Vizualização de Objetos Fornecidos em Coordenadas do Mundo em um Plano Projetivo Baseado em Perspectiva Cônica/Paralela

Gabriel P. Belini[1], **Thiago Bucalão**[2]
Professor: Dante Alves Medeiros Filho, Dr.

1

Resumo. *Nesse trabalho iremos implementar em linguagem Python um software interativo que quando munido dos seguintes dados: Ponto de vista, três pontos não colineares, número de vértices do objeto, número de superfícies do objeto, número de vértices por superfície do objeto, vértices do objeto no World Coordinate System (WCS) e ordem com a qual os vértices devem ser ligados para a formação da imagem, consegue mostrar ao usuário as projeções Cônica e Paralela desse objeto em relação ao plano fornecido.*

1. Introdução

Nesta sessão iremos explicar os passos mais relevantes para a formação das imagens produzidas pelo software.

1.1. Formação do Plano

O plano de projeção é necessário para a formação das imagens perspectivas, ele é fornecido pelo usuário através de três pontos não colineares, *P1, P2, P3*, cada um com coordenadas *(x, y, z)*. A partir desses pontos o software calcula dois vetores paralelos ao plano $(vetor_1 \, e \, vetor_2)$ que são utilizados para achar o vetor *n* normal ao plano por meio do produto vetorial entre esses dois vetores.

1.2. Encontrando *d*

Para encontrar o valor da variável *d*, $d = d_0 - d_1$, que será utilizado nas matrizes perspectivas devemos primeiro calcular d_0 através da fórmula $d_0 = x_0.n_x + y_0.n_y + z_0.n_z$ e em seguida d_1, utilizando $d_1 = a.n_x + b.n_y + c.n_z$.

1.3. Matrizes Perspectivas

Munido dos dados necessários o software monta as matrizes perspectivas Paralela e Cônica para que seja possível obter as coordenadas no plano de projeção desejado. As matrizes perspectivas cônica e paralela são respectivamente exemplificadas abaixo:

$$\begin{bmatrix} d+anx & any & anz & -ad_0 \\ bnx & d+bny & bnz & -bd_0 \\ cnx & cny & d+cnz & -cd_0 \\ nx & ny & nz & -d_1 \end{bmatrix} \begin{bmatrix} d+anx & -any & -anz & ad_0 \\ bnx & d+bny & bnz & bd_0 \\ cnx & cny & d-cnz & cd_0 \\ 0 & 0 & 0 & d_1 \end{bmatrix}$$

1.4. Projeção em Coordenadas Homogêneas, Cartesianas e do Plano

Fazendo $P' = M_{per}.P$ obtemos as coordenadas homogêneas do objeto para o plano de projeção referente a matriz utilizada nos cálculos. Partindo das coordenadas homogêneas podemos em seguida obter as coordenadas cartesianas dividindo todos os elementos de cada coluna da matriz por seu respectivo *w*, dessa forma a linha 4 da matriz será então formada apenas por 1's.

Para a obtenção das coordenadas do plano basta criarmos uma outra matriz contendo apenas as duas primeiras linhas da matriz de coordenadas cartesianas, ou seja, basta deletar as linhas *z* e *w* da matriz de coordenadas cartesianas.

1.5. Projeção da Imagem no Dispositivo

Com todos os dados necessários podemos então transformar as coordenadas obtidas em coordenadas do dispositivo, fizemos isso utilizando a biblioteca do *Python* chamada *matplotlib*

2. Detalhes da Implementação

A linguagem escolhida foi o *Python* pois é a linguagem que possuímos maior domínio e também acreditamos que a mesma facilita a implementação devido a algumas características específicas como a facilidade para utilização de Listas, e diversas outras estruturas de dados.

O código foi dividido em diversas funções que calculam de forma isolada os diferentes valores utilizados na formação das imagens, isso permite que o código produzido nesse trabalho possa ser reutilizado em alguma outra aplicação, além disso todas as funções possuem nomes intuitivos e uma breve descrição antes de cada declaração explicando de forma sucinta e direta o que cada função faz, isso permite que pessoas que mesmo não tendo domínio na linguagem *Python* entendam o que foi feito e como foi feito.

A parte de Interção Humano Computador (IHC) foi feita utilizando a biblioteca *Tkinter*, a qual é amplamente utilizada por programadores *Python* e possui muito material disponível na *internet* o que facilita o aprendizado e utilização. O tipo de *layout* utilizado foi o *grid layout* o qual posiciona os objetos no *frame* tratando-o como se fosse uma matriz. A figura da página seguinte mostra como é a interface final do programa:

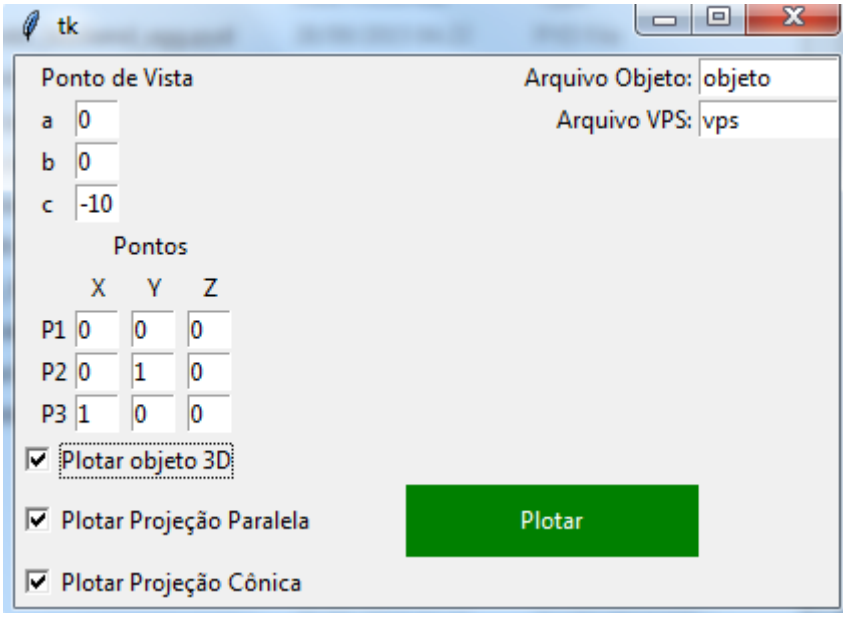

Figure 1. Tela principal do programa.

Nela podemos ver os campos para as coordenadas *(a, b, c)* do ponto de vista, os campos para as coordenadas *(x, y, z)* dos pontos que definem o plano, dois *text inputs* no canto superior direito, sendo que o de cima pede o nome do arquivo contendo os detalhes do objeto tais como vértices, número de vértices, etc. E o de segundo pede o nome do arquivo que contém a ordem com a qual os vértices devem ser ligados. No canto inferior esquerdo temos três checkboxes que quando marcados possuem um valor específico de retorno para cada um, o que possibilita selecionar quais projeções devem ser mostradas para o usuário dependendo de quais checkboxes estão selecionados.

Para a vizualização das imagens no dispositivo escolhemos a biblioteca *matplotlib*, essa escolha foi feita pois nunca haviamos utilizado bibliotecas que fizessem o que era pedido no trabalho e após algumas pesquisas na *internet* pareceu ser propícia para a solução do nosso problema.

3. Como Utilizar o Software

O software necessita de dois arquivos previamente preenchidos e fornecidos pelo usuário, esses arquivos devem estar em formato .txt e ter a seguinte forma:

Arquivo do Objeto:

```
8       #Número de vértices (NV)
0 0 0   #Coordenadas do vértice 1
1 0 0   .
1 1 0   .
0 1 0   .
0 1 1   .
0 0 1   .
1 0 1   .
1 1 1   #Coordenadas do vértice N
6       #Número de superfícies (NS)
4       #Número de vértices por superfície (NVPS)
4       .
4       .
4       .
4       .
4       #Número de vértices por superfície (NVPS)
```

Figure 2. Formato do Arquivo do Objeto, esse arquivo deve ter esse formato porém sem as descrições de cada linha contidas nesse exemplo.

Arquivo de Vértices por Superfície:

```
6 7 8 5 6 #Sequência de vértices para superfície 1
7 2 3 8 7 #Sequência de vértices para superfície 2
2 1 4 3 2 .
6 1 4 5 6 .
2 7 6 1 2 .
8 3 4 5 8 #Sequência de vértices para superfície N
```

Figure 3. Formato do Arquivo de Vértices por Superfície, nele cada número indica um respectivo vértice, a ordem dos números deve ser de acordo com o arquivo contendo as coordenadas dos vértices, primeiro vértice = vértice 1 e assim por diante.

Para facilitar a execução do software um arquivo *.exe* foi criado e será anexado junto deste relatório, para executá-lo basta clicar duas vezes no arquivo $CG2015\_IHC\_final.exe$, arquivos .txt com descrição do objeto e vértices por superfície são fornecidos com os respectivos nomes: objeto.txt e vps.txt, para testar o programa uma instância válida é apresentada na figura abaixo:

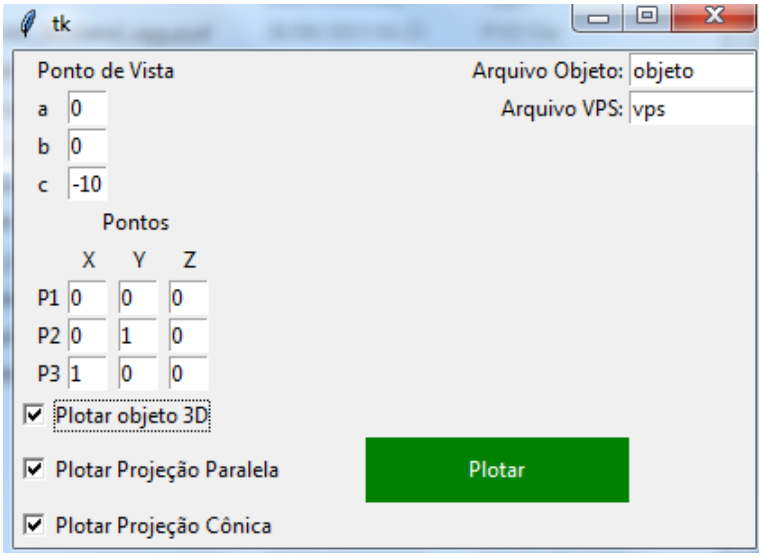

Figure 4. Exemplo de inputs válidos para execução do programa.

Os *outputs* gerados através das entradas mostradas acima são mostrados na figura abaixo:

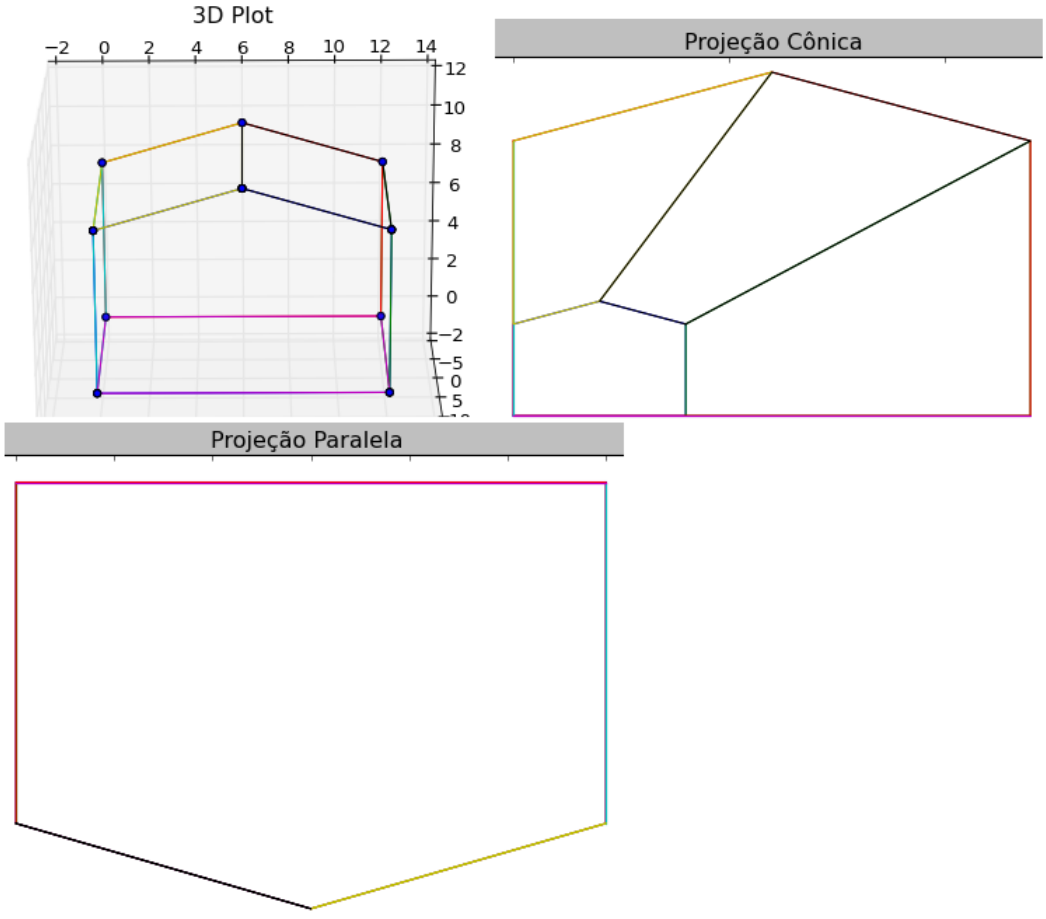

Figure 5. Imagens geradas a partir da entrada de teste citada acima.

O código fonte também é fornecido e para executá-lo basta ter *Python 3* instalado no computador e instalar as bibliotecas *numpy* e *matplotlib*.

References

[1] Tutorial Tkinter, disponível em, https://www.youtube.com/watch?v=RJB1Ek2Ko_Ylist=PL6gx4

[2] Documentação da biblioteca matplotlib, disponível em, http://matplotlib.org/.

[3] MEDEIROS FILHO, D. A., "Notas de Aula CG-08-PROJEÇÕES", 2015.