Fruit Ninja

with computer vision

Subject: Imagen digital
University: University of Extramadura

Gianmaria Casamenti
gcasamen@alumnos.unex.es

User Manual

This manual provides instructions on how to install and use the application.

Introduction

Description:
This Project is a reproduction of the famous FruitNinja game for mobile devices.
The game was developed to be played in augmented reality, the user has the mission to cut fruit and avoid bombs.

System Requirements:

- **Operating System**: Windows, macOS, or Linux
- **Required Libraries**: Python, OpenCV, MediaPipe
- **Hardware**: Webcam

Installation

1. **Clone the Repository**

   ```
   git clone https://github.com/GiammaCode/fruitNinja_opencv
   cd fruit-ninja-opencv
   ```

2. **Install Dependencies** Make sure you have Python installed, then install the necessary libraries:

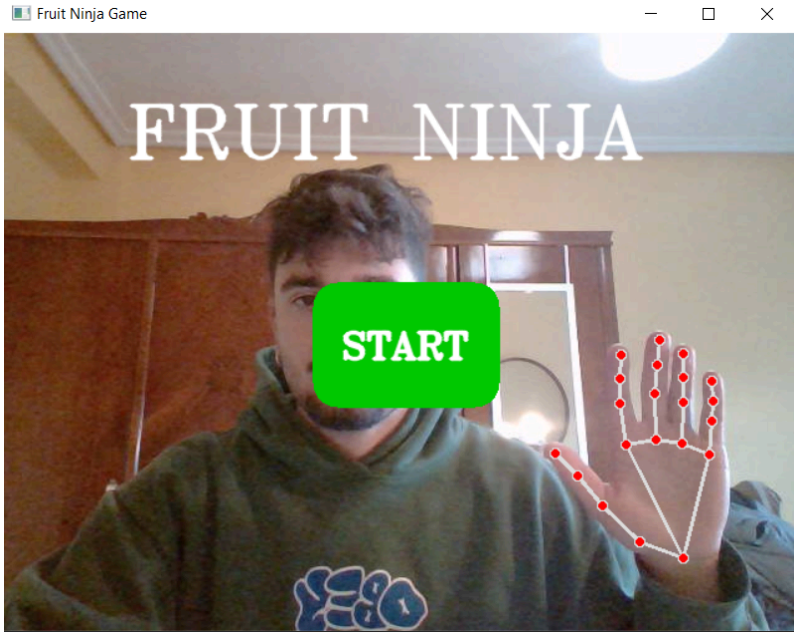
   ```
   pip install -r requirements.txt
   ```

3. **Start the Game**

   ```
   python __init__.py
   ```

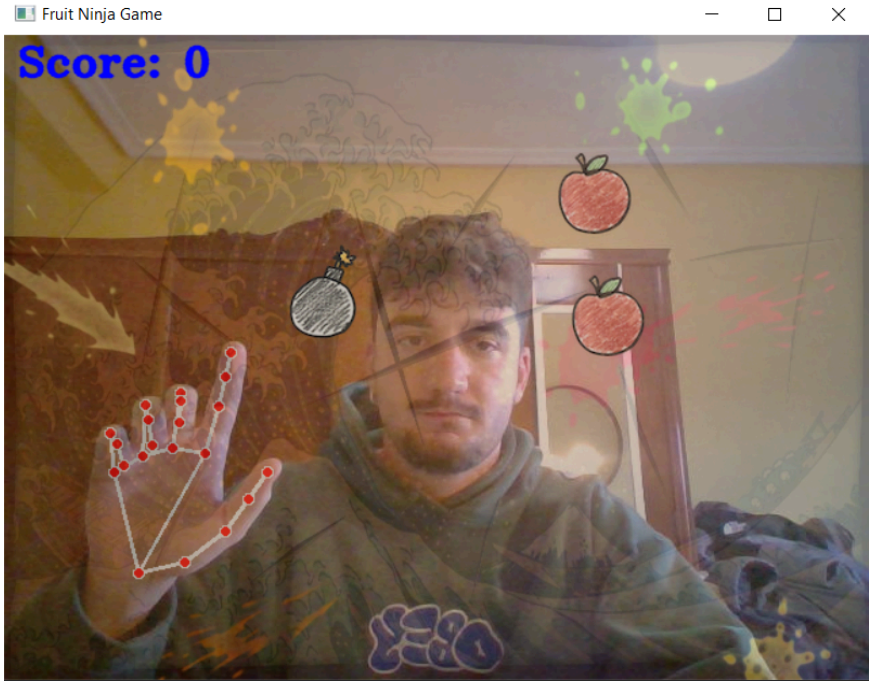
Gameplay

Main Menu:
When you start the application, a pulsating "Start" button will appear on the screen. Hover your hand over the button to start the game.

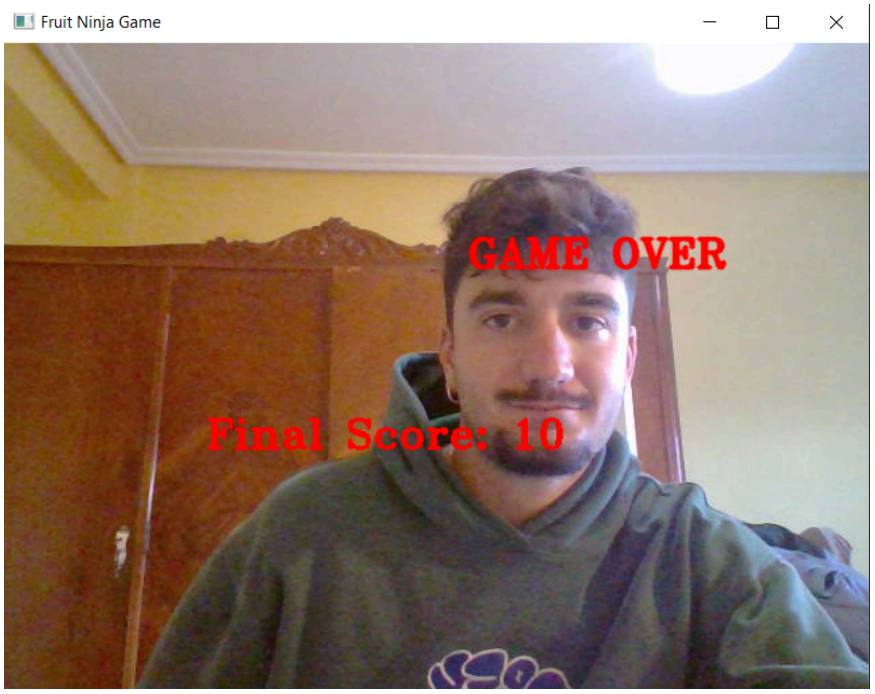


In-Game:

- **Slicing Fruits**: Move your hand over the fruits to slice them and score points.
- **Avoiding Bombs**: Avoid moving your hand over the bombs; otherwise, the game will end.
- **Score Display**: The current score is displayed in real-time at the top of the screen.



Game Over Screen: If the score falls below zero, a "Game Over" screen appears with your final score.



Developer Guide

This guide explains the architecture of the project, with descriptions of the main classes and files.

Project structure

`game/entities/`:

- **`Fruit`**: Manages the fruit object, including movement, collision detection, and rendering on the screen.
- **`Bomb`**: Similar to `Fruit`, but represents a bomb that ends the game when sliced.

`game/utils/`:

- **`HandTracker`**: Uses MediaPipe to detect the user's hand position.
- **`ScoreManager`**: Manages and updates the player's score.

`game/screen/`:

- **`MenuScreen`**: Displays the main menu with a start button.
- **`LoseScreen`**: Shows the "Game Over" screen when the game ends.

`gameScreen.py`: The entry point for the application, containing the `GameController` class, which manages the game loop and state transitions (menu, playing, game over).

key classes and methods

Game Controller

- **Core Methods**:
  - `spawn_fruit` and `spawn_bomb`: Create and add new `Fruit` and `Bomb` objects to the game.
  - `detect_cut_fruit` and `detect_cut_bomb`: Detect collisions between the player's hand and fruits or bombs.
  - `apply_background`: Applies a semi-transparent background to the game frame.
  - `run`: Executes the main game loop, updating the game state based on user interactions.

HandTracker

- Uses MediaPipe to detect the real-time position of the user's hand.
- `track(frame)`: Returns the position of the index fingertip to interact with objects on the screen.

ScoreManager

- `update_score(points)`: Updates the player's score by adding a specified number of points.
- `get_score()`: Returns the current score.

MenuScreen and LoseScreen

- `MenuScreen.draw(frame)`: Draws the main menu with a pulsating "Start" button.
- `LoseScreen.draw(frame)`: Displays the game-over screen with the final score.