Digital 3D Anthropometry

5. 3D View Class

Sungmin Kim

SEOUL NATIONAL UNIVERSITY

Introduction

❖ 3D View Class 의 설계

- **3차원 그래픽의 개요**
  - ✓ Surface graphics
  - ✓ Volume graphics
  - ✓ Lighting and shading
- **TOpenGL**
  - ✓ 3차원 모델을 2차원 화면에 표시하는 클래스
    - – Rendering Context 만들기
    - – 장면 그리기
    - – 시점 변환

Computer Graphics

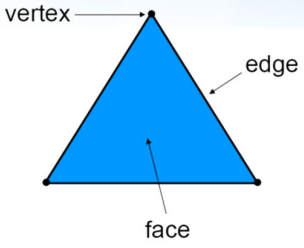
❖ Surface Graphics

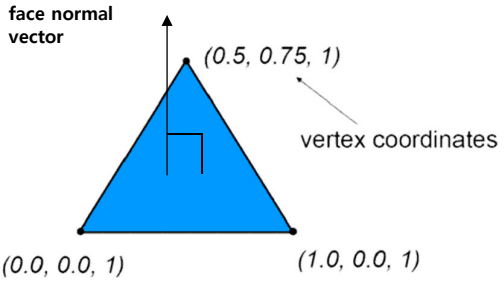
- Polygonal Mesh 를 이용해서 물체의 표면만을 그리는 방법
- 장면의 복잡성에 따라 처리속도가 달라짐
- 하드웨어 가속이 용이함
- 대부분의 그래픽 어플리케이션에 사용됨

Computer Graphics

❖ Surface Graphics

- 메쉬 (Mesh) 구조 모델링
  - ✓ 주로 삼각형 mesh 가 사용됨
    - vertex, edge, face 로 구성

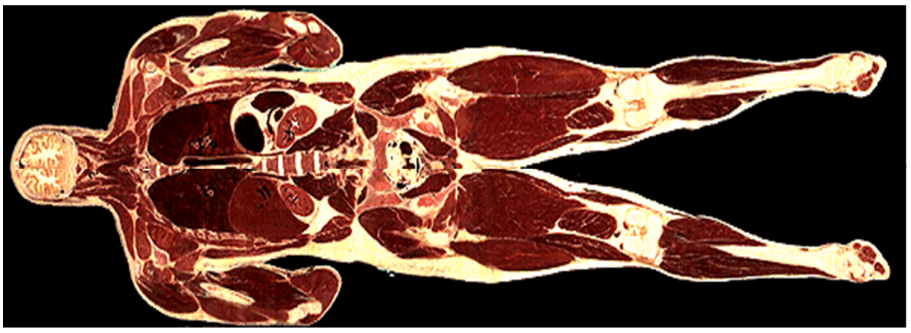


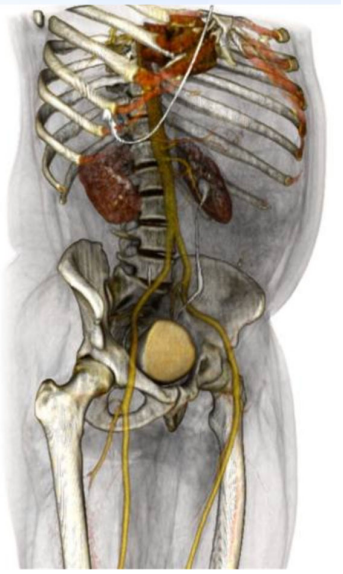


Computer Graphics

❖ Volume Graphics

- **복셀 (Voxel, Volume Element) 기반의 모델링**
  - ✓ 장면의 복잡도와 관계없이 일정한 처리 속도를 가짐
  - ✓ 확립된 하드웨어 가속 방법이 없음
    - – Compositing 프로세스의 복잡성에 기인
  - ✓ 의학 분야 등에 사용

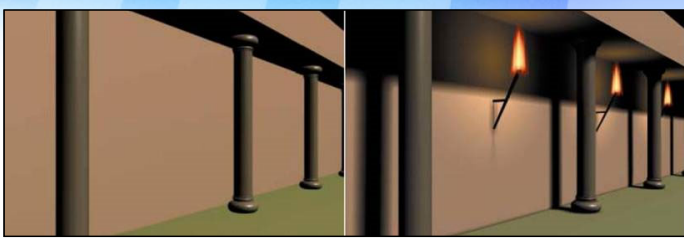


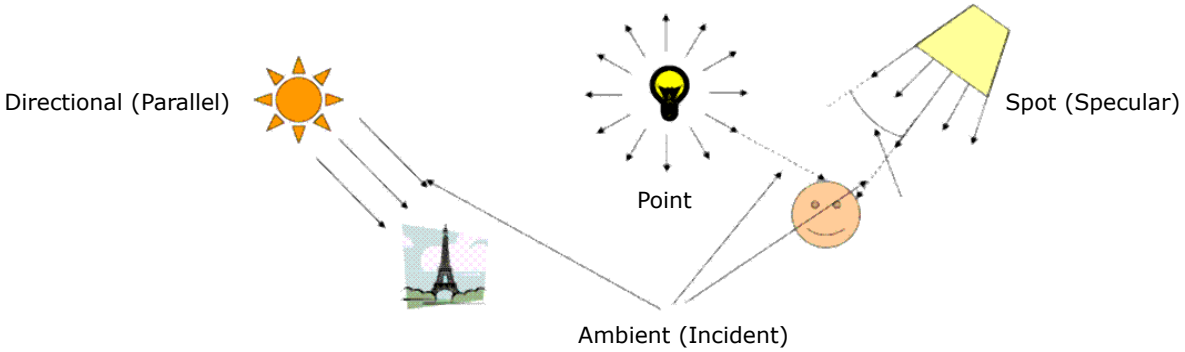


Lighting

❖ 조명

- **중요성**
  - ✓ 장면의 사실성을 높여줌
- **어려움**
  - ✓ 다양한 광원의 효과
    - – 일광, 불, 전구 등
  - ✓ 물체의 표면 성질과 색상의 효과
    - – 반짝이는 표면, 거친 표면 등





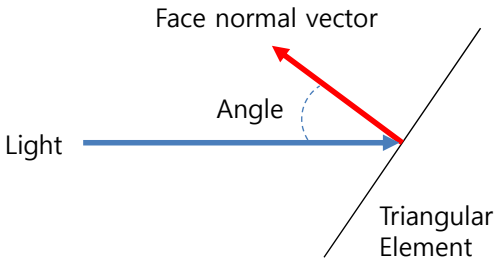
Shading

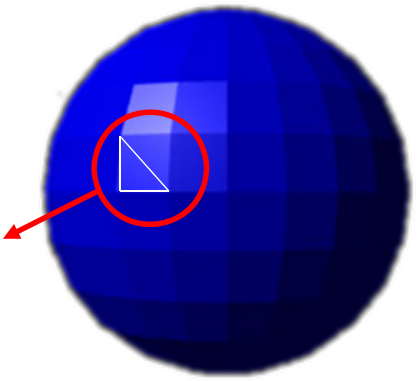
❖ 정의

- 빛의 반사를 고려한 색상의 계산
  - ✓ 물체의 mesh 의 각 점에 대해 계산
  - ✓ Mesh 내부의 점들에 대해 interpolation

❖ 종류

- **Flat shading**
- **Gouraud shading**
- **Phong shading**



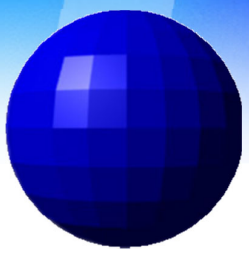


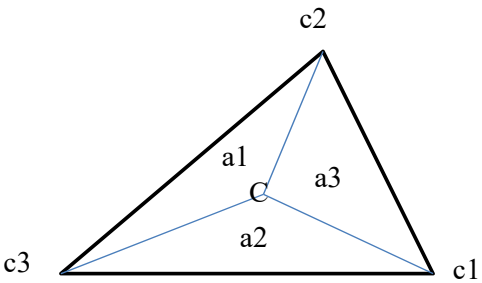
Flat Shading

Shading

❖ Gouraud Shading

- Mesh 의 각 꼭지점에 대해 계산
- Barycentric Coordinate 를 써서 interpolation





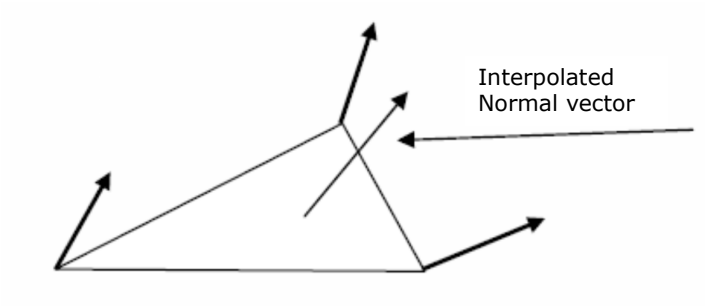
$$C = \frac{c1 \times a1 + c2 \times a2 + c3 \times a3}{a1 + a2 + a3}$$

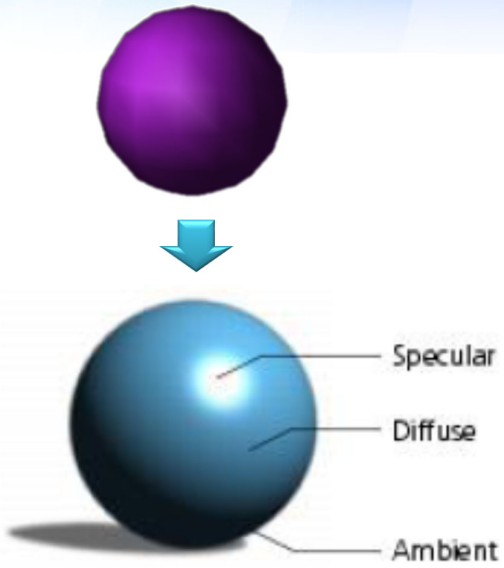
Barycentric Coordinate System

Shading

❖ Phong Shading

- Mesh 꼭지점의 normal vector 를 interpolation
- Interpolated normal 을 이용한 shading
- Specular Light 에 의한 highlight 이 구현됨





Rendering

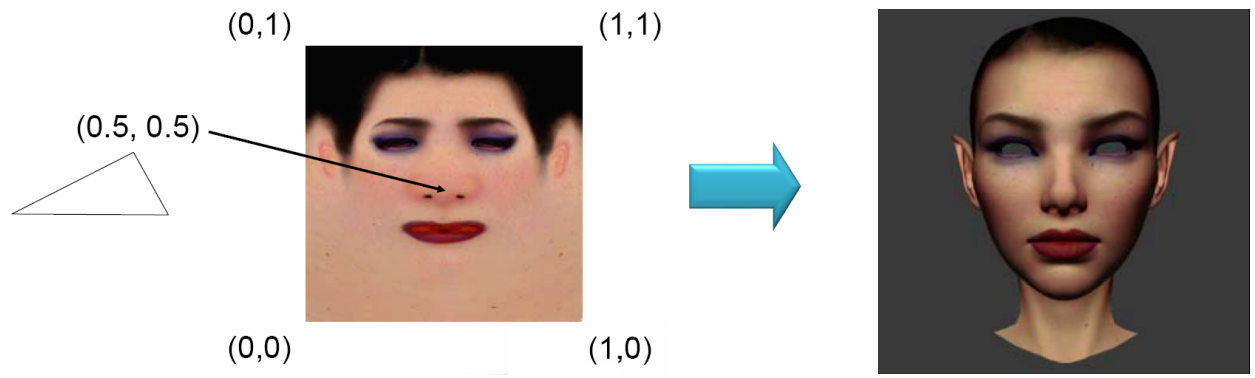
❖ Global Illumination

- 광학 현상을 충실히 묘사한 렌더링으로 매우 사실적인 표현이 가능함
  - ✓ 반사, 굴절, 투명도, 그림자 등의 표현을 위한 엄청난 계산이 필요함
  - ✓ 1968년에 최초의 Ray Tracing 알고리즘이 개발 되었으며 2018년에 실시간 Ray Tracing 구현됨

Texture Mapping

❖ 텍스쳐 매핑

- **중요성**
  - ✓ 물체의 복잡도를 높이지 않고 사실성을 향상할 수 있는 방법
- **방법**
  - ✓ 물체의 형상에 맞는 2D 이미지를 생성
  - ✓ 메쉬의 꼭지점에 텍스쳐 좌표를 설정



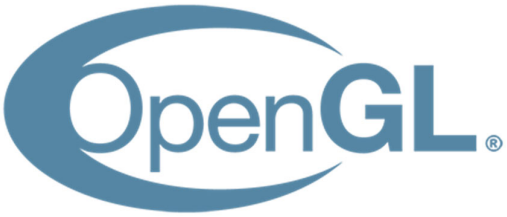
OpenGL

❖ OpenGL (Open Graphics Library) 의 역사

- **1992년 실리콘 그래픽스사에서 만든 2차원 및 3차원 그래픽스 표준 규격**
  - ✓ 1980년대에 다양한 그래픽 하드웨어에 맞는 소프트웨어를 개발하는 일은 매우 어려웠음
    - – 소프트웨어 개발자들이 개별 하드웨어에 맞추어 맞춤식 인터페이스와 드라이버를 작성
  - ✓ 실리콘 그래픽스 (SGI) IrisGL API (Application Programming Interface) 를 오픈소스로 전환
    - – 프로그래밍 언어 간, 플랫폼 간의 교차 응용 프로그래밍을 지원
  - ✓ 마이크로소프트사의 Direct3D와 함께 컴퓨터 그래픽 세계를 양분
    - – 표준안이 여러 관련 업체의 토론과 제안으로 이루어지기에 버전 업데이트는 느린 편이다
    - – 비영리 기술 컨소시엄인 크로노스 그룹에 의하여 관리되고 있다

❖ OpenGL (Open Graphics Library) 의 역사

- **약 250여개의 API 함수 호출을 이용**
  - ✓ 단순한 기하도형에서부터 복잡한 삼차원 장면까지 생성 가능
  - ✓ 게임, CAD, 가상현실, 정보시각화, 비행 시뮬레이션 등의 분야에서 활용
- **특징**
  - ✓ 서로 다른 하드웨어에 대해 단일한 API를 제공
  - ✓ 하드웨어 플랫폼의 능력 차이를 보완함



❖ OpenGL 의 확장

GLSL	OpenGL의 상위 레벨 셰이딩 언어
GLUT	윈도 시스템에 독립적인 OpenGL 프로그램을 작성하도록 도와주는 도구
SDL	Simple DirectMedia Layer
GLU	OpenGL 프로그램을 위한 추가적인 함수를 제공
GLee	OpenGL 프로그램을 위한 단순한 추가 라이브러리 제공
GLEW	OpenGL 확장 Wrangler 라이브러리 제공
GLUI	GLUT로 만들어진 GUI 툴킷으로 버튼, 체크박스 등의 GUI 기능을 제공
GLFW	OpenGL 응용 프로그램 개발을 위한 이식 가능한 프레임워크
GLM	GLSL 규격에 기반한 OpenGL을 위한 C++ 수학 툴킷
SFML	간단하고 빠른 멀티미디어 라이브러리
GLUX	OpenGL 유틸리티 및 보조 라이브러리

❖ C++ Builder에서 OpenGL 의 작동 순서

화면버퍼와 같은 속성의 Rendering Context 작성	OnCreate, OnResize
장면 시점 설정 등 그리기 준비 작업	OnPaint
Rendering Context 에 장면을 그리기	OnPaint
Rendering Context 와 화면버퍼를 서로 바꾸기	OnPaint
Rendering Context 를 삭제	OnClose

❖ OpenGL의 기초

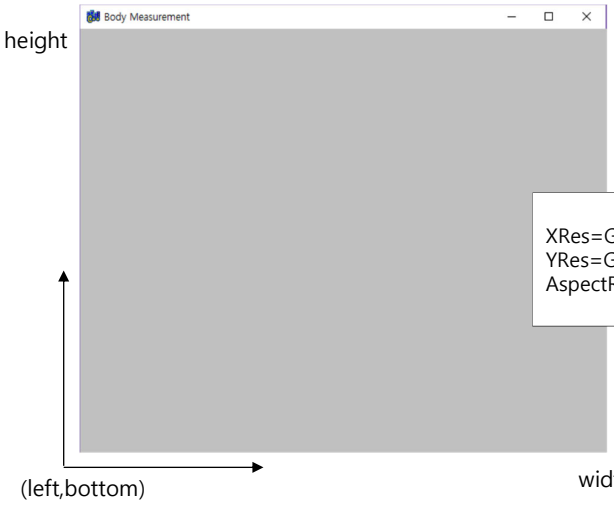
- **Rendering Context 만들기**
  - ✓ 화면과 호환되는 Pixel Format 의 선택 및 이를 바탕으로 한 Rendering Context 설정

```
DWORD dwFlags=PFD_DRAW_TO_WINDOW | PFD_SUPPORT_OPENGL | PFD_DOUBLEBUFFER;
PIXELFORMATDESCRIPTOR   pfd= {
                    sizeof(PIXELFORMATDESCRIPTOR),
                    1,
                    dwFlags,
                    PFD_TYPE_RGBA,
                    24,
                    0,0,0,0,0,0,
                    0,0,
                    0,0,0,0,0,
                    24,
                    0,
                    0,
                    PFD_MAIN_PLANE,
                    0,
                    0,0,
                    };
PixelFormat = ChoosePixelFormat(DC, &pfd);
if (PixelFormat){
        SetPixelFormat(DC, PixelFormat, &pfd);
        RC=wglCreateContext(DC);
        }
```

❖ OpenGL의 기초

▪ 장면 그리기 준비

✓ 화면의 크기와 종횡비 등을 결정하기




```
XRes=GetDeviceCaps(DC,LOGPIXELSX);
YRes=GetDeviceCaps(DC,LOGPIXELSY);
AspectRatio=((GLfloat)Width*(GLfloat)YRes)/((GLfloat)Height*(GLfloat)XRes);
```


❖ OpenGL의 기초

▪ 장면 그리기 시작

✓ 배경 지우기

✓ Depth buffer초기화

✓ 조명 및 재질 설정

```
wglMakeCurrent(DC,RC);
glClearDepth(1);
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_LEQUAL);
glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);
glClearAccum(0,0,0,0);
glDisable(GL_CULL_FACE);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_ACCUM_BUFFER_BIT);
glEnable( GL_COLOR_MATERIAL);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
```

❖ OpenGL의 기초

▪ 장면 그리기 시작

✓ 시점 및 모델 표시 방법 결정

– 행렬을 이용한 순차적 변환 또는 명시적인 지정

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(60.0f,AspectRatio,1,10000);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,100,0,0,0,0,1,0); // eye point, scene center, up vector
```


❖ OpenGL의 기초

▪ 장면 그리기 종료

✓ Rendering Context 와 화면 Buffer를 바꾸기

– Flicker 방지

```
glFlush();
SwapBuffers(DC);
wglMakeCurrent(DC,NULL);
```

▪ OpenGL 세션 종료

```
wglMakeCurrent(DC,0);
if (RC) wglDeleteContext(RC);
```

❖ TOpenGL Class 설계

- **C++ Builder의 문제**
  - ✓ OnCreate, OnResize, OnPaint Event 에서 Device Context 가 변할 수 있다

```
#include        <vcl.h>
#include        <gl/gl.h>
#include        <gl/glu.h>
#include        "TPoint3D.h"

class   TOpenGL
{
public  :
        TOpenGL(HDC,int,int,int,int);
        ~TOpenGL();

        HDC     LastDC;
        HGLRC   RC;

        int     PixelFormat;
        int     Left,Bottom,Width,Height,XRes,YRes;
        float   AspectRatio;

        void    CreateRenderingContext(HDC);
        void    BeginDraw(HDC);
        void    EndDraw(HDC);
};
```


❖ TOpenGL Class 설계

- **생성자와 파괴자**

```
TOpenGL::TOpenGL(HDC DC,int l,int b,int r,int t)
{
CreateRenderingContext(DC);                 // Rendering Context 만들기

Left=l;
Bottom=b;
Width=r-l;
Height=t-b;
XRes=GetDeviceCaps(DC,LOGPIXELSX);      // 해당 Device context 의 해상도 구하기
YRes=GetDeviceCaps(DC,LOGPIXELSY);
AspectRatio=((GLfloat)Width*(GLfloat)YRes)/((GLfloat)Height*(GLfloat)XRes);
}

TOpenGL::~TOpenGL()
{
wglMakeCurrent(LastDC,0);                   // 최종적으로 사용된 DC 를 이용
if (RC) wglDeleteContext(RC);
RC=0;
}
```

❖ TOpenGL Class 설계

- **CreateRenderingContext 함수**

```
void    TOpenGL::CreateRenderingContext(HDC DC)
{
DWORD dwFlags=PFD_DRAW_TO_WINDOW | PFD_SUPPORT_OPENGL | PFD_DOUBLEBUFFER;
PIXELFORMATDESCRIPTOR   pfd= {
                    sizeof(PIXELFORMATDESCRIPTOR),
                    1,
                    dwFlags,
                    PFD_TYPE_RGBA,
                    24,
                    0,0,0,0,0,0,
                    0,0,
                    0,0,0,0,0,
                    24,
                    0,
                    0,
                    PFD_MAIN_PLANE,
                    0,
                    0,0,
                    };
PixelFormat = ChoosePixelFormat(DC, &pfd);
if (PixelFormat){
        SetPixelFormat(DC, PixelFormat, &pfd);
        RC=wglCreateContext(DC);
        }
}
```


❖ TOpenGL Class 설계

- **장면 그리기 준비**

```
void    TOpenGL::BeginDraw(HDC DC)
{
if (!RC) return;
wglMakeCurrent(DC,RC);
glClearDepth(1);
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_LEQUAL);
glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);
glClearAccum(0,0,0,0);
glDisable(GL_CULL_FACE);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_ACCUM_BUFFER_BIT);

glEnable(GL_COLOR_MATERIAL);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);

glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(60.0f,AspectRatio,1,10000);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(0,0,100,0,0,0,0,1,0); // eye/center/up
}
```

❖ TOpenGL Class 설계

- 장면 그리기 종료

```
void    TOpenGL::EndDraw(HDC DC)
{
LastDC=DC;   // 최종적으로 사용한 DC를 기록
glFlush();
SwapBuffers(DC);
wglMakeCurrent(DC,NULL);
}
```


❖ 테스트 프로그램 개발

- 새 프로젝트 시작
  - ✓ SDI 프로그램을 작성
    - – TMainForm 의 생성자/파괴자

```
#include "TOpenGL.h"
...
class TMainForm : public TForm
{
...

        TOpenGL *GL;
};
```

```
__fastcall TMainForm::TMainForm(TComponent* Owner) : TForm(Owner)
{
GL=0;
}

void __fastcall TMainForm::FormClose(TObject *Sender, TCloseAction &Action)
{
if (GL) delete GL;
GL=0;
Action=caFree;
}
```

❖ 테스트 프로그램 개발

▪ TOpenGL 클래스 초기화

✓ OnCreate, OnResize event handler 작성

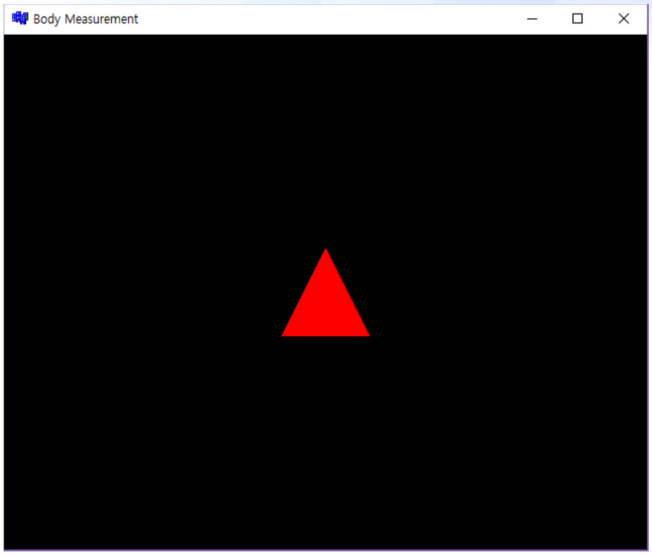
```
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
GL=new TOpenGL(Canvas->Handle,0,0,ClientWidth,ClientHeight);
}

void __fastcall TMainForm::FormResize(TObject *Sender)
{
if (GL) delete GL;
GL=new TOpenGL(Canvas->Handle,0,0,ClientWidth,ClientHeight);
FormPaint(this);
}
```


❖ 테스트 프로그램 개발

▪ 장면 그리기

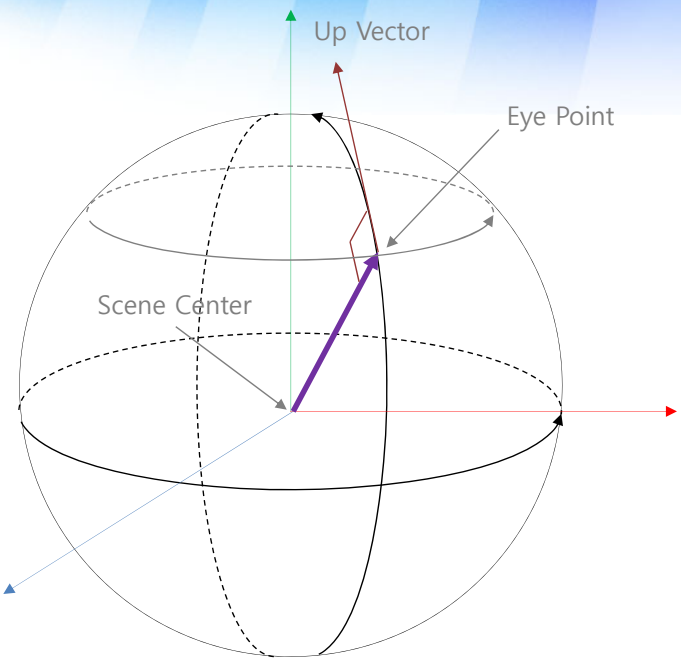
✓ OnPaint event handler 작성

```
void __fastcall TMainForm::FormPaint(TObject *Sender)
{
if (GL){
        GL->BeginDraw(Canvas->Handle);

        glBegin(GL_TRIANGLES);
        glColor3f(1,0,0);
        glNormal3f(0,0,1);
        glVertex3f(-10,-10,0);
        glVertex3f(10,-10,0);
        glVertex3f(0,10,0);
        glEnd();

        GL->EndDraw(Canvas->Handle);
        }
}
```



❖ 테스트 프로그램 개발

- **시점 변환 기능**
  - ✓ 회전이동
    - – EyePoint 와 UpVector 를 조절
  - ✓ 확대축소
    - – Scene Center 와 EyePoint 거리 조절
    - – Scene Center 와 EyePoint 위치 이동



❖ 테스트 프로그램 개발

- **Mouse Event Handler 정의**
  - ✓ TOpenGL 클래스에 mouse event handler 함수를 정의

```
void __fastcall TMainForm::FormMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift, int X, int Y)
{
if (GL) GL->MouseDown(Button,Shift,X,Y);
}

void __fastcall TMainForm::FormMouseMove(TObject *Sender, TShiftState Shift, int X, int Y)
{
if (GL){
        if (GL->MouseMove(Shift,X,Y)) FormPaint(this);
        }
}

void __fastcall TMainForm::FormMouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift, int X, int Y)
{
if (GL) GL->MouseUp(Button,Shift,X,Y);
}
```

❖ 테스트 프로그램 개발

▪ 시점 변경에 필요한 변수 및 함수 설정

```
bool        DragStart;
TPoint3D    O_SceneCenter,O_EyePoint,O_UpVector;    // 초기 viewport
TPoint2D    Rotation,Start;                         // 회전량, 마우스 시작점
void        CalculateViewport();                    // 시점 계산 함수

void        MouseDown(TMouseButton Button,TShiftState Shift, int X, int Y);
bool        MouseMove(TShiftState Shift,int X, int Y);
void        MouseUp(TMouseButton Button,TShiftState Shift, int X, int Y);
```

```
TOpenGL::TOpenGL(HDC DC,int l,int b,int r,int t)
{
...
O_SceneCenter.Set(0,0,0);
O_EyePoint.Set(0,0,100);
O_UpVector.Set(0,1,0);
Rotation.Set(0,0);
CalculateViewport();
DragStart=false;
}
```


❖ 테스트 프로그램 개발

▪ 시점 회전

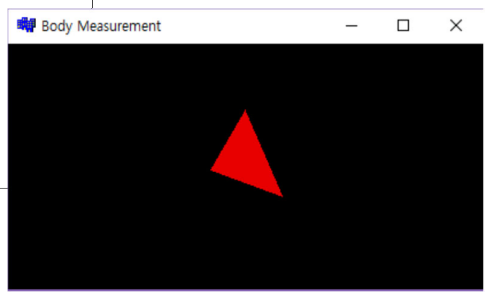
```
void    TOpenGL::MouseDown(TMouseButton Button,TShiftState Shift, int X, int Y)
{
if (!DragStart){
        DragStart=true;
        Start.Set(X,Y);             // 마우스 시작점
        }
}

bool    TOpenGL::MouseMove(TShiftState Shift,int X, int Y)
{
if (DragStart){
        Rotation.x+=(float)(X-Start.x)/2.0*M_PI/180.0;      // 경도 변화량
        Rotation.y+=(float)(Y-Start.y)/2.0*M_PI/180.0;      // 위도 변화량
        CalculateViewport();                                // 시점 재계산
        Start.Set(X,Y);                                     // 마우스 시작점 재 설정
        return true;
        }
return false;
}

void    TOpenGL::MouseUp(TMouseButton Button,TShiftState Shift, int X, int Y)
{
if (DragStart) DragStart=false;
}
```

❖ 테스트 프로그램 개발

▪ 시점 회전

✓ 시점 재계산

```
void    TOpenGL::CalculateViewport()
{
TPoint3D        O,YAxis,Normal;
O.Set(0,0,0);
YAxis.Set(0,1,0);

// 경도 방향 회전

EyePoint=O_EyePoint.RotatedAbout(O_SceneCenter,YAxis,Rotation.x);
UpVector=O_UpVector.RotatedAbout(O,YAxis,Rotation.x);

// 위도 방향 회전

Normal=O_SceneCenter.CrossProduct(EyePoint,O_SceneCenter+YAxis);
Normal.Normalize();
EyePoint=EyePoint.RotatedAbout(O_SceneCenter,Normal,Rotation.y);
UpVector=UpVector.RotatedAbout(O,Normal,Rotation.y);
}
```



❖ 테스트 프로그램 개발

▪ 시점 이동

✓ 시점 이동에 필요한 변수 추가

```
class TOpenGL
{
...
bool            DragStart,Rotating;
float           Distance;
TPoint3D        O_SceneCenter,O_EyePoint,O_UpVector;
TPoint2D        Rotation,Start;
...
}
```

```
TOpenGL::TOpenGL()
{
...
Distance=100;
CalculateViewport();
}
```

❖ 테스트 프로그램 개발

▪ 시점 이동

✓ 시점 이동에 필요한 코드 추가

```
void    TOpenGL::MouseDown(TMouseButton Button,TShiftState Shift, int X, int Y)
{
if (!DragStart){
        DragStart=true;
        Start.Set(X,Y);
        Rotating=(Button==mbLeft);
        }
}
```


❖ 테스트 프로그램 개발

▪ 시점 이동

✓ 시점 이동에 필요한 코드 추가

```
bool    TOpenGL::MouseMove(TShiftState Shift,int X, int Y)
{
if (DragStart){
        if (Rotating){
                Rotation.x+=(float)(X-Start.x)/2.0*M_PI/180.0;
                Rotation.y+=(float)(Y-Start.y)/2.0*M_PI/180.0;
                }
        else{
                float dy=Y-Start.y;
                if (dy>0) Distance*=1.02f;
                else Distance*=0.98f;
                }
        CalculateViewport();
        Start.Set(X,Y);
        return true;
        }
return false;
}
```

❖ 테스트 프로그램 개발

▪ 시점 이동

✓ 시점 이동에 필요한 코드 추가

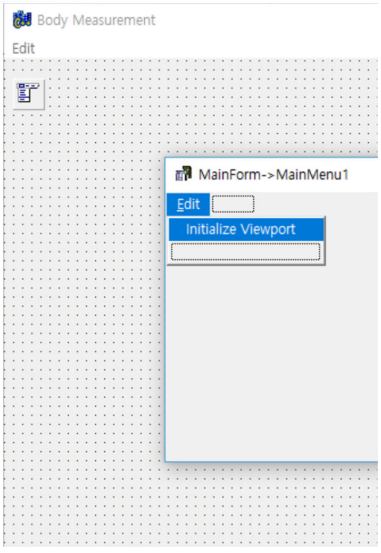
```
void    TOpenGL::CalculateViewport()
{
TPoint3D        O,YAxis,Normal;
O.Set(0,0,0);
YAxis.Set(0,1,0);
O_EyePoint.Set(0,0,Distance);
EyePoint=O_EyePoint.RotatedAbout(O_SceneCenter,YAxis,Rotation.x);
UpVector=O_UpVector.RotatedAbout(O,YAxis,Rotation.x);

Normal=O_SceneCenter.CrossProduct(EyePoint,O_SceneCenter+YAxis);
Normal.Normalize();
EyePoint=EyePoint.RotatedAbout(O_SceneCenter,Normal,Rotation.y);
UpVector=UpVector.RotatedAbout(O,Normal,Rotation.y);
}
```

✓ 시점을 상하좌우로 평행 이동하려면 (Translation) 어떻게 해야할까 ? (Homework)

❖ 테스트 프로그램 개발

▪ 시점 초기화

✓ 메뉴를 추가




```
void __fastcall TMainForm::InitializeViewport1Click(TObject *Sender)
{
if (GL){
        GL->InitViewport(100);
        FormPaint(this);
        }
}
```

```
void    TOpenGL::InitViewport(float dist)
{
Distance=dist;
O_SceneCenter.Set(0,0,0);
O_UpVector.Set(0,1,0);
Rotation.Set(0,0);
CalculateViewport();
}
```

```
TOpenGL::TOpenGL(HDC DC,int l,int b,int r,int t)
{
...
InitViewport(100);
DragStart=false;
}
```