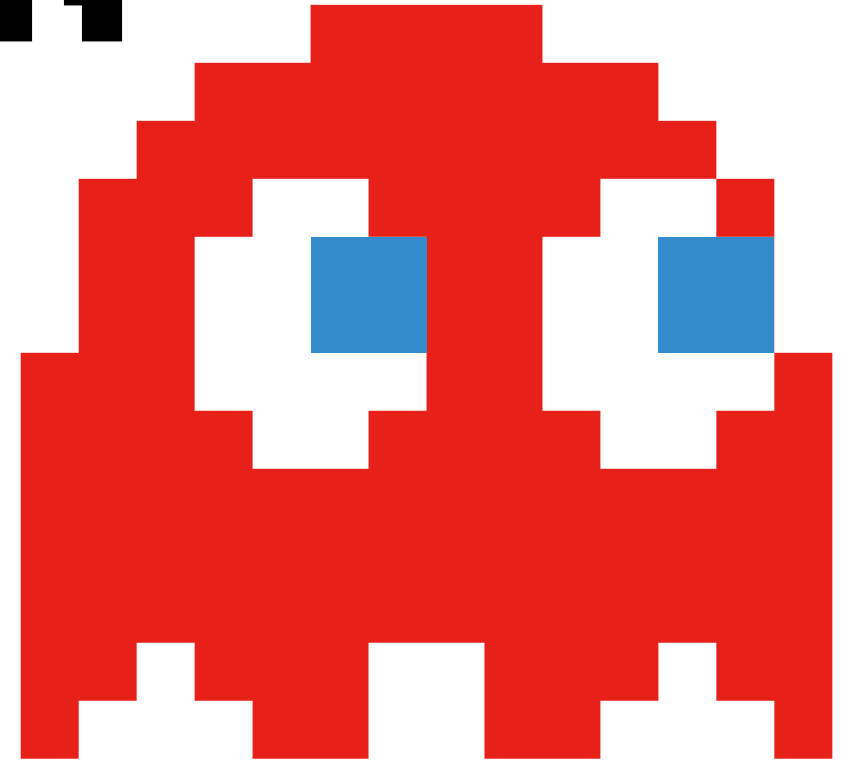


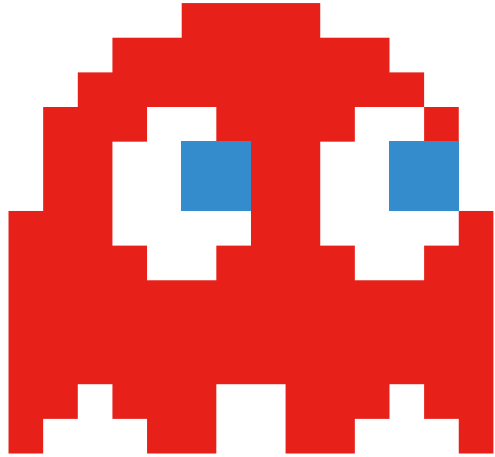


# PAC-MAN

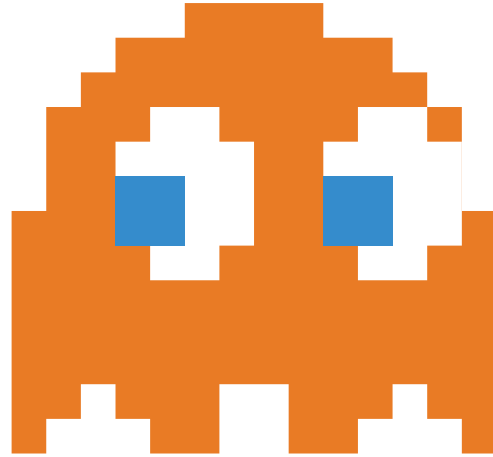
서동민



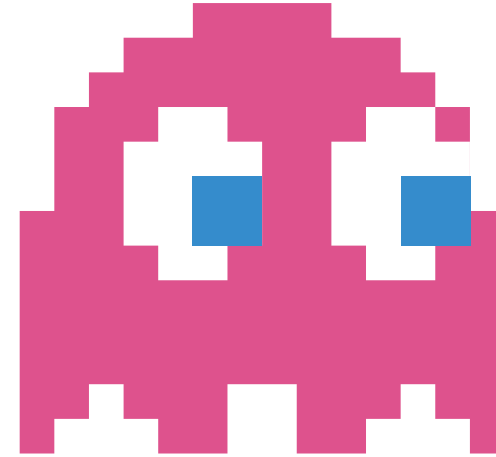
# INDEX



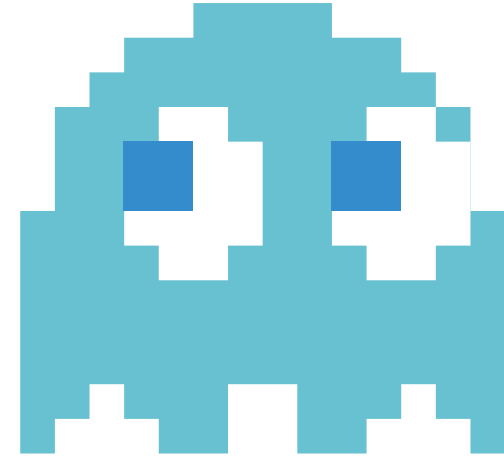
**Timeline**  
3p ~ 4p



**Design**  
5p ~ 9p



**Code**  
10p ~ 17p



**Review**  
18p ~ 19p



# Timeline

# Timeline



Design  
4-24 ~ 4-25

PUG  
4-26

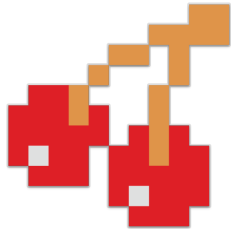
SCSS  
4-27

JavaScript  
4-28 ~ 5-2

Database  
5-3

Server  
5-4 ~ 5-5



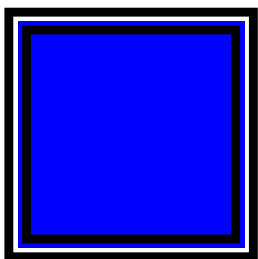
Design 

# Design



Press Start 2P.....

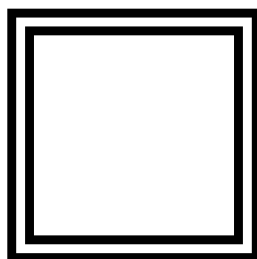
Standard font



#0000FF



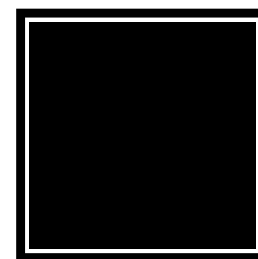
Wall  
color



#FFFFFF



Cookie  
color



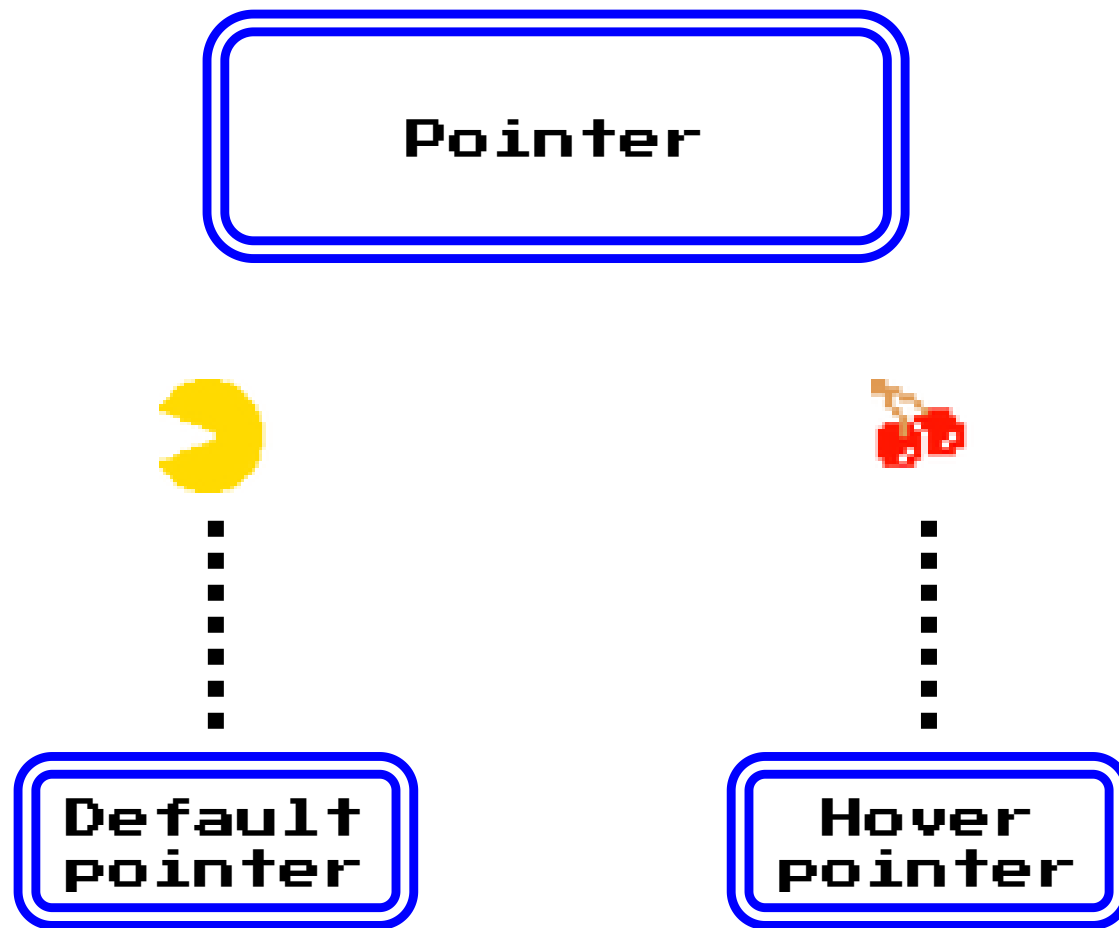
#000000



Road  
color

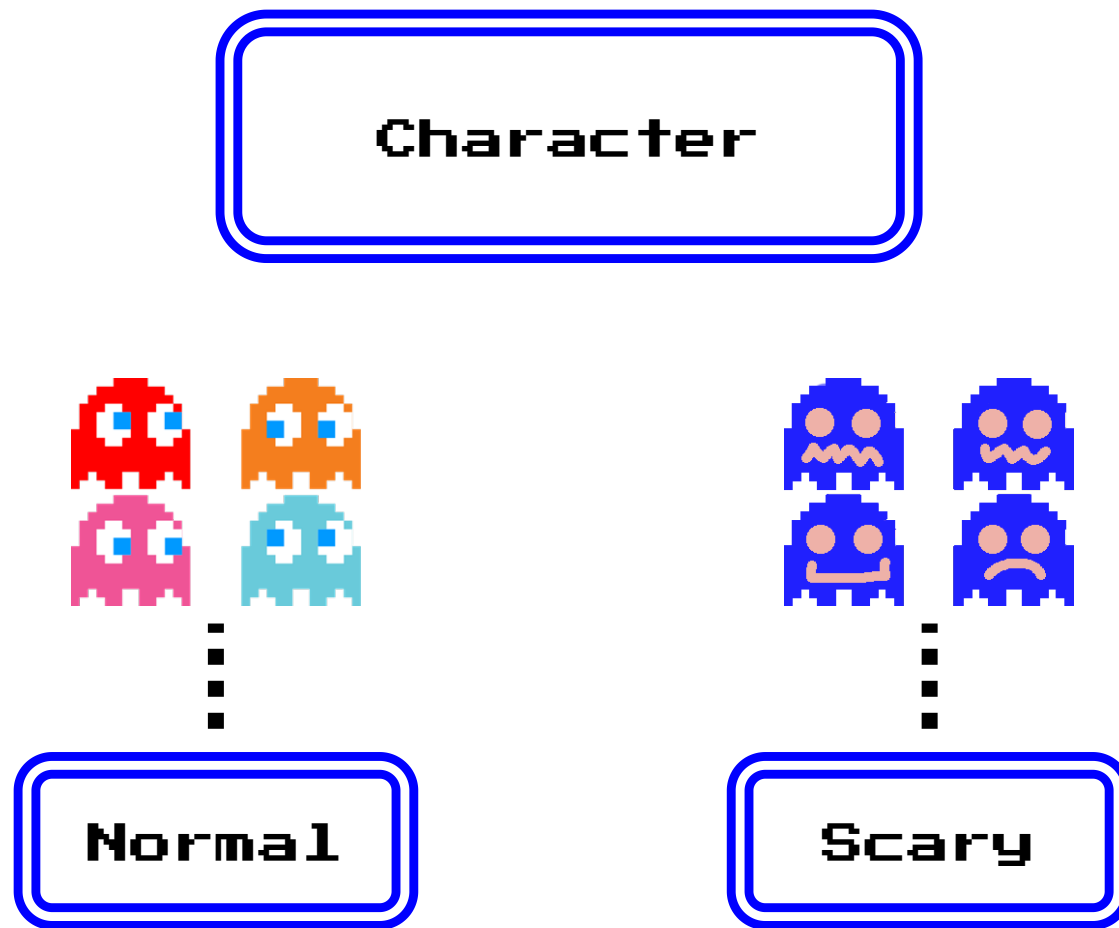
기존의 컨셉에서 가져온 색상과 픽셀이라는 분위기에 맞는 폰트를 사용

# Design



팩맨 게임에 맞는 포인터를 제공

# Design



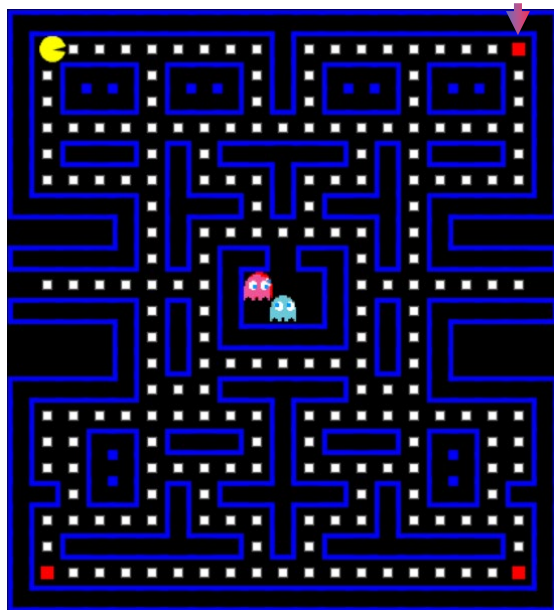
팩맨이 알약을 먹었을 때 변하는 귀신들의 모습



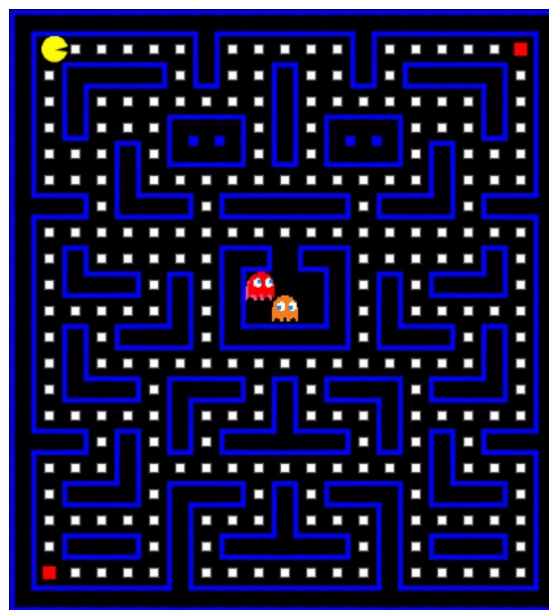
# Design



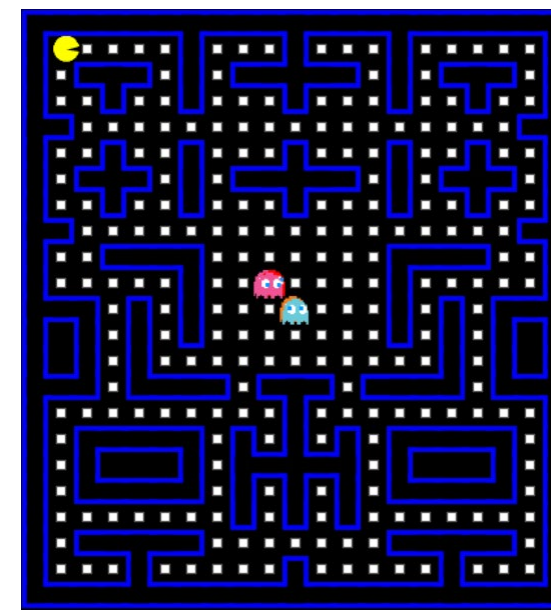
팩맨이 고스트를 잡아먹게 해주는 알약



LEVEL 1



LEVEL 2



LEVEL 3

Map  
design

다양한 맵을 제공하여 기존의 팩맨 게임과 다른 즐거움을 제공

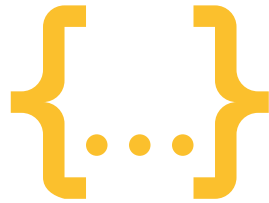


Code 

# Code



Programming  
Language



PUG, JavaScript, JSON, SCSS, Node.js MySQL를 사용한 코드 구현

# Code



```
1 const express = require("express");  
2 const mysql = require("mysql2");  
3 const expressApp = express();  
4 const dbConfig = require("../config/dbInfo.js");  
5 const mysqlConnection = mysql.createPool(dbConfig);  
6 const bodyParser = require("body-parser");
```

Express.js 웹 프레임워크

mysql2

mysql은 call back을 사용하지만  
mysql2는 promise를 사용

POST 방식을 위한 미들웨어



```
1 const mysqlPromise = mysqlConnection.promise();
```

promise 방식 mysql



```
1 await mysqlPromise.query(  
2   `INSERT INTO top_list (score, level) VALUES ('${saveScore}', '${saveLevel})`  
3 );  
4 let [result2] = await mysqlPromise.query(  
5   `SELECT score, level FROM top_list ORDER BY score DESC LIMIT 300`  
6 );
```

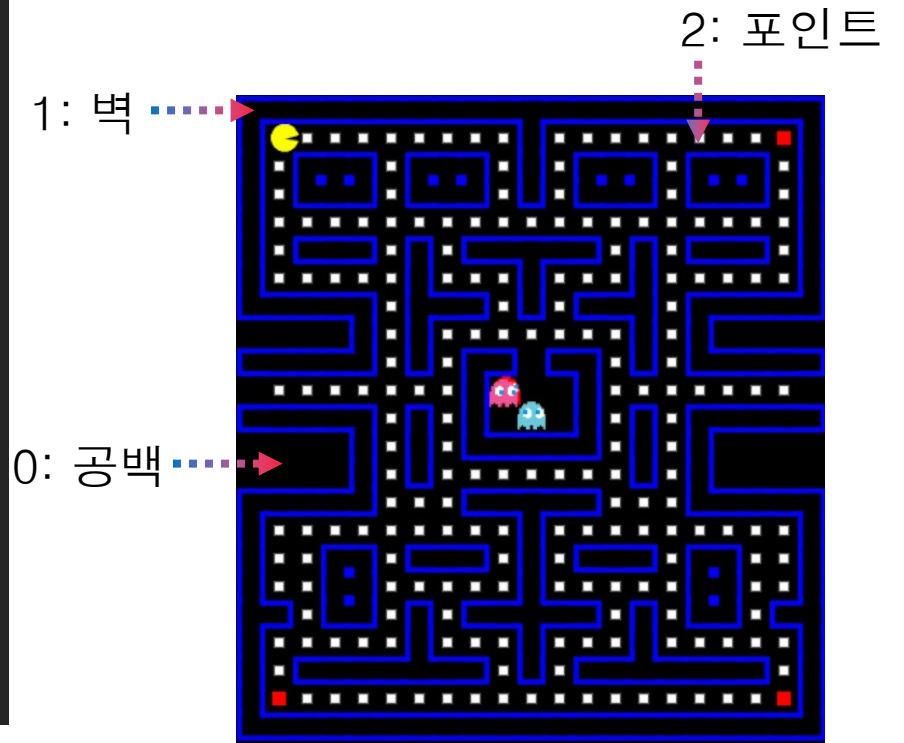
promise await 방식 사용

# Code



```
1  const mapLevel1 = [  
2    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],  
3    [1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1],  
4    [1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 2, 1, 1, 1, 2, 1, 1, 1, 2],  
5    [1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 2, 1, 1, 1, 2, 1, 1, 1, 2],  
6    [1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1],  
7    [1, 2, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 2],  
8    [1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 1],  
9    [1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1],  
10   [0, 0, 0, 0, 1, 2, 1, 2, 2, 2, 2, 2, 2, 1, 2, 1, 0, 0, 0, 0],  
11   [1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 0, 1, 1, 2, 1, 2, 1, 1, 1, 1],  
12   [1, 2, 2, 2, 2, 2, 2, 2, 1, 0, 0, 0, 1, 2, 2, 2, 2, 2, 2, 1],  
13   [1, 1, 1, 1, 1, 2, 1, 2, 1, 0, 0, 0, 1, 2, 1, 2, 1, 1, 1, 1],  
14   [0, 0, 0, 0, 1, 2, 1, 2, 1, 1, 1, 1, 1, 2, 1, 2, 1, 0, 0, 0],  
15   [0, 0, 0, 0, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 0, 0, 0],  
16   [1, 1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 1],  
17   [1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1],  
18   [1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 1, 2, 1, 1, 1, 2, 1, 1, 1, 2],  
19   [1, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 1],  
20   [1, 1, 2, 2, 1, 2, 1, 2, 1, 1, 1, 1, 1, 2, 1, 2, 1, 2, 2, 1],  
21   [1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 1],  
22   [1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2],  
23   [1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1],  
24   [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],  
25 ];
```

맵을 구현하기 위한 배열  
1은 벽, 2는 포인트, 0은 공백이다.



# Code



```
1  const onGhostCollision = () => {  
2    lives--;  
3    restartPacmanAndGhosts();  
4    if (lives === 0) {  
5      $("#deadScore").val(parseInt($("#deadScore").val()) + parseInt(score));  
6      $("#loseButton").trigger("click");  
7    }  
8  };
```

고스트와 충돌하는 경우  
Lives를 감소시키고

만약 lives가 0이라면  
점수 페이지로 이동된다.



```
1  window.addEventListener("keydown", (event) => {  
2    let keyCodeValue = event.key;  
3    setTimeout(() => {  
4      if (  
5        keyCodeValue === 37 ||  
6        keyCodeValue === "ArrowLeft" ||  
7        keyCodeValue === 65 ||  
8        keyCodeValue === "a"  
9      ) {  
10     pacman.nextDirection = DIRECTION_LEFT;
```

keyCode는 deprecated되었다.  
event.key를 통해 구현했다.

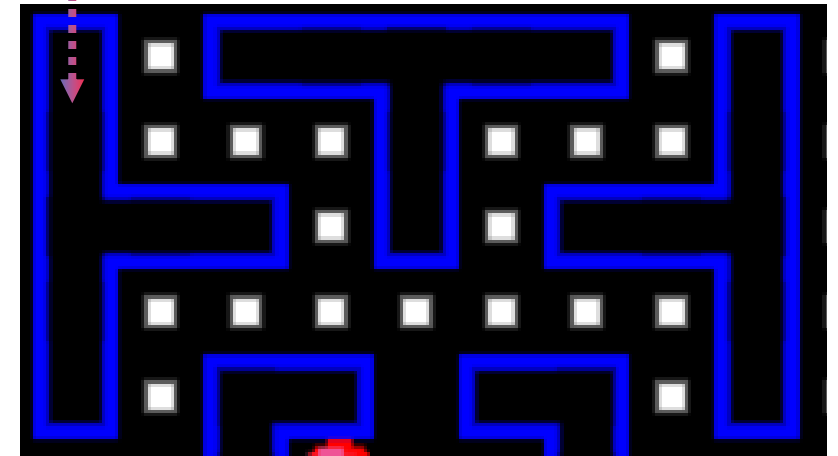
# Code



```
1  const drawWalls = () => {  
2    for (let i = 0; i < map.length; i++) {  
3      for (let j = 0; j < map[0].length; j++) {  
4        if (map[i][j] = 1) {  
5          createRect(  
6            j * oneBlockSize,  
7            i * oneBlockSize,  
8            oneBlockSize,  
9            oneBlockSize,  
10           "blue"  
11          );  
12          if (j > 0 && map[i][j - 1] = 1) {  
13            createRect(  
14              j * oneBlockSize,  
15              i * oneBlockSize + wallOffset,  
16              wallSpaceWidth + wallOffset,  
17              wallSpaceWidth,  
18              wallInnerColor  
19            );  
20          }  
21        }  
22      }  
23    }  
24  }
```

벽에 해당하는 부분의 주변에 벽을 의미하는 1이 존재하면 연결된 벽으로 보여주기 위한 코드

다른 벽과 붙어있는 부분은 선이 출력되지 않는다.



# Code



```
1  const secondArray = [];
2  let countNumber = 0;
3  countArray.forEach((v, i, a) => {
4    secondArray.push(
5      v.filter((v1) => {
6        return v1 === 2;
7      })
8    );
9  });
10 console.log(secondArray);
11 const thirdArray = [];
12 secondArray.forEach((v) => {
13   v.forEach((v1) => {
14     v1 === 2 && countNumber++;
15   });
16 });
17 console.log(countNumber);
```

배열에서 2만 추출하여  
총 포인트를 확인하기 위한 코드



# Code



```
1  checkCollision() {  
2      let returnValue = false;  
3      if (  
4          map[this.getMapY()][this.getMapX()] = 1 ||  
5          map[this.getMapYRightSide()][this.getMapX] = 1 ||  
6          map[this.getMapY()][this.getMapXRightSide()] = 1 ||  
7          map[this.getMapYRightSide()][this.getMapXRightSide()] = 1  
8      ) {  
9          returnValue = true;  
10     }  
11     return returnValue;  
12 }
```

벽이 있는지 없는지 확인하는 코드  
map[i][j]에서 i는 y축을 j는 x축을  
담당한다.

팩맨이 위로 이동하면 i는 -로  
아래로 이동하면 +가 된다.

팩맨이 좌로 이동하면 j는 -로  
우로 이동하면 +가 된다.

이런 원리를 사용해서 팩맨의 위치  
가 2나 3이면 이동할 수 있지만  
0이면 이동하려던 방향에서 뒤로  
한 칸 이동한다.



# Review



## 서동민

JavaScript 캔버스를 활용하여 Pacman 게임을 개발하는 것은 큰 도전이었다. Express와 MySQL2를 이용한 경험을 통해 Promise 문법에 익숙해질 수 있었지만, 캔버스를 다루는 능력은 아직 부족하다는 것을 느꼈다. 게임 개발에 있어 캔버스를 자유롭게 다루는 능력은 중요한 요소라고 생각했다. 하지만 나는 캔버스를 자유롭게 다루지 못했다. 따라서, 캔버스에 대한 이해를 더욱 향상시키기 위해 더 많은 공부와 연습이 필요하다고 느꼈다.

Express와 MySQL2를 활용하여 백엔드를 구현하는 과정에서 Promise 문법을 사용하여 비동기 작업을 처리하고 데이터베이스와의 상호작용을 효율적으로 다룰 수 있었다. Promise의 활용은 코드의 가독성과 유지보수성을 향상시키는 데 도움이 되었다고 느꼈다. Callback으로 이루어진 코드에서는 Callback 지옥에 빠지는 경우가 많았지만 Promise를 사용하면 그런 문제가 없었다.

Pacman 게임을 개발하며 얻은 경험은 도전적이고 흥미로운 시간을 선사해 주었다. 앞으로도 기회가 된다면 계속해서 성장하고, 캔버스를 효과적으로 활용하는 방법을 습득하여 더 멋진 게임을 만들어내고 싶다고 느꼈다. 이번 프로젝트를 통해 개발자로서의 성장을 경험할 수 있었고, 새로운 기술과 도구들을 접하고, 문제를 해결하고, 기능을 추가하면서 스스로의 역량을 높일 수 있었다. 그러나 여전히 부족한 부분이 많다고 느껴 좀 더 노력해야 할 필요성을 느꼈다. 지속적인 학습과 개발 과정에서의 도전을 통해 더 나은 개발자로 성장하기 위해 앞으로도 끊임없이 노력해야 된다고 느꼈고 공부를 할 수 록 더욱 편하고 깔끔하게 코드를 작성할 수 있다는 것을 느꼈다.