

MINI GAME ROOM

개발자 : 송재완, 김지향, 이예진





1. 개발환경
2. 게임소개
3. 주요기능
4. 향후계획

GAME LIST



BREAK-OUT



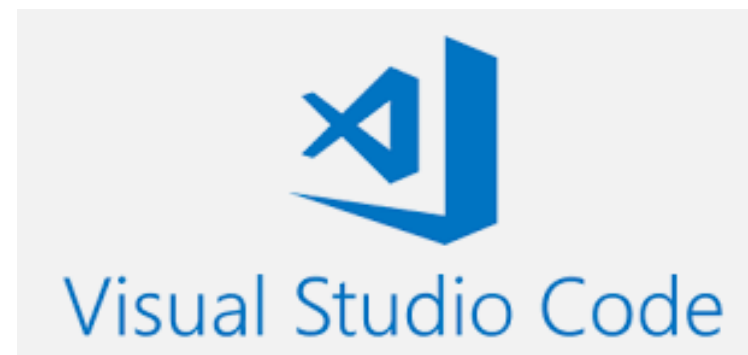
TETRIS



COOKIE-RUN



AIMHERO



Google
Developers

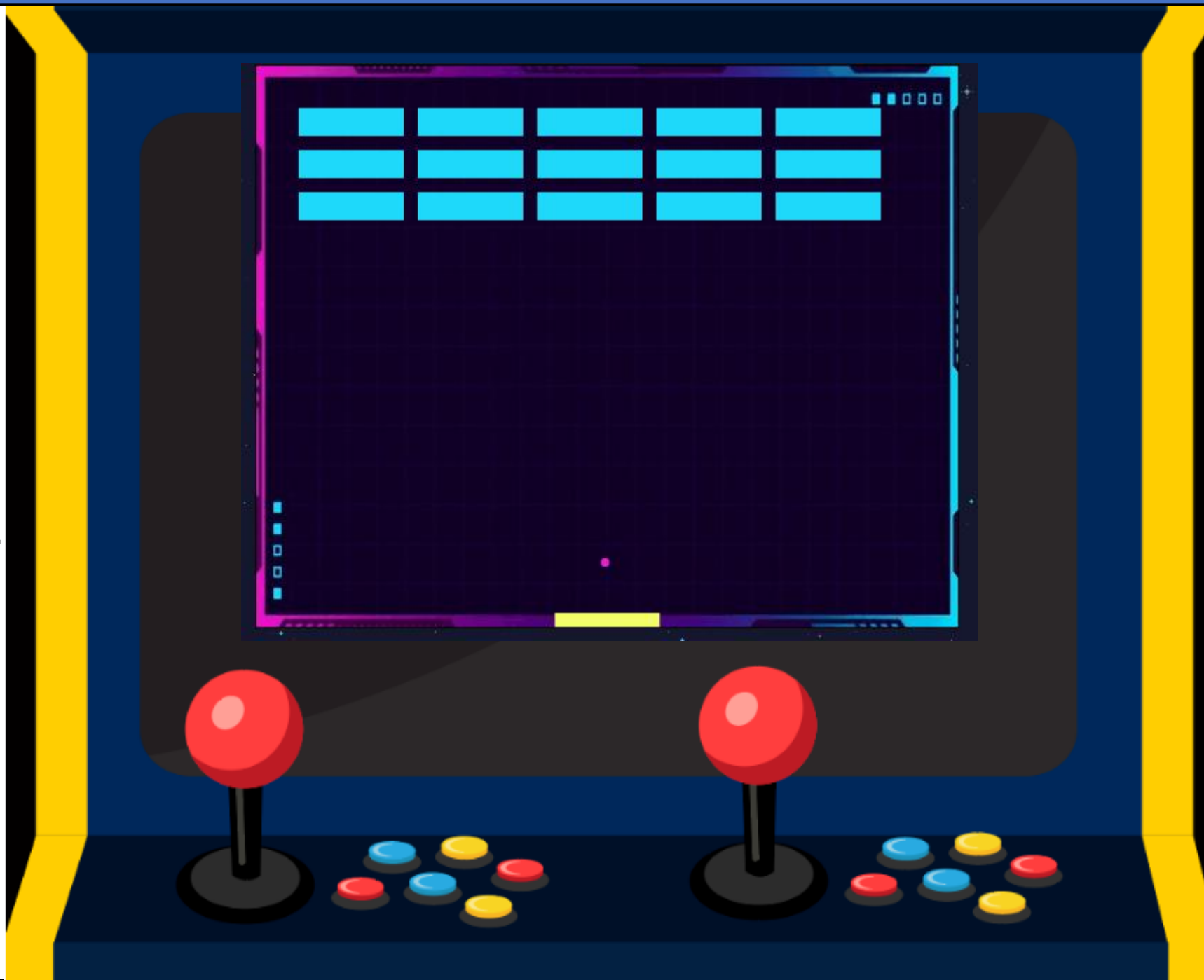
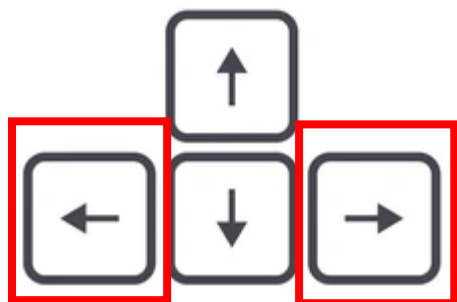




BREAK-OUT(김지향)

- 패들을 움직여 공을 튕겨 벽돌을 맞추는 게임
- 패들을 벗어나거나 벽돌을 다 깨면 게임 끝

<조작키>





1단계 :시각화

→ CANVAS 태그 사용

2단계 : 벽돌생성

→ 벽돌은 2차원배열 변수로 설계(각 벽돌마다 x, y 좌표 및 status 변수 포함)

그리기 함수(status가 1인상태만 그리기)

3단계 : 패들생성

→ 패들 그리기 함수, 이벤트 리스너 함수

* 방향키 조작 감지하는 플래그변수

4단계 : 공 생성

→ 공 그리기 함수





5단계 :충돌감지 함수 설계

→ 공이 좌, 우 벽에 충돌하면 x좌표의 방향을 반대로 전환
벽돌과 충돌하면 해당 벽돌의 status 값을 0,
cnt변수 값을 -1씩 저장

6단계 : 게임종료 함수 설계

→ 상황1 - 패들을 벗어난 범위에 공이 떨어지는 경우
상황2 - 모든 벽돌을 다 맞춘 경우(cnt변수 값으로 판단)

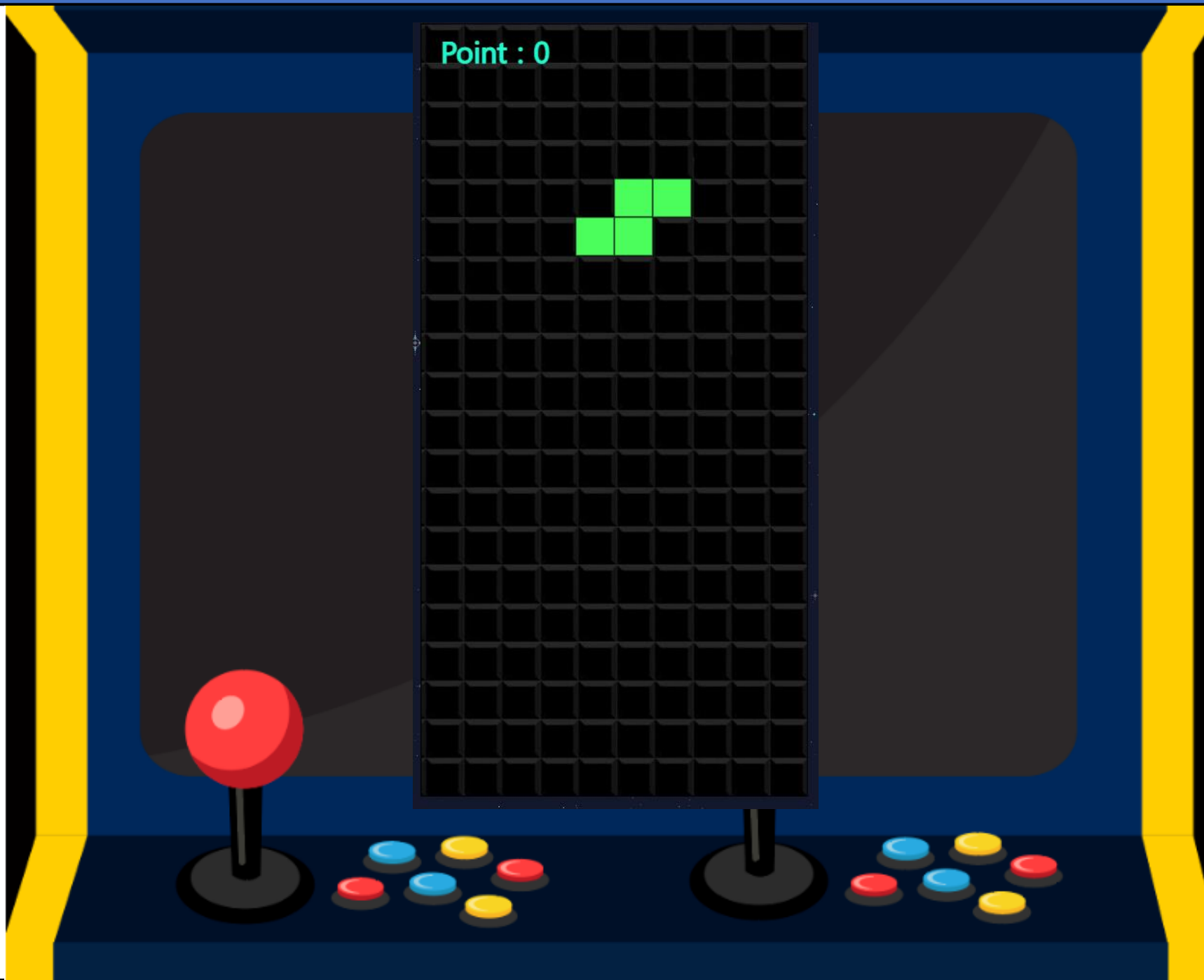
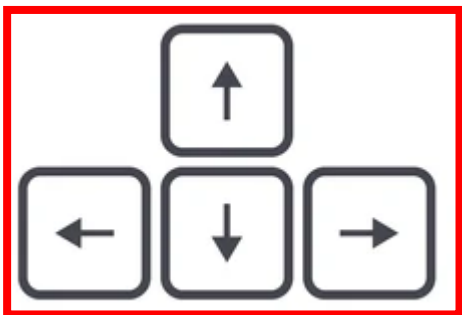




TETRIS(송재완)

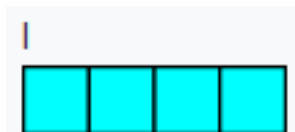
- 내려오는 도형을 모양을 바꾸며 맞추어 쌓는 게임
- 한줄(10칸)을 다 채우면 줄 삭제, 맨 위에 닿으면 게임 끝

<조작키>

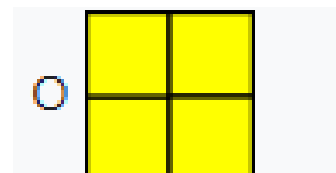




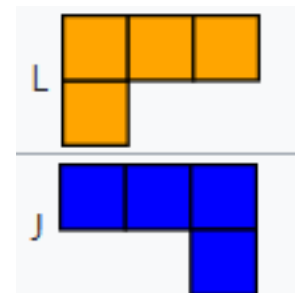
I TYPE



O TYPE



일반 TYPE



1단계 :시각화

→ CANVAS 태그 사용

2단계 : 도형생성

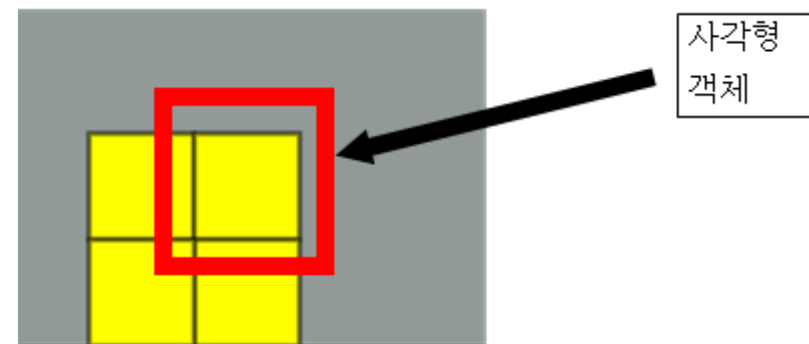
→ 7가지 타입도형(각각 4조각), 조각, 공통항목은 클래스로 설계

*조각 자원 : 조각 그리기 시작할 좌표 변수, 길이, 플래그 변수, 색 변수, 그리기 함수

*도형 자원 : 각 클래스는 공통항목 객체, 색 변수, 좌표지정함수 포함

(I & O타입과 일반타입 구분)

*공통항목 자원 : 조각객체 변수, 색 적용 함수, 도형 그리는 함수, 회전함수





3단계 : 게임실행

→ requestAnimationFrame() 함수 사용

4단계 : 충돌감지

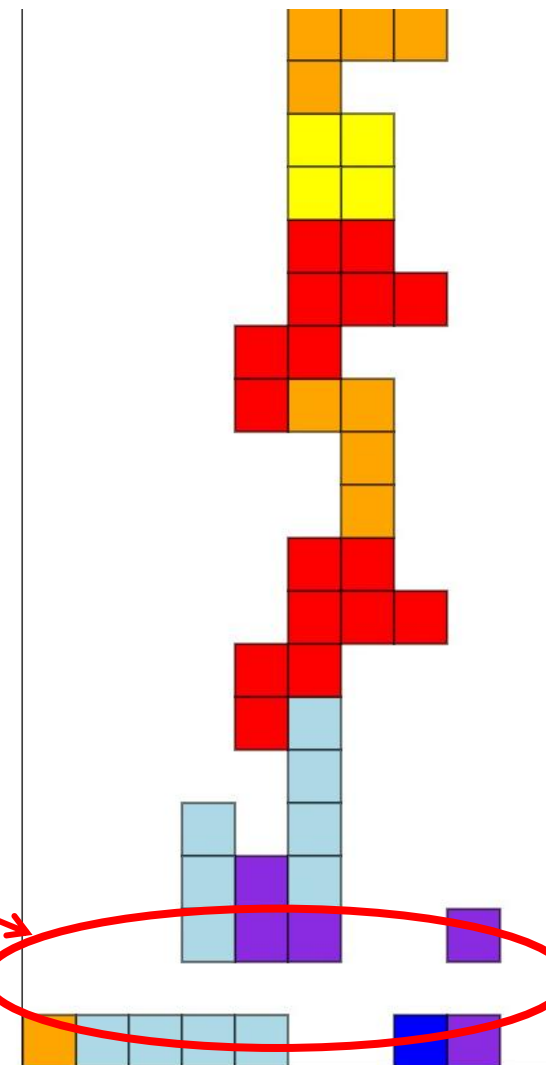
→ 객체와 객체의 범위가 겹치는 순간을 충돌로 판단

5단계 : 줄 삭제

→ 같은 y축 좌표를 갖는 조각객체가 10개가 되는 순간을 한줄 완성으로 판단하고 해당 줄의 조각 객체를 더 이상 그리지 않게 함수 설계

6단계 : 삭제된 줄 채우기

→ 지워진 y축 좌표보다 큰 위치에 있는 객체들을 전부 한칸씩 내려주는 함수 설계



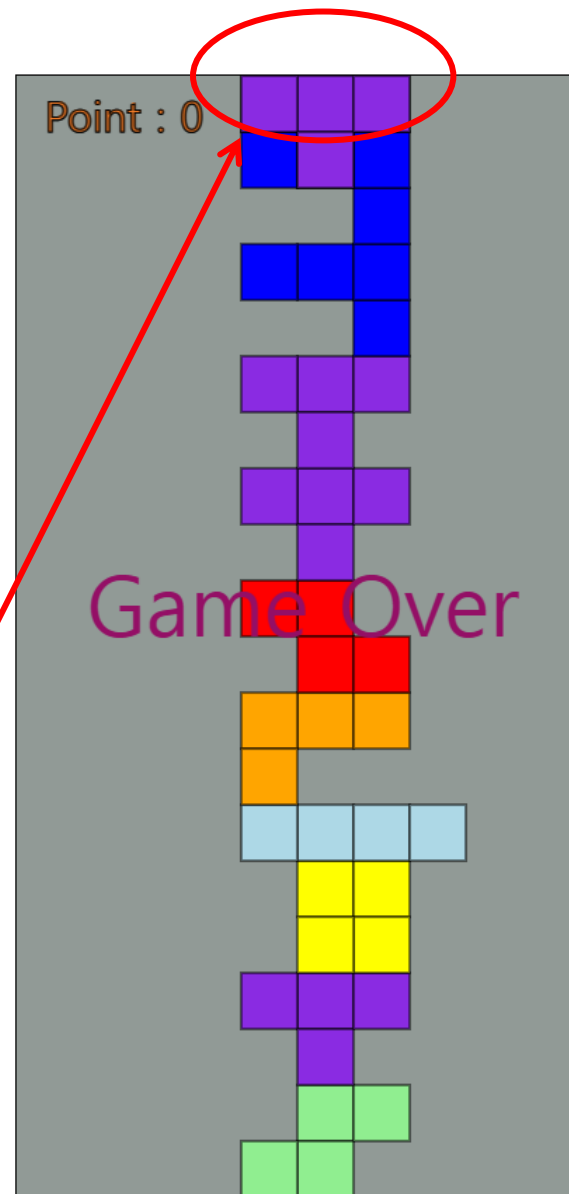


7단계 : 이벤트 리스너

→ 키로 입력 받아 이동하는 도형의 범위와 기존에 쌓인 도형의 범위의 충돌을 감지하는 하기 위해 함수를 설계하여 4단계(충돌감지) 함수에 인자값으로 넘겨줌

8단계 : 게임종료

→ 도형객체가 생성위치에서 충돌 감지되면 게임 종료로 판단하여 `cancelAnimationFrame()` 함수를 사용하여 반복하고 있던 함수 중지





COOKIE-RUN(송재완)

- 시각화 : CANVAS 태그 사용
- 키보드 조작으로 점프하여 장애물을 넘는 게임

<조작키> : CTRL + SPACE





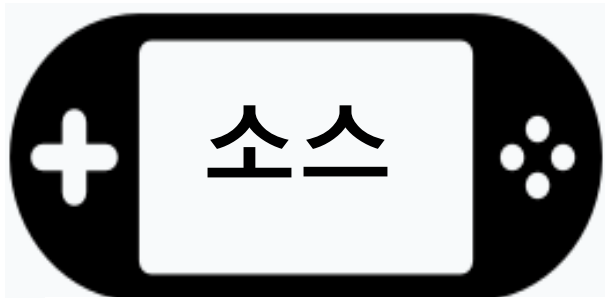
AIMHERO(송재완)

- 시각화 : CANVAS 태그 사용
- 목표물을 마우스 클릭하여 지우는 게임

<조작키>

:마우스, 클릭





TETRIS(송재완)

```
25     <div>
26         <canvas id="gameRoom" width="400" height="800"></canvas>
27     </div>
28 </body>
29
30 </html>
31 <script>
32     const gameRoom = document.getElementById('gameRoom');
33     var tool = gameRoom.getContext('2d');
```

```
54 // 사각형 한조각 클래스
55 class Piece {
56     x = 0;
57     y = 0;
58     len = 40;
59     pFlag = false; // 다시 그릴지 말지 결정할 플래그
60     nowColor = 'tomato'
61     drawPiece() {
62         if (!this.pFlag) {
63             tool.beginPath();
64             tool.strokeStyle = 'black';
65             tool.fillStyle = this.nowColor;
66             tool.fillRect(this.x, this.y, this.len, this.len);
67             tool.strokeRect(this.x, this.y, this.len, this.len);
68             tool.stroke();
69             tool.fill();
70         }
71     }
72 }
```

```
83 // 모양잡기전 공통메서드 묶어놓은 클래스
84 class MyType {
85     p1 = new Piece();
86     p2 = new Piece();
87     p3 = new Piece();
88     p4 = new Piece();
89
90     x = 200; // 가운데에서 출발
91     y = 0;
92
93     setColor(color) {
94         this.p1.nowColor = color;
95         this.p2.nowColor = color;
96         this.p3.nowColor = color;
97         this.p4.nowColor = color;
98     }
99
100
101     drawI() {
102         this.p1.drawPiece();
103         this.p2.drawPiece();
104         this.p3.drawPiece();
105         this.p4.drawPiece();
106     }
}
```

```

108 rotateChk(){ // 회전시 다른 객체의 범위와 겹치는지 체크
109     let pArr = [this.p2, this.p3, this.p4];
110     let newX = 0;
111     let newY = 0;
112     let m = new MyType();
113     let tempArr = [m.p2,m.p3,m.p4];
114     m.p1.x = this.x;
115     m.p2.y = this.y;
116     for (let j = 0; j < pArr.length; j++) {
117         for (let i = -40; i <= 40; i += 40) {
118             if ((this.p1.x + i) == (pArr[j].x)) {
119                 newY = this.p1.y + i;
120                 if ((this.p1.y) == (pArr[j].y)) {
121                     newX = this.p1.x;
122                 } else if ((this.p1.y - 40) == (pArr[j].y)) {
123                     newX = this.p1.x + 40;
124                 } else if ((this.p1.y + 40) == (pArr[j].y)) {
125                     newX = this.p1.x - 40;
126                 }
127             }
128         }
129         tempArr[j].y = newY;
130         tempArr[j].x = newX;
131     }
132     let t = new tempType(m);
133     if(crashBound(t, fullArr, pieceBound)){
134         return true;
135     }
136     return false;
137
138
139 }

```

```

142 rotate(p1, p2, p3, p4) {
143     let pArr = [p2, p3, p4];
144     let newX = 0;
145     let newY = 0;
146     // p1기준으로 40만큼의 반경에 다른 조각들 감지확인후 시계방향 회전
147     // console.log(p1.x + "/" + p1.y);
148     for (let j = 0; j < pArr.length; j++) {
149         for (let i = -40; i <= 40; i += 40) {
150             if ((p1.x + i) == (pArr[j].x)) {
151                 newY = p1.y + i;
152                 if ((p1.y) == (pArr[j].y)) {
153                     newX = p1.x;
154                 } else if ((p1.y - 40) == (pArr[j].y)) {
155                     newX = p1.x + 40;
156                 } else if ((p1.y + 40) == (pArr[j].y)) {
157                     newX = p1.x - 40;
158                 }
159             }
160         }
161         // for문 끝날때 값 바꿈
162         pArr[j].y = newY;
163         pArr[j].x = newX;
164     }
165
166 }

```

```

168 nowRetate(rCnt) { // 회전수 적용할 함수
169     let num = rCnt % 4;
170     switch (num) {
171         case 1:
172             this.rotate(this.p1, this.p2, this.p3, this.p4);
173             break;
174         case 2:
175             this.rotate(this.p1, this.p2, this.p3, this.p4);
176             this.rotate(this.p1, this.p2, this.p3, this.p4);
177             break;
178         case 3:
179             this.rotate(this.p1, this.p2, this.p3, this.p4);
180             this.rotate(this.p1, this.p2, this.p3, this.p4);
181             this.rotate(this.p1, this.p2, this.p3, this.p4);
182             break;
183         default:
184             break;
185     }
186 }

```

```

199 // I모양
200 class TypeI {
201
202     m = new MyType(); // 공통항목 객체 생성
203
204     tColor = 'lightblue'; // 색지정
205
206     setWidth() { // 가로 그리기
207         this.m.p1.x = this.m.x;
208         this.m.p2.x = this.m.p1.x - 40;
209         this.m.p3.x = this.m.p1.x + 40;
210         this.m.p4.x = this.m.p3.x + 40;
211
212         this.m.p1.y = this.m.y;
213         this.m.p2.y = this.m.p1.y;
214         this.m.p3.y = this.m.p1.y;
215         this.m.p4.y = this.m.p1.y;
216
217         this.m.setColor(this.tColor);
218         this.m.drawI();
219     }
220     setHeight() { // 세로 그리기
221         this.m.p1.x = this.m.x;
222         this.m.p2.x = this.m.p1.x;
223         this.m.p3.x = this.m.p1.x;
224         this.m.p4.x = this.m.p1.x;
225
226         this.m.p1.y = this.m.y;
227         this.m.p2.y = this.m.p1.y - 40;
228         this.m.p3.y = this.m.p1.y + 40;
229         this.m.p4.y = this.m.p3.y + 40;
230         this.m.setColor(this.tColor);
231         this.m.drawI();
232     }
233
234 }

```

```

// 1모양
class TypeL {
    m = new MyType(); // 공통항목 객체 생성

    tColor = 'orange'; // 색지정

    standard() { // 가운데 중심 기준으로 그리기
        this.m.p1.x = this.m.x;
        this.m.p2.x = this.m.p1.x - 40;
        this.m.p3.x = this.m.p1.x - 40;
        this.m.p4.x = this.m.p1.x + 40;

        this.m.p1.y = this.m.y;
        this.m.p2.y = this.m.p1.y;
        this.m.p3.y = this.m.p1.y + 40;
        this.m.p4.y = this.m.p1.y;

        this.m.nowRetate(rotateCnt);
    }
}

```

```

278
279 // 0모양 : 예외적으로 메서드 다름..
280 class Type0 {
281     m = new MyType(); // 공통항목 객체 생성
282
283     tColor = 'yellow'; // 색지정
284
285     setPoint() { // 그리기
286         this.m.p1.x = this.m.x;
287         this.m.p2.x = this.m.p1.x + 40;
288         this.m.p3.x = this.m.p2.x;
289         this.m.p4.x = this.m.p3.x - 40;
290
291         this.m.p1.y = this.m.y;
292         this.m.p2.y = this.m.p1.y;
293         this.m.p3.y = this.m.p2.y + 40;
294         this.m.p4.y = this.m.p3.y;
295
296         this.m.setColor(this.tColor);
297         this.m.drawI();
298     }
299 }
300

```



```

384 function pieceBound(p1, p2) {
385     if (!p1.pFlag) { // 그리지 않는 객체는 범위에서 제외
386         if (!p2.pFlag) {
387             if ((p1.x < p2.x + p2.len / 2) &&
388                 (p1.x + p1.len / 2 > p2.x - p2.len / 2) &&
389                 (p1.y <= p2.y + p2.len / 2) &&
390                 (p1.y + p1.len / 2 >= p2.y - p2.len / 2)) {
391                 return false;
392             }
393         }
394     }
395     return true;
396 }

```

```

416 // 배열내 객체와 새로 뽑은 객체의 충돌범위 설정
417 function crashBound(t, arr, boundFunction) { // 새로 뽑은 객체와 기존의 배열을 인자값으로 전달
418     let tempCnt = 0;
419     for (v of arr) {
420         if ((boundFunction(t.m.p1, v.m.p1)
421             && boundFunction(t.m.p2, v.m.p1)
422             && boundFunction(t.m.p2, v.m.p1)
423             && boundFunction(t.m.p1, v.m.p1))
424             && (boundFunction(t.m.p1, v.m.p2)
425                 && boundFunction(t.m.p2, v.m.p2)
426                 && boundFunction(t.m.p3, v.m.p2)
427                 && boundFunction(t.m.p4, v.m.p2))
428             && (boundFunction(t.m.p1, v.m.p3)
429                 && boundFunction(t.m.p2, v.m.p3)
430                 && boundFunction(t.m.p3, v.m.p3)
431                 && boundFunction(t.m.p4, v.m.p3))
432             && (boundFunction(t.m.p1, v.m.p4)
433                 && boundFunction(t.m.p2, v.m.p4)
434                 && boundFunction(t.m.p3, v.m.p4)
435                 && boundFunction(t.m.p4, v.m.p4))) { // 충돌이 없을 때 true 충돌이 있다면 false
436             tempCnt++;
437             // console.log(tempCnt+"/"+arr.length);
438         }
439     }
440     if (tempCnt == arr.length) { // 충돌이 없는경우
441         // console.log("aaa");
442         return true;
443     }
444     return false; // 충돌이 있는경우
445 }

```

```

    console.log("game over");
  } else {
    if ((i.m.p1.x == 200) && (i.m.p1.y == 0)) { // 도형이 나오자마자 멈춘 경우
      console.log("game over");
      b.gameoverDraw();
      cancelAnimationFrame(now);
    }
  }
}

```

```

475 function fall() {
476   let now = requestAnimationFrame(fall)
477   // ...

```

```

// 한줄 완성 감지...
// 한줄완성이란 y좌표가 같은 조각객체가 10개가 되는것.
// 새로운 객체의 y좌표가 멈추는 순간 10번째 y좌표가 되는순간을 감지..
function lineClear() { // 삭제가 일어난다면 다른 리턴값을 주고 삭제가 된 줄을 다시 채워야함..
  for (p = 800; p > 0; p -= 40) {
    let lineCnt = 0;
    let lineArr = [];
    for (v = 0; v < fullArr.length; v++) { // call by reference를 위해 for of가 아닌 일반 for문 사용
      if (p == fullArr[v].m.p1.y) {
        lineCnt++;
        lineArr.push(fullArr[v].m.p1)
      }
      if (p == fullArr[v].m.p2.y) {
        lineCnt++;
        lineArr.push(fullArr[v].m.p2)
      }
      if (p == fullArr[v].m.p3.y) {
        lineCnt++;
        lineArr.push(fullArr[v].m.p3)
      }
      if (p == fullArr[v].m.p4.y) {
        lineCnt++;
        lineArr.push(fullArr[v].m.p4)
      }
    }
    if ((lineCnt % 10 == 0) && (lineCnt != 0)) { // 10개 채운 경우 peice 객체 삭제해야한다..
      for (v = 0; v < lineArr.length; v++) { // call by reference를 위해 for of가 아닌 일반 for문 사용
        lineArr[v].pFlag = true; // 다시 못그리게 플래그 바꾸기
      }
      replacePeice(p); // 빈칸 채우기
      b.nowScore = parseInt(b.nowScore) + 1000; // 점수증가
    }
  }
}

```

```

543 // 줄 지워지면 해당줄 매꾸기
544 // y좌표가 해당줄보다 작다면 +40씩 준다.
545 function replacePeice(yPoint) { // y좌표 인자값으로 받음
546     for (let i = 0; i < fullArr.length; i++) {
547         (yPoint > fullArr[i].m.p1.y ? (fullArr[i].m.p1.y += 40) : (null));
548         (yPoint > fullArr[i].m.p2.y ? (fullArr[i].m.p2.y += 40) : (null));
549         (yPoint > fullArr[i].m.p3.y ? (fullArr[i].m.p3.y += 40) : (null));
550         (yPoint > fullArr[i].m.p4.y ? (fullArr[i].m.p4.y += 40) : (null));
551     }
552 }
553

```

```

632 // +40만큼 미리 값을 줬을경우 충돌한다면 false : right
633 function preRightPieceBound(p1, p2) {
634     if (!p1.pFlag) { // 그리지 않는 객체는 범위에서 제외
635         if (!p2.pFlag) {
636             if ((p1.x + p1.len / 2 < p2.x + p2.len / 2) &&
637                 (p1.x + p1.len > p2.x - p2.len / 2) &&
638                 (p1.y <= p2.y + p2.len / 2) &&
639                 (p1.y + p1.len / 2 >= p2.y - p2.len / 2)) {
640                 return false;
641             }
642         }
643     }
644     return true;
645 }

```

```

617 // -40만큼 미리 값을 줬을경우 충돌한다면 false : left
618 function preLeftPieceBound(p1, p2) {
619     if (!p1.pFlag) { // 그리지 않는 객체는 범위에서 제외
620         if (!p2.pFlag) {
621             if ((p1.x - p1.len / 2 > p2.x - p2.len / 2) &&
622                 (p1.x - p1.len < p2.x + p2.len / 2) &&
623                 (p1.y <= p2.y + p2.len / 2) &&
624                 (p1.y + p1.len / 2 >= p2.y - p2.len / 2)) {
625                 return false;
626             }
627         }
628     }
629     return true;
630 }

```

```

647 // +40만큼 미리 값을 줬을경우 충돌한다면 false : down
648 function preDownPieceBound(p1, p2) {
649     if (!p1.pFlag) { // 그리지 않는 객체는 범위에서 제외
650         if (!p2.pFlag) {
651             if ((p1.x < p2.x + p2.len / 2) &&
652                 (p1.x + p1.len / 2 > p2.x - p2.len / 2) &&
653                 (p1.y + p1.len < p2.y + p2.len) &&
654                 (p1.y + p1.len * 2 > p2.y - p2.len)) {
655                 return false;
656             }
657         }
658     }
659     return true;
660 }

```

```

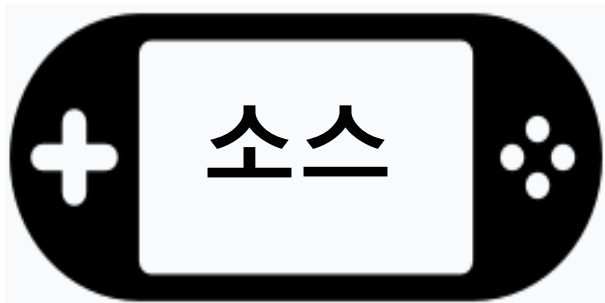
685 } else if (e.keyCode == 37) { // left
686     if (crashBound(i, fullArr, preLeftPieceBound)) {
687         chkLeftBound(i);
688     }
689 } else if (e.keyCode == 40) { // down
690     if (crashBound(i, fullArr, preDownPieceBound)) {
691         if (i.m.y < 640) {
692             i.m.y += 40;
693         }
694     }
695 } else if (e.keyCode == 39) { // right
696     if (crashBound(i, fullArr, preRightPieceBound)) {
697         chkRightBound(i);
698     }
699 }

```

```

664 window.addEventListener('keydown', e => {
665     // console.log(e.keyCode);
666     if (e.keyCode == 38) { // up (spin)
667         // 회전시에 범위를 확인하고 보정작업 필요.
668         if (idx == 1) {
669             if(i.m.p1.x == 0){
670                 i.m.x +=40;
671             }else if(i.m.p1.x == 360){
672                 i.m.x -=80;
673             }
674             f = (!f);
675         } else if (idx != 0) {
676             if(i.m.p1.x == 0){
677                 i.m.x +=40;
678             }else if(i.m.p1.x == 360){
679                 i.m.x -=40;
680             }
681             if(i.m.rotateChk()){
682                 rotateCnt++;
683             }
684         }

```



BREAK-OUT(김지향)

```
<body>
  <div>
    <canvas id="gameRoom" width="500" height="400">
    </canvas>
  </div>
</body>
<footer>
  <div> point : <span id="point"> </span></div>
  <div> run time : <span id="time"></span></div>
</footer>
```

```
//벽돌 2차원배열 생성
for (var c = 0; c < brickColumnCount; c++) {
  bricks[c] = [];
  for (var r = 0; r < brickRowCount; r++) {
    bricks[c][r] = { x: 0, y: 0, status: 1 };
  }
}

//벽돌 그리기
function drawBrick() {
  for (var c = 0; c < brickColumnCount; c++) {
    for (var r = 0; r < brickRowCount; r++) {
      if (bricks[c][r].status == 1) {
        var brickX = (c * (brickWidth + brickPadding)) + brickoffsetLeft;
        var brickY = (r * (brickHeight + brickPadding)) + brickoffsetTop;
        bricks[c][r].x = brickX;
        bricks[c][r].y = brickY;
        ctx.beginPath();
        ctx.rect(brickX, brickY, brickWidth, brickHeight);
        ctx.fillStyle = "rgb(32, 217, 250)";
        ctx.fill();
        ctx.closePath();
      }
    }
  }
}
```

```

//공 그리기
function drawBall() {
    ctx.beginPath();
    ctx.arc(x, y, r, 0, Math.PI * 2);
    ctx.fillStyle = "rgb(246, 32, 250)";
    ctx.fill();
    ctx.closePath();
}

// paddle 그리기
function drawPaddle() {
    ctx.beginPath();
    ctx.rect(paddleX, paddleY, paddleWidth, paddleHeight);
    ctx.fillStyle = "rgb(245, 255, 107)";
    ctx.fill();
    ctx.closePath();
}

```

```

// setInterval함수를 이용하여 계속 갱신할 함수
function draw() {
    ctx.clearRect(0, 0, mycanvas.width, mycanvas.height);
    drawBrick();
    drawBall();
    drawPaddle();
    ballmove();
    collisionDetection();
}

var interval = setInterval(draw, 10);

```

```

//공 벽 충돌 , 패들 위치 변경 함수
function ballmove() {
    if (x + dx > mycanvas.width - r || x + dx < r) { //좌우 벽에 닿으면 방향전환
        dx = -dx;
    }
    if (y + dy < r) { //상하 벽에 닿으면 방향전환
        dy = -dy;
    }
    if (y + dy > mycanvas.height - r) {
        if (paddleX < x + dx && x + dx < paddleX + paddleWidth) {
            dy = -dy;
        } else {
            alert("Game Over" + "\n" + "point:" + point + "\n" + "게임시간:" + time);
            document.location.reload();
            clearInterval(interval);
            clearInterval(timeInterval);
        }
    }
    if (rightPressed && paddleX < mycanvas.width - paddleWidth) {
        paddleX += 7;
    }
    else if (leftPressed && paddleX > 0) {
        paddleX -= 7;
    }
    if (time >= 5 && time % 5 == 0) {
        x += 0.3;
        y += 0.3;
    }
    x += dx;
    y += dy;
}

```

//벽돌 충돌감지

```
function collisionDetection() {
    for (var c = 0; c < brickColumnCount; c++) {
        for (var r = 0; r < brickRowCount; r++) {
            var b = bricks[c][r];
            if (b.status == 1) {
                if (x > b.x && x < b.x + brickWidth && y > b.y && y < b.y + brickHeight) {
                    dy = -dy;
                    b.status = 0;
                    cnt -= 1;
                    point += 100;
                    document.getElementById("point").innerHTML = point;
                    if (cnt == 0) {
                        time = 0;
                        point = 0;
                        alert("참 잘했어요. 다음 stage에서 만나요.");
                        clearInterval(interval);
                        clearInterval(timeInterval);
                    }
                }
            }
        }
    }
}
```

//방향키 눌렀을때 리스너 작동설정

```
document.addEventListener("keydown", keyDownHandler, false);
document.addEventListener("keyup", keyUpHandler, false);
```

// 키눌러서 방향 전환

```
function keyDownHandler(e) {
    if (e.key == 37 || e.key == "ArrowLeft") {
        leftPressed = true;
    }
    else if (e.key == 39 || e.key == "ArrowRight") {
        rightPressed = true;
    }
}
```

//키 안눌렀을때 , 해당변수들 초기화

```
function keyUpHandler(e) {
    if (e.key == 37 || e.key == "ArrowLeft") {
        leftPressed = false;
    }
    else if (e.key == 39 || e.key == "ArrowRight") {
        rightPressed = false;
    }
}
```

//런타임 표시

```
const timeInterval = setInterval(() => {
    time += 1;
    document.getElementById("time").innerHTML = time;
}, 1000);
```



공통 : NODE.js를 사용하여
서버 구축을 통한 멀티 플레이 기능 추가



개인별

송재완 - 철권 게임 개발

김지향 - 가상캐릭터 채팅게임 개발

이예진 - 메이플스토리 같은 스토리게임 개발