11월 4주차 레포트 기계공학부 이원목

Lighting and Shader

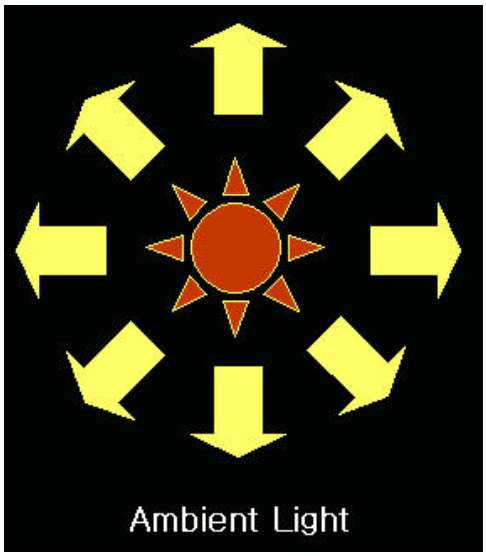
```
glShadeModel(GL_SMOOTH)
glEnable(GL_NORMALIZE)

glClearColor(0.0, 0.0, 0.0, 1.0)
glClearDepth(1.0)
glEnable(GL_DEPTH_TEST)
glEnable(GL_LIGHTING)

ambient = [1.0,1.0,1.0,1.0]
diffuse=[1.0,1.0,1.0,0.2]
specular=[1.0,1.0,1.0,0.2]
position=[5.0,5.0,5.0,5.0]

mat_ambient=[0.5,0.5,0.5,0.0]
mat_diffuse=[0.6,0.6,0.6,0.0]
mat_specular=[0.7,0.7,0.7,0.0]
mat_emissive=[0.0,0.0,0.0,0.0]
mat_shininess=[30.0]

glPushMatrix()
glPushMatrix()
glLightfv(GL_LIGHT0, GL_AMBIENT, ambient)
glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse)
glLightfv(GL_LIGHT0, GL_SPECULAR, specular)
glLightfv(GL_LIGHT0, GL_POSITION, position)
glEnable(GL_LIGHT0)
glPopMatrix()

glPushMatrix()
glEnable(GL_COLOR_MATERIAL)
glMaterialfv(GL_FRONT_AND_BACK,GL_AMBIENT,mat_ambient)
glMaterialfv(GL_FRONT_AND_BACK,GL_DIFFUSE,mat_diffuse)
glMaterialfv(GL_FRONT_AND_BACK,GL_SPECULAR,mat_specular)
glMaterialfv(GL_FRONT_AND_BACK,GL_SHININESS,mat_shininess)
glMaterialfv(GL_FRONT_AND_BACK,GL_EMISSION,mat_emissive)
glPopMatrix()
glPopMatrix()
```

1. Lighting

1-1. 주변광 (Ambient Light)

모든 방향에서 나타나는 조명

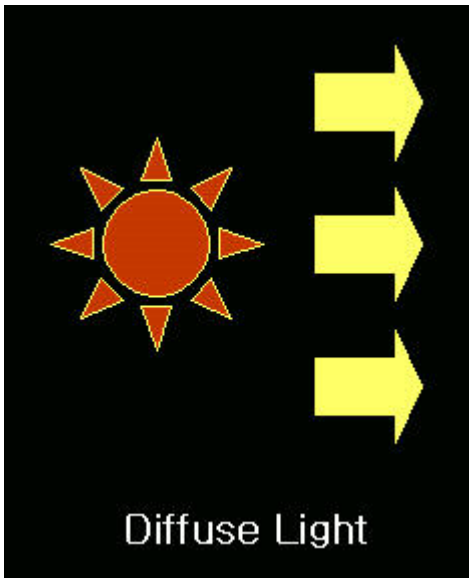
전체적인 분위기를 만들어내는 조명 (ex : 태양)



1-2. 발산광 (Diffuse Light)

한 방향으로부터 나오는 조명

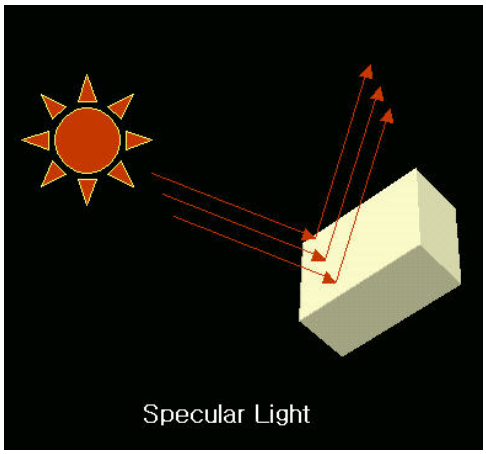
특정한 방향과 위치를 지닌다



1-3. 반사광 (Specular Light)

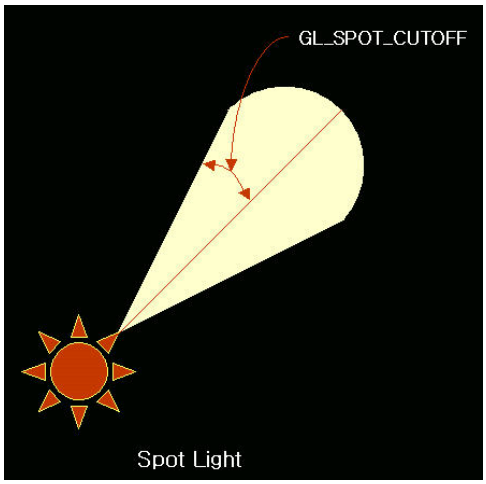
반사광은 일정한 방향으로부터 나온다

높은 반사율일수록 오브젝트 표면에서 흰색에 가까운 색상을 형성



1-4. 광원 (Spot Light)

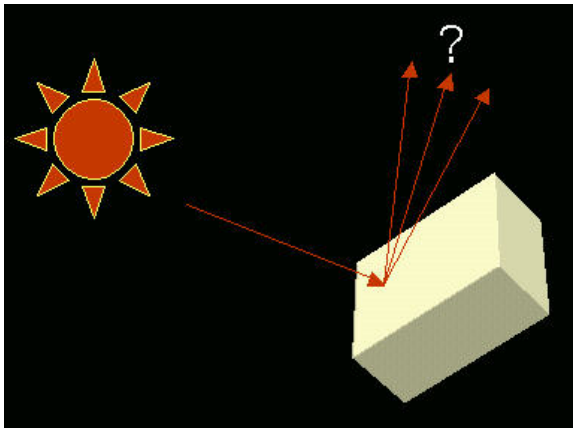
무대 위의 한 인물 혹은 한 곳에 집중된 광선

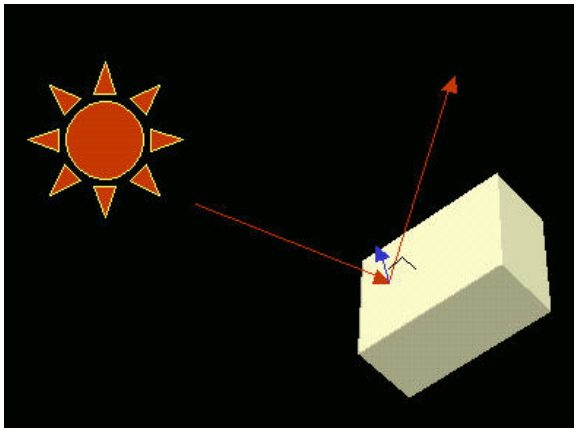


조명 켜기

```
glEnable( GL_LIGHTING ) # 조명 사용
glEnable( GL_LIGHT0 ) # 조명 중에서 0번 조명을 사용

glLightfv( GL_LIGHT0, GL_AMBIENT, AmbientLightValue ) #Ambient 조명의 성질을 설정한다.
glLightfv( GL_LIGHT0, GL_DIFFUSE, DiffuseLightValue ) #Diffuse 조명의 성질을 설정한다.
glLightfv( GL_LIGHT0, GL_SPECULAR, SpecularLightValue ) #Specular 조명의 성질을 설정한다.
glLightfv( GL_LIGHT0, GL_POSTION, PotionLightValue ) # 조명의 위치(광원)를 설정한다.

AmbientLightValue[] = { 0.3f, 0.3f, 0.3f, 1.0f };
DiffuseLightValue[] = { 0.7f, 0.7f, 0.7f, 1.0f };
SpecularLightValue[] = { 1.0f, 1.0f, 1.0f, 1.0f };
PositionLightValue[] = { 0.0f, 0.0f, 1.0f, 0.0f };

## RGBA로 형성
## RGB : (Red, Blue, Green) + Alpha (투명도)
```

1-5. 법선 벡터 설정





빛이 반사되는 평면에서의 법선벡터를 알면 빛의 반사각과 입사각을 구할 수 있다.

```
nv = getNormal( Vector1, Vector2, Vector3)
glNormal3f(nv.x,nv.y.nv.z) 활용하여 법선벡터를 구하자
```

2. 재질 설정

2-1. 주변광(Ambient)과 발산광(Diffuse) 설정

mat-Ambient : 오브젝트 표면이 주변광에 의해서 반사되는 속정

mat-Diffuse : 오브젝트 표면의 발산의 속성

2-2. 반사광(Specular)과 광택(Shininess) 설정

mat-Specular : 표면에서 가장 반사율이 높은 표면에서 나오는 빛의 반사효과

mat-Shininess : 빛을 반사하는 속성의 집중도

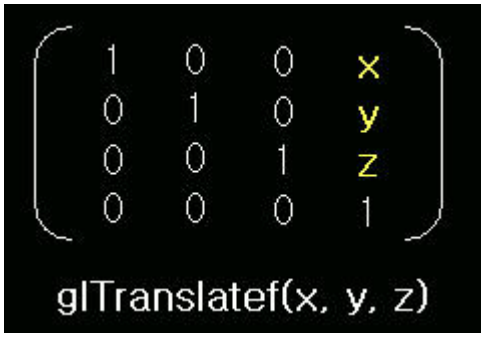
2-3. 발광(Emissive) 재질

mat-Emissive : 오브젝트 스스로가 빛을 내는 속성

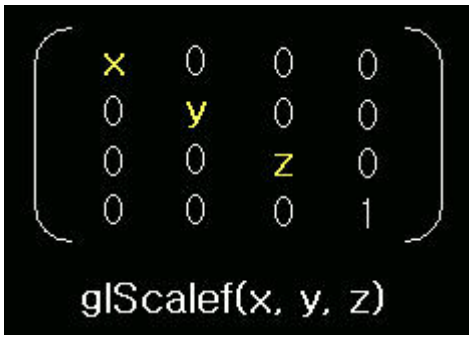
```
glMaterialfv( GL_FRONT, GL_AMBIENT_AND_DIFFUSE, materialAmbient )
glMaterialfv( GL_FORNT, GL_SPECULAR, materialSpecular )

materialAmbient[] = { 0.0f, 0.7f, 0.0f, 1.0f }
materialSpecular[] = { 1.0f, 1.0f, 1.0f, 1.0f }
```

OpenGL 변환

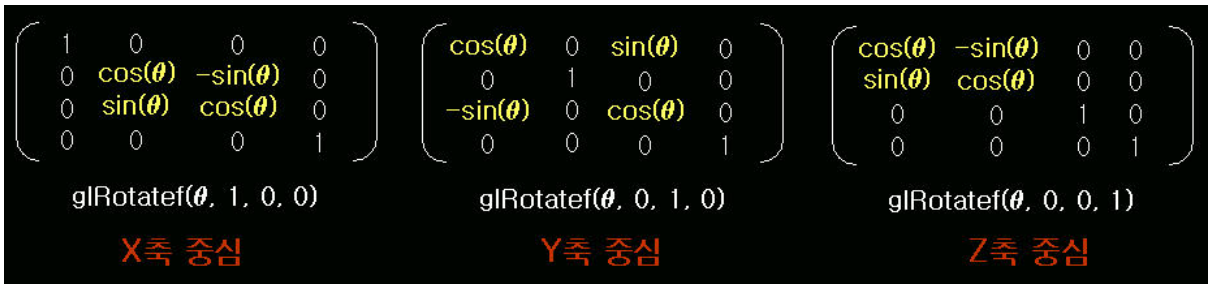
1. glTranslatef(GLfloat X , GLfloat Y , GLfloat Z)



2. glScalef(GLfloat X , GLfloat Y , GLfloat Z)



3. glRotatef(GLfloat angle , GLfloat X , GLfloat Y , GLfloat Z)



```
glLoadIdentity() ## 모델뷰 행렬 초기화
glTranslatef( X, Y, Z ) ## 박스 이동
glRotatef( A, 1.0, 0.0, 0.0) ## Euler-X축 회전
glRotatef( B, 0.0, 1.0, 0.0) ## Euler-Y축 회전
glRotatef( C, 0.0, 0.0, 1.0) ## Euler-Z축 회전
glScalef( X, Y, Z) ## 박스 확대
DrawUnitBox()
```

Viewer 만들기

```
def __init__(self, parent=None) :
    ~~~
    self.world= ~~~
    self.world.setTimeStep(0.005)

    self.timer=QTimer()
    self.timer.timeout.connect(self.update)
    self.timer.start(1000/30)

def paintGL(self):
    self.world.step()
```

1. Viewer Update

1-1. QTimer

QTimer란, PyQt에서 시간의 경과를 체크할 수 있는 객체

1-2. QTimer.start(ms)

QTimer가 시간을 체크하기 시작

1-3. QTimer.setInterval(ms)

QTimer의 Interval를 설정

1-4. QTimer.timeout.connect(함수)

매 Interval마다 어떤 함수를 실행할지를 결정

2. Dartpy Update

2-1. World.setTimeStep(s)

Dartpy내 World에서 사용될 TimeStep 설정

2-2. World.step()

Dartpy내 World에서 한 step만큼 시뮬레이션을 진행시킴

3. Paint Update

```
def Drawskeleton(self):
        for i in range(self.Numbody):
            Body=np.array(self.Human.getBodyNode(i).getWorldTransform().translation())
            Scale=np.array(self.Human.getBodyNode(i).getShapeNode(0).getShape().getSize())
            Rotation=np.array(self.Human.getBodyNode(i).getWorldTransform().rotation())
            Euler=self.cal_Euler(Rotation)

            glPushMatrix()
            glTranslatef(Body[0],Body[1],Body[2])
            glRotatef(Euler[0],1,0,0)
            glRotatef(Euler[1],0,1,0)
            glRotatef(Euler[2],0,0,1)
            glScalef(Scale[0],Scale[1],Scale[2])
            self.Drawunitbox()
            glPopMatrix()

            glFlush()
```

3-1. getWorldTransform.translation()

주어진 BodyNode의 World Coordinate 3차원 좌표계 나타냄

3-2. getWorldTransform.rotation()

주어진 BodyNode의 World Coordinate에서의 각도를 3*3 Matrix로 나타냄

>> 여기서 3*3를 활용하여 어떻게 각도를 추출할 것인가?

```
def cal_Euler(self,Rotation):
        A=-math.asin(Rotation[1][2])*180/(math.pi)
        B=math.asin(Rotation[0][2])*180/(math.pi)
        C=-math.asin(Rotation[0][1])*180/(math.pi)
        Euler=np.array([A,B,C])
        return Euler
```

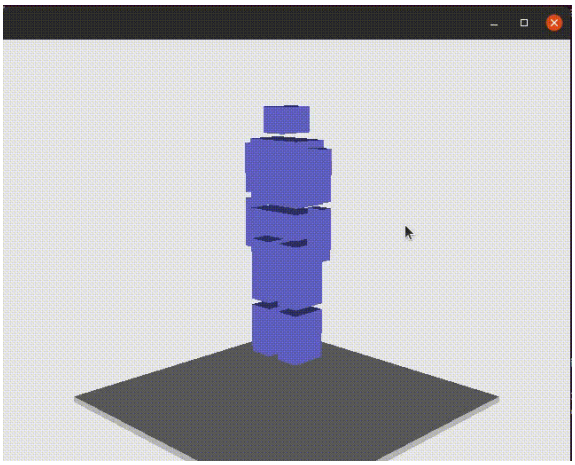
Euler-X(3*3) + Euler-Y(3*3) + Euler-Z(3*3) 일 경우 해결 가능
>> 하지만 시뮬레이션 결과라 다른 모습

Euler-X(3*3) X Euler-Y(3*3) X Euler-Z(3*3) 인 것으로 생각

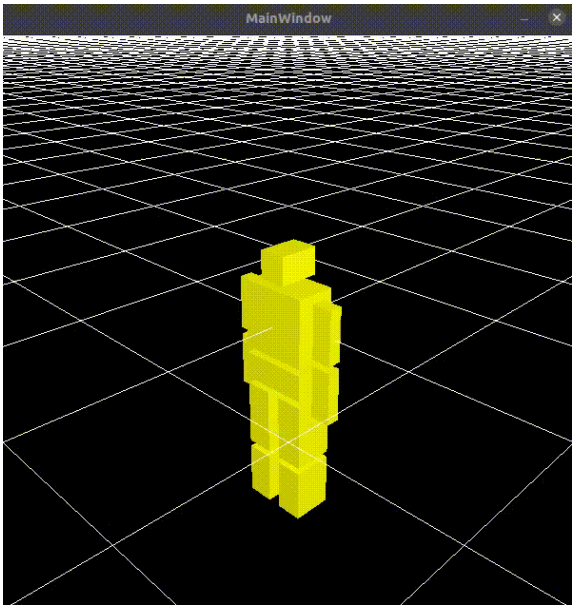
>> 여기서 Euler-X, Y, Z 추출할 방법을 생각해야함

4. MyViewer과 DartViewer 비교

4.1 DartViewer



4.2 MyViewer



>> 동작의 경우에는 제대로 구현이 되는 것처럼 보이나,
>> 시뮬레이션의 속도차가 발생한다 (수정 필요)