รายงานโปรเจควิชา DE461: Introduction to Computer Vision

เรื่อง Neon Snake Game: Hand-Tracking Edition

เสนอ

อาจารย์บรรพตรี คมขำ

จัดทำโดย

นายปัณณธร สุวรรณาศรัย รหัสนิสิต 66102010175

นายพลวัต พงศ์ทิพย์พนัส รหัสนิสิต 66102010249

นายพลวิชญ์ วนิดา รหัสนิสิต 66102010584

รายงานนี้เป็นส่วนหนึ่งของรายวิชา DE461 Introduction to Computer Vision

ภาคเรียนที่ 1 ปีการศึกษาพุทธศักราช 2568

มหาวิทยาลัยศรีนครินทรวิโรฒ

บทคัดย่อ

โครงงานนี้นำเสนอการพัฒนาเกม “Neon Snake Game” บนเว็บแอปพลิเคชัน โดยมีเป้าหมายในการสร้างเกมงูที่สามารถควบคุมทิศทางการเคลื่อนที่ได้ด้วยนิ้วมือผ่านเทคโนโลยีตรวจจับมือแบบเรียลไทม์ (MediaPipe Hands) เพื่อเพิ่มความแม่นยำและความสะดวกในการใช้งาน ทั้งบนคอมพิวเตอร์และอุปกรณ์พกพา โดยเกมนี้พัฒนาต่อยอดมาจาก Snake Face ของ Paul Rubenstein ที่ใช้การควบคุมด้วยการหันศีรษะผ่านระบบตรวจจับใบหน้า แต่ปรับเปลี่ยนให้ใช้การชี้นิ้วเพื่อให้เหมาะสมกับผู้เล่นมากขึ้น

นอกจากนี้ ระบบยังพัฒนาให้มีการปรับระดับความยาก ฟีเจอร์อุปสรรครูปแบบใหม่ (electric orb), achievement และอินเทอร์เฟซในสไตล์ neon พร้อมบันทึกผลสถิติผู้เล่นในแต่ละรอบการเล่น รวมถึงฟีเจอร์เสริม “Face Yoga Challenge” ซึ่งให้ผู้เล่นทำท่าบริหารใบหน้าและตรวจสอบความถูกต้องผ่านการจับ landmark ใบหน้าด้วย face-api.js ระหว่างเปลี่ยนด่าน

ซึ่งจากผลการทดลองใช้งานจริงพบว่า ระบบสามารถตรวจจับมือและควบคุมเกมงูได้อย่างลื่นไหล มีความแม่นยำสูง โดยมีฟีเจอร์เสริมอย่างการตรวจจับใบหน้าช่วยเพิ่มความสนุกสนานและการมีส่วนร่วม รวมถึงความแปลกใหม่ให้กับตัวเกมแม้จะไม่ได้เป็นฟีเจอร์หลักของตัวเกม โดยรวมแล้วโครงงานนี้สะท้อนถึงศักยภาพของการนำเทคโนโลยี Computer Vision มาประยุกต์ใช้ในการพัฒนาเกมอินเทอร์แอคทีฟที่ตอบโจทย์ประสบการณ์ผู้ใช้ร่วมสมัยได้อย่างมีประสิทธิภาพ

คำสำคัญ: Computer Vision, Hand Tracking, Gesture Control

คำนำและกิตติกรรมประกาศ

รายงานฉบับนี้จัดทำขึ้นในหัวข้อ **“Neon Snake Game: ระบบเกมงูควบคุมด้วยท่าทางนิ้วมือบนเว็บแอปพลิเคชัน”** ซึ่งเป็นส่วนหนึ่งของรายวิชา **DE461 Introduction to Computer Vision** ภาคเรียนที่ 1 ปีการศึกษา 2568 โดยมีวัตถุประสงค์เพื่อศึกษาการนำเทคโนโลยี Computer Vision มาประยุกต์ใช้ในการสร้างเกมอินเทอร์แอคทีฟ โดยเน้นการตรวจจับมือแบบเรียลไทม์ผ่าน MediaPipe Hands เพื่อเพิ่มความแม่นยำและความสะดวกในการควบคุม รวมทั้งต่อยอดจากโครงงานต้นแบบ Snake Face ที่ใช้การตรวจจับศีรษะในการเล่น เปลี่ยนไปเป็นการใช้ gesture ของนิ้วมือ และเสริมฟีเจอร์ใหม่ให้สามารถเพิ่มความท้าทายและประสบการณ์การเล่นสำหรับผู้ใช้ทุกกลุ่ม

ผู้จัดทำเล็งเห็นว่าการพัฒนาเกมด้วยเทคนิคภาพและการประมวลผลท่าทาง เป็นโอกาสสำคัญในการเรียนรู้และทดลองเทคโนโลยี Computer Vision ซึ่งจะเป็นประโยชน์ทั้งในเชิงวิชาการและการพัฒนาโซลูชันจริงในอนาคต รายงานฉบับนี้จึงมุ่งเน้นให้ผู้อ่านเข้าใจหลักการออกแบบ วิธีการประยุกต์ใช้เครื่องมือ และขั้นตอนการพัฒนา รวมถึงการนำเทคนิคการตรวจจับใบหน้ามาเสริมเพื่อเพิ่มมิติของความสนุกและการมีส่วนร่วมในเกม

คณะผู้จัดทำขอกราบขอบพระคุณ **อาจารย์บรรพตรี คมขำ** ผู้ให้คำแนะนำ ข้อเสนอแนะ และตรวจสอบความถูกต้องในการดำเนินโครงการ ตลอดจนการจัดทำรายงานฉบับนี้ด้วยความเอื้อเฟื้อ รวมถึงบรรยากาศที่สนับสนุนการเรียนรู้ และการส่งเสริมให้นักศึกษาสร้างสรรค์ผลงานอย่างเต็มศักยภาพ นอกจากนี้ขอขอบคุณเพื่อนๆ กลุ่ม “CV_sigma” ทุกคน ที่ร่วมแรงร่วมใจกันวางแผน ออกแบบ และแก้ไขปัญหา จนโครงงานนี้สำเร็จลุล่วงตามเป้าหมายและผู้ให้ feedback รวมถึงอาจารย์ที่มีส่วนช่วยให้การพัฒนางานมีความสมบูรณ์ยิ่งขึ้น

ทั้งนี้หากรายงานฉบับนี้ยังมีข้อบกพร่องใด ๆ ทางคณะผู้จัดทำขอน้อมรับข้อเสนอแนะจากอาจารย์และผู้อ่านทุกท่าน เพื่อนำไปปรับปรุงและพัฒนาผลงานให้ดียิ่งขึ้นในอนาคต

คณะผู้จัดทำ
(CV_sigma)

สารบัญ

สารบัญ (ต่อ)

บทที่ 1
บทนำ

ความเป็นมาและปัญหาของโครงการ

ในปัจจุบัน เทคโนโลยี Computer Vision ได้เข้ามามีบทบาทในการสร้างประสบการณ์แบบอินเทอร์แอคทีฟที่หลากหลายมากขึ้นในวงการเกม โดยเฉพาะในเกมที่เน้นการควบคุมผ่าน gesture หรือท่าทางของผู้เล่นจริงๆ ซึ่งช่วยให้การมีส่วนร่วมและความสนุกมีความแปลกใหม่อย่างเห็นได้ชัด อย่างไรก็ตาม เกมคลาสสิคประเภท “Snake” ที่ผ่านการสร้างสรรค์ในรูปแบบต่าง ๆ ยังคงใช้วิธีควบคุมแบบเดิม เช่น คีย์บอร์ด เมาส์ หรือทัชสกรีน ซึ่งอาจไม่ตอบโจทย์กลุ่มผู้เล่นที่ต้องการความหลากหลายทางอินเทอร์แอคทีฟ โดยเฉพาะผู้ใช้ในยุคที่ใช้มือถือและ webcam เป็นหลัก

ซึ่งโปรเจค Neon Snake Game นี้ได้รับแรงบันดาลใจมาจากงานต้นแบบ "Snake Face" ของ Paul Rubenstein ที่ใช้เทคนิค head-pose detection ในการควบคุมทิศทางงู ผ่านการขยับศีรษะและมุมมองจากกล้อง webcam ซึ่งถึงแม้ว่าจะสร้างประสบการณ์ใหม่ แต่ในทางปฏิบัติยังพบข้อจำกัดในด้านความแม่นยำ (เนื่องจากแสง, มุมกล้อง, ท่าทางศีรษะ) และความสะดวกในการขยับของศรีษะของผู้เล่น โดยเฉพาะเมื่อเล่นผ่านอุปกรณ์พกพา

จากปัญหาเหล่านี้ กลุ่มผู้พัฒนาจึงเห็นความสำคัญในการพัฒนาการควบคุมใหม่ที่ต่อยอดจาก head-pose detection สู่การควบคุมงูผ่าน “hand tracking” ด้วยเทคโนโลยี MediaPipe Hands ซึ่งออกแบบมาเพื่อให้สามารถจับท่าทางมือและนิ้วแบบเรียลไทม์ จากกล้องมือถือหรือ webcam ส่งผลให้การควบคุมมีความแม่นยำ ตอบสนองได้ดี ใช้งานง่ายสำหรับทุกอุปกรณ์ และเหมาะกับผู้เล่นทั่วไป รวมถึงผู้ที่มีข้อจำกัดทางกายภาพ

นอกจากนี้ โปรเจคยังพัฒนาและเพิ่มฟีเจอร์ที่ช่วยเสริมความสนุกและความท้าทายของเกม อาทิ ระบบ electric orb (อุปสรรคในแต่ละด่าน), achievement ระบบคะแนนและสถิติต่าง ๆ พร้อมเพิ่มกิจกรรมเสริม เช่น Face Yoga Challenge ที่ตรวจจับ landmark ใบหน้าเพื่อสร้างปฏิสัมพันธ์และกระตุ้นการมีส่วนร่วมของผู้เล่น ซึ่งทั้งหมดนี้สะท้อนถึงการประยุกต์ใช้ Computer Vision ในการพัฒนาระบบเกมอินเทอร์แอคทีฟที่จัดการกับข้อจำกัดเดิม และปูทางไปสู่ประสบการณ์เล่นเกมในยุคใหม่อีกด้วย

ดังนั้นกลุ่มของเราจึงเลือกจัดทำโครงการ **“Neon Snake Game: ระบบเกมงูควบคุมด้วยท่าทางนิ้วมือบนเว็บแอปพลิเคชัน”** โดยมุ่งเน้นการสร้างและทดสอบระบบเกมงูที่ควบคุมด้วย hand tracking ผ่านเทคโนโลยี MediaPipe Hands พร้อมออกแบบขั้นตอนการติดตามผลเพื่อประเมินความแม่นยำในการควบคุม ความสะดวกของการเล่น ฟีเจอร์ที่เสริมเข้ามา เช่น electric orb, achievement และ Face Yoga Challenge รวมทั้งบันทึกข้อมูลผลการเล่น คะแนน ระบบสถิติ และการตอบรับจากผู้เล่นในแต่ละรอบ เพื่อใช้วิเคราะห์ประสิทธิภาพของระบบและประสบการณ์ที่ได้รับของผู้เล่นที่มีประสิทธิภาพ

วัตถุประสงค์ของโครงการ

1.เพื่อศึกษาออกแบบพัฒนาเกมงู (Neon Snake Game) ที่สามารถควบคุมทิศทางการเคลื่อนที่ด้วยท่าทางนิ้วมือ ผ่านเทคโนโลยีตรวจจับมือแบบเรียลไทม์ (MediaPipe Hands)

2.เพื่อประยุกต์ใช้เทคนิค Computer Vision ในระบบเกม เพื่อเพิ่มประสบการณ์ การมีส่วนร่วม และความสะดวกสำหรับผู้ใช้งานผ่านการควบคุมด้วย gesture

3.เพื่อศึกษาและเปรียบเทียบประสิทธิภาพของการควบคุมเกมงูรูปแบบใหม่ กับแบบดั้งเดิม เช่น การกด keyboard หรือ การ touch

4.เพื่อเพิ่มฟีเจอร์เสริม เช่นระบบอุปสรรค electric orb, achievement, และกิจกรรม Face Yoga Challenge โดยตรวจจับท่าทางใบหน้าด้วย face-api.js เพื่อประเมินการมีส่วนร่วมและความสนุกของตัวเกมที่เพิ่มมากขึ้น

ความสำคัญของการทำโครงการ

การพัฒนาเกม “Neon Snake Game” มีความสำคัญทั้งในด้านวิชาการและการประยุกต์ใช้เทคโนโลยี Computer Vision ในชีวิตจริง ด้วยเหตุผลดังต่อไปนี้

- **มุ่งส่งเสริมความรู้และทักษะด้าน Computer Vision** ผ่านการสร้างผลงานที่นำเทคนิค hand tracking และ face detection มาใช้งานจริงในระบบเกมอินเทอร์แอคทีฟ ช่วยให้ผู้เรียนและนักพัฒนามีโอกาสทดลองประยุกต์ใช้โมเดลปัญญาประดิษฐ์ในบริบทการใช้งานที่จับต้องได้
- **ขยายขีดความสามารถและประสบการณ์ของผู้ใช้เกม** โดยเปลี่ยนรูปแบบการควบคุมจาก keyboard/mouse เดิม ให้สามารถสั่งงานและโต้ตอบกับเกมผ่าน gesture ที่เป็นธรรมชาติ รวมถึงลดข้อจำกัดสำหรับผู้ใช้งานบนอุปกรณ์พกพา หรือกลุ่มที่มีปัญหาด้านกายภาพ

- **สร้างนวัตกรรมใหม่ด้าน UI/UX และฟีเจอร์เกม** ด้วยการออกแบบธีม neon และระบบ achievement ที่กระตุ้นให้ผู้เล่นเกิดแรงบันดาลใจและเกิด engagement สูงขึ้น นอกจากนี้ ฟีเจอร์เสริม Face Yoga Challenge ช่วยสร้างความสนุกสนาน ความท้าทาย และพัฒนากิจกรรมร่วมในกลุ่มผู้เล่น
- **ใช้เป็นต้นแบบสำหรับการพัฒนาโปรเจคในอนาคต** ทั้งในสาย Computer Vision, Human-Computer Interaction และการออกแบบเกมแนวใหม่ที่เน้นความหลากหลายในการมีส่วนร่วม
- **เพิ่มโอกาสให้การเรียนรู้และการทดลองทางเทคนิค** สามารถนำต่อยอดในระดับอุตสาหกรรม เช่น เกมเพื่อการศึกษา เพื่อความบันเทิง หรือเพื่อคนพิการ (accessible gaming)

ขอบเขตของการศึกษาโครงการ

เพื่อให้การดำเนินโครงการ “Neon Snake Game” มีความชัดเจนและเป็นระบบ กลุ่มผู้จัดทำได้กำหนดขอบเขตของโครงการไว้ดังนี้

ขอบเขตด้านเวลา (Duration)

- โครงการ Neon Snake Game เริ่มดำเนินการพัฒนาและเก็บข้อมูลทดลองตั้งแต่วันที่ 26 สิงหาคม พ.ศ. 2568 เป็นต้นมา โดยสิ้นสุดการทดสอบจริงและสรุปผลภายในภาคเรียนที่ 1 ปีการศึกษา 2568

ขอบเขตด้านสถานที่ (Location)

- พัฒนาและทดลองบนอุปกรณ์ส่วนตัว/ห้องเรียน/ห้องปฏิบัติการที่มีอินเทอร์เน็ตและกล้อง

ขอบเขตด้านเนื้อหา (Content)

- โครงการเน้นการออกแบบและพัฒนาเกมงู (Snake Game) ที่ควบคุมด้วยมือแบบเรียลไทม์ (MediaPipe Hands) รวมฟีเจอร์ electric orb, achievement, Face Yoga Challenge (face-api.js) และระบบสถิติของผู้เล่น

ตัวแปรที่ศึกษา (Variables)

- **ตัวแปรต้น (Independent Variable):** รูปแบบการควบคุมเกม มือ: hand tracking
- **ตัวแปรตาม (Dependent Variables):** คะแนน, ความแม่นยำทิศทาง, เวลาเล่น, ความสำเร็จ/feedback ในแต่ละฟีเจอร์
- **ตัวแปรควบคุม (Control Variables):** สภาพแวดล้อมของการทดสอบ (เช่น ใช้ในห้องที่มีแสงกลางวันหรือไฟฟ้า ไม่มีแสงย้อน, ระยะกล้อง), รุ่นของ browser และอุปกรณ์ที่ใช้รันโปรแกรม, ขนาดหน้าต่างเกม และความละเอียดของกล้อง web/มือถือ

สมมุติฐานของโครงการ

- การควบคุมเกมงูด้วยเทคโนโลยีตรวจจับมือแบบเรียลไทม์ (MediaPipe Hands) จะทำให้ผู้เล่นสามารถสั่งงานและควบคุมทิศทางของงูได้อย่างแม่นยำและลื่นไหลมากกว่าการใช้วิธีแบบเดิม
- การเพิ่มฟีเจอร์ Face Yoga Challenge ที่ตรวจจับใบหน้าผ่าน face-api.js จะช่วยเพิ่มความสนุก ความท้าทาย และการมีส่วนร่วมของผู้เล่น
- การออกแบบ UI/UX ในสไตล์ neon และการใช้ระบบ achievement ภายในเกมจะช่วยสร้างแรงจูงใจและประสบการณ์ที่ดีกับผู้ใช้
- ระบบที่สร้างขึ้นสามารถนำไปประยุกต์และต่อยอดกับแอปพลิเคชันหรือเกมอื่น ๆ ที่ต้องการควบคุมผ่าน gesture หรือเทคโนโลยี Computer Vision ได้

นิยามศัพท์เฉพาะ

- **Computer Vision:** หมายถึง เทคโนโลยีการประมวลผลภาพด้วยวิธีการทางคอมพิวเตอร์ เพื่อให้ระบบสามารถตรวจจับ วิเคราะห์ และเข้าใจข้อมูลจากภาพ/วิดีโอ เช่น ท่าทางมือ ใบหน้า ฯลฯ
- **Hand Tracking:** หมายถึง การตรวจจับและติดตามตำแหน่งท่าทางของมือมนุษย์ด้วยเทคนิค Computer Vision และปัญญาประดิษฐ์แบบเรียลไทม์
- **MediaPipe Hands:** หมายถึง โมเดล deep learning จาก Google สำหรับตรวจจับ landmark จุดสำคัญ 21 จุดบนมือมนุษย์จาก input ภาพกล้อง (webcam/mobile) เพื่อใช้งานได้แบบ real-time บนเว็บหรือแอป
- **Snake Game:** หมายถึง เกมคลาสสิคที่ให้ผู้เล่นควบคุมงูเพื่อรับคะแนน หลีกเลี่ยงอุปสรรคต่าง ๆ และสะสมความยาวงูให้ได้มากที่สุด
- **Gesture Control:** หมายถึง การควบคุมทิศทางหรือการดำเนินการในระบบ/โปรแกรมหรือเกม ด้วยท่าทาง (gesture) ของมือแทนการใช้ปุ่มกดหรือเมาส์
- **face-api.js:** หมายถึง ไลบรารี JavaScript สำหรับตรวจจับใบหน้า, landmark/จุดสำคัญบนใบหน้า, การวิเคราะห์ expression เพื่อประยุกต์ใช้ในงาน interactive หรือ face challenge
- **Face Yoga Challenge:** หมายถึง กิจกรรมในเกมที่ให้ผู้เล่นจำลองท่าบริหารใบหน้าตามโจทย์ โดยระบบจะตรวจจับและประเมินผลการทำท่าโยคะผ่าน camera และใบหน้า
- **Electric Orb (Obstacle):** หมายถึง อุปสรรคที่ปรากฏในเกมงู มีลักษณะพิเศษ หากผู้เล่นสัมผัสจะเกิดผลวิเศษ เช่น ลดคะแนนหรือความยาวของงู

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

ในการดำเนินโครงการ “Neon Snake Game: ระบบเกมงูควบคุมด้วยมือบนเว็บแอปพลิเคชัน” คณะผู้จัดทำได้ศึกษางานต้นแบบ เทคโนโลยี และงานวิจัยที่เกี่ยวข้อง เพื่อใช้เป็นฐานความรู้ในออกแบบ ฟีเจอร์ระบบ และวิเคราะห์ผลการทดลอง โดยสรุปสาระสำคัญได้ดังนี้

แนวคิดและทฤษฎีที่เกี่ยวข้อง

1.แนวคิด Human-Computer Interaction (HCI) และ Gesture Control

- **HCI** เป็นศาสตร์ที่ศึกษาเรื่องการสร้างปฏิสัมพันธ์ระหว่างมนุษย์กับคอมพิวเตอร์ การประยุกต์ gesture หรือท่าทางในการควบคุมเกมช่วยเพิ่ม accessibility และ interactivity ให้กับ user (Shneiderman, 2016) ได้
- **Gesture recognition** การทำให้ตัวเกมสามารถรับคำสั่ง/ความตั้งใจจากผู้เล่นผ่านการขยับมือหรือท่าทางต่าง ๆ แทนเมาส์หรือคีย์บอร์ด

2.หลักการ Computer Vision ในงาน real-time tracking

- **Computer Vision** คือการให้คอมพิวเตอร์ "เข้าใจ" ข้อมูลจากภาพหรือวิดีโอ เช่น แยกวัตถุ แยกมนุษย์ ตรวจจับ pose ฯลฯ (Szeliski, 2022)
- **การนำ real-time hand/face landmark detection** อาทิเช่น MediaPipe Hands, face-api.js มาประยุกต์ใช้ในระบบอินเทอร์แอคทีฟ

3. MediaPipe Hands และ Face API

- **MediaPipe Hands** เป็น framework ของ Google สำหรับตรวจจับ landmark มือ 21 จุด บน browser/app แบบ real-time ด้วย deep learning
- **Face-api.js** คือไลบรารีใน JavaScript สำหรับตรวจจับ landmark ใบหน้าและแอปลิเคชันเชิง interactive

เอกสารทางเทคนิคและงานวิจัยที่เกี่ยวข้อง

1.งานต้นแบบเกมงูควบคุมด้วย Gesture

- Snake Face โดย Paul Rubenstein (2020) ใช้ head pose detection ควบคุมทิศทางเกมงูบนเว็บ
- งาน gesture control อื่นๆ ที่ใช้ MediaPipe ร่วมกับ Unity/WebGL สร้างเกม หรือสั่งงาน UI

2.วิธีวัดผลความแม่นยำและประสบการณ์ผู้ใช้ในเกม

- งานวิจัยเกี่ยวกับ hand tracking (Google, 2023) เสนอ metrics เรื่อง latency, accuracy, และ usability ในการควบคุม real-world application

3.งานเปรียบเทียบ UI/UX เกมแนว Interactive

- งานศึกษาความพึงพอใจ/การมีส่วนร่วมของผู้ใช้เกมที่ควบคุมด้วยมือ เทียบกับเกมที่ใช้ปุ่มหรือเมาส์แบบเดิม (User studies, 2021; ACM)

Paul Rubenstein. (2020). Snake Face. Retrieved from https://github.com/paruby/snake-face

Google AI. MediaPipe Hands. Retrieved from https://ai.google.dev/edge/solutions/vision/hand_landmarker

face-api.js. Retrieved from https://github.com/justadudewhohacks/face-api.js

บทที่ 3
วิธีดำเนินการโครงการ / การออกแบบและพัฒนาระบบ

1. โครงสร้างการพัฒนาและกระบวนการทำงาน

โครงการ Neon Snake Game ใช้กระบวนการพัฒนาแบบ SDLC (Software Development Life Cycle) ที่แบ่งขั้นตอนสำคัญ ได้แก่ การวิเคราะห์ความต้องการ ออกแบบ พัฒนา ทดสอบ และประเมินผล โดยโครงสร้างระบบแบ่งออกเป็น 3 ส่วนหลักดังนี้:

1.1 ส่วนตรวจจับท่าทางมือ/ใบหน้า

1.1.1 ใช้ MediaPipe Hands สำหรับตรวจจับ 21 จุด landmark บนมือ เพื่ออ่านค่าตำแหน่งนิ้วที่ใช้ควบคุมทิศทางงู

1.1.2 ใช้ face-api.js สำหรับตรวจจับ landmark ใบหน้า ในกิจกรรม Face Yoga Challenge

1.2 ส่วนกลไกเกมและฟีเจอร์

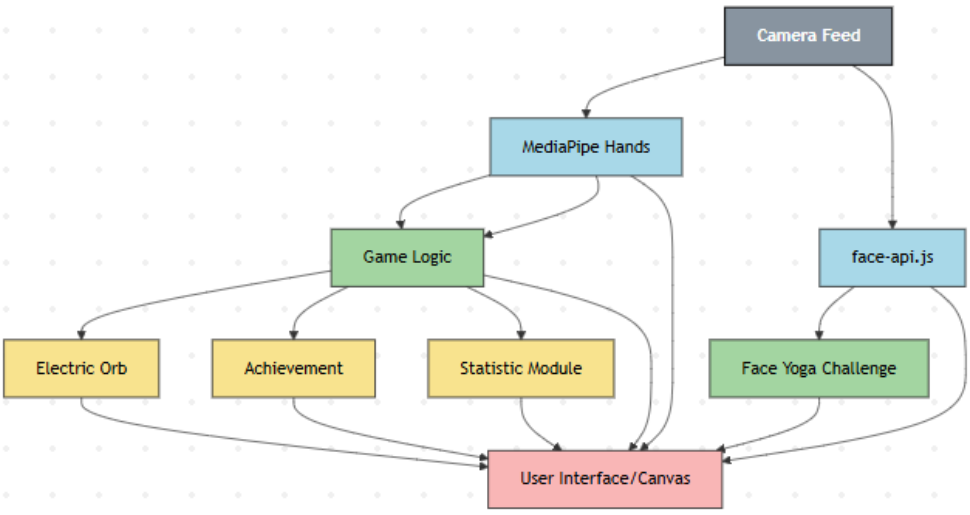
1.2.1 พัฒนา game loop สำหรับควบคุมงู, อาหาร, electric orb, achievement, และ Face Yoga Challenge

1.2.2 เพิ่มการตอบสนองและความท้าทาย เช่น electric orb (อุปสรรค) achievement, และระบบภารกิจ level-up

1.3 ส่วนอินเทอร์เฟซผู้ใช้และระบบบันทึกสถิติ

1.3.1 ออกแบบ UI/UX สไตล์ neon, แสดงคะแนน ความยาวงู achievement ผลการเล่นแบบเรียลไทม์

1.3.2 ระบบบันทึกภาพ/สถิติของผู้เล่นสำหรับการวิเคราะห์ผลลัพธ์และพัฒนาในอนาคต



2. เครื่องมือและเทคโนโลยีที่ใช้

2.1 ภาษาและเครื่องมือพัฒนา:

2.1.1 HTML5 และ CSS3 สำหรับโครงสร้างและการตกแต่ง UI

2.1.2 JavaScript เป็นภาษาหลักสำหรับ game logic และการเชื่อมต่อกับ AI model

2.1.3 WebGL/Canvas สำหรับ rendering ภาพเกมและเอฟเฟคแบบ real-time

2.2 เทคโนโลยี Core:

2.2.1 MediaPipe Hands API (จาก Google): สำหรับตรวจจับและวิเคราะห์ท่าทางมือแบบ real-time พร้อม landmark 21 จุด

2.2.2 face-api.js: ตรวจจับ landmark และ expression ใบหน้า รองรับกิจกรรม Face Yoga Challenge

2.3 Framework และ Library เพิ่มเติม:

2.3.1 TailwindCSS/Orbitron/Exo2 (ตกแต่ง UI)

2.3.2 Web Audio API (เสียง/achievement)

2.4 ฮาร์ดแวร์และแพลตฟอร์ม:

2.4.1 Laptop/PC ที่มี webcam หรือ smartphone ที่มี browser รองรับ MediaPipe

2.4.2 บราวเซอร์ Chrome, Firefox, Edge (version ล่าสุด)

2.5 โครงสร้างไฟล์สำคัญ:

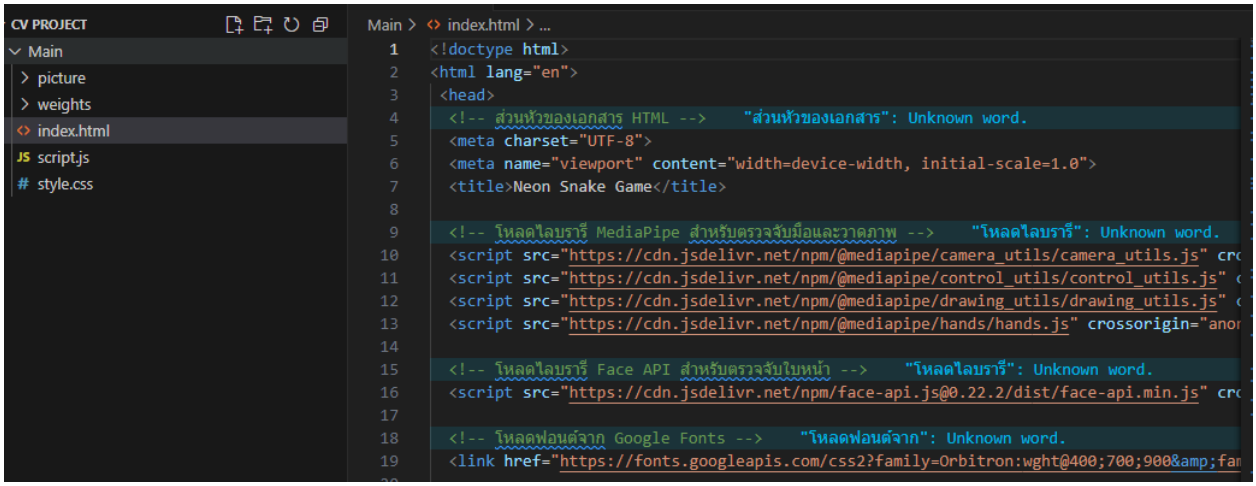
2.5.1 index.html โครงสร้างเว็บเกม

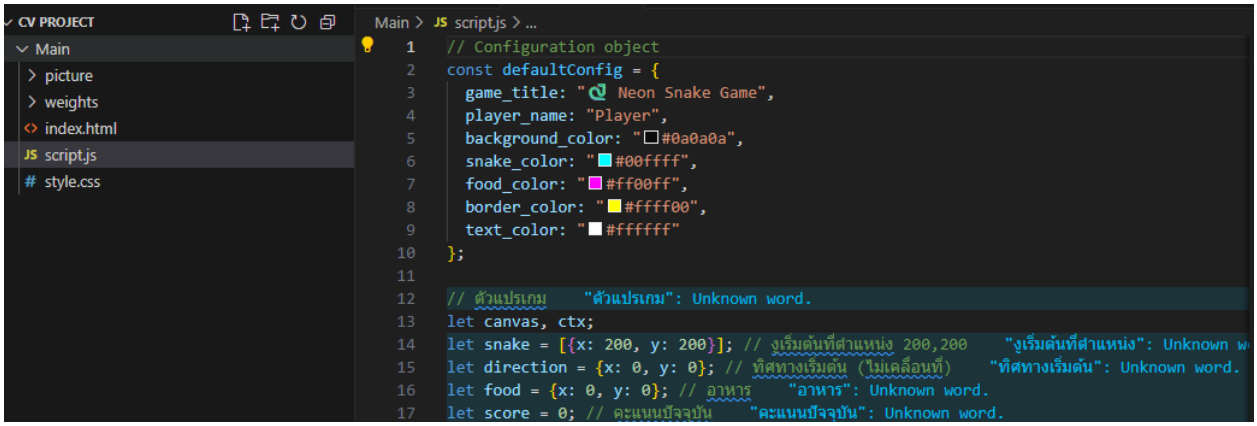
2.5.2 style.css สร้างตกแต่งสไตล์ neon theme

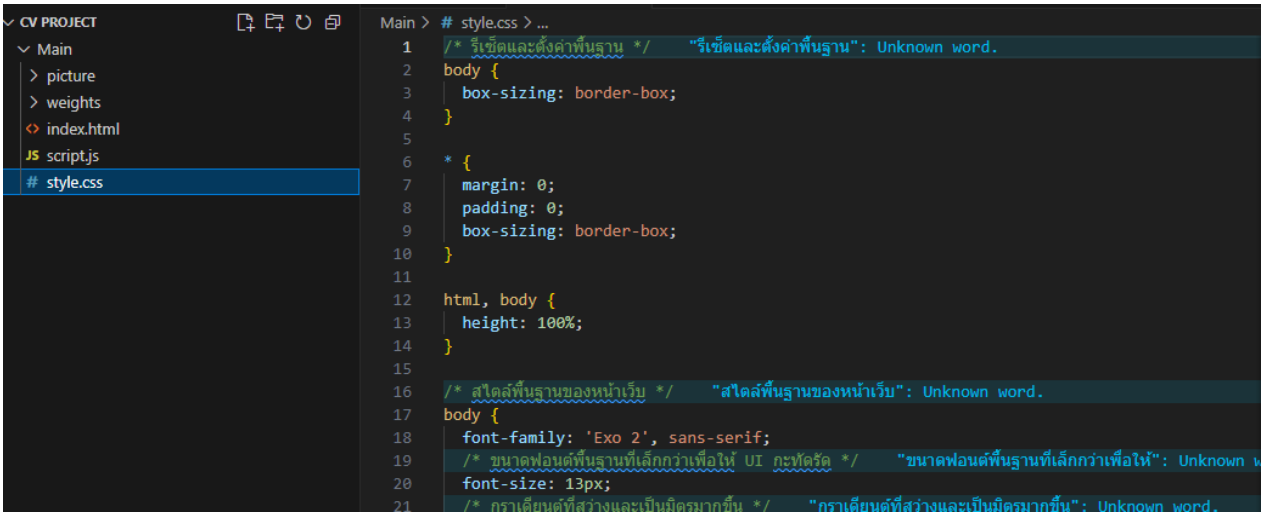
2.5.3 script.js logic เกม, การเชื่อมต่อกับ MediaPipe/face-api.js

2.5.4 /weights/ ไฟล์โมเดลสำหรับ hand tracking & face landmark

2.5.5 /picture/ ภาพประกอบหรือภาพวิเคราะห์ผลลัพธ์







3. ขั้นตอนการพัฒนาและกลไกการทำงานหลัก

3.1 Hand Tracking Mapping: ในขั้นตอนนี้ใช้ MediaPipe Hands ตรวจจับตำแหน่ง landmark 21 จุด โดยเฉพาะจุดปลายนิ้วชี้ (index finger) ซึ่งนำค่าพิกัด (x, y) ของ landmark มาสร้างเงื่อนไขการ mapping กับทิศทางการเคลื่อนไหวของงูบนจอ เช่น หากตำแหน่งนิ้วชี้อยู่เหนือ threshold แกนแนวตั้ง ให้กำหนดทิศทางงู "ขึ้น" และหากอยู่ซ้ายขวาเกิน threshold ให้ขยับ "ซ้าย/ขวา" ตามการ mirror ของกล้อง

3.2 Game Logic & Feature Implementation: ระบบกลไกเกมเริ่มจาก game loop ที่คอย update สถานะงู (ตำแหน่ง หัวงู ความยาว), ตรวจ collision กับขอบจอและ food, implement ระบบ wall wrap รอบขอบ, spawn และตรวจจับ electric orb โดยทุก tick จะมีการวาด UI ใหม่และปรับความเร็วความยากตามคะแนนและระดับของเกม

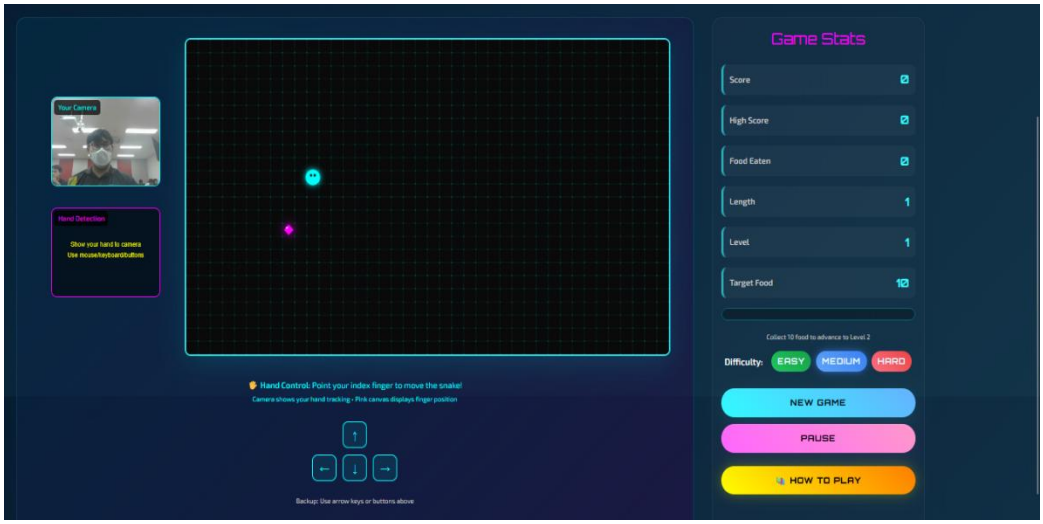
3.3 Face Yoga Challenge: หลังผู้เล่นผ่านแต่ละ level จะมี “Face Yoga Challenge” ใช้ face-api.js ตรวจจับ landmark ใบหน้า เปรียบเทียบกับ template pose เป้าหมาย เมื่อผู้

เล่นทำลักษณะของใบหน้าถูกต้องระบบจะให้ผ่านด่าน โดยระบบจะจับ similarity ของจุด key landmark หลักบนใบหน้าผู้เล่นเทียบกับตัวอย่างของตัวระบบ

3.4 UI/UX และ Achievement: ออกแบบหน้าตาเกม สไตล์ neon สีสด ใช้ฟอนต์ Orbitron แสดงคะแนน ความยาว กล่อง achievement แบบ interactive พร้อม animation พิเศษ (เช่น เอฟเฟคไฟนีออน/เสียงเมื่อ unlock achievement)

3.5 การทดสอบและประเมินผล: ทดสอบความแม่นยำของ hand tracking ในหลาย lighting/ทิศทาง ความลื่นไหลของเกม ประสบการณ์การใช้งาน และบันทึกสถิติผลการเล่นทั้งคะแนนและจำนวนรอบการผ่าน requirement ของแต่ละด่าน

3.6 ส่วนการตรวจจับใบหน้า (Face Detection) ระบบเลือกใช้ไลบรารี face-api.js ซึ่งทำงานร่วมกับโมเดล SSD MobileNet V1 และโมเดล 68 Face Landmark Points โดยระบบจะทำการระบุจุดพิกัดสำคัญบนใบหน้าจำนวน 68 จุด (ประกอบด้วย กรอบหน้า 17 จุด, คิ้ว 10 จุด, จมูก 9 จุด, ดวงตา 12 จุด, และริมฝีปาก 20 จุด) กลไกการตรวจสอบความถูกต้องในด่าน Face Yoga Challenge ทำงานโดยการคำนวณค่า Euclidean Distance ระหว่างพิกัดของจุด Landmark บนใบหน้าผู้เล่น เปรียบเทียบกับพิกัดของภาพต้นแบบ (Target Pose) เพื่อหาค่าความเหมือน (Similarity Score) หากค่าความผิดพลาดต่ำกว่าเกณฑ์ที่กำหนด (Threshold) ระบบจึงจะอนุญาตให้ผู้เล่นผ่านด่านได้ ซึ่งช่วยให้การตรวจจับมีความละเอียดครอบคลุมทั้งรูปทรงหน้าและการขยับอวัยวะต่างๆ บนใบหน้า"



How to Play

Game Objective

Control the neon snake to eat food and grow longer. Avoid hitting your own body! The snake wraps around walls, so you won't die from hitting edges.

Hand Controls

Point Up: Snake moves up

Point Down: Snake moves down

Point Right: Snake moves left

Point Left: Snake moves right

Note: Camera is mirrored like a selfie camera

Keyboard Controls

↑	↓	←	→
Up	Down	Left	Right

Scoring & Levels

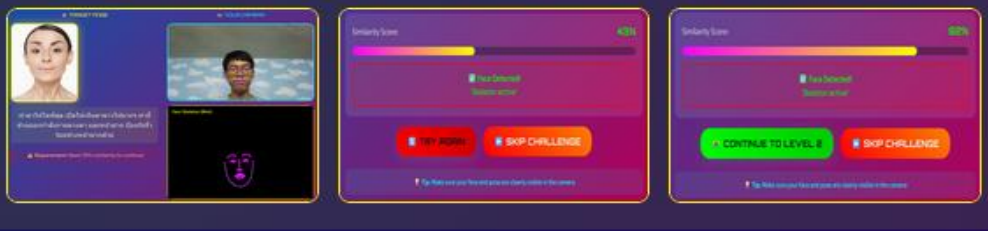
- **+10 points** for each diamond food eaten
- **Level up** every 10 food collected
- **FACE YOGA CHALLENGE:** After leveling up, you must copy the example pose to continue!
- **Speed increases** with each level
- **Achievements** unlock at milestone scores

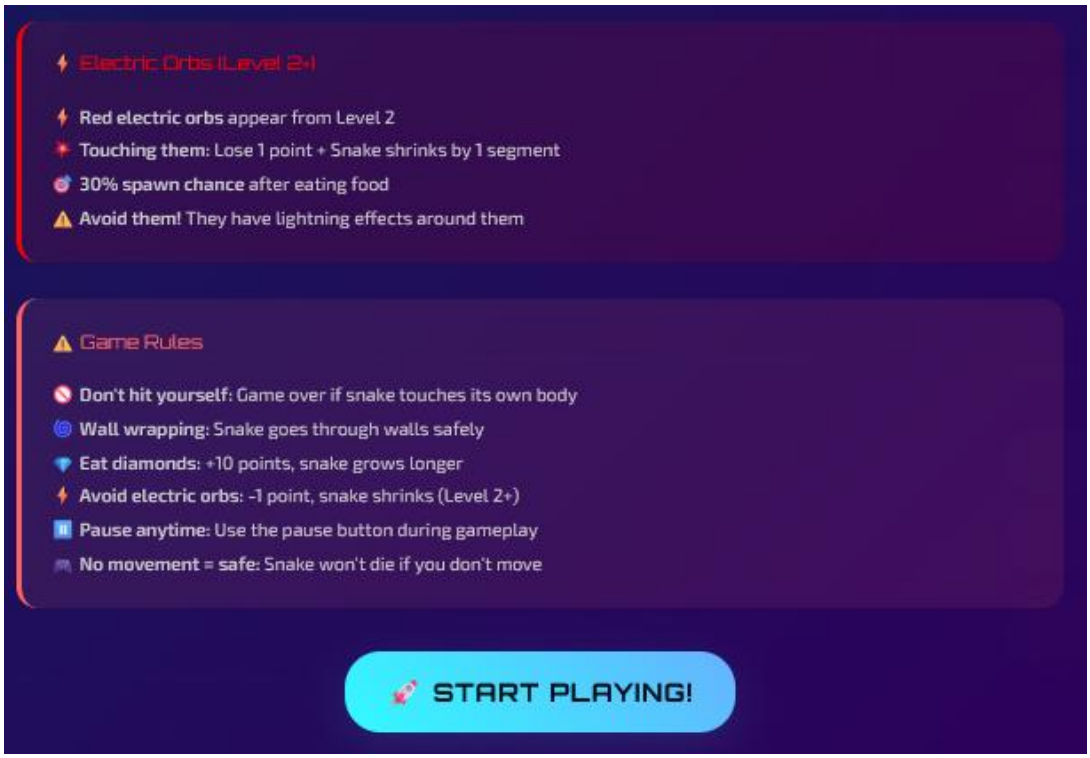
Level Up Challenge (Face Yoga)

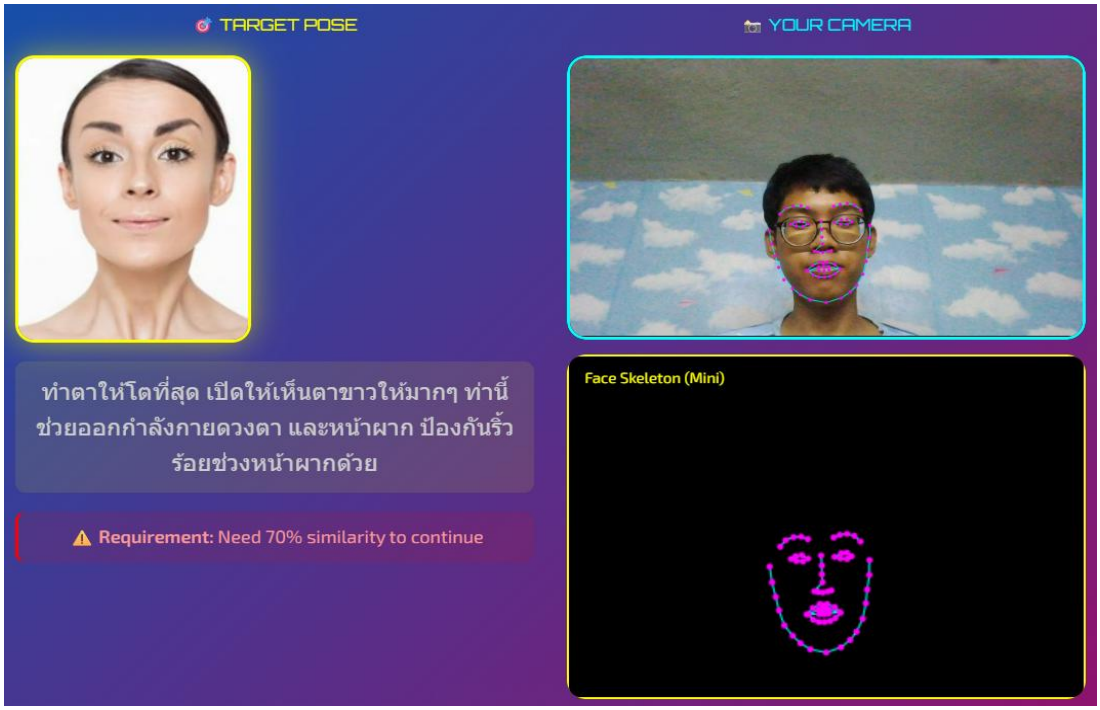
To advance to the next level, you must complete a **Face Yoga Challenge!**

This is a fun break to stretch your facial muscles before the game gets faster.

- When you level up, the game will **pause**.
- A new window will show an example pose (like "Puff Cheeks" or "Wide Eyes").
- Your camera will show a **face skeleton** to help guide you.
- Your mission: **Copy the example pose** until the AI detects a match.
- Once you succeed, the game resumes at the **new, faster speed!**
- To pass, your Similarity Score must beat the Required Percentage. Match the pose to continue!




Electric Orbs (Level 2+)
Red electric orbs appear from Level 2
Touching them: Lose 1 point + Snake shrinks by 1 segment
30% spawn chance after eating food
Avoid them! They have lightning effects around them
Game Rules
Don't hit yourself: Game over if snake touches its own body
Wall wrapping: Snake goes through walls safely
Eat diamonds: +10 points, snake grows longer
Avoid electric orbs: -1 point, snake shrinks (Level 2+)
Pause anytime: Use the pause button during gameplay
No movement = safe: Snake won't die if you don't move
START PLAYING!


TARGET POSE
YOUR CAMERA
ทำตาให้โตที่สุด เปิดให้เห็นตาขาวให้มากๆ ท่านี้ช่วยออกกำลังกายดวงตา และหน้าผาก ป้องกันริ้วร้อยช่วงหน้าผากด้วย
Requirement: Need 70% similarity to continue
Face Skeleton (Mini)

4. สรุปผลและภาพรวม

จากขั้นตอนการออกแบบและพัฒนาตามที่กล่าวมา ระบบ Neon Snake Game สามารถสรุปกระบวนการทำงานโดยรวมได้ดังนี้ เริ่มจากรับสัญญาณภาพจากกล้องของผู้เล่น ผ่านการตรวจจับท่าทางมือด้วย MediaPipe Hands และตรวจจับใบหน้ากับ face-api.js เมื่อมีการเข้าสู่โหมด Face Yoga Challenge จากนั้นข้อมูล gesture จะถูกส่งต่อไปยังกลไกควบคุมเกม (Game Logic) ซึ่งจะคอยอัปเดตสถานะงู อาหาร อุปสรรคพิเศษ คะแนน achievement และระดับความยากต่าง ๆ ทั้งหมดจะแสดงผลผ่านอินเทอร์เฟซที่ออกแบบในสไตล์ neon พร้อมกับระบบแสดงสถิติ สถานะ และคำแนะนำแบบเรียลไทม์ได้อย่างเป็นระบบ

บทที่ 4
ผลการดำเนินโครงการ

1. ผลการพัฒนาโปรแกรม

จากการดำเนินโครงการ Neon Snake Game (Hand Tracking Edition) ระบบที่ได้มีคุณสมบัติและองค์ประกอบดังนี้:

1.4 **ระบบเกมงู** ผู้เล่นสามารถควบคุมทิศทางการเคลื่อนที่ของงูด้วยการชี้นิ้วผ่านกล้องแบบเรียลไทม์ โดยใช้โมเดล MediaPipe Hands ของ Google สำหรับตรวจจับมือและตำแหน่งนิ้วมือ

1.5 **การจับตำแหน่งปลายนิ้ว (index finger tip)** จะถูกนำมาคำนวณเพื่อแปลงเป็นคำสั่ง “ขึ้น ลง ซ้าย ขวา” ทำให้งูตอบสนองต่อท่าทางผู้ใช้ได้ทันที

1.6 **การประมวลผลภาพและกล้อง** ใช้ Webcam และ TensorFlow.js เพื่อดึงภาพและประมวลผลในเบราว์เซอร์พร้อมทั้งแสดงผลแบบเรียลไทม์บน canvas มี overlay จุด (landmark) และ skeleton มือและใบหน้า

1.7 **สำหรับฟีเจอร์พิเศษ โครงการได้ดัดแปลงจากต้นแบบ Snake Face** (ที่ใช้การหมุนศีรษะ) มาทำ control ด้วยนิ้ว และพัฒนาเพิ่ม ดังนี้

- 1.7.1 ระบบ Level และ Progress Bar สำหรับวัดความก้าวหน้าของผู้เล่น
- 1.7.2 ระบบ Pose Challenge ตอนเปลี่ยนเลเวล โดยใช้ face-api.js ในการตรวจจับใบหน้า
- 1.7.3 Achievement แสดงรางวัลต่าง ๆ เมื่อถึงเงื่อนไขที่กำหนด
- 1.7.4 Electric Orb (ไอเท็มอันตราย) ที่ลดคะแนนหากงูชน
- 1.7.5 Dynamic Sound Effect ที่หลากหลาย เช่น เสียงกินอาหาร ช็อตไฟ หรือเลเวลอัพ
- 1.7.6 ตกแต่ง UI และพื้นหลังสไตล์ Neon grid เพิ่มความน่าสนใจและย้อนยุค
- 1.7.7 การโชว์และสัดส่วนที่ทำการตรวจจับตรงบริเวณใบหน้าและมือของผู้เล่น

2. ผลการทดสอบระบบ

จากการทดสอบระบบบนอุปกรณ์ PC พบว่า:

รายการทดสอบ	ผลลัพธ์/สรุป	หมายเหตุ
การตรวจจับนิ้วด้วยกล้อง	ตรวจจับได้แม่นยำและต่อเนื่อง High Precision	ในสภาพแสงปกติ
การตอบสนองของงู	เคลื่อนที่ลื่นไหล ไม่มีอาการหน่วงที่ชัดเจน	FPS เฉลี่ย 50–69
ระบบเสียงและกราฟิก	แสดงผลครบถ้วนตามฟังก์ชัน	
ระบบ Pose Challenge	ต้องใช้แสงกล้องเพียงพอ	แนะนำให้ใช้กล้อง HD
การบันทึกคะแนน	สามารถบันทึกข้อมูลและสถิติได้ถูกต้อง	ใช้งานได้ตามวัตถุประสงค์

ข้อสังเกต:

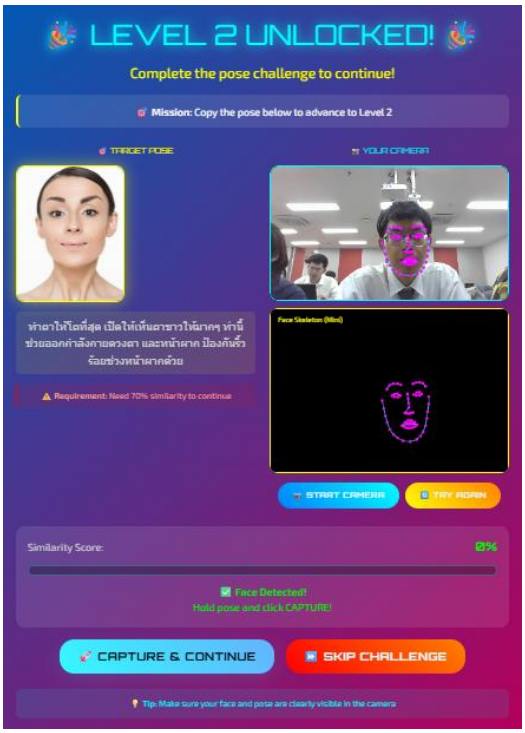
1.ระบบมีประสิทธิภาพดีที่สุดในสภาพระดับแสงที่ปกติและในสภาพกล้องที่มีความละเอียดสูง

2.การทดสอบการเล่นจริงพบว่าผู้ใช้สามารถปรับตัวกับ gesture-based control ได้ภายในไม่กี่รอบและเกิดความสนุกจากฟีเจอร์เสริม เช่น Achievement และ Pose Challenge

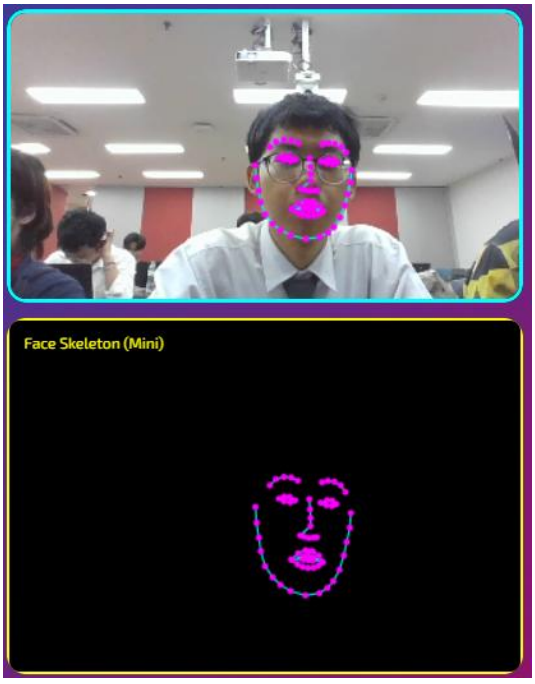
3. ตัวอย่างผลลัพธ์ของระบบ

3.1 ภาพหน้าจอการเล่นเกม (แสดงงู, อาหาร, orb, และ progress bar)

3.2 ภาพหน้าจอการเข้าสู่โหมด Pose Challenge



3.3 ภาพการตรวจจับใบหน้าและ skeleton จาก face-api.js



บทที่ 5
สรุปผลการทำโครงการ อภิปรายผล และข้อเสนอแนะ

สรุปผลการทำโครงการ

โครงการ Neon Snake Game: Hand-Tracking Edition มีจุดมุ่งหมายเพื่อพัฒนาเกมงูที่ควบคุมทิศทางผ่านท่าทางนิ้วมือแบบเรียลไทม์ ด้วยการประยุกต์ใช้เทคโนโลยี Computer Vision โดยใช้โมเดล MediaPipe Hands ทำให้การเล่นเกมเป็นไปอย่างลื่นไหล ผู้เล่นไม่จำเป็นต้องใช้ keyboard หรือ touch screen อีกต่อไป ตัวเกมได้พัฒนาฟีเจอร์เสริม เช่น Level system, Electric Orb, Progress Bar, Achievement รวมถึง Pose Challenge ด้วย face-api.js เพื่อเพิ่มความท้าทายและความสนุกในการเล่น

จากการทดสอบระบบ พบว่า MediaPipe Hands สามารถตรวจจับและติดตามพิกัดนิ้วมือได้อย่างแม่นยำและตอบสนองได้ทันที (Real-time responsiveness) ในสภาพแสงปกติ โดยผู้เล่นสามารถควบคุมทิศทางงูได้ตามความต้องการโดยไม่พบความหน่วงที่ส่งผลกระทบต่อการเล่นและให้ประสบการณ์ใหม่ที่แตกต่างจากเกมงูแบบเดิมๆ ส่งผลให้ผู้เล่นมีแรงจูงใจและสนุกกับการปลดล็อค achievement และฟีเจอร์พิเศษต่างๆของตัวเกม ทำให้เกมมีความน่าสนใจและมีการโต้ตอบกับผู้เล่นมากขึ้นได้

อภิปรายผล

ผลการดำเนินโครงการพบว่าเทคโนโลยี Computer Vision สามารถนำมาประยุกต์ควบคุมเกมแบบ real-time ได้จริง ทำให้เกิด user experience ใหม่ที่สะดวกและเข้าถึงกลุ่มผู้เล่นที่หลากหลาย อย่างไรก็ตามยังมีข้อจำกัดสำคัญ เช่น ประสิทธิภาพตรวจจับลดลงในแสงน้อย รวมถึงกล้องที่มีความละเอียดต่ำ ซึ่งส่งผลต่อ FPS และความแม่นยำของ gesture หรือฟีเจอร์บางส่วน เช่น Pose Challenge ที่อาจทำให้ FPS ลดลง อย่างไรก็ตามตัวเกมจะทำงานได้มีประสิทธิภาพที่สุดเมื่อใช้กล้องและแสงที่เหมาะสม

จากการพัฒนาและพิสูจน์แนวคิด พบว่าฟีเจอร์พิเศษที่เพิ่มเข้ามา (เช่น Electric Orb, ระบบ achievement, เสียงประกอบ, UI neon) ช่วยสร้างแรงจูงใจและเพิ่มความสนุก ผู้เล่นเรียนรู้และปรับตัวกับ gesture-based control ได้ดีภายในไม่กี่รอบ

ตารางสรุป: ปัญหาและอุปสรรค และแนวทางแก้ไขระหว่างการทำโครงการ

ปัญหาหรืออุปสรรคที่พบของสมาชิกภายในกลุ่มระหว่างทำเนินการทำโครงการ	วิธีการแก้ไข / แนวทางปรับปรุง
1.ความแม่นยำลดลงเมื่อแสงน้อย ระบบอาจจับท่าทางนิ้วผิดพลาด/หลุด	แนะนำให้ใช้งานในที่แสงเพียงพอ, เพิ่มฟังก์ชันแจ้งเตือนกรณีแสงไม่เหมาะสม, ปรับปรุงการปรับ threshold อัตโนมัติ
2.กล้อง/PC รุ่นเก่า FPS ต่ำอาจจะส่งผลทำให้เกมกระตุก/หน่วง/ควบคุมไม่ทัน	ลดขนาดโมเดล, พัฒนาโหมด low-resource, ปรับความละเอียดภาพอัตโนมัติ, แนะนำอุปกรณ์ที่เหมาะสม
3.ตรวจจับ gesture บางท่าได้ยาก ส่งผลทำให้บังคับงูผิดทิศ, ผู้เล่นเกิดอาการหงุดหงิดได้	ปรับโมเดลและ threshold ให้เหมาะกับขนาดมือหลากหลาย, เพิ่มระบบ calibration/สอน gesture ก่อนเล่น
4.ระบบ Pose Challenge ต้องการแสงสูง ทำให้การ Landmark ใบหน้าไม่ครบ, ด่านผ่านยาก	เพิ่มฟีเจอร์ ให้ผู้เล่นสามารถ ข้ามหรือ skips ในส่วนมินิเกมตรงนี้ได้

ข้อเสนอแนะ

1.ควรปรับ threshold ตรวจจับให้สามารถปรับอัตโนมัติตาม lighting หรือขนาดมือ

การตั้งค่า threshold แบบคงที่อาจไม่เหมาะกับทุกผู้ใช้หรือในทุกสภาพแสงการใช้งาน เช่น แสงน้อย หรือมือของเด็กอาจทำให้ระบบจับ gesture หลุดได้ง่าย. การวิเคราะห์ความสว่างภาพและข้อมูลขนาดมือแบบอัตโนมัติจะช่วยให้โมเดลปรับค่า threshold ได้เอง เพิ่มความแม่นยำและลดข้อผิดพลาดในแต่ละสถานการณ์ต่างๆได้

2.เพิ่มเทคนิค smoothing, temporal averaging เพื่อลด jitter ของ gesture ที่เกิดการสั่น

Hand tracking แบบเรียลไทม์ อาจมีปัญหาค่าตำแหน่งนิ้วมือสั่นหรือเปลี่ยนแปลงเร็วเกินไป (jitter) ส่งผลให้การควบคุมเกมผิดเพี้ยน การใช้เทคนิค smoothing หรือค่าเฉลี่ยข้ามเวลา (temporal averaging) จะช่วยให้การเคลื่อนไหวดูต่อเนื่องและ gameplay ลื่นไหลขึ้น โดยไม่เกิดการผิดทิศบ่อยๆ.

3.รองรับ gesture หลายแบบ เช่น หยุดงู, สองมือเร่งความเร็ว, การเล่นแบบ multiplayer

การขยายรูปแบบ gesture เช่น เพิ่มการกำมือเพื่อหยุดงู การใช้งานสองมือร่วมกัน หรือฟีเจอร์ผู้เล่นหลายคน (multiplayer) จะช่วยเพิ่มความหลากหลายและความสนุกในเกม รวมถึงช่วยให้เกมรองรับผู้ใช้ได้ทุกช่วงวัย/ความต้องการมากขึ้น. การออกแบบ gesture แบบใหม่จะต้องทดสอบการใช้งานจริงและปรับโค้ด/โมเดลให้รองรับ movement ที่หลากหลาย

4.พัฒนา Mobile Optimization ลดขนาดโมเดลและปรับ UI ให้รองรับสำหรับมือถือและเพิ่มระบบ leaderboard เปรียบเทียบคะแนนแบบออนไลน์

เพื่อตอบโจทย์ผู้ใช้บนมือถือและอุปกรณ์ low-end ควรลดขนาดโมเดลให้โหลด/ประมวลผลได้รวดเร็ว และออกแบบ UI ที่เหมาะกับหน้าจอขนาดเล็ก เพิ่มระบบแข่งขัน/leaderboard ออนไลน์เพื่อเปิดกว้างให้ผู้เล่นได้เปรียบเทียบและเกิด community ภายในตัวเกม

5.นำไปต่อยอดด้านเกมฝึกสมาธิ, สื่อการเรียนรู้สำหรับเด็ก, หรือโปรแกรมบำบัดด้านการเคลื่อนไหว

เทคโนโลยี hand tracking และ gesture recognition สามารถนำไปปรับใช้ในงานด้านอื่นที่ไม่ใช่เกม เช่น สื่อการสอน interactive สำหรับเด็ก ให้ฝึกควบคุมด้วย gesture เกมแนวฝึกสมาธิต่างๆได้

บรรณานุกรม

รูเบนสไตน์, พอล. (2563). Snake Face: เกมงูที่ควบคุมด้วยการเคลื่อนไหวของศีรษะโดยใช้ TensorFlow.js และ MediaPipe. จาก: https://paruby.github.io/snake-face/

Google Research. (2566). MediaPipe Hands – ระบบตรวจจับมือและนิ้วแบบเรียลไทม์.

จาก: https://developers.google.com/mediapipe/solutions/vision/hand_landmarker

Face-api.js Documentation. (2023). JavaScript API for face detection and face landmark recognition.

จาก: https://github.com/justadudewhohacks/face-api.js

TensorFlow.js. (2566). *ไลบรารีจาวาสคริปต์สำหรับการฝึกและประยุกต์โมเดลแมชชีนเลิร์นนิงในเว็บเบราว์เซอร์.* จาก: https://www.tensorflow.org/js

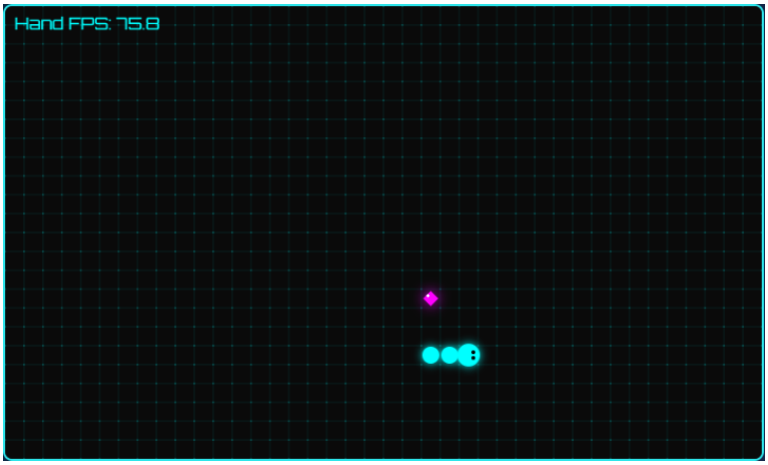
ภาคผนวก

ภาคผนวก ก : ตัวอย่างภาพหน้าจอของระบบ (Screenshots)

- หน้าเริ่มต้นของเกม (Start / Menu Screen)

ภาพที่ 1 หน้าจอเริ่มเกม

- หน้าขณะเล่นเกม (Game Play Screen)



ภาพที่ 2 หน้าจอหลักของเกม

- หน้าจอเมื่อจบเกม / แสดงคะแนน (Game Over Screen)

ภาพที่ 3 หน้าจอ Game Over

- หน้าต่างกล้องตรวจจับมือ (Hand Tracking View)

ภาพที่ 4 กล้องตรวจจับมือ

ภาคผนวก ข : โค้ดสำคัญของระบบ (Core Code Snippets)

- ฟังก์ชันตรวจจับมือ (hand detection function)

```
// ตัวจัดการผลลัพธ์การติดตามมือ
Tabnine | Edit | Test | Explain | Document
function onHandResults(results) {
  handLatency = performance.now() - handTrackingStartTime; // 2. หยุดจับเวลาและบันทึกค่า
  if (!handCtx || !handCanvas) return;

  // ล้าง canvas มือ
  handCtx.clearRect(0, 0, handCanvas.width, handCanvas.height);

  // วาดข้อความสถานะ
  handCtx.fillStyle = '#00ffff';
  handCtx.font = '12px Arial';
  handCtx.textAlign = 'center';

  try {
    if (results.multiHandLandmarks && results.multiHandLandmarks.length > 0) {
      const landmarks = results.multiHandLandmarks[0];

      // วาดจุดสังเกตมือ
      if (typeof drawConnectors !== 'undefined' && typeof drawLandmarks !== 'undefined') {
        drawConnectors(handCtx, landmarks, HAND_CONNECTIONS, {color: '#00ffff', lineWidth: 2});
        drawLandmarks(handCtx, landmarks, {color: '#ff00ff', lineWidth: 1, radius: 3});
      } else {
        // Fallback: วาดจุด landmarks เอง
        drawCustomHandLandmarks(handCtx, landmarks);
      }

      // รับตำแหน่งปลายนิ้วชี้ (landmark 8)
      const indexFingerTip = landmarks[8];
```

```
      // อัพเดทตำแหน่งนิ้ว
      lastFingerPosition = {
        x: indexFingerTip.x,
        y: indexFingerTip.y
      };

      // วาดตัวบ่งชี้ตำแหน่งนิ้ว
      handCtx.fillStyle = '#ffff00';
      handCtx.beginPath();
      handCtx.arc(
        indexFingerTip.x * handCanvas.width,
        indexFingerTip.y * handCanvas.height,
        8, 0, 2 * Math.PI
      );
      handCtx.fill();

      // วาดสถานะ
      handCtx.fillStyle = '#00ff00';
      handCtx.fillText('Hand Detected ✓', handCanvas.width / 2, 20);

      // ควบคุมงูตามตำแหน่งนิ้ว
      if (gameRunning && !gamePaused) {
        controlSnakeWithFinger(indexFingerTip);
      }
    } else {
      // ไม่พบมือ
      handCtx.fillStyle = '#ffff00';
      handCtx.fillText('Show your hand to camera', handCanvas.width / 2, handCanvas.height / 2 - 10);
      handCtx.fillText('Use mouse/keyboard/buttons', handCanvas.width / 2, handCanvas.height / 2 + 10);
```

- ฟังก์ชันควบคุมงูตามตำแหน่งนิ้ว

```
// ควบคุมงูด้วยตำแหน่งนิ้ว
Tabnine | Edit | Test | Explain | Document
function controlSnakeWithFinger(fingerTip) {
  const centerX = 0.5;
  const centerY = 0.5;
  const threshold = 0.15; // ความไวที่ปรับแล้ว

  const deltaX = fingerTip.x - centerX;
  const deltaY = fingerTip.y - centerY;

  // กำหนดทิศทางตามตำแหน่งนิ้วเทียบกับศูนย์กลาง
  // การแมปทิศทางที่แก้ไขแล้ว - ตอนนี้ตรงกับความคาดหวังทางภาพ
  if (Math.abs(deltaX) > Math.abs(deltaY)) {
    // การเคลื่อนที่ในแนวนอน - FIXED: left/right mapping
    if (deltaX > threshold && direction.x === 0) {
      changeDirection('left'); // ชี้นิ้วไปทางขวา = เคลื่อนที่ไปทางซ้าย (เหมือนกล้องเซลฟี่)
    } else if (deltaX < -threshold && direction.x === 0) {
      changeDirection('right'); // ชี้นิ้วไปทางซ้าย = เคลื่อนที่ไปทางขวา
    }
  } else {
    // การเคลื่อนที่ในแนวตั้ง
    if (deltaY > threshold && direction.y === 0) {
      changeDirection('down'); // ขึ้ลง = เคลื่อนที่ลง
    } else if (deltaY < -threshold && direction.y === 0) {
      changeDirection('up'); // ขี้ขึ้น = เคลื่อนที่ขึ้น
    }
  }
}
```

- ส่วนการอัปเดตเฟรมเกม (Game Loop)

```
function updateGame() {
  if (!gameRunning || gamePaused) return;

  // เคลื่อนที่เฉพาะเมื่อมีการกำหนดทิศทาง (ป้องกันเกมโอเวอร์เมื่อไม่มีอินพุต)
  if (direction.x === 0 && direction.y === 0) {
    return; // ไม่เคลื่อนที่ถ้าไม่มีการกำหนดทิศทาง
  }

  // เคลื่อนย้ายงู
  const head = {x: snake[0].x + direction.x, y: snake[0].y + direction.y};

  // การห่อหุ้มผนัง - งูทะลุผ่านผนังแทนที่จะตาย
  if (head.x < 0) {
    head.x = canvas.width - 20;
  } else if (head.x >= canvas.width) {
    head.x = 0;
  }

  if (head.y < 0) {
    head.y = canvas.height - 20;
  } else if (head.y >= canvas.height) {
    head.y = 0;
  }

  // ตรวจสอบการชนตัวเอง - ข้ามหัว (ดัชนี 0) เนื่องจากเรากำลังเปรียบเทียบกับตำแหน่งหัวใหม่
  for (let i = 1; i < snake.length; i++) {
    if (head.x === snake[i].x && head.y === snake[i].y) {
      gameOver();
      return;
    }
  }
```

```
snake.unshift(head);

// ตรวจสอบการชนอาหาร
if (head.x === food.x && head.y === food.y) {
  score += 10;
  foodCollected++;

  // เล่นเอฟเฟกต์เสียงอาหาร
  playFoodSound();

  // อัพเดท UI
  document.getElementById('currentScore').textContent = score;
  document.getElementById('snakeLength').textContent = snake.length;
  document.getElementById('foodCount').textContent = foodCollected;

  // อัพเดทความคืบหน้าเลเวล
  const progress = (foodCollected % foodPerLevel) / foodPerLevel * 100;
  document.getElementById('levelProgress').style.width = progress + '%';

  // ตรวจสอบการเลเวลอัพ
  if (foodCollected % foodPerLevel === 0) {
    levelUp();
  } else {
    const remaining = foodPerLevel - (foodCollected % foodPerLevel);
    document.getElementById('progressText').textContent = `Collect ${remaining} more food to advance to Level ${currentL
  }
```

```
    generateFood();
    generateElectricOrb(); // สร้างลูกบอลไฟฟ้าหลังจากกินอาหาร

    // การตรวจสอบความสำเร็จ
    if (score === 50) {
      showAchievement("First Milestone!", "You scored 50 points!");
    } else if (score === 100) {
      showAchievement("Century!", "You reached 100 points!");
    } else if (score === 200) {
      showAchievement("Snake Master!", "200 points achieved!");
    }
  }
  // ตรวจสอบการชนลูกบอลไฟฟ้า
  else if (electricOrb.active && head.x === electricOrb.x && head.y === electricOrb.y) {
    // เอฟเฟกต์ลูกบอลไฟฟ้า: เสีย 1 คะแนนและหดสั้นลง 1 ปล้อง
    score = Math.max(0, score - 1);

    // เล่นเอฟเฟกต์เสียงไฟฟ้า
    playElectricSound();

    // หดงูสั้นลง 1 ปล้องถ้าเป็นไปได้
    if (snake.length > 1) {
      snake.pop();
    }

    // อัพเดท UI
    document.getElementById('currentScore').textContent = score;
    document.getElementById('snakeLength').textContent = snake.length;
```

```
    // ปิดใช้งานลูกบอลไฟฟ้า
    electricOrb.active = false;

    // แสดงความสำเร็จเตือน
    showAchievement("⚡ Electric Shock!", "Lost 1 point and 1 segment!");

    // ไม่ทำให้งูยาวขึ้น (ไม่มี else clause)
  } else {
    snake.pop();
  }

}
```

- การเรียกใช้งาน TensorFlow.js (Face-API)

```
onFrame: async () => {
  // เช็คว่า modal ยังเปิดอยู่ (gamePaused)
  if (!gamePaused) return;

  // ปรับขนาด canvas ให้ตรงกับ video (เหมือนเดิม)
  if (canvasElement.width !== videoElement.videoWidth) {
    canvasElement.width = videoElement.videoWidth;
    canvasElement.height = videoElement.videoHeight;
  }

  // 5. ⭐⭐⭐ เรียกใช้ face-api.js ⭐⭐⭐
  // ใช้ TinyFaceDetector เพื่อความเร็ว และ FaceLandmark68TinyNet สำหรับ skeleton
  const detections = await faceapi.detectAllFaces(
    videoElement,
    new faceapi.TinyFaceDetectorOptions({ inputSize: 320, scoreThreshold: 0 })
  ).withFaceLandmarks(true); // <-- ✅ แก้ไขเป็น .withFaceLandmarks(true)
  console.log(detections);
  // 6. ⭐⭐⭐ วาดผลลัพธ์ด้วย face-api.js ⭐⭐⭐
  canvasCtx.clearRect(0, 0, canvasElement.width, canvasElement.height);
  if (miniCtx) miniCtx.clearRect(0, 0, miniCanvas.width, miniCanvas.height);
  lastDetections = detections;
  if (detections && detections.length > 0) {
    // --- วาดบน Canvas ใหญ่ ---
    // ปรับขนาดผลลัพธ์ให้ตรงกับ canvas ใหญ่
    const resizedDetections = faceapi.resizeResults(detections, {
      width: canvasElement.width,
      height: canvasElement.height
    });
```

```
  // ใช้วิธีวาดของ face-api.js (จะวาด skeleton 68 จุด)
  faceapi.draw.drawFaceLandmarks(canvasElement, resizedDetections);

  // --- วาดบน Canvas เล็ก (Mini) ---
  if (miniCtx) {
    // ปรับขนาดผลลัพธ์ให้ตรงกับ canvas เล็ก
    const resizedDetectionsMini = faceapi.resizeResults(detections, {
      width: miniCanvas.width,
      height: miniCanvas.height
    });
    // วาดลง canvas เล็ก
    faceapi.draw.drawFaceLandmarks(miniCanvas, resizedDetectionsMini);
  }

  // อัปเดต Feedback (เหมือนเดิม)
  if (feedbackEl.style.color !== "■#00ff00") {
    feedbackEl.innerHTML = "✅ <strong>Face Detected!</strong><br>Hold pose and click CAPTURE!";
    feedbackEl.style.color = "■#00ff00";
  }
  document.getElementById('levelUpCaptureBtn').disabled = false;
} else {
  // ⭐ 1. ล้าง Landmark ถ้าไม่เจอหน้า
  lastDetections = null;
```

```
// (โค้ดไม่เจอหน้า)
showLevelUpMiniFaceStatus("No face detected");
canvasCtx.fillStyle = '■#ffff00';
canvasCtx.font = '16px Arial';
canvasCtx.textAlign = 'center';
canvasCtx.fillText('No face detected', canvasElement.width / 2, canvasElement.height / 2);
```

ภาคผนวก ค : รายละเอียดชุดข้อมูล (Dataset Information)

โครงการนี้ใช้ชุดข้อมูลจาก **Google MediaPipe Hands** ซึ่งเป็นชุดข้อมูลสำหรับฝึกโมเดลตรวจจับมือและนิ้วแบบเรียลไทม์ โดยโมเดลประกอบด้วย keypoints 21 จุดต่อมือ และใช้ landmark coordinates เพื่อคำนวณทิศทางการเคลื่อนไหวของนิ้ว”