Projeto3-PA

HÍTALO TARGINO BEZERRA
MATEUS SALVIANO DEL CARBON MACIEL
Versão 1.0
Quinta, 22 de Dezembro de 2022

Sumário

Table of contents

Índice Hierárquico

Hierarquia de Classes

Esta lista de hierarquias está parcialmente ordenada (ordem alfabética):

Índice dos Componentes

Lista de Classes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

Índice dos Arquivos

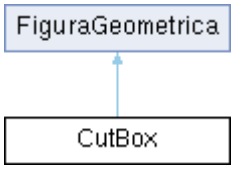
Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

Classes

Referência da Classe CutBox

`#include <cutbox.h>`
Diagrama de hierarquia para CutBox:



Membros Públicos

- **CutBox** (int x0, int x1, int y0, int y1, int z0, int z1)
- void **draw** (**Sculptor** &s)

Outros membros herdados

Construtores e Destrutores

CutBox::CutBox (int *x0*, int *x1*, int *y0*, int *y1*, int *z0*, int *z1*)

```
4                          {
5   this->x0 = x0; this->x1 = x1;
6   this->y0 = y0; this->y1 = y1;
7   this->z0 = z0; this->z1 = z1;
8 }
```

Funções membros

void CutBox::draw (Sculptor & *s*) `[virtual]`

Implementa **FiguraGeometrica** (*p.13*).

```
10                                {
11   int x, y, z;
12   for (x = x0; x <= x1; x++) {
13     for (y = y0; y <= y1; y++) {
14       for (z = z0; z <= z1; z++) {
15         s.cutVoxel(x, y, z);
16       }
17     }
18   }
19 }
```

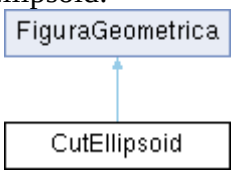
A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**cutbox.h**
- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**cutbox.cpp**

Referência da Classe CutEllipsoid

`#include <cutellipsoid.h>`

Diagrama de hierarquia para CutEllipsoid:



Membros Públicos

- **CutEllipsoid** (int **xcenter**, int **ycenter**, int **zcenter**, int **rx**, int **ry**, int **rz**)
- **~CutEllipsoid** ()
- void **draw** (**Sculptor** &t)

Atributos Protegidos

- int **xcenter**
- int **ycenter**
- int **zcenter**
- int **rx**
- int **ry**
- int **rz**
- float **r**
- float **g**
- float **b**
- float **a**

Construtores e Destrutores

CutEllipsoid::CutEllipsoid (int *xcenter*, int *ycenter*, int *zcenter*, int *rx*, int *ry*, int *rz*)

```
5
{
6     xcenter = _xcenter;
7     ycenter = _ycenter;
8     zcenter =  zcenter;
9     rx      =  rx;
10     ry      = _ry;
11     rz      = _rz;
12 }
```

CutEllipsoid::~CutEllipsoid ()

```
14                              {
15
16 }
```

Funções membros

void CutEllipsoid::draw (Sculptor & *t*) `[virtual]`

Implementa **FiguraGeometrica** (*p.13*).

```
18                                      {
19     for(int x=(xcenter-rx); x<(xcenter+rx); x++){
20         for(int y=(ycenter-ry); y<(ycenter+ry); y++){
```

```
21              for(int z=(zcenter-rz); z<(zcenter+rz); z++){
22                  float t1 =
((float)(x-xcenter)/(float)rx)*((float)(x-xcenter)/(float)rx);
23                  float t2 =
((float)(y-ycenter)/(float)ry)*((float)(y-ycenter)/(float)ry);
24                  float t3 =
((float)(z-zcenter)/(float)rz)*((float)(z-zcenter)/(float)rz);
25      //                cout << t1 << " = " << x << " - "  << xcenter << " / " << rx
<<  endl;
26
27                  if(t1+t2+t3<=1.0){
28                      t.cutVoxel(x,y,z);
29      //                    cout << t1 << " " << t2 << " "  << t3 << endl;
30                  }
31              }
32          }
33      }
34 }
```

Atributos

float CutEllipsoid::a [protected]

float CutEllipsoid::b [protected]

float CutEllipsoid::g [protected]

float CutEllipsoid::r [protected]

int CutEllipsoid::rx [protected]

int CutEllipsoid::ry [protected]

int CutEllipsoid::rz [protected]

int CutEllipsoid::xcenter [protected]

int CutEllipsoid::ycenter [protected]

int CutEllipsoid::zcenter [protected]

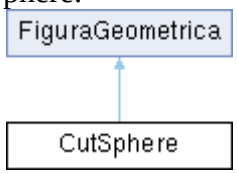
A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**cutellipsoid.h**
- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**cutellipsoide.cpp**

Referência da Classe CutSphere

`#include <cutsphere.h>`

Diagrama de hierarquia para CutSphere:



Membros Públicos

- **CutSphere** (int **xcenter**, int **ycenter**, int **zcenter**, int **radius**)
- **~CutSphere** ()
- void **draw** (**Sculptor** &t)

Atributos Protegidos

- int **xcenter**
- int **ycenter**
- int **zcenter**
- int **radius**
- float **r**
- float **g**
- float **b**
- float **a**

Construtores e Destrutores

CutSphere::CutSphere (int *xcenter*, int *ycenter*, int *zcenter*, int *radius*)

```
3                                                                                {
4     xcenter = _xcenter;
5     ycenter =  ycenter;
6     zcenter =  zcenter;
7     radius  =  radius;
8 }
```

CutSphere::~CutSphere ()

```
10                      {
11
12 }
```

Funções membros

void CutSphere::draw (Sculptor & *t*) `[virtual]`

Implementa **FiguraGeometrica** (*p.13*).

```
14                                {
15     for(int x=(xcenter-radius); x<(xcenter+radius); x++){
16         for(int y=(ycenter-radius); y<(ycenter+radius); y++){
17             for(int z=(zcenter-radius); z<(zcenter+radius); z++){
18                 float x2 = (float)(x-xcenter)*(float)(x-xcenter);
19                 float y2 = (float)(y-ycenter)*(float)(y-ycenter);
20                 float z2 = (float)(z-zcenter)*(float)(z-zcenter);
21                 float r2 = (float)radius*(float)radius;
22
23                 if(x2+y2+z2<r2){
```

```
24                      t.cutVoxel(x,y,z);
25                  }
26              }
27          }
28      }
29  }
```

Atributos

float CutSphere::a `[protected]`

float CutSphere::b `[protected]`

float CutSphere::g `[protected]`

float CutSphere::r `[protected]`

int CutSphere::radius `[protected]`

int CutSphere::xcenter `[protected]`

int CutSphere::ycenter `[protected]`

int CutSphere::zcenter `[protected]`

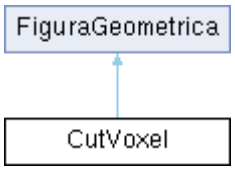
A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**cutsphere.h**
- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**cutsphere.cpp**

Referência da Classe CutVoxel

`#include <cutvoxel.h>`

Diagrama de hierarquia para CutVoxel:



Membros Públicos

- **CutVoxel** (int _x, int _y, int _z, float _r, float _g, float _b, float _a)
- **~CutVoxel** ()
- void **draw** (**Sculptor** &t)

Atributos Protegidos

- int **x**
- int **y**
- int **z**
- float **r**
- float **g**
- float **b**
- float **a**

Construtores e Destrutores

CutVoxel::CutVoxel (int _x, int _y, int _z, float _r, float _g, float _b, float _a)

```
3
{
4      x =  x;
5      y =  y;
6      z = _z;
7      r = _r;
8      g =  g;
9      b =  b;
10      a =  a;
11 }
```

CutVoxel::~CutVoxel ()

```
13                      {
14
15 }
```

Funções membros

void CutVoxel::draw (Sculptor & t) [virtual]

Implementa **FiguraGeometrica** (*p.13*).

```
17                                  {
18      t.cutVoxel(x, y, z);
19 }
```

Atributos

float CutVoxel::a `[protected]`

float CutVoxel::b `[protected]`

float CutVoxel::g `[protected]`

float CutVoxel::r `[protected]`

int CutVoxel::x `[protected]`

int CutVoxel::y `[protected]`

int CutVoxel::z `[protected]`

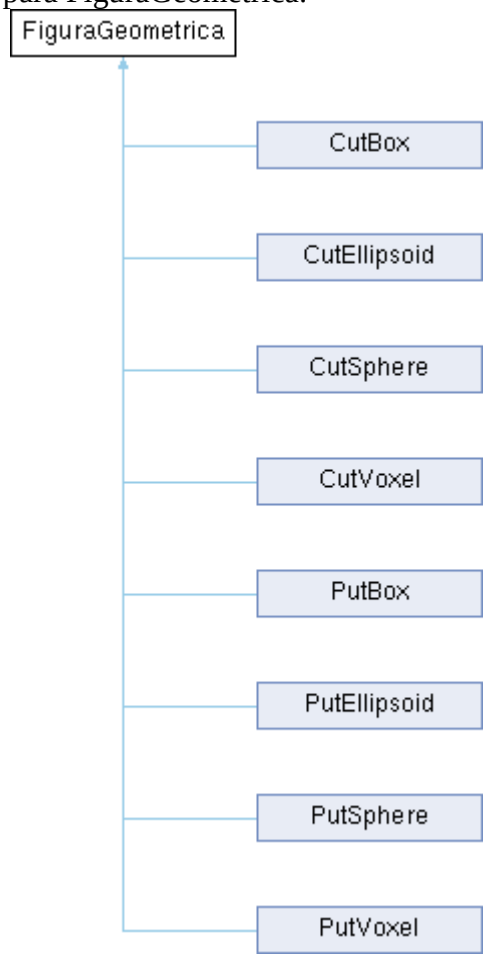
A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**cutvoxel.h**
- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**cutvoxel.cpp**

Referência da Classe FiguraGeometrica

`#include <figurageometrica.h>`

Diagrama de hierarquia para FiguraGeometrica:



Membros Públicos

- **FiguraGeometrica** ()
- virtual **~FiguraGeometrica** ()
- virtual void **draw** (**Sculptor** &s)=0

Atributos Protegidos

- float **r**
- float **g**
- float **b**
- float **a**

Construtores e Destrutores

FiguraGeometrica::FiguraGeometrica ()

```
3 {}
```

virtual FiguraGeometrica::~FiguraGeometrica () `[inline], [virtual]`

```
11 {}
```

Funções membros

virtual void FiguraGeometrica::draw (Sculptor & s) `[pure virtual]`

Implementado por **CutBox** (*p.5*), **PutBox** (*p.14*), **PutSphere** (*p.17*), **CutEllipsoid** (*p.6*), **CutSphere** (*p.8*), **CutVoxel** (*p.10*), **PutEllipsoid** (*p.15*) e **PutVoxel** (*p.19*).

Atributos

float FiguraGeometrica::a `[protected]`

float FiguraGeometrica::b `[protected]`

float FiguraGeometrica::g `[protected]`

float FiguraGeometrica::r `[protected]`

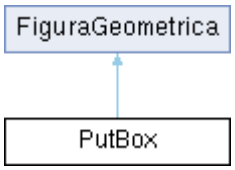
A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**figurageometrica.h**
- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**figurageometrica.cpp**

Referência da Classe PutBox

`#include <putbox.h>`

Diagrama de hierarquia para PutBox:



Membros Públicos

- **PutBox** (int x0, int x1, int y0, int y1, int z0, int z1, float **r**, float **g**, float **b**, float **a**)
- void **draw** (**Sculptor** &s)

Outros membros herdados

Construtores e Destrutores

PutBox::PutBox (int *x0*, int *x1*, int *y0*, int *y1*, int *z0*, int *z1*, float *r*, float *g*, float *b*, float *a*)

```
5 {
6   this->x0 = x0; this->x1 = x1;
7   this->y0 = y0; this->y1 = y1;
8   this->z0 = z0; this->z1 = z1;
9   this->r = r; this->g = g; this->b = b; this->a = a;
10 }
```

Funções membros

void PutBox::draw (Sculptor & s) `[virtual]`

Implementa **FiguraGeometrica** (*p.13*).

```
13 {
14   int x, y, z;
15   s.setColor(r,g,b,a);
16   for (x = x0; x <= x1; x++) {
17     for (y = y0; y <= y1; y++) {
18       for (z = z0; z <= z1; z++) {
19         s.putVoxel(x, y, z);
20       }
21     }
22   }
23 }
```

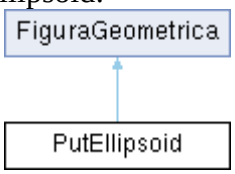
A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**putbox.h**
- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**putbox.cpp**

Referência da Classe PutEllipsoid

`#include <putellipsoid.h>`

Diagrama de hierarquia para PutEllipsoid:



Membros Públicos

- **PutEllipsoid** (int _xcenter, int _ycenter, int _zcenter, int _rx, int _ry, int _rz, float _r, float _g, float _b, float _a)
- **~PutEllipsoid** ()
- void **draw** (**Sculptor** &t)

Atributos Protegidos

- int **xcenter**
- int **ycenter**
- int **zcenter**
- int **rx**
- int **ry**
- int **rz**
- float **r**
- float **g**
- float **b**
- float **a**

Construtores e Destrutores

PutEllipsoid::PutEllipsoid (int *_xcenter*, int *_ycenter*, int *_zcenter*, int *_rx*, int *_ry*, int *_rz*, float *_r*, float *_g*, float *_b*, float *_a*)

```
4
{
5     xcenter =  xcenter;
6     ycenter = _ycenter;
7     zcenter = _zcenter;
8     rx      = _rx;
9     ry      =  ry;
10     rz      =  rz;
11     r       =  r;
12     g       = _g;
13     b       = _b;
14     a       = _a;
15     }
```

PutEllipsoid::~PutEllipsoid ()

```
17                              {
18
19 }
```

Funções membros

void PutEllipsoid::draw (Sculptor & *t*) `[virtual]`

Implementa **FiguraGeometrica** (*p.13*).

```
21                                      {
22     t.setColor(r, g, b, a);
23
24     for(int x=(xcenter-rx); x<(xcenter+rx); x++){
25         for(int y=(ycenter-ry); y<(ycenter+ry); y++){
26             for(int z=(zcenter-rz); z<(zcenter+rz); z++){
27                 float t1 =
((float)(x-xcenter)/(float)rx)*((float)(x-xcenter)/(float)rx);
28                 float t2 =
((float)(y-ycenter)/(float)ry)*((float)(y-ycenter)/(float)ry);
29                 float t3 =
((float)(z-zcenter)/(float)rz)*((float)(z-zcenter)/(float)rz);
30 //                cout << t1 << " = " << x << " - "  << xcenter << " / " << rx <<
endl;
31
32                 if(t1+t2+t3<=1.0){
33                     t.putVoxel(x,y,z);
34 //                    cout << t1 << " " << t2 << " "  << t3 << endl;
35                 }
36             }
37         }
38     }
39 }
```

Atributos

float PutEllipsoid::a [protected]

float PutEllipsoid::b [protected]

float PutEllipsoid::g [protected]

float PutEllipsoid::r [protected]

int PutEllipsoid::rx [protected]

int PutEllipsoid::ry [protected]

int PutEllipsoid::rz [protected]

int PutEllipsoid::xcenter [protected]

int PutEllipsoid::ycenter [protected]

int PutEllipsoid::zcenter [protected]

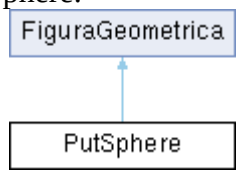
A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**putellipsoid.h**
- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**putellipsoid.cpp**

Referência da Classe PutSphere

`#include <putsphere.h>`

Diagrama de hierarquia para PutSphere:



Membros Públicos

- **PutSphere** (int xcenter, int ycenter, int zcenter, int radius, float r, float g, float b, float a)
- **~PutSphere** ()
- void **draw** (**Sculptor** &s)

Outros membros herdados

Construtores e Destrutores

PutSphere::PutSphere (int *xcenter*, int *ycenter*, int *zcenter*, int *radius*, float *r*, float *g*, float *b*, float *a*)

```
3
{
4     xcenter = _xcenter;
5     ycenter = _ycenter;
6     zcenter =  zcenter;
7     radius  =  radius;
8     r       = _r;
9     g       = _g;
10     b       = _b;
11     a       =  a;
12 }
```

PutSphere::~PutSphere ()

```
14                      {
15
16 }
```

Funções membros

void PutSphere::draw (Sculptor & *s*) `[virtual]`

Implementa **FiguraGeometrica** (*p.13*).

```
18                                  {
19     for(int x=(xcenter-radius); x<(xcenter+radius); x++){
20         for(int y=(ycenter-radius); y<(ycenter+radius); y++){
21             for(int z=(zcenter-radius); z<(zcenter+radius); z++){
22                 float x2 = (float)(x-xcenter)*(float)(x-xcenter);
23                 float y2 = (float)(y-ycenter)*(float)(y-ycenter);
24                 float z2 = (float)(z-zcenter)*(float)(z-zcenter);
25                 float r2 = (float)radius*(float)radius;
26
27                 if(x2+y2+z2<r2){
28                     t.putVoxel(x,y,z);
29                 }
30             }
31         }
```

```
32      }
33 }
```

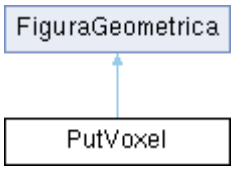
A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**putsphere.h**
- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**putsphere.cpp**

Referência da Classe PutVoxel

`#include <putvoxel.h>`

Diagrama de hierarquia para PutVoxel:



Membros Públicos

- **PutVoxel** (int _x, int _y, int _z, float _r, float _g, float _b, float _a)
- **~PutVoxel** ()
- void **draw** (**Sculptor** &t)

Atributos Protegidos

- int **x**
- int **y**
- int **z**
- float **r**
- float **g**
- float **b**
- float **a**

Construtores e Destrutores

PutVoxel::PutVoxel (int _x, int _y, int _z, float _r, float _g, float _b, float _a)

```
3
{
4      x =  x;
5      y =  y;
6      z = _z;
7      r = _r;
8      g =  g;
9      b =  b;
10      a =  a;
11 }
```

PutVoxel::~PutVoxel ()

```
13                    {
14
15 }
```

Funções membros

void PutVoxel::draw (Sculptor & t) [virtual]

Implementa **FiguraGeometrica** (*p.13*).

```
17                                {
18      t.setColor(r, g, b, a);
19      t.putVoxel(x, y, z);
20 }
```

Atributos

float PutVoxel::a `[protected]`

float PutVoxel::b `[protected]`

float PutVoxel::g `[protected]`

float PutVoxel::r `[protected]`

int PutVoxel::x `[protected]`

int PutVoxel::y `[protected]`

int PutVoxel::z `[protected]`

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**putvoxel.h**
- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**putvoxel.cpp**

Referência da Classe Sculptor

A classe **Sculptor** é responsável por armazenar os voxels e realizar as operações de desenho.
`#include <sculptor.h>`

Membros Públicos

- **Sculptor** (int _nx, int _ny, int _nz)
  *Construct a new **Sculptor** object.*

- **~Sculptor** ()
- void **setColor** (float r, float g, float b, float alpha)
- void **putVoxel** (int x, int y, int z)
- void **cutVoxel** (int x, int y, int z)
- void **putSphere** (int xcenter, int ycenter, int zcenter, int radius)
- void **cutSphere** (int xcenter, int ycenter, int zcenter, int radius)
- void **putEllipsoid** (int xcenter, int ycenter, int zcenter, int rx, int ry, int rz)
  putEllipsoid desenha um elipsoide a partir da equacao

- void **cutEllipsoid** (int xcenter, int ycenter, int zcenter, int rx, int ry, int rz)
- void **writeOFF** (const char *filename)

Descrição detalhada

A classe **Sculptor** é responsável por armazenar os voxels e realizar as operações de desenho.

Ela é capaz de armazenar uma matriz tridimensional que é manipulada por diversos métodos capazes de desenhar (ou apagar) figura diferentes.

Construtores e Destrutores

Sculptor::Sculptor (int _nx, int _ny, int _nz)

Construct a new **Sculptor** object.

Parâmetros

_nx	é o tamanho da matriz na dimensão x
_ny	é o tamanho da matriz na dimensão y
_nz	é o tamanho da matriz na dimensão z • item 1 • item 2 • item 3

```
17                                                  {
18   int i;
19
20   nx =  nx;
21   ny =  ny;
22   nz =  nz;
23   r = g = b = a = 0.5;
24
25   v = new Voxel **[nx];
26   if (v == nullptr) {
27     std::cout << "alloc error\n";
```

```
28      exit(0);
29    }
30
31    v[0] = new Voxel *[nx * ny];
32    if (v[0] == nullptr) {
33      std::cout << "alloc error\n";
34      exit(0);
35    }
36
37    v[0][0] = new Voxel[nx * ny * nz];
38    if (v[0][0] == nullptr) {
39      std::cout << "aalloc error\n";
40      exit(0);
41    }
42
43    // Ajusta os ponteiros para as linhas
44    // cada plano pula de nl em nl linhas
45    for (i = 1; i < nx; i++) {
46      v[i] = v[i - 1] + ny;
47    }
48
49    // ajusta os ponteiros para as colunas
50    // cada linha pula de nc em nc colunas
51    for (i = 1; i < nx * ny; i++) {
52      v[0][i] = v[0][i - 1] + nz;
53    }
54  }
```

Sculptor::~Sculptor ()

```
56                      {
57    delete[] v[0][0];
58    delete[] v[0];
59    delete[] v;
60  }
```

Funções membros

void Sculptor::cutEllipsoid (int *xcenter*, int *ycenter*, int *zcenter*, int *rx*, int *ry*, int *rz*)

```
142                                                                                {
143     double dx, dy, dz;
144     int x, y, z;
145
146     for (x = 0; x < nx; x++) {
147       for (y = 0; y < ny; y++) {
148         for (z = 0; z < nz; z++) {
149           dx = (double)(x - x0) * (double)(x - x0);
150           dy = (double)(y - y0) * (double)(y - y0);
151           dz = (double)(z - z0) * (double)(z - z0);
152           // ponto dentro do elipsoide
153           if (dx / (rx * rx) + dy / (ry * ry) + dz / (rz * rz) < 1) {
154             cutVoxel(x, y, z);
155           }
156         }
157       }
158     }
159   }
```

void Sculptor::cutSphere (int *xcenter*, int *ycenter*, int *zcenter*, int *radius*)

```
104                                                          {
105     double r2;
106     int x, y, z;
107
108     r2 = (double)rr * (double)rr;
109     for (x = 0; x < nx; x++) {
110       for (y = 0; y < ny; y++) {
111         for (z = 0; z < nz; z++) {
112           if ((double)(x - x0) * (double)(x - x0) +
113               (double)(y - y0) * (double)(y - y0) +
114               (double)(z - z0) * (double)(z - z0) <
115               r2) {
```

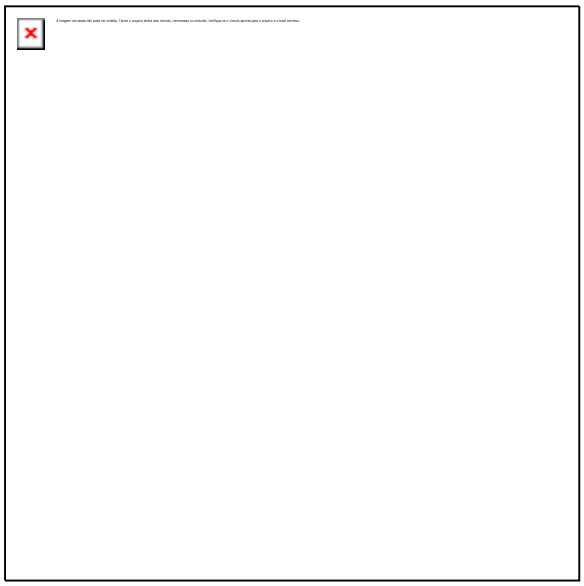
```
116           cutVoxel(x, y, z);
117         }
118       }
119     }
120   }
121 }
```

void Sculptor::cutVoxel (int *x*, int *y*, int *z*)

```
69                                                  {
70   if (x0 >= 0 && x0 < nx && y0 >= 0 && y0 < ny && z0 >= 0 && z0 < nz) {
71     v[x0][y0][z0].show = false;
72   }
73 }
```

void Sculptor::putEllipsoid (int *xcenter*, int *ycenter*, int *zcenter*, int *rx*, int *ry*, int *rz*)

putEllipsoid desenha um elipsoide a partir da equacao



Parâmetros

xcenter	
ycenter	
zcenter	
rx	
ry	
rz	

```
123                                                                    {
124   double dx, dy, dz;
125   int x, y, z;
126
127   for (x = 0; x < nx; x++) {
128     for (y = 0; y < ny; y++) {
129       for (z = 0; z < nz; z++) {
130         dx = (double)(x - x0) * (double)(x - x0);
131         dy = (double)(y - y0) * (double)(y - y0);
132         dz = (double)(z - z0) * (double)(z - z0);
133         // ponto dentro do elipsoide
134         if (dx / (rx * rx) + dy / (ry * ry) + dz / (rz * rz) < 1) {
135           putVoxel(x, y, z);
136         }
137       }
138     }
139   }
140 }
```

void Sculptor::putSphere (int *xcenter*, int *ycenter*, int *zcenter*, int *radius*)

```
85                                                          {
86   double r2;
87   int x, y, z;
88
89   r2 = (double)rr * (double)rr;
90   for (x = 0; x < nx; x++) {
91     for (y = 0; y < ny; y++) {
92       for (z = 0; z < nz; z++) {
93         if ((double)(x - x0) * (double)(x - x0) +
94             (double)(y - y0) * (double)(y - y0) +
95             (double)(z - z0) * (double)(z - z0) <
96             r2) {
97           putVoxel(x, y, z);
98         }
99       }
100     }
101   }
102 }
```

void Sculptor::putVoxel (int *x*, int *y*, int *z*)

```
75                                                  {
76   if (x0 >= 0 && x0 < nx && y0 >= 0 && y0 < ny && z0 >= 0 && z0 < nz) {
77     v[x0][y0][z0].show = true;
78     v[x0][y0][z0].r = r;
79     v[x0][y0][z0].g = g;
80     v[x0][y0][z0].b = b;
81     v[x0][y0][z0].a = a;
82   }
83 }
```

void Sculptor::setColor (float *r*, float *g*, float *b*, float *alpha*)

```
62                                                                  {
63   this->r = r;
64   this->g = g;
65   this->b = b;
66   a = alpha;
67 }
```

void Sculptor::writeOFF (const char * *filename*)

```
161                                                 {
162   int total, index, x, y, z;
163
164   std::ofstream f;
165   total = 0;
166   f.open(filename);
167   f << "OFF\n";
168   // conta a quantidade de voxels ativos
169   for (x = 0; x < nx; x++) {
170     for (y = 0; y < ny; y++) {
171       for (z = 0; z < nz; z++) {
172         if (v[x][y][z].show == true) {
173           total++;
174         }
175       }
176     }
177   }
178   // escreve no de vertices e faces (0 arestas! nao usado)
179   f << total * 8 << " " << total * 6 << " 0 \n";
180   for (x = 0; x < nx; x++) {
181     for (y = 0; y < ny; y++) {
182       for (z = 0; z < nz; z++) {
183         if (v[x][y][z].show == true) {
184           // escreve os vertices do cubo
185           f << x - 0.5 << " " << y + 0.5 << " " << z - 0.5 << "\n";  // 0
186           f << x - 0.5 << " " << y - 0.5 << " " << z - 0.5 << "\n";  // 1
187           f << x + 0.5 << " " << y - 0.5 << " " << z - 0.5 << "\n";  // 2
188           f << x + 0.5 << " " << y + 0.5 << " " << z - 0.5 << "\n";  // 3
189           f << x - 0.5 << " " << y + 0.5 << " " << z + 0.5 << "\n";  // 4
190           f << x - 0.5 << " " << y - 0.5 << " " << z + 0.5 << "\n";  // 5
191           f << x + 0.5 << " " << y - 0.5 << " " << z + 0.5 << "\n";  // 6
192           f << x + 0.5 << " " << y + 0.5 << " " << z + 0.5 << "\n";  // 7
193         }
```

```
194       }
195     }
196   }
197   total = 0;
198   // escreve os indices das faces
199   // pulando vertices de 8 em 8
200   for (x = 0; x < nx; x++) {
201     for (y = 0; y < ny; y++) {
202       for (z = 0; z < nz; z++) {
203         if (v[x][y][z].show == true) {
204           index = total * 8;
205           // face 0
206           f << std::fixed;
207           f << std::setprecision(2);
208           f << 4 << " " << index + 0 << " " << index + 3 << " " << index + 2 <<
" " << index + 1 << " ";
209           f << v[x][y][z].r << " " << v[x][y][z].g << " " << v[x][y][z].b << " "
<< v[x][y][z].a << "\n";
210           // face 1
211           f << 4 << " " << index + 4 << " " << index + 5 << " " << index + 6 <<
" " << index + 7 << " ";
212           f << v[x][y][z].r << " " << v[x][y][z].g << " " << v[x][y][z].b << " "
<< v[x][y][z].a << "\n";
213           // face 2
214           f << 4 << " " << index + 0 << " " << index + 1 << " " << index + 5 <<
" " << index + 4 << " ";
215           f << v[x][y][z].r << " " << v[x][y][z].g << " " << v[x][y][z].b << " "
<< v[x][y][z].a << "\n";
216           // face 3
217           f << 4 << " " << index + 0 << " " << index + 4 << " " << index + 7 <<
" " << index + 3 << " ";
218           f << v[x][y][z].r << " " << v[x][y][z].g << " " << v[x][y][z].b << " "
<< v[x][y][z].a << "\n";
219           // face 4
220           f << 4 << " " << index + 3 << " " << index + 7 << " " << index + 6 <<
" " << index + 2 << " ";
221           f << v[x][y][z].r << " " << v[x][y][z].g << " " << v[x][y][z].b << " "
<< v[x][y][z].a << "\n";
222           // face 5
223           f << 4 << " " << index + 1 << " " << index + 2 << " " << index + 6 <<
" " << index + 5 << " ";
224           f << v[x][y][z].r << " " << v[x][y][z].g << " " << v[x][y][z].b << " "
<< v[x][y][z].a << "\n";
225           // incrementa total de cubos
226           total++;
227         }
228       }
229     }
230   }
231   f.close();
232 }
```

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**sculptor.h**
- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**sculptor.cpp**

Referência da Estrutura Voxel

O struct **Voxel** é responsável por armazenar as informações de cada voxel.
`#include <sculptor.h>`

Atributos Públicos

- float **r**
  r define a intensidade do vermelho (com valores de 0 a 1) g define a intensidade do verde (com valores de 0 a 1) b define a intensidade do azul (com valores de 0 a 1)

- float **g**
- float **b**
- float **a**
  a

- bool **show**
- bool **isOn**
  isOn define a intensidade da transparência (com valores de 0 a 1)

Descrição detalhada

O struct **Voxel** é responsável por armazenar as informações de cada voxel.

Atributos

float Voxel::a

a

float Voxel::b

float Voxel::g

bool Voxel::isOn

isOn define a intensidade da transparência (com valores de 0 a 1)

float Voxel::r

r define a intensidade do vermelho (com valores de 0 a 1) g define a intensidade do verde (com valores de 0 a 1) b define a intensidade do azul (com valores de 0 a 1)

bool Voxel::show

A documentação para essa estrutura foi gerada a partir dos seguintes arquivos:

- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**sculptor.h**
- D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/**voxel.h**

Arquivos

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/cutbox.cpp

```
#include "cutbox.h"
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/cutbox.h

```
#include "figurageometrica.h"
```

Componentes

- class **CutBox**

cutbox.h

Vá para a documentação desse arquivo.

```
1 #ifndef CUTBOX_H
2 #define CUTBOX_H
3
4 #include "figurageometrica.h"
5
6 class CutBox : public FiguraGeometrica
7 {
8   int x0, x1, y0, y1, z0, z1;
9 public:
10   CutBox(int x0, int x1, int y0, int y1,
11          int z0, int z1);
12   void draw(Sculptor &s);
13 };
14
15 #endif // CUTBOX_H
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/cutellipsoid.h

```
#include "figurageometrica.h"
```

Componentes

- class **CutEllipsoid**

cutellipsoid.h

Vá para a documentação desse arquivo.

```
1 #ifndef CUTELLIPSOID_H
2 #define CUTELLIPSOID_H
3
4 #include "figurageometrica.h"
5
6 class CutEllipsoid : public FiguraGeometrica{
7     protected:
8         int xcenter, ycenter, zcenter, rx, ry, rz;
9         float r, g, b, a;
10     public:
11         CutEllipsoid(int xcenter, int ycenter, int zcenter, int rx, int ry, int rz);
12         ~CutEllipsoid();
13         void draw(Sculptor &t);
14 };
15
16 #endif
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/cutellipsoide.cpp

```
#include "cutellipsoid.h"
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/cutsphere.cpp

```
#include "cutsphere.h"
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/cutsphere.h

```
#include "figurageometrica.h"
```

Componentes

- class **CutSphere**

cutsphere.h

Vá para a documentação desse arquivo.

```
1 #ifndef CUTSPHERE_H
2 #define CUTSPHERE_H
3
4 #include "figurageometrica.h"
5
6 class CutSphere : public FiguraGeometrica
7 {
8 protected:
9     int xcenter, ycenter, zcenter, radius;
10     float r, g, b, a;
11 public:
12     CutSphere(int xcenter, int ycenter, int zcenter, int radius);
13     ~CutSphere();
14     void draw(Sculptor &t);
15 };
16
17 #endif
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/cutvoxel.cpp

```
#include "cutvoxel.h"
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/cutvoxel.h

```
#include "figurageometrica.h"
```

Componentes

- class **CutVoxel**

cutvoxel.h

Vá para a documentação desse arquivo.

```
1 #ifndef CUTVOXEL_H
2 #define CUTVOXEL_H
3
4 #include "figurageometrica.h"
5
6 class CutVoxel : public FiguraGeometrica{
7     protected:
8         int x, y, z;
9         float r, g, b, a;
10     public:
11         CutVoxel(int _x, int _y, int _z, float _r, float _g, float _b, float _a);
12         ~CutVoxel();
13         void draw(Sculptor &t);
14 };
15
16 #endif
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/figurageometrica.cpp

```
#include "figurageometrica.h"
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/figurageometrica.h

```
#include "sculptor.h"
#include <iostream>
```

Componentes

- class **FiguraGeometrica**

figurageometrica.h

Vá para a documentação desse arquivo.

```
1 #ifndef FIGURAGEOMETRICA_H
2 #define FIGURAGEOMETRICA_H
3 #include "sculptor.h"
4 #include <iostream>
5
6 class FiguraGeometrica{
7 protected:
8   float r, g, b, a;
9 public:
10   FiguraGeometrica();
11   virtual ~FiguraGeometrica() {}
12   virtual void draw(Sculptor &s)=0;
13 };
14
15
16 #endif // FIGURAGEOMETRICA_H
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/main.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib>
#include <vector>
#include <cstring>
#include <sstream>
#include "sculptor.h"
#include "figurageometrica.h"
#include "putbox.h"
#include "cutbox.h"
#include "putvoxel.h"
#include "cutvoxel.h"
#include "putsphere.h"
#include "cutsphere.h"
#include "putellipsoid.h"
#include "cutellipsoid.h"
```

Funções

- int **main** ()

Funções

int main ()

```
21          {
22   Sculptor *s;
23   int nx, ny, nz;
24   int x0, x1, y0, y1, z0, z1;
25   float r, g, b, a;
26   std::vector<FiguraGeometrica*> f;
27   std::ifstream fin;
28   std::string str;
29
30   fin.open("toto.txt");
31
32   if(!fin.is_open()){
33     cout << "não abriu o arquivo\n";
34     std::exit(0);
35   }
36
37   // processa o arquivo enquanto houver linhas
38   // para ler
39   while(fin.good()){
40     fin >> str;
41     if(!fin.good()){
42       break;
43     }
44     if(str.compare("dim")==0){
45         // li do arquivo a linha dim 10 10 10
46         fin >> nx >> ny >> nz;
47         std::cout << nx << "x" << ny << "x" << nz;
48         std::cout << std::endl;
49         s  = new Sculptor(nx, ny, nz);
50     }
51
52     else if(str.compare("putbox")==0){
53         //  putbox 0 9 0 9 0 9 0 0 1.0 1.0
54         fin >> x0 >> x1 >> y0 >> y1 >> z0 >> z1;
```

```
55          fin >> r >> g >>  b >> a;
56          f.push back(new PutBox(x0, x1, y0, y1, z0, z1, r, g, b, a));
57      }
58
59      else if(str.compare("cutbox")==0){
60          fin >> x0 >> x1 >> y0 >> y1 >> z0 >> z1;
61          f.push_back(new CutBox(x0, x1, y0, y1, z0, z1));
62      }
63
64      else if(str.compare("putsphere") == 0) {
65          int xcenter, ycenter, zcenter, radius;
66          fin >> xcenter >> ycenter >> zcenter >> radius >> r >> g >> b >> a;
67          f.push_back(new PutSphere(xcenter, ycenter, zcenter, radius, r, g, b, a));
68      }
69
70      else if (str.compare("cutsphere") == 0) {
71          int xcenter, ycenter, zcenter, radius;
72          fin >> xcenter >> ycenter >> zcenter >> radius;
73          f.push_back(new CutSphere(xcenter, ycenter, zcenter, radius));
74      }
75
76      else if (str.compare("putellipsoid") == 0) {
77          int xcenter, ycenter, zcenter, rx, ry, rz;
78          fin >> xcenter >> ycenter >> zcenter >> rx >> ry >> rz >> r >> g >> b >>
a;
79          f.push back(new PutEllipsoid(xcenter, ycenter, zcenter, rx, ry, rz,r, g,
b, a));
80      }
81
82      else if (str.compare("cutellipsoid") == 0) {
83          int xcenter, ycenter, zcenter, rx, ry, rz;
84          fin >> xcenter >> ycenter >> zcenter >> rx >> ry >> rz;
85          f.push back(new CutEllipsoid(xcenter, ycenter, zcenter, rx, ry, rz));
86      }
87    }
88
89
90    for(size t i=0; i<f.size(); i++){
91      std::cout << "draw\n";
92      f[i]->draw(*s);
93    }
94    for(size_t i=0; i<f.size(); i++){
95      delete f[i];
96    }
97
98    s->writeOFF("testetoto.off");
99
100
101     return 0;
102 }
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/putbox.cpp

```
#include "putbox.h"
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/putbox.h

```
#include "figurageometrica.h"
```

Componentes

- class **PutBox**

putbox.h

Vá para a documentação desse arquivo.

```
1 #ifndef PUTBOX_H
2 #define PUTBOX_H
3
4 #include "figurageometrica.h"
5
6 class PutBox : public FiguraGeometrica{
7   int x0, x1, y0, y1, z0, z1;
8 public:
9   PutBox(int x0, int x1, int y0, int y1,
10          int z0, int z1, float r, float g,
11          float b, float a);
12   void draw(Sculptor &s);
13 };
14
15 #endif // PUTBOX_H
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/putellipsoid.cpp

```
#include "putellipsoid.h"
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/putellipsoid.h

```
#include "figurageometrica.h"
```

Componentes

- class **PutEllipsoid**

putellipsoid.h

Vá para a documentação desse arquivo.

```
1 #ifndef PUTELLIPSOID_H
2 #define PUTELLIPSOID_H
3
4 #include "figurageometrica.h"
5
6 class PutEllipsoid : public FiguraGeometrica{
7     protected:
8         int xcenter, ycenter, zcenter, rx, ry, rz;
9         float r, g, b, a;
10     public:
11         PutEllipsoid(int _xcenter, int _ycenter, int _zcenter, int _rx, int _ry, int
_rz, float _r,float _g,float _b,float _a);
12         ~PutEllipsoid();
13         void draw(Sculptor &t);
14 };
15 #endif
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/putsphere.cpp

```
#include "putsphere.h"
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/putsphere.h

```
#include "figurageometrica.h"
#include <iostream>
```

Componentes

- class **PutSphere**

Definições e Macros

- #define **PUSPHERE_H**

Definições e macros

#define PUSPHERE_H

putsphere.h

Vá para a documentação desse arquivo.

```
1 #ifndef PUTSPHRE_H
2 #define PUSPHERE_H
3 #include "figurageometrica.h"
4 #include <iostream>
5
6 class PutSphere : public FiguraGeometrica {
7 private:
8   int xcenter, ycenter, zcenter, radius;
9   float r, g, b, a;
10
11 public:
12   PutSphere(int xcenter, int ycenter, int zcenter, int radius, float  r, float  g, float
b, float a);
13   ~PutSphere();
14   void draw(Sculptor &s);
15 };
16
17 #endif
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/putvoxel.cpp

```
#include "putvoxel.h"
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/putvoxel.h

```
#include "figurageometrica.h"
#include <iostream>
```

Componentes

- class **PutVoxel**

putvoxel.h

Vá para a documentação desse arquivo.

```
1 #ifndef PUTVOXEL_H
2 #define PUTVOXEL_H
3
4 #include "figurageometrica.h"
5 #include <iostream>
6
7 class PutVoxel : public FiguraGeometrica{
8     protected:
9         int x, y, z;
10         float r, g, b, a;
11     public:
12         PutVoxel(int _x, int _y, int _z, float _r, float _g, float _b, float _a);
13         ~PutVoxel();
14         void draw(Sculptor &t);
15 };
16
17 #endif
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/sculptor.cpp

```
#include "sculptor.h"
#include <cmath>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <queue>
#include <vector>
```

Funções

- void **troca** (int &x, int &y)

Funções

void troca (int & *x*, int & *y*)

```
10                                  {
11   int tmp;
12   tmp = x;
13   x = y;
14   y = tmp;
15 }
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/sculptor.h

Componentes

- struct **Voxel**
  *O struct **Voxel** é responsável por armazenar as informações de cada voxel.*
- class **Sculptor**
  *A classe **Sculptor** é responsável por armazenar os voxels e realizar as operações de desenho.*

sculptor.h

Vá para a documentação desse arquivo.

```
1 #ifndef SCULPTOR_HPP
2 #define SCULPTOR_HPP
3
8 struct Voxel {
9   float r;
10   float g;
11   float b;
12   float a;
13   bool show;
14 };
15
23 class Sculptor {
24  private:
25   Voxel*** v;
26   int nx;
27   int ny;
28   int nz;             // Dimensions
29   float r, g, b, a;  // Current drawing color
30  public:
43   Sculptor(int  nx, int  ny, int  nz);
44   ~Sculptor();
45   void setColor(float r, float g, float b, float alpha);
46   void putVoxel(int x, int y, int z);
47   void cutVoxel(int x, int y, int z);
48   void putSphere(int xcenter, int ycenter, int zcenter, int radius);
49   void cutSphere(int xcenter, int ycenter, int zcenter, int radius);
64   void putEllipsoid(int xcenter, int ycenter, int zcenter, int rx, int ry,
65                     int rz);
66   void cutEllipsoid(int xcenter, int ycenter, int zcenter, int rx, int ry,
67                     int rz);
68   void writeOFF(const char* filename);
69 };
70
71 #endif
```

Referência do Arquivo D:/Documentos/Documents/UFRN/Programacao Avancada/PAprojeto3/voxel.h

Componentes

- struct **Voxel**
  *O struct **Voxel** é responsável por armazenar as informações de cada voxel.*

voxel.h

Vá para a documentação desse arquivo.

```
1 #ifndef VOXEL_H
2 #define VOXEL_H
3
4 struct Voxel {
12   float r, g, b;
16   float a;
20   bool isOn;
21 };
22
23 #endif // VOXEL_H
```

Sumário

INDEX