**Meclawnism: Gesture-based Mechanics for Claw Machine Game**

Mohamad Razlan Nazmi Ramli1, a), Quek Albert1, b), Goh Hui Ngo1, c), Ting Choo Yee1, d)

1*Faculty of Computing and Informatics, Multimedia University, Persiaran Multimedia, 63100 Cyberjaya, Selangor, Malaysia*

*b)Corresponding author: quek.albert@mmu.edu.my*

*a) razlannazmi@gmail.com*

*c)hngoh@mmu.edu.my*

*d)cyting@mmu.edu.my*

**Abstract.** This paper presents *Meclawnism*, a gesture-based PC game that reimagines the classic claw machine using hand gesture recognition as its primary input method. It is constructed using Unity3D and Python-based MediaPipe model to allow dual-hand input as a basic game-mechanics for item collection and obstacle demolition. There are ten custom action gestures mapped in creating a novel bimanual gameplay experience. A total of 12 participants evaluated the game using structure questionnaires. Results show that the game achieved high level of engagement and emotional satisfaction, despite of initial challenges in mastering the gesture mechanics. It demonstrates the feasibility and application of gesture-based input for immersive casual gaming experiences.

# **INTRODUCTION**

As interactive technology continues to evolve, video games have emerged not only as a form of entertainment but also as platforms for cognitive development, rehabilitation, and social commentary [1],[2]. Among the many advancements, the integration of natural user interfaces—such as voice, motion, and gesture recognition—has significantly enriched gameplay experiences.

Gesture recognition, offers an intuitive and immersive way for users to interact with virtual environments, assisting them with tasks spanning from routine exercise to performing yoga exercise [3]. Gamification of these advancements provides a new landscape for researchers to explore. Combining with gesture recognition, researchers explore the possibilities to improve left brain thinking using Virtual Reality Technology [4].

In this study, we explore the application of hand gesture recognition in casual PC gaming through the development of *Meclawnism*, a claw machine-themed game controlled entirely by dual-hand gestures. Using the MediaPipe framework and Unity3D, we implement a custom system capable of recognizing ten distinct hand gestures, each mapped to core game mechanics. The project aims to investigate the feasibility of using accessible and affordable hardware to support engaging, gesture-driven gameplay.

This paper contributes (i) the design and implementation of a dual-hand gesture-controlled game, (ii) a real-time integration framework between Python and Unity via socket programming, and (iii) a user evaluation based on gameplay experience, system functionality, and learning curve difficulty.

# **RELATED WORKS**

The working principles of a hand gesture recognition sensor are data acquisition, pre-processing, feature extraction, classification, and post-processing [5]. The implementation of gestures recognition systems is not a totally new concept, as some previous researchers have already done some experimentation in developing gestures recognition systems implemented on games using frameworks with a combination of libraries and packages.

Kumar et al. [6] and Patel et al. [7] implemented different methods of gesture recognition system on two games, which are Subway Surfers and Hill Climb Racing. Created using OpenCV and MediaPipe, the methods of hand gesture recognition system created by Kumar et al. [6] include image capturing and applying scaling, rotation or translation if needed, image processing, feature extraction, classification using artificial neural networks (ANN), and evaluation [6]. Patel et al. [7] utilized MediaPipe and ML Kit Pose Detection in their system, involving the process of image capture, hand segmentation, noise reduction & edge detection, and gesture feature extraction. The algorithm can used for gesture feature extraction includes Support Vector Machine (SVM), Hidden Markov model (HMM), Histogram of oriented gradient (HOG), and deep neural network [7]. Rashmika et al. [8] used PyCharm, OpenCV, MediaPipe, and Numpy to create a hand detection system together with a two player ping pong game, where two ping pong pad is controlled by the right hand of one player, and the left hand of the other player. Methods used from the packages include Imread, Imshow, Flip, and VideoCapture [8]. The gesture recognition system developed by Saman & Stanciu [9] operates on OpenCV and is used to recognize nine different hand mobility exercise’s gestures, while the Tetris game is developed using Construct2 Engine. Each of the gestures has their own corresponding virtual keypress command, which is a keyboard press action equivalent in the game. The system’s recognition component works on several procedural principles, which are image acquisition, image filtering, image segmentation through a thresholding process to detect the hand’s pixels, determination of hand contour, selection of valid features, and recognition [9].

Hand gesture recognition models can be created using the concept of machine learning, with algorithms such as convolutional neural networks (CNNs) process image or sensor data to accurately classify gestures [10]. MediaPipe framework can be used to create an end-to-end perception pipeline, acting as a graph of modular components, that process any kind of sensory data in order to perform computer vision inference. [11]. The main architectures of this framework are MediaPipe’s graph, which are visual or textual representation of a pipeline, and calculator, which is the basic processing unit in a MediaPipe graph. Each calculator is built to carry out a particular function and can be coupled with other calculators to form intricate pipelines that reduces development time and enhances maintainability [12].

# **GAME CONCEPT**

This game, named *Meclawmism*, is designed to implement the main concept of a claw machine as the main theme. The player will have to control a claw-machine-like system to complete missions (see Figure 1). The game design is set to be less complicated, easy to learn with clear and simple guidelines, and balanced between controlling skills and in-game challenges since mastering the controls will provide the player with an uncommon learning curve experience.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| (a) | (b) | (c) | (d) |

**FIGURE 1.** (a) HexGear claw structure design, (b) HexGear claw model design, (c) HexGear hammer structure design, (d) HexGear hammer model design

The HexGear Mechanical System is built up of two parts, which are the Arm and Hand, connected with a joint. The Hand parts are different for each mode.

The HexGear Claw mode is equipped with the ability to relocate collectables into the designated Target Box, controlled by the player’s right hand gestures inputs. This HexGear Claw mechanic consists of two phases. The first phase is the ‘Grabbing’ phase, which is lowering the system’s floating height. Upon reaching the lowest height, the claw will close automatically, trapping a collectible inside it, while preventing any movements. The second phase is the ‘Lifting’ phase, which involves lifting up the system back to its original height while preserving the claw in a closed state, and reenables movements. If no action gestures are performed, the system will return into its idle state. The HexGear Hammer is equipped with a shockwave blasting ability to demolish destructible obstacles. The HexGear Hammer mechanic consists of two phases. The first phase is the 'Charging' phase, in which the system will increase gradually to its floating height limit. The second phase is the ‘Blasting’ phase, which is blasting the shockwave while rapidly lowering the system height if the system is fully charged. Otherwise, the system will only return to its idle state.

## **Controls**

The hand gesture recognition system has been trained to detect, recognize and distinguish sets of gesture patterns performed by both right and left hand simultaneously (see Table 1). The system can move in four horizontal directions, controlled by only the left hand’s gestures inputs as depicted in Figure 2. Meanwhile, modes’ abilities can only be done by right hand’s gestures inputs.

|  |  |  |
| --- | --- | --- |
| **TABLE 1.** Implemented hand gestures, corresponding in game actions and sample gesture images | | |
| **Hand gesture** | **Hand** | **In-game actions** |
| Open | Both | No corresponding actions, used as a standby gesture. |
| Close | Both | No corresponding actions, used as a standby gesture. |
| Pointer-up | Left | Move the system forward. |
| Pointer-down | Left | Move the system backward. |
| Thumb-out right | Left | Move the system to the right. |
| Thumb-out left | Left | Move the system to the left. |
| Five-fingers pinch | Right | Claw’s items grabbing mechanism:   * Lower the system. Upon reaching the lowest height limit, close the claw.   Transformation platform hitting mechanics:   * Lower the system to hit the transfiguration platform. |
| Two-fingers  pinch | Right | Claw’s items lifting mechanics:   * Lift up the system to its highest height limit without opening the claw. |
| Rock sign | Right | Hammer’s charging mechanics:   * Lift up the system. Upon reaching the highest height limit, effects applied indicating the hammer is fully charged. |
| Fox sign | Right | Hammer’s shockwave blasting:   * In fully-charged state, rapidly lower the system and create a shockwave.   Transformation platform hitting mechanics:   * In normal state, lower the system to hit the transfiguration platform. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| (a) | (b) | (c) | (d) | (e) |
|  |  |  |  |  |
| (f) | (g) | (h) | (i) | (j) |

**FIGURE 2.** (a) Open sign, (b) Close sign, (c) Pointer-up sign, (d) Pointer-down sign, (e) Thumb-out right sign, (f) Thumb-out left sign, (g) Five-fingers pinch sign, (h) Two-fingers pinch sign (i) Rock sign, (j) Fox sign

## **Game Structure**

Four game modes with different levels of challenges have been developed in this game as depicted in Figure 3. Tutorial game mode is the onboarding level intended to guide the players, especially new players, to learn the controls of the in-game mechanics, understand the gameplay, and be exposed to the elements in the games. Simulation game mode is inspired from the experience of playing an arcade claw machine. In this scene, the player will need to operate the HexGear Claw in the playable area designed similar to a claw machine structure. Arcane Labyrinth game mode is designed with the motivation to provide an adventure experience centered around the claw machine concept. This game mode provides various challenges for the players to face in a dungeon style area, with multiple rooms and elements. Kindergarten game mode is the playground scene for the players to practice the hand gestures mechanics. This level does not contain any mission for the player to complete. Figure 4 shows the mode of the game.

A black screen with white text

AI-generated content may be incorrect.

**FIGURE 3.** Game structure flowboard

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| (a) | (b) | (c) | (d) |

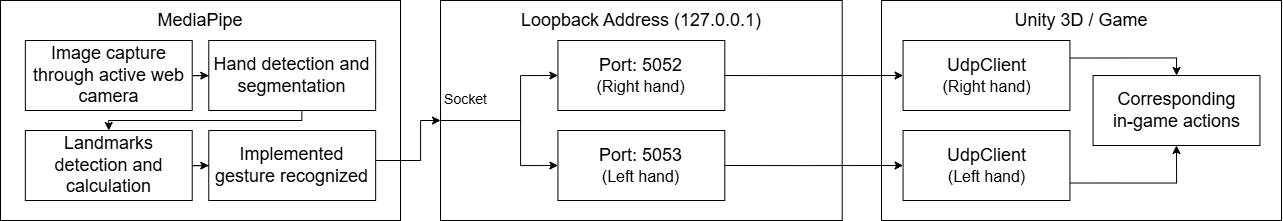
**FIGURE 4.** (a) Tutorial game mode, (b) Simulation game mode, (c) Arcane Labyrinth game mode, (d) Kindergarten game mode

# **FRAMEWORK AND IMPLEMENTATION**

Three softwares are selected to be used, which are Visual Studio Code for the hand gesture recognition system development, Unity 3D for the game development and Visual Studio 2022 for game’s C# scripting.

For this project, a pre-trained model, created using the MediaPipe framework and Python libraries, is used. The original repository was firstly created by Kazuhito Takahashi, fully in Japanese. The translated version of this program is created by Nikita Kiselov. This model requires the machine to have MediaPipe, OpenCV, Tensorflow, tf-nightly, scikit-learn, and matplotlib of specific version installed [13]. Based on each frame captured through the web camera, any hand detected will be segmented and implemented with 21 hand landmarks, based on MediaPipe’s hand detection module. Each landmark point holds a coordinate for the calculation of distance and ratio between each point to determine the performed type of gesture.

The method used to integrate between the game and model is by transferring out the model’s output to the local host address in the computer for the game engine to listen to. From the model, socket programming, which is a method that allows communication on a network connection between two nodes, is used [14]. Using the imported library, a socket object is created and two ports are declared, one for each hand output. Both ports are listened to in every frame using User Datagram Protocol (UDP), a simple protocol used to deliver data to a remote host [15]. Any captured recognition data, in arrays of bytes, will then be encoded into strings to initiate game actions. Figure 5 shows the integration architecture.



**FIGURE 5.** Integration architecture diagram

# **TESTING AND RESULTS**

A closed beta testing session was conducted on 29 June 2024 at the Entrepreneurship Development Centre (EDC), Multimedia University, Cyberjaya (see Figure 6). The session involved 12 participants who were introduced to the game concept, unique mechanics, and gesture-based control system. Each participant spent approximately 15–20 minutes playing through various game modes before completing a structured feedback form via a QR code linked to a Google Form. The evaluation instrument combined three established survey models: the Gameplay Experience Questionnaire (GEQ), the Gameplay Features Questionnaire (GFQ), and the Functionality Testing Questionnaire (FTQ). All questions were close-ended with Likert-scale responses, except for the final open-ended feedback prompt.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| (a) | (b) | (c) |

**FIGURE 6.** (a), (b) and (c) photos of beta testing session

The GEQ assessed emotional and cognitive aspects of gameplay across seven components: Positive Affect, Negative Affect, Competence, Sensory Immersion, Flow, Tension, and Challenge. Based on the GEQ’s results, participants reported high scores in Positive Affect and Sensory Immersion, indicating that the game was enjoyable and visually engaging, even for those unfamiliar with gesture-based controls. Notably, Flow and Challenge scores were both high, suggesting that the game maintained a balanced level of difficulty that kept players focused and motivated. Moderate Tension and Competence scores implied that while players felt challenged, they did not experience excessive frustration, although some struggled initially with mastering the gesture inputs. The GFQ focused on design aspects, including learning curve, fun factor, challenge, game completeness, and overall gameplay experience. The Fun Factor and Overall Gameplay categories for this project received consistently high ratings. However, the Learning Curve was rated moderately, highlighting the steep entry barrier due to the novelty of gesture-based input. Several participants expressed that while the game was enjoyable, it required significant effort to internalize and consistently perform the gestures correctly. The FTQ evaluated technical reliability, particularly the performance of the gesture recognition model and system responsiveness. The developed gesture recognition system was rated as generally reliable, with minimal lag or misinterpretation during most gameplay. That said, some gestures—particularly those involving finer finger articulation—were occasionally misclassified, especially under varying lighting conditions. Players who struggled early on found their performance improving after 5–10 minutes of gameplay, suggesting that familiarity and muscle memory play a key role in control mastery.

From the testing session (see table 2), *Meclawnism* delivers an engaging and emotionally rewarding experience. The dual-hand gesture system offers novel and immersive control but introduces a learning curve that may affect casual users without appropriate tutorial for onboarding. Future iterations will explore gesture calibration and adaptive difficulty to cover greater accessibility.

|  |  |  |
| --- | --- | --- |
| **TABLE 2.** Summary of player evaluation for GEQ, GFQ, and FTQ. Higher scores indicate more favourable responses. | | |
| **Metric** | **Mean Score (1-5)** | **Interpretation** |
| GEQ - Positive Affect | 4.750 | Highly enjoyable and emotionally engaging |
| GEQ - Flow | 4.333 | Players felt absorbed in the experience |
| GEQ - Challenge | 3.250 | Balanced difficulty level |
| GEQ - Tension | 1.667 | Mild tension, generally manageable |
| GFQ - Fun Factor | 4.542 | Very enjoyable gameplay |
| GFQ - Learning Curve | 3.333 | Moderate to high difficulty in mastering controls |
| FTQ - Gesture Recognition Accuracy | 4.333 | Reliable but improvable |
| FTQ - Control Responsiveness | 4.250 | Generally responsive, occasional delay |

# **CONCLUSION**

*Meclawnism* introduces a novel gesture-controlled game that integrates MediaPipe’s hand gesture recognition into a Unity3D-based claw machine simulation. The system supports dual-hand gesture inputs mapped to unique and intuitive in-game actions. User feedback gathered through Questionnaires such as GEQ, GFQ, and FTQ indicate that there is a high level of engagement and emotional satisfaction, with a steep learning curve due to the unfamiliar input method. The results have shown the viability of hand gesture interfaces for casual gaming, particularly in delivering accessible and immersive experiences despite of a steep learning curve. In the future, we will focus on improving gesture recognition robustness through gesture calibration and refining a better user onboarding experience.

# **REFERENCES**

1. A.A. Zaveri, R. Mashood, N. Faisal, M. Parveen, N. Sami, M. Nazar, and S. Imtiaz, “MobiTest – A Software for Mobile-Based Testing,” Journal of Informatics and Web Engineering  **3**(3), 1–20 (2024). https://doi.org/10.33093/jiwe.2024.3.3.1
2. J. Thau, “Gaming Beyond Entertainment: How Video Games Are Shaping The Future,” Forbes, (n.d.). https://www.forbes.com/councils/forbestechcouncil/2024/05/22/gaming-beyond-entertainment-how-video-games-are-shaping-the-future/
3. M.L. Arnumukti, A.P. Yunus, and A.S. Babale, “Improve Exercise Movement: Detecting Mistakes on Yoga with Mediapipe and MLP,” IJORAS **7**(1), 64–71 (2025). https://doi.org/10.33093/ijoras.2025.7.1.8
4. K.A. Teo, K.S. Sim, C.K. Toa, and K.L. Lew, “Development of Virtual Reality-Based Left Brain System,” Journal of Informatics and Web Engineering   **3**(3), 98–120 (2024). https://doi.org/10.33093/jiwe.2024.3.3.6
5. “Hand Gesture Recognition Sensor- A Comprehensive Guide,” (n.d.).https://www.linkedin.com/pulse/hand-gesture-recognition-sensor-comprehensive-guide-medtechsecure/
6. G.M. Kumar, V. Manohar, B. Ravi, S.V.S. Prasad, S. Paluvatla, and R. Sateesh, “Game Controlling using Hand Gestures,” in *2022 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC)*, (IEEE, Bhubaneswar, India, 2022), pp. 1–5.https://doi.org/10.1109/assic55218.2022.10088337
7. U. Patel, S. Rupani, V. Saini, and X. Tan, “Gesture Recognition Using MediaPipe for Online Realtime Gameplay,” in *2022 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, (IEEE, Niagara Falls, ON, Canada, 2022), pp. 223–229. https://doi.org/10.1109/wi-iat55865.2022.00039
8. C. Rashmika, G. Varsha, and U. Sairam, “Gesture-Based Control of Ping-Pong: Developing an Interactive Game Interface,” in *2023 4th International Conference on Smart Electronics and Communication (ICOSEC)*, (IEEE, Trichy, India, 2023), pp. 1085–1089. https://doi.org/10.1109/icosec58147.2023.10276168
9. O. Saman, and L. Stanciu, “Hand Mobility Recovery Through Game Controlled by Gestures,” in *2019 IEEE 13th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, (IEEE, Timisoara, Romania, 2019), pp. 241–244. https://doi.org/10.1109/saci46893.2019.9111632
10. “MediaPipe Solutions guide | Google AI Edge,” Google AI for Developers, (n.d.). https://ai.google.dev/edge/mediapipe/solutions/guide
11. G. Boesch, “MediaPipe: A Guide Google’s Open Source Framework,” Viso.Ai, (2023). https://viso.ai/computer-vision/mediapipe/
12. T. Jain. *Graph and Calculators in Mediapipe.* Medium. https://medium.com/@jaintarun7/graph-and-calculators-in-mediapipe-673d95a03e07 (2023)
13. N. Kiselov, “kinivi/hand-gesture-recognition-mediapipe,” (2025). <https://github.com/kinivi/hand-gesture-recognition-mediapipe>
14. “Guide to Python Socket Programming,” Built In, (n.d.). https://builtin.com/data-science/python-socket
15. karelz, “UdpClient Class (System.Net.Sockets),” (n.d.). https://learn.microsoft.com/en-us/dotnet/api/system.net.sockets.udpclient?view=net-8.0