Journal of Information Science and Technology

JIST Journal of Information Science and Technology
Volume 15, NO 2 | JUL – DEC 2025 | 65-76
ISSN: 2651-1053 (Online)
ISSN: 1906-9553 (Print)

แนวทางการพัฒนาเกมซอฟต์แวร์อย่างรวดเร็วบนระบบนิเวศแบบเว็บของ Unity

The Guideline For Rapid Software Game Development On Unity Web-based Ecosystem

วาทิต สีมาคุปต์, ภูวดล ศิริกองธรรม

Vatit Simakupt, Puwadol Sirikongtham

si.vatit_st@tni.ac.th, puwadol@tni.ac.th

คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีไทย-ญี่ปุ่น กรุงเทพฯ

Faculty of Information Technology, Thai-Nichi Institute of Technology, Bangkok

Received: September 30, 2025; Revised: December 10, 2025; Accepted: December 15, 2025; Published: December 27, 2025

ABSTRACT – This study proposes a guideline for rapid content-based game development within the Unity web ecosystem, focusing on processes that clearly separate team roles to improve flexibility, usability, and collaboration efficiency. By comparing traditional CMS-based approaches with a web-driven process, the findings show that the proposed method reduces overall development time by multiple of times, enables independent task execution, and ensures stronger protection of the game's core source code. The proposed process thus provides a practical and scalable solution for modern game production, where speed, security, and adaptability are critical.

KEY WORDS -- Game Development, Unity, Web Application, Game Content, Workflow

บทคัดย่อ -- งานวิจัยนี้มีวัตถุประสงค์เพื่อเสนอแนวทางการพัฒนาเกมเชิงเนื้อหาที่รวดเร็วและมีประสิทธิภาพภายใต้ระบบนิเวศแบบเว็บของ Unity โดยเน้นการสร้างกระบวนการทำงานที่แยกบทบาทของทีมงานออกจากกันอย่างชัดเจน เพื่อเพิ่มความยืดหยุ่น ความสะดวก และลดความซับซ้อนในการทำงานร่วมกัน ผลการศึกษาเปรียบเทียบระหว่างระบบ CMS ดั้งเดิมกับกระบวนการทำงานที่ขับเคลื่อนด้วยเว็บ พบว่าระบบใหม่นี้สามารถลดเวลาในการทำงานโดยรวมได้หลายเท่าตัว เพิ่มประสิทธิภาพการทำงานแบบอิสระ และยังช่วยปกป้องซอร์สโค้ดหลักของโปรแกรมได้อย่างมีประสิทธิภาพ แนวทางที่นำเสนอจึงเป็นทางเลือกที่เหมาะสมสำหรับการผลิตเกมสมัยใหม่ที่ต้องการความรวดเร็ว ความปลอดภัย และการขยายตัวในอนาคต

คำสำคัญ -- การพัฒนาเกม, Unity, เว็บแอปพลิเคชัน, เนื้อหาเกม, กระบวนการทำงาน

1. บทนำ

อุตสาหกรรมการพัฒนาเกมในปัจจุบันมีการพัฒนาเปลี่ยนแปลงอย่างรวดเร็วและซับซ้อนขึ้นเรื่อย ๆ โดยเฉพาะในกลุ่มเกมที่เน้นเนื้อหา (content-based games) ซึ่งต้องมีการสร้างและอัปเดตเนื้อหาอย่างต่อเนื่องเพื่อให้ผู้เล่นได้รับประสบการณ์ที่สดใหม่และน่าสนใจ การสร้างเนื้อหาในลักษณะนี้ไม่สามารถชะลอหรือล่าช้าได้ เพราะจะส่งผลโดยตรงต่อความต่อเนื่องของเกมและความพึงพอใจของผู้เล่น

ปัจจุบันเกมขนาดใหญ่ส่วนมากมักถูกออกแบบโดยเลือกใช้ระบบสถาปัตยกรรมแบบเซิร์ฟเวอร์-ไคลเอนต์ (server - client architecture) ซึ่งทำให้เนื้อหาที่ปรับปรุงล่าสุดสามารถส่งไปยังผู้เล่นได้ทันที การพัฒนาเนื้อหาในลักษณะนี้มักมีความท้าทายหลายประการ หนึ่งในนั้นคือการใช้เครื่องมือจัดการและแก้ไขเนื้อหาภายในตัวเกม เช่น Unity Editor แม้ว่าเครื่องมือเหล่านี้จะทรงพลังและมีฟังก์ชันครอบคลุม แต่ก็มีข้อจำกัดหลายประการ เช่น ต้องใช้ทรัพยากรเครื่องสูง ใช้งานยากสำหรับผู้ที่ไม่ชำนาญด้านเทคนิค และไม่ยืดหยุ่นเท่าการสร้างอินเตอร์เฟซบนเว็บ

นอกจากนี้ การให้สิทธิ์เข้าถึงโค้ดต้นฉบับของเกม แก่ทีมศิลปินหรือผู้สร้างเนื้อหาเพียงเพื่อให้เข้าถึงการใช้งาน Unity Editor อาจเป็นความเสี่ยงด้านทรัพย์สินทางปัญญาและความซับซ้อนในการจัดการ ในทางกลับกัน เทคโนโลยีเว็บสมัยใหม่สามารถช่วยแก้ปัญหาเหล่านี้ได้อย่างมีประสิทธิภาพ เว็บแอปพลิเคชันสามารถเข้าถึงได้ง่าย ติดตั้งและปรับใช้งานสะดวก และสามารถสร้างเครื่องมือแก้ไขเนื้อหาให้อิสระออกจากเอนจินหลัก ทำให้โปรแกรมเมอร์และศิลปินสามารถทำงานแยกกันโดยไม่เกิดคอขวดในการพัฒนา

ด้วยความสามารถของ Unity Web-Target [1] ในการสามารถทำงานร่วมกับ JavaScript นักพัฒนาสามารถปรับแต่งควบคุมพฤติกรรมเกมในเวลารันไทม์ได้ ทำให้สามารถทดสอบเนื้อหาใหม่ได้ทันที โดยใช้ข้อมูลทั้งจากเซิร์ฟเวอร์หรือจากภายในเครื่องผู้ใช้เอง ซึ่งยังทำให้ทีมงานสามารถสร้างเนื้อหาภายในสภาพแวดล้อมแยกส่วน (sandbox) ส่งผลให้เป็นการเพิ่มประสิทธิภาพในการทำงานร่วมกัน

จากงานวิจัยที่เกี่ยวข้องพบว่ายังไม่เคยมีแนวทางที่เน้นการสร้าง Workflow ที่ครอบคลุมและยืดหยุ่นสำหรับการพัฒนาเกมเชิงเนื้อหาบน Unity Web-based Ecosystem งานวิจัยนี้มุ่งเน้นการสำรวจและกำหนดแนวทางการพัฒนาระบบสำหรับแก้ไขเนื้อหาแบบเว็บสำหรับเกม Unity โดยศึกษาวิธีการฉีดข้อมูลในเวลารันไทม์ รองรับการทดสอบเนื้อหาแบบแยกส่วน (sandbox) และกำหนดรูปแบบการทำงานที่เหมาะสมสำหรับเกมแต่ละประเภท โดยมีสมมุติฐานคาดหวังให้ได้ผลลัพธ์ในการเพิ่มความรวดเร็วในการทำงาน และเพิ่มประสิทธิภาพการทำงานร่วมกัน อาทิเช่น การลดการใช้เวลา และจำนวนบุคคลกรในการทำงานร่วมกัน เป็นต้น

2. แนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้อง

2.1 Workflow การพัฒนาเกม

การพัฒนาเกมเป็นกระบวนการที่ซับซ้อนและมีหลายขั้นตอน ซึ่งแต่ละขั้นตอนยังต้องอาศัยความร่วมมือระหว่างนักพัฒนาระบบ ศิลปิน และผู้สร้างเนื้อหา เพื่อให้เกมออกมาอย่างสมบูรณ์และน่าสนใจ โดยทั่วไปกระบวนการพัฒนาเกมสามารถแบ่งออกเป็น ขั้นตอนหลัก (Phases) และ พื้นที่การทำงานเฉพาะด้าน (Areas)

2.1.1 ขั้นตอนหลักในการพัฒนาเกม (Phases)

แนวคิด (Concept) - ขั้นตอนนี้เป็นการวางแนวคิดของเกม กำหนดประเภทเกมและรูปแบบการเล่น รวมถึงเป้าหมายของผู้เล่นและเนื้อเรื่องหลัก การวางแนวคิดที่ชัดเจนจะช่วยให้ทีมงานสามารถกำหนดกรอบการพัฒนาและทรัพยากรที่ต้องใช้ได้อย่างแม่นยำ

การออกแบบ (Design) - ในขั้นตอนนี้ นักออกแบบเกมจะสร้างแผนผังระบบเกม (game design document) ระบุโครงสร้างเกม เมคานิกส์ของเกม ตัวละคร และด่านหรือเลเวลต่างๆ รวมถึงการวางโครงสร้างข้อมูลระบบเนื้อหา ที่สามารถเพิ่มหรืออัปเดตได้อย่างต่อเนื่อง

การพัฒนา (Implementation) - ขั้นตอนที่นักพัฒนาโปรแกรมเมอร์จะทำการเขียนโค้ดเกม สร้างระบบและเมคานิกส์ตามแผนงานที่กำหนด และการสร้างเนื้อหามักต้องทำงานร่วมกับศิลปินและผู้สร้างเนื้อหา เพื่อให้ภาพกราฟิก เสียง และข้อมูลในเกมตรงตามแนวคิด

การทดสอบ (Testing) - การทดสอบเกมเป็นขั้นตอนสำคัญเพื่อค้นหาข้อผิดพลาด ตรวจสอบความถูกต้องของระบบ สมดุลของเกม และประเมินประสบการณ์ผู้เล่น

การเผยแพร่ (Publishing) - หลังจากเกมผ่านการทดสอบและปรับปรุงแล้ว เกมจะถูกเผยแพร่สู่ผู้เล่น ซึ่งในกรณีของเกมที่เน้นเนื้อหา การเผยแพร่ต้องรองรับการอัปเดตเพิ่มเติมหรือแก้ไขเนื้อหาอย่างต่อเนื่อง โดยไม่ต้องให้ผู้เล่นดาวน์โหลดซ้ำทั้งหมดเกินความจำเป็น

2.1.2 พื้นที่การทำงานเฉพาะด้าน (Areas)

เมคานิกส์การเล่น (Gameplay Mechanics) – การออกแบบและพัฒนากลไกการเล่น เป็นหัวใจหลักสำคัญของ

การสร้างระบบเกม ซึ่งรวมไปถึง อาทิ การเคลื่อนไหวของตัวละคร ระบบต่อสู้ ระบบภารกิจ และการตอบสนองของเกมต่อผู้เล่น

การออกแบบเลเวลและเนื้อหา (Level and Content Design) – การสร้างด่านเลเวลและเนื้อหา และการเพิ่มความท้าทายใหม่ให้แก่ผู้เล่น ส่วนนี้มีความเกี่ยวข้องกับระบบเกมแต่ละระบบสูง ตามลักษณะเฉพาะตัว จึงมักจำเป็นต้องมีการพัฒนาเครื่องมือแก้ไขข้อมูลเฉพาะของระบบนั้นๆ

การแก้ไขเนื้อหารอง (Secondary Content Editing) - การปรับปรุงหรือแก้ไขเนื้อหาที่ไม่ใช่ระบบหลัก เช่น ข้อความ แอนิเมชัน เสียงประกอบ หรือเนื้อหาเนื้อเรื่อง ในส่วนนี้หากระบบออกแบบมาดี จะทำให้นักออกแบบสามารถทำงานอิสระได้โดยไม่ต้องคอยพึ่งพาความช่วยเหลือจากโปรแกรมเมอร์

2.2 เอนจินเกม (Game Engine)

เอนจินเกมคือซอฟต์แวร์หลักที่ช่วยนักพัฒนาในการสร้างเกม โดยให้เครื่องมือและโครงสร้างพื้นฐานสำหรับการเขียนโค้ด การจัดการกราฟิก เสียง ฟิสิกส์ และการโต้ตอบของผู้เล่น [2] ซึ่งแตกต่างจากไลบรารีมัลติมีเดียขั้นพื้นฐาน (primitive multimedia libraries) ที่มักให้เฉพาะฟังก์ชันพื้นฐาน เช่น การวาดภาพหรือเล่นเสียง แต่ไม่มีเครื่องมือครบวงจรสำหรับการพัฒนาเกม

2.2.1 ความแตกต่างระหว่าง Game Engine กับ Multimedia Libraries

ความครบวงจร: เอนจินเกมจะรวมฟีเจอร์หลายด้าน [3] เช่น ระบบฟิสิกส์ AI การจัดการเนื้อหา และระบบแอนิเมชัน ในขณะที่ไลบรารีมัลติมีเดียมักเน้นเฉพาะงานเดียว

เครื่องมือสร้างเนื้อหา: เอนจินเกมมักมาพร้อมกับ editor ที่สามารถสร้างและแก้ไขเนื้อหาได้โดยตรง ในขณะที่ไลบรารีพื้นฐานจะต้องเขียนระบบเองทั้งหมด

2.2.2 เอนจินเกมหลักในตลาด

Unity – เป็นเอนจินเกมที่ได้รับความนิยมสูง [4] เนื่องจากรองรับหลายแพลตฟอร์ม มีเครื่องมือครบครันสำหรับทั้งเกม 2D และ 3D และสนับสนุนการทำงานร่วมกับ Web Target ทำให้สามารถรันเกมบนเว็บได้อย่างยืดหยุ่น

Unreal Engine – มีความสามารถด้านกราฟิกและฟิสิกส์ขั้นสูง เหมาะสำหรับเกม AAA และเกมที่ต้องการภาพกราฟิกสมจริงสูง

Godot – เป็นเอนจินเกมโอเพนซอร์สที่เหมาะกับเกมขนาดเล็กถึงกลาง มีภาษา scripting ที่เรียนรู้ง่าย และสามารถปรับแต่ง UI/UX สำหรับการสร้างเครื่องมือแก้ไขเนื้อหาได้

2.3 เว็บแอปพลิเคชัน (Web Application)

เว็บแอปพลิเคชันเป็นเครื่องมือสำคัญในการสร้างอินเตอร์เฟซที่ใช้งานง่ายและเข้าถึงได้สะดวกสำหรับผู้สร้างเนื้อหาและทีมงานศิลปิน เทียบกับเครื่องมือแก้ไขเนื้อหาแบบดั้งเดิมบนเอนจินเกม เว็บแอปสามารถให้ประสบการณ์การใช้งานที่ยืดหยุ่น ปลอดภัย และสามารถปรับแต่งได้ตามความต้องการของทีม

2.3.1 ระบบ UI/UX ที่ซับซ้อน (Complex UI/UX Ecosystem)

เว็บแอปพลิเคชันสามารถสร้างอินเตอร์เฟซการทำงานที่มีความซับซ้อนและปรับแต่งได้ [5] ตามแต่ละประเภทเกม เช่น สามารถสร้างตัวแก้ไขด่านเลเวลที่ซับซ้อน ระบบจัดการคลังทรัพยากร หรือเครื่องมือแก้ไขบทสนทนาและภารกิจ

2.3.2 การเข้าถึงและปรับใช้ได้ง่าย

เว็บแอปสามารถเข้าถึงได้จากอุปกรณ์หลายประเภท เช่น คอมพิวเตอร์ตั้งโต๊ะ แล็ปท็อป หรือแม้กระทั่งแท็บเล็ต ทำให้ทีมงานไม่จำเป็นต้องติดตั้งซอฟต์แวร์หนัก ๆ บนเครื่องตนเอง นอกจากนี้ การปรับใช้และอัปเดตเครื่องมือแก้ไขเนื้อหาเป็นเรื่องง่ายเพียงอัปเดตจากฝั่งเซิร์ฟเวอร์ ขจัดความยุ่งยากในการติดตั้งซ้ำเมื่อมีการแก้ไขระบบ

2.3.3 การรวมเข้ากับเทคโนโลยีเว็บสมัยใหม่

เว็บแอปสามารถทำงานร่วมกับเทคโนโลยีเว็บล่าสุด เช่น JavaScript, WebAssembly, HTTP API, และระบบฐานข้อมูล ทำให้สามารถดึงข้อมูลจากเซิร์ฟเวอร์หลักหรือทดสอบเนื้อหาแบบแยกส่วนได้ทันที เทคนิคเช่นการฉีดข้อมูล (data injection) และการโอเวอร์ไลด์ข้อมูล (overriding) ช่วยให้ทีมงานสร้างและทดสอบเนื้อหาใหม่โดยไม่กระทบต่อเซิร์ฟเวอร์หลัก

2.4 ความสามารถในการทำงานร่วมกันของภาษาโปรแกรม (Programming Language Interoperability)

การทำงานร่วมกันระหว่างภาษาโปรแกรม [6] เป็นหัวใจสำคัญในการพัฒนาเนื้อหาเกมบนหน้าเว็บ ซึ่งต้องให้เอนจินเกมและเว็บแอปสามารถแลกเปลี่ยนข้อมูลและเรียกใช้งานฟังก์ชันซึ่งกันและกันได้อย่างราบรื่น เทคนิคเหล่านี้ช่วยให้ทีมโปรแกรมเมอร์และผู้สร้างเนื้อหาสามารถทำงานแยกส่วนกันได้อย่างอิสระ โดยไม่กระทบต่อการทำงานของฝ่ายระบบ

2.4.1 ABI – Application Binary Interface

ABI เป็นมาตรฐานที่กำหนดวิธีการที่โปรแกรมเรียกใช้ฟังก์ชันและเข้าถึงหน่วยความจำของซอฟต์แวร์ในระดับไบนารี การทำงานร่วมกันบนมาตราฐาน ABI ช่วยให้โมดูลที่เขียนด้วยภาษาโปรแกรมต่างกันสามารถติดต่อสื่อสารกันได้

2.4.2 API – Application Programming Interface

API เป็นชุดของคำสั่งและฟังก์ชันที่จัดทำขึ้นเฉพาะกิจเพื่อเปิดทางให้โปรแกรมหนึ่งสามารถเรียกใช้งานอีกโปรแกรมหนึ่งได้ โดย API มีหลากหลายรูปแบบ อาทิ การทำงานผ่านเครือข่ายอินเตอร์เน็ต เป็นต้น

2.4.3 FFI – Foreign Function Interface

FFI เป็นเทคนิคที่ทำให้โปรแกรมหนึ่งสามารถเรียกใช้ฟังก์ชันหรือโมดูลที่เขียนด้วยภาษาต่างกัน ตัวอย่างเช่น เว็บแอปที่เขียนด้วย JavaScript สามารถเรียกใช้งานฟังก์ชัน C# ของ Unity Web build ได้ทันที FFI จึงเป็นหัวใจหลักในการช่วยเพิ่มความยืดหยุ่นในการสร้างระบบ

2.5 JavaScript และ ECMAScript

JavaScript เป็นภาษาสคริปต์หลักของเว็บแอปพลิเคชัน มีบทบาทสำคัญในการสร้างอินเตอร์เฟซแบบโต้ตอบ และการเชื่อมต่อกับเอนจินเกมผ่านเว็บ ของ Unity Web Target

2.5.1 JavaScript

JavaScript เป็นภาษาที่รันได้บนเว็บเบราว์เซอร์โดยตรง [7] ทำให้สามารถสร้าง UI/UX ที่ซับซ้อน และทำงานร่วมกับ Web API ของเบราว์เซอร์ได้ เช่น การกรอกข้อมูล, การดึงข้อมูลจากไฟล์ หรือจากเซิร์ฟเวอร์, การส่งค่าไปยังเอนจินเกม Unity ผ่านฟังก์ชัน Interop หรือ HTTP Request/Response

2.5.2 ECMAScript

ECMAScript คือชื่อและการระบุมาตรฐานย่อย อย่างเป็นทางการ [8] ของภาษา JavaScript ที่กำหนดโครงสร้างและฟีเจอร์ของภาษา เช่น การใช้งานโมดูล, ฟังก์ชันแบบอะซิงโครนัส (async/await), และตัวแปรแบบ block scope (let/const) มาตรฐานนี้ช่วยให้การพัฒนาเว็บแอปเป็นไปอย่างมีระเบียบ และสามารถปรับปรุงให้รองรับฟีเจอร์ใหม่ ๆ ของเบราว์เซอร์ได้อย่างต่อเนื่อง

2.6 WebAssembly

WebAssembly (Wasm) เป็นเทคโนโลยีที่ช่วยให้โปรแกรมที่คอมไพล์จากภาษาโปรแกรมมิ่งระดับสูง เช่น C# (Unity) สามารถรันบนเว็บเบราว์เซอร์ได้อย่างมีประสิทธิภาพใกล้เคียงกับโปรแกรมเนทีฟ [9]

ประสิทธิภาพสูง: รันโค้ดได้ใกล้เคียงกับโค้ดเนทีฟ ลดความหน่วงเวลาในการประมวลผล

ความปลอดภัย: รันใน sandbox ของเบราว์เซอร์ จึงไม่สามารถเข้าถึงระบบไฟล์หรือทรัพยากรภายนอกโดยตรง

ความยืดหยุ่น: สามารถทำงานร่วมกับ JavaScript เพื่อเรียกฟังก์ชันและส่งค่ากลับไปมาแบบเรียลไทม์

2.7 WebGL และ WebGPU

WebGL และ WebGPU เป็นเทคโนโลยีกราฟิกบนเว็บที่ช่วยให้ Unity Web Build สามารถเรนเดอร์กราฟิก 2D/3D ได้อย่างมีประสิทธิภาพภายในเบราว์เซอร์ [10] แม้จะรันในสภาพแวดล้อมของเว็บ ซึ่งช่วยให้ผู้สร้างเนื้อหาและทีมศิลปินสามารถเห็นผลลัพธ์ของงานศิลปะและการออกแบบเลเวลได้ทันทีแบบเรียลไทม์

2.7.1 WebGL

มาตรฐานกราฟิกที่อนุญาตให้รันโค้ดกราฟิก 3D/2D ได้ประสิทธิภาพใกล้เคียงกับระบบเนทีฟ อาทิเช่น OpenGL บนเว็ปเบราว์เซอร์ [11] โดยไม่ต้องติดตั้งปลั๊กอินเพิ่มเติม เป็นเทคโนโลยีเก่ารุ่นแรกของการทำระบบกราฟฟิกประสิทธิภาพสูงบนระบบเว็ปเบราว์เซอร์

2.7.2 WebGPU

เทคโนโลยีกราฟิกรุ่นใหม่ที่ให้ประสิทธิภาพสูงกว่า WebGL [12] และสนับสนุนฟีเจอร์กราฟิกสมัยใหม่ เช่น การเรนเดอร์แบบ compute shader และการประมวลผลกราฟิกแบบ

ขนาน ทำให้สามารถแสดงผลได้อย่างรวดเร็วและมีรายละเอียดสูงยิ่งกว่า [13]

2.8 งานวิจัยที่เกี่ยวข้อง

Creating Audio Games Online with a Browser-Based Editor [14]: งานวิจัยนี้มุ่งเน้นการพัฒนาเครื่องมือแก้ไข (Editor) ที่ทำงานผ่านเว็บเบราว์เซอร์สำหรับสร้างเนื้อหาเกมที่มีการเล่นผ่านเสียงเป็นหลัก ซึ่งแตกต่างจากงานงานวิจัยของเราที่เน้นการพัฒนากระบวนการทำงาน (workflow) ที่สามารถประยุกต์ใช้ได้กับหลายประเภทเกม งานวิจัยดังกล่าวมีขอบเขตจำกัดเฉพาะเกมประเภทเดียวและไม่อาศัยเอนจินเกมมาตรฐานอย่าง Unity หรือ Unreal จึงมีความยืดหยุ่นน้อยกว่าแนวทางที่เสนอในงานวิจัยของเราซึ่งสามารถครอบคลุมการสร้างเนื้อหาหลากหลายประเภทได้กว้างกว่า

Generative Audio and Real-Time Soundtrack Synthesis in Gaming Environments [15]: งานนี้นำเสนอ workflowที่ออกแบบมาเพื่อการสร้างเนื้อหาด้านเสียงและดนตรีภายในเกม โดยเฉพาะเกมแนวดนตรีหรือจังหวะ (Rhythm) ซึ่งมีขอบเขตจำกัดอยู่ในมิติของการออกแบบเสียงดนตรีเท่านั้น และไม่ได้ใช้เครื่องมือบนเว็บ แม้ว่าจะช่วยเพิ่มความเข้าใจเกี่ยวกับ workflow เฉพาะทาง แต่ยังไม่ตอบโจทย์การสร้างเนื้อหาหลากหลายชนิดเชิงกว้างและการร่วมมือผ่านแพลตฟอร์มเว็บอย่างที่งานวิจัยของเรามุ่งเน้น

Using Unity and Shield UI for Displaying 3D Medical Education Objects in the Web Browser [16]: งานนี้ประยุกต์ใช้ความสามารถของ Unity และ WebGL เพื่อแสดงผลข้อมูลสามมิติ สำหรับการศึกษาในด้านการแพทย์ โดยแสดงให้เห็นศักยภาพของเทคโนโลยีเว็บในการจัดการข้อมูลที่ซับซ้อน อย่างไรก็ตาม งานดังกล่าวไม่ได้เน้นที่การพัฒนาเกมหรือการสร้าง workflow ในการจัดการเนื้อหา แต่เป็นการใช้ WebGL ในมิติอื่น ซึ่งแม้จะสะท้อนถึงความยืดหยุ่นของเทคโนโลยีเว็บ แต่ยังไม่สามารถนำมาใช้โดยตรงในบริบทของการพัฒนาเกมซึ่งเป็นหัวใจหลักของงานวิจัยของเรา

ตาราง 2.8: การเปรียบเทียบงานวิจัยที่เกี่ยวข้อง

งานวิจัย	เกมเอนจิน	Editor ภายนอก	ประเภทเนื้อหา
Creating Audio Games Online with a Browser-Based Editor [14]	ไม่ได้ใช้	Web-Based Editor	เกมดนตรี
Generative Audio and Real-Time Soundtrack Synthesis in Gaming Environments [15]	Unity	DAW (โปรแกรมทำเพลง)	เกมดนตรี
Using Unity and Shield UI for Displaying 3D Medical Education Objects in the Web Browser [16]	Unity	Web-Based Editor	แอพลิเคชั่นทางการแพทย์
แนวทางของงานวิจัยนี้	Unity	Web-Based Editor	เกมหลายหลายประเภท

จากตาราง 2.8 จะเห็นได้ว่า งานวิจัยนี้มีความแตกต่างจากงานวิจัยอื่น ๆ โดยมุ่งเน้นการพัฒนา workflow ที่รองรับการทำงานภายในระบบ Unity Engine ซึ่งเป็นเครื่องมือหลักในการพัฒนาเกมดิจิทัลที่ได้รับความนิยมในระดับสากล โดยลักษณะเด่นของงานวิจัยคือการนำเสนอแนวทางการสร้างและปรับปรุงเนื้อหาเกมผ่าน Editor แบบ Web-based ซึ่งช่วยให้ผู้ใช้สามารถเข้าถึงและจัดการองค์ประกอบของเกมได้จากทุกที่โดยไม่จำเป็นต้องพึ่งพาการติดตั้งซอฟต์แวร์เฉพาะทางเพิ่มเติม นอกจากนี้ งานวิจัยยังให้ความสำคัญกับการออกแบบกระบวนการสร้างเนื้อหาที่มีความยืดหยุ่นสูง สามารถนำไปประยุกต์ใช้ได้กับเกมหลากหลายประเภท ทั้งนี้ เพื่อขยายขอบเขตความเป็นไปได้ในการพัฒนาเกม และอำนวยความสะดวกต่อทั้งนักพัฒนาและผู้ที่ไม่มีพื้นฐานทางเทคนิคมากนัก

3. วิธีการดำเนินการวิจัย

3.1 การกำหนดปัจจัยประเมินผล

เพื่อวิเคราะห์และประเมินประสิทธิภาพของระบบ จำเป็นต้องกำหนดปัจจัยที่ชัดเจนและสามารถวัดผลได้ โดยปัจจัยหลักแบ่งออกเป็นสองด้านคือ ความสะดวกในการใช้งาน (Ease of Use) และ ประสิทธิภาพในการทำงานร่วมกัน (Collaboration Efficiency)

3.1.1 ความสะดวกในการใช้งาน (Ease of Use)

ปัจจัยนี้มุ่งเน้นการประเมินคุณสมบัติของระบบที่ช่วยให้ผู้ใช้เข้าถึงและจัดการข้อมูลหรือเนื้อหาเกมได้อย่างง่ายดาย รวดเร็ว และปลอดภัย โดยมีเกณฑ์ย่อยดังนี้

การเข้าถึง (Accessibility): การเปรียบเทียบระหว่างรูปแบบการใช้งานที่ต้องพึ่งพา Unity Editor หรือการติดตั้งบนอุปกรณ์ (mobile/desktop deploy) กับการเข้าถึงผ่านเว็บเบราว์เซอร์ที่สะดวกและไม่ต้องติดตั้งเพิ่มเติม

ความสามารถในการปรับแต่ง (Customization): การรองรับความต้องการเฉพาะของแต่ละเกมหรือเนื้อหา ซึ่งระบบแบบเว็บสามารถปรับ UI และ Workflow ให้เหมาะกับทีมงานได้ดีกว่าการปรับแต่งภายใน Unity Editor

การทำงานแบบแยกอิสระ (Isolation): การรองรับให้ผู้พัฒนาหรือผู้แก้ไขเนื้อหาสามารถสร้างและทดสอบข้อมูลได้บนเครื่องของตนเอง โดยไม่รบกวนระบบเซิร์ฟเวอร์หลักหรือทีมงานอื่น

ประสิทธิภาพการถ่ายโอนข้อมูล (Data Transfer): การลดการพึ่งพาการส่งข้อมูลปริมาณมากจากระยะไกล (remote server) โดยเปิดโอกาสให้ผู้ใช้สามารถทดสอบด้วยข้อมูลเฉพาะส่วนที่ต้องแก้ไขได้ในเครื่องตนเอง

การปกป้องซอร์สโค้ด (Source Code Protection): การควบคุมสิทธิ์ในการเข้าถึง โดยไม่จำเป็นต้องให้ทีมศิลปินหรือ Content Editor เข้าถึง Unity Project ตรง ๆ ซึ่งช่วยลดความเสี่ยงด้านความซับซ้อนและการรั่วไหลของทรัพย์สินทางปัญญา

3.1.2 ประสิทธิภาพในการทำงานร่วมกัน (Collaboration Efficiency)

ปัจจัยด้านนี้พิจารณาความสามารถของระบบในการลดความซ้ำซ้อนและเพิ่มความคล่องตัวของทีมงานที่ประกอบด้วยหลายบทบาท โดยเกณฑ์ย่อยประกอบด้วย

เวลาเฉลี่ยในการทำงาน (Average Task Duration): การเปรียบเทียบเวลาที่ใช้ในการทำงานแต่ละประเภท (เช่น การทดสอบไฟล์กราฟิก การสร้างเนื้อหาบทสนทนา หรือการออกแบบด่าน) ระหว่างกระบวนการแบบดั้งเดิมกับกระบวนการใหม่ที่ใช้เว็บเป็นเครื่องมือ

น้ำหนักความสำคัญต่อประเภทเกม (Task Weight per Game Genre): การประเมินว่าประเภทของเกมแต่ละประเภท (Casual, Strategy, Rhythm, MMO) มีการพึ่งพากิจกรรมย่อยมากน้อยเพียงใด โดยให้ค่าความสำคัญในระดับ 1–5 เพื่อสะท้อนถึงลักษณะเฉพาะของแต่ละประเภทเกม

อัตราการพัฒนาเชิงปริมาณ (Improvement Ratio): การวัดสัดส่วนของการประหยัดเวลาและแรงงาน (time × manpower) ที่ได้จากกระบวนการใหม่ เทียบกับกระบวนการดั้งเดิมในแต่ละประเภทเกม

3.2 การกำหนดประเภทเกมเป้าหมาย

ในการออกแบบกระบวนการทำงาน และพัฒนาระบบจำเป็นต้องพิจารณาลักษณะเฉพาะของเกมแต่ละประเภทโดยเฉพาะ ความแตกต่างด้านสถาปัตยกรรมระบบ ข้อมูลเกม และวิธีการจัดการเนื้อหา ซึ่งจะส่งผลโดยตรงต่อประสิทธิภาพของกระบวนการทำงานที่นำเสนอ

3.2.1 เกม Casual/Puzzle

เกมประเภทนี้มักมี สถาปัตยกรรมระบบแบบ Client-Authoritative โดยเน้นการประมวลผลหลักอยู่ที่เครื่องผู้เล่น ข้อมูลเกมประกอบด้วย ด่าน/ระดับ (Level Data) และ การตั้งค่าเกมเพลย์พื้นฐาน (Gameplay Configs) ซึ่งขนาดข้อมูลไม่ใหญ่มาก การจัดการเนื้อหาจึงสามารถทำได้ด้วยไฟล์ JSON หรือฐานข้อมูลขนาดเล็ก ระบบต้องสามารถให้ผู้สร้างเนื้อหาทำงานแบบแยกเครื่อง (local sandbox) ได้สะดวก

3.2.2 เกม Turn-based Strategy

เกมประเภทนี้มี สถาปัตยกรรมแบบ Hybrid Client-Server เนื่องจากต้องตรวจสอบผลลัพธ์ของผู้เล่นหลายคนและรักษาสถานะเกมที่ถูกต้อง ข้อมูลเกมมีทั้ง Master Data สำหรับ

ไอเทม, Content Data สำหรับด่านและแผนที่ ระบบเนื้อหาจำเป็นต้องรองรับการแก้ไขและทดสอบแบบ isolated ก่อน commit กลับไปยังเซิร์ฟเวอร์กลาง

3.2.3 เกม Music/Rhythm

เกมแนวนี้มี สถาปัตยกรรม Client-Authoritative เช่นเดียวกับ Casual แต่เน้น Content ที่มีความละเอียดสูง เช่น เสียง เพลง และกราฟิกสอดคล้องกับจังหวะ การจัดการเนื้อหาและ asset ต้องสามารถปรับแต่งและตรวจสอบผลลัพธ์แบบ real-time ได้

3.2.4 เกม Real-time MMO

เกม MMO มี สถาปัตยกรรม Server-Authoritative ซึ่งเกมเพลย์หลักทั้งหมดขึ้นกับเซิร์ฟเวอร์ เนื่องจากข้อมูลมีปริมาณมากและซับซ้อน การสร้างและทดสอบเนื้อหาใหม่ต้องสามารถทำได้แบบ isolated โดยใช้ local sandbox และมีวิธี inject ข้อมูลชั่วคราวจากเซิร์ฟเวอร์เพื่อดูผลลัพธ์แบบทันที

3.3 วิเคราะห์สถาปัตยกรรมระบบเกมที่พบบ่อย

การวิเคราะห์สถาปัตยกรรมระบบเกมเป็นขั้นตอนสำคัญในการออกแบบ workflow และ framework เพราะสถาปัตยกรรมของเกมแต่ละประเภทมีความแตกต่างทั้งในด้านการจัดการข้อมูล การสื่อสารระหว่างเซิร์ฟเวอร์และไคลเอนต์ และการประมวลผลสถานะเกม

3.3.1 การควบคุมการดำเนินเกม (Server vs Client Gameplay Authoritative)

เกมหลายประเภท โดยเฉพาะเกมออนไลน์แบบ MMO หรือเกมวางแผนแบบ Turn-based มักใช้สถาปัตยกรรมที่เซิร์ฟเวอร์เป็นตัวกำหนดกฎการเล่น (Gameplay Authoritative) โดยไคลเอนต์ทำหน้าที่เป็นเพียงตัวแสดงผลและรับอินพุตจากผู้เล่น:

ข้อดี: ลดความเสี่ยงการโกงเกม, ควบคุมสถานะเกมได้เป็นศูนย์กลาง

ข้อจำกัด: การปรับแก้ไขระหว่างการทดลองทำได้ยากกว่า เนื่องจากผู้สร้างเนื้อหาต้องสามารถทดสอบข้อมูลใหม่โดยไม่กระทบเซิร์ฟเวอร์หลัก

3.3.2 ส่วนประกอบและรูปแบบข้อมูลของเกม (Game Data Components and Patterns)

ในการพัฒนาเกมโดยเฉพาะเกมที่มีเนื้อหาจำนวนมาก การจัดหมวดหมู่ข้อมูลและกำหนดรูปแบบการจัดเก็บข้อมูลเป็นสิ่งจำเป็นอย่างยิ่ง เนื่องจากเกมแต่ละประเภทมักมีความซับซ้อนของข้อมูลที่แตกต่างกัน การระบุส่วนประกอบข้อมูลที่สำคัญและกำหนด pattern ในการจัดการ ทำให้สามารถออกแบบระบบให้รองรับทั้งการสร้าง และการทดสอบเนื้อหาได้อย่างมีประสิทธิภาพ การแบ่งหมวดหมู่และรูปแบบการจัดเก็บทำให้ระบบสามารถปรับตัวต่อแนวทางการสร้างเกมที่แตกต่างกันในแต่ละ genre ได้ เช่น เกม casual อาจเน้นการจัดการเนื้อหาแบบง่ายและ dynamic ในขณะที่เกม MMO หรือ strategy อาจต้องรองรับข้อมูลผู้เล่นและสถานะเกมแบบเรียลไทม์ การวิเคราะห์และกำหนด components และ pattern จึงเป็นพื้นฐานสำคัญที่จะช่วยให้การพัฒนาสอดคล้องกับเป้าหมายของการสร้างเกมเชิงเนื้อหาอย่างมีประสิทธิภาพ

3.4 การพัฒนา Framework และ Workflow

หลังจากวิเคราะห์สถาปัตยกรรมระบบเกมและประเภทเกมเป้าหมายแล้ว ขั้นตอนถัดไปคือการพัฒนา framework และ workflow ที่สามารถทำงานร่วมกับ Unity Web build ได้อย่างมีประสิทธิภาพ โดยมีเทคนิคที่นำมาให้หลักๆดังนี้

3.4.1 Native Function Call

การเรียกใช้งานฟังก์ชันพื้นฐานของ Unity ผ่าน Web browser ทำให้สามารถสร้าง workflow ที่ผู้สร้างเนื้อหาสามารถทดสอบและปรับแต่งผลลัพธ์ของเนื้อหาได้ทันที เช่น การโหลด asset, การอัปเดตสถานะเกม, หรือการควบคุมการเล่น สามารถเรียกใช้งานจาก web UI ได้โดยตรง ช่วยให้ผู้สร้างเนื้อหาทดสอบการทำงานของ asset และข้อมูลแบบเรียลไทม์โดยไม่ต้องรอการ build ใหม่จาก Unity Editor

3.4.2 Web Request Injection

เทคนิคการฉีดข้อมูลผ่าน HTTP request หรือ web form ช่วยให้สามารถปรับแต่งข้อมูลเกมและ assets ใน runtime ได้

ผู้สร้างเนื้อหาสามารถส่งข้อมูลใหม่ เช่น ด่าน, ไอเท็ม, หรือเนื้อเรื่อง เข้าไปในเกมโดยตรงจาก web UI โดยไม่ต้องเปลี่ยนแปลงระบบการสื่อสารของตัวเกม

3.5 การประเมินผลและวิเคราะห์ผลลัพธ์

หลังจากพัฒนา framework และ workflow สำหรับการสร้างและแก้ไขเนื้อหาแบบเว็บ การประเมินผลและวิเคราะห์ผลลัพธ์ถือเป็นขั้นตอนสำคัญ เพื่อพิสูจน์ว่าแนวทางนี้สามารถตอบสนองต่อความต้องการของทีมงานและประเภทเกมต่างๆ ได้อย่างมีประสิทธิภาพ

4. ผลการดำเนินการ

4.1 ผลการวิเคราะห์สถาปัตยกรรมข้อมูลเกมทั่วไป

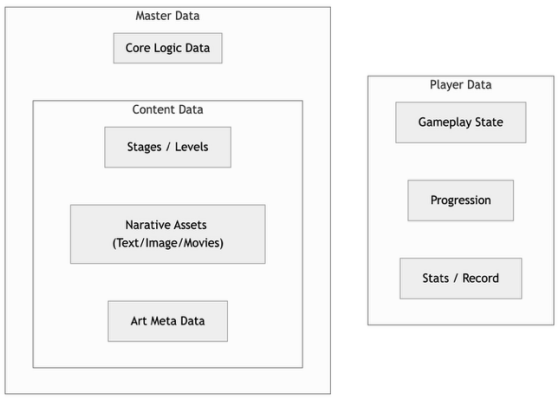
จากการศึกษาและสำรวจเกมแต่ละประเภท พบว่าโครงสร้างข้อมูลและสถาปัตยกรรมระบบมีลักษณะเฉพาะที่แตกต่างกันชัดเจน ดังนี้

Master Data / Meta Data: ข้อมูลหลักของเกม เช่น หน่วยตัวละคร ไอเทม ความสามารถ และค่าพารามิเตอร์พื้นฐาน มักถูกเก็บแยกออกจากเนื้อหาเฉพาะด่านหรือระดับ

Content Data (Stages / Levels): ข้อมูลด่าน/ระดับมีความหลากหลายและขึ้นกับประเภทเกม สำหรับเกม Puzzle จะมีรูปแบบเรียบง่าย ขณะที่เกม Strategy หรือ MMO จะมีข้อมูลซับซ้อน เช่น แผนที่, อุปกรณ์, และภารกิจที่ต้องจัดการร่วมกับผู้เล่นหลายคน

Player Progression Data: ข้อมูลความคืบหน้าของผู้เล่น รวมถึงคะแนน, ไอเทมที่สะสม, และสถานะของเนื้อหาแต่ละด่าน การจัดการข้อมูลประเภทนี้จะมีปริมาณเพิ่มขึ้นตามจำนวนผู้เล่น

Gameplay State / Snapshot: ข้อมูลสถานะเกมปัจจุบัน เช่น ตำแหน่งตัวละคร, เหตุการณ์ที่เกิดขึ้น, และผลลัพธ์ของการกระทำของผู้เล่น มักจำเป็นต้องจัดเก็บแยกต่างหาก เพื่อเพิ่มความรวดเร็วของการประมวลผล และเพื่อรองรับการทดสอบและแก้ไขทันทีในทุกสถานะ



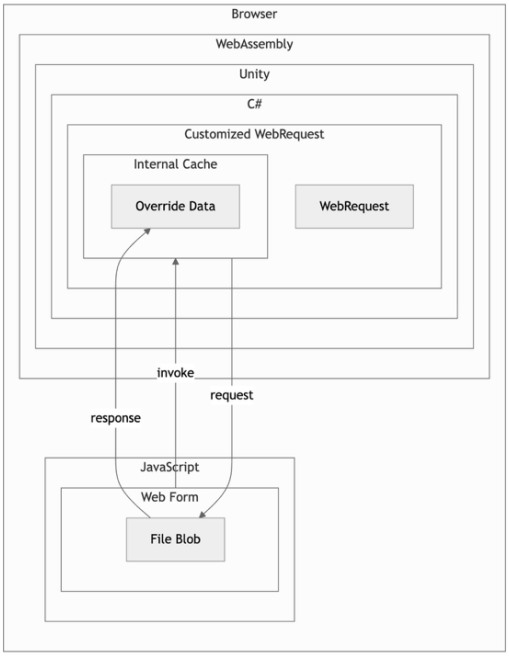
รูปที่ 1. สถาปัตยกรรมข้อมูลเกม

4.2 ผลการพัฒนาระบบและกระบวนการทำงาน

จากข้อกำหนดและการวิเคราะห์สถาปัตยกรรมข้อมูล เราได้ทำการสร้างระบบที่มีส่วนประกอบทั้ง 2 ฝั่ง (C# กับ JavaScript) ทำงานร่วมกัน โดยหัวใจหลักคือการสร้าง Custom Wrapper Class ของ Unity WebRequest ทำให้สามารถควบคุมการทำงานต่างๆที่ตัวเกมกระทำผ่าน HTTP Request ได้อย่างหลากหลาย โดยประกอบไปด้วยความสามารถต่างๆ อาทิเช่น

CDN Asset Override – เป็นการส่งคำสั่งจาก JavaScript ไปยัง Unity ให้ทำการอ่านไฟล์ผ่าน Blob URL เพื่อนำเนื้อหาของไฟล์นั้นๆ ผ่านกระบวนการแปลง Format ข้อมูลของ Unity WebRequest Module ก่อนแล้วจึงเก็บลงใน Internal Cache ไว้ล่วงหน้า โดยอิงกับ URL นั้นๆ ก่อนที่ URL นั้นจะถูกเรียกอ่านจริงในช่วงเวลารันตัวเกม

HTTP Request Injection – เป็นการให้ JavaScript ทำการสร้างข้อมูล JSON API Request/Response ขึ้น ตามที่ปรับใน Web UI เพื่อส่งเนื้อหานั้นไปเก็บใน Internal Cache เช่นเดียวกับการ Override Asset แต่สามารถทำได้ทั้งข้อมูลขาเข้า Response และขาออก Request เพื่อปรับแต่งการสื่อสารของไคลแอนท์กับฝั่งเซิร์ฟเวอร์



รูปที่ 2. สถาปัตยกรรมระบบที่ได้พัฒนาขึ้น

```
// Assume UnityInstance is your WebGL build instance
function injectContent(jsonData) {
    // Convert JSON object to string
    const jsonString = JSON.stringify(jsonData);

    // Send the content to Unity runtime via SendMessage
    // GameObject: "WebContentManager", Method: "ApplyContent", Param:
jsonString
    UnityInstance.SendMessage('WebContentManager', 'ApplyContent',
jsonString);
}

// Example usage
const newContent = {
    stageName: "Level_01",
    dialogues: ["Welcome!", "Collect all items to proceed."],
    assetUrls: ["assets/character.png", "assets/item.png"]
};

// Trigger content injection
injectContent(newContent);
```

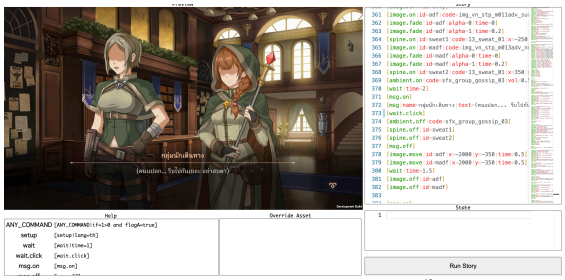
รูปที่ 3. ตัวอย่างโค้ดของระบบที่ได้พัฒนาขึ้น

ระบบที่พัฒนาขึ้นสามารถรองรับรูปแบบการทำงาน อาทิเช่น

การทดสอบ Art Assets: ศิลปินสามารถโหลดและทดสอบไฟล์กราฟิกหรือโมเดล 3 มิติบนเครื่อง local ได้ทันที โดย สามารถปรับการ Override ค่า Master/Meta Data ได้โดยตรงบน Web UI

การสร้างเนื้อหาเรื่องราว (Narrative Content): นักออกแบบสามารถสร้างเนื้อหาลำดับบทสนทนา และเนื้อหาเชิงเรื่องราว พร้อมทั้งดูผลลัพธ์แบบ Realtime บน Local Sandbox

การสร้างด่าน/ระดับ (Stages/Levels): ระบบรองรับการแก้ไขข้อมูลด่านพร้อมการ Override ข้อมูลในฝั่งเซิร์ฟเวอร์เพื่อดูผลลัพธ์แบบทันที โดยหากเป็นข้อมูลที่ต้องคำนวนในฝั่ง Server ควรออกแบบระบบให้ทำงานกับ State Snapshot เพื่อความสะดวกในการ Override ข้อมูล



รูปที่ 4. ตัวอย่าง UI ของระบบที่ได้พัฒนาขึ้น ประกอบด้วย Unity Viewport Canvas, HTML Form สำหรับ Text Input, File Upload, Input Button และรวมไปถึง Monaco Editor สำหรับ Syntax Highlighting & Checking

เมื่อเปรียบเทียบกับระบบภายในดั้งเดิมของ Unity เช่น Addressables / Scriptable Object ระบบของเรามีความยืดหยุ่นมากกว่า เนื่องจากเป็นการทำงานบนระบบพื้นฐานของ HTTP Web Resource ทั่วไป และทำงานได้โดยไม่ต้องจำกัดอยู่ภายใน GUI Editor ของ Unity และไม่จำกัดอยู่แค่เพียงขั้นตอนก่อนการ Build ตัวเกม

4.3 การประเมินผลและวิเคราะห์

ในบทนี้จะนำเสนอผลการประเมินเชิงปริมาณของกรอบการทำงานที่พัฒนาขึ้น เปรียบเทียบกับระบบ CMS แบบดั้งเดิม โดยวัดทั้งด้าน ความสะดวกในการใช้งาน (Ease-of-Use) และ ประสิทธิภาพในการทำงานร่วมกัน (Collaboration Efficiency) ผ่านตารางสรุปผล พร้อมคำอธิบายเชิงวิเคราะห์

ตาราง 4.3.1: การเปรียบเทียบ Ease-of-Use ระหว่าง CMS แบบดั้งเดิม และ Web UI บน Unity Web

เกณฑ์ประเมิน	CMS แบบดั้งเดิม	Web UI
Accessibility (การเข้าถึง)	ใช้งานผ่าน Unity Editor หรือ Mobile App ซึ่งมีความยากในการ Deploy Update	ใช้งานผ่านหน้าเว็ป และเป็น Version ล่าสุดเสมอ
Customization (การปรับแต่ง UI)	มีข้อจำกัดสำหรับการสร้าง UI ใน Unity Editor	สามารถสร้างและออกแบบได้อย่างอิสระและสะดวกกว่า
Isolation (การแยกการทดสอบ)	ต้องพึ่งพา Server กลางในการทดสอบ	ทดสอบแยกได้ในเครื่องตัวเอง
Data Transfer	ต้องอัปโหลดข้อมูลไปยัง Server ก่อน	ทดสอบได้ทันทีในเครื่องตัวเอง

(การรับส่งข้อมูล)		
Source Code Protection (การปกป้องซอร์สโค้ด)	ไม่สามารถทำได้กับการใช้งานผ่าน Unity Editor	ทำได้เพราะเป็น Build Version

จากตาราง 4.3.1 จะเห็นได้ว่าระบบใหม่ที่ได้พัฒนาขึ้นสามารถตอบสนองได้ทุกข้อกำหนดที่ได้วางไว้ในหัวข้อ 3.1.1

ในการประเมินประสิทธิภาพในการทำงานร่วมกัน เป็นการเก็บข้อมูลการทดลองเชิงสังเกตการณ์ โดยดำเนินการทดสอบให้ทีมนักพัฒนาเกมที่มีทักษะระดับปานกลาง จากหลากหลายสตูดิโอเกมขนาดเล็ก จำนวนกลุ่มตัวอย่าง 10 กลุ่ม เข้ารับการชี้แจงอธิบายการใช้งานและการทำงานของระบบ แล้วให้ทดลองปฏิบัติงานแต่ละประเภท โดยใช้ระบบการทำงานแบบเดิม เทียบกับระบบการทำงานแบบใหม่ แล้วจึงได้บันทึกผลการใช้เวลาและจำนวนคน ได้ดังตารางนี้

ตาราง 4.3.2: เวลาเฉลี่ยในการทำงานร่วมกันของแต่ละงาน

ลักษณะงาน	CMS แบบดั้งเดิม			Web UI			n	t	p
	จำนวนคน	เวลาเฉลี่ย $\bar{X}$ (นาที)	s.d.	จำนวนคน	เวลาเฉลี่ย $\bar{X}$ (นาที)	s.d.			
การทดสอบ Art Assets	2	58.2	4.2	1	45	2.4	10	8.63	0
การสร้างเนื้อหาเรื่องราว (Narrative Content)	2	119.3	5.6	1	86	4.1	10	15.17	0
การสร้างด่าน/ระดับ (Stages/Levels)	2	126.1	5.9	1	62	3.2	10	30.20	0

จากตาราง 4.3.2 จะเห็นได้ว่าจำนวนผู้เกี่ยวข้องในแต่ละการทำงานลดลงจาก 2 ที่ต้องมีการทำงานของผู้แก้ไขพร้อมกับโปรแกรมเมอร์ เป็นเพียงแค่คนเดียว และใช้เวลาน้อยลงเนื่องจากไม่ต้องมีการรออีกฝ่ายไปมา ปริมาณข้อมูลที่ใช้ในระหว่างทดสอบเป็นขนาดไม่ใหญ่มากอยู่ในระหว่างหลักร้อย MB จึงเพียงพอต่อการทำงานใน Ram ของเครื่องคอมพิวเตอร์ทั่วไป

ตาราง 4.3.3: น้ำหนักความสำคัญของแต่ละงานต่อประเภทเกม

ประเภทเกม	การทดสอบ Art Assets	การสร้างเนื้อหาเรื่องราว	การสร้างด่าน/ระดับ
Casual/Puzzle	4	1	5
Turn-based Strategy	2	3	5
Music/Rhythm	4	1	5
Real-time MMO	3	3	4

จากตาราง 4.3.3 เป็นการจำลองประมาณค่าน้ำหนักของความสำคัญงานแต่ละประเภท ตารางนี้แสดงว่าประเภทเกมแต่ละแบบตามในหัวข้อ 3.2 ว่าต้องพึ่งพาการทำงานของงานต่างๆ ในหัวข้อ 4.2 มากน้อยเพียงใด โดยใช้คะแนนน้ำหนัก 1–5 เพื่อสะท้อนผลกระทบของแต่ละงานต่อประสิทธิภาพโดยรวม

หลังจากนั้นเราจึงได้ทำการคำนวณหาค่าอัตราการพัฒนาเชิงถ่วงน้ำหนักโดยใช้สูตรตามสมการ (1)

$$WIR = \sum \left(\frac{T_{baseline} \times P}{T_{proposed} \times P} \times W \right) \quad (1)$$

WIR = Weighted Improvement Ratio (อัตราการพัฒนาถ่วงน้ำหนัก)

T baseline = เวลาเฉลี่ยของงานเมื่อใช้ระบบเดิม (Baseline CMS)

T proposed = เวลาเฉลี่ยของงานเมื่อใช้ระบบที่เสนอ (Web-based Workflow)

P = จำนวนบุคลากรที่เกี่ยวข้องกับงานนั้น ๆ

W = ค่าน้ำหนักความสำคัญของงานในแต่ละประเภทเกม

ตาราง 4.3.4: อัตราการพัฒนาประสิทธิภาพ

	ค่าน้ำหนักความสำคัญของงานแต่ละประเภท			อัตราการพัฒนาถ่วงน้ำหนัก
ประเภทเกม	**Art Assets**	**Narrative Content**	**Stages/Levels**	
Casual/Puzzle	40%	10%	50%	3.34x
Turn-based Strategy	20%	30%	50%	3.37x
Music/Rhythm	40%	10%	50%	3.34x
Real-time MMO	30%	30%	40%	3.22x

จากตาราง 4.3.4 ตารางนี้แสดงอัตราการพัฒนาของการใช้เวลาในการทำงานเมื่อเปรียบเทียบระบบ CMS แบบเดิมกับระบบ Web-based Editor ของเรา โดยคำนวณแบบถ่วงน้ำหนักสำหรับแต่ละประเภทเกม และได้รวมผลจากการลดจำนวนบุคลากรที่เกี่ยวข้องลงครึ่งหนึ่งเข้าไปด้วย จะเห็นได้ว่าผลจากการใช้ระบบที่นำเสนอทำให้ตัวเลขอัตราประสิทธิภาพการทำงานร่วมกันเพิ่มขึ้นอย่างมีนัยสำคัญ ทั้งนี้ผลกระทบหลักมาจากการลดการใช้เวลาพร้อมกันในการทำงาน ที่เป็นการสิ้นเปลืองเวลาในการรอกันไปมา มีความแตกต่างในแต่ละประเภทงานและประเภทเกมเพียงเล็กน้อยเท่านั้น

5. บทสรุป

งานวิจัยนี้ได้นำเสนอแนวทางการพัฒนาเกมบน Unity Web-based Ecosystem โดยมุ่งเน้นการสร้างกรอบการทำงานสำหรับ การสร้างและทดสอบเนื้อหาเกมอย่างรวดเร็ว ปลอดภัย และยืดหยุ่น ผลการศึกษาแสดงให้เห็นว่าระบบ Web-based Editor สามารถช่วยให้ศิลปินและนักออกแบบเนื้อหาทำงานได้โดยไม่ต้องเข้าถึงซอร์สโค้ดของ Unity และสามารถทดสอบเนื้อหาแบบ isolated local sandbox ก่อน commit กลับไปยังเซิร์ฟเวอร์หลัก ทำให้ลดความซับซ้อนและคอขวดในการทำงานร่วมกัน

ผลการประเมินเปรียบเทียบกับระบบ CMS แบบดั้งเดิม พบว่า ความสะดวกในการใช้งาน (Ease-of-Use) และประสิทธิภาพการทำงานร่วมกัน (Collaboration Efficiency) เพิ่มขึ้นอย่างชัดเจน โดยเฉพาะในด้าน การเข้าถึง, การปรับแต่ง, การแยกทดสอบ, และการปกป้องซอร์สโค้ด การทำงานแบบ local sandbox และ runtime data injection ช่วยให้ผู้สร้างเนื้อหาสามารถเห็นผลลัพธ์ทันที ลดเวลาการทดสอบและการปรับแก้ซ้ำ

นอกจากนี้ ระบบสามารถรองรับ ประเภทเกมที่หลากหลาย ตั้งแต่ Casual/Puzzle, Turn-based Strategy, Music Rhythm ไปจนถึง MMO โดยการปรับใช้ workflow และสถาปัตยกรรมระบบให้เหมาะสมกับแต่ละประเภทเกม ส่งผลให้การจัดการเนื้อหาและการทดสอบมีความยืดหยุ่นและประสิทธิภาพสูง

อย่างไรก็ตาม แม้ว่าผลการทดสอบจะออกมาดีมาก แต่ก็ยังขาดการนำไปทดลองกับทีมเกมขนาดใหญ่ และระบบเกมขนาดใหญ่ที่มีความซับซ้อนสูงมากๆ ว่าแนวคิดนี้สามารถขยาย Scale-Up ได้หรือไม่

โดยสรุป งานวิจัยนี้เสนอ แนวทางการพัฒนาเกมบน Web-based Unity ที่ช่วยเพิ่มความรวดเร็ว ความปลอดภัย และความสะดวกในการสร้างเนื้อหาเกม พร้อมสนับสนุนการทำงานร่วมกันระหว่างทีมงานที่แตกต่างบทบาทอย่างมีประสิทธิภาพ

เอกสารอ้างอิง

[1] Unity Technologies. (n.d.-b). Unity - Manual: Web introduction. Retrieved March 10, 2024, from https://docs.unity3d.com/2023.3/Documentation/Manual/webgl-intro.html

[2] Unity Technologies. (n.d.-a). Unity - Manual: Creating and Using Scripts. Retrieved March 10, 2024, from https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html

[3] Etaiwi, L., Ullmann, G., Guéhéneuc, Y.-G., & Hamel, S. (n.d.). Consensus in Game Engine Architectures: An Overview of Subsystem Coupling in Game Engines. Transfer, 31, 22.

[4] Hussain, A., Shakeel, H., Hussain, F., Uddin, N., & Ghouri, T. L. (2020). Unity game development engine: A technical survey. Univ. Sindh J. Inf. Commun. Technol, 4(2), 73–81.

[5] Vyas, R. (2022). Comparative Analysis on Front-End Frameworks for Web Applications. International Journal for Research in Applied Science and Engineering Technology, 10(7), 298–307.

[6] Cross-Language Interoperability. (n.d.). Microsoft Developer Network (msdn.microsoft.com). Retrieved March 10, 2024, from https://msdn.microsoft.com/en-us/subscriptions/50b25433-59f2-4c6f-9774-7d777cde6b0a%28v=vs.90%29

[7] JavaScript | MDN. (2024, March 5). https://developer.mozilla.org/en-US/docs/Web/JavaScript

[8] ECMA-262. (n.d.). Ecma International. Retrieved March 10, 2024, from https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

[9] Specifications—WebAssembly. (n.d.). Retrieved March 10, 2024, from https://webassembly.org/specs/

[10] Aldahir, A. (2022). Evaluation of the performance of WebGPU in a cluster of WEB-browsers for scientific computing. https://www.diva-portal.org/smash/record.jsf?pid=diva2:1674447

[11] WebGL 2.0 Specification. (n.d.). Retrieved March 10, 2024, from https://registry.khronos.org/webgl/specs/latest/2.0/

[12] WebGPU. (n.d.). Retrieved March 10, 2024, from https://www.w3.org/TR/webgpu/

[13] Fransson, E., & Hermansson, J. (2023). Performance comparison of WebGPU and WebGL in the Godot game engine. https://www.diva-portal.org/smash/record.jsf?pid=diva2:1762429

[14] Urbanek, M., Güldenpfennig, F., & Habiger, M. (2019). Creating Audio Games Online with a Browser-Based Editor. 272–276. https://doi.org/10.1145/3356590.3356636

[15] Bossalini, C., Raffe, W., & Andres Garcia, J. (2020). Generative Audio and Real-Time Soundtrack Synthesis in Gaming Environments: An exploration of how dynamically rendered soundtracks can introduce new artistic sound design opportunities and enhance the immersion of interactive audio spaces. 32nd Australian Conference on Human-Computer Interaction, 281–292. https://doi.org/10.1145/3441000.3441075

[16] Nikolova, A., Georgiev, V., & Mitreva, E. (2019). Using Unity and Shield UI for Displaying 3D Medical Education Objects in the Web Browser. https://doi.org/10.55630/dipp.2018.8.22