รายงาน

รายวิชา CPSC 421 ปัญญาประดิษฐ์

จัดทำโดย

นายสุริยา พุฒดวง

รหัสนิสิต 6108111005

คณะบริหารธุรกิจและรัฐประศาสนศาสตร์

สาขาวิทยาการคอมพิวเตอร์

เสนอ

อ.วิเชพ ใจบุญ

มหาวิทยาลัยเนชั่น ปีการศึกษา 1/2564

คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของวิชา CPSC 421 ปัญญาประดิษฐ์ โดยมีจุดประสงค์เพื่อศึกษาเรื่องของการใช้ AI สร้างหมากรุก เพื่อเป็นแนวทางในการศึกษา และเสริมทักษะความรู้ให้กับตัวเองและผู้อ่านรายงานฉบับนี้

ข้าพเจ้าหวังว่ารายงานฉบับนี้จะทำให้ทุกท่านได้ประโยชน์ไม่มากก็น้อยหากผิดพลาดประการใดข้าพเจ้าขออภัยมา ณ ที่นี้ด้วย

ท้ายนี้ ขอขอบคุณ อ.วิเชพ ใจบุญ ผู้ให้ความรู้และแนวทางในการศึกษาเพื่อพัฒนาวินัยในการทำงานต่อไป

สุริยา พุฒดวง

ผู้จัดทำ

สารบัญ

ความเป็นมาของปัญญาประดิษฐ์

ปัญญาประดิษฐ์ได้เริ่มการ ศึกษาในปี ค.ศ.1950 โดยอาจารย์จาก ประเทศอเมริกาและอังกฤษ นิยามของปัญญาประดิษฐ์ได้ถูกกำหนดขึ้นในปี 1956 โดย John McCarthy ได้มีการศึกษา และพัฒนางานด้านปัญญาประดิษฐ์และได้มีการตั้งเกณฑ์ทดสอบเพื่อที่จะระบุว่า เครื่องจักรกลหรือระบบคอมพิวเตอร์สามารถคิดได้เหมือนมนุษย์ออกมาโดย Alan Turing นักคณิตศาสตร์ชาวอังกฤษ แต่จนบัดนี้เครื่องจักรกลหรือระบบคอมพิวเตอร์ก็ยังไม่สามารถผ่านเกณฑ์ของ Alan Turing ได้เลย ณ ปัจจุบันระบบปัญญาประดิษฐ์ยังไม่สามารถสร้างคำตอบที่แปลกใหม่หรือคำตอบที่มา จากการคิดค้นขึ้นมาใหม่ของระบบเองได้ เพียงแต่เป็นการลอกเลียนความสามารถของมนุษย์ได้เท่านั้น

ปัญญาประดิษฐ์ (Artificial Intelligence : AI) คือ ศาสตร์แขนงหนึ่งทางด้านวิทยาศาสตร์และเทคโนโลยีที่มีพื้นฐานมาจากวิชาวิทยา การคอมพิวเตอร์ ชีววิทยา จิตวิทยา ภาษาศาสตร์ คณิตศาสตร์ และวิศวกรรมศาสตร์ เป้าหมายคือ การพัฒนาระบบคอมพิวเตอร์ให้มีพฤติกรรมเลียนแบบมนุษย์ รวมทั้งเลียนแบบความเป็นอัจฉริยะของมนุษย์

ลักษณะงานของปัญญาประดิษฐ์

1. Cognitive Science

งาน ด้านนี้เน้นงานวิจัยเพื่อศึกษาว่าสมองของมนุษย์ทำงานอย่างไร และมนุษย์คิดและเรียนรู้อย่างไร จึงมีพื้นฐานที่การประมวลผลสารสนเทศในรูปแบบของมนุษย์ประกอบด้วยระบบต่างๆ

ระบบผู้เชี่ยวชาญ (Expert Systems)

ระบบเครือข่ายนิวรอน (Neural Network)

ระบบแบ็บแน็ต (Papnet)

ฟัสซี่โลจิก (Fuzzy Logic)

เจนเนติกอัลกอริทึม (Genetic Algorithm)

เอเยนต์ชาญฉลาด (Intelligent Agents)

ระบบการเรียนรู้ (Learning Systems)

2. Roboics

พื้นฐานของวิศวกรรมและสรีรศาสตร์ เป็นการพยายามสร้างหุ่นยนต์ให้มีความฉลาดและถูกควบคุมด้วยคอมพิวเตอร์แต่ สามารถเครื่องไหวได้เหมือนกับมนุษย์

3. Natural Interface งานด้านนี้ได้ชื่อว่าเป็นงานหลักที่สำคัญที่สุดของปัญญาประดิษฐ์ และพัฒนาบนพื้นฐานของภาษาศาสตร์ จิตวิทยา และวิทยาการคอมพิวเตอร์ประกอบด้วยงานด้านต่างๆ

ระบบที่มีความสามารถในการเข้าใจภาษามนุษย์ (Natural Language)

ระบบภาพเสมือนจริง (Virtual Reality)

ระบบปัญญาประดิษฐ์แบบผสมผสาน (Hybrid AI Systems)

ระบบผู้เชี่ยวชาญ (Expert Systems)

เป็นระบบที่ช่วยในการแก้ปัญหาหรือช่วยในการตัดสินใจโดยใช้วิธีเดียวกับผู้เชี่ยวชาญที่เป็นมนุษย์

องค์ประกอบของผู้เชี่ยวชาญ

1. ฐานความรู้ (Knowledge Base) เป็นส่วนของความรู้ของผู้เชี่ยวชาญทั้งหมด ซึ่งจะเก็บไว้ในฐานข้อมูลของระบบ

2. โปรแกรมของระบบผู้เชี่ยวชาญ (Expert System Software หรือ Software Resources) แบ่งออกได้ 2 ส่วน

   1) ส่วนที่ใช้ในการประมวลผลความรู้จากฐานความรู้

   2) ส่วนที่ใช้ในการติดต่อสื่อสารกับผู้ใช้

ระบบสารสนเทศภูมิศาสตร์ (Geographic Information Systems : GIS)

คือกระบวนการทำงานเกี่ยวกับข้อมูลที่มีความสัมพันธ์ในเชิงพื้นที่ด้วย ระบบคอมพิวเตอร์ เช่น ที่อยู่ บ้านเลขที่ สัมพันธ์กับตำแหน่งในแผนที่ ตำแหน่ง เส้นรุ้ง เส้นแวง เป็นต้น

องค์กรและหน่วยงานที่เกียวข้องและให้การสนับสนุนทางด้าน ปัญญาประดิษฐ์ ได้แก่

1. The Ameriscan Association for Artificial Intelligence (AAAI)

2. The European Coordinating Committee for Artificial Intelligence (ECCAI)

3. The society for Artificial Intelligence and Simulation of Behavior (AISB)

องค์กรทั้งสามนี้ให้การสนับสนุนการค้นคว้าทางด้าน ปัญญาประดิษฐ์ องค์กรทั้งในกลุ่มที่สนใจใน ปัญญาประดิษฐ์ เป็นพิเศษโดยใช้สื่อว่า SIGART คือ The Association for Computing Machinery (ACM)

ปัญญาประดิษฐ์ (Artificial intelligence : AI) หมายถึง การทำให้คอมพิวเตอร์สามารถคิดหาเหตุผลได้ เรียนรู้ได้ ทำงานได้เหมือนสมองมนุษย์ หรือการพัฒนาให้ระบบคอมพิวเตอร์มีลักษณะการทำงานใกล้เคียงกับระบบการประมวลผล และการตอบสนองของมนุษย์ที่มีต่อแต่ละสถานการณ์ เพื่อให้คอมพิวเตอร์สามารถปฏิบัติงานแทนที่มนุษย์ได้อย่างมีประสิทธิภาพ เช่น หุ่นยนต์ หรือ robot เป็นต้น

ปัญญาประดิษฐ์ (Artificial intelligence : AI) คือความพยายามในการพัฒนาระบบคอมพิวเตอร์ (ทั้งฮาร์ดแวร์ และ ซอฟต์แวร์) ให้มีพฤติกรรมเลียนแบบมนุษย์ ระบบต่างๆจะต้องมีความสามารถเข้าใจภาษามนุษย์ ทำงานที่ต้องใช้การประสานงาน ระหว่างส่วนต่างๆ (โรโบติก - robotics) ใช้อุปกรณ์ที่สามารถรับทราบ และตอบสนอง ด้วยพฤติกรรม และภาษา (ระบบการมอง และ การออกเสียง) การเลียนแบบความเชี่ยวชาญและการตัดสินใจของมนุษย์ (ระบบผู้เชี่ยวชาญ) ระบบดังกล่าวยังต้องแสดง ความสามารถทางตรรกะ การใช้เหตุผล สัญชาตญาณ และใช้หลักการสมเหตุสมผล (common sense) ที่มีคุณภาพ ในระดับเดียวกับมนุษย์ รูปแสดงองค์ประกอบต่างๆ ของระบบปัญญาประดิษฐ์ อีกสิ่งหนึ่งที่สำคัญไม่แพ้กันคือ เครื่องจักรชาญฉลาด (intelligent machine) หรืออุปกรณ์ที่แสดงความสามารถที่กล่าวถึงนี้

การนำระบบผู้เชี่ยวชาญไปใช้งานในด้านต่างๆ (Putting expert systems to work)

1. ด้านการผลิต (Production)

2. การตรวจสอบ (Inspection)

3. การประกอบชิ้นส่วน (Assembly)

4. ด้านบริการ (Field service)

5. ด้านการซ่อมแซมโทรศัพท์ (Telephone repair)

6. การตรวจสอบบัญชี (Auditing)

7. การคิดภาษี (Tax accounting)

8. การวางแผนด้านการเงิน (Financial planning)

9. ด้านการลงทุน (Investments)

10. ด้านบุคคล (Personnel)

11. ด้านการตลาด และการขาย (Marketing and sales)

12. การอนุมัติสินเชื่อ (Credit authorization)

13. หน่วยงานด้านบริการของรัฐ (Human services agency)

14. การทำนายทางการแพทย์ (Medical prognosis) ระบบผู้เชี่ยวชาญ เป็นระบบที่ใช้คอมพิวเตอร์วินิจฉัยโรค

เทคนิคการค้นหาเมื่อมีคู่ปรปักษ์

ปัญหาของเกมที่มีการแข่งขันกันระหว่างผู้เล่น 2 คน จะไม่สามารถใช้เทคนิคการค้นหาปกติเพื่อแก้ไขปัญหาได้ เนื่องจากต้องมีการค านึงถึงการเล่นของฝ่ายตรงข้ามด้วย ดังนั้นเพื่อแก้ปัญหาในลักษณะนี้จึงมีการประยุกต์ใช้เทคนิคการค้นหาเมื่อมีคู่ปรปักษ์ (Adversarial Search) โดยปกติเกมที่น ามาพิจารณานั้นผู้เล่นจะต้องทราบข้อมูลภายในเกมขณะนั้นทั้งหมด เช่น เกมหมากรุก และเกม tic-tac-toe เป็นต้น

ส่วนประกอบพื้นฐานที่ต้องกำหนด

ส่วนประกอบพื้นฐานที่ต้องกำหนดใน Adversarial Search Techniques มีดังนี้

- **Initial state** กำหนดสถานะเริ่มต้น

- **Successor Function** กำหนดเซตของการกระทำทั้งหมดที่เป็นไปได้

- **Terminal Test** ตัวกำหนดการสิ้นสุด สถานะสิ้นสุดเรียกว่า Terminal state

- **Utility Function** ฟังก์ชันที่กำหนดค่าของ Terminal state เป็นตัวเลขบ่งบอกผลลัพธ์ของเกม เช่น ชนะ (+1) แพ้ (-1) เสมอ (0) ถ้าบางเกมมีการวัดผลที่ซับซ้อนอาจใช้ค่าได้

เทคนิควิธีการค้นหาคำตอบของเกมประเภทนี้มีเทคนิคที่นิยมกันอยู่ 2 เทคนิคคือ เทคนิควิธีการค้นหาแบบ **Minimax** และ เทคนิควิธีการค้นหาแบบ **Alpha-Beta Pruning** ซึ่งโดยปกติถ้าเป็นการแก้ไข

ปัญหาที่มีจำนวนเส้นทางสำรวจน้อย สามารถใช้วิธีการค้นหาแบบ Minimax ได้เลยเนื่องจากง่ายต่อการพัฒนาโปรแกรม แต่หากปัญหานั้นมีจำนวนเส้นทางสำรวจกว้างจะใช้วิธีการค้นหาแบบ Alpha-Beta Pruning (α-β Pruning) มาช่วยตัดบางเส้นทางที่ไม่จำเป็นออกไป ทำให้การค้นหาใช้ทรัพยากรน้อยลงและรวดเร็วมากขึ้น

วิธีการค้นหาแบบ Minimax

Minimax จะมีลักษณะการค้นหาแบบ Depth-First Search โดยพิจารณาค่าต่างๆ จากโหนดใบไม้และเลือกค่าที่เหมาะสมขึ้นมาบนโหนดพ่อแม่ ทำเช่นนี้ไปจนกว่าโหนดรากจะได้รับค่า เกมที่นำมาพิจารณามักเป็นเกมที่เล่นกัน 2 คน กำหนดให้ผู้เล่นคนหนึ่งเป็น “MAX” (แทนผู้เล่นคือเรา) วัตถุประสงค์ของ MAX คือ การทำคะแนนให้มากที่สุดหรือหาทางชนะ ส่วนผู้เล่นอีกคนเป็น “MIN” วัตถุประสงค์ คือ พยายามทำให้ MAX ได้ค่าน้อยที่สุด หรือพยายามทำให้ MAX แพ้นั่นเอง โดยสัญลักษณ์ของโหนด MAX และ MIN

ตัวอย่างการใช้ Minimax แก้ปัญหาเกม tic-tac-toe

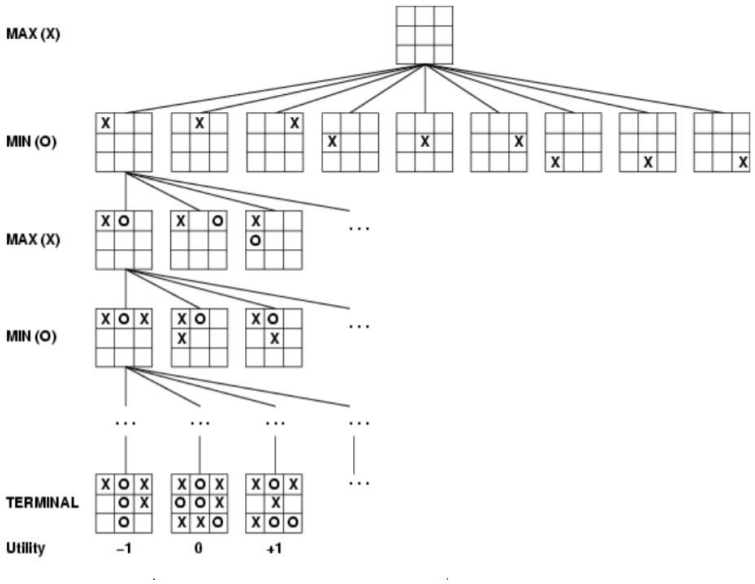
Initial state กำหนดสถานะเริ่มต้น โดยจะแบ่งตารางเป็นแบบ 3x3 และ state จะกำหนดเป็น Array ทั้งหมด 9 ช่อง เมื่อผู้เล่นทำการเลือก state ที่จะวางสัญลักษณ์ของตัวเอง state นั่นก็จะไม่สามารถวางซ้ำได้ ในแต่ละตาผู้เล่นจะได้รับทรัพยากรการวางสัญลักษณ์ของตัวเอง 1 ครั้ง

Path Cost การกระทำของผู้เล่นในแต่ละตา เมื่อทำการวางสัญลักษณ์ของตัวเอง ก็จะใช้ทรัพยากรของตัวเองไป 1

Successor Function กำหนดเซตของการกระทำทั้งหมดที่เป็นไปได้

Terminal State ตัวกำหนดการสิ้นสุด สถานะสิ้นสุดเรียกว่า Terminal state ได้แก่ [0, 1, 2], [3, 4, 5], [6, 7, 8], [0, 3, 6], [1, 4, 7], [2, 5, 8], [0, 4, 8], [6, 4, 2]

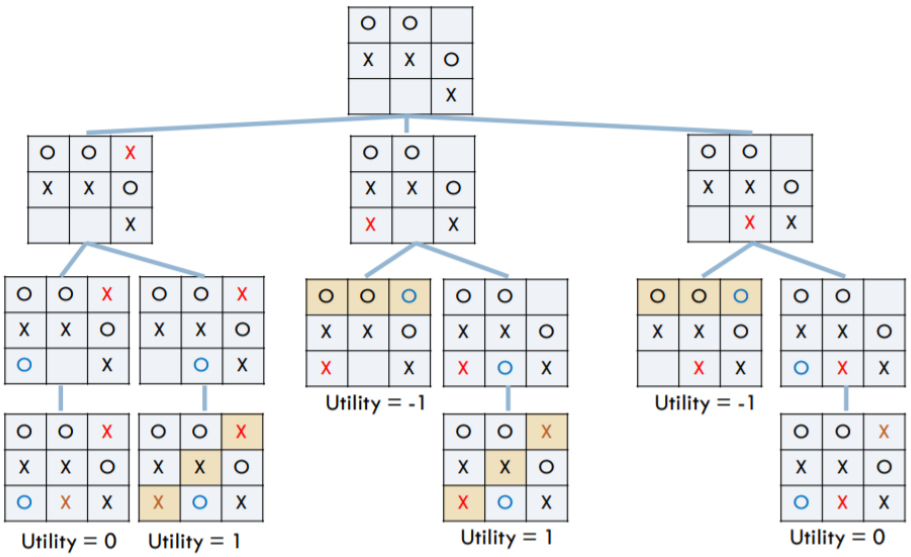
Utility Function ฟังก์ชันที่กำหนดค่าของ Terminal state เป็นตัวเลขบ่งบอกผลลัพธ์ของเกม เช่น ชนะ (+1) แพ้ (-1) เสมอ (0) ถ้าบางเกมมีการวัดผลที่ซับซ้อนอาจใช้ค่าได้



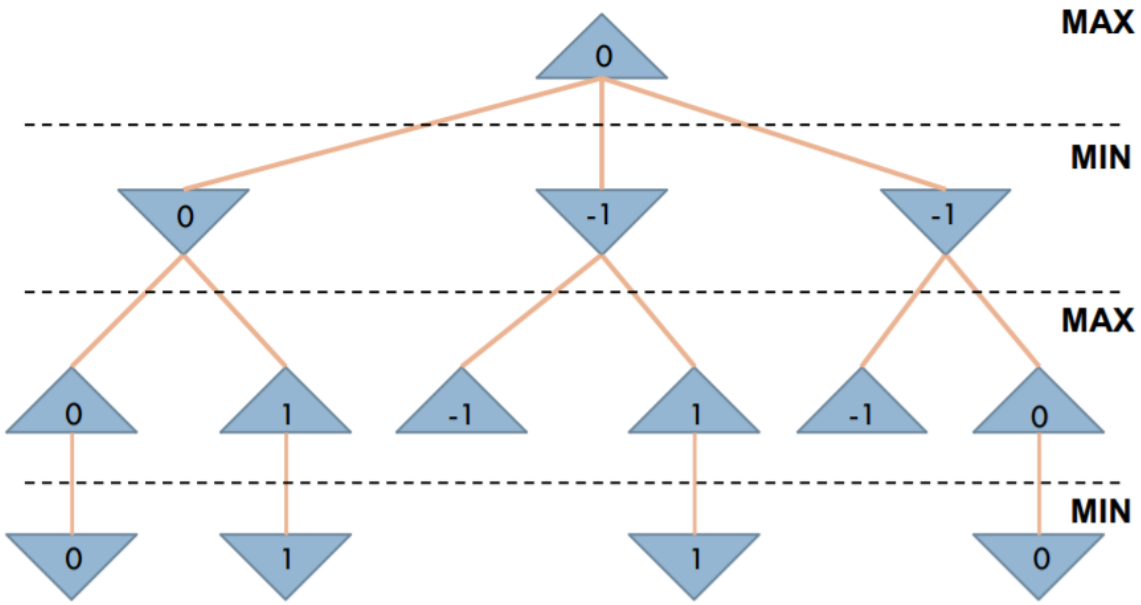
ภาพเปรียบเทียบ MAX(X) และ MIN(O)

อธิบายหลักการของอัลกอริทึม Minimax ที่ประยุกต์ใช้กับแกม tic-tac-toe

กำหนด X คือเราเป็นโหนด Max และ O คือฝ่ายตรงข้ามเป็นโหนด Min โดยที่กำหนดค่า Utility Function ของผู้เล่น X (เรา) แพ้มีค่าเป็น -1, ถ้าผู้เล่น X ชนะมีค่าเป็น 1 และค่าเสมอกันมีค่าเป็น 0



ภาพที่ **Terminal State** กับ **Utility Function**



ภาพ Utility Function ที่เป็นไปได้

เมื่อทำการคำนวณของ MinimaxValue ประจำโหนดต่างๆ ภายในต้นไม้ Minimax จะทำ ให้ได้ต้นไม้ดังภาพข้างบน และจะพบว่าค่า MinimaxValue ของสถานะปัจจุบันคือ 0 และเส้นทางที่จะทำให้ได้ค่านี้คือการเลือกเดินทางไปทางโหนดด้านซ้าย ซึ่งหมายความว่าถ้าเลือกเดินทางด้านซ้ายสถานการแย่ที่สุดก็คือเสมอ แต่ถ้าหากเลือกเดินทางไปยังโหนดกลาง หรือโหนดด้านขวา ค่า MinimaxValue ของสถานะคือ -1 สถานการแย่สุดก็คือแพ้

ขั้นตอนการสร้างเกม tic-tac-toe ด้วยถาษา HTML CSS และ Javascript

1. **วางโครงสร้างของตารางเกม tic-tac-toe ด้วยภาษา HTML**

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8" />
    <title>Tic Tac Toe</title>
    <link rel="stylesheet" href="style.css" />
</head>

<body>
    <!-- ตารางเกม มี 9 state จัดรูปแบบ 3x3 -->
    <table>
        <tr>
```

```
            <td class="cell" id="0"></td>
            <td class="cell" id="1"></td>
            <td class="cell" id="2"></td>
        </tr>
        <tr>
            <td class="cell" id="3"></td>
            <td class="cell" id="4"></td>
            <td class="cell" id="5"></td>
        </tr>
        <tr>
            <td class="cell" id="6"></td>
            <td class="cell" id="7"></td>
            <td class="cell" id="8"></td>
        </tr>
    </table>
    <!-- แสดงข้อความเมื่อถึง Terminal State หรือ Game over -->
    <div class="endgame">
        <div class="text"></div>
    </div>
    <!-- ปุ่ม Reset เกม -->
    <button onClick="startGame()">Replay</button>
    <script src="script.js"></script>
</body>
</html>
```

2. ตกแต่งด้วย CSS

```
td {
    border:  2px solid #333;
    height:  100px;
    width:  100px;
    text-align:  center;
    vertical-align:  middle;
    font-family:  "Comic Sans MS", cursive, sans-serif;
    font-size:  70px;
    cursor: pointer;
}
table {
    border-collapse: collapse;
    position: absolute;
    left: 50%;
    margin-left: -155px;
    top: 50px;
}
```

```
table tr:first-child td {
    border-top: 0;
}
table tr:last-child td {
    border-bottom: 0;
}
table tr td:first-child {
    border-left: 0;
}
table tr td:last-child {
    border-right: 0;
}
.endgame {
  display: none;
  width: 200px;
  top: 120px;
  background-color: rgba(205,133,63, 0.8);
  position: absolute;
  left: 50%;
  margin-left: -100px;
  padding-top: 50px;
  padding-bottom: 50px;
  text-align: center;
  border-radius: 5px;
  color: white;
  font-size: 2em;
}
```

3. **เขียนสคริปการทำงานของโปรแกรมด้วยภาษา JavaScript**

- ขั้นแรกกำหนด และประกาศค่าตัวแปรต่างๆ ที่จะใช้งานในโปรแกรม แล้วสั่งฟังก์ชั่นเริ่มเกม

```
var origBoard;
const huPlayer = "O";
const aiPlayer = "X";
const winCombos = [
  [0, 1, 2],
  [3, 4, 5],
  [6, 7, 8],
  [0, 3, 6],
  [1, 4, 7],
  [2, 5, 8],
  [0, 4, 8],
  [6, 4, 2],
];
const cells = document.querySelectorAll(".cell");
startGame();
```

- ฟังก์ชั่น startGame โดยการทำงานจะทำการเคลีย์ค่าทุกอย่างออกจาก State และ กำหนด State 9 ช่อง และเมื่อผู้เล่นทำการคลิกที่ State ก็จะไปทำฟังก์ชั่น trunClick

```
function startGame() {
  document.querySelector(".endgame").style.display = "none";
  origBoard = Array.from(Array(9).keys());
  for (var i = 0; i < cells.length; i++) {
    cells[i].innerText = "";
    cells[i].style.removeProperty("background-color");
    cells[i].addEventListener("click", turnClick, false);
  }
}
```

- ฟังก์ชั่น turnClick โดยการทำงานจะรับค่า State ของผู้เล่นที่เลือกส่งมา แล้วทำการเช็คว่าเป็นตำแหน่งที่เป็นเลขของ State หรือไม่ เมื่อเข้าเงื่อนไข ก็จะส่งเลข State และชื่อของผู้เล่นไปทำในฟังก์ชั่น trun
- ขั่นต่อมาก็ทำการเช็คว่าผู้เล่นชนะ หรือเสมอรึยัง หากไม่มีทั้ง 2 กรณีให้ไปทำคำสั่งในเงื่อนไข โดยให้เรียกใช้ฟังก์ชั่น turn เหมือนกัน แต่จะส่ง ค่าที่ได้จากฟังก์ชั่นที่ได้จาก ฟังก์ชั่น bestSpot และค่าที่สองเป็นผู้เล่น AI

```
function turnClick(square) {
  if (typeof origBoard[square.target.id] == "number") {
    turn(square.target.id, huPlayer);
    if (!checkWin(origBoard, huPlayer) && !checkTie())
      turn(bestSpot(), aiPlayer);
  }
}
```

- ฟังก์ชั่น turn โดยการทำงานของฟังก์ชั่นนี้จะทำการรับค่าที่ส่งมาแปลงตำแหน่งของ State ที่รับมาเป็นสัญลักษณ์ของผู้เล่นที่ส่งมา จากนั้นให้ทำการสร้างตัวแปร gamewon มาเพื่อนำไปรับค่าจาก checkWin หาก ฟังก์ชั่น checkWin เป็นจริง ให้ทำการเรียกใช้ฟังก์ชั่น gameOver ต่อไป

```
function turn(squareId, player) {
  origBoard[squareId] = player;
  document.getElementById(squareId).innerText = player;
  let gameWon = checkWin(origBoard, player);
  if (gameWon) gameOver(gameWon);
}
```

- ฟังก์ชั่น checkWin จะรับค่า ที่อยู่บน State มาทำการเช็คโดยแปลงบจากสัญลักษณ์ของแต่ละผู้เล่นเป็น เซต เพื่อนำมาตรวจสอบกับ Terminal State หากมีผู้ชนะก็จะรีเทิร์น gameWon กลับไป

```
function checkWin(board, player) {
  // reduce method a คือ ผลจากแต่ละรอบ e คือ ค่าของสมาชิก array ปัจจุบัน a คือ index ของ array ปัจจุบัน
  let plays = board.reduce((a, e, i) => (e === player ? a.concat(i) : a), []);
  let gameWon = null;
  for (let [index, win] of winCombos.entries()) {
    if (win.every((elem) => plays.indexOf(elem) > -1)) {
      gameWon = { index: index, player: player };
      break;
    }
  }
  return gameWon;
}
```

- ฟังก์ชั่น gameOver จะเปลี่ยนพื้นหลังที่เป็น Terminal State และเรียกใช้ฟังก์ชั่น declareWinner

```
function gameOver(gameWon) {
  for (let index of winCombos[gameWon.index]) {
    document.getElementById(index).style.backgroundColor =
      gameWon.player == huPlayer ? "blue" : "red";
  }
  for (var i = 0; i < cells.length; i++) {
    cells[i].removeEventListener("click", turnClick, false);
  }
  declareWinner(gameWon.player == huPlayer ? "You win!" : "You lose.");
}
```

- ฟังก์ชั่น declareWinner โดยจะทำการประกาศชื่อคนที่ชนะ จากค่าที่รับมาลงให้ Tag HTML

```
function declareWinner(who) {
  document.querySelector(".endgame").style.display = "block";
  document.querySelector(".endgame .text").innerText = who;
}
```

- ฟังก์ชั่น checkTie โดยจะทำการเรียกใช้ฟังก์ชั่น emptySquares ว่ามี State เหลือเท่าไหร่ หากเหลือ 0 ให้ทำการใส่สีทุก State เป็นสีเขียว และเรียกฟังก์ชั่น declareWinner เพื่อประกาศว่าเสมอ

```
function checkTie() {
  if (emptySquares().length == 0) {
    for (var i = 0; i < cells.length; i++) {
      cells[i].style.backgroundColor = "green";
      cells[i].removeEventListener("click", turnClick, false);
    }
    declareWinner("Tie Game!");
    return true;
  }
  return false;
}
```

- ฟังก์ชั่น emptySquares เป็นฟังก์ชั่นที่จะตรวจสอบว่า State ตอนนี้ว่างเท่าไหร่แล้ว และรีเทิร์น State ที่ว่างอยู่ทั้งหมดออกไป

```
function emptySquares() {
  return origBoard.filter((s) => typeof s == "number");
}
```

- ฟังก์ชั่น bestSpot จะเป็นการเรียกใช้ฟังก์ชั่น minimax เพื่อใช้อัลกอริทึม minimax ในการค้นหา State ที่ดีที่สุด แล้วรีเทิร์น State นั้นออกไป

```
function bestSpot() {
  return minimax(origBoard, aiPlayer).index;
}
```

- ฟังก์ชั่น minimax เป็นฟังก์ชั่นที่รับค่า State ปัจจุบันมาคำนวณหาเส้นทางที่ดีที่สุด โดยจะนำ State ที่ว่างมาวนลูปแล้วเก็บเป็นเส้นทางต่างๆ ทั้งหมด จากนั้นนำเส้นทางที่เป็นไปได้ทั้งหมดนั้นมาคำนวณหาเส้นทางที่ดีที่สุดตามหลักการของ minimax เมื่อได้ State ที่ดีที่สุดก็จะรีเทิร์นกลับไปเป็นคำตอบของ AI

```
function minimax(newBoard, player) {
  var availSpots = emptySquares();
// เป็นการเช็ค Utility Function ของ Terminal State ว่าได้กี่คะแนน
  if (checkWin(newBoard, huPlayer)) {
    return { score: -10 };
  } else if (checkWin(newBoard, aiPlayer)) {
    return { score: 10 };
  } else if (availSpots.length === 0) {
    return { score: 0 };
  }
// เช็ค Successor Function หรือความเป็นไปได้ของการกระทำทั้งหมด
  var moves = [];
  for (var i = 0; i < availSpots.length; i++) {
    var move = {};
    move.index = newBoard[availSpots[i]];
    newBoard[availSpots[i]] = player;

    if (player == aiPlayer) {
      var result = minimax(newBoard, huPlayer);
      move.score = result.score;
    } else {
```

```
      var result = minimax(newBoard, aiPlayer);
      move.score = result.score;
    }

    newBoard[availSpots[i]] = move.index;

    moves.push(move);
  }
  var bestMove;
  // เป็นการเช็คคะแนนจาก Successor Function หรือความเป็นไปได้ของการกระทำทั้งหมด
  // ว่าการเคลื่อนไหวในเซตไหนของฝั่ง AI มี score สูงสุด และฝั่งผู้เล่นมี scroe ต่ำสุด
  // จากนั้นให้ทำการรีเทิร์น index ที่มี score สูงสุดออกไป
  if (player === aiPlayer) {
    var bestScore = -10000;
    for (var i = 0; i < moves.length; i++) {
      if (moves[i].score > bestScore) {
        bestScore = moves[i].score;
        bestMove = i;
      }
    }
  } else {
    var bestScore = 10000;
    for (var i = 0; i < moves.length; i++) {
      if (moves[i].score < bestScore) {
        bestScore = moves[i].score;
        bestMove = i;
      }
    }
  }
  return moves[bestMove];
}
```

- Source code ทั้งหมดของ JavaScript

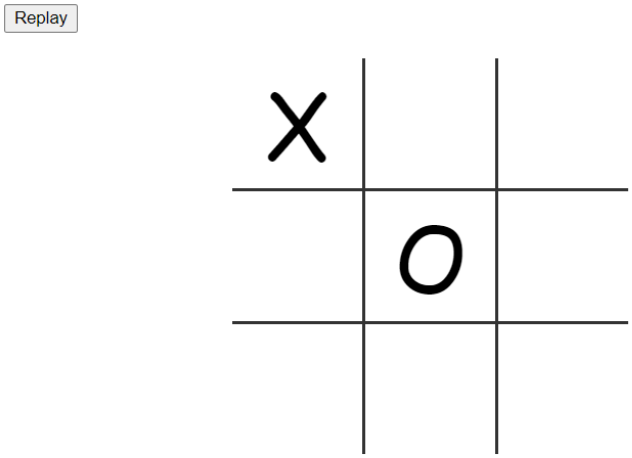
```
var origBoard;
const huPlayer = "O";
const aiPlayer = "X";
const winCombos = [
  [0, 1, 2],
  [3, 4, 5],
  [6, 7, 8],
  [0, 3, 6],
  [1, 4, 7],
  [2, 5, 8],
```

```
  [0, 4, 8],
  [6, 4, 2],
];
const cells = document.querySelectorAll(".cell");
startGame();

function startGame() {
  document.querySelector(".endgame").style.display = "none";
  origBoard = Array.from(Array(9).keys());
  for (var i = 0; i < cells.length; i++) {
    cells[i].innerText = "";
    cells[i].style.removeProperty("background-color");
    cells[i].addEventListener("click", turnClick, false);
  }
}

function turnClick(square) {
  if (typeof origBoard[square.target.id] == "number") {
    turn(square.target.id, huPlayer);
    if (!checkWin(origBoard, huPlayer) && !checkTie())
      turn(bestSpot(), aiPlayer);
  }
}

function turn(squareId, player) {
  origBoard[squareId] = player;
  document.getElementById(squareId).innerText = player;
  let gameWon = checkWin(origBoard, player);
  if (gameWon) gameOver(gameWon);
}

function checkWin(board, player) {
  // reduce method a คือ ผลจากแต่ละรอบ e คือ ค่าของสมาชิก array ปัจจุบัน a คือ index ของ array ปัจจุบัน
  let plays = board.reduce((a, e, i) => (e === player ? a.concat(i) : a), []);
  let gameWon = null;
  for (let [index, win] of winCombos.entries()) {
    if (win.every((elem) => plays.indexOf(elem) > -1)) {
      gameWon = { index: index, player: player };
      break;
    }
  }
  return gameWon;
}

function gameOver(gameWon) {
```

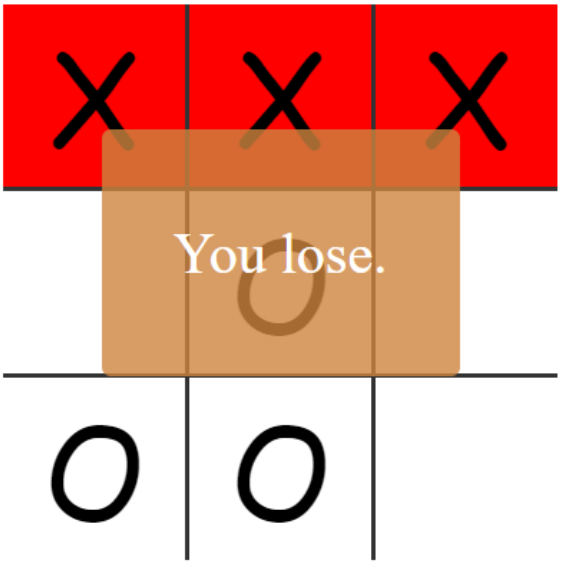
```
  for (let index of winCombos[gameWon.index]) {
    document.getElementById(index).style.backgroundColor =
      gameWon.player == huPlayer ? "blue" : "red";
  }
  for (var i = 0; i < cells.length; i++) {
    cells[i].removeEventListener("click", turnClick, false);
  }
  declareWinner(gameWon.player == huPlayer ? "You win!" : "You lose.");
}

function declareWinner(who) {
  document.querySelector(".endgame").style.display = "block";
  document.querySelector(".endgame .text").innerText = who;
}

function checkTie() {
  if (emptySquares().length == 0) {
    for (var i = 0; i < cells.length; i++) {
      cells[i].style.backgroundColor = "green";
      cells[i].removeEventListener("click", turnClick, false);
    }
    declareWinner("Tie Game!");
    return true;
  }
  return false;
}

function emptySquares() {
  return origBoard.filter((s) => typeof s == "number");
}

function bestSpot() {
  return minimax(origBoard, aiPlayer).index;
}

function minimax(newBoard, player) {
  var availSpots = emptySquares();
// เป็นการเช็ค Utility Function ของ Terminal State ว่าได้กี่คะแนน
  if (checkWin(newBoard, huPlayer)) {
    return { score: -10 };
  } else if (checkWin(newBoard, aiPlayer)) {
    return { score: 10 };
  } else if (availSpots.length === 0) {
    return { score: 0 };
  }
```

```
// เช็ค Successor Function หรือความเป็นไปได้ของการกระทำทั้งหมด
  var moves = [];
  for (var i = 0; i < availSpots.length; i++) {
    var move = {};
    move.index = newBoard[availSpots[i]];
    newBoard[availSpots[i]] = player;

    if (player == aiPlayer) {
      var result = minimax(newBoard, huPlayer);
      move.score = result.score;
    } else {
      var result = minimax(newBoard, aiPlayer);
      move.score = result.score;
    }

    newBoard[availSpots[i]] = move.index;

    moves.push(move);
  }
  var bestMove;
  // เป็นการเช็คคะแนนจาก Successor Function หรือความเป็นไปได้ของการกระทำทั้งหมด
  // ว่าการเคลื่อนไหวในเซตไหนของฝั่ง AI มี score สูงสุด และฝั่งผู้เล่นมี scroe ต่ำสุด
  // จากนั้นให้ทำการรีเทิร์น index ที่มี score สูงสุดออกไป
  if (player === aiPlayer) {
    var bestScore = -10000;
    for (var i = 0; i < moves.length; i++) {
      if (moves[i].score > bestScore) {
        bestScore = moves[i].score;
        bestMove = i;
      }
    }
  } else {
    var bestScore = 10000;
    for (var i = 0; i < moves.length; i++) {
      if (moves[i].score < bestScore) {
        bestScore = moves[i].score;
        bestMove = i;
      }
    }
  }
  return moves[bestMove];
}
```

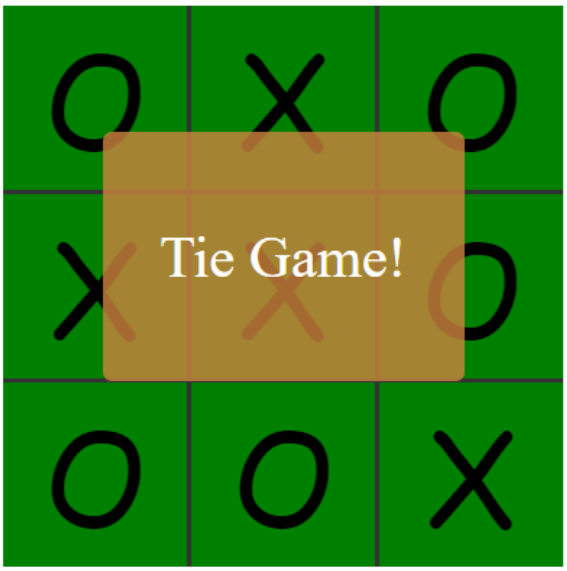
หน้าตาของโปรแกรม



ภาพ ตารางของเกม ที่เป็น State ทั้ง 9



ภาพ การชนะของ AI Utility Function มีค่าเป็น +1



ภาพ เสมอ Utility Function มีค่าเป็น 0

บรรณานุกรม

ผู้ช่วยศาตราจารย์ ดร.ชูพันธุ์ รัตนโภคา. (2559). ความรู้เบื้องต้นทางปัญญาประดิษฐ (เอกสารคำสอน). มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ