4K－8

X Visual Effect Extension における ダイナミックプレーンアーキテクチャ

高野 元　松浦 宏*　的場 ひろし
NEC C&C 研究所　(株)NEC 情報システムズ*

1 はじめに

ウィンドウシステムにおいてウィンドウ操作が不連続な画面変化を引き起こすことを、我々はユーザインターフェース上の問題と考えている。これに対処するために、画面変化にフェード・ワイプといった視覚効果を取り入れてユーザの使用感を向上させる試みを、X Window System 上の拡張機能 X Visual Effect Extension [1, 2, 3] として実現している。

視覚効果は VRAM への描画関数に手を入れて実現しているが、X サーバのウィンドウ管理方式の仕様上、EXPOSE 処理やバッキングストア処理と視覚効果処理の調停が上手くいかずに、ウィンドウ画面が崩れてしまう問題があった。

この問題に対処するために、視覚効果処理時に動的に生成される“ダイナミックプレーン”という概念を導入する。この上で、ウィンドウイメージの再構築が行われ、視覚効果描画と独立に処理される。本稿では、このダイナミックプレーンプレーンについて説明する。

2 X Visual Effect Extension

XVE は、表1のように視覚効果とウィンドウ操作を対応付けている。

表 1 XVE における視覚効果とウィンドウ操作

視覚効果	ウィンドウ操作	関数例
フェードイン ワイプイン	出現	XMapWindow()
フェードアウト ワイプアウト	消失	XUnmapWindow()
ズーム	リサイズ	XResizeWindow()
スライド	移動	XMoveWindow()

また XVE サーバは、視覚効果を出力する“視覚効果処理ルーチン”と、そこから参照されるパラメータ集合“エフェクトコンテキスト”を導入して機能拡張している (図 1)。視覚効果処理ルーチンは、DDX-mi レイヤにおいてウィンドウ描画を行う関数を交換 (wrap) することで実現している。

"The Dynamic-Plane Architecture in the X Visual Effect Extension," Hajime Takano, Hiroshi Matsuura*, Hiroshi Matoba, C&C Reserch Labs., NEC Corporation, NEC Informatec Systems, Ltd.*

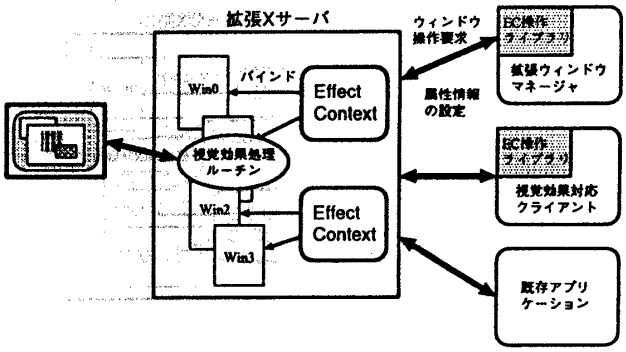


図 1 XVE のアーキテクチャ

3 XVE の描画問題

XVE の視覚効果機能を実装する場合、X のウィンドウ管理モデルが原因となって、下記の問題が生じる。

階層管理による問題: X サーバはウィンドウをウィンドウ木によって階層的に管理しており、これに従ってウィンドウの可視領域を計算する[1]。あるウィンドウが操作されると、ウィンドウ木と各ウィンドウの可視領域が再計算され、親から子の順に再帰的に描画されていく。

このため、フェードインやワイプインといった視覚効果が、親からこの順に一つずつかかって行くことになる。

背景の再描画の問題: あるウィンドウをアンマップした時点は、背景に隠されていたウィンドウの内容が欠けており、後からアプリケーションまたはバッキングストアによって再描画しなければならない。すなわち、フェードアウトやワイプアウトを実行すると、ウィンドウが消えた後に存在するはずのウィンドウのイメージが存在しないことになる。

描画モデルによる問題: X は VRAM に描画できなかったラスタイメージは捨ててしまう。また、フェードやワイプといった視覚効果は、描画すべきマスク領域の形を連続的に変化させていく。つまり、視覚効果処理中に描画要求のあったグラフィクスオブジェクトは、リクエストのあった時点のマスク領域内にしか描画されない。よって、描画したはずのグラフィックオブジェクトが欠けてしまい、イメージが不完全なものに

[1] ウィンドウは、自分の弟あるいは子供にしか隠されることはない。

なる。

以

- 関係があるウィンドウでも独立して描画する
- 視覚効果のマスク領域外への描画は捨てられる

ことに起因している。

4 ダイナミックプレーン・アーキテクチャ

前節の議論によれば、より良い視覚効果表示を実現するためには、

1. 関係があるウィンドウはまとめて描画する
2. 視覚効果のマスク領域外への描画も捨てない

ことが求められる。

これに対処するために、“ダイナミックプレーン”(以下DP) という概念を導入する。DPは、VRAMとは独立のウィンドウ描画プレーンであり、必要に応じて主記憶上に動的に確保される。

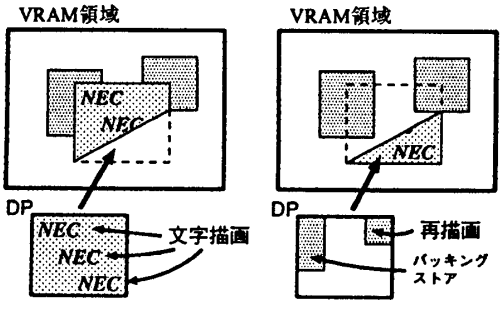
一つ一つのウィンドウへの描画は、まずDPに対して行われ、ウィンドウ全体のイメージが構築されてからVRAMへ描画する方法を採用する。すなわち、サブウィンドウおよびグラフィクスオブジェクトの再描画、バッキングストア内のイメージは、まずDPに描画され、これを視覚効果用の描画アルゴリズムがVRAMへ描画する。

4.1 ワイプの例

ワイプイン、ワイプアウトの例を示したのが図2である。ワイプインは、DP上に表示したいウィンドウの最終的なイメージを構成し、このイメージの一部を連続的にVRAMへコピーしていく。

一方ワイプアウトは、現在背景に隠されているイメージを、EXPOSEイベントやバッキングストアを用いてDP上で修復し、このイメージの一部を連続的にVRAMへコピーしていく。

いずれも、ダイナミックプレーンは、操作するウィンドウと同じ領域を持つことになる。



(a) ワイプインの例　(b) ワイプアウトの例

図 2 ダイナミックプレーンを用いたワイプ

4.2 ズームの例

ズームによるウィンドウ拡大の例を示したのが図2である。DPはウィンドウの拡大後の大きさを確保し、現在表示中のウィンドウイメージをDPにコピーする。サイズ変更に伴うアプリケーションからのリクエストは、DP上に描画される。このDP上のイメージを連続的にVRAMにコピーする。

5 実装

XVEサーバは、その視覚効果機能をDDX-miレイヤで実行している。よってDPの実装に当たっては、

- 視覚効果を伴うウィンドウへの描画を行う時、VRAMではなくDP領域とした主記憶上に描画する

ようにDDX-miレイヤに変更を加えた。DPからVRAMへの描画に用いる視覚効果処理ルーチンは、これまでのプロトタイプに変更を加えずに使用できた。DDX-miレイヤレベルでの実装なので、移植性は高くさまざまなプラットフォームに移植可能である。

これにより、ウィンドウまたは背景のイメージは一体となった視覚効果を表示することができるようになった。図3は、ワイプアウトの様子を示しているが、背景があたかもそこに存在しているように表示されていることがわかる。

図 3 プロトタイプ画面例

6 おわりに

XVE の視覚効果を、より品質の高いものにするために、ダイナミックプレーンの概念を導入した。これにより、関係のあるウィンドウ - サブウィンドウ、あるいは背景が一まとまりのイメージとして視覚効果表示できるようになった。また、DDX-mi レイヤでの実装につき、高い移植性も実現している。

参考文献

[1] 高野元, 松浦宏, 的場ひろし. X Visual Effect Extension (1) ～視覚効果を用いたウィンドウ操作とそのアーキテクチャ～. 情報処理学会第 49 回全国大会, No. 2V-5. 情報処理学会学会, 1994.

[2] 松浦宏, 高野元. X Visual Effect Extension (2) ～インプリメントと評価～. 情報処理学会第 49 回全国大会, No. 2V-6. 情報処理学会学会, 1994.

[3] Hajime Takano, Hiroshi Matsuura, and Hiroshi Matoba. XVE : X Visual Effect Extension. In *The X Resource*, No. Issue Eleven. X Consortium, 1995.