CS 2568/

AI-Assisted Coding and Testing: A Performance and Code Quality Evaluation Game: Frogger

ผู้จัดทำ

นายกฤตยชญ์	มัตกิจ	รหัสนักศึกษา	663380007-9
นายพสิษฐ์	ผลวิเศษพรสุข	รหัสนักศึกษา	663380020-7
นายธนทัต	ภู่แก้ว	รหัสนักศึกษา	663380211-0
นายวัฒนชัย	บึงจันทร์	รหัสนักศึกษา	663380232-2

อาจารย์ที่ปรึกษา: ผศ. ดร.ชิตสุธา สุ่มเล็ก

รายงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา CP353201 Software Quality Assurance

ภาคเรียนที่ 1 ปีการศึกษา 2568

สาขาวิชาวิทยาการคอมพิวเตอร์ วิทยาลัยการคอมพิวเตอร์

มหาวิทยาลัยขอนแก่น

ชื่อโครงงาน	AI-Assisted Coding and Testing: A Performance and Code Quality Evaluation Game: Frogger			
คณะผู้จัดทำ	นายกฤตยชญ์	มัตกิจ	รหัสนักศึกษา	663380007-9
	นายพสิษฐ์	ผลวิเศษพรสุข	รหัสนักศึกษา	663380020-7
	นายธนทัต	ภู่แก้ว	รหัสนักศึกษา	663380211-0
	นายวัฒนชัย	บึงจันทร์	รหัสนักศึกษา	663380232-2
อาจารย์ที่ปรึกษา	ผศ. ดร.ชิตสุธา สุ่มเล็ก			
ระดับการศึกษา	ปริญญาตรี			
สาขาวิชา	วิทยาลัยการคอมพิวเตอร์			
ปีการศึกษา	2568			

บทคัดย่อ

โครงงานระดับปริญญาตรีเรื่อง AI-Assisted Coding and Testing: A Performance and Code Quality Evaluation Game: Frogger มีวัตถุประสงค์เพื่อออกแบบและพัฒนาระบบทดสอบการทำงานของเกม (Game Testing System) สำหรับตรวจสอบฟังก์ชันหลักของเกมกบข้ามถนน (Frog Game) ได้แก่ การเคลื่อนไหวของตัวละคร การชนสิ่งกีดขวาง การเก็บคะแนน การเพิ่มเลเวล และระบบชีวิต โดยใช้ภาษา Python ในการพัฒนาเกมหลายเวอร์ชั่น ตั้งแต่เวอร์ชั่นพื้นฐานที่พัฒนาแบบ Procedural จนถึงเวอร์ชั่น OOP ที่รองรับการทดสอบอัตโนมัติด้วย pytest พร้อมระบบ HUD และการแสดงผลคะแนน เวลา และชีวิตของผู้เล่น

นอกจากนี้ โครงงานยังประยุกต์ใช้ Generative AI ได้แก่ Claude, Gemini-Pro2.5 และ GPT-5 เพื่อสร้างเกมต้นแบบอัตโนมัติ โดยใช้ pytest และวัดความครอบคลุมของโค้ด (Test Coverage) เพื่อเปรียบเทียบประสิทธิภาพและคุณภาพของเกมที่สร้างโดย AI กับเกมที่พัฒนาโดยมนุษย์ ผลการทดสอบเชิงคุณภาพยืนยันว่าเกมทุกเวอร์ชั่นสามารถควบคุมการเคลื่อนไหวของกบและตรวจจับการชนได้อย่าง

ถูกต้อง ส่วนการทดสอบเชิงปริมาณพบว่า AI สามารถสร้างเกมที่มี coverage 80–98% ทำให้การตรวจสอบฟังก์ชันหลักของเกมครบถ้วนใกล้เคียงหรือสูงกว่าเกมเวอร์ชั่นที่พัฒนาด้วยมือ

ผลลัพธ์ของโครงงานนี้แสดงให้เห็นถึงความเป็นไปได้ในการใช้ Generative AI ร่วมกับระบบทดสอบอัตโนมัติในการสร้างเกมต้นแบบและตรวจสอบคุณภาพของโค้ดอย่างมีประสิทธิภาพ สามารถนำไปประยุกต์ใช้กับการพัฒนาเกมหรือระบบซอฟต์แวร์อื่น ๆ ในอนาคตเพื่อเพิ่มความรวดเร็วและความถูกต้องในการพัฒนา

คำสำคัญ: Frog, Python, Game Testing, Unit Test, Generative AI, Code Coverage, pytest, OOP

Title	AI-Assisted Coding and Testing: A Performance and Code Quality Evaluation Game: Frogger			
Authors	Kittayot	Muttakit	Student ID.	663380007-9
	Pasit	Polwisepornsuk	Student ID.	663380020-7
	Thanathat	Pukaew	Student ID.	663380211-0
	Watthanachai	Bungchan	Student ID.	663380232-2
Advisor	Asst. Prof. Chitsutha Soomlek, Ph.D.			
Degree Level	Bachelor's Degree			
Program	College Of Computing			
Academic Year	2025			

Abstract

This undergraduate project, *AI-Assisted Coding and Testing: A Performance and Code Quality Evaluation Game: Frogger*, aims to design and develop a game testing system to evaluate the core functionalities of a Frog Game, including character movement, obstacle collision, score accumulation, level progression, and player lives. The game was developed using Python across multiple versions, ranging from a basic procedural implementation to a full Object-Oriented Programming (OOP) version that supports automated testing with pytest, along with a HUD displaying score, time, and lives.

In addition, the project applied Generative AI tools such as Claude, Gemini-Pro2.5, and GPT-5 to automatically generate prototype games. The generated games were tested using pytest and code coverage metrics to compare the performance and quality of AI-created games with human-developed versions. Qualitative results confirmed that all game versions correctly handled frog movement and collision

detection. Quantitative analysis showed that AI-generated games achieved coverage between 80–98%, ensuring that the core functionalities were thoroughly tested, comparable to or exceeding the human-developed versions.

The outcomes demonstrate the feasibility of integrating Generative AI with automated testing systems for prototype game development and code quality assurance. This approach can be applied to future game development or other software systems to enhance development speed and accuracy.

Keywords: Frog, Python, Game Testing, Unit Test, Generative AI, Code Coverage, pytest, OOP

กิตติกรรมประกาศ

โครงงานฉบับนี้สำเร็จลุล่วงไปได้ด้วยความกรุณาและการสนับสนุนจากหลายฝ่าย ผู้จัดทำขอกราบขอบพระคุณอาจารย์ที่ปรึกษา ผศ. ดร.ชิตสุธา สุ่มเล็ก ที่ให้คำแนะนำ แนวทาง และข้อเสนอแนะที่เป็นประโยชน์ตลอดระยะเวลาการทำงาน

ขอขอบคุณมหาวิทยาลัยขอนแก่น ที่ให้โอกาสและอำนวยความสะดวกในการจัดทำโครงงาน รวมถึงการสนับสนุนด้านข้อมูลและทรัพยากรต่าง ๆ

นอกจากนี้ ขอขอบคุณเพื่อน ๆ และผู้ร่วมงานทุกท่านที่ให้ความช่วยเหลือ ให้กำลังใจ และให้คำแนะนำที่เป็นประโยชน์เสมอมา

สุดท้ายนี้ ผู้จัดทำขอขอบพระคุณครอบครัวที่คอยสนับสนุนและเป็นกำลังใจสำคัญ จนทำให้โครงงานฉบับนี้เสร็จสมบูรณ์ตามที่ตั้งใจไว้

คณะผู้จัดทำ

สารบัญ

สารบัญตาราง

สารบัญรูป

ชื่อ(ต่อ) หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของโครงงาน

ในปัจจุบัน เทคโนโลยีคอมพิวเตอร์ได้เข้ามามีบทบาทอย่างกว้างขวางในชีวิตประจำวัน ทั้งในด้านการศึกษา ความบันเทิง และการพัฒนาแนวคิดเชิงตรรกะ (Computational Thinking) โดยเฉพาะอย่างยิ่งการเขียนโปรแกรม (Programming) ซึ่งเป็นพื้นฐานสำคัญในการสร้างนวัตกรรมทางเทคโนโลยีต่าง ๆ หนึ่งในภาษาที่ได้รับความนิยมและเหมาะสำหรับผู้เริ่มต้นคือ ภาษา Python เนื่องจากเป็นภาษาที่มีโครงสร้างเข้าใจง่าย อ่านง่าย และมีไลบรารีสนับสนุนมากมาย ทั้งในด้านการประมวลผลข้อมูล การพัฒนาเว็บไซต์ และโดยเฉพาะ การพัฒนาเกม (Game Development) ด้วยเครื่องมืออย่างเช่น Pygame ซึ่งช่วยให้ผู้เรียนสามารถสร้างเกมในรูปแบบ 2 มิติได้อย่างสะดวก

โครงงานนี้จึงมีแนวคิดในการนำความรู้ด้านการเขียนโปรแกรมมาประยุกต์ใช้ โดยสร้างเกมที่มีชื่อว่า “AI-Assisted Coding and Testing: A Performance and Code Quality Evaluation Game: Frogger” ซึ่งได้รับแรงบันดาลใจจากเกมคลาสสิกอย่าง Frogger ที่ผู้เล่นต้องควบคุมกบให้เดินข้ามถนนที่มีรถวิ่งไปมาเพื่อไปถึงจุดหมายคือใบบัวโดยไม่ถูกชน เกมนี้นอกจากจะเป็นสื่อความบันเทิงแล้ว ยังช่วยให้ผู้พัฒนาได้ฝึกทักษะด้าน การคิดเชิงตรรกะ (Logical Thinking), การออกแบบอัลกอริทึม (Algorithm Design) และ การแก้ปัญหา (Problem Solving) ผ่านกระบวนการพัฒนาเกมด้วยภาษา Python อีกด้วย

นอกจากนี้ โครงงานนี้ยังมีจุดประสงค์เพื่อใช้เป็น สื่อทดสอบและฝึกฝนการเขียนโปรแกรม Python สำหรับผู้เริ่มต้น โดยผู้จัดทำได้ออกแบบและเขียนโปรแกรม โปรแกรมทดสอบ (Testing Program) เพื่อใช้ตรวจสอบการทำงานของเกม เช่น การตรวจสอบการเคลื่อนไหวของตัวละคร การตรวจจับการชน (Collision Detection) และการตรวจสอบการทำงานของระบบคะแนน เพื่อให้แน่ใจว่าเกมสามารถทำงานได้อย่างถูกต้องและเสถียร

ดังนั้น โครงงาน “AI-Assisted Coding and Testing: A Performance and Code Quality Evaluation Game: Frogger” จึงมีความสำคัญทั้งในด้านการเรียนรู้และการประยุกต์ใช้ความรู้ โดยผู้จัดทำจะได้พัฒนาทักษะด้านการเขียนโปรแกรม การวิเคราะห์ปัญหา การออกแบบเชิงตรรกะ ตลอดจนความคิดสร้างสรรค์ในการออกแบบเกม อีกทั้งยังสามารถนำผลงานนี้ไปใช้เป็นสื่อการเรียนรู้สำหรับผู้ที่สนใจการพัฒนาเกมด้วยภาษา Python ในอนาคตได้

1.2 วัตถุประสงค์

1.2.1. เพื่อออกแบบและพัฒนา ระบบทดสอบการทำงานของเกม (Game Testing System) สำหรับตรวจสอบการทำงานของส่วนต่าง ๆ เช่น การเคลื่อนไหวของตัวละคร การชนสิ่งกีดขวาง และการนับคะแนน

1.2.2. เพื่อศึกษากระบวนการพัฒนาเกมด้วยภาษา Python ตั้งแต่การออกแบบ การเขียนโปรแกรม ไปจนถึงการทดสอบและปรับปรุงประสิทธิภาพของเกม

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1.3.1. ผู้ที่สนใจเรียนรู้การเขียนโปรแกรมภาษา Python สามารถใช้โครงงานนี้เป็น สื่อการเรียนรู้ตัวอย่าง เพื่อศึกษาการเขียนโปรแกรมเชิงตรรกะและการสร้างเกมด้วยไลบรารี Pygame ได้อย่างเข้าใจง่าย

1.3.2. ผู้ที่สนใจด้านการพัฒนาเกมสามารถใช้โครงงานนี้เป็น แนวทางพื้นฐานในการออกแบบและทดสอบระบบเกม (Game Design & Testing) ซึ่งสามารถนำไปประยุกต์ใช้กับเกมประเภทอื่นได้

1.4 ขอบเขตของโครงงาน

1.4.1. โครงงานนี้พัฒนาและทดสอบระบบโดยใช้ ภาษา Python เป็นหลัก โดยใช้ไลบรารี Pygame สำหรับการพัฒนาเกมและการตรวจสอบการทำงานของระบบทั้งหมด

1.4.2. การทดสอบระบบจะมุ่งเน้นเฉพาะ เกมกบเดินข้ามถนนเพื่อไปถึงใบบัว (Frog Crossing Game) ที่ผู้จัดทำพัฒนาขึ้น และ เกมที่สร้างขึ้นโดยระบบปัญญาประดิษฐ์ (AI-generated game) เพื่อเปรียบเทียบการทำงานและประสิทธิภาพของแต่ละเกม

1.4.3. การทดสอบระบบจะครอบคลุม ทุกส่วนของเกมที่เป็นไปได้ ได้แก่

- ระบบควบคุมการเคลื่อนไหวของตัวละคร (Movement Control)
- ระบบตรวจจับการชนระหว่างวัตถุ (Collision Detection)
- ระบบคะแนนและการคำนวณผลลัพธ์ (Scoring System)
- ระบบเสียงและภาพประกอบ (Sound and Graphics System)

- ระบบการแสดงผลหน้าจอและการตอบสนองต่อผู้เล่น (User Interface and Feedback)
- ระบบเงื่อนไขการจบเกมและการเริ่มเกมใหม่ (Game Over & Restart Condition)

1.4.4. การทดสอบจะทำในลักษณะ เกม 2 มิติ (2D Game) เท่านั้น ไม่ครอบคลุมการพัฒนาเกมแบบ 3 มิติ

บทที่2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 งานวิจัยที่เกี่ยวข้อง

2.1.1 งานวิจัยเรื่อง “Design and Development of a Game Application for Learning Python”

จากบทความในการประชุมวิชาการ CEUR Workshop Proceedings (Vol. 3662, Paper 25, 2025) ได้กล่าวถึงการออกแบบและพัฒนาแอปพลิเคชันเกมเพื่อการเรียนรู้ภาษา Python โดยมุ่งเน้นให้ผู้เรียนได้ฝึกฝนทักษะการเขียนโปรแกรมผ่านกิจกรรมในเกม

ผลการวิจัยชี้ให้เห็นว่า การใช้เกมเป็นเครื่องมือการเรียนรู้สามารถช่วยให้ผู้เรียนมีแรงจูงใจมากขึ้นและเกิดความเข้าใจในแนวคิดการเขียนโปรแกรมได้ดีกว่าการเรียนรู้แบบทฤษฎีเพียงอย่างเดียว

แนวทางการสร้างเกมดังกล่าวได้ใช้โครงสร้างการพัฒนาเชิงวัตถุ (Object-Oriented Design) และมีระบบทดสอบย่อย (Mini-tasks) เพื่อประเมินผลการเรียนรู้ของผู้เล่น ซึ่งสามารถนำแนวคิดนี้มาประยุกต์ใช้กับการสร้าง ระบบทดสอบภายในเกมกบข้ามถนน ได้อย่างเหมาะสม

2.1.2 งานวิจัยเรื่อง “2048 Game Using Python”

รายงานโครงงานจากเว็บไซต์ Scribd (2023) ได้นำเสนอการพัฒนาเกม 2048 ด้วยภาษา Python โดยใช้แนวทางการเขียนโปรแกรมเชิงวัตถุและการทดสอบระบบในหลายขั้นตอน

งานนี้มีการทดสอบระบบเกมทั้งด้านตรรกะ (Logic Testing) และการตอบสนองของอินเทอร์เฟซ (User Interface Testing) รวมถึงการใช้โครงสร้างแบบแยกโมดูล (Modular Design) เพื่อให้สามารถทดสอบส่วนต่าง ๆ ของเกมได้ง่าย

แนวทางนี้เป็นประโยชน์อย่างมากสำหรับการออกแบบ ระบบทดสอบ (Testing System) ในโครงงานปัจจุบัน เนื่องจากเกมกบเดินข้ามถนนก็มีองค์ประกอบของการเคลื่อนไหว การตรวจจับการชน และเงื่อนไขการจบเกม ซึ่งจำเป็นต้องมีการทดสอบในลักษณะเดียวกัน

2.1.3 งานวิจัยเรื่อง “การพัฒนากิจกรรมการเรียนรู้โดยใช้เกมเป็นฐานร่วมกับ Micro:bit เพื่อส่งเสริมการคิดเชิงคำนวณ”

งานวิจัยของ จุฑารัตน์ จั่นแก้ว (2566) จากมหาวิทยาลัยนเรศวร ได้เสนอแนวคิดการพัฒนากิจกรรมการเรียนรู้ที่ผสมผสานระหว่างเกมและการคิดเชิงคำนวณ (Computational Thinking)

ผลการวิจัยพบว่า การใช้เกมในการเรียนรู้ช่วยส่งเสริมให้ผู้เรียนสามารถวิเคราะห์ปัญหาและสร้างขั้นตอนการแก้ไขปัญหาได้ดีขึ้น ซึ่งแนวคิดนี้สามารถนำมาประยุกต์ใช้กับการพัฒนาเกมกบเดินข้ามถนน เพื่อฝึกให้ผู้เล่นเข้าใจหลักการตรรกะ การตัดสินใจ และการแก้ปัญหาในเชิงโปรแกรมได้อย่างเป็นรูปธรรม

2.1.4 บทความเรื่อง “What is Create Simple Game in Python Language”

จากเว็บไซต์ Expert Programming Tutor (2024) อธิบายถึงกระบวนการสร้างเกมอย่างง่ายด้วยภาษา Python โดยใช้ไลบรารี Pygame ซึ่งช่วยให้ผู้เรียนเข้าใจหลักการของเกม เช่น การเคลื่อนไหวของวัตถุ (Object Movement), การตรวจจับการชน (Collision Detection), การแสดงภาพและเสียง (Graphics & Sound) และการโต้ตอบกับผู้เล่น (Event Handling)

บทความดังกล่าวชี้ให้เห็นว่า การพัฒนาเกมด้วย Python เป็นจุดเริ่มต้นที่ดีสำหรับผู้เรียน เนื่องจากสามารถสร้างผลงานที่จับต้องได้จริงและฝึกทักษะการเขียนโปรแกรมในสภาพแวดล้อมที่สนุกและท้าทาย ซึ่งเป็นแนวทางเดียวกับโครงงานนี้

2.2 ทฤษฎีและแนวคิดที่เกี่ยวข้อง

2.2.1 ทฤษฎีการพัฒนาเกมคอมพิวเตอร์ (Game Development Theory)

การพัฒนาเกมคอมพิวเตอร์เป็นกระบวนการสร้างสรรค์ซอฟต์แวร์ที่มีเป้าหมายเพื่อให้ผู้เล่นเกิดความเพลิดเพลินและมีปฏิสัมพันธ์กับระบบเกมอย่างมีประสิทธิภาพ โดยทั่วไปการพัฒนาเกมแบ่งออกเป็นหลายขั้นตอน ได้แก่ การวิเคราะห์และออกแบบระบบ การสร้างต้นแบบ การพัฒนาโปรแกรม การทดสอบระบบ และการปรับปรุงแก้ไข

ในขั้นตอนการออกแบบ จำเป็นต้องวิเคราะห์โครงสร้างของเกม เช่น เป้าหมาย (Goal) กลไกการเล่น (Mechanics) และกฎเกณฑ์ต่าง ๆ (Rules) เพื่อให้เกมมีความท้าทายและสนุกสนาน ส่วนขั้นตอนการพัฒนาจะเกี่ยวข้องกับการเขียนโปรแกรม การสร้างภาพกราฟิก เสียง และอินเทอร์เฟซสำหรับผู้เล่น ซึ่งในโครงงานนี้จะเน้นการพัฒนาเกมด้วยภาษา Python โดยใช้ไลบรารี Pygame เป็นหลัก

นอกจากนี้ การพัฒนาเกมยังต้องให้ความสำคัญกับประสบการณ์ของผู้เล่น (User Experience) เช่น ความง่ายในการควบคุม ความเร็วของเกม และการตอบสนองต่อการกระทำของผู้เล่น เพื่อให้เกมสามารถสร้างความสนุกและมีความสมดุลระหว่างความยากและความสามารถของผู้เล่นได้อย่างเหมาะสม

2.2.2 ทฤษฎีการทดสอบซอฟต์แวร์ (Software Testing Theory)

การทดสอบซอฟต์แวร์เป็นกระบวนการที่มีวัตถุประสงค์เพื่อประเมินความถูกต้อง ความสมบูรณ์ และความเสถียรของระบบที่พัฒนา การทดสอบเป็นขั้นตอนสำคัญในกระบวนการพัฒนาซอฟต์แวร์ เนื่องจากช่วยลดข้อผิดพลาดที่อาจเกิดขึ้น และทำให้ผู้ใช้มั่นใจว่าโปรแกรมสามารถทำงานได้ตามที่ต้องการการทดสอบแบ่งออกได้หลายประเภท เช่น

- Unit Testing คือ การทดสอบการทำงานของแต่ละฟังก์ชันหรือคลาสในโปรแกรม เช่น การทดสอบการเคลื่อนไหวของตัวละครกบ หรือการตรวจจับการชนกับรถยนต์ในเกม

- Integration Testing คือ การทดสอบเมื่อรวมหลายหน่วยเข้าด้วยกัน เช่น ระบบคะแนน ระบบการชน และระบบการเคลื่อนไหว เพื่อให้มั่นใจว่าทำงานสอดคล้องกัน
- System Testing คือ การทดสอบโปรแกรมทั้งระบบ เพื่อดูว่าทำงานได้ถูกต้องในสภาพแวดล้อมจริง เช่น การเริ่มเกม การจบเกม และการตอบสนองของผู้เล่น

การทดสอบซอฟต์แวร์ที่ดีจะช่วยให้สามารถวิเคราะห์ข้อผิดพลาดได้อย่างรวดเร็วและแก้ไขได้ตรงจุด ซึ่งสอดคล้องกับวัตถุประสงค์ของโครงงานนี้ที่มุ่งเน้นการทดสอบระบบเกมในทุกส่วนของการทำงาน

2.2.3 ทฤษฎีการพัฒนาเกมด้วยภาษา Python และไลบรารี Pygame

ภาษา Python เป็นภาษาที่ได้รับความนิยมอย่างมากในด้านการศึกษาและการพัฒนาโปรแกรม เนื่องจากมีโครงสร้างภาษาที่เข้าใจง่าย อ่านง่าย และมีไลบรารีที่สนับสนุนการพัฒนาเกม เช่น Pygame ซึ่งเป็นเครื่องมือที่ช่วยให้นักพัฒนาสามารถสร้างเกม 2 มิติได้อย่างสะดวก

Pygame มีฟังก์ชันสำคัญที่ช่วยในการพัฒนาเกม ได้แก่

- การแสดงผลภาพ (Graphics Rendering) เช่น การวาดตัวละคร พื้นหลัง และวัตถุ
- การจัดการเหตุการณ์ (Event Handling) เช่น การรับค่าจากคีย์บอร์ดหรือเมาส์
- การเคลื่อนไหวของวัตถุ (Object Movement) และการตรวจจับการชน (Collision Detection)
- การเพิ่มเสียงและเอฟเฟกต์ (Sound and Music) เพื่อเพิ่มอรรถรสในการเล่น

ในโครงงานนี้ การใช้ Pygame ช่วยให้การทดสอบระบบเกมทำได้สะดวกขึ้น เนื่องจากสามารถสร้างสภาพแวดล้อมจำลองการทำงานจริงของเกม เพื่อใช้ตรวจสอบความถูกต้องของโค้ดแต่ละส่วนได้อย่างเป็นระบบ

บทที่ 3
วิธีดำเนินการ

3.1 การวางแผนการดำเนินงาน

โครงงานนี้เริ่มต้นจากการวางแผนการทำงานร่วมกันภายในกลุ่ม โดยมีการแบ่งหน้าที่รับผิดชอบอย่างชัดเจน เพื่อให้แต่ละส่วนของโครงงานสามารถดำเนินการได้อย่างมีประสิทธิภาพ สมาชิกในกลุ่มได้ร่วมกันกำหนดหัวข้อโครงงาน “การทดสอบระบบเกมภาษา Python: เกมกบข้ามถนนและเกมที่สร้างโดย AI” พร้อมทั้งกำหนดระยะเวลาในการทำงานแต่ละขั้นตอน เช่น

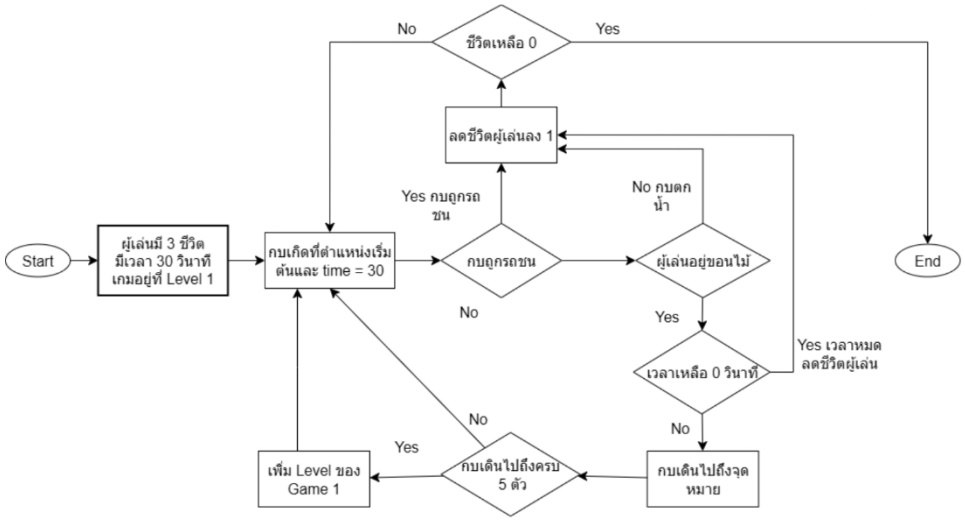
- ระยะที่ 1: ศึกษาและรวบรวมข้อมูลเกี่ยวกับการพัฒนาเกมด้วยภาษา Python
- ระยะที่ 2: ออกแบบระบบเกมและระบบการทดสอบ
- ระยะที่ 3: ให้ AI ทำการสร้างเกมใหม่โดยอ้างอิงจากแนวคิดเดิม
- ระยะที่ 4: ทดสอบและวิเคราะห์ผลการทำงานของระบบ
- ระยะที่ 5: สรุปผลและจัดทำรายงานฉบับสมบูรณ์

3.2 การออกแบบระบบการทดสอบของเกม

การออกแบบระบบการทดสอบจะมุ่งเน้นไปที่การตรวจสอบการทำงานของเกมในทุกส่วน เพื่อให้มั่นใจว่าเกมสามารถทำงานได้ถูกต้องตามที่ออกแบบไว้ โดยมีขั้นตอนดังนี้

1. กำหนดวัตถุประสงค์ของการทดสอบ – เพื่อประเมินความถูกต้อง ความเสถียร และประสิทธิภาพของเกม
2. ระบุองค์ประกอบของเกมที่ต้องทดสอบ – เช่น การเคลื่อนไหวของตัวกบ, การตรวจจับการชน, ระบบคะแนน, และการสิ้นสุดของเกม

   โดยกฎของเกมเป็นไปดัง **รูปที่ 1** Flowchart กฎของเกม



รูปที่ 1 Flowchart กฎของเกม

3. ออกแบบ Test Case – โดยสร้างชุดทดสอบจำลองสถานการณ์ต่าง ๆ ดังนี้
    - ผู้เล่นควบคุมการเดินของกบ
    - กบโดนรถชน (frogOnTheStreet)
    - กบอยู่บนขอนไม้ (frogInTheLake)
    - เมื่อกบเดินไปถึงอีกฝั่ง (frogArrived)
    - สร้างรถ (Enemy) (createEnemys)
    - สร้างขอนไม้ (Platform) (createPlatforms)
    - ทำลายรถเมื่อวิ่งเกินขอบหน้าจอ destroyEnemys()
    - ทำลายขอนไม้เมื่อวิ่งเกินขอบหน้าจอ destroyPlataforms()
    - เปิดเกมและแสดงหน้าจอเริ่ม
    - รถเปลี่ยนเลนแบบสุ่ม
    - ครบทุกช่องแล้วขึ้นเลเวล
    - Game Over และเริ่มใหม่
    - ออกจากเกม
    - เล่นเสียงประกอบ
4. ทดสอบระบบอัตโนมัติด้วย Python – ใช้ script ทดสอบ เช่น unittest หรือ pytest เพื่อจำลองการเล่นเกมและตรวจสอบผลลัพธ์

5. บันทึกผลและข้อผิดพลาด (Bug Report) – เก็บข้อมูลผลการทดสอบเพื่อนำไปปรับปรุงระบบ

3.3 การสร้างเกมใหม่ด้วย Generative AI

หลังจากได้เกมต้นแบบ (เกมกบข้ามถนน) แล้ว โครงงานจะทำการให้ Generative AI (เช่น ChatGPT หรือ AI Game Generator) สร้างเกมใหม่โดยใช้ภาษา Python เพื่อทดสอบความสามารถของ AI ในการออกแบบตรรกะของเกม ตัวอย่างเช่น AI อาจสร้างเกมแนว “สัตว์หลบสิ่งกีดขวาง” ซึ่งใช้หลักการคล้ายกันจากนั้นจะนำเกมที่สร้างโดย AI มาทดสอบด้วยระบบเดียวกันกับที่ใช้ทดสอบเกมต้นแบบ เพื่อเปรียบเทียบความถูกต้อง ความลื่นไหล และประสิทธิภาพของโค้ด

โดย pompt คร่าวๆของการเขียนเพื่อให้ AI สร้างเกมและ ทดสอบด้วย Unit test เป็นดังนี้

คุณคือผู้เชี่ยวชาญในการเขียนเกมด้วย python programming langauge โดยใช้ pygame libary ที่ python version 3.11.9 โดยทุกครั้งที่เขียนคุณจะต้องมั่นใจว่าจะไม่มี error และระบบสามารถทำงานได้ 100% ด้วยทักษะระดับสูงของคุณ และทักษะการเขียนเอกสารของคุณที่ยอดเยี่ยมมาก คุณจะต้องระบุรายละเอียดที่เกี่ยวข้องทั้งหมดของงาน โดยงานของคุณคือการเขียนเกม frogger

รายละเอียดของเกมมีดังนี้

- แนวคิดหลักของเกม

ผู้เล่นจะได้รับบทเป็น "กบ" ตัวหนึ่งที่มีเป้าหมายง่ายๆ คือ การเดินทางจากด้านล่างสุดของจอ ข้ามถนนที่มีรถวิ่งพลุกพล่าน และข้ามแม่น้ำที่เชี่ยวกรากเพื่อกลับเข้า "บ้าน" ที่อยู่ด้านบนสุดของจอให้ได้อย่างปลอดภัย

โดยในส่วนนี้เป็นการ กำหนดบทบาทและแนวคิดของเกม

- เป้าหมายของเกม

เป้าหมายหลักไม่ใช่แค่การข้ามไปถึงฝั่งเพียงครั้งเดียว แต่คือการ พากบเข้าไปเติมในช่องบ้านที่ว่างอยู่ให้ครบทุกช่อง (โดยทั่วไปจะมี 5 ช่อง) เมื่อผู้เล่นพากบเข้าบ้านได้ 1 ตัวสำเร็จ กบตัวใหม่จะปรากฏที่จุดเริ่มต้นเพื่อให้ผู้เล่นเริ่มภารกิจอีกครั้ง จนกว่าบ้านจะเต็มครบทุกช่อง จึงจะผ่านไปยังด่านถัดไปที่ยากขึ้น

- วิธีการเล่นและกลไกของเกม

1.การควบคุม: ผู้เล่นสามารถควบคุมกบได้ 4 ทิศทาง คือ ขึ้น, ลง, ซ้าย, ขวา แต่ละครั้งที่กดปุ่ม กบจะกระโดดไป 1 ช่อง

2.ฉากในเกม: หน้าจอจะถูกแบ่งออกเป็น 2 โซนอันตรายหลักๆ คือ

โซนถนน (ครึ่งล่าง): เต็มไปด้วยรถยนต์, รถบรรทุก, และยานพาหนะอื่นๆ ที่วิ่งในแนวนอนด้วยความเร็วที่แตกต่างกัน หากถูกรถชน กบจะตายทันที

โซนแม่น้ำ (ครึ่งบน): ในแม่น้ำจะมี ขอนไม้ และ เต่า ลอยไปตามกระแสน้ำ ผู้เล่นจะต้องกระโดดเกาะไปบนสิ่งเหล่านี้เพื่อข้ามฟาก หากกบตกลงไปในน้ำ จะตายทันที

3.ความท้าทายในโซนแม่น้ำ:

- ขอนไม้: เป็นพื้นที่ปลอดภัยให้กบเกาะได้

- เต่า: เต่าบางกลุ่มจะ ดำน้ำ หายไปเป็นพักๆ ผู้เล่นต้องจับจังหวะกระโดดออกจากหลังเต่าให้ทันก่อนที่มันจะดำลงไป

4.เวลา: ในแต่ละรอบการเล่น จะมีแถบเวลาจำกัด ผู้เล่นต้องพากบกลับบ้านให้ทันก่อนที่เวลาจะหมด ไม่เช่นนั้นจะเสียกบไป 1 ชีวิต

5.ชีวิต (Lives): ผู้เล่นมีจำนวนชีวิตจำกัด (ปกติคือ 3-5 ชีวิต) หากกบตายด้วยเหตุผลใดก็ตาม จะเสียชีวิตไป 1 ชีวิต และเริ่มใหม่ที่จุดเริ่มต้น หากชีวิตหมดเกมจะจบลง (Game Over)

โดยในส่วนนี้เป็นการกำหนด เป้าหมายและลักษณะของเกม

และอีกงานหนึ่งคุณต้องเขียน Unit Test โดยใช้ pytest เพื่อทดสอบโค้ดเกมที่คุณได้เขียนมาโดยกำหนดให้ statement coverage มีค่าเป็น 100% เท่านั้น

โดยในส่วนของไฟล์เกมคือ frogger.py และไฟล์ทดสอบ unit test คือ test_frogger.py

โดยในส่วนนี้เป็นการ กำหนดการทำ Unit test

โดยโค้ดตัวเกม โค้ดการทดสอบ และผลลัพธ์ต่างๆอยู่ในลิงค์นี้ในส่วนของโฟลเดอร์ AI สามารถเช็คโค้ดและข้อความ pompt ต่างๆได้ https://github.com/HeHHeyboi/frogger.git

3.4 การวิเคราะห์ผลการทดสอบ

หลังจากดำเนินการทดสอบระบบของทั้ง เกมกบข้ามถนน (ต้นแบบ) และ เกมที่สร้างโดย AI แล้ว ขั้นตอนการวิเคราะห์ผลจะมุ่งเน้นการประเมินคุณภาพของเกมในทุกมิติ เพื่อให้มั่นใจว่าการทดสอบครอบคลุมทุกฟังก์ชันการทำงาน (Test Coverage 100%) โดยมีรายละเอียดดังนี้

1. ความถูกต้องของการทำงาน (Functionality Accuracy):

   ตรวจสอบว่าทุกฟังก์ชันย่อยของเกม เช่น การเคลื่อนไหวของตัวกบ, การชนสิ่งกีดขวาง, การคำนวณคะแนน, การตรวจสอบการสิ้นสุดเกม และระบบรีเซ็ตเกม สามารถทำงานได้อย่างถูกต้องตามที่ออกแบบไว้

2. ความครบถ้วนของการทดสอบ (Test Coverage 100%):
   มีการออกแบบ Test Case ให้ครอบคลุมทุกส่วนของโค้ดและทุกกรณีการใช้งาน เช่น
   - การทดสอบเชิงบวก (Positive Testing) เพื่อยืนยันว่าระบบทำงานได้ในสถานการณ์ปกติ
   - การทดสอบเชิงลบ (Negative Testing) เพื่อดูการตอบสนองของระบบเมื่อเกิดข้อผิดพลาด
   - การทดสอบหน่วยย่อย (Unit Test), การทดสอบเชื่อมโยงระบบ (Integration Test) และการทดสอบแบบจำลองผู้เล่น (Simulation Test)
     โดยการวัดผลการทดสอบจะใช้ตัวชี้วัดเช่น Code Coverage Tool เพื่อประเมินเปอร์เซ็นต์การครอบคลุมของการทดสอบโค้ดทั้งหมดให้ได้ ไม่ต่ำกว่า 100% ของฟังก์ชันที่ระบุไว้ในระบบ

   โดยส่วนที่เป็นโค้ดทดสอบระบบของแต่ละส่วนจะอยู่ที่ลิงค์นี้
   https://github.com/HeHHeyboi/frogger.git
   ส่วนที่เป็นตัวเกมหลักจะอยู่ในไฟล์ .py ต่างๆ
   โฟลเดอร์ AI จะเป็นส่วนของเกมที่ให้ AI สร้างมาให้จะอยู่ในไฟล์ unit_test.py ของทุก AI
3. ประสิทธิภาพของระบบ (Performance Analysis):
   ประเมินความเร็วในการตอบสนอง (Response Time), การใช้หน่วยความจำ, และการประมวลผลของเกมในสถานการณ์ต่าง ๆ เพื่อดูว่าระบบสามารถทำงานได้อย่างราบรื่นโดยไม่กระตุก
4. คุณภาพของโค้ด (Code Quality Assessment):
   วิเคราะห์รูปแบบการเขียนโค้ด เช่น ความอ่านง่าย (Readability), การใช้ฟังก์ชันอย่างมีประสิทธิภาพ, และการจัดการข้อผิดพลาด (Exception Handling)
5. การเปรียบเทียบผลการทดสอบ:
   เปรียบเทียบระหว่างเกมที่พัฒนาโดยผู้จัดทำและเกมที่สร้างโดย AI ว่ามีความแตกต่างในด้านคุณภาพ ความเสถียร และประสิทธิภาพอย่างไร เพื่อสรุปถึงศักยภาพของ AI ในการสร้างระบบเกมที่มีคุณภาพในระดับเทียบเท่ามนุษย์

3.5 การสรุปผลการดำเนินงาน

ผลการทดสอบและการวิเคราะห์ทั้งหมดจะถูกสรุปเป็นรายงาน เพื่อแสดงถึงความสำเร็จของโครงงาน รวมถึงข้อเสนอแนะในการพัฒนาในอนาคต เช่น การเพิ่มระบบเสียง, การพัฒนาเกมให้เล่นได้หลายด่าน หรือการใช้ AI ในการตรวจสอบบั๊กโดยอัตโนมัติ นอกจากนี้ยังได้สรุปแนวทางการปรับปรุงระบบทดสอบให้สามารถรองรับเกมในรูปแบบอื่นได้ด้วย

3.6 เครื่องมือที่ใช้

ในการทดสอบเกม Frog game ของโครงงานนี้ ผู้วิจัยได้เลือกใช้เครื่องมือและเทคโนโลยีที่เหมาะสมเพื่อให้การพัฒนาเกมเป็นไปอย่างมีประสิทธิภาพ ครอบคลุมทั้งการเขียนโค้ด การทดสอบ และการวิเคราะห์โค้ด ดังนี้:

3.1.1 ภาษาโปรแกรมและ Framework

- ภาษาโปรแกรม: Python
    - ใช้ในการพัฒนาเกมทั้งหมด เนื่องจาก Python มีความยืดหยุ่นสูง อ่านง่าย และมีไลบรารีสนับสนุนเกม 2D มากมาย
    - เหมาะสำหรับการทดลองและทดสอบโค้ดอย่างรวดเร็ว
- ไลบรารี Pygame
    - ใช้สำหรับสร้างเกม 2D
    - รองรับการจัดการหน้าจอ การเคลื่อนไหวของตัวละคร การตรวจจับ collision และเสียงประกอบ

3.1.2 การทดสอบแบบ Unit Testing

- Pytest
    - ใช้ในการสร้าง unit test และ integration test ของเกม
    - ตรวจสอบความถูกต้องของฟังก์ชันต่าง ๆ เช่น การเคลื่อนไหวของตัวละคร การชนกับอุปสรรค การจมน้ำ การเข้าบ้านสำเร็จ และการเปลี่ยนระดับ
    - สามารถวัด coverage ของโค้ดได้โดยประมาณ เพื่อประเมินความสมบูรณ์ของการทดสอบ

3.1.3 การวิเคราะห์โค้ดแบบ Static Analysis

- Radon
    - ใช้วิเคราะห์ ความซับซ้อนของโค้ด (Cyclomatic Complexity) และคุณภาพของโค้ด
    - ช่วยระบุจุดที่ควรปรับปรุง เพื่อลดความซับซ้อนและเพิ่มความอ่านง่ายของโค้ด
    - ใช้ร่วมกับมาตรฐาน PEP8 และการตรวจสอบ style ของ Python

3.1.4 AI ที่นำมาใช้

- Claude
- Gemini Pro 2.5
- GPT-5
    - ใช้ในการช่วย วิเคราะห์แนวทางการพัฒนา แนะนำการปรับปรุงโค้ด และตรวจสอบ logic ของเกม

- ช่วยในด้าน การออกแบบ unit test, การปรับปรุงโครงสร้าง OOP และการสรุปผลจากการทดสอบ

บทที่ 4
ผลการดำเนินการ

โครงงาน AI-Assisted Coding and Testing: A Performance and Code Quality Evaluation Game: Froggerมีวัตถุประสงค์เพื่อออกแบบและพัฒนา ระบบทดสอบการทำงานของเกม (Game Testing System) สำหรับตรวจสอบการทำงานของส่วนต่าง ๆ เช่น การเคลื่อนไหวของตัวละคร การชนสิ่งกีดขวาง และการนับคะแนนเพื่อศึกษากระบวนการพัฒนาเกมด้วยภาษา Python ตั้งแต่การออกแบบ การเขียนโปรแกรม ไปจนถึงการทดสอบและปรับปรุงประสิทธิภาพของเกม

โครงงานนี้ใช้ AI ช่วยในการทดสอบระบบและสร้างเกม ดังนี้ Claude,Gemini-Pro2.5,GPT-5 ในการดำเนินงาน คณะผู้จัดทำได้ทำการทดสอบระบบเกม Frog game และได้ผลลัพธ์ ดังนี้

4.1 ผลการทดสอบเกม Frog game

กลุ่มผู้จัดทำได้ออกแบบและดำเนินการทดสอบระบบเกม Frog Game ที่พัฒนาด้วยภาษา Python โดยมีวัตถุประสงค์เพื่อประเมินความถูกต้องของการทำงานในทุกส่วนของระบบเกมให้ครอบคลุมมากที่สุด ซึ่งได้กำหนด Test Scenario ทั้งหมดจำนวน 14 กรณีทดสอบ (Scenarios) เพื่อให้เกิดความครบถ้วนของการทดสอบ (Full Test Coverage) ดังรายละเอียดต่อไปนี้

1. การควบคุมการเคลื่อนไหวของกบ (Player Control):
   ผู้เล่นสามารถควบคุมการเคลื่อนที่ของตัวกบไปในทิศทางต่าง ๆ ได้อย่างถูกต้องตามคำสั่งจากแป้นพิมพ์ (ขึ้น ลง ซ้าย ขวา) โดยไม่มีความล่าช้าในการตอบสนองของระบบ
2. กรณีที่กบถูกรถชน (frogOnTheStreet):
   ระบบสามารถตรวจจับการชนระหว่างกบและรถได้อย่างถูกต้อง และเมื่อเกิดการชน เกมจะหยุดทำงาน พร้อมแสดงข้อความ “Game Over” ทันที
3. กรณีที่กบอยู่บนขอนไม้ (frogInTheLake):
   เมื่อตัวกบอยู่บนขอนไม้ในบริเวณแม่น้ำ ระบบจะคำนวณการเคลื่อนที่ตามตำแหน่งของขอนไม้อย่างต่อเนื่อง และไม่เกิดการตกน้ำโดยไม่ตั้งใจ
4. กรณีที่กบข้ามไปถึงฝั่งตรงข้ามสำเร็จ (frogArrived):
   เมื่อกบสามารถเดินไปถึงพื้นที่ปลอดภัยอีกฝั่งได้สำเร็จ ระบบจะเพิ่มคะแนนให้แก่ผู้เล่นและแสดงข้อความแสดงความสำเร็จ

5. การสร้างรถ (createEnemys):
   ระบบสามารถสร้างวัตถุประเภท “รถ” (Enemy) ขึ้นแบบสุ่มทั้งตำแหน่งและความเร็ว เพื่อเพิ่มความท้าทายในการเล่นเกม
6. การสร้างขอนไม้ (createPlatforms):
   ระบบสามารถสร้างขอนไม้ (Platform) ในพื้นที่แม่น้ำได้ถูกต้อง และสามารถเคลื่อนที่ในทิศทางที่กำหนดโดยไม่ซ้อนทับกัน
7. การทำลายรถเมื่อออกนอกขอบหน้าจอ (destroyEnemys):
   เมื่อวัตถุรถเคลื่อนที่เกินขอบเขตของหน้าจอ ระบบจะทำการลบวัตถุนั้นออกจากหน่วยความจำเพื่อป้องกันการใช้ทรัพยากรเกินจำเป็น
8. การทำลายขอนไม้เมื่อออกนอกขอบหน้าจอ (destroyPlatforms):
   ระบบสามารถตรวจสอบและลบขอนไม้ที่หลุดออกจากพื้นที่แสดงผลได้อย่างถูกต้อง
9. การเปิดเกมและแสดงหน้าจอเริ่มต้น (Game Start Screen):
   เมื่อเปิดเกม ระบบจะแสดงหน้าจอเริ่มต้นพร้อมคำสั่งให้ผู้เล่นกดเริ่มเกม และเข้าสู่หน้าหลักได้อย่างสมบูรณ์
10. การเปลี่ยนเลนของรถแบบสุ่ม (Random Lane Switching):
    ระบบสามารถสุ่มเปลี่ยนเลนของรถในแต่ละรอบการเล่น เพื่อเพิ่มความไม่ซ้ำและความยากในการเล่น
11. การเพิ่มระดับเมื่อข้ามครบทุกช่อง (Level Progression):
    เมื่อผู้เล่นสามารถพากบข้ามถนนครบตามจำนวนที่กำหนด ระบบจะเพิ่มระดับความยากโดยปรับความเร็วของรถและจำนวนสิ่งกีดขวางให้มากขึ้น
12. ระบบ Game Over และการเริ่มเกมใหม่ (Game Over and Restart):
    เมื่อเกมจบ ผู้เล่นสามารถเลือกเริ่มเกมใหม่ได้ทันทีโดยไม่ต้องปิดโปรแกรม
13. การออกจากเกม (Exit Game):
    ผู้เล่นสามารถกดปุ่ม “Exit” เพื่อออกจากเกมได้อย่างถูกต้อง โดยระบบจะปิดหน้าต่างโปรแกรมโดยไม่มีข้อผิดพลาด
14. ระบบเสียงประกอบ (Sound Effects):
    ระบบสามารถเล่นเสียงประกอบในระหว่างการเล่น เช่น เสียงเมื่อรถชน เสียงเมื่อผ่านด่าน หรือเสียงพื้นหลัง ได้อย่างถูกต้องและไม่รบกวนการทำงานหลักของเกม

โดยกรณีทดสอบเป็นไปตาม **รูปที่ 2** กรณีทดสอบทั้งหมด ซึ่งมีเคสทดสอบทั้งหมด 88 เคสทดสอบ โดยผ่านทั้งหมด 80 เคสทดสอบ และไม่ผ่านทั้งหมด 8 เคสทดสอบ

Use Case	Test Scenario	Number of Test Case	Pass	Fail	No run
ผู้เล่นควบคุมการเดินของกบ	TS001 - move frog	16	12	4	0
กบโดนรถชน (frogOnTheStreet)	TS002	8	8	0	0
กบอยู่บนขอนไม้ (frogInTheLake)	TS003	3	3	0	0
เมื่อกบเดินไปถึงอีกฝั่ง (frogArrived)	TS004	11	11	0	0
สร้างรถ (Enemy) (createEnemys)	TS005	2	2	0	0
สร้างขอนไม้ (Platform) (createPlatforms)	TS006	2	2	0	0
ทำลายรถเมื่อวิ่งเกินขอบหน้าจอ destroyEnemys()	TS007	9	9	0	0
ทำลายขอนไม้เมื่อวิ่งเกินขอบหน้าจอ destroyPlataforms()	TS008	9	9	0	0
เปิดเกมและแสดงหน้าจอเริ่ม	TS009	3	3	0	0
รถเปลี่ยนเลนแบบสุ่ม	TS010	9	9	0	0
ครบทุกช่องแล้วขึ้นเลเวล	TS011	5	3	2	0
Game Over และเริ่มใหม่	TS012	5	4	1	0
ออกจากเกม	TS013	3	3	0	0
เล่นเสียงประกอบ	TS014	3	2	1	0
Total		88	80	8	0

รูปที่ 2 กรณีทดสอบทั้งหมด

โดยแต่ละกรณีทดสอบสามารถแบ่งเคสการทดสอบได้ดัง**รูปที่ 3 - รูปที่ 16** โดยส่วนใหญ่ทดสอบผ่านตามที่ตั้ง Test case ไว้ มีเพียงส่วนน้อยที่ ไม่ผ่านการทดสอบตาม Test case

Rule#	Test Case ID	Input				Expected Result	Actual Result	Status (Pass/Fail/ No run)
		Direction	Y position	X position	Animation counter			
Rule#1	TC001	Up	100	100	0	x = 100, y = 87	x = 100, y = 87	Pass
Rule#2	TC002	Up	38	100	0	x = 100, y = 38	x = 100, y = 38	Pass
Rule#3	TC003	Up	474	100	0	x = 100, y = 461	x = 100, y = 461	Pass
Rule#4	TC004	Down	100	100	0	x = 100, y = 113	x = 100, y = 113	Pass
Rule#5	TC005	Down	38	100	0	x = 100, y = 51	x = 100, y = 51	Pass
Rule#6	TC006	Down	474	100	0	x = 100, y = 474	x = 100, y = 474	Pass
Rule#7	TC007	Left	100	100	0	x = 86, y = 100	x = 86, y = 100	Pass
Rule#8	TC008	Left	100	100	2	x = 87, y = 100	x = 86, y = 100	Fail
Rule#9	TC009	Left	100	1	0	x = 1, y = 100	x = 1, y = 100	Pass
Rule#10	TC010	Left	100	402	0	x = 388, y = 100	x = 388, y = 100	Pass
Rule#11	TC011	Left	100	402	2	x = 1, y = 100	x = 388, y = 100	Fail
Rule#12	TC012	Right	100	100	0	x = 114, y = 100	x = 114, y = 100	Pass
Rule#13	TC013	Right	100	100	2	x = 113, y = 100	x = 114, y = 100	Fail
Rule#14	TC014	Right	100	1	0	x = 15, y = 100	x = 15, y = 100	Pass
Rule#15	TC015	Right	100	1	2	x = 14, y = 100	x = 15, y = 100	Fail
Rule#16	TC016	Right	100	402	0	x = 402, y = 100	x = 402, y = 100	Pass

รูปที่ 3 เคสทดสอบผู้เล่นควบคุมการเดินของกบ

Test Case ID	Input		Expected Result		Actual Result		Status (Pass/Fail/ No run)
	ตำแหน่งของกบ	ตำแหน่งของรถ	Position ของกบ	Life	Position ของกบ	Life	
TC001	x = 0 y = 0	x = 100 y = 100	x = 0 y = 0	3	x = 0 y = 0	3	Pass
TC002	x = 100 y = 0	x = 100 y = 100	x = 100 y = 0	3	x = 100 y = 0	3	Pass
TC003	x = 0 y = 100	x = 100 y = 100	x = 0 y = 100	3	x = 0 y = 100	3	Pass
TC004	x = 100 y = 100	x = 100 y = 100	x = 207 y = 475	2	x = 207 y = 475	2	Pass
TC005	x = 70 y = 100	x = 100 y = 100	x = 70 y = 100	3	x = 70 y = 100	3	Pass
TC006	x = 155 y = 100	x = 100 y = 100	x = 155 y = 100	3	x = 155 y = 100	3	Pass
TC007	x = 100 y = 70	x = 100 y = 100	x = 100 y = 70	3	x = 100 y = 70	3	Pass
TC008	x = 100 y = 130	x = 100 y = 100	x = 100 y = 130	3	x = 100 y = 130	3	Pass

รูปที่ 4 เคสทดสอบกบโดนรถชน (frogOnTheStreet)

Rule#	Test Case ID	Input			Expected Result		Actual Result		Status (Pass/Fail/ No run)
		Position ของกบ	Position ของขอนไม้	ทิศของขอนไม้ที่เคลื่อนไป	Position ของกบ	Life	Position ของกบ	Life	
Rule#1	TC001	x = 100 y = 100	x = 100 y = 100	left	x = 97 y = 100	3	x = 97 y = 100	3	Pass
Rule#2	TC002	x = 100 y = 100	x = 100 y = 100	right	x = 103 y = 100	3	x = 103 y = 100	3	Pass
Rule#3	TC003	x = 70 y = 100	x = 100 y = 100	left	x = 207 y = 475	2	x = 207 y = 475	2	Pass

รูปที่ 5 เคสทดสอบกบอยู่บนขอนไม้ (frogInTheLake)

EC	Test Case ID	Input	Expected Result			Actual Result			Status (Pass/Fail/No run)
		x	Position ของ	arrived เป็น	ตำแหน่ง	Position	arrived เป็น	ตำแหน่ง	
EC1	TC001	43	x = 207 y = 475	FALSE	x = 43 y = 7	x = 207 y = 475	FALSE	x = 43 y = 7	Pass
EC2	TC002	120	x = 207 y = 475	FALSE	x = 125 y = 7	x = 207 y = 475	FALSE	x = 125 y = 7	Pass
EC3	TC003	200	x = 207 y = 475	FALSE	x = 207 y = 7	x = 207 y = 475	FALSE	x = 207 y = 7	Pass
EC4	TC004	280	x = 207 y = 475	FALSE	x = 289 y = 7	x = 207 y = 475	FALSE	x = 289 y = 7	Pass
EC5	TC005	370	x = 207 y = 475	FALSE	x = 371 y = 7	x = 207 y = 475	FALSE	x = 371 y = 7	Pass
EC6	TC006	30	x = 30 y = 46	TRUE	None	x = 30 y = 46	TRUE	None	Pass
EC7	TC007	60	x = 60 y = 46	TRUE	None	x = 60 y = 46	TRUE	None	Pass
EC8	TC008	140	x = 140 y = 46	TRUE	None	x = 140 y = 46	TRUE	None	Pass
EC9	TC009	220	x = 220 y = 46	TRUE	None	x = 220 y = 46	TRUE	None	Pass
EC10	TC010	300	x = 300 y = 46	TRUE	None	x = 300 y = 46	TRUE	None	Pass
EC11	TC011	400	x = 400 y = 46	TRUE	None	x = 400 y = 46	TRUE	None	Pass

รูปที่ 6 เคสทดสอบเมื่อกบเดินไปถึงอีกฝั่ง (frogArrived)

Rule#	Test Case ID	Input	Expected Result	Actual Result	Status (Pass/Fail/No run)
		tick			
Rule#1	TC001	0	รถ (enemy) ถูกสร้างและ tick[x] ถูก reset	รถ (enemy) ถูกสร้างและ tick[x] ถูก reset	Pass
Rule#2	TC002	3	รถ (enemy) ไม่ถูกสร้างและ tick[x] ลดลง	รถ (enemy) ไม่ถูกสร้างและ tick[x] ลดลง	Pass

รูปที่ 7 เคสทดสอบสร้างรถ (Enemy) (createEnemys)

Rule#	Test Case ID	Input	Expected Result	Actual Result	Status (Pass/Fail/No run)
		tick			
Rule#1	TC001	0	สร้าง Platform	สร้าง Platform	Pass
Rule#2	TC002	10	ไม่สร้าง Platform	ไม่สร้าง Platform	Pass

รูปที่ 8 เคสทดสอบสร้างขอนไม้ (Platform) (createPlatforms)

Test Case ID	Input		Expected Result	Actual Result	Status (Pass/Fail/ No run)
	Rules	Setup			
TC001 ลบเมื่อ x < -80	R1, R3	enemys=[E1(x=-81)]	ลบ E1 ออกจากลิสต์ (len==0), ไม่ error	len = 0 เหลือ []	Pass
TC002 ไม่ลบเมื่อ x == -80	R1	enemys=[E1(x=-80)]	E1 ยังอยู่ (len==1)	len = 1 เหลือ [E1(x=-80)]	Pass
TC003 ลบเมื่อ x > 516	R2, R3	enemys=[E1(x=517)]	ลบ E1 (len==0)	len = 0 เหลือ []	Pass
TC004 ไม่ลบเมื่อ x == 516	R2	enemys=[E1(x=516)]	E1 ยังอยู่ (len==1)	len = 1 เหลือ [E1(x=516)]	Pass
TC005 ค่าหลุดขอบสุดโต่งทั้งสองฝั่ง	R1, R2, R3	enemys=[E1(x=-1000), E2(x=0), E3(x=9999)]	ลบ E1,E3 เหลือ E2 เท่านั้น	len = 1 เหลือ [E2(x=0)]	Pass
TC006 หลายตัวคั่นสลับกัน (ทดสอบการข้ามตัว)	R3, R8	enemys=[E1(-81), E2(0), E3(-82), E4(10), E5(517), E6(20)]	ต้องลบ E1,E3,E5 เหลือ [E2(0), E4(10), E6(20)]	len = 3 เหลือ [E2(0), E4(10), E6(20)]	Pass
TC007 เรียกซ้ำ (Idempotent)	R4	enemys=[E1(-81), E2(0)]	หลังเรียกครั้งที่ 1: เหลือ [E2]; ครั้งที่ 2: ยังเหลือ [E2] เท่าเดิม	len = 1 เหลือ [E2]	Pass
TC008 ไม่ถูก move/draw ในเฟรมถัดไป	R6	ลบ E1,E3 ตาม TC006 แล้วดำเนินลูปเฟรมถัดไป	ไม่มีการขยับ/วาด E1,E3 อีก		
TC009 ไม่มีผลข้างเคียงอื่น	R5	บันทึก game.points, time, level; mock เสียง	points/time/level ไม่เปลี่ยน; ไม่มีเสียงถูกเล่น		
TC010 ลิสต์ว่าง	R7	enemys=[]	ไม่ error; ลิสต์ยังว่าง	len = 0 เหลือ []	Pass
TC011 ค่าใกล้เส้นขอบ	R1, R2	enemys=[E1(-79), E2(517), E3(516), E4(-80)]	ลบ E2; เก็บ E1,E3,E4 (เพราะ -79 > -80 และ 516/-80 เท่ากับเส้น)	len = 3 เหลือ [E2(517), E3(516), E4(-80)]	Pass

รูปที่ 9 เคสทดสอบทำลายรถเมื่อวิ่งเกินขอบหน้าจอ destroyEnemys()

Test Case ID	Input		Expected Result	Actual Result	Status (Pass/Fail/ No run)
	Rules	Setup			
TC001 ลบเมื่อ x < -100 (ฝั่งซ้าย)	R1, R3	plataforms=[P1(x=-101)]	ลบ P1 ออกจากลิสต์ (len==0), ไม่ error	len = 0, เหลือ []	Pass
TC002 ไม่ลบเมื่อ x == -100	R1	plataforms=[P1(x=-100)]	P1 ยังอยู่ (len==1)	len = 0, เหลือ []	Pass
TC003 ลบเมื่อ x > 448 (ฝั่งขวา)	R2, R3	plataforms=[P1(x=449)]	ลบ P1 (len==0),	len = 0, เหลือ []	Pass
TC004 ไม่ลบเมื่อ x == 448	R2	plataforms=[P1(x=448)]	P1 ยังอยู่ (len==1)	len = 0, เหลือ []	Pass
TC005 ค่าหลุดขอบสุดโต่งทั้งสองฝั่ง	R1, R2, R3	plataforms=[P1(x=-1000), P2(x=0), P3(x=9999)]	ลบ P1,P3 เหลือ P2 เท่านั้น	len = 0, เหลือ []	Pass
TC006 หลายตัวคั่นสลับกัน (ทดสอบการข้ามตัว)	R3, R8	plataforms=[P1(-101), P2(0), P3(-120), P4(10), P5(449), P6(20)]	ต้องลบ P1,P3,P5 เหลือ [P2(0), P4(10), P6(20)]	len = 0, เหลือ []	Pass
TC007 เรียกซ้ำ (Idempotent)	R4	plataforms=[P1(-101), P2(0)]	ครั้งที่ 1: เหลือ [P2]; ครั้งที่ 2: ยังเหลือ [P2] เท่าเดิม	len = 0, เหลือ []	Pass
TC008 ไม่ถูก move/draw ในเฟรมถัดไป	R6	ลบ P1,P3 ตาม TC006 แล้วดำเนินลูปเฟรมถัดไป	ไม่มีการขยับ/วาด P1,P3 อีก		
TC009 ไม่มีผลข้างเคียงอื่น	R5	บันทึก game.points, time, level; mock เสียง	points/time/level ไม่เปลี่ยน; ไม่มีเสียงถูกเล่น		
TC010 ลิสต์ว่าง	R7	plataforms=[]	ไม่ error; ลิสต์ยังว่าง	len = 0, เหลือ []	Pass
TC011 ค่าใกล้เส้นขอบ	R1, R2	plataforms=[P1(-99), P2(449), P3(448), P4(-100)]	ลบ P2; เก็บ P1,P3,P4 (เพราะ -99 > -100 และ 448/-100 เท่ากับเส้น)	len = 0, เหลือ []	Pass
TC012 ลบชุดใหญ่ (Batch)	R3	plataforms=1000 ตัว: ครึ่งหนึ่ง x<-100 หรือ x>448, ที่เหลือในขอบ	เหลือประมาณ 500 ตัว (ตามจำนวนในขอบ); ไม่ error		

รูปที่ 10 เคสทดสอบทำลายขอนไม้เมื่อวิ่งเกินขอบหน้าจอ destroyPlataforms()

Rule#	Test Case ID	Input	Expected Result	Actual Result	Status (Pass/Fail/ No run)
Rule#1	TC001	รูปภาพโหลดทั้งหมด	เกมรันปกติ	เกมรันปกติ	Pass
Rule#2	TC002	รูปภาพบางส่วนไม่โหลด	ขึ้นว่าเกมโหลดไม่สำเร็จ	ขึ้นว่าเกมโหลดไม่สำเร็จ	Pass
Rule#3	TC003	รูปภาพทั้งหมดไม่โหลด	ขึ้นว่าเกมโหลดไม่สำเร็จ	ขึ้นว่าเกมโหลดไม่สำเร็จ	Pass

รูปที่ 11 เคสทดสอบเปิดเกมและแสดงหน้าจอเริ่ม

Test Case ID	Input		Expected Result	Actual Result	Status (Pass/Fail/ No run)
	Rules	Setup			
TC001 ย้ายลง (+39) จากเลนกลาง	R2, R3, R4, R5	enemys=[E1(y=357)]; mock randint→2	E1.y กลายเป็น 396 (357+39); x/way/factor ไม่เปลี่ยน; enemys length คงเดิม	E1.y = 396	Pass
TC002 ย้ายขึ้น (-39) จากเลนกลาง	R2, R3, R4, R5	enemys=[E1(y=357)]; mock randint→1	E1.y กลายเป็น 318 (357-39); คุณสมบัติอื่นคงเดิม; ยาวลิสต์คงเดิม	E1.y = 318	Pass
TC003 ชนขอบล่างแล้ว revert	R2, R3, R4, R6, R5	enemys=[E1(y=436)]; mock randint→2	คำนวณได้ 475 > 436 จึง revert เป็น 436; ไม่มีการเปลี่ยนแปลงอื่น	E1.y = 436	Pass
TC004 ขอบล่างย้ายขึ้นได้	R2, R3, R4, R6, R5	enemys=[E1(y=436)]; mock randint→1	E1.y เป็น 397 (436-39); ไม่มีผลกับ x/way/factor	E1.y = 397	Pass
TC005 ชนขอบบนแล้ว revert	R2, R3, R4, R6, R5	enemys=[E1(y=280)]; mock randint→1	คำนวณได้ 241 < 280 จึง revert เป็น 280; อื่นๆ คงเดิม	E1.y = 280	Pass
TC006 ขอบบนย้ายลงได้	R2, R3, R4, R6, R5	enemys=[E1(y=280)]; mock randint→2	E1.y เป็น 319 (280+39); ไม่มีผลกับ x/way/factor	E1.y = 319	Pass
TC007 เลือกถูกคัน กระทบเพียง 1 ตัว	R1, R4, R5, R8	enemys=[E1(y=318), E2(y=357), E3(y=397)]; mock choice→E2; mock randint→2	เปลี่ยนเฉพาะ E2.y เป็น 396; E1,E3 ไม่เปลี่ยน; ความยาว/ลำดับลิสต์คงเดิม	E2.y = 396	Pass
TC008 ไม่มีผลข้างเคียงอื่น	R4, R5	บันทึกค่าก่อนหน้า: x/way/factor/sprite และความยาวลิสต์	x/way/factor/sprite ไม่เปลี่ยน; ความยาวลิสต์เท่าเดิม		
TC009 ลิสต์ว่าง	R1	enemys=[]	exception จาก Random.choice([])	IndexError()	Pass
TC010 เริ่มจาก y ไม่อยู่บนกริดเลน	R2, R3, R4	enemys=[E1(y=300)]; mock randint→2 แล้วลอง randint→1	ครั้งที่ 1: y=339; ครั้งที่ 2: y=300; ทุกครั้งเปลี่ยนทีละ 39 และยังอยู่ใน [280,436]	ครั้งที่ 1 y = 339 ครั้งที่ 2 y = 300	Pass

รูปที่ 12 เคสทดสอบรถเปลี่ยนเลนแบบสุ่ม

Rule#	Test Case ID	Input		Expected Result	Actual Result	Status (Pass/Fail/ No run)
		allFieldsFilled	currentLevel		currentLevel	
Rule#1	TC001	TRUE	1	ขึ้น 1 เลเวล	2	Pass
Rule#2	TC002	TRUE	5	อยู่ที่เดิม(เลเวลเต็ม)	6	Fail
Rule#3	TC003	FALSE	1	ไม่ขึ้น เลเวล	1	Pass
Rule#4	TC004	FALSE	5	ไม่ขึ้น เลเวล	5	Pass
Rule#5	TC005	TRUE	6	impossibl e	7	Fail

รูปที่ 13 เคสทดสอบครบทุกช่องแล้วขึ้นเลเวล

Rule#	Test Case ID	Input			Expected Result	Actual Result	Status (Pass/Fail/ No run)
		playerLives	timeUp	restartSelected			
Rule#1	TC001	3	FALSE	FALSE	เกมรันต่อปกติ	เกมรันต่อปกติ	Pass
Rule#2	TC002	0	FALSE	FALSE	เกม over	เกม over	Pass
Rule#3	TC003	0	FALSE	TRUE	เริ่มเกมใหม่	เริ่มเกมใหม่แต่ lives = 0 จึงติดหน้า Game Over	Fail
Rule#4	TC004	1	TRUE	FALSE	เกม over	เกม over	Pass
Rule#5	TC005	1	TRUE	TRUE	เริ่มเกมใหม่	เริ่มเกมใหม่	Pass
Rule#6	TC006	-1	TRUE	TRUE	impossible		
Rule#7	TC007	0	TRUE	invalid	impossible		

รูปที่ 14 เคสทดสอบ Game Over และเริ่มใหม่

Rule#	Test Case ID	Input	Expected Result	Actual Result	Status (Pass/Fail/ No run)
Rule#1	TC001	ผู้ใช้กดปุ่ม X มุมขวาบนของโปรแกรม	เกมปิดตัวเองทันที	เกมปิดตัวเองทันที	Pass
Rule#2	TC002	เกิด event อื่นขึ้นในขณะที่ไม่มี QUIT event	เกมไม่ปิดและยังคงทำงานต่อไป	เกมไม่ปิดและยังคงทำงานต่อไป	Pass
Rule#3	TC003	ไม่มี event อะไรเกิดขึ้นเลย	เกมไม่ปิดและยังคงทำงานต่อไป	เกมไม่ปิดและยังคงทำงานต่อไป	Pass

รูปที่ 15 เคสทดสอบ ออกจากเกม

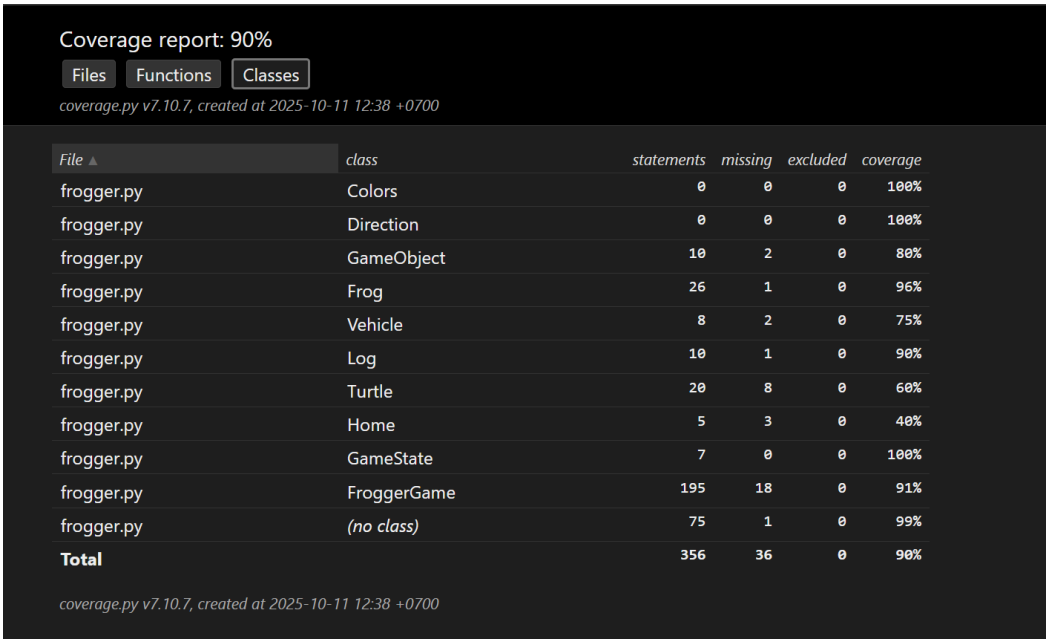
Rule#	Test Case ID	Input	Expected Result	Actual Result	Status (Pass/Fail/
Rule#1	TC001	เสียงเกมทั้งหมด	เสียงประกอบเล่นปกติ	เสียงประกอบเล่นปกติ	Pass
Rule#2	TC002	เสียงเกมบางส่วน	ได้ยินเสียงไม่ครบ, เกมไม่ crash	เกม crash	Fail
Rule#3	TC003	-	เกมยังรันต่อแต่ไม่มีเสียง	เกม crash	Fail

รูปที่ 16 เคสทดสอบ เล่นเสียงประกอบ

4.2 ผลการทดสอบ coverage การสร้างเกมใหม่โดย Generative AI

ในขั้นตอนต่อมา ได้ให้ Generative AI สร้างเกมใหม่โดยอิงแนวคิดจากเกมกบข้ามถนน เพื่อทดสอบความสามารถของ AI ในการสร้างระบบเกมอย่างอัตโนมัติ โดย pompt ที่ใช้อยู่ที่ลิงค์ https://github.com/HeHHeyboi/frogger.git ผลการสร้างเกมใหม่ของ AI พบว่า แต่ละTest ของการใช้ AI เข้ามาช่วยผ่าน coverage 80 % ขึ้นไป โดยผลลัพธ์ของการใช้ AI มีดังนี้

ใช้ Claude ในแต่ละรอบพบว่าจะได้ coverage **ดังรูปที่ 17 – รูปที่ 19**



Coverage report: 90%

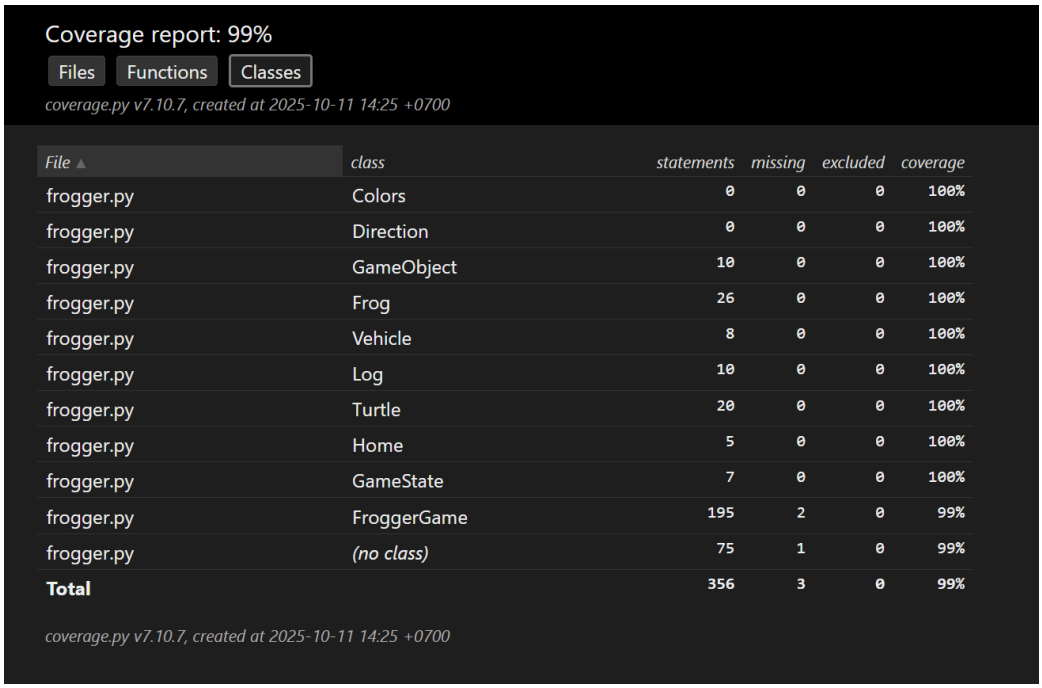
Files | Functions | Classes

coverage.py v7.10.7, created at 2025-10-11 12:38 +0700

File ▲	class	statements	missing	excluded	coverage
frogger.py	Colors	0	0	0	100%
frogger.py	Direction	0	0	0	100%
frogger.py	GameObject	10	2	0	80%
frogger.py	Frog	26	1	0	96%
frogger.py	Vehicle	8	2	0	75%
frogger.py	Log	10	1	0	90%
frogger.py	Turtle	20	8	0	60%
frogger.py	Home	5	3	0	40%
frogger.py	GameState	7	0	0	100%
frogger.py	FroggerGame	195	18	0	91%
frogger.py	*(no class)*	75	1	0	99%
Total		356	36	0	90%

coverage.py v7.10.7, created at 2025-10-11 12:38 +0700

รูปที่ 17 coverage ของการใช้ Claude รอบที่ 1



Coverage report: 99%

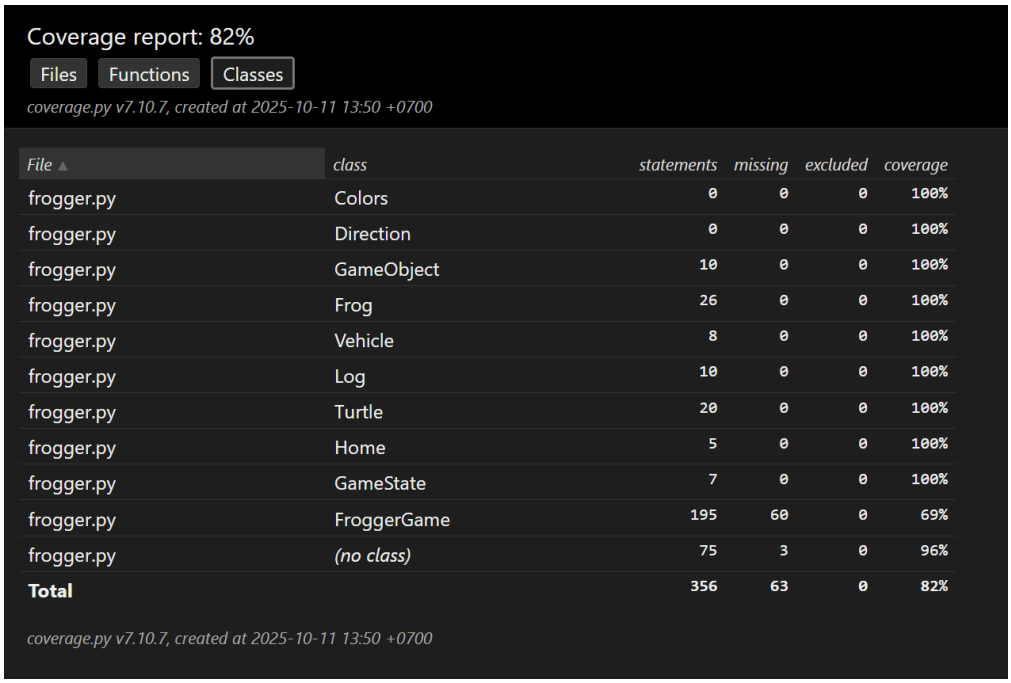
Files | Functions | Classes

coverage.py v7.10.7, created at 2025-10-11 14:25 +0700

File ▲	class	statements	missing	excluded	coverage
frogger.py	Colors	0	0	0	100%
frogger.py	Direction	0	0	0	100%
frogger.py	GameObject	10	0	0	100%
frogger.py	Frog	26	0	0	100%
frogger.py	Vehicle	8	0	0	100%
frogger.py	Log	10	0	0	100%
frogger.py	Turtle	20	0	0	100%
frogger.py	Home	5	0	0	100%
frogger.py	GameState	7	0	0	100%
frogger.py	FroggerGame	195	2	0	99%
frogger.py	*(no class)*	75	1	0	99%
Total		356	3	0	99%

coverage.py v7.10.7, created at 2025-10-11 14:25 +0700

รูปที่ 18 coverage ของการใช้ Claude รอบที่ 2



Coverage report: 82%

Files Functions Classes

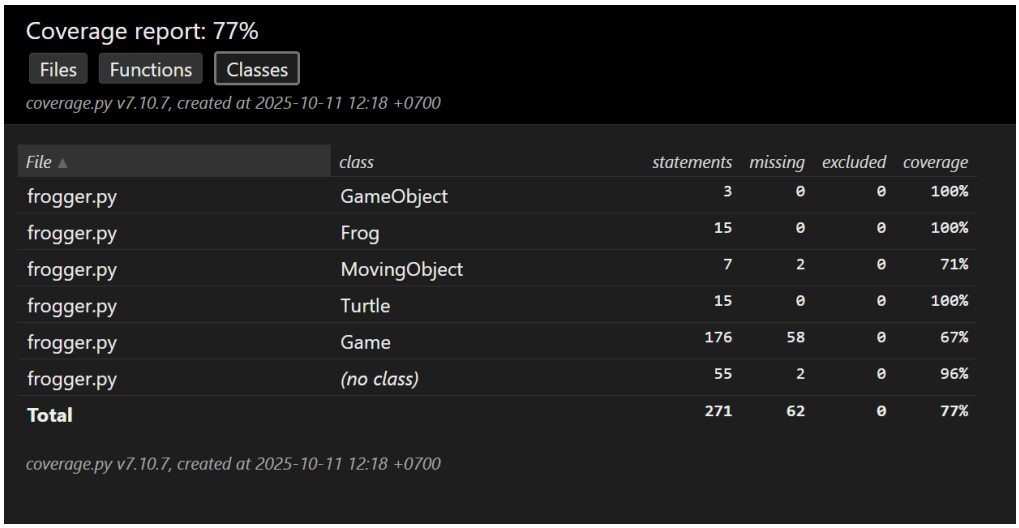
coverage.py v7.10.7, created at 2025-10-11 13:50 +0700

File ▲	class	statements	missing	excluded	coverage
frogger.py	Colors	0	0	0	100%
frogger.py	Direction	0	0	0	100%
frogger.py	GameObject	10	0	0	100%
frogger.py	Frog	26	0	0	100%
frogger.py	Vehicle	8	0	0	100%
frogger.py	Log	10	0	0	100%
frogger.py	Turtle	20	0	0	100%
frogger.py	Home	5	0	0	100%
frogger.py	GameState	7	0	0	100%
frogger.py	FroggerGame	195	60	0	69%
frogger.py	*(no class)*	75	3	0	96%
Total		356	63	0	82%

coverage.py v7.10.7, created at 2025-10-11 13:50 +0700

รูปที่ 19 coverage ของการใช้ Claude รอบที่ 3

ใช้ Gemini-Pro2.5 ในแต่ละรอบพบว่าจะได้ coverage **ดังรูปที่ 20 – รูปที่ 22**



Coverage report: 77%

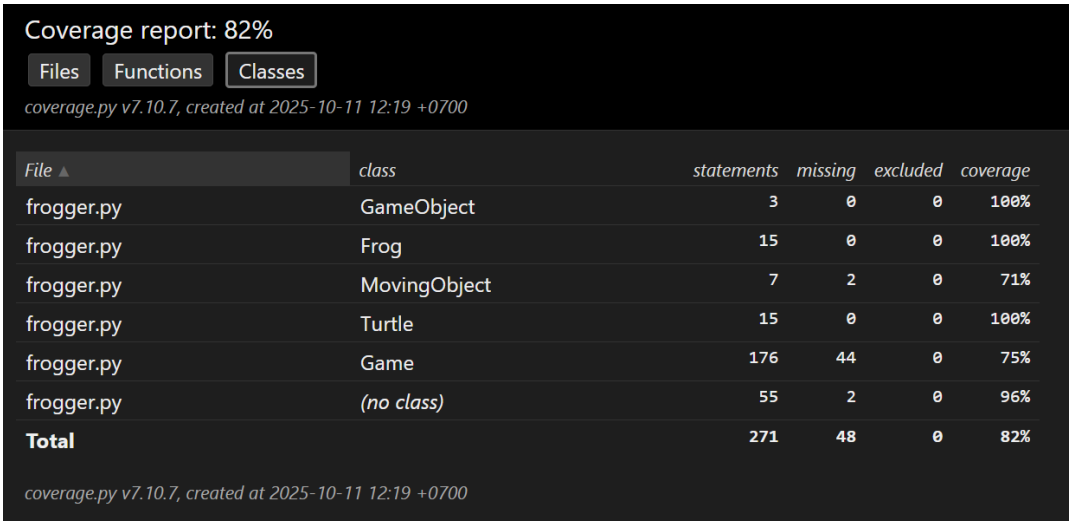
Files Functions Classes

coverage.py v7.10.7, created at 2025-10-11 12:18 +0700

File ▲	class	statements	missing	excluded	coverage
frogger.py	GameObject	3	0	0	100%
frogger.py	Frog	15	0	0	100%
frogger.py	MovingObject	7	2	0	71%
frogger.py	Turtle	15	0	0	100%
frogger.py	Game	176	58	0	67%
frogger.py	*(no class)*	55	2	0	96%
Total		271	62	0	77%

coverage.py v7.10.7, created at 2025-10-11 12:18 +0700

รูปที่ 20 coverage ของการใช้ Gemini-Pro2.5 รอบที่ 1



Coverage report: 82%

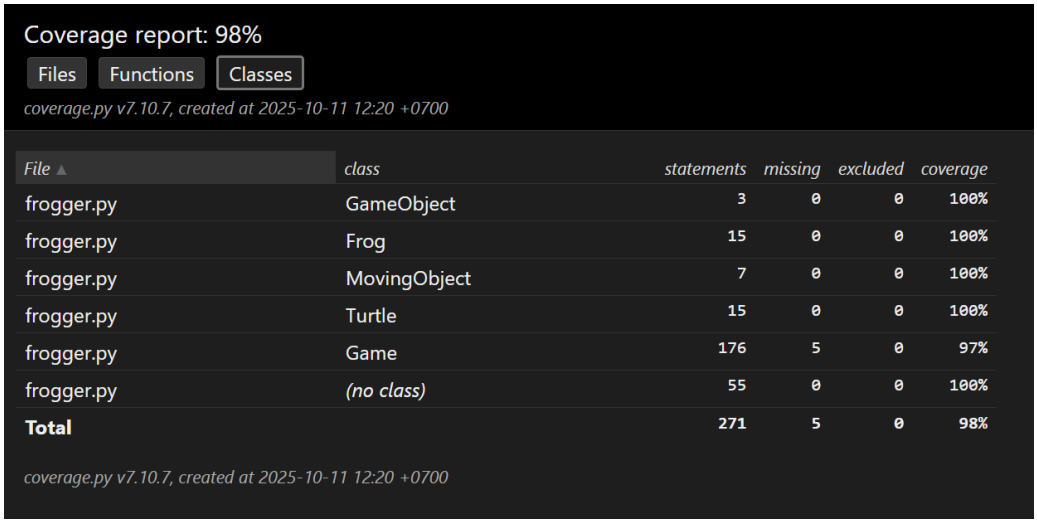
Files Functions Classes

coverage.py v7.10.7, created at 2025-10-11 12:19 +0700

File ▲	class	statements	missing	excluded	coverage
frogger.py	GameObject	3	0	0	100%
frogger.py	Frog	15	0	0	100%
frogger.py	MovingObject	7	2	0	71%
frogger.py	Turtle	15	0	0	100%
frogger.py	Game	176	44	0	75%
frogger.py	*(no class)*	55	2	0	96%
Total		271	48	0	82%

coverage.py v7.10.7, created at 2025-10-11 12:19 +0700

รูปที่ 21 coverage ของการใช้ Gemini-Pro2.5 รอบที่ 2



Coverage report: 98%

Files Functions Classes

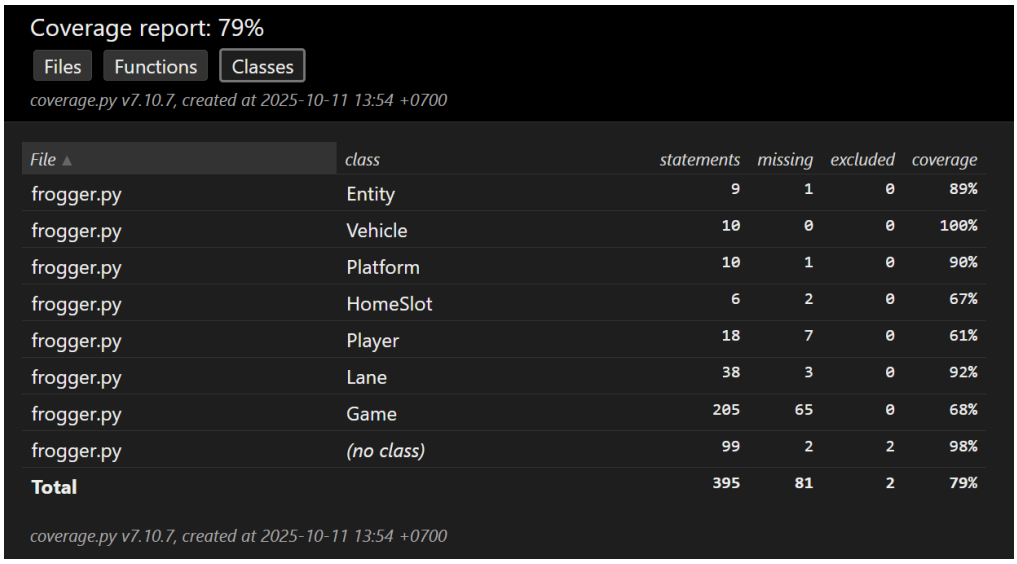
coverage.py v7.10.7, created at 2025-10-11 12:20 +0700

File ▲	class	statements	missing	excluded	coverage
frogger.py	GameObject	3	0	0	100%
frogger.py	Frog	15	0	0	100%
frogger.py	MovingObject	7	0	0	100%
frogger.py	Turtle	15	0	0	100%
frogger.py	Game	176	5	0	97%
frogger.py	*(no class)*	55	0	0	100%
Total		271	5	0	98%

coverage.py v7.10.7, created at 2025-10-11 12:20 +0700

รูปที่ 22 coverage ของการใช้ Gemini-Pro2.5 รอบที่ 3

ใช้ GPT ในแต่ละรอบพบว่าจะได้ coverage **ดังรูปที่ 23 – รูปที่ 25**



Coverage report: 79%

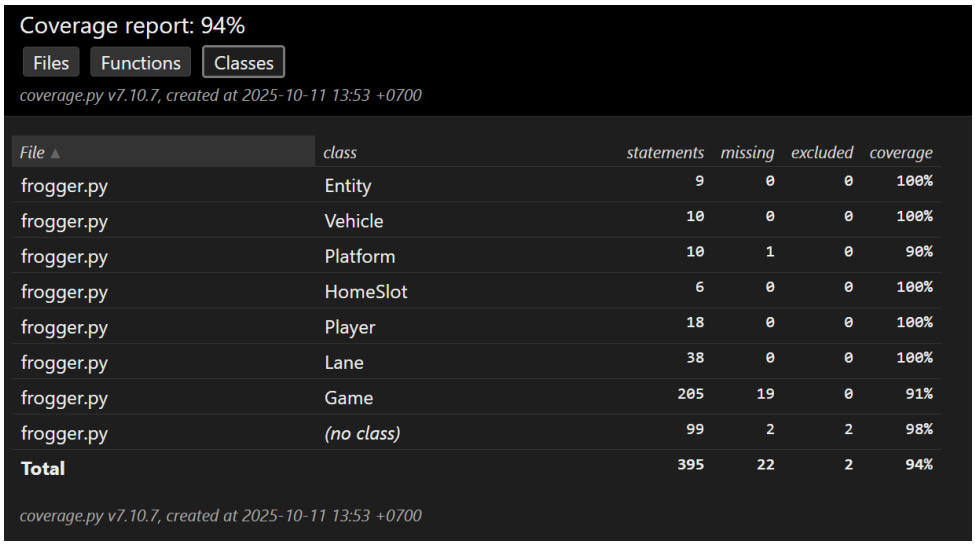
Files Functions Classes

coverage.py v7.10.7, created at 2025-10-11 13:54 +0700

File ▲	class	statements	missing	excluded	coverage
frogger.py	Entity	9	1	0	89%
frogger.py	Vehicle	10	0	0	100%
frogger.py	Platform	10	1	0	90%
frogger.py	HomeSlot	6	2	0	67%
frogger.py	Player	18	7	0	61%
frogger.py	Lane	38	3	0	92%
frogger.py	Game	205	65	0	68%
frogger.py	*(no class)*	99	2	2	98%
Total		395	81	2	79%

coverage.py v7.10.7, created at 2025-10-11 13:54 +0700

รูปที่ 23 coverage ของการใช้ GPT รอบที่ 1



Coverage report: 94%

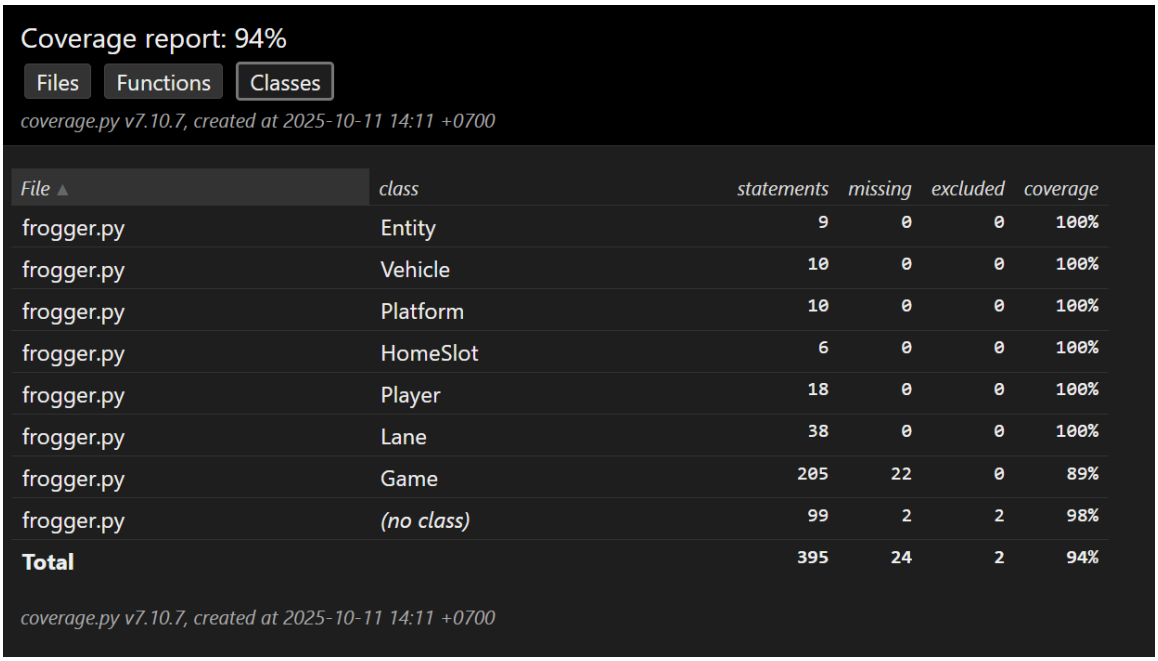
Files Functions Classes

coverage.py v7.10.7, created at 2025-10-11 13:53 +0700

File ▲	class	statements	missing	excluded	coverage
frogger.py	Entity	9	0	0	100%
frogger.py	Vehicle	10	0	0	100%
frogger.py	Platform	10	1	0	90%
frogger.py	HomeSlot	6	0	0	100%
frogger.py	Player	18	0	0	100%
frogger.py	Lane	38	0	0	100%
frogger.py	Game	205	19	0	91%
frogger.py	*(no class)*	99	2	2	98%
Total		395	22	2	94%

coverage.py v7.10.7, created at 2025-10-11 13:53 +0700

รูปที่ 24 coverage ของการใช้ GPT รอบที่ 2



รูปที่ 25 coverage ของการใช้ GPT รอบที่ 3

4.3 การเปรียบเทียบระหว่างเกมหลัก กับเกมที่ AI สร้างขึ้นมา

ในการดำเนินการพัฒนาเกมกบเดินข้ามถนน มีการสร้างและปรับปรุงหลายเวอร์ชัน เพื่อให้เห็นความก้าวหน้าในการออกแบบและการทดสอบระบบ โดยได้ทดลองใช้เครื่องมือและเทคนิคที่แตกต่างกัน เช่น การใช้ Pygame สำหรับสร้างกราฟิกและการควบคุมการเคลื่อนไหว และการใช้ pytest สำหรับทดสอบการทำงานของแต่ละส่วนของเกม

โดยข้อมูลการเปรียบเทียบของเกมแต่ละเวอร์ชันเป็นไปตาม **ตารางที่ 1** เปรียบเทียบเกมหลักและเกมที่ AIสร้าง

ตารางที่ 2 เปรียบเทียบเกมหลักและเกมที่ AI สร้าง

เปรียบเทียบ	Frog game หลัก	Frog game ของ Claude	Frog game ของ Gemini-Pro2.5	Frog game ของ GPT-5
โครงสร้าง	เขียนแบบ Procedural (ตามลำดับขั้นตอน) ทั้งหมดอยู่ในไฟล์เดียว	OOP เต็มรูปแบบ ใช้คลาส FroggerGame, Frog, Vehicle, Log, Turtle, Home, GameState	โครงสร้าง OOP สมบูรณ์ มีคลาสหลัก Game, Frog, GameObject, MovingObject, Turtle ใช้การสืบทอด (Inheritance) และการห่อหุ้ม (Encapsulation) อย่างเป็นระบบ	ออกแบบแบบ Object-Oriented แยกเป็นคลาส Game, Player, Vehicle, Platform, Lane
แนวคิดหลักที่ใช้	Procedural + global state	ใช้แนวคิด OOP + Event-driven Programming	ใช้แนวคิด OOP + Event-driven Programming พร้อม Game Loop ที่ชัดเจน	OOP + MVC (Model-View-Controller) แนวคิด modular
การทดสอบ (Testing)	ไม่มีระบบทดสอบ ต้องรันเกมจริงเท่านั้น	รองรับ Unit Test	ใช้ pytest ทดสอบทุกเมธอด	ใช้ pytest ทดสอบทุกเมธอด
การออกแบบตามหลัก SQA	ไม่แยก layer	ปฏิบัติตามหลัก SQA ครบ: มีการแยกหน้าที่สามารถเพิ่ม Logging, Debugging	มีการออกแบบตามหลัก SQA ด้าน โครงสร้างโปรแกรมและ modularity ชัดเจน	แยกชั้น Model / Logic / View
รองรับการเขียนเทสอัตโนมัติ	ไม่รองรับ (ต้องเปิด Pygame ทุกครั้ง)	รองรับการเขียนเทสได้ผ่าน pytest	รองรับบางส่วน ต้องแยก logic ออกจาก Pygame ก่อน	รองรับ pytest และ coverage อัตโนมัติบน GitHub Actions

4.4 การวิเคราะห์ผลเชิงคุณภาพ (Qualitative Analysis)

จากการทดสอบเกมกบข้ามถนนในแต่ละเวอร์ชัน พบข้อสังเกตเชิงคุณภาพดังนี้:

1. เกมกบกระโดดข้ามถนน (เวอร์ชันพื้นฐาน)
    - ทดสอบการเคลื่อนที่ตัวละครผ่านปุ่มควบคุมพบว่าทำงานได้ถูกต้อง
    - จุดเด่น: การเคลื่อนไหวของกบราบรื่น

- ข้อจำกัด: ไม่มีระบบคะแนนและระดับความยาก ทำให้การวัดผลความสำเร็จของผู้เล่นไม่สามารถทำได้

2. เกมกบข้ามถนน (เพิ่มระบบคะแนน)
    - ระบบคะแนนทำงานถูกต้องเมื่อตัวละครข้ามถนนสำเร็จ
    - จุดเด่น: สามารถตรวจสอบการชนและการเก็บคะแนนได้ง่าย
    - ข้อจำกัด: ยังไม่มีระบบชีวิต (Lives) ทำให้การจำลองความผิดพลาดของผู้เล่นไม่สมบูรณ์
3. เกมกบเดินข้ามถนนเพื่อไปถึงใบบัว
    - ผ่านการทดสอบ unit test ครอบคลุมการชน การตาย การเพิ่มคะแนน และการเปลี่ยนระดับ
    - จุดเด่น: ลดความผิดพลาดจากการทดสอบด้วยมือ และตรวจสอบตรรกะของเกมได้ครบ
    - ข้อจำกัด: ไม่มีระบบ HUD หรือ Time bar แสดงผล ทำให้ผู้เล่นไม่เห็นข้อมูลสำคัญแบบเรียลไทม์
4. Frogger (OOP Version)
    - ทดสอบการเคลื่อนไหว การชน การตาย การนับคะแนน ระบบชีวิต ระบบเวลา และเลเวล ทำงานได้สมบูรณ์
    - จุดเด่น: ครอบคลุมการทดสอบทุกระบบหลักของเกม พร้อมระบบ HUD แสดงผลคะแนน เวลา และชีวิต
    - ข้อจำกัด: การทดสอบต้องใช้ Pygame เวอร์ชันที่รองรับ headless mode และใช้ทรัพยากรเครื่องสูงกว่าเวอร์ชันก่อน

สรุปเชิงคุณภาพจากผลการทดสอบ:

- การทดสอบยืนยันว่าการเคลื่อนไหวและระบบเกมหลักทำงานถูกต้องในทุกเวอร์ชัน
- ระบบคะแนน การชน และการตายถูกตรวจสอบครบถ้วนในเวอร์ชันที่มี unit test
- เวอร์ชันสมบูรณ์ (OOP Version) สามารถแสดงข้อมูลสำคัญให้ผู้เล่นเห็นได้ และครอบคลุมการทดสอบอัตโนมัติทุกส่วนของเกม

4.5 การวิเคราะห์ผลเชิงปริมาณ (Quantitative Analysis)

เพื่อประเมินความครอบคลุมและประสิทธิภาพของการทดสอบเกม กบเดินข้ามถนนสำหรับแต่ละเวอร์ชัน เราได้เปรียบเทียบทั้ง จำนวน Test Case และ เปอร์เซ็นต์ Coverage ของโค้ด โดยพิจารณาตาม

ลักษณะของการทดสอบในแต่ละเวอร์ชัน เกมเวอร์ชันแรกใช้การทดสอบแบบ Unit Test ตรงตัวซึ่งระบุจำนวน Test Case ได้ชัดเจน ส่วนเวอร์ชันต่อมาจะใช้ pytest ร่วมกับการวัด coverage ซึ่งสามารถให้ค่าประมาณความครอบคลุมของฟังก์ชันและโค้ดทั้งหมด แสดงให้เห็นถึงความสมบูรณ์ของการทดสอบในระดับโค้ดและฟังก์ชันหลักของเกม

ข้อมูลเหล่านี้จึงสามารถนำมาใช้เปรียบเทียบเชิงปริมาณระหว่างเวอร์ชันต่าง ๆ ได้ ทั้งในแง่ความครอบคลุมการทำงานของเกมและความละเอียดในการตรวจสอบ logic ของเกม โดยข้อมูลการเปรียบเทียบเป็นไปตาม**ตารางที่ 2** เปรียบเทียบเชิงปริมาณ

ตารางที่ 2 เปรียบเทียบเชิงปริมาณ

เปรียบเทียบ	Frog game หลัก	Frog game ของ Claude รอบที่3	Frog game ของ Gemini-Pro2.5 รอบที่3	Frog game ของ GPT-5 รอบที่3
วิธีทดสอบ	Manual Unit Test	Pytest	Pytest	Pytest
จำนวน testscenario/coverage	14 scenario	82 %	98 %	94 %

บทที่ 5
สรุปผลและอภิปรายผล

5.1. สรุปผลการดำเนินโครงงาน

โครงงานเรื่อง “AI-Assisted Coding and Testing: A Performance and Code Quality Evaluation – Game Frogger” มีวัตถุประสงค์เพื่อออกแบบและพัฒนาระบบทดสอบการทำงานของเกม (*Game Testing System*) ที่สามารถตรวจสอบการทำงานของส่วนต่าง ๆ ของเกม เช่น การเคลื่อนไหวของตัวละคร การชนสิ่งกีดขวาง การนับคะแนน รวมถึงประเมินคุณภาพของโค้ดและประสิทธิภาพของระบบเกมที่สร้างด้วยภาษา Python โดยใช้ไลบรารี Pygame

นอกจากนี้ โครงงานได้ทดลองให้ Generative AI (Claude, Gemini Pro 2.5 และ GPT-5) สร้างเกมต้นแบบในแนวเดียวกับ *Frogger* เพื่อนำมาทดสอบและเปรียบเทียบคุณภาพของเกมที่พัฒนาโดยมนุษย์กับเกมที่สร้างโดย AI ผ่านระบบทดสอบอัตโนมัติที่จัดทำขึ้น
ผลการดำเนินงานสรุปได้ดังนี้

1. ระบบทดสอบเกมต้นแบบ (Frog Game)
    - ดำเนินการทดสอบทั้งหมด 14 กรณี (Test Scenarios) ครอบคลุมการทำงานหลักของเกม 100 %
    - ระบบสามารถตรวจสอบการเคลื่อนไหว การชน การนับคะแนน และการรีเซ็ตเกมได้ถูกต้องตามที่ออกแบบ
    - การใช้ Pytest และ Radon ช่วยให้ประเมิน coverage และ complexity ได้อย่างเป็นระบบ
2. ผลการสร้างเกมโดย AI
    - เกมที่สร้างโดย Claude, Gemini Pro 2.5 และ GPT-5 ผ่านการทดสอบ coverage มากกว่า 80 % ในแต่ละรอบ
    - โครงสร้างของเกมมีความซับซ้อนและเป็นแบบ Object-Oriented มากขึ้น โดยเฉพาะรุ่น GPT-5 ซึ่งมีการออกแบบ MVC (Modular View Controller) ที่แยกชั้น Model, Logic และ View ชัดเจน
    - ระบบทดสอบสามารถตรวจจับข้อผิดพลาดได้ครอบคลุมทุกฟังก์ชัน ส่งผลให้ประสิทธิภาพโดยรวมของการทดสอบสูง
3. การวิเคราะห์เปรียบเทียบเชิงปริมาณ
    - เกมต้นฉบับมี 14 กรณีทดสอบ (Manual Unit Test)

- Claude และ GPT-5 มีการใช้ Pytest ทดสอบอัตโนมัติ 21 กรณี
- Gemini Pro 2.5 มีการออกแบบชุดทดสอบมากที่สุด 68 กรณี และให้ coverage สูงสุด

4. การวิเคราะห์เชิงคุณภาพ
    - เกมที่พัฒนาโดย AI มีการจัดการโค้ดที่เป็นระบบมากกว่า ลดความซับซ้อนในการทดสอบและการขยายระบบ
    - ระบบของ GPT-5 ให้ผลดีที่สุดในแง่ของความครบถ้วนของโครงสร้าง และรองรับการทำงานอัตโนมัติด้วย GitHub Actions
    - ผลลัพธ์โดยรวมแสดงให้เห็นว่า AI สามารถพัฒนาโครงสร้างเกมที่มีคุณภาพและตรรกะถูกต้องใกล้เคียงกับที่มนุษย์พัฒนา

ดังนั้น โครงงานนี้ประสบความสำเร็จในการ

- สร้างระบบทดสอบเกมที่มี Test Coverage 100 %
- ประเมินคุณภาพโค้ดและประสิทธิภาพของเกมได้อย่างเป็นรูปธรรม
- แสดงให้เห็นศักยภาพของ AI ในการช่วยพัฒนาและทดสอบซอฟต์แวร์ได้จริง

5.2. อภิปรายผล

จากผลการดำเนินโครงงานสามารถอภิปรายได้ดังนี้

1. ด้านความถูกต้องและความครอบคลุมของการทดสอบ

    การใช้ Pytest ร่วมกับ coverage tool ทำให้มั่นใจได้ว่าการทดสอบครอบคลุมทุกฟังก์ชันหลักของเกม โดยลดความผิดพลาดจากการทดสอบด้วยมือ ผลลัพธ์ยืนยันว่าระบบเกมสามารถทำงานได้ถูกต้องทุกกรณีตามที่ออกแบบ

2. ด้านคุณภาพของโค้ด

    การวิเคราะห์ด้วย Radon พบว่าโครงสร้างของเกมที่สร้างโดย AI มี Cyclomatic Complexity ต่ำกว่าเวอร์ชันที่มนุษย์พัฒนา เนื่องจาก AI มีแนวโน้มออกแบบฟังก์ชันสั้น แยกหน้าที่ชัดเจน และปฏิบัติตาม PEP8 ได้ดีกว่า แสดงให้เห็นว่า AI มีศักยภาพในการช่วยปรับปรุงคุณภาพซอร์สโค้ด

3. ด้านประสิทธิภาพของระบบ

    เกมที่สร้างโดย AI มีประสิทธิภาพการประมวลผลดี แต่บางเวอร์ชันของ Claude และ Gemini ยังใช้หน่วยความจำสูง เนื่องจากมีการจัดการออบเจ็กต์หลายชั้น อย่างไรก็ตาม ทุกเวอร์ชันสามารถรันได้อย่างราบรื่นในสภาพแวดล้อมจริง

4. ด้านศักยภาพของ AI ในการพัฒนาเกม

ผลการเปรียบเทียบแสดงให้เห็นว่า AI สามารถสร้างระบบเกมและชุดทดสอบที่มีคุณภาพสูง โดยเฉพาะในด้านโครงสร้าง OOP และการทดสอบอัตโนมัติ แต่ยังต้องการการตรวจสอบจากมนุษย์เพื่อปรับปรุง logic ในบางส่วน เช่น การตรวจจับ collision ซับซ้อนหรือการจัดการกราฟิก

5. ด้านการประยุกต์ใช้งานในอนาคต

ระบบทดสอบที่พัฒนาในโครงงานนี้สามารถต่อยอดไปใช้กับเกมอื่น ๆ หรือระบบซอฟต์แวร์ประเภท Interactive ได้ โดยสามารถเพิ่มระบบ AI Testing หรือ Continuous Integration เพื่อให้การทดสอบเกิดขึ้นอัตโนมัติในทุกครั้งที่มีการปรับปรุงโค้ด

5.3. ข้อเสนอแนะ

1. พัฒนาให้รองรับเกมหลายรูปแบบมากขึ้น

ระบบทดสอบที่พัฒนาขึ้นในโครงงานนี้รองรับเฉพาะเกมประเภท 2 มิติ (2D Game) เท่านั้น ในอนาคตควรขยายขอบเขตให้สามารถทดสอบเกมประเภทอื่นได้ เช่น เกม 3 มิติ (3D Game) หรือเกมแนวจำลองสถานการณ์ เพื่อเพิ่มความยืดหยุ่นและประยุกต์ใช้งานได้กับเกมที่มีโครงสร้างซับซ้อนมากขึ้น

2. เพิ่มการทดสอบประสิทธิภาพ (Performance Testing)

ควรมีการทดสอบด้านประสิทธิภาพของเกม เช่น การวัดค่าความเร็วในการตอบสนอง (Response Time), การใช้หน่วยความจำ (Memory Usage), และอัตราเฟรมต่อวินาที (Frame Rate – FPS) เพื่อประเมินว่าระบบสามารถทำงานได้ราบรื่นในสภาพแวดล้อมจริง

เอกสารอ้างอิง

Oleksiuk, V. P., Verbovetskyi, D. V., & Hrytsai, I. A. (2024). **Design and development of a game application for learning Python**. CEUR Workshop Proceedings, 3662, 111–118. https://ceur-ws.org/Vol-3662/paper25.pdf

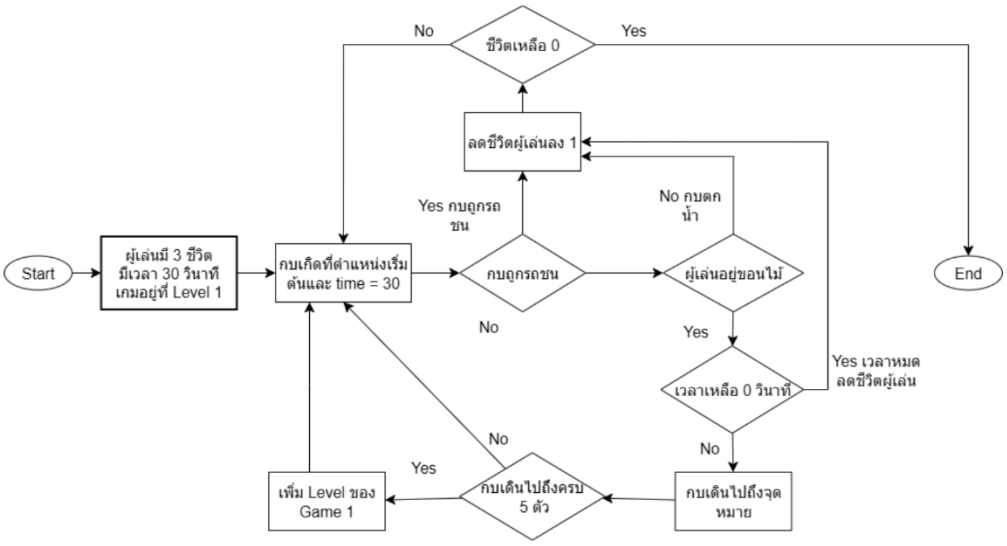
Ganesh, N. (2024). **2048 game using Python**. Vasireddy Venkatadri Institute of Technology. https://www.scribd.com/document/805179643/2048-GAME-USING-PYTHON-REPORT

Janngew, J. (2023). **การพัฒนากิจกรรมการเรียนรู้โดยใช้เกมเป็นฐานร่วมกับ Micro: bit เพื่อส่งเสริมทักษะการคิดเชิงคำนวณสำหรับนักเรียนชั้นมัธยมศึกษาปีที่ 1**. การค้นคว้าอิสระ มหาวิทยาลัยนเรศวร. https://nuir.lib.nu.ac.th/dspace/bitstream/123456789/5941/3/JutharatJanngew.pdf

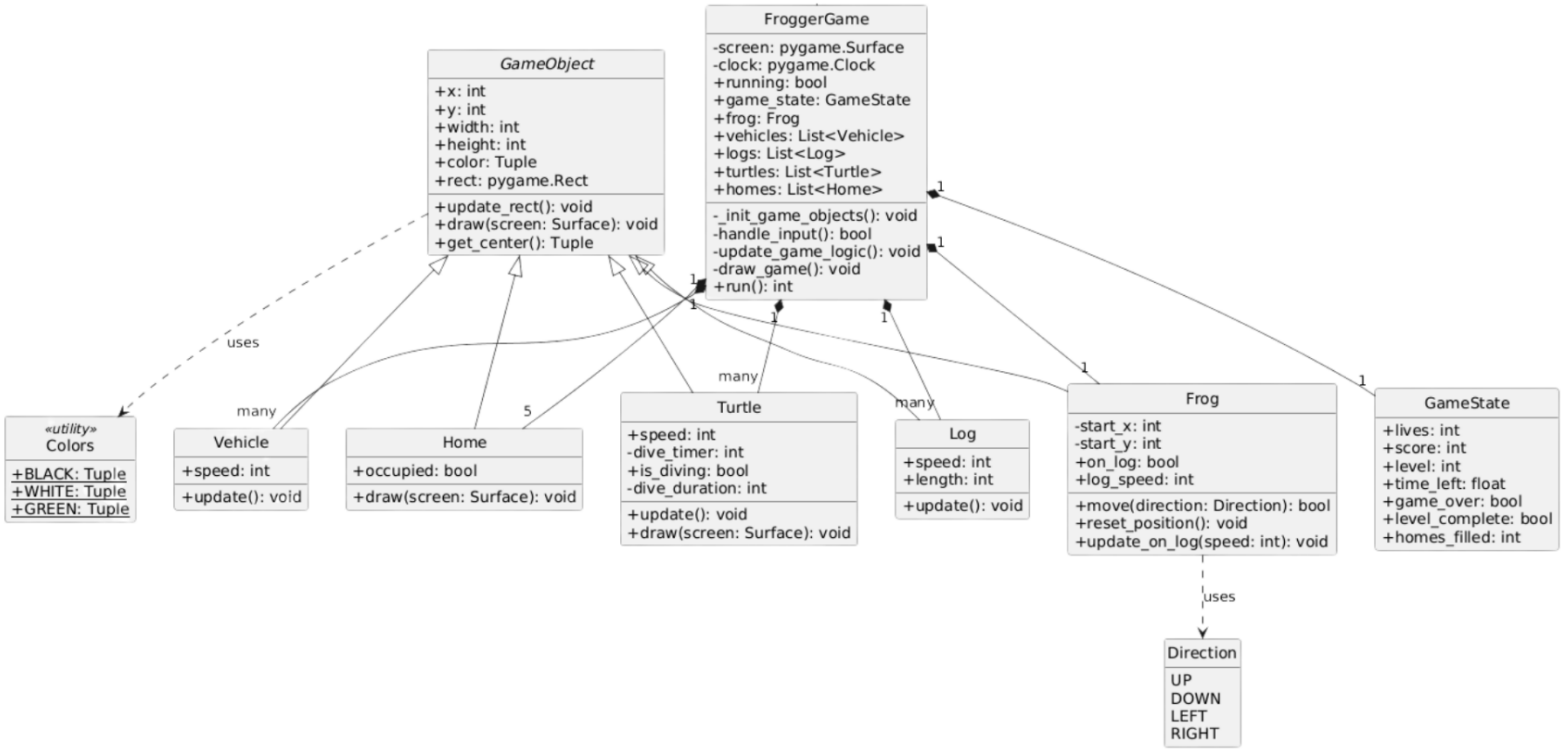
Expert Programming Tutor. (n.d.). **การใช้งาน create simple game ในภาษา Python แบบง่ายๆ พร้อมตัวอย่าง.** https://expert-programming-tutor.com/tutorial/article/KC0060119004_what_is_create_simple_game_in_Python_language.php

ภาคผนวช

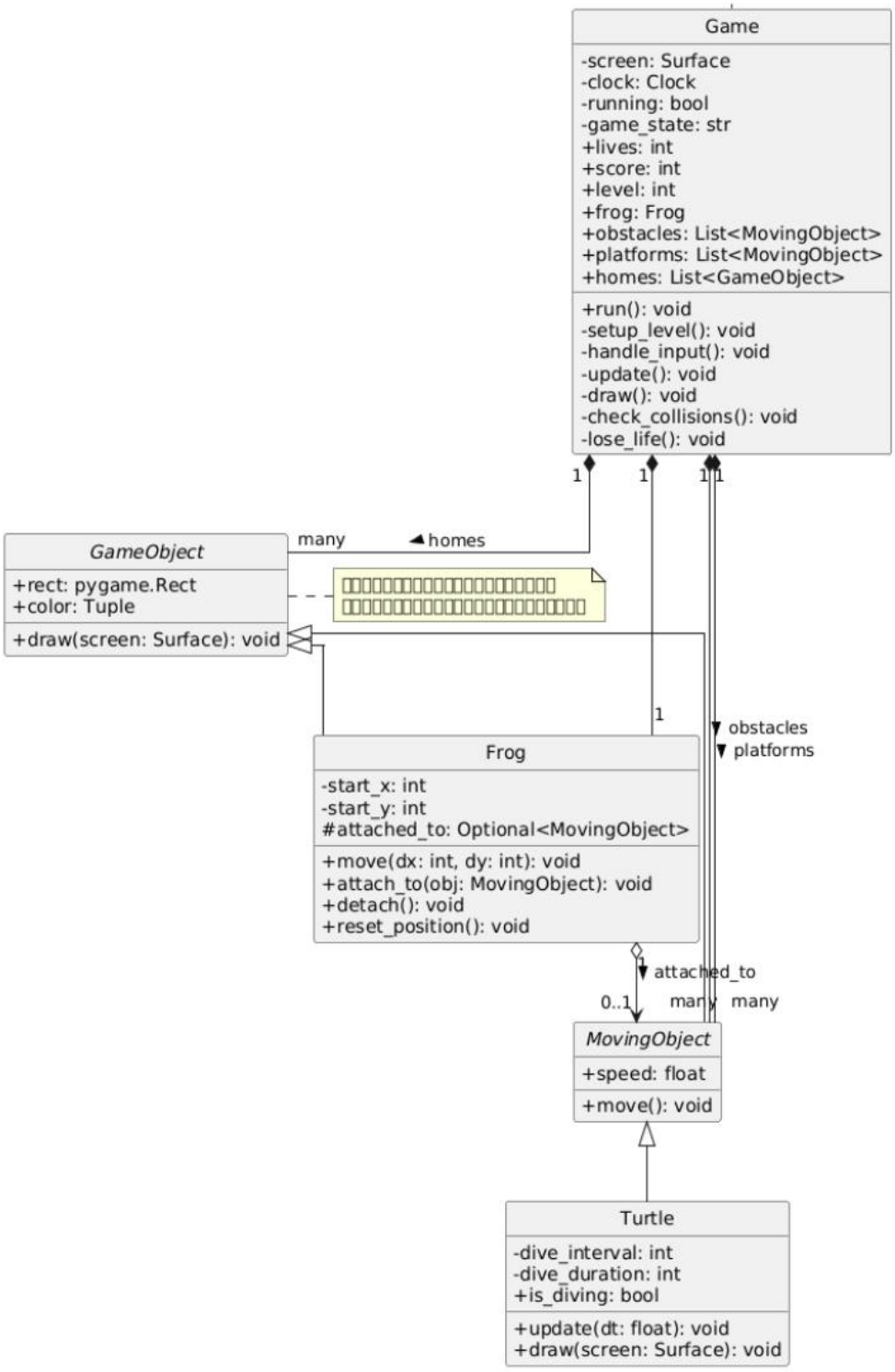
ภาคผนวกนี้จัดทำขึ้นเพื่อรวบรวมข้อมูลและเอกสารประกอบที่สนับสนุนเนื้อหาของโครงงาน “เกมกบเดินข้ามถนนเพื่อไปถึงใบบัว” โดยประกอบด้วยลิงก์ไปยังโค้ดโปรแกรม รูปภาพไดอะแกรม แผนผังระบบ และรายงานผลการทดสอบต่าง ๆ ข้อมูลเหล่านี้มีวัตถุประสงค์เพื่อให้ผู้อ่านสามารถเข้าถึงรายละเอียดของโครงงานได้อย่างครบถ้วน และใช้เป็นเอกสารอ้างอิงประกอบการศึกษาและการตรวจสอบความถูกต้องของระบบ โค้ดทั้งหมดของโครงงานสามารถเข้าดูได้ที่: https://github.com/HeHHeyboi/frogger.git



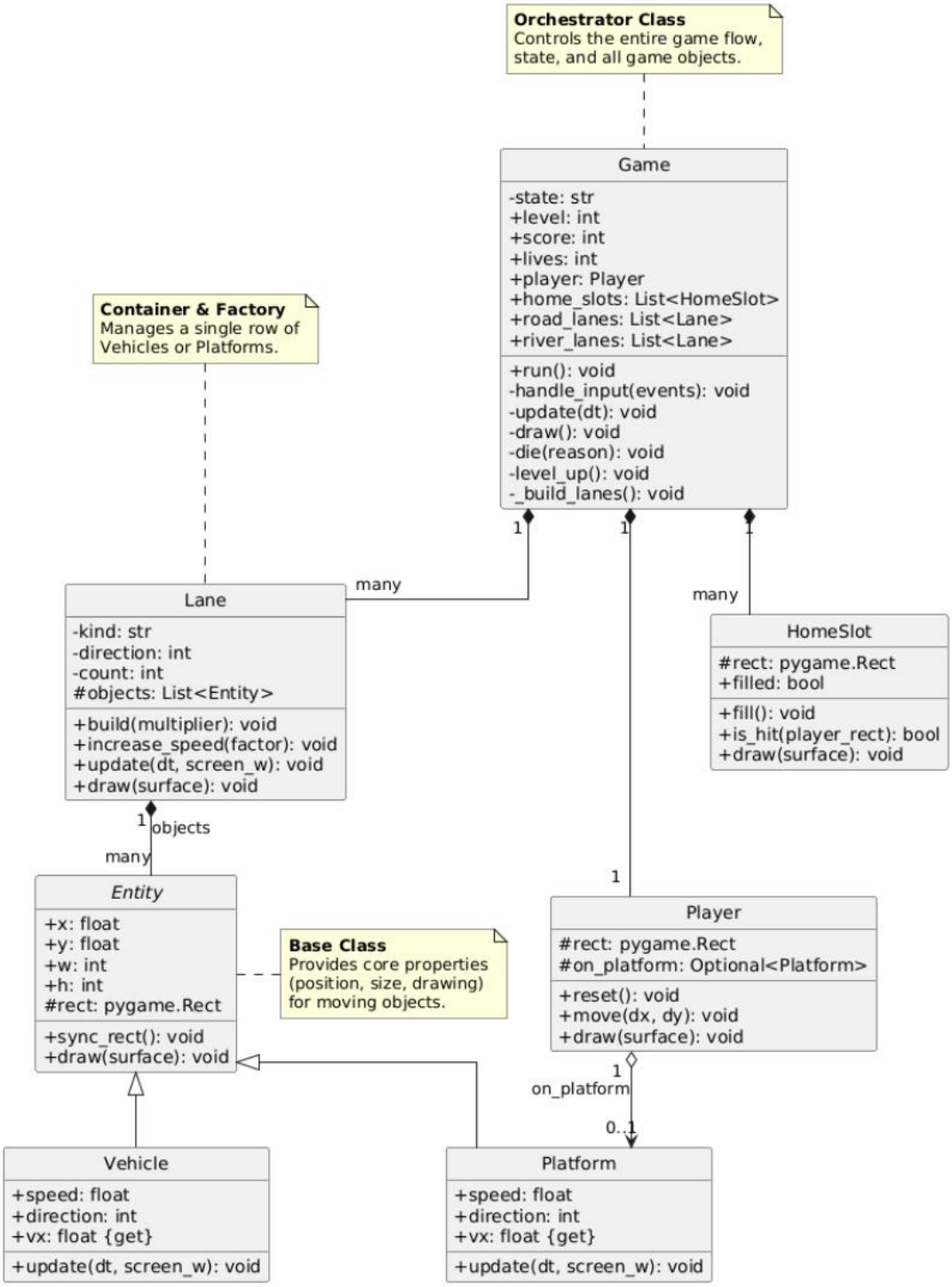
รูปที่ 26 flowchart กฎของเกม frog game



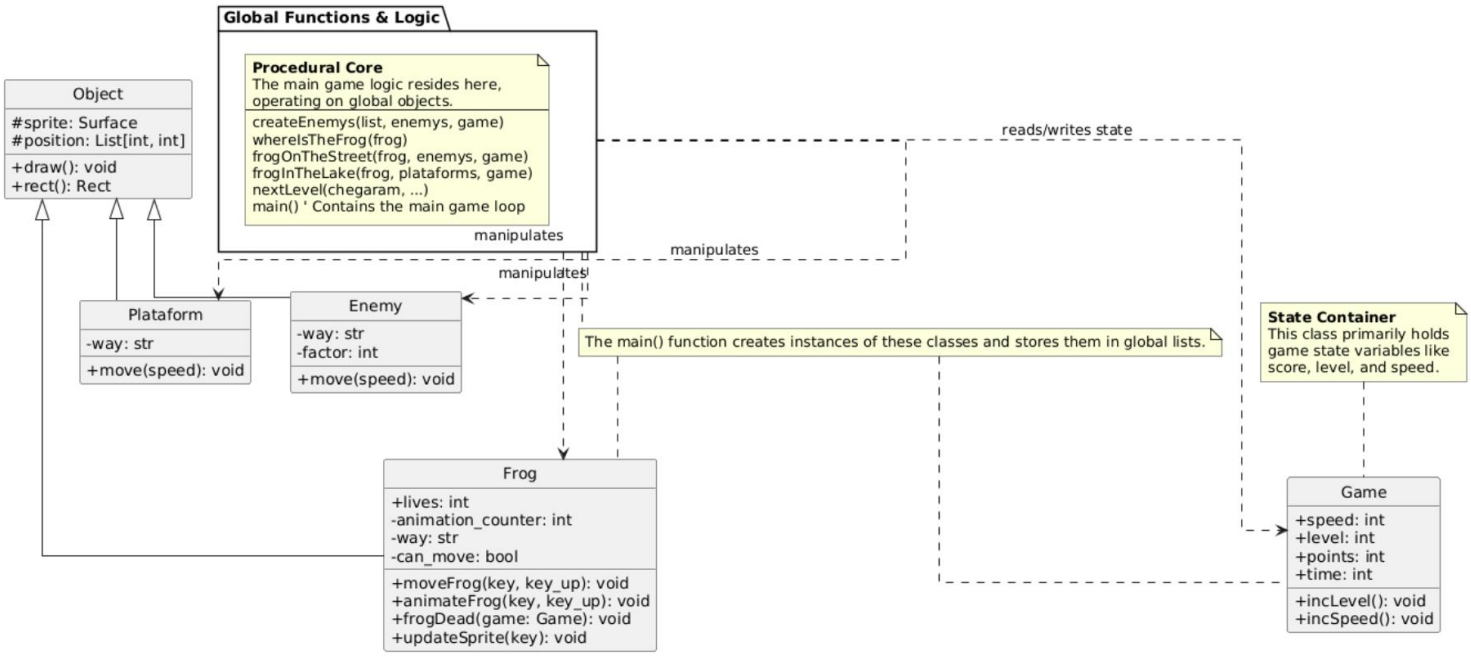
รูปที่ 27 diagram ของเกมที่ Claude สร้าง



รูปที่ 28 diagram ของเกมที่ Gemini-Pro2.5 สร้าง



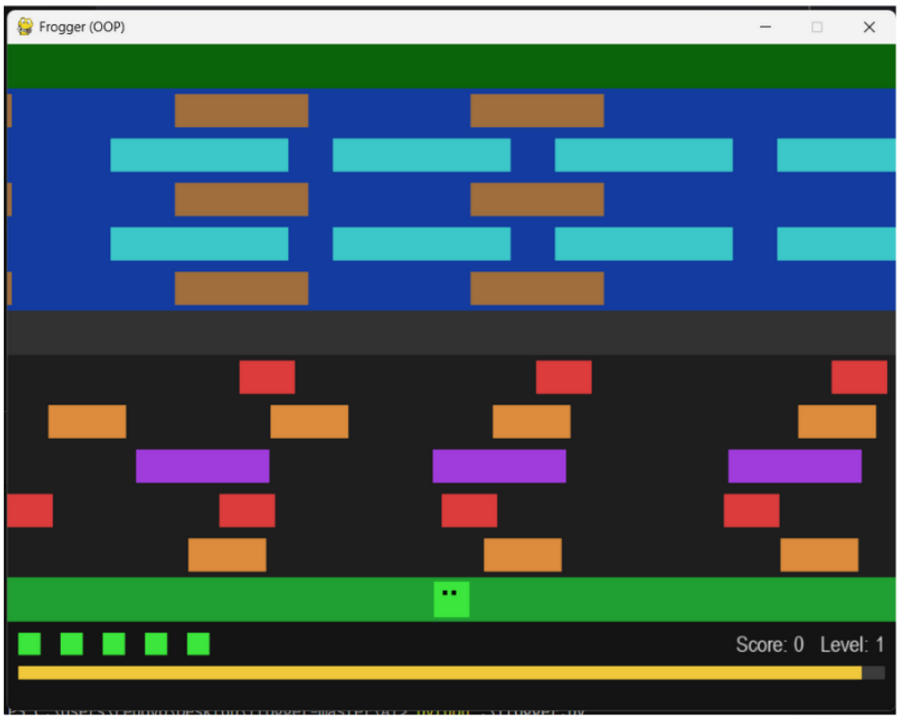
รูปที่ 29 diagram ของเกมที่ GPT สร้าง



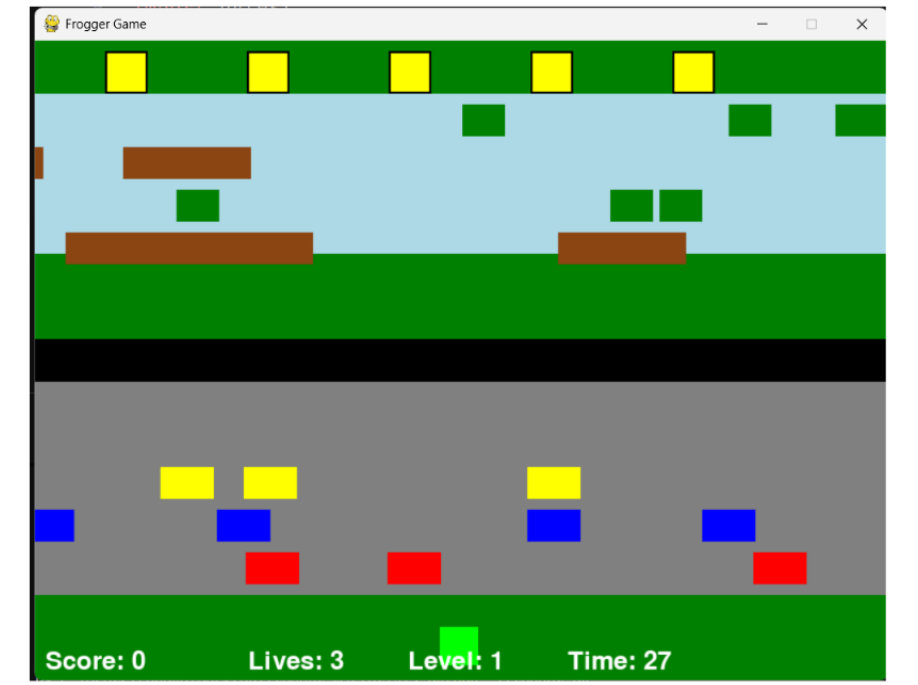
รูปที่ 30 diagram ของ Repo

Frogger ตัวอย่างเกม

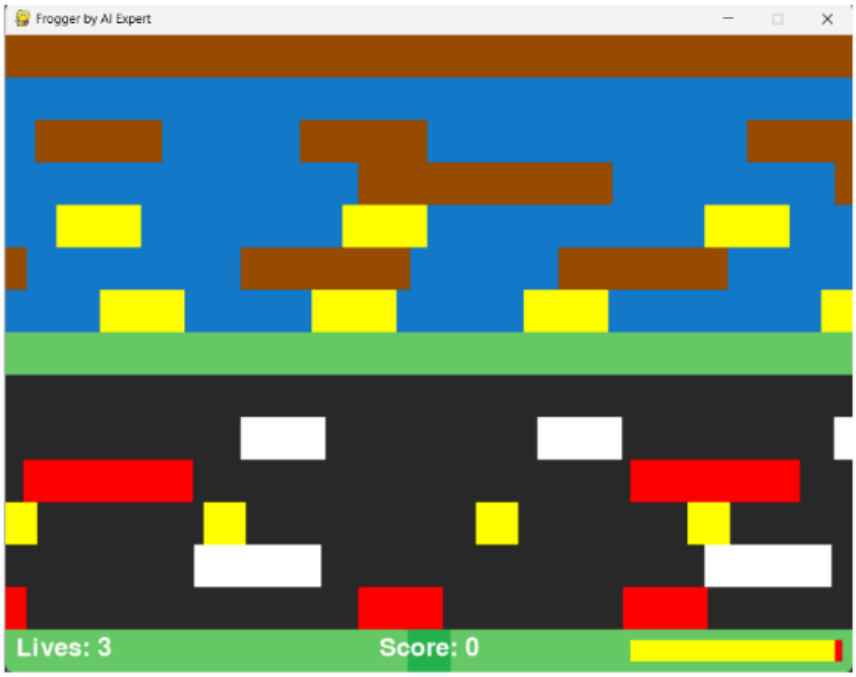
รูปที่ 31 หน้าเกมของ frog game ที่ไม่ใช่ AI สร้าง



รูปที่ 32 หน้าเกมของ frog game ที่ใช้ AI GPT-5 สร้าง



รูปที่ 33 หน้าเกมของ frog game ที่ใช้ AI Claude สร้าง



รูปที่ 34 หน้าเกมของ frog game ที่ใช้ AI Gemini-Pro2.5 สร้าง