Rotação e 3 Dimensões

COMPUTAÇÃO GRÁFICA

Funções –Rotação

Podemos fazer a rotação pelo processo geométrico básico:

```
X’ = X * cos ɵ - Y * sin ɵ;
Y’ = X * sin ɵ + Y * cos ɵ;
```

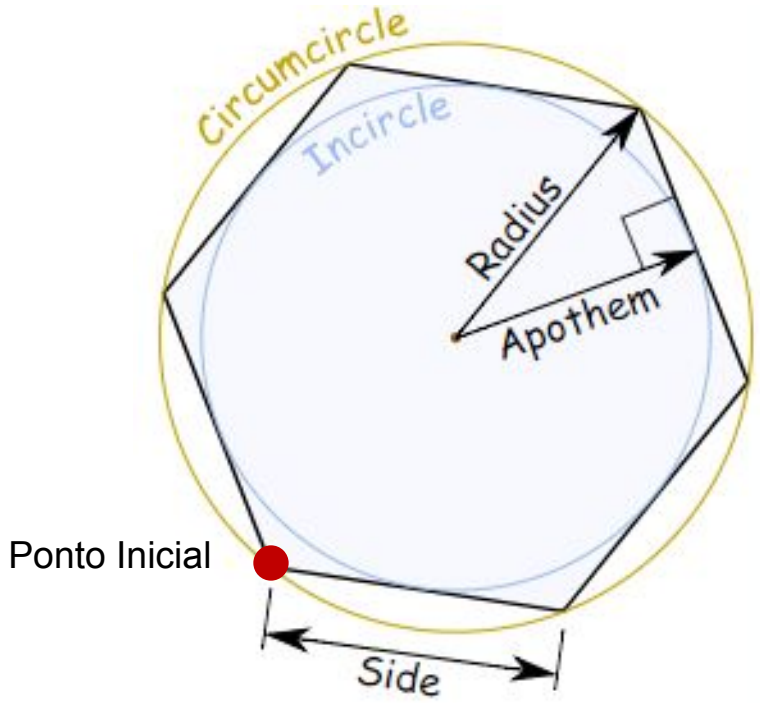
Lembrando que esses cálculos são a partir da origem.

Funções –Rotação

Geralmente esses dados são armazenados em uma matriz de rotação:

$$A = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Polígono



Calcular o Apothem

$$\text{apothem} = \frac{s}{2\tan\left(\frac{180}{n}\right)}$$

Representação 3D

Vamos utilizar o código que está no material didático como teste3d.cpp.

Por enquanto não vamos olhar a fundo cada ponto dele, vamos inicialmente utiliza-lo para ter uma noção dos componentes presentes em 3 dimensões e alguns conceitos aplicados a eles.

Também veremos alguns facilitadores para operações básicas em formas.

Representação 3D

Além de definir a cor de fundo no momento de desenhar a tela, em 3D também definimos a profundidade de limpeza do buffer e como (e se) será feito o culling.

```
glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
glClearDepth(1.0f);
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_LEQUAL);
```

Representação 3D

Para começar a trabalhar com operações aos desenhos que faremos, fazemos operações dentro da matriz de ModelView, entenderemos melhor a diferença entre ela e a Projection em outro momento.

```
glMatrixMode(GL_MODELVIEW);
```

Representação 3D

A função de LoadIndentity reseta as operações da matriz atual para o valor padrão, já que todas as operações aplicadas à elas são cumulativas.

```
glLoadIdentity();
```

Representação 3D

As funções de translação, rotação e escala tem a sintaxe muito parecida, elas modificam a matriz de maneira cumulativa e afetam todas as operações de desenho feitas depois delas.

```
glTranslatef(1.5f, 0.0f, -7.0f);
```

Representação 3D

Desenhar os quadriláteros segue o mesmo processo dos desenhos de linhas que fizemos, somente com mais vértices.

```
glBegin(GL_QUADS);
glColor3f(0.0f, 1.0f, 0.0f);
glVertex3f(1.0f, 1.0f, -1.0f);
glVertex3f(-1.0f, 1.0f, -1.0f);
glVertex3f(-1.0f, 1.0f, 1.0f);
glVertex3f(1.0f, 1.0f, 1.0f);
```

Representação 3D

Para modificar o observador, temos que modificar a matrix para Projection, nela conseguimos determinar como será a visão de perspectiva, por exemplo.

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(45.0f, aspect, 0.1f, 100.0f);
```