웹프로그래밍 게임 보고서

웹프로그래밍 벽돌깨기 게임 보고서

제출 : 2025-06-08

07조

202411292 박준건

202411235 강동훈

202411348 이시헌

202411343 이수현

1. 시나리오

> *과제와 시험을 전부 망친 나는 이제 남은 것이 없다.. 하지만 웹 프로그래밍은 상대평가니까, 다른 사람의 과제를 망치면 내 점수가 오르는 것이 아닐까? 굉장히 기발한 아이디어다! 나는 남의 과제를 망쳐서, 내 평균을 끌어올리기로 결심한다.*

1.1. 단계 정보

1. 쉬움에서는 우리가 가장 처음에 만들었던 실습을 제거하려고 시도한다. html 태그만을 제거합니다. 마우스로 바를 좌우로 움직이며 공을 튀겨 실습을 구성하는 태그(블럭)를 부숴나가면 되며, 중간중간 폭탄 블럭이 있어 광역으로 태그를 부술 수 있다.

2. 중간에서는 우리가 수업 중반때쯤 했던 playjs 실습을 망친다. 중간 단계에서는 또한 '여러 번 맞춰야 부숴지는 블럭' 이 나온다.

3. 어려움 단계는 제이쿼리를 망가트리는 것을 목표로 한다. 또한 시간제한이 생긴다. 폭탄과 강화 블럭, 시간 제한 속에서 본인이 할 수 있는 모든 것을 해보자.

2. 메뉴얼

game.html을 실행하면 실행할 수 있습니다.

2.1. 기본 조작법

P키 / E키 / Esc키 : 게임을 일시정지하거나, 해제한다.

Q키 : 즉시 메인메뉴로 이동한다.

R : 게임을 다시 시작한다.

본 게임에서는 공을 튀기는 바의 조작은 마우스로 할 수 있으며, 방향키를 통한 조작도 가능하다. \

2.2. 기믹 설명

폭탄 블럭 : 본체가 부숴지면 상 하 좌 우 네 블럭을 같이 부숩니다.

강화 블럭 : 3번 쳐야만 부숴집니다.

시간 제한 : 하드 모드부턴 시간 제한이 생깁니다.

3. 주요 기능 및 기능 구현을 위한 기술적 상세 명세

이 프로젝트의 핵심은, 두 개의 html 문서를 서로 상호작용시키며 변수를 업데이트, id/class로 구분된 객체들을 적제적소에 show 하고 hide하는 것이었습니다. 이를 위해 사용된 기술적 명세는 다음과 같습니다.

3.1. iframe 기반 연동 시스템

게임의 가장 핵심적인 특징은 iframe을 활용하여 저희가 지금까지 진행했던 HTML/CSS/JS 실습 코드를 실시간으로 로딩하고, 이를 게임과 연동, 실시간 수정하는 것입니다. 구체적인 원리는 다음과 같습니다.

```
const htmlCodeE = `
  <div id = "wrapper">
    <header id = "header">
      <h1 class = "title">
        웹 프로그래밍 완전 정복
      </h1>
      <h2 class = "subtitle">
        건국대학교 컴퓨터공학부
      </h2>
   --(중략)--
```

해당 html 이외에도, 본인 학번.html로 이용했던 실습 문서를 전부 str 형태로 불러옵니다.

```
let tempCss, tempJs, htmlCode, css, js;
switch (difficulty) {
case 0: {
  htmlCode = htmlCodeE;
  tempCss = cssCodeE;
  tempJs = jsCodeE;
```

```
  break;
}
case 1: {
  htmlCode = htmlCodeN;
  tempCss = cssCodeN;
  tempJs = jsCodeN;
}
case 2: {
  htmlCode = htmlCodeN;
  tempCss = cssCodeN;
  tempJs = jsCodeN;
}
}
css = `<style>${tempCss}</style>`;
js = `<script>${tempJs}<\/script>`;

const fullHTML = `
      <!DOCTYPE html>
      <html lang="en">
        <head>
          <meta charset="UTF-8" />
            ${css}
      <link rel="preconnect" href="https://fonts.googleapis.com">
      <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
      <link href="https://fonts.googleapis.com/css2?family=Sour+Gummy:ital,v
          </head>
        <body>
            ${htmlCode}
            ${js}
        </body>
      </html>
`;

const iframe = document.getElementById("labFrame");
const doc = iframe.contentDocument || iframe.contentWindow.document;
doc.open();
doc.write(fullHTML);
```

```
doc.close();
}
```

이후 필수적인 html 헤더를 더하고, iframe 내장 기능을 이용해 새로운 dom 트리를 만들고, 그 트리 안에 정리된 str을 넣어줍니다. 결과적으로 한 페이지에는 두 개의 페이지가 공존하게 됩니다.

이후, 외부 JS 스크립트에서 iframe 내부의 DOM 요소에 접근하고 제어함으로써, 특정 벽돌을 깰 때 실습 요소에 시각적 변화 또는 기능적인 반응을 연출할 수 있도록 구성하였습니다.

다음은 html과 css를 직접적으로 바꾸는 함수입니다.

```
/보조용 함수들 두개
function removeHtmlTagFromIframe(id) {
  const iframe = document.getElementById("labFrame");
  if (!iframe || !iframe.contentWindow || !iframe.contentDocument) return;

  const element = iframe.contentDocument.getElementById(id);
  if (element) {
    element.style.display = "none";
  } else {
    console.warn(`Element with id '${id}' not found in iframe.`);
  }
}



function changeCssTagFromIframe(id, cssProperty, value) {
  const iframe = document.getElementById("labFrame");
  if (!iframe || !iframe.contentWindow || !iframe.contentDocument) return;

  const element = iframe.contentDocument.getElementById(id);
  if (element) {
    element.style[cssProperty] = value;
  } else {
```

```
    console.warn(`Element with id '${id}' not found in iframe.`);
  }
}
```

이외에도, iframe에 직접적으로 선택자로 접근해서 값을 얻어내고, 변경하기도 하였습니다. 다음은 그 예시입니다.

iframe 에서 내부 css 변경하는 법

```
const iframe = document.getElementById("labFrame");
const iframeDoc = iframe.contentDocument || iframe.contentWindow.docume

// 원하는 요소 가져오기 (id, class 등으로)
const target = iframeDoc.querySelector("#가져올Id") || iframeDoc.querySelect

if (target) {
target.style.backgroundColor = "black";
target.style.color = "white";
}
```

예시: 단어장 색상 바꾸기

```
const target = iframeDoc.querySelector(".lab.wordBook");
if (target) {
  target.style.backgroundColor = "#222";
  target.style.color = "#0f0";
}
```

예시: 빨강 테두리만 강조

```
const textElem = iframeDoc.getElementById("innerTest");
if (textElem) {
  textElem.style.border = "2px solid red";
  textElem.style.padding = "5px";
}
```

예시: 색상표의 속성 흐리게

```
const boxes = iframeDoc.querySelectorAll(".lab.colorList .ctbox");
boxes.forEach(box => {
  box.style.filter = "grayscale(100%)";
  box.style.opacity = "0.5";
});
```

예시: 이미지 반전시키기

```
const img = iframeDoc.getElementById("image");
if (img) {
  img.style.filter = "invert(1)";
}
```

예시: 텍스트 깜빡이기 효과

```
const target = iframeDoc.getElementById("innerTest");
if (target) {
  target.style.animation = "blink 1s infinite";

  const style = iframeDoc.createElement("style");
  style.innerHTML = `
    @keyframes blink {
      0%, 100% { opacity: 1; }
      50% { opacity: 0; }
    }
  `;
  iframeDoc.head.appendChild(style);
}
```

기능 추가.

```
//난이도 별로 핸들러 세트
const allEffectHandlers = {
  0: { // Easy
    remove: effectHandlers.remove
  },
```

```
  1: { // Normal
    remove: effectHandlers.remove,
    changeColor: effectHandlers.changeColor
  },
  2: { // Hard
    breakCalculator: effectHandlers.breakCalculator,
    breakGugudan: effectHandlers.breakGugudan,
    breakNumGame: effectHandlers.breakNumGame,
    breakWordBook: effectHandlers.breakWordBook,
    breakClickHere: effectHandlers.breakClickHere,
    breakImageToggle: effectHandlers.breakImageToggle,
    breakColorList: effectHandlers.breakColorList,
    breakFlashBox: effectHandlers.breakFlashBox,
    breakMovingBox: effectHandlers.breakMovingBox,
    breakHangman: effectHandlers.breakHangman,
    remove: effectHandlers.remove  // 하드에서도 remove 가능
  }
};
```

이벤트 핸들러 생성.

```
function createHardElements() {
  const totalBrickCount = (brickRowCount + extraRow) * brickColumnCount;
  const hardTargets = [
    desEleHard.find(el => el.selector === ".lab.calculator"),
    desEleHard.find(el => el.selector === ".lab.gugudan"),
    desEleHard.find(el => el.selector === ".lab.numGame"),
    desEleHard.find(el => el.selector === ".lab.wordBook"),
    desEleHard.find(el => el.selector === ".lab.clickHere"),
    desEleHard.find(el => el.selector === ".lab.image-toggle"),
    desEleHard.find(el => el.selector === ".lab.colorList"),
    desEleHard.find(el => el.selector === ".lab.flashBox")
  ].filter(Boolean); // null 제거
```

3.2. 애니메이션 충돌 및 처리

requestAnimationFrame(draw) 메서드를 활용하여 공, 패들, 벽돌, 점수, 효과 등을 지속적으로 렌더링하며, 게임 루프를 구현하였습니다.

collisionDetection()과 bounceBall() 함수는 공과 벽돌, 패들 간의 충돌을 감지하고, 이에 따라 공의 방향을 반사시키거나 벽돌을 제거하는 기능을 수행합니다.

벽돌 객체는 isBomb, isSecure 등의 속성을 포함하고 있어, 일반 블록과 다른 특수 효과를 적용할 수 있도록 설계되어 있습니다. 예를 들어, isBomb 속성이 true일 경우, 파괴 시 주변 블록도 동시에 제거하는 기능을 수행합니다.

3.3. 여러 개의 화면을 보여주는 방법

```
function allHide(){
  //전부 다 hide하는 함수
    $(".menu-page").hide();//메뉴 페이지 hide
    $("#game-wrapper").hide();//game hide
    $("#clear-panel").hide(); // 스토리 화면 검열
  }
```

해당 hide 함수를 이용하면, 모든 페이지를 제이쿼리로 select해서 지우는 것이 가능합니다. 이후 필요한 페이지만 show 하는 방식으로 구성하였습니다.

3.4. 블록을 관리했던 법

```
if(difficulty == 0 || difficulty == 1){
    const bricks = [];
    let tag = layout[r][c];
    const isBomb = tag === "bomb";
    const isTopRow = r < extraRow;
    const isSecure = (difficulty !== 0 && Math.random() < 0.2);
```

```
    const hp = isSecure ? 3 : null;
    return {
     x: c * (brickWidth + brickPadding) + brickOffsetLeft,
     y: (r - extraRow) * (brickHeight + brickPadding) + brickOffsetTop,
     status: isTopRow ? 0 : 1,
     isBomb: isBomb,
     isHidden: isTopRow ? 1 : 0,
     targetSelector: element?.selector,
     effect: element?.effect,
     color: element?.color,
     isSecure: isSecure,
     secureState: isSecure,
     hp: hp,
     tag:tag
    };
  }
```

블럭을 생성할 때, 블럭 내부에서 접근 가능한 속성값을 만드는 방식으로 해결하였습니다.

3.5. 실제 블록끼리의 상호작용

예시 : 폭탄 블럭

```
function triggerBombChain(c, r) {
  console.log(`[DEBUG] triggerBombChain at (${c}, ${r})`);
  const directions = [
    [0, -1], [0, 1], [-1, 0], [1, 0],
  ];
  for (const [dc, dr] of directions) {
    const nc = c + dc;
    const nr = r + dr;
    if (
      nc >= 0 && nc < brickColumnCount &&
      nr >= 0 && nr < bricks[nc]?.length
      ) {
```

```
      console.log(`[DEBUG] 인접 벽돌 제거 시도: (${nc}, ${nr})`);
      destroyBrick(nc, nr);
    }
  }
}
```

해당 코드를 이용하여서, 인접 블럭들에 대해서 재귀적으로 블록을 제거하는 호출을 사용하는 식으로 특수 기믹을 구현하였습니다.

3.6. 난이도 구성

난이도별 벽돌 배열은 다음과 같이 구성되어 있습니다:

쉬움(Easy)

기본 블록: 4x1 배열 2줄 (총 8개)

특수 블록: 폭탄 블록

블록 구성 : article 태그 4개, footer 태그 2개, header 태그 1개

비고

article 태그를 제거하면 article이 순서대로 제거되고, footer은 footer를, header는 마지막에 페이지 전체를 제거합니다.

중간(Normal)

기본 블록: 4x3 배열 1줄, 4x1 배열 3줄 (총 24개)

특수 블록: 폭탄 + 강화 블록

블록 구성 : body 태그 3개, main-menu 태그 2개, lab 태그 2개, header 태그 2개, container 태그 4개, footer 태그 2개.

비고

이 단계에서부터는 html 태그가 사라지지 않고, css 태그가 NormalModeGameFun의 if 문을 따라가며 정해진 count 이상으로 블록이 사라졌을 경우 삭제됩니다.

어려움(Hard)

기본 블록: 4x4 배열 1줄, 4x1 배열 4줄 (총 32개)
특수 블록: 폭탄 + 강화 블록 + 시간제한

비고

이 단계에서는 제이쿼리를 이용해서, 동적인 실습이 특정 상태에서 멈춰버려서 망가지게 된 것처럼 보이도록 유도하였습니다.

3.4 기타 기능(초기화 등)

게임 시작 버튼 클릭 시, 메뉴 및 설정 요소들이 숨겨지고 game-wrapper가 활성화되어 본격적인 게임 화면으로 전환됩니다.

설정 메뉴를 통해 인게임 사운드 및 공의 스킨 등을 사용자가 선택할 수 있으며, 이는 실제 게임 화면에 즉시 반영됩니다.

게임은 자동 난이도 조정 시스템을 포함하고 있으며, 쉬움 모드를 클리어하면 자동으로 중간 모드로, 중간 모드를 클리어하면 어려움 모드로 넘어갑니다. 물론, 사용자가 처음부터 원하는 난이도를 직접 선택하는 것도 가능합니다.

4. 토의사항

문제점 & 새롭게 발견한 사실 :

Iframe 활용 관련

본 프로젝트는 원래 '실습과제' 라는 교육적 목적에 초점을 두고 기획됐습니다. 초기 방향은 특정 벽돌을 제거할 때마다 해당 벽돌에 연결된 실습 과제의 코드가 동적으로 변경되고, 그에 따라 실시간으로 HTML 화면의 시각적 변화가 나타나는 컨셉이었습니다.

그러나 이 개념을 단일 HTML 파일 내에서 구현하려고 시도했을 때 여러 기술적 한계에 부딪히게 되었고, 특히 외부 코드와의 상호작용을 구현하는 방식에서 많은 어려움을 겪었습니다. 인터넷을 통해 다양한 방법을 찾아보았으나 마땅한 해결책을 찾지 못해, 이 프로젝트의 핵심 컨셉 자체를 포기해야 하는 것이 아닌가 하는 위기감도 들었습니다.

이 문제를 해결하기 위해 지도 교수님과 상담한 결과, HTML을 하나의 객체처럼 다루는 방식을 제안받았습니다. 즉, iframe 내부에 실습용 HTML 문서를 삽입하고, 이를 외부에서 제어하는 방식으로 실습과 게임을 연결하는 구조를 제시받은 것입니다.

해당 조언을 바탕으로 구현 방식을 전환하여, iframe을 활용한 실습 환경 조작 시스템을 도입하게 되었고, 결과적으로 게임의 컨셉을 유지하면서도 기술적으로 완성도 있는 구조를 구현할 수 있게 되었습니다.

이 과정은 단순한 문제 해결을 넘어, 프로젝트 개발 초기의 아이디어를 현실적인 기술로 구체화하는 전환점이 되었으며, 프로젝트의 교육적, 기능적 깊이를 동시에 강화한 중요한 경험이 되었습니다.

5. 기여도

강동훈 : 25% / iFrame 구현, hard모드 블록깨기에 따른 실습화면 변화 구현

박준건 : 25% / 디자인,css 구현, easy/normal 모드 블록깨기에 따른 실습화면 변화 구현

이시헌 : 25% / 난이도에 따른 폭탄, 강화블럭, 시간제한 기믹 구현

이수현 : 25% / 배경음악, 게임 오버 등등 전체적인 게임 설정 구현