

ksgame project#2

2018870559 정성재

multipurpose software for the family
center for giving them you're looking
your face and
I'm sure you'll find the best of
the world in the world
don't worry though, the best of the world is right
in front of you

0425935 5686

4000 000000 0000 00
0000 0000 0000 00
0000 0000 0000 00

the best of the world is right
in front of you

the best of the world is right
in front of you

GSAT

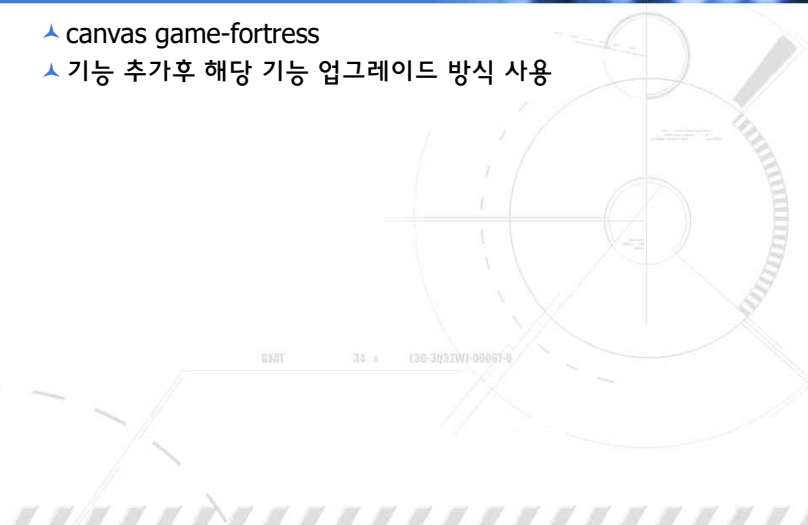
34

L36-3Q32WI-00067-5

98442123-435 3

선택한 source code와 코드 수정 방법

- ▶ canvas game-fortress
- ▶ 기능 추가후 해당 기능 업그레이드 방식 사용



초기 code

```

DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta http-equiv="X-UA-Compatible" content="ie=edge" />
  <title>fortress</title>
  <style>
    body {
      margin: 0;
      height: 100vh;
      display: flex;
      justify-content: center;
      align-items: center;
    }
    #fortress {
      border: 1px solid black;
    }
  </style>
</head>
<body>
  <canvas id="fortress" width="1000px" height="700px"></canvas>
  <script>
    const canvas = document.getElementById("fortress");
    const ctx = canvas.getContext("2d");
    const width = canvas.width;
    const height = canvas.height;
    const tankWidth = 50;
    const tankHeight = 50;
    let tankX = 0;
    const tankDx = 3;
    let tankLeftPressed = false;
    let tankRightPressed = false;
    let tankCenterX;
    let tankCenterY;
    let cannonAngle = Math.PI / 4;
    const cannonAngleIDIF = Math.PI / 60;
    const cannonLength = tankWidth * Math.sqrt(2);
    const targetWidth = Math.floor(Math.random() * 100 + 30);
    const targetHeight = Math.floor(Math.random() * 100 + 10);
    const targetX = Math.floor(Math.random() * (500 - targetWidth) + 500);
    const targetY = height - targetHeight;

    const draw = () => {
      ctx.clearRect(0, 0, width, height);
      tankCenterX = tankX + 0.5 * tankWidth;
      tankCenterY = height - 0.5 * tankHeight;
      if (tankLeftPressed && tankX > 0) {
        tankX -= tankDx;
      }
      if (tankRightPressed && tankX + tankWidth < width) {
        tankX += tankDx;
      }
      drawTank();
      drawTarget();
      drawMissile();
    };

    const drawTank = () => {
      ctx.lineWidth = 5;
      ctx.lineCap = "round";
      ctx.beginPath();
      ctx.moveTo(tankX, height - tankHeight);
      ctx.lineTo(tankX + tankWidth, height - tankHeight);
      ctx.lineTo(tankX + tankWidth, height);
      ctx.lineTo(tankX, height);
      ctx.lineTo(tankX, height - tankHeight);
      ctx.moveTo(tankCenterX, tankCenterY);
      ctx.lineTo(
        tankCenterX + cannonLength * Math.cos(cannonAngle),
        tankCenterY - cannonLength * Math.sin(cannonAngle)
      );
      ctx.stroke();
      ctx.closePath();
    };

    const drawTarget = () => {
      ctx.fillRect(targetX, targetY, targetWidth, targetHeight);
      ctx.fillStyle = "red";
    };

    const drawMissile = () => {
      draw();
    };

    const keydownHandler = event => {
      if (event.keyCode === 37) {
        tankLeftPressed = true;
      } else if (event.keyCode === 39) {
        tankRightPressed = true;
      } else if (event.keyCode === 38 && cannonAngle <= Math.PI) {
        cannonAngle += cannonAngleIDIF;
      } else if (event.keyCode === 40 && cannonAngle >= 0) {
        cannonAngle -= cannonAngleIDIF;
      }
    };

    const keyupHandler = event => {
      if (event.keyCode === 37) {
        tankLeftPressed = false;
      } else if (event.keyCode === 39) {
        tankRightPressed = false;
      }
    };

    const start = setInterval(draw, 10);
    document.addEventListener("keydown", keydownHandler, false);
    document.addEventListener("keyup", keyupHandler, false);
  </script>
</body>
</html>

```

기능(0)

```
script;
const canvas = document.getElementById("fortress");
const ctx = canvas.getContext("2d");
const width = canvas.width;
const height = canvas.height;
const tankWidth = 50;
const tankHeight = 50;
let tankX = 0;
const tankDx = 3;
let tankLeftPressed = false;
let tankRightPressed = false;
let tankCenterX;
let tankCenterY;
let cannonAngle = Math.PI / 4;
const cannonAngleDIF = Math.PI / 60;
const cannonLength = tankWidth * Math.sqrt(2);
const targetWidth = Math.floor(Math.random() * 100 + 30);
const targetHeight = Math.floor(Math.random() * 100 + 10);
const targetX = Math.floor(Math.random() * (500 - targetWidth) + 500);
const targetY = height - targetHeight;
```

```
const draw = () => {
  ctx.clearRect(0, 0, width, height);
  tankCenterX = tankX + 0.5 * tankWidth;
  tankCenterY = height - 0.5 * tankHeight;
  if (tankLeftPressed && tankX > 0) {
    tankX -= tankDx;
  }
  if (tankRightPressed && tankX + tankWidth < width) {
    tankX += tankDx;
  }
  drawTank();
  drawTarget();
  drawMissile();
};
```

```
let missileRadius = 5;
let missileX;
let missileY;
let isCharging = false;
let isFired = false;
let isHit = false;
let gauge = Math.PI;
const gaugeDIF = Math.PI / 60;
const gaugeBarRadius = 30;
let missilePower;
let missileDx;
let missileDy;
const GRAVITY_ACCELERATION = 0.098;
```

기능 추가를 위해 필요한 변수를 선언

기능(1-1) 파워 게이지

```
const keydownHandler = event => {  
  if (event.keyCode === 37) {  
    tankLeftPressed = true;  
  } else if (event.keyCode === 39) {  
    tankRightPressed = true;  
  } else if (event.keyCode === 38 && cannonAngle <= Math.PI / 2) {  
    cannonAngle += cannonAngleDIF;  
  } else if (event.keyCode === 40 && cannonAngle >= 0) {  
    cannonAngle -= cannonAngleDIF;  
  } else if (event.keyCode === 32 && !isFired) {  
    isCharging = true;  
  }  
};  
const keyupHandler = event => {  
  if (event.keyCode === 37) {  
    tankLeftPressed = false;  
  } else if (event.keyCode === 39) {  
    tankRightPressed = false;  
  } else if (event.keyCode === 32 && !isFired) {  
    isCharging = false;  
    isFired = true;  
  }  
};
```

- ▶ keydownHandler와 keyupHandler 아래쪽에 추가
- ▶ 스페이스바(keyCode로 32)가 키다운상태이면 게이지 충전
을 시작하고(isCharging = true) 상태가 해제되면 게이지
충전이 끝나면서(isCharging = false) 미사일이 발사
(isFired = true)

L3G-3Q3ZW1-00067-9

기능(1-2)-파워게이지

```
const draw = () => {  
  ctx.clearRect(0, 0, width, height);  
  tankCenterX = tankX + 0.5 * tankWidth;  
  tankCenterY = height - 0.5 * tankHeight;  
  if (tankLeftPressed && tankX > 0) {  
    tankX -= tankDx;  
  }  
  if (tankRightPressed && tankX + tankWidth < width) {  
    tankX += tankDx;  
  }  
  if (isCharging && !isFired) {  
    if (gauge < Math.PI * 2) {  
      gauge += gaugeDIF;  
    }  
  }  
  drawTank();  
  drawTarget();  
  drawMissile();  
};
```

- ▶ draw() 함수 내부에서 isCharging에 따라서 게이지가 충전
- ▶ isCharing이 true이고 아직 발사되지 않았을 때 gauge는 충전
- ▶ gauge는 초기값인 Math.PI 에서 시작해서 Math.PI * 2 까지 충전

기능(1-3) 파워게이지

```
const draw = () => {  
  const draw = () => {  
    ctx.clearRect(0, 0, width, height);  
    tankCenterX = tankX + 0.5 * tankkidth;  
    tankCenterY = height - 0.5 * tankHeight;  
    if (tankLeftPressed && tankX > 0) {  
      tankX -= tankDx;  
    }  
    if (tankRightPressed && tankX + tankWidth < width) {  
      tankX += tankDx;  
    }  
    if (isChanging && !isFired) {  
      if (gauge < Math.PI * 2) {  
        gauge += gaugeDIF;  
      }  
      drawGauging();  
    }  
    drawTank();  
    drawTarget();  
    drawMissile();  
  };  
  const drawGauging = () => {  
    ctx.beginPath();  
    ctx.arc(  
      tankCenterX,  
      tankCenterY - cannonLength,  
      gaugeBarRadius,  
      Math.PI,  
      gauge,  
      false  
    );  
    ctx.stroke();  
  };  
};
```

- ▶ 충전된 게이지를 그리는 drawGauge() 함수를 작성하고 충전될 때마다 <canvas>에 그려지도록 조건문 안에서 호출

기능(1-4) 파워 게이지

```
const keyupHandler = event => {  
  if (event.keyCode === 37) {  
    tankLeftPressed = false;  
  } else if (event.keyCode === 39) {  
    tankRightPressed = false;  
  } else if (event.keyCode === 32 && !isFired) {  
    isCharging = false;  
    isFired = true;  
    gauge = Math.PI;  
  }  
};
```

▲ 키다운 해제시 게이지 초기화를 위해 $\text{gauge} = \text{Math.PI}$ 문장 삽입

기능(2-1) 미사일 발사

```
const draw = () => {
  ctx.clearRect(0, 0, width, height);
  tankCenterX = tankX + 0.5 * tankWidth;
  tankCenterY = height - 0.5 * tankHeight;
  if (tankLeftPressed && tankX > 0) {
    tankX -= tankDx;
  }
  if (tankRightPressed && tankX + tankWidth < width) {
    tankX += tankDx;
  }
  if (isCharging && !isFired) {
    if (gauge < Math.PI * 2) {
      gauge += gaugeDIF;
    }
    drawGauging();
  }
  if (!isFired) {
    missileX = tankCenterX + cannonLength * Math.cos(cannonAngle);
    missileY = tankCenterY - cannonLength * Math.sin(cannonAngle);
  }
  drawTank();
  drawTarget();
  drawMissile();
};

const drawMissile = () => {
  ctx.beginPath();
  ctx.arc(missileX, missileY, missileRadius, 0, Math.PI * 2);
  ctx.fillStyle = "blue";
  ctx.fill();
  ctx.closePath();
};
```

- ▶ 캐논의 끝부분에 미사일이 보여지도록 미사일을 그리는 함수 선언
- ▶ if 조건으로 미사일이 발사중이 아닐 때 캐논의 끝부분에 위치

L3G-3Q3ZW1-00067-9

기능(2-2) 미사일 발사

```
if (!isFired) {  
    missileX = tankCenterX + cannonLength * Math.cos(cannonAngle);  
    missileY = tankCenterY - cannonLength * Math.sin(cannonAngle);  
} else {  
    missileDy += GRAVITY_ACCELERATION;  
    missileX = missileX + missileDx;  
    missileY = missileY - missileDy;  
}
```

▶ 발사되었을 때의 움직이는 미사일을 그리기 위해 if 조건문에 아래 코드를 추가

▶ 더 알아보기: 미사일이 날아가는 동안에는 x방향으로는 missileDx라는 동일한 힘을(등속도) 받고, y방향으로는 missileDy(중력가속도)의 영향으로 아래쪽으로 점점 큰 힘(등가속도)을 받도록 설계

기능(2-3) 미사일 발사

```
const keyupHandler = event => {  
  if (event.keyCode === 37) {  
    tankLeftPressed = false;  
  } else if (event.keyCode === 39) {  
    tankRightPressed = false;  
  } else if (event.keyCode === 32 && !isFired) {  
    isCharging = false;  
    isFired = true;  
    missilePower = gauge * 1.6;  
    missileDx = missilePower * Math.cos(cannonAngle);  
    missileDy = missilePower * Math.sin(cannonAngle);  
    gauge = Math.PI;  
  }  
};
```

- ▶ missileDx와 missileDy가 각도와 파워게이지에 따라서 정해지도록 keyupHandler에 아래 코드를 추가
- ▶ 키다운 상태인 스페이스바가 상태 해제 될때 isCharging이 false로 되면서 파워게이지의 충전이 끝나고 isFired가 true로 되면서 미사일의 발사가 감지됩니다. 충전된 gauge에 의해서 미사일의 파워와, 그에 따른 x방향 속도, y방향 속도가 결정됩니다. gauge는 다시 처음값으로 초기화

기능(3-1) 판정

```
const draw = () => {  
    // ... 생략  
    checkMissile();  
    drawTank();  
    drawTarget();  
    drawMissile();  
};  
const checkMissile = () => {};
```

▶ checkMissile() 함수를
작성하고 draw() 함수
내부에서 호출

기능(3-2) 판정-명중하지 않았을때

```
const checkMissile = () => {  
  // canvas 왼쪽, 오른쪽, 아래 밖에 닿으면  
  if (missileX <= 0 || missileX >= width || missileY >= height) {  
    isFired = false;  
  }  
};
```

- ▶ 미사일이 명중되지 않은 경우는 미사일이 왼쪽, 오른쪽, 아래쪽 canvas 테두리에 닿았을 때가 되도록 구현
- ▶ 명중되지 않았을 경우 다시 발사 할수 있도록 초기화 해준다

기능(3-3) 판정-명중했을때

```
const checkMissile = () => {  
  // canvas 왼쪽, 오른쪽, 아래 벽에 닿으면  
  if (missileX <= 0 || missileX >= width || missileY >= height) {  
    isFired = false;  
  }  
  // target 명중  
  if (  
    missileX >= targetX &&  
    missileX <= targetX + targetWidth &&  
    missileY >= targetY  
  ) {  
    isHit = true;  
    clearInterval(start);  
    if (confirm("명중입니다. 다시 하시겠습니까?")) {  
      location.reload();  
    }  
  }  
};
```

- ▶ 미사일이 명중된 경우는 미사일의 x좌표와 y좌표가 목표물의 테두리 안에 닿았을 때가 되도록 구현
- ▶ 미사일이 명중되면 isHit가 true가 되면 clearInterval()로 인해 draw() 함수의 실행이 중지

기능(3-4) 판정

```
const draw = () => {  
  // ... 선택  
  
  checkMissile();  
  drawTank();  
  if (!isHitted) {  
    drawTarget();  
    drawMissile();  
  }  
};
```

GSAT

34

(36-39)

- ▶ draw() 함수 내부에서 isHitted에 따라 미사일과 목표물이 그려지도록 수정
- ▶ 미사일이 목표물에 명중되어 isHitted가 true가 되면 목표물과 미사일이 더이상 보이지 않게된다

```
<docType html>  
<html lang="en">  
  <head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>  
    <meta http-equiv="x-ua-compatible" content="ie=edge"/>  
    <title>Fortress</title>  
    <style>  
      body {  
        margin: 0;  
        height: 100vh;  
        display: flex;  
        justify-content: center;  
        align-items: center;  
      }  
      #fortress {  
        border: 1px solid black;  
      }  
    </style>  
  </head>  
  <body>  
    <canvas id="fortress" width="1000px" height="700px"></canvas>  
    <script>  
      const canvas = document.getElementById("fortress");  
      const ctx = canvas.getContext("2d");  
      const width = canvas.width;  
      const height = canvas.height;  
      const tankWidth = 50;  
      const tankHeight = 50;  
      let tankX = 0;  
      const tankY = 3;  
      let tankLeftPressed = false;  
      let tankRightPressed = false;  
      let tankCenterX;  
      let tankCenterY;
```

```
let cannonAngle = Math.PI / 4;
const cannonVelocDIF = Math.PI / 60;
const cannonLength = tankWidth * Math.sqrt(2);
const targetWidth = Math.floor(Math.random() * 100 + 30);
const targetHeight = Math.floor(Math.random() * 100 + 10);
const targetX = Math.floor(Math.random() * 500 - targetWidth) + 500;
const targetY = height - targetHeight;
let missileRadius = 5;
let missileX;
let missileY;
let isCharging = false;
let isFired = false;
let isLiftd = false;
let isLifted = false;
let gauge = Math.PI;
const gaugeDIF = Math.PI / 60;
const gaugeStarRadius = 30;
let missilePower;
let missileId;
const SHOTVIT_ACCELERATION = 0.099;
const draw = () => {
  ctx.clearRect(0, 0, width, height);
  tankContext = tankX + 0.5 * tankWidth;
  tankContextY = height - 0.5 * tankHeight;
  if (tankLifftressed && tankX > 0) {
    tankX -= tankWdth;
  }
  if (tankHeighttressed && tankX + tankWidth < width) {
    tankX += tankWdth;
  }
  if (isCharging && isFired) {
    if (gauge < Math.PI * 2) {
      gauge += gaugeDIF;
    }
    drawGauge();
  }
}
```

```

}
if (!isFired) {
    missileX = tankCenterX + cannonLength * Math.cos(cannonAngle);
    missileY = tankCenterY - cannonLength * Math.sin(cannonAngle);
} else {
    missileY += GRAVITY_ACCELERATION;
    missileX = missileX + missileDx;
    missileY = missileY - missileDy;
}
checkMissile();
drawLink();
drawLink();
if (!isHitTarget) {
    drawTarget();
    drawMissile();
}
};

current checkMissile = () => {
    // cannon 발사, 로켓통, 미사일 발사
    if (missileX <= 0 || missileX >= width || missileY >= height) {
        isFired = false;
    }
    // target 맞음
    if (
        missileX >= targetX &&
        missileX <= targetX + targetWidth &&
        missileY >= targetY
    ) {
        isHitTarget = true;
        clearInterval(start);
        if (confirm("맞습니다. 다시 하시겠습니까?")) {
            location.reload();
        }
    }
};

current drawMissile = () => {

```

```

ctx.beginPath();
ctx.moveTo(x, y);
ctx.fillStyle = "blue";
ctx.fill();
ctx.closePath();
});

const drawDrawing = () => {
  ctx.beginPath();
  ctx.arc(
    tankCenterX,
    tankCenterY, - cannon.length,
    cannonRadius,
    Math.PI,
    false
  );
  ctx.stroke();
};

const drawTank = () => {
  ctx.lineWidth = 5;
  ctx.strokeStyle = "red";
  ctx.beginPath();
  ctx.moveTo(tankX, height + tankHeight);
  ctx.lineTo(tankX + tankWidth, height + tankHeight);
  ctx.lineTo(tankX + tankWidth, height);
  ctx.lineTo(tankX, height);
  ctx.moveTo(tankX, height + tankHeight);
  ctx.lineTo(
    tankCenterX + cannon.length * Math.cos(cannonAngle),
    tankCenterY - cannon.length * Math.sin(cannonAngle)
  );
  ctx.stroke();
  ctx.closePath();
};

```

```
const drawTarget = {} <= {
    canvas: document.getElementById('target'), targetWidth, targetHeight, 2
};
ctx.fillStyle = "red";
draw();
const keyDownHandler = event => {
    if (event.keyCode === 37) { // ←
        canLeftPressed = true;
    } else if (event.keyCode === 39) { // →
        canRightPressed = true;
    } else if (event.keyCode === 38 || cannonAngle < Math.PI)
        cannonAngle -= cannonAngleDef;
    else if (event.keyCode === 40 || cannonAngle > 0) {
        cannonAngle += cannonAngleDef;
    } else if (event.keyCode === 32 || !isFired) {
        isCharging = true;
    }
};
const keyUpHandler = event => {
    if (event.keyCode === 37) { // ←
        canLeftPressed = false;
    } else if (event.keyCode === 39) { // →
        canRightPressed = false;
    } else if (event.keyCode === 32 || !isFired) {
        isCharging = false;
        isFired = true;
        missilePower = gauge * 2;
        missile = missilePower + Math.cos(cannonAngle);
        missileY = missilePower + Math.sin(cannonAngle);
        gauge = Math.PI;
    }
};
const start = setInterval(draw, 60);
document.addEventListener('keydown', keyDownHandler, false);
document.addEventListener('keyup', keyUpHandler, false);
}()
</script>
```


업그레이드(1)-각도 확장

```
} else if (event.keyCode === 38 && cannonAngle <= Math.PI / 2) {  
  cannonAngle += cannonAngleDIF;  
}
```

```
} else if (event.keyCode === 38 && cannonAngle <= Math.PI ) {  
  cannonAngle += cannonAngleDIF;  
}
```



▲ $\text{Pi}/2$ 를 Pi 로 변경하여 각도 확장

GSAT

34

L3G-3Q32W1-00067-9

업그레이드(2)-키다운 후에 출력

- 키다운 횟수를 지칭하는 변수 선언 후 조건문에 해당 변수가 1 이상 일때 메시지를 출력하도록 구현하고 벽에 맞았을 경우 그 횟수를 초기화
- 이때 키다운 상태에서 변수 값을 올릴 경우 키다운 상태일동안 계속 올라가게 된다. 이 경우 타겟 근접 시 스페이스바를 키다운 중인 상태 일때 메시지가 출력되므로 반드시 키다운을 해제한 상태로 넘어갔을 때 변수 값을 올려주도록 한다

```
let missilePower;  
let missileDx;  
let missileDy;  
const GRAVITY_ACCELERATION = 0.098;  
let presscount = 0;  
const draw = () => {
```

```
if (missileX <= 0 || missileX >= width || missileY >= height) {  
  isFired = false;  
  presscount = 0;  
}
```

```
const keyupHandler = event => {  
  if (event.keyCode === 37) {  
    tankLeftPressed = false;  
  } else if (event.keyCode === 39) {  
    tankRightPressed = false;  
  } else if (event.keyCode === 32 && !isFired) {  
    isCharging = false;  
    isFired = true;  
    presscount++;  
    missilePower = gauge * 1.6;  
    missileDx = missilePower * Math.cos(cannonAngle);  
    missileDy = missilePower * Math.sin(cannonAngle);  
    gauge = Math.PI;  
  }  
};
```

업그레이드 최종 코드

```

1 class Hero {
2     constructor(
3         name,
4         health=100,
5         attack=10,
6         defense=10,
7         speed=10,
8         magic=0,
9         xp=0,
10         level=1,
11         expToNextLevel=100
12     ) {
13         this.name = name;
14         this.health = health;
15         this.attack = attack;
16         this.defense = defense;
17         this.speed = speed;
18         this.magic = magic;
19         this.xp = xp;
20         this.level = level;
21         this.expToNextLevel = expToNextLevel;
22     }
23
24     // Getter
25     get health() {
26         return this._health;
27     }
28
29     // Setter
30     set health(health) {
31         this._health = health;
32     }
33
34     // Getter
35     get attack() {
36         return this._attack;
37     }
38
39     // Setter
40     set attack(attack) {
41         this._attack = attack;
42     }
43
44     // Getter
45     get defense() {
46         return this._defense;
47     }
48
49     // Setter
50     set defense(defense) {
51         this._defense = defense;
52     }
53
54     // Getter
55     get speed() {
56         return this._speed;
57     }
58
59     // Setter
60     set speed(speed) {
61         this._speed = speed;
62     }
63
64     // Getter
65     get magic() {
66         return this._magic;
67     }
68
69     // Setter
70     set magic(magic) {
71         this._magic = magic;
72     }
73
74     // Getter
75     get xp() {
76         return this._xp;
77     }
78
79     // Setter
80     set xp(xp) {
81         this._xp = xp;
82     }
83
84     // Getter
85     get level() {
86         return this._level;
87     }
88
89     // Setter
90     set level(level) {
91         this._level = level;
92     }
93
94     // Getter
95     get expToNextLevel() {
96         return this._expToNextLevel;
97     }
98
99     // Setter
100     set expToNextLevel(expToNextLevel) {
101         this._expToNextLevel = expToNextLevel;
102     }
103 }
104
105 // Create a hero
106 const hero = new Hero('Thor', 100, 10, 10, 10, 0, 0, 1, 100);
107
108 // Print hero's stats
109 console.log(hero);
110
111 // Update hero's stats
112 hero.health = 80;
113 hero.attack = 15;
114 hero.defense = 15;
115 hero.speed = 15;
116 hero.magic = 5;
117 hero.xp = 50;
118 hero.level = 2;
119 hero.expToNextLevel = 200;
120
121 // Print hero's stats
122 console.log(hero);
123
124 // Create a hero
125 const hero2 = new Hero('Iron Man', 100, 10, 10, 10, 0, 0, 1, 100);
126
127 // Print hero's stats
128 console.log(hero2);
129
130 // Update hero's stats
131 hero2.health = 80;
132 hero2.attack = 15;
133 hero2.defense = 15;
134 hero2.speed = 15;
135 hero2.magic = 5;
136 hero2.xp = 50;
137 hero2.level = 2;
138 hero2.expToNextLevel = 200;
139
140 // Print hero's stats
141 console.log(hero2);
142
143 // Create a hero
144 const hero3 = new Hero('Spider-Man', 100, 10, 10, 10, 0, 0, 1, 100);
145
146 // Print hero's stats
147 console.log(hero3);
148
149 // Update hero's stats
150 hero3.health = 80;
151 hero3.attack = 15;
152 hero3.defense = 15;
153 hero3.speed = 15;
154 hero3.magic = 5;
155 hero3.xp = 50;
156 hero3.level = 2;
157 hero3.expToNextLevel = 200;
158
159 // Print hero's stats
160 console.log(hero3);
161
162 // Create a hero
163 const hero4 = new Hero('Captain America', 100, 10, 10, 10, 0, 0, 1, 100);
164
165 // Print hero's stats
166 console.log(hero4);
167
168 // Update hero's stats
169 hero4.health = 80;
170 hero4.attack = 15;
171 hero4.defense = 15;
172 hero4.speed = 15;
173 hero4.magic = 5;
174 hero4.xp = 50;
175 hero4.level = 2;
176 hero4.expToNextLevel = 200;
177
178 // Print hero's stats
179 console.log(hero4);
180
181 // Create a hero
182 const hero5 = new Hero('Hulk', 100, 10, 10, 10, 0, 0, 1, 100);
183
184 // Print hero's stats
185 console.log(hero5);
186
187 // Update hero's stats
188 hero5.health = 80;
189 hero5.attack = 15;
190 hero5.defense = 15;
191 hero5.speed = 15;
192 hero5.magic = 5;
193 hero5.xp = 50;
194 hero5.level = 2;
195 hero5.expToNextLevel = 200;
196
197 // Print hero's stats
198 console.log(hero5);
199
200 // Create a hero
201 const hero6 = new Hero('Black Widow', 100, 10, 10, 10, 0, 0, 1, 100);
202
203 // Print hero's stats
204 console.log(hero6);
205
206 // Update hero's stats
207 hero6.health = 80;
208 hero6.attack = 15;
209 hero6.defense = 15;
210 hero6.speed = 15;
211 hero6.magic = 5;
212 hero6.xp = 50;
213 hero6.level = 2;
214 hero6.expToNextLevel = 200;
215
216 // Print hero's stats
217 console.log(hero6);
218
219 // Create a hero
220 const hero7 = new Hero('Scarlet Witch', 100, 10, 10, 10, 0, 0, 1, 100);
221
222 // Print hero's stats
223 console.log(hero7);
224
225 // Update hero's stats
226 hero7.health = 80;
227 hero7.attack = 15;
228 hero7.defense = 15;
229 hero7.speed = 15;
230 hero7.magic = 5;
231 hero7.xp = 50;
232 hero7.level = 2;
233 hero7.expToNextLevel = 200;
234
235 // Print hero's stats
236 console.log(hero7);
237
238 // Create a hero
239 const hero8 = new Hero('Vision', 100, 10, 10, 10, 0, 0, 1, 100);
240
241 // Print hero's stats
242 console.log(hero8);
243
244 // Update hero's stats
245 hero8.health = 80;
246 hero8.attack = 15;
247 hero8.defense = 15;
248 hero8.speed = 15;
249 hero8.magic = 5;
250 hero8.xp = 50;
251 hero8.level = 2;
252 hero8.expToNextLevel = 200;
253
254 // Print hero's stats
255 console.log(hero8);
256
257 // Create a hero
258 const hero9 = new Hero('Wanda Maximoff', 100, 10, 10, 10, 0, 0, 1, 100);
259
260 // Print hero's stats
261 console.log(hero9);
262
263 // Update hero's stats
264 hero9.health = 80;
265 hero9.attack = 15;
266 hero9.defense = 15;
267 hero9.speed = 15;
268 hero9.magic = 5;
269 hero9.xp = 50;
270 hero9.level = 2;
271 hero9.expToNextLevel = 200;
272
273 // Print hero's stats
274 console.log(hero9);
275
276 // Create a hero
277 const hero10 = new Hero('Doctor Strange', 100, 10, 10, 10, 0, 0, 1, 100);
278
279 // Print hero's stats
280 console.log(hero10);
281
282 // Update hero's stats
283 hero10.health = 80;
284 hero10.attack = 15;
285 hero10.defense = 15;
286 hero10.speed = 15;
287 hero10.magic = 5;
288 hero10.xp = 50;
289 hero10.level = 2;
290 hero10.expToNextLevel = 200;
291
292 // Print hero's stats
293 console.log(hero10);
294
295 // Create a hero
296 const hero11 = new Hero('Sage', 100, 10, 10, 10, 0, 0, 1, 100);
297
298 // Print hero's stats
299 console.log(hero11);
300
301 // Update hero's stats
302 hero11.health = 80;
303 hero11.attack = 15;
304 hero11.defense = 15;
305 hero11.speed = 15;
306 hero11.magic = 5;
307 hero11.xp = 50;
308 hero11.level = 2;
309 hero11.expToNextLevel = 200;
310
311 // Print hero's stats
312 console.log(hero11);
313
314 // Create a hero
315 const hero12 = new Hero('Scarlet Witch', 100, 10, 10, 10, 0, 0, 1, 100);
316
317 // Print hero's stats
318 console.log(hero12);
319
320 // Update hero's stats
321 hero12.health = 80;
322 hero12.attack = 15;
323 hero12.defense = 15;
324 hero12.speed = 15;
325 hero12.magic = 5;
326 hero12.xp = 50;
327 hero12.level = 2;
328 hero12.expToNextLevel = 200;
329
330 // Print hero's stats
331 console.log(hero12);
332
333 // Create a hero
334 const hero13 = new Hero('Wanda Maximoff', 100, 10, 10, 10, 0, 0, 1, 100);
335
336 // Print hero's stats
337 console.log(hero13);
338
339 // Update hero's stats
340 hero13.health = 80;
341 hero13.attack = 15;
342 hero13.defense = 15;
343 hero13.speed = 15;
344 hero13.magic = 5;
345 hero13.xp = 50;
346 hero13.level = 2;
347 hero13.expToNextLevel = 200;
348
349 // Print hero's stats
350 console.log(hero13);
351
352 // Create a hero
353 const hero14 = new Hero('Doctor Strange', 100, 10, 10, 10, 0, 0, 1, 100);
354
355 // Print hero's stats
356 console.log(hero14);
357
358 // Update hero's stats
359 hero14.health = 80;
360 hero14.attack = 15;
361 hero14.defense = 15;
362 hero14.speed = 15;
363 hero14.magic = 5;
364 hero14.xp = 50;
365 hero14.level = 2;
366 hero14.expToNextLevel = 200;
367
368 // Print hero's stats
369 console.log(hero14);
370
371 // Create a hero
372 const hero15 = new Hero('Sage', 100, 10, 10, 10, 0, 0, 1, 100);
373
374 // Print hero's stats
375 console.log(hero15);
376
377 // Update hero's stats
378 hero15.health = 80;
379 hero15.attack = 15;
380 hero15.defense = 15;
381 hero15.speed = 15;
382 hero15.magic = 5;
383 hero15.xp = 50;
384 hero15.level = 2;
385 hero15.expToNextLevel = 200;
386
387 // Print hero's stats
388 console.log(hero15);
389
390 // Create a hero
391 const hero16 = new Hero('Scarlet Witch', 100, 10, 10, 10, 0, 0, 1, 100);
392
393 // Print hero's stats
394 console.log(hero16);
395
396 // Update hero's stats
397 hero
```

[illegible]

```

    window = new javax.swing.JFrame(
        "window = window, windowTitle = window");
    window.setVisible(true);
}

@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == button) {
        double temp = 0;
        double result = 0;
        for (int i = 0; i < 10; i++) {
            temp = temp + 1;
            result = result + temp;
        }
        double avg = result / 10;
        System.out.println("Average = " + avg);
    }
}
}

```

[illegible]

```

153 tankRightPressed = true;
154 } else if (event.keyCode === 38 && cannonAngle <= Math.PI) {
155   cannonAngle += cannonAngleDIF;
156 } else if (event.keyCode === 40 && cannonAngle >= 0) {
157   cannonAngle -= cannonAngleDIF;
158 } else if (event.keyCode === 32 && !isFired) {
159   isCharging = true;
160 }
161 }
162
163 const keyupHandler = event => {
164   if (event.keyCode === 37) {
165     tankLeftPressed = false;
166   } else if (event.keyCode === 39) {
167     tankRightPressed = false;
168   } else if (event.keyCode === 32 && !isFired) {
169     isCharging = false;
170     isFired = true;
171     pressCount++;
172     missilePower = gauge * 1.6;
173     missileX = missilePower * Math.cos(cannonAngle);
174     missileY = missilePower * Math.sin(cannonAngle);
175     gauge = Math.PI;
176   }
177 }
178
179 const start = setInterval(draw, 10);
180 document.addEventListener("keydown", keydownHandler, false);
181 document.addEventListener("keyup", keyupHandler, false);
182
183 </script>
184
185 </body>
186 </html>

```

이상입니다.

참조 사이트: <https://codingbroker.tistory.com/76>

index: <https://relaxed-rolypoly-95eed8.netlify.app/>

github: <https://github.com/immotals/ksgame/tree/main/canvas>

GSAT

34

L3G-3Q32W1-00067-9