2002 年度　卒業論文

Mac OS における画像ファイルの入出力

奈良女子大学理学部情報科学科

数理情報学講座 和田研究室

許　潤玉

Introduction

本論文は，Mac OSにおける画像ファイルの入出力についてまとめたものである．ここでは，アプリケーションソフトDeltaViewerを実例としてあげ，このDeltaViewerにおける画像入出力の操作性を向上させるために行った処理について具体的に述べる．なお，DeltaViewerとは，和田研究室で開発しているアプリケーションソフトで，共焦点レーザー顕微鏡などから得られる二次元連続断面画像から任意の境界を抽出して，計算によって物体の表面を再構築し，それを，OpenGLを用いてコンピュータ画面上に表示するものである．

本研究では，画像ファイルの入出力ついて大きく２つの改良を試みた．

対応ファイル形式の拡張

DeltaViewer における入出力を， QuickTime のサポートする画像ファイル形式全てに対応させた．

ダイアログボックスの改良

ダイアログボックスを拡張して画像形式ポップアップメニューを追加し，ファイル拡張子と FileType の整合性を損なうことなく，それを各種イベントにも対応させた．また，ファイル名入力による画像形式の判別を可能にした．

目次

１．旧バージョンの DeltaViewer における入出力

旧バージョンの DeltaViewer における読み込み可能な画像ファイル形式は TIFF ファイル（圧縮なし）のみであった．これは，当初 DeltaViewer が，共焦点レーザー顕微鏡から得られる画像を取扱うことを前提として作られたためである．

共焦点レーザー顕微鏡は，生物学の分野でよく使用されているもので，共焦点機能を用いることで焦点面以外の画像を取り除き，焦点面のみの画像を取得できるため，物体の連続断面画像をコンピュータに読み込むことができる．この共焦点レーザー顕微鏡における保存可能なファイル形式はメーカーによって異なるのだが，その中で共通して保存可能であるファイル形式が TIFF ファイルであるため，それに対応させたものと思われる．

しかし，メーカーによっては様々なファイル形式で画像を保存することができるので，TIFF ファイルしか対応しないのではあまりにも柔軟性がなさすぎる．また，将来は共焦点レーザー顕微鏡から得た画像だけでなく，電子顕微鏡など，様々な機器から得た連続断面画像も取扱うことを想定して，DeltaViewer における入力を拡張する必要があった．

さらに出力部分においては, DeltaViewer 独自の３Ｄ画像の保存は可能なのだが，二次元画像の保存機能は搭載していなかった．従ってその操作性を向上させるためには，この保存機能も追加することが望ましいと考えられた．

２．対応ファイル形式の拡張

本研究では，Mac OS のライブラリである QuickTime の使用により，DeltaViewer の入出力を QuickTime のサポートする画像ファイル形式全てに対応させようと試みた．

２．１．　QuickTime 概要

今回使用した QuickTime とは，一般的に知られている QuickTime プレーヤーではなく，QuickTime ライブラリのことである．より高いレベルのソフトウェアはタイムベースのデータをコントロールするために使用することができる．QuickTime はビデオデータの処理に加えて，画像や音楽，テキストなど，様々なオブジェクトを扱うことができ，アプリケーションに加えることができる．

QuickTime は，Mac OS，Windows95/98/NT のためのプラットフォーム技術である．Mac OS 及び Windows95/98/NT におけるアプリケーションは異なって作られているが，QuickTime へのインターフェースは事実上同一である．

QuickTime は，コンピュータの内臓ソフトとして搭載されていることが多く，仮に搭載されていなかったとしても，Apple のサイトで無料配布されているので簡単に得ることができる．また，QuickTime の７０を超えるフォーマットの入出力データは，最も一般的なマルチメディアおよび圧縮基準を含んでいる．さらに将来，QuickTime の新しいバージョンが新しいフォーマットをサポートしても，既存の QuickTime アプリケーションも自動的にアクセスし，新しいフォーマットでも動くようになるという利点もある．

QuickTime を使用するために，時間に基づいたデータと関係のある，atoms，media structures，components，time management，sprites など，いくつかの概念を理解する必要がある．そこで本研究では，この概念の中の components に着目したいと思う．

QuickTime は，プラグイン・コンポーネントをサポートしていて，このコンポー

ネントを用いれば，アプリケーションは， QuickTime によって成される全ての技術および装置のことを知る必要がなくなる．画像の圧縮および解凍のような多くの QuickTime サービスが，コンポーネントによって提供されている．また，コンポーネントには，システム全体あるいは特定のアプリケーションに用いることができる様々なコードが含まれている．各 QuickTime コンポーネントは，アプリケーションに的確なインターフェースを提供することができる．従ってコンポーネントを用いれば，アプリケーションは，様々な技術をインプリメントしたり管理しなくても良いのである．

QuickTime には約 180 の内蔵コンポーネントがあり，さらに新しいコンポーネントを作成するための API もある．例えば，特別なデータ暗号化アルゴリズムをサポートするコンポーネントを作成したとすると，アプリケーションは，そのアルゴリズムをインプリメントしなくても，このコンポーネントにアクセスすれば暗号化を実行することができる．アプリケーションはシステムレベル・コンポーネント・マネジャーをコールすることにより，コンポーネントにアクセスすることができる．

＜ コンポーネントの例 ＞

- movie controller components：アプリケーションに，標準ユーザー・インターフェースによってムービーを演じさせる．
- image compressor components：画像データを圧縮または解凍する．
- image compression dialog components：ユーザーに圧縮オペレーション用のパラメータを指定させる．
- image transcoder components：圧縮ファイルを，あるフォーマットから別のフォーマットに変換する．
- video digitizer components：アプリケーションに，外部装置によってビデオのデジタル化をコントロールさせる．
- data-exchange components：アプリケーションに，QuickTime ムービーの中および外にある様々なタイプのデータを移動させる．
- video output components：QuickTime ムービーをビデオに変換させる．
- graphics import components：QuickTime ムービーにグラフィックスを入力させる．
- music components：QuickTime ムービーの中の音楽トラックを処理し，合

成させる.

- effects and transitions components:QuickTime ムービーの中の 133 SMPTE の標準効果をインプリメントする.
- preview components：ファイル内の視覚的なプレビューを表示し作成するムービーツールボックスの標準ファイルプレビュー機能によって，使用される.

本研究では，このコンポーネントの中の，入力においては Graphics Importers，出力においては Graphics Exporters というコンポーネントを用いた.

２.２. Graphics Importer Components

Graphics Importer Components は，様々なファイルフォーマットや圧縮アルゴリズムを使用して格納されたグラフィックス画像を，開いたり表示したりするために使用される．Apple が提供するコンポーネントは，多くの共通した圧縮画像タイプを読込むことができる.

＜ 画像ファイルを読み込むために使う関数の例 ＞

```
void drawFile(const FSSpec *fsspec, const Rect *boundsRect)
{
    GraphicsImportComponent     gi;
    GetGraphicsImporterForFile(fsspec, &gi);
    GraphicsImportSetBoundsRect(gi, boundsRect);
    GraphicsImportDraw(gi);
    CloseComponent(gi);
}
```

読み込み処理

まずは GetGraphicsImporterForFile()に FSSpec を渡し，画像タイプに対応した

GraphicsImportComponent（gi）をオープンさせる．ここでは画像タイプを意識する必要はないが，もし指定されたファイルが Graphics Importer で取り扱えないタイプであれば，この時点でエラーが返される．先ほども述べた通り QuickTime では，バージョンが上がるたびに取り扱い可能な画像タイプが増えている．現在では PICT，JPEG，BMP，GIF，TIFF などの一般的な画像ファイルだけでなく，Adobe Photoshop のネイティブドキュメントや FlashPix ファイルなども取り扱い可能である．続いて GraphicsImportGetNaturalBounds()により画像ファイルの矩形サイズ（Rect）を得る．GraphicsImportSetGWorld()では，描画対象となる CGrafPtr（GWorld）をセットする．ここにオープンしたウィンドウの CGrafPtr をセットしておかないと，画像の描画はデスクトップ上になされてしまう．GraphicsImportSetBoundsRect()で描画矩形領域を指示し，後は GraphicsImportDraw()を呼ぶだけで，オープンしたウィンドウ上に画像が描画される．最後に CloseComponent()で GraphicsImportComponent を閉じれば，すべての処理は完了する．

＜ 現バージョンの QuickTime における，Graphics Importer Components がサポートしているファイル形式 ＞

- QuickDraw PICT
- QuickTime Image
- MacPaint
- Photoshop (versions 2.5 and 3.0)
- Silicon Graphics
- GIF
- JFIF/JPEG
- QuickDraw GX Picture (if QuickDraw GX is installed).
- BMP
- PNG
- TGA
- TIFF
- FlashPix file format on PowerPC and Windows platforms.

２.３. Graphics Exporter Components

Graphics Exporter Components は，様々なフォーマットの画像ファイルにグラフィックスを出力するための標準のインターフェースを提供するものである．QuickTime は，GIF や PNG など，希望の出力フォーマットに基づいた Graphics Exporter Components を自動的に選択してくれる．また，異なるイメージフォーマットは，解像度や圧縮などのような様々な特徴をサポートする．

＜ 画像ファイルを出力するために使う関数の例 ＞

```
void writeGWorldToImageFile( GWorldPtr gw, const FSSpec *fileSpec )
{
    GraphicsExportComponent    ge = 0;
    OpenADefaultComponent( GraphicsExporterComponentType,
                                        kQTFileTypePNG, &ge );
    GraphicsExportSetInputGWorld( ge, gw );
    GraphicsExportSetOutputFile( ge, fileSpec );
    GraphicsExportDoExport( ge, nil );
    CloseComponent( ge );
}
```

書き出し処理

最初に OpenDefaultComponent()で Graphics Exporter Components をオープンする．続いて画像データが保存されているオフスクリーンをロックし，それを GraphicsExportSetInputGWorld()に渡して作成する画像の対象とする．ファイル保存先の FSSpec 構造体を GraphicsExportSetOutputFile()で指示し，最後に GraphicsExportSetOutputFileTypeAndCreator()で出力ファイルのファイルタイプとクリエータを指定して GraphicsExportDoExport()を実行すれば，画像ファイルが保存される．

＜ 現バージョンの QuickTime における，Graphics Exporter Components がサポートしているファイル形式 ＞

- BMP
- JFIF/ JPEG
- MacPaint
- Photoshop
- PNG
- QuickDraw PICT
- QuickTime Image
- Silicon Graphics
- Targa
- TIFF
- JPEG2000

３．ダイアログボックスの設定・改良

Mac OS においてダイアログボックスのコントロールは，Mac OS のライブラリである Navigation Services が行っている．本研究ではこの Navigation Services を使ったプログラミングを行うことにより，ダイアログボックスを拡張して画像形式ポップアップメニューを追加し，それを各種イベントにも対応させた．また，ファイル名入力による画像形式の判別を可能にした. この章では, Navigation Services の説明と，具体的な対応方法について述べたいと思う.

３.１．Navigation Services 概要

Navigation Servicesとは，Mac OSに対応したファイル入出力のためのダイアログボックスを取扱うライブラリのことである.

ナビゲーションサービスはいくつかの重要な問題を解決するために開発された．標準ファイル・ダイアログボックスとFinderのファイル・インタフェースがさまざまな点で異なるため，Mac OS でのファイル参照は多くのユーザーにとって混乱のもとになっていた．また，ユーザーは，標準ファイル・パッケージが開発された当時とは比較にならないほど大きなサイズのボリュームをナビゲートする必要がある．このようにサイズの大きなデータ空間には拡張機能が必要とされた.

ナビゲーションサービスには，ドキュメント管理の領域で大幅に拡張されたさまざまなツールが用意されている．まず1 つの拡張は，複数のファイルを同時に選択してオープンできる機能である．また，ユーザーがマウントされている複数のボリュームを簡単に選択し，最近オープンしたファイルやフォルダを選択したり，気に入った項目を含む独自のリストを作成することを可能にするボタンも用意されている．さらに，遅延変換を選択し，ファイルをネイティブ・フォーマットで暫定的に保存し，ユーザーがファイルを閉じるまで処理時間の浪費となる変換タスクを行わないようにすることもできる．ナビゲーションサービスの拡張機能は簡単に使用することができる.

ナビゲーションサービスに含まれる関数はどれもシンプルで自己完結しており，できるかぎりカスタム・コードを書かなくてもすむように設計されている．また，このAPI では，Appearance Manager のダイアログボックスおよびユーザー・コン

トロールを対象とする拡張部分が自動的にサポートされ，複数のアプリケーションにわたってより一貫していてわかりやすいユーザー・インタフェースを実現できる．

（a）必要条件

Navigation Services1.0 は，ナビゲーション共有ライブラリ，Appearance Manager 1.0.1 またはそれ以降，およびMac OS 7.5.5 またはそれ以降を必要とする．ナビゲーションサービスの一部の拡張機能では，次のような他のコンポーネントを必要とする．

- グラフィックファイルのプレビューを作成して表示するにはQuickTime が必要である．QuickTime がインストールされていないと，プレビュー・オプションは無効になる．
- ファイルの変換とファイルのタイプの正しい表示を行うにはMacintosh Easy Open が必要である．
- オンライン・ヘルプを表示するにはApple Guide が必要となる．

（b）Navigation Servicesユーザー・インターフェース

ナビゲーションサービスは，ファイルのオープンと保存に使用する改善されたユーザー・インタフェースを提供する．この中には，ファイル・システムの参照と管理を行うための，使いやすいさまざまな機能が含まれている．ダイアログボックスには次のものがある．

- “開く”
- “保存”
- “ファイル選択”
- “フォルダ選択”
- “ボリューム選択”
- “ファイル・オブジェクト選択”
- “新規フォルダ作成”

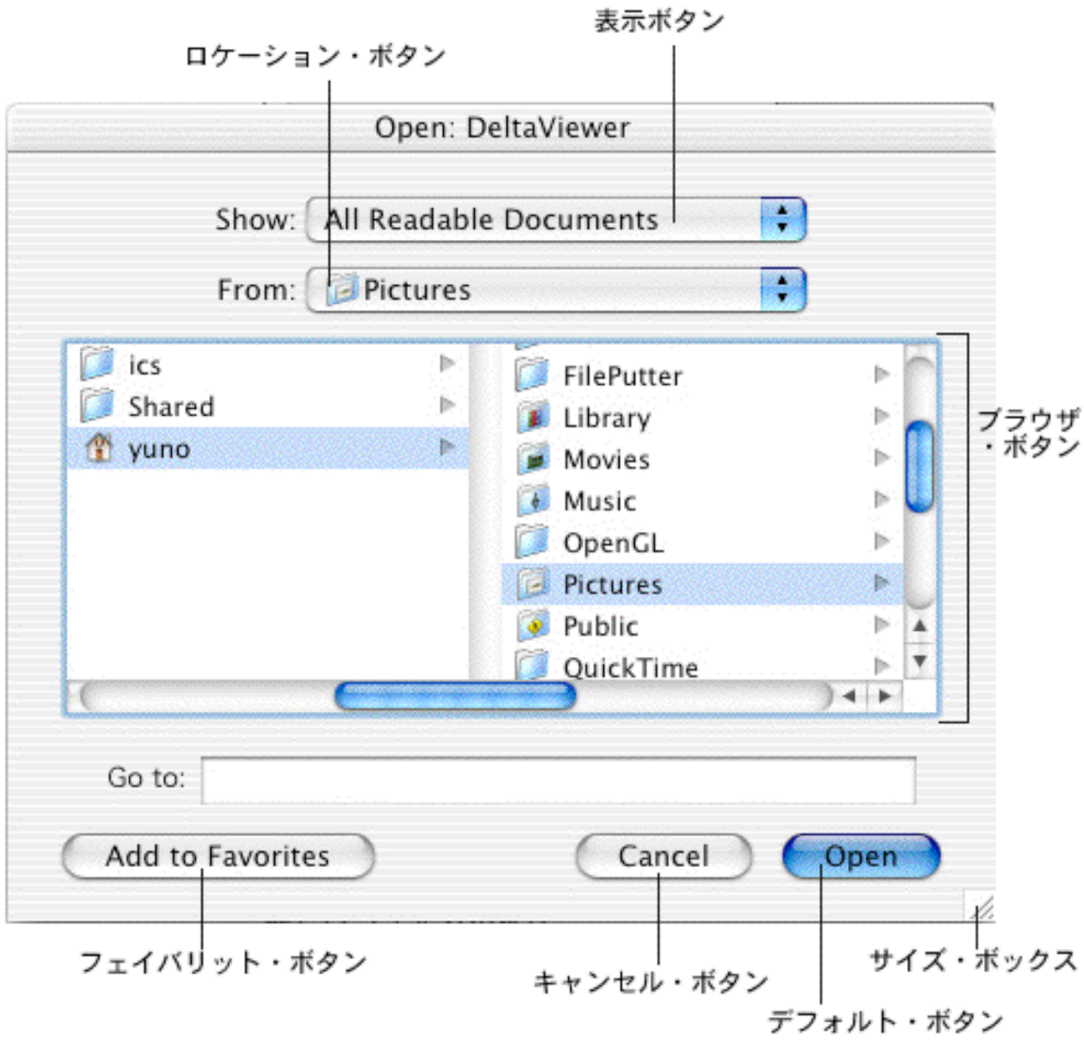
“変更を保存”と“変更を破棄”という２種類のアラート・ボックスも用意されている．また，全てのダイアログボックスは，いくつかの基本的なインターフェース

要素を共有している．以下，これらの要素を示す．

- ブラウザ・リスト
- ロケーション・ポップアップ・メニュー・ボタン
- ショートカット，フェイバリット・ボタン
- デフォルト・ボタン
- “キャンセル”ボタン
- サイズ・ボックス

図１に“Open”ダイアログボックスで使用されているこれらの要素を示す．

＜図１＞　標準ダイアログボックスの構成要素



３.２. Navigation　Servicesの使い方

ナビゲーションサービスは，アプリケーションがファイルのオープンと保存を行うためにシンプルかつ柔軟な方法を提供することを目的として設計されている．ファイルをオープンするための主要な方法としてはNavGetFile 関数の呼び出しがあり，ファイルを保存するための主要な方法としてはNavPutFile 関数の呼び出しがある．また，ユーザーにファイル・オブジェクト（ファイル，フォルダ，またはボリューム）の選択を要求するNavChooseFile関数が用意されている．本研究では主に，これら３つの 関数を使用してプログラミングを行った.

３.２.１. ファイルのオープン

NavGetFile 関数は，オープンするファイルの選択をユーザーに要求する，“Open”ダイアログボックスを表示する.

表示ポップアップ・メニューを使用すると，ユーザーは，ブラウザ・リストに表示され，ナビゲーションサービスによってオープンされるファイル・タイプを選択できるようになる．使用可能なfileTypeのリストは，NavGetFile 関数を呼び出したアプリケーションとTranslation Manager に含まれるサービスが提供する情報から構築される．NavDialogOptions 型の構造体のdialogOptionFlagsフィールドにkNavNoTypePopup 定数を指定すると，表示ポップアップ・メニュー・ボタンは表示されない.

ナビゲーションサービスではTranslation Manager を使って，アプリケーションが読み込むことのできるファイルを自動的にオープンして変換する．ナビゲーションサービスは展開された表示ポップアップ・メニュー（図２）を表示して，オープンできるファイル・タイプをユーザーに知らせる.

＜図２＞　表示ポップアップメニューの例　（Photoshopより）

（a）オープン時のファイル変換

ユーザーがNavTypeList 構造体で識別されていないfileTypeを選択した場合は，選択されたファイルを変換する必要がある．変換を必要とする「ファイル1」という名前のファイルをユーザーがオープンすると，ナビゲーションサービスは適切なfileTypeを使って，「ファイル1（変換済）」という名前の新しいファイルをオリジナル・ファイルと同じディレクトリに作成する．NavGetFile 関数は，戻り値を返す前に変換を自動的に実行する．しかし，NavDialogOptions 型の構造体のdialogOptionFlagsフィールドにkNavDontAutoTranslate定数を指定することで，この機能を無効にすることができる．自動変換を無効にすると，アプリケーションの判断で必要に応じてNavTranslateFile 関数を呼び出すことになる．

ユーザーが複数選択を行うと，NavReplyRecord 型の構造体のfileTranslation フィールドは変換レコードの配列をポイントする．NavReplyRecord 構造体の選択フィールドで識別されるそれぞれのファイル選択に対して1 つのレコードが対応する．自動変換を無効にしている場合は，このリストを使って独自の変換を行う必要がある．変換を行う必要のないエントリは空白で，空白のエントリは無視される．

自動変換を実行するとき，Finder で新しいフォルダを作成するときと同じよう

な方法で，ナビゲーションサービスは変換されたファイルに適切な名前を付けようとする．

３．２．２．ファイルの選択

NavChooseFile関数は，ユーザーにファイルの選択を要求するダイアログボックスを表示する．選択するファイルは，初期設定ファイルやその他の特殊ファイルなどである．

この関数は単純な関数であるため，組み込み変換は使用できない．ネイティブでないfileTypeをオープンしたい場合は，NavTypeList 型の構造体を指定するか，アプリケーション定義のフィルタ関数をインプリメントする必要がある．

アプリケーションでは，NavDialogOptions 型の構造体のメッセージ・フィールドに文字列を指定することができる．このメッセージはブラウザ・リストの下に表示される．文字列を指定しないと，メッセージは表示されない．

３．２．３．ファイルの保存

NavPutFile 関数は，図３のようなダイアログボックスを表示する．

＜図３＞　“保存”ダイアログボックスの例

ユーザーは，“新規フォルダ”ボタンを使って，ファイルを保存するための新規フォルダを作成できる．

“保存”ダイアログボックスでは，フォーカス・リングを表示して，ブラウザ・リストと編集可能テキスト・フィールドのどちらがキーボード・フォーカス（すべてのキー操作を受け付ける）を持っているかを示す．

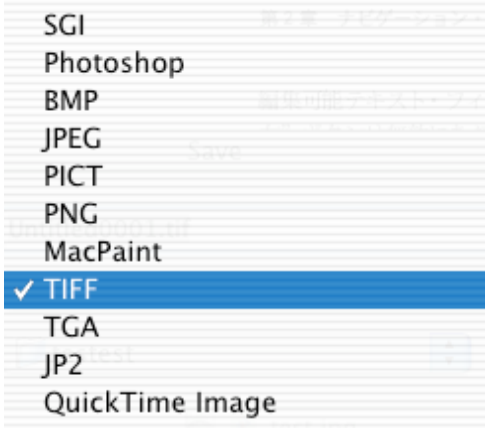
編集可能テキスト・フィールドにファイル名が表示されていないとき，“保存”ボタンは無効になる．

(a) ファイル・フォーマット・オプションの表示

NavPutFile 関数はフォーマット・ポップアップ・メニュー・ボタンを表示し，

新しいファイルまたはファイルのコピーが保存される方法をユーザーが選択できるようにする．図４ が，このメニューの例である．

＜図４＞　フォーマット・ポップアップ・メニューの例



フォーマット・ポップアップ・メニューの先頭の項目は，アプリケーションがNavPutFile 関数のfileType およびfileCreator パラメータで指定したファイルのタイプによって定義される．メニュー項目の名前はTranslation Manager から取得される．この項目を設定した後，ナビゲーションサービスはTranslation Manager を呼び出して，選択可能なfileTypeを示すその下のメニュー項目を表示するかどうかを決定する．

（b）保存時のファイル変換

アプリケーションは，保存するファイルのデフォルトのfileTypeとfileCreatorをNavPutFile 関数に指定する．ナビゲーションサービスではこの情報を使って，Translation Manager から取得した使用可能な変換方法を含むポップアップメニューを作成する．ユーザーがネイティブ・タイプとは異なる出力fileTypeを選択すると，ナビゲーションサービスは変換仕様を準備し，そのためのハンドルをNavReplyRecord 型の構造体のfileTranslation フィールドに設定する．独自の変換の提供を選択する場合は，NavReplyRecord 構造体のtranslationNeeded フィールドにtrue を設定しておく必要がある．

ファイルを最初に保存するとき，アプリケーションではユーザーがその種類を閉じるまでNavCompleteSave 関数の呼び出しを待つようにしなくてはならない．こうすることで，ユーザーがファイルを使って作業をしている間，アプリケーションはネイティブ・フォーマットでファイルを保存することができる．ファイルのコピーを保存するときは，NavPutFile 関数から戻り値が返された直後にNavCompleteSave 関数を呼び出すようにすればよい．

デフォルトの設定で，NavPutFile 関数は変換されたファイルをオリジナル・ファイルのコピーとして保存する．オリジナルを変換ファイルで置き換えるには，NavCompleteSave 関数のhowToTranslate パラメータにkNavTranslateInPlace 定数を渡せばよい．

（c）ファイルのプレビュー表示

ナビゲーションサービスでは，ファイルをオープンするすべてのダイアログボックスにプレビュー領域が用意されている．この領域のオン/オフはユーザーによって切り替えられる．プレビュー領域が表示されていると，有効な‘ pnot ’ リソースを含む任意のファイルのプレビューが自動的に表示される．プレビューの表示を要求するには，NavDialogOptions 型の構造体のdialogOptionFlagsフィールドにkNavAllowPreviews 定数を設定する．また，独自のプレビューを用意し，プレビュー領域にカスタム・コントロールを追加することもできる．

３.３．タイプ・ポップアップ・メニューの拡張

“Open”ダイアログボックスの表示ポップアップメニューと，“保存”ダイアログボックスのフォーマット・ポップアップ・メニューは，タイプ・ポップアップ・メニューと呼ばれる．アプリケーションで，いずれかのタイプ・ポップアップ・メニューに独自のメニュー項目を追加する必要がある場合は，NavMenuItemSpec 型の構造体を使って，追加するそれぞれのメニュー項目を記述しなければならない．この方法により，オープンまたは保存の対象となる特殊なファイルタイプや，ファイルを保存するさまざまな方法を追加できるようになる．メニュー項目を設定するに

は,１ つ, または複数のNavMenuItemSpec構造体へのハンドルをNavDialogOptions 型の構造体のpopupExtension フィールドに追加すればよい．NavMenuItemSpec 構造体のmenuItemName フィールドに値を指定する必要はないが，この値が使用できる場合，ナビゲーションサービスはこの値を検索キーとして使用する．このフィールドへの値の指定を選択しない場合は，フィールドに空白文字列を設定しなければならない．　選択されたポップアップメニュー項目を処理してテストするには，ナビゲーションサービスがアプリケーションのイベント処理関数を呼び出すときに，kNavCBPopupMenuSelect メッセージ定数に応答すればよい.

callBackParms パラメータを使って，選択されたメニュー項目に関する情報を取得する．NavCBRec 型の構造体のparam フィールドは，メニュー項目を記述するNavMenuItemSpec 型の構造体へのポインタである．たとえば，type およびcreator フィールドを比較することで，アプリケーションは特定のメニュー項目に応答することができる.

アプリケーションがタイプ・ポップアップ・メニューを拡張しようとしているのに，イベント処理関数を準備しないと，ナビゲーションサービスはparamErr (-50) を返す．フィルタ処理を必要とするメニュー項目を追加する場合は，フィルタ関数をインプリメントする必要がある.

３.４．カスタム・コントロールの追加

ナビゲーションサービスAPI は，標準ファイル・パッケージを使用する場合なら，インターフェースのカスタマイズが要求されるような，最も一般的な状況を容易に処理できるように設計されている．すべての機能を検討した上で，それでもなおナビゲーション・サービス・ダイアログ・ボックスにカスタム・コントロールを用意する必要がある場合は，次の処理を行わなければならない.

i. イベント処理関数をインプリメントし，“Open”または“保存”ダイアログボックスが表示されているときにナビゲーションサービスとの間でデータをやり取りできるようにする.

ii. kNavCBCustomize 定数に応答する. この定数は, NavCBRec 型の構造体のparam フィールドから取得できる．NavCBRec構造体のcustomRectフィールドは，ウ

インドウのローカル座標を使って矩形の領域を定義していて，左上隅の座標はカスタマイズ矩形のアンカー・ポイントを定義している．これは，ナビゲーションサービスがカスタム・ダイアログ・ボックス項目を追加するためにアプリケーションに提供する領域である．アプリケーションは，必要とされるカスタマイズ矩形のサイズを指定する値を渡すことで応答する．アプリケーションが応答を行い，イベント処理関数を終了すると，ナビゲーションサービスはcustomRect フィールドの内容を検査して，要求されたサイズによって生成されるダイアログ・ウインドウが画面上に収まりきるかどうかを判断する．結果的に生成されるウインドウのサイズが大きすぎると，ナビゲーションサービスは，矩形の領域を画面に収まりきる最大のサイズに設定し，kNavCBCustomize 定数を使って再度アプリケーションに通知を行うことで応答する．アプリケーションでは，ナビゲーションサービスが適切な矩形の値を指定するまで，customRect フィールドを検査して異なるサイズを要求することで，「ネゴシエーション」を続けることができる．カスタム領域の最小サイズは，幅400 ピクセル×高さ40 ピクセルある．

iii. カスタマイズ矩形が確立されると，アプリケーションはNavCBRec 構造体のparamフィールドに含まれるkNavCBStart定数をチェックしなければならない．この定数は，ナビゲーションサービスがダイアログボックスを表示しているかどうかを識別する．この定数を取得した後，独自のインターフェース要素をカスタマイズ矩形に追加できる．この処理を最も簡単に行うには，‘DITL’リソースを用意し（ローカル座標で，カスタマイズ矩形のアンカー・ポイントが基準になる），NavCustomControl 関数の selector パラメータにkNavCtlAddControlList定数を渡す．

iv. ユーザーがダイアログボックスを閉じると，ナビゲーションサービスはNavCBRec 構造体のparam フィールドにkNavCBTerminate 定数を指定する．この定数を必ずチェックしなければならない．この定数はコントロールまたはリソースを廃棄するシグナルとなる．

３.５．各種イベントへの具体的な対応方法

Windows や UNIX においては，‘.jpg’‘.gif’などのファイルの拡張子によってそのファイルのファイル形式を判別している.それに対し,従来Mac OSではFileTypeと呼ばれる情報を Finder が管理していて，その情報をもとにファイル形式を判別している．しかし，Mac OS Xからはその両方の方式を扱うようになった.

本研究では，その整合性を損なうことなく，ユーザーの意のままに画像ファイルを出力できるよう改良を重ねた.

＜ FileTypeの例 ＞

・ SGI	‘.SGI’
・ Photoshop	‘8BPS’
・ BMP	‘BMPf’
・ JPEG	‘JPEG’
・ PICT	‘PICT’
・ PNG	‘PNGf’
・ MacPaint	‘PNTG’
・ TIFF	‘TIF ’
・ TGA	‘TPIC’
・ JP2	‘jp2 ’
・ QuickTime Image	‘qtif’

対応するべき事項を次にあげる.

- ポップアップメニューが変更された場合，EditFileName の拡張子をそれに対応したデフォルトの拡張子に自動的に変更させる．ただし，デフォルトではないが正しい拡張子が入力されていた場合は，変更しない.

- EditFileName に拡張子が入力された場合，ポップアップメニューをそれに対応したメニューに変更させる.

これらを行うための方法として，以下，各種イベントごとの処理方法について述べる.

（a）ダイアログボックス設定後，最初に起こるイベント

・kNavCBStart

デフォルトのファイル名およびコンポーネントをダイアログボックスに設定し，ポップアップメニューをそのコンポーネントの MenuItem に初期設定する．この作業を行わないと，デフォルトのフォーマットを設定しても，ポップアップメニューは GraphicsImporter で最初にヒットしたコンポーネント（現在でいうと SGI）に設定されてしまう．

続いて，(b)，(c) についてそれぞれ説明を行うが，共通して行う処理として先述する．

まず，ファイル名編集フィールドに入力されたファイル名がコンパティブルかどうか判定する．本研究では，この処理にあたるものとして，JudgeCompatibleOrNot() という関数を作成し対応させた．以下，その処理方法について述べる．

最初にファイル名編集フィールドに入力されたファイル名の拡張子に，大文字，小文字が混合して入力されていないかどうか判定する．混合して入力されていた場合，その拡張子は誤ったものとみなす．この処理は QTGetFileNameExtension() 関数の戻り値に関係するものである．ファイル名から拡張子を得る際，QTGetFileNameExtension()関数を用いるのだが，この関数で得られる拡張子はその入力に関係なくすべて大文字で値が返される．例えば入力が‘PicT’や‘pIcT’であったとしても，返される値は‘PICT’であるため，正誤判断ができない．従って事前にこの処理を行う必要がある．

さらに具体的な処理方法について述べる．

まず，大文字，小文字に相当する Flag を用意し，それぞれを False に設定しておく．次に EditFileName の最後尾の文字列から読み込み，‘.’があればそれ以下を拡張子としてみなす．その拡張子とみなした文字列を再度読み込み，大文字，小文字の入力が見つかった場合は，それぞれの Flag を True にする．それぞれの Flag が True であれば，それは誤りとみなし False を返す．どちらか一方が true であれば，それは正しいものとみなす．

次に，判定手段として FileType が必要となる．ここで，FileType を得る方法は大きく３つある．

１． Graphics Exporters から得たフォーマットリストから得る方法．

２． EditFileName から得る方法

３．ポップアップメニューによって選択された MenuItem から得る方法.

これらの具体的な処理方法いついては“３.６．FileType を得る方法”に記述する.

（b）ダイアログボックスで，ポップアップメニューを変更したとき

・kNavCBPopupMenuSelect

まず，２で得た FileType と１で得た FileType を照合し，同値であれば何もしない．同値でなければ次の処理を行う．３で得た FileType からそれに対応したコンポーネントを得る．次にそのコンポーネントを使用してデフォルトの拡張子を得る．そしてそれを現在入力されているファイル名の拡張子として変更させる.

（c）ファイル名に拡張子が入力された場合

・kNavCBEvent（case keyUp）

ポップアップメニューが変更されたときと同様に，まず，２で得た FileType と１で得た FileType とを照合し，同値のものが見つかればそれに相当するメニューアイテムなどをダイアログに設定する．同値のものが見つからない，もしくは，２では FileType が得られなかっなら，ポップアップメニューは変更せず，現在のフォーマットのままにする.

３.６．FileType を得る方法

“３.５”で述べた FileType を得る方法―以下３つについて具体的に説明する.

１．Graphics Exporters から得たフォーマットリストから得る方法.

２．EditFileName から得る方法

３．ポップアップメニューによって選択された MenuItem から得る方法.

１．Graphics Exporters から得たコンポーネントリストから得る方法.

＜ 関数例 ＞

```
void findGraphicsExporterComponents()
```

```
{
        Component c = 0;
        ComponentDescription      cd;
        ComponentDescription      cdesc;
        cd.componentType = GraphicsExporterComponentType;
        cd.componentSubType = 0;
        cd.componentManufacturer = 0;
        cd.componentFlags = 0;
        cd.componentFlagsMask = graphicsExporterIsBaseExporter;

        while((c = FindNextComponent(c, &cd)) != 0) {
                OSErr    err = GetComponentInfo(c, &cdesc,
                                                NULL, NULL, NULL);
                ...                 // add component c to the list.
        }
}
```

ComponentDescription 型の構造体の使用により，componentType を GraphicsExporterComponentType と定義する．FindNextComponent 関数の使　用により，GraphicsExporter の Component を全て読み込む．このリストを利用して，それぞれのコンポーネントの FileType を得ることができる．

2．EditFileName から得る方法

まず，QTGetFileNameExtension()関数よりファイル名から拡張子を得る．そしてその拡張子をすべて小文字に変換し，デフォルトの拡張子以外の拡張子との照合を行い，相当するものがあるかを判定する．あればその拡張子の FileType を返す．なければ次に，それを 1 で得たそれぞれの FileType と照合し，相当するものがあるかどうかを判定する．同様に存在すればその FileType を返す．

3．ポップアップメニューによって選択された MenuItem から得る方法．

＜ イベント処理関数内での例 ＞

```
myNavEventProc( NavEventCallbackMessage          inSelector,
                NavCBRecPtr                      ioParams,
                NavCallBackUserData              ioUserData)
{
     NavEventDataInfo  info = ioParams->eventData.eventDataParms;
     NavMenuItemSpecPtr         p = (NavMenuItemSpecPtr)(info.param);
}
```

NavMenuItemSpec 型の構造体 p を宣言し，上述の例のように定義すれば，
p->menuType から，現在ポップアップメニューで選択されているメニューアイテムの FileType を得ることができる．

参考文献

杉浦秀子：DletaViewer プロジェクト

参考 URL

http://developer.apple.com/quicktime/

http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmImporter.htm

http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmExporter.htm

http://developer.apple.com/techpubs/macosx/Carbon/Files/NavigationServices/Prog_Navigation_Services/

謝辞

卒業研究をするにあたり，本当にたくさんの方々にお世話になりました．指導教官である和田昌昭教授，そして和田研究室で共にがんばってきた伊佐友江さん，藤原左知子さん，細井絵里子さん，本当にありがとうございました．そして最後に，こんな私を大学まで通わせてくれた両親に御礼申し上げます．