

HTML 웹 애플리케이션 실습

목차

- 변경점
- 수정 실습 1
- 수정 실습 2
- 수정 실습 3
- 최종 코드

변경점

- 아이템 추가

벽돌을 깨면 일정 확률로 아이템이 드랍됨.

아이템 종류 : 공 개수 증가(+2), 공 개수 증가(*2), 패들 길이 증가
패들에 아이템이 접촉하면 효과 발동.

- 레벨, 목숨 개념 삭제 + 벽돌 개수 조정

스테이지가 넘어가면서 무한대로 진행되던 게임에서 클리어가 가능한 게임으로 변경.
기존 벽돌 개수로는 플레이 타임이 너무 짧기에 벽돌 개수를 늘려 플레이 타임 연장.

- 벽돌 배치 변경

기존의 벽돌 배치 : 직사각형 모양

변경된 벽돌 배치 : 현재 강의에 사용되는 챕 GPT의 로고를 본뜬 모양

수정 실습 1

실습 프롬포트 : 벽돌을 깼을 때 20퍼센트 확률로 아이템이 나오게 해줘 아이템 종류는 공 개수 증가(+2), 공 개수 증가(*2), 패들 길이 증가(지속시간 5초) 아이템은 위에서 천천히 내려오다가 패들에 닿으면 얻어지도록 공 개수 증가는 공의 현재 위치 기준에서 공이 증가하게 만들어줘

결과 : 만들어진 html파일이 매우 크기 때문에 확인 질문에 대한 답을 요구함.

공 복제 방식

A. dx/dy 그대로 복제

B. 랜덤 약간 추가

패들 증가

A. 110 → 160px 사이즈 OK?

B. 너가 원하는 특정 크기 있음?

코드 제공 방식

A. 전체 HTML 완성본 다시 만들어줘

B. 필요한 수정 부분만 덧붙여줘

답변 제시 :

1. 공 복제 방식

A : 속도는 그대로 방향은 진행 방향에서 약간 조정해서 복제

2. 패들 증가

A : OK

3. 코드 제공 방식

A : 전체 완성본 다시 만들어줘

수정 실습 2

실습 프롬포트 : 레벨과 목숨 개념은 삭제하고 벽돌을 더 작고 촘촘하게 해서 개수를 늘려줘

결과 : 레벨이 삭제됨. 공이 모두 떨어지면 게임이 끝나는 방식으로 변경됨. 벽돌의 크기가 기존 $64 * 20$ 크기에서 $40 * 16$ 크기로 작아졌고, 벽돌 배치는 $14줄 * 16열$ 개로 변경됐지만 벽돌 개수가 늘어나면서 패들과 벽돌 사이의 공간이 작아지는 문제 발생

추가 요구사항 : 패들이랑 벽돌 사이를 좀더 늘려줘

결과 : 벽돌과 패들 사이의 간격에 대한 답 요구

패들과 벽돌 사이 간격을 120으로 할까?

아니면 150, 200처럼 더 넓게 할까?

원하는 숫자 말해주면 그 값으로 완전체 다시 만들어줄게!

이후 값을 200으로 설정했지만 차이점이 보이지 않았고, 마우스로 패들을 움직이는 기능이 삭제되는 문제가 발생함. 게임 화면의 크기로 인한 문제로 판단되어 게임 화면 크기 조정을 요구함.

추가 요구사항 : 마우스로 움직이는 기능이 사라졌는데? 그리고 벽돌 이랑 패들 사이 거리가 안늘어났어 게임 화면을 세로로 더 길게 만들어줘

결과 : 마우스로 움직이는 기능이 복구되었고, 벽돌과 패들 사이 거리가 원했던 만큼 늘어남.

수정 실습 3

실습 프롬포트 : 벽돌을 상단으로 몰아놓고 크기를 줄인다음 개수를 늘려줘 그리고 벽돌은 gpt로고 모양으로 배치해줘

결과 : 상단으로 깔끔하게 배치되었지만, 원하는 모양이 나오지는 않음.

추가 요구사항 : gpt로고가 이거 아니냐? 이모양으로 만들어줘 필요하면 벽돌 크기랑 개수 조정해도 됨



결과 : 모양은 비슷하게 나왔지만 해상도가 떨어짐.

추가 요구사항 : 벽돌 개수 더 늘려서 로고를 더 뚜렷하게 나타내줘

결과 : png 파일의 경로와 이미지를 분석하여 더 뚜렷하게 나타냄. 주의점도 설명해줌.

좋아, 이번엔 **진짜 GPT 로고 이미지(gpt 꽃 모양)**를 그대로 샘플링해서 벽돌을 훨씬 촘촘하게 찍어내는 방식으로 바꿔봤어.

△ 준비물 딱 하나

이 HTML 파일과 같은 폴더에 gpt_logo.png 이름으로 방금 그 GPT 로고 이미지를 저장해둬.

(파일명만 맞으면 됨)

아래 코드 전체를 gpt-breakout.html 정도로 저장해서 실행하면 돼.

최종 코드

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8"/>
    <title>GPT 로고 벽돌깨기</title>
    <style>
        *margin: 0; padding: 0; box-sizing: border-box; }
        body{
            min-height: 100vh;
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            background: #0b0e1a;
            color: #fff;
            font-family: system-ui, -apple-system, sans-serif;
        }
        h1margin-bottom: 10px; }
        .game-wrapper{
            position: relative;
            overflow: hidden;
            border-radius: 12px;
            box-shadow: 0 0 20px rgba(0,0,0,0.7);
        }
        canvas{
            display: block;
            background: #050818;
        }
        .center-text{
            position: absolute;
            inset: 0;
            display: flex;
            align-items: center;
            justify-content: center;
            pointer-events: none;
            color: white;
        }
    </style>
</head>
<body>
    <h1>GPT 로고 벽돌깨기</h1>
    <div class="game-wrapper">
        <div class="center-text">
            
        </div>
    </div>
</body>
</html>
```

```
        font-size: 18px;
        text-shadow: 0 0 5px black;
        text-align: center;
        padding: 0 10px;
    }

```

```
</style>
</head>
<body>
<h1>GPT 로고 벽돌깨기</h1>
<div class="game-wrapper">
    <canvas id="gameCanvas" width="800" height="900"></canvas>
    <div class="center-text" id="centerText">gpt_logo.png    로고    불러오는
    중...</div>
</div>
<script>
/* =====
캔버스 및 기본 설정
===== */
const canvas=document.getElementById("gameCanvas");
const ctxcanvas=canvas.getContext("2d");
const centerText=document.getElementById("centerText");
let isRunning=false;
let ballStuck=true;
/* =====
패들 설정
===== */
const ORIGINAL_PADDLE_WIDTH110;
const ENLARGED_PADDLE_WIDTH160;
const PADDLE_HEIGHT14;
const PADDLE_MARGIN_BOTTOM40;
let paddleWidth=ORIGINAL_PADDLE_WIDTH;
let paddleX=(canvas.width-paddleWidth)/2;
let rightPressed=false;
let leftPressed=false;
let isPaddleEnlarged=false;
let paddleEnlargeEndTime0;
/* =====
===== */
const BALL_RADIUS8;
```

```

let balls= [];
/* =====
벽돌(로고 챔플링) 설정
===== */
// 그리드 해상도 (값 올리면 벽돌 개수 ↑, 로고 더 부드러워짐)
const GRID_COLS60;
const GRID_ROWS60;
// 벽돌 크기
const BRICK_WIDTH10;
const BRICK_HEIGHT6;
const BRICK_PADDING2;
// 상단 쪽으로 올리기
const BRICK_OFFSET_TOP40;
// 나머지는 로고 로딩 후 결정됨
let BRICK_OFFSET_LEFT0;
let BRICK_COLSGRID_COLS;
let BRICK_ROWSGRID_ROWS;
// 로고에서 추출한 마스크 (true = 벽돌 놓을 자리)
let brickMasknull;
// 실제 벽돌 데이터
let bricks= [];
let bricksRemaining0;
/* =====
아이템 설정
===== */
const ITEM_DROP_PROB0.2;
const ITEM_SIZE18;
const ITEM_FALL_SPEED2;
const ITEM_PLUS2"PLUS2";
const ITEM_MULT2"MULT2";
const ITEM_PADDLE"PADDLE";
let items= [];
/* =====
초기화 함수
===== */
function initPaddle() {
    paddleX= (canvas.width-paddleWidth) / 2;
}
function createBall() {

```

```

const speed4;
const angle= (Math.random() * 0.60.2) * Math.PI;
return{
  x:paddleXpaddleWidth2,
  y:canvas.heightPADDLE_MARGIN_BOTTOMPADDLE_HEIGHTBALL_RADIUS,
  dx:speedMath.cos(angle),
  dy:-Math.abs(speedMath.sin(angle)),
  radius:BALL_RADIUS
};
}
}

function resetBalls() {
  balls= [createBall()];
  ballStucktrue;
}

/* =====
로고 이미지 → 벽돌 마스크 생성
===== */
function createBrickMaskFromImage(img) {
  const offdocument.createElement("canvas");
  off.widthGRID_COLS;
  off.heightGRID_ROWS;
  const octxoff.getContext("2d");
  // 로고를 위쪽에 맞춰서 그리드에 맞게 스케일링
  const scaleMath.min(GRID_COLSmg.width, GRID_ROWSmg.height);
  const drawWimg.widthscale;
  const drawHimg.heightscale;
  const dx= (GRID_COLSdrawW) / 2; // 가운데 정렬
  const dy0; // 위쪽으로 붙이기
  octx.clearRect(0, 0, GRID_COLS, GRID_ROWS);
  octx.drawImage(img, dx, dy, drawW, drawH);
  const dataoctx.getImageData(0, 0, GRID_COLS, GRID_ROWS).data;
  brickMask= [];
  for(let y0; y < GRID_ROWS; y++) {
    brickMask[y] = [];
    for(let x0; x < GRID_COLS; x++) {
      const idx= (yGRID_COLSw) * 4;
      const rdata[idx];
      const gdata[idx1];
      const bdata[idx2];
    }
  }
}

```

```

        const adata[idx3];
        const brightness0.299r0.587g0.114b;
        // 배경(흰색) 제외, 검은 선 부분만 벽돌로
        const isBrick a > 10 && brightness < 230;
        brickMask[y][x] = isBrick;
    }
}

// 가로 위치 가운데 정렬
const totalWidthGRID_COLS* (BRICK_WIDTHBRICK_PADDING);
BRICK_OFFSET_LEFT= (canvas.widthtotalWidth) / 2;
BRICK_COLSGRID_COLS;
BRICK_ROWSGRID_ROWS;
}

/* =====
벽돌 배열 초기화 (마스크 기반)
===== */
function initBricks() {
    bricks= [];
    bricksRemaining0;
    if(!brickMask) return;
    for(let row0; row < BRICK_ROWS; row++) {
        bricks[row] = [];
        for(let col0; col < BRICK_COLS; col++) {
            if(!brickMask[row][col]) {
                bricks[row][col] = null;
                continue;
            }

            const xBRICK_OFFSET_LEFTcol* (BRICK_WIDTHBRICK_PADDING);
            const yBRICK_OFFSET_TOProw* (BRICK_HEIGHTBRICK_PADDING);
            bricks[row][col] = { x, y, status:1};
            bricksRemaining++;
        }
    }
}

function resetGame() {
    paddleWidthORIGINAL_PADDLE_WIDTH;
    isPaddleEnlargedfalse;
    items= [];
    initPaddle();
}

```

```
initBricks();
resetBalls();
isRunningfalse;
ballStucktrue;
centerText.textContent"Space 또는 클릭해서 시작";
centerText.style.opacity"1";
}

/* =====
=====
function drawPaddle() {
    ctx.fillStyle"#39c6ff";
    const ycanvas.heightPADDLE_MARGIN_BOTTOMPADDLE_HEIGHT;
    ctx.fillRect(paddleX, y, paddleWidth, PADDLE_HEIGHT);
}

function drawBalls() {
    for(const b of balls) {
        ctx.beginPath();
        ctx.arc(b.x, b.y, b.radius, 0, Math.PI2);
        ctx.fillStyle"#ff2d8d";
        ctx.fill();
        ctx.closePath();
    }
}
function drawBricks() {
    if(!bricks.length) return;
    for(let r0; r < bricks.length; r++) {
        const rowbricks[r];
        if(!row) continue;
        for(let c0; c < row.length; c++) {
            const brow[c];
            if(!b|| b.status== 0) continue;
            ctx.fillStyle`hsl(${(rc) * 4}, 70%, 50%)`;
            ctx.fillRect(b.x, b.y, BRICK_WIDTH, BRICK_HEIGHT);
        }
    }
}
function drawItems() {
    for(const it of items) {
        if(!it.active) continue;
```

```
let color="#fff", text="";
if(it.type === ITEM_PLUS2) { color="#ffee00"; text"+2"; }
if(it.type === ITEM_MULT2) { color="#ff0059"; text"x2"; }
if(it.type === ITEM_PADDLE){ color="#00aaff"; text"P"; }
ctx.fillStylecolor;
ctx.beginPath();
ctx.arc(it.x, it.y, ITEM_SIZE2, 0, Math.PI2);
ctx.fill();
ctx.closePath();
ctx.fillStyle"#000";
ctx.font"10px system-ui";
ctx.textAlign"center";
ctx.textBaseline"middle";
ctx.fillText(text, it.x, it.y);
}
}
/*
[이]템 처리
=====
function maybeDropItem(x, y) {
if(Math.random() > ITEM_DROP_PROB) return;
const types= [ITEM_PLUS2, ITEM_MULT2, ITEM_PADDLE];
items.push({
x,
y,
type:types[Math.floor(Math.random() * types.length)],
active:true,
vy:ITEM_FALL_SPEED
});
}
function updateItems() {
const paddleTop=canvas.height-PADDLE_MARGIN_BOTTOM-PADDLE_HEIGHT;
for(const it of items) {
if(!it.active) continue;
it.y+= it.vy;
if(
it.yITEM_SIZE2>= paddleTop &&
it.x>= paddleX &&
it.x<= paddleX+paddleWidth
```

```
        applyItemEffect(it.type);
        it.activefalse;
    }
    if(it.y > canvas.heightITEM_SIZE) it.activefalse;
}
items.items.filter(i =>i.active);
}

function applyItemEffect(type) {
    if(balls.length === 0) return;
    const baseballs[0];
    if(type === ITEM_PLUS2) {
        spawnExtraBalls(base, 2);
    } else if(type === ITEM_MULT2) {
        const snapshot= [...balls];
        for(const b of snapshot) {
            spawnExtraBalls(b, 1);
        }
    } else if(type === ITEM_PADDLE) {
        enlargePaddle();
    }
}
function spawnExtraBalls(base, count) {
    const speedMath.sqrt(base.dx** 2base.dy** 2);
    const baseAngleMath.atan2(base.dy, base.dx);
    const spread10Math.PI180;
    for(let i0; i < count; i++) {
        const anglebaseAngle+ (i === 0? -spreadspread);
        balls.push({
            x:base.x,
            y:base.y,
            dx:speedMath.cos(angle),
            dy:speedMath.sin(angle),
            radius:BALL_RADIUS
        });
    }
}
/*
페들 버프
=====
*/
```

```
function enlargePaddle() {
    const centerpaddleXpaddleWidth2;
    paddleWidthENLARGED_PADDLE_WIDTH;
    paddleXcenterpaddleWidth2;
    paddleXMath.max(0, Math.min(paddleX, canvas.widthpaddleWidth));
    isPaddleEnlargedtrue;
    paddleEnlargeEndTimeDate.now() + 5000;
}
function updatePaddleEnlarge() {
    if(!isPaddleEnlarged) return;
    if(Date.now() > paddleEnlargeEndTime) {
        const centerpaddleXpaddleWidth2;
        paddleWidthORIGINAL_PADDLE_WIDTH;
        paddleXcenterpaddleWidth2;
        isPaddleEnlargedfalse;
    }
}
/*
= 업데이트 =
*/
function updateBalls() {
    if(ballStuck) {
        const bballs[0];
        b.xpaddleXpaddleWidth2;
        b.ycanvas.heightPADDLE_MARGIN_BOTTOMPADDLE_HEIGHTBALL_RADIUS;
        return;
    }
    const alive= [];
    for(const ball of balls) {
        // 벽
        if(ball.xball.dx < ball.radius|| ball.xball.dx >
        canvas.widthball.radius)
            ball.dx*= -1;
        if(ball.yball.dy < ball.radius)
            ball.dy*= -1;
        handlePaddleCollision(ball);
        handleBrickCollision(ball);
        ball.x+= ball.dx;
        ball.y+= ball.dy;
    }
}
```

```

        if(ball.y < canvas.heightball.radius) alive.push(ball);
    }
    ballsalive;
    if(balls.length === 0) {
        gameOver();
    }
}
function handlePaddleCollision(ball) {
    const topcanvas.heightPADDLE_MARGIN_BOTTOMPADDLE_HEIGHT;
    if(
        ball.yball.radius >= top &&
        ball.x >= paddleX &&
        ball.x <= paddleXpaddleWidth &&
        ball.dy > 0
    ) {
        const hitPos= (ball.x- (paddleXpaddleWidth2)) / (paddleWidth2);
        const anglehitPos* (75Math.PI180);
        const speedMath.sqrt(ball.dx** 2ball.dy** 2);
        ball.dx=speedMath.sin(angle);
        ball.dy= -Math.abs(speedMath.cos(angle));
    }
}
function handleBrickCollision(ball) {
    if(!bricks.length) return;
    for(let r0; r < bricks.length; r++) {
        const rowbricks[r];
        if(!row) continue;
        for(let c0; c < row.length; c++) {
            const brow[c];
            if(!b || b.status==0) continue;
            if(
                ball.x > b.x &&
                ball.x < b.xBRICK_WIDTH &&
                ball.y > b.y &&
                ball.y < b.yBRICK_HEIGHT
            ) {
                ball.dy*= -1;
                b.status0;
                bricksRemaining--;
            }
        }
    }
}

```

```
maybeDropItem(b.xBRICK_WIDTH2, b.y);
if(bricksRemaining<= 0) {
    isRunning=false;
    centerText.textContent="GPT 로고 올 클리어! Space / 클릭으로 다시 시작";
    centerText.style.opacity="1";
}
return;
}
}
}
}

/* =====
메인 루프
===== */
function updatePaddle() {
    if(rightPressed) paddleX+= 6;
    if(leftPressed) paddleX-= 6;
    paddleX=Math.max(0, Math.min(paddleX, canvas.width-paddleWidth));
}
function gameLoop() {
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    drawBricks();
    drawPaddle();
    drawBalls();
    drawItems();
    if(isRunning) {
        updatePaddle();
        updateBalls();
        updateItems();
        updatePaddleEnlarge();
    }
    requestAnimationFrame(gameLoop);
}
/* =====
입력 처리
===== */
document.addEventListener("keydown", e =>{
    if(e.code==="ArrowRight") rightPressed=true;
})
```

```
if(e.code === "ArrowLeft") leftPressedtrue;
if(e.code === "Space") startGame();
});

document.addEventListener("keyup", e =>{
  if(e.code === "ArrowRight") rightPressedfalse;
  if(e.code === "ArrowLeft") leftPressedfalse;
});

/* 마우스 이동 */
document.addEventListener("mousemove", e =>{
  const rectcanvas.getBoundingClientRect();
  const mouseXe.clientXrect.left;
  const scaleXcanvas.widthrect.width;
  const canvasXmouseXscaleX;
  paddleXcanvasXpaddleWidth2;
  paddleXMath.max(0, Math.min(paddleX, canvas.widthpaddleWidth));
});

canvas.addEventListener("click", startGame);
centerText.addEventListener("click", startGame);

function startGame() {
  // 클리어 후 다시 시작
  if(!isRunning && bricksRemaining <= 0) {
    resetGame();
  }
  // 게임 오버 후 다시 시작
  if(!isRunning && balls.length === 0) {
    resetGame();
  }
  ballStuckfalse;
  isRunningtrue;
  centerText.style.opacity = "0";
}

function gameOver() {
  isRunningfalse;
  centerText.textContent = "게임 오버! Space / 클릭으로 재시작";
  centerText.style.opacity = "1";
  // 공은 비워둔다. startGame에서 resetGame 호출.
  balls = [];
}

/* =====
```

```
로고 이미지 로드 후 게임 시작
=====
function loadLogoAndStart() {
    const imgnew Image();
    img.src="gpt_logo.png"; // 같은 폴더에 이 이름으로 저장해두기
    img.onload= () =>{
        createBrickMaskFromImage(img);
        resetGame();
        centerText.textContent="Space 또는 클릭해서 시작";
        gameLoop();
    };
    img.onerror= () =>{
        centerText.textContent=
            "gpt_logo.png 로고를 같은 폴더에 넣어야 합니다.";
        console.error("gpt_logo.png 파일을 찾을 수 없습니다.");
        // 그래도 게임 루프는 돌려둠 (벽돌 없이)
        gameLoop();
    };
}
loadLogoAndStart();
</script>
</body>
</html>
```

PS. 수정실습에서 GPT와의 대화는 짚은 글씨로 작성