# DESCRIPTION OF GRAPHIC (DOG) FORMAT

Marc W. Cygnus
April 1992



A DOG is a human-readable description of one or more graphical objects. It is meant only to contain graphically renderable object information and optional comments.

A comment is indicated by a semicolon as in LISP; all text after the semicolon and on the same line is ignored.

A single object does not have to be a solid, nor even be connected. An `object' is simply a collection of graphics primitives which (for now) do not move in relation to one another and are transformed together. For example, if you wanted a number of cars in the parking lot of a virtual building, you could define all of the cars in a single DOG object. If you wanted to move a single car, however, that car would have to be its own object. If you wanted the doors on the car to open, the doors would have to be individual objects separate from the car.

Object descriptions look like:

```
object FooBar { ... }         or    object { ... }
```

where "..." are transform operations or graphical primitives, as described below. Objects may have names associated with them, as in the first example which is named "FooBar", or may be nameless.

Most objects, especially ones created with high-level modelling tools such as Alias or WaveFront, will consist entirely of graphical primitives and color information. For hand tweaking, however, DOG recognizes simple transform operations for scaling, rotation, and translation. These operations apply sequentially as they appear in the DOG text; thus, take care when concatenating more than one rotation or a scale followed by some other transform.

Basic object transformations are listed below. Note that all arguments are reals and that objects are defined in the standard right-hand coordinate system (x right, y up, z toward you).

```
- xlate <x> <y> <z>
          Translate origin to (x,y,z)
- rotatex <d>
- rotatey <d>
- rotatez <d>
          Rotate about indicated axis by <d> degrees
- scale <x> <y> <z>
          Scale coordinates as indicated
- uscale <s>
          Scale all three coordinates uniformally by <s>
          (same as scale <s> <s> <s>)
- color <r> <g> <b>
          Set object's diffuse reflectivity as indicated
          (r,g,b are [0.0,1.0] and indicate percentage of reflected
          primary)

    Graphical primitives are given below:

- polyline { <x0> <y0> <z0> <x1> <y1> <z1> ... }
          Polylines must have at least two coordinates; coordinates are
          given as three reals.

- polygon { <vertex1> <vertex2> <vertex3> ... }
          Polygons must have at least three vertices.  Vertices must
          be coplanar and must form a convex polygon.  A vertex consists
          of three reals followed by an optional vertex normal and an
          optional texture vertex.  The normal, if given, consists of 'n'
          followed by three reals.  The texture vertex, if given,
          consists of 't' followed by two reals.

          Vertex examples:
            0.0 1.0 2.0
            0.0 1.0 2.0 n 0.0 1.0 0.0
            0.0 1.0 2.0 t 0.5 0.3
            0.0 1.0 2.0 n 0.0 1.0 0.0 t 0.5 0.3

- polymesh {  { <vertex1> <vertex2> ... <vertex N> }
          { ... }
          ...
         }
```

==========

Syntax Key

```
  <foo>           : construct name "foo"
  [x]?            : zero or one instance of x (ie: x is optional)
  [x]+            : one or more instances of x
  [x]*            : zero or more instances of x
  regex: ...      : regular expression follows
  "foo"           : literal string
  'a'       : literal character


<DOGFile>   =     [ <ObjectDef> ]+

<ObjectDef> =     "object" [ <ObjectName> ]? '{' <ObjDefBody> '}'

<ObjectName>      =     <String>

<String>    =     regex: [a-zA-Z0-9-_]+

<Real>            =     any standard Real syntax understood by strtod();
                  loosely, [+-]? (  ( [0-9]+ { \.[0-9]* } )
                              | ( [0-9]* \.[0-9]+ )  )
                               { [eE][+-]?[0-9]+ }
                  where () groups and {} indicates an optional element.

<ObjDefBody>      =     [ <ObjectPrim> ]+

<ObjectPrim>      =     <XlateElement> | <RotateElement> | <ScaleElement>
                  | <ColorElement> | <PolylineDef> | <PolygonDef>

<XlateElement>    =     "xlate" <Real> <Real> <Real>

<RotateElement>   =     [ "rotatex" | "rotatey" | "rotatez" ] <Real>

<ScaleElement>    =     "scale" <Real> <Real> <Real>
                  | "uscale" <Real>

<ColorElement>    =     "color" <Real> <Real> <Real>

<PolylineDef>     =     "polyline" '{'
                    <PolyVert> <PolyVert> [ <PolyVert> ]* '}'

<PolygonDef>      =     "polygon" '{' <PolyVert> <PolyVert> <PolyVert>
                  [ <PolyVert> ]* '}'

<PolyMeshDef>     =     "polymesh" '{'
                    <VertList> <VertList> [ <VertList> ]* '}'
```

```
<VertList>  =     '{' <PolyVert> <PolyVert> [ <PolyVert> ]* '}'

<PolyVert>  =     <Real> <Real> <Real>
                  [ 'c' <Real> <Real> <Real> ]?
                  [ 'n' <Real> <Real> <Real> ]?
                  [ 't' <Real> <Real> ]?
```