Texture-mapped font for OpenGL ES 1.x

An efficient approach to implement fonts in OpenGL is to use a single texture-mapped quad for each character. This technique has very good performance and presents nice results. At this link [1], there is more information on this subject.

glFont

The glFont tool [2] is a well-known Windows application and API to generate textures for fonts, and to render them using OpenGL. According to its licence, this tool is free to be used in any program, commercial or non-commercial [3]. The original author is Brad Fish (brad.fish at gmail.com).

Here is the font class converted to the Symbian OS API (The original author gently allowed the conversion to be published here):

```
//*******************************************************************
// glfont2.h -- Header for glfont2.cpp
// Copyright (c) 1998-2002 Brad Fish
// See glfont.html for terms of use
// May 14, 2002
//
// Symbian OS port - June 2007
// Luis Valente
//
//*******************************************************************

#ifndef GLFONT2_H
#define GLFONT2_H

#include <e32base.h>
#include <GLES/gl.h>

//_____________________________________________________________________________
//
// Simple class to output text as texture-mapped triangles. Does not support
// unicode strings. Reference point when drawing: top-left.
//

class GLFont
{
  public:

   /**
    * Factory-method.
    */
   static GLFont* NewL (const TDesC & aFilename);

  public:

   /**
    * Destructor.
    */
   ~GLFont ();

  public:


   /**
    * Retrieves the texture width and height.
    */
   void GetTexSize (TInt & aWidth, TInt & aHeight);

   /**
    * Retrieves the character interval.
    */
   void GetCharInterval (TInt & aStart, TInt & aEnd);

   /**
    * Retrieves the character dimensions.
    */
   void GetCharSize (TText8 c, TInt & aWidth, TInt aHeight);


   /**
    * Calculates the dimensions of a string.
    */
   void GetStringSize (const TDesC8 & aText, TInt & aWidth, TInt & aHeight);

   /**
    * Renders a string.
    */
   void DrawString (const TDesC8 & aText, GLfixed aX, GLfixed aY);

   /**
    * Sets required states for the font.
    */
   void BeginDraw ()
   {
    glEnable (GL_BLEND);
    glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    glEnable (GL_TEXTURE_2D);
    glEnableClientState (GL_TEXTURE_COORD_ARRAY);
   }

   /**
    * Turns off required states.
    */
   void EndDraw ()
   {
    glDisable (GL_BLEND);
    glDisable (GL_TEXTURE_2D);
    glDisableClientState (GL_TEXTURE_COORD_ARRAY);
   }

 private:
```


(C) Copyright Nokia 2013. All rights reserved.

```
    /**
     * Default constructor.
     */
   GLFont ();

   /**
    * Final part of the two-phase constructor.
    */
   void ConstructL (const TDesC & aFilename);

   /**
    * Loads the font file.
    */
   void LoadFileL (RFs & aFs, const TDesC & aFilename);

   /**
    * Destroys the font.
    */
   void Destroy ();


  private:

   // single character
   struct GLFontChar
   {
      GLfixed dx, dy;
      GLfixed tx1, ty1;
      GLfixed tx2, ty2;
   };

   // font header
   struct GLFontHeader
   {
      GLuint tex;
      TInt   texWidth, texHeight;
      TInt   startChar, endChar;
      GLFontChar *chars;
   };

 private:

   GLFontHeader iHeader;
};

//*****************************************************************
#endif
```

Here is the class implementation:

```
//*****************************************************************
// glfont2.cpp -- glFont Version 2.0 implementation
// Copyright (c) 1998-2002 Brad Fish
// See glfont.html for terms of use
// May 14, 2002
//
// Symbian OS port - June 2007
// Luis Valente - lpvalente
//
//*****************************************************************

// Symbian OS headers
#include <s32file.h>
#include <eikenv.h>
#include <eikappui.h>
#include <eikapp.h>
#include "glfont2.h"
#include "FixedMath.h"


// GLFontChar structure as stored in file
struct GLFontCharFile
{
   TReal32 dx, dy;
   TReal32 tx1, ty1;
   TReal32 tx2, ty2;
};

// GLFontHeaderFile structure as stored in file
struct GLFontHeaderFile
{
   TInt32  tex;
   TInt32  texWidth, texHeight;
   TInt32  startChar, endChar;
   TUint32 chars;
};



//_____________________________________________________________________________
//
// Default constructor.
//

GLFont::GLFont ()
{
   // Initialize iHeader to safe state
   iHeader.tex = 0;
   iHeader.texWidth = 0;
   iHeader.texHeight = 0;
   iHeader.startChar = 0;
   iHeader.endChar = 0;
   iHeader.chars = NULL;

   // OpenGL texture
   glGenTextures (1, &iHeader.tex);
}

//_____________________________________________________________________________
//
// Destructor.
//

GLFont::~GLFont ()
{
   // Destroy the font
```


(C) Copyright Nokia 2013. All rights reserved.

```
   Destroy();

   // delete texture
   glDeleteTextures (1, &iHeader.tex);
}

//_____________________________________________________________________
//
// Factory-method.
//

GLFont * GLFont::NewL (const TDesC & aFilename)
{
   GLFont* f = new (ELeave) GLFont();
   CleanupStack::PushL (f);

   f->ConstructL (aFilename);

   CleanupStack::Pop ();
   return f;
}

//_____________________________________________________________________
//
// Second part of the two-phase construction.
//

void GLFont::ConstructL (const TDesC & aFilename)
{
   // Destroy the old font if there was one, just to be safe
   Destroy();

   // Open file session with server
   RFs session;
   User::LeaveIfError (session.Connect());
   CleanupClosePushL (session);

   // retrieve private application folder
   TFileName path;
   session.PrivatePath (path);

   // retrieve full application path on device
   #ifndef __WINS__
      TFileName appFullName =
      CEikonEnv::Static()->EikAppUi()->Application()->AppFullName();

      TParse parse;
      parse.Set (appFullName, NULL, NULL);
      path.Insert (0, parse.Drive());
   #endif

   // update filename with full path
   TFileName fullFilename (path);
   fullFilename.Append (aFilename);

   // load file
   LoadFileL (session, fullFilename);

   // close server session
   CleanupStack::PopAndDestroy();

}

//_____________________________________________________________________
//
// Loads the font file.
//

void GLFont::LoadFileL (RFs & aFs, const TDesC & aFilename)
{
   // Open input file
   RFileReadStream readStream;

   User::LeaveIfError (readStream.Open (aFs, aFilename, EFileRead));
   readStream.PushL();

   // Read the iHeader from file
   GLFontHeaderFile headerFile;

   headerFile.tex       = readStream.ReadInt32L ();
   headerFile.texWidth  = readStream.ReadInt32L();
   headerFile.texHeight = readStream.ReadInt32L();
   headerFile.startChar = readStream.ReadInt32L();
   headerFile.endChar   = readStream.ReadInt32L();
   headerFile.chars     = readStream.ReadUint32L();

   // copy iHeader file to actual iHeader
   iHeader.texWidth  = headerFile.texWidth;
   iHeader.texHeight = headerFile.texHeight;
   iHeader.startChar = headerFile.startChar;
   iHeader.endChar   = headerFile.endChar;

   // Allocate space for character array
   TInt numChars = iHeader.endChar - iHeader.startChar + 1;
   iHeader.chars = new (ELeave) GLFontChar [numChars];

   // Read character array
   for (TInt i = 0; i < numChars; ++i)
   {
      iHeader.chars [i].dx  =  FloatToFixed (readStream.ReadReal32L () );
      iHeader.chars [i].dy  =  FloatToFixed (readStream.ReadReal32L () );
      iHeader.chars [i].tx1 =  FloatToFixed (readStream.ReadReal32L () );
      iHeader.chars [i].ty1 =  FloatToFixed (readStream.ReadReal32L () );
      iHeader.chars [i].tx2 =  FloatToFixed (readStream.ReadReal32L () );
      iHeader.chars [i].ty2 =  FloatToFixed (readStream.ReadReal32L () );
   }


   // Read texture pixel data
   TInt numTexBytes = iHeader.texWidth * iHeader.texHeight * 2;
   TUint8 * texBytes = new (ELeave) TUint8 [numTexBytes]; CleanupStack::PushL (texBytes);

   readStream.ReadL (texBytes, numTexBytes);

   // Create OpenGL texture
   glBindTexture   (GL_TEXTURE_2D, iHeader.tex);
   glTexParameterf (GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
   glTexParameterf (GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
   glTexParameterf (GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
   glTexParameterf (GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
```


(C) Copyright Nokia 2013. All rights reserved.

```
   glTexEnvf (GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);

   glTexImage2D (GL_TEXTURE_2D,
                 0,
                 GL_LUMINANCE_ALPHA,
                 iHeader.texWidth,
                 iHeader.texHeight,
                 0,
                 GL_LUMINANCE_ALPHA,
                 GL_UNSIGNED_BYTE,
                 (GLvoid *)texBytes);

   // Free texture pixels memory
   CleanupStack::Pop ();
   delete [] texBytes;

   // Close input file
   readStream.Close();
   readStream.Pop();
}

//_____________________________________________________________________________
//
// Font destruction.
//

void GLFont::Destroy ()
{
   //Delete the character array if necessary
   if (iHeader.chars)
   {
      delete [] iHeader.chars;
      iHeader.chars = 0;
   }
}

//_____________________________________________________________________________
//
// Retrieves the texture dimensions.
//

void GLFont::GetTexSize (TInt & aWidth, TInt & aHeight)
{
   aWidth  = iHeader.texWidth;
   aHeight = iHeader.texHeight;
}

//_____________________________________________________________________________
//
// Retrieves the character interval.
//

void GLFont::GetCharInterval (TInt & aStart, TInt & aEnd)
{
   aStart = iHeader.startChar;
   aEnd   = iHeader.endChar;
}

//_____________________________________________________________________________
//
// Retrieves the dimensions of a character.
//

void GLFont::GetCharSize (TText8 aChar, TInt & aWidth, TInt aHeight)
{
   // Make sure character is in range
   if (aChar < iHeader.startChar || aChar > iHeader.endChar)
   {
      // Not a valid character, so it obviously has no size
      aWidth  = 0;
      aHeight = 0;
   }
   else
   {
      GLFontChar* fontChar;

      // Retrieve character size
      fontChar = & iHeader.chars [aChar - iHeader.startChar];
      aWidth  = FixedToInt (MultiplyFixed (fontChar->dx, IntToFixed (iHeader.texWidth) ) );
      aHeight = FixedToInt (MultiplyFixed (fontChar->dy, IntToFixed (iHeader.texHeight) ) );
   }
}

//_____________________________________________________________________________
//
// Retrieves the dimensions of a string.
//

void GLFont::GetStringSize (const TDesC8 & aText, TInt & aWidth, TInt & aHeight)
{
   // Height is the same for now...might change in future
   aHeight = FixedToInt (MultiplyFixed (iHeader.chars [iHeader.startChar].dy, IntToFixed (iHeader.texHeight) ) );

   // texWidth as fixed
   const GLfixed texWidthx = IntToFixed (iHeader.texWidth);

   // Calculate width of string
   GLfixed widthx = 0;
   for (TInt i = 0; i < aText.Length(); i++)
   {
      // Make sure character is in range
      const TText8 c = aText [i];
      if (c < iHeader.startChar || c > iHeader.endChar)
         continue;

      // Get pointer to glFont character
      const GLFontChar* fontChar = & iHeader.chars [c - iHeader.startChar];

      // Get width and height
      widthx += MultiplyFixed (fontChar->dx, texWidthx);
   }

   // Save width
   aWidth = FixedToInt (widthx);
}

//_____________________________________________________________________________
//
// Renders a string. Reference point is top-left.
//
```


(C) Copyright Nokia 2013. All rights reserved.

```
void GLFont::DrawString (const TDesC8 & aText, GLfixed aX, GLfixed aY)
{
   // vertex arrays to render the string
   GLfixed  vertices [4*2];
   GLfixed texCoords [4*2];
   const GLubyte indices [] = {1, 2, 0, 3};

   glVertexPointer  (2, GL_FIXED, 0, vertices);
   glTexCoordPointer (2, GL_FIXED, 0, texCoords);

   // Bind texture
   glBindTexture (GL_TEXTURE_2D, iHeader.tex);

   // Loop through characters
   for (TInt i = 0; i < aText.Length(); i++)
   {
      // Make sure character is in range
      TText8 c = aText [i];
      if (c < iHeader.startChar || c > iHeader.endChar)
         continue;

      // Get pointer to glFont character
      GLFontChar* fontChar = &iHeader.chars [c - iHeader.startChar];

      // Get width and height
      GLfixed width =  MultiplyFixed (fontChar->dx, IntToFixed (iHeader.texWidth) );
      GLfixed height = MultiplyFixed (fontChar->dy, IntToFixed (iHeader.texHeight) );

      // Specify texture coordinates
      texCoords [0] = fontChar->tx1; texCoords [1] = fontChar->ty1;
      texCoords [2] = fontChar->tx1; texCoords [3] = fontChar->ty2;

      texCoords [4] = fontChar->tx2; texCoords [5] = fontChar->ty2;
      texCoords [6] = fontChar->tx2; texCoords [7] = fontChar->ty1;

      // and vertices
      vertices [0] = aX;         vertices [1] = aY;
      vertices [2] = aX;         vertices [3] = aY - height;

      vertices [4] = aX + width; vertices [5] = aY - height;
      vertices [6] = aX + width; vertices [7] = aY;

      // draw
      glDrawElements (GL_TRIANGLE_STRIP, 4, GL_UNSIGNED_BYTE, indices);

      // Move to next character
      aX += width;
   }

}
```

And here is the fixed-point math operations required by this code:

```
//-----------------------------
#ifndef FIXED_MATH_H_
#define FIXED_MATH_H_
//-----------------------------

// INCLUDES
#include <e32base.h>
#include <e32std.h>
#include <e32math.h>
#include <GLES/gl.h>

// FUNCTIONS

inline GLfixed IntToFixed (GLint aValue)
{ return aValue << 16; }


inline GLfixed FloatToFixed (GLfloat aValue)
{ return (GLfixed) (aValue * 65536.0f); }


inline GLint FixedToInt (GLfixed aValue)
{ return aValue >> 16; }


inline GLfloat FixedToFloat (GLfixed aValue)
{ return (GLfloat) (aValue * (1 / 65536.0f)); }


inline GLfixed MultiplyFixed (GLfixed op1, GLfixed op2)
{
  TInt64 r = (TInt64)op1 * (TInt64)op2;
  return (GLfixed) (r >> 16);
}


//-----------------------------
#endif
```

Notes and limitations

This font class is intended to render 2D text. So, it is important to set up the required orthographic projection before rendering. The following code is an example of how to do this:

```
// Rect() is a method that returns the current drawing rectangle.
// The (0,0) point will be the on bottom-left corner.

glViewport (0, 0, Rect().Width(), Rect().Height());

glMatrixMode (GL_PROJECTION);
  glLoadIdentity ();
  glOrthox (0, IntToFixed (Rect().Width()), 0, IntToFixed (Rect().Height()), -1, 1);

 glMatrixMode (GL_MODELVIEW);
   glLoadIdentity ();
```


(C) Copyright Nokia 2013. All rights reserved.

Also, the `BeginDraw()` and `EndDraw()` methods (or equivalent code) should be called to set up required states for rendering. For example, if blending is not enabled, the quad corresponding to the character becomes noticeable.

Currently, this class supports 8 bit descriptors only. The class should be extended to support 16 bits descriptors and resource strings.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.