FINAL YEAR PROJECTS

REPORT

COMP1682 VUONG MY CHAU

Professor: Hồ Nguyễn Phú Bảo

[GCS18484]

[001272667]

Table of Contents

Chapter 1 : Introduction to AI Virtual Painter 5
1.1 Overview 5
1.2 Project Objectives: 6
1.3 Project Plan: 8
1.4 Project Outcome: 9
1.5 Project Evaluation: 11
Chapter 2: Literature Review 12
2.1 Hand Tracking Models: 12
2.2 Virtual Painting Systems: 13
2.3 User Experience and Interface Design: 13
2.4 Applications and Future Directions: 14
Chapter 3: Technology and Tools 14
3.1. OpenCV (Open Source Computer Vision Library): 14
3.2. MediaPipe: 15
3.3. NumPy (Numerical Python): 16
3.4. Python Programming Language: 16
3.5. Computer Vision Techniques: 17
3.6. Object-Oriented Programming (OOP) Paradigm: 18
Chapter 4: Software Product Requirements 18
4.1 Comparison with Comparable Offerings 18
4.2 Use Case Diagrams 22
4.3 Sequence Diagrams 23
4.4 Class Diagram 24
4.5 Sitemap 24
Chapter 5: Review of Software Development Methodologies 25
5.1 Waterfall Model 25
5.2 Spiral Model 26
5.3 RAD (Rapid Application Development) and Prototyping 27
5.4 Agile Methodology 27
5.5 Your selection of a software development methodologies and your justification 28
Chapter 6: Design and Implementation of your demo product 29
6.1 Product Analysis and Design 29
6.2 Features include with screenshots 30
6.3 Product Implementation 33
6.4 Evaluation of your product 40
Chapter 7 Conclusion: 43
7.1 What did you learn from this project? 43

7.2 What are the results of this project? 43
7.3 Continue to develop this project 44
References 45

Table of Figure

Fig 1: Use Case Diagrams AI Virtual Painter 22

Fig 2: Sequence Diagrams 23

Fig 3: Class Diagram 24

Fig 4: Sitemap 24

Fig 5: Displayed in pink 31

Fig 6: Displayed in blue 31

Fig 7: Displayed in green 32

Fig 8: Image of a black eraser 33

Fig 9: Color has been removed 33

Proposed topic: Build virtual painted AI hand tracking verion computer

Chapter 1 : Introduction to AI Virtual Painter

1.1 Overview

Welcome to the future of interaction and creativity with the Virtually Drawn AI Hand Tracking PC Edition. This innovative technology combines artificial intelligence with advanced hand tracking to revolutionize the way we interact with computers and unleash our creativity like never before.

- The foundation of this innovation lies in advanced hand tracking technology. Using advanced sensors and machine learning algorithms, the system accurately detects and interprets hand movements in real time. Whether you are gesturing, pointing, or creating complex movements, the computer seamlessly converts your movements into digital commands with stunning precision.
- Imagine drawing on canvas without touching a physical brush. With the built-in Virtually Drawn AI Hand Tracking PC Edition, this will become a reality. Through intuitive hand gestures, users can express their artistic vision directly on the virtual canvas. Whether you are an experienced artist or just starting out, the immersive painting experience will enhance creativity in whole new ways.
- Further enhancing the user experience is the integration of artificial intelligence. The computer learns from your drawing style, preferences and techniques, providing personalized suggestions and support along the way. From suggesting color palettes to providing stroke guidance, AI adapts to your needs, promoting growth and experimentation in your artistic journey.

- In addition to drawing capabilities, the Virtually Painted AI Hand Tracking Edition PC also boasts a versatile range of applications. From 3D modeling and sculpting to virtual design prototyping, intuitive hand tracking systems open up endless possibilities across a variety of creative fields. Furthermore, its seamless integration with existing software platforms ensures compatibility and ease of use for both professionals and enthusiasts.

- In short, the Virtually Drawn AI Hand Tracking Computer Edition represents a quantum leap in human-computer interaction and creative expression. By combining advanced hand tracking technology with artificial intelligence, this innovative system redefines the way we interact with computers and unleashes our creativity. Whether you are an artist, designer or enthusiast, embrace the future of digital expression with this groundbreaking technology.

1.2 Project Objectives:

Enhanced user experience:

- Customer Engagement: Our goal with the Virtually Drawn AI Hand Tracking Edition Calculator project was to engage users by providing an immersive and intuitive interface. We will ensure easy navigation, allowing users to seamlessly explore the system's features, functions and innovative capabilities.

- Intuitive Design: Our goal is to create more than just a technological marvel; We wanted to create an engaging platform that users would love to interact with. The interface will have an attractive-looking layout, ensuring users can easily find the tools and options they need. We prioritized a clean and intuitive design that fits perfectly with the brand's ethos. The Virtual Drawn AI Hand Tracking Edition Computer is built.

Administrative and Functional Controls for the Build Virtual Painted AI Hand Tracking Version Computer:

- Administrative Empowerment: Within the Build Virtual Painted AI Hand Tracking Version Computer ecosystem, administrators will wield comprehensive control over the system's functionalities. Equipped with secure login credentials, administrators will efficiently manage various aspects of the platform's operation.

- Product Management: Administrators will have the authority to update, edit, and organize the virtual painting features and tools available to users. This includes managing brush styles, color palettes, canvas settings, and other creative options.
- User Account Management: The system will enable administrators to oversee user accounts, including registration, authentication, and access privileges. They will have the capability to manage user profiles, preferences, and activity logs to ensure a personalized and secure experience for each user.
- System Configuration: Administrators will have the flexibility to configure system settings and parameters to optimize performance and functionality. This includes managing hardware resources, software updates, and integration with external platforms or services.

Functional Control:

- Virtual Painting Environment: The Build Virtual Painted AI Hand Tracking Version Computer will provide administrators with granular control over the virtual painting environment. They will have the ability to customize the interface, tools, and interactions to meet the diverse needs and preferences of users.
- Real-time Monitoring: Administrators will have access to real-time monitoring tools to track user interactions and system performance. This includes monitoring virtual painting sessions, user engagement metrics, and system usage statistics to identify trends and opportunities for improvement.
- Security Measures: Administrators will implement robust security measures to safeguard sensitive data and protect against unauthorized access or malicious activities. This includes encryption protocols, authentication mechanisms, and intrusion detection systems to ensure the integrity and confidentiality of user information.

- By empowering administrators with these comprehensive controls, the Build Virtual Painted AI Hand Tracking Version Computer ensures a secure, efficient, and user-friendly experience for both administrators and users alike..

1.3 Project Plan:

- Exploration and Analysis:

Conduct thorough market research and analysis to understand the specific needs and preferences of potential users.

Gather comprehensive information about existing virtual painting technologies, competitor offerings, and emerging market trends in AI and hand tracking.

- Requirement Identification and Functionality:

Define clear functionalities for both users and administrators within the virtual painting environment.

Compile a detailed list of features to be implemented, including virtual brush options, color palettes, canvas settings, and administrative controls for managing user accounts and system configurations.

- Design and Development:

Develop a user interface design that prioritizes superior user experience (UX) principles, ensuring intuitive navigation and seamless interaction with virtual painting tools.

Implement features and functionalities for both users and administrators, focusing on flexibility, performance, and compatibility with AI and hand tracking technologies.

- Testing and Evaluation:

Conduct rigorous testing across various platforms and devices to ensure optimal performance and compatibility with different operating environments.

Gather feedback from a test group of users to identify areas for improvement and refinement of the virtual painting experience.

- Deployment and Support:

Roll out the virtual painting platform, prioritizing stability, security, and scalability to accommodate growing user demand.

Provide comprehensive support and training for administrators to effectively manage user accounts, system configurations, and performance monitoring.

- Management and Upgrades:

Continuously monitor the virtual painting platform's performance and user feedback to identify opportunities for enhancements and upgrades.

Execute regular updates and upgrades to maintain competitiveness and ensure the platform remains at the forefront of AI and hand tracking technology.

- Marketing and Distribution:

Develop a strategic marketing plan to promote the Build Virtual Painted AI Hand Tracking Version Computer, highlighting its unique features and capabilities to attract users.

Establish efficient distribution channels to ensure widespread access to the virtual painting platform, leveraging digital platforms and partnerships with relevant industry stakeholders.

- Effectiveness Evaluation:

Evaluate the effectiveness of the virtual painting platform based on key metrics such as user engagement, adoption rates, and customer satisfaction.

Adjust marketing and development strategies based on performance data to optimize project success and drive continuous improvement.

- This comprehensive project plan emphasizes the importance of collaboration across various departments to ensure the successful development, deployment, and management of the Build Virtual Painted AI Hand Tracking Version Computer, delivering a cutting-edge virtual painting experience to users worldwide.

1.4 Project Outcome:

- Exceptional Virtual Painting Environment:

The virtual painting interface design is optimized, delivering a seamless and immersive experience for users engaging in digital artistry.

Users can navigate the interface effortlessly, exploring a range of virtual brushes, color palettes, and canvas settings to unleash their creativity.

- Flexible and Efficient Functionality:

The implemented functionalities, from virtual brush selection to canvas manipulation, are designed for flexibility and efficiency.

Administrators have robust control over the virtual painting environment, enabling them to manage brush options, user accounts, and system configurations with ease.

- Data Security and Privacy:

The system is engineered with stringent security measures to safeguard users' personal information and digital artwork during virtual painting sessions.

Built-in encryption protocols ensure the confidentiality and integrity of user data, fostering trust and confidence among users.

- Strategic Marketing and Promotion:

A comprehensive digital marketing strategy has effectively promoted the Build Virtual Painted AI Hand Tracking Version Computer, generating widespread interest and engagement.

Targeted advertising campaigns and promotional initiatives have attracted a diverse audience of digital artists and enthusiasts to explore the platform's capabilities.

- Continuous Evaluation and Enhancement:

Through ongoing evaluation of key performance metrics, we continuously refine and optimize the virtual painting platform to enhance user experience and satisfaction.

Feedback from users and administrators informs iterative improvements and feature enhancements to meet evolving needs and preferences.

- Positive User Feedback and Engagement:

The Build Virtual Painted AI Hand Tracking Version Computer has garnered positive feedback from users and administrators alike, praising its intuitive interface and advanced features.

Users express satisfaction with the platform's responsiveness and functionality, fostering a vibrant community of digital artists and creators.

These outcomes underscore the dedication and collaborative efforts of our project team in delivering a state-of-the-art virtual painting platform. We remain committed to advancing digital artistry and providing an unparalleled creative experience for users of the Build Virtual Painted AI Hand Tracking Version Computer.

1.5 Project Evaluation:

The development and deployment of the Build Virtual Painted AI Hand Tracking Version Computer project have been a comprehensive endeavor marked by several notable aspects:

- Achievement of Objectives:

The project successfully met its primary objectives, delivering an intuitive user interface, streamlined functionalities for both users and administrators, and robust security measures to protect user data.

- User Experience and Interface:

Feedback from users has highlighted the intuitive and visually appealing nature of the virtual painting interface, providing users with a seamless and engaging creative experience.

Users commend the ease of navigation and accessibility of virtual painting tools, contributing to high levels of user satisfaction and engagement.

- Functionality and Performance:

Functionality implementation, including virtual brush options, color palettes, and canvas settings, was executed effectively, ensuring smooth and responsive operations.

The system's performance has been commendable, with efficient processing of virtual painting commands and responsive interactions meeting user demands.

- Marketing Impact and Engagement:

The digital marketing strategy effectively increased awareness and interest in the Build Virtual Painted AI Hand Tracking Version Computer, attracting a diverse audience of digital artists and enthusiasts.

Engagement metrics reflect positive user interactions, indicating the successful execution of promotional campaigns and the appeal of the platform's features.

- Continuous Improvement:
Ongoing evaluations and data analysis have identified areas for improvement, leading to regular updates and enhancements to further enhance the virtual painting experience and optimize platform performance.
- Stakeholder Feedback:
Stakeholders, including users and administrators, have expressed satisfaction with the platform's functionality, design, and overall performance, reinforcing its value and relevance.
- Future Strategies:
Insights from project evaluations will inform future strategies aimed at continually improving the platform's functionality, security, and user engagement, ensuring it remains at the forefront of virtual painting technology.
In conclusion, the Build Virtual Painted AI Hand Tracking Version Computer project has successfully delivered a robust, user-centric platform that meets the objectives and expectations of both the brand and its users.

Chapter 2: Literature Review

In recent years, advancements in computer vision and machine learning have revolutionized human-computer interaction, particularly in the domain of hand tracking technology. Hand tracking systems play a crucial role in enabling natural and intuitive interaction with digital interfaces, ranging from virtual reality (VR) environments to creative applications such as virtual painting.

2.1 Hand Tracking Models:

- Hand tracking models form the backbone of hand tracking systems, providing the ability to detect and track the intricate movements and gestures of human hands. One prominent approach to hand tracking involves the use of convolutional neural networks (CNNs) trained on large-scale annotated datasets. These CNN-based models, such as the ones used, leverage deep learning techniques to learn the distinguishing features of hand images and predict the locations of key landmarks, e.g. such as the finger joints and the center of the palm. There is also growing interest in using recurrent neural networks (RNN) and attention mechanisms to capture temporal dependence and spatial relationships in hand movements. These models enable more robust and context-aware hand tracking, particularly in dynamic environments where hands may undergo complex interactions with objects or undergo occlusions.Furthermore, recent research has explored the integration of hand tracking models with probabilistic graphical models, such as hidden Markov models (HMMs) and conditional random fields (CRFs), to incorporate temporal constraints and smooth hand trajectories. These hybrid models combine the strengths of deep learning for feature representation and graphical models for sequential inference, resulting in enhanced accuracy and robustness in hand tracking.

2.2 Virtual Painting Systems:

- Virtual painting systems leverage hand tracking technology to simulate traditional painting techniques in a digital environment. These systems enable users to express their creativity using virtual brushes, colors, and canvases, facilitated by intuitive hand gestures and movements. The script exemplifies the integration of hand tracking with computer vision and graphical rendering techniques to create an immersive painting experience.Beyond basic painting functionalities, modern virtual painting systems explore advanced features such as brush dynamics, texture simulation, and multi-user collaboration. By incorporating machine learning algorithms for stroke prediction and style transfer, these systems empower users to explore diverse artistic styles and experiment with new creative techniques.

2.3 User Experience and Interface Design:

- Central to the success of hand tracking-based applications is the user experience (UX) and interface design. Intuitive interaction mechanisms, responsive feedback, and visually appealing interfaces are essential for engaging users and facilitating creative expression. Both the scripts prioritize user-centric design principles, providing seamless interaction workflows and customizable painting tools to enhance user engagement and satisfaction.

2.4 Applications and Future Directions:

- Hand tracking technology and virtual painting systems have broad applications across various domains, including entertainment, education, healthcare, and design. In addition to virtual painting, hand tracking can enable gesture-based interaction in VR environments, sign language recognition, rehabilitation therapy, and interactive storytelling.Future research directions may focus on improving the robustness and generalization capabilities of hand tracking models, particularly in challenging scenarios such as occlusions, varying lighting conditions, and diverse hand poses. Furthermore, advancements in hardware technologies, such as depth-sensing cameras and wearable devices, offer new opportunities for enhancing the accuracy and spatial resolution of hand tracking systems.Moreover, the integration of hand tracking with other modalities, such as gaze tracking, voice recognition, and haptic feedback, can enrich the user experience and enable novel interaction paradigms. Collaborative artwork creation, real-time feedback mechanisms, and personalized artistic assistance are exciting avenues for innovation in the intersection of hand tracking and virtual painting.

Chapter 3: Technology and Tools

3.1. OpenCV (Open Source Computer Vision Library):

- Purpose: OpenCV is a powerful open-source library that provides a comprehensive suite of tools for computer vision and image processing tasks.
- Applications: It is used for tasks such as image and video manipulation, object detection and tracking, facial recognition, and gesture analysis.
- Usage: Both scripts leverage OpenCV for tasks such as reading video frames from a webcam, image manipulation, drawing shapes and text on images, and displaying images in real-time.

✚ Advantages:

- Flexibility and Versatility: OpenCV provides a wide range of tools and algorithms for image and video processing tasks, from object detection to motion tracking.

- Performance and Reliability: OpenCV is optimized to ensure high performance and reliability in real-time computer vision applications.
- Ease of Use: OpenCV is written in C++ and Python, two popular and accessible programming languages.

Disadvantages:

- Technical Knowledge Requirements: Using OpenCV requires developers to have knowledge of image processing and computer programming.

3.2. MediaPipe:

- Purpose: MediaPipe is an open-source framework developed by Google that offers solutions for various multimedia processing tasks, including hand tracking, pose estimation, and object detection.
- Applications: It is utilized for real-time, high-precision hand tracking, enabling the detection and localization of hand landmarks and gestures.
- Usage: The "HandTrackingModule.py" script integrates the MediaPipe Hands module to detect hand landmarks and gestures, facilitating functionalities such as finger counting and hand gesture recognition.

Advantages:

- Accuracy and Stability: MediaPipe offers precise and stable computer vision solutions for detection and tracking applications.
- Flexible Design: MediaPipe has a flexible design and can be easily integrated into different projects.

Disadvantages:

- Limited Customization: MediaPipe may be limited in customization and fine-tuning of algorithms for specific application needs.

3.3. NumPy (Numerical Python):

- Purpose: NumPy is a fundamental package for scientific computing in Python, providing support for multidimensional arrays, mathematical functions, and linear algebra operations.
- Applications: It is extensively used for numerical computations, data manipulation, and array operations in various scientific and engineering applications.
- Usage: Both scripts utilize NumPy arrays for efficient representation and manipulation of image data, enabling tasks such as array slicing, element-wise operations, and matrix transformations.

- Advantages:
  - Flexibility and Power: NumPy provides a variety of functions and algorithms for scientific computing and data processing.
  - High Performance: NumPy is optimized for high performance and efficiency in handling large data.

- Disadvantages:
  - Learning Curve: Using NumPy requires users to have knowledge of scientific computing and Python programming.

3.4. Python Programming Language:

- Purpose: Python is a versatile, high-level programming language known for its simplicity, readability, and extensive ecosystem of libraries and frameworks.
- Applications: It is widely used in diverse domains such as web development, data science, machine learning, and scientific computing.
- Usage: Both scripts are written in Python, leveraging its rich set of libraries and concise syntax to implement functionalities such as class definitions, loop structures, conditional statements, and function calls.

- Advantages:

- Ease of Learning and Use: Python has simple and readable syntax, making it an ideal programming language for beginners.
- Large Community: Python has a large and diverse community, providing support and a variety of documentation for users.

- Disadvantages:

- Limited Performance: Python is often not optimized for high performance, especially in applications requiring fast and efficient processing.

3.5. Computer Vision Techniques:

- Purpose: Computer vision techniques encompass a range of algorithms and methodologies for extracting meaningful information from digital images and videos.
- Applications: They are employed for tasks such as image segmentation, feature extraction, object detection, and motion tracking.
- Usage: The scripts employ computer vision techniques to detect and track hand landmarks, recognize hand gestures, perform image processing operations, and overlay virtual elements onto the camera feed.

- Advantages:

- Detection and Tracking Capabilities: Computer vision techniques provide methods and algorithms for detecting and tracking objects in images and videos.
- Wide Range of Applications: Computer vision techniques are used in various fields such as face recognition, image classification, and expression recognition.

- Disadvantages:

- Complexity: Some computer vision algorithms can be complex and require in-depth knowledge of image processing and mathematics.

3.6. Object-Oriented Programming (OOP) Paradigm:

- Purpose: Object-oriented programming is a programming paradigm that organizes code into objects, which encapsulate data and behavior.
- Applications: It facilitates modular, reusable, and maintainable code development, promoting code organization and abstraction.
- Usage: Both scripts are structured using object-oriented principles, with classes defined for encapsulating functionalities such as hand detection, landmark tracking, and canvas rendering.

✚ Advantages:

- Structured Organization: Object-oriented programming helps organize source code logically and systematically, making it easy to maintain and extend.
- Reusability: Objects can be reused in different software, saving time and effort in development.

✚ Disadvantages:

- Complexity: Object-oriented programming can be complex for beginners or those unfamiliar with this programming style.

Chapter 4 :Software Product Requirements

4.1 Comparison with Comparable Offerings

4.1.1 Google's Quick, Draw!: This is a web application developed by Google, allowing users to draw simple images on a drawing board and use artificial intelligence to guess what the image is. Although there is no hand tracking function, it provides a similar experience of drawing and interacting with images.

- Advantages:

- Simple and Intuitive: Quick, Draw! offers a straightforward and intuitive interface, making it easy for users to engage with and create simple drawings.
- AI Guessing Game: The AI guessing game adds an element of fun and challenge, as users can see if the AI can correctly identify their drawings, enhancing the overall user experience.
- No Installation Required: Being a web-based application, Quick, Draw! does not require any installation, allowing users to access and use it directly from their web browser without the need for additional software.

- Disadvantages:

- Limited Drawing Features: Quick, Draw! is primarily designed for simple drawings, so it lacks advanced drawing features and tools compared to dedicated drawing software.
- Lack of Hand Tracking: Since Quick, Draw! does not support hand tracking, users are limited to using a mouse or touchscreen for drawing, which may not provide the same level of precision and natural interaction as hand tracking.
- Dependency on AI: The accuracy of the AI's guesses depends on the complexity and quality of the drawings, which may vary and lead to inconsistent results. Additionally, users may find the AI's inability to correctly identify their drawings frustrating at times.

4.1.2 Microsoft Paint 3D: Microsoft Paint 3D is a free application built into the Windows 10 operating system. It provides 2D and 3D drawing tools for users to create digital works of art. Although there is no hand tracking feature, users can use a mouse or stylus to draw and interact with images.

- Advantages:

- Integrated with Windows: As a built-in application in Windows 10, Paint 3D is readily available to users without the need for separate installation or additional cost.

- 2D and 3D Drawing: Paint 3D offers both 2D and 3D drawing capabilities, allowing users to create a wide range of digital artworks, from simple sketches to complex 3D models.
- Compatibility: Paint 3D supports various file formats, making it compatible with other software and platforms, facilitating seamless sharing and collaboration.

✚ Disadvantages:

- Limited Advanced Features: While Paint 3D is suitable for basic drawing tasks, it lacks advanced features and tools found in dedicated graphic design software, limiting its suitability for professional or advanced users.
- No Hand Tracking Support: Similar to Quick, Draw!, Paint 3D does not support hand tracking, relying on traditional input devices such as a mouse or stylus for drawing, which may not offer the same level of precision or natural interaction as hand tracking.
- Learning Curve: Despite its simplicity, mastering Paint 3D's 3D drawing features and interface may require some learning and practice, especially for users unfamiliar with 3D modeling concepts.

4.1.3 Adobe Fresco: Adobe Fresco is a professional digital drawing application for mobile devices and tablets. It offers natural drawing tools like pencil, watercolor, and ink, as well as photo overlay and animation features. Although there is no hand tracking function, users can use a stylus or stylus controller to create digital works of art.

✚ Advantages:

- Professional Drawing Tools: Adobe Fresco provides a comprehensive set of professional drawing tools, including natural media brushes, vector brushes, and live brushes, catering to the needs of professional artists and illustrators.
- Integration with Adobe Creative Cloud: Being part of the Adobe Creative Cloud ecosystem, Fresco seamlessly integrates with other Adobe software such as Photoshop and Illustrator, allowing for smooth workflow and collaboration.
- Cross-Platform Support: Adobe Fresco is available on both mobile devices and tablets, offering users the flexibility to create and work on their artworks across different devices and platforms.

Disadvantages:

- Subscription-Based Model: Adobe Fresco is subscription-based software, requiring users to pay a monthly or annual fee for access, which may not be affordable for all users, especially hobbyists or casual artists.
- Complexity: While Adobe Fresco offers powerful drawing features, its extensive toolset and advanced capabilities may be overwhelming for beginners or users with limited experience in digital art.
- Hardware Requirements: To fully utilize Adobe Fresco's features, users may need to have compatible hardware, such as a stylus or drawing tablet, which could add to the overall cost of using the software.

4.1.4 Tilt Brush by Google: Tilt Brush is a virtual reality application that allows users to draw in 3D space using interesting tools and effects. Although it lacks traditional hand tracking, it uses VR technology to track the user's hand and body movements in 3D space, creating a consistent dynamic drawing experience.

Advantages:

- Immersive 3D Drawing Experience: Tilt Brush offers an immersive 3D drawing experience, allowing users to create artworks in a virtual environment using intuitive hand gestures and movements.
- Creative Freedom: Tilt Brush provides users with a wide range of brushes, effects, and tools to unleash their creativity and explore different artistic styles and techniques.
- VR Integration: Tilt Brush leverages VR technology to track users' hand and body movements in real-time, providing a more natural and expressive way of creating art compared to traditional drawing software.

Disadvantages:

- VR Hardware Required: Tilt Brush requires compatible VR hardware, such as a VR headset and motion controllers, which may be costly and inaccessible to some users.
- Steep Learning Curve: Mastering Tilt Brush's VR-based interface and controls may require some time and practice, especially for users who are new to virtual reality or 3D art.

- Limited Output Options: While Tilt Brush allows users to create stunning 3D artworks, the output options are limited primarily to sharing within the Tilt Brush community or exporting as videos or images, which may not be suitable for all use cases or industries.

4.2 Use Case Diagrams

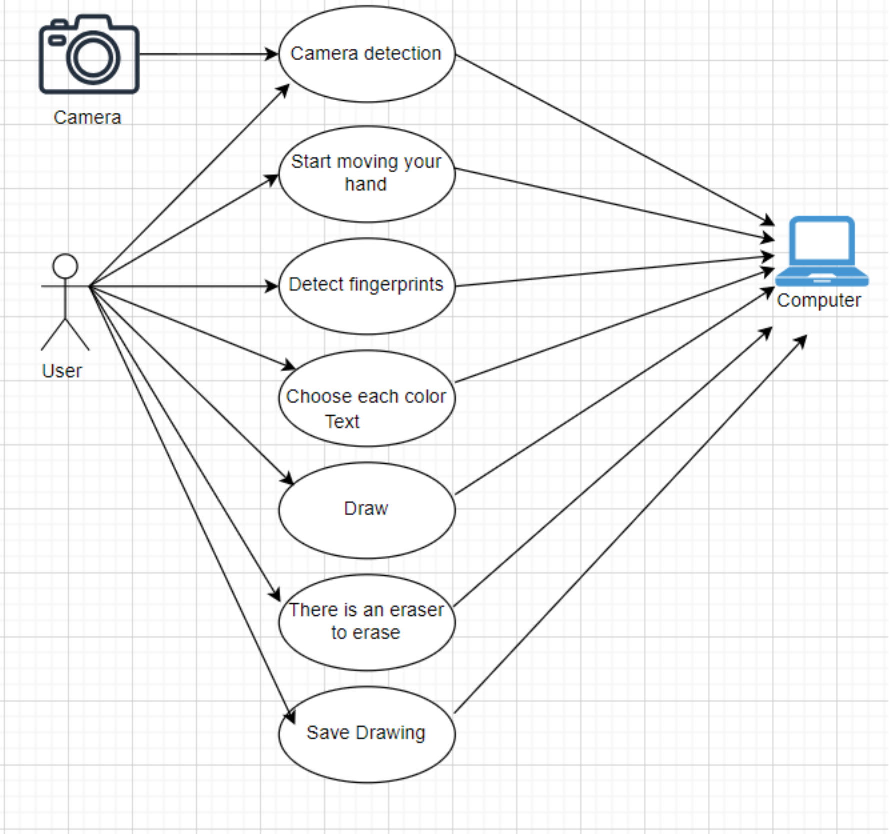


Fig 1: Use Case Diagrams AI Virtual Painter

- Use Cases: The functions that can be used are drawing on the computer screen and choosing a color. There will be 3 colors for the user to choose from and there will be an eraser so the user can erase and save the image. Just turn on the camera on the laptop and the drawing operations will be performed

4.3 Sequence Diagrams

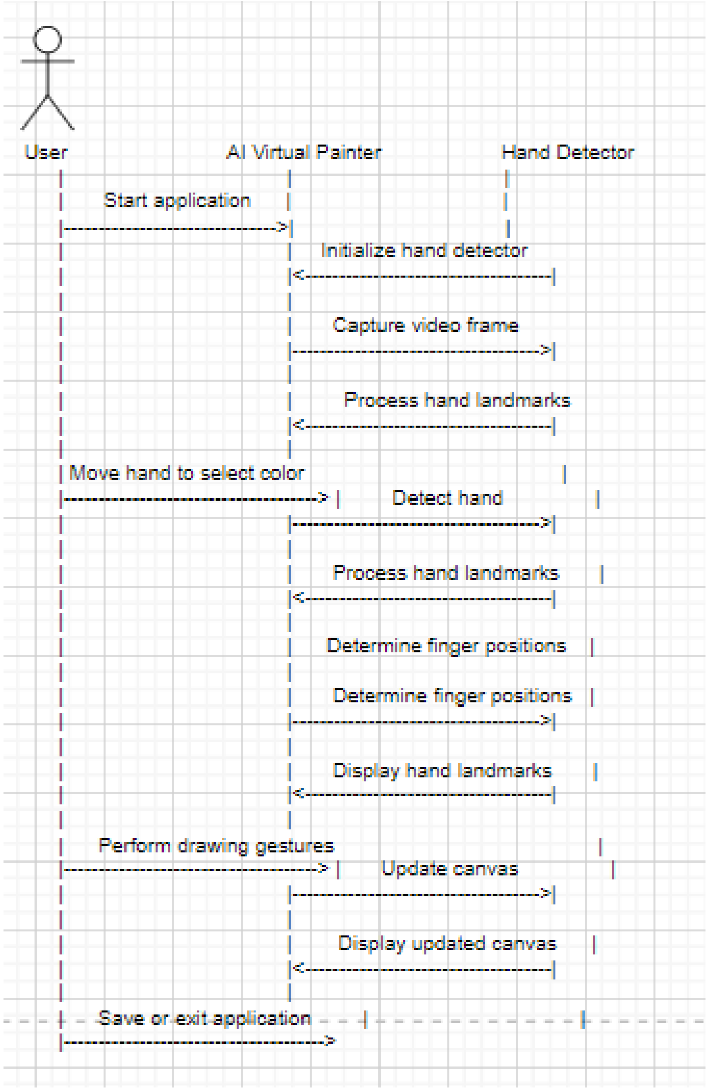


Fig 2: Sequence Diagrams

- The user starts the application and performs actions such as selecting colors or drawing by moving their hand.
- AI Virtual Painter initializes the hand detection module and captures frames from the video.
- The hand detection module processes the hand landmarks and identifies the positions of the fingers.
- Subsequently, AI Virtual Painter displays the hand landmarks and updates the drawing canvas.

- This process continues until the user saves or exits the application.
- This diagram helps understand how the components in the system interact with each other during the usage of AI Virtual Painter.

4.4 Class Diagram

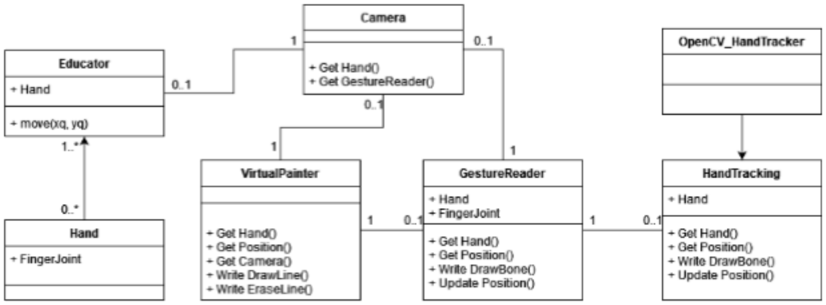


Fig 3: Class Diagram

- Class diagrams have functions that describe the relationship between one class and another. hand, camera, virtual painter, gesture reader, hand tracking, OpenCV_HandTracker. And each class has its own properties and operations within each class.

4.5 Sitemap

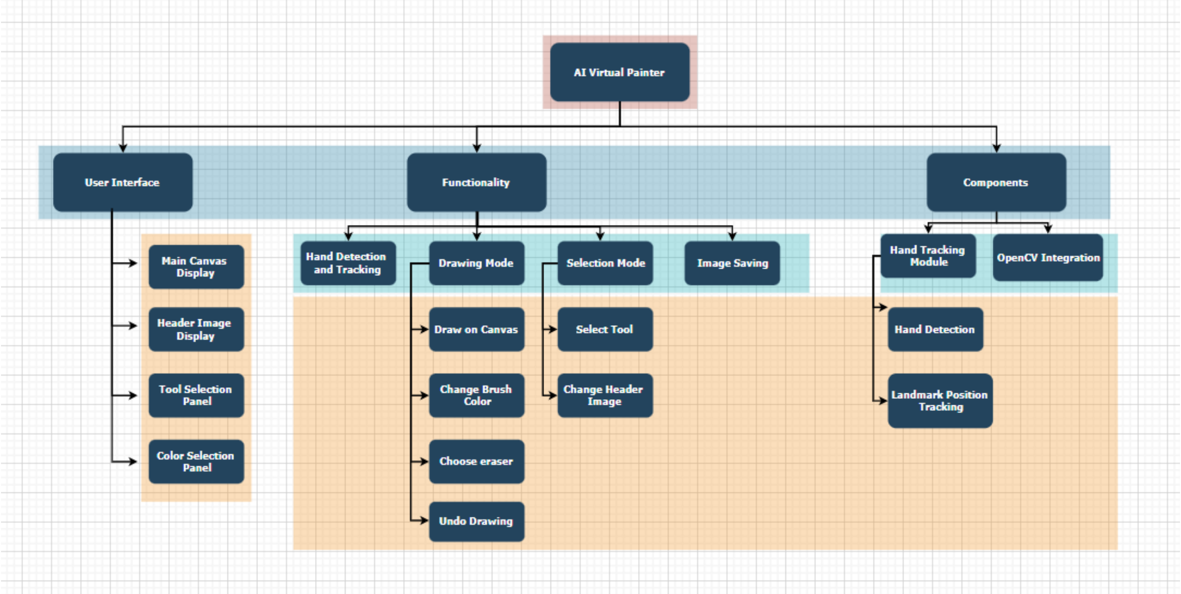


Fig 4: Sitemap

- User Interface:
- Displays user interface elements including the main drawing panel, header image display, tool selection buttons, and color picker.
- Functionality:
- Hand Detection and Tracking: Detect and track the position of the user's hand on the camera.
- Drawing Mode: Drawing mode, allowing users to draw on the drawing board and adjust parameters such as color and size of the pen.
- Selection Mode: Selection mode, allowing users to select tools and header images.
- Image Saving: Stores the drawn image into an image file.
- Components:
- Hand Tracking Module: Hand detection and tracking module uses the Mediapipe library.
- OpenCV Integration: OpenCV integration for image processing and user interface rendering.

Chapter 5: Review of Software Development Methodologies

5.1 Waterfall Model

Overview: The Waterfall model follows a sequential, linear approach to software development. It consists of distinct phases such as requirements gathering, design, implementation, testing, deployment, and maintenance. Each phase relies on the completion of the previous one and typically doesn't allow for iteration. This structured approach ensures a clear understanding of project requirements at the outset and comprehensive documentation at each stage. However, its rigidity can be a limitation when adapting to changing requirements or when issues are discovered later in the process.

- Advantages:

- Provides a structured and systematic approach.
- Offers clear documentation for each phase.
- Suitable for projects with well-defined and stable requirements.

- Disadvantages:

- Limited flexibility for changes or iterations after initial planning.
- Issues discovered late in the process can be costly to rectify.
- Client feedback and changes are challenging to accommodate without revisiting earlier phases.

5.2 Spiral Model

Overview: The Spiral model combines the Waterfall's sequential approach with elements of iterative development and risk management. It involves cycles (spirals) where each loop includes planning, risk analysis, engineering, and evaluation. This iterative nature allows for better risk management, accommodating changes and refining requirements. By addressing high-risk aspects early and allowing for iterations, the Spiral model promotes a more flexible development process.

- Advantages:

- Risk management is incorporated throughout the development life cycle.
- Supports iteration and prototyping to accommodate changing requirements.
- Provides flexibility in responding to evolving project needs.

- Disadvantages:

- Can be complex to manage with multiple cycles and iterations.
- Costlier due to the constant evaluation and refinement cycles.
- The extensive planning and documentation might slow down progress.

5.3 RAD (Rapid Application Development) and Prototyping

Overview: RAD is an iterative approach focused on rapid development and quick feedback. It emphasizes user involvement and prototyping to accelerate the software development process. Prototyping allows stakeholders to visualize and interact with the system early, providing valuable insights and facilitating quick adjustments based on user feedback.

- Advantages:

- Accelerates development by emphasizing user involvement and rapid prototyping.
- Allows stakeholders to visualize the end product early in the process.
- Enables quick adaptation to changing requirements based on user feedback.

- Disadvantages:

- Initial analysis might lack depth, leading to potential issues in later stages.
- Continuous changes based on feedback might result in scope creep.
- Relies heavily on user involvement, which can be demanding and time-consuming.

5.4 Agile Methodology

Overview: Agile is a collaborative, iterative approach that focuses on adaptability, customer involvement, and continuous improvement. It encompasses various methodologies like Scrum, Kanban, and XP, emphasizing teamwork, flexibility, and delivering working software in short cycles (iterations or sprints).

- Advantages:

- Emphasizes customer satisfaction through continuous delivery and iteration.
- Enhances adaptability to changes with regular feedback and iterations.
- Encourages teamwork, collaboration, and transparency.

- Disadvantages:

- Requires skilled and empowered teams for effective implementation.
- Frequent communication and active stakeholder involvement are essential, demanding more resources.
- May face challenges in scaling for larger projects or distributed teams.)

5.5 Your selection of a software development methodologies and your justification

- I chose Agile Methodology to develop the AI Virtual Painter project because it provides flexibility and the ability to quickly adapt to changing requirements during development. With the Agile Scrum method, we can divide the project into short Sprints and focus on delivering value to customers after each Sprint. This helps optimize the development process, enhance team collaboration, and provide quick feedback from users.

- By using Agile, we can flexibly adjust and improve project deliverables based on feedback from customers and stakeholders. By breaking the project into smaller parts, we can control risk and optimize agility during development.

- In addition, Agile also creates a positive working environment and encourages creativity from every team member. By focusing on delivering value to customers and adapting quickly to change, we can ensure that the AI Virtual Painter project will be developed efficiently and deliver results. real value for users.

Chapter 6: Design and Implementation of your demo product

6.1 Product Analysis and Design

- GUI Design:

- User-Centric Approach: Conduct research, surveys, and gather feedback from potential users to understand their expectations and usage habits. Design an interface that satisfies user needs and improves user experience.
- Clean and Intuitive Design: Create a clean, neat, and easy-to-use interface by simplifying elements and arranging features clearly for smooth user experience.
- Responsive Design: Ensure the application is compatible and displays optimally on all devices, including desktops, mobile phones, and tablets, to provide convenient access and interaction across platforms.
- Aesthetically Pleasing Elements: Choose colors, fonts, and images that match the brand and product goals to create an attractive interface that engages users positively.

- Analysis:

- User Requirements Gathering: Conduct conversations, surveys, and interviews with potential users to understand their needs, expectations, and challenges. Gain insights into popular features and important factors for users.
- Technical Feasibility Study: Evaluate technologies, programming languages, databases, and development tools to ensure technical feasibility and effectiveness of the product. Choose optimal solutions based on project requirements and limitations.
- Competitive Analysis: Research competitors in the industry to understand trends, business strategies, strengths, and weaknesses. Optimize strategy to deliver a superior user experience and be competitive in the market.

- Design (Basic and Detailed):
    - Basic Design: Determine the basic structure of the AI Virtual Painter, including key features and interactions between users and the system.
    - Detailed Design: Further detail specific elements of the product, such as user interface wireframes and activity diagrams to specify interactions between components in the system. Prepare for the development process with careful consideration of every detail.
- Using OpenCV and Mediapipe Technology:
    - During development, utilize OpenCV and Mediapipe technology for hand detection and tracking functionalities. OpenCV provides computer vision and image processing capabilities, while Mediapipe offers a comprehensive framework for building perception pipelines.
    - Utilize the capabilities of OpenCV and Mediapipe to create a robust and accurate hand detection and tracking system within the AI Virtual Painter application. This technology choice ensures the application's ability to effectively detect and track hand movements for interactive painting experiences.

6.2 Features include with screenshots

6.2.1 Picture shows hip color

- When I select the pink color with two fingers, I immediately get the pink color and draw it with one finger.

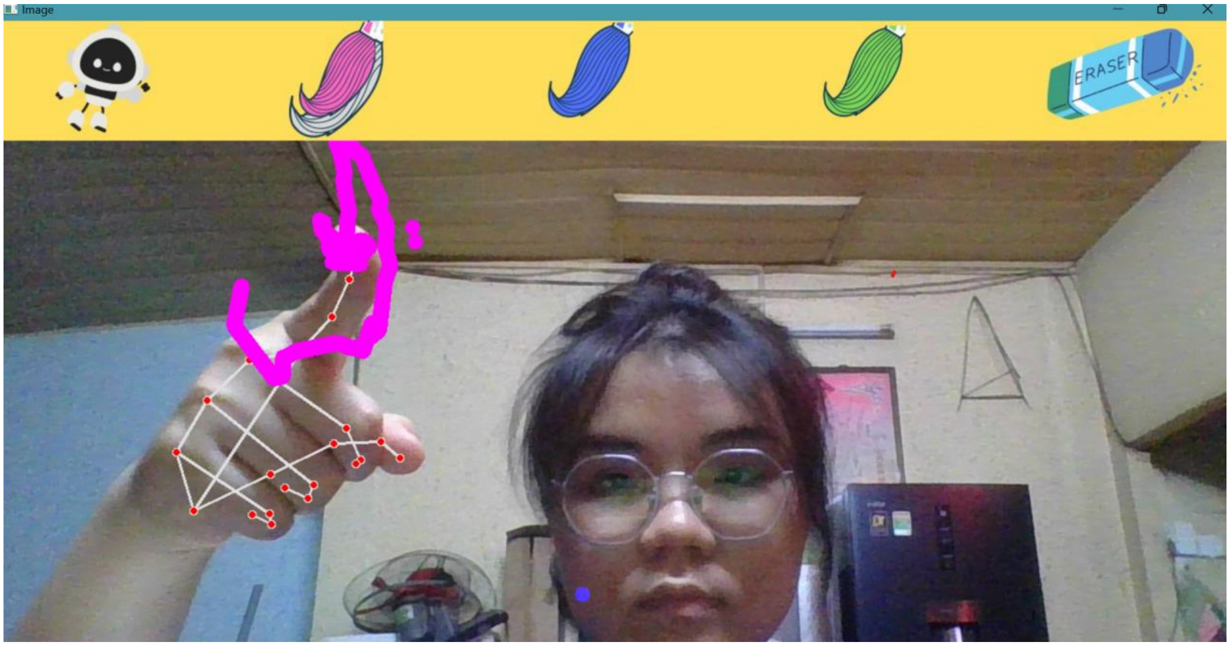


Fig 5: Displayed in pink

6.2.2 Picture shows hip color

- When I select the blue color with two fingers, I immediately get the blue color and draw it with one finger.

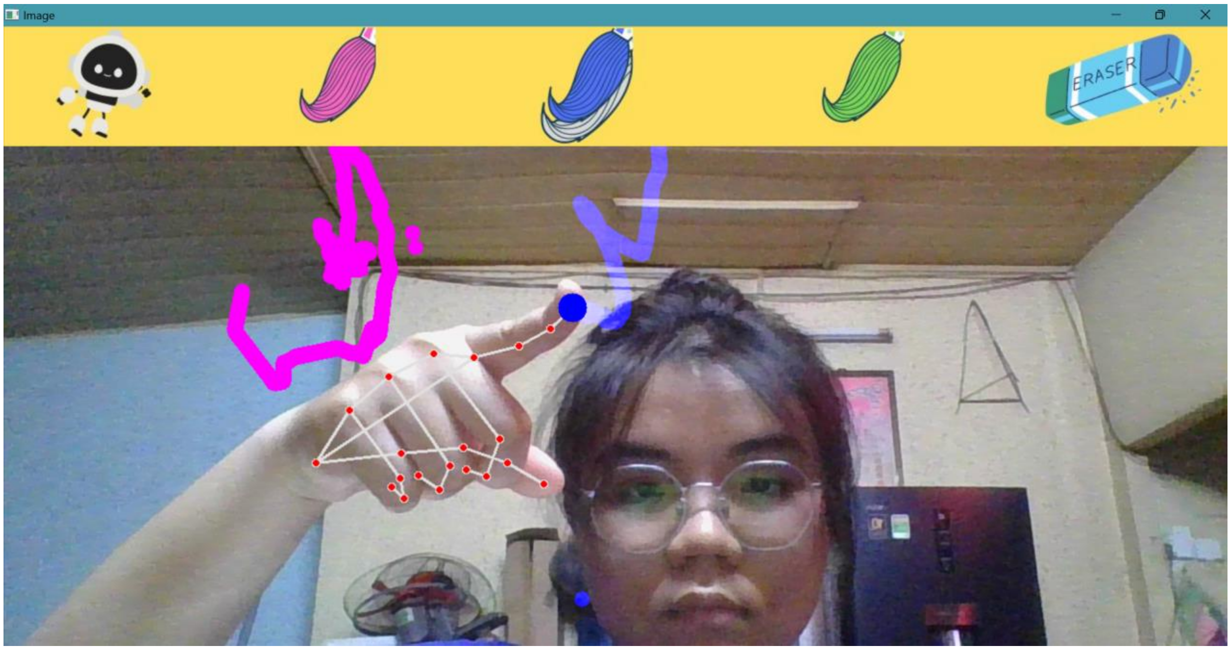


Fig 6: Displayed in blue

6.2.3 Picture shows hip color

- When I select the green color with two fingers, I immediately get the green color and draw it with one finger.

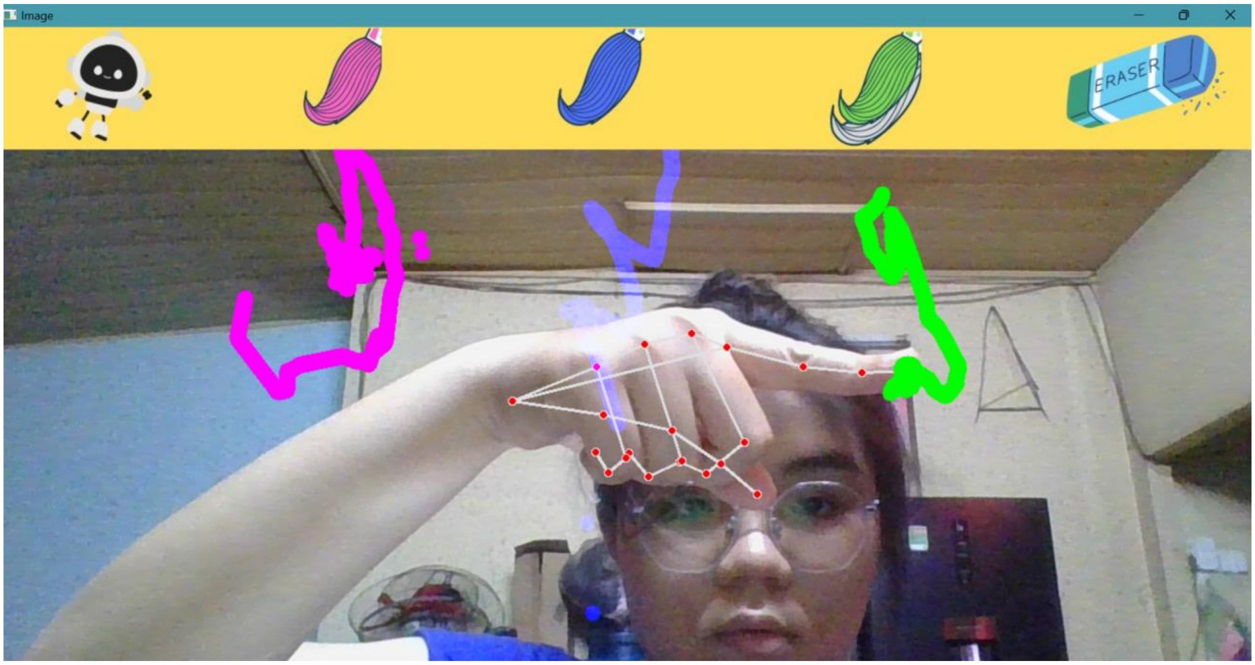


Fig 7: Displayed in green

6.2.3 The image shows the eraser

- The image shows when I use two fingers to select the eraser and a black circle appears

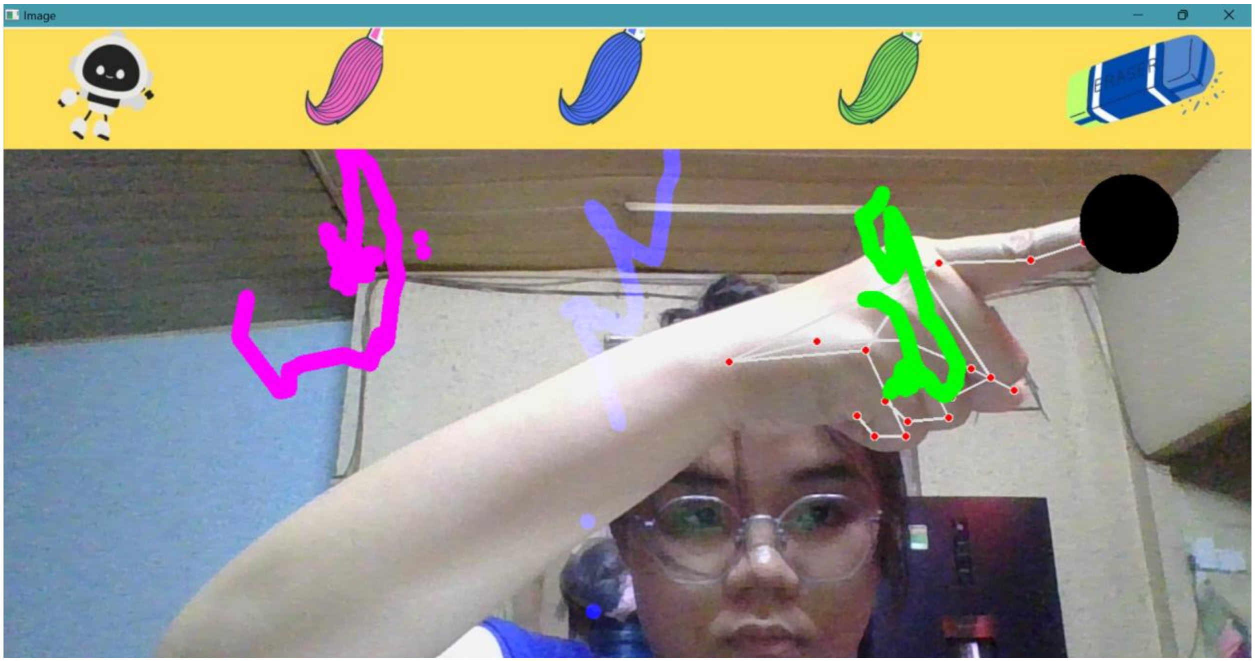


Fig 8: Image of a black eraser

- This image is when I ran my finger over the three colors above to be able to erase the color

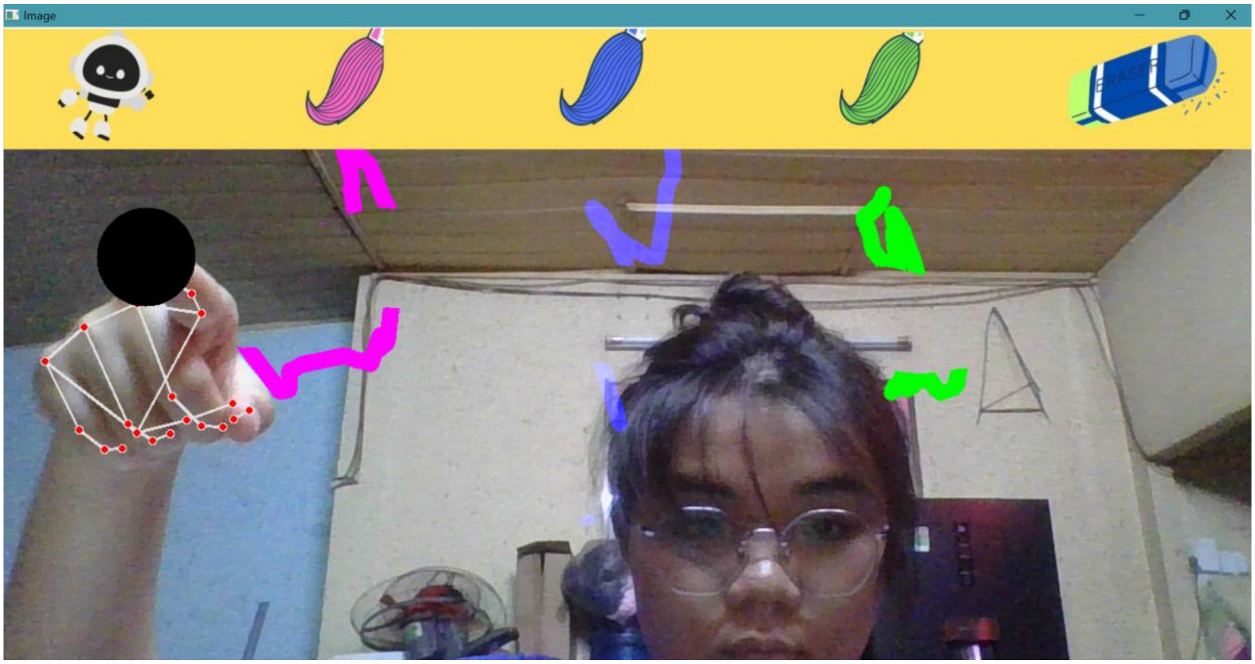
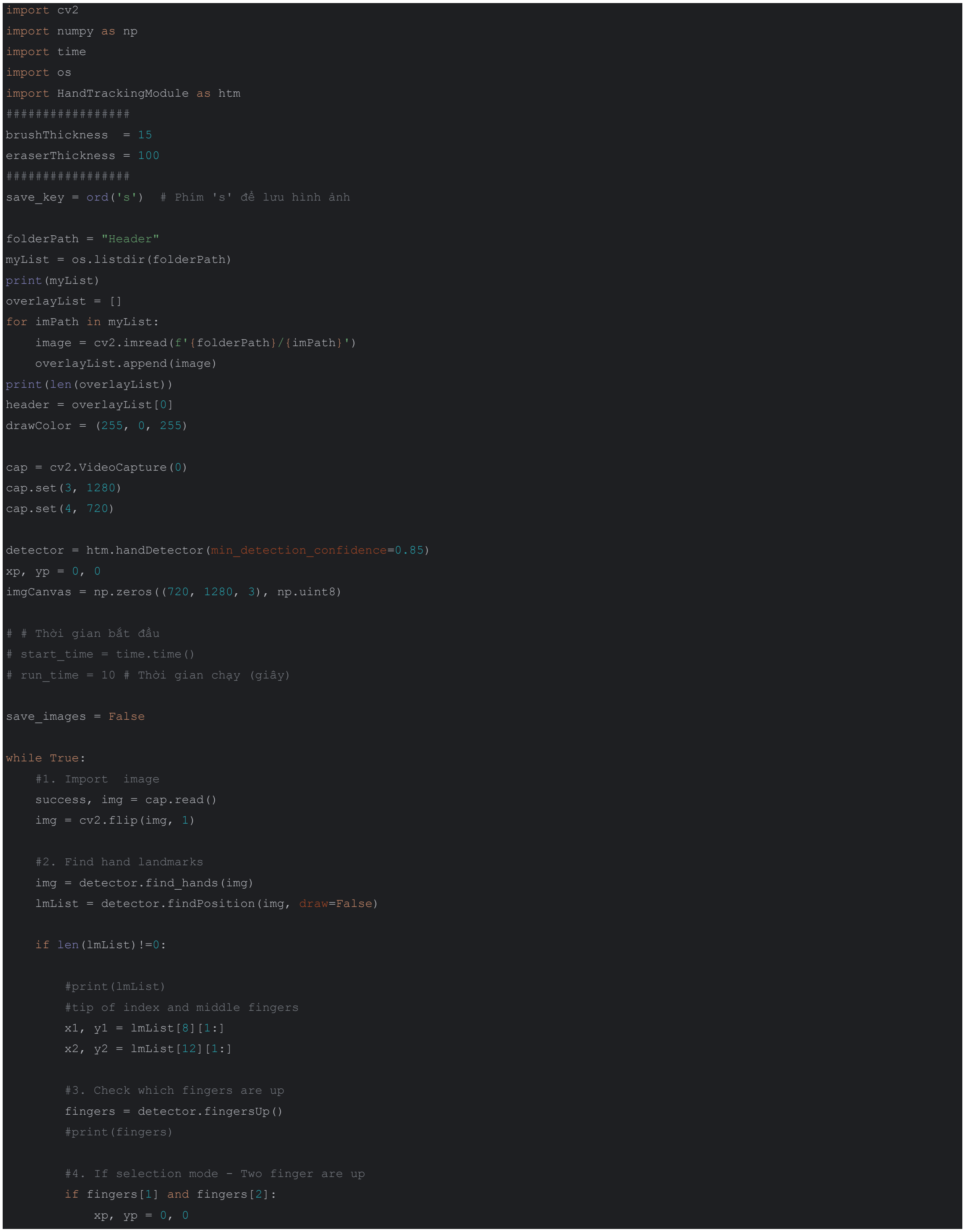


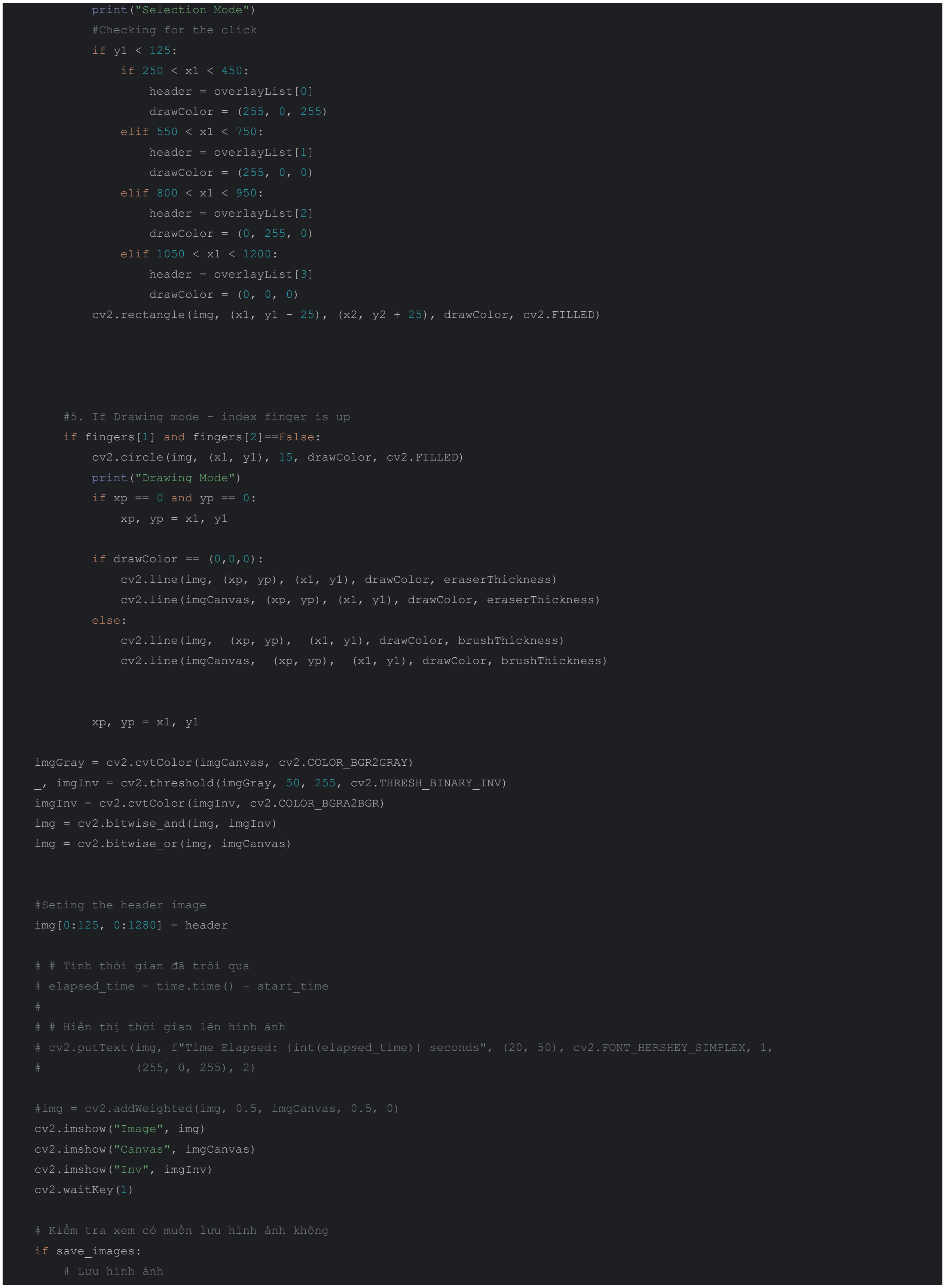
Fig 9: Color has been removed

6.3 Product Implementation

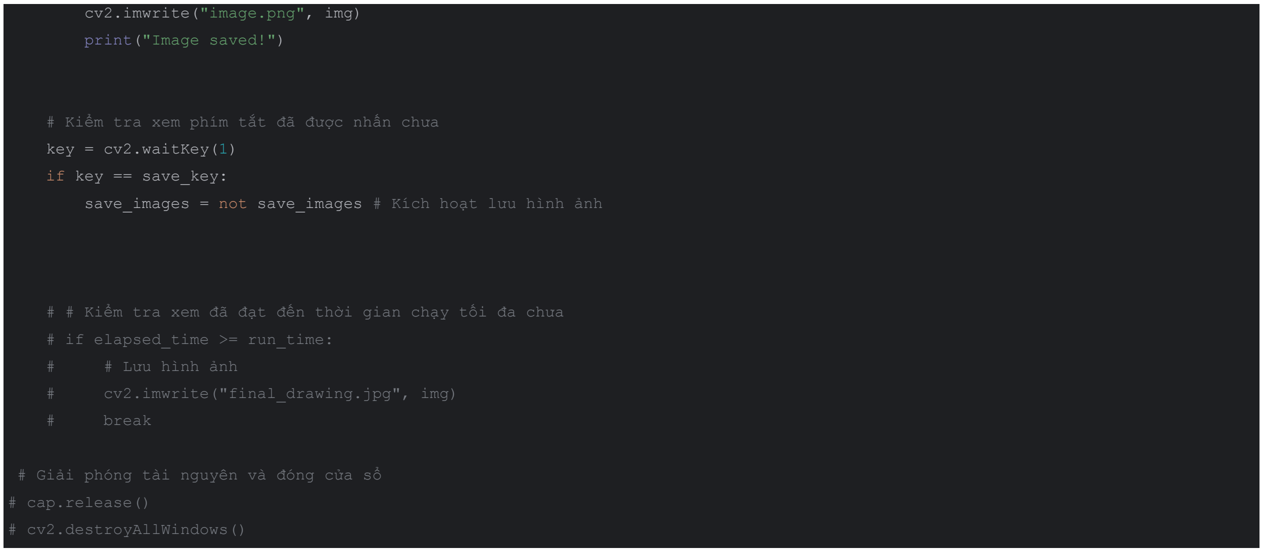
6.3.1 VirtualPainter.py



```
import cv2
import numpy as np
import time
import os
import HandTrackingModule as htm
#################
brushThickness  = 15
eraserThickness = 100
#################
save_key = ord('s')  # Phím 's' để lưu hình ảnh

folderPath = "Header"
myList = os.listdir(folderPath)
print(myList)
overlayList = []
for imPath in myList:
    image = cv2.imread(f'{folderPath}/{imPath}')
    overlayList.append(image)
print(len(overlayList))
header = overlayList[0]
drawColor = (255, 0, 255)

cap = cv2.VideoCapture(0)
cap.set(3, 1280)
cap.set(4, 720)

detector = htm.handDetector(min_detection_confidence=0.85)
xp, yp = 0, 0
imgCanvas = np.zeros((720, 1280, 3), np.uint8)

# # Thời gian bắt đầu
# start_time = time.time()
# run_time = 10 # Thời gian chạy (giây)

save_images = False

while True:
    #1. Import  image
    success, img = cap.read()
    img = cv2.flip(img, 1)

    #2. Find hand landmarks
    img = detector.find_hands(img)
    lmList = detector.findPosition(img, draw=False)

    if len(lmList)!=0:

        #print(lmList)
        #tip of index and middle fingers
        x1, y1 = lmList[8][1:]
        x2, y2 = lmList[12][1:]

        #3. Check which fingers are up
        fingers = detector.fingersUp()
        #print(fingers)

        #4. If selection mode - Two finger are up
        if fingers[1] and fingers[2]:
            xp, yp = 0, 0
```



```
        print("Selection Mode")
        #Checking for the click
        if y1 < 125:
            if 250 < x1 < 450:
                header = overlayList[0]
                drawColor = (255, 0, 255)
            elif 550 < x1 < 750:
                header = overlayList[1]
                drawColor = (255, 0, 0)
            elif 800 < x1 < 950:
                header = overlayList[2]
                drawColor = (0, 255, 0)
            elif 1050 < x1 < 1200:
                header = overlayList[3]
                drawColor = (0, 0, 0)
        cv2.rectangle(img, (x1, y1 - 25), (x2, y2 + 25), drawColor, cv2.FILLED)




    #5. If Drawing mode - index finger is up
    if fingers[1] and fingers[2]==False:
        cv2.circle(img, (x1, y1), 15, drawColor, cv2.FILLED)
        print("Drawing Mode")
        if xp == 0 and yp == 0:
            xp, yp = x1, y1

        if drawColor == (0,0,0):
            cv2.line(img, (xp, yp), (x1, y1), drawColor, eraserThickness)
            cv2.line(imgCanvas, (xp, yp), (x1, y1), drawColor, eraserThickness)
        else:
            cv2.line(img,  (xp, yp),  (x1, y1), drawColor, brushThickness)
            cv2.line(imgCanvas,  (xp, yp),  (x1, y1), drawColor, brushThickness)


        xp, yp = x1, y1

imgGray = cv2.cvtColor(imgCanvas, cv2.COLOR_BGR2GRAY)
_, imgInv = cv2.threshold(imgGray, 50, 255, cv2.THRESH_BINARY_INV)
imgInv = cv2.cvtColor(imgInv, cv2.COLOR_BGRA2BGR)
img = cv2.bitwise_and(img, imgInv)
img = cv2.bitwise_or(img, imgCanvas)


#Seting the header image
img[0:125, 0:1280] = header

# # Tính thời gian đã trôi qua
# elapsed_time = time.time() - start_time
#
# # Hiển thị thời gian lên hình ảnh
# cv2.putText(img, f"Time Elapsed: {int(elapsed_time)} seconds", (20, 50), cv2.FONT_HERSHEY_SIMPLEX, 1,
#             (255, 0, 255), 2)

#img = cv2.addWeighted(img, 0.5, imgCanvas, 0.5, 0)
cv2.imshow("Image", img)
cv2.imshow("Canvas", imgCanvas)
cv2.imshow("Inv", imgInv)
cv2.waitKey(1)

# Kiểm tra xem có muốn lưu hình ảnh không
if save_images:
    # Lưu hình ảnh
```



```
        cv2.imwrite("image.png", img)
        print("Image saved!")


    # Kiểm tra xem phím tắt đã được nhấn chưa
    key = cv2.waitKey(1)
    if key == save_key:
        save_images = not save_images # Kích hoạt lưu hình ảnh



    # # Kiểm tra xem đã đạt đến thời gian chạy tối đa chưa
    # if elapsed_time >= run_time:
    #     # Lưu hình ảnh
    #     cv2.imwrite("final_drawing.jpg", img)
    #     break

 # Giải phóng tài nguyên và đóng cửa sổ
# cap.release()
# cv2.destroyAllWindows()
```

This code is a Python script for creating a simple drawing application using OpenCV. Let's break it down:

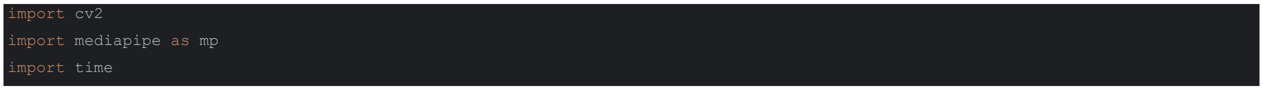
1. Importing Libraries:
    - import cv2: Importing OpenCV for image processing.
    - import numpy as np: Importing NumPy for numerical operations.
    - import time: Importing the time module for time-related operations.
    - import os: Importing the os module for interacting with the operating system.
    - import HandTrackingModule as htm: Importing a custom module named HandTrackingModule, which presumably contains functions for hand detection and tracking.
2. Setting Up Constants:
    - brushThickness = 15: Setting the thickness of the brush used for drawing.
    - eraserThickness = 100: Setting the thickness of the eraser.
    - save_key = ord('s'): Setting the key 's' to save the image.
3. Loading Overlay Images:
    - Loading overlay images from the "Header" folder and storing them in overlayList.
    - Printing the list of overlay images.
    - Setting the initial header image and draw color.
4. Setting Up Video Capture:

- Initializing video capture from the default camera (0).
- Setting the resolution of the video capture to 1280x720 pixels.

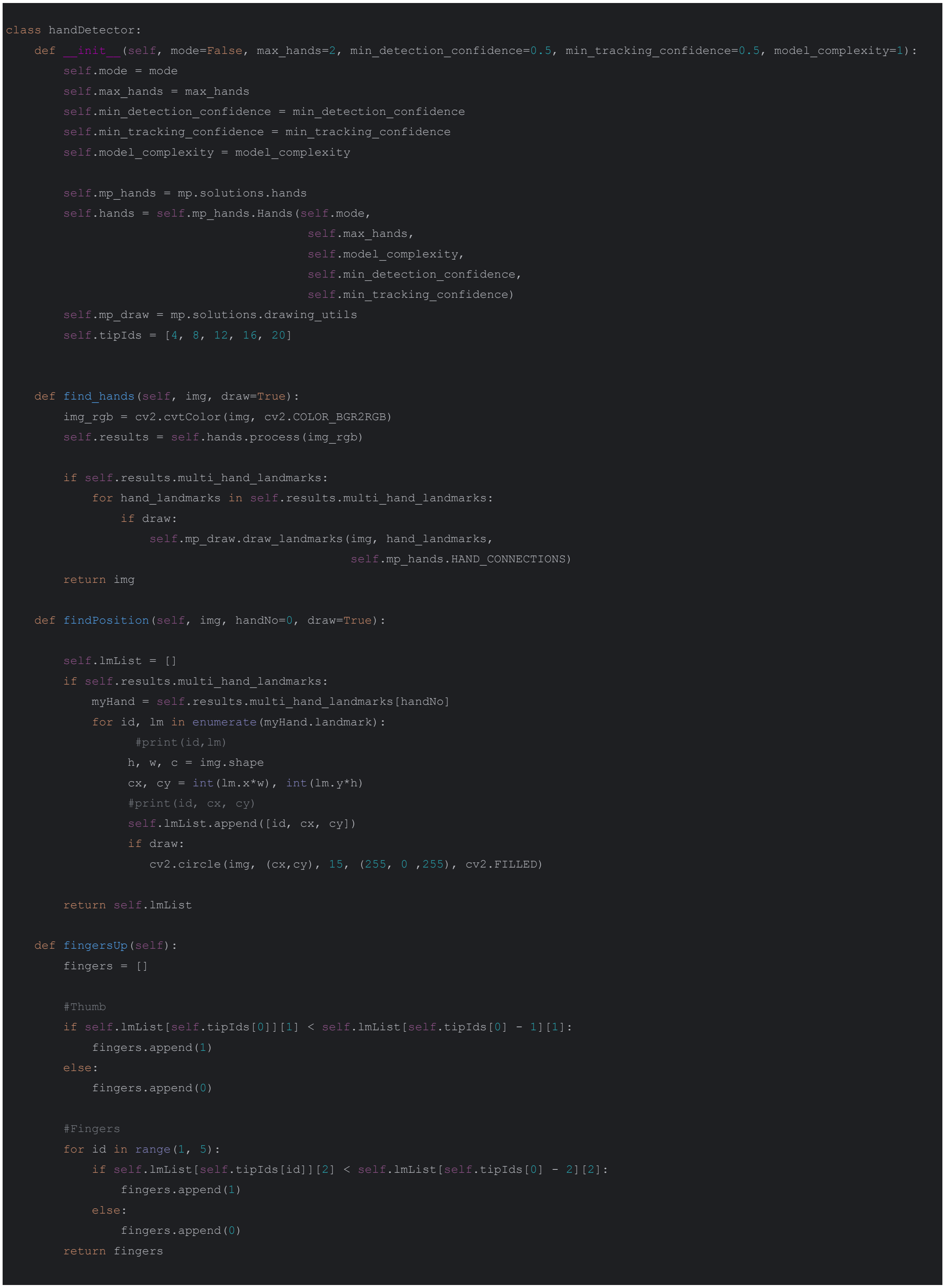
5. Initializing Hand Detector:
   - Creating an instance of the handDetector class from the HandTrackingModule.
   - Setting the minimum detection confidence for hand detection.
6. Initializing Variables:
   - Setting initial values for variables like xp, yp, and imgCanvas.
7. Main Loop:
   - Inside an infinite loop:
     - Reading a frame from the video capture.
     - Flipping the frame horizontally to avoid mirror effect.
     - Detecting hands in the frame and getting landmark positions.
     - Determining the finger positions to check for drawing or selection mode.
     - Drawing based on finger positions and tracking hand movement.
     - Creating a canvas for drawing and displaying it.
     - Handling key presses to save images or exit the program.
8. Displaying Images:
   - Displaying the frame with drawing overlay, canvas, and inverted canvas.
9. Saving Images:
   - Checking if the user wants to save the image by pressing the specified key.
   - If yes, saving the current canvas as an image.
10. Exiting the Program:
    - Exiting the loop and releasing resources when the program is terminated.

This code essentially creates a drawing application where users can draw on a canvas using their fingers detected via a webcam. It provides features like changing colors and saving the drawings as images.

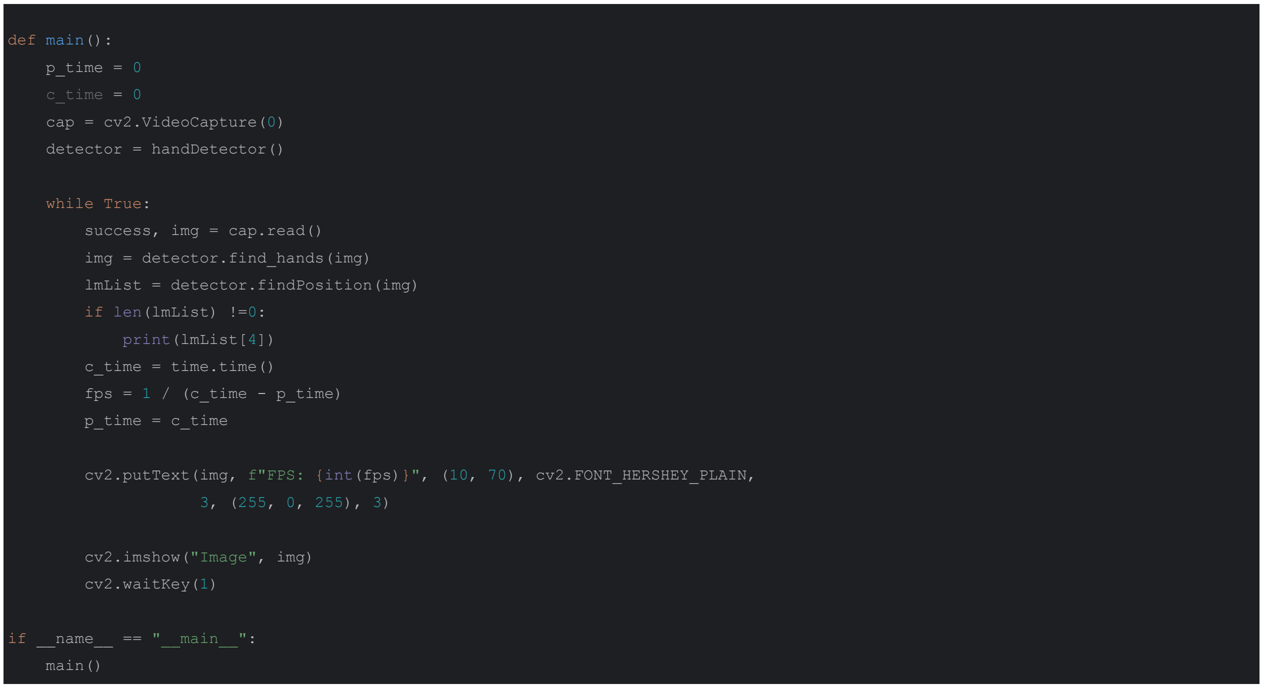
6.3.2 HandTrackingModule.py



```
import cv2
import mediapipe as mp
import time
```



```
class handDetector:
    def __init__(self, mode=False, max_hands=2, min_detection_confidence=0.5, min_tracking_confidence=0.5, model_complexity=1):
        self.mode = mode
        self.max_hands = max_hands
        self.min_detection_confidence = min_detection_confidence
        self.min_tracking_confidence = min_tracking_confidence
        self.model_complexity = model_complexity

        self.mp_hands = mp.solutions.hands
        self.hands = self.mp_hands.Hands(self.mode,
                                          self.max_hands,
                                          self.model_complexity,
                                          self.min_detection_confidence,
                                          self.min_tracking_confidence)
        self.mp_draw = mp.solutions.drawing_utils
        self.tipIds = [4, 8, 12, 16, 20]


    def find_hands(self, img, draw=True):
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(img_rgb)

        if self.results.multi_hand_landmarks:
            for hand_landmarks in self.results.multi_hand_landmarks:
                if draw:
                    self.mp_draw.draw_landmarks(img, hand_landmarks,
                                                self.mp_hands.HAND_CONNECTIONS)
        return img

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

        self.lmList = []
        if self.results.multi_hand_landmarks:
            myHand = self.results.multi_hand_landmarks[handNo]
            for id, lm in enumerate(myHand.landmark):
                  #print(id,lm)
                 h, w, c = img.shape
                 cx, cy = int(lm.x*w), int(lm.y*h)
                 #print(id, cx, cy)
                 self.lmList.append([id, cx, cy])
                 if draw:
                    cv2.circle(img, (cx,cy), 15, (255, 0 ,255), cv2.FILLED)

        return self.lmList

    def fingersUp(self):
        fingers = []

        #Thumb
        if self.lmList[self.tipIds[0]][1] < self.lmList[self.tipIds[0] - 1][1]:
            fingers.append(1)
        else:
            fingers.append(0)

        #Fingers
        for id in range(1, 5):
            if self.lmList[self.tipIds[id]][2] < self.lmList[self.tipIds[0] - 2][2]:
                fingers.append(1)
            else:
                fingers.append(0)
        return fingers
```



```
def main():
    p_time = 0
    c_time = 0
    cap = cv2.VideoCapture(0)
    detector = handDetector()

    while True:
        success, img = cap.read()
        img = detector.find_hands(img)
        lmList = detector.findPosition(img)
        if len(lmList) !=0:
            print(lmList[4])
        c_time = time.time()
        fps = 1 / (c_time - p_time)
        p_time = c_time

        cv2.putText(img, f"FPS: {int(fps)}", (10, 70), cv2.FONT_HERSHEY_PLAIN,
                    3, (255, 0, 255), 3)

        cv2.imshow("Image", img)
        cv2.waitKey(1)

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

The following code is a Python program that uses the OpenCV and MediaPipe libraries to detect and draw landmark points on the hand from a live video stream provided by the webcam.

1. Importing libraries:
    - import cv2: OpenCV library for image and video processing.
    - import mediapipe as mp: MediaPipe library, used for hand landmark detection and tracking.
2. Class handDetector:
    - __init__: Initialization function to set parameters for hand detection.
    - find_hands: Method to find and draw landmark points on the hand in the image.
    - findPosition: Method to find the position of landmark points on the hand.
    - fingersUp: Method to determine whether fingers are up or not.
3. Function main:
    - Sets up the video stream from the webcam.
    - Creates a handDetector object.
    - Inside an infinite loop:
        - Reads video frames from the webcam.

  - Uses handDetector to find and draw landmark points on the hand in the image.
  - Computes and displays the Frames Per Second (FPS).
  - Displays the image in a window.

4. Check if the program is being run as a standalone script:
   - If the program is being run as a standalone script (__name__ == "__main__"), the main() function is called to execute the program.

This program uses the MediaPipe hand detection algorithm to detect and draw landmark points on the hand in real-time from the video provided by the webcam.

6.4 Evaluation of your product

1. Product reviews:

- The AI Virtual Painter project represents a significant achievement, embodying a blend of commendable strengths and areas ripe for improvement. Through a comprehensive evaluation, we can discern the project's positive attributes while identifying key areas that warrant further attention and enhancement.

2. Positive aspects:

- User-centric design: One of the project's standout features is its user-centric design philosophy. The user interface has been meticulously crafted to prioritize user needs, resulting in an intuitive and seamless user experience. By employing principles of user-centered design, the project ensures that users can effortlessly navigate through the application, select tools, and unleash their creativity with ease. The interface's responsiveness across a diverse range of devices further underscores its accessibility and user-friendliness, catering to a broad audience of users.

- Functional robustness: AI Virtual Painter showcases robust functionality, effectively fulfilling its intended purposes while adhering closely to the initial design objectives. The project encompasses a comprehensive suite of features, ranging from brush selection and color customization to canvas manipulation and artwork sharing. Each feature has been thoughtfully implemented to provide users with a versatile and engaging painting experience. Furthermore, the project's adherence to functional requirements and its ability to seamlessly integrate various painting tools contribute to its overall effectiveness and utility.

- Technological sophistication: The project leverages cutting-edge technologies, including computer vision and machine learning algorithms, to deliver an innovative and compelling painting platform. By harnessing the power of computer vision, the project enables users to interact with virtual painting tools using hand gestures, revolutionizing the traditional painting experience. Additionally, machine learning algorithms facilitate intelligent brushstroke prediction and color recommendation, enhancing the user's creative process and enabling the generation of stunning artworks. The seamless integration of these technologies underscores the project's technological sophistication and its ability to push the boundaries of digital artistry.

3. Areas needing improvement:

- Enhanced User Interaction: While the project excels in functionality, there remains an opportunity to elevate the user interaction aspect further. Implementing additional interactive elements, such as customizable brush behaviors or dynamic canvas effects, could enhance user engagement and foster a more immersive painting experience. By incorporating interactive features that respond intuitively to user inputs, the project can elevate the overall level of interactivity and creativity, thereby enhancing user satisfaction and retention.

- Performance optimization: Continuous optimization of performance metrics is essential to ensure a smooth and responsive painting experience. While the project currently delivers on functionality, optimizing aspects such as processing speed, real-time rendering performance, and system responsiveness can further enhance the user experience. By prioritizing performance optimization efforts, the project can minimize latency, reduce loading times, and deliver a seamless painting experience across different devices and usage scenarios.

- Market adaptation: Exploring strategies for market adaptation and conducting in-depth market analysis are crucial steps in maximizing the project's impact and reach. Understanding user preferences, market trends, and competitive dynamics can provide valuable insights for refining the project's feature set, tailoring marketing strategies, and identifying opportunities for differentiation. By continuously monitoring market dynamics and aligning the project with evolving user needs, the project can remain relevant and competitive in a rapidly evolving digital landscape.

4. Future developments:

- Introducing product delivery functionality represents an exciting opportunity for future expansion and enhancement. By enabling users to export their digital artworks in various formats, share them on social media platforms, or even order prints of their creations, the project can extend its utility and appeal to a broader audience. Additionally, exploring collaborative painting features, virtual art galleries, or integration with online art communities could further enrich the project's ecosystem and foster a vibrant community of digital artists.

5. Conclusion:

- In conclusion, AI Virtual Painter stands as a testament to innovation, creativity, and technological prowess. While the project boasts several commendable strengths, including its user-centric design, functional depth, and technological sophistication, there are areas that can be further refined and improved. By prioritizing enhancements in user interaction, performance optimization, and market adaptation, the project can continue to evolve and thrive in the competitive landscape of digital artistry. Through continuous iteration, analysis of user feedback, and strategic innovation, AI Virtual Painter is poised to achieve enduring success and captivate digital artists worldwide.

Chapter 7 Conclusion:

Through the development of the AI Virtual Painter project, I have acquired valuable insights into the intersection of artificial intelligence and creative applications. This project has provided me with a deeper understanding of hand detection and tracking algorithms, as well as the implementation of responsive and user-centric design principles in crafting intuitive user interfaces.

7.1 What did you learn from this project?

- Through this project, I gained valuable insights into the integration of artificial intelligence (AI) technologies, such as computer vision and machine learning, into creative applications like the AI Virtual Painter. I learned about the intricacies of hand detection and tracking algorithms using OpenCV and Mediapipe, and how to implement them effectively to enable users to interact with the virtual painting canvas. Additionally, I deepened my understanding of user-centric design principles and the importance of responsive and intuitive user interfaces in enhancing user experience.

7.2 What are the results of this project?

- The project yielded a functional AI Virtual Painter application capable of detecting and tracking hand movements in real-time, allowing users to create digital artworks through intuitive gestures. The application's user interface was designed to be clean, responsive, and aesthetically pleasing, providing users with a seamless painting experience across various devices. Furthermore, the project showcased the potential of AI technologies in enhancing creativity and artistic expression in digital mediums.

7.3 Continue to develop this project

To further develop this project, several avenues can be explored:

- Enhancing the accuracy and robustness of hand detection and tracking algorithms to improve the responsiveness and precision of the painting interactions.
- Introducing additional features and tools to expand the creative capabilities of the AI Virtual Painter, such as different brush styles, color palettes, and advanced editing options.
- Integrating cloud-based services to enable collaborative painting sessions and the sharing of artwork among users.
- Conducting user testing and gathering feedback to iteratively refine the application based on user preferences and needs.
- Exploring opportunities for integration with other AI technologies, such as style transfer algorithms, to provide users with innovative ways to create and customize their digital artworks.

References

1. Ijraset (no date) *IJRASET Journal for Research in Applied Science and Engineering Technology, Computer Vision Based Virtual Sketch Using Detection*. Available at: https://www.ijraset.com/research-paper/computer-vision-based-virtual-sketch-using-detection (Accessed: 28 April 2024).
2. (No date a) *(PDF) Deep Learning-based real-time AI virtual mouse system using computer vision to avoid COVID-19 spread*. Available at: https://www.researchgate.net/publication/355586801_Deep_Learning-Based_Real-Time_AI_Virtual_Mouse_System_Using_Computer_Vision_to_Avoid_COVID-19_Spread (Accessed: 28 April 2024).
3. Ramadhan, G., Wiratama, J. and Permana, A.A. (no date) *Development of virtual painting method using opencv library with finger gesture on online learning platform, JIKA (Jurnal Informatika)*. Available at: https://jurnal.umt.ac.id/index.php/jika/article/view/6875 (Accessed: 28 April 2024).
4. GeeksforGeeks (2023) *Face and hand landmarks detection using Python - Mediapipe, opencv, GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/face-and-hand-landmarks-detection-using-python-mediapipe-opencv/ (Accessed: 28 April 2024).
5. Murali, G.G. *et al.* (2023) *An AI based virtual drawing system using Python, International Journal of Engineering Research & Technology*. Available at: https://www.ijert.org/an-ai-based-virtual-drawing-system-using-python (Accessed: 28 April 2024).
6. (No date a) *OpenCV documentation index*. Available at: https://docs.opencv.org/ (Accessed: 28 April 2024).
7. (No date a) *Mediapipe.dev*. Available at: https://mediapipe.dev/ (Accessed: 28 April 2024).
8. (No date a) *NumPy documentation*. Available at: https://numpy.org/doc/ (Accessed: 28 April 2024).
9. *Python 3.12.3 documentation* (no date) *3.12.3 Documentation*. Available at: https://docs.python.org/3/ (Accessed: 28 April 2024).
10. (No date a) *Computer vision: Algorithms and applications, 2nd ed.* Available at: https://szeliski.org/Book/ (Accessed: 28 April 2024).
11. *Learn Computer Vision, deep learning, and opencv for free...* (no date) *PyImageSearch*. Available at: https://pyimagesearch.com/free-opencv-computer-vision-deep-learning-crash-course/ (Accessed: 28 April 2024).
12. *Python for Data Analysis, 3E* (no date) *Wes McKinney*. Available at: https://wesmckinney.com/book/ (Accessed: 28 April 2024).
13. (No date) *Object-oriented programming in python: Goldwasser, Michael H., Letscher, David: 9780136150312: Amazon.com: Books*. Available at: https://www.amazon.com/Object-Oriented-Programming-Python-Michael-Goldwasser/dp/0136150314 (Accessed: 28 April 2024).
14. (No date a) *OpenCV: Computer vision projects with python*. Available at: https://hlevkin.com/hlevkin/45pythonImageProc/Howse-Joshi-Beyeler_opencv_computer_vision_projects_with_python.pdf (Accessed: 28 April 2024).
15. (No date a) *Google*. Available at: https://quickdraw.withgoogle.com/ (Accessed: 28 April 2024).

16. Corporation, M. (no date) *Paint 3D - official app in the Microsoft Store*, *Microsoft Apps*. Available at: https://www.microsoft.com/en-us/p/paint-3d/9nblggh5fv99 (Accessed: 28 April 2024).
17. (No date a) *Digital painting and drawing app | adobe fresco*. Available at: https://www.adobe.com/products/fresco.html (Accessed: 28 April 2024).
18. *Tilt Brush* (no date) *Tilt Brush by Google*. Available at: https://www.tiltbrush.com/ (Accessed: 28 April 2024).
19. Royce, W.W., Profile, V. and Metrics, O.M.A. (1987) *Managing the development of large software systems: Concepts and techniques: Proceedings of the 9th International Conference on Software Engineering*, *ACM Conferences*. Available at: https://dl.acm.org/doi/10.5555/41765.41801 (Accessed: 29 April 2024).
20. (No date) *A spiral model of software development and ...* Available at: https://www.cse.msu.edu/~cse435/Homework/HW3/boehm.pdf (Accessed: 29 April 2024).
21. (No date a) *(PDF) introduction to software reuse*. Available at: https://www.researchgate.net/publication/250034287_Introduction_to_Software_Reuse (Accessed: 29 April 2024).
22. (No date a) *Manifesto for Agile Software Development*. Available at: https://www.researchgate.net/profile/Godfried-Adaba/post/Looking-for-papers-on-the-software-development-process-Any-recommendations-will-be-appreciated/attachment/59d639e579197b8077997172/AS:403742915612673@1473271220194/download/Manifesto+of+Agile+Software+Development.pdf (Accessed: 29 April 2024).