Projeto Escultor3D

AUTHOR

Versão 1.0

Domingo, 26 de Junho de 2022

# Sumário

Table of contents

# Índice dos Componentes

## Lista de Classes

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

**Sculptor**

**Voxel**

# Índice dos Arquivos

## Lista de Arquivos

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

**Projeto Escultor3D/main.cpp**

**Projeto Escultor3D/sculptor.cpp**

**Projeto Escultor3D/sculptor.h**

# Classes

## Referência da Classe Sculptor

#include <sculptor.h>

### Membros Públicos

1. **Sculptor** (int \_nx, int \_ny, int \_nz)
2. **~Sculptor** ()
3. void **setColor** (float r, float g, float b, float alpha)
4. void **putVoxel** (int x, int y, int z)
5. void **cutVoxel** (int x, int y, int z)
6. void **putBox** (int x0, int x1, int y0, int y1, int z0, int z1)
7. void **cutBox** (int x0, int x1, int y0, int y1, int z0, int z1)
8. void **putSphere** (int xcenter, int ycenter, int zcenter, int radius)
9. void **cutSphere** (int xcenter, int ycenter, int zcenter, int radius)
10. void **putEllipsoid** (int xcenter, int ycenter, int zcenter, int rx, int ry, int rz)
11. void **cutEllipsoid** (int xcenter, int ycenter, int zcenter, int rx, int ry, int rz)
12. void **writeOFF** (char \*filename)

### Construtores e Destrutores

#### Sculptor::Sculptor (int *\_nx*, int *\_ny*, int *\_nz*)

10 {

11 nx = nx1;

12 ny = ny1;

13 nz = nz1;

14 r = 0.5;

15 g = 0.5;

16 b = 0.5;

17 a = 1;

18

19 // Alocação dinâmica da matriz 3d

20 v = new Voxel\*\*[nx];

21 for (int i =0; i<nx; i++){

22 v[i] = new Voxel\*[ny];

23

24 for (int j =0; j<ny; j++){

25 v[i][j]= new Voxel[nz];

26 }

27 }

28 }

#### Sculptor::~Sculptor ()

30 {

31 for(int i = 0; i < nx; i++){

32 for (int j = 0; j < ny; j++){

33 delete[] v[i][j];

34 }

35 }

36 for (int i=0; i <nx; i++){

37 delete[] v[i];

38 }

39

40 delete[] v;

41

42 nx = 0;

43 ny = 0;

44 nz = 0;

45 };

### Funções membros

#### void Sculptor::cutBox (int *x0*, int *x1*, int *y0*, int *y1*, int *z0*, int *z1*)

80 {

81 for(int x = x0; x<x1; x++){

82 for (int y = y0; y<y1; y++){

83 for (int z = z0; z<z1; z++){

84 v[x][y][z].isOn = false;

85 }

86 }

87 }

88 }

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

134 {

135 double newx, newy, newz;

136

137 for (int x = 0; x < nx; x++){

138 for (int y = 0; y < ny; y++){

139 for (int z = 0; z < nz; z++){

140 newx = ((float)(x-xcenter)\*(float)(x-xcenter))/(rx \* rx);

141 newy = ((float)(y-ycenter)\*(float)(y-ycenter))/(ry \* ry);

142 newz = ((float)(z-zcenter)\*(float)(z-zcenter))/(rz \* rz);

143

144 if ((newx + newy + newz) < 1){

145 cutVoxel(x,y,z);

146 }

147 }

148 }

149 }

150 }

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

104 {

105 for(int i = -radius; i<radius; i++){

106 for(int j = -radius; j<radius; j++){

107 for(int k = -radius; k<radius; k++){

108 if((i\*i+j\*j+k\*k) < radius\*radius){

109 cutVoxel(i+xcenter,j+ycenter,k+zcenter);

110 }

111 }

112 }

113 }

114 }

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

61 {

62 v[x][y][z].isOn = false;

63 }

#### void Sculptor::putBox (int *x0*, int *x1*, int *y0*, int *y1*, int *z0*, int *z1*)

64 {

65 for(int x = x0\_; x<x1\_; x++){

66 for (int y = y0\_; y<y1\_; y++){

67 for (int z = z0\_; z<z1\_; z++){

68 v[x][y][z].isOn=true;

69 v[x][y][z].r = r;

70 v[x][y][z].g = g;

71 v[x][y][z].b = b;

72 v[x][y][z].a = a;

73 }

74 }

75 }

76

77 }

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

116 {

117 float newx, newy, newz;

118

119 for (int x = 0; x < nx; x++){

120 for (int y = 0; y < ny; y++){

121 for (int z = 0; z < nz; z++){

122 newx = ((float)(x-xcenter)\*(float)(x-xcenter))/(rx \* rx);

123 newy = ((float)(y-ycenter)\*(float)(y-ycenter))/(ry \* ry);

124 newz = ((float)(z-zcenter)\*(float)(z-zcenter))/(rz \* rz);

125

126 if ((newx + newy + newz) < 1){

127 putVoxel(x,y,z);

128 }

129 }

130 }

131 }

132 }

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

90 {

91 for(int i = -radius; i<=radius; i++){

92 for(int j = -radius; j<=radius; j++){

93 for(int k = -radius; k<=radius; k++){

94

95 if ((i\*i+j\*j+k\*k) < radius\*radius){

96 putVoxel(i+xcenter,j+ycenter,k+zcenter);

97 }

98 }

99 }

100 }

101

102 }

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

54 {

55 v[x][y][z].isOn = true;

56 v[x][y][z].r = r;

57 v[x][y][z].g = g;

58 v[x][y][z].b = b;

59 v[x][y][z].a = a;

60 }

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

48 {

49 this->r = r;

50 this->g = g;

51 this->b = b;

52 this->a = alpha;

53 }

#### void Sculptor::writeOFF (char \* *filename*)

152 {

153 int total, index;

154 ofstream arquivo;

155 total = 0;

156 arquivo.open(filename);

157 arquivo << "OFF\n";

158

159 for (int i=0; i < nx; i++){

160 for (int j = 0; j <ny; j++){

161 for (int k=0; k <nz; k++){

162 if(v[i][j][k].isOn == true){

163 total++;

164 }

165 }

166 }

167 }

168

169 arquivo << total \* 8 << " " << total \* 6 << " " << 0 << "\n";

170

171 for (int i=0; i < nx; i++){

172 for (int j = 0; j <ny; j++){

173 for (int k=0; k <nz; k++){

174 if(v[i][j][k].isOn == true){

175 arquivo << i - 0.5 << " " << j + 0.5 << " " << k - 0.5 << "\n" << flush;

176 arquivo << i - 0.5 << " " << j - 0.5 << " " << k - 0.5 << "\n" << flush;

177 arquivo << i + 0.5 << " " << j - 0.5 << " " << k - 0.5 << "\n" << flush;

178 arquivo << i + 0.5 << " " << j + 0.5 << " " << k - 0.5 << "\n" << flush;

179 arquivo << i - 0.5 << " " << j + 0.5 << " " << k + 0.5 << "\n" << flush;

180 arquivo << i - 0.5 << " " << j - 0.5 << " " << k + 0.5 << "\n" << flush;

181 arquivo << i + 0.5 << " " << j - 0.5 << " " << k + 0.5 << "\n" << flush;

182 arquivo << i + 0.5 << " " << j + 0.5 << " " << k + 0.5 << "\n" << flush;

183 }

184 }

185 }

186 }

187

188 total = 0;

189

190 for (int i=0; i < nx; i++){

191 for (int j = 0; j < ny; j++){

192 for (int k=0; k < nz; k++){

193 if(v[i][j][k].isOn == true){

194 index = total\*8;

195 arquivo << fixed;

196

197 arquivo << 4 << " " << index + 0 << " " << index + 3 << " " << index + 2 << " " << index + 1 << " ";

198 arquivo << setprecision(2) << v[i][j][k].r << " " << setprecision(2) << v[i][j][k].g << " " << setprecision(2) << v[i][j][k].b << " " <<setprecision(2) << v[i][j][k].a << setprecision(2) << "\n";

199

200 arquivo << 4 << " " << index + 4 << " " << index + 5 << " " << index + 6 << " " << index + 7 << " ";

201 arquivo << setprecision(2) << v[i][j][k].r << " " << setprecision(2) << v[i][j][k].g << " " << setprecision(2) << v[i][j][k].b << " " << setprecision(2) << v[i][j][k].a << setprecision(2) << "\n";

202

203 arquivo << 4 << " " << index + 0 << " " << index + 1 << " " << index + 5 << " " << index + 4 << " ";

204 arquivo << setprecision(2) << v[i][j][k].r << " " << setprecision(2) << v[i][j][k].g << " " << setprecision(2) << v[i][j][k].b << " " << setprecision(2) << v[i][j][k].a << setprecision(2) << "\n";

205

206 arquivo << 4 << " " << index + 0 << " " << index + 4 << " " << index + 7 << " " << index + 3 << " ";

207 arquivo << setprecision(2) << v[i][j][k].r << " " << setprecision(2) << v[i][j][k].g << " " << setprecision(2) << v[i][j][k].b << " " << setprecision(2) << v[i][j][k].a << setprecision(2) << "\n";

208

209 arquivo << 4 << " " << index + 7 << " " << index + 6 << " " << index + 2 << " " << index + 3 << " ";

210 arquivo << setprecision(2) << v[i][j][k].r << " " <<setprecision(2) << v[i][j][k].g << " " << setprecision(2) << v[i][j][k].b << " " <<setprecision(2) << v[i][j][k].a << setprecision(2) << "\n";

211

212 arquivo << 4 << " " << index + 1 << " " << index + 2 << " " << index + 6 << " " << index + 5 << " ";

213 arquivo << setprecision(2) << v[i][j][k].r << " " << setprecision(2) << v[i][j][k].g << " " << setprecision(2) << v[i][j][k].b << " " << setprecision(2) << v[i][j][k].a << setprecision(2) << "\n";

214

215 total++;

216 }

217 }

218 }

219 }

220

221 arquivo.close();

222 }

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

1. Projeto Escultor3D/**sculptor.h**
2. Projeto Escultor3D/**sculptor.cpp**

## Referência da Estrutura Voxel

#include <sculptor.h>

### Atributos Públicos

1. float **r**
2. float **g**
3. float **b**
4. float **a**
5. bool **isOn**

### Atributos

#### float Voxel::a

#### float Voxel::b

#### float Voxel::g

#### bool Voxel::isOn

#### float Voxel::r

#### A documentação para essa estrutura foi gerada a partir do seguinte arquivo:

1. Projeto Escultor3D/**sculptor.h**

# Arquivos

## Referência do Arquivo Projeto Escultor3D/main.cpp

#include <iostream>

#include "sculptor.h"

#include <fstream>

#include <cstdlib>

### Funções

1. int **main** ()

### Funções

#### int main ()

8 {

9 // cria um escultor cuja matriz tem 50x50x50 voxels

10 Sculptor sonic(50,50,50);

11

12 //cabeça

13 sonic.setColor(0,0.2,0.8,1.0); // azul

14 sonic.putBox(20,29,20,27,20,28);

15 sonic.cutBox(20,21,20,27,20,21);

16 sonic.cutBox(20,21,20,27,27,28);

17 sonic.cutBox(28,29,20,27,20,21);

18 sonic.cutBox(28,29,20,27,27,28);

19 sonic.cutBox(20,29,20,21,20,21);

20 sonic.cutBox(20,29,20,21,27,28);

21 sonic.cutBox(20,29,26,27,20,21);

22 sonic.cutBox(20,29,26,27,27,28);

23 sonic.cutBox(20,21,20,21,20,28);

24 sonic.cutBox(20,21,26,27,20,28);

25 sonic.cutBox(28,29,20,21,20,28);

26 sonic.cutBox(28,29,26,27,20,28);

27

28 //orelhas

29 sonic.setColor(1,0.8,0.6,1.0);//pele

30 sonic.putVoxel(20,26,25);

31 sonic.putVoxel(28,26,25);

32 sonic.setColor(0,0.2,0.8,1.0); // azul

33 sonic.putVoxel(19,26,25);

34 sonic.putVoxel(19,25,25);

35 sonic.putVoxel(19,27,25);

36 sonic.putVoxel(20,27,25);

37 sonic.putVoxel(21,27,25);

38 sonic.putVoxel(20,26,24);

39 sonic.putVoxel(29,26,25);

40 sonic.putVoxel(29,27,25);

41 sonic.putVoxel(27,27,25);

42 sonic.putVoxel(29,25,25);

43 sonic.putVoxel(28,27,25);

44 sonic.putVoxel(28,26,24);

45

46 //olhos

47 sonic.setColor(1.0,1.0,1.0,1.0);//branco

48 sonic.putVoxel(22,24,27);

49 sonic.putVoxel(22,23,27);

50 sonic.putVoxel(22,22,27);

51 sonic.putVoxel(23,24,27);

52 sonic.setColor(0.2,0.6,0.2,1.0);//verde

53 sonic.putVoxel(23,23,27);

54 sonic.setColor(0,0,0,1.0);//preto

55 sonic.putVoxel(23,22,27);

56 sonic.setColor(1.0,1.0,1.0,1.0);//branco

57 sonic.putVoxel(24,22,27);

58 sonic.putVoxel(25,24,27);

59 sonic.putVoxel(26,24,27);

60 sonic.putVoxel(26,23,27);

61 sonic.putVoxel(26,22,27);

62 sonic.setColor(0.2,0.6,0.2,1.0);//verde

63 sonic.putVoxel(25,23,27);

64 sonic.setColor(0,0,0,1.0);//preto

65 sonic.putVoxel(25,22,27);

66

67 //bochecha

68 sonic.setColor(1,0.8,0.6,1.0);//pele

69 sonic.putBox(21,28,20,22,27,29);

70

71 //nariz

72 sonic.setColor(0,0,0,1.0);//preto

73 sonic.putVoxel(24,21,29);

74 sonic.setColor(0.6,0,0,1.0);//vermelho

75 sonic.putVoxel(22,20,29);

76

77 //crista

78 sonic.setColor(0,0.2,0.8,1.0); // azul

79 sonic.putVoxel(24,27,23);

80 sonic.putVoxel(24,27,22);

81 sonic.putVoxel(24,27,21);

82 sonic.putVoxel(24,27,20);

83 sonic.putVoxel(24,27,19);

84 sonic.putVoxel(24,27,18);

85 sonic.putVoxel(24,27,17);

86 sonic.putVoxel(24,27,16);

87 sonic.putVoxel(24,27,15);

88 sonic.putVoxel(24,26,20);

89 sonic.putVoxel(24,26,19);

90 sonic.putVoxel(24,26,18);

91 sonic.putVoxel(24,26,17);

92 sonic.putVoxel(24,26,16);

93 sonic.putVoxel(24,25,17);

94 sonic.putVoxel(24,25,18);

95 sonic.putVoxel(24,25,19);

96 sonic.putVoxel(24,24,19);

97 sonic.putVoxel(24,24,18);

98 sonic.putVoxel(24,23,19);

99 sonic.putVoxel(24,22,19);

100 sonic.putVoxel(24,22,18);

101 sonic.putVoxel(24,22,17);

102 sonic.putVoxel(24,22,16);

103 sonic.putVoxel(24,22,15);

104 sonic.putVoxel(24,21,16);

105 sonic.putVoxel(24,21,17);

106 sonic.putVoxel(24,21,18);

107 sonic.putVoxel(24,21,19);

108 sonic.putVoxel(24,20,20);

109 sonic.putVoxel(24,20,19);

110 sonic.putVoxel(24,20,18);

111 sonic.putVoxel(24,20,17);

112 sonic.putVoxel(24,19,18);

113 sonic.putVoxel(24,19,19);

114 sonic.putVoxel(24,19,20);

115 sonic.putVoxel(24,19,21);

116 sonic.putVoxel(24,18,21);

117 sonic.putVoxel(24,17,21);

118 sonic.putVoxel(24,17,20);

119 sonic.putVoxel(24,17,19);

120 sonic.putVoxel(24,17,18);

121 sonic.putVoxel(24,17,17);

122 sonic.putVoxel(24,16,18);

123 sonic.putVoxel(24,16,19);

124 sonic.putVoxel(24,16,20);

125 sonic.putVoxel(24,16,21);

126 sonic.putVoxel(24,15,20);

127 sonic.putVoxel(24,15,21);

128 sonic.putVoxel(24,15,19);

129

130 //corpo

131 sonic.setColor(0,0.2,0.8,1.0); // azul

132 sonic.putBox(22,27,15,20,22,27);

133 sonic.setColor(1,0.8,0.6,1.0);//pele

134 sonic.putVoxel(23,18,26);

135 sonic.putVoxel(24,18,26);

136 sonic.putVoxel(25,18,26);

137 sonic.putVoxel(23,17,26);

138 sonic.putVoxel(24,17,26);

139 sonic.putVoxel(25,17,26);

140 sonic.putVoxel(23,16,26);

141 sonic.putVoxel(24,16,26);

142 sonic.putVoxel(25,16,26);

143

144 //braços

145 sonic.setColor(1,0.8,0.6,1.0);//pele

146 sonic.putBox(18,22,18,19,25,26);

147 sonic.putBox(18,19,18,19,25,29);

148 sonic.setColor(1.0,1.0,1.0,1.0);//branco

149 sonic.putBox(17,20,17,20,29,32);

150 sonic.setColor(1,0.8,0.6,1.0);//pele

151 sonic.putBox(27,28,18,19,21,25);

152 sonic.putBox(27,28,15,19,20,21);

153 sonic.setColor(1.0,1.0,1.0,1.0);//branco

154 sonic.putBox(26,29,12,15,19,22);

155

156 //pernas

157 sonic.setColor(0,0.2,0.8,1.0); // azul

158 sonic.putBox(26,27,15,16,26,30);

159 sonic.putBox(26,27,12,16,29,30);

160 sonic.setColor(1.0,1.0,1.0,1.0); // branco

161 sonic.putBox(25,28,11,12,28,31);

162 sonic.setColor(0.8,0.2,0,1.0); // vermelho

163 sonic.putBox(25,28,9,11,28,34);

164 sonic.setColor(1.0,1.0,1.0,1.0); // branco

165 sonic.putBox(25,28,10,11,32,33);

166 sonic.setColor(0,0.2,0.8,1.0); // azul

167 sonic.putBox(22,23,11,16,23,24);

168 sonic.putBox(22,23,11,12,20,24);

169 sonic.setColor(1.0,1.0,1.0,1.0); // branco

170 sonic.putBox(21,24,10,13,19,20);

171 sonic.setColor(0.8,0.2,0,1.0); // vermelho

172 sonic.putBox(21,24,7,13,17,19);

173 sonic.setColor(1.0,1.0,1.0,1.0); // branco

174 sonic.putBox(21,24,8,9,18,19);

175

176 //chao

177 sonic.setColor(0.2,0.4,0,1.0); // verde

178 sonic.putBox(15,35,6,7,15,35);

179 sonic.setColor(0.3,0.6,0,1.0); // verde2

180 sonic.putBox(15,35,6,7,15,18);

181 sonic.putBox(15,35,6,7,21,24);

182 sonic.putBox(15,35,6,7,27,30);

183 sonic.putBox(15,35,6,7,33,35);

184 sonic.setColor(0.4,0.2,0,1.0); // marrom

185 sonic.putBox(15,35,4,6,15,35);

186

187 //anel

188 sonic.setColor(1.0,0.8,0,1.0);

189 sonic.putSphere(24,20,35,5);

190 sonic.cutSphere(24,20,36,5);

191 sonic.cutSphere(24,20,33,4);

192

193

194

195

196 // grava a escultura digital no arquivo "sonic.off"

197 sonic.writeOFF("sonic.off");

198 }

## Referência do Arquivo Projeto Escultor3D/sculptor.cpp

#include <iostream>

#include "sculptor.h"

#include <fstream>

#include <cstdlib>

#include <iomanip>

## Referência do Arquivo Projeto Escultor3D/sculptor.h

### Componentes

1. struct **Voxel**
2. class **Sculptor**

## sculptor.h

Vá para a documentação desse arquivo.1 struct Voxel {

2 float r,g,b; // Cores

3 float a;

4 // Transparencia

5 bool isOn; // Incluido ou nao

6 };

7

8 class Sculptor {

9 private:

10 Voxel \*\*\*v;

11 // 3D matrix

12 int nx,ny,nz; // Dimensions

13 float r,g,b,a; // Current drawing color

14

15 public:

16 Sculptor(int \_nx, int \_ny, int \_nz);

17 ~Sculptor();

18 void setColor(float r, float g, float b, float alpha);

19 void putVoxel(int x, int y, int z);

20 void cutVoxel(int x, int y, int z);

21 void putBox(int x0, int x1, int y0, int y1, int z0, int z1);

22 void cutBox(int x0, int x1, int y0, int y1, int z0, int z1);

23 void putSphere(int xcenter, int ycenter, int zcenter, int radius);

24 void cutSphere(int xcenter, int ycenter, int zcenter, int radius);

25 void putEllipsoid(int xcenter, int ycenter, int zcenter, int rx, int ry, int rz);

26 void cutEllipsoid(int xcenter, int ycenter, int zcenter, int rx, int ry, int rz);

27 void writeOFF(char \*filename);

28 };

# Sumário

INDEX