Non-Photorealistic Walkthroughs Using Flash

Roman Ženka
zenkar1@fel.cvut.cz

Pavel Slavík
slavik@cslab.felk.cvut.cz

Department of Computer Science and Engineering
Faculty of Electrical Engineering, Czech Technical University
Karlovo nám. 13, 121 35 Prague, Czech Republic

ABSTRACT

We describe a method allowing users to walk through 3D scenes using Macromedia Flash player. The method is optimised for low data transfer rates and low demands on the displaying device, which makes it suited especially for mobile devices. The walkthroughs are partially pre-computed on a server and partially calculated directly on the PDA. Non-photorealistic rendering is used for more eye-pleasing results and for achieving higher data compression.

Keywords

Flash, 3D walkthroughs, non-photorealistic rendering, mobile computing, user interfaces

1. INTRODUCTION

3D graphics on the World Wide Web has relatively long history and several solutions are available nowadays. Let us mention VRML/X3D; relatively recent ViewPoint technology [Vie03a]; QuickTime VR [Che95a] as a few examples.

These technologies were designed with desktop computers in mind. Our goal was to find a method to be used on PDA devices, which are known to have limited resources compared to desktop computers.

From a variety of options we had, we decided to choose Macromedia Flash – a tool for rendering interactive 2D data, supported by browsers on a wide variety of platforms. We hoped that a simple 2D viewer could be faster on a slow device than a fully 3D application.

This article describes our experiences with interactive 3D graphics in Flash as well as methods to make this combination possible and useful.

2. PREVIOUS WORK

Since the demand for interactive 3D graphics in Flash was large, several solutions were already proposed and many commercial tools exist. These include various libraries for creating interactive animations, as well as plugins for calculating Flash animations from classical 3D. As relatively big and new solutions for producing 3D presentation in Flash we should mention Swift3D [Swf03a] by Electric Rain and Plasma [Pla03a] by Discreet. Our work differs from the commercial tools by utilization of non-photorealistic rendering. Instead of producing pre-computed movie clips, as those we can often see on the internet, we generate the animations dynamically.

3. 3D IN FLASH

The Flash player allows rendering of 2D vector graphics stored in binary SWF format. Flash also supports animations and interaction. The player also contains an ActionScript interpreter for simple scripting.

These features allow several approaches for displaying 3D data in Flash:

Fully Pre-rendered Movie Clips

It is possible to render a 3D scene into vector images and transfer these data as a movie. Each frame of the movie is transferred separately.

Key Frames and Shape Blending

Shape blending between key frames allows reducing the animation size. Flash unfortunately allows only linear interpolation of shapes. Complex transformations have to be approximated by piecewise linear ones.

Client-side Calculations

Flash's scripting language allows writing a simple 3D rendering engine, which runs on the client side. This rendering is unfortunately extremely slow, so it is useful only for very simple 3D scenes containing very few polygons (below 20 in our experience).

Which Technology to Choose?

The fastest solution is to transfer already calculated frames. The speed gain however causes the amount of transferred data grow rapidly.

Since client-side calculations can be used for extremely simple scenes only (which might be often the case, as we show further). Wise utilization of the morphing algorithm generally leads to best results.

4. NAVIGATION

The 3D scene often has to allow interactive navigation. For PDAs, we developed two ways:

Tap and Go

The users tap the screen at a place they desire to walk towards. The server responds by sending an animation of the user walking towards the desired place, using optimal path connecting the two places. The path does not have to be shortest, it should rather be most visually pleasing and informative.

Cursor Keys Navigation

PDA devices usually possess four cursor keys. If these keys have to be used for navigation, all calculations have to be performed directly on the PDA, which is often not feasible.

Combined navigation

The best solution is to combine both approaches into one hybrid method of navigation. There are two ways how to mix pre-computed data and data calculated interactively during the interaction with the user:

4.1.1 Constraining user movement

If we limit the movement of the user, we can transfer only pre-calculated data for such motion. Navigating then reduces to controlled playing of a simple animation. This can be used e.g. for navigating through narrow corridors, where only moving forward and backward is possible.

4.1.2 Free movement

Free movement is possible within a simple scene only. Complicated scenes can be explored in lower level of detail, becoming more detailed when the user stops moving.

5. RENDERING

When rendering, it is necessary to sacrifice certain amount of scene detail that is difficult to describe as vectors. To simplify the scene, we use methods of non-photorealistic rendering, namely toon shading and hatching.

Toon Shading

A toon shader is basically a common shader with extra quantization step. Since the shading uses just discrete levels, the shaded image can be turned into polygons easily (see Fig. 1)

A sweep-line algorithm similar to polygon filling can do the necessary calculation without needing to create a bitmap that is subsequently vectorized.

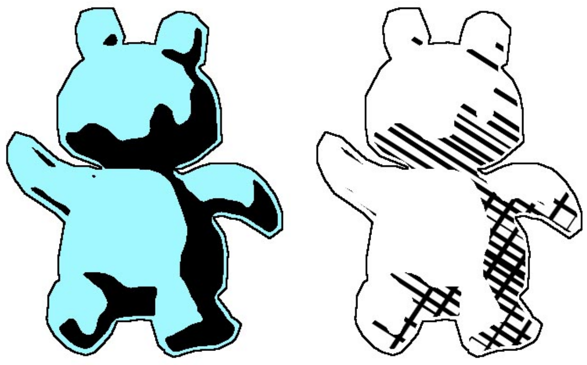

Figure 1: Toon shading and hatching

Hatching

Instead of filling the regions, the silhouette and a few hatching strokes can be used. The paper [Sou03a] by Mario Sousa describes a method for reducing the lines needed for conveying the shape and shading.

Panoramic images

Flash does not natively support any method allowing display of spherical or cylindrical panoramas. It is however possible to transfer pseodo-3D data and calculate the deformations on the client side. Rotating a panoramic image requires fewer calculations, than if full 3D data were used for same rotation.

A method described in [Woo97a] allows calculation of panoramic images for complex camera trajectories. These panoramas are perfect for Flash, since they offer sufficient quality and very high data compression.

6. THE SWF FORMAT

In order to write an effective application, we have to know in detail how the SWF format works. A full description can be found at [Mac02a], here we mention just the parts which are of a special interest for us.

The SWF format consists of tags – small blocks of data of given type. There are two main types of tags:

- Definition tags
- Display tags

The *definition tags* define new objects and store them into a dictionary.

The *display tags* reference the objects already defined by *definition tags* and instruct the player to display them at given position and depth.

Separation of definition and display tags allows object reuse as well as streaming.

Shape definition tag

Flash supports filled and stroked shapes consisting of straight-line segments and quadratic Bezier curves. Even a circle has to be approximated using Beziers.

The coordinates are stored as deltas from last position. Coordinate units are twips (1/20th of a pixel). Care is taken to store only as many bits of the coordinate as necessary.

Since 1/20th of a pixel is too high precision for rough renderings, the shapes should be rendered at cca 1/32nd of their size and displayed using a scale matrix (see below). 5 bits can be saved per each coordinate. To store the additional matrix we need 26 extra bits. It is easy to see that this approach will pay off for more than 5 deltas stored, which equals to three line segments.

Morph definition tag

Two shape definition tags together form a morph definition. Memory consumption is nearly identical to two separate shapes; the only extra memory needed is a specification of the *ratio* field in the object placing tag, which takes 2 bytes. Morph definition pays off when at least three frames are defined using it.

Display tags

The objects can be placed on the screen using an affine transform specified by a matrix. A typical display tag with transformation matrix fits into 12 bytes, without a transformation just 6 bytes suffice.

Compression

The data within the SWF file can be compressed using zlib to achieve even smaller file sizes.

7. REDUCING DATA SIZE

Except low-level methods for compression, offered by the format, there are many other ways of lowering the amount of data transferred:

Object reuse

Once an object is transferred to the Flash player, it can be subsequently referenced many times from within the file. If we manage to find a similarity in objects we transfer, we can save a lot of bandwidth. Symmetry of the scene, objects placed several times within it, repetitive animation – all these factors can be utilized.

Reuse of an object does not pay off only for separate line segments.

Lowering data precision

Flash stores data of lower precision using lower amount of bits. Rounding of object coordinates can save bandwidth, but it degrades the image quality.

Non-photorealistic rendering can be used to simulate the imprecision of human-drawn images. Users tend to forgive not only rounding errors, but often also missing lines or other artefacts, in case this kind of rendering is used.

Approximating the animation

Perspective transformation of moving objects has to be approximated using several affine transformations. We chose the precision of approximation according to moving object's importance (function of object's size and color). Non-photorealistic rendering allows small artefacts caused by approximation without disturbing the user.

8. STREAMING

Streaming is used to provide immediate response to user actions. We mix animation data and high-quality picture of the final animation frame as shown at Fig. 2. Although same data are transferred, the user does not experience so long delays.

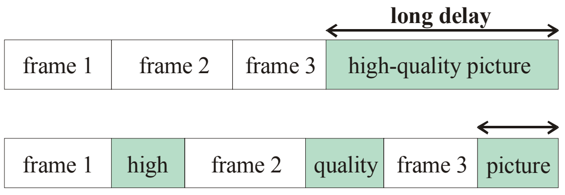

Figure 2: Reducing the delay by streaming

9. RESULTS

We did several experiments to determine the usefulness of our proposed method. The tests were performed on HP iPaq 2215 equipped with 400MHz ARMv4 processor.

Speed

We did several tests for determining the speed of rendering on the PDA.

Our tests clearly indicate that we have to limit the scene complexity severely in order to achieve decent framerates. The speed of Flash on a PDA was somewhat disappointing to us, we can however compare it to the speed of Flash on a desktop computer in full screen mode.

Test	Quality		
	Low	Medium	High
16 x 4-vertex poly.	38fps	33fps	29fps
100 lines	26 fps	13fps	7fps
3191 triangles	9.5fps	6.6fps	3.4fps

Table 1: Speed of Flash on a PDA

Data Size

We tested the size of resulting Flash movie without compression. Zlib format, as described in RFC 1950 can be optionally used for encoding the entire SWF file contents. The compression usually ranges from 10% - 70%. The mentioned 100 lines test was compressed only by 10%, the polygons by 30%.

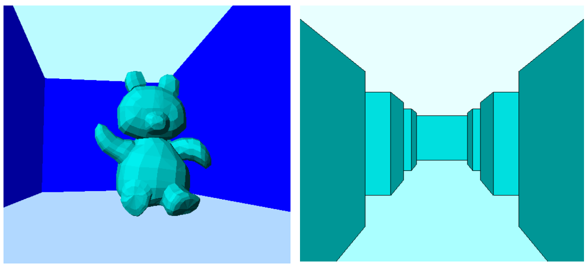

Figure 3: Testing walkthroughs

For testing we used two walkthroughs (see Fig. 3). Both walkthroughs had 100 frames. We were able to reduce the size of the data file about 100 times (from 4MB to 40KB) in the first scene (the reduction was caused mainly by object reuse, toon shading and shape blending). The simpler scene did not allow so many opportunities the data were reduced only 4 times (46KB to 11KB).

10. USE CASES

Our technology can be used anywhere the user has to interact with large 3D environment which cannot be downloaded all at once – city or building tours, cheap augmented reality, simple virtual worlds. Interaction and collaboration is also possible in Flash.

11. CONCLUSIONS AND FUTURE WORK

By simplifying the animation we achieved small file sizes and fast rendering, making our method suitable for being used on a PDA.

In the future we would like to implement smarter shading and shadow calculation for more eye pleasing results.

Since the scenes are transmitted in non-photorealistic way, it would be very interesting to create a non-photorealistic authoring tool for them. Using a tool for modelling the scene by sketching is being considered.

12. ACKNOWLEDGMENTS

This project has been partly supported by the Ministry of Education, Youth and Sports of the Czech Republic under research program No. Y04/98: 212300014 (Research in the area of information technologies and communications) and under FRVŠ grant No. 2477 TO G1.

We used a model created using Teddy [Ted99a] in our testing scenes.

13. REFERENCES

[Sou03a] Mario Costa Sousa and Przemyslaw Prusinkiewicz. A Few Good Lines: Suggestive Drawing of 3D Models, Eurographics 2003 Conference Proceedings, pp. 381-390, 2003

[Vie03a] Viewpoint Media Player, Viewpoint Corporation, http://www.viewpoint.com

[Swf03a] Swift3D Modeller, Electric Rain, http://www.swift3d.com

[Pla03a] Plasma, Discreet http://www.discreet.com/ /products/plasma/

[Mac02a] Macromedia Flash (SWF) File Format Specification, http://www.macromedia.com

[Ted99a] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A sketching interface for 3D freeform design. SIGGRAPH 99 Conference Proceedings, pp. 409-416, 1999

[Woo97a] Daniel Wood, Adam Finkelstein, John Hughes, Craig Thayer, and David Salesin. Multiperspective panoramas for cel animation. SIGGRAPH 97 Conference Proceedings, pp. 243-250, 1997

[Che95a] Shenchang Eric Chen. Quicktime VR – an image-based approach to virtual environment navigation. In *Siggraph 95 Conference Proceedings*, pp. 29-38, 1995