

! この記事は最終更新日から5年以上が経過しています。



@satoruhiga

投稿日 2014年06月06日 更新日 2014年06月06日

openFrameworksパフォーマンスチューニング



openFrameworks

ツール等

Instruments.app

Xcodeに付属のプロファイラ

アプリの稼働中にどの部分がボトルネックになっているかを確認できたり、頑張ればメモリリークも発見できる時がある。

自分が説明するのもアレなので詳しくは[ここ](#)とか...

- 得に何もしなくてもXcodeのメニューから起動できるので楽チン
- Releaseビルドになっていると関数名のシンボルが消えて何のコッチャになってしまうので注意 (最近ではRelease/Debugの速度もかわらないのでずっとDebugでもいいかな、と誤ってしまいます)




107



106



- Product > Profile  で起動
- Time Profiler を選択
- 左側の Call Tree のリストの中にあるチェックボックスを **上から4つ入れる**
- メソッド/関数の処理負荷の高い順にリストがソートされるので眺める

という流れです

ofxTimeMeasurements

GitHub - armadillu/ofxTimeMeasurements: OpenFrameworks add-on to easily measure execution times on different parts of your code.

<https://github.com>

armadillu/
ofxTimeMeasurements

OpenFrameworks add-on to easily measure execution times on different parts of your code.



4 Contributors 0 Issues 91 Stars 18 Forks

導入が楽で素朴な機能のいい奴

- 単純な仕組みなので余分なオーバーヘッドが入りこむ余地がない
- 関数の時間など計測する時に明示的にSTART/STOPのマクロを仕込まないといけな
い、が、逆に変な挙動もないのでわかりやすい
- ただSTART/STOP間の時間をはかってるだけなのでDebug/Releaseビルド関係なく計
測できる
- FlashとかThree.jsの左上のアレみたいな機能がついててればもっといいのだが

自力

```
void ofApp::draw()
{
    // ....

    stringstream ss;
    ss << "framerate: " << ofToString(ofGetFrameRate(), 0);
    ofDrawBitmapString(ss.str(), 10, 20);
}
```

- 何の導入もないので一番楽
- draw/updateの境なく1フレーム単位でのレートになるのでofxTimeMeasurementsと比較して切り分けは多少雑か
- `ofSetWindowTitle(ofToString(ofGetFrameRate()), 0);` というのもよくやるけど **60fps以上で動かす時にこれが原因でフレームレートが激落ち君になったことがあるので注意** (おそらくOS側UIアップデートでウィンドウタイトル変更の処理をブロックしてしまうタイミングがあるのだと思われます)

方針

とりあえず自力でやるのが手間もなくていいと思いますが、ofxTimeMeasurements導入も手間としてはそんなにかわらないので最初から入れててもいいかな... という気もします。

注意点としては `ofSetVerticalSync(true);` になっていると処理がどんなに早く終わってもディスプレイの垂直同期を待ってからの描画になるので60fpsで頭打ちになります。

この場合十分に処理が早いといいのですが、何かの問題で1フレームあたりの処理にコンスタントに $1000\text{ms}/60 = 16.6666\text{ms}$ 以上かかってしまった場合、本来描画するはずの垂直同期に間にあわなくなり最悪次の垂直同期まで待つことになるので急にレートが30fpsまで落ちることがあります。(最近ないけど、昔はよくありました)

なので見た目のブラッシュアップまでも行かないベースシステムの開発中の時は:

```
void ofApp::setup()
{
    ofSetVerticalSync(false);
    ofSetFrameRate(0);

    // ...
}
```

ので最近のマイブームです。

見た目的な部分に入ってきた場合には垂直同期が入っていないとティアリングがおきるので入れといてください

ofローカルなパフォーマンスTIPS

- `ofGetElapsedTime()` は呼びすぎると結構処理が重い。フレームあたりで一意の時間があればいいだけであれば単純にグローバル変数にキャッシュしてそれを使ったほうがいい
- `ofToString()`、`ofSplitString()` 等々も内部は `std::stringstream` を使っていたりするので結構重い。`sscanf()` は比較的高速なのでそっちにするとよい
- `ofImage::getColor()` / `ofImage::setColor()` は `ofColor` のコピーをやりとしているので辛い。`ofImage::getPixels()` でポインタをもらってガリガリやったほうが倍ぐらい高速
- `ofMatrix4x4` も地味に最適化されてないので遅い部類。以前 `mat4 * vec3` の計算を大量にしなければいけない時はその部分だけ `libSIMDx86` に置き換えたりしましたがかなり高速化できました。

もっとあったっけな、思い出したら追記します...



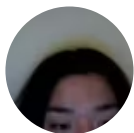
新規登録して、もっと便利にQiitaを使ってみよう

1. あなたにマッチした記事をお届けします
2. 便利な情報をあとで効率的に読み返せます

[ログインすると使える機能について](#)

新規登録

ログイン



@satoruhiga

フォロー



- 

[Redacted text]
- 

[Redacted text]
- 

[Redacted text]
- 

[Redacted text]
- 

[Redacted text]

コメント

この記事にコメントはありません。



新規登録

すでにアカウントを持っている方は[ログイン](#)

Qiita

How developers code is here.

© 2011-2022 Qiita Inc.

ガイドとヘルプ

About

利用規約

プライバシーポリシー

ガイドライン

デザインガイドライン

ご意見

ヘルプ

広告掲載

コンテンツ

リリースノート

公式イベント

公式コラム


募集

アドベントカレンダー

Qiita 表彰プログラム


API

SNS

 Qiita（キータ）公式

 Qiita マイルストーン

 Qiita 人気の投稿

 Qiita（キータ）公式

Qiita 関連サービス

Qiita Team

Qiita Jobs

Qiita Zine

Qiita 公式ショップ

運営

運営会社

採用情報

Qiita Blog