컴퓨터 그래픽스
미니 프로젝트

<유모차 제품 카탈로그>

12121696
전승주

목차

1.1) Modeling 개요

1.1.1)
기본적으로 Quadric object를 주로 사용하여 모델링하였습니다.
Quadric object는 local 좌표계의 z축을 중심으로
렌더링되므로 glTranslatef()와 glRotatef()를 이용하여
물체를 그리고자 하는 방향으로 축을 계속적으로 변경하면서 유모차의 형상을
모델링 하였습니다.

1.1.2)
미세하게 축을 변경하여 유모차의 곡면을 표현했지만 완벽한 곡면을 표현하는 것
은 어려웠습니다.
현재 축 변경에 사용되는 코드 양도 상당하며,
매끄럽지 못한 애니메이션 결과로 보아 컴파일 시 약간 무리가 가고 있다고
생각됩니다.

1.1.3)
코드 양이 길고 복잡하여 축의 흐름을 따라가지 못하면
작성자 이외에 이해하기 힘든 부분이 많다고 생각됩니다.

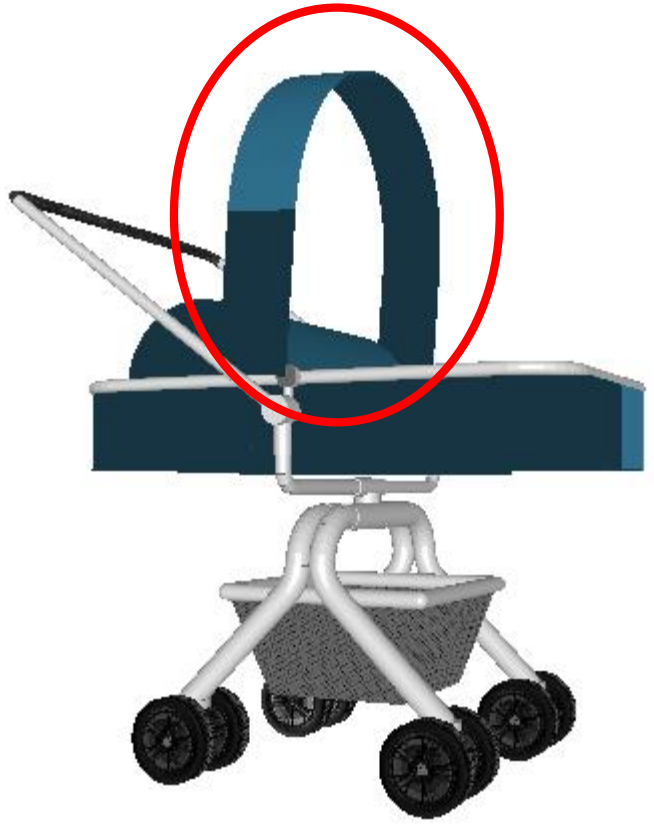
최종 모델링 형상

1.2) 캐노피 modeling

캐노피는 유모차의 그늘 가리개를 의미합니다.

캡처 사진의 빨간 원 표시된 부분은
직사각형 polygon과 quadric object인 cylinder로
이루어져 있습니다.
위에 반원통처럼 그려진 cylinder는 clipping plane
을 통해 절반을 잘라서 구현하였습니다.

이 부분을 for문을 통해 회전각도를 변경하며 반복
하여, 캐노피를 그리게 됩니다.



1.2) 캐노피 modeling

전역변수 :

```
// clipping plane
float cx = 0;
float cy = 1;
float cz = 0;
GLdouble eq[4] = { cx, cy, cz, 0 };    // clipping plane의 normal 벡터 = (cx, cy, cz)
```

draw() :

```
for (int i = 0; i<3; i++) {
    glPushMatrix();
        glTranslatef(0, 0, 0.7*i);
        glRotatef(10 * i + canopy_Angle0, 1, 0, 0);
        glBindTexture(GL_TEXTURE_2D, bodyColor);
        glBegin(GL_QUADS);
            glTexCoord2f(0, 0); glVertex3f(-16.25, 0, 0);
            glTexCoord2f(0, 1); glVertex3f(-16.25, 0, 7);
            glTexCoord2f(1, 1); glVertex3f(-16.25, 14.5, 7);
            glTexCoord2f(1, 0); glVertex3f(-16.25, 14.5, 0);
        glEnd();
        glBegin(GL_QUADS);
            glTexCoord2f(0, 0); glVertex3f(16.25, 0, 0);
            glTexCoord2f(0, 1); glVertex3f(16.25, 0, 7);
            glTexCoord2f(1, 1); glVertex3f(16.25, 14.5, 7);
            glTexCoord2f(1, 0); glVertex3f(16.25, 14.5, 0);
        glEnd();
        glTranslatef(0, 14.5, 0);
        glClipPlane(GL_CLIP_PLANE0, eq);
        glEnable(GL_CLIP_PLANE0);
        gluCylinder(m_pQuadric, 16.25, 16.25, 7, 100, 100);
        glDisable(GL_CLIP_PLANE0);
    glPopMatrix();
}
```

2. Texture Mapping

이번 프로젝트에 사용된 Texture Image는 크게
5가지로 구성됩니다.

2.1) 바퀴
2.2) 색상
2.3) 그물망
2.4) 구조 기둥
2.5) 손잡이

2.1) 바퀴

바퀴 Texture Image는 2가지로 구성됩니다.
- 바퀴 모양 (wheel.bmp)
- 타이어 (wheel_side.bmp)

Wheel.bmp : 256 X 256
Wheel_side.bmp : 512 X 256

Wheel.bmp

Wheel_side.bmp

2.1) 바퀴

전역변수 :

```
// Texture Mapping
AUX_RGBImageRec * pRGBImage;
GLuint wheel[2];
GLuint bodyColor;
GLuint mesh;
GLuint frame;
GLuint handle_texture;
```

2.1) 바퀴

Init() :

```
// texture mapping
glGenTextures(2, wheel);
glGenTextures(1, &bodyColor);
glGenTextures(1, &mesh);
glGenTextures(1, &frame);
glGenTextures(1, &handle_texture);

glBindTexture(GL_TEXTURE_2D, wheel[0]);
pRGBImage = auxDIBImageLoadA("wheel.bmp");
glTexImage2D( GL_TEXTURE_2D, 0, 3, pRGBImage->sizeX, pRGBImage->sizeY, 0, GL_RGB,
                GL_UNSIGNED_BYTE, pRGBImage->data);
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 );

glBindTexture(GL_TEXTURE_2D, wheel[1]);
pRGBImage = auxDIBImageLoadA("wheel_side.bmp");
glTexImage2D( GL_TEXTURE_2D, 0, 3, pRGBImage->sizeX, pRGBImage->sizeY, 0, GL_RGB,
                GL_UNSIGNED_BYTE, pRGBImage->data);
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 );
```

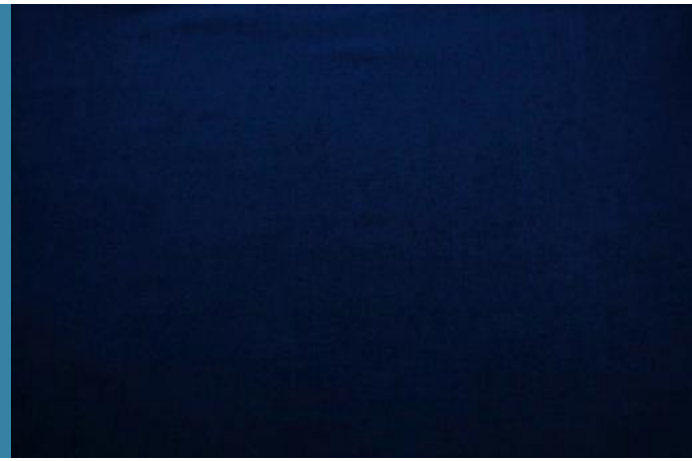
glGenTextures()를 이용하여 선언하였던 전역변수에 대하여 Texture name을 할당받고, Image파일을 불러와 성질들을 정의

2.2) 색상

색상 Texture Image는 4가지로 구성됩니다.
- Cyan (cyan.bmp)
- Dark blue (dark_blue.bmp)
- Purple (purple.bmp)
- Gold (gold)

cyan.bmp : 512 X 256
dark_blue.bmp : 512 X 256
purple.bmp : 512 X 256
gold.bmp : 512 X 256

Cyan.bmp



Dark blue.bmp

Purple.bmp

Gold.bmp

2.2) 색상

Init() :

```
// texture mapping
glGenTextures(2, wheel);
glGenTextures(1, &bodyColor);
glGenTextures(1, &mesh);
glGenTextures(1, &frame);
glGenTextures(1, &handle_texture);
```

glGenTextures()를 이용하여 선언하였던 전역변수에 대하여 Texture name을 할당받고, Image파일을 불러와 성질들을 정의.
초기 색상은 cyan이고, 메뉴 선택에 의해 색상 설정이 가능하다.

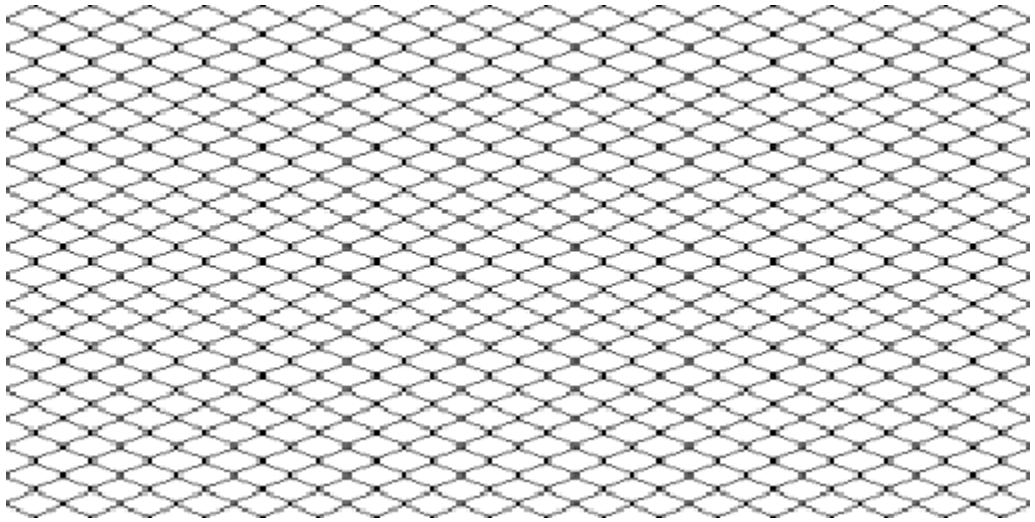
```
glBindTexture(GL_TEXTURE_2D, bodyColor);
pRGBImage = auxDIBImageLoadA("cyan.bmp");  // 초기 color
glTexImage2D( GL_TEXTURE_2D, 0, 3, pRGBImage->sizeX, pRGBImage->sizeY, 0, GL_RGB,
                GL_UNSIGNED_BYTE, pRGBImage->data);
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT );
glTexParameteri( 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 );
```

```
glTexEnvi( GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE );   // Polygon의 원래 색상은 무시하고 texture로 덮음
glEnable( GL_TEXTURE_2D );
```

2.3) 그물망

그물망 Texture Image는 유모차의 아래 부분 그물 바구니에 사용된다.
- mesh.bmp

mesh.bmp : 512 X 256



Mesh.bmp

2.3) 그물망

Init() :

```
// texture mapping
glGenTextures(2, wheel);
glGenTextures(1, &bodyColor);
glGenTextures(1, &mesh);
glGenTextures(1, &frame);
glGenTextures(1, &handle_texture);

glBindTexture(GL_TEXTURE_2D, mesh);
pRGBImage = auxDIBImageLoadA("mesh.bmp");
glTexImage2D( GL_TEXTURE_2D, 0, 3, pRGBImage->sizeX, pRGBImage->sizeY, 0, GL_RGB,
                GL_UNSIGNED_BYTE, pRGBImage->data);
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 );
```

glGenTextures()를 이용하여 선언하였던 전역변수에 대하여 Texture name을 할당받고, Image파일을 불러와 성질들을 정의.

2.4) 구조 기둥

구조 기둥 Texture Image는 유모차의 구조에 해당하는
기둥에 사용된다.
철제 이미지도 texture mapping해보았으나
디자인이 깔끔해 보이지 않아서 흰 색을 사용하였다.
- frame.bmp

frame.bmp : 512 X 256

frame.bmp

2.4) 구조 기둥

Init() :

```
// texture mapping
glGenTextures(2, wheel);
glGenTextures(1, &bodyColor);
glGenTextures(1, &mesh);
glGenTextures(1, &frame);
glGenTextures(1, &handle_texture);

glBindTexture(GL_TEXTURE_2D, frame);
pRGBImage = auxDIBImageLoadA("frame.bmp");
glTexImage2D( GL_TEXTURE_2D, 0, 3, pRGBImage->sizeX, pRGBImage->sizeY, 0, GL_RGB,
                GL_UNSIGNED_BYTE, pRGBImage->data);
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 );
```

glGenTextures()를 이용하여 선언하였던 전역변수에 대하여 Texture name을 할당받고, Image파일을 불러와 성질들을 정의.

2.5) 손잡이

손잡이 Texture Image는 손잡이 부분의 가죽 재질을 표현하기 위해 사용됩니다.

- handle.bmp

handle.bmp : 512 X 256

handle.bmp

2.5) 손잡이

Init() :

```
// texture mapping
glGenTextures(2, wheel);
glGenTextures(1, &bodyColor);
glGenTextures(1, &mesh);
glGenTextures(1, &frame);
glGenTextures(1, &handle_texture);

glBindTexture(GL_TEXTURE_2D, handle_texture);
pRGBImage = auxDIBImageLoadA("handle.bmp");  // 초기 color
glTexImage2D(GL_TEXTURE_2D, 0, 3, pRGBImage->sizeX, pRGBImage->sizeY, 0, GL_RGB,
    GL_UNSIGNED_BYTE, pRGBImage->data);
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);


glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);   // Polygon의 원래 색상은 무시하고 texture로 덮음
glEnable(GL_TEXTURE_2D);
```

glGenTextures()를 이용하여 선언하였던 전역변수에 대하여 Texture name을 할당받고, Image파일을 불러와 성질들을 정의.

3. Interaction

이번 프로젝트에서 구현한 Interaction은 10가지 입니다.
3.1) 3.2) 3.3) 3.4) 3.6) 3.8) 3.9) 마우스 클릭을 통한 menu 선택
3.5) 손잡이 up : 'w'　손잡이 down : 's'
3.7) 캐노피 up : 'e'　캐노피 down : 'd'
3.10) 키보드 방향키와 zoom in : 'z' , zoom out : 'x'

3.1) 바퀴 회전
3.2) 몸체 회전
3.3) 등받이 각도 조절
3.4) 손잡이 길이 조절
3.5) 손잡이 각도 조절
3.6) 캐노피 조절
3.7) 캐노피 각도 조절
3.8) 색상 설정
3.9) 모드 설정
3.10) 시점 변화 및 Zoom-in and out

3.1) 바퀴(wheel) 회전

전역변수 :

```
// 바퀴 회전
boolean wheelRotation_On = false;
int wheelRotation_Angle = 0;
```

Idle() :

```
if(wheelRotation_On)
    wheelRotation_Angle += 1;
if(wheelRotation_Angle >= 360)
    wheelRotation_Angle = 0;
```

Main_menu _function :

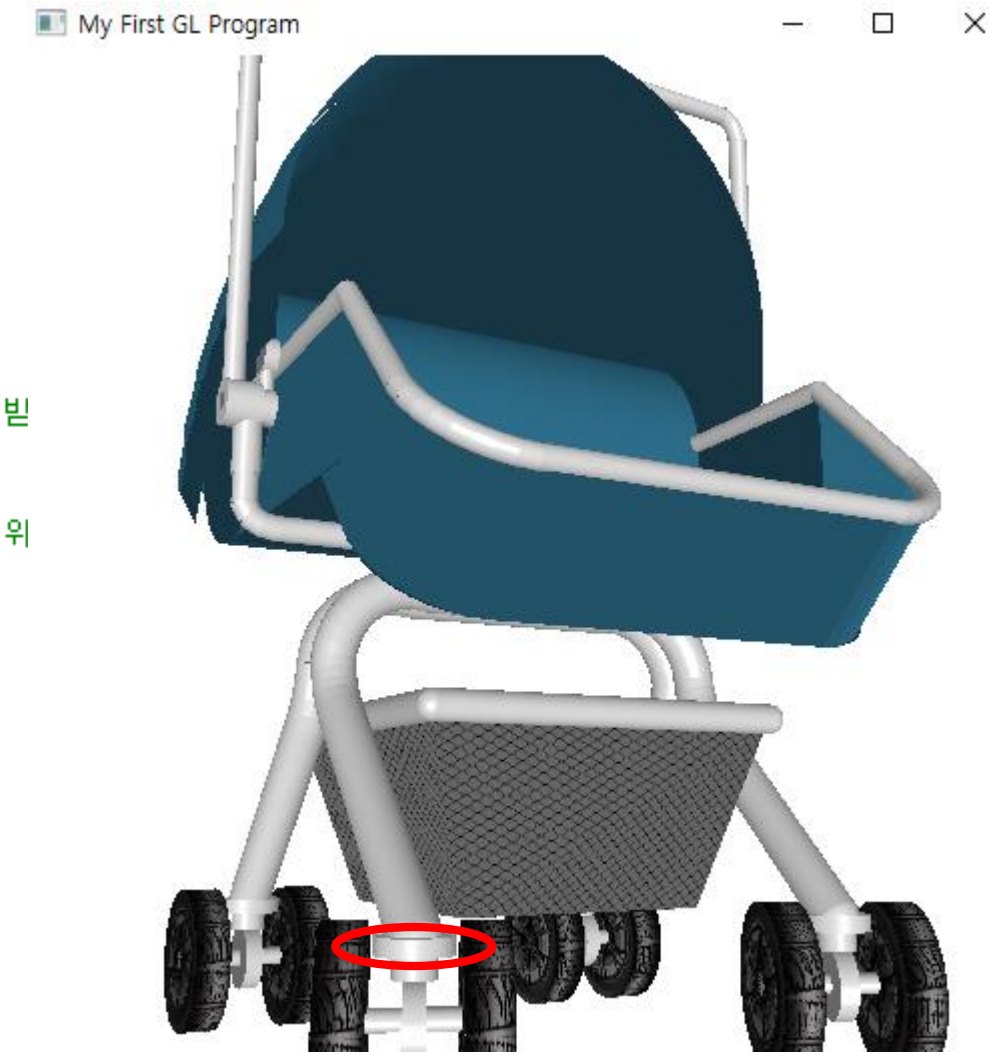
```
void main_menu_function(int option)
{
    printf("Main menu %d has been selected ₩n", option);
    if(option == 999)
        exit(0);

    switch(option){
        // 바퀴 회전
        case 1: if(!(wheelRotation_On))
                    wheelRotation_On = true;
                else
                    wheelRotation_On = false;
                break;
        // 몸체 회전
        case 2: if(!(bodyRotation_On))
                    bodyRotation_On = true;
                else
                    bodyRotation_On = false;
                break;
    }
}
```

Main menu에서 바퀴 회전이 선택되면 wheelRotation_On이 true가 되어 idle()에서 wheelRotation_Angle값이 증가하게 된다. 360도가 되면 0이 되도록 설정하였다.

3.1) 바퀴(wheel) 회전

Draw() :

```
// 바퀴와 기둥 연결 부위
glRotatef(45, 1, 0, 0);
glPushMatrix();
// 바퀴 회전 기능
    glRotatef(wheelRotation_Angle, 0, 0, 1);
    gluDisk(m_pQuadric, 0, 1.8, 100, 100);
    gluCylinder(m_pQuadric, 1.8, 1.8, 0.8, 100, 100);  // 작은 빈
    glTranslatef(0, 0, 0.8);
    gluDisk(m_pQuadric, 0, 1.8, 100, 100);
    gluCylinder(m_pQuadric, 1, 1, 1, 100, 100);  // 작은 연결부위
```



Draw()의 바퀴와 기둥을 연결하는 부분에서
glRotatef()를 호출하여 바퀴가 회전하도록 하였다.
그 부위는 옆 캡처의 빨간색 표시이다.
빨갛게 표시한 원판이 회전하면 그 아래의 부품과 바퀴는 이 원판의 좌표계에 종속되어 있어서 함께 회전하게 된다.

3.2) 몸체(body) 회전

전역변수 :

```
// 몸체 회전
boolean bodyRotation_On = false;
int bodyRotation_Angle = 0;
```

Main_menu _function :

```
void main_menu_function(int option)
{
    printf("Main menu %d has been selected \n", option);
    if(option == 999)
        exit(0);

    switch(option){
        // 바퀴 회전
        case 1: if(!(wheelRotation_On))
                    wheelRotation_On = true;
                else
                    wheelRotation_On = false;
                break;
        // 몸체 회전
        case 2: if(!(bodyRotation_On))
                    bodyRotation_On = true;
                else
                    bodyRotation_On = false;
                break;
    }
}
```

Idle() :

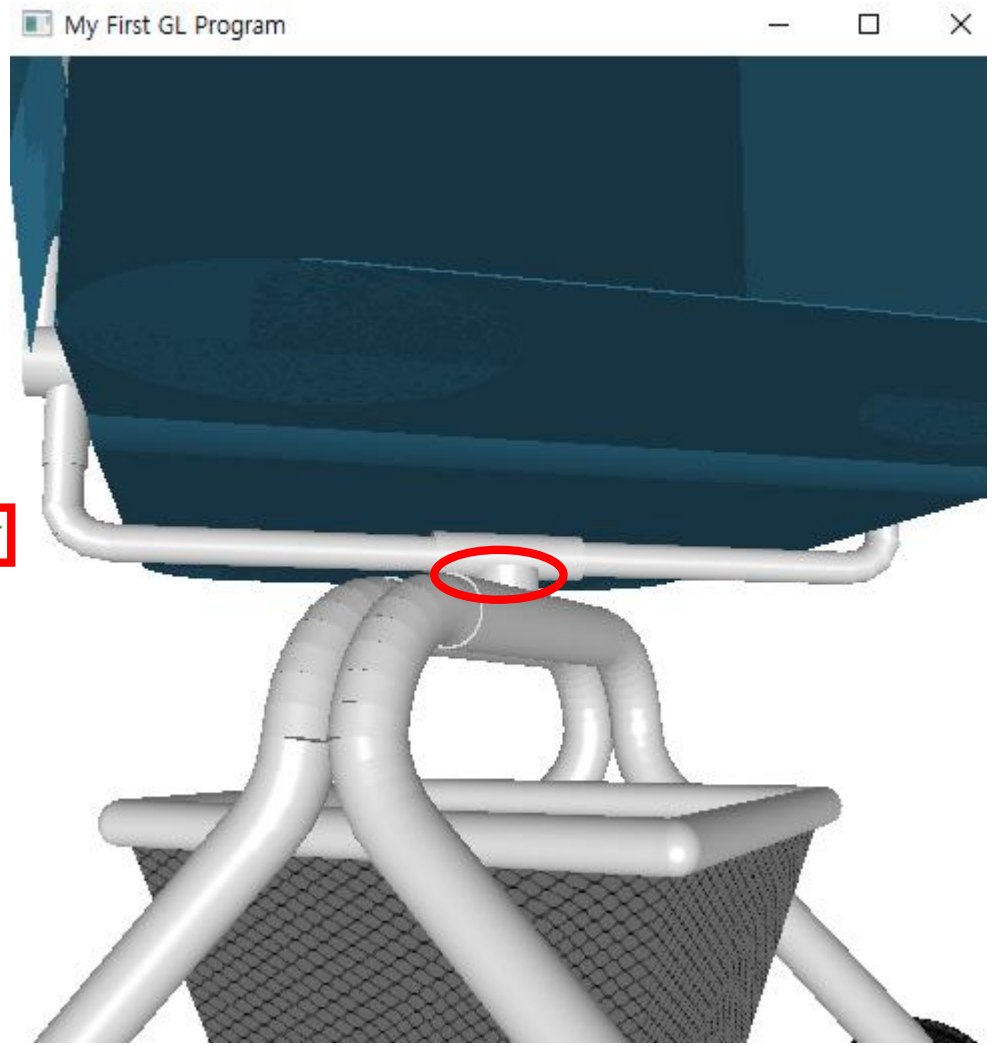
```
if(bodyRotation_On)
    bodyRotation_Angle += 2;
if(bodyRotation_Angle >= 360)
    bodyRotation_Angle = 0;
```

Main menu에서 몸체 회전이 선택되면 bodyRotation_On이 true가 되어 idle()에서 bodyRotation_Angle값이 증가하게 된다. 360도가 되면 0이 되도록 설정하였다.

3.2) 몸체(body) 회전

Draw() :

```
glPushMatrix();  // 몸체 회전을 위한 glPushMatrix()
glRotatef(-bodyRotation_Angle, 0, 1, 0);   /*** 몸체 회전 ***/
// ************ 몸체 및 손잡이 ************
// 오른쪽 (x축 방향)
glPushMatrix();
    glRotatef(90, 0, 1, 0);
    glPopMatrix();
glPopMatrix();   // ************ 몸체 및 손잡이 부분 끝 ************

glPopMatrix();  // 몸체 회전을 위한 glPopMatrix();  draw()의 시작 부분에 glPushMatr

// ***************** 하체 부분 *****************

// 바구니
glPushMatrix();
    glPushMatrix();   // 바구니 그리기
```



Draw()의 시작 부분에서 glPushMatrix()와 glPopMatrix()로 상체와 하체를 구분하였다.
따라서 상체 부분 전체를 회전시키면 몸체가 회전하게 된다.

실제로 캡처 사진의 빨간 원에 표시된 디스크가 몸체의 회전을 담당하는 부품이다. 저 부품이 회전하면서 이 부품에 종속된 몸체는 회전하게 된다.

3.3) 등받이 각도 조절

전역변수 :

```
// 등받이 각도 조절. 0단계 ~ 4단계
boolean back_Mode[5] = {true, false, false, false, false};
int back_Angle = 0;
```

Idle() :

```
back_Mode_func( );
```

3.3) 등받이 각도 조절

sub_menu_function :

```
void sub_menu_function(int option)
{
    printf("submenu %d has been selected \n", option);

    switch(option){
        case 0: back_Mode[0] = true;
                back_Mode[1] = false;
                back_Mode[2] = false;
                back_Mode[3] = false;
                back_Mode[4] = false;
                break;
        case 1: back_Mode[0] = false;
                back_Mode[1] = true;
                back_Mode[2] = false;
                back_Mode[3] = false;
                back_Mode[4] = false;
                break;
        case 2: back_Mode[0] = false;
                back_Mode[1] = false;
                back_Mode[2] = true;
                back_Mode[3] = false;
                back_Mode[4] = false;
                break;
        case 3: back_Mode[0] = false;
                back_Mode[1] = false;
                back_Mode[2] = false;
                back_Mode[3] = true;
                back_Mode[4] = false;
                break;
        case 4: back_Mode[0] = false;
                back_Mode[1] = false;
                back_Mode[2] = false;
                back_Mode[3] = false;
                back_Mode[4] = true;
                break;
    }
```

Sub menu에서 등받이 각도 단계가 선택되면 해당 case에 맞게 back_Mode를 활성화/비활성화 시킨다.

3.3) 등받이 각도 조절

Back_Mode_func :

```
void back_Mode_func( )
{
    for(int i=0; i<5; i++){
        if(back_Mode[i]){
            switch(i){
                case 0: if(back_Angle == 0){}
                        else{ back_Angle -= 1; }
                        break;
                case 1: if(back_Angle < 5){ back_Angle += 1; }
                        else if(back_Angle == 5){}
                        else if(back_Angle > 5){ back_Angle -= 1; }
                        break;
                case 2: if(back_Angle < 10){ back_Angle += 1; }
                        else if(back_Angle == 10){}
                        else if(back_Angle > 10){ back_Angle -= 1; }
                        break;
                case 3: if(back_Angle < 15){ back_Angle += 1; }
                        else if(back_Angle == 15){}
                        else if(back_Angle > 15){ back_Angle -= 1; }
                        break;
                case 4: if(back_Angle < 20){ back_Angle += 1; }
                        else if(back_Angle == 20){}
                        else if(back_Angle > 20){ back_Angle -= 1; }
                        break;
            }
        }
    }
}
```

Idle()에서 호출되고,
등받이 각도 조절을 처리하는 함수이다.

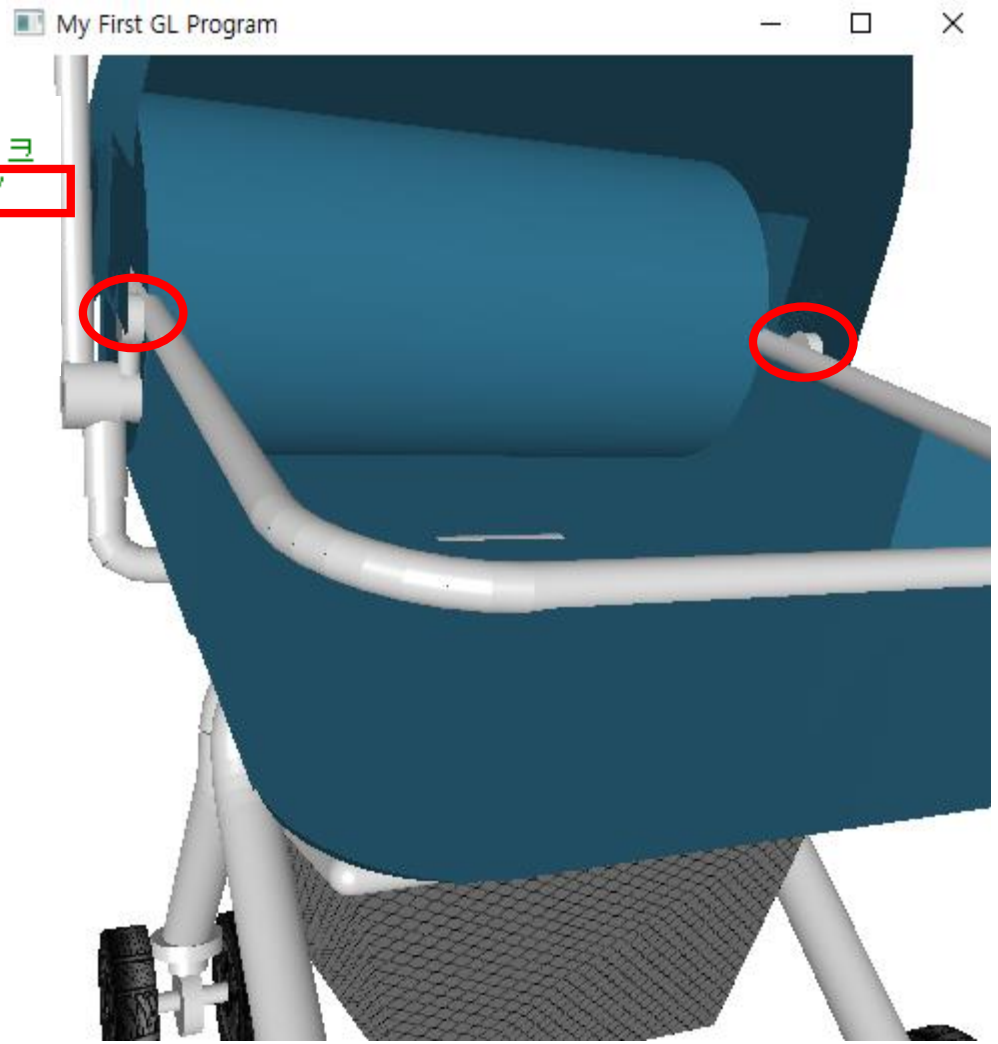
For문을 통해 해당하는 등받이 각도 단계를 찾고
각 단계에서 각도에 대한 처리를 해준다.

각 단계마다 5도씩 증가하며, 해당 단계에서의
각도에서 멈춰있도록 구현하였다.
예를 들어, 2단계가 선택되면 등받이 각도는
10도가 되도록 설정되는 식이다.

3.3) 등받이 각도 조절

Draw() :

```
glPushMatrix();  // 본체 오른쪽 뼈대
    glRotatef(180, 0, 1, 0);  // 본체 뼈대 받침 기둥 아래쪽 디스크
    glRotatef(-back_Angle, 0, 0, 1);  /*** 등받이 각도 조절 ***/
    gluCylinder(m_pQuadric, 1.0, 1.0, 0.8, 100, 100);
    glTranslatef(0, 0, 0.8);
    gluDisk(m_pQuadric, 0, 1.0, 100, 100);
    glPushMatrix();  // 본체 뼈대 받침 기둥
```



캡처 사진의 빨갛게 표시한 디스크가 몸체의 뼈대를 고정하는 부품이다.
이 부품을 회전시킴으로써 등받이 각도를 조절할 수 있다.

3.4) 손잡이 길이 조절

전역변수 :

```
// 손잡이 길이 조절. 1단 ~ 3단
#define HANDLE_SIZE  5
boolean handle[3] = {true, false, false};
int handle_2 = 0;    // 2단
int handle_3 = 0;    // 3단.   기본은 1단.
```

Idle() :

```
handle_func( );
```

3.4) 손잡이 길이 조절

sub_menu_function2 :

```
void sub_menu_function2(int option)
{
    printf("submenu2 %d has been selected ₩n", option);

    switch(option){
        case 1: handle[0] = true;
                handle[1] = false;
                handle[2] = false;
                break;
        case 2: handle[0] = false;
                handle[1] = true;
                handle[2] = false;
                break;
        case 3: handle[0] = false;
                handle[1] = false;
                handle[2] = true;
                break;
    };
}
```

Sub menu에서 손잡이 길이 단계가 선택되면 해당 case에 맞게 handle[]을 활성화/비활성화 시킨다.

3.4) 손잡이 길이 조절

handle_func :

```
void handle_func()
{
    for(int i=0; i<3; i++){
        if(handle[i]){
            switch(i+1)
            {
                case 1: if(0 < handle_3 && handle_3 <= HANDLE_SIZE){ handle_3 -= 1; }
                        else if(handle_3 == 0)
                        {
                            if(0 < handle_2 && handle_2 <= HANDLE_SIZE){ handle_2 -= 1; }
                            else if(handle_2 == 0) {}
                        }
                        break;
                case 2: if(0 < handle_3 && handle_3 <= HANDLE_SIZE){ handle_3 -= 1; }
                        else if(handle_3 == 0)
                        {
                            if(0 <= handle_2 && handle_2 < HANDLE_SIZE){ handle_2 += 1; }
                            else if(handle_2 == HANDLE_SIZE) {}
                        }
                        break;
                case 3: if(0 <= handle_2 && handle_2 < HANDLE_SIZE){ handle_2 += 1; }
                        else if(handle_2 == HANDLE_SIZE)
                        {
                            if(0 <= handle_3 && handle_3 < HANDLE_SIZE){ handle_3 += 1; }
                            else if(handle_3 == HANDLE_SIZE) {}
                        }
                        break;
            };
        }
    }
}
```

Idle()에서 호출되고,
손잡이 길이 조절을 처리하는 함수이다.

For문을 통해 해당하는 손잡이 길이 단계를 찾고
각 단계에서 각도에 대한 처리를 해준다.

각 case는 두 가지 경우에 대해서 처리한 것이다.
예를 들어, case 1은 1단계가 선택되었으므로
현재 2단계 -> 1단계 또는 현재 3단계 -> 1단계인 경우
이다. 이 두 가지 경우를 고려하여
전역변수 handle_2 or handle_3 값을 변화시켜
손잡이 길이가 조절되도록 하였다.

3.4) 손잡이 길이 조절

Draw() :

```
gluCylinder(m_pQuadric, 0.6, 0.6, 25, 100, 100);    // 손잡이 기둥1
glTranslatef(0, 0, 25-HANDLE_SIZE);
glTranslatef(0, 0, handle_2);
gluCylinder(m_pQuadric, 0.61, 0.61, HANDLE_SIZE, 100, 100);    // 손잡이 기둥2
glTranslatef(0, 0, handle_3);
gluCylinder(m_pQuadric, 0.62, 0.62, HANDLE_SIZE, 100, 100);    // 손잡이 기둥3
// 손잡이
glTranslatef(0, 0, HANDLE_SIZE-0.5);
glRotatef(10, -1, 0, 0);
```

기본적으로 손잡이 기둥1에 의해 표현되고
손잡이 기둥2와 기둥3은 겹쳐 있다가
해당 단계가 설정되면 glTranslatef()에 의해 손잡이 길이가 조절된다.

3.4) 손잡이 길이 조절

1단계(기본)

2단계

3단계

3.5) 손잡이 각도 조절

전역변수 :

```
// 손잡이 각도 조절
int handle_Angle = 0;
```

keyboard() :

```
if (key == 'w') {    // handle up
    if (handle_Angle == -90) {}
    else { handle_Angle -= 1; }
}
if (key == 's') {    // handle down
    if (handle_Angle == 90) {}
    else { handle_Angle += 1; }
}
```

각도가 90도 이상 넘어가지 않도록 구현했습니다.

3.5) 손잡이 각도 조절

draw() :

```
// 손잡이 회전 원통 부분
glPushMatrix();
    glRotatef(-handle_Angle, 0, 0, 1);    /*** 손잡이 각도 조절 ***/
    gluDisk(m_pQuadric, 0.5, 1.3, 100, 100);
    gluCylinder(m_pQuadric, 1.3, 1.3, 1, 100, 100);
    glTranslatef(0, 0, 1);
    gluDisk(m_pQuadric, 0.5, 1.3, 100, 100);
    // 손잡이 기둥
```

캡처 사진의 빨간 원 표시의 회전 원통 부품이 회전하면서 손잡이 각도가 조절되게 됩니다.

3.6) 캐노피 조절

전역변수 :

```
// 캐노피 조절 . 0단계 ~ 3단계
boolean canopy[4] = { true, false, false, false };
int canopy_Angle0 = 0;
int canopy_Angle1 = 0;
int canopy_Angle2 = 0;
```

Idle() :

```
canopy_Angle_idle( );
```

3.6) 캐노피 조절

sub_menu
_function :

```
void sub_menu_function3(int option)
{
    printf("submenu3 %d has been selected ₩n", option);

    switch (option) {
    case 0: canopy[0] = true;
        canopy[1] = false;
        canopy[2] = false;
        canopy[3] = false;
        break;
    case 1: canopy[0] = false;
        canopy[1] = true;
        canopy[2] = false;
        canopy[3] = false;
        break;
    case 2: canopy[0] = false;
        canopy[1] = false;
        canopy[2] = true;
        canopy[3] = false;
        break;
    case 3: canopy[0] = false;
        canopy[1] = false;
        canopy[2] = false;
        canopy[3] = true;
        break;
    };
}
```

Sub menu에서 캐노피 단계가 선택되면 해당 case에 맞게 canopy[]를 활성화/비활성화 시킨다.

3.6) 캐노피 조절

canopy_Angle_idle() : Idle()에서 호출되고,
캐노피 단계 조절을 위한 각도를 처리하는 함수이다.

for문을 통해 해당하는 캐노피 단계를 찾고
각 단계에서 각도에 대한 처리를 해준다.

0단계:
canopy_Angle0 = 0
canopy_Angle1 = 0
canopy_Angle2 = 0

1단계:
canopy_Angle0 = 30
canopy_Angle1 = 0
canopy_Angle2 = 0`

2단계:
canopy_Angle0 = 60
canopy_Angle1 = 30
canopy_Angle2 = 0

3단계:
canopy_Angle0 = 90
canopy_Angle1 = 60
canopy_Angle2 = 30

3.6) 캐노피 조절

canopy_Angle_idle() :

```
void canopy_Angle_idle( )
{
    for (int i = 0; i<4; i++) {
        if (canopy[i]) {
            switch (i)
            {
            case 0: if (canopy_Angle0 == 0) {}
                else if (0 < canopy_Angle0) { canopy_Angle0 -= 1; }

                if (canopy_Angle1 == 0) {}
                else if (0 < canopy_Angle1) { canopy_Angle1 -= 1; }

                if (canopy_Angle2 == 0) {}
                else if (0 < canopy_Angle2 && canopy_Angle2 <= 30) { canopy_Angle2 -= 1; }
                break;

            case 1: if (canopy_Angle0 == 30) {}
                else if (0 <= canopy_Angle0 && canopy_Angle0 < 30) { canopy_Angle0 += 1; }
                else if (30 < canopy_Angle0 && canopy_Angle0 <= 90) { canopy_Angle0 -= 1; }

                if (canopy_Angle1 == 0) {}
                else if (0 < canopy_Angle1 && canopy_Angle1 <= 60) { canopy_Angle1 -= 1; }

                if (canopy_Angle2 == 0) {}
                else if (0 < canopy_Angle2 && canopy_Angle2 <= 30) { canopy_Angle2 -= 1; }
                break;

            case 2: if (canopy_Angle0 == 60) {}
                else if (canopy_Angle0 < 60) { canopy_Angle0 += 1; }
                else if (canopy_Angle0 > 60) { canopy_Angle0 -= 1; }

                if (canopy_Angle1 == 30) {}
                else if (canopy_Angle1 < 30) { canopy_Angle1 += 1; }
                else if (canopy_Angle1 > 30) { canopy_Angle1 -= 1; }

                if (canopy_Angle2 == 0) {}
                else if (canopy_Angle2 > 0) { canopy_Angle2 -= 1; }
                break;

            case 3: if (canopy_Angle0 == 90) {}
                else if (canopy_Angle0 < 90) { canopy_Angle0 += 1; }

                if (canopy_Angle1 == 60) {}
                else if (canopy_Angle1 < 60) { canopy_Angle1 += 1; }

                if (canopy_Angle2 == 30) {}
                else if (canopy_Angle2 < 30) { canopy_Angle2 += 1; }
                break;
            };
```

3.6) 캐노피 조절

draw() :

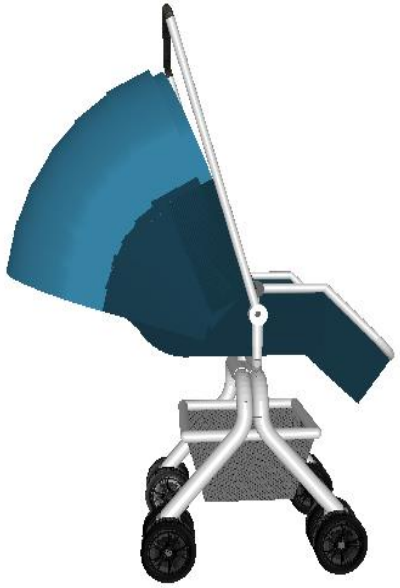
```
for (int i = 0; i<3; i++) {
    glPushMatrix();
        glTranslatef(0, 0, 0.7*i);
        glRotatef(10 * i + canopy_Angle0, 1, 0, 0);
        glBindTexture(GL_TEXTURE_2D, bodyColor);
        glBegin(GL_QUADS);
            glTexCoord2f(0, 0); glVertex3f(-16.25, 0, 0);
            glTexCoord2f(0, 1); glVertex3f(-16.25, 0, 7);
            glTexCoord2f(1, 1); glVertex3f(-16.25, 14.5, 7);
            glTexCoord2f(1, 0); glVertex3f(-16.25, 14.5, 0);
        glEnd();
        glBegin(GL_QUADS);
            glTexCoord2f(0, 0); glVertex3f(16.25, 0, 0);
            glTexCoord2f(0, 1); glVertex3f(16.25, 0, 7);
            glTexCoord2f(1, 1); glVertex3f(16.25, 14.5, 7);
            glTexCoord2f(1, 0); glVertex3f(16.25, 14.5, 0);
        glEnd();
        glTranslatef(0, 14.5, 0);
        glClipPlane(GL_CLIP_PLANE0, eq);
        glEnable(GL_CLIP_PLANE0);
        gluCylinder(m_pQuadric, 16.25, 16.25, 7, 100, 100);
        glDisable(GL_CLIP_PLANE0);
    glPopMatrix();
}
```

위 for문이 3번 반복되며 캐노피를 그리게 되는데,
이 때 3개의 for문 각각에 glRotatef()에 canopy_Angle을 더하여준다.

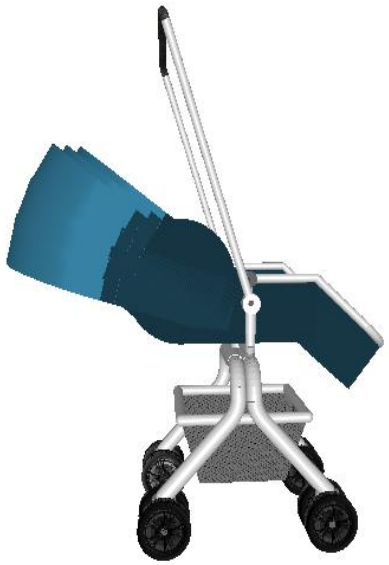
3.6) 캐노피 조절

0단계(기본)

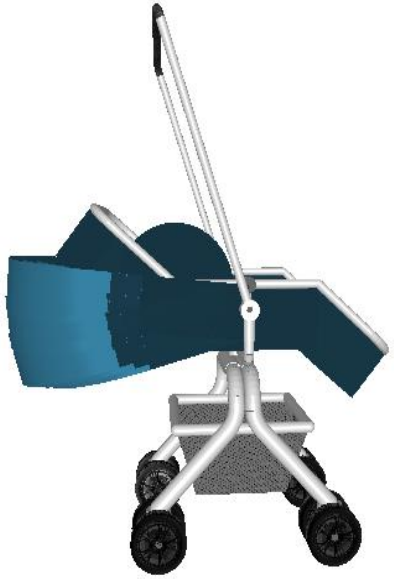
1단계



2단계



3단계



3.7) 캐노피 각도 조절

전역변수 :
```
// 캐노피 각도 조절.
int canopy_Angle3 = 0;
```

keyboard() :
```
if (key == 'e') {    // canopy up
    if (canopy_Angle3 == -30) {}
    else { canopy_Angle3 -= 1; }
}
if (key == 'd') {    // canopy down
    if (canopy_Angle3 == 30) {}
    else { canopy_Angle3 += 1; }
}
```

각도가 30도 이상 넘어가지 않도록 구현했습니다.

3.7) 캐노피 각도 조절

draw() :

```
glRotatef(-5 + canopy_Angle3, 1, 0, 0);    /*** 캐노피 각도 조절 ***/
for (int i = 0; i<3; i++) {
    glPushMatrix();
        glTranslatef(0, 0, 0.7*i);
        glRotatef(10 * i + canopy_Angle0, 1, 0, 0);
        glBindTexture(GL_TEXTURE_2D, bodyColor);
```

캐노피를 그리는 3개의 for문에 들어가기 전에
glRotatef()를 통해 전체 캐노피 각도를
조절하도록 했습니다.

3.8) 색상 설정

Sub_menu_
function3 :

```
void sub_menu_function3(int option)
{
    printf("submenu3 %d has been selected \n", option);

    switch(option){
        case 1: glBindTexture(GL_TEXTURE_2D, bodyColor);
                pRGBImage = auxDIBImageLoadA("cyan.bmp");
                glTexImage2D( GL_TEXTURE_2D, 0, 3, pRGBImage->sizeX, pRGBImage->sizeY, 0, GL_RGB,
                              GL_UNSIGNED_BYTE, pRGBImage->data);
                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 );
                break;
        case 2: glBindTexture(GL_TEXTURE_2D, bodyColor);
                pRGBImage = auxDIBImageLoadA("dark_blue.bmp");
                glTexImage2D( GL_TEXTURE_2D, 0, 3, pRGBImage->sizeX, pRGBImage->sizeY, 0, GL_RGB,
                              GL_UNSIGNED_BYTE, pRGBImage->data);
                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 );
                break;
        case 3: glBindTexture(GL_TEXTURE_2D, bodyColor);
                pRGBImage = auxDIBImageLoadA("purple.bmp");
                glTexImage2D( GL_TEXTURE_2D, 0, 3, pRGBImage->sizeX, pRGBImage->sizeY, 0, GL_RGB,
                              GL_UNSIGNED_BYTE, pRGBImage->data);
                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 );
                break;
        case 4: glBindTexture(GL_TEXTURE_2D, bodyColor);
                pRGBImage = auxDIBImageLoadA("gold.bmp");
                glTexImage2D( GL_TEXTURE_2D, 0, 3, pRGBImage->sizeX, pRGBImage->sizeY, 0, GL_RGB,
                              GL_UNSIGNED_BYTE, pRGBImage->data);
                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 );
                break;

    };
}
```

Init()에서 해주던 설정을
sub menu function에서 처리함으로써
색상 설정이 가능하다.

3.9) 모드 설정

전역변수 :

```
// 유람 모드와 유모차 모드
boolean cradle_Mode = true;     // 유람 모드
boolean stoller_Mode = false;   // 유모차 모드
int mode_Angle = 0;
```

Idle() :

```
mode_Angle_idle( );
```

모드 설정이란
유람 모드(cradle) 와
유모차 모드(stoller)를 말한다.

유람 모드에서 아기가 편안하게 잠들 수 있도록 몸체 부분이 수평상태가 되는 모드이다.

유모차 모드는 일반적인 유모차 모형을 유지하는 모드이다.

3.9) 모드 설정

유람 모드

유모차 모드

3.9) 모드 설정

sub_menu_function4 :

```
void sub_menu_function4(int option)
{
    printf("submenu4 %d has been selected ₩n", option);

    switch(option){
        case 1: cradle_Mode = true;
                stoller_Mode = false;
                break;
        case 2: cradle_Mode = false;
                stoller_Mode = true;
                break;
    };
}
```

Sub menu에서 특정 모드가 선택되면 해당 case에 맞게 전역변수를 활성화/비활성화 시킨다.

3.9) 모드 설정

Mode_Angle_
idle :

```
void mode_Angle_idle( )
{
    if(cradle_Mode)
    {
        if(0 < mode_Angle && mode_Angle <= 45)
            mode_Angle -= 1;
        else if(mode_Angle == 0) {}
    }
    else if(stoller_Mode)
    {
        if(0 <= mode_Angle && mode_Angle < 45)
            mode_Angle += 1;
        else if(mode_Angle == 45) {}
    }
}
```

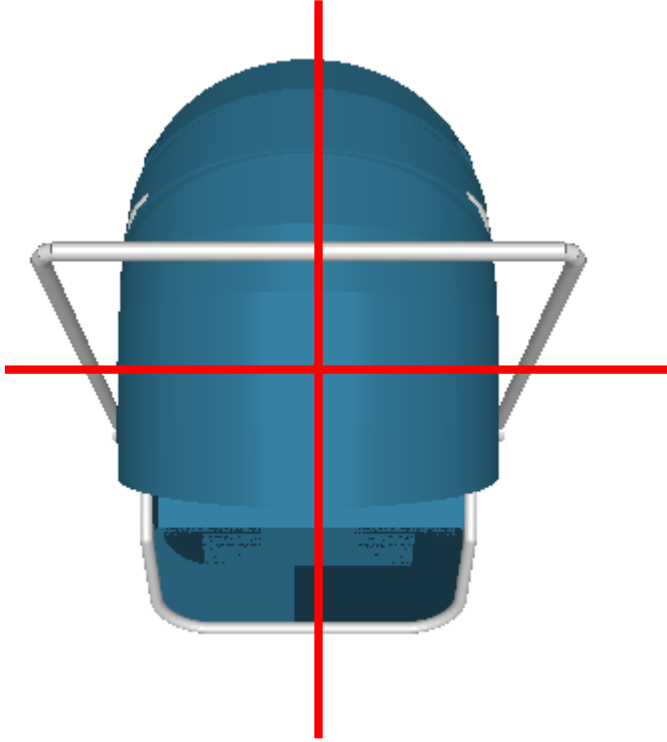
Idle()에서 호출되고,
모드 설정 시 몸체의 각도 조절을 처리하는 함수이다.

유람 모드에서는 mode_Angle이 0이 되도록
유모차 모드에서는 mode_Angle이 45도가 되도록
구현되었다.

3.9) 모드 설정

Draw() :

```
glPushMatrix(); /*** 유모차 모드 : 오른쪽 뒤 ***/
    glRotatef(-mode_Angle, 1, 0, 0);
    glBindTexture(GL_TEXTURE_2D, bodyColor);
    glRotatef(-90, 0, 1, 0);
glPushMatrix(); /*** 유모차 모드 : 오른쪽 앞 ***/
    glRotatef(mode_Angle, 1, 0, 0);
    gluCylinder(m_pQuadric, 0.6, 0.6, 10, 100, 100);
    glPushMatrix();  // 오른쪽 앞 뼈대 곡선 부분.  90도 변경을 위해 9번 시행
        glTranslatef(0, 0, 9.9);
glPushMatrix();  // /*** 유모차 모드 : 왼쪽 앞 ***/
    glTranslatef(0, 0, 10);
    gluSphere(m_pQuadric, 0.6, 100, 100);
    glRotatef(mode_Angle, 1, 0, 0);
    gluCylinder(m_pQuadric, 0.6, 0.6, 10, 100, 100);
glPushMatrix();  // /*** 유모차 모드 : 왼쪽 뒤 ***/
    glRotatef(-mode_Angle, 1, 0, 0);
    gluCylinder(m_pQuadric, 0.6, 0.6, 10, 100, 100);
    glPushMatrix();
```



유모차의 모델링은 크게 상체와 하체로 나뉘고
상체는 다시 위 사진처럼 4등분하여 오른쪽/왼쪽, 앞/뒤로 구분하여 구현되었다.
따라서 유모차의 등과 다리 부분의 몸체가 구부러 지려면 각 부위에 대한 각도 조절이 필요하다.
따라서 위와 같이 4부분에서 glRotatef()를 호출하여 각도를 조절하고 있다.
뒤 쪽에서는 위 방향으로 45도 구부러져 등받이 역할을 하게되고,
앞 쪽에서는 아래 방향으로 45도 구부러져 다리 받침대 역할을 하게 될 것이다.

3.10) 카메라 시점 변화와 Zoom in and out

special keyboard :

```
void specialkeyboard(int key, int x, int y)
{
    switch(key){
        case GLUT_KEY_RIGHT:
            azimuth+=4;
            break;

        case GLUT_KEY_LEFT:
            azimuth-=4;
            break;

        case GLUT_KEY_UP:
            elevation+=4;
            break;

        case GLUT_KEY_DOWN:
            elevation-=4;
            break;
    }

    if(elevation >= 360)         // 주기 360
        elevation -= 360;
    else if(elevation <= -360)
        elevation += 360;

    if(azimuth >= 360)         // 주기 360
        azimuth -= 360;
    else if(azimuth <= -360)
        azimuth += 360;
```

specialkeyboard()에서 방향키 입력에 따라 Elevation과 azimuth값을 조절한다.

Draw()에서 아래와 같이 up벡터를 실시간으로 갱신하여 카메라의 반전이 없도록 한다.

draw() :

```
upy = distance / cos(elevation * PI / 180);
gluLookAt(eyex, eyey, eyez, 0, 0, 0, -eyex, upy-eyey, -eyez);
```

```
eyex = distance * cos(elevation * PI / 180) * sin(azimuth * PI / 180);
eyey = distance * sin(elevation * PI / 180);
eyez = distance * cos(elevation * PI / 180) * cos(azimuth * PI / 180);
}
```

3.10) 카메라 시점 변화와 Zoom in and out

keyboard :

```
void keyboard(unsigned char key, int x, int y)
{
    printf("You pressed %c ₩n", key);

    if(key == 'z'){    // Zoom-in
        if(distance >= 0)
            distance -= 3;
    }
    if(key == 'x')    // Zoom-out
        distance += 3;

    if(elevation >= 360)          // 주기 360
        elevation -= 360;
    else if(elevation <= -360)
        elevation += 360;

    if(azimuth >= 360)          // 주기 360
        azimuth -= 360;
    else if(azimuth <= -360)
        azimuth += 360;


    eyex = distance * cos(elevation * PI / 180) * sin(azimuth * PI / 180);
    eyey = distance * sin(elevation * PI / 180);
    eyez = distance * cos(elevation * PI / 180) * cos(azimuth * PI / 180);
}
```

Keyboard()에서
'z'가 입력되면 distance 값을 감소시키고
'x'가 입력되면 distance 값을 증가시킨다