CORE TEAM

씨애랑 동아리 소프트웨어 전시회

|

소프트웨어융합대학

-20205206 운석현

-20205146 김재형

-20205209 유수경

CHAPTER 01

역할 분담과 주제 선정 이유

역할 분담

- 운석현 - 주제선정 및 ppt 제작, 자료 조사
- 김재형 – 코드 분석 및 오류 수정
- 유수경 – 자료조사, 코드 분석, ppt제작

주제 선정 이유

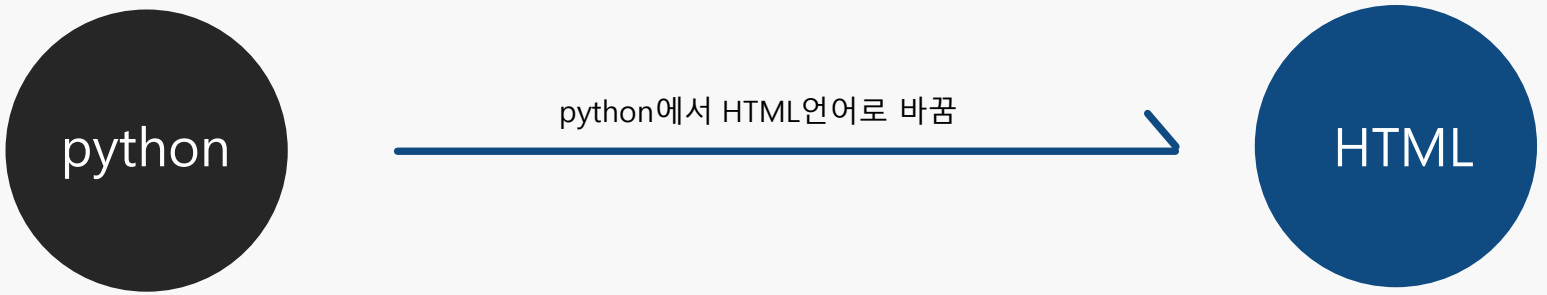
휴대폰 오목게임과 같이 컴퓨터로도 쉽게 접근 할 수 있는 컴퓨터 오목게임을 만들고 싶었다.
또한 인공지능과 오목을 두며 실력을 기르는 것을 목표로 하여 주제를 선정했다.

CHAPTER 02

프로젝트를 위해 사용한 언어.

02

무슨 언어를 사용하여 제작했는가.



처음에는 python언어와 c++언어를 사용해서 오목을 제작하려 했으나 소스들을 찾고 그 소스들을 공부하고 분석하는 과정이 너무 어려웠다. 또한 추가적인 자료가 부족했기에, 쉽게 접근 할 수 있는 HTML언어를 선택해 오목 프로그램을 만들었다.

CHAPTER 03

프로젝트 설명과 특징.

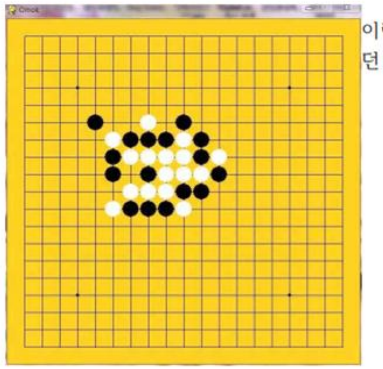
프로젝트 설명과 특징

주제
-오목 시스템

플랫폼
-데스크톱

개발 도구 및 기술
-파이썬(python)

설명
-데스크톱을 이용해서 누구나 쉽게 이용할 수 있는 오목 시스템입니다.
평소 우리가 즐겨하던 오목의 규칙 등을 그대로 적용해 오목을 해봤던 사람이라면 누구든지 쉽게 즐길 수 있고 이용할 수 있습니다.



이런 식의 인터페이스를 적용해서 우리가 알던 오목 게임의 모습을 만들 것입니다.

초반 계획서의 모습과 실제로 완성된 뒤의 모습.

CHAPTER 04

프로젝트를 하면서 어려웠던 점.

04

프로젝트를 하면서 어려웠던 점.

코로나로 인해 팀원들을 만나 활동을 하는 것에 있어서 많은 제약을 받았음.

처음 파이썬으로 프로그램을 구현 하려 했을 때 관련 자료가 없어 시간 낭비를 많이 했었음.

또한 자료와 오픈 소스들을 가져와서 실행해 봐도 오류가 나오는 등 여러 문제들이 발생해서 어려움을 겪음.

CHAPTER 05

프로젝트 설명

프로그래밍 중 핵심 설명

```
class Gomoku {



}
```

05-1

```
line = 15;

board = [
  new Array("", "", "", "", "", "", "", "", "", "", "", "", "", "", ""),
  new Array("", "", "", "", "", "", "", "", "", "", "", "", "", "", ""),
  new Array("", "", "", "", "", "", "", "", "", "", "", "", "", "", ""),
  new Array("", "", "", "", "", "", "", "", "", "", "", "", "", "", ""),
  new Array("", "", "", "", "", "", "", "", "", "", "", "", "", "", ""),
  new Array("", "", "", "", "", "", "", "", "", "", "", "", "", "", ""),
  new Array("", "", "", "", "", "", "", "", "", "", "", "", "", "", ""),
  new Array("", "", "", "", "", "", "", "", "", "", "", "", "", "", ""),
  new Array("", "", "", "", "", "", "", "", "", "", "", "", "", "", ""),
  new Array("", "", "", "", "", "", "", "", "", "", "", "", "", "", ""),
  new Array("", "", "", "", "", "", "", "", "", "", "", "", "", "", ""),
  new Array("", "", "", "", "", "", "", "", "", "", "", "", "", "", ""),
  new Array("", "", "", "", "", "", "", "", "", "", "", "", "", "", ""),
  new Array("", "", "", "", "", "", "", "", "", "", "", "", "", "", ""),
  new Array("", "", "", "", "", "", "", "", "", "", "", "", "", "", "")
];
```

줄 수 설정

게임판(오목판) 설정

```
player = "●";

turn({
  row,
  col
}) {

  if (this.board[row][col] != "") {
    swal({
      icon: "error",
      title: "돌 위에 돌을 둘 수 없습니다.",
      button: "다시두기"
    });
    return;
  } else {
```

현재 플레이어

턴 넘기기

만약 바둑판에 돌이 있을 경우 그 위에 돌을 또 두지 못하게 경고 띄우기

```
    this.board[row][col] = this.player;
  }

  this.player = this.player === "●" ? "○" : "●";

  document.querySelector(".currentTurn").textContent =
    "Current Turn : " + this.player;
}
```

돌이 없을 경우 현재 플레이어에 대한 표시를 게임판의 해당 위치에 넣어주기

현재 플레이어 변경,
현재 플레이어가 ●이면 다음 플레이어가 ○,
현재 플레이어가 ●이 아니면
다음 플레이어는 ●이다.

현재 차례가 누구인지
알려주는 역할을 한다.

```
checkWinner() {




                    }
```

```
for (let c = 2; c < this.line - 2; c++) {
      for (let r = 0; r < this.line; r++) {
    if (
      this.board[r][c - 2] == "●" &&
      this.board[r][c - 1] == "●" &&
      this.board[r][c] == "●" &&
      this.board[r][c + 1] == "●" &&
      this.board[r][c + 2] == "●"
    ) {
      return this.board[r][c];
      break;
    } else if (
      this.board[c - 2][r] == "●" &&
      this.board[c - 1][r] == "●" &&
      this.board[c][r] == "●" &&
      this.board[c + 1][r] == "●" &&
      this.board[c + 2][r] == "●"
    ) {
      return this.board[c][r];
      break;
    }
  }
}
```

흑돌이 가로에 5개 연속으로
놓여질 때 승리.

```
//흰돌도 마찬가지로
this.board[r][c - 2] == "○" &&
this.board[r][c - 1] == "○" &&
  this.board[r][c] == "○" &&
this.board[r][c + 1] == "○" &&
  this.board[r][c + 2] == "○"
```

흑돌이 세로에 5개 연속으로
놓여질 때 승리.

```
//흰돌
this.board[c - 2][r] == "○" &&
this.board[c - 1][r] == "○" &&
this.board[c][r] == "○" &&
this.board[c + 1][r] == "○" &&
this.board[c + 2][r] == "○"
```

```
for (let c = 2; c < this.line - 2; c++) {
      for (let r = 2; r < this.line - 2; r++) {
        if (
          this.board[r - 2][c - 2] == "●" &&
          this.board[r - 1][c - 1] == "●" &&
          this.board[r][c] == "●" &&
          this.board[r + 1][c + 1] == "●" &&
          this.board[r + 2][c + 2] == "●"
        ) {
          return this.board[r][c];
          break;
        }
        if (
          this.board[r + 2][c - 2] == "●" &&
          this.board[r + 1][c - 1] == "●" &&
          this.board[r][c] == "●" &&
          this.board[r - 1][c + 1] == "●" &&
          this.board[r - 2][c + 2] == "●"
        ) {
          return this.board[r][c];
          break;
        }
      }
    }
```

흑돌이 대각선 (좌 -> 우)
연속으로 5개가 놓여질 때 승리.

```
this.board[r - 2][c - 2] == "○" &&
this.board[r - 1][c - 1] == "○" &&
this.board[r][c] == "○" &&
this.board[r + 1][c + 1] == "○" &&
this.board[r + 2][c + 2] == "○"
```

흑돌이 대각선 (우 -> 좌)
연속으로 5개가 놓여질 때 승리.

```
this.board[r + 2][c - 2] == "○" &&
this.board[r + 1][c - 1] == "○" &&
this.board[r][c] == "○" &&
this.board[r - 1][c + 1] == "○" &&
this.board[r - 2][c + 2] == "○"
```

```
const omok = new Gomoku();

const rowEls = document.querySelectorAll(".board__row");
rowEls.forEach((rowEl, rowIndex) => {
  const colEls = rowEl.querySelectorAll(".board__col");
  colEls.forEach((colEls, colIndex) => {
    // 각각의 col(오목판의 빈칸)을 클릭할 경우 이벤트 발생
    colEls.addEventListener("click", e => {

      omok.turn({

        // 이때 인자로 row는 rowIndex를 col은 colIndex를 넘긴다.
        row: rowIndex,
        col: colIndex
      });
      draw();
    });
  });
});
```

생성자 생성하기

정의해놓은 turn이라는 메소드 사용

화면을 그려낼 draw라는 메소드 사용

```
function draw() {

  omok.board.forEach((rowArr, rowIndex) => {
    const rowEl = rowEls[rowIndex];
    const colEls = rowEl.querySelectorAll(".col__grid");

    rowArr.forEach((col, colIndex) => {

      if (col == "●") {
        colEls[colIndex].classList.add("black");

      } else if (col == "○") {
        colEls[colIndex].classList.add("white");
      }
    });
  });
```

오목판의 배열 순회

각각의 가로줄을 돌면서
세로열의 요소와 인덱스를 확인

만약 col이 "●"이면 클래스에
black 이라는 클래스를 더한다.

만약 col이 "○"이면 클래스에
white 라는 클래스를 더한다.

```
const winner = omok.checkWinner();

if (winner) {
  // 흑돌이 이겼을때,
  if (winner == "●") {
    document.querySelector(".currentTurn").textContent = "Game Over";

    swal({
      icon: "success",
      title: "축하드립니다! 흑돌(" + winner + ")의 승리입니다",
      button: "다시하기"
    }).then(value => {
      window.location.reload();
    });
```

승자 판별 실행

턴 수를 알려주는 창에서는 game over라는 텍스트를 띄움

객체 형식의 alert를 내보냄

버튼을 클릭했을 때 새로 고쳐서 다시 시작상태로 만든다

```
// 백돌이 이겼을 때, 흑돌이 이겼을 때와 같이 코드를 짜준다.
  } else if (winner == "○") {
    document.querySelector(".currentTurn").textContent = "Game Over";

    swal({
      icon: "success",
      title: "축하드립니다! 백돌(" + winner + ")의 승리입니다",
      button: "다시하기"
    }).then(value => {
      window.location.reload();
    });
  }

    // 버튼을 클릭하지 않을 경우에
    setTimeout(function () {
      // 10초동안 기다렸다가 새로 고쳐서 다시 시작상태로 만든다.
      window.location.reload();
    }, 10000);
  }
}
```

CHAPTER 06

프로젝트 후 소감

06

혼자 오목 게임을 할 수 있도록 인공지능과 함께하는 오목 게임이 목적이었지만 아직 머신러닝과 같은 심화 과목을 공부해보지 못했고, 어떤 프로그램으로 어떻게 구현해야 할지도 많은 고민을 했습니다. 준비과정에서 많은 어려움을 겪다가 인공지능과 사람이 아닌 사람과 사람이 둘 수 있는 오목 웹페이지를 구현하게 되었습니다. 전시회가 끝난 뒤에도 원래 목적이었던 인공지능과의 오목 두기 게임을 구현하기 위해 공부할 예정입니다.

Thank You :)