COMPUTER GRAPICS TERM PROJECT

대화형 3차원 카탈로그 제작

정보통신공학과

12201933 이승은

제품 선택

아이언맨 슈트

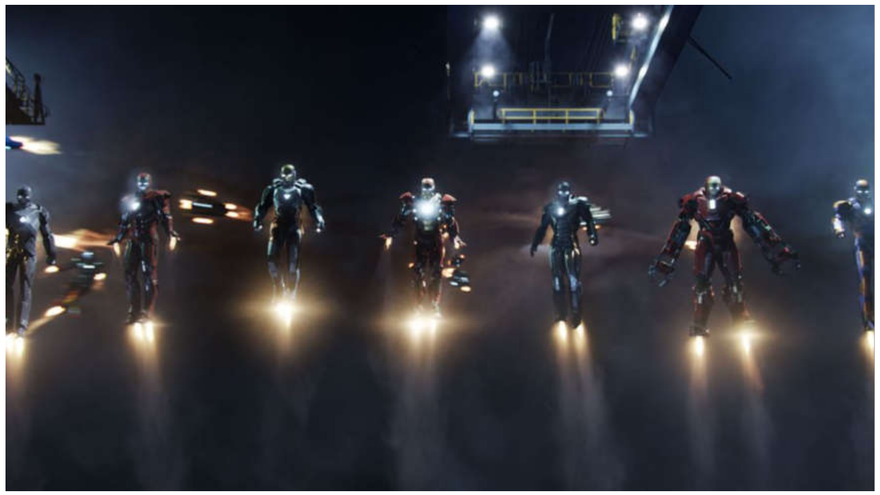
기능 소개

리펄서 건

황색의 빔을
발사하는 무기

스마트 탄환

슈트의 어께 부분에
수납된 탄환



하우스파티 프로토콜

아이언맨3 에서 등장한
아이언맨 슈트들이 일제히 출격하여
적을 공격하는 프로토콜

키보드 입력

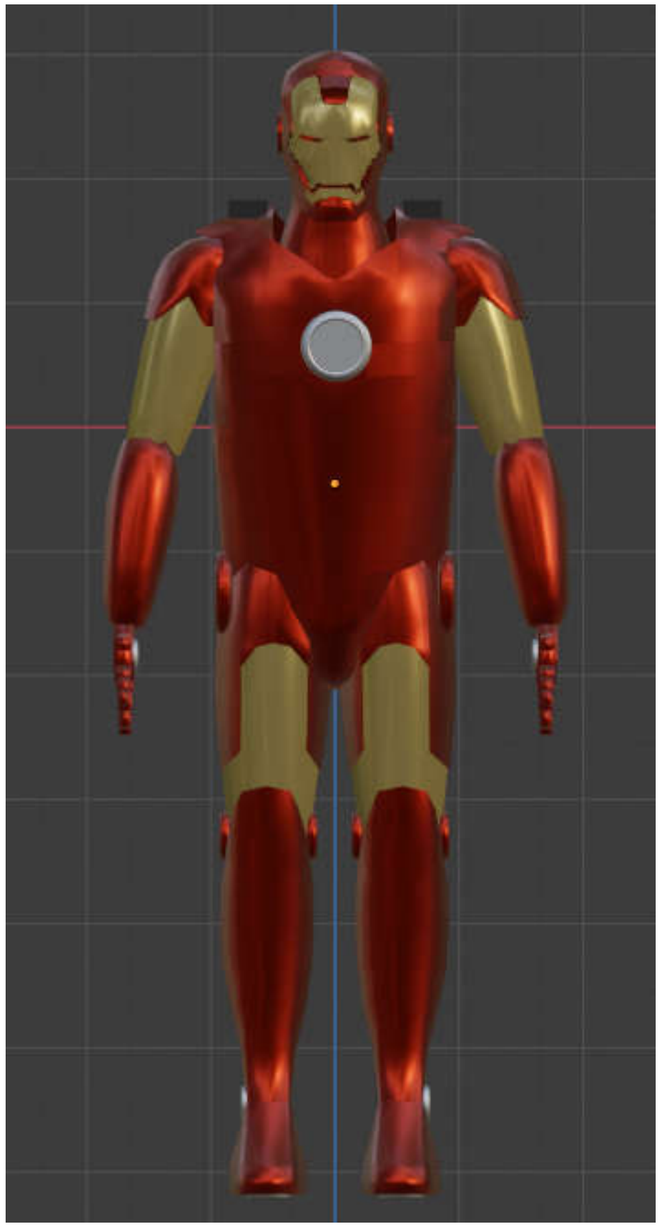
버튼	기능
→	카메라 오른쪽으로 회전
←	카메라 왼쪽으로 회전
↑	카메라 위쪽으로 회전
↓	카메라 아래쪽으로 회전
r	리펄서 건
s	스마트 탄환
h	하우스파티 프로토콜

마우스 입력

버튼	기능
mouse wheel	Zoom In & Out
left button	스마트 탄환 mode에서 탄환 발사
right button	메인 메뉴
	init, 종료
	색상 변경 (red, pink, blue)
	배경 변경 (hall, universe, ocean)
	Assemble
	Individual

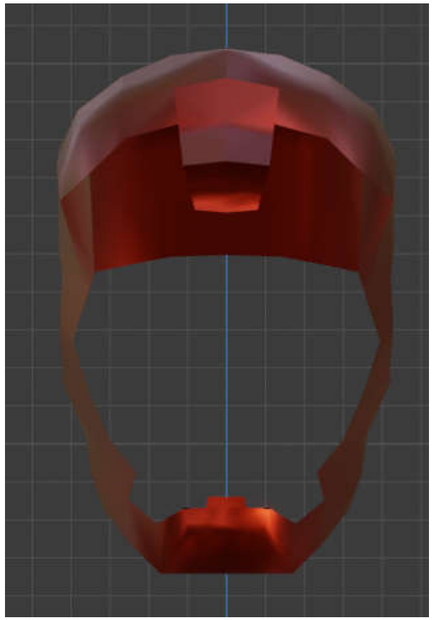
제품 모델링


Collection
Arc
Arm1 Gold
Arm1 Red
Arm2
Body
Face
Foot Arc
Gun
Hand
Hand Arc
Helmet
Leg1 Gold
Leg1 Red
Leg2
Neck
Silver



모델링 상세

Face

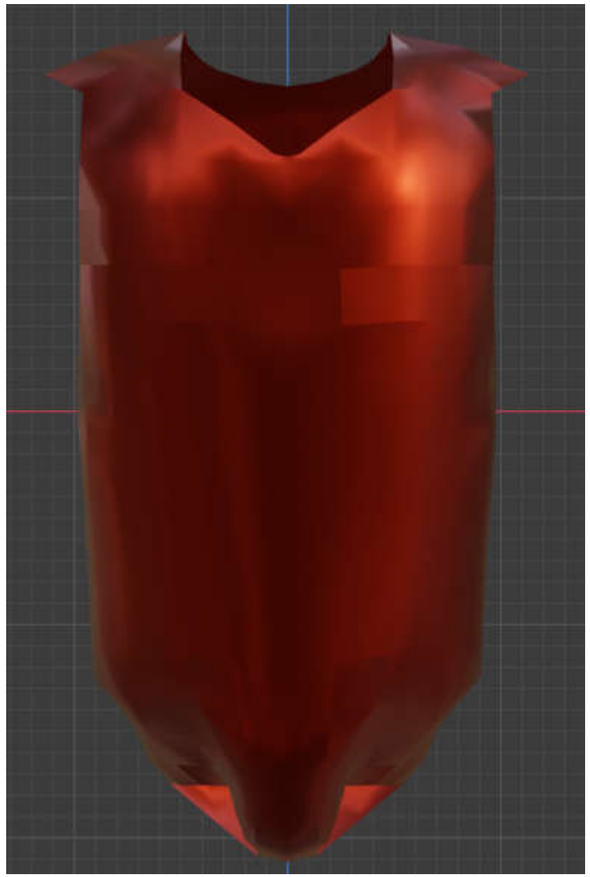


Helmet

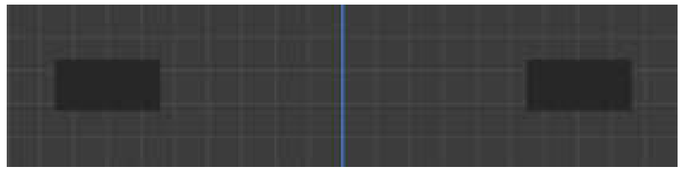
Neck

->

모델링 상세



body

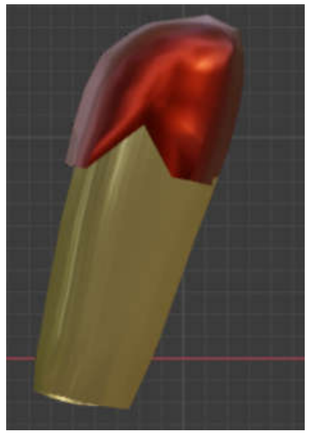


Gun

Arc

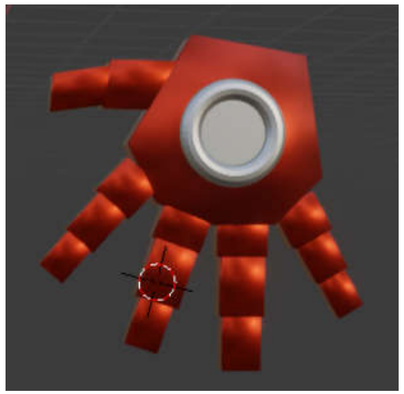
->

모델링 상세



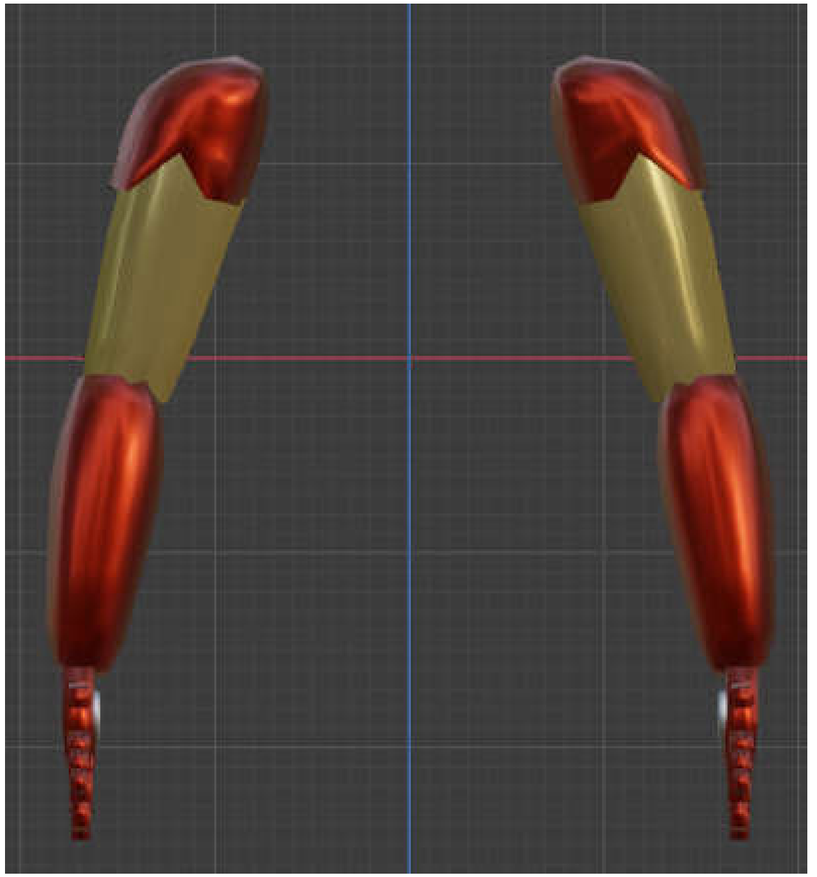
Arm1 Gold +
Arm1 Red

Arm2

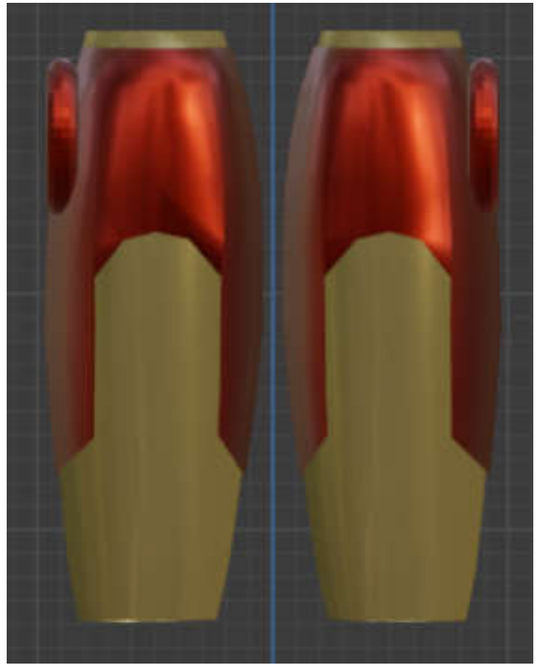


Hand + Hand Arc

->



모델링 상세

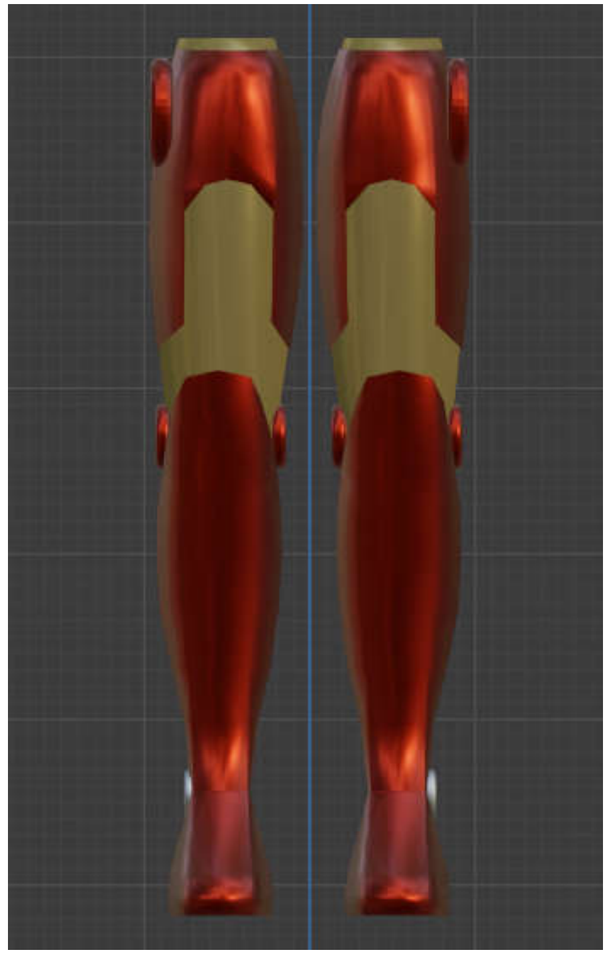


Leg1 Gold +
Leg1 Red

Leg2

Silver

->



기능 구현

```
void setTextureMapping() {
    int imgWidth, imgHeight, channels;
    uchar* img = readImageData("img/gold.bmp", &imgWidth, &imgHeight, &channels); // gold
    glGenTextures(1, &goldtexture);
    glBindTexture(GL_TEXTURE_2D, goldtexture);
    glTexImage2D(GL_TEXTURE_2D, 0, 3, imgWidth, imgHeight, 0, GL_RGB, GL_UNSIGNED_BYTE, img);
    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
```

↑ setTextureMapping 함수 :
bmp 파일을 불러와 텍스처로 지정

```
void draw_obj(ObjParser* objParser, GLuint texture) { // draw ironman obj
    glDisable(GL_BLEND);
    glBindTexture(GL_TEXTURE_2D, texture); // bind texture
    glBegin(GL_TRIANGLES);
    for (unsigned int n = 0; n < objParser->getFaceSize(); n += 3) {
        glNormal3f(objParser->normal[objParser->normalIdx[n] - 1].x,
            objParser->normal[objParser->normalIdx[n] - 1].y,
            objParser->normal[objParser->normalIdx[n] - 1].z);
        glVertex3f(objParser->vertices[objParser->vertexIdx[n] - 1].x,
            objParser->vertices[objParser->vertexIdx[n] - 1].y,
            objParser->vertices[objParser->vertexIdx[n] - 1].z);

        glNormal3f(objParser->normal[objParser->normalIdx[n + 1] - 1].x,
            objParser->normal[objParser->normalIdx[n + 1] - 1].y,
            objParser->normal[objParser->normalIdx[n + 1] - 1].z);
        glVertex3f(objParser->vertices[objParser->vertexIdx[n + 1] - 1].x,
            objParser->vertices[objParser->vertexIdx[n + 1] - 1].y,
            objParser->vertices[objParser->vertexIdx[n + 1] - 1].z);

        glNormal3f(objParser->normal[objParser->normalIdx[n + 2] - 1].x,
            objParser->normal[objParser->normalIdx[n + 2] - 1].y,
            objParser->normal[objParser->normalIdx[n + 2] - 1].z);
        glVertex3f(objParser->vertices[objParser->vertexIdx[n + 2] - 1].x,
            objParser->vertices[objParser->vertexIdx[n + 2] - 1].y,
            objParser->vertices[objParser->vertexIdx[n + 2] - 1].z);
    }
    glEnd();
    glEnable(GL_BLEND);
}
```

← draw_obj 함수 :
obj 파일을 load하고 지정한 texture를 바인딩

기능 구현

draw 함수 :
individual, assemble, repulsorbeam, color 변수 값에 따라 ironman을 그림

```
void draw_ironman() {
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    if (individual == 0) { // indiviual == 0 -> the whole ironman obj
        if (assemble == 1) { // assemble == 1 -> translate
            glPushMatrix();
            glTranslatef(0, 0, 10 - a); // face translate
        }
        draw_obj(face, goldtexture); // draw face with gold texture
        if (assemble == 1) {  // if assemble is on
            glPopMatrix();
            glPushMatrix();
            glTranslatef(-10 + a, 0, 0); // arm1goldleft translate
        }
        draw_obj(arm1goldleft, goldtexture); // draw arm1goldleft with gold texture
```

assemble의 값이 1이면
idle 함수에서 사용한 변수 a를
glTranslatef에 적용해서
각 부품의 움직임 애니메이션 구현

```
    else if (individual == 30) { // individual - face
        draw_obj(face, goldtexture);
    }
    else if (individual == 31) { // individual - helmet
        if (color == 0) draw_obj(helmet, redtexture);
        else if (color == 10) draw_obj(helmet, pinktexture);
        else if (color == 11) draw_obj(helmet, bluetexture);
    }
```

individual의 값이 00이면
ironman 전체를 그리고
individual의 값이 00이 아니면
각 individual의 값에 따라 선택된 부분만을 그림

기능 구현

draw 함수 :
individual, assemble, repulsorbeam, color 변수 값에 따라 ironman을 그림

```
if (assemble == 1) { // if assemble is on
    glPopMatrix();
    glPushMatrix();
    glTranslatef(0, 0, 10 - a); // arm1redright translate
}
if (repulsorbeam == 1) { // if repulsor beam is on
    glPushMatrix();
    glRotatef(-r, 1, 0, 0); // arm1redright rotate
    glTranslatef(0, -0.6, 1.4);
}
if (color == 0) draw_obj(arm1redright, redtexture); // draw arm1redright with red texture if color == 0
else if (color == 10) draw_obj(arm1redright, pinktexture); // draw arm1redright with pink texture if color == 10
else if (color == 11) draw_obj(arm1redright, bluetexture); // draw arm1redright with blue texture if color == 11
if (repulsorbeam == 1) { // if repulsor beam is on
    glPopMatrix();
}
if (assemble == 1) { // if assemble is on
    glPopMatrix();
```

repulsorbeam의 값이 10이면
오른쪽 팔이 회전하도록 idle 함수에서 사용한 변수 r을 glRotatef 함수에 적용

color의 값이 00이면 red texture mapping
10이면 pink texture를 mapping
11이면 blue texture를 mapping

기능 구현

```
void setEnvironmentMap() {
    int imgWidth, imgHeight, channels;
    /* hall of armor */
    uchar* img0 = readImageData("img/lefthall.bmp", &imgWidth, &imgHeight, &channels);
    uchar* img1 = readImageData("img/righthall.bmp", &imgWidth, &imgHeight, &channels);
    uchar* img2 = readImageData("img/black.bmp", &imgWidth, &imgHeight, &channels);
    uchar* img3 = readImageData("img/black.bmp", &imgWidth, &imgHeight, &channels);
    uchar* img4 = readImageData("img/black.bmp", &imgWidth, &imgHeight, &channels);
    uchar* img5 = readImageData("img/casee1.bmp", &imgWidth, &imgHeight, &channels);

    glGenTextures(1, &hallCubeTex);
    glBindTexture(GL_TEXTURE_CUBE_MAP, hallCubeTex);
    glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X, 0, GL_RGBA, imgWidth, imgHeight, 0, GL_RGB, GL_UNSIGNED_BYTE, img0);
    glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_X, 0, GL_RGBA, imgWidth, imgHeight, 0, GL_RGB, GL_UNSIGNED_BYTE, img1);
    glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Y, 0, GL_RGBA, imgWidth, imgHeight, 0, GL_RGB, GL_UNSIGNED_BYTE, img2);
    glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Y, 0, GL_RGBA, imgWidth, imgHeight, 0, GL_RGB, GL_UNSIGNED_BYTE, img3);
    glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Z, 0, GL_RGBA, imgWidth, imgHeight, 0, GL_RGB, GL_UNSIGNED_BYTE, img4);
    glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Z, 0, GL_RGBA, imgWidth, imgHeight, 0, GL_RGB, GL_UNSIGNED_BYTE, img5);

    glBindTexture(GL_TEXTURE_CUBE_MAP, hallCubeTex);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_WRAP_R, GL_CLAMP_TO_EDGE);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
```

← setEnvironmentMap 함수:
6개의 bmp 파일을 불러와 cube 형태의 텍스처로 지정
한번의 바인딩으로 6개의 영상 지정
텍스처 속성 지정
텍스처 좌표 자동 생성

* 코드 일부분만 캡쳐

draw_skyBox 함수:
Cube texture를 이용하여 sky box 드로잉 →

* 코드 일부분만 캡쳐

```
void draw_skyBox(GLuint texture) {
    glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
    glDisable(GL_LIGHTING);
    glEnable(GL_TEXTURE_CUBE_MAP);
    glBindTexture(GL_TEXTURE_CUBE_MAP, texture); // bind texture
    float g_nSkysize = 200;
    glBegin(GL_QUADS);
    //px
    glTexCoord3d(1, -1, -1); glVertex3f(g_nSkysize, -g_nSkysize, -g_nSkysize);
    glTexCoord3d(1, -1, 1); glVertex3f(g_nSkysize, -g_nSkysize, g_nSkysize);
    glTexCoord3d(1, 1, 1); glVertex3f(g_nSkysize, g_nSkysize, g_nSkysize);
    glTexCoord3d(1, 1, -1); glVertex3f(g_nSkysize, g_nSkysize, -g_nSkysize);
    //nx
    glTexCoord3d(-1, -1, -1); glVertex3f(-g_nSkysize, -g_nSkysize, -g_nSkysize);
    glTexCoord3d(-1, -1, 1); glVertex3f(-g_nSkysize, -g_nSkysize, g_nSkysize);
    glTexCoord3d(-1, 1, 1); glVertex3f(-g_nSkysize, g_nSkysize, g_nSkysize);
    glTexCoord3d(-1, 1, -1); glVertex3f(-g_nSkysize, g_nSkysize, -g_nSkysize);
```

기능 구현

```
void cubeTextureMapping() { // smart gun mode - cube texture
    glGenTextures(6, cubeTex);
    int imgWidth, imgHeight, channels;
    for (int i = 0; i < 6; i++) {
        glBindTexture(GL_TEXTURE_2D, cubeTex[i]);
        char buf[100];
        sprintf(buf, "img/TexImage%d.bmp", i);
        buf[strlen(buf)] = 0;
        unsigned char* img = readImageData(buf, &imgWidth, &imgHeight, &channels);
        glTexImage2D(GL_TEXTURE_2D, 0, 3, imgWidth, imgHeight, 0, GL_RGB, GL_UNSIGNED_BYTE, img);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    }
}
```

```
void sphereTextureMapping() { // smart gun mode
    glGenTextures(1, sphereTex);
    glBindTexture(GL_TEXTURE_2D, *sphereTex);
    int width, height, channels;
    unsigned char* img = readImageData("img/EARTH.bmp", &width, &height, &channels);
    glTexImage2D(GL_TEXTURE_2D, 0, 3, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, img);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
}
```

```
void cylinderTextureMapping() { // smart gun mode - cylinder texture
    glGenTextures(3, cylinderTex);

    glBindTexture(GL_TEXTURE_2D, cylinderTex[0]);
    int width, height, channels;
    unsigned char* img = readImageData("img/CIDER_T.bmp", &width, &height, &channels);
    glTexImage2D(GL_TEXTURE_2D, 0, 3, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, img);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);

    glBindTexture(GL_TEXTURE_2D, cylinderTex[1]);
    img = readImageData("img/coke.bmp", &width, &height, &channels);
    glTexImage2D(GL_TEXTURE_2D, 0, 3, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, img);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);

    glBindTexture(GL_TEXTURE_2D, cylinderTex[2]);
    img = readImageData("img/CIDER_B.bmp", &width, &height, &channels);
    glTexImage2D(GL_TEXTURE_2D, 0, 3, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, img);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
}
```

cubeTextureMapping 함수 : 반복문을 통해 6개의 bmp 파일을 불러와 각각 텍스처로 지정
sphereTextureMapping 함수 : bmp 파일을 불러와 텍스처 지정
cylinderTextureMapping 함수 : 3개의 bmp 파일을 불러와 각각 텍스처 지정

기능 구현

```
void draw_Cylinder() { // smart gun mode - draw cylinder
    glColor3f(1.0, 1.0, 1.0);

    /* 윗면 - Disk */
    glPushMatrix();
    glBindTexture(GL_TEXTURE_2D, cylinderTex[0]);
    glTranslatef(-2, 1, 0);
    glRotatef(-90, 1, 0, 0);
    gluDisk(qobj, 0, 1, 12, 1);
    glPopMatrix();

    /* 옆면 - Cylinder */
    glPushMatrix();
    glBindTexture(GL_TEXTURE_2D, cylinderTex[1]);
    glTranslatef(-2, -2, 0);
    glRotatef(-90, 1, 0, 0);
    gluCylinder(qobj, 1, 1, 3, 12, 1);
    glPopMatrix();

    /* 아랫면 - Disk */
    glPushMatrix();
    glBindTexture(GL_TEXTURE_2D, cylinderTex[2]);
    glTranslatef(-2, -2, 0);
    glRotatef(90, 1, 0, 0);
    gluDisk(qobj, 0, 1, 12, 1);
    glPopMatrix();
}
```

```
void draw_Sphere() { // smart gun mode - draw sphere
    glColor3f(1.0, 1.0, 1.0);
    glBindTexture(GL_TEXTURE_2D, *sphereTex);
    glRotatef(-90, 1, 0, 0);
    gluSphere(qobj, 2, 24, 24);
    glRotatef(90, 1, 0, 0);
}
```

```
void draw_textureCube() { // smart gun mode - draw cube
    glColor3f(1.0, 1.0, 1.0);
    glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);

    glBindTexture(GL_TEXTURE_2D, cubeTex[0]);
    glBegin(GL_QUADS);
    glNormal3f(-1.0, 0, 0);   // -x axis
    glTexCoord2f(0, 0); glVertex3f(-1.0, 1.0, -1.0);
    glTexCoord2f(1, 0); glVertex3f(-1.0, -1.0, -1.0);
    glTexCoord2f(1, 1); glVertex3f(-1.0, -1.0, 1.0);
    glTexCoord2f(0, 1); glVertex3f(-1.0, 1.0, 1.0);
    glEnd();

    glBindTexture(GL_TEXTURE_2D, cubeTex[1]);
    glBegin(GL_QUADS);
    glNormal3f(1.0, 0, 0);   // x axis
    glTexCoord2f(0, 0); glVertex3f(1.0, 1.0, 1.0);
    glTexCoord2f(1, 0); glVertex3f(1.0, -1.0, 1.0);
    glTexCoord2f(1, 1); glVertex3f(1.0, -1.0, -1.0);
    glTexCoord2f(0, 1); glVertex3f(1.0, 1.0, -1.0);
    glEnd();
```

draw_textureCube 함수 : cube를 그리고 각각 텍스처 바인딩 진행
sphereTextureMapping 함수 : sphere를 그리고 텍스처 바인딩 진행
cylinderTextureMapping 함수 : cylinder를 그리고 각각 텍스처 바인딩 진행

기능 구현

```
void idle() {
    if (assemble == 1) { // obj assemble
        a = a + 0.02;
        if (a >= 10) {
            a = 10;
        }
    }
    if (houseparty == 1) { // house party protocol
        h[0] = h[0] + 0.6;
        if (h[0] >= 20) {
            h[0] = 20;
        }
        h[1] = h[1] + 0.45;
        if (h[1] >= 20) {
            h[1] = 20;
        }
        h[2] = h[2] + 0.3;
        if (h[2] >= 20) {
            h[2] = 20;
        }
        h[3] = h[3] + 0.5;
        if (h[3] >= 25) {
            h[3] = 25;
        }
        h[4] = h[4] + 0.45;
        if (h[4] >= 25) {
            h[4] = 25;
        }
    }
    if (repulsorbeam == 1) { // repulsor beam
        r = r + 1; // rotation angle
        if (r >= 90) {
            r = 90;
        }
    }
    glutPostRedisplay();
}
```

idle 함수

assemble, houseparty,
repulsorbeam 기능에 따라
물체의 움직임,
애니메이션 효과를 주기 위해
컴퓨터의 유휴 시간에 하는 일인
idle 콜백함수를 작성

각 기능이 on 되면
움직임을 나타내는 변수의
값을 변경시킴

기능 구현

```
void housepartyProtocol() {
    // bottom
    glPushMatrix();
    draw_ironman();
    glPopMatrix();

    // bottom left
    glPushMatrix();
    glRotatef(15.0f, 0.0f, 1.0f, 0.0f);
    glTranslatef(-5, 0, -20 + h[0]);
    draw_ironman();
    glPopMatrix();

    glPushMatrix();
    glRotatef(15.0f, 0.0f, 1.0f, 0.0f);
    glTranslatef(-10, 0, -20 + h[1]);
    draw_ironman();
    glPopMatrix();

    glPushMatrix();
    glRotatef(15.0f, 0.0f, 1.0f, 0.0f);
    glTranslatef(-15, 0, -20 + h[2]);
    draw_ironman();
    glPopMatrix();
```

housepartyProtocol 함수

glPushMatrix(), glPopMatrix() 함수를 이용해
여러 오브젝트를 불러오고
idle 함수에서 사용한 변수 h를 통해 각각의 오브젝트에
애니메이션 효과를 줌

* 코드 일부분만 캡쳐

기능 구현

```
void smartGun() {
    glEnable(GL_TEXTURE_2D);
    if (heart[0] != 0) { // cube
        glPushMatrix();
        glTranslatef(5, 0, 11);
        glColor3f(1, 1, 1);
        draw_textureCube();
        glPopMatrix();

        glDisable(GL_TEXTURE_2D);
        glPushMatrix();
        glTranslatef(6.2, 2.1, 11); // heart
        for (int i = 0; i < heart[0]; i++) {
            glColor3f(1, 0, 0);
            glutSolidSphere(0.2, 30, 30);
            glTranslatef(-0.6, 0, 0);
        }
        glPopMatrix();
    }
```

smartGun 함수

glPushMatrix(), glPopMatrix() 함수를 이용해
여러 물체를 불러오고
반복문을 통해 남아 있는 생명의 개수만큼
물체의 위에 원을 그림

* 코드 일부분만 캡쳐

기능 구현

draw 함수 – 그리기 콜백 함수

화면, 카메라를 설정하고
smartgun 변수의 값이 1이면
smartGun 실행,
background 변수 값에 따라
배경을 설정하고
ironman에 배경이 반사되는
효과를 Disable함,
houseparty 변수의 값이 1이면
housepartyProtocol 실행,
아니면 draw_ironman 실행

```
void draw() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    cam[0] = radius * sin(theta * M_PI / 180) * sin(phi * M_PI / 180);
    cam[1] = radius * cos(theta * M_PI / 180);
    cam[2] = radius * sin(theta * M_PI / 180) * cos(phi * M_PI / 180);
    gluLookAt(cam[0], cam[1], cam[2], center[0], center[1], center[2], up[0], up[1], up[2]);

    if (smartgun == 1) { // smartgun mode
        smartGun();
    }

    if (background == 0) { // background : universe
        draw_skyBox(universeCubeTex);
    }
    else if (background == 1) { // hall of armor
        draw_skyBox(hallCubeTex);
    }
    else if (background == 2) { // ocean
        draw_skyBox(oceanCubeTex);
    }

    glDisable(GL_TEXTURE_GEN_S); // ironman's mirror affect off
    glDisable(GL_TEXTURE_GEN_T);
    glDisable(GL_TEXTURE_GEN_R);
    glDisable(GL_TEXTURE_CUBE_MAP);

    if (houseparty == 1) { // house party mode
        housepartyProtocol();
    }
    else {
        //draw_axis();
        draw_ironman();
    }
    glFlush();
    glutSwapBuffers();
```

기능 구현

```
void keyboard(unsigned char key, int x, int y) {
    if (key == 'R' || key == 'r') { // repulsor beam mode
        printf("Repulsor Beam mode has been selected\n");
        if (repulsorbeam == 0) { // off -> on
            smartgun = 0;
            repulsorbeam = 1;
            PlaySound("sound/repulsor.wav", 0, SND_FILENAME | SND_ASYNC);
        }
        else { // on -> off
            repulsorbeam = 0;
            r = 0;
        }
    }
```

keyboard 함수 – 키보드 콜백 함수

입력된 키보드의 값에 따라 각 mode에 필요한
변수 값을 변경하고
적절한 sound를 play함

*코드 일부분만 캡쳐

```
void specialkeyboard(int key, int x, int y) { // move camera position
    if (smartgun == 0) {
        if (key == GLUT_KEY_LEFT) {
            phi -= 2.5;
            if (phi < 0) phi = 355;
        }
        else if (key == GLUT_KEY_RIGHT) {
            phi += 2.5;
            if (phi >= 360) phi = 0;
        }
        else if (key == GLUT_KEY_UP) {
            if (theta > 10) theta -= 2.5;
        }
        else if (key == GLUT_KEY_DOWN) {
            if (theta < 170) theta += 2.5;
        }
    }
    glutPostRedisplay();
}
```

specialkeyboard 함수 – 특수키보드 콜백 함수

방향키 키보드의 입력에 따라
카메라의 위치를 이동
smartgun mode일때는 카메라 위치 고정

기능 구현

```
void mouse(int button, int state, int x, int y) {
    if (smartgun == 1) { // when smart gun mode is on
        if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) { // shoot
            //printf("x: %d, y: %d\n", x, y);
            if (x > 385 && x < 470 && y > 378 && y < 457 && heart[0] > 0) {
                heart[0]--; // cube
            }
            if (x > 561 && x < 634 && y > 318 && y < 388 && heart[1] > 0) {
                heart[1]--; // sphere
            }
            if (x > 765 && x < 822 && y > 356 && y < 437 && heart[2] > 0) {
                heart[2]--; // cylinder
            }
            PlaySound("sound/gun.wav", 0, SND_FILENAME | SND_ASYNC);
        }
        glutPostRedisplay();
    }
}
```

mouse 함수 – 마우스 콜백 함수

smartgun mode에서 왼쪽 마우스가 클릭되면
마우스 좌표 위치에 따라
해당 물체의 생명을 감소하고
적절한 sound를 play함

```
void mouseWheel(int button, int dir, int x, int y) {
    if (smartgun == 0) { // when smart gun mode is off
        if (dir > 0) {
            if (radius > 2) radius--; // zoom in
        }
        else {
            if (radius < 100) radius++; // zoom out
        }
        glutPostRedisplay();
    }
}
```

mouseWheel 함수 – 마우스휠 콜백 함수

마우스 휠을 올리면 radius 값을 감소해
카메라를 Zoom in하고
마우스 휠을 내리면 radius 값을 증가시켜
카메라를 Zoom out함
smartgun mode일때는 카메라 위치 고정

기능 구현

```
void main_menu(int option) {
    if (option == 99) exit(0); // exit
    else if (option == 1) { // init
        glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
        individual = 0; // individual
        color = 0; // color
        background = 0; // background
        // assemble variable
        assemble = 0; a = 0;
        // house party variable
        houseparty = 0;
        h[0] = 0; h[1] = 0; h[2] = 0; h[3] = 0; h[4] = 0;
        // smart gun variable
        smartgun = 0;
        radius = 11; theta = 80; phi = 1; center[1] = 0;
        heart[0] = 5; heart[1] = 5; heart[2] = 5;
        // repulsor variable
        repulsorbeam = 0; r = 0;
        printf("Init has been selected\n");
    }
    else if (option == 2) { // assemble
        if (assemble == 0) { // off -> on
            assemble = 1;
            PlaySound("sound/move.wav", 0, SND_FILENAME | SND_ASYNC);
        }
        else { // on -> off
            assemble = 0;
            a = 0;
        }
        printf("Assemble has been selected\n");
    }
    glutPostRedisplay();
}
```

```
void sub_menu1(int option) { // color
    if (option == 0) { // red
        color = 0;
        printf("Red has been selected\n");
    }
void sub_menu2(int option) { // background
    if (option == 20) { // universe
        background = 0;
        printf("Universe has been selected\n");
    }
void sub_menu3(int option) { // individual
    if (option == 30) { // face
        individual = 30;
        printf("Face has been selected\n");
    }
```

↑ sub_menu1, sub_menu2, sub_menu3 함수 :
각각 color, background, individual을 선택
*코드 일부분만 캡쳐

← main_menu 함수 :
선택된 menu의 option 값에 따라
init – 시작 상태로 초기화
exit – 종료
assemble – assemble mode의 on/off

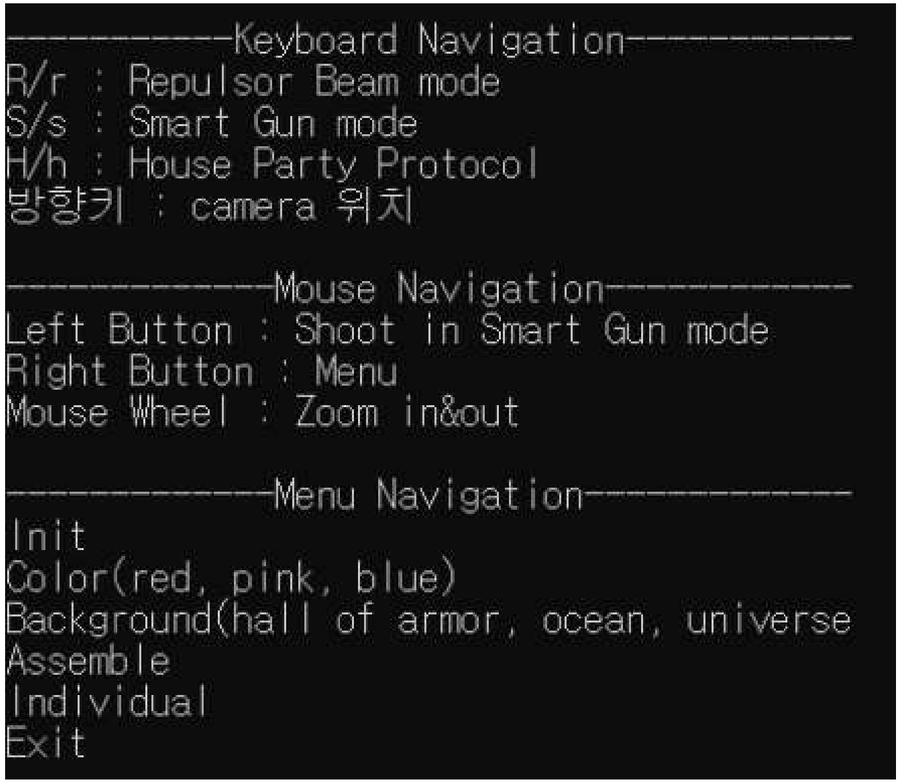
기능 구현

```
void printInstruction() {
    /* 조작법 console 출력 */
    printf("\n-----------Keyboard Navigation-----------\n");
    printf("R/r : Repulsor Beam mode\n");
    printf("S/s : Smart Gun mode\n");
    printf("H/h : House Party Protocol\n");
    printf("방향키 : camera 위치\n");

    printf("\n-------------Mouse Navigation------------\n");
    printf("Left Button : Shoot in Smart Gun mode\n");
    printf("Right Button : Menu\n");
    printf("Mouse Wheel : Zoom in&out\n");

    printf("\n-------------Menu Navigation-------------\n");
    printf("Init\n");
    printf("Color(red, pink, blue)\n");
    printf("Background(hall of armor, ocean, universe\n");
    printf("Assemble\n");
    printf("Individual\n");
    printf("Exit\n\n");
}
```

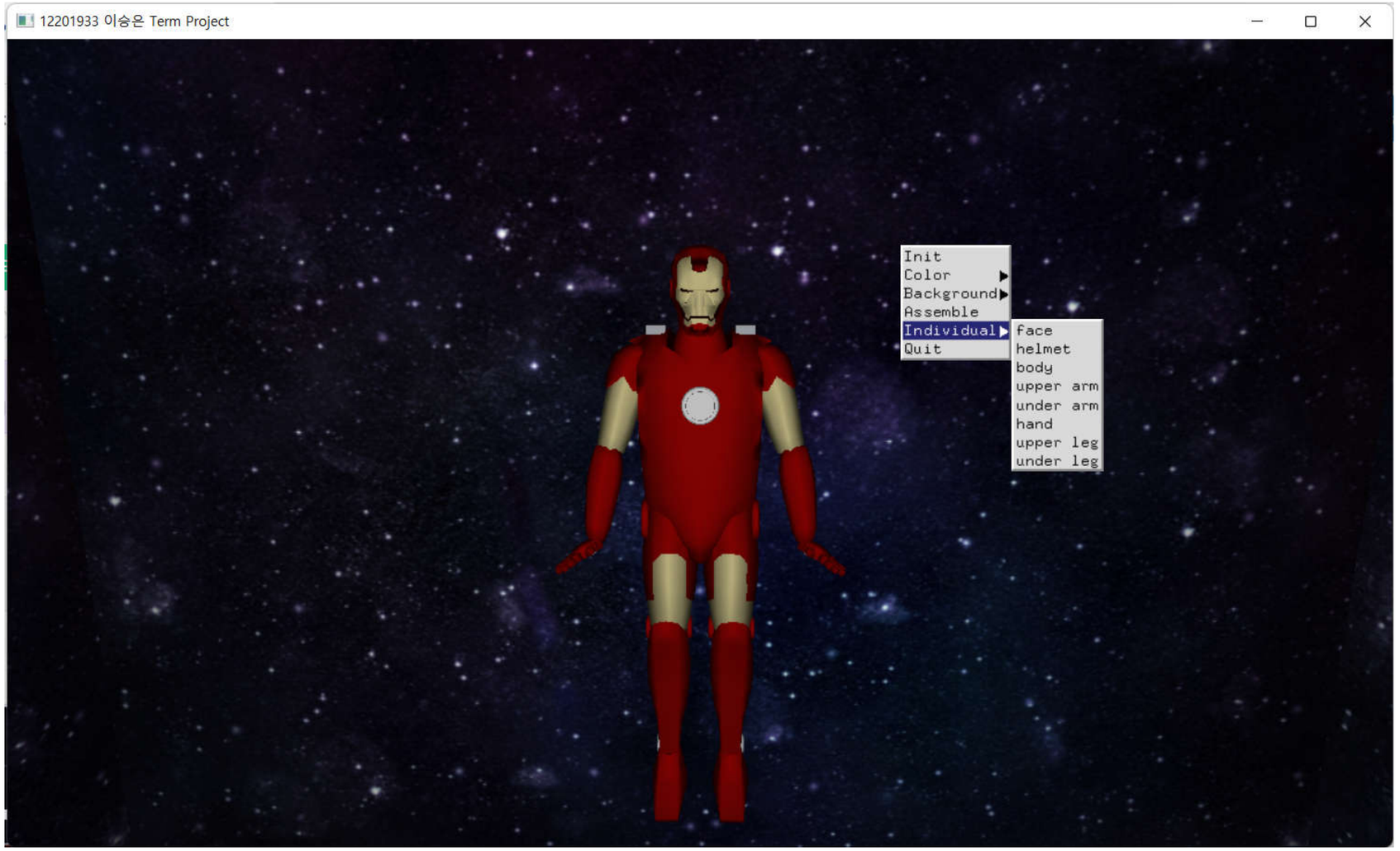
printInstruction 함수 :
조작법 console 출력, init 함수에서 호출



콘솔 실행 화면

실행 화면

시작 화면(red, universe) + 메뉴



실행 화면

pink, blue color

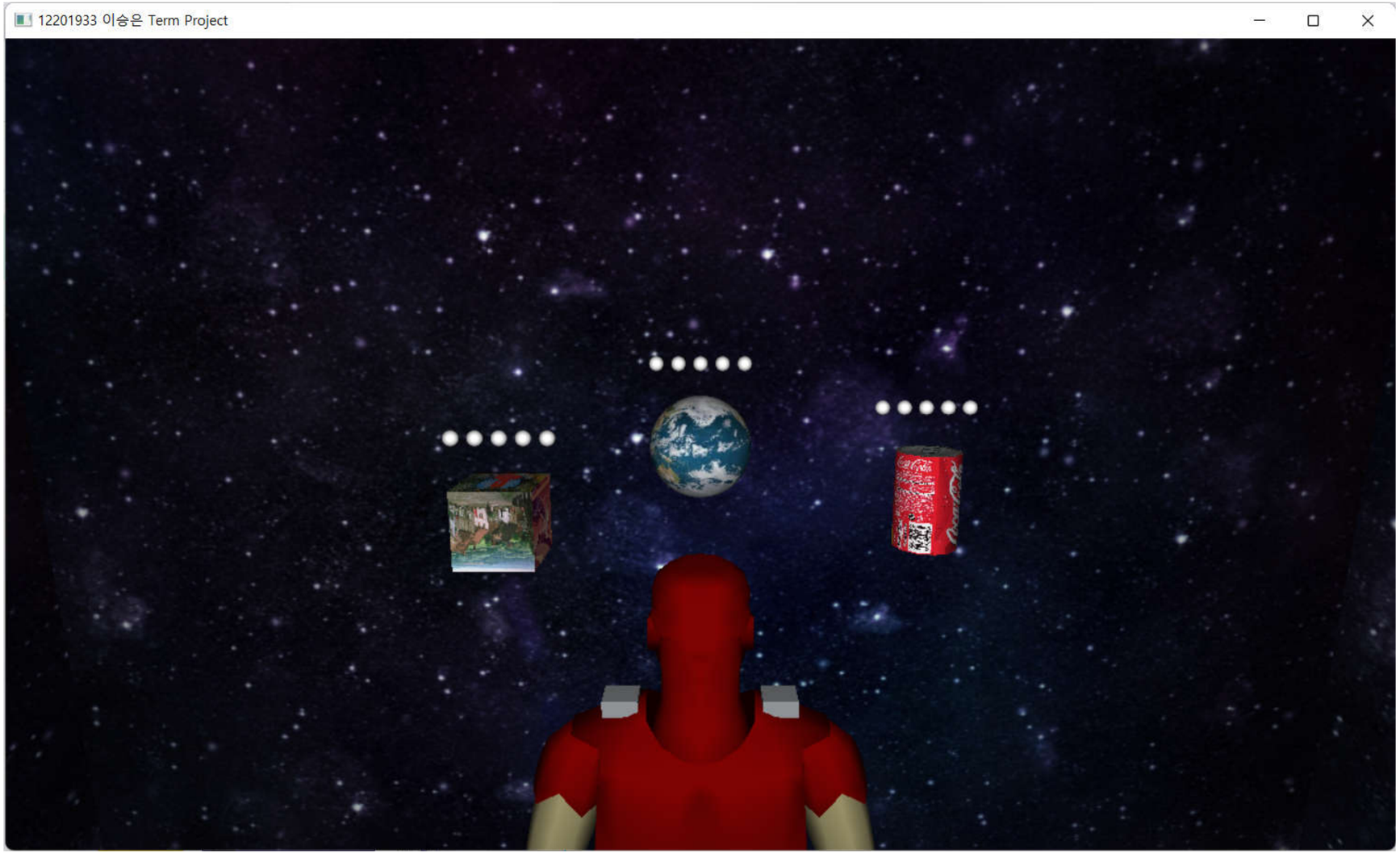
hall of armor, ocean background

실행 화면

assemble mode(애니메이션 도중 캡쳐)

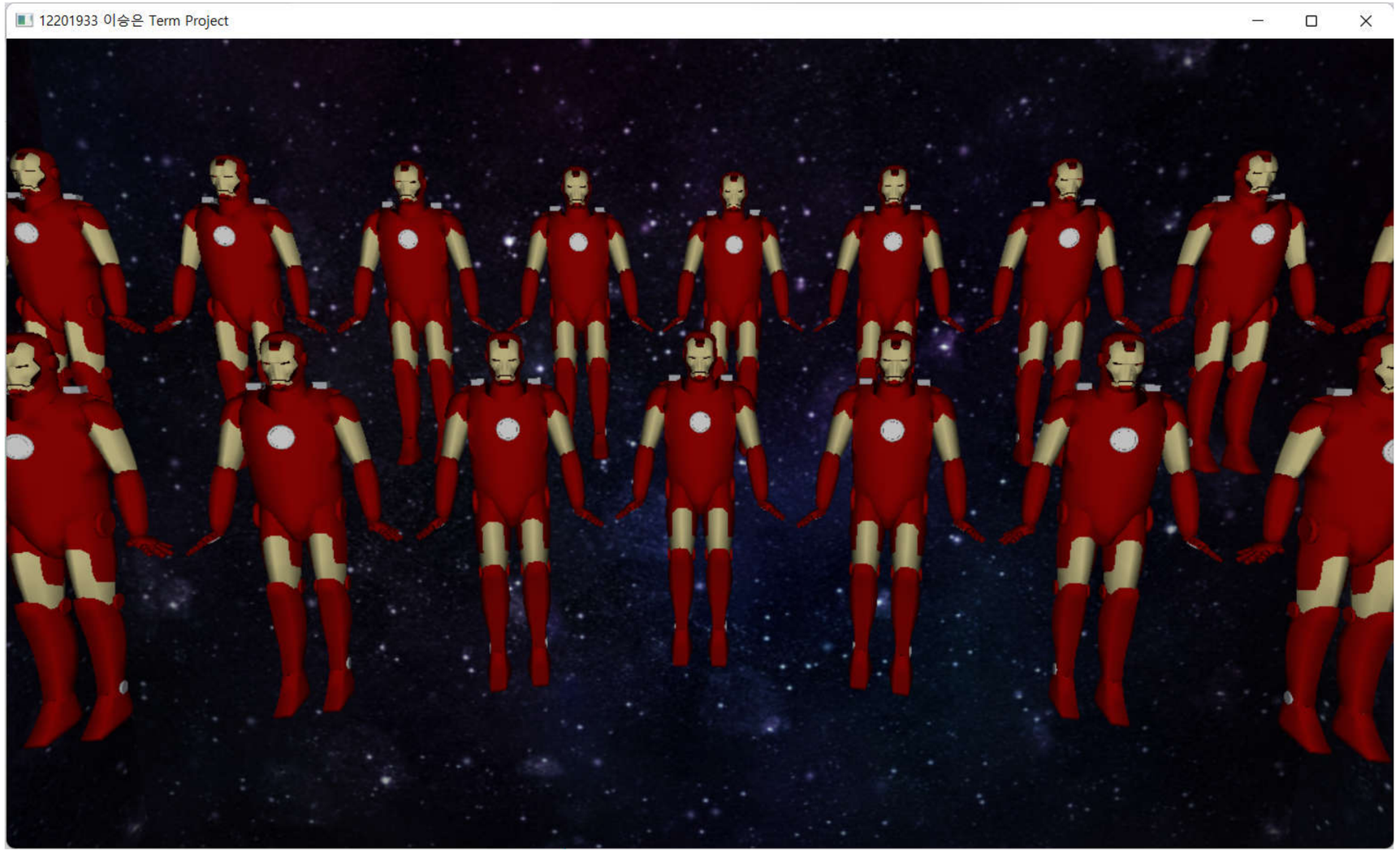
실행 화면

smartgun mode



실행 화면

housepartyProtocol mode(애니메이션 완료 후 캡쳐)



실행 화면

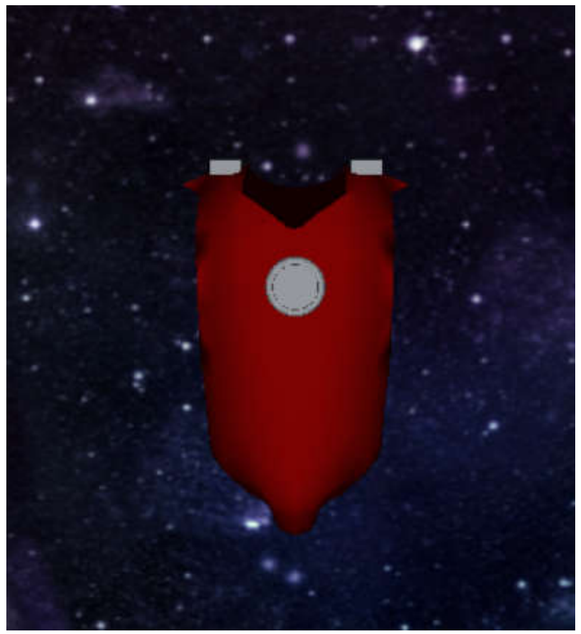
repulsorbeam mode(애니메이션 완료 후 캡쳐)

실행 화면

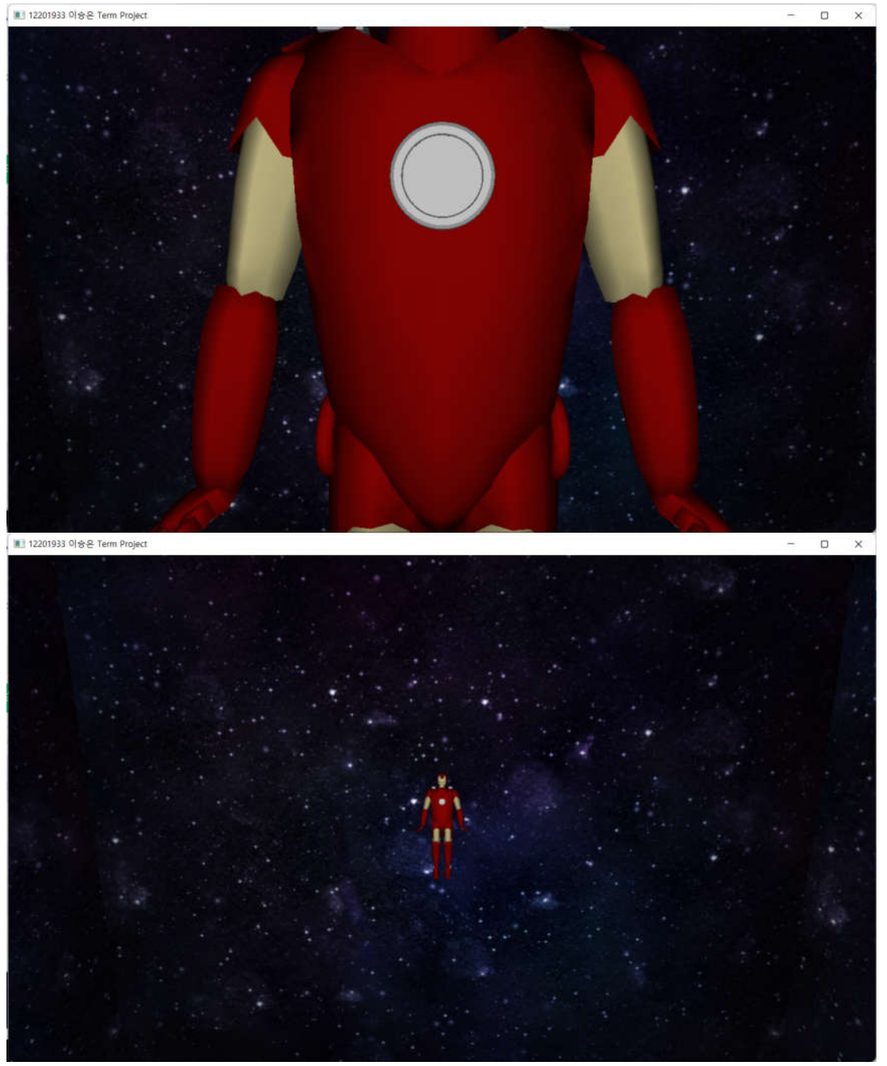
housepartyProtocol + repulsorbeam mode
(애니메이션 완료 후 캡쳐)

실행 화면

individual mode(일부 캡쳐)



Zoom in&out



실행 화면

카메라 위치 이동

감사합니다.