รายงานการวิจัยฉบับสมบูรณ์

การศึกษาการประยุกต์ใช้งานอุปกรณ์คินเนคต์สำหรับส่วนติดต่อผู้ใช้แบบสามมิติ

Study of the Applications of Kinect Devices for 3D Graphical User Interfaces

ดร. นัทธพงศ์ จึงธีรพานิช

ดร. อุกฤษฏ์ วัชรีฤทัย

ได้รับทุนสนับสนุนงานวิจัยจากเงินรายได้ ประจำปีงบประมาณ 2555

วิทยาลัยนานาชาติ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อโครงการ การศึกษาการใช้งานอุปกรณ์คินเนคต์สำหรับส่วนติดต่อผู้ใช้แบบสามมิติ
แหล่งเงิน เงินรายได้ วิทยาลัยนานาชาติ
ประจำปีงบประมาณ 2555 จำนวนเงินที่ได้รับการสนับสนุน 50,000 บาท
ระยะเวลาทำการวิจัย 18 เดือน ตั้งแต่ 1 ต.ค. 2554 ถึง 31 มี.ค. 2556

ชื่อ-สกุล หัวหน้าโครงการและผู้ร่วมโครงการวิจัย พร้อมระบุ หน่วยงานต้นสังกัดและอีเมล์

1. หัวหน้าโครงการ ดร.นัทธพงศ์ จึงธีรพานิช วิทยาลัยนานาชาติ
โทรศัพท์ 02-329-8261 อีเมล์ kjnattha@kmitl.ac.th
2. ผู้ร่วมวิจัย ดร.อุกฤษฏ์ วัชรีฤทัย วิทยาลัยนานาชาติ
โทรศัพท์ 02-329-8261 อีเมล์ kwukrit@kmitl.ac.th

บทคัดย่อ

ความท้าทายหนึ่งในการออกแบบส่วนติดต่อผู้ใช้แบบสามมิติคือจะออกแบบอย่างไรให้ง่ายต่อการใช้งาน และในขณะเดียวกันได้ใช้ประโยชน์จากการแสดงผลและโต้ตอบแบบสามมิติได้อย่างเต็มที่ ในแอพพลิเคชันประเภทนี้การโต้ตอบกับผู้ใช้โดยใช้อุปกรณ์อินพุตแบบเดิมๆ เช่นเมาส์หรือแป้นพิมพ์ไม่สะดวก โครงงานวิจัยนี้จึงได้ศึกษาถึงวิธีการและประสิทธิผลของการประยุกต์ใช้อุปกรณ์คินเนคต์เพื่อช่วยให้ผู้ใช้สามารถใช้งานส่วนติดต่อผู้ใช้แบบสามมิติได้สะดวกมากขึ้น อุปกรณ์คินเนคต์เป็นอุปกรณ์ซึ่งพัฒนาโดยบริษัทไมโครซอฟต์โดยมีวัตถุประสงค์แรกเริ่มคือให้ผู้เล่นเกมสามารถบังคับตัวละครในเกมได้อย่างเป็นธรรมชาติ ตัวอุปกรณ์ซึ่งประกอบด้วยกล้องจับภาพวิดีโอและวัดระยะห่างของจุดจากตัวอุปกรณ์สามารถอ่านค่าตำแหน่งของข้อต่อที่สำคัญในร่างกายของผู้ใช้ ซึ่งแอพพลิเคชันสามารถนำข้อมูลนี้ไปใช้วิเคราะห์ท่าทางของผู้ใช้ได้ ในปัจจุบันอุปกรณ์คินเนคต์ได้ถูกนำมาวิจัยและประยุกต์ใช้กับแอพพลิเคชันหลากหลายประเภท ในโครงงานวิจัยนี้ผู้วิจัยได้ศึกษาการพัฒนาแอพพลิเคชันที่ติดต่อกับอุปกรณ์คินเนคต์โดยใช้ Microsoft Kinect for Windows SDK การพัฒนาส่วนติดต่อผู้ใช้แบบสามมิติด้วย OpenGL และพัฒนาแอพพลิเคชันตัวอย่างที่เป็นประโยชน์ เช่นแอพพลิเคชันสำหรับวาดกราฟในแบบสามมิติซึ่งผู้ใช้สามารถปรับการแสดงผลในปริภูมิสามมิติได้โดยใช้ท่าทาง แอพพลิเคชันสำหรับฝึกท่ารำในการรำไทย และแอพพลิเคชันเกมสำหรับฝึกฝนและจำลองการแข่งขันชกมวย นอกจากนี้ผู้วิจัยยังได้ศึกษาถึงการใช้งานอุปกรณ์คินเนคต์จำนวนสองชุดพร้อมกันเพื่อช่วยขยายพื้นที่การปฏิบัติงานของอุปกรณ์ ผลที่ได้จากการศึกษาในโครงงานวิจัยนี้จึงน่าจะเป็นประโยชน์อย่างยิ่งสำหรับผู้ที่ต้องการนำอุปกรณ์คินเนคต์ไปใช้ในแอพพลิเคชันที่มีส่วนติดต่อผู้ใช้แบบสามมิติ

คำสำคัญ: คินเนคต์, ส่วนติดต่อผู้ใช้แบบสามมิติ, OpenGL, HCI

Research Title: Study of the Applications of Kinect Devices for 3D Graphical User Interfaces

Researcher: Dr. Natthapong Jungteerapanich and Dr. Ukrit Watchareeruetai

Faculty: International College **Department:** School of Engineering and Technology

ABSTRACT

One of the challenges in 3D user interface design is how to design the interface that is easy to use while at the same time utilizes the full potential of the 3D interface. Interactions using traditional input devices, such as a mouse or a keyboard, are inconvenient. This project has studied the method and the effectiveness of using Kinect devices in applications with 3D interface to facilitate user interactions. The Kinect device was initially developed by Microsoft Inc. to enable gamers to control game characters in a more natural way. The device, which consists of an array of video and distance-sensing cameras, is capable of locating the positions of important joints within the user's body. This joint information can then be used to analyze the user's body gesture. Currently, Kinect devices have been applied to a wide range of applications. In this research project, we studied how to develop an application which utilizes the data from the device using Microsoft Kinect for Windows SDK and, more specifically, how to develop an application with 3D interface using OpenGL that utilizes Kinect as an input device. We developed a number of useful sample applications, including a 3D-graph plotting application that allows the user to navigate the 3D space by making gestures, an application teaching correct postures in Thai classical dancing, and a game application for boxing practice and simulating boxing matches. We also studied the use of two Kinect devices to expand their operating ranges. The result from this study should be particularly useful for the audience who would like to adopt Kinect devices in an application with 3D graphical user interface.

Keywords: Kinect, 3D Graphical User Interface, OpenGL, HCI

กิตติกรรมประกาศ

ผู้วิจัยขอขอบพระคุณบุคลากรของวิทยาลัยนานาชาติทุกท่านที่ช่วยอำนวยความสะดวกให้กับข้าพเจ้ารวมทั้งความช่วยเหลือต่างๆในการทำวิจัยนี้ ขอขอบคุณนักศึกษาในหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมซอฟต์แวร์ (นานาชาติ) ดังรายนามต่อไปนี้ นางสาวกรองกาญจน์ พืชพันธุ์ นางสาวจุฑามาศ คร้อสวัสดิ์ นายชิติสรรค์ ทรัพย์เตชิตมณี นายทักษ์ดนัย ทองศรี นายธนาธิป จันทร์สุวรรณ และนางสาววรินทร์พร บุญประถัมภ์ สำหรับการช่วยพัฒนาโปรแกรม การวิจัยครั้งนี้ได้รับทุนสนับสนุนการวิจัยจากสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง จากแหล่งทุนเงินรายได้วิทยาลัยนานาชาติประจำปีงบประมาณ พ.ศ. 2555

ดร. นัทธพงศ์ จึงธีรพานิช

ดร. อุกฤษฏ์ วัชรีฤทัย

สารบัญ

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

ภาพที่ **หน้า**

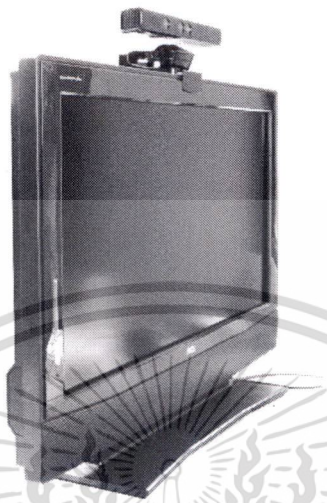
บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงการ

ในปัจจุบันเทคโนโลยีคอมพิวเตอร์ด้านการประมวลผลและแสดงผลกราฟิกสามมิติได้พัฒนาขึ้นเป็นอย่างมาก เครื่องคอมพิวเตอร์ที่มีความสามารถด้านนี้มีราคาถูกลงและมีความแพร่หลายมากขึ้น จึงเป็นผลให้ซอฟต์แวร์จำนวนไม่น้อยในปัจจุบันมีการแสดงผลกราฟิกสามมิติ ซึ่งนอกจากจะช่วยให้การแสดงผลมีความสวยงามและเสมือนจริงแล้ว ยังช่วยให้แสดงผลข้อมูลที่มีความซับซ้อนให้ผู้ใช้เข้าใจได้ง่ายขึ้นกว่าการแสดงผลในรูปแบบสองมิติ [1, 2] ตัวอย่างเช่น ซอฟต์แวร์แสดงภาพสามมิติของร่างกายผู้ป่วยเพื่อช่วยในการวินิจฉัยโรคหรือช่วยในการผ่าตัด ซอฟต์แวร์แสดงแผนที่ในรูปแบบสามมิติ หรือซอฟต์แวร์สำหรับแสดงผลข้อมูลทางสถิติ คณิตศาสตร์ หรือวิทยาศาสตร์ในรูปแบบสามมิติ เป็นต้น นอกจากนี้ยังมีความพยายามที่จะออกแบบส่วนติดต่อผู้ใช้ของระบบปฏิบัติการให้แสดงผลในรูปแบบสามมิติ เพื่อช่วยให้ผู้ใช้ค้นหาข้อมูลหรือเรียกใช้ฟังก์ชันต่างๆ ของระบบปฏิบัติการซึ่งมีจำนวนมากได้ง่ายขึ้น ซอฟต์แวร์เหล่านี้ไม่เพียงแต่แสดงผลกราฟิกสามมิติเท่านั้น แต่ยังเปิดโอกาสให้ผู้ใช้ควบคุมมุมมองสามมิติหรือโต้ตอบกับวัตถุสามมิติบนหน้าจอ (เช่น การหมุนวัตถุ การซูมเข้าหรือซูมออก หรือการเปลี่ยนองศาของมุมมองในแนวต่างๆ เป็นต้น) โดยส่วนใหญ่ผู้ใช้ใช้งานซอฟต์แวร์เหล่านี้ผ่านเมาส์หรือคีย์บอร์ด ปัญหาที่พบคือการควบคุมการแสดงผลกราฟิกสามมิติโดยใช้เมาส์หรือแป้นพิมพ์ไม่สะดวกและไม่เป็นธรรมชาติ ผู้ใช้จำเป็นต้องใช้เวลาเรียนรู้และฝึกฝนวิธีการใช้เมาส์และคีย์บอร์ดสำหรับควบคุมการแสดงผลกราฟิกสามมิติก่อนจึงสามารถใช้งานซอฟต์แวร์ได้คล่องแคล่ว จากปัญหานี้จึงทำให้ผู้วิจัยเกิดความคิดที่จะนำเอาอุปกรณ์คินเนคต์มาใช้เป็นอุปกรณ์อินพุตสำหรับโต้ตอบกับซอฟต์แวร์ที่มีส่วนติดต่อผู้ใช้แบบสามมิติ เพื่อทำให้การติดต่อและควบคุมการแสดงผลสามมิติสะดวก ง่าย และเป็นธรรมชาติมากขึ้น

คินเนคต์เป็นอุปกรณ์อินพุตสำหรับรับรู้ท่าทางการเคลื่อนไหวร่างกายของผู้ใช้ คินเนคต์ผลิตขึ้นโดยบริษัทไมโครซอฟต์ตั้งแต่ปลายปีพ.ศ. 2553 เพื่อเป็นอุปกรณ์เสริมสำหรับเครื่องเล่นเกม Xbox 360 โดยตัวอุปกรณ์ถูกออกแบบให้ติดตั้งบนจอภาพ เมื่อใช้งานในเกมที่รองรับ ผู้เล่นสามารถตอบโต้กับเกมโดยยืนหันหน้าเข้าจอภาพและทำท่าทางต่างๆ อย่างเป็นธรรมชาติ ต่อมาเนื่องจากมีผู้สนใจทดลองเชื่อมต่อและใช้งานคินเนคต์กับเครื่องคอมพิวเตอร์เป็นจำนวนมาก บริษัทไมโครซอฟต์จึงตัดสินใจจำหน่ายชุดพัฒนาคินเนคต์สำหรับระบบปฏิบัติการวินโดวส์ (Kinect for Windows SDK) สำหรับทดลองและวิจัย ทางผู้วิจัยเล็งเห็นว่าคินเนคต์เป็นอุปกรณ์ที่สามารถใช้เป็นอุปกรณ์อินพุตสำหรับซอฟต์แวร์ที่มีส่วนติดต่อผู้ใช้แบบสามมิติ และสามารถนำมาประยุกต์ใช้ในงานต่างๆ ได้หลากหลาย รวมถึงใช้

สนับสนุนการเรียนการสอนรายวิชาซึ่งมีความเกี่ยวข้องกับกราฟิกสามมิติ เช่น วิชาแคลคูลัสและเรขาคณิตวิเคราะห์ หรือวิชาคอมพิวเตอร์กราฟิก เป็นต้น



ภาพที่ 1.1 การติดตั้งอุปกรณ์คินเนคต์บนจอภาพ
(ภาพประกอบจาก http://www.gizmodo.com.au)

1.2 วัตถุประสงค์และขอบเขตของโครงการ

โครงการวิจัยชิ้นนี้มีวัตถุประสงค์ดังต่อไปนี้

- เข้าใจโครงสร้าง หลักการทำงาน และการใช้งานอุปกรณ์คินเนคต์
- เพื่อเข้าใจการพัฒนาโปรแกรมที่มีส่วนติดต่อผู้ใช้แบบสามมิติ โดยผู้ใช้สามารถใช้งานผ่านอุปกรณ์คินเนคต์
- เพื่อวิเคราะห์และประเมินความเหมาะสมของการประยุกต์ใช้อุปกรณ์คินเนคต์กับส่วนติดต่อผู้ใช้แบบสามมิติในงานด้านต่างๆ รวมถึงงานสนับสนุนการเรียนการสอน

1.3 วิธีดำเนินการวิจัย

การดำเนินการวิจัยของโครงการนี้แบ่งได้เป็น 3 ส่วนหลักๆ ได้แก่ การศึกษาโครงสร้าง การทำงาน และการเขียนโปรแกรมใช้งานอุปกรณ์คินเนคต์ การพัฒนาตัวอย่างโปรแกรมที่ใช้งานอุปกรณ์คินเนคต์ และการวิเคราะห์และประเมินความเหมาะสมของการใช้งาน

แผนงานในการดำเนินงานวิจัยอย่างละเอียดเป็นไปตามลำดับขั้นตอนดังนี้

- จัดซื้อวัสดุและอุปกรณ์ที่จำเป็นต่อการดำเนินงาน
- ศึกษาถึงโครงสร้างฮาร์ดแวร์และหลักการทำงานของอุปกรณ์คินเนคต์อย่างละเอียด
- ศึกษาการพัฒนาโปรแกรมโดยใช้ Microsoft Visual Studio 2010

- ทดสอบการพัฒนาโปรแกรมอย่างง่ายเพื่อติดต่อกับอุปกรณ์คินเนคต์
- ศึกษาการพัฒนาโปรแกรมที่ประมวลผลและแสดงผลกราฟิก 3 มิติ
- พัฒนาตัวอย่างโปรแกรมที่มีส่วนติดต่อผู้ใช้แบบ 3 มิติ โดยผู้ใช้สามารถใช้งานผ่านอุปกรณ์คินเนคต์
- ศึกษาการใช้งานอุปกรณ์คินเนคต์ 2 ชุดพร้อมกัน
- พัฒนาตัวอย่างโปรแกรมที่ใช้งานอุปกรณ์คินเนคต์ 2 ชุดพร้อมกัน
- ศึกษาวิเคราะห์ผล และเขียนรายงานสรุป

บทที่ 2
การดำเนินการวิจัย

บทนี้จะกล่าวถึงโครงสร้างภายใน หลักการทำงาน รวมทั้งการเขียนโปรแกรมเพื่อสั่งงานอุปกรณ์คินเนคต์ การพัฒนาโปรแกรมเพื่อแสดงผลภาพสามมิติ

2.1 โครงสร้างอุปกรณ์คินเนคต์

คินเนคต์ประกอบด้วยส่วนประกอบสำคัญคือ กล้องวิดีโอ เซ็นเซอร์วัดความลึก มัลติอาเรย์ไมโครโฟน และมอเตอร์สำหรับปรับทิศทาง (ภาพที่ 2.1) นอกจากจะสามารถรับรู้ท่าทางการเคลื่อนไหวของผู้ใช้แล้ว คินเนคต์ยังสามารถรับรู้สีหน้าของผู้ใช้ (Facial Expression Recognition) หาตำแหน่งของแหล่งกำเนิดเสียง (Acoustic Source Localization) และรับรู้คำพูด (Speech Recognition) ได้

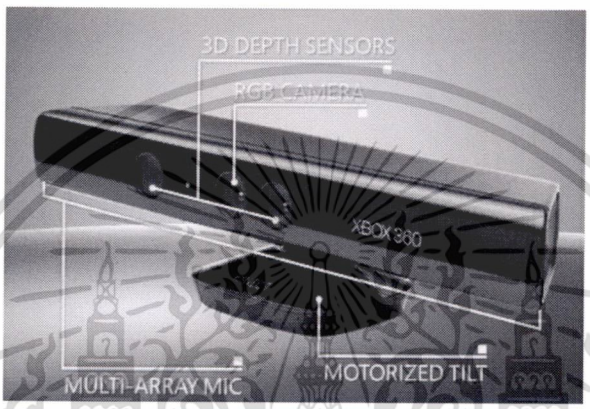
ข้อมูลทางเทคนิคพื้นฐานของอุปกรณ์คินเนคต์สามารถสรุปได้ดังตารางที่ 2.1

ตารางที่ 2.1 ข้อมูลทางเทคนิคของอุปกรณ์คินเนคต์ [5]

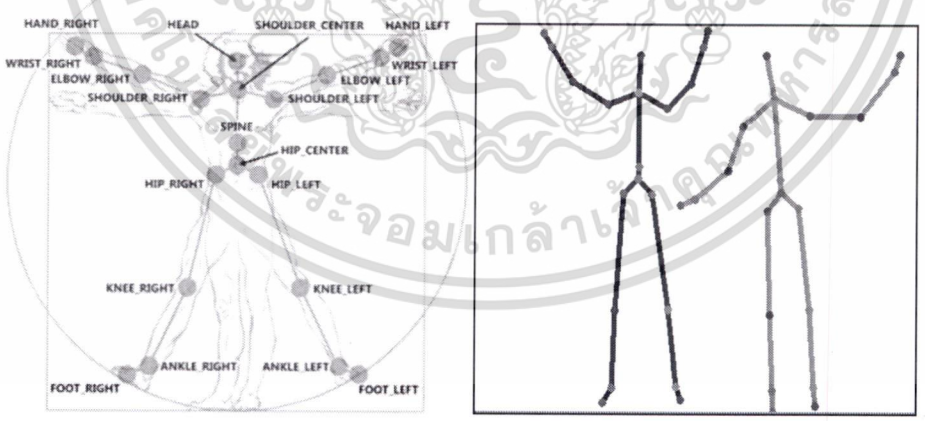
กล้องวิดีโอ	RGB with Bayer color filter
สตรีมข้อมูลจากกล้องวิดีโอ	VGA (640 x 480), 32-bit colors, 30 FPS
เซ็นเซอร์วัดความลึก	Infrared laser projector + monochrome CMOS sensor
สตรีมข้อมูลความลึก	QVGA (320 x 240), 16-bit depth, 30 FPS
ไมโครโฟน	A four-microphone array with 24-bit analog-to-digital converter (ADC) and internal signal processing including acoustic echo cancellation and noise suppression
สตรีมข้อมูลเสียง	16-kHz, 16-bit mono pulse code modulation (PCM)
ระยะปฏิบัติงาน	1.2 – 3.5 meters
ความกว้างของมุมมอง	43° vertical by 57° horizontal field of view
การปรับองศาของกล้องในแนวตั้ง	±27°

ข้อมูลที่คินเนคต์ส่งออกมาจะถูกประมวลผลโดยซอฟต์เวอร์ไดรเวอร์ซึ่งติดต่อบนเครื่องคอมพิวเตอร์ โปรแกรมเมอร์สามารถสั่งงานและดึงข้อมูลจากคินเนคต์มาใช้งานผ่าน API ซึ่งมาพร้อมกับชุด Kinect for Windows SDK จุดเด่นของชุด API ของคินเนคต์ (ซึ่งเรียกว่า Natural User Interface API หรือ NUI API) คือสามารถวิเคราะห์โครงสร้างทางกายภาพของผู้ใช้งานที่อยู่ภายในมุมมองของกล้อง โดยสามารถระบุตำแหน่งของจุดข้อต่อที่สำคัญบนร่างกายของผู้ใช้จำนวน 20 จุด เรียกว่า Skeleton Joint Positions (ภาพที่ 2.2) และยังสามารถติดตามการเคลื่อนที่ของจุดเหล่านั้น พร้อมทั้งส่งข้อมูลมาให้

โปรแกรมในรูปแบบของสตรีมของตำแหน่งของแต่ละจุด API ของคินเนคต์สามารถติดตาม Skeleton Joint Positions ของผู้ใช้งานที่อยู่ในมุมมองกล้องได้ถึงสองคนในเวลาเดียวกัน ข้อมูลเหล่านี้จะถูกใช้สำหรับรับรู้ท่าทางและการเคลื่อนไหวของผู้ใช้งานโปรแกรม โดยผู้พัฒนาโปรแกรมไม่จำเป็นต้องเขียนโปรแกรมเพื่อประมวลผลสตรีมของภาพและความลึกที่ส่งมาจากกล้องและเซ็นเซอร์วัดความลึกในคินเนคต์ด้วยตนเอง แต่หากผู้พัฒนาโปรแกรมต้องการก็สามารถทำได้โดยอ่านสตรีมของภาพและความลึกผ่าน NUI API



ภาพที่ 2.1 ส่วนประกอบภายนอกของคินเนคต์ [5]

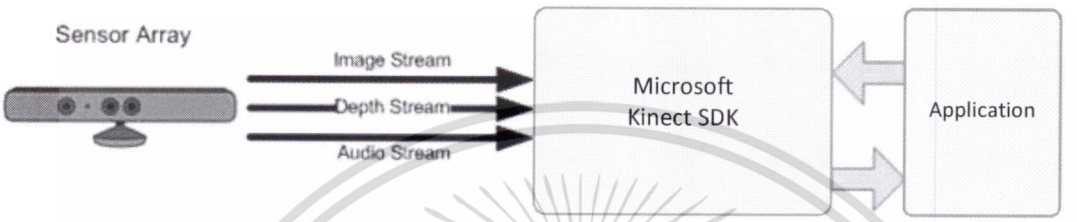


ภาพที่ 2.2 ตัวอย่าง Skeleton Joint Position ซึ่งระบุตำแหน่งสำคัญบนร่างกายจำนวน 20 จุด [5]

2.2 การเขียนโปรแกรมใช้งานอุปกรณ์คินเนคต์

2.2.1 ไลบรารี Microsoft Kinect SDK

Microsoft Kinect SDK ประกอบด้วยซอฟต์แวร์ไลบราลีและเครื่องมือสำหรับช่วยพัฒนาแอพพลิเคชันที่ใช้งานอุปกรณ์คินเนคต์ โดยทำหน้าที่เป็นส่วนเชื่อมต่อระหว่างแอพพลิเคชันกับตัวอุปกรณ์ ดังภาพที่ 2.3



ภาพที่ 2.3 การเชื่อมต่อระหว่างอุปกรณ์คินเนคต์, Kinect SDK, และแอพพลิเคชัน
(ภาพจาก http://msdn.microsoft.com/)

ซอฟต์แวร์ไลบราลีที่มาพร้อมกับ Microsoft Kinect SDK มีทั้งที่ออกแบบมาเพื่อใช้พัฒนาแอพพลิเคชันโดยใช้ภาษา C++ สำหรับแอพพลิชันที่ทำงานแบบ Native บน Windows และที่ใช้ภาษา C# หรือ C++ สำหรับแอพพลิเคชันที่ทำงานบน Common Language Runtime (CLR) Virtual Machine ส่วนที่สำคัญของไลบราลีมีดังนี้

1. Natural User Interface (NUI) API ใช้สำหรับสั่งงานและอ่านสตรีมของข้อมูล Skeleton (ซึ่งประกอบด้วยตำแหน่งของข้อต่อในร่างกายของผู้ใช้) สตรีมของข้อมูลภาพและความลึกของจุดในภาพ สตรีมของเสียง
2. Audio API ใช้สำหรับสั่งงานและอ่านสตรีมของข้อมูลเสียงจากไมโครโฟนอาร์เรย์ในอุปกรณ์คินเนคต์ และข้อมูลตำแหน่งของแหล่งกำเนิดเสียง
3. Kinect Developer Toolkit ประกอบไปด้วยไลบราลีและเครื่องมือที่เป็นประโยชน์ในงานประยุกต์ต่างๆ ดังนี้ ไลบราลีสำหรับวิเคราะห์ใบหน้าและสีหน้าของผู้ใช้ ไลบราลีสำหรับวิเคราะห์มือของผู้ใช้ในการโต้ตอบกับแอพพลิเคชัน และไลบราลีสำหรับช่วยสร้างแบบจำลองสามมิติของผู้ใช้

2.2.2 การสั่งงานอุปกรณ์คินเนคต์ผ่าน Microsoft Kinect SDK

ตัวอย่างโค้ดโปรแกรมสำหรับค้นหาอุปกรณ์คินเนคต์ที่เชื่อมต่อกับเครื่องคอมพิวเตอร์ตอนที่แอพพลิเคชันกำลังเริ่มทำงาน

```
// Find the first Kinect and connect to it
// Adapted from sample code available at
// http:// http://www.microsoft.com/en-us/kinectforwindows/develop/

HRESULT MyApplication::CreateFirstConnected()
{
    INuiSensor * pNuiSensor;

    int iSensorCount = 0;
    HRESULT hr = NuiGetSensorCount(&iSensorCount);
    if (FAILED(hr)) return hr;


    // Look at each Kinect sensor
    for (int i = 0; i < iSensorCount; ++i)
    {
        hr = NuiCreateSensorByIndex(i, &pNuiSensor);
        if (FAILED(hr)) continue;

        hr = pNuiSensor->NuiStatus();
        if (S_OK == hr)
        {
            m_pNuiSensor = pNuiSensor;
            break;
        } else {
            pNuiSensor->Release();
        }
    }
    if (NULL != m_pNuiSensor)
    {
        // Initialize the Kinect
        hr = m_pNuiSensor->NuiInitialize(NUI_INITIALIZE_FLAG_USES_SKELETON);

        if (SUCCEEDED(hr))
        {
            //Create an event that will be signaled when
            //skeleton data is available
            m_hNextSkeletonEvent = CreateEventW(NULL, TRUE, FALSE, NULL);

            // Open a skeleton stream to receive skeleton data
            hr = m_pNuiSensor->
                        NuiSkeletonTrackingEnable(m_hNextSkeletonEvent, 0);
        }
    }
    if (NULL == m_pNuiSensor || FAILED(hr))
    {
        SetStatusMessage(L"No ready Kinect found!");
        return E_FAIL;
    }
    return hr;
}
```

ตัวอย่างโค้ดโปรแกรมสำหรับอ่านข้อมูล Skeleton จากอุปกรณ์

```
// Call UpdateKinectST on each iteration of the application's update loop.
// Adapted from sample code available at
// http:// http://www.microsoft.com/en-us/kinectforwindows/develop/
void MyApplication::UpdateKinectST()
{
   // Wait for 0ms, just quickly test if it is time to process a skeleton
   if ( WAIT_OBJECT_0 == WaitForSingleObject(m_hNextSkeletonEvent, 0) )
   {
     NUI_SKELETON_FRAME skeletonFrame = {0};

     // Get the skeleton frame that is ready
     if (SUCCEEDED(m_pNuiSensor->NuiSkeletonGetNextFrame(0, &skeletonFrame)))
```

```
      {
        // Process the skeleton frame
        SkeletonFrameReady(&skeletonFrame);
      }
    }
}

// Draws the skeletons in the skeleton data obtained from Kinect
void MyApplication::SkeletonFrameReady(NUI_SKELETON_FRAME* pSkeletonFrame)
  {
  for (int i = 0; i < NUI_SKELETON_COUNT; i++)
    {
        const NUI_SKELETON_DATA & skeleton = pSkeletonFrame->SkeletonData[i];

        switch (skeleton.eTrackingState)
        {
        case NUI_SKELETON_TRACKED:
            DrawTrackedSkeletonJoints(skeleton);
            break;
        case NUI_SKELETON_POSITION_ONLY:
            DrawSkeletonPosition(skeleton.Position);
            break;
        }
    }
}
```

2.2.3 การใช้งานอุปกรณ์คินเนคต์ 2 ชุดพร้อมกัน

อุปกรณ์คินเนคต์มีระยะปฏิบัติการ (Operating range) ที่จำกัด ดังแสดงในตารางที่ 2.2

ตารางที่ 2.2 ข้อมูลระยะปฏิบัติการของอุปกรณ์คินเนคต์ [1]

Horizontal viewing angle	57°
Vertical viewing angle	43°
Distance from sensor	0.8m – 4m (normal mode) 0.4m – 3m (near mode)

คินเนคต์จะสามารถคำนวณตำแหน่งของข้อต่อในร่างกายของผู้ใช้ได้แม่นยำเมื่อ

(1) ผู้ใช้อยู่ห่างจากอุปกรณ์ในระยะที่เหมาะสม ระยะระหว่าง 1.5m ถึง 3.0m เป็นระยะที่จะได้ข้อมูลที่แม่นยำที่สุด

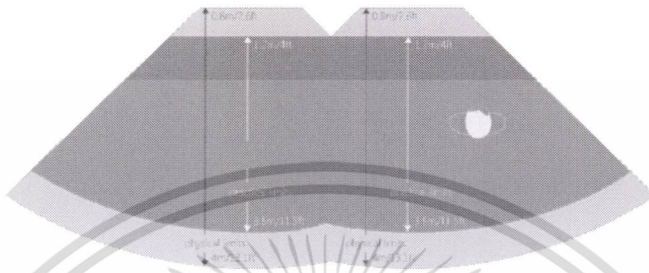
(2) ผู้ใช้ยืนอยู่ในแนวตรงกับอุปกรณ์ คือไม่อยู่ทางด้านซ้ายหรือด้านขวาของอุปกรณ์มากเกินไป

(3) อุปกรณ์ไม่อยู่ในระดับที่สูงหรือต่ำกว่าผู้ใช้มากเกินไป และ

(4) ผู้ใช้ควรยืนหันร่างกายด้านหน้าเข้าหาอุปกรณ์ ถ้ายืนหันข้างให้ ข้อมูลที่ได้จะไม่แม่นยำ

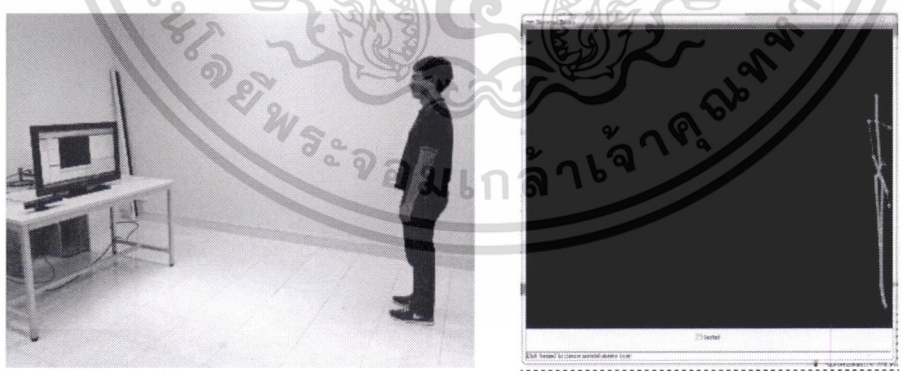
การใช้อุปกรณ์คินเนคต์มากกว่าหนึ่งชุดสามารถช่วยเพิ่มระยะการทำงานของระบบ และในทางทฤษฎียังสามารถช่วยให้ได้ข้อมูลตำแหน่งของข้อต่อในร่างกายที่ถูกต้องและแม่นยำยิ่งขึ้น ใน

โครงการ ผู้วิจัยได้ทดลองใช้งานอุปกรณ์คินเนคต์จำนวน 2 ชุด โดยวางอุปกรณ์ทั้งสองในระดับความสูงเดียวกัน ในแนวเดียวกัน แต่ห่างออกจากกันระยะหนึ่ง (ตามภาพที่ 2.4) การจัดวางแบบนี้จะช่วยเพิ่มพื้นที่ในแนวกว้างที่โปรแกรมจะสามารถหาตำแหน่งของผู้ใช้และข้อมูลตำแหน่งข้อต่อได้อย่างแม่นยำ

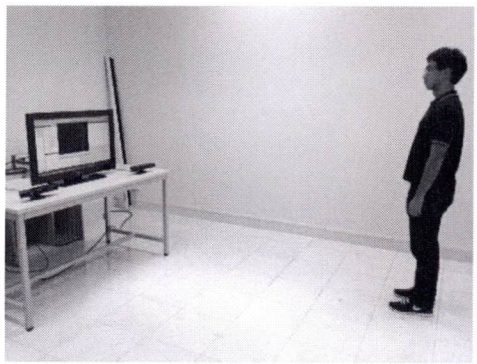


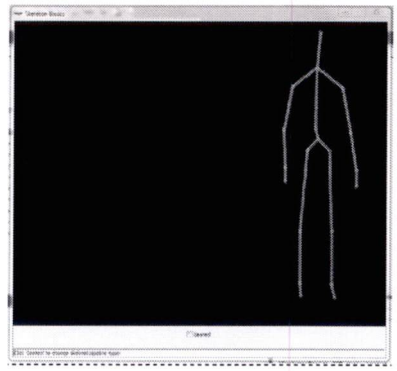
ภาพที่ 2.4 การใช้ข้อมูลจากอุปกรณ์คินเนคต์ 2 ชุดที่วางห่างกันระยะหนึ่งสามารถช่วยเพิ่มพื้นที่ในแนวกว้างที่โปรแกรมสามารถหาข้อมูลตำแหน่งข้อต่อของผู้ใช้ได้อย่างแม่นยำ

ผู้วิจัยได้ทดลองสร้างโปรแกรมที่อ่านค่าข้อมูลตำแหน่งข้อต่อในร่างกายของผู้ใช้จากคินเนคต์ทั้งสองชุด แต่เลือกใช้ข้อมูลจากตัวอุปกรณ์ที่ผู้ใช้ยืนอยู่ใกล้กว่า ผลการทดลองเป็นไปตามที่คาดคือ พื้นที่ที่โปรแกรมสามารถหาข้อมูลตำแหน่งข้อต่อได้ถูกต้องกว้างมากขึ้น ดังแสดงให้เห็นในภาพที่ 2.5 และภาพที่ 2.6



ภาพที่ 2.5 กรณีที่ใช้อุปกรณ์คินเนคต์เพียงชุดเดียว ภาพทางด้านซ้ายแสดงการทดลองให้ผู้ใช้ยืนอยู่ใกล้กับขอบของระยะปฏิบัติงานของอุปกรณ์ จะเห็นว่าข้อมูลตำแหน่งข้อต่อที่อุปกรณ์คำนวณมาได้มีความผิดพลาดสูง





ภาพที่ 2.6 กรณีที่ใช้อุปกรณ์คินเนคต์ 2 ชุดดังรูป โดยให้ผู้ใช้ยืนอยู่ในตำแหน่งเดียวกันกับในการทดลองแรก จะเห็นว่าข้อมูลตำแหน่งข้อต่อที่โปรแกรมคำนวณมาได้มีความถูกต้องมากกว่า

2.3 การพัฒนาโปรแกรมที่ประมวลผลและแสดงผลภาพสามมิติ

ผู้วิจัยได้ศึกษาการใช้งานเฟรมเวิร์กต่างสำหรับการพัฒนาแอพพลิเคชันซึ่งมีส่วนติดต่อผู้ใช้แบบกราฟิกดังต่อไปนี้

2.3.1 Windows Form และ Windows Presentation Foundation (WPF)

ทั้งสองเฟรมเวิร์คนี้เป็นเฟรมเวิร์คสำหรับพัฒนาแอพพลิเคชันที่มีส่วนติดต่อผู้ใช้แบบมาตรฐาน เช่น มี Button, Text Field, Scroll Bar ฯลฯ Windows Presentation Foundation (WPF) ถูกพัฒนาขึ้นมาภายหลัง Windows Form โดยปรับสถาปัตยกรรมให้แยกส่วนของ View และส่วน Controller ออกจากกันอย่างชัดเจน และยังเพิ่มฟังก์ชันในการแสดงผลที่มีความสามารถมากขึ้น

2.3.2 Microsoft XNA

เป็นเฟรมเวิร์กที่สร้างขึ้นมาเพื่อช่วยให้การพัฒนาเกมและแอพพลิเคชันที่มีการแสดงผลภาพเคลื่อนไหวทำได้สะดวก ง่าย และรวดเร็วมากขึ้น Microsoft XNA ทำงานบนพื้นฐานของเทคโนโลยี Microsoft Direct X และถูกออกแบบเพื่อให้การแสดงผลภาพเคลื่อนไหวทั้ง 2 มิติ และ 3 มิติมีประสิทธิภาพสูง (สูงกว่า Windows Presentation Foundation) แต่มีความซับซ้อนน้อยกว่าการใช้งานไลบรารี Microsoft Direct X โดยตรง Microsoft XNA ไม่มีคลาสสำหรับสร้างส่วนติดต่อผู้ใช้มาตรฐาน (เช่น Button, Text Field, Scroll Bar ฯลฯ) ทำให้การพัฒนาแอพพลิเคชันที่มีส่วนติดต่อผู้ใช้แบบมาตรฐานไม่ง่ายเท่ากับ Windows Form หรือ Windows Presentation Framework

2.3.3 OpenGL [3, 4]

OpenGL เป็นกราฟิก API ตัวหนึ่งซึ่งทรงพลังและเป็นที่นิยมใช้กันอย่างแพร่หลาย OpenGL สนับสนุนงานกราฟิกคุณภาพสูงทั้งแบบสองมิติและแบบสามมิติ มีฟังก์ชันต่างๆ มากมาย ยกตัวอย่างเช่น ฟังก์ชันในการจัดการกับโมเดลของวัตถุ การแปลงทางเรขาคณิต การจัดแสง การจัดมุมมองของภาพ

รวมทั้งการติดต่อกับอุปกรณ์ควบคุมอย่างเช่น เมาส์ และ คีย์บอร์ด เป็นต้น ในโครงการวิจัยนี้ ผู้วิจัยได้ศึกษาการใช้งาน OpenGL เบื้องต้น (ร่วมกับนักศึกษาในหลักสูตรวศ.บ. วิศวกรรมซอฟต์แวร์ (นานาชาติ)) เพื่อนำมาใช้ในการประมวล และแสดงผลฟังก์ชันสองตัวแปร $z = f(x, y)$

โค้ดภาษา C++ ต่อไปนี้เป็นตัวอย่างการประกาศคลาสสำหรับพล็อตฟังก์ชัน $z = \cos(x) + \sin(y)$ โดย $-5 \leq x \leq 5$ และ $-5 \leq y \leq 5$ ซึ่งในโค้ดดังกล่าวจะมีการประกาศเวกเตอร์ (อาเรย์ชนิดหนึ่งในภาษา C++) ชื่อ `vertices` สำหรับเก็บค่าพิกัด x, y, z ของจุดที่จะพล็อตและเวกเตอร์ชื่อ `indices` สำหรับเก็บลำดับของแต่ละพิกัด

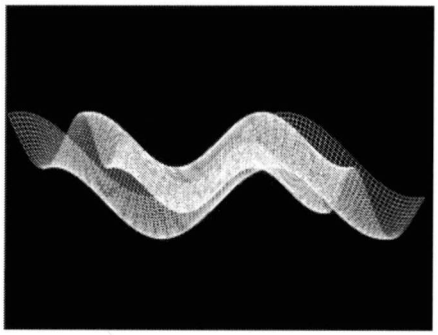
```
class Surface
{
protected:
	std::vector<GLfloat> vertices;
	std::vector<GLushort> indices;
	float xMin, xMax;
	float yMin, yMax;
	float zMin, zMax;
	int xCount, yCount;
	float scale;
	float increment;

	float f(float x, float y);
	float max2(float x, float y);
	float max3(float x, float y, float z);
public:
	Surface();
	void draw(GLfloat x, GLfloat y, GLfloat z);
};
```

การสร้างโมเดลของฟังก์ชันสองตัวแปรถูกกำหนดในคอนสตรัคเตอร์ของคลาส Surface ดังต่อไปนี้

```
Surface::Surface()
{
	xMin = -5; xMax = 5;
	yMin = -5; yMax = 5;
	zMin = 100000; zMax = -100000;
	increment = 0.1;
	xCount = (int)((xMax - xMin)/increment);
	yCount = (int)((yMax - yMin)/increment);

	vertices.resize((xCount+1) * (yCount+1) * 3);
	std::vector<GLfloat>::iterator v = vertices.begin();
	float x=xMin;
	float z;
	for(int i=0;  i<=xCount ; i++, x+=increment) {
		float y=yMin;
		for(int j=0; j<=yCount; j++, y+=increment) {
			z = f(x,y);
			//find zMax and zMin
			if( z <= zMin ) zMin = z;
			if( z >= zMax ) zMax = z;
			//assign each point (x, y, z) to vector
```

```
                *v++ = y;
                *v++ = z;
                *v++ = -x;
            }
        }

        indices.resize(xCount * yCount * 4);
        std::vector<GLushort>::iterator it = indices.begin();
        for(int i=0; i<xCount ; i++) {
            for(int j=0; j<yCount; j++) {
                *it++ = i*(yCount+1) + j;
                *it++ = i*(yCount+1) + j + 1;
                *it++ = (i+1)*(yCount+1) + j + 1;
                *it++ = (i+1)*(yCount+1) + j;
            }
        }
}

void Surface::draw(GLfloat x, GLfloat y, GLfloat z)
{
        glMatrixMode(GL_MODELVIEW);
        glPushMatrix();

        glTranslatef(x,y,z+Z_OFFSET);

        scale=1/max3( max2(abs(xMax),abs(xMin)),
                      max2(abs(yMax),abs(yMin)),
                      max2(abs(zMax),abs(zMin)) );
        glScalef(scale, scale, scale);

        glEnableClientState(GL_VERTEX_ARRAY);

        glVertexPointer(3, GL_FLOAT, 0, &vertices[0]);
        glDrawElements(GL_QUADS, indices.size(), GL_UNSIGNED_SHORT,
                                                    &indices[0]);
        glPopMatrix();
}
```

Nested loop แรกในโค้ดข้างต้นทำการคำนวณค่า `z` จากเมธอด `f(x, y)` และทำการเก็บค่าพิกัดทั้งสามค่าลงในเวกเตอร์ `vertices` ในขณะที่ Nested loop อันถัดมาใช้ในการกำหนดลำดับของพิกัดลงในเวกเตอร์ `indices` ทั้งนี้เพื่อกำหนดลำดับที่ถูกต้องในการพล็อตสมการ เนื่องจากผิวโค้งใน OpenGL นั้นประกอบไปด้วยพื้นผิวสี่เหลี่ยมมาประกอบกันเป็นจำนวนมาก การวาดผิวโค้งจะเริ่มการการวาดสี่เหลี่ยมหนึ่งอันซึ่งผู้วิจัยจะต้องระบุตำแหน่งพิกัดของมุมสี่มุมที่จะวาด โดยผ่านทางเวกเตอร์ `vertices` และ `indices` หลังจากนั้นก็ทำการวาดสี่เหลี่ยมอันต่อไปจนกว่าจะเสร็จสิ้น ภาพที่ 2.7 เป็นตัวอย่างสมการที่วาดขึ้นมาโดยใช้โมเดลในโค้ดดังกล่าว



ภาพที่ 2.7 ผิวโค้งจากสมการ $z = \cos(x) + \sin(y)$

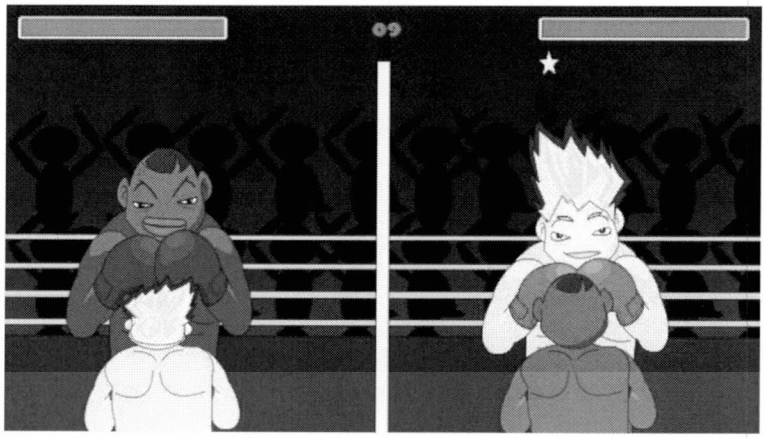
บทที่ 3
อภิปรายและวิจารณ์ผลการทดลอง

ในบทนี้ผู้วิจัยจะกล่าวถึงตัวอย่างแอพพลิเคชันที่ใช้งานอุปกรณ์คินเนคต์ที่ได้ออกแบบและพัฒนาขึ้นในโครงการวิจัยนี้ (ร่วมกับนักศึกษาชั้นปีที่ 3 ของหลักสูตร วศ.บ. วิศวกรรมซอฟต์แวร์ (นานาชาติ)) ได้แก่ แอพพลิเคชันเกมจำลองการชกมวย แอพพลิเคชันสำหรับฝึกท่ารำในการรำไทย แอพพลิเคชันสำหรับสนับสนุนการเรียนการสอนวิชาคณิตศาสตร์ นอกจากนี้ผู้วิจัยจะอภิปรายถึงความเหมาะสมของการประยุกต์ใช้งานอุปกรณ์คินเนคต์และปัญหาที่พบ

3.1 แอพพลิเคชันเกมจำลองการชกมวย

แอพพลิเคชันนี้เป็นแอพพลิเคชันที่ผู้วิจัยออกแบบและพัฒนาขึ้น เป็นเกมจำลองการแข่งขันชกมวยทั้งในรูปแบบการประลองระหว่างผู้เล่นกับตัวละครที่ควบคุมโดยแอพพลิเคชัน (ภาพที่ 3.1) และการประลองระหว่างผู้เล่นสองคน (ภาพที่ 3.2) ผู้เล่นสามารถบังคับตัวละครได้โดยการทำท่าทางด้านหน้าอุปกรณ์คินเนคต์เสมือนกับกำลังแข่งขันชกมวยจริง

ภาพที่ 3.1 ภาพสกรีนช็อตจากการเล่นเกมประลองการชกมวยกับตัวละครศัตรูซึ่งบังคับโดยคอมพิวเตอร์



ภาพที่ 3.2 ภาพสกรีนช็อตจากการเล่นเกมซึ่งผู้เล่นสองคนบังคับตัวละครของตนเองเพื่อแข่งขันชกมวย

หลักการทำงาน

โปรแกรมอ่านท่าทางการเคลื่อนไหวของข้อต่อบนร่างกายของผู้ใช้ผ่านอุปกรณ์คินเนคต์ แล้วนำข้อมูลมาวิเคราะห์ว่าเป็นท่าทางต่อไปนี้หรือไม่

a) ชกด้วยหมัดซ้าย
b) ชกด้วยหมัดขวา
c) หลบโดยเบี่ยงตัวไปทางด้านซ้าย
d) หลบโดยเบี่ยงตัวไปทางด้านขวา
e) ยกแขนขึ้นกัน (บล็อค)

ถ้าพบว่าผู้ใช้ทำท่าทางข้างต้น โปรแกรมจะแสดงภาพเคลื่อนไหวของตัวละครที่ผู้เล่นบังคับในท่านั้นๆ ซึ่งภาพเคลื่อนไหวของตัวละครทั้ง 5 ท่าทางข้างต้น ได้ถูกสร้างไว้ก่อนหน้าและเก็บไว้ในลักษณะเฟรมของรูปภาพ

ความเหมาะสมและปัญหาที่พบ

จากการทดลองใช้งานแอพพลิเคชันโดยนักศึกษาและคณาจารย์ในวิทยาลัยนานาชาติจำนวนมากกว่า 10 คนพบว่าโดยรวมแอพพลิเคชันเกมจำลองการชกมวยที่พัฒนาขึ้นสามารถใช้งานได้ดีอย่างเป็นที่น่าพอใจ ผู้เล่นสามารถบังคับตัวละครได้อย่างเป็นธรรมชาติ การตอบสนองต่อท่าทางอยู่ในเกณฑ์ที่ผู้ทดลองใช้งานทุกคนยอบรับได้ จึงสามารถสรุปได้ว่าอุปกรณ์คินเนคต์มีความเหมาะสมในการใช้ควบคุมตัวละครในเกมจำลองการชกมวย

ปัญหาที่พบมีดังนี้

1. การที่โปรแกรมจะสามารถวิเคราะห์ท่าทางของผู้เล่นได้นั้น ผู้เล่นจะต้องเริ่มทำท่าทางนั้นไปแล้ว ซึ่งเป็นผลให้การแสดงผลท่าทางตัวละครบนหน้าจอช้ากว่าการแสดงท่าทางของผู้เล่นเล็กน้อย ซึ่งในการทดลองใช้งานจริงพบว่าความล่าช้าที่เกิดขึ้นอยู่ในเกณฑ์ที่ยอบรับได้

2. โปรแกรมถูกออกแบบมาให้สามารถอ่านท่าทางของผู้เล่น 5 ท่าทางดังกล่าวข้างต้นเท่านั้น ในการใช้งานจริงผู้ที่ใช้งานที่ไม่ทราบจะพยายามทำท่าทางที่นอกเหนือจากท่าทางดังกล่าว ซึ่งเป็นผลให้ภาพเคลื่อนไหวของตัวละครไม่เป็นไปตามที่ผู้ใช้งานคาดหวัง คือในบางครั้งตัวละครไม่มีการเคลื่อนไหวเพราะโปรแกรมวิเคราะห์แล้วพบว่าการเคลื่อนไหวของผู้เล่นไม่ตรงกับท่าทางที่กำหนด หรือในบางครั้งตัวละครอาจเคลื่อนไหวในท่าที่ไม่ตรงกับท่าทางของผู้เล่น เพราะการเคลื่อนไหวของผู้เล่นมีส่วนซึ่งคล้ายกับหนึ่งในท่าทางที่กล่าวข้างต้น (เกิด False Positive)
3. การเก็บภาพเคลื่อนไหวของท่าทางของตัวละคร ในรูปแบบของเฟรมของภาพเป็นวิธีที่สิ้นเปลืองหน่วยความจำเป็นอย่างมาก (ในเกมมีตัวละครทั้งหมด 3 ตัวละคร แต่ละตัวละครมี 5 ท่าทาง แต่ละท่าทางมีเฟรมของภาพประมาณ 20 ภาพ รวมทั้งสิ้นมีภาพทั้งหมดประมาณ 300 ภาพ) และเป็นผลให้แอพพลิเคชันใช้เวลาค่อนข้างนานเมื่อเริ่มทำงานในครั้งแรก (ประมาณ 1 นาที) อีกทั้งแอพพลิเคชันยังต้องใช้หน่วยความจำของเครื่องมากเกินควร (ประมาณ 4 GB) วิธีหนึ่งในแก้ปัญหานี้คือใช้การบีบอัดข้อมูลภาพในหน่วยความจำ และคืนหน่วยความจำของภาพตัวละครที่โปรแกรมยังไม่แสดงผลเพื่อช่วยลดการใช้งานหน่วยจำของเครื่อง

3.2 แอพพลิเคชันสำหรับฝึกท่ารำในการรำไทย

แอพพลิเคชันนี้มีวัตถุประสงค์เพื่อช่วยให้ผู้ใช้เรียนรู้และฝึกฝนท่ารำมาตรฐานของการรำไทยที่ถูกต้อง โดยในการใช้งาน แอพพลิเคชันจะกำหนดชื่อท่าเพื่อให้ผู้ใช้ทำท่ารำนั้นด้านหน้าอุปกรณ์คินเนคต์ แล้วแอพพลิเคชันจะให้ข้อมูลกับผู้ใช้ว่าท่าทางของผู้ใช้ถูกต้องมากน้อยเพียงใด และมีส่วนใดของร่างกายที่อยู่ในตำแหน่งที่ไม่ถูกต้อง ภาพที่ 3.3 แสดงตัวอย่างการใช้งานแอพพลิเคชัน

<u>หลักการทำงาน</u>

ในระหว่างการพัฒนา ผู้พัฒนาได้ทำการเก็บตำแน่งข้อต่อของผู้รำแม่แบบในท่ารำมาตรฐานที่ทำได้อย่างถูกต้องโดยอ่านข้อมูลผ่านอุปกรณ์คินเนคต์ แล้วบันทึกข้อมูลนี้ในรูปแบบของไฟล์ข้อความ (Text File) และเมื่อผู้ใช้ใช้งานโปรแกรมเพื่อฝึกฝนท่ารำ โปรแกรมจะอ่านข้อมูลตำแหน่งของข้อต่อบนร่างกายของผู้ใช้ผ่านทางอุปกรณ์คินเนคต์ แล้วนำมาเปรียบเทียบว่าตรงกับข้อมูลตำแหน่งที่จัดเก็บไว้ก่อนหน้านี้หรือไม่ โปรแกรมจะแสดงภาพบนหน้าจอซึ่งแสดงให้เห็นตำแหน่งของข้อต่อบนร่างกายของผู้ใช้ ข้อต่อใดที่ตำแหน่งไม่ตรงกับตำแหน่งที่ควรจะเป็น โปรแกรมจะแสดงภาพข้อต่อเป็นจุดสีแดง

ภาพที่ 3.3 ภาพสกรีนช็อตจากแอพพลิเคชันสำหรับการฝึกท่ารำไทย

ความเหมาะสมและปัญหาที่พบ

จากการทดลองใช้งานแอพพลิเคชันพบว่าแอพพลิเคชันสามารถอ่านท่ารำของผู้ใช้และนำมาวิเคราะห์แล้วให้ข้อมูลความถูกต้องของท่ารำแก่ผู้ใช้ได้ แต่ความแม่นยำในการระบุระดับความถูกต้องของท่ารำยังอยู่ในเกณฑ์ต่ำ ผู้วิจัยพบว่าปัจจัยสำคัญที่ทำให้เกิดความผิดพลาดมีดังนี้

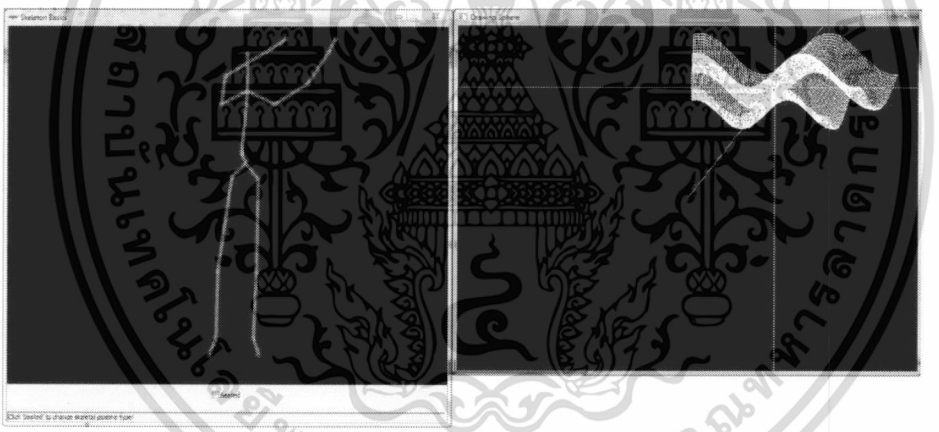
1. สัดส่วนของร่างกายผู้ใช้ไม่ตรงกับสัดส่วนของร่างกายผู้รำแม่แบบ (ผู้ใช้บางคนแขนขายาวกว่าหรือสั้นกว่าผู้รำแม่แบบ) ทำให้แม้ว่าผู้ใช้จะทำท่ารำได้ถูกต้องตามแบบ แต่ตำแหน่งของข้อต่อของผู้ใช้ไม่ตรงกับตำแหน่งข้อต่อของผู้รำแม่แบบ ผู้วิจัยได้แก้ปัญหาในเบื้องต้นโดยให้ผู้ใช้กำหนดว่าผู้ใช้มีรูปร่างสูง (สูง 165 ซม. ขึ้นไป) หรือเตี้ยกว่านั้น แล้วสร้างชุดข้อมูลท่ารำแม่แบบเป็นสองชุดสำหรับผู้ใช้ทั้งสองประเภท
2. ผู้ใช้ยืนอยู่ในตำแหน่งที่ไม่ตรงกับตำแหน่งที่ผู้รำแม่แบบยืน เช่นยืนชิดไปทางด้านซ้ายหรือด้านขวามากเกินไป หรือยืนใกล้หรือไกลอุปกรณ์คินเนคต์มากเกินไป ซึ่งเป็นผลให้ข้อมูลตำแหน่งของข้อต่อของผู้ใช้ไม่ตรงกับข้อมูลของผู้รำแม่แบบ วิธีแก้ไขเบื้องต้นที่ผู้วิจัยได้ทดลองคือเก็บภาพเคล้าโคลง (Silhouette) ของร่างกายผู้รำแม่แบบในแต่ละท่ารำ แล้วนำมาแสดงซ้อนบนภาพวิดีโอที่คินเนคต์ถ่ายผู้ใช้ (ในแบบ Real Time) เพื่อเป็นแนวให้ผู้ใช้ปรับตำแหน่งการยืนเพื่อให้ร่างกายของผู้ใช้บนวิดีโอตรงกับภาพเค้าโครง วิธีที่น่าจะมีประสิทธิภาพมากกว่าคือใช้การเปรียบเทียบตำแหน่งบนพิกัดบนร่างกายผู้ใช้ (User Coordinate) เช่นพิกัดของข้อต่อเมื่อเทียบกับกึ่งกลางของร่างกายผู้รำ แทนที่จะใช้พิกัดอุปกรณ์คินเนคต์ (Device Coordinate) โปรแกรมจะต้องแปลงข้อมูลตำแหน่งของข้อต่อแต่ละข้อต่อซึ่งคินเนคต์ส่งมาเป็นพิกัดบน Device Coordinate เป็นพิกัดบน User Coordinate ก่อนนำข้อมูลมาจัดเก็บหรือวิเคราะห์

3. โปรแกรมไม่ใช้พิกัดในแนวแกน Z ของตำแหน่งข้อต่อ (ซึ่งเป็นระยะห่างของข้อต่อจากอุปกรณ์คินเนคต์) มาพิจารณา จึงเป็นข้อจำกัดทำให้โปรแกรมไม่สามารถใช้สอนท่ารำซึ่งผู้รำต้องยื่นแขนหรือขาไปทางด้านหน้าหรือด้านหลังของร่างกายได้

เห็นได้ว่าปัจจัยดังกล่าวซึ่งส่งผลต่อความแม่นยำในการระบุระดับความถูกต้องของท่ารำของโปรแกรมไม่ใช้เกิดจากข้อจำกัดของอุปกรณ์คินเนคต์ แต่เกิดจากตัวแอพพลิเคชันเองซึ่งผู้วิจัยเห็นว่าหากได้รับพัฒนาต่อเพื่อแก้ปัญหาดังกล่าวจะสามารถนำไปใช้เพื่อเป็นประโยชน์ในการฝึกฝนท่ารำไทยได้จริง

3.3 แอพพลิเคชันสำหรับสนับสนุนการเรียนการสอนวิชาคณิตศาสตร์

แอพพลิเคชันนี้มีวัตถุประสงค์เพื่อช่วยให้นักเรียนนักศึกษามีความสนใจในการเรียนคณิตศาสตร์มากขึ้น ในแอพพลิเคชันดังกล่าว จะมีการพล็อตพื้นผิวจากสมการสองตัวแปร $z = f(x, y)$ โดยใช้โค้ด OpenGL โดยผู้ใช้สามารถปรับเปลี่ยนตำแหน่งของพื้นผิวได้โดยผ่านอุปกรณ์คินเนคต์ ซึ่งช่วยให้การปรับตำแหน่งและมุมมองมีความง่ายดายและเป็นธรรมชาติกว่าการควบคุมวัตถุสามมิติโดยใช้เมาส์หรือว่าคีย์บอร์ด



ภาพที่ 3.4 ภาพสกรีนช็อตขณะใช้งานแอพพลิเคชันสนับสนุนการเรียนการสอนวิชาคณิตศาสตร์

<u>หลักการทำงาน</u>

แอพพลิเคชันใช้ไลบรารี OpenGL ในการวาดพื้นผิวสามมิติจากสมการสองตัวแปร $z = f(x, y)$ ผู้ใช้สามารถปรับตำแหน่งของพื้นผิวบนหน้าจอได้โดยยื่นมือทั้งสองข้างไปด้านหน้าลำตัวให้อยู่ในระดับและระนาบเดียวกันและมือทั้งสองข้างห่างกันประมาณ 1 – 2 ฟุต หลังจากนั้นโปรแกรมจะเริ่มจับตำแหน่ง (Track) ของมือของผู้ใช้ เมื่อผู้ใช้ขยับมือทั้งสองไปพร้อมๆ กัน (ทางซ้าย-ขวา บน-ล่าง และเข้า-ออก) โปรแกรมจะปรับตำแหน่งของพื้นผิวที่วาดบนหน้าจอตามการเคลื่อนที่ของมือผู้ใช้ เมื่อผู้ใช้แยกมือทั้งสองออกจากกัน โปรแกรมจะหยุดการจับตำแหน่ง

ความเหมาะสมและปัญหาที่พบ

จากการทดลองใช้งานแอพพลิเคชันผู้วิจัยพบว่าผู้ใช้สามารถปรับตำแหน่งของพื้นผิวได้รวดเร็วและแม่นยำ แต่ยังไม่สะดวกเท่าที่ด้วยเหตุผลต่อไปนี้

1. หลังจากผู้ใช้สั่งงานโปรแกรมให้ปรับตำแหน่งของภาพโดยใช้มือทั้งสองไปยังตำแหน่งที่ต้องการแล้ว หากผู้ใช้ลืมแยกมือทั้งสองออกจากกัน โปรแกรมจะยังคงจับตำแหน่งของมือผู้ใช้อยู่ ซึ่งทำให้ตำแหน่งของภาพบนหน้าจอเปลี่ยนไป
2. หลังจากผู้ใช้สั่งงานโปรแกรมให้ปรับตำแหน่งของภาพโดยใช้มือทั้งสองแล้ว ในจังหวะที่ผู้ใช้แยกมือทั้งสองออกจากกัน ภาพบนหน้าจอจะเปลี่ยนตำแหน่งไปเล็กน้อย เนื่องจากในจังหวะนั้นโปรแกรมยังคงจับตำแหน่งของมือผู้ใช้อยู่ ผู้วิจัยเห็นว่าจะเหมาะสมกว่าหากผู้ใช้สามารถสั่งให้โปรแกรมเริ่มจับหรือหยุดจับตำแหน่งของมือผู้ใช้โดยใช้ท่าทางที่ไม่มีผลต่อตำแหน่งของมือผู้ใช้ (เช่นใช้การกำมือหรือคลายมือ เป็นต้น) หรือใช้เสียงสั่งการ เป็นต้น
3. ในการจับตำแหน่งมือของผู้ใช้ โปรแกรมจะวัดว่ามือของผู้ใช้เคลื่อนที่ไปในทิศทางใด เป็นระยะเท่าไร เมื่อเทียบกับตำแหน่งของมือเมื่อตอนที่ผู้ใช้เริ่มสั่งการ ซึ่งทำให้ระยะการเปลี่ยนตำแหน่งของภาพบนหน้าจอในการสั่งการแต่ละครั้งถูกจำกัดโดยระยะที่ผู้ใช้สามารถย้ายตำแหน่งของมือไปได้ ซึ่งขึ้นอยู่กับความยาวของแขนของผู้ใช้ ผู้วิจัยเห็นว่าโปรแกรมควรจับความเร็วของการเปลี่ยนตำแหน่งของมือของผู้ใช้ด้วย ถ้าผู้ใช้เปลี่ยนตำแหน่งของมืออย่างรวดเร็ว โปรแกรมจะปรับตำแหน่งของภาพบนหน้าจอในระยะที่มากกว่าในกรณีที่ผู้ใช้เปลี่ยนตำแหน่งของมืออย่างช้าๆ
4. ผู้ใช้โปรแกรมยังไม่สามารถใช้ท่าทางเพื่อสั่งให้โปรแกรมขยายหรือลดขนาดของพื้นผิวบนหน้าจอ หรือหมุนพื้นผิวที่วาดในแกนต่างๆ ได้
5. โปรแกรมยังไม่มีอินเทอร์เฟซสำหรับให้ผู้ใช้กำหนดสมการที่จะใช้วาดได้โดยตรง

ผู้วิจัยเห็นว่าการสั่งให้โปรแกรมปรับตำแหน่งของภาพโดยใช้ตำแหน่งของมือทั้งสองไม่เหมาะสม เพราะทำให้ตำแหน่งของภาพบนหน้าจอเปลี่ยนไปโดยไม่เจตนา การออกแบบท่าทางเพื่อสั่งงานโปรแกรมผ่านคินเนคต์จะต้องคำนึงถึงปัญหานี้ด้วย โดยรวมแอพพลิเคชันยังต้องได้รับการพัฒนาต่อ (ดังที่อธิบายในข้อ 3 – 5) จึงจะสามารถนำไปใช้งานได้จริง

บทที่ 4
สรุปผลและงานวิจัยในขั้นต่อไป

4.1 สรุปผลการวิจัย

ในโครงการวิจัยนี้ ผู้วิจัยได้เรียนรู้หลักการทำงานของอุปกรณ์ ศึกษาและทดลองพัฒนาโปรแกรมเพื่อใช้งานอุปกรณ์คินเนคต์ผ่าน Microsoft Kinect SDK และได้ศึกษาและทดลองพัฒนาแอพพลิเคชันสำหรับสนับสนุนการเรียนการสอนวิชาคณิตศาสตร์ซึ่งมีส่วนติดต่อผู้ใช้แบบสามมิติโดยผู้ใช้สามารถใช้งานผ่านอุปกรณ์คินเนคต์ได้ นอกจากนี้ผู้วิจัยได้ร่วมกับนักศึกษาในหลักสูตรวิศวกรรมซอฟต์แวร์พัฒนาตัวอย่างแอพพลิเคชันเกมจำลองการชกมวย และแอพพลิเคชันสำหรับฝึกท่ารำในการรำไทย ผู้วิจัยสามารถสรุปข้อดีและข้อเสียของการใช้งานอุปกรณ์คินเนคต์ได้ดังนี้

<u>ข้อดี</u>

1. การพัฒนาโปรแกรมที่ติดต่อกับอุปกรณ์คินเนคต์โดยใช้ Microsoft Kinect SDK ทำได้ไม่ยาก ผู้พัฒนาสามารถเลือกภาษาที่ใช้ในการพัฒนาได้ทั้งภาษา C++ และภาษา C#
2. Microsoft Kinect SDK มีไลบรารีที่สามารถตรวจจับตำแหน่งของข้อต่อบนร่างกายผู้ใช้ได้ ผู้พัฒนาโปรแกรมไม่จำเป็นต้องวิเคราะห์ภาพเพื่อตรวจจับตำแหน่งของข้อต่อด้วยตนเอง
3. อุปกรณ์สามารถวัดระยะห่างจากกล้องถึงวัตถุได้ ซึ่งช่วยให้ผู้พัฒนาโปรแกรมสามารถช่วยแยกแยะผู้ใช้งานและวัตถุอื่นๆออกจากกัน
4. Microsoft Kinect SDK สนับสนุนการใช้อุปกรณ์คินเนคต์มากกว่าหนึ่งชุดพร้อมกัน ซึ่งสามารถเพิ่มระยะปฏิบัติการหรือเพิ่มความแม่นยำในการตรวจจับข้อต่อได้ ดังที่อธิบายในหัวข้อ 2.2.3

<u>ข้อเสีย</u>

1. Microsoft Kinect SDK ถูกออกแบบมาสำหรับเครื่องมือการพัฒนาโปรแกรมใน Microsoft Visual Studio บนระบบปฏิบัติการ Microsoft Windows เท่านั้น และโปรแกรมที่พัฒนาขึ้นมาสามารถใช้งานได้บนระบบปฏิบัติการ Microsoft Windows เท่านั้น
2. อุปกรณ์คินเนคต์ ทั้งในด้านฮาร์ดแวร์และซอฟต์แวร์ ถูกออกแบบมาสำหรับการใช้งานของผู้ใช้ที่ยืนห่างจากจอภาพ 1 – 2 เมตร จึงไม่เหมาะกับแอพพลิเคชันซึ่งผู้ใช้ต้องอยู่ใกล้กับจอภาพ เช่นในงานที่ภาพบนจอมีรายละเอียดสูง

3. ด้วยข้อจำกัดทางด้านฮาร์ดแวร์ (คือความละเอียดของกล้อง) และข้อจำกัดของไลบราลี Microsoft Kinect SDK เวอร์ชันที่ผู้วิจัยใช้งาน (รุ่น 1.5) โปรแกรมที่พัฒนาขึ้นยังไม่สามารถจับท่าทางการเคลื่อนไหวของนิ้วมือของผู้ใช้ได้ ซึ่งเป็นผลให้เกิดข้อจำกัดในการออกแบบส่วนติดต่อผู้ใช้ คือต้องออกแบบให้ผู้ใช้สั่งงานผ่านแขน ขา ศรีษะ และลำตัวเท่านั้น
4. การตรวจจับตำแหน่งของข้อต่อยังไม่แม่นยำเท่าที่ควรถึงแม้ว่าผู้ใช้ยืนอยู่ภายในระยะปฏิบัติการของอุปกรณ์ ยกตัวอย่างเช่น ในกรณีที่บางส่วนของร่างกายผู้ใช้ถูกบดบัง

4.2 งานวิจัยในขั้นต่อไป

ผู้วิจัยมีแผนที่จะศึกษาเทคนิคการรู้จำท่าทางการเคลื่อนไหว (Gesture) ต่างๆเพื่อที่ผู้ใช้จะสามารถติดต่อกับโปรแกรมได้อย่างสะดวกและเป็นธรรมชาติมากขึ้น โดยมีประเด็นที่น่าสนใจดังต่อไปนี้

1. เทคนิคและอัลกอริทึมการรู้จำท่าทางการเคลื่อนไหวซึ่งมีความทนทานต่อความผิดพลาดของตำแหน่งข้อต่อ
2. การเปรียบเทียบความคล้ายคลึงของท่าทางการเคลื่อนไหวเพื่อประยุกต์ใช้ในแอพพลิเคชันสำหรับช่วยฝึกฝนท่าทางของผู้ใช้ เช่น การฝึกฝนรำไทย การฝึกฝนท่าทางที่ถูกต้องในกีฬาต่างๆ เป็นต้น
3. การออกแบบท่าทางการเคลื่อนไหวที่ใช้ในการสั่งงานโปรแกรมที่เหมาะสม โดยเฉพาะอย่างยิ่งโปรแกรมที่มีส่วนติดต่อผู้ใช้แบบสามมิติ

เอกสารอ้างอิง

[1] D. A. Bowman et al., "New Directions in 3D User Interfaces", The International Journal of Virtual Reality, 5(2), pp. 3-14, 2006.

[2] D. A. Bowman et al., "3D User Interfaces: New Directions and Perspectives", IEEE Computer Graphics and Applications, 28(6), pp. 20-36, 2008.

[3] D. Shreiner, OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1 (7th Edition), Addison-Wesley Professional, 2009.

[4] J. L. McKesson, "Learning Modern 3D Graphics Programming". Available at "http://www.arcsynthesis.org/gltut/".

[5] Microsoft Research, "Kinect for Windows SDK beta: Programming Guide", July 2011. Available at "http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/".

ภาคผนวก

โค้ดโปรแกรมสนับสนุนการเรียนการสอนวิชาคณิตศาสตร์

```
//------------------------------------------------------------------------------
// KinectGraph.h
//------------------------------------------------------------------------------
// Code adapted from Skeleton Basics-D2D C++ Sample by Microsoft Corp.
// available at http://msdn.microsoft.com/en-us/library/hh973085.aspx
//------------------------------------------------------------------------------

#include "resource.h"
#include "NuiApi.h"
#include "Gesture.h"

class CSkeletonBasics
{
    static const int cScreenWidth  = 320;
    static const int cScreenHeight = 240;
    static const int cStatusMessageMaxLen = MAX_PATH*2;

public:
    /// <summary>
    /// Constructor
    /// </summary>
    CSkeletonBasics();

    /// <summary>
    /// Destructor
    /// </summary>
    ~CSkeletonBasics();

    /// <summary>
    /// Handles window messages, passes most to the class instance to handle
    /// </summary>
    /// <param name="hWnd">window message is for</param>
    /// <param name="uMsg">message</param>
    /// <param name="wParam">message data</param>
    /// <param name="lParam">additional message data</param>
    /// <returns>result of message processing</returns>
    static LRESULT CALLBACK MessageRouter(HWND hWnd, UINT message,
                                          WPARAM wParam, LPARAM lParam);

    /// <summary>
    /// Handle windows messages for a class instance
    /// </summary>
    /// <param name="hWnd">window message is for</param>
    /// <param name="uMsg">message</param>
    /// <param name="wParam">message data</param>
    /// <param name="lParam">additional message data</param>
    /// <returns>result of message processing</returns>
    LRESULT CALLBACK DlgProc(HWND hWnd, UINT message,
                             WPARAM wParam, LPARAM lParam);

    /// <summary>
    /// Creates the main window and begins processing
    /// </summary>
    /// <param name="hInstance"></param>
```

```
    /// <param name="nCmdShow"></param>
    int                     Run(HINSTANCE hInstance, int nCmdShow);

private:
    HWND                    m_hWnd;

    bool                    m_bSeatedMode;

    // Current Kinect
    INuiSensor*             m_pNuiSensor;

    // Skeletal drawing
    ID2D1HwndRenderTarget*   m_pRenderTarget;
    ID2D1SolidColorBrush*    m_pBrushJointTracked;
    ID2D1SolidColorBrush*    m_pBrushJointInferred;
    ID2D1SolidColorBrush*    m_pBrushBoneTracked;
    ID2D1SolidColorBrush*    m_pBrushBoneInferred;
    D2D1_POINT_2F            m_Points[NUI_SKELETON_POSITION_COUNT];

    // Direct2D
    ID2D1Factory*           m_pD2DFactory;

    HANDLE                  m_pSkeletonStreamHandle;
    HANDLE                  m_hNextSkeletonEvent;

    /// <summary>
    /// Main processing function
    /// </summary>
    void                    Update();

    /// <summary>
    /// Create the first connected Kinect found
    /// </summary>
    /// <returns>S_OK on success, otherwise failure code</returns>
    HRESULT                 CreateFirstConnected();

    /// <summary>
    /// Handle new skeleton data
    /// </summary>
    void                    ProcessSkeleton();

    /// <summary>
    /// Ensure necessary Direct2d resources are created
    /// </summary>
    /// <returns>S_OK if successful, otherwise an error code</returns>
    HRESULT                 EnsureDirect2DResources( );

    /// <summary>
    /// Dispose Direct2d resources
    /// </summary>
    void                    DiscardDirect2DResources( );

    /// <summary>
    /// Draws a bone line between two joints
    /// </summary>
    /// <param name="skel">skeleton to draw bones from</param>
    /// <param name="joint0">joint to start drawing from</param>
    /// <param name="joint1">joint to end drawing at</param>
    void                    DrawBone(const NUI_SKELETON_DATA & skel,
                                     NUI_SKELETON_POSITION_INDEX bone0,
                                     NUI_SKELETON_POSITION_INDEX bone1);
```

```
    /// <summary>
    /// Draws a skeleton
    /// </summary>
    /// <param name="skel">skeleton to draw</param>
    /// <param name="windowWidth">width (in pixels) of output buffer</param>
    /// <param name="windowHeight">height (in pixels) of output buffer</param>
    void                    DrawSkeleton(const NUI_SKELETON_DATA & skel,
                                         int windowWidth, int windowHeight);

    /// <summary>
    /// Converts a skeleton point to screen space
    /// </summary>
    /// <param name="skeletonPoint">skeleton point to tranform</param>
    /// <param name="width">width (in pixels) of output buffer</param>
    /// <param name="height">height (in pixels) of output buffer</param>
    /// <returns>point in screen-space</returns>
    D2D1_POINT_2F           SkeletonToScreen(Vector4 skeletonPoint,
                                             int width, int height);

    Vector4                 SkeletonToScreenVector(Vector4 skeletonPoint,
                                                   int width, int height);

    /// <summary>
    /// Set the status bar message
    /// </summary>
    /// <param name="szMessage">message to display</param>
    void                    SetStatusMessage(WCHAR* szMessage);
};

//------------------------------------------------------------------------------
// KinectGraph.cpp
//------------------------------------------------------------------------------
// Code adapted from Skeleton Basics-D2D C++ Sample by Microsoft Corp.
// available at http://msdn.microsoft.com/en-us/library/hh973085.aspx
//------------------------------------------------------------------------------
#include "stdafx.h"
#include <strsafe.h>
#include "KinectGraph.h"
#include "resource.h"
#include "OpenGL.h"

extern float distance;
extern float spherePosX;
extern float spherePosY;
extern float surfaceDistance;
extern float surfacePosX;
extern float surfacePosY;

static const float g_JointThickness = 3.0f;
static const float g_TrackedBoneThickness = 6.0f;
static const float g_InferredBoneThickness = 1.0f;

Gesture gesture;

/// <summary>
/// Entry point for the application
/// </summary>
/// <param name="hInstance">handle to the application instance</param>
/// <param name="hPrevInstance">always 0</param>
/// <param name="lpCmdLine">command line arguments</param>
```

```
/// <param name="nCmdShow">whether to display minimized, maximized, or
normally</param>
/// <returns>status</returns>
int APIENTRY wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPWSTR
lpCmdLine, int nCmdShow)
{
    CSkeletonBasics application;
       initOpenGL();
    application.Run(hInstance, nCmdShow);
}

/// <summary>
/// Constructor
/// </summary>
CSkeletonBasics::CSkeletonBasics() :
    m_pD2DFactory(NULL),
    m_hNextSkeletonEvent(INVALID_HANDLE_VALUE),
    m_pSkeletonStreamHandle(INVALID_HANDLE_VALUE),
    m_bSeatedMode(false),
    m_pRenderTarget(NULL),
    m_pBrushJointTracked(NULL),
    m_pBrushJointInferred(NULL),
    m_pBrushBoneTracked(NULL),
    m_pBrushBoneInferred(NULL),
    m_pNuiSensor(NULL)
{
    ZeroMemory(m_Points,sizeof(m_Points));
}

/// <summary>
/// Destructor
/// </summary>
CSkeletonBasics::~CSkeletonBasics()
{
    if (m_pNuiSensor)
    {
        m_pNuiSensor->NuiShutdown();
    }

    if (m_hNextSkeletonEvent && (m_hNextSkeletonEvent != INVALID_HANDLE_VALUE))
    {
        CloseHandle(m_hNextSkeletonEvent);
    }

    // clean up Direct2D objects
    DiscardDirect2DResources();

    // clean up Direct2D
    SafeRelease(m_pD2DFactory);

    SafeRelease(m_pNuiSensor);
}

/// <summary>
/// Creates the main window and begins processing
/// </summary>
/// <param name="hInstance">handle to the application instance</param>
/// <param name="nCmdShow">whether to display minimized, maximized, or
/// normally</param>
int CSkeletonBasics::Run(HINSTANCE hInstance, int nCmdShow)
{
```

```
    MSG       msg = {0};
    WNDCLASS  wc  = {0};

    // Dialog custom window class
    wc.style         = CS_HREDRAW | CS_VREDRAW;
    wc.cbWndExtra    = DLGWINDOWEXTRA;
    wc.hInstance     = hInstance;
    wc.hCursor       = LoadCursorW(NULL, IDC_ARROW);
    wc.hIcon         = LoadIconW(hInstance, MAKEINTRESOURCE(IDI_APP));
    wc.lpfnWndProc   = DefDlgProcW;
    wc.lpszClassName = L"SkeletonBasicsAppDlgWndClass";

    if (!RegisterClassW(&wc))
    {
        return 0;
    }

    // Create main application window
    HWND hWndApp = CreateDialogParamW(
        hInstance,
        MAKEINTRESOURCE(IDD_APP),
        NULL,
        (DLGPROC)CSkeletonBasics::MessageRouter,
        reinterpret_cast<LPARAM>(this));

    // Show window
    ShowWindow(hWndApp, nCmdShow);

    const int eventCount = 1;
    HANDLE hEvents[eventCount];

    // Main message loop
    while (WM_QUIT != msg.message)
    {
        hEvents[0] = m_hNextSkeletonEvent;

        // Check to see if we have either a message (by passing in QS_ALLEVENTS)
        // Or a Kinect event (hEvents)
        // Update() will check for Kinect events individually, in case more
        //than one are signalled
        DWORD dwEvent = MsgWaitForMultipleObjects(eventCount, hEvents, FALSE,
                                                  INFINITE, QS_ALLINPUT);

        // Check if this is an event we're waiting on and not a timeout or
        // message
        if (WAIT_OBJECT_0 == dwEvent)
        {
            Update();
        }

        if (PeekMessageW(&msg, NULL, 0, 0, PM_REMOVE))
        {
            // If a dialog message will be taken care of by the dialog proc
            if ((hWndApp != NULL) && IsDialogMessageW(hWndApp, &msg))
            {
                continue;
            }

            TranslateMessage(&msg);
            DispatchMessageW(&msg);
        }
```

```
    }

    return static_cast<int>(msg.wParam);
}

/// <summary>
/// Main processing function
/// </summary>
void CSkeletonBasics::Update()
{
    if (NULL == m_pNuiSensor)
    {
        return;
    }

    // Wait for 0ms, just quickly test if it is time to process a skeleton
    if ( WAIT_OBJECT_0 == WaitForSingleObject(m_hNextSkeletonEvent, 0) )
    {
        ProcessSkeleton();
    }
}

/// <summary>
/// Handles window messages, passes most to the class instance to handle
/// </summary>
/// <param name="hWnd">window message is for</param>
/// <param name="uMsg">message</param>
/// <param name="wParam">message data</param>
/// <param name="lParam">additional message data</param>
/// <returns>result of message processing</returns>
LRESULT CALLBACK CSkeletonBasics::MessageRouter(HWND hWnd, UINT uMsg, WPARAM
wParam, LPARAM lParam)
{
    CSkeletonBasics* pThis = NULL;

    if (WM_INITDIALOG == uMsg)
    {
        pThis = reinterpret_cast<CSkeletonBasics*>(lParam);
        SetWindowLongPtr(hWnd, GWLP_USERDATA,
                         reinterpret_cast<LONG_PTR>(pThis));
    }
    else
    {
        pThis = reinterpret_cast<CSkeletonBasics*>(::GetWindowLongPtr(hWnd,
                                                   GWLP_USERDATA));
    }

    if (pThis)
    {
        return pThis->DlgProc(hWnd, uMsg, wParam, lParam);
    }

    return 0;
}

/// <summary>
/// Handle windows messages for the class instance
/// </summary>
/// <param name="hWnd">window message is for</param>
/// <param name="uMsg">message</param>
/// <param name="wParam">message data</param>
```

```
/// <param name="lParam">additional message data</param>
/// <returns>result of message processing</returns>
LRESULT CALLBACK CSkeletonBasics::DlgProc(HWND hWnd, UINT message, WPARAM
wParam, LPARAM lParam)
{
    switch (message)
    {
    case WM_INITDIALOG:
        {
            // Bind application window handle
            m_hWnd = hWnd;

            // Init Direct2D
            D2D1CreateFactory(D2D1_FACTORY_TYPE_SINGLE_THREADED,
                              &m_pD2DFactory);

            // Look for a connected Kinect, and create it if found
            CreateFirstConnected();
        }
        break;

    case WM_CLOSE:
    // If the titlebar X is clicked, destroy app
        DestroyWindow(hWnd);
        break;

    case WM_DESTROY:
    // Quit the main message pump
        PostQuitMessage(0);
        break;


    case WM_COMMAND:
    // Handle button press
    // If for the near mode control and a clicked event, change near mode
        if (IDC_CHECK_SEATED == LOWORD(wParam) && BN_CLICKED == HIWORD(wParam))
        {
            // Toggle out internal state for near mode
            m_bSeatedMode = !m_bSeatedMode;

            if (NULL != m_pNuiSensor)
            {
                // Set near mode for sensor based on our internal state
                m_pNuiSensor->NuiSkeletonTrackingEnable(m_hNextSkeletonEvent,
                          m_bSeatedMode ?
                          NUI_SKELETON_TRACKING_FLAG_ENABLE_SEATED_SUPPORT : 0);
            }
        }
        break;
    }

    return FALSE;
}

/// <summary>
/// Create the first connected Kinect found
/// </summary>
/// <returns>indicates success or failure</returns>
HRESULT CSkeletonBasics::CreateFirstConnected()
{
    INuiSensor * pNuiSensor;
```

```
    int iSensorCount = 0;
    HRESULT hr = NuiGetSensorCount(&iSensorCount);
    if (FAILED(hr))
    {
        return hr;
    }

    // Look at each Kinect sensor
    for (int i = 0; i < iSensorCount; ++i)
    {
        // Create the sensor so we can check status, if we can't create it,
move on to the next
        hr = NuiCreateSensorByIndex(i, &pNuiSensor);
        if (FAILED(hr))
        {
            continue;
        }

        // Get the status of the sensor, and if connected, then we can
initialize it
        hr = pNuiSensor->NuiStatus();
        if (S_OK == hr)
        {
            m_pNuiSensor = pNuiSensor;
            break;
        }

        // This sensor wasn't OK, so release it since we're not using it
        pNuiSensor->Release();
    }

    if (NULL != m_pNuiSensor)
    {
        // Initialize the Kinect and specify that we'll be using skeleton
        hr = m_pNuiSensor->NuiInitialize(NUI_INITIALIZE_FLAG_USES_SKELETON);
        if (SUCCEEDED(hr))
        {
            // Create an event that will be signaled when skeleton data is
            // available
            m_hNextSkeletonEvent = CreateEventW(NULL, TRUE, FALSE, NULL);

            // Open a skeleton stream to receive skeleton data
            hr = m_pNuiSensor->NuiSkeletonTrackingEnable(m_hNextSkeletonEvent,
                                                         0);
        }
    }

    if (NULL == m_pNuiSensor || FAILED(hr))
    {
        SetStatusMessage(L"No ready Kinect found!");
        return E_FAIL;
    }

    return hr;
}

/// <summary>
/// Handle new skeleton data
/// </summary>
void CSkeletonBasics::ProcessSkeleton()
```

```
{
    NUI_SKELETON_FRAME skeletonFrame = {0};

    HRESULT hr = m_pNuiSensor->NuiSkeletonGetNextFrame(0, &skeletonFrame);
    if ( FAILED(hr) )
    {
        return;
    }

    // smooth out the skeleton data
    m_pNuiSensor->NuiTransformSmooth(&skeletonFrame, NULL);

    // Endure Direct2D is ready to draw
    hr = EnsureDirect2DResources( );
    if ( FAILED(hr) )
    {
        return;
    }

    m_pRenderTarget->BeginDraw();
    m_pRenderTarget->Clear( );

    RECT rct;
    GetClientRect( GetDlgItem( m_hWnd, IDC_VIDEOVIEW ), &rct);
    int width = rct.right;
    int height = rct.bottom;

    for (int i = 0 ; i < NUI_SKELETON_COUNT; ++i)
    {
        NUI_SKELETON_TRACKING_STATE trackingState = skeletonFrame.
                                                    SkeletonData[i].eTrackingState;

        if (NUI_SKELETON_TRACKED == trackingState)
        {
            // We're tracking the skeleton, draw it
            DrawSkeleton(skeletonFrame.SkeletonData[i], width, height);
        }
        else if (NUI_SKELETON_POSITION_ONLY == trackingState)
        {
            // we've only received the center point of the skeleton, draw that
            D2D1_ELLIPSE ellipse = D2D1::Ellipse(
                SkeletonToScreen(skeletonFrame.SkeletonData[i].Position, width,
                                 height),
                g_JointThickness,
                g_JointThickness
                );

            m_pRenderTarget->DrawEllipse(ellipse, m_pBrushJointTracked);
        }
    }

       display();

    hr = m_pRenderTarget->EndDraw();

    // Device lost, need to recreate the render target
    // We'll dispose it now and retry drawing
    if (D2DERR_RECREATE_TARGET == hr)
    {
        hr = S_OK;
        DiscardDirect2DResources();
```

```
    }
}

/// <summary>
/// Draws a skeleton
/// </summary>
/// <param name="skel">skeleton to draw</param>
/// <param name="windowWidth">width (in pixels) of output buffer</param>
/// <param name="windowHeight">height (in pixels) of output buffer</param>
void CSkeletonBasics::DrawSkeleton(const NUI_SKELETON_DATA & skel, int
windowWidth, int windowHeight)
{
    int i;
       Vector4 leftHandVector = SkeletonToScreenVector(skel.SkeletonPositions[
                                NUI_SKELETON_POSITION_HAND_LEFT], 10, 10);
       Vector4 rightHandVector = SkeletonToScreenVector(skel.SkeletonPositions[
                                 NUI_SKELETON_POSITION_HAND_RIGHT], 10, 10);

       Vector4 graphOrigin = {surfacePosX, surfacePosY, surfaceDistance, 1};
       gesture.update(leftHandVector, rightHandVector, graphOrigin);

       surfaceDistance = graphOrigin.z;
       surfacePosX = graphOrigin.x;
       surfacePosY = graphOrigin.y;

       for (i = 0; i < NUI_SKELETON_POSITION_COUNT; ++i)
       {
           m_Points[i] = SkeletonToScreen(skel.SkeletonPositions[i],
                                          windowWidth, windowHeight);
       }

    // Render Torso
    DrawBone(skel, NUI_SKELETON_POSITION_HEAD,
                                       NUI_SKELETON_POSITION_SHOULDER_CENTER);
    DrawBone(skel, NUI_SKELETON_POSITION_SHOULDER_CENTER,
                                       NUI_SKELETON_POSITION_SHOULDER_LEFT);
    DrawBone(skel, NUI_SKELETON_POSITION_SHOULDER_CENTER,
                                       NUI_SKELETON_POSITION_SHOULDER_RIGHT);
    DrawBone(skel, NUI_SKELETON_POSITION_SHOULDER_CENTER,
                                       NUI_SKELETON_POSITION_SPINE);
    DrawBone(skel, NUI_SKELETON_POSITION_SPINE,
                                       NUI_SKELETON_POSITION_HIP_CENTER);
    DrawBone(skel, NUI_SKELETON_POSITION_HIP_CENTER,
                                       NUI_SKELETON_POSITION_HIP_LEFT);
    DrawBone(skel, NUI_SKELETON_POSITION_HIP_CENTER,
                                       NUI_SKELETON_POSITION_HIP_RIGHT);

    // Left Arm
    DrawBone(skel, NUI_SKELETON_POSITION_SHOULDER_LEFT,
                                       NUI_SKELETON_POSITION_ELBOW_LEFT);
    DrawBone(skel, NUI_SKELETON_POSITION_ELBOW_LEFT,
                                       NUI_SKELETON_POSITION_WRIST_LEFT);
    DrawBone(skel, NUI_SKELETON_POSITION_WRIST_LEFT,
                                       NUI_SKELETON_POSITION_HAND_LEFT);

    // Right Arm
    DrawBone(skel, NUI_SKELETON_POSITION_SHOULDER_RIGHT,
                                       NUI_SKELETON_POSITION_ELBOW_RIGHT);
    DrawBone(skel, NUI_SKELETON_POSITION_ELBOW_RIGHT,
                                       NUI_SKELETON_POSITION_WRIST_RIGHT);
    DrawBone(skel, NUI_SKELETON_POSITION_WRIST_RIGHT,
```

```
                                        NUI_SKELETON_POSITION_HAND_RIGHT);

    // Left Leg
    DrawBone(skel, NUI_SKELETON_POSITION_HIP_LEFT,
                                        NUI_SKELETON_POSITION_KNEE_LEFT);
    DrawBone(skel, NUI_SKELETON_POSITION_KNEE_LEFT,
                                        NUI_SKELETON_POSITION_ANKLE_LEFT);
    DrawBone(skel, NUI_SKELETON_POSITION_ANKLE_LEFT,
                                        NUI_SKELETON_POSITION_FOOT_LEFT);

    // Right Leg
    DrawBone(skel, NUI_SKELETON_POSITION_HIP_RIGHT,
                                        NUI_SKELETON_POSITION_KNEE_RIGHT);
    DrawBone(skel, NUI_SKELETON_POSITION_KNEE_RIGHT,
                                        NUI_SKELETON_POSITION_ANKLE_RIGHT);
    DrawBone(skel, NUI_SKELETON_POSITION_ANKLE_RIGHT,
                                        NUI_SKELETON_POSITION_FOOT_RIGHT);

    // Draw the joints in a different color
    for (i = 0; i < NUI_SKELETON_POSITION_COUNT; ++i)
    {
        D2D1_ELLIPSE ellipse = D2D1::Ellipse( m_Points[i], g_JointThickness,
                                              g_JointThickness );
        if(!gesture.getState())
            m_pRenderTarget->DrawEllipse(ellipse, m_pBrushJointInferred);
        else
            m_pRenderTarget->DrawEllipse(ellipse, m_pBrushJointTracked);
    }
}

/// <summary>
/// Draws a bone line between two joints
/// </summary>
/// <param name="skel">skeleton to draw bones from</param>
/// <param name="joint0">joint to start drawing from</param>
/// <param name="joint1">joint to end drawing at</param>
void CSkeletonBasics::DrawBone(const NUI_SKELETON_DATA & skel,
NUI_SKELETON_POSITION_INDEX joint0, NUI_SKELETON_POSITION_INDEX joint1)
{
    NUI_SKELETON_POSITION_TRACKING_STATE joint0State = skel.
                                        eSkeletonPositionTrackingState[joint0];
    NUI_SKELETON_POSITION_TRACKING_STATE joint1State = skel.
                                        eSkeletonPositionTrackingState[joint1];

    // If we can't find either of these joints, exit
    if (joint0State == NUI_SKELETON_POSITION_NOT_TRACKED ||
        joint1State == NUI_SKELETON_POSITION_NOT_TRACKED)
    {
        return;
    }

    // Don't draw if both points are inferred
    if (joint0State == NUI_SKELETON_POSITION_INFERRED &&
        joint1State == NUI_SKELETON_POSITION_INFERRED)
    {
        return;
    }

    // We assume all drawn bones are inferred unless BOTH joints are tracked
    if (joint0State == NUI_SKELETON_POSITION_TRACKED &&
        joint1State == NUI_SKELETON_POSITION_TRACKED)
```

```
    {
        m_pRenderTarget->DrawLine(m_Points[joint0], m_Points[joint1],
m_pBrushBoneTracked, g_TrackedBoneThickness);
    }
    else
    {
        m_pRenderTarget->DrawLine(m_Points[joint0], m_Points[joint1],
m_pBrushBoneInferred, g_InferredBoneThickness);
    }
}

/// <summary>
/// Converts a skeleton point to screen space
/// </summary>
/// <param name="skeletonPoint">skeleton point to tranform</param>
/// <param name="width">width (in pixels) of output buffer</param>
/// <param name="height">height (in pixels) of output buffer</param>
/// <returns>point in screen-space</returns>
D2D1_POINT_2F CSkeletonBasics::SkeletonToScreen(Vector4 skeletonPoint, int
width, int height)
{
    LONG x, y;
    USHORT depth;

    // Calculate the skeleton's position on the screen
    // NuiTransformSkeletonToDepthImage returns coordinates in
    // NUI_IMAGE_RESOLUTION_320x240 space
    NuiTransformSkeletonToDepthImage(skeletonPoint, &x, &y, &depth);

    float screenPointX = static_cast<float>(x * width) / cScreenWidth;
    float screenPointY = static_cast<float>(y * height) / cScreenHeight;

    return D2D1::Point2F(screenPointX, screenPointY);
}

/// <summary>
/// Ensure necessary Direct2d resources are created
/// </summary>
/// <returns>S_OK if successful, otherwise an error code</returns>
HRESULT CSkeletonBasics::EnsureDirect2DResources()
{
    HRESULT hr = S_OK;

    // If there isn't currently a render target, we need to create one
    if (NULL == m_pRenderTarget)
    {
        RECT rc;
        GetWindowRect( GetDlgItem( m_hWnd, IDC_VIDEOVIEW ), &rc );

        int width = rc.right - rc.left;
        int height = rc.bottom - rc.top;
        D2D1_SIZE_U size = D2D1::SizeU( width, height );
        D2D1_RENDER_TARGET_PROPERTIES rtProps = D2D1::RenderTargetProperties();
        rtProps.pixelFormat = D2D1::PixelFormat( DXGI_FORMAT_B8G8R8A8_UNORM,
                                                 D2D1_ALPHA_MODE_IGNORE);
        rtProps.usage = D2D1_RENDER_TARGET_USAGE_GDI_COMPATIBLE;

        // Create a Hwnd render target, in order to render to the window set
        // in initialize
        hr = m_pD2DFactory->CreateHwndRenderTarget(rtProps,
                           D2D1::HwndRenderTargetProperties(
```

```
                                GetDlgItem( m_hWnd, IDC_VIDEOVIEW), size),
                                &m_pRenderTarget);

        if ( FAILED(hr) )
        {
            SetStatusMessage(L"Couldn't create Direct2D render target!");
            return hr;
        }

        //light green
        m_pRenderTarget->CreateSolidColorBrush(D2D1::ColorF(0.27f, 0.75f,
                        0.27f), &m_pBrushJointTracked);

        m_pRenderTarget->CreateSolidColorBrush(
                        D2D1::ColorF(D2D1::ColorF::Yellow, 1.0f),
                        &m_pBrushJointInferred);
        m_pRenderTarget->CreateSolidColorBrush(
                        D2D1::ColorF(D2D1::ColorF::Green, 1.0f),
                        &m_pBrushBoneTracked);
        m_pRenderTarget->CreateSolidColorBrush(
                        D2D1::ColorF(D2D1::ColorF::Gray, 1.0f),
                        &m_pBrushBoneInferred);
    }

    return hr;
}

/// <summary>
/// Dispose Direct2d resources
/// </summary>
void CSkeletonBasics::DiscardDirect2DResources( )
{
    SafeRelease(m_pRenderTarget);
    SafeRelease(m_pBrushJointTracked);
    SafeRelease(m_pBrushJointInferred);
    SafeRelease(m_pBrushBoneTracked);
    SafeRelease(m_pBrushBoneInferred);
}

/// <summary>
/// Set the status bar message
/// </summary>
/// <param name="szMessage">message to display</param>
void CSkeletonBasics::SetStatusMessage(WCHAR * szMessage)
{
    SendDlgItemMessageW(m_hWnd, IDC_STATUS, WM_SETTEXT, 0, (LPARAM)szMessage);
}

//------------------------------------------------------------------------------
// OpenGL.h
//------------------------------------------------------------------------------
#ifndef OPEN_GL_H
#define OPEN_GL_H

#include <iostream>
#include <GL/freeglut.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <vector>
#include <math.h>
#include "MathFunctions.h"
```

```
const float Z_FAR = 1000;
const float Z_NEAR = 1;
int initOpenGL();
void display();

#endif

//-------------------------------------------------------------------------
// OpenGL.cpp
//-------------------------------------------------------------------------
#include "OpenGL.h"

int screen_width=800;
int screen_height=600;

extern float surfaceDistance = 0.0f;
static float surfaceDistance_increment = 0.0f;
extern float surfacePosX = 0.0f;
static float surfacePosX_increment = 0.0f;
extern float surfacePosY = 0.0f;
static float surfacePosY_increment = 0.0f;

int filling = 1;

Surface surface;
Axis axis;

void init(void)
{
        glClearColor(0.0, 0.0, 0.0, 0.0);
        glShadeModel(GL_SMOOTH);
        glViewport(0,0,screen_width,screen_height);
        glEnable(GL_DEPTH_TEST);
        glPolygonMode (GL_FRONT_AND_BACK, GL_FILL);
}

void resize (int width, int height)
{
        screen_width=width;
        screen_height=height;

        glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glViewport(0,0,screen_width,screen_height); // Viewport transformation

        glMatrixMode(GL_PROJECTION); // Projection transformation
        glLoadIdentity();
        gluPerspective(45.0f, (GLfloat)screen_width/(GLfloat)screen_height,
                        1.0f, 1000.0f);

        glutPostRedisplay ();
}

void keyboard (unsigned char key, int x, int y)
{

        switch (key)
        {
        case 27:
             exit(0);
             break;
```

```
        case 'r': case 'R':
            if (filling==0)
            {
                glPolygonMode (GL_FRONT_AND_BACK, GL_FILL);
                filling=1;
            }
            else
            {
                glPolygonMode (GL_FRONT_AND_BACK, GL_LINE);
                filling=0;
            }
            break;

        //Surface control key
        case 'u': case 'U':
            surfaceDistance_increment = surfaceDistance_increment + 0.05f;
            break;
        case 'o': case 'O':
            surfaceDistance_increment = surfaceDistance_increment - 0.05f;
            break;
        case 'i': case 'I':
            surfacePosY_increment = surfacePosY_increment + 0.05f;
            break;
        case 'k': case 'K':
            surfacePosY_increment = surfacePosY_increment - 0.05f;
            break;
        case 'j': case 'J':
            surfacePosX_increment = surfacePosX_increment - 0.05f;
            break;
        case 'l': case 'L':
            surfacePosX_increment = surfacePosX_increment + 0.05f;
            break;
        }
}

void display()
{
        float const screen_aspect = (float)screen_width / (float)screen_height;

        glViewport(0, 0, screen_width, screen_height);

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(70, screen_aspect, Z_NEAR, Z_FAR);

        glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
        axis.draw(surfacePosX, surfacePosY, surfaceDistance);

        surface.draw(surfacePosX, surfacePosY, surfaceDistance);

        glutSwapBuffers();
}

int initOpenGL()
{
        int argc = 0;
        char** argv = NULL;
```

```
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
        glutInitWindowSize(screen_width,screen_height);
        glutInitWindowPosition(0,0);
        glutCreateWindow("Kinect Graph");
        glutDisplayFunc(display);
        glutIdleFunc(display);
        glutReshapeFunc(resize);
        glutKeyboardFunc(keyboard);
        init();

        return 0;
}

//------------------------------------------------------------------------------
// MathFunctions.h
//------------------------------------------------------------------------------
#ifndef MATH_FUNCTION_H
#define MATH_FUNCTION_H

#include <iostream>
#include <GL/freeglut.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <vector>
#include <math.h>

const float M_PI_2 = 1.57079632679489661923132169163975144;
const float M_PI = 3.14159265358979323846264338327950288;
const float Z_OFFSET = -5;


class Surface
{
protected:
        std::vector<GLfloat> vertices;
        std::vector<GLushort> indices;
        float xMin, xMax;
        float yMin, yMax;
        float zMin, zMax;
        int xCount, yCount;
        float scale;
        float increment;

        float f(float x, float y);
        float max2(float x, float y);
        float max3(float x, float y, float z);
public:
        Surface();
        void draw(GLfloat x, GLfloat y, GLfloat z);
};

class Axis
{
protected:
        float rangeX, rangeY, rangeZ;
public:
        Axis(float rx = 10000, float ry = 10000, float rz = 10000);
        void draw(GLfloat x, GLfloat y, GLfloat z);
};
```

```
#endif

//-------------------------------------------------------------------------
// MathFunctions.cpp
//-------------------------------------------------------------------------
#include "MathFunctions.h"

float Surface::f(float x, float y)
{
	return cos(x)+sin(y);
}

float Surface::max2(float x, float y) {
	if(x > y ) return x;
	return x;
}

float Surface::max3(float x, float y, float z) {
	if(x > y && x > z) return x;
	if(y > x && y > z) return x;
	return z;
}

Surface::Surface()
{
	xMin = -5;
	xMax = 5;
	yMin = -5;
	yMax = 5;
	zMin = 100000;
	zMax = -100000;
	increment = 0.1;
	xCount = (int)((xMax - xMin)/increment);
	yCount = (int)((yMax - yMin)/increment);

	vertices.resize((xCount+1) * (yCount+1) * 3);
	std::vector<GLfloat>::iterator v = vertices.begin();
	float x=xMin;
	float z;
	for(int i=0;  i<=xCount ; i++, x+=increment) {
	     float y=yMin;
	     for(int j=0; j<=yCount; j++, y+=increment) {
		z = f(x,y);
		//find max min
		if( z <= zMin ) zMin = z;
		if( z >= zMax ) zMax = z;
		//assign to vector
		*v++ = y;
		*v++ = z;
		*v++ = -x;
	     }
	}

	indices.resize(xCount * yCount * 4);
	std::vector<GLushort>::iterator it = indices.begin();
	for(int i=0; i<xCount ; i++) {
	     for(int j=0; j<yCount; j++) {
		*it++ = i*(yCount+1) + j;
		*it++ = i*(yCount+1) + j + 1;
		*it++ = (i+1)*(yCount+1) + j + 1;
		*it++ = (i+1)*(yCount+1) + j;
```

```
            }
        }
}

void Surface::draw(GLfloat x, GLfloat y, GLfloat z)
{
        glMatrixMode(GL_MODELVIEW);
        glPushMatrix();

        glTranslatef(x,y,z+Z_OFFSET);

        scale=1/max3( max2(abs(xMax),abs(xMin)), max2(abs(yMax),abs(yMin)),
                      max2(abs(zMax),abs(zMin)) );
        glScalef(scale, scale, scale);

        glEnableClientState(GL_VERTEX_ARRAY);

        glVertexPointer(3, GL_FLOAT, 0, &vertices[0]);
        glDrawElements(GL_QUADS,indices.size(),GL_UNSIGNED_SHORT,&indices[0]);
        glPopMatrix();
}

Axis::Axis(float rx, float ry, float rz):
rangeX(rx),rangeY(ry),rangeZ(rz)
{

}

void Axis::draw(GLfloat x, GLfloat y, GLfloat z)
{
    z = z + Z_OFFSET;

    //x-axis
    glBegin(GL_LINES);
    glVertex3f(x - rangeX, y, z);
    glVertex3f(x + rangeX, y, z);
    glEnd();

    //y-axis
    glBegin(GL_LINES);
    glVertex3f(x, y - rangeY, z);
    glVertex3f(x, y + rangeY, z);
    glEnd();

    //z-axis
    glBegin(GL_LINES);
    glVertex3f(x, y, z - rangeZ);
    glVertex3f(x, y, z + rangeZ);
    glEnd();
}

//------------------------------------------------------------------------
// Gesture.h
//------------------------------------------------------------------------
#ifndef GESTURE_H
#define GESTURE_H

#include "stdafx.h"
#include "NuiApi.h"
#include <cmath>
#include <iostream>
```

```
const int STATE_OFF = 0;
const int STATE_ON  = 1;

const float MIN_HAND_DISTANCE = 0.2;
const float MAX_HAND_DISTANCE = 0.4;
const float HAND_Y_THRESHOLD  = 0.03;
const float HAND_Z_THRESHOLD  = 0.03;

const float SCALE_X = 2;
const float SCALE_Y = 2;
const float SCALE_Z = 4;

class Gesture
{
protected:
        int state;
        Vector4 prevPos;
public:
        Gesture();
        void update(const Vector4& leftHand, const Vector4& rightHand,
                    Vector4& origin);
        bool getState() { return state; }
};

#endif

//---------------------------------------------------------------------------
// Gesture.cpp
//---------------------------------------------------------------------------
#include "Gesture.h"

Gesture::Gesture()
{
        state = STATE_OFF;
        prevPos.x = 0;
        prevPos.y = 0;
        prevPos.z = 0;
}

float distance(const Vector4& leftHand, const Vector4& rightHand)
{
        float difX = abs(leftHand.x - rightHand.x);
        float difY = abs(leftHand.y - rightHand.y);
        float difZ = abs(leftHand.z - rightHand.z);
        return sqrt(difX*difX + difY*difY + difZ*difZ);
}

void Gesture::update(const Vector4& leftHand, const Vector4& rightHand,
Vector4& origin)
{
        //check state
        if( state == STATE_OFF ) {
             //check position of left hand & right hand
             float difX = abs(leftHand.x - rightHand.x);
             float difY = abs(leftHand.y - rightHand.y);
             float difZ = abs(leftHand.z - rightHand.z);
             bool isCorrectGesture = ( difX >= MIN_HAND_DISTANCE &&
                                       difX <= MAX_HAND_DISTANCE &&
                                       difY <= HAND_Y_THRESHOLD &&
                                       difZ <= HAND_Z_THRESHOLD);
```

```
            if( isCorrectGesture ) {
                  state = STATE_ON;
                  prevPos.x = leftHand.x;
                  prevPos.y = leftHand.y;
                  prevPos.z = leftHand.z;
            }
      }
      else { //state == STATE_ON
            if(distance(leftHand, rightHand) <= MAX_HAND_DISTANCE) {
                  state = STATE_ON;

                  //find difference between current and previous hand position
                  float changeX = leftHand.x - prevPos.x;
                  float changeY = leftHand.y - prevPos.y;
                  float changeZ = leftHand.z - prevPos.z;

                  //update origin
                  origin.x += SCALE_X * changeX;
                  origin.y += SCALE_Y * changeY;
                  origin.z += SCALE_Z * changeZ;

                  //save previous hand position
                  prevPos.x = leftHand.x;
                  prevPos.y = leftHand.y;
                  prevPos.z = leftHand.z;
            }
            else {
                  state = STATE_OFF;
            }
      }
}
```

ข้อมูลประวัติคณะผู้วิจัย

ประวัติส่วนตัว

ชื่อ-สกุล ดร. นัทธพงศ์ จึงธีรพานิช

เพศ ■ ชาย □หญิง วันเดือนปีเกิด 23 กันยายน 2523 อายุ 32 ปี

สถานภาพ ■ โสด □ สมรส

ตำแหน่งปัจจุบัน อาจารย์ สาขาวิชาวิศวกรรมและเทคโนโลยี วิทยาลัยนานาชาติ

ประวัติการศึกษา

ชื่อย่อปริญญา	สาขา	สถาบันที่จบ	ปีที่จบ
Ph.D.	Informatics	School of Informatics University of Edinburgh	2553
M.Sc.	Advance Computing	Department of Computing Imperial College London (UK)	2546
วศ.บ. (เกียรตินิยมอันดับหนึ่ง)	วิศวกรรมคอมพิวเตอร์	คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง	2544

สาขาวิจัยที่มีความชำนาญพิเศษ: โมดอลมิวแคลคิวลัส (Modal μ-Calculus), ตรรกศาสตร์สำหรับวิทยาศาสตร์คอมพิวเตอร์ (Logic for Computer Science), การตรวจสอบความถูกต้องของซอฟต์แวร์ (Software Verification)

ทุนการศึกษาและทุนวิจัยที่เคยได้รับ

ปี พ.ศ.	ทุนการศึกษาและทุนวิจัย	สถาบันที่ให้
2547 - 2551	ทุน Overseas Research Studentship (ORS) Awards	รัฐบาลประเทศอังกฤษ

ผลงานวิจัย/งานสร้างสรรค์

การประชุมวิชาการระดับนานาชาติ

1. N. Jungteerapanich, "A Tableau System for the Modal μ-Calculus," International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (Tableaux '09), Oslo, Norway, 2009.

การประชุมวิชาการระดับภูมิภาค ระดับประเทศ และงานสัมมนาวิชาการ

1. N. Jungteerapanich, "Tableau Method for the Satisfiability Problem of the Modal Mu-Calculus", In Proc. of the 3rd Regional Conference on ICT Applications for Industries and Small Companies in ASEAN countries (RCICT 2011), 2011.
2. N. Jungteerapanich, "A Tableau System and the Small Model Theorem for the Π_2 - Fragment of the Modal μ-Calculus," International Workshop on Tableaux vs Automata as Logical Decision Methods (AutoTab '09), Oslo, Norway, 2009.

ประวัติส่วนตัว

ชื่อ-สกุล ดร. อุกฤษฎ์ วัชรีฤทัย

เพศ ■ ชาย ☐หญิง วันเดือนปีเกิด 28 ธันวาคม 2522 อายุ 33 ปี

สถานภาพ ■ โสด ☐ สมรส

ตำแหน่งปัจจุบัน อาจารย์ สาขาวิชาวิศวกรรมและเทคโนโลยี วิทยาลัยนานาชาติ

ประวัติการศึกษา

ชื่อย่อปริญญา	สาขา	สถาบันที่จบ	ปีที่จบ
D. Eng	Media Science	Graduate School of Information Science, Nagoya University	2553
MS. of Information Science	Media Science	Graduate School of Information Science, Nagoya University	2550
วศ.บ. (เกียรตินิยมอันดับหนึ่ง)	วิศวกรรมไฟฟ้า	คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์	2545

สาขาวิจัยที่มีความชำนาญพิเศษ: การประมวลผลและรู้จำภาพ (Image Processing and Pattern Recognition), การโปรแกรมเชิงพันธุกรรม (Genetic Programming), ทัศนศาสตร์คอมพิวเตอร์เชิงวิวัฒนาการ (Evolutionary Computer Vision), ไบโอเมตริกส์ (Biometrics)

รางวัลด้านวิชาการ/ด้านวิจัย/งานสร้างสรรค์ (ด้านศิลปะ หรืออื่นๆ) ที่ได้รับ

ปี พ.ศ.	ชื่อรางวัล	สถาบันที่ให้
2554	IPSJ Digital Courier Funai Young Researcher Encouragement Award	Funai Foundation for Information Technology (FFIT)
2551	Presentation award (The 72nd Technical Meeting of IPSJ SIG-MPS (IPSJ MPS-72))	The Information Processing Society of Japan (IPSJ)

2549	Finalist for The Best Paper Award (2006 IEEE International Conference on Cybernetics and Intelligent Systems)	Institute of Electrical and Electronics Engineers (IEEE)

ทุนการศึกษาและทุนวิจัยที่เคยได้รับ

ปี พ.ศ.	ทุนการศึกษาและทุนวิจัย	สถาบันที่ให้
2553	ทุนสร้างสรรค์นวัตกรรมภาครัฐ	สำนักงานคณะกรรมการข้าราชการและพลเรือน (ก.พ.)
2551	ทุนสนับสนุนการวิจัยในหัวข้อ Automatic Extraction of Image Features Based on Evolutionary Computation	The Hori Information Science Promotion Foundation
2547	ทุนรัฐบาลญี่ปุ่น	The Ministry of Education, Culture, Sports, Science and Technology of Japan (MEXT)
2541	ทุนการศึกษา (สำหรับผู้ที่ได้คะแนนสอบเอ็นทรานซ์ลำดับที่ 1-6 ของคณะ)	คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์

ผลงานวิจัย/งานสร้างสรรค์

ผลงานวิจัย/งานสร้างสรรค์ที่ตีพิมพ์เผยแพร่ (ระดับชาติและนานาชาติ)

1. Ukrit Watchareeruetai, Akisato Kimura, Robert Cheng Bao, Takahito Kawanishi, and Kunio Kashino, "Interest point detection based on stochastically derived stability," IPSJ Transactions on Computer Vision and Applications, vol.3, pp.186-197, December 2011.
2. Ukrit Watchareeruetai, Yoshinori Takeuchi, Tetsuya Matsumoto, Hiroaki Kudo, and Noboru Ohnishi, "Redundancies in linear GP, canonical transformation, and its exploitation: a demonstration on image feature synthesis," Genetic Programming and Evolvable Machines, vol.12, no.1, pp.49-77, March 2011.
3. Ukrit Watchareeruetai and Noboru Ohnishi, "A new color-based lawn weed detection method and its integration with texture-based methods: a hybrid approach," IEEJ Transactions on Electronics, Information and Systems, vol.131, no.2, pp.355-366, February, 2011.
4. Ukrit Watchareeruetai, Tetsuya Matsumoto, Yoshinori Takeuchi, Hiroaki Kudo, and Noboru Ohnishi, "Multi-objective genetic programming with redundancy-regulations for automatic

construction of image feature extractors," IEICE Transactions on Information and Systems, vol.E93-D, no.9, pp.2614-2625, September 2010.

5. Ukrit Watchareeruetai, Yoshinori Takeuchi, Tetsuya Matsumoto, Hiroaki Kudo, and Noboru Ohnishi, "Evaluations of feature extraction programs synthesized by redundancy-removed linear genetic programming: a case study on lawn weed detection," Journal of Information Processing, vol.18, pp.164-174, April 2010. (Reprinted in: Information and Media Technologies, vol.5, no.2, pp.566-576, June 2010) ***(IPSJ Digital Courier Funai Young Researcher Encouragement Award)***
6. Ukrit Watchareeruetai, Tetsuya Matsumoto, Noboru Ohnishi, Hiroaki Kudo, and Yoshinori Takeuchi, "Acceleration of genetic programming by hierarchical structure learning: a case study on image recognition program synthesis," IEICE Transactions on Information and Systems, vol.E92-D, no.10, pp.2094-2102, October 2009.
7. Ukrit Watchareeruetai, Yoshinori Takeuchi, Tetsuya Matsumoto, Hiroaki Kudo, and Noboru Ohnishi, "Efficient construction of image feature extraction programs by using linear genetic programming with fitness retrieval and intermediate-result caching," in Foundation of Computational Intelligence Volume 4: Bio-inspired Data Mining, A. Abraham et al. (eds.), Springer-Verlag, pp.355-375, 2009. (ISBN 978-3-642-01087-3)
8. Ukrit Watchareeruetai, Yoshinori Takeuchi, Tetsuya Matsumoto, Hiroaki Kudo, and Noboru Ohnishi, "Computer vision based methods for detecting weeds in lawns," Machine Vision and Applications, vol.17, no.5, pp. 287-296, October 2006.
9. อุกฤษฎ์ วัชรีฤทัย, "การสังเคราะห์เชิงวิวัฒนาการของโปรแกรมประมวลผลภาพเพื่อตรวจแยกพื้นม่านตา," NECTEC Technical Journal, ฉบับที่ 10 (22), หน้า 173-179, 2010.

การเสนอผลงานวิชาการ

1. Ukrit Watchareeruetai, "Hierarchical structure genetic programming with generation adaptable learning nodes," Proceedings of the Forth AUN/SEED-Net Regional Conference on Information and Communication Technology, pp.222-228, Ho Chi Minh City, Vietnam, October 18-19, 2011.
2. Ukrit Watchareeruetai, "A research on automatic construction of image processing programs," Proceedings of the Third AUN/SEED-Net Regional Conference in Electrical and Electronics Engineering, pp.122-126, Manila, Philippines, September 8-9, 2010.
3. Ukrit Watchareeruetai, Akisato Kimura, Robert Cheng Bao, Takahito Kawanisi, and Kunio Kashino, "StochasticSIFT: Interest point detection based on stochastically-derived

stability," Proceedings of Meeting on Image Recognition and Understanding 2010 (MIRU2010), Kushiro, Japan, July 27-29, 2010.

4. Ukrit Watchareeruetai, Tetsuya Matsumoto, Yoshinori Takeuchi, Hiroaki Kudo, and Noboru Ohnishi, "Construction of image feature extractors based on multi-objective genetic programming with redundancy regulations," Proceedings of 2009 IEEE Conference on Systems, Man, and Cybernetics (SMC2009), pp.1365-1370, Texas, USA, October 11-14, 2009.
5. Ukrit Watchareeruetai, Tetsuya Matsumoto, Noboru Ohnishi, Hiroaki Kudo, and Yoshinori Takeuchi, "Acceleration of genetic programming by hierarchical structure learning: a study on image recognition program synthesis," Learning and Intelligent Optimization Conference (LION3), Trento, Italy, January 14-18, 2009.
6. Ukrit Watchareeruetai, Yoshinori Takeuchi, Tetsuya Matsumoto, Hiroaki Kudo, and Noboru Ohnishi, "Linear GP with redundancy-removed recombination for synthesis of image feature extraction programs," IPSJ SIG Technical Report (2008-MPS-72), vol.2008, no.126, pp.33-36, Osaka, Japan, December 17-18, 2008. (***Presentation Award***)
7. Ukrit Watchareeruetai, Yoshinori Takeuchi, Tetsuya Matsumoto, Hiroaki Kudo, and Noboru Ohnishi, "Improving search performance of linear genetic programming based image recognition program synthesis by redundancy-removed recombination," Proceedings of 2008 IEEE Conference on Soft Computing in Industrial Applications (SMCia/08), pp.393-398, Muroran, Japan, June 25-27, 2008.
8. Ukrit Watchareeruetai, Yoshinori Takeuchi, Tetsuya Matsumoto, Hiroaki Kudo, and Noboru Ohnishi, "Transformation of redundant representations of linear genetic programming into canonical forms for efficient extraction of image features," Proceedings of 2008 IEEE Congress on Evolutionary Computation (CEC-08), IEEE World Congress on Computational Intelligence, pp.1996-2003, Hong Kong, China, June 1-6, 2008.
9. Ukrit Watchareeruetai, Yoshinori Takeuchi, Tetsuya Matsumoto, Hiroaki Kudo, and Noboru Ohnishi, "Modified lawn weed detection: utilization of edge-color based SVM and grass-model based blob inspection filterbank," Proceedings of 14th International Conference on Neural Information Processing (ICONIP-07) Part II, LNCS 4985, Springer-Verlag Berlin Heidelberg, pp.30-39, Kitakyushu, Japan, November 13-16, 2007.
10. Ukrit Watchareeruetai, Yoshinori Takeuchi, Tetsuya Matsumoto, Hiroaki Kudo, and Noboru Ohnishi, "A lawn weed detection in winter season based on color information," Proceedings of

IAPR Conference on Machine Vision Applications (MVA 2007), pp.524-527, Tokyo, Japan, May 16-18, 2007.

11. Ukrit Watchareeruetai, Yoshinori Takeuchi, Tetsuya Matsumoto, Hiroaki Kudo, and Noboru Ohnishi, "Lawn weeds detection methods based on image processing techniques," IEICE Technical Report of PRMU 2006, vol.106, no.301, pp.65-70, Tokyo, Japan, October 19-20, 2006.
12. Ukrit Watchareeruetai, Yoshinori Takeuchi, Tetsuya Matsumoto, Hiroaki Kudo, and Noboru Ohnishi, "Computer vision based methods for detecting weeds in lawns," Proceedings of 2006 IEEE International Conference on Cybernetics and Intelligent Systems (CIS 2006), pp.323-328, Bangkok, Thailand, June 7-9, 2006 ***(Finalist for the Best Paper Award)***.
13. Vutipong Areekul,Teesid Leelasawassuk, Kittiwat Suppasriwasuseth, Suksan Jirachawang, and Ukrit Watchareeruetai, "Progress research on fingerprint verification algorithm in Thailand," Proceedings of 2006 Electrical Engineering/Electronics, Computer, Telecommunications, and Information Technology (ECTI) International Conference (ECTI-CON 2006), Ubon Ratchathani, Thailand, May 10-13, 2006.
14. Ukrit Watchareeruetai, Yoshinori Takeuchi, Tetsuya Matsumoto, Hiroaki Kudo, and Noboru Ohnishi, "Image processing based weed detection in lawn," 2005 Tokai-Section Joint Conference of the 8 Institutes of Electrical and Related Engineers, Nagoya, Japan, September 15-16, 2005.
15. Vutipong Areekul, Ukrit Watchareeruetai, Kittiwat Suppasriwasuseth, and Sawasd Tantaratana, "Separable Gabor filter realization for fast fingerprint enhancement," Proceedings of IEEE International Conference on Image Processing (ICIP 2005), vol. 3, pp.253-256, Genoa, Italy, September 11-14, 2005.
16. Vutipong Areekul, Ukrit Watchareeruetai, and Sawasd Tantaratana, "Fast separable Gabor filter for fingerprint enhancement," Proceedings of First International Conference on Biometrics Authentication (ICBA 2004), LNCS 3072, Springer-Verlag Berlin Heidelberg, pp.403-409, Hong Kong, China, July 17-19, 2004.
17. อุกฤษฏ์ วัชรีฤทัย, "วิธีการตรวจจับวัชพืชในสนามหญ้าฤดูหนาวโดยใช้กฎเกณฑ์ฟัซซี่," การประชุมทางวิชาการวิศวกรรมไฟฟ้าครั้งที่33 (EECON-33), เล่มที่ 2, หน้า 1237-1240, เชียงใหม่, ประเทศไทย, 1-3 ธันวาคม 2010
18. ธีรภัทร จำปีรัตน์ อุกฤษฏ์ วัชรีฤทัย กิตติวัฒน์ ศุภศรีวสุเศรษฐ์ และ วุฒิพงศ์ อารีกุล, "การเปรียบเทียบลายนิ้วมือโดยใช้คุณลักษณะต่างๆของมินูเทียร์ร่วมกับการอ้างอิงจุดโฟกัส," การประชุมทางวิชาการวิศวกรรมไฟฟ้าครั้งที่27 (EECON-27), เล่ม 2, หน้า 185-188, ขอนแก่น, ประเทศไทย, 11-12 พฤศจิกายน 2004

19. อุกฤษฎ์ วัชรีฤทัย และ วุฒิพงศ์ อารีกุล, "การตรวจสอบลายนิ้วมือโดยใช้การเปรียบเทียบการสุ่มเส้น ," การประชุมทางวิชาการวิศวกรรมไฟฟ้าครั้งที่25 (EECON-25), หน้า 25-29, สงขลา, ประเทศไทย, 21-22 พฤศจิกายน 2002