ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Đinh Anh Thái

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

KỸ THUẬT HIỂN THỊ FLASHVIDEO DỰA TRÊN GNASH

KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Công nghệ phần mềm

Cán bộ hướng dẫn: PGS. TS. Nguyễn Việt Hà
ThS. Vũ Quang Dũng

HÀ NỘI-2010

Lời cảm ơn

Lời đầu tiên tôi xin bày tỏ lòng biết ơn sâu sắc tới các thầy giáo PGS. TS. Nguyễn Việt Hà- phó Hiệu trưởng trường Đại học Công nghệ, ĐHQGHN và ThS. Vũ Quang Dũng- Giảng viên bộ môn Công nghệ phần mềm, khoa Công nghệ thông tin, trường Đại học Công nghệ, ĐHQGHN. Các thầy đã hướng dẫn tôi tận tình trong suốt quá trình thực hiện khóa luận cũng như trong suốt năm học vừa qua.

Tôi xin bày tỏ lòng biết ơn tới các thầy, cô giáo trong Khoa Công nghệ thông tin, Trường Đại học Công nghệ, ĐHQGHN. Các thầy cô đã tận tình dạy bảo, chỉ dẫn tôi và luôn tạo điều kiện tốt nhất cho chúng tôi học tập trong suốt quá trình học đại học. Các thầy cô đã tận tình giúp đỡ cho tôi để hoàn thành khóa luận tốt nghiệp này.

Tôi xin cám ơn phòng thí nghiệm Công nghệ phần mềm Toshiba-Coltech, mô hình liên kết giữa trường Đại học Công nghệ và Tập đoàn Toshiba, đã cho tôi những định hướng nghiên cứu hiện đại, theo kịp xu hướng phát triển của thế giới.

Tôi xin cảm ơn các bạn sinh viên trong phòng thí nghiệm đã cho tôi những ý kiến đóng góp giá trị khi thực hiện đề tài này.

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

Hà Nội, 15 tháng 5 năm 2010

Sinh viên

Đinh Anh Thái

Tổng quan

Tổng quan

Khóa luận này mang tới những hiểu biết, kỹ thuật cơ bản về cách thể hiện FlashVideo dựa trên Gnash nhằm hướng tới mục tiêu của dự án tại phòng thí nghiệm Toshiba-Coltech về *"3D Visualization Framework"*. Trong giai đoạn đầu tìm hiểu của dự án, chúng tôi đã tìm hiểu về máy ảo ActionScript và một số kỹ thuật khác liên quan tới công nghệ Adobe Flash, cách thể hiện đối tượng đồ họa 3D thông qua thư viện nguồn mở PaperVision 3D.

Abstract

In this thesis, we present our investigation in Flash technology and its details. This thesis is a part of jointed project of Toshiba-Coltech laboratory in 3D Visualization framework, that takes us to approach the technology from learning new techniques of ActionScript virtual machine and 3D graphics with OpenGLES 2.0 environment. The first part is to be done by me - Dinh Anh Thai, and the other will be done by Le Viet Son.

Tóm tắt nội dung

Công nghệ 3D ngày càng được sử dụng rộng rãi và phổ biến tới người sử dụng, từ người dùng phổ thông tới những chuyên gia công nghệ. Hiện nay, công nghệ 3D đã và đang phát triển mạnh nhằm mang tới một thể hiện đồ họa sống động, gần với cuộc sống thực cho con người. Cùng với sự định hướng của Tập đoàn Toshiba, phòng thí nghiệm công nghệ phần mềm Toshiba-Coltech cũng hướng tới nghiên cứu công nghệ này- Kỹ thuật hiển thị đồ họa 3D trên hệ thống nhúng.

Sau quá trình lựa chọn công nghệ để thực hiện đồ họa 3D trên hệ thống nhúng, chúng tôi hướng tới sử dụng công nghệ Flash- Công nghệ độc quyền cung cấp bởi Adobe Systems Incorprated. Flash với ưu điểm gọn nhẹ, dễ dàng thực thi trên nhiều nền tảng, môi trường khác nhau và được sử dụng phổ biến trên rất nhiều hệ thống khác nhau: hệ thống y tế, PC, thiết bị cầm tay, đồ gia dụng...

Adobe Flash(Tên gọi khác: *Macromedia Flash*) là một công nghệ chứa nền tảng đa phương tiện được sử dụng để tạo hoạt họa, video, và các tương tác gồm MacroMedia Flash- Chương trình tạo ra các tập tin Flash, và Flash Player- Ứng dụng có nhiệm vụ chơi và hiển thị các tập tin Flash. Flash dùng kỹ thuật đồ họa vector và đồ họa điểm(raster graphics). Flash đi kèm với cùng một ngôn ngữ kịch bản riêng gọi là ActionScript, ActionScript được sử dụng để tạo các tương tác, các hoạt cảnh, hành động trong phim Flash.

Trong thời gian thực hiện khóa luận, chúng tôi đã cơ bản nắm được kỹ thuật hiển thị và thực thi đồ họa, cách xử lý sự kiện để hiển thị các đối tượng trong tệp tin Flash, cách để chương trình chơi Flash dùng để thực thi tệp tin SWF. Dựa trên một số mã nguồn mở, tôi đưa ra giải pháp thể hiện 3D trên phần mềm nguồn mở Gnash kết hợp với Tamarin. Hiện tại, Gnash chỉ hỗ trợ thực thi đồ họa 2D cho tệp tin Flash và Tamarin là máy ảo ActionScript nguồn mở được Adobe cùng với Mozilla cung cấp cho cộng đồng nhưng không cung cấp kèm theo cách hiển thị đồ họa cho tệp tin Flash.

Để thực hiện giải pháp, chúng tôi đề xuất mô hình cho việc kết hợp Gnash với Tamarin và cùng với đó là phương pháp thực thi. Bằng việc thêm máy ảo mới, Gnash sẽ hỗ trợ tốt hơn cho ActionScript 3 và qua đó sẽ hỗ trợ thực thi hiệu quả những phiên bản

sau của SWF(phiên bản 9 và 10). Trong ActionScritp 3 đã hỗ trợ những đối tượng cơ bản, hiệu ứng đơn giản cho đồ họa 3D, do đó với mô hình mới này, Gnash sẽ thực thi được đồ họa 3D. Bằng việc sử dụng thư viện đồ họa 3D hỗ trợ cho ActionScript 3, hoàn toàn có thể tạo được những hiệu ứng, phim Flash 3D phức tạp.

Trong khóa luận này, tôi đã thực hiện một chương trình để thể hiện kỹ thuật hiển thị FlashVideo dựa vào Gnash đối với tệp tin SWF và hiển thị tiến trình xử lý các đối tượng ActionScript.

Mục lục

Danh sách hình vẽ

Bảng từ viết tắt

Từ viết tắt	Từ hoặc cụm từ
3D	3 Dimension(ba chiều)
abc	ActionScript Byte Code
AS	ActionScript(Ngôn ngữ kịch bản của công nghệ Flash)
AGG	Anti-Grain Geometry(Engine thể hiện đồ họa 2D)
API	Application Programming Interface
AVM	ActionScript Virtual Machine(Máy ảo ActionScript)
GUI	Graphics User Interface(Giao diện người dùng đồ họa)
LIR	Low-Level Intermediate Representation
LLVM	Low-Level Virtual Machine
MIR	Macromedia Intermidiate Representation
OpenGL	Open Graphics Library(Thư viện đồ họa 2D và 3D)
OpenGLES	OpenGL Embeded Systems(Thư viện đồ họa OpenGL cho hệ thống nhúng)
PC	Personal Computer(Máy tính cá nhân)
PP3D	PaperVision 3D (Thư viện Flash 3D)
JIT	Just-in-time
SWF	Small Web Format hoặc Shockwave Flash
VM	Virtual Machine(Máy ảo)
ZCT	Zero Count Table

Bảng 1: Bảng từ viết tắt

CHƯƠNG 1

Đặt vấn đề

1.1 Thực trạng

Ngày nay, cùng với sự phát triển của khoa học kỹ thuật công nghệ, thiết bị điện tử đã trở thành công cụ hỗ trợ hữu ích trong công việc của con người. Những thiết bị này được sản xuất, sử dụng trong hầu khắp các lĩnh vực của xã hội, từ sản xuất công nghiệp tới sản xuất nông nghiệp và cung cấp dịch vụ. Các thiết bị điện tử làm cuộc sống con người trở nên đơn giản, thuận tiện hơn, tăng năng suất, hiệu quả... Những thiết bị điện tử này, từ những thiết bị lớn như robot trong công nghiệp, những siêu máy tính... tới những thiết bị nhỏ bé, gắn bó với từng cá nhân như máy tính cá nhân, PDA... đã hỗ trợ cho người sử dụng một cách hiệu quả. Do nhu cầu của con người luôn luôn thay đổi, những thiết bị này cũng thay đổi không ngừng, liên tục đổi mới, đưa ra những đặc điểm mới để hỗ trợ cho nhu cầu đó.

Thiết bị điện tử hỗ trợ cá nhân được sản xuất với mục đích cung cấp cho số lượng lớn người dùng, và trong số này phần lớn là người dùng phổ thông. Vì vậy, thiết bị đó phải dễ sử dụng, với giao diện thân thiện. Trong những năm gần đây, kỹ thuật nền tảng sử dụng đồ họa 2D đã có những bước tiến mới với sự xuất hiện của đồ họa 3D đã mang đến những thiết bị đầu tiên sử dụng đồ họa 3D được sử dụng rộng rãi như Tivi 3D, điện thoại di động 3D... Sự phát triển của công nghệ này nhằm nhằm mang tới sự thỏa mãn nhu cầu ngày càng cao của người dùng về thẩm mỹ, về chức năng... của sản phẩm.

Một vấn đề đối với thiết bị hỗ trợ cá nhân nói riêng và thiết bị nhúng nói chung: năng lực hạn chế của bộ xử lý, dung lượng lưu trữ của bộ nhớ và bộ nhớ thực thi... Những ứng dụng thực thi trên PC nhưng không thể thực thi trên thiết bị nhúng nên những kỹ thuật xử lý hình ảnh thông thường từ 2D sang 3D áp dụng cho PC có thể sẽ không được áp dụng đúng đắn với hệ nhúng. Những đặc điểm này dẫn đến nhu cầu phát triển hệ thống đồ họa không phụ thuộc nền tảng(hệ điều hành, năng lực xử lý...) và đặc biệt là tính gọn nhẹ, đơn giản và có khả năng tạo những hiệu ứng 3D một cách mềm mại, uyển

chuyển. Trước nhu cầu đó, chúng tôi đã lựa chọn công nghệ Flash, một công nghệ hiện nay được sử dụng rất phổ biến vì tính trực quan, tính gọn nhẹ và có khả năng hỗ trợ khá tốt đồ họa 3D.

1.2 Hướng tiếp cận

Trong khuôn khổ hợp tác giữa trường Đại học Công nghệ- ĐHQGHN và tập đoàn Toshiba, trung tâm công nghệ phần mềm thuộc tập đoàn đã đưa ra bốn mô hình sử dụng các kỹ thuật đồ họa 3D khác nhau: Compiz Fusion(C/C++), Flash, Qt và Wide Studio. Sau khi so sánh các đặc điểm, ưu và nhược điểm của từng kỹ thuật, trung tâm đã thống nhất dùng Flash để thực thi đồ họa 3D với sự hỗ trợ của PaperVision 3D[1].

Với mục đích định hướng nghiên cứu bắt kịp với trình độ thế giới, được sự hỗ trợ từ trung tâm, chúng tôi tiến hành nghiên cứu kỹ thuật xây dựng 3D cho thiết bị nhúng, đây là xu hướng mới đang được phát triển khá mạnh trên thế giới trong thời điểm này.

Mục đích của quá trình nghiên cứu là xây dựng một chương trình khung[1] hiển thị giao diện người dùng đồ họa 3D dựa trên công nghệ Flash, được thực thi bởi chương trình chơi Flash[2] hỗ trợ *OpenGLES* 2.0. Hướng nghiên cứu này còn bao gồm thuật toán hiển thị 3D trực quan dựa trên những thư viện 3D và thư viện hiển thị trực quan trên công nghệ Flash đã có sẵn(*PP3D[1], Flare[2]*...). Cùng với đó, dự án cần nghiên cứu việc phân tích dữ liệu để hiển thị trực quan 3D dựa trên phương pháp ước lượng. Mục tiêu cần đạt được của dự án là cung cấp một chương trình khung cho việc hiển thị dữ liệu trực quan 3D dựa trên Flash, cùng với những hiệu ứng sinh động, chạy mượt mà trên hệ thống nhúng sử dụng OpenGLES, thực thi bởi chương trình chơi Flash.

Hình 1.1: Ví dụ về hiển thị trực quan 3D

Nhiệm vụ

[1] Framework
[2] Flash Player

Đối với thiết bị nhúng việc tạo ra giao diện đồ họa 3D từ *OpenGLES* 1.1/2.0 trực tiếp là rất phức tạp, vì lẽ đó chúng ta cần phải xây dựng một chương trình khung để giảm thiểu sự phức tạp này. Ngoài ra, một vài sản phẩm được cung cấp bởi *Toshiba* đã được cài đặt ứng dụng Flash như điện thoại di động, tivi số, một số ứng dụng phân tích dữ liệu...

Nhưng các thiết bị của *Toshiba*, những ứng dụng được cài đặt chỉ mới hỗ trợ hoạt họa 2D và cần phải được cải tiến để thực thi 3D trên *Flash Lite*. Toshiba đã có một số ứng dụng về phân tích dữ liệu trực quan và bây giờ cần phải cải tiến để phù hợp với công nghệ Flash.

Trong thời gian nghiên cứu hiện tại, chúng tôi hướng tới mục tiêu xây dựng một hệ thống máy ảo riêng để thực thi Flash và có hỗ trợ 3D, nhằm hướng tới mục tiêu thực thi trên thiết bị nhúng. Để tiến tới mục đích này, chúng tôi tiến hành tìm hiểu chương trình chơi Flash mã nguồn mở Gnash, thư viện Flash 3D: Papervision 3D và đưa ra giải pháp cho việc thực thi trên thiết bị nhúng.

Để thực hiện được mục đích trên, nhằm hướng tới mục tiêu xây dựng đồ họa 3D cho thiết bị nhúng, trong khuôn khổ khóa luận này tập trung chủ yếu vào việc xác định, tìm hiểu và đưa ra kỹ thuật hiển thị Flash trước mắt là trên hệ thống PC, và sau đó sẽ là hệ thống nhúng dựa trên bộ xử lý ARM.

1.3 Phạm vi nghiên cứu

Trong giai đoạn tìm hiểu đầu, chúng tôi cần phải tìm hiểu chương trình chơi Flash nguồn mở- Gnash đưa lên trên thiết bị nhúng và can thiệp vào chương trình này sao cho có thể thực thi hầu hết các phiên bản của ngôn ngữ ActionScript(tương ứng là các phiên bản của SWF) và xử lý đồ họa 3D trên thiết bị nhúng.

Trong khuôn khổ khóa luận này, tôi có nhiệm vụ tìm hiểu kỹ thuật hiển thị FlashVideo, luồng xử lý trong Gnash, cơ chế thực thi phim Flash và đưa ra giải pháp thực hiện đồ họa 3D vào Gnash cũng như chuyển chương trình chơi Flash lên hệ thống nhúng. Nhiệm vụ còn lại tìm hiểu cách thức xây dựng đối tượng Flash 3D, nghiên cứu biện pháp để Gnash thực thi Flash 3D dựa vào PaperVision 3D được giao cho bạn Lê Viết Sơn.

Do đây là một vấn đề phức tạp liên quan đến nhiều lĩnh vực: đồ họa máy tính, kỹ thuật xử lý Flash, công nghệ chạy run-time, bộ thông dịch..., thời gian thực hiện khóa luận này là không đủ nên kết quả đạt được chưa được như mong đợi của mục tiêu đề ra. Trong thời gian tiếp theo, tôi sẽ tiếp tục hoàn thiện đề tài này để đạt được kết quả mong muốn.

CHƯƠNG 2

Cơ sở lý thuyết

Flash được biết đến chủ yếu trên các trang Website với các thể hiện như: quảng cáo, chiếu phim trực tuyến, tạo các thành phần cho trang Web cũng như tạo các game tương tác... bởi sự gọn nhẹ, tính linh hoạt và sinh động. Flash là một công nghệ được phát triển bởi Macromedia và hiện nay được biết tới như một sản phẩm của Adobe Systems. Với Flash, Macromedia và Adobe đưa ra một số kỹ thuật được áp dụng cho công nghệ này như máy ảo, ngôn ngữ kịch bản hành động(ActionScript), tệp tin SWF, chương trình chơi Flash(*Flash Player*)...

Do đặc điểm đó, Flash có những kỹ thuật riêng để thể hiện đối tượng đồ họa, đặc biệt là đồ họa 3D. PaperVision 3D là một thư viện hỗ trợ đồ họa 3D cho Flash. Với những cơ sở lý thuyết được giới thiệu, chúng tôi tiến tới xây dựng một hệ thống riêng độc lập với công nghệ Adobe Flash(gồm máy ảo AS, quá trình biên dịch và thực thi) trong thời gian tới.

2.1 ActionScript - Flash - SWF

2.1.1 Ngôn ngữ ActionScript

ActionScript[3] là một ngôn ngữ lập trình hướng đối tượng[1] với các đối tượng như *class, interface và packages* được dùng như lệnh kịch bản(*script*) cho các phim dùng *Adobe Flash*.

- ActionScript là ngôn ngữ kịch bản dựa trên *ECMAScritpt*[4]. ActionScript được sử dụng chính cho mục đích phát triển các thành phần trên Website và thi hành trên ứng dụng *Adobe Flash Player*(Tệp tin Flash- phần mở rộng có dạng *SWF* là một dạng được sử dụng trên các trang web).

[1] Object-oriented programming language

- ActionScritpt được thiết kế để điều khiển những hiệu ứng hình ảnh vector 2D đơn giản được tạo bởi *AdobeFlash*. Những phiên bản sau này có thêm chức năng cho phép tạo trò chơi trên web và vô vàn những ứng dụng Internet với dữ liệu media như âm thanh và hình ảnh.
- ActionScript bắt đầu là một ngôn ngữ kịch bản cho công cụ *Macromedia Flash* và hiện tại được phát triển bởi *Adobe Systems* như là *Adobe Flash*. Những phiên bản ban đầu của *Flash* chỉ cung cấp giới hạn những đặc điểm tương tác như những hành động đơn giản gồm *play, stop, getURL, gotoAndPlay*
- Từ phiên bản *Flash 4* được phát hành năm 1999, tập hợp những hành động đơn giản trở thành ngôn ngữ kịch bản nhỏ với những khả năng mới như cho phép khai báo biến, phương thức, câu lệnh rẽ nhánh, lặp.

2.1.2 Công nghệ Flash

Flash[5] là một công nghệ nền đa phương tiện thường được sử dụng để tạo hoạt họa, video, tương tác với các trang web, và để phát triển những ứng dụng trên Internet(trò chơi, quảng cáo. . .). Flash sử dụng kỹ thuật đồ họa vector và đồ họa điểm để tạo hoạt họa cho chữ, hình vẽ, ảnh, hỗ trợ thực thi luồng âm thanh, video. Nó chứa bên trong một ngôn ngữ hướng đối tượng gọi là *ActionScript*. Để hiển thị nội dung Flash, chúng ta phải sử dụng một số phần mềm chơi Flash như *Adobe Flash Player* hoặc *Flash Lite* trên thiết nhúng.

Flash được giới thiệu lần đầu vào năm 1996 bởi Macromedia và hiện tại là Adobe Systems. Tiền thân của ứng dụng Flash là *SmartSketch*- Một ứng dụng cho máy tính dùng bút chạy trên hệ điều hành *Penpoint OS* phát triển bởi *Jonathan Gay*, sau đó được đưa vào phát triển trên hệ điều hành *Microsoft Windows* và *Mac OS*.

Cho đến khi Internet trở nên phổ biến *SmartSketch* được biết tới với tên gọi mới là *FutureSplash*. Năm 1995, *SmartSketch* được thay đổi với đặc điểm hoạt họa frame-by-frame [2] với tên gọi mới là *FutureSplash Animator*.

Năm 1996, Macromedia đưa ra tên gọi mới là Flash, sự viết tắt của *"Future"* và *"Splash"*. Tệp tin Flash được gọi với tên *Shockwave Flash* và có phần mở rộng là *.swf*.

Flash là một công nghệ đóng hoàn toàn, được độc quyền sở hữu 100% bởi Adobe Systems, một công ty công nghệ lớn của Mỹ với những sản phẩm nổi tiếng như: *Adobe PhotoShop, Adobe Reader, Adobe Creative Suite*. . . Các công nghệ của hãng được sử dụng rộng rãi trên toàn thế giới như chương trình đọc tài liệu Adobe Reader - đọc tệp tin *.pdf*[3], chương trình chơi Flash được bổ sung vào trình duyệt web như Adobe Flash

[2] Khung hình nối tiếp khung hình
[3] Portable Document File, một định dạng tài liệu rất phổ biến trên Internet

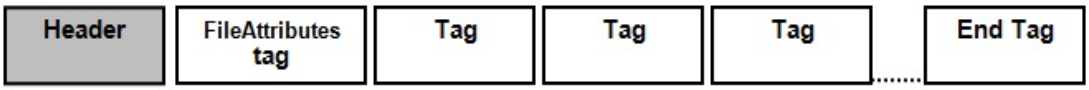
Player tới những sản phẩm chuyên dụng như Adobe Creative Suite...

Do được sử dụng rộng rãi trên toàn thế giới, Adobe phổ biến với số lượng khá lớn các tài liệu hướng dẫn sử dụng, hướng dẫn thi hành, cơ chế... Tuy nhiên, những tài liệu này chủ yếu hướng tới đối tượng là người sử dụng cuối cùng(người dùng sản phẩm của hãng) và không hướng tới mục đích cung cấp tài liệu đặc tả kỹ thuật cho những người phát triển dựa trên công nghệ này. Việc một sản phẩm được phổ biến một cách rộng rãi không đồng nghĩa với việc đó là công nghệ mở. Nhất là khi những sản phẩm này được kiểm soát hoàn toàn và chỉ được giới thiệu bởi một mình Adobe. Họ có quyền quyết định tới tương lai của công nghệ này...

2.1.3 Tệp tin Flash - SWF

SWF(ShockWave Flash)[6] hoặc tên gọi khác là *Small Web Format* hay *Flash movies* hoặc *Flash games*. Tệp tin Flash có phần mở rộng là .swf và có thể được sử dụng như một thành phần thêm vào trên web.

SWF có thể được tạo ra từ một vài sản phẩm của Adobe như *Flash, FlexBuilder*(hoặc dạng *MXMLC*). Mọi tệp SWF đều bắt đầu bằng phần Header, với 3 byte dấu tiếp theo là những tag chứa các thông tin như định dạng, định nghĩa dữ liệu, các hằng... được đặc tả chi tiết trong tài liệu mô tả SWF[6]. Thuộc tính *FileAttributes* chỉ cần thiết cho SWF8 và các phiên bản sau(hình vẽ 2.1).



Hình 2.1: Cấu trúc file SWF

Trong file SWF, các tag có thể có là *tag điều khiển(Control Tag)* và *tag định nghĩa(Defination Tag)*.

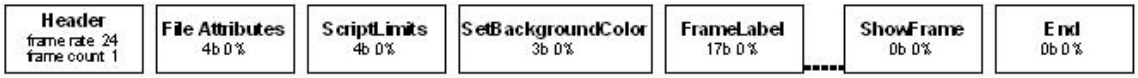
- Tag định nghĩa định nghĩa một đối tượng được biết đến như là đặc điểm của đối tượng được lưu trữ trong danh sách thuật ngữ.
- Tag điều khiển chứa nhiều đặc điểm và cách điều khiển luồng của file, quản lý một vài mặt tổng thể của files, frames và playback của files SWF như *setBackgroundColor, FrameLabel*.
- Flash Player xử lý tất cả các tags của file SWF cho đến khi một tag gọi là *ShowFrame* được gọi tới. Tại thời điểm này, danh sách hiển thị được chuyển tới màn hình và Flash Player tiếp tục gọi frame tiếp theo để xử lý.

Hình vẽ 2.2 là các tag có trong tập tin SWF sau khi được phân tích thành dạng XML[4]. Chúng ta có thể dễ dàng nhận thấy cấu trúc của tệp tin SWF bao gồm tuần tự các tag nối tiếp nhau bao gồm các thuộc tính, các giá trị... Ví dụ đối với thẻ *Header* có các thuộc tính định nghĩa cho phim Flash: số lượng Frame(Frame count), tỉ lệ Frame(Frame rate), tiếp đó là một danh sách các tag nối tiếp nhau như File Attributes(các thuộc tính của file SWF: chứa ABC tag hay DoAction tag, chứa dữ liệu Meta: hasMetaData...). Với ví dụ này ta có thể thấy đây là tệp tin được sinh ra từ AS 3 vì chứa DoABC tag, tiếp theo đó là các thông tin định nghĩa các đối tượng hằng(chuỗi, số nguyên...), các *QName*[5]- định nghĩa một định danh và thuộc tính mới, định nghĩa các đối tượng, các phương thức trong AS...

Hình 2.2: Ví dụ các tag có trong tập tin abc sau khi giải mã(Flash-SWF)

Hình vẽ 2.3 thể hiện một phần của nội dung tệp tin nhị phân SWF ứng với mô hình được mô tả trong hình vẽ 2.1.

[4]Ngôn ngữ đánh dấu mở rộng
[5]qualified name



Hình 2.3: Minh họa cho nội dung tệp tin SWF

Qua quá trình phân tích này, chúng ta dễ dàng nhận thấy sự tương đồng của hình 2.3 với hình 2.1.

2.2 ActionScript Virtual Machine(AVM)

Để tệp tin Flash có thể thực thi trên mọi nền tảng khác nhau mà không phụ thuộc vào hệ điều hành(Windows, Linux, Solaris, MacOS...) cũng như kiểu PC(Mac, Solaris...), *Flash* đưa ra giải pháp dùng máy ảo ActionScript(*ActionScript Virtual Machine- AVM)*. Cũng giống như công nghệ Java của Sun với máy ảo Java, máy ảo ActionScript thực thi qua hai giai đoạn: dịch mã trung gian và thông dịch thành mã thực thi trên máy vật lý.

2.2.1 Giới thiệu AVM

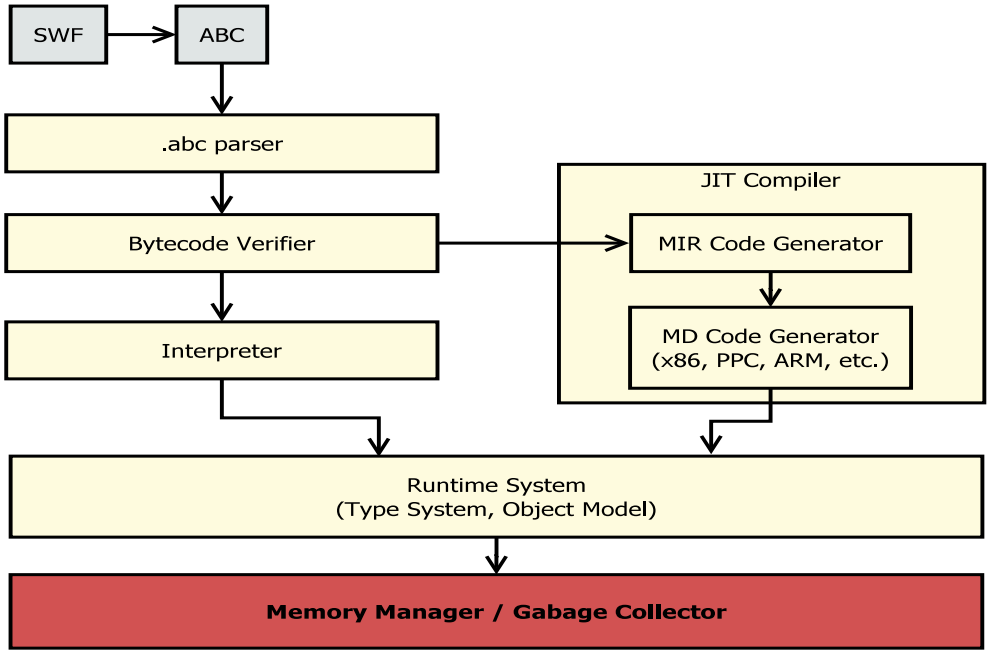
AVM cài đặt bên trong một cơ chế gọi là run-time compiler[6] để chuyển những chỉ lệnh từ AVM tới những chỉ lệnh xử lý đặc biệt của bộ vi xử lý(processor-specific intructions).

Máy ảo AVM được thiết kế để thực thi mã và những phương thức thân chứa khái niệm của những thông tin về phương thức(method infomation), vùng dữ liệu cục bộ(a local data area), vùng chứa những hằng số, vùng heap với cơ chế non-premititve cho đối tượng dữ liệu được tạo ra trong lúc thực thi và một môi trường thực thi run-time(hình vẽ 2.4[7]).

2.2.2 Kiến trúc của AVM

Đầu vào của máy ảo ActionScript là file có phần mở rộng là abc[7], sau khi qua quá trình phân tích của tệp tin *.swf*. Quá trình *.abc parser* để phân tích những thành phần của file .abc thành những mã bytecode để truyền vào cho quá trình tiếp theo là xác thực bytecode(Bytecode Verifier). Tại đây sẽ được chuyển sang quá trình chạy JIT nếu mã bytecode còn chứa những chỉ lệnh phức tạp và quá trình thông dịch khi mã bytecode chỉ

[6]Biên dịch lúc chạy
[7]Dùng abcdump để tạo file .abc



Hình 2.4: Kiến trúc AVM

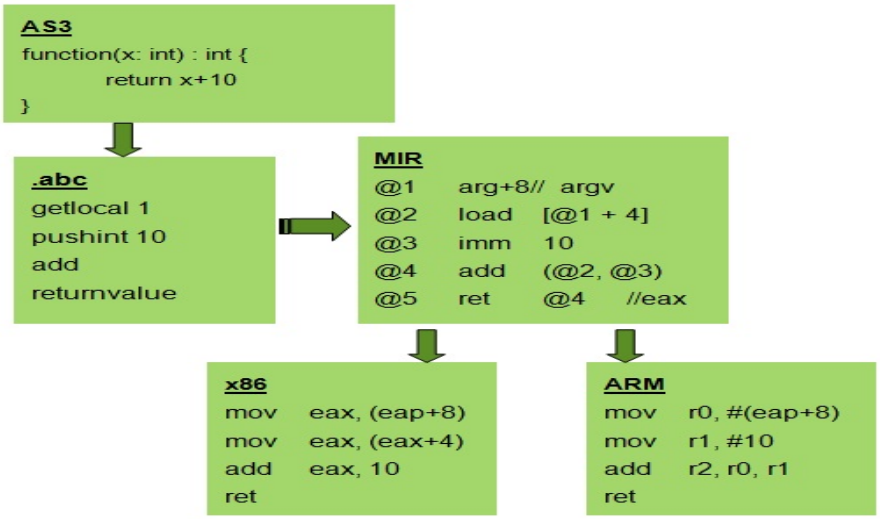
chứa những chỉ lệnh đã có sẵn. Kết thúc quá trình này, máy ảo sẽ chuyển sang quá trình thực thi đối tượng trên hệ thống thật(Runtime System), chuyển toàn bộ chỉ lệnh từ máy ảo sang chỉ lệnh cho bộ xử lý máy thật và được quản lý bởi trình quản lý bộ nhớ và bộ dọn dẹp bộ nhớ.

Để một chương trình Flash được thực thi trên thiết bị sử dụng, chúng phải qua một quá trình chuyển mã gồm nhiều giai đoạn(hình vẽ 2.5). Ban đầu, người phát triển mã hóa chương trình dưới dạng AS, sau đó chương trình biên dịch FlexSDK cho ra mã trung gian chứa trong file *.abc* và MIR. Các mã trung gian với mục đích chính để chương trình được sinh ra không phụ thuộc môi trường, phương thức thực thi cũng như vận chuyển dễ dàng giữa các môi trường như truyền qua mạng, các hệ thống Windows, Linux, Mac OS... Từ các mã trung gian này, máy ảo với bộ biên dịch JIT mới bắt đầu sinh ra mã thật tùy thuộc vào hệ thống đang cài đặt bộ biên dịch này.

MIR[8] là một dạng mã được dùng cho máy ảo Flash, mã này độc lập với mã máy tính vật lý. Ưu điểm của MIR là đơn giản, gần với mã máy vật lý, là thành phần trung gian giữa bytecode và JIT, là đầu vào của JIT. MIR được thiết kế để tối ưu quá trình biên dịch giữa mã chương trình với mã máy, với MIR, người lập trình không cần quan tâm tới môi trường thực thi của chương trình cuối cùng.

File .abc được xử lý trong AVM qua bốn bước chính là nạp, liên kết, xác thực và thi hành [8].

[8]Mã máy Macromedia trung gian



Hình 2.5: Quá trình chuyển mã của công nghệ Flash

- Trong quá trình nạp, file .abc được đọc vào trong bộ nhớ và giải mã, phân tích.
- Trong quá trình liên kết, một vài tên được tham chiếu từ vùng riêng của cấu trúc file ABC để được xử lý và sau đó trả lại kết quả là một cấu trúc dữ liệu phức tạp hơn rất nhiều liên kết các đối tượng cùng nhau.
- Quá trình xác thực là tương đối giữa các đối tượng, tên đối tượng được nhắc đến phải rõ ràng, kết quả theo tập hợp phải được mạch lạc . . .
- Quá trình thực thi, thể hiện mã bytecodes biên dịch trong file ABC và chạy thông dịch suốt quá trình thực hiện tính toán. Trong quá trình này, việc xác thực xảy ra liên tục với luồng chỉ lệnh và nội dung thực thi chứa trong stack: chỉ lệnh không được bên ngoài mảng dữ liệu bytecode đã có, chỉ lệnh phải chứa một kiểu phương thức chính xác. . .
- Quá trình xác thực liên tục được thực hiện ở mỗi bước, ở bước nào có lỗi, AVM sẽ đưa ra một thông điệp là *VerifyError* và thông điệp này có thể được bắt trong quá trình thực thi bởi chương trình.

Dựa vào kiến trúc AVM và mục tiêu của dự án, với mục đích chính là thực thi Flash 3D trên thiết bị nhúng, tôi đặc biệt quan tâm tới các quá trình: chuyển tệp *SWF* sang dạng *.abc*, chuyển từ *.abc* vào quá trình *.abc parser* và đặc biệt là quá trình *JIT Compiler* vì nó liên hệ trực tiếp tới quá trình chuyển sang mã máy của hệ thống nhúng.

Mã máy(Bytecode) được lưu trữ trong dạng nhị phân có phần mở rộng là *SWF*, được thông dịch bởi *Virtual Machine* của *Flash Player*. Mã máy chứa các kiểu để định nghĩa những thành phần như: dữ liệu hằng, chuỗi, số. . . , định nghĩa không gian tên(*namespace*), định nghĩa phương thức, định nghĩa kiểu, ngoài ra mã máy còn cần có khả

năng mô tả kiểu, phương thức được định nghĩa lại từ những kiểu cơ bản đã có sẵn. Trong đó điều không thể thiếu là bảng ngoại lệ, một cơ chế để xử lý và trả lại cho chương trình những hành động không phù hợp với máy ảo. Mã máy được đọc vào và lưu trữ trong một ngăn xếp định hướng(Stack orientd abstract machine). Ngoài ra trong mã máy còn chứa các mã như tạo đối tượng, truy cập địa chỉ(slot access), tìm kiếm thuộc tính(property search).

Cơ chế biên dịch thời gian chạy đó là cơ chế xác thực lại quá trình thực thi, thông dịch[9] và sau đó là 2 quá trình biên dịch JIT(gồm dịch từ mã máy ảo và dịch mã máy ảo tới mã máy thật vật lý) và đi cùng với cơ chế này là cơ chế dọn dẹp bộ nhớ(*Garbage Collector*)- cơ chế thu hồi bộ nhớ, con trỏ lạc, không còn được sử dụng.

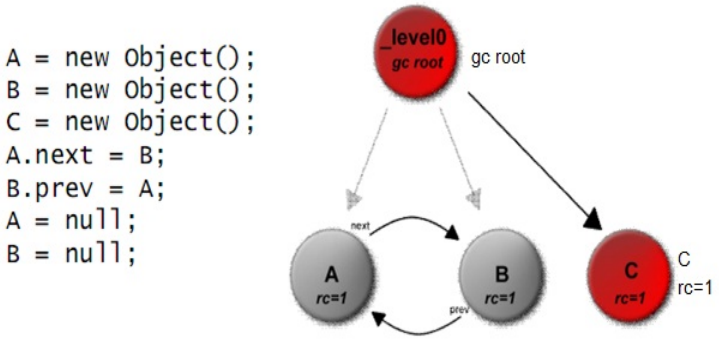
2.2.3 Bộ dọn dẹp bộ nhớ AVM

AVM Garbage Collector[7] một cơ chế quản lý tự động thu hồi bộ nhớ khi tham chiếu không còn được dùng đến, giải phóng tài nguyên cho hệ thống khi đối tượng không còn được sử dụng. Nhiệm vụ chính của bộ dọn dẹp bộ nhớ là tìm kiếm những đối tượng trong chương trình có thể không được sử dụng tiếp ở các lần chạy tiếp theo và thu hồi tài nguyên được sử dụng bởi những đối tượng này.

- Thư viện sử dụng lại(Reusable library)
  - Chỉ thu hồi rác
  - Thêm mới, xóa (Không quản lý bộ nhớ)
  - Gỡ lỗi
- Đặc điểm
  - Kế thừa những kỹ thuật chính được sử dụng trong Java như: sử dụng *Heap*, cơ chế xóa bộ nhớ khi đối tượng không còn được sử dụng, cách kiểm tra đối tượng còn được sử dụng, đếm tham chiếu...
  - Sử dụng thuật toán làm chậm tham chiếu đếm[10]: Chỉ duy trì RC[11] cho heap này tới heap khác, bỏ qua các Stack và các thanh ghi, đặt biến đếm Zero trong Zero Count Table(ZCT), quét Stack khi ZCT là đầy, xóa các đối tượng trong ZCT khi không tìm thấy trong stack
  - Tập gia tăng[12]: mỗi 30ms là giới hạn cho việc làm mới tập hợp, tập hợp cung cấp con trỏ thông minh, sự gia tăng của tập hợp có thể bị ngắt một cách trực tiếp(incremental = interruptible)

[9] leaner interpreter
[10] DRC: Deferred Reference Counting
[11] Reference Counter: Bộ đếm tham chiếu con trỏ
[12] Incremental Collection

Khi một vùng nhớ được cấp phát, với thuật toán RC[13] mỗi một vùng nhớ mới sẽ có một biến đếm đi kèm với nó. Tùy vào đối tượng này được cấp phát có quan hệ với những đối tượng khác mà bộ đếm gắn cho một biến đếm khác nhau. Sau đó, đối với những đối tượng được gán nhãn cao nhất- khi không được sử dụng nữa(không được tham chiếu bởi đối tượng khác) thì GC sẽ sử dụng cơ chế xóa để xóa đối tượng này. Garbage Collection không thể biết vùng nào còn trống để có thể cấp phát bởi nó được quản lý bởi Hệ điều hành do đó GC chỉ có nhiệm vụ giải phóng vùng này để trao lại quyền quản lý cho hệ điều hành. Chi tiết được thể hiện dưới hình 2.6.



Hình 2.6: Cơ chế của bộ dọn dẹp bộ nhớ

2.2.4 Bộ xác thực AVM

AVM Verifier là một cơ chế xác thực hành động, mã máy có trong kiến trúc AVM. Hành động xác thực không xảy ra tất cả cùng một lúc. Mã xác thực được đặt vào máy ảo cho đến khi hành động thực sự diễn ra như một số đối tượng phụ thuộc được tải vào máy ảo và sau đó tham chiếu được chuyển tiếp nếu còn tiếp tục hoặc trả lại kết quả cho quá trình thông dịch tiếp sau. Để thực hiện cơ chế này, những đặc điểm chính của bộ xác thực như sau:

- Cấu trúc tổng thể, nhiệm vụ
  - Đảm bảo nhãn hoạt động đúng với chỉ lệnh
  - Không bị sai lạc ở cuối của mã
  - Tham chiếu tới tài nguyên một cách chính xác, đúng đắn
- Kiểu an toàn(Type safety)
  - Dataflow phân tích tĩnh
  - Ràng buộc kiểu

[13]Reference Counting

  - Làm giảm tốc độ xử lý

- Thêm lựa chọn việc tạo mã MIR(*Macromedia Intermediate Representation*) để tạo chỉ lệnh(IR[14]) trong khi xác thực và có thể chạy độc lập quá trình gồm: xác thực và tạo IR.

2.2.5 Bộ thông dịch AVM

AVM Interpreter là cơ chế thực thi mã sau khi qua quá trình xác thực dữ liệu .abc trong file SWF, giá trị được tạo ra là 32 bit(hình 2.7).

- Ngăn xếp định hướng
  - Không phân luồng
  - Mã hóa 32bit
  - Thực thi trực tiếp từ bộ đệm kiểm chứng(verified buffer)
  - Không có sự sửa mã bytecode

2.2.6 AVM Just-in-Time Compiler(JIT)

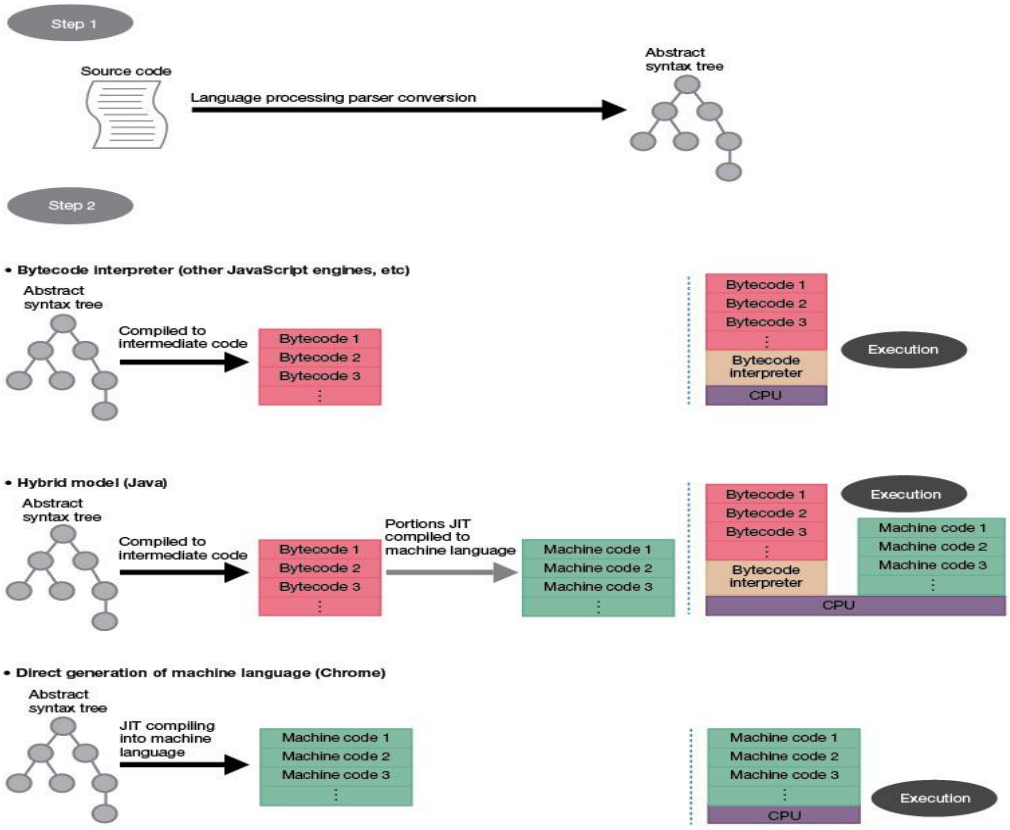
Một hệ thống để chuyển một ngôn ngữ bậc cao hoặc mã máy tới ngôn ngữ tự nhiên ngay lập tức trước khi chuyển tới quá trình xử lý thực sự của mã máy vật lý. Just-In-Time là một cơ chế chính được cài đặt cho công nghệ máy ảo như Microsoft .NET và Java. Sử dụng MIR là mã gián tiếp để chuyển mã máy ảo sang máy thật(hình 2.7).

- Lần chạy đầu tiên
  - Trình bày trực tiếp(MIR) những hành động xảy ra đồng thời với việc kiểm chứng
  - Liên kết cùng nhau
  - Hằng gấp(Constant folding)
  - Copy và truyền hằng(copy and constant propagation)
  - Loại bỏ những biểu hiện con chung(common sub-expression elimination (CBE))
- Lần chạy thứ 2
  - Sinh mã gốc
  - Lựa chọn chỉ lệnh

[14]Intermediate Representation

– Đăng ký, cấp phát

– Loại bỏ mã không còn dùng nữa(Dead code elimination (DCE))



Hình 2.7: Các quá trình của AVM

2.3 Tamarin

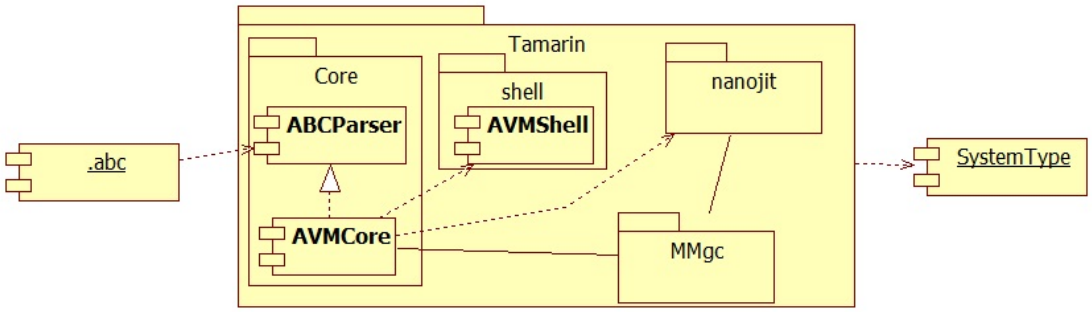
Trong năm 2006, Adobe phân phối công cụ(engine) AS 3 như là một dự án mã nguồn mở cho cộng đồng cùng với dự án Mozilla. Tên của dự án mới này là *Tamarin*. Kiến trúc của dự án này đã thay đổi, phát triển theo thời gian, các đặc điểm chính của những phương pháp tiếp cận khác nhau được mô tả ngắn gọn như dưới đây.

2.3.1 Giới thiệu Tamarin

Mục đích của dự án Tamarin là cung cấp mã nguồn mở với hiệu năng cao cho ngôn ngữ AS 3. "Tamarin" được cài đặt gồm 2 thành phần: bộ biên dịch JIT với hiệu năng cao và bộ thông dịch.

Máy ảo Tamarin được sử dụng trong chương trình Adobe Flash Player và được cho

phép sử dụng trong các dự án bên ngoài Adobe. Bộ biên dịch JIT(thường gọi là "Nano-JIT") là một sự hợp tác phát triển giữa hai bên: Tamarin và Mozilla TraceMonkey[9].



Hình 2.8: Kiến trúc Tamarin

Trong phần này, chúng tôi chỉ giới thiệu những hiểu biết cơ bản về *Tamarin*, để hiểu chi tiết hơn các bạn có thể tìm thấy trong khóa luận "Phương pháp thực thi đồ họa Flash 3D dựa trên PaperVision 3D"[15].

Dựa vào kiến trúc Tamarin(hình vẽ 2.8) và AVM(hình vẽ 2.4), chúng ta dễ dàng thấy rằng đầu vào của Tamarin là file *.abc* còn đầu vào của một chương trình chơi Flash là tệp tin *SWF*. Tamarin có thành phần chính là gói *core* chứa AVMCore và thành phần biên dịch JIT, bộ quản lý bộ nhớ MMgc[16]. Tệp tin *.abc* sau quá trình phân tích(*ABCParser*) được đưa vào máy ảo qua AVMCore. Sau đó, giống như kiến trúc AVM, Tamarin đưa ra những mã máy của máy vật lý.

Tamarin là máy ảo chỉ dùng cho những tệp tin được sinh ra từ AS 3. Trong quá trình phân tích tệp tin *ABCparser*, Tamarin đưa ra những thuộc tính, những tag, hàm theo chuẩn AS 3(vì ngôn ngữ AS 3 có nhiều thay đổi so với AS 1, 2 trước đó). Tệp tin AS 3 chứa tag mới như *DoABC, DoABCDefine*...trong khi với AS 1, 2 chứa tag *DoAction, DoInitAction*...

Bộ biên dịch ActionScript là một thành phần có trong bộ công cụ phát triển Flash: Flex SDK[10].

2.3.2 Mục đích dự án Tamarin

Mục đích chính của dự án Tamarin là hỗ trợ đa nền tảng của phần cứng, bao gồm cả ARM cho hệ thống nhúng và X64 cho hệ điện toán 64 bit ngoài hệ thống x86 thông thường. Việc phát triển Tamarin không ngoài mục đích:

- Cải tiến hiệu năng thực thi thời gian thực(run-time)

[15] Khóa luận của bạn Lê Viết Sơn
[16] Macromedia Garbage Collector

- Phát triển bộ biên dịch thời gian thực(run-time compiler)

2.3.3 Tamarin central

Tamarin central là phiên bản ổn định của bộ công cụ. Được Adobe phân phối chính thức với sự cộng tác của Mozilla, nó bao gồm một kiến trúc cơ sở cơ bản và mã đã được biên dịch ổn định. Công cụ này bao gồm cả bộ thông dịch và bộ công cụ JIT(*NanoJIT*)-thành phần trung gian giữa(LIR) và một số nền tảng cụ thể(x86, x86-64, PPC, ARM).

2.3.4 Tamarin redux

Tamarin redux là một sản phẩm mới được phát triển của dự án. Nó được thực hiện bằng cách cải tiến phiên bản trước(*Tamarin central*). Điều thú vị nhất của dự án là xuất hiện mã trung gian *Forth*- Được sử dụng như là một back-end cho quá trình biên dịch mã máy AS. Thực tế, công cụ này chứa 2 lần biên dịch. Mã AS đầu tiên được chuyển tới mã trung gian *Forth* và sau đó được chuyển tới mã trung gian của *NanoJIT*- MIR, rồi mới kết thúc quá trình biên dịch.

Forth bản thân nó là một ngôn ngữ cơ sở ngăn xếp(stack based language).

2.4 PaperVision 3D

PaperVision 3D[11] là một thư viện đồ họa Flash 3D nguồn mở được đưa ra bởi bên thứ ba, cung cấp giao diện lập trình ứng dụng cho phép người lập trình tạo các chương trình Flash 3D dựa trên FlexSDK.

Chi tiết hơn về PaperVision 3D được giới thiệu trong khóa luận của bạn Lê Viết Sơn về "Phương pháp xử lý đồ họa 3D của PaperVision 3D". Trong phần này tôi xin giới thiệu những hiểu biết cơ bản về PaperVision 3D.

2.4.1 Giới thiệu

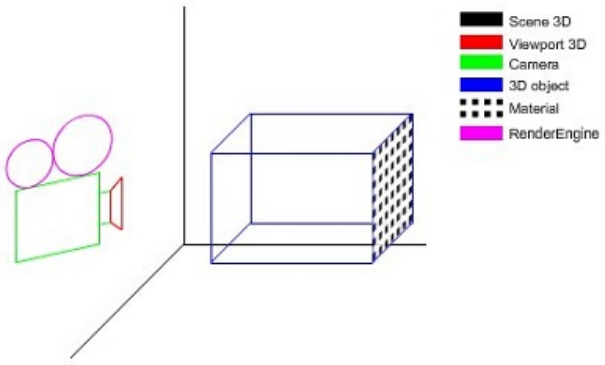
PaperVision 3D được hỗ trợ sử dụng trên các công cụ tạo Flash được cung cấp bởi Adobe như: Adobe Flash, Flex Builder... Với việc sử dụng PP3D, tệp tin Flash được tạo ra có đồ họa bắt mắt, sinh động với nội dung hấp dẫn, lôi cuốn như các game đối kháng, truyện tranh tương tác có đồ họa phức tạp...

PaperVision 3D gồm 2 gói chính là:

- Gói ascollada(*AS COLLAborative Design Activity*): những kịch bản ActionScript được thiết kế sẵn để tạo ra file .dae có thể chuyển đổi được cho ứng dụng tương tác 3D.
- Gói PaperVision 3D: gồm các thành phần tạo 3D và xử lý sự kiện, tương tác người dùng.

2.4.2 Đặc điểm PaperVision 3D

PaperVision 3D sử dụng các kỹ thuật trong đồ họa máy tính để xây dựng hình ảnh 3D trên màn hình, nó bao gồm các thành phần chính như: *Camera, Viewport, Scene, Render*(được thể hiện trong hình vẽ 2.9).



Hình 2.9: Các thành phần thể hiện đồ họa 3D

Scene: là thành phần chứa tất cả các đối tượng của không gian 3D. Các đối tượng này được lưu trữ trong cấu trúc dữ liệu dạng cây.

Camera: Xác định điểm nhìn mà chúng ta đang xem *Scene*, được dùng để thay đổi góc nhìn, hướng, trọng tâm hiển thị. . .

Viewport: là vùng chứa những đối tượng mà *Camera* có thể nhìn thấy, thể hiện một phần trong *Scene 3D*.

2.5 Gnash

GNU[12] với mục đích cung cấp cho cộng đồng một hệ thống những phần mềm được sử dụng, phân phối tự do, tự do tùy biến phần mềm đã thực hiện kế hoạch tạo chương trình chơi Flash độc lập với Adobe Flash Player.

2.5.1 Giới thiệu

Gnash[13] là một phần mềm ứng dụng mã nguồn mở thuộc dự án GNU[12] dùng để thực thi các tệp tin Flash một cách độc lập hoặc đóng vai trò là plugin hỗ trợ trình duyệt web(tương thích với trình duyệt Mozilla, Firefox, Konqueror). Phiên bản Gnash hiện tại(0.8.7) hỗ trợ tới phiên bản SWF 9, ActionScript[3] 2.0 và đang tiếp tục phát triển để hỗ trợ ActionScript 3.0.

2.5.2 Kiến trúc

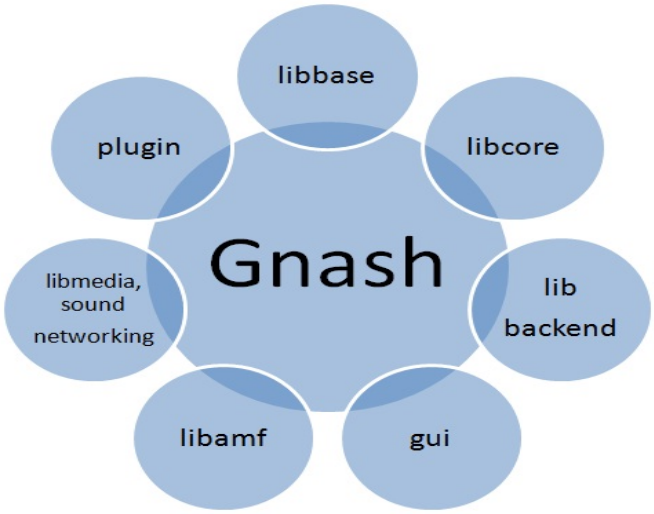
Mã nguồn Gnash gồm các gói: *libcore, libbase, backend, GUI, doc, cygnal, libltd, libamf, libmedia, libnet, libsound, utilities*. Dựa vào chức năng của từng gói và mục tiêu của dự án, tôi đã đặc biệt quan tâm tới các gói: *libcore, libbase và backend*.

- *libcore*:
  Xử lý, đưa ra cơ chế chủ yếu để thực thi tệp tin Flash:
  - Định nghĩa các đối tượng cơ bản của AS trong gói *asobj* như kiểu nguyên, màu sắc, hành động, đối tượng...
  - Các lớp được định nghĩa bởi AS trong gói *asobj/flash*
  - Định nghĩa các tag có trong cấu trúc tệp tin SWF[6] trong gói *swf*
  - Cách thức chuyển dữ liệu từ mã máy nhị phân(tập tin Flash) sang mã máy ảo AS(AVM) trong gói *swf* và gói *parser*
  - Xây dựng máy ảo AS(AVM) để thực thi các mã máy(ActionScript Byte Code)- Tương tự như AVM trong Flash Player của Adobe
  - Cách thức chuyển hành động từ AVM sang máy vật lý
  - Các quá trình xử lý các đối tượng (asobj), mã bytecode Flash (abc), "decoding",...
- *libbase*:
  Bao gồm các lớp nền tảng: lớp hỗ trợ nhập xuất(từ tệp tin, từ URL), thực thi tài nguyên của máy vật lý, quản lý bộ nhớ, con trỏ(bộ thu hồi bộ nhớ- Garbage Collector), các đối tượng đồ họa 2D...
- *backend:*
  Bước xử lý trung gian để thể hiện hình ảnh cuối cùng ra màn hình, hỗ trợ *AGG, OpenGL, Cairo*. Backend sử dụng các hàm, giao diện(interface) được cung cấp bởi *libcore, libbase*,...Tôi quan tâm chủ yếu đến phần hỗ trợ OpenGL của *backend*. Do Flash không có các thành phần cơ bản như: đường tròn, hình chữ nhật, tam

giác,... các đối tượng đều được chuyển về điểm và đường cong trong khi OpenGL chỉ hiểu các thành phần cơ bản như điểm, đường tròn, tam giác, hình chữ nhật. Mục tiêu là chỉnh sửa *backend* để Gnash hỗ trợ OpenGLES.

- *gui:*
  Sử dụng *backend* và kết quả thực thi từ VM, phân tích(*parser* trong libcore) để cho ra hình ảnh trên màn hình hiển thị.

Các gói được sử dụng trong Gnash(hình vẽ 2.10):

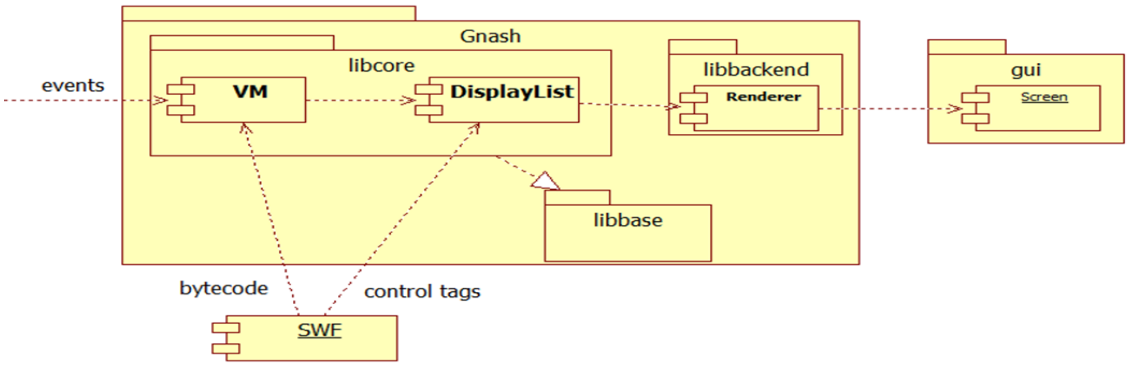


Hình 2.10: Các thành phần trong Gnash

Chú thích(các gói chứa trong *libcore*):

- *abc* định nghĩa, định dạng các tag, hàm, đối tượng... trong các khối mã bytecode(đơn vị tính theo byte vật lý) theo dạng mã máy AS
- *swf* định nghĩa các tag trong SWF: control, action, sound... các tag này được VM thực thi
- *vm* thực thi các tag sau khi được nhận vào mã máy(*abc*)
- *parser* đóng vai trò phân tích tệp tin SWF, kết quả cho ra là *SWFMovieDefinition* mà phần *gui* có thể hiểu và hiển thị được

gui sử dụng *backend* và kết quả nhận được từ *parser* để hiển thị ra màn hình cuối. Tuần tự quá trình xử lý(hình vẽ 2.11):



Hình 2.11: Quá trình xử lý qua các thành phần Gnash

Chú thích:

- Tệp tin SWF chứa 2 thành phần chính là tag điều khiển(control tags) và mã máy(bytecode). Tag điều khiển có thể có: *PlaceObject, RemoveObject, SetBackgroundColor, ShowFrame, StartSound*...tag định nghĩa bao gồm: *DefineShape, DefineBitsJPEG2, DefineSound, SoundStreamBlock*...
- VM chứa các thành phần: *JIT compiler, garbage collector, application domains*, và chuyển tới DisplayList
- DisplayList Cấu trúc dữ liệu trung tâm để hiển thị mọi thứ lên trên màn hình
- Renderer: chuyển DisplayList tới bề mặt hiển thị hình ảnh(Màn hình)

Chú ý: Lúc xử lý và có các tags rồi thì VM mới bắt đầu được khởi tạo chứ không khởi tạo ngay từ đầu.

Trong VM có các hàm để nhận biết phiên bản của SWF và AVM. Ngoài ra, AVM với chức năng là xử lý mã máy ở dạng *.abc* và bắt các sự kiện trong lúc thực thi phim Flash. Khi AVM đọc được tag điều khiển ví dụ như *ShowFrame*, nó sẽ đưa toàn bộ những đối tượng, thuộc tính...đang được xử lý trong nó sang *DisplayList* để thể hiện lên trên màn hình hoặc với tag *PlaceObject* đọc được từ đầu vào, AVM sẽ thực hiện di chuyển đối tượng chứa trong tham số của *PlaceObject* tới vị trí mới có trong mã máy đang đọc và thể hiện lên màn hình thông qua việc chuyển sang *DisplayList*.

Với những đặc điểm này, VM có trong Gnash phải sử dụng tới rất nhiều cơ chế để thực hiện việc kiểm soát dữ liệu vào ra: quản lý bộ nhớ(GarbageCollector), kiểm tra tính chính xác của mã máy(bộ xác thực), và quan trọng nhất là thực thi mã máy ActionScript tới mã máy thật thông qua bộ thông dịch và JIT. Với đầy đủ những đặc điểm này, VM đã thể hiện mình là một máy ảo ActionScript khá hoàn chỉnh cho việc chơi Flash.

Trở lại với ví dụ trong hình 2.2 , với mã bytecode được nạp vào từ tệp tin *.abc*, VM đọc lần lượt từng mã vào trong bộ nhớ như *Cpool* để nạp vào *NamingPool* và đặt

những namespaces vào trong VM, sau đó tới lượt các phương thức được nạp vào như: *getlex, callpropvoid, getproperty, pushscope*..., và mã cuối cùng của đoạn tệp tin abc là tag *showframe*, máy ảo sẽ đưa những đối tượng sau khi được xử lý sang danh sách hiển thị.

2.5.3 Đặc điểm của Gnash

Gnash đơn thuần là chương trình chạy tệp tin Flash, Gnash hỗ trợ việc xử lý tệp tin trên máy và qua giao thức mạng. Tuy nhiên, phiên bản hiện tại của Gnash chưa hỗ trợ tốt cho việc xử lý những tệp tin Flash được tạo ra bởi AS 3 do một số lớp, thuộc tính được định nghĩa trong AS 1/2 đã được xóa bỏ hoặc bị thay đổi, và một số lớp thuộc tính mới được cung cấp thêm.

Gnash chưa hỗ trợ hiển thị tệp tin Flash có những đối tượng đồ họa 3D với hiệu ứng phức tạp và những tệp tin định dạng SWF ở phiên bản về sau.

Trong phiên bản AS3, Adobe Systems đã giới thiệu VM mới có tên là AVM 2[7] để thực thi những mã máy được sinh ra từ AS3. Đi kèm theo đó, đặc tả SWF[6] cũng thay đổi để phù hợp với VM mới. Mỗi tệp tin SWF chỉ chứa một kiểu duy nhất: khối ABC(ABC blocks) cho AS 3 hoặc khối DoAction(DoAction blocks) tồn tại trong AS 1/2.

Mục đích chính của khóa luận này là cung cấp cơ sở lý thuyết hướng tới việc thể hiện một cách trực quan đồ họa 3D trên hệ thống nhúng, trên cơ sở lý thuyết đó để phát triển hệ thống riêng(gồm máy ảo và hệ thống biên dịch) dành cho hệ thống nhúng. Cung cấp một số giải pháp toàn diện để thực hiện một cách dễ dàng 3D trên hệ thống nhúng với hiệu năng cao.

CHƯƠNG 3

Bài toán

Trên cơ sở lý thuyết trên, chúng tôi tiến hành xác định cụ thể bài toán, phương pháp thực thi và cách thức tiến hành để hướng tới mục tiêu của dự án. Việc nghiên cứu thể hiện đồ họa 3D trên thiết bị nhúng có giá trị thực tiễn và ứng dụng rất cao trong thời điểm hiện tại. Với sự xuất hiện của những thiết bị hiển thị 3D đầu tiên(tivi 3D, điện thoại 3D...) hứa hẹn đây là một công nghệ tiềm năng, có khả năng phát triển và phổ biến rộng rãi trong tương lai.

Mục đích chính của khóa luận này nhằm cung cấp cơ sở lý thuyết để xây dựng một hệ thống riêng, độc lập với Adobe Flash. Qua quá trình thực hiện, chúng tôi đưa ra một số bài toán cần thiết để xây dựng được hệ thống này.

3.1 Cơ sở

Hiện tại chúng tôi- nhóm phát triển của phòng thí nghiệm Toshiba-Coltech đang thực hiện giai đoạn đầu của dự án, tập trung chủ yếu vào tìm hiểu cấu trúc, cách tổ chức, chỉnh sửa *Gnash* và đưa ra một vài demo cũng như đưa thêm thư viện đồ họa 3D vào *Gnash*. Để có thể hiển thị hình ảnh, chạy video và chơi tệp tin Flash trên một chương trình độc lập với môi trường liên quan đến các vấn đề như:

1. Đồ họa máy tính: Cách xử lý đồ họa 2D, 3D, sự khác biệt giữa các môi trường đồ họa, phương thức thể hiện khác nhau của môi trường thực thi(*OpenGL|ES, Gtk, GtkGlExt*[1]...)
2. Máy ảo: các mã máy, các chỉ lệnh được cung cấp, kiến trúc máy ảo, cách thực thi mã máy, cách biên dịch, thông dịch
3. Ngôn ngữ kịch bản hành động(ActionScript), công nghệ Flash

[1]Thư viện lập trình đồ họa máy tính

4. Tệp tin Flash(SWF): Cách tổ chức tệp tin, nội dung tệp tin, quá trình đọc dữ liệu nhị phân sang mã máy ảo...
5. Tamarin: một dự án hợp tác giữa Adobe và Mozilla trong việc phát triển máy ảo Flash cho cộng đồng phát triển. Mozilla dựa vào đó để phát triển dự án TraceMonkey-Tích hợp Flash vào trình duyệt Mozilla. Adobe tiếp tục phát triển những tính năng mới cho máy ảo.

3.2 Giải pháp

3.2.1 Khái quát

Vì *Flash* là một công nghệ đóng nên việc tìm hiểu, sửa chữa và tùy biến cho phù hợp với nhu cầu, mục đích sử dụng là rất khó khăn, cũng chính vì lý do đó mà *Apple*-Một hãng công nghệ lớn khác không hỗ trợ Flash cho những sản phẩm của mình. Nhiệm vụ đặt ra khi phát triển, tùy biến sản phẩm dựa trên Flash đòi hỏi phải nắm bắt chi tiết công nghệ, kỹ thuật, đặc tả cũng như cách xử lý của Flash. Phiên bản hiện tại của Flash đã hỗ trợ một số đối tượng 3D cơ bản. Để phục vụ mục tiêu của dự án, mục đích chính của khóa luận này nhằm giới thiệu quá trình tìm hiểu, xây dựng lại hệ thống có sẵn, qua đó nắm bắt được nền tảng công nghệ để thực hiện.

Dựa trên cơ sở này, Gnash là một lựa chọn tốt so với việc sử dụng phần mềm đóng hoàn toàn được cung cấp bởi Adobe. Đây là một dự án nguồn mở, tuy nhiên vẫn tồn tại một số hạn chế, tồn tại lớn nhất của Gnash đó là chưa hỗ trợ đầy đủ cho những phiên bản sau của SWF (Gnash 0.8.7). Gnash hỗ trợ tốt nhất cho SWF phiên bản 7, tương đối đầy đủ với SWF phiên bản 8, và chưa hỗ trợ đầy đủ với SWF 9 và đặc biệt chưa hỗ trợ phiên bản 10.

3.2.2 Nội dung

- Hỗ trợ Gnash thực thi đồ họa 3D, sử dụng thư viện đồ họa PaperVision3D, bằng cách cải tiến hoặc thêm mới lớp as_object để phù hợp với các lớp có trong AS 3.

  Gnash với những đối tượng as_object được định nghĩa cho AS 2 trong khi đó, PP3D là thư viện AS được viết dựa vào AS 3. Đối tượng trong AS 3 có thuộc tính dựa trên đặc điểm và được thiết kế theo tính kế thừa, cùng khuôn dạng. Trong khi đó, AS 2 thiết kế đối tượng dựa vào những phương thức để thiết lập trạng thái cho đối tượng.

- Cải tiến máy ảo có trong Gnash.

Mặc dù Gnash hiện tại có hai máy ảo chạy độc lập nhau là VM và Machine[2]. Đối với VM đã được thiết kế tương đối đầy đủ và hỗ trợ đầy đủ cho AS 1/2 nhưng với Machine thì không phải vậy. Machine chưa thực thi một cách đúng đắn đối với mã máy có nguồn gốc AS 3. Hiện tại Machine vẫn coi những mã máy AS 3 thực thi như đối với mã máy AS 2. Do đó chúng tôi cần phải thay đổi Machine để thực thi một cách đúng đắn AS 3 bằng cách cải tiến Machine(tốn rất nhiều công sức và thời gian) hoặc bằng cách chuyển quyền điều khiển mã máy AS 3 sang cho Tamarin.

- Cung cấp lớp *DisplayObject* cho *Tamarin*.

  Tamarin được cung cấp chỉ bao gồm máy ảo AS 2(AVM2) và NanoJIT. Tamarin chỉ cung cấp những lớp cơ bản có trong AS 3 mà không đưa ra một lớp rất quan trọng trong việc hiển thị hình ảnh của Flash đó là *DisplayObject*. Bằng cách viết lại DisplayObject dựa trên thuộc tính của đối tượng và đặc điểm của chương trình quản lý màn hình(X11, Gtk...) hoặc bằng cách cải tiến, cài đặt DisplayObject được cung cấp sẵn trong Gnash để tương thích với đối tượng hiển thị trong AS 3.

- Cải tiến phần *renderer* của Gnash hỗ trợ thêm lựa chọn OpenGLES.

  Hiện tại Gnash mới chỉ hỗ trợ việc dùng các thư viện đồ họa: AGG, Cairo và mặc định là GTK.

Dựa trên các hướng giải quyết trên, chúng tôi rút ra phương án tốt nhất là sử dụng Tamarin để hỗ trợ Gnash, tức là đưa ra giải pháp để Tamarin hoạt động cùng Gnash. Vì Gnash đã cung cấp sẵn những đối tượng thực thi phần hiển thị như *DisplayObject, DisplayList*... và cơ bản Gnash cũng đã hỗ trợ OpenGL và chúng tôi sẽ không phải xử lý nhiều tới máy ảo AVM2 vì quá phức tạp nếu chỉnh sửa *Machine*. Trong những phần tiếp theo của khóa luận sẽ nói tới giải pháp này.

3.3 Kỹ thuật hiển thị Flash Video

3.3.1 Cấu trúc dữ liệu lưu trữ đối tượng hiển thị

Display List là một thành phần trong Gnash để tổ chức tất cả các phần tử đồ họa(gồm MovieClips, text, Jpgs...) và hiển thị chúng trên màn hình. Nói cách khác, *Display List* là một cấu trúc dữ liệu dùng để tổ chức các phần tử đồ họa và sau đó hiển thị lên trên màn hình. Nguyên tắc của cấu trúc dữ liệu này là tuân theo mô hình mỗi MovieClips sẽ có một độ sâu nhất định với MovieClips khác.

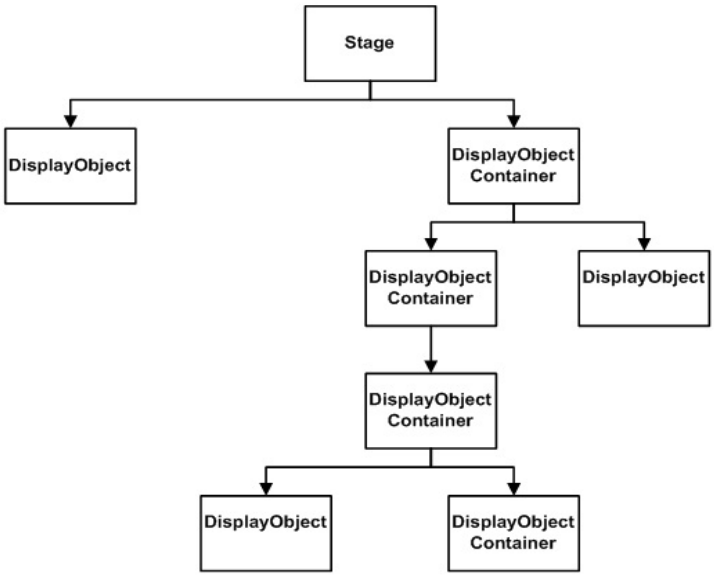
[2]Hai máy ảo được định nghĩa trong Gnash

Kiến trúc của bất cứ chương trình thực thi tệp tin Flash nào cũng phải chứa DisplayList, đây là một cấu trúc dạng cây với mỗi giai đoạn(Stage) nó cần thực thi là gốc và nhánh là DisplayObjectContainers, cuối cùng DisplayObjects là nút lá(hình vẽ 3.1).

Stage là một thành phần cơ sở để chứa những thành phần *DisplayObjects* và với mỗi trạng thái(hoặc frame- khung) tồn tại duy nhất một *Stage*. Trong Gnash, Stage tương ứng với lớp movie_root, một thành phần tồn tại và được định nghĩa trong tệp tin Flash nhị phân(*SWF*) và được chương trình thực thi phân tích từ tệp tin này. *Stage* chứa tất cả các thành phần mà bạn có thể nhìn thấy từ tệp tin Flash của bạn, nó chứa mọi thứ hoặc nó chứa một thành phần được tham chiếu bởi một thành phần khác trong Flash. Stage là một thành phần của DisplayList và được thể hiện lên màn hình ở mỗi Frame.

DisplayObjectContainer là một bộ chứa *DisplayObjecs*. Trong Gnash, *MovieClip* là lớp có chức năng tương tự- Chứa đựng chỉ những phần tử đồ họa và những hành vi giống như quản lý về độ sâu, gắn với những đối tượng khác.

Tất cả những thành phẩn ẩn(hoặc chưa được định nghĩa), từ những trường văn bản tới những bức ảnh hoặc hình vẽ, và những *stage*, đều được mở rộng từ lớp DisplayObject. Tất cả các dạng của DisplayObjects có cùng thuộc tính như *x, y, visible*... và mỗi lớp con lại chứa những đặc tính riêng cho đối tượng của nó. DisplayObject là thành phần đại diện cho tất cả những đối tượng đồ họa được sử dụng trong Flash.



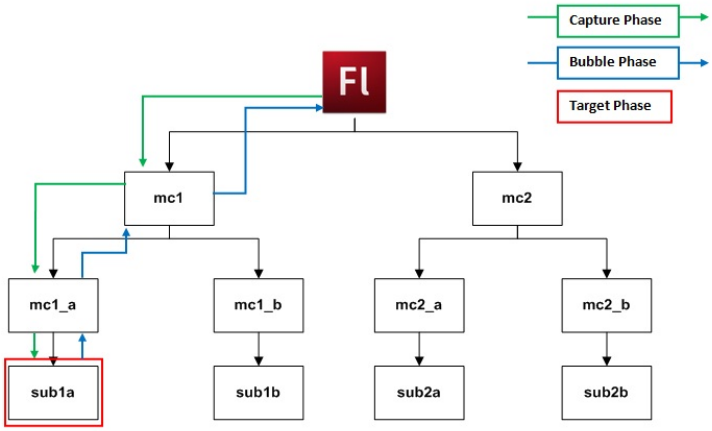
Hình 3.1: Cấu trúc của DisplayList

Sử dụng danh sách hiển thị dạng cây, con trỏ của đối tượng sẽ đi qua mọi phần tử trong *DisplayObjectContainer* với mỗi trạng thái tương ứng và vẽ nó.

3.3.2 FlashVideo với các sự kiện

Flash hỗ trợ nhiều kiểu sự kiện từ các thiết bị như chuột, bàn phím. Với việc sử dụng những sự kiện, Flash làm cho tệp tin có tính tương tác cao đúng như tên gọi của ngôn ngữ là ActionScript(kịch bản hành động). Đối với mỗi tệp tin, *movie* sau khi được đưa vào trong *DisplayList*, những sự kiện sẽ được thực hiện khi trải qua 3 quá trình(được thể hiện ở hình vẽ 3.2):

1. The capture phase(Giai đoạn bắt sự kiện): sự kiện được nghe bắt đầu đi xuống từ *stage* tới các mục tiêu hoặc nguồn gốc của sự kiện.
2. The target phase(Giai đoạn tìm mục tiêu): Sự kiện được nghe từ chính bản thân nó.
3. The building phase(Giai đoạn thực thi): sự kiên quay trở lại nghe trên danh sách hiển thị, đi ngược từ mục tiêu tới trạng thái đầu.



Hình 3.2: FlashVideo với các sự kiện

Dưới đây là ví dụ về cách thực thi các sự kiện trong Gnash.

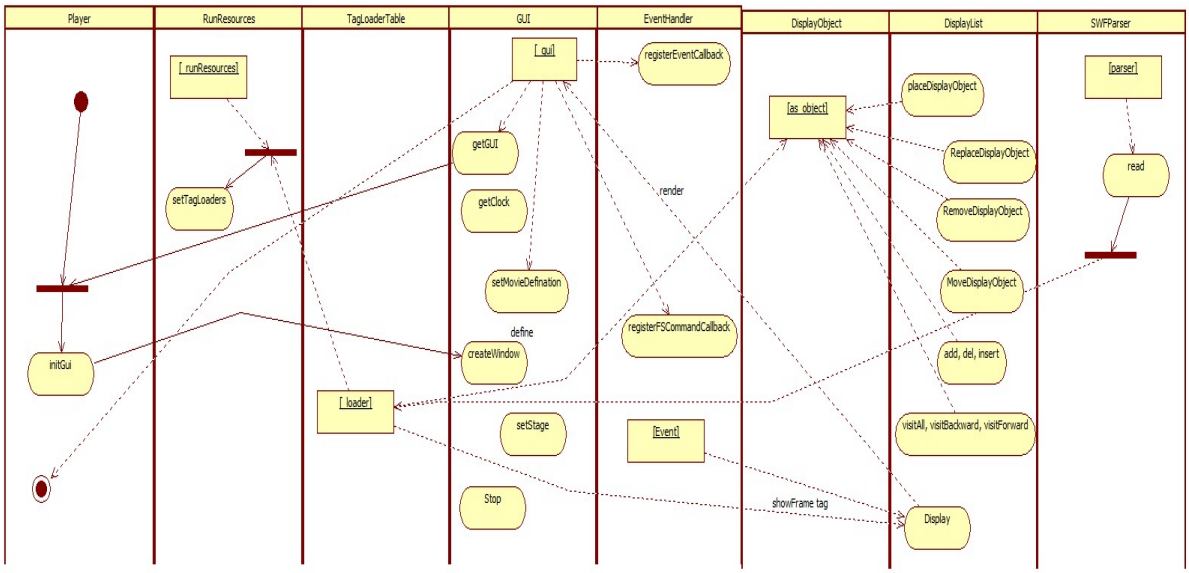
Trở lại ví dụ ở hình vẽ 3.4, với sự kiện là chuột(*MouseEvent*) được người sử dụng chọn vào *sub1a*, chúng ta dễ dàng nhận thấy ở giai đoạn bắt sự kiện, những đối tượng lần lượt được lựa chọn từ stage $\Rightarrow$ mc1a $\Rightarrow$ mc1_a $\Rightarrow$ sub1a. Sự kiện này được tạo và đi dần từ *stage* xuống tới đối tượng sử dụng của sự kiện.

Sự kiện này, tiếp tục được xử lý bởi đối tượng được chọn, đây là bước tìm mục tiêu. Sau cùng, sự kiện này sẽ được lần ngược lên từ đối tượng được chọn tới stage để các đối tượng phía trên xử lý.

Chú ý: Luồng sự kiện của *displaylist* không thể chỉ cho chúng ta thấy phần nào của danh sách này sẽ xử lý sự kiện mà chỉ chỉ cho ta biết phần nào không thể xử lý sự kiện này.

3.4 Áp dụng

Căn cứ vào mã nguồn được cung cấp, dựa vào các chức năng và luồng sự kiện trong mã nguồn, chúng tôi đã dựng lại cơ chế thực hiện của *GNASH* như hình vẽ 3.3.



Hình 3.3: Luồng xử lý đối tượng đồ họa trong Gnash

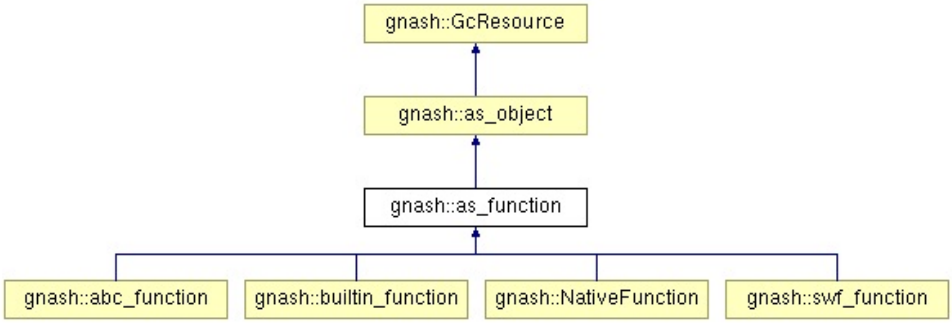
Đầu vào của Gnash là tập tin có phần mở rộng .swf, sau khi qua quá trình phân tích từ *SWFParser*, các tag được lưu trữ vào cấu trúc dữ liệu dạng bảng *tagTableLoader*. Từ đây, với từng sự kiện được bắt trên GUI hoặc những hành động của tập tin Flash được định nghĩa, tác động tới danh sách hiển thị(*DisplayList*) và sau đó, nạp các dữ liệu từ danh sách các tag, đối tượng được lưu trữ trong TagLoaderTable tới GUI.

Nhìn vào hình vẽ trên ta nhận ra ngay rằng, đối tượng chủ yếu cần được quan tâm là *as_object*, một thành phần quan trọng để định nghĩa các đối tượng và sau đó được thực thi bởi máy ảo. Sở dĩ hiện tại Gnash không thể thực hiện được những tệp tin Flash ở version cao là do ở AS 3 có thêm một số thuộc tính mới cũng như đã thay đổi cách thể hiện thuộc tính cho từng đối tượng của Flash. Vì lý do này, AS 3 đã đưa vào một máy ảo mới, đó là AVM2 nhưng hiện tại Gnash chưa hỗ trợ đầy đủ đối với AVM2 và chúng ta phải mất rất nhiều công sức để hoàn chỉnh vấn đề này.

Giải pháp được đưa ra ở đây đó là: cải tiến lớp as_object để có thể thể hiện đầy đủ những thuộc tính cũng như những phương thức mới có trong AS 3 và chỉnh sửa hoàn chỉnh AVM 2 hiện có của Gnash để có thể tạo ra những mã máy phù hợp với nội dung mong muốn được hiển thị bởi GUI.

Vì as_object liên quan tới nhiều lớp khác như: *GcResources*, *as_function*- Một lớp định nghĩa những hàm được cung cấp cho đối tượng as_object, được định nghĩa như

những hàm của đối tượng có trong AS 2. Do đó nhiệm vụ đặt ra là chỉnh sửa *as_function* để có thể thực thi được những thuộc tính của đối tượng như trong AS 3. Chi tiết những lớp liên quan được thể hiện như hình vẽ 3.4.



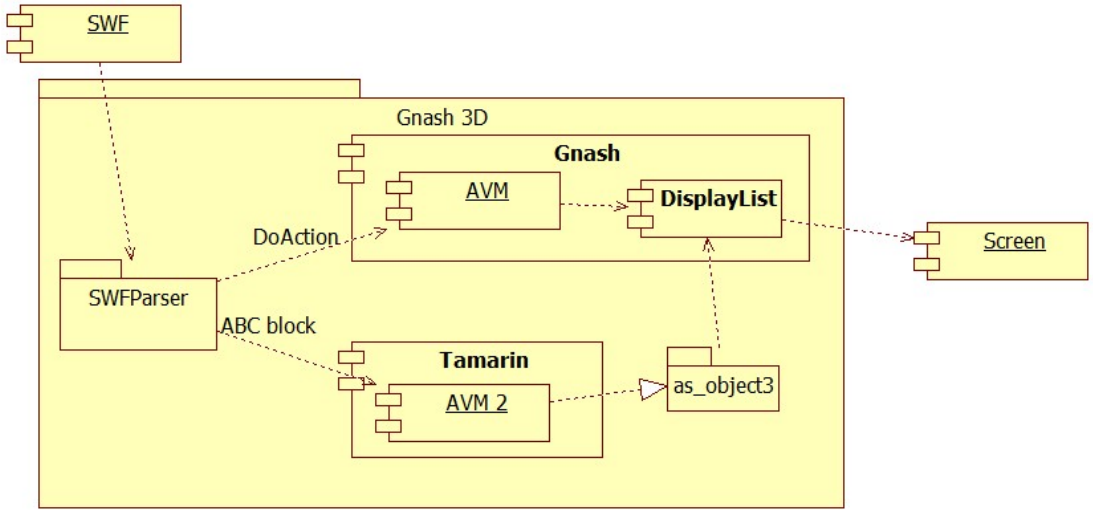
Hình 3.4: Sơ đồ kế thừa as_object

Hiện tại Gnash cung cấp hai máy ảo chạy độc lập: máy ảo *vm* để chạy cho những tệp tin Flash của phiên bản cũ(từ phiên bản 8 trở lại) đã khá hoàn chỉnh và đầy đủ, máy ảo *Machine* chưa được hoàn chỉnh để thực thi những nội dung của tập tin Flash từ phiên bản 9. Để phù hợp với mục tiêu và nhiệm vụ trong giai đoạn này của dự án, chúng tôi đã sử dụng Tamarin- Máy ảo cho phép xử lý mã máy từ phiên bản 9 trở lên của Flash. Tamarin được biết đến như là AVM 2 được sử dụng để thay thế cho máy ảo Machine có trong Gnash. Trong quá trình thực hiện, chúng tôi đã sử dụng biện pháp chuyển quyền điều khiển của Gnash cho máy ảo Tamarin thực thi nếu gặp những tập tin Flash có phần header chứa nội dung của phiên bản 9 trở lên.

Trong Gnash, máy ảo không được khởi tạo ngay khi chương trình chạy mà chỉ khi xác định được tag sinh ra từ phiên bản nào mới bắt đầu khởi tạo máy ảo tương ứng để thực thi tag đó. Mỗi kiểu tệp tin Flash có một cách định nghĩa hành động khác nhau: DoAction đối với AS2 và ABC blocks đối với AS 3. Một tệp tin SWF chỉ chứa duy nhất một kiểu hành động: hoặc DoAction hoặc ABC blocks.

AVM 2 chỉ thực thi mã theo ABC blocks và không thực thi mã theo DoAction. Đối với Gnash, khi máy ảo VM đọc vào tag ABC blocks thì hành động sẽ được chuyển sang xử lý bởi Machine, còn khi gặp tag DoAction, máy ảo VM sẽ tạo ra VM mới, hủy VM cũ để thực hiện tác vụ mới.

Dựa trên cơ sở này, tôi đề xuất việc sử dụng Tamarin theo cơ chế sau: khi ABC blocks tồn tại trong tệp tin Flash, lớp SWFParser sẽ phân tích thành các khối ABC để nạp vào máy ảo Tamarin. Đối với khối DoAction vẫn giữ nguyên kiến trúc cũ. Hình vẽ 3.5 thể hiện kiến trúc của mô hình này.



Hình 3.5: Mô hình Gnash thực thi đồ họa 3D

3.4.1 Thực thi đồ họa 3D trên thiết bị nhúng

Để hỗ trợ OpenGL ES, trong giai đoạn từ đưa những đối tượng as_object sang danh sách hiển thị để xuất hiện khi renderer lên màn hình, cần thêm một gói renderer để hỗ trợ chuyển đổi từ đối tượng hiện tại sang OpenGLES.

Vì Gnash hiện tại đã thực thi OpenGL, bài toán đặt ra là chuyển từ OpenGL⇒ OpenGLES. Như chúng ta đã biết, OpenGLES là một thư viện thu gọn của OpenGL chuẩn để có thể thi hành trên những thiết bị di động như điện thoại di động hoặc các thiết bị cầm tay khác.

Hiện tại đa số các ứng dụng được viết với chuẩn OpenGL và Flash là một trong số đó. Khi một ứng dụng cũng như tệp tin được sinh ra từ OpenGL, hiển nhiên một số chức năng cũng như hiệu ứng sẽ không được thực thi với OpenGLES vì OpenGLES đã giản lược đi một số hàm, phương thức có trong OpenGL để có thể thực thi được trên thiết bị có tài nguyên, năng lực hạn chế. Chỉ 10% API của OpenGL tồn tại trong OpenGLES, 50% API của OpenGL phải thay đổi, sửa chữa với ít tham số hoặc để tốn ít tài nguyên hơn, nhưng phần lớn đặc điểm chính của OpenGL là không được hỗ trợ trên OpenGLES[14]. Ngoài ra đa số các ứng dụng đồ họa 3D được phát triển trên PC. Vấn đề đặt ra là yêu cầu chuyển đổi từ OpenGL sang OpenGLES để thiết bị có thể sử dụng một cách rộng rãi những tài nguyên được cung cấp và dễ dàng cho cộng đồng người phát triển. Dogless[14] là một phát minh để dùng cho mục đích này.

Với môi trường giả lập trên PC mô phỏng PowerVR Insider[3] được cung cấp cho người phát triển phần cứng và phần mềm trên thiết bị nhúng. PowerVR đưa ra một giải

[3]Một chip đồ họa nhúng cho phép thực thi trên nền tảng OpenGLES 2.0

pháp mạnh mẽ, linh hoạt cho việc thực hiện ứng dụng 2D/3D/vector với GPU[4] bao gồm cả xử lý và tạo ảnh 2D/3D, mã hóa- giải mã video. Tất cả các API chính được hỗ trợ bao gồm: OpenGL ES 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 Vector cho thiết bị nhúng, với các kỹ thuật chống răng cưa, và đưa ra các chức năng xử lý hình ảnh chuyên sâu.

Trên thực tế, PowerVR đã trở thành chip xử lý đồ họa phổ biến nhất sử dụng trong điện thoại di động để thể hiện hình ảnh 2D, 3D, tăng tốc đồ họa Vector(với dòng cao cấp) và được sử dụng bởi các công ty công nghệ bán dẫn di động hàng đầu với hàng trăm sản phẩm như Samsung...

3.4.2 Hiển thị 3D trên Gnash dựa trên PaperVision 3D

Hiện tại, thư viện đồ họa 3D dùng cho Flash được dùng kết hợp với sự hỗ trợ của bộ Framework Flex SDK do Adobe cung cấp. Do mục tiêu của dự án cần chuyển đồ họa Flash 3D lên thiết bị di động do đó cần thiết phải chỉnh sửa bộ thư viện này. Mục đích chính để thư viện có thể chạy trên OpenGLES với thiết bị nhúng. Chi tiết về kỹ thuật này được trình bày trong khóa luận của bạn Lê Viết Sơn về "Phương pháp xử lý đồ họa 3D của Papervision 3D". Trong tài liệu đó, các bạn có thể tìm thấy cơ chế, giải pháp cho việc đưa PaperVision 3D vào Gnash, một số thực nghiệm thể hiện PP3D.

[4]Graphics Processing Unit

CHƯƠNG 4

Thực nghiệm

Với những cơ sở lý thuyết và những giải pháp được đề xuất, chúng tôi tiến hành một số phương pháp xác định khả năng thực thi của đồ họa 3D đối với máy ảo. Dựa vào một số máy ảo Flash nguồn mở, chúng tôi thực hiện một số đánh giá dựa vào quá trình so sánh kết quả thực hiện chủ yếu dựa vào thời gian thực hiện test.

4.1 Các so sánh, đánh giá

Mục đích chính của việc so sánh là chứng minh khả năng thực thi của các máy ảo ActionScript với một số bộ test. Các số liệu test được sử dụng từ công cụ test Java Grande Benchmarks[15], được thực hiện trên một sô máy ảo flash nguồn mở như Lightspark, Tamarin Central, Tamarin Tracing, Tamarin Redux. Ngoài ra, chúng tôi có sử dụng một số kết quả đã được phân tích để phục vụ cho mục đích so sánh và đánh giá các khả năng cùng hiệu suất giữa chúng, nhằm lựa chọn ra bộ mã nguồn thích hợp cho mục đích dự án.

4.1.1 LightSpark

Lightspark[16]- Một chương trình mã nguồn mở thực thi Flash(Flash Player) đơn giản hơn rất nhiều so với Gnash vì chỉ chứa những thành phần cơ bản của Flash Player: thực thi thời gian chạy với chế độ đồ họa nhưng hỗ trợ rất tốt cho ActionScript 3 như đặc tả lớp, các phương thức, cách thừa kế, mô tả đối tuọng...

Lightspark sử dụng máy ảo LLVM[17][1] thay vì sử dụng một máy ảo riêng. LLVM cung cấp một chương trình khung với các API để thực thi trên máy ảo này. Lightspark phụ thuộc rất nhiều vào LLVM, từ tốc độ biên dịch tới tối ưu mã. Lightspark sử dụng

[1] Low-Level Virtual Machine

OpenGL để hiển thị hình ảnh, và thực thi đa luồng.

Hiện tại Lightspark vẫn đang được phát triển để tối ưu hơn tốc độ và khả năng thực thi phim Flash.

4.1.2 Tamarin

Hiện tại, Tamarin có 2 phiên bản là tamarin-Tracing, phiên bản thử nghiệm của máy ảo ActionScript và phiên bản thứ 2 đã và đang được phát triển bởi cộng đồng nguồn mở: Tamarin-redux. Tamarin-tracing được sửa đổi để hướng tới mục đích tối ưu hóa các quá trình trong lúc thực thi và những mã chưa định nghĩa với việc sử dụng một bộ nhớ nhỏ. Tamarin-redux là phiên bản để giới thiệu những tính năng mới và sửa những lỗi phát hiện thêm trong quá trình thực thi phát sinh bởi Tamarin-Central. Adobe cũng cung cấp kèm theo công cụ cho việc kiểm tra tính năng, hiệu suất của Tamarin trong thực thi Flash(bộ *test-suite*). Bộ máy ảo được phát triển bởi một đội chuyên nghiệp trong thời gian kéo dài hằng năm nay.

4.1.3 Kiểm nghiệm

Chương trình kiểm nghiệm được thực thi dựa trên việc so sánh hiệu năng thực thi ba chương trình trên. Kết quả test thực tế được sử dụng và phân phối từ DHPC Java Grande Benchmark[15]. Một phiên bản của bộ test này được đưa vào mã nguồn Tamarin và có thể được biên dịch tới mã máy AS sử dụng công cụ biên dịch ASC của Adobe. Sau test này được thực thi Lightspark.

Bộ công cụ này[18] gồm những kiểm tra sau:

Crypt mã hóa và giải mã dữ liệu ngẫu nhiên sử dụng thuật toán IDEA symmetric, cung cấp một lượng lớn những kiểm tra về số nguyên và toán tử logic.

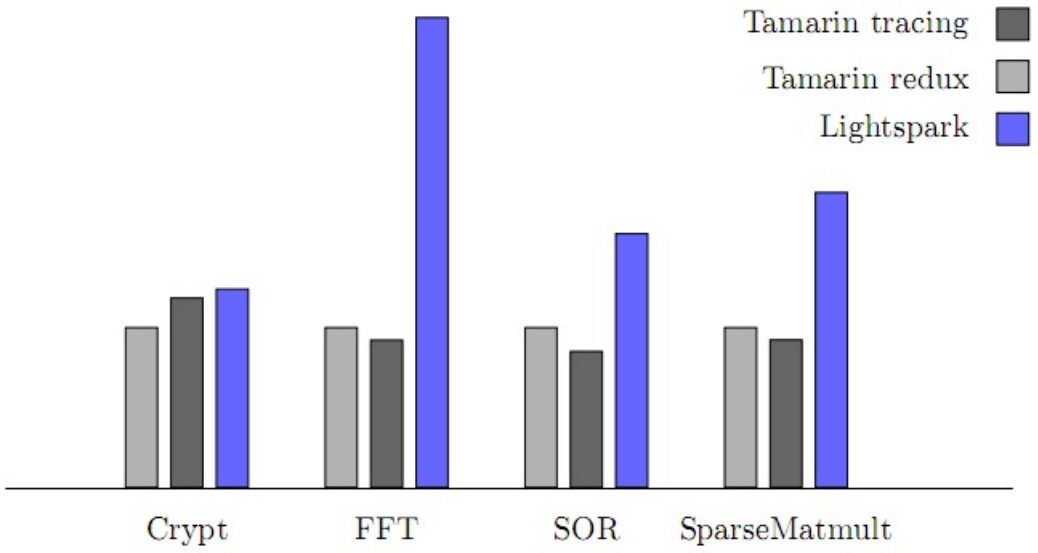
FFT tính toán chuỗi Fast Fourier Transform trên dữ liệu ngẫu nhiên, bao gồm cả dữ liệu dấu phẩy động.

Series Tính toán N hệ số đầu tiên của chuỗi Fourier, để chứng minh khả năng gọi hàm trong lớp Toán học.

SOR Tính toán giải pháp của một hệ thống tương tác trực tuyến sử dụng *Successive over-relaxation*.

SparseMatMult Thực thi nhân ma trận trên một ma trận ngẫu nhiên, chứng minh khả năng truy xuất bộ nhớ.

Hình vẽ 4.1[19] cho chúng ta thấy rằng Lightspark thực thi tốn thời gian hơn rất nhiều so với Tamarin và đối với kiểm tra về Crypt, chúng ta thấy kết quả khá tương đồng



Hình 4.1: Biểu đồ so sánh kết quả thực thi

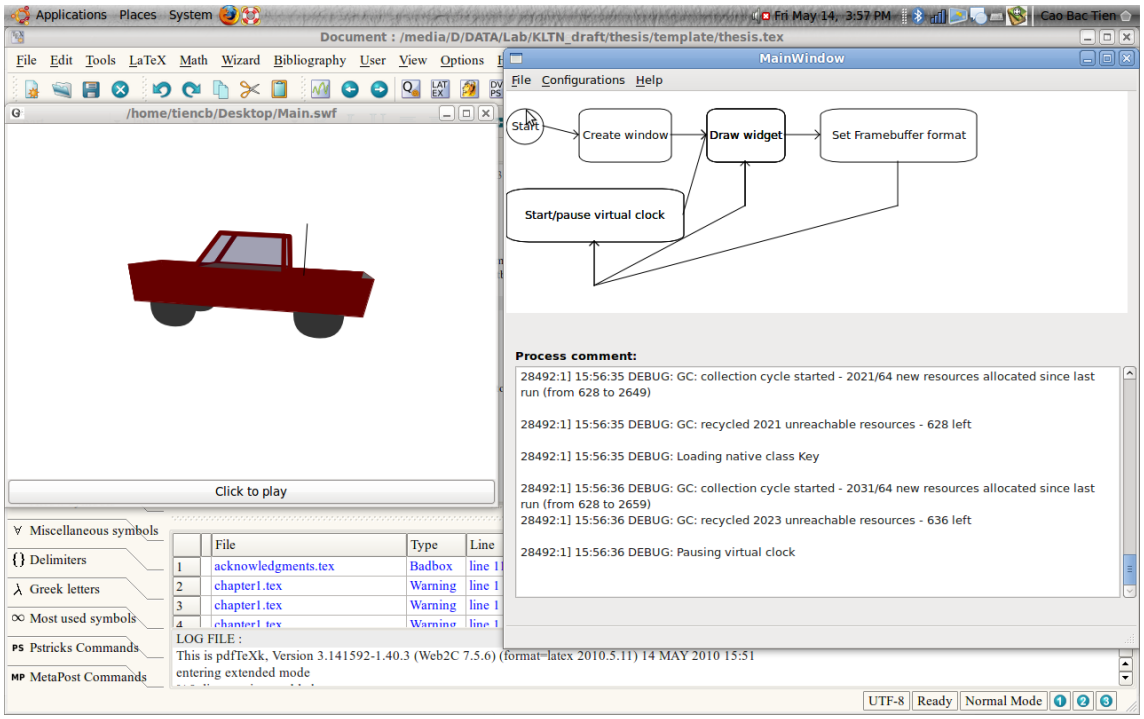
giữa các chương trình này.

4.2 Demo

Dưới đây là chương trình được chúng tôi thực hiện để thể hiện cơ chế xử lý FlashVideo của Gnash đối với tệp tin SWF. Chương trình thể hiện cơ chế để một tệp tin SWF- Đầu vào của Gnash, thực thi và thể hiện lên màn hình. Giao diện chính của chương trình như hình vẽ 4.2. Chương trình gồm các phần chính là:

- **Process Comment:** Thể hiện luồng đầu vào, quá trình phân tích, thực thi và chạy của Gnash đối với tệp tin SWF, các đối tượng được phân tích, được gọi và hiển thị.
- **Display Component:** Thể hiện trực quan hình vẽ luồng thực thi đối với từng tệp tin đầu vào cụ thể. Thể hiện sự ràng buộc giữa các đối tượng khác nhau được tạo ra trong quá trình thực thi của Gnash.
- **DisplayMovie:** Thể hiện nội dung của tệp tin SWF đầu vào, cho phép người dùng tương tác với tệp tin Flash.

Vùng thể hiện tiến trình(*Process Comment*) cho chúng ta thấy rằng(đối với tập tin hiện tại), sử dụng những tài nguyên nào, Gnash đã gọi và khởi tạo các đối tượng ra sao, quá trình thực thi đối với tệp tin đó lần lượt ra sao. Quá trình này còn thể hiện cách thực thi của máy ảo được cài đặt trong Gnash đã sử dụng, phân phối và thu hồi bộ nhớ các đối tượng đã cấp phát ra sao.



Hình 4.2: Luồng xử lý Video của Flash- Gnash Player

Vùng hiển thị(*Display Comment*)- Phần phía trên của *Process Comment*, cho chúng ta một hình dung cụ thể về cách xử lý các đối tượng, cách xử lý đối với một số sự kiện(events) và mối quan hệ giữa các quá trình trong Gnash. Với mô hình này, các quá trình của Gnash được thể hiện rõ ràng, dễ hiểu nhằm phục vụ cho mục đích thể hiện kỹ thuật Flash Video trong Gnash trong khóa luận này.

Phần cuối cùng, phần hiển thị nội dung của tệp tin đầu vào *.swf*. Tại phần cửa sổ này, toàn bộ nội dung tệp tin Flash sẽ được thực thi, hiển thị giống như với một số chương trình chơi Flash khác(Flash Player). Với phần thể hiện này của tệp tin Flash giúp chúng ta dễ dàng kiểm tra sự xuất hiện của đối tượng này so với những nội dung được thể hiện ở 2 phần còn lại là *Process Comment* và *Display Comment*.

CHƯƠNG 5

Kết luận

Tổng kết

Trong quá trình thực hiện đề tài, tôi đã đạt được một số kết quả như sau: *đưa ra cơ chế thể hiện FlashVideo của Gnash, đưa ra mô hình đề xuất cho việc thể hiện Flash 3D, cung cấp một số hướng thực hiện của đề tài, cơ chế để đưa flash 3D lên thiết bị nhúng.* Qua quá trình thực hiện dựa vào một số kết quả đã chứng tỏ tính đúng đắn trong giải pháp cũng như khả năng thực hiện. Đây chính là tiền đề quan trọng để chúng tôi tiếp tục thực hiện việc xây dựng hệ thống riêng độc lập với Adobe Flash để thực thi trên thiết bị nhúng.

Mặc khác, chúng tôi cũng hi vọng với những giải pháp, kết quả được nêu ra ở bài viết này, sẽ tạo điều kiện thuận lợi cho các nhóm, các hướng nghiên cứu khác về xử lý Flash 3D, đặc biệt là việc chuyển Flash 3D vào thiết bị, một hướng nghiên cứu mới và rất tiềm năng trong hiện tại và tương lai gần.

Hướng phát triển

Qua khóa luận này, chúng ta có thể nhận thấy khả năng phát triển tiếp theo của đề tài đối với việc thực thi Flash 3D trên PC nói chung và trên thiết bị nói riêng là rất tiềm năng và còn cần phải được quan tâm, nghiên cứu sâu sắc, triệt để hơn nữa. Việc cá nhân hóa thiết bị điện tử đang là xu thế trên thế giới và dĩ nhiên đi kèm với đó là đồ họa máy tính cho những thiết bị này ngày càng phải tốt hơn, trực quan hơn...

Những cơ sở lý thuyết, giải pháp, thực nghiệm trên đây là tiền đề cho chúng tôi tiếp tục công việc xây dựng Flash 3D để thể hiện một số cấu trúc dữ liệu phức tạp, phân tích và tăng tính trực quan cho người sử dụng trong những bài toán sử dụng số liệu tính toán lớn, phức tạp.

Tài liệu tham khảo

[1] PaperVision 3D. Forum papervision 3d, June 21 1994. URL http://http://blog.papervision3d.org/. (last accessed February 7, 2004).

[2] Flare, data visualization for the web. URL http://flare.prefuse.org/.

[3] *Action Script*. Adobe Systems.

[4] *ECMAScript Language Specification*. Ecma International, December 2009. URL http://www.ecmascript.org/.

[5] *Adobe Flash*. Adobe and Macromedia.

[6] *SWF File Format Specification Version 10*. Adobe Systems Incorporated, November 2008.

[7] *Adobe Flash Player ActionScript Virtual Machine*. Adobe Systems, December 6, 2006.

[8] *ActionScript Virtual Machine 2 overview*. Adobe Systems, May 2007.

[9] Tracemonkey. URL https://wiki.mozilla.org/JavaScript:TraceMonkey.

[10] Flex sdk framework. URL http://opensource.adobe.com/wiki/display/flexsdk/Flex+SDK.

[11] Jeff Winder and Paul Tondeur. *Papervision3d essentials*. Packt Publishing Ltd., 32 Lincoln Road Olton Birmingham, B27 6PA, UK, 1999.

[12] Gnu project. URL http://www.gnu.org/gnu/thegnuproject.html.

[13] *Gnash Documents*. Gnash and GNU. URL http://www.gnu.org/software/gnash/.

[14] *OpenGL to openGL/ES translator and openGL/ES simulator*. EUROPEAN PATENT APPLICATION, 20.04.2007.

[15] Dhpc java benchmarks. URL http://www.dhpc.adelaide.edu.au/projects/javagrande/benchmarks/.

[16] Alessandro Pignotti. Lightspark project homepage. URL http://lightspark.sourceforge.net.

[17] The llvm compiler infrastructure. URL http://llvm.org/.

[18] PD Coddington JA Mathew and KA Hawick. Dhcp grande benchmark. URL http://www.dhpc.adelaide.edu.au/projects/javagrande/benchmarks/dhpc_bench.tar.gz.

[19] Alessandro Pignotti. An efficient actionscript 3.0 just-in-time compiler implementation. Master's thesis, University of Pisa, Pisa, Italy, 09/2008.