災害情報共有インタフェースおよびSDGミドルウエアの開発

上田 真史　　竹内 郁雄
masa-u@nue.ci.i.u-tokyo.ac.jp　　nue@nue.org
電気通信大学大学院電気通信学研究科　　東京大学情報理工学系研究科

概要

災害発生時には白地図が使用されることが多いが、紙地図ベースの作業は煩雑であり情報の混乱も招く。我々は、白地図の電子化を基本として、Single Display Groupware を採用し、多人数同時操作も可能な災害情報共有支援システムの開発および研究を行っている。本稿では、これまでに開発した情報共有支援システムと、システムと同時に開発を行っている汎用 SDG ミドルウエアを紹介する。

1　はじめに

災害発生時の災害対応本部などでは、地図を用いて災害状況をまとめたり、対応指示を行ったりすることが一般的である。地図上で課題を解きながら防災訓練を行っていこうという取り組み [1] もある。これらの作業に大規模な災害対応システムを用いている自治体もあるが、机上に紙の地図を広げて作業を行うことがほとんどである。

紙の地図上での災害対応作業は煩雑であり混乱も多い。また、地図に記入された情報を広く共有したり、二次利用したりすることにも手間が伴う。そこで我々はこの地図を電子化することを基本として採り上げ、災害情報の共有と再利用を促進することを目標とした災害対応支援システムを研究、開発している。

2　SDG

災害対応システム上では多人数が同時に作業を行う。本研究では多人数対応システムの実現に Single Display Groupware (以下 SDG) のスタイルを採用する。SDG は Computer Supported Cooperative Work (CSCW) の一形態であり、その名の通り単一の画面を用いるスタイルのグループウエアである [2]。

SDG は単一の画面を用いるため、必然的にユーザ全員が同じ場所でシステムを使用する。ひとつの場所で共同作業を行うのでユーザ同士が密接な連携作業を行える。この特長は災害対応システムには特に有効であると考えられる。また、画面がひとつあればよいことから、システムの小型化と低コスト化が可能である。

ここでは、SDG として、1 枚の画面と複数のマウスを用いるスタイルを採り上げる。このスタイルは標準的な構成の PC 1 台に USB ハブと USB マウスを追加すればよく、多人数で扱うシステムとしては最も小規模な部類である。災害対応システムというと大規模なものが多いが、大規模システムは運用や保守にコストがかかる。SDG を採用することでコストを削減し、誰にでも取扱いやすく、また開発もしやすいシステムを実現できる。さらに、どこにでもある装置を用いればよいという特徴は、緊急を要する災害対応に適していると考えられる。

Developing User Interfaces for Disaster Information Sharing along with a General-Purpose SDG Middleware
Masafumi Ueda, Graduate School of Electro-Communications, University of Electo-Communications
Ikuo Takeuchi, Graduate School of Information Science and Technology, The University of Tokyo

3　開発のスタイル

SDG を用いた災害情報システムを開発する上で、複数の入力デバイスからのデータを扱う部分は汎用の SDG ミドルウエアとして利用できるよう、モジュール化して開発を行っている。

現在、SDG ミドルウエアのアプリケーションとして開発を行っている災害情報システムは 2 つである。主に、アプリケーションそれぞれの機能実装にあわせて SDG ミドルウエアを強化するという方法で開発を行っているが、ミドルウエアが災害情報システムのスタイルに引きずられないよう、SDG ゲーム等の開発も行っている。

第 4 節と第 5 節では、我々が開発を行っているアプリケーションプログラムと SDG ミドルウエアについてそれぞれ紹介する。

4　アプリケーションデザイン

4.1　避難所インタフェース

災害発生時に避難所で住民の避難状況、安否状況を把握するためのシステムを開発している。避難所では、避難民がどこから避難してきたかを白地図や名簿を用いて把握する作業がある。このシステムはこれを電子的に行うものである。電子的に白地図記入を行うことで、避難情報を即座に管理および伝播できると期待される。また、白地図記入自体は小規模な作業であるため、マウスを複数用いて同時に入力作業を行えるようにすれば、災害直後の早急な情報収集が期待できる。

本システムのメリットは SDG によりどこでも運用可能なことである。災害直後の避難所は、小学校や公民館といった、設備が限られた、災害対策本部を常に設置していない場所であることが多い。そのような場所に従来の大規模な電子情報システムを常に用意しておくのは難しいが、SDG であれば事務室にあるノートパソコンであっても運用が可能である。多人数入力対応も事務室から USB マウスをかき集めればよい。

4.2　災害対策本部インタフェース

災害対策本部においてレスキュー隊など現場の隊員とコミュニケーションをとって情報共有をすることを目的としたインタフェースである。各隊の司令官がそれぞれマウスを持ち、隊の統制や情報管理を行う。

このシステムは前述の避難情報システムや現場の隊員の情報端末と連携して動作するよう設計されている。紙の白地図では災害対策本部と避難所の情報共有は手間がかかる。電子化により情報共有を促進し、連携作業をスムーズにすることができる。

繰り返しになるが、本システムも SDG により簡単に運用できるメリットを備えている。更に将来の展望として、水平面状に設置したテーブルトップディスプレイを使用し、それを皆で囲むスタイルを適用することも計画している。まさに机上の白地図と同様のスタイルが実現できるものと考えている。

4.3　ゲームへの応用

SDG ミドルウエアの汎用性を高めるためのテストケースとして、災害用インタフェースだけでなく、マルチマウスマインスイーパの開発を行った (図 2)。

これはマインスイーパを複数人で行うゲームで、一人用マインスイーパを発展させたルールにしてある。たとえば、地雷のマスを踏むとゲームオーバーになるのではなく、ペナルティ点を課されさらにそのマスから一定時間動けなくするようにした。こうすることによって、マウスを操作してもカーソルはブルブル震えるだけという感覚を味わわせることができる。このルールの追加はミドルウエアにマウスカーソルの移動範囲制御を追加するきっかけとなった。その他にも、地雷除去面積の競り合いにした場合にユーザ同士の干渉はどう発生するのか、など、ミドルウエアを発展させるための考察を行うのに有用なプラットフォームとなった。

多人数用のゲームは専用のコントローラを用いたものが家庭用機器などでも既に実現されていて、これらも広義の SDG とみることができる。パソコン用のゲームで多人数対応のものは

図 1: 避難所インタフェースの画面

ネットワーク経由が主流であるが、SDG を用いることで新たなプレイスタイルを創出できるだろう。特にこのマインスイーパの例では、ゲーム参加者同士が直接会話をする、あるいは他人のアクションを画面上だけでなく直接観測できるマルチモーダルインタラクションが可能である等、ネットワークゲームでは不可能なインタラクションが見られた。

5 ミドルウエア

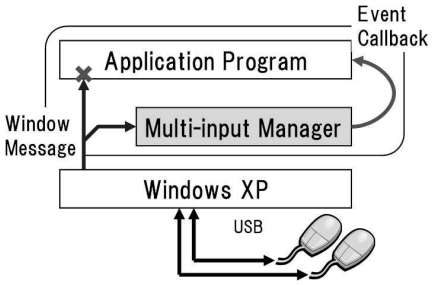

図 3: ミドルウエア機構概念図

我々の開発しているシステムで用いている SDG ミドルウエアは WIN32 API を用い C++ で開発している。DLL としてコンパイルし、アクセス用のクラスを用意することで、C#でも利用できるようにしている。ただし、Windows XP から用意されている API を利用するため、Windows 2000 以前の Windows では利用できない。

現在のところ、マウスやトラックボール等の相対座標入力を行う装置に対応している。ペンタブレット等の絶対座標入力を行う装置やキーボードへの対応は今後の課題である[1] 。

5.1 マルチ入力デバイスの基礎

複数の入力デバイスからのデータを取得するのには Raw Input という API を利用している。

通常 (Raw Input を利用しない場合)、Windows API からは各ウインドウに向けウインドウメッセージの形でユーザ入力のデータが送られるが、このメッセージは計算機に接続された全てのマウスやキーボードを総合したイベント

[1] ペンタブレットは現状でも動作させることはできるが、ペンタブレットの座標から画面座標へのマッピングを行う機構が実装されていない。

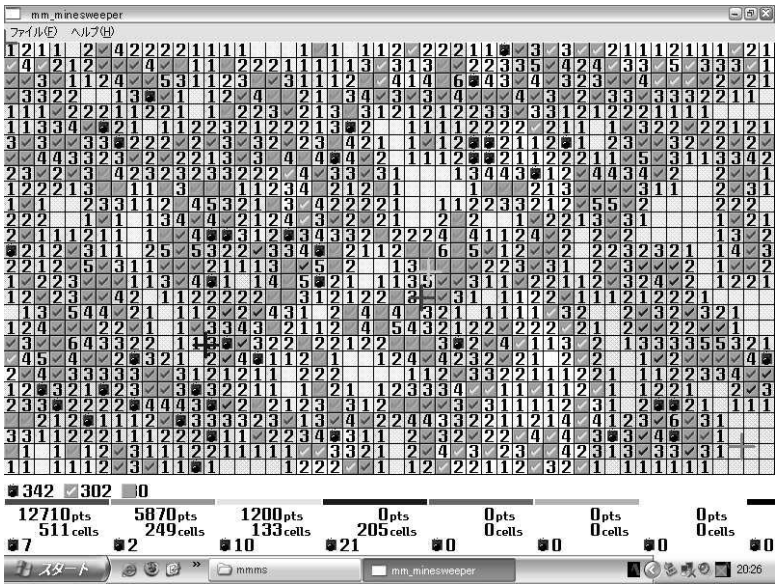

図 2: マルチマウスマインスイーパの画面

であり、入力データと入力デバイスの対応関係が把握できない。特にマウスは全てのマウスの動作を単純に加算したものが 1 台のマウスの動作として送られるため、マルチ入力としての利用は不可能である。

しかし Raw Input API を用いると、入力データからそれを発生させたデバイスの ID を取得できるようになる。ただしこの方法ではマウスやキーボードの入力の生データしか取得できないため、少々レベルの高いイベント、例えばダブルクリックの検知などはアプリケーションプログラム側で処理する必要がある。これらの処理のためにミドルウエアが必要であり、またミドルウエアとして独立させることで、様々なアプリケーションプログラムからマルチ入力を利用しやすくできる。

5.2 本 SDG ミドルウエアの機構

現在開発を行っている SDG ミドルウエアは、Raw Input API に係わるデータ処理を担当し、アプリケーションプログラムがなるべくシングル入力のときと同様に動作できるようにするものである。

具体的には、以下の動作をする。

5.2.1 Raw Input データの解析

前述の通り、Raw Input API からはデバイスの生入力データしか取得できない。例えばマウスの場合は単位時間あたりの移動距離とボタンやホイールの状態変化のみである。アプリケーションプログラムからマウスを利用する場合、利用しやすいのはマウスの現在位置やクリック動作であるので、Raw Input API からのデータをデバイス毎に累算処理し蓄積する。

5.2.2 マウスカーソルの描画

マウスを複数接続してマルチユーザ対応にする場合、無論マウスカーソルもマウスの台数分表示しなければならない。しかしながら Windows が描画するマウスカーソルは 1 つのみであり、かつ全てのマウスを総合した動作をしてしまう。そこでカーソル描画をミドルウエアが行う。

マウスカーソルは単純な十字型をはじめ透過色つきのビットマップ等も使用できる。Windows 標準のカーソルに比べ遜色のない描画表示が可

能である。

5.2.3 マウスカーソルの管理

SDG で問題となる点のひとつとして、多数のユーザが同時に操作を行った場合の操作の競合や干渉がある。この問題の解決の補助として、ミドルウエアレベルでマウスカーソルを管理できるようにしている。現時点では、マウス毎に

- 有効/無効の切替え
- 移動範囲の制限
- スピードの指定
- XY 軸の回転角度の指定

ができる。特に移動範囲の制限は、多様な作業レイアウトに対応できるよう、任意の多角形で指定できるようにしてある。

5.2.4 イベント管理

シングル入力の場合、ユーザの入力があると、Windows からアプリケーションプログラムへウインドウメッセージが送られる。所謂入力イベントというものである。マルチ入力でも同様の実装形態とするのがアプリケーション開発者にとって扱いやすいと考えられる。

ウインドウメッセージは、Windows からアプリケーションプログラムへ `WndProc()` というコールバック関数を通じて送られる。本ミドルウエアでもこれと同様の機構を採用し、アプリケーションプログラムはミドルウエアへコールバック関数を登録し、ミドルウエアからのイベントの受渡しはこのコールバック関数を用いる。

コールバックの際には、ウインドウメッセージと同様のイベントデータにカーソル ID を加えてアプリケーションプログラムへ送信する。カーソル ID 毎、イベント毎に違った関数をコールバックするようにもできる。

5.2.5 子ウインドウの管理

ボタンやメニュー、スクロールバー等、従来のシングル入力用コントロールはそのままではマルチ入力環境で利用できない。Windows ではボタン等もウインドウ (子ウインドウ) という扱いであり、親ウインドウの `WndProc()` に従来のウインドウメッセージが届かなくなると、子ウインドウの `WndProc()` も同じ状況になるためである。同じ理由から、ウインドウの移動や消去もマウスオリエンテッドではできない。

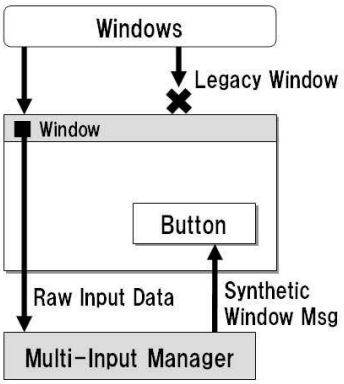

図 4: マルチ入力時の子ウインドウへのメッセージの流れ

この問題は SDG を構築する上で大きな障害となっている。実際、これまでの避難所インタフェースや対策本部インタフェースの開発では、かなりの量のコードをこの問題の解決のために書かねばならなかった。特に避難所では Windows のボタンの代替として動作をするボタンパーツを、その描画から実装するという手段をとった。こういった代替パーツをミドルウエアで用意するのも解決策の一つではあるが、Windows のコントロールウインドウ群は実に多彩であり、全てを網羅するのは現実的ではない。

一方で、従来のコントロールをそのまま使う場合、それらコントロールに向けてウインドウメッセージを合成するなど、何らかの仲介処理が必要である。本ミドルウエアには、子ウインドウを登録しておくと、その子ウインドウの領域内でイベントが起こった場合にウインドウメッセージを合成し、そのメッセージで子ウインドウを駆動する、という機能を実装している (図 4)。特定のデバイスのみについてこの機能を適

用することもできるので、特権を持ったマウスカーソルのみが押せるボタン、なども作成可能である。

ただし、この機構には未だ少々不具合が残っているため、更なる調整が必要である。

5.3 プログラム全体の流れ

あるウインドウについてアプリケーションプログラムが本ミドルウエアを使用するように登録 (register) すると、そのウインドウに Raw Input メッセージが届くようになる。また従来の (legacy な) ウインドウメッセージは届かないようにすることができる。

次に、アプリケーションプログラムは、そのウインドウに届いた Raw Input メッセージを逐一ミドルウエアへ通知する。ミドルウエアはこの Raw Input メッセージを解析し、カーソル描画を行い、イベントが発生した場合はコールバックによりアプリケーションプログラムへ通知する。

シングル入力の場合は、ウインドウの `WndProc()` が入力メッセージを受け付け、適切な処理を行う箇所であるが、本ミドルウエアを利用する場合、`WndProc()` は Raw Input メッセージをミドルウエアへフォワードする関数として用いる。かわりにマルチ入力用 `WndProc()` となるコールバック関数を用意し、入力の処理はそちらで行う。

それ以外の点についてはシングル入力の場合とほぼ同一である。

6 実験

2005 年 11 月 20 日に豊橋市で開催された災害対応訓練で、避難所インタフェースと災害対応本部インタフェースの 2 つについて実験を行った。その結果、それぞれ SDG として多人数に同時に活用されているところを観察できた。ここでは避難所インタフェースについて、簡単に紹介する。

6.1 実験の方法

災害対応訓練に参加した住民を被験者とし、4.1 節で紹介した避難所インタフェースを用いて避難情報の入力を行ってもらった。被験者となった住民は約 30 人、年齢はまちまちである。

実験装置はノートパソコン 1 台、表示用プロジェクタ 1 台、USB マウス 6 台である。180cm 幅の長机をスクリーン手前に設置し、その上にマウス 6 台を並べた。

スクリーンには豊橋市飽海町の地図を投影、その町域を 6 つに分割し、それぞれの区域にマウスをひとつずつ割り当てた (図 1)。町域にはそれぞれ異なる色をつけ、マウス本体の色と対応させた。それぞれのマウスカーソルは、他のマウスカーソルとの干渉を避けるため、割当地域から出ないように設定した。割当地域は凹多角形を含む多角形である。

避難してきた住民はまず自分の住んでいる町域の色のマウスを持ち、それを使って自分の住んでいる建物を探す。マウスカーソルが建物を指し示すと、その建物に表札情報がある場合はポップアップ表示を行うので、地図が読めない人でもポップアップを案内として自宅を簡単に見つけることができた。自宅が見つかったら、カーソルをその上に置いてマウスボタンをクリックすると、次のような簡単な質問ダイアログが現れる (今回は非常に簡単なものに制限した)。

- 避難者は何人であるか
- 避難すべき人が全員避難所に揃ったか
- 建物の状態はどうであったか

避難者はそれぞれに多岐選択式で答える。3 つのダイアログ全てに回答てすると入力終了である。それぞれのダイアログ画面は、銀行の ATM のように、画面上のボタンをマウスを使いクリックすることで次へと進むタイプである。

なお、入力方法についての説明は画面上のヘルプと係員の補助のみとし、事前の説明などは行わなかった。

6.2 被験者の反応

被験者の年齢によりシステム操作には大きな違いが見られた。30 代くらいまでの若い世代の被験者はマウス入力に戸惑うことなくスムーズに入力を行った。特に 10 代以下の子供は非常

にスピーディに入力を行っていた。一方、年配の被験者はマウス入力そのものがうまくできず、なかなか入力できなかった。特に反応が敏感でカーソルの動きの速いマウスでは入力がしづらいようであった。マウスのボタンがどこなのかわからないというコメントもあった。

被験者は概ね複数人で相談しながら入力作業を行っていた。自分の家はどれであるのかを近所の住民とお互い確認しながら入力していた。このため自宅の建物の把握はスムーズであったが、一方で、1つのマウスに対して複数人のグループで操作するために机のスペースを大きく占拠し、並行入力数の増加の妨げとなっている場面もあった。

実験を通じて、被験者が自分のマウスカーソル以外のマウスカーソルについて気にする様子は見られなかった。複数の被験者がそれぞれ別のマウスを持ち、別の区域にある建物について並行して入力を行う場面も多く見られ、多人数対応システムとして活用されている様子が観察できた。

7 これからの課題

これまでのソフトウエア開発および豊橋市での実験で、いくつかの課題が明らかになった。

7.1 カーソル描画の問題

5.2.2 節で解説した通り、マウスカーソルはミドルウエアで描画している。この描画作業はGDIを用いて行っているが、GDIはアプリケーションプログラムが通常の画面描画を行うのと同一のレイヤーで動作するものである。したがって、アプリケーションプログラムより見かけ上は手前に位置するマウスカーソル描画とアプリケーションプログラムの描画が干渉することがある。その結果マウスカーソルが上書きされ消えてしまったり、画面にゴミが残るといった、不自然な画面表示が発生することがある。

カーソル描画とアプリケーションプログラムの描画の調停を完璧に行えばこの問題は起こらないが、この調停が簡単なものではないことがわかった。特に汎用ミドルウエアという視点で見た場合、プログラムコードの管理が煩雑になり開発者に負担がかかるのは好ましくない。

Windowsのカーソルは、GDIでなくグラフィクスアクセラレータのハードウエアカーソル機能を用いて描画している。この機能を汎用的に使えるのかどうかを検討したり、その他にも自由形状のウインドウを用いる等、GDIの一般の描画との分離性に優れたカーソル描画法が必要である。

7.2 キーボード入力への対応

Raw Input を用いると複数台接続したキーボードそれぞれの入力データをとることもできる。今回は人数入力をATM式で行ったが、キーボードを利用したほうがよいこともある。また、例えば避難民に名前を入力してもらう際などは、キーボードのほうがよいだろう。

ミドルウエアをキーボードにも対応できるようにしたいが、キーボード入力において問題になるのが日本語入力環境である。入力するユーザが複数であればインプットメソッドも複数必要である。WindowsのIMEを利用できるのか、そうでない場合何を利用できるのか、変換入力窓はどうするのか、等、通常想定されない入力環境を用意しないといけないため、考慮すべきことは多い。

7.3 高齢者にマウス操作は難しい

豊橋市の実験で、避難所のシステムは中年程度までの被験者には概ね好評であったのに対し、高齢の被験者にはマウス操作に不慣れであることもあいまって、あまり好い評価は得られなかった。

SDGにおいて、マウスを増殖させるのは安価に並行作業を実現するのに理想的である。しかしながら、特に若年者から高齢者までを対象としたシステムにおいては、マウスは操作が難しいデバイスであるということが、安価であるというメリットを引き下げてしまう。

特に今回の実験で実感したこととして、敏感で速すぎるマウスは高齢者には特に扱いづらいこと、無線マウスもマウス速度が安定しないた

め高齢者には扱いづらいこと、がある。また、マウスボタンがマウス本体と一体化したようなデザインのものでは、マウスボタンがどこにあるのかわからないという問題も発生した[2]。

マウスの速度はミドルウエアで調整できるようにしてあるので、実験前に念入りに調整すべきであった。マウスボタンについては、目立つように丸いシールを貼るなどの配慮が必要であると思われる。

8 関連研究

SDG のミドルウエアやアプリケーションには以下のようなものがある。

SDG Toolkit[3] は、我々のミドルウエアと同様の SDG ミドルウエアである。SDG Toolkit は C#等 .NET SDK 用のコンポーネントであり、フォームに貼り付けた領域の中でマルチ入力を利用するというスタイルをとっている。Raw Input を利用する点は本ミドルウエアと同様である。SDG Toolkit はキーボードにも対応するが、5.2.3 節で述べたようなマウスカーソルの詳細な動作管理は実装されていない。また、5.2.5 節で述べたコントロールの問題へは、代替パーツを用意することで対応しているなど、複数の点でアプローチが異なっている。

KidPad[4] は、SDG アプリケーションの例である。KidPad は SDG を子供用のストーリーボードへ応用したものであり、子供たちは SDG を効果的に利用したという結果が出ている。

その他 今回の避難所のインタフェースでは、ユーザ同士の干渉を避けるために居住地域ごとにマウスカーソルを割り当て、その動作を割当地域から出られないよう制限する手法をとったが、ユーザ同士の干渉防止措置をとらずともユーザは自ら干渉を避けるという研究結果もある [5]。

[2]ホイールがボタンに見えるようであった。ホイールクリックもボタンクリックと同じ処理にしてあったのだが…

9 まとめ

災害発生時の情報共有支援を目指し、白地図の電子化を基本とした情報共有支援システムを開発している。システムは多人数が同時に作業できるよう SDG のスタイルを採用している。SDG は小規模なシステムで安価に多人数対応システムが構築できる。

SDG の実現には Windows の Raw Input API を用い、Raw Input 周りの処理を行う部分は汎用 SDG ミドルウエアとして開発を行った。ミドルウエアにはイベント処理や子ウインドウの管理などを盛り込み、シングル入力のアプリケーションを開発するのとほぼ同様に SDG アプリケーションを開発できるようにした。

豊橋市の災害対応訓練で我々の情報共有支援システムを運用し、SDG として多人数が同時に活用できることを確認した。同時に今後課題とすべき点も明らかになった。これらを解決しながら、さらに面白いアプリケーションを開発し、ミドルウエアの完成度を高めていく。

参考文献

[1] 静岡県災害図上訓練 DIG: `http://www.e-quakes.pref.shizuoka.jp/dig/`, 静岡県地震防災センター

[2] Stewart, J., Bederson, B., Druin, A.: Single Display Groupware: A Model for Co-present Collaboration, *Proc ACM SIGCHI*, 1999

[3] Tse, E., Greenberg, S.: Rapidly Prototyping Single Display Groupware through the SDG Toolkit, *Proc Australasian user interface conference*, 2004

[4] KidPad: `http://www.kidpad.org/`, University of Maryland

[5] Tse, E., Histon, J., Scott, S., Greenberg, S.: Avoiding Interference: How People Use Spatial Separation and Partitioning in SDG Workspaces, *Proc ACM CSCW*, 2004