Projeto Escultor 3D - Programação Orientada a Objetos

Professor: Agostinho de Medeiros Brito Junior
Discentes: Newton Leonardo Leite Filho, Camila Raquel Sena de Almeida

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
DEPARTAMENTO DE ENGENHARIA DA COMPUTAÇÃO E AUTOMAÇÃO
PROGRAMAÇÃO AVANÇADA - DCA 0803
TURMA: 02
SEMESTRE: 2021.2

9 de janeiro de 2022

Sumário

UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Objetivos Gerais

- A partir dos conhecimentos obtidos durantes as aulas de programação orientada a objetos, desenvolver um programa em C++ para modelar em um arquivo .off a escultura 3D de uma pirâmide;
- Explicar o desenvolvimento do código e demonstrar seu funcionamento;
- Mostrar o resultado final do projeto.

UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Introdução

- A fim de contruir o modelo virtual de uma pirâmide, utilizamos alocação dinâmica de memória para reservar espaço no disco rígido necessário para a construção da matriz tridimensional na qual modelamos a escultura 3D;
- Criamos uma classe para a manipulação de matrizes tridimensionais e nela incluímos métodos que nos permitiram construir uma pirâmide virtual e definir suas propriedades, como cor e transparência.
- Utilizamos fluxo de dados para inserir em um arquivo .off todas as informações da matriz tridimensional que nós manipulamos de modo que nosso software local de visualição 3D pudesse interpretar as informações contidas no arquivo e, a partir delas, construir nossa escultura 3D de uma pirâmide a mostrá-la na tela para o usuário.

UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Sumário

Classe Sculptor

```
1  #ifndef SCULPTOR_H
2  #define SCULPTOR_H
3  struct Voxel{
4    float r, g, b; // Cores
5    float a; // Transparencia
6    bool isOn; // Incluido ou nao
7  };
8  class Sculptor{
9  protected:
10   Voxel*** v; // 3D matrix
11   int nx, ny, nz; // Dimensões
12   float r, g, b, a; // Cores e transparencia
13 public:
14   Sculptor(int _nx, int _ny, int _nz);
15   ~Sculptor();
16   void setColor(float r, float g, float b, float alpha);
17   void putVoxel(int x, int y, int z);
18   void cutVoxel(int x, int y, int z);
19   void putBox(int x0, int x1, int y0, int y1, int z0, int z1);
20   void cutBox(int x0, int x1, int y0, int y1, int z0, int z1);
21   void putSphere(int xcenter, int ycenter, int zcenter, int radius);
22   void cutSphere(int xcenter, int ycenter, int zcenter, int radius);
23   void putEllipsoid(int xcenter, int ycenter, int zcenter, int rx, int ry, int rz);
24   void cutEllipsoid(int xcenter, int ycenter, int zcenter, int rx, int ry, int rz);
25   void writeOFF(char* filename);
26 };
27 #endif // SCULPTOR_H
```

RN
O RIO GRANDE DO NORTE

Sumário

UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Bibliotecas

```
1   #include <string>
2   #include <iostream>
3   #include <fstream>
4   #include <iomanip>
5   #include <cmath>
6   #include <vector>
7   #include <queue>
8   #include "Sculptor.hpp"
9
10  using namespace std;
```

Construtor de Classe Sculptor

```
12  Sculptor::Sculptor(int _nx, int _ny, int _nz) {
13    nx = _nx;
14    ny = _ny;
15    nz = _nz;
16    v = new Voxel **[nx]; // Alocação dinâmica de memória
17    for (int i = 0; i < nx; i++) {
18      v[i] = new Voxel *[ny];
19    }
20    for (int i = 0; i < nx; i++) {
21      for (int j = 0; j < ny; j++) {
22        v[i][j] = new Voxel[nz];
23
24      }
25    }
26    for (int i = 0; i < nx; i++) { // Laço de repetição para atribuir valor inicial para todos os voxels
27      for (int j = 0; j < ny; j++) {
28        for (int k = 0; k < nz; k++) {
29          v[i][j][k].isOn = false;
30          v[i][j][k].r = 0;
31          v[i][j][k].g = 0;
32          v[i][j][k].b = 0;
33          v[i][j][k].a = 0;
34        }
35      }
36    }
37  }
```

Destrutor de Classe Sculptor

```
39  Sculptor::~Sculptor() {
40
41    for (int i = 0; i < nx; i++) { // Liberando as memórias, análogo ao free() da linguagem C
42      for (int j = 0; j < ny; j++) {
43        delete[] v[i][j];
44      }
45    }
46    for (int i = 0; i < nx; i++) {
47      delete[] v[i];
48    }
49    delete[] v;
50  }
```

Método setColor

```
51  void Sculptor::setColor(float _r, float _g, float _b, float _a){
52      r = _r;
53      g = _g;
54      b = _b;
55      a = _a;
56  }
```

Métodos cutVoxel e putVoxel

```
57  void Sculptor::cutVoxel(int x0, int y0, int z0){
58      v[x0][y0][z0].isOn = false;
59  }
60  void Sculptor::putVoxel(int x0, int y0, int z0){
61      v[x0][y0][z0].isOn = true;
62      v[x0][y0][z0].r = r;
63      v[x0][y0][z0].g = g;
64      v[x0][y0][z0].b = b;
65      v[x0][y0][z0].a = a;
66  }
```

UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Métodos cutBox e putBox

```
67  void Sculptor::putBox(int x0, int x1, int y0, int y1, int z0, int z1){
68
69      for (int i = x0; i < x1; i++){
70          for (int j = y0; j < y1; j++){
71              for (int k = z0; k < z1; k++){
72
73      v[i][j][k].isOn = true;
74      v[i][j][k].r = r;
75      v[i][j][k].g = g;
76      v[i][j][k].b = b;
77      v[i][j][k].a = a;
78              }
79          }
80      }
81
82  }
83  void Sculptor::cutBox(int x0, int x1, int y0, int y1, int z0, int z1){
84      for (int i = x0; i < x1; i++){
85          for (int j = y0; j < y1; j++){
86              for (int k = z0; k < z1; k++){
87
88      v[i][j][k].isOn = false;
89
90              }
91          }
92      }
93  }
```

Método putEllipsoid

```
94  void Sculptor::putEllipsoid (int x0, int y0, int z0, int rx, int ry, int rz){
95      double dx, dy, dz;
96      for (int i = 0; i < nx; i++){
97          for (int j = 0; j < ny; j++){
98              for (int k = 0; k < nz; k++){
99              dx = ((double)(i-x0)*(double)(i-x0))/(rx*rx);
100             dy = ((double)(j-y0)*(double)(j-y0))/(ry*ry);
101             dz = ((double)(k-z0)*(double)(k-z0))/(rz*rz);
102
103             if ((dx + dy + dz) < 1){
104                 v[i][j][k].isOn = true;
105                 v[i][j][k].r = r;
106                 v[i][j][k].g = g;
107                 v[i][j][k].b = b;
108                 v[i][j][k].a = a;
109
110             }
111             }
112         }
113     }
114 }
```

Método cutEllipsoid

```
115  void Sculptor::cutEllipsoid (int x0, int y0, int z0, int rx, int ry, int rz){
116      double dx, dy, dz;
117      for (int i = 0; i < nx; i++){
118          for (int j = 0; j < ny; j++){
119              for (int k = 0; k < nz; k++){
120              dx = ((double)(i-x0)*(double)(i-x0))/(rx*rx);
121              dy = ((double)(j-y0)*(double)(j-y0))/(ry*ry);
122              dz = ((double)(k-z0)*(double)(k-z0))/(rz*rz);
123              if ((dx + dy + dz) < 1){
124                  v[i][j][k].isOn = false;
125
126              }
127              }
128          }
129      }
130  }
```

Método putSphere

```
131  void Sculptor::putSphere(int x0, int y0, int z0, int r){
132  double dx, dy, dz;
133  for (int i = 0; i < nx; i++){
134          for (int j = 0; j < ny; j++){
135              for (int k = 0; k < nz; k++){
136              dx = ((double)(i-x0)*(double)(i-x0));
137              dy = ((double)(j-y0)*(double)(j-y0));
138              dz = ((double)(k-z0)*(double)(k-z0));
139
140              if ((dx + dy + dz) < (r*r)){
141              v[i][j][k].isOn = true;
142              v[i][j][k].r = r;
143              v[i][j][k].g = g;
144              v[i][j][k].b = b;
145              v[i][j][k].a = a;
146
147              }
148              }
149          }
150  }
151  }
```

Método cutSphere

```
152  void Sculptor::cutSphere(int x0, int y0, int z0, int r){
153  double dx, dy, dz;
154  for (int i = 0; i < nx; i++){
155          for (int j = 0; j < ny; j++){
156              for (int k = 0; k < nz; k++){
157              dx = ((double)(i-x0)*(double)(i-x0));
158              dy = ((double)(j-y0)*(double)(j-y0));
159              dz = ((double)(k-z0)*(double)(k-z0));
160
161              if ((dx + dy + dz) < (r*r)){
162              v[i][j][k].isOn = false;
163
164              }
165              }
166          }
167  }
168  }
```

Arquivos OFF

```
OFF
NVertices  NFaces  NArestas
x[0]  y[0]  z[0]
...
x[NVertices-1]  y[NVertices-1]  z[NVertices-1]
Nv v[0] v[1] ... v[Nv-1]  r g b a
Nv v[0] v[1] ... v[Nv-1]  r g b a
...
Nv v[0] v[1] ... v[Nv-1]  r g b a
```

UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Método writeOFF

```
170  void Sculptor::writeOFF(char* filename) {
171    ofstream outFile;
172    outFile.open(filename);
173
174    int qntdVoxel = 0;
175
176    for (int i = 0; i < nx; i++) { // Conta quantos voxels tem na escultura
177      for (int j = 0; j < ny; j++) {
178        for (int k = 0; k < nz; k++) {
179          if (v[i][j][k].isOn) {
180            qntdVoxel++;
181          }
182        }
183      }
184    }
```

Método writeOFF

```
186  outFile << "OFF" << endl << qntdVoxel * 8 << " " << qntdVoxel * 6 << " " << 0 << endl; // Determina o número de vértices e
     faces da escultura
187
188
189  for (int i = 0; i < nx; i++) { // Determina as coordenadas centrais de cada voxel
190    for (int j = 0; j < ny; j++) {
191      for (int k = 0; k < nz; k++) {
192        if (v[i][j][k].isOn) {
193          outFile << i - 0.5 << " " << j + 0.5 << " " << k - 0.5 << endl;
194          outFile << i - 0.5 << " " << j - 0.5 << " " << k - 0.5 << endl;
195          outFile << i + 0.5 << " " << j - 0.5 << " " << k - 0.5 << endl;
196          outFile << i + 0.5 << " " << j + 0.5 << " " << k - 0.5 << endl;
197          outFile << i - 0.5 << " " << j + 0.5 << " " << k + 0.5 << endl;
198          outFile << i - 0.5 << " " << j - 0.5 << " " << k + 0.5 << endl;
199          outFile << i + 0.5 << " " << j - 0.5 << " " << k + 0.5 << endl;
200          outFile << i + 0.5 << " " << j + 0.5 << " " << k + 0.5 << endl;
201        }
202      }
203    }
204  }
```

UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Método writeOFF

```
206   int vertice = 0;
207   outFile << setiosflags(ios::fixed); // Fixa a saida das cores em decimal float <iomanip>
208
209   for (int i = 0; i < nx; i++) { // Atribuindo as faces através dos vértices
210     for (int j = 0; j < ny; j++) {
211       for (int k = 0; k < nz; k++) {
212         if (v[i][j][k].isOn) {
213           outFile << 4 << " " << vertice + 0 << " " << vertice + 3 << " " << vertice + 2 << " " << vertice + 1 << " " << v[i][j]
              [k].r << " " << v[i][j][k].g << " " << v[i][j][k].b << " " << v[i][j][k].a << endl;
214           outFile << 4 << " " << vertice + 4 << " " << vertice + 5 << " " << vertice + 6 << " " << vertice + 7 << " " << v[i][j]
              [k].r << " " << v[i][j][k].g << " " << v[i][j][k].b << " " << v[i][j][k].a << endl;
215           outFile << 4 << " " << vertice + 0 << " " << vertice + 1 << " " << vertice + 5 << " " << vertice + 4 << " " << v[i][j]
              [k].r << " " << v[i][j][k].g << " " << v[i][j][k].b << " " << v[i][j][k].a << endl;
216           outFile << 4 << " " << vertice + 0 << " " << vertice + 4 << " " << vertice + 7 << " " << vertice + 3 << " " << v[i][j]
              [k].r << " " << v[i][j][k].g << " " << v[i][j][k].b << " " << v[i][j][k].a << endl;
217           outFile << 4 << " " << vertice + 3 << " " << vertice + 7 << " " << vertice + 6 << " " << vertice + 2 << " " << v[i][j]
              [k].r << " " << v[i][j][k].g << " " << v[i][j][k].b << " " << v[i][j][k].a << endl;
218           outFile << 4 << " " << vertice + 1 << " " << vertice + 2 << " " << vertice + 6 << " " << vertice + 5 << " " << v[i][j]
              [k].r << " " << v[i][j][k].g << " " << v[i][j][k].b << " " << v[i][j][k].a << endl;
219           vertice = vertice + 8;
220         }
221       }
222     }
223   }
224   outFile.close();
```

UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Sumário

int main

```
1   #include "Sculptor.hpp"
2
3   int main() {
4     Sculptor piramideasteca(44,44,20);
5     //Cor Beje
6     piramideasteca.setColor(1.0, 0.93, 0.59, 1.0);
7     //Blocos
8     piramideasteca.putBox(2,42,2,42,0,1);
9     piramideasteca.putBox(2,42,2,42,1,2);
10    piramideasteca.putBox(4,40,4,40,2,3);
11    piramideasteca.putBox(4,40,4,40,3,4);
12    piramideasteca.putBox(6,38,6,38,4,5);
13    piramideasteca.putBox(6,38,6,38,5,6);
14    piramideasteca.putBox(8,36,8,36,6,7);
15    piramideasteca.putBox(8,36,8,36,7,8);
16    piramideasteca.putBox(10,34,10,34,8,9);
17    piramideasteca.putBox(10,34,10,34,9,10);
18    piramideasteca.putBox(12,32,12,32,10,11);
19    piramideasteca.putBox(12,32,12,32,11,12);
20    piramideasteca.putBox(14,30,14,30,12,13);
21    piramideasteca.putBox(14,30,14,30,13,14);
22    piramideasteca.putBox(16,28,16,28,14,15);
23    piramideasteca.putBox(16,28,16,28,15,16);
24    piramideasteca.putBox(18,26,18,26,16,17);
25    piramideasteca.putBox(18,26,18,26,17,18);
```

UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

int main

```
27  piramideasteca.setColor(1.0, 0.8, 0.6, 1.0); //mudando cor da escada e do topo
28  piramideasteca.putBox(20,24,20,24,18,20);
29
30  //Escadas
31  piramideasteca.putBox(0,2,20,24,0,2);
32  piramideasteca.cutBox(0,1,20,24,1,2);
33  piramideasteca.putBox(2,4,20,24,2,4);
34  piramideasteca.cutBox(2,3,20,24,3,4);
35  piramideasteca.putBox(4,6,20,24,4,6);
36  piramideasteca.cutBox(4,5,20,24,5,6);
37  piramideasteca.putBox(6,8,20,24,6,8);
38  piramideasteca.cutBox(6,7,20,24,7,8);
39  piramideasteca.putBox(8,10,20,24,8,10);
40  piramideasteca.cutBox(8,9,20,24,9,10);
41  piramideasteca.putBox(10,12,20,24,10,12);
42  piramideasteca.cutBox(10,11,20,24,11,12);
43  piramideasteca.putBox(12,14,20,24,12,14);
44  piramideasteca.cutBox(12,13,20,24,13,14);
45  piramideasteca.putBox(14,16,20,24,14,16);
46  piramideasteca.cutBox(14,15,20,24,15,16);
47  piramideasteca.putBox(16,18,20,24,16,18);
48  piramideasteca.cutBox(16,17,20,24,17,18);
```

UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

int main

```
50  piramideasteca.putBox(42,44,20,24,0,2);
51  piramideasteca.cutBox(43,44,20,24,1,2);
52  piramideasteca.putBox(40,42,20,24,2,4);
53  piramideasteca.cutBox(41,42,20,24,3,4);
54  piramideasteca.putBox(38,40,20,24,4,6);
55  piramideasteca.cutBox(39,40,20,24,5,6);
56  piramideasteca.putBox(36,38,20,24,6,8);
57  piramideasteca.cutBox(37,38,20,24,7,8);
58  piramideasteca.putBox(34,36,20,24,8,10);
59  piramideasteca.cutBox(35,36,20,24,9,10);
60  piramideasteca.putBox(32,34,20,24,10,12);
61  piramideasteca.cutBox(33,34,20,24,11,12);
62  piramideasteca.putBox(30,32,20,24,12,14);
63  piramideasteca.cutBox(31,32,20,24,13,14);
64  piramideasteca.putBox(28,30,20,24,14,16);
65  piramideasteca.cutBox(29,30,20,24,15,16);
66  piramideasteca.putBox(26,28,20,24,16,18);
67  piramideasteca.cutBox(27,28,20,24,17,18);
```

UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

int main

```
69  piramideasteca.putBox(20,24,0,2,0,2);
70  piramideasteca.cutBox(20,24,0,1,1,2);
71  piramideasteca.putBox(20,24,2,4,2,4);
72  piramideasteca.cutBox(20,24,2,3,3,4);
73  piramideasteca.putBox(20,24,4,6,4,6);
74  piramideasteca.cutBox(20,24,4,5,5,6);
75  piramideasteca.putBox(20,24,6,8,6,8);
76  piramideasteca.cutBox(20,24,6,7,7,8);
77  piramideasteca.putBox(20,24,8,10,8,10);
78  piramideasteca.cutBox(20,24,8,9,9,10);
79  piramideasteca.putBox(20,24,10,12,10,12);
80  piramideasteca.cutBox(20,24,10,11,11,12);
81  piramideasteca.putBox(20,24,12,14,12,14);
82  piramideasteca.cutBox(20,24,12,13,13,14);
83  piramideasteca.putBox(20,24,14,16,14,16);
84  piramideasteca.cutBox(20,24,14,15,15,16);
85  piramideasteca.putBox(20,24,16,18,16,18);
86  piramideasteca.cutBox(20,24,16,17,17,18);
```

UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

int main

```
88    piramideasteca.putBox(20,24,42,44,0,2);
89    piramideasteca.cutBox(20,24,43,44,1,2);
90    piramideasteca.putBox(20,24,40,42,2,4);
91    piramideasteca.cutBox(20,24,41,42,3,4);
92    piramideasteca.putBox(20,24,38,40,4,6);
93    piramideasteca.cutBox(20,24,39,40,5,6);
94    piramideasteca.putBox(20,24,36,38,6,8);
95    piramideasteca.cutBox(20,24,37,38,7,8);
96    piramideasteca.putBox(20,24,34,36,8,10);
97    piramideasteca.cutBox(20,24,35,36,9,10);
98    piramideasteca.putBox(20,24,32,34,10,12);
99    piramideasteca.cutBox(20,24,33,34,11,12);
100   piramideasteca.putBox(20,24,30,32,12,14);
101   piramideasteca.cutBox(20,24,31,32,13,14);
102   piramideasteca.putBox(20,24,28,30,14,16);
103   piramideasteca.cutBox(20,24,29,30,15,16);
104   piramideasteca.putBox(20,24,26,28,16,18);
105   piramideasteca.cutBox(20,24,27,28,17,18);
106
107   piramideasteca.writeOFF((char*)"piramideasteca.off");
108   return 0;
109 }
```

UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Sumário

Demonstração

https://replit.com/@newtonlfilho/ProjetoEscultor3D#main.cpp

Resultados

https://github.com/NewtonStealther13/Projeto_Escultor_3D

Referências

- Notas de aula, Professor Agostinho de Medeiros Brito Junior - DCA0803 – Programação Avançada – Departamento de Engenharia da Computação e Automação, UFRN – 2021.2

OBRIGADO!!!

UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE