

# **PRINCÍPIOS BÁSICOS DE USO DO COMPUTADOR NA VISUALIZAÇÃO DE ENTIDADES GEOMÉTRICAS EUCLIDEANAS NO ESPAÇO EM LABORATÓRIOS DO PROINFO/MEC**

Autor: Jorge Barros de Abreu - SEEDF  
E-mail: ficmatin10@gmail.com

## **Resumo**

Mostra uma forma de usar o computador como ferramenta pedagógica auxiliar na visualização de pontos, retas, planos, etc, através do software livre Geomview e da biblioteca livre de rotinas geométricas para programas de computador denominada OOGL. O artigo destina-se a professores do ensino básico e mostra como criar as entidades geométricas básicas e o uso do software para visualizar os resultados. Os computadores do PROINFO/MEC possuem o sistema operacional GNU/Linux/Debian compatível.

**Palavras-chave:** geometria espacial computador geomview OOGL

## **1 Introdução**

O Geomview e a OOGL foram desenvolvidos pelo “Geometry Center” da Universidade de Minnesota (EUA). Apesar de o “Geometry Center” ter sido desfeito, o geomview continuou e é uma excelente ferramenta de visualização de objetos tridimensionais. Todas as técnicas de criação de figuras aqui expostas podem ser reduzidas à montagem de figuras espaciais definindo-as como compostas por polígonos planos. Por exemplo, o tetraedro regular é composto por quatro triângulos equiláteros e o cone reto de folha simples pode ser visualizado no computador como sendo construído por 100 triângulos isósceles e um polígono regular de 100 lados na base. De outra forma, você pode definir um tetraedro mostrando somente os pontos dos 4 vértices, mas pode também mostrar além dos 4 pontos dos 4 vértices mais 4 pontos centrais de cada face ou mais três pontos em cada face ou mais 100 pontos em cada face. No computador seu tetraedro apareceria mais definido à medida que você aumentasse o número de pontos. Vale ressaltar que o geomview trata pontos como vértices e trás tudo para dentro do cubo unitário mas isso pode ser modificado.

## **2 Sistema de Coordenadas**

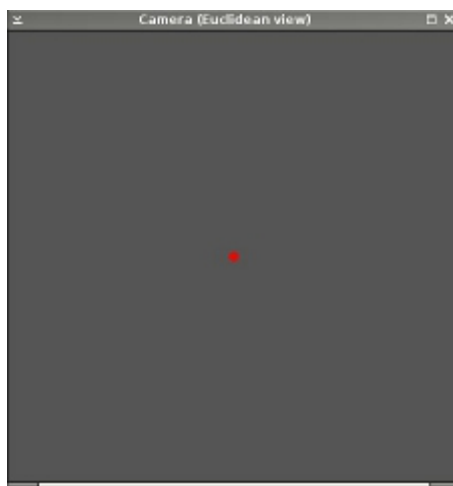
O geomview traz um arquivo chamado axis.list que mostra os eixos coordenados no espaço 3D de Euclides no intervalo  $[-1,1]$  para os três eixos.

## **3 Ponto no Espaço**

Para desenhar no espaço 3D de Euclides um ponto, crie o arquivo ponto.vect:

```
(progn
  (new-geometry "[Ponto]"
    appearance {
      linewidth 10 #controla o diametro do ponto
    }

    VECT
    1 1 1          # uma linha poligonal com um
                  # vertice e uma cor.
    1              # um vertice na linha poligonal
                  # por que estamos fazendo um ponto.
    1              # uma cor.
    0 0 0          # as coordenadas do ponto xyz.
    1 0 0 1        #a cor do ponto vermelho
  )
  (normalization "World" none)
)
```

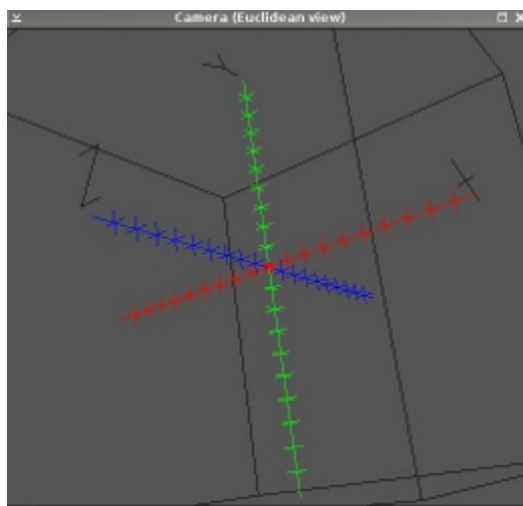


O comando para se obter a visualização acima é:

```
geomview -b 255 255 255 ponto.vect
```

### 3.0.1 Ponto nos Eixos

Para colocar o ponto acima no sistema de eixos coordenados localize o arquivo axis.list que acompanha o geomview, copie-o para o mesmo diretório do ponto.vect.



A imagem acima é obtida com:

```
geomview -b 255 255 255 ponto.vect axis.list
```

### 3.0.2 Três pontos

Crie o arquivo chamado “tres\_pontos.vect” com o seguinte conteúdo:

```
(progn
  (new-geometry "[Tres Pontos]"
    appearance {
      linewidth 10 #controla o diametro do ponto
    }
    VECT
    3 3 3          # ao todo no arquivo temos tres linhas
                  # poligonais com tr^es v^ertices e
                  # tr^es cores.
    1 1 1          # um v^ertice em cada linha poligonal por
                  # que estamos fazendo um ponto em cada
                  # poligonal.
    1 1 1          # uma cor para cada ``linha poligonal``
                  # que nesse caso ``\`e um ponto``.
    0 0 0.3        # as coordenadas do ponto 1 xyz.
    0 0.3 0        # as coordenadas do ponto 2 xyz.
    0.3 0 0        # as coordenadas do ponto 3 xyz.
    1 0 0 1        # a cor do ponto 1
    0 1 0 1        # a cor do ponto 2
    0 0 1 1        # a cor do ponto 3
  )
  (normalization "World" none)
  (bbox-draw "[Tres Pontos]" off)
)
```

### 3.0.3 Seis Pontos

Crie o arquivo chamado “seis\_pontos.vect” com o seguinte conteúdo:

```
(progn
  (new-geometry "[Seis Pontos]"
    appearance {
      linewidth 10 #controla o diametro do ponto
    }

    VECT
    6 6 6          # ao todo no arquivo temos seis linhas
                  # poligonais com seis v\'ertices e
                  # seis cores.
    1 1 1 1 1 1    # quantos vertices em cada linha
                  # poligonal.
    1 1 1 1 1 1    # uma cor para cada ``linha poligonal``
                  # que nesse caso ``\``e um ponto``.
    0 0 0.3        # as coordenadas xyz do ponto 1.
    0 0.3 0        # as coordenadas xyz do ponto 2.
    0.3 0 0        # as coordenadas xyz do ponto 3.
    0 0 -0.3       # as coordenadas xyz do ponto 4.
    0 -0.3 0       # as coordenadas xyz do ponto 5.
    -0.3 0 0       # as coordenadas xyz do ponto 6.
    1 0 0 1        # a cor do ponto 1
    0 1 0 1        # a cor do ponto 2
    0 0 1 1        # a cor do ponto 3
    1 1 0 1        # a cor do ponto 4
    1 1 0 1        # a cor do ponto 5
    0 1 1 1        # a cor do ponto 6
  )
  (normalization "World" none)
  (bbox-draw "[Seis Pontos]" off)
)
```

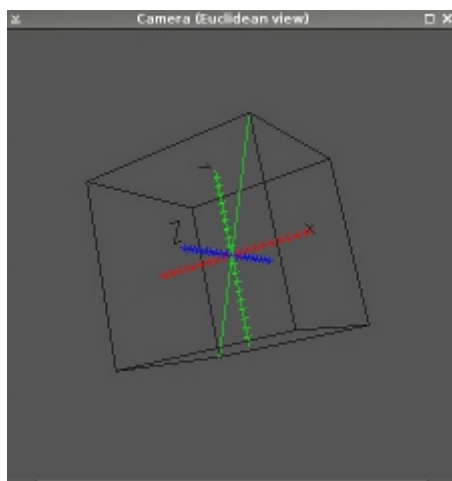
## 4 Segmento no Espaço

Para desenhar o segmento que vai de (-1,-1,-1) a (1,1,1) crie um arquivo chamado segmento.vect com o seguinte conteúdo:

```
VECT
1 2 1          # uma linha com dois vertices e uma cor.
2              # dois vertices extremidades do segmento
1              # uma cor
-1 -1 -1 1 1 1 # as coordenadas do inicio e fim
0 1 0 1        # a cor
```

Para colocar o segmento acima no nosso sistema de eixos use:

```
geomview -b 255 255 255 segmento.vect axis.list
```



Na prática essa forma de desenhar segmento acaba sendo a mesma para se desenhar retas. Agora divirta-se com o seguinte comando:

```
geomview -b 255 255 255 segmento.vect tres_pontos.vect axis.list
```

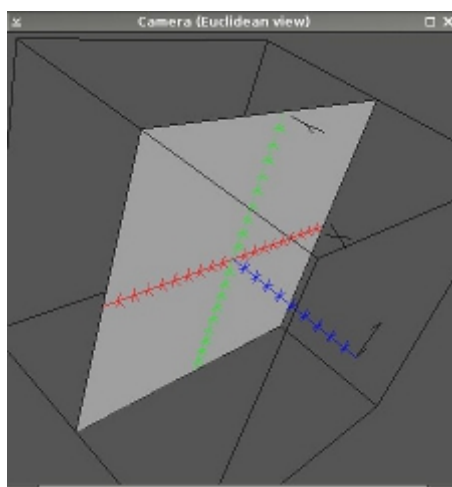
Novamente use o seguinte comando:

```
geomview -b 255 255 255 segmento.vect seis_pontos.vect axis.list
```

## 5 Planos

Um plano pode ser representado por um polígono de quatro lados. Rigorosamente falando precisamos apenas de três pontos para definir um plano mas por razões estéticas usaremos os quatro para o polígono de quatro lados.

A figura abaixo foi conseguida com “geomview quadrado.quad axis.list” e pode ser usada para representar o plano xy:



O arquivo quadrado.quad acompanha o geomview com o nome original de “square.quad”. Agora veja o que ocorre com (digite em uma linha no terminal):

```
geomview -b 255 255 255 segmento.vect seis_pontos.vect
axis.list quadrado.quad
```

Vamos exibir os três planos coordenados xy, yz e xz. O comando “geomview quadrado.quad axis.list” exibe o plano xy e o eixo tridimensional. Precisamos construir o plano yz e vamos fazer isso usando como base o quadrado.quad. Copie o arquivo quadrado.quad com o nome xy.quad e novamente com o nome de yz.quad. Altere o arquivo yz.quad de modo que ele fique da seguinte forma:

```
QUAD
0 -1 -1
0 -1 1
0 1 1
0 1 -1
```

Crie o arquivo xz.quad:

```
QUAD
1 0 1
1 0 -1
-1 0 -1
-1 0 1
```

O arquivo xy.quad está assim:

```
QUAD
-1 -1 0
1 -1 0
1 1 0
-1 1 0
```

Agora veja o que ocorre com (digite em uma linha no terminal):

```
geomview -b 255 255 255 axis.list xy.quad yz.quad xz.quad
```

Desenhar no geomview o plano que passa pelos pontos (0,0,0), (-1,-1,-1) e forma um ângulo de -45 graus com o plano xz. Grave o arquivo com o nome de 45\_0.quad.

```
QUAD
-1 1 -1
-1 0 1
1 -1 1
1 0 -1
```

A equação geral do plano representado acima é:

$$x + 2y + z = 0 \quad (1)$$

Desenhar no geomview o plano que passa pelos pontos (0,0,0), (-1,-1,-1) e é perpendicular ao plano anterior. Grave o arquivo com o nome de 45\_1.quad.

```
QUAD
-1 1 -1
1 1 1
1 -1 1
-1 -1 -1
```

A equação geral do plano representado acima é:

$$x - 2y + z = 0 \quad (2)$$

Para visualizar os dois planos acima nos eixos coordenados use:

```
geomview -b 255 255 255 axis.list 45_0.quad 45_1.quad
```

Repare que a intersecção entre esses dois planos forma uma reta, o que nos leva ao próximo tópico.

## 6 Reta no Espaço

A reta no espaço pode ser entendida como o eixo de um feixe de (ao menos dois) planos. Como no feixe tem infinitos planos pode-se dizer que existem infinitas maneiras de se representar algébricamente uma reta no espaço usando equações de planos.

### 6.1 Brincando de Construir Retas

Para auxiliar na montagem das retas e ressaltar o aspecto pedagógico de que uma reta é constituída por um conjunto de pontos foi desenvolvido um programa que desenha uma reta partindo de dois pontos dados no formato geomview. Copie o código fonte para o seu Linux Educacional e, supondo que você tenha chamado o arquivo de `reta.c`, compile-o com o seguinte comando:

```
gcc -Wall -pedantic -std=c99 reta.c -o reta
```

Será criado um arquivo chamado `reta` e que deve ser chamado partir do terminal:

```
reta
```

Apareceram perguntas que após serem respondidas com os números de sua preferência teremos, mais ou menos, o seguinte:

1. Informe a coordenada x do primeiro ponto de r: .5
2. Informe a coordenada y do primeiro ponto de r: .5
3. Informe a coordenada z do primeiro ponto de r: .5
4. Informe a coordenada x do segundo ponto de r: .6
5. Informe a coordenada y do segundo ponto de r: .6
6. Informe a coordenada z do segundo ponto de r: .6
7. Informe quantos pontos deseja que sejam calculados: 20
8. Informe o tamanho de cada ponto: 10
9. Informe o valor de R do RGB para a cor: 1
10. Informe o valor de G do RGB para a cor: 1
11. Informe o valor de B do RGB para a cor: 1
12. O primeiro ponto de r e' (0.500000, 0.500000, 0.500000).
13. O segundo ponto de r e' (0.600000, 0.600000, 0.600000).





```

1.000000 1.000000 1.000000 1
1.000000 1.000000 1.000000 1
1.000000 1.000000 1.000000 1
1.000000 1.000000 1.000000 1
1.000000 1.000000 1.000000 1
1.000000 1.000000 1.000000 1
1.000000 1.000000 1.000000 1
1.000000 1.000000 1.000000 1
1.000000 1.000000 1.000000 1
1.000000 1.000000 1.000000 1
1.000000 1.000000 1.000000 1
}

```

- A aparência fica melhor se forem especificados coordenadas de ponto no intervalo fechado  $[-1,1]$ .
- Os três valores de RGB devem estar no intervalo fechado  $[0,1]$ .
- O tamanho 10 de cada ponto é um tamanho razoável.
- A cada execução do programa é gerado um `saida.list`.
- Para desenhar duas retas execute o programa para a primeira reta e renomeie o “`saida.list`” para “`saida1.list`” por exemplo. Execute novamente o programa para desenhar a segunda reta.
- Para visualizar as duas retas desenhadas em “`saida.list`” e “`saida1.list`” no eixo cartesiano use o seguinte comando: `geomview saida.list saida1.list axis.list`.

## 7 Instalando o Software

Caso o `geomview` não esteja instalado no seu LE 4.0 proceda da seguinte forma (como usuário professor e com acesso ‘a internet):

- clique sobre o ícone LE no canto inferior esquerdo da tela do computador do laboratório do PROINFO,
- clique sobre a palavra Sistema,
- clique sobre Terminal Konsole,
- digite “`sudo apt-get install geomview`”.

O código fonte da documentação em português encontra-se no arquivo `geomview-pt_BR.texi` em:

<http://geomview.cvs.sourceforge.net/viewvc/geomview/geomview/doc/>

Algumas perguntas frequentes em:

<http://sourceforge.net/projects/geometriaptbr/files/?source=navbar>

Se for desejável ter a versão mais atual vá em:

<http://www.geomview.org/download/>

## 8 Outros Programas

- <http://www.jasoncantarella.com/webpage/index.php?title=Vecttools>
- <http://hendrix.imm.dtu.dk/software/polyr/code/polyr.tar.gz>  
<http://hendrix.imm.dtu.dk/software/polyr/doc/polyrdoc.ps.gz>
- <http://tetgen.berlios.de/>

## 9 Códigos Fonte

Listagem 1: (Arquivo reta6sbm.c) Construindo Retas.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #define          SIZE      256
4 #define          SAIDA      "saida.list"
5 int main ()
6 {
7     /* Declaracao de Variaveis */
8     int i=0;
9     char entrada[SIZE];
10    long double x0=0, y0=0, z0=0, x1=0, y1=0, z1=0, x=0,y=0,z
        =0,t=0;
11    long double quantidade=20, tamanho=10,R=0,G=0,B=0;
12    unsigned int ch=0;
13    FILE *saida;
14    printf ("Informe a coordenada x do primeiro ponto de r1: ")
        ;
15    fgets(&entrada[SIZE], SIZE, stdin);
16    x0=atof(&entrada[SIZE]);
17    printf ("Informe a coordenada y do primeiro ponto de r1: ")
        ;
18    fgets(&entrada[SIZE], SIZE, stdin);
19    y0=atof(&entrada[SIZE]);
20    printf ("Informe a coordenada z do primeiro ponto de r1: ")
        ;
21    fgets(&entrada[SIZE], SIZE, stdin);
22    z0=atof(&entrada[SIZE]);
23    printf ("Informe a coordenada x do segundo ponto de r1: ");
24    fgets(&entrada[SIZE], SIZE, stdin);
25    x1=atof(&entrada[SIZE]);
26    printf ("Informe a coordenada y do segundo ponto de r1: ");
27    fgets(&entrada[SIZE], SIZE, stdin);
28    y1=atof(&entrada[SIZE]);
29    printf ("Informe a coordenada z do segundo ponto de r1: ");
30    fgets(&entrada[SIZE], SIZE, stdin);
31    z1=atof(&entrada[SIZE]);

```

```

32 printf ("Informe quantos pontos deseja que sejam calculados
    : ");
33 fgets(&entrada[SIZE], SIZE, stdin);
34 quantidade=atof(&entrada[SIZE]);
35 printf ("Informe o tamanho de cada ponto: ");
36 fgets(&entrada[SIZE], SIZE, stdin);
37 tamanho=atof(&entrada[SIZE]);
38 printf ("Informe o valor de R do RGB para a cor: ");
39 fgets(&entrada[SIZE], SIZE, stdin);
40 R=atof(&entrada[SIZE]);
41 printf ("Informe o valor de G do RGB para a cor: ");
42 fgets(&entrada[SIZE], SIZE, stdin);
43 G=atof(&entrada[SIZE]);
44 printf ("Informe o valor de B do RGB para a cor: ");
45 fgets(&entrada[SIZE], SIZE, stdin);
46 B=atof(&entrada[SIZE]);
47 printf("O primeiro ponto de r1 e' (%Lf, %Lf, %Lf).\n",x0,y0,
    ,z0);
48 printf("O segundo ponto de r1 e' (%Lf, %Lf, %Lf).\n",x1,y1,
    ,z1);
49 printf("A quantidade de pontos solicitada e' %Lf.\n",
    quantidade);
50 printf("O tamanho de cada ponto e' %Lf.\n",tamanho);
51 printf("A cor de cada ponto e' (%Lf, %Lf, %Lf).\n",R,G,B);
52 if ((saida=fopen(SAIDA, "w")) == NULL)
53     {
54         perror("Desculpe , mas nao consigo criar
            saida.txt");
55         return(1);
56     }
57 fputs("    {\n    appearance {\n        linewidth ",saida);
58 sprintf(entrada, "%Lf", tamanho);//converte de Lf para char
59 while ((ch=entrada[i]) != '.' )
60 {
61     putc(ch,saida);//copia somente ate o ponto decimal
62     i++;
63 }
64 fputs("\n    }\n\n    VECT\n    ",saida);
65 ///poligonais de um ponto, vertices, cores
66 sprintf(entrada, "%Lf", quantidade);//converte de Lf para
    char
67 i=0;
68 while ((ch=entrada[i]) != '.' )//poligonais
69 {
70     putc(ch,saida);//copia somente ate o ponto decimal
71     i++;
72 }
73 putc(' ',saida);

```

```

74  sprintf(entrada , "%Lf" , quantidade); //converte de Lf para
    char
75  i=0;
76  while ((ch=entrada[i]) != '.' ) //vertices
77  {
78  putc(ch,saida); //copia somente ate o ponto decimal
79  i++;
80  }
81  putc(' ',saida);
82  sprintf(entrada , "%Lf" , quantidade); //converte de Lf para
    char
83  i=0;
84  while ((ch=entrada[i]) != '.' ) //cores
85  {
86      putc(ch,saida); //copia somente ate o ponto decimal
87      i++;
88  }
89  fputs("\n    ",saida);
90  //quantidade de vertices em cada poligonal
91  for (i=1; i<quantidade; i++)
92  {
93      putc('1',saida);
94      putc(' ',saida);
95  }
96  putc('1',saida);
97  fputs("\n    ",saida);
98  //quantidade de cores em cada poligonal
99  for (i=1; i<quantidade; i++)
100  {
101      putc('1',saida);
102      putc(' ',saida);
103  }
104  putc('1',saida);
105  putc('\n',saida);
106  //calculo dos pontos
107  for (t=-quantidade/2; t<quantidade/2; t++)
108  {
109      x=(x0+(x1-x0)*t);
110      y=(x0+(y1-y0)*t);
111      z=(x0+(z1-z0)*t);
112      fputs("    ",saida);
113      sprintf(entrada , "%Lf" , x);
114      fputs(entrada , saida);
115      fputs(" ",saida);
116      sprintf(entrada , "%Lf" , y);
117      fputs(entrada , saida);
118      fputs(" ",saida);
119      sprintf(entrada , "%Lf" , z);

```

```

120         fputs(entrada , saida);
121         fputs("\n", saida);
122     }
123     //a cor
124     for (i=1; i<=quantidade; i++)
125     {
126         fputs("    ", saida);
127         sprintf(entrada , "%Lf %Lf %Lf " , R,G,B);
128         fputs(entrada , saida);
129         fputs("l", saida);
130         if ( i==quantidade ) break;
131         fputs("\n", saida);
132     }
133     fputs("\n    }\n", saida);
134     fclose(saida);
135     printf("A saida encontra-se em %s.\n",SAIDA);
136     return(0);
137 }

```

## Sumário

<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Sistema de Coordenadas</b>	<b>1</b>
<b>3</b>	<b>Ponto no Espaço</b>	<b>1</b>
3.0.1	Ponto nos Eixos . . . . .	2
3.0.2	Três pontos . . . . .	3
3.0.3	Seis Pontos . . . . .	4
<b>4</b>	<b>Segmento no Espaço</b>	<b>4</b>
<b>5</b>	<b>Planos</b>	<b>5</b>
<b>6</b>	<b>Reta no Espaço</b>	<b>7</b>
6.1	Brincando de Construir Retas . . . . .	7
<b>7</b>	<b>Instalando o Software</b>	<b>9</b>
<b>8</b>	<b>Outros Programas</b>	<b>10</b>
<b>9</b>	<b>Códigos Fonte</b>	<b>10</b>

## Referências

- [lehmann(1991)] Lehmann, Charles H. *Geometria Analítica*. Tradução de Ruy Pinto da Silva Sieczkowski. 7<sup>a</sup> ed. São Paulo: Globo, 1991.
- [manual(2016)] Phillips, Mark et al. *Manual do Geomview*. Tradução de Jorge Barros de Abreu.