3J-4 YyonX における Yy-server の設計

田中啓介[†1]，古坂孝史[†2]，井田昌之[†1]
青山学院大学情報科学研究センター 研究教育開発室

1 はじめに

Yy Window Tool Kit[1] は既存のウィンドウシステム上で一つのアプリケーションプログラムとして動作し，複数のウィンドウを用いた計算機利用環境を提供する．下層ウィンドウシステム (NWS: Native Window System) に依存しない共通のウィンドウ操作環境を提供する機構として *Yy-server* を設計し，LISP 言語で記述される処理の中核を *Yy-client* として設定した [1].

本稿では，X Window System を NWS として設計した *YyonX* における *Yy-server* について述べる．

2 Yy-serverでのウィンドウの扱い - テリトリ -

Yy-server では *Yy-client* における処理の単位であるウィンドウや，そのウィンドウ上に描画された文字・図形の意味的な区別を行なわない．*Yy-server* では *Yy-client* からの描画等のウィンドウ操作の対象をすべて「テリトリ」という概念で扱う．*Yy-client* の描画操作はテリトリに対する描画となる．テリトリは矩形の領域であり，その領域を画面上に表示することが出来る．

テリトリは表示上の単位となるとともに入力の単位ともなる．画面上には複数のテリトリを混在させることが出来る．そして，それぞれのテリトリがマウスイベントやキー入力イベントを独立して取り扱うことが出来る．

Yy-client は画面上の任意の位置にテリトリを設定でき，あるいは消滅させることができる．また，各テリトリに対して図形の描画，テリトリの属性の変更が可能である．テリトリ内でのイベントの発生は *Yy-server* から *Yy-client* に伝えられる．イベントはテリトリごとに認識されるので *Yy-client* ではイベント処理の異なる領域を別のテリトリとして宣言するだけでよい．

Yy-client における「ウィンドウ」[2] もテリトリの集合体で実現される．テリトリの分割 / 定義はクライアント側で行なう．テリトリの集合をウィンドウ操作上の概念に置き換えることもクライアントの役割である．

Yy-server では 定義されたテリトリについて，1) テリトリ間の相互関係の把握，2) テリトリに関するイベントの取り扱い，3) テリトリに対する文字・図形の描画 / 表示，の管理を行なう．

3 テリトリの相互関係

テリトリは階層的に定義される親子関係を持つ．すべてのテリトリは *Yy-client* でのルートウィンドウに対して割り当てられるテリトリに対して直接的，あるいは間接的に子のテリトリとなる．この親子関係により次のような処理が可能である．

Design of *Yy-server* for *YyonX*,
Keisuke TANAKA[†1], Takashi KOSAKA[†2], Masayuki IDA[†1],
[†1]Aoyama Gakuin University, [†2]Aoyama Gakuin University/CSK Corp.

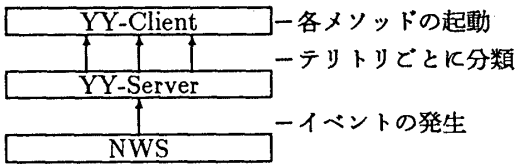

図 1: イベント処理の流れ

1. 親のテリトリの消滅はその子供であるすべてのテリトリの 消滅につながる．
2. あるテリトリの表示を停止するとそのすべての子テリトリも 表示上消える．
3. 親テリトリの表示位置の変更は子テリトリの位置の変更となる．但し，表示位置の変更を行なったテリトリと各子テリトリの 相対的な位置関係は変わらない．

Yy-client はこの機能を利用してテリトリ群に関する一括処理を行なうことができる．

テリトリの大きさは親子の関係に無関係に設定できる．但し，子供のテリトリの表示は親テリトリの表示範囲によって制限される．*Yy-client* におけるウィンドウ表示はウィンドウに対応する大きさのテリトリを宣言し，その子テリトリとして world に対応するテリトリを宣言している．この宣言により表示上はウィンドウ相当のテリトリで制限され，描画は world 相当のテリトリに行なうことが可能となる．

テリトリはその情報定義の新旧を表現する世代のデータを持つ．直接の親子関係にないテリトリは，画面上では世代の新しい方を上にして表示する．したがって，画面上での上下関係は世代の変更によって可能となる．

4 テリトリに対するイベント

すべてのテリトリでマウスイベント / キー入力イベントを取り扱うことができる．NWS で発生したイベントは *Yy-server* に伝えられる．NWS ではテリトリの判別ができないため，*Yy-server* がどのテリトリに対するイベントであるかを判断する．そして対応するテリトリへのイベントを *Yy-client* に伝える．*Yy-client* ではイベントに対応したメソッドを起動し処理を進める (図 1).

しかし，多数存在するテリトリのすべてでイベント報告を行なうことはサーバ＝クライアント間の通信量増大につながるので，各テリトリには処理すべきイベントのためのマスクを設定する．*Yy-server* はこのマスクによって設定されたイベントのみを *Yy-client* に報告する．

4.1 マウスイベント

アプリケーションによっては，あるイベントに対しての処理中に他のイベントが発生すれば直ちにそちらの処理を行なう場合もある．そこで，マウスイベントには優先度を設定する．優先度としては，1) 特別優先度，2)

高優先度，3) 通常優先度，の三種を設定できる．3 はあるイベントに対する処理中は *Yy-client* には報告されないイベントに対するものである．処理の終了を待って，順次報告され処理が行なわれる．2 は他のイベントに対しての処理中であっても *Yy-client* に報告されるイベントに対するものである．*Yy-client* では必要に応じて処理を中断し，そのイベントに対する処理を行なうことが出来る．但し，*Yy-client* の中で他のイベントの発生による割り込みを防ぐために 2 のイベントの報告を遅らせることは可能である．1 はこのような割込禁止状態であっても強制的に割り込みを発生させるための優先度である．これは強制的なアプリケーションの終了などを想定して設定している．

Yy-server で扱うことのできるマウスイベントは以下の通りである．

1. マウスカーソルのテリトリへの in / out
2. マウスカーソルのテリトリ中への設定時間以上の駐留
3. マウスカーソルがテリトリの中で移動
4. マウスボタンの押下 / 開放

4.2 キー入力イベント

キー入力に関しては優先度を設けず順次テリトリ内でのキー入力を *Yy-client* に報告する．

キー入力処理部ではかな漢字変換を可能にしている．*Yy-client* には変換の結果が通常の入力と変わらない形で報告され，途中経過は報告されない．かな漢字変換機能は，キーボードからの変換起動キーの入力，*Yy-client* からの起動要求によって起動される．変換の起動キーや変換キーなどの設定は *Yy-client* によってなされる．また，かな漢字変換に使用する機構や辞書なども *Yy-client* から指定される．

5 テリトリへの描画，画面への表示

テリトリ内に描画した文字や図形は，そのテリトリが表示されている場合に画面に表示される．表示する文字，図形は *Yy-client* から指定される．

テリトリへ描画されたものは，テリトリが破壊されるまで *Yy-server* 内に保持される．描画内容の保存はテリトリ表示の有無に関わらず行なわれる．したがって，テリトリの移動などにともなって生じる画面の再描画の際に *Yy-client* から表示する情報を再度送る必要はない．

6 テリトリ管理のためのデータ

Yy-server では各テリトリに対し次のようなデータを持つ．

- 親テリトリ
- 子テリトリ
- 位置：親テリトリの原点からの相対位置
- テリトリの大きさ
- 角度：斜めのテリトリの存在を仮定する
- 背景色，背景パターン
- テリトリ内の表示物
- 描画の可否
- 表示の可否：親テリトリが表示されない場合はこの情報に関わらずテリトリの内容の表示は行なわれない
- 世代：テリトリの画面上での上下関係の制御情報

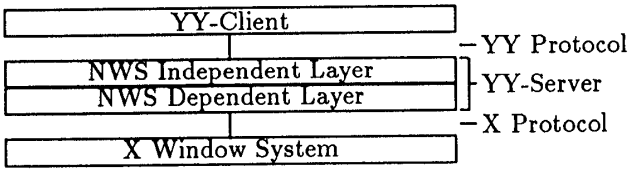

図 2: X Window System 上での実現

- イベントに対するマスク，優先度
- かな漢字変換処理のための情報

7 サーバ＝クライアント間の通信 -Yy Protocol-

Yy-server と *Yy-client* の間はプロセス間通信機能を利用して情報の交換を行なう．*Yy-client* がテリトリを作成するごとに，テリトリ制御のための通信路とイベント入力のための通信路が仮想的に設定される．

テリトリ制御のための通信路は描画指示や位置変更などのテリトリに対する操作を行なったり，テリトリから情報を得るための通信路であり，同期をとって通信が行なわれる．イベント入力はサーバ側からのイベントの報告に使用される，この通信はクライアント側の状態に関わらず非同期に行なわれることもある．

8 X Window System 上での実現

現在，青山学院大学においては X Window System 上に *Yy-server* の実現を進めている．X Window System から見た Window は *Yy-client* でのルートウィンドウに対応したテリトリのみである．

テリトリの制御はすべて NWS に依存しないレイヤで処理される．NWS に依存するレイヤは必要な描画命令を NWS に伝え，あるいは NWS からのイベントの報告をすべて NWS 依存レイヤに伝えるインタフェースの役割のみを果たし，それ以外の特別な処理は行なわない．X Window System 対応版ではこのレイヤは Xlib 関数の呼び出しのための関数群だけである．

ルートウィンドウに対するテリトリを除くすべてのテリトリは X Window System からは操作の対象とはならない．これによってテリトリの制御を NWS から独立させた．そのため NWS 依存レイヤ関数群を他の NWS に対応するものに変更するだけで他の NWS への移植が可能となる．NWS に要求される機能は 1) ルートテリトリに対応する Window の確保，消滅，2) 1 の Window 内での描画制御，3) 1 の Window 内でのイベント確保，という三種に限定された．

このように NWS での利用機能が簡単になった反面，NWS の機能と重複する機能を *Yy-server* 中に持たせる必要が生じた．このため *Yy-server* と X Window System (X サーバ) 間の通信量が大きくなってしまうことになった．この欠点を克服するため，NWS にある程度の機能の分担を可能とするような拡張を検討している．

参考文献

[1] 井田昌之他，"*YyonX*: 概要設計"，情報処理学会第 40 回全国大会, March 1990.

[2] 古坂孝史他，"*YyonX* における YYWS の設計"，情報処理学会第 40 回全国大会, March 1990.