

컴퓨터 그래픽스 최종 프로젝트 발표

# 3D Bounce Ball

팀 : 굽신교  
2013182018 박민욱  
2013182021 박장호



# CONTENTS

01

---

게임 소개

02

---

구조 소개

03

---

진행 사항

04

---

역할 분담

05

---

주요 코드

06

---

문제점 및 해  
결 방안

“

## 게임 소개

”



바운스볼  
RAON GAMES



RAON GAMES의 바운스볼을 모티브로 만든 게임입니다.

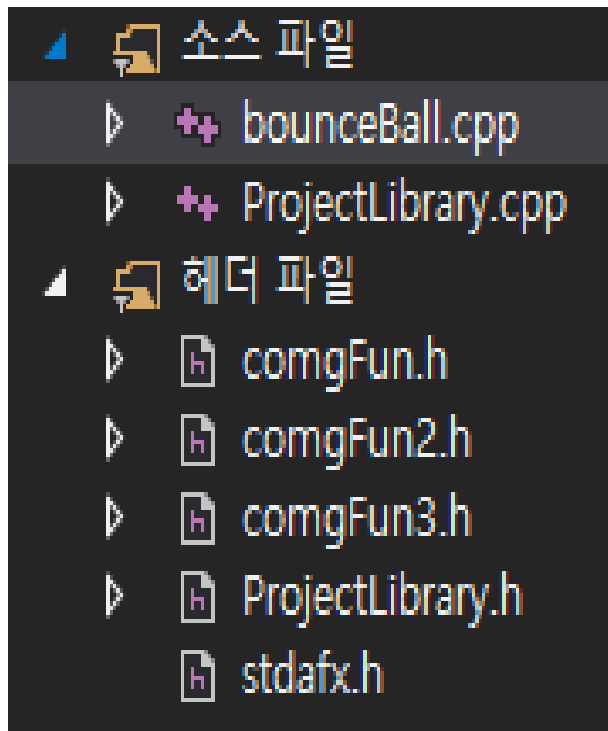
공의 시점 회전을 이용한 이동이 가능합니다.

스테이지 10개 별 각양각색 테마를 적용했습니다.

“

## 구조 소개

”



팀원 박민욱 자체 제작 라이브러리 사용

(강의 내용 및 과제 + freeglut기반 라이브러리)

# “ 진행 사항 ”

로고/타이틀 화면

오브젝트 중력 적용

바운딩 박스를 이용한 충돌 체크

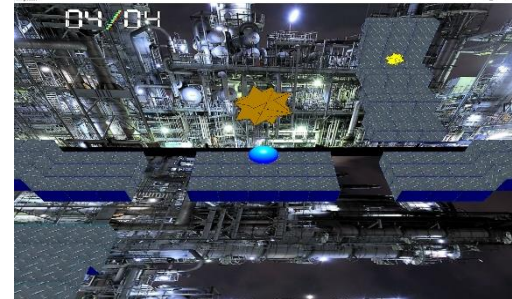
카메라 회전

맵 설계 및 제작

조명 처리 및 오브젝트에 대한 스포트 라이트

배경과 일부 오브젝트 텍스처 매핑

배경음/효과음



## “역할 분담”

### 박민욱

게임 프레임워크 제작  
오브젝트 운동 구현  
카메라 시점 변환 구현  
로고/타이틀 화면 전환 구현  
전반적 키 입력 구현  
게임 기획 담당  
충돌처리

### 박장호

맵 설계 및 구현  
리소스 수집  
텍스처 매핑  
조명 구현  
사운드 처리  
게임 테마 담당

## “ 주요 코드 (메인) ”

```
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(1000, 1000);
    glutCreateWindow("glutBase");
    initLighting();

    xRotated = yRotated = zRotated = 0.0;

    //glutIdleFunc(idleFunc);
    glutTimerFunc(100, TimerFunc, 1);
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);

    glutMouseFunc(Mouse);
    glutMouseWheelFunc(MouseWheel);
    glutMotionFunc(MouseDrag);
    glutKeyboardFunc(Keyboard);
    glutKeyboardUpFunc(KeyboardUp);
    Start();
    glutMainLoop();
    return 0;
}
```

메인 함수입니다.

화면 모드나 사이즈, 조명 초기화, x/y/z축 초기화  
및 glut 라이브러리에서 사용할 함수들을 호출합  
니다.

TimerFunc에서 bounceBall의 움직임을,  
display 함수에서 화면을 보여주는 작업을 진행  
합니다.

Start 함수는 게임 기본 정보를 초기화하는 함수  
입니다. 초기화 후 MainLoop로 들어갑니다.

## “ 주요 코드 (메인) ”

```
void Start()
{
    srand((unsigned)time(NULL));

    masterMatrix->thisCamera.eyeY += 1000.0;
    masterMatrix->thisCamera.eyeZ += 1000.0;

    simpleGameState = LogoState;

    LoadStage(currentStageIndex);
    LoadResource();

    bounceBall->InitializeBall(50, false, false,
        GLVector3(0, 0, 0), GLVector3(0, 15, 0), 1);
    bounceBall->boundingBoxSize = 45;

    testClass2->color4D = GLVector4(0.7, 0.5, 0, 1);
}
```

Start 함수입니다. 게임에 필요한 정보를 한꺼번에 초기화합니다.

masterMatrix는 월드 전체의 화면에 적용될 행렬입니다. 멤버에 화면 관련 변수나 로직 등이 담겨 있습니다.(카메라)

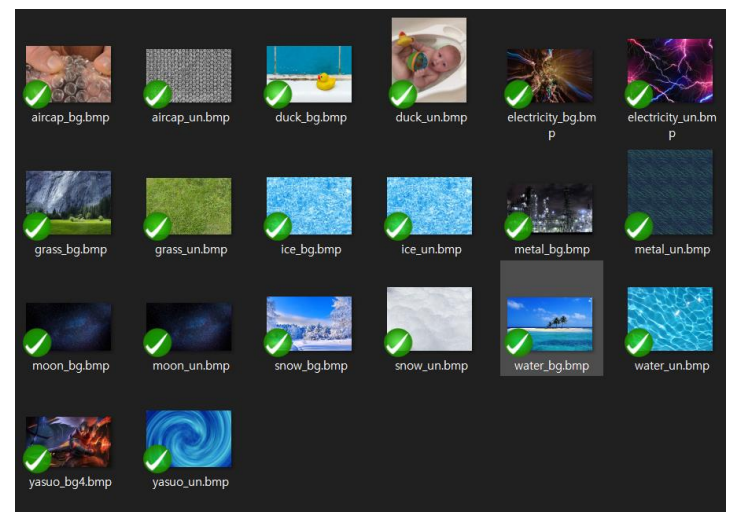
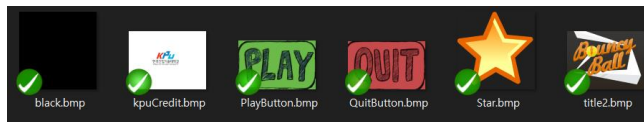
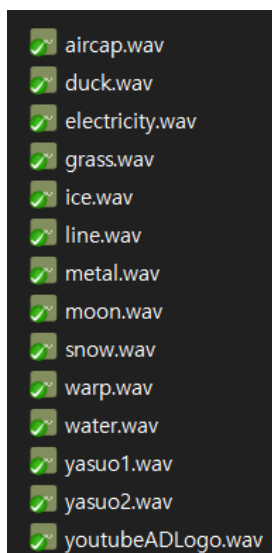
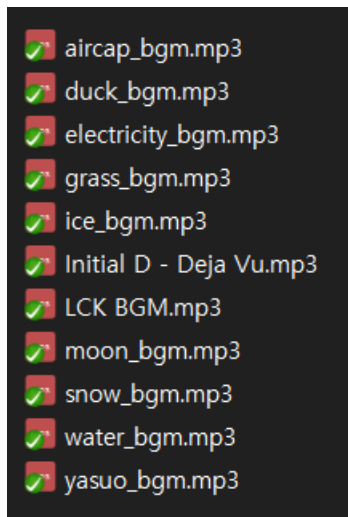
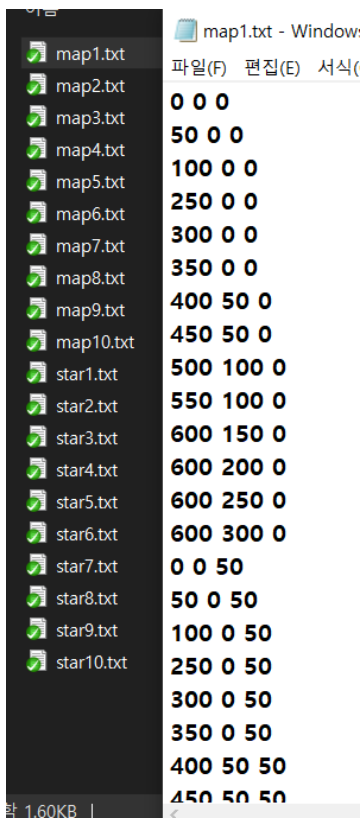
게임 상태에 따라 게임 진행이 달라집니다. 처음은 LogoState로 시작합니다.

Stage와 Resource를 로드합니다.

bounceBall은 유저가 직접 움직일 오브젝트입니다.



# “ 주요 코드(맵 설계) ”



설계/구현한 맵 데이터 및  
수집한 리소스입니다.

## “ 주요 코드(맵 구현) ”

```
void LoadStage(int stageNumber)
{
    FILE *fp_map1, *fp_star1;

    string str1 = "mapData/map";
    string str2 = "mapData/star";
    string extension = ".txt";
```

```
    fopen_s(&fp_map1, str1.c_str(), "r");
    while (fscanf_s(fp_map1, "%lf %lf %lf", &(map_1[map1_index].x), &(map_1[map1_index].y), &(map_1[map1_index].z)) != EOF)
    {
        map1_index++;
    }
    fclose(fp_map1);

    fopen_s(&fp_star1, str2.c_str(), "r");
    while (fscanf_s(fp_star1, "%lf %lf %lf", &(star_1[star1_index].x), &(star_1[star1_index].y), &(star_1[star1_index].z)) != EOF)
    {
        star1_index++;
    }
    fclose(fp_star1);
```

```
for (int i = 0; i < map1_index; i++)
{
    if (cubes.size() <= i)
    {
        cubes.push_back(Polygon0(masterMatrix));
    }
    cubes[i].boundingBoxSize = 50;

    cubes[i].SetPosition(map_1[i].x * 2, map_1[i].y * 2, map_1[i].z * 2);
}

for (int i = 0; i < star1_index; i++)
{
    if (stars.size() <= i)
    {
        stars.push_back(Polygon0(masterMatrix));
    }
    stars[i].boundingBoxSize = 50;

    stars[i].SetPosition(star_1[i].x * 2, star_1[i].y * 2, star_1[i].z * 2);
    stars[i].color4D = GLVector4(1, 1, 0, 1);
    stars[i].objectActivate = true;
}
```

맵 데이터를 읽어와 벡터(cubes)에 넣고, 이를 위치시킵니다.

## “ 주요 코드(텍스처 매핑) ”

```
GLubyte * LoadDIBitmap(const char *filename, BITMAPINFO **info);
std::vector<BITMAPINFO*> m_bitInfo; // 비트맵 헤더 저장할 변수
std::vector<GLubyte*> m_bitmap; // 비트맵 데이터를 가리킬 포인터
std::vector<GLuint> texture;

std::vector<BITMAPINFO*> backgroundBitmapInfo; // 비트맵 헤더 저장할 변수
std::vector<GLubyte*> backgroundBitmapData; // 비트맵 데이터를 가리킬 포인터
std::vector<GLuint> backgroundBitmapTexture;

std::vector<BITMAPINFO*> backgroundBitmapInfo2; // 비트맵 헤더 저장할 변수
std::vector<GLubyte*> backgroundBitmapData2; // 비트맵 데이터를 가리킬 포인터
std::vector<GLuint> backgroundBitmapTexture2;

std::vector<BITMAPINFO*> uIBitmapInfo; // 비트맵 헤더 저장할 변수
std::vector<GLubyte*> uIBitmapData; // 비트맵 데이터를 가리킬 포인터
std::vector<GLuint> uIBitmapTexture;
```

```
texture.push_back(GLint());
m_bitInfo.push_back(&BITMAPINFO());
m_bitmap.push_back(LoadDIBitmap("tiles/ice_box.bmp", &m_bitInfo.back()));
```

```
backgroundBitmapTexture2.push_back(GLint());
backgroundBitmapInfo2.push_back(&BITMAPINFO());
backgroundBitmapData2.push_back(LoadDIBitmap("background/ice_un.bmp", &backgroundBitmapInfo2.back()));
```

cubes 벡터는 PolygonO이라는 클래스를 자료형으로 받는 벡터입니다. 이 벡터에는 DrawTexture~ 함수가 있는데 이 함수에 매개변수로 전달할 리소스들을 미리 정리합니다.

## “ 주요 코드(텍스처 매핑) ”

```
void display(void)
{
```

```
    glColor4f(1, 1, 1, 1);
    glEnable(GL_TEXTURE_2D);
    for (int i = 0; i < map1_index; ++i)
    {
        cubes[i].DrawTextureCube(50, texture[currentStageIndex]);
    }
    glDisable(GL_LIGHTING);
    if (quadOrSphereMap)
    {
        backgroundCube->DrawTextureCube2(5000,
            backgroundBitmapTexture[currentStageIndex], backgroundBitmapTexture2[currentStageIndex]);
    }
    else
    {
        backgroundCube->DrawTextureSphere(7500, 30, backgroundBitmapTexture[currentStageIndex]);
    }
    glEnable(GL_LIGHTING);
    glDisable(GL_TEXTURE_2D);

    for (int i = 0; i < star1_index; ++i)
    {
        stars[i].DrawStar(50);
    }
}
```

display 함수에서, 앞서 로드했던 리소스들이 담긴 변수들과 지정했던 위치값을 이용해 배경과 맵을 그립니다.

## “ 주요 코드 (조명 구현) ”

```
void initLighting()
{
    // Enable lighting
    glEnable(GL_LIGHTING);

    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, qaAmbientLight);
    glLightModelfv(GL_LIGHT_MODEL_LOCAL_VIEWER, 0.0);
    glLightModelfv(GL_LIGHT_MODEL_TWO_SIDE, 0.0);

    //핵심코드
    // Set lighting intensity and color
    glLightfv(GL_LIGHT3, GL_AMBIENT, qaAmbientLight);
    glLightfv(GL_LIGHT3, GL_DIFFUSE, qaDiffuseLight);
    glLightfv(GL_LIGHT3, GL_SPECULAR, qaSpecularLight);

    qaSpotLightPosition[0] = bounceBall->Position().x;
    qaSpotLightPosition[1] = bounceBall->Position().y + 100;
    qaSpotLightPosition[2] = bounceBall->Position().z;
    qaSpotLightPosition[3] = 1;

    glLightfv(GL_LIGHT4, GL_AMBIENT, qaAmbientLight);
    glLightfv(GL_LIGHT4, GL_DIFFUSE, qaDiffuseLight);
    glLightfv(GL_LIGHT4, GL_SPECULAR, qaSpecularLight);

    glLightfv(GL_LIGHT4, GL_POSITION, qaSpotLightPosition);
    glLightfv(GL_LIGHT4, GL_SPOT_DIRECTION, qaSpotLightDirection);
    glLightf(GL_LIGHT4, GL_SPOT_CUTOFF, spotLightDegree);
    glLightf(GL_LIGHT4, GL_SPOT_EXPONENT, spotLightExponent);

    // Set the light position
    glLightfv(GL_LIGHT3, GL_POSITION, qaLightPosition);

    glEnable(GL_DEPTH_TEST);
}
```

glLightModelfv 함수로 조명 모델 매개변수 AMBIENT를 설정합니다.

glLightfv 함수로 AMBIENT, DIFFUSE, SPECULAR 각 조명에 3번, 4번 LIGHT를 입히고 위치를 설정합니다. bounceBall을 따라다닐 SpotLight 조명 또한 같은 방법으로 설정합니다.

## “ 주요 코드(사운드 처리) ”

```
//철,아이스,뽕뽕이,야스오,오리,전기,초원,달,눈,물
```

```
LPCSTR soundFileName[]
```

```
{
```

```
    "wavFile/metal.wav",  
    "wavFile/ice.wav",  
    "wavFile/aircap.wav",  
    "wavFile/yasuo1.wav",  
    "wavFile/duck.wav",  
    "wavFile/electricity.wav",  
    "wavFile/grass.wav",  
    "wavFile/moon.wav",  
    "wavFile/snow.wav",  
    "wavFile/water.wav",  
    "wavFile/yasuo2.wav",  
    "wavFile/line.wav",  
    "wavFile/warp.wav",  
    "wavFile/youtubeADLogo.wav"
```

```
};
```

```
LPCSTR musicFileName[]
```

```
{
```

```
    "open \\mp3\\LCK BGM.mp3\\" type mpegvideo alias mp3",  
    "open \\mp3\\ice_bgm.mp3\\" type mpegvideo alias mp3",  
    "open \\mp3\\aircap_bgm.mp3\\" type mpegvideo alias mp3",  
    "open \\mp3\\Initial D - Deja Vu.mp3\\" type mpegvideo alias mp3",  
    "open \\mp3\\duck_bgm.mp3\\" type mpegvideo alias mp3",  
    "open \\mp3\\electricity_bgm.mp3\\" type mpegvideo alias mp3",  
    "open \\mp3\\grass_bgm.mp3\\" type mpegvideo alias mp3",  
    "open \\mp3\\moon_bgm.mp3\\" type mpegvideo alias mp3",  
    "open \\mp3\\snow_bgm.mp3\\" type mpegvideo alias mp3",  
    "open \\mp3\\water_bgm.mp3\\" type mpegvideo alias mp3",  
    "open \\mp3\\yasuo_bgm.mp3\\" type mpegvideo alias mp3"
```

```
};
```

먼저 사운드를 재생하기 위해 리소스 파일을 배열에 담아줍니다.

## “ 주요 코드(사운드 처리) ”

```
for (int i = 0; i < star1_index; ++i)
{
    if (stars[i].objectActivate)
    {
        if (bounceBall->CollisionCheckOriginal(*bounceBall, stars[i]))
        {
            stars[i].objectActivate = false;
            PlaySoundA(soundFileName[11], NULL, SND_ASYNC);
        }
    }
}
```

```
void BounceBall::StateEnter()
{
    else if (this->bounceBallProperty.state == JumpUp)
    {
        this->bounceBallProperty.velocity.y = this->bounceBallProperty.jumpPower.y;
        if (this->soundFileName != "")
        {
            PlaySoundA(this->soundFileName, NULL, SND_ASYNC | SND_NOSTOP);
        }
    }
    else if (this->bounceBallProperty.state == SuperJumpup)
    {
        this->bounceBallProperty.velocity.y = this->bounceBallProperty.superJumpPower.y;
        if (this->soundFileName != "")
        {
            PlaySoundA(this->soundFileName, NULL, SND_ASYNC | SND_NOSTOP);
        }
    }
}
```

스테이지의 별과 닿으면 사운드가 재생되거나, 스테이지 종류에 따라 공 튀기는 소리가 달라지게 재생되도록 처리했습니다.

이외에도 Title 사운드나 BGM, 공이 바닥으로 추락했을 때의 소리도 구현했습니다.

## “ 문제점 및 해결방안 ”

팀원이 수업 시작부터 기말 팀 프로젝트까지 본인이 만든 라이브러리를 써야겠다 주장해 난감했습니다.(약 9000줄 정도 되는 라이브러리를 기말 프로젝트라는 짧은 기간에 다 받아들이기 힘들었습니다.)

저는 freeglut 라이브러리만 다룰 줄 알았지만, 제가 구현해야 하는 부분에 어떤 함수를 어떻게 쓰면 좋을지 소통하여 해결할 수 있었습니다.

```
ProjectLibrary.cpp  ↵ ✕
{
8669     glVertex3f(cube_vertices[cube_faces[0][i]][0] * size, c
8670     glEnd();
8671
8672
8673
8674     glEnable(GL_LIGHTING);
8675     glDisable(GL_TEXTURE_2D);
8676     this->masterMatrixs->glLoadMasterMatrix();
8677
8678
8679 }
8680
8681 }
8682
8683
8684
```