โปรแกรมสามมิติ สำหรับการจำลองการสวมใส่เสื้อผ้าชุดต่างๆ

VIRTUAL COSTUME STUDIO

นาย สุวพงษ์ ไชยมี

นาย อาร์ม ชัยอรุณดีกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

โปรแกรมสามมิติ สำหรับการจำลองการสวมใส่เสื้อผ้าชุดต่างๆ

VIRTUAL COSTUME STUDIO

โดย

นาย สุวพงษ์ ไชยมี

นาย อาร์ม ชัยอรุณดีกุล

อาจารย์ที่ปรึกษา

ดร. สมศักดิ์ วลัยรัชต์

ดร. อรัญญา วลัยรัชต์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ ปีการศึกษา 2547

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมสามมิติ สำหรับการจำลองการสวมใส่เสื้อผ้าชุดต่างๆ
VIRTUAL COSTUME STUDIO

คณะผู้จัดทำ นายสุวพงษ์ ไชยมี รหัส 44010363
นายอาร์ม ชัยอรุณดีกุล รหัส 44010379

..... อาจารย์ที่ปรึกษา
(ดร. สมศักดิ์ วลัยรัชต์)

..... อาจารย์ที่ปรึกษา
(ดร. อรัญญา วลัยรัชต์)

โปรแกรมสามมิติ สำหรับจำลองการสวมใส่เสื้อผ้าชุดต่างๆ

นาย สุวพงษ์ ไชยมี 44010562
นาย อาร์ม ชัยอรุณดีกุล 44010614
ดร. สมศักดิ์ วลัยรัชต์ อาจารย์ที่ปรึกษา
ดร. อรัญญา วลัยรัชต์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2547

บทคัดย่อ

โครงงานนี้เป็น การพัฒนาโปรแกรมจำลองการสวมเสื้อผ้าในชุดต่างๆ เพื่อเป็นทางเลือกใหม่ในการตัดสินใจ ซื้อเครื่องแต่งกาย โดยไม่ต้องลองของจริง มีข้อดีก็คือ สามารถช่วยตัดสินใจซื้อ ได้อย่างสะดวกและรวดเร็ว หรือบางกรณีที่มีเสื้อผ้าแบบใหม่ๆออกมาโดยยังไม่มีของจริงให้ลอง ซึ่งสามารถสั่งซื้อได้ แล้วไม่มั่นใจว่าถ้าใส่แล้วจะเข้ากับตัวผู้ซื้อ นี่เป็นทางเลือกที่น่าสนใจเลยทีเดียว และความสามารถของโปรแกรมที่สำคัญอีกประการหนึ่งก็คือ สามารถที่จะกำหนดสัดส่วนของร่างกาย ส่วนสูง น้ำหนัก สีผิว หรือแม้กระทั่งรอบอก รอบเอว ความยาวแขน ตามสัดส่วนของผู้ใช้ได้ เพื่อประโยชน์ของผู้ใช้ที่จะได้เห็นแบบจำลองที่ใกล้เคียงตัวจริงมากที่สุด แต่ข้อจำกัดของโปรแกรมนั้นยังไม่สามารถรู้ได้ว่า ลักษณะของเนื้อผ้าเป็นอย่างไร โปร่ง ใส่สบายหรือไม่ หรือว่ากระชับขนาดไหน อย่างไร โปรแกรมนี้พัฒนาขึ้นมานั้นใช้ภาษา C++ และ ไลบรารี่กราฟฟิค 3 มิติของ DirectX 9.0 และเครื่องมือที่ใช้สร้างแบบจำลอง 3 มิติทั้งตัวบุคคลและ เครื่องแต่งกายได้แก่โปรแกรม Poser และ 3D Max

VIRTUAL COSTUME STUDIO

Mr. Suvapong Chaimee 44010562
Mr. Arm Chaiarundeekul 44010614
Dr. Somsak Walairacht advisor
Dr. Aranya Walairacht advisor
Academic Year 2004

ABSTRACT

Virtual costume studio is a development of a clothes trying program to be a new way in determination of buying clothes by user must not to try on real clothes. The advantage is user can decide to buy quickly and convenient. In another case, there is the latest design come out and does not in stock and can be ordered then user may be doubt if it is suited himself. So it is the interested way. Another extra, this project is able to set the proportion of body, the height, coloring, even so the size of chest, hip, waistline and the length of each part of user also. For avail of users that could see the virtual model of themselves realistically. But this virtual model also has a disadvantage. For example, user can not know how the cloth's texture is. Is it comfortable or not. This program was developed in C++ with 3D Graphic of DirectX 9.0 library. Poser and 3D Max as tools to manage 3Ds Model in this program

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลายฝ่ายด้วยกัน โดยเฉพาะอาจารย์ที่ปรึกษาทั้งสองท่านที่ให้โอกาสข้าพเจ้าได้ทำปริญญานิพนธ์ฉบับนี้ คอยให้ความเอาใจใส่ แนะนำและความช่วยเหลือเสมอมา คือ ดร. สมศักดิ์ วลัยรัชต์ และ ดร. อรัญญา วลัยรัชต์ ซึ่งต้องขอขอบพระคุณเป็นอย่างสูง

ขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้จัดเตรียมความสะดวก เพื่อให้การวิจัยและพัฒนาโปรแกรมเป็นไปได้ด้วยความสะดวกและรวดเร็ว รวมทั้งยังมีอินเทอร์เน็ตความเร็วสูง ให้บริการสำหรับการค้นคว้าหาข้อมูลและความรู้ต่างๆ ซึ่งสุดท้ายแล้วก็ได้ประกอบกันเป็นส่วนหนึ่งของโครงงานนี้

ขอขอบคุณเพื่อนๆ น้องๆ ในปฏิบัติการมัลติมีเดียที่คอยสร้างความครื้นเครงยามอยู่ในห้อง เป็นกำลังใจเสมอมา และที่ขาดไม่ได้ต้องขอคุณห้องมัลติมีเดียที่ให้ที่พักอาศัย ในเวลามีงานที่เร่งรีบ และเป็นที่พักผ่อนในเวลาที่เหนื่อยล้า

และสุดท้ายนี้ก็ขอขอบคุณบุคคลที่สำคัญที่สุดในชีวิตที่ทำให้ข้าพเจ้ามีวันนี้ นั่นคือ บิดา มารดา และบุคคลในครอบครัว อันเป็นที่เคารพรัก ซึ่งได้เลี้ยงดู คอยสั่งสอนข้าพเจ้ามาเป็นอย่างดี พร้อมให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลัง ความรักเสมอมา ข้าพเจ้าขอกราบพระคุณ ณ ที่นี้ด้วย

สุวพงษ์ ไชยมี

อาร์ม ชัยอรุณดีกุล

สารบัญ

สารบัญรูปภาพ

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

ปัจจุบันนี้ ได้มีการนำเทคโนโลยีเข้ามาเป็นส่วนประกอบในชีวิตประจำวันมากขึ้นเรื่อยๆ ในทุกๆด้าน เพื่อที่จะช่วยอำนวยความสะดวกสบายให้แก่ผู้ใช้ และรวมทั้งยังสามารเป็นการประหยัดเวลาในการทำสิ่งต่างๆไม่ให้ล่าช้าเหมือนเช่นดังแต่ก่อนด้วย ซึ่งเทคโนโลยีนี้สามารถนำไปประยุกต์ใช้ร่วมกับสิ่งต่างๆที่เกี่ยวข้องกับชีวิตของคนเราได้แทบจะทุกเรื่อง

ในการที่เทคโนโลยีก้าวหน้าขึ้นเรื่อยๆนั้น ก็ได้มีการพัฒนาเกี่ยวข้องกับเรื่องของการสร้างภาพจำลองของสิ่งของต่างๆ ซึ่งรวมถึงการสร้างภาพจำลองของสิ่งที่ยังไม่ได้เกิดขึ้นจริงๆขึ้น

ตัวอย่างเช่นการเลือกซื้อเสื้อผ้าสำเร็จรูปสักตัวนั้น ผู้ที่ต้องการจะซื้อเสื้อผ้านั้นต้องทำการลองเสื้อในร้านที่ต้องการจะซื้อเสื้อผ้านั้น โดยในการลองเสื้อนั้น ต้องเปลี่ยนเสื้อผ้าในการลองซึ่งเป็นเรื่องที่เสียเวลามาก เพราะหากเมื่อทำการลองเสื้อผ้าแล้วได้เสื้อผ้าที่ไม่ถูกใจหรือขนาดที่ไม่ถูกใจก็ต้องนำเสื้อผ้ามาลองใหม่อีก ซึ่งจะทำให้เสียเวลาอย่างมากในการลองเสื้อผ้าใหม่อีกครั้ง และอาจจะมีห้องลองเสื้อที่จำกัดทำให้ต้องเสียเวลารอคิวในการเข้าใช้นาน ซึ่งจะยิ่งทำให้เสียเวลามากขึ้นไปอีก ดังนั้น จึงได้ทำให้เกิดโครงงานนี้ขึ้น เพื่อที่ให้ผู้ที่ต้องการจะลองเสื้อผ้าสามารถลองเสื้อผ้าได้โดยไม่ต้องทำการลองเสื้อผ้าเองจริงๆ โดยการสร้างโปรแกรมประยุกต์เพื่อใช้จำลองรูปร่างของผู้ใช้งานเพื่อนำไปลองเสื้อผ้าที่มีอยู่แล้วหรือเสื้อผ้าที่คาดว่ากำลังจะผลิตออกมาวางจำหน่ายในโปรแกรมประยุกต์ เพื่อความสะดวกสบายในการลองเสื้อผ้า

ในการสร้างโปรแกรมประยุกต์ที่ใช้สำหรับสวมใส่เสื้อผ้านั้น โปรแกรมประยุกต์ควรเป็นโปรแกรมที่แสดงผลเป็นภาพสามมิติเพื่อที่จะทำให้การลองเสื้อผ้าดูสมจริงมากเท่าที่จะเป็นไปได้และดูมีความน่าสนใจมากยิ่งขึ้น จึงได้มีการนำการแสดงผลแบบภาพสามมิติเข้ามาใช้ในการพัฒนาโปรแกรม

สำหรับโครงการนี้ได้เป็นการนำเสนอแบบจำลองของห้องเสื้อเสมือนในร้านขายเสื้อผ้าต่างๆโดยได้มีการสร้างหุ่นจำลองสามมิติสำหรับจำลองรูปร่างของผู้ใช้ ในการสร้างหุ่นจำลองสามมิตินั้นจะใช้โปรแกรมในการช่วยสร้างคือ โปรแกรม ทรีดี สตูดิโอ แม็กซ์ เวอร์ชั่น 5.0 (3D Studio Max 5.0) และโปรแกรม โพสเซอร์ เวอร์ชั่น 5 (Poser 5) ส่วนการเขียนโปรแกรมเพื่อให้เครื่องที่ใช้สามารถแสดงผลภาพ 3 มิติได้นั้น จะใช้คอมโพเน็นท์ ไดเร็กซ์เอ็กซ์ กราฟฟิก (DirectX Graphics) ของไดเร็กซ์เอ็กซ์ 9 (DirectX 9) สำหรับการรับค่าจากอุปกรณ์อินพุท เช่น เมาส์(Mouse), คีย์บอร์ด(keyboard) จะใช้คอมโพเน็นท์ ไดเร็กซ์อินพุท (DirectInput) ของ ไดเร็กซ์เอ็กซ์ 9 (DirectX 9) และใช้เครื่องมือในการสร้างโปรแกรม โดยโปรแกรม ไมโครซอฟท์ วิชวล ซีพลัสพลัส 6.0 (Microsoft Visual C++ 6.0) ในการเขียนโปรแกรมด้วยภาษาซีพลัสพลัส(C++) โดยการสร้างโปรเจคในรูปแบบ MFC เพื่อให้โปรแกรมมีรูปแบบที่ตอบสนองกับผู้ใช้มากขึ้น (interaction program)

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อให้ผู้ที่ต้องการจะทำการลองสวมใส่เสื้อผ้า ได้เห็นตัวเองได้ใส่เสื้อผ้านั้นโดยไม่ต้องทำการสวมใส่เองจริงๆ

1.2.2 เพื่อศึกษาเทคโนโลยี ในการสร้างโปรแกรมประยุกต์จำลอง ภาพ 3 มิติโดยใช้คอมโพเน็นท์ ไดเร็กซ์เอ็กซ์กราฟิค ของ ไดเร็กซ์เอ็กซ์ 9

1.2.3 เพื่อศึกษาการพัฒนาโปรแกรมประยุกต์ที่ใช้ติดต่อกับอุปกรณ์อินพุทต่างๆ โดยใช้คอมโพเน็นท์ ได้เร็กซ์อินพุท ของ ไดเร็กซ์เอ็กซ์ 9

1.2.4 เพื่อศึกษาการพัฒนาโปรแกรมประยุกต์ ด้วยการเขียนโปรแกรมในเชิงวัตถุ ด้วยภาษาซีพลัสพลัส

1.2.5 เพื่อพัฒนาโปรแกรมประยุกต์ให้มีการตอบสนองต่อผู้ใช้งาน

1.3 ขอบเขตของโครงงาน

โครงงานนี้จะเป็นการสร้างโปรแกรมประยุกต์ที่จำลองห้องลองเสื้อขึ้นมาโดยจะมีรูปแบบเสื้อผ้าให้เลือกสำหรับการลอง

โดยภายในห้องลองเสื้อจำลองจะมีหุ่นจำลอง โดยสามารถเลือกเพศของหุ่นจำลองได้ และสามารถ ปรับขนาดของหุ่นจำลองได้ ตามสัดส่วนของผู้ที่ใช้โปรแกรม โดยมีรูปแบบการนำเสนอออกมาในรูปแบบของกราฟฟิก 3 มิติ เพื่อใช้สำหรับลองเสื้อผ้า โดยการรับขนาดของสัดส่วนนั้นผู้ใช้จะต้องทำการใส่ค่าของสัดส่วนต่างๆ ลงใน “Dialog Box” ที่สร้างไว้สำหรับในแต่ละสัดส่วน เช่นการเพิ่มความยาวของแขน เพิ่มขนาดของรอบอก โดยในหุ่นจำลองสามมิติของผู้หญิงนั้นสามารถเลือกขนาดของหน้าอก (CUP) ได้ด้วย เมื่อทำการสร้างหุ่นจำลองเสร็จสิ้นแล้วก็นำหุ่นจำลองที่สร้างนั้นมาทำการใส่เสื้อผ้า โดยเสื้อผ้านั้นจะมีการปรับตามขนาดของหุ่นจำลองเพื่อให้ดูเมื่อหุ่นนั้นได้ใส่เสื้อผ้าอยู่ โดยสามารถบังคับหุ่นจำลองสามมิติให้สามารถหมุนรอบตัวเองได้ ขยับตัวได้ในบางส่วนเช่น แขน และ สามารถบังคับกล้องเคลื่อนที่ เข้าและออก จากตัวหุ่นจำลองได้

1.4 วิธีการดำเนินงาน

1.4.1 ศึกษาภาพรวมของโครงงานและศึกษาเพื่อเลือกเครื่องมือในการใช้พัฒนาโครงการ

1.4.2 วางขอบเขตของการทำงานของโปรแกรมประยุกต์

1.4.3 วิเคราะห์และออกแบบระบบ และ กำหนดเครื่องมือที่จะนำไปพัฒนาโปรแกรมประยุกต์

1.4.4 ศึกษาการพัฒนาโปรแกรมประยุกต์ด้วยโปรแกรมไมโครซอฟท์ วิชวล ซีพลัสพลัส 6.0 สำหรับในการจำลองภาพกราฟฟิค 3 มิติ ด้วย ไดเร็กซ์เอ็กซ์ 9.0 และ ในการสร้างภาพกราฟฟิค 3 มิติ เช่น ทรีดี สตูดิโอ แม็กซ์ 5.0

1.4.5 ศึกษาการพัฒนาโปรแกรมโดยการรับค่าจากอุปกรณ์อินพุทด้วย คอมโพเน็นท์ ไดเร็กซ์อินพุท

1.4.6 ทำการออกแบบละสร้างส่วนควบคุมต่างๆของโปรแกรมประยุกต์

1.4.7 นำโปรแกรมในแต่ละส่วนมาทดสอบ

1.4.8 นำโปรแกรมในแต่ละส่วนมารวมเข้าด้วยกัน

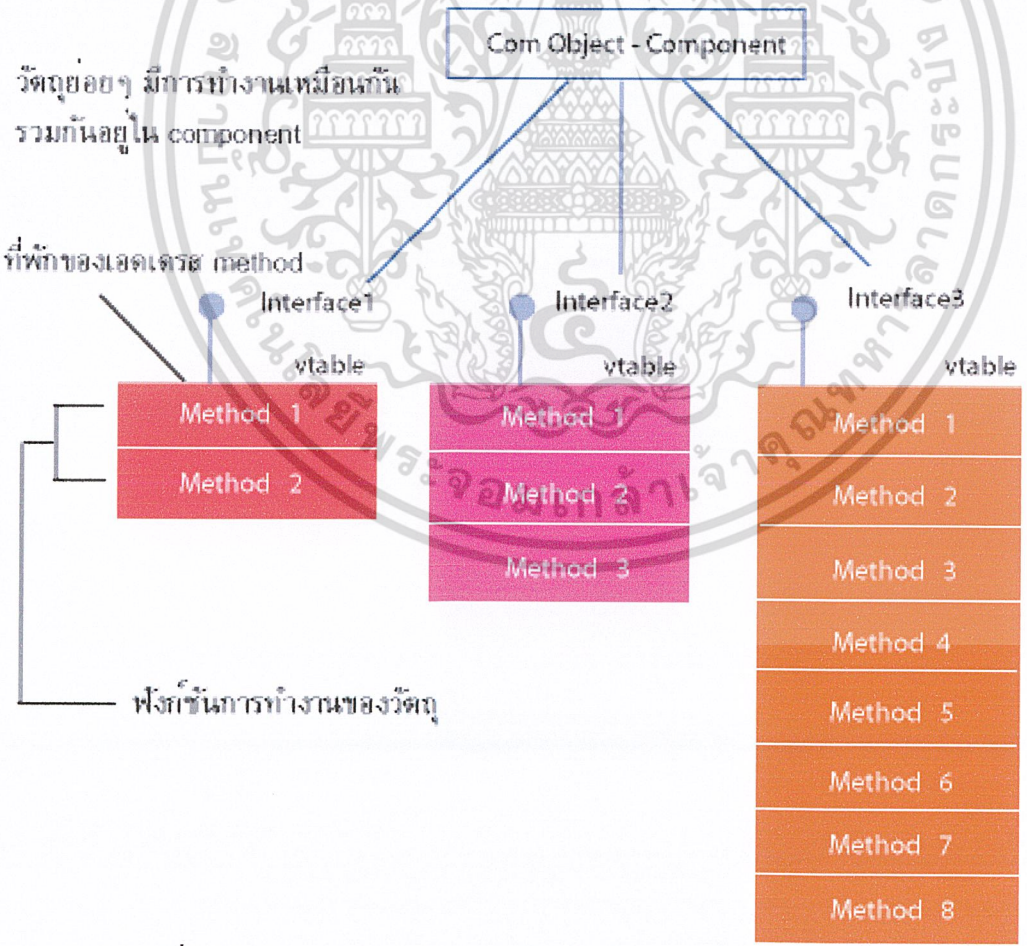
1.4.9 แก้ไขส่วนที่ผิดพลาดและเพิ่มเติมการทำงานบางอย่างเข้าไปเพื่อในมีการงานที่ถูกต้อง

บทที่ 2
ทฤษฎีและหลักการคอมโพเน็นท์กับไดเร็กซ์เอ็กซ์

2.1 คอมโพเน็นท์ออบเจ็กท์โมเดล (Component Object Model – COM)

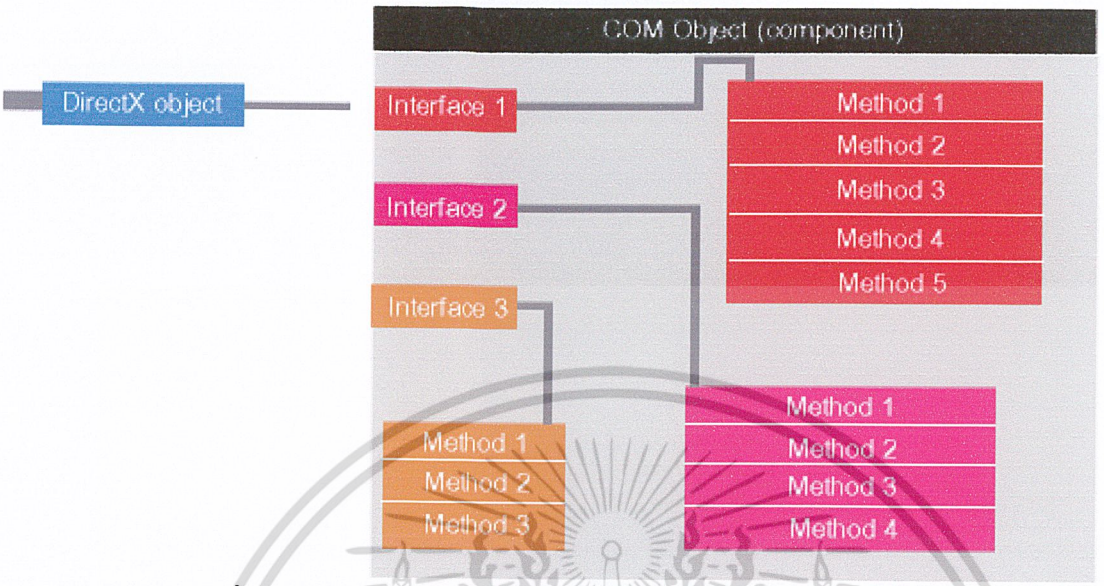
หลักการคอมโพเน็นท์ออบเจ็กท์โมเดลนี้ เป็นวิธีที่ทางไมโครซอฟท์ คิดขึ้นมาเพื่อทำให้ วัตถุต่างๆ ที่มีความเกี่ยวข้องหรือไม่เกี่ยวข้องกันก็ตาม สามารถติ ดต่อแลกเปลี่ยนข้อมูลกันได้ และสามารถนำส่วนการใช้งานมาใช้ใหม่ในโปรแกรมที่จะเขียนขึ้นใหม่ได้ โดยไม่ขึ้นกับภาษาที่ใช้เขียน ไม่ขึ้นกั บระบบปฏิบัติการ แนวคิดวิธีที่จะทำให้ได้ผลดังกล่าว คือ ใช้หลักการออกแบบ วัตถุ (Object) ให้เป็นคอมโพเน็นท์ คือเป็นการรวมเอาวัตถุ (Object) ย่อยๆ ต่างๆ ที่มีหน้าที่การทำงานในแบบเดียวกัน มารวมกันเป็นคอมโพเน็นท์ โดยเรียก วัตถุ (Object) ย่อยๆ เหล่านั้นใหม่ว่าอินเตอร์เฟส (Interface)

คอมโพเน็นท์นั้น จะเป็นชุดที่รวมเอาวัตถุย่อยๆ หรืออินเตอร์เฟส ที่มีฟังก์ชั่นการทำงานในแบบเดียวกัน และในแต่ละอินเตอร์เฟสนั้น จะมีดีไวซ์ตัวหนึ่งที่มีชื่อว่า วีเทเบิล (vtable) สำหรับพักแอดเดรสของฟังก์ชั่น (Method) ตามในรูปที่ 1



รูปที่ 2-1 แสดงความสัมพันธ์ของคอมโพเน็นท์, อินเตอร์เฟสและฟังก์ชั่น

2.2 ไดเร็กซ์เอ็กซ์ออบเจ็กท์ กับ คอมโพเน็นท์ออบเจ็กท์



รูปที่ 2-2 แสดงการติดต่อกันระหว่าง COM ออบเจ็กท์ กับ ไดเร็กซ์ ออบเจ็กท์

คอมโพเน็นท์ เปรียบเสมือน กล่องที่มีรูหรือช่องไว้ให้เสียบ ซึ่งอาจจะมีหลายๆ รู แต่ละรูเปรียบเสมือนอินเตอร์เฟส โดยแต่ละอินเตอร์เฟส ก็จะมีฟังก์ชั่นการทำงาน (Method) สำหรับไดเร็กซ์เอ็กซ์ออบเจ็กท์ (DirectX Object) นั้น ก็เปรียบเสมือนแจ็ค ที่สามารถเสียบที่รูหรืออินเตอร์เฟสได้ หากวัตถุของไดเร็กเอ็กซ์ เสียบเข้าที่อินเตอร์เฟสใด ก็สามารถเรียกใช้งานฟังก์ชั่นการทำงานในอินเตอร์เฟสนั้นๆได้ หากต้องการใช้งานฟังก์ชั่นการทำงานอื่น ที่อยู่ในอินเตอร์เฟสอื่นๆ ก็ต้องสร้างไดเร็กซ์เอ็กซ์ออบเจ็กท์ขึ้นมาใหม่ เพื่อนำไปเสียบกับอินเตอร์เฟสที่จะใช้งานฟังก์ชั่นการทำงานที่ต้องการ และถ้าเลิกใช้อินเตอร์เฟสใด ก็ต้องถอดแจ็คหรือวัตถุนั้นออก

กล่าวได้ว่า อินเตอร์เฟสแต่ละอัน ก็จะมีฟังก์ชั่นการทำงานต่างๆ ของตัวเอง หากต้องการจะใช้งานฟังก์ชั่นการทำงานได้ จะต้องสร้างไดเร็กซ์เอ็กซ์ออบเจ็กท์ที่ชี้ไปยังอินเตอร์เฟสที่มีฟังก์ชั่นการทำงานที่ต้องการ โดยการสร้างไดเร็กซ์เอ็กซ์ออบเจ็กท์นี้ สามารถทำได้โดยการเรียกใช้ฟังก์ชั่น ที่ ไดเร็กซ์เอ็กซ์เตรียมไว้ให้

การปรับปรุงความสามารถของวัตถุประเภท COM จะใช้การสร้างอินเตอร์เฟสตัวใหม่ขึ้นมา เพื่อใช้รองรับความสามารถใหม่มากกว่าจะเปลี่ยนแปลงฟังก์ชั่นที่มีอยู่ในอินเตอร์เฟสปัจจุบัน

2.3 อินเตอร์เฟสของ COM ออบเจ็กท์

ทุก COM ออบเจ็กท์ จะสนับสนุนอินเตอร์เฟสที่ชื่อ IUnknown อินเตอร์เฟสนี้เป็นอินเตอร์เฟสมาตรฐาน ซึ่งคอยควบคุมความสามารถในการเข้าใช้งานของไดเร็กซ์เอ็กซ์ออบเจ็กท์ ที่เข้ามาใช้งาน

อินเตอร์เฟสต่างๆ โดยทุกอินเตอร์เฟสจะสืบทอดมาจากคลาส IUnknown ซึ่งเป็นคลาสที่ใช้สร้าง IUnknown อินเตอร์เฟส

สำหรับ IUnknown อินเตอร์เฟสนี้จะมีเมธอดอยู่ 3 ตัวด้วยกันคือ AddRef(), QueryInterface()และ Release() ดังนั้นอินเตอร์เฟสตัวอื่นๆ ที่สืบทอดไปจึงมีเมธอดทั้ง 3 นี้ด้วยเสมอ สังเกตได้จาก ไฟล์Header สำหรับใช้งานอินเตอร์เฟส เช่น C:\DXSDK\include\dinput.h ในบรรทัดที่มีการใช้ #define INTERFACE จะมีเมธอดทั้ง 3 ที่สืบทอดมาจาก IUnknow อยู่บรรทัดแรกๆ ซึ่ง COM ออบเจ็กท์ จะใช้ IUnknown เป็นตัวจัดการไดเร็กซ์เอ็กซ์ออบเจ็กท์ที่เข้ามาใช้อินเตอร์เฟส ที่อยู่ใน COM ออบเจ็กท์นั้นๆ

• AddRef() จะทำการเพิ่มค่าการนับแก่ reference count ของออบเจ็กท์ ไปอีก 1 เมื่ออินเตอร์เฟสหรือโปรแกรมประยุกต์อื่นๆ เข้าใช้งานออบเจ็กท์นั้นๆ

• QueryInterface() ตรวจดูว่าออบเจ็กท์นั้นมีอินเตอร์เฟสที่ต้องการหรือไม่ และยังสามารถใช้สร้างไดเร็กซ์เอ็กซ์ออบเจ็กท์ตัวใหม่ จากไดเร็กซ์เอ็กซ์ออบเจ็กท์เดิมที่มีอยู่ได้ด้วย โดยจะไปสอบถาม COM ออบเจ็กท์ ซึ่งเป็นเจ้าของอินเตอร์เฟสที่ไดเร็กซ์เอ็กซ์ออบเจ็กท์ตัวเดิมกำลังใช้อยู่ว่า มีอินเตอร์เฟสที่ไดเร็กซ์เอ็กซ์ออบเจ็กท์ตัวใหม่ต้องการหรือไม่

• Release() ลดค่าการนับแก่ reference count ของออบเจ็กท์ ลงไป 1 เมื่อลดลงไปถึง 0 ออบเจ็กท์จะถูกทำลาย

ทุกครั้งที่สร้างไดเร็กซ์เอ็กซ์ออบเจ็กท์ ซึ่งก็คือ การต้องการจะติดต่อกับอินเตอร์เฟสตัวใดในคอมโพเน็นท์ของไดเร็กซ์เอ็กซ์ โดย COM ออบเจ็กท์ จะเรียก QueryInterface() เพื่อตรวจดู่ามีอินเตอร์เฟสที่ ต้องการใช้มีอยู่ในออบเจ็กท์หรือไม่ หากมี จะเตรียมพื้นที่ในหน่วยความจำแล้วส่งตำแหน่งแอดเดรสกลับมา พร้อมทั้งเรียกใช้ AddRef() เพื่อเพิ่มการนับแก่ Reference Count อีก 1 ถ้ามีไดเร็กซ์เอ็กซ์ออบเจ็กท์ตัวอื่นๆ มาใช้อินเตอร์เฟสอื่นๆ หรือใช้อินเตอร์เฟสซ้ำกันกับไดเร็กซ์เอ็กซ์ออบเจ็กท์ตัวอื่น Reference Count ก็จะถูกบวกค่าเพิ่มขึ้นไปเรื่อยๆ

โดยหน่วยความจำที่ไดเร็กซ์เอ็กซ์ออบเจ็กท์ใช้นี้จะยังคงอยู่ตลอด หากไม่ทำการเรียกใช้ Release() แม้จะปิดโปรแกรมไปแล้วก็ตาม ซึ่งทำให้เกิดปัญหาเรื่องทรัพยากรตอบสนองไม่เพียงพอ หรือ Low resource ดังนั้น จะต้องเรียกใช้ Release() เมื่อไม่ต้องการใช้งานแล้วเสมอ เพื่อเป็นการคืนหน่วยความจำ และลดค่า Referenc Count ทีละ 1 เมื่อลดจนเหลือ 0 ก็แสดงว่า ไม่มีไดเร็กซ์เอ็กซ์ออบเจ็กท์ตัวใดใช้ COM ออบเจ็กท์นั้นแล้ว กลไกภายใน COM ก็จะลบ COM ออบเจ็กท์ ตัวนั้นออกไปจากหน่วยความจำโดยอัตโนมัติ

2.4 ไดเร็กซ์เอ็กซ์ 9 (DirectX 9)

2.4.1 ความรู้เบื้องต้นเกี่ยวกับไดเร็กซ์เอ็กซ์

ไมโครซอฟท์ ไดเร็กซ์เอ็กซ์ (Microsoft DirectX) เป็นชุดของแอพพลิเคชั่นโปรแกรมมิ่ง อินเตอร์เฟส (Application Programming Interface – API) ได้รับการพัฒนาและออกแบบเพื่อใช้จัดเตรียมอินเตอร์

เฟส สำหรับควบคุมฮาร์ดแวร์ทางด้านมัลติมีเดีย บนระบบปฏิบัติการ ไมโครซอฟท์ วินโดว์ (Microsoft Windows) ได้อย่างมีประสิทธิภาพ และเป็นเครื่องมือให้โปรแกรมเมอร์ใช้คำสั่งควบคุมฮาร์ดแวร์ได้อย่างใกล้ชิด โดยไม่ต้องยุ่งยากกับการสร้างส่วนโปรแกรมติดต่อในระดับล่าง ซึ่งเครื่องมือที่ใช้ติดต่อจแตกต่างกันไปตามประเภทของอุปกรณ์ ทำให้โปรแกรมเมอร์สร้างซอฟต์แวร์เพื่อทำงานทางด้านมัลติมีเดียได้เป็นอย่างดี เนื่องจากการติดต่อในแต่ละคอมโพเน็นท์จะเป็นของอุปกรณ์ประเภทหนึ่งๆ เป็นอิสระในการเขียนโปรแกรม

API ของ ไดเร็กซ์เอ็กซ์นั้น สร้างขึ้นมาบนรากฐานของ HAL (Hardwaer Abstraction Layer) สามารถซ่อนลักษณะของดีไวซ์ (Device) ที่เกี่ยวพันอยู่กับฮาร์ดแวร์ และออกแบบมาเพื่อให้สามารถรองรับความขยายตัวของฮาร์ดแวร์ใหม่ๆ ที่จะออกมาในอนาคต ดังนั้นจึงสามารถรองรับความสามารถของฮาร์ดแวร์ที่มีความสามารถเร่งความเร็วใหม่ๆ ที่ปัจจุบันยังไม่มี ด้วยความสามารถในการจำลองการทำงานผ่าน HEL (Hardware Emulation Layer) หรือเลือกที่จะหลีกเลี่ยงไม่ใช้ความสามารถนี้ ถ้า HEL ไม่สนับสนุน ทำให้โปรแกรมเมอร์พัฒนางานทางด้านมัลติมีเดียง่ายขึ้น ไม่จำเป็นต้องทดสอบกับอุปกรณ์ทุกตัวในท้องตลาด เพียงแต่ทดสอบกับไดเร็กซ์เอ็กซ์ก็เพียงพอแล้ว สำหรับผู้สร้างอุปกรณ์ฮาร์ดแวร์ ก็ต้องสร้าง ไดรเวอร์ของอุปกรณ์นั้นๆ เพื่อให้สามารถทำงานเข้ากับไดเร็กซ์เอ็กซ์ได้

2.4.2 ชุดพัฒนาซอฟต์แวร์สำหรับไดเร็กซ์เอ็กซ์ (DirectX software Development Kit : DirectX SDK)

ชุดพัฒนาซอฟต์แวร์สำหรับไดเร็กซ์เอ็กซ์ นี้ประกอบด้วยชุดของไลบรารี่, ไฟล์ DLL, ไฟล์ header รวมไปถึงเอกสาร และตัวอย่างโปรแกรม ซึ่งช่วยในการพัฒนาซอฟต์แวร์ทางด้านมัลติมีเดีย ซึ่งการที่จะใช้ไดเร็กซ์เอ็กซ์ได้นั้น ก็ต้องทำการลงชุดพัฒนาซอฟต์แวร์สำหรับไดเร็กซ์เอ็กซ์นี้ก่อน

2.4.3 คอมโพเน็นท์ของไดเร็กซ์เอ็กซ์ 9

ไดเร็กซ์เอ็กซ์ 9 (DirectX 9) จะประกอบด้วย Component 6 ตัวคือ

2.4.3.1 Direct3D

ไดเร็กซ์ทรีดีเป็นกราฟฟิก API ที่ใช้ในการสร้างภาพสามมิติโดยจะเป็นตัวกลางระหว่างแอพพลิเคชันและอุปกรณ์กราฟฟิก (การ์ดจอ) ความสัมพันธ์ระหว่างแอพพลิเคชัน, ไดเร็กซ์สามดี และฮาร์ดแวร์

2.4.3.2 DirectMusic

ไดเร็กซ์มิวสิคเป็น คอมโพเน็นท์ ซึ่งใช้จัดการเกี่ยวกับไฟล์เสียงเพลง(.mid)โดยโหลด ไฟล์มาเก็บไว้ใน DirectMusic Buffer ก่อน จากนั้นจึงค่อยเล่นเพลงตามลำดับที่ต้องการ

2.4.3.3 DirectSound

ไดเร็กซ์ซาวด์เป็น คอมโพเน็นท์ ซึ่งใช้จัดการเกี่ยวกับไฟล์เสียงก็มีการทำงานคล้ายๆกับไดเร็กซ์มิวสิค โดยเป็นการใส่เสียง(.wav) ลงไป

2.4.3.4 DirectInput

DirectInput ใช้จัดการทางด้านการรับข้อมูลจากอุปกรณ์อินพุต เช่นจาก คีย์บอร์ด, เมาส์, จอยสติ๊กเป็นต้น เพื่อเข้าถึงข้อมูลด้วยการติดต่อโดยตรงกับฮาร์ดแวร์ด้วยฮาร์ดแวร์ไดรเวอร์ (Hardware Driver) ซึ่งจะทำงานได้เร็วกว่าการติดต่อด้วยเมสเสจของวินโดว์ นอกจากนี้ยังสนับสนุนการทำงานกับอุปกรณ์แบบแรงตอบสนอง (Force Feedback) อีกด้วย

2.4.3.5 DirectPlay

DirectPlay เป็นมีเดียที่เป็นอิสระบนเน็ตเวิร์ก ซึ่งหมายถึง สามารถรันได้บนเครือข่ายแบบ TCP/IP, เครือข่ายแบบ IPX หรือแม้แต่การต่อตรงจากโมเด็ม และการเป็นอิสระทางเน็ตเวิร์กนี้จะสามารถถูกพัฒนาต่อไปได้ในอนาคต นั่นคือ สามารถรองรับอุปกรณ์และโปรโตคอลใหม่ๆ เช่น IPV6, multicast โดยปกติ DirectPlay ใช้จัดการการทำงานด้านเน็ตเวิร์ค โดยไม่จำเป็นต้องไปยุ่งกับการทำงานในระดับล่าง มีหน้าที่การทำงานสำหรับการส่งข้อมูลทีเดียวให้หลายๆเครื่องเป็นกรุ๊ปได้ สามารถเขียนโปรแกรมติดต่อทางด้านเน็ตเวิร์ค ทั้งในการติดต่อผ่านเน็ตเวิร์คระหว่างโปรแกรมแบบมัลติเพลเยอร์ (Multiplayer) แบบเพียร์ทูเพียร์ (Peer-to-Peer) และแบบไคลเอ็นท์/เซิร์ฟเวอร์ (Client/Server)

2.4.3.6 DirectShow

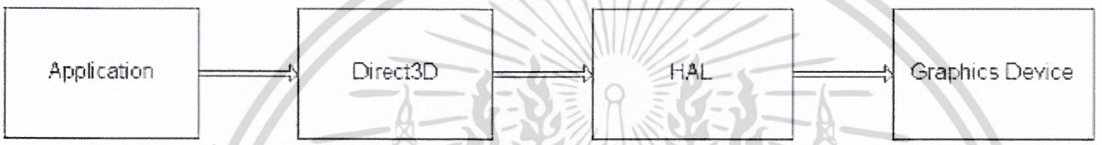
DirectShow ใช้สำหรับการทำงานด้านมัลติมีเดีย เป็นสถาปัตยกรรมทางด้านมีเดียสตรีมมิ่ง (media-streaming) ของไมโครซอฟท์ วินโดว์ ที่มีความสามารถในการแคปเจอร์ (Capture) สูงสามารถบรรจุข้อมูลที่เป็นวีดีโอ และการบีบอัดข้อมูลไฟล์ (Audio data compressed) ได้หลายรูปแบบ เช่น การเล่นไฟล์วีดีโอ ประเภท MPEG, Audio-Video(AVI), MPEG-1 Layer3(MP3) และไฟล์นามสกุล .wav เป็นต้น

บทที่ 3

Direct 3D

3.1 ความรู้เบื้องต้นเกี่ยวกับไดเร็กซ์ทรีดี

ไดเร็กซ์ทรีดี เป็นแอพพลิเคชั่นโปรแกรมมิ่งอินเทอร์เฟสระดับล่าง ซึ่งเป็นเครื่องมือสำหรับโปรแกรมเมอร์ที่ต้องการสร้างเกมส์หรือโปรแกรมมัลติมีเดียที่แสดงผล 3 มิติ ที่มีประสิทธิภาพสูง บนระบบปฏิบัติการวินโดว์ ซึ่งการใช้เอพีไอไดเร็กซ์ทรีดีนี้ ทำให้โปรแกรมสามารถติดต่อโดยตรงกับฮาร์ดแวร์เร่งความเร็วในระดับล่าง จึงมีความยืดหยุ่นในการใช้งานสูง โดยแอพพลิเคชั่น, ไดเร็กซ์ทรีดี, และฮาร์ดแวร์มีความสัมพันธ์ดังรูปที่ 1



รูปที่ 3-1 ความสัมพันธ์ระหว่างแอพพลิเคชั่น, ไดเร็กซ์ทรีดี, และฮาร์ดแวร์

จากรูปที่หนึ่งจะเห็นได้ว่ามี ฮาร์ดแวร์แอบสเตร็กชันเลเยอร์ (HAL-Hardware Abstraction Layer) ขั้นอยู่ระหว่างไดเร็กซ์ทรีดีและกราฟฟิคดีไวซ์ เหตุที่ไดเร็กซ์ทรีดี ไม่สามารถติดต่อกับฮาร์ดแวร์ดีไวซ์ได้โดยตรงนั้น เนื่องจากฮาร์ดแวร์ดีไวซ์ในท้องตลาดมีมากมายหลายชนิด และแต่ละชนิดก็มีความสามารถและการนำไปใช้งานที่ต่างกัน ดังนั้น ไดเร็กซ์ทรีดีจึงต้อง HAL ไปใช้ในการติดต่อซึ่งดีไวซ์ HAL นี้จะจัดการความเร็วทางด้านฮาร์ดแวร์ บนพื้นฐานการทำงานที่สนับสนุนกราฟฟิคดีไวซ์

3.2 การเริ่มต้นการตั้งค่าให้กับไดเร็กซ์ทรีดี

ในการเริ่มต้นนำไดเร็กซ์ทรีดีไปใช้งานสามารถทำได้ตามขั้นตอนดังต่อไปนี้

1. ชี้ pointer ไปยัง IDirect3D9 อินเตอร์เฟส ซึ่งอินเตอร์เฟสนี้ถูกใช้สำหรับค้นหาข้อมูลที่เกี่ยวกับฟิสิคอลฮาร์ดแวร์ดีไวซ์บนระบบ และการสร้าง IDirect3DDevice9 อินเตอร์เฟส ซึ่งเป็นออฟเจ็กของ ซีพลัสพลัส ซึ่งใช้แทนฟิสิคอลฮาร์ดแวร์ดีไวซ์ที่เราใช้สำหรับการแสดงกราฟฟิค 3 มิติ
2. ตรวจสอบความสามารถของอุปกรณ์ฮาร์ดแวร์ (กราฟฟิคการ์ด) จาก D3DCAPS9 โดยจะต้องตรวจสอบความสามารถในการทำฮาร์ดแวร์เวอร์แท็กโพรเซสซิ่ง(Hardware vertex processing) ก่อนที่จะทำการสร้าง IDirect3DDevice9
3. ตั้งค่าให้กับ D3DPRESENT_PARAMETERS สตรัคเจอร์ ซึ่งสตรัคเจอร์นี้จะประกอบด้วยตัวแปรที่ใช้กำหนดค่าเฉพาะของ IDirect3Device9 อินเตอร์เฟสที่ซึ่งจะต้องถูกสร้างขึ้น

4. สร้างIDirect3Device9ออฟเจ็คตามการตั้งค่าโครงสร้าง D3DPRESENT_PARAMETERS โดย IDirect3DDevice9 ออฟเจค จะเป็นออฟเจคที่สร้างขึ้นเป็นตัวแทน ของอุปกรณ์ฮาร์ดแวร์ที่นำมาใช้ในการแสดงผลกราฟฟิค 3 มิติ

3.2.1 การได้มาของ IDirect3D9 อินเตอร์เฟส

การเริ่มต้นการตั้งค่าของไดเร็กซ์ทรีดี เริ่มโดยการได้มาของพอยท์เตอร์ ชี้ไปยัง IDirect3D9 อินเตอร์เฟส ซึ่งทำได้อย่างง่ายดายโดยการใช้ ไดเร็กซ์ทรีดีฟังก์ชันตามตัวอย่าง

```
IDirect3D9* _d3d9;
_d3d9 = Direct3DCreate9(D3D_SDK_VERSION);
```

โดย IDirect3D9 ออบเจ็ค จะถูกใช้เป็นสองอย่างคือ เป็นตัวระบุอุปกรณ์ และ ใช้ในการสร้างวัตถุ

IDirect3DDevice9 จะสามารถระบุได้ทั้งความสามารถ, โหมดในการแสดงภาพ, รูปแบบ และข้อมูลอื่น ๆ เกี่ยวกับการ์ดจอที่ใช้อยู่

3.2.2 การตรวจสอบสำหรับฮาร์ดแวร์เวอร์เท็กซ์โพรเซสซิ่ง

เมื่อทำการสร้าง IDirect3DDevice9 ออบเจ็ค ต้องมีการกำหนดชนิดของเวอร์เท็กซ์โพรเซสซิ่งที่นำมาใช้ร่วมกันก่อน ในการทำนั้นเราต้องทำการตั้งค่าเริ่มต้น D3DCAPS9 ตามความสามารถของ primary display adapter ซึ่งจะทำตามตัวอย่างดังนี้

```
HRESULT IDirect3D9::GetDeviceCaps(
UINT Adapter,
D3DDEVTYPE DeviceType,
D3DCAPS9 *pCaps
);
```

ซึ่งจะมีการใช้งานดังโค้ดข้างล่าง

```
// Fill D3DCAPS9 structure with the capabilities of the
// primary display adapter.
 D3DCAPS9 caps;
d3d9->GetDeviceCaps(
D3DADAPTER_DEFAULT, // Denotes primary display adapter.
deviceType, // Specifies the device type, usually D3DDEVTYPE_HAL.
&caps); // Return filled D3DCAPS9 structure that contains
// the capabilities of the primary display adapter.
// Can we use hardware vertex processing?
int vp = 0;
if( caps.DevCaps & D3DDEVCAPS_HWTRANSFORMANDLIGHT )
{
// yes, save in 'vp' the fact that hardware vertex
// processing is supported.
vp = D3DCREATE_HARDWARE_VERTEXPROCESSING;
}
else
{
// no, save in 'vp' the fact that we must use software
```

```
        // vertex processing.
        vp = D3DCREATE_SOFTWARE_VERTEXPROCESSING;
        }
```

โดยที่ตัวแปร vp นั้นจะเป็นตัวเก็บค่าเวอร์เท็กซ์ที่นำมาใช้

3.2.3 การกำหนดค่าใน D3DPRESENT_PARAMETERS สตรัคเจอร์

ในการกำหนดค่าใน D3DPRESENT_PARAMETERS สตรัคเจอร์ นั้นจะใช้ลักษณะเฉพาะบางอย่างของ IDirect3DDevice9 ออบเจ็ค ซึ่งสามารถทำได้ดังนี้

```
typedef struct _D3DPRESENT_PARAMETERS_ {
        UINT BackBufferWidth;
        UINT BackBufferHeight;
        D3DFORMAT BackBufferFormat;
        UINT BackBufferCount;
        D3DMULTISAMPLE_TYPE MultiSampleType;
        DWORD MultiSampleQuality;
        D3DSWAPEFFECT SwapEffect;
        HWND hDeviceWindow;
        BOOL Windowed;
        BOOL EnableAutoDepthStencil;
        D3DFORMAT AutoDepthStencilFormat;
        DWORD Flags;
        UINT FullScreen_RefreshRateInHz;
        UINT PresentationInterval;
} D3DPRESENT_PARAMETERS;
```

โดยมีตัวอย่างการกำหนดค่าดังนี้

```
D3DPRESENT_PARAMETERS d3dpp;
d3dpp.BackBufferWidth = 800;
d3dpp.BackBufferHeight = 600;
d3dpp.BackBufferFormat = D3DFMT_A8R8G8B8; //pixel format
d3dpp.BackBufferCount = 1;
d3dpp.MultiSampleType = D3DMULTISAMPLE_NONE;
d3dpp.MultiSampleQuality = 0;
d3dpp.SwapEffect = D3DSWAPEFFECT_DISCARD;
d3dpp.hDeviceWindow = hwnd;
d3dpp.Windowed = false; // fullscreen
d3dpp.EnableAutoDepthStencil = true;
d3dpp.AutoDepthStencilFormat = D3DFMT_D24S8; // depth format
d3dpp.Flags = 0;
d3dpp.FullScreen_RefreshRateInHz = D3DPRESENT_RATE_DEFAULT;
d3dpp.PresentationInterval = D3DPRESENT_INTERVAL_IMMEDIATE;
```

3.2.4 การสร้าง IDirect3DDevice9 อินเตอร์เฟส

หลังจากกำหนดค่าตัวแปรต่าง ๆ ของ D3DPRESENT_PARAMETERS แล้ว ก็จะสามารถสร้างวัตถุ IDirect3DDevice9 ด้วยคำสั่งข้างล่างนี้

```
HRESULT IDirect3D9::CreateDevice(
        UINT Adapter,
        D3DDEVTYPE DeviceType,
        HWND hFocusWindow,
        DWORD BehaviorFlags,
        D3DPRESENT_PARAMETERS *pPresentationParameters,
        IDirect3DDevice9** ppReturnedDeviceInterface
);
```

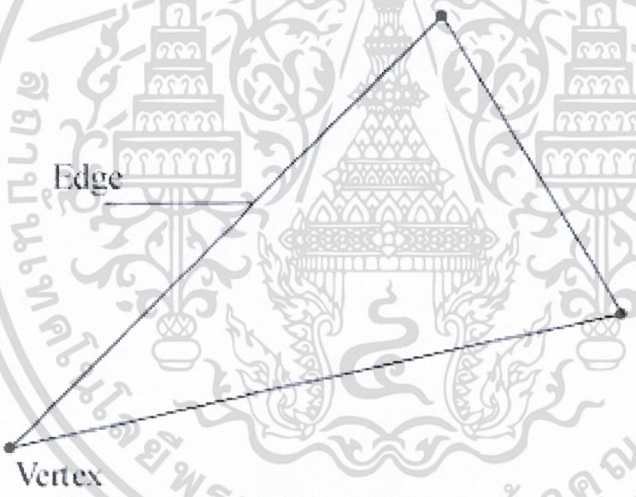
ตัวอย่างของการสร้าง IDirect3DDevice9

```
IDirect3DDevice9* device = 0;
hr = d3d9->CreateDevice(
        D3DADAPTER_DEFAULT, // primary adapter
        D3DDEVTYPE_HAL, // device type
        hwnd, // window associated with device
        D3DCREATE_HARDWARE_VERTEXPROCESSING, // vertex processing type
        &d3dpp, // present parameters
        &device); // returned created device
if( FAILED(hr) )
{
        ::MessageBox(0, "CreateDevice() - FAILED", 0, 0);
        return 0;
}
```

3.2 การเรนเดอร์ไปป์ไลน์

3.2.1 การแสดงรูปทรงสามมิติ

ฉาก (scene) ในคอมพิวเตอร์กราฟฟิก เกิดจากการรวมกันของวัตถุ และรูปทรง 3 มิติ โดยวัตถุจะถูกแสดงด้วยเครือข่ายสามเหลี่ยม (triangle mesh) เราจะเรียกรูปสามเหลี่ยมแต่ละรูปว่า รูปพื้นฐาน (primitive) นอกจากนี้ยังมีรูปพื้นฐานอื่น ๆ อีก เช่น เส้นตรง, จุด แต่รูปพื้นฐานเหล่านี้ไม่สามารถนำมาสร้างวัตถุ 3 มิติได้จุดที่เกิดจากการตัดกันของเอจ (edge) 2 เส้น จะเรียกว่า เวอร์เท็กซ์ (vertex) โดยที่รูปสามเหลี่ยมหนึ่ง ๆ จะมีเวอร์เท็กซ์ด้วยกัน 3 จุด ซึ่งแต่ละจุดจะเชื่อมต่อถึงกันด้วยเอจ ดังรูปที่ 2



รูปที่3-2 สามเหลี่ยมที่ถูกสร้างโดยจุดสามจุด

โดยการกำหนดกำหนดเวอร์เท็กซ์นั้นต้องใช้การสร้างสตรัคโดยมีวิธีการสร้างดังนี้

```
struct ColorVertex
{
        float _x, _y, _z; // position
        DWORD _color;
};
struct NormalTexVertex
{
        float _x, _y, _z; // position
        float _nx, _ny, _nz; // normal vector
        float _u, _v; // texture coordinates
};
```

การสร้างรูปทรงสามมิติที่เกิดจากสามเหลี่ยมหลายรูปนั้น จะต้องทำการสร้างอาร์เรย์เวอร์เท็กซ์ไว้เก็บเวอร์เท็กซ์ต่างๆ ซึ่งจะมีการกำหนดอาร์เรย์ดังนี้

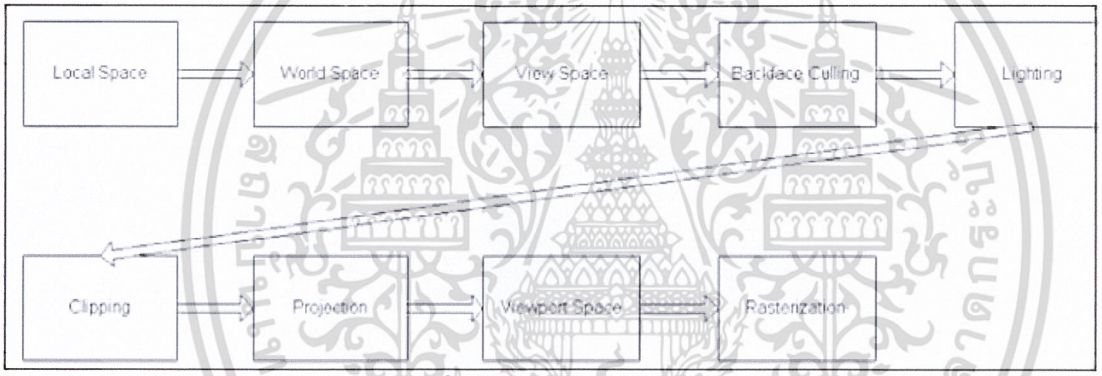
```
Vertex rect[6] = {v0, v1, v2, // triangle0
v0, v2, v3}; // triangle1
```

3.2.2 กล้องเสมือน

กล้องนั้นถือว่าเป็นส่วนหนึ่งของเวิร์ด ให้ผู้ที่ใช้งานสามารถมองเห็นส่วนของเวิร์ดซึ่งนำมาสร้างเป็นภาพ 2 มิติ โดยที่กล้องจะมีคุณสมบัติต่าง ๆ เช่น ตำแหน่ง (position), ทิศทางที่กล้องหันไป (orientation), ขอบเขตการมองเห็น (volume of space) เป็นต้น ซึ่งคุณสมบัติต่าง ๆ เหล่านี้จะเป็นตัวกำหนดภาพที่ผู้ใช้โปรแกรมสามารถมองเห็น

3.2.3 การเรนเดอร์ไปป์ไลน์ (Rendering Pipeline)

Rendering Pipeline คือ ลำดับอนุกรมของการทำงานตั้งแต่การสร้างวัตถุในโลก 3 มิติ จนไปถึงการนำไปแสดงผลบนหน้าจอที่เป็น 2 มิติ



รูปที่ 3-3 rendering pipeline

โดยแบ่งการทำงานเป็นเป็นขั้นตอนต่างๆดังนี้

1. **Local Space** เป็นขั้นตอนในการกำหนดตำแหน่งของสามเหลี่ยมที่ประกอบกันขึ้นเป็นวัตถุในระบบแกนโคออร์ดิเนตของวัตถุนั้น ๆ
2. **World Space** เป็นขั้นตอนในการกำหนดตำแหน่งของวัตถุซึ่งถูกสร้างในขั้นตอนของ Local space ให้อยู่ในตำแหน่งที่ต้องการภายในระบบแกนโคออร์ดิเนตของเวิร์ด
3. **View Space** เป็นขั้นตอนในการเปลี่ยนตำแหน่งกล้องให้มาอยู่ที่จุดออริจิน (0,0,0) และให้หน้ากล้องหันไปทางทิศ +z โดยการเปลี่ยนตำแหน่งกล้องนี้จะทำให้ตำแหน่งของวัตถุเปลี่ยนแปลงตามไปด้วย
4. **Backface Culling** เป็นขั้นตอนการตัดส่วนของวัตถุ (สามเหลี่ยม) ที่หันหลังให้กับกล้องซึ่งเป็นส่วนที่ไม่สามารถมองเห็นได้ออกไป
5. **Lighting** ขั้นตอนการคำนวณเรื่องแสงเพื่อให้ภาพที่ได้ออกมามีความสมจริงมากขึ้น

6. **Clipping** เป็นขั้นตอนการตัดวัตถุหรือส่วนของวัตถุที่อยู่ภายนอกขอบเขตการมองเห็น (Viewing Volume) ออกไป
7. **Projection** เป็นขั้นตอนในการทำให้วัตถุที่อยู่ใน 3 มิติ เกิดเป็นภาพ 2 มิติบนฉาก ซึ่งการทำโปรเจคชันนั้น มีอยู่หลายแบบ
8. **Viewport Transform** เป็นขั้นตอนการแปลงจากระบบแกนโคออร์ดิเนตของโปรเจคชันวินโดว์ (Projection window) ที่เกิดจากการทำโปรเจคชัน ให้กลายเป็นระบบแกนโคออร์ดิเนตของฉากสี่เหลี่ยมบนจอ ที่เรียกว่า Viewport
9. **Rasterization** เป็นขั้นตอนการใช้เทคนิคต่างๆ มาปรับคุณภาพของรูปให้ดูเรียบและสวยงามขึ้น

3.3 การวาดรูปในไดเร็กซ์ทรีดี

3.3.1 เวอร์เท็กซ์และอินเด็กซ์บัฟเฟอร์

ในการสร้างภาพ 3 มิตินั้น จะต้องสร้างโดยนำรูปสามเหลี่ยมซึ่งเป็นรูปพื้นฐานมาประกอบกันจนเกิดเป็นรูปที่ต้องการ และในการสร้างรูปสามเหลี่ยมแต่ละรูปนั้น จะต้องมีการกำหนดเวอร์เท็กซ์ให้กับรูปสามเหลี่ยมนั้น โดยเวอร์เท็กซ์ที่กำหนดจะเก็บอยู่ในหน่วยความจำที่เรียกว่า เวอร์เท็กซ์บัฟเฟอร์

นอกจากนี้ในการสร้างรูป 3 มิติที่ต้องมีการใช้เวอร์เท็กซ์ร่วมกัน จะต้องมีการใช้อินเด็กซ์บัฟเฟอร์สำหรับรูปสามเหลี่ยมแต่ละรูปเพื่อใช้เป็นที่เก็บค่าพอยน์เตอร์ที่ชี้ไปยังเวอร์เท็กซ์ต่าง ๆ ที่ถูกเก็บไว้ในเวอร์เท็กซ์บัฟเฟอร์

การสร้างเวอร์เท็กซ์และอินเด็กซ์บัฟเฟอร์สามารถสร้างได้ดังนี้

```
HRESULT IDirect3DDevice9::CreateVertexBuffer(
        UINT Length,
        DWORD Usage,
        DWORD FVF,
        D3DPOOL Pool
        IDirect3DVertexBuffer9** ppVertexBuffer,
        HANDLE* pSharedHandle
);
HRESULT IDirect3DDevice9::CreateIndexBuffer(
        UINT Length,
        DWORD Usage,
        D3DFORMAT Format,
        D3DPOOL Pool,
        IDirect3DIndexBuffer9** ppIndexBuffer,
        HANDLE* pSharedHandle
);
```

สำหรับตัวอย่างในการสร้างเวอร์เท็กซ์บัฟเฟอร์ที่ใช้เก็บเวอร์เท็กซ์จำนวน 36 เวอร์เท็กซ์ จะแสดงได้ดังโค้ดด้านล่าง

```
IDirect3DIndexBuffer9* ib;
_device->CreateIndexBuffer(
36 * sizeof( WORD ),
D3DUSAGE_DYNAMIC | D3DUSAGE_WRITEONLY,
D3DFMT_INDEX16,
D3DPOOL_MANAGED,
&ib,
0);
```

หลังจากได้สร้างเวอร์เท็กซ์บัฟเฟอร์และอินเด็กซ์บัฟเฟอร์ตามขนาดที่ต้องการใช้แล้ว ก็จะเข้าสู่ขั้นตอนของการกำหนดค่าเวอร์เท็กซ์ และอินเด็กซ์ที่จะเก็บลงไปในเวอร์เท็กซ์บัฟเฟอร์และอินเด็กซ์บัฟเฟอร์ โดยการจะเข้าไปอ่านหรือเขียนข้อมูลในบัฟเฟอร์ดังกล่าว จะต้องใช้ฟังก์ชัน Lock เพื่อที่จะเอาค่าพอยน์เตอร์ที่ชี้ไปยังบัฟเฟอร์ส่วนที่ต้องการอ่านหรือเขียนมาเสียก่อน โดยฟังก์ชัน Lock มีโครงสร้างดังนี้

```
HRESULT IDirect3DVertexBuffer9::Lock(
        UINT OffsetToLock,
        UINT SizeToLock,
        BYTE** ppbData,
        DWORD Flags
);
HRESULT IDirect3DIndexBuffer9::Lock(
        UINT OffsetToLock,
        UINT SizeToLock,
        BYTE** ppbData,
        DWORD Flags
);
```

และหลังจากที่ได้เขียนหรืออ่านค่าจากบัฟเฟอร์ที่ทำการ Lock เสร็จเรียบร้อยแล้ว จะต้องเรียกใช้ฟังก์ชัน Unlock ด้วย ดังแสดงในตัวอย่างโค้ดด้านล่าง

```
Vertex* vertices;
vb->Lock(0, 0, (void**)&vertices, 0); // lock the entire buffer

vertices[0] = Vertex(-1.0f, 0.0f, 2.0f); // write vertices to
vertices[1] = Vertex( 0.0f, 1.0f, 2.0f); // the buffer
vertices[2] = Vertex( 1.0f, 0.0f, 2.0f);

vb->Unlock(); // unlock when you're done accessing the buffer
```

3.3.2 การวาดภาพ (Rendering)

ก่อนที่จะวาดภาพ (Render) ในไดเร็กซ์สามดีนั้น จะต้องมีการกำหนดค่าให้กับ Render State ก่อน โดยไดเร็กซ์สามดีจะมี Render state ให้ปรับได้หลากหลายขึ้นอยู่กับความต้องการของโปรแกรมเมอร์ โดย Render state ดังกล่าวจะมีค่าดีฟอลต์อยู่ ซึ่งถ้าไม่มีการปรับค่าใด ๆ ก็จะเรียกใช้ค่าดีฟอลต์นั้น สำหรับวิธีการปรับค่า Render state นั้นจะใช้ฟังก์ชัน SetRenderState ดังแสดงด้านล่าง

```
HRESULT IDirect3DDevice9::SetRenderState(
D3DRENDERSTATETYPE State, // the state to change
DWORD Value // value of the new state
);
```

ตัวอย่างการปรับค่า render state

```
_device->SetRenderState(D3DRS_FILLMODE, D3DFILL_WIREFRAME);
```

3.3.3 การเตรียมตัวก่อนการวาดรูป

ในการเตรียมตัววาดรูปนั้นจะมี 3 ขั้นตอนดังต่อไปนี้

1. ปรับข้อมูลที่ต้องการวาด (Set the stream source) ซึ่งทำตามขั้นตอนดังต่อไปนี้

```
HRESULT IDirect3DDevice9::SetStreamSource(
        UINT StreamNumber,
        IDirect3DVertexBuffer9* pStreamData,
        UINT OffsetInBytes,
        UINT Stride
);
```

2. ตั้งค่ารูปแบบเวอร์เท็กซ์ (Set the vertex format) โดยหลังจากการกำหนดรูปแบบเวอร์เท็กซ์แล้ว การวาดภาพครั้งต่อ ๆ มา จะใช้รูปแบบของเวอร์เท็กซ์ตามที่ได้กำหนดไว้ล่าสุด

```
_device->SetFVF( D3DFVF_XYZ | D3DFVF_DIFFUSE | D3DFVF_TEX1 );
```

3. ตั้งค่าอินเด็กซ์บัฟเฟอร์ (Set the index buffer) ในกรณีที่มีการใช้อินเด็กซ์บัฟเฟอร์ จะต้องมีการกำหนดอินเด็กซ์บัฟเฟอร์ที่จะใช้ในการวาดด้วย โดยที่จะกำหนดอินเด็กซ์บัฟเฟอร์ในการวาดได้ครั้งละ 1 อินเด็กซ์บัฟเฟอร์เท่านั้น

```
_device->SetIndices( _ib ); // pass copy of index buffer pointer
```

3.3.4 การวาดรูปด้วยเวอร์เท็กซ์และอินเด็กซ์บัฟเฟอร์

หลังจากได้เตรียมข้อมูลก่อนการวาดภาพแล้ว ก็จะมาถึงขั้นตอนของการวาดภาพ 3 มิติ โดยการใช้ฟังก์ชัน DrawPrimitive หรือ DrawIndexedPrimitive โดย 2 ฟังก์ชันนี้จะนำค่าเวอร์เท็กซ์ที่อยู่ในเวอร์

DrawPrimitive

วิธีดังตัวอย่างนี้เป็นการใช้วิธี draw Primitive

```
HRESULT IDirect3DDevice9::DrawPrimitive(
        D3DPRIMITIVETYPE PrimitiveType,
        UINT StartVertex,
        UINT PrimitiveCount
);
```

ตัวอย่าง

```
// draw four triangles.
_device->DrawPrimitive( D3DPT_TRIANGLELIST, 0, 4);
```

DrawIndexedPrimitiv

วิธีดังตัวอย่างนี้เป็นการใช้วิธี draw Primitive

```
HRESULT IDirect3DDevice9::DrawIndexedPrimitive(
D3DPRIMITIVETYPE Type,
INT BaseVertexIndex,
UINT MinIndex,
UINT NumVertices,
UINT StartIndex,
UINT PrimitiveCount
);
```

ตัวอย่าง

```
_device->DrawIndexedPrimitive(D3DPT_TRIANGLELIST, 0, 0, 8, 0, 12);
```

ซึ่งโปรแกรมเมอร์จะเลือกใช้ฟังก์ชันใดก็ได้ ขึ้นอยู่กับความเหมาะสม

นอกจากนี้ การเรียกใช้ฟังก์ชันในการวาดภาพทั้ง DrawPrimitive หรือ DrawIndexedPrimitive จะต้องเรียกใช้ระหว่างฟังก์ชัน BeginScene และ EndScene ดังแสดงในโค้ดด้านล่าง

```
device->BeginScene();
        device->DrawPrimitive(...);
_device->EndScene();
```

3.3.5 D3DX Geometric Objects

ในไลบรารีของ D3DX จะมีรูปทรงมาตรฐานที่สามารถเรียกใช้ได้เลยอยู่ด้วยกัน 6 ฟังก์ชัน ดังนี้

D3DXCreateBox

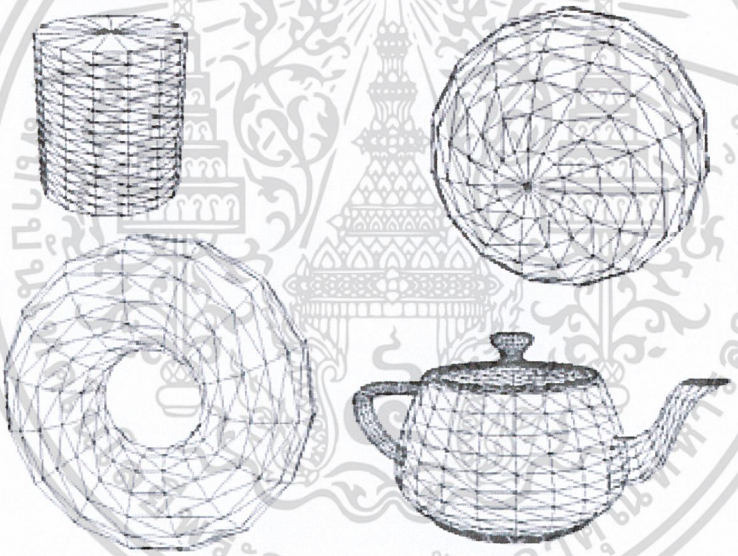
D3DXCreateSphere

D3DXCreateCylinder

D3DXCreateTeapot

D3DXCreatePolygon

D3DXCreateTorus



*รูปที่3-4 ออปเจคที่ถูกสร้างขึ้นและวาดโดยการใช้ฟังก์ชัน D3DXCreate**

โดยรูปทรงมาตรฐานทั้งหมดนี้จะมีโครงสร้างเป็นแบบ ID3DXMesh โดยวิธีการเรียกใช้ฟังก์ชันเหล่านี้ แสดงได้ดังตัวอย่างด้านล่าง

```
HRESULT D3DXCreateTeapot(
        LPDIRECT3DDEVICE9 pDevice, // device associated with the mesh
        LPD3DXMESH* ppMesh, // pointer to receive mesh
        LPD3DXBUFFER* ppAdjacency // set to zero for now
);
```

ตัวอย่างการใช้ฟังก์ชันในการสร้างกาน้ำ

```
ID3DXMesh* mesh = 0;
D3DXCreateTeapot(_device, &mesh, 0);
```

```
_device->BeginScene();
        mesh->DrawSubset(0);
_device->EndScene();
```

เมื่อสร้างเสร็จแล้วก็ต้องมีการ release ค่า

```
_mesh->Release();
_mesh = 0;
```

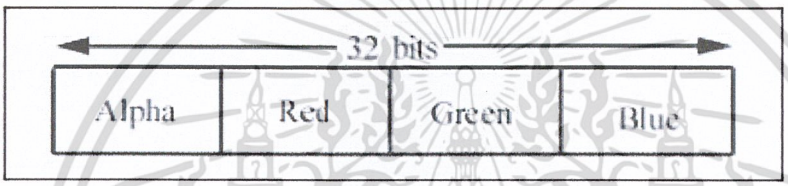
บทที่ 4
วิธีการเกี่ยวกับ Direct 3D ที่นำมาใช้

4.1 สี (Color)

4.1.1 การใส่สี (Color Representation)

ในไดเร็กซ์ทรีดี นั้นสีจะถูกกำหนดในรูปแบบของ RGB ซึ่งจะใช้สีสามสีคือ แดง เขียว น้ำเงิน ในการกำหนดสีต่างๆ โดยใช้การผสมกันของสีทั้งสามสีซึ่งจะทำให้ได้สีออกมามากมาย

ในการใส่สีในไดเร็กซ์ทรีดีนั้นเราสามารถทำได้ 2 วิธีคือ หนึ่งใช้ D3DCOLOR มี 32 บิต ซึ่งแบ่งออกเป็นสีส่วน ส่วนละ 8 บิต ตามรูปภาพ



รูปที่4-1 แสดงการแบ่ง 32-bit color

โดยค่า alpha จะเป็นการแสดงวาสว่าง ความมืดของแสง

ในการกำหนดสีนั้นเราสามารถกำหนดได้อยู่ในช่วง 0-255 โดยมีการใช้ดังนี้

```
D3DCOLOR brightRed = D3DCOLOR_ARGB(255, 255, 0, 0);
D3DCOLOR someColor = D3DCOLOR_ARGB(255, 144, 87, 201);
```

หรืออีกทางหนึ่งเราสามารถใช้ marco D3DCOLOR_XRGB แทนได้

```
#define D3DCOLOR_XRGB(r,g,b) D3DCOLOR_ARGB(0xff,r,g,b)
```

วิธีการใส่สีในไดเร็กซ์ทรีดีอีกวิธีหนึ่งคือการใช้ D3DCOLORVALUE สตรัคเจอร์โดยวิธีนี้เป็นการกำหนดค่าให้ของแม่สีเป็นจุดทศนิยม โดยวิธีนี้สามารถใส่ค่าของแม่สีในช่วง 0 – 1 โดยมีการกำหนดดังข้างล่างต่อไปนี้

```
typedef struct _D3DCOLORVALUE {
        float r; // the red component, range 0.0-1.0
        float g; // the green component, range 0.0-1.0
        float b; // the blue component, range 0.0-1.0
        float a; // the alpha component, range 0.0-1.0
} D3DCOLORVALUE;
```

หรือไม่ก็เราสามารถใช้ D3DXCOLOR สตรัคเจอร์ ซึ่งบรรจุสมาชิกข้อมูลที่เหมือนกันเป็น D3DCOLORVALUE แต่มีฟังก์ชันต่าง ๆ เพิ่มขึ้นมาเพื่อช่วยให้การใช้สีง่ายขึ้น

4.1.2 สีของเวอร์เท็กซ์ (Vertex Colors)

สีของ primitive ต่าง ๆ เช่น รูปสามเหลี่ยมจะถูกกำหนดโดยสีของเวอร์เท็กซ์ ดังนั้น ในการสร้างสีให้กับวัตถุสามมิติใด ๆ จะต้องกำหนดสีให้กับเวอร์เท็กซ์ของวัตถุนั้น ๆ เสียก่อน ดังนี้

```
struct ColorVertex
{
        float _x, _y, _z;
        D3DCOLOR _color;
        static const DWORD FVF;
};
const DWORD ColorVertex::FVF = D3DFVF_XYZ | D3DFVF_DIFFUSE;
```

4.1.3 การเกลี่ยสี (Shading)

Shading จะอยู่ในขั้นตอนของการทำ Rasterization ในไปป์ไลน์ โดยจะเป็นการนำสีของเวอร์เท็กซ์มาคำนวณหาสีของ primitive เพื่อนำไปหาสีของพิกเซลที่จะแสดงบนหน้าจอ

ไดเร็กซ์สามดีจะมีการทำ Shading อยู่ 2 แบบ คือ Flat Shading และ Gouraud Shading Flat Shading จะใช้สีของเวอร์เท็กซ์แรกของรูปสามเหลี่ยมในการคำนวณหาสีของสามเหลี่ยมนั้น โดยไม่สนใจสีของเวอร์เท็กซ์อีก 2 เวอร์เท็กซ์ที่เหลือ ดังนี้

```
ColorVertex t[3];
t[0]._color = D3DCOLOR_XRGB(255, 0, 0);
t[1]._color = D3DCOLOR_XRGB(0, 255, 0);
t[2]._color = D3DCOLOR_XRGB(0, 0, 255);
```

4.2 แสง (Lighting)

4.2.1 ส่วนประกอบของแสง

โดยส่วนประกอบของแสงแบบได้เป็น 3 ส่วน คือ

- **Ambient Light** Light เป็นส่วนประกอบของแสงที่ให้ความสว่างกับฉากทั้งหมดอย่างเท่าเทียมกัน โดยไม่ขึ้นกับระยะห่างจากแหล่งกำเนิดแสงและไม่ขึ้นกับทิศทางการหันหน้าของวัตถุที่ทำกับแหล่งกำเนิดแสง
- **Diffuse Light** เป็นส่วนประกอบของแสงที่เมื่อตกกระทบวัตถุแล้วจะสะท้อนออกไปทุกทิศทุกทาง ดังนั้นแสงแบบนี้จะขึ้นกับทิศทางของแสงและลักษณะของวัตถุเท่านั้น ไม่ขึ้นกับตำแหน่งของกล้อง
- **Specular Light** เป็นส่วนประกอบของแสงที่เมื่อตกกระทบวัตถุแล้วจะสะท้อนออกไปในทิศทางใดทิศทางหนึ่งเท่านั้น โดยแสงแบบนี้จะทำให้เห็นแสงสว่างได้ในบางมุม ดังนั้นจะต้องนำทั้งตำแหน่งของกล้อง ทิศทางของแสง และลักษณะของวัตถุมาคำนวณ

ในการกำหนดค่าสีให้กับแสงแต่ละแบบ จะใช้โครงสร้างของ D3DCOLORVALUE หรือ D3DXCOLOR ก็ได้ ดังนี้

```
D3DXCOLOR redAmbient(1.0f, 0.0f, 0.0f, 1.0f);
D3DXCOLOR blueDiffuse(0.0f, 0.0f, 1.0f, 1.0f);
D3DXCOLOR whiteSpecular(1.0f, 1.0f, 1.0f, 1.0f);
```

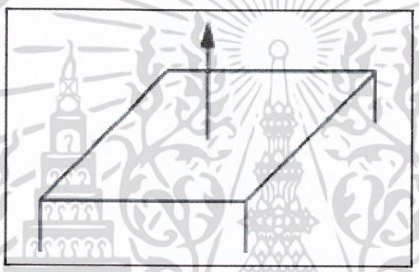
4.2.2 ชนิดของวัตถุ (Materials)

สีของวัตถุที่เรามองเห็นในโลกความเป็นจริงนั้น เกิดจากการที่แสงส่องไปที่วัตถุแล้วสะท้อนกลับมาที่ตาเรา โดยในไดเร็กซ์ทรีดีนั้นก็ใช้การทำงานในลักษณะเดียวกัน โดยจะมีการกำหนดชนิดของวัตถุ(Materials) ขึ้นมา ซึ่งชนิดของวัตถุนี้จะเป็นตัวกำหนดว่าแสงชนิดไหนจะถูกสะท้อนออกมาในลักษณะใด โดยจะกำหนดชนิดของวัตถุด้วย D3DMATERIAL9 โดยจะมีการกำหนดดังนี้

```
typedef struct _D3DMATERIAL9 {
        D3DCOLORVALUE Diffuse, Ambient, Specular, Emissive;
        float Power;
} D3DMATERIAL9;
```

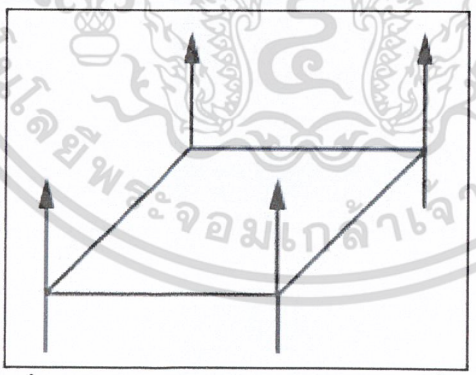
4.2.3 เส้นปกติของเวอร์เท็กซ์ (Vertex Normal)

เส้นปกติของพื้นผิว (face normal) จะเป็นตัวบอกทิศทางที่พื้นผิวหันหน้าอยู่ ดังรูป



รูปที่4-2 แสดงเส้นปกติของพื้นผิววัตถุ

สำหรับเส้นปกติของเวอร์เท็กซ์ (vertex normal) นั้นจะกำหนดเส้นปกติที่แต่ละเวอร์เท็กซ์แทนการกำหนดเส้นปกติของพื้นผิว ดังข้างล่าง



รูปที่4-3 แสดงเส้นปกติของเวอร์เท็กซ์ของพื้นผิว

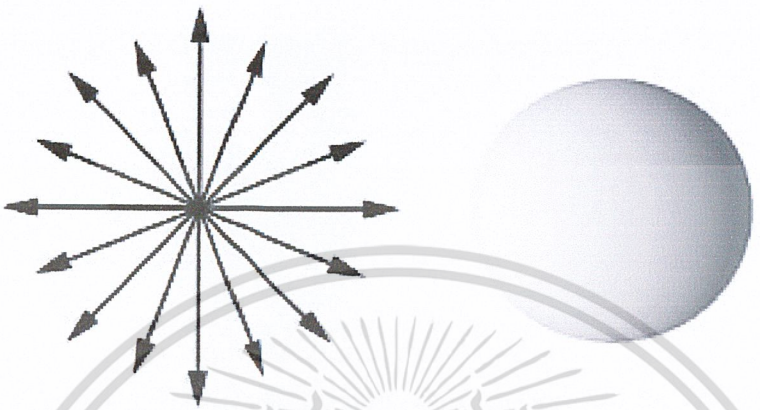
เนื่องจากไดเร็กซ์ทรีดีจะคำนวณมุมที่แสงตกกระทบพื้นผิวที่ทุก ๆ เวอร์เท็กซ์ของรูป ดังนั้นจึงต้องทราบเส้นปกติของเวอร์เท็กซ์เพื่อนำมาใช้ในการคำนวณ ซึ่งตัวอย่างของโครงสร้างของเวอร์เท็กซ์ที่มีการนำเส้นปกติของเวอร์เท็กซ์มาใช้ จะแสดงได้ดังโค้ดด้านล่าง

```
struct Vertex
{
        float _x, _y, _z;
        float _nx, _ny, _nz;
        static const DWORD FVF;
};
const DWORD Vertex::FVF = D3DFVF_XYZ | D3DFVF_NORMAL;
```

4.2.4 แหล่งกำเนิดแสง (Light Source)

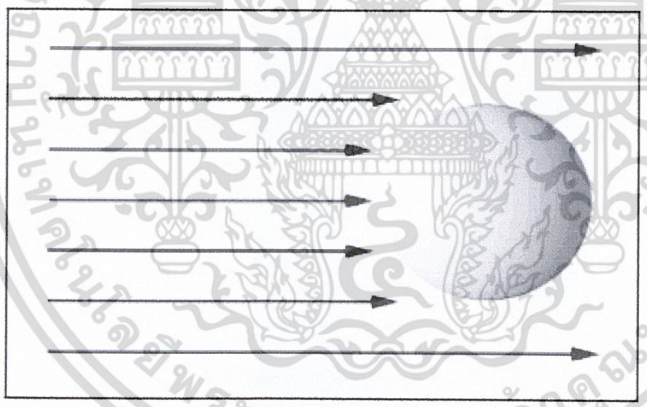
ในไดเร็กซ์ทรีดีสามารถใช้แหล่งกำเนิดแสงได้ 3 แบบคือ

- **Point lights** เป็นแหล่งกำเนิดแสงที่ต้องระบุตำแหน่ง แหล่งกำเนิดแสงแบบจุดนี้จะส่องแสงออกมาทุกทิศทุกทางดังรูป



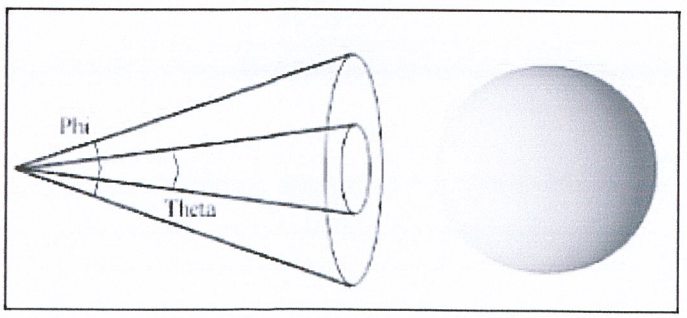
รูปที่4-4แหล่งกำเนิดแสงแบบpoint light

- **Directional Light** เป็นแหล่งกำเนิดแสงที่ไม่มีตำแหน่ง และส่องแสงเป็นลำแสงขนานในทิศทางใดทิศทางหนึ่งตามที่กำหนด



รูปที่ 4-5 แหล่งกำเนิดแสงแบบDirectional light

- **Sport lights** เป็นแหล่งกำเนิดแสงที่มีตำแหน่งและส่องแสงออกมาเป็นรูปกรวยในทิศทางที่กำหนด โดยกรวยแสงจะถูกระบุด้วยค่า 2 ค่า ได้แก่ มุม θ (theta) และมุม φ (phi) โดยมุม θ จะใช้เป็นมุมของกรวยใน และมุม φ จะเป็นมุมของกรวยนอก



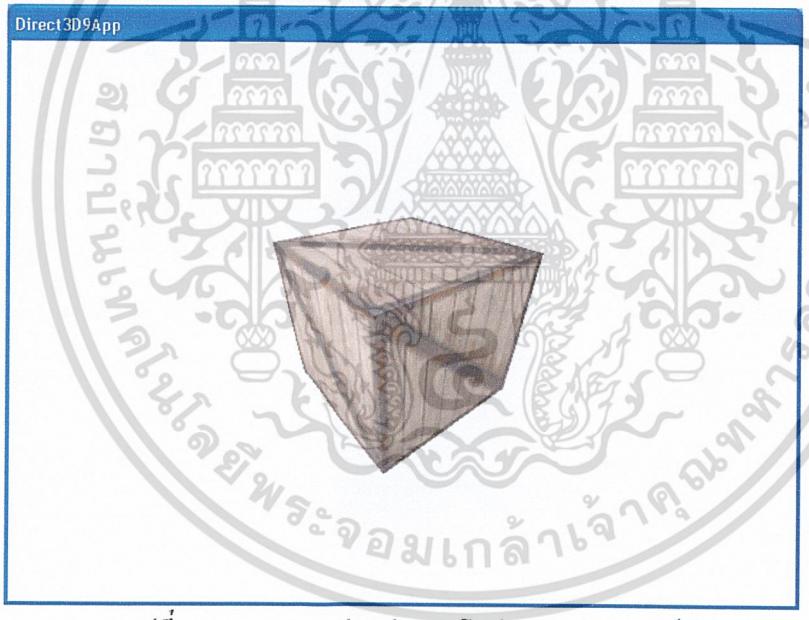
รูปที่ 4-6 แหล่งกำเนิดแสงแบบSport light

ซึ่งในไดเร็กซ์ทรีดีแหล่งกำเนิดแสงทั้ง 3 แบบ จะถูกแสดงได้ด้วย D3DLIGHT9 สตรัคเจอร์ดังนี้

```
typedef struct _D3DLIGHT9 {
        D3DLIGHTTYPE Type;
        D3DCOLORVALUE Diffuse;
        D3DCOLORVALUE Specular;
        D3DCOLORVALUE Ambient;
        D3DVECTOR Position;
        D3DVECTOR Direction;
        float Range;
        float Falloff;
        float Attenuation0;
        float Attenuation1;
        float Attenuation2;
        float Theta;
        float Phi;
} D3DLIGHT9;
```

4.3 การใส่รูปภาพบนพื้นผิววัตถุ (Texture Mapping)

การแปะรูปภาพคือการนำรูปภาพแปะลงบนโพลีกอน เพื่อเป็นการเพิ่มรายละเอียดแลดูความสมจริงในกับวัตถุ เช่นการทำการแปะรูปภาพลงบนลูกบาศก์ ทำให้กลายเป็นลังไม้ที่ดูสมจริง

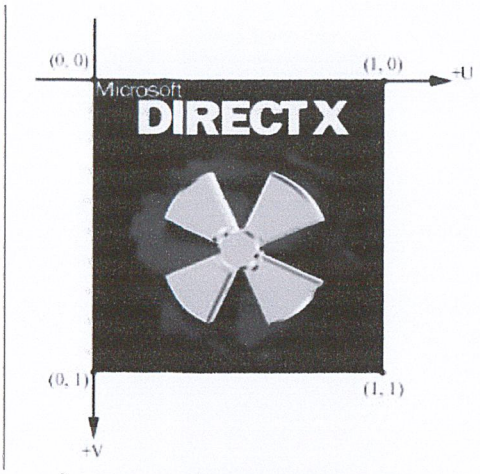


รูปที่ 4-7 แสดงการแปะรูปลงบนโพลีกอนรูปลูกบาศก์

ในไดเร็กซ์ทรีดี รูปภาพจะที่นำไปแปะจะถูกแสดงด้วย IDirect3DTexture9 อินเตอร์เฟส

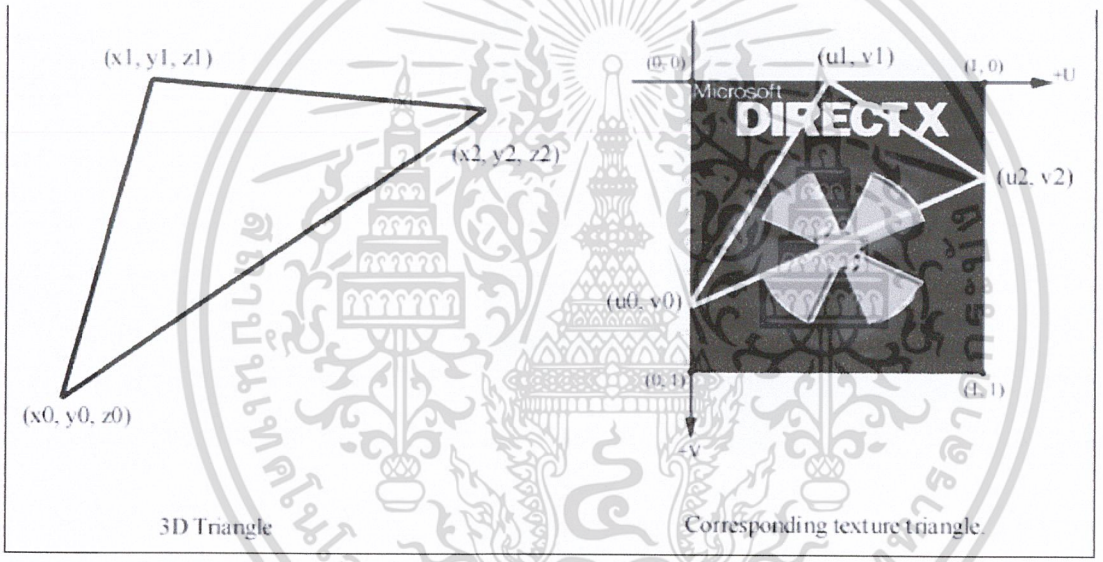
4.3.1 เท็กซ์เจอร์โคออดิเนต (texture Coordinate)

ในไดเร็กซ์ทรีดีใช้ระบบเท็กซ์เจอร์ออดิเนตซึ่งประกอบไปด้วย แกน U และแกน V โดยจุดใน U และ V นั้นเป็นการบอกอีลีเมนต์บนเท็กซ์เจอร์ซึ่งเราเรียกว่า เท็กซ์เซล (Texel) โดยมีลักษณะดังรูป



รูปที่4-8แสดงระบบของเท็กซ์เจอร์โคออดิเนต

สำหรับรูป สามเหลี่ยมสามมิติ เราต้องกำหนดสามเหลี่ยมบนเท็กซ์เจอร์ ที่เราต้องการจะแปะมันลงไป ดังรูปข้างล่าง



รูปที่4-9แสดงการแปะเท็กซ์เจอร์ลงบนวัตถุ 3 มิติ

โดยสามารถทำได้ตามคำสั่งข้างล่างนี้

```
struct Vertex
{
        float _x, _y, _z;
        float _nx, _ny, _nz;
        float _u, _v; // texture coordinates
        static const DWORD FVF;
};
const DWORD Vertex::FVF = D3DFVF_XYZ | D3DFVF_NORMAL | D3DFVF_TEX1;
```

ซึ่งในขณะนี้สามเหลี่ยมทุกๆรูป สร้างจาก สามเวอร์เท็กซ์ ซึ่งกำหนดตรงกับเท็กซ์เจอร์สามเหลี่ยมจากเท็กซ์เจอร์โคออดิเนต

4.3.2 การสร้างและการนำเท็กซ์เจอร์ไปใช้งาน

ข้อมูลเท็กซ์เจอร์ โดยปกติเราจะอ่านมาจากอิมเมจไฟล์ซึ่งถูกเก็บอยู่บนดิสก์เราต้องทำการโหลดมันเข้าไปเป็น IDirect3DTexture9 ออปเจ็ก ในการทำเช่นนี้เราสามารถใช้ D3DX ฟังก์ชัน ตามนี้

```
HRESULT D3DXCreateTextureFromFile(
        LPDIRECT3DDEVICE9 pDevice, // device to create the texture
        LPCSTR pSrcFile, // filename of image to load
        LPDIRECT3DTEXTURE9* ppTexture // ptr to receive the created texture
);
```

4.3.3 การกรอง (filtering)

จากวิธีที่ผ่านมาเท็กซ์เจอร์สามเหลี่ยมจะมีขนาดไม่เท่ากับ ฉากสามเหลี่ยม เมื่อ เท็กซ์เจอร์สามเหลี่ยมมีขนาดเล็กกว่าฉากสามเหลี่ยม เท็กซ์เจอร์สามเหลี่ยมจะถูกเรียกเป็น “magnified to fit” แต่ถ้าเท็กซ์เจอร์สามเหลี่ยมใหญ่กว่าฉากสามเหลี่ยม เท็กซ์เจอร์จะถูกเรียกเป็น “minified to fit” ในทั้ง 2 กรณี จะเกิดการ distortion ในไดเร็กซ์ทรีดี เราจะใช้วิธีการที่เรียกว่า การกรอง (filtering) เพื่อเป็นการช่วยให้ distortion ดูเรียบขึ้น

ในไดเร็กซ์ทรีดีจะ มีวิธีการกรองอยู่ 3 ชนิดซึ่งแต่ละอันจะมีระดับคุณภาพต่างกัน

- **Nearest point filtering** เป็นการกรองที่ตั้งไว้แรกเริ่มซึ่งจะจัดการได้เฉพาะผลที่ผิดพรากมากๆ แต่ให้การประมวลผลที่รวดเร็ว วิธีการใช้ดังโค้ดข้างล่าง

```
Device->SetSamplerState(0, D3DSAMP_MAGFILTER, D3DTEXF_POINT);
Device->SetSamplerState(0, D3DSAMP_MINFILTER, D3DTEXF_POINT);
```

- **Linear filtering** เป็นการกรองที่ให้ผลค่อนข้างดีและสามารถทำได้อย่างรวดเร็วแล้วในฮาร์ดแวร์ขณะนี้ วิธีการใช้ดังโค้ดข้างล่าง

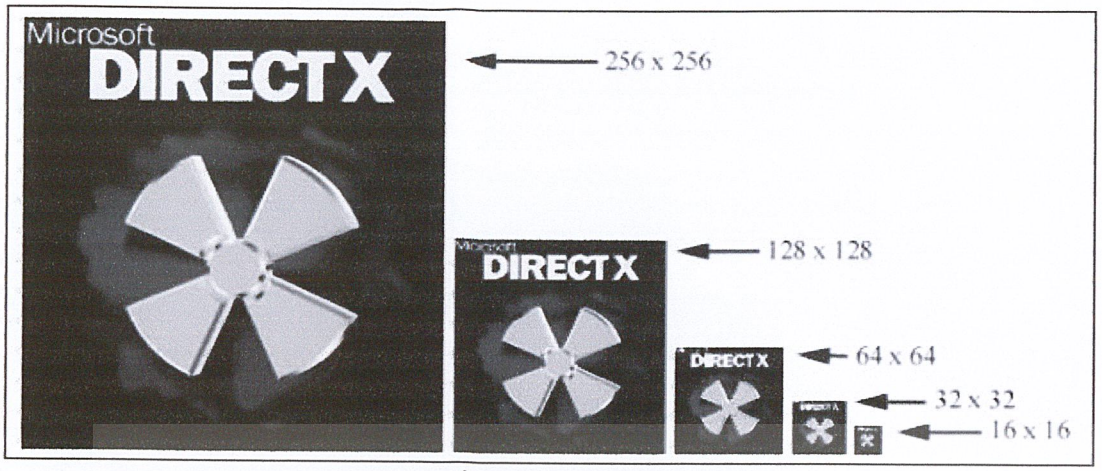
```
Device->SetSamplerState(0, D3DSAMP_MAGFILTER, D3DTEXF_LINEAR);
Device->SetSamplerState(0, D3DSAMP_MINFILTER, D3DTEXF_LINEAR);
```

- **Anisotropic filtering** เป็นการกรองที่ให้ผลได้ดีที่สุดแต่ก็ใช้ระยะเวลาในการคำนวณนานที่สุด วิธีการใช้ดังโค้ดข้างล่าง

```
Device->SetSamplerState(0, D3DSAMP_MAGFILTER, D3DTEXF_ANISOTROPIC);
Device->SetSamplerState(0, D3DSAMP_MINFILTER, D3DTEXF_ANISOTROPIC);
```

4.3.4 การทำชุดของ texture (Mipmaps)

ในการทำ texturing นั้น สามารถสร้างชุดของ texture ไว้ก่อนได้ ซึ่งภายในชุดนั้น จะมี texture ที่มีขนาดแตกต่างกัน โดย texture ที่มีขนาดเล็กกว่าจะคำนวณได้มาจาก texture ที่มีขนาดใหญ่กว่า และจะเรียก texture แต่ละขนาดที่อยู่ภายในชุดนั้นว่า mipmap โดยรูปที่ 22 จะแสดงการตัวอย่างของการทำชุดของ mipmap



รูปที่4-9 texture mipmap

สำหรับวิธีการใช้ mipmap นั้นจะต้องกำหนดค่า mipmap filter ด้วย ซึ่งสามารถทำได้ดังนี้

```
Device->SetSamplerState(0, D3DSAMP_MIPFILTER, Filter);
```

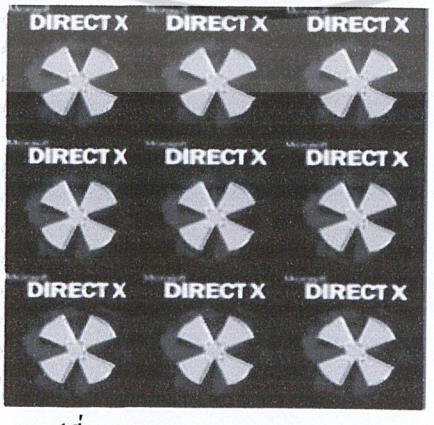
ซึ่งค่า Filter ในโค้ดด้านบนนั้น จะมีค่าได้ 3 ค่า คือ

- D3DTEXF_NONE คือ ไม่มีการใช้ mipmap
- D3DTEXF_POINT คือ ใช้ mipmap ที่มีขนาดใกล้เคียงกับวัตถุมากที่สุด
- D3DTEXF_LINEAR คือ ใช้ mipmap 2 รูปที่มีขนาดใกล้เคียงกับวัตถุมาทำการเฉลี่ยเพื่อหาค่าขนาดของ texture ที่จะนำไปใช้จริง

โหมดการทำ texturing (Address Mode)

โหมดการทำ texturing จะถูกใช้ในกรณีที่มีการกำหนดค่าของแกน u และแกน v ให้กับเวอร์เท็กซ์ของวัตถุมากกว่า 1 หรือน้อยกว่า 0 โดยโหมดการทำ texturing จะมีอยู่ด้วยกัน 4 แบบ ได้แก่

- **โหมด Wrap** เป็นโหมดที่จะได้ภาพเดิมซ้ำ ๆ กันถ้ามีการกำหนดค่าแกน u และแกน v เกินช่วง [0,1] โดยรูปที่ 4-10 จะแสดงภาพที่ได้จากการทำงานในโหมดนี้ โดยกำหนดค่าแกน u และแกน v ของเวอร์เท็กซ์ของสี่เหลี่ยมเป็น (0,0) (0,3) (3,0) และ (3,3)



รูปที่ 4-10 แสดงการทำ wrap mode

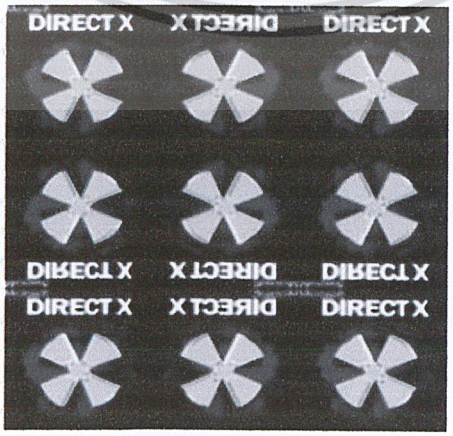
- โหมด **Border Color** เป็นโหมดที่จะได้ภาพในส่วนที่เกินจากค่าแกน u และแกน v ในช่วง [0,1] เป็นดังรูปที่ 4-11

รูปที่4-11 แสดงการทำ BorderColor

- โหมด **Clamp** เป็นโหมดที่จะได้ภาพในส่วนที่เกินจากค่าแกน u และแกน v ในช่วง [0,1] เป็นดังรูปที่ 4-12

รูปที่ 4-12แสดงการทำ Clamp

- โหมด **Mirror** เป็นโหมดที่จะได้ภาพในส่วนที่เกินจากค่าแกน u และแกน v ในช่วง [0,1] เป็นภาพกลับจาก texture เดิมในลักษณะคล้าย ๆ ภาพที่เกิดจากการมองกระจก ดังรูปที่ 4-13



รูปที่ 4-13 แสดงการทำ Mirror

บทที่ 5
โปรแกรมสำเร็จรูปในการสร้างวัตถุ

5.1 3D Studio Max

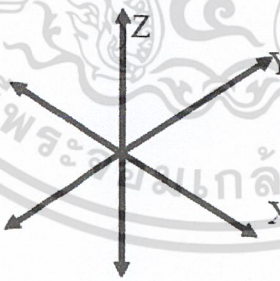
5.1.1 ความรู้เบื้องต้นเกี่ยวกับทรีดีแม็กซ์

โปรแกรมทรีดีแม็กซ์ เป็นโปรแกรมสร้างภาพ 3 มิติแบบโมเดล 3 มิติของค่ายดิสครีท (discreet)ซึ่งในเวอร์ชันแรก ๆ นั้น โปรแกรมใช้ทรัพยากรเครื่องที่สูงมากและต้องใช้บนวินโดส์เอ็นที (WindowNT) เท่านั้น ทำให้ผู้ใช้งานส่วนใหญ่มักจะเป็นบริษัทที่มีเงินทุนสูงเท่านั้น แต่ในปัจจุบันทางผู้ผลิตได้พัฒนาโปรแกรมทำให้สามารถใช้งานได้กับคอมพิวเตอร์ตามบ้านได้ จึงมีผู้ใช้งานเป็นจำนวนมาก อีกสาเหตุหนึ่งที่มีผู้ใช้งานเป็นจำนวนมาก เนื่องจากโปรแกรมทรีดีแม็กซ์นี้ มีความสามารถอย่างมากในการสร้างวัตถุ 3 มิติที่มีความซับซ้อนได้อย่างสมบูรณ์ และสามารถสร้างงานภาพเคลื่อนไหว (Animation) ในระดับสูงได้ อีกด้วย

5.1.2 ความรู้พื้นฐานเกี่ยวกับโลก 3 มิติของโปรแกรมทรีดีแม็กซ์

5.1.2.1 แกนอ้างอิงในการสร้างวัตถุ 3 มิติ

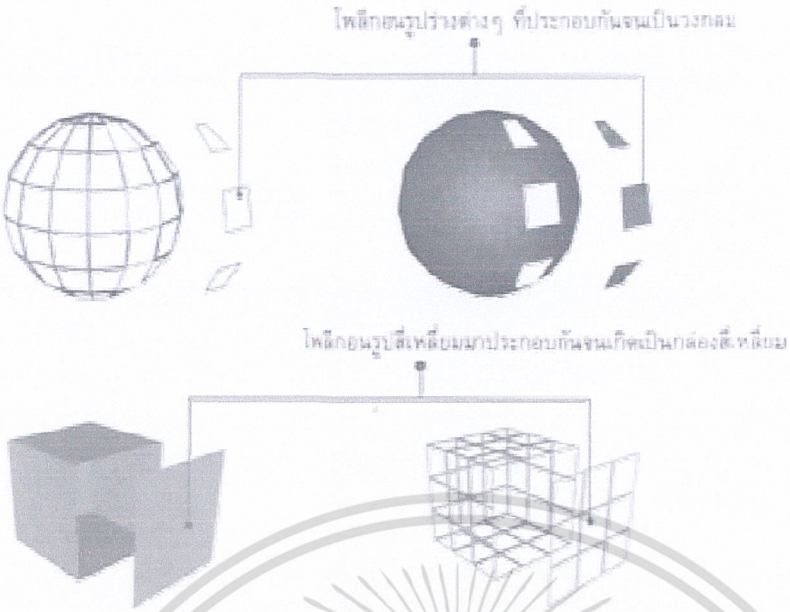
โปรแกรมทรีดีสตูดิโอ แม็กซ์ ใช้ระบบคาร์ทีเซียน (Cartesian) ซึ่งอ้างอิงจุด 3 จุดในแนวแกน X, Y และ Z โดยจุดเริ่มต้นจะมีพิกัด (0,0,0) เราสามารถทราบตำแหน่งของวัตถุ 3 มิติได้โดยวัดกับจุดนี้



รูปที่ 5-1 ระบบพิกัด 3 มิติ

5.1.2.2 โพลีกอน (Polygon)

โพลีกอนคือ แผ่นระนาบ 2 มิติที่มีหลายเหลี่ยม และเป็นชิ้นส่วนที่โปรแกรมนำมาต่อกันจนเกิดเป็นรูปร่างต่าง ๆ ของวัตถุที่เราต้องการสร้าง ดังนั้นถ้าเราใช้จำนวนโพลีกอนมาก จะได้ภาพที่ละเอียด เสียเวลาคำนวณมาก ถ้าเราใช้จำนวนโพลีกอนน้อย จะได้ภาพความละเอียดน้อย ประหยัดเวลาคำนวณ



รูปที่ 5-2 แสดงการนำโพลีกอนมาต่อกันจนเป็นวัตถุแบบต่าง ๆ

รูปที่ 5-3 ภาพ A ใช้โพลีกอนน้อย ส่วนภาพ B ใช้โพลีกอนมาก

5.1.3 ความสามารถในการทำงานสามมิติของโปรแกรม 3D Studio Max

ความสามารถในการสร้างวัตถุรูปทรงต่าง ๆ โปรแกรม ทรีดีสตูดิโอ แม็กซ์ ได้พัฒนาชุดคำสั่งและเครื่องมือในการสร้างวัตถุรูปแบบต่าง ๆ ขึ้นมาให้เลือกใช้งานมากมาย ไม่ว่าจะเป็นเครื่องมือในการสร้างวัตถุรูปทรงเรขาคณิต หรือการสร้างวัตถุที่มีรูปร่างซับซ้อน เช่น ตัวการ์ตูน ฯลฯ

ความสามารถในการกำหนดพื้นผิวให้วัตถุ การกำหนดพื้นผิวเป็นอีกส่วนหนึ่งที่ช่วยให้วัตถุสามมิติสวยงามและเหมือนจริง โปรแกรมมีชุดคำสั่งในการกำหนดพื้นผิวมามากมาย ทำให้การกำหนดพื้นผิวให้กับวัตถุเป็นเรื่องง่าย

ความสามารถในการเรนเดอร์ (Render) การเรนเดอร์คือ การประมวลผลภาพของวัตถุที่สร้างขึ้นมาทั้งหมด ทำให้เราเห็นรายละเอียดของวัตถุได้ทั้งพื้นผิว แสงและเงาการเรนเดอร์ภาพในโปรแกรมทรีดีสตูดิโอ แม็กซ์ ได้พัฒนาขึ้นจนสามารถกำหนดองค์ประกอบต่าง ๆของภาพได้เหมือนจริงมาก ไม่ว่าจะ

เป็นเรื่องความละเอียดของพื้นผิว แสงและเงาที่เกิดขึ้น นอกจากนี้ยังสามารถใส่เอฟเฟ็ค (Effect) เพิ่มเติมเข้าไปในขณะเรนเดอร์เช่น การใส่ประกายแสงลงบนภาพ

ความสามารถในการสร้างภาพเคลื่อนไหว ทรีดีสตูดิโอ แม็กซ์ได้นำคำสั่งต่าง ๆ ในการกำหนดการเคลื่อนไหวให้กับวัตถุที่ต้องการมาให้เราใช้งานได้อย่างสะดวกสบายมากขึ้น เช่น กำหนดให้วัตถุเคลื่อนไหวตามแรงดึงดูดของโลก หรือคำนวณทิศทางของวัตถุที่เกิดจากการชนกัน เพียงแค่เรากำหนดทิศทางที่ต้องการให้เกิดแรงดึงดูดและแรงในการชนเท่านั้น ที่เหลือโปรแกรมจะคำนวณให้โดยอัตโนมัติ

5.2 Poser

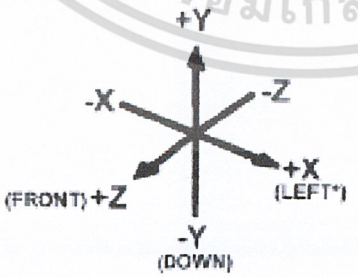
5.2.1 ความรู้เบื้องต้นเกี่ยวกับ Poser

Poser เป็นโปรแกรมที่ช่วยในการสร้างและออกแบบ 3D character และสามารถสร้างการเคลื่อนไหวให้กับ 3D character นั้น ผลิตโดยบริษัท Curious Labs ซึ่งโปรแกรม poser ได้ถูกออกแบบมาเพื่อให้สามารถสร้าง 3D Human ได้อย่างง่ายดายและสะดวกมาก โดยปัจจุบันโปรแกรม Poser ได้พัฒนามาถึง version 5 ซึ่งมีความสามารถในการนำรูปใบหน้าที่เป็น texture มาแปะ ลงในหุ่นจำลอง 3 มิติที่สร้างขึ้นและสามารถสร้างผมและเสื้อผ้าแบบ dynamic ได้อีกด้วย

5.2.2 ความรู้พื้นฐานเกี่ยวกับ Poser

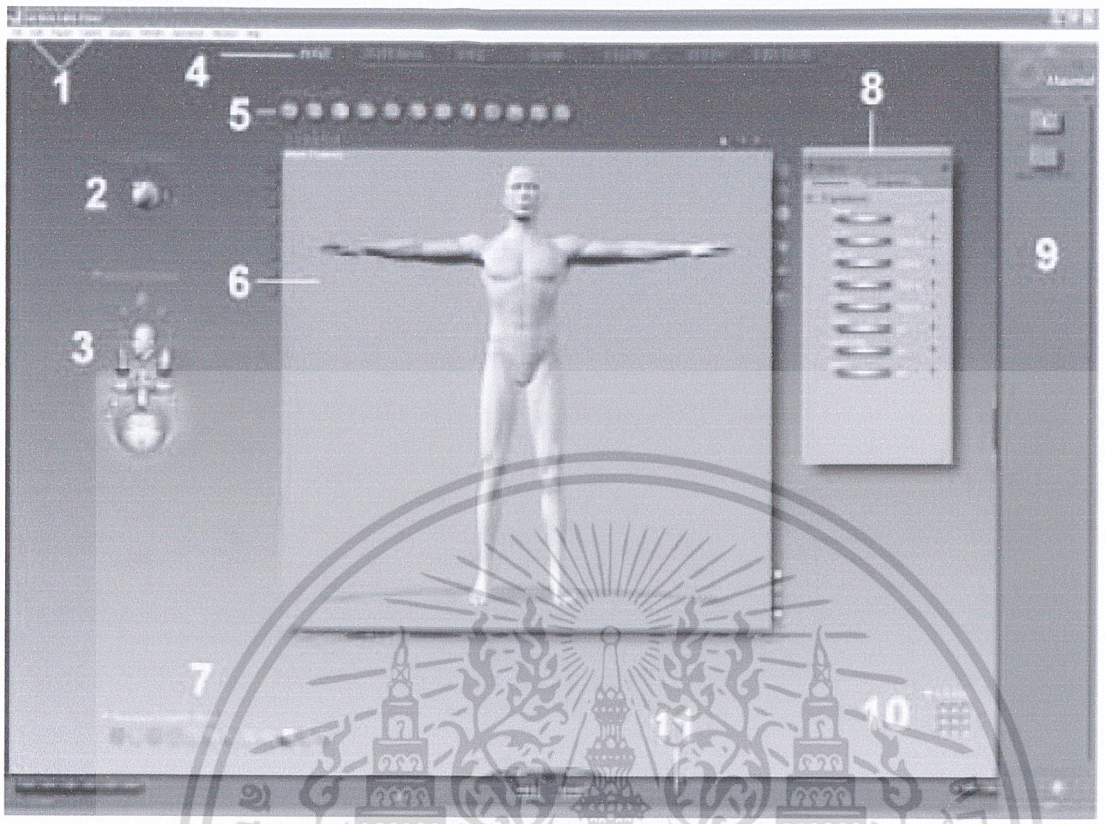
5.2.2.1 แกนอ้างอิงในการสร้างวัตถุ 3 มิติ

โปรแกรมทรีดีสตูดิโอ แม็กซ์ ใช้ระบบคาร์ทีเซียน (Cartesian) ซึ่งอ้างอิงจุด 3 จุดในแนวแกน X, Y และ Z โดยจุดเริ่มต้นจะมีพิกัด (0,0,0) เราสามารถทราบตำแหน่งของวัตถุ 3 มิติได้โดยวัดกับจุดนี้



รูปที่ 5-4 ระบบพิกัด 3 มิติในโพสเซอร์

5.2.2.2 การใช้งาน Poser



รูปที่ 5-5 แสดงหน้าจอหลักของโปรแกรม poser

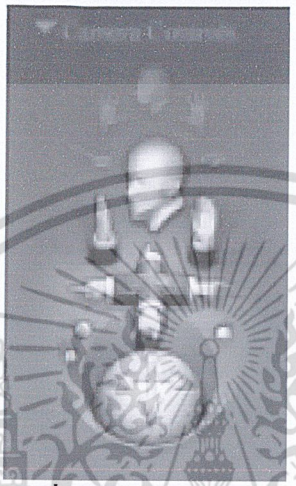
จากรูปเป็นการแสดง Poser Room ซึ่งเปรียบเสมือนหน้าจอหลักของโปรแกรม poser โดยมีส่วนควบคุมต่างๆดังนี้

1. **Menu** bar
2. **Light** control
3. **Camera** control
4. **Room** tabs
5. **Editing** tool
6. **Document** window
7. **Display** control
8. **Properties & Parameters** palettes
9. **Library** palettes
10. **Memory** dots
11. **Animation** controls

การเลือกมุมกล้อง

ใช้เครื่องมือ Camera Controls ช่วยทำให้สามารถที่จะเลื่อนกล้องไปในตำแหน่งต่างๆ ทุกทิศทางในสตูดิโอได้อย่างรวดเร็ว เพื่อที่จะวางมุมมองในการจัดท่าของหุ่น

ตำแหน่งกล้องที่จับภาพในแต่ละมุม สามารถปรับเปลี่ยนได้ มีดังนี้ Main, Auxiliary, Left, Right, Top, Bottom, Font, Back, Right Hand, Left Hand, Face, Posing, Dolly และ Flyaround View



รูปที่ 5-5 Camera controls

5.2.2.3 การปรับรูปร่างและขนาดของหุ่น

ขนาดและรูปร่างของหุ่นจำลอง สามารถปรับเปลี่ยนได้ เช่น สร้างท้องให้ใหญ่ขึ้น ดึงหัวให้กว้างขึ้น โดยใช้เครื่องมือ shaping และ parameter dials

<u>การใช้เครื่องมือ shaping</u>

วิธีที่ง่ายที่สุดก็คือ การเลือกเครื่องมือ Scale หรือเครื่องมือ Taper คลิกบนส่วนที่ต้องการจะปรับ และลากหากต้องการการควบคุมการปรับเปลี่ยนที่ถูกต้องแน่นอนใน 3 มิติ ให้ใช้ Parameter Dials

<u>การใช้ Parameter dials</u>

เมื่อทำการคลิกไปยังส่วนที่ต้องการจะปรับขนาด ค่าใน Parameter dials จะเปลี่ยนเพื่อที่จะแสดงค่าของส่วนนั้นให้ควบคุม

- **Taper** การปรับ taper จะเป็นการเปลี่ยน scale ที่ปลายด้านหนึ่งเท่านั้น อีกด้านจะไม่เปลี่ยน scale โดยหากลากไปทางด้านขวา จะเป็นการเพิ่ม Taper ที่ปลายของส่วนที่อยู่ไกลจากศูนย์กลางของร่างกายทำให้หดลง แต่ถ้าลากไปทางด้านซ้ายก็จะเพิ่มขนาดให้ Taper
- **Scale** ใช้ปรับเปลี่ยนขนาดทั้ง 3 มิติ ตามสัดส่วน ลาก Dial ไปทางขวาเพื่อที่จะเพิ่ม Scale โดยจะมีอยู่ 3 ชนิด คือ X Scale, Y Scale, Z Scale

5.2.2.4 การ export

สามารถ Export วัตถุที่บรรจุในหน้าต่างเอกสาร เพื่อแปลงเป็น image file โดยอาจสามารถทำการบันทึกหุ่นโดยให้รูปแบบการแสดงเป็น Wireframe image

และยังสามารถ Export Geometry ที่บรรจุใน Poser File ใน Format อื่นด้วย โดยสามารถที่จะโยกย้ายถ่ายโอนสู่โปรแกรม 3D Graphic อื่นรูปแบบที่สามารถ ถ่ายโอนได้มีดังนี้ OBJ, 3D Studio, Detailer, DXF, RIB, Wavefront, 3DMF และ VRML/H-Anim

บทที่ 6
การออกแบบและพัฒนาระบบ

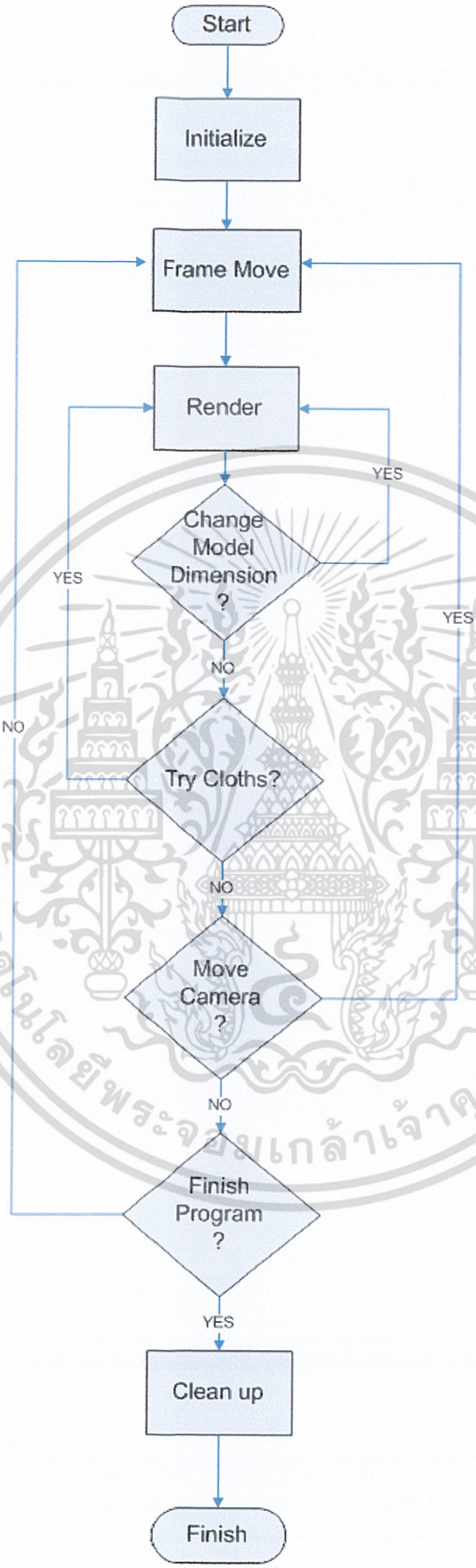
ในการออกแบบและพัฒนาระบบนั้นจะสามารถแบ่งได้เป็น 3 ส่วนคือ

1. การสร้างโปรแกรมประยุกต์ 3 มิติ
2. การสร้างหุ่นจำลอง 3 มิติ
3. การสร้างเสื้อผ้า

6.1 การสร้างโปรแกรมประยุกต์ 3 มิติ

ในส่วนของโปรแกรมประยุกต์ได้ใช้โปรแกรม Virtual Studio C++ในการเขียน และสร้างแอปพลิเคชันแบบ MFC ซึ่งทำให้โปรแกรมประยุกต์มีความสามารถในการตอบสนอง (interactive) ต่อผู้ใช้มากขึ้น และมีการเพิ่มโค้ดในส่วนของ DirectX 9.0เข้าไปทำให้ตัวโปรแกรมประยุกต์สามารถแสดงภาพส่วนของโลก 3 มิติ และแสดงปุ่มกดที่แสดงเสื้อผ้า ที่ใช้ในการลองเสื้อ และปุ่มควบคุมต่างๆ

โดยขั้นตอนการทำงานของแอปพลิเคชันนั้น สามารถแสดงได้ด้วยโฟล์วชาร์ทดังต่อไปนี้



รูปที่ 6-1 โฟล์วชาร์ทแสดงการทำงานของโปรแกรมประยุกต์ในการสร้างภาพ 3 มิติ

อธิบายการทำงานในแต่ละส่วนของโฟล์วชาร์ท

Start เป็นการเริ่มการทำงานของโปรแกรม

Initialize ใช้ในการกำหนดค่าเริ่มต้นต่างๆ โดยมีการกำหนดค่าเริ่มต้นดังนี้

- **InitInput**

  เป็นการกำหนดไดเร็กซ์อินพุทออปเจคในกับโปรแกรม

```
hr = m_pInputDeviceManager->Create( hWnd, NULL, m_diafGame,
                                    StaticInputAddDeviceCB, this )
```

- **InitDeviceObject**

  ทำการจองพื้นที่ในหน่วยความจำไว้ให้กับวัตถุทั้งที่สร้างขึ้นเองภายในโปรแกรม และที่นำเข้ามาจาก .x file

  การสร้างวัตถุในโปแกรม

  โดยมีตัวอย่างในการสร้างวัตถุในโปแกรมดังนี้

```
m_room = new Room(m_pd3dDevice);
    hr = m_room->Setup();
    if( FAILED( hr ) )
      return DXTRACE_ERR( "m_room->Setup", hr );
```

โดยคำสั่งข้างบนนั้นจะเป็นการเรียนฟังก์ชัน Setup ซึ่งเป็นการกำหนดจุดยอดต่างๆให้กับวัตถุ ที่ต้องการจะสร้าง โดยสามารถระบุรายละเอียดของเวอร์เท็กซ์ได้ เช่น กำหนดตำแหน่งของเวอร์เท็กซ์ในเวิร์ล, กำหนดเส้นปกติของเวอร์เท็กซ์, กำหนดค่าของแกน u และค่าของแกน v ที่ใช้ในการทำ texturing เป็นต้น ซึ่งในโปรแกรมประยุกต์นี้ได้ทำการสร้างวัตถุโดยโปรแกรม คือ ฉากที่ใช้กั้นทำห้องลองเสื้อ, ส่วนของเสื้อผ้าใดบางส่วน, ส่วนของหุ่นจำลอง 3 มิติในบางส่วน ซึ่งจะกล่าวในหัวข้อถัดไป

การนำวัตถุเข้ามาจาก .x file

โดยมีตัวอย่างนารสร้างวัตถุในโปรแกรมดังนี้

```
m_whead = new WHead(m_pd3dDevice);
    hr = m_whead->Setup();
    if( FAILED( hr ) )
      return DXTRACE_ERR( "m_whead->Setup", hr );
```

ซึ่งจะเห็นว่ามีการกำหนดเหมือนกันกับการสร้างวัตถุเองโดยโปรแกรม ซึ่งส่วนที่ต่างกันของทั้ง 2 แบบนี้คือ โครงสร้างภายในของ ฟังก์ชัน Setup ซึ่งจะทำการอธิบายส่วนของเรื่องการสร้างหุ่นจำลอง 3 มิติ

- **RestoreDeviceObject** เป็นขั้นตอนของการกำหนดค่าเริ่มต้นต่าง ๆ ให้กับแอพพลิเคชัน แต่จะเรียกใช้หลังจากทำฟังก์ชัน InitDeviceObjects เสร็จแล้ว ซึ่งมีการกำหนดค่าต่าง ๆ ดังนี้

<u>การกำหนดค่าเริ่มต้นในกับกล้อง</u> โดยในโปรแกรมมีการกำหนดดังนี้

```
D3DXMATRIX matView;
   D3DXVECTOR3 vEyePt = m_camera->GetEyePt();
        D3DXVECTOR3 vLookatPt = m_camera->GetLookatPt();
        D3DXVECTOR3 vUpVec = m_camera->GetUpVec();
 //-------------------------
        vEyePt[1] = 3.5;
        vEyePt[2] = -5.0;
        vLookatPt[2] = 1.0;
        m_vRightVec[0] = 1.0;
        m_vRightVec[1] = 0.0;
        m_vRightVec[2] = 0.0;
        m_camera->SetViewParams(vEyePt, vLookatPt, vUpVec);
//-------------------------
        D3DXMatrixLookAtLH( &matView, &vEyePt, &vLookatPt, &vUpVec );
   m_pd3dDevice->SetTransform( D3DTS_VIEW, &matView );
```

ในการกำหนดค่าเริ่มต้นของกล้องนั้นจะกำหนดตามค่า vEyePt โดยในโปแกรมนี้กำหนดค่าให้เป็น (0, 3.5, -5) และทำการกำหนดเวคเตอร์ทิศขวาของกล้อง(m_vRightVec)เป็น (1,0,0) จากนั้นใช้ฟังก์ชัน D3DXMatrixLookAtLH (&matView, &vEyePt, &vLookatPt, &vUpVec) ในการสร้างเมทริกซ์ matView และทำการเรียกใช้ฟังก์ชัน SetTransform(D3DTS_VIEW, &matView) เพื่อให้ matView เป็นเมทริกซ์ที่ใช้สำหรับการเปลี่ยนจากแกนของเวิร์ลมาเป็นแกนของกล้อง

<u>การกำหนดค่าเริ่มต้นในเรื่องแสง</u> โดยในโปรแกรมมีการกำหนดดังนี้

```
D3DLIGHT9 light;
   D3DUtil_InitLight( light, D3DLIGHT_DIRECTIONAL, -1.0f, -1.0f, 2.0f );
   m_pd3dDevice->SetLight( 0, &light );
   m_pd3dDevice->LightEnable( 0, TRUE );
   m_pd3dDevice->SetRenderState( D3DRS_LIGHTING, TRUE );
```

จากโปรแกรมข้างบนนั้นเป็นการกำหนดแสงแบบ Directional โดยมรจุดกำเนิดแงอยู่ที่จุด (-1.0,-1.0, 2.0) ของระบบแกนของเวิลด์

<u>Frame Move</u> เป็นส่วนที่ทำการรอรับค่าต่างๆ ที่นำมาใช้ในการเปลี่ยนแปลงหมุนกล้อง โดยในโปรแกรมประยุกต์นี้ จะเป็นมุมกล้องในลักษณะของมุมมองบุคคลที่ 1 ซึ่งจะให้ภาพเหมือนกับการ

มองเห็นจริงของผู้ใช้ ซึ่งโปรแกรมนี้ได้ใช้การเคลื่อนที่ของมุมกล้องไปมาภายในโลก 3 มิติ โดยในโปรแกรมนี้มุมกล้องสามารถเลื่อนได้ 2 ลักษณะ คือ การเคลื่อนที่เข้าออก (ไปข้างหน้าและถอยหลัง) และการเคลื่อนที่ขึ้นลง

<u>การเคลื่อนที่เข้าออก</u>

สำหรับการเคลื่อนที่ในโลก 3 มิติ นั้น ก็คือการเคลื่อนที่ในแนวระนาบกล้อง ซึ่งคือการเคลื่อนที่ในแนวระนาบแกน z ของ world space โดยแกน +z นั้นจะพุ่งเข้าหาจอ, แกน –z นั้นจะเคลื่อนที่พุ่งเข้าหาจอซึ่งการเคลื่อนที่เข้าออกนี้ควบคุมโดยคีย์บอร์ด ในส่วนของสมการการเคลื่อนที่มีรูปแบบดังนี้

```
vTmpEyePt[2] += (float)(m_fElapsedTime * m_UserInput.fAxisZoom * cos(m_fWorldRotY));
if (vTmpEyePt[2] < 0 && vTmpEyePt[2] > -10)
{
    vEyePt[0] += (float)(-m_fElapsedTime * m_UserInput.fAxisZoom * sin(m_fWorldRotY));
    vEyePt[2] += (float)(m_fElapsedTime * m_UserInput.fAxisZoom * cos(m_fWorldRotY));
}
```

<u>การเคลื่อนที่ขึ้นลง</u>

ในการเคลื่อนที่ขึ้นและลงในนั้น เป็นการเคลื่อนที่กล้องในแนวแกน +y และ แกน –y โดยการเลื่อนขึ้นนั้นสามารถทำได้โดยกดคีย์บอร์ดปุ่มขึ้นและปุ่มลง ซึ่งมีสมการในการเคลื่อนที่ดังนี้

```
if( m_UserInput.fAxisRotateUD )
{
    vEyePt[1] += (float)(-m_fElapsedTime * m_UserInput.fAxisRotateUD * sin(m_fWorldRotY));
    vEyePt[1] += (float)(m_fElapsedTime * m_UserInput.fAxisRotateUD * cos(m_fWorldRotY));
}
```

<u>Render</u> เป็นขั้นตอนของการแสดงภาพ โดยจะนำภาพที่เราได้สร้างขึ้นมาจากขั้นตอนก่อนหน้านี้ไปแสดงในเวิลด์ เพื่อเป็นการแสดงภาพออกมาหน้าจอ

ในการวาดภาพ 3 มิติ นั้นใช้คำสั่งในการวาดดังนี้

```
m_mhead->Draw(degree,body_len/22,sumhigh,changeskin,skin)
```

ซึ่งการโค้ดด้านบนนั้นเป็นการวาด ออปเจค m_whead ซึ่งได้ทำการสร้างไว้ตั้งแต่ส่วนของ Initialize โดยในฟังก์ชัน Draw นั้นสามารถทำการปรับค่าต่างๆของ ออปเจคได้ก่อน(translate, rotate,

scaling) และหลังจากนั้นก็จะเป็นการกำหนด material และ texture ก่อนที่จะเตรียมนำไปวาด แล้วจึงนำไปวาดโดยใช้ฟังก์ชัน DrawIndexedPrimitive

โดยในโปรแกรมประยุกต์นี้จะทำการเปลี่ยนค่าต่างๆของหุ่นจำลอง 3 มิติ หรือ การใส่เสื้อให้หุ่นจำลอง 3 มิติ ในส่วนของ render นี้

Clean up เป็นการลบค่าต่าง ๆ ที่เก็บลงไว้ในหน่วยความจำก่อนที่ทำการเลิกใช้งานโปรแกรม โดยสามารถทำได้ดังนี้

```
m_whead->Cleanup();
```

ซึ่งเป็นการเรียกใช้ฟังก์ชัน cleanup ซึ่งภายในฟังก์ชัน cleanup นั้นจะการเรียกใช้ฟังก์ชัน Release ซึ่งเป็นการปล่อยหน่วยความจำต่างๆที่ได้จองไว้ในการสร้างออปเจค

End เป็นส่วนการจบการทำงานของโปรแกรม

การสร้างปุ่มกดเป็นรูปภาพ

นอกจากการทำงานที่กล่าวมาในข้างต้นแล้วยังมีการสร้างปุ่มกดสำหรับกดเผื่อใช้ในการควบคุม ซึ่งจะมีปุ่มที่สร้างโดยนำรูปภาพมาแปะซึ่งมีวิธีการสร้างดังนี้

การสร้างรูปภาพให้ปุ่ม เป็นการนำรูปภาพมาสร้างเป็นปุ่มเพื่อทำให้ปุ่มกดมีความสวยงาม สื่อความหมาย และทำให้โปรแกรมประยุกต์ดูน่าสนใจมากยิ่งขึ้น โดยใช้คลาส CBitmapButton ซึ่งต้องทำการ include <afxext.h> ก่อน โดยจะมี Method ที่สำคัญ 2 ตัวดังนี้

```
m_button1.Create("", WS_CHILD | WS_VISIBLE | BS_OWNERDRAW, CRect(475, 20, 565,95), this, 100);
m_button1.LoadBitmaps(IDB_BITMAP2,IDB_BITMAP1,IDB_BITMAP2,IDB_BITMAP2);
```

โดยทำการสร้างไว้ในฟังก์ชัน OnInitialUpdate เพื่อเป็นการตั้งค่าเริ่มต้นให้กับโปรแกรม เมื่อสร้างปุ่มกดที่ต้องการได้แล้ว สิ่งต่อไปก็คือการจัดการกับ Message ของปุ่มกดนั้นครับ เราจะต้องสร้าง Member Function ของวินโดว์ที่ปุ่มกดนั้นสังกัดอยู่ โดยจะตั้งชื่อว่าอะไรก็ได้ สมมุติว่าเป็นชื่อ BmpButtonClicked ก็แล้วกันครับ โดยจะมีรูปแบบดังนี้

```
afx_msg void BmpButtonClicked(UINT nID);
```

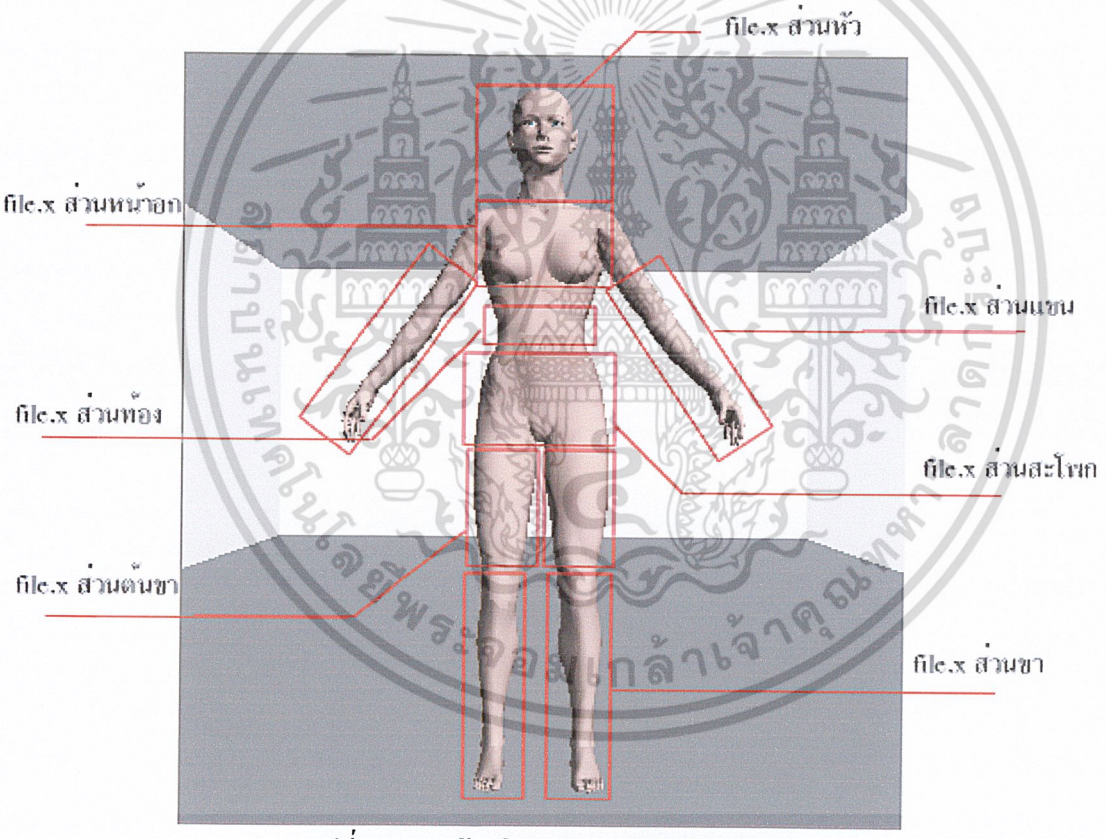
ที่ใช้ afx_msg นำหน้าก็เพื่อเป็นการเตือนตัวเองให้รู้ว่า ฟังก์ชั่นนี้จัดการเรื่อง Message อะครับ จะเห็นว่ามี Parameter เป็นหมายเลขประจำปุ่มกด ซึ่ง Windows จะส่งค่ามาให้แบบอัตโนมัติทันทีที่มีการกดปุ่ม ซึ่งหมายเลขจะเป็นไปตามที่กำหนดไว้ตอนสร้างปุ่ม ในฟังก์ชั่นเราก็ตรวจสอบว่า nID เป็นหมายเลขใด ก็ให้ไปทำงานตามหน้าที่ของหมายเลขนั้น เมื่อได้ฟังก์ชั่นแล้ว เราจะต้องใช้ ON_CONTROL_RANGE ในส่วนของ BEGIN_MESSAGE_MAP ด้วย ซึ่งรูปแบบจะเป็นดังนี้

```
ON_CONTROL_RANGE(BN_CLICKED, 100, 300, BmpButtonClicked)
```

โดยตัวเลข 100 เป็นหมายเลขของปุ่มแรก และ 300 เป็นหมายเลขของปุ่มสุดท้าย

6.2 การสร้างหุ่นจำลอง 3 มิติ

ในการสร้างหุ่นจำลอง 3 มิติในโปรแกรมประยุกต์นั้นเราจะทำโดยการนำ อวัยวะต่างๆซึ่งสร้างโดยโปรแกรม โพสเซอร์ 5.0 และ ทรีดีแม็กซ์สตูดิโอ 5.0 โดยนำอวัยวะที่สร้างขึ้นมาซึ่งประกอบด้วยส่วน หัว หน้าอก ท้อง สะโพก แขน ต้นขา และ ขา มาทำการประกอบกันในโปรแกรมประยุกต์ ทำให้เกิดเป็นตัวหุ่นจำลองสามมิติขึ้นมา โดยจะทำการสร้างแบบหุ่นจำลองเป็น เพศ ชายและหญิงโดยสามารถปรับขนาดของสัดส่วนต่างๆ สีผิว และสามารถยับแขน หมุนตัวได้ มีลักษณะดังรูป



รูปที่6-2 การสร้างหุ่นจำลอง 3 มิติในโปรแกรม

6.2.1 การนำ ภาพวัตถุ 3 มิติ มาใช้ในโปรแกรม

หลังจากที่ได้ออกแบบและได้สร้างวัตถุ 3 มิติจากโปรแกรม โพสเซอร์ 5 (Poser 5) และ ทรีดีสตูดิโอ แม็กซ์ 5 (3D studio MAX) แล้ว เราต้องทำการ export ออกมาเป็น file ประเภท .x ซึ่ง สำหรับ 3DsMax นั้นเราต้องใช้โปรแกรมเสริม (Plug-in) ซึ่งจะเป็นการเพิ่มความสามารถในการ export file ให้เป็น .x file ได้ ขั้นตอนต่อไปคือการทำการโหลด .x file เข้ามาใช้ในโปรแกรมของเรา

ในโปรแกรม นั้นเราได้สร้างคลาสขึ้นมาสำหรับการโหลดและการเรนเดอร์ mesh จาก .x file โดยที่ตัว Constructor ของคลาสที่เราสร้างนั้น จะเอาไว้โหลด mesh จาก .x file และเก็บไว้ในหน่วยความจำ ตัว Destructor เอาไว้คืนหน่วยนความจำที่เราเอาไว้เก็บ mesh และ method render แสดงถึงวิธีการเรนเดอร์ ตัว mesh ที่ได้โหลดเอาไว้

Constructor ของคาสที่เราสร้างนั้น จะรับค่า 2 ตัวแปร ตัวแรกเป็นตัวแปรพอยท์เตอร์ ซึ่งชี้ไปที่ตัว m_pdevice และตัวที่ 2 เป็นสตริงที่เอาไว้เก็บเส้นทาง(path) ของ .x file ที่จะเอาไว้โหลด เราใช้ฟังก์ชัน D3DXLoadMeshFromX ในการโหลด mesh เข้าไว้ในหน่วยความจำ จากนั้นเราได้สร้าง array ขึ้นมา 2 ตัว โดยตัวแรกเอาไว้เก็บ material และอีกตัวเอาไว้เก็บ texture ของโมเดลเพื่อนำใช้วาดใน ส่วนของ render

โค้ดตัวอย่างของการโหลดโมเดล .x file

```
hr = D3DXLoadMeshFromX(
        "wmodel/w_head.x",
        D3DXMESH_MANAGED,
        m_pDevice,
        &adjBuffer,
        &mtrlBuffer,
        0,
        &numMtrls,
        &m_pSourceMesh);
```

และยังมีโค้ดในส่วนการปรับปรุงคุณภาพของ .x file ให้ดีขึ้นโดยสามารถทำได้ดังโค้ดข้างล่าง

```
hr = m_pSourceMesh->OptimizeInplace(
        D3DXMESHOPT_ATTRSORT |
        D3DXMESHOPT_COMPACT  |
        D3DXMESHOPT_VERTEXCACHE,
        (DWORD*)adjBuffer->GetBufferPointer(),
        (DWORD*)adjBuffer->GetBufferPointer(), // new adjacency info
        0, 0);
```

6.2.2 การวาดหุ่นจำลอง 3 มิติ

ในการวาดภาพหุ่นจำลอง 3 มิติให้แสดงผลออกมาทางหน้าจอแสดงผลได้นั้นจะต้องเรียกใช้คลาสของ .x file ในฟังก์ชัน Draw โดยคำสั่งส่วนที่ใช้ในการวาดภาพคือ

```
for(int i = 0; i < (int)m_mtrls.size(); i++)
{
        m_pPMesh->DrawSubset(i);
}
```

และในการวาดหุ่นจำลอง 3 มิติจะทำการวาดอวัยวะตั้งแต่ส่วนขาขึ้นไปจนถึงส่วนหัว ไล่ทีละส่วน เพื่อที่อวัยวะส่วนต่างๆจะได้ต่อกันได้อย่างถูกต้อง ดังต่อไปนี้

เท้า → ขา → ต้นขา → สะโพก → เอว → อก → แขน → หัว

6.2.3 การหมุนตัวของหุ่นจำลอง 3 มิติ

ในการหมุนตัวหุ่นจำลอง 3 มิตินั้นเราสามารถทำได้โดยการกดปุ่มลูกศร ซ้าย ขวา โดยการกดปุ่มซ้ายเป็นการหมุนหุ่นจำลอง 3 มิติไปทางซ้าย และการกดขวาก็เป็นการปุ่มหุ่นจำลอง 3 มิติ ไปทางขวา ซึ่งในการหมุนตัวนี้ ต้องทำการติดต่อกับคีย์บอร์ดโดยมีวิธีการสร้างดังนี้ เริ่มแรกทำการ map ปุ่มก่อน

```
{ INPUT_ROTATE_LEFT,        DIKEYBOARD_LEFT,                0,
TEXT("Rotate left"), },
{ INPUT_ROTATE_RIGHT,       DIKEYBOARD_RIGHT,               0,
TEXT("Rotate right"), },
```

แล้วทำการตรวจสอบว่าเป็นการกดปุ่มไหน โดยทำการตรวจสอบที่ตัวแปร rgdod[j].uAppData ว่าเป็น input ใด

```
case INPUT_ROTATE_LEFT:   pInputDeviceState->bButtonRotateLeft  =
bButtonState; break;
case INPUT_ROTATE_RIGHT:  pInputDeviceState->bButtonRotateRight =
bButtonState; break;
```

เมื่อตรวจสอบเรียบร้อยแล้วก็จะแยกไปทำหากเป็นปุ่มซ้ายจะไปทำงานที่

```
if( pInputDeviceState->bButtonRotateLeft )
        degree += 0.02f;
```

หากเป็นการกดปุ่มซ้ายจะไปทำที่คำสั่ง

```
else if( pInputDeviceState->bButtonRotateRight )
        degree -= 0.02f;
```

โดยค่า degree นั้นจะถูกนำไปใช้ต่อในคลาสของ .x file ที่ต้องการให้หมุนในฟังก์ชัน Draw โดยจะทำการหมุนรอบตัวเองโดยใช้คำสั่ง D3DXMatrixRotationY ซึ่งเป็นคำสั่งที่ใช้ในการหมุนรอบแกน y โดยสามารถทำได้ดังนี้

```
D3DXMatrixRotationY(&m_world , degree);
```

6.2.4 การปรับขนาดของหุ่นจำลอง 3 มิติ

เมื่อได้นำ .x file เข้ามาให้โปรแกรมใช้งานได้แล้ว ในการที่จะสร้างหุ่นจำลอง 3 มิติในโปรแกรมประยุกต์ให้มีขนาดสัดส่วนของอวัยวะต่างๆตรงตามสัดส่วนที่ผู้ใช้ใส่เข้าไป ได้แก่ขนาดของความยาว (สูง) และความหนา ของอวัยวะต่างๆเพื่อให้หุ่นจำลอง 3 มิติมีขนาดเหมือนผู้ใช้มากที่สุด จึงจำเป็นที่จะต้องมีการปรับขนาดทั้งในส่วนของความยาว และความกว้างของ .x file ที่ได้สร้างเอาไว้ก่อนหน้านี้แล้วให้มีขนาดต่างจากขนาดตั้งต้นเสียก่อน มีซึ่งการปรับขนาดของ .x file นี้สามารถแบ่งได้เป็นการปรับในส่วนของความยาว และในส่วนของเส้นรอบวง

6.2.4.1 การปรับในส่วนของความยาว

การปรับขนาดของ .x file ในส่วนของความยาว (สูง) ของอวัยวะส่วนต่างๆจะปรับโดยใช้สัดส่วนเท่ากับ

Length = ความยาวที่ผู้ใช้ใส่เข้ามา / ความยาวตั้งต้นของ .x file

เมื่อได้สัดส่วนความยาวที่ต้องปรับแล้ว โดยค่า Length นั้นจะถูกนำไปใช้ต่อในคลาสของ .x file ในฟังก์ชัน Draw โดยจะทำการปรับขนาดโดยใช้คำสั่ง D3DXMatrixScaling

โดยการที่จะให้ขนาดของ .x file เปลี่ยนเฉพาะในแนวดิ่งสามารถทำได้โดยใช้คำสั่ง

```
D3DXMatrixScaling(&m_scaling,1,Length,1);
```

ซึ่งจะเป็นการปรับสัดส่วนของ .x file ตามแนวแกน y ให้มีสัดส่วนเท่ากับ length เท่า เทียบกับความยาวตั้งต้นของ .x file

6.2.4.2 การปรับในส่วนของเส้นรอบวง

การปรับขนาดของ .x file ในส่วนของความกว้าง (เส้นรอบวง) ของอวัยวะส่วนต่างๆจะปรับโดยใช้สัดส่วนเท่ากับ

Round = ความกว้างที่ผู้ใช้ใส่เข้ามา / ความกว้างตั้งต้นของ .x file

เมื่อได้สัดส่วนของความกว้างที่ต้องปรับแล้ว โดยค่า Round นั้นจะถูกนำไปใช้ต่อในคลาสของ .x file ในฟังก์ชัน Draw โดยจะทำการปรับขนาดโดยใช้คำสั่ง D3DXMatrixScaling

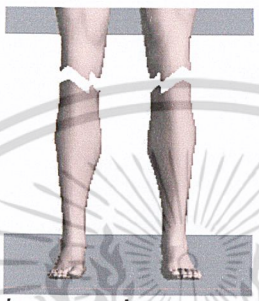
โดยการที่จะให้ขนาดของ .x file เปลี่ยนเฉพาะในแนวราบสามารถทำได้โดยใช้คำสั่ง

```
D3DXMatrixScaling(&m_scaling,Round,1,round);
```

ซึ่งจะเป็นการปรับสัดส่วนของ .x file ตามแนวแกน x และ z ให้มีสัดส่วนเท่ากับ round เท่า เทียบกับความกว้างตั้งต้นของ .x file

6.2.5 การเคลื่อนตำแหน่งของหุ่นจำลอง 3 มิติ

เมื่อได้ทำการปรับขนาดอวัยวะส่วนต่างๆของหุ่นจำลอง 3 มิติแล้ว จะได้อวัยวะส่วนต่างๆที่มีขนาดความสูงไม่เท่าเดิม ดังนั้นตำแหน่งที่จะต้องเริ่มต้นวาดชิ้นส่วนอวัยวะต่างๆจึงต้องเปลี่ยนแปลงไปจากตำแหน่งเริ่มต้นเดิมด้วย



รูปที่ 6-3 แสดงส่วนที่ไม่ต่อกันเนื่องจากการปรับขนาดขาให้เล็กลง

การที่จะวาดอวัยวะส่วนต่างๆให้ได้ตรงตามที่ควรจะเป็นนั้น จะต้องมีการเคลื่อนตำแหน่งของชิ้นส่วนอวัยวะต่างๆให้มาอยู่ถูกที่ด้วย โดยระยะทางที่จะต้องเคลื่อนที่สามารถหาได้โดยหาตำแหน่งที่จะต้องเริ่มต้นวาดอวัยวะส่วนนั้นจริงๆ กับตำแหน่งเริ่มต้นเมื่ออวัยวะนั้นๆถูกปรับขนาดแล้ว จากนั้นนำค่าทั้งสองมาลบกัน ก็จะได้ระยะทางที่จะต้องทำการเคลื่อนตำแหน่ง ดังสูตร

Translate = ตำแหน่งที่จะต้องเริ่มต้นวาด - ตำแหน่งเริ่มต้นเมื่อถูกปรับขนาดแล้ว

โดยค่า Translate นั้นจะถูกนำไปใช้ต่อในคลาสของ .x file ในฟังก์ชัน Draw โดยจะทำการเคลื่อนที่วัตถุโดยใช้คำสั่ง D3DXMatrixTranslation ซึ่งถ้าต้องการให้เคลื่อนที่ไปในแนวแกน y สามารถทำได้โดย

```
D3DXMatrixTranslation(&m_translation , 0, Translate, 0);
```

โดยมีวิธีหาตำแหน่งตำแหน่งที่จะต้องเริ่มต้นวาดและ ตำแหน่งเริ่มต้นเมื่อถูกปรับขนาดแล้วดังนี้

6.2.5.1 การหาตำแหน่งที่จะต้องเริ่มต้นวาด

ต้องคิดออกมาตั้งแต่ตอนที่ทำการวาดชิ้นส่วนก่อนหน้าแล้ว

- ถ้าอวัยวะส่วนถัดไปเป็นส่วนที่ต้องต่อขึ้นไปข้างบน เช่น จากส่วนต้นขาไปเป็นส่วนสะโพก ให้เลือกจุดที่สูงที่สุดของชิ้นส่วนนั้นๆ

- ถ้าอวัยวะส่วนถัดไปเป็นส่วนที่ต้องต่อออกไปข้างๆ เช่น จากส่วนช่วงไหล่ไปเป็นต้นแขน ให้เลือกจุดที่มีความกว้างมากที่สุด

- เมื่อได้ตำแหน่งที่ต้องการแล้วให้นำไปคูณกับค่าสัดส่วนที่ต้องปรับขนาดเพื่อจะได้รู้ว่าเมื่อทำการปรับขนาดแล้วตำแหน่งนี้จะมีค่าเท่าใด

- เมื่อได้ค่าออกมาแล้วให้นำไปบวกกับค่าที่วัตถุนี้ต้องเคลื่อนที่ ก็จะได้ตำแหน่งเริ่มต้นของวัตถุต่อไป

6.2.5.2 การหาตำแหน่งเริ่มต้นของวัตถุ

- ถ้าอวัยวะนั้นเป็นส่วนที่ต่อขึ้นมาจากข้างล่าง เช่น จากส่วนสะโพกขึ้นมาจากส่วนต้นขา ให้เลือกจุดที่ต่ำที่สุดของชิ้นส่วนนั้นๆ

- ถ้าอวัยวะนั้นเป็นส่วนที่ต่อออกมาข้างๆ เช่น จากส่วนต้นแขนมาจากช่วงไหล่ ให้เลือกจุดที่มีความกว้างน้อยที่สุด

- เมื่อได้ตำแหน่งที่ต้องการแล้วให้นำไปคูณกับค่าสัดส่วนที่ต้องปรับขนาดเพื่อจะได้รู้ว่าเมื่อทำการปรับขนาดแล้วตำแหน่งนี้จะมีค่าเท่าใด ก็จะได้ตำแหน่งเริ่มต้นของวัตถุนั้นๆแล้ว

เมื่อนำมาเขียนเป็นสมการจะได้ว่า

$$Translate_2 = ((Ymax_1 * Scale_1) + Translate_1) - (Ymin_2 * Scale_2)$$

6.2.6 การสร้างโพลีกอน(polygon)ส่วนเชื่อมต่อระหว่างอวัยวะ

ในส่วนนี้ทำการสร้างโดยการวาดpolygon แต่ละvertex เองโดยการสร้างแบบจำลอง 3 มิติในโปรแกรม ทรีดีสตูดิโอแม็กซ์ 5 (3D studio Max5) หลังจากนั้นก็ทำการ export ให้เป็น .x file เมื่อทำการสร้างเป็น .x file แล้วก็การดูค่า vertex ที่อยู่ใน .x file แล้วจึงนำค่า vertex เหล่านั้นมาสร้างเป็น polygon โดยจะทำการสร้างเป็นสามเหลี่ยมหลายๆอันมาต่อกันเป็นรูปร่าง

การสร้างสร้าง polygon ในลักษณะนี้สามารถทำได้โดยการนำค่า vertex ทั้งหมดมาใส่ใน vertex buffer โดยใช้คำสั่ง CreateVertexBuffer ในการสร้าง โดยจะต้องมีการใส่ค่าของจำนวน vertex ทั้งหมดไว้ด้วย ดังตัวอย่าง

```
m_pDevice->CreateVertexBuffer(
            191 * sizeof(Vertex),
            D3DUSAGE_WRITEONLY,
            FVF_VERTEX,
```

```
                D3DPOOL_MANAGED,
                &m_pVB,
                0);
```

หลังจากนั้นก็ทำการสร้าง index buffer เพื่อใช้ในการเก็บค่าอินเด็กซ์ของโพลีกอนแต่ละตัวที่สร้างขึ้นมา โดยจะต้องมีการใส่ค่าของจำนวน index ทั้งหมดเช่นกัน โดยการสร้าง index buffer นั้นจะใช้คำสั่ง CreateIndexBuffer ในการสร้าง โดยวิธีการสร้างดังตัวอย่าง

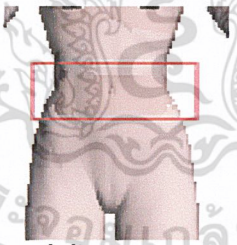
```
m_pDevice->CreateIndexBuffer(
                870 * sizeof(WORD),
                D3DUSAGE_WRITEONLY,
                D3DFMT_INDEX16,
                D3DPOOL_MANAGED,
                &m_pIB,
                0);
```

เมื่อทำการกำหนดเสร็จเรียบร้อยแล้วเราต้องทำการวาดโดยใช้คำสั่งดังนี้

```
m_pDevice->DrawIndexedPrimitive(
                D3DPT_TRIANGLELIST,
                0,
                0,
                191,
                0,
                290);
```

โดย เลข 191 หมายถึง จำนวน vertex ทั้งหมด และเลข 290 หมายถึงจำนวน สามเหลี่ยมทั้งหมดที่สร้างขึ้นมา

เหตุผลที่ต้องสร้างด้วยวิธีนี้เนื่องจากการสร้างเช่นนี้สามารถ ปรับ vertex ที่ต้องการได้และจะทำให้ได้ polygon ที่มีรายละเอียดและดูสมจริงมากขึ้น ซึ่งจะได้ผลดังรูป



รูปที่ 6-4 แสดงส่วนที่เชื่อมต่อระหว่างสะโพกกับท้อง

6.2.7 การเปลี่ยนสีผิวของหุ่นจำลองสามมิติ

ในส่วนนี้สามารถทำไดโดยการใช้คำสั่ง D3DXCreateTextureFromFile ในฟังก์ชัน Draw ซึ่งเป็นฟังก์ชันของคลาสอวัยวะส่วนต่างๆที่สร้างขึ้นมาสำหรับ ใช้ในการวาดโมเดลลงในโปรแกรม ซึ่งฟังก์ชันนี้จะถูกเรียกใช้ใน ฟังก์ชัน render ซึ่งเรียกใช้ตลอดเวลา มีตัวอย่างการใช้ดังนี้

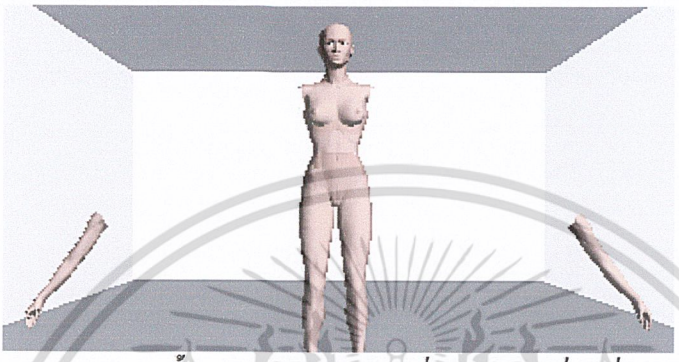
```
if (changeskin == true)
        {
                D3DXCreateTextureFromFile(
                                m_pDevice,
                                texfile,
                                &tex);
```

```
}
m_pDevice->SetTexture(0, tex);
```

โดยทำการใส่เส้นทาง (path) ของtexture ที่ต้องการนำไปแปะไว้ในตัวแปร texfile

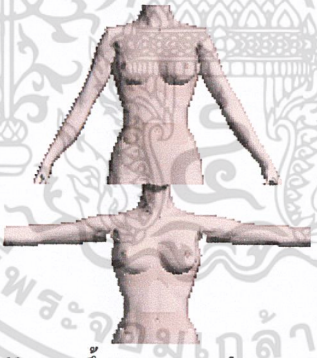
6.2.8 การขยับแขนของหุ่นจำลอง 3 มิติ

ในส่วนของการขยับแขนให้สามารถหมุนขึ้น – ลงได้นั้น จำเป็นที่จะต้องใช้คำสั่ง D3DXMatrixRotationZ เพื่อที่จะให้แขนทั้ง 2 ข้างหมุนขึ้น – ลง ได้ในแนวแกน z



รูปที่ 6-5 การขยับแขนขึ้น – ลงของหุ่นจำลองเมื่อยังไม่ได้เคลื่อนที่แขนกลับมา

แต่การหมุนในแนวแกน z นั้น วัตถุจะถูกหมุนไปยังตำแหน่งที่ไม่ถูกต้อง ดังนั้นจึงต้องทำการเคลื่อนที่วัตถุนั้นกลับมายังตำแหน่งที่ถูกต้องด้วย โดยใช้ D3DXMatrixTranslation



รูปที่ 6-6 การขยับแขนขึ้น – ลงของหุ่นจำลองที่สมบูรณ์แล้ว

ส่วนของคำสั่งที่เกี่ยวข้องกับการหมุนแขนซ้ายมีดังนี้

```
D3DXMatrixRotationZ(&m_world , degree2);
D3DXMatrixRotationY(&m_rotation , degree);
D3DXMatrixMultiply(&m_world,&m_world,&m_rotation);
```

คำสั่งข้างบนเป็นการหมุนวัตถุในแนวแกน z ก่อน จากนั้นจึงหมุนในแนวแกน y

```
D3DXMatrixTranslation(&m_translation ,
(Xtranslate/10 + (sin(degree2)*(5.6)*scale*(chest_r/36)) + (1-scale)*sin(degree2)*6.25 )*cos(degree),
translate/10 + 7.13*(chest_r/36) - (cos(degree2)*7.15)*(chest_r/36) + (1-scale)*sin(degree2)*0.6,
5 - sin(degree)*(sin(degree2)*5.6*scale*(chest_r/36)) - (1-scale)*sin(degree2)*6.25*sin(degree));
D3DXMatrixMultiply(&m_world,&m_world,&m_translation);
```

คำสั่งข้างบนเป็นส่วนของการเคลื่อนที่ให้วัตถุที่ถูกหมุนในแนวแกน z กลับไปยังตำแหน่งก่อนที่จะถูกหมุน โดยที่

```
(Xtranslate + (sin(degree2)*(5.6)*scale*(chest_r/36)) + (1-scale)*sin(degree2)*6.25 )*cos(degree)
```

คือการเคลื่อนที่ในแนวแกน x ซึ่งตัวแปร

- Xtranslate เป็นการเคลื่อนที่ในกรณีที่ส่วนหน้าอกถูกขยายหรือลดขนาด
- (sin(degree2)*(5.6)*scale*(chest_r/36)) เป็นการเคลื่อนที่ให้ส่วนแขนกลับมายังตำแหน่งในแกน x ก่อนที่จะถูกหมุน
- (1-scale)*sin(degree2)*6.25 เป็นการเคลื่อนที่ให้ส่วนแขนกลับมายังตำแหน่งในแกน x ก่อนที่จะถูกหมุนในกรณีที่ส่วนแขนถูกปรับเปลี่ยนขนาด
- cos(degree) เป็นการกำหนดสัดส่วนในการเคลื่อนที่เมื่อหุ่นจำลอง 3 มิติถูกหมุนรอบแกน y

```
Translate + 7.13*(chest_r/36) - (cos(degree2)*7.15)*(chest_r/36) + (1-scale)*sin(degree2)*0.6
```

คือการเคลื่อนที่ในแนวแกน y

- Translate เป็นการเคลื่อนที่ในกรณีที่ส่วนหน้าอกถูกขยายหรือลดความสูง
- 7.13*(chest_r/36) - (cos(degree2)*7.15)*(chest_r/36) เป็นการเคลื่อนที่ให้ส่วนแขนกลับมายังตำแหน่งในแกน y ก่อนที่จะถูกหมุน
- (1-scale)*sin(degree2)*0.6 เป็นการเคลื่อนที่ให้ส่วนแขนกลับมายังตำแหน่งในแกน y ก่อนที่จะถูกหมุนในกรณีที่ส่วนแขนถูกปรับเปลี่ยนขนาด

```
((sin(degree2)*5.6*scale*(chest_r/36)) - (1-scale)*sin(degree2)*6.25)*sin(degree)
```

คือการเคลื่อนที่ในแนวแกน z

- sin(degree2)*5.6*scale*(chest_r/36) เป็นการเคลื่อนที่ให้ส่วนแขนกลับมายังตำแหน่งในแกน z ก่อนที่จะถูกหมุน

- (1-scale)*sin(degree2)*6.25) เป็นการเคลื่อนที่ให้ส่วนแขนกลับมายังตำแหน่งในแกน z ก่อนที่จะถูกหมุนในกรณีที่ส่วนแขนถูกปรับเปลี่ยนขนาด
- sin(degree) เป็นการกำหนดสัดส่วนในการเคลื่อนที่เมื่อหุ่นจำลอง 3 มิติถูกหมุนรอบแกน y

ส่วนคำสั่งที่เกี่ยวข้องกับการหมุนแขนขวาจะมีการเปลี่ยนแปลงค่าบางอย่าง ดังนี้

```
D3DXMatrixRotationZ(&m_world , -degree2);
D3DXMatrixRotationY(&m_rotation , degree);
D3DXMatrixMultiply(&m_world,&m_world,&m_rotation);

D3DXMatrixTranslation(&m_translation ,
(-Xtranslate/10 + (sin(degree2)*(-5.6)*scale*(chest_r/36)) + (scale-1)*sin(degree2)*6.25 )*cos(degree),
translate/10 + 7.13*(chest_r/36) - (cos(degree2)*7.15)*(chest_r/36) + (1-scale)*sin(degree2)*0.6,
5 + sin(degree)*(sin(degree2)*5.6*scale*(chest_r/36)) + (1-scale)*sin(degree2)*6.25*sin(degree));
D3DXMatrixMultiply(&m_world,&m_world,&m_translation);
```

6.3 การสร้างเสื้อผ้า

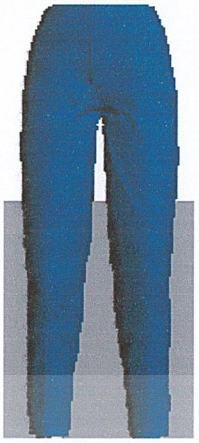
ในการสร้างเสื้อผ้า โดยเริ่มต้นนั้นเราได้ทำการสร้างชิ้นส่วนของเสื้อผ้าในโปรแกรม โพสเซอร์ 5.0 และ ทรีดีแม็กซ์สตูดิโอ 5.0 หลังจากนั้นก็ทำการ export ออกมาเป็น .x file โดยเสื้อผ้าในแต่ละแบบจะมีรูปแบบการสร้างไม่เหมือนกัน โดยเสื้อผ้าจะมีการสร้างในลักษณะเดียวกับการสร้างตัวคน คือ สามารถหมุนได้ ขยับช่วงแขนของเสื้อได้ (ในกรณีที่เสื้อมีแขน) สามารถเปลี่ยนลวดลายของเสื้อได้ โดยมีวิธีการเช่นเดียวกับการเปลี่ยนสีของผิวตัวหุ่นจำลองสามมิติ

```
if (changetshirttex == true)
        {
                D3DXCreateTextureFromFile(
                                        m_pDevice,
                                        texfile,
                                        &tex);
        }
        m_pDevice->SetTexture(0, tex);
```

ซึ่ง ฟังก์ชัน SetTexture จะนำค่าของ tex ไปใส่ลงบนโพลีกอนที่สร้างขึ้นและเมื่อทำการเรียกใช้ฟังก์ชัน Draw ใน ฟังก์ชันrender ก็จะเป็นการวาดออกมา ดังตัวอย่าง

```
m_wtjeans->Draw(degree,wjeanstex,changejeanstex);
m_wmjeans->Draw(degree,wjeanstex,changejeanstex);
m_wdjeans->Draw(degree,wjeanstex,changejeanstex);
```

ซึ่งผลลัพธ์ที่ได้จะเป็นโพลีกอนเสื้อผ้าที่ทำการแปะลวดลายลงเรียบร้อยแล้วดังรูป

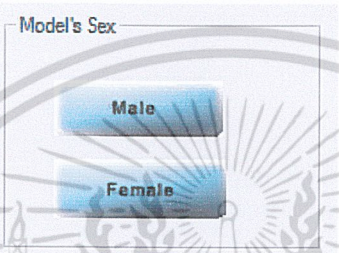


รูปที่ 6-7 แสดงการแปะtexture ลงบนpolygon

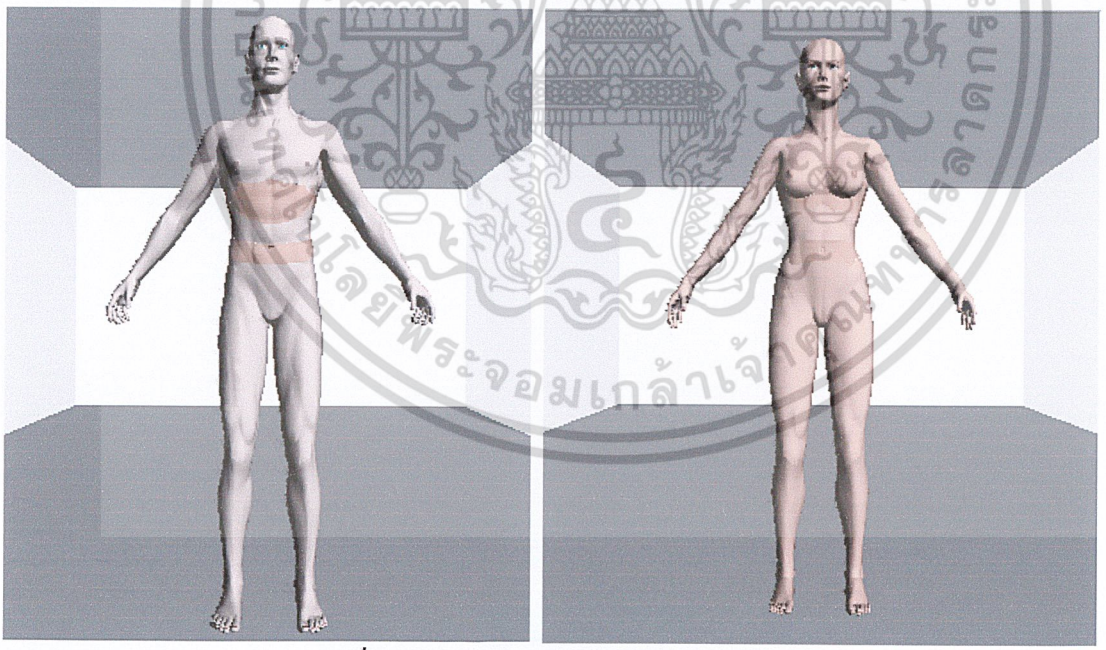
บทที่ 7
การทำงานของระบบ

7.1 การเลือกเพศของหุ่นจำลอง

การเลือกเพศของหุ่นจำลอง 3 มิติ สามารถทำได้โดย การเลือกจากปุ่ม โดยปุ่ม Male เป็นการเลือกว่าต้องการสร้างหุ่นจำลอง 3 มิติเป็นเพศชาย และปุ่ม Female ก็เป็นการเลือกว่าต้องการสร้างหุ่นจำลอง 3 มิติเป็นเพศหญิง



รูปที่ 7-1แสดงปุ่มที่ใช้ในการเลือกเพศ



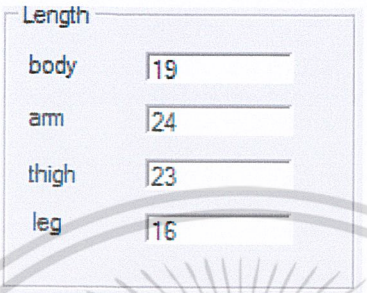
รูปที่7-2 แสดงรูปหุ่นจำลองเพศชายและหญิง

7.2 การรับสัดส่วนของตัวหุ่นจำลอง 3 มิติ

ในการรับค่าสัดส่วนต่างๆ โดยแบ่งเป็น การรับสัดส่วนความยาว, การรับส่วนสัดในแนวเส้นรอบวง, และในกรณีที่เป็นหุ่นจำลอง 3 มิติ ผู้หญิง ก็จะมีการเปลี่ยนแปลงขนาดของหน้าอก (CUP)

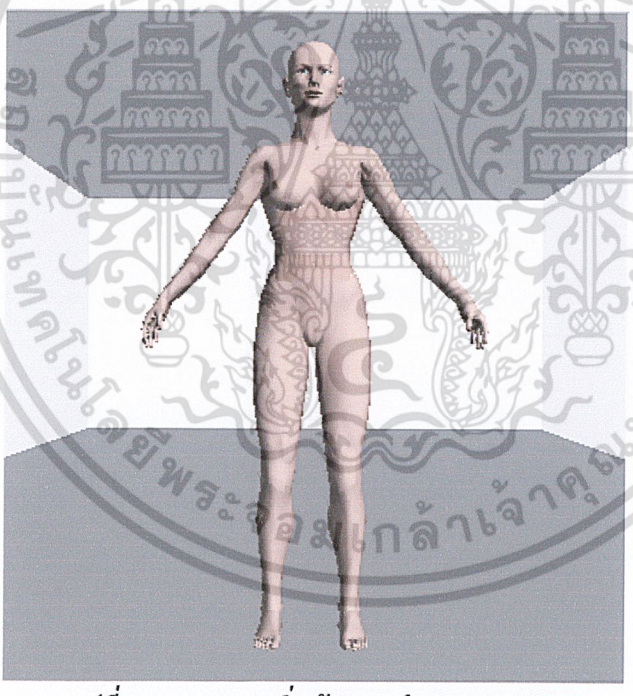
7.2.1 การปรับสัดส่วนความยาวของหุ่นจำลอง 3 มิติ

ในการรับค่าสัดส่วนความยาวของหุ่นจำลอง 3 มิติ นั้นสามารถปรับค่าได้ 4 ค่า คือความยาวของลำตัว (ตั้งแต่ต้นคอไปถึงก้นกบ), ความขาวของแขน (ตั้งแต่หัวไหล่ไปจนถึงหลังมือ), ความยาวของต้นขา (ตั้งแต่สะโพกไปถึงหัวเข่า), ความขาวของขา (ตั้งแต่เข่าไปถึงเท้า) โดยค่าเหล่านี้รับโดยใส่ค่าลงไปใน edit box ที่ได้สร้างไว้ ดังรูป 7-3 โดยค่าเหล่านี้มีหน่วยเป็นนิ้ว



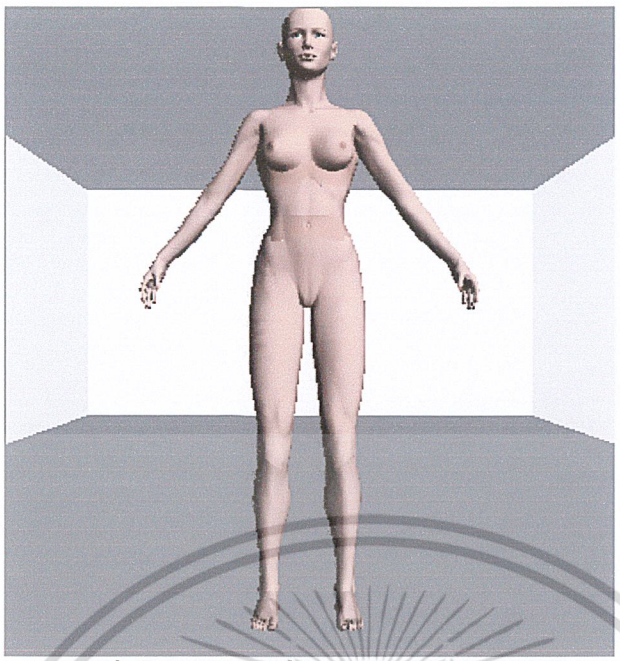
รูปที่ 7-3 Edit box ในใช้ในการรับค่าความยาวของสัดส่วน

จากรูปเมื่อใส่ค่าดังกล่าวหุ่นจำลอง 3 มิติ ที่ได้จะมีขนาดดังรูป 7-4



รูปที่ 7-4 แสดงภาพเริ่มต้นของหุ่นจำลอง 3 มิติ

เมื่อทำการปรับค่าในช่อง thigh ให้เป็น 26 และ ในช่อง leg .ให้เป็น 17 จะสังเกตผลเห็นว่าหุ่นจำลอง 3 มิติ นั้นมีช่วงขาที่ยาวเพิ่มขึ้น ดังรูป 7-5



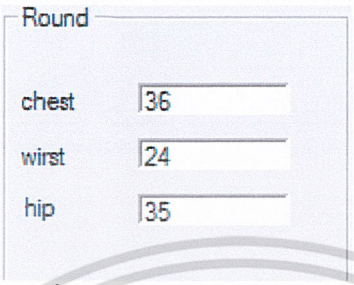
รูปที่ 7-5 แสดงการเพิ่มขนาดของต้นขาและขา

หากลองลดขนาดของแขนจาก 24 ให้เป็น 20 และลดขนาดความยาวของลำตัวจาก 19 ให้เป็น 16 จะได้ผลลัพธ์ดังรูป 7-6 ซึ่งจะสังเกตเห็นว่าส่วนของแขนและลำตัวจะมีขนาดสั้นลง

รูปที่ 7-6 แสดงการลดขนาดของแขนและลำตัว

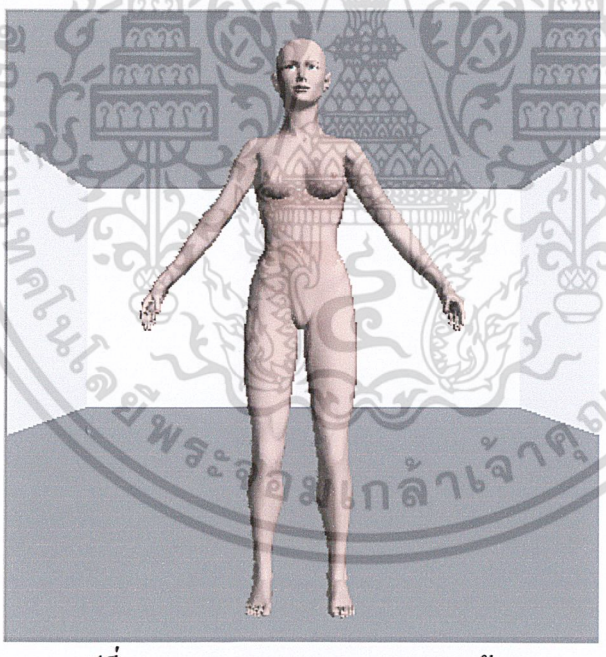
7.2.2 การปรับสัดส่วนความยาวโดยรอบของหุ่นจำลอง 3 มิติ

ในการปรับสัดส่วนความยาวโดยรอบของหุ่นจำลองจำลอง 3 มิตินั้น สามารถปรับได้ 3 ค่า ได้แก่ ส่วนของ รอบอก รอบเอว และสะโพก โดยค่าเหล่านี้สามารถทำการปรับได้โดยใส่ค่าลงไปใน edit box ที่สร้างขึ้นดังรูป 8-7



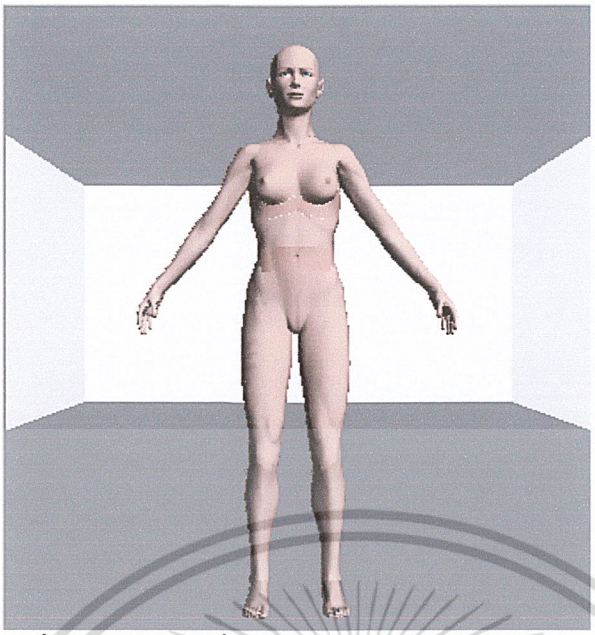
รูปที่ 7-7 Edit box ที่ใช้ในการรับค่าสัดส่วนความยาวโดยรอบตัว

โดยปกติหุ่นจำลองจะมีรูปร่างเริ่มต้นดังรูป 8-2 หากทำการปรับสัดส่วนของหุ่นจำลอง 3 มิติ โดยเปลี่ยนค่าหน้าอกโดยลดจาก 36 เป็น 34 โดยการเปลี่ยนค่าในช่อง chest จะได้ผลดังรูปที่ 7-8



รูปที่ 7-8 แสดงภาพการลดขนาดของหน้าอก

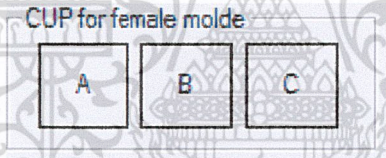
หากลองเพิ่มขนาดรอบเอว จาก 24 เป็น 26 และลดขนาดสะโพกลงจาก 36 เป็น 34 จะได้ผลลัพธ์ดัง รูปที่ 7-9 ซึ่งจะสังเกตเห็นว่า เอวมีขนาดใหญ่ขึ้นและสะโพกมีขนาดที่เล็กลง



รูปที่ 7-9 แสดงภาพเพิ่มขนาดเอวและลดขนาดของสะโพก

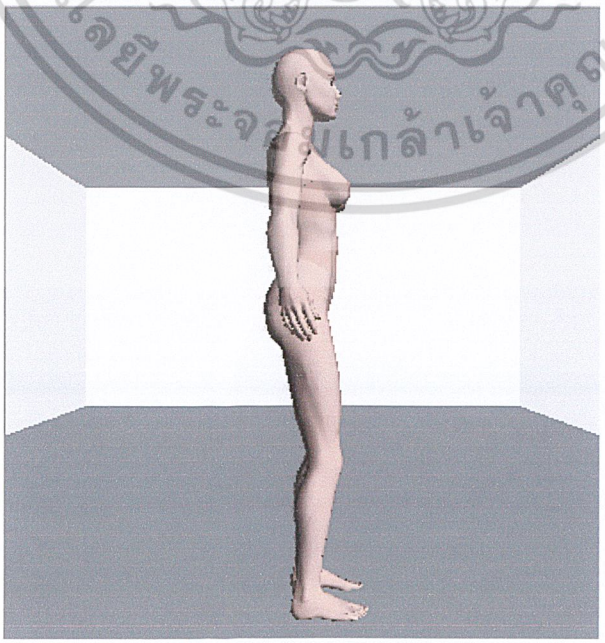
7.2.3 การปรับขนาดของหน้าอก (เฉพาะหุ่นจำลอง 3 มิติเพศหญิง)

ในการปรับค่าของหน้าอกนั้นสามารถปรับได้ 3 ขนาดคือ คัพ (CUP) A, คัพ B และ คัพ C ซึ่งแต่ละคัพนั้นจะมีขนาดไม่เท่ากัน โดยสามารถเลือก คัพได้โดยการกดปุ่ม ดังรูป



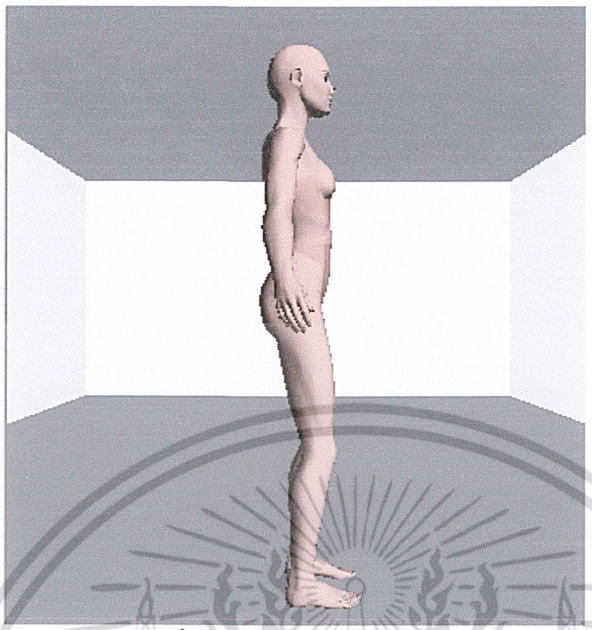
รูปที่ 7-10 แสดงปุ่มปรับขนาดของหน้าอก

โดยค่าเริ่มต้นที่ตั้งไว้จะเป็น คัพ C ซึ่งมีขนาดดังรูป 7-11



รูปที่ 7-11 แสดงขนาดของหน้าอกคัพ C

เมื่อทำการเลือกขนาดของหน้าอกเป็นคัพ A จะได้หน้าอกที่มีขนาดเล็กลงดังรูป 7-12



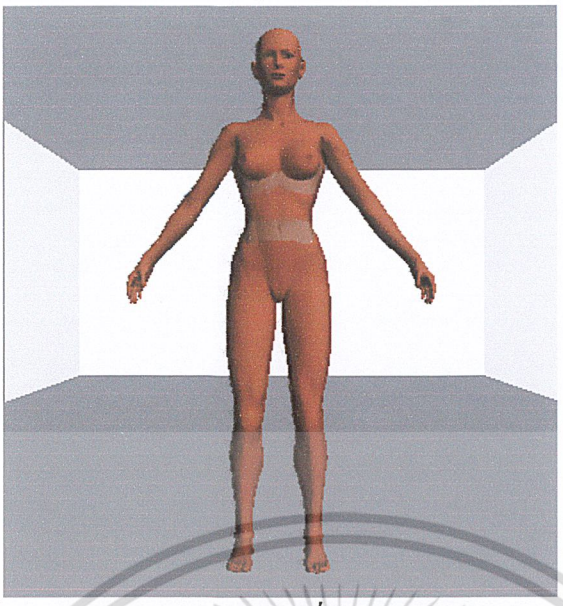
รูปที่ 7-12 แสดงขนาดของหน้าอกคัพ A

7.3 การเปลี่ยนสีผิวของหุ่นจำลอง 3 มิติ

ในการปรับค่าของสีผิวนั้นสามารถทำได้โดยการเลือกกดปุ่ม ปรับผิวที่สีผิวโดยมีสีให้เลือกที่ดังรูป 8-13

รูปที่ 7-13 แสดงปุ่มที่ใช้ในการปรับสีผิว

เมื่อทำการกดปุ่มสีผิวของตัวหุ่นจำลองจะเปลี่ยนไปตามสีที่ได้เลือกไว้ เช่นเมื่อทำการกดปุ่ม จะได้ผลลัพธ์ดังรูป 7-14

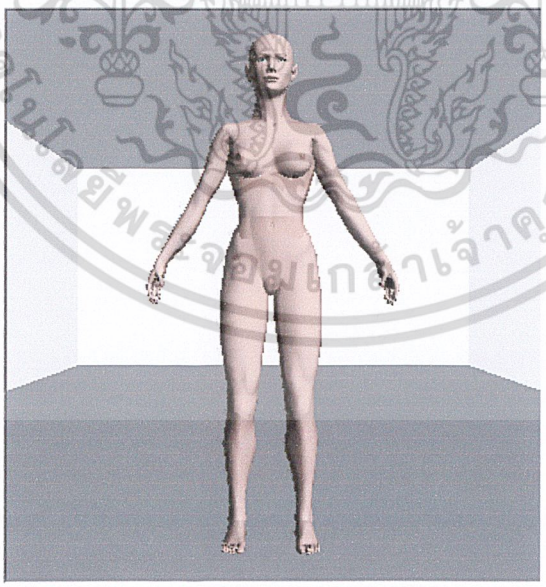


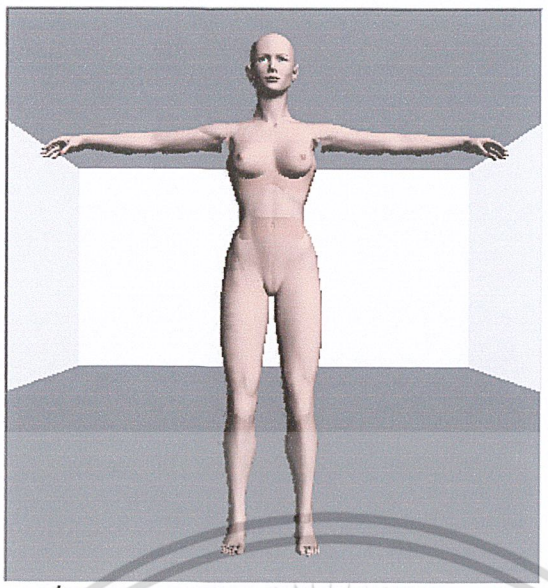
รูปที่ 7-14 แสดงผลลัพธ์ของการเปลี่ยนสีผิวของหุ่นจำลอง 3 มิติ

7.4 การขยับแขนของหุ่นจำลอง 3 มิติ

การขยับแขนขึ้น – ลงของหุ่นจำลอง 3 มิติสามารถทำได้โดยคลิกเมาส์ปุ่มขวาในโปรแกรม ถ้าคลิกเมาส์ในขณะที่แขนหุบลงอยู่ ก็จะเป็นการทำให้แขนกางออก แต่ถ้าคลิกเมาส์ในขณะที่แขนกางออกอยู่ก็จะเป็นการทำให้แขนหุบลง

เมื่อสั่งให้ขยับแขนของหุ่นจำลอง 3 มิติจะได้ผลลัพธ์ดังที่แสดงในรูป

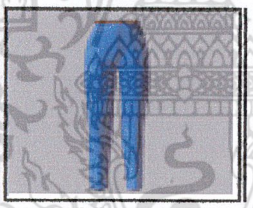




รูปที่ 7-15 แสดงการขยับแขนของหุ่นจำลอง 3 มิติ

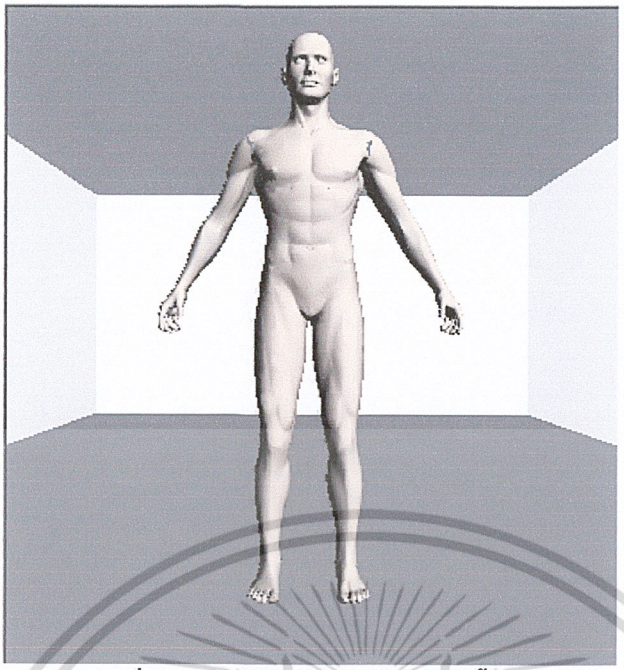
7.5 การใส่เสื้อผ้า

การใส่เสื้อผ้านั้นหลังจากที่เราได้ทำการเลือกเสื้อผ้าที่จะใส่เรียบร้อยแล้วนั้น ก็ทำการกดไปที่ปุ่มเสื้อผ้านั้นซึ่งมีลักษณะดังรูป



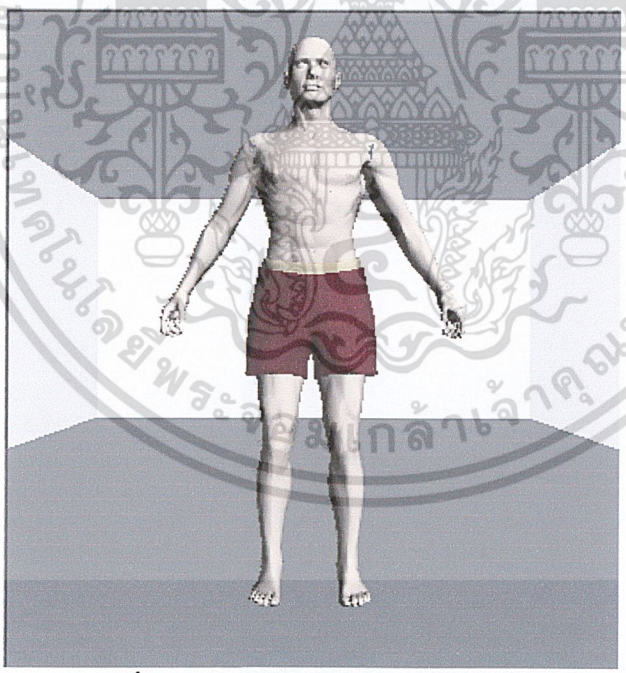
รูปที่ 7-16 แสดงปุ่มที่ใช้ในการกดเพื่อสวมใส่เสื้อผ้า

เมื่อทำการกดปุ่มเลือกแล้วจะได้ผลลัพธ์ดังที่แสดงในรูปที่



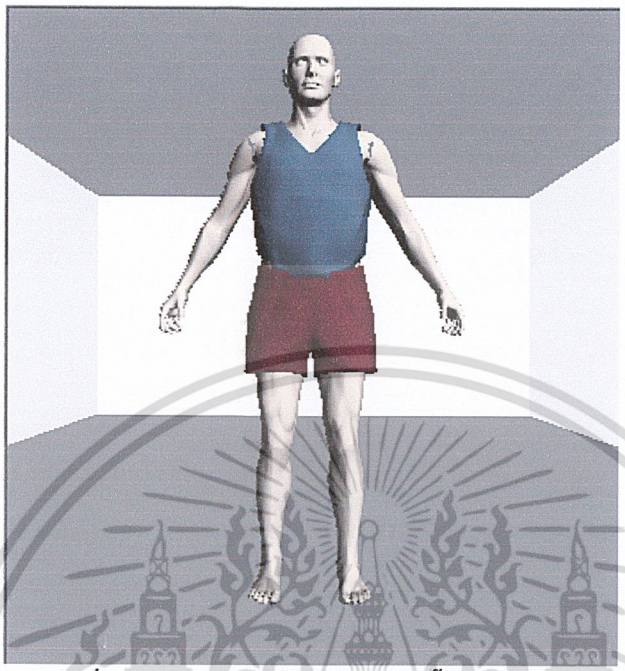
รูปที่ 7-17 แสดงผลก่อนการสวมใส่เสื้อผ้า

เมื่อทำการกดปุ่มเลือกใส่กางเกงแล้วจะได้ผลลัพธ์ดังที่แสดงในรูปที่



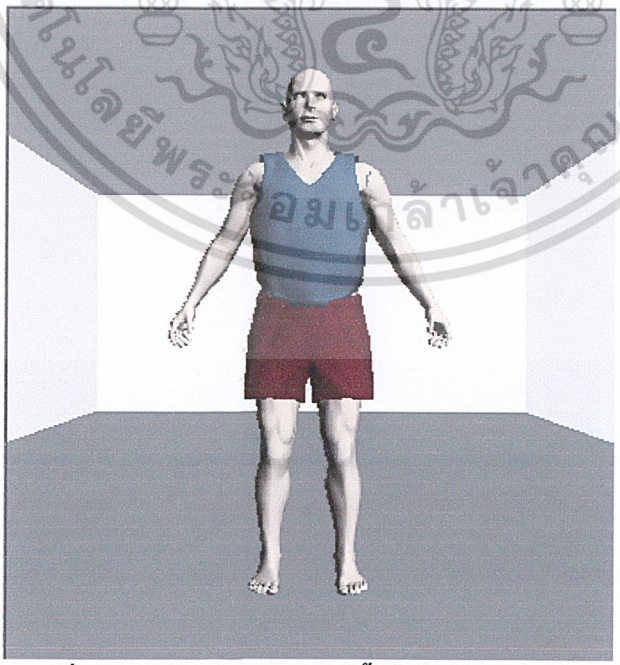
รูปที่ 7-18 แสดงผลหลังการสวมใส่กางเกง

เมื่อทำการกดปุ่มเลือกใส่เสื้อแล้วจะได้ผลลัพธ์ดังที่แสดงในรูปที่



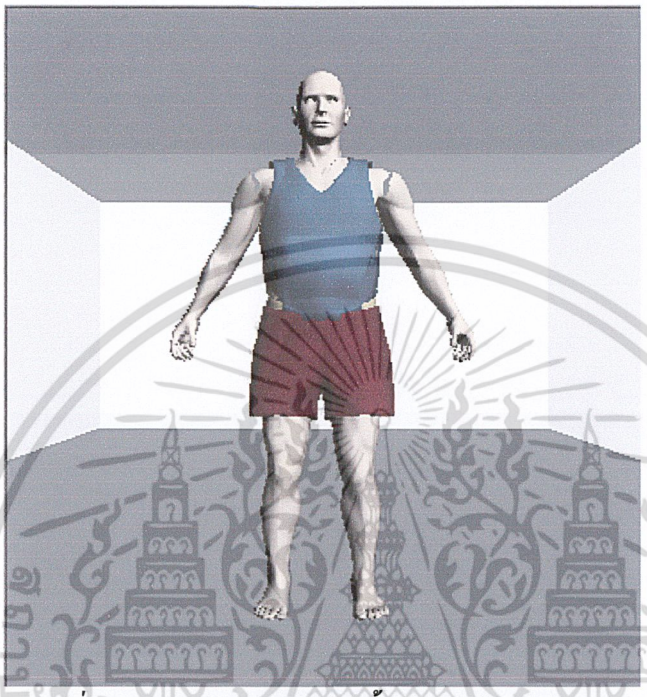
รูปที่ 7-19 แสดงผลหลังการสวมใส่เสื้อและกางเกง

สามารถปรับความสูงของหุ่นจำลอง 3 มิติได้ในขณะที่สวมใส่เสื้อผ้าอยู่ โดยเสื้อที่สวมใส่ก็ยังคงสวมใส่อยู่บนตัวหุ่นจำลอง 3 มิติอยู่

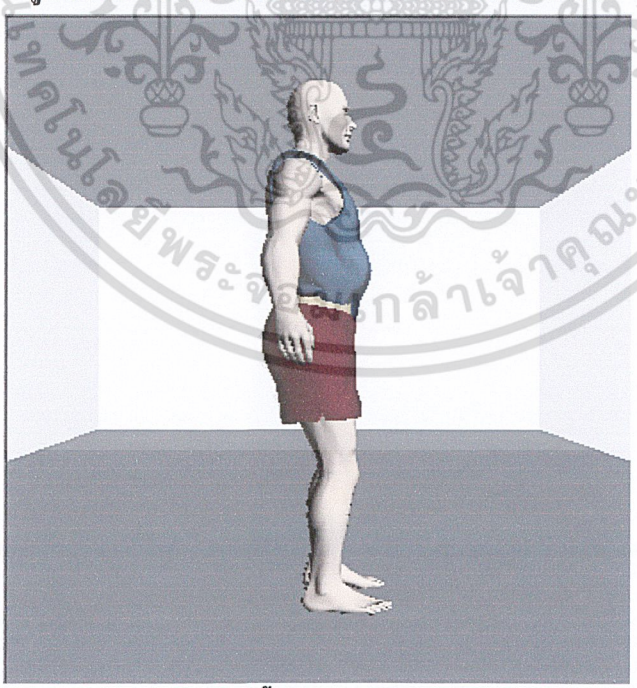


รูปที่ 7-20 แสดงผลการสวมใส่เสื้อผ้าและปรับความสูง

สามารถปรับสัดส่วนของหุ่นจำลอง 3 มิติได้เช่นกัน โดยที่เสื้อผ้าก็จะถูกปรับให้ได้สัดส่วนที่ดูเสมือนจริงด้วย เช่นถ้าส่วนเอวของหุ่นจำลอง 3 มิติถูกขยายให้ใหญ่ขึ้น เสื้อตรงส่วนของหน้าท้องก็จะพองออกเพื่อให้ได้สัดส่วนกับขนาดตัวที่ใหญ่ขึ้น



รูปที่ 7-21 แสดงผลการสวมใส่เสื้อผ้าและปรับขนาดตัว



รูปที่ 7-22 แสดงผลการสวมใส่เสื้อผ้าและปรับขนาดตัวจากมุมมองด้านข้าง

บทที่ 8
บทสรุป

8.1 บทสรุป

ตัวโปรแกรมประยุกต์สามารถ สร้างตัวหุ่นจำลอง 3 มิติโดยที่สามารถทำการเลือกเพศของตัวหุ่นจำลอง 3 มิติได้ ทำการปรับสัดส่วนต่างๆของตัวหุ่นจำลอง 3 มิติได้โดยการใส่ค่าลงในโปรแกรมประยุกต์สามารถทำการปรับสีผิวได้ และ สามารถทำการใส่เสื้อผ้าได้ โดยที่เสื้อผ้านั้นมีลวดลายและรูปแบบที่หลากหลายให้เลือก นั่นก็คือสามารถทำตามรูปแบบของโปรแกรมที่ออกแบบไว้ได้ทั้งหมด

แต่ถึงการทำงานจะเสร็จสมบูรณ์ตมที่คาดไว้นั้นแต่ก็ยังมีปัญหาที่ควรคำนึงถึงคือ ความล่าช้าในการเปิดโปรแกรม เนื่องจากต้องนำ .x file มาใช้ในโปรแกรมเป็นจำนวนมาก จึงทำให้โปรแกรมในส่วนของ Initialize นั้นทำการจองพื้นที่หน่วยความจำเป็นจำนวนมาก จึงต้องเสียเวลานานในการเริ่มต้นการใช้งานโปรแกรม

8.2 ปัญหาและอุปสรรค

- ในการสร้างโปรแกรมที่ทำการจำลองสัดส่วนของผู้คนจริงๆเข้าไปนั้น เป็นเรื่องที่ละเอียดอ่อนมาก เนื่องจากสัดส่วนของแต่ละคนไม่เท่ากัน การที่จะทำให้เหมือนโดยสมบูรณ์นั้นเป็นเรื่องที่กระทำได้ยาก และเนื่องจากการปรับขนาดโดยการทำการ Scale และ Translate ไฟล์ .x ที่ทำการสร้างไว้ก่อนแล้ว จึงทำให้ไม่สามารถปรับขนาดสัดส่วนได้สมจริงมาก ในบางผู้ใช้ เช่นในผู้ใช้ที่ขนาดตัวที่อ้วนมากก็ หรือ ผอมมากก็จะไม่ได้สัดส่วนที่สมจริงมากนัก

- การเปลี่ยนแปลงรูปแบบเสื้อผ้าหรือเพิ่มรูปแบบเสื้อผ้านั้น จำเป็นต้องทำการปรับปรุงในส่วนของโปรแกรมใหม่ทำให้ เกิดความไม่สะดวกเวลาจะเพิ่มเสื้อผ้าเข้าไปนั้นในโปรแกรม

8.3 แนวทางในการพัฒนาต่อ

1. การเพิ่ม ส่วนของการใส่หน้าตาและทรงผมของผู้ใช้เข้าไปในหุ่นจำลอง 3 มิติเพื่อให้ผู้ใช้สามารถเห็นภาพในการลองซื้อได้ชัดเจนมากขึ้น
2. การเปลี่ยนฉากในห้องลองเสื้อเป็นรูปแบบต่างๆ เช่น เป็นทะเล ในเมือง หรือบนหิมะเพื่อให้ผู้ใช้ได้เห็นภาพได้ชัดเจนขึ้นไปอีก
3. การเพิ่ม interaction ให้กับตัวโปรแกรม เช่น กดที่แขนแล้วหุ่นสามารถขับแขน กดที่ขาหุ่นสามารถขยับขาได้ หรือกดที่ตัวสามารถบิดตัวได้
4. ปรับปรุงให้สามารถเพิ่มลวดลาย ของเสื้อเข้าไปได้โดยไม่ต้องปรับปรุงในส่วนของโปรแกรม

บรรณานุกรม

[1] **ปิยะบุตร สุทธิดารา :** "3ds max 6 basic",DIGI ART Infopress Graphic Book ,2547

[2] **นิรุธ อำนวยศิลป์ :** "เขียนเกมอย่างมืออาชีพด้วย Visual C++ และ DirectX",DEV BOOK Infopress Developer Book ,2545

[3] **ยุทธนา ลีลาศวัฒนกุล :** "Visual C++ 6.0" info press, 2544

[4] **Frank D. Luna :** "Introduction to 3D Game Programming with DirectX 9.0" ,Wordware Publishing, Inc.

[5] **Peter Walsh :** "Advanced 3D Game Programming with DirectX 9.0" ,Wordware Publishing

[6] **Curious Lab :** "POSER5 Manual Guide" ,Curious Lab

[7] http://www.cwinapp.com/tutorials/011.asp

[8] http://msdn.microsoft.com/

[9] http://www.primidi.com/2004/01/28.html#a721

[10] http://www.dressingsim.com/DFL_en/product/haoreba/index.html

[11] http://rnainc.jp/index2.html

[12] http://www.curiouslabs.com/go/tutorials/

[13] http://www.curiouslabs.com/article/archive/318/