2025.10.28 ~ 10.31

웹 게임 만들기

햄스터의 씨앗 찾기

임욱

구디 아카데미 96기

목표 설정 및 설계

기술 선정 이유와 구현 방법

목표 설정

javascript 다양한 함수들과 기능 활용해 웹 게임을 만들어보자

2025.10.28 ~ 10.31

1.마우스 이벤트	가장 접근하기 쉽고 다양한 종류의 게임을 만들 수 있을 것이라 생각했습니다.
2. 키보드 이벤트	실습 경험은 적지만 마우스 이벤트보다 제약조건이 적을 것이라 기대했습니다.
3. 폼 이벤트	게임의 주요기능으로 사용하기에 제한적이라 예상했습니다.

핵심기능

01.

방향키 조작

addEventListener( )
활용으로 입력된 키를 인식하고 위치를 반영합니다.

02.

무작위 위치, 움직임

random( )
무작위 위치에 목표를 생성하고 움직입니다.

03.

시간 설정

setInterval()
clearInterval( )
제한 시간, 움직임을 설정합니다

게임설명

클리어 조건

점수판 아래의 개수만큼 씨앗 획득

난이도 설정

1번 : 무작위 위치에 목표 생성
2번 : 무작위 움직의 목표 생성

Timeline

페이지 로드 및 대기

전역 변수 및 상수 설정
타이틀 화면을 출력합니다.

titleScreen() startGameBtn
onclick 대기

게임시작 및 초기화

titleScreen을 숨기고,
게임을 화면을 준비합니다
.

startGame()
Player , Goal
startTimer(), setInterval()
addEventListener('keydown', ...)

게임 실행

이벤트를 처리하고 충돌 검사,
점수 반영, 위치 리셋, 점수판
반영 등을 수행합니다.

Player.handleKeydown()
Player.checkCollisions()
score++, Goal.reset()
board()

게임종료 및 재시작

점수(score) 확인 후 메세지를
호출합니다.

checkScore(scroe)
gameOver(message)
location.reload()

기능 구현

객체와 함수를 생성해 기능을 구현합니다.

Player

1

handleKeydown

```
switch (key) {
  case 'ArrowLeft': newX -= STEP; moved = true; break;
  case 'ArrowRight': newX += STEP; moved = true; break;
  case 'ArrowUp': newY -= STEP; moved = true; break;
  case 'ArrowDown': newY += STEP; moved = true; break;
  default: return;
}

if(newX < 0){ newX = 0; }
else if(newX>canvasWidth-ELEMENT_SIZE)
  { newX = canvasWidth - ELEMENT_SIZE; }
if(newY < 0){ newY = 0; }
else if(newY>canvasHeight-ELEMENT_SIZE)
  { newY = canvasHeight - ELEMENT_SIZE; }

this.x = newX;
this.y = newY;
this.element.style.left = `${this.x}px`;
this.element.style.top = `${this.y}px`;

if(moved) { this.checkCollisions(); }
}
this.move=()=>{document.addEventListener('keydown',
        this.handleKeydown); }
```

2

checkCollisions

```
this.checkCollisions = function(){
    const size= ELEMENT_SIZE - 30;
  if(this.x<this.goal.x+size&&this.x+size>this.goal.x
  &&this.y<this.goal.y+size&&this.y+siz>this.goal.y){

      score++;
      board();
      checkScore(score);
      this.goal.reset();

      return true;
    }
    return false;
  }
```

Goal

1 startMove

```
if(this.moveInterval)  clearInterval(this.moveInterval);

 this.vX = randomSpeed(ELEMENT_SPEED);
 this.vY = randomSpeed(ELEMENT_SPEED);

 this.moveInterval = setInterval(() => {
if(this.x <= 0 || this.x >= canvasWidth - ELEMENT_SIZE)
  { this.vX *= -1; }
if(this.y <= 0 || this.y >= canvasHeight - ELEMENT_SIZE)
  { this.vY *= -1; }

   this.x += this.vX;
   this.y += this.vY;

   this.element.style.left = this.x + 'px';
   this.element.style.top = this.y + 'px';
}, 1000/60);
```

2 reset

```
this.reset = function(){
      this.x = position(canvasWidth - ELEMENT_SIZE);
      this.y = position(canvasHeight - ELEMENT_SIZE);
      this.element.style.left = this.x + 'px'
      this.element.style.top = this.y + 'px'

      if (this.level == 2) { this.startMove(); }
      else if (this.moveInterval){
clearInterval(this.moveInterval); }
    }
```

function

1 startTimer

```
currentTime = TIME_LIMIT
 if(timerInterval){ clearInterval(timerInterval); }

 timerInterval = setInterval( ( ) => {
  if (isGameOver) {
   clearInterval(timerInterval);
   return;
  }

  currentTime--;
  board();

  if(currentTime <= 0){
   clearInterval(timerInterval);
   gameOver("Game over");
  }
 }, 1000);
```

2 board

```
const divScore = document.getElementById('score');
  divScore.innerHTML = '';
    for (let i = 1; i <= total; i++) {
     let imageUrl = '';
     if (i <= score) { imageUrl = 'img/seed.png'; }
     else { imageUrl = 'img/seed2.png'; }

     divScore.innerHTML += `<div class="seed"><img
src='${imageUrl}'></div>`;
    }
    const divTimer = document.getElementById('timer');
    divTimer.innerHTML = `남은 시간: ${currentTime}초`;
```

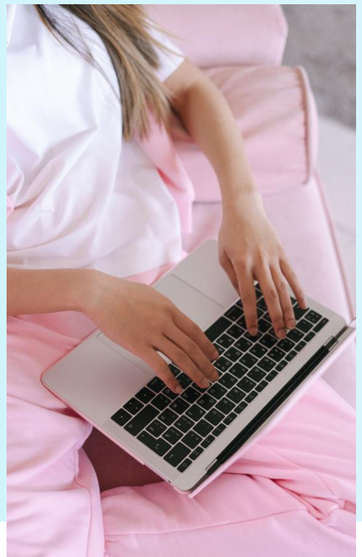
마치며

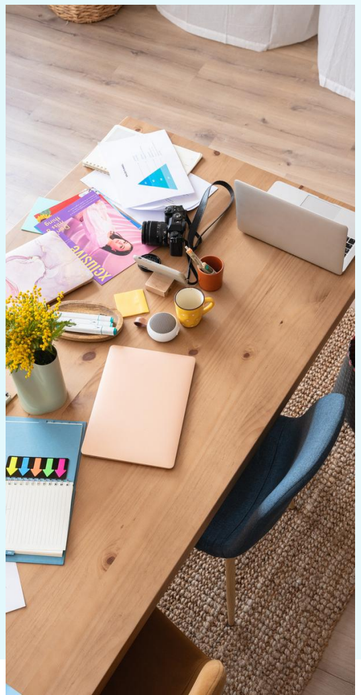
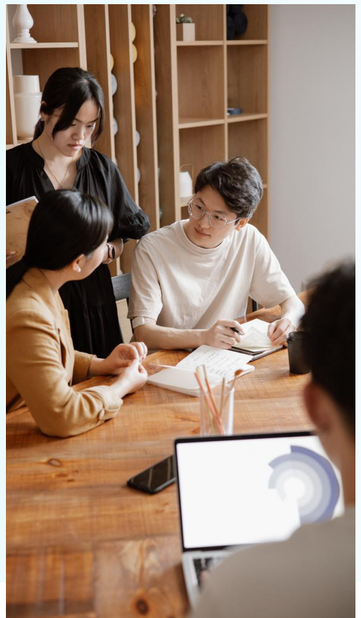
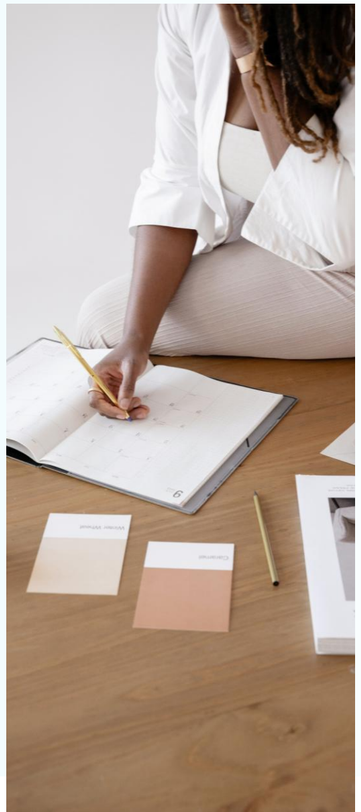
느낀점 및 개선점

느낀점 및 개선점

객체를 이용하며 인스턴스의 속성과 메서드 동작에 대해 깊게 생각할 기회가 되었습니다. 그리고 앞으로의 프로젝트에 대해 생각할 기회였다고 생각합니다.

장애물 난이도 구현을 하지 못한 점과 혼자 고심하기보다 의견을 더 나누면 좋았을 것이라는 아쉬움이 남습니다.

감사합니다