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

2008－DD－69（2）
2008 / 11 / 27

知的Web技術に基づく携帯電話向け情報編纂システムとその応用

浅見 昌平† 近藤 圭佑† 大囿 忠親† 新谷 虎松†

† 名古屋工業大学大学院情報工学専攻
E-mail: †asami@toralab.ics.nitech.ac.jp

あらまし 携帯電話向けのFlashコンテンツを作成に関連して，既存のWebブラウザを利用した知的Webシステムについて述べる．本システムでは，データ入力用のテンプレートを作成し，そのテンプレートに対して様々なデータを入力することでFlashコンテンツを自動生成する．また，Flashコンテンツを自動的に生成するためのMobileStackMLを用いることで，既存のWebページから情報を自動抽出し，携帯電話用コンテンツを効果的に生成可能になる．
キーワード 携帯電話，情報編纂，情報抽出，MobileStackML

Implementing an Information Compilation System based on Intelligent Web for Mobile Phones

Shohei ASAMI†, Keisuke KONDO†, Tadachika OZONO†, and Toramatsu SHINTANI†

† Graduate School of Engineering, Nagoya Institute of Technology
E-mail: †asami@toralab.ics.nitech.ac.jp

Abstract The cellular phone constrains many producer to create a mobile content because of three problems on cellular phone, including file size, compatibility, and usability. We propose a method for generating mobile contents in restricted environments utilizing the information compilar system. We proposed in which we use the card model for designing effectively mobile contents.
Key words Cellular Phone, Information Compilar, Information Extraction, MobileStackML

1. はじめに

本研究では，携帯電話向けコンテンツの作成における課題を解決するためのシステムを構築した．携帯電話向けにコンテンツを作成する場合，コンテンツのファイルサイズ，携帯電話のキャリアや機種の違いによる互換性，コンテンツの閲覧性など，多くの制約が存在する．これらの制約を満たしつつ，効果的なコンテンツを作成するためには，職人芸的な技術が必要である．すなわち，互換性と閲覧性の制約を解消するためには，携帯電話向けのコンテンツに関する深い専門知識が必要であり，携帯電話向けコンテンツ作成の難易度やコストを高くする原因となっている．

本研究の目的は，携帯電話向けのコンテンツの製作を容易にすることである．そのために，携帯電話向けのコンテンツを製作する過程で制約となるファイルサイズ，互換性，および閲覧性の制約を満たすコンテンツを自動的に生成する携帯電話向け情報編纂システムの実現を目指す．これにより，コンテンツ制作者がこれらの問題を意識せず製作できることが可能になる．本研究では，カードモデル[2]を用いることで，携帯電話の小さな画面に適合するコンテンツの自動生成が可能になった．

本システムの応用として，XML-RPCプロトコルのような知的Web技術を用いて外部アプリケーションと連携可能な仕組みを構築する．知的Web技術は，UDDI，WSDL，SOAPのような標準規格を用いたWebサービスシステム間の接続を実現する．本システムにおいても，Webページからの情報抽出と連携して，自動的に携帯電話向けコンテンツを生成するための技術を利用する．本システムにおける外部アプリケーションとの連携では，入力される全てのデータを本システムが提供するAPIに与えることによって，コンパイルされたFlashコンテンツが得られる．この仕組みを利用すると，例えば，レシピサイトから料理のレシピを携帯電話向けのFlashコンテンツに変換するサービスを実現することができる．

以降，本論文では，2章でカードモデルについて説明し，コンテンツを生成するためのシステムモデルについて述べる．3章で，実装したカードモデルに基づくコンテンツを生成するためのシステムについて説明する．また，作成したコンテンツの実行例を示す．4章では，MobileStackMLに基づく外部アプリケーションとの連携について述べる．5章で，本システムとカードモデルについて考察を行い，最後に6章で本論文をまとめる．

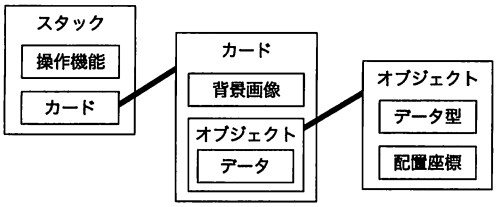

図 1　カードモデル
Fig. 1　Card Model

2. 携帯電話向けコンテンツのモデル

2.1 カードモデル

コンテンツ作成の際，携帯電話の画面が小さいことは制作者にとって大きな制約である．本研究では，掲載する情報を効果的に配置するためにカードモデルを採用する．カードモデルとは，携帯電話の画面に適合する大きさの背景画像を用意し，その上のレイヤーに情報を配置するモデルである．

図 1 にカードモデルを示す．カードモデルにおいて，カードは背景画像とオブジェクトから構成される．オブジェクトは，格納されるデータの型と配置される位置座標を持つ．オブジェクトには，テキスト，画像，および Flash 型のデータが格納される．また，オブジェクトが配置される座標は，必ず背景画像上でなければならない．カードは，複数のオブジェクトが配置され，単一の背景画像を与えられた Flash コンテンツである．さらに，複数のカードを結合し，表示されるカードの切り替えを行うための操作機能を付加したものをスタックと呼ぶ．スタックでは，全てのカードの背景画像は共通である．

カードモデルは，前述した携帯電話における制約のいくつかを解消する．閲覧性に関して，1 枚のカードが携帯電話の画面に適合しているため，画面外にコンテンツが隠れることを防ぐ．また，スタックに付加した操作機能により，複数のカードをめくることで表示切り替えを可能にしている．互換性に関して，Flash コンテンツにコンパイルされることで，携帯電話のキャリアや機種の違いを意識することなくコンテンツを作成することができる．ファイルサイズに関して，共通の背景画像を持ち，情報を掲載する部分のみ交換することでファイルサイズの増大を防ぐ．また，スタックに持たせるカードの枚数を調整することで，ファイルサイズを一定量まで任意に減らすことができる．このように，カードモデルは，携帯電話における閲覧性，互換性，およびファイルサイズの制約を解消する．

2.2 スタック生成モデル

図 2 にスタック生成モデルを示す．スタックは，複数のカードと操作機能を結合することによって生成される．1 つ 1 つのカードは，レイアウトを作成するステップとレイアウトにデータを入力するステップに分かれて作成する．

まず，カードが持つ背景画像，およびオブジェクトの配置を行うステップについて述べる．このステップではカードのレイアウトを設計する．カードのレイアウト設計では，カード上に置くオブジェクトの設計をするためのフォアグラウンド，およびカードの背景を設計するためのバックグラウンドをそれぞれ作成する．フォアグラウンドでは，オブジェクトに入力するデータの型や，配置する座標，オブジェクト同士の重なりを処理するためのレイヤーを決定する．バックグラウンドでは，様々な文字や画像を組み合わせて，1 枚の背景画像を作成する．作成したフォアグラウンドとバックグラウンドを合成することによって，カードのレイアウトを作成する．

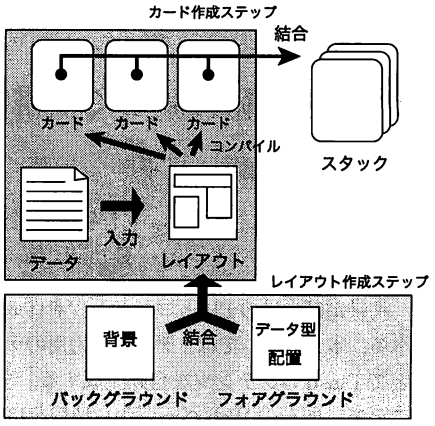

図 2　スタック生成モデル
Fig. 2　System Model

次に，カードを作成するステップについて述べる．スタックに追加する 1 つ 1 つのカードは，共通のレイアウトを使用して作成される．レイアウト中のオブジェクトに対し，それぞれが持つデータ型に従ってデータを入力することで 1 つのカードを生成する．テキスト型のオブジェクトには文字を，画像型のオブジェクトには画像のパスを，そして Flash 型のオブジェクトには Flash のパスを入力する．また，レイアウト中のオブジェクト全てにデータを入力する必要はない．コンパイラは，このステップで作成したカードを順に結合し，操作機能を付加することで 1 つのスタックを生成する．スタックは，Flash コンテンツであり，1 枚 1 枚のカードをめくる操作によって閲覧することができる．

2.3 MobileStackML

本研究では，XML に基づくスタックの仕様記述形式として MobileStackML を定義した．

図 3 に MobileStackML によるスタックの仕様記述を示す．MobileStackML に記述しなければならない情報として，使用するレイアウト名，書き出すスタックの名前，およびカードの仕様記述がある．使用するレイアウト名は，レイアウトエディタを用いて作成されたカードのレイアウトを参照する識別子である．書き出すスタックの名前は，swf ファイルの名前となり，ユーザが自由に命名することができる．

カードの仕様記述には，カード 1 枚につき，レイアウトに受け渡すデータの型と値をオブジェクト数だけ指定する．図 4 に MobileStackML によるオブジェクトの仕様記述の例を示す．member 要素の中には，name 要素と value 要素がある．オブジェクトのデータ型を指定する場合，name 要素に type，value

```
<?xml version="1.0"?>
<methodCall>
 <methodName>wisdomcard.compileStack</methodName>
 <params>
  <!-- スタックの仕様記述 -->
  <param>
   <value>
      <struct>
      <!-- 使用するレイアウト名 -->
      <member>
        <name>layout</name>
        <value><string>db.name:demo1</string></value>
      </member>
      <!-- 書き出すスタックの名前 -->
      <member>
        <name>name</name>
        <value><string>example</string></value>
      </member>
      <!-- カードの仕様記述の配列 -->
      <member>
            ....
```

図 3 MobileStackML によるスタックの仕様記述
Fig. 3 Stack Specification on MobileStackML

```
<!-- データ型 -->
<member>
<name>type</name>
<value><string>image</string></value>
</member>
<!-- データ -->
<member>
<name>data</name>
<value><string>http://...</string></value>
</member>
```

図 4 MobileStackML によるオブジェクトの仕様記述
Fig. 4 Object Specification on MobileStackML

要素に string 要素でデータ型（string，image，flash）を記述する．例では，データ型に image を指定している．次に，オブジェクトに入力するデータを指定する．member 要素の name 要素に data，value 要素に string 要素でデータを記述する．例では，画像の URL を指定している．データ型と入力するデータを指定することで，1 枚のカード中の 1 つのオブジェクトを生成することができる．

MobileStackML で複数のカードを作成したい場合，カードの仕様記述の中に，配列としてカードの情報を記述する．1 枚のカードの記述は，前述したデータ型と入力するデータの繰り返しパターンであるため，これを array 要素の中に繰り返し記述すればよい．

3. 携帯電話向け情報編纂システム

本研究では，スタック生成モデルに基づき，携帯電話向け情報編纂システムを実装した．本システムでは，コンテンツ作成フェーズを 2 つに分け，カードのレイアウトを作成する機構をレイアウトエディタ，カードのレイアウトにデータを入力する機構をカードエディタと呼ぶ．

本システムは，Web ブラウザ上で動作する Web アプリケーションとして実装した．カードの設計や，データ入力をクライアントサイドで処理し，Flash コンパイルや，画像化，画像の減色などはサーバサイドで処理する．

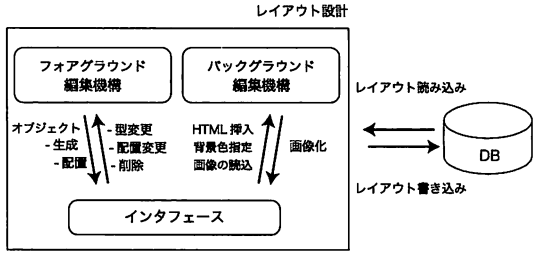

図 5 レイアウトエディタのシステム構成図
Fig. 5 System Outline of Layout Editor

3.1 レイアウトエディタ

図 5 にレイアウトエディタのシステム構成図を示す．レイアウトエディタは，インタフェース，フォアグラウンド編集機構，バックグラウンド編集機構，およびデータベースから構成される．レイアウトエディタでは，ユーザがインタフェースを介してカードのレイアウトを設計することができる．インタフェースは，フォアグラウンドとバックグラウンドの編集状態をそれぞれ描画する機能，オブジェクトを GUI で編集操作できる機能，およびバックグラウンドの背景に画像や HTML を貼付ける機能を持つ．

フォアグラウンド編集機構は，カードが持つオブジェクトを編集する機構である．オブジェクトを管理する機能を持ち，オブジェクトの生成，配置，変更，および削除ができる．オブジェクトの生成機能，および配置機能を用いて，指定されたデータ型を持つオブジェクトをインタフェースに与え，指定した位置に表示させることができる．ユーザは，インタフェースを介して，オブジェクトの型，位置座標，大きさ，レイヤーの変更，および削除を行う．これらの操作によって，ユーザはカード上のオブジェクト配置を設計する．

バックグラウンド編集機構は，カードの背景画像を編集する機構である．カードの背景画像は，最終的に 1 枚の画像でなければならない．しかし，レイアウト設計の段階では，1 枚の画像である必要はない．バックグラウンド編集機構では，背景をキャンパスに見立て，複数の画像を任意の位置に貼付けることや，背景を塗りつぶす操作，さらに HTML タグを挿入することを許容する．バックグラウンド編集機構は，編集結果を受け取り，サーバ側でキャンパスの描画状態を 1 枚の画像に変換する処理を行う．得られた 1 枚の画像は，カードの背景画像としてインタフェースに描画される．

データベースには，レイアウトエディタを用いて設計されたカードのレイアウトが格納される．格納されるデータは，レイアウトの背景情報（画像のパス），オブジェクトの情報（型，位置座標），およびレイアウト名である．レイアウト名は，作成したレイアウトを参照するするための識別子として付与する．フォアグラウンド編集機構で作成したオブジェクトの情報，およびバックグラウンド編集機構で作成した背景画像を取得し，カードのレイアウト情報としてデータベースに格納する．また，既に作成されたカードのレイアウトを読み込み，レイアウトエディタで再び編集することもできる．

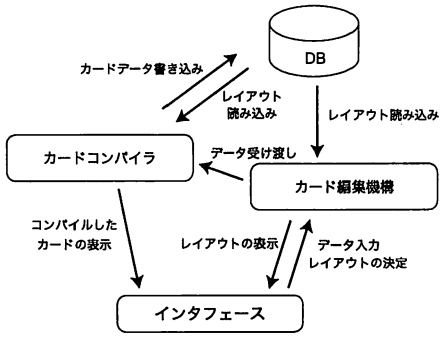

図 6　カードエディタのシステム構成図
Fig. 6　System Outline of Card Editor

3.2　カードエディタ

図 6 にカードエディタのシステム構成図を示す．カードエディタは，インタフェース，カード編集機構，カードコンパイラ，およびデータベースから構成される．データベースは，レイアウトエディタと共通のデータベースを用いる．カードエディタでは，インタフェースを介して，ユーザが選択したレイアウトにデータ入力を行うことができる．また，ユーザはインタフェースを用いて，スタックに挿入するカードの順番を変更することができる．インタフェースは，作成されたレイアウトを選択する機能，データ入力を行う機能，カードの結合順を変更する機能を持つ．

カード編集機構は，カードを生成するために必要な機能を持つ．カード編集機構は，サーバ上のデータベースから格納されているレイアウトを読み込み，インタフェースに表示する．ユーザは，インタフェースに表示されているレイアウトから任意のレイアウトを選択し，作成するカードのレイアウトを決定する．カード編集機構は，選択されたレイアウトの情報を基に，データを入力するためのフォームを生成する．ユーザは，このフォームにデータを入力することで，カード編集機構にデータを受け渡すことができる．カード編集機構は，受けとったデータと，レイアウトへの参照をカードコンパイラに与える．

カードに入力するデータには，付加情報を埋め込むことができる．埋め込む情報は，他の Web ページへ移動するためのリンク情報，および指定したカードへ移動するためのアンカー情報である．スタックが持つ操作機能は，付加情報が埋め込まれたデータに対し，特別な処理を行う．例えば，テキストに Web ページの URL を埋め込むと，生成されたスタックではテキストにフォーカスが合わせられ，埋め込まれた URL へ移動することができる．また，テキストにアンカー情報が埋め込まれた場合，テキストをクリックすることで指定された番号のカードまで自動的にめくられる．

カードコンパイラは，カード編集機構から受け取ったデータを基に，カードを生成する機能を持つ．まず，受け取ったデータをデータベースに格納し，使用するレイアウトを読み込む．次に，レイアウトが持つ各オブジェクトに受け取ったデータを入力する．このとき，受け取ったデータの型と，レイアウトのオブジェクトのデータ型の整合性を調べる．整合性が取れていれば，データの入力は正常に行われる．入力されたデータは，画像化され，レイアウトが持つ背景画像の上に描画される．全てのデータがレイアウト上のオブジェクトに入力された後，Flash コンテンツへコンパイルされ，インタフェースに表示される．

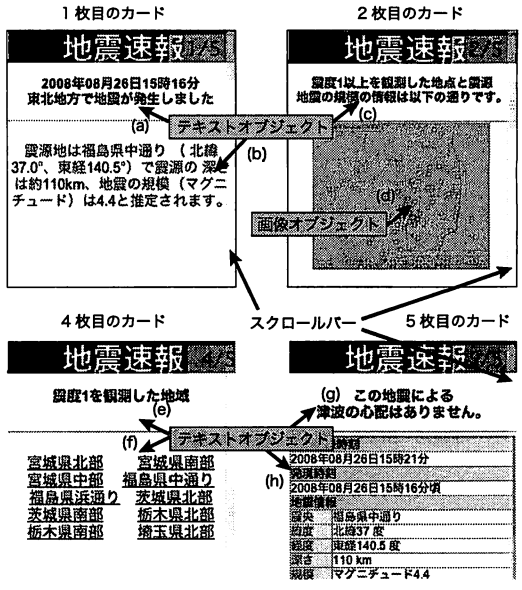

図 7　生成されたスタックの例
Fig. 7　One of Stack Example

前述のように，ユーザはカードエディタを用いて複数のカードを作成することができる．生成されたカードは，画像化された入力データ，および背景画像を持つ Flash コンテンツになっている．これらのカードを結合し，1 つのスタックを生成するために，再度カードコンパイラを用いる．ユーザはインタフェースを介して，結合するカードの順序を決定する．その後，スタックを生成する要求を送ると，カードコンパイラがカードの結合処理を行う．カードの結合処理では，複数のカードを結合し，カードをめくるための操作機能を付加する．カードの結合処理において，カードの背景画像は共通の画像を用いるため，容量が削減される．

3.3　実 行 例

図 7 に本システムを用いて生成されたスタックの例を示す．作成されたスタックは 5 枚のカードから構成され，図では 1 枚目，2 枚目，4 枚目，5 枚目を示している．ここでは，5 枚のカードの中から，見た目が異なる 4 枚のカードを示して説明を行う．カードの背景画像には，レイアウトエディタを用いて作成された地震速報というタイトルが掲載されている．1 枚目のカードと 2 枚目のカードの共通する部分は 1 枚の背景画像として用いられている．カードの右側にはスクロールバーが設置されており，現在見ているカードの相対的な位置を知ることができる．

例で用いたレイアウトには，図の (a)，(b)，(c)，(e)，(f)，

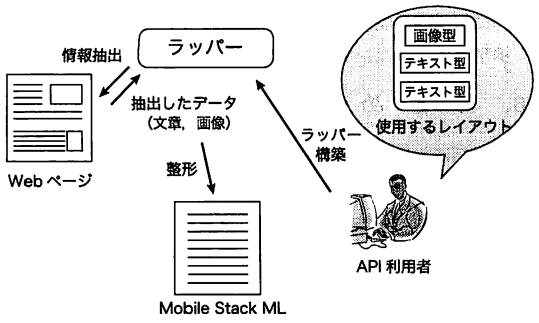

図 8　既存の Web ページから MobileStackML の作成
Fig. 8　Creating MobileStackML using Web Wrapper

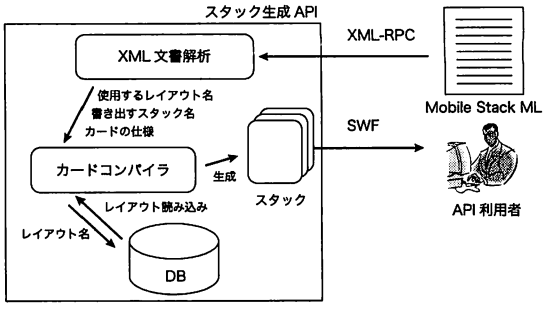

図 9　MobileStackML に基づくスタックの生成
Fig. 9　Creating Stacks using MobileStackML

(g)，(h) に示されるテキスト型のオブジェクト，および (d) に示される画像型のオブジェクトが配置されている．(a)，(c)，(e)，(g) のテキストオブジェクトは，同じオブジェクトであり，入力されたデータだけが異なっている．(b) と (d) のオブジェクトは，同じ位置に配置されているが，異なるオブジェクトである．2 つのオブジェクトは，異なるデータ型を持ち，レイヤーが重なって配置されている．1 枚目には，テキスト型のオブジェクトにだけデータが入力されており，2 枚目には，画像型のオブジェクトにだけデータが入力されている．同様に，4 枚目のカードの (f) と 5 枚目のカードの (h) も，1 枚目のカードの (b) と同じテキスト型のオブジェクトである．これらのオブジェクトは，入力されたデータのフォーマットが異なるため，別のオブジェクトに見えてしまう．特に，(h) には，HTML 形式のテキストを入力しているため，複雑な表を描画できている．

4. 情報抽出に基づくスタック生成

MobileStackML を利用することで，カードエディタを用いて手動でスタックを作成する方法以外に，外部プログラムによるスタック生成を自動化できる．既存の Web ページに対するラッパーを与え，MobileStackML に変換することで，既存の Web ページをスタックへと情報編纂することができる．ここでは，MobileStackML によってスタックを返すシステムをスタック生成 API と呼ぶ．

スタック生成 API は，MobileStackML で定義されたスタック記述を受け取ることによって，スタックを生成し，SWF ファイルを返す．スタック生成 API を呼び出すためのプロトコルとして XML-RPC を用いる．XML-RPC とは，RPC プロトコルの一種であり，エンコードに XML，転送機構に HTTP を採用している．RPC プロトコルとは，Remote Procedure Call [1] と呼ばれ，外部のコンピュータにあるサブルーチンや手続きを呼び出すことを実現する技術である．XML-RPC は，他の規格と比べると，小数のデータ型やコマンドだけを定義した単純なプロトコルである．

図 8 に既存の Web ページから MobileStackML を生成する流れを示す．まず，スタック生成 API の利用者は，使用するカードのレイアウトを決定する．API 利用者は，使用するカードのレイアウトが持つオブジェクトと配置を考慮し，Web ページからラッパーを構築する．ラッパーに必要な機能として，使用するカードのレイアウトが持つオブジェクトの型に適合するデータを抽出すること，およびカード生成 API が要求する MobileStackML に整形することである．

図 9 に MobileStackML に基づくスタック生成の流れを示す．2.3 節で述べた MobileStackML の仕様に従い，スタック生成 API に与える．スタック生成 API は，与えられた MobileStackML を解析し，使用するレイアウト名，書き出すスタック名，およびカードの仕様を取得する．カードの仕様は，カードコンパイラが解釈できる形に整えられる．カードコンパイラは，MobileStackML の解析結果を受け取り，データベースからレイアウト名を指定してレイアウトを読み込む．読み込んだレイアウトに対し，受け取ったカードの仕様から各オブジェクトにデータ入力を行い，カードを 1 枚づつ生成する．MobileStackML に指定されている全てのカードを生成した後，カードを結合し，操作機能を付加してスタックを生成する．その後，生成したスタックの SWF ファイルへの参照 URL を返す．MobileStackML をスタック生成 API に与えたユーザは，受け取った URL を参照することで生成されたスタックを得ることができる．

MobileStackML は，特に，Web ページからの情報抽出が可能なラッパーとの親和性が高く，レシピサイトから料理データのスタックを生成することも容易に実装できる．文献 [3] では，本手法を用いてレシピサイトの材料データを抽出した結果をスタック生成 API に渡している．

5. 考　　察

本システムを用いて携帯電話向けコンテンツを作成することについて考察する．本研究の目的は，携帯電話向けコンテンツを製作する過程で制約となるファイルサイズ，互換性，および閲覧性の問題を解消することである．本手法では，カードモデルに基づいて作成するコンテンツをモデル化し，モデルに準じた作成を行うためのシステム構築を行った．

まず，作成した携帯電話向けコンテンツの互換性について考察する．本システムで作成された Flash コンテンツは，Flash Lite 1.1 でコンパイルされている．カードモデルでは，Flash 上のスクリプトに複雑な処理が必要ないため，Flash Lite 1.1

表 1　本システムで作成された Flash コンテンツのファイルサイズ

スタック名	カードの枚数	リンク数	ファイルサイズ (KB)
地震速報	5	0	56
ニュースの見出し	5	30	76
医科大学の相談室	4	2	60
料理レシピ	9	0	96
PC カタログ	5	5	60

の機能で十分である．Flash Lite 1.1 は，現在普及している多くの携帯電話で再生することができ，キャリアや機種の違いを意識することなくコンテンツの互換性を保つことができる．

次に，作成した携帯電話向けコンテンツのファイルサイズについて考察する．通常，携帯電話には通信制約が存在し，100KB 以下のデータしか転送できないこともある．本システムで作成された Flash コンテンツのファイルサイズを表 1 に示す．本システムを用いて，タイプの異なる 5 つのスタックを作成した．例えば，地震速報は，図 7 に示したスタックの例である．図のように，2 枚目のカードには地図を画像として貼付けてあり，その他のカードでは地震の詳細を文章で伝えている．地震速報のスタックは，ファイルサイズが 56KB であった．地震速報にはリンクは埋め込まれておらず，カードの枚数は 5 枚である．ニュースの見出しは，ニュースサイトのトップページにあるヘッドラインのリンクを取得し，MobileStackML を用いて自動的に生成されたスタックである．カードの多くがリンク情報を持つテキストで構成されているため，リンク数が 30 あるが，ファイルサイズは 76KB に納まっている．同様に，医科大学の相談室，料理レシピも，ラッパーを用いて自動的に作成されたスタックである．これらのスタックは，リンク情報を持たないテキストで構成されているため，カード 1 枚当たりのファイルサイズが小さい．一方，PC カタログは，PC の画像をカード 1 枚ずつに配置したスタックである．カードの中心に大きく画像があり，各画像にリンク情報が埋め込まれている．しかし，スタック全体のファイルサイズは 60KB であった．

表の結果から，スタックのファイルサイズは，1 枚辺り 10KB 強であることが分かる．リンク情報を多く含む場合，ファイルサイズは 15KB 程度増加してしまうが，100KB 以内に納めることは容易である．ファイルサイズが 100KB を超えた場合でも，スタックに結合するカードの枚数を調整することで，ファイルサイズを自由に調整することができる．また，カードモデルは，共通の背景をカード間で利用しているため，最もファイルサイズを圧迫する画像を複数用意する必要がない．データ入力されていないオブジェクトも，カードコンパイル時に削除されているため，ファイルサイズ削減に有効である．

最後に，作成した携帯電話向けコンテンツの閲覧性について考察する．スタックは，複数のカードをめくりながら閲覧する．1 枚のカードは，携帯電話の画面に適合するサイズで作られているため，カードを閲覧する際にスクロール操作が必要ない．スタックに付加された操作機能により，携帯電話の上下キーでカード間の画面遷移を実現している．スタックの制作者は，これらの閲覧操作に関する機能や，画面サイズを意識することなく，携帯電話向けコンテンツを作成することができる．

6. ま と め

携帯電話におけるいくつかの制約は，携帯電話向けコンテンツを作成する際に課題となる．本研究では，ファイルサイズ，互換性，および閲覧性に関する制約を解消し，携帯電話向けコンテンツ作成システムを構築する．これらの制約を解消するために，カードモデルを提案する．カードモデルは，コンテンツのレイアウトを設計する作業と，コンテンツのデータを入力する作業を分割する．レイアウトは，背景画像の作成と，データを入力するためのオブジェクトの作成によって設計する．この作業で作成されたレイアウトに対し，個々のオブジェクトにデータ入力をすることで，カードと呼ばれる Flash コンテンツを作成することができる．また，作成した複数のカードを結合し，操作機能を付加することで，スタックと呼ぶ 1 つの Flash コンテンツにコンパイルする．

本システムは，自動的にスタックを生成するための MobileStackML を用いて外部アプリケーションと連携が可能である．定められた仕様に従って MobileStackML を記述することで，外部のアプリケーションはスタックを生成する機能を利用できる．MobileStackML は，特に，Web ページからの情報抽出が可能なラッパーとの親和性が高く，レシピサイトから料理データのスタックを生成することも容易に実装できる．

カードモデルは，本研究の目的であるファイルサイズ，互換性，および閲覧性に関する制約を解消する．1 枚当たりのカードのファイルサイズは 10KB 程度であり，スタックに含むカード枚数を任意に変更可能であることから，ファイルサイズの制約を解消している．作成したスタックは，Flash Lite 1.1 でコンパイルされていることから，現在普及している多くの携帯電話での動作が保証されている．また，閲覧性に関して，カードの大きさは携帯電話の画面に適合するサイズで作成されているため，制作者が閲覧性の問題を意識することなくカードを作成することができる．

文　献

[1] Bruce J. Nelson: “Remote Procedure Call,” PhD thesis, Carnegie-Mellon University, 1981.

[2] 大囿忠親, 柿元宏晃, 佐野博之, 平田紀史, 新谷虎松: “携帯電話における情報閲覧支援のための情報編纂システムについて,” 第 7 回情報科学技術フォーラム講演論文集 (CD-ROM), 2008.

[3] 柿元宏晃, 佐野博之, 平田紀史, 大囿忠親, 新谷虎松: “カードモデルに基づく情報編纂システムを利用したレシピ検索システムの試作,” 第 7 回情報科学技術フォーラム講演論文集 (CD-ROM), 2008.

[4] 佐野博之, 柿元宏晃, 平田紀史, 大囿忠親, 新谷虎松: “携帯電話向け情報編纂システムのためのパソコン用書類変換機構の試作,” 第 7 回情報科学技術フォーラム講演論文集 (CD-ROM), 2008.

[5] 平田紀史, 柿元宏晃, 佐野博之, 大囿忠親, 新谷虎松: “携帯電話向け情報編纂システムのためのコンテンツ作成システムの試作,” 第 7 回情報科学技術フォーラム講演論文集 (CD-ROM), 2008.