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
- レイトーシングにおける数値誤差について -

金森 由博*

2012 年 12 月 12 日

1　レイトーシングでよくある数値誤差の原因と対策

雛形となるレイトレーシングのプログラムでは、実数の型として、倍精度浮動小数点数 (double 型) ではなく単精度浮動小数点数 (float 型) を使って計算している。これは float 型での計算の方が高速なためである。しかし double 型に比べて数値誤差の影響を受けやすい。このレジュメでは、レイトレーシングの計算でよくある数値誤差の原因とその対策について紹介する。

1.1　初歩的なミス

■変数の初期化忘れ:　C/C++ 言語では、変数を初期化しないとデタラメな値が入っている[*1]。そのデタラメな値を計算に使えば結果ももちろんデタラメになる。ちゃんと初期化すること。

■ゼロ除算している:　ゼロで除算すると計算結果は #IND (invalid division) という値になってしまう。除算の分母になる数について、絶対値が非常に小さい値になっていたら、場合分けして計算すること。

■ベクトルを正規化し忘れている:　計算に単位ベクトルを使うべきところに、正規化されていない (大きさが 1 でない) ベクトルを使ってしまわないように。方向ベクトルは正規化が必要になることが多い。なお、ゼロベクトルの場合、ベクトル・マトリクスライブラリの正規化関数 normalize を使うと、関数の内部でゼロ除算が発生してしまうことに注意。

■int 型変数による除算:　int 型変数で除算するとき、float 型にキャストしないと、正しい除算結果が得られない。以下に悪い例・正しい例を示す。

int 型変数による除算

```
int N = 10;
float invN = 1/N;              // ダメ。計算結果は 0 になる
int invN2 = 1/(float)N;        // これもダメ。計算結果を int 型に格納すると 0 になる
float invN3 = 1/(float)N;      // OK。計算結果は 0.1 になる
```

* kanamori@cs.tsukuba.ac.jp

*1 ただし Visual Studio などの統合開発環境 (IDE) のデバッグモードでは初期化していない変数はゼロで初期化される。

■int 型へのキャストの振る舞い:　テクスチャ座標の計算やバイリニア補間など、浮動小数点数 x の小数部分を計算したい場合がある。x の小数部分とは、数式で書くと $x - \lfloor x \rfloor$ である ($\lfloor x \rfloor$ は x を超えない最大の整数)。$\lfloor x \rfloor$ を計算するには、$x \geq 0$ ならば int 型へのキャスト $(\mathrm{int})x$ で求められる。しかし $x < 0$ の場合、int 型にキャストするとゼロに近い整数に切り上げられてしまう。int 型へのキャストではなく、数学関数の floor 関数を使えば正しく $\lfloor x \rfloor$ を計算できる。

2　数学関数の使い方について

■Arctan の計算には atan 関数より atan2 関数:　Arctan の計算には、引数の数が 1 つの atan 関数が使えるが、これは $\mathrm{Arctan}(y/x)$ の計算において $x = 0$ となるとき使えない。代わりに、引数の数が 2 つの atan2 関数を使えば、atan2(y,x) という風に計算が可能である。また返り値の範囲は、atan 関数は $[-\pi/2, \pi/2]$ だが、atan2 関数は $[-\pi, \pi]$ となる。

■sqrt 関数の引数に負の数を入れない:　理論上は計算結果が必ずゼロ以上になるような場合でも、数値誤差により、絶対値の非常に小さな負の数になる場合がある。これを sqrt 関数の引数に入れてしまうと、計算結果は不正な値 (#QNAN) になる。これを防ぐには、sqrt 関数の引数として引き算の計算結果を入れる場合に、負の数になっていないか事前に調べる。負の数になっていて、かつ、理論上の計算結果は必ずゼロ以上なら、その計算結果はゼロにしてしまってよい。

■acos 関数は引数が 1 または-1 に近いと精度が悪い:　二つの単位ベクトルのなす角を計算するには、それら 2 つのベクトルの内積を acos 関数の引数とすればよい。ただし acos 関数の引数が 1 または-1 に近い場合、すなわち、なす角が 0 または π に近い場合、acos 関数は精度が悪くなる。これを回避するため、ベクトル・マトリクスライブラリには angle 関数というものを用意しているので、そちらを使ってほしい。参考までに、この angle 関数はより数値的に安定な atan2(sinθ, cosθ) を計算している (θ は 2 つの単位ベクトルのなす角)。詳しくはソースコードを参照してほしい。