เกมเพื่อสุขภาพด้วยพื้นฐานการติดตามท่าทางมนุษย์

Healthcare Gaming based on Human Pose Tracking

นายธนิน ลิ้มปิยวรรณ

Thanin Limpiyawan

นายเฉลิมพล นวลศรี

Chalermpol Nualsri

นายอภิชาติ เปรมชูเกียรติ

Apichat Peamchookeat

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

มหาวิทยาลัยศรีนครินทรวิโรฒ ปีการศึกษา พ.ศ. 2563

คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ

ชื่อหัวข้อโครงงาน **เกมเพื่อสุขภาพด้วยพื้นฐานการติดตามท่าทางมนุษย์**
Healthcare Gaming based on Human Pose Tracking

รายชื่อนิสิต	นายธนิน	ลิ้มปิยวรรณ	60102010335
	นายเฉลิมพล	นวลศรี	60102010561
	นายอภิชาติ	เปรมชูเกียรติ	60102010586

ปริญญา วิทยาศาสตร์บัณฑิต (วท.บ.)

ภาควิชา วิทยาการคอมพิวเตอร์

อาจารย์ที่ปรึกษาโครงงาน อ.ดร.วีระ สอิ้ง

ลงชื่อ.....

(อ.ดร.วีระ สอิ้ง)

อาจารย์ที่ปรึกษาโครงงาน

บทคัดย่อ

งานวิจัยนี้มีวัตถุประสงค์เพื่อพัฒนาระบบสำหรับออกกำลังกายด้วยการเต้นเพื่อความสนุกสนาน โดยระบบที่พัฒนาขึ้นใหม่นี้จะนำท่าทางของการทำกายภาพบำบัดมาใช้ เพื่อเป็นรูปแบบของท่าเต้นที่อยู่ในการออกกำลังกาย ระบบที่พัฒนาขึ้นจะใช้หลักการของการตรวจสอบโครงสร้างกระดูกของผู้เล่น (Skeleton tracking) จากการนำกล้องสามมิติที่มีชื่อว่า Intel Realsense รุ่น D435 มาใช้ตรวจจับภาพที่มีความสามารถในวัดระยะทางระหว่างผู้ออกกำลังกายกับตำแหน่งของกล้องได้อย่างแม่นยำ เพื่อนำผลลัพธ์ที่ได้ไปควบคุมโมเดลสามมิติจำลอง โดยระบบที่พัฒนาขึ้นนี้จะทำการกำหนดลักษณะของการทำคะแนนในการออกกำลังกาย ด้วยสัญลักษณ์วงกลม ที่มีขอบเขตของการทำคะแนนอยู่ 4 รูปแบบ คือ มือซ้าย มือขวา ข้อศอกซ้าย และข้อศอกขวา เมื่อมีการเคลื่อนที่ของโมเดลสามมิติเข้าขอบเขตของตัวทำคะแนนที่กำหนด ระบบจะทำการแสดงสัญลักษณ์ 2 รูปแบบ คือ Hit หรือ Perfect เพื่อนำไปใช้ในการคำนวณคะแนนที่จะได้ หลังจากการทำท่าทางกายภาพในแต่ละครั้ง แต่ถ้าหากโดนสัญลักษณ์ที่ไม่ใช่การจบท่าทางระบบจะไม่แสดงสัญลักษณ์ใดๆ สุดท้ายเมื่อครบเวลาที่กำหนดไว้ในการออกกำลังกายระบบจะแสดงผลสรุปของการทำคะแนนคะแนน เพื่อใช้ในการกระตุ้นผู้ออกกำลังกายให้มีความต้องการที่จะทำคะแนนให้ได้มากขึ้นต่อไป หลังจากที่ได้ทำการทดลองกับอาสาสมัคร มีความพึงพอใจในการเล่นเกมในระดับมากที่สุดที่ 54.2% และมีความกระตุ้นให้อยากออกกำลังกายในระดับมากที่สุดที่ 33.3%

Abstract

This research was proposed to develop a new exercise system for funny dancing. The proposed system will adopt the posture of physical therapy to be a form of choreography in the exercise. The developed system uses the principle of skeleton tracking by using a 3D camera named the Intel Realsense model D435 that precisely measures the distance between the exerciser and camera position. This result controls a simulated 3D model in the exercise system. In addition, this system defines the nature of the scoring in the exercise by using a circle symbol that consists of four scoring areas: left hand, right hand, left elbow, and right elbow. The system will display two types of symbols, Hit or Perfect, to be used in calculating the exercise scores in each physical posture. However, the exerciser hits a missing symbol that the system will not show any symbol. Finally, the finish time of exercise will display a summary score for stimulating the exercise that needs more score. After done experiments with volunteer, the highest level of the satisfaction is 54.2% and the highest level of stimulating the exercise is 33.3%

กิตติกรรมประกาศ

การดำเนินวิจัยในครั้งนี้ต้องขอขอบคุณอาจารย์ทางภาควิชาวิทยาการคอมพิวเตอร์ที่ได้ให้ยืมใช้เครื่องมือและอุปกรณ์ต่างๆ รวมถึงห้องวิจัยในการทำวิจัยของพวกเราในครั้งนี้ที่ได้ทดสอบการทำงานของวิจัยและที่สำคัญเลยต้องขอขอบคุณอาจารย์ที่ปรึกษาวิจัยที่ได้ให้คำปรึกษา แนะนำ ช่วยเหลือ และอุปกรณ์การใช้งานต่างๆ คอยสนับสนุนตลอดระยะเวลาที่ได้ทำงานจนกระทั่งโครงงานชิ้นนี้ได้ออกมาสมบูรณ์และมีประสิทธิภาพเป็นอย่างมาก

ขอขอบคุณมหาวิทยาลัยศรีนครินทรวิโรฒแห่งนี้ ที่ได้ให้ความอำนวยสะดวกเป็นแหล่งที่ศึกษาสำหรับการทำวิจัยและยังเป็นแหล่งความรู้เรียนเพิ่มเติมที่สามารถนำมาประยุกต์และปรับปรุงให้เหมาะสมในการทำงานวิจัยครั้งนี้ได้จนสำเร็จ

ขอขอบคุณพระคุณ อ.ดร.วีระ สอิ้ง ที่คอยสนับสนุนทุกๆอย่างพร้อมให้คำแนะนำในการโครงงานครั้งนี้จนออกมาสำเร็จ

สุดท้ายนี้ทางกลุ่มนิสิตนั้นต้องขอขอบพระคุณคณะอาจารย์ทุกท่านที่ได้ให้ความรู้คำแนะนำและคอยช่วยเหลือพวกเราตั้งแต่ได้เข้ามาเรียนที่นี้จนมาถึงช่วงสุดท้าย และได้ทำวิจัยจบครั้งนี้สำเร็จไปด้วยดี ดังนั้นคณะผู้จัดทำนั้นได้หวังว่า โครงงานเกมเพื่อสุขภาพด้วยพื้นฐานของท่าทางมนุษย์(Gaming Healthcare based on Human Gesture) จะเป็นประโยชน์ต่อการศึกษาในภายภาคหน้า ตลอดจนสามารถที่จะนำไปพัฒนาต่อยอดได้ในอนาคต

สารบัญ

บทคัดย่อ.....ค

Abstract .....ง

กิตติกรรมประกาศ .....จ

สารบัญ .....ฉ

สารบัญรูปภาพ.....ซ

สารบัญตาราง .....ญ

บทที่ 1 1

บทนำ.....1

1.1 ที่มาและความสำคัญ 1

1.2 วัตถุประสงค์.....2

1.3 ขอบเขตของโครงงาน 2

1.4 ประโยชน์ที่คาดว่าจะได้รับ 2

บทที่ 2.....3

องค์ความรู้ที่เกี่ยวข้อง 3

2.1 การตรวจจับโครงสร้าง (Skeleton tracking) 3

2.2 การเชื่อมต่อกล้อง (Camera Connection).....5

2.3 การสร้างโมเดลสามมิติ 6

2.4 การพัฒนาเกม.....7

2.5 ท่ากายภาพบำบัด.....8

2.6 งานวิจัยที่เกี่ยวข้อง.....8

บทที่ 3..... 11

วิธีการดำเนินงาน 11
3.1 ขั้นตอนการดำเนินงาน 11
3.2 แผนการดำเนินงานตลอดโครงงาน 12
3.3 อุปกรณ์และเครื่องมือที่ใช้ในการทำวิจัย 13
3.4 การออกแบบระบบ 13
3.5 ปัญหาและอุปสรรค 15
บทที่ 4 16
ผลการดำเนินโครงงาน 16
บทที่ 5 37
สรุปผล อภิปรายผล และข้อเสนอแนะ 37
5.1 สรุปผลการศึกษาค้นคว้า 37
5.2 อภิปรายผล 37
5.3 ข้อเสนอแนะ 38
บรรณานุกรม 39
ภาคผนวก 41

สารบัญรูปภาพ

ภาพประกอบที่ 1 Nuitrack.....3
ภาพประกอบที่ 2 แสดงโครงกระดูกของ Nuitrack.....4
ภาพประกอบที่ 3 แสดง Module ของ Nuitrack.....4
ภาพประกอบที่ 4 Intel Realsense D435 5
ภาพประกอบที่ 5 ส่วนประกอบต่างๆ ของ Intel Realsense D435 6
ภาพประกอบที่ 6 Blender 6
ภาพประกอบที่ 7 Unity 7
ภาพประกอบที่ 8 ผู้เล่นกำลังใช้ dancing pad เพื่อเล่น Dance Dance Revolution.....7
ภาพประกอบที่ 9 แผนภาพการทำงานของระบบ..... 14
ภาพประกอบที่ 10 แสดงความสูงและระยะห่างของกล้องและคน 16
ภาพประกอบที่ 11 แสดงการใช้งาน Skeleton tracking..... 17
ภาพประกอบที่ 12 แสดงท่าทางเพื่อให้ model ขยับ..... 17
ภาพประกอบที่ 13 แสดง model ที่ขยับ 18
ภาพประกอบที่ 14 แสดงผลที่ได้เมื่อทำพร้อมกัน..... 18
ภาพประกอบที่ 15 แสดง model เมื่อทำการกลับด้านกล้อง..... 19
ภาพประกอบที่ 16 แสดงผลลัพธ์ของการกลับด้านกล้อง..... 19
ภาพประกอบที่ 17 สัญลักษณ์เพื่อทำคะแนนของข้อศอก..... 20
ภาพประกอบที่ 18 สัญลักษณ์เพื่อทำคะแนนของมือ 20
ภาพประกอบที่ 19 แสดงตำแหน่งของตัวทำคะแนนที่มือบนโมเดล..... 21
ภาพประกอบที่ 20 แสดงตำแหน่งของตัวทำคะแนนที่ข้อศอกบนโมเดล..... 21
ภาพประกอบที่ 21 เอฟเฟ็กต์เมื่อได้ Good และ Hit 22
ภาพประกอบที่ 22 เอฟเฟ็กต์เมื่อได้ Perfect..... 22
ภาพประกอบที่ 23 โมเดลที่สร้างขึ้นด้วย โปรแกรม Blender..... 23
ภาพประกอบที่ 24 ท่ากายภาพบำบัดที่ 1 ขั้นตอนที่ 1 23
ภาพประกอบที่ 25 ท่ากายภาพบำบัดที่ 1 ขั้นตอนที่ 2 24

ภาพประกอบที่ 26 ท่ากายภาพบำบัดที่ 1 ขั้นตอนที่ 3 24
ภาพประกอบที่ 27 ท่ากายภาพบำบัดที่ 2 ขั้นตอนที่ 1 25
ภาพประกอบที่ 28 ท่ากายภาพบำบัดที่ 2 ขั้นตอนที่ 2 25
ภาพประกอบที่ 29 ท่ากายภาพบำบัดที่ 2 ขั้นตอนที่ 3 26
ภาพประกอบที่ 30 ท่ากายภาพบำบัดที่ 2 ขั้นตอนที่ 4 26
ภาพประกอบที่ 31 ท่ากายภาพบำบัดที่ 2 ขั้นตอนที่ 5 27
ภาพประกอบที่ 32 ท่ากายภาพบำบัดที่ 3 ขั้นตอนที่ 1 27
ภาพประกอบที่ 33 ท่ากายภาพบำบัดที่ 3 ขั้นตอนที่ 2 28
ภาพประกอบที่ 34 ท่ากายภาพบำบัดที่ 3 ขั้นตอนที่ 3 28
ภาพประกอบที่ 35 เมนูหลัก 29
ภาพประกอบที่ 36 เมนูเลือกประเภท 29
ภาพประกอบที่ 37 หน้าต่างสรุปผลคะแนน 30
ภาพประกอบที่ 38 ทดสอบการทำงานของระบบนับคะแนนเมื่อยังไม่จบท่า 31
ภาพประกอบที่ 39 ทดสอบระบบนับคะแนนเมื่อจบท่า 31
ภาพประกอบที่ 40 วิดีโอสาธิตออกกำลังกายแสดงที่บนขวาของจอ 32
ภาพประกอบที่ 41 แสดงข้อต่อต่าง ๆ ที่ Nuitrack SDK สามารถตรวจจับได้ 33
ภาพประกอบที่ 42 หนึ่งในอาสาสมัครกำลังทดลองเล่นเกม 34
ภาพประกอบที่ 43 หนึ่งในอาสาสมัครกำลังทดลองเล่นเกม 34
ภาพประกอบที่ 44 หนึ่งในอาสาสมัครกำลังทดลองเล่นเกม 35
ภาพประกอบที่ 45 หนึ่งในอาสาสมัครกำลังทดลองเล่นเกม 35
ภาพประกอบที่ 46 กราฟความพึงพอใจที่ได้จากการทดลองเล่นเกม 36
ภาพประกอบที่ 47 กราฟความกระตุ้นเพื่อให้อยากออกกำลังกายหลังจากได้ทดลองเล่นเกม 36

สารบัญตาราง

ตารางที่ 1 ตารางแผนการดำเนินงาน... 12

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญ

เนื่องจากในปัจจุบันสถานการณ์การแพร่ระบาดของ Covid-19 มีผลกระทบต่อการดำเนินชีวิต ทำให้มีการใช้มาตรการกักตัวและงดการพบปะกัน (Social distancing) จนทำให้ไม่สามารถออกกำลังกายตามสถานที่ต่าง ๆ ได้ เช่น ฟิตเนส สวนสาธารณะ สนามกีฬา ฯลฯ ทำให้มีวิธีการออกกำลังกายอย่างจำกัด

เมื่อไม่สามารถออกกำลังกายได้อย่างเต็มที่ สิ่งที่จะช่วยให้ทุกคนหันมาสนใจกลับมาออกกำลังกายกันได้คือความสนุกในขณะที่กำลังออกกำลังกาย ยกตัวอย่างเช่นเกม Ring Fit Adventure ใน Nintendo Switch ซึ่งเป็นเกมออกกำลังกายโดยใช้จุดสนใจเป็นการขยับท่าทางของผู้เล่นในการทำท่าออกกำลังกายต่าง ๆ ผ่านการใช้งาน controller ทางผู้วิจัยจึงต้องการสร้างเกมที่ใช้ในการออกกำลังกายที่มีการขยับท่าทางผ่านทางการใช้กล้อง Intel Realsense D435

โดยกลุ่มวิจัยจะศึกษาเกี่ยวกับ Skeleton tracking โดยใช้ Nuitrack SDK ในการทำ skeleton model ผ่านการใช้กล้อง Intel Realsense D435 และ Nuitrack SDK การทำโมเดลตัวละครแบบ 3D ผ่านโปรแกรม Blender แล้วนำมา import เข้าไปในโปรแกรม Unity เพื่อดำเนินการสร้างเกมออกกำลังกายโดยใช้ท่ากายภาพบำบัด

1.2 วัตถุประสงค์

1.2.1 เพื่อให้คนส่วนใหญ่หันมาออกกำลังกายกันมากขึ้น

1.2.2 เพื่อสร้างความบันเทิงให้กับผู้ใช้ขณะออกกำลังกาย

1.2.3 เพื่อศึกษาเกี่ยวกับการใช้งาน skeleton tracking

1.2.4 เพื่อพัฒนาเกมที่มีประโยชน์และสร้างความสนุกให้แก่ผู้ใช้งาน

1.2.5 เป็นแนวทางให้ผู้ที่สนใจต่อยอดได้

1.3 ขอบเขตของโครงงาน

1.3.1 การใช้โปรแกรม Unity ในการพัฒนาเกมเพื่อสุขภาพโดยที่นำท่ากายภาบำบัดมาประยุกต์ใช้

1.3.2 การใช้ Nuitrack SDK ควบคู่กับการใช้กล้อง Intel Realsense D435 ในการทำskeleton tracking เพื่อทำโมเดลตัวละคร

1.3.3 การใช้โปรแกรม Blender ในการสร้างโมเดลตัวละครแบบ 3D

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 พัฒนาเกมที่สามารถสร้างความบันเทิงและมีประโยชน์ต่อร่างกาย โดยใช้ท่ากายภาพบำบัด

1.4.2 นำมาพัฒนาต่อยอดเพื่อใช้ในการบำบัดร่างกายผู้ป่วย

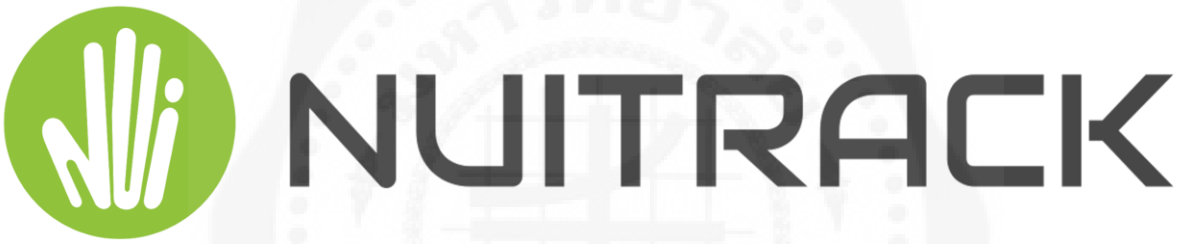
บทที่ 2

องค์ความรู้ที่เกี่ยวข้อง

2.1 การตรวจจับโครงสร้าง (Skeleton tracking)

2.1.1 Nuitrack SDK

งานวิจัยนี้เลือกใช้ Nuitrack ดังภาพประกอบที่ 1 ในการตรวจจับโครงสร้างและจดจำท่าทางของมนุษย์ ซึ่ง Nuitrack คือ Software Development Kit ที่มีความสามารถดังกล่าว และมีความสามารถของ Natural User Interface (NUI) บนหลายแพลตฟอร์มทั้งบน Android, Windows และ Linux [1]

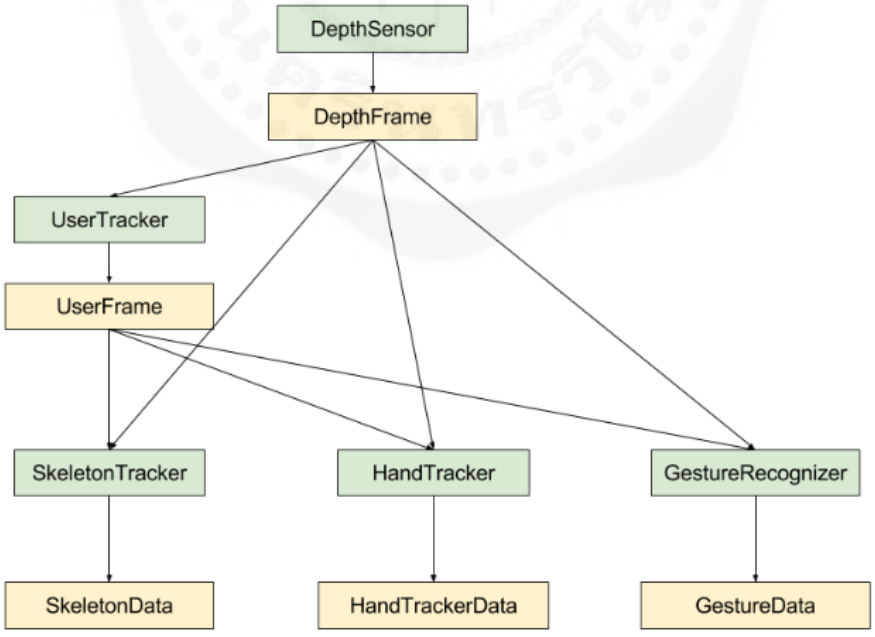


ภาพประกอบที่ 1 Nuitrack

2.1.2 Nuitrack Skeleton System

โครงกระดูกของ **Nuitrack** มี **19** ข้อต่อ โดยแต่ละข้อต่อจะมีตำแหน่งที่แตกต่างและสอดคล้องกัน ดังภาพประกอบที่ **2** [2]

ภาพประกอบที่ 2 แสดงโครงกระดูกของ Nuitrack



ภาพประกอบที่ 3 แสดง Module ของ Nuitrack

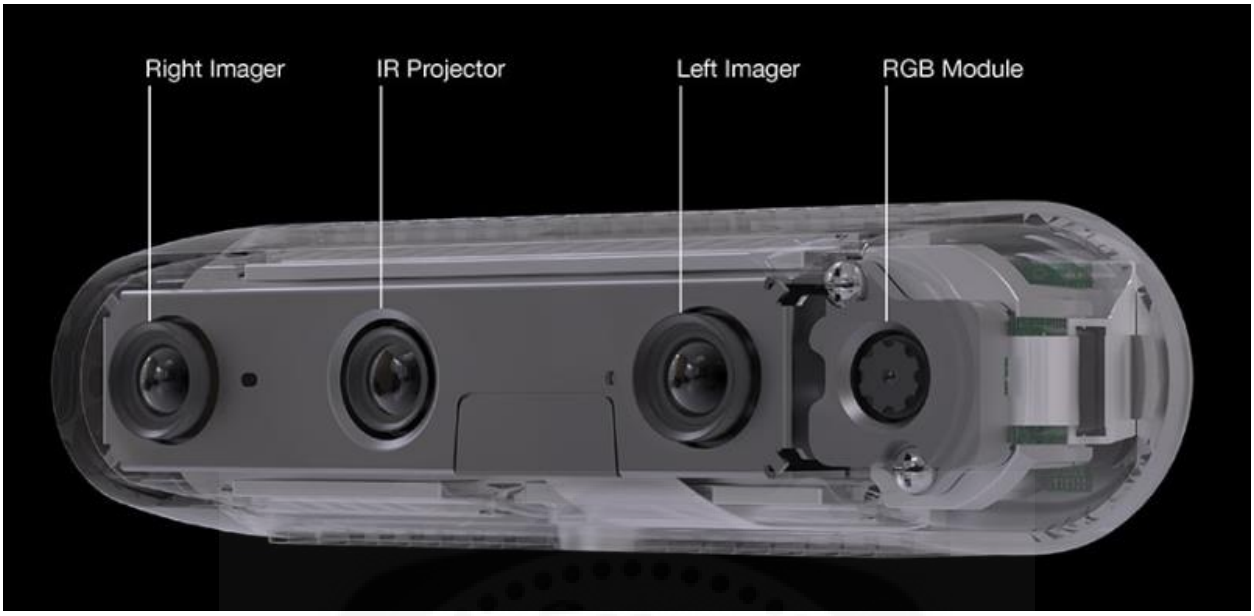
2.2 การเชื่อมต่อกล้อง (Camera Connection)

2.2.1 Intel® Realsense™ Depth Camera D435

การพัฒนาเกมเพื่อสุขภาพจำเป็นต้องใช้อุปกรณ์ที่มีความสามารถในการตรวจจับความลึก เพื่อใช้ในการตรวจจับโครงสร้างของมนุษย์ โดยงานวิจัยนี้เลือกใช้ Intel Realsense รุ่น D435 ดังภาพประกอบที่ 4 ซึ่งเป็นกล้องที่รองรับการทำงานในการวัดความลึกของภาพได้อย่างมีคุณภาพ [3]

ภาพประกอบที่ 4 Intel Realsense D435

ซึ่งกล้อง Intel Realsense D435 ประกอบไปด้วยเซนเซอร์จำนวน 2 ตัวโดยจะใช้รูปภาพภาพจากวิดีโอจากเซนเซอร์ทั้งซ้ายและขวาในการเปรียบเทียบเพื่อหาจุดสมดุลของรูปภาพในแต่ละพิกเซล กล้องวัดความลึกและ RGB Module ที่ใช้ในการตรวจจับสี ดังภาพประกอบที่ 5 [4]



ภาพประกอบที่ 5 ส่วนประกอบต่างๆ ของ Intel Realsense D435

2.3 การสร้างโมเดลสามมิติ

2.3.1 Blender

งานวิจัยนี้เลือกใช้โมเดลสามมิติในการแทนร่างกายของมนุษย์ซึ่งจะไปแทนที่กระดูกที่สร้างขึ้นทำให้โมเดลสามมิตินี้สามารถขยับได้ ซึ่งได้เลือกใช้โปรแกรม Blender ดังภาพประกอบที่ 6 ซึ่งเป็น open source โดย Blender สามารถใช้ในการสร้างภาพสามมิติ เช่น ภาพนิ่ง ภาพเคลื่อนไหวสามมิติ การทำ visual effect และการตัดต่อวิดีโอ [5]

ภาพประกอบที่ 6 Blender

2.4 การพัฒนาเกม

2.4.1 Unity

งานวิจัยนี้เลือกใช้เกมเอนจินที่มีชื่อว่า Unity ดังภาพประกอบที่ 7 สำหรับใช้ในการพัฒนาเกม ซึ่งรองรับภาษา C# และ Javascript Unity มีความสามารถในการรับ input ได้หลากหลาย เช่น คีย์บอร์ดและเมาส์ จอยสติ๊ก VR และ AR คอลโทรลเลอร์ ฯลฯ [6] และยังมีไลบรารี่ให้ใช้งานอีกมากมาย โดยงานวิจัยนี้จะใช้ไลบรารี่ Nuitrack Skeleton Tracking ในการทำให้ Unity สามารถใช้งานการตรวจจับโครงสร้างได้ [7]

ภาพประกอบที่ 7 Unity

2.4.2 Rhythm game

งานวิจัยนี้เลือก Rhythm game หรือ rhythm action ในการนำมาพัฒนาเกมเพื่อสุขภาพ ซึ่ง rhythm game เป็นเกมแนว music-themed ซึ่งเป็นส่วนหนึ่งของเกมแนว action โดยที่จะมุ่งเน้นไปทางด้านการเต้นหรือการจำลองการแสดงเครื่องดนตรีผ่านการใช้งาน input devices ต่าง ๆ เช่นการใช้ dancing pad ในการเล่นเกมเต้น ซึ่งวิธีการเล่นคือใช้เท้าเหยียบไปที่ลูกศรบน dancing pad ตามที่ปรากฏอยู่บนหน้าจอให้ตรงจังหวะ ดังภาพประกอบที่ 8 [8]

ภาพประกอบที่ 8 ผู้เล่นกำลังใช้ dancing pad เพื่อเล่น Dance Dance Revolution

2.5 ท่ากายภาพบำบัด

งานวิจัยนี้เลือกท่าทางจากการทำกายภาพบำบัดเพื่อใช้เป็นแบบในการกำหนดท่าทางและตำแหน่งของร่างกาย โดยกายภาพบำบัดคือศาสตร์ฟื้นฟูสุขภาพด้วยการออกกำลังกายและใช้อุปกรณ์พิเศษเพื่อรักษาผู้ป่วยให้กลับมาเคลื่อนไหวตามปกติได้มากที่สุดและการทำกายภาพบำบัดจะช่วยเสริมสร้างความแข็งแรงและการเคลื่อนไหวของร่างกาย [9]

2.6 งานวิจัยที่เกี่ยวข้อง

2.6.1 Psychomotor Game Learning Using Skeletal Tracking Method With Leap Motion Technology [10]

งานวิจัยนี้ได้พัฒนาแอพพลิเคชั่นในรูปแบบของเกมที่ใช้เว็บแคม ซึ่งเทคโนโลยีเว็บแคมมีข้อจำกัดคือไม่สามารถใช้งานในรูปแบบ 3 มิติได้ โดยได้เสนอการพัฒนาเกม psychomotor โดยใช้เทคโนโลยี Leap Motion เทคโนโลยีนี้สามารถใช้งานในรูปแบบ 3 มิติได้ โดยใช้วิธี Skeleton tracking เพื่อระบุและตรวจจับกระดูกและนิ้วมือมนุษย์ถูกนำมาใช้ ซึ่งได้มีการนำกล้อง IR จำนวน 2 ตัวและ LED จำนวน 3 ตัวมาใช้

ข้อดี : งานวิจัยนี้มีอัตราความแม่นยำ 96.5% สำหรับการตรวจจับมือมนุษย์ด้วยอุปกรณ์ Leap Motion และเทคโนโลยีนี้สามารถลบล้างสัญญาณรบกวนที่เกิดจากแสงแดด

ข้อเสีย : ระยะทางระหว่างมือของผู้ใช้ที่อยู่เหนืออุปกรณ์ Leap Moition ที่จะสามารถทำให้อุปกรณ์ใช้งานได้คือ 20 ถึง 50 ซม.

2.6.2 Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields [11]

งานวิจัยนี้ได้เสนอวิธีการตรวจจับท่าทางของคนหลายคนแบบ 2 มิติในรูปภาพอย่างมีประสิทธิภาพ วิธีการคือใช้ non-parametric presentation ซึ่งอ้างถึง Part Affinity Fields (PAFs) เพื่อเชื่อมโยงส่วนต่าง ๆ ของร่างกายของบุคคลในภาพ สถาปัตยกรรมนี้ได้มีการใช้ greedy bottom-up ทำให้ความแม่นยำสูงตามเวลาจริงโดยไม่คำนึงถึงจำนวนคนในภาพ สถาปัตยกรรมนี้ถูกออกแบบมาเพื่อใช้ในการเชื่อมโยงตำแหน่งเข้าด้วยกันผ่าน branches ทั้งสองในกระบวนการการทำนายในระดับเดียวกัน

ข้อดี : งานวิจัยนี้สามารถตรวจจับคนได้พร้อมกันหลายคน

ข้อเสีย : งานวิจัยนี้สามารถตรวจจับได้แค่ 2 มิติจึงทำให้เวลาที่มีการหันข้างหรือมีคนยืนซ้อนกันจะทำให้เกิดข้อผิดพลาด

2.6.3 Kinect Augmented Reality Gear Game Design [12]

งานวิจัยนี้นำเสนอเกี่ยวกับการพัฒนาเกมทางวิทยาศาสตร์และเทคโนโลยีสำหรับนักเรียนชั้นประถมด้วยการรับค่าตำแหน่งของกระดูกมนุษย์ด้วย kinect sensor เราสามารถบอกการเคลื่อนไหวของแต่ละบุคคลได้ประกอบกับท่าทาง การอ่านการเคลื่อนไหวของมือและการออกแบบสถานการณ์ภายในเกม

ข้อดี : งานวิจัยนี้มีวัตถุประสงค์คล้ายกับโปรเจคของกลุ่มวิจัยคือทำเกมที่ใช้ natural user interface จัดการกับเกมและมีแนวเกมที่น่าสนใจ

ข้อเสีย : งานวิจัยนี้มีการใช้งาน เครื่องมือที่ต่างจาก โปรเจคของเราทั้งหมด และ มีการ track ส่วนที่ใช้ คือ มือสองข้างเท่านั้น

2.6.4 Enhanced Computer Vision with Microsoft Kinect Sensor: A Review [13]

งานวิจัยนี้นำเสนอเกี่ยวกับ algorithms และ application การมองเห็นของคอมพิวเตอร์ที่ใช้ Microsoft Kinect sensor โดยจะแบ่งตามประเภทของปัญหาการมองเห็นที่สามารถแก้ไขหรือปรับปรุงได้โดยใช้ Kinect sensor หัวข้อที่ครอบคลุม ได้แก่ การประมวลผลล่วงหน้า การติดตามและการจดจำวัตถุ การวิเคราะห์กิจกรรมของมนุษย์มือ และการวิเคราะห์ท่าทางและการทำแผนที่ 3 มิติในร่ม

ข้อดี : งานวิจัยนี้บอกวิธีที่หลากหลายในการเพิ่มประสิทธิภาพในการมองเห็นของคอมพิวเตอร์โดยแต่ละวิธีการจะมีการสรุปผลและเปรียบเทียบ

ข้อเสีย : งานวิจัยนี้เพียงส่วนน้อยที่สามารถนำมาปรับใช้กับงานโปรเจคได้ และมีข้อเสียเล็กน้อยเช่น สภาพพื้นที่ที่จะทำวิจัย และขอบเขตของ depth sensor ที่จะสามารถตรวจจับได้

2.6.5 Monitoring the human posture in industrial environment: a feasibility study [14]

งานวิจัย นี้กล่าวถึงวิธีการทำการเฝ้าดูท่าทางของมนุษย์ในสภาพแวดล้อมที่ไม่เอื้ออำนวย วิธีการนั้น มีพื้นฐานมากจากการทำงานร่วมกันระหว่างกล้อง 3D (Microsoft Kinect) และระบบ motion capture แบบสวม

ใส่ (notch system) ที่ใช้ใน inertial measurement units เพื่อระบุถึงท่าทางของร่างกายข้อมูลที่ได้จากการรวมกันในระดับ feature ส่งผลให้เอาชนะขีดจำกัดของวิธีการทั้งสองแบบได้ โดย

ข้อดี : งานวิจัยนี้บอกถึงข้อเสียของกล้อง depth camera และบอกถึงการแก้ไขโดยการผนวก การทำงานของ motion capture เข้ากับการใช้งานกล้อง depth camera

ข้อเสีย : งานวิจัยนี้มีอุปกรณ์ที่เกินจำเป็นสำหรับโปรเจคของเรามากจนเกินไปและงานวิจัยนี้เน้นเรื่องความแม่นยำมากเกินกว่าระดับที่ใช้ในโปรเจค

บทที่ 3

วิธีการดำเนินงาน

3.1 ขั้นตอนการดำเนินงาน

3.1.1 ศึกษาค้นคว้าเอกสาร และงานวิจัยที่เกี่ยวข้อง

3.1.2 วางแผนการทำงานและออกแบบระบบ

3.1.3 พัฒนา Model

3.1.4 เพิ่มระบบเกมออกกำลังกายโดยใช้ท่ากายภาพบำบัด

3.1.5 ทดสอบการทำงานของระบบ

3.1.6 แก้ไขข้อผิดพลาด

3.1.7 นำเกมมาให้อาสาสมัครเล่น

3.1.8 ให้อาสาสมัครทำแบบสอบถาม

3.1.9 สรุปผลงานวิจัย

3.2 แผนการดำเนินงานตลอดโครงงาน

แผนการดำเนินงานวิจัย	**ปีการศึกษา 2563**			
	ม.ค. 64	**ก.พ. 64**	**มี.ค. 64**	**เม.ย. 64**
วางแผนโครงงาน				
1. วางแผนการทำงานและออกแบบระบบ				
2. ศึกษาท่ากายภาพบำบัด				
ดำเนินการทำงานวิจัย				
3. พัฒนา Model				
4. เพิ่มระบบเกมออกกำลังกายโดยใช้ท่ากายภาพบำบัด				
ทดสอบการทำงานของระบบ				
5. ทดสอบการทำงานของระบบ				
6. แก้ไขข้อผิดพลาด				
ทำการทดลอง				
7. นำเกมมาให้อาสาสมัครเล่น				
8. ให้อาสาสมัครทำแบบสอบถาม				
สรุปและประเมินผล				
9. สรุปผลงาน เพื่อนำเสนอ				

ตารางที่ 1 ตารางแผนการดำเนินงาน

3.3 อุปกรณ์และเครื่องมือที่ใช้ในการทำวิจัย

3.3.1 อุปกรณ์

3.3.1.1 Computer - CPU Core i5-3470

- RAM 12 GB

- NVIDIA GeForce 605

- Windows 10 64-bit

3.3.1.2 Intel® Realsense™ Depth Camera D435

3.3.1.3 ขาตั้งกล้อง

3.3.2 ซอฟแวร์

3.3.2.1 Unity

3.3.2.2 Blender

3.3.2.3 Nuitrack

3.3.3 ภาษาที่ใช้

3.3.3.1 C#

3.4 การออกแบบระบบ

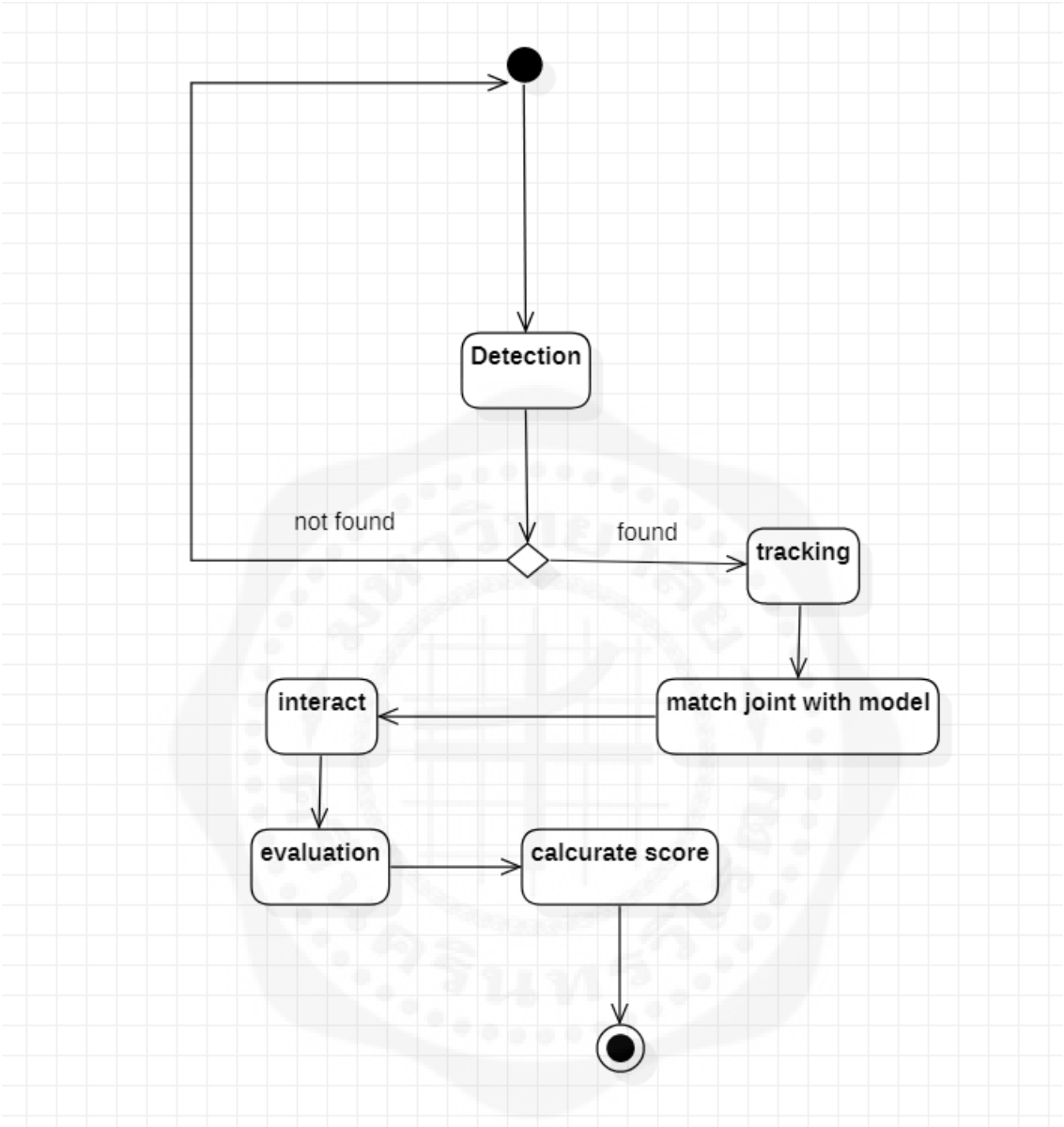
ขั้นตอนการออกแบบระบบ มี 4 ขั้นตอน ดังนี้

3.4.1 Detection คือ การตรวจจับบุคคลแบบ real time โดยการใช้กล้องตรวจจับความลึก Intel Realsense D435

3.4.2 Tracking คือ การติดตามจุดต่างๆ ของร่างกายเพื่อให้คอมพิวเตอร์สามารถรู้ว่าตำแหน่งใดคือส่วนใด โดยการใช้ Nuitrack SDK

3.4.3 Interaction คือ การมีปฏิสัมพันธ์ต่างๆ ภายในเกมเพื่อให้ได้ผลลัพธ์ เช่น การขยับมือไปทางสัญลักษณ์

3.4.4 Evaluation คือ การประเมินประสิทธิภาพของการมีปฏิสัมพันธ์กับสัญลักษณ์



ภาพประกอบที่ 9 แผนภาพการทำงานของระบบ

ขั้นตอนการทำงานของระบบ

ขั้นตอนการทำงานของระบบมีดังนี้

1. สร้าง Model ตัวละครในรูปแแบบของ 3D Model
2. ทำการ track ส่วนข้อต่อของ Model ที่ทำขึ้นเพื่อที่จะให้ model ตอบสนองต่อการเคลื่อนไหวได้
3. ทำการกำหนดตำแหน่งสัญลักษณ์
4. สัญลักษณ์จะแสดงขึ้นทีละอันจนจบเกมหรือหมดเวลา
5. ถ้าส่วนมือหรือส่วนข้อศอกของ Model สัมผัสกับสัญลักษณ์ คำนวณคะแนนที่ได้หลังจากจบท่ากายภาพ 1 ครั้ง โดยถ้าหากโดนสัญลักษณ์ที่ไม่ใช่การจบท่าจะแสดงเอฟเฟ็กต์ Hit และหากโดนสัญลักษณ์ที่เป็นการจบท่าจะแสดงเอฟเฟ็กต์ Perfect
6. สรุปคะแนนเมื่อสิ้นสุดเกม

3.5 ปัญหาและอุปสรรค

3.5.1 ประสิทธิภาพในการคำนวณ ข้อต่อของร่างกาย ยังตอบสนองได้ไม่ดีเท่าที่ต้องการ

3.5.2 การทำ model เพื่อเชื่อมกับข้อต่อของ Nuitrack SDK มีความซับซ้อนมาก

3.5.3 การจำกัดเวลาในการใช้งาน Nuitrack SDK สำหรับการใช้งานโดยไม่เสียค่าบริการ

3.5.4 การใช้งานโปรแกรมที่มีความซับซ้อนมาก ยากต่อการทำความเข้าใจ

3.5.5 มุมกล้อง ระยะห่างจากกล้อง และตำแหน่งการยืนที่ต้องกำหนดไว้

3.5.6 การ import model เข้าสู่ Unity ทำให้ model เกิดการบิดเบี้ยว

บทที่ 4

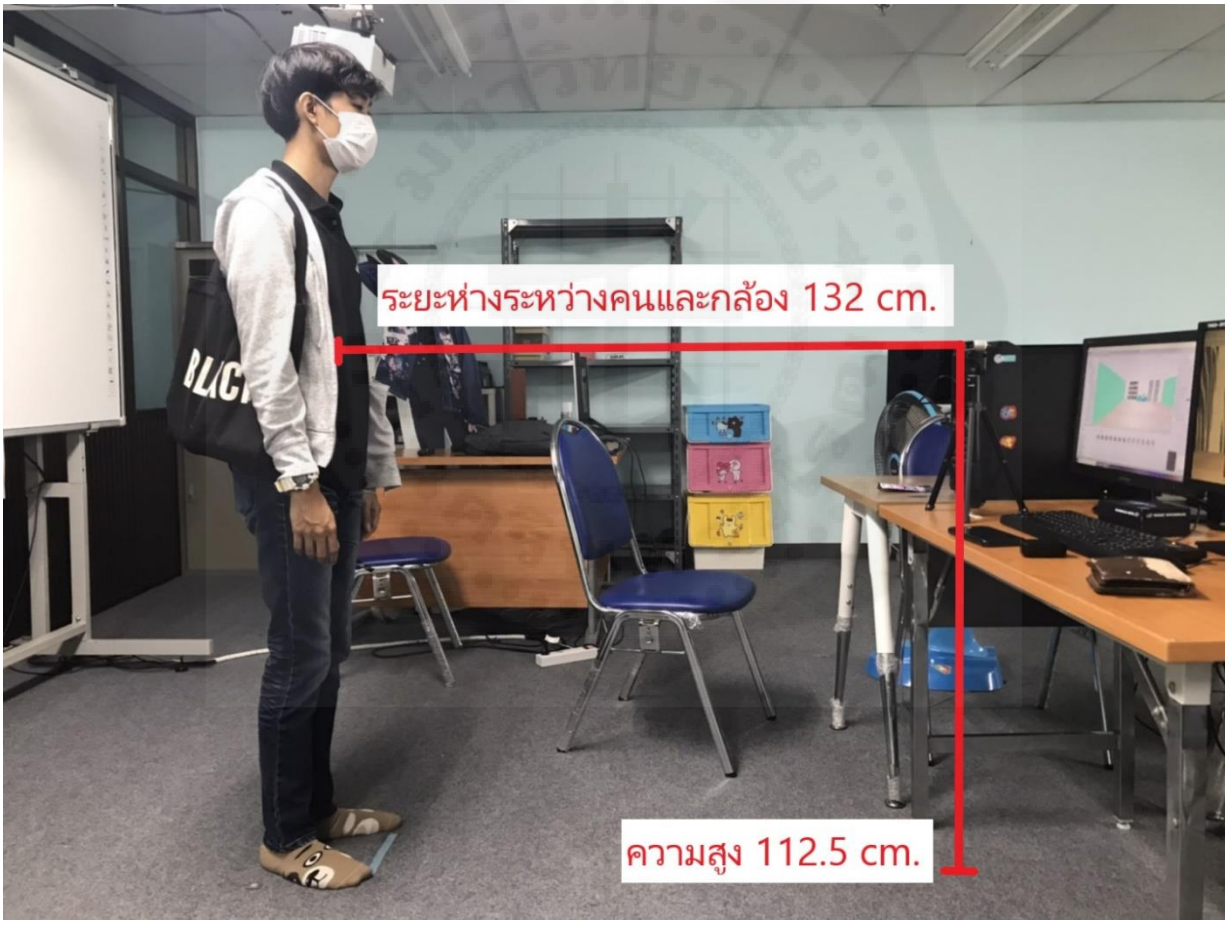
ผลการดำเนินโครงงาน

การดำเนินโครงงานในครั้งนี้ผู้จัดทำได้เสนอผลการดำเนินโครงงาน โดยแสดงผลลัพธ์ดังต่อไปนี้

4.1 ระยะความสูงของกล้องและระยะห่างของคนและกล้องดังภาพประกอบที่ 10 โดยค่าที่แนะนำคือ

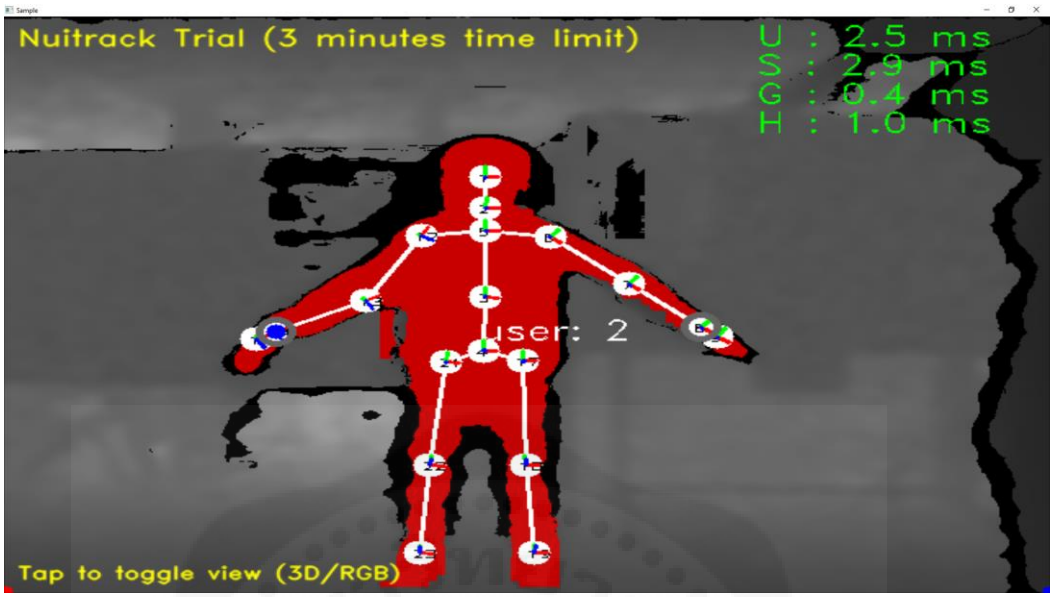
4.1.1 ความสูงประมาณ 112.5 ซม.

4.1.2 ความห่างระหว่างคนและกล้องประมาณ 132 ซม.



ภาพประกอบที่ 10 แสดงความสูงและระยะห่างของกล้องและคน

4.2 แสดง Skeleton tracking ขณะใช้ Intel Realsense D435 คู่กับ Nuitrack ดังภาพประกอบที่ 11

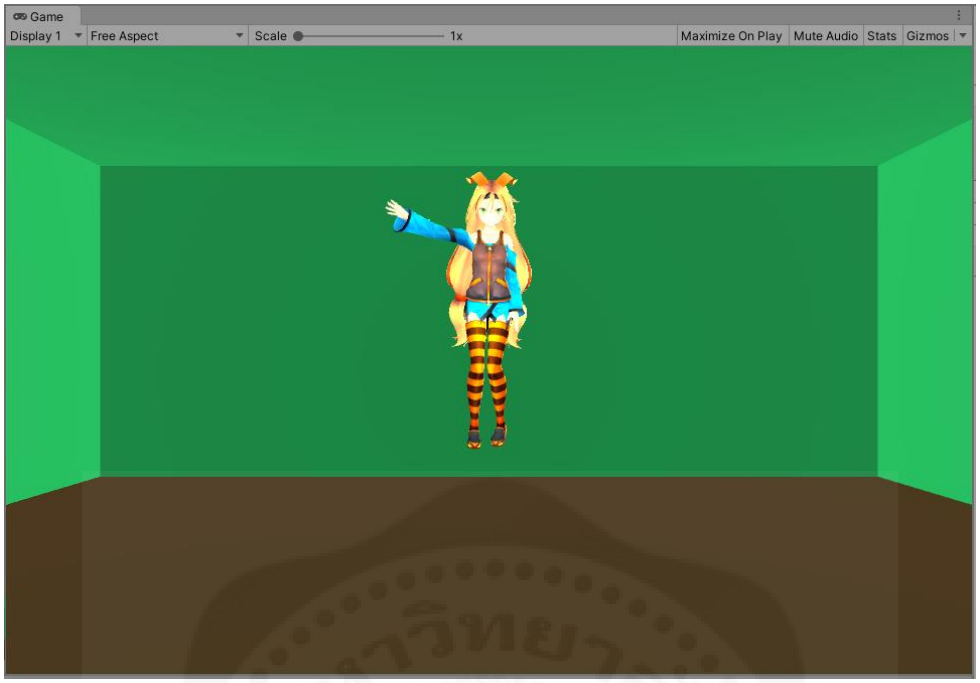


ภาพประกอบที่ 11 แสดงการใช้งาน Skeleton tracking

4.3 แสดงผลการขยับร่างกายบน Unity

4.3.1 ทดลองนำ Model ตัวละครที่ Nuitrack มีให้มาใช้งานดังภาพประกอบที่ 12, 13 และ 14

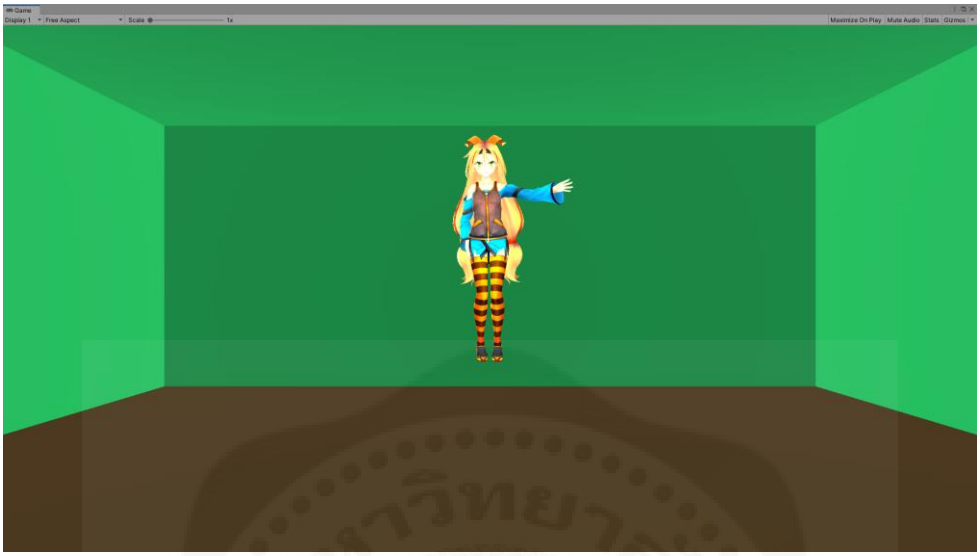
ภาพประกอบที่ 12 แสดงท่าทางเพื่อให้ model ขยับ



ภาพประกอบที่ 13 แสดง model ที่ขยับ

ภาพประกอบที่ 14 แสดงผลที่ได้เมื่อทำพร้อมกัน

4.3.2 ผลหลังจากการกลับด้านกล้องดังภาพประกอบที่ 15 และ 16

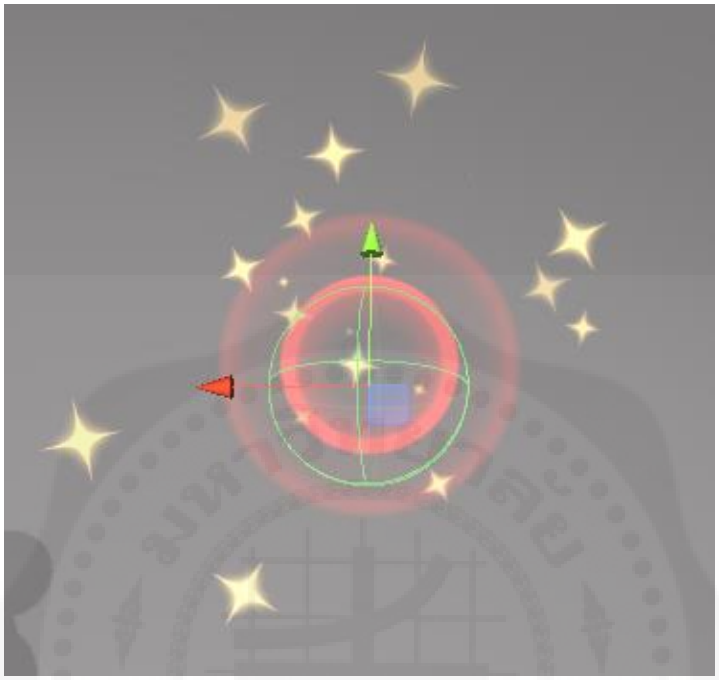


ภาพประกอบที่ 15 แสดง model เมื่อทำการกลับด้านกล้อง

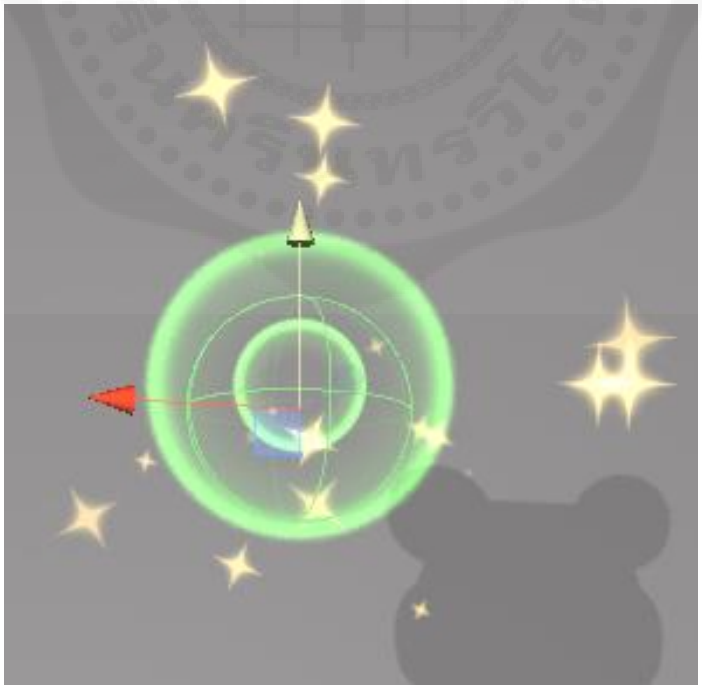
ภาพประกอบที่ 16 แสดงผลลัพธ์ของการกลับด้านกล้อง

4.4 เพิ่มระบบต่าง ๆ ภายในเกม

4.4.1 เพิ่มสัญลักษณ์เพื่อทำคะแนนของมือและข้อศอกที่ตำแหน่งต่าง ๆ ภายในเกมดังภาพประกอบที่ 17 และ 18

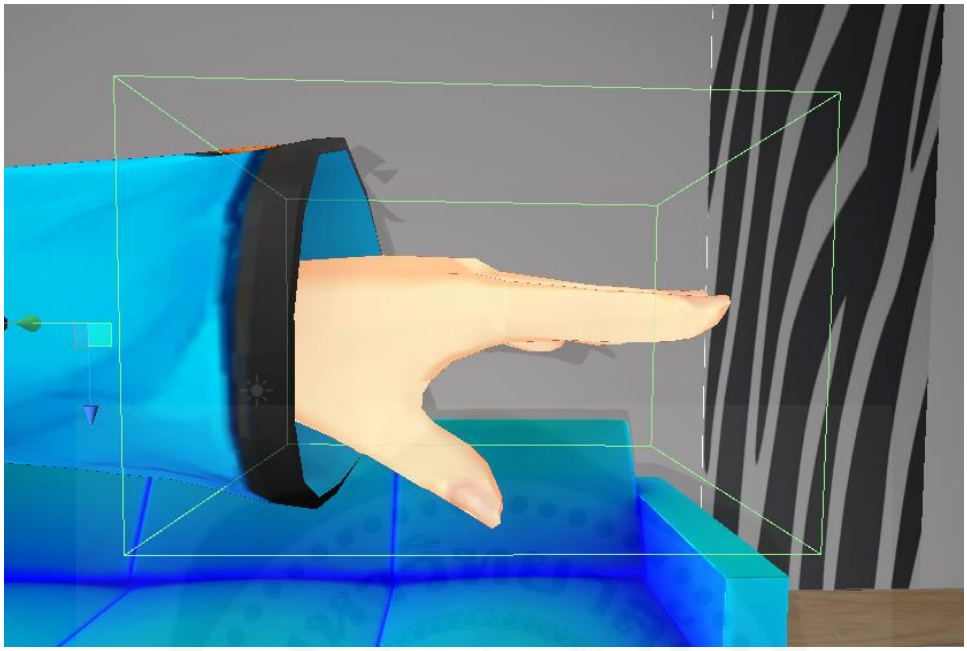


ภาพประกอบที่ 17 สัญลักษณ์เพื่อทำคะแนนของข้อศอก

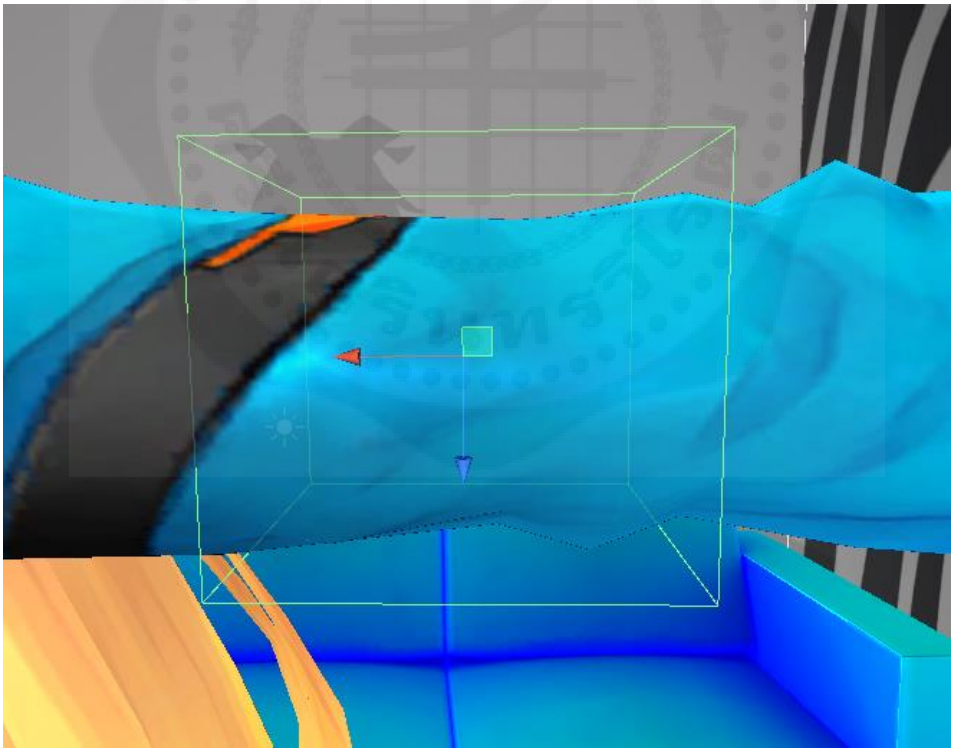


ภาพประกอบที่ 18 สัญลักษณ์เพื่อทำคะแนนของมือ

4.4.2 เพิ่มขอบเขตของตัวทำคะแนนดังภาพประกอบที่ 19



ภาพประกอบที่ 19 แสดงตำแหน่งของตัวทำคะแนนที่มือบนโมเดล



ภาพประกอบที่ 20 แสดงตำแหน่งของตัวทำคะแนนที่ข้อศอกบนโมเดล

4.4.3 เมื่อขอบเขตของตัวทำคะแนนส่วนมือหรือส่วนข้อศอกของ Model สัมผัสกับสัญลักษณ์ คำนวณคะแนนที่ได้หลังจากจบท่ากายภาพ 1 ครั้ง โดยถ้าหากโดนสัญลักษณ์ที่ไม่ใช่การจบท่าจะแสดงเอฟเฟ็กต์ Hit และหากโดนสัญลักษณ์ที่เป็นการจบท่าจะแสดงเอฟเฟ็กต์ Perfect ดังภาพประกอบที่ 21 และ 22

ภาพประกอบที่ 21 เอฟเฟ็กต์เมื่อได้ Good และ Hit

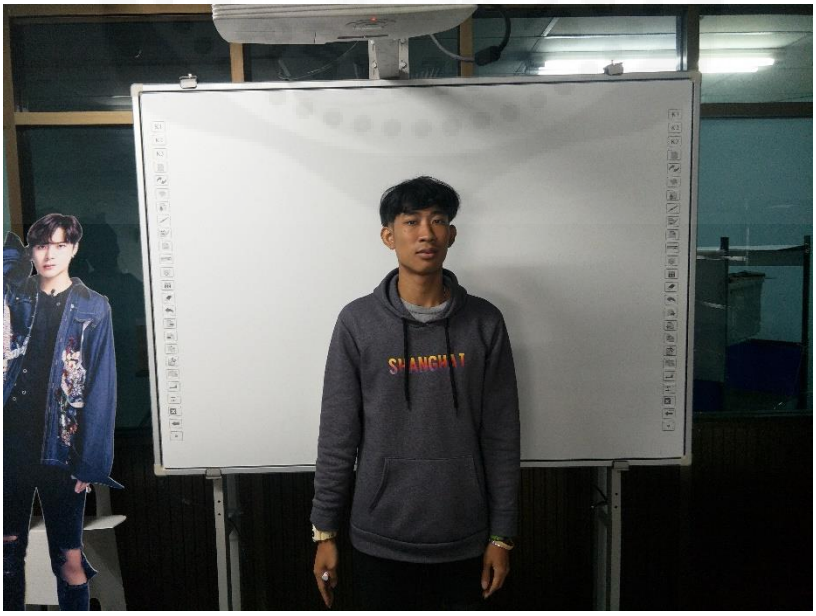
ภาพประกอบที่ 22 เอฟเฟ็กต์เมื่อได้ Perfect

4.5 ทำการสร้างโมเดลด้วย Blender แล้ว import เข้ามาใช้และนำตัวทำคะแนนมาใส่ที่มือของโมเดล ดังภาพประกอบที่ 23

ภาพประกอบที่ 23 โมเดลที่สร้างขึ้นด้วย โปรแกรม Blender

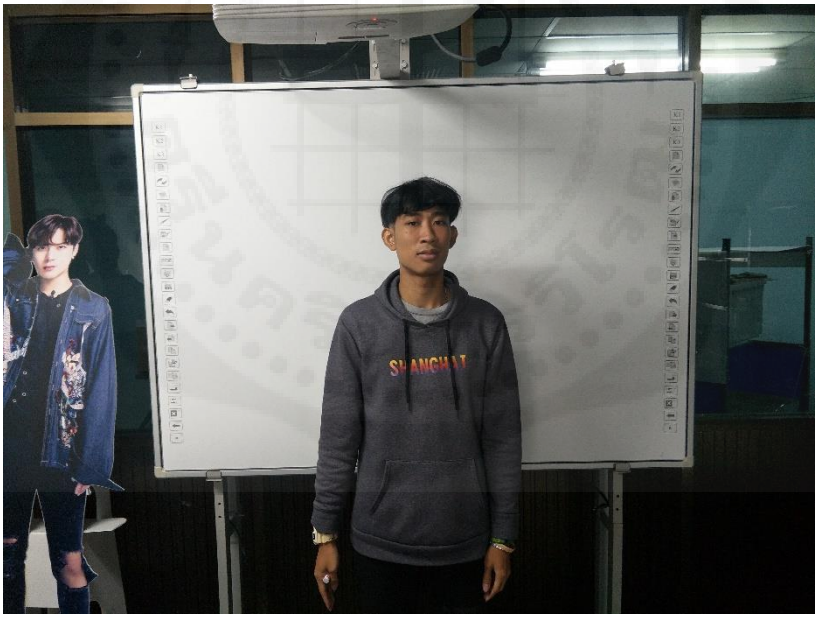
4.6 ท่ากายภาพบำบัดทั้ง 3 ท่า

4.6.1 ท่ากายภาพบำบัดที่ 1 ดังภาพประกอบที่ 24, 25 และ 26



ภาพประกอบที่ 24 ท่ากายภาพบำบัดที่ 1 ขั้นตอนที่ 1

ภาพประกอบที่ 25 ท่ากายภาพบำบัดที่ 1 ขั้นตอนที่ 2

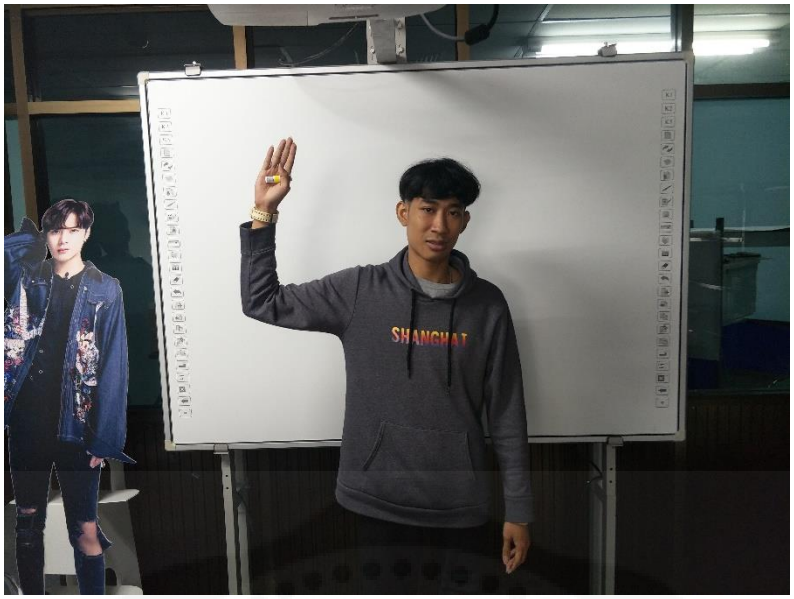


ภาพประกอบที่ 26 ท่ากายภาพบำบัดที่ 1 ขั้นตอนที่ 3

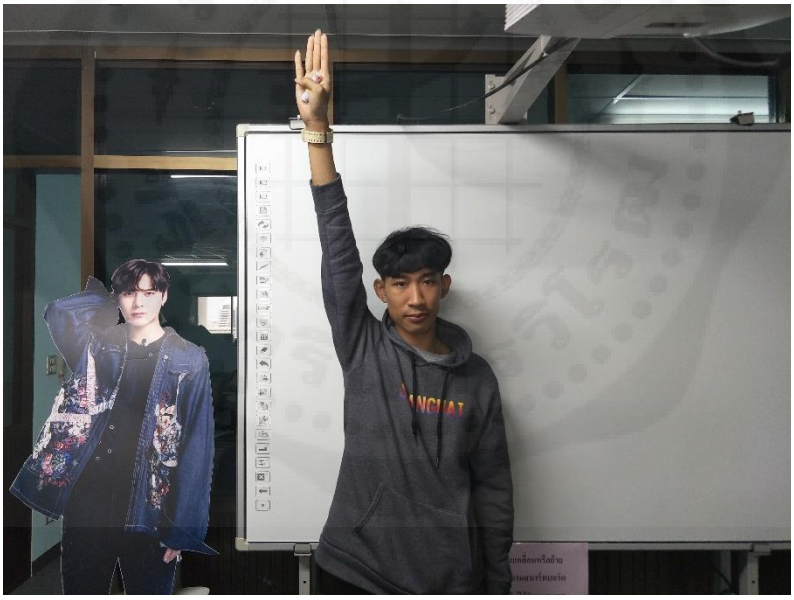
4.6.2 ท่ากายภาพบำบัดที่ 2 ดังภาพประกอบที่ 27, 28, 29, 30 และ 31

ภาพประกอบที่ 27 ท่ากายภาพบำบัดที่ 2 ขั้นตอนที่ 1

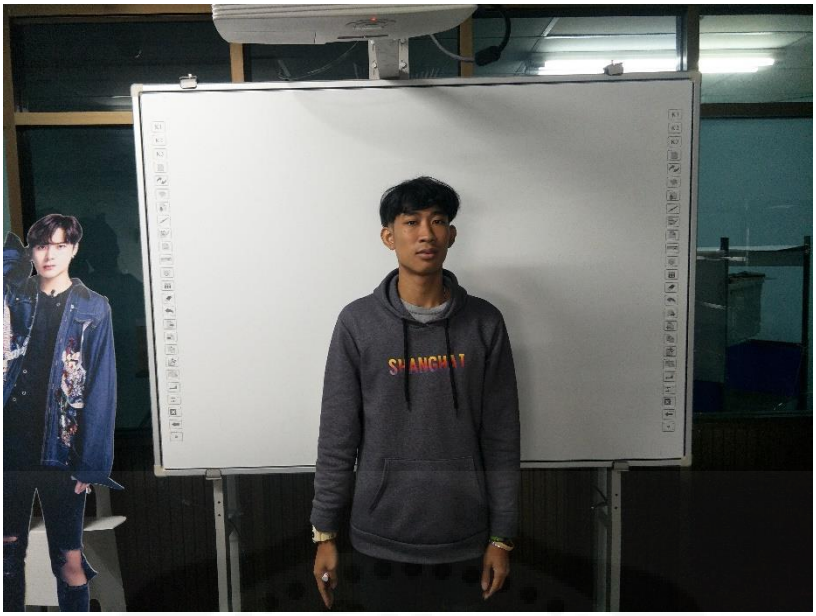
ภาพประกอบที่ 28 ท่ากายภาพบำบัดที่ 2 ขั้นตอนที่ 2



ภาพประกอบที่ 29 ท่ากายภาพบำบัดที่ 2 ขั้นตอนที่ 3

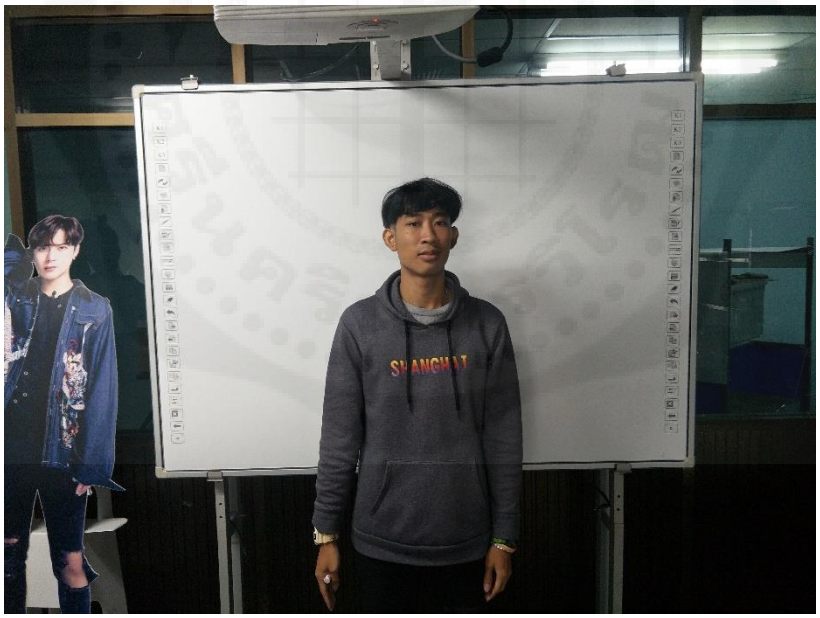


ภาพประกอบที่ 30 ท่ากายภาพบำบัดที่ 2 ขั้นตอนที่ 4



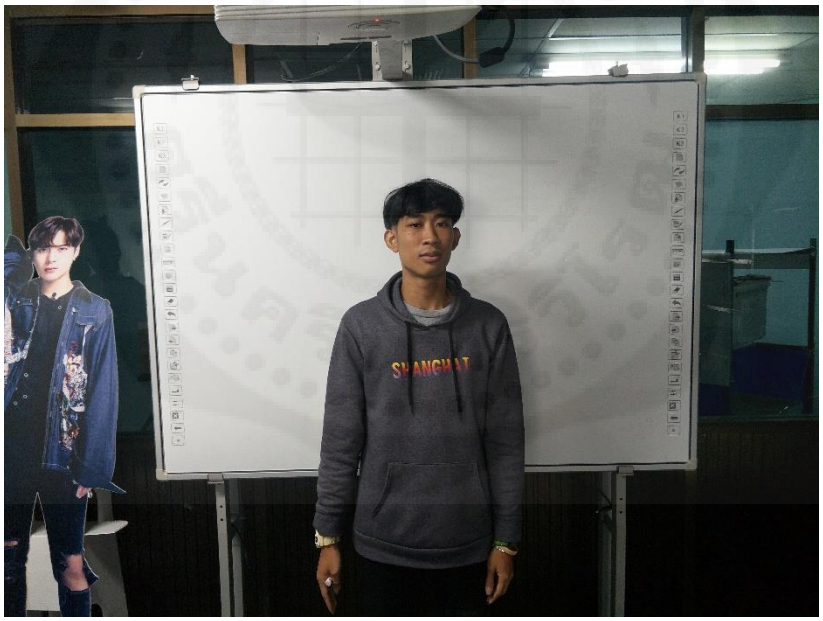
ภาพประกอบที่ 31 ท่ากายภาพบำบัดที่ 2 ขั้นตอนที่ 5

4.6.3 ท่ากายภาพบำบัดที่ 3 ดังภาพประกอบที่ 32, 33 และ 34



ภาพประกอบที่ 32 ท่ากายภาพบำบัดที่ 3 ขั้นตอนที่ 1

ภาพประกอบที่ 33 ท่ากายภาพบำบัดที่ 3 ขั้นตอนที่ 2



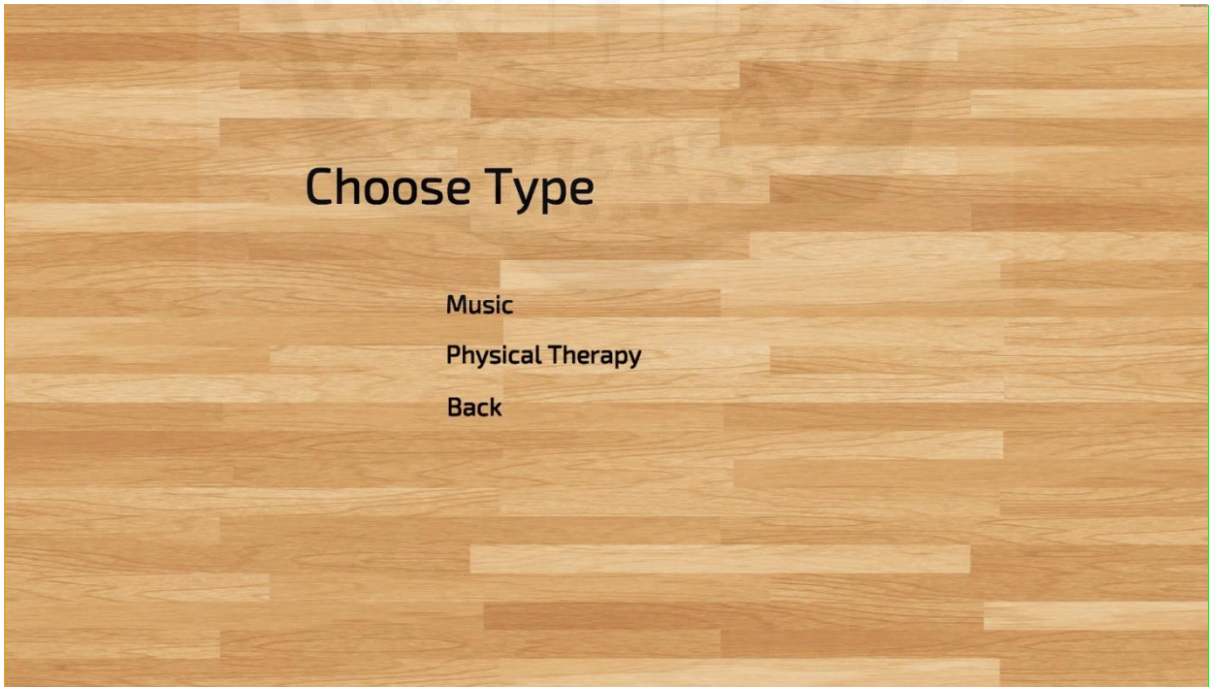
ภาพประกอบที่ 34 ท่ากายภาพบำบัดที่ 3 ขั้นตอนที่ 3

4.7 หน้าต่างแสดงอินเตอร์เฟซของเกม

4.7.1 เมนูหลัก ดังภาพประกอบที่ 35

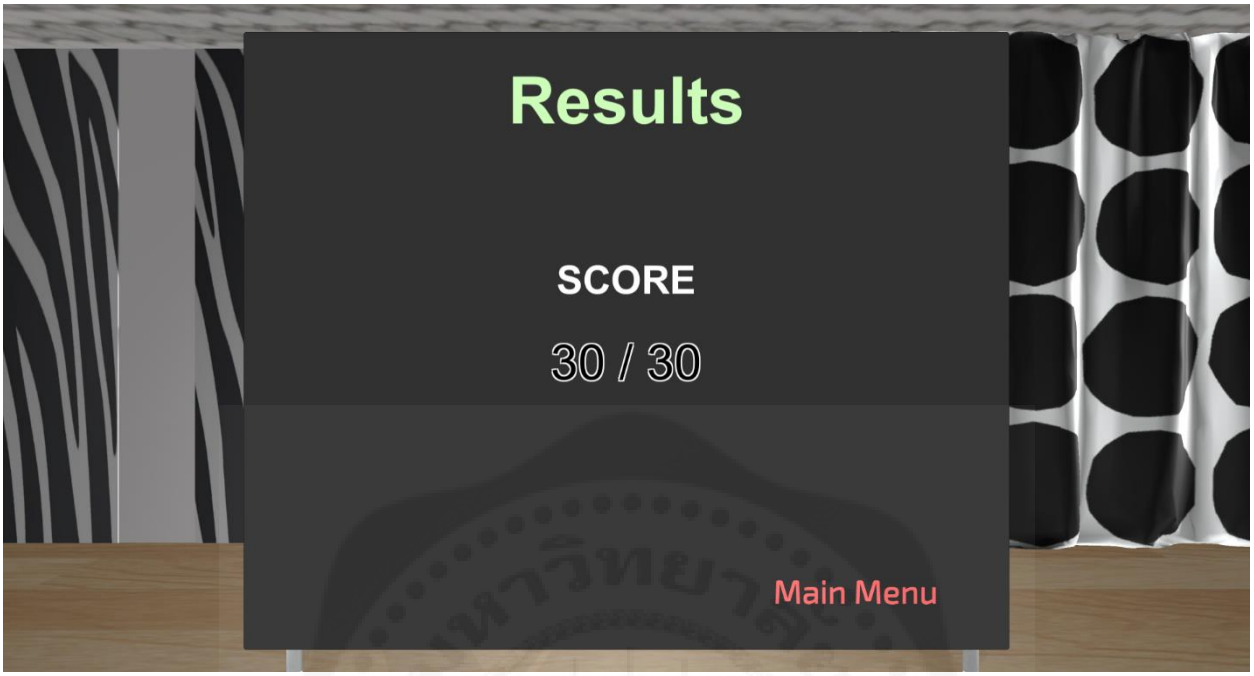
ภาพประกอบที่ 35 เมนูหลัก

4.7.2 เมนูเลือกประเภท ดังภาพประกอบที่ 36



ภาพประกอบที่ 36 เมนูเลือกประเภท

4.7.3 หน้าต่างสรุปคะแนน ดังภาพประกอบที่ 37



ภาพประกอบที่ 37 หน้าต่างสรุปผลคะแนน

4.8 ทดลองการทำงานของระบบ ถ้าขอบเขตของตัวทำคะแนนสัมผัสกับสัญลักษณ์ที่แสดงอยู่ที่ไม่ใช่สัญลักษณ์จบท่าจะแสดงเอฟเฟ็กต์ Hit ถ้าขอบเขตของตัวทำคะแนนสัมผัสกับสัญลักษณ์ที่แสดงอยู่ที่เป็นสัญลักษณ์จบท่าจะแสดงเอฟเฟ็กต์ Perfect และจะได้คะแนนเพิ่ม 1 คะแนนต่อการจบ 1 ท่า ดังภาพประกอบที่ 38 และ 39

ภาพประกอบที่ 38 ทดสอบการทำงานของระบบนับคะแนนเมื่อยังไม่จบท่า

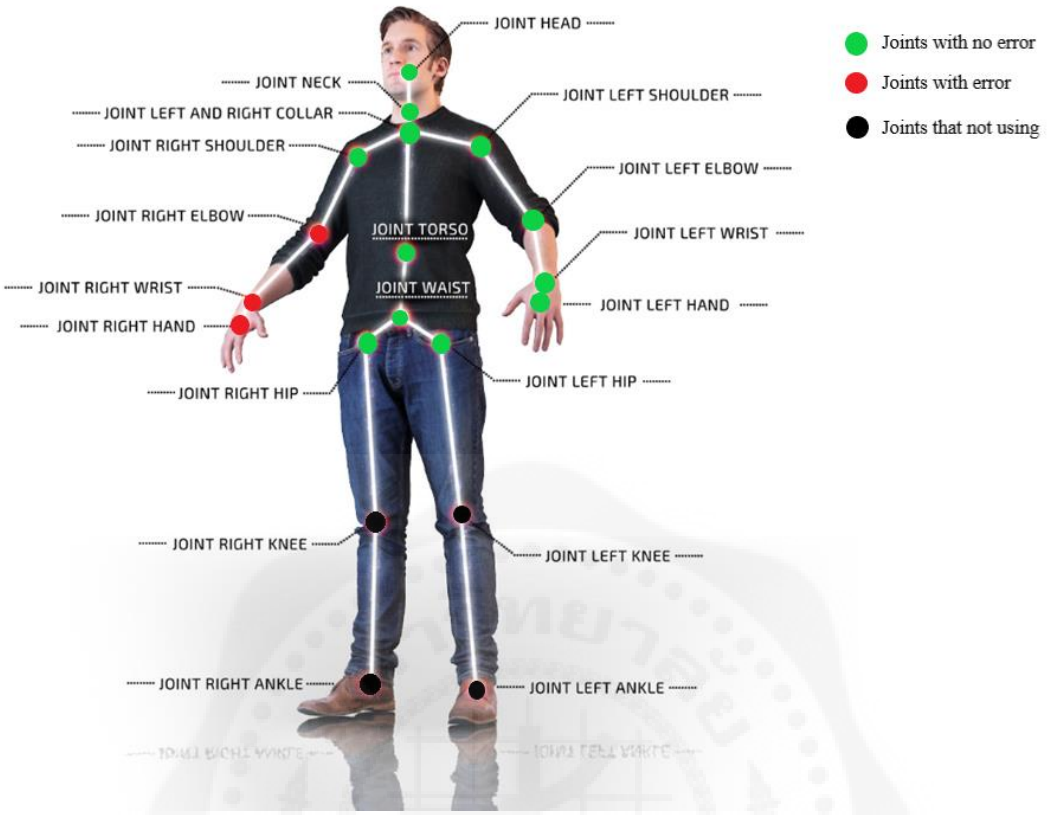
ภาพประกอบที่ 39 ทดสอบระบบนับคะแนนเมื่อจบท่า

4.9 วิดีโอสาธิตท่าแสดงออกกำลังกาย ดังภาพประกอบที่ 40

ภาพประกอบที่ 40 วิดีโอสาธิตออกกำลังกายแสดงที่บนขวาของจอ

4.10 ผลการทดลองการใช้ Nuitrack SDK ในการทำ skeleton tracking

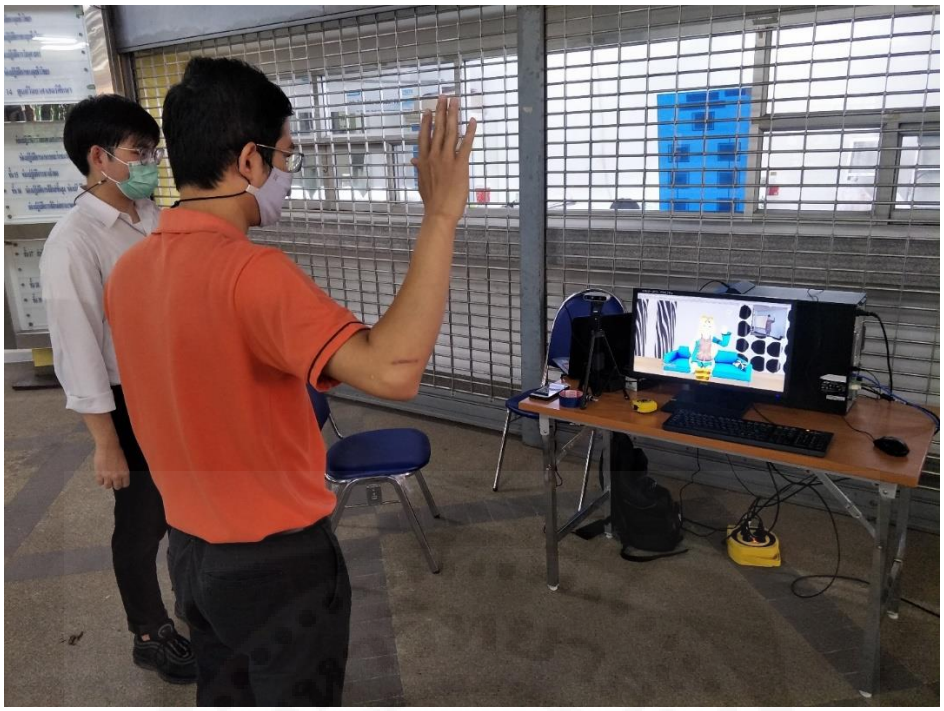
จากการทดลอง Nuitrack SDK มีประสิทธิภาพในการระบุตำแหน่งส่วนต่าง ๆ ของร่างกายได้ 84 เปอร์เซนต์ของข้อต่อทั้งหมด โดยนับเป็น 16 จาก 19 ข้อต่อ ซึ่ง 3 ข้อต่อที่มีความผิดพลาดเกิดจากสถานที่ที่ใช้ในการทำการทดลองคือมือขวา ข้อมือขวา และศอกขวา และมี 4 ข้อต่อที่ไม่ได้ใช้คือข้อต่อเข่าขวา เข่าซ้าย เท้าซ้าย และเท้าขวา เมื่อวัดข้อต่อที่ใช้จริง จะสามารถระบุตำแหน่งได้ 80 เปอร์เซนต์ของท่าทาง โดยนับเป็น 12 จาก 15 ข้อต่อ ดังภาพประกอบที่ 41



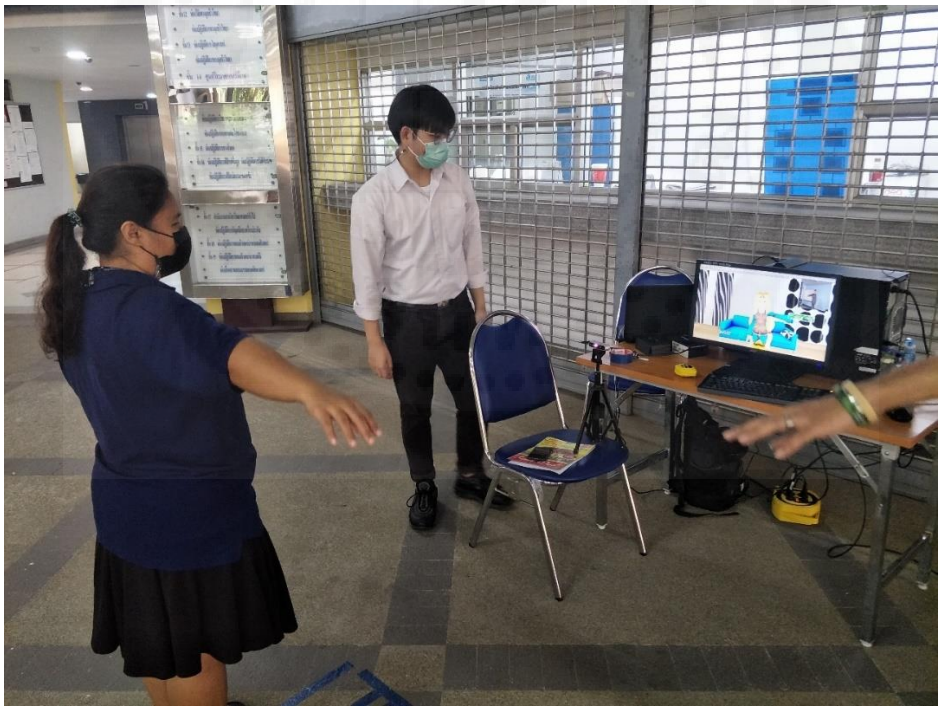
ภาพประกอบที่ 41 แสดงข้อต่อต่าง ๆ ที่ Nuitrack SDK สามารถตรวจจับได้

4.11 ผลการเก็บรวบรวมข้อมูล

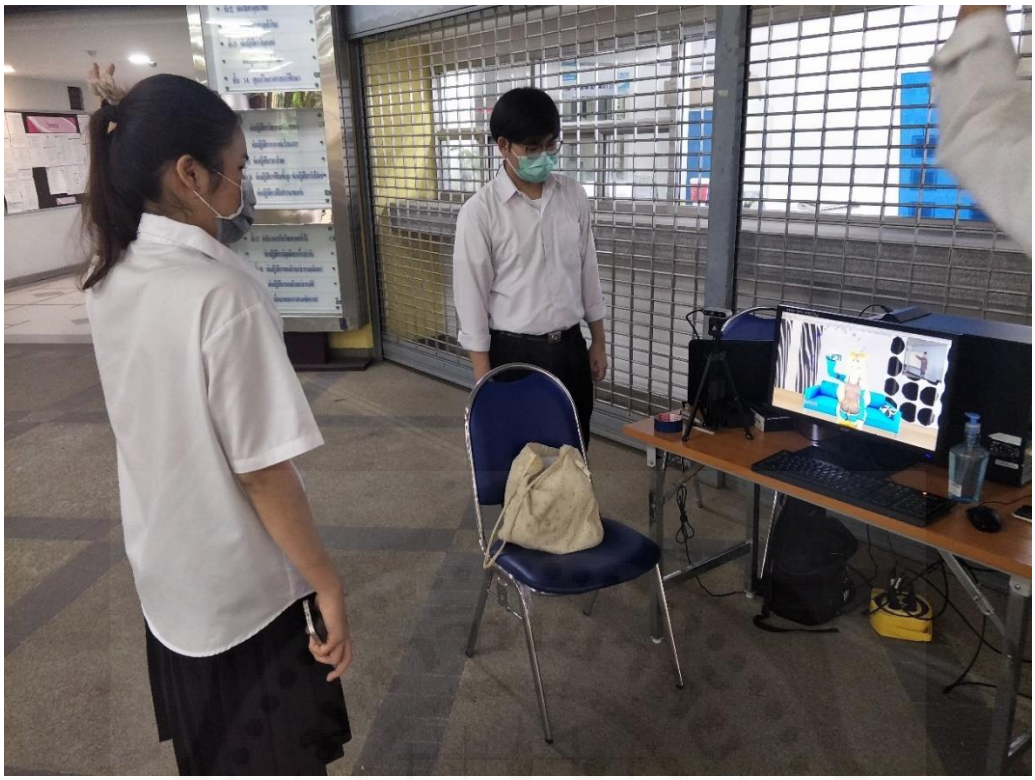
จากการเก็บรวบรวมข้อมูลโดยให้อาสาสมัครเล่นเกมทั้งหมด 24 คน ดังภาพประกอบที่ 42, 43, 44 และ 45 และทำแบบสอบถามได้ผลลัพธ์ความพึงพอใจที่มีต่อเกมคือ ระดับปานกลาง 3 คน ระดับมาก 9 คน และระดับมากที่สุด 13 คน และได้รับแรงกระตุ้นให้อยากออกกำลังกายมากน้อยเพียงใดคือ ระดับน้อย 1 คน ระดับปานกลาง 4 คน ระดับมาก 11 คน ระดับมากที่สุด 8 คน ดังภาพประกอบที่ 46 และ 47



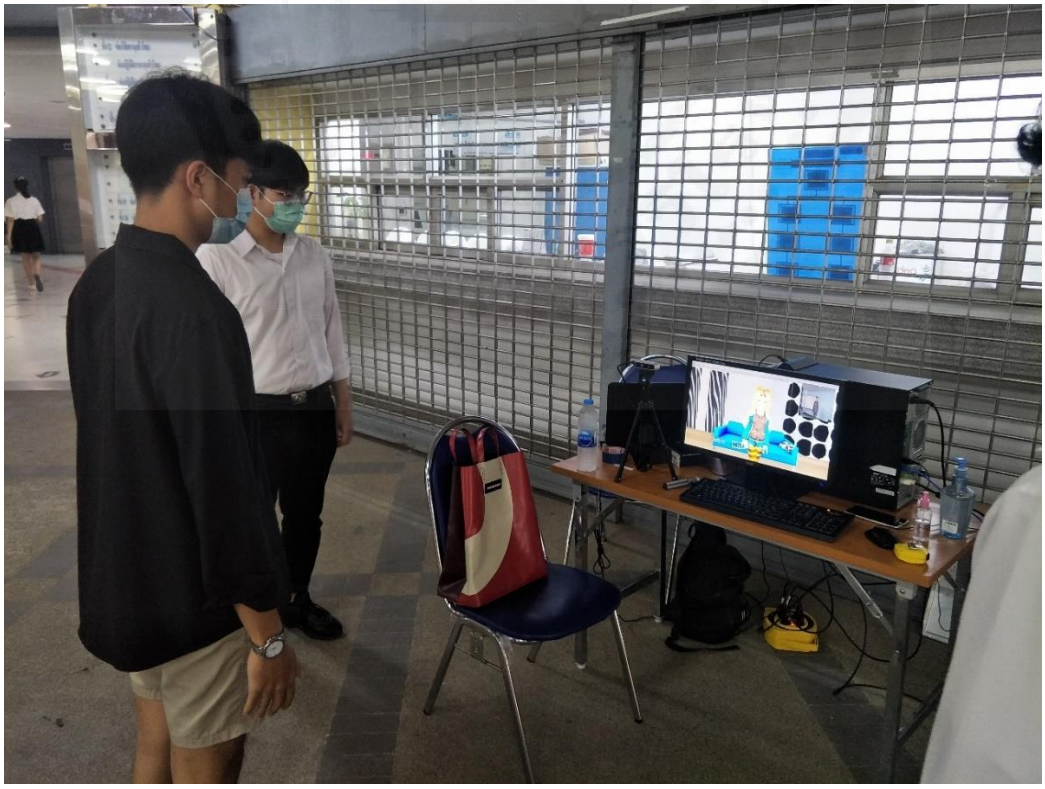
ภาพประกอบที่ 42 หนึ่งในอาสาสมัครกำลังทดลองเล่นเกม



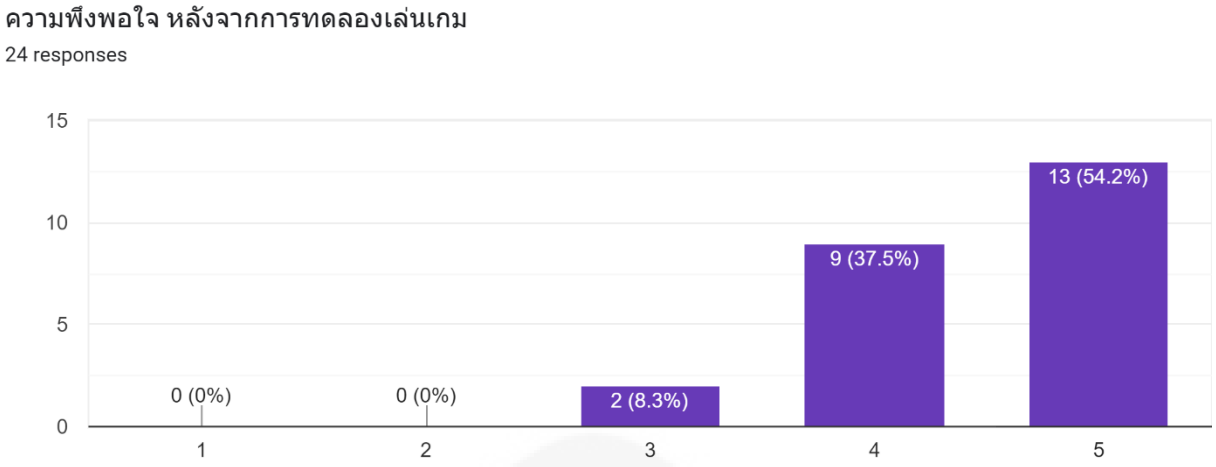
ภาพประกอบที่ 43 หนึ่งในอาสาสมัครกำลังทดลองเล่นเกม



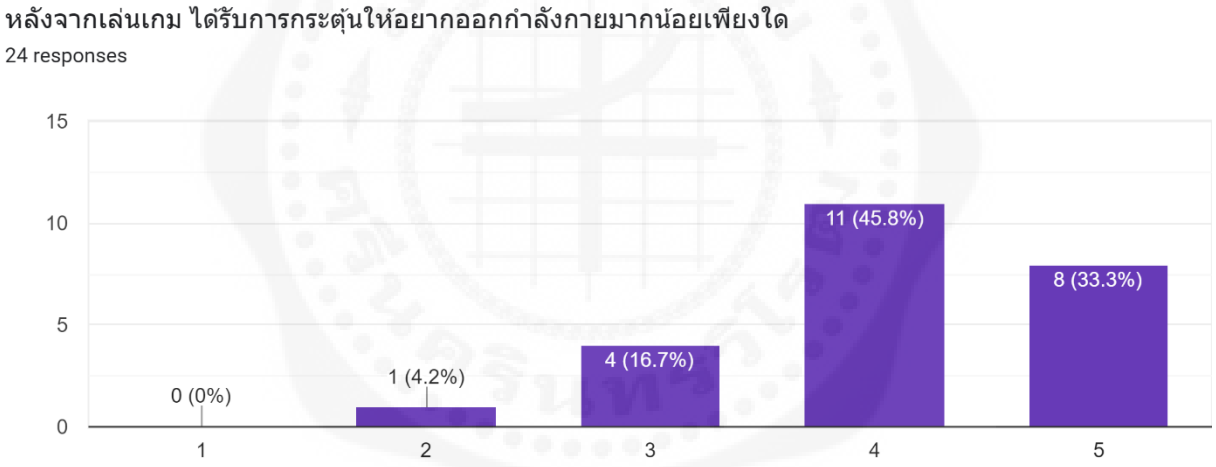
ภาพประกอบที่ 44 หนึ่งในอาสาสมัครกำลังทดลองเล่นเกม



ภาพประกอบที่ 45 หนึ่งในอาสาสมัครกำลังทดลองเล่นเกม



ภาพประกอบที่ 46 กราฟความพึงพอใจที่ได้จากการทดลองเล่นเกม



ภาพประกอบที่ 47 กราฟความกระตุ้นเพื่อให้อยากออกกำลังกายหลังจากได้ทดลองเล่นเกม

บทที่ 5

สรุปผล อภิปรายผล และข้อเสนอแนะ

5.1 สรุปผลการศึกษาค้นคว้า

เนื่องจากสถานการณ์การแพร่ระบาดของ Covid-19 ทำให้สามารถออกกำลังกายได้อย่างจำกัด จึงต้องการพัฒนาเกมสามมิติเพื่อสุขภาพโดยใช้กล้องตรวจจับความลึก Intel RealSense รุ่น D435 คู่กับ Nuitrack SDK เพื่อทำการตรวจจับโครงสร้างและการขยับท่าทางของมนุษย์ จากการดำเนินโครงการ Nuitrack SDK มีประสิทธิภาพในการระบุตำแหน่งส่วนต่าง ๆ ของร่างกายได้ 84 เปอร์เซ็นต์จากสภาพของห้องทดลอง และการทดลองกับอาสาสมัครทั้งหมด 24 คนมีความพึงพอใจในการเล่นเกมระดับมากที่สุด 54.2 เปอร์เซ็นต์ ระดับมาก 37.5 เปอร์เซ็นต์ และระดับปานกลาง 8.3 เปอร์เซ็นต์ และได้รับการกระตุ้นให้อยากออกกำลังกายระดับมากที่สุด 33.3 เปอร์เซ็นต์ ระดับมาก 45.8 เปอร์เซ็นต์ ระดับปานกลาง 16.7 เปอร์เซ็นต์ และระดับน้อย 4.2 เปอร์เซ็นต์

5.2 อภิปรายผล

จากการดำเนินโครงการ Nuitrack SDK สามารถตรวจจับได้ถูกต้องแต่เนื่องด้วยสภาพของห้องทดลองทำให้ไม่สามารถตรวจจับได้อย่างมีประสิทธิภาพทำให้มีความผิดพลาดเกิดขึ้น

ข้อจำกัดของระบบคือเนื่องจากเป็นเกมสามมิติจะต้องวางตำแหน่งและความสูงของกล้อง ระยะห่างของกล้องกับร่างกายของผู้เล่นให้ถูกต้อง ร่างกายของผู้เล่นจะต้องยืนตัวตรงหันหน้าเข้ากล้อง ถ้ายืนเบี้ยวจะทำให้การเล่นเกมไม่มีประสิทธิภาพ สัญลักษณ์ที่แสดงจะอยู่ในรูปแบบสามมิติ ซึ่งการนำมือหรือศอกเอื้อมไปหาจะไม่สามารถโดนได้ถ้าไม่อยู่ในตำแหน่งที่ถูกต้อง ทำให้มีความยากในการเล่นและแสดงให้เห็นว่าจะต้องทำท่าทางให้ถูกต้องถึงจะสามารถผ่านไปท่าต่อไปได้

หลังจากได้ทำการทดลองกับอาสาสมัคร ส่วนสูงของอาสาสมัครทำให้ตำแหน่งของ model มีความคลาดเคลื่อนจึงทำให้ตำแหน่งไม่ตรงกับที่กำหนดไว้

จากแบบสอบถาม สัญลักษณ์ภายในเกมยากต่อการมองเห็นและคาดเดาตำแหน่งทำให้ประสิทธิภาพในการเล่นลดลง

5.3 ข้อเสนอแนะ

ควรจะมีอุปกรณ์ช่วยเพื่อให้สภาพของสถานที่มีความเสถียรเมื่อถูกตรวจจับด้วยกล้องมากขึ้น เช่น พื้นหลังเป็นกระจกกล้องจะไม่สามารถตรวจจับได้จึงต้องนำสิ่งของมาบังไว้

คุณภาพกล้อง และ SDK ที่ดีกว่า Nuitrack จะช่วยเพิ่มความลื่นไหลของการ track ยิ่งขึ้น หากซื้อ license ของ Nuitrack SDK แล้วจะสามารถใช้ได้นานเกิน 3 นาที

ความสูงของแต่ละบุคคลไม่เท่ากันทำให้การเล่นเกมไม่ราบลื่นจึงต้องมีการปรับตัวตามสถานการณ์ขณะทดลอง ซึ่งสามารถแก้ได้โดยการให้กรอกความสูงก่อนเล่นเกมและทำการปรับตำแหน่ง model ภายในเกม

สัญลักษณ์เล็กเกินไปทำให้มองไม่ชัดและคาดเดาตำแหน่งสัญลักษณ์ยาก แก้ไขได้โดยการเพิ่มสัญลักษณ์ต่าง ๆ ที่มือหรือข้อศอกเพื่อให้สามารถรับรู้ได้ว่าเข้าใกล้หรือออกห่างจากสัญลักษณ์

Nuitrack SDK และ Unity มีการอัพเดทอยู่บ่อยครั้งทำให้ควรจะศึกษาข้อมูลอยู่เสมอ

บรรณานุกรม

[1] “Nuitrack: Overview.” http://download.3divi.com/Nuitrack/doc/Overview_page.html (accessed Oct. 19, 2020).

[2] “Nuitrack: Nuitrack SDK Architecture.” http://download.3divi.com/Nuitrack/doc/Architecture_page.html (accessed Oct. 19, 2020).

[3] “Depth Camera D435 – Intel® RealSense™ Depth and Tracking Cameras.” https://www.intelrealsense.com/depth-camera-d435/ (accessed Oct. 19, 2020).

[4] “Projectors for D400 Series Depth Cameras.” https://dev.intelrealsense.com/docs/projectors (accessed Oct. 19, 2020).

[5] “Introduction — Blender Manual.” https://docs.blender.org/manual/en/latest/getting_started/about/introduction.html (accessed Oct. 19, 2020).

[6] “Unity - Manual: Input.” https://docs.unity3d.com/Manual/Input.html (accessed Oct. 19, 2020).

[7] “Nuitrack Skeleton Tracking | Packs | Unity Asset Store.” https://assetstore.unity.com/packages/templates/packs/nuitrack-skeleton-tracking-127675 (accessed Oct. 19, 2020).

[8] “Rhythm game - Wikipedia.” https://en.wikipedia.org/wiki/Rhythm_game (accessed Oct. 19, 2020).

[9] “การกายภาพร่างกาย. กายภาพบำบัด (Physical Therapy) คือ... | by samzun | Medium.” https://medium.com/@sclub_tum/การกายภาพร่างกาย-8972990c5f91 (accessed Apr. 16, 2021).

[10] R. A. Pambudi, N. Ramadijanti, and A. Basuki, “Psychomotor game learning using skeletal tracking method with leap motion technology,” in *Proceedings - 2016 International Electronics Symposium, IES 2016*, Feb. 2017, pp. 142–147, doi: 10.1109/ELECSYM.2016.7860991.

[11] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields *.” Accessed: Oct. 19, 2020. [Online]. Available: https://youtu.be/pW6nZXeWlGM.

[12] J. Y. Chen, C. H. Liu, C. H. Hsieh, S. Y. Huang, W. K. Wang, and B. H. Nien, "Kinect augmented reality gear game design," in *Proceedings of the 2017 IEEE International Conference on Applied System Innovation: Applied System Innovation for Modern Technology, ICASI 2017*, Jul. 2017, pp. 373–375, doi: 10.1109/ICASI.2017.7988429.

[13] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with Microsoft Kinect sensor: A review," *IEEE Trans. Cybern.*, vol. 43, no. 5, pp. 1318–1334, Oct. 2013, doi: 10.1109/TCYB.2013.2265378.

[14] M. Tarabini *et al.*, "Monitoring the human posture in industrial environment: A feasibility study," in *2018 IEEE Sensors Applications Symposium, SAS 2018 - Proceedings*, Apr. 2018, vol. 2018-January, pp. 1–6, doi: 10.1109/SAS.2018.8336710.

ภาคผนวก

CameraPreCulling.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CameraPreCulling : MonoBehaviour
{
    Camera cam;

    void Start()
    {
        cam = GetComponent<Camera>();
    }

    void OnPreCull()
    {
        cam.ResetWorldToCameraMatrix();
        cam.ResetProjectionMatrix();
        cam.projectionMatrix = cam.projectionMatrix * Matrix4x4.Scale(new Vector3(-1, 1, 1));
    }

    // Set it to true so we can watch the flipped Objects
    void OnPreRender()
    {
        GL.invertCulling = true;
    }

    // Set it to false again because we dont want to affect all other cammeras.
```

```
    void OnPostRender()
    {
        GL.invertCulling = false;
    }
}
```

MenuScript.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class MenuScript : MonoBehaviour
{
    public void LoadScene(string scenename)
    {
        SceneManager.LoadScene(scenename);
    }

    public void ExitGame() {
        Application.Quit();
    }
}
```

beatscroller.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class beatscroller : MonoBehaviour
```

```
{
    public float beatTempo;
    public bool hasStarted;

    // Start is called before the first frame update
    void Start()
    {
        beatTempo = beatTempo / 60f;
    }

    // Update is called once per frame
    void Update()
    {
        if (!hasStarted)
        {
            // if (Input.anyKeyDown)
            // {
            //     hasStarted = true;
            // }
        }
        else
        {
            transform.position -= new Vector3(0f, 0f,beatTempo * Time.deltaTime );
        }
    }
}
```

beatscrollerLeft.cs

```
using System.Collections;
using System.Collections.Generic;
```

```
using UnityEngine;

public class beatscrollerLeft : MonoBehaviour
{
    public float beatTempo;
    public bool hasStarted;

    // Start is called before the first frame update
    void Start()
    {
        beatTempo = beatTempo / 60f;
    }

    // Update is called once per frame
    void Update()
    {
        if (!hasStarted)
        {
            // if (Input.anyKeyDown)
            // {
            //     hasStarted = true;
            // }
        }
        else
        {
            transform.position += new Vector3(beatTempo * Time.deltaTime, 0f,0f );
        }
    }
}
```

beatscrollerRight.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class beatscrollerRight : MonoBehaviour
{
    public float beatTempo;
    public bool hasStarted;

    // Start is called before the first frame update
    void Start()
    {
        beatTempo = beatTempo / 60f;
    }

    // Update is called once per frame
    void Update()
    {
        if (!hasStarted)
        {
            // if (Input.anyKeyDown)
            // {
            //     hasStarted = true;
            // }
        }
        else
        {
            transform.position -= new Vector3(beatTempo * Time.deltaTime, 0f,0f );
        }
```

```
    }
}
```

RiggedAvatar.cs

```
using UnityEngine;
using System.Collections.Generic;

public class RiggedAvatar : MonoBehaviour
{
    [Header("Rigged model")]
    [SerializeField]
    ModelJoint[] modelJoints;
    [SerializeField]
    nuitrack.JointType rootJoint = nuitrack.JointType.Torso;
    /// <summary> Model bones </summary>
    Dictionary<nuitrack.JointType, ModelJoint> jointsRigged = new
Dictionary<nuitrack.JointType, ModelJoint>();

    void Start()
    {
        for (int i = 0; i < modelJoints.Length; i++)
        {
            modelJoints[i].baseRotOffset = modelJoints[i].bone.rotation;
            jointsRigged.Add(modelJoints[i].jointType.TryGetMirrored(), modelJoints[i]);
        }
    }

    void Update()
    {
        if (CurrentUserTracker.CurrentSkeleton != null)
ProcessSkeleton(CurrentUserTracker.CurrentSkeleton);
```

```
    }

    void ProcessSkeleton(nuitrack.Skeleton skeleton)
    {
        //Calculate the model position: take the root position and invert movement along the
Z axis
        Vector3 rootPos = Quaternion.Euler(0f, 180f, 0f) * (0.01f *
skeleton.GetJoint(rootJoint).ToVector3());
        transform.position = rootPos;

        foreach (var riggedJoint in jointsRigged)
        {
            //Get joint from the Nuitrack
            nuitrack.Joint joint = skeleton.GetJoint(riggedJoint.Key);

            ModelJoint modelJoint = riggedJoint.Value;

            //Calculate the model bone rotation: take the mirrored joint orientation, add a basic
rotation of the model bone, invert movement along the Z axis
            Quaternion jointOrient = Quaternion.Inverse(CalibrationInfo.SensorOrientation) *
(joint.ToQuaternionMirrored()) * modelJoint.baseRotOffset;

            modelJoint.bone.rotation = jointOrient;
        }
    }
}
```

NoteObject.cs

```
using System.Collections;
using System.Collections.Generic;
using System.IO;
```

```
using UnityEngine;
public class NoteObject : MonoBehaviour
{
    public bool canBeHit;
    public GameObject hitEffect, goodEffect, perfectEffect, missEffect;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        if (canBeHit)
        {
            gameObject.SetActive(false);
            GameManager.instance.NoteHit();
        }
    }

    //after this is me mario mother fucker
    private void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Activator")
        {
            canBeHit = true;
        }
        else if(other.tag == "Hit")
```

```
        {
            Debug.Log("Hit");
            GameManager.instance.NormalHit();
            Instantiate(hitEffect, transform.position, hitEffect.transform.rotation);
            gameObject.SetActive(false);
        }
        else if(other.tag == "Good")
        {
            Debug.Log("Good");
            GameManager.instance.GoodHit();
            Instantiate(goodEffect, transform.position, goodEffect.transform.rotation);
            gameObject.SetActive(false);
        }
        else if(other.tag == "Perfect")
        {
            Debug.Log("Perfect");
            GameManager.instance.PerfectHit();
            Instantiate(perfectEffect, transform.position, perfectEffect.transform.rotation);
            gameObject.SetActive(false);
        }
        else if(other.tag == "Missed")
        {
            canBeHit = false;
            Debug.Log("Missed");
            GameManager.instance.NoteMissed();
            Instantiate(missEffect, transform.position, missEffect.transform.rotation);
            gameObject.SetActive(false);
        }
    }
}
```

GameManager,cs

```
using System.Collections;
using System.Collections.Generic;
using System.Globalization;
using UnityEngine;
using UnityEngine.UI;

public class GameManager : MonoBehaviour
{
    public bool isStartPlaying;
    public AudioSource theMusic;
    public beatscroller theBS;
    public static GameManager instance;

    public int currentScore;
    public int scorePerNote = 100;
    public int scorePerGoodNote = 125;
    public int scorePerPerfectNote = 150;

    public int currentMultiplier;
    public int multiplierTracker;
    public int[] multiplierThresholds;

    public Text scoreText;
    public Text multiText;

    public float totalNotes;
    public float normalHits;
    public float goodHits;
    public float perfectHits;
```

```
    public float missedHits;

    public GameObject resultsScreen;
    public Text percentHitText, normalsText, goodText, perfectText, missesText, rankText,
finalScoreText;

    //Start is called before the first frame update
    void Start()
    {
        instance = this;
        scoreText.text = "Score: 0";
        currentMultiplier = 1;
        totalNotes = FindObjectsOfType<NoteObject>().Length;
    }

    // Update is called once per frame
    void Update()
    {
        if (!isStartPlaying)
        {
            isStartPlaying = true;
            theBS.hasStarted = true;
            theMusic.Play();
        }
        else
        {
            if(!theMusic.isPlaying && !resultsScreen.activeInHierarchy)
            {
                resultsScreen.SetActive(true);
                normalsText.text = "" + normalHits;
```

```
goodText.text = goodHits.ToString();
perfectText.text = perfectHits.ToString();
missesText.text = "" + missedHits;
float totalHit = normalHits + goodHits + perfectHits;
float percentHit = (totalHit / totalNotes) * 100f;
percentHitText.text = percentHit.ToString("F1") + "%";
string rankVal = "F";
if (percentHit > 50)
{
    rankVal = "D";
    if (percentHit > 60)
    {
        rankVal = "C";
        if (percentHit > 70)
        {
            rankVal = "B";
            if (percentHit > 80)
            {
                rankVal = "A";
                if (percentHit > 90)
                {
                    rankVal = "S";
                }
            }
        }
    }
}

rankText.text = rankVal;
```

```
            finalScoreText.text = currentScore.ToString();
        }
    }
}

public void NoteHit()
{
    Debug.Log("Hit On Time");
    if (currentMultiplier - 1 < multiplierThresholds.Length)
    {
        multiplierTracker++;
        if (multiplierThresholds[currentMultiplier - 1] <= multiplierTracker)
        {
            multiplierTracker = 0;
            currentMultiplier++;
        }
    }

    multiText.text = "Multipiler: x" + currentMultiplier;
    currentScore += scorePerNote * currentMultiplier;
    scoreText.text = "Score: " + currentScore;
}

 public void NormalHit()
 {
     currentScore += scorePerNote * currentMultiplier;
     NoteHit();
     normalHits++;
 }
```

```
    public void GoodHit()
    {
        currentScore += scorePerGoodNote * currentMultiplier;
        NoteHit();
        goodHits++;
    }

    public void PerfectHit()
    {
        currentScore += scorePerPerfectNote * currentMultiplier;
        NoteHit();
       perfectHits++;
    }

   public void NoteMissed()
   {
       Debug.Log("Missed Note");
       currentMultiplier = 1;
       multiplierTracker = 0;
       multiText.text = "Multipiler: x" + currentMultiplier;
       missedHits++;
   }
}
```

GameManagerScene3.cs

```
using System.Collections;
using System.Collections.Generic;
using System.Globalization;
using UnityEngine;
using UnityEngine.UI;
```

```
public class GameManagerScene3 : MonoBehaviour
{

    public int totalScore = 1;

    public bool isStartPlaying;
    public AudioSource theMusic;
    public static GameManagerScene3 instance;

    public int currentScore;
    public int scorePerNote = 1;

    public Text scoreText;

    public float totalNotes;
    public float normalHits;

    //Start is called before the first frame update
    void Start()
    {
        instance = this;
        scoreText.text = "Score: 0";
        totalNotes = FindObjectsOfType<NoteObject>().Length;
    }

    // Update is called once per frame
    void Update()
    {
        if (!isStartPlaying)
        {
```

```
            isStartPlaying = true;
            theMusic.Play();
        }
    }

    public void NoteHit()
    {
        currentScore += scorePerNote /* * currentMultiplier*/;
        scoreText.text = "Score: " + currentScore ;
    }

    public void NormalHit()
    {
        NoteHit();
        normalHits++;
    }
}
```

Set.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Set : MonoBehaviour
{
    public bool canBeHit;
    public bool isPerfect;
    public GameObject hitEffect, perfectEffect;
    public AudioSource hits;
```

```
// Start is called before the first frame update
void Start()
{

}

// Update is called once per frame
void Update()
{
    if (canBeHit)
    {
        gameObject.SetActive(false);
        //GameManagerScene3.instance.NoteHit();
    }

}
private void OnTriggerEnter(Collider other)
{
    if (gameObject.tag == "LHN" && other.tag == "LH")
    {
        canBeHit = true;
        Debug.Log("Hit");
        // GameManagerScene3.instance.NormalHit();
        if(isPerfect)
        {
            isPerfect = false;
            Instantiate(perfectEffect, transform.position, hitEffect.transform.rotation);
        }
        else
        {
```

```
            Instantiate(hitEffect, transform.position, hitEffect.transform.rotation);
        }


        gameObject.SetActive(false);
        hits.Play();
    }
    else if (gameObject.tag == "RHN" && other.tag == "RH" )
    {
        canBeHit = true;
        Debug.Log("Hit");
        // GameManagerScene3.instance.NormalHit();
        if(isPerfect)
        {
            isPerfect = false;
            Instantiate(perfectEffect, transform.position, hitEffect.transform.rotation);
        }
        else
        {
            Instantiate(hitEffect, transform.position, hitEffect.transform.rotation);
        }
        gameObject.SetActive(false);
        hits.Play();
    }
    else if (gameObject.tag == "REN" && other.tag == "RE" )
    {
        canBeHit = true;
        Debug.Log("Hit");
        //GameManagerScene3.instance.NormalHit();
        if(isPerfect)
        {
```

```
            isPerfect = false;
            Instantiate(perfectEffect, transform.position, hitEffect.transform.rotation);
        }
        else
        {
            Instantiate(hitEffect, transform.position, hitEffect.transform.rotation);
        }
        gameObject.SetActive(false);
        hits.Play();
    }
    else if (gameObject.tag == "LEN" && other.tag == "LE" )
    {
        canBeHit = true;
        Debug.Log("Hit");
        // GameManagerScene3.instance.NormalHit();
        if(isPerfect)
        {
            isPerfect = false;
            Instantiate(perfectEffect, transform.position, hitEffect.transform.rotation);
        }
        else
        {
            Instantiate(hitEffect, transform.position, hitEffect.transform.rotation);
        }
        gameObject.SetActive(false);
        hits.Play();
    }
  }
}
```

Order.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Order : MonoBehaviour
{
    public Set set1, set2, set3, set4, set5 ;
    public int poseTracking;
    public int number = 1, loop, checkloop;

    // Start is called before the first frame update
    void Start()
    {



    }



    // Update is called once per frame
    void Update()
    {
        if(number == 1)
        {
            if(poseTracking == 1)
            {
                if (set1.canBeHit)
                {
                    Second();
                }
```

```
            if (set2.canBeHit)
            {
                set2.canBeHit= false;
                GameManagerScene3.instance.NormalHit();
                if(checkloop == loop)
                {
                    poseTracking = 2;
                }
                else
                {
                    checkloop++;
                    First();
                }
            }
        }
    }else if(number == 2)
    {
        if(poseTracking == 1)
        {
            if (set1.canBeHit)
            {
                Second();
            }
            if (set2.canBeHit)
            {
                Third();
            }
            if (set3.canBeHit)
            {
                Forth();
```

```
            }
            if (set4.canBeHit)
            {
                Fifhth();
            }
            if(set5.canBeHit)
            {
                set5.canBeHit= false;
                GameManagerScene3.instance.NormalHit();
                if(checkloop == loop)
                {
                    poseTracking = 2;
                }
                else
                {
                    checkloop++;
                    First();
                }
            }
        }
    }
}

public void First()
{
    poseTracking = 1;
    set1.gameObject.SetActive(true);
}

public void Second()
```

```
{
    if(number == 1)
    {
        set2.gameObject.SetActive(true);
        set2.isPerfect = true;
        set1.canBeHit = false;
        Debug.Log("set 1");
    }
    if(number == 2)
    {
        set2.gameObject.SetActive(true);
        set1.canBeHit = false;
        Debug.Log("set 2");
    }
}

public void Third()
{
    if(number == 2)
    {
        set3.gameObject.SetActive(true);
        set3.isPerfect = false;
        set2.canBeHit = false;
    }
}

public void Forth()
{
    if(number == 2)
    {
```

```
            set4.gameObject.SetActive(true);
            set3.canBeHit = false;
        }
    }

    public void Fifhth()
    {
        if(number == 2)
        {
            set5.gameObject.SetActive(true);
            set5.isPerfect = true;
            set4.canBeHit = false;
        }
    }
}
```

OrderPose.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.IO;

public class OrderPose : MonoBehaviour
{
    public Order childs; //store Order
    public GameObject Mother; //Store OrderPose
    public GameObject Father; //Store videoPlaylist
    public GameManagerScene3 total; //for get and set score in GameManagerScene3
    public GameObject video; // Store video
```

```
public GameObject resultsScreen; //Store Canvas resultscreen
public int count= 0;
public Text score;
public string inp_ln;
public string directory;
public int numlines = 6;
public int rotate = 0;

// Start is called before the first frame update
void Start()
{

    directory = Application.dataPath + "//reads.txt";
    //write new file "reads" if file doesn't exist
    if(!File.Exists(directory)){
        StreamWriter sw = new StreamWriter(directory);
        Debug.Log(sw);
        for(int i = 1 ;i <= numlines;i++){
            sw.WriteLine("pose"+ i +" loop=3");
        }
        sw.Close();

    }
    Debug.Log(directory);

    readTextFile(directory);
    //loop every child in Mother(OrderPose) Hierarchy
    foreach(Transform child in Mother.transform )
    {
        Order Tchild = child.GetComponent(typeof(Order)) as Order;
```

```
        count = count + Tchild.loop;
    }
    total.totalScore= count;
    //show Debug
    Debug.Log(count);
    rotation();
    childs.First();
    videoP();
    video.SetActive(true);
}

// Update is called once per frame
void Update()
{
    if(childs.poseTracking == 2){
        childs.poseTracking = 3;
        video.SetActive(false);
        rotate ++;
        rotation();
        videoP();
        video.SetActive(true);
        childs.First();
        //show Debug
        Debug.Log("Hello " + childs);
        if(rotate == 6)
        {
            video.SetActive(false);
            score.text = total.currentScore.ToString() + " / " + total.totalScore;
            resultsScreen.SetActive(true);
        }
```

```
    }
}
//change child(Order) to next child in hierarchy
public void rotation(){
    if(rotate <= 5){
        Transform child = Mother.transform.GetChild(rotate);
        childs = child.GetComponent(typeof(Order)) as Order;
        //show Debug
        Debug.Log("rotation time" + rotate);
    }
}
//change video(GameObject) to next child in hierarchy
public void videoP(){
    if(rotate <= 5){
        Transform child = Father.transform.GetChild(rotate);
        video = child.gameObject;
        //show Debug
        Debug.Log("video time" + rotate);
    }
}

//read text file "reads"
void readTextFile(string file_path)
{
    StreamReader inp_stm = new StreamReader(file_path);
    if(!inp_stm.EndOfStream)
    {
        foreach(Transform child in this.transform )
        {
            Order Tchild = child.GetComponent(typeof(Order)) as Order;
```

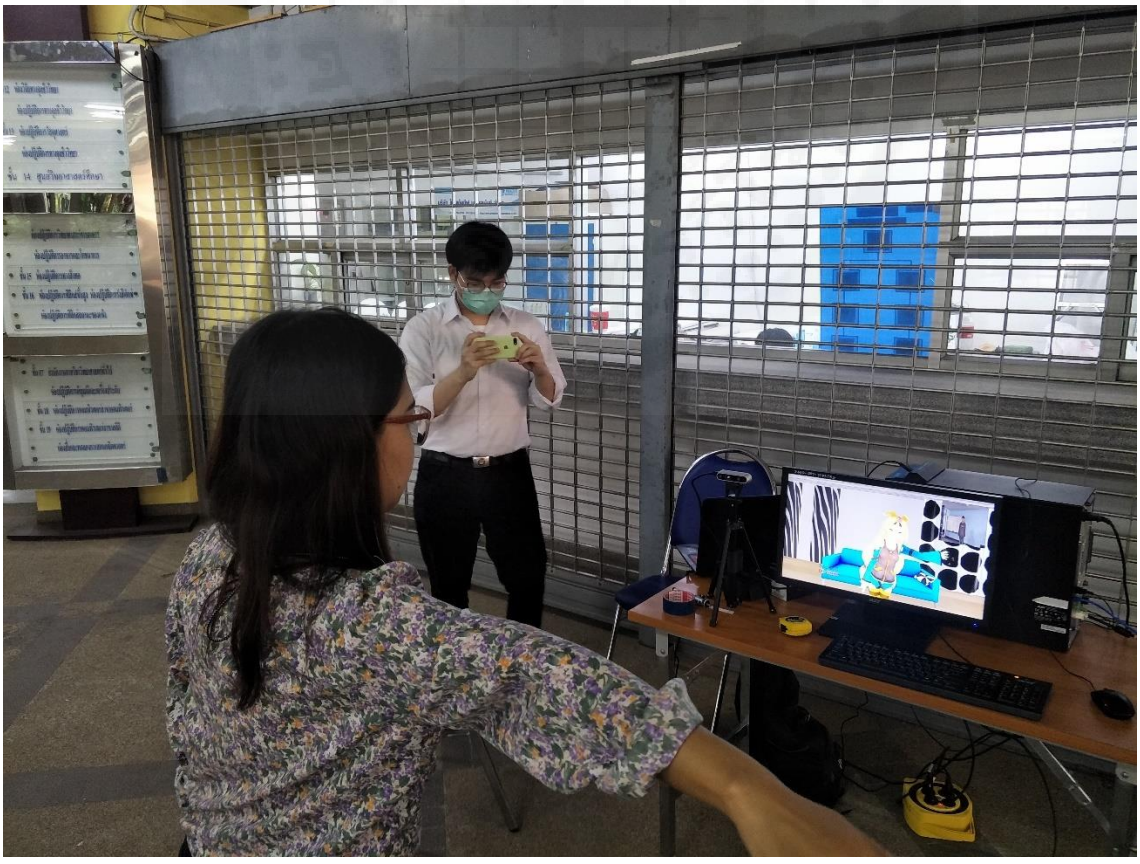
```
            inp_ln = inp_stm.ReadLine(); //อ่าน 1 บรรทัด
            Tchild.loop = int.Parse(inp_ln.Substring(inp_ln.Length - 1));

        }
    }
    inp_stm.Close();
  }
}
```

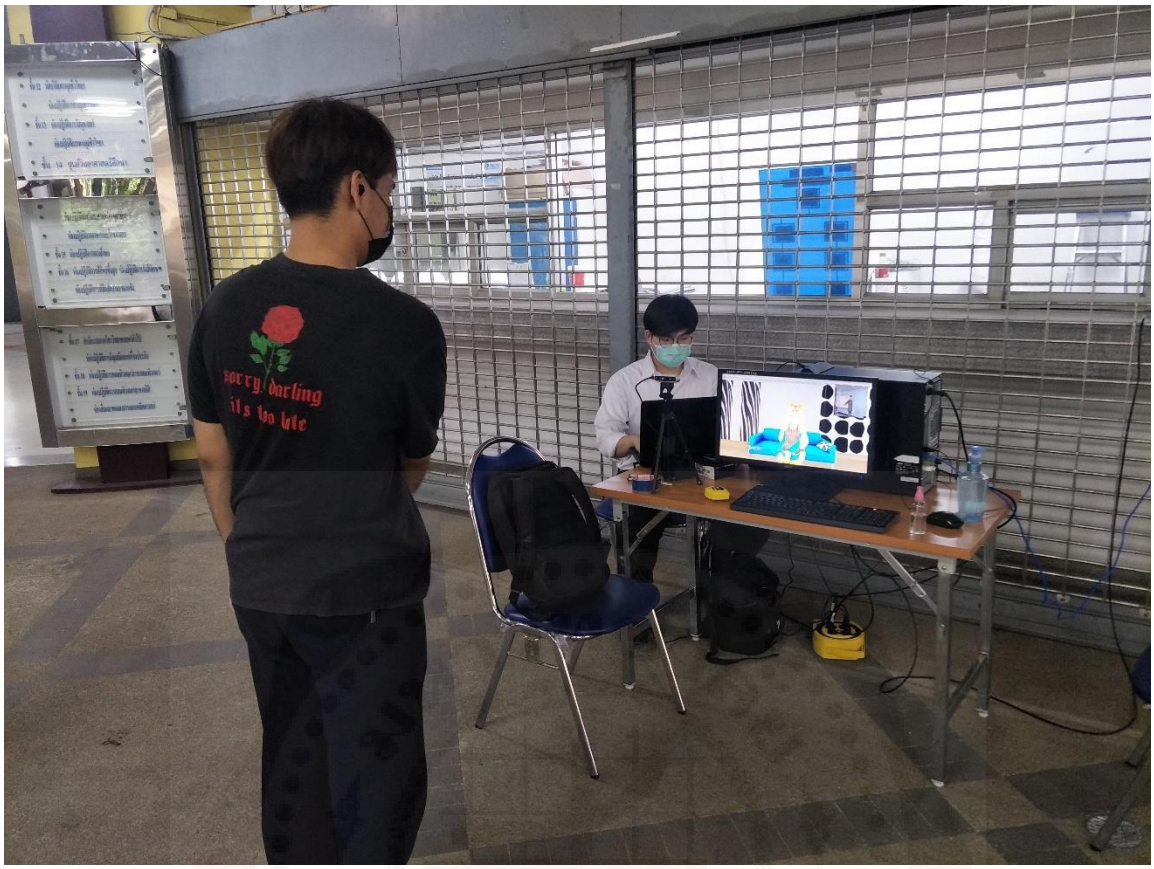
UITimer.cs

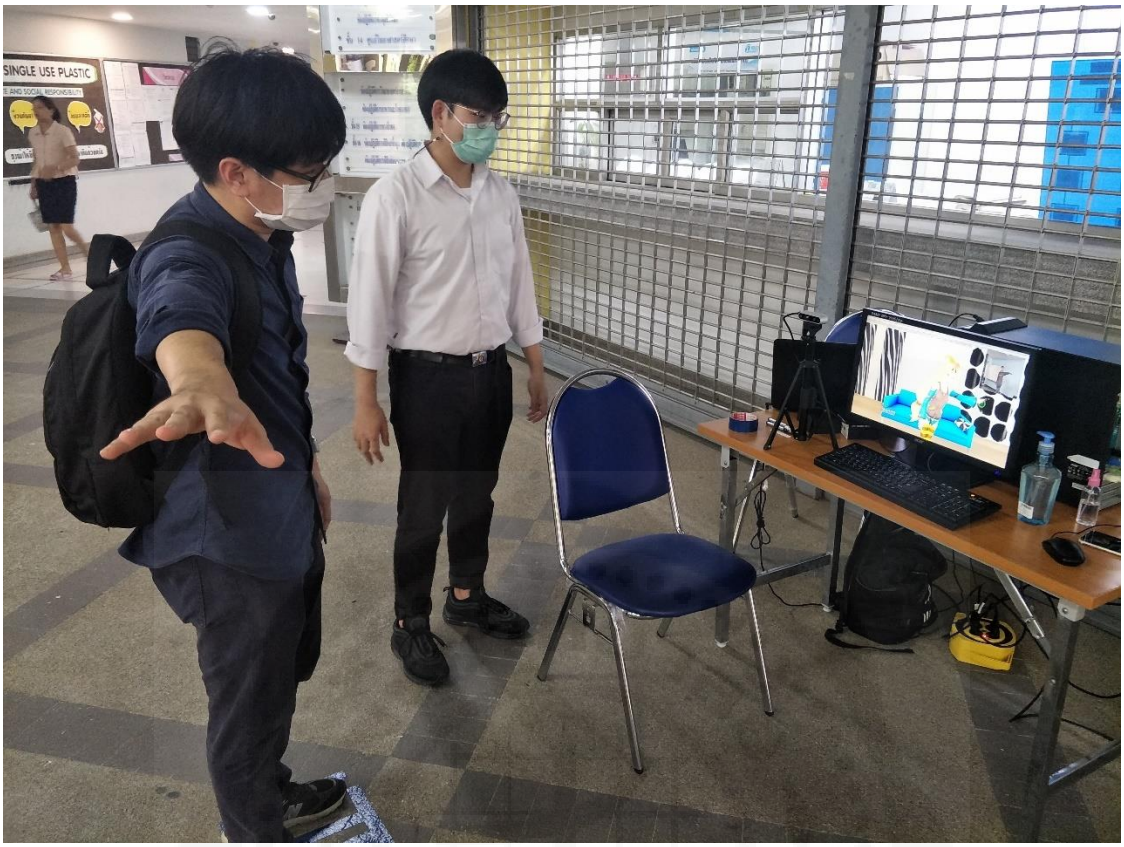
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class UITimer : MonoBehaviour
{
 public Text TimerText;
 public bool playing;
 private float Timer;
 public GameObject resultsScreen;
 public GameManagerScene3 inputScore;
 public Text score ;

 void Update () {

        if(playing == true){

          Timer += Time.deltaTime;
    //Debug.Log(Timer);
```

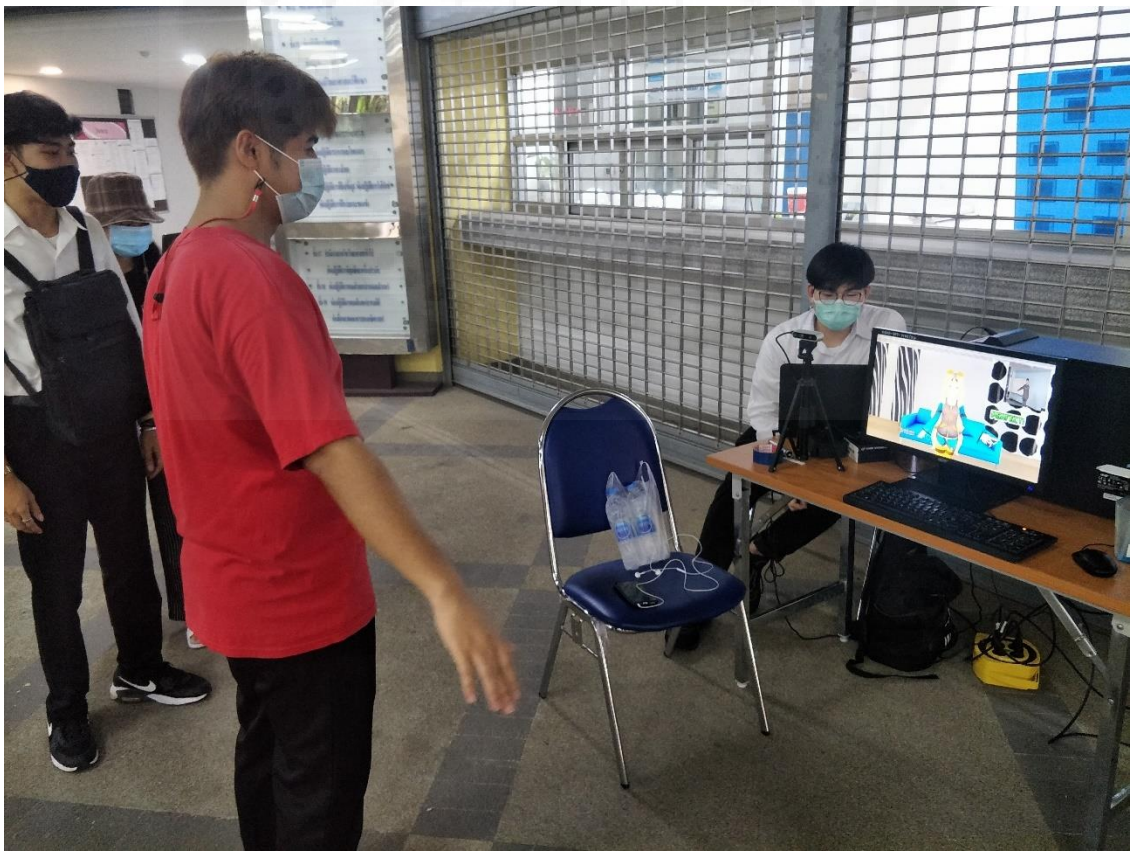
```
            int minutes = Mathf.FloorToInt(Timer / 60F);
            int seconds = Mathf.FloorToInt(Timer % 60F);
            int milliseconds = Mathf.FloorToInt((Timer * 100F) % 100F);
            TimerText.text = minutes.ToString ("00") + ":" + seconds.ToString ("00") + ":" +
milliseconds.ToString("00");
        }
    if(Timer > 170.0F && !resultsScreen.activeInHierarchy){

        score.text = inputScore.currentScore.ToString() + " / " + inputScore.totalScore;
        resultsScreen.SetActive(true);
        TimerText.gameObject.SetActive(false);
        // Time.timeScale=0;
    }
  }
}
```

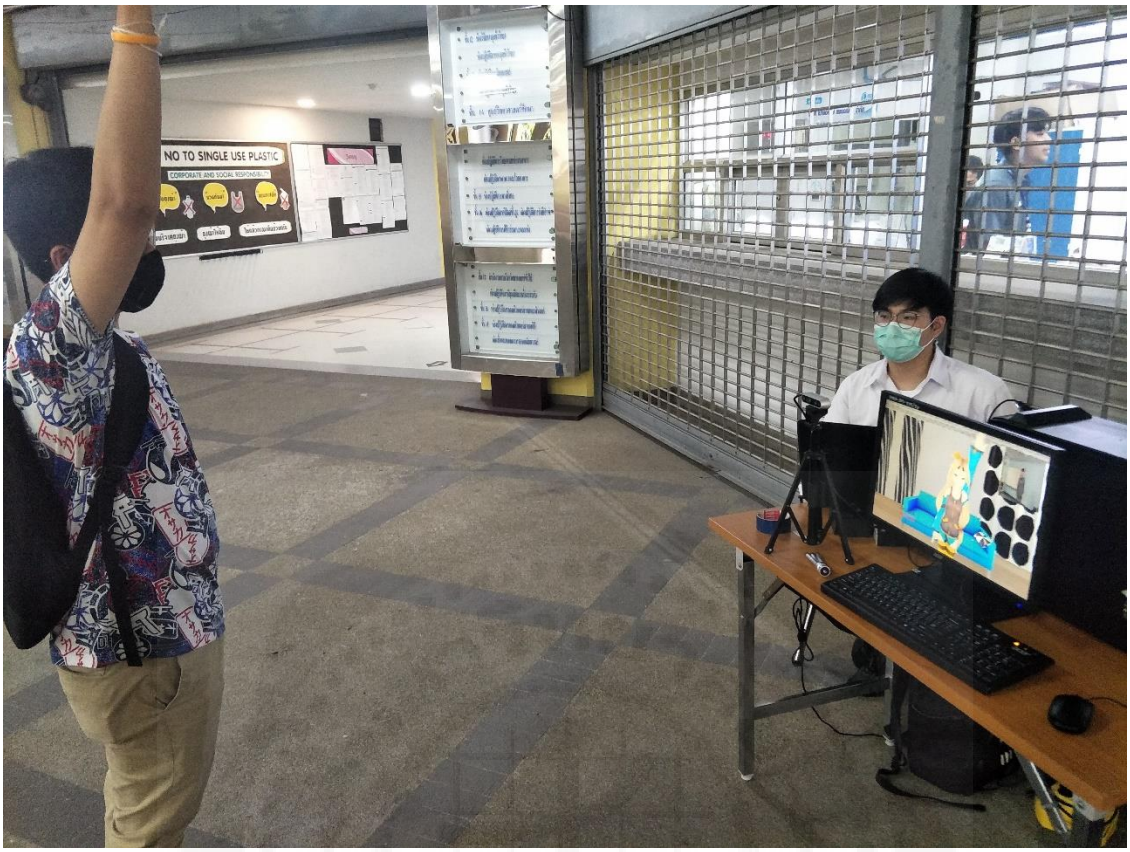
ภาพผลการทดลองในอาสาสมัคร




sorry, darling
it's too late


SINGLE USE PLASTIC




NO TO SINGLE USE PLASTIC