|  |  |
| --- | --- |
| **Index** | **Page No.** |
| Cover Page | **1** |
| DECLARATION | **2** |
| Acknowledgment | **3** |
| Problem Statement | **5** |
| Objective | **6** |
| Introduction | **7** |
| Libraries Used | **8** |
| Design And Architecture | **16** |
| Software Components | **17** |
| Architecture Workflow | **18** |
| Implementation | **19** |
| Proposed System | **23** |
| System Design | **24** |
| Sample Examples | **25** |
| Drawbacks | **29** |
| Conclusion | **30** |
| Future references | **31** |

**Hand Gesture Based Snake Game**

**Problem Statement:**

In the digital gaming landscape, traditional control mechanisms can sometimes limit user engagement and experience. The challenge addressed by this project is to design, implement, and optimize a Hand Gesture-Based Snake Game. The primary focus is on creating an accurate and responsive system that can recognize a variety of hand gestures, translating them into seamless control of the snake in real-time. This involves overcoming technical challenges related to computer vision, gesture algorithms, and integration with game development

This project aims to develop an innovative and interactive version of the classic Snake game by implementing hand gesture controls. Unlike traditional input methods, such as keyboard or touchscreen, this project will leverage computer vision and gesture recognition technologies to allow users to control the snake's movements through hand gestures captured by a camera or a similar sensing device

**OBJECTIVE**

The goal of the game is to eat as many as the food item as possible as a player. The snake grows larger by consuming more and more food items. In this player can control the movement of the snake by using index finger of the hand using Computer Vision (AI). The snake moves as player start moving hand freely.

**Introduction**

The Snake Game Using Hand Tracking/Gesture Recognition is Game name where the tips of a particular hand of the human is detected and used to manipulate the movements of the snake which is detected through the web cam or the camera of the system. It is the concern of the user to avoid t the snake to intersect itself where the snake head touches the remaining body of the snake and should no stop the hand movement while playing the game if it comes off then the game is over To director the user or the player about the game or the projection of the body of the snake I used a green colored round shape as a pointer for the head of the snake in the game and green color line surrounded by the red limitations. Hands are used for controlling the game-based interfaces for snake games as an alternative for using keys of the keyboard or touching a mouse. In this game has you continue playing you get more and more food items which the player has to eat white playing and keeps growing the body of the snake and the game is getting difficult to play the game.

**CV2** and **CVZONE** are the libraries used for the development of the game. This project is fragment of the field of hand recognition through the camera, which is booming in gaming industries and gaining more popularity because it allows users to gain best gaming experience while playing the game.

**LIBRARIES USED:**

**Random**

A "random" library in programming typically refers to a module or library that provides functions related to generating pseudorandom numbers. Pseudorandom numbers are not truly random but are generated using a deterministic algorithm. The "random" library is commonly used in programming languages like Python and Java to introduce unpredictability or randomness into programs. It allows developers to create applications that involve random selection, shuffling, or simulations.

**Math library**

In [computer science](https://en.wikipedia.org/wiki/Computer_science" \o "Computer science), a **math library** (or maths library) is a component of a [programming language](https://en.wikipedia.org/wiki/Programming_language" \o "Programming language)'s [standard library](https://en.wikipedia.org/wiki/Standard_library" \o "Standard library) containing [functions](https://en.wikipedia.org/wiki/Subroutine" \o "Subroutine) (or [subroutines](https://en.wikipedia.org/wiki/Subroutine" \o "Subroutine)) for the most common [mathematical functions](https://en.wikipedia.org/wiki/Mathematical_function" \o "Mathematical function), such as [trigonometry](https://en.wikipedia.org/wiki/Trigonometry" \o "Trigonometry) and [exponentiation](https://en.wikipedia.org/wiki/Exponentiation" \o "Exponentiation). Bit-twiddling and control functionalities related to [floating point numbers](https://en.wikipedia.org/wiki/Floating_point_numbers" \o "Floating point numbers) may also be included (such as in C).

**CV ZONE**

**CV as Computer Vision:**

Computer Vision (CV) is a field of artificial intelligence (AI) and computer science that focuses on enabling computers to interpret and understand visual information from the world. It involves the development of algorithms and systems that allow machines to analyze and make decisions based on visual data, such as images or videos. Computer Vision has applications in various domains, including image and video recognition, object detection, facial recognition, autonomous vehicles, and medical image analysis.It is a computer vision package that makes it easy to run Image processing and AI functions

**Workflow:**

**Initialization:**

Initialize the webcam for capturing video frames.

Load necessary libraries for computer vision and hand tracking.

Set up the game environment, including the snake and initial conditions.

**Main Loop:**

Continuously capture video frames from the webcam.Use hand tracking to identify and track the user's hand(s).

Analyze hand gestures to control the snake's movements.

**Game Logic:**

Interpret hand gestures as commands for the snake (e.g., move left, right, up, down).

Update the game state based on user input and internal logic.

Check for collisions, update the score, and manage other game-related events.

**Display:**

Render the game environment, snake, and other elements on the screen.

Provide real-time visual feedback of the user's hand and gestures.

Update the display based on the current game state.

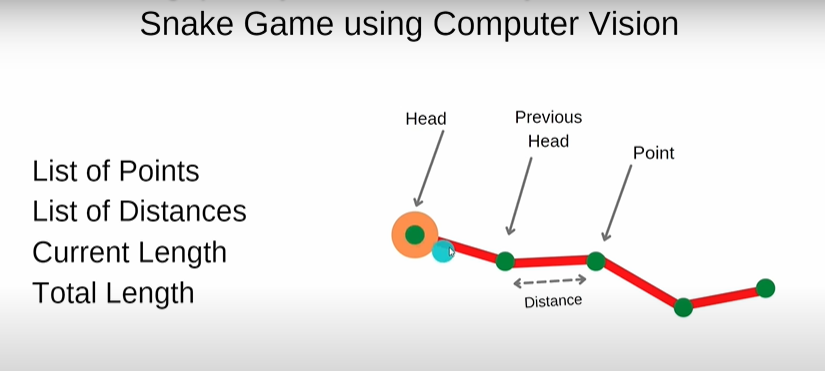
**User Interaction**:

Allow the player to control the snake through hand gestures.

Provide a responsive and intuitive interface for gesture recognition.

**Termination:**

End the game loop when a predefined condition is met (e.g., player quits or game over).

**Fig no. 1**

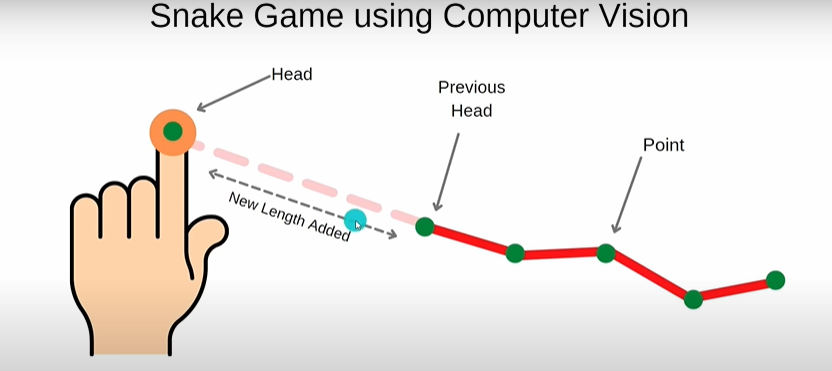


Fig no. 2

**CV2 Library**

**CV2 (OpenCV):**

cv2 is a Python library used for computer vision and image processing tasks. In the context of a Hand Gesture-Based Snake Game, you might use OpenCV to capture video frames from a camera, process these frames to detect hand gestures, and then translate those gestures into commands for controlling the snake's movements in the game.

**Hand Movement**

In this project we keep track of finger tips of the particular hands so for the free flow of the hands there is no restriction for the hands direction. When the system detects the hands, it particularly mentions hands if it is right or left hands. If our both hands are revealed towards the webcam the system detects only one hand at a time.

**Hand Module (Hand detection using MEDIAPIPE)**

**Media pipe:**

The ability to perceive the shape and motion of hands can be a vital component in improving the user experience across a variety of technological domains and platforms. For example, it can form the basis for sign language understanding and hand gesture control, and can also enable the overlay of digital content and information on top of the physical world in augmented reality. While coming naturally to people, robust real-time hand perception is a decidedly challenging computer vision task, as hands often occlude themselves or each other (e.g. finger/palm occlusions and hand shakes) and lack high contrast patterns.

MediaPipe Hands is a high-fidelity hand and finger tracking solution. It employs machine learning (ML) to infer 21 3D landmarks of a hand from just a single frame. Whereas current state-of-the-art approaches rely primarily on powerful desktop environments for inference, our method achieves real-time performance on a mobile phone, and even scales to multiple hands. We hope that providing this hand perception functionality to the wider research and development community will result in an emergence of creative use cases, stimulating new applications and new research avenues.

Shown In fig no 3, 4.



**Figno 4**



**FigNo 5**

**DESIGN AND ARCHITECTURE:**

**Hardware Components:**

**Camera:**

Use a camera to capture hand gestures. A webcam or a specialized camera like a depth-sensing camera (e.g., Kinect) could be suitable.

**LEDs or Display:**

Optionally, include LEDs or a display to provide feedback to the user or enhance the gaming experience.

**Depth Sensor (Optional):**

For more accurate hand tracking and gesture recognition, you might consider a depth sensor. Depth sensors provide additional information about the distance of objects from the camera.

Examples include the Intel RealSense Depth Camera or similar products.

**Infrared (IR) Light Source (Optional):**

Some gesture-based systems use IR light sources to enhance tracking, especially in low-light conditions. Infrared LEDs can be placed around the camera to illuminate the user's hands.

Consider integrating IR LEDs or an external IR light source if your camera lacks sufficient IR capabilities.

**Microphone (Optional):**

If you plan to include voice commands as part of the user interface, a microphone is necessary.

Use a good-quality microphone to capture clear voice commands from the user.

**Wearable Devices (Optional):**

Wearable devices, such as smart gloves or wristbands, can enhance the accuracy of hand tracking by providing additional data points.

These devices might include accelerometers, gyroscopes, and other sensors to capture hand movements more precisely.

**Computer or Embedded System:**

The processing power of the system is crucial for real-time hand tracking and gesture recognition. Use a computer with a decent CPU and GPU.

Alternatively, consider embedded systems with sufficient computing capabilities if you aim for a compact or standalone solution.

**Mounting Hardware:**

Sturdy mounts or brackets for fixing the camera and any additional sensors in a stable position.

Tripods or wall mounts can be used, depending on the setup.

**Display (Optional):**

The game display, where the snake game graphics will be rendered. This could be a monitor, television, or even a virtual reality (VR) headset, depending on your intended user experience.

**Power Supply:**

Ensure a reliable power supply for all components. If using embedded systems, consider power-efficient solutions.

**Cables and Connectors:**

High-quality cables and connectors to establish connections between the camera, sensors, and the processing unit.

**Cooling System (If using a powerful computer):**

If your system involves heavy computation, ensure proper cooling to prevent overheating.

**Networking Components (If applicable):**

For multiplayer or online functionality, consider networking components like Wi-Fi or Ethernet connectivity

**Software Components:**

**Image Processing:**

Implement an image processing algorithm to detect and track hand gestures from the camera feed. Libraries like OpenCV can be useful for this task.

**Gesture Recognition:**

Develop a gesture recognition system to interpret hand movements as commands for the snake game. Machine learning models, such as neural networks or other pattern recognition algorithms, can be trained for this purpose.

**Snake Game Logic:**

Develop the snake game using a programming language and framework of your choice. Ensure that the game logic can interpret the commands received from the gesture recognition system.

**User Interface:**

Design a user interface that displays the game and any additional information. This could include score, game over messages, and feedback based on hand gestures.

**Architecture Workflow:**

* **Capture Hand Gestures:**

The camera captures the user's hand gestures.

* **Image Processing:**

Process the camera feed to isolate and track the hand using computer vision techniques.

* **Gesture Recognition:**

Analyze the tracked hand movements to recognize specific gestures that correspond to game commands (e.g., move left, move right, speed up).

* **Snake Game Logic:**

Update the game state based on the received gestures. For example, if the user swipes left, move the snake to the left.

* **User Interface:**

Display the updated game state on the user interface, including the snake's position, score, and any relevant information.

**IMPLEMENTATION**

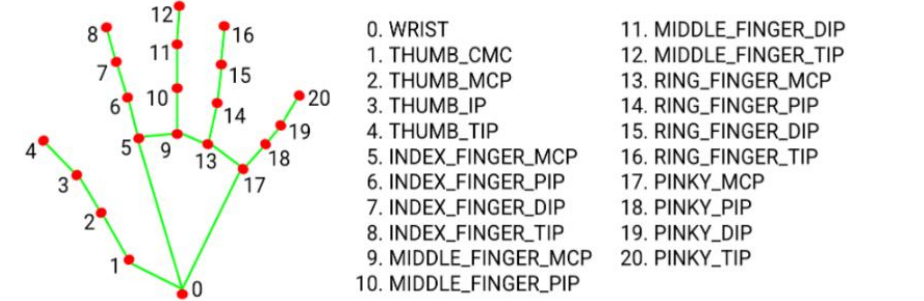
**OPENCV**

The Open-Source Computer Vision (OpenCV) programming function library was created by Intel with real-time computer vision as its primary focus. It works on various platforms. Real-time image processing is its core area of focus. OpenCV is adaptable to some particular systems, such as digital signal processors, thanks to the C interface that was originally designed for the library. To promote adoption by a larger audience, wrappers have been created for languages including C#, Python, Ruby, and Java.

**Detection**

This phase plays very important role in functioning of the game where it retrieves the information from the live camera and detects particular point of the hand where the game starts. This detection of the particular hand is achieved through hand detection module which is used for detection of hands using computer vision (OpenCV) which helps the detection of hands through live videos which is achieved using webcam.

Shown in FigNo.6



**Fig No 6 Hand Detection Point**

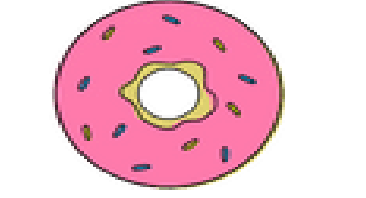
**Hand Movement**

In this project we keep track of finger tips of the particular hands so for the free flow of the hands there is no restriction for the hand’s direction. When the system detects the hands, it particularly mentions hands if it is right or left hands. If our both hands are revealed towards the webcam the system detects only one hand at a time

**Food Item Display**

In this game the snake has to eat a particular food item for scoring a point, but these food items are allocated at different coordinates of the screen where these appear randomly on the screen every time when snake eats a food item it gets one point and also gets large in size.

Shown in Fig No.7

**Fig no.7 Food items**

**Snake Projection**

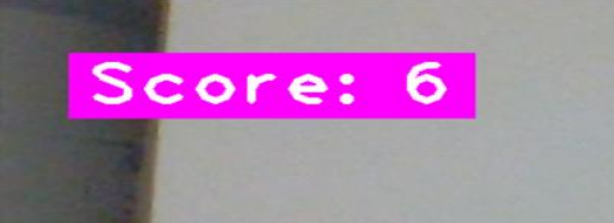
The main module after the hand detection is the “snake” in a snake game, in this project after the detection of the hands and as soon as hand movement starts there will be a green line surround by the red boundaries around the green line is the snake in the game. Shown in fig.no 8



**Fig No. 8 Snake projection**

**Game Score**

The game score is decided by the number of food item consumed by the snake where a particular food item represents one point and for every cycle the score will be added to the existing score and it will be displayed at left upper side corner of the screen. Shown in Fig No. 9

**Fig No.9 Game Score**

**PROPOSED SYSTEM**

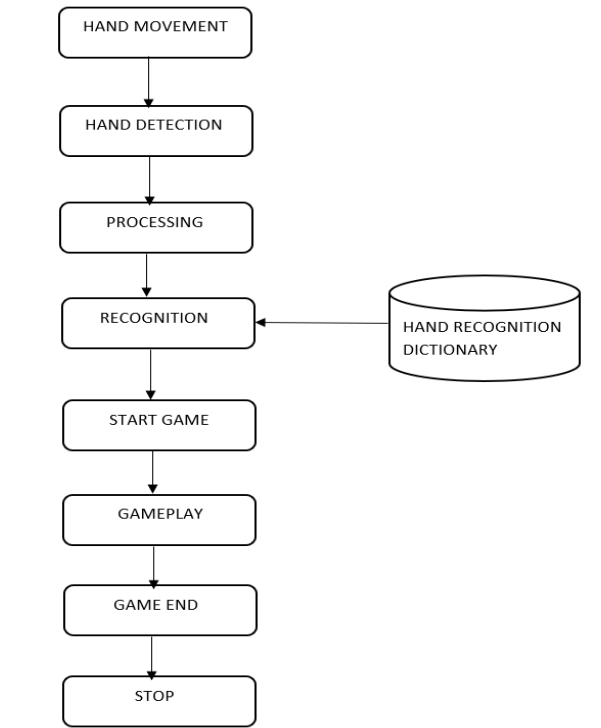
In the system we are using the hand recognition technology where t the game starts as soon as the system detects the finger tip of the particular hand and the snake movement is depended on the movement of the hands where there is no delay for on screen display of the game and after the game is over there is a separate on-screen display shoeing of the score of the user and the displaying the message of Game Over Web-Cam or the laptop front camera is a mainly used device for this project where whole game runs through open webcam where there is window showing the user face where the user must show his hand Infront of the camera and detect the tips of the particular finger. Shown in figure 10.



**Fig No 10. player setup system**

**System Design**

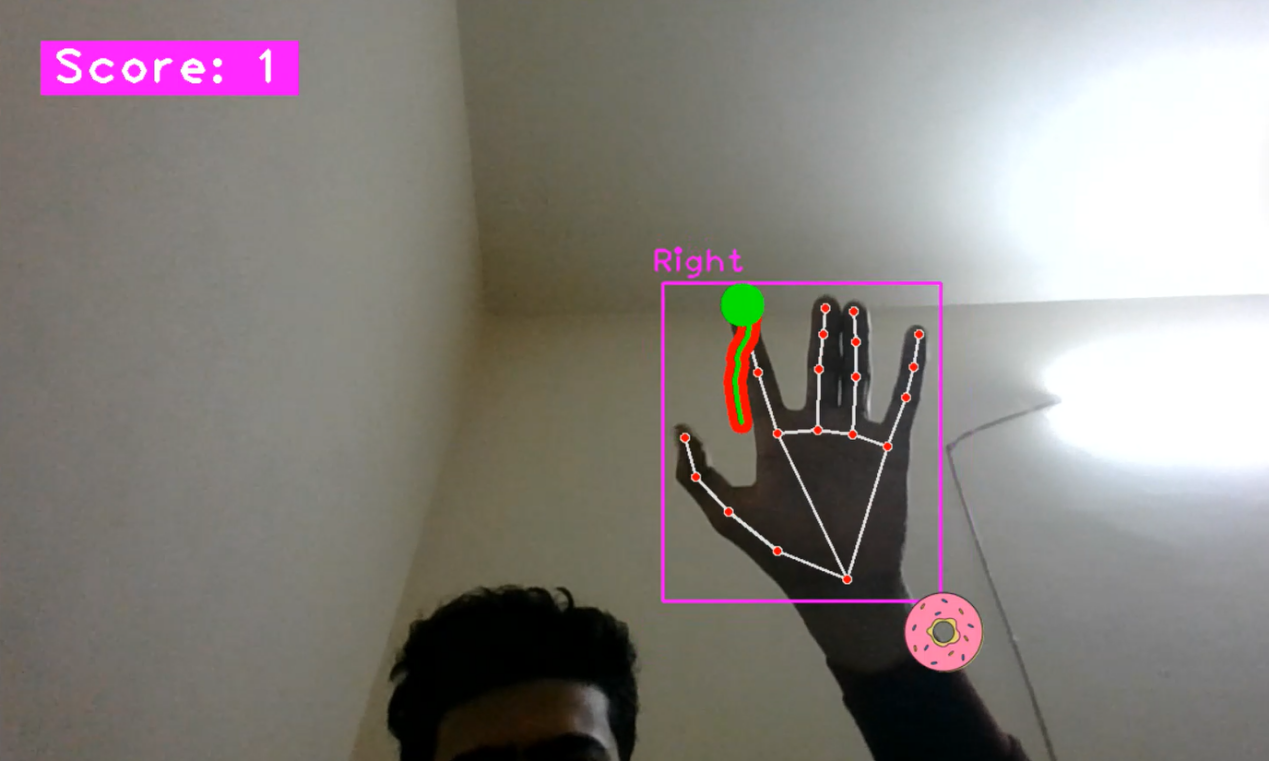
System design is used to create the system in accordance with how the project is functioning, It focus on preparing the modules and the specification which are needed for the system and also how those modules are interconnected and how the data are shared from one to another to produce the system efficiently.



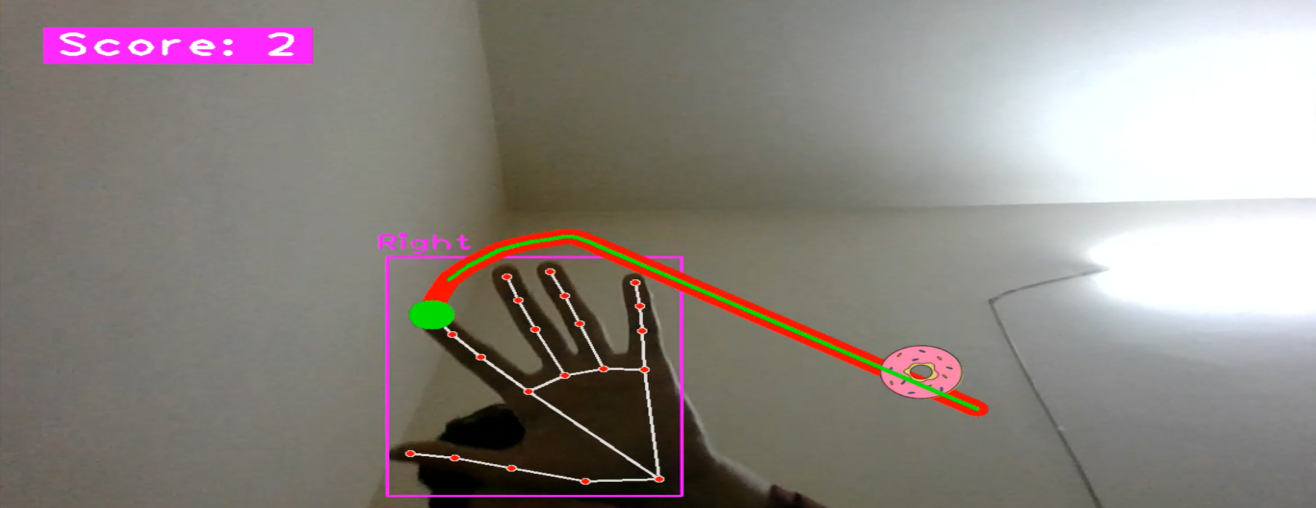
**Fig No 11. Flowchart diagram of Snake game**

**Sample Test Examples**

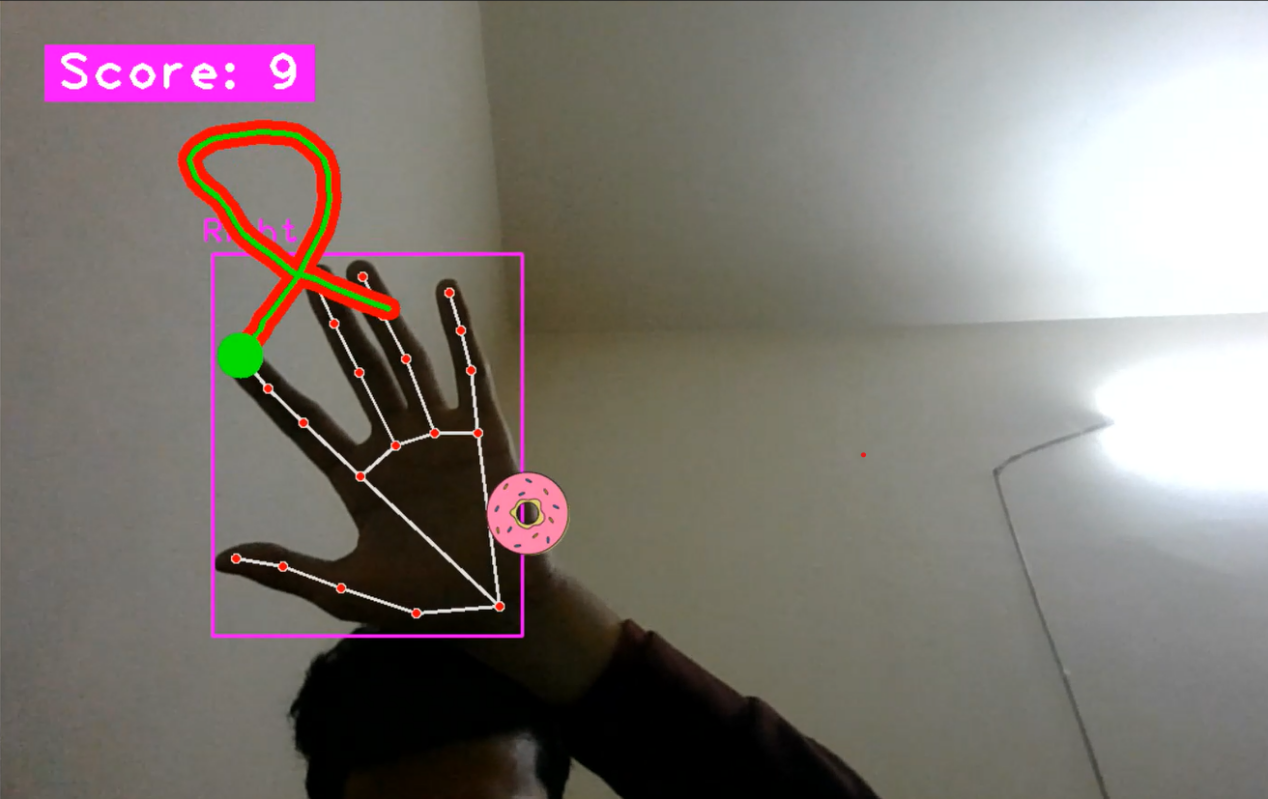
**1.Starting of the Game**



1. **When the snake consumes the donut, its length increases.This dynamic element adds an engaging aspect to the game, challenging the player to strategically navigate the snake to collect the donuts and grow in length.**

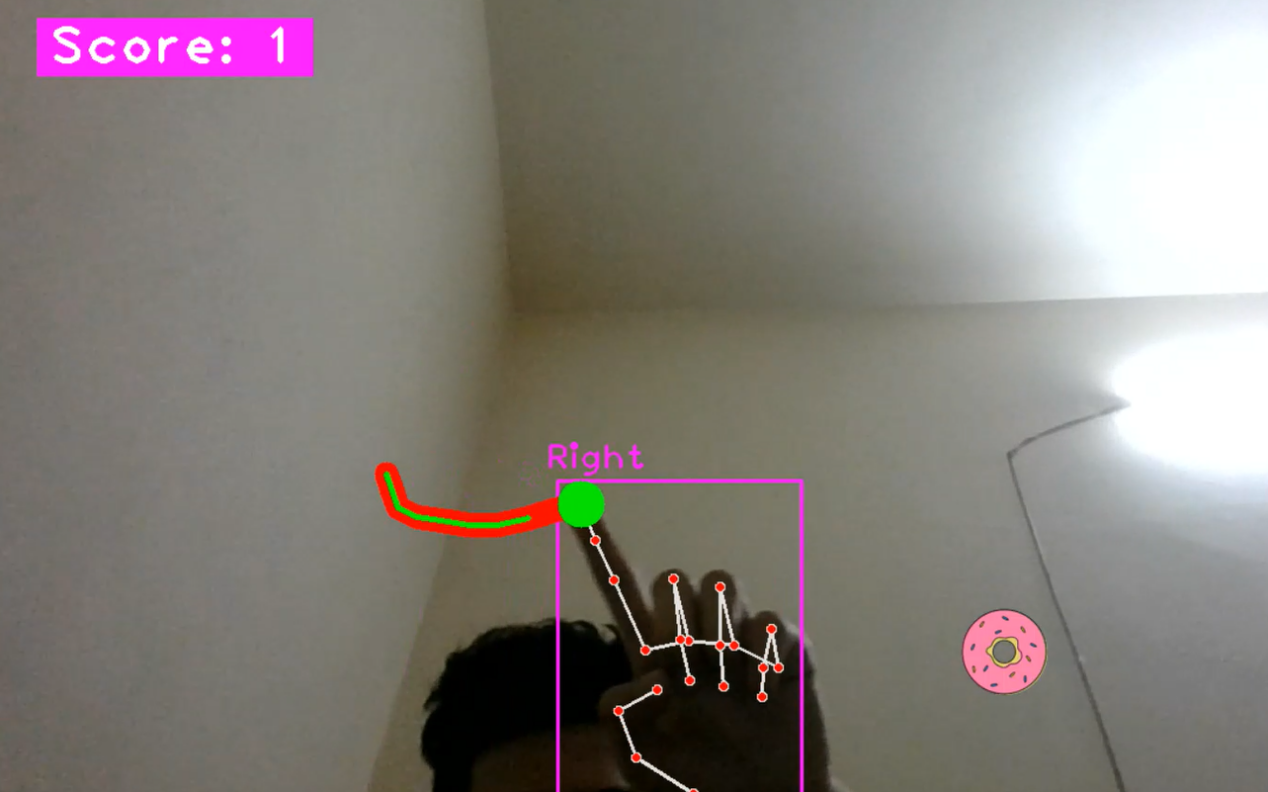


1. **Game over occurs when the snake collides itself.**

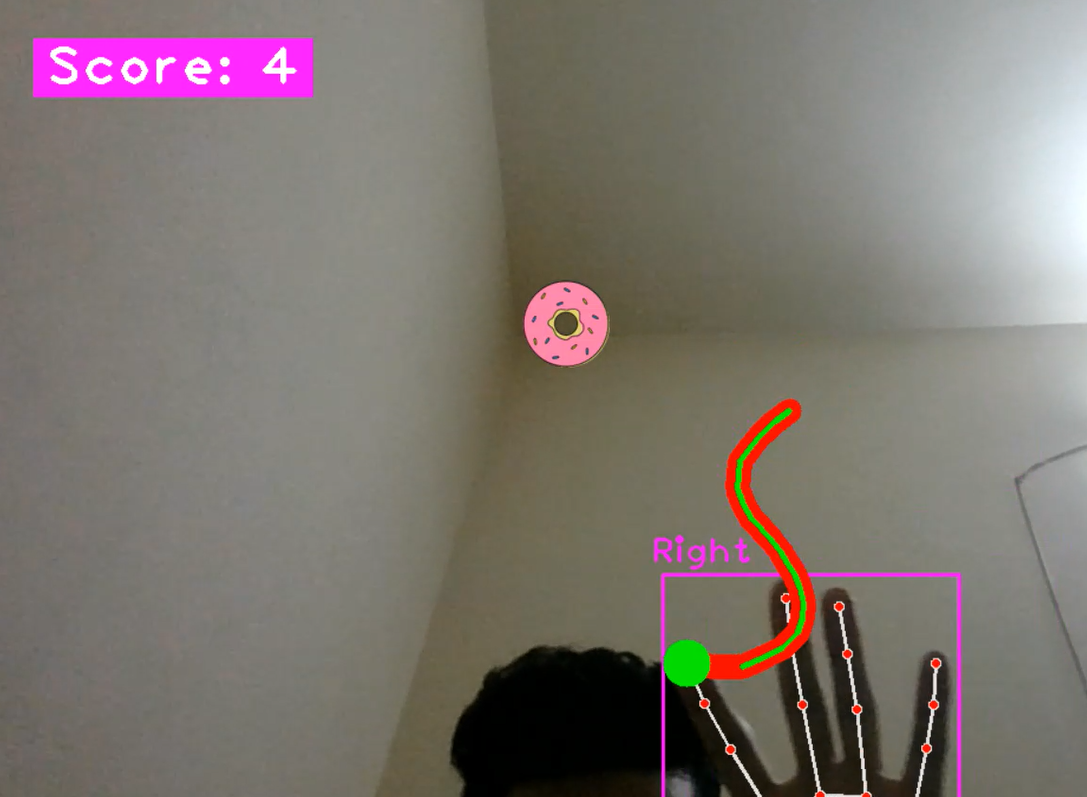


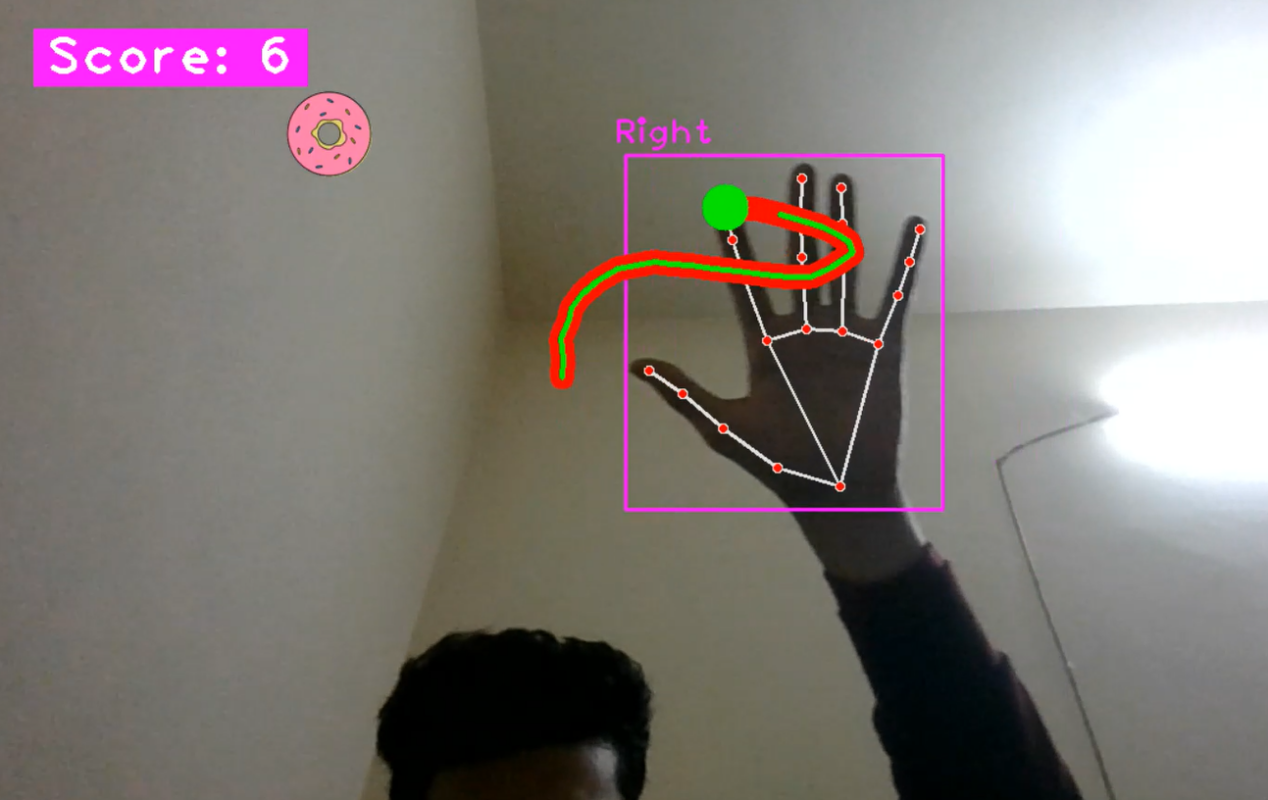


1. **The food (DONUT) apperas randomly on the screen.**



**IN this figure** The DONUT appered in the right side of the screen.

**IN this figure** The DONUT appered in the middle of the screen.



**IN this figure** The DONUT appered in the left side of the screen.

**SOME DRAWBACKS FOR EXISTING SYSTEM:**

In the existing system some of these have only the gesture control where the system can detect only the direction of the fingers of the hands and the snake moves in towards the direction of the hands without tracking the path of the snake. Some of these have pre-defined data set for the system where the system detects the hands based upon the data set given to the system and the movement of the snake is decided. And some of these are cursor based which are functioned through mouse, joy stick or touch-pad for playing the game.

**CONCLUSION**

Snake game is a computer action game, whose goal is to control a snake to move and collect food in a map. In this paper we develop a controller based on movement rating functions considering smoothness, space, and food. Scores given by these functions are aggregated by linear weighted sum, and the snake takes the action that leads to the highest score. To find a set of good weight values, we apply an evolutionary algorithm. We examine several algorithm variants of different crossover and environmental selection operators. Experimental results show that our design method is able to generate smart controllers.

**Future Direction:**

**Hand Tracking Libraries:**

Continue monitoring advancements in hand tracking libraries. Libraries like MediaPipe by Google (https://mediapipe.dev/) and OpenPose (https://github.com/CMU-Perceptual-Computing-Lab/openpose) offer sophisticated hand and pose tracking capabilities.

**Computer Vision Frameworks:**

Stay updated on computer vision frameworks that might simplify complex vision tasks. TensorFlow and PyTorch, for example, continually evolve and can be used for custom gesture recognition models.

**Gesture Recognition Models:**

Explore deep learning models for gesture recognition. Custom-trained models can improve accuracy and allow for recognizing a broader range of gestures.

**3D Hand Tracking:**

Investigate technologies that provide more advanced 3D hand tracking. This can enhance the precision and naturalness of hand movements in the game.

**Virtual Reality (VR):**

Consider exploring how hand tracking can be integrated into virtual reality environments. Technologies like Oculus Hand Tracking(https://developer.oculus.com/documentation/hand tracking/latest/) can provide a more immersive experience.

**Augmented Reality (AR):**

Look into AR technologies that allow users to interact with the game in the real world. Combining hand tracking with AR can create unique and engaging experiences.

**Natural User Interfaces (NUI):**

Keep an eye on developments in natural user interfaces. Technologies like Microsoft Kinect, although older, have paved the way for more natural and intuitive interactions.

**Machine Learning for Gesture Recognition:**

As machine learning continues to advance, explore more sophisticated algorithms and models for gesture recognition. Online platforms like TensorFlow Hub (https://tfhub.dev/) and Hugging Face (https://huggingface.co/models) provide pre-trained models that can be fine-tuned for specific tasks.

**Community and Open Source Projects:**

Join communities related to computer vision, gesture recognition, and game development. Keep an eye on open-source projects that demonstrate innovative approaches to gesture-based interactions.

**Sensor Technologies:**Investigate sensor technologies beyond webcams, such as depth sensors or LiDAR, which can provide more detailed information about hand positions.

**Interactive Displays:**

Consider how gesture-based interactions might evolve with emerging interactive display technologies, such as touchless displays or holographic displays.

**REFERENCES**

**OpenCV Documentation:**

Official OpenCV documentation (https://docs.opencv.org/) is a crucial resource for understanding computer vision concepts and functions.

Learn about image processing, feature detection, and video capture using OpenCV.

**cvzone Library:**

The cvzone library, as mentioned in previous responses, is a useful tool for hand tracking. Refer to the GitHub repository: https://github.com/cvzone/cvzone

Explore the documentation and examples to understand how to use it for hand tracking.

**Snake Game Tutorial:**

Understand the basics of creating a simple snake game. There are many tutorials available online for different platforms and programming languages.

For Python and Pygame, you can refer to tutorials like this one: https://pythonspot.com/snake-with-pygame/

**Integrating Hand Tracking:**

Once you have a basic snake game, learn how to integrate hand tracking. Understand how to get hand landmarks and track finger positions.

A guide on integrating hand tracking with OpenCV: https://learnopencv.com/hand-tracking-and-finger-counting-using-opencv-python/

**Gesture Recognition:**

Based on hand landmarks, implement a gesture recognition system. Define gestures for controlling the snake, such as moving left, right, up, or down.

Tutorial on finger tracking and gesture recognition: https://github.com/opencv/opencv/blob/master/samples/python/tutorial\_code/TrackingMotion/hand\_histogram.py

**Real-time Feedback:**

Consider providing real-time visual feedback for the player. Overlay hand images or highlight recognized gestures on the screen.

Explore functions for overlaying images with cvzone.

**GitHub Repositories:**

Search GitHub for existing projects or repositories that implement hand gesture-based games. You can find valuable insights and code examples.Example repository using hand tracking for a snake game: https://github.com/snehpandya/handtracking