Computação Gráfica

Computação Gráfica

OBjeto 3D - Iluminação com Luz Direcional

O método "mostrarFacesPrincipaisComShading" recebe o próprio objeto como parâmetro, um ponto chamado "luzDirecional", que representa a luz apontada para o objeto, e um float chamado "indiceLuzAmbiente", que representa a intensidade da luz ambiente.

O cálculo da tonalidade da face é feito multiplicando a cor base pelo cosseno do ângulo que a normal da face faz com a luz direcional e pela intensidade da luz ambiente.

```
def mostrarFacesPrincipaisComShading(self, luzDirecional, IndiceluzAmbiente):
    obj = self
    facesVisiveis = []
    for face in obj.faces:
        pontoMedioFace = self.getCentroFace(face)
        vetorLuzDirecional = [luzDirecional.x - pontoMedioFace.x, luzDirecional.y - pontoMedioFace.y, luzDirecional.z - pontoMedioFace.z]
        vetorAB = [face.vertices[0].x - face.vertices[1].x, face.vertices[0].y - face.vertices[1].y, face.vertices[0].z - face.vertices[1].z]
        vetorAC = [face.vertices[0].x - face.vertices[2].x, face.vertices[0].y - face.vertices[2].y, face.vertices[0].z - face.vertices[2].z]
        vetorNormal = [vetorAB[1]*vetorAC[2] - vetorAC[1]*vetorAB[2], vetorAB[2]*vetorAC[0] - vetorAC[2]*vetorAB[0], vetorAB[0]*vetorAC[1] - vetorAC[0]*vetorAB[1]]
        produtoEscalar = vetorLuzDirecional[0]*vetorNormal[0] + vetorLuzDirecional[1]*vetorNormal[1] + vetorLuzDirecional[2]*vetorNormal[2]
        if produtoEscalar > 0:
            moduloVetorLuzDirecional = math.sqrt(pow(vetorLuzDirecional[0], 2) + pow(vetorLuzDirecional[1], 2) + pow(vetorLuzDirecional[2], 2))
            moduloVetorNormal = math.sqrt(pow(vetorNormal[0], 2) + pow(vetorNormal[1], 2) + pow(vetorNormal[2], 2))
            cosseno = produtoEscalar / (moduloVetorLuzDirecional * moduloVetorNormal)
            facesVisiveis.append([face, cosseno])

    obj.girarY(45)
    obj.girarX(35.26)
    obj.transladar(590, 260, 0)

    for face in facesVisiveis:
        listaPontos2D = []
        for ponto in face[0].vertices:
            listaPontos2D.append(ponto.get2D())
        pygame.draw.polygon(screen, (0, 0, 255 * face[1] * IndiceluzAmbiente), listaPontos2D)
    obj.transladar(-590, -260, 0)
    obj.girarX(-35.26)
    obj.girarY(-45)
```

OBjeto 3D - Iluminação com Luz Direcional

Alguns métodos auxiliares são utilizados para o cálculo das faces visíveis, bem como para a representação gráfica do objeto e do observador em 2D.

```
def mostrar2D(self):
    for aresta in self.arestas:
            pygame.draw.lines(screen,BLACK,False, aresta.get2D(), 1)

def getCentro(self):
    somaX = 0
    somaY = 0
    somaZ = 0
    quantidadePontos = len(self.vertices)
    for p in self.vertices:
        somaX += p.x
        somaY += p.y
        somaZ += p.z
    return ponto(somaX / quantidadePontos, somaY / quantidadePontos, somaZ / quantidadePontos)

def getCentroFace(self, face):
    somaX = 0
    somaY = 0
    somaZ = 0
    quantidadePontos = len(face.vertices)
    for p in face.vertices:
        somaX += p.x
        somaY += p.y
        somaZ += p.z
    return ponto(somaX / quantidadePontos, somaY / quantidadePontos, somaZ / quantidadePontos)
```

OBjeto 3D - Iluminação com Luz Direcional

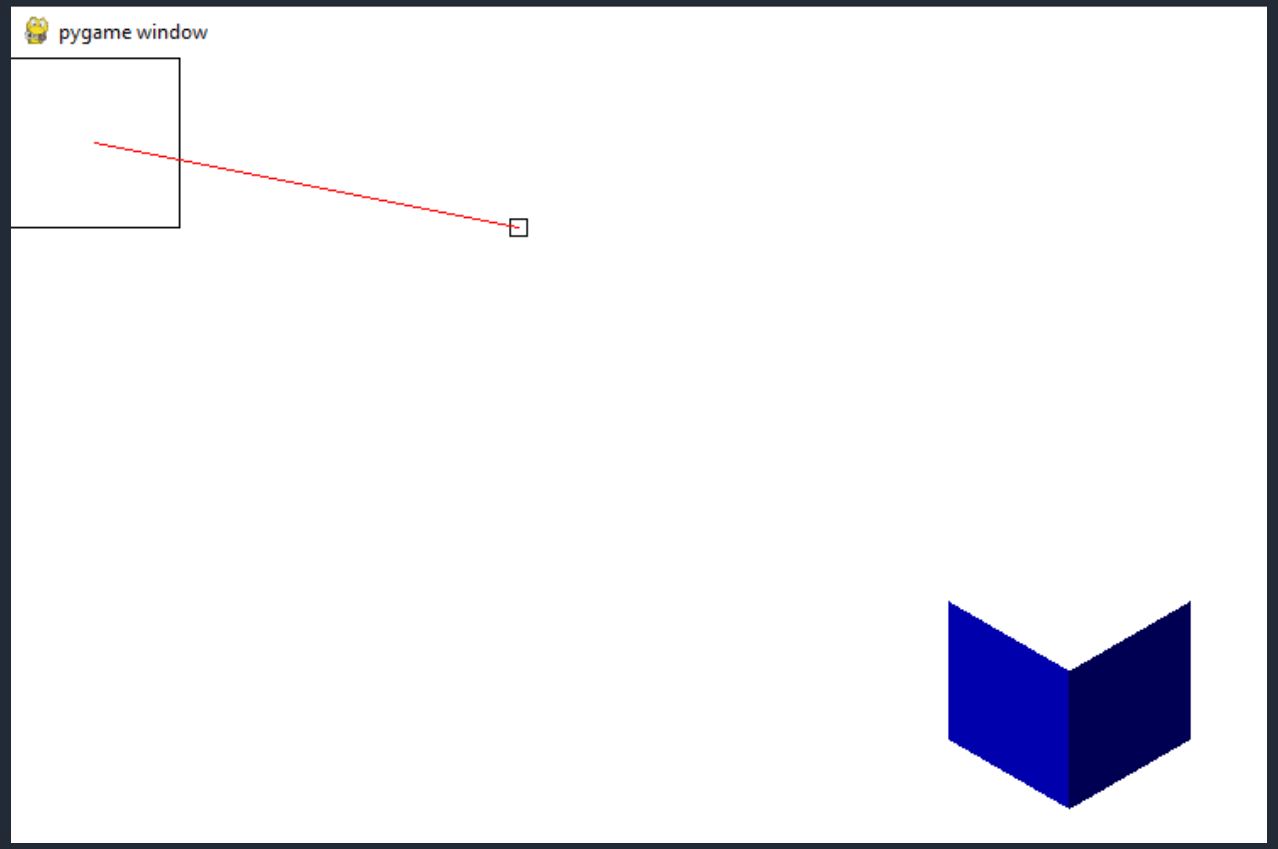
O objeto da luz direcional é definida apenas para representação, mas o ponto da luz, que é utilizado para cálculos em si, é o ponto chamado de "luzDir" na imagem abaixo.

A intensidade da luz ambiente é definida pela variável "indiceLuzAmbiente". Neste caso, o valor escolhido foi 0,8.

```
luzDir = ponto(3000, -5000, 1000)
pontosLuzDir = [ponto(luzDir.x - 5, luzDir.y - 5, luzDir.z), ponto(luzDir.x + 5, luzDir.y - 5, luzDir.z), ponto(luzDir.x - 5, luzDir.y + 5, luzDir.z), ponto(luzDir.x + 5, luzDir.y + 5, luzDir.z)]
arestasLuzDir = [aresta(pontosLuzDir[0], pontosLuzDir[1]), aresta(pontosLuzDir[0], pontosLuzDir[2]), aresta(pontosLuzDir[2], pontosLuzDir[3]), aresta(pontosLuzDir[1], pontosLuzDir[3])]
objetoLuzDir = objeto(pontosLuzDir, arestasLuzDir, [])

indiceLuzAmbiente = 0.8
```

OBjeto 3D - Iluminação com Luz Direcional



A visualização é dada por duas vistas:

Uma vista 2D, com o observador em relação ao centro do objeto, com vista para a face central do objeto.

Uma vista isométrica, mostrando as faces visíveis do objeto com a cor adequada.

OBjeto 3D

Referências

Livro - Teoria da Computação Gráfica 1a Edição, Aura Conci
https://drive.google.com/file/d/15VeqNBTyThiO-OSWQ2N_6iP9DAgWvUej/view?usp=sharing

Slides - Modelos de sombreamento-cores-AS8-2021
https://docs.google.com/presentation/d/1VYWv1XKTmKtucqfKbTCfDvHUj-N3DMPa/edit?usp=sharing&ouid=101569738458696108021&rtpof=true&sd=true

Computação Gráfica

OBjeto 3D

Aluno

Pedro Henrique Mendes Pereira

Disciplina

Computação Gráfica - 2021.1

Professora

Aura Conci

Curso

Ciência da Computação

UFF – Niterói