รายงานการวิจัย
ระบบไร้สายสำหรับการสร้างและปรับปรุงข้อความบนไฟล์เอกสาร
Wireless System for Text Creation and Modification for Document File

หัวหน้าโครงการวิจัย
นางพนารัตน์ เชิญถนอมวงศ์

ผู้วิจัยร่วม
นายสุรเชษฐ์ สุวรรณประทีป
นายอนิรุทธิ์ ธีระบุตร

ได้รับทุนสนับสนุนงานวิจัยจากเงินรายได้ ประจำปีงบประมาณ 2556
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

รายงานการวิจัย
ระบบไร้สายสำหรับการสร้างและปรับปรุงข้อความบนไฟล์เอกสาร
Wireless System for Text Creation and Modification for Document File

หัวหน้าโครงการวิจัย
นางพนารัตน์ เชิญถนอมวงศ์

ผู้วิจัยร่วม
นายสุรเชษฐ์ สุวรรณประทีป
นายอนิรุทธิ์ ธีระบุตร

ได้รับทุนสนับสนุนงานวิจัยจากเงินรายได้ ประจำปีงบประมาณ 2556
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

กิตติกรรมประกาศ

ผู้วิจัยขอมอบความสำเร็จนี้ให้แก่บิดา มารดา อันเป็นที่รักยิ่ง ผู้ซึ่งคอยให้กำลังใจและสนับสนุนผู้จัยในทุกเรื่อง ขอขอบพระคุณครูบาอาจารย์ทุกท่าน ที่ได้มอบความรู้ คำปรึกษา แนะนำ รวมทั้งกำลังใจให้กับผู้วิจัย

ขอขอบคุณ คุณธนพงษ์ ชื่นอุระจิตร และสมาชิก Wireless Information Network (WIN) Lab ทุกคนให้ความช่วยเหลือจนงานวิจัยสำเร็จสมบูรณ์

ขอขอบคุณสาขาวิชาวิศวกรรมคอมพิวเตอร์ หลักสูตรวิศวกรรมสารสนเทศ ที่ได้เอื้อเฟื้ออุปกรณ์และสถานที่ ในการทำวิจัย

พนารัตน์ เชิญถนอมวงศ์
สุรเชษฐ์ สุวรรณประทีป
อนิรุทธิ์ ธีระบุตร

ชื่อโครงการ (ภาษาไทย) ระบบไร้สายสำหรับการสร้างและปรับปรุงข้อความบนไฟล์เอกสาร

ชื่อโครงการ(ภาษาอังกฤษ) Wireless System for Text Creation and Modification for Document File

แหล่งเงิน คณะวิศวกรรมศาสตร์

ประจำปีงบประมาณ 2556 จำนวนเงินที่ได้รับการสนับสนุน 80,000 บาท

ระยะเวลาทำการวิจัย 1 ปี ตั้งแต่ 1 ตุลาคม 2555 ถึง 30 กันยายน 2556

ชื่อ-สกุล หัวหน้าโครงการ และผู้ร่วมโครงการวิจัย พร้อมระบุ หน่วยงานต้นสังกัดและ อีเมล์

<u>หัวหน้าโครงการวิจัย</u>

ชื่อ-สกุล (ภาษาไทย) ดร. พนารัตน์ เชิญถนอมวงศ์

ชื่อ-สกุล (ภาษาอังกฤษ) Dr. Panarat Cherntanomwong

ตำแหน่งทางวิชาการ อาจารย์ สาขา วิศวกรรมคอมพิวเตอร์ (หลักสูตรวิศวกรรมสารสนเทศ)

สัดส่วนการวิจัย 50%

คณะ วิศวกรรมศาสตร์

E-mail krpanara@kmitl.ac.th

<u>ผู้ร่วมวิจัย</u>

ชื่อ-สกุล (ภาษาไทย) นายสุรเชษฐ์ สุวรรณประทีป

ชื่อ-สกุล (ภาษาอังกฤษ) Mr. Surachet Suwanprathip

ตำแหน่งทางวิชาการ .....-.....สัดส่วนการวิจัย 25%

ภาควิชา วิศวกรรมคอมพิวเตอร์ (หลักสูตรวิศวกรรมสารสนเทศ) คณะ วิศวกรรมศาสตร์

โทรศัพท์ 02-3298000 ต่อ 3449 โทรสาร 02-3298327

E-mail kmitpop@gmail.com

ชื่อ-สกุล (ภาษาไทย) นายอนิรุทธิ์ ธีระบุตร

ชื่อ-สกุล (ภาษาอังกฤษ) Mr. Anirut Teerabut

ตำแหน่งทางวิชาการ .....-.....สัดส่วนการวิจัย 25%

ภาควิชา วิศวกรรมคอมพิวเตอร์ (หลักสูตรวิศวกรรมสารสนเทศ) คณะ วิศวกรรมศาสตร์

โทรศัพท์ 02-3298000 ต่อ 3449 โทรสาร 02-3298327

E-mail Anirut.311@gmail.com

คำสำคัญ (Keywords) การนำเสนอ เทคโนโลยีสารสนเทศ กล้อง Kinect

บทคัดย่อ

ปัจจุบันการนำเสนอผลงาน สินค้าและบริการ มีอิทธิพลต่อผู้ผลิตและผู้บริโภคเป็นอย่างมาก เห็นได้ชัดในการโฆษณาทางช่องรายการโทรทัศน์ต่างๆ ถ้าสินค้านั้นสามารถนำเสนอได้ออกมาให้ผู้บริโภคเข้าใจ และดึงดูดใจได้ สินค้านั้นก็จะได้รับความนิยมและมีผลการตอบรับที่ดีกว่าสินค้าที่ขาดการนำเสนอที่ดี ตัวอย่างที่ดีเกี่ยวกับการนำเสนอสินค้า คือ โทรศัพท์สมาร์ทโฟน ไอโฟน(Smart Phone : iPhone) ที่มีสตีฟ จอบส์ เป็นผู้นำเสนอที่มีคุณภาพ ซึ่งโครงงานนี้ได้เล็งเห็นประโยชน์ของการนำเสนอผลงานนี้ จึงได้นำเทคโนโลยีไร้สายมาปรับใช้กับการเรียนในห้องเรียน ทำให้การเรียนในห้องเรียนมีบรรยากาศที่ดี นักศึกษาหรืออาจารย์ที่ใช้โปรแกรมในการนำเสนอผลงานแบบไร้สายนี้สามารถใช้สื่อการนำเสนอได้สะดวก ราบรื่น และสามารถดึงดูดผู้ฟังได้ เพราะทุกครั้งที่ผู้นำเสนอใช้โปรแกรมควบคุมการนำเสนอแบบไร้สายจะต้องแสดงท่าทางต่างๆเพื่อควบคุมโปรแกรม ผู้ฟังก็สามารถควบคุมการนำเสนอในแบบไร้สายได้เช่นกัน ทำให้การเรียนมีวิธีการในการปฏิสัมพันธ์ระหว่างผู้สอนกับผู้เรียนมากยิ่งขึ้น และยังมีคุณสมบัติพิเศษอื่นๆอีกมากมายที่จะทำให้การเรียนในห้องเรียนเปลี่ยนไปในทางที่ดี หลักการทำงานที่สำคัญของโครงงานนี้อยู่ที่การรวมคุณสมบัติการตรวจจับร่างกายมนุษย์แบบไร้สาย การนำระบบตรวจจับลักษณะท่าทางของมนุษย์ และการแปลงลายมือให้อยู่ในรูปตัวอักษรที่อ่านง่ายเข้าไว้ด้วยกันเป็นหนึ่งเดียว นอกจากการนำเสนอแล้วโครงงานนี้ยังสามารถประยุกต์ใช้กับชีวิตประจำวันได้อีกมากมาย

Abstract

Nowadays, product and service presentation play a key influence on manufacturers and consumers. As can be seen from the advertisement on media today, the better the product presents, the greater extent of consumers it can reach, which also means a more resonant in response and higher sales. One good example can be seen from Smart Phone: iPhone advertisement where Steve Jobs was featured. This advertisement shows the advantage of applying wireless technology into classroom setting, which introduces a better learning environment for the students and lecturers. As a result, it helps facilitate and motivate the learners as opposed to the traditional classroom. This thesis answers all the above mentioned statements because both the presenters and listeners are able to control over the wireless presentation, enabling learning to be more interactive between the presenters and the learners with many additional attributes that will benefit a positive change in the set learning environment. The principle of this project lies in the integration of all wireless detecting attributes on human body, actions, postures and transformation of handwriting. Moreover, this project can be applied to many other routinely used technologies in the future.

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

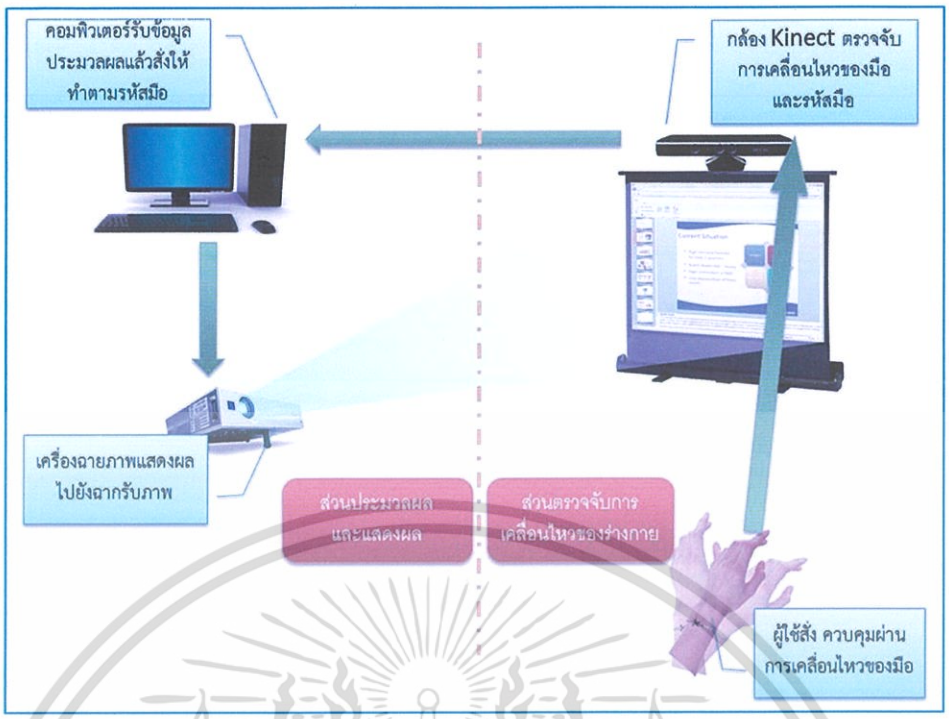
บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องจากระบบการสอนของอาจารย์ หรือระบบการนำเสนองานต่างๆนั้น ในปัจจุบันนี้ถือว่าพัฒนาไปไกลมากขึ้นจากอดีต โดยได้มีการพัฒนาซอร์ฟแวร์เพื่อช่วยให้ผู้นำเสนอได้รับความสะดวกมากขึ้น กล่าวคือผู้นำเสนอสามารถจัดทำสื่อนำเสนอทั้งหมดในรูปของไฟล์ไว้ล่วงหน้าได้ พร้อมกันนี้ยังทำให้สื่อนำเสนอมีความน่าสนใจมากขึ้น ตัวอย่างของซอร์ฟแวร์ที่ได้รับความนิยมในการจัดทำสื่อนำเสนอ เช่นโปรแกรมพาวเวอร์พอยต์ (Powerpoint) และโปรแกรมโอเพนออฟฟิซ (Open office) และการใช้ซอร์ฟแวร์เหล่านี้ในการนำเสนอให้กับคนจำนวนมาก จำเป็นต้องใช้เครื่องฉายภาพ (Projector) เข้ามาร่วมด้วย

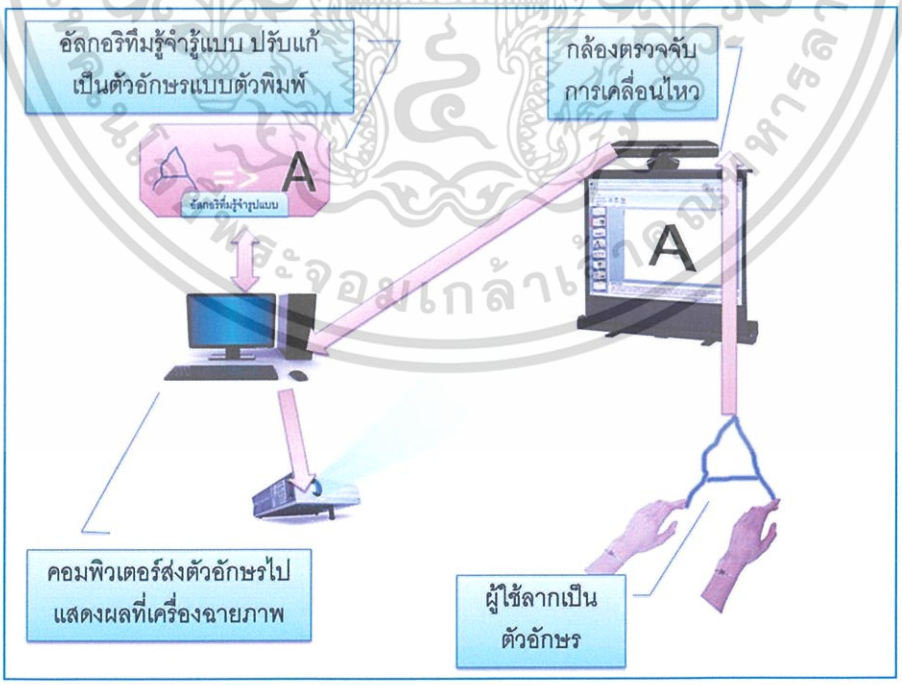
ซอร์ฟแวร์ในการนำเสนอได้รับการพัฒนาอย่างต่อเนื่อง ซึ่งทำให้มีฟังก์ชันมากมายที่ทำให้สามารถเพิ่มเติมรายละเอียดบนสื่อนำเสนอเหล่านั้นได้ทันทีที่ต้องการ เช่นฟังก์ชันที่ทำให้สามารถเขียนข้อความ วาดรูป และเน้นข้อความได้ในขณะนำเสนอโดยใช้เมาส์ทำหน้าที่เปรียบเสมือนเป็นปากกา แต่ข้อจำกัดคือการใช้เมาส์ดังกล่าว ผู้นำเสนอต้องอยู่ที่คอมพิวเตอร์เท่านั้น ทำให้ในระหว่างการนำเสนออาจต้องมีการเดินไปเดินมาระหว่างเวทีในการนำเสนอกับคอมพิวเตอร์ ซึ่งทำให้การนำเสนออาจขาดช่วงและไม่น่าสนใจอีกต่อไป ดังนั้นเพื่อแก้ปัญหาดังกล่าว จึงมีแนวความคิดที่จะพัฒนาระบบไร้สายสำหรับการสร้างและปรับปรุงข้อความบนสื่อนำเสนอ โดยที่ผู้นำเสนอสามารถที่จะเขียนข้อความ วาดรูป และเน้นข้อความได้เลยโดยการเขียนบนอากาศ ดังนั้นผู้นำเสนอไม่มีความจำเป็นที่จะต้องเดินกลับไปยังคอมพิวเตอร์อีก ทำให้การนำเสนอมีความต่อเนื่องและดูมีความเป็นมืออาชีพมากขึ้น เทคโนโลยีที่สามารถนำมาใช้ในการพัฒนามีอยู่อย่างหลากหลายเช่นเทคโนโลยีกล้องตรวจจับการเคลื่อนที่ของวัตถุที่เรียกว่ากล้อง Kinect หรือเทคโนโลยีการสื่อสารไร้สายซึ่งมีอยู่หลากหลายมาตรฐาน เป็นต้น เมื่อพิจารณาเทคโนโลยีของการสื่อสารไร้สายแล้ว จะต้องมีการสร้างอุปกรณ์ไร้สายขนาดเล็กซึ่งทำหน้าที่เป็นปากกา และจะต้องมีการสร้างฉากซึ่งต้องประกอบไปด้วยเครื่องรับส่งหลายตัวในการทำหน้าที่ติดต่อกับอุปกรณ์ไร้สายขนาดเล็กนั้น ดังนั้นในงานวิจัยนี้ได้เลือกใช้เทคโนโลยีของกล้อง Kinect ซึ่งในที่นี้สามารถพิจารณาได้ว่านิ้วของผู้นำเสนอเปรียบเสมือนปากกา ดังนั้นหากกล้องตรวจจับการเคลื่อนที่ของนิ้วมือได้ ก็สามารถแปลงลักษณะการเคลื่อนที่ของนิ้วให้อยู่ในรูปของตัวอักษรได้เช่นกัน จากนั้นตัวอักษรดังกล่าวซึ่งอยู่ในลักษณะของลายมือจะถูกแปลงให้อยู่ในลักษณะของตัวอักษรแบบตัวพิมพ์ได้อีกด้วย

ระบบนี้ประกอบไปด้วย 2 ส่วนหลัก คือ 1) อุปกรณ์การ 2) ส่วนประมวลผลและแสดงผลการทำงานของระบบ โดยภาพรวมของโครงงานแสดงได้ดังรูปที่ 1



รูปที่ 1 ภาพรวมการทำงานของระบบ

สำหรับส่วนการแปลงตัวอักษรที่ซึ่งอยู่ในลักษณะของลายมือ (จากการเคลื่อนไหวของมือ) ให้อยู่ในลักษณะของตัวอักษรแบบตัวพิมพ์ สามารถแสดงการทำงานได้จากบล็อกไดอะแกรมดังรูปที่ 2



รูปที่ 2 บล็อกไดอะแกรมแสดงการทำงานของการแปลงรูปแบบตัวอักษรซึ่งอยู่ในลักษณะของลายมือให้อยู่ในรูปของตัวอักษรแบบตัวพิมพ์

1.2 วัตถุประสงค์ของโครงการวิจัย

1.2.1 เพื่อพัฒนาระบบไร้สายสำหรับการสร้างและปรับปรุงข้อความบนไฟล์เอกสารและสื่อนำเสนอได้

1.2.2 เพื่อพัฒนาการตรวจจับการเคลื่อนไหวของร่างกายด้วยเทคโนโลยีกล้อง Kinect โดยใช้เทคนิคการวิเคราะห์จากโครงกระดูก โดยกำหนดให้ระยะห่างระหว่างผู้นำเสนองานและกล้อง Kinnect เท่ากับ 5 เมตร

1.2.3 เพื่อเพิ่มประสิทธิภาพในการนำเสนอผลงานให้น่าสนใจและเข้าใจได้ง่ายขึ้น

1.2.4 เพื่อให้เกิดความสะดวกในการนำเสนอ โดยไม่จำเป็นต้องอยู่กับคอมพิวเตอร์ตลอดเวลา

1.3 ขอบเขตของโครงการวิจัย

1.3.1 ผู้ใช้สามารถเขียนข้อความ วาดรูป และเน้นข้อความบนอากาศได้ โดยใช้การเคลื่อนไหวของร่างกาย โดยกำหนดให้ระยะทางสูงสุดระหว่างผู้นำเสนองานและกล้อง Kinnect เท่ากับ 5 เมตร ซึ่งเป็นความกว้างของห้องเรียนขนาดเล็ก เพื่อใช้ในการทดสอบการทำงานเบื้องต้น

1.3.2 สามารถแปลงตัวอักษรลักษณะของลายมือให้อยู่ในลักษณะของตัวอักษรแบบตัวพิมพ์ได้

1.3.3 ผู้ใช้สามารถใช้การเคลื่อนไหวของร่างกายในการจัดการหน้าต่างบนหน้าจอคอมพิวเตอร์ได้ กล่าวคือสามารถเปลี่ยนหน้าของไฟล์เอกสารจากหน้าหนึ่งไปยังอีกหน้าหนึ่งได้ ย่อหน้าต่างของโปรแกรมที่ทำงานอยู่ได้ลากและวางไอคอนหรือหน้าต่างของโปรแกรมที่เปิดอยู่ได้ และสามารถเปิดและปิดโปรแกรมได้

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 ได้ศึกษาและเรียนรู้การทำงานของอุปกรณ์ Kinect เพื่อใช้ในการนำเสนอผลงาน

1.4.2 ผู้นำเสนอรู้สึกพึงพอใจในใช้ระบบนี้ในการนำเสนอ

1.4.3 สามารถนำความรู้ที่ได้ไปพัฒนาต่อยอดเพื่อนำไปใช้ได้จริงในการนำเสนอในห้องประชุมหรือห้องเรียนได้

1.5 ระเบียบวิธีวิจัย

1.5.1 ศึกษาทฤษฎีที่เกี่ยวข้อง ซึ่งอยู่บนพื้นฐานของการวิเคราะห์ข้อมูลความลึก ข้อมูลสี และข้อมูลตำแหน่งของกระดูก เพื่อนำมาคำนวณหาระยะทางระหว่างร่างกายมนุษย์และกล้อง

รวมถึงการวิเคราะห์โครงกระดูกทั้งหมดของร่างกาย จากนั้นจะใช้เทคนิคการติดตามการเคลื่อนไหวของร่างกาย และกำหนดลักษณะของการเคลื่อนไหวมือและนิ้วเพื่อใช้ในการควบคุมไฟล์เอกสาร

1.5.2 ศึกษาและจัดเตรียมเครื่องมือและอุปกรณ์ที่เกี่ยวข้อง

1.5.3 ออกแบบระบบการทำงานโดยรวม

1.5.4 ออกแบบฮาร์ดแวร์บนพื้นฐานของต้นทุนที่ต่ำ โดยเลือกใช้วัสดุและอุปกรณ์ที่เหมาะสม

1.5.5 ติดตั้งระบบ และทำการทดสอบการใช้งานทั้งหมด

1.5.6 เคราะห์และแก้ไขข้อผิดพลาดจากการทดสอบการใช้งาน แล้วทำการทดสอบใหม่ไปเรื่อยๆ จนกระทั่งได้ค่าความผิดพลาดที่ลู่เข้าค่าใดค่าหนึ่งที่น้อยที่สุด

1.5.7 ประเมินผลโครงการ และจัดทำรายงานโครงการฉบับสมบูรณ์เสนอต่อคณะวิศวกรรมศาสตร์ ซึ่งเป็นผู้ให้ทุนสนับสนุนโครงการวิจัย

1.6 อุปกรณ์ที่ต้องใช้

1.6.1 ฮาร์ดแวร์

- กล้อง Kinect 1 ตัว
- คอมพิวเตอร์ 1 เครื่อง
- โปรเจคเตอร์ 1 เครื่อง

1.6.2 ซอฟต์แวร์

- Microsoft Visual C#
- Kinect for Window

1.7 แผนการดำเนินงานโครงการวิจัย

รดำเนินงาน	ระยะเวลา												หมายเหตุ
	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	
ศึกษาทฤษฎีกี่ยวข้อง	←	→											
ศึกษาและเตรียมเรื่องมือและอุปกรณ์ที่ยวข้อง		←	—	→									

รดำเนินงาน	ระยะเวลา												หมายเหตุ
	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	
ออกแบบ บบการ งานโดยรวม				← →									
ออกแบบ ร์ดแวร์บน ฐานของ ทุนที่ต่ำ โดย วกใช้วัสดุ ะอุปกรณ์ที่ มาะสม					←	—	→						
ติดตั้งระบบ ะทำการ สอบการใช้ งทั้งหมด								←	—	—	→		
วิเคราะห์ ะแก้ไข ผิดพลาด กการ เสอบการใช้ น แล้วทำ รทดสอบ มไปเรื่อยๆ เกระทั่งได้ค่า ามผิดพลาด ูเข้าค่าใดค่า ่งที่น้อยที่สุด										←	—	→	
ประเมินผล รงการ และ ทำรายงาน รงการฉบับ บูรณ์												← →	

บทที่ 2
ทฤษฎีพื้นฐานที่ใช้

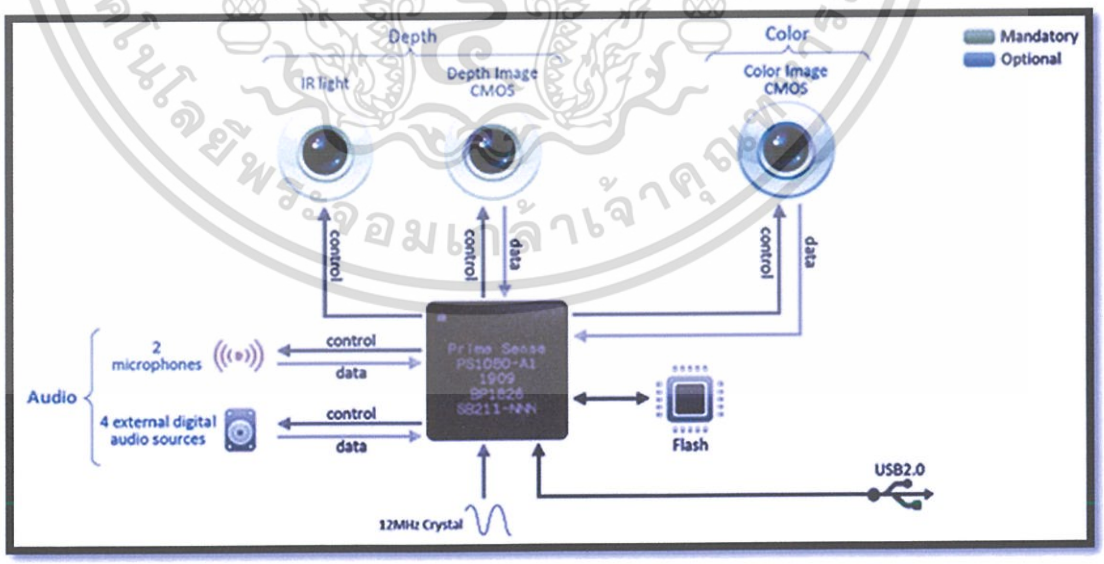
2.1 กล้อง Kinect

Kinect for Xbox 360 หรือ Kinect เป็นอุปกรณ์รับการตรวจจับการเคลื่อนไหวโดย Microsoft สำหรับการควบคุมเกมวีดีโอ Xbox 360 โดยใช้อุปกรณ์ต่อพ่วงเพิ่มเติมแบบเว็บแคม สำหรับตัวควบคุม Xbox 360 ซึ่งสามารถให้ผู้ใช้ควบคุมและโต้ตอบกับ Xbox 360 โดยไม่ต้องสัมผัสเครื่องบังคับเกมเลยแต่จะผ่านอินเตอร์เฟสของผู้ใช้ที่มีความเป็นธรรมชาติโดยการใช้ท่าทางและการพูดออกคำสั่ง โดยมีวัตถุประสงค์เพื่อที่จะขยายผู้เล่นของ Xbox 360 ให้มากขึ้น Kinect ต้องแข่งขันกับ Wii Remote Plus และ PlayStation Move กับตัวควบคุมการเคลื่อนไหวของตาของ PlayStation สำหรับเครื่องเล่นในบ้านอย่าง Wii และ PlayStation 3 ตามลำดับ

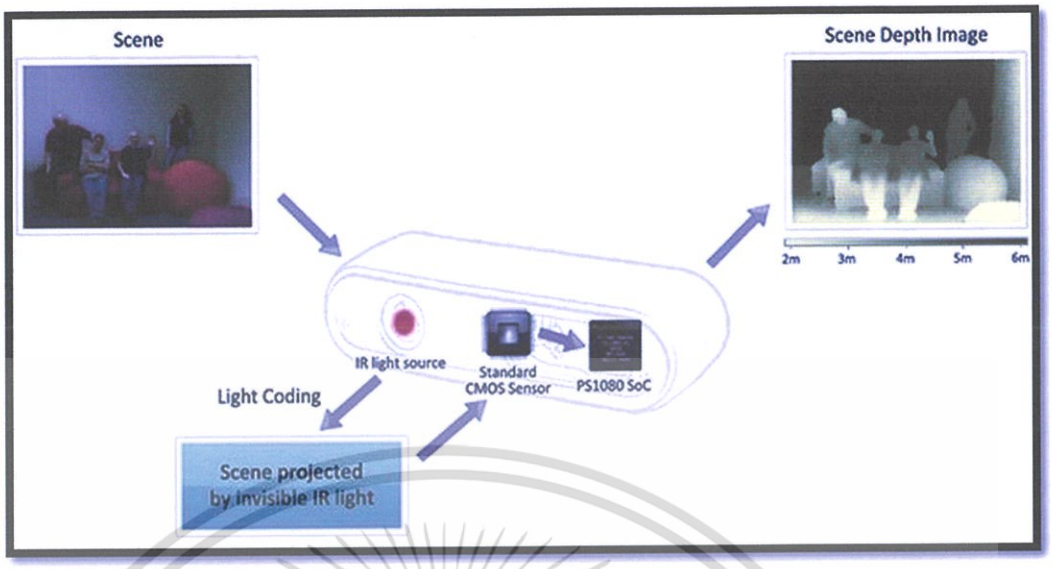
2.1.1 องค์ประกอบของ Kinect

องค์ประกอบภายในของ Kinect มีดังนี้

1. ไมโครโฟน 4 ตัว
2. กล้อง 2 ตัว (infrared CMOS และ color CMOS) และ IR projector
3. พัดลม
4. 64 MB ของ Hynix DDR2 SDRAM
5. มอเตอร์
6. Accelerometer 3 แกน
7. เซ็นเซอร์หลัก PS1080-A2



รูปที่ 2.1 การทำงานภายในของ Kinect [3]



รูปที่ 2.2 การรับค่าอินพุตของ Kinect [4]

2.1.2 วิธีการติดตามการเคลื่อนไหวของเซ็นเซอร์กล้อง

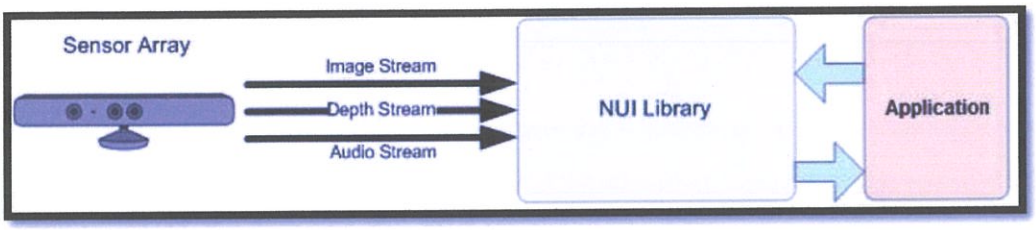
การติดตามการเคลื่อนไหวจะมีสองส่วนหลัก คือ โปรเจคเตอร์ และ IR VGA camera จากการเล่นเกมส์โดยใช้กล้อง กล้องจะสามารถแยกตัวผู้เล่นออกจากพื้นหลังได้โดยการใช้ความลึกทำให้มีการแสดงเฉดสีของร่างกายเป็นสีแดง หรือเขียว ส่วนอื่นๆจะแสดงด้วยสีเทา โดยจะมีหลักการทำงาน คือ เมื่อเซ็นเซอร์กล้องจับภาพของร่างกายมนุษย์ได้ก็จะใช้หลักการพื้นฐานทั่วไปในการคิดโดยคิดความสูงจาก x-foot tall ถึง x-foot tall มีแขน 2 ข้าง และมีขา 2 ข้าง แม้ว่าผู้เล่นจะสวมเสื้อผ้า หรือมีความยาวของผมเกินไหล่ก็ตาม กล้องก็ยังสามารถแยกแยะได้ ซึ่งมีความถูกต้องแม่นยำมาก กล้องจะรับภาพทั้งหมด 30 เฟรมต่อวินาที และนำมาจัดเรียงเพื่อให้ได้โครงกระดูกร่างกายออกมา แม้ว่าจะมีการเคลื่อนที่ หรือเคลื่อนไหว ก็จะมีการจัดเรียงใหม่เพื่อให้ได้โครงกระดูกของผู้เล่นออกมาทั้งหมด ข้อเสียอย่างเดียวก็คือ ไม่มีข้อต่อที่นิ้วมือ ทำให้การพัฒนาเกมส์อย่างเช่นเกมส์จำพวกยิงปืนอาจจะไม่สามารถพัฒนาได้

2.2 Software Development Kit (SDK)

Kinect สำหรับ Windows Software Development Kit (SDK) Beta จากงานวิจัยของ Microsoft เป็นชุดเครื่องมือเริ่มต้นสำหรับนักพัฒนาแอพพลิเคชั่น ชุดเครื่องมือนี้ทำให้ใช้งานได้ง่ายขึ้นสำหรับงานวิจัยในมหาวิทยาลัยและกลุ่มบุคคลที่ต้องการที่จะสร้างประสบการณ์ใหม่ๆโดยใช้เทคโนโลยีเซ็นเซอร์ของ Kinect สำหรับ Xbox 360 บนเครื่องคอมพิวเตอร์ที่ทำงานบนระบบปฏิบัติการ Windows 7

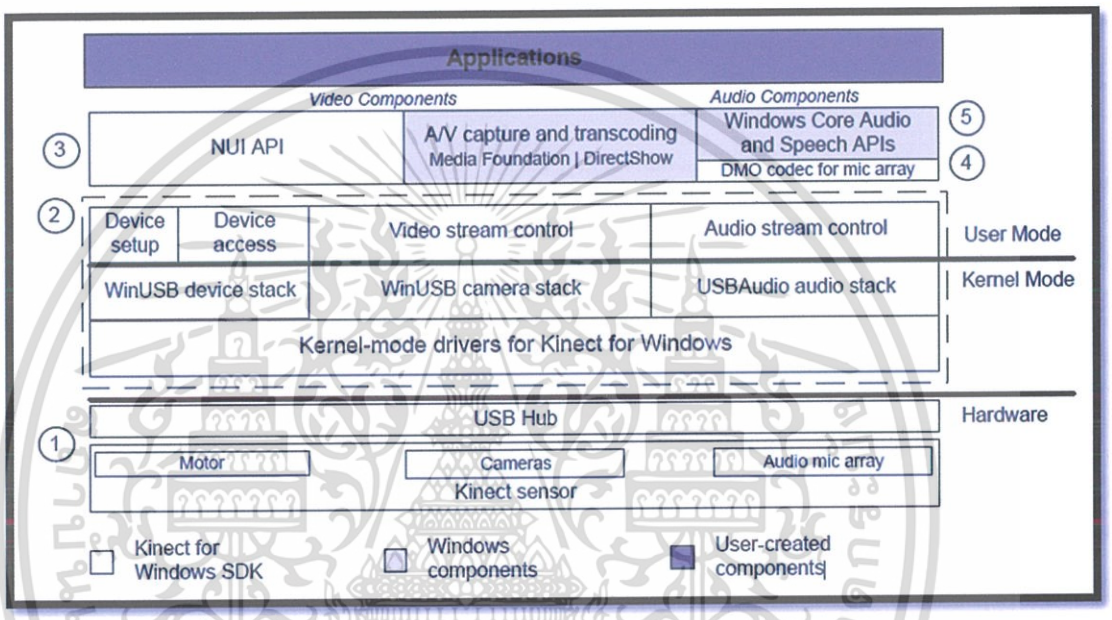
2.2.1 กล้อง Kinect สำหรับสถาปัตยกรรมวินโดวส์

กล้อง Kinect มีซอฟท์แวร์ไลบรารี่ที่เรียกว่า Windows SDK Beta และเครื่องมือต่างๆที่ช่วยให้นักพัฒนานำเอาไลบรารี่และเครื่องมือเหล่านี้มาใช้สำหรับติดต่อกับแอพพลิเคชั่นได้ โดยมีรูปแบบการรับอินพุต การตรวจจับ และการตอบสนองกับเหตุการณ์ต่างๆในโลกแห่งความเป็นจริงได้ ดังที่จะแสดงในรูปที่ 2.3



รูปที่ 2.3 การตอบสนองของฮาร์ดแวร์และซอฟท์แวร์กับแอพพลิเคชั่น [5]

องค์ประกอบของกล้อง Kinect สำหรับ Windows SDK Beta ซึ่งจะแสดงให้เห็นในรูปที่ 2.4



รูปที่ 2.4 กล้อง Kinect สำหรับสถาปัตยกรรม Windows SDK Beta [6]

องค์ประกอบของกล้อง Kinect สำหรับ Windows SDK Beta ที่แสดงในรูปที่ 2.4 ประกอบด้วย

1. อุปกรณ์ฮาร์ดแวร์ของกล้อง Kinect

องค์ประกอบของฮาร์ดแวร์ประกอบด้วย เซ็นเซอร์ของ Kinect และ USB hub ซึ่งเชื่อมต่อกับคอมพิวเตอร์

2. Microsoft Kinect drivers

ไดรเวอร์ของระบบปฏิบัติการ Windows 7 สำหรับเซ็นเซอร์ของ Kinect ซึ่งจะถูกติดตั้งในส่วนของการติดตั้ง SDK Beta แล้ว โดย Microsoft Kinect drivers จะรองรับ

- แนวไมโครโฟนของเซ็นเซอร์ของ Kinect จะทำหน้าที่เป็นอุปกรณ์เครื่องเสียงในระดับ kernelmode ที่เข้าถึงผ่านมาตรฐานเครื่องเสียง APIs ในระบบปฏิบัติการ Windows
- ส่งข้อมูลกระแสภาพและความลึก
- มีฟังก์ชั่นการนับอุปกรณ์เมื่อเปิดใช้งานแอพพลิเคชั่นเพื่อใช้เซ็นเซอร์ของ Kinect มากกว่า 1 ตัวที่เชื่อมต่อกับคอมพิวเตอร์อยู่

3. NUI API

ชุดของ APIs ที่รับข้อมูลจากเซ็นเซอร์ภาพและควบคุมอุปกรณ์ Kinect

4. KinectAudio DMO

Kinect DMO จะช่วยขยายการรองรับแนวไมโครโฟนในระบบปฏิบัติการ Windows 7 เพื่อแสดงการทำงานของการแปลงของรูปแบบรังสีและแหล่งที่มา

5. Windows 7 standard APIs

เสียง, การพูด, และสื่อ APIs ในระบบปฏิบัติการ Windows 7

2.2.2 NUI API NUI API

แกนหลักของกล้อง Kinect สำหรับ Windows API ซึ่งจะเป็นส่วนการรองรับภาพพื้นฐาน และรูปแบบการจัดการอุปกรณ์ ประกอบด้วย

- การเข้าถึงเซ็นเซอร์ของ Kinect ที่เชื่อมต่อกับคอมพิวเตอร์
- การเข้าถึงกระแสข้อมูลภาพและความลึกจากเซ็นเซอร์ภาพ Kinect
- การส่งข้อมูลภาพและความลึกที่ผ่านการประมวลแล้วไปรองรับการตรวจจับโครงกระดูก

การเริ่มต้นสำหรับ NUI API

Microsoft Kinect drivers นั้นรองรับการใช้ Kinect หลายตัวบนคอมพิวเตอร์เครื่องเดียว NUI API ประกอบด้วยฟังก์ชั่นที่นับจำนวนเซ็นเซอร์เพื่อพิจารณาว่ามีเซ็นเซอร์กี่ตัวที่เชื่อมต่อกับตัวกล้อง แล้วเก็บชื่อแต่ละเซ็นเซอร์ไว้ และเปิดแต่ละตัว แล้วตั้งค่าคุณลักษณะสตรีมมิ่งของแต่ละเซ็นเซอร์ แม้ว่า SDK Beta จะรองรับการใช้เซ็นเซอร์ Kinect หลายตัวในแอพพลิเคชั่น แต่จะมีแค่แอพพลิเคชั่นเดียวที่สามารถใช้แต่ละเซ็นเซอร์ได้ในเวลานั้นๆ

การนับจำนวนเซ็นเซอร์และการเข้าถึง

การเริ่มต้น NUI API และการใช้เซ็นเซอร์ของ Kinect ในภาษา C#

1. สร้างออบเจ็คท์ Runtime ใหม่และให้ค่าภายในว่าง ดังโค๊ดต่อไปนี้

```
nui = new Runtime();
```

คอนสตรัคเตอร์นี้สร้างออบเจ็คท์ที่แสดงอุปกรณ์เซ็นเซอร์ Kinect ตัวแรกในระบบ

2. เรียก Runtime.Initialize เพื่อเริ่ม NUI API สำหรับเซ็นเซอร์

3. เรียกเมธอดเพิ่มเติมในการจัดการอินเตอร์เฟสเพื่อแสดงภาพและข้อมูลโครงกระดูกและจัดการกล้อง

4. เรียก Runtime.Shutdown เมื่อใช้เซ็นเซอร์ Kinect เสร็จแล้ว

ถ้าแอพพลิเคชั่นใช้เซ็นเซอร์มากกว่า 1 ตัว

1. เรียก MSR_NuiDeviceCount เพื่อนับจำนวนเซ็นเซอร์ที่ปรากฏอยู่
2. สร้างออบเจ็คท์ Runtime ใหม่และผ่านดัชนีของเซ็นเซอร์ ดังโค๊ดต่อไปนี้

```
nui = new Runtime(index);
```

คอนสตรัคเตอร์นี้สร้างออบเจ็คท์ที่แสดงอุปกรณ์เซ็นเซอร์ Kinect แต่ละตัวในระบบ

3. เรียก Runtime.Initialize เพื่อเริ่ม NUI API สำหรับเซ็นเซอร์

4. เรียกเมธอดเพิ่มเติมในการจัดการอินเตอร์เฟสเพื่อแสดงภาพและข้อมูลโครงกระดูกและจัดการกล้อง

5. เรียก Runtime.Shutdown เมื่อใช้เซ็นเซอร์เสร็จแล้ว

ตัวเลือกการเริ่มต้น

NUI API จัดการข้อมูลจากเซ็นเซอร์ Kinect ผ่าน multi-stage pipeline ในตอนแรกแอพพลิเคชั่นจะเจาะจงระบบย่อยที่ใช้ ดังนั้น runtime สามารถเริ่มเรียกส่วนของ pipeline แอพพลิเคชั่นสามารถเลือกหนึ่งหรือมากกว่าหนึ่งตัวเลือกดังนี้

- Color แอพพลิเคชั่นแสดงข้อมูลภาพสีจากเซ็นเซอร์
- Depth แอพพลิเคชั่นแสดงข้อมูลภาพความลึกจากเซ็นเซอร์
- Depth and player index แอพพลิเคชั่นแสดงข้อมูลความลึกจากเซ็นเซอร์และเรียกดัชนีของผู้เล่น ที่เครื่องกลไกการตรวจจับโครงกระดูกสร้างขึ้น
- Skeleton แอพพลิเคชั่นใช้ข้อมูลตำแหน่งโครงกระดูก

ตัวเลือกเหล่านี้จะพิจารณาชนิดข้อมูลและความละเอียดที่ถูกต้องสำหรับแอพพลิเคชั่น ตัวอย่างเช่น ถ้าแอพพลิเคชั่นไม่ได้ชี้มาที่การเริ่มต้น NUI API ที่ใช้ความลึก ก็ไม่สามารถที่จะเปิดข้อมูลความลึกในตอนหลังได้

2.2.3 ข้อมูลภาพ NUI

NUI API ต้องการที่จะปรับปรุงการตั้งค่าสำหรับแนวเซ็นเซอร์ของ Kinect และเปิดให้สามารถเข้าถึงข้อมูลภาพจากแนวเซ็นเซอร์ได้ กระแสข้อมูลถูกส่งเป็นเฟรมภาพนิ่งที่ต่อเนื่องกัน ในการเริ่มต้น NUI แอพพลิเคชั่นจะระบุกระแสที่วางแผนไว้ว่าจะใช้ แล้วเปิดกระแสเหล่านั้นกับรายละเอียดกระแสที่เฉพาะเพิ่มเติม ซึ่งประกอบด้วย ความละเอียดของกระแส ชนิดภาพ และจำนวนของบัฟเฟอร์ที่ runtime ต้องใช้เก็บเฟรมที่เข้ามา ถ้า runtime ใช้บัฟเฟอร์จนเต็มก่อนที่แอพพลิเคชั่นจะได้รับและปล่อยเฟรมออกไป runtime จะทิ้งเฟรมที่อยู่นานที่สุดและนำบัฟเฟอร์นั้นมาใช้ใหม่ ผลลัพธ์นั้นมีความเป็นไปได้ที่เฟรมจะตกหล่น แอพพลิเคชั่นสามารถร้องขอบัฟเฟอร์ได้มากถึง 4 บัฟเฟอร์ ซึ่ง 2 บัฟเฟอร์ก็เพียงพอแล้ว แอพพลิเคชั่นสามารถเข้าถึงประเภทของข้อมูลภาพจากแนวเซ็นเซอร์ได้ดังนี้

- ข้อมูลสี
- ข้อมูลความลึก
- ข้อมูลแต่ละส่วนของผู้เล่น

ข้อมูลภาพสี : คุณภาพ, รูปแบบ และแบนด์วิดท์

ข้อมูลภาพสีจะแสดงที่ระดับ 2 คุณภาพและใน 2 รูปแบบที่แตกต่างกัน

- ระดับคุณภาพพิจารณาว่าข้อมูลถูกส่งจากเซ็นเซอร์ Kinect ไปคอมพิวเตอร์ได้เร็วเท่าไร
- รูปแบบสีที่แสดงพิจารณาว่าข้อมูลภาพที่ส่งกลับมายังโค๊ดแอพพลิเคชั่นถูกเข้ารหัสเป็น RGB หรือว่า YUV

คุณภาพภาพและแบนด์วิดท์

แนวเซ็นเซอร์ใช้การเชื่อมต่อผ่าน USB ส่งผ่านข้อมูลไปยังคอมพิวเตอร์ และการเชื่อมต่อนั้นจะจัดการขนาดของแบนด์วิธให้ คุณภาพข้อมูลภาพที่เลือกใช้นั้นจะเป็นตัวกำหนดว่าต้องใช้แบนด์วิธเท่าไหร่ ภาพคุณภาพสูงส่งข้อมูลมากในแต่ละเฟรมและอัพเดท น้อยครั้ง ในขณะที่ภาพคุณภาพทั่วไปจะอัพเดทบ่อยกว่าพร้อมกับมีการสูญเสียบ้างในคุณภาพของภาพเนื่องจากการบีบอัด

- ที่คุณภาพทั่วไป ข้อมูลภาพสีเบเยอร์ที่เซ็นเซอร์ส่งกลับที่ 1280×1024 ถูกบีบอัดและแปลงเป็น RGB ก่อนที่จะส่งไปยัง runtime จากนั้น runtime จะคลายการบีบอัดข้อมูลก่อนส่งผ่านข้อมูลไปยังแอพพลิเคชั่น การใช้การบีบอัดทำให้เป็นไปได้ว่าจะส่งข้อมูลสีที่อัตราเฟรมสูงเป็น 30 เฟรมต่อวินาที แต่อัลกอริทึมที่ใช้ทำให้เกิดการสูญเสียบางส่วนของภาพ
- ที่คุณภาพสูง ข้อมูลภาพสีจะไม่ถูกบีบอัดในเซ็นเซอร์ ซึ่งจะส่งไปยัง runtime แบบที่ถูกจับภาพต้นฉบับโดยเซ็นเซอร์ เพราะข้อมูลที่ไม่ถูกบีบอัด ข้อมูลมากต้องถูกส่งต่อเฟรมและอัตราเฟรมสูงสุดไม่เกิน 15 เฟรมต่อวินาที เช่นเดียวกัน ข้อมูลที่ไม่ถูกบีบอัดต้องการให้ระบบ NUI จัดสรรบัฟเฟอร์ขนาดใหญ่กว่า

ต้องประเมินว่าจะใช้ข้อมูลสีประเภทใดจึงจะเหมาะกับแอพพลิเคชั่นมากที่สุด: ข้อมูลคุณภาพสูงใช้บัฟเฟอร์ขนาดใหญ่ที่อัตราเฟรมที่ต่ำหรืออัตราเฟรมที่สูงกว่าแต่มีการสูญเสียและใช้หน่วยความจำน้อยกว่า

รูปแบบ ข้อมูลสีมี 2 รูปแบบ

- สี RGB มี 32 บิท linear X8R8G8B8-formatted color bitmaps ใน sRGB color space ถ้าจะใช้ข้อมูล RGB แอพพลิเคชั่นควรที่จะระบุสีหรือชนิดภาพสี YUV เมื่อเปิดสตรีม
- สี YUV มี 16 บิท gamma-corrected linear UYVY-formatted color bitmaps ที่ gamma correction ใน YUV space จะเท่ากับ sRGB gamma ใน RGB space เพราะกระแส YUV ใช้ 16 บิทต่อพิกเซล รูปแบบนี้ใช้หน่วยความจาน้อยในการเก็บข้อมูลบิทแม็พและจัดสรรบัฟเฟอร์หน่วยความจำน้อยเมื่อเปิดสตรีม ถ้าจะใช้ข้อมูล YUV แอพพลิเคชั่นควรระบุชนิดภาพ YUV ต้นฉบับเมื่อเปิดสตรีม ข้อมูล YUV จะปรากฏแค่ที่ความละเอียด 640×480 และที่ 15 เฟรมต่อวินาทีเท่านั้น

สีทั้งสองรูปแบบจะถูกคำนวณจากข้อมูลกล้องเดียวกัน ดังนั้นข้อมูล YUV และข้อมูล RGB จะแสดงภาพเดียวกัน เลือกรูปแบบข้อมูลที่สะดวกที่สุดในการใช้งานแอพพลิเคชั่น

ข้อมูลความลึก

กระแสข้อมูลความลึกมีเฟรมที่มากถึง 13 บิทในแต่ละพิกเซลที่ให้ระยะห่างในหน่วยมิลลิเมตรไปยังวัตถุที่อยู่ใกล้ที่สุดที่แต่ละพิกัด x และ y ในเขตของเซ็นเซอร์ของภาพ กระแสข้อมูลความลึกที่ปรากฏมีดังนี้

- เฟรมขนาด 640×480 พิกเซล
- เฟรมขนาด 320×240 พิกเซล
- เฟรมขนาด 80×60 พิกเซล

แอพพลิเคชั่นสามารถประมวลข้อมูลจากกระแสความลึกที่รองรับรูปแบบที่กำหนดเองต่างๆ ดังเช่น การตรวจจับการเคลื่อนไหวของผู้ใช้ หรือการระบุวัตถุและไม่สนใจพื้นหลังในขณะที่แอพพลิเคชั่นกำลังทำงาน

ข้อมูลแต่ละส่วนของผู้เล่น

ใน Kinect สำหรับ Widows SDK Beta ระบบ Kinect ประมวลผลข้อมูลเซ็นเซอร์เพื่อระบุคน 2 คนที่อยู่ด้านหน้าของแนวเซ็นเซอร์และจากนั้นสร้างแผนที่แต่ละส่วนของผู้เล่น แผนที่นี้เป็นบิทแม็พในแต่ละค่าพิกเซลที่สอดคล้องกับดัชนีผู้เล่นของคนที่อยู่ใกล้กับกล้องที่สุดในขอบเขตภาพ แม้ว่าข้อมูลแต่ละส่วนของผู้เล่นเป็นกระแสตรรกะที่แยกจากกัน ในทางปฏิบัติข้อมูลความลึกและข้อมูลผู้เล่นแต่ละส่วนถูกประสานเป็นเฟรมเดียว

- 13 บิตที่มีความสำคัญสูงของแต่ละพิกเซลแสดงระยะทางจากเซ็นเซอร์ไปยังวัตถุที่อยู่ใกล้ที่สุดในหน่วยมิลลิเมตร
- 3 บิตที่สำคัญต่ำของแต่ละพิกเซลแสดงดัชนีผู้เล่นที่ถูกตรวจจับซึ่งมองเห็นที่พิกัดพิกเซลของ x และ y บิทเหล่านี้ถูกเก็บเป็นค่าจำนวนเต็มและไม่ถูกใช้เป็นธงในขอบเขตบิท

ค่าดัชนีผู้เล่นที่เป็นศูนย์แสดงว่าไม่มีผู้เล่นที่ถูกพบในบริเวณนั้น ค่า 1 และ 2 ระบุตัวผู้เล่น แอพพลิเคชั่นใช้ข้อมูลแต่ละส่วนของผู้เล่นได้อย่างปกติเหมือนเป็นหน้ากากที่แยกระบุผู้ใช้หรือบริเวณที่สนใจจากสีต้นฉบับและภาพความลึก

ข้อมูลภาพที่ดึงมาได้

โค้ดแอพพลิชั่นรับเฟรมล่าสุดของข้อมูลภาพโดยเรียกเมธอดการดึงเฟรมและการส่งผ่านบัฟเฟอร์ ถ้าเฟรมสุดท้ายของข้อมูลพร้อมแล้ว จะทำการคัดลอกไปยังบัฟเฟอร์ ถ้าโค้ดเรียกเฟรมของข้อมูลเร็วกว่าที่เฟรมใหม่จะมาถึง สามารถเลือกได้ว่าจะรอเฟรมต่อไปหรือส่งทันทีและลองใหม่ภายหลัง NUI Image Camera API ไม่เคยทำให้เฟรมเดียวกันของข้อมูลมีมากกว่า 1 ครั้ง แอพพลิเคชั่นสามารถใช้อย่างใดอย่างหนึ่งจากโมเดลการใช้ 2 แบบนี้

- Polling Model เป็นทางเลือกที่ง่ายที่สุดในการอ่านเฟรมข้อมูล

เริ่มต้นโค้ดแอพพลิเคชั่นเรียกเฟรมและระบุว่าจะรอเฟรมข้อมูลต่อไปนานแค่เท่าไร(ระหว่าง 0 ถึงจำนวนอินฟินิตี้ของมิลลิวินาที) เมธอดการเรียกจะคืนค่าเมื่อเฟรมข้อมูลใหม่พร้อมหรือเมื่อหมดเวลารอ แล้วแต่ว่าอะไรมาก่อน การระบุการรอแบบอินฟินิตี้เป็นเหตุให้การเรียกเฟรมข้อมูลถูกบล็อคและรอตราบเฟรมต่อไปมาถึง เมื่อการร้องขอคืนค่าเสร็จสมบูรณ์ เฟรมใหม่พร้อมสำหรับการประมวลผล ถ้าค่า time-out ถูกกำหนดเป็น 0 โค้ดแอพพลิเคชั่นสามารถโพลล์การเสร็จสมบูรณ์ของเฟรมใหม่ขณะที่ทำงานอื่นบนเทรดเดียวกัน

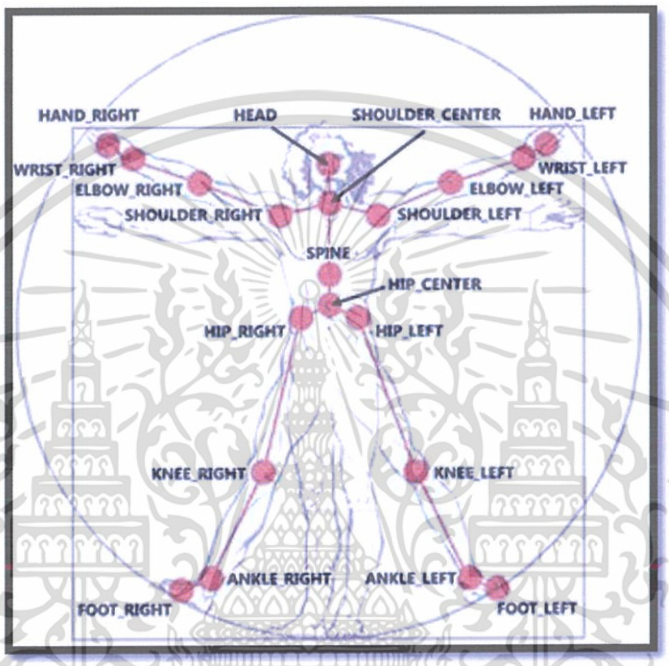
ภาษา C# เรียก ImageStream.GetNextFrame

- Event Model รองรับความสามารถในการรวบรวมการดึงเฟรมโครงกระดูกไปยังกลไกลแอพพลิเคชั่นด้วยความยืดหยุ่นที่มากกว่าและแม่นยำมากกว่า ในภาษา C# ใช้โดยการติดเหตุการณ์ Runtime.DepthFrameReady หรือ Runtime.ImageFrameReady ไปยังตัวจัดการเหตุการณ์ที่เหมาะสม เมื่อข้อมูลใหม่พร้อมเหตุการณ์จะส่งสัญญาณและตัวจัดการจะทำงานและเรียก ImageStream.GetNextFrame เพื่อรับเฟรม

2.2.4 การตรวจสอบโครงกระดูกของ NUI

NUI Skeleton API มีข้อมูลเกี่ยวกับรายละเอียดและข้อมูลที่สอดคล้องกับตำแหน่งของผู้เล่นที่ยืนอยู่หน้าเซ็นเซอร์ Kinect ได้สูงถึง 3 คน

ข้อมูลที่ส่งไปยังโค๊ดแอพพลิเคชั่นเป็นชุดของจุด เรียก *skeleton positions* ซึ่งประกอบด้วยโครงกระดูกดังที่แสดงในรูปที่ 2.5 โครงกระดูกแสดงตำแหน่งและท่าทางปัจจุบันของผู้ใช้ แอพพลิเคชั่นที่ใช้ข้อมูลโครงกระดูกต้องเจาะจงด้วยที่การเริ่มต้นของ NUI และต้องเปิดใช้งานการตรวจจับโครงกระดูก



รูปที่ 2.5 ตำแหน่งจุดของโครงกระดูกที่สัมพันธ์กับร่างกายมนุษย์ [7]

การดึงข้อมูลโครงกระดูก

โค๊ดแอพพลิเคชั่นรับเฟรมล่าสุดของข้อมูลโครงกระดูกในวิธีเดียวกับที่รับเฟรมของข้อมูลภาพ โดยการเรียกเมธอดดึงเฟรมและส่งผ่านไปยังบัฟเฟอร์ แอพพลิเคชั่นสามารถใช้ polling model หรือไม่ก็ event model ในแบบเดียวกับเฟรมภาพ แอพพลิเคชั่นต้องเลือกหนึ่งโมเดลหรือแบบอื่น ซึ่งไม่สามารถใช้ทั้งสองโมเดลพร้อมกันได้

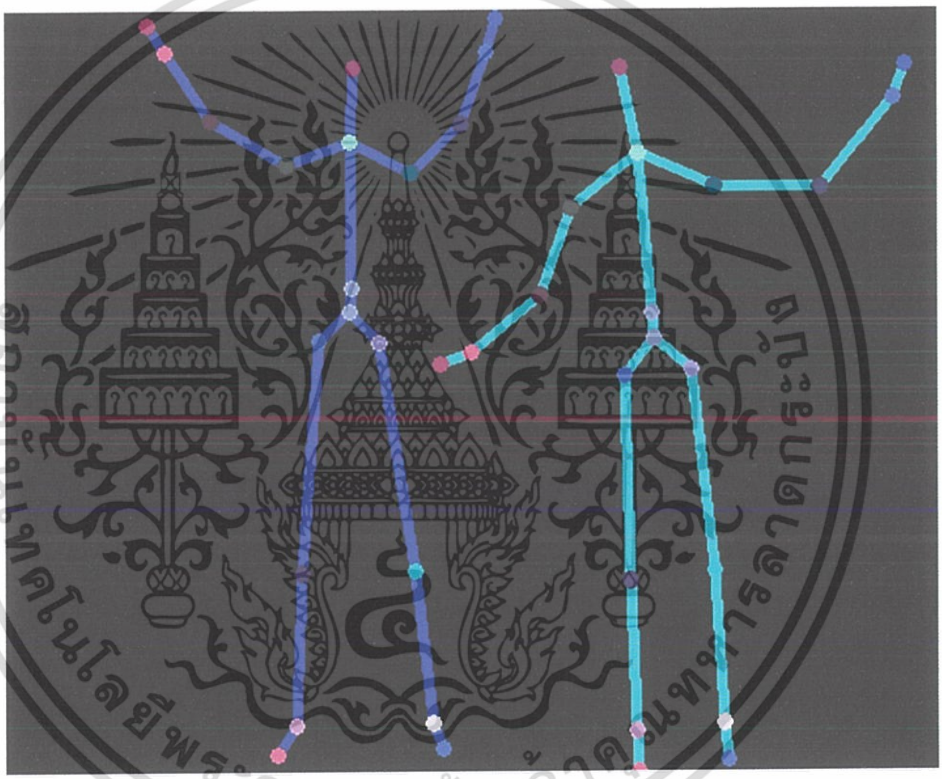
การใช้ polling model : ภาษา C# เรียก **SkeletionEngine.GetNextFrame** การใช้ event model : ภาษา C# ใช้โดยการติดเหตุการณ์ไปกับ **Runtime.SkeletionFrameReady** ไปยังตัวจัดการเหตุการณ์ที่เหมาะสม เมื่อเฟรมใหม่ของข้อมูลโครงกระดูกพร้อม เหตุการณ์จะส่งสัญญาณและตัวจัดการจะรันและเรียก **SkeletonEngine.GetNextFrame** เพื่อรับเฟรม

กลไกลการตรวจจับโครงกระดูกจะประมวลข้อมูลเฟรมความลึกเพื่อคำนวณแรงดึงดูดปกติและระนาบพื้นราบ ถ้าแอพพลิเคชั่นเจาะจงที่ตอนเริ่มต้นว่าจะใช้การตรวจจับโครงกระดูก กลไกการตรวจจับโครงกระดูกจะส่งสัญญาณเฟรมโครงกระดูกแต่ละครั้งที่ประมวลข้อมูลความลึก หรือไม่ก็ในขณะที่โครงกระดูกปรากฏในเฟรม แอพพลิเคชั่นที่ใช้ค่าแรงดึงดูดปกติและระนาบพื้นราบสามารถดึง

เฟรมโครงกระดูก เฟรมโครงกระดูกที่ดึงมาจะประกอบด้วย เวลาของภาพความลึกที่สอดคล้องกัน ดังนั้นแอพพลิเคชั่นสามารถจับคู่ข้อมูลโครงกระดูกกับข้อมูลภาพความลึก

การตรวจจับโครงกระดูกแบบแอคทีฟและพาสซีฟ

กลไกการตรวจจับโครงกระดูกมีการตรวจจับโครงกระดูกแบบเต็มตัวสำหรับหนึ่งหรือสองผู้เล่นในขอบเขตที่มองเห็นของเซ็นเซอร์ เมื่อผู้เล่นถูกตรวจจับแบบแอคทีฟ จะเรียกเพื่อรับเฟรมโครงกระดูกถัดไปซึ่งคืนค่าข้อมูลโครงกระดูกที่สมบูรณ์สำหรับผู้เล่น การตรวจจับแบบพาสซีฟจะเป็นไปอย่างอัตโนมัติสำหรับผู้เล่นมากถึง 4 คนที่เพิ่มเข้ามาในขอบเขตที่มองเห็นของเซ็นเซอร์ เมื่อผู้เล่นที่ถูกจับแบบพาสซีฟ เฟรมโครงกระดูกจะบรรจุแค่ข้อมูลตำแหน่งของผู้เล่นเท่านั้น โดยพื้นฐานโครงกระดูก 2 โครงแรกที่ระบบการตรวจจับโครงกระดูกพบจะเป็นแบบแอคทีฟ ดังแสดงในรูปที่ 2.6



รูปที่ 2.6 การตรวจสอบแบบแอคทีฟสำหรับผู้เล่น 2 คน

Runtime จะคืนค่าข้อมูลโครงกระดูกในเฟรมโครงกระดูกที่บรรจุอาร์เรย์ของข้อมูลของโครงสร้างข้อมูลโครงกระดูก มีเพียงหนึ่งอันของแต่ละโครงกระดูกที่ระบบการตรวจสอบโครงกระดูกจะจำไว้ ไม่ใช่ทุกเฟรมโครงกระดูกที่บรรจุลงในข้อมูลโครงกระดูก เมื่อการตรวจสอบโครงกระดูกทำงาน runtime จะส่งสัญญาณเหตุการณ์เป็นโครงกระดูกทุกเวลาที่มันประมวลเฟรมความลึก สำหรับทุกโครงกระดูกที่คืนค่า จะมีข้อมูลดังนี้

- การตรวจจับสถานะปัจจุบันของโครงกระดูกที่เชื่อมโยงกัน
- สำหรับโครงกระดูกที่ถูกตรวจจับแบบพาสซีฟ ค่านี้จะบ่งชี้การตรวจจับแค่ตำแหน่ง
- สำหรับโครงกระดูกที่ถูกตรวจจับแบบแอคทีฟ ค่าจะบ่งชี้โครงกระดูกที่ถูกตรวจจับ

- ไอดีการตรวจจับจะมีลักษณะเฉพาะที่กำหนดให้แก่ผู้เล่นแค่คนเดียวตราบที่ผู้เล่นเคลื่อนไหวไปรอบจอภาพ

ไอดีการตรวจจับเป็นการรันตีถึงว่าผู้เล่นคนเดิมจะยังคงสามารถใช้ได้อย่างต่อเนื่องเป็นเวลานานตราบเท่าที่ผู้เล่นยังอยู่ในขอบเขตที่มองเห็นได้ ไอดีการตรวจจับที่ให้เป็นการรันตีว่าจะยังเป็นดัชนีเดียวกันในอาร์เรย์ข้อมูลโครงกระดูกตราบเท่าที่ไอดีการตรวจสอบถูกใช้งานอยู่ ถ้าไอดีการตรวจสอบของโครงกระดูกของโครงกระดูกที่ดัชนีหนึ่งในอาร์เรย์มีการเปลี่ยนแปลง หนึ่งในสองสิ่งที่จะเกิดขึ้น คือ ไม่ผู้เล่นที่ถูกตรวจจับออกจากขอบเขตการมองเห็นและการตรวจจับเริ่มจับผู้เล่นอื่นในขอบเขตการมองเห็น หรือผู้เล่นที่ถูกตรวจจับออกจากขอบเขตการมองเห็นแล้วกลับมา และกำลังถูกตรวจจับอีกครั้ง

- ตำแหน่ง (ของประเภท Vector4) ที่บ่งชี้จุดศูนย์กลางมวลสำหรับผู้เล่นนั้น ค่านี้เป็นค่าจะแสดงแค่ตำแหน่งของผู้เล่นแบบพาสซีฟ
- สำหรับผู้เล่นแบบแอคทีฟ จะคืนค่าข้อมูลซึ่งประกอบด้วยข้อมูลโครงกระดูกแบบเต็มตัวในปัจจุบัน
- สำหรับผู้เล่นแบบพาสซีฟ จะคืนค่าข้อมูลซึ่งมีแค่ตำแหน่งพื้นฐานและข้อมูลคุณลักษณะเฉพาะ และไม่มีข้อมูลโครงกระดูก

2.2.5 การแปลง NUI

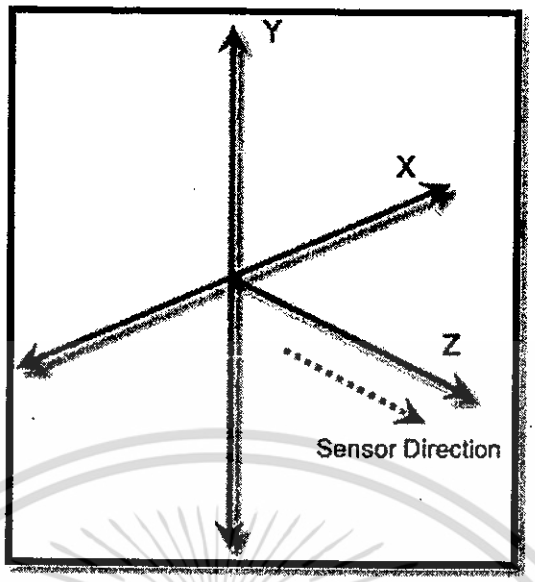
ในส่วนนี้จะอธิบายเกี่ยวกับระบบพิกัดต่างๆที่ถูกใช้กับการตรวจจับโครงกระดูกและการรองรับ API ที่มีไว้สำหรับการแปลงระหว่างระยะเหล่านี้

ระยะภาพความลึก

เฟรมภาพของแผนที่ความลึกเป็นขนาด 640×480, 320×240 หรือ 80×60 พิกเซล กับแต่ละพิกเซลที่แสดงถึงระยะทางไปยังวัตถุที่ใกล้ที่สุดในแต่ละพิกัด x และ y ในหน่วยมิลลิเมตร ค่าพิกเซลที่เป็น 0 จะบ่งชี้ว่าเซ็นเซอร์ไม่พบวัตถุใดเลยภายในขอบเขตที่อยู่ พิกัด x และ y ของเฟรมภาพไม่ได้แสดงหน่วยทางกายภายในห้อง แต่เป็นพิกเซลบนเซ็นเซอร์ การแปลความหมายของพิกัด x และ y ขึ้นอยู่กับคุณลักษณะของแสงและเซ็นเซอร์ภาพ

ระยะโครงกระดูก

ตำแหน่งโครงกระดูกผู้เล่นจะอยู่ในพิกัด x, y และ z ไม่เหมือนกับพิกัดของระยะภาพความลึก ทั้ง 3 พิกัดอยู่ในหน่วยเมตร แกน x, y และ z เป็นแกนร่างกายของเซ็นเซอร์ความลึก ซึ่งเป็นระบบพิกัดมือขวาที่วางบนแนวเซ็นเซอร์ที่จุดกำเนิดด้วยแกน z ที่เป็นบวกซึ่งขยายออกไปยังทางซ้าย (ตามแนวเซ็นเซอร์) ดังที่แสดงในรูปที่ 2.7



รูปที่ 2.7 ระบบพิกัดระยะโครงกระดูกสำหรับแนวเซ็นเซอร์[8]

แนวเซ็นเซอร์และการชดเชยความลาดเอียง

การวางตัวของแนวเซ็นเซอร์มีผลกับภาพที่กล้องบันทึก ตัวอย่างเช่น กล้องอาจจะวางบนพื้นผิวที่ไม่ได้ระดับหรือแนวเซ็นเซอร์อาจจะหมุนตั้งฉากกับขอบเขตการมองเห็นของเซ็นเซอร์ ในกรณีเหล่านี้แกน y ของระยะโครงกระดูกมักจะไม่ตั้งฉากกับพื้นหรือขนานกับแรงโน้มถ่วง ภาพที่ออกมาจะเป็นคนที่ควรยืนตรงก็จะเป็นยืนตะแคง

สำหรับเหตุผลเหล่านี้ แต่ละเฟรมโครงกระดูกบรรจุเวกเตอร์ที่อธิบายถึงค่าแรงโน้มถ่วงปกติ ซึ่งเป็นค่าที่วัดจาก accelerometer ภายใน 3 แกนที่ถูกปรับให้สอดคล้องกับเซ็นเซอร์ภาพ กรณีที่ไม่มีการเคลื่อนไหว accelerometer จะวัดแรงปกติเป็น 1-g เทียบกับแรงโน้มถ่วง ซึ่งจะทำเป็นเวกเตอร์ทิศขึ้น แรงปกติกับแรงโน้มถ่วงสามารถหาได้ในขอบเขต SkeletonFrame.NormalToGravity สำหรับภาษา C#

การคำนวณพื้น

แต่ละเฟรมโครงกระดูกจะบรรจุเวกเตอร์ระนาบพื้น ซึ่งบรรจุสัมประสิทธิ์ของสมการการประมาณค่าระนาบพื้น ระบบการตรวจสอบโครงกระดูกจะอัพเดทค่าประมาณนี้สำหรับแต่ละเฟรมและใช้เป็นระนาบสำหรับการเคลื่อนไหวพื้นหลังและแต่ละส่วนของผู้เล่น ระนาบทั่วไปเป็นไปตามสมการ

$$Ax + By + Cz + D = 0$$

โดย

A = vFloorClipPlane.x

B = vFloorClipPlane.y

C = vFloorClipPlane.z

D = vFloorClipPlane.w

สมการถูกนอมอไลซ์แล้วดังนั้น การแปลความหมายทางกายภาพของ D เป็นความสูงของกล้องจากพื้นในหน่วยเมตร ซึ่งไม่มีค่าซึ่งก็คือพื้นอาจไม่จำเป็นต้องเห็นตลอดเวลา ในกรณีนี้ ระนาบพื้นเป็น

เวกเตอร์ศูนย์ ระนาบพื้นสามารถหาได้ในขอบเขต SkeletonFrame.FloorClipPlane สำหรับภาษา C#

การทำเงาโครงกระดูก

โดยพื้นฐาน ระบบโครงกระดูกจะเป็นเงาของผู้ใช้ที่ถูกตรวจจับ สำหรับแอพพลิเคชั่นที่ใช้อวาตาร์เพื่อแทนตัวผู้ใช้ ดังนั้นการทำเงาควรทำให้ดี ถ้าอวาตาร์แสดงให้หันหน้าเข้าหาจอภาพ อย่างไรก็ตามถ้าหน้าอวาตาร์หันมาทางผู้ใช้ การทำเงาควรที่จะแสดงอวาตาร์แบบกลับด้าน ซึ่งขึ้นอยู่กับความต้องการ แอพพลิเคชั่นสามารถสร้างเมทริกซ์การแปลงจากเงาโครงกระดูก และจากนั้นประยุกต์เมทริกซ์กับจุดในอาร์เรย์ที่บรรจุตำแหน่งโครงกระดูกสำหรับโครงกระดูกนั้น แอพพลิเคชั่นจะรับผิดชอบในการเลือกระนาบที่เหมาะสมในการสะท้อนกลับไป

2.3 ความรู้เบื้องต้นเกี่ยวกับการใช้งานมุมมองโครงกระดูก

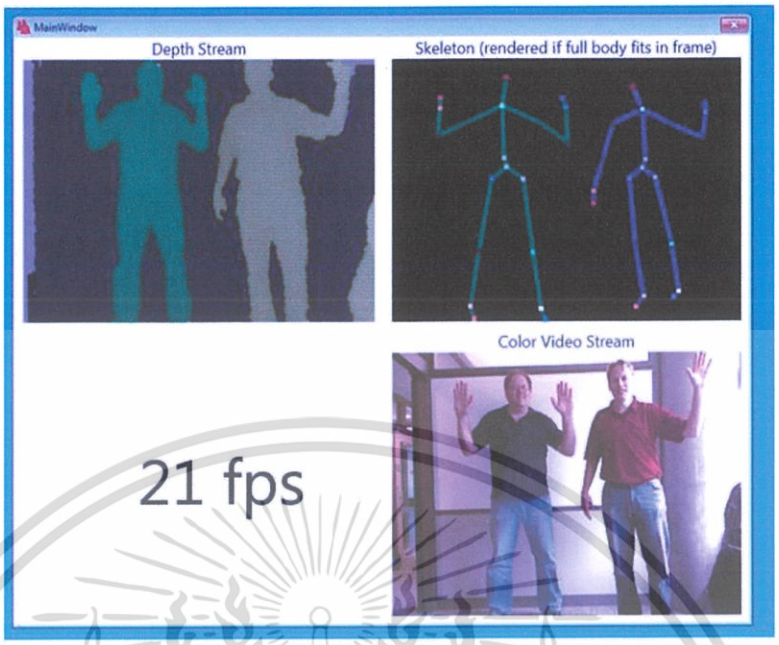
เซ็นเซอร์ของ Kinect ที่ใช้ในเครื่อง Xbox 360 จะประกอบด้วยกล้องซึ่งทำหน้าที่ส่งข้อมูลความลึก ข้อมูลสี และข้อมูลตำแหน่งกระดูก ซึ่งมีการสร้างชุดเครื่องมือสำหรับพัฒนาโปรแกรมเพื่อใช้สำหรับติดต่อกับผู้ใช้ ในที่นี้เราจะใช้ภาษา C# ในการติดต่อกับเซ็นเซอร์กล้อง เราสามารถยกตัวอย่างการใช้งานเซ็นเซอร์กล้องได้ดังต่อไปนี้

- เราสามารถแสดงระดับของสีเทาที่แตกต่างกันไปในตำแหน่งการยืน (ความลึก) ที่แตกต่างกัน
- สามารถแสดงข้อมูลตำแหน่งกระดูกเป็นเส้นโครงกระดูกได้ถึง2คน
- ภาพจากเซ็นเซอร์กล้องสามารถแสดงในรูปแบบ RGB

เซ็นเซอร์กล้องจะทำงานได้ดีที่สุดเมื่อผู้ใช้มีตำแหน่งยืนห่างจากเซ็นเซอร์ประมาณ 2.6 - 13.12 ฟุต (0.8 - 4 เมตร) ซึ่งเป็นตำแหน่งที่สามารถจับภาพผู้ใช้ได้ทั้งร่างกาย

ไลบรารี่ที่ใช้กับเซ็นเซอร์กล้อง Kinect

- Window_Loaded : ใช้ในการติดต่อกับผู้ใช้
- nui_DepthFrameReady, nui_SkeletonFrameReady, and nui_ColorFrameReady : ใช้ในการจัดการกับเหตุการณ์ต่างๆ
- convertDepthFrame : ใช้ในการแปลงข้อมูลภาพ 16 – bit เป็น 32 – bit เพื่อใช้ในการแสดงผล
- getBodySegment : ใช้ในการดึงส่วนที่เป็นกระดูก
- Window_Closed : ใช้ในการเลิกติดต่อกับผู้ใช้



รูปที่ 2.8 โปรแกรมตัวอย่างสาหรับการใช้มุมมองโครงกระดูก [9]

ในส่วนของ MainWindow จะมีการสร้างและประกาศตัวแปร ดังต่อไปนี้

- Runtime nui : ใช้ในการรับค่าจากเซ็นเซอร์
- totalFrames, lastFrames และ lastTime : ใช้ในการคำนวณจำนวนเฟรมต่อวินาที
- RED_IDX, GREEN_IDX, and BLUE_IDX : ใช้การกำหนดค่าอ้างอิงของ RGB
- depthFrame32 : เป็นอาเรย์ที่ใช้ในการเก็บข้อมูลความลึก
- jointColors : ใช้ระบุสีให้แก่ข้อต่อแต่ละส่วน

```
namespace SkeletalViewer
public partial class MainWindow : Window
{
  public MainWindow()
  {
    InitializeComponent();
  }

  Runtime nui;
  int totalFrames = 0;
  int lastFrames = 0;
  DateTime lastTime = DateTime.MaxValue;

  const int RED_IDX = 2;
  const int GREEN_IDX = 1;
  const int BLUE_IDX = 0;
  byte[] depthFrame32 = new byte[320 * 240 * 4];
```

```
Dictionary<JointID,Brush> jointColors =
    new Dictionary<JointID,Brush>() {
    {JointID.HipCenter, new SolidColorBrush(Color.FromRgb(169, 176, 155))},
    {JointID.Spine, new SolidColorBrush(Color.FromRgb(169, 176, 155))},
    {JointID.ShoulderCenter, new SolidColorBrush(Color.FromRgb(168, 230,29))},
    {JointID.Head, new SolidColorBrush(Color.FromRgb(200, 0, 0))},
    {JointID.ShoulderLeft, new SolidColorBrush(Color.FromRgb(79, 84, 33))},
    {JointID.ElbowLeft, new SolidColorBrush(Color.FromRgb(84, 33, 42))},
    {JointID.WristLeft, new SolidColorBrush(Color.FromRgb(255, 126, 0))},
    {JointID.HandLeft, new SolidColorBrush(Color.FromRgb(215, 86, 0))},
    {JointID.ShoulderRight, new SolidColorBrush(Color.FromRgb(33, 79, 84))},
    {JointID.ElbowRight, new SolidColorBrush(Color.FromRgb(33, 33, 84))},
    {JointID.WristRight, new SolidColorBrush(Color.FromRgb(77, 109, 243))},
    {JointID.HandRight, new SolidColorBrush(Color.FromRgb(37, 69, 243))},
    {JointID.HipLeft, new SolidColorBrush(Color.FromRgb(77, 109, 243))},
    {JointID.KneeLeft, new SolidColorBrush(Color.FromRgb(69, 33, 84))},
    {JointID.AnkleLeft, new SolidColorBrush(Color.FromRgb(229, 170, 122))},
    {JointID.FootLeft, new SolidColorBrush(Color.FromRgb(255, 126, 0))},
    {JointID.HipRight, new SolidColorBrush(Color.FromRgb(181, 165, 213))},
    {JointID.KneeRight, new SolidColorBrush(Color.FromRgb(71, 222, 76))},
    {JointID.AnkleRight, new SolidColorBrush(Color.FromRgb(245, 228, 156))},
    {JointID.FootRight, new SolidColorBrush(Color.FromRgb(77, 109, 243))}
};
// . . . SkeletalViewer namespace code continues
```

ในส่วนของ Window_Loaded จะมีการรับค่าวิดีโอ ข้อมูลความลึก และจะทำงานเมื่อมีการติดต่อกับเซ็นเซอร์แล้วเท่านั้น

```
private void Window_Loaded(object sender, EventArgs e)
{
  nui = new Runtime();
  try
  {
    nui.Initialize(RuntimeOptions.UseDepthAndPlayerIndex |
          RuntimeOptions.UseSkeletalTracking | RuntimeOptions.UseColor);
  }
  catch (InvalidOperationException)
  {
    // Display error message; omitted for space
    return;
  }

  try
  {
    nui.VideoStream.Open(ImageStreamType.Video, 2,
        ImageResolution.Resolution640x480, ImageType.Color);
    nui.DepthStream.Open(ImageStreamType.Depth, 2,
        ImageResolution.Resolution320x240, ImageType.DepthAndPlayerIndex);
  }
```

```
catch (InvalidOperationException)
{
  // Display error message; omitted for space
  return;
}
lastTime = DateTime.Now;

nui.DepthFrameReady += new EventHandler<ImageFrameReadyEventArgs>
      (nui_DepthFrameReady);
nui.SkeletonFrameReady +=
      new EventHandler<SkeletonFrameReadyEventArgs>
      (nui_SkeletonFrameReady);
nui.VideoFrameReady +=
      new EventHandler<ImageFrameReadyEventArgs>
      (nui_ColorFrameReady);
}
```

การทำงานจะเริ่มที่ Runtime.Initialize ก่อนที่จะทำงานที่ส่วนอื่นๆ โดยที่ Runtime.Initialize จะเริ่มรับภาพที่ได้มาจากเซ็นเซอร์ และยังมี InvalidOperationException ใช้ในการตรวจสอบความผิดพลาดเมื่อไม่พบเซ็นเซอร์ ซึ่งเป็นหลักการทำงานในส่วนของ try/catch และในส่วนนี้ก็ยังมีพารามิเตอร์อื่นอีก ดังต่อไปนี้

- UseDepthAndPlayerIndex
- UseColor
- UseSkeletalTracking
- UseDepth

ในส่วนถัดมาจะมีการรับค่าวิดีโอและข้อมูลความลึกจาก nui.VideoStream.Open และ nui.DepthStream.Open ในส่วน InvalidOperationException จะใช้ในการตรวจสอบความผิดพลาดเมื่อไม่พบข้อมูล ซึ่งเป็นหลักการทำงานในส่วนของ try/catch และในส่วนนี้ก็ยังมีพารามิเตอร์อื่นอีก ดังต่อไปนี้

- ImageStreamType.Video or ImageStreamType.Depth
- ImageResolution : แสดงความละเอียดภาพ
- ImageType : แสดงประเภทของภาพ

ซึ่งพารามิเตอร์แต่ละตัวจะมีการกำหนดค่าได้อย่างจำกัด เช่น การนำไปใช้ใน Stream color images :

- ความละเอียดของภาพจะอยู่ที่ Resolution1280x1024 และ Resolution640x480
- ประเภทของภาพจะมี 3 ประเภท คือ Color, ColorYUV และ ColorYUVRaw

การนำไปใช้ใน Stream depth and player index data :

- ความละเอียดของภาพจะอยู่ที่ Resolution320x240 และ Resolution80x60
- ประเภทของภาพจะมีประเภทเดียว คือ DepthAndPlayerIndex

ในส่วนถัดมาจะมีการเอาค่าเวลาในปัจจุบันมาเก็บไว้ในตัวแปรที่มีชื่อว่า lastTime เพื่อใช้ในการคำนวณค่าเฟรมต่อวินาที และทำหน้าที่จัดการเหตุการณ์ที่มีชื่อว่า nui_DepthFrameReady , nui_SkeletonFrameReady และ nui_ColorFrameReady

กระบวนการจัดการกับข้อมูลวิดีโอ

เมื่อมีการใช้ nui_ColorFrameReady จะมีการรับภาพวิดีโอจากกล้องเพื่อนำมาแสดง โดยใช้โค้ดดังต่อไปนี้

```
void nui_ColorFrameReady(object sender, ImageFrameReadyEventArgs e)
{
  PlanarImage Image = e.ImageFrame.Image;
  video.Source = BitmapSource.Create(
      Image.Width, Image.Height, 96, 96, PixelFormats.Bgr32, null,
      Image.Bits, Image.Width * Image.BytesPerPixel);
}
```

กระบวนการจัดการกับข้อมูลความลึก

เมื่อมีการใช้ nui_DepthFrameReady จะมีการรับภาพจากกล้องความลึก และนำมาแสดงเป็นบิตแมปโดยการใช้ BitmapSource.Create และคำนวณจำนวนเฟรมต่อวินาทีจากตัวแปร totalFrames, lastFrames และ lastTime และแสดงผลที่ frameRate.Text

```
void nui_DepthFrameReady(object sender, ImageFrameReadyEventArgs e)
{
  PlanarImage Image = e.ImageFrame.Image;
  byte[] convertedDepthFrame = convertDepthFrame(Image.Bits);

  depth.Source = BitmapSource.Create(Image.Width, Image.Height, 96, 96,
      PixelFormats.Bgr32, null, convertedDepthFrame, Image.Width * 4);

  ++totalFrames;

  DateTime cur = DateTime.Now;
  if (cur.Subtract(lastTime) > TimeSpan.FromSeconds(1))
  {
    int frameDiff = totalFrames - lastFrames;
    lastFrames = totalFrames;
    lastTime = cur;
    frameRate.Text = frameDiff.ToString() + " fps";
  }
}
```

เซ็นเซอร์ Kinect จะให้ข้อมูลความลึกเป็นค่าอาร์เรย์ 16 บิต ซึ่งการแยกแยะระดับความลึกที่ 16 บิต จะพบว่ามันยากเกินกว่าที่จะสามารถแยกแยะได้ จึงต้องมีการแปลงภาพจาก 16 บิต (gray-scale) เป็น 32 บิต(RGB) โดยมีขั้นตอน ดังต่อไปนี้

- เก็บค่าหมายเลขของผู้เล่นไว้ในตัวแปร player
- เก็บค่าข้อมูลความลึกไว้ในตัวแปร realDepth
- ผลที่ได้จะได้ว่าถ้าผู้ใช้มีตำแหน่งเข้าใกล้เซ็นเซอร์กล้องภาพที่ได้จะมีความสว่าง แต่ถ้าผู้ใช้มีตำแหน่งออกห่างจากเซ็นเซอร์กล้องภาพที่ได้จะมีสีเข้มขึ้น
- กำหนดค่าสีให้แก่ผู้ใช้

```
byte[] convertDepthFrame(byte[] depthFrame16)
{
  for (int i16 = 0, i32 = 0;
       i16 < depthFrame16.Length && i32 < depthFrame32.Length;
       i16 += 2, i32 += 4)
  {
    int player = depthFrame16[i16] & 0x07;
    int realDepth = (depthFrame16[i16+1] << 5) | (depthFrame16[i16] >> 3);
    byte intensity = (byte)(255 - (255 * realDepth / 0x0fff));

    depthFrame32[i32 + RED_IDX] = 0;
    depthFrame32[i32 + GREEN_IDX] = 0;
    depthFrame32[i32 + BLUE_IDX] = 0;

    switch (player)
    {
      case 0:
        depthFrame32[i32 + RED_IDX] = (byte)(intensity / 2);
        depthFrame32[i32 + GREEN_IDX] = (byte)(intensity / 2);
        depthFrame32[i32 + BLUE_IDX] = (byte)(intensity / 2);
        break;
      case 1:
        depthFrame32[i32 + RED_IDX] = intensity;
        break;
      case 2:
        depthFrame32[i32 + GREEN_IDX] = intensity;
        break;
      case 3:
        depthFrame32[i32 + RED_IDX] = (byte)(intensity / 4);
        depthFrame32[i32 + GREEN_IDX] = (byte)(intensity);
        depthFrame32[i32 + BLUE_IDX] = (byte)(intensity);
        break;
      case 4:
        depthFrame32[i32 + RED_IDX] = (byte)(intensity);
        depthFrame32[i32 + GREEN_IDX] = (byte)(intensity);
        depthFrame32[i32 + BLUE_IDX] = (byte)(intensity / 4);
        break;
      case 5:
        depthFrame32[i32 + RED_IDX] = (byte)(intensity);
        depthFrame32[i32 + GREEN_IDX] = (byte)(intensity / 4);
        depthFrame32[i32 + BLUE_IDX] = (byte)(intensity);
        break;
      case 6:
        depthFrame32[i32 + RED_IDX] = (byte)(intensity / 2);
        depthFrame32[i32 + GREEN_IDX] = (byte)(intensity / 2);
        depthFrame32[i32 + BLUE_IDX] = (byte)(intensity);
        break;
      case 7:
        depthFrame32[i32 + RED_IDX] = (byte)(255 - intensity);
        depthFrame32[i32 + GREEN_IDX] = (byte)(255 - intensity);
        depthFrame32[i32 + BLUE_IDX] = (byte)(255 - intensity);
        break;
    }
  }
  return depthFrame32;
}
```

กระบวนการจัดการกับข้อมูลโครงกระดูก

เมื่อมีการใช้ nui_SkeletonFrameReady จะมีการดึงข้อมูลกระดูกและนำมาวาดเพื่อระบุเส้นโครงกระดูกกับข้อต่อ

```
void nui_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
{
  SkeletonFrame skeletonFrame = e.SkeletonFrame;
  int iSkeleton = 0;
  Brush[] brushes = new Brush[6];
  brushes[0] = new SolidColorBrush(Color.FromRgb(255, 0, 0));
  brushes[1] = new SolidColorBrush(Color.FromRgb(0, 255, 0));
  brushes[2] = new SolidColorBrush(Color.FromRgb(64, 255, 255));
  brushes[3] = new SolidColorBrush(Color.FromRgb(255, 255, 64));
  brushes[4] = new SolidColorBrush(Color.FromRgb(255, 64, 255));
  brushes[5] = new SolidColorBrush(Color.FromRgb(128, 128, 255));

  skeleton.Children.Clear();

  foreach (SkeletonData data in skeletonFrame.Skeletons)
  {
    if (SkeletonTrackingState.Tracked == data.TrackingState)
    {
      // Draw bones
      Brush brush = brushes[iSkeleton % brushes.Length];
      skeleton.Children.Add(getBodySegment(data.Joints, brush,
          JointID.HipCenter, JointID.Spine, JointID.ShoulderCenter,
          JointID.Head));
      skeleton.Children.Add(getBodySegment(data.Joints, brush,
          JointID.ShoulderCenter, JointID.ShoulderLeft, JointID.ElbowLeft,
          JointID.WristLeft, JointID.HandLeft));
      skeleton.Children.Add(getBodySegment(data.Joints, brush,
          JointID.ShoulderCenter, JointID.ShoulderRight, JointID.ElbowRight,
          JointID.WristRight, JointID.HandRight));
      skeleton.Children.Add(getBodySegment(data.Joints, brush,
          JointID.HipCenter, JointID.HipLeft, JointID.KneeLeft,
          JointID.AnkleLeft, JointID.FootLeft));
      skeleton.Children.Add(getBodySegment(data.Joints, brush,
          JointID.HipCenter, JointID.HipRight, JointID.KneeRight,
          JointID.AnkleRight, JointID.FootRight));
      // Draw joints
      foreach (Joint joint in data.Joints)
      {
        Point jointPos = getDisplayPosition(joint);
        Line jointLine = new Line();
        jointLine.X1 = jointPos.X - 3;
        jointLine.X2 = jointLine.X1 + 6;
        jointLine.Y1 = jointLine.Y2 = jointPos.Y;
        jointLine.Stroke = jointColors[joint.ID];
        jointLine.StrokeThickness = 6;
        skeleton.Children.Add(jointLine);
      }
    }
    iSkeleton++;
  }
}
```

จะมีการเก็บข้อมูลโครงกระดูกไว้ในตัวแปร skeletonFrame จากนั้นจะใช้ตัวแปร iSkeleton และ brush ในการวาดโครงร่างเส้นกระดูกและข้อต่อในแต่ละส่วน โดยจะวาดกระดูกก่อนที่จะวาดข้อต่อ และจะมีขั้นตอนดังต่อไปนี้

- จะระบุข้อต่อของกระดูกจาก Microsoft.Research.Kinect.Nui.JointsCollection
- ระบุสีที่ใช้ในการวาดเส้นโครงร่างกระดูก
- จะมีการเก็บค่าตำแหน่งกระดูกไว้ที่ตัวแปร JointID

ในส่วนของ getDisplayPosition จะเก็บตำแหน่งและส่งค่าไปเก็บที่ jointPos และจะส่งต่อไปที่ jointLine เพื่อใช้ในการวาดเส้น ข้อมูลโครงกระดูก , ข้อมูลภาพสี และข้อมูลความลึกจะมีการทำงานที่แตกต่างกัน ดังนั้นจึงต้องมีการทำให้ทั้ง 3 ส่วนนี้มีความสอดคล้องกัน ดังต่อไปนี้

- จะใช้ SkeletonEngine.SkeletonToDepthImage ในการแปลงพิกัดของโครงกระดูกในช่วง [-1.0, 1.0] เป็น [0.0-1.0] ของข้อมูลความลึก
- ในส่วนของ NuiCamera.GetColorPixelCoordinatesFromDepthPixel จะมีการเปลี่ยนภาพสีให้มีขนาด 320x240
- ในส่วนของ NuiCamera.GetColorPixelCoordinatesFromDepthPixel จะมีการเปลี่ยนภาพสีให้มีขนาด 640x480
- จะมีการเปลี่ยนแปลงขนาดของภาพสีเป็นขนาดของโครงกระดูกโดยการหาร โดยการนำค่า x หารด้วย 640 และนาผลลัพธ์ที่ได้มาคูณด้วยความกว้าง ส่วนค่า y หารด้วย 480 และนำผลลัพธ์ที่ได้มาคูณด้วยความสูง

```
private Point getDisplayPosition(Joint joint)
{
  float depthX, depthY;
  nui.SkeletonEngine.SkeletonToDepthImage(joint.Position,
      out depthX, out depthY);
  depthX = Math.Max(0, Math.Min(depthX * 320, 320));
  depthY = Math.Max(0, Math.Min(depthY * 240, 240));

  int colorX, colorY;
  ImageViewArea iv = new ImageViewArea();
  // only ImageResolution.Resolution640x480 is supported at this point
  nui.NuiCamera.GetColorPixelCoordinatesFromDepthPixel(
      ImageResolution.Resolution640x480, iv, (int)depthX, (int)depthY,
      (short)0, out colorX, out colorY);

  return new Point((int)(skeleton.Width * colorX / 640.0),
      (int)(skeleton.Height * colorY / 480));
}
```

2.4 Control Mouse Curser

จากการทำงานของการตรวจจับตำแหน่งของโครงกระดูกจากเซ็นเซอร์ของ Kinect จะได้ข้อมูลของโครงสร้างของกระดูก ซึ่งการระบุโครงสร้างของโครงกระดูกนั้น จะมีการระบุพิกัดหรือตำแหน่งของโครงกระดูกหรือข้อต่อส่วนต่างๆ เพื่อใช้ในการวาดโครงกระดูก โดยค่าตำแหน่งกระดูกจะถูกเก็บไว้ที่ตัวแปร JointID

การนำตำแหน่งของโครงกระดูกมาใช้ในการควบคุมเคอร์เซอร์เมาส์นั้น จะเป็นการนำค่าของตำแหน่งที่ตัวแปร JointID ระบุพิกัดของเคอร์เซอร์เมาส์ในระบบปฏิบัติการ ซึ่งจะใช้ตำแหน่งข้อต่อส่วนของมือขวาส่งค่าไปยังระบบปฏิบัติการ จะทำการสร้างตัวแปร JointRight เพื่อใช้ในการระบุมือขวาคือ

```
Joint jointRight = sd.Joints[JointID.HandRight];
```

หลังจากนั้นจะดึงสเกลของตำแหน่งมือออกมาจากตัวแปร JointID โดยจะสร้างตัวแปรที่ใช้เก็บค่าของตัวเลข 2 ตัวเพื่อนำมาเก็บค่าของตำแหน่งในพิกัดของ X และ Y ที่ได้จากตัวแปร jointRight

```
int cursorX, cursorY;

cursorX = (int)jointRight.Position.X;
cursorY = (int)jointRight.Position.Y;
```

เมื่อได้ตำแหน่งของมือในรูปแบบของตัวแปรที่เป็นตัวเลขแล้ว จะนำค่าตำแหน่งที่ได้ส่งไปยังระบบปฏิบัติการเพื่อทำการเลื่อนตำแหน่งของเคอร์เซอร์เมาส์ต่อไป การคลิกของเคอร์เซอร์เมาส์จะใช้ตัวแปรในรูปแบบจริงเท็จเช่น

```
bool leftclick;
```

เพื่อใช้ในการระบุการคลิกของเคอร์เซอร์เมาส์ โดยจะใช้การตรวจสอบเงื่อนไขเพื่อกำหนดค่าให้กับตัวแปร rightclick และส่งค่าให้กับระบบปฏิบัติการเพื่อทำงานต่อไป

2.4.1 SendInput

ฟังก์ชันหลักที่ใช้ในการส่งค่าเพื่อควบคุมการทำงานของเคอร์เซอร์เมาส์ในระบบปฏิบัติการ การใช้งานฟังก์ชัน SendInput จะต้องมีการดึงไลบรารี่ของระบบปฏิบัติการที่ชื่อว่า "user32.dll" ซึ่งเป็นไลบรารี่ที่มีฟังก์ชัน SendInput ที่ใช้ในการส่งค่าควบคุมการทำงานของเคอร์เซอร์เมาส์ด้วย โดยโครงสร้างของฟังก์ชัน SendInput นั้นจะอยู่ในรูปแบบดังนี้

```
uint SendInput(uint numInputs, Input[] inputs, int size);
```

- `uint numInputs` จำนวนของโครงสร้างของอาร์เรย์
- `Input[] inputs` อาร์เรย์หรือโครงสร้างของข้อมูลที่ส่งเข้าฟังก์ชัน
- `int size` ขนาดของข้อมูลที่เข้าสู่ฟังก์ชัน

2.4.2 SendMouseInput

เมธอดที่ใช้ในการรับค่าของพิกัดตำแหน่งที่ได้จากการดึงค่าจากโครงสร้างกระดูกนั้น จะใช้เมธอดที่เรียกว่า SendMouseInput ซึ่งจะมีโครงสร้างดังนี้

```
SendMouseInput(
int positionX,
int positionY,
int maxX,
int maxY,
bool leftDown)
```

- `int positionX` ค่าของตำแหน่งในพิกัดแกน X
- `int positionY` ค่าของตำแหน่งในพิกัดแกน X
- `int maxX` ค่าของขนาดในแกน X ของหน้าจอที่มากที่สุด
- `int maxY` ค่าของขนาดในแกน Y ของหน้าจอที่มากที่สุด
- `bool leftDown` ค่าของการคลิกของเคอร์เซอร์เมาส์

รูปแบบของโครงสร้างการส่งค่าในโปรแกรมหลักเป็นดังนี้

```
SendMouseInput(
cursorX, cursorY,
(int)SystemParameters.PrimaryScreenWidth,
(int)SystemParameters.PrimaryScreenHeight,
leftClick);
```

ค่า Flag ที่ใช้ในการกำหนดรูปแบบของเคอร์เซอร์เมาส์ เช่น การเคลื่อนที่ของเคอร์เซอร์เมาส์ การคลิกซ้าย หรือการคลิกขวา จะมีค่าคงที่ที่ถูกกำหนดขึ้นเพื่อให้ระบบปฏิบัติการทำงานตามการควบคุมของผู้ใช้ ซึ่งค่าหลักๆที่ใช้งานมีดังนี้

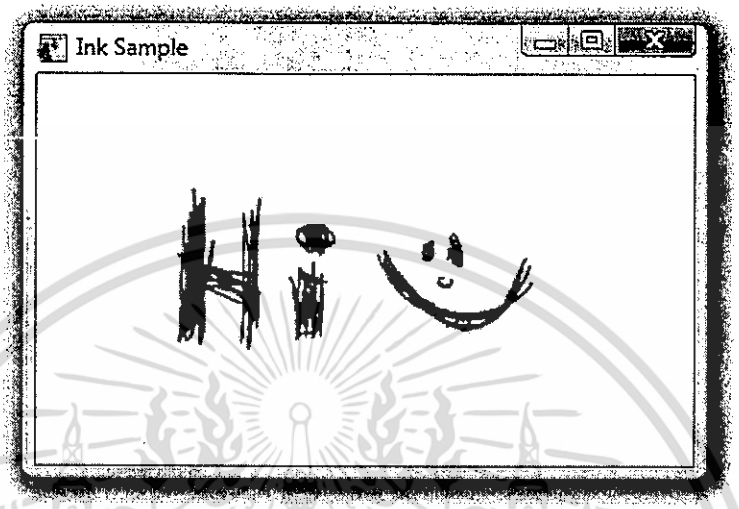
• MOUSEEVENTF_MOVE	0x0001	เคอร์เซอร์เมาส์เคลื่อนที่
• MOUSEEVENTF_LEFTDOWN	0x0002	คลิกซ้ายกดลง
• MOUSEEVENTF_LEFTUP	0x0004	คลิกซ้ายปล่อย
• MOUSEEVENTF_RIGHTDOWN	0x0008	คลิกขวากดลง
• MOUSEEVENTF_RIGHTUP	0x0010	คลิกขวาปล่อย

2.5 Recognition by Ink Analysis

การวาดหรือเขียนตัวอักษรโดยการใช้มือวาดแล้วแปลงจากการเขียนด้วยมือเป็นตัวอักษรที่เป็นเท็กซ์ที่มีรูปแบบตัวอักษรเป็นตัวพิมพ์ จะเป็นการใช้โครงสร้างการทำงานของไมโครซอฟท์ที่มีส่วนหนึ่งที่เรียกว่า Ink analysis ซึ่งมีการพัฒนาเครื่องมือนี้มาตั้งแต่ในระบบปฏิบัติการไมโครซอฟต์วินโดว์เอ็กซ์พี(Microsoft Windows XP) โดยจะเป็นการเปลี่ยนรูปแบบของลายมือเขียนให้อยู่ในรูปของตัวอักษรแบบพิมพ์ โดยมีการพัฒนาใช้กับแท็บเลตพีซีในยุคสมัยนั้นเพื่อให้การแก้ไขหรือปรับปรุงไฟล์งานเอกสารในลักษณะที่ไม่มีคีย์บอร์ด แต่จะใช้การเขียนบนแท็บแล็ตแล้วแปลงเป็นตัวอักษรแทน

2.5.1 Inkcanvas

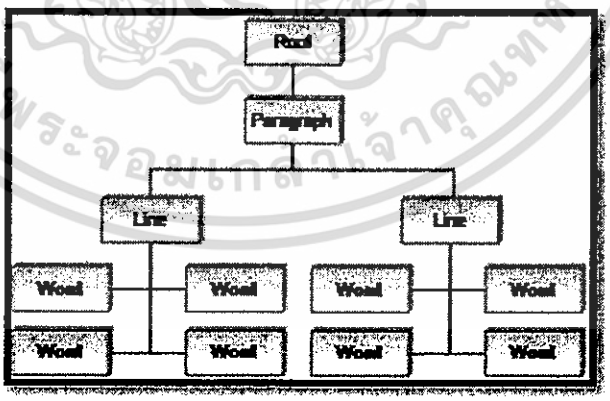
Windows Presentation Foundations หรือ WPF จะมีส่วนของ canvas ที่ใช้ในการวาดหรือลากเส้นและเส้นที่ลากได้นั้นจะเรียกว่า Stroke ซึ่ง Inkcanvas จะนำส่วนนี้ไปวิเคราะห์เพื่อที่จะทำการแปลงจากตัวอักษรที่เป็นลายมือให้อยู่ในรูปของตัวอักษรที่เป็นตัวพิมพ์



รูปที่ 2.9 โปรแกรมการแสดงผลการเขียนบน Ink canvas [10]

2.5.2 Ink Analysis

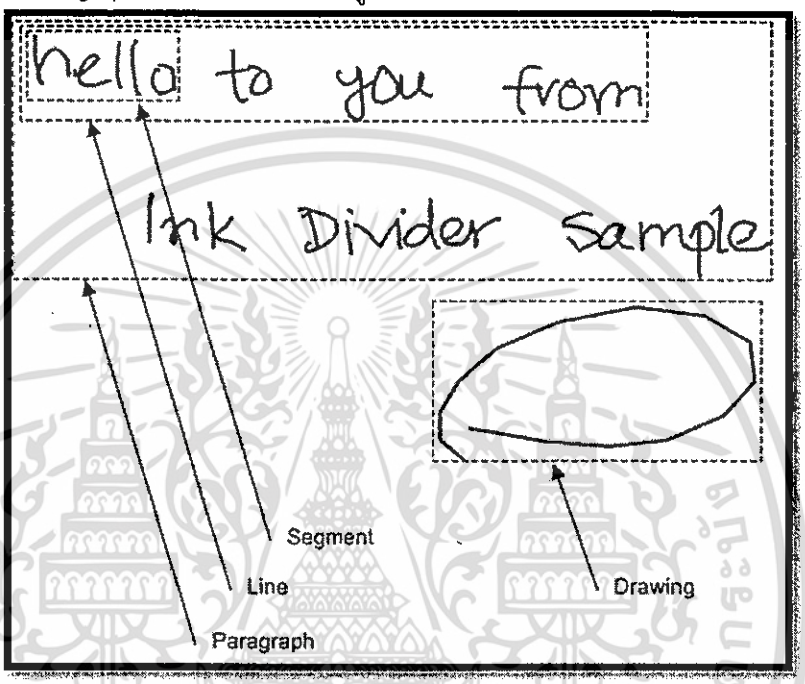
ในส่วนของการตรวจจับการแปลงตัวอักษรจะดึง Stroke จาก Inkcanvas มาทำการแปลงให้เป็นตัวอักษรให้เป็นตัวพิมพ์ โดยมีหลักการแปลงคือจะตรวจสอบทิศทางการวาดของเส้นที่ผู้ใช้วาดแล้วนำมาเปรียบเทียบกับตัวอักษรทั้งหมดที่มีอยู่ จากนั้น Ink Analysis จะเลือกตัวอักษรที่ดีที่สุดที่สอดคล้องกับทิศทางการวาดของผู้ใช้ โดยก่อนการแปลงจะต้องมีการแบ่งระดับว่าขอบเขตการแปลงทั้งหมดมีเท่าใด ดังรูปที่ 2.10



รูปที่ 2.10 ไดอะแกรมของการตรวจจับและการแปลงเป็นตัวอักษร [11]

ในการทำงานของ Ink Analysis จะมีความสามารถในการแยกแยะคำได้หลายระดับ คือ

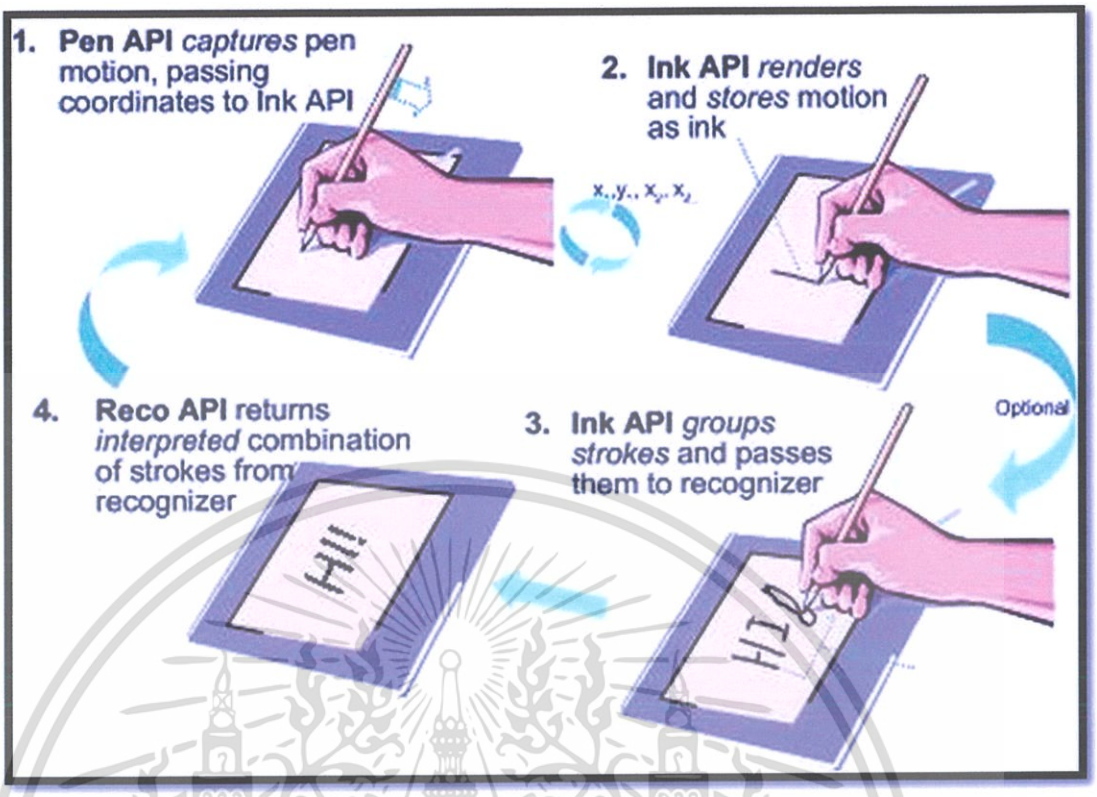
- Drawing เป็นเส้นที่สร้างขึ้นแล้วไม่มีความหมาย ไม่ต้องนำไปแปลง
- Segment/Word เป็น คำที่สามารถนำเข้าสู่กระบวนการแปลงได้
- Line กลุ่มของคำที่อยู่ในบรรทัดเดียวกันที่สามารถแยกเป็นแต่ละคำเพื่อแปลงเป็นตัวอักษรได้
- Paragraph ข้อความทั้งหมดที่อยู่บน Inkcanvas ที่สามารถแปลงได้(ไม่รวม Drawing)



รูปที่ 2.11 การแบ่งระดับของกลุ่มคำใน Inkcanvas [12]

สำหรับหลักการแปลง สามารถเปรียบเทียบกับการใช้ Tablet PC เพราะมีการเขียนลงบนจอแสดงผลเช่นเดียวกันกับ Inkcanvas ที่เขียนลงบนพื้นที่สำหรับเขียน

- อันดับแรกเมื่อมีการวาดเขียนลงบนพื้นที่ Inkcanvas จะตรวจจับได้ว่าการเคลื่อนที่ไปในทิศทางไหนโดยระบุเป็นค่าแกน x,y ตามระบบพิกัดสี่เหลี่ยม (Coordinate System) และเก็บข้อมูลการเคลื่อนที่นั้น โดยไม่ใช้การเก็บข้อมูลเป็นรูปภาพหรือ Pixel
- Inkcanvas จะจัดกลุ่มของข้อความว่าตัวอักษรใดคือคำเดียวกัน และตัวอักษรใดที่ไม่ใช่คำเดียวกันก็จะนำมาแยกออกจากกัน เพื่อเตรียมส่งข้อมูลดังกล่าวให้กับขั้นตอนการแปลงเป็นตัวอักษร
- สุดท้าย Inkcanvas จะทำการแปลงตัวอักษรที่เป็นลายมือที่ได้รับมาให้เป็นตัวอักษรที่เป็นตัวพิมพ์ ในส่วนของการวิเคราะห์นี้ทางไมโครซอฟท์เป็นผู้พัฒนา จึงไม่สามารถแก้ไขหรือเข้าไปศึกษาวิธีการทำงานโดยละเอียดได้ เพราะเป็นส่วนที่อยู่ในไลบรารี่ IACore.dll, IAWinFx.dll, IALoader.dll



รูปที่ 2.12 ลำดับการทำงานในการแปลงตัวอักษรของ Inkcanvas [13]

2.5.3 Gistafigure

เป็นฟังก์ชันการแปลงตัวแปรที่อยู่ในรูปของ Stringให้เป็นรูปภาพและวางลงบน Inkcanvas

```
GistaFigure figure = new GistaFigure(t.GetRecognizedString(), a, de, be);
```

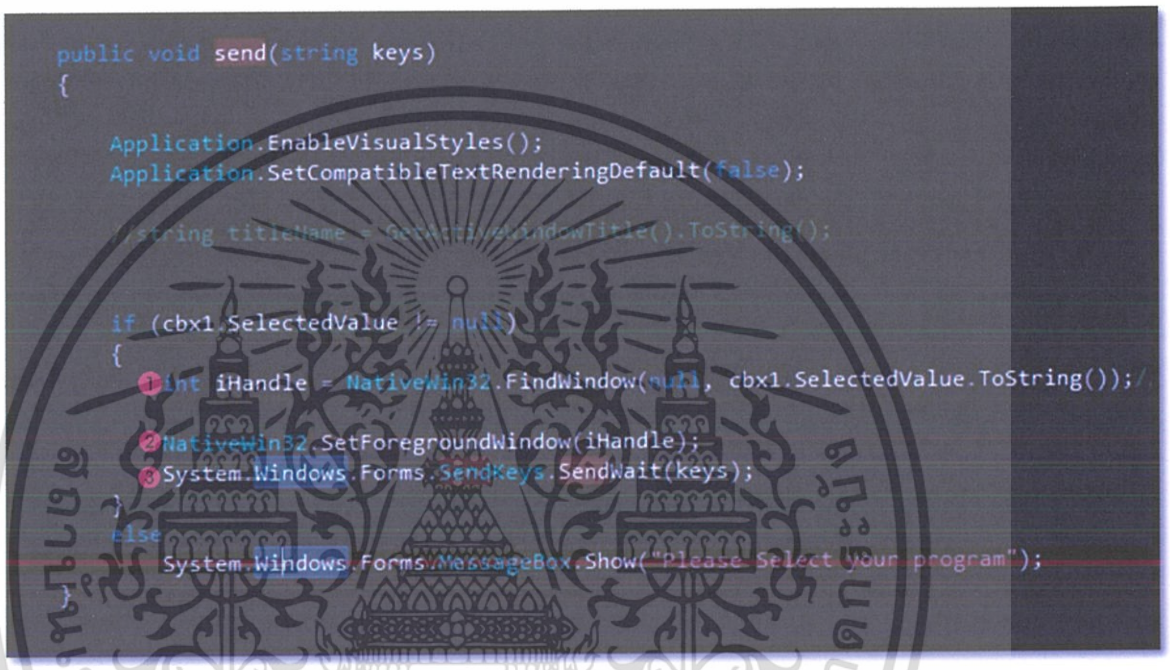
จะใช้ class GistaFigure ที่มีความสามารถในการรับตัวอักษรที่เป็นตัวพิมพ์มาแปลงให้อยู่ในรูปของรูปภาพที่แสดงบน InkCanvas ได้ โดยจะต้องส่งค่าต่างๆ ดังนี้

- t.GetRecognizedString() คือ String ที่ต้องการแปลง
- a คือ ตำแหน่งกึ่งกลางตัวอักษรที่รับเข้ามา
- de คือ ค่าขนาดของตัวอักษรที่รับเข้ามา
- be คือ ค่าสีของตัวอักษรที่ต้องการแสดง

2.6 Nativewin32

เป็นฟังก์ชันการทำงานเพื่อส่งค่าตัวอักษรที่ Inkcanvas แปลงได้แล้ว ไปสู่โปรแกรมเป้าหมายที่ได้เลือกไว้ โดยมีหลักการทำงานดังนี้

1. ฟังก์ชันนี้จะทำงานหน้าโปรแกรมทั้งหมดที่ทำงานอยู่ในขณะนั้น เพื่อรอให้ผู้ใช้เลือกโปรแกรมเป้าหมาย
2. ฟังก์ชันจะทำการค้นหาพื้นที่ในส่วนที่สามารถพิมพ์ได้ของโปรแกรมที่เลือกไว้
3. ส่งค่าตัวอักษรที่แปลงได้เข้าสู่โปรแกรมเป้าหมาย



```
public void send(string keys)
{

    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);

    //string titleName = GetActiveWindowTitle().ToString();

    if (cbx1.SelectedValue != null)
    {
        int iHandle = NativeWin32.FindWindow(null, cbx1.SelectedValue.ToString());

        NativeWin32.SetForegroundWindow(iHandle);
        System.Windows.Forms.SendKeys.SendWait(keys);
    }
    else
        System.Windows.Forms.MessageBox.Show("Please Select your program");
}
```

รูปที่ 2.13 ฟังก์ชันการทำงานของ Nativewin32

2.7 Control PowerPoint

ในการควบคุมการทำงานของโปรแกรมการนำเสนอ จะเป็นการทำงานร่วมกัน 2 ระบบ คือ ระบบตรวจจับโครงกระดูก โดยจะเน้นในส่วนของบริเวณมือและศีรษะ และระบบการส่งสัญญาณการควบคุมโปรแกรมการนำเสนอ โดยมีวิธีการทำงานดังนี้

- ตรวจจับพิกัดของมือซ้ายเมื่อเทียบกับศีรษะว่ามืออยู่เยื้องไปทางซ้ายของศีรษะประมาณ 45 เซนติเมตรหรือไม่ ถ้าใช่ให้เลื่อนสไลด์ย้อนกลับ
- ตรวจจับพิกัดของมือขวาเมื่อเทียบกับศีรษะว่ามืออยู่เยื้องไปทางขวาของศีรษะประมาณ 45 เซนติเมตรหรือไม่ ถ้าใช่ให้เลื่อนสไลด์ไปข้างหน้า

```
if (rightHand.Position.X > head.Position.X + 0.45)
{
    if (!isBackGestureActive && !isForwardGestureActive)
    {
        isForwardGestureActive = true;
        System.Windows.Forms.SendKeys.SendWait("{Right}");
    }
}
else
{
    isForwardGestureActive = false;
}

if (leftHand.Position.X < head.Position.X - 0.45)
{
    if (!isBackGestureActive && !isForwardGestureActive)
    {
        isBackGestureActive = true;
        System.Windows.Forms.SendKeys.SendWait("{Left}");
    }
}
else
{
    isBackGestureActive = false;
}
```

รูปที่ 2.14 บริเวณการทำงานในการเลื่อนสไลด์

และเมื่อผู้ใช้ต้องการย่อขยายหน้าต่างโปรแกรม มีการทำงานดังนี้

- เมื่อผู้ใช้ยกมือขวาขึ้นเหนือศีรษะ จะเข้าเงื่อนไขแรก คือ ค่า rightHand.Position.Y > head.position.Y โปรแกรมจะสั่งให้ย่อโปรแกรม(ใช้ในเวลาที่ต้องการนำเสนอผลงาน)
- เมื่อผู้ใช้ยกมือซ้ายขึ้นเหนือศีรษะ จะเข้าเงื่อนไขที่สอง คือ ค่า leftHand.Position.Y > head.position.Y โปรแกรมจะสั่งให้ย่อโปรแกรม(ใช้ในเวลาที่ต้องการนำเสนอผลงาน)

```
if (rightHand.Position.Y > head.Position.Y)
{
    if (!isleup && !isriup)
    {
        isriup = true;
        this.WindowState = System.Windows.WindowState.Normal;
    }
}
else
{
    isriup = false;
}

if (leftHand.Position.Y > head.Position.Y)
{
    if (!isleup && !isriup)
    {
        isleup = true;
        this.WindowState = System.Windows.WindowState.Minimized;
    }
}
```

รูปที่ 2.15 บริเวณการทำงานในการย่อ-ขยายหน้าจอ

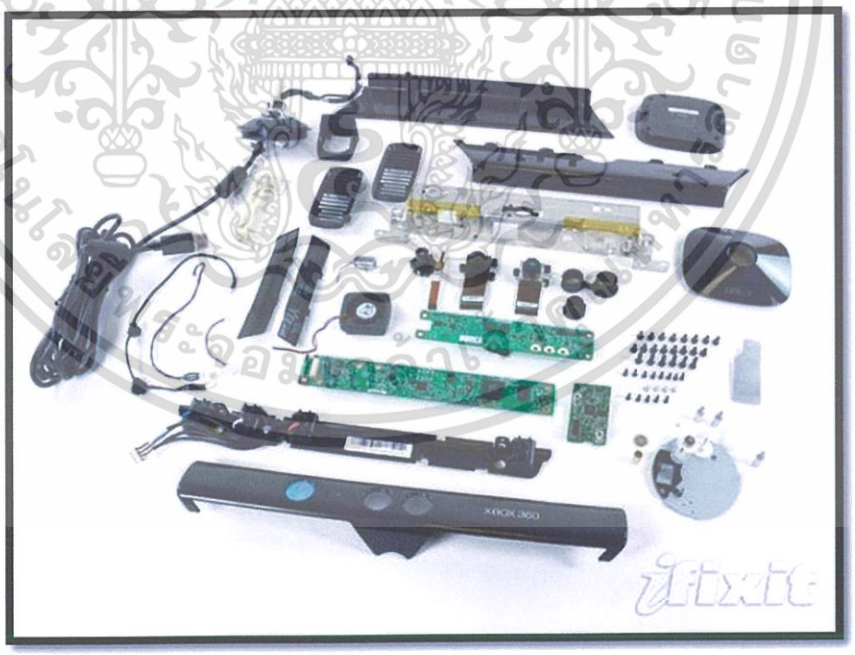
บทที่ 3
การออกแบบ

3.1 อุปกรณ์ฮาร์ดแวร์

3.1.1 กล้อง Kinect

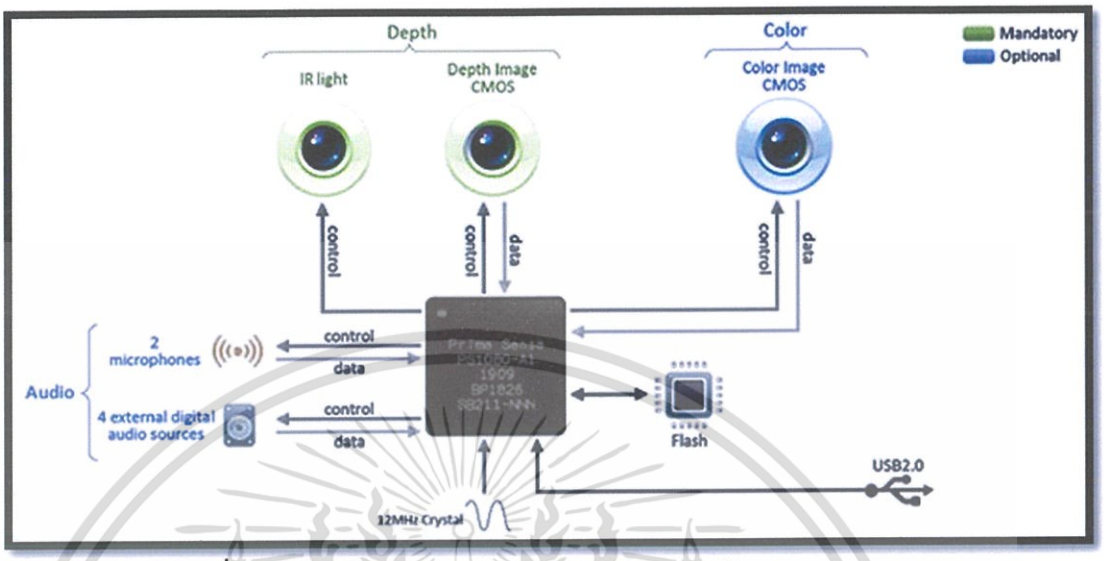
กล้อง Kinect เป็นอุปกรณ์ที่ใช้รับภาพและข้อมูลความลึกจากตัวผู้ใช้ แสดงได้ดังรูปที่ 3.1 และอุปกรณ์ภายในกล้อง Kinect แสดงได้รับรูปที่ 3.2

รูปที่ 3.1 อุปกรณ์กล้อง Kinect พร้อมสาย USB Cable [14]



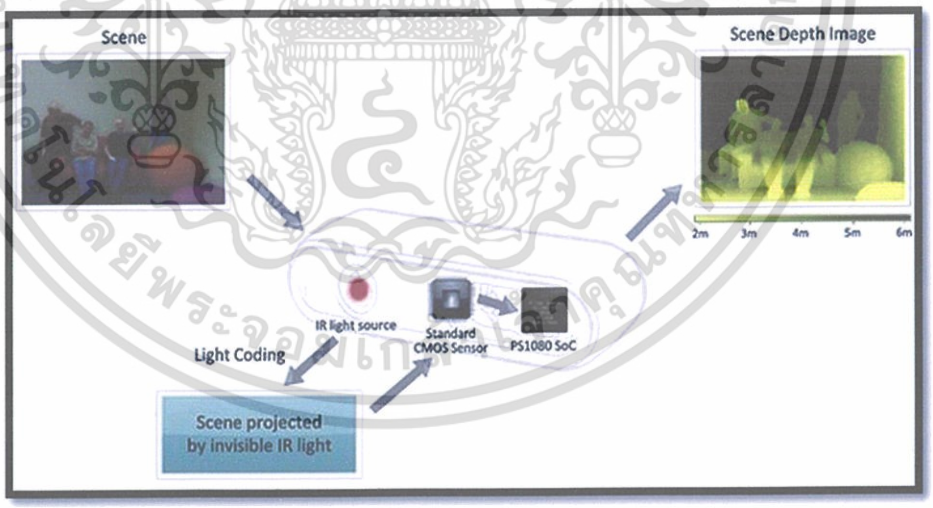
รูปที่ 3.2 อุปกรณ์ภายในกล้อง Kinect [15]

ไดอะแกรมการทำงานภายในของกล้อง Kinect แสดงได้ดังรูปที่ 3.3 ซึ่งมีชิพ PS1080 ที่เป็นระบบหลายการรับรู้ ทำหน้าที่ในการทำให้ภาพความลึก ภาพสี และสตรีมเสียงนั้นตรงกัน แล้วส่งผ่านข้อมูลต่างไปยังคอมพิวเตอร์ผ่านสาย USB โดยอัลกอริทึมต่างๆจะทางานบนชิพ PS1080



รูปที่ 3.3 ไดอะแกรมแสดงการทำงานภายในของกล้อง Kinect [16]

ไดอะแกรมแสดงการทำงานภายนอกของกล้อง Kinect แสดงได้ดังรูปที่ 3.4 โดยมีกล้องบันทึกภาพ ซึ่งจะให้ข้อมูลเป็นภาพสี และกล้องการจับภาพด้วยแสงอินฟราเรด ซึ่งจะให้ข้อมูลความลึก จากนั้นจะนำอินพุตเหล่านี้ไปประมวลผลด้วยชิพ PS1080 แสดงผลออกมาเป็นภาพข้อมูลความลึก

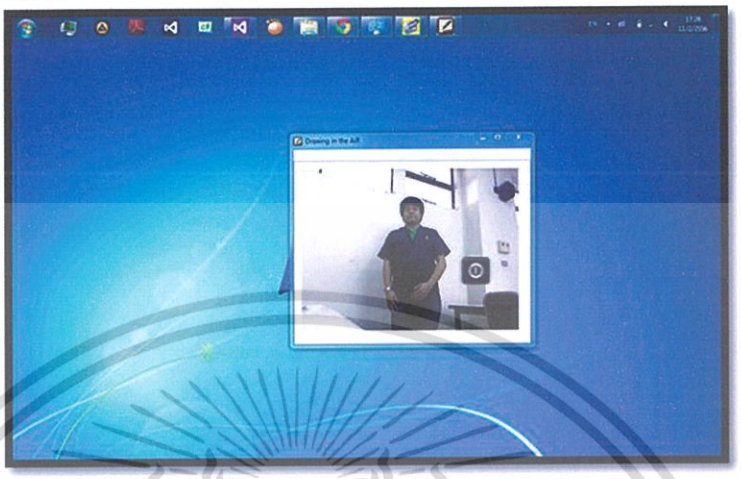


รูปที่ 3.4 ไดอะแกรมแสดงการทำงานภายนอกของกล้อง Kinect [17]

3.2 โปรแกรมซอฟต์แวร์

3.2.1 อินเตอร์เฟสสำหรับเริ่มต้นโปรแกรม

อินเตอร์เฟสนี้จะแสดงภาพการเริ่มต้นโปรแกรมแสดงได้ดังรูปที่ 3.5 ผู้ใช้จะต้องกดไอคอนสีแดงเพื่อเข้าโปรแกรม



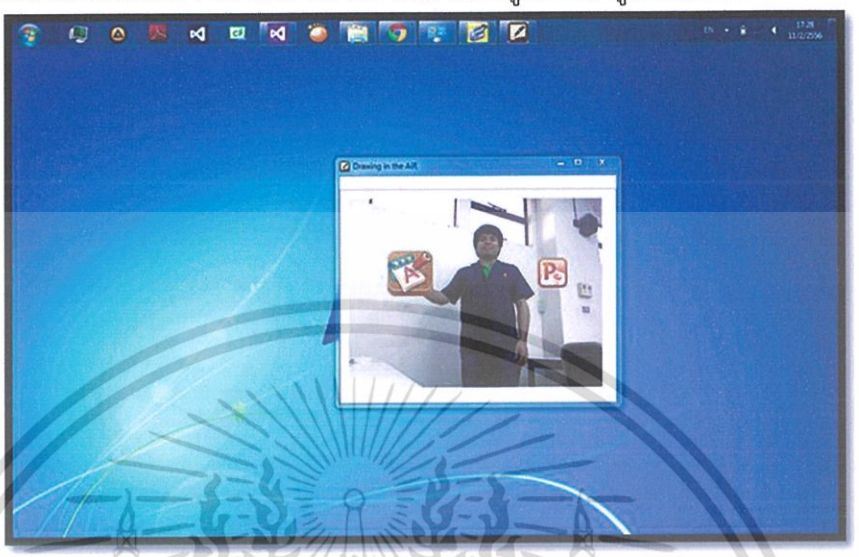
รูปที่ 3.5 อินเตอร์เฟสหน้าแรกที่รอผู้ใช้มาเปิดโปรแกรม

เมื่อผู้ใช้วางมือไว้ที่ไอคอน จะเปลี่ยนเป็นสีเขียวเพื่อแสดงว่าได้เขาใช้โปรแกรมแล้ว แสดงได้ดังรูปที่ 3.6

รูปที่ 3.6 อินเตอร์เฟสหน้าแรกเมื่อผู้ใช้ต้องการเปิดโปรแกรม

3.2.2 อินเตอร์เฟสสำหรับเลือกโหมดการทำงาน

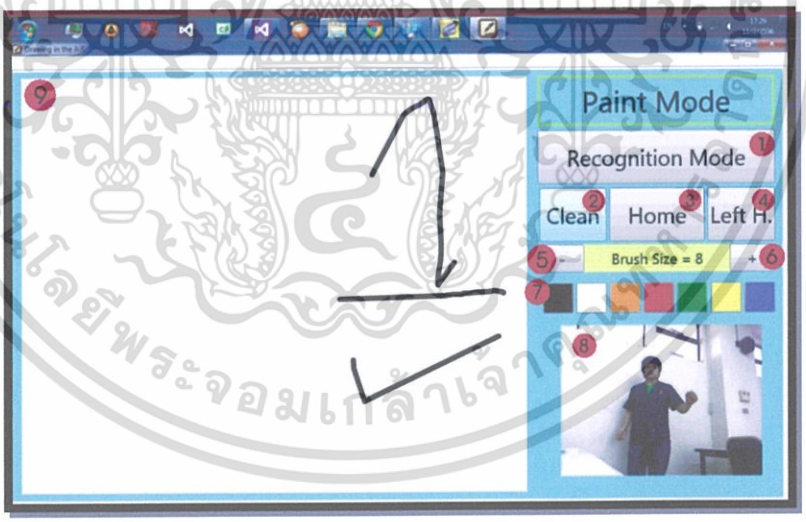
เนื่องจากการทำงานของโปรแกรมจะแบ่งออกเป็น 2 โหมดคือ 1) โหมดควบคุมการนำเสนอ 2) โหมดวาดเขียนและโหมดการแปลงตัวอักษร อินเตอร์เฟสนี้จึงออกแบบมาให้ผู้ใช้เลือกว่าจะควบคุมการนำเสนอหรือจะวาดเขียนตัวอักษร แสดงได้ดังรูปที่ 3.7 ผู้ใช้ได้เลือกโหมดวาดเขียน



รูปที่ 3.7 ภาพผู้ใช้เลือกเข้าไปในโปรแกรมวาดเขียน

3.2.3 อินเตอร์เฟสโปรแกรมวาดเขียน

หลังจากผู้ใช้ได้เลือกโหมดวาดเขียนแล้ว ผู้ใช้สามารถวาดเขียนตัวอักษรกลางอากาศได้โดยวาดในพื้นที่สีขาว ซึ่งแสดงได้ดังรูปที่ 3.8

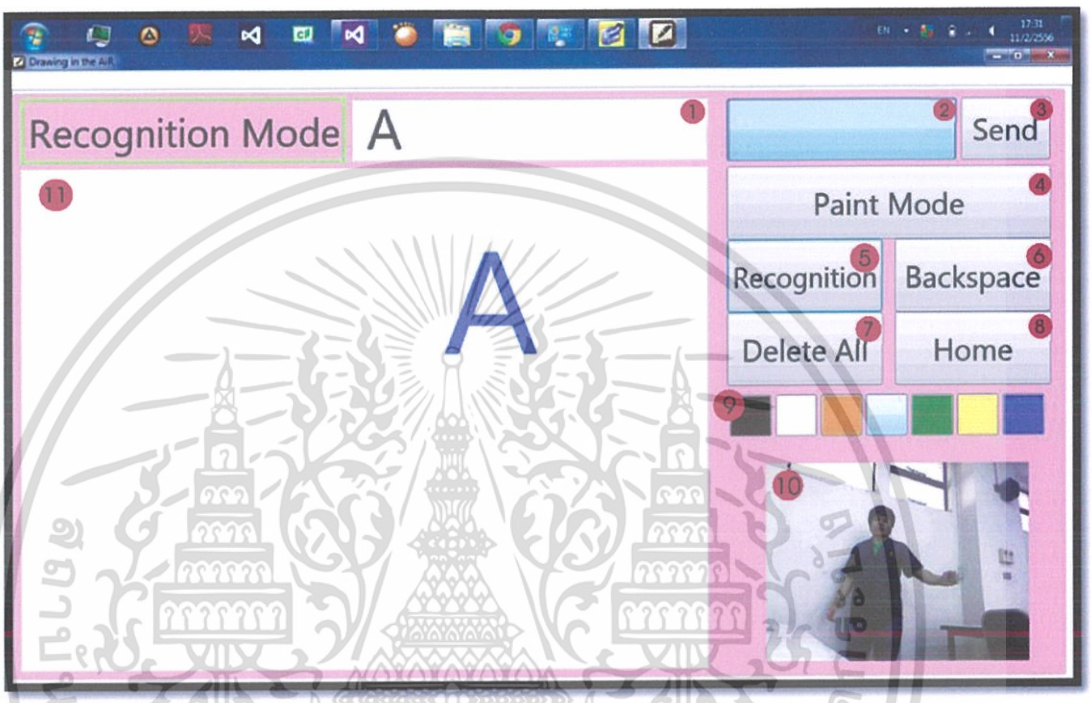


1.ปุ่มเข้าสู่โปรแกรมรู้จำอักษร
2.ปุ่มล้างพื้นที่วาดรูป
3.ปุ่มกลับไปหน้าเริ่มต้นโปรแกรม
4.ปุ่มเปลี่ยนมือซ้าย-ขวา
5.ปุ่มลดขนาดแปรง
6.ปุ่มเพิ่มขนาดแปรง
7.ปุ่มเปลี่ยนสีแปรง
8.หน้าต่างแสดงรูปผู้ใช้ที่กล้องถ่ายได้
9.พื้นที่วาดรูป

รูปที่ 3.8 โปรแกรมวาดเขียน

3.2.4 อินเตอร์เฟสโปรแกรมรู้จำและส่งอักษร

ผู้ใช้สามารถวาดเขียนตัวอักษร(ลายมือ)กลางอากาศแล้วจะถูกแปลงเป็นตัวอักษร(ตัวพิมพ์) ได้โดยกดปุ่มหมายเลข 5 หลังจากที่ทำการวาดเส้นเสร็จเสร็จแล้ว และสามารถลบ(Backspace) โดยกดที่ปุ่มหมายเลข 6 หรือส่งตัวอักษรไปใส่ไว้ในโปรแกรมดัดแปลงข้อความ(TextEditor) โดยทำการเลือกโปรแกรมที่ต้องการส่งในลิสต์บ็อกต์หมายเลข 2 และกดที่ปุ่มหมายเลข 3 เพื่อส่งตัวอักษรออกจากโปรแกรม

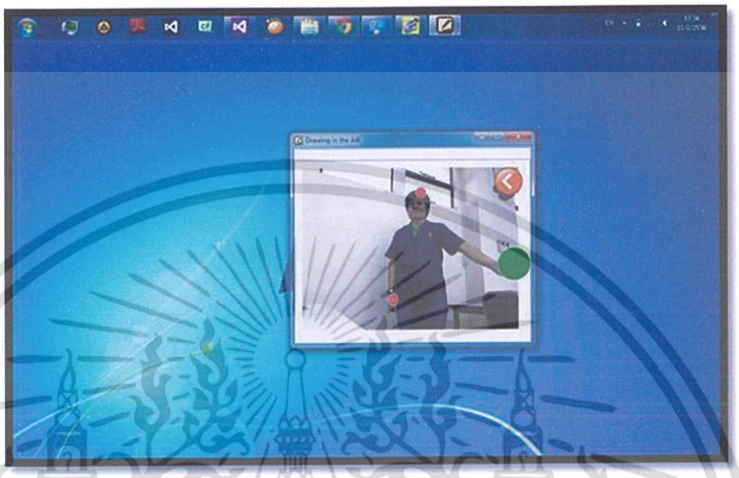


1.แถบรับข้อความที่ผู้ใช้วาด
2.ปุ่มเลือกโปรแกรมที่ส่งส่งตัวอักษรออกไป
3.ปุ่มส่งตัวอักษร
4.ปุ่มเข้าสู่โปรแกรมวาดรูป
5.ปุ่มแปลงจากลายมือเป็นตัวพิมพ์
6.ปุ่มลบตัวอักษรทีละตัว
7.ปุ่มล้างพื้นที่วาดตัวอักษรและข้อความ
8.ปุ่มกลับไปหน้าเริ่มต้นโปรแกรม
9.ปุ่มเปลี่ยนสีแปรง
10.หน้าต่างแสดงรูปผู้ใช้ที่กล้องถ่ายได้
11.พื้นที่วาดรูป

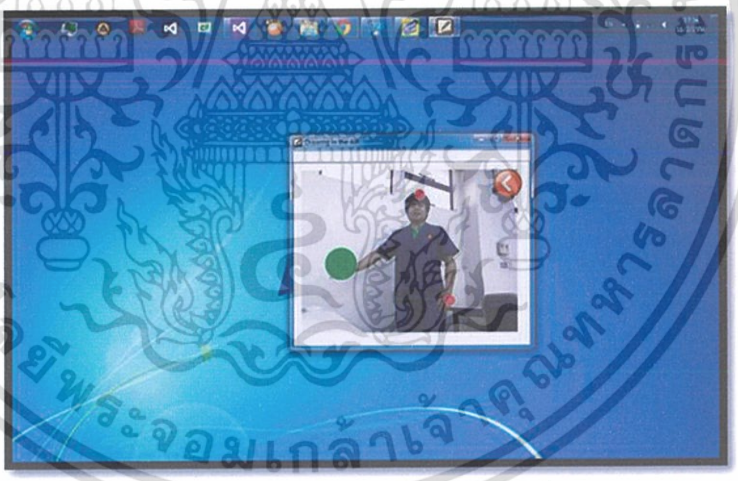
รูปที่ 3.9 โปรแกรมรู้จำอักษร

3.2.5 อินเตอร์เฟสโปรแกรมควบคุมการนำเสนอ

จากรูปที่ 3.7 ถ้าผู้ใช้เลือกโหมดการทำงานเป็นโหมดควบคุมการนำเสนอ อินเตอร์เฟสจะเปลี่ยนเป็นรูปที่ 3.10 โดยที่ผู้ใช้สามารถแกว่งมือซ้ายไปทางซ้ายเพื่อเลื่อนการนำเสนอไปหน้าที่แล้วดังรูปที่ 3.11 แกว่งมือขวาไปด้านขวาเพื่อเลื่อนการนำเสนอไปหน้าถัดไป ยกมือซ้ายขึ้นระดับหูเพื่อซ่อนหน้าต่างโปรแกรมดังรูปที่ 3.12 และยกมือขวาขึ้นระดับหูเพื่อเรียกโปรแกรมกลับมาสู่หน้าจอแสดงได้ดังรูปที่ 3.13

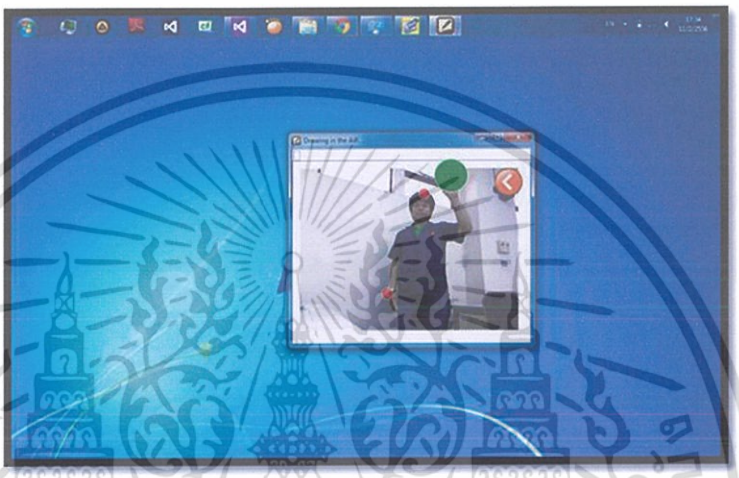


รูปที่ 3.10 ผู้ใช้ต้องการเลื่อนไปหน้าถัดไป



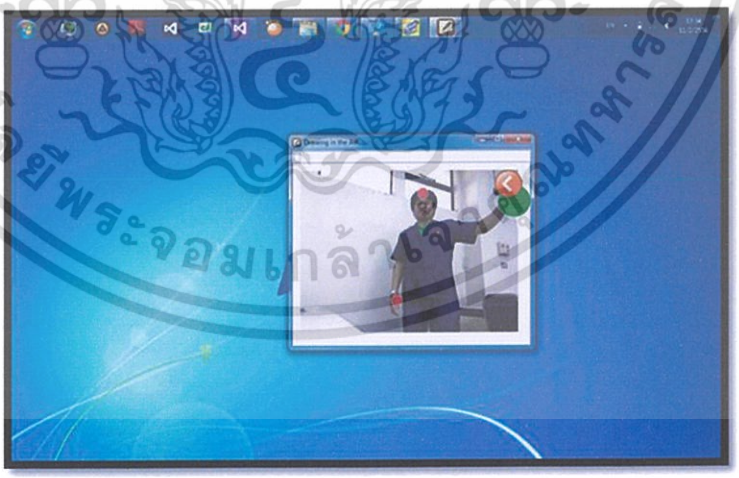
รูปที่ 3.11 ผู้ใช้ต้องการเลื่อนไปหน้าที่แล้ว

รูปที่ 3.12 ผู้ใช้ต้องการซ่อนโปรแกรม



รูปที่ 3.13 ผู้ใช้ต้องการเรียกโปรแกรมกลับมาสู่หน้าจอ

ถ้าผู้ใช้ต้องการกลับสู่หน้าเมนูหลัก ทำได้โดยวางมือขวาไว้ตามรูปที่ 3.14 เพื่อกลับสู่หน้าเมนู

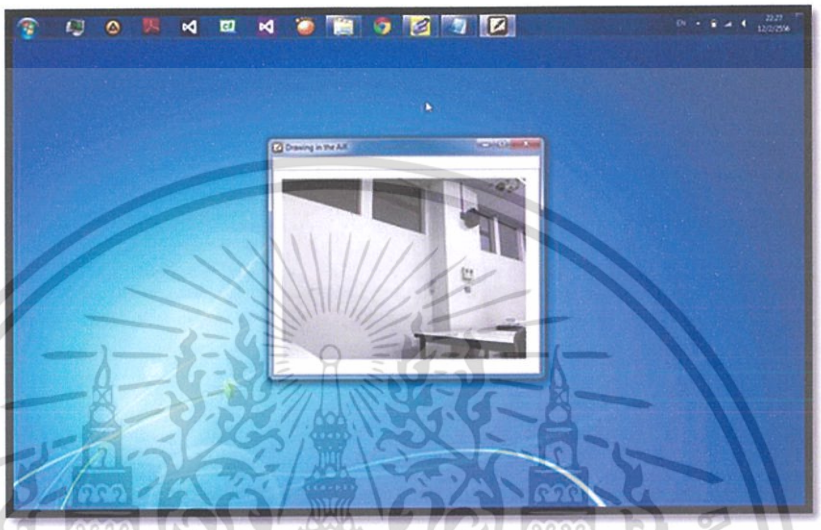


รูปที่ 3.14 ผู้ใช้ต้องการกลับไปสู้หน้าเมนูหลัก

บทที่ 4
ผลการทดลอง

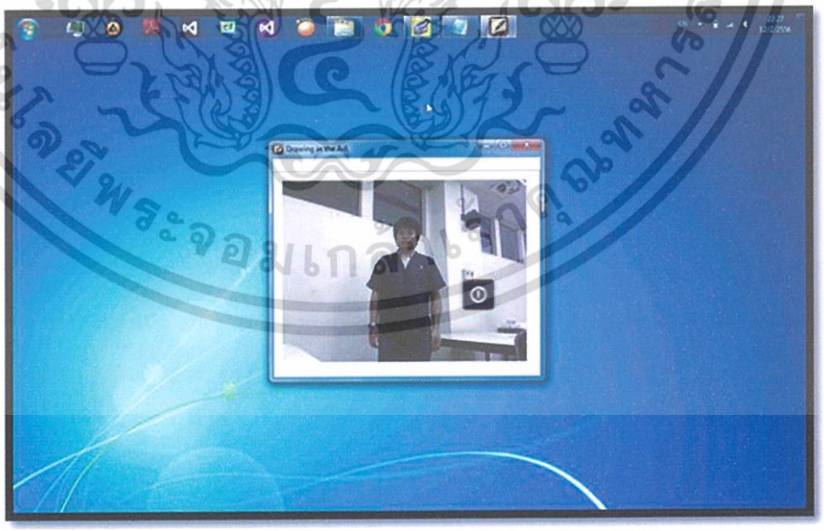
4.1 ผลการทำงานในช่วงเมื่อเริ่มต้นโปรแกรม

จากการทำการทดลองการใช้งานโปรแกรมในส่วนเริ่มต้นโปรแกรมนั้นทุกอย่างสามารถแสดงผลได้ตามที่วางแผนไว้โดยที่รูปที่ 4.1 แสดงหน้าต่างของโปรแกรมโดยจะมีแค่ภาพจากกล้องธรรมดา



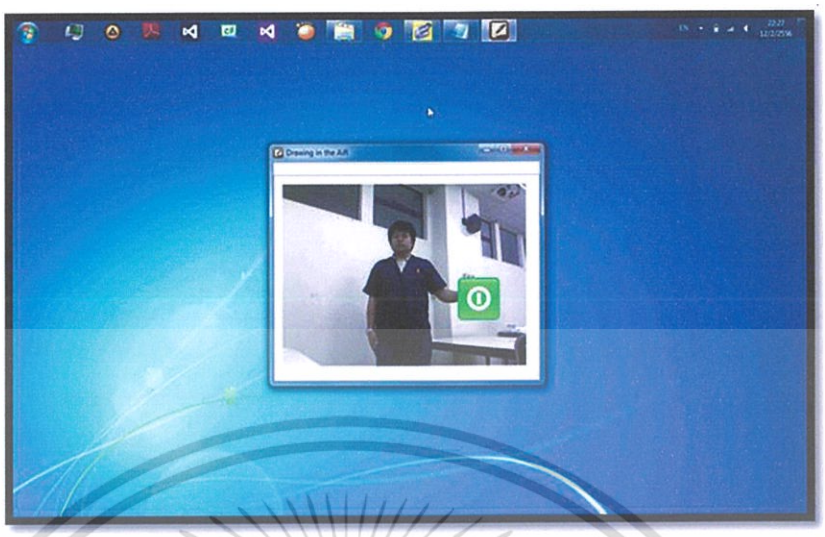
รูปที่ 4.1 หน้าต่างของโปรแกรมเมื่อไม่มีการตรวจจับคน

จากรูปที่ 4.1 โปรแกรมจะยังไม่มีปุ่มหรืออินเตอร์เฟสแสดงออกมา จนกว่ากล้องจะมาสารถตรวจจับคนได้ จึงจะแสดงไอคอนออกมา ดังรูปที่ 4.2



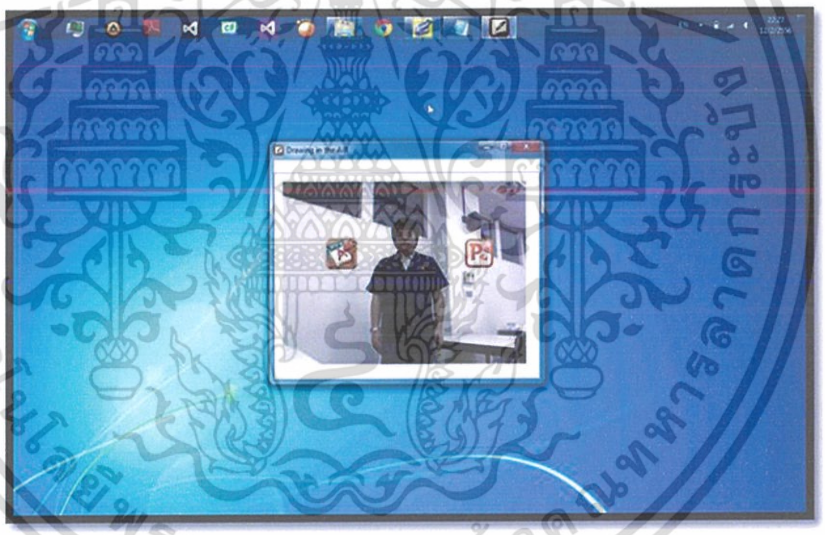
รูปที่ 4.2 หน้าต่างโปรแกรมเมื่อสามารถตรวจจับคนและแสดงไอคอนออกมา

จากนั้นเมื่อนำมือไปวางตรงไอคอน  สีของไอคอนจะเปลี่ยนแปลงไปเป็นสีเขียวดังรูปที่ 4.3



รูปที่ 4.3 หน้าต่างแสดงไอคอนสีเขียวพร้อมที่จะเข้าสู่โปรแกรมหลัก

เมื่อวางมือไว้ที่ไอคอนประมาณ 2 วินาที จะเข้าสู่หน้าเมนูหลัก ดังรูป 4.4



รูปที่ 4.4 หน้าต่างแสดงหน้าเมนูหลักของโปรแกรม

ในส่วนของหน้าต่างหลักนี้ จะมีไอคอนอยู่ 2 ไอคอนคือ

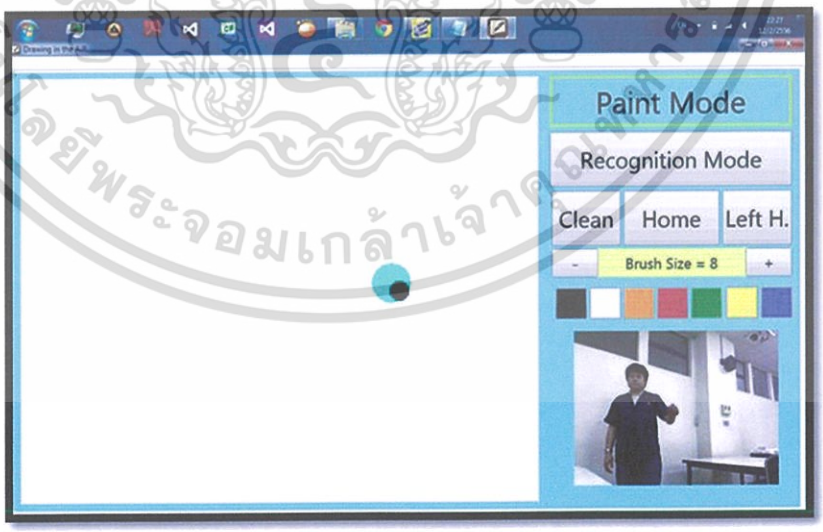
1.  ไอคอนที่เข้าสู่โปรแกรมการควบคุมการนำเสนอ
2.  ไอคอนที่เข้าสู่โปรแกรมวาดภาพและโปรแกรมการรู้จำตัวอักษร

4.2 ผลการทำงานในโหมดโปรแกรมวาดภาพ

จากการทำการทดลองการใช้งานโปรแกรมในส่วนของการวาดภาพนั้น สามารถวาดภาพโดยใช้มือขวาควบคุมการวาดได้สะดวกแต่ต้องระวังไม่ให้มือของเราเคลื่อนที่ผ่านลำตัว เพราะตัวกล้องจะสับสนระหว่างมือกับลำตัว (ซึ่งแก้ไขได้โดยการเปลี่ยนเวอร์ชันการทำงานของกล้อง Kinect ให้เป็นเวอร์ชัน 1.6 จะสามารถเคลื่อนที่ผ่านลำตัวได้ดีขึ้น) และสามารถใช้งานปุ่มต่างๆได้สะดวกเพราะมีความใหญ่กว่าปุ่มทั่วไปจึงสามารถสังเกตเห็นได้แม้ยืนอยู่ใกล้หน้าจอ ผู้ใช้สามารถเข้าสู่โปรแกรมโดยนำมือไปวางที่ไอคอนทางด้านซ้ายเป็นเวลา 2 วินาที ไอคอนจะมีขนาดใหญ่ขึ้น ดังรูปที่ 4.5

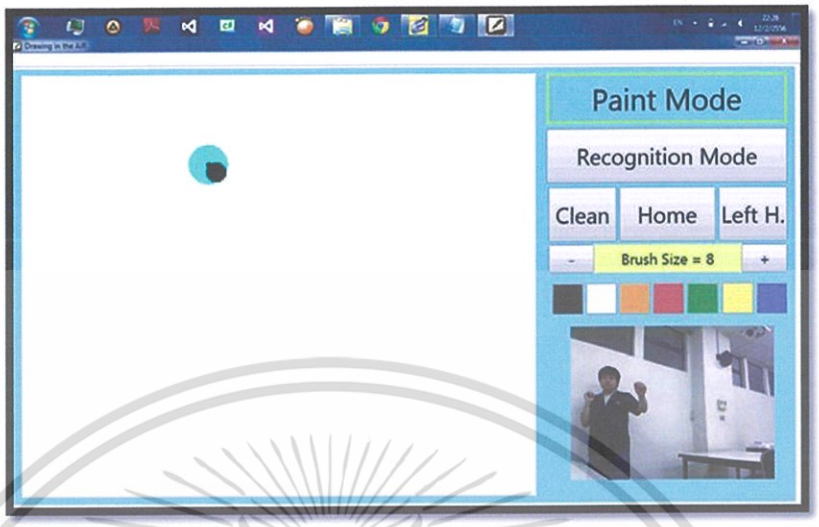
รูปที่ 4.5 หน้าต่างแสดงการวางมือไว้ที่ไอคอนทางด้ายซ้ายเพื่อเข้าสู่โปรแกรมวาดภาพและโปรแกรมการรู้จำตัวอักษร

เมื่อวางมือไว้ครบ 2 วินาที จะเข้าสู่หน้าโปรแกรมในส่วนของ Paint Mode ดังรูปที่ 4.6



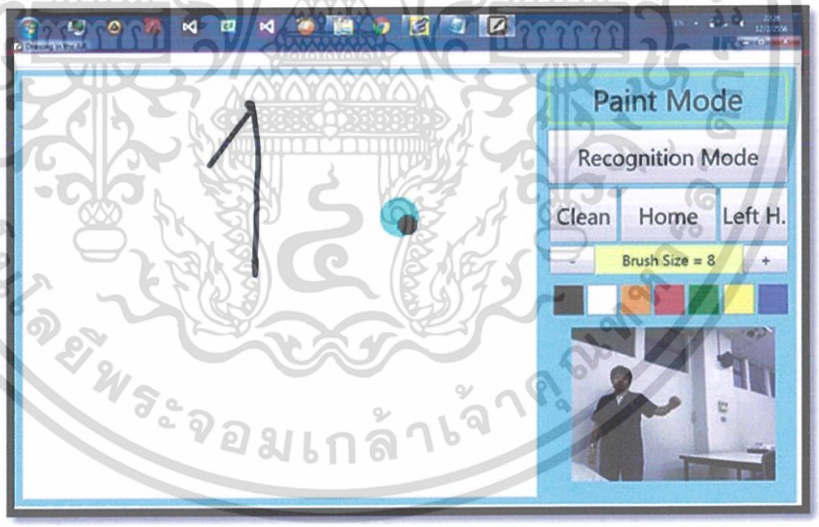
รูปที่ 4.6 แสดงหน้าต่างของโปรแกรม Paint Mode

ส่วนการใช้งานมีหลักการง่ายๆคือ มือขวาใช้ควบคุมเคอร์เซอร์เมาส์ของระบบปฏิบัติการ และใช้มือซ้ายในการคลิกเมาส์ โดยการยกมือขึ้นมาไว้สูงในระดับหัวไหล่ข้างตัว ดังรูปที่ 4.7



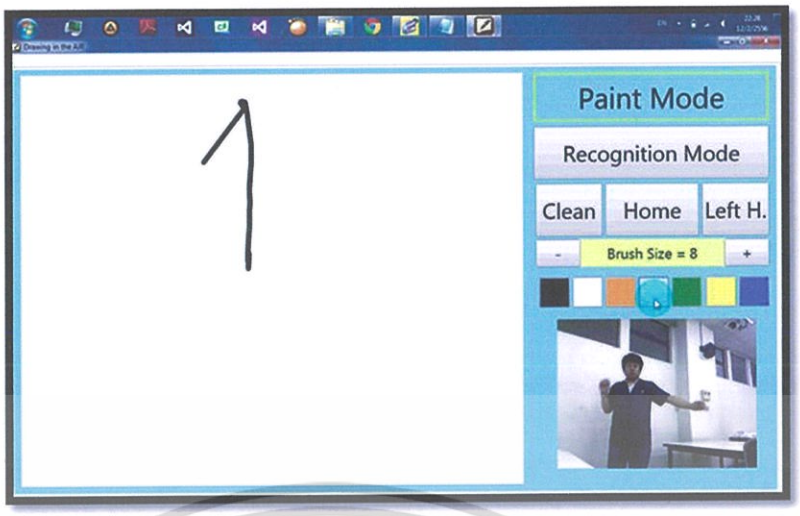
รูปที่ 4.7 ภาพแสดงการคลิกเมาส์โดยการยกมือซ้ายขึ้น

จากรูปจะเห็นว่าเมื่อทำการยกมือซ้ายขึ้น จะเป็นการสั่งการคลิกเมาส์ แล้วจึงใช้มือขวาเลื่อนเพื่อทำการวาดเส้นลงบนกรอบในพื้นที่วาดรูป เมื่อทำการเขียนเสร็จจึงนำมือซ้ายลง จะได้รูปเส้นตามที่เขียน ดังรูปที่ 4.8



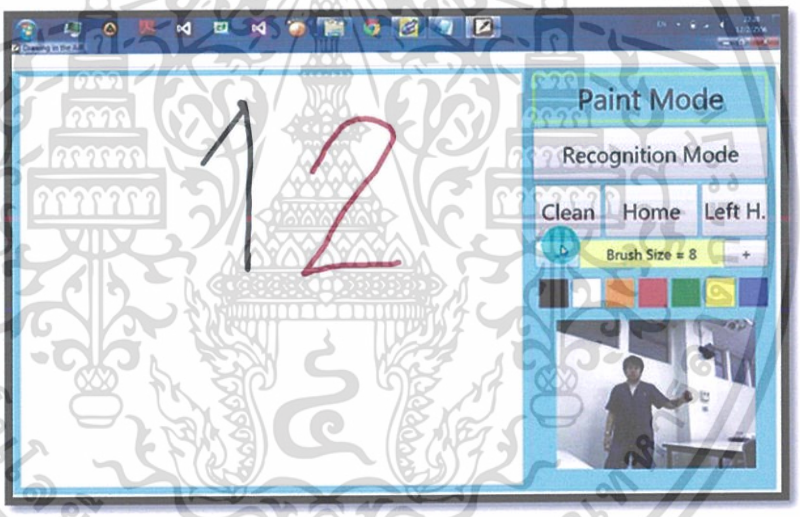
รูปที่ 4.8 ภาพแสดงเส้นจากการวาดด้วยมือ

จากภาพจะได้เส้นที่เกิดขึ้นจากการเขียนเส้นด้วยมือตามที่ผู้ใช้ต้องการ นอกจากนั้นแล้วผู้ใช้ยังสามารถเลือกสีที่จะใช้เขียนได้ด้วย โดยการนำเมาส์มาคลิกบริเวณแถบสีเหนือภาพจากกล้อง ดังรูปที่ 4.9



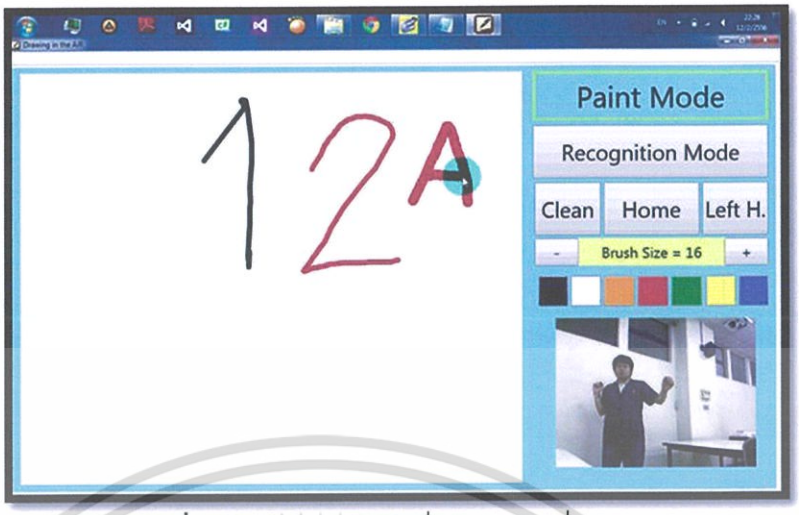
รูปที่ 4.9 แสดงการคลิกเพื่อเลือกสีที่ใช้ในการวาดเส้น

จากภาพจะเป็นการคลิกเพื่อเลือกสีแดง แล้วจึงทำการวาดเส้นลงบนพื้นที่วาดรูปได้ จะได้เส้นตามที่ผู้ใช้ต้องการวาด ดังรูปที่ 4.10



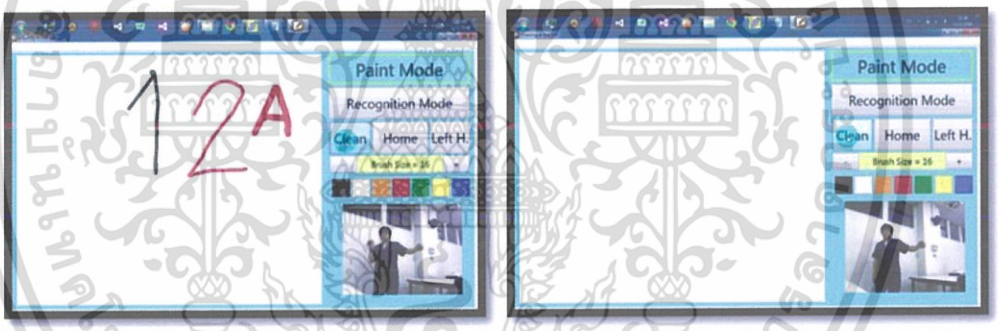
รูปที่ 4.10 ภาพแสดงเส้นการวาดเส้นด้วยสีแดง

ถ้าหากผู้ใช้ต้องการลดหรือเพิ่มขนาดของเส้นที่ใช้วาดก็สามารถทำได้โดยการคลิกบริเวณปุ่มบวกหรือปุ่มลบได้ และจะมีการแสดงสถานะของขนาดของเส้นในปัจจุบัน การคลิกที่ปุ่มบวกหรือปุ่มลบนี้จะทำการเพิ่มหรือลดค่าขนาดลงทีละ 2 พิกเซล แต่ผู้ใช้จะไม่สามารถลดค่าได้ต่ำกว่า 2 พิกเซลได้ และเมื่อผู้ใช้เลือกขนาดเรียบร้อย ผู้ใช่สามารถวาดเส้นลงบนพื้นที่วาดรูปและจะได้ขนาดเส้นตามที่ผู้ใช้ต้องการ ดังรูปที่ 4.11



รูปที่ 4.11 ภาพแสดงเส้นที่ผู้ใช้เลือกเปลี่ยนขนาด

จากภาพผู้ใช้ได้เลือกขนาดเส้นที่ 16 แล้วทำการวาดเส้นลงบนพื้นที่วาดรูป จะได้เส้นที่มีลักษณะขนาดใหญ่กว่าเส้นที่ยังไม่ได้ปรับขนาด ถ้าหากผู้ใช้ต้องการลบเส้นที่วาดทั้งหมด สามารถคลิกที่ปุ่ม Clean ได้ ดังรูปที่ 4.12

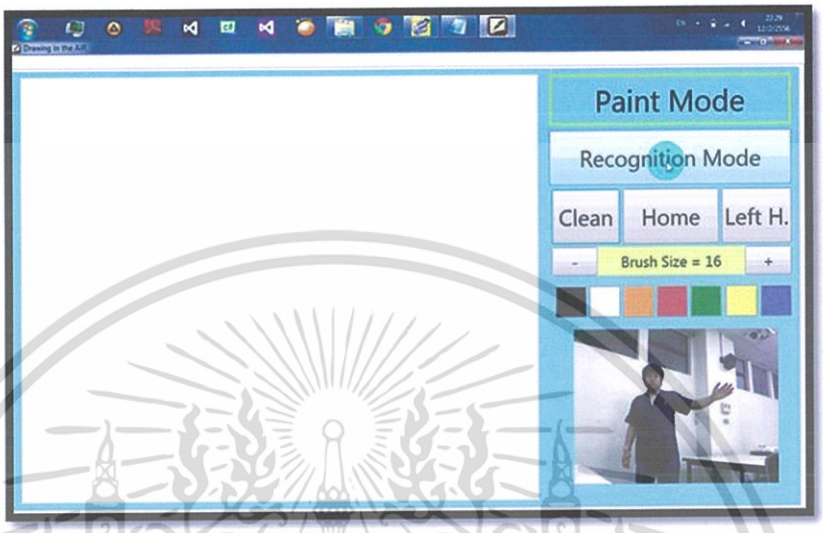


รูปที่ 4.12 ภาพแสดงก่อนและหลังการลบเส้นที่วาด

หลังจากที่ลบเส้นแล้ว ผู้ใช้สามารถวาดเส้นใหม่ได้อีกครั้งตามที่ผู้ใช้ต้องการ และถ้าหากผู้ใช้ถนัดซ้ายจะมีปุ่ม Left H. เพื่อให้ผู้ใช่สามารถสลับเปลี่ยนการควบคุมเอคร์เซอร์เมาส์ไปที่มือซ้ายได้ และเปลี่ยนมือขวาใช้ในการคลิกแทน

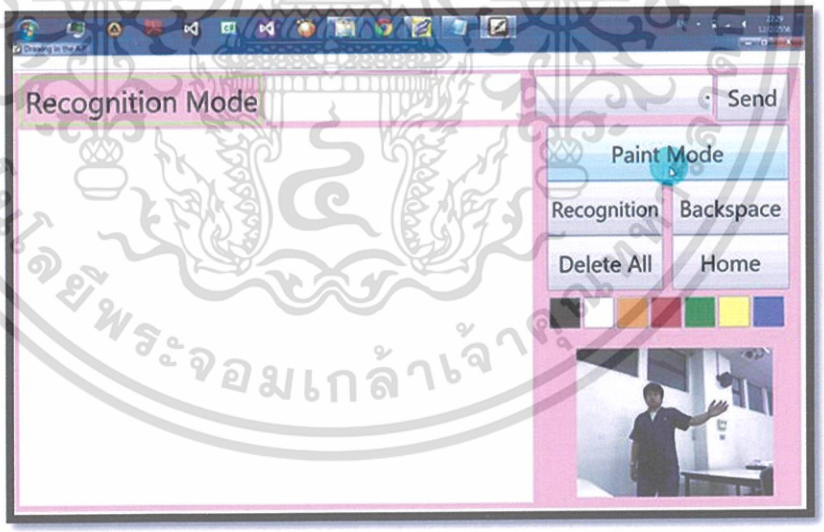
4.3 ผลการทำงานในโหมดโปรแกรมรู้จำตัวอักษร

สำหรับโปรแกรมรู้จำและส่งตัวอักษรสามารถทำงานได้ดีมากเพราะสามารถตรวจจับตัวอักษรที่ผู้ใช้ได้วาดลงบนพื้นที่ได้อย่างถูกต้อง และสามารถส่งตัวอักษรไปยังโปรแกรมเป้าหมายได้อย่างถูกต้องเช่นกัน ผู้ใช้สามารถเข้าสู่โปรแกรมโดยการคลิกที่ปุ่ม Recognition Mode จากในหน้าของ Paint Mode ได้ดังรูปที่ 4.13



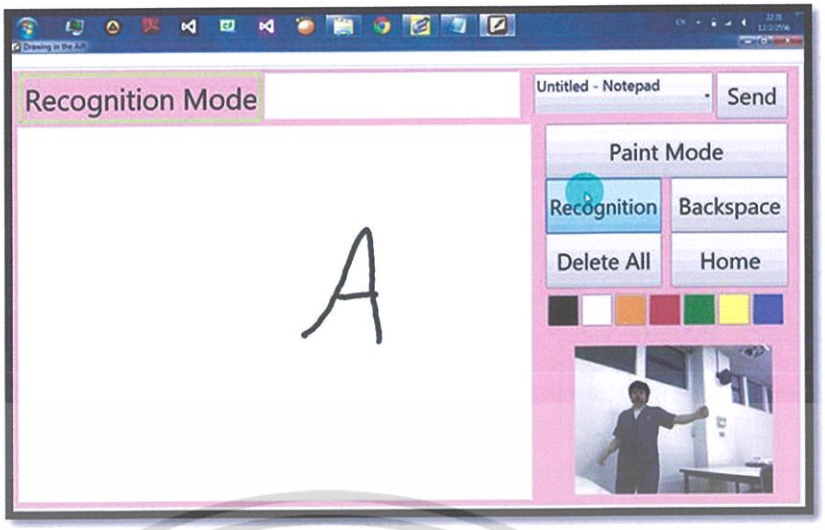
รูปที่ 4.13 แสดงการวางมือเพื่อคลิกเข้าสู่โปรแกรมรู้จำตัวอักษร

เมื่อเข้าสู่โหมดของโปรแกรมการรู้จำตัวอักษรแล้ว จะแสดงหน้าต่างดังรูปที่ 4.14



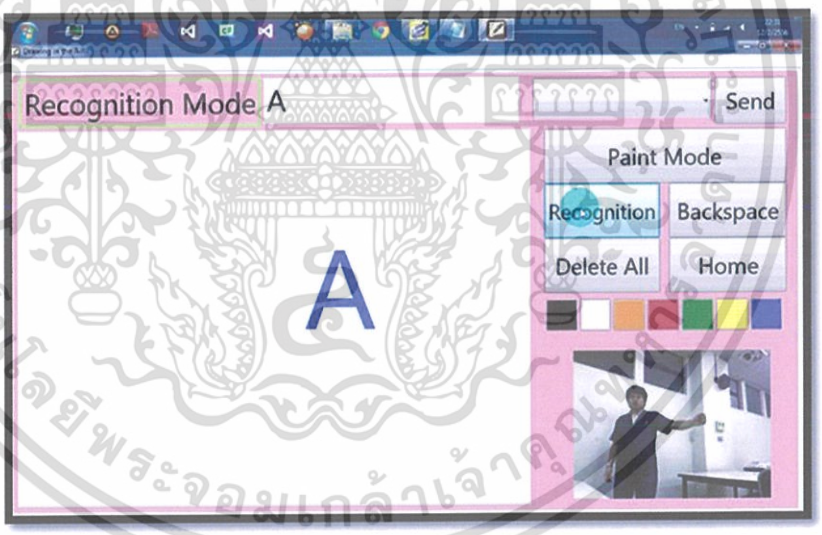
รูปที่ 4.14 หน้าต่างในโหมดโปรแกรมรู้จำตัวอักษร

ในโหมดนี้ผู้ใช้สามารถเขียนข้อความลงบนพื้นที่วาดรูปได้ดังรูปที่ 4.15



รูปที่ 4.15 ภาพแสดงการวาดเส้นในโหมดการรู้จำตัวอักษร

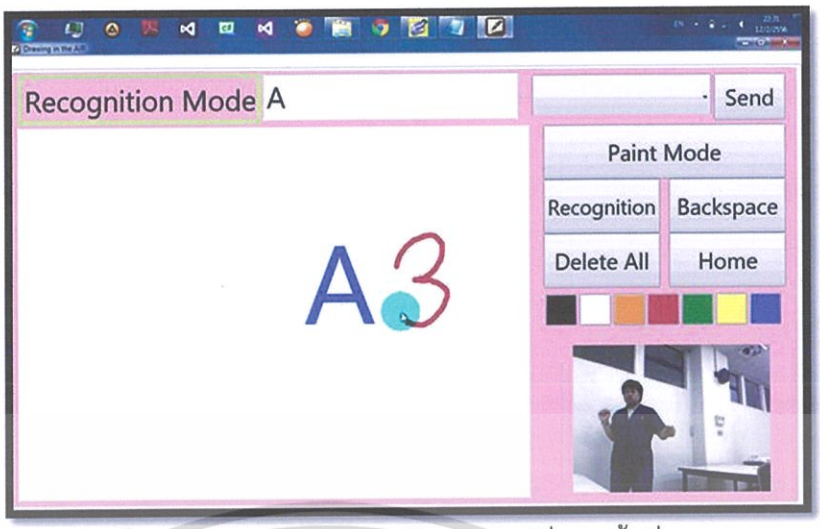
หลังจากที่ผู้ใช้วาดเส้นตัวอักษรที่เป็นลักษณะลายมือเสร็จแล้ว ผู้ใช้สามารถเปลี่ยนแปลงตัวอักษรในลักษณะที่เป็นลายมือให้อยู่ในรูปตัวอักษรที่เป็นลักษณะตัวพิมพ์ได้ โดยการกดปุ่ม Recognition ดังรูปที่ 4.15 โปรแกรมจะทำการเรียนรู้จากลักษณะเส้นที่เขียน และแปลงเป็นตัวอักษรได้แสดงได้ดังรูปที่ 4.16



รูปที่ 4.16 ภาพแสดงการแปลงเป็นตัวอักษรให้อยู่ในรูปตัวพิมพ์

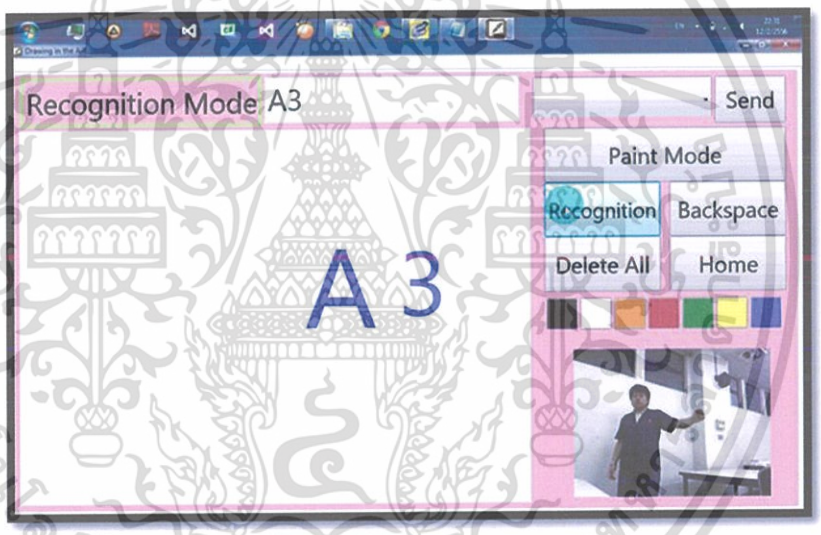
จากรูป โปรแกรมจะทำการแปลงเป็นตัวอักษรได้ เมื่อแปลงเสร็จสมบูรณ์แล้ว จะแสดงตัวอักษรในรูปของตัวพิมพ์บนพื้นที่วาดรูปโดยใช้ gistafigure (ทฤษฎีบทที่ 2) เพื่อทำการแปลงข้อความที่อยู่ในตัวแปรแบบ String ให้เป็นรูปภาพ แล้วแสดงออกบนพื้นที่วาดรูป และแสดงตัวอักษรในกล่องข้อความด้านบนของโปรแกรมอีกด้วย

นอกจากนั้นผู้ใช้สามารถเขียนตัวอักษรโดยใช้สีอื่นวาดลงบนพื้นที่วาดรูปได้ แสดงได้ดังรูปที่ 4.17



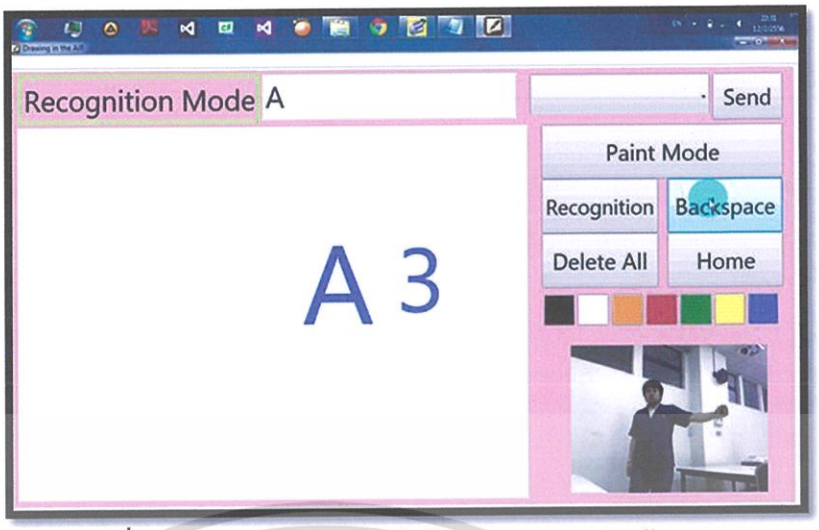
รูปที่ 4.17 ภาพแสดงการวาดด้วยปากกาสีอื่นบนพื้นที่วาดรูป

เมื่อผู้ใช้วาดตัวอักษรด้วยปากกาสีแดงเสร็จดังรูปที่ 4.17 ผู้ใช้จึงทำการคลิกที่ปุ่ม Recognition โปรแกรมจึงทำการแปลงเป็นตัวอักษรได้ แสดงได้ดังรูปที่ 4.18



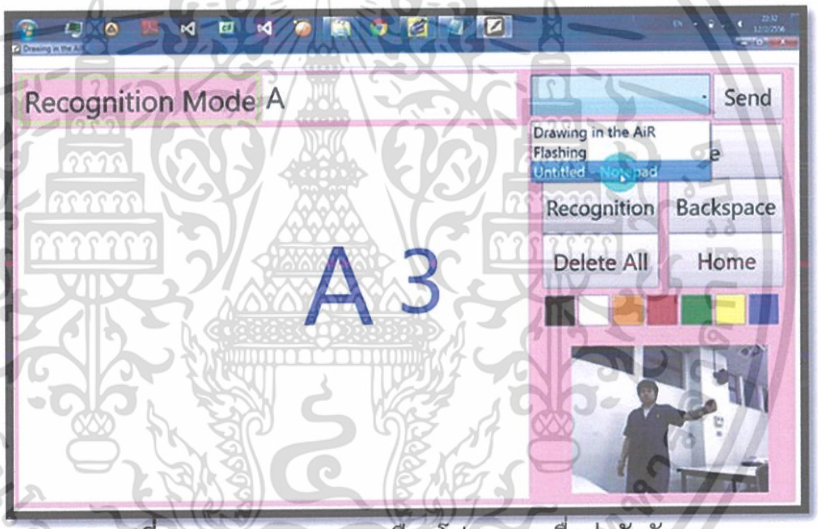
รูปที่ 4.18 ภาพแสดงการแปลงเป็นตัวอักษรแบบตัวพิมพ์จากปากกาสีอื่น

ข้อความที่เปลี่ยนเป็นตัวอักษรในรูปตัวพิมพ์จะแสดงเหมือนกับการเขียนแบบปากกาสีดำ และถ้าหากผู้ใช้ต้องการลบตัวอักษรล่าสุดที่ไม่ต้องการส่งออกไปสู่โปรแกรมอื่น ในกรณีที่ผู้ใช้เขียนตัวอักษรผิดหรือโปรแกรมทำการเปลี่ยนตัวอักษรผิดไม่ตรงตามที่ผู้ใช้ต้องการ ก็สามารถลบตัวอักษรได้โดยทำการคลิกที่ปุ่ม Backspace เพื่อลบตัวอักษร ปุ่มนี้จะทำหน้าที่ในการลบตัวอักษรในกล่องข้อความด้านบนได้ทีละตัว แสดงได้ดังรูปที่ 4.19



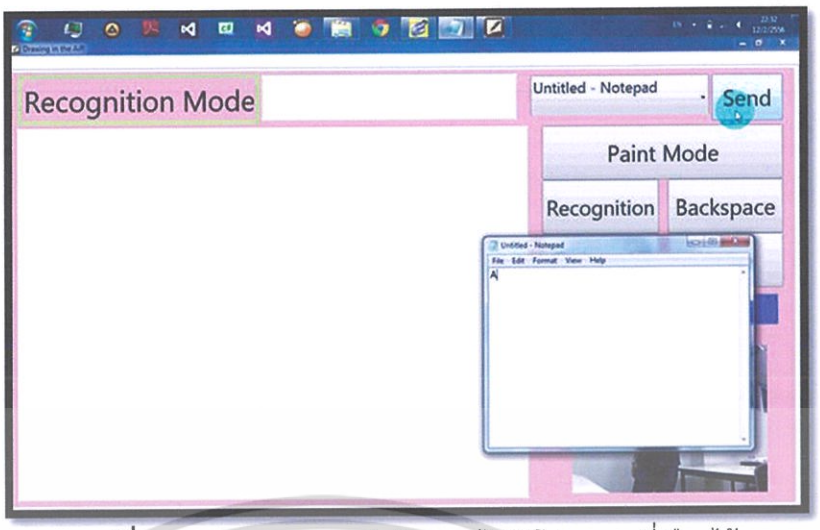
รูปที่ 4.19 ภาพแสดงการลบตัวอักษรออกจากกล่องข้อความ

ในกรณีที่ผู้ใช้ต้องการส่งตัวอักษรหรือข้อความที่แสดงในกล่องข้อความไปยังโปรแกรมอื่น สามารถทำการเลือกโปรแกรมที่ต้องการส่งข้อความได้ แสดงได้ดังรูปที่ 4.20



รูปที่ 4.20 ภาพแสดงการเลือกโปรแกรมเพื่อส่งตัวอักษร

เมื่อผู้ใช้ได้ทำการเลือกโปรแกรมเรียบร้อยแล้ว เมื่อผู้ใช้จะส่งข้อความไปยังโปรแกรมที่เลือกไว้ ให้ผู้ใช้ทำการคลิกที่ปุ่ม Send เมื่อคลิกแล้วโปรแกรมจะทำการส่งข้อความที่อยู่ในกล่องข้อความให้กับโปรแกรม โดยข้อความที่ถูกส่งไปนั้นจะแสดงขึ้น ณ บริเวณที่เคอร์เซอร์ของการพิมพ์ในโปรแกรมนั้นๆวางอยู่ ในรูปที่ 4.21 เป็นการส่งข้อความ A ไปยังโปรแกรม Notepad

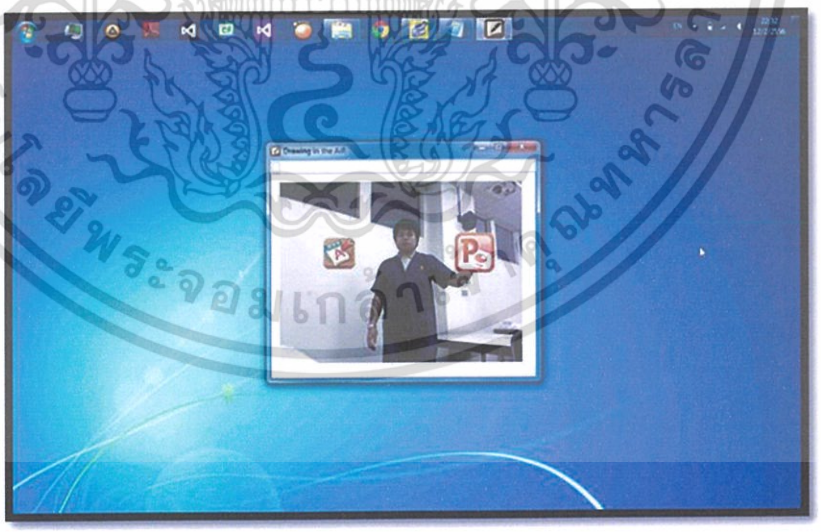


รูปที่ 4.21 ภาพแสดงการส่งข้อความไปยังโปรแกรมที่เลือกไว้

จากรูป โปรแกรมสามารถส่งข้อความ A จากโปรแกรมไปยังโปรแกรมที่เลือกไว้ได้และถ้าผู้ใช้ต้องการลบข้อความบนหน้าจอทั้งหมดและในกล่องข้อความด้วย สามารถทำได้โดยการคลิกที่ปุ่ม Delete All เพื่อทำการลบข้อความออกจากโปรแกรมทั้งหมด

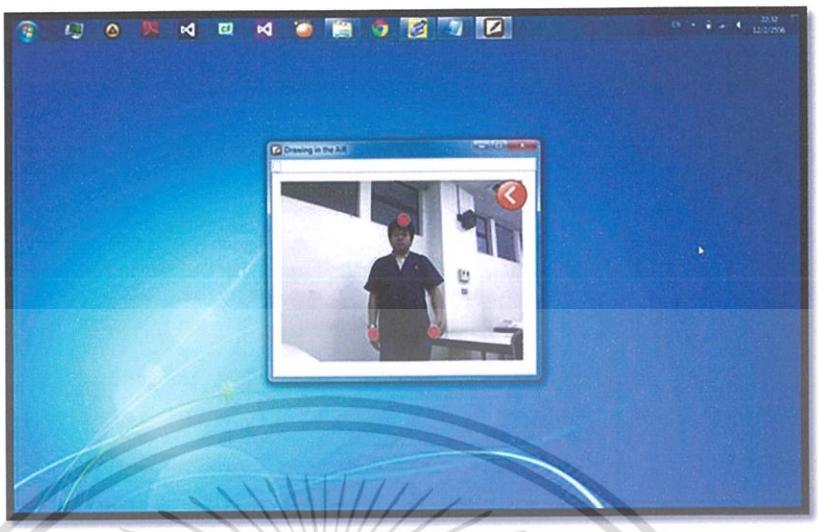
4.4 ผลการทำงานในโหมดโปรแกรมควบคุมโปรแกรมนำเสนอ

โปรแกรมในส่วนสุดท้ายเป็นโปรแกรมควบคุมการนำเสนอ สามารถทำงานอย่างมีประสิทธิภาพ คือ สามารถควบคุมโปรแกรมการนำเสนอได้ครบถ้วนทั้งเลื่อนแผ่นสไลด์ไปด้านหน้าหรือด้านหลังได้ และซ่อนโปรแกรมระหว่างการนำเสนอได้ ผู้ใช้สามารถเข้าสู่โปรแกรมโดยนำมือไปวางที่ไอคอนทางด้านขวาเป็นเวลา 2 วินาที ไอคอนจะมีขนาดใหญ่ขึ้น แสดงได้ดังรูป 4.22



รูปที่ 4.22 หน้าต่างแสดงการวางมือไว้ที่ไอคอนทางด้ายขวาเพื่อเข้าสู่โปรแกรมการควบคุมการนำเสนอ

เมื่อผู้ใช้วางมือไว้นานประมาณ 2 วินาที จะเข้ามาสู่หน้าต่างของโปรแกรมที่ใช้ในการควบคุมการนำเสนอ แสดงหน้าต่างของโปรแกรมได้ดังรูปที่ 4.23



รูปที่ 4.23 แสดงหน้าต่างของโปรแกรมการควบคุมการนำเสนอ

ในหน้าต่างของโปรแกรมจะมีการแสดงลูกบอลสีแดงไว้ 3 จุดคือ บริเวณหัวและมือทั้ง 2 ข้าง เพื่อแสดงสถานะปกติคือลูกบอลมีสีแดงในกรณีที่ยังไม่มีการสั่งการทำงานของโปรแกรมหรือส่งค่าคำสั่งเพื่อที่จะใช้ในการควบคุมการนำเสนอ

การควบคุมการนำเสนอโดยใช้มือในการควบคุม มีการแสดงสถานะดังรูปที่ 4.24 รูปด้านซ้ายจะแสดงลูกบอลเปลี่ยนสถานะเมื่อมืออยู่ในตำแหน่งดังรูป บริเวณมือขวาลูกบอลสีแดงจะเปลี่ยนเป็นสีเขียวขนาดใหญ่ เมื่อมีการเปลี่ยนสถานะของมือขวาโปรแกรมจะส่งคำสั่งในการเลื่อนสไลด์ของการนำเสนอไปข้างหน้า และเมื่อมือซ้ายอยู่ในตำแน่งดังรูป จะมีการเปลี่ยนสถานะโดยลูกบอลสีแดงเปลี่ยนเป็นสีเขียวขนาดใหญ่ โปรแกรมจะส่งคำสั่งการย้อนกลับของสไลด์

รูปที่ 4.24 แสดงสถานะของมือทั้ง 2 ข้างเพื่อส่งคำสั่งการควบคุมการนำเสนอ

ในขณะที่มีการนำเสนองาน ผู้ใช้สามารถย่อหน้าต่างโปรแกรมเก็บไว้ในทาสก์บาร์ได้ โดยการยกมือซ้ายขึ้นให้อยู่ในบริเวณเหนือศีรษะ หน้าต่างของโปรแกรมที่ใช้ในการนำเสนอจะย่อลงเก็บไว้ แสดงได้ดังรูปที่ 4.25

รูปที่ 4.25 ภาพแสดงการย่อหน้าต่างเก็บไว้เมื่อผู้ใช้ยกมือซ้าย

และเมื่อผู้ใช้ต้องการเรียกหน้าต่างโปรแกรมเดิมกลับมา เพียงแค่ผู้ใช้ยกมือขวาขึ้นเหนือศีรษะ หน้าต่างของโปรแกรมจะขยายออกมาสู่หน้าจออีกครั้งหนึ่ง แสดงได้ดังรูปที่ 4.26

รูปที่ 4.26 ภาพแสดงการขยายหน้าต่างเมื่อผู้ใช้ยกมือขวา

ผู้ใช้สามารถกลับไปสู่หน้าต่างของเมนูหลักได้ เมื่อผู้ใช้นำมือขวาไปวางไว้ที่ไอคอน  ตามรูปที่ 4.27 เป็นเวลาประมาณ 2 วินาที ไอคอนจะเปลี่ยนเป็นสีฟ้า และจะกลับไปสู่หน้าเมนูหลักได้

รูปที่ 4.27 ภาพการวางตำแหน่งมือเพื่อกลับสู่หน้าเมนูหลัก

4.5 ผลการทดลองตามขอบเขตที่กำหนด

ขอบเขตที่ 1 ผู้ใช้สามารถเขียนข้อความ วาดรูปบนอากาศโดยใช้การเคลื่อนไหวของร่างกาย การวาดผู้ใช้จะตั้งยกมือซ้ายขึ้นมาในลักษณะพับข้อศอก แล้วใช้มือขวาในการวาด มีปุ่มสลับมือในกรณีที่ผู้ใช้รู้สึกอยากเปลี่ยนมือหรือไม่ถนัด สามารถเปลี่ยนสีหรือขนาดแปลงได้

ขอบเขตที่ 2 สามารถแปลงตัวอักษรลักษณะของลายมือให้อยู่ในลักษณะของตัวอักษรแบบตัวพิมพ์ได้โดยเปลี่ยนไปที่โปรแกรม Recognition ก็จะมีปุ่มรู้จำลายมือให้เป็นตัวอักษร แต่ในส่วนนี้จะไม่สามารถเปลี่ยนขนาดของแปรงได้

ขอบเขตที่ 3 ผู้ใช้สามารถใช้การเคลื่อนไหวของร่างกายในการจัดการหน้าต่างบนหน้าจอคอมพิวเตอร์ได้ กล่าวคือสามารถเปลี่ยนหน้าของไฟล์เอกสารจากหน้าหนึ่งไปยังอีกหน้าหนึ่งได้ โดยการ เลื่อนมือซ้ายไปทางซ้ายหรือเลื่อนมือขวาไปทางขวา ย่อหน้าต่างของโปรแกรมที่ทำงานอยู่ได้ ลากโดยการยกมือซ้ายขึ้น ลากวางไอคอนหรือหน้าต่างของโปรแกรมที่เปิดอยู่ได้โดยวิธีการจะเหมือนกับการวาดรูปคือยกมือข้างซ้ายค้างไว้แล้วใช้มือข้างขวาเลื่อนไอคอนหรือหน้าต่างของโปรแกรม และสามารถเปิดและปิดโปรแกรมได้โดยวิธีการเลื่อนมือซ้ายไปด้านซ้าย

บทที่ 5
สรุปผลและแนวทางการแก้ไข

5.1 สรุปผลการทำงาน

จากการศึกษาการทำงานของกล้อง Kinect และการศึกษาโปรแกรมของซอฟต์แวร์สำเร็จรูปที่ใช้ในการทำงาน เราสามารถสร้างโปรแกรมที่ตรวจจับมือทั้งสองข้าง และศีรษะของมนุษย์ได้ สามารถเลื่อนเคอร์เซอร์เมาส์โดยใช้มือควบคุมผ่านกล้อง Kinect ได้และสามารถสั่งการให้เคอร์เซอร์เมาส์ทำการคลิกได้ และนอกจากนั้นก็ยังสามารถใช้รูปแบบการเคลื่อนที่ของมือควบคุมการนำเสนอ เช่น เลื่อนสไลด์ไปข้างหน้าหรือถอยหลังได้ และมีการสลับการทำงานของโปรแกรมเมื่อผู้ใช้ทำการนำเสนอได้อีกด้วย สามารถสลับการนำเสนอเข้าสู่การใช้มือวาดรูปหรือเขียนคำอธิบายประกอบได้ และยังสามารถนำลายมือที่ผู้นำเสนอเขียน แปลงเป็นตัวอักษรได้ สุดท้ายตัวอักษรที่ได้ยังสามารถส่งเข้าไปที่โปรแกรมนำเสนอ(ในโหมดแก้ไข) หรือโปรแกรมประเภทเอกสารได้ แต่ก็ยังมีส่วนที่เราเปลี่ยนแปลงก็คือ การตรวจจับและนับนิ้วมือโดยใช้ OpenCV ซึ่งเป็นการลดประสิทธิภาพการทำงานของระบบ เพราะ OpenCV จะใช้การประมวลผลสูง ส่งผลให้การทำงานในส่วนควบคุมเคอร์เซอร์เมาส์ไม่มีประสิทธิภาพตามไปด้วย จึงใช้การตรวจจับลักษณะของมือแทน

5.2 ปัญหาระหว่างการดำเนินงาน

- ตัวกล้องและชุดซอฟต์แวร์พัฒนา SDK ของกล้อง Kinect ยังไม่สามารถตรวจจับนิ้วมือได้
- เมื่ออัพเกรดชุดซอฟต์แวร์พัฒนา SDK แล้ว จะต้องเปลี่ยนลักษณะการเขียนโปรแกรมใหม่ทั้งหมด ทำให้การทำงานของผู้พัฒนาล้าช้า
- ตรวจจับใบหน้าโดยใช้ Emgu OpenCV มีความล่าช้ามาก
- การเขียนโปรแกรมแบบใช้ไทม์เมอร์เพื่อใช้ในการคลิกเมาส์ เป็นการสร้างความซับซ้อนให้กับระบบ ทำให้ระบบรวนได้
- การเคลื่อนที่ของมือผ่านลำตัว จะทำให้เคอร์เซอร์เมาส์เคลื่อนที่ไม่ปกติ
- การเขียนตัวอักษรในทิศทางที่ผิดปกติ ทำให้โปรแกรมแปลงตัวอักษรผิดพลาด

5.3 แนวทางการแก้ไข

- อัพเกรดชุดซอฟต์แวร์พัฒนา SDK ให้เป็นเวอร์ชันใหม่ล่าสุด
- เปลี่ยนชุดซอฟต์แวร์พัฒนา SDK เป็นเวอร์ชันเดิม และใช้การตรวจจับลักษณะมือแทนการตรวจจับนิ้วมือ
- ลบการทำงานของ Emgu OpenCV ออกแล้วใช้การทำงานแบบตรวจจับการเคลื่อนที่และตำแหน่งของมือ
- เปลี่ยนการทำงานแบบไทม์เมอร์เป็นการใช้ไทมสแปน เพื่อเข้าสู่โปรแกรมอื่นๆ และการใช้การพับแขนเป็นการคลิก
- ต้องอัพเกรดชุดซอฟต์แวร์พัฒนา SDK เป็นเวอร์ชันใหม่ล่าสุดเท่านั้น เพราะซอฟต์แวร์ชุดนี้สามารถยกเลิกการตรวจจับโครงสร้างของร่างกายบางส่วนได้
- ผู้ใช้ต้องเขียนในรูปแบบที่ถูกต้องตามหลักตัวอักษร

5.4 แนวทางการพัฒนาต่อ

- เพิ่มเติมส่วนของการรู้จำลักษณะของนิ้วมือ โดยการอัพเกรดชุดซอฟต์แวร์พัฒนา SDK และศึกษาวิธีการเขียนโปรแกรมรูปแบบใหม่ เพราะแตกต่างจากชุดซอฟต์แวร์พัฒนา SDK เวอร์ชันเดิมอย่างสิ้นเชิง
- นำการรู้จำลักษณะมือมาควบคุมการนำเสนอ
- ยกเลิกการตรวจจับร่างกายในส่วนลำตัว เพื่อให้การเคลื่อนที่ของมือผ่านลำตัวทำได้อย่างคล่องแคล่ว

บรรณานุกรม

[1] กิตตินันท์ พลสวัสดิ์, คู่มือเรียนและใช้งาน Visual C# 2010 ฉบับสมบูรณ์, บริษัท ไอดีซี พรีเมียร์ จำกัด, 2554

[2] ศุภชัย สมพาณิช, ก้าวสู่รูปแบบของการเขียนโปรแกรมแนวใหม่ WPF, บริษัท ไอดีซี อินโฟ ดิสทริบิวเตอร์ เซ็นเตอร์ จำกัด, 2551

[3] http://wiki.labomedia.org/index.php/Kinect

[4] http://www.robotworld.org.tw/data/for_trade1-d.php?News_ID=4795

[5] http://msdn.microsoft.com/en-us/library/ms646310(v=vs.85).aspx

[6] http://techon.nikkeibp.co.jp/article/NEWS/20110712/193272

[7] http://msdn.microsoft.com/en-us/library/jj131025.aspx

[8] http://msdn.microsoft.com/en-us/library/hh973078.aspx

[9] http://professorluizlopes.wordpress.com/2012/09/15/kinect-visao-geral

[10] http://blog.kirupa.com/?m=200801

[11] http://msdn.microsoft.com/en-us/library/windows/desktop/ms704040(v=vs.85).aspx

[12] http://msdn.microsoft.com/en-us/library/ms839538.aspx

[13] http://msdn.microsoft.com/en-us/library/ms839535.aspx

[14] http://www.microsoftuser.com/

[15] http://www.ifixit.com/Teardown/Microsoft+Kinect+Teardown/4066/3

[16] http://wiki.labomedia.org/index.php/Kinect

[17] http://www.robotworld.org.tw/data/for_trade1-d.php?News_ID=4795

[18] http://channel9.msdn.com/Series/KinectSDKQuickstarts/Getting-Started

[19] http://www.i-programmer.info/programming/hardware/2714-getting-started-with-microsoft-kinect-sdk-depth.html

[20] http://abhijitjana.net/2011/09/14/development-with-kinect-net-sdk-part-i-installation-and-development-environment-setup/

[21] http://www.wpftutorial.net/

[22] http://ratree21.wordpress.com/2012/05/26/wpf/

[23] http://www.narisa.com/forums/index.php?showtopic=29370

[24] http://msdn.microsoft.com/en-us/library/ms754130.aspx

[25] http://www.codeproject.com/Articles/462527/Camera-Face-Detection-in-Csharp-Using-Emgu-CV-Open

[26] http://msdn.microsoft.com/en-us/library/windows/desktop/ms704040(v=vs.85).aspx

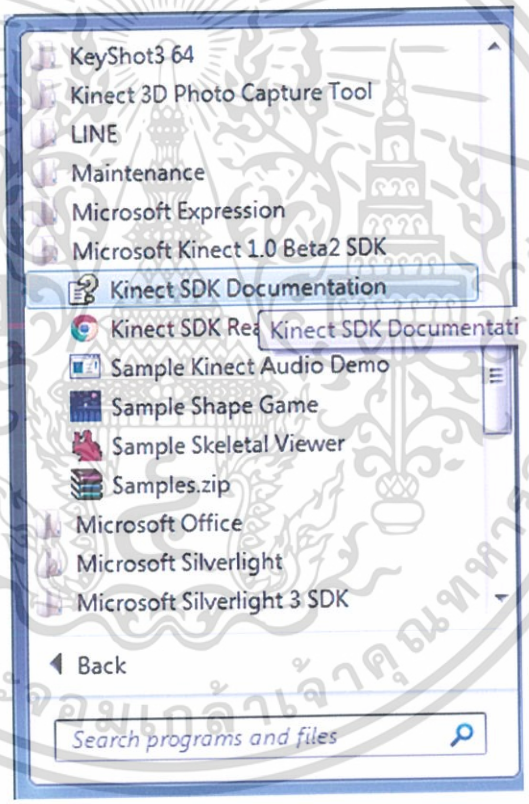
ภาคผนวก ก.

ความรู้เบื้องต้นเกี่ยวกับชุดเครื่องมือสำหรับนักพัฒนาโปรแกรมรุ่นทดลอง และวิธีการติดตั้งโปรแกรมที่จำเป็น

ความรู้เบื้องต้นเกี่ยวกับชุดเครื่องมือสำหรับนักพัฒนาโปรแกรมรุ่นทดลอง

"Kinect for Window" เป็นชุดเครื่องมือสำหรับนักพัฒนาโปรแกรมรุ่นทดลอง ที่ทางไมโครซอฟท์ได้ปล่อยออกมาเพื่อให้นักพัฒนาโปรแกรมที่มีความสนใจใน Kinect ได้เข้ามามีส่วนร่วมในการพัฒนาโปรแกรม และเพื่อแลกเปลี่ยนข่าวสาร ประสบการณ์ในการใช้เทคโนโลยีเซ็นเซอร์ Kinect บน Window 7 "Kinect for Window" โดยโครงงานนี้ใช้เวอร์ชัน 1.0 beta 2 แต่ถ้าผู้พัฒนาต้องการพัฒนาเพิ่มขึ้น สามารถใช้เวอร์ชัน 1.6 ได้แต่รูปแบบคำสั่งจะแตกต่างกันโดยสิ้นเชิง สำหรับเวอร์ชัน 1.0 beta 2 นั้นจะประกอบไปด้วย

- ไดร์ฟเวอร์สำหรับการใช้อุปกรณ์เซ็นเซอร์ Kinect บนคอมพิวเตอร์ที่รัน Windows 7
- ไลบรารี่ในการเชื่อมต่อ
- ตัวอย่างโค๊ด (Sample.zip)
- โปรแกรม Sample Skeletal Viewer และ Sample Shape Game



รูปที่ ก.1 โปรแกรมต่างๆและตัวอย่างโค๊ดที่ได้หลังจากลงชุดเครื่องมือสำหรับนักพัฒนาโปรแกรมรุ่นทดลองเวอร์ชัน 1.0 beta2

ความต้องการของระบบ

โปรแกรม "Kinect for Window" จะต้องถูกติดตั้งบนเครื่องคอมพิวเตอร์เท่านั้น จะไม่สามารถทำงานในเครื่องเสมือนได้ (virtual machine) ระบบปฏิบัติการที่สนับสนุน

- Windows 7 (x86 or x64)
- Windows 8 Developer Preview (September 2011 Build)

ความต้องการทางด้านฮาร์ดแวร์

- 32 bit (x86) or 64 bit (x64) processor
- คอมพิวเตอร์ที่ใช้ dual-core, 2.66-GHz หรือเร็วกว่า
- Windows 7 ที่สนับสนุนการแสดงผลทางกราฟิกของ Microsoft DirectX 9.0c
- 2 GB of RAM (แนะนำที่ 4 GB of RAM)
- Kinect for Xbox 360® sensor

ความต้องการทางด้านซอฟต์แวร์

- Microsoft Visual Studio ® 2010 Express หรือรุ่นอื่นๆของ Visual Studio เช่น Visual Studio Ultimate 2012 เป็นต้น
- Microsoft. NET Framework 4.0 (ติดตั้งกับ Visual Studio)
- ตัวอย่าง SkeletalViewer สำหรับ C++
- DirectX Software Development Kit รุ่นล่าสุด
- DirectX End - User Runtime Web Installer
- ตัวอย่างคำพูด (สำหรับ x86 เท่านั้น)
- แพลทฟอร์ม Microsoft Speech -- Runtime Server รุ่น 10.2 (รุ่น x86)
- แพลทฟอร์ม Microsoft Speech -- ชุดพัฒนาซอฟแวร์ รุ่น 10.2 (รุ่น x86)
- Kinect for Windows Runtime Language Pack รุ่น 0.9

ชุดเครื่องมือนี้จะมีความสามารถ ดังต่อไปนี้

- ติดตามโครงกระดูกสำหรับภาพของหนึ่งหรือสองคน ที่ Kinect เซ็นเซอร์ของกล้องสามารถจับได้
- กล้องความลึกมีความสามารถในการรู้ระยะทางของวัตถุโดยการใช้ XYZ จากเซ็นเซอร์กล้อง
- มีอารเรย์ไมโครโฟนในการประมวลเสียง

การดาวน์โหลดและติดตั้ง “Kinect for Window”

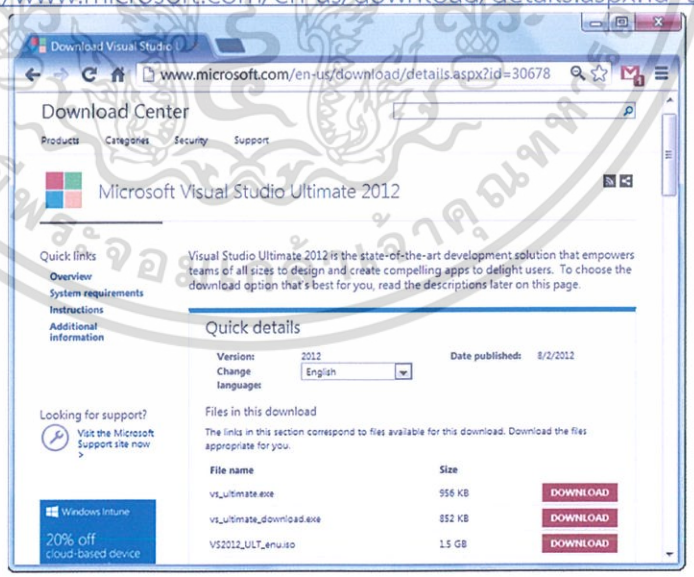
ขั้นตอนการติดตั้ง

1. ตรวจสอบความพร้อมด้านต่างๆ ดังต่อไปนี้

- ความต้องการของระบบ
- ระบบปฏิบัติการ Windows 7 (x86 or x64)
- ความต้องการด้าน Hardware
- ระบบประมวลผล dual-core, 2.66-GHz ขึ้นไป
- DirectX® 9.0c
- RAM 2 GB ขึ้นไป
- Kinect for Xbox 360®
- ความต้องการด้าน Software

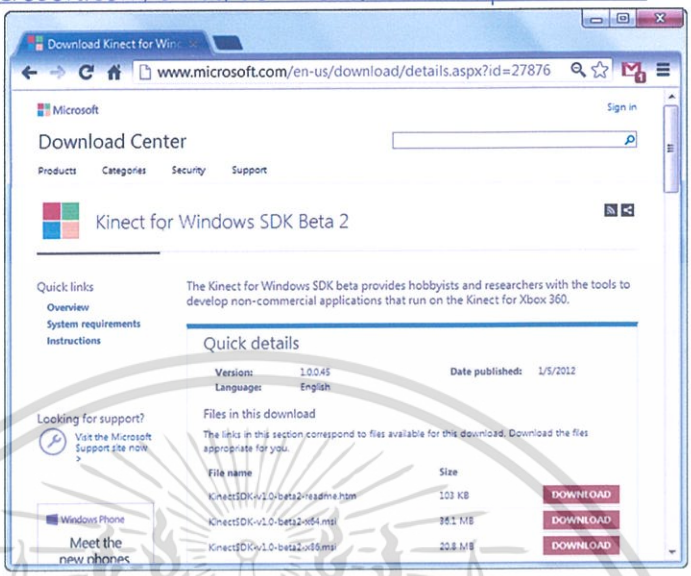
  - Microsoft Visual Studio ® 2010 Express หรือรุ่นอื่นๆของ Visual Studio เช่น Visual Studio Ultimate 2012 เป็นต้น
  - Microsoft .NET Framework 4.0 (ติดตั้งมาพร้อมกับ Visual Studio)
  - Kinect for Windows SDK Beta

2. สามารถดาวน์โหลดโปรแกรม Microsoft Visual Studio® 2010 Express ได้ที่ http://www.microsoft.com/en-us/download/details.aspx?id=30678



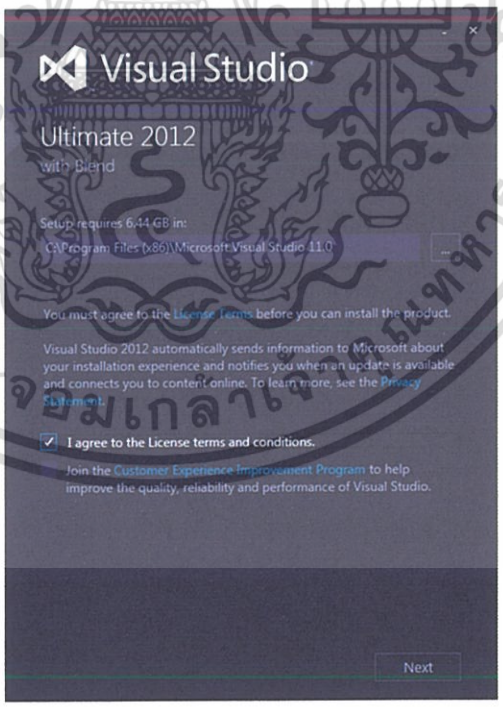
รูปที่ ก.2 หน้าเว็บไซต์สำหรับดาวน์โหลดโปรแกรม Microsoft Visual Studio 2012 [13]

และดาวน์โหลด Kinect for Windows SDK Beta ได้ที่
http://www.microsoft.com/en-us/download/details.aspx?id=27876

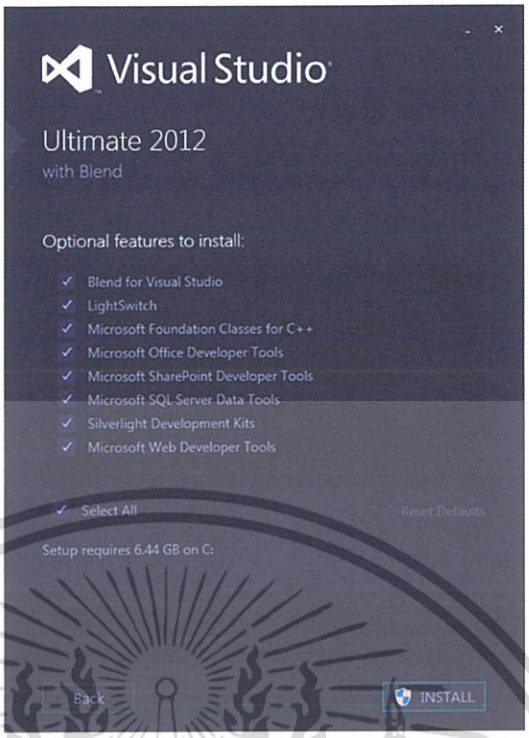


รูปที่ ก.3 หน้าเว็บไซต์สำหรับดาวน์โหลดไดร์ฟเวอร์ [14]

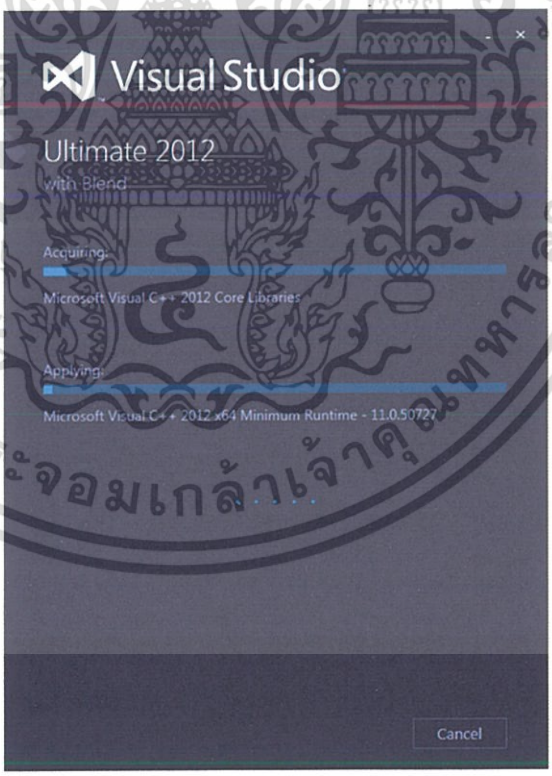
3. ทำการติดตั้ง Microsoft Visual Studio Ultimate 2012 เหมือนการติดตั้งโปรแกรมทั่วไป



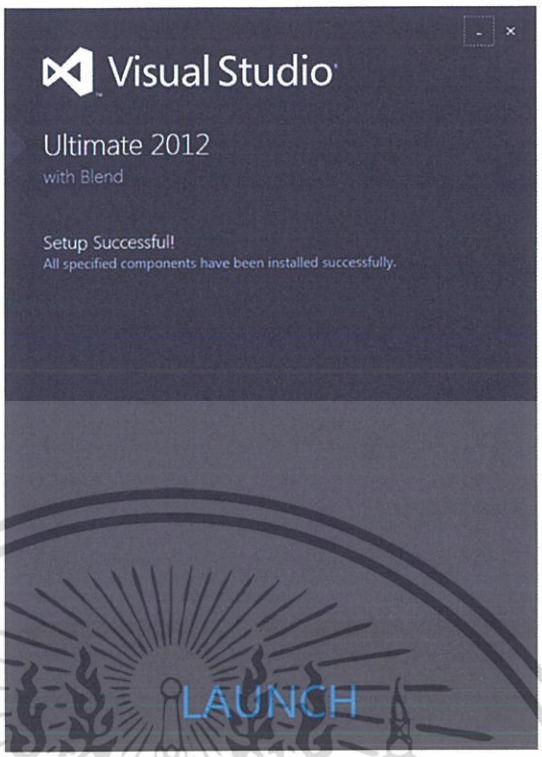
รูปที่ ก.4 หน้าการติดตั้งโปรแกรม Microsoft Visual Studio Ultimate 2012 เลือก I agree to the License terms and conditions. แล้วคลิก Next



รูปที่ ก.5 เลือก Select All แล้วคลิก INSTALL

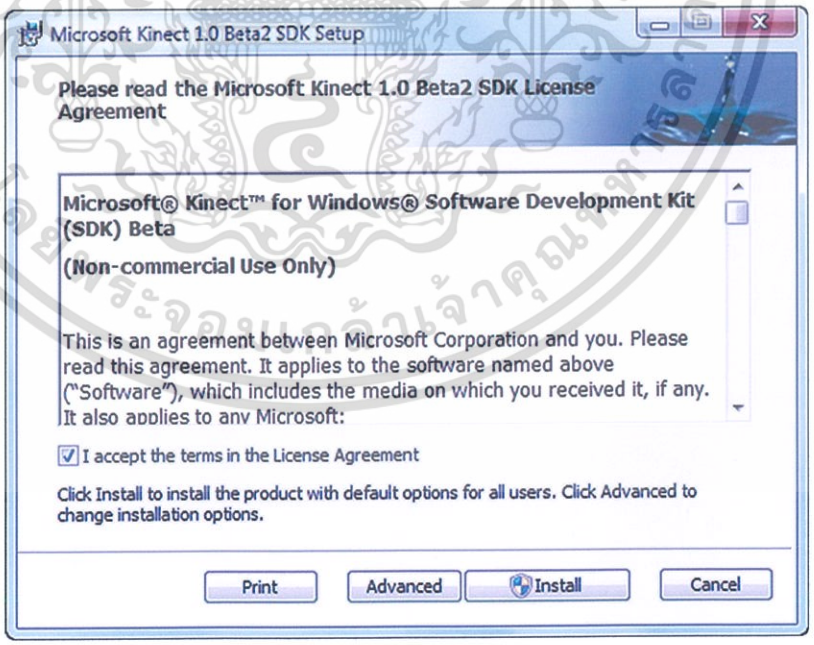


รูปที่ ก.6 ตัวติดตั้งจะทำการติดตั้งโปรแกรมต่างๆ โดยใช้เวลาสักครู่

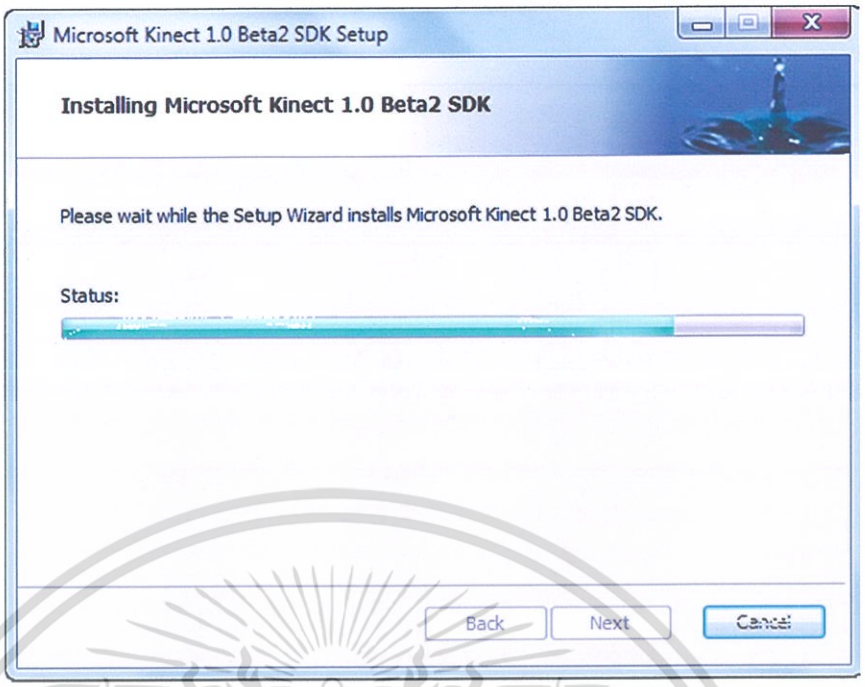


รูปที่ ก.7 เมื่อติดตั้งเสร็จเรียบร้อยแล้วสามากด LAUNCH เพื่อเข้าสู่โปรแกรม หรือกดปุ่มปิดเพื่อลงไดร์ฟเวอร์ Kinect

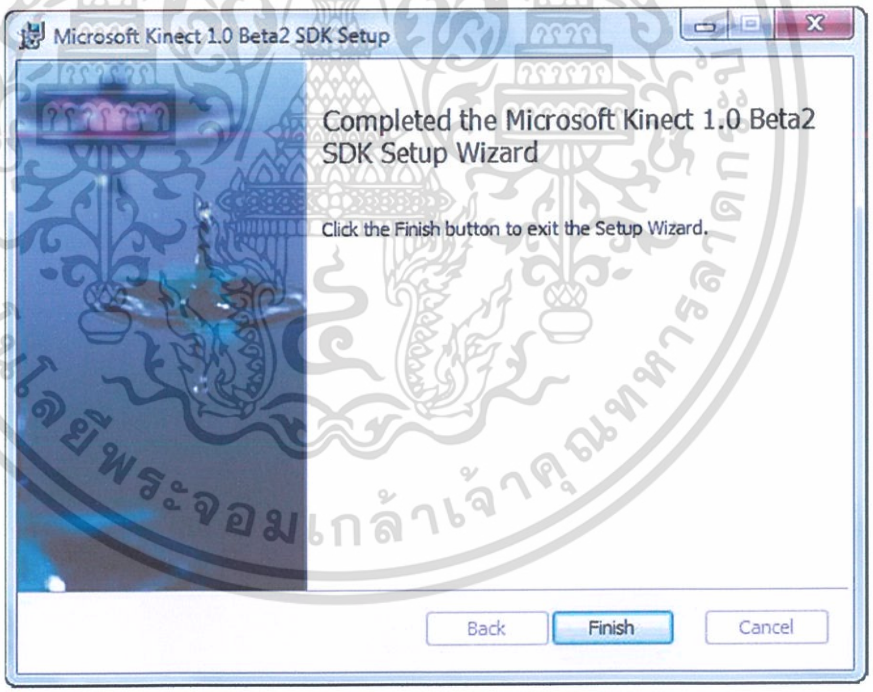
4. ทำการติดตั้ง Microsoft Kinect 1.0 Beta2 SDK เหมือนการติดตั้งโปรแกรมปกติทั่วไป



รูปที่ ก.8 ตัวติดตั้งจะบอกข้อตกลงต่างๆ ให้ทำเครื่องหมายถูกหน้าช่อง I accept the terms in the License Agreement แล้วกด Next



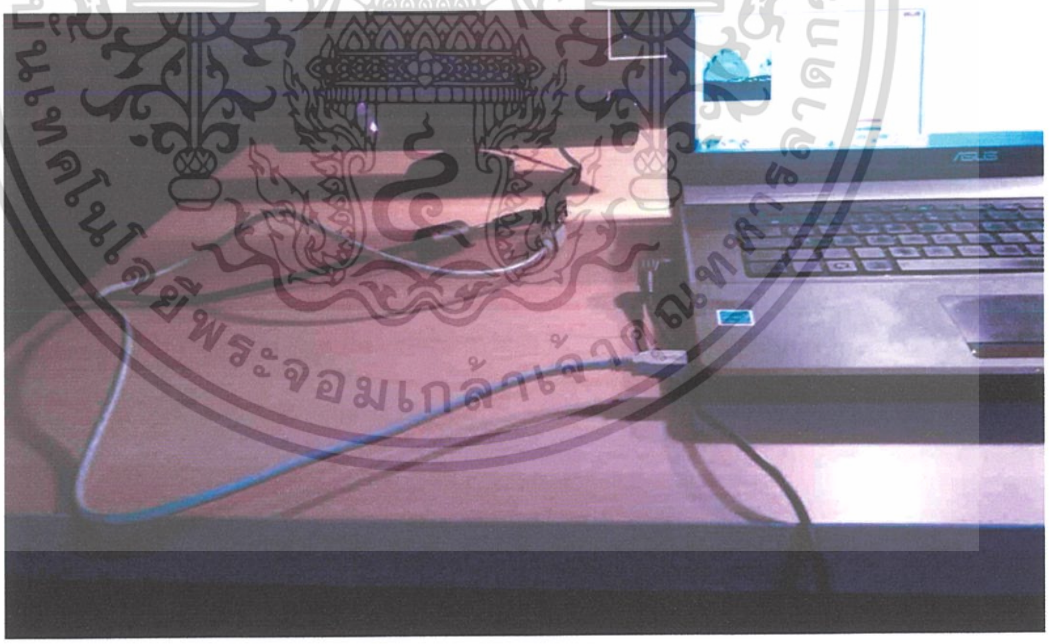
รูปที่ ก.9 ตัวติดตั้งจะทำการติดตั้งโดยใช้เวลาสักครู่



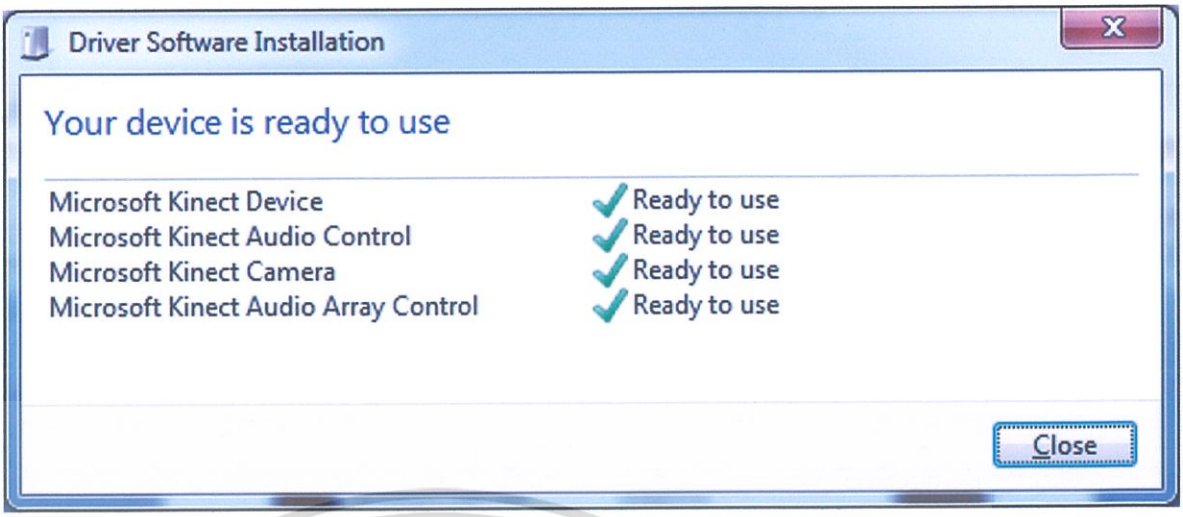
รูปที่ ก.10 เมื่อติดตั้งเสร็จให้กด Finish

5. ทำการเชื่อมต่อ Kinect for Xbox 360® ผ่าน Kinect USB Cable

รูปที่ ก.11 Kinect for Xbox 360® และ Kinect USB Cable



รูปที่ ก.12 การเชื่อมต่อ Kinect for Xbox 360® และคอมพิวเตอร์ผ่าน Kinect USB Cable

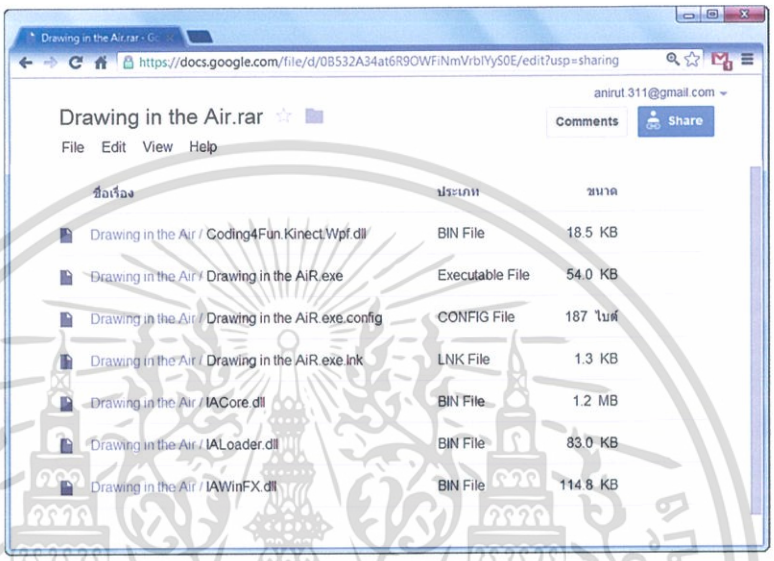


รูปที่ ก.13 ระบบจะทำการติดตั้ง Driver โดยใช้เวลาสักครู่

ภาคผนวก ข.
วิธีการใช้งานโปรแกรมเบื้องต้น

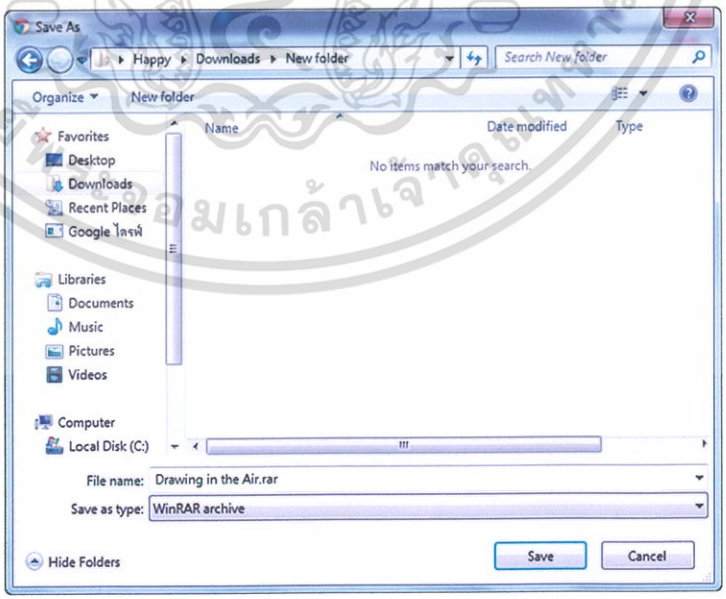
วิธีการใช้งานโปรแกรมเบื้องต้น

1. ถ้าผู้ใช้ยังไม่มีโปรแกรม สามารถดาวโหลดโปรแกรมได้ที่
https://docs.google.com/file/d/0B532A34at6R9OWFiNmVrblYyS0E/edit?usp=sharing
จากนั้นให้กดปุ่ม File จากนั้นกดปุ่ม Download หรือกดปุ่มคีย์บอร์ด Ctrl+S



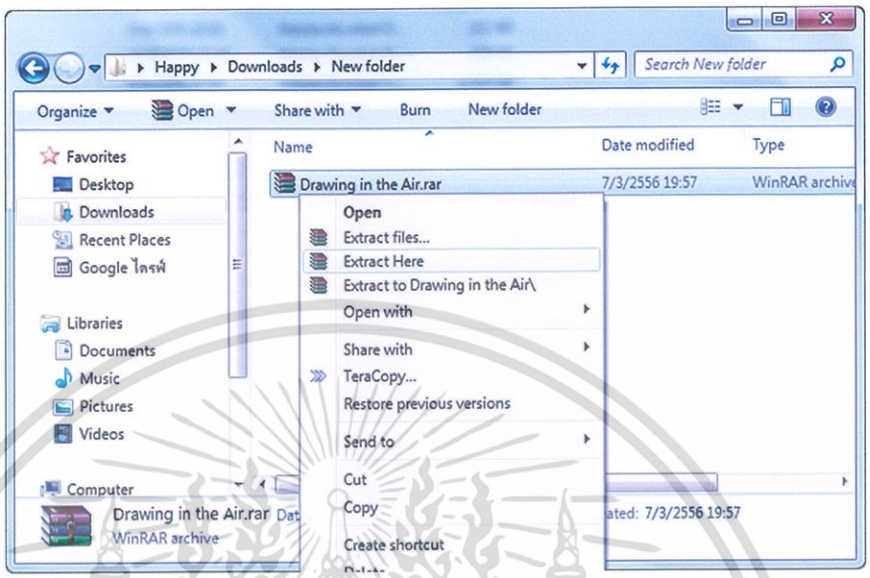
รูปที่ ข.1 หน้าต่างสำหรับดาวน์โหลดโปรแกรม Drawing in the Air [15]

2. จะได้หน้าต่างดังรูปที่ ข.2 ให้เลือกโฟลเดอร์ที่ต้องการบันทึก จากนั้นกดปุ่ม Save



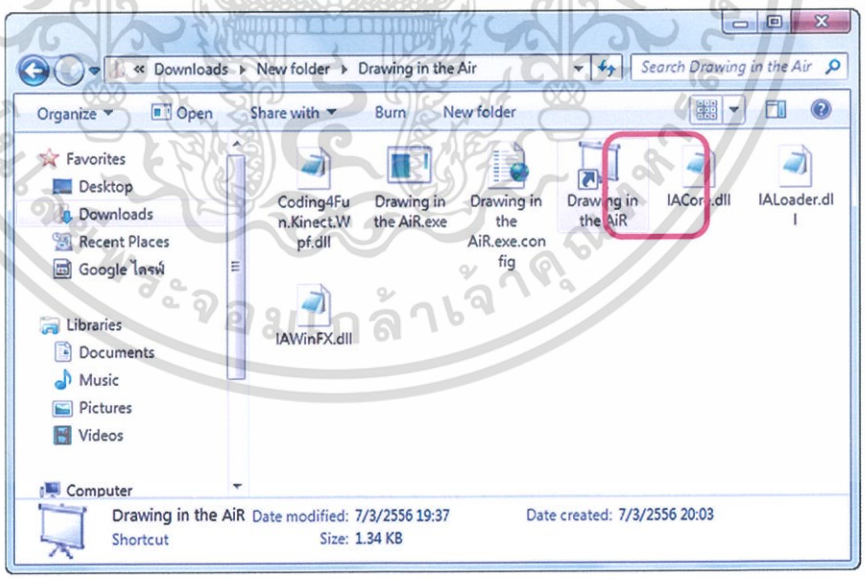
รูปที่ ข.2 หน้าต่างสำหรับบันทึกโปรแกรมลงบนคอมพิวเตอร์

4. ให้เข้าไปหาโปรแกรมที่ดาวน์โหลดมาแล้วคลิกขวาที่ไฟล์ เลือก Extract Here

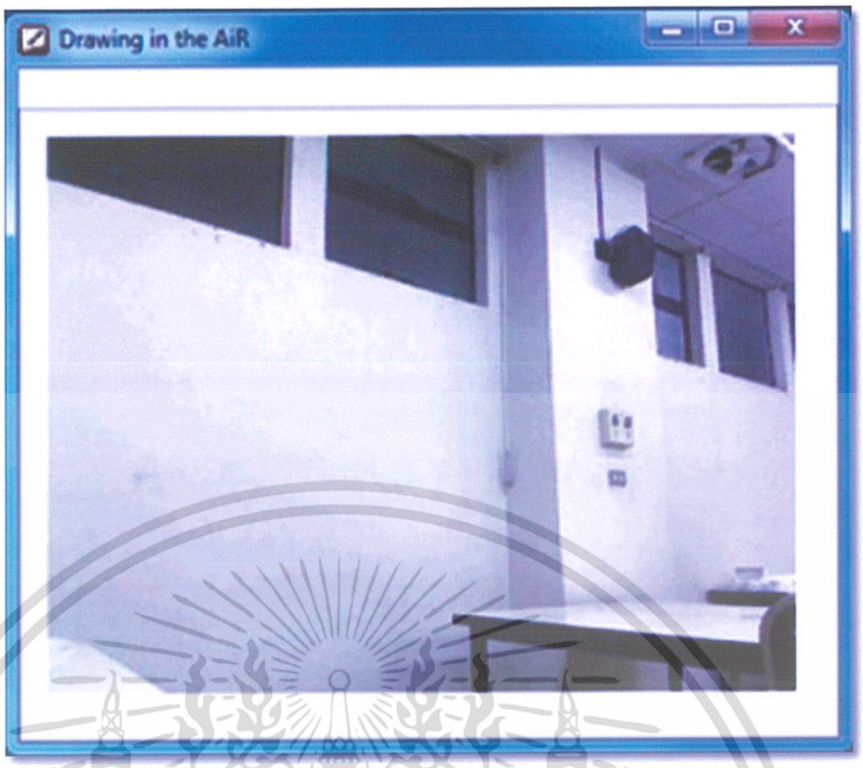


รูปที่ ข.3 การดึงโปรแกรมออกมาจากโปรแกรม Winrar

5. เข้าไปในโฟลเดอร์ จะเจอไฟล์ Drawing in the Air ให้ดับเบิ้ลคลิกที่โปรแกรมนั้น(รูปฉากรับภาพ) เพื่อเปิดโปรแกรม



รูปที่ ข.4 เปิดโปรแกรม Drawing in the Air



รูปที่ ข.5 รูปร่างโปรแกรมขณะเริ่มทำงาน

6. เมื่อมีคนเข้ามาใช้งานจะแสดงปุ่มเพื่อเข้าไปใช้งานโปรแกรม

รูปที่ ข.6 ปุ่มสีแดงคือปุ่มสำหรับเปิดโปรแกรม

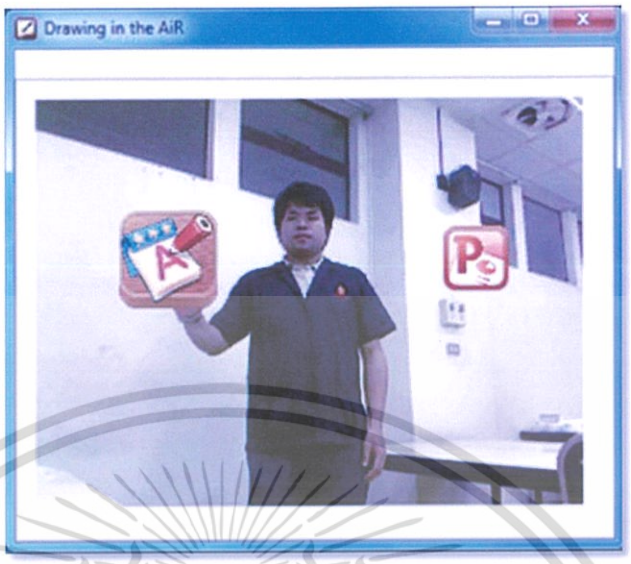
7. เมื่อนำมือมาวางไว้ที่ปุ่มสีแดงจะ ปุ่มสีแดงเปลี่ยนสีเป็นสีเขียวเพื่อแสดงว่าสามารถเข้าสู่โปรแกรมได้

รูปที่ ข.7 ปุ่มสีแดงเปลี่ยนเป็นสีเขียว

8. ส่วนนี้ผู้ใช้สามารถเลือกได้ว่าจะใช้งานโปรแกรมใด สามารถเลือกได้ 2 โปรแกรม คือ โปรแกรมวาดภาพ(ซ้าย)และโปรแกรมควบคุมโปรแกรมนำเสนอ(ขวา)

รูปที่ ข.8 ปุ่มขวามือคือโปรแกรมควบคุมโปรแกรมนำเสนอ และซ้ายมือเป็นโปรแกรมวาดภาพ

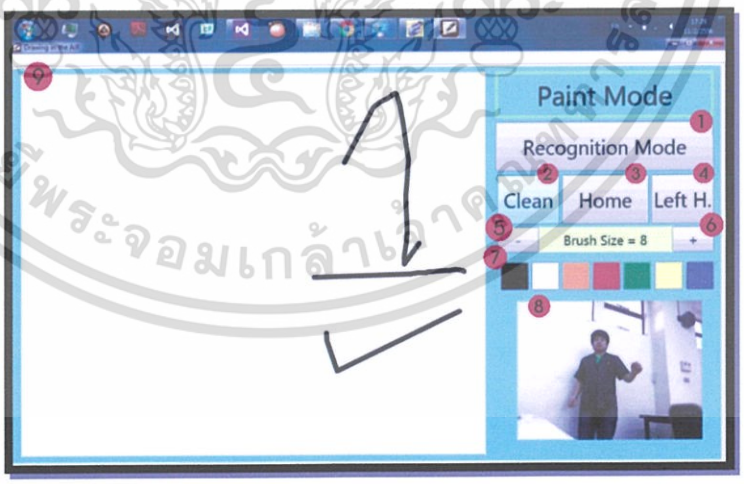
9. ผู้ใช้สามารถเลือกโปรแกรมได้โดยวางมือไว้ที่ไอคอน



รูปที่ ข.9 ผู้ใช้เลือกโปรแกรมวาดภาพ

10. ส่วนของโปรแกรมวาดภาพประกอบด้วย 9 ส่วนดังภาพ

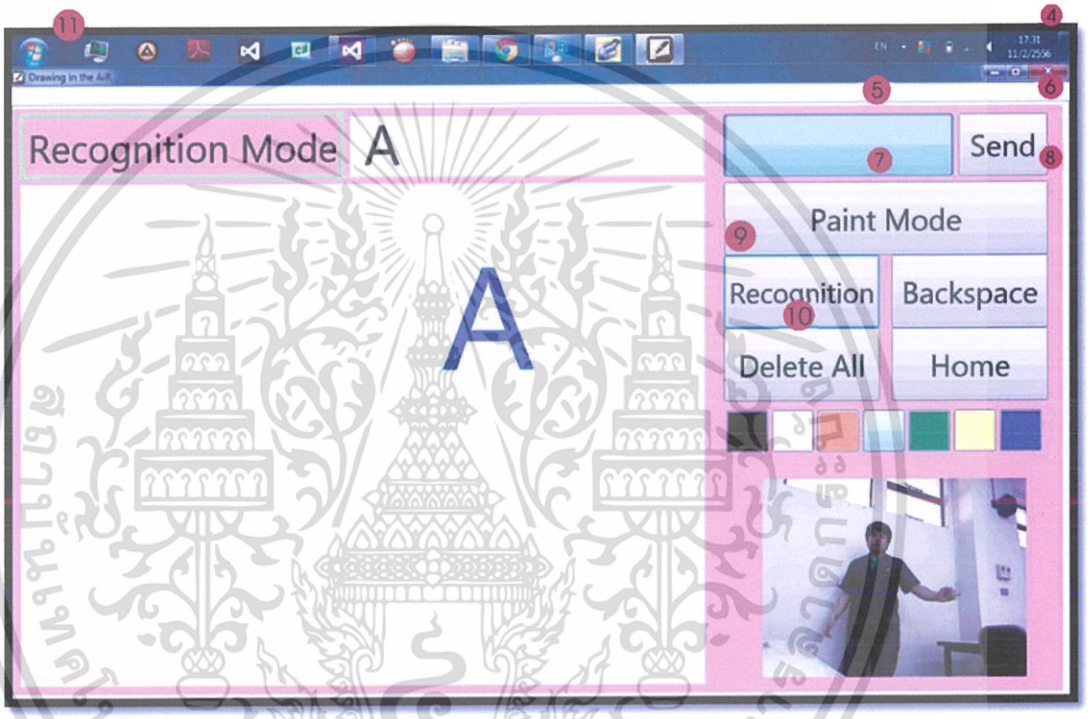
1.ปุ่มเข้าสู่โปรแกรมรู้จำอักษร
2.ปุ่มล้างพื้นที่วาดรูป
3.ปุ่มกลับไปหน้าเริ่มต้นโปรแกรม
4.ปุ่มเปลี่ยนมือซ้าย-ขวา
5.ปุ่มลดขนาดแปรง
6.ปุ่มเพิ่มขนาดแปรง
7.ปุ่มเปลี่ยนสีแปรง
8.หน้าต่างแสดงรูปผู้ใช้ที่กล้องถ่ายได้
9.พื้นที่วาดรูป



รูปที่ ข.10 โปรแกรมวาดภาพ

11. ถ้าผู้ใช้กดปุ่ม Recognition จะไปที่โปรแกรมแปลงลายมือเป็นตัวอักษร โดยมีส่วนประกอบสำคัญ 11 ส่วน ดังนี้

1.แถบรับข้อความที่ผู้ใช้วาด
2.ปุ่มเลือกโปรแกรมที่ส่งส่งตัวอักษรออกไป
3.ปุ่มส่งตัวอักษร
4.ปุ่มเข้าสู่โปรแกรมวาดรูป
5.ปุ่มแปลงจากลายมือเป็นตัวพิมพ์
6.ปุ่มลบตัวอักษรทีละตัว
7.ปุ่มล้างพื้นที่วาดตัวอักษรและข้อความ
8.ปุ่มกลับไปหน้าเริ่มต้นโปรแกรม
9.ปุ่มเปลี่ยนสีแปรง
10.หน้าต่างแสดงรูปผู้ใช้ที่กล้องถ่ายได้
11.พื้นที่วาดรูป

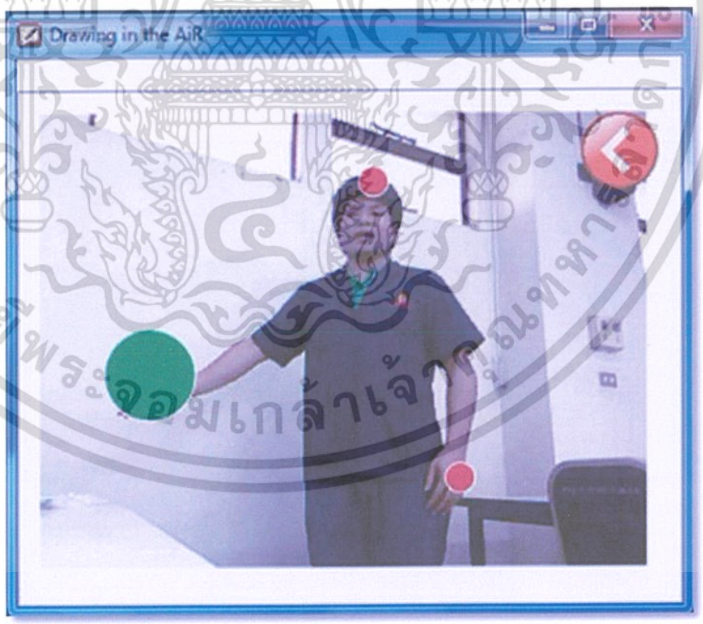


รูปที่ ข.11 โปรแกรมแปลงลายมือเป็นตัวอักษร

12. และถ้าผู้ใช้เลือกโปรแกรมควบคุมโปรแกรมนำเสนอ ถ้าผู้ใช้สามารถแกว่งมือซ้ายไปทางซ้าย เพื่อเลื่อนการนำเสนอไปหน้าที่แล้ว

รูปที่ ข.12 โปรแกรมควบคุมโปรแกรมนำเสนอ ผู้ใช้ต้องการเลื่อนการนำเสนอไปหน้าที่แล้ว

13. ถ้าผู้ใช้สามารถแกว่งมือขวาไปด้านขวาเพื่อเลื่อนการนำเสนอไปหน้าถัดไป



รูปที่ ข.13 โปรแกรมควบคุมโปรแกรมนำเสนอ ผู้ใช้ต้องเลื่อนการนำเสนอไปหน้าถัดไป

14. ถ้าผู้ใช้ต้องการย่อโปรแกรมควบคุมลง ให้ยกมือขวาขึ้น ยกมือขวาขึ้นอีกครั้งเพื่อเรียกโปรแกรมกลับมาแสดงบนหน้าจอ และยกมือซ้ายขึ้นเพื่อนำเสนอผลงานแบบเต็มหน้าจอ(ต้องเปิดโปรแกรมเพาเวอร์พอยต์เตรียมไว้ก่อน สั่งย่อโปรแกรมควบคุมลง และยกมือซ้ายขึ้นเพื่อนำเสนอ)

รูปที่ ข.14 ผู้ใช้ยกมือขวาขึ้นเพื่อย่อโปรแกรม

รูปที่ ข.15 โปรแกรมควบคุมถูกย่อลง