



Protótipo de Ferramenta/Plug-in  
para geração de imagens 3D a partir  
de imagens raster 2D em Grayscale  
para o Photoshop®



Ricardo Fachini

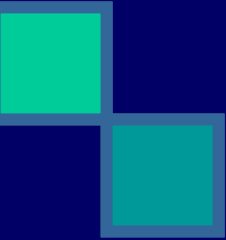

**Formando**

Antônio Carlos Tavares

**Orientador**




# Roteiro

- 
- Introdução
  - Photoshop
  - Módulos de Plug-in
  - OpenGL
  - Desenvolvimento
  - Conclusões e Extensões
- 

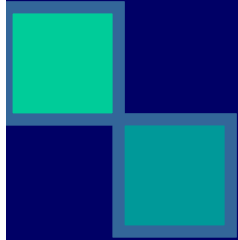



# Introdução

- Mercado Atual
  - Motivação
    - Novidade
    - Crescimento Intelectual
    - Desafio
  - Objetivos do Trabalho
    - Geração Imagens 3D no Photoshop a partir de Imagens 2D em Grayscale
- 

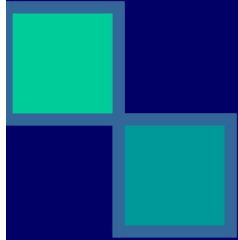



# Photoshop

- 
- O Photoshop
  - Quando e quem utiliza o Photoshop
  - Arquivos do Photoshop
    - Formatos
    - Layers
- 

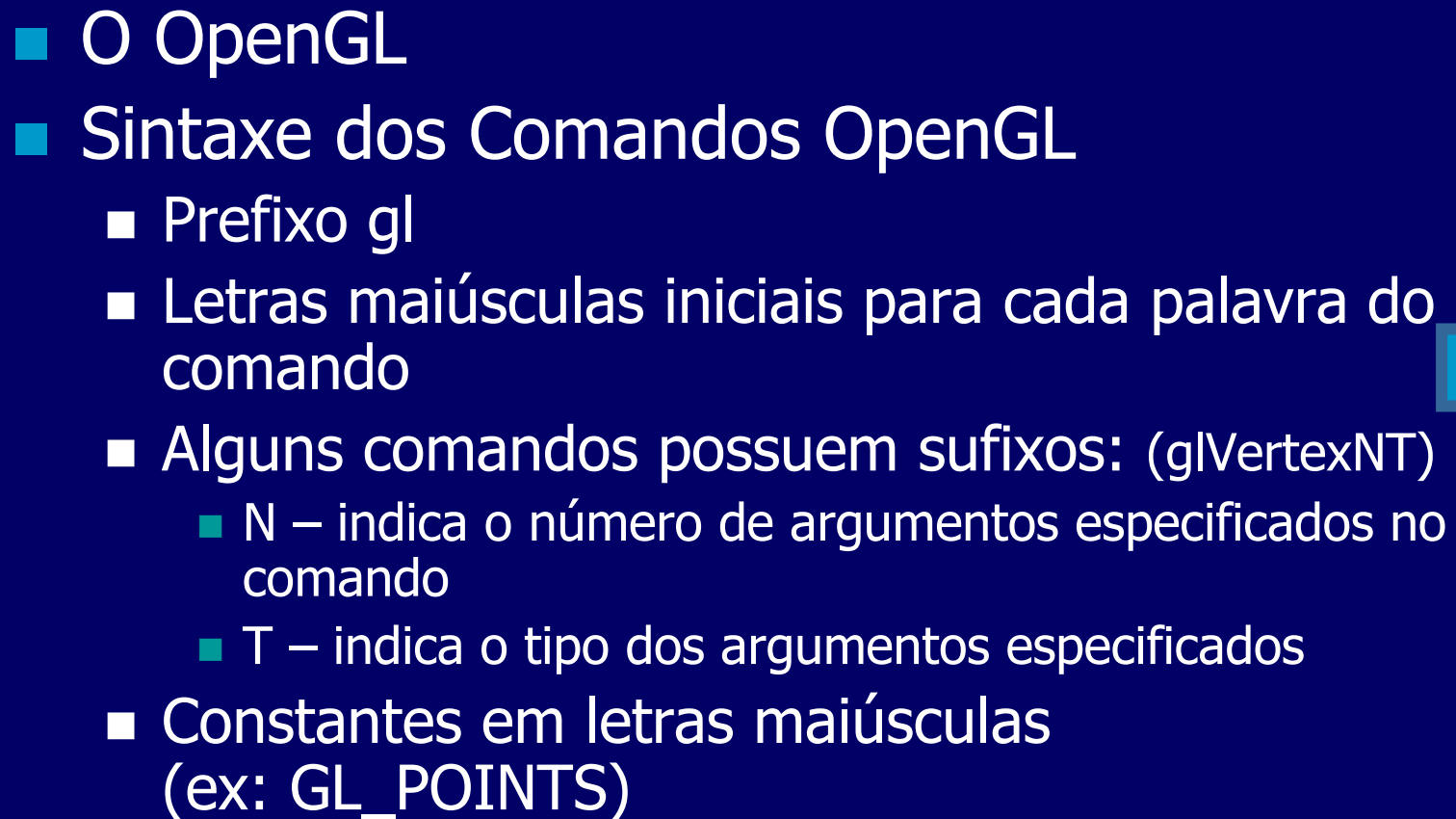


# Módulos de Plug-in

- 
- Módulos e Servidores de Plug-in
  - Alguns Tipos de Plug-ins do Photoshop
    - Automation
    - Import
    - Export
    - Filter
  - Extensões dos Plug-ins ( Filter - .8bf )
- 



# OpenGL

- O OpenGL
  - Sintaxe dos Comandos OpenGL
    - Prefixo gl
    - Letras maiúsculas iniciais para cada palavra do comando
    - Alguns comandos possuem sufixos: (glVertexNT)
      - N – indica o número de argumentos especificados no comando
      - T – indica o tipo dos argumentos especificados
    - Constantes em letras maiúsculas (ex: GL\_POINTS)
- 




# OpenGL



- Alguns Sufixos

b	8-bits integer
s	16-bits integer
i	32-bits integer
f	32-bits floating-point
d	64-bits floating-point



Ex: glColor3f (a1,a2,a3)  
glVertex2f (a1,a2)



# OpenGL




- Desenho Geométrico de Primitivas

- glBegin (Tipo de Primitiva)
- glVertex (Vértice)

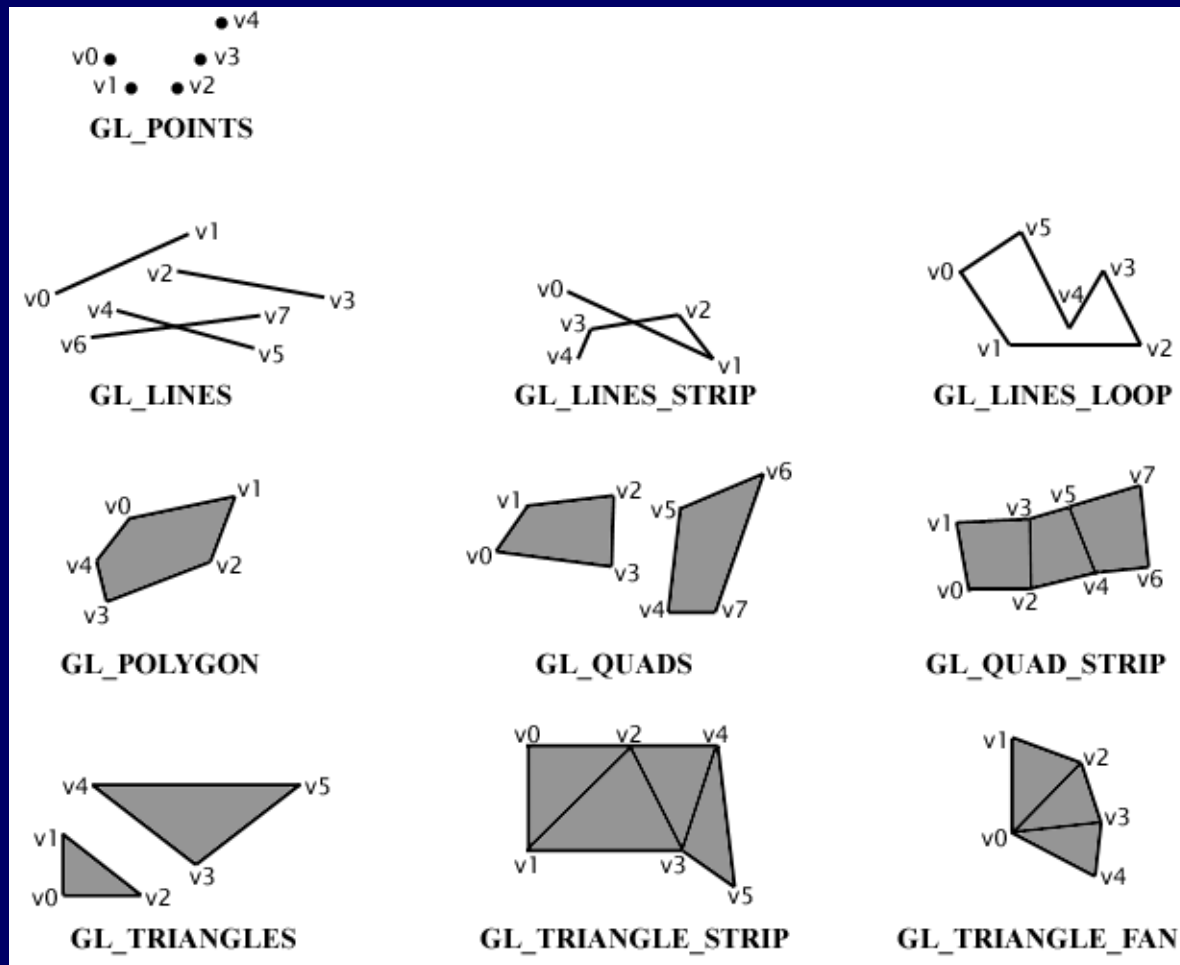
- Tipos de Primitivas

GL\_POINTS, GL\_LINES, GL\_POLYGON,  
GL\_TRIANGLES, GL\_QUADS,  
GL\_LINE\_STRIP, GL\_LINE\_LOOP,  
GL\_TRIANGLE\_STRIP, GL\_TRIANGLE\_FAN,  
GL\_QUAD\_STRIP



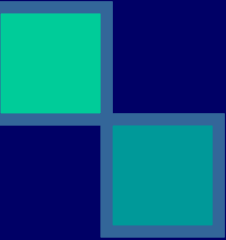



# OpenGL



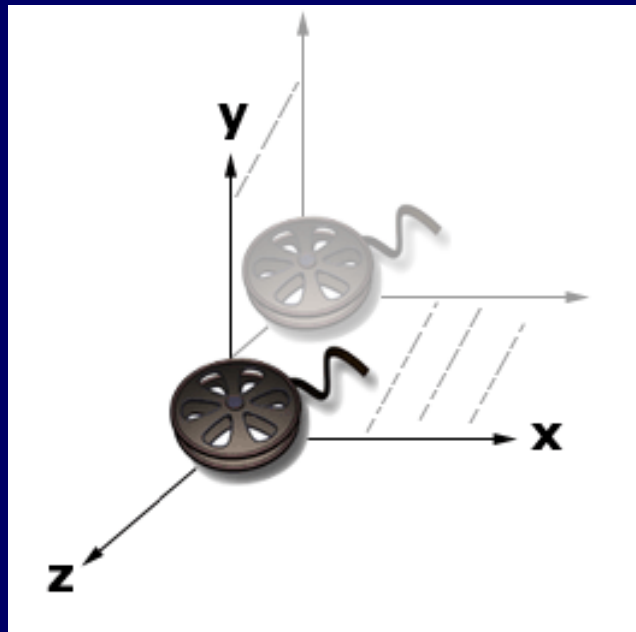


# OpenGL

- Sistema de Coordenadas
    - Regra da Mão Direita
  - Transformações Geométricas
    - Translação
    - Rotação
    - Escala
- 
- 

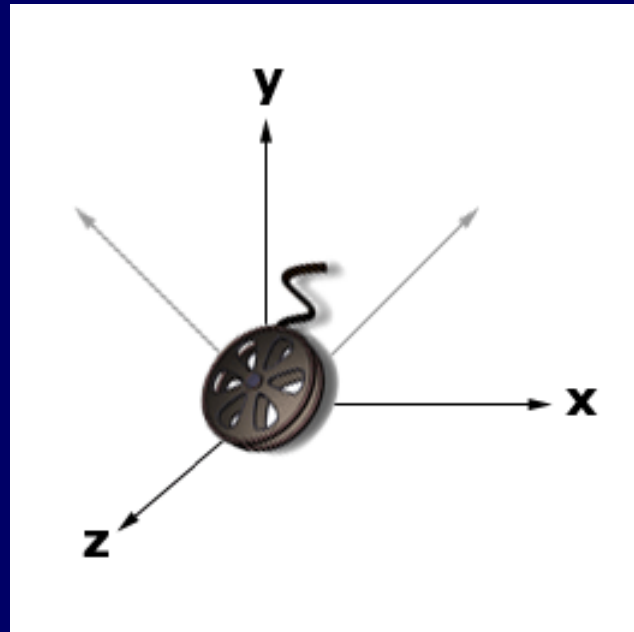
# OpenGL

Translação: `glTranslate{fd} (x,y,z)`



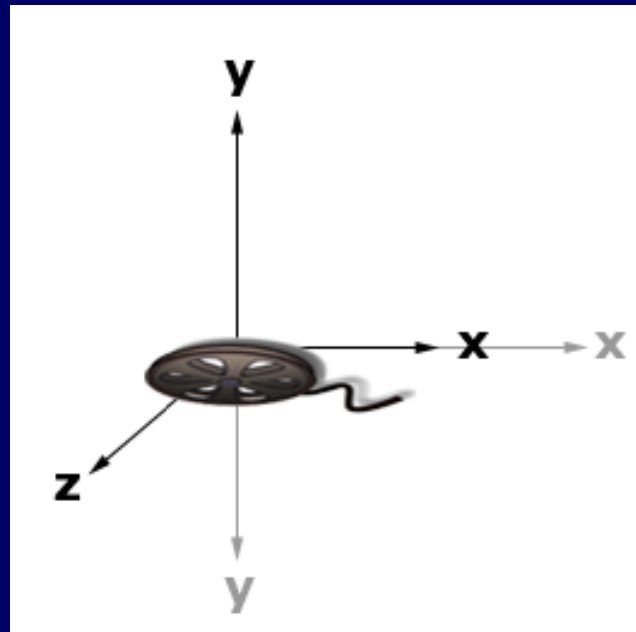
# OpenGL

Rotação: `glRotate {fd} (ângulo,x,y,z)`




# OpenGL

Escala: glScale {fd} (x,y,z)



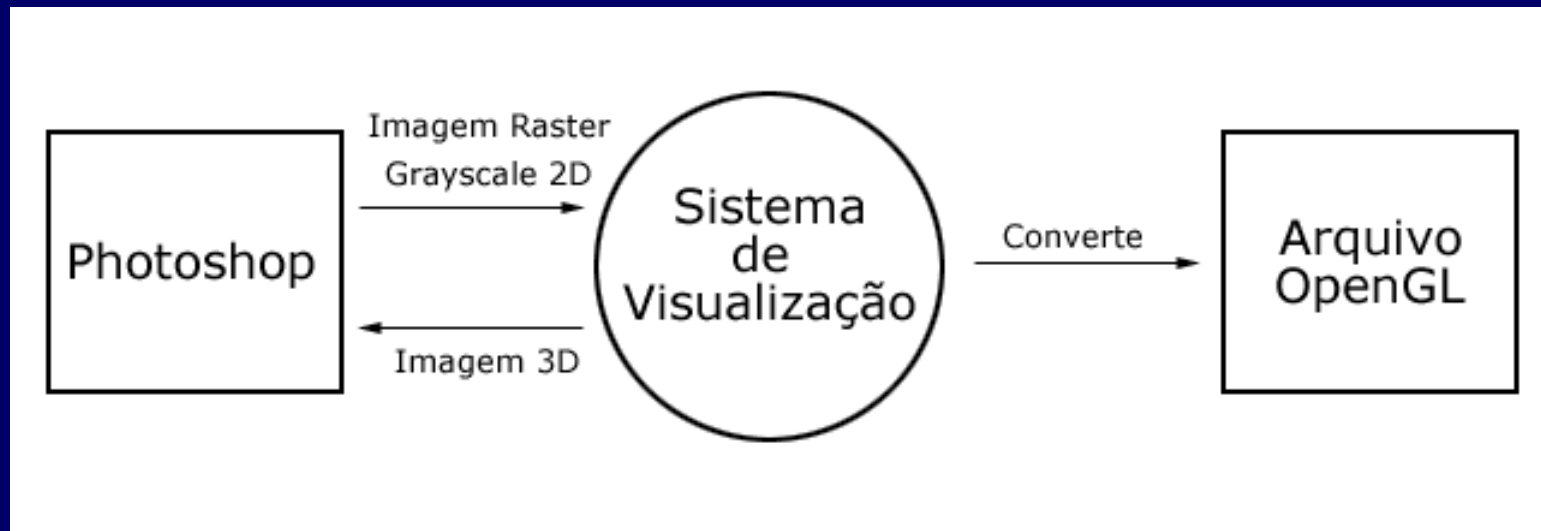


# Desenvolvimento

- Especificação
    - Linguagem de Programação Visual C++ 6.0
    - Biblioteca Gráfica OpenGL
    - PIPL
    - Algoritmo de Conversão 2D para 3D
    - Algoritmo de Retorno da imagem do Plug-in para o Photoshop
- 

# Desenvolvimento

## Diagrama de Contexto



# Desenvolvimento

## Fluxograma





# Desenvolvimento

## ■ Adobe Photoshop PIPL

### Estrutura do PIPL

```
typedef struct PIPropertyList
{
    int32          version; /* Versão Corrente = 0
    int32          count;   /* 0 = Nenhuma Propriedade
    PIProperty     properties[1];
} PIPropertyList;
```

```
typedef struct PIProperty
{
    OSType         vendorID;
    OSType         propertyKey;
    int32          propertyID;
    int32          propertyLength;
    char           propertyData[1];
} PIProperty;
```

# Desenvolvimento

## Algumas Propriedades da Estrutura PIPL

Nome	Chave	Descrição
PIKindProperty	'kind'	Tipo de Plug-in: '8BFM'=Filter module
PIVersionProperty	'vers'	16 bits / 32 bits
PIPriorityProperty	'prty'	Utilizado para definir a ordem de carga do plug-in no Photoshop.
EnableInfo	'enbl'	Define quando o plug-in estará habilitado para ser utilizado, através do modo de imagem. Alguns tipos aceitos: GrayScaleMode, IndexedMode, RGBMode, entre outros.
PICategoryProperty	'catg'	Especifica qual o nome do sub-menu que será colocado no menu <b>Filter</b> . ( ex: TCC )
PINameProperty	'name'	Nome do plug-in que será colocado no sub-menu especificado na propriedade PIRCategoryProperty. ( ex: Plug-in 3D )
PIWin32X86CodeProperty	'wx86'	Especifica para qual estrutura irá ser colocado os valores da imagem do Photoshop

# Desenvolvimento

## Obtenção dos pixels válidos

```
IndiceAtual = 0;
QtdPontos = 0;
for (LinhaAtual=0;LinhaAtual<LinhasPhotoshop;LinhaAtual++)
{
    for (ColunaAtual=0;ColunaAtual<ColunasPhotoshop;ColunaAtual++)
    {
        Cor=DadosPhotoshop[ColunaAtual + (LinhaAtual * ColunasPhotoshop)]
        If (Cor != 255)
        {
            IndicePontos[QtdPontos] = IndiceAtual;
            CoresValidas[QtdPontos] = Cor;
            QtdPontos++;
        }
        IndiceAtual++;
    }
}
```

# Desenvolvimento

## Conversão para OpenGL

```
RazaoX = 2 / ColunasPhotoshop;  
RazaoY = 2 / LinhasPhotoshop;  
for (i=0;i<QtdPontos;i++)  
{  
    Coluna = IndicePontos[i]%ColunasPhotoshop; /*Pega o Resto da Divisão  
    Linha = IndicePontos[i]/ColunasPhotoshop; /*Linha é um inteiro  
    Z      = (((256-CoresValidas[i])/256) * -1);  
    glColor3f(CoresValidas/256, CoresValidas/256, CoresValidas/256);  
    if RazaoX == RazaoY  
    {  
        glVertex3f((RazaoX * Coluna) - 1,((RazaoY * Linha) - 1) * -1, 0);  
        glVertex3f((RazaoX * Coluna) - 1,((RazaoY * Linha) - 1) * -1, Z);  
    }  
    ...  
}
```

# Desenvolvimento

## Conversão para OpenGL

```
...
else
{
    if (RazaoX > RazaoY)
    {
        AreaLivre = 1 - (RazaoY / RazaoX)
        glVertex3f((RazaoY * Coluna) + AreaLivre - 1, ((RazaoY * Linha) - 1) * -1, 0);
        glVertex3f((RazaoY * Coluna) + AreaLivre - 1, ((RazaoY * Linha) - 1) * -1, Z);
    }
    else
    {
        AreaLivre = 1 - (RazaoX / RazaoY)
        glVertex3f((RazaoX * Coluna) - 1, ((RazaoX * Linha) + AreaLivre - 1) * -1, 0);
        glVertex3f((RazaoX * Coluna) - 1, ((RazaoX * Linha) + AreaLivre - 1) * -1, Z);
    }
}
}
```

# Desenvolvimento

## Conversão OpenGL (Bitmap) para Photoshop

```
short itemWidth = (short)(pChannel->bounds.right - pChannel->bounds.left);
short itemHeight = (short)(pChannel->bounds.bottom - pChannel->bounds.top);
float Razao_x = (float)itemWidth / (float)filterWidth;
float Razao_y = (float)itemHeight / (float)filterHeight;
int Size = itemWidth * itemHeight;
LPBYTE lpBits = g_clientCapture.m_pBits;
INT* pw32Bits = (INT*)lpBits;
INT* vetor_x = new int[itemWidth];
INT* vetor_y = new int[itemHeight];

...
```

# Desenvolvimento

## Conversão OpenGL (Bitmap) para Photoshop

```
...
if (Razao_x = Razao_y)
{
    if ((int)Razao_x > 0)
    {
        for (a = 0; a < itemWidth; a++)
        {
            vetor_x[(int) a] = 1;
            vetor_y[(int) a] = 1;
        }
        a = 0;
        for (qtd = 0; qtd < filterWidth; qtd++)
        {
            vetor_x[(int) a] = 0;
            vetor_y[(int) a] = 0;
            a = a + Razao_x;
        }
    }
}
...

```

# Desenvolvimento

## Conversão OpenGL (Bitmap) para Photoshop

```
for (y = 0; y < itemHeight; y++)
{
    int i = y * itemWidth;
    if (vetor_y[y] != 0)
    {
        if (vetor_y[y] == 1) //copia linha anterior
            for (int a = itemWidth; a > 0; a--)
                data[(Size - (y * itemWidth)) - a] = data[(Size - ((y - 1) * itemWidth)) - a];
        else {
            for (x = (itemWidth - 1); x > -1; x--) {
                if (vetor_x[((x + 1) - itemWidth) * -1] == 0) {
                    b = (BYTE)(*pw32Bits);
                    pw32Bits++;
                }
                int nGray = b;
                data[((Size - i) - x) - 1] = nGray;
            }
        }
    }
}
```






# Desenvolvimento

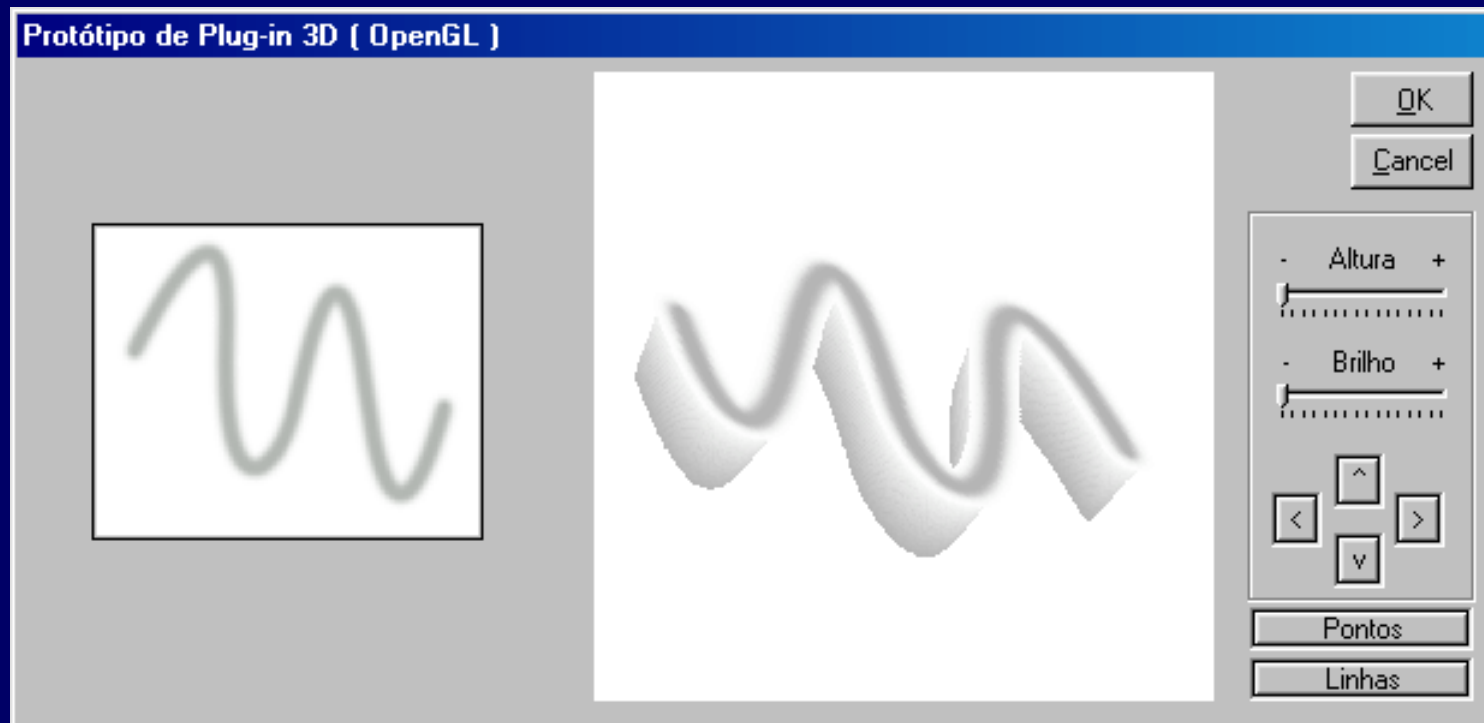


## Implementação

- Baseado em um exemplo da Adobe
  - Permite visualização da imagem em 3D em formato de linhas e pontos
  - Permite rotação, escala e translação da imagem
  - Permite ajustar o brilho e altura da imagem
- 

# Desenvolvimento

## Implementação






# Conclusão



- **Objetivos propostos alcançados**

- Funções básicas do sistema

- Carregar arquivos raster 2D em grayscale para dentro do plug-in
      - Converter o arquivo raster 2D para o formato de arquivo OpenGL
      - Ser capaz de apresentar em 3D a imagem raster
      - Possibilitar a manipulação da imagem 3D
      - Retornar a imagem 3D do plug-in para o Photoshop
- 

- **Dificuldades encontradas**

- Pouco material referente a criação de plug-ins para o Photoshop
  - Falta de conhecimento profundo da linguagem Visual C++

- **Primeiro plug-in brasileiro para o Photoshop**



# Conclusão

- Extensão
    - Outros formatos de imagens ( ex: RGB )
    - Geração de imagens 3D a partir de outras primitivas
- 
- 