**Falling Dot Hunter (떨어지는 점 사냥꾼)**

**게임 제작 보고서**

**1. 프로젝트 개요**

* 게임 제목 : Falling Dot Hunter
* 제작자 : 김수종
* 제작 기간 : 2025 / 10 / 24 ~ 2025 / 10 / 23 ( 2일 )
* 사용 언어 : HTML, JAVAScript, CSS
* 팀 구성 : 개인
* 플랫폼 : Chorme, Edge 등

**2. 게임 기획 및 컨셉**

* 기획 의도 : 짧은 플레이 타임에 반복 플레이가 가능한 “손맛 중심” 클릭 액션 게임. 떨어지는 점(닷)을 처리해 리듬감 있게 점수를 쌓고, 보스를 연타로 격파해 점진적 성장을 느끼게 함.
* 장르 : 아케이드/클리커(액션).
* 핵심 아이디어 :
* 라운드마다 떨어지는 공을 클릭해 1점씩 획득
* 모든 공 처리 후 “경고 연출 → 보스전”. 보스는 레벨×10회 클릭으로 처치.
* 보스 처치 시 레벨이 1 상승. 다음 라운드로 루프.
* 타겟 유저/플레이 스타일 :
* 연타 손맛을 선호하는 캐주얼 유저.
* 짧게 자주 즐기는 스타일. 정확도보다 빠른 클릭에 초점

**3. 게임 시스템 구조**

* 게임 진행 방식 :

시작 연출(GAME START) → 라운드 시작

라운드: 공 30개가 순차 낙하. 클릭 시 +1점, 놓치면 목숨 감소

라운드 종료: 화면에 공이 남지 않으면 경고(WARNING, 화면 붉은 점멸)

보스전: 제한시간 25초, 보스 HP는 “레벨×10(클릭 수)”로 표시, 색상 랜덤

보스 처치: 레벨+1, LEVEL UP 연출 후 다음 라운드

시간초과/미스 누적: 목숨 0이면 Game Over

* 다이어그램 :

도표, 텍스트, 스크린샷, 기술 도면이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**4. 기술적 구현 내용**

* 주요 스크립트 설명 :

js/game.js: 메인 루프, 상태머신(intro/round/warning/boss/levelup), HUD 그리기, 입력 처리, 재시작 버튼 리스너

js/circles.js: 라운드별 공 스폰/낙하/클릭 판정. 라운드 개수(30), 남은 스폰 수·화면 잔여 여부로 라운드 종료 판단

js/boss.js: 단순 보스(큰 원) 이동, 피격 펄스, HP(=레벨×10)와 제한시간, 중앙에 남은 클릭 수 표시, 타이머 우상단 표시

js/effects.js: 클릭 이펙트 링 생성/업데이트/정리

js/background.js: 배경 연출(파티클/그라디언트 등)

html/css: index.html(메인 화면), game.html(게임 화면), menu.css/style.css(스타일)

* 클래스/컴포넌트 설명 + 간단한 코드 스냅샷

- 전역 네임스페이스 Game 사용(IIFE 모듈)

G.startRound = function (totalCount) {

roundTotal = totalCount; spawned = 0; circles.length = 0;

};

G.roundTrySpawn = function () {

const need = Math.max(MIN\_ACTIVE, 2 + Math.floor((getLevel() - 1) / 3));

while (circles.length < need && spawned < roundTotal) {

circles.push(makeCircle()); spawned++;

}

};

G.isRoundDone = function () {

return spawned >= roundTotal && circles.length === 0;

};

- 보스 로직(레벨×10 클릭, 중앙 표시, 타이머 우상단)

G.spawnBossNow = function () {

const maxHp = Math.max(10, getLevel() \* 10); // 레벨마다 +10

boss = { /\* x,y,r,vx,vy 등 \*/, color: pick(PALETTE), maxHp, hp: maxHp, timeLeft: 25, pulseT: 0, scale: 1 };

};

G.drawBoss = function () {

// 본체 그리기...

// HP 남은 클릭 수를 중앙에 크게

ctx.font = "bold 26px Segoe UI, Arial";

ctx.lineWidth = 6; ctx.strokeStyle = "rgba(0,0,0,0.55)"; ctx.strokeText(`${boss.hp}`, 0, 0);

ctx.fillStyle = "#fff"; ctx.fillText(`${boss.hp}`, 0, 0);

// 우상단 타이머 캡슐

const label = `BOSS ${boss.timeLeft.toFixed(1)}s`;

- 게임 루프/상태/재시작

let level = 1;

const getLevel = () => level;

// 상태 전환

if (state === "intro") { /\* 오버레이 후 Game.startRound(30) \*/ }

else if (state === "round") { Game.roundTrySpawn(); Game.updateCircles(dt, 1); if (Game.isRoundDone()) { state="warning"; } }

else if (state === "warning") { /\* WARNING 연출 후 boss 스폰 \*/ }

else if (state === "boss") { Game.updateBoss(dt); Game.drawBoss(); }

else if (state === "levelup") { /\* LEVEL UP 연출 후 다음 라운드 \*/ }

// 보스 콜백

Game.initBoss(canvas, ctx, getLevel,

() => { level += 1; state = "levelup"; overlayTimer = 1.5; overlayText = `LEVEL ${level}`; },

() => { lives--; state = lives<=0 ? (gameOver(), "over") : (Game.startRound(30), "round"); },

Game.spawnEffect

);

// 안전 재시작(페이지 리로드)

document.getElementById("btnRestart")?.addEventListener("click", () => window.location.reload());

* 문제 해결 경험 :

Score와 보스 타이머 겹침: 타이머를 캔버스 우측 상단으로 이동하고 점수 배지에 pointer-events: none 적용해 클릭 방해 제거.

가속 문제: 시간 경과에 따른 전역 속도 가속을 제거하고 “보스 처치 시 레벨+1”로 난이도 상승을 일관되게 설계.

점수 일원화: 모든 공 클릭 점수를 1점으로 통일해 가시성/학습 비용 감소.

HUD 개편: 외부 점수 배지, 내부 레벨/하트 배치. 하트 간격(space) 조정으로 가독성 개선.

재시작 기능: 초기에는 내부 상태만 리셋하려다 참조 정리 이슈로 에러가 발생. 가장 안전한 방법으로 페이지 리로드 방식(window.location.reload) 채택해 런타임 상태를 완전 초기화.

콘솔 에러 처리: …(유니코드 줄임표) 같은 플레이스홀더로 인한 구문 에러 발생 → 파일 전체 검색/정리, 전역 네임스페이스(IIFE) 일관화(effects.js 등)로 undefined 초기화 함수 호출 방지.

* 구현 중 겪었던 오류/버그, 어떻게 해결했는지 기술

증상: 게임 시작 시 Uncaught Error (가장 많이 발생한 에러)

원인: 파일에 유니코드 줄임표(…) 같은 플레이스홀더가 남아 구문 오류 발생.

해결: game.js를 유효 코드로 전면 교체하고 스크립트 로딩 순서 재점검(background → effects → circles → boss → game).

증상: Game.initEffects is not a function

증상: 점수 배지와 보스 타이머가 겹침

원인: 둘 다 상단 중앙에 위치.

해결: 보스 타이머를 캔버스 우측 상단으로 이동. 점수 배지에 pointer-events: none으로 클릭 방해 제거.

증상: 보스 처치해도 점수가 오르지 않음

원인: 처치 콜백에 보상 값 전달 누락.

해결: boss.js에서 onDefeat에 boss.maxHp 전달, game.js에서 reward를 점수에 가산.

**5. 디자인**

* UI 구성 :

점수: 캔버스 밖 상단 중앙 고정 배지

레벨: 캔버스 상단 좌측 배지

목숨: 캔버스 하단 중앙 하트(3칸)

재시작: 캔버스 하단 버튼으로 페이지 새로고침(안전 리셋).

**6. 테스트 및 피드백**

* 테스트 방법 : 자가 테스트
* 발견된 문제점 및 개선 : 보스가 단숙 노가다 클릭이라 여러가지 패턴이 있었으면 좋겠음. 하지만 라운드마다 다 다른 보스, 다른 패턴을 구현하려면 한계가 있음.

**7. 회고 및 느낀 점**

* 어떤 점이 가장 어려웠고, 어떤 점에서 성취감을 느꼈는가?

게임은 단순했지만 아이디어를 생각해내는 것이 가장 어려웠다.

특히 보스 기능을 좀 제대로 구현해보고 싶었지만 능력이 부족하여 구현을 못한점이 아쉬웠다.

* 다음에 만들고 싶은 게임 아이디어

뱀서류 게임을 만들면 좋겠다고 생각했다.

* 학습적으로 어떤 성장을 했는가

AI의 도움으로 여러 코드 형식들을 배울 수 있었다.

**8. 스크린샷 및 게임 URL**

* 주요 장면 스크린샷
* 기능별 화면

게임 메인 화면 (기본적인 게임 설명 포함)

텍스트, 스크린샷, 폰트, 운영 체제이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

게임 화면 (인게임, 게임오버, 보스)

스크린샷, 텍스트, 정보기기, 전자 기기이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다. 텍스트, 스크린샷, 정보기기, 멀티미디어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다. 텍스트, 스크린샷, 멀티미디어, 정보기기이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

* URL 주소 : [**https://rlatnwhd.github.io/Falling-Dot-Hunter/**](https://rlatnwhd.github.io/Falling-Dot-Hunter/)

**9. 코드**

* js (파일명 : background.js )

(function (G) {

let canvas, ctx, getLevel;

let bgDots = [];

G.initBackground = function (c, context2d, levelGetter) {

canvas = c; ctx = context2d; getLevel = levelGetter;

bgDots = Array.from({ length: 60 }, () => ({

x: Math.random() \* canvas.width,

y: Math.random() \* canvas.height,

r: Math.random() \* 2 + 0.6,

s: Math.random() \* 20 + 10,

a: Math.random() \* 0.6 + 0.2

}));

};

G.drawBackground = function (dt) {

const g = ctx.createLinearGradient(0, 0, 0, canvas.height);

g.addColorStop(0, "#0f2027");

g.addColorStop(0.5, "#203a43");

g.addColorStop(1, "#2c5364");

ctx.fillStyle = g;

ctx.fillRect(0, 0, canvas.width, canvas.height);

for (const p of bgDots) {

p.y += (p.s + getLevel() \* 6) \* dt;

if (p.y > canvas.height + 5) { p.y = -5; p.x = Math.random() \* canvas.width; }

ctx.beginPath();

ctx.fillStyle = `rgba(255,255,255,${p.a})`;

ctx.arc(p.x, p.y, p.r, 0, Math.PI \* 2);

ctx.fill();

}

const v = ctx.createRadialGradient(

canvas.width / 2, canvas.height / 2, Math.min(canvas.width, canvas.height) / 2.2,

canvas.width / 2, canvas.height / 2, Math.min(canvas.width, canvas.height) / 1.05

);

v.addColorStop(0, "rgba(0,0,0,0)");

v.addColorStop(1, "rgba(0,0,0,0.35)");

ctx.fillStyle = v;

ctx.fillRect(0, 0, canvas.width, canvas.height);

};

})(window.Game = window.Game || {});

* js (파일명 : boss.js )

(function (G) {

let canvas, ctx, getLevel;

let bgDots = [];

G.initBackground = function (c, context2d, levelGetter) {

canvas = c; ctx = context2d; getLevel = levelGetter;

bgDots = Array.from({ length: 60 }, () => ({

x: Math.random() \* canvas.width,

y: Math.random() \* canvas.height,

r: Math.random() \* 2 + 0.6,

s: Math.random() \* 20 + 10,

a: Math.random() \* 0.6 + 0.2

}));

};

G.drawBackground = function (dt) {

const g = ctx.createLinearGradient(0, 0, 0, canvas.height);

g.addColorStop(0, "#0f2027");

g.addColorStop(0.5, "#203a43");

g.addColorStop(1, "#2c5364");

ctx.fillStyle = g;

ctx.fillRect(0, 0, canvas.width, canvas.height);

for (const p of bgDots) {

p.y += (p.s + getLevel() \* 6) \* dt;

if (p.y > canvas.height + 5) { p.y = -5; p.x = Math.random() \* canvas.width; }

ctx.beginPath();

ctx.fillStyle = `rgba(255,255,255,${p.a})`;

ctx.arc(p.x, p.y, p.r, 0, Math.PI \* 2);

ctx.fill();

}

const v = ctx.createRadialGradient(

canvas.width / 2, canvas.height / 2, Math.min(canvas.width, canvas.height) / 2.2,

canvas.width / 2, canvas.height / 2, Math.min(canvas.width, canvas.height) / 1.05

);

v.addColorStop(0, "rgba(0,0,0,0)");

v.addColorStop(1, "rgba(0,0,0,0.35)");

ctx.fillStyle = v;

ctx.fillRect(0, 0, canvas.width, canvas.height);

};

})(window.Game = window.Game || {});

* js (파일명 : circles.js )

(function (G) {

let canvas, ctx, getLevel, onMiss;

const circles = [];

// 라운드 관리

let roundTotal = 0; // 이번 라운드에 떨어질 총 개수(예: 15)

let spawned = 0; // 지금까지 스폰한 개수

const MIN\_ACTIVE = 2; // 동시에 화면에 최소 유지

const types = [

{ color: "red", value: 1, w: 3 },

{ color: "blue", value: 1, w: 4 },

{ color: "green", value: 1, w: 2 },

{ color: "yellow",value: 1, w: 1 },

{ color: "purple",value: 1, w: 1 }

];

G.initCircles = function (c, context2d, levelGetter, onMissCb) {

canvas = c; ctx = context2d; getLevel = levelGetter; onMiss = onMissCb;

};

function pickType() {

const total = types.reduce((s, t) => s + t.w, 0);

let r = Math.random() \* total;

for (const t of types) { if ((r -= t.w) <= 0) return t; }

return types[0];

}

function getRadiusByLevel() { return Math.max(10, 30 - (getLevel() - 1) \* 3); }

function makeCircle() {

const { color, value } = pickType();

const r = getRadiusByLevel();

const x = Math.random() \* (canvas.width - r \* 2) + r;

const y = -r - Math.random() \* 50;

const base = 80 + Math.random() \* 80; // 80~160 px/s

const speed = base \* (1 + (getLevel() - 1) \* 0.15); // 레벨 가중

return { x, y, r, color, value, speed };

}

// 라운드 시작

G.startRound = function (totalCount) {

roundTotal = totalCount;

spawned = 0;

circles.length = 0;

};

// 스폰(동시 개수 유지, 총량 초과 금지)

G.roundTrySpawn = function () {

const need = Math.max(MIN\_ACTIVE, 2 + Math.floor((getLevel() - 1) / 3));

while (circles.length < need && spawned < roundTotal) {

circles.push(makeCircle());

spawned++;

}

};

// 진행 업데이트(낙하/그리기)

G.updateCircles = function (dt, globalSpeed) {

for (let i = circles.length - 1; i >= 0; i--) {

const c = circles[i];

c.y += c.speed \* globalSpeed \* dt;

ctx.beginPath();

ctx.arc(c.x, c.y, c.r, 0, Math.PI \* 2);

ctx.fillStyle = c.color;

ctx.fill();

// 화면 아래로 사라짐(미스)

if (c.y - c.r > canvas.height) {

onMiss && onMiss();

if (spawned < roundTotal) {

circles[i] = makeCircle(); // 아직 스폰 남았으면 대체

spawned++;

} else {

circles.splice(i, 1); // 더 스폰 불가면 제거

}

}

}

};

// 클릭 판정

G.clickCircles = function (mx, my, spawnEffect) {

for (let i = circles.length - 1; i >= 0; i--) {

const c = circles[i];

const dx = mx - c.x, dy = my - c.y;

if (dx \* dx + dy \* dy <= c.r \* c.r) {

const add = 1; // 모든 색 1점

spawnEffect && spawnEffect(c.x, c.y, c.color);

if (spawned < roundTotal) {

circles[i] = makeCircle();

spawned++;

} else {

circles.splice(i, 1);

}

return add;

}

}

return 0;

};

// 라운드 종료 조건: 지정 개수 모두 스폰했고, 화면에 남은 공이 없음

G.isRoundDone = function () {

return spawned >= roundTotal && circles.length === 0;

};

})(window.Game = window.Game || {});

* js (파일명 : effects.js )

(function (G) {

let ctx;

const effects = [];

G.initEffects = function (c, context2d) {

ctx = context2d;

};

G.spawnEffect = function (x, y, color) {

effects.push({ x, y, radius: 8, alpha: 1, color });

};

G.updateEffects = function (dt) {

for (let i = effects.length - 1; i >= 0; i--) {

const e = effects[i];

e.radius += 120 \* dt;

e.alpha -= 2 \* dt;

if (e.alpha <= 0) { effects.splice(i, 1); continue; }

ctx.save();

ctx.globalAlpha = Math.max(0, e.alpha);

ctx.strokeStyle = e.color;

ctx.lineWidth = 3;

ctx.beginPath();

ctx.arc(e.x, e.y, e.radius, 0, Math.PI \* 2);

ctx.stroke();

ctx.restore();

}

};

})(window.Game = window.Game || {});

* js (파일명 : game.js )

const canvas = document.getElementById("gameCanvas");

const ctx = canvas.getContext("2d");

const scoreSpan = document.getElementById("score");

const livesSpan = document.getElementById("lives");

const levelSpan = document.getElementById("level");

let score = 0, lives = 3;

let elapsed = 0, lastTime = 0;

let running = true;

let level = 1; // 추가: 수동 레벨

const getLevel = () => level; // 변경: 경과시간 기반 → 수동 레벨 반환

// 시스템 초기화

Game.initBackground(canvas, ctx, getLevel);

Game.initEffects(canvas, ctx);

Game.initCircles(canvas, ctx, getLevel, () => { lives--; if (lives <= 0) gameOver(); });

Game.initBoss(

canvas, ctx, getLevel,

(reward) => { // 보스 처치

score += reward || 0; // 보스 시작 체력만큼 점수 추가

level += 1;

state = "levelup";

overlayTimer = 1.5;

overlayText = `LEVEL ${level}`;

},

() => { // 시간초과

lives--; if (lives <= 0) { gameOver(); return; }

state = "round";

Game.startRound(30);

},

Game.spawnEffect

);

// 상태 머신

let state = "intro"; // intro → round → warning → boss → levelup → round ...

let overlayTimer = 1.4; // 오버레이 텍스트 표시 시간

let overlayText = "GAME START";

let warningTimer = 0;

// 중앙 큰 텍스트 + 간단한 애니메이션

function drawCenterText(text, t) {

// t: 0~1 (등장/퇴장 페이드)

const alpha = t < 0.2 ? t / 0.2 : t > 0.8 ? (1 - t) / 0.2 : 1;

const scale = 1 + 0.08 \* Math.sin(t \* Math.PI); // 은은한 펄스

ctx.save();

ctx.translate(canvas.width / 2, canvas.height / 2);

ctx.scale(scale, scale);

ctx.fillStyle = `rgba(255,255,255,${Math.max(0, Math.min(1, alpha))})`;

ctx.font = "bold 36px Pretendard, Segoe UI, Arial";

ctx.textAlign = "center";

ctx.textBaseline = "middle";

ctx.fillText(text, 0, 0);

ctx.restore();

}

// 경고(빨간 플래시)

function drawWarningOverlay(timeLeft, total = 2.5) {

const t = 1 - timeLeft / total; // 0->1

const flash = 0.35 + 0.35 \* (0.5 + 0.5 \* Math.sin(t \* 20)); // 빠른 점멸

ctx.save();

ctx.fillStyle = `rgba(255,0,0,${flash})`;

ctx.fillRect(0, 0, canvas.width, canvas.height);

ctx.restore();

// 텍스트

ctx.save();

ctx.fillStyle = "#fff";

ctx.font = "bold 28px Arial";

ctx.textAlign = "center";

ctx.textBaseline = "middle";

ctx.fillText("WARNING", canvas.width/2, canvas.height/2 - 22);

ctx.font = "bold 22px Arial";

ctx.fillText("보스전", canvas.width/2, canvas.height/2 + 16);

ctx.restore();

}

// HUD 그리기 도우미 -------------------------------------------

function roundRectPath(x, y, w, h, r) {

const rr = Math.min(r, w/2, h/2);

ctx.beginPath();

ctx.moveTo(x + rr, y);

ctx.arcTo(x + w, y, x + w, y + h, rr);

ctx.arcTo(x + w, y + h, x, y + h, rr);

ctx.arcTo(x, y + h, x, y, rr);

ctx.arcTo(x, y, x + w, y, rr);

ctx.closePath();

}

function drawHeart(x, y, size, fill, stroke, alpha = 1) {

// size: 전체 크기(px). (x, y)는 중심

const r = size;

ctx.save();

ctx.globalAlpha = alpha;

ctx.beginPath();

ctx.moveTo(x, y + r \* 0.25);

ctx.bezierCurveTo(x + r \* 0.5, y - r \* 0.5, x + r \* 1.2, y + r \* 0.3, x, y + r);

ctx.bezierCurveTo(x - r \* 1.2, y + r \* 0.3, x - r \* 0.5, y - r \* 0.5, x, y + r \* 0.25);

ctx.closePath();

if (fill) {

const g = ctx.createLinearGradient(x, y - r, x, y + r);

g.addColorStop(0, fill);

g.addColorStop(1, "#b32638");

ctx.fillStyle = g;

ctx.fill();

}

if (stroke) {

ctx.lineWidth = 2;

ctx.strokeStyle = stroke;

ctx.stroke();

}

ctx.restore();

}

// HUD 그리기

function drawHUD() {

// 외부 DOM 점수는 계속 갱신(상단 중앙 배지)

if (scoreSpan) scoreSpan.textContent = score;

if (livesSpan) livesSpan.textContent = lives;

if (levelSpan) levelSpan.textContent = getLevel();

const M = 10;

// 상단 좌측: 레벨 배지

ctx.save();

const lv = getLevel();

const lvText = `LV ${lv}`;

ctx.font = "bold 14px Segoe UI, Arial";

const lvTw = ctx.measureText(lvText).width;

const lvW = lvTw + 18, lvH = 24;

const lvX = M, lvY = M;

roundRectPath(lvX, lvY, lvW, lvH, 10);

const lg = ctx.createLinearGradient(lvX, lvY, lvX, lvY + lvH);

lg.addColorStop(0, "rgba(255,255,255,0.10)");

lg.addColorStop(1, "rgba(255,255,255,0.02)");

ctx.fillStyle = lg;

ctx.fill();

ctx.strokeStyle = "rgba(255,255,255,0.15)";

ctx.stroke();

ctx.fillStyle = "#d7e3ff";

ctx.textAlign = "center";

ctx.textBaseline = "middle";

ctx.fillText(lvText, lvX + lvW / 2, lvY + lvH / 2);

ctx.restore();

// 하단 중앙: 목숨 하트(간격 넓힘)

ctx.save();

const MAX\_LIVES = 3;

const heartSize = 16;

const space = 28; // 간격 넓힘

const panelW = MAX\_LIVES \* heartSize + (MAX\_LIVES - 1) \* (space - heartSize) + 24;

const panelH = 34;

const panelX = (canvas.width - panelW) / 2;

const panelY = canvas.height - panelH - M;

roundRectPath(panelX, panelY, panelW, panelH, 12);

ctx.fillStyle = "rgba(0,0,0,0.45)";

ctx.fill();

ctx.strokeStyle = "rgba(255,255,255,0.14)";

ctx.stroke();

const startX = panelX + (panelW - ((MAX\_LIVES - 1) \* space + heartSize)) / 2 + heartSize / 2;

const y = panelY + panelH / 2 - 2;

for (let i = 0; i < MAX\_LIVES; i++) {

const cx = startX + i \* space;

if (i < lives) {

drawHeart(cx, y, heartSize, "#ff5a6d", "#ffb4c0", 1);

} else {

drawHeart(cx, y, heartSize, null, "rgba(255,255,255,0.5)", 0.7);

}

}

ctx.restore();

}

function update(now) {

if (!running) return;

if (!lastTime) lastTime = now;

const dt = Math.min(0.05, (now - lastTime) / 1000);

lastTime = now;

elapsed += dt;

const globalSpeed = 1; // 변경: 시간 가속 제거(원 속도는 circles.js에서 레벨로 반영)

// 공통 배경

Game.drawBackground(dt);

if (state === "intro") {

overlayTimer -= dt;

drawCenterText(overlayText, Math.max(0, 1 - overlayTimer / 1.4));

if (overlayTimer <= 0) {

state = "round";

Game.startRound(30); // 라운드당 30개 유지

}

}

else if (state === "round") {

Game.roundTrySpawn();

Game.updateCircles(dt, globalSpeed);

if (Game.isRoundDone()) {

state = "warning";

warningTimer = 2.5; // 경고 연출 길이

}

}

else if (state === "warning") {

warningTimer -= dt;

drawWarningOverlay(warningTimer, 2.5);

if (warningTimer <= 0) {

state = "boss";

Game.spawnBossNow();

}

}

else if (state === "boss") {

Game.updateBoss(dt);

Game.drawBoss();

// 승패 처리는 콜백에서 상태 변경

}

else if (state === "levelup") {

overlayTimer -= dt;

drawCenterText(overlayText, Math.max(0, 1 - overlayTimer / 1.5));

if (overlayTimer <= 0) {

state = "round";

Game.startRound(30); // 다음 라운드 시작

}

}

Game.updateEffects(dt);

drawHUD();

if (running) requestAnimationFrame(update);

}

function gameOver() {

running = false;

ctx.save();

ctx.fillStyle = "rgba(0,0,0,0.6)";

ctx.fillRect(0, 0, canvas.width, canvas.height);

ctx.fillStyle = "#fff";

ctx.font = "28px Arial";

ctx.textAlign = "center";

ctx.textBaseline = "middle";

ctx.fillText(`Game Over`, canvas.width / 2, canvas.height / 2 - 10);

ctx.font = "18px Arial";

ctx.fillText(`Score: ${score}`, canvas.width / 2, canvas.height / 2 + 22);

ctx.restore();

}

// 입력

canvas.addEventListener("click", (e) => {

if (!running) return;

const rect = canvas.getBoundingClientRect();

const mx = e.clientX - rect.left, my = e.clientY - rect.top;

if (state === "boss") {

if (Game.hitBoss(mx, my)) return;

}

if (state === "round") {

const gained = Game.clickCircles(mx, my, Game.spawnEffect);

if (gained) score += gained;

}

});

requestAnimationFrame(update);

// 재시작 버튼: 페이지 전체 리로드(상태 완전 초기화)

(() => {

const btn = document.getElementById('btnRestart');

if (btn) btn.addEventListener('click', () => {

window.location.reload(); // 에러 없이 깔끔한 재시작

});

})();

* html (파일명 : game.html )

<!DOCTYPE html>

<html lang="ko">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<link rel="stylesheet" href="css/style.css" />

</head>

<body>

<canvas id="gameCanvas" width="400" height="600"></canvas>

<div id="scoreDisplay">점수: <span id="score">0</span></div>

<div class="controls">

<button id="btnRestart" class="btn-restart">게임 재시작</button>

</div>

<!-- 전역 네임스페이스 방식: 로딩 순서 중요 -->

<script src="js/background.js"></script>

<script src="js/effects.js"></script>

<script src="js/circles.js"></script>

<script src="js/boss.js"></script>

<script src="js/game.js"></script>

</body>

</html>

* html (파일명 : index.html )

<!DOCTYPE html>

<html lang="ko">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<title>Falling Dot Hunter - 메인</title>

<link rel="stylesheet" href="css/menu.css" />

</head>

<body>

<div class="bg">

<div class="orb orb-1"></div>

<div class="orb orb-2"></div>

<div class="orb orb-3"></div>

</div>

<main class="wrap">

<header class="brand">

<h1 class="logo">Falling Dot Hunter</h1>

<p class="tag">스피드 클릭커 게임</p>

</header>

<section class="card">

<div class="card\_\_body">

<h2>게임 설명</h2>

<ul class="desc">

<li>라운드마다 공이 30개 떨어집니다. 공을 클릭해 점수를 얻으세요.</li>

<li>모든 공이 사라지면 보스전이 시작됩니다.</li>

<li>보스는 레벨 × 10회 클릭 시 처치됩니다.</li>

<li>보스 제한 시간은 30초입니다. 처치하면 레벨이 1 상승합니다.</li>

<li>조작: 마우스 클릭만 사용합니다.</li>

</ul>

<a class="btn btn-start" href="game.html">게임 시작</a>

<div class="tips">

<h3>TIP</h3>

<p>정확도보다 연타가 중요합니다. 큰 원의 중심에 가깝게 연타하면 안정적으로 히트를 넣을 수 있어요.</p>

</div>

</div>

</section>

<footer class="footer">

<span>© Silla University 2025</span>

</footer>

</main>

</body>

</html>

* css (파일명 : menu.css )

:root{

--bg-1:#0f2027;

--bg-2:#203a43;

--bg-3:#2c5364;

--text:#eaf2ff;

--muted:#a9b7c6;

--accent:#59d6ff;

--btn:#6dd5fa;

--btn-2:#2980b9;

}

\*{ box-sizing:border-box; }

html,body{ height:100%; }

body{

margin:0;

color:var(--text);

font-family: "Pretendard", "Segoe UI", "Noto Sans KR", system-ui, -apple-system, Arial, sans-serif;

background: radial-gradient(1200px 700px at 20% -10%, #1a2a3a 0%, transparent 60%),

radial-gradient(900px 600px at 120% 110%, #0a2a2a 0%, transparent 60%),

linear-gradient(160deg, var(--bg-1), var(--bg-2) 50%, var(--bg-3));

overflow: hidden;

}

/\* 뒤 배경 점 패턴 + 오브 글로우 \*/

.bg::after{

content:"";

position:fixed; inset:0;

background:

radial-gradient(circle at 20px 20px, rgba(255,255,255,0.08) 2px, transparent 3px) 0 0/40px 40px,

radial-gradient(circle at 20px 20px, rgba(0,0,0,0.22) 2px, transparent 3px) 20px 20px/40px 40px;

opacity:.18; pointer-events:none;

}

.orb{

position:fixed; border-radius:50%; filter: blur(40px); opacity:.65; mix-blend-mode: screen;

animation: float 14s ease-in-out infinite;

}

.orb-1{ width:260px; height:260px; background:#6dd5fa; top:8%; left:8%; animation-delay:-2s; }

.orb-2{ width:320px; height:320px; background:#a78bfa; right:10%; top:18%; animation-delay:-6s; }

.orb-3{ width:220px; height:220px; background:#ffd166; left:14%; bottom:12%; animation-delay:-10s; }

@keyframes float{

0%,100%{ transform: translateY(-6px); }

50%{ transform: translateY(6px); }

}

.wrap{

position:relative;

max-width:920px;

margin:0 auto;

padding:56px 18px 28px;

min-height:100%;

display:flex;

flex-direction:column;

gap:22px;

}

.brand{ text-align:center; margin-top:10px; }

.logo{

margin:0 0 6px;

font-size: clamp(28px, 6vw, 48px);

letter-spacing: .5px;

text-shadow: 0 8px 24px rgba(0,0,0,.45);

}

.tag{

margin:0; color:var(--muted);

font-size: clamp(14px, 2.6vw, 16px);

}

.card{

margin: 0 auto;

width:min(860px, 92%);

background: rgba(0,0,0,.28);

border:1px solid rgba(255,255,255,.16);

border-radius:20px;

box-shadow: 0 20px 60px rgba(0,0,0,.35), inset 0 0 40px rgba(255,255,255,.04);

backdrop-filter: blur(10px);

-webkit-backdrop-filter: blur(10px);

}

.card\_\_body{ padding:26px 22px 30px; }

.card h2{ margin:0 0 12px; font-size:22px; }

.desc{

margin:0 0 18px 0; padding-left:18px; line-height:1.7;

}

.desc li{ margin:6px 0; color:#e9f2ff; }

.desc li::marker{ color:#8cd3ff; }

.btn{

display:inline-block;

text-decoration:none;

color:#06202e;

font-weight:800;

letter-spacing:.4px;

padding:13px 22px;

border-radius:999px;

background: linear-gradient(180deg, var(--btn), var(--btn-2));

box-shadow: 0 14px 30px rgba(0,0,0,.35), inset 0 0 18px rgba(255,255,255,.25);

border:1px solid rgba(255,255,255,.28);

transition: transform .15s ease, filter .2s ease, box-shadow .2s ease;

}

.btn:hover{ transform: translateY(-2px); filter: brightness(1.05); }

.btn:active{ transform: translateY(0); filter: brightness(.98); }

.btn-start{ font-size:18px; }

.tips{ margin-top:18px; color:#d7e3ff; }

.tips h3{ margin:0 0 6px; font-size:16px; color:#bfe3ff; }

.footer{

margin-top:auto;

text-align:center;

color:rgba(255,255,255,.6);

font-size:12px;

padding-bottom:10px;

}

@media (max-width:480px){

.card\_\_body{ padding:20px 16px 24px; }

.btn-start{ width:100%; text-align:center; }

}

:root{

--bg-1:#0f2027;

--bg-2:#203a43;

--bg-3:#2c5364;

--text:#eaf2ff;

--muted:#a9b7c6;

--accent:#59d6ff;

--btn:#6dd5fa;

--btn-2:#2980b9;

}

\*{ box-sizing:border-box; }

html,body{ height:100%; }

body{

margin:0;

color:var(--text);

font-family: "Pretendard", "Segoe UI", "Noto Sans KR", system-ui, -apple-system, Arial, sans-serif;

background: radial-gradient(1200px 700px at 20% -10%, #1a2a3a 0%, transparent 60%),

radial-gradient(900px 600px at 120% 110%, #0a2a2a 0%, transparent 60%),

linear-gradient(160deg, var(--bg-1), var(--bg-2) 50%, var(--bg-3));

overflow: hidden;

}

/\* 뒤 배경 점 패턴 + 오브 글로우 \*/

.bg::after{

content:"";

position:fixed; inset:0;

background:

radial-gradient(circle at 20px 20px, rgba(255,255,255,0.08) 2px, transparent 3px) 0 0/40px 40px,

radial-gradient(circle at 20px 20px, rgba(0,0,0,0.22) 2px, transparent 3px) 20px 20px/40px 40px;

opacity:.18; pointer-events:none;

}

.orb{

position:fixed; border-radius:50%; filter: blur(40px); opacity:.65; mix-blend-mode: screen;

animation: float 14s ease-in-out infinite;

}

.orb-1{ width:260px; height:260px; background:#6dd5fa; top:8%; left:8%; animation-delay:-2s; }

.orb-2{ width:320px; height:320px; background:#a78bfa; right:10%; top:18%; animation-delay:-6s; }

.orb-3{ width:220px; height:220px; background:#ffd166; left:14%; bottom:12%; animation-delay:-10s; }

@keyframes float{

0%,100%{ transform: translateY(-6px); }

50%{ transform: translateY(6px); }

}

.wrap{

position:relative;

max-width:920px;

margin:0 auto;

padding:56px 18px 28px;

min-height:100%;

display:flex;

flex-direction:column;

gap:22px;

}

.brand{ text-align:center; margin-top:10px; }

.logo{

margin:0 0 6px;

font-size: clamp(28px, 6vw, 48px);

letter-spacing: .5px;

text-shadow: 0 8px 24px rgba(0,0,0,.45);

}

.tag{

margin:0; color:var(--muted);

font-size: clamp(14px, 2.6vw, 16px);

}

.card{

margin: 0 auto;

width:min(860px, 92%);

background: rgba(0,0,0,.28);

border:1px solid rgba(255,255,255,.16);

border-radius:20px;

box-shadow: 0 20px 60px rgba(0,0,0,.35), inset 0 0 40px rgba(255,255,255,.04);

backdrop-filter: blur(10px);

-webkit-backdrop-filter: blur(10px);

}

.card\_\_body{ padding:26px 22px 30px; }

.card h2{ margin:0 0 12px; font-size:22px; }

.desc{

margin:0 0 18px 0; padding-left:18px; line-height:1.7;

}

.desc li{ margin:6px 0; color:#e9f2ff; }

.desc li::marker{ color:#8cd3ff; }

.btn{

display:inline-block;

text-decoration:none;

color:#06202e;

font-weight:800;

letter-spacing:.4px;

padding:13px 22px;

border-radius:999px;

background: linear-gradient(180deg, var(--btn), var(--btn-2));

box-shadow: 0 14px 30px rgba(0,0,0,.35), inset 0 0 18px rgba(255,255,255,.25);

border:1px solid rgba(255,255,255,.28);

transition: transform .15s ease, filter .2s ease, box-shadow .2s ease;

}

.btn:hover{ transform: translateY(-2px); filter: brightness(1.05); }

.btn:active{ transform: translateY(0); filter: brightness(.98); }

.btn-start{ font-size:18px; }

.tips{ margin-top:18px; color:#d7e3ff; }

.tips h3{ margin:0 0 6px; font-size:16px; color:#bfe3ff; }

.footer{

margin-top:auto;

text-align:center;

color:rgba(255,255,255,.6);

font-size:12px;

padding-bottom:10px;

}

@media (max-width:480px){

.card\_\_body{ padding:20px 16px 24px; }

.btn-start{ width:100%; text-align:center; }

}

* css (파일명 : style.css )

/\* 페이지 배경 \*/

body {

min-height: 100vh;

margin: 0;

font-family: "Segoe UI", system-ui, Arial, sans-serif;

background:

radial-gradient(circle at 20% 10%, rgba(255,255,255,0.08) 0 12%, transparent 13%) 0 0/80px 80px,

linear-gradient(135deg, #0f2027, #203a43 60%, #2c5364);

}

/\* 캔버스 카드 스타일 \*/

canvas {

border: 2px solid rgba(255,255,255,0.15);

border-radius: 16px;

display: block;

margin: 24px auto;

background: linear-gradient(180deg, #233242, #1a2633); /\* JS 비활성 시의 폴백 \*/

box-shadow:

0 12px 32px rgba(0,0,0,0.35),

inset 0 0 40px rgba(255,255,255,0.05);

}

#scoreDisplay {

/\* 상단 중앙 고정 배지 \*/

position: fixed;

top: 10px;

left: 50%;

transform: translateX(-50%);

z-index: 1000;

pointer-events: none; /\* 캔버스 클릭을 가리지 않음 \*/

padding: 8px 18px;

border-radius: 999px;

color: #f7f9fc;

font-size: 18px;

font-weight: 700;

background: rgba(0,0,0,0.45);

border: 1px solid rgba(255,255,255,0.18);

box-shadow: 0 8px 20px rgba(0,0,0,0.28), inset 0 0 24px rgba(255,255,255,0.06);

text-shadow: 0 2px 6px rgba(0,0,0,0.4);

backdrop-filter: blur(6px);

-webkit-backdrop-filter: blur(6px);

}

.controls{

margin: 10px 0 0;

display: flex;

justify-content: center;

}

.btn-restart{

padding: 10px 16px;

border-radius: 10px;

border: 1px solid rgba(255,255,255,0.22);

background: linear-gradient(180deg, #6dd5fa, #2980b9);

color: #06202e;

font-weight: 800;

letter-spacing: .3px;

cursor: pointer;

box-shadow: 0 10px 24px rgba(0,0,0,.28), inset 0 0 16px rgba(255,255,255,.22);

transition: transform .12s ease, filter .15s ease;

}

.btn-restart:hover{ transform: translateY(-1px); filter: brightness(1.05); }

.btn-restart:active{ transform: translateY(0); filter: brightness(.98); }