リスト1.3 | myGlsl.h(OpenGL/GLSLインターフェイス)

```
GLuint vertexShader, fragmentShader;
//shader fileを読み込みコンパイルする
void readShaderCompile(GLuint shader, const char *file)
{
  FILE *fp;
  char *buf;
  GLsizei size, len;
  GLint compiled;

  // ファイルを開く
  fp = fopen(file, "rb");
  if(!fp) printf("ファイルを開くことができません%s¥n", file);

  //ファイルの末尾に移動し現在位置を得る
  fseek(fp, 0, SEEK_END);
  size = ftell(fp);//ファイルサイズを取得

  // ファイルサイズのメモリを確保
  buf = (GLchar *)malloc(size);
  if (buf == NULL) {
    printf("メモリが確保できませんでした¥n");
  }

  // ファイルを先頭から読み込む
  fseek(fp, 0, SEEK_SET);
  fread(buf, 1, size, fp);
  //シェーダ・オブジェクトにプログラムをセット
  glShaderSource(shader, 1, (const GLchar **)&buf, &size);
  //シェーダ読み込み領域の解放
  free(buf);
  fclose(fp);

  // シェーダのコンパイル
  glCompileShader(shader);
  glGetShaderiv( shader, GL_COMPILE_STATUS, &compiled );

  if ( compiled == GL_FALSE )
  {
    printf( "コンパイルできませんでした!!: %s ¥n ", file);
    glGetProgramiv( shader, GL_INFO_LOG_LENGTH, &size );
    if ( size > 0 )
    {
      buf = (char *)malloc(size);
      glGetShaderInfoLog( shader, size, &len, buf);
      printf(buf);
      free(buf);
    }
  }
}
//リンクする
void link( GLuint prog )
{
  GLsizei       size, len;
  GLint linked;
  char *infoLog ;

  glLinkProgram( prog );
  glGetProgramiv( prog, GL_LINK_STATUS, &linked );
  if ( linked == GL_FALSE )
  {
    printf("リンクできませんでした!! ¥n");
    glGetProgramiv( prog, GL_INFO_LOG_LENGTH, &size );
    if ( size > 0 )
```

```
    {
      infoLog = (char *)malloc(size);
      glGetProgramInfoLog( prog, size, &len, infoLog );
      printf(infoLog);
      free(infoLog);
    }
  }
}

void initGlsl(GLuint *program, const char *vertexFile)
{
  //glewの初期化
  GLenum err = glewInit();
  if (err != GLEW_OK)
  {
    printf("Error: %s¥n", glewGetErrorString(err));
  }
  // GPU,OpenGL情報
  printf("VENDOR= %s ¥n", glGetString(GL_VENDOR));
  printf("GPU= %s ¥n", glGetString(GL_RENDERER));
  printf("OpenGL= %s ¥n", glGetString(GL_VERSION));
  printf("GLSL= %s ¥n", glGetString(GL_SHADING_LANGUAGE_VERSION));
  //シェーダーオブジェクトの作成
  vertexShader = glCreateShader(GL_VERTEX_SHADER);
  //シェーダーの読み込みとコンパイル
  readShaderCompile(vertexShader, vertexFile);
  // シェーダ・プログラムの作成
  *program = glCreateProgram();
  // シェーダ・オブジェクトをシェーダ・プログラムに関連付ける
  glAttachShader(*program, vertexShader);
  // シェーダ・オブジェクトの削除
  glDeleteShader(vertexShader);
  // シェーダ・プログラムのリンク
  link(*program);
}

void initGlsl(GLuint *program, const char *vertexFile, const char *fragmentFile)
{
  //glewの初期化
  GLenum err = glewInit();
  if (err != GLEW_OK)
  {
    printf("Error: %s¥n", glewGetErrorString(err));
  }
  // GPU,OpenGL情報
  printf("VENDOR= %s ¥n", glGetString(GL_VENDOR));
  printf("GPU= %s ¥n", glGetString(GL_RENDERER));
  printf("OpenGL= %s ¥n", glGetString(GL_VERSION));
  printf("GLSL= %s ¥n", glGetString(GL_SHADING_LANGUAGE_VERSION));
  //シェーダーオブジェクトの作成
  vertexShader = glCreateShader(GL_VERTEX_SHADER);
  fragmentShader = glCreateShader(GL_FRAGMENT_SHADER);
  //シェーダーの読み込みとコンパイル
  readShaderCompile(vertexShader, vertexFile);
  readShaderCompile(fragmentShader, fragmentFile);
  // プログラム・オブジェクトの作成
  *program = glCreateProgram();
  // シェーダ・オブジェクトをシェーダ・プログラムに関連付ける
  glAttachShader(*program, vertexShader);
  glAttachShader(*program, fragmentShader);
  // シェーダ・オブジェクトの削除
  glDeleteShader(vertexShader);
  glDeleteShader(fragmentShader);
  // シェーダ・プログラムのリンク
  link(*program);
```

```
}
```