ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Lê Viết Sơn

XÂY DỰNG ĐỒ HỌA 3D DỰA TRÊN CÔNG NGHỆ FLASH DÀNH CHO THIẾT BỊ NHÚNG

PHƯƠNG PHÁP XỬ LÝ 3D CỦA PAPERVISION3D

KHOÁ LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Công Nghệ Thông Tin

HÀ NỘI - 2010

Lời cảm ơn

Trước tiên, em muốn gửi lời cảm ơn sâu sắc nhất đến hai thầy giáo PGS-TS Nguyễn Việt Hà và Th.S Vũ Quang Dũng. Các thầy đã tận tình hướng dẫn chúng em trong học tập, đồng thời tạo điều kiện trong công việc nghiên cứu tại phòng thí nghiệm Toshia-Coltech suốt hai năm qua.

Chúng em xin bày tỏ lòng cảm ơn đến những thầy cô giáo công tác, giảng dạy tại trường đại học Công Nghệ - Đại học Quốc Gia Hà Nội. Những kiến thức, phương pháp qúy báu thầy cô truyền đạt sẽ là hành trang giúp chúng em vững bước trong tương lai.

Tôi xin cảm ơn các thành viên phòng thí nghiệm Toshiba-Coltech, các bạn đã cho tôi nhiều ý kiến qúy báu khi thực hiện khóa luận.

Cuối cùng con xin gửi tới bố mẹ và toàn thể gia đình lòng biết ơn và tình cảm yêu thương sâu sắc nhất.

Hà Nội, 19 tháng 5 năm 2010

Sinh viên

Lê Viết Sơn

Tóm tắt nội dung

Ngày nay, thiết bị di động trở nên phổ biến và mang lại nhiều tiện ích trong cuộc sống. Với sự phát triển của đồ họa máy tính, giao diện các ứng dụng ngày càng đẹp mắt và thân thiện với người dùng. Đồ họa ba chiều được sử dụng rộng rãi trên máy tính cá nhân hay những thiết bị có cấu hình mạnh và mang lại hiệu quả to lớn. Tuy nhiên, thiết bị nhúng có cấu hình thấp hơn bởi vậy chúng ta cần phải xây dựng hệ thống phù hợp để đáp ứng được nhu cầu về tốc độ xử lý và hiển thị.

Khóa luận tập trung tìm hiểu phương pháp xử lý đồ họa 3D cho thiết bị di động.

Mục lục

Danh sách hình vẽ

Bảng từ viết tắt

Ký hiệu	Dạng đầy đủ
2D	2 Dimension
3D	3 Dimension
ABC	ActionScript Bytecode
AVM	ActionScript Virtual Machine
AS	ActionScript
JIT	Just-In-Time
MMgc	Memory manager, garbage collector
OpenGL	Open Graphics Library
OpenGLES	OpenGL Embeded Systems
SWF	Small Web Format
VM	Virtual Machine

CHƯƠNG 1

Đặt vấn đề

1.1 Thực trạng

Ngày nay các thiết bị điện tử trở nên phổ biến và mang lại nhiều lợi ích trong cuộc sống. Từ những hệ thống công cộng, dịch vụ tự động như bán vé tàu xe cho đến các thiết bị trong gia đình như tivi, máy giặt, tủ lạnh thậm chí đến từng cá nhân như điện thoại di động, PDAs[1]. Với sự phát triển của công nghệ, ứng dụng đồ họa trên trên các thiết bị nhúng ngày càng đòi hỏi cao về tốc độ xử lý, khả năng tương tác nhanh và đáp ứng được nhu cầu của người dùng. Đặc biệt, sử dụng đồ họa 3D cho các ứng dụng mang lại nhiều hiệu quả về tính thân thiện, dễ sử dụng, và đẹp mắt. Flash là một công nghệ có nhiều ưu điểm để phát triển trên các thiết bị nhúng bởi tính gọn nhẹ, đơn giản, hơn nữa Flash hỗ trợ đồ họa 3D khá tốt. Hình 1.1 minh họa ứng dụng biểu đồ sử dụng phương pháp hiển thị trực quan 3D.

Hình 1.1: Biểu đồ hiển thị trực quan 3D

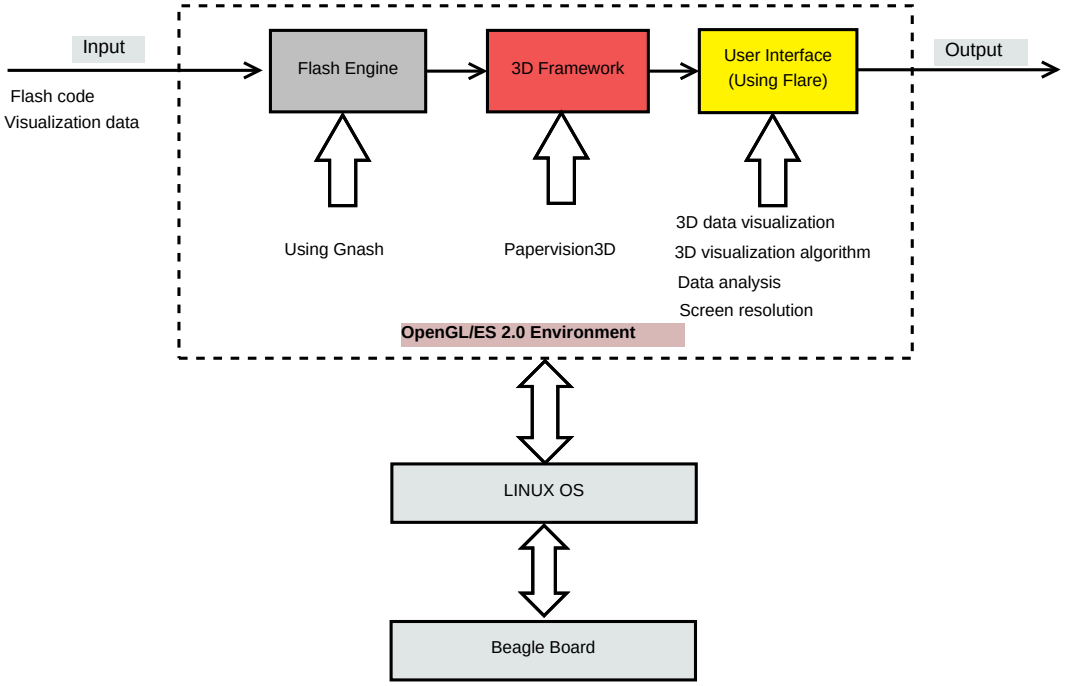
[1] Personal Digital Assitants

1.2 Phạm vi nghiên cứu

Khóa luận này được trình bày trong khuôn khổ dự án xây dựng hệ thống xử lý đồ họa 3D và framework sử dụng các thuật toán xử lý, sắp xếp dữ liệu. Mục tiêu khóa luận là chứng minh được tính khả thi cũng như tính công nghệ áp dụng trong dự án.

Dự án được đề xuất thực hiện trong thời gian 3 với các giai đoạn như sau:

- Giai đoạn một: hướng vào nghiên cứu phát triển *3D Flash* cho thiết bị di động. Sử dụng Gnash làm máy Flash trên *"Beagle Board"*[2] môi trường *Linux* hỗ trợ đồ họa OpenGLES.
- Giai đoạn hai: xây dựng các thuật toán, mẫu xử lý dữ liệu 3D.
- Giai đoạn ba: làm mịn các kết quả ở giai đoạn hai.



Hình 1.2: Tổng quan dự án

Trong giai đoạn đầu tiên, chúng tôi hướng đến phương pháp xử lý đồ họa 2D, 3D Flash dựa vào OpenGLES trên hệ thống nhúng.

[2] http://beagleboard.org/

1.3 Cách tiếp cận

Trước hết, để hiểu được về công nghệ Flash, chúng tôi[3] tập trung vào thành phần liên quan bao gồm: ngôn ngữ AS, trình chơi Flash[4], máy ảo, tệp tin. Tuy nhiên, công nghệ Flash của Adobe khép kín và mang tính thương mại nên gây cản trở cho việc tìm hiểu. Nguồn tài liệu tham khảo rất hạn chế. Do vậy, lựa chọn của chúng tôi là tiếp cận những phần mềm nguồn mở, dựa vào đó để phát triển cho dự án.

1.4 Cấu trúc khoá luận

Các phần còn lại của khóa luận có cấu trúc như sau:

- Chương 2 là cơ sở lý thuyết công nghệ Flash, mô hình và nguyên tắc hoạt động của các dự án nguồn mở chúng tôi quan tâm: Gnash, Tamarin, Papervision3D.
- Chương 3 mô tả bài toán xử lý 3D Flash trên thiết bị di động, đồng thời đề xuất ý tưởng, giải pháp và mô hình cho bài toán đó.
- Chương 4 trình bày về thực nghiệm để chứng minh tính khả thi, đúng đắn cho giải pháp đã nêu trong chương 3.
- Chương 5 tổng kết những gì đã đạt được trong quá trình nghiên cứu, đồng thời đưa ra hướng phát triển trong giai đoạn tiếp theo.

[3] Nhóm sinh viên nghiên cứu tại phòng thí nghiệm Toshiba-Coltech
[4] Flash Player

CHƯƠNG 2

Cơ sở lý thuyết

Để giải quyết bài toán trong khóa luận này, chúng tôi tìm hiểu một số vấn đề về lý thuyết: công nghệ *Flash*, trình chơi Flash nguồn mở - Gnash[1], máy ảo *Tamarin*, *Papervision3D*[2]-thư viện Flash nguồn mở và môi trường đồ họa OpenGLES trên hệ thống nhúng.

Các phần mềm, dự án nguồn mở trong phạm vi dự án có rất ít tài liệu kèm theo. Do vậy, việc tìm hiểu các vấn đề lý thuyết liên quan tới các mã nguồn mở cũng là công việc chúng tôi tự khám phá, và xây dựng lại các mô hình dựa trên những nghiên cứu đó.

2.1 Công nghệ Flash

2.1.1 Giới thiệu

Flash[3] là nền đa phương tiện được phát triển từ ngôn ngữ *C++*, dùng để đưa các ứng dụng hoạt họa, video và tương tác vào các trang *Web*. Flash là công nghệ khép kín, thương mại hóa, được phân phối và phát triển bởi Adobe[1]. Flash thường được sử dụng cho quảng cáo và trò chơi điện tử. Flash dùng kỹ thuật đồ họa véc tơ để tạo hoạt họa. Flash hỗ trợ hai luồng âm thanh, hình ảnh đồng thời xử lý sự kiện với đầu vào là tương tác người dùng thông qua thiết bị ngoại vi.

Các tệp Flash thực thi được trên *Flash Player* ở nhiều nền tảng hệ điều hành *(Windows, Linux, Mac OS)*. Ngôn ngữ tạo Flash là AS[4]

[1]Công ty lớn chuyên về phần mềm xử lý đồ họa

2.1.2 Ngôn ngữ ActionScript

AS là ngôn ngữ theo kịch bản dựa vào ECMAScript, được thiết kế bởi công ty Macromedia[2]. Hiện tại, AS được sở hữu bởi Adobe. Từ năm 1998 đến nay, ngôn ngữ AS đã có ba phiên bản. Phiên bản mới nhất ActionScript 3.0 có nhiều cải tiến và khác biệt hoàn toàn so với hai phiên bản trước đó.

Lập trình ngôn ngữ AS tạo ra được chương trình là các tệp SWF về giải trí, đồ họa thông qua các công cụ phát triển lập trình và bộ biên dịch *asc* [3].

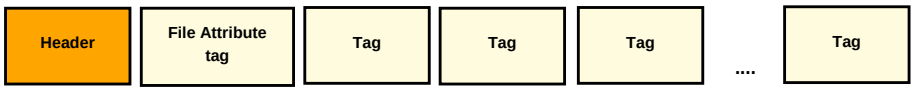
2.1.3 Tệp tin SWF

Tệp SWF[5] có định dạng tệp nhằm đưa các dữ liệu đồ họa véc tơ, văn bản, phim hình và âm thanh lên Internet. Flash Player là phần mềm hỗ trợ chơi tệp SWF, Gnash cũng là phần mềm nguồn mở chức năng tương tự.

Định dạng tệp SWF được thiết kế tiện dụng cho việc truyền dữ liệu qua mạng Internet, không phải cho chuyển đổi đồ họa giữa các trình chỉnh sửa. Phiên bản hiện tại của SWF là 10 tương ứng với Flash Player 10 của hãng Adobe. Tệp Flash có thể được tạo ra từ những công cụ lập trình của Adobe như: Flash, Flex Builder (sử dụng bộ biên dịch *asc*).

Cấu trúc của tệp Flash bao gồm phần Header, các thẻ điều khiển (control tag) và thẻ định nghĩa (Definition tag) như trên hình 2.1

- Thẻ định nghĩa xác định đặc điểm của đối tượng (đặc điểm này lưu trữ trong từ điển).
- Thẻ điều khiển chứa cách điều khiển các luồng, quản lý mặt tổng thể, các khung hình và cách thực thi tệp.
- Trình chơi Flash xử lý tất cả các thẻ của tệp SWF cho đến khi gọi thẻ ShowFrame. Tại thời điểm này, danh sách hiển thị (Display List) được chuyển đến màn hình và trình chơi chờ xử lý khung hình tiếp theo.



Hình 2.1: Cấu trúc tệp Flash

Hình 2.2 minh họa các thẻ trong tệp SWF sau khi được phân tích dưới dạng XML và tách ra dạng ABC. Chúng ta có thể nhận thấy cấu trúc này bao gồm tuần tự các thẻ

[2]Công ty sản xuất phần mềm xử lý đồ họa
[3]ActionScript Compiler

liên tiếp và giá trị. Ví dụ như thẻ tiêu đề (header) có các thuộc tính chính cho tệp Flash là số lượng khung hình, tỉ lệ khung. Thẻ thuộc tính tệp (file attribute tag) chỉ có ở SWF phiên bản 8 trở đi. Ở hình 2.2 thấy thẻ DoABC, thẻ này chứa thông tin về các hằng, thẻ DoABC chỉ có ở AS 3.

Hình 2.2: Minh họa thẻ trong tệp SWF

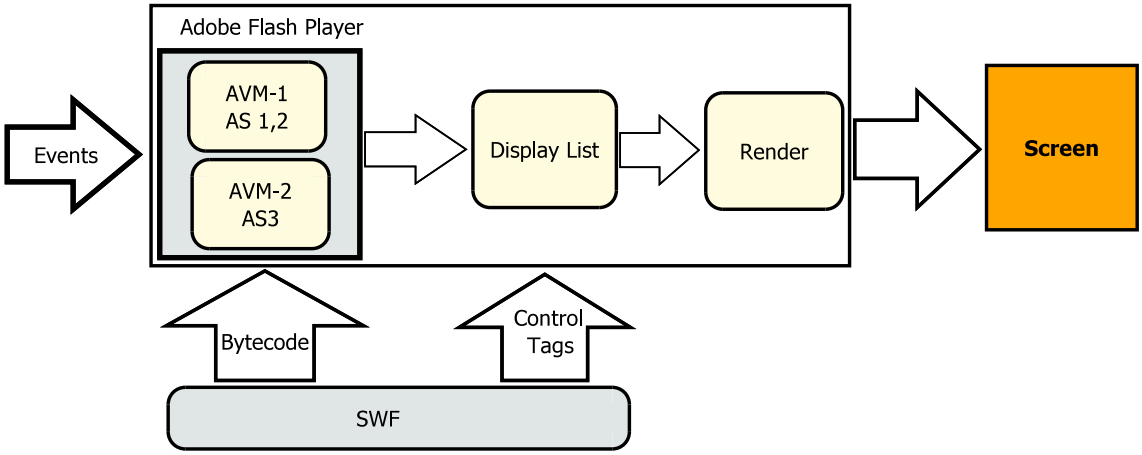
2.1.4 Flash Player

Flash Player [6] là phần mềm chơi các tệp SWF, được sử dụng trong trình duyện Web hoặc cài đặt trực tiếp trên hệ điều hành. Flash Player dùng kỹ thuật hiển thị đồ họa véc tơ để giảm thiểu tối đa dung lượng tệp, sử dụng ít băng thông mạng và giảm thời gian tải tệp. Nhiều trình duyệt hiện nay hỗ trợ việc cài đặt Flash Player làm trình chơi tệp SWF như: Mozilla Firefox, Safari, IE, Opera. Flash Player có nhiều phiên bản tương thích với các nền tảng, kiến trúc khác nhau trên máy tính hay thiết bị di động.

Nguyên tắc hoạt động chung

Từ năm 2006, tương ứng với ngôn ngữ AS 3.0 là Flash Player phiên bản 9. Quá trình thực thi tệp Flash bởi Flash Player 9 dưới hình 2.3 như sau:

- Đầu tiên tệp SWF được phân tích ra làm 2 thành phần ABC và Control Tags (các tag điều khiển).
- Tiếp theo phần ABC được chuyển cho máy ảo AVM thực thi (gồm AVM1 và AVM2), việc xác định phiên bản của ngôn ngữ AS tương ứng với ABC sẽ quyết định giao quyền xử lý cho máy ảo nào.
- Sau đó, các thẻ điều khiển được đưa vào Display List [4].
- Cuối cùng, kết quả xử lý của máy ảo kết hợp với Display List chuyển sang bước vẽ đầu ra cho chương trình.



Hình 2.3: Hoạt động của Adobe Flash Player phiên bản 9

Hoạt động của Flash Player với Papervision3D

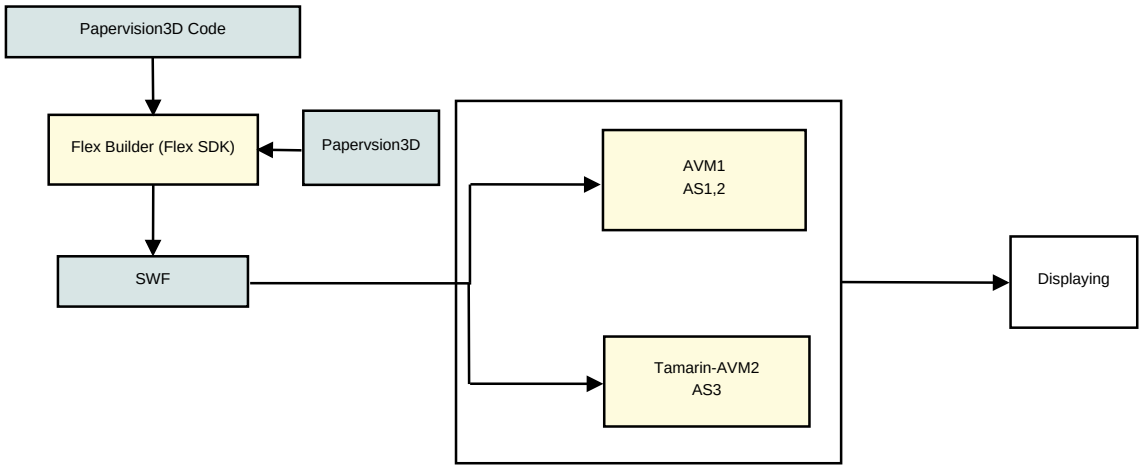
Flash Player hỗ trợ thực thi Papervision3D vì nó có hai máy ảo hoạt động bên trong. Máy ảo thực thi 3D Flash là AVM2. Hình 2.4 minh họa cơ chết hoạt động của Flash Player thực thi 3D Flash - Papervision3D.

2.2 Gnash

2.2.1 Giới thiệu

Gnash là một ứng dụng nguồn mở được phát triển bởi dự án GPLFlash. Phần mềm này thực thi các tệp tin Flash độc lập hoặc là plug-in[5] để thực thi Flash được nhúng vào Web. Chức năng của Gnash cũng như Adobe Flash Player đã giới thiệu ở trên.

[4]Thành phần trong xử lý Flash
[5]Phần mềm cài thêm vào trình duyện



Hình 2.4: Mô hình Flash player thực thi Papervision3D

Gnash được giới thiệu lần đầu tiên vào năm 2005 bởi John Gilmore[1] và đứng đầu nhóm phát triển là Rob Savoye[1].

2.2.2 Mô hình xây dựng

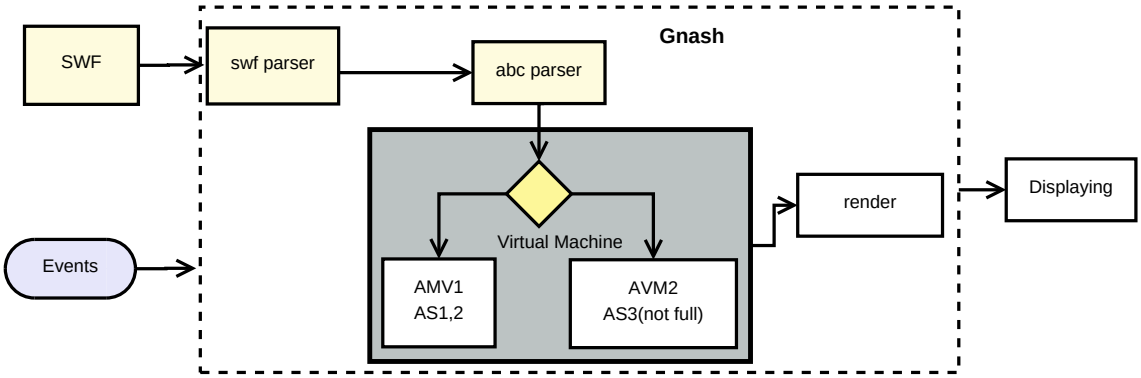
Mô hình xây dựng được đề cập chi tiết trong khóa luận của bạn *Đinh Anh Thái*, người cùng tôi tham gia nghiên cứu trong dự án này. Nhờ những kết quả tìm hiểu về *Gnash* của bạn, tôi mô hình cách xử lý Flash của Gnash ở phần tiếp theo.

2.2.3 Cơ chế hoạt động

Gnash xây dựng hai máy ảo để thực thi ABC. Máy ảo AVM1 hỗ trợ xử lý AS 1.0, 2.0, máy ảo AVM2 chưa hỗ trợ hết cho AS 3.0. Do đó, với 2D Flash hiển thị được được trên Gnash còn Papervision3D và đa số chương trình AS 3.0 là không.

Gnash nhận đầu vào là tệp SWF, sau khi đi qua bộ *swfparser*, tách được thành phần của ABC. Sau đó, các thành phần ABC sẽ được xử lý bởi máy ảo AVM[6]. Cuối cùng, đầu ra của máy ảo kết hợp với phần hiển thị của Gnash là kết quả của chương trình. Hình 2.5 mô tả hoạt động của Gnash

[6] Sẽ được nêu rõ trong phần tiếp theo



Hình 2.5: Cơ chế hoạt động của Gnash

2.3 Tamarin - AVM2

2.3.1 Giới thiệu

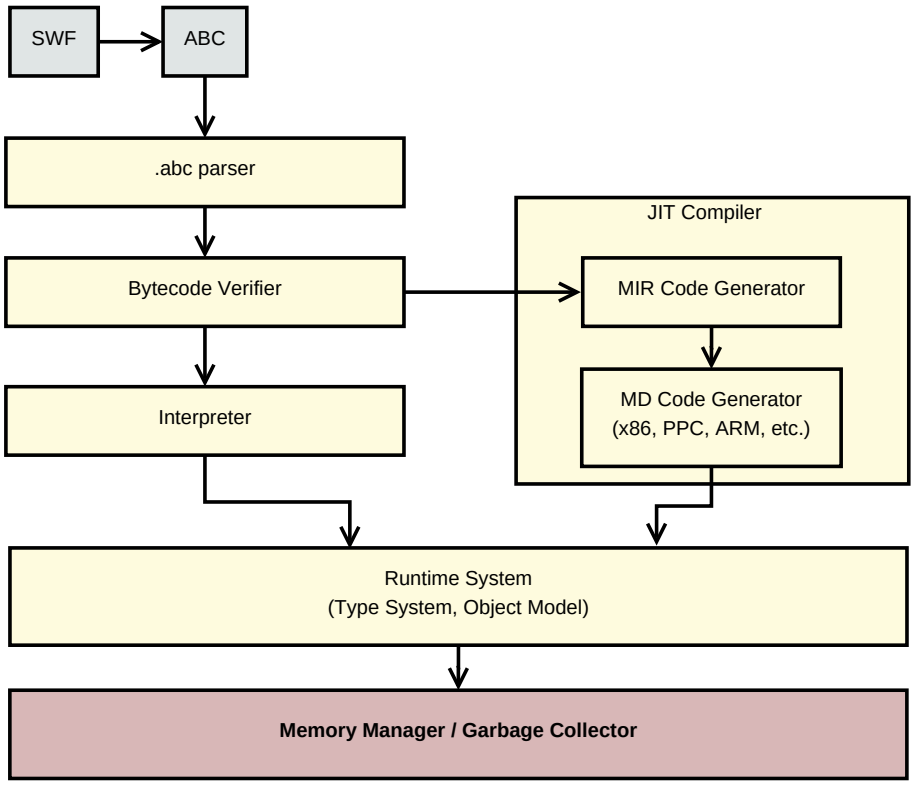
Tamarin là 1 dự án nguồn mở của *Mozilla*[7]. Mục đích của dự án là xây dựng máy ảo AVM2[7] thực thi ngôn ngữ AS 3.0 với hiệu suất cao. Máy ảo Tamarin được sử dụng bên trong phần mềm Adobe Flash Player đã đề cập ở trên, và còn có thể được sử dụng với các dự án khác ngoài Adobe. Mục đích của *Mozilla* là xây dựng tamarin để đưa vào bên trong Mozilla TraceMonkey - máy JavaScript dùng cho trình duyệt Web Mozilla Firefox. [8]

Ngoài ra, mục đích dự án Tamarin là hỗ trợ đa nền tảng phần cứng, bao gồm cả ARM[9]. Cho bộ vi xử lý x64 và hệ điện toán 64 bit với 2 mục tiêu: cải tiến hiệu năng thời điểm thực thi (run-time) và phát triển bộ biên dịch ở thời điểm thực thi (run-time compiler).

2.3.2 Kiến trúc, nguyên lý hoạt động

Máy ảo *Tamarin - AVM2* nhận đầu vào là tệp ABC[10] thu được sau khi phân tích tệp Flash. Bộ *.abc parser* phân tích tiếp tệp ABC để lấy ra những mã bytecode để chyển cho quá trình xác thực tiếp theo *Bytecode Verifier*. Tại đây, những mã *Bytecode* chứa lệnh đơn giản sẽ được bộ thông dịch -*Interpreter* xử lý và những mã chứa lệnh phức tạp sẽ chuyển sang cho bộ biên dịch JIT - *JIT Compiler*. Kết thúc quá trình này, máy ảo sẽ chuyển sang thực thi đối tượng trên hệ thống thật, chuyển toàn bộ các lệnh trên máy ảo sang cho bộ xử lý của máy thật và được quản lý bộ nhớ - *Memory Manager /Garbage Collector*. Các bước thực hiện trên được minh họa ở hình 2.6

[7] http://www.mozilla.org/
[8] Trình duyện Web phổ biến hiện nay
[9] Bộ xử lý cho thiết bị nhúng
[10] có dạng *.abc



Hình 2.6: Kiến trúc AVM

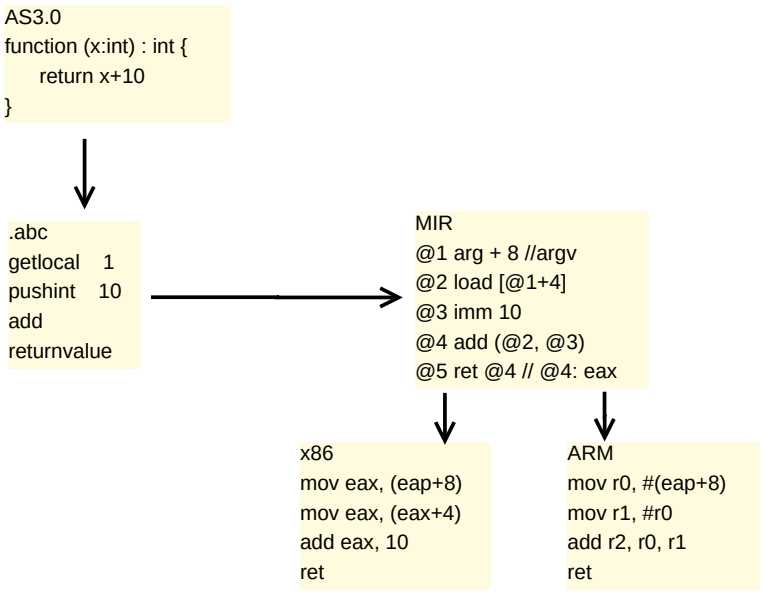
Thực thi tệp Flash qua nhiều giai đoạn, chi tiết như sau. Đầu tiên, từ mã nguồn AS được tạo ra bởi người lập trình, tiếp theo trình biên dịch *asc* của FlexSDK sử dụng và tạo ra mã trung gian có trong tệp ABC và MIR. Các mã trung gian với mục đích tạo ra tính di động cho chương trình Flash mà không phụ thuộc vào nền tảng hệ điều hành khác nhau. Từ các mã trung gian này, bộ biên dịch JIT sinh ra mã thật tùy thuộc vào nền tảng bộ xử lý của phần cứng sử dụng. Ví dụ 2.7 minh họa quá trình chuyển đổi, sinh mã trung gian và mã máy cho hai bộ xử lý khác nhau.

Để hiểu thêm về tập lệnh và kiểu dữ liệu, có thể xem ở phụ lục A

MIR là mã trung gian dùng trong AVM, mã này độc lập với mã máy thật. Ưu điểm của MIR là đơn giản, gần với mã máy vật lý, là thành phần trung gian giữa *bytecode*. MIR được thiết kế để tối ưu quá trình biên dịch giữa mã chương trình với mã máy, do đó việc lập trình trên nền tảng hệ điều hành không bị ràng buộc bởi môi trường.

Tệp ABC được xử lý AVM2 qua bốn bước chính gồm nạp, liên kết, xác thực và thực thi như sau:

- Quá trình nạp, tệp ABC được đọc vào bộ nhớ, giải mã và phân tích;
- Trong quá trình liên kết, một số tên được tham chiếu từ vùng riêng của cấu trúc tệp ABC để xử lý và sau đó trả lại kết quả là một cấu trúc dữ liệu phức tạp hơn rất nhiều



Hình 2.7: Minh họa đoạn mã được thực thi bởi AVM2

liên kết các đối tượng cùng nhau;

- Quá trình xác thực là tương tác giữa đối tượng, đối tượng phải được gọi tường minh;
- Quá trình thực thi, mã bytecode trong tệp ABC được biên dịch trong quá trình tính toán. Việc xác thực với các luồng lệnh xảy ra liên tục: lệnh này không được nằm ngoài mảng bytecode;
- Quá trình xác thực được thực hiện ở tất cả các bước trên. Tại mỗi bước, nếu xuất hiện lỗi, AVM đưa ra thông điệp *Verify Error* để có thể bắt lỗi chương trình;

Trong phần tiếp theo sẽ đi sâu vào cơ chế hoạt động của các thành phần chính mà chúng liên quan đến hiệu suất của máy ảo *Tamarin-AVM2*.

2.3.3 Bộ biên dịch JIT

Giới thiệu

Một hệ thống cần chuyển mã bậc cao hoặc mã bytecode sang mã máy ở thời điểm thực thi, chỉ ngay trước khi gọi đoạn mã đó. Biên dịch JIT[8] là kỹ thuật chính được cài đặt ở hầu hết các máy ảo của Microsoft .NET [11] và Java[12] . Bộ biên dịch này được sử dụng nhiều trong các dự án của *Mozilla* và có tên Naojit[9].

Nanojit là thư viện C++ thực thi đa nền tảng, nhiệm vụ của nó là sinh ra mã máy.

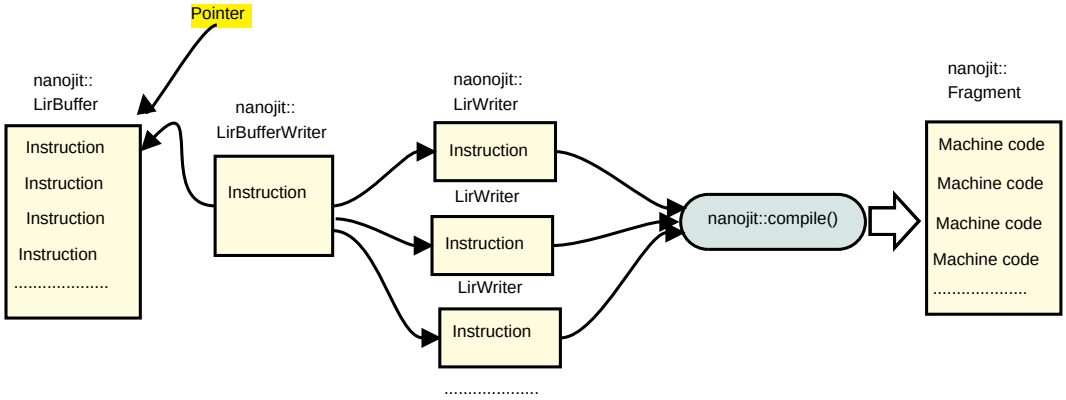
[11] Framework của Microsoft
[12] Công nghệ của Sun Micro System

Đầu vào của Nanojit là (Low-level Intermediate Representation) luồng lệnh mã trung gian bậc thấp nanojit.

Cơ chế hoạt động

Máy ảo tamarin sử dụng Nanojit tạo ra đối tượng nanojit::LirBuffer để lưu lệnh LIR. Ngoài ra đối tượng nanojit cũng được sinh ra để ghi các lệnh lưu ở nano::LirBuffer vào bộ đệm. Sau đó, bộ đệm có thể gói các LibBufWriter vào 0 hoặc nhiều đối tượng LirWriter. Chuỗi lệnh sắp xếp trong đối tượng LirWriter có dạng đường ống. Mỗi LirWriter có thể tối ưu hoặc thực thi nhiệm vụ trong chương trình cho hệ thống và cho LirBuffer.

Mỗi lần với các lệnh trong LirBuffer, AVM2 gọi hàm nanojit::compile() để tạo ra mã máy. Mã máy này được lưu trong nanojit::Fragment. Cơ chế hoạt động của *nanojit* được minh họa ở hình 2.8



Hình 2.8: Hoạt động của Nanojit trong tamarin

2.3.4 Bộ thông dịch

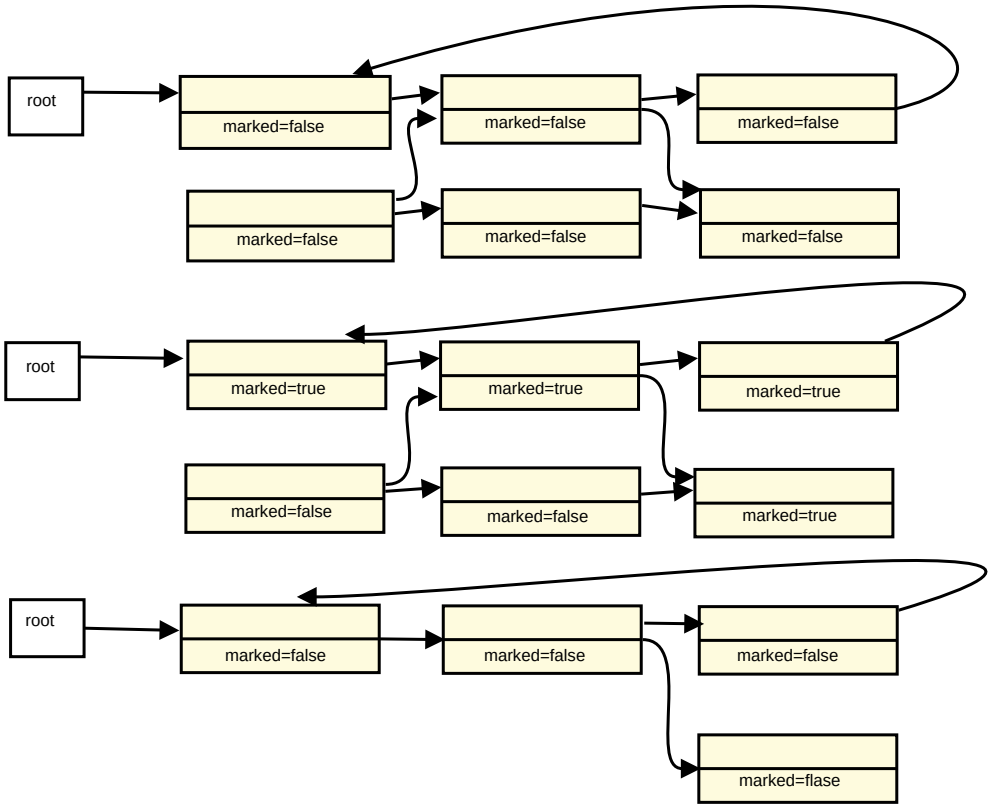
Bộ thông dịch của hệ thống đồng thời giải mã lệnh và thực thi nó bằng các hàm được viết bằng các ngôn ngữ lập trình bậc cao. Trong ngôn ngữ AS, phương thức được khai báo tĩnh với tối đa số lượng ngăn xếp và dữ liệu cục bộ cần thiết. Do đó bộ thông dịch cấp phát cho mỗi phương thức một mảng con trỏ tới đối tượng của ngăn xếp.

2.3.5 Quản lý bộ nhớ

Quản lý, dọn dẹp bộ nhớ (Memory management/Garbage Collection). Cơ chế hoạt động của MMgc[10]:

MMgc sử dụng thuật toán phổ biến là mark/sweep được minh họa ở hình 2.9. Mỗi đối tượng trong hệ thống được kết hợp với nhau bởi bit đánh dấu dọn dẹp bộ nhớ thực

hiện gồm 2 bước: đánh dấu và quét. Trong bước đánh dấu, tất cả các bit đánh dấu được xóa là chưa mark. Khối dọn dẹp được xem như là con trỏ "root"- điểm bắt đầu tại dữ liệu của các ứng dụng. Nó bắt đầu xét tất cả các đối tượng, bắt đầ từ "root" và mở rộng ra ngoài. Với mỗi đối tượng nó gặp, bit đánh dấu được gán.



Hình 2.9: Minh họa thuật toán mark/sweep

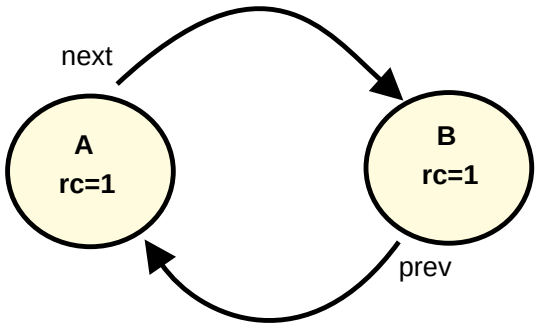
Sau khi kết thúc bước đánh dấu, bước quét (Sweep) bắt đầu. Trong mỗi lần quét, các đối tượng không được đánh dấu mark bit sẽ bị xóa và thu hồi bộ nhớ. Nếu một đối tượng không được đánh dấu bit trong suốt quá trình Mark có nghĩa là "root" không hướng ra nó hay nó không được dùng trong chương trình tại thời điểm đó.

Nhược điểm: thuật toán được tách nhỏ ra gồm ClearMarks/Mark/Finalize/Sweep. Cài đặt thực tế ClearMarks/Finalize/Sweep đối với GC thực hiện cho tất cả đối tượng. Ta thấy, mark không cần thực hiện đầu tiên do nó cần phải đợi Sweep, tương tự bước hoàn thành việc dọn dẹp không cần phải thực hiện trong 1 chu kỳ. Vì thời gian sử dụng ở bước Mark nên hiệu năng cao không tăng.

Conservation Collection (Thu dọn bảo toàn): MMgc là bộ dọn dẹp kiểu mark/sweep bảo toàn, có nghĩa là nó không thu hồi tất cả bộ nhớ khi có thể làm. Quyết định "bảo toàn" và không thu hồi bộ nhớ có thể không được sử dụng thỉnh thoảng thực hiện theo cài đặt của thuật toán. Để có sự bảo toàn này, MMgc xem như mỗi vùng bộ nhớ có chứa con trỏ GC. Trường hợp xấu xảy ra, khi vùng bộ nhớ chứa con trỏ GC nhưng thực sự nó

không có dữ liệu, như vậy xảy ra "memory leak". "Memory leak" sẽ tăng theo thời gian, chương trình của chúng ta sẽ tiêu tốn hàng trăm Mega Bytes. Nhưng với bộ thu hồi bảo toàn (conservation GC) thực hiện ngẫu nhiên, vấn đề bộ nhớ bị lãng phí tăng nhanh theo thời gian không diễn ra.

Deferred Reference Counting (DRC) MMgc sử dụng DRC. DRC là chiến thuật để lấy đối tượng trung gian thu hồi lại trong khi MMgc vẫn thực hiện với hiệu năng cao. Đây là 1 kiểu quản lý bộ nhớ tự động. Tuy nhiên khi gặp phải hai đối tượng A và B như ở hình 2.10 có biến đếm và tham chiếu đến nhau sẽ gây ra lỗi, biến đếm của chúng sẽ khác 0 khi mà không có đối tượng nào trong hệ thống trỏ đến nó. Các đối tượng này sẽ không bao giờ bị hủy. Việc này sẽ làm bộ nhớ lãng phí tăng lên nhanh chóng, hiệu năng chương trình giảm.



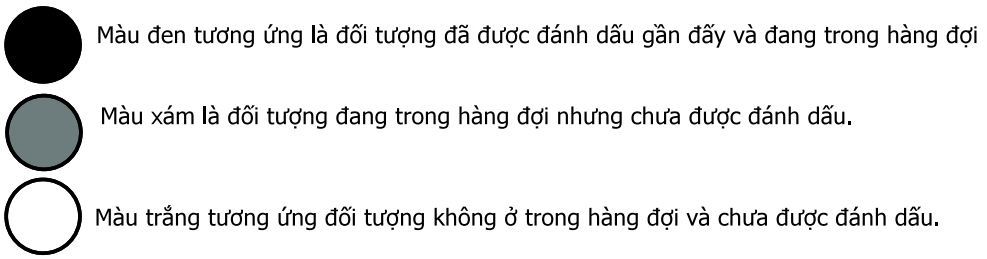
Hình 2.10: Lỗi tham chiếu vòng

Zero Count Table (Bảng đếm không): Khi tham chiếu của đối tượng được đếm là 0, nếu nó bị hủy ngay lập tức có thể gây ra con trỏ lạc (dangling pointer) trên stack. Để giải quyết vấn đề này, sử dụng cơ chế gọi là ZCT. Khi một đối tượng có tham chiếu được đếm 0 thì nó không bị hủy ngay mà đưa vào ZCT. Khi bảng ZCT đầy, nó bắt đầu hủy một vài đối tượng.

Incremental Collection: Flash Player sử dụng hoạt họa và phim hình do đó mà nó phải duy trì tỉ lệ các hình đơn phù hợp. Ứng dụng thực thi nó càng trở nên lớn và tiêu tốn nhiều bộ nhớ. Điều không may là Flash Player sử dụng thời gian tạm dừng là 60s để cho việc thu hồi bộ nhớ, điều này có thể gây ra việc tạm dừng không giới hạn được. Dừng công việc của GC là cách tránh tạm dừng không xác định. Có hai vấn đề ở đây: Làm thế nào để đếm được đối tượng chuyển trạng thái ở heap giữa việc tăng đánh dấu. Thời gian dành cho việc đánh dấu mỗi lần tăng.

Mark consistency (đánh dấu đồng nhất): Một bộ tìm kiếm đối tượng tốt (đối tượng không sử dụng) không hủy đối tượng đang hoạt động. Để làm được điều này, cần giải thích cho bộ tìm kiếm hiểu biết đối tượng mới hoặc chưa được đánh dấu được lưu trữ bên trong đối tượng mà ta đã đánh dấu. Cài đặt cơ chế này bằng cách lưu đối tượng mới hay chưa đánh dấu vào một đối tượng đã được thuật toán đánh dấu xử lý và không ở trong

hàng đợi. Như vậy sẽ không hủy đối tượng conf sử dụng và không làm cho con trỏ tới nó bị lạc. Thuật toán ba màu, mô tả ba trạng thái ở hình 2.11, dựa vào thuật toán ba màu có chắn ở một số nghiên cứu.



Hình 2.11: Sử dụng thuật toán ba màu

Hoạt động của thuật toán ba màu như sau: trong lần đầu tiên, "root" bị đưa vào hàng đợi, do đó sau bước này "roots" có màu xám và tất cả đối tượng còn lại là màu trắng. Tiếp theo, hàng đợi sẽ được xử lý đối tượng trong hai bước, từ màu xám sang đen và từ trắng sang xám. Khi nào con trỏ tới đối tượng màu trắng được chuyển thành đối tượng màu đen thì chúng ta tách ra hai việc, vẽ màu trắng sang đen và đưa đối tượng trắng vào hàng đợi. Với cài đặt như vậy, những trường hợp sau đều giải quyết được:

- Màu xám được chuyển sang đen, xám trắng - đối tượng có màu xám ở trong hàng đợi và nó sẽ bị đánh dấu trước khi quét;
- Màu trắng sang xám - đối tượng trắng sẽ được đánh dấu cho đến lúc đối tượng xám được đánh dấu;
- Màu trắng sang trắng - hoặc nó sẽ chuyển sang xám hoặc không trong trường hợp tất cả đối tượng đã đánh dấu;
- Màu đen chuyển sang đen, xám, trắng - với màu đen, nó đã bị đánh dấu.

2.4 Papervision3D

2.4.1 Giới thiệu

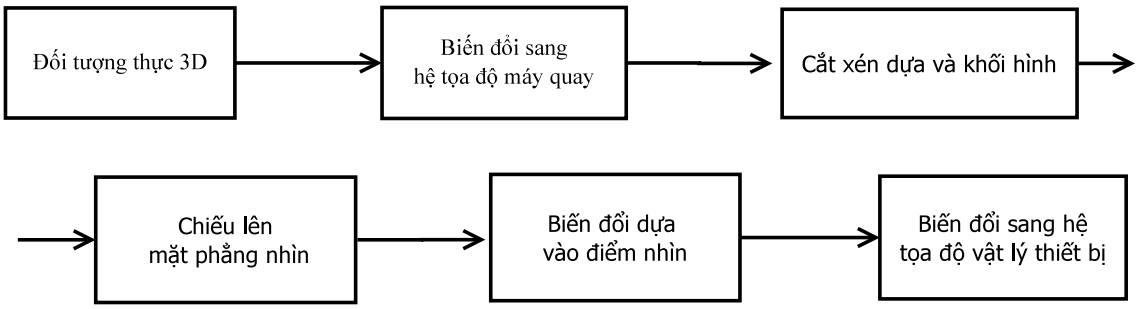
Papervision3D là một thư viện nguồn mở 3D Flash. Bao gồm tập hợp các tệp được viết bằng ActionScritpt 3.0. Nó cung cấp giao diện lập trình ứng dụng cho phép người phát triển tạo các chương trình 3D Flash. Papervision3D xây dựng 3D thời gian thực,

tạo ra những chương trình đồ họa 3D đẹp mắt, lôi cuốn và người dùng có thể tương tác. Sản phẩm của Papervision3D từ những ảnh quảng cáo trên trang Web cho đến những trò chơi trực tuyến đối kháng, chiến tranh đồ họa phức tạp.

Papervision3D được sử dụng trên các công cụ của Adobe như: Flash, Flex Builder (Bộ biên dịch là Flex SDK).

Papervision3D xây dựng 3D Flash tuân theo kỹ thuật của đồ họa máy tính như trong hình 2.12. Các bước này thực hiện như sau:

- Xây dựng, xác định hệ tọa độ máy quay từ hệ tọa độ thế giới thực. Papervision3D sử dụng hệ tọa độ Đề Các 3 chiều;
- Các vật thể sau khi được xác định tọa độ sẽ được cắt xén. Cắt xén để đảm bảo khung nhìn chỉ hiển thị trong phạm vi của nó;
- Sau đó, chiếu đối tượng lên mặt phẳng nhìn;
- Phép biến đổi dựa vào điểm nhìn này bản chất đưa 3D về 2D;
- Cuối cùng là chuyển hệ tọa độ máy quay, chương trình sang hệ tọa độ vật lý thiết bị.

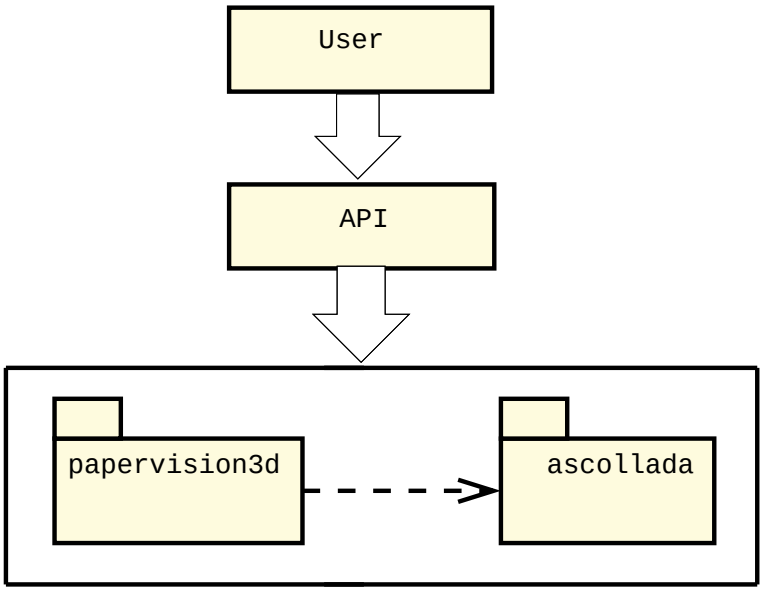


Hình 2.12: Kỹ thuật đồ họa 3D trong đồ họa máy tính

2.4.2 Kiến trúc, thành phần

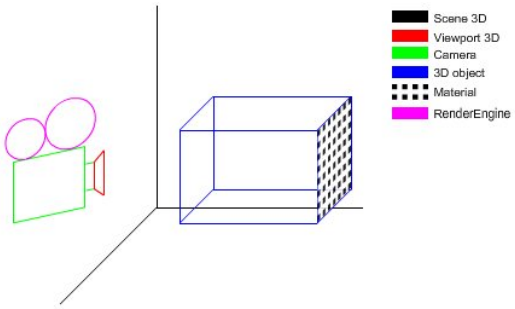
Mô hình chung các gói trong thư viện Papervision3D như hình2.13 gồm hai gói chính:

- Gói ascollada: collada (COLLAborative Design Activity) là những kịch bản được thiết kế sẵn để tạo ra ra tệp có dạng * *.dae* có thể chuyển đổi được cho ứng dụng tương tác 3D. Ascollada hỗ trợ 3D với ngôn ngữ AS.
- Gói papervision3d gồm các thành phần chính tạo 3D và xử lý sự kiện, tương tác với người dùng.



Hình 2.13: Mô hình xây dựng thư viện Papervision3D

Để xây dựng hình ảnh 3D trên các thiết bị hiển thị 2D ta cần sử dụng kỹ thuật của đồ họa máy tính mà cần có những thành phần: Camera, Viewport, Scene, Render



Hình 2.14: Thành phần 3D trong đồ họa máy tính

Sau đây là chi tiết về các thành phần để tạo ra đồ họa 3D cho Flash dựa vào Papervision3D:

Mô hình xây dựng Papervision3D

Ngôn ngữ AS đến phiên bản 3.0 chỉ hỗ trợ việc tạo ra ứng dụng hoạt họa 2D. Do đó, những thư viện 3D hỗ trợ cho ngôn ngữ AS phải xây dựng toàn bộ những thành phần cơ bản 3D của đồ họa máy tính. Mã nguồn Papervision3D bao gồm các gói là các thành phần và các phép biến đổi trên đồ họa 3D. Mô hình các gói chính bao gồm: gói "core" xây dựng các thành phần cơ bản để xây dựng và hỗ trợ đồ họa 3D gồm các gói bên trong như hình 2.15 gồm:

- Animation: các thuật toán vẽ đường cong: Bezier, đường cong tuyến tính, vẽ đường

cong theo từng bước; Các phương pháp chuyển đổi hình ảnh tạo hoạt cảnh.

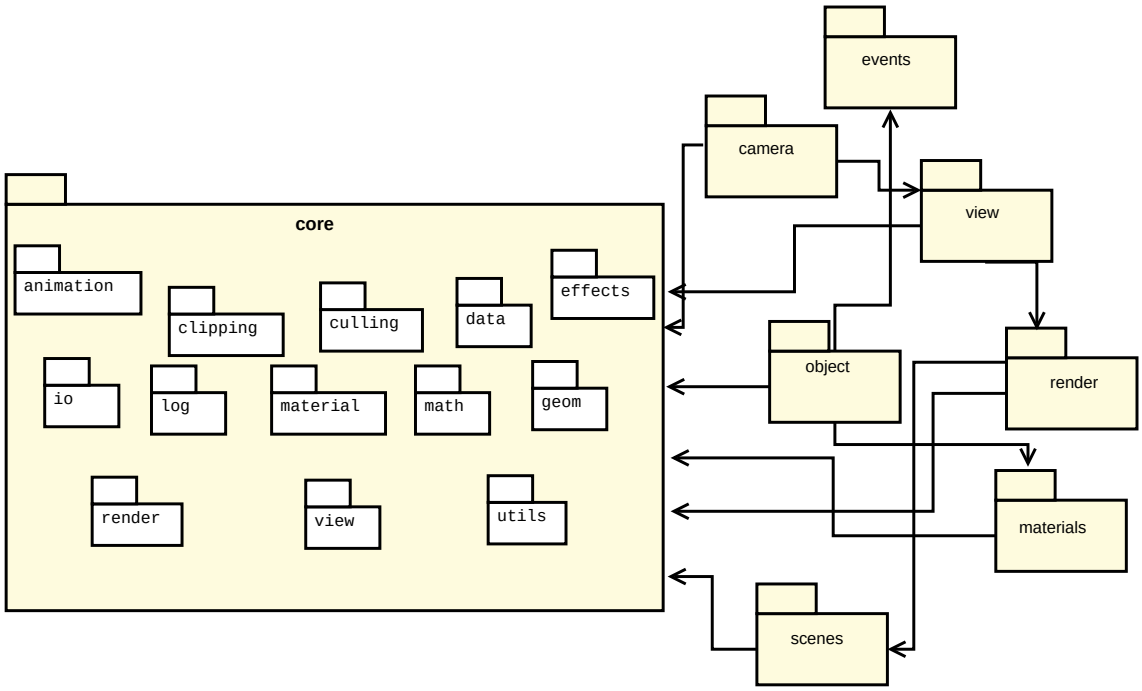
- Clipping: thuật toán cắt xén đối tượng 3D. Viewport là 1 khung nhìn mà Camera quan sát được trong 1 khung; cảnh (scene) ở không gian 3D, do đó mà chỉ hữu hạn những đối tượng trong khung nhìn được hiển thị cần phải được cắt xén;
- Culling: xác định mặt hiện, mặt hiển thị của vật thể trong không gian 3D tới Camera;
- Controller: có nhiệm vụ điều khiển về hoạt họa, bề mặt của đối tượng;
- Effects: các hiệu ứng trong đồ họa 3D. Papervision3D sử dụng các hiệu ứng cho 2D từ gói flash.filters để áp dụng cho đồ họa 3D như làm mờ (blurfilter), chiếu sáng (glowfliter);
- geom: định nghĩa các hình cơ bản như điểm ảnh 3D, tam giác 3D, đoạn thẳng 3D, điểm 3D;
- Io: xuất chương trình Papervision3D ra dữ liệu dạng Collada (*.dae);
- Material: Định nghĩa độ sáng, độ mịn của bóng;
- Math: thực hiện các tính toán với ma trận 3D, 2D cho các phép biến đổi;
- Render: cơ sở cho cách chiếu, tạo bóng cho vật thể 3D;
- view: là cơ sở cho phương pháp cài đặt, xây dựng khung nhìn trong không gian 3D;

Các gói: Camera, events, view, render, materials, scenes, object, render được cài đặt dựa vào các lớp trong gói core ở trên. Cấu tạo và nhiệm vụ của những thành phần này đã được nêu kỹ trong chương 2, phần Papervision3D. Như vậy, bản thân Papervision3D đã định nghĩa các thành phần trong đồ họa 3D, các phép biến đổi liên quan như thuật toán cắt xén, thuật toán xác định mặt hiện, thuật toán tạo bóng cho vật thể. Đồng thời những kỹ thuật xử lý với Flash như xử lý sự kiện với tương tác của người dùng, chất liệu, màu sắc cho đối tượng Flash và các hiệu ứng cũng được Papervision3D thừa kết và phát triển từ AS 3.0

Scene

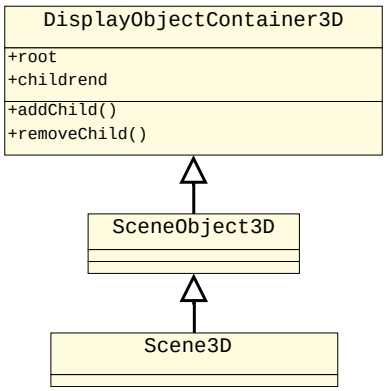
Là được hợp thành bởi tất cả các đối tượng 3D trong không gian 3D. Giả sử tạo ra một đối tượng trong chương trình, để có thể đưa nó xuất hiện ra thì cần phải cho đối tượng này thuộc Scene.

Trong Papervision3D Scene[11] đóng vai trò như nơi chứa tất cả đối tượng của không gian 3D. Các đối tượng được lưu trữ theo cấu trúc dữ liệu dạng cây, được thêm và bỏ qua các phương thức tương ứng là addChld() và removeChild(). Scene3D thừa kế từ



Hình 2.15: Quan hệ giữa các gói trong Papervision3D

lớp SceneObject3D và do tính chất nêu trên nên lớp DisplayObject3D là lớp cơ sở của SceneObject3D.



Hình 2.16: Scene3D

Camera

Có thể tưởng tượng chức năng của nó như một Camera thật mà nó ở trong không gian 3D và ghi hình lại tất cả các hoạt động bên trong Scene của chương trình. Camera xác định điểm nhìn mà chúng ta đang xem Scene. Nó không phải là một vật thể hiện hữu trong chương trình do đó không cần phải cho vào Scene. Giống như Camera thông thường Camera trong Papervision3D có thể phóng to, hướng trọng tâm, và các tính năng khác. Vì lý do hiệu năng của chương trình, Camera có thể bỏ qua những đối tượng ở vị

trí quá xa nó, như vậy tiết kiệm được thời gian tính toán cho đối tượng đó.

Trong thư viện Papervision3D, có xây dựng bốn loại Camera theo các cách sử dụng khác nhau. Thuật ngữ Camera chỉ chung 4 loại này. Theo sơ đồ cấu tạo thì Camera thừa kế từ lớp CameraObject3D mà lớp này lại là lớp con của DisplayObject3D. Camera có các đặc tính cơ bản là:

- Focus (tiêu điểm) và field of view (góc quan sát). Focus là số dương thể hiện khoảng cách giữa Camera và viewport (mặt phẳng chiếu). Field of view là góc thể hiện vùng quan sát của Camera. Giá trị của Focus và field có sự liên hệ với nhau là khi giá trị focus lớn thì khoảng cách giữa Camera và mặt phẳng chiếu tăng lên và field of view (góc quan sát) giảm đi. Và ngược lại góc quan sát tăng lên khi giá trị focus bé.
- Zoom: Khi chúng ta phóng to tranh ảnh hay phim, thì ảnh mà ta nhìn thấy được kéo rộng ra. Tức là cảnh đuợc mở rộng ra mà không cần phải đưa Camera lại gần. Đây chính là cơ chế zoom trong Papervision3D.
- Mối quan hệ giữa Zoom, Focus và Field of View: Nếu giảm giá trị của zoom hoặc focus thì giá trị field of view cũng giảm. Ngược lại, tăng hoặc giảm field of view sẽ làm cho giá trị focus tăng hoặc giảm tương ứng, nhưng không làm thay đổi giá trị của zoom.
- Near và Far: Khi cắt một hình bằng hai mặt phẳng gần và xa thì giá trị near và far chính là khoảng cách giữa Camera đối với mặt phẳng gần và mặt phẳng xa tương ứng. Đặt giá trị cho thuộc tính near cũng như gán cho focus.

Papervision3D xây dựng bốn loại Camera:

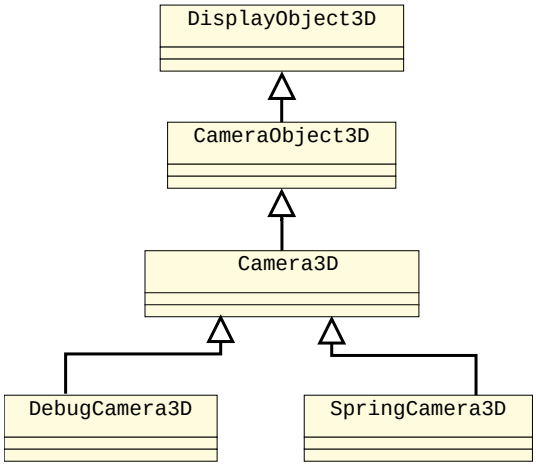
Target Camera là loại Camera luôn hướng nhìn đến 1 đích. Kiểu dữ liệu thuộc tính của Target Camera là DisplayObject3D.

Free Camera hoạt động như target Camera ngoại trừ không có đích để hướng vào. Nó luôn nhìn theo hướng thẳng tiến theo trục z trong hệ tọa độ 3 chiều.

Debug Camera: khi xây dựng một chương trình 3D Flash dựa vào Papervision3D chúng ta có thể sẽ thiếu sót phần nào đấy trong không gian 3D. Để lưu trữ vị trí Camera và những cài đặt cho người dùng kiểm tra lại chương trình của mình là lý do tạo ra Debug Camera. Debug Camera xây dựng những cách điều hướng chương trình sử dụng bàn phím và chuột, đồng thời hiển thị những thống kê những gì sẽ tiếp tục xảy ra với chương trình.

Spring Camera: là kiểu Camera luôn di chuyển theo đối tượng trong chương trình. Không chỉ vậy, nó còn có hiệu ứng lò xo khi đối tượng tăng tốc hoặc rẽ. Spring Camera sử dụng những tính chất vật lý, tạo ra một lò xo theo tưởng tượng giữa Camera và đối

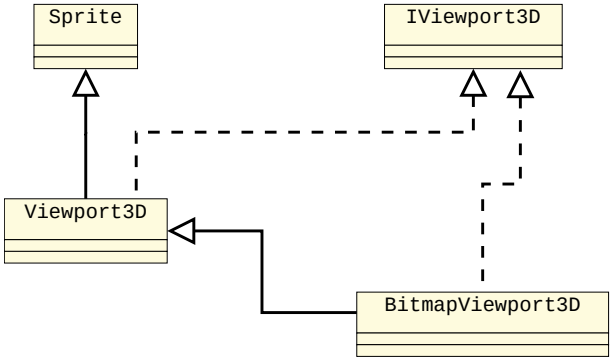
tượng. Khi đối tượng di chuyển, lò xo này sẽ được kéo giãn ra hoặc co lại, tạo ra sự di chuyển liên tục trông như thật.



Hình 2.17: Camera trong Papervision3D

Viewport

Viewport là một vùng chứa hiển thị những gì Camera của chương trình thấy được . Có thể tưởng tượng như nó là ống kính của Camera, thể hiện một phần trong scene 3D . Bước cắt xén đối tượng được thực hiện với viewport.



Hình 2.18: Viewport trong Papervision3D

Object

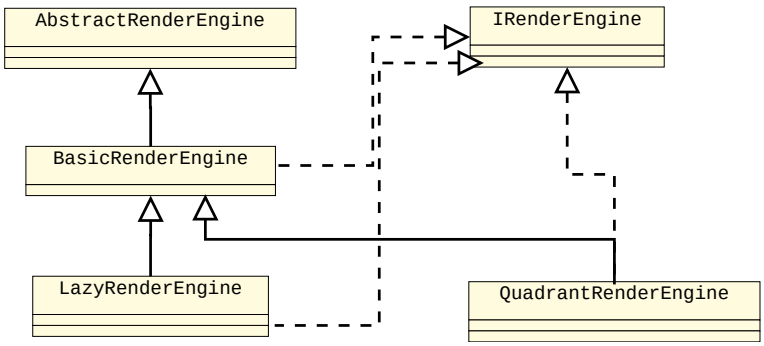
Một hình trong không gian 3D được gọi là vật thể hay đối tượng 3D, hay DisplayObject3D trong Papervision3D. Các đối tượng cơ bản được xây dựng sẵn trong Papervision3D là: điểm 3D, đường thẳng, hình nón, hình lập phương, hình trụ, máy bay giấy, máy bay, hình cầu.

Material

Chất liệu được sử dụng trong Papervision3D để tô phủ lên bề mặt để tạo cho đối tượng 3D trông thật, sống động và đẹp mắt. Việc mô hình hóa đối tượng 3D được thực hiện thông qua xây dựng từ các tam giác.

Render

Để vẽ lên khung nhìn ta cần đến render engine (máy vẽ). Nó liên tục cập nhật những gì xảy ra trong khung cảnh ở không gian 3D mà Camera ghi lại và vẽ lại trên khung hình (viewport).



Hình 2.19: Xây dựng Render

2.4.3 Một số kết luận

Mối quan hệ giữa Papervision3D và AS 3.0

Papervision3D dựa vào ngôn ngữ AS 3.0 để xây dựng phương pháp hiển thị đồ họa 3D. Các tính chất tạo hoạt họa, chuyển động, hiệu ứng được đều thừa kế và mở rộng từ AS 3.0.

So sánh OpenGL và Papervision3D

Cách vẽ đối tượng, cơ chế hoạt động của OpenGL và Papervision3D có một số khác biệt là:

- Papervision3D sử dụng hệ trục tọa độ phải[13] còn OpenGL sử dụng hệ tọa độ trái [14].

[13] trục z đi vào trong
[14] trục z đi ra

- Thuật toán xác định mặt hiện mà OpenGL sử dụng là *z-buffer*[15] trong khi Papervision3D sử dụng thuật toán *painter's algorithm* [16].
- Papervision3D là thư viện hỗ trợ cho việc lập trình, OpenGL không chỉ hỗ trợ lập trình mà còn giao tiếp với phần cứng đồ họa của thiết bị cài đặt nó.

So sánh Papervision3D với OpenGLES

Papervision3D có thể thực thi như OpenGL, OpenGLES chỉ sử dụng một phần các hàm API của OpenGL, do đó chương trình 3D từ Papervision3D đẹp và phong phú hơn. Hơn nữa cơ chế tương tác với Flash cũng tăng thêm sự hấp dẫn, lôi cuốn.

Thư viện Papervision3D được xây dựng với nhiều hàm, hỗ trợ lập trình tốt để tạo ra các chương trình 3D đẹp mắt. Việc lập trình OpenGLES gặp khó khăn khi phải cân nhắc đến tài nguyên hệ thống mà chương trình sử dụng, trong khi với Papervision3D việc này được máy ảo xử lý.

Như vậy lựa chọn Papervision3D để phát triển xây dựng đồ họa 3D để thực thi trên thiết bị nhúng mang lại nhiều ưu điểm hơn OpenGLES.

2.5 Môi trường đồ họa OpenGLES trên PowerVR

Tương tự Papervision3D, OpenGL và OpenGLES cũng là thư viện hỗ trợ đồ họa 3D. OpenGL được sử dụng trên môi trường PC còn OpenGLES sử dụng trên thiết bị nhúng. Khi một tệp tin được tạo khi sử dụng OpenGL, sử dụng nó để thực thi trên môi trường thiết bị nhúng với OpenGLES thì sẽ có những thành phần không xuất hiện được. Bởi vì, chỉ có 10% các hàm trong OpenGL được sử dụng ở OpenGLES, 50% số hàm đó từ OpenGL phải chỉnh sửa thay đổi với tham số sao cho sử dụng ít tài nguyên hơn. Bởi vì năng lực xử lý của thiết bị nhúng bị hạn chế hơn.

Với môi trường giả lập trên PC mô phỏng PowerVR insider [17] được cung cấp cho người phát triển phần cứng và phần mềm trên thiết bị nhúng. PowerVR đưa ra một giải pháp mạnh mẽ, linh hoạt cho việc thực thi những ứng dụng 2D/3D véc tơ với GPU [18] bao gồm cả xử lý, tạo ảnh 2D/3D mã hóa - giải mã Video. Tất cả các API chính được hỗ trợ bao gồm: OpenGLES 2.0/1.1, OpenVG 1.1, OpenGL 2.0/3.0, DirectX 9/10 và OpenCL. Đặc biệt với các phiên bản tiếp theo (từ phiên bản 5), PowerVR cung cấp giải pháp cho tất cả các dạng tiếp theo của đồ họa 3D, 2D, đồ họa véc tơ cho thiết bị nhúng với các kỹ

[15] bộ đệm z
[16] Thuật toán người thợ sơn
[17] Một chip đồ họa nhúng hỗ trợ thực thi OpenGLES 2.0
[18] Graphics Processing Unit

thuật chống răng cưa, và đưa các chức năng xử lý hình ảnh chuyên sâu.

Trên thực tế, PowerVR đã trở thành chip xử lý đồ họa phổ biết nhất sử dụng trong điện thoại di động để hiển thị hình ảnh 2D, 3D, tăng tốc đồ họa véc tơ [19] và được sử dụng bởi các công ty, tập đoàn điện tử hàng đầu thế giới.

[19] với phiên bản cao cấp

CHƯƠNG 3

Kỹ thuật xử lý đồ họa 3D Flash

Dựa vào phần cơ sở lý thuyết ở trên, chương này sẽ mô tả bài toán đồng thời nêu ra phương pháp giải quyết và mô hình minh họa cho giải pháp trên.

3.1 Mô tả bài toán

Do dự án mang tính dài hạ, ở thời điểm hiện tại chúng tôi chỉ dừng lại ở mức độ tìm hiểu, phân tích và kiểm thử các mã nguồn mở một cách tối thiểu để đưa ra tính thực thi của dự án.

Trong phạm vi dự án, chúng tôi sẽ xây dựng kỹ thuật xử lý 3D Flash trên hệ thống nhúng (trước hết là giả lập có hỗ trợ đồ họa OpenGLES 2.0 như PowerVR SDK [1]. Sử dụng Gnash để xây dựng trình xử lý Flash, hiển thị Flash Video nhằm mục đích thực thi được 3D Flash của Papervision3D. Tiếp đấy, chúng tôi sẽ đưa mô hình giả lập lên thiết bị di động để chơi được 3D Flash.

3.2 Mô hình đề xuất

3.2.1 Ý tưởng

Từ những tìm hiểu về lý thuyết ở chương hai, kết hợp với yêu cầu của bài toán đặt ra chúng tôi có một số nhận xét.

- Việc xử lý đồ họa 3D Flash cũng theo cơ chế xử lý AS 3.0. Papervision3D được phát triển dựa trên AS 3.0, kỹ thuật về tương tác, sự kiện với người dùng hoàn toàn tương tự. Các vật thể 3D cũng được mô hình từ các đối tượng cơ bản của AS 3.0.

[1] PowerVR đã giới thiệu trong chương hai

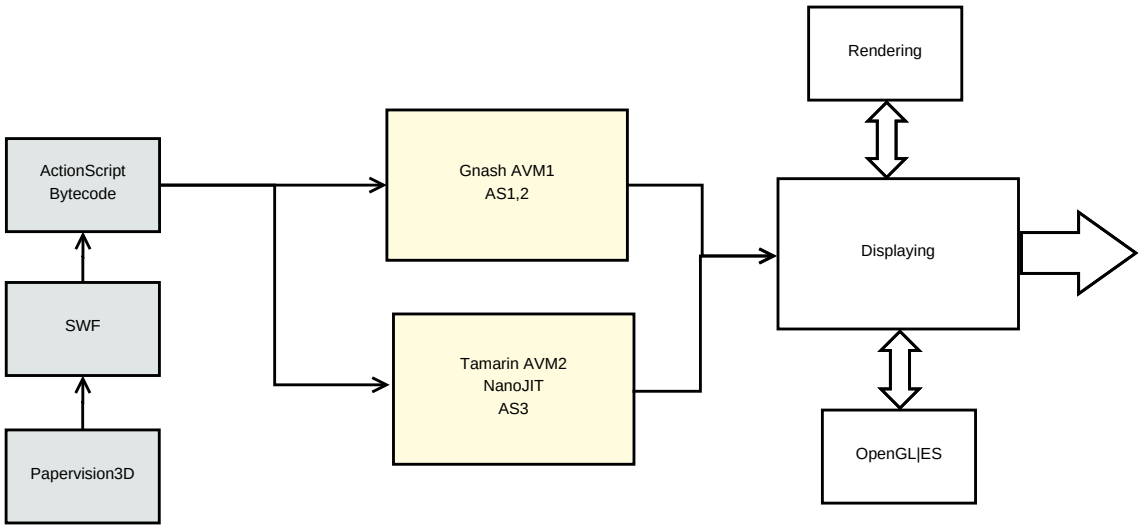
Từ Flash Player phiên bản 9 trở đi có thể hiển thị Papervision3D mà không cần phải cài đặt thêm thư viện nào;

- Máy ảo AVM2 của Gnash chưa hỗ trợ hết cho AS 3.0 nên Papervision3D không chơi được trên Gnash. Phần hiển thị của Gnash hỗ trợ nhiều công nghệ, nền tảng đồ họa (OpenGL, agg, gtk, . . .) chứng tỏ xử lý hình ảnh của Gnash tốt;
- Máy ảo Tamarin-AVM2 thực thi ngôn ngữ AS 3.0 với hiệu năng cao, tối ưu việc vận hành bởi bộ biên dịch JIT và cơ chế quản lý bộ nhớ MMgc [2] Xử lý 3D Flash của Papervision3D cũng như với AS 3.0 nên máy ảo Tamarin-AVM2 có thể đảm trách tốt với đồ họa 3D.

3.2.2 Giải pháp

Theo ý tưởng trên, để xử lý đồ họa 3D của Papervision3D chúng tôi cần có một máy ảo thực thi AS 3.0. Kết hợp đầu ra của máy ảo này với phần hiển thị Gnash sẽ cho ra kết qủa hiển thị 3D Flash. Vì vậy, giải pháp sử dụng máy ảo tamarin- AVM2 cho việc xử lý 3D của Papervision3D, kết hợp với chức năng hiển thị đồ họa của Gnash sẽ thu được máy chơi 3D Flash hiệu quả. Giải pháp được minh họa bởi hình 3.1

- Với AS 1.0 và 2.0 việc thực thi ABC được tiếp tục với máy ảo AVM1 của Gnash.
- Với AS 3.0 và Papervision3D được xử lý bởi máy ảo tamarin-AVM2. Sau khi xử lý ABC, tamarin-AVM2 chuyển đầu ra cho Gnash thực hiện bước hiển thị.



Hình 3.1: Mô hình kết hợp tamarin và Gnash

[2] Phần Tamarin, chương hai

CHƯƠNG 4

Thực nghiệm

Trên cơ sở thực nghiệm này, chúng tôi có sử dụng các dự án nguồn mở thực thi Flash khác nhau để đưa vào so sánh tìm ra phương án tối ưu cho hệ thống.

Trong phần thực nghiệm này chúng tôi có sử dụng một số công cụ bao gồm:

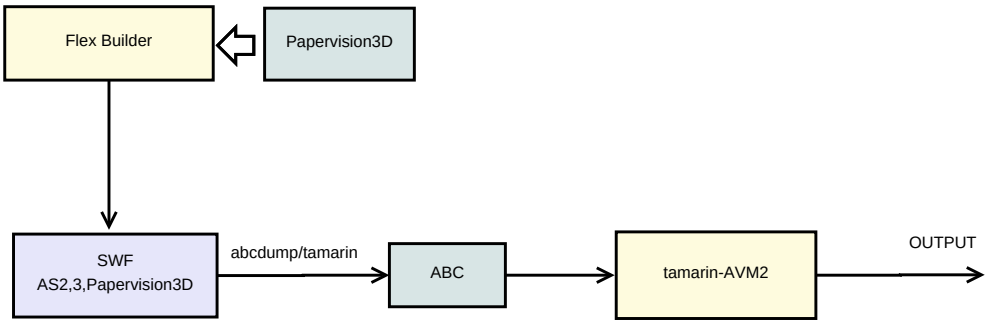
- Bộ công cụ phát triển Java: JDK
- Bộ lập trình phát triển Flex Builder 3.
- Bộ biên dịch Flash nguồn mở Flex SDK 3.4
- Công cụ tách mã ABC từ các tệp SWF "abcdump" trong gói tamarin-central

Môi trường thực nghiệm:

- Bộ xử lý: Intel Core 2 Duo 1.6GHz
- Bộ nhớ RAM: 1.5GB
- Hệ điều hành : Ubuntu 9.10 / Linux

Mô hình thực nghiệm như hình 4.1 gồm các bước như sau:

- Sử dụng bộ phát triển Flex Buider có sử dụng thư viện Papervision3D để sinh ra tệp SWF.
- Dùng công cụ abcdump trong gói util của tamarin để tách tệp SWF Papervision3D sang dạng ABC. Phần phụ lục B nói chi tiết hơn về cách thức thực nghiệm sử dụng công cụ *"abcdump"*.
- Thực thi tệp ABC với máy ảo tamarin.



Hình 4.1: Mô hình xử lý 3D

4.1 Kết quả thực thi

4.1.1 Cấu tạo của ABC

Sử dụng công cụ "abcdump" trong gói tamarin-central, chúng tôi tách được thành phần ABC của các tệp Flash AS 3.0. Với Papervision3D, cũng sinh được tệp ABC tương ứng. Từ những phép bóc tách này, chúng tôi hiểu rõ được bản chất thành phần của tệp, đồng thời nắm được cơ chế nhận đầu vào của máy ảo tamarin-AVM2.

4.1.2 Cấu hình phần cứng cần thiết

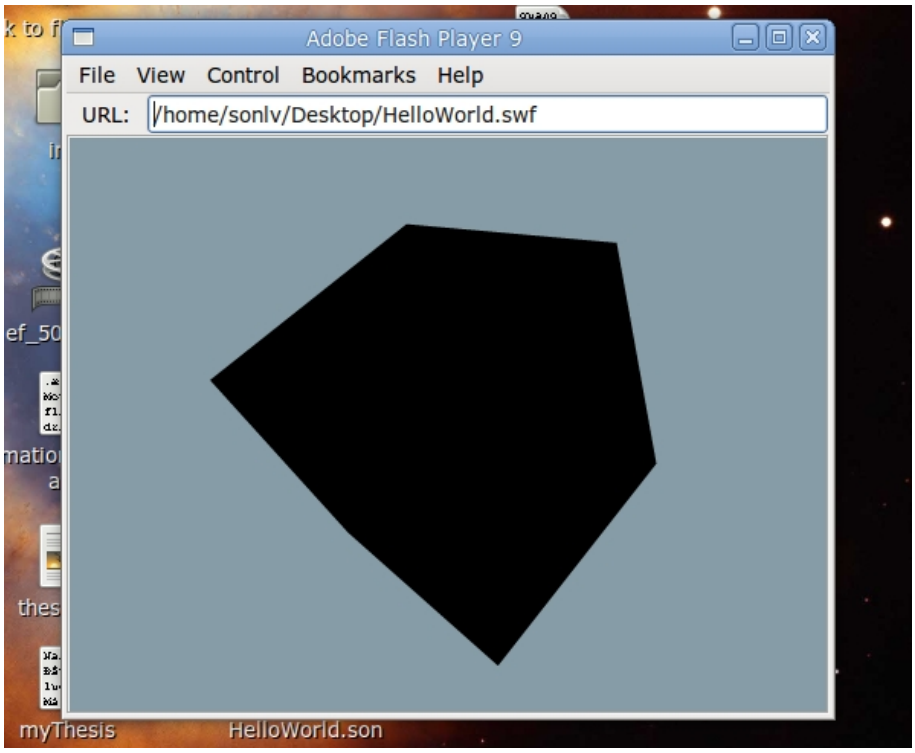
Để thực thi được đồ họa 3D Flash của Papervision3D, thiết bị di động tối thiểu cần phải có chip hỗ trợ xử lý OpenGLES và có khả vận hành máy ảo tamarin và Gnash. Dựa vào những thực nghiệm về đồ họa, chúng tôi đưa ra cấu hình tối thiểu để có thể đáp ứng cho việc xử lý 3D Flash. Cấu hình đề xuất như ở bảng dưới:

Bộ xử lý	Tốc độ xử lý	Bộ nhớ RAM tối thiểu
ARM v7 PBX A9	600 - 800 MHz	128MB
ARM v7 PB A8	550 MHz	64MB
ARM v6 RealView EB	200 - 400 MHz	32MB
ARM v6 RealView PB	200 - 400 MHz	32MB

4.2 Minh họa thực nghiệm

Sử dụng môi trường phát triển lập trình Flex Builder kết hợp với thư viện Papervision3D, chúng tôi tạo ra chương trình 3D Flash đơn giản.

Tệp Flash chạy trên phần mềm Flash Player, hiển thị như hình 4.2. Ví dụ minh họa hình lập phương quay trong không gian 3D. Đoạn mã Papervision3D dùng để tạo chương trình trên được nói chi tiết trong phần phụ lục C



Hình 4.2: Thực thi 3D của Papervision3D trên Flash Player

Khóa luận này đang trong giai đoạn thực hiện của dự án nên chưa có thực nghiệm trên thiết bị nhúng. Tuy nhiên, kết quả sau khi hoàn thành dự án, chúng tôi xây dựng môi trường xử lý 3D Flash trên thiết bị di động mà chất lượng hình ảnh và tốc độ xử lý đạt như minh họa trên.

CHƯƠNG 5

Kết luận và hướng phát triển

5.1 Kết luận

Trong giai đoạn đầu của dự án, chúng tôi đã đạt được một số kết quả: hiểu cơ chế xử lý, xây dựng dồ họa 3D Flash-Papervision3D, cơ chế hiển thị Flash của Gnash, kiểm tra tính khả thi sử dụng máy ảo Tamarin, và đề ra giải pháp cho máy Flash xử lý 3D-Paperision3D. Giao đoạn tìm hiểu, xây dựng các vấn đề lý thuyết này là nền tảng cho việc xây dựng mô hình thực thi 3D Flash trên thiết bị di động. Với thời gian không cho phép,

5.2 Hướng phát triển

Trong khuôn khổ những gì đã thực hiện, chúng tôi đã chứng minh tính khả thi cho việc xử lý 3D Flash trên thiết bị nhúng. Đồng thời, lý thuyết và mô hình đã xây dựng được là cơ sở cho các bước tiếp theo của dự án để đưa hệ thống xử lý Flash hoàn thiện trên hệ thống nhúng, môi trường OpenGLES. Hơn nữa, những thuật toán xử lý 3D sẽ dựa vào lý thuyết tìm hiểu về Papervision3D.

PHỤ LỤC A

Kiểu và tập lệnh trong AVM2

A.1 Một số kiểu dữ liệu trong AVM2

- int (nguyên) biểu diễn giá trị nguyên độ lớn 32bit, khoảng giá trị của nó là:
- uint (nguyên không dấu) có độ dài 32bit, giá trị nằm trong khoảng
- double (số thực dấu phẩy động)
- string (xâu ký tự) biểu diễn lần lượt các ký tự Unicode. String ở dạng UTF-8 và độ lớn có thể là 2
- namespace
- null biểu diễn không đối tượng nào.
- undefined biểu diễn không có giá trị

A.2 Tóm tắt tập lệnh AVM2

Các lệnh chỉ định kiểu dữ liệu mà nó thực hiện bởi hậu tố ở sau tên lệnh như : _b (Boolean) , _a (any), _i (int), _d (double), _s (string), _u (unsigned), _o (object).

- Lệnh tải và lưu. Truy cập vào thanh ghi cục bộ thông qua những lệnh sau: getlocal, getlocal0, getlocal1, getlocal2, getlocal3, setlocal0, setlocal1, setlocal2, setlocal3.
- Lệnh về số học. Phép cộng thực thi bởi những lệnh: increment, increment_i, inclocal, inclocal_i, add, add_i. Phép trừ gồm: decrement, decrement_i, delocal, delocal_i, subtract, subtract_i. Phép nhân chia: multiply, multiply_i, divide và modulo. Để đảo dấu giá trị dùng: negate, negate_i.

- Phép tính trên bit: bitnot, bitand, bitor, bitxor, lshift, rshift, urshift.
- Chuyển đổi kiểu trong ngôn ngữ AS lỏng lẻo, đối tượng có thể chuyển đổi kiểu dữ liệu để thực hiện các phép toán. Trong một vài trường hợp, chuyển đổi kiểu yêu cầu lệnh coerce gồm: coerce, convert_b, coerce_a, convert_i, convert_d, convert_s, convert_u, convert_u và convert_o.
- tạo đối tượng và điều chỉnh các lệnh bởi: newclass, newobject, newarray, newactivation. Gọi các hàm constructor: constructor, constructsuper và constructprop. Namespace được tạo động bằng dxns và dxnslate.
- Quản lý Stack (ngăn xếp): pushnull, pushundefined, pushtrue, pushfalse, pushnan, pushbyte, pushshort, pushstring, pushint, pushdouble, pushscope và pushnamespace. Pop để lấy giá trị khỏi đầu ngăn xếp, swap chuyển đổi giá trị của 2 giá trị ở trên.
- Lệnh chuyển điều khiển: lệnh này chuyển việc thực thi tới một lệnh khác là các cấu trúc rẽ nhánh: iflt, ifnlt, ifnle, ifge, ifngt, ifnge, ifeq, ifne, ifstricteq, ifstrictne, iftrue và iffalse. Lệnh label dùng để vị trí đích của lệnh rẽ nhánh.
- Gọi hàm và trả về của hàm được thực hiện bởi hàm: call để gọi hàm. Các kiểu gọi tĩnh, động, gọi phương thức của lớp cơ sở như sau: callmethod, callstatic, callsuper. Để lấy giá trị trả về: callpropvoid, callsupervoid được sử dụng trong callproperty và callsuper tương ứng.
- Lệnh về bắt lỗi: throw
- Lệnh gỡ rối chương trình: debugfile và debugline chỉ ra giá trị ở thời điểm cụ thể để tìm ra chỗ không hợp lý.

PHỤ LỤC B

Công cụ Abcdump / Tamarin

Gói *tamarin-central* cung cấp công cụ *abcdump*. Chúng tôi sử dụng bộ biên dịch *asc*[1] để tạo tệp thực thi của *abcdump* trên môi trường Ubuntu/Linux.

Sử dụng ví dụ như chương trình ở hình 4.2 và *abcdump* tách ra được đoạn mã có phần đầu:

```
decompressed swf 105344 -> 240585
size [Rect 0 0 10000 7500]
frame rate 24
frame count 1
FileAttributes 4b 0%
77 (invalid) 458b 0%
EnableDebugger2 31b 0%
DebugID 16b 0%
ScriptLimits 4b 0%
SetBackgroundColor 3b 0%
ProductInfo 26b 0%
FrameLabel 11b 0%
DoABC2 239972b 99%
```

Dựa vào phần đầu mã ABC thu được thông tin về tệp Flash Papervision3d là: kích thước, tốc độ khung hình, số khung, thuộc tính ....
Phần tiếp theo:

```
abc name frame1
magic 2e0010
```

[1] actionscript compiler

```
Cpool numbers size 233 0 %
Cpool strings count 3143 size 90348 37 %
Cpool namespaces count 326 size 954 0 %
Cpool nssets count 69 size 749 0 %
Cpool names count 1626 size 6712 2 %
MethodInfo count 1122 size 8757 3 %
InstanceInfo count 126 size 9211 3 %
ClassInfo count 126 size 1852 0%
ScriptInfo size 954 0 %
MethodBodies count 1062 size 120186 50 %
```

Phần này cung cấp thông tin về các đối tượng, lớp trong tệp Flash trên.
Phần cuối cùng (trích dẫn) các lệnh cần được AVM thực thi theo kịch bản:

```
OPCODE    SIZE      % OF 110533
getlex              11012     9%
getlocal            5690      5%
setproperty         3874      3%
initproperty        3224      2%
debugfile           3123      2%
callpropvoid        2698      2%
pushstring          596       0%
kill                552       0%
convert_u           6         0%
ifstricteq          4         0%
bitor               4         0%
greaterequals       4         0%
throw               3         0%
pushundefined       2         0%
modulo              2         0%
strictequals        2         0%
in                  2         0%
lshift              1         0%

SymbolClass 15b 0%
ShowFrame 0b 0%
End 0b 0%
```

PHỤ LỤC C

Mã nguồn chương trình Papervision3D

```
package {
    import flash.events.Event;
    // Import thu vien xu ly su kien cua Flash AS 3.0
    import org.papervision3d.materials.ColorMaterial;
    // Import thu vien mau cua Papervision3d
    import org.papervision3d.materials.utils.MaterialsList;
    //Import chat lieu
    import org.papervision3d.objects.primitives.Cube;
    // Import hinh lap phuong trong Papervision3d
    import org.papervision3d.view.BasicView;
    //Thua ke lop BasicView trong goi view de tao khung hien thi
    public class HelloWorld extends BasicView { // Khai bao lop
        private var cube:Cube;
        private var matList:MaterialsList;
        private var color:ColorMaterial;
        public function HelloWorld() {//Bat dau ham khoi tao
            color=new ColorMaterial(0x000000);// Khoi tao mau den
            matList = new MaterialsList();
            matList.addMaterial(color, "all");// gan mau
            cube=new Cube(matList);
            // Khoi tao doi tuong hinh lap phuong
            scene.addChild(cube);
            // Them hinh lap phuong tren khong gian 3D
            startRendering();
            // Thuc hien ve len viewport
        }// ket thuc ham khoi tao
```

```
        override protected function
            onRenderTick(event:Event = null):void {
            cube.yaw(2);//tao quy dao
            cube.roll(2);//tao quay cho vat the
            super.onRenderTick(event);//
        }// ket thuc phuong thuc onRenderTick
    }
}
```

Tài liệu tham khảo

[1] GNU. URL http://www.gnu.org/software/gnash/.

[2] Papervision3d. URL http://blog.papervision3d.org/.

[3] Adobe, . URL http://adobe.com/.

[4] Adobe, . URL http://www.actionscript.org/.

[5] Adobe Systems Incoporated. *SWF File Format Specification Version 10*. adobe, 2008.

[6] Adobe, . URL http://www.adobe.com/products/flashplayer/.

[7] Adobe Systems Incoporated. *ActionScript Virtual Machine 2 (AVM2) Overview*. San Jose, 2006.

[8] Adobe Systems Incoporated. *Adobe Flash Player ActionScript Virtual Machine*. adobe, 2006.

[9] Mozilla, . URL https://developer.mozilla.org/En/Nanojit.

[10] Mozilla, . URL https://developer.mozilla.org/en/MMgc#Allocating_objects.

[11] Pault Tondeur Jeff Winder. *Papervision3D Essential*. Birmingham - Mumbai, 2009.