空き CPU 時間を活用した地図の再描画手法の提案*

7C－3

溝口文雄*　木下信幸†

東京理科大学 理工学部‡

1 はじめに

ハードディスクやCD-ROMなどに格納された地図情報を読み込み，統計処理や道案内など各種のサービスを行なう地図情報処理システム（GIS）が普及している．

地図情報を画面に表示することは（図1），これらのシステムにおける最も基本的なサービスの１つである．しかし，既存の表示法では，再描画のスピードが遅く，広範な領域をスクロールする場合，地図の欠落やチラツキが目立ち，必ずしも滑らかなスクロールが行なわれているわけではなかった．

そこで本稿では，CPUの空き時間にこれから描画するべき領域情報を先読みし，データ読み込みの際のオーバーヘッドを減少させ，再描画のスピードを早めることにより，滑らかなスクロールを実現する再描画手法を提案する．これらはJava言語のマルチスレッド機能を用いて実現する．

2 再描画時間

2.1 先読みがない場合

地図の再描画時間は(1)ハードディスクやCD-ROM上の，数値地図が格納されているデータファイルから，地図情報を読み込みメモリ上の配列に格納する時間(2)配列に格納されている地図情報を加工し，画面に描画・表示する時間の２つに分けることができる．このとき再描画全体の時間を１００とすると(1)，(2)の時間が再描画時間に占める割合は(1)が６１，(2)が３９となる．

2.2 先読みがある場合

地図の再描画時間は(1')先読みが完全に実施された後に再描画を行なう場合は，前述の(2)に相当する時間のみとなる．(2')先読みが不完全な状態で再描画を行なう場合は，前述の(1')における残りのデータを読み込む時間と，(2)に相当する時間を加えたものとなる．

*Reducing idle CPU Time on buffering the Giographical Data for map drawing

†Fumio MIZOGUCHI, Nobuyuki KINOSHITA

‡Faculty of Sci. and Tech. Science University of Tokyo

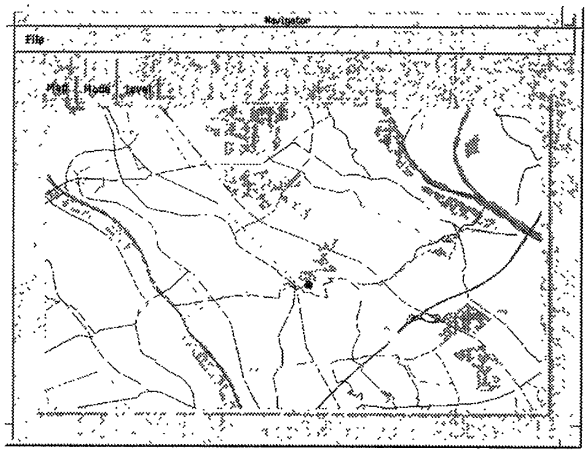

図 1: 地図表示システム

3 先読みの手法

3.1 CPU の空き時間の抽出

制御スレッドには実行優先順位があるので，相対的な実行速度に変化をもたせることが可能である．つまり，優先順位が高くなるほど実行されやすく，また低くなるほど実行が後回しになる．地図情報の先読み処理においては，このようなスレッドの特性を利用することでCPUの空き時間の抽出を実現している．以下に先読みスレッドの特徴を挙げる．

- 独立したスレッド
  地図情報の先読み機能は描画・表示処理を行なうスレッドとは独立した別のスレッドとして定義する

- 極端に小さいプライオリティ
  先読み処理スレッドは他の実行スレッドよりも極端に低い優先順位を設定する

- ポインタの保持
  先読みした地図情報は配列に格納される．このと

き，どこまで格納したかのポインタを保持し，先読み途中で再描画命令が発生した場合に備える．

3.2 先読みデータの予測

画面のスクロールには方向と幅がある．本稿ではスクロールの方向は上下，左右など8方向，スクロールの幅は0から300までピクセル単位の指定が行なえるようになっている．したがって，先読みすべき図葉はスクロール方向×スクロール幅によって定まる画面上の座標から予測している．予測された図葉に対応する数値地図上の図葉情報を先読みする．

3.3 先読みのアルゴリズム

地図情報の先読み処理は，他の処理とは独立したスレッドを用い，CPUタイムの空き時間にデータを読み込むことで，システムの応答性を損なうことなく実現されている．以下にそのアルゴリズムを示す．

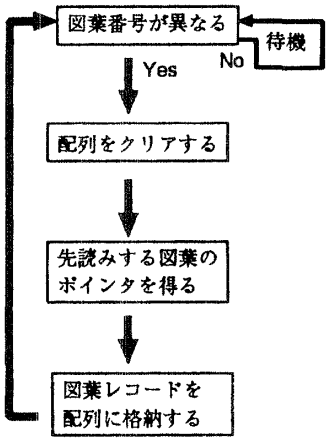

図 2: 先読みのアルゴリズム

3.4 先読みのプログラム

先読み部分は優先順位の低いスレッドとして実装される．優先順位の設定は先読みスレッドを初期化するときに行なう．以下はその初期化メソッドでsetPriority()により，先読みスレッドの優先順位を最小の優先順位よりも2つ大きい優先順位に設定している．

```
public preRead() {
    super("preRead");
    ...
    setPriority(Thread.MIN_PRIORITY + 2);
    this.start();
}
```

また，図2にある先読みを終えて，次の先読みまで待機する箇所は，wait()メソッドとフラグを利用することで，先読みスレッドと描画・表示スレッドの同期を行なっている．

```
private final synchronized void waitForNext() {
        try {
            while (!ChangeNumber)
              wait();
        } catch (InterruptedException e) {}

    }
```

4 比較，評価

本稿では，先読みスレッドを実装した地図描画システムと，スクロールするたびに地図情報を読み込む，先読みスレッドを実装していないシステムの2つについて比較・評価を行なった．実験は2つの手法の再描画時間の計測である．図3の計測データは，スクロールする方向，幅をランダムに選択し100回の再描画を行い，それぞれ再描画にかかった時間を計測したものである．縦軸が描画速度を表し単位はミリセカンド，横軸は再描画の回数である．

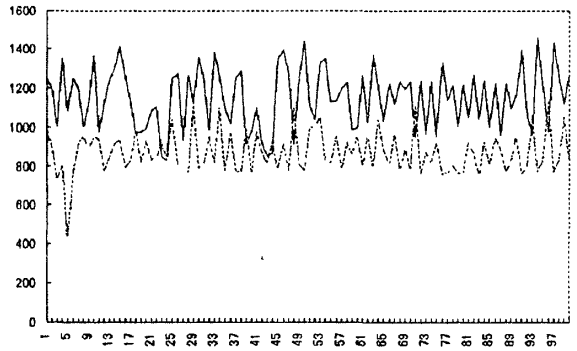

図 3: 再描画時間の比較

グラフからは先読みスレッドを実装した方が再描画を高速に行なえることが読みとれる．時間軸方向のに挙動が激しいのは，単位が微小であるため，計測マシンの負荷に大きく依存していることが原因と考えられる．

5 まとめ

本稿では，Javaスレッドの優先順位の設定により，CPUタイムの空き時間を効率良く利用することで，システムの応答性を損なわないデータの先読みを実現した，数値地図の再描画手法について提案し，単一スレッドの逐次処理よりも高速に再描画できることを示した．

参考文献

[1] 数値地図ユーザーズガイド，建設省国土地理院，(財) 日本地図センター

[2] Solarisカーネルおよびアプリケーションのマルチスレッドアーキテクチャ，日本サン・マイクロシステムズ，1995年8月