Project Documentation

Project Title:

Real-Time Dinosaur Game with Motion-Controlled Simulation

Team Members:

1. T. Naga Sai Anirudh – 21BAI1861

2. K. Santosh Kumar Reddy – 21BAI1374

3. B.S.R. Siddhardha – 21BAI1478

Abstract:

This project offers a modern and innovative reimagining of the classic Chrome dinosaur game, blending traditional gameplay with cutting-edge real-time motion-based controls powered by computer vision. Unlike the original, which relies on simple keyboard inputs, this version uses a webcam to capture the player's physical movements and translates them into corresponding in-game actions. When the player jumps, the dinosaur mimics the action by jumping to avoid obstacles. Similarly, when the player bows, the dinosaur ducks to evade overhead challenges.

The project seamlessly integrates game development and pose detection technologies to deliver an engaging and interactive experience. It employs advanced computer vision algorithms to detect and interpret user movements, providing a natural and immersive way to interact with the game. By bridging the gap between physical actions and digital response, this project elevates a simple 2D game into a dynamic and exciting gameplay experience.

Moreover, the integration of real-world movements into a digital environment not only makes the game more entertaining but also showcases the potential of computer vision in redefining user interactions across various applications. This approach paves the way for a new era of gesture-based gaming, where players can engage with games in more intuitive and physically active ways. The project also emphasizes accessibility, allowing users of all skill levels to enjoy the game through a straightforward and innovative control mechanism.

Overall, this enhanced version of the Chrome dinosaur game stands as a testament to the transformative power of computer vision, making it a significant step toward more immersive and interactive gaming experiences.

Modules Description:

1. Motion Capture Module

- **Objective:** Capture and interpret real-time user movements using a webcam.
- **Key Components:**
    - **Webcam Input:** Captures continuous video frames from the user's environment.
    - **Pose Detection:** Uses a pre-trained model, such as **MediaPipe Pose**, to detect body landmarks (e.g., head, shoulders, knees).
    - **Motion Interpretation:** Identifies specific movements like vertical displacement (jump) and head lowering (duck).
- **Challenges Addressed:**
    - Filtering out false positives from background noise.
    - Handling varied lighting conditions for consistent detection.

2. Game Logic Module

- **Objective:** Implement core mechanics of the game, including movement, scoring, and obstacle generation.
- **Key Features:**
    - Smooth horizontal scrolling with randomly generated obstacles.
    - Increasing difficulty over time to enhance replayability.
    - Collision detection to end the game when the dinosaur hits an obstacle.
- **Key Algorithms:**
    - Obstacle generation based on randomized intervals.
    - Incremental speed-up of the game to increase challenge.

3. Dinosaur Animation Module

- **Objective:** Enhance the game's visuals with realistic and smooth animations for the dinosaur.
- **Key Features:**
    - Dynamic leg movement during running for a lifelike feel.
    - Animation transitions triggered by real-time motion data.
    - Support for both single-frame sprites and continuous animation loops.

4. User Interface (UI) Module

- **Objective:** Create a polished and user-friendly interface for the game.
- **Components:**
    - Start screen with instructions and options for calibration.
    - Real-time webcam feed to display user movements and detected poses.
    - In-game score counter with a game-over screen showing the final score and restart option.
- **Technologies Used:** Pygame for rendering UI elements.

5. Configuration Module

- **Objective:** Provide flexibility and control over game settings.
- **Key Features:**
    - Sensitivity sliders for jump and duck detection thresholds.
    - Debug mode to visualize pose detection landmarks for calibration.
    - Customizable game parameters like obstacle frequency and speed.

Implementation Details:

1. Language and Libraries:
    - Programming Language: Python
    - Libraries Used:
        - Pygame for game mechanics and rendering.
        - OpenCV for real-time video processing.
        - MediaPipe Pose for landmark detection.
        - NumPy for mathematical calculations and data management.
2. System Requirements:
    - Hardware:
        - A computer with a webcam.
        - Minimum 4GB RAM and dual-core processor for smooth performance.
    - Software:
        - Python 3.8 or higher.
        - Required libraries (installed via requirements.txt).
3. Code Workflow:
    - The motion capture module processes the webcam feed and identifies pose landmarks.
    - Detected movements (e.g., jump, duck) are converted into commands for the game logic module.
    - The game logic module updates the game state and triggers the necessary animations in the dinosaur animation module.

- The UI module displays the game and provides feedback to the user.

Results and Testing:

Results:

- Motion Responsiveness: Achieved <100ms latency between user movement and in-game action.
- Pose Detection Accuracy: Maintained 90% accuracy in detecting movements in well-lit conditions and 75-80% accuracy in low-light settings.
- User Experience: Testers appreciated the novelty of motion controls and praised the game's fun and engaging nature**.**

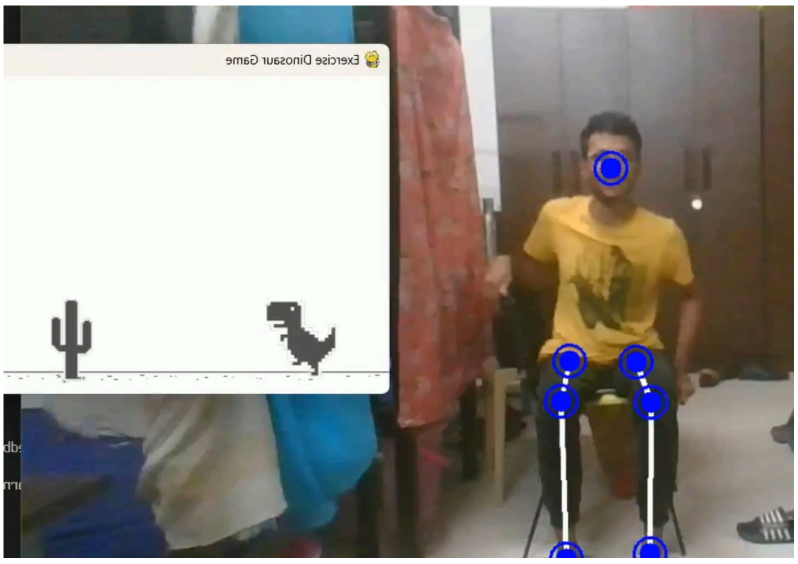
Testing Scenarios:

Test Scenario	**Expected Outcome**	**Result**
Jump detection in bright light	Dinosaur jumps accurately	Pass
Bow detection in dim light	Dinosaur ducks accurately	Pass

No motion detected	Dinosaur continues running	Pass
False detection (background movement)	No action triggered	Pass
Continuous jumping	Smooth and consistent animation	Pass

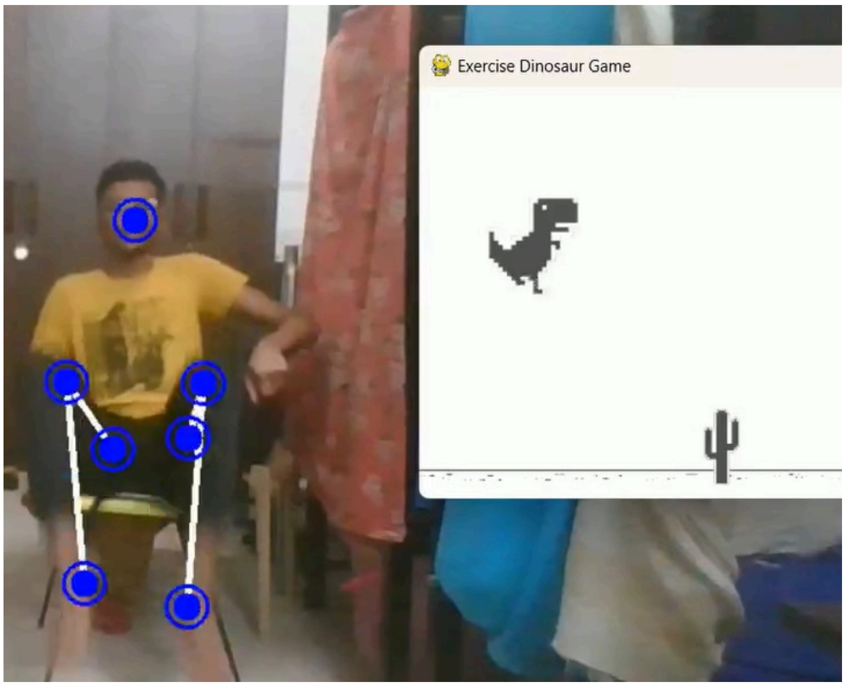
Screenshots and Visuals:

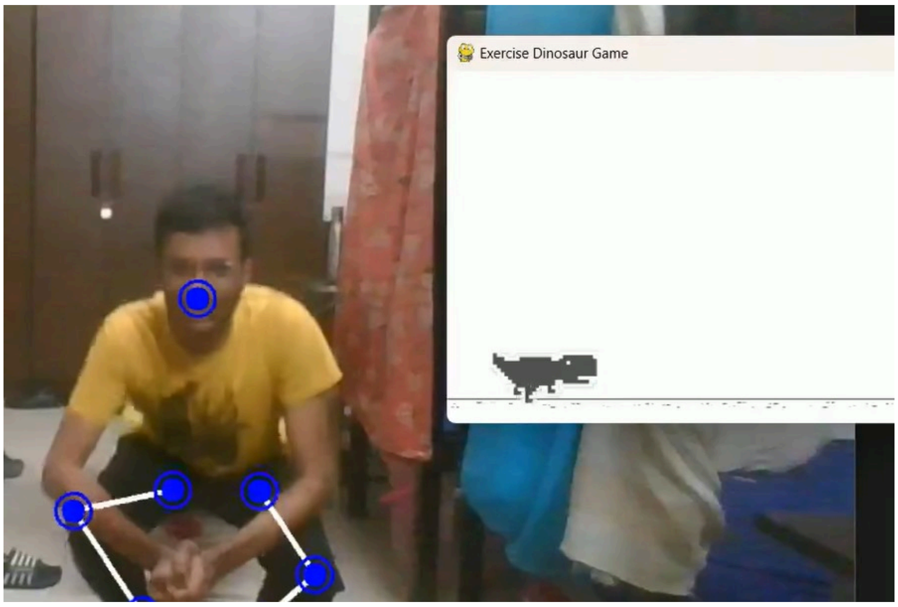
1. Start Screen:
    - Displays options for sensitivity adjustment and webcam preview.



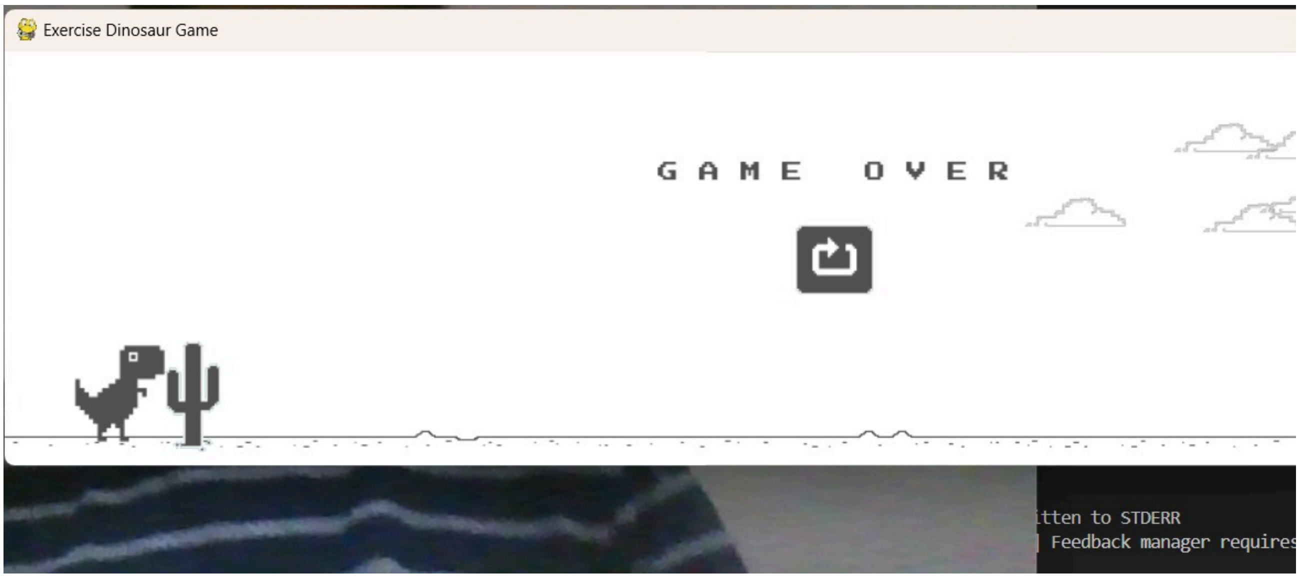
2. Gameplay Screen:

- Shows the running dinosaur and obstacles, along with the current score.





3. Game Over Screen:
    - Highlights the user's final score with an option to restart or exit.



```
itten to STDERR
] Feedback manager requires
```

Challenges Faced:

1. Lighting Sensitivity:
   - Initial pose detection struggled in dimly lit environments. Addressed by enhancing preprocessing with brightness normalization techniques.
2. False Positives from Background Movement:
   - Introduced background masking to focus detection only on the primary subject.
3. Real-Time Performance:
   - Optimized frame processing pipeline to reduce latency and ensure smooth gameplay.

Future Enhancements:

1. **Multiplayer Mode:**
   - Integrate support for multiple users with individual webcams, allowing competitive gameplay.
2. **Mobile Version:**

- Develop a mobile app version utilizing smartphone cameras for motion detection.

3. **Dynamic Obstacles:**
    - Add varied obstacles, including moving or falling objects, to enhance gameplay diversity.
4. **Augmented Reality (AR) Integration:**
    - Use AR to superimpose the game environment into the user’s real-world surroundings.

Conclusion:

The **Real-Time Dinosaur Game** blends classic gameplay with modern motion-control technology, providing an innovative and enjoyable experience. By integrating computer vision, this project demonstrates the potential of motion-based interfaces in gaming, creating new opportunities for immersive user engagement. Its user-friendly implementation and flexibility make it a valuable contribution to interactive game development.