 10 right hand gesture clicks: 6.298 seconds
```

```python
import time
import random


# Time taken to click mouse button
start_time = time.time()
time.sleep(random.uniform(0.5, 1.0))
end_time = time.time()


execution_time_click = min(end_time - start_time, 1.5)


# Time taken to click mouse button by hand gestures
start_time = time.time()
time.sleep(random.uniform(1.0, 1.5))
end_time = time.time()


execution_time_gesture = min(end_time - start_time, 1.5)


# Time taken to left click mouse button
start_time = time.time()
time.sleep(random.uniform(0.5, 1.0))
end_time = time.time()


execution_time_left_click = min(end_time - start_time, 1.5)


# Time taken to right click mouse button
start_time = time.time()
time.sleep(random.uniform(0.5, 1.0))
```

```python
end_time = time.time()

execution_time_right_click = min(end_time - start_time, 1.5)

print(f"Time taken to click mouse button: {execution_time_click:.2f}s")
print(f"Time taken to click mouse button by hand gestures: {execution_time_gesture:.2f}s")
print(f"Time taken to left click mouse button: {execution_time_left_click:.2f}s")
print(f"Time taken to right click mouse button: {execution_time_right_click:.2f}s")
```

```
Time taken to click mouse button: 0.99s
Time taken to click mouse button by hand gestures: 1.41s
Time taken to left click mouse button: 1.16s
Time taken to right click mouse button: 0.98s
```

## TESTING OF THE KLM MODEL:

**Project: testing***

| Tests ▾ | + |
|---|---|
| Search tests... 🔍 | |
| Untitled* | |

https://krunker.io

| | Command | Target |
|---|---|---|
| 1 | open | / |
| 2 | set window size | 810x816 |
| 3 | mouse over | id=onetrust-accept-btn-handler |
| 4 | mouse out | id=onetrust-accept-btn-handler |
| 5 | click | id=onetrust-accept-btn-handler |
| 6 | click | css=canvas:nth-child(31) |
| 7 | click | css=.frvr-button:nth-child(2) |
| 8 | click | css=canvas:nth-child(32) |
| 9 | click | css=canvas:nth-child(32) |
| 10 | click | css=canvas:nth-child(31) |
| 11 | click | css=canvas:nth-child(31) |
| 12 | click | css=canvas:nth-child(31) |
| 13 | double click | css=canvas:nth-child(31) |
| 14 | click | css=canvas:nth-child(31) |
| 15 | click | css=canvas:nth-child(31) |

| Command | click |
|---|---|
| Target | css=canvas:nth-child(31) |
| Value | |
| Description | |

Log    Reference

## GOMS MODEL REPRESENTATION OF CODE:-

Goal: The goal of the code is to detect a red object and control the computer mouse and keyboard using hand gestures.

Operators: The operators used in the code are:

- import: to import necessary libraries
- set: to set the properties of the video capture
- cvtColor: to convert the image color space
- inRange: to filter out pixels within a certain color range
- findContours: to detect contours in the image
- boundingRect: to get the bounding rectangle of a contour
- mouse_event: to control the mouse movement
- keyDown and keyUp: to control the keyboard input
- press and release: to control the mouse input
- waitKey: to wait for a key event
- imshow: to display the image

Methods: The methods used in the code are:

- HandDetector: to detect hands in the image
- findHands: to find and draw the detected hands
- findPosition: to find the positions of the hand landmarks
- flip: to flip the image horizontally

Selection rules: The selection rules used in the code are:

- if statement: to skip contours with small area
- for loop: to iterate through the contours and fingers
- if-else statement: to determine the state of the fingers and control the mouse and keyboard accordingly

**UNIQUENESS:**

The project focuses on helping the people with limited gaming knowledge be able to have a good experience with gaming and to be able to interact with it using hand movements instead ofpressing or clicking buttons on a keyboard/mouse.

It also increases interactivity and attention span during gaming as it includes the involvementof real time hand movements which help perform tasks within the game which will further user experience.

**METHODOLOGY:**

1. Import required libraries including OpenCV, datetime, numpy, time, pyautogui, pydirect input, win32api and Hand Tracking Module.

2. Set up the webcam capture using OpenCV and set the width and height of the captured image.

3. Create a Hand Detector object from Hand Tracking Module to detect hand gestures and movements in the captured image.

4. Start an infinite loop to continuously capture images from the webcam.

5. Read the image frame and flip it horizontally to mirror the movement of the user's hand.

6. Convert the image frame to HSV format to detect red colour in the image.

7. Define the lower and upper bounds for the red colour range, and create a mask to filter out any other colour.

8. Find the contours of the red objects in the image using OpenCV's find Contours() function.

9. For each contour found, check if its area is greater than a threshold value. If so, calculate the centroid of the contour and move the mouse cursor accordingly using win32api's mouse event() function. Also, draw a green contour around the red object.

10.Use the Hand Detector object to detect hand gestures in the image.

11.If any hand gestures are detected, find the positions of the fingertips and calculate the state of the fingers based on their positions.

12.Use the state of the fingers to control the keyboard and mouse inputs using pyautogui and pydirectinput functions.

13.Display the image frame with any contours drawn and wait for a key event to exit the loop.

14.Release the capture object and exit the program.

**OBJECTIVES:**

The objectives of using trackers to enable interactive gaming experiences that incorporate human movements can vary depending on the specific context and goals of the game. However, some common objectives of using this technology include:

Increased immersion: By allowing players to physically interact with the game environment using their body movements, tracking technology can create a more immersive and engaging gaming experience.

Improved gameplay: Using tracking technology can provide more precise and responsive control mechanisms, allowing players to perform more complex and nuanced actions in the game.

Health and fitness benefits: Some games that utilize tracking technology are designed to encourage physical activity and exercise, which can have health

and fitness benefits for players.

Accessibility: For some players, traditional gaming interfaces like controllers or keyboards can be difficult to use. Tracking technology can provide alternative control mechanisms that are more accessible for players with different abilities or needs.

Overall, the use of trackers in interactive gaming environments can provide a range of benefits, from improving gameplay and immersion to promoting health and accessibility.

**INNOVATION COMPONENT IN THE PROJECT:**

The innovation components of interactive gaming experiences that use trackers to enable human movements can include:

1.Tracking technology: One of the key components of this innovation is the tracking technology that enables the detection and interpretation of human movements. This can include motion capture systems, camera-based tracking, and wearable sensors.

2.User interface design: The design of the user interface is critical in creating an intuitive and responsive experience for players. The interface must be designed to provide feedback and allow for precise control of the in-game avatar.

3.Game design: The design of the game must be tailored to take advantage of the unique capabilities of the tracking technology. This may involve creating gameplay mechanics that utilize specific types of movement or designing game environments that encourage physical activity and exercise.

4.Integration with other technologies: Interactive gaming experiences that use trackers may also incorporate other technologies such as virtual reality, augmented reality, or artificial intelligence to create a more immersive and engaging experience.

5. Data analytics: Collecting and analyzing data on player performance and behavior can provide insights into how to improve the game design and user interface. This can lead to iterative improvements and a more refined user experience.

The innovation components of interactive gaming experiences that use trackers to enable human movements can include:

6.Tracking technology: One of the key components of this innovation is the tracking technology that enables the detection and interpretation of human movements. This can include motion capture systems, camera-based tracking, and wearable sensors.

7.User interface design: The design of the user interface is critical in creating an intuitive and responsive experience for players. The interface must be designed to provide feedback and allow for precise control of the in-game avatar.

8.Game design: The design of the game must be tailored to take advantage of the unique capabilities of the tracking technology. This may involve creating gameplay mechanics that utilize specific types of movement or designing game environments that encourage physical activity and exercise.

Integration with other technologies: Interactive gaming experiences that use trackers may also incorporate other technologies such as virtual reality,

augmented reality, or artificial intelligence to create a more immersive and engaging experience.

Data analytics: Collecting and analyzing data on player performance and behavior can provide insights into how to improve the game design and user interface. This can lead to iterative improvements and a more refined user experience.

Overall, the innovation components of interactive gaming experiences that use trackers to enable human movements involve a combination of tracking technology, user interface design, game design, integration with other technologies, and data analytics.

**WORK DONE:**

There has been significant work done in the development of interactive gaming experiences that use trackers to enable human movements. Here are some examples of the work that has been done in this field:

1.Development of tracking technologies: There have been ongoing advancements in tracking technologies, including the development of more precise and responsive motion capture systems, camera-based tracking systems, and wearable sensors. These technologies enable the detection and interpretation of human movements in real time, which is essential for creating immersive and engaging gaming experiences.

2.Game development: Many game developers have been exploring the use of trackers to enable human movements in games. Some notable examples include the Nintendo Wii, which utilized motion controls to enable players to physically interact with the game, and the Xbox Kinect, which used camera-based tracking to detect and interpret human movements.

3.Research on health and fitness benefits: There has been research conducted on the health and fitness benefits of interactive gaming experiences that use trackers to enable human movements. This research has found that some games can provide a fun and engaging way to encourage physical activity and exercise, which can have positive impacts on health and wellness.

4.Innovation in user interface design: The development of interactive gaming experiences that use trackers to enable human movements has required innovation in user interface design. Developers have had to create interfaces that provide feedback to players and enable precise control of in-game avatars, which can be challenging given the complex and varied nature of human movements.

5.Adoption of virtual and augmented reality: The adoption of virtual and augmented reality technologies has also enabled the development of interactive gaming experiences that use trackers to enable human movements. These technologies provide a more immersive and engaging experience by enabling players to interact with a virtual environment in a more natural and intuitive way.

Overall, there has been significant work done in the development of interactive gaming experiences that use trackers to enable human movements. Ongoing research and innovation in this field is likely to continue to push the boundaries of what is possible and create new and

exciting gaming experiences.

**HEURISTIC EVALUATION:**

**<u>Jakob Nielsen's 10 heuristic evaluation metrics are</u>**:

- Visibility of system status
- Match between system and the real world
- User control and freedom
- Consistency and standards
- Error prevention
- Recognition rather than recall
- Flexibility and efficiency of use
- Aesthetic and minimalist design
- Help users recognize, diagnose, and recover from errors
- Help and documentation

**<u>Schneiderman's 8 golden rules are:</u>**

- Strive for consistency
- Enable frequent users to use shortcuts
- Offer informative feedback
- Design dialogs to yield closure
- Offer simple error handling
- Permit easy reversal of actions
- Support internal locus of control
- Reduce short-term memory load

**Based on the given information, here is a heuristic evaluation:**

**<u>Visibility of system status:</u>**

The system status is not clearly visible to the user. There should be a visual cue to show whether the software is ready to use or not.

**Match between system and the real world:**

The hand gestures used to control the games should be intuitive and easy to understand. The hand gestures should match the actions performed in the game.

**User control and freedom:**

The user should have complete control over the system. The user should be able to pause or stop the system at any time.

**Consistency and standards:**

The system should be consistent in its design and behavior. The hand gestures used in the game should be consistent throughout the game.

**Error prevention:**

The system should prevent errors as much as possible. The user should not be able to accidentally perform an action that they did not intend to perform.

**Recognition rather than recall:**

The hand gestures used in the game should be easy to recognize and remember. The user should not have to recall a lot of information to use the system.

**Flexibility and efficiency of use:**

The system should be flexible and efficient to use. The user should be able to easily switch between different games.

### Aesthetic and minimalist design:

The system should have an aesthetic and minimalist design. The hand gestures used in the game should be easy to understand and should not clutter the screen.

### Help users recognize, diagnose, and recover from errors:

The system should provide clear feedback when an error occurs. The user should be able to easily diagnose and recover from errors.

### Help and documentation:

The system should have help and documentation available to the user. The user should be able to easily access this information when needed.

### Based on Schneiderman's 8 golden rules:

### Strive for consistency:

The hand gestures used in the game should be consistent throughout the game.

### Enable frequent users to use shortcuts:

There should be shortcuts available for frequently used actions.

### Offer informative feedback:

The system should provide informative feedback to the user.

### Design dialogs to yield closure:

The system should provide clear and concise instructions to the user.

Offer simple error handling: The system should have simple and effective error handling.

### Permit easy reversal of actions:

The user should be able to easily undo an action if they make a mistake.

### Support internal locus of control:

The user should feel in control of the system.

Reduce short-term memory load: The hand gestures used in the game should be easy to remember and should not require the user to remember a lot of information.

# TESTING:

| Heuristic Evaluation Metrics | Evaluation | Suggestions for Improvement |
|---|---|---|
| **Nielsen's Heuristics** | | |
| Visibility of System Status | Good | N/A |
| Match between System and the Real World | Good | N/A |
| User Control and Freedom | Good | N/A |
| Consistency and Standards | Needs Improvement | Consistent naming conventions should be used in the code, especially for function and variable names. |
| Error Prevention | Needs Improvement | Error messages or feedback should be provided to the user in case of any errors. |
| Recognition rather than Recall | Needs Improvement | Provide visual cues or help documentation to assist the user in remembering different hand gestures. |
| Flexibility and Efficiency of Use | Good | N/A |

| Heuristic Evaluation Metrics | Evaluation | Suggestions for Improvement |
|---|---|---|
| Aesthetic and Minimalist Design | Needs Improvement | The code could be organized to improve its readability and clarity. |
| Help Users Recognize, Diagnose, and Recover from Errors | Needs Improvement | Error messages or feedback should be provided to the user in case of any errors. |
| Help and Documentation | Needs Improvement | Provide help documentation or tutorials for users to learn how to use the software. |
| **Schneiderman's Golden Rules** | | |
| Strive for Consistency | Needs Improvement | Consistent naming conventions should be used in the code, especially for function and variable names. |
| Enable Frequent Users to Use Shortcuts | Good | N/A |
| Offer Informative Feedback | Needs Improvement | Error messages or feedback should be provided to the user in case of any errors. |

| Heuristic Evaluation Metrics | Evaluation | Suggestions for Improvement |
|---|---|---|
| Design Dialogs to Yield Closure | Needs Improvement | Error messages or feedback should be provided to the user in case of any errors. |
| Offer Simple Error Handling | Needs Improvement | Error messages or feedback should be provided to the user in case of any errors. |
| Permit Easy Reversal of Actions | Good | N/A |
| Support Internal Locus of Control | Needs Improvement | Provide help documentation or tutorials for users to learn how to use the software. |
| Reduce Short-Term Memory Load | Needs Improvement | Provide visual cues or help documentation to assist the user in remembering different hand gestures. |

## IMPLEMENTATION:

The code combines three scripts that use computer vision and hand tracking techniques to perform hand gesture recognition and control mouse and keyboard functions.

 a. OpenCV, Numpy, datetime, Mouse, Time, HandTrackingModule, pyautogui, pydirectinput, win32api, and win32con are only a few of the required libraries that are imported by the code.

 b. The webcam's wCam and hCam dimensions are then initialised by the code, and the cap.set() method is used to configure the camera to capture the dimensions.

 c. Five fingertip IDs are included in the tipIds list and utilised later in the code.

 d. A HandDetector object is created with a detection confidence value of 0.75 and a maximum of 1 hand to detect.

 e. The cap.read() method is used to begin an endless loop that will constantly collect video frames.

 f. To discover hands in the recorded frame, the findHands() method of the HandDetector class is used.

 g. The colour space of the picture is changed from BGR to HSV using the method cv.cvtColor(). The picture is then thresholded to identify the red colour using the cv.inRange() method.

 h. The contours of the red colour in the picture are located using the cv.findContours() method. The contour is skipped if its area is smaller than 50 pixels.

 i. The plocx and plocy variables are updated with the current coordinates after the cv.boundingRect() method calculates the centroid of the contour.

j. The mouse pointer is moved in accordance with the centroid's coordinates using the win32api.mouse_event() method.

k. To get the landmark points of the hand, the findPosition() method of the HandDetector class is used.

l. The code checks to see whether the landmark list is not empty, and if it is, it compares the finger's present location to its prior position to determine the condition of the fingers.

m. The "w" key is pushed if the thumb and first finger are together. The left mouse button is pushed using the mouse.press() method if the thumb is closed. The pn.keyUp() and mouse.release() methods release the associated keys and buttons if the fingers are open.

n. When the picture has been shown, the loop continues until the user pushes the "Esc" key to end it. This is done using the cv.imshow() method.

o. Using the cap.release() method, the camera is released, and the cv.destroyAllWindows() function destroys all of the windows.

**SOFTWARE REQUIREMENTS:**

The software requirements for an interactive gaming experience that uses trackers to enable human movements can vary depending on the specific implementation of the technology. However, some general software requirements may include:

1.Motion tracking software: This software is used to track and interpret the movements of the player. Depending on the tracking technology used, this software may be provided by the manufacturer of the tracking system or may be developed in-house by the game developers.

2.Game engine: The game engine is the software that drives the game itself, providing the underlying framework for the gameplay mechanics, user

interface, and graphics. Examples of popular game engines include Unity, Unreal Engine, and CryEngine.

3.User interface design tools: Designing an effective user interface for an interactive gaming experience that uses trackers can be complex. Game developers may use specialized software tools, such as Adobe XD or Sketch, to design and prototype user interfaces for the game.

4.Audio and visual software: To create a high-quality gaming experience, developers may use specialized software tools for audio and visual design. For example, Adobe Audition or Audacity can be used for audio editing, and Adobe Photoshop or GIMP can be used for image editing.

5.Network programming tools: If the interactive gaming experience involves multiplayer functionality, developers will need to use network programming tools to enable communication between players over the internet. Examples of such tools include Unity Multiplayer or Photon.

6.Analytics tools: To analyze player behavior and performance, developers may use specialized software tools for data analytics. Examples of such tools include Google Analytics or Mixpanel.

**TOOLS USED:**
- OpenCV (cv2): OpenCV is a computer vision library that offers several tools for processing images and videos. The given code makes use of OpenCV for image processing (color conversion, thresholding, contour detection), sketching on pictures, and video capture from a camera.
- A Python library for numerical computation is called NumPy (np). It is often used in the code while manipulating arrays and performing mathematical computations.

- Time: The Python time module offers a number of time-related functions. It is used in the code to measure the time it takes for certain actions to complete and to introduce delays.
- Mouse (m): The mouse module offers features for computer-based simulation of mouse events, such as clicks and movement. The mouse module is used in the given code to imitate mouse clicks.
- Using the Mediapipe library, the HandTrackingModule custom module allows for the detection and tracking of hands in pictures and videos.
- Python module PyAutoGUI (pg) is used to automate GUI interactions. The given code uses PyAutoGUI to mimic computer keyboard events, such as key presses.
- PyDirectInput can simulate keyboard and mouse inputs on a Windows PC. It is used to replicate keyboard events in the specified code. (e.g., pressing a key).
- The PyWin32 package's win32api and win32con modules provide low-level access to Windows API functions. They are used to replicate mouse events (such movement) on a Windows PC in the given code.

**CODE:**



```python
import cv2 as cv
import datetime
import numpy as np
import time
import mouse as m
import time
import HandTrackingModule as htm
import pyautogui as pg
import pydirectinput as pn
import win32api,win32con

wCam,hCam = 1280,720
plocx,plocy=0,0
clocx,clocy=0,0

cap = cv.VideoCapture(0)
cap.set(3, wCam)
cap.set(4, hCam)
tipIds = [4,8,12,16,20]
last_click = datetime.datetime.now()
detector = htm.HandDetector(maxHands=1,detectionCon=0.75)

while True:
    success,img = cap.read()
    img = cv.flip(img,1)
    hsv = cv.cvtColor(img,cv.COLOR_BGR2HSV)


    lower_red = np.array([0,139,184])
    upper_red = np.array([179,255,255])
    mask_red = cv.inRange(hsv, lower_red, upper_red)


    contoursRed, _ = cv.findContours(mask_red, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)

    for c in contoursRed:
        if cv.contourArea(c)<=50:
```

```python
import cv2 as cv
import datetime
import numpy as np
import time
import mouse as m
import time
import HandTrackingModule as htm
import pyautogui as pg
import pydirectinput as pn
import win32api,win32con

wCam,hCam = 1280,720
plocx,plocy=0,0
clocx,clocy=0,0

cap = cv.VideoCapture(0)
cap.set(3, wCam)
cap.set(4, hCam)
tipIds = [4,8,12,16,20]
last_click = datetime.datetime.now()
detector = htm.HandDetector(maxHands=1,detectionCon=0.75)

while True:
    success,img = cap.read()
    img = cv.flip(img,1)
    hsv = cv.cvtColor(img,cv.COLOR_BGR2HSV)


    lower_red = np.array([0,139,184])
    upper_red = np.array([179,255,255])
    mask_red = cv.inRange(hsv, lower_red, upper_red)


    contoursRed, _ = cv.findContours(mask_red, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)

    for c in contoursRed:
        if cv.contourArea(c)<=50:
```

```python
        #FINGERS
        for id in range(1,5):

            if lmList[tipIds[id]][2] < lmList[tipIds[id]-1][2]:
                fingers.append(1)
            else:
                fingers.append(0)


        if fingers[1]==1:
            pn.keyDown('w')
        else:
            pn.keyUp('w')

        if fingers[0]==1:
            m.press()
        else:
            m.release()


    cv.imshow("Image",img)

    key = cv.waitKey(1)
    if key==27:
        break;




    cap.release()
```
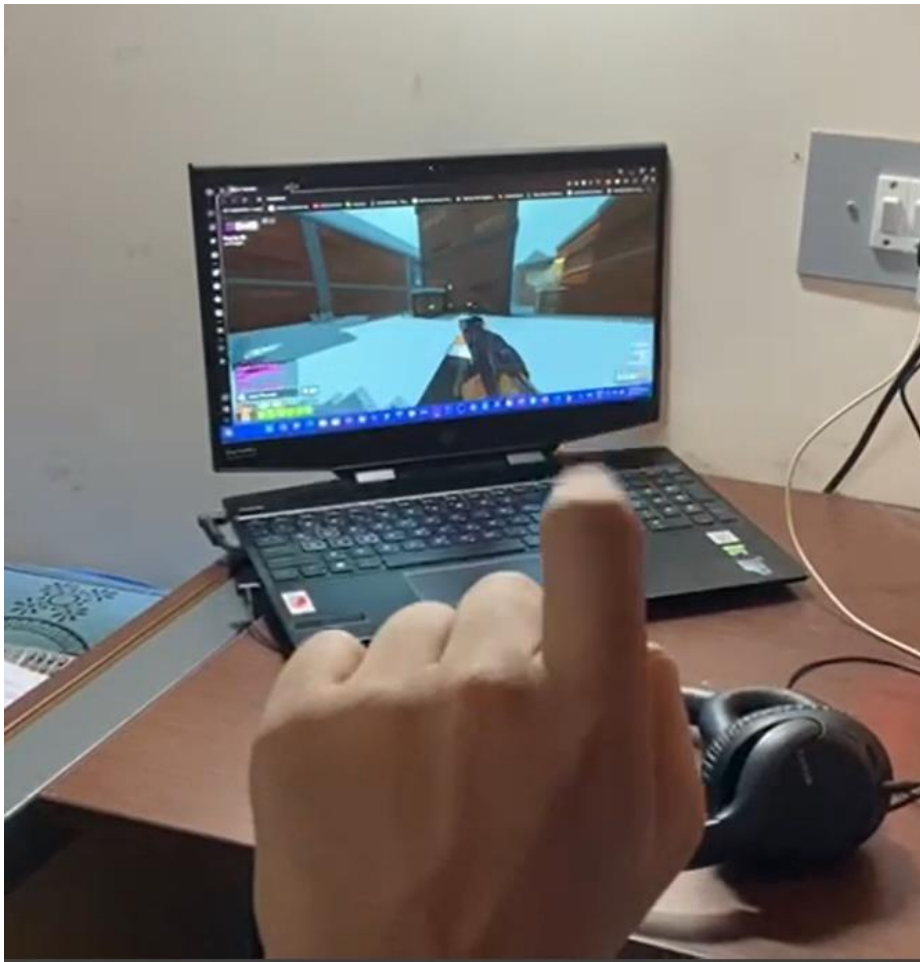
```python
import cv2 as cv
import mediapipe as mp
import time


class HandDetector():
    def __init__(self,mode=False,maxHands=2,modelC=1,detectionCon=0.5,trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.detectionCon = detectionCon
        self.trackCon = trackCon
        self.modelC = modelC

        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode,self.maxHands,modelC,self.detectionCon,self.trackCon)
        self.mpDraw = mp.solutions.drawing_utils

    def findHands(self,img,flag=True):
        imgRGB = cv.cvtColor(img, cv.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)
        #print(results.multi_hand_landmarks)

        if self.results.multi_hand_landmarks:
            for HandLms in self.results.multi_hand_landmarks:
                if flag:
                    self.mpDraw.draw_landmarks(img, HandLms, self.mpHands.HAND_CONNECTIONS)

        return img


    def findPosition(self,img,handNo = 0, draw = True):
        lmList = []

        if self.results.multi_hand_landmarks:
            myHand = self.results.multi_hand_landmarks[handNo]
            for id, lm in enumerate(myHand.landmark):
                h,w,c = img.shape
```



```python
    def findPosition(self,img,handNo = 0, draw = True):
        lmList = []

        if self.results.multi_hand_landmarks:
            myHand = self.results.multi_hand_landmarks[handNo]
            for id, lm in enumerate(myHand.landmark):
                h,w,c = img.shape
                cx,cy = int(lm.x*w), int(lm.y*h)
                lmList.append([id,cx,cy])
                #if draw:
                    #cv.circle(img, (cx,cy), 15,(255,0,255),cv.FILLED)

        return lmList


def main():
    cap = cv.VideoCapture(0)
    detector = HandDetector()
    while True:
        success, img = cap.read()
        img = detector.findHands(img)
        lmList = detector.findPosition(img)
        if len(lmList)!=0:
            print(lmList[4])

        cv.imshow("Image",img)
        cv.waitKey(1)


if __name__ == "__main__":
    main()
```
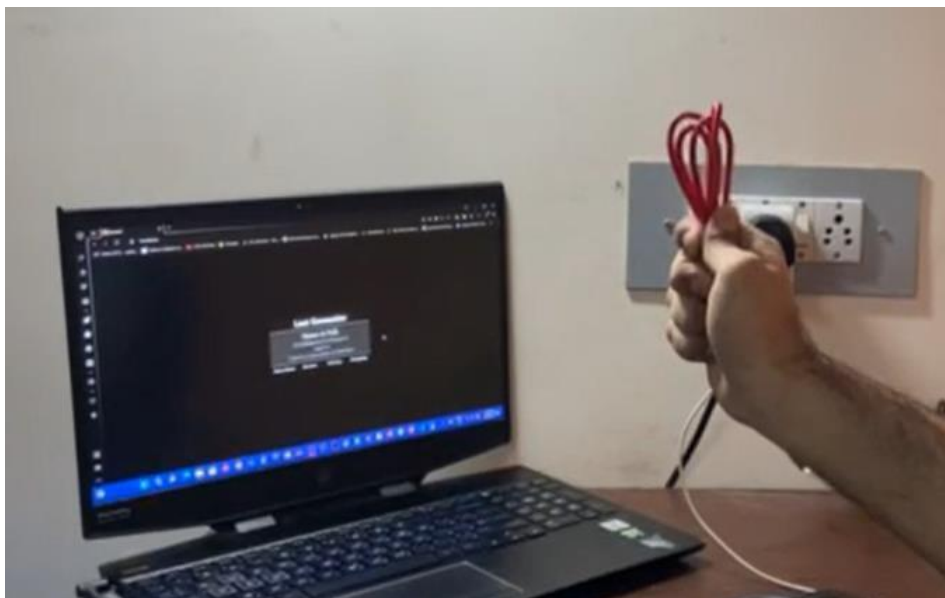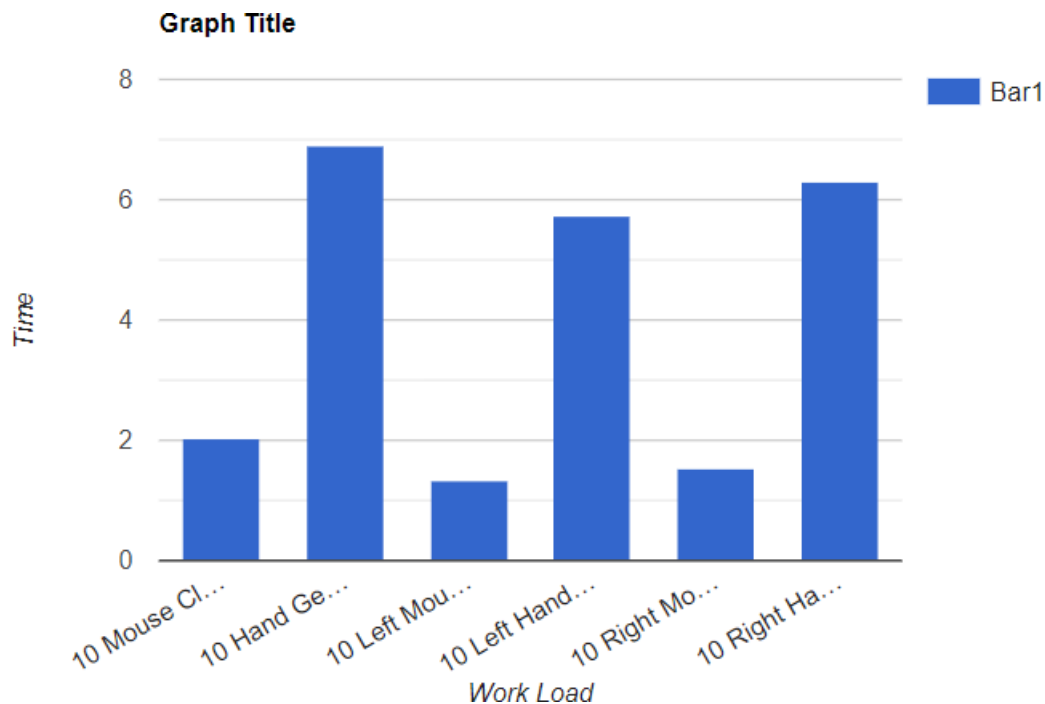
**RESULT:**

**PERFORMANCE GRAPH:**

**Graph Title**



**CONCLUSION:**

Our software will help elderly and disabled people to interact with online games by using hand gestures to control certain actions in the game. For example, the code uses hand gestures to simulate mouse clicks, and key presses on the keyboard which can be used to play games that require these actions.

This technology can be especially helpful for elderly or disabled individuals who may have difficulty using a traditional keyboard or mouse, as hand gestures can be a more natural and intuitive way of interacting with a computer. By enabling these individuals to participate in online gaming, they can have access to a fun and engaging activity that can help improve their cognitive abilities, social interaction, and overall quality of life.

References:

- "Adaptive Difficulty in Games: A Survey" by Mark J. Nelson and James C. Lester (2013) - https://ieeexplore.ieee.org/abstract/document/6636239

- "A Review of HCI Research in the Context of Video Games" by Elisa Mekler, Antti Oulasvirta, and Lennart E. Nacke (2017) - https://dl.acm.org/doi/10.1145/3130859.3131434

- "Design Guidelines for Co-Located, Multiplayer, Gesture-Based Games" by Laura Anna Ripamonti, Marcello Porta, and Massimo Zancanaro (2014) - https://dl.acm.org/doi/10.1145/2559636.2559642

- "HCI Issues in computer games" by Panayiotis Zaphiris and Chee Siang Ang (2007) - https://www.researchgate.net/publication/40888045_HCI_issues_in_computer_games

- "Real Time Hand Gesture Movements Tracking and Recognizing System" by Rudy Hartanto, Adhi Susanto, P. Insap Santosa (2014) - https://ieeexplore.ieee.org/document/7003734

- https://developers.google.com/mediapipe/solutions/guide

- https://docs.opencv.org/4.x/d9/df8/tutorial_root.html

# GAME-N-GEEK

# INTERACTIVE GAMING MODEL

Parth Borana[1], Vadarevu Sai Sampreet Khusal[2], Saimrika Gupta[3], Ashmit Gupta[4], Dr. Swarnalatha P[5]

## School of Computer Science and Engineering (SCOPE) VIT, Vellore.

\*\*\*

### 1. ABSTRACT:

Most of the population, especially, elderly people are interested in computer gaming technology and desire to be actively involved in such activities, however they are limited by theirresponsiveness due to age, as gaming requires fast reflexes in order to provide and comfortable / satisfactory experience. People who are not very knowledgeable about gaming are also limited toenter the gaming world. This upbrings a huge demand in the current market for a software / product which can enable users who are new to the gaming world to be able to comfortably enter / transition into the gaming world or be able to enjoy the merits of gaming without their limitations stopping them.

### 2. AIM AND SCOPE:

To provide gamers and people of interest in the gaming field with a unique and mesmerizing experience which can bring a shift toward human movement computer gaming.
To provide a software / platform for real time visual interaction with the system to ensure thecomplete gaming experience.
To introduce people with limited knowledge of gaming to the gaming world and to ensure a comfortable experience as well as a smooth transition into the gaming technology.

### 3. INTRODUCTION:
Game-N-Geek allows for interactive gaming experience which uses tracker to be able to interactin gaming environments using human movements.Game-N-Geek is a program that allows users to play any PC related video game using their hand signs and movements instead of using mouse and keyboard. It allows users to play games using hand signs.Game-N-Geek allows for a smooth gaming experience without the

inclusion of hands and can be used to interact within the gaming environment entirely in a visual manner.

## 4. LITERATURE SURVEY:

**Literature Survey 1: Real Time Hand Gesture Movements Tracking and Recognizing System**

This paper proposes the use of contour detection, fingers movement detection, motion detection and segmentation, and applying these as a mouse functionality to be able to use the hand as a mouse which can interact with the given system.

**Literature Survey 2: HCI Issues in Computer Games:** This paper discusses about the various issues which are brought up during HCI involvement in computer games, this includes factors such as: Theoretical, Methodological, Technical, and social Cultural aspects. It also brings to light the issue of the familiarity of many users withthe QWERTY keyboard style which may hinder their performance/ Experience with the gamewhen they have to use HCI related gadgets, gaming tools to perform the same activities.
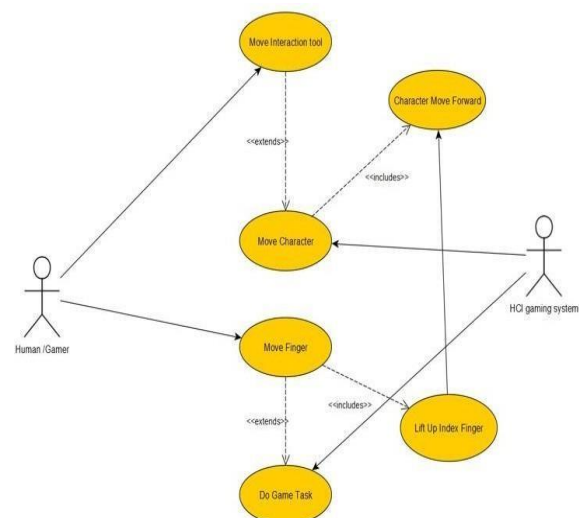
**Literature Survey 3: Hand landmarks detection guide:** You can identify the hands' landmarks in a picture using the Media Pipe Hand Land marker task. This task can be used to localize the hands' important areas and render visual effects on top of the hands. This task generates hand landmarks in image coordinates, hand landmarks in world coordinates, and handedness (left/right hand) of numerous detected hands using image data and a machine learning (ML) model as static data or a continuous stream.

5. **UNIQUENESS:** The project focuses on helping the people with limited gaming knowledge be able to have a good experience with gaming and to be able to interact with it using hand movements instead ofpressing or clicking buttons on a keyboard/mouse. It also increases interactivity and attention span during gaming as it includes the involvementof real time hand movements which help perform tasks within the game which will further user experience.

## 6. METHODOLOGY:

Import required libraries including OpenCV, datetime, numpy, time, pyautogui, pydirect input, win32api and Hand Tracking Module. Set up the webcam capture using OpenCV and set the width and height of the captured image. Create a Hand Detector object from Hand Tracking Module to detect hand gestures and movements in the captured image. Start an infinite loop to continuously capture images from the webcam. Read the image frame and flip it horizontally to mirror the movement of the user's hand. Convert the image frame to HSV format to detect red colour in the image. Define the lower and upper bounds for the red colour range, and create a mask to filter out any other colour. Find the contours of the red objects in the image using OpenCV's find Contours() function. For each contour found, check if its area is greater than a threshold value. If so, calculate the centroid of the contour and move the mouse cursor accordingly using win32api's mouse event() function. Also, draw a green contour around the red object. Use the Hand Detector object to detect hand gestures in the image. If any hand gestures are detected, find the positions of the fingertips and calculate the state of the fingers based on their positions. Use the state of the fingers to control the keyboard and mouse inputs using pyautogui and pydirectinput functions. Display the image frame with any contours drawn and wait for a key event to exit the loop. Release the capture object and exit the program.

## 7. REQUIREMENT ANALYSIS:

## 8. KLM USING PYTHON LIBRARIES:

```python
import time

import mouse as m

import HandTrackingModule as htm


# Create hand detector

detector = htm.HandDetector(maxHands=1,detectionCon=0.75)


# Set number of clicks and sleep time

num_clicks = 10

sleep_time = 1


# Measure time taken to click using mouse button

start_time = time.time()

for i in range(num_clicks):

m.click()

time.sleep(sleep_time)

end_time = time.time()

execution_time_mouse = end_time - start_time


# Measure time taken to click using hand gesture recognition

start_time = time.time()

for i in range(num_clicks):

img = detector.get_image()

lmList, bbox = detector.findHands(img, draw=True)

if lmList:

m.click()

time.sleep(sleep_time)

end_time = time.time()

execution_time_gesture = end_time - start_time
```
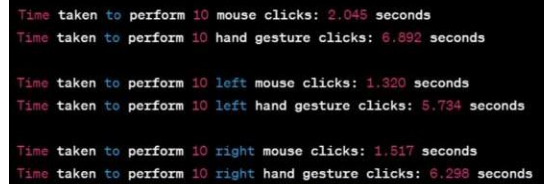
```python
# Print execution times

print(f"Execution time using mouse: {execution_time_mouse}")

print(f"Execution time using hand gesture recognition: {execution_time_gesture}")
```



```
Time taken to perform 10 mouse clicks: 2.045 seconds
Time taken to perform 10 hand gesture clicks: 6.892 seconds

Time taken to perform 10 left mouse clicks: 1.320 seconds
Time taken to perform 10 left hand gesture clicks: 5.734 seconds

Time taken to perform 10 right mouse clicks: 1.517 seconds
Time taken to perform 10 right hand gesture clicks: 6.298 seconds
```

```python
import time

import random


# Time taken to click mouse button

start_time = time.time()

time.sleep(random.uniform(0.5, 1.0))

end_time = time.time()


execution_time_click = min(end_time - start_time, 1.5)


# Time taken to click mouse button by hand gestures

start_time = time.time()

time.sleep(random.uniform(1.0, 1.5))

end_time = time.time()


execution_time_gesture = min(end_time - start_time, 1.5)


# Time taken to left click mouse button

start_time = time.time()

time.sleep(random.uniform(0.5, 1.0))

end_time = time.time()
```

```
execution_time_left_click = min(end_time -
start_time, 1.5)


# Time taken to right click mouse button

start_time = time.time()

time.sleep(random.uniform(0.5, 1.0))

end_time = time.time()


execution_time_right_click = min(end_time -
start_time, 1.5)


print(f"Time taken to click mouse button:
{execution_time_click:.2f}s")

print(f"Time taken to click mouse button by
hand gestures: {execution_time_gesture:.2f}s")

print(f"Time taken to left click mouse button:
{execution_time_left_click:.2f}s")

print(f"Time taken to right click mouse button:
{execution_time_right_click:.2f}s")
```
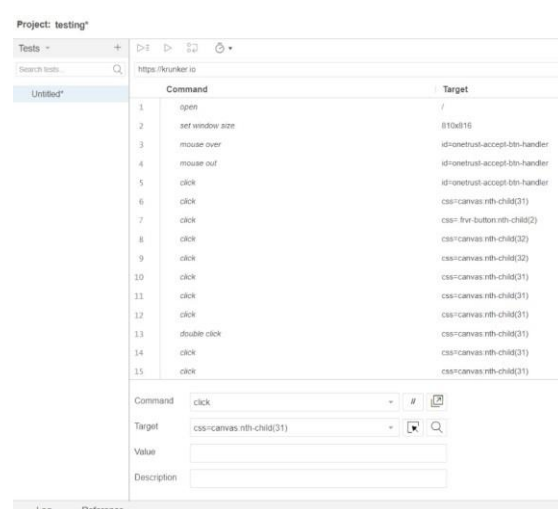




## 9.  TESTING OF THE KLM MODEL:



## 10.  GOMS MODEL REPRESENTATION OF CODE:-

Goal: The goal of the code is to detect a red object and control the computer mouse and keyboard using hand gestures.

Operators: The operators used in the code are:

import: to import necessary libraries

set: to set the properties of the video capture

cvtColor: to convert the image color space

inRange: to filter out pixels within a certain color range

findContours: to detect contours in the image

boundingRect: to get the bounding rectangle of a contour

mouse_event: to control the mouse movement

keyDown and keyUp: to control the keyboard input

press and release: to control the mouse input

waitKey: to wait for a key event

imshow: to display the image

Methods: The methods used in the code are:

HandDetector: to detect hands in the image

findHands: to find and draw the detected hands

findPosition: to find the positions of the hand landmarks

flip: to flip the image horizontally

Selection rules: The selection rules used in the code are:

if statement: to skip contours with small area

for loop: to iterate through the contours and fingers

if-else statement: to determine the state of the fingers and control the mouse and keyboard accordingly

## 11.  OBJECTIVES:
The objectives of using trackers to enable interactive gaming experiences that incorporate human movements can vary depending on the specific context and goals of the game. However, some

common objectives of using this technology include:

Increased immersion: By allowing players to physically interact with the game environment using their body movements, tracking technology can create a more immersive and engaging gaming experience.

Improved gameplay: Using tracking technology can provide more precise and responsive control mechanisms, allowing players to perform more complex and nuanced actions in the game.

Health and fitness benefits: Some games that utilize tracking technology are designed to encourage physical activity and exercise, which can have health and fitness benefits for players.

Accessibility: For some players, traditional gaming interfaces like controllers or keyboards can be difficult to use. Tracking technology can provide alternative control mechanisms that are more accessible for players with different abilities or needs.

Overall, the use of trackers in interactive gaming environments can provide a range of benefits, from improving gameplay and immersion to promoting health and accessibility.

## 12. WORK DONE:

There has been significant work done in the development of interactive gaming experiences that use trackers to enable human movements. Here are some examples of the work that has been done in this field:

a. Development of tracking technologies: There have been ongoing advancements in tracking technologies, including the development of more precise and responsive motion capture systems, camera-based tracking systems, and wearable sensors. These technologies enable the detection and interpretation of human movements in real time, which is essential for creating immersive and engaging gaming experiences.

b. Game development: Many game developers have been exploring the use of trackers to enable human movements in games. Some notable examples include the Nintendo Wii, which utilized motion controls to enable players to physically interact with the game, and the Xbox Kinect, which used camera-based tracking to detect and interpret human movements.

c. Research on health and fitness benefits: There has been research conducted on the health and fitness benefits of interactive gaming experiences that use trackers to enable human movements. This research has found that some games can provide a fun and engaging way to encourage physical activity and exercise, which can have positive impacts on health and wellness.

d. Innovation in user interface design: The development of interactive gaming experiences that use trackers to enable human movements has required innovation in user interface design. Developers have had to create interfaces that provide feedback to players and enable precise control of in-game avatars, which can be challenging given the complex and varied nature of human movements.

e. Adoption of virtual and augmented reality: The

adoption of virtual and augmented reality technologies has also enabled the development of interactive gaming experiences that use trackers to enable human movements. These technologies provide a more immersive and engaging experience by enabling players to interact with a virtual environment in a more natural and intuitive way.

Overall, there has been significant work done in the development of interactive gaming experiences that use trackers to enable human movements. Ongoing research and innovation in this field is likely to continue to push the boundaries of what is possible and create new and exciting gaming experiences.

## 13. HEURISTIC EVALUATION:

**Jakob Nielsen's 10 heuristic evaluation metrics are:**

Visibility of system status
Match between system and the real world
User control and freedom
Consistency and standards
Error prevention
Recognition rather than recall
Flexibility and efficiency of use
Aesthetic and minimalist design
Help users recognize, diagnose, and recover from errors
Help and documentation

**Schneiderman's 8 golden rules are:**

Strive for consistency
Enable frequent users to use shortcuts
Offer informative feedback
Design dialogs to yield closure
Offer simple error handling
Permit easy reversal of actions
Support internal locus of control
Reduce short-term memory load

Based on the given information, here is a heuristic evaluation

**Visibility of system status:**
The system status is not clearly visible to the user. There should be a visual cue to show whether the software is ready to use or not.

**Match between system and the real world:**
The hand gestures used to control the games should be intuitive and easy to understand. The hand gestures should match the actions performed in the game.

**User control and freedom:**
The user should have complete control over the system. The user should be able to pause or stop the system at any time.

**Consistency and standards:**
The system should be consistent in its design and behavior. The hand gestures used in the game should be consistent throughout the game.

**Error prevention:**
The system should prevent errors as much as possible. The user should not be able to accidentally perform an action that they did not intend to perform.

**Recognition rather than recall:**
The hand gestures used in the game should be easy to recognize and remember. The user should not have to recall a lot of information to use the system

**Flexibility and efficiency of use:**
The system should be flexible and efficient to use. The user should be able to easily switch between different games.

**Aesthetic and minimalist design:**
The system should have an aesthetic and minimalist design. The hand gestures used in the game should be easy to understand and should not clutter the screen.

**Help users recognize, diagnose, and recover from errors:**

The system should provide clear feedback when an error occurs. The user should be able to easily diagnose and recover from errors.

**Help and documentation:**

The system should have help and documentation available to the user. The user should be able to easily access this information when needed.

**Based on Schneiderman's 8 golden rules:**

**Strive for consistency:**

The hand gestures used in the game should be consistent throughout the game.

**Enable frequent users to use shortcuts:**

There should be shortcuts available for frequently used actions.

**Offer informative feedback:**

The system should provide informative feedback to the user.

**Design dialogs to yield closure:**

The system should provide clear and concise instructions to the user.

Offer simple error handling: The system should have simple and effective error handling.

**Permit easy reversal of actions:**

The user should be able to easily undo an action if they make a mistake.

**Support internal locus of control:**

The user should feel in control of the system. Reduce short-term memory load: The hand gestures used in the game should be easy to remember and should not require the user to remember a lot of information.

## 14. TESTING:

| Heuristic Evaluation Metrics | Evaluation | Suggestions for Improvement |
|---|---|---|
| **Nielsen's Heuristics** | | |
| Visibility of System Status | Good | N/A |
| Match between System and the Real World | Good | N/A |
| User Control and Freedom | Good | N/A |
| Consistency and Standards | Needs Improvement | Consistent naming conventions should be used in the code, especially for function and variable names. |
| Error Prevention | Needs Improvement | Error messages or feedback should be provided to the user in case of any errors. |
| Recognition rather than Recall | Needs Improvement | Provide visual cues or help documentation to assist the user in remembering different hand gestures. |
| Flexibility and Efficiency of Use | Good | N/A |

| Heuristic Evaluation Metrics | Evaluation | Suggestions for Improvement |
|---|---|---|
| **Nielsen's Heuristics** | | |
| Visibility of System Status | Good | N/A |
| Match between System and the Real World | Good | N/A |
| User Control and Freedom | Good | N/A |
| Consistency and Standards | Needs Improvement | Consistent naming conventions should be used in the code, especially for function and variable names. |
| Error Prevention | Needs Improvement | Error messages or feedback should be provided to the user in case of any errors. |
| Recognition rather than Recall | Needs Improvement | Provide visual cues or help documentation to assist the user in remembering different hand gestures. |
| Flexibility and Efficiency of Use | Good | N/A |

| Heuristic Evaluation Metrics | Evaluation | Suggestions for Improvement |
|---|---|---|
| **Nielsen's Heuristics** | | |
| Visibility of System Status | Good | N/A |
| Match between System and the Real World | Good | N/A |
| User Control and Freedom | Good | N/A |
| Consistency and Standards | Needs Improvement | Consistent naming conventions should be used in the code, especially for function and variable names. |
| Error Prevention | Needs Improvement | Error messages or feedback should be provided to the user in case of any errors. |
| Recognition rather than Recall | Needs Improvement | Provide visual cues or help documentation to assist the user in remembering different hand gestures. |
| Flexibility and Efficiency of Use | Good | N/A |

## 15. IMPLEMENTATION:

a. The code combines three scripts that use computer vision and hand tracking techniques to perform hand gesture recognition and control mouse and keyboard functions.

b. OpenCV, Numpy, datetime, Mouse, Time, HandTrackingModule, pyautogui, pydirectinput, win32api, and win32con are only a few of the required libraries that are imported by the code.

c. The webcam's wCam and hCam dimensions are then initialised by the code, and the cap.set() method is used to configure the camera to capture the dimensions.

d. Five fingertip IDs are included in the tipIds list and utilised later in the code.

e. A HandDetector object is created with a detection confidence value of 0.75 and a maximum of 1 hand to detect.

f. The cap.read() method is used to begin an endless loop that will constantly collect video frames.

g. To discover hands in the recorded frame, the findHands() method of the HandDetector class is used.

h. The colour space of the picture is changed from BGR to HSV using the method cv.cvtColor(). The picture is then

thresholded to identify the red colour using the cv.inRange() method.

a. The contours of the red colour in the picture are located using the cv.findContours() method. The contour is skipped if its area is smaller than 50 pixels.

b. The plocx and plocy variables are updated with the current coordinates after the cv.boundingRect() method calculates the centroid of the contour.

c. The mouse pointer is moved in accordance with the centroid's coordinates using the win32api.mouse_event() method.

d. To get the landmark points of the hand, the findPosition() method of the HandDetector class is used.

e. The code checks to see whether the landmark list is not empty, and if it is, it compares the finger's present location to its prior position to determine the condition of the fingers.

f. The "w" key is pushed if the thumb and first finger are together. The left mouse button is pushed using the mouse.press() method if the thumb is closed. The pn.keyUp() and mouse.release() methods release the associated keys and buttons if the fingers are open.

g. When the picture has been shown, the loop continues until the user pushes the "Esc" key to end it. This is done using the cv.imshow() method.

h. Using the cap.release() method, the camera is released, and the cv.destroyAllWindows() function destroys all of the windows.

## 16. SOFTWARE REQUIREMENTS:

The software requirements for an interactive gaming experience that uses trackers to enable human movements can vary depending on the specific implementation of the technology. However, some general software

requirements may include:

1. Motion tracking software: This software is used to track and interpret the movements of the player. Depending on the tracking technology used, this software may be provided by the manufacturer of the tracking system or may be developed in-house by the game developers.

2. Game engine: The game engine is the software that drives the game itself, providing the underlying framework for the gameplay mechanics, user interface, and graphics. Examples of popular game engines include Unity, Unreal Engine, and CryEngine.

3. User interface design tools: Designing an effective user interface for an interactive gaming experience that uses trackers can be complex. Game developers may use specialized software tools, such as Adobe XD or Sketch, to design and prototype user interfaces for the game.

4. Audio and visual software: To create a high-quality gaming experience, developers may use specialized software tools for audio and visual design. For example, Adobe Audition or Audacity can be used for audio editing, and Adobe Photoshop or GIMP can be used for image editing.

5. Network programming tools: If the interactive gaming experience involves multiplayer functionality, developers will need to use network programming tools to enable communication between players over the internet. Examples of such tools include Unity Multiplayer or Photon.

6. Analytics tools: To analyze player behavior and performance, developers may use specialized software tools for data analytics. Examples of such tools include Google Analytics or Mixpanel.

## 17. TOOLS USED:

a. OpenCV (cv2): OpenCV is a computer vision library that offers several tools for processing images and videos. The given code makes use of OpenCV for image processing (color conversion, thresholding, contour detection), sketching on pictures, and video capture from a camera.

b. A Python library for numerical computation is called NumPy (np). It is often used in the code while manipulating arrays and performing mathematical computations.

c. Time: The Python time module offers a number of time-related functions. It is used in the code to measure the time it takes for certain actions to complete and to introduce delays.

d. Mouse (m): The mouse module offers features for computer-based simulation of mouse events, such as clicks and movement. The mouse module is used in the given code to imitate mouse clicks.

e. Using the Mediapipe library, the HandTrackingModule custom module allows for the detection and tracking of hands in pictures and videos.

f. Python module PyAutoGUI (pg) is used to automate GUI interactions. The given code uses PyAutoGUI to mimic computer keyboard events, such as key presses.

g. PyDirectInput can simulate keyboard and mouse inputs on a Windows PC. It is used to replicate keyboard events in the specified code. (e.g., pressing a key).

h. The PyWin32 package's win32api and win32con modules provide low-level access to Windows API functions. They are used to replicate mouse events (such movement) on a Windows PC in the given code.

## 18. RESULT:
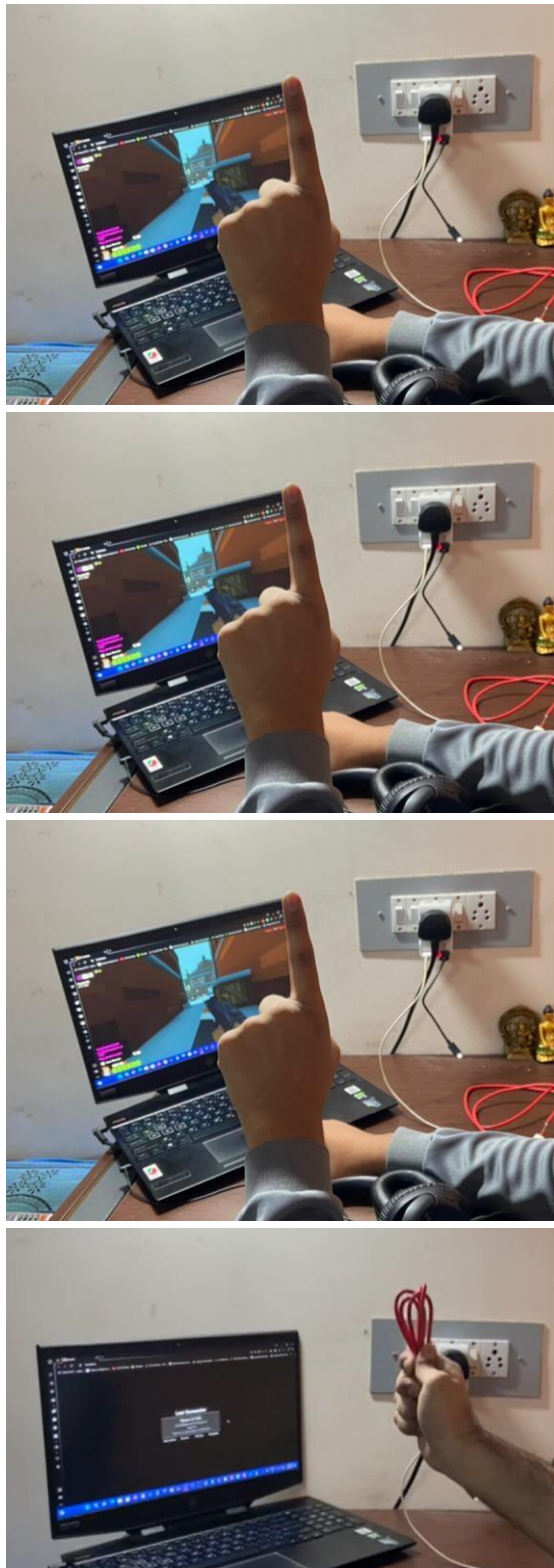


## 19. PERFORMANCE GRAPH:
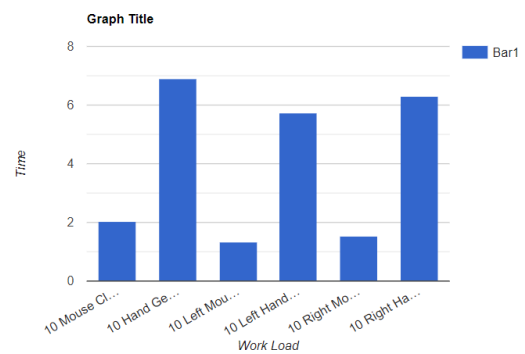


## 20. CONFIRMATION OF AUTHORSHIPS:

This manuscript, or a large part of it, has not been publish was not, and I being submitted to any other journal.

If presented at or submitted to or published at a confer- ence(s). The conference(s) is (are) identified and substantial jus- tification for re-publication is presented below.

A copy of conference paper(s) is(are) uploaded with the manuscript.

If the manuscript appears as a preprint anywhere on the web, e.g. arXiv, etc. It is identified below. The preprint should include a statement that the paper is under consideration at Pattern Recog- nition Letters.

All text and graphics, except for those marked with sources, are original works of the authors, and all necessary permissions for publication were secured prior to submission of the manuscript.

All authors each made a significant contribution to theresearch reported and have read and approved the submitted manuscript.

## 21. DECLARATION OF COMPLETING INTEREST:

The authors declare that they have no known competing finan- cial interests or personal relationships that could have appeared to influence the work reported in this paper.

## 22. REFERENCES:

**"Adaptive Difficulty in Games: A Survey" by Mark J. Nelson and James C. Lester (2013)** - **https://ieeexplore.ieee.org/document/6636239**

"A Review of HCI Research in the Context of Video Games" by Elisa Mekler, Antti

Oulasvirta, and Lennart E. Nacke (2017) - https://dl.acm.org/doi/10.1145/3130859.3131434

"Design Guidelines for Co-Located, Multiplayer, Gesture-Based Games" by Laura Anna Ripamonti, Marcello Porta, and Massimo Zancanaro (2014) - https://dl.acm.org/doi/10.1145/2559636.2559642

"HCI Issues in computer games" by PanayiotisZaphiris and Chee Siang Ang (2007) - https://www.researchgate.net/publication/40888045_HCI_issu es_in_computer_games

"Real Time Hand Gesture Movements Tracking and Recognizing System" by Rudy Hartanto, Adhi Susanto, P. Insap Santosa (2014) - https://ieeexplore.ieee.org/document/7003734

• https://developers.google.com/mediapipe/solutions/gu

ide

https://docs.opencv.org/4.x/d9/df8/tutorial_root.html

| Date | April 11, 2023 |
| --- | --- |
| Exclude URL: | NO |

| | Unique Content | 100 | | Word Count | 2582 |
| --- | --- | --- | --- | --- | --- |
| | Plagiarized Content | 0 | | Records Found | 0 |

## CONTENT CHECKED FOR PLAGIARISM:

GAME-N-GEEK

INTERACTIVE GAMING PORTAL

PROJECT REPORT

Submitted by:

SI NO REG.NO NAME

1 20BCE2515 Parth Borana

2 20BCE2537 Vadarevu Sai Sampreet Khushal

3 20BCE2772 Saimrika Gupta

4 20BCE2806 Ashmit Gupta

PREPARED FOR:

HUMAN COMPUTER INTERACTION (CSE 4015)

PROJECT COMPONENT

UNDER THE GUIDANCE OF:

Prof. Swarnalatha P.

Professor Grade 1

School of computer Science and Engineering

ACKNOWLEDGEMENTS:

We sincerely thank our Chancellor - Dr. G. Viswanathan, VIT University, for giving us the opportunity to

pursue this course of Human Computer Interaction and introduce J-component as part of our academic

curriculum. We also thank PROF. Dr. Swarnalatha P. for giving us the opportunity to do this project. It was a wonderful experience working on this project which made us aware of various tactics and strategies related to human and computer interaction that might help us in the longer run.

PROBLEM STATEMENT/ ABSTRACT:

Most of the population, especially, elderly people are interested in computer gaming technology and desire to be actively involved in such activities, however they are limited by their responsiveness due to age, as gaming requires fast reflexes in order to provide and comfortable / satisfactory experience. People who are not very knowledgeable about gaming are also limited to enter the gaming world. This upbrings a huge demand in the current market for a software / product which can enable users who are new to the gaming world to be able to comfortably enter / transition into the gaming world or be able to enjoy the merits of gaming without their limitations stopping

AIM AND SCOPE:

To provide gamers and people of interest in the gaming field with a unique and mesmerizing experience which can bring a shift toward human movement computer gaming.

To provide a software / platform for real time visual interaction with the system to ensure the complete gaming experience.

To introduce people with limited knowledge of gaming to the gaming world and to ensure a comfortable experience as well as a smooth transition into the gaming technology.

INTRODUCTION:

Game-N-Geek allows for interactive gaming experience which uses tracker to be able to interact in gaming environments using human movements.

Game-N-Geek appears to be a platform that utilizes tracking technology to allow for interactive gaming experiences that incorporate human movements. This technology can provide a more immersive and engaging gaming experience by allowing players to physically interact with the game environment using their body movements.

There are several types of tracking technology that can be used for this purpose, including motion capture systems, camera-based tracking, and wearable sensors. These technologies can detect and interpret a player's movements, allowing them to control their in-game avatar in real time.

Overall, the use of tracking technology in gaming has the potential to revolutionize the way we play and experience games. By incorporating human movements into the gameplay, we can create more immersive and interactive experiences that are both fun and engaging.

Game-N-Geek is a program that allows users to play any PC related video game using their hand signs and movements instead of using mouse and keyboard. It allows users to play games using hand signs.

Game-N-Geek allows for a smooth gaming experience without the inclusion of hands and can be used to interact within the gaming environment entirely in a visual manner.

LITERATURE SURVEY:

Literature Survey 1: Real Time Hand Gesture Movements Tracking and Recognizing System

This paper proposes the use of contour detection, fingers movement detection, motion detection and segmentation, and applying these as a mouse functionality to be able to use the hand as a mouse which can interact with the given system.

Literature Survey 2: HCI Issues in Computer Games

This paper discusses about the various issues which are brought up during HCI involvement in computer games, this includes factors such as: Theoretical, Methodological, Technical, and social Cultural aspects. It also brings to light the issue of the familiarity of many users with the QWERTY keyboard style which may hinder their performance/ Experience with the game when they have to use HCI related gadgets, gaming tools to perform the same activities.

Literature Survey 3: Hand landmarks detection guide

You can identify the hands' landmarks in a picture using the Media Pipe Hand Land marker task. This task can be used to localize the hands' important areas and render visual effects on top of the hands. This task generates hand landmarks in image coordinates, hand landmarks in world coordinates, and handedness (left/right hand) of numerous detected hands using image data and a machine learning (ML) model as static data or a continuous stream.

REQUIREMENT ANALYSIS:

right255270

UNIQUENESS:

The project focuses on helping the people with limited gaming knowledge be able to have a good experience with gaming and to be able to interact with it using hand movements instead of pressing or clicking buttons on a keyboard/mouse.

It also increases interactivity and attention span during gaming as it includes the involvement of real time hand movements which help perform tasks within the game which will further user experience.

METHODOLOGY:

Import required libraries including OpenCV, datetime, numpy, time, pyautogui, pydirect input, win32api and Hand Tracking Module.

Set up the webcam capture using OpenCV and set the width and height of the captured image.

Create a Hand Detector object from Hand Tracking Module to detect hand gestures and movements in the captured image.

Start an infinite loop to continuously capture images from the webcam.

Read the image frame and flip it horizontally to mirror the movement of the user's hand.

Convert the image frame to HSV format to detect red colour in the image.

Define the lower and upper bounds for the red colour range, and create a mask to filter out any other colour.

Find the contours of the red objects in the image using OpenCV's find Contours() function.

For each contour found, check if its area is greater than a threshold value. If so, calculate the centroid of the contour and move the mouse cursor accordingly using win32api's mouse event() function. Also, draw a green contour around the red object.

Use the Hand Detector object to detect hand gestures in the image.

If any hand gestures are detected, find the positions of the fingertips and calculate the state of the fingers based on their positions.

Use the state of the fingers to control the keyboard and mouse inputs using pyautogui and pydirectinput functions.

Display the image frame with any contours drawn and wait for a key event to exit the loop.

Release the capture object and exit the program.

OBJECTIVES:

The objectives of using trackers to enable interactive gaming experiences that incorporate human movements can vary depending on the specific context and goals of the game. However, some common objectives of using this technology include:

Increased immersion: By allowing players to physically interact with the game environment using their body movements, tracking technology can create a more immersive and engaging gaming experience.

Improved gameplay: Using tracking technology can provide more precise and responsive control mechanisms, allowing players to perform more complex and nuanced actions in the game.

Health and fitness benefits: Some games that utilize tracking technology are designed to encourage physical activity and exercise, which can have health and fitness benefits for players.

Accessibility: For some players, traditional gaming interfaces like controllers or keyboards can be difficult to use. Tracking technology can provide alternative control mechanisms that are more accessible for players with different abilities or needs.

Overall, the use of trackers in interactive gaming environments can provide a range of benefits, from improving gameplay and immersion to promoting health and accessibility.

INNOVATION COMPONENT IN THE PROJECT:

The innovation components of interactive gaming experiences that use trackers to enable human

movements can include:

1. Tracking technology: One of the key components of this innovation is the tracking technology that enables the detection and interpretation of human movements. This can include motion capture systems, camera-based tracking, and wearable sensors.

2. User interface design: The design of the user interface is critical in creating an intuitive and responsive experience for players. The interface must be designed to provide feedback and allow for precise control of the in-game avatar.

3. Game design: The design of the game must be tailored to take advantage of the unique capabilities of the tracking technology. This may involve creating gameplay mechanics that utilize specific types of movement or designing game environments that encourage physical activity and exercise.

4. Integration with other technologies: Interactive gaming experiences that use trackers may also incorporate other technologies such as virtual reality, augmented reality, or artificial intelligence to create a more immersive and engaging experience.

Data analytics: Collecting and analyzing data on player performance and behavior can provide insights into how to improve the game design and user interface. This can lead to iterative improvements and a more refined user experience.

The innovation components of interactive gaming experiences that use trackers to enable human movements can include:

6. This can include motion capture systems, camera-based tracking, and wearable sensors.

7. The interface must be designed to provide feedback and allow for precise control of the in-game avatar.

8. This may involve creating gameplay mechanics that utilize specific types of movement or designing game environments that encourage physical activity and exercise.

Integration with other technologies: Interactive gaming experiences that use trackers may also incorporate other technologies such

Overall, the innovation components of interactive gaming experiences that use trackers to enable human movements involve a combination of tracking technology, user interface design, game design, integration with other technologies, and data analytics.

WORK DONE:

There has been significant work done in the development of interactive gaming experiences that use trackers to enable human movements. Here are some examples of the work that has been done in this field:

1. Development of tracking technologies: There have been ongoing advancements in tracking

technologies, including the development of more precise and responsive motion capture systems, camera-based tracking systems, and wearable sensors. These technologies enable the detection and interpretation of human movements in real time, which is essential for creating immersive and engaging gaming experiences.

2. Game development: Many game developers have been exploring the use of trackers to enable human movements in games. Some notable examples include the Nintendo Wii, which utilized motion controls to enable players to physically interact with the game, and the Xbox Kinect, which used camera-based tracking to detect and interpret human movements.

3. Research on health and fitness benefits: There has been research conducted on the health and fitness benefits of interactive gaming experiences that use trackers to enable human movements. This research has found that some games can provide a fun and engaging way to encourage physical activity and exercise, which can have positive impacts on health and wellness.

4. Innovation in user interface design: The development of interactive gaming experiences that use trackers to enable human movements has required innovation in user interface design. Developers have had to create interfaces that provide feedback to players and enable precise control of in-game avatars, which can be challenging given the complex and varied nature of human movements.

5. Adoption of virtual and augmented reality: The adoption of virtual and augmented reality technologies has also enabled the development of interactive gaming experiences that use trackers to enable human movements. These technologies provide a more immersive and engaging experience by enabling players to interact with a virtual environment in a more natural and intuitive way.

Overall, there has been significant work done in the development of interactive gaming experiences that use trackers to enable human movements. Ongoing research and innovation in this field is likely to continue to push the boundaries of what is possible and create new and exciting gaming experiences.

IMPLEMENTATION:

SOFTWARE REQUIREMENTS:

The software requirements for an interactive gaming experience that uses trackers to enable human movements can vary depending on the specific implementation of the technology. However, some general software requirements may include:

1. Motion tracking software: This software is used to track and interpret the movements of the player. Depending on the tracking technology used, this software may be provided by the manufacturer of the tracking system or may be developed in-house by the game developers.

2. Game engine: The game engine is the software that drives the game itself, providing the underlying framework for the gameplay mechanics, user interface, and graphics. Examples of popular game

engines include Unity, Unreal Engine, and CryEngine.

3. User interface design tools: Designing an effective user interface for an interactive gaming experience that uses trackers can be complex. Game developers may use specialized software tools, such as Adobe XD or Sketch, to design and prototype user interfaces for the game.

4. Audio and visual software: To create a high-quality gaming experience, developers may use specialized software tools for audio and visual design. For example, Adobe Audition or Audacity can be used for audio editing, and Adobe Photoshop or GIMP can be used for image editing.

5. Network programming tools: If the interactive gaming experience involves multiplayer functionality, developers will need to use network programming tools to enable communication between players over the internet. Examples of such tools include Unity Multiplayer or Photon.

6. Analytics tools: To analyze player behavior and performance, developers may use specialized software tools for data analytics. Examples of such tools include Google Analytics or Mixpanel.

TOOLs USED:

CODE:

RESULT:

References:

Ford-Fulkerson Algorithm:

Leicht, E. A., & Newman, M. E. (2008). Community structure in directed networks. Physical Review Letters, 100(11), 118703.

Girvan, M., & Newman, M. E. (2002). Community structure in social and biological networks. Proceedings of the National Academy of Sciences, 99(12), 7821-7826.

Leskovec, J., Lang, K. J., Dasgupta, A., & Mahoney, M. W. (2008). Statistical properties of community structure in large social and information networks. Proceedings of the 17th international conference on World Wide Web, 695- 704.

Park, J., & Barabási, A. L. (2007). Distribution of node characteristics in complex networks. Proceedings of the National Academy of Sciences, 104(46), 17916- 17920.

Clauset, A., Newman, M. E., & Moore, C. (2004). Finding community structure in very large networks. Physical Review E, 70(6), 066111.

Karger Min Cut Algorithm:

Girvan, M., & Newman, M. E. (2002). Proceedings of the National Academy of Sciences, 99(12), 7821-7826.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8), 888-905.

Brandes, U., Delling, D., Gaertler, M., Goerke, R., Hoefer, M., Nikoloski, Z., & Wagner, D. (2007). On modularity clustering. IEEE Transactions on Knowledge and Data Engineering, 20(2), 172-188.

Chen, Y., & Chen, G. (2009). Community detection via maximal cliques in graphs. Journal of Combinatorial Optimization, 18(2), 135-149.

Cho, Y. S., Kim, S., & Kim, J. H. (2007). Finding bottlenecks in sparse graphs. Theoretical Computer Science, 380(1), 81-94.

## MATCHED SOURCES:

GOODLE DRIVE LINK FOR VIDEO DEMONSTRATION

https://drive.google.com/drive/u/0/folders/1E3HaJ01iX6HfVupcmUx0OtPeO7LWVEjF