REPORT

컴퓨터 그래픽스 중간과제

과목명 : 컴퓨터 그래픽스

교수명 : 송인식 교수님

학 과 : 소프트웨어

학 번 : 32170699

이 름 : 김수연

제출일 : 2020년 6월 19일

DKU 단국대학교 DANKOOK UNIVERSITY

<목차>

-배경 설명 및 문제 정의

저는 webgl 을 통해 이왕이면 제작자와 보는 사람 모두가 즐길 수 있는 게임 프로젝트를 만들고자 하였고, 이에 따라 이번 프로젝트를 계획하게 되었습니다.

- 기존의 처리 방법 (사례 조사 등) 및 해결하고자 하는 방법 (선정 이유 및 차별성)

구글 및 깃허브에서 여러가지 게임 프로젝트를 찾아보았고, 제가 수업에서 배운 것들로 충분히 이해하고 프로그램을 수정할 수 있는 게임 프로젝트를 선정하였습니다. 화려하고 자세한 그래픽을 구현하는 게임 프로젝트도 많았지만, 시간과 경험 부족 문제로 인해 상대적으로 난이도가 낮더라도 제가 배운 것을 토대로 확실하게 구현, 수정이 가능한 프로젝트를 제작하고자 합니다.
제가 선택한 프로젝트는 기본적인 슈팅게임 입니다. 적들이 날아오면 한편, 기본적인 게임이라서 그런지 수정할 사항이 꽤나 많이 눈에 띄었습니다. 제가 ***수정한 사항은*** 다음과 같습니다.

1. 적들이 날아오는 속도 감소 시키기(최고속도, 최저 속도 및 속도 렌더링 함수 추가)

2. 적들이 날아오는 시간 고르게 분배하여 구현(현재 게임 초반에 너무 많은 적들이 한꺼번에 날아옴)- 적들의 위치 범위 증가시키기 및 기존의 카메라 렌더링을 활용하여 보완

3 적들의 수명(hp)이 줄어드는 간격 수정(현재 -20hp 씩은 적들이 너무 쉽게 제거됨)

4. 적들의 기본 수명 증가

5. 게임 주인공 선박(ship)의 크기 조정, 눈에 더 잘 띄기 위한 색깔 조정

6. 배경 디자인 수정

7. 레이저 포인터 색상 수정

8. 적들이 게임 선박에 딱 붙어 뭉치지 않도록 일정한 거리 유지하도록 구현

9. 날아오는 적들의 모양 더욱 다양하도록 각의 수를 증가시킴

10. 사용자가 움직이는 마우스 방향대로 선박도 회전(paddle rotation 으로 구현)

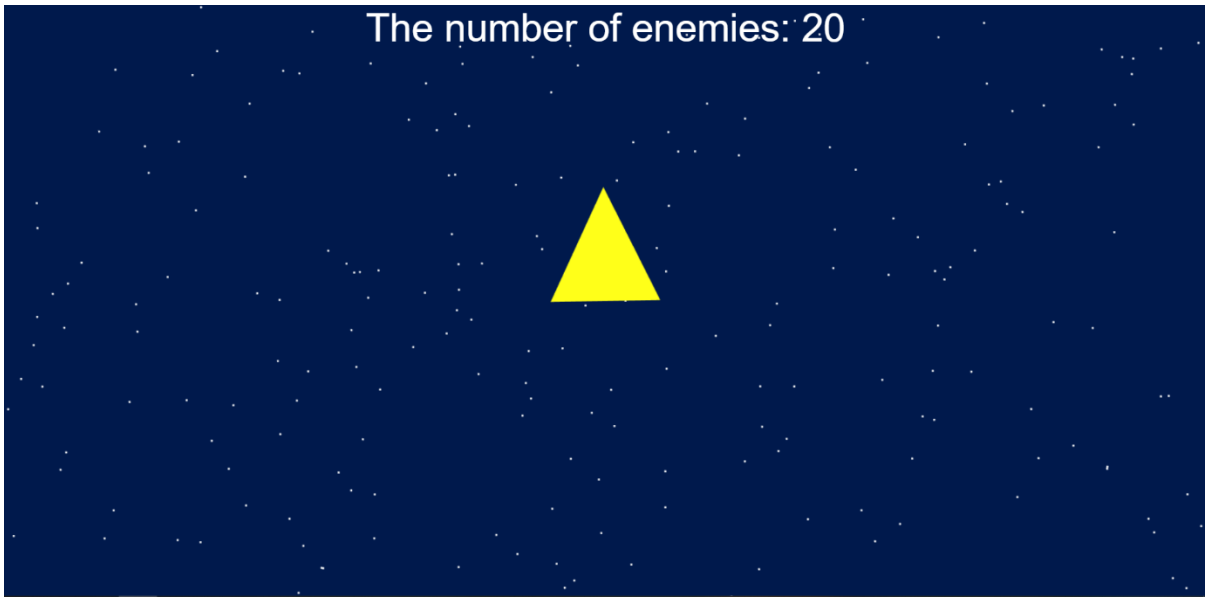
- 제공 기능(요구사항 명세서)

이 프로젝트의 제공 기능은 다음과 같습니다.

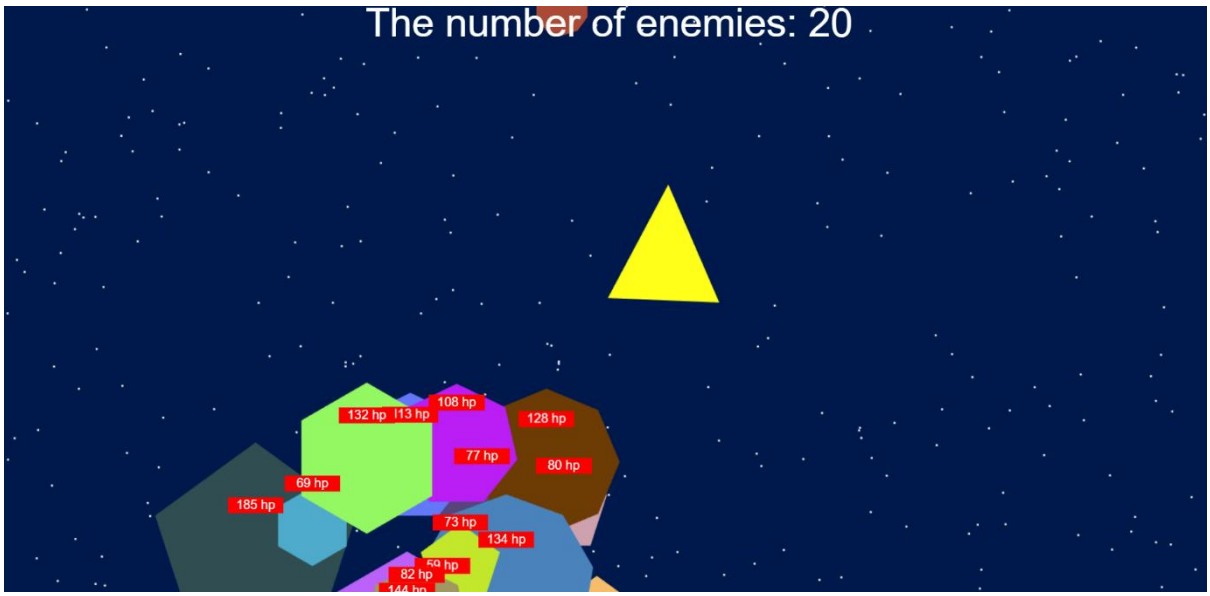
요구사항 명	요구사항	비고
배경 색	게임의 배경 색상, 디자인 설정	Webgl canvas 으로 설정
플레이어(선박) 움직이기	플레이어(선박)이 위, 아래, 좌, 우로 움직일 수 있도록 한다.	키보드에서 A: 좌, D: 우, W: 위, S: 아래 키로 설정, 선박의 움직임을 entity 클래스에 x,y 축의 변경 값을 저장함으로써 구현
적들이 날아옴	무작위 도형 모양의 적들이 날아오도록 한다.	별 필드 레이어가 각각 다른 수의 별을 가지며 플레이어 기준 카메라 위치에 따라 다른 속도로 이동. 위치는 스타 필드 외부에 무작위로 설정.
플레이어가 레이저 쏘아 적들 물리치기	플레이어가 적들을 클릭하면 적들의 수명이 10 씩 줄어든다.	마우스 왼쪽 버튼을 클릭했을 때 발생하는 마우스 클릭을 수신.
남은 적들의 수 표시	게임 화면 한쪽에 남아있는 적들의 수를 표시한다. (즉 적의 수가 줄어들 때 마다 이를 반영하여 화면에 표시)	적들을 하나씩 제거할 때 마다 1 씩 증가하는 함수와 게임 화면 출력 클래스 연결
게임 종료 시 다시 플레이 할 의사 물어보기	게임이 끝나고 또 플레이 할 것인지 의사를 물어보는 텍스트와 게임을 한번 더 플레이 할 수 있는 재 시작 버튼을 제공한다.	적의 수 값이 0 이 되면 게임 종료 텍스트와 버튼 출력

게임 플레이 시간 계산	적들을 다 물리칠 때까지 걸린 시간을 계산한다.	적들의 값이 0 이 될 때 계산한 시간 값을 출력

-개략적인 간단한 메뉴 구성 등 사용자 인터페이스

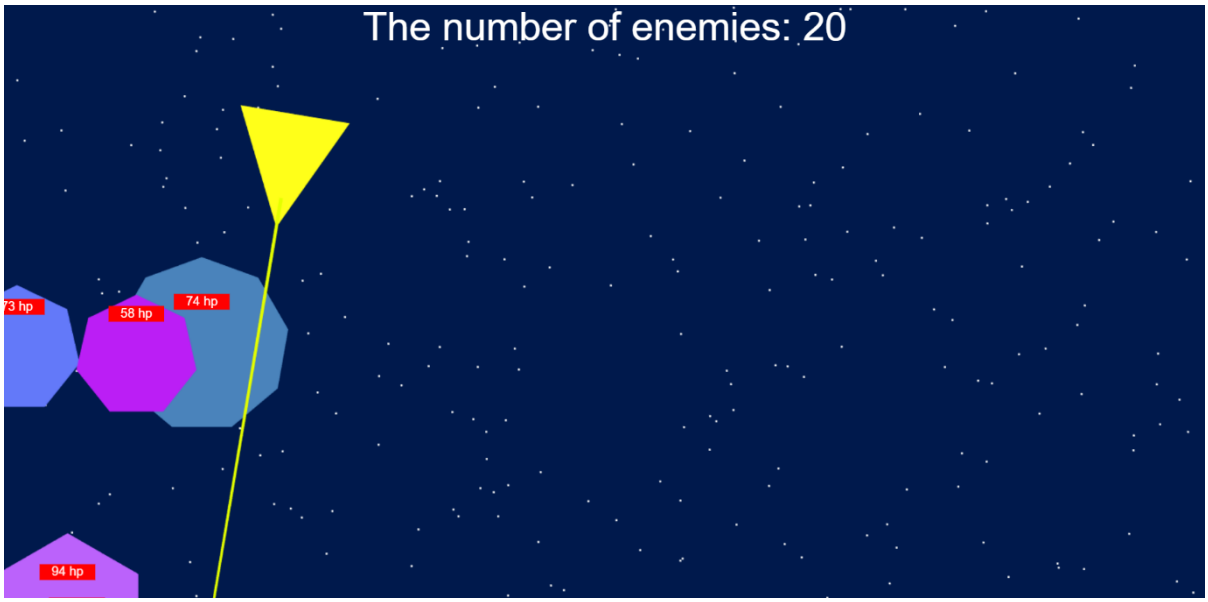


먼저 게임 시작 시의 첫 화면 입니다. 위의 The number of enemies 라는 텍스트는 남은 적들의 수를 나타내 주는 것으로 현재는 적들이 도착하지 않아 적들의 초기 값인 20 을 표시하고 있습니다. 이후 게임이 진행되고 적들을 하나 둘씩 처치함에 따라 저 값이 줄어드는 것을 확인 할 수 있습니다.

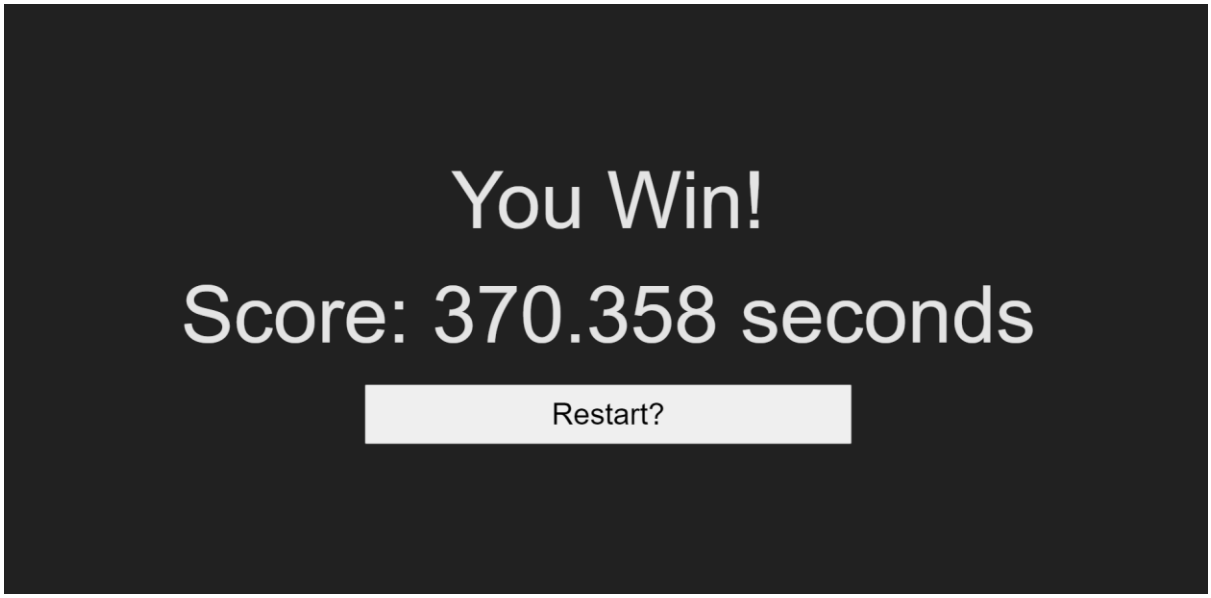


다음으로 시간이 몇 초가 지나면 위와 같이 플레이어 카메라를 기준으로 여러 도형의 적들이 다가온 것을 확인할 수 있습니다. 또한 플레이어 선박과 일정한

거리를 유지하도록 구현한 것이 확인 가능합니다. (distance = 0.5 이하로 가까워지지 않도록 구현함) (참고: 해당 화면은 'a' 키로 플레이어가 살짝 왼쪽으로 이동한 상태)



다음으로 위의 화면을 보면 앞서 요구사항에서 설명 드렸듯이 레이저를 쏘며 적을 물리치는 것을 확인하실 수 있습니다. (플레이어는 방향키를 통해 오른쪽 아래로 이동한 상태입니다.)캡쳐를 할 수 없지만 도형에 마우스 왼쪽 클릭을 하여 슈팅이 되면 해당 도형의 hp 가 10 씩 감소됩니다.



마지막으로 enemies=0 이 되고 게임이 종료되면 사용자의 게임 플레이 시간을 출력해 주며 게임 재 시작을 할 수 있는 버튼을 제공합니다. (restart 버튼 누르면 게임 재 시작)

-사용자 시나리오 제시

1. 사용자는 index.html 을 열어 게임에 접속합니다.
2. 도형들이 날아오면 마우스 왼쪽을 클릭하여 적들의 hp 를 0 되게 만들어 제거한다.
3. 도형들의 숫자, 즉 화면 위쪽의 enemies 가 0 이 되면 게임이 종료될 때까지 플레이한다.
4. 게임 종료 후 자신의 플레이 시간(기록)을 확인한다.

(4_1. 게임을 다시 플레이하고 싶다면 restart 버튼을 클릭한다.)

- 예상 문제점 및 대응 방안

예상 문제점과 대응 방안은 다음과 같습니다.

먼저 이전에 보여드린 두번째 캡쳐 화면과 수정할 내용에서 언급했던 것처럼

1. 게임을 플레이하며 게임 초반에 적들이 한꺼번에 몰려온다.

-> 적들이 움직이는 속도를 조정하여 이러한 문제를 해결할 예정

2. 여러 적들이 옴으로써 플레이어 삼각형의 파란 삼각형 모양이 눈에 잘 띄지 않는 문제

-> 플레이어의 모양을 수정, 크기를 더욱 키워 해결

3. 적들의 속도가 너무 빠르며 날아오는 범위가 작다는 문제

-> 적들의 최고, 최저 속도 수정 및 속도 조절 함수를 추가하였다.

4. 사용자가 움직이는 방향대로 마우스 회전이 안됨

-> rotation 모듈 추가하여 마우스 방향 회전도 가능하게 수정함.

- 개발 일정

날짜	개발 일정
5/22	제안서 작성
5/31	적들이 날아오는 속도 감소 시키기, 적들의 수명(hp)이 줄어드는 간격 수정
6/5	적들이 날아오는 시간 고르게 분배하여 구현
6/15	마우스 회전 기능 추가
6/18	플레이어의 모양(단편적인 삼각형)수정_shading, 그림자, 반사 이용

~ 제출 전까지	세세한 디자인 수정 및 오류 점검

<코드 설명>- 특징적인 코드 위주로

```
move() { //마우스 회전 기능 추가
    counter++
    // Rotation
    const padding = 0.05
    const boundLeft = -1.0 + padding
    const boundRight = 1.0 - padding
    const boundTop = 1.0 - padding
    const boundBottom = -1.0 + padding
    // Move horizontally
    if (keyTracker["KeyA"] == true) {
        // Move left
        const newX = this.x - this.xv
        if (newX < boundLeft) {
            this.x = boundLeft
        } else {
            this.x = newX
        }
    } else if (keyTracker["KeyD"] == true) {
        // Move right
        const newX = this.x + this.xv
        if (newX > boundRight) {
            this.x = boundRight
        } else {
            this.x = newX
        }
    }
    // Move vertically
    if (keyTracker["KeyS"] == true) {
        // Move down
        const newY = this.y - this.yv
        if (newY < boundBottom) {
            this.y = boundBottom
        } else {
            this.y = newY
        }
    } else if (keyTracker["KeyW"] == true) {
        // Move up
        const newY = this.y + this.yv
        if (newY > boundTop) {
            this.y = boundTop
```

```
/* 마우스 움직일때 마다 회전 하도록 구현  */
document.addEventListener("mousemove", e => {
    mouseX = map(e.clientX, 0, canvas.clientWidth, -1, 1)
    mouseY = map(e.clientY, 0, canvas.clientHeight, 1, -1)
})

document.addEventListener("mousedown", e => {
    mousePressed = true
    if (mousePressedInterval == null) {
        mousePressedInterval = window.setInterval(whilePressed, 250)
        whilePressed()
    }
})
document.addEventListener("mouseup", e => {
    mousePressed = false
    window.clearInterval(mousePressedInterval)
    mousePressedInterval = null
})

updateViewport()
```

-마우스 회전 기능 추가(위의 두 캡쳐본 참고)

```
/* updates the keyTracker, used for keyboard input */
document.addEventListener("keydown", e => keyTracker[e.code] = true);
document.addEventListener("keyup", e => keyTracker[e.code] = false);

/* Generate enemy entities in random locations */
for (let i = 0; i < numEnemies; i++) {
    // Put enemies outside edges of playable area
    let randX = (Math.random() - 0.6) * 2
    let randY = (Math.random() - 0.6) * 2 // 적들이 날아오는 random location을 더 넓게 설정
    if (randX < 0) {
        randX -= 1.0
    } else {
        randX += 1.0
    }
    if (randY < 0) {
        randY -= 1.0
    } else {
        randY += 1.0
    }
```

(주석 참고)

```
class Enemy extends Entity {
    constructor(initialX, initialY, initialXv, initialYv, shapeData, sizeFactor) {
        super(initialX, initialY, initialXv, initialYv, shapeData)
        this.hpBox = null
        this.sizeFactor = sizeFactor
        this.health = 1500 * this.sizeFactor //적의 수명 결정 - 조금 더 긴 수명으로 설정
        this.colors = initColors(this.points.length, Math.random(), Math.random(), Math.random(), 1)
        this.entityBuffer = initBuffer(this.points)
        this.colorBuffer = initBuffer(this.colors)
    }

    move() {
        const coords = getPlayerCoords()
        const playerX = coords[0]
        const playerY = coords[1]
        const distanceX = this.x - playerX
        const absDistanceX = Math.abs(distanceX)
        const distanceY = this.y - playerY
        const absDistanceY = Math.abs(distanceY)
        // Move enemy left if its position is to the right of the player else move right
        if (distanceX > 0) {
            // If velocity is greater than remaining distance to move to the player, only move the remaining distance
            if (this.xv > absDistanceX) {
                this.x -= absDistanceX + 0.5 //적이 ship에 너무 딱 달라붙지 않도록 수정(원래는 0이었음)
            } else {
                this.x -= this.xv
            }
```

(주석 참고)

```
function getShipShape() {
    const h = 0.135
    const d1 = 0.065 // ship의 모양을 조금 더 키운다.(눈에 더 잘 띄도록)

    let p1 = vec3(d1, 0.0, 1.0)
    let p2 = vec3(0, h, 1.0)
    let p3 = vec3(-d1, 0.0, 1.0)
    return [p1, p2, p3]
}

function getEnemyShape(radius) {
    // Generate random shapes with at minimum 4 vertices
    const numPoints = Math.floor(Math.random() * 7) + 4 // 날아오는 적의 모양이 더 다양하도록 더 큰 수를 곱해줌
    // console.log(`Generating enemy shape with ${numPoints} points.`)
    const angle = 2 * Math.PI / numPoints
    let shapePoints = []
    for (let i = 0; i < numPoints; i++) {
        x = radius * Math.sin(i * angle)
        y = radius * Math.cos(i * angle)
        shapePoints.push(vec3(x, y, 1))
    }
    return shapePoints
```

(주석 참고)

<소감>

상대적으로 어렵지 않은 주제를 골랐다고 생각했는데 프로젝트를 많이 수정하다 보니 생각보다 난이도가 있었고, 시간도 꽤 소요되었다. 또한 원래 계획에 게임 도중 재시작을 할 수 있도록 적들의 수를 표시해주는 것 밑에 재시작 버튼을 넣으려고 했는데 html 의 body 부분에서 onload init 때문에 버튼을 따로 추가하기가 매우 까다로워 그 부분은 구현하지 못하였다. 한편 코딩을 하면 바로바로 시각적인 결과물이 나타나는 것에 흥미를 느끼는 나에게는 이번

프로젝트가 흥미로웠고, 게임 개발이나 웹, 멀티미디어, 그래픽스 쪽으로 진로를 정해도 좋을 것 같다는 생각을 하였다.

- 참고 문헌

http://blog.rightbrain.co.kr/?p=2465
https://medium.com/@mklab.co/%EC%9E%91%EC%84%B1%EB%B2%95-%EC%9A%94%EA%B5%AC%EC%82%AC%ED%95%AD-%EB%AA%85%EC%84%B8%EC%84%9C-requirements-specification-ad3533d6d5b8
https://tug.org/pracjourn/2007-4/walden/color.pdf(여러 entity 들의 색상 변환에 참고)

- 웹사이트 (GitHub)

https://github.com/suyeonkimm/computer_graphics_project