MP2GL: Prototyping 3D Objects with MetaPost and OpenGL

Denis Roegel, University of Nancy, France
roegel@loria.fr

EuroTEX 2005
Pont-à-Mousson, France
7–11 March 2005
Summary

- Limitations of MetaPost
- Motivations of MP2GL
- OpenGL
- MP2GL interface
- 3D equations
- Text
- Animations
- Lights, color, ... 
- Limitations of MP2GL
- Related work
Limitations of MetaPost

- MetaPost is not adapted to 3D
- a hidden faces algorithm needs to be implemented
- lights
- colors
- numerical limitations
- static view
Motivations of MP2GL

Two remedies can be envisioned, at a non-macro level:

- extending the core MetaPost with 3D support: a lot of work in perspective… (no case known)

- using an external processor:
  - not 3D-aware: a lot of work… (3DLDF)
  - 3D-aware: easy (MP2GL)
Question: do we need a 3D-MetaPost?

In order to answer that question, let’s see good reasons to use 2D-MetaPost:

- \( \text{T}_{\text{E}}\text{X} \) partner;

- high-quality technical drawings;

- vector graphics;

- declarative approach;

- nice types;

- fun!
Among these reasons, the intrinsic MetaPost ones are:

- \text{$\TeX$ partner;}

- declarative approach;

- nice types;

A 3D-MetaPost would include the 2D-MetaPost features, plus:

- 3D vector graphics;

- animations.
Our proposal: MP2GL

MP2GL = MetaPost to OpenGL

- OpenGL: standard API for graphics in the industry;

- rendering of 3D scenes, with lights, shading, hidden faces removal ($z$-buffer), etc.

- scenes can be saved in bitmap, and in PS, using the GL2PS library.
Most 3D objects that one wants to build are:

- either geometrically very simple (cube, ...)
- or obtained simply from 2D objects (prism, ...)
- or composed of simpler 3D objects.

These three ways of building an object are supported in MP2GL.
Main features of MP2GL

- MetaPost input language;
- structures for points and homogeneous coordinates;
- interface to OpenGL objects (polyhedra, ...)
- ability to build low-level objects;
- MetaPost paths can be used to build prisms;
- equations can be used in 3D;
- production of C-code with a minimal animation interface;
• a scene can be saved in bitmap and PS;

• \TeX\ labels can be added and later adjusted;

• the C output is editable and can be used without MetaPost;

• objects can be created on the OpenGL side;

MP2GL should be seen as a *gateway* from MetaPost to OpenGL, both for the objects and for the user.
OpenGL

An OpenGL scene is created by translations and rotations, and polygons defined by vertices.

A tetrahedron at \((-4.1, 5, 12.3)\) is obtained with:

```gl
    glTranslatef(-4.1, 5, 12.3);
    glutSolidTetrahedron();
```

A square can be built with:

```gl
    glBegin(GL_POLYGON);
    glNormal3f(0, 0, 1);
    glVertex3f(0, 0, 0);
    glVertex3f(1, 0, 0);
    glVertex3f(1, 1, 0);
    glVertex3f(0, 1, 0);
    glEnd();
```
The color type is used for points:

```
def Point = color enddef;
def Xpart = redpart enddef;
def Ypart = greenpart enddef;
def Zpart = bluepart enddef;
```
MP2GL interface: transformations

- Translations, rotations, etc.

- MetaPost mimicks OpenGL:
  
  - Translate $\rightarrow$ glTranslatef
  
  - ...
  
  - PushPosition, PopPosition

These transformations are:

- output in C;

- processed internally in order to maintain a “current transformation” which can be used if necessary;
MP2GL interface: basic objects

Polyhedra, sphere, cone, torus, teapot, disk, partial disk, cylinder.
MP2GL interface:
path-based objects

path p;
p=......--cycle;
storepath(p,"Path_P");

new_prism("Prism1","Path_P",3cm);

begin_scene;
  use_object("Prism1");
end_scene;
A more complex example, made of four paths:
MP2GL interface: low-level

A cube can be constructed face by face:

```python
def build_cube_face=
    begin_convex_polygon;
    normal(0,0,-1);
    vertex(0,0,0);
    vertex(0,1,0);
    vertex(1,1,0);
    vertex(1,0,0);
    vertex(1,1,0);
    vertex(1,0,0);
    end_convex_polygon;
enddef;
```
MP2GL interface:
low-level (cont’ed)

beginobject("Cube");
  set_diffuse_color(1.0,0.0,0.0);
  build_cube_face; % bottom face
PushPosition;
  Translate(1,0,0);RotateY(-90);
  set_diffuse_color(0.0,1.0,0.0);
  build_cube_face;
...
endobject
3D equations

Equations are a powerful way to define positions through linear constraints in MetaPost.

\[
D = .5[B,C]; E = .5[C,A]; F = .5[A,B];
I = whatever[B,E] = whatever[A,D];
\]

The same mechanism can be used on points in space, within MetaPost.
3D equations: splitting a tetrahedron ... (1)

Point $K$ can be obtained by similar means:

\[ E = \frac{1}{2}[B,C]; \quad F = \frac{1}{2}[C,D]; \quad G = \frac{1}{2}[B,D]; \]
\[ H = \frac{1}{2}[A,D]; \quad I = \frac{1}{2}[A,B]; \quad J = \frac{1}{2}[A,C]; \]
\[ K = \text{whatever}[G,J] = \text{whatever}[H,E]; \]
3D equations: 
... into four tetrahedra (2)

new_tetrahedron("t1",A,B,C,K);
new_tetrahedron("t2",C,B,D,K);
new_tetrahedron("t3",A,C,D,K);
new_tetrahedron("t4",B,A,D,K);

We are going to move these four tetrahedra, and the four new vertices will be used to insert a sphere.

MetaPost can be used to compute the center of that sphere, as well as its radius, by ordinary equations (and a little bit of whatever abuse...):

... 
V9=V1+whatever*V2+whatever*V3 
=V4+whatever*V2+whatever*V5 
=V6+whatever*V7+whatever*V8;

(six different values of whatever!)
3D equations:  
... into four tetrahedra (3)

The resulting construction is the following:

```plaintext
begin_scene;
    PushPosition;
        TranslateV(K-D);
        use_object("t1");
    PopPosition;
    ...
    TranslateV(V9);
    wire_sphere(norm(W1-V9),30,30);
end_scene;
```
• \( \TeX \) labels can be added at specific projected locations; the \( \TeX \) output can be edited;

• labels are not in space;

• text in space could be obtained with the appropriate objects, created from paths;

• other features could be added, for instance fake labels, or a second processing through MetaPost.
Animations

- the output of MP2GL is an interactive animation;

- the user has a minimal interface for moving around the scene; all positions are reachable;

- the scene can be saved as JPEG or PS; other formats (such as PPM, or PDF) are easy to add;

- the animation can be edited, for instance for changing the lights, colors, the camera, etc.

- the animation could be extended in order to produce a series of bitmaps, which could then be made into an MPEG;
• lights are hardwired, but future versions of MP2GL may make them changeable; they can currently be changed in OpenGL;

• colors follow the OpenGL model (emission, ambient, diffuse, specular);
Limitations of MP2GL

Many features will be added (for instance NURBS), but there are also hard(er) limitations:

- **of GL2PS:**
  - no textures in PS output;
  - no transparency in PS output;

- **of OpenGL:**
  - no shadows
  - no CSG construction
Related work

- 3d,
- m3dplain,
- featpost,
- pstricks
- 3DLDF: most promising.
Conclusion

The aim of this study was:

- to examine the feasibility of a bridge between MetaPost and OpenGL;

- to easily obtain 3D vector graphics for inclusion in a document;

- to obtain 3D objects for further processing, for instance in an independent OpenGL application.

These goals have all be met, although many features still have to be added.
Thanks!

http://www.loria.fr/~roegel
roegel@loria.fr