

i アプリコンテンツ開発ガイド for 504i
～ i アプリオプション・i アプリ拡張編 ～
第 2.0 版

平成 14 年 11 月 1 日
株式会社 N T T ドコモ



本製品または文書は著作権法により保護されており、その使用、複製、再頒布および逆コンパイルを制限するライセンスのもとにおいて頒布されます。NTT ドコモ（その他に許諾者がある場合は当該許諾者も含めて）の書面による事前の許可なく、本製品および関連する文書のいかなる部分も、いかなる方法によっても複製することが禁じられます。フォントを含む第三者のソフトウェアは、著作権法により保護されており、その提供者からライセンスを受けているものです。

Sun、Sun Microsystems、Java、J2ME および J2SE は、米国およびその他の国における米国 Sun Microsystems, Inc. の商標または登録商標です。サンのロゴマークは、米国 Sun Microsystems, Inc. の登録商標です。

「i モード」、「i アプリ／アイアプリ」、「i-appli」ロゴ、「i メロディ／アイメロディ」は NTT ドコモの商標です。

その他、掲載されている会社名、製品名、サービス名は各社の商標または登録商標です。なお、本書では、コピーライト及び TM 表記はしておりません。

文書は現状有姿で提供されており、その商品性、特定目的への適合性、第三者の権利の非侵害を含み、明示的または黙示のいかなる条件、表明および保証も行われず、また提供されません。但し、かかる保証の否定が、適用される法律の下で無効である場合は、この限りではありません。

Copyright 2000-2002 NTTDoCoMo, Inc. 2-11-1 Nagata-cho, Chiyoda-ku, Tokyo, 100-6150, Japan and Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California, 94303, U.S.A. All rights reserved.

本書は、ムーバ 504i シリーズおよび 2002 年秋発売の FOMA 携帯電話に搭載される Java アプリケーション実行環境（本書中では i アプリ実行環境と呼びます）をターゲットとして記述されています。本書内ではデジタル・ムーバ 503i シリーズの i アプリ実行環境を DoJa-1.0 プロファイル、ムーバ 504i シリーズなどの i アプリ実行環境を DoJa-2.0 プロファイルと呼びます（DoJa-2.1 プロファイルと記載されている場合、特に 2002 年秋発売の FOMA 携帯電話に搭載されている i アプリ実行環境を指します）。これらは便宜上の呼称であり、いかなる製品の商標名などを指し示すものでもありません。

本書の内容は、以前のプロファイルから存在している機能であっても、以前のプロファイルにのみ対応した携帯電話には適用されませんのでご注意ください。本書で解説するプロファイルには、既存機能に関する未定義の仕様の明確化も含まれています。

目 次

第1章 はじめに	4
第2章 2D グラフィックス描画	7
2.1 イメージマップ	7
2.2 ズーム座標系	8
2.3 スプライト	9
2.4 ピクセル操作	9
2.5 透明色付きイメージ	9
2.6 パレット付きイメージ	10
2.7 イメージ回転・反転・拡大縮小表示	10
2.8 描画クリッピング	11
2.9 ラスタオペレーション	11
2.10 その他	12
第3章 2D アニメーション再生	13
第4章 3D グラフィックス描画	14
4.1 高レベル 3D グラフィックス描画	14
4.2 低レベル 3D グラフィックス描画	15
第5章 サウンド再生	18
5.1 MFi/Phrase	18
5.2 一時停止・再開	19
5.3 エフェクト	19
5.4 再生時間制御	20
5.5 再生同期イベント	20
第6章 リソース制御	21
第7章 サブディスプレイ描画	24
第8章 端末ネイティブのデータストレージ アクセス	25
第9章 カメラ制御	27
9.1 カメラの制御	27
9.2 撮影画像の携帯電話間での交換	28
第10章 ポインティングデバイス	31

第1章

はじめに

i アプリ実行環境は、次の4つのAPIレイヤから構成されています。

- J2ME/CLDCで規定されたAPI
- i アプリ基本API
- i アプリオプションAPI
- i アプリ拡張API

本書では、i アプリオプションAPIおよびi アプリ拡張APIとして採用されているものについての概要を解説します。

なお、J2ME/CLDC API については『Java 2 Platform, Micro Edition (J2ME) Connected Limited Device Configuration (CLDC)』（<http://java.sun.com/products/cldc/>）を、i アプリ実行環境のAPIレイヤ構成およびi アプリ基本APIの詳細については『i アプリコンテンツ開発ガイド for 504i 〜詳細編〜』を参照してください。

i アプリオプションAPIおよびi アプリ拡張APIには、以下のような機能カテゴリに属するAPIが含まれています。

大分類	小分類	解説
2D グラフィックス描画	イメージマップ	縦 m、横 n 枚の、複数のイメージの組み合わせで構成されるイメージマップを取り扱います。イメージマップ上には任意の矩形領域（ウィンドウ）を設定し、そのウィンドウ内のイメージをグラフィックスコンテキストに描画することができます。
	ズーム座標系	グラフィックスコンテキストに描画を行う際に使用する座標系として、256 倍に拡大された座標系を使用します。
	スプライト	スプライトを、位置属性および可視属性を持つイメージとして取り扱います。この機能では、1つ1つのスプライト実体を表す Sprite クラスと、Sprite オブジェクトを集合管理するための SpriteSet クラスを提供します。
	ピクセル操作	グラフィックスコンテキスト上の特定のピクセル位置の色情報を取得することができます。また特定のピクセル位置に、指定された色で点を描画することができます。
	透明色付きイメージ	透明色を指定することができるイメージです。任意の色を透明色に指定したり、透明色の使用有無を切り替えたりすることができます。
	パレット付きイメージ	色パレットを指定することができるイメージです。パレット付きイメージに設定するパレットの内容を変更することで、イメージの表示色を変更することができます。
	イメージ回転・反転・縮小拡大表示	イメージ表示の際、回転や反転、拡大縮小の効果を加えます。
	描画クリッピング	グラフィックスコンテキストに対して直接描画クリッピング領域を設定する機能を提供します。

	ラストオペレーション	描画元のピクセルの色とそれに対応する描画先ピクセルの色を一定のルールで演算し、その結果得られた色でレンダリングを行います
	その他	2D グラフィックス描画に関する、上記分類に属さない単機能的なメソッドを提供します。
2D アニメーション再生	-----	独立した API ではありませんが、VisualPresenter がサポートするメディアデータを拡張し、スムーズな 2D アニメーションの再生を行います。 なお本機能においては、2D アニメーションデータのデータフォーマットはメーカーにより異なる可能性があります。従って、本機能は i アプリ拡張 API に属します。
3D グラフィックス描画	高レベル 3D グラフィックス描画	3D で表現されたモデルの描画を簡単な API 呼び出しで行うための API を提供します。3D モデルにはポーズ（アクション）を取らせることができます。また、回転や視点移動を行うための演算もサポートしています。 なお本機能においては、3D モデルデータのデータフォーマットはメーカーにより異なる可能性があります。従って、本機能は i アプリ拡張 API に属します。
	低レベル 3D グラフィックス描画	高レベル 3D グラフィックス描画機能と比較して、アプリケーションからより細かな描画制御を行うことができる API を提供します。 なお本機能においては、3D モデルデータのデータフォーマットはメーカーにより異なる可能性があります。従って、本機能は i アプリ拡張 API に属します。
サウンド再生	MFi/Phrase	端末の持つ同時発音可能数の範囲で、複数の i メロディ形式データを並列再生する機能を提供します。
	一時停止・再開	再生中のサウンドを一時停止（pause()）および再生再開（restart()）を行う機能を提供します。
	エフェクト	サウンドデータに設定されている音階やパンポット、テンポなどを一定の範囲で変更して再生する機能を提供します。
	再生時間制御	再生中の AudioPresenter から、現在再生中のサウンドの先頭からの位置を取得する機能を提供します。また、サウンドデータの先頭からではなく、特定の位置からの再生を行う機能を提供します。
	再生同期イベント	特定のチャンネルの特定のキーのノートに対応してイベントを発生させる、再生同期イベント機能を提供します。
リソース制御	音量制御	メロディ、効果音、マイク（イヤホンマイク接続時のみ）の音量を制御します。
	折りたたみ開閉状態取得	折りたたみ型端末やフリップ型端末において、本体やフリップの開閉状態を取得します。
	効果音再生	メロディデータを用意することなく、端末にプリセットされた簡単な効果音を再生することができます。
	ソフトキー背景色制御	ソフトキーラベルの背景色を制御します。
	追加キー制御	テンキー、上下左右決定キー、ソフトキー以外の特種なキーをサポートします。
	パイプレータ制御	パイプレータのオンオフを制御します。
	背景 LCD イルミネーション制御	背面 LCD イルミネーションを備える端末において、その点灯状態を制御します。
	背面ライト制御	背面に i モードマークのライトを備える端末において、その点灯状態を制御します。
	明度・コントラスト制御	ディスプレイやサブディスプレイの明度やコントラストを制御します。
	メモ LED 制御	メモ LED のオンオフを制御します。
サブディスプレイ描画	-----	端末のサブディスプレイに描画する機能を提供します。

端末ネイティブのデータストレージアクセス	-----	端末ネイティブのデータストレージとは、待ち受け画像など端末のネイティブアプリケーションが使用するデータストレージです。この機能は、このデータストレージに i アプリがアクセスする機能を提供します。 ただし i アプリからアクセス可能な領域は、データストレージの中で i アプリ用に用意された一部分だけであり、任意のデータストレージ領域にアクセスすることはできません。
カメラ制御	-----	アプリケーションからカメラ機能を起動します。撮影は、起動されたカメラ機能をユーザーが操作することにより行われます。アプリケーションから撮影（シャッター操作）を行うことはできません。撮影された結果の画像は、アプリケーションからメディアイメージとしてもバイトストリームとしても使用することができます。
ポインティングデバイス	-----	ポインティングデバイスをアプリケーションから使用します。ユーザーはポインティングデバイスを使用して、画面上を自由に移動可能なカーソルを操作することができます。 ポインティングデバイスは、Canvas 上でのみ利用することができます。

以降、各機能について概要を解説します。各機能で使用するメソッドの詳細については、『i アプリコンテンツ開発ガイド for 504i 〜オプション API リファレンス編〜』を参照してください。

なお、これらの API はメソッド単位、フィールド単位でメーカーが実装の有無を決定します。各機種におけるメソッドやフィールドのサポート状況は、別途 NTT ドコモより公開されます。

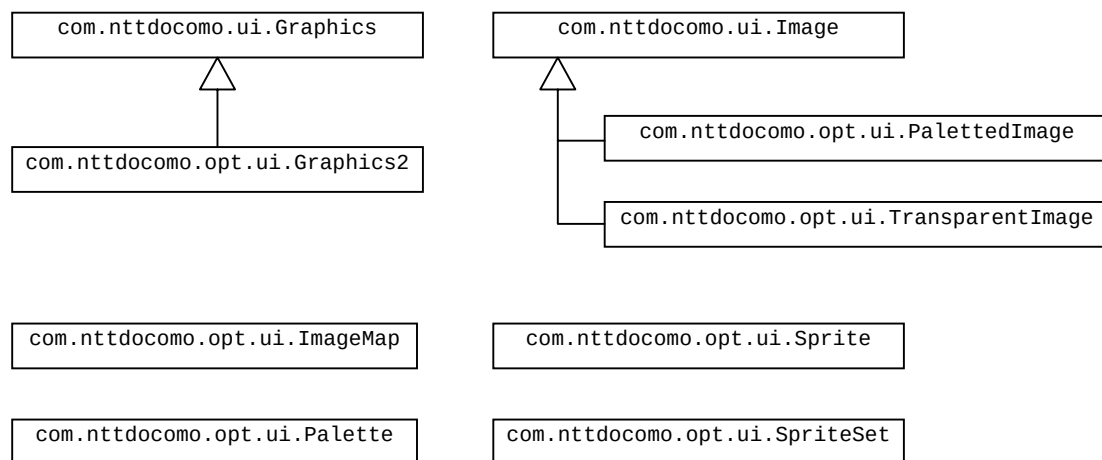
本書で規定される機能には、将来のプロファイルにおいて i アプリ基本 API に移行する可能性のあるものも含まれています。ただし、逆に将来仕様が変更または削除される可能性のあるものも含まれていますのでご注意ください。

第 2 章

2D グラフィックス描画

2D グラフィックス描画機能では、i アプリ基本 API における低レベルグラフィックス描画およびイメージ処理を拡張します。これらの API を使用することで、アプリケーションによる画面の表示に特殊な効果を与えることができます。

2D グラフィックス描画機能を構成するクラス群を以下に示します。



2D グラフィックス描画のいずれかの機能をサポートする携帯電話では、`Canvas.getGraphics()` などによりシステムから取得するグラフィックスコンテキストは `com.nttdocomo.opt.ui.Graphics2` クラス (拡張グラフィックスコンテキスト) のオブジェクトとなっています。

2.1 イメージマップ

イメージマップ (`com.nttdocomo.opt.ui.ImageMap` クラス) は、小さなイメージを縦横に並べて見かけ上大きなイメージを作成する機能を提供します。

イメージマップにイメージを割り当てる単位 (イメージマップ上の升目) をセルと呼びます。イメージマップには、イメージマップを構成するイメージの集合 (配列: 最大 128 個指定可能) と、個々のセルにどのイメージを割り当てるかを示すマップデータを指定します。以下は、6×6 個のセルから構成されるイメージマップにイメージを割り当てる様子を示しています。

<セルへのイメージ割り当て>

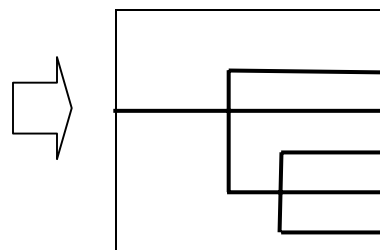
	0		→		5
0	0	0	0	0	0
	0	0	4	1	1
	1	1	3	1	1
↓	0	0	2	4	1
	0	0	5	3	1
5	0	0	0	5	1

※升目内の数字がイメージ配列要素を示します。

<イメージ配列>

0	
1	—
2	
3	+
4	┐
5	└

<イメージマップの内容>



このようにイメージマップは、数種類の小さな画像の組み合わせで表現される大きな画像を最小限のデータ量で取り扱いたい場合に適しています。

イメージマップは `Graphics2.drawImageMap()` メソッドを使用してグラフィックスコンテキスト上に描画します。イメージマップはその全体を描画対象とすることもできますが、イメージマップ上にウィンドウと呼ぶ矩形領域を設定することにより、その領域だけを描画対象とすることもできます。ウィンドウの位置はイメージマップ全体のサイズの範囲内で自由に変更することができるため、ディスプレイ内で大きなイメージマップがスクロールするようなアプリケーションを容易に記述することができます。

2.2 ズーム座標系

`com.nttdocomo.opt.ui.Graphics2` クラスでは描画時に指定する座標位置について、ディスプレイ上のピクセル位置と座標系が 1 対 1 に対応するノーマルモードの他に、座標系が実際のピクセル位置の 256 倍に対応するズームモードをサポートします。

座標系の切り替えは、`Graphics2.setCoordinateMode()` メソッドを使用していきます。このメソッドの引数に `Graphics2.CM_NORMAL` を指定するとノーマルモードに、`Graphics2.CM_ZOOM` を指定するとズームモードになります。

ズームモードでは、実際に描画される位置は描画メソッド呼び出し時に指定された位置の 1/256 のスケールになります。例えばズームモードにおいて

```
drawLine( 2560, 2560, 25600, 25600 );
```

という直線を描画した場合、ディスプレイ上は (10, 10) – (100, 100) の 2 点を結ぶ直線が表示されます。

なお、ズームモードにおいては `Graphics.setOrigin()` で設定されている座標オフセットもズームの対象となります。すなわち、上記直線を描画する際に座標オフセットが

```
setOrigin( 5120, 5120 );
```

と設定されていた場合、ディスプレイ上は (30, 30) – (120, 120) の 2 点を結ぶ直線が表示されます(*1)。

(*1) 詳細には、 $((2560+5120)/256, (2560+5120)/256) - ((25600+5120)/256, (25600+5120)/256)$ を結ぶ直線となります。

2.3 スプライト

2D グラフィックス描画機能では、簡易なスプライト表示機能をサポートしています。スプライト機能は、`com.nttdocomo.opt.ui.Sprite` クラスと `com.nttdocomo.opt.ui.SpriteSet` クラスから構成されます。

`Sprite` クラスは、画面上を移動する 1 つ 1 つのスプライトを表現し、属性としてイメージオブジェクトと表示位置を持ちます。また `Sprite` クラスの持つイメージを実際にディスプレイに表示する際には、以下の操作を行うことができます

- ・ 可視・非可視設定
- ・ 反転表示・回転表示設定
- ・ スプライト上のピクセルと背景となるピクセルとの色の演算（ラスタオペレーション：2.9 項参照）

多くの場合、アプリケーションは 1 つの画面上にスプライトを複数表示します。このため 2D グラフィックス描画機能では、それら複数のスプライトを集合管理するための `SpriteSet` クラスを導入しています。アプリケーションは、同時に使用するスプライトを全てスプライトセットに登録します。画面へのスプライトの表示は、`Graphics2.drawSpriteSet()` メソッドにスプライトセットを指定することで行います。`Sprite` オブジェクトをそのまま描画することはできません。

スプライトセットでは、内部で保持しているスプライト間の衝突判定を行うこともできます。衝突判定は、各々のスプライトの位置およびスプライトに登録されているイメージの矩形のサイズに基づいて行われます。

2.4 ピクセル操作

2D グラフィックス描画機能では、ピクセル操作用に以下の機能をサポートしています。

- ・ グラフィックスコンテキストの特定ピクセル位置の色情報取得(`Graphics2.getPixel()`メソッド)
- ・ 点の描画 (`Graphics2.setPixel()`メソッド)

i アプリ基本 API では点の描画には `Graphics.drawPolyline()` を使用しますが、その場合は描画指示と描画色の指定は別々に行う必要があります。これに対し `Graphics2.setPixel()` メソッドでは、描画指示と同時に描画色を指定することができます。

`Graphics2.getPixel()`、`Graphics2.setPixel()` の両メソッドとも、色情報には色を表す機種固有の整数値 (`Graphics.getColorOfRGB()` メソッドなどで返される値) が使用されます。

2.5 透明色付きイメージ

2D グラフィックス描画機能では、アプリケーションによって自由に透明色を指定できる透明色付きイメージ (`com.nttdocomo.opt.ui.TransparentImage` クラス) をサポートしています。透明色付きイメージはスタティックメソッド

`TransparentImage.createTransparentImage()`を、透明色付きイメージとして取り扱いたいイメージを引数に指定して呼び出すことにより取得します。

i アプリ基本 API の `Image` クラスでは、元となる GIF データが透過 GIF である場合に限りその GIF データに設定されている透過色が透明色として扱われます。それに対し、透過色付きイメージでは透過 GIF を使用した画像でなくても、またイメージ中で使用されているどの色に対しても、動的に透明色の指定を行うことができます。透明色の指定には、色を表す機種固有の整数値 (`Graphics.getColorOfRGB()` メソッドなどで返される値) を使用します。

透明色の指定を有効にするには、`TransparentImage.setTransparentEnabled()` メソッドを使用します。初期状態では、透明色の指定は無効化されています。

`TransparentImage` クラスは i アプリ基本 API の `Image` クラスを継承しています。ただし `Image` クラスで定義されている `getGraphics()` メソッドは `TransparentImage` オブジェクトに対して呼び出すことはできません。アプリケーションがこのメソッドを呼び出した場合は例外が発生します。

2.6 パレット付きイメージ

2D グラフィックス描画機能では、アプリケーションから色パレットを指定することのできるパレット付きイメージ (`com.nttdocomo.opt.ui.PalettedImage` クラスと `com.nttdocomo.opt.ui.Palette` クラス) をサポートしています。

パレット付きイメージはスタティックメソッド

`PalettedImage.createPalettedImage()` を、パレット付きイメージとして取り扱いたい GIF データのバイト配列を引数に指定して呼び出すことにより取得します。このためパレット付きイメージは、メモリ上に作成した GIF データからイメージを取得する目的で使用することもできます。

パレット付きイメージからは、`getPalette()` メソッドおよび `setPalette()` メソッドを使用してパレットの参照や設定を行うことができます。パレットの色情報には、色を表す機種固有の整数値 (`Graphics.getColorOfRGB()` メソッドなどで返される値) が使用されます。

`PalettedImage` クラスは i アプリ基本 API の `Image` クラスを継承しています。ただし `Image` クラスで定義されている `getGraphics()` メソッドは `PalettedImage` オブジェクトに対して呼び出すことはできません。アプリケーションがこのメソッドを呼び出した場合は例外が発生します。

2.7 イメージ回転・反転・拡大縮小表示

`Graphics2` クラスでは、イメージ描画時にそのイメージを回転、反転、拡大縮小表示するための機能をサポートしています。

回転表示・反転表示

イメージの回転表示および反転表示は、`Graphics2.setFlipMode()` メソッドに反転方法を指定することで行います。このメソッドを呼び出すと、それ以降のイメージ描画時に指定された反転方法が適用されます。

指定可能な反転方法には、以下のものがあります。

- `Graphics2.FLIP_HORIZONTAL` : 横方向に鏡像反転することを示します。
- `Graphics2.FLIP_NONE` : 反転表示しないことを示します。
- `Graphics2.FLIP_ROTATE` : 縦横方向に鏡像反転（180 度回転）することを示します。
- `Graphics2.FLIP_VERTICAL` : 縦方向に鏡像反転することを示します。

拡大縮小表示

イメージの拡大縮小表示は、イメージの描画指示に `Graphics2.drawImage()` メソッドを使用することで行います。このメソッドの引数には、イメージとグラフィックスコンテキスト上の描画位置の他に以下を設定します。

- イメージ描画先グラフィックスコンテキスト上の矩形領域の幅と高さ
- 描画元イメージのうち、描画対象とする矩形領域の左上座標、および幅と高さ

これにより、描画元の矩形領域内の内容が、描画先の矩形領域にちょうど収まるように拡大または縮小されて表示されます。

2.8 描画クリッピング

i アプリ基本 API では、`Canvas.repaint()` メソッドが描画クリッピング機能をサポートしています。これに対し 2D グラフィックス描画機能では、グラフィックスコンテキスト（`Graphics` クラス）のレベルでの描画クリッピング機能をサポートします。

グラフィックスコンテキストのレベルで描画クリッピングを行えるため、描画メソッド（`Canvas.paint()`）の内部で自由にクリッピング領域を設定したり、イメージから取得したグラフィックスコンテキストへの描画でも描画クリッピング機能を利用することができます。

2D グラフィックス描画機能における描画クリッピングは以下のメソッドから構成されます。

- `Graphics.clearClip()`
現在設定されているクリッピング領域を解除します。
- `Graphics.clipRect()`
引数で指定された矩形領域と現在設定されているクリッピング領域が重なり合っている部分を、新たなクリッピング領域として設定します。
- `Graphics.setClip()`
引数で指定された矩形領域を新たなクリッピング領域として設定します。

2.9 ラスタオペレーション

`Graphics2` クラスでは、描画元のピクセルの色とそれに対応する描画先ピクセルの色を一定のルールで演算し、その結果得られた色でレンダリングを行う機能（ラスタオペレーション）をサポートしています。

レンダリング時にどのような色演算を使用するかは、`Graphics2.setRenderMode()` メソッドで指定します。このメソッドの引数には演算方法、描画元の色への乗数、描画先の色への乗数を指定します。乗数は 1.0 を 256 分割した単位（255 を設定すると 1.0）にて、0～255 の範囲で設定します。以下に、サポートされる演算方法とその内容を解説

します。以下の説明において、src は描画元のピクセル色の値、dst は描画先のピクセル色の値、srcRatio は描画元の色への乗数、dstRatio は描画先の色への乗数を意味します。

- **Graphics2.OP_ADD (加算)**

描画元のピクセル色の値と描画先のピクセル色の値を加算します。計算式は以下の通りです。

$$\text{dst} = \text{src} * \text{srcRatio} + \text{dst} * \text{dstRatio}$$

- **Graphics2.OP_REPL (置き換え)**

描画元のピクセル色の値で、描画先のピクセル色を置き換えます。計算式は以下の通りです。

$$\text{dst} = \text{src} * \text{srcRatio}$$

srcRatio が 1 の場合、ラスタオペレーションを使用しないレンダリングと同じ意味になります。

- **Graphics2.OP_SUB (減算)**

描画先のピクセル色の値から描画元のピクセル色の値を減算します。計算式は以下の通りです。

$$\text{dst} = \text{dst} * \text{dstRatio} - \text{src} * \text{srcRatio}$$

いずれにおいても、src、dst の値には色を表す機種固有の整数値

(`Graphics.getColorOfRGB()` メソッドなどで返される値) が使用されます。

2.10 その他

Graphics クラスおよび Graphics2 クラスでは、前項までに解説した分類以外にも、2D グラフィックス描画に関する単機能的なメソッドをいくつか提供しています。以下、各機能について解説します。

描画内容のコピー (`Graphics.copyArea()`)

引数で指定されたグラフィックスコンテキスト上の矩形領域内の描画内容を、指定された距離だけ離れた位置にコピーして描画します。

アニメーション GIF 内の特定フレームのイメージ描画 (`Graphics2.drawNthImage()`)

引数で指定されたアニメーション GIF のメディアイメージ内の、特定のフレームのイメージを指定位置に描画します。

数値の描画 (`Graphics2.drawNumber()`)

引数で指定された int 型の数値を、指定された桁数が収まる領域に右詰めで描画します。

グラフィックスコンテキストからのイメージ生成 (`Graphics2.getImage()`)

引数で指定されたグラフィックスコンテキスト上の矩形領域の内容が描画されている、新たなイメージを生成します。

2 色の中間色の取得 (`Graphics2.getIntermediateColor()`)

引数で指定された 2 つの色の中間色を取得します。指定された一方の色への近さを 0~255 の範囲で指定することもできます。なお色の情報には、色を表す機種固有の整数値 (`Graphics.getColorOfRGB()` メソッドなどで返される値) が使用されます。

第3章

2D アニメーション再生

i アプリ基本 API では動画再生機能として、VisualPresenter によるアニメーション GIF の再生をサポートしています。アニメーション GIF は、その作成手段も広く普及しておりコンテンツの流通量も豊富である一方、容量、再生品質、再生時間の面で最新の 2D アニメーション技術と比較して劣っています。

2D アニメーション再生機能は、最新の 2D アニメーション技術を活用した動画コンテンツを再生する手段を提供します。

2D アニメーション再生のための特別なクラスやメソッドはありません。アニメーション GIF の再生と同様、2D アニメーションの再生には VisualPresenter を使用します。

なお、DoJa-2.0 プロファイルでは 2D アニメーションデータのフォーマットは規定していません。このため使用する 2D アニメーションデータは本機能を採用するメーカーにより異なる可能性があり、本機能は i アプリ拡張 API に位置付けられます。

メーカーの採用している 2D アニメーションデータのフォーマットやコンテンツの作成方法は、別途メーカーより公開されます。

第 4 章

3D グラフィックス描画

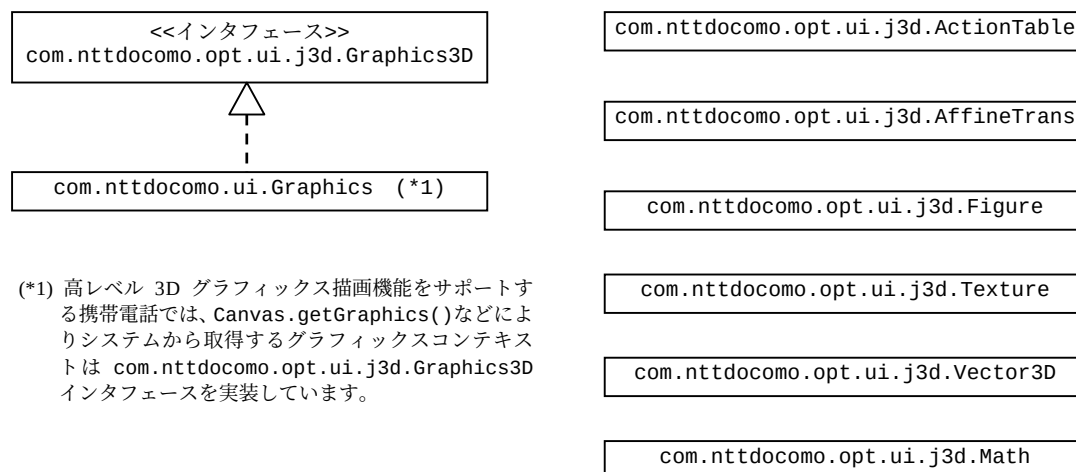
3D グラフィックス描画機能では、外部から取り込んだ 3D モデルデータを操作、表示する機能を提供します。3D グラフィックス描画機能は高レベル API と低レベル API の 2 つから構成されており、それぞれで独立した API セットが定義されています。高レベル 3D グラフィックス描画機能と低レベルグラフィックス描画機能を混在して使用することはできません。

以下、両 API について解説します。

4.1 高レベル 3D グラフィックス描画

高レベル 3D グラフィックス描画機能では、比較的簡単な API 呼び出しで 3D モデルの描画を行うための API セットを提供します。

高レベル 3D グラフィックス描画機能を構成するクラス群を以下に示します。



各クラス、インタフェースの機能概要は以下の通りです。

クラス・インタフェース	機能概要
Graphics3D	高レベル 3D グラフィックス描画機能をサポートするグラフィックスコンテキストが備えるべきメソッドを定義するインタフェースです。
ActionTable	3D モデルのアクション（アニメーションデータ）を保持するクラスです。アニメーションデータは市販の 3D オーサリングツールなどを利用して作成します。 ある特定の瞬間に 3D モデルが取るべきポーズをフレームと呼び、アクションは複数のフレームのシーケンスとして構成されます。ActionTable には複数のアクションを保持することができます。後述の Figure オブジェクトにフレームを設定することで、3D モデルにポーズを取らせることができます。
AffineTrans	3 次元アフィン変換用の行列を扱うクラスです。3D モデルの回転や視点の移動には、このクラスのオブジェクトを使用します。
Figure	3D モデルのデータを保持するクラスです。3D モデルを描画するには、Graphics3D.drawFigure() メソッドを使用します。
Texture	テクスチャのデータを保持するクラスです。テクスチャには、3D モデルのポリゴンに設定するモデルマッピングテクスチャと、環境マッピング使用時に環境球に設定する環境マッピングテクスチャの 2 種類があります。本クラスのオブジェクトを生成する際に、どちらの用途のテクスチャであるかを指定します。
Vector3D	3 次元の固定小数点数ベクトルを表すクラスです。内積・外積計算、正規化といったユーティリティメソッドを備えています。
Math	高レベル 3D グラフィックス描画機能で使用する単位系に適合した数値演算ユーティリティです。なお、このクラスで取り扱える数値の精度はメーカーにより異なる場合があります。

CLDC をベースとしている i アプリ実行環境では、3D 処理で多用される浮動小数点演算を使用することができません。このため高レベル 3D グラフィックス描画機能で使用するクラス・メソッドでは、CLDC で利用可能な整数型を使用して擬似的に小数点以下の値を扱うための独自の単位系を導入しています。導入されている単位系の詳細は API リファレンスを参照してください。

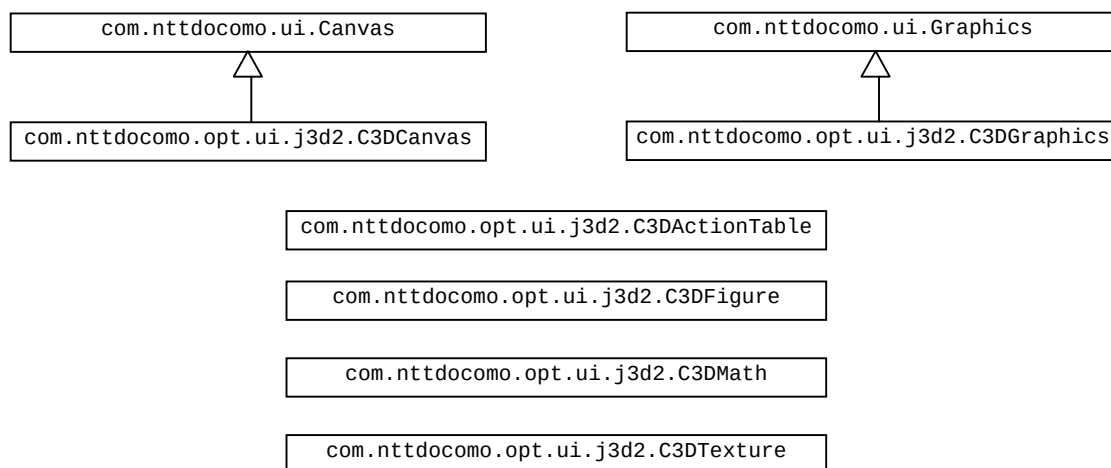
なお、DoJa-2.0 プロファイルでは 3D グラフィックスデータのフォーマットは規定していません。このため使用する 3D グラフィックスデータは本機能を採用するメーカーにより異なる可能性があり、本機能は i アプリ拡張 API に位置付けられます。

メーカーの採用している 3D グラフィックスデータのフォーマットやコンテンツの作成方法は、別途メーカーより公開されます。

4.2 低レベル 3D グラフィックス描画

低レベル 3D グラフィックス描画機能では、高レベル 3D 描画機能と比較して、アプリケーションからより細かな描画制御を行うための API セットを提供します。

低レベル 3D グラフィックス描画機能を構成するクラス群を以下に示します。



各クラス、インタフェースの機能概要は以下の通りです。

クラス・インタフェース	機能概要
C3DCanvas	低レベル 3D グラフィックス描画機能を使用するための表示面です。低レベル 3D グラフィックス描画機能を使用する際には、このクラスのインスタンスを作成して <code>Display.setCurrent()</code> メソッドでディスプレイに表示します。アプリケーションは、C3DCanvas オブジェクトを 1 つだけ作成することができます。また、C3DCanvas と他の Canvas オブジェクトを同時に保持することはできません。 C3DCanvas オブジェクトからは、<code>getC3DGraphics()</code> メソッドを使用して C3DGraphics オブジェクトを取得することができます。Canvas クラスから継承した <code>getGraphics()</code> メソッドは、本クラスでは使用することはできません。
C3DGraphics	低レベル 3D グラフィックス描画機能をサポートするグラフィックスコンテキストです。このグラフィックスコンテキストは 2D グラフィックス描画と 3D グラフィックス描画の両方をサポートしています。<code>set3DMode()</code> メソッドで 2D 描画モードが指定されている場合は Graphics クラスから継承した 2D 描画用メソッドが、3D 描画モードが指定されている場合はこのクラスで追加定義された 3D 描画用メソッドが有効になります。 なお、通常の Graphics では画面左上が原点ですが、C3DGraphics では画面左下が原点となります。
C3DActionTable	3D モデルのアクション（アニメーションデータ）を保持するクラスです。アニメーションデータは市販の 3D オーサリングツールなどを利用して作成します。 C3DActionTable では、1 つのオブジェクトで複数のアクションを保持できる点は j3d パッケージの ActionTable と同様ですが、個々のアクションは複数のフレームではなく時間軸に沿った連続データとして表現されています。従って、このクラスで特定のポーズを取得するには、アクション内の経過時間を指定します。
C3DFigure	3D モデルのデータを保持するクラスです。3D モデルを描画するには、<code>C3DGraphics.callList()</code> メソッドを使用します。
C3DMath	低レベル 3D グラフィックス描画機能で使用する単位系に適合した数値演算ユーティリティです。なお、このクラスで取り扱える数値の精度はメーカにより異なる場合があります。
C3DTexture	テクスチャのデータを保持するクラスです。

CLDC をベースとしている i アプリ実行環境では、3D 処理で多用される浮動小数点演算を使用することができません。このため低レベル 3D グラフィックス描画機能で利用されるクラス・メソッドでは、CLDC で利用可能な整数型を使用して擬似的に小数点以下の

値を扱うための独自の単位系を導入しています。導入されている単位系の詳細は API リファレンスを参照してください。

なお、DoJa-2.0 プロファイルでは 3D グラフィックスデータのフォーマットは規定していません。このため使用する 3D グラフィックスデータは本機能を採用するメーカーにより異なる可能性があり、本機能は i アプリ拡張 API に位置付けられます。

メーカーの採用している 3D グラフィックスデータのフォーマットやコンテンツの作成方法は、別途メーカーより公開されます。

第5章

サウンド再生

サウンド再生機能では、i アプリ基本 API の `AudioPresenter` の拡張機能を提供します。サウンド再生機能は、`com.nttdocomo.ui.AudioPresenter` に定義される拡張メソッドと、`com.nttdocomo.opt.ui.AudioPresenter2` に定義される拡張属性から構成されます。

サウンド再生機能で拡張される機能には、以下のようなものがあります。

- ・ MFi/Phrase
- ・ 一時停止・再開
- ・ エフェクト
- ・ 再生時間制御
- ・ 再生同期イベント

以下、各機能の概略を解説します。

5.1 MFi/Phrase

通常、`AudioPresenter` では同時には1つのサウンドデータしか再生することはできません。これに対し MFi/Phrase では、携帯電話の持つ音源の最大同時発音数（チャンネル数）を複数の仮想的なトラックに分割し、チャンネル数の制限の範囲内で複数のサウンドデータを同時に再生する機能を提供します。DoJa-2.0 プロファイルでは、使用可能なトラック数は4つまでとなります。

MFi/Phrase は以下の手順で使します。

- 1) `AudioPresenter.getAudioPresenter()` メソッドを使用して、複数の `AudioPresenter` オブジェクトを取得します。
- 2) 各々の `AudioPresenter` に対し、`setAttribute()` メソッドでトラック番号を重複しないように割り当てます。`setAttribute()` メソッドには、属性として `AudioPresenter2.TRACK_ASSIGNMENT` を、値としてトラック番号（0～3）を指定します。
- 3) 各々の `AudioPresenter` に対し、`MediaSound` オブジェクトを設定します。`MediaSound` の元となるサウンドデータに MFi/Phrase のための特殊な設定を行う必要はありませんが、同時発音数の合計がチャンネル数を超えることはできません。
- 4) 各々の `AudioPresenter` に対し、`play()` メソッドで再生を指示します。

MFfi/Phrase のトラック割り当てが行われた **AudioPresenter** の再生開始は、MFfi/Phraes のトラック割り当てが行われていない全ての **AudioPresenter** が停止した状態でなければ行うことはできません。また、MFfi/Phraes のトラック割り当てが行われていない **AudioPresenter** の再生は、MFfi/Phraes のトラック割り当てが行われている全ての **AudioPresenter** が停止した状態でなければ行うことはできません。

5.2 一時停止・再開

i アプリ基本 API では、**AudioPresenter** は再生の開始(**play()**)および停止(**stop()**)のみサポートしています。サウンド再生の拡張では、これに加え一時停止 (**pause()**) および再開 (**restart()**) をサポートしています。

一時停止・再開をサポートする携帯電話では、メディアリスナーに対して一時停止時に **AudioPresenter.AUDIO_PAUSED** イベントが、再開時に **AudioPresenter.AUDIO_RESTARTED** イベントが通知されます。

5.3 エフェクト

サウンド再生の拡張では、**AudioPresenter2** の属性としてキーシフトやテンポ、ボリュームなどが定義されており、アプリケーションからこれらの属性を制御することができます。この機能を使用することで、再生するサウンドに一定の効果を掛けることができます。

AudioPresenter2 に定義された属性は、**AudioPresenter.setAttribute()** メソッドに指定して使用します。

以下に **AudioPresenter2** で定義されているエフェクト機能の属性、および指定可能な属性値を示します。

属性	意味・取り得る属性値
KEY_SHIFT	メディア再生時の音階の変化を示す属性です。属性値には、どの程度音階を変化させるかを示す数値を指定します。属性値に <i>n</i> を指定した場合、元のサウンドデータから (<i>n</i> ×半音階) だけ音階を増減させた上で再生を行います。携帯電話がサポートしている変動量を超えた値が設定された場合、携帯電話がサポートする最大または最小の値に補正されて設定されます。
LOOP	ループを示す属性です。属性値には、ループ回数を設定します。
PANPOT	パンポット（ステレオ効果における音の位置）を示す属性です。属性値には 0～127 までの数値を指定します。パンポットのデフォルト値は、64（中央）です。
TEMPO	テンポを示す属性です。属性値には、テンポを変化させる割合（単位パーセント）を指定します。携帯電話がサポートしている変動量を超えた値が設定された場合、携帯電話がサポートする最大または最小の値に補正されて設定されます。 この属性を設定することにより AudioPresenter.getCurrentTime() の結果が影響を受けるか否かはメーカーに依存します。
VOLUME	ボリュームを示す属性です。属性値には、0～127 までの数値を指定します。ボリュームのデフォルト値は 100 です。

LOOP 属性以外のものは、**AudioPresenter** が再生を行っている最中に属性値を変更すると、その時点から再生に効果を与えます。**AudioPresenter** が再生を行っている最中に LOOP 属性を設定しても無視されます。

5.4 再生時間制御

サウンド再生の拡張では、サウンドの特定の位置から再生を開始する機能を提供しています。

サウンドの特定の位置から再生を行うには、`AudioPresenter.play(int time)`メソッドを使用します。このメソッドの引数には、再生を開始する位置を、サウンドデータの先頭からのミリ秒数として設定します。

5.5 再生同期イベント

サウンド再生の拡張では、サウンド再生における同期イベント発生機能をサポートします。同期イベントとは、サウンドの再生進行に同期してアプリケーション（メディアリスナー）に通知されるメディアイベントです。`AudioPresenter.setAttribute()`メソッドを使用して、同期イベント発生可否属性（`AudioPresenter.SYNC_MODE`）に対しオン（`AudioPresenter.ATTR_SYNC_ON`）またはオフ

（`AudioPresenter.ATTR_SYNC_OFF`）を指定することでイベントの発生有無を制御することができます。

同期イベント機能を利用する場合、サウンドデータを構成するうちの任意の1チャンネルをイベント発生源として指定します。指定されたチャンネルの指定されたノートメッセージに連動して、メディアリスナーに同期イベントが通知されます。

なお、イベント発生源として指定されたチャンネルも発音の対象となります。発音が不要な場合、サウンドデータ作成時においてそのチャンネルに対し消音指定を行うようにしてください。

第 6 章

リソース制御

i アプリ基本 API では、以下のリソースが `com.nttdocomo.ui.PhoneSystem` クラスに定義されています。

- ・ バックライト
- ・ メール受信状態
- ・ メッセージ受信状態

リソース制御の拡張では、`com.nttdocomo.ui.PhoneSystem` クラス、`com.nttdocomo.opt.ui.PhoneSystem2` クラス、`com.nttdocomo.ui.Display` クラスに対し、以下のリソースを追加定義しています。なお、`PhoneSystem2` クラスは `PhoneSystem` クラスのサブクラスです。

- ・ 音量制御 (`PhoneSystem2` クラス)
- ・ 折りたたみ開閉状態取得 (`PhoneSystem` クラス)
- ・ ソフトキーの背景色制御 (`PhoneSystem2` クラス)
- ・ 追加キー制御 (`Display` クラス、`PhoneSystem2` クラス)
- ・ バイブレータ制御 (`PhoneSystem` クラス)
- ・ 背面 LCD イルミネーション制御 (`PhoneSystem2` クラス)
- ・ 背面ライト制御 (`PhoneSystem2` クラス)
- ・ メモ LED 制御 (`PhoneSystem2` クラス)
- ・ 明度・コントラスト制御 (`PhoneSystem2` クラス)
- ・ 携帯電話組み込みの効果音再生 (`PhoneSystem` クラス)

以下、各機能の概要について解説します。

<音量制御>

携帯電話の各種音量を制御する、参照・設定の両方が可能な属性です。以下の属性が `PhoneSystem2` クラスに定義されています。

- ・ `DEV_MELODY_VOLUME` (メロディ再生音量)
- ・ `DEV_MIC_VOLUME` (イヤホンマイクのマイク音量)
- ・ `DEV_SE_VOLUME` (携帯電話組み込みの効果音再生音量)

属性値には、`PhoneSystem2.ATTR_VOLUME_MIN` (最小) から `PhoneSystem2.ATTR_VOLUME_MAX` (最大) までの間の値を設定します。

＜折りたたみ開閉状態取得＞

折りたたみ型またはフリップ型の携帯電話において本体またはフリップの開閉状態を取得するための、参照のみ可能な属性です。属性 `PhoneSystem.DEV_FOLDING` に対し属性値 `PhoneSystem.ATTR_FOLDING_OPEN` が設定されている場合は開状態、`PhoneSystem.ATTR_FOLDING_CLOSE` が設定されている場合は閉状態であることを示します。ストレート型の携帯電話で `PhoneSystem.DEV_FOLDING` を参照すると、常に値-1が返されます。

＜ソフトキーの背景色制御＞

ソフトキー表示エリアおよびインジケータエリアの背景色を制御するための、参照・設定の両方が可能な属性です。属性 `PhoneSystem2.DEV_INDICATOR_BACKGROUND` に対し、色を表す機種固有の整数値 (`Graphics.getColorOfRGB()` メソッドなどで返される値) を設定することでソフトキー表示エリアおよびインジケータ表示エリアの背景色を変更することができます。

＜追加キー制御＞

i アプリ基本 API では、キーイベントの発生するキーとして数字キー（*と#を含む）、上下左右キー、選択（決定）キーおよび2つのソフトキーを規定しています。追加キー制御では、これら以外のキーをサポートします。

- ・ ページダウンキー、ページアップキー

対応する定数として、`com.nttdocomo.ui.Display` クラスに `KEY_PAGE_DOWN` および `KEY_PAGE_UP` が定義されています。

- ・ 上下左右キーの8方向サポート

対応する定数として、`com.nttdocomo.ui.Display` クラスに `KEY_LOWER_LEFT`、`KEY_LOWER_RIGHT`、`KEY_UPPER_LEFT`、`KEY_UPPER_RIGHT` が定義されています。

なお、8方向サポートを有効にするには、属性 `PhoneSystem2.DEV_KEY_SLANT` に対して属性値 `PhoneSystem2.ATTR_KEY_SLANT_ON` を設定します。

＜バイブレータ制御＞

バイブレータのオンオフを制御するための、参照・設定の両方が可能な属性です。属性 `PhoneSystem.DEV_VIBRATOR` に対し属性値 `PhoneSystem.ATTR_VIBRATOR_ON` を設定した場合はオン、`PhoneSystem.ATTR_VIBRATOR_OFF` を設定した場合はオフとなります。

なおオン状態にされたバイブレータは、省電力のため一定時間後にシステムによりオフ状態に戻される場合があります。

＜背面 LCD イルミネーション制御＞

背面 LCD（サブディスプレイ）のバックライトの点灯色を制御するための、設定のみ可能な属性です。属性 `PhoneSystem2.DEV_ILLUMINATION` に対し、背面 LCD イルミネーションの点灯色を表す属性値（`PhoneSystem2` クラスで定義されている、`ATTR_ILLUMINATION` で始まる名前の属性値）を設定します。なお、属性値として、点灯色と `PhoneSystem2.ATTR_ILLUMINATION_GRADUALLY` の論理和を取ったものを設定すると、その点灯色で徐々に点灯します。

<背面ライト制御>

背面にある i モードマークのライトの点灯色を制御するための、設定のみ可能な属性です。属性 `PhoneSystem2.DEV_I_LIGHT` に対し、背面ライトの点灯色を表す属性値（`PhoneSystem2` クラスで定義されている、`ATTR_I_LIGHT` で始まる名前の属性値）を設定します。

<メモ LED 制御>

メモ LED の点灯色を制御するための、参照・設定の両方が可能な属性です。属性 `PhoneSystem2.DEV_MEMO_LED` に対し、色を表す機種固有の整数値（`Graphics.getColorOfRGB()` メソッドなどで返される値）を設定することでメモ LED の点灯色を選択できます。

なお、属性値として `PhoneSystem2.ATTR_MEMO_LED_ON` を設定した場合はデフォルトの点灯色で点灯します。また、`PhoneSystem2.ATTR_MEMO_LED_OFF` を設定した場合はメモ LED を消灯します。

<明度・コントラスト制御>

ディスプレイおよびサブディスプレイの明度およびコントラストを制御するための、参照・設定の両方が可能な属性です。以下の属性が `PhoneSystem2` クラスに定義されています。

- ・ `DEV_DISPLAY_BRIGHTNESS`（ディスプレイの明度）
- ・ `DEV_DISPLAY_CONTRAST`（ディスプレイのコントラスト）
- ・ `DEV_SUBDISPLAY_BRIGHTNESS`（サブディスプレイの明度）
- ・ `DEV_SUBDISPLAY_CONTRAST`（サブディスプレイのコントラスト）

明度の属性値には、`PhoneSystem2.ATTR_BRIGHTNESS_MIN`（最小）から `PhoneSystem2.ATTR_BRIGHTNESS_MAX`（最大）までの間の値を設定します。またコントラストの属性値には、`PhoneSystem2.ATTR_CONTRAST_MIN`（最小）から `PhoneSystem2.ATTR_CONTRAST_MAX`（最大）までの間の値を設定します。

<携帯電話組み込みの効果音再生>

`PhoneSystem` クラスには以下の 5 つの効果音が定義されており、効果音をサポートする携帯電話では `PhoneSystem.playSound()` メソッドを使用してそれらの効果音を再生することができます。

- ・ `SOUND_ALARM`（アラームの通知）
- ・ `SOUND_CONFIRM`（処理受け付けの通知）
- ・ `SOUND_ERROR`（エラーの通知）
- ・ `SOUND_INFO`（情報の通知）
- ・ `SOUND_WARNING`（警告の通知）

なお、これらの効果音に実際に割り当てられる音色はメーカーにより異なります。

第7章

サブディスプレイ描画

サブディスプレイ描画機能では、サブディスプレイ（携帯電話の背面などに取り付けられている小さな液晶ディスプレイ）に対し描画する機能を提供します。サブディスプレイへの描画には、`com.nttdocomo.opt.ui.SubDisplay` クラスを使用します。

`SubDisplay` クラスは、サブディスプレイのデバイス情報を取得するメソッドとサブディスプレイへの描画メソッドを備えるシンプルなユーティリティクラスです。

<サブディスプレイのデバイス情報取得>

`SubDisplay` クラスには、サブディスプレイのデバイス情報を取得するためのメソッドとして以下のものが用意されています。

- `getHeight()` : スクリーンの高さを取得します。
- `getWidth()` : スクリーンの幅を取得します。
- `isColor()` : スクリーンがカラー表示をサポートしているかどうかを調べます。
- `numColors()` : スクリーンで利用できる色数を取得します。

i アプリ基本 API の `Display` クラスにも同名のメソッドが用意されていますが、それらのメソッドと上記メソッドの使用方法は共通です。

<サブディスプレイへの描画>

サブディスプレイへの描画は、ディスプレイへの描画と異なり直接 `SubDisplay` クラスに描画指示を行います。サブディスプレイに描画できるものはイメージ（`Image` オブジェクト）のみです。

サブディスプレイへのイメージの描画には、`SubDisplay.setImage()` メソッドを使用します。

第 8 章

端末ネイティブのデータストレージ アクセス

端末ネイティブのデータストレージとは、待ち受け画面に設定する待ち受け画像データや着信メロディ、さまざまな状況に応じて画面に表示される 3D キャラクタデータなど、携帯電話のネイティブアプリケーションが使用するデータを保存する領域を指します。端末ネイティブのデータストレージアクセス機能では、アプリケーションからこれらのデータ領域にアクセスする手段を提供します。DoJa-2.0 プロファイルでは、利用可能なデータストレージは待ち受け画像保存領域と 3D キャラクタデータ保存領域の 2 つです。

ただし、これらの領域に保存されるデータに対してアプリケーションから制限なくアクセスできることは、ユーザー保護・コンテンツの著作権保護の観点から好ましくありません。

本機能では端末ネイティブのデータストレージの中でも、アプリケーションがアクセスすることを許された特別なエリア（以降、許可エリアと呼びます）にのみ、アプリケーションがアクセスすることを許可しています。許可エリアには、以下のような制限があります。

- ・ ユーザーは、ブラウザなどからのユーザー操作による待ち受け画像や 3D キャラクタデータの保存領域として許可エリアを使用することはできません。ユーザーが使用するエリアとアプリケーションが使用するエリアは明確に分離されており、相互に上書きすることはできないようになっています。
- ・ この機能を使用してアプリケーションが許可エリアにデータを書き込む際、i アプリ実行環境はユーザーに確認を求めるダイアログを表示します。このダイアログでユーザーの許可が得られなかった場合、データの書き込みは行われません。
- ・ 許可エリアに保存された待ち受け画像や 3D キャラクタデータを待ち受け画面やキャラクタとして表示できるようにするには、ユーザーは許可エリアの待ち受け画像や 3D キャラクタデータを使用することを明示的に携帯電話に設定しなければなりません。
- ・ アプリケーションは、許可エリアにデータの書き込みのみ行うことができます。許可エリアのデータを読み出すことはできません。

<待ち受け画像保存領域へのアクセス>

待ち受け画像保存領域に待ち受け画像を保存するには、`com.nttdocomo.opt.ui.PhoneResource` クラスの `setStandbyImage()` メソッドを使用します。このメソッドには、引数として待ち受け画像として保存したいイメージ（Image オブジェクト）を指定します。

<3D キャラクタデータ保存領域へのアクセス>

3D キャラクタデータ保存領域に 3D キャラクタデータを保存するには、
`com.nttdocomo.opt.ui.PhoneResource` クラスの `set3DCharacter()` メソッドを使用します。携帯電話で使用される 3D キャラクタデータのフォーマットは携帯電話により異なるため、このメソッドには、その携帯電話に適したデータをバイト配列の形で指定します。

第 9 章

カメラ制御

カメラ制御機能では、携帯電話に搭載されているカメラ機能にアプリケーションからアクセスする手段を提供します。動画撮影に対応している機種では、静止画像だけでなく動画像の取り扱いをサポートする場合があります。

またカメラ機能を搭載した携帯電話の中には、ネイティブアプリケーションの機能として、携帯電話間で赤外線ポートを介して撮影画像を交換できる機能を持った機種があります。i アプリの OBEX 外部接続機能を使用することで、このような携帯電話のネイティブアプリケーションと別の携帯電話上の i アプリとの間で画像データを交換することもできます。

9.1 カメラの制御

カメラ制御機能は `com.nttdocomo.opt.ui.Camera` クラスで提供されます。携帯電話には複数のカメラデバイスが搭載されていることがありますが、個々のデバイスはカメラ ID（個々のデバイスに固定的に割り当てられた、0 から始まる整数）により識別されます。アプリケーションは `Camera.getCamera()` メソッドにカメラ ID を指定することにより、カメラオブジェクトを取得することができます。

この機能を使用することでアプリケーションは、カメラオブジェクトに対し以下のような制御を行うことができます。

- ・ カメラデバイスの設定実行
- ・ ネイティブのカメラ機能の起動
- ・ 撮影画像の取得

(A) カメラデバイスの設定実行

カメラデバイスの設定内容には、デバイス属性と撮影画像サイズの 2 つがあります。

デバイス属性の設定は、`Camera.setAttribute()` メソッドにより行います。規定されている属性には連写モード、画質、マイク音量があります（画質とマイク音量は動画撮影時のみ有効）が、実際のカメラデバイスがどの属性に対応しているかは機種により異なります。アプリケーションは、`Camera.isAvailable()` メソッドにより、その携帯電話で対応している属性を判定することができます。また、`Camera.getAttribute()` メソッドにより各属性の現在の属性値を知ることができます。

撮影画像サイズの設定は `Camera.setImageSize()` メソッドを使用していきます。画像サイズは縦横のドット数で表します。指定されたサイズがその携帯電話でサポートされていない場合、システムは指定されたサイズをサポートしているサイズに補正します。

ユーザーは、アプリケーションがカメラデバイスに設定した内容を撮影時に変更することはできません。

(B) ネイティブのカメラ機能の起動

アプリケーションは `Camera.takePicture()` メソッドを呼び出すことで、静止画撮影モードでカメラ機能を起動することができます。またアプリケーションからの動画撮影に対応している機種では、`Camera.takeMovie()` メソッドを呼び出すことで動画撮影モードでカメラ機能を起動することもできます。

カメラ機能を起動するとアプリケーションの実行は中断し、カメラ操作画面がディスプレイに表示されます。実際の撮影はユーザーの操作に基づいてのみ行われ、アプリケーションがシャッターの操作などを行うことはできません。

ユーザーがカメラ操作画面を終了させると、アプリケーションの実行が再開されます。カメラ機能終了時にユーザーが撮影画像の保存を行っていた場合、その画像はカメラオブジェクト内に保持されています。

なお、待ち受けアプリケーションが非活性化状態にある場合、そのアプリケーションはカメラ機能を起動することはできません。

(C) 撮影画像の取得

カメラオブジェクト内に保持された画像は、次にカメラ機能が起動されるか、アプリケーションが終了するか、または `Camera.disposeImages()` メソッドにより明示的に破棄されるまでカメラオブジェクト内に保持されます。アプリケーションは、カメラオブジェクト内に保持されている画像にアクセスすることができます。それ以外の、ネイティブの画像保存領域に保存されている画像にアクセスすることはできません。

カメラオブジェクトに保持されている画像をメディアイメージの形式で取得するには、`Camera.getImage()` メソッドを使用します。また、画像をバイトストリーム（静止画像の場合は JPEG フォーマット、動画画像の場合は i モーションフォーマット）の形式で取得するには、`Camera.getInputStream()` メソッドを使用します。これらのメソッドの引数には、カメラオブジェクト内に保存されている画像のインデックスを指定します。画像のインデックスには通常は 0 を指定しますが、静止画の連写撮影を行っている場合には、インデックスで目的の画像を指定することができます。

9.2 撮影画像の携帯電話間での交換

カメラ機能を搭載した携帯電話の中には、ネイティブアプリケーションの機能として、携帯電話間で赤外線ポートを介して撮影画像を交換できる機能を持った機種があります。この機能は OBEX 外部接続機能に基づいて実現されているため、OBEX 外部接続機能を使用する i アプリとこのネイティブアプリケーションの間でも画像の交換を行うことができます。

ネイティブアプリケーションが画像を交換する際には、IrMC vNote バージョン 1.1 に準拠したオブジェクトを使用します。ただし、本機能では画像交換に必要な追加情報を送受信するために vNote のオブジェクトプロパティを拡張しています。

以下に、本機能で使用するオブジェクトプロパティの内容を示します。

項番	プロパティ	値	備考
1	VERSION	1.1	vNote のバージョン
2	X-DOCOMO-TYPE	JPEG または GIF	画像データのフォーマット種別
3	X-DOCOMO-SIZE	<width>X<height>	画像のサイズ (横 X 縦) X はアルファベット半角大文字 (ASCII コード 0x58)
4	X-DOCOMO-FILESIZE	<filesize>byte	画像のファイルサイズ (バイト単位)
5	X-DOCOMO-FILENAME	備考欄参照	ファイル名 (拡張子を含め半角英数で 24 バイト以内の文字列)
6	SUMMARY	備考欄参照	画像タイトル (QUOTED-PRINTABLE エンコーディングを適用した、SHIFT-JIS を含む 18 バイト以内の文字列)
7	DATE	備考欄参照	撮影または編集日時 (UTC 表記)
8	X-DOCOMO-BODY	備考欄参照	Base64 形式でエンコードされた JPEG または GIF データ 本プロパティ値の終端には<CR><LF>を付加してください

以下に、画像データを交換するための vNote オブジェクトの記述例を示します。

```

BEGIN:VNOTE
VERSION:1.1
X-DOCOMO-TYPE:JPEG
X-DOCOMO-SIZE:120X120
X-DOCOMO-FILESIZE:3000byte
X-DOCOMO-FILENAME:picture_001.jpg
SUMMARY;CHARSET=SHIFT_JIS;ENCODING=QUOTED-PRINTABLE:=89=E6=91=9C
DATE:20020715T153530Z
X-DOCOMO-BODY;ENCODING=BASE64:<Base64 形式でエンコーディングされた画像データ本体+<CR><LF>>
END:VNOTE

```

※各行の行末は<CR><LF> (0x0d0a) としてください。X-DOCOMO-BODY についてはデータ末尾の<CR><LF>と合わせて 2 つの<CR><LF>が連続することになります。

i アプリ側からネイティブアプリケーション側に画像データを送信する場合、ネイティブアプリケーション側は OBEX サーバーとして動作します。i アプリ側は OBEX クライアントとして実装し、画像データ (vNote オブジェクト) を PUT オペレーションで送信するようにします。

またネイティブアプリケーション側から i アプリ側に画像データを送信する場合、ネイティブアプリケーション側は OBEX クライアントとして動作します。i アプリ側は OBEX サーバーとして実装し、ネイティブアプリケーションから PUT オペレーションで送信される画像データ (vNote オブジェクト) を受信するようにします。

注意事項：

- 画像の交換においては、vNote オブジェクトの複数送受信を行うことはできません。画像データは 1 回のオペレーションで 1 つだけ送信することができます。
- i アプリが OBEX 外部接続機能を使用して送受信できるデータのサイズは 30KByte までに制限されています。画像の交換においては、上記の vNote オブジェクト全体のサイズが 30KByte 以内である必要があります。これは、i アプリ側が画像を送信する場合も受信する場合も同様です。
- i アプリからネイティブアプリケーション側に画像データを送信する際、PUT オペレーションで送信するオブジェクトの Name ヘッダは"mypic.vnt"としてください。
- i アプリからネイティブアプリケーション側に画像データを送信する際、送信可能な画像サイズ (X-DOCOMO-SIZE で示される画像の縦横サイズ) は受信側携帯電話のカメラがサポートする撮影画像サイズに制限されます。

第 10 章

ポインティングデバイス

ポインティングデバイスは、画面上を自由に移動可能なカーソルを操作するための入力デバイスです。ユーザーはポインティングデバイス进行操作してカーソルを移動させることで、画面の座標空間上の位置をアプリケーションに伝えることができます。ポインティングデバイスの機能は `com.nttdocomo.opt.ui.PointingDevice` クラスで提供されます。なお、ポインティングデバイス機能は低レベル API 利用時のみ使用することができます。

ポインティングデバイスは通常は無効化されています。ポインティングデバイスの有効、無効を切り替えるには、`PointingDevice.setEnabled()` メソッドを使用します。また、`PointingDevice.isEnabled()` メソッドにより、現在ポインティングデバイスが有効であるか否かを確認することができます。

ポインティングデバイスが有効になっている状態でカレントフレームに `Canvas` を設定すると、ユーザーがポインティングデバイスの操作を開始した際にポインティングデバイス操作開始イベント (`Display.POINTER_MOVED_EVENT`) がその `Canvas` の `processEvent()` メソッドに通知されるようになります。このイベントの通知を受けて `PointingDevice.getX()` メソッドおよび `PointingDevice.getY()` メソッドを呼び出すことで、画面上のカーソルの位置を取得することができます。これらのメソッドで取得されるカーソル位置は画面の左上を原点とした絶対座標となります。