The First Chapter of love

จัดทำโดย

นาย ณัฐชนนท์ ศิรมลพิวัฒน์ 1670700044 Section 227E

นาย จีรภัทร วัชรมูล 1670703162 Section 227E

นาย อัครวินท์ เสรีบุษกร 1670703311 Section 227E

นาย เมฆินทร์ ขันธวิชัย 1670703675 Section 227E

อาจารย์ผู้สอน

อาจารย์สราวุธิ ราษฎร์นิยม

หลักสูตรวิทยาศาสตร์บัณฑิต

ภาคการศึกษาที่ 1 ปีการศึกษา 2567

เอกสารนี้เป็นส่วนหนึ่งของวิชา CS318 Object-Oriented Programming

ภาควิชาวิทยาการคอมพิวเตอร์ คณะเทคโนโลยีสารสนเทศและนวัตกรรม

สารบัญ

บทที่ 1 บทนำ 3

บทที่ 2 การออกแบบหน้าจอ และ ขั้นตอนการทำงาน 4-6

2.1 หน้าเมนูเกม 4

2.2 หน้าหลักเกม 5

2.3 หน้าเมนูเกม 6

บทที่ 3 การออกแบบคลาส 7-20

3.1 Class Diagram 7

3.2 อธิบายการทำงานของ Class 8-10

3.3 Class และ Jframe ทั้งหมดในโปรแกรม 11-20

บทที่ 4 การทดสอบการทำงานและปัญหาที่พบ 21

บทที่ 5 คู่มือการใช้งานระบบ 22-29

5.1 หน้าเมนูเกม (Menu game) 22

5.2 หน้าหลักเกม (Main game) (Menu game ปุ่ม New game) 23-26

5.3 หน้าเมนูเกม (Menu game) (ปุ่ม Continue) 27

5.4 หน้าเมนูเกม (Menu game) (ปุ่ม Setting) 28

5.5 หน้าเมนูเกม (Menu game) (ปุ่ม Credit) 29

5.6 หน้าเมนูเกม (Menu game) (ปุ่ม Exit) 29

บทที่ 1 บทนำ

1.1 บทนำ

ในปัจจุบัน อุตสาหกรรมเกมมีการเติบโตอย่างต่อเนื่อง โดยเฉพาะเกมแนว Visual Novel ซึ่งเป็นเกมที่เน้นการเล่าเรื่องและการโต้ตอบกับผู้เล่นผ่านตัวเลือก ซึ่งต้องอาศัยการจัดการข้อมูลที่ซับซ้อน เช่น การจัดการบทสนทนา (Dialogue), การจัดการตัวเลือก (Choices), และระบบความสัมพันธ์ตัวละคร (Relationship System)

ทางผู้จัดทำจึงได้พัฒนาโปรแกรมเกม "The First Chapter of Love" ขึ้น โดยประยุกต์ใช้ภาษา Java และไลบรารี Swing ในการสร้าง Graphic User Interface (GUI) โดยมุ่งเน้นการออกแบบโครงสร้างโปรแกรมตามหลักการ Object-Oriented Programming (OOP) เพื่อสร้าง Game Engine ขนาดเล็กที่สามารถจัดการ Scene, Character และ Game Logic ได้อย่างเป็นระบบ ซึ่งจะเป็นพื้นฐานสำคัญในการพัฒนาซอฟต์แวร์ที่มีความซับซ้อนต่อไป

1.2 วัตถุประสงค์

1. เพื่อศึกษาและประยุกต์ใช้ภาษา Java และ Java Swing ในการพัฒนาแอปพลิเคชัน

2. เพื่อออกแบบและพัฒนาระบบ Game Engine สำหรับเกมแนว Visual Novel ที่รองรับระบบตัวเลือกและการเปลี่ยนฉาก

3. เพื่อศึกษาการจัดการข้อมูลสถานะผู้เล่น (Player State) และระบบบันทึกข้อมูล (Save/Load) ด้วย Serialization

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1. ผู้จัดทำมีความรู้ความเข้าใจและสามารถประยุกต์ใช้ภาษา Java และไลบรารี Swing ในการพัฒนาแอปพลิเคชันที่มีส่วนติดต่อกราฟิก (GUI) ได้อย่างมีประสิทธิภาพ

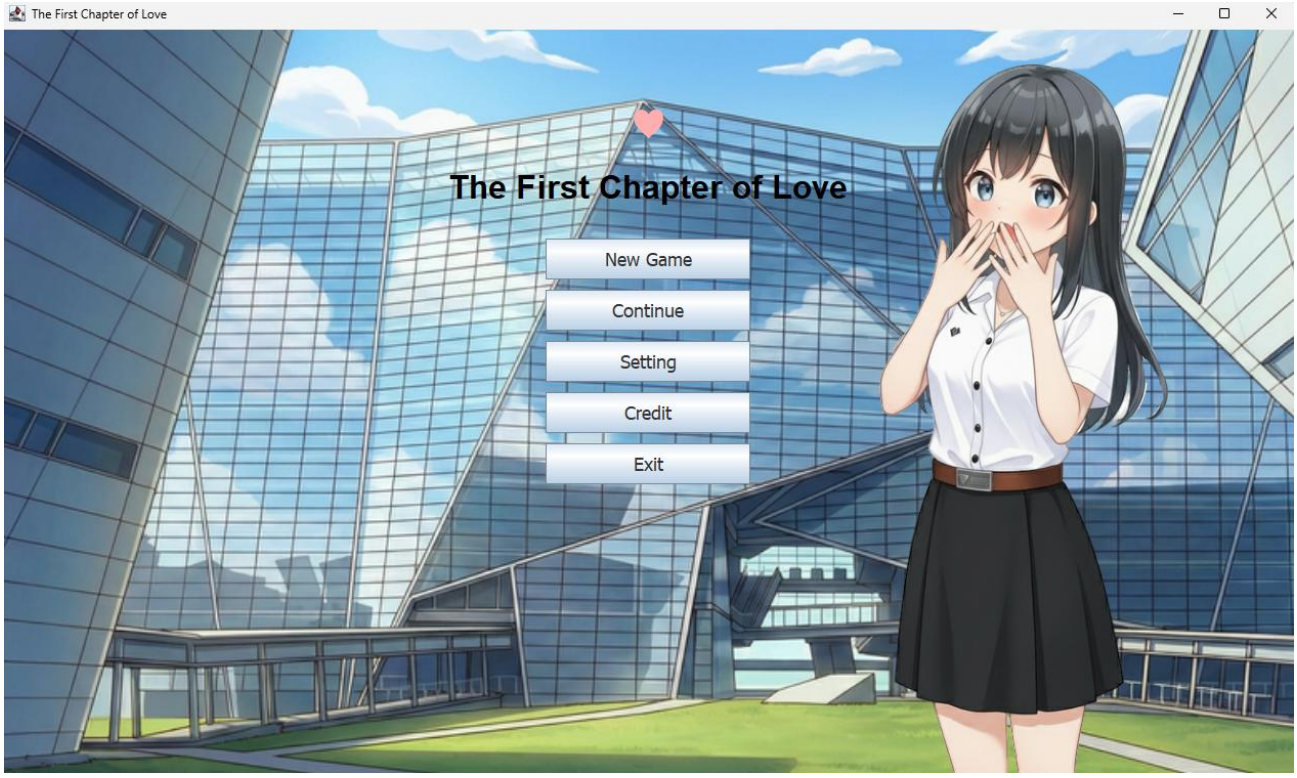
2. สามารถออกแบบและพัฒนา Game Engine ขนาดเล็ก ที่รองรับระบบบทสนทนา, ระบบตัวเลือก และการจัดการฉาก ตามหลักการ OOP ได้

3. สามารถพัฒนาระบบจัดการข้อมูลสถานะผู้เล่น (Player State) และระบบบันทึกข้อมูล (Save/Load System) เพื่อความต่อเนื่องในการใช้งานโปรแกรม

4. ได้ผลงานเกมคอมพิวเตอร์แนว Visual Novel ที่สมบูรณ์ สามารถนำไปเผยแพร่หรือใช้เป็นต้นแบบในการศึกษาการสร้างเกมต่อไปได้

บทที่ 2 การออกแบบหน้าจอ และ ขั้นตอนการทำงาน

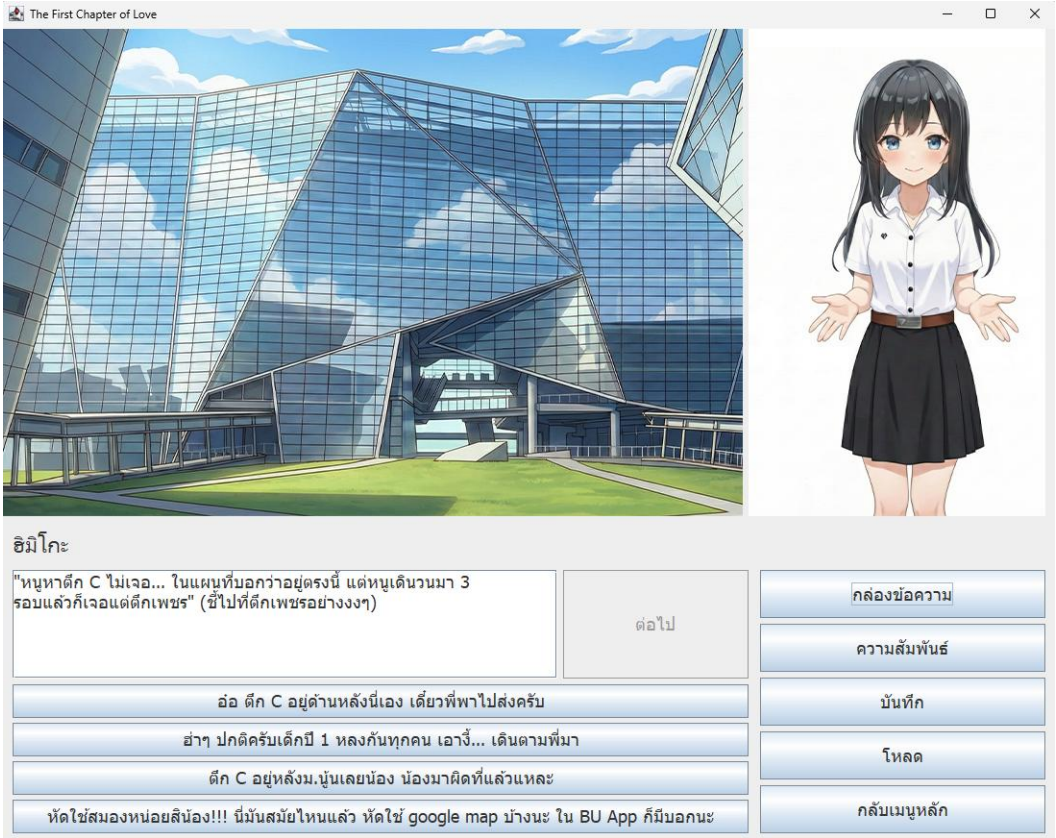
2.1 หน้าเมนูเกม (Main menu game)



2.1 ภาพหน้าเมนูเกม

โดยหน้า Main Menu จะมีปุ่มดังนี้ ปุ่ม New Game เมื่อกด จะเข้าสู่หน้าเล่นเกม แต่จะเป็นการเริ่มเกม ใหม่ทันที , ปุ่ม Continue เมื่อกด จะเข้าสู่หน้าเล่นเกม แต่จะเป็นการเริ่มจากที่ Save ล่าสุด , ปุ่ม Setting เมื่อกด หน้าต่างเล็กๆ จะเด้งขึ้นมา เพื่อลดเสียงเกมตามที่ต้องการ , ปุ่ม Credit เมื่อกด หน้าต่างเล็กๆ จะเด้งขึ้นมา พร้อมชื่อผู้จัดทำเกม, ปุ่ม Exit เมื่อกด จะเป็นการปิดตัวเกมทันที

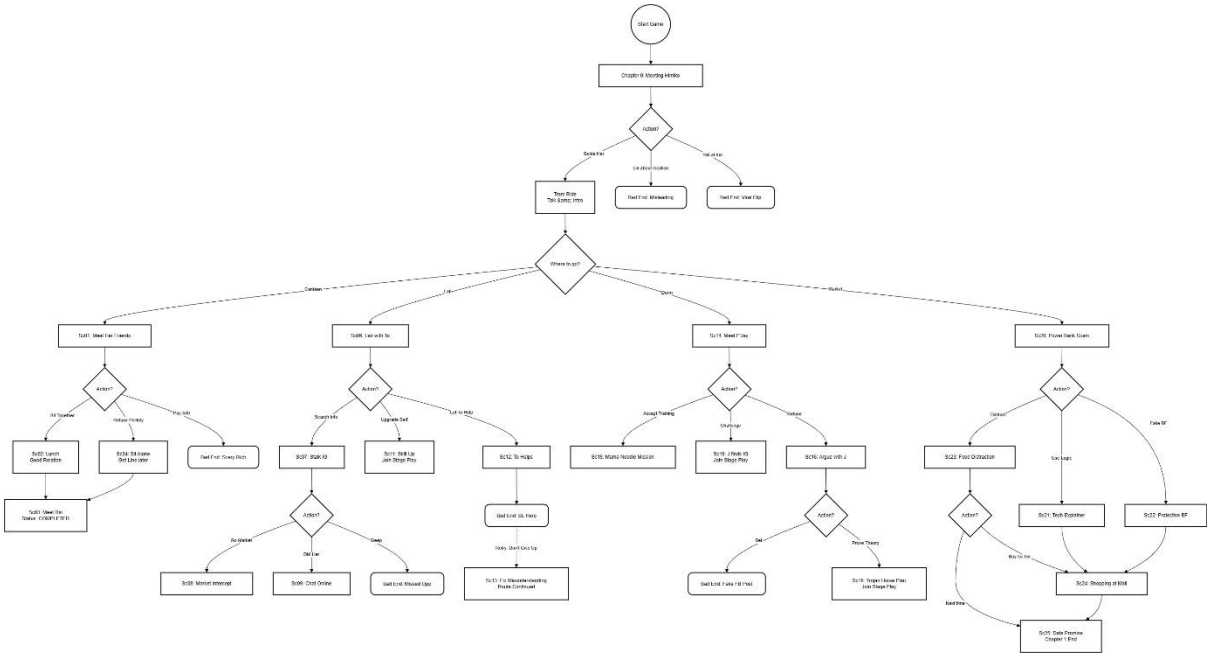
2.2 หน้าหลักเกม (Main Game)



2.2 ภาพหน้าหลักเกม

โดยหน้า Main Game จะมีปุ่มดังนี้ ปุ่ม ต่อไป เมื่อกด จะเข้าสู่หน้าเล่นเกม , ปุ่ม ตัวเลือก 4 ตัวเลือก จะเป็นการเลือกเพื่อไปในเหตุการณ์ต่างๆ , ปุ่ม กล่องข้อความ แสดง Dialog Log หรือ ข้อความเพื่อย้อนกลับไปดูข้อมูลความย้อนหลัง , ปุ่ม บันทึก คือการบันทึกเนื้อเรื่อง ในจุดที่เล่นอยู่ , ปุ่ม โหลด เพื่อโหลดข้อมูลที่บันทึกไว้ เรียกกลับมาใช้เพื่อเล่นต่ออีกครั้ง , ปุ่มกลับสู่เมนูหลัก เพื่อกลับสู่หน้าเมนูหลัก

2.3 ขั้นตอนการทำงาน



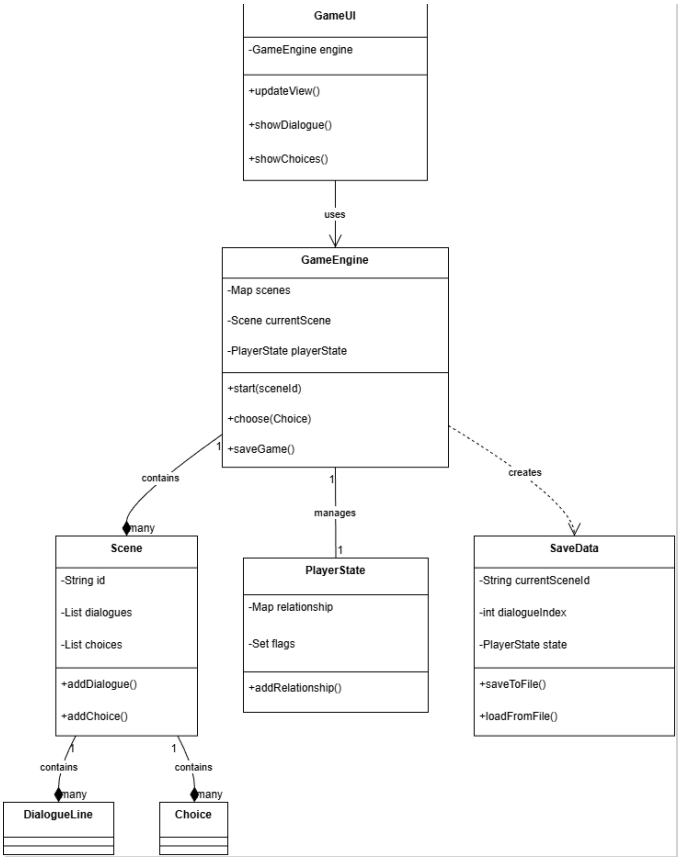
2.3 Flowchart

ลิ้งก์ Google Drive ข้อมูล รูปภาพ และ ไฟล์ Flowchart (หากข้อมูลรูปภาพไม่ชัดเจน)

https://drive.google.com/drive/folders/192ryssilAPweXcd0Y8YDBMt20cQf6ABv?usp=sharing

บทที่ 3 การออกแบบคลาส

3.1 Class Diagram



3.1 Class Diagram

ลิ้งก์ Google Drive ข้อมูล รูปภาพ และ ไฟล์ Flowchart (หากข้อมูลรูปภาพไม่ชัดเจน)

https://drive.google.com/drive/folders/192ryssiIAPweXcd0Y8YDBMt20cQf6ABv?usp=sharing

3.2 **อธิบายการทำงานของ** Class

1. **ส่วนแกนหลักและควบคุมเกม**

1.1 GameEngine: **เป็นคลาสหลัก**

เก็บรวบรวมข้อมูลฉาก (Scene) และตัวละคร (Character) ทั้งหมด

จัดการสถานะปัจจุบันของเกมว่าอยู่ที่ฉากไหน (currentScene)

ประมวลผลเมื่อผู้เล่นเลือกตัวเลือกผ่านเมธอด choose()

เชื่อมต่อระหว่างข้อมูลเกม (GameData) กับส่วนแสดงผล (UI)

1.2 PlayerState: **ทำหน้าที่เป็นหน่วยความจำของเกม เก็บข้อมูลสถานะของผู้เล่น**

ค่าความสัมพันธ์กับตัวละคร (relationship)

ตัวแปรสถานะเหตุการณ์ (flags)

ข้อมูลนี้จะถูกส่งไปบันทึกเมื่อทำการเซฟเกม

GameData : ทำหน้าที่เป็นแหล่งรวมเนื้อหา (Content Provider) โดยสร้าง Instance ของ Scene, Dialogue, และ Choice ทั้งหมด แล้วส่งเข้าไปเก็บใน GameEngine

2. **ส่วนแบบจำลองข้อมูล**

ทำหน้าที่เก็บโครงสร้างข้อมูลของเนื้อเรื่องและองค์ประกอบในเกม

2.1 Scene : **ตัวแทนของ "ฉาก" หนึ่งฉาก ประกอบด้วย**

รายการบทสนทนา (List<DialogueLine>)

รายการตัวเลือก (List<Choice>)

ภาพพื้นหลังและเพลงประกอบฉาก

2.2 **DialogueLine:** เก็บข้อมูลบทพูด 1 บรรทัด ประกอบด้วย ผู้พูด (Speaker), ข้อความ (Text), และอารมณ์/ท่าทาง (Expression)

2.3 **Choice:** เก็บข้อมูลตัวเลือกสำหรับผู้เล่น รวมถึงผลลัพธ์ที่จะเกิดขึ้น เช่น ID ของฉากถัดไป (nextSceneId) หรือค่าความสัมพันธ์ที่เปลี่ยนไป

2.4 **Character:** เก็บข้อมูลตัวละคร เช่น ชื่อ, ID, และ Path ของรูปภาพท่าทางต่างๆ (Expressions)

2.5 **SaveData:** คลาสสำหรับทำ Serialization เพื่อบันทึกข้อมูลสำคัญ (currentSceneId, dialogueIndex, PlayerState) ลงไฟล์

3. **ส่วนแบบจำลองข้อมูล**

ทำหน้าที่เก็บโครงสร้างข้อมูลของเนื้อเรื่องและองค์ประกอบในเกม

3.1 Scene: **ตัวแทนของ "ฉาก" หนึ่งฉาก ประกอบด้วย**:

รายการบทสนทนา (List<DialogueLine>)

รายการตัวเลือก (List<Choice>)

ภาพพื้นหลังและเพลงประกอบฉาก

DialogueLine: เก็บข้อมูลบทพูด 1 บรรทัด ประกอบด้วย ผู้พูด (Speaker), ข้อความ (Text), และอารมณ์/ท่าทาง (Expression)

Choice: เก็บข้อมูลตัวเลือกสำหรับผู้เล่น รวมถึงผลลัพธ์ที่จะเกิดขึ้น เช่น ID ของฉากถัดไป (nextSceneId) หรือค่าความสัมพันธ์ที่เปลี่ยนไป

Character: เก็บข้อมูลตัวละคร เช่น ชื่อ, ID, และ Path ของรูปภาพท่าทางต่างๆ (Expressions)

SaveData: คลาสสำหรับทำ Serialization เพื่อบันทึกข้อมูลสำคัญ (currentSceneId, dialogueIndex, PlayerState) ลงไฟล์

4. **ส่วนสนับสนุนและเครื่องมือ** (Utilities)

SoundPlayer: จัดการระบบเสียง ทั้งเพลงพื้นหลัง (BGM) และเสียงเอฟเฟกต์ (SFX) โดยรองรับการปรับระดับความดัง (Volume Control)

5. **หลักการ OOP ที่นำมาใช้**

5.1 Encapsulation (**การห่อหุ้มข้อมูล**)

มีการกำหนด Access Modifier เป็น private สำหรับตัวแปรภายในคลาส (เช่น relationship ใน PlayerState) และเข้าถึงผ่าน Method get/add เพื่อป้องกันการแก้ไขข้อมูลโดยตรงจากภายนอกที่ไม่ถูกต้อง

5.2 Single Responsibility Principle (**หลักการความรับผิดชอบเดียว**)

แยกการทำงานชัดเจน เช่น SoundPlayer มีหน้าที่เล่นเสียงเพียงอย่างเดียว ไม่ยุ่งเกี่ยวกับ Logic ของเกม หรือ SaveData มีหน้าที่จัดการเรื่องไฟล์เท่านั้น

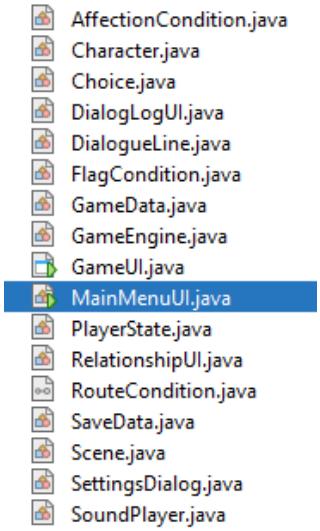
5.3 Abstraction (**ความเป็นนามธรรม**)

การใช้ Interface RouteCondition (ถ้ามีในโค้ดส่วนเงื่อนไข) ช่วยให้สามารถสร้างเงื่อนไขการเลือกตอบที่หลากหลายได้โดยไม่ต้องแก้โค้ดหลัก

5.4 Separation of Concerns (**การแยกส่วนความสนใจ**)

แยกระหว่าง Logic (GameEngine) และ UI (GameUI) ออกจากกัน ทำให้สามารถแก้ไขหน้าตาโปรแกรมได้โดยไม่กระทบกับกฎของเกม

3.3 Class และ Jframe ทั้งหมดในโปรแกรม



3.2 ภาพ Class and Jframe

3.3.1 GameEngine ควบคุมระบบเกม, เก็บฉากทั้งหมด, จำฉากปัจจุบัน

```java
package com.mycompany.visualnoveldemo;

import java.util.*;

/**
 * แกนหลักของเกม:
 *  - เก็บ scene ทั้งหมด
 *  - จำว่าอยู่ scene ไหน
 *  - เก็บสถานะผู้เล่น (ความสัมพันธ์)
 *  - จัดการเมื่อผู้เล่นเลือก choice
 */
public class GameEngine {
    private final Map<String, Character> characters = new HashMap<>();
    private final Map<String, Scene> scenes = new HashMap<>();
    private Scene currentScene;

    private PlayerState playerState = new PlayerState();

    /** เพิ่ม scene เข้า engine (ต้องเรียกจาก GameData) */
    public void addScene(Scene scene) {
        if (scene == null || scene.getId() == null) return;
        scenes.put(scene.getId(), scene);
    }

    /** เริ่มเกมที่ scene ตาม id */
    public void start(String sceneId) {
        currentScene = scenes.get(sceneId);
    }

    /** อ่าน scene ปัจจุบัน (เอาไปใช้ใน UI) */
    public Scene getCurrentScene() {
        return currentScene;
    }

    /** อ่านสถานะผู้เล่น (ใช้ดู relationship) */
    public PlayerState getPlayerState() {
        return playerState;
    }
```

3.3 ภาพ GameEngine Class image 1

```java
public void setPlayerState(PlayerState state) {
    if (state != null) {
        this.playerState = state;
    }
}

/** Helper สำหรับดู affection ของตัวละครจาก UI หรือ GameData */
public int getAffection(String characterId) {
    return playerState.getRelationship(characterId);
}

/** คืน list choice ของ scene ปัจจุบัน */
public List<Choice> getAvailableChoices() {
    if (currentScene == null) return Collections.emptyList();

    List<Choice> choices = currentScene.getChoices();
    if (choices == null) return Collections.emptyList();

    List<Choice> visible = new ArrayList<>();
    for (Choice c : choices) {
        if (c != null && c.isVisible(playerState)) {
            visible.add(c);
        }
    }
    return visible;
}

/** เวลาผู้เล่นเลือก choice */
public void choose(Choice choice) {
    if (choice == null) return;

    // 1) อัปเดตค่าความสัมพันธ์
    if (choice.getAffectionTargetId() != null && choice.getAffectionDelta() != 0) {
        playerState.addRelationship(
                choice.getAffectionTargetId(),
                choice.getAffectionDelta()
        );
    }
```

3.4 ภาพ GameEngine Class image 2

```java
    // 2) เปลี่ยนฉากไป nextSceneId
    Scene next = scenes.get(choice.getNextSceneId());
    if (next != null) {
        currentScene = next;
    }
}

public void addCharacter(Character c) {
    characters.put(c.getId(), c);
}

public java.util.List<Character> getAllCharacters() {
    return new java.util.ArrayList<>(characters.values());
}

// สร้างข้อมูลเซฟจากสถานะปัจจุบัน
public SaveData toSaveData(int dialogueIndex) {
    String sceneId = (currentScene != null) ? currentScene.getId() : null;
    return new SaveData(sceneId, dialogueIndex, playerState);
}


// โหลดสถานะจาก SaveData
public void loadFromSaveData(SaveData data) {
    if (data == null) return;

    this.playerState = data.getPlayerState();

    if (data.getCurrentSceneId() != null) {
        Scene s = scenes.get(data.getCurrentSceneId());
        if (s != null) {
            this.currentScene = s;
        }
    }
}
```

3.5 ภาพ GameEngine Class image 3

3.3.2 PlayerState เก็บค่าความสัมพันธ์ และสถานะเหตุการณ์

```java
package com.mycompany.visualnoveldemo;

import java.io.Serializable;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

/**
 * เก็บสถานะของผู้เล่น
 *  - relationship: ค่าความสัมพันธ์ของแต่ละตัวละคร
 *  - flags: ตัวแปรสถานะต่าง ๆ (ใช้กับ FlagCondition)
 *
 * รองรับทั้งเมธอดแบบใหม่ (getRelationship/addRelationship)
 * และเมธอดแบบเก่า (getAffection/addAffection/hasFlag/setFlag)
 */
public class PlayerState implements Serializable {

    private static final long serialVersionUID = 1L;

    // ความสัมพันธ์กับตัวละคร: key = characterId, value = score
    private final Map<String, Integer> relationship = new HashMap<>();

    // แฟลกสถานะต่าง ๆ
    private final Set<String> flags = new HashSet<>();

    // ---------- เมธอดหลักสำหรับ Relationship ----------

    /** อ่านค่าความสัมพันธ์ ถ้าไม่เคยมีให้คืน 0 */
    public int getRelationship(String characterId) {
        return relationship.getOrDefault(characterId, 0);
    }

    /** เพิ่ม/ลดค่าความสัมพันธ์ */
    public void addRelationship(String characterId, int delta) {
        int current = relationship.getOrDefault(characterId, 0);
        relationship.put(characterId, current + delta);
    }
```

3.6 ภาพ PlayerState Class image 1

```java
    public Map<String, Integer> getRelationshipMap() {
        return relationship;
    }

    // ---------- alias สำหรับโค้ดเก่า (ชื่อเดิม) ----------

    public int getAffection(String characterId) {
        return getRelationship(characterId);
    }

    public void addAffection(String characterId, int delta) {
        addRelationship(characterId, delta);
    }

    /** ให้ RelationshipUI ใช้ชื่อเก่าก็ได้ด้วย */
    public Map<String, Integer> getAffectionMap() {
        return getRelationshipMap();
    }

    // ---------- ระบบ Flag ----------

    public boolean hasFlag(String flagName) {
        return flags.contains(flagName);
    }

    public void setFlag(String flagName) {
        flags.add(flagName);
    }

    public void clearFlag(String flagName) {
        flags.remove(flagName);
    }

    public Set<String> getFlags() {
        return flags;
    }

    @Override
    public String toString() {
        return "PlayerState{" +
                "relationship=" + relationship +
                ", flags=" + flags +
                '}';
    }
}
```

3.7 ภาพ PlayerState Class image 2

3.3.3 SaveData เก็บข้อมูลสำหรับเซฟไฟล์ ฉากปัจจุบัน, บรรทัดปัจจุบัน, สถานะผู้เล่น

```java
package com.mycompany.visualnoveldemo;

import java.io.*;
import java.nio.file.Files;
import java.nio.file.Path;

/**
 * ข้อมูลที่ใช้เซฟเกม
 *  - currentSceneId: เราอยู่ฉากไหน
 *  - dialogueIndex: อยู่บรรทัดบทพูดที่เท่าไร
 *  - playerState: ค่าความสัมพันธ์ / flag ต่าง ๆ
 */
public class SaveData implements Serializable {

    private static final long serialVersionUID = 1L;

    private final String currentSceneId;
    private final int dialogueIndex;
    private final PlayerState playerState;

    public SaveData(String currentSceneId, int dialogueIndex, PlayerState playerState) {
        this.currentSceneId = currentSceneId;
        this.dialogueIndex = dialogueIndex;
        this.playerState = playerState;
    }

    public String getCurrentSceneId() {
        return currentSceneId;
    }

    public int getDialogueIndex() {
        return dialogueIndex;
    }

    public PlayerState getPlayerState() {
        return playerState;
    }
```

3.8 ภาพ Savedata Class image 1

```java
    // ---------- helper สำหรับอ่าน/เขียนไฟล์ ----------

    public static void saveToFile(String filename, SaveData data) throws IOException {
        try (ObjectOutputStream oos =
                new ObjectOutputStream(new FileOutputStream(filename))) {
            oos.writeObject(data);
        }
    }

    public static SaveData loadFromFile(String filename) throws IOException, ClassNotFoundException {
        if (!Files.exists(Path.of(filename))) {
            return null;
        }
        try (ObjectInputStream ois =
                new ObjectInputStream(new FileInputStream(filename))) {
            Object obj = ois.readObject();
            return (SaveData) obj;
        }
    }
}
```

3.9 ภาพ Savedata Class image 2

3.3.4 Scene เก็บข้อมูล 1 ฉาก พื้นหลัง, เพลง, รายการบทพูด, รายการตัวเลือก

```java
package com.mycompany.visualnoveldemo;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class Scene {

    private final String id;
    private String backgroundPath;
    private boolean ending = false;

    private final List<DialogueLine> dialogues = new ArrayList<>();
    private final List<Choice> choices = new ArrayList<>();

    private String bgmPath;

    public void setBgmPath(String path) {
        this.bgmPath = path;
    }

    public String getBgmPath() {
        return bgmPath;
    }

    public Scene(String id) {
        this.id = id;
    }

    public String getId() {
        return id;
    }

    public String getBackgroundPath() {
        return backgroundPath;
    }
```

3.10 ภาพ Scene Class image 1

```java
    public void setBackgroundPath(String backgroundPath) {
        this.backgroundPath = backgroundPath;
    }

    public void addDialogue(DialogueLine line) {
        dialogues.add(line);
    }

    public List<DialogueLine> getDialogues() {
        return Collections.unmodifiableList(dialogues);
    }

    // 💡 สำคัญ: ให้เก็บ choices ไว้ใน scene
    public void addChoice(Choice choice) {
        choices.add(choice);
    }

    public List<Choice> getChoices() {
        return Collections.unmodifiableList(choices);
    }

    public void setEnding(boolean ending) {
        this.ending = ending;
    }

    public boolean isEnding() {
        return ending;
    }
}
```

3.11 ภาพ Scene Class image 2

3.3.5 DialogueLine เก็บข้อมูล 1 ประโยคพูด ใครพูด, พูดว่าอะไร, ทำหน้าแบบไหน

```java
package com.mycompany.visualnoveldemo;

public class DialogueLine {

    private final Character speaker;
    private final String text;
    private final String expressionKey;   // ท่าทาง
    private final String backgroundPath;  // ✅ BG เฉพาะบรรทัด (optional)

    // ของเดิม (ไม่ระบุท่าทาง / BG)
    public DialogueLine(Character speaker, String text) {
        this(speaker, text, null, null);
    }

    // ระบุท่าทาง
    public DialogueLine(Character speaker, String text, String expressionKey) {
        this(speaker, text, expressionKey, null);
    }

    // ✅ ใหม่: ระบุทั้งท่าทาง + BG
    public DialogueLine(Character speaker, String text,
                        String expressionKey, String backgroundPath) {
        this.speaker = speaker;
        this.text = text;
        this.expressionKey = expressionKey;
        this.backgroundPath = backgroundPath;
    }

    public Character getSpeaker() {
        return speaker;
    }

    public String getText() {
        return text;
    }

    public String getExpressionKey() {
        return expressionKey;
    }

    public String getBackgroundPath() {
        return backgroundPath;
    }
```

3.12 ภาพ DialogueLine Class image 1

3.3.6 Character เก็บข้อมูลตัวละคร ชื่อ, ไฟล์รูปท่าทางต่างๆ

```java
package com.mycompany.visualnoveldemo;

import java.util.HashMap;
import java.util.Map;

public class Character {
    private final String id;
    private final String name;

    private final Map<String, String> expressions = new HashMap<>();
    private String defaultExpression = "default";

    public Character(String id, String name) {
        this.id = id;
        this.name = name;
    }

    public String getId() { return id; }
    public String getName() { return name; }

    public Character addExpression(String key, String path) {
        expressions.put(key, path);
        return this;
    }

    public void setDefaultExpression(String key) {
        this.defaultExpression = key;
    }

    public String getAvatarPath() {
        return getAvatarPath(defaultExpression);
    }

    public String getAvatarPath(String expressionKey) {
        if (expressionKey == null || expressionKey.isEmpty()) {
            expressionKey = defaultExpression;
        }
        String p = expressions.get(expressionKey);
        if (p == null) p = expressions.get(defaultExpression); // fallback
        return p;
    }
```

3.13 ภาพ Character Class image 1

3.3.7 Choice เก็บข้อมูลตัวเลือก ข้อความปุ่ม, เงื่อนไข, ผลลัพธ์ต่อค่าความสัมพันธ์

```java
public class Choice {

    private final String text;
    private final String nextSceneId;

    private final String affectionTargetId;
    private final int affectionDelta;

    // ✅ เงื่อนไขการแสดงผล
    private final String conditionTargetId; // เช่น "himiko"
    private final int minAffection;         // เช่น 10

    // ใช้กรณีทั่วไป (ไม่ยุ่งกับความสัมพันธ์)
    public Choice(String text, String nextSceneId) {
        this(text, nextSceneId, null, 0, null, Integer.MIN_VALUE);
    }

    // ใช้กรณีมีผลกับค่าความสัมพันธ์
    public Choice(String text, String nextSceneId,
                  String affectionTargetId, int affectionDelta) {
        this(text, nextSceneId, affectionTargetId, affectionDelta, null, Integer.MIN_VALUE);
    }

    // ✅ ใช้กรณีมีผลกับความสัมพันธ์ + มีเงื่อนไขให้โชว์
    public Choice(String text, String nextSceneId,
                  String affectionTargetId, int affectionDelta,
                  String conditionTargetId, int minAffection) {
        this.text = text;
        this.nextSceneId = nextSceneId;
        this.affectionTargetId = affectionTargetId;
        this.affectionDelta = affectionDelta;
        this.conditionTargetId = conditionTargetId;
        this.minAffection = minAffection;
    }

    public String getText() {
        return text;
    }
```

3.14 ภาพ Choice Class image 1

```java
    public String getNextSceneId() {
        return nextSceneId;
    }

    public String getAffectionTargetId() {
        return affectionTargetId;
    }

    public int getAffectionDelta() {
        return affectionDelta;
    }

    public String getConditionTargetId() {
        return conditionTargetId;
    }

    public int getMinAffection() {
        return minAffection;
    }

    // ✅ เช็คว่า choice นี้ "ควรแสดง" ไหม
    public boolean isVisible(PlayerState state) {
        if (conditionTargetId == null) return true; // ไม่มีเงื่อนไข = โชว์เสมอ
        if (state == null) return false;
        return state.getRelationship(conditionTargetId) >= minAffection;
    }

    @Override
    public String toString() {
        return "Choice{" +
                "text='" + text + '\' +
                ", nextSceneId='" + nextSceneId + '\' +
                ", affectionTargetId='" + affectionTargetId + '\' +
                ", affectionDelta=" + affectionDelta +
                ", conditionTargetId='" + conditionTargetId + '\' +
                ", minAffection=" + minAffection +
                '};
    }
```

3.15 ภาพ Choice Class image 2

3.3.7 SoundPlayer เก็บสถานะการเล่นเสียง และค่าระดับความดัง (Volume)

```java
package com.mycompany.visualnoveldemo;

import javax.sound.sampled.*;
import java.io.IOException;
import java.net.URL;

public class SoundPlayer {

    private static Clip bgmClip;
    private static FloatControl bgmGain;
    private static float volume = 0.8f; // 0.0 - 1.0

    public static void playBGM(String resourcePath, boolean loop) {
        stopBGM();
        try {
            URL url = SoundPlayer.class.getResource(resourcePath);
            if (url == null) {
                System.out.println("BGM not found: " + resourcePath);
                return;
            }

            AudioInputStream ais = AudioSystem.getAudioInputStream(url);
            bgmClip = AudioSystem.getClip();
            bgmClip.open(ais);

            if (bgmClip.isControlSupported(FloatControl.Type.MASTER_GAIN)) {
                bgmGain = (FloatControl) bgmClip.getControl(FloatControl.Type.MASTER_GAIN);
                applyVolumeToGain();
            } else {
                bgmGain = null;
            }

            if (loop) bgmClip.loop(Clip.LOOP_CONTINUOUSLY);
            bgmClip.start();

        } catch (UnsupportedAudioFileException | IOException | LineUnavailableException ex) {
            ex.printStackTrace();
        }
    }
```

3.16 ภาพ SoundPlayer Class image 1

```java
    public static void stopBGM() {
        if (bgmClip != null) {
            bgmClip.stop();
            bgmClip.close();
            bgmClip = null;
        }
        bgmGain = null;
    }

    public static void setVolume(float v) {
        volume = clamp01(v);
        applyVolumeToGain();
    }

    public static float getVolume() {
        return volume;
    }

    private static void applyVolumeToGain() {
        if (bgmGain == null) return;

        // map 0..1 -> dB (เงียบมาก ~ -60dB ถึง 0dB)
        float min = bgmGain.getMinimum(); // มักเป็น -80
        float max = bgmGain.getMaximum(); // มักเป็น 0
        float db;

        if (volume <= 0.0001f) db = min;
        else {
            // ใช้ log ทำให้ช่วงเสียง "ธรรมชาติ" กว่า linear
            db = (float) (20.0 * Math.log10(volume));
            if (db < min) db = min;
            if (db > max) db = max;
        }

        bgmGain.setValue(db);
    }
```

3.17 ภาพ SoundPlayer Class image 2

```java
private static float clamp01(float x) {
    return Math.max(0f, Math.min(1f, x));
}

// ---------- เล่นเสียงเอฟเฟกต์ (ไม่ loop) ----------
public static void playSFX(String resourcePath) {
    try {
        URL url = SoundPlayer.class.getResource(resourcePath);
        if (url == null) {
            System.out.println("SFX not found: " + resourcePath);
            return;
        }

        AudioInputStream ais = AudioSystem.getAudioInputStream(url);
        Clip clip = AudioSystem.getClip();
        clip.open(ais);

        // ใช้ volume เดียวกับ BGM
        if (clip.isControlSupported(FloatControl.Type.MASTER_GAIN)) {
            FloatControl gain =
                    (FloatControl) clip.getControl(FloatControl.Type.MASTER_GAIN);

            float min = gain.getMinimum();
            float max = gain.getMaximum();
            float db = (float) (20.0 * Math.log10(Math.max(0.0001f, volume)));
            gain.setValue(Math.max(min, Math.min(max, db)));
        }

        clip.start();

    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
```

3.18 ภาพ SoundPlayer Class image 3

บทที่ 4 การทดสอบการทำงาน และ ปัญหาที่พบ

โปรแกรมทำงานได้ ตามที่คาดหวังไว้ แต่ออาจจะติดปัญหานิดหน่อยเรื่องการออกแบบไม่เป็นไปตามที่หวัง

ปัญหาที่พบ

1. เนื้อหาของเกมทำได้ไม่ครบทั้งหมดจาก 4 Chapter เหลือแค่ 1 Chapter เพราะทำไม่ทัน

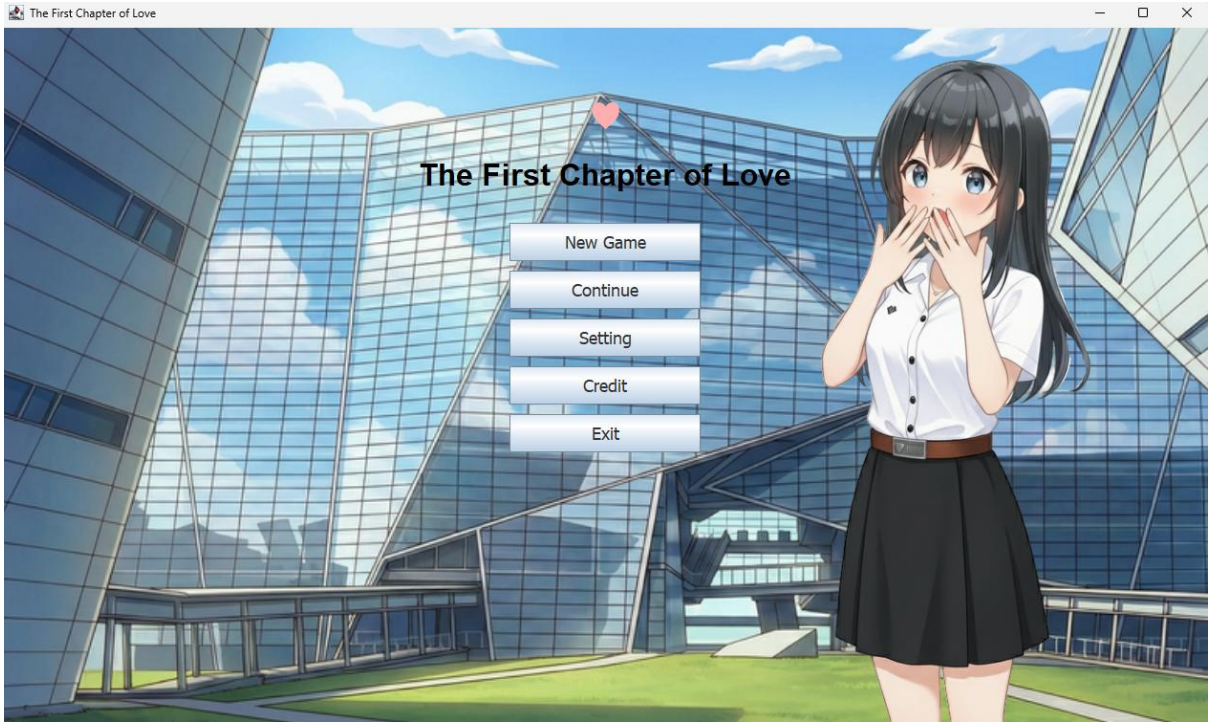
(แต่ 1 Chapter เนื้อหาก็เยอะเอาเรื่องเลยคับ อาจารย์ 555555)

2. โปรแกรมไม่อำนวยต่อการสร้างเกมทำให้เกิดความลำบากในการออกแบบ

3. โหมด Ai ไม่สิ้นสิดทำไม่ได้ เพราะ API Keys จำกัดการใช้ Token

บทที่ 5 คู่มือการใช้งานระบบ

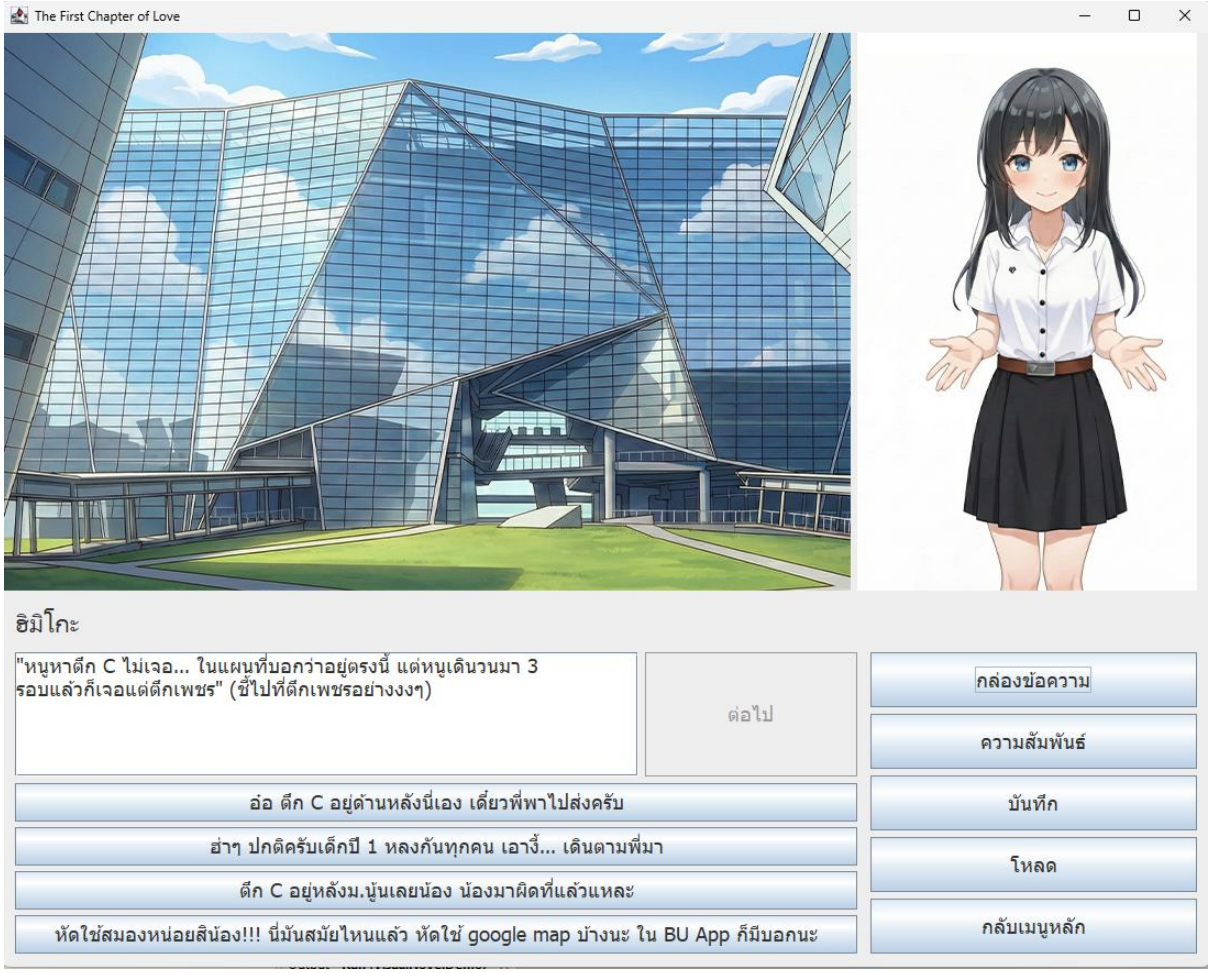
5.1 หน้าเมนูเกม (Menu game)



5.1 ภาพหน้าเมนูเกม

โดยหน้า Main Menu จะมีปุ่มดังนี้ ปุ่ม New Game เมื่อกด จะเข้าสู่หน้าเล่นเกม แต่จะเป็นการเริ่มเกม ใหม่ทันที , ปุ่ม Continue เมื่อกด จะเข้าสู่หน้าเล่นเกม แต่จะเป็นการเริ่มจากที่ Save ล่าสุด , ปุ่ม Setting เมื่อกด หน้าต่างเล็กๆ จะเด้งขึ้นมา เพื่อลดเสียงเกมตามที่ต้องการ , ปุ่ม Credit เมื่อกด หน้าต่างเล็กๆ จะเด้งขึ้นมา พร้อมชื่อผู้จัดทำเกม, ปุ่ม Exit เมื่อกด จะเป็นการปิดตัวเกมทันที

5.2 **หน้าหลักเกม** (Main Game) (Menu game **ปุ่ม** New game)



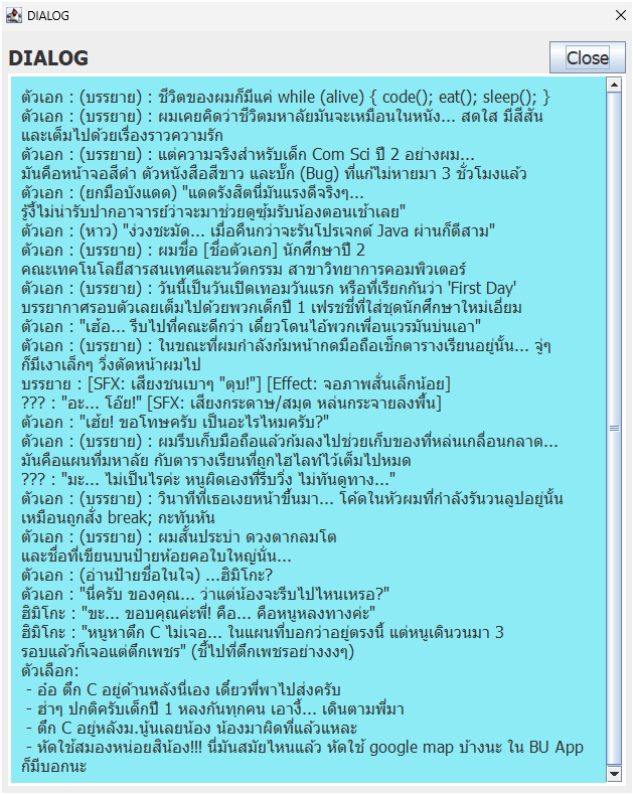
5.2 ภาพหน้าหลักเกม

โดยหน้าหลักเกม จะมีปุ่มดังนี้

1. ปุ่ม ต่อไป เมื่อกดปุ่มนี้ โปรแกรมจะขยับ index ของบทพูดไปข้างหน้า 1 หน้า

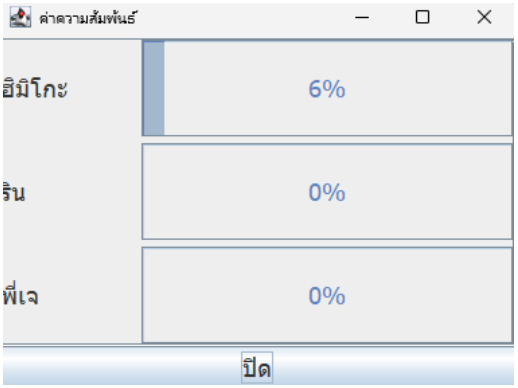
2. ปุ่มตัวเลือก 4 ตัวเลือก จะส่งข้อมูลตัวเลือกกลับไปที่ engine.choose(c) เพื่อคำนวณค่าความสัมพันธ์และเปลี่ยน Scene ไปยังฉากถัดไปตามการเลือกของผู้เล่น (ทีเด็ด คือ บางครั้งการตอบดีก็ไม่ใช่จะถูกทางเสมอไป)

3. กล่องข้อความ เปิดหน้าต่าง DialogLogUI เพื่อดูประวัติบทสนทนาย้อนหลังโดยในหน้านี้จะมี ปุ่ม Close เพื่อปิดหน้า Dialog Log



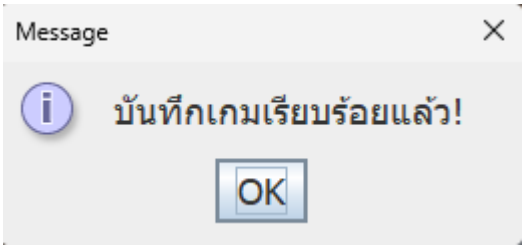
5.3 หน้า Dialog Log

4. ความสัมพันธ์ เมื่อกดจะเปิดหน้าต่าง Relationship เพื่อดูหลอดคะแนนความจีบติดของตัวละครแต่ละตัว และ จะมี ปุ่มปิด เพื่อปิดหน้า Relationship



5.4 ภาพหน้า Dialog Log

5. บันทึก เมื่อกดจะเรียกเมธอด saveGame() เพื่อเขียนสถานะปัจจุบัน (ฉากที่อยู่, บทพูดที่เท่าไหร่,ค่าความสัมพันธ์) ลงไฟล์ savegame.dat และ แสดง Message box แสดงขึ้นมาเมื่อบันทึกข้อมูลลงไฟล์สำเร็จแล้ว

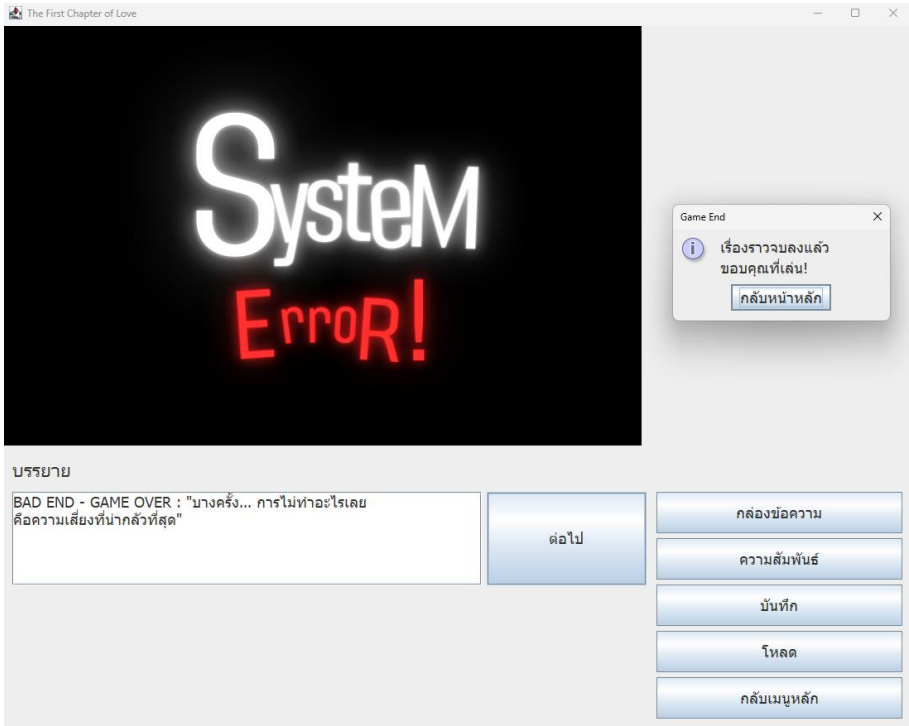


5.5 ภาพ Message box saveGame

6. โหลด เมื่อกด จะเรียกเมธอด loadGame() เพื่ออ่านไฟล์เซฟและพากลับมาที่จุดเดิมที่บันทึกไว้ และ แสดง Message box แสดงขึ้นมาเมื่อโหลดไฟล์เกมสำเร็จแล้ว

5.6 ภาพ loadGame

7. เมื่อผู้เล่นเลือกตัวเลือกที่เลวร้าย ผู้เล่นอาจจะเจอฉากจบ และ จะแสดง Message box แสดงขึ้นมาและมีปุ่มกลับสู่หน้าหลัก เพื่อ กลับสู่หน้าหลักเกม

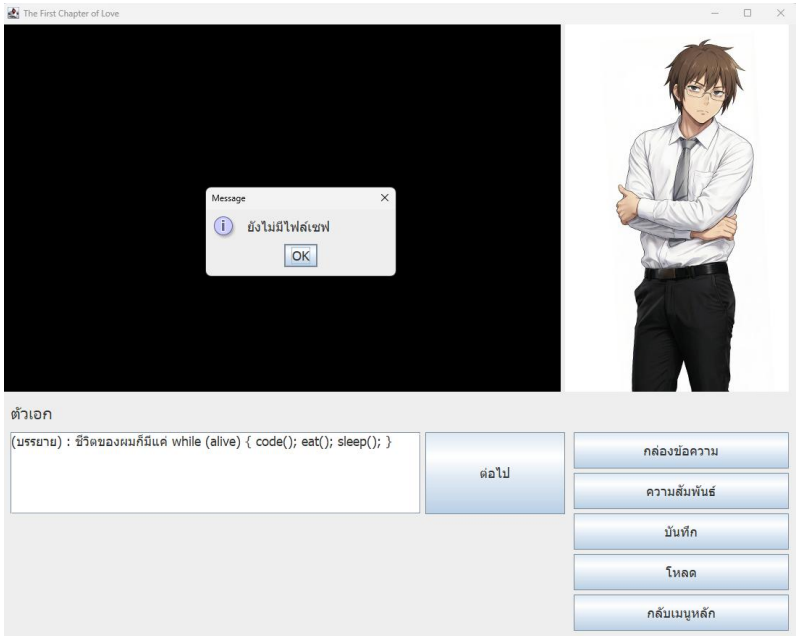


5.7 ภาพ End Game

5.3 **หน้าหลักเกม** (Menu game) (**ปุ่ม** Continue)

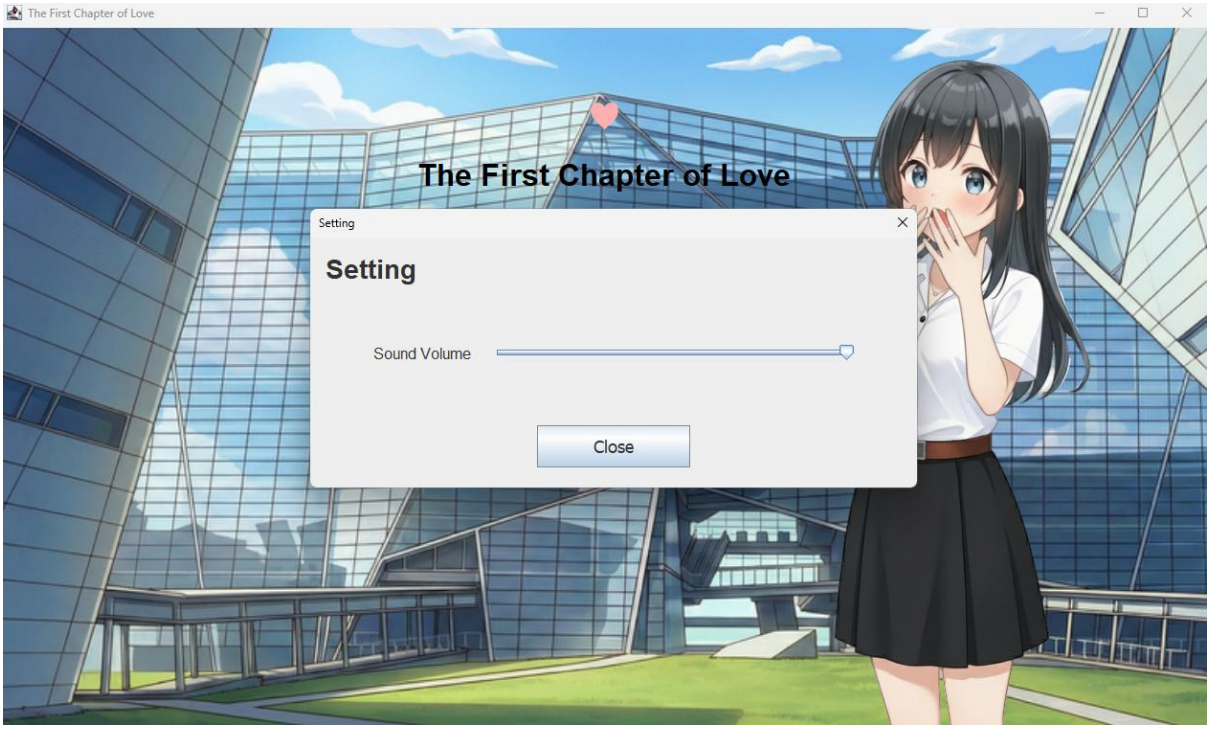
5.8 ภาพ End Game

เมื่อกด จะเรียกเมธอด loadGame() เพื่ออ่านไฟล์เซฟและพากลับมาที่จุดเดิมที่บันทึกไว้ และ แสดง Message box แสดงขึ้นมาเมื่อโหลดไฟล์เกมสำเร็จแล้ว แต่ถ้าไม่เคยบันทึกไว้ จะมีข้อความแจ้งเตือนเด้งขึ้นมาว่า "ยังไม่มีไฟล์เซฟ" และจะเข้าเกม มาหน้าแรก เพื่อให้ ผู้เล่น เล่นเกมได้ทันทีต่อไป



1.8 ภาพ Unknow Save

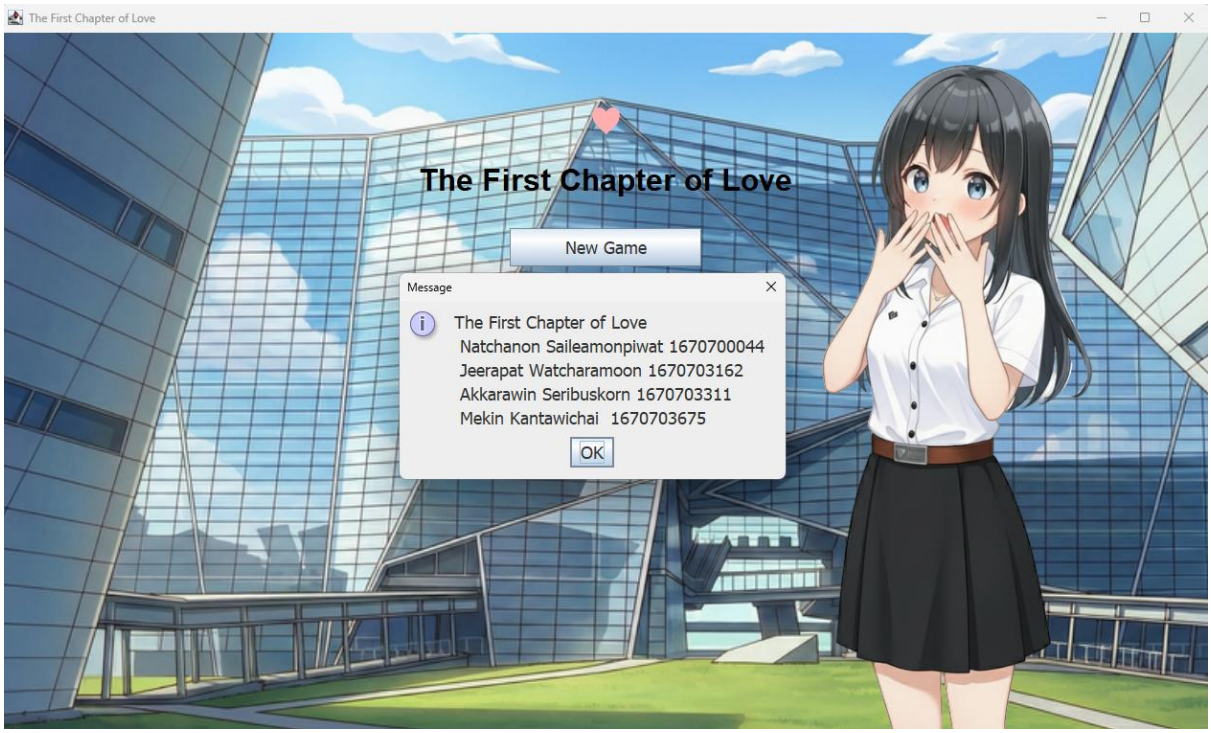
5.4 **หน้าหลักเกม** (Menu game) (**ปุ่ม** Setting)



5.9 ภาพ Setting

เมื่อกดปุ่ม Setting หน้าต่างเล็กๆ จะเด้งขึ้นมาเพื่อให้ **ปรับระดับความดังของเสียง** (Sound Volume) ผ่านตัวเลื่อน (Slider)

5.5 **หน้าหลักเกม** (Menu game) (**ปุ่ม** Credit)



5.10 ภาพ Credit

เมื่อกดปุ่ม Credit หน้าต่างเล็กๆ จะเด้งขึ้นมาบอกถึงผู้จัดทำ เพื่อให้เครดิตผู้พัฒนาเกม

5.6 **หน้าหลักเกม** (Menu game) (**ปุ่ม** Credit)

เมื่อกดปุ่ม Exit จะ ปิดโปรแกรมทันที