บทที่ 20
ทฤษฎี Finite State Machine ในเกม 3 มิติ

ทฤษฎี Finite State Machine (FSM) คือระบบที่ถูกนิยามขึ้นมาโดยมีการทำงานที่จำกัด มองการทำงานให้เป็นระบบแยกย่อย ใช้สำหรับงานที่ต้องการเฉพาะเจาะจง เช่น นาฬิกาปลุก มีสเตทการทำงานแบ่งออกเป็น (1) บอกเวลา (2) ส่งเสียงปลุกตามเวลา (3) เปิด-ปิดเสียงปลุก เป็นต้น การทำงานเป็นสเตทสามารถจำลองให้คอมพิวเตอร์ทำงานได้ตามคำสั่งของผู้เขียนโปรแกรม โดยมากทฤษฎี FSM ถูกนำไปใช้งานในส่วนของ AI เพื่อจำลองให้คอมพิวเตอร์แสดงพฤติกรรมโต้ตอบเสมือนเป็นมนุษย์คนหนึ่ง และสามารถนำมาประยุกต์ใช้กับเกม 3 มิติได้อีกด้วย

20.1 การสร้าง Finite State Machine

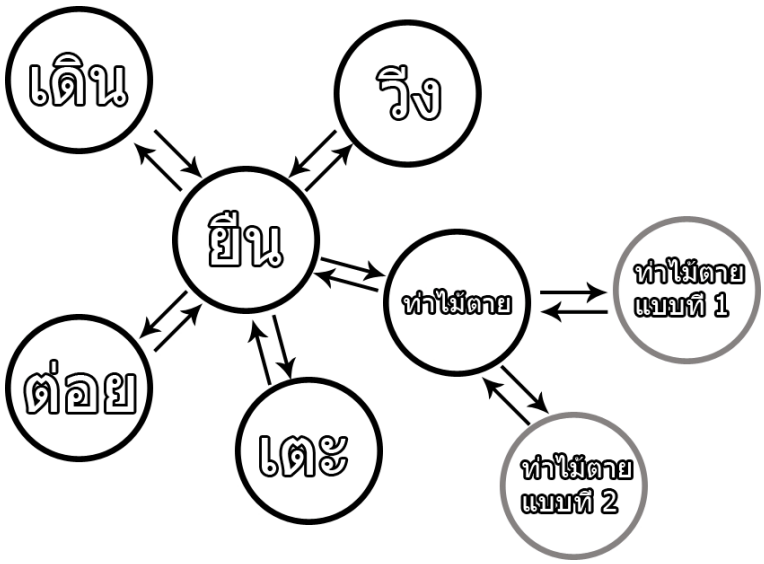
Finite State Machine มีขั้นตอนที่สำคัญ 2 ขั้นตอนคือ (1) นิยามสเตท (2) เงื่อนไขการเปลี่ยนสเตท โดยให้สังเกตุจากโจทย์ต่อไปนี้

> บริษัทเขียนเกมแห่งหนึ่งต้องการสร้างนักสู้ 1 คนโดยนำไปใช้ในเกมต่อสู้ 3 มิติ ผู้อ่านเป็นผู้เชี่ยวชาญในการวิเคราะห์เกมต่อสู้จะแก้ปัญหาให้กับบริษัทนี้อย่างไร

วิธีการวิเคราะห์ปัญหา จะนำวิธีการสร้าง Finite State Machine เข้ามาช่วย ตามทฤษฎีจะมี 2 ขั้นตอนที่ต้องทำก่อนคือ

(1) นิยามสเตท จากโจทย์นักสู้ 1 คนให้ผู้อ่านลองจินตนาการตามไปว่าพื้นฐานต้องมีสเตทการแสดงออกอะไรบ้าง อาจนิยามสเตทได้ดังนี้ ยืน เดิน วิ่ง ต่อย เตะ และท่าไม้ตาย เมื่อนิยามสเตทได้แล้วขั้นตอนต่อไปคือเงื่อนไขในการเปลี่ยนสเตท

(2) เงื่อนไขการเปลี่ยนสเตท จากสเตทยืนไปเดิน โดยทั่วไปจะถูกผู้เล่นสั่งการโดยใช้คีย์บอร์ดหรือจอยสติ๊กกดปุ่มขึ้น ลง ซ้าย ขวา วิ่งอาจกดปุ่มเดินหน้าอย่างเร็ว 2 ที ต่อย แตะกดปุ่มโจมตี ส่วนท่ามให้ตายอาจใช้การควงปุ่มเป็นครึ่งวงกลมแล้วกดปุ่มต่อยหรือเตะ เงื่อนไขต่างๆ เหล่านี้จะแสดงให้เห็นตามแผนภาพด้านล่าง



ภาพการทำงานเงื่อนไขการเปลี่ยนสเตท FSM

จากการวิเคราะห์ข้อมูลเบื้องต้นโดยใช้ทฤษฎี FSM ทำให้ได้ทราบถึงการทำงานของนักสู้ 1 คนที่มีพฤติกรรมยืน เดิน วิ่ง ต่อย เตะ และท่าไม้ตาย ในขั้นต่อไปจะนำสิ่งที่วิเคราะห์เขียนเป็นโค้ดโปรแกรม โดย

ก่อนที่จะเขียนส่วนของ FSM เพิ่มผู้อ่านต้องทำากรโหลดโมเดลที่มีท่า Animation ดังที่กล่าวข้างต้นไว้ก่อนจึงนำโค้ดต่อไปนี้ไปเขียนเพิ่มเติม

Step 1 of 2

```
#define STATE_IDLE      0
#define STATE_STAND     1
#define STATE_WALK      2
#define STATE_RUN       3
#define STATE_PUNCH     4
#define STATE_KICK      5
#define STATE_SKILL     6
int state = STATE_STAND;
```

- นิยามตัวแปรเพื่อใช้ในการแสดงท่าทางต่างๆคือท่ายืน เดิน วิ่ง ต่อย เตะ และท่าไม้ตาย และมีตัวแปรชื่อ state มีหน้าที่ควบคุมการเปลี่ยนท่า

Step 2 of 2

```
if( g_input.keyPressed(KEY_F1) ) state = STATE_STAND;
if( g_input.keyPressed(KEY_F2) ) state = STATE_WALK;
if( g_input.keyPressed(KEY_F3) ) state = STATE_RUN;
if( g_input.keyPressed(KEY_F4) ) state = STATE_PUNCH;
if( g_input.keyPressed(KEY_F5) ) state = STATE_KICK;
if( g_input.keyPressed(KEY_F6) ) state = STATE_SKILL;
switch( state ) {
        case STATE_STAND:
                node->setFrameLoop( 0, 150 ); state = STATE_IDLE;
        break;
        case STATE_WALK:
                node->setFrameLoop( 659, 719 );  state = STATE_IDLE;
        break;
        case STATE_RUN:
```

```
                node->setFrameLoop( 780, 798 );  state = STATE_IDLE;
        break;
        case STATE_PUNCH:
                node->setFrameLoop( 1114, 1140 ); state = STATE_IDLE;
        break;
        case STATE_KICK:
                node->setFrameLoop( 1207, 1234 ); state = STATE_IDLE;
        break;
        case STATE_SKILL:
                node->setFrameLoop( 1661, 1689 );  state = STATE_IDLE;
        break;
}
```

- ใช้ปุ่ม F1 - F6 เพื่อเปลี่ยนท่า Animation โดยเซ็ทตัวแปรชื่อ state ที่ใช้ควบคุมให้เป็นสเตทที่ต้องการคือท่ายืน เดิน วิ่ง ต่อย เตะ และท่าไม้ตาย ตามลำดับ
- ใช้คำสั่ง switch() เช็คเงื่อนไขการทำงานของสเตทต่างและใช้คำสั่ง setFrameLoop() เพื่อกำหนดคีย์เฟรมเริ่มและจบ

จากตัวอย่างทำให้ผู้อ่านสามารถควบคุม ตัวละครให้แสดงพฤติกรรมตามที่โจทย์ได้ตั้งคำถาม อีกทั้งยังสามารถเพิ่มเติมส่วนต่างๆ ตามแต่ละสเตทได้ตามที่ต้องการ

สรุป Finite State Machine คือระบบที่ถูกนิยามขึ้นมาโดยมีการทำงานที่จำกัด มีขั้นตอนที่สำคัญ 2 ขั้นตอนคือ (1) นิยามสเตท (2) เงื่อนไขการเปลี่ยนสเตท ในการวิเคราะห์ข้อมูลอาจมีการสร้างแผนภาพ FSM ออกมาก่อนเพื่อความชัดเจน จากนั้นจึงลงมือเขียนโค้ดโปรแกรมเพื่อให้ได้งานตามที่ออกแบบไว้