REPORT

과 목 명 : 컴퓨터그래픽스
담담교수 : 송인식 교수님
소　　속 : 소프트웨어학과
학　　번 : 32151671
이　　름 : 박민혁

단국대학교
Dankook University

목차

I. 프로젝트 개요

(1) 배경 설명 (변경사항)

- 처음에 제시했던 제안서는 책으로 정리하는 프로젝트였습니다. 하지만 오픈소스를 참고하여 만들려고 했지만, 내용이 너무 어려웠고 또한 추가 되는 내용이 적어서 다른 프로젝트를 진행했습니다. 이 프로젝트는 수업시간에 배운 것을 토대로 만들 수 있는 것으로 선정하였습니다. 비교적 간단한 프로젝트지만 수업시간에 배운 내용으로만 구현할 수 있는 게임입니다.

(2) 문제 정의

- 일반적인 테트리스 게임입니다. 간단한 프로젝트를 진행 하되 제가 만들 수 있는 것으로 선정하게 되었습니다.

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

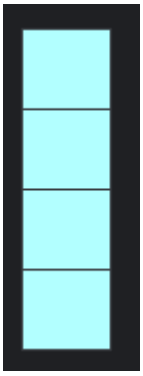
- 기존에 테트리스 게임은 많이 나와 있습니다. 정말 잘 구현 되어 있습니다. 하지만 앞서 제시한 프로젝트처럼 오픈소스를 활용하여 만드는 것 보다 처음부터 끝까지 제가 만들고 싶었고 어렸을 때 즐겨하던 테트리스가 생각나 기획하게 되었습니다. 또한 기존의 initShaders.js MV.js webgl-utils.js 파일을 이용해 만들 예정입니다.

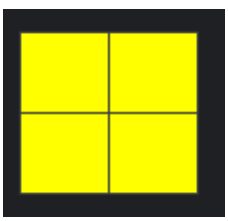
II. 프로젝트 시행

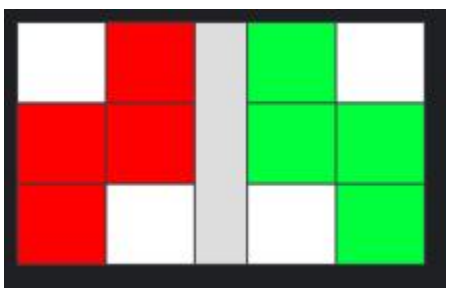
(1) 제공 기능(요구사항 명세서)

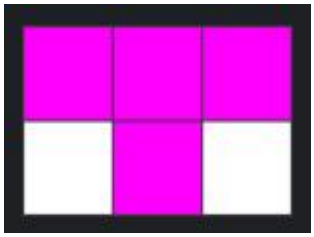
- Control : 화살표키를 이용하여 컨트롤 하게 됩니다.
- ↑(방향키) : 테트리스 피스를 시계 반대 방향으로 90도 회전시킵니다.
- →←(방향키) : 테트리스 피스를 왼쪽 및 오른쪽으로 이동시킵니다.
- ↓(방향키) : 테트리스 조각 스피드를 가속합니다.
- q : 게임을 종료합니다. (r을 눌러 게임 재 시작 가능)
- r : 게임 다시 시작합니다.
- j : 게임의 난이도가 높아집니다.
- k : 게임의 난이도가 낮아집니다.

- 기본 피스 모음









(2) 예상 문제점 및 대응 방안

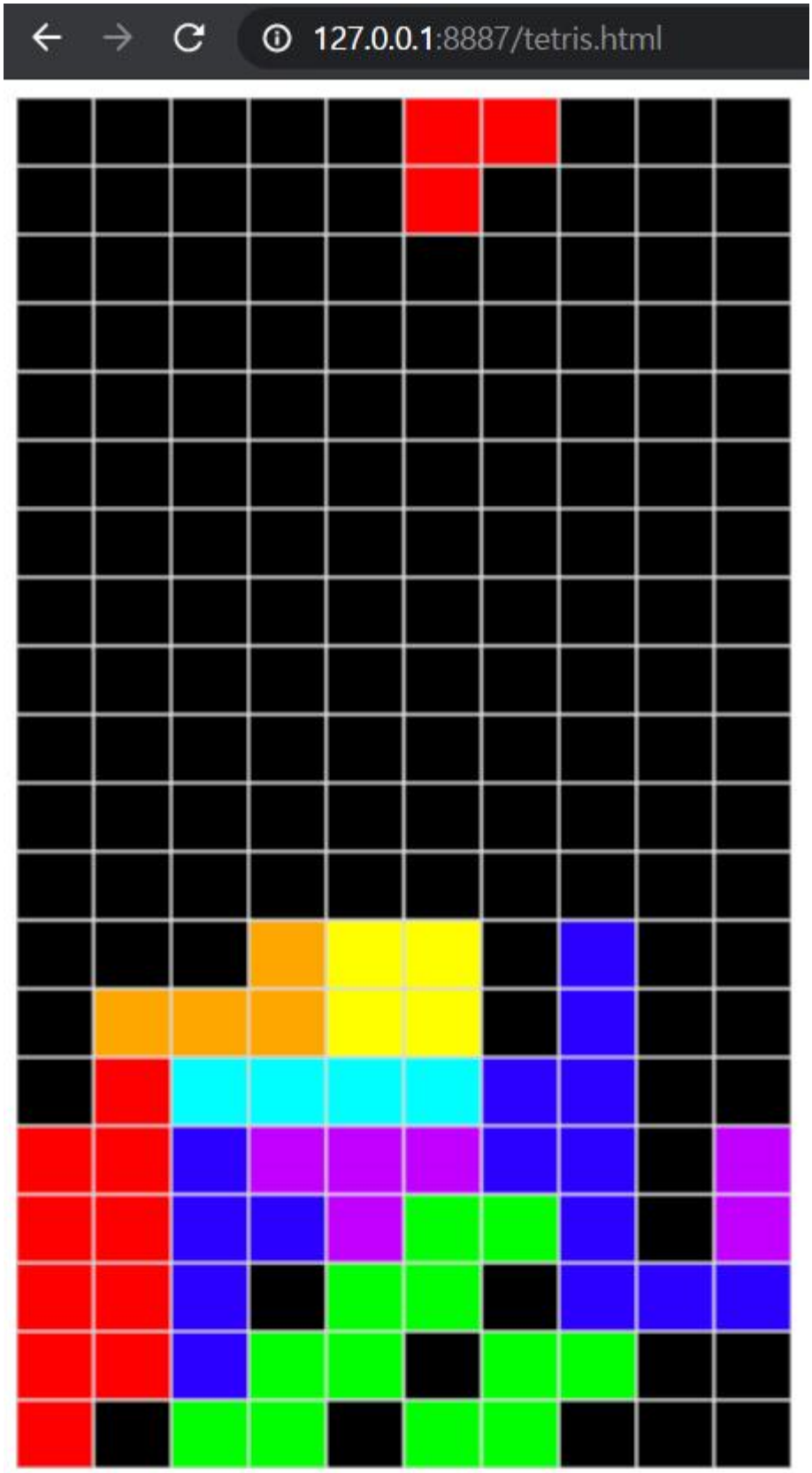
I. 문제점

- 본래 테트리스 게임은 시간이 지날수록 스피드가 빨라집니다. 또한 스페이스 바를 눌렀을 때 자동으로 바로 테트리스 피스가 내려가게 됩니다. 이러한 부분은 조금 더 연구해 봐야 될 것 같습니다.

II. 대응 방안

- 게임 난이도를 설정할 수 있도록 구현하고, 난이도를 올릴수록 스피드가 빨라지게 할 예정입니다.

(3) 결과 화면



III. 코드 설명 및 느낀점

(1) 코드설명

tetris.html : 구현한 화면을 보여주는 페이지입니다.

tetris.js

```
// Draw grid rows
var rowGridSpacing = (2 - padding * 2) / numberOfRows;
var colGridSpacing = (2 - padding * 2) / numberOfCols;

var lowerTetrisPieceFlag = true;
var gameTerminatedFlag = false;

// Draw grid
for (var i = 0; i <= numberOfRows; i++) {
    gridVertices.push(vec2(-1 + padding, -1 + padding + i*rowGridSpacing));
    gridVertices.push(vec2(1 - padding, -1 + padding + i*rowGridSpacing));
}

for (var i = 0; i <= numberOfCols; i++) {
    gridVertices.push(vec2(-1+padding + i*colGridSpacing, 1-padding));
    gridVertices.push(vec2(-1+padding + i*colGridSpacing, -1+padding));
}
```

- 행과 열을 그려 테트리스 판을 그리는 코드입니다.

```
// Takes speed of game in parameter
function setGameSpeed(speed) {
        clearInterval(gameInterval);
        console.log("Setting interval to " + speed);
        gameInterval = window.setInterval(lowerTetrisPiece, speed);
}
```

- 테트리스 스피드를 조절하는 코드입니다. 이 부분 아래 코드를 보면 키설정과 속도 조절하는 코드가 나와 있습니다.

```
var tetriminoPieces = [
    {
        type: "oPiece",
        styles: [
            {
                type: 1,
                orientation: [
                    vec2(-1, 0),
                    vec2(0, 0),
                    vec2(-1, -1),
                    vec2(0, -1)
                ]
            }
        ],
        color: vec4(1.0, 1.0, 0.0, 1.0)
    },
    {
        type: "iPiece",
        styles: [
            {
                type: 1,
                orientation: [
                    vec2(0, 0),
                    vec2(1, 0),
                    vec2(-1, 0),
                    vec2(-2, 0)
                ]
            },
            {
                type: 2,
                orientation: [
                    vec2(0, 0),
                    vec2(0, 1),
                    vec2(0, -1),
                    vec2(0, -2)
                ]
            }
        ],
        color: vec4(0.0, 1.0, 1.0, 1.0)
    },
    {
        type: "sPiece",
        styles: [
            {
                type: 1,
                orientation: [
                    vec2(0, 0),
                    vec2(1, 0),
                    vec2(0, -1),
                    vec2(-1, -1)
                ]
            },
            {
                type: 2,
                orientation: [
                    vec2(0, 0),
                    vec2(0, 1),
                    vec2(1, 0),
                    vec2(1, -1)
                ]
            }
        ],
        color: vec4(0.0, 1.0, 0.0, 1.0)
    },
    {
        type: "zPiece",
        styles: [
            {
                type: 1,
                orientation: [
                    vec2(0, 0),
                    vec2(-1, 0),
                    vec2(0, -1),
                    vec2(1, -1)
                ]
            },
            {
                type: 2,
                orientation: [
                    vec2(0, 0),
                    vec2(0, -1),
                    vec2(1, 0),
                    vec2(1, 1)
                ]
            }
        ],
```

```
    type: "jPiece",
    styles: [
        {
            type: 1,
            orientation: [
                vec2(0, 0),
                vec2(-1, 0),
                vec2(1, 0),
                vec2(1, -1)
            ]
        },
        {
            type: 2,
            orientation: [
                vec2(0, 0),
                vec2(0, -1),
                vec2(0, 1),
                vec2(1, 1)
            ]
        },
        {
            type: 3,
            orientation: [
                vec2(0, 0),
                vec2(-1, 0),
                vec2(-1, 1),
                vec2(1, 0)
            ]
        },
        {
            type: 4,
            orientation: [
                vec2(0, 0),
                vec2(0, -1),
                vec2(0, 1),
                vec2(-1, -1)
            ]
        }
    ],
    color: vec4(0.169, 0.0, 1.0, 1.0)
},
{
    type: "lPiece",
    styles: [
        {
            type: 1,
            orientation: [
                vec2(0, 0),
                vec2(-1, 0),
                vec2(1, 0),
                vec2(-1, -1)
            ]
        },
        {
            type: 2,
            orientation: [
                vec2(0, 0),
                vec2(0, -1),
                vec2(0, 1),
                vec2(1, -1)
            ]
        },
        {
            type: 3,
            orientation: [
                vec2(0, 0),
                vec2(-1, 0),
                vec2(1, 0),
                vec2(1, 1)
            ]
        },
        {
            type: 4,
            orientation: [
                vec2(0, 0),
                vec2(0, -1),
                vec2(0, 1),
                vec2(-1, 1)
            ]
        }
    ],
    color: vec4(1.0, 0.631, 0.0, 1.0)
},
```

```
type: "tPiece",
styles: [
    {
        type: 1,
        orientation: [
            vec2(0, 0),
            vec2(-1, 0),
            vec2(1, 0),
            vec2(0, -1)
        ]
    },
    {
        type: 2,
        orientation: [
            vec2(0, 0),
            vec2(0, -1),
            vec2(0, 1),
            vec2(1, 0)
        ]
    },
    {
        type: 3,
        orientation: [
            vec2(0, 0),
            vec2(-1, 0),
            vec2(1, 0),
            vec2(0, 1)
        ]
    },
    {
        type: 4,
        orientation: [
            vec2(0, 0),
            vec2(0, -1),
            vec2(0, 1),
            vec2(-1, 0)
        ]
    }
],
color: vec4(0.737, 0.0, 1.0, 1.0)
},
```

- 앞서 설명 드린 테트리스 피스에 대한 코드입니다. 전부 네모 블록이기에 vec2를 이용해 좌표를 설정했습니다. 또한 방향키를 눌렀을 때 모양이 바뀌어야 하므로 type을 설정해 모양이 바뀌는 좌표도 설정했습니다.

그 외) 그 외 코딩은 코딩에 주석을 달았습니다.

(4) 느낀점

이번 프로젝트를 통해 Webgl이 어떻게 구현이 되는지 전반적인 구조를 이해할 수 있었습니다. 하지만 three.js를 하지 못한 건 아쉽지만 과제를 기반으로 할 수 있는 과제를 선정하였기에 난이도는 떨어져도 의미있는 과제였습니다.

사실 프로젝트를 진행하면서 어려운 점이 많았습니다. 평상시 과제와 달리 어려웠습니다. 그래서 오픈소스들을 많이 찾아보았고 이해할 수 있는 오픈소스를 선정하였습니다. 코딩을 분석하여 제 코딩으로 옮겼습니다. 이러

한 과정에서 전반적인 구조를 이해할 수 있었으며 모르는 부분을 검색하여 알아갈 수 있었습니다.

초기에 프로젝트 제안서로 제출한 내용이 아닌 수정해서 구현했습니다. 초기 프로젝트 제안서를 진행했을 때 많은 자료들을 찾아보지 못하고 교수님이 올려준 자료만 참고하여 만들려고 했기에 그랬던 것 같습니다. 하지만 오픈소스들을 더 찾아보고 내가 할 수 있고 이해할 수 있는 분야를 선정하여 프로젝트 주제를 재선정 하게 되었습니다.

비록 대면수업이 아닌 비 대면으로 이 수업을 받은 것이 아쉽습니다. 왜냐면 웹과 관련된 강의가 과에 별로 없기 때문입니다. 하지만 과제들을 풀어보면서 실습에 대한 효과를 얻을 수 있었으며, 비 대면으로 한 결과 프로젝트를 접할 수 있었다고 생각합니다.