社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－DPS－124（11）
2005－ GN － 57（11）
2005／9／22

複数機器の入出力協調ミドルウェアGLIAの開発

西村真一†　　河崎雄太††　　由井薗隆也††

ネットワーク上に存在する多入出力を扱うシステムの共通基盤としてミドルウェアGLIAを開発した．GLIAはマウスを入力装置，ウィンドウ画面を出力装置とし，複数マウスによる1ウィンドウ画面の操作，1マウスによる複数画面の操作，これらを組み合わせた複数マウスによる複数ウィンドウの操作を実現している．GLIAの操作性を評価した結果，多画面間でGUI部品を移動する作業や，多画面を用いた協調作業で操作性が向上することが分かった．またGLIAによる既存アプリケーションの多入力化は，主要な変更をイベント処理部分内で押さえることができた．

Development of Middleware GLIA for Cooperative-Multiple Input/Output

Shinichi Nishimura†　Yuuta Kawasaki††　Takaya Yuizono††

GLIA has been developed as middleware, which supports development of application that treats multiple input/output devices on network. Now, GLIA handles a mouse as a input and a display as an output. Some application have been implimented with GLIA to manipulate multiple displays with multiple mice. We found the followings from the results of experiments. (1) The efficiency to move GUI components between two displays and to work cooperatively using multiple displays was improved. (2) Modifying existing application to handle multiple input/output, the changes of program ware mainly in the method of event handling.

1. はじめに

近年，携帯電話，情報家電，特殊な小型センサなど，ネットワークに接続可能な入出力機器が増加している．それにともない，これらの機器を組み合わせて動作するシステムが多数登場している．

このようなシステムの開発には，少なくとも入出力装置へ直接アクセスする環境と，入出力装置同士を結合するネットワーク環境を開発する手間が生じる．

†島根大学大学院総合理工学研究科
†Graduate School of Science and Engineering , Shimane University
††島根大学総合理工学部
††Faculty of Science Engineering , Shimane University

本研究では多入出力を扱うシステムの共通基盤として，複数機器の入出力協調をおこなうミドルウェアGLIA (GLue Interactive Action)を開発した．

2. ミドルウェアGLIA

2.1 方針

GLIAは多入出力を扱うアプリケーションの開発を容易にするために，多入出力同士が柔軟にネットワーク上で結合する機能と，多入力が同時に入力操作ができるGUI環境を提供する．

また多機種のOS上で動作することを想定しているため，機種依存の低いJava仮想機械上で実行する．複数のマウスから入力情報を取得するのにはJinput[1]という既存のJavaAPIを使用した．Jinputをもちいるとポーリング(定期調査)方式でマウスの入力情報を取得できる．

2.2 実装方法

(1)多入力のためのGUI環境

多入力のためのGUI環境として，複数のマウスにより同時に入力操作ができるマルチカーソル環境を制作した．具体的には「マウスモニタ」と呼ばれるオブジェクトを実装し，次のような制御を行っている．まずマウスモニタがマウスの入力情報を取得し，それを元にマウスカーソルを描画する．次に対応するマウスイベントを生成し，GUI部品へ送信する(図1(a))．またマウスが複数接続されているときは，各マウスに対応するように複数のマウスモニタを生成し，独立に動作させることでマルチカーソル環境を実現する(図1(b))．

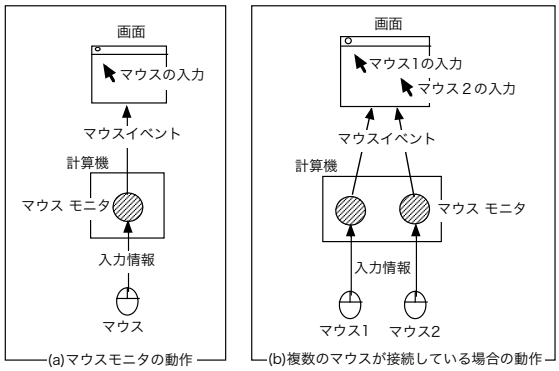

図1 複数マウスの実現方法

(2) 多入出力のネットワーク結合機能

多入出力のネットワーク結合機能はマウスモニタをネットワーク上で複製することで実現する．

図２でこれを説明する．ここでは計算機Aにマウス Aと画面Aが接続され，計算機Bに画面Bが接続されている．ここでマウスAを画面Bへ接続したいという要求がGLIAにくると，GLIAは次のような制御を行い，マウスAと画面Bを接続する．まずマウスイベントの送信を停止し， マウスモニタを計算機B上で複製する． 次にマウスAからの入力情報を，複製マウスモニタへ送信する．複製マウスモニタは，マウスの情報を元にマウスイベント生成し画面Bへ送信する．

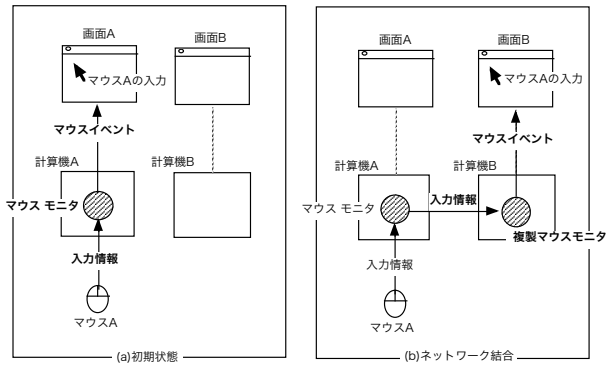

図2 入力装置と出力装置のネットワーク結合

2.3 GLIAによるアプリケーション開発方法

GLIAのGUI部品とイベントモデルは，JavaのGUI開発環境であるSwingを拡張している．よってSwingを学習したプログラマであれば， GLIAのGUIアプリケーション開発は，多入力にともなうイベント処理の記述方法を学習するだけでよい．

このことを示すために，マウス番号Nのマウスが押すと"Hello World"と表示するGUIボタンをGLIAをもちいて作成した．図3(a)にそのプログラムのイベント処理部分を示した．また合わせて図3(b)にSwingプログラムのイベント処理部分を示す．

GLIAプログラムは次のように実行される．1: マウスでボタンを押すとメソッド mouseClicked が呼ばれる．2: マウスイベントをGLIAで使われるマウスイベントへ変換する．3: マウス番号がNであれば，4: "Hello World"と表示する．

Swingプログラムは次のように実行される． 1: マウスでボタンを押すとメソッド mouseClicked が呼ばれる． 2: "Hello World"と表示する．

```
1: mouseClicked(MouseEvent e){
2:    GMouseEvent me=(GMouseEvent) e;
3:    if (me.Number == N){
4:       print("Hello World");
5:    }
6: }
```

(a) GLIA

```
1: mouseClicked(MouseEvent e){
2:    print("Hello World");
3: }
```

(b) Swing

図3 ”HelloWorld”プログラム記述例

2.3 GLIAによるアプリケーション開発例

GLIAを用いて作成したアプリケーションの実行画面を紹介する．

(1) マルチカーソル環境をテストするために，複数ユーザが同時に描画できるペイントソフトを作成した(次ページ図４)．このペイントソフトは各マウスごとに色の選択，線の太さの選択，描いた線の消去ができる機能を持つ．

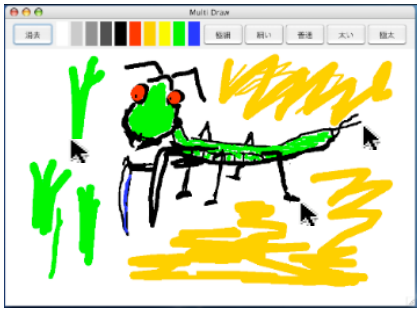

図4 GLIAプリケーションの実行画面

(2) GLIAの多入出力のネットワーク結合機能を示すために，GLIAを用いてマウスと画面を動的に結合するプログラムを作成した．このプログラムはネットワークに複数の計算機が接続され，それぞれに画面とマウスが接続されている状態で動作する．このプログラムを用いると任意のマウスで，ネットワーク上のすべての画面を操作することができる．

3. 評価実験

3.1 操作性実験

GLIAの操作性を評価するために，KUSANAGI [2] (複数画面を用いてKJ法を支援するグループウェア)をＧＬＩＡをもちいて改良した．これをKUSANAGI-Gと呼ぶ．KUSANAGIは複数のディスプレイ画面を組み合わせて，１つの大きな画面として扱うことができる(図５)．しかし１つのマウスで複数のディスプレイ画面を操作することはできない．一方，KUSANAGI-Gは，１つのマウスで複数のディスプレイ画面を操作でき，１入力多出力の操作性が改善されると期待できる．これを確かめるためにPing実験，まとめ作業実験，GLIAアプリケーション使用実験を行った．

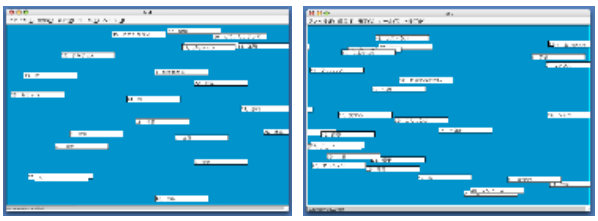

図5 KUSANAGIの動作画面

(1) Ping操作実験

Ping実験は結合した２画面を用いて１名の作業者が行う実験である(図６)．KUSANAGIを用いると２画面を操作するために２つのマウスが必要だが，KUSANAGI-Gを用いると２画面を操作するために１つのマウスだけでよい．Ping実験では，２画面を用いた作業での操作性を調査する．

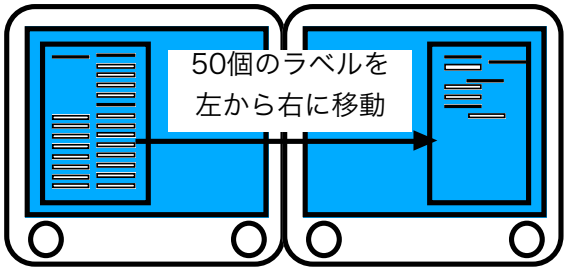

図6 Ping実験

(2) ２人によるラベルまとめ作業

まとめ作業実験では，結合した２画面を使って２名による共同作業を行う(図７)．具体的には無作為に100個の単語を選択し，ラベルとして２画面上にランダムに配置する．その状態から作業者がラベルを意味の似たグループにまとめる作業を行う．この作業ではKUSANAGIを用いた場合は１人が１画面しか操作できないが，KUSANAGI-Gを用いた場合は１人で２画面を操作できる．これによりラベルの移動しやすさと，共同作業の効率が上がるかを調査した．

また実験終了後に操作性に関する５段階評価のアンケート調査を行った．

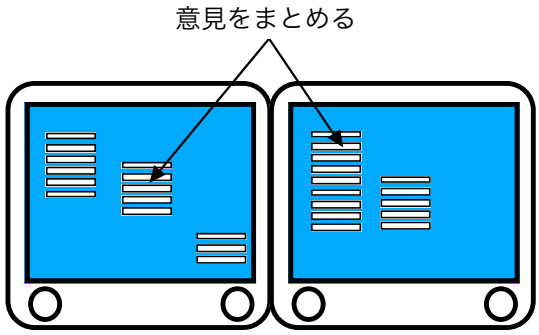

図7 ラベルの共同まとめ作業実

(3) GLIAアプリケーション使用実験

GLIAアプリケーションの総合的な使用感を調査するために，実験参加者にGLIAアプリケーションを実際に操作してもらった後，記述式のアンケート調査を行った．実験に使用したアプリケーションは多入力化したペイントソフトとKUSANAGI-Gである．

多入力化したペイントソフトは複数マウスで同時に描画できる領域を持ち，各マウスごとに，線の太さと色の選択，描いた絵の消去ができる．

3.2 GLIAによるアプリケーション開発の評価

GLIAは多入出力用のGUIをSwing（JavaのGUI開発環境)と類似の方法で開発できる．これにより既存のSwingアプリケーションを多入力へ対応させることが効率よくできると期待できる．

これを評価するために，実際にGLIAを用いて既存アプリケーションを多入力化し，そのときに変更したプログラム行数を計測した．評価に使用した既存アプリケーションは，KUSANAGI，○×ゲーム，ペイントソフトである．以下で各プログラムの具体的な機能を説明する．

(1) KUSANAGI

KUSANAGIの多入力化するために，複数マウスにより意見を移動できる機能，２画面間のドラッグ継続機能を追加した．２画面間のドラッグ継続とは，画面間で意見をドラッグしつつ移動できる機能で，画面間での意見の移動をスムーズにするために追加した．

(2) ○×ゲーム

Swingを用いて○×ゲームを作成し．それをGLIAで２つのマウスで操作できる○×ゲームに記述し直した．このときに追加した機能は，２つのマウスで操作できる機能，先攻，後攻の順にしか駒を置けない機能である．

(3) ペイントソフト

Swingをもちいてペイントソフトを作成し．それをGLIAを用いて複数ユーザ用ペイントソフトに記述し直した．Swingによるペイントソフトでは，マウスで線を描画できる領域を持ち，色の選択，線の太さ選択，描いた絵の消去ができる．GLIAにより多入力化したペイントソフトは，複数マウスで同時に線を描画できる領域を持ち，各マウスごとに色の選択，線の太さの選択，描いた線の消去ができる．

3.3 実験環境

実験には島根大学総合理工学部の学部生３名，院生２名，教員１名の計６名が参加した．実験に使用した計算機は『Mac OS X ver.10.3， 17インチ 700MHz PowerPC G4，256M』である．

４．実験結果と考察

4.1 操作性実験

操作性評価実験の結果を表１，表2，表３にまとめる．表１は平均作業時間と，まとめ作業実験の操作性に関する５段階評価のアンケート結果である． 表２は実験参加者にKUSANAGI-Gと

表1 評価結果

	KUSANAGI-G	KUSANAGI	
Ping実験の平均作業時間	３分５３秒	４分５８秒	*
まとめ作業実験の平均作業時間	8分24秒	7分30秒	
操作は正確か	3.3	4.0	**
操作は容易か	4.5	3.5	*
操作は素早いか	4.3	3.8	*
多画面結合の機能は有効か	4.2	3.8	
共同作業は有効か	4.3	3.7	

T検定の結果：*p<0.01,**p<0.05

表2 比較結果

比較内容	平均値
ラベル移動の行い易さ	4
作業を行い易さ	3.7
コミュニケーションの取り易さ	3.7

表3 GLIAアプリケーション使用後のア ンケート結果

，分担作業では有効
，２画面を扱えることで作業し易かった
，自由に操作できてよい
，相手の動作との競合が起きやすい
，２画面を扱うときはマウスの反応が悪い

KUSANAGIを５段階評価で比較させた結果であり点数が高いほどKUSANAGI-Gの評価が高い．表３は複数ユーザ対応ペイントソフトとKUSANAGI-Gを使用してのアンケート結果である．

表１のPing実験の平均作業時間は，KUSANAGI-Gの方が短い．これよりKUSANAGI-Gを用いることで，画面間でGUI部品を移動するような作業の操作性を改善できることが分かった．

また表１の評価結果より，簡単さ，素早さの評価はKUSANAGI-Gの方が高かった．これより多画面での作業で簡単さ，素早さの操作性が改善されたと言える．しかし正確さの評価はKUSANAGIの方が高い．これはGLIAが生成するマウスカーソルの動きが不正確なのが主な原因だと考えられ，今後改善する必要がある．

表２よりすべての項目でKUSANAGI-Gの方が評価が高く，多画面での作業を行いやすいと参加者は評価している．

表３よりGLIAを用いることで協調作業において自分の思うように作業できる部分が増え，協調作業がスムーズに進むことが分かった．また相手と行う作業が重なったときに，アプリケーション側で排他的な制御を取る必要があることも分かった．

4.2 GLIAによるアプリケーション開発の評価

表４にGLIAをもちいて既存アプリケーションを多入力化する際に変更した行数をまとめた．１列目はアプリケーション名，２列目は変更前のプログラム行数，３列目は変更後のプログラム行数，４列目はプログラムの変更行数である．

表４よりオセロ，○×ゲームのようにイベント処理が少ないアプリケーションは変更行数が少なく，ペイントソフトのようにイベント処理が多いアプリケーションは変更行数が多い結果となった(ここでKUSANGIはドラッグ継続機能をつけているので考察の対象から外した)．

これを詳しく診るために，各アプリケーションのマウスイベント処理数と多入力化のための変更行数を比較し，表５にまとめた．１列目はアプリケーション名，２列目はマウスイベント処理関数の数，３列目はマウスイベント処理関数の行数，

表4 多入力化によるプログラム行数の変化

	Swing	GLIA	多入力化による変更行数
KUSANAGI	7661	7686	27 (1%以下)
オセロ	723	733	14 (2%)
○×ゲーム	133	137	8 (6%)
ペイント	181	227	60 (30%)

単位：行

表5 マウスイベント処理数と変更行数

	マウスイベント		多入力化による変更行数
	関数の数	行数	
オセロ	1	14	14
○×ゲーム	1	11	8
ペイント	5	33	60

単位：行

４列目はプログラムの変更行数である．表5よりマウスイベント処理関数の数と変更行数のあいだに相関があり，主にマウスイベント処理部分を変更することで多入力化できることを示すことができた．マウスイベント処理部分の変更は多入力化のためには必要な変更なので，GLIAを用いることで既存のSwingアプリケーションを効率よく多入力化できると言える．

4.3 関連研究

マルチマウス環境の実現例を紹介する．Groupkit[3]はグループウェアを開発するためのツールキットで，これで開発されたアプリケーションは，１人が１台のPCを使いマルチカーソル環境を実現した共有ウィンドウ上で協調作業を行うことができる． PebblesDraw[4]はPDAをマウスのように使用するシステムで，複数のPDAで同時に１台のPC画面を共有操作できる．MID[5]は複数のマウスで１台のPC画面を操作できるJavaパッケージで，JavaをつかったGUI開発ができる．

GLIAの目指すソフトウェアによる出力装置の協調を実現したシステムを紹介する．GDA[6]はPDAをもちいた共同作業を支援するグループウェアで，複数のPDA画面を結合させて1つの大きな画面と見立てたり(画面結合)，1つの画面を複数のPDA画面で共有することができる（画面共有）．KUSANAGI[2]は複数画面の画面結合と画面共有ができ，それを用いてＫＪ法を支援する．Dynamo[7]は複数のユーザが複数の画面を操作できるシステムで，2台のディスプレイの画面結合ができるが，この結合をソフトウェアで実現しているか不明である．

以上のシステムを機能ごとに比較したものを次ページ表6に載せた．

マルチカーソル環境は上記のように多くあるが，Javaで実装されたものが無かった．MIDはJavaで作られているが，アプリケーション制作者がGUI部品を作成しなければならないので不便だった．GLIAではSwingコンポーネントを使用することが可能であり，比較的簡単にマルチカーソル環境上のアプリケーションを開発することができる．

またGDA，KUSANAGIのようなグループウェアは多出力同士を協調させる機能を持っている．GLIAでもこのような機能を持ちアプリケーションに提供したい．

表6 実現機能の比較

機能	GLIA	KUSANAGI +GLIA(参考)	GroupKit	Pebbles Draw	MID	GDA	Dynamo
マルチカーソル	○	○	○	○	△	×	○
マウスと画面のネットワーク結合	○	○	×	×	△	×	○
画面結合	×	○	×	×	×	○	ー
画面共有	×	○	○	×	×	○	ー

○：実現，△：一部実現，×：実現していない，ー：不明

6 おわりに

本研究では多入出力を扱うシステムの共通基盤としてミドルウェアGLIAを開発した．現在，GLIAはマウスを入力装置，ウィンドウ画面を出力装置とし，複数マウスによる1ウィンドウ画面の操作，マウスによる複数画面の操作，これを組み合わせて複数マウスによる複数ウィンドウの操作が可能である．またGLIAはGUIの開発をSwingのGUI開発方法と類似な方法で行うことができる．

GLIAを用いて開発したアプリケーションの操作性を評価した結果，次のようなことが分った．

(1) GLIAを用いることで，画面間でGUI部品を移動するような作業や，多画面を用いた協調作業で操作性が向上した．
(2) GLIAを用いて既存アプリケーションを多入力化させたところ，主要な変更をイベント処理部分内に押さえることができた．

今後は操作性に関する正確さを改善することと，対応する入出力装置を増やすことを予定している．例えばキーボード，無線入出力機器を検討している．

参考文献

[1] https://jinput.dev.java.net/

[2] 樫野 晶文 et al.:“複数PC画面を利用するKJ法支援ソフトの提案”,IEEE HISS論文集,pp.219-220(2003).

[3] Roseman, M. et al.: “GroupKit: A groupware toolkit for building real-time conferencing applications.” Proc. of CSCW’92, ACM, pp.43–50 (1992).

[4] Myers, B. et al.: “Collaboration Using Multiple PDAs Connected to a PC”, Proc. of CSCW’98, ACM, pp.285-294 (1998).

[5]Hourcade, H.P. et al.: “Architecture and Implementation of a Java Package for Multiple Input Devices (MID),” HCIL Tech Report No. 99-08 (1999).

[6] Munemori, J. et al.: “Group Digital Assistant: Combined or Shared PDA Screen”, Proc. of ICDCS’04, IEEE, pp. 682-689 (2004).

[7] Izadi, S. et al.: " Dynamo: A public interactive surface supporting the cooperative sharing and exchange of media " , Proc. of UIST’03, ACM, pp. 159-168 (2003),