Multi-Platform 3D Graphics Library ver 2.0

1 Introduction

The purpose of Multi-Platform 3D Graphics library is to provide high-performance & flexible 3D graphics to high-level programmers. So that they do not have to build their own graphics routines for each target-platforms.

2 Requirement

Hardware :

Minimum - 486 DX 33+ Mhz, 4 MB RAM, VGA Graphics Card.
Recomended - Pentium 90+ Mhz, 8-16 MB RAM, VESA compatible SVGA.

Supported Operating System :

DOS, 32-Bit DOS, Windows 3.1, Windows 95, Windows NT, OS/2 3.0, Unix with X Windows system.

Supported Compilers :

Watcom C/C++ 10.0+, Visual C/C++ 2.0+, Borland C/C++ 4.0+, GNU C/C++ Compiler.

* You need MPGFX library to use MPG3D.

3 Source Code

The source code with gfx prefix is the C++ modules for this library. The source code with lgfx prefix is the Low-Level routine for this library.

G3D sources : Extension C++ or CXX.

g3d3ds - Autodesk 3dstudio's 3DS file tool.
g3d3ds - ASCII file tool.
g3dcam - 3D Camera object class.
g3ddraw - 3D drawing class.
g3dfile - 3D file tool.
g3dgem - GEM file tool.
g3dgeo - ViewSpace GEO file tool.
g3dmath - 3D vector tool.
g3dmatrl - Material class.
g3dlight - 3D light object class.

g3dobj - 3D object base class.
g3dpobj - 3D polygon object class.
g3dpoly - 3D polygon class.
g3dshape - 3D shape class.
g3dsystm - 3D system class.
g3dworld - 3D world class.

Low-Level sources : Extension ASM or CPP or CXX.

lg3dras? - 3D raster oprations.
lg3dmat? - 3D math routines.

* Note - ? will be replaced by one of the following characters.

b - For Borland compiler.
g - For GNU compiler.
v - For Visual C/C++ compiler.
w - For Watcom C/C++ compiler.

Use appropriate low-level source for your compiler.

4 Setup for MPG3D Library.

Follow these steps to start programming with MPG3D.

1 - Create directory called "MPG3D" or something you like.
2 - Under this directory, create "source" and "include".
3 - From source disks, copy all the files with cpp,cxx,asm extensions to the directory mpgfx/source.
4 - From source disks, copy all the files with h extensions to the directory mpg3d/include.
5 - Include all the g3d?.cpp (or cxx) modules and appropriate low-level source files to your project or makefile.
6 - You must define a macro before start compiling. See below.

Macros for Compiler :

__FORBORLAND__ - Compiling for Borland.
__FORGCC__ - Compiling for GCC.
__FORVISUAL__ - Compiling for Visual.
__FORWATCOM__ - Compiling for Watcom.

Macros for Target Operating System :

__FORDOS__ - 16 Bit DOS.
__FORDOS4GW__ - 32 Bit DOS 4GW Extender.
__FORWIN16__ - 16 Bit Windows.
__FORWIN32__ - 32 Bit Windows. (Windows 95, NT, Win32s)
__FOROS2__ - 32 Bit OS/2.
__FORUNIX__ - Unix with X Wndows.

Macros for Tools or class library :

__FORMFC__ - Make it compatible to MFC class library.
__FOROWL__ - Make it compatible to OWL class library.
__FORMOTIF__ - Make it compatible to Motil Toolkit.

* Exmaple - If you want to compile for Windows 95 using Watcom C/C++ with MFC library, define :

__FORWATCOM__ __FORWIN32__ __FORMFC__

7 - Make mpg3d/include directory visible to your project.
8 - Include stdgfx.h and mpg3d.h file in your source code.

5 Programming with MPG3D

5.1 Important Classes.

G3DSYSTEM - This is the main class. Almost all operations must go through this class's interface.

G3DWORLD - This is the world class, containing various objects.

G3DOBJECT - Base 3D Object class.

G3DPOLYHEDRON - 3D polyhedron class inherited from G3DOBJECT.

G3DLIGHT - 3D light object class inherited from G3DOBJECT.

G3DCAMERA - 3D camera object class inherited from
G3DOBJECT.

G3DMATERIAL - Material is a class defines a 3D surface.

G3DMATERIALLIB - Material library. Collection and
interface to many materials.

5.2 Start programming

You must follow these steps to start programming with MPG3D.

1. Create an instance of G3DSYSTEM.
2. Create a world. (Load objects, add lights, etc ...)
3. Create material library.
4. Create a camera.
5. Set some attribues of 3D system. (Destination screen,
Viewport,
Viewdistance, etc..)
6. Show view through a camera.

Example 1

```
void main ()
  {
    G3DSYSTEM *G3D = new G3DSYSTEM ();
    G3DWORLD *World = new G3DWORLD ();

    LINKEDLIST<G3DOBJECT*> *ObjectList;
    ObjectList = new LINKEDLIST<G3DOBJECT*> ();

    if (G3D->Load3DFile ( “test.3ds”, ObjectList,
                    COUNTER_CLOCK_WISE,
                    1.0 )==FAILURE)
      {
       printf (“Can not load this objects.”);
       exit(0);
      }

    G3DLIGHT *Light = new G3DLIGHT ();
    COLORTABLE *ShadeTable = new COLORTABLE ();
    ShadeTable->Load ( “shade.tbl” );
    Light->SetPosition ( 0, 0, 0 );
    Light->SetShadeTable ( ShadeTable );
    Light->SetMinIntensity ( 0 );
    Light->SetMaxIntensity ( ShadeTable->GetNumLevels() );
```

```
World->AddObject ( Light );

G3DMATERIALLIB *MaterialLib = new MATERIALLIB ();
MaterialLib->CreateTextures ( 1 );
MaterialLib->CreateMaterials ( 1 );
if (MaterialLib->LoadTexture(0,“test.pcx”,
                        TEXTURE_STATIC,
                       ShadeTable->GetPalette())
   ==FAILURE)
  exit(0);

G3DMATERIAL *Material = MaterialLib->GetMaterial(0);
Material->Attributes = 0;
Material->Ambient = 15;  // Choose color 15
Material->SetTexture ( MaterialLib->GetTexture(0) );

LISTOBJECT* ObjectNode;
ObjectNode = ObjectList->GetHead ();
while (ObjectNode!=NULL)
  {
   G3DOBJECT *Object = ObjectNode->Data;
   Object->SetLightSource ( Light );
   Object->SetPosition ( 0, 0, 500 );

   if (Object->GetObjectType()==
       OBJECT_TYPE_POLYHEDRON)
     {
      G3DPOLYHEDRON *PolyObject =
        (G3DPOLYHEDRON*)Object;
      PolyObject->SetMaterial ( Material );
     } // End if
   World->AddObject ( Object );
   ObjectNode = ObjectNode->NextObject;
  }

G3D->SetWorld ( World, FALSE );

G3DCAMERA *Camera = new G3DCAMERA ();
Camera->SetPosition ( 0, 0, 0 );
Camera->SetAngle ( 0, 0, 0 );

IMAGE *Vscreen = new IMAGE ();

DISPLATDATA Data;
Data.Mode = M320x200x256;
Grafix.SetDisplay ( &Data );

VScreen->Create ( IMAGE_8BIT, Grafix.GetWidth(HVGA),
```

```
                    Grafix.GetHeight(HVGA) );
    VScreen->Clear ( 0 );

    G3D->SetViewPort ( 0, 0, VScreen->GetWidth()-1,
                    VScreen->GetHeight()-1 );

    G3D->SetViewDistance ( 200.0 );
    G3D->SetScreenCenter ( VScreen->GetWidth()/2,
                        VScreen->GetHeight()/2 );
    G3D->ShowView ( Camera );
    getch ();

    Grafix.ResetDisplay ();
  } // End of main
```

5.3 G3DSYSTEM

These are the interfaces to G3DSYSTEM.

G3DSYSTEM ();
- Constructor of 3D System.

~G3DSYSTEM ();
- Destructor of 3D System.

VOID SetDestination (IMAGE *Dest);
- Set the Destination of 3D System.
  All 3D rasterization will be done to this destination.

  Dest - IMAGE type.

VOID SetViewDistance (float ViewDistance);
- Set the distance between camera and view plane.

  ViewDistance - Distance in 4 bytes float.

VOID SetScreenCenter (LONG CenterX, LONG CenterY);
- Set the center of the screen.

  CenterX - Center X coord.
  CenterY - Center Y coord.

VOID SetMaterialLib (G3DMATERIALLIB *NewLib);
- Set the Material Library.

  NewLib - NewMaterial Library.

VOID SetWorld (G3DWORLD *NewWorld, BOOLEAN DeleteOld);
- Set the world.

NewWorld - World to be set.
DeleteOld - Boolean. If this is TRUE, then
the previous world will be deleted.

VOID SetShadeFlags (LONG Flags);
- Set the Shading limit.

Flags - Valid values are

SHADE_NONE - No shading.
SHADE_FLAT - Flat shading.
SHADE_GOURAUD - Gouraud shading.

VOID SetFaceFlags (LONG Flags);
- Set the Surface limit.

Flags - Valid values are

FACE_WIREFRAME - Faces will be wireframes.
FACE_SOLID - Solid color face.
FACE_TEXTURE - Texture mapped face.

VOID SetDepthCueing (BOOLEAN OnOff, float Scale);
- Set depth cueing attributes.

OnOff - If TRUE, then depth cueing will be done.
Scale - Scaling factor. If this value is small,
depth cueing gets faster. (darker).

VOID SetBlendTable (COLORTABLE *Table);

- Set the blending table. For transparent.

Table - Table of blend type.

VOID SetHazing (BOOLEAN OnOff, float Scale,
COLORTABLE *Table);
- Set the hazing (Foggy) effect.

OnOff - Set TRUE for hazing on.
Scale - Scaling factor. If this values is small,
hazing gets faster.
Table - Haze table.

G3DMATERIALLIB* GetMaterialLib () { return MaterialLib; };

- Returns the material library.

G3DMATERIAL *FindMaterialByName (STRING Name);
- Find the material by searching the name.
  If no material is found, returns NULL.

  Name - String to be used for search.

G3DMATERIAL *FindMaterialByID (LONG SearchID);
- Find the material by searching the ID.
  If no material is found, returns NULL.

  ID - Number to be used for ID.

G3DMATERIAL *FindMaterialByIndex (LONG Index);
- Find the material by Index of array.
  If no material is found, returns NULL.

  Index - Index to the array of material.
      Starting from 0 to Num-1.

G3DWORLD* GetWorld () { return World; };
- Retuns the world.

G3DOBJECT* FindObjectByName (STRING SearchName);
- Find object by searching the name.
  If no object is found, returns NULL.

  SearchName - String to be used for search.

G3DOBJECT* FindObjectByID (LONG SearchID);
- Find object by searching the ID.
  If no object is found, returns NULL.

  SearchID - String to be used for search.

BOOLEAN Load3DFile (STRING FileName,
            LINKEDLIST<G3DOBJECT*> *ObjectList,
            BOOLEAN ClockWise, double Scale);
- Loads the objects in data file.

  FileName - File to be opened. Currently, following files
      are supported.

    3DS - Audesk 3D Studio's 3DS file.
    ASC - ASCII file.
    GEM - Geometry file.
    GEO - ViewSpace's GEO file.

ObjectList - Linked list of G3DOBJECT type.
Clockwise - Set TRUE, if polygons must be loaded as clockwised.
Scale - Set the scal factor of object.

BOOLEAN Save3DFile (STRING FileName,
LINKEDLIST<G3DOBJECT*> *ObjectList,
BOOLEAN ClockWise, double Scale,
INT Type);
- Saves the objects to data file.

FileName - File to be created. Currently, following files are supported.

GEO - ViewSpace's GEO file.

ObjectList - Linked list of G3DOBJECT type.
Clockwise - Set TRUE, if polygons be must saved as clockwised.
Scale - Set the scal factor of object.

BOOLEAN Init ();
- Initializes the 3D system. Must be called before Rendering.

VOID ShowView (G3DCAMERA *Camera);
- Displays a view from camera to the destination.

Camera - G3DCAMERA type.

VOID SetViewPort (LONG x1, LONG y1, LONG x2, LONG y2);
- Sets the viewport for rasterization.

x1 - Left extend for viewport.
y1 - Top extend for viewport.
x2 - Right extend for viewport.
y2 - Bottom extend for viewport.

VOID SetNearClipZ (float Z);
- Sets the minimum distance from camera for an object to be visible.

Z - Distance in float.

FLPVECTOR3D ComputeNextPos (FLPVECTOR3D StartP,
FLPVECTOR3D Angle,
FLPVECTOR3D Vector);

- Computes the next position from starting position and angle and speed.

StartP - Start coord.
Angle - Angle to travel.
Vector - Speed in each x,y,z coord.

```
LONG CheckCollision ( FLPVECTOR3D StartPt,
                      FLPVECTOR3D EndPt,
                      COLLIDEDATA *CollideList,
                      LONG MaxNum,
                      float CollideDist, float Gap );
```

- Check if collision occured while traveling from start point to end point. Returns the number of shapes that collided.

StartPt - Start coord.
EndPt - End coord.
CollideList - Arrays of collide data.

```
struct COLLIDEDATA
  {
   G3DSHAPE *Shape;
   float CollideT;
  }; // End of COLLIDEDATA
```

MaxNum - Maximum number of shapes to fill CollideList.
CollideDist - Minumum distance ratio for collision to occur. For example, 1.0 is exactly when collision occrus. 1.2 is when coliision occurs if vector travelled 0.2 times more distance.
Gap - Distance for polygon extend.