COMP4687 – COMPUTER VISION CONTEST PROJECT

TEAM 3

Zeynep Nur Kaya

Berkay Aydemir

Sertaç Çakır

Sami Güvenç

İrem Serra Yağız

Introduction: Color – Hunter Game

- **Project Concept:** Color-Hunter is a high-speed, interactive gaming application that bridges the gap between the physical environment and digital space. Developed using **JavaFX** and the **OpenCV** library, it transforms everyday objects into game controllers through computer vision.
  Core Mechanics:
- **Real-Time Processing:** Captures and analyzes live video at **30 FPS** to provide a seamless user experience.
- **ROI-Based Analysis:** Focuses on a specific **Region of Interest (ROI)** for optimized performance and accurate color sampling.
- **Mathematical Precision:** Utilizes the **Euclidean Distance** formula in the **RGB color space** to compute the similarity between a physical object and the digital target.
- **Dynamic Feedback:** Features an arcade-style scoring system, real-time accuracy percentage, and visual alerts (freeze-frames and flash notifications).
- **The Mission:** To demonstrate the power of **Human-Computer Interaction (HCI)** by challenging players to find and match physical colors within a competitive, 30-second time limit.

Problem Statement & Motivation

- **Heading: Why Color-Hunter?**
- **Static Interaction:** Most traditional games rely on keyboards or controllers, leading to a sedentary experience.
- **Gap in HCI:** There is a need for more intuitive and physically engaging ways to interact with software.
- **Objective:** To leverage **Computer Vision** to turn the real world into a playground, encouraging users to observe their physical environment.

Design Considerations

- **Design & User Experience (UX)**
- **Real-Time Feedback:** High FPS (30 frames per second) for zero-lag interaction.
- **Visual Hierarchy:** Large font sizes and color-coded HUD (Score in Green, Timer in Red).
- **Arcade Mechanics:** * **Success Freeze:** A 3-second static frame to confirm the player's victory.
  - **New Task Flash:** A 250ms high-impact notification for rapid task switching.

Implementation - Architecture

- **System Architecture**
- **Framework: JavaFX** for the Graphical User Interface (GUI).
- **Core Engine: OpenCV 3.4.16** for image capture and processing.
- **Design Pattern: Model-View-Controller (MVC)** to separate UI (FXML) from logic (Java).
- **Threading:** Used ScheduledExecutorService for concurrent image processing without freezing the UI.

Implementation - Technical Details

- **The Algorithm & Math**
- **ROI (Region of Interest):** Only the center box is analyzed to optimize CPU usage.
- **Color Conversion:** Transforming raw **BGR** camera data into the standard **RGB** space for human-understandable analysis.
- **Euclidean Distance:** Measuring color similarity using the 3D distance formula:

$$d = \sqrt{(R_2 - R_1)^2 + (G_2 - G_1)^2 + (B_2 - B_1)^2}$$

- **Dominance Logic:** A secondary filter to ensure color purity (e.g., checking if Blue is significantly higher than Red and Green).

Key Functionalities

- **Core Functionalities**
- **Auto-Targeting:** Randomized target generation (Red, Green, Blue).
- **Accuracy Meter:** Real-time percentage feedback based on mathematical closeness.
- **Game Loop:** Automated cycle between target assignment, detection, and scoring.
- **Safe Shutdown:** Ensures camera hardware is released properly when the application closes.

Reference to Used Libraries

- **Libraries & Development Tools**
- **OpenCV:** Essential for VideoCapture, Imgproc (drawing/text), and Core (matrix operations).
- **JavaFX:** For the multimedia interface and event handling.
- **Java NATIVE:** System.loadLibrary for connecting Java to the OpenCV DLLs.
- **Scene Builder:** For designing the FXML layout visually.

Implementation Challenges

- **Challenges & Solutions**
- **Challenge:** Environmental lighting affects color detection accuracy.
  - *Solution:* Integrated a **Dominance Filter** to prevent gray/white objects from being misidentified.
- **Challenge:** UI lagging during heavy image processing.
  - *Solution:* Implemented multi-threading for frame capturing.
- **Challenge:** Mirror effect confusion.
  - *Solution:* Applied Core.flip to make the camera feel like a real mirror.

Target User Group

- **Who is it for?**
- **Education:** Children learning colors through an interactive and active medium.
- **Active Gaming:** Users looking for a "fit-game" experience that requires physical movement.
- **Developers/Researchers:** A baseline project for Human-Computer Interaction (HCI) research.

Conclusion & Future Work

- **Final Thoughts & Expansion**
- **Current State:** A stable, performant, and engaging 3-color arcade game.
- **Future Work:** * Implementing **HSV (Hue, Saturation, Value)** color space for broader color support (Orange, Purple, etc.).
  - Adding **Hand Gesture Recognition** for menu navigation.
  - Developing a multiplayer mode via network sockets.

Thank You For Listening!

Questions?

Suggestions?