GAME PROGRAMMING

มหาวิทยาลัยราชภัฏสวนสุนันทา

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์และเทคโนโลยีอุตสาหกรรม

GAME INDUSTRY OVERVIEW

การเติบโตของตลาดโลก

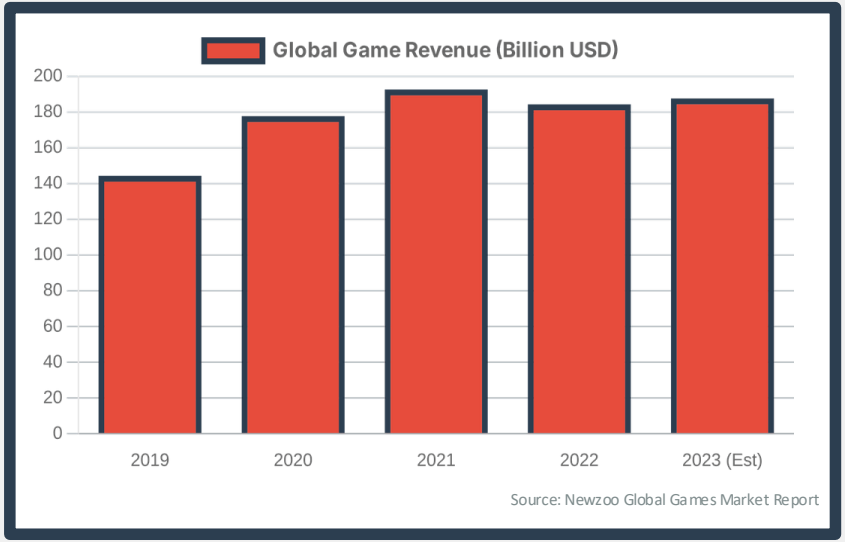
มูลค่าตลาดเกมทั่วโลกเติบโตอย่างต่อเนื่อง คาดการณ์ว่าจะสูงถึง 2 แสนล้านดอลลาร์สหรัฐ

โอกาสในสายอาชีพ

Game Developer, Designer, Technical Artist, AI Engineer และ Game Game Engine Programmer

ความสำคัญของการเรียนรู้

สร้างทักษะการแก้ปัญหาที่ซับซ้อน การจัดการทรัพยากร และการประมวลผลแบบ Real-time



EVOLUTION OF VIDEO GAMES

1970s

ยุคบุกเบิก: กำเนิดเกม Arcade เช่น Pong และ Space Invaders เน้นกลไกพื้นฐาน

1980s

ยุคทอง: เครื่องเกมคอนโซลครองตลาด (NES, Sega) กำเนิดตัวละครไอคอนอย่าง Mario

1990s

ยุค 3D: การปฏิวัติด้วยกราฟิก 3 มิติ (PlayStation) เปลี่ยนมิติการเล่นเกมไปตลอดกาล

2000s+

ยุคปัจจุบัน: เกมออนไลน์, มือถือ และเทคโนโลยี VR/AR ที่เชื่อมต่อคนทั้งโลก

GAME GENRES EXPLORATION

ACTION

เน้นการทดสอบทักษะการตอบสนอง ความแม่นยำ และจังหวะของผู้เล่น

e.g., Platformer, Shooter, Fighting

ADVENTURE

เน้นการสำรวจ การแก้ปริศนา และการดำเนินเนื้อเรื่องที่เข้มข้น

e.g., Point-and-Click, Visual Novels

RPG

ผู้เล่นสวมบทบาทตัวละคร พัฒนาทักษะ และตัดสินใจที่มีผลต่อเนื้อเรื่อง

e.g., Action RPG, MMORPG

STRATEGY

เน้นการวางแผน การบริหารทรัพยากร และการใช้ความคิดเชิงกลยุทธ์

e.g., RTS, Turn-Based Strategy

SIMULATION

จำลองสถานการณ์หรือกิจกรรมในโลกจริงให้ผู้เล่นได้สัมผัส

e.g., Life Sim, Vehicle Simulation

OTHERS

หมวดหมู่อื่นๆ เช่น กีฬา, ปริศนา (Puzzle), และเกมแนวผสมผสาน (Hybrid)

e.g., FIFA, Tetris, Portal

GAME DESIGN PRINCIPLES

CLEAR GOALS

เป้าหมายที่ชัดเจน: ผู้เล่นต้องเข้าใจทันทีว่าต้องทำอะไร และความสำเร็จในเกมคืออะไร เพื่อสร้างทิศทางในการเล่น

BALANCED CHALLENGE

ความท้าทายที่เหมาะสม: รักษาสมดุลระหว่างความยากและทักษะของผู้เล่น (Flow State) ไม่ให้ยากจนท้อหรือง่ายจนเบื่อ

REWARD SYSTEM

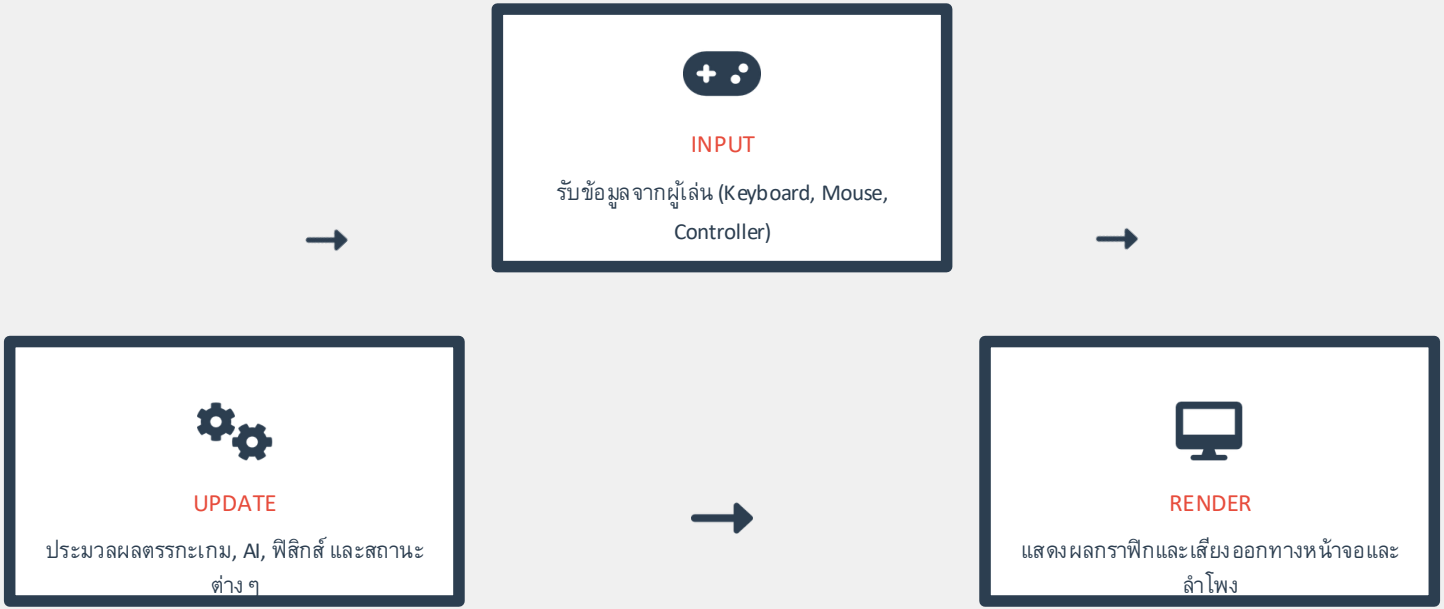
ระบบการให้รางวัล: การให้ Feedback เชิงบวกเมื่อผู้เล่นทำสำเร็จ เพื่อสร้างแรงจูงใจและสารโดพามีนในสมอง

ENGAGING MECHANICS

กลไกเกมที่น่าสนใจ: วิธีการโต้ตอบ (Interaction) ที่สนุกและลื่นไหล ซึ่งเป็นหัวใจสำคัญที่ทำให้ผู้เล่นอยากกลับมาเล่นซ้ำ

** การผสมผสานหลักการเหล่านี้อย่างลงตัวคือหัวใจของการสร้างประสบการณ์ที่น่าจดจำ*

THE HEART OF GAMES: GAME LOOP



วัฏจักรนี้ทำงานซ้ำ 60+ ครั้งต่อวินาที (FPS)

2D/3D GRAPHICS & RENDERING

2D GRAPHICS

- **Sprites:** รูปภาพ 2 มิติที่ใช้แทนตัวละครหรือวัตถุ สามารถทำ Animation ได้โดยการเปลี่ยนเฟรมภาพ
- **Tilemaps:** การสร้างฉากขนาดใหญ่จาก "กระเบื้อง" รูปภาพขนาดเล็ก ช่วยประหยัดหน่วยความจำ
- **Parallax Scrolling:** การเลื่อนเลเยอร์ฉากหลังด้วยความเร็วที่ต่างกันเพื่อสร้างมิติความลึก

3D GRAPHICS

- **3D Models:** วัตถุที่ประกอบขึ้นจาก Polygons (Vertices, Edges, Faces) ในพื้นที่ 3 มิติ
- **Textures & UV:** การนำรูปภาพ 2D มาห่อหุ้มโมเดล 3D เพื่อเพิ่มรายละเอียดพื้นผิว
- **Shaders:** โปรแกรมขนาดเล็กที่ทำงานบน GPU เพื่อคำนวณแสง เงา และและเอฟเฟกต์พิเศษ

GAME PHYSICS FUNDAMENTALS

Collision Detection

การตรวจสอบว่าวัตถุสองชิ้นมีการซ้อนทับหรือชนกันหรือไม่ (e.g., AABB, Circle, Raycasting)

Collision Response

การคำนวณผลลัพธ์หลังการชน เช่น การกระดอน (Bouncing) หรือการหยุดนิ่ง

Gravity & Forces

การจำลองแรงโน้มถ่วง แรงเสียดทาน และแรงผลักต่างๆ เพื่อให้วัตถุเคลื่อนที่อย่างสมจริง

PHYSICS ENGINE CORE

```
F = m * a
```

Rigid Body: วัตถุที่มีรูปทรงคงที่ ไม่บิดเบี้ยว

Kinematics: การเคลื่อนที่โดยไม่คำนึงถึงแรง

Dynamics: การเคลื่อนที่ที่เกิดจากแรงกระทำ

Constraints: ข้อจำกัดการเคลื่อนที่ (e.g., Joints)

Integration: การคำนวณตำแหน่งใหม่จากความเร็ว

ARTIFICIAL INTELLIGENCE IN GAMES

PATHFINDING

การหาเส้นทางที่สั้นที่สุดจากจุดหนึ่งไปยังอีกจุดหนึ่ง โดยหลบหลีกสิ่งกีดขวาง อัลกอริทึมที่นิยมที่สุดคือ **A* (A-Star)**

FINITE STATE MACHINES (FSM)

การกำหนดสถานะและพฤติกรรมของ AI เช่น **Idle, Patrol, Chase, Attack** และเงื่อนไขในการเปลี่ยนสถานะ

BEHAVIOR TREES

โครงสร้างแบบต้นไม้ที่ซับซ้อนกว่า FSM ช่วยให้ AI ตัดสินใจได้หลากหลายและดูเป็นธรรมชาติมากขึ้น

MACHINE LEARNING

การใช้ **Reinforcement Learning** เพื่อให้ AI เรียนรู้และพัฒนาทักษะการเล่นเกมด้วยตัวเองผ่านการลองผิดลองถูก

MODERN GAME ENGINES

UNITY

- **Language:** C# (Easy to learn)
- **Focus:** Mobile, 2D, and Indie Games
- **Asset Store:** Huge library of assets
- **Community:** Massive support & tutorials

BEST FOR: Cross-platform & Rapid Prototyping

UNREAL ENGINE

- **Language:** C++ & Blueprints
- **Graphics:** Photorealistic Rendering
- **Visual Scripting:** Powerful Blueprints
- **Industry:** AAA Games & Film Production

BEST FOR: High-end Graphics & Large Scale Projects

CROSS-PLATFORM DEVELOPMENT

PC

High Performance: ข้อ จำกัดด้านฮาร์ดแวร์น้อยที่สุด รองรับกราฟิกขั้นสูง

Open Platform: อิสระในการเผยแพร่ (Steam, Epic, Itch.io)

Input Variety: รองรับทั้ง Keyboard/Mouse และ Controller

MOBILE

Touch Controls: การออกแบบ UI/UX ที่เน้นการสัมผัสและท่าทาง

Constraints: ข้อ จำกัดด้านพลังงาน ประมวลผล แบตเตอรี่ และความร้อน

Accessibility: เข้าถึงผู้เล่นได้ง่ายและกว้างขวางที่สุดในโลก

CONSOLE

Fixed Hardware: พัฒนาได้ง่ายเพราะสเปกเครื่องคงที่ (PS5, Xbox, Switch)

Strict Approval: กระบวนการตรวจสอบคุณภาพที่เข้มงวดจากเจ้าของแพลตฟอร์ม

Living Room: เน้นประสบการณ์การเล่นบนหน้าจอขนาดใหญ่

ANY QUESTIONS?

KEY TAKEAWAYS

- **Game Loop Mechanics**
- **Physics Simulation**
- **Engine Selection**
- **2D/3D Rendering**
- **Game AI Logic**
- **Platform Constraints**

สาขาวิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยราชภัฏสวนสุนันทา

THANK YOU FOR YOUR ATTENTION!