S06 – Texturas

COMPUTAÇÃO GRÁFICA

Configurar Texturas

Aqui uma textura externa de um arquivo é carregada, para isso são necessárias duas informações:

```
BitMapFile* image[1];

image[0] = getBMPData("Textures/launch.bmp");

glBindTexture(GL_TEXTURE_2D, texture[0]);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);

glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, image[0]->sizeX, image[0]->sizeY, 0,
    GL_RGB, GL_UNSIGNED_BYTE, image[0]->data);
```

Composição do Bitmap

Diferente do .obj que é um arquivo de texto, o .bmp é um arquivo binário, onde cada bit tem significância isolada, precisamos ler de acordo com a sua estrutura para conseguir as informações que precisamos:

Bitmap File Structure

Block	Field	Width	Description
BITMAPFILEHEADER **Fields:** 5 **Width:** 14 bytes	FileType	2 bytes	A 2 character string value in ASCII. It must be 'BM' or '0x42 0x4D'
	FileSize	4 bytes	An integer (unsigned) representing entire file size in bytes (number of bytes in a BMP image file)
	Reserved	2 bytes	To be utilized by an image processing application. Initialized to '0' integer (unsigned) value.
	Reserved	2 bytes	To be utilized by an image processing application. Initialized to '0' integer (unsigned) value.
	PixelDataOffset	4 bytes	An integer (unsigned) representing the offset of actual pixel data in bytes.
BITMAPINFOHEADER **Fields:** 11 **Width:** 40 bytes	HeaderSize	4 bytes	An integer (unsigned) representing the size of the header in bytes. It should be '40' in decimal.
	ImageWidth	4 bytes	An integer (signed) representing the width of the final image in pixels.
	ImageHeight	4 bytes	An integer (signed) representing the height of the final image in pixels.
	Planes	2 bytes	An integer (unsigned) representing the number of color planes. Should be '1' in decimal.
	BitsPerPixel	2 bytes	An integer (unsigned) representing the number of bits a pixel takes to represent a color.
	Compression	4 bytes	An integer (unsigned) representing the value of compression to use. Should be '0' in decimal.
	ImageSize	4 bytes	An integer (unsigned) representing the final size of the compressed image. Should be '0' in decimal.
	XpixelsPerMeter	4 bytes	An integer (signed). Should be set to '0' in decimal to indicate no preference of the target device.
	YpixelsPerMeter	4 bytes	An integer (signed). Should be set to '0' in decimal to indicate no preference of the target device.
	TotalColors	4 bytes	An integer (unsigned) representing the number of colors in the color pallet.
	ImportantColors	4 bytes	An integer (unsigned) representing the number of important colors. Ignored by setting '0' in decimal.
COLOR TABLE **Fields:** 4 x entries **Width:** 4 x entries	Red	1 bytes	An integer (unsigned) representing Red color channel intensity.
	Green	1 bytes	An integer (unsigned) representing Green color channel intensity.
	Blue	1 bytes	An integer (unsigned) representing Blue color channel intensity.
	Reserved	1 bytes	An integer (unsigned) reserved for other uses. Should be set to '0' in decimal
PIXEL DATA			An array of pixel values with padding bytes. A pixel value defines the color of the pixel.

semi-optional

Wrapping das Texturas

Outra coisa que precisamos ter em mente são as formas de wraping da textura:

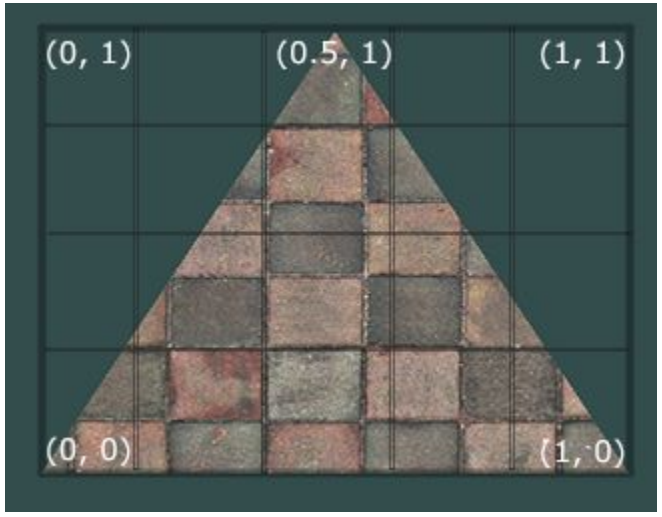
Configurar Texturas

Com isso conseguimos ver que os dados das imagens são carreados, depois o wrapping é configurado para largura (s) e altura (t)

```
BitMapFile* image[1];

image[0] = getBMPData("Textures/launch.bmp");

glBindTexture(GL_TEXTURE_2D, texture[0]);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
```

Coordenadas s e t

O padrão de coordenadas st aqui é um espelho vertical do uv



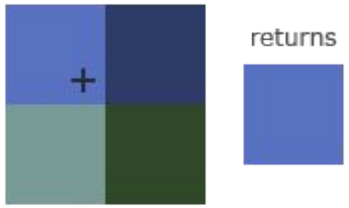
Filtro do Texel

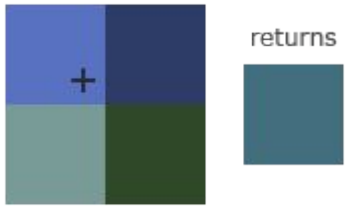
O valor final do pixel mapeado da textura precisa ser configurado, não existe uma relação direta da textura com a forma mostrada, e essa mudança aumenta quanto menor for a textura.

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
```

Filtro do Texel

Os dois principais filtros: NEAREST (esquerda), pega o valor do pixel da textura mais próximo ao texel, LINEAR (direita) faz uma média ponderada dos pixels próximos para definir a cor do pixel.





Filtro do Texel

GL_NEAREST

GL_LINEAR

Carregar a imagem para a textura

Finalmente passamos as informações dos dados e do tamanho da imagem para as texturas.

```
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, image[0]->sizeX, image[0]->sizeY, 0,
    GL_RGB, GL_UNSIGNED_BYTE, image[0]->data);
```

Texturas Procedurais

Também podemos gerar texturas procedurais, como nesse caso um tabuleiro de xadrez, que pode ser mapeado da mesma forma que um bitmap:

```
int i, j;
for (i = 0; i < 64; i++)
    for (j = 0; j < 64; j++)
        if ((((i / 8) % 2) && ((j / 8) % 2)) || (!((i / 8) % 2) && !((j / 8) % 2)))
        {
            chessboard[i][j][0] = 0x00;
            chessboard[i][j][1] = 0x00;
            chessboard[i][j][2] = 0x00;
        }
        else
        {
            chessboard[i][j][0] = 0xFF;
            chessboard[i][j][1] = 0xFF;
            chessboard[i][j][2] = 0xFF;
        }
```

Texturas Procedurais

Após criar as texturas precisamos ativá-las no código, assim como fizemos com a luz semana passada, e definir algumas configurações de ambiente:

```
glEnable(GL_TEXTURE_2D);
```

Desenho

Por fim temos a função de desenho, que funciona normalmente como as demais, com a diferença de especificarmos a posição de mapeamento da textura para o vértice seguinte (mais ou menos igual a textura).

```
glClear(GL_COLOR_BUFFER_BIT);

glLoadIdentity();
gluLookAt(0.0, 0.0, 20.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
glRotatef(angle, 0.0, 1.0, 0.0);

glBindTexture(GL_TEXTURE_2D, texture[id]);

glBegin(GL_POLYGON);
glTexCoord2f(0.0, 0.0); glVertex3f(-10.0, -10.0, 0.0);
glTexCoord2f(1.0, 0.0); glVertex3f(10.0, -10.0, 0.0);
glTexCoord2f(1.0, 1.0); glVertex3f(10.0, 10.0, 0.0);
glTexCoord2f(0.0, 1.0); glVertex3f(-10.0, 10.0, 0.0);
glEnd();
```