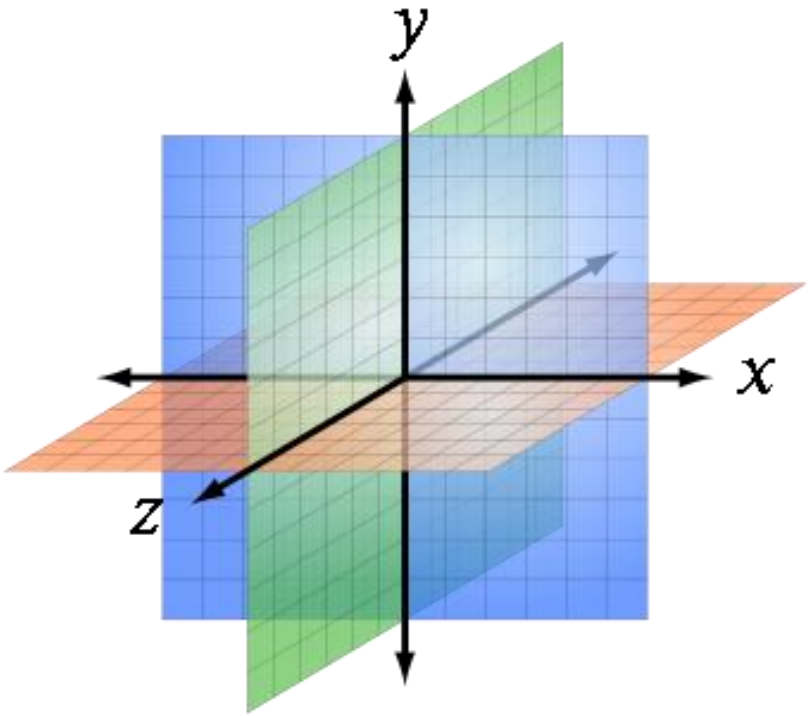
Visualização Tridimensional

Chessman Kennedy Faria Corrêa

Sistema de Coordenadas

- A visualização tridimensional requer o uso de um sistema de coordenadas 3D (**eixos x, y e z**).

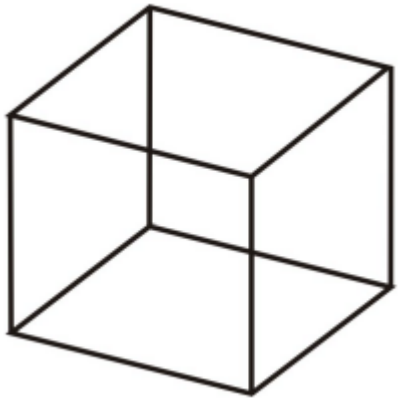


O Espaço 3D

- A visualização 3D requer usar o conceito de cena.
- Nesta cena, todos os objetos são posicionados usando-se o sistema de coordenadas com tridimensional.

Objetos 3D

- Os objetos 3D possuem 3 dimensões:
  - Altura
  - Largura
  - Profundidade

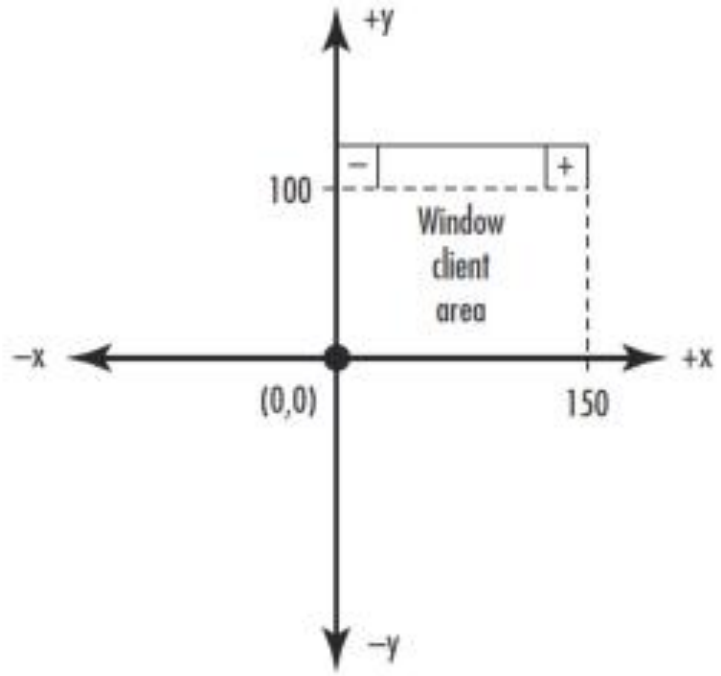


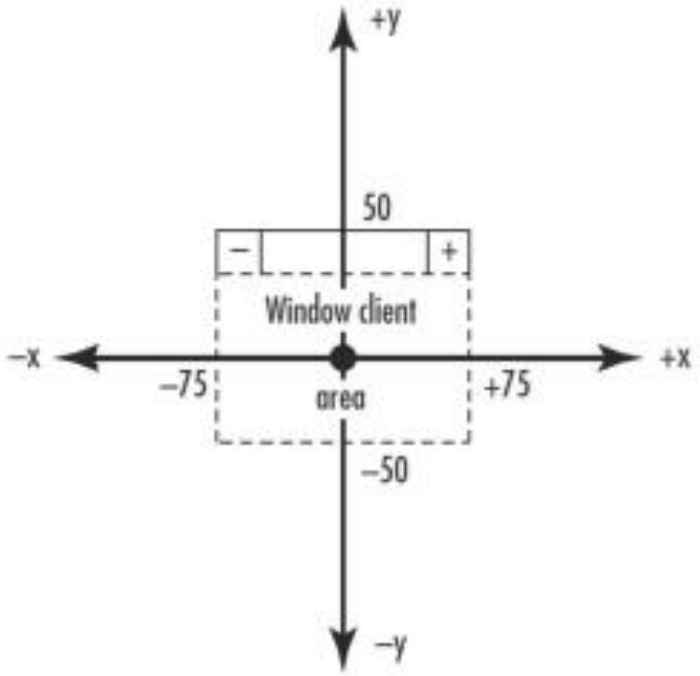
Simulação do Espaço 3D

- Consegue-se simular o espaço 3D da seguinte forma:
  - Objetos mais próximos devem ser maiores que os objetos mais distantes.
  - Objetos mais próximos devem ter mais detalhes que os objetos mais distantes. **Texturas** são muito usadas para se obter este efeito.
  - Usar efeitos de iluminação e sobra.
  - Mudar a intensidade da cor dos objetos.
  - Esconder partes que estão sendo obstruídas pelos objetos da frente.

Região de Corte (Clipping)

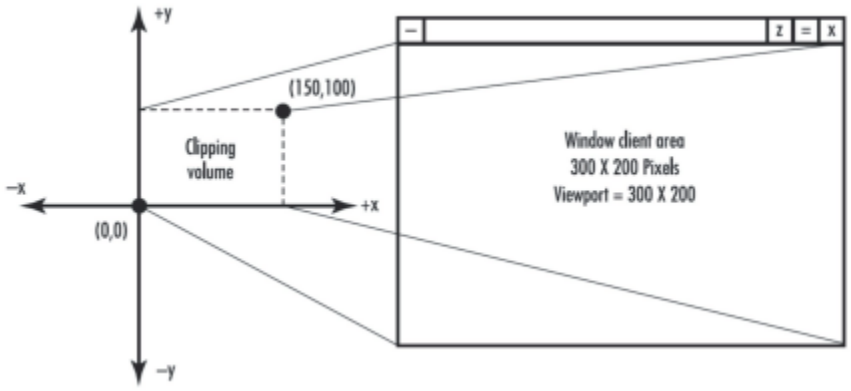
- Determina a **janela de visualização** no espaço de coordenadas cartesiano. O que está fora desta espaço não pode ser visto pelo observador.



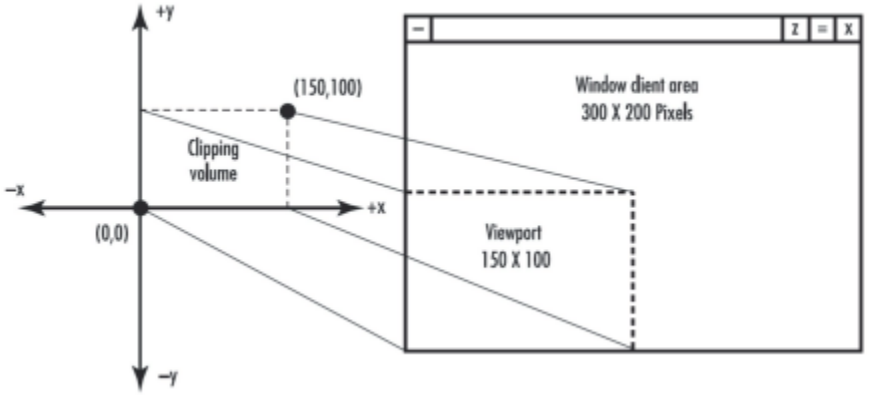


Viewport

- Região de corte dentro da janela de visualização.
- Apenas o que está dentro da região do viewport é exibido para o usuário.



Viewport ocupando toda a região da janela de visualização.



Viewport ocupando parte da região da janela de visualização.

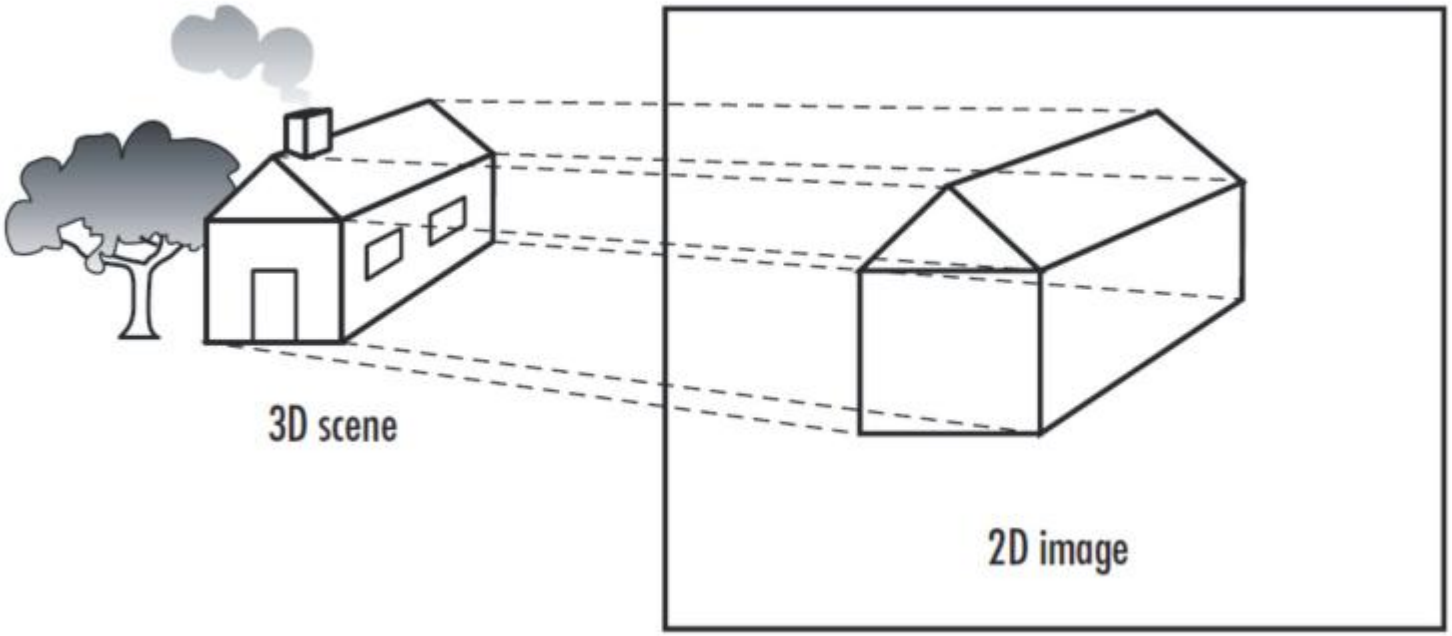
Renderização

- A renderização (*rendering*) é a criação de uma imagem 2D para a tela a partir de um conjunto de outras imagens ou de processamento matemático.
- É na renderização que ocorre a geração de uma imagem 2D a partir de uma cena 3D.
- A renderização gera uma **projeção** do espaço 3D (as coordenadas 3D são projetadas em um espaço de coordenadas 2D).

Projeção

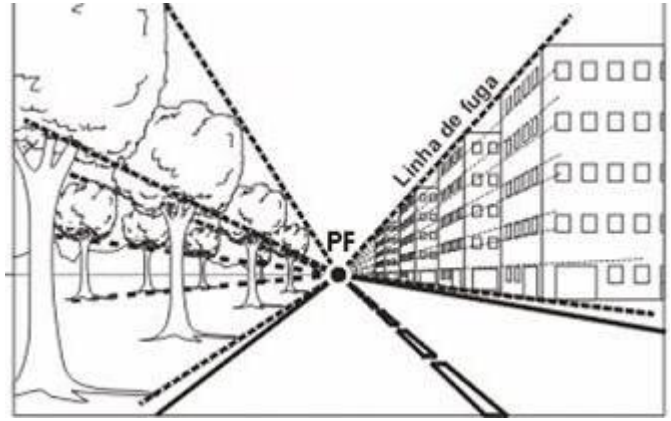
- É a geração da imagem 2D a partir do espaço 3D. Isto significa que as coordenadas 3D são projetadas em um espaço de coordenadas 2D.
- Dependendo de onde se encontram, objetos podem ser desenhados **totalmente** (estão na frente de outros objetos), **parcialmente** (estão sendo obstruídos) ou não **desenhados** (estão atrás de outros objetos).
- A projeção á gerada a partir raios de projeção, chamados de **projetantes**.

Projeção

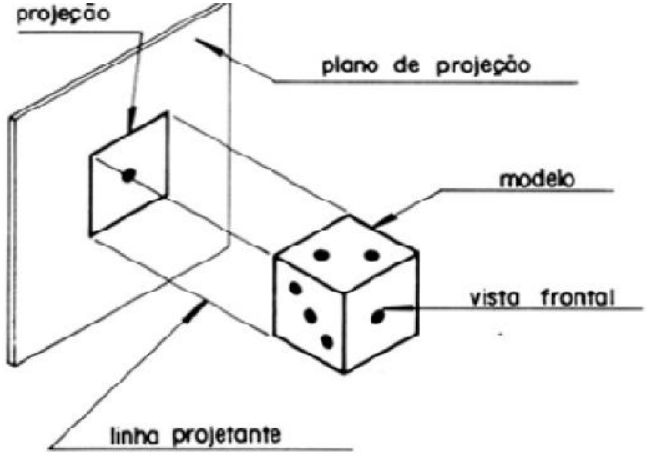


Tipos de Projeção

- **Paralela** (ou **ortográfica**): As projeções são realizadas de forma paralela (não há noção de distância).
- **Perspectiva: As projeções são realizadas a partir de um ponto no horizonte (sensação de distância).**



Projeção perspectiva

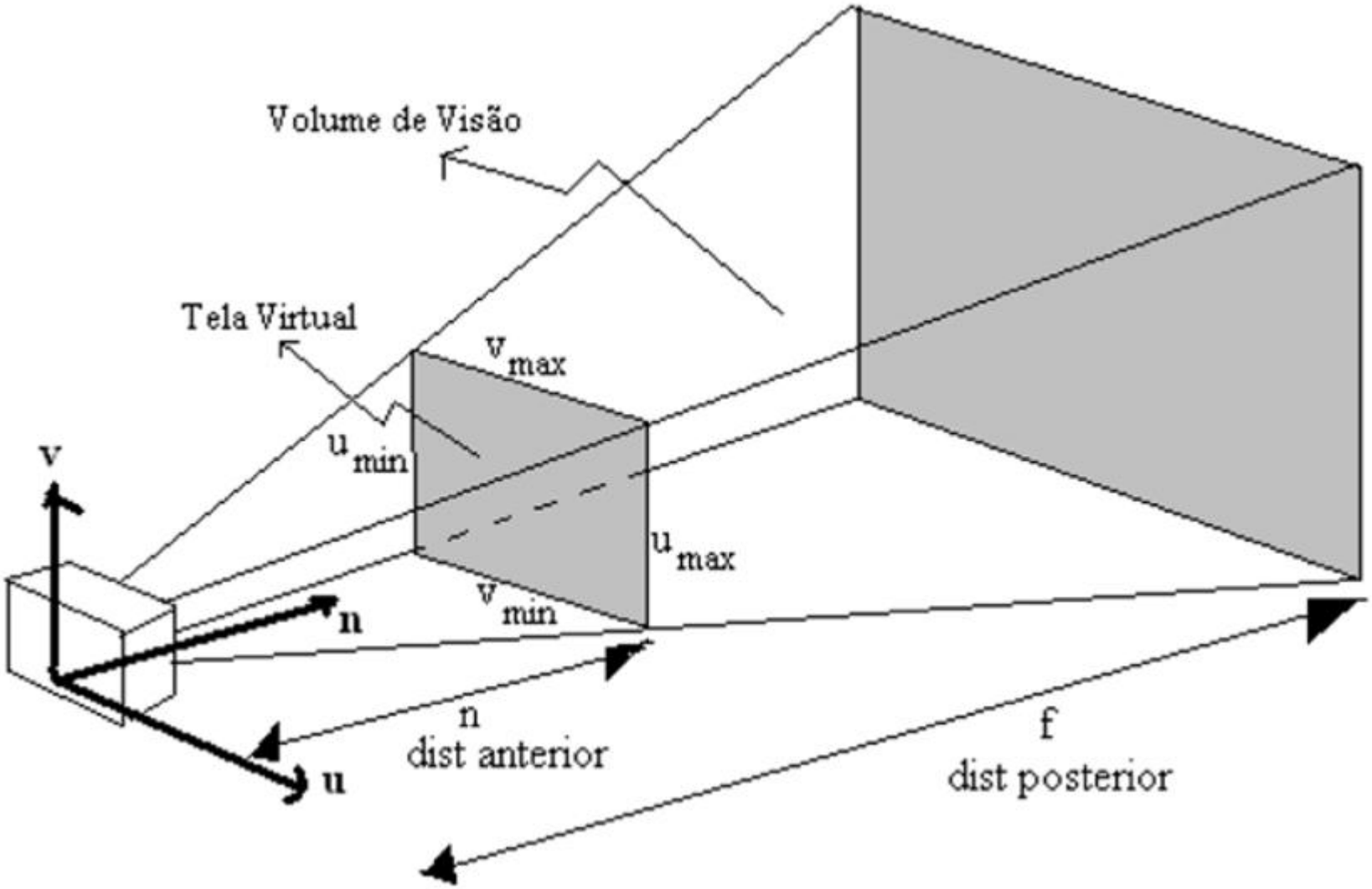


Projeção ortográfica

Câmera e Projeção

- A imagem a ser exibida para o usuário é 2D.
- A **câmera** determina o que o usuário irá ver.
- Com base na posição da câmera, a imagem 2D á gerada a partir da posição dos objetos em relação à câmera.

Câmera


Volume de Visão
Tela Virtual
v max
u min
u max
v min
v
n
u
n
dist anterior
f
dist posterior

Projeção em Perspectiva

- A função usada para definir a perpectiva é **gluPerspective**(GLdouble *fovy*, GLdouble *aspect*, GLdouble *zNear*, GLdouble *zFar*);

- **Parâmetros:**
  - ***fovy****: campo do ângulo de visão em graus (na direção y)*.
  - ***aspect****:* campo do ângulo de visão (na direção x). Deve ser o valor da operação largura/altura.
  - ***zNear*Specifies**: distância mais próxima do observador em relação a área de corte (sempre positivo).
  - ***zFar*Specifies**: distância mais afastada do observador da área de corte (sempre positivo).

Projeção Ortográfica

- A função usada para definir a projeção ortográfica é **void glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble**
- **top, GLdouble near, GLdouble far);**
- Parâmetros:
  - **left** e **right**: limites mínimo e máximo do eixo x (largura).
  - **bottom** e **up**: limites mínimo e máximo do eixo y (altura)
  - **near** e **far**: limites mínimo e máximo do eixo z (profundidade).

Posicionamento da Câmera

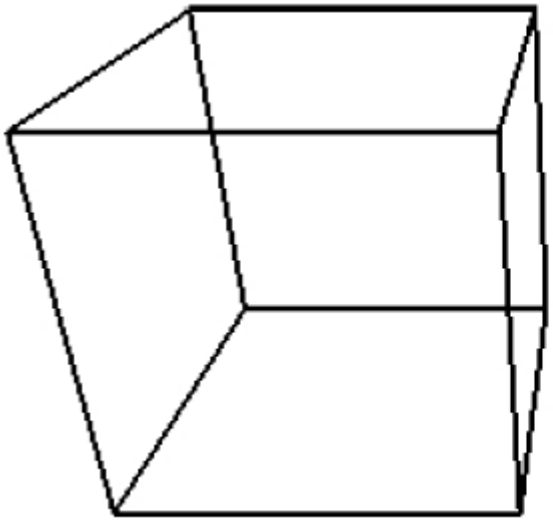
- Use a função **void gluLookAt**(double eyex, double eyey, double eyez, double centerx, double centery, double centerz, double upx, double upy, double upz)
- Parâmetros:
  - **eyex**: coordenada x do centro da câmera
  - **eyey**: coordenada y do centro da câmera
  - **eyez**: coordenada z do centro da câmera
  - **centerx**: coordenada x do ponto de referência
  - **centery**: coordenada y do ponto de referência
  - **centerz**: coordenada z do ponto de referência
  - **upx**: coordenada x do vetor de orientação (up vector)
  - **upy**: coordenada y do vetor de orientação
  - **upz:** coordenada z do vetor de orientação

Alguns Objetos 3D Prontos

- Cubo
- Esfera
- Cone
- Chaleira

Cubo

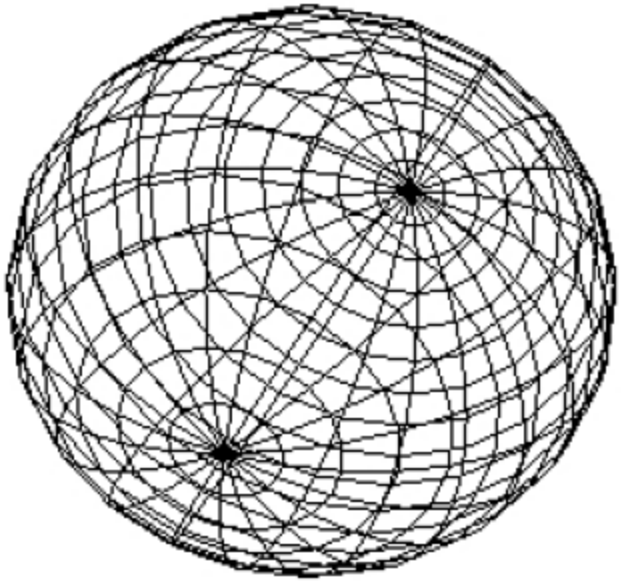
- Cubo aramado:
  - void **glutWireCube**(GLdouble size);
- Cubo sólido:
  - void **glutSolidCube**(GLdouble size);
- O parâmetro size é o tamanho do cubo.

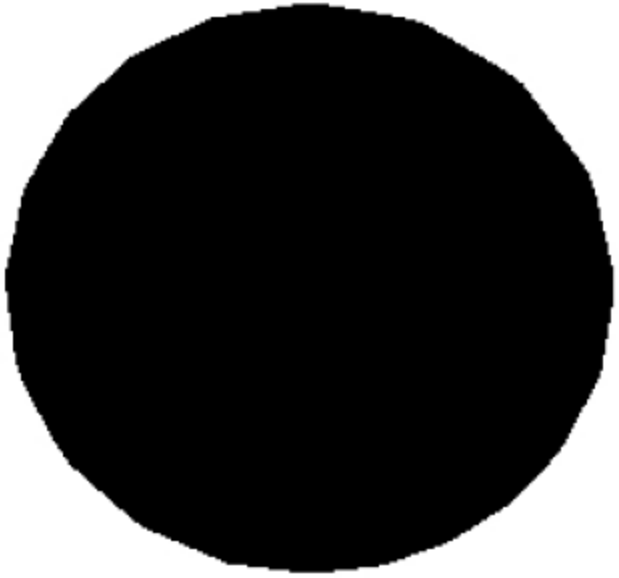


Esfera

- Esfera aramada:
  - void **glutWireSphere**(GLdouble radius, GLint slices, GLint stacks);
- Esfera sólida:
  - void **glutSolidSphere**(GLdouble radius, GLint slices, GLint stacks);
- Parâmetros:
  - radius: raio da esfera.
  - slices: quantidade **linhas longitudinais** em torno do eixo z.
  - stacks: quantidade de **linhas latitudinais** em torno do eixo z.
  - As interseções das linhas longitudinais e latitudinais formam as faces da malha poligonal. Quanto maior a quantidade de linhas, melhor é qualidade da esfera.

Esfera

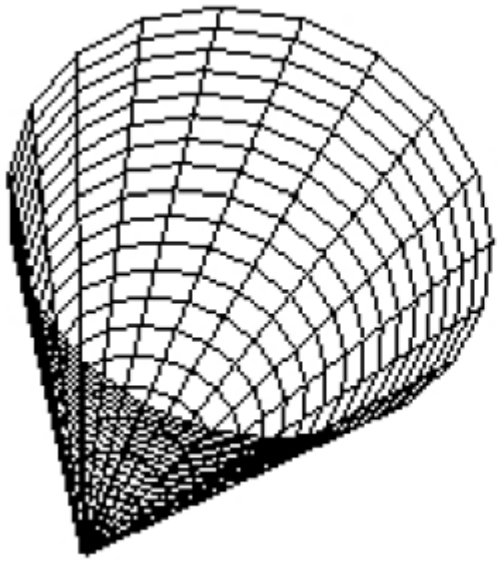




Cone

- Cone aramado:
  - void glutWireCone(GLdouble radius, GLdouble height, GLint slices, GLint stacks);
- Cone sólido:
  - void glutSolidCone(GLdouble radius, GLdouble height, GLint slices, GLint stacks);
- Parâmetros:
  - radius: raio da base do cone
  - height: altura do cone
  - slices: quantidade **linhas longitudinais** em torno do eixo z.
  - stacks: quantidade de **linhas latitudinais** em torno do eixo z.
  - As interseções das linhas longitudinais e latitudinais formam as faces da malha poligonal. Quanto maior a quantidade de linhas, melhor é qualidade do cone.

Cone



Chaleira

- Chaleira aramada:
  - void glutWireTeapot(GLdouble size);
- Chaleira sólida:
  - void glutSolidTeapot(GLdouble size);
- O parâmetro size é o tamanho da chaleira.

Exemplo - Chaleira

```
#include <gl/glut.h>

GLfloat angulo, aspecto;

// Função callback chamada para fazer o desenho
void desenhar(void)
{
    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(0.0f, 0.0f, 1.0f);

    // Desenha o teapot com a cor corrente (wire-frame)
    glutWireTeapot(50.0f);

    // Executa os comandos OpenGL
    glutSwapBuffers();
 }
```

Exemplo - Chaleira

```
// Inicializa parâmetros de rendering
void inicializar (void)
{
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    angulo=45;
}
```

Exemplo - Chaleira

```
// Função usada para especificar o volume de visualização
void configurarVisualizacao(void)
{
    // Especifica sistema de coordenadas de projeção
    glMatrixMode(GL_PROJECTION);
    // Inicializa sistema de coordenadas de projeção
    glLoadIdentity();

    // Especifica a projeção perspectiva
    gluPerspective(angulo,aspecto,0.1,500);

    // Especifica sistema de coordenadas do modelo
    glMatrixMode(GL_MODELVIEW);
    // Inicializa sistema de coordenadas do modelo
    glLoadIdentity();

    // Especifica posição do observador e do alvo
    gluLookAt(0,80,200, 0,0,0, 0,1,0);
}
```

Exemplo - Chaleira

```
// Função callback chamada quando o tamanho da janela é alterado
void alterarTamanhoJanela(GLsizei w, GLsizei h)
{
    // Para previnir uma divisão por zero
    if ( h == 0 ) h = 1;

    // Especifica o tamanho da viewport
    glViewport(0, 0, w, h);

    // Calcula a correção de aspecto
    aspecto = (GLfloat)w/(GLfloat)h;

    configurarVisualizacao();
}
```

Exemplo - Chaleira

```
// Função callback chamada para gerenciar eventos do mouse
void eventoMouse(int button, int state, int x, int y)
{
    if (button == GLUT_LEFT_BUTTON)
        if (state == GLUT_DOWN) {  // Zoom-in
            if (angulo >= 10) angulo -= 5;
        }
    if (button == GLUT_RIGHT_BUTTON)
        if (state == GLUT_DOWN) {  // Zoom-out
            if (angulo <= 130) angulo += 5;
        }
    configurarVisualizacao();
    glutPostRedisplay();
}
```

Exemplo - Chaleira

```
// Programa Principal
int main(void)
{
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(350,300);
    glutCreateWindow("Visualizacao 3D");
    glutDisplayFunc(desenhar);
    glutReshapeFunc(alterarTamanhoJanela);
    glutMouseFunc(eventoMouse);
    inicializar();
    glutMainLoop();
}
```

Outro Exemplo - Cubo

```
#include <stdlib.h>
#include <GL/glut.h>

GLfloat aspecto;
GLfloat angulo;

void inicializar (void)
{
    angulo=60;
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glLineWidth(2.0);
}
```

Outro Exemplo - Cubo

```
void desenhar(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0.0f, 0.0f, 0.0f);
    glutWireCube(50);
    glFlush();
}

void especificaParametrosVisualizacao(void)
{
    // Especifica sistema de coordenadas de projeção
    glMatrixMode(GL_PROJECTION);
    // Inicializa sistema de coordenadas de projeção
    glLoadIdentity();
    // Especifica a projeção perspectiva(angulo,aspecto,zMin,zMax)
    gluPerspective(60,aspecto,0.5,500);
    // Especifica sistema de coordenadas do modelo
    glMatrixMode(GL_MODELVIEW);
    // Inicializa sistema de coordenadas do modelo
    glLoadIdentity();
    // Especifica posição do observador e do alvo
    gluLookAt(40,angulo,100, 0,0,0, 0,1,0);
}
```

Outro Exemplo - Cubo

```
void alteraTamanhoJanela(GLsizei w, GLsizei h)
{
    // Para previnir uma divisão por zero
    if ( h == 0 ) h = 1;

    // Especifica as dimensões da viewport
    glViewport(0, 0, w, h);

    // Calcula a correção de aspecto
    aspecto = (GLfloat)w/(GLfloat)h;

    especificaParametrosVisualizacao();
}

void responderTeclado (unsigned char key, int x, int y)
{
    if (key == 27)
        exit(0);
}
```

Outro Exemplo - Cubo

```
void responderMouse(int button, int state, int x, int y)
{
    if (button == GLUT_LEFT_BUTTON)
        if (state == GLUT_DOWN) {  // Zoom-in
            if (angulo >= 10) angulo -= 5;
        }
    if (button == GLUT_RIGHT_BUTTON)
        if (state == GLUT_DOWN) {  // Zoom-out
            if (angulo <= 130) angulo += 5;
        }
    especificaParametrosVisualizacao();
    glutPostRedisplay();
}
```

Outro Exemplo - Cubo

```
int main(void)
{
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(5,5);
    glutInitWindowSize(450,450);
    glutCreateWindow("Desenho de um cubo");
    glutDisplayFunc(desenhar);
    glutReshapeFunc(alteraTamanhoJanela);
    glutKeyboardFunc (responderTeclado);
    glutMouseFunc(responderMouse);
    inicializar();
    glutMainLoop();
    return 0;
}
```