覗き窓、並べたら大画面
– どこでも簡単に使える大画面システム –

上田 真史
masa-u@nue.ci.i.u-tokyo.ac.jp
東京大学大学院情報理工学系研究科

— 概要 —

大画面を扱うにはハードウエアを増設するなど費用や手間の面で難点が多く、また大画面システムの機動性は低かった。これらの問題を解決するために、ハードウエアを使わずソフトウエアのみで、ネットワークを通じて接続された PC のディスプレイを統合するシステム「そら」を開発したので、これを紹介する。

「そら」は統合ディスプレイのモデルとして「覗き窓方式」をベースにしている。仮想画面を覗く窓を並べることで大画面を実現するスタイルで、この方式の採用により、運用を柔軟に、かつ簡単に設定できるようになった。ノート PC でも運用できるので、大画面システムの機動性の問題も解決できる。「そら」には更にマルチマウス対応機構を組み込み、マルチマウス対応ワークスペースとしても機能する。

1 はじめに

計算機のディスプレイ領域はときにデスクトップとも呼ばれる。本物の机で作業するとき広い机のほうが作業効率がよいのと同様に、計算機のデスクトップも広いほうが作業効率がよい、という場合がある。広い机でないとできない作業があるように、計算機においても広いデスクトップがないとできない作業がある。

近頃は大型のディスプレイも手に入るようになり、ディスプレイ領域の拡大は、費用を考えなければ比較的簡単にできるようになってきた。とはいえこの費用はまだ十万円単位であり、かつ専門的な知識が必要な部分が未だ残っていることもあり、誰でも気軽に、というわけにはいかない。

一方、計算機が複数存在する環境は近頃では珍しくない。また、そういう環境では計算機同士がネットワーク接続されていることが多い。ここに着目し、ディスプレイを多数統合することで大きな領域を実現するシステムを開発した。

2 大画面の難点

計算機のディスプレイ領域については、特に地図を扱ったり、多くの情報をいちどに閲覧する監視業務などの分野において、大領域のものが求められている。広いディスプレイ領域を実現するには、単純に大きなディスプレイを使用するか、もしくはディスプレイを複数台並べて擬似的に大画面として使用する。しかし現在、一般的に手に入るディスプレイは WUXGA (1920 × 1200) 程度のものであり、一般的な PC ではこれを 2 台程度までしか接続できない。

WUXGA より大きなディスプレイを利用しようと思ったり、3 台以上のディスプレイを PC に接続しようと思うと、とたんに技術面および費用面でハードルが高くなるのが現在の状況である。

2.1 費用

WUXGA のディスプレイは、2007 年 11 月現在 5 万円程度で入手できる。これより大きなディスプレイとしては WQXGA (2560 × 1600) のものなどが市販されているが、価格が 14 万円ほどに跳ね上がる。さらに、WUXGA より大きなディスプレイを駆動するにはそれに対応したディスプレイカードが必要であり、1 万円以上の追加費用が必要である。

1 台の PC は概ね 1 台もしくは 2 台のディスプレイを接続できるようになっているが、3 台以上のディスプレイを接続しようとすると、やはりディスプレイカードの増設が必要である。

Inspection windows make up a large display – An easy-to-use multiple display integration system
Masafumi Ueda, Graduate School of Information Science and Technology, The University of Tokyo

1 枚のディスプレイカードには 2 台のディスプレイを接続できるようになっていることがほとんどであり、PC の拡張スロットは多くとも 5 本程度である。特殊なディスプレイカードでは 8 台接続可能というものが存在するが、特殊ゆえに価格は 10 万円以上である。

2.2 技術

WUXGA より大きなディスプレイに対応したディスプレイカードを PC に増設する作業は、PC のケースを開けて適切なスロットに適切なカードを挿入する、というものである。ディスプレイカードの接続規格にはたくさんの種類がある。その PC にとって適切なものを選び、さらに PC 内部に適切に装着するのは、素人には容易ではない[1]。専門家に増設を依頼すれば、技術料としてさらなる費用もかかる。

より根本的な問題として、グラフィックカードがそもそも増設できない PC も存在する。良い例がノート PC である。ノート PC は小型化のために拡張機器を接続する端子を減らしている。グラフィックカードを装着できるようなスロットは備わっていないので、誰がどうがんばってもディスプレイの追加接続はできない。

2.3 機動性

大きなディスプレイは重量も大きく、容易には移動できない。ディスプレイが何台もつながる PC は、PC を移動できなければそのハードウエアを組み換えねばならず、いずれにしろどこにでも持っていくことはできない。そしてノート PC にはディスプレイカードを増設できないのである。

いざ大画面が欲しい、一時的に大画面が欲しい、といった機動性の要求されるとき、ハードウエアで対応するのは難しい。

3 方針

PC にたくさんのディスプレイを接続したり大きなディスプレイを接続するのは難しいが、PC に 1 台だけディスプレイを接続するのはとても簡単である。ディスプレイのつながらない PC というのはまず存在しない。近頃は PC がたくさん存在する環境はありふれている。例えばオフィスなどがそうであるが、そこには当り前のようにたくさんの PC とたくさんのディスプレイが 1 対 1 で接続されている。

もうひとつ接続するのが簡単であるのはネットワークである。DHCP などの普及によって、PC のうしろにある Ethernet ポートにケーブルを接続するだけでよい環境ができあがっている。前述の「PC がたくさん存在する」環境においても、全ての PC がネットワークで相互に接続されていることは珍しくない。

以上を組み合わせると、ネットワークを経由して、たくさんの PC につながっているたくさんのディスプレイにアクセスできるということになる。そこで、ネットワーク経由でディスプレイにアクセスするソフトウエアさえあれば、特にハードウエアを買い足す必要もなく、1 台の PC から複数のディスプレイを制御できるだろう。さらに、それらのディスプレイを 1 箇所に集めて並べてしまえば、大型の統合ディスプレイを実現できる。ハードウエアの制約がないので、ノート PC でもシステムを構築でき、大画面システムの機動性の向上も期待できる。

4 覗き窓方式

大型ディスプレイを実現するためにネットワーク上の複数のディスプレイを統合して使用するという方法は既に実装されている。実装の例は 8 節で解説するが、いずれも、もともと存在するディスプレイ領域を連結して使おうというものである。

今回私は統合ディスプレイ環境を実現するのに「覗き窓方式」を考案、採用した。この方式では、統合されるディスプレイを「領域」としてではなく「覗き窓」として扱う。大型ディスプレイが必要な計算機には、必要なだけの大きさの仮想画面領域を用意し、各々の覗き窓がその領域の一部を覗く、という方式である。覗いている部分の少しずつ異なる覗き窓を並べると、その結果大型画面ができるしくみである。

[1] 最近になって USB 接続のグラフィックアダプタが出現し、このハードルは若干下げられている

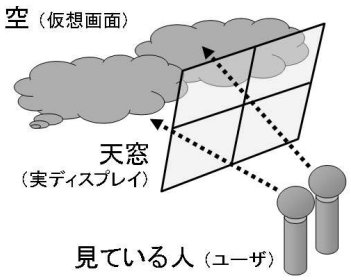

図 1: そらモデル

各々のディスプレイを天窓、仮想画面領域を空、と考えるとわかりやすいだろう。空に描画を行い、天窓を通じてそれを見るのである。図 1 にこのモデルを示す。天窓が大きければ空の広い部分が見えるし、天窓が小さくともそれがたくさん並んでいれば見える空は広い。

このモデルから、開発したシステムを仮にそらと呼んでいる。

4.1 覗き窓方式によるメリット

そらには、覗き窓方式を採用したことによる以下のような特長がある。

仮想画面側での表示設定が不要 仮想画面はあくまで仮想空間を提供するものであり、表示がどう行われるかとは切り離されている。どのディスプレイの隣にどのディスプレイがあるか、といった情報は仮想画面を用意する段階では不要であり、設定にかかる負担を取り除ける。

ダイナミックな再配置が可能 実際にディスプレイを並べてみて、ちょっと横の解像度が足りない、ちょっと縦に長すぎる、というようなときも、そらではそのつどディスプレイを配置しなおすだけでよい。ディスプレイやそれを駆動している計算機の配置が変わっても、覗き窓の位置が変わるだけなので仮想画面に影響はなく、システム全体を再起動する必要がない。仮想画面上で既にアプリケーションソフトが動いていても、それが動いているまま再配置が可能である。

表示内容の共有が可能 空はどこからでも見えるものである。たとえば隣り合った部屋に同じ方角の窓があったとしたら、どちらの部屋からでも同じ空模様が見えるだろう。さらには、いくらか離れて建っている家においても、同じ方角の窓からは概ね同じ空模様が見える。

そらについても同様のことが可能である。つまり、大画面を多数のディスプレイで実現するのと同時に、別のディスプレイ群を使って同じ画面を表示できる。単純に 2 つの覗き窓を同じ場所を見るように設定すればよい。もちろん、共有画面の接続および切り離しも、仮想画面そのものに影響を及ぼすことなく行える。インターネットを介して遠隔地で大画面を共有することも可能である。

重複した表示が可能 一般の紙地図は大抵、隣の図面との境界部分は隣の図面と少し重なるようにつくられている。そらの覗き窓方式は、この重なりも再現できる。すべてのディスプレイには枠がついているので、ディスプレイを並べるとディスプレイの境目にどうしてもすき間が空いてしまう。通常のマルチディスプレイでは、ディスプレイの境目で内容に関係なく表示がちぎられて見にくく感じてしまうことがあるが、そらでは地図を並べたような表示が可能である。

逆に、ちょうどディスプレイの枠の太さだけ表示をとばし、ディスプレイの枠があたかも本物の窓枠であるかのように自然な表示を行うこともできる。

5 システム詳解

そらは Windows 上で動くシステムであり、図 2 のようなシステム構成になっている。角丸四角形で示したプログラムが今回開発したプログラムである。

仮想ディスプレイドライバは仮想画面を用意するためのデバイスドライバ、仮想ディスプレイサーバは仮想画面の内容を仮想ディスプレイビューワに送信するためのプログラム、仮想ディスプレイビューワは「覗き窓」プログラムである。

この節では、それぞれのプログラムについて解説する。

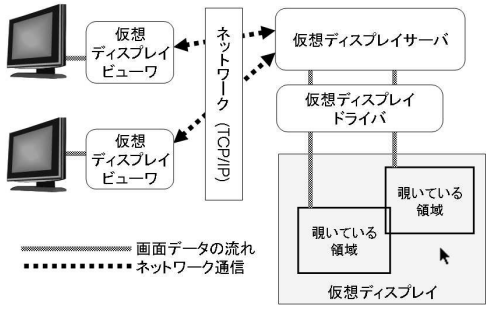

図 2: システム全体図

5.1 仮想ディスプレイドライバ

仮想ディスプレイドライバは、Windows 用のデバイスドライバである。通常のディスプレイドライバとは異なり、メインメモリ上に描画面を確保し、そこに描かれた内容は後述の仮想ディスプレイサーバから読み取れるようにする。

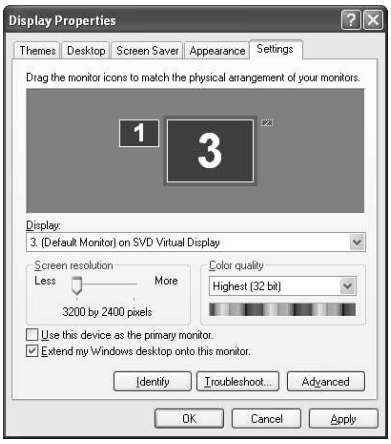

図 3: 仮想画面のプロパティ

仮想ディスプレイドライバをインストールすると、Windows の画面の設定では新しいディスプレイがひとつ追加されたように認識される。図 3 の 3 番が仮想画面である。この仮想画面は Windows からは通常のディスプレイと全く同じように扱われる。既存の Windows アプリケーションプログラムはほぼ全て[2]が仮想画面上で問題なく動作する。このことは、そら用にあらためてプログラムを対応させる必要はないということである。さらに、タスクバーを仮想画面に移動したり、Windows の起動時に仮想画面をメインディスプレイとして使うように設定したり、といったことも可能である。

仮想画面のサイズはメインメモリの許す限りどのような大きさにも設定できる。手許にあるディスプレイの大きさと数とその並べかたにあわせ、自由に様々な統合ディスプレイをつくれるようにするためである。なお仮想画面は 1 ピクセルあたり 4 バイトのメモリを消費する[3]。

5.2 仮想ディスプレイサーバ

仮想ディスプレイサーバは Windows 用のアプリケーションプログラムである。仮想画面の持つ描画面メモリの内容を読み出し、IP を通じてそれを仮想ディスプレイビューワに届ける。

仮想ディスプレイドライバが Windows から受け取った描画命令はメモリ上の仮想画面で実行されるとともに、いったん記録される。仮想ディスプレイサーバはこの記録を読み取り、効率のよいように構成しなおした後、仮想ディスプレイビューワに送信する。なお、仮想画面のうちどの領域を転送するかは、仮想ディスプレイビューワが指定するものとし、また動作中にいつでも変更できることとした。

描画記録を構成しなおすのは、仮想ディスプレイサーバと仮想ディスプレイビューワの間の通信回線が、ディスプレイのデータにとって充分高速でない場合が多いからである。ディスプレイのデータは 1 枚あたり MB 単位の容量になるので、頻繁に書き換えが行われた場合に全て転送するのは無理である。重複して描画された部分を取り除く、データを圧縮する、ビューワが見ていない部分を無視する等の処理を行っている。

[2]DirectX などによるハードウエアアクセラレーションを必須とするアプリケーションプログラムは動作できない。

[3]UXGA サイズでは、合計で約 7.3MB のメモリが必要である。カーネル空間のメモリを使用するため、32 ビット環境では最大数百 MB のメモリしか使用できない。この場合利用可能な解像度は最大で UXGA ディスプレイ 30 枚分程度である。

5.3 仮想ディスプレイビューワ

仮想ディスプレイビューワは Windows 用のアプリケーションプログラムである。仮想ディスプレイビューワは、仮想ディスプレイサーバから転送された描画データを実際のディスプレイに映すために、再描画を行う。

注意すべきであるのは、仮想ディスプレイビューワは**実際の**ディスプレイ領域で動作させるソフトウエアであるということである[4]。仮想画面に描かれたデータは仮想ディスプレイドライバから仮想ディスプレイサーバに転送され、さらに TCP/IP を通じ仮想ディスプレイビューワに転送、最終的に仮想ディスプレイビューワが通常のディスプレイ領域にて再描画することで、仮想画面に描画されたデータが実際のディスプレイに現れる（図 4）。

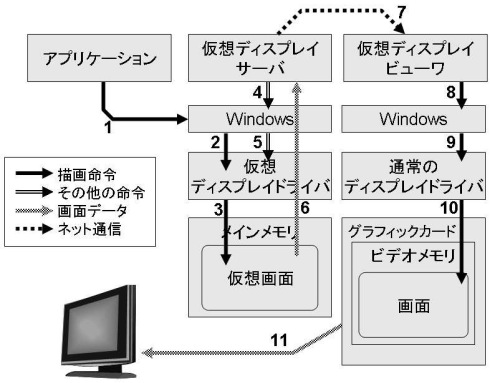

図 4: 描画命令が実際のディスプレイに反映されるまで

仮想ディスプレイビューワは前述のとおり「覗き窓」である。この窓が仮想ディスプレイ上のどの領域を再描画するかは、起動中にユーザが自由に設定できる。領域の指定はビューワごとに行われ、それぞれ制限なく指定できる。

ビューワの配置を現実のディスプレイの配置と同じにする必要はなく、また、同じ領域を複数のビューワが再描画したり、全く再描画されない領域があったり、1 台のディスプレイにいくつのビューワを表示してもよい。

[4]もちろん仮想画面上でも動作するが、合わせ鏡のようなおかしな表示を引き起こしてしまうだけである。

仮想ディスプレイビューワは統合画面を構築する上で重要なソフトウエアである。統合画面を簡単に、かつフレキシブルに構築できるよう、全画面表示や仮想画面自動探索など、様々な機能を実装した。

特に自動探索機能は重要である。この機能は同一ネットワークセグメント上にある仮想画面を自動的に探すものである。ユーザはこの機能により、仮想画面を持つ PC のアドレスを入力することなく仮想画面に接続できる。統合ディスプレイの構築をマウスクリックだけでできるようにする機能である。

6 マルチマウス機構

1 つのワークスペース上で複数の入力デバイスを同時に扱い、複数のユーザが同時に作業できるシステムを Single Display Groupware [1] という。このうちマウスを複数扱うものをマルチマウスシステムと呼んでいる。

マルチマウス環境では自ずとユーザ数が増えるので、ディスプレイ領域が足りなくなりやすい。単純にコストをかけてディスプレイ領域を拡大してもよいが、マルチマウスシステムはもともとマウスを複数接続するだけで構築できるものである。ディスプレイ領域拡大のためにコストをかけてしまうと、マルチマウスシステムの安価であるという特色が失われてしまう。**そら**の開発ではこのジレンマを解決することも目標とした。

私はこれまで、Windows 上の単一のプログラムでマルチマウス環境を実現する **MMTk** を開発し、現在も公開している [2]。昨年はネットワークを通じこれを拡大するためのシステム **νMMTk**[3] を開発していたが、**そら**にはこの主要機能を取り込んだ。

6.1 マルチマウス API

そらのマルチマウス API は MMTk のものから一新した。MMTk では、あるプログラムがマルチマウス環境を開始すると、そのプログラムが

全てのマウス情報を奪ってしまう[5]。大型のディスプレイ上で 1 つのプログラムしか動作できないのは、ディスプレイの使いかたとしては非常にもったいない。

マウス操作に関する情報には、マウスカーソルの位置やクリックなどがある。この情報をシステムからアプリケーションプログラムに届けるのには、Window Message を利用することとした。Window Message の受信機構は、Windows 用でかつウインドウを出すプログラムであれば必ず備わっている。コネクションレスであり複数のプログラムに対しても並行して送信でき、受信するプログラムは対応しない Window Message を無視してよいことになっている。マルチマウスシステムからアプリケーションプログラムへ無差別にメッセージを送信しても、非対応のプログラムはそれを無視するだけなので問題は起こらない。

Window Message では通常のマウスのメッセージもやりとりされている。マルチマウスシステムからアプリケーションプログラムへ送信されるメッセージは、通常のマウスメッセージと互換性を持つように設計し、既存のソフトウエアをマルチマウスに対応させる際に開発コストを下げられるようにした。

6.2 マルチマウス処理の組み込み

マルチマウス対応処理は統合ディスプレイの処理とは関係がない。そらとは別ソフトウエアとしてもよいが、マルチマウス処理において通信する経路はそらのディスプレイ情報の通信経路と似通っている。システム構築の手間を省くため、マルチマウス処理はそらに組み込んだ。

マウスの情報を読み仮想画に届ける処理は仮想ディスプレイビューワが行い、その情報を仮想画面上のアプリケーションプログラムに届ける処理は仮想ディスプレイサーバが行うようにした。

図 5 に、そらのマルチマウス機構の略図を示す。

[5]全てのマウス情報を奪ってしまうので、マルチマウス非対応のプログラムも同時には動作できない。

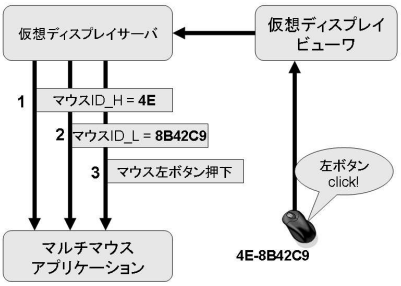

図 5: マルチマウスメッセージの例

7 運用例と「そら」の特長

そらは様々な運用が可能である。単純に大型統合ディスプレイシステムとして、画面共有システムとして、マルチマウス対応ワークスペースとして、いずれの運用も、以上を組み合わせた運用も可能である。

図 6: ノート PC を 4 台使用している例

図 6 は、ノート PC を 4 台使用して統合ディスプレイを運用している例である。左側の 2 台の PC は画面を 180 度回転させて右側の PC の表示と連続するようにしてある。解像度は 2048 × 1536 である。そら はハードウエアの増設が不要であるので、このようにノート PC でも利用できる。大画面をポータブルにできるので、会議室等で臨時に大画面を利用することもできる。

図 9 は、プロジェクタ 6 台を横に並べ、横幅 10m の超大型ディスプレイを運用している例である。解像度は 6144 × 768 である。このような運用は、美術展示などにも応用できるだろう。

図 7 は、ディスプレイ端のデータを隣のディ

図 7: ディスプレイ端を重複表示している例

スプレイに重複して表示している例である[6]。この例のように地図を表示する場合などに、ディスプレイ端を重複させて見やすくするといった表示方法が可能である。

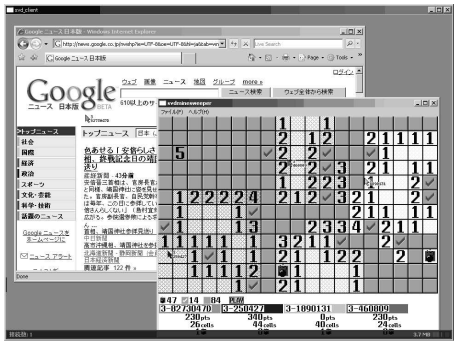

図 8: マルチマウスアプリケーションと通常のアプリケーションが共存している例

図 8 は、マルチマウス対応マインスイーパー (右下) とマルチマウス非対応の Internet Explorer (IE、左上) が仮想画面上に共存している例である。マインスイーパーは簡単な実装で動作しており、IE については一切の変更を加えずに動作している。

IE は仮想画面のある PC のマウスでしか操作できないが、マインスイーパーは仮想ディスプレイビューワが動作している PC 全てのマウスで同時に操作できる。そらには画面共有の機能があるので、遠隔でこの画面を共有すると、マインスイパーは何も変更を加えずとも「ネットワーク対応ゲーム」に変身する。マルチマウスにさえ対応すれば自動的にネットワーク対応ソフトになるのもそらの特長である。

[6]写真は少々加工して、上下のディスプレイの距離を縮めてある。

8 他のシステムとの比較

8.1 マルチディスプレイ環境

MaxiVista[4] はそらに最も近いシステムである。Windows PC に仮想ディスプレイドライバを用意し、サーバとクライアントによって他の PC に接続されたディスプレイをセカンダリディスプレイとして使用する。MaxiVista はネットワーク越しに最大 3 台までのディスプレイを接続可能である。クライアントはあくまでセカンダリディスプレイというデザインであり、制御は仮想画面のある PC から、ディスプレイ単位で行う。

Distributed Multihead X[5] は X Window System 用のディスプレイ統合システムである。ネットワークを通じ、X の display を連結する仕組みであり、連結の方法は起動時に指定する必要がある。

GLIA[6] はネットワークを通じてマウスとキーボードの情報を転送し、複数の PC と複数のディスプレイでマルチマウス対応マルチディスプレイ環境を実現するものである。マルチマウスに対応したシステムであるが、各々のディスプレイはあくまでそれが接続されている PC のディスプレイであり、グラフィック的には統合されていない。ディスプレイ全てを使用するプログラムを動かす場合、ディスプレイの数だけプログラムのインスタンスを起動し、プログラムが自律的に通信して協調する必要がある[7]。

8.2 マルチディスプレイ以外

Synergy[7] はネットワークを通じてマウスとキーボードの情報を転送し、複数の PC のデスクトップ領域が連続しているようにみせかけるソフトウエアである。Synergy はディスプレイ統合ソフトウエアではないため、「左のディスプレ

[7]これは実は昨年私が開発していた νMMTk とほぼ同じデザインである。

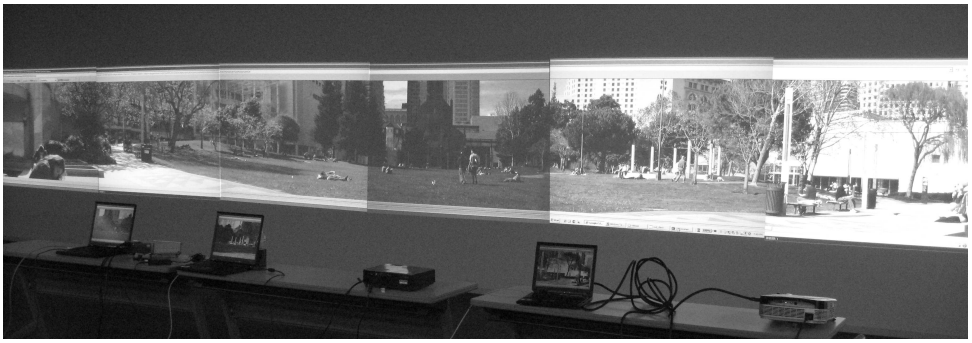

図 9: 横幅 10m のディスプレイを構築した例

イから右のディスプレイへウインドウをドラッグする」といったことはできない。

VNC[8] はネットワークを通じてキーボードとディスプレイとマウス (KVM) の情報を転送し、計算機をリモートコントロールするソフトウエアである。通信する情報の内容はそらと同様であるが、全く異なる応用である。

9 今後の予定

そらはまだ最低限の実装しかなされていない。今後一般に公開する予定だが、そのためにはインストーラの用意や仮想ディスプレイビューワの強化、通信の暗号化など、主にユーザインタフェースとセキュリティの部分で改良が必要である。これらの改良を行った後、ユーザマニュアルやマルチマウス対応開発キット等とともに、なるべく早い段階で公開する予定である。

10 まとめ

大画面を扱うにはハードウエアを増設するなど費用や手間の面で難点が多く、また大画面の機動性は低かった。これらの問題を解決するために、ハードウエアを使わずソフトウエアのみで、ネットワークを通じて接続された PC のディスプレイを統合するシステム「そら」を開発した。

「そら」は統合ディスプレイのモデルとして「覗き窓方式」をベースにしており、設定を柔軟に、かつ簡単にできる。ノート PC でも運用できるので、大画面システムの機動性の問題も解決できる。「そら」には更にマルチマウス対応機構を組み込み、柔軟な大画面システムのマルチマウス対応ワークスペースとしての利用を可能とした。

この開発は、独立行政法人情報処理推進機構の 2006 年度下期未踏ユースにおいて、早稲田大学理工学部 筧捷彦 PM の採択により実施したものです。筧 PM にはこの場を借りて感謝いたします。

参考文献

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

[2] 上田 真史: 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, 電気通信大学大学院電気通信学研究科 修士論文, 2006

[3] 上田 真史, 竹内 郁雄: ネットサーフィンするマウスカーソル – マルチ計算機マルチマウスシステムの開発, 第 48 回プログラミング・シンポジウム, 2007

[4] MaxiVista: `http://www.maxivista.com/`

[5] Distributed Multihead X: `http://dmx.sourceforge.net/`

[6] 西村 真一, 由井薗 隆也, 宗森 純: 複数のネットマウスにより大きな共同作業空間構築を支援するミドルウエア GLIA, 情報処理学会論文誌 48(7) pp2278-2290, 2007

[7] Synergy: `http://synergy2.sourceforge.net/`

[8] RealVNC: `http://www.realvnc.com/`

夏のプログラミング・シンポジウム 2007 報告

2007 年夏のプログラミング・シンポジウムは「First Programming Languages プログラミング言語の実力と美学」というテーマで, 8 月 8 日 (水)〜10 日 (金) に信州上山田温泉ホテル清風園で開催した. (2005 年 9 月の夏のシンポジウムも同じ清風園であった.)

まずその趣意書には

> 近頃「情報科学離れ」をよく耳にします. その前に「プログラミング離れ」があるのではないでしょうか. 計算機の恩恵を受けるには, プログラムを書かなければなりません. そういうプログラミングの前に, プログラムを書くこと自体が楽しい作業であるということが, 悲しいことに忘れられつつあります. プログラムを書かずに使うパソコンは, 宝の持ち腐れです.
>
> プログラミング言語の提供する機能を充分に活用し, 解決したい問題を美しく書き上げる知的活動の存在をもっと知って欲しい, もっと広めたい. そのような願いを込め, シンポジウムでは初心者でも楽に使え, 実用的な問題も直ぐに解決できるプログラミングにつき, 言語とプログラムをめぐって討論したいと考えました.

と書いた. 今回は全員発表とした. 約 20 件の通常発表と, 夜のプログラミング言語応援演説である. 応援演説の方は出来ればその言語でカレンダーのプログラムを書くことにした.

通常発表の話題には以下のような例を載せた.

・Computer Programming は本当に Art か.
・分かりやすく，能率よく，見た目も美しいプログラムを書く訓練法.
・初心者に，プログラミングの本質を一目で分からせるにはどうするか.
・Dijkstra のプログラムのエレガンス，精神を伝えるには どうすべきか.
・「Bach First Lessons 16 Short Pieces for Piano」のプログラミング版を作る.
・ACM ICPC のように，短時間で答だけ合えばよいというコンテストの是非.
・大学の入学試験で今でも BASIC 風の言語しか使われないのは問題だ.
・プログラミングの "Art & Crafts" 対 "Scientific Discipline".
・効率改善の経験談（高速化，短小化）.
・人生を変えたプログラミング言語.
・特定のプログラミング言語に惚れる理由.
・自分にとって一番大事なプログラミング言語.
・一番素敵に見せられる言語，一番見てくれがきれいな言語.
・今後はどういうプログラミング言語が作られるか，作りたいか.
・インスタントプログラミング.
・ゴルファー: いかに短い文字数でプログラムを書くか.
・あっと驚くアルゴリズム（A'stonishing Programming）.

こうして通常発表 20 件, 応援演説 21 件が集まった. 発表も応援演説もしないのが 1 人で参加者は 42 名であった. 特筆すべきは高校生が 1 人参加したこと, 数日前に都内であった Light Language Spirit の参加者が何人も来たことであった.

通常発表についてはこの報告集に論文があるので, そちらを見て欲しい. 萩谷君の Excel でカレンダーを書くのは面白かった. でももっとまともなプログラミング言語があるのになぜ?と思った. 角田君は最近の学生がプログラミングが苦手な理由を検討している. 三廻部君は ICPC のような短時間プログラミングでも, 読みやすくいいプログラムが書けると主張する (本当か).

応援演説でもカレンダーを書いて貰ったが, ruby 陣営は最初から require 'date' と書いてしまうので, 閏年の計算アルゴリズムを工夫せず, 他人任せにしてしまった.

obfuscated programming language という恐ろしいものも知った. Whitespace と Unlambda である. BrainF*ck は難解プログラミング言語で, これらではついにカレンダーは出来なかったようだ.

Logo でカレンダーを書くのも大変そうであった. 数字の出力はディジタル時計のような 7 セグメント LED の形を亀で書こうというのであった. きっと地上絵のようなカレンダーが出来るのだろう.

竹内君のカレンダーは資料はなかったが, 火星用などもある規模雄大な宇宙カレンダー構想であった. これについては「明日に向かってプログラめ!!」に紹介がある.

http://rikunabi-next.yahoo.co.jp/tech/docs/ct_s03600.jsp?p=001157

カレンダー用の IC を作ったという報告もあった.

久しぶりに夏の幹事を引き受けた. 昔は 2 人くらいでやっていたのに, いまは幹事が大勢いて, 当日以外は結構楽な仕事になっている. 自己推薦で幹事になった伊知地君がご存じのとおり世話焼きなので, 彼からのメイルの発信件数も断トツで, 伊知地君に追いまくられて準備が進んだという感じであった.

いろいろな業務を分担した幹事の皆さん, ご苦労様でした.

2007 年 夏のプログラミング・シンポジウム幹事団
幹事長　和田 英一
伊知地 宏
齊藤 正伸
笹田 耕一
繁冨 利恵
山下 伸夫