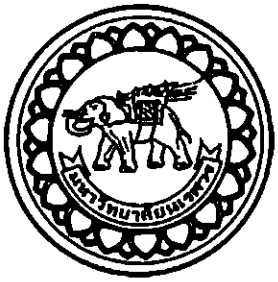


เกมแข่งรถจักรยานยนต์ผาดโผนเอ็กไซต์ไบค์แบบหลายผู้เล่นบนโทรศัพท์มือถือระบบปฏิบัติการแอนดรอยด์

MULTIPLAYER EXCITEBIKE MOBILE RACING GAME ON ANDROID

นายชิตพล เพ็ชรัตน์ รหัส 49360419

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร
ปีการศึกษา 2556

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อโครงงาน	**เกมแข่งรถจักรยานยนต์ผาดโผนเอ็กไซต์ไบค์แบบหลายผู้เล่น บนโทรศัพท์มือถือระบบปฏิบัติการแอนดรอยด์**
ผู้ดำเนินโครงงาน	นายชิตพล เพ็ชรัตน์ รหัส 49360419
ที่ปรึกษาโครงงาน	ผศ.ดร. พนมขวัญ ริยะมงคล
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2556

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

..... ที่ปรึกษาโครงงาน

(ผศ.ดร. พนมขวัญ ริยะมงคล)

..... กรรมการ

(อาจารย์ภาณุพงศ์ สอนคม)

..... กรรมการ

(อาจารย์เศรษฐา ตั้งค้าวานิช)

ชื่อหัวข้อโครงงาน	**เกมแข่งรถจักรยานยนต์ผาดโผนเอ็กไซต์ไบค์แบบหลายผู้เล่นบนโทรศัพท์มือถือระบบปฏิบัติการแอนดรอยด์**
ผู้ดำเนินโครงงาน	**นายชิตพล เพ็ชรัตน์ รหัส 49360419**
ที่ปรึกษาโครงงาน	**ผศ.ดร. พนมขวัญ ริยะมงคล**
สาขาวิชา	**วิศวกรรมคอมพิวเตอร์**
ภาควิชา	**วิศวกรรมไฟฟ้าและคอมพิวเตอร์**
ปีการศึกษา	2556

.....

บทคัดย่อ

การศึกษาการทำงานและออกแบบพัฒนาโปรแกรมการใช้งานของโทรศัพท์มือถือบนระบบปฏิบัติการแอนดรอยด์ (Android Operating System) ประกอบกับการนำเทคโนโลยีระบบเครือข่ายไร้สาย (Wireless LAN) ที่มีในโทรศัพท์มือถือนั้น ๆ มาช่วยพัฒนาเกมบนโทรศัพท์มือถือเพื่อให้สามารถใช้งานได้พร้อมกันหลายเครื่อง เกมที่ได้พัฒนาขึ้นคือ เกมแข่งรถจักรยานยนต์ผาดโผนเอ็กไซต์ไบค์ซึ่งสามารถเล่นได้พร้อมกันไม่เกิน 4 คน

Project titel	Multiplayer Excitebike Mobile Racing Game On Android
Name	Mr. Chitpon Pedrat ID. 49360419
Project advisor	ASST. PROF. DR. Panomkhawn Riyamongkol
Major	Computer Engineering
Department	Electrical and Computer Engineering
Academic year	2556

.....

Abstract

Today the use of cellphones is considered important for daily living. The cellphones are therefore developed for more functions, including the widely-used cellphones with Android operating system. This project aims to study the functioning system of the cellphones and to develop the functioning program for the cellphones with Android operating system so that the users can maximize its usage. The technology of Wireless LAN system in the cellphones is then applied for developing the games on the cellphones in order to enhance the entertainment among users.

กิตติกรรมประกาศ

การจัดทำปริญญานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยการได้รับคำแนะนำ และความช่วยเหลือจากอาจารย์พนมขวัญ ริยะมงคล และคณาจารย์ทุกท่านที่ประสิทธ์ประสาทวิชาความรู้ในด้านต่างๆ จนทำให้ปริญญานิพนธ์ฉบับนี้สำเร็จเป็นรูปเป็นร่างขึ้นมาได้

ดังนั้นผู้จัดทำจึงขอขอบคุณอาจารย์ทุกท่านที่ให้ความรู้ และความช่วยเหลือ ทำให้ปริญญานิพนธ์ฉบับนี้สำเร็จลงด้วยดี เหนือสิ่งอื่นใด ขอกราบขอบพระคุณพ่อกับแม่ที่คอยดูแลลูกส่งเสียให้ลูกได้เรียน และสั่งสอนลูกให้ลูกเป็นคนดีและให้คำปรึกษาทุกครั้งที่มีปัญหา จนทำให้ลูกมีวันนี้ได้ และขอบคุณเพื่อนๆ ที่คอยช่วยเหลือกัน และคอยเป็นกำลังใจให้เสมอมา

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญภาพ

ภาพที่ หน้า

สารบัญ (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

ในปัจจุบันโทรศัพท์มือถือได้เข้ามามีบทบาทในชีวิตประจำวันอย่างมาก ซึ่งนอกจากในด้านการติดต่อสื่อสารแล้ว เรายังใช้โทรศัพท์เคลื่อนที่ในด้านอื่น ๆ เช่น ถ่ายรูป ฟังเพลง เล่นอินเตอร์เน็ต รวมถึงการเล่นเกม และนอกจากจะเล่นเกมคนเดียวแล้ว เราสามารถที่จะเล่นเกมกับคนอื่น ๆ ได้ด้วย ทางผู้จัดทำโครงงาน จึงได้เล็งเห็นถึงความน่าสนใจที่จะพัฒนาเกมบนโทรศัพท์มือถือที่สามารถเล่นได้ตั้งแต่ 2 คนขึ้นไปโดยเชื่อมต่อกันผ่านทางระบบเครือข่ายไร้สาย (Wireless LAN) เพื่อเพิ่มความบันเทิงและความสนุกสนานให้แก่ผู้ใช้งานโทรศัพท์มือถือมากยิ่งขึ้นโดยเกมที่จัดทำขึ้นนี้ ทำขึ้นเลียนแบบเกม Excitebike จากเครื่องเล่นเกม Famicom ซึ่งเป็นลิขสิทธิ์ของบริษัท Nintendo และจัดทำขึ้นมาเพื่อการศึกษาเท่านั้น

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาวิธีการสร้างเกม (Game) แบบหลายผู้เล่นบนโทรศัพท์มือถือระบบปฏิบัติการแอนดรอยด์ (Android Operating System)
2. เพื่อศึกษาการติดต่อสื่อสารโดยใช้เทคโนโลยีระบบเครือข่ายไร้สาย (Wireless LAN) บนโทรศัพท์มือถือระบบปฏิบัติการแอนดรอยด์ (Android Operating System)
3. เพื่อประยุกต์ใช้งานภาษาจาวา (Java) บนโทรศัพท์มือถือ
4. เพื่อสร้างความบันเทิงและความสนุกสนานให้กับผู้ใช้งานโทรศัพท์มือถือ

1.3 ขอบเขตของโครงงาน

สร้างเกม (Game) แบบหลายผู้เล่นบนโทรศัพท์มือถือระบบปฏิบัติการแอนดรอยด์ (Android) ที่สามารถติดต่อสื่อสารผ่านทางเทคโนโลยีระบบเครือข่ายไร้สาย (Wireless LAN) โดยใช้ภาษาจาวา(Java)

1.4 ข้อจำกัดของโครงงาน

โปรแกรมนี้สามารถใช้งานบนโทรศัพท์มือถือระบบปฏิบัติการแอนดรอยด์ (Android Operating System) ตั้งแต่รุ่น 2.2 ขึ้นไปที่มีหน้าจอขนาด 320x240, 480x320, 800x480 พิกเซล (Pixel)

1.5 ผลที่คาดว่าจะได้รับ

1. ได้เข้าใจวิธีการสร้างเกมบนโทรศัพท์มือถือระบบปฏิบัติการแอนดรอยด์ (Android Operating System)
2. ได้เข้าใจถึงการเขียนโปรแกรมใช้งานซ็อกเก็ต (Socket Programming) บนโทรศัพท์มือถือระบบปฏิบัติการแอนดรอยด์ (Android Operating System)
3. เข้าใจการวาดภาพและการสร้างกราฟิกต่าง ๆ

บทที่ 2
หลักการและทฤษฎีที่เกี่ยวข้อง

2.1 แอนดรอยด์ (Android)

แอนดรอยด์ (Android) หรือ ระบบปฏิบัติการแอนดรอยด์ (Android Operating System) เป็นชื่อเรียกชุดซอฟท์แวร์ หรือแพลตฟอร์ม (Platform) สำหรับอุปกรณ์อีเล็กทรอนิกส์ ที่มีหน่วยประมวลผลเป็นส่วนประกอบ อาทิเช่น คอมพิวเตอร์, โทรศัพท์ (Telephone), โทรศัพท์เคลื่อนที่ (Cell phone), อุปกรณ์เล่นอินเตอร์เน็ตขนาดพกพา (MID) เป็นต้น แอนดรอยด์ (Android) นั้น ถือกำเนิดอย่างเป็นทางการในวันที่ 5 พฤศจิกายน 2550 โดยบริษัท กูเกิล จุดประสงค์ของแอนดรอยด์ (Android) นั้น มีจุดเริ่มต้นมาจากบริษัท Android Inc. ที่ได้นำเอาระบบปฏิบัติการลีนุกซ์ (Linux) ซึ่งนิยมนำไปใช้งานกับเครื่องแม่ข่าย (Server) เป็นหลัก นำมาลดทอนขนาดตัว (แต่ไม่ลดทอนความสามารถ) เพื่อให้เหมาะสมแก่การนำไปติดตั้งบนอุปกรณ์พกพา ที่มีขนาดพื้นที่จัดเก็บข้อมูลที่จำกัด โดยหวังว่า แอนดรอยด์ (Android) นั้นจะเป็นหุ่นยนต์ตัวน้อย ๆ ที่คอยช่วยเหลืออำนวยความสะดวกแก่ผู้ที่พกพามัน ไปในทุกที่ ทุกเวลา

แอนดรอยด์เป็นซอฟต์แวร์ที่มีโครงสร้างแบบเรียงทับซ้อนหรือแบบสแต็ก (Stack) ซึ่งรวมเอาระบบปฏิบัติการ (Operating System), มิดเดิลแวร์ (Middleware) และแอพพลิเคชั่น (Application) ที่สำคัญเข้าไว้ด้วยกัน เพื่อใช้สำหรับทำงานบนอุปกรณ์พกพาเคลื่อนที่ (Mobile Devices) เช่น โทรศัพท์มือถือ เป็นต้น การทำงานของแอนดรอยด์ (Android) มีพื้นฐานอยู่บนระบบลีนุกซ์ เคอร์เนล (Linux Kernel) ซึ่งใช้ Android SDK (Software Development Kit) เป็นเครื่องมือสำหรับการพัฒนาแอพพลิเคชั่น (Application) บนระบบปฏิบัติการแอนดรอยด์ (Android Operating System) และใช้ภาษาจาวา (Java) ในการพัฒนา

2.1.1 คุณลักษณะสำคัญที่มีในแอนดรอยด์ ได้แก่

1. แพลตฟอร์มสาหรับโปรแกรมประยุกต์ (Application Framework)
2. Dalvik virtual machine ที่ทำขึ้นเพื่ออุปกรณ์มือถือโดยเฉพาะ
3. มีโปรแกรมเบราว์เซอร์ในตัว (Webkit Engine) ซึ่งเป็นโอเพ่นซอร์ส (Open Source)
4. มีกราฟิกแบบ 2D และ 3D โดยใช้ OpenGL
5. มีระบบฐานข้อมูล
6. มีการสนับสนุนทางด้านมัลติมิเดีย ตั้งแต่เสียง ภาพ วีดีโอ (MPEG4, H.264, MP3, AAC, AMR,JPG, PNG, GIF)
7. รองรับโทรศัพท์แบบ GSM (ขึ้นกับอุปกรณ์ฮาร์ดแวร์)
8. Bluetooth, EDGE, 3G, and WiFi(ขึ้นกับอุปกรณ์ฮาร์ดแวร์)
9. Camera, GPS, compass, and accelerometer (ขึ้นกับอุปกรณ์ฮาร์ดแวร์)
10. มีเครื่องมือพัฒนาหลากหลาย ตั้งแต่โปรแกรมจำลอง (Emulator) สำหรับการทดสอบโปรแกรม
11. มีปลั๊กอิน สาหรับ Eclipse IDE

สถาปัตยกรรมของแอนดรอยด์ (Android Architecture) นั้นถูกแบ่งออกเป็นลำดับชั้นออกเป็น4ชั้นได้แก่

1. ชั้นแอพพลิเคชั่น (Application)

ชั้นนี้จะเป็นชั้นที่อยู่บนสุดของโครงสร้างสถาปัตยกรรม Android ซึ่งเป็นส่วนของแอพพลิเคชันที่พัฒนาขึ้นมาใช้งาน เช่น แอพพลิเคชั่นรับ/ส่งอีเมล์, SMS, ปฏิทิน, แผนที่, เว็บเบราเซอร์, รายชื่อผู้ติดต่อ เป็นต้น ซึ่งแอพพลิเคชั่นจะอยู่ในรูปแบบของไฟล์ .apk โดยทั่วไปแล้วจะอยู่ในไดเร็คทอรี data/app

2. ชั้นแอพพลิเคชั่นเฟรมเวิร์ค (Application Framework)

ในชั้นนี้จะอนุญาตให้นักพัฒนาสามารถเข้าเรียกใช้งาน โดยผ่าน API (Application Programming Interface) ซึ่งแอนดรอยด์ (Android) ได้ออกแบบไว้เพื่อลดความซ้ำซ้อนในการใช้งาน application component โดยในชั้นนี้ประกอบด้วยแอพพลิเคชั่นเฟรมเวิร์คดังนี้

- **View System** เป็นส่วนที่ใช้ในการควบคุมการทำงานสำหรับการสร้าง Application เช่น lists, grids, text boxes, buttons และ embeddable web browser
- **Location Manager** เป็นส่วนที่จัดการเกี่ยวกับค่าตำแหน่งของเครื่องอุปกรณ์พกพาเคลื่อนที่
- **Content Provider** เป็นส่วนที่ใช้ควบคุมการเข้าถึงข้อมูลที่มีการใช้งานร่วมกัน (Share data) ระหว่าง Application ที่แตกต่างกัน เช่น ข้อมูลผู้ติดต่อ (Contact)
- **Resource Manager** เป็นส่วนที่จัดการข้อมูลต่างๆ ที่ไม่ใช่ส่วนของโค้ดโปรแกรม เช่น รูปภาพ, localized strings, layout ซึ่งจะอยู่ในไดเร็คทอรี res/
- **Notification Manager** เป็นส่วนที่ควบคุมอีเวนต์ (Event) ต่างๆ ที่แสดงบนแถบสถานะ (Status bar) เช่น ในกรณีที่ได้รับข้อความหรือสายที่ไม่ได้รับและการแจ้งเตือนอื่นๆ เป็นต้น
- Activity Manager เป็นส่วนควบคุม Life Cycle ของ Application

3. ชั้นไลบรารี (Library)

แอนดรอยด์ (Android) ได้รวบรวมกลุ่มของไลบรารีต่างๆ ที่สำคัญและมีความจำเป็นเอาไว้มากมาย เพื่ออำนวยความสะดวกให้กับนักพัฒนาและง่ายต่อการพัฒนาโปรแกรม โดยตัวอย่าง ของไลบรารีที่สำคัญเช่น

- **System C library** เป็นกลุ่มของไลบรารีมาตรฐานที่อยู่บนพื้นฐานของภาษา C ไลบรารี (libc) สำหรับ embedded system ที่มีพื้นฐานมาจาก Linux
- **Media Libraries** เป็นกลุ่มการทำงานมัลติมีเดีย เช่น MPEG4, H.264, MP3, AAC, AMR, JPG, และ PNG
- **Surface Manager** เป็นกลุ่มการจัดการรูปแบบหน้าจอ การวาดหน้าจอ
- **2D/3D library** เป็นกลุ่มของกราฟิกแบบ 2 มิติ หรือ SGL (Scalable Graphics Library) และแบบ 3 มิติ หรือ OpenGL
- **FreeType** เป็นกลุ่มของบิตแมป (Bitmap) และเวคเตอร์ (Vector) สำหรับการเรนเดอร์ (Render) ภาพ
- **SQLite** เป็นกลุ่มของฐานข้อมูล โดยนักพัฒนาสามารถใช้ฐานข้อมูลนี้เก็บข้อมูล Application ต่างๆ ได้
- **Browser Engine** เป็นกลุ่มของการแสดงผลบนเว็บเบราเซอร์โดยอยู่บนพื้นฐานของ Webkit ซึ่งจะมีลักษณะคล้ายกับ Google Chrome

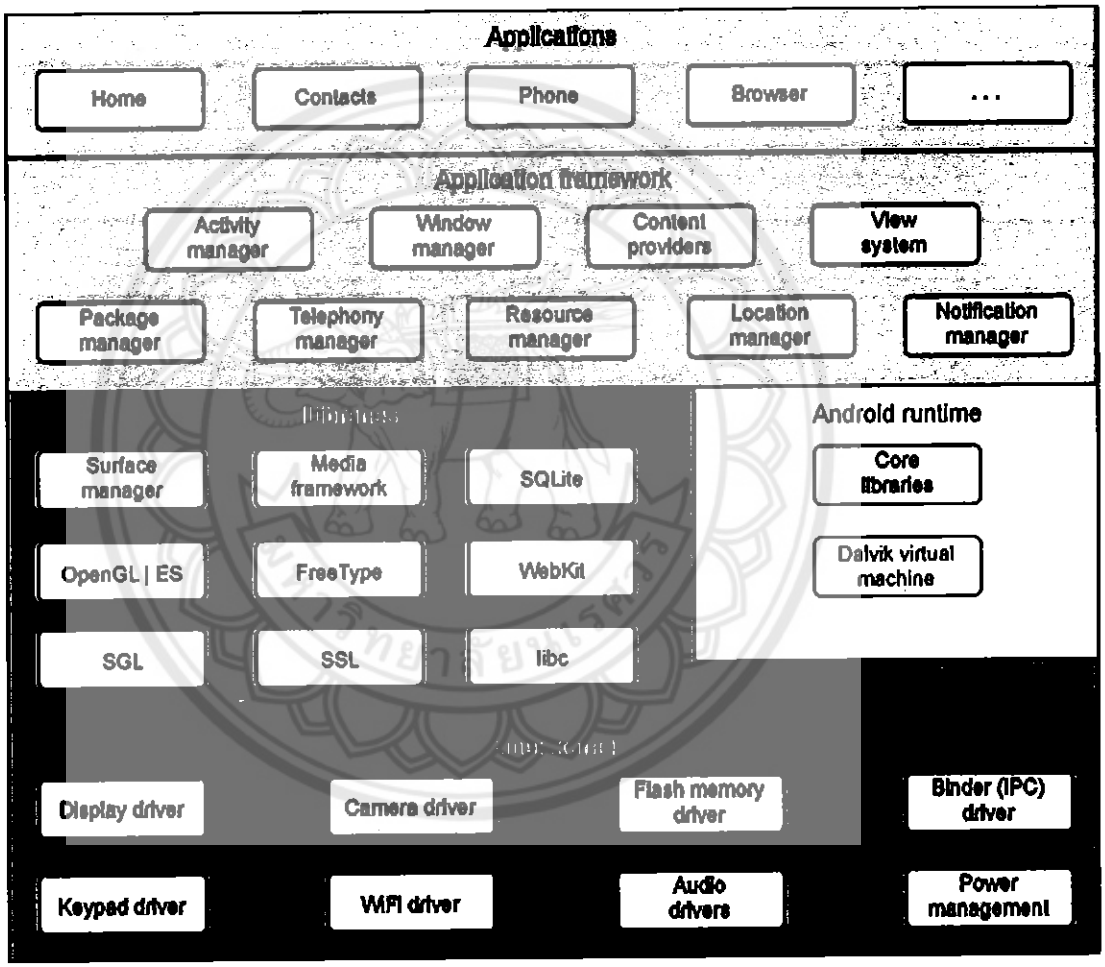
3.1 Android Runtime

เป็นชั้นย่อยที่อยู่ในชั้นไลบรารี ซึ่งจะประกอบด้วย 2 ส่วนหลักคือ

- **Dalvik VM (Virtual Machine)** ส่วนนี้ถูกเขียนด้วยภาษา Java เพื่อใช้เฉพาะการใช้งานในอุปกรณ์เคลื่อนที่ Dalvik VM จะแตกต่างจาก Java VM (Virtual Machine) คือ Dalvik VM จะรันไฟล์ .dex ที่คอมไพล์มาจากไฟล์ .class และ .jar โดยมี tool ที่ชื่อว่า dx ทำหน้าที่ในการบีบอัดคลาส Java ทั้งนี้ไฟล์ .dex จะมีขนาดกะทัดรัดและเหมาะสมกับอุปกรณ์เคลื่อนที่มากกว่า .class เพื่อต้องการใช้พลังงานจากแบตเตอรี่อย่างมีประสิทธิภาพสูงสุด
- **Core Java Library** ส่วนนี้เป็นไลบรารีมาตรฐาน แต่ก็มีความแตกต่างจากไลบรารีของ Java SE (Java Standard Edition) และ Java ME (Java Mobile Edition)

4. ชั้นลีนุกซ์เคอร์เนล (Linux Kernel)

ระบบ แอนดรอยด์ (Android) นั้นถูกสร้างบนพื้นฐานของระบบปฏิบัติการลีนุกซ์ (Linux) โดยในชั้นนี้จะมีฟังก์ชันการทำงานหลายๆ ส่วน แต่โดยส่วนมากแล้วจะเกี่ยวข้องกับฮาร์ดแวร์โดยตรงเช่น การจัดการหน่วยความจำ (Memory Management) การจัดการโพรเซส (Process Management) การเชื่อมต่อเครือข่าย (Networking) เป็นต้น



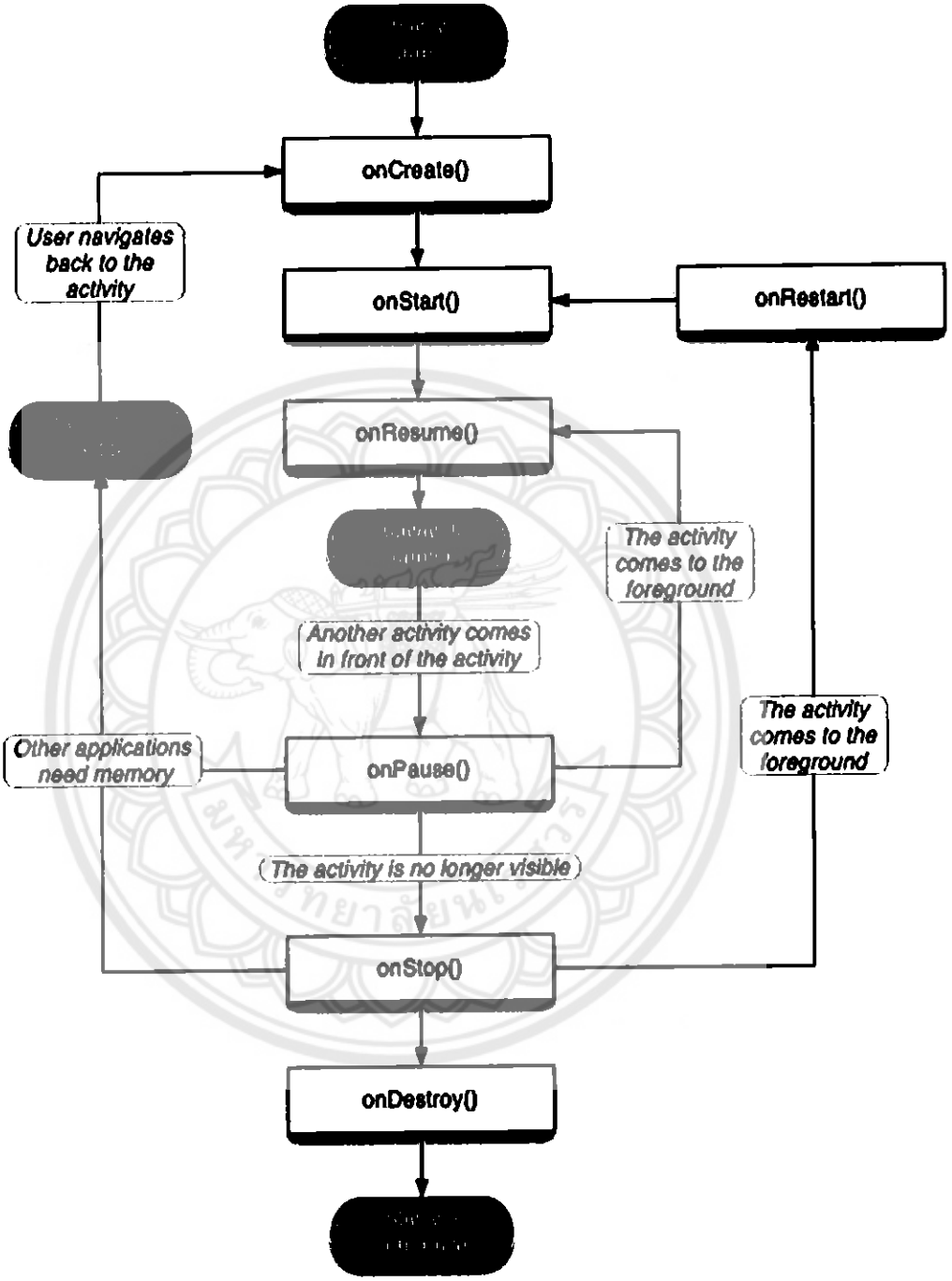
ภาพที่ 2-1 สถาปัตยกรรมแอนดรอยด์ (Android)

2.2 การพัฒนาแอนดรอยด์แอพพลิเคชั่น(Android Application)

Activity คือแอพพลิเคชั่นคอมโพเน็น (Application Component) ที่ทำหน้าที่ควบคุมการแสดงผลสื่อประสานกับผู้ใช้ (User Interface) รวมถึงควบคุมการมีปฏิสัมพันธ์ระหว่างผู้ใช้กับสื่อประสานกับผู้ใช้ (User Interface) ด้วย เช่น การโทรออก, การถ่ายรูป, การส่งอีเมล์ เป็นต้น

โดยปกติแล้ว ในแอพพลิเคชั่น (Application) หนึ่ง ๆ จะมี Activity มากกว่าหนึ่ง Activity เสมอ ซึ่งแต่ละ Activity ก็จะมีหน้าที่ที่แตกต่างกันไป เช่น Activity สำหรับควบคุมหน้าจอแสดงรายชื่ออีเมล์, Activity สำหรับควบคุมหน้าจอแบบฟอร์มเพื่อส่งอีเมล์ เป็นต้น และในแต่ละ Activity ใด ๆ ก็สามารถสั่งให้ Activity อื่น ๆ ทำงานได้เสมอ

อย่างไรก็ตาม ในช่วงเวลาหนึ่ง ๆ จะมีเพียง Activity เดียวเท่านั้นที่พร้อมมีปฏิสัมพันธ์กับผู้ใช้ นั่นหมายถึงเมื่อมี Activity ใด ๆ เริ่มทำงานขึ้นมาใหม่ จะทำให้ Activity เดิมที่กำลังทำงานอยู่เปลี่ยนสเตจ (State) ของตนเองไป ซึ่งสเตจ (State) หลัก ๆ ของ Activity ได้แก่ Create, Start, Resume, Pause, Stop, และ Destroy ทั้งนี้เพื่อควบคุมการทำงานของ Activity ให้เหมาะสมกับ State ที่เปลี่ยนไปนั้น จะต้องเขียนคำสั่งไว้ภายใน LifeCycle Callback Method ต่าง ๆ เพื่อควบคุมการทำงานของแต่ละ State นั้น ๆ เช่น onCreate ( ) เป็น LifeCycle Callback Method ที่จะถูกทำงานเมื่อ Activity เปลี่ยน State มาเป็น Create, onStart ( ) เป็น LifeCycle Callback Method ที่จะถูกทำงานเมื่อ Activity เปลี่ยนสเตจ (State) มาเป็น Start เป็นต้น



ภาพที่ 2-2 การทำงานของแอนดรอยด์แอพพลิเคชั่น (Android application life cycle)

การสร้าง Activity ทำได้โดยการสร้างคราส (Class) และให้สืบทอดจากคราส (Class) Activity หรือสืบทอดจากคราส (Class) ใดๆ ก็ตามที่ได้รับการสืบทอดมาจากคราส (Class) Activity

นอกจากนี้ให้ทำการ Override LifeCycle Callback Method ต่าง ๆ เพื่อควบคุมการทำงานให้เหมาะสมกับ State ที่เปลี่ยนแปลงไปมาของ Activity ด้วย ซึ่ง LifeCycle Callback Method หลัก ๆ ได้แก่ onCreate ( ), onStart ( ), onResume ( ), onPause ( ), onStop ( ), และ onDestroy ( )

ตัวอย่างการสร้างคราส (Class) Activity

```
public class NuttdotmeActivity extends Activity{
 public void onCreate ( Bundle savedInstanceState ){
  super.onCreate ( savedInstanceState );
 }
 protected void onStart ( ){
  super.onStart ( );
 }
 protected void onResume ( ){
  super.onResume ( );
 }
 protected void onPause ( ){
  super.onPause ( );
 }
 protected void onStop ( ){
  super.onStop ( );
 }
 protected void onDestroy ( ){
  super.onDestroy ( );
 }
}
```

สื่อประสานกับผู้ใช้ (User Interface) ในแอนดรอยด์ (Android) มีโครงสร้างลักษณะเป็นแบบ Hierachy ซึ่งประกอบด้วยออบเจ็ค (Object) 2 ประเภท คือ View และ ViewGroup โดย ViewGroup เป็นออบเจ็ค (Object) ที่คอยทำหน้าที่เป็นเลเอ้าท์ (Layout) ของ View เช่น LinearLayout, RelativeLayout, GridLayout เป็นต้น และ View เป็นออบเจ็ค (Object) ที่มองเห็นได้และสามารถมีปฏิสัมพันธ์กับผู้ใช้ได้ เช่น Button, TextView, ImageView เป็นต้น ทั้งนี้จากโครงสร้างของสื่อประสานกับผู้ใช้ (User Interface) ที่มีลักษณะเป็นแบบ Hierachy นั้น สามารถกล่าวได้ว่า Node ใด ๆ ที่เป็น Parent Node จะเป็นออบเจ็ค (Object) ประเภท ViewGroup เสมอ และ Node ใด ๆ ที่เป็น Leaf Node จะเป็นออบเจ็ค (Object) ประเภท View เสมอ

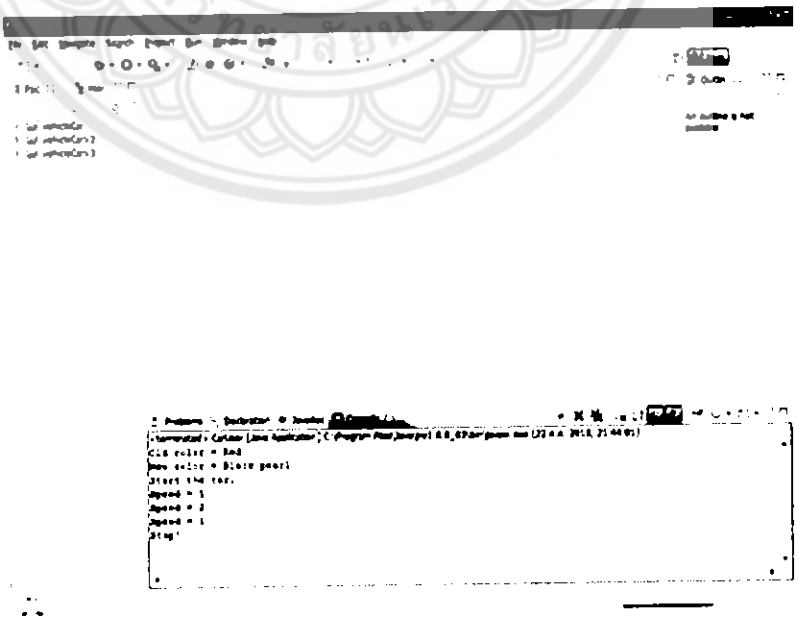
ในส่วนของการสร้างสื่อประสานกับผู้ใช้ (User Interface) ใน Activity นั้น สามารถทำได้ 2 วิธี วิธีแรกคือการสร้างไว้ใน XML File ซึ่งข้อดีคือเป็นการแยกส่วนที่เป็นสื่อประสานกับผู้ใช้ (User Interface) ออกจากพฤติกรรมของระบบอย่างชัดเจน และอีกวิธีหนึ่งคือการเขียนคำสั่งในการสร้างสื่อประสานกับผู้ใช้ (User Interface) ไว้ภายใน Activity เลย ทั้งนี้โดยส่วนมากแล้วนิยมใช้ 2 วิธีข้างต้นร่วมกันในการสร้างสื่อประสานกับผู้ใช้ (User Interface) โดยโครงสร้างหลักของสื่อประสานกับผู้ใช้ (User Interface) จะถูกเขียนไว้ใน XML File และส่วนใดที่ต้องการให้เกิดการเปลี่ยนแปลงในขณะ Run Time ก็จะเขียนคำสั่งไว้ภายใน Activity

2.3 ซอฟต์แวร์ที่เกี่ยวข้อง

2.3.1 Eclipse คือโปรแกรมที่ใช้สำหรับพัฒนาภาษาจาวา (Java) ซึ่งโปรแกรม Eclipse เป็นโปรแกรมหนึ่งที่ใช้ในการพัฒนาแอพพลิเคชั่นเซิร์ฟเวอร์ (Application Server) ได้อย่างมีประสิทธิภาพ

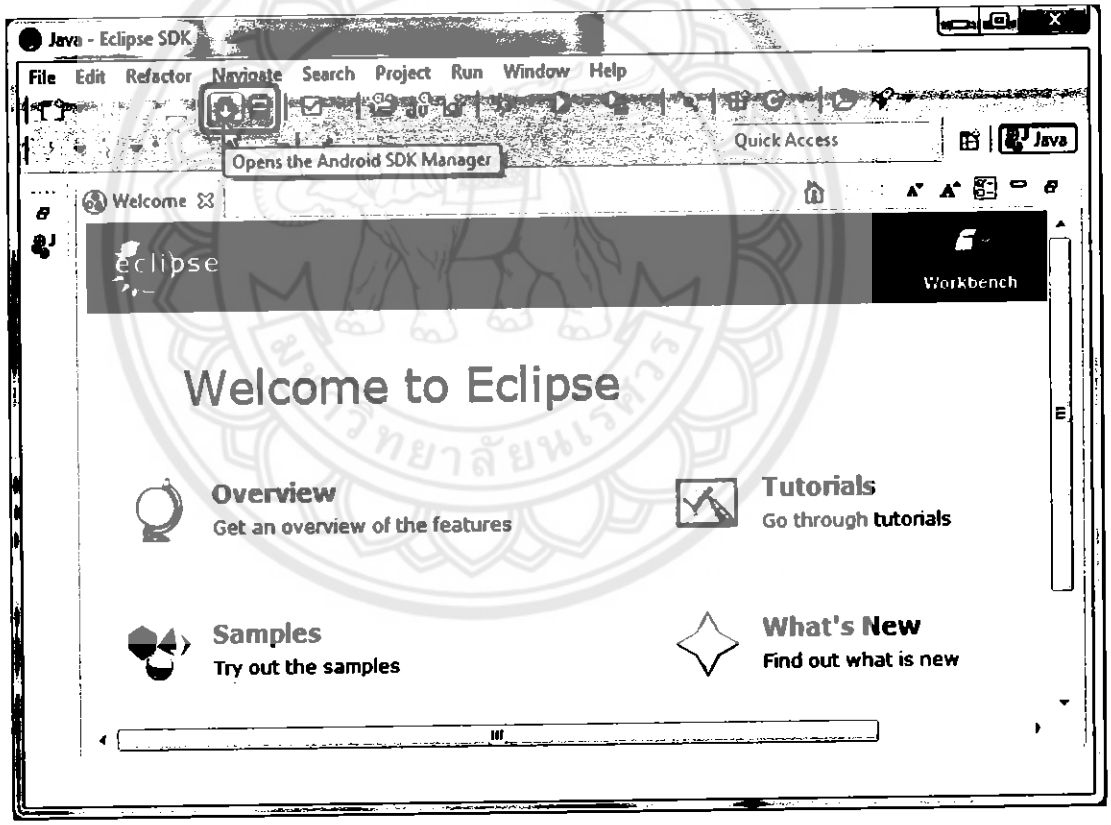
Eclipse มีองค์ประกอบหลักที่เรียกว่า Eclipse Platform ซึ่งให้บริการพื้นฐานหลักสำหรับรวบรวมเครื่องมือต่างๆจากภายนอกให้สามารถเข้ามาทำงานร่วมกันในสภาพแวดล้อมเดียวกัน และมีองค์ประกอบที่เรียกว่า Plug-in Development Environment (PDE) ซึ่งใช้ในการเพิ่มความสามารถในการพัฒนาซอฟต์แวร์มากขึ้น เครื่องมือภายนอกจะถูกพัฒนาในรูปแบบที่เรียกว่า Eclipse plug-ins ดังนั้นหากต้องการให้ Eclipse ทำงานใดเพิ่มเติม ก็เพียงแต่พัฒนา plugin สำหรับงานนั้นขึ้นมาและนำปลั๊กอิน (Plug-in) นั้นมาติดตั้งเพิ่มเติมให้กับ Eclipse ที่มีอยู่เท่านั้น Eclipse Plug-in ที่มีมาพร้อมกับ Eclipse เมื่อทำการติดตั้ง ก็คือองค์ประกอบที่เรียกว่า Java Development Toolkit (JDT) ซึ่งเป็นเครื่องมือในการเขียนและ ดีบัค (Debug) โปรแกรมภาษาจาวา (Java)

ข้อดีของโปรแกรม Eclipse คือ ติดตั้งง่าย สามารถใช้ได้กับ J2SDK ได้ทุกเวอร์ชั่น รองรับภาษาต่างประเทศอีกหลายภาษา มีปลั๊กอิน (Plug-in) ที่ใช้เสริมประสิทธิภาพของโปรแกรม สามารถทำงานได้กับไฟล์หลายชนิด เช่น HTML, Java, C, JSP, EJB, XML และ GIF และ ใช้งานได้กับระบบปฏิบัติการ วินโด (Windows),ลีนุกซ์ (Linux)และ แมค (Mac OS)



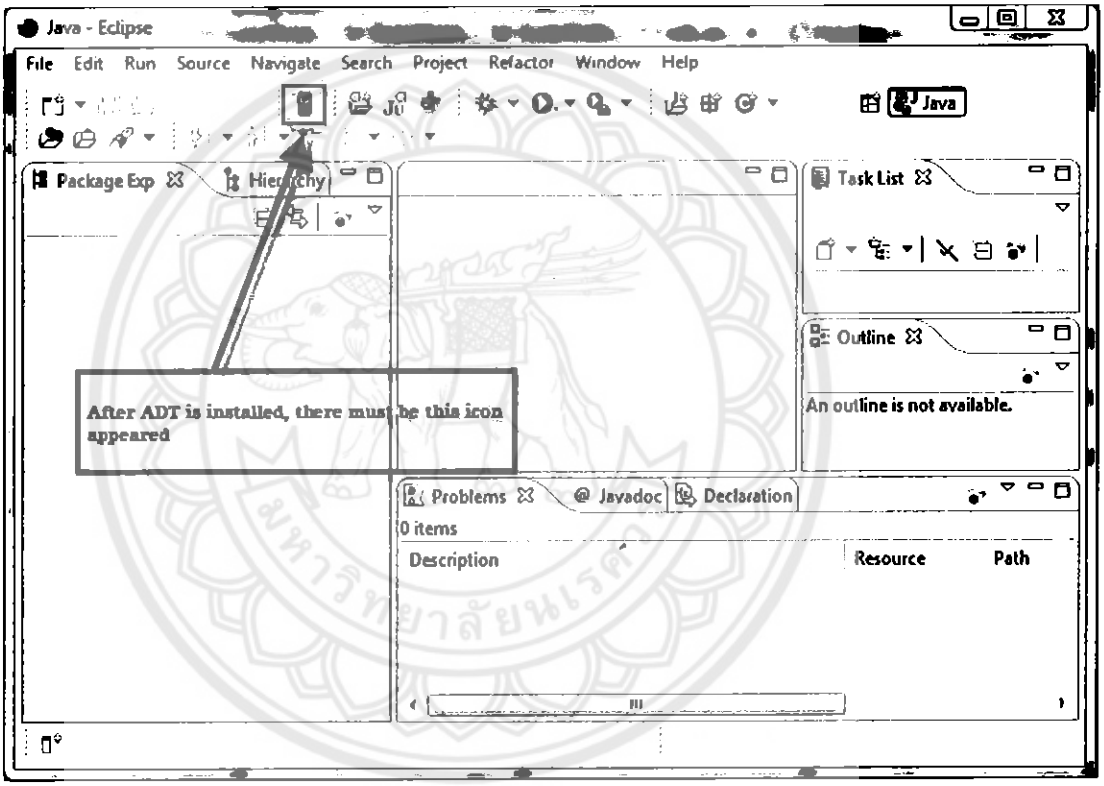
ภาพที่ 2-3 หน้าต่างโปรแกรม Eclipse

2.3.2 Android SDK ย่อมาจาก Android Software Development Kit เป็นชุดโปรแกรมที่ทางบริษัท กูเกิล พัฒนาออกมาเพื่อแจกจ่ายให้นักพัฒนาแอพพลิเคชั่น (Application) หรือผู้สนใจทั่วไปดาวน์โหลดไปใช้กันโดยไม่มีค่าใช้จ่าย ซึ่งนี่ก็เป็นหนึ่งในปัจจัยที่ทำให้แอพพลิเคชั่น (Application) บนแอนดรอยด์ (Android) นั้นเพิ่มขึ้น อย่างรวดเร็ว ซึ่งในชุด SDK นั้นจะมีโปรแกรมและไลบรารี่ (Library) ต่างๆ ที่จำเป็นต่อการพัฒนาแอพพลิเคชั่น (Application) บนแอนดรอยด์ (Android) อย่างเช่น โปรแกรมจำลอง (Emulator) ซึ่งทำให้ผู้ใช้สามารถสร้างแอพพลิเคชั่น (Application) และนำมาทดลองทดสอบบนโปรแกรมจำลอง (Emulator) ก่อน โดยมีสภาวะแวดล้อมเหมือนมือถือที่ทดสอบบนระบบปฏิบัติการแอนดรอยด์ (Android Operating System) จริง



ภาพที่ 2-4 ไอคอนแสดง plugin Android SDK

2.3.3 ADT ย่อมาจาก Android Development Tools คือ เครื่องมือที่ใช้เพื่อให้การเขียนโปรแกรมแอนดรอยด์ (Android) สะดวกสบายยิ่งขึ้น เช่นในการเริ่มโปรเจค (Project) ใหม่ ก็จะกำหนดค่าเริ่มต้น รวมทั้งโค้ด (Code) พื้นฐานมาให้เลย ร่วมถึงความสามารถในการส่งตัวแอพพลิเคชั่น (Application) ออกมาในรูปของ ไฟล์ (File) .apk ที่จะสามารถนำไปใช้ติดตั้งบนเครื่องที่ใช้แอนดรอยด์ (Android) ได้ทันที ADT เป็นปลั๊กอิน (Plugin) ของโปรแกรม Eclipse และ ADT นี้ก็รวมอยู่เป็นส่วนหนึ่งของ Android SDK

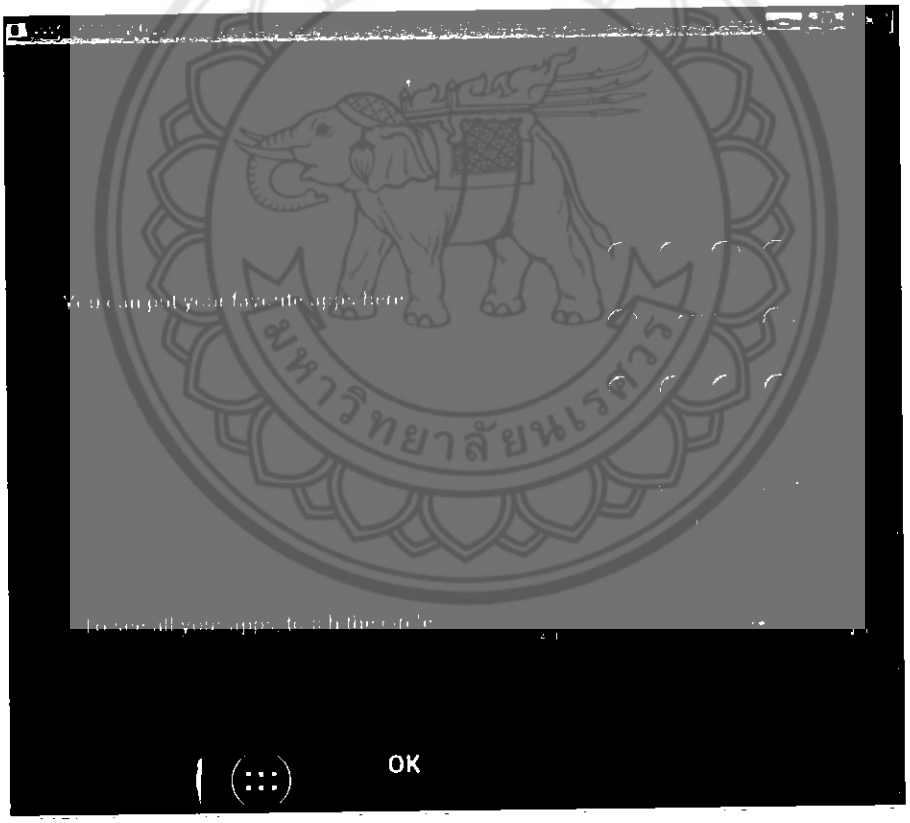


ภาพที่ 2-5 ไอคอนแสดง plugin Android Development Tools

2.2.4 AVD ย่อมาจาก Android Virtual Device คือ การจำลอง (Emulator) เครื่องโทรศัพท์มือถือระบบปฏิบัติการแอนดรอยด์ (Android Operating System) บนเครื่องคอมพิวเตอร์ เพื่อเอาไว้ทดสอบโปรแกรม หรือ โค้ด (Code) โปรแกรมที่ได้เขียนขึ้น

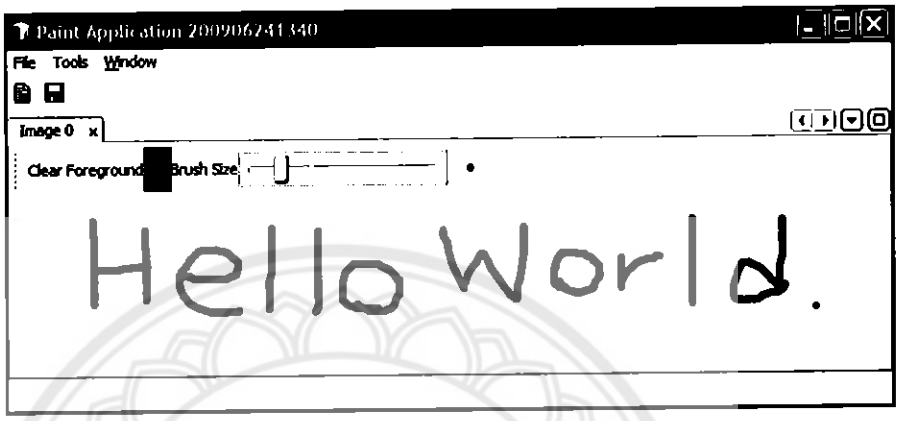
ข้อดีของ AVD

AVD นั้น เป็นโปรแกรมจำลอง (Emulator) ที่เพิ่มความสะดวกสบายในการพัฒนาแอพพลิเคชั่น (Application) สำหรับแอนดรอดย์ (Android) โดยหลังจากที่ผู้พัฒนาเขียนแอพพลิเคชั่น(Application) เสร็จแล้ว ก็สามารถส่งแอพพลิเคชั่น (Application) ไปลองทดสอบบนคอมพิวเตอร์ที่ได้ทำให้เป็นโปรแกรมจำลอง (Emulator) ดูได้เลย



ภาพที่ 2-6 ภาพแสดงตัวอย่าง AVD บนคอมพิวเตอร์

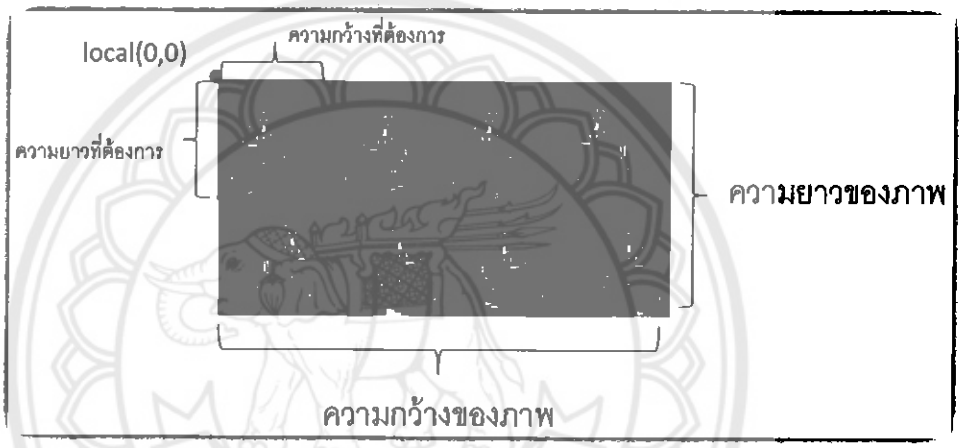
2.3.5 ภาษาจาวา (Java Programming Language) เป็นภาษาที่ใช้กันอย่างแพร่หลาย มีจุดเด่นที่ไม่ยึดติดอยู่กับระบบปฏิบัติการ สามารถนำไปใช้งานได้กับทุกแพลตฟอร์ม



ภาพที่ 2-7 โปรแกรมที่สร้างด้วยภาษาจาวา

2.4 คอมพิวเตอร์กราฟิก (Computer Graphic)

2.4.1 Sprite Animation ใน XNA นั้น ใช้ตัวแปร Texture2D ในการเก็บรูปภาพที่ใช้งานแต่หากศัพท์ที่นิยมใช้กันทั่วไปในการพัฒนาเกม อาจจะเรียกสิ่งเหล่านี้ว่า "Sprite" โดยปกติที่ผ่านมาได้มีการโหลดรูปเพื่อใช้ในเกมโดย 1 ภาพ เป็นภาพที่พร้อมใช้งาน แต่หากเกมมีเป็น 100 ภาพ มันก็เสียเวลาที่โหลดคอนเทนท์ (Content) นานเกินไป ดังนั้นเราจึงมีวิธีการจัดภาพที่มีลักษณะการใช้งานเข้าด้วยกันมาอยู่ในภาพเดียวกัน และเลือกที่จัดการโดยการคลิป (Clip)



ภาพที่ 2-8 การคลิป (Clip) ของ Sprite

2.5 ระบบเครือข่ายไร้สาย (Wireless LAN)

ไวเลส (Wireless) หมายถึง เครือข่ายไร้สาย มักใช้กับระบบเครือข่าย ไม่ว่าจะเป็นในองค์กร หรือในระบบเครือข่ายอินเตอร์เน็ต ระบบเครือข่ายไร้สาย (Wireless LAN : WLAN) หมายถึง เทคโนโลยีที่ช่วยให้การติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์ 2 เครื่อง หรือกลุ่มของเครื่องคอมพิวเตอร์สามารถสื่อสารกันได้ รวมถึงการติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์กับอุปกรณ์เครือข่ายคอมพิวเตอร์ด้วยเช่นกัน โดยปราศจากการใช้สายสัญญาณในการเชื่อมต่อ แต่จะใช้คลื่นวิทยุเป็นช่องทางการสื่อสารแทน การรับส่งข้อมูลระหว่างกันจะผ่านอากาศ ทำให้ไม่ต้องเดินสายสัญญาณ และติดตั้งใช้งานได้สะดวกขึ้น ระบบเครือข่ายไร้สายใช้แม่เหล็กไฟฟ้าผ่านอากาศ เพื่อรับส่งข้อมูลข่าวสารระหว่างเครื่องคอมพิวเตอร์ และระหว่างเครื่องคอมพิวเตอร์กับอุปกรณ์เครือข่าย โดยคลื่นแม่เหล็กไฟฟ้านี้อาจเป็นคลื่นวิทยุ (Radio) หรืออินฟาเรด (Infrared) ก็ได้ การสื่อสารผ่านเครือข่ายไร้สายมีมาตรฐาน IEEE802.11 เป็นมาตรฐานกำหนดรูปแบบการสื่อสาร ซึ่งมาตรฐานแต่ละตัวจะบอกถึงความเร็วและคลื่นความถี่สัญญาณที่แตกต่างกันในการสื่อสารข้อมูล เช่น 802.11b และ 802.11g ที่ความเร็ว 11 Mbps และ 54 Mbps ตามลำดับ

2.6 TCP/IP Protocol (Transmission Control Protocol/Internet Protocol)

TCP/IP Protocol เป็นชุดของโพรโตคอลที่ถูกใช้ในการสื่อสารผ่านเครือข่ายอินเทอร์เน็ต โดยมีวัตถุประสงค์เพื่อให้สามารถสื่อสารจากต้นทางข้ามเครือข่ายไปยังปลายทางได้และสามารถหาเส้นทางที่จะส่งข้อมูลไปได้เองโดยอัตโนมัติถึงแม้ว่าในระหว่างทางอาจจะผ่านเครือข่ายที่มีปัญหาโพรโตคอลก็ยังคงหาเส้นทางอื่นในการส่งผ่านข้อมูลไปให้ถึงปลายทางได้

2.7 การเขียนโปรแกรมแบบซ็อกเก็ต (Socket Programming)

การเขียนโปรแกรมแบบซ็อกเก็ต (Socket Programming) คือ การเขียนโปรแกรมที่มีการติดต่อรับส่งข้อมูลกันผ่านเครือข่ายโดยโปรแกรมจะถูกแบ่งออกเป็นสองส่วน ได้แก่

- Server Process คือ โปรแกรมที่ทำหน้าที่อยู่บนเครื่องแม่ข่ายที่จะคอยให้บริการแก่เครื่องลูกข่าย โดยที่โปรแกรมของเครื่องแม่ข่ายนั้นจะต้องเปิดใช้งานอยู่ตลอดเวลาเพื่อคอยตรวจสอบว่ามีเครื่องลูกข่ายเข้ามาขอเชื่อมต่อหรือไม่ ถ้ามีโปรแกรมก็จะรับการเชื่อมต่อนั้นและทำตามคำร้องขอที่เครื่องลูกข่ายต้องการ

- Client Process คือ โปรแกรมที่อยู่บนฝั่งของลูกข่าย ซึ่งลูกข่ายจะเป็นผู้ที่เริ่มทำการเชื่อมต่อไปยังแม่ข่าย และเป็นผู้ขอใช้บริการจากแม่ข่าย การที่โพรเซส (Process) ของทั้งแม่ข่ายและลูกข่ายจะสามารถติดต่อและสื่อสารกันได้นั้นจำเป็นที่จะต้องมีกฎเกณฑ์ที่ทั้งสองฝ่ายใช้เหมือนกันจึงจะสามารถสื่อสารกันได้ และการรับส่งข้อมูลกันของ

เครื่องแม่ข่ายและลูกข่ายนั้น ตัวโปรแกรมประยุกต์จะรับส่งข้อมูลกันผ่านทาง ซ็อกเก็ต (Socket) ซึ่งเป็นช่องทางการสื่อสารที่โปรแกรมประยุกต์จะอ้างถึงได้โดยการใช้หมายเลขพอร์ต (Port) และ ไอพีแอดเดรส (IP Address) การเขียนโปรแกรมแบบซ็อกเก็ต (Socket programming) นั้นสามารถเลือกได้ ว่าจะให้แอพพลิเคชั่นเรียกใช้บริการของชั้น ทรานสปอร์ต (Transport) โดยการใช้โพรโตคอล TCP ซึ่งจะให้บริการในการส่งข้อมูลที่มีความน่าเชื่อถือ หรือ UDP ที่ไม่รับรองว่าข้อมูลจะถึงปลายทางหรือถึงปลายทางแต่ไม่ตรงตามลำดับแต่มีความเร็วในการส่งข้อมูลมากกว่า TCP

บทที่ 3
ขั้นตอนการดำเนินงาน

3.1 กำหนดขอบเขต

เอ็กไซต์ไบค์ (Excitebike) เป็นเกมแนวเรซซิ่ง (Racing) ซึ่งจะเป็นเกมจำลองการแข่งรถจักรยานยนต์ โดยสามารถเล่นพร้อมกันได้สูงสุด 4 คน โดยการแข่งจะให้กำหนดจำนวนรอบสนามในการแข่งขัน โดยในแต่ละรอบในการแข่งขันแต่ละสนามจะได้รับคะแนนตามลำดับการเข้าเส้นชัยก่อนหลัง และเมื่อมีผู้เข้าเส้นชัยเป็นคนแรกจะทำการนับเวลาถอยหลัง 20 วินาที เมื่อครบแล้วจะทำการจบเกมในสนามนั้น และเริ่มนับคะแนนตามระยะที่ผู้เล่นคนอื่นๆอยู่ห่างจากเส้นชัยเรียงตามลำดับ โดยเมื่อแข่งขันจนครบรอบการแข่งขันตามที่ได้กำหนดไว้ในตอนแรกแล้วระบบจะทำการรวมคะแนนในแต่ละสนามและแสดงผลลำดับที่ตามจำนวนคะแนนรวม โดยเกมเอ็กไซต์ไบค์ (Excitebike) จะสามารถเล่นได้บนโทรศัพท์มือถือระบบปฏิบัติการแอนดรอยด์ (Android Operating System) ตั้งแต่รุ่น 2.2 ขึ้นไปที่มีหน้าจอขนาด 320x240, 480x320, 800x480 พิกเซล (Pixel) โดยการเชื่อมต่อกันนั้นจะใช้การเชื่อมต่อผ่านระบบเครือข่ายไร้สาย (Wireless LAN)

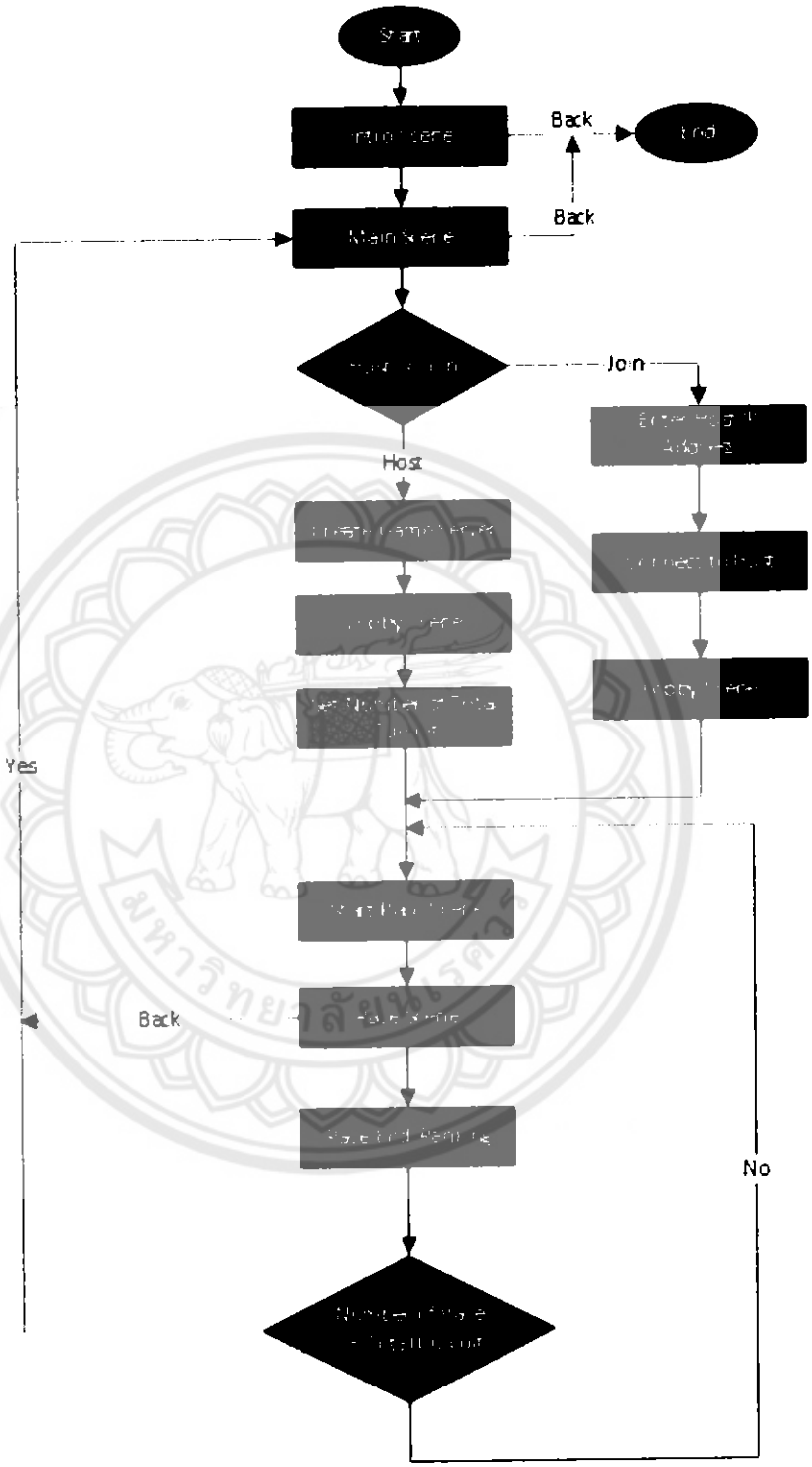
3.2 การออกแบบตัวโครงสร้างของเกม

โครงสร้างการทำงานของโปรแกรมจะเริ่มจากหน้าจอเริ่มต้นเกม (IntroScene) เมื่อทำการสัมผัสหน้าจอ (Touch Screen) จะเข้าสู้หน้าจอหลัก (MainScene) เพื่อทำการเลือกประเภทของการเล่นว่าจะเป็นโฮส (Host) หรือไคลเอนต์ (Client)

- กรณีเลือกเป็นโฮส (Host) จะต้องทำการใส่ชื่อตัวละครและเลือก Create Game Sever จะเข้าสู้หน้าจอต่อไปคือหน้าจอล็อบบี้ (LobbyScene) เพื่อรอการเชื่อมต่อจากไคลเอนต์ (Client) ในหน้าจอนี้ผู้เล่นที่เป็นโฮส (Host) จะสามารถเลือกจำนวนรอบของสนามที่ใช้แข่งและจะเป็นผู้กดเริ่มเกมเมื่อผู้เล่นพร้อม

- กรณีเลือกเป็นไคลเอนต์ (Client) ให้ทำการใส่ชื่อตัวละครและไอพีของเครื่องโฮส (Host IP) เมื่อเรียบร้อยแล้วให้ทำการเลือก Join Game จะเป็นการเชื่อมต่อกับโฮส (Host) และเข้าสู่หน้าจอล็อบบี้ (LobbyScene) เพื่อรอการเริ่มเกมจากโฮส (Host)

เมื่อโฮส (Host) ทำการกดเริ่มเกมจะเข้าสู้หน้าจอเริ่มการแข่ง (StartRaceScene) และเข้าสู่หน้าจอแข่งขัน (RaceScene) เพื่อทำการแข่ง เมื่อทำการแข่งขันจบจะเข้าสู่หน้าจอจบการแข่งขันและจัดลำดับแสดงผลคะแนน (RaceEnd/Ranking) เกมจะทำการตรวจสอบว่าครบตามจำนวนรอบสนามที่กำหนดหรือไม่ถ้ายังไม่ครบตามที่กำหนดจะทำการเริ่มแข่งสนามต่อไปจนครบตามจำนวนที่กำหนดและทำการจบเกม



ภาพที่ 3-1 โครงสร้างหลักของเกม

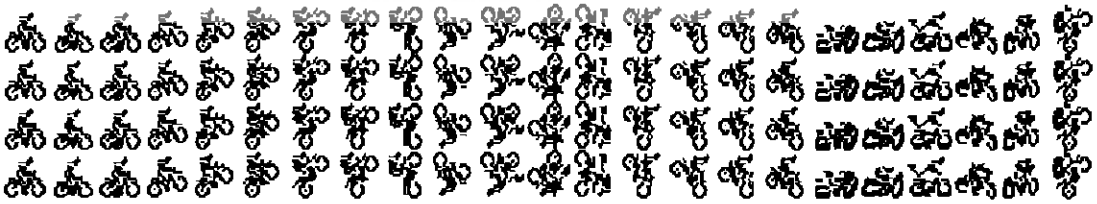
3.3 การออกแบบกราฟฟิก (Graphic Design) และเสียงภายในเกม

การออกแบบกราฟฟิกภายในเกมจะแบ่งออกเป็นส่วนย่อยๆ ได้ 4 ส่วนได้แก่

3.3.1 การควบคุม ภาพในเกม เป็นการควบคุมโดยระบบสัมผัสออกแบบให้มีลักษณะโปร่งใส เพื่อไม่ให้บังทัศนะวิสัยการมองโดยแบ่งออกเป็น 2 ส่วนได้แก่ ส่วนของการบังคับการเคลื่อนที่ตัวละคร ใกล้ไกล (เปลี่ยนเลน) ยกล้อหรือกดหน้ารถลง และในส่วนของการบังคับความเร็วในการขับ (ความเร็วในการเลื่อนของฉากพื้นหลัง) โดยจะมีการบังคับความเร็วแบบธรรมดา และการเร่งสปีด มีรายละเอียด ดังนี้

ภาพที่ 3-2 ส่วนของการติดต่อผู้ใช้

3.3.2 การออกแบบตัวละครในเกม จะใช้การออกแบบตัวละครเลียนแบบเครื่องเล่นเกมแฟมิคอม (Famicom) การออกแบบตัวละครขนาดเล็ก เหมาะกับการออกแบบเกมที่มีขนาดแบบพิกเซล (Pixel) มีรายละเอียดดังนี้



ภาพที่ 3-3 ภาพตัวละครภายในเกม

แอคชั่นของตัวละครที่แสดงจะใช้การคลิปภาพเป็นตำแหน่งพิกเซล (Pixel) จากภาพเพื่อแสดงตามอีเว้นและตำแหน่งต่างๆ ที่เกิดขึ้นขณะนั้นๆ เช่นการล้ม หรือกระโดด โดยตัวละครภายในเกมจะมีแกนการแสดงผล 2 แกนคือ แกนใกล้ไกล (เปลี่ยนเลน) และแกนขึ้นลง (ขึ้นเนินกระโดด)

3.3.3 การออกอุปสรรคภายในเกม

ออกแบบโดยการอ้างอิงตามเกมแฟมมิคอม (Famicom) โดยการจัดวางอุปสรรคจะทำโดยการคลิปภาพเป็นพิกเซล (Pixel) และกำหนดหมายเลขของอุปสรรคแล้วทำการเลือกหมายเลขมาจัดว่างทับสนามตามตำแหน่งพิกเซล (Pixel) ที่ต้องการ โดยอุปสรรคทั้งหมดประกอบด้วย

ก. เนินกระโดด แบ่งได้ 4 ชนิด ได้แก่ เนินมุม 30 องศา,45 องศา,60 องศา และเนินพิเศษ โดยแต่ละเนินจะให้ค่าความสูงในการกระโดดแตกต่างกันไปตามมุมของเนิน

ข. ที่กั้นทาง เมื่อเคลื่อนที่ผ่านโดยไม่กดยกล้อจะทำให้รถเกิดการล้ม

ค. ยางมะตอย และพื้นที่ถนนขาด เมื่อทำการเคลื่อนที่ผ่านจะทำให้ค่าความเร็วสูงสุดที่สามารถทำได้ลดลง

ง. สัญลักษณ์ลดอุณหภูมิ เมื่อเคลื่อนที่ผ่านจะทำให้ค่าอุณหภูมิกลับไปเป็นค่าเริ่มต้น

จ. เนินใหญ่แบ่งได้ 2 เนินมีลักษณะไม่เหมือนกัน

ฉ. LAP 1 เป็นตัวบ่งบอกว่าถึงครึ่งทางของการแข่งขันแล้วโดยจะจัดวางไว้กึ่งกลางของระยะทางทั้งหมดแล้วทำการเรียงลำดับอุปสรรคเหมือนอุปสรรคก่อนหน้า LAP 1 อีกครั้งโดยบวกค่าตำแหน่งที่ว่างเริ่มเข้าไป

ช. FINISH ใช้เป็นเนินที่แสดงการจบการแข่งขันใช้ว่างไว้เนินสุดท้ายในสนาม

ในส่วนของรายละเอียดในการออกแบบการทำงานของแต่ละอุปสรรคจะขอกล่าวในขั้นตอนถัดไปในส่วนของการออกแบบและวิเคราะห์ระบบ (System Analysis and Design)

ภาพที่ 3-4 อุปสรรคภายในเกม

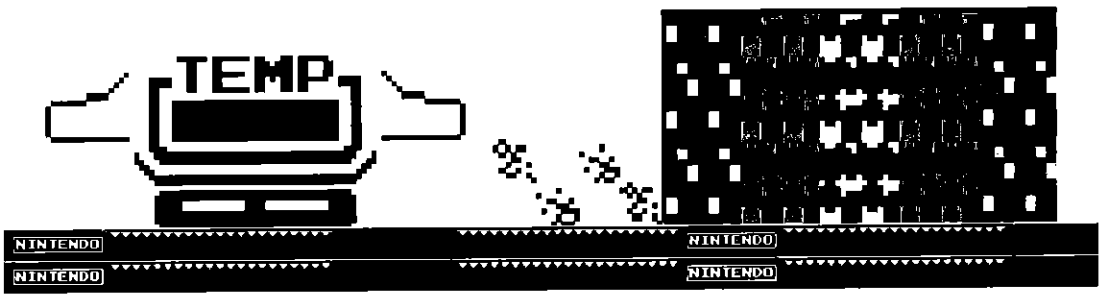
3.3.4 การออกแบบพื้นสนาม

ออกแบบให้สีในแต่ละสนามต่างกันโดยจะใช้การคลิปภาพ และนำมากันต่อกันโดยในหนึ่งหน้าจอจะใช้ภาพมาต่อกันทั้งหมด 10 ภาพและเมื่อสนามทำการเลื่อนภาพไปจะนำภาพที่เลื่อนไปวนมาใช้ใหม่อีกครั้งจนจบการแข่งขัน

ภาพที่ 3-5 โครงสร้างพื้นสนาม

3.3.5 การออกแบบส่วนแสดงผลสถานะตัวละครและส่วนที่เหลือ

พื้นหลังและเส้นคาดจะใช้หลักการทำงานคล้ายๆกับการออกแบบพื้นสนามคือการต่อภาพและเลื่อนภาพวนไปมา ส่วนการออกแบบแถบอุณหภูมิและในส่วนของครัวจะเกิดขึ้นตามเหตุการณ์ต่างๆที่กำหนด ค่าแถบอุณหภูมิจะใช้การวาดเริ่มขึ้นตามการนับเวลาการกด

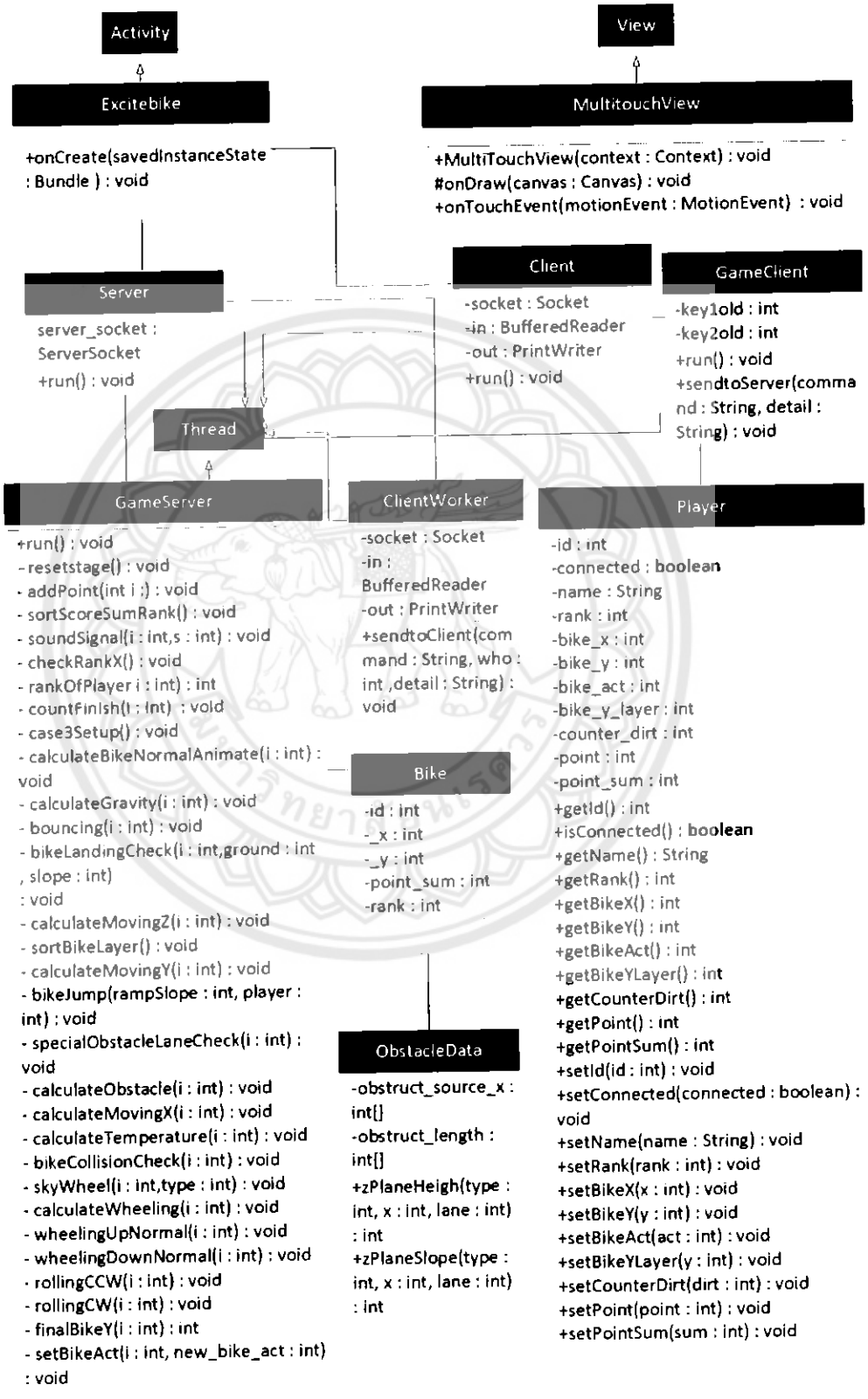


ภาพที่ 3-6 ส่วนแสดงผลสถานะตัวละครและส่วนที่เหลือ

3.3.6 เสียงภายในเกม

เสียงภายในเกมใช้การอัดเสียงเกมเครื่องแฟมมิคอม (Famicom) เป็นไฟล์ .mp3 และเรียกใช้งานตามเหตุการณ์ที่เกิดขึ้น

3.4 การออกแบบและวิเคราะห์ระบบ (System Analysis and Design)



ภาพที่ 3-7 คลาสไดอะแกรม

โครงสร้างของคลาสหลักภายในเกม

หน้าที่ของคลาสต่าง ๆ ในภายในเกม	
คลาส	**หน้าที่**
Excitebike	เป็นคลาส Activity หลักของเกม เริ่มทำงานเมื่อมีการเปิดเกมขึ้นมา เมื่อเริ่มทำงานจะโหลดไฟต่างๆ มาเก็บไว้ และ สร้างออบเจ็ค (Object) ขึ้นมาจากคลาสอื่นๆ เพื่อเริ่มทำงานต่อไป
Server	คลาสนี้จะถูกเรียกใช้เมื่อผู้ใช้เลือกเป็นโฮส (Host) ของเกม โดยจะสร้างซ็อกเก็ตเซิฟเวอร์ (Socket Server) ขึ้นมา เพื่อรอรับการเชื่อมต่อจากไคลเอนต์ (Client) ผ่านทางซ็อกเก็ต (Socket)
ClientWorker	คลาสนี้จะถูกเรียกใช้เมื่อมีไคลเอนต์ (Client) ทำการเชื่อมต่อเข้ามา โดยจะทำหน้าที่ รอรับข้อมูลที่ไคลเอนต์ (Client) ส่งมาผ่านทางซ็อกเก็ต (Socket) เพื่อนำไปประมวลผลต่อไป
GameServer	คลาสนี้จะถูกเรียกใช้โดยคลาสเซิฟเวอร์ (Server) เป็นคลาสที่ทำหน้าที่ประมวลส่วนของตัวเกมทั้งหมดทำโดยนำค่าที่ไคลเอนต์ (Client) แต่ละคนส่งมา นำมาประมวลผลและส่งค่าที่ได้กลับไปหาไคเอนต์(Client)
Client	คลาสนี้จะถูกเรียกใช้เมื่อผู้ใช้เริ่มเชื่อมต่อไปหาเซิฟเวอร์ (Server) ผ่านทางซ็อกเก็ต (Socket) เมื่อเชื่อมต่อสำเร็จ ก็จะทำหน้าที่ รอรับข้อมูลที่เซิฟเวอร์ (Server) ส่งมา เพื่อนำไปประมวลผลต่อไป
GameClient	ทำหน้าที่ นำค่าอินพุต (Input) ต่างๆ ของผู้ใช้ ส่งไปหาเซิฟเวอร์ (Server)
MultiTouchView	เป็นคลาสที่ทำหน้าที่ นำข้อมูลตัวแปรต่างๆ ทั้งหมดที่เซิฟเวอร์ (Server) ส่งมาให้ มาแสดงผลเป็นกราฟฟิก (Graphic) ของเกม และ รับค่าอินพุต (Input) จากผู้ใช้

ตารางที่ 3-1 โครงสร้างของคลาสภายในเกม

หน้าที่และการทำงานในส่วนย่อยของคลาสต่างๆ

Excitebike

onCreate(savedInstanceState : Bundle) void

แอนดรอยด์ (Android) จะเรียก onCreate() เมื่อ Activity Start ในหนึ่งช่วงเวลาของแอพพลิเคชั่น (Application) นั้น อาจมีการ Create และ Destroy Activity อยู่เรื่อยๆ ส่วน Bundle savedInstanceState เป็นออบเจ็ค (Object) ของ import android.os.Bundle; ทำหน้าที่จัดการทรัพยากร(Resource) และสถานะของโปรแกรม

ClientWorker

sendtoClient(command : String, who : int, detail : String) : void

เป็นตัวส่งข้อมูลไปบอกกับไคลเอนต์ (Client) ที่ ออบเจ็ค (Object) ของ ClientWorker นี้ทำการเชื่อมต่ออยู่ โดย command หมายถึง คำสั่ง who ผู้เล่นคนไหน และ detail หมายถึง รายละเอียดของคำสั่งนั้น ตัวอย่างเช่น คำสั่ง sendtoClient("bike_x", 1, "2698"); จะทำการส่ง String "bike_x:1:2698" ไปหาไคลเอนต์ (Client) ที่ทำการเชื่อมต่ออยู่ ซึ่งจะบอกว่า ค่าตำแหน่งในแกน x ของผู้เล่นคนที่ 1 อยู่ที่ 2698

GameServer

resetstage()

ทำหน้าที่รีเซ็ตค่าตัวแปรต่างๆ ถูกเรียกเมื่อมีการเปลี่ยนสเตจ (Stage)

addPoint(i : int) : void

ทำหน้าที่ให้คะแนนกับผู้เล่น (Player) จากอันดับการเข้าเส้นชัย

sortScoreSumRank()

คำสั่งนี้จะถูกเรียกหลังจากผู้เล่นได้รับคะแนนจากการเข้าเส้นชัย เพื่อเรียงอันดับของผู้เล่นจากคะแนนรวมทั้งหมด

soundSignal(i : int, s : int, offset : int)

เป็นคำสั่งที่จะทำการส่งข้อมูลไปบอกกับไคลเอนต์ (Client) ว่าให้ไคลเอนต์ (Client) ทำการเล่นหรือหยุดเล่นไฟล์เสียงที่กำหนด คำสั่งนี้เรียกใช้โดยคลาส GameServer

checkRankX()

คำสั่งที่ใช้ในการเรียงอันดับของผู้เล่น (Player) ขณะแข่งขันกันอยู่ โดยผู้เล่นที่ค่าตำแหน่งในแกน x สูงที่สุด จะอยู่ในอันดับแรก คำสั่งนี้เรียกใช้โดยคลาส GameServer

countFinish(i : int) : void

คำสั่งที่คำนวณเกี่ยวกับสถานะของผู้เล่นเกี่ยวกับการเข้าเส้นชัย ตรวจเช็คว่าเข้าเส้นชัยแล้วหรือยัง คำสั่งนี้เรียกใช้โดยคลาส GameServer

calculateBikeNormalAnimate(i : int) : void

คำสั่งที่คำนวณเกี่ยวกับสถานะแอนนิเมชั่น (Animation) ของ bike โดยคำนวณจากสถานะต่างๆ ในตอนนั้น ว่าควรจะแสดงผลภาพใด คำสั่งนี้เรียกใช้โดยคลาส GameServer

calculateGravity(i : int) : void

คำสั่งที่คำนวณเกี่ยวกับการเคลื่อนที่ของรถ ถูกเรียกใช้เมื่อรถของผู้เล่นมีสถานะเคลื่อนไหวอยู่กลางอากาศ โดยจะขยับตำแหน่งในแกน z ของรถลงมาเรื่อย ๆ ด้วยความเร่ง 0.3 pixel/frame2

bouncing(i : int) : void

คำสั่งที่คำนวณเกี่ยวกับการเด้งกับพื้นของ bike ว่าหากการตกกระทบพื้นครั้งนั้นไม่ใช่การเด้งกับพื้น ก็ให้รถมีสถานะเด้งกับพื้น แล้วลอยขึ้นไปในอากาศอีกครั้ง คำสั่งนี้เรียกใช้โดยเมธอด bikeLandingCheck()

bikeLandingCheck(i : int, ground : int, slope : int) : void

ถูกเรียกโดยเมธอด calculateGravity() จะเรียกทุกครั้งที่เมธอด calculateGravity()ทำงาน เพื่อตรวจสอบว่าตำแหน่ง bike z ตกกระทบพื้นสนามหรือยัง และถ้าตกกระทบพื้นแล้ว จะเกิดอะไรขึ้นบ้าง

calculateMovingZ(i : int) : void

คำสั่งที่คำนวณเกี่ยวกับการเคลื่อนที่ของ bike ในแนวแกน z (แนวดิ่ง) ตัวอย่างเช่น เมื่อรถกำลังเคลื่อนที่ผ่านสิ่งกีดขวาง ตำแหน่งในแกน z ก็จะเปลี่ยนตามความสูงของสิ่งกีดขวางนั้น คำสั่งนี้ เรียกใช้โดยคลาส GameServer

calculateMovingY(i : int) : void

คำสั่งที่คำนวณเกี่ยวกับการเคลื่อนที่ของ bike ในแนวแกน y (เปลี่ยนเลน) โดยตำแหน่งในแกน y จะเปลี่ยนเมื่อผู้เล่นนั้นมีการกดปุ่มขึ้นลง เรียกใช้โดย GameServer

bikeJump(rampSlope : int, player : int) : void

คำสั่งที่คำนวณเกี่ยวกับสถานะของรถ เมื่อรถมาอยู่ที่จุดสูงสุดของเนิน ทำให้รถเกิดความเร็วในแกน Z ถูกเรียกโดยเมธอด calculateObstacle()

specialObstacleLaneCheck(i : int) : void

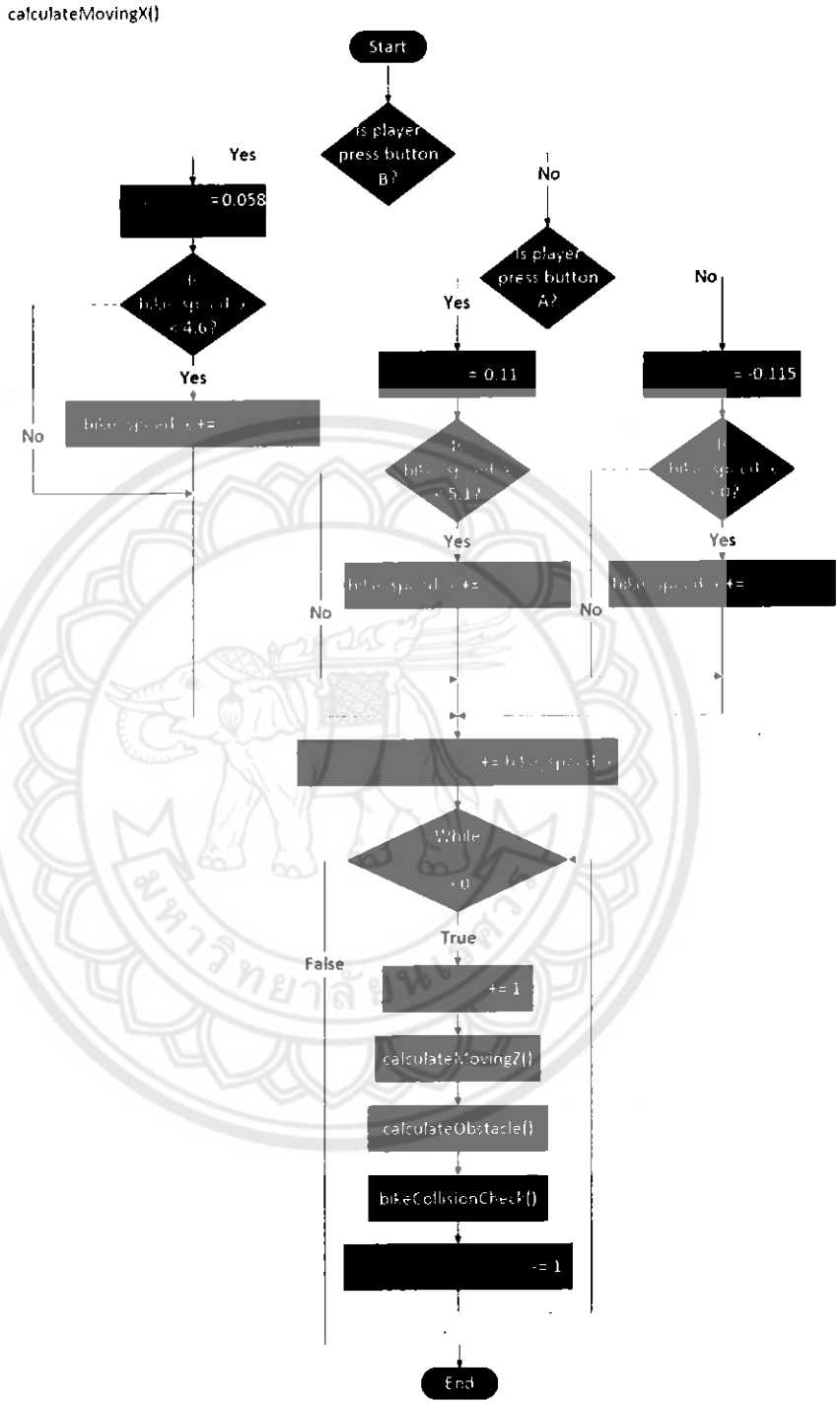
คำสั่งที่คำนวณเกี่ยวกับการเคลื่อนที่ของ bike ในแนวแกน y โดยตรวจสอบการเปลี่ยนเลนบนเนินแบบพิเศษ ถูกเรียกโดยเมธอด calculateMovingY()

calculateObstacle(i : int) : void

คำสั่งที่คำนวณเกี่ยวกับการตรวจสอบสถานะของรถเมื่ออยู่บนเนิน โดยจะเปรียบเทียบตำแหน่งรถกับตำแหน่งของสิ่งกีดขวางว่าจะเกิดอะไรขึ้นถ้ารถอยู่ในตำแหน่งนั้นถูกเรียกโดยเมธอด calculateMovingX()

calculateMovingX(i : int) : void

คำสั่งที่คำนวณเกี่ยวกับการเคลื่อนที่ของ bike ในแนวแกน x โดยจะทำการเช็คว่าผู้เล่นกดปุ่มอะไรหรือไม่ในทุกรอบการคำนวณ และทำการเร่งความเร็วรถตามปุ่มที่กด(ความเร็วในการเลื่อนฉากสนาม) โดยมี 3 กรณีได้แก่ ผู้เล่นกดปุ่ม A จะมีความเร่ง 0.058 เพิ่มไปเรื่อยๆจนความเร็วสูงสุดที่ 4.6 พิกเซลต่อเฟรม ผู้เล่นกด B จะมีความเร่ง 0.11 เพิ่มไปเรื่อยๆจนความเร็วสูงสุดที่ 5.1 พิกเซลต่อเฟรม และผู้เล่นไม่ได้กดจะมีความเร่งเป็น -0.115 จนเหลือ 0 เรียกใช้โดยคลาส GameServer



ภาพที่ 3-8 เมธอด calculateMovingX

calculateTemperature(i : int) : void

คำสั่งที่คำนวณเกี่ยวกับความร้อนของรถ เช่นเมื่อเร่งเครื่องนานๆ ความร้อนก็จะสูงขึ้นเรื่อยๆ และเมื่อความร้อนถึงจุดสูงสุด รถก็วิ่งต่อไม่ได้ ต้องหยุดพักจนความร้อนลดกลับมาเป็นปกติ เรียกใช้โดยคลาส GameServer

bikeCollisionCheck(i : int) : void

คำสั่งที่คำนวณเกี่ยวกับการชนกันของรถ โดยเปรียบเทียบตำแหน่งของรถทั้งสามแกนของแต่ละคัน หากขอบเขตของรถมีการซ้อนทับกัน รถที่ตำแหน่งแกน x น้อยกว่า จะอยู่ในสถานะคว่ำ เรียกใช้โดยเมธอด calculateMovingX() และ calculateMovingY()

skyWheel(i : int, type : int) : void

คำสั่งที่คำนวณเกี่ยวกับการยกล้อของรถกลางอากาศ เช่น ขณะที่รถลอยอยู่กลางอากาศ ถ้าผู้เล่นกดปุ่มลูกศรซ้าย รถจะตกพื้นช้าลง แต่ถ้ากดปุ่มลูกศรขวา รถจะตกพื้นเร็วขึ้น แต่ความเร็วในแกน x จะเพิ่มขึ้น เรียกใช้โดยเมธอด calculateWheeling()

calculateWheeling(i : int) : void

คำสั่งที่คำนวณเกี่ยวกับการยกล้อของรถ ถ้าผู้เล่นกดปุ่มลูกศรซ้าย รถก็จะยกล้อหน้าขึ้น โดยเรียกใช้เมธอด wheelingUpNormal() แต่ถ้าปล่อยปุ่มก็จะเรียกใช้ wheelingDownNormal() เรียกใช้โดย คลาส GameServer

wheelingUpNormal(i : int) : void

คำสั่งที่คำนวณสถานะของรถขณะยกล้อหน้าขึ้น โดยรถจะยกล้อสูงขึ้นเรื่อยๆ หากยกสูงเกินไปจะเสียการควบคุมและล้ม เรียกใช้โดยเมธอด calculateWheeling()

wheelingDownNormal(i : int) : void

คำสั่งที่คำนวณสถานะของรถขณะยกล้อหน้าลง โดยเมื่อผู้เล่นปล่อยปุ่มลูกศรซ้าย ขณะที่รถกำลังยกล้อ ล้อหน้าก็จะลดต่ำลงจนเป็นสถานะปกติ เรียกใช้โดยเมธอด calculateWheeling()

rollingCCW(i : int) : void

คำสั่งที่คำนวณสถานะกลิ้งแบบม้วนหลังของ bike โดย เมื่อรถอยู่สถานะนี้ ผู้เล่นจะควบคุมรถไม่ได้ และความเร็วจะลดลงจนกระทั่งรถหยุด ผู้เล่นจึงจะกลับมาควบคุมรถได้

rollingCW(i : int) : void

คำสั่งที่คำนวณสถานะกลิ้งแบบม้วนหน้าของbike โดย เมื่อรถอยู่สถานะนี้ ผู้เล่นจะควบคุมรถไม่ได้ และความเร็วจะลดลงจนกระทั่งรถหยุด ผู้เล่นจึงจะกลับมาควบคุมรถได้

GameClient

sendtoServer(command : String, detail : String) : void

เป็นส่งข้อมูลไปบอกกับ Server โดย command หมายถึง คำสั่ง และ detail หมายถึงรายละเอียดของคำสั่งนั้น ตัวอย่างเช่น คำสั่งsendtoServer ("key1", "1"); จะทำการส่ง String "key1:1" ไปหา Server ซึ่งจะบอกว่า ขณะนี้ผู้เล่นคนนี้กำลังกดปุ่ม key1 อยู่

MultiTouchView

px(x : int) : int

เป็นการแปลงค่าอินพุต x ซึ่งจะเป็นตำแหน่งในแกน x บนหน้าจอขนาดปกติ เป็นเอาต์พุตตำแหน่งในแกน x บนหน้าจอในขนาดที่ตรงกับขนาดของจอของเครื่องผู้เล่น

py(y : int) : int

เป็นการแปลงค่าอินพุต y ซึ่งจะเป็นตำแหน่งในแกน y บนหน้าจอขนาดปกติ เป็นเอาต์พุตตำแหน่งในแกน y บนหน้าจอในขนาดที่ตรงกับขนาดของจอของเครื่องผู้เล่น

drawIntroPage(canvas : Canvas) : void

เป็นเมธอดที่ทำการแสดงผลหน้าจอตอนเกมอยู่ในสถานะ Intro เรียกใช้โดยเมธอด onDraw()

drawLobbyText(canvas : Canvas) : void

เป็นเมธอดทีทำการแสดงผลข้อความบนหน้าจอตอนเกมอยู่ในสถานะ Lobby เรียกใช้โดยเมธอด onDraw()

drawLobbyBike(canvas : Canvas, i : int) : void

เป็นเมธอดที่ทำการแสดงผลรูปรถบนหน้าจอตอนเกมอยู่ในสถานะ Lobby เรียกใช้โดย เมธอด onDraw()

drawStartCurcuitPage(canvas : Canvas) : void

เป็นเมธอดที่ทำการแสดงผลหน้าจอตอนเกมอยู่ในสถานะ Start Curcuit เรียกใช้โดย เมธอด onDraw()

onDraw(canvas : Canvas) : void

เป็นเมธอดที่ทำการแสดงผลภาพตามค่าตัวแปรต่างๆ ที่ Server ส่งมา โดยจะแสดงผลทุกๆ 25 มิลลิวินาที

onTouchEvent(motionEvent : MotionEvent) : void

เป็นเมธอดที่จะถูกเรียกเมื่อผู้เล่นมีการกดที่จอสัมผัส เป็นการรับค่าอินพุตจากผู้เล่น โดยจะแบ่งเป็นสองส่วนคือ ส่วน ลูกศรซ้ายขวาขึ้นลง 4 ปุ่ม และ ส่วนเร่งเครื่อง 2 ปุ่ม

onKeyUp(keyCode : int, event : KeyEvent) : void

เป็นเมธอดที่จะถูกเรียกเมื่อผู้เล่นมีการกดปุ่มต่างๆ โดยในเกมจะใช้งานเพียงปุ่ม Back เท่านั้น เพื่อใช้ในการกดเพื่อออกเกม หรือออกจากการแข่งขัน

Bike

ใช้ในการเก็บค่าต่างๆของผู้เล่น และ เก็บข้อมูลตำแหน่ง x y z ของรถเพื่อใช้ในการเรียงลำดับ

ObstacleData

ใช้ในการเก็บค่าต่างๆของสนามและอุปสรรคในเกมโดยการใส่อุปสรรคในสนามเพิ่มจะทำผ่านคลาสนี้

Player

ใช้เพื่อเก็บค่าต่างของผู้เล่นแต่ละคน เช่น ชื่อตัวละคร หรือ เก็บข้อมูลตำแหน่ง x y z ของรถคล้ายกับคลาส Bike แต่จะเรียกใช้โดยไคลเอนต์(Client)

3.5 การเขียนโปรแกรมเพื่อติดต่อสื่อสารกันระหว่างเครื่องผ่านระบบเครือข่าย

การพัฒนาโปรแกรมในระดับของการส่งข้อมูลในรูปแบบของสายของข้อมูล (Data Stream) ก็คือการเขียนโปรแกรมเพื่อติดต่อกันแบบซ็อกเก็ต (Socket) ซึ่งเป็นการเขียนโปรแกรมที่ถือว่าส่งผ่านข้อมูลกันในระดับของชั้นเน็ตเวิร์กที่ต่ำ ที่สุดที่เรียกว่าการติดต่อกันแบบจุดต่อจุด ที่ต้องเขียนโปรแกรมสำหรับจัดการการติดต่อจัดการให้บริการ และการเข้าถึงข้อมูลต่าง ๆ เอง ทั้งหมด

3.5.1 Socket Programming

ซ็อกเก็ต คือ จุดปลายของการติดต่อสื่อสารในชั้นของทรานสปอร์ตเลเยอร์ (Transport layer) ในชั้นนี้เรามักจะกล่าวในระดับของระบบเครือข่ายคือติดต่อสื่อสารกันระหว่างเซิร์ฟเวอร์และ ไคลเอนต์ในลักษณะของจุดต่อจุด ซึ่งการติดต่อจะมีโปรโตคอลที่ใช้ในการสื่อสาร คือ TCP หากเปรียบเทียบซ็อกเก็ตกับโทรศัพท์ที่ต้องติดต่อกันระหว่างผู้รับกับผู้ส่งกล่าวคือต้องรอการ ตอบรับอีกฝั่งหนึ่งของการสื่อสารแต่ขณะเดียวกันโปรโตคอล UDP จะเปรียบเสมือนกับกล่อง จดหมายที่ไม่จำ เป็นต้องมีผู้รับอีกฝั่งอยู่ตลอดเวลาเพียงแต่ส่งจดหมายในกล่องแล้วก็เสร็จสิ้นภารกิจเป็นต้น การสื่อสารโดยใช้ซ็อกเก็ตนี้จะมีส่วนสำคัญอยู่สองส่วนคือไคลเอนต์และเซิร์ฟเวอร์ โครงสร้างข้อมูลของซ็อกเก็ตนั้นมีหลากหลายโครงสร้างจะขึ้นอยู่กับการนำ ไปใช้ เช่น จุดปลาย (End-Point) ระหว่างโพรเซส (Process) ในเครื่องเดียวกัน กรณีนี้อาจแทบไม่ได้ใช้งาน จุดปลายระหว่างโพรเซสของเครือข่าย เป็นต้น

3.5.2 TCP Sockets

การสร้างลิงค์ติดต่อสื่อสารแบบ TCP/IP ที่เป็น sockets เป็นการเชื่อมต่อแบบ connection-orientated ซึ่งนั้นก็หมายความว่าการสนทนาระหว่างเครื่อง client กับ server จะทำการเชื่อมต่อตลอดเวลาที่สนทนานอกเสียจากมันจะเสีย ซึ่งการสนทนากันหรือการแลกเปลี่ยนข้อมูลระหว่าง client กับ server นั้นจะต้องเป็นไปตามกฎของ protocol ซึ่งจะการกำหนดลักษณะการทำงานนั้น ทำให้สามารถแบ่งขั้นตอนการทำงานของ server ได้ 5ขั้นตอน ดังนี้

1. สร้าง object ของ ServerSocket

ServerSocket constructor ต้องการค่า หมายเลข port ? (มีค่าตั้งแต่ 1024-65535 โดยไม่คิดรวมหมายเลข port สงวน) เป็นค่า argument ตัวอย่างเช่น

```
ServerSocket servSock = new ServerSocket(1234);
```

จากตัวอย่างนี้จะทำให้ server ให้บริการ client ที่ทำการเชื่อมต่อมายัง port 1234

2. ทำให้ server อยู่ในสถานะพร้อมบริการ

Server จะอยู่ในสถานะที่รอให้บริการได้นั้นจะต้องมีการเรียกใช้ method accept ของ class ServerSocket ซึ่งจะreturn ค่าเป็น object ของ Socket class ตัวอย่างการใช้งานเช่น

```
Socket link = servSock.accept();
```

3. ตั้งค่า input และ output stream

methods getInputStream และ getOutputStream เป็น method ที่ class Socket ใช้สำหรับ อ่านค่าที่ได้จาก stream ในการเชื่อมต่อกันใน 2 ขั้นตอน ในการเชื่อมต่อสื่อสาร client ได้สร้าง connection เราสามารถใช้ Scanner object ในการอ่านค่าจาก InputStream object โดย return ค่าโดน method getInputStream โดยอยู่ในรูป string ? orientated input โดยถูกกำหนดจาก standard input stream,System.in

ตัวอย่างเช่น

```
Scanner input = new Scanner(link.getInputStream());
```

ลักษณะคล้ายกัน เราก็สามารถส่ง output โดยใช้ PrintWriter object ในการส่งค่า output ไปยัง OutputStream object โดยใช้ method ที่ชื่อว่า getOutputStream ?ซึ่งสามารถใช้ได้โดยการกำหนดค่าใน constructor ของ PrintWriter โดยกำหนดค่า agument 2 ค่า

ตัวอย่าง

```
PrintWriter output = new PrintWriter(link.getOutputStream(),true);
```

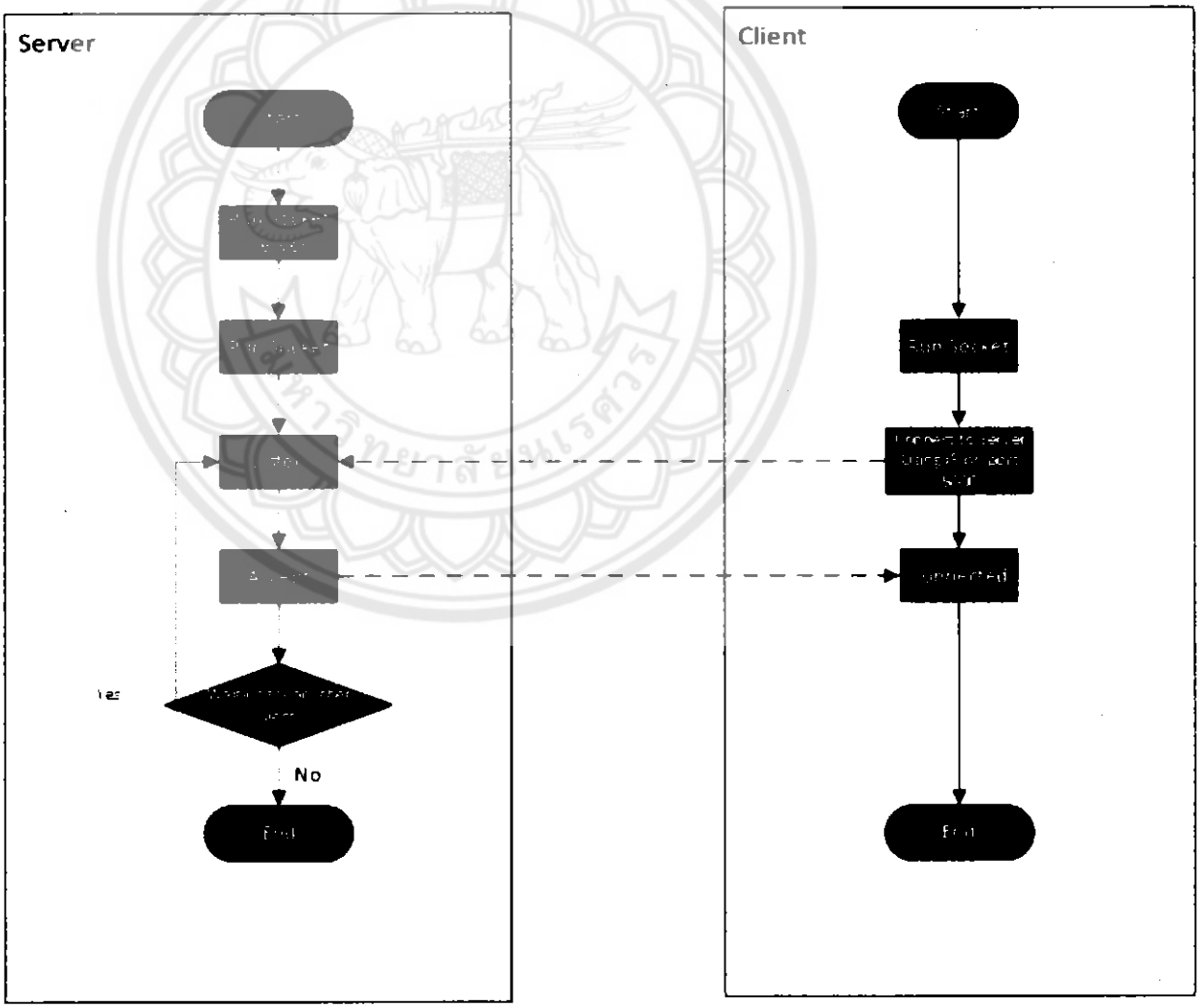
4. ส่งและรับข้อมูล

หลังจากที่ทำการ set up Scanner และ PrintWrite ให้สามารถส่งและรับข้อมูลได้แล้วก็จะทำการรับและส่งข้อมูล โดยใช้ method nextLine สำหรับอ่านข้อมูลที่รับเข้ามา และใช้ method println สำหรับส่งข้อมูล

ตัวอย่างการใช้งาน

```
output.println("Awaiting data…");
String input = input.nextLine();
```

5. สุดท้ายเมื่อเสร็จสิ้นการเชื่อมต่อก็ต้องทำการปิดการเชื่อมต่อ



ภาพที่ 3-9 Flowchart แสดงการเชื่อมต่อแบบ Socket

บทที่ 4
ผลการดำเนินงาน

การที่ได้พัฒนาโปรแกรมเกมบนโทรศัพท์มือถือระบบปฏิบัติการแอนดรอยด์ (Android Operating System) แบบหลายผู้เล่น ทางผู้จัดทำได้ทราบถึงข้อจำกัดหลายอย่างในการพัฒนาระบบในด้านต่าง ๆ ทั้งในส่วนที่เป็นโปรแกรมที่จะนำมาพัฒนาบนโทรศัพท์มือถือ ซึ่งโปรแกรมที่นำมาใช้นั้นต้องรองรับกับโทรศัพท์มือถือที่ใช้งานได้ทั่วไป และ ได้ทราบถึงข้อจำกัดที่เกี่ยวกับโทรศัพท์มือถือที่มีหน่วยความจำ และความเร็วของการประมวลผลบนโทรศัพท์มือถือที่มีจำกัด ทำให้ทางผู้จัดทำสามารถปรับปรุงแก้ไขในส่วนต่าง ๆ มาโดยตลอดและสามารถที่จะพัฒนาโปรแกรมเกมบนโทรศัพท์มือถือแบบหลายผู้เล่นให้สมบูรณ์

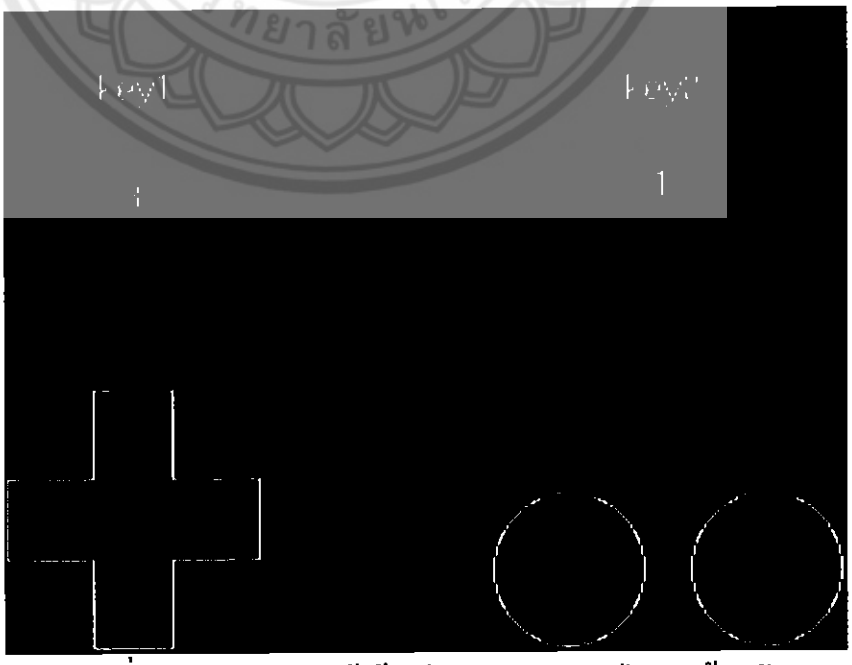
ในบทนี้จะนำเสนอผลการดำเนินงาน และการเข้าถึงเกมบนโทรศัพท์มือถือ ซึ่งจะมีส่วนของรายละเอียดของเมนูการใช้งานของโปรแกรมเกมบนโทรศัพท์มือถือ โดยจะแสดงในส่วนของหน้าจอของการจำลอง (Emulator) โดยจะใช้โปรแกรม Android SDK เป็นโปรแกรมที่ใช้ในการคอมไพล์โค้ดของโปรแกรม และทำเป็นแพ็คเกจแล้ว จะได้ไฟล์ที่มีนามสกุล .apk เพื่อให้สามารถโหลดลงบนโทรศัพท์มือถือได้

4.1 ผลการทดสอบโปรแกรมแต่ละขั้นตอนการเขียน

1. การเขียนแอปพลิเคชั่นแอนดรอยด์ให้รองรับการรับอินพุตบนจอทัชสกรีนแบบหลายปุ่ม (Multitouch)

โดยจะออกแบบให้เมธอด onTouchEvent() ในคลาส MutitouchView รองรับค่าอินพุตจากผู้ใช้ได้พร้อมกันสองค่า โดยแต่ละค่าคือปุ่มกดฝั่งซ้าย และฝั่งขวา โดยการสัมผัสของผู้ใช้อยู่ในขอบเขตของปุ่มใด ก็จะอ่านค่าตามปุ่มนั้น โดย

Key1	Key2
ไม่กดปุ่มใดๆ = 0	ไม่กดปุ่มใดๆ = 0
ปุ่มซ้าย = 1	ปุ่ม A = 1
ปุ่มขวา = 2	ปุ่ม B = 2
ปุ่มบน = 3	
ปุ่มล่าง = 4	

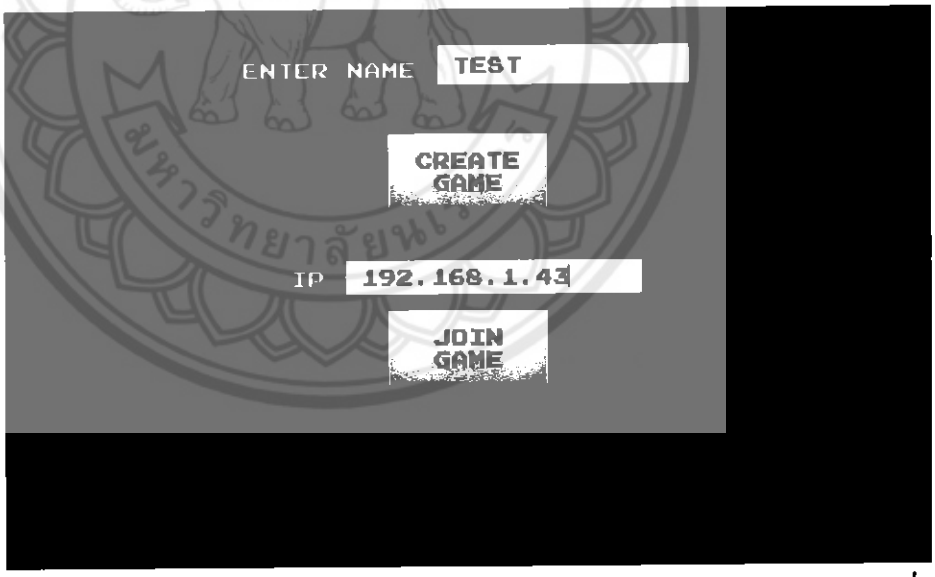


ภาพที่ 4-1 รูปแสดงขณะผู้ใช้กดปุ่มลูกศรบน และปุ่ม A พร้อมกัน

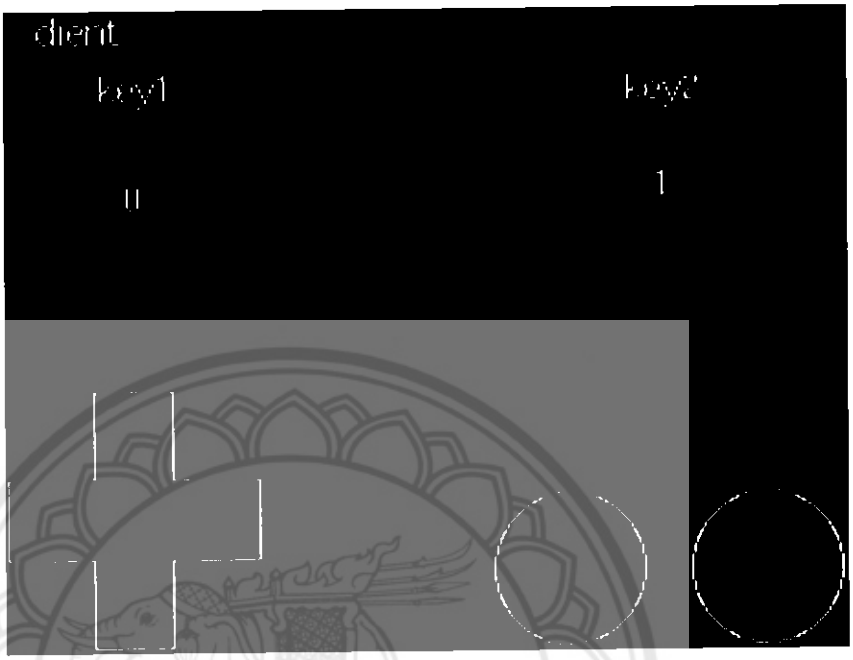
2. การเขียนแอปพลิเคชั่นแอนดรอยให้ส่งข้อมูลหากันผ่านเน็ตเวิร์ก (Network) โดยใช้ซ็อกเก็ต (Socket)

โดยจะทำการเขียนคลาส Server และ Client โดยคลาส Server จะทำการ เปิดซ็อกเก็ตเซิฟเวอร์ (Socket Server) บน Port 6000 จากนั้นจะเปิดซ็อกเก็ต (Socket) เพื่อรอการเชื่อมต่อจากไคลเอนต์ (Client) เมื่อไคลเอนต์ (Client) ทำการเชื่อมต่อเข้ามาแล้ว จะรอรับค่าปุ่มกดที่ไคลเอนต์ (Client) ส่งมา แล้วนำมาแสดงผล และส่งข้อมูลนั้นกลับไปหาไคลเอนต์ (Client)

ในส่วนของไคลเอนต์ (Client) จะทำการเชื่อมต่อกับเซิฟเวอร์ (Server) โดยการใส่ไอพีแอดเดรสของเครื่องเซิฟเวอร์ (Server) เพิ่มเชื่อมต่อผ่านซ็อกเก็ต (Socket) ทำการเชื่อมต่อเข้ามาแล้ว ผู้ใช้สามารถกดปุ่มใดๆ เพื่อส่งค่าปุ่มกดไปหาเซิฟเวอร์ (Server) และรอรับข้อมูลจากเซิฟเวอร์ (Server) เพื่อนำมาแสดงผล



ภาพที่ 4-2 รูปแสดงไคลเอนต์(Client) ทำการใส่ IP Address ของเซิฟเวอร์ (Server) เพื่อทำการเชื่อมต่อ



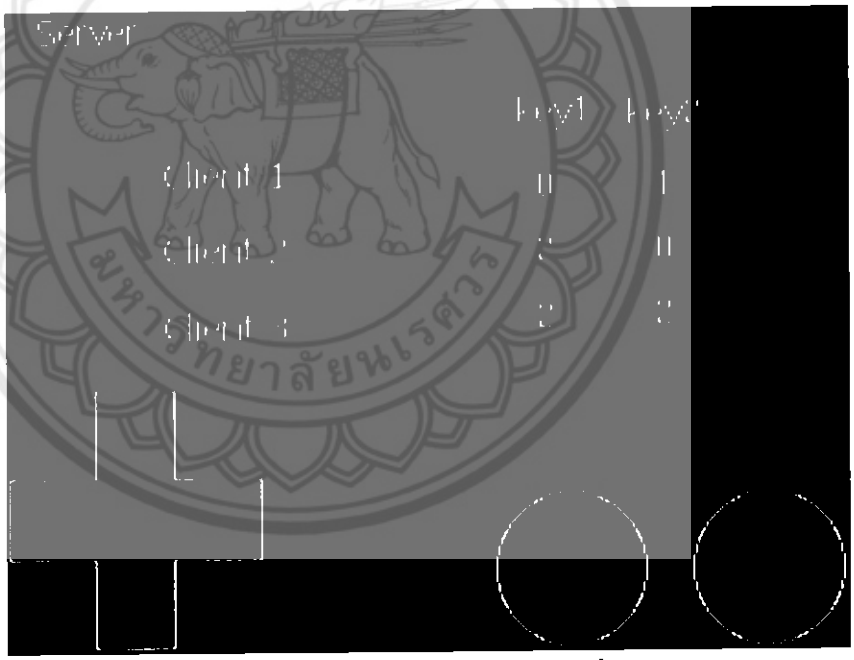
ภาพที่ 4-3 รูปหน้าจอไคลเอนต์ (Client) แสดงผลค่าปุ่มกดที่ได้รับกลับมาจากเซิฟเวอร์ (Server)

ภาพที่ 4-4 รูปหน้าจอเซิฟเวอร์ (Server) แสดงผลค่าปุ่มกดที่ได้รับจากไคลเอนต์ (Client)

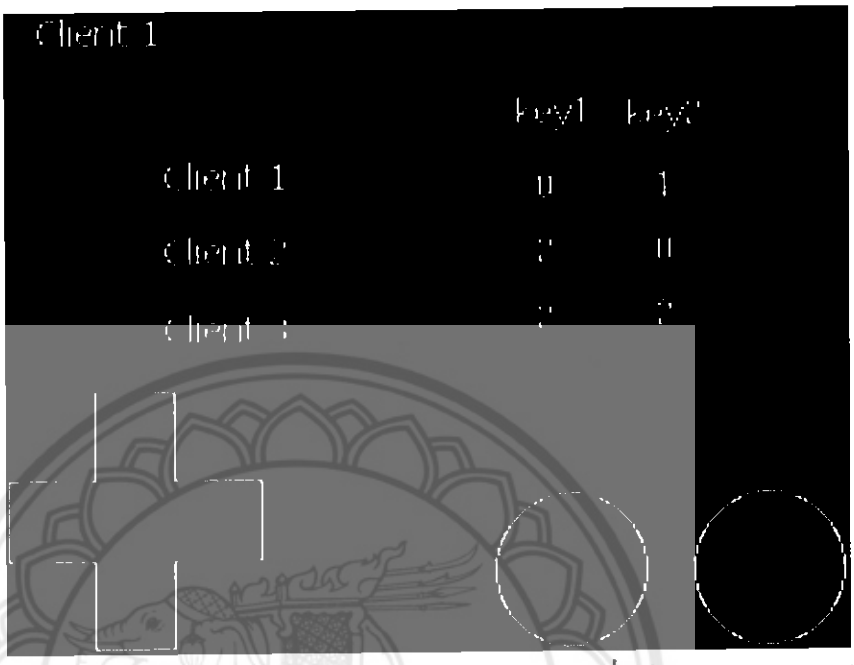
3. การเขียนแอปพลิเคชั่นแอนดรอยด์ให้เซิฟเวอร์ (Server) สามารถรองรับการเชื่อมต่อไคลเอนต์ (Client)ได้พร้อมกันมากกว่า 1 คน

โดยทำการเขียนคลาสที่ชื่อว่า ClientWorker เพิ่มเติมจากคลาส Server โดย ออบเจ็ค (Object) 1 ออบเจ็ค (Object) ของคลาส ClientWorker จะทำการเปิดซ็อกเก็ต (Socket) ของตัวเอง เพื่อรองรับการเชื่อมต่อจากไคลเอนต์ (Client) ซึ่งเซิฟเวอร์ (Server) จะสร้างออบเจ็ค (Object) นั้น ออกมาตามจำนวนไคลเอนต์ (Client) ที่ทำการเชื่อมต่อ

ส่วนการส่งข้อมูลและการแสดงผล เมื่อเซิฟเวอร์ (Server) ได้รับข้อมูลปุ่มกดจากไคลเอนต์ (Client) คนใด เซิฟเวอร์ (Server) จะส่งข้อมูลนั้นไปหาไคลเอนต์ (Client)ทุกคน แล้วแสดงผลข้อมูลนั้นออกทางหน้าจอ



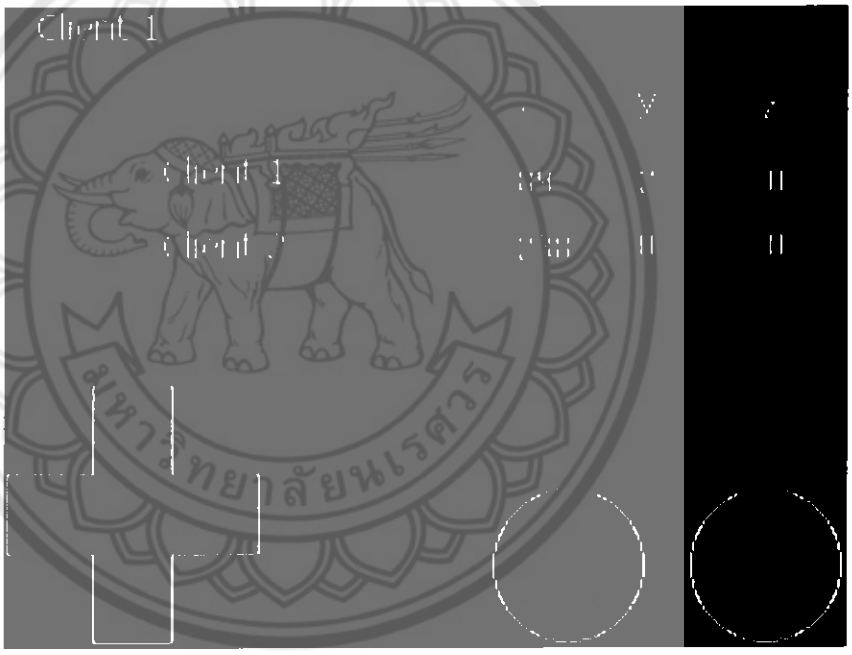
ภาพที่ 4-5 รูปหน้าจอเซิฟเวอร์ (Server) แสดงผลค่าปุ่มกดที่ได้รับจากไคลเอนต์ (Client) แบบหลายผู้เล่น



ภาพที่ 4-6 รูปหน้าจอไคลเอนต์ (Client) แสดงผลค่าปุ่มกดที่ได้รับกลับมาจากเซิฟเวอร์ (Server) แบบหลายผู้เล่น

4. การเขียนแอปพลิเคชั่นแอนดรอยในส่วนของตัวเกม

เพิ่มคลาสที่ทำงานเกี่ยวกับตัวเกม คือ GameServer โดยจะถูกเรียกใช้โดยเซิฟเวอร์ (Server) โดยจะนำข้อมูลปุ่มกดมาจากไคลเอนต์ (Client) แล้วนำมาคำนวณ โดยหากไคลเอนต์ (Client) กดปุ่ม A หรือ B จะทำการเพิ่มค่า x ของไคลเอนต์ (Client) นั้น หากไคลเอนต์ (Client) กดปุ่ม ขึ้น หรือ ลง จะทำการเพิ่มหรือลด ค่า y แล้วส่งข้อมูลที่คำนวณได้ กลับไปหาไคลเอนต์ (Client) ทุกคน แล้วทำการหยุดพัก 25 มิลลิวินาที แล้วก็นำข้อมูลปุ่มกดมาจากไคลเอนต์ (Client) มาคำนวณอีกครั้ง วนไปเรื่อยๆ



ภาพที่ 4-7 รูปหน้าจอ แสดงผลค่าตำแหน่งของผู้เล่น

5. การเขียนแอปพลิเคชั่นแอนดรอยในส่วนของการแสดงผลตัวเกม

ในส่วนของการแสดงผลไคลเอนต์ (Client) จะนำข้อมูลที่เซิฟเวอร์ (Server) ส่งมามาเก็บไว้ จากนั้น จะใช้เมธอด onDraw() ในคลาส MutitouchView จะนำข้อมูลมาแสดงผลเป็นภาพ แล้วทำการหยุดพัก 25 มิลลิวินาทีเช่นเดียวกัน แล้วนำข้อมูลที่เซิฟเวอร์ (Server) ส่งมาใหม่ มาแสดงผลอีกครั้ง

ภาพที่ 4-8 รูปหน้าจอ แสดงผลค่าตำแหน่งของผู้เล่นส่วนของการแสดงผลตัวเกม

4.2 การเข้าสู่โปรแกรม

การเข้าสู่โปรแกรมนั้นสามารถทำได้โดยใช้โปรแกรมจำลอง (Emulator) ที่ชื่อ Android Virtual Device ซึ่งเป็นโปรแกรมในการทดสอบภาษาจาวา (JAVA) ว่าที่เขียนโปรแกรมเกมขึ้นมาว่าถูกต้องหรือไม่ โดยจะมีการตรวจสอบของโค้ดที่เขียนขึ้นมา และทำเป็นแพ็คเกจเพื่อใช้ติดตั้งบนโทรศัพท์มือถือ การเข้าสู่เกมบนโทรศัพท์มือถือระบบปฏิบัติการแอนดรอยด์ (Android Operating System) แบบหลายผู้เล่น จะมีการทำงานทำงานโดยประกอบด้วยตัวเมนูของเกมและการเล่นเกม โดยเมนูจะมีการแบ่งการทำงานดังนี้

4.2.1 หน้าจอเริ่มต้นเกม (Intro Game)

4.2.2 หน้าเลือกประเภทของการเล่นและใส่ชื่อตัวละครและโฮสไอพี (Host ip)

4.2.3 หน้าของผู้เล่นที่เป็นโฮส (Host)

4.2.4 หน้าของผู้เล่นที่เป็นไคลเอนต์ (Client)

4.1.5 หน้าจอล็อบบี้ (Lobby Room) และเลือกจำนวนรอบสนามแข่ง

4.2.6 หน้าจอแสดงชื่อสนามและรายชื่อผู้เล่น

4.2.7 หน้าจอตัวเกม

4.2.8 หน้าจอแสดงผลคะแนนรวม

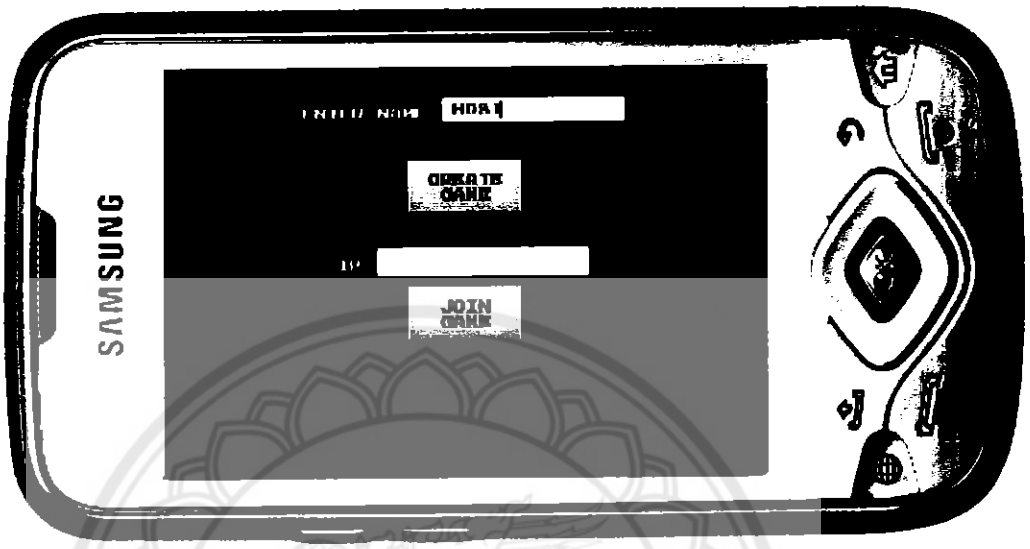
4.2.1 หน้าจอเริ่มต้นเกม (Intro Game)

เป็นตัวเปิดของเกมบนโทรศัพท์มือถือ

ภาพที่ 4-9 หน้าจอเริ่มต้นเกม (Intro Game)

เมื่อทำการสัมผัสหน้าจอจะไปยังหน้าถัดไป

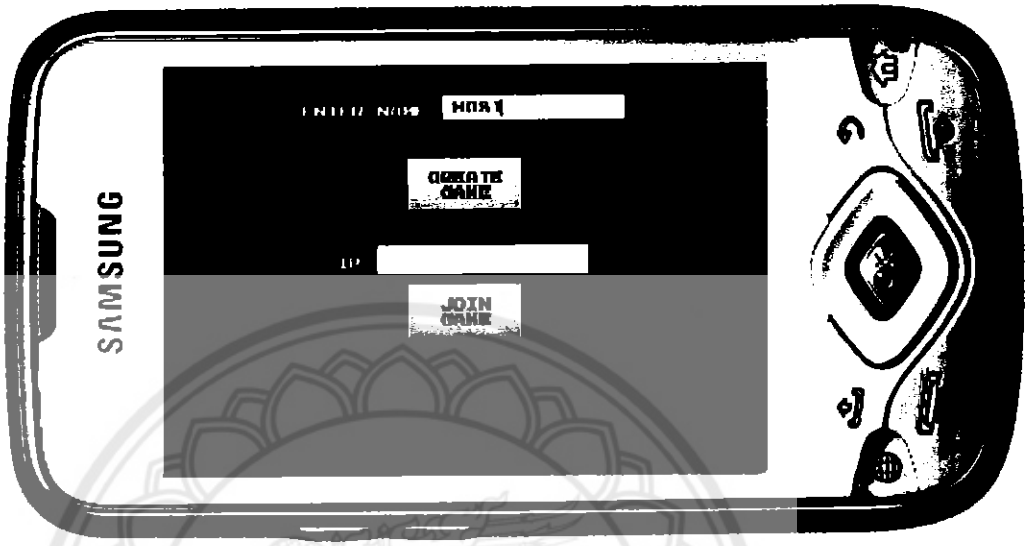
4.2.2 หน้าเลือกประเภทของการเล่นและใส่ชื่อตัวละครและ โฮสไอพี (Host ip)



ภาพที่ 4-10 หน้าจอเลือกประเภทของการเล่นและใส่ชื่อตัวละครและ โฮสไอพี (Host ip)

ในหน้าจอนี่ผู้เล่นจะต้องทำการกำหนดชื่อของตัวละคร และเลือกชนิดของการเล่นโดย ถ้าเป็นเครื่องโฮส (Host) ให้เลือก Create Game แต่ถ้าเป็นเครื่องไคลเอนต์ (Client) ให้ทำการใส่เลขไอพี (IP) ของเครื่องโฮส (Host) แล้วทำการเลือก Join Game

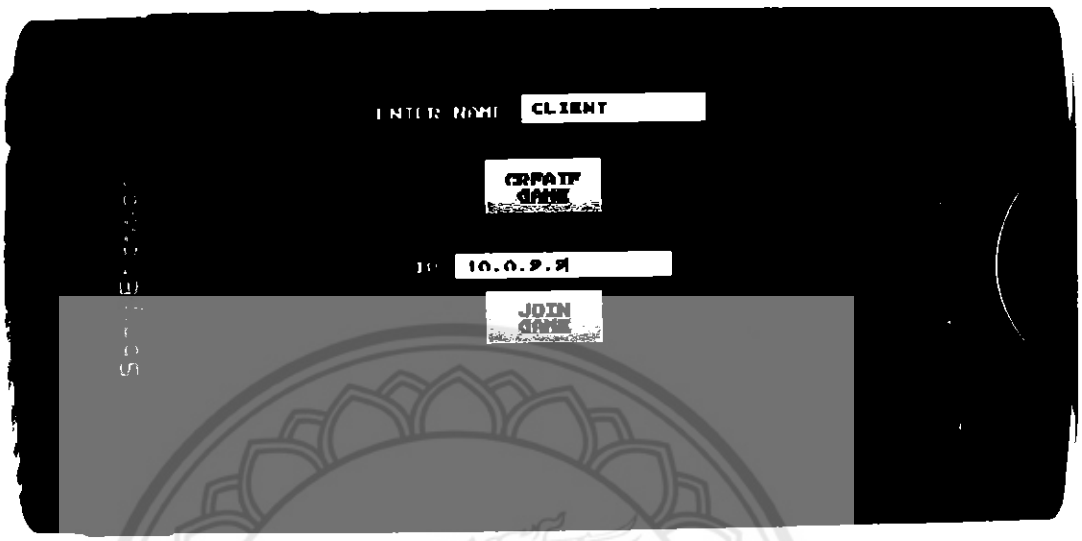
4.2.3 หน้าของผู้เล่นที่เป็นโฮส (Host)



ภาพที่ 4-11 หน้าจอของผู้เล่นที่เป็นโฮส (Host)

เมื่อผู้เล่นเลือกเป็นโฮส (Host) ก็จะเข้ามาสู่ หน้าของ หน้าจอล็อบบี้ (Lobby Room) ซึ่งในหน้านี้โปรแกรมจะทำการสร้างซ็อกเก็ต (Socket) ขึ้นมาหนึ่งช่อง เพื่อรอรับการเชื่อมต่อจากไคลเอนต์ (Client)

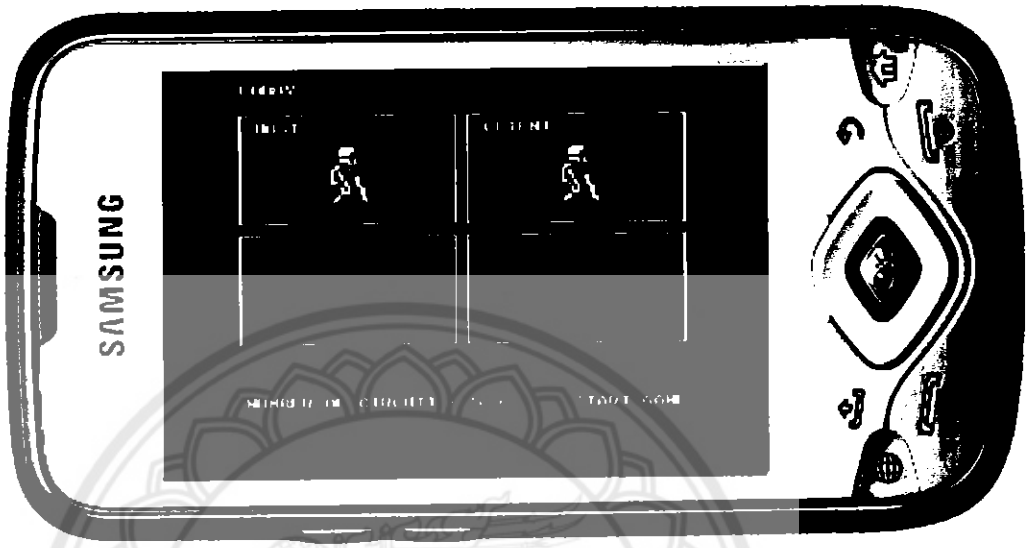
4.2.4 หน้าของผู้เล่นที่เป็นไคลเอนต์ (Client)



ภาพที่ 4-12 หน้าจอของผู้เล่นที่เป็นไคลเอนต์ (Client)

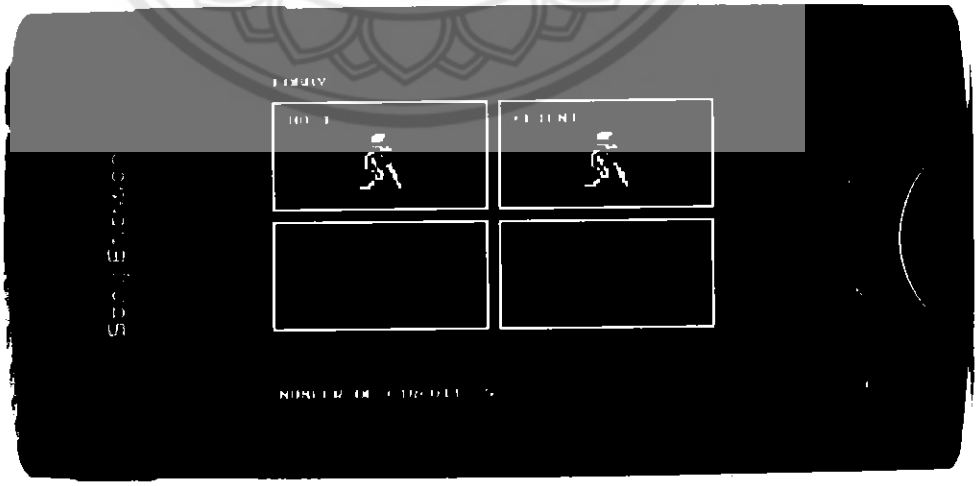
เมื่อผู้เล่นที่เป็นไคลเอนต์ (Client) เชื่อมต่อเข้ามาโปรแกรม ก็จะทำการสร้างซ็อกเก็ต (Socket) ขึ้นมาอีกหนึ่งช่อง เพื่อรอรับการเชื่อมต่อจากไคลเอนต์ (Client) เครื่องต่อไปเรื่อยๆ จนกว่าผู้เล่นในห้องจะเต็มหรือเมื่อโฮส (Host) สั่งเริ่มเกม

4.2.5 หน้าจอล็อบบี้ (Lobby Room) และเลือกจำนวนรอบสนามแข่ง



ภาพที่ 4-13 หน้าจอของผู้เล่นที่เป็นโฮส(Host) เมื่อมีไคลเอนต์ (Client)เชื่อมต่อหา

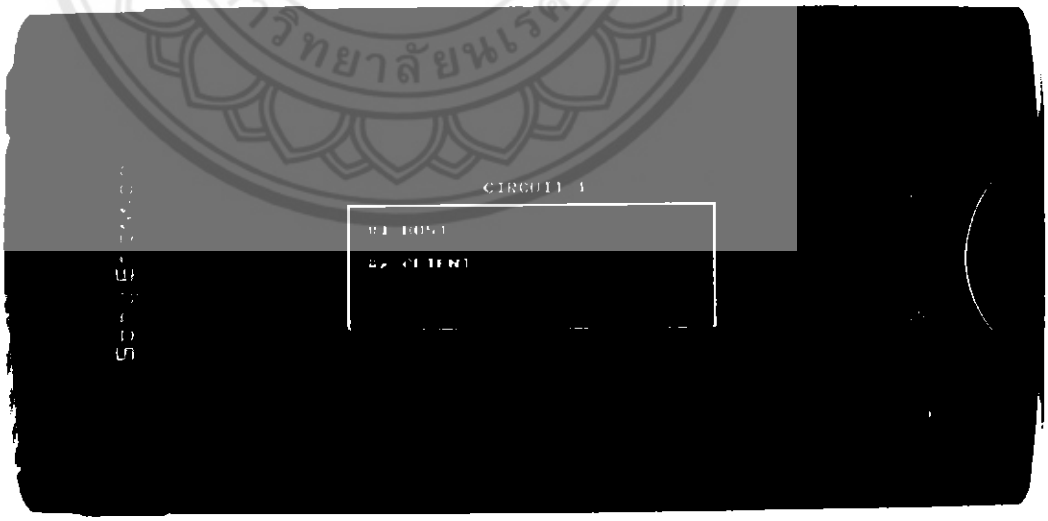
เมื่อผู้เล่นเลือกประเภทเรียบร้อยแล้วจะเข้ามาภายในหน้านี้โดยผู้เล่นที่เป็นโฮส (Host) จะสามารถเลือกจำนวนรอบสนามของการแข่งขัน และสั่งเริ่มเกมได้จากหน้านี้



ภาพที่ 4-14 หน้าจอของผู้เล่นที่เป็นไคลเอนต์ (Client) เมื่อทำการเชื่อมต่อกับโฮส (Host) สำเร็จ

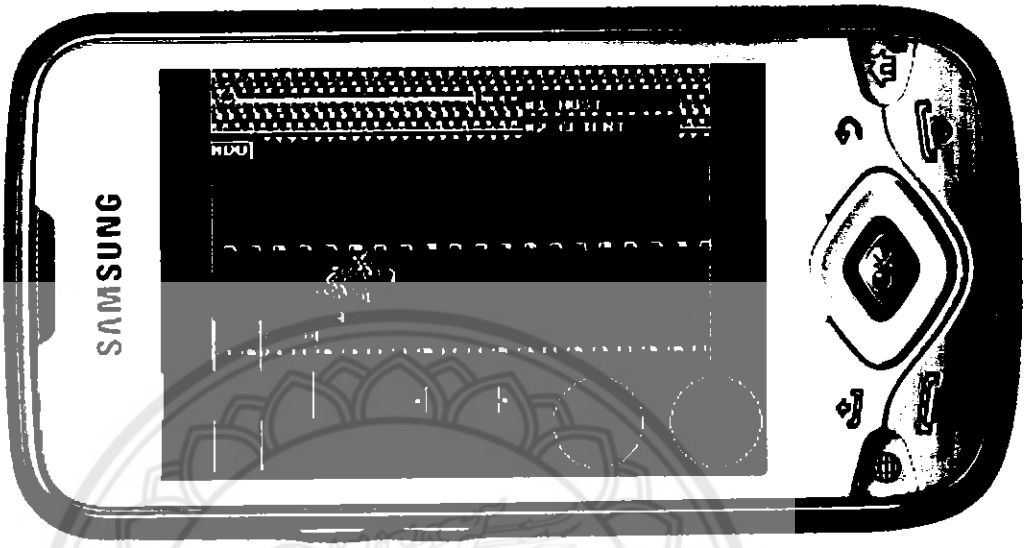
4.2.6 หน้าจอแสดงชื่อสนามและรายชื่อผู้เล่น

ภาพที่ 4-15 หน้าจอแสดงชื่อสนามและรายชื่อผู้เล่นที่เป็นโฮส (Host)



ภาพที่ 4-16 หน้าจอแสดงชื่อสนามและรายชื่อผู้เล่นที่เป็นไคลเอนต์ (Client)

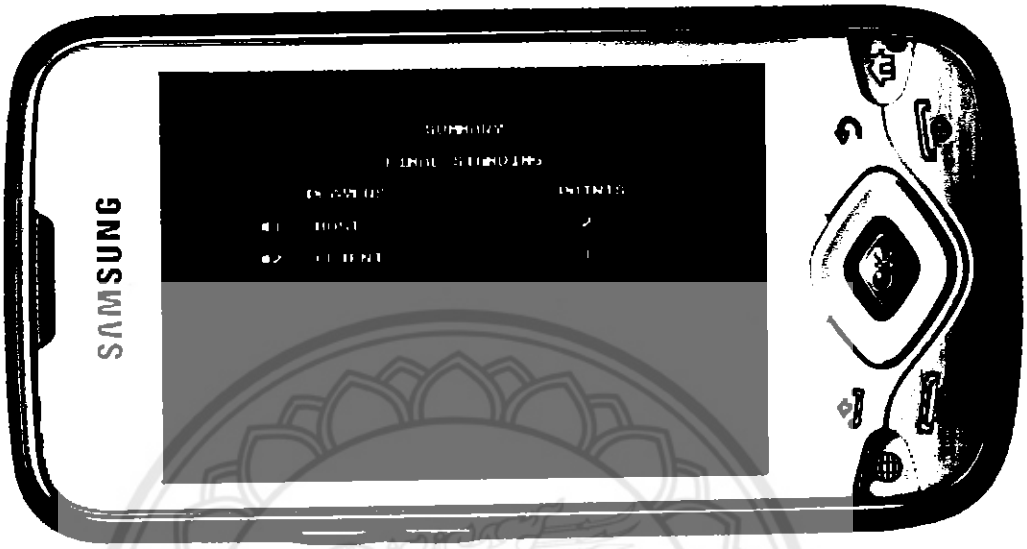
4.2.7 หน้าจอตัวเกม



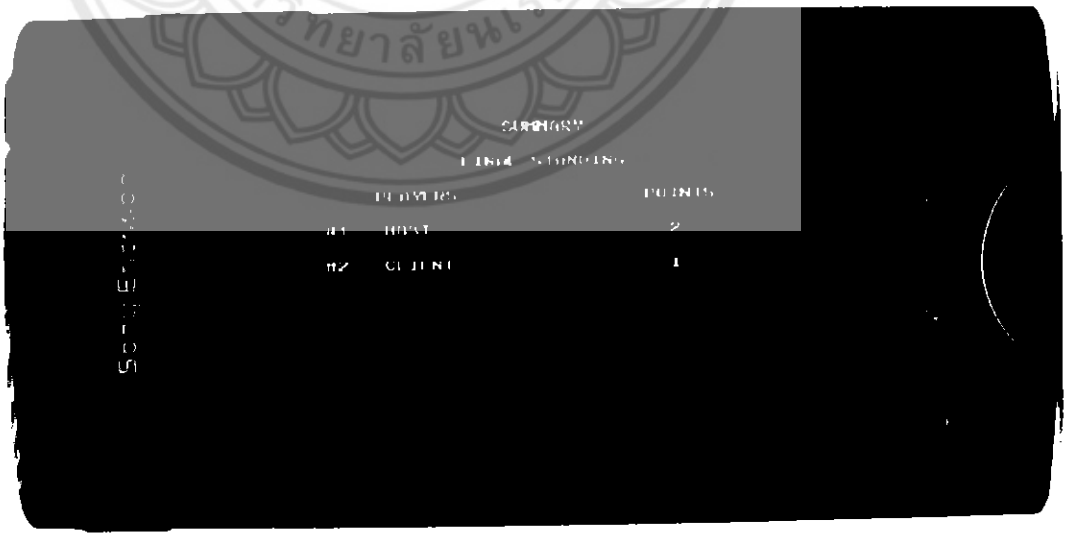
ภาพที่ 4-17 หน้าจอเกมของเครื่องที่เป็นโฮส (Host)

ภาพที่ 4-18 หน้าจอเกมของเครื่องที่เป็นไคลเอนต์ (Client)

4.2.8 หน้าจอแสดงผลคะแนนรวม



ภาพที่ 4-19 หน้าจอแสดงผลคะแนนรวมของเครื่องที่เป็นโฮส (Host)



ภาพที่ 4-20 หน้าจอแสดงผลคะแนนรวมของเครื่องที่เป็นไคลเอนต์ (Client)

บทที่ 5
สรุป ปัญหา วิจารณ์ผล

5.1 บทสรุป

การพัฒนาเกมบนโทรศัพท์เคลื่อนที่โดยใช้เทคโนโลยีระบบเครือข่ายไร้สาย (Wireless LAN) ในการสื่อสาร ทางผู้จัดทำได้ทราบถึงข้อจำกัดในการพัฒนาเกม ทั้งในเรื่องของหน่วยความจำของโทรศัพท์มือถือ มาตรฐานการรองรับการทำงานของมือถือแต่ละรุ่น ที่มีข้อจำกัดแตกต่างกันไป และข้อจำกัดต่าง ๆ ของเทคโนโลยีระบบเครือข่ายไร้สาย (Wireless LAN) ทำให้ทางผู้จัดทำ ได้มีการปรับปรุงการพัฒนาเกมให้สามารถเล่นบนมือถือที่มีการรองรับด้วยมาตรฐานเดียวกันได้ เพื่อให้การพัฒนาเกมมีความสมบูรณ์

ในด้านการพัฒนาเกมบนมือถือ ทางผู้พัฒนาจะต้องมีการวิเคราะห์การออกแบบองค์ประกอบของตัวเกม กติกาของเกม วิธีการเล่นเกม และรูปแบบการทำงานของเกม เพื่อให้เกมมีความสมบูรณ์มากที่สุด และสามารถใช้งานได้จริงโดยจะมีวิธีที่จะรวบรวมข้อมูลของการสร้างภาพกราฟฟิก และรายละเอียดของตัวเกม การเก็บข้อมูลคะแนน การทำงานตั้งแต่การสื่อประสานกับผู้ใช้ (User Interface) ไปจนถึงการจบการเล่นเกม

5.2 ปัญหาที่เกิดขึ้นในโครงงาน

ปัญหาที่เกิดขึ้นกับโครงงาน เกิดจากการขาดประสบการณ์ในการใช้เทคโนโลยีระบบเครือข่ายไร้สาย (Wireless LAN) ในการเชื่อมต่อ และทักษะในการสร้างเกมโดยใช้ภาษาจาวา (JAVA) ทำให้มีการพัฒนาโปรแกรมมีความล่าช้า และในโครงงานนี้จากการพัฒนาเกม มีข้อผิดพลาดเกิดขึ้นหลายอย่าง ได้แก่

5.2.1 ปัญหาเรื่องภาพพิกเซล (Pixel) จากการซิมูเลเตอร์ (Simulator) มีขนาดไม่พอดีกับมือถือที่นำมาใช้ในการพัฒนาเกม จากการได้ศึกษาถึงเกมต่าง ๆ ทำให้รู้ว่าเกมแต่ละเกมนั้นจะมีรุ่นของโทรศัพท์เป็นตัวกำหนด โดยโทรศัพท์แต่ละรุ่นนั้นจะมีขนาดหน้าจอไม่เท่ากัน ทำให้เกมที่ได้โปรแกรมขึ้นมานี้ ไม่อาจสามารถขยายให้ใหญ่ หรือย่อขนาดอัตโนมัติเองตามหน้าจอของโทรศัพท์มือถือที่ทำการรันโปรแกรมเกมได้

5.2.2 เนื่องจากการพัฒนาและทดสอบโปรแกรมนั้นทำการพัฒนาและทดสอบโปรแกรมโดยใช้การจำลองเครื่องโทรศัพท์มือถือระบบปฏิบัติการแอนดรอยด์ในคอมพิวเตอร์การทำการทดสอบโปรแกรมจึงค่อนข้างช้าและค้างบ้างเป็นบางครั้งเนื่องประสิทธิภาพของเครื่องที่ใช้ในการทดสอบมีน้อย

5.3 วิเคราะห์ผลการดำเนินงาน

5.3.1 ความสามารถของเกมที่ทำการพัฒนา

การพัฒนาเกมโทรศัพท์มือถือบนระบบปฏิบัติการแอนดรอยด์โดยใช้เชื่อมต่อกันผ่านระบบเครือข่ายไร้สายนั้นมีข้อดีต่างๆ ที่แตกต่างกับการเชื่อมต่อแบบอื่นหลายข้อ ได้แก่

- การเชื่อมต่อกันผ่านระบบเครือข่ายไร้สาย (Wireless LAN) นั้นสามารถเชื่อมต่อและเล่นพร้อมกันได้ครั้งละหลายๆคน
- เกมที่พัฒนาโดยใช้การเชื่อมต่อกันผ่านระบบเครือข่ายไร้สาย (Wireless LAN) สามารถเล่นได้ทุกที่ทุกเวลาโดยที่ไม่ต้องทำการเชื่อมต่ออินเตอร์เน็ต
- การเชื่อมต่อและเล่นเกมสามารถเชื่อมต่อและเล่นกับผู้เล่นที่อยู่ไกลออกไปได้เหมือนกันกับการเชื่อมต่อผ่านเซิฟเวอร์ (Server) โดยการเชื่อมต่อผ่านทางอินเตอร์เน็ต

5.3.2 ข้อจำกัดของเกมที่ทำการพัฒนา

ระบบของเกมที่ทำการพัฒนามีข้อจำกัดของการพัฒนาและการใช้งานดังต่อไปนี้

- เนื่องจากการประมวลผลต่างๆ ภายในเกมใช้เครื่องโฮส (Host) เป็นตัวประมวลผลหลัก เครื่องโฮส (Host) จึงต้องมีความสามารถในการประมวลผลที่สูง
- จากเหตุผลในข้อที่แล้วและความไม่ชำนาญในการพัฒนาระบบ ทำให้เมื่อเล่นพร้อมกันหลายคนระบบเกิดการหน่วงและช้า จึงสามารถรองรับผู้เล่นได้สูงสุดเพียง 4 คน
- ทางผู้จัดทำได้ใส่ขนาดของหน้าจอโทรศัพท์ที่ใช้ในการเล่นเกมไว้ 3 ขนาดคือ 320x240, 480x320, 800x480 พิกเซล หากทำการเล่นในหน้าจอที่ใหญ่กว่าขนาดที่กำหนดจะทำให้การแสดงผลภาพไม่เต็มหน้าจอ
- เกมสามารถเล่นได้บนโทรศัพท์มือถือระบบปฏิบัติการแอนดรอยด์ เวอร์ชั่น 2.2 ขึ้นไป

5.4 แนวทางการพัฒนาในอนาคต

ในอนาคตโทรศัพท์มือถือระบบปฏิบัติการแอนดรอยด์จะได้รับการพัฒนาให้มีความสามารถในการทำงานมากขึ้น และหากมีการเขียนโค้ด (Code) การทำงานของโปรแกรมให้รัดกุมมากขึ้นจะทำให้การใช้งานสามารถใช้งานได้พร้อมกันหลายคนมากขึ้น และหากมีการเพิ่มขนาดของหน้าจอที่ใช้งานในโปรแกรมให้หลากหลายมากขึ้น ก็จะทำให้การใช้งานสะดวกมากยิ่งขึ้นอีกในอนาคต

เอกสารอ้างอิง

[1] จาวาเบื้องต้น. [ออนไลน์]. เข้าถึงจาก:

http://androidthai.in.th/java-intro.html.

[2] จะเขียนแอนดรอยด์ ต้องรู้ ?. [ออนไลน์]. เข้าถึงจาก:

http://androidthai.in.th/conternt-android.html.

[3] Java - Networking (Socket Programming). [ออนไลน์]. เข้าถึงจาก:

http://www.tutorialspoint.com/java/java_networking.htm.

[4] Android-introduction. [ออนไลน์]. เข้าถึงจาก:

http://tutorial.function.in.th/android-introduction

[5] ดร. จักรชัย โสอินทร์ และพงษ์ศธร จันทร์ยอย. Basic Android App Development.

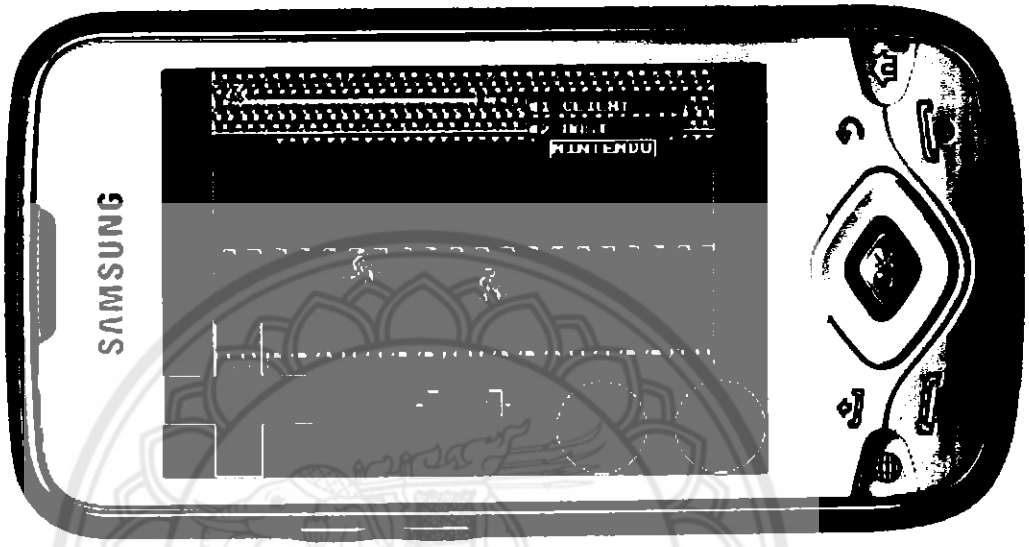
[6] James Steele and Nelson To. The Android Developer's Cookbook.

[7] ธีรวัฒน์ ประกอบผล. (2553) .คู่มือการเขียนโปรแกรมภาษา JAVA.

ภาคผนวก ก

คู่มือเกม Excitebike

วิธีการเล่น

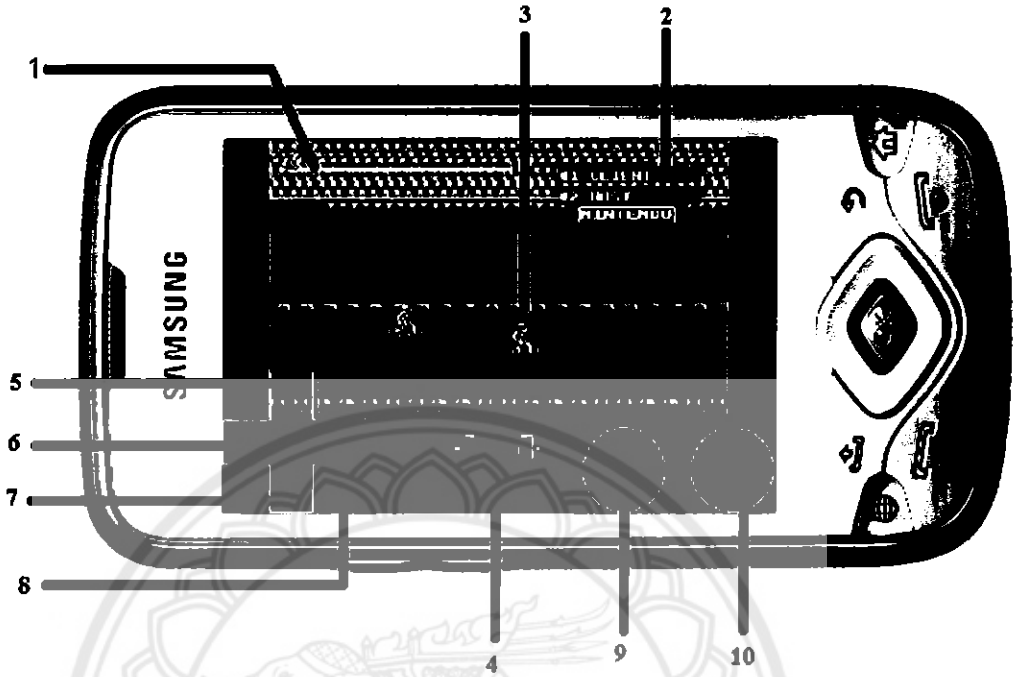


ภาพที่ ก-1 เกม Excitebike

1. เมื่อเริ่มเกม Host จะทำการกำหนดรอบของสนามแข่งขัน และเมื่อเริ่มแต่ละสนามจะมีสัญญาณปล่อยตัว ผู้เล่นแต่ละคนจะต้องบังคับรถจักรยานยนต์ของตนให้ถึงเส้นชัยโดยแข่งขันกัน
2. สิ่งที่ผู้เล่นแต่ละคนสามารถทำได้คือ
   - บังคับตัวละครให้เคลื่อนที่เปลี่ยนเลนโดยกดกดปุ่มขึ้นลง
   - บังคับตัวละครให้เคลื่อนที่ข้างหน้าโดยกด A หรือ B ค้างไว้ โดยปุ่ม A จะเป็นการเร่งความเร็ว เมื่อกดจะทำให้แถบของอุณหภูมิเพิ่มขึ้น เมื่อแถบอุณหภูมิเต็มจะทำให้เครื่องยนต์โอเวอร์ฮีตผู้เล่นจะไม่สามารถควบคุมได้ในระยะเวลาหนึ่ง

- ยกล้อโดยการกดปุ่มเคลื่อนที่ทางซ้าย และ กดหน้ารถลงโดยการกดปุ่มเคลื่อนที่ทางขวา โดยเมื่อยกล้อขณะขึ้นเนินกระโดดจะทำให้กระโดดได้สูงขึ้นและเมื่อกดหน้ารถลงขณะขึ้นเนินกระโดดจะทำให้กระโดดได้ไกลขึ้น แต่ถ้ากดยกล้อ หรือกดหน้ารถลงมากเกินมุมที่กำหนดจะทำให้รถล้ม

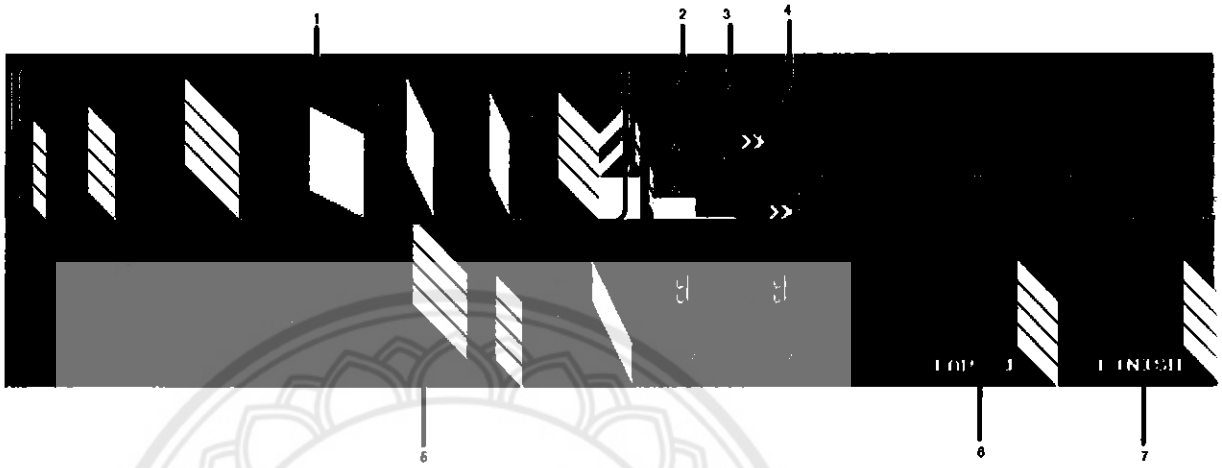
3. เมื่อเกิดการเบียดหรือชนกันจะทำให้ตัวละครของผู้เล่นล้มลง โดยผู้ที่อยู่ในตำแหน่งที่ช้ากว่าจะเป็นฝ่ายล้ม
4. เมื่อผู้เล่นขึ้นเนินกระโดดจะสามารถกระโดดเหินได้โดยระยะทางจะขึ้นอยู่กับความชันและลักษณะของเนินกับการบังคับของผู้เล่น
5. เมื่อมีผู้เล่นใดเข้าเส้นชัยแล้วระบบจะทำการนับเวลาถอดหลัง 20 วินาที เมื่อครบแล้วจะทำการนับคะแนนในสนามนั้นตามลำดับระยะทางของผู้เล่นทันที แม้จะมีผู้เล่นที่ไม่เข้าเส้นชัยก็ตาม
6. เมื่อแข่งขันครบตามจำนวนที่ Host ระบุไว้ จะทำการรวมคะแนนในแต่ละสนามและจัดอันดับผู้เล่นตามคะแนนรวม



ภาพที่ ก-2 แสดงปุ่มที่ใช้ในการเล่นเกม

1. แถบแสดงตำแหน่งของผู้เล่น ระยะทางที่วิ่งมา และระยะทางที่เหลือ
2. แถบแสดงลำดับที่ และรายชื่อของผู้เล่นแต่ละคน
3. ตัวละคร
4. แถบแสดงค่าอุณหภูมิเมื่อเต็มจะทำให้เกิดโอเวอร์ฮีต ผู้เล่นไม่สามารถควบคุมได้ระยะเวลาหนึ่ง
5. ปุ่ม ขึ้น ใช้เพื่อเปลี่ยนเลนไปทางเลนด้านซ้าย
6. ปุ่ม เคลื่อนที่ทางซ้าย ใช้เพื่อบังคับให้ตัวละครยกล้อหน้ารถขึ้นเพื่อให้สามารถกระโดดได้สูงขึ้นเมื่อขึ้นเนินกระโดด แต่หากกดปุ่มมากเกินไปจนเกินมุมที่กำหนดจะทำให้รถล้ม
7. ปุ่ม ลง ใช้เพื่อเปลี่ยนเลนไปทางเลนด้านขวา
8. ปุ่ม เคลื่อนที่ทางขวา ใช้เพื่อบังคับให้ตัวละครกดล้อหน้ารถลงเพื่อให้สามารถกระโดดได้ไกลขึ้นเมื่อขึ้นเนินกระโดด แต่หากกดปุ่มมากเกินไปจนเกินมุมที่กำหนดจะทำให้รถล้ม
9. ปุ่ม A ใช้เพื่อให้รถเคลื่อนที่ไปข้างหน้าแบบเร่งความเร็วเมื่อกดใช้จะทำให้แถบอุณหภูมิเพิ่มขึ้นเรื่อยๆ
10. ปุ่ม B ใช้เพื่อให้รถเคลื่อนที่ไปข้างหน้าแบบปกติ

อุปสรรคภายในเกม



ก-3 แสดงอุปสรรคภายในเกม

1. เนินกระโดด เนินกระโดดแต่ละเนินจะให้ระยะกระโดดและความสูงของการกระโดดไม่เท่ากันขึ้นอยู่กับความชันและการบังคับ
2. ที่กั้นทาง เมื่อเคลื่อนที่ผ่านที่กั้นทางโดยไม่ยกล้อจะทำให้ตัวละครของผู้เล่นล้ม
3. ยางมะตอย เมื่อเคลื่อนที่ผ่านยางมะตอยจะทำให้ตัวละครของผู้เล่นเคลื่อนที่ช้าลง
4. สัญลักษณ์ลดTEMP เมื่อผู้เล่นเคลื่อนที่ผ่านสัญลักษณ์ลดTEMP แถบอุณหภูมิของผู้เล่นจะถูกรีเซ็ทใหม่
5. เนินชัน จะมีลักษณะและความชันที่แตกต่างกันตามแต่ละชนิด
6. LAP 1 เป็นตัวบอกว่าผู้เล่นได้เล่นผ่านมาถึงครึ่งทางแล้ว
7. FINISH เป็นตัวบอกว่าผู้เล่นมาถึงเส้นชัยแล้ว