情報メディア実験 III・レイトレーシングによるコンピュータグラフィクス入門
課題用プログラムのアップデートについて

2011/12/21

担当: 金森由博 (kanamori@cs.tsukuba.ac.jp)

機能追加やバグフィックスなどのために、課題用のプログラムを更新しました。面倒をかけて申し訳ありませんが、**新しいプログラムのソースコードをダウンロードし、「1. コードの移植手順」に従って、これまで自分で書いたコードを古いソースコードから移植してください**。新しいプログラムの使い方や新機能については「2. 新機能」を見てください。なお、Windows 以外の環境ではコンパイルできません。**Windows 以外の OS で開発している人は、「3. Windows 以外の OS で開発する場合」を見てください**。

１．コードの移植手順

移植が必要なソースコード・関数は以下の通りです。**ひとつ移植したら、コンパイルが通るかを確認し、テスト用のシーンファイル（※以前のシーンファイルとは書式が異なります）を使ってレンダリングした結果がサンプル画像と一致するかどうか、確認してください**。シーンファイルは Config フォルダの中、サンプル画像は NewScreenshots フォルダの中にあります。

① Triangle.cpp の hit および shadowHit 関数

新しい課題の都合で、変数名の変更および新しい変数の追加を行いました。自分で書いたコードをコピペしたのち、次の修正をお願いします。

・頂点の座標を表す変数 v0, v1, v2 を、それぞれ m_Vertices[0], m_Vertices[1], m_Vertices[2] に修正してください。

・法線ベクトルを頂点ごとに持たせるように変更したので、hit 関数で計算する法線ベクトルは、それらの線形補間として計算するように修正してください。具体的には、重心座標 α, β, γ (レジュメ「レイと三角形の交差判定」を参照) および、新たに追加された法線ベクトルを表すメンバ変数 m_Normals[0], m_Normals[1], m_Normals[2] を使って、交点での法線ベクトルを

```
vec3 n = α * m_Normals[0] + β * m_Normals[1] + γ * m_Normals[2];
record.normal = normalize( n );
```

と計算してください。上記の α, β, γ の変数名は自分で定義したものに置き換えてください。

・同様に、頂点ごとにテクスチャ座標を持たせるようにしたので、メンバ変数 m_Texcoords[0], m_Texcoords[1], m_Texcoords[2] を用いて、交点でのテクスチャ座標を

```
vec2 texCoord = α * m_Texcoords[0] + β * m_Texcoords[1] + γ * m_Texcoords[2];
record.tex_coords.set( texCoord.x, texCoord.y, 0.f );
```

と計算してください。

② DiffuseMaterial.h および PhongMaterial.h の calcReflectedLight 関数

自分で書いたコードをそのままコピペしてください。なお、これらのヘッダファイルでは、わざわざ MyAlgebra::vec3 や MyAlgebra::Real などと長い型名を書かかずに、単に vec3 や Real と書けるように

しました (Material.h で typedef しています)。MyAlgebra:: が残っていても問題ありません。

③ RecursiveRayTracer.cpp の traceRec 関数

関数の中身が一部（変数名など）書き変わっていますが、反射・屈折の部分は自分で書いたコードをそのままコピペしてもらえば大丈夫なはずです。

④ テクスチャマッピング関係

以下のファイルの各関数について、それぞれ自分で書いたコードをそのままコピペしてください。

- TexturedMaterial.h の calcReflectedLight 関数 (※MyAlgebra:: は省略可)
- TexturedPlane.cpp の hit 関数
- TexturedSphere.cpp の hit 関数
- SolidTexturedSphere.cpp の hit 関数

⑤ アンチエリアシング関係

以下のファイルの各関数について、それぞれ自分で書いたコードをそのままコピペしてください。

- RegularNxNSampler.h の sample 関数
- JitterSampler.h の sample 関数
- MultiJitterSampler.h の sample 関数

２．新機能

- 画面をマウスでドラッグすることで、カメラを操作できるようにしました。左ドラッグでカメラの回転、中ドラッグでズームイン/ズームアウト、右ドラッグで左右に平行移動します。
- シーンを別の視点から、OpenGL で表示するウィンドウを追加しました。カメラが空間中をどのように移動するかを確認することができます。
- シーンファイルでより複雑な記述を可能にしました。書式は、これまでのものとは互換性がなく、講義「インタラクティブ CG」で使われているシーンファイルと似ています（同じではありません）。
- デバッグ用に物体表面の法線ベクトルを疑似カラーで表示できるようにしました。
- 三角形メッシュの読み込みをサポートしました。Wavefront OBJ 形式 (拡張子は obj) および PLY 形式 (拡張子は ply) を読み込めます。ただし、各面の頂点数が 3 でない場合（つまり三角形でない場合）、プログラムがクラッシュすることがあります。
- 物体を回転・拡大縮小・平行移動して配置できるよう、座標変換をサポートしました。
- 表示できる立体形状として、直方体 (Axis-Aligned Box) をサポートしました。
- 一様グリッド (Uniform Grid) による高速化をサポートしました。
- メタボールのレンダリングをサポートしました。
- Image-based Lighting (IBL) において、環境マップのサンプル数を任意に指定できるようにしました。
- バッチ処理をサポートしました。ウィンドウを表示することなく、指定されたシーンファイルを読み込んで画像を出力します。連番のついたファイル名の画像を出力することで、アニメーションを作成することができます。

3. Windows 以外の OS で開発する場合

現在のプログラムは、一部の機能（環境マップのサンプリング）のために Visual Studio 2008 でプリコンパイルした lib ファイルを使っているので、MacOS や Linux ではコンパイルできません。必要なソースコードを別途提供しますので、相談してください。取り急ぎ、レジュメ「環境マッピング」の内容までは、古いソースコードの EnvironmentMap.h/cpp を新しいもので置き換えれば対応できるので、そのようにして実装を進めてください。なお…実験開始のときにコンパイルが通らなくて修正した部分は、再度修正してもらうことになるかと思います…すみません。