モバイル環境向けアニメーション技術の実装と評価

水口 充　西畑 実　高倉 正樹　廣嶋 規

シャープ株式会社技術本部

携帯電話やPDAなどを対象としたモバイルブロードバンドが脚光を浴びているが、コストやインフラの整備等の課題があり、膨大な量のデータを扱うことはまだ難しい。この問題に対し、データにはベクトル図形で表現されたキーフレームのみを含めておき、再生時にキーフレームを補間して表示フレームを生成するアニメーション方式を実装した。本方式は小容量で長時間のコンテンツを記述できるので、伝送コストを低く抑えることができるという特長を持つ。予めすべての表示フレームをデータに含めておく方法と比べると再生時の処理量は多くなるが、評価分析の結果、データ量が小さい場合には処理能力が低い機器でも動きの品質に遜色なく再生が可能であることが明らかになった。

Implementation and Evaluation of an Animation Technique for Mobile Environment

Mitsuru Minakuchi, Minoru Nishihata,
Masaki Takakura, Tadashi Hiroshima

Sharp Corporation

Mobile broadband for cellular phone and PDAs is getting popular, however, it is still difficult to handle with big amount of data because of cost and infrastructure issues. This paper proposes a new animation solution, that the transmission data consist of several key frames drawn as vector figures, and the interpolation generates all the final frames to be displayed in realtime. This method reduces transmission cost because of its small size of data for its long time contents. It may need processing for playing back compared with the method which needs all the frames to be displayed beforehand. The evaluation shows that even a low-end device can play back any contents without reducing the quality when the size is small enough.

1　はじめに

近年、携帯電話やPDA(Personal Data Assistants)などの携帯端末機器の性能が向上し、マルチメディアデータをモバイル環境で扱うことが可能になったが、以下のような課題がある。

通信路の制約 現在のブロードバンドの通信路においても、まだ実際的に通信可能なデータ量は制限されている。この制約は緩和されていくであろうが、扱うマルチメディアデータも増大化すると予想され、インフラ整備のコスト等も考慮するとサービスにとって通信路が十分に広いという状況にはなかなか至りにくい。

ハードウェアの制約 携帯端末機器のバッテリー消費を抑え連続使用時間を長くするためには処理量を小さく抑えなければならない。また、処理能力の高いCPUやDSP、大量のメモリを搭載することはコストの面でも不利である。

多様な端末への対応 モバイル環境で利用される端末装置は多様なものがある。このため、複数の品質のデータを準備しておいて、端末装置ごとに適した品質のデータを利用する方法が多く採られている。しかし、コンテンツの作成はコストがかかる作業であるため、できるだけ同一のマルチメディアデータを多くの端末装置で利用可能にすることが望ましい。

上記の課題に対し、本稿では、マルチメディアデータとしてアニメーション[1] を選び、キーフレーム補間によるアニメーション方式 [1] を携帯電話や PDA 向けに実装し、上述のような制約下での有効性を検証したので報告する。

2 関連研究

図形を補間することによるモーフィングアニメーションの研究は古くから数多くなされているが [2] [3]、対応付けられていない図形を自然にモーフィングするなどの、コンテンツの作成の手間を簡略化することを主眼としていた。補間生成されたフレームはキーフレームと併せてフィルムに記録するというように、一般的な動画と同じ型式で扱うことを前提としており、作成されたデータ自体を伝送することは想定していない。このため、補間のための処理量には重点が置かれていなかった。

これに対し、本方式はデータを伝送することを目的としており、再生時にフレームを補間生成することを特長としている。このためには補間の処理はできるだけ単純かつ高速であることが望ましい。そこで、図形の対応付けはデータの作成時に明示的に行なうことにした。このようなデータの作成方法は煩雑でありコンテンツ作成者の負荷となる。我々はコンテンツ作成ツールのユーザインタフェースを工夫することによってこの問題を解決している。

また、一般的なアニメーションを扱うデータ形式には、MPEG やアニメーション GIF などの一般的な動画形式や、Macromedia 社の Flash[4] がある。

前者の一般的な動画形式は複数のフレーム画像から構成されており、空間的および時間的な類似性を利用してそれぞれのフレーム画像のデータを圧縮している。しかし、MPEG 形式は圧縮率を高くすると、アニメーションのような同一色による塗潰し領域と線とから構成される画像に対してはエッジ部分のノイズが目立つという問題がある。アニメーション GIF は可逆的な圧縮方式であるため画質の低下はないが、圧縮率は高くない。また、256 色までしか表現できないため表現力に限界がある。

Flash では個々のフレームはベクトル図形で表現されており、本提案の方式と類似しているが、形状を補間して生成されたすべてのフレームは予めデータに含まれている。Flash 方式との違いについては評価のセクションで考察する。

[1] 本稿では 2 次元の手書きやコンピュータグラフィクスによる動画などの、いわゆる「アニメ」のことを指す。

3 実装

本方式のアニメーション（e-アニメータ）は以下の特長を持つ。これらの特長によりデータ量を小さくして伝送を容易にすることができる。

- キーフレームはベクトル図形で表現する。
- データには表示され得るすべてのフレームを含めるのではなく、キーフレームのみを含める。
- 再生時には表示するフレームをリアルタイムにキーフレームから補間して生成する。

一般的なアニメーションは塗りつぶし領域と線から構成されているので、キーフレームを線分、曲線、多角形などのベクトル図形で表現することによってデータ量を小さくすることができる。また、拡大／縮小が可能であるため、スクリーンサイズに依存せずに表現の品質を保つことができる。

また、従来のコンピュータを利用したアニメーションの作成技術ではデータの作成時にキーフレームの補間を行って表示されるフレームを生成し、すべてのフレームをフィルムなどに記録していたのに対し、本方式はキーフレームの補間を再生時に行うようにしている。

図 1 は本方式によるコンテンツおよび再生の例を示している。アニメーションデータは複数のキーフレームから構成されており、それぞれのキーフレームに含まれている図形間の対応やキーフレーム間の再生時間などの情報が合わせて記録されている。

このデータは次の手順を繰り返すことによって再生される。

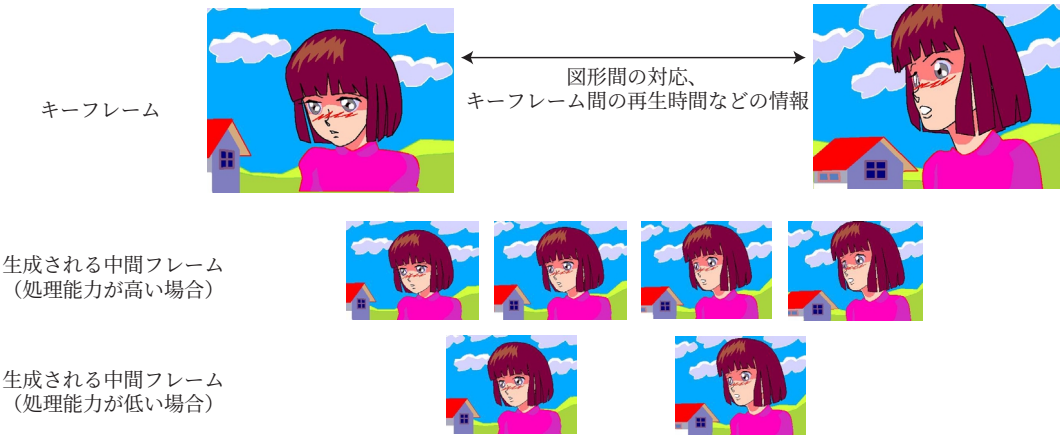

図 1: コンテンツおよび再生の例

1. 再生開始時からの経過時間を取得する。
2. 経過時間の前後の再生時刻に相当するキーフレームを参照する。
3. キーフレームに含まれる図形を経過時刻に応じた比率で線形補間して中間フレームを生成する。
4. 生成された中間フレームを表示する。

3 の線形補間の処理は具体的には、図形を構成している、キーフレーム間で対応する点の座標を次式のように直線的に内挿する（図 2）。

$$x = \{x_n(t_{n+1} - t) + x_{n+1}(t - t_n)\}/(t_{n+1} - t_n)$$
$$y = \{y_n(t_{n+1} - t) + y_{n+1}(t - t_n)\}/(t_{n+1} - t_n)$$

ここで、(x, y) は内挿される点の座標、t は再生開始時からの経過時間、(x_n, y_n) は前のキーフレームの点の座標、t_n は前のキーフレームの再生時刻、(x_{n+1}, y_{n+1}) は後のキーフレームの点の座標、t_{n+1} は後のキーフレームの再生時刻、である。

また、曲線の補間も通過点列の座標を内挿することによって行なうことができる。更に、補間処理は点の座標以外にも、色や線の太さなどの図形の属性にも同様に適用することが可能である。

補間処理に要する時間は再生を行なう機器の処理能力に依存するので生成されるフレームの数は異なるが、表示される時刻に応じた内容のフレームを生成することによって実時間性を保つことができる。中間フレームの生成は再生時に行なわれるため、ある程度の処理能力が要求されるが、一般的な動画のデコードの処理に比べると処理量は小さくて済む。

また、我々はコンテンツデータを作成・編集するためのツールを開発した（図 3）。図 4 はデータ作成の基本の流れである。あるキーフレームをベクトル図形で描画し、変形したい図形を次のキーフレームにコピーする (1)。そして、図形を移動させたり (2)、図形の頂点を移動させる (3) などの変型を施す。このようにして作成されたデータは、図 2 で説明したようにして、別途設定されたキーフレーム間の時間のアニメーションとして再生される。作成ツールの詳細については本稿では説明を省略するが、製品版が入手可能である [5][6]。

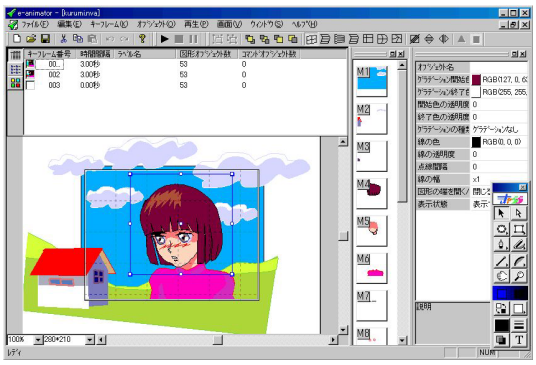

図 3: コンテンツ作成ツール

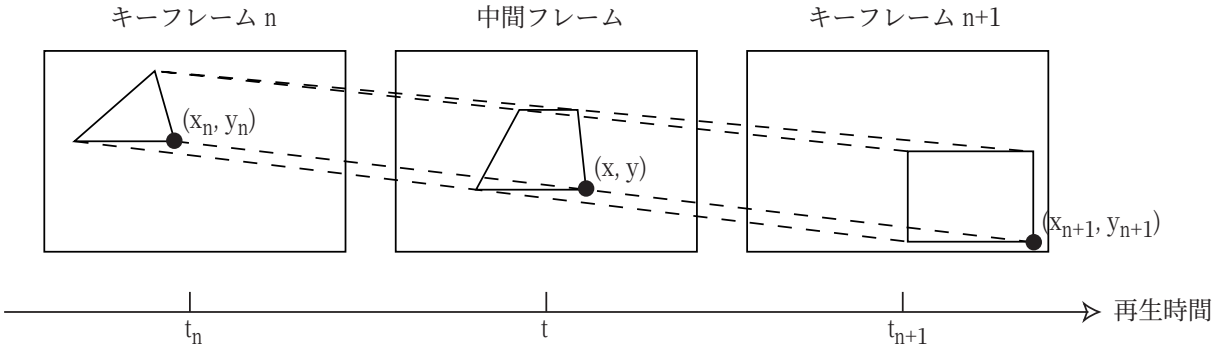

図 2: 補間方法の模式図

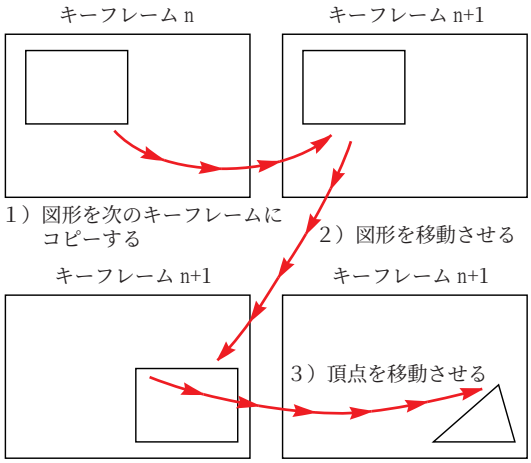

図 4: データ作成の基本の流れ

4 評価

本方式の最大の特長は、キーフレーム補間を再生時に行なうことにある。この特長により、先に述べた Flash のような、キーフレーム補間をコンテンツ作成時に行ない、生成されたフレームをデータに含めている方式（以後 Flash 方式とする）と比べると、データ量を小さく抑えることができるので、モバイル向けの通信環境での伝送に適している。反面、データを再生するための処理量は Flash 方式よりも多くなる。そこで、これらの効果と影響を比較することにした。

4.1 データ量の比較

データ量はコンテンツの内容に大きく依存するが、同じ内容を表現しているコンテンツであれば、Flash 方式では単位時間当たりのフレーム数、本方式ではデータに含められているキーフレームの数に依存する。これらの関係を整理すると以下のようになる。

Flash 方式 1 フレーム当たりの平均データ量を d_f、データに含まれている単位時間当たりのフレーム数を n とすると、単位時間の再生に要する平均データ量 D_f は $D_f = nd_f$ となる。

本方式 1 キーフレーム当たりの平均データ量を d_e、単位時間当たりの平均キーフレーム数を k とすると、単位時間の再生に要する平均データ量 D_e は $D_e = kd_e$ となる。

本方式のデータ構造を図形データにリンク情報を加えたものであると単純化すると、$d_e = d_f + l$ となる。ここで、l は 1 キーフレーム当たりに含まれるリンク情報のデータ量である。

k は言い換えると、データ全体に含まれているキーフレームの数をデータ全体を再生するために設定された再生時間で割った値である。この値はコンテンツの内容に依存する。すなわち、平行移動のような単純な動きのみで構成されていたり、ゆっくりした動きを表現している場合にはキーフレームを少なくすることができるので k の値を小さくすることができる。逆に、変化の大きい急激な動きを表現するためにはキーフレーム数を大きくする必要があり、k の値は大きくなる。また、本方式では補間処理は線

形のみであるため、回転などの動きを表現するためには幾つかのキーフレームを用いる必要がある。しかしながら、一般的には $k \ll n$ であると言え、また d_f に比べると l は小さいので、$D_e < D_f$ となる。つまり、再生時間が同じコンテンツであれば一般的に本方式の方がデータ量は小さく、また、同じデータ量であれば本方式の方が再生時間の長いコンテンツを表現できる。

4.2 フレームレートの比較

本方式は再生時にキーフレーム補間の処理を行なうので、1 フレームを表示するために要する時間は Flash 方式よりも長くなる。このため、再生機器の処理能力が低い場合は、本方式で再生可能なフレームレートは Flash 方式よりも低くなる。一方、再生機器の処理能力が高い場合、Flash 方式では予めデータに含まれているフレームレート以上では再生できないのに対し、本方式では可能な限りフレームレートを上げて再生することができる。再生機器の処理能力とフレームレートの関係をまとめると以下のようになる。なお、アニメーションの再生に要する処理については、単純化のために、描画のために要する座標変換、ラスタライズ、フレームバッファへの書き込みなどの多様な処理を総合して論じることにし、専用のグラフィクスチップ等による並列化などの効果は考慮しないことにする。

Flash 方式 再生時に表示される単位時間当たりのフレーム数を f_f、1 フレームを描画するために要する平均の演算処理量を a、再生機器が単位時間当たりに処理可能な処理量を C とすると、

$$C \geq na \text{ の場合} \quad f_f = n$$
$$C < na \text{ の場合} \quad f_f = C/a$$

となる。後者は再生の処理が間に合わずに幾つかのフレームをとばして再生している場合を表している。

本方式 さらに、本方式において表示される単位時間当たりのフレーム数を f_e、表示される 1 フレームを生成するために要する平均の補間処理量を i とすると、

$$f_e = C/(a+i)$$

となる。

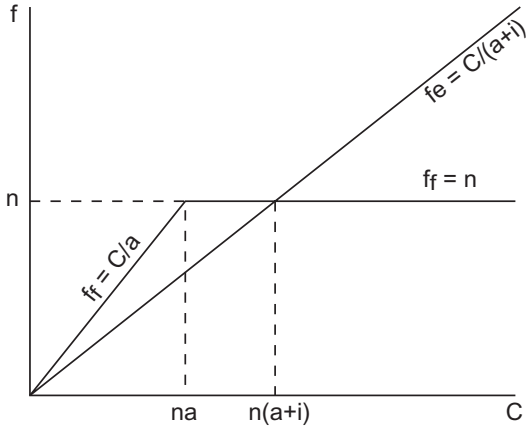

図 5: 処理能力とフレーム数

以上の関係を図示すると図 5 のようになる。この図より、n が小さい程、$C < n(a+i)$ における f_f と f_e の差は小さくなることが分かる。これは、Flash 方式のデータに含まれているフレーム数が少ない程、すなわちデータ量が小さい程、本方式の方が Flash 方式と同等のフレーム数を表示するために要する処理量が多い領域においても、実際に表示されるフレーム数の差は小さくなることを表している。また、その差が最大になるのは $C = na$ の時で、その値は $ni/(a+i)$ である。

4.3 考察

データ量の比較で述べたように、本方式は同じデータ量で表現可能な再生時間が Flash 方式よりも長い。k と n との関係、および d_f と l との関係はコンテンツに大きく依存するため一概には言えないが、幾つかのコンテンツデータで $D_f/D_e = 9 \sim 20$ となっている。

また、通信でデータを伝送するためには、単位時間当たりの最大の通信量 T に対して $T \geq D_e, D_f$ でなければならない。d_f が同じ場合には n の採りうる最大値は T に比例するので、同じ内容のコンテンツを表現するデータにおいて、T が小さい程 Flash 方式での単位時間当たりのフレーム数は制限される。前述のように n が小さい程 $C < n(a+i)$ における f_f と f_e の差は小さいので、現在の携帯電話のような T が小さい場合では、再生機器の処理能力が低くてもフレームレートの差は小さくて済む。逆に、再生

機器の処理能力が高い場合では、本方式ではフレームレートを際限なく大きくして再生することが可能であるので、同じコンテンツデータを多様な再生機器でベストエフォートの品質で利用可能である。これに対し、Flash 方式ではデータ量を小さくするためにフレームレートを低くすると処理能力が高い再生機器では動きの品質が不十分になる。

携帯電話での実装による具体的な数値では、コンテンツの内容や各種条件によって変動するが、おおよそ $a:i=5:1$ となっている。このとき、$f_f - f_e$ の最大値は $ni/(a+i) = n/6$ となる。一般的には秒 10 フレーム程度あればアニメーションとして認識されるが、$f_f = 10$ の時に $f_e = 8.3$ となる。このように、本方式では最悪の条件下でも、少しぎこちない動きになるものの十分アニメーションとして再生することが可能である。

5 まとめ

キーフレームをベクトル図形で表現し、再生時には表示するフレームをリアルタイムにキーフレームから補間して生成するアニメーション方式について有効性を評価した。その結果、モバイル環境においてもコンテンツを伝送することが容易であり、処理能力が低い端末装置に対しても予めフレームデータが含まれている方式と比べて遜色のない動きの品質を提供することができることが分かった。

我々は本方式を PDA および携帯電話に実装し、コンテンツデータのダウンロードサービスなどの運用を行なっている。携帯電話向けには 6k バイトの容量で 5〜20 秒の再生時間となるデータを配信しており、従来のアニメーション GIF による壁紙に比べて、音や画像を含めた豊かな表現のコンテンツを扱うことができることを実証している。

今後の課題としては、データの作成をより容易にすることが挙げられる。現在、PC 上で動作する編集ソフトを提供しているが、ユーザが慣れるまで時間がかかるという問題が判明している。しかしながら、小中学生などを対象とした利用実験では比較的短時間で習熟できることが観察されている。このことから、本手法がパラパラアニメのような従来の概念とは異なったアニメ作成手法であり、この概念により適している編集方法を提供する必要があると考えている。

また、現在はメールに添付して送受信を可能にしているが、アニメーションをコミュニケーション手段として利用するために、携帯端末上でのアニメーションの作成を可能にすることを検討している。このための簡便な操作方法を開発することも課題である。

謝辞

再生プログラムおよびオーサリングツールの実装を行なったアニメーション開発メンバーに感謝致します。

参考文献

[1] 西畑実, 高倉正樹. ベクトルアニメーションシステム EVA, 第 13 回 NICOGRAPH/MULTIMEDIA 論文コンテスト論文集, pp.22–29, 1997.

[2] T. W. Sederberg. A Physically Based Approach to 2-D Shape Blending, SIGGRAPH'92 Coference Proceedings, pp. 25–34, 1992.

[3] M. Alexa, D. Cohen-Or, D. Levin. As-Rigid-As-Possible Shape Interpolatioin, SIGGRAPH2000 Conference Proceedings, pp. 157–164, 2000.

[4] http://www.macromedia.com/software/flash/

[5] http://www.sharp.co.jp/sc/excite/evademo/evahome.htm

[6] http://anime.galamo.com/eanime/servlet/eanime.Page/menu