Bump Mapping with MathGL3d

Jens-Peer Kuska

Interdisziplinäres Zentrum für Bioinformatik
Universität Leipzig
Härtselstr. 16-18
D-04107 Leipzig
kuska@informatik.uni-leipzig.de

Bump mapping is a technique frequently used in 3D graphics to add geometric detail to an object, without increasing the number of polygons needed for the object. A bump map is a texture that modifies the surface normal of the object and create the illusion of small geometric structure on the surface. The paper show how bump mapping with MathGL3d \cite{5} can be used for scientific visualisation.

Introduction

In many computer games, bump mapping is a frequently used technique to increase the geometric detail of surfaces without the need of processing more polygons. A bump map is a gray level texture, the is used to modify the surface normal for the lighting calculation. Bump mapping was introduced by James Blinn \cite{1}. The normal way, how a graphics processor compute the lighting of a single polygon is, that for every vertex the lighting computation is performed and when a triangle is drawn on screen the colors for the vertices are interpolated to create the illusion of a continuous curved surface. For bump mapping this is not possible because the polygon mesh of the surface is much coarser than the geometric detail from the bump texture. For bump mapping one needs a per-pixel lighting calculation that is available only on modern graphics cards. For bump mapping MathGL3d \cite{5} needs the to modify the normal processing pipe line for 3d data to implement the per pixel lighting calculation.

```
In[2]:= Get["OpenGLViewer" ]
```

We can check if bump mapping with MathGL3d is supported on the graphics card:

```
In[9]:= And @@ (MemberQ[(OpenGLExtensions /. OpenGLInformation[]), #] & /@ {"GL_ARB_fragment_program", "GL_ARB_vertex_program", "GL_ARB_texture_cube_map", "GL_ARB_multitexture"})
```

```
Out[9]= True
```

As a first example a Julia fractal is mapped onto a part of the Riemann sphere. Using a color texture and a bump texture alone gives the following images.
The right column show the colorized Julia map and the texture mapping, the right column show the corresponding gray map and the bump map.

MathGL3d can also combine the color texture with the bump map to produce the image below.

```
In[31]:= MVShow3D[surf, MVTexture -> bimage, 
MVTextureMapType -> MVMeshVUMapping, MVNewScene -> True]; 
```

In[32]:= MVPasteGraphics[];
Internal Details

MathGL3d implements a tangent-space bump mapping [2]. This means, that MathGL3d needs not only the surface normal, it also need a tangent vector on the surface. From the normal vector and a single tangent vector the local coordinate system can be computed to perform the lighting computation with the new surface normal information from the bump map.

That is the reason why bump mapping only work with parametric surfaces. The vertex processor of graphics hardware transform the vertex, the normal and the tangent and pass this information to the fragment processor (that's why we need the "GL_ARB_vertex_program" extension). The fragment processor does lighting calculation for every pixel (with help of the "GL_ARB_fragment_program" extension). The lighting is done by looking up the diffuse and specular light from a pre-calculated cube-texture map. The pre-calculation of the light in a cube map texture allow MathGL3d to render bump maps in a single pass. Otherwise one would need to render the bumped objects for every light source once. Since one needs a texture for the color, a texture for the bump information and a cube map for the light MathGL3d need the "GL_ARB_multitexture" extension. The usage of a texture to look up the light allow a very fast rendering, but it has the disadvantage, that only directional lights can be processed in the lighting calculation.
MathGL3d take a gray level bitmap as bump texture. The visual height of the bump map is controlled by the MVBumpScale option.

The outline of the sphere show clearly that the bumps are an illusion of the lighting calculation.

Beside a texture with the color information one can combine the bump map with surface colors.

\begin{verbatim}
In[2]:= squareKnot = {-22\ Cos[t] - 128\ Sin[t] - 44\ Cos[3\ t] - 78\ Sin[3\ t],
11\ Cos[t] - 43\ Cos[3\ t] + 34\ Cos[5\ t] - 39\ Sin[5\ t],
70\ Cos[3\ t] - 40\ Sin[3\ t] + 8\ Cos[5\ t] - 9\ Sin[5\ t]};
In[9]:= surf = FrenetFrame[squareKnot, t, phi, 20];
\end{verbatim}
In[68]:= mesh = ParametricPlot3D[
    Evaluate[Append[surf, SurfaceColor[Hue[t/Pi]]]],
   {phi, 0, 2 Pi}, {t, 0, 2 Pi}, PlotPoints -> {32, 128},
   ViewPoint -> {-1.10, -2.11, 0.968},
   ViewCenter -> {0.45, 0.5, 0.5}];

Since the pattern is only a $128 \times 128$ texture it is repeated $2 \times 32$ times over the parametric surface.

In[70]:= MVShow3D[mesh, MVBumpTexture -> "`scottgrayheight.png",
   MVScaleTexture -> {2, 32}, MVTextureMapType -> MVMeshUVMapping,
   MVNewScene -> True, MVGrayBackground -> True];
   MVPasteGraphics[];
Scientific Visualization

While bump mapping is frequently used in computer games, it can be used also for scientific visualization. In many visualization tasks one need to display as much information as possible and the bumps offer a way to use the local reflection in addition to the shape of the surface and the surface color. Bump mapping can be used with line integral convolution textures \[3,4\] to create surface that look like sand streamed.

As a first example, the electric field of a water molecule is plotted on the surface of spheres arround the atom centers. The usual image with color textures and without bump gives a idea how the field lines look.

The bump mapped version gives the picture below.
The bump mapping gives a more intuitive picture of the field on the surfaces due to the sand-stream effect.

Bump mapping can be combined with other effects like the line illumination in MathGL3d. The next example show a line integral convolution as bump map in the \( x y \)-plane of the magnetic field that a wire in the shape of a trefoil knot produce.

\[
\begin{align*}
\dot{x}(t) &= (\lambda - b) x(t) + (z(t) + d(1 - z(t)^2)) x(t) - c y(t) \\
\dot{y}(t) &= c x(t) + (\lambda - b) y(t) + y(t) (z(t) + d(1 - z(t)^2)) \\
\dot{z}(t) &= -x(t)^2 - y(t)^2 - z(t)^2 + \lambda z(t)
\end{align*}
\]
together with the line integral convolution of the vector fields as bump map and the magnitude of the vector field as color texture.

![Image of vector field and bump mapping]

**Conclusions**

Bump mapping is a useful technique for scientific visualization. It allows the display of an additional parameter beside color and geometry on surfaces. It can be combined with other effects like line illumination and volume rendering. When available, MathGL3d will extend the bump mapping to true displacement mapping.

**References**


