MD2 format/ Interesting Worlds

CS116A

Chris Pollett

Nov. 10, 2004.

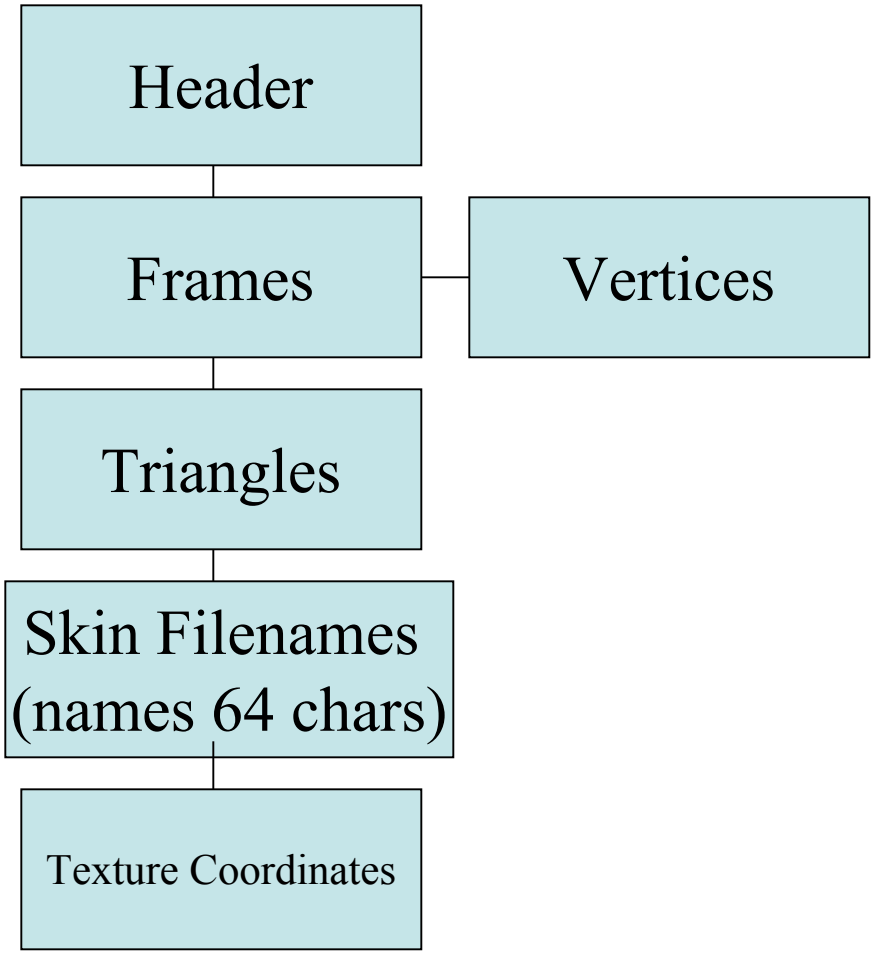
Outline

- MD2 file format
- Pop and Quake
- Ballworld Game

MD2 and MD3 file formats

- Used in projects based on the id software's Quake engine.
- Can be used with the Pop Framework.
- Can use Milkshape 3D to create new Models. (Milkshake 3D was originally to create Half-life MDL format models but has various exporters)
- Milkshakp is free for 30 days. Relatively cheap to buy.

MD2 file Structure



Header

```
struct SMD2Header
{
	int m_iMagicNum; //ID Polygon 2 = 844121161
	int m_iVersion; // always 8
	int m_iSkinWidthPx; // one skin on model at a time
	int m_iSkinHeightPx;
	int m_iFrameSize;
	int m_iNumSkins;
	int m_iNumVertices;
	int m_iNumTexCoords;
	int m_iNumGLCommands;
	int m_iNumFrames;
	int  m_iOffsetskins; //how many bytes from start of file to skins
	int m_iOffsetTexCoords;
	int m_iOffsetTriangles;
	int m_iOffsetFrames; int m_iOffsetGlCommands; int m_iFileSize;
};
```

Frame and Vertices

```
struct SMD2Frame
{
	float m_fScale[3];
	float m_fTrans[3];
	char m_caName[16];
	SMD2Vert m_avertices[m_iNumVertices];
};

struct SMD2Vert
{
	byte m_bVert[3]; //compressed need to mult by scale, add trans
	byte m_bNormal;
};
```

If just draw these points with OpenGL should look vaguely like a figure.

Triangles

```
struct SMD2Triangle
{
	unsigned short m_sVertIndices[3]; //indices into array of SMD2Verts
	unsigned short m_sTexIndices[3]; //will get to in a moment
};
```

Could draw these one by one using

```
glBegin(GL_TRIANGLES);
//look up vertices in triangle
// scale and translate
// do glVertex3fv to draw
glEnd();
```

Adding Skins

- Each element in m_sTexIndices[3]; from a SMD2Triangle corresponds to a point in a texture file.
- The relevant structure looks like:

```
struct SMD2TexCoord
{
        unsigned short m_sTex[2]; /* need to convert to 0--1 range
         for OpenGL. (Divide by m_iSkinWidthPx and height) */
}
```

- Pop understands skins which are .pcx or .bmp files
- To plot a triangle for each i =0,1,2 call:
  - glTexCoord2fv with the SMD2TexCoord specified by the m_sTexIndices[i].
  - Then call glVertex3fv with the m_bVert[3] corresponding to the given m_sVertIndices[i]

Animations

- Animation is achieved by drawing one frame waiting a little while then drawing the next frame.
- To make the animation not seem jerky should interpolate between corresponding points in successive frames. (And make the interpolation correspond to time elapsed).

GL Commands

- Drawing triangle by triangle is slow.
- Triangle strips and triangle fans are faster to draw.
- The GL commands part of an MD2 file can be used to list fans and strips.
- The relevant struct for a command looks like:

```
struc SMD2CommandElt
{
   float x,y; //texture coordinates
   int vertexIndex; // index of vertex
};
```

More Commands

- Commands are organized into m_iNumGLCommands many groups
- First element in group is a signed int.
- The absolute value of the int indicates the number of points in the group
- The sign indicates whether to draw a fan (neg) or a strip (pos).
- A zero indicates the end of the list

Pop and Quake

- The code for MD2 files for Pop was originally written by a Giavinh Pham.
- The relevant files to look at are:
  - quakemd2model.h
  - quakeMD2model.cpp
  - spritequake.h
  - spritequake.cpp
  - GraphicsOpenGL.h
  - graphicsOpenGL.cpp

More on Pop and Quake

- To use Quake Models in Pop use a tool like Milkshape to create your model.
- Your textures/skins should be .pcx or .bmp files and should be in same directory as your .md2 file
- Then subclass cSpriteQuake like:

```
class cSpriteQuakeAlien : public cSpriteQuake
{
DECLARE_SERIAL(cSpriteQuakeAlien)
public:
    cSpriteQuakeAlien():
        cSpriteQuake("models\\Invader\\Tris.MD2",
        "models\\Invader\\alien.bmp",
        cVector(0.0, 0.0, 0.48)){} /*vector is offset to apply when drawing model */
};
```

- Or just call 3 component constructor of cSpriteQuake
- Can create a new instance using default constructor then add to critter
- Useful methods for set/getting frame and animating can be found in the class.

Ballworld Game Specification

- Concept -- game is a sidescroller. Objects come toward you and you have to avoid letting them hit you.
- Appearance -- some picture
- Controls -- player uses hopper controls. Player uses left/right arrows to move left/right. Up arrow to hop. Can add hops to a hop.
- Behavior -- goal is to move to the right end of world and hop into the hoop there. If a player jumps a ball he score a point. If player bumps into a ball he loses a health point. Jumping into the hoop also scores a point and resets you to left of world.

Ballworld Design

- _border.set(100.0, 12.0, 0.0) is done to make world long and thin.
- To see only part of the world pviewer->zoom(4.0); is done.
- To make view track player…pview->pviewpointcritter()->setTrackPlayer(TRUE);
- This is all done in cGameBallWorld::initializeView
- So the world does not bounce up and down with hops so add:

  virtual int worldShape(){return cGame::SHAPE_XSCROLLER;}

  to game header. Can also do vertical scroller by changin X->Y.

More Ballworld Design

- cListenerHopper took some tweaking
- This had to be meshed with cCritterBallworldPlayer
- collide of both the player and cCritterTreasure had to be overridden
- A cForceGravity was used on cCritterBallProp to make things move to left of screen and also so would fall to ground.
- setBounciness(.9) used to make collision with bottom of screen slightly inelastic.