LỜI CẢM ƠN

Trên thực tế không có sự thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác. Trong suốt thời gian từ khi bắt đầu học tập ở giảng đường đại học đến nay, em đã nhận được rất nhiều sự quan tâm, giúp đỡ của quý Thầy Cô, gia đình và bạn bè. Với lòng biết ơn sâu sắc nhất, em xin gửi đến quý Thầy Cô ở Khoa Công Nghệ Phần Mềm – Trường Đại Học Công Nghệ Thông Tin đã cùng với tri thức và tâm huyết của mình để truyền đạt vốn kiến thức quý báu cho chúng em trong suốt thời gian học tập tại trường. Và đặc biệt, trong học kỳ này, Khoa đã tổ chức cho chúng em được thực hiện "Khóa Luận Tốt Nghiệp". Em xin chân thành cảm ơn ThS. Trần Anh Dũng đã tận tâm hướng dẫn chúng em thực hiện và hoàn thành tốt Khóa Luận. Nếu không có những lời hướng dẫn, dạy bảo của Thầy thì em nghĩ Khóa Luận này khó có thể thành công được. Một lần nữa, em xin chân thành cảm ơn Thầy. Mặc dù đã cố gắng nhưng với thời gian thực hiện Khóa luận có hạn nên không thể tránh khỏi những thiếu sót, nhóm chúng em rất mong nhận được những ý kiến đóng góp quý báu của quý Thầy Cô và các bạn học cùng lớp để kiến thức của em được hoàn thiện hơn.

Sau cùng, em xin kính chúc quý Thầy Cô trong Khoa Khoa Công Nghệ Phần Mềm và Thầy Trần Anh Dũng thật dồi dào sức khỏe, niềm tin để tiếp tục thực hiện sứ mệnh cao đẹp của mình là truyền đạt kiến thức cho thế hệ mai sau.

Trân trọng.

TP. HCM, ngày 09 tháng 01 năm 2014

Sinh viên thực hiện

(ký và ghi họ tên)

LỜI MỞ ĐẦU

Ngày nay, các thiết bị di động hay điện thoại thông minh đang trở nên phổ biến và phát triển vô cùng mạnh mẽ. Theo đó là các ứng dụng và trò chơi trên nền tảng thiết bị di động cũng ngày càng trở nên phong phú và ứng dụng vào lợi ích thực tiễn vô cùng lớn. Hơn thế nữa, việc lập trình các ứng dụng và trò chơi trên thiết bị di động cũng đơn giản, không cần thiết phải có một tổ chức lớn nào mà chỉ cần cá nhân hay nhóm nhỏ cũng có thể làm được những ứng dụng tiện ích hay trò chơi hấp dẫn. Do đó, nó đem lại lợi nhuận rất lớn cho các công ty, tổ chức, các lập trình viên cũng như học sinh sinh viên.

Hiện nay, các nền tảng di động phổ biến như iOS, Android, Windows Phone …, đang là những nền tảng mạnh nhất hiện tại. Trong đó, Windows Phone đã và đang phát triển rất mạnh và có khả năng vượt mặt các đàn anh của nó. Cùng với vô số các thiết bị di động ứng dụng nền tảng Windows Phone ra đời và điểm nhấn đó là sự ra đời của Windows Phone 8 vừa qua đang làm nên cơn sốt trên nền tảng điện thoại thông minh. Các thiết bị ứng dụng nền tảng Windows Phone 8 đã đạt được thành quả vô cùng lớn như Lumia 920 đã trở thành điện thoại được người dùng yêu thích nhất vào năm 2012.

Dựa trên nền tảng là ứng dụng dành cho học tập và nghiên cứu về các hiện tượng vật lý dựa vào các mô phỏng. Từ đó ứng dụng mô phỏng Vật lý trên Windows Phone ra đời nhằm đáp ứng nhu cầu đó. Mục tiêu của đề tài là xây dựng một ứng dụng mô phỏng Vật lý trên thiết bị Windows Phone dành cho các đối tượng nghiên cứu và tìm hiểu về môn Vật Lý cần có những mô phỏng ngoài thực tế cũng như trong điều kiện toán học lí tưởng của môn học vật lý trong nhà trường ở cấp học phổ thông trung học. Ứng dụng mô phỏng cũng đồng thời thay thế cho việc sử dụng các thiết bị thí nghiệm Vật lý thất với giá cả không hề rẻ cho một bộ thiết bị.

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

TÓM TẮT KHÓA LUẬN

Hiện nay với sự phát triển nhanh chóng của Công nghệ thông tin trên Thế giới nói chung và lĩnh vực trên thiết bị di động thông minh nói riêng, cùng phát triển theo đó là sự phát triển nhanh chóng của phần mềm dành cho các thiết bị di động. Cùng với đó là sự phát triển của giáo dục nghiên cứu và ứng dụng Vật lý trong trường học. Điều đó đã thúc đẩy phải đáp ứng được đầy đủ các phương tiện và thiết bị dành cho việc nghiên cứu và học tập. Nhưng các thiết bị dành cho nghên cứu có giá cả khá đắt và không phải ai cũng có đủ khả năng có được các thiết bị đó. Dựa vào các yếu tố trên Nhóm quyết định xây dựng ứng dụng mô phỏng Vật lý trên thiết bị Windows Phone nhằm giải quyết các vấn đề trên.

Tại sao lại là Windows Phone? Windows Phone là một nền tảng mới và non trẻ nhất so với Android và IOS, nhưng lại mang một tiềm năng lớn là nền tảng của tương lai.

Hướng tiếp cận của nhóm chính là công nghệ XNA Windows Phone. Nhằm bao quát hết các đối tượng sử dụng thiết bị Windows Phone 7 & 8 thay vì chọn một công nghệ mới chỉ dành cho Windows Phone 8.

Vấn đề đặt ra là cần một ứng dụng là ngoài khả năng mô phỏng của một thiết bị thật mà còn cần phải mô phỏng lí tưởng trong toán học Vật lý. Và để giải quyết vấn đề trên Nhóm xây dựng cơ chế Nhập thông số cho mô phỏng nhằm áp dụng cho giải bài toán trong Vật lý. Đảm bảo các thông số được áp dụng đúng và chính xác.

Kết quả đạt được trong quá trình xây dựng ứng dụng từ bước mô tả, phân tích yêu cầu, thiết kế, cài đặt cho đến bước thử nghiệm và đánh giá cuối cùng là đã xây dựng thành công một ứng dụng mô phỏng đúng theo đề cương ban đầu. Các thông số trong quá trình mô phỏng có sự sai số rất thấp chỉ 1/100000 và có thể chấp nhận được. Giao

diện thân thiện và dễ thao tác, tốc độ ứng dụng chạy khá mượt và không có hiện tượng giật hay đứng khi sử dụng.

CHƯƠNG 1: GIỚI THIỆU

1.1 Giới thiệu chung về phần mềm mô phỏng

Để làm được một thí nghiệm khoa học nói chung và vật lý nói riêng, ta cần phải có các dụng cụ thí nghiệm. Các dụng cụ thí nghiệm có khi rất đắt, dễ hư hỏng hoặc bị hư hại làm cho độ chính xác của một thì nghiệm vật lý không còn chính xác nữa. Thời nay, dưới sự phát triển của công nghệ thông tin và sự bùng nổ của kỉ nguyên công nghệ di động là chính những điện thoại thông minh ta hoàn toàn có thể thực hiện gần như mọi thứ máy tính có thể làm được. Các thí nghiệm vật lý cũng như vậy, hoàn toàn có thể được mô phỏng trên các thiết bị Windows Phone giúp người dùng dễ dàng thực hiện các thí nghiệm vật lý hơn. Đồng thời các mô phỏng vật lý cũng đưa ra các thông số phân tích, giúp người thực hành thí nghiệm không phải đo đạc hoặc tính toán nhiều làm mất nhiều thời gian và công sức.

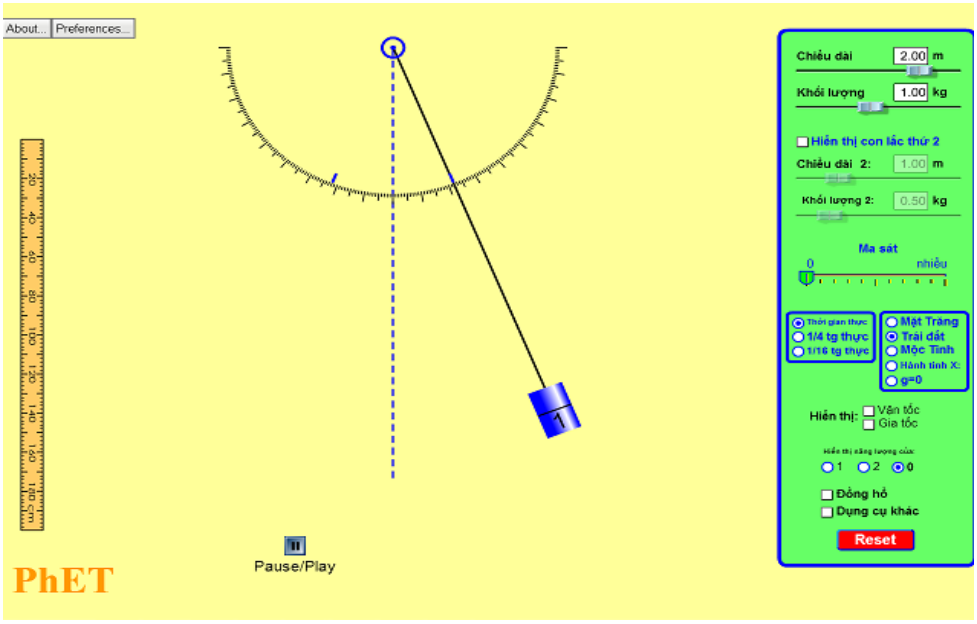
Mô phỏng vật lý là một khái niệm đã có từ lâu khi công nghệ thông tin phát triển. Các phần mềm mô phỏng cũng có rất nhiều dạng. Một số phần mềm phổ biến hiện nay là phần mềm do nhóm phát triển PhET (University of Colorado), phần mềm Crocodile, các phần mềm mô phỏng bằng flash trên nền web và một số phần mềm mô phỏng khác. Các phần mềm này đều chạy trên máy tính.

Việc xây dựng các mô phỏng Vật lý trên các thiết bị điện thoại thông minh nói chung và thiết bị Windows Phone nói riêng chưa thật sự nhiều và chưa đáp ứng được nhu cầu học tập tìm hiểu.

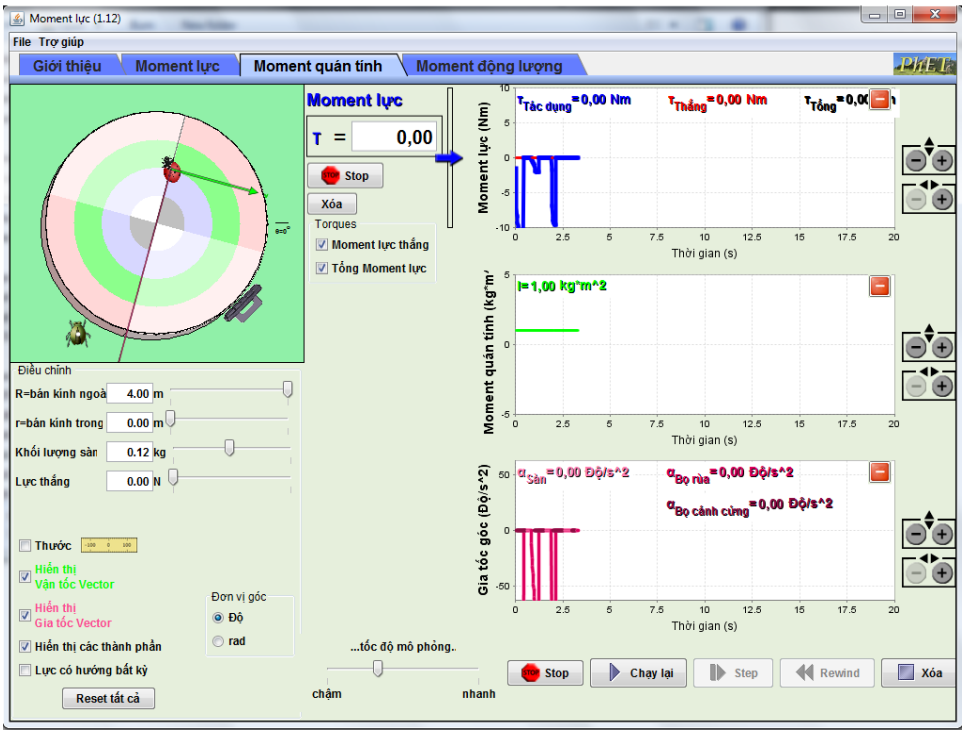
Để hiểu rõ hơn về mô phỏng Vật lý ta tìm hiểu một số các phần mềm mô phỏng Vật lý trên máy tính được đánh giá tốt và được sử dụng nhiều nhất hiện nay:

- **PhET** là một dự án gồm nhiều phần mềm mô phỏng vật lý của đại học Colorado. Bao gồm các phần mềm trên nền web và desktop. PhET là phần

mềm được nhiều người sử dụng (gần 100 triệu lượt dùng) và đã nhận được giải thưởng The Tech Award 2011 (vinh danh công nghệ giúp ích cho nhân loại).Một số hình ảnh trong PhET:

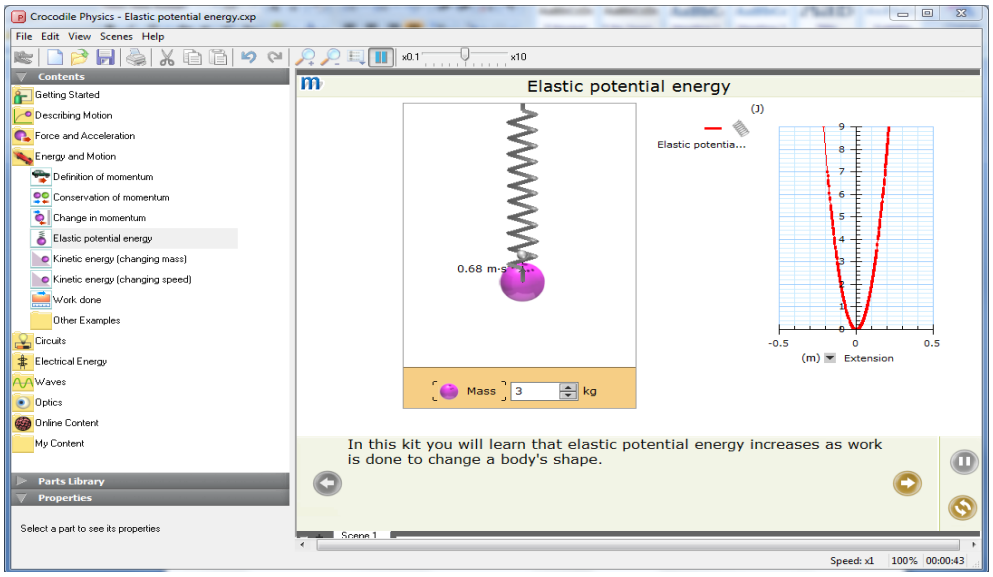


Hình 1.1



Hình 1.2

- **Phần mềm Crocodile** là một phần mềm mô phỏng vật lý khá đầy đủ. Crocodile mô phỏng rất nhiều dạng vật lý như: cơ học, quang học, điện.



Hình 1.3

1.2 Giới thiệu về phần mềm mô phỏng Vật lý của nhóm:

Phần mềm mô phỏng Vật lý của nhóm sẽ sử dụng công nghệ XNA/Silverlight trên Windows Phone để xây dựng. Cùng với đó là các kiến thức về Vật lý được sử dụng trong mô phỏng.

Các mô phỏng Vật lý trong ứng dụng có thể có:

Mô phỏng dao động của con lắc đơn: Một con lắc đơn được treo trên một sợi dây có chiều dài l và cho con lắc chuyển động hình vòng cung, mô phỏng được thực hiện với các điều kiện khác nhau như môi trường lí tưởng, môi trường nước, môi trường trong dầu có chỉ số alpha khác nhau, giải bài tập liên quan tới chuyên đề định trước.

Mô phỏng dao động con lắc lò xo: một con lắc lò xo được treo thẳng đứng. Phần mềm sẽ mô phỏng dao động của con lắc lò xo đó và đưa ra những phân tích, vẽ đồ thị, giải bài tập liên quan tới chuyên đề định trước.

Mô phỏng chuyển động ném xiên: Chuyển động ném xiên của một vật. Khi vật đó chạm mặt đất thì dừng lại. Mô phỏng đưa ra các phân tích quỹ đạo chuyển động của vật, phân tích vận tốc và các dạng năng lượng của vật, giải bài tập liên quan tới chuyên đề định trước.

Mô phỏng momen động lượng: dạng hiện tượng một thanh dài đồng chất chuyển động tròn quanh một trục cố định và vận tốc của vật thay đổi sau khi tăng chiều dài của vật, giải bài tập liên quan tới chuyên đề định trước.

Mô phỏng momen lực: dạng hiện tượng một đĩa tròn đồng chất chuyển động tròn quanh một trục cố định và vận tốc của vật thay đổi sau khi tăng chiều dài của vật, giải bài tập liên quan tới chuyên đề định trước.

Mô phỏng dao động điều hòa của sóng: dạng hiện tượng là thể hiện sự dao động điều hòa của sóng trên mặt nước tĩnh, tính toán thời gian vận tốc và sự giao thoa của hai sóng trên mặt nước, giải bài tập liên quan tới chuyên đề định trước.

Mô phỏng tổng hợp: dạng tổng hợp là một game. Không cần phải quá lớn nhưng cần tổng hợp lại được tất cả các chuyển động đã mô phỏng.

Phần mềm mô phỏng cần đạt được các tiêu chuẩn sau:

- Giao diện chương trình thân thiện.
- Nội dung thực hiện gần với thực tế.
- Đảm bảo hiển thị chi tiết quá trình chuyển động.
- Đảm bảo giải bài tập chính xác theo từng chuyên đề.
- Thiết kế Game giải trí tổng hợp các mô phỏng vật lý trên.

CHƯƠNG 2: NỀN TẢNG WINDOWS PHONE

2.1 Lập trình Di động

Những năm gần đây, sự cạnh tranh khốc liệt giữa các hãng sản xuất thiết bị di động đã mở ra một lối đi mới cho các nhà sản xuất ứng dụng trên thiết bị di động. Các công ty sản xuất và gia công phần mềm ngày càng ra sức mở rộng tầm hoạt động của mình sang mảng sản xuất ứng dụng trên điện thoại. Điều này đã đem lại cho các bạn trẻ đam mê công nghệ một hướng đi mới khi lựa chọn nghề nghiệp cho mình.

2.2 Xu hướng mới của ngành phát triển phần mềm

Nếu cách đây 10 năm, ngành công nghệ thông tin với vị trí lập trình phần mềm được nhiều bạn trẻ theo đuổi thì 4 năm trở lại đây, vị trí đó đã dần nhường chỗ cho các vị trí như "Lập trình viên điện thoại di động" hay "Lập trình viên Android hoặc iPhone". Sự phát triển đa dạng của các thiết bị di động cùng với sự bùng nổ của ngành công nghệ phần mềm trên lĩnh vực giải trí, nội dung số tại Việt Nam làm cho nghề lập trình ứng dụng trên điện thoại thông minh nghiễm nhiên trở thành vô cùng "hot".

Tuy nhiên đây là một ngành thật sự mới mẻ, nên việc tìm kiếm nguồn nhân lực có khả năng, đam mê, sáng tạo và nhiều kinh nghiệm chuyên môn là một thách thức với các nhà tuyển dụng. Phần lớn các ứng viên đều đã được đào tạo những ngôn ngữ lập trình cơ bản hoặc thậm chí là dồi dào kinh nghiệm về lập trình, nhưng khi bước vào lĩnh vực lập trình di động vẫn cần phải đào tạo lại. Chính vì vậy, bên cạnh giải pháp tuyển dụng đại trà rồi đào tạo nguồn lực mới, các nhà tuyển dụng cũng sẵn sàng trả lương cao để săn lùng các ứng viên có kinh nghiệm trong lĩnh vực này. Mức lương khởi điểm của vị trí lập trình di

động thường cao hơn khoảng 30%-40% so với các vị trí lập trình thông thường. Đặc biệt là trong lĩnh vực lập trình trên hệ điều hành Windows Phone thì nhu cầu nhân lực thực sự trở thành một trào lưu mới đối với các hãng công nghệ.

2.3 Môi trường làm việc của Lập trình viên di động

Nếu tham gia vào các công ty sản xuất phần mềm, bạn sẽ được thỏa mãn niềm đam mê trong môi trường làm việc đầy sang tạo, được trực tiếp làm việc trên các thiết bị di động tối tân nhất, lương cao, nhiều cơ hội thăng tiến... Ngoài ra, nếu bạn là người sáng tạo và quen làm việc độc lập, thì có thể tự phát triển ứng dụng và đưa lên Android Market hay Apple Store, các ứng dụng tính phí trên Android Market/Apple Store đang dần dần cân bằng với các ứng dụng miễn phí. Còn nếu bạn có nhóm phát triển thì việc nhận các việc làm freelance và các dự án out-source cũng đem lại một nguồn thu tương đối tốt hiện nay.

Xây dựng ứng dụng trên Smartphone là lĩnh vực có tốc độ phát triển nhanh nhất trong ngành CNTT với 1.4 tỉ thuê bao và dự báo sẽ có khoảng 452 triệu Smartphone được bán ra vào năm 2012 và 6,7 triệu ứng dụng sẽ được tải về trước năm 2014. Có thể nói, chưa bao giờ thị trường thiết bị di động và các ứng dụng trên thiết bị di động lại sôi nổi và được quan tâm như hiện nay. Một lượng lớn những lập trình viên chuyên nghiệp đang là nhu cầu rất cấp thiết cho thị trường hấp dẫn và thú vị này.

2.4 Sơ lược về nền tảng Windows Phone

2.4.1 Hệ điều hành Windows Phone

Windows Phone là hệ điều hành của Microsoft dành cho smartphone kế tục nền tảng Windows Mobile, mặc dù chúng không tương thích với nhau. Khác với Windows Mobile, Windows Phone tập trung vào sự phát triển của Marketplace - nới các nhà phát triển có thể cung cấp sản phẩm (miễn phí hoặc

có phí) tới người dùng. Windows Phone được bán vào tháng 10 năm 2010 và đầu năm 2011 tại Châu Á.

Phiên bản mới nhất hiện tài là Windows Phone 8. Microsoft còn đang phát triển bản Windows Phone Apollo Plus, và trong tương lai có thể còn có Windows Blue (hay có thể là Windows 9) giúp tương thích với hệ điều hành Windows trên máy tính. Với Windows Phone, Microsoft đã phát triển giao diện người dùng mới mang tên Modern (trước đây tên là Metro) - tích hợp khả năng liên kết với các phần cứng và phần mềm của hãng thứ ba một cách dễ dàng.

Microsoft Windows Phone là điện thoại có số lượng người tiêu dùng rất lớn bởi vì nó có có tất cả các tính năng mà người dùng đã quen thuộc như Iphone của Apple và điện thoại thông minh hỗ trợ Android, như Motorola Droid và HTC Incredible. Những tính năng này bao gồm cảm ứng đa điểm, giao diện người dùng đẹp (UI) có bổ sung một thiết kế mới hiện đại của Microsoft có tên là Metro, các dịch vụ mạng xã hội như Facebook, và hỗ trợ tài khoản e-mail phổ biến như Yahoo, Hotmail, Google, AOL, hoặc nếu bạn là một công ty người dùng, Microsoft Exchange.

Độc đáo hơn, điện thoại cùng với một phiên bản của Microsoft Office có thế sử dụng để đọc, chỉnh sửa, lưu và đồng bộ bất kỳ tập tin Word, bảng tính Excel, và các định dạng Office khác, làm cho nó trở thành một chiếc điện thoại tuyệt vời dành cho những người sử dụng Office tại nhà hoặc tại văn phòng. Windows Phone cũng có thể tích hợp với Xbox LIVE, làm cho nó một sự lựa chọn tuyệt vời cho game thủ. Microsoft Windows Phone sử dụng phần mềm Zune để đồng bộ hóa các ứng dụng cài đặt, hình ảnh, âm thanh, sao lưu và cập nhật hệ thống. Là một nhà phát triển ứng dụng, bạn cũng sẽ sử dụng Zune kết hợp với Visual Studio để gỡ lỗi các ứng dụng của bạn trên một thiết bị thực sự.

Microsoft cũng giới thiệu về trung tâm cho các điện thoại sử dụng Windows Phone: một trung tâm mà người sử dụng có thể lưu trữ tất cả các địa chỉ liên lạc và kết nối mạng xã hội, một trung tâm âm nhạc nơi người dùng có thể nghe, tải về, và mua nhạc cùng với một số ứng dụng trung tâm, còn được gọi là Marketplace, nơi mà bạn sẽ được quan tâm nhất, vì bạn sẽ được xuất bản và bán các ứng dụng mà bạn tạo ra.

Khi Giới thiệu về Windows Phone Microsoft đã áp đặt các chi tiết kỹ thuật phần cứng trên điện thoại của các nhà sản xuất, giúp cho bạn có thể dễ dàng phát triển một ứng dụng mà không cần lo lắng về các thiết bị phần cứng cụ thể của máy. Đối với bất kỳ phiên bản tương lai của điện thoại, bạn đều được đảm bảo rằng ứng dụng bạn viết ngày hôm nay sẽ làm việc tốt trên bất kì thương hiệu nào của điện thoại. Đương nhiên là bạn phải biết ngôn ngữ mà bạn cần để làm việc trên Windows Phone. Ngôn ngữ cho sự lựa chọn hiên nay là C#, Visual Basic (VB). Ngoài ra bạn còn có hai lựa chọn là: Silverlight hoặc XNA. Silverlight và XNA là hai sử dụng cốt lõi của .NET Framework.

2.4.2 Hoàn cảnh ra đời

Sau thành công của nền tảng Windows dành cho PC, Microsoft tiếp tục bước vào nền tảng dành cho các thiết bị di động. Windows Phone bắt đầu được nhen nhóm vào đầu năm 2004 như là một bản nâng cấp cho Windows Mobile với tên mã "Photon", nhưng công việc diễn ra rất chậm và dự án phải bị hủy. Năm 2008, dự án được khởi động trở lại, nhưng lần này không phải là một bản nâng cấp mà là một hệ điều hành mới hoàn toàn. Mặc dù được dự kiến phát hành vào năm 2009, nhưng sự chậm trễ trong việc phát triển dẫn tới phiên bản Windows Mobile 6.5 vẫn được phát hành.Việc kết thúc hỗ trợ cho Windows Mobile chỉ diễn ra vào ngày 15/7/2011.

Trong giai đoạn này Windows Phone được phát triển khá nhanh, kéo theo đó là việc không thể tương thích với các phiên bản cũ do không kịp thời gian chuẩn bị cho việc đó.

Tên mã của dự án Windows Phone là "Photon". Ban đầu tên gọi dự định sẽ là Windows Phone . Tuy nhiên vào ngày 22 tháng 4 năm 2010, Microsoft chính thức thông báo tên gọi phiên bản đầu tiên là Windows Phone 7 - tương xứng với hệ điều hành Windows 7 dành cho PC.

Metro Design:

Microsoft đang nhắm mục tiêu Windows Phone 7 đến với những người bận rộn và cung cấp giao diện người dùng hấp dẫn, Microsoft đã đưa ra thiết kế Metro. Metro là một thiết kế xuất phát từ ngành công nghiệp giao thông vận tải, kiểu chữ và thiết kế hình ảnh giao diện giúp cho những người bận rộn liên tục quét và đi, và vì điều này, Metro thiết kế nhấn mạnh vào thiết kế đơn giản và sạch sẽ. Metro được thiết kế theo năm nguyên tắc. Đầu tiên nguyên tắc nhấn mạnh về sạch sẽ, ánh sáng, cởi mở, nhanh chóng để loại bỏ lộn xộn, và kiểu chữ, khi người tiêu dùng sẽ được sử dụng điện thoại để đọc e-mail, tin nhắn SMS, Facebook, và Twitter trong khi trên đường đi. Nguyên tắc thứ hai của Metro là thiết kế tập trung vào nội dung, thiết kế lấy việc trình bày nội dung làm tiền đề chính. Nguyên tắc thứ ba tập trung vào việc tích hợp phần cứng và phần mềm. Nguyên tắc thứ tư đặt trọng tâm vào cử chỉ, thiết kế cho phép sử dụng cảm ứng đa điểm đẳng cấp thế giới. Cuối cùng, các khái niệm thiết kế Metro tập trung vào một ứng dụng là có hồn và sống động, nơi mà thông tin quan trọng nhất cho người sử dụng được trình bày một cách rõ ràng và dễ dàng cho người dùng truy cập.

Windows Phone 7:

Windows Phone 7 được ra mắt vào ngày 15 tháng 2 năm 2010 ở Mobile World Congress tại Barcelona, Tây Ban Nha và chính thức bán ra vào ngày 8 tháng 11 năm 2010 tại Mỹ.

Ban đầu Microsoft phát hành bản cập nhật No Do, tiếp sau đó là bản nâng cấp lớn Mango (còn được biết là Windows Phone 7.5) vào tháng 5 2011. Bản cập nhật này bao gồm phiên bản di động của Internet Explorer 9, đa nhiệm cho phần mềm của công ty thứ ba, hợp nhất Twitter vào People Hub, và cho phép đăng nhập SkyDrive.

Một bản nâng cấp nhỏ được phát hành năm 2012 là "Tango". Trong bản cập nhật này, Microsoft đã sửa những lỗi bug, hạ thấp cấu hình tối thiểu cho Windows Phone xuống chip 800Ghz và RAM 256Mb để phù hợp cho những máy giá rẻ cấu hình thấp.

Tháng 1 năm 2012, Microsoft tung ra bản Windows Phone 7.8. Nó bổ sung thêm những tính năng từ Windows Phone 8, chẳng hạn như màn hình chủ, tăng số lượng tông màu lên 20 và khả năng đặt màn hình khóa là hình ảnh trong ngày của Bing. Windows Phone 7.8 nhằm kéo dài tuổi thọ của các thiết bị Windows Phone 7, vì chúng không thể nâng cấp lên Windows Phone 8 bởi giới hạn phần cứng. Windows Phone 7.8 vẫn sẽ được Microsoft hỗ trợ trong thời gian tới song song với Windows Phone 8. Dự kiến Microsoft ngừng hỗ trợ bản 7.8 kể từ ngày 9 tháng 9 năm 2014.

Windows Phone 8:

Ngày 20 tháng 6 năm 2012, Microsoft giới thiệu Windows Phone 8, một thế hệ hệ điều hành mới, và 4 tháng sau vào ngày 29 tháng 10 năm 2012, Microsoft bắt đầu bán phiên bản này. Windows Phone 8 thay thế lõi kiến trúc Windows CE trên Windows Phone 7 thành kernel của Windows NT vốn được

thiết kế cho Windows 8, chính vì vậy điều này đã làm cho ứng dụng dễ dàng được port giữa hai hệ điều hành. Ngoài ra, Windows Phone 8 còn hỗ trợ CPU đa nhân, nhiều độ phân giải, tùy biến Start Screen, bổ sung IE10, Nokia Maps thay thế Bing Maps. Theo Microsoft, Windows Phone 8 sẽ được hỗ trợ đến ngày 8 tháng 7 năm 2014.

2.5 Lập trình Windows Phone

Như bạn đã biết Windows Phone là một nền tảng mới, đầy tiềm năng cho lập trình viên và những ai yêu thích công nghệ Microsoft nói chung và nền tảng Windows Phone nói riêng.

Hiện nay để xây dựng một ứng dụng cho nền tảng Windows Phone có rất nhiều cách thức và ngôn ngữ để xây dung chẳng hạn như bạn có thể dụng C#, HTML5,… Nhưng chủ yếu vẫn là ngôn ngữ C# được dùng để xây dựng ứng dụng.

Microsoft đã không phát minh ra bất kỳ ngôn ngữ mới hoặc các khuôn khổ mới cho các ứng dụng chạy nền tảng Windows Phone mà đơn giản chỉ là thích nghi với các khuôn khổ hiện tại của mình. Điều này có nghĩa là bạn sẽ có thể cho chương trình sử dụng C# với .NET Framework. .NET cung cấp các thư viện lớp cơ sở phổ biến mà mỗi lập trình viên Microsoft.NET đã quen thuộc, bao gồm cả hỗ trợ đa luồng, XML, LINQ, bộ sưu tập, các sự kiện, dữ liệu, trường hợp ngoại lệ, nhập xuất IO, mô hình dịch vụ, mạng, văn bản, vị trí, sự phản ánh, nguồn tài nguyên, thời gian chạy, an ninh, và chẩn đoán.

Nền tảng ứng dụng cho Windows Phone bao gồm hai khuôn khổ chính là: Silverlight và XNA. Silverlight chủ yếu sử dụng cho các ứng dụng kinh doanh đơn giản và trò chơi 2D. Silverlight sử dụng các ứng dụng Extensible Markup Language (XAML) là khai báo đánh dấu ngôn ngữ để tạo ra giao diện người

dùng hấp dẫn. Các nhà thiết kế sẽ có sự linh hoạt rất lớn trong việc tạo ra giao diện người dùng cho Windows Phone bằng cách sử dụng các công cụ quen thuộc như Adobe Illustrator, Photoshop, và thiết kế Microsoft Expression để tạo ra dựa trên giao diện người dùng có thể dễ dàng xuất sang XAML. XNA là chủ yếu được sử dụng để tạo ra trò chơi, và khung đi kèm với một công cụ trò chơi cho phép bạn tạo ra các vòng lặp dựa trên trò chơi và cũng cung cấp một công cụ 2D, cho phép bạn tạo ra các trò chơi 2D.

Application:

Để viết một ứng dụng phục vụ một hay nhiều nhu cầu trong đời sống thì có thể sử dụng Silverlight định nghĩa sẵn cho lập trình viên những công cụ cũng như những thành phần sẵn có. Công cụ Expression Blend giúp lập trình viên từ người biết hay không biết nhiều về design cũng có thể tùy chỉnh giao diện cho chương trình của mình một cách dễ dàng nhất.

Game:

Để viết game cho Windows Phone thì khi còn là Windows Phone 7 thì ta sử dụng XNA để xây dựng. Hiện nay với Windows Phone 8 bạn còn có thêm những lựa chọn khác là DirectX, HTML5.0,…

Giả lập Windows Phone (Windows Phone Emulator):

Điện thoại Windows giả lập được tích hợp trong Visual Studio mô phỏng một thiệt bị thực tế. Tuy nhiên, có những điều bạn không thể làm trong giả lập, giống như thử nghiệm gia tốc, GPS, la bàn, FM radio, tin nhắn SMS, tính năng e-mail, gọi điện thoại, danh sách liên lạc, máy ảnh, và các đòi hỏi về thiết bị vật lý. Tuy nhiên, một kỹ thuật gọi là phản ứng Tiện ích mở rộng, bạn có thể sử dụng để mô phỏng các nguồn cấp dữ liệu mà bạn có thể mong đợi trên một điện thoại thực sự. Ví dụ, bạn sẽ tìm hiểu sự phản ứng lại của thiết bị như thế nào,

bạn có thể mô phỏng đo gia tốc và GPS để bạn có thể làm việc với thiệt bị giả lập mà không cần phải có thiệt bị thật.

CHƯƠNG 3: GIỚI THIỆU CÔNG NGHỆ XNA WINDOWS PHONE

3.1 Tổng quan về XNA

3.1.1 Lịch sử phát triển XNA:

XNA được phát triển bởi Microsoft cách đây vài năm nhưng khi đó nó vẫn là một bí mật và không được nhiều người biết đến. Vào năm 2004, Microsoft giới thiệu XNA lần đầu tiên tại hội nghị phát triển trò chơi lớn nhất hàng năm (Game Developers Conference). XNA không chỉ là một framework giống DirectX, nó còn chứa nhiều công cụ và thậm chí là một môi trường phát triển tương thích với Visual Studio để giúp phát triển các ứng dụng trò chơi một cách dễ dàng.

XNA có thể là một kế thừa của DirectX Framework, nhưng khi Direct2D 10 Beta cho Windows Vista được phát hành vào cuối năm 2005, DirectX vẫn là framework đồ họa dành cho hệ điều hành mới này. Sau đó, đầu năm 2006 tại GDC bản Microsoft XNA Build March 2006 CTP được phát hành. XNA Build là một công cụ cho phép quản lí các tiến trình xây dựng các tác vụ phức tạp, tương tự như là MsBuild và Ants, nhưng phức tạp và hiệu quả hơn. Nhưng sau một thời gian không có gì xảy ra, đặc biệt là các nhóm phát triển vừa và nhỏ không thật sự cần một công cụ quản lí xây dựng phức tạp.

Tại hội nghị Gamefest vào tháng tám (hội nghị phát triển trò chơi mới được tổ chức bởi Microsoft), ở đó Microsoft thông báo XNA Game Studio Express beta 1 được phát hành vào 30/8/2006. Bản beta đầu tiên chỉ chứa một dụng cụ khởi đầu, “Space Wars”, và XNA không bao gồm nhiều tính năng 2D. Nhiều nhà phát triển và người yêu thích thử dùng XNA và viết nhiều trò chơi 2D nhỏ với sự giúp đỡ của lớp Sprite trong XNA. Khi đó, việc tạo mô hình 2D theo ý mình là một điều rất khó.

3.1.2 Các phiên bản của XNA Framework:

XNA ban đầu là tên của dự án phát triển, Xbox New Architecture. Microsoft dự định phát hành máy chơi game Xbox, tuy nhiên năm 2005 Xbox 360 được ra đời thay cho tên Xbox và XNA được dùng cho công cụ phát triển game trong toàn bộ hệ thống Microsoft.

- Bộ công cụ XNA được công bố ngày 24 Tháng Ba năm 2004, tại Hội nghị các nhà phát triển game tại San Jose, California.
- Bản Preview của XNA đã được phát hành vào ngày 14 Tháng 3, 2006.
- XNA Game Studio 2.0 được phát hành vào tháng Mười Hai năm 2007, tiếp theo là XNA Game Studio 3.0 vào ngày 30 tháng Mười 2008.
- XNA Game Studio 4.0 được phát hành vào ngày 16 Tháng Chín năm 2010 cùng với công cụ phát triển Windows Phone 7.

XNA hiện tại là toàn bộ môi trường phát triển Game của Microsoft, bao gồm bộ Kit phát triển game cho Xbox và XNA Game Studio.

3.2 Cấu trúc cơ bản một Project XNA:

Trong file *Game1.cs*:

File *Game1.cs* là nơi tất cả các đối tượng được khai báo và thực hiện việc xử lí.

Các lệnh using cần cho XNA:

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
```

Biến đầu tiên trong class Game1:

```
GraphicsDeviceManager graphics;
SpriteBatch spriteBatch;
```

Hàm khởi tạo:

```
public Game1()
{
graphics = newGraphicsDeviceManager(this);
Content.RootDirectory = "Content";
}
```

Hàm tạo thông tin giá trị đầu tiên cho các thành phần trong game:

```
protected override void Initialize()
{
 base.Initialize();
}
```

Load Content của Game:

```
protectedoverridevoid LoadContent()
 {
// Create a new SpriteBatch, which can be used to draw textures.
 spriteBatch = newSpriteBatch(GraphicsDevice);
 }
```

UnloadContent khi chuyển sang màn hình khác hoặc thoát Game

```
protectedoverridevoid UnloadContent()
 {
// Free the previously alocated resources
 spriteBatch.Dispose();
 }
```

Hàm này sẽ cập nhật các giá trị của các biến trong game để tạo ra sự thay đổi đặc trưng vốn có của game:

```
protectedoverridevoid Update(GameTime gameTime)
 {
// Allows the game to exit
if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
this.Exit();
// TODO: Add your update logic here
base.Update(gameTime);
 }
```

Hàm này giúp cho việc render các đối tượng do người dùng tạo, hiện ra trên màn hình:

```
protectedoverridevoid Draw(GameTime gameTime)
 {
 graphics.GraphicsDevice.Clear(Color.CornflowerBlue);
base.Draw(gameTime);
 }
```

Trong file *Program.cs*:

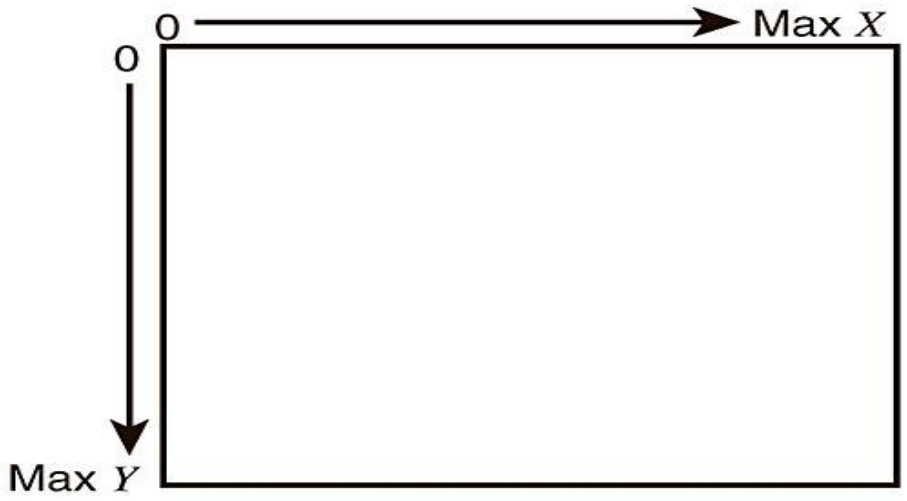
```
using System;
namespace DemoGame2DXNA
{
#if WINDOWS || XBOX
staticclassProgram
 {
staticvoid Main(string[] args)
 {
using (Game1 game = newGame1())
 {
 game.Run();
 }
```

Nội dung File này nên không chỉnh sửa(tuy nhiên có thể thay đổi tùy coder).

3.3 Đồ họa 2D trong XNA

3.3.1 Hệ tọa độ 2D:

Hệ tọa độ 2D trong XNA có hơi khác so với hệ tọa độ 2D trong hình học. Hệ toa độ được sử dụng trong XNA được gọi là hệ tọa độ màn hình. Theo mặc định của một project XNA đối với chế độ màn hình nằm ngang thì ta sẽ có màn hình có chiều dài 800 pixel và chiều cao là 480 pixel, đối với màn hình đúng thì ngược lại.



Hình 3.1

3.3.2 Một số định nghĩa:

Sprite:

- Sprite là một hình ảnh thường dùng để hiển thị thông tin như: thanh trạng thái của đối tượng, số mạng sống, chú thích, điểm số. Để vẽ sprite, chúng ta phải tạo đối tượng Texture2D. Ta có thể hiệu chỉnh sprite như: tạo màu sắc, thực hiện phép quay, phép tỉ lệ bằng phương thức Draw. Sau khi vẽ sprite, gọi phương thức End trước khi gọi phương thức Present.

Sprite Origin:

- Khi vẽ sprite, gốc sprite là khái niệm quan trọng. Gốc là một điểm cụ thể trên sprite được mặc định ở góc trên bên trái của sprite hoặc tại tọa độ (0,0). Phương thức Draw vẽ gốc của sprite tại vị trí được chỉ định trên màn hình.
- Ví dụ: nếu ta vẽ pixel sprite 50x50 tại tọa độ (400,200) mà không xác định gốc, góc trên bên trái của sprite sẽ là điểm (400,200). Nếu dùng điểm trung tâm của sprite 50x50 với gốc là (25,25), để vẽ sprite tại vị trí đó, ta phải cộng thêm tọa độ gốc. Trong trường hợp này, vị trí là (425,425) và gốc là (25,25).

SpriteDepth:

- Sprite cũng có khái niệm độ sâu. Khi vẽ sprite, chúng ta có thể xác định độ sâu giữa 0 và 1 (đây là một số thực). Sprite vẽ tại độ sâu 0 nằm "trước" màn hình và sẽ phủ vài sprite vẽ ở độ sâu thấp hơn. Sprite ở độ sâu 1 nằm "sau" màn hình và sẽ bị phủ bởi những sprite vẽ ở độ sâu nhỏ hơn 1.

Sprite Batching:

- Thông thường, đối tượng SpriteBatch không thay đổi vài trạng thái tạo hoặc vẽ sprite cho đến khi gọi phương thức End. Điều này được gọi là chế độ Deferred. Ở chế độ Deferred, SpriteBatch lưu các thông tin mỗi khi gọi phương thức Draw cho đến khi gọi phương thức End. Khi gọi phương thức End, SpriteBatch thay đổi các thiết bị cài đặt đồ họa và vẽ mỗi sprite một loạt. Sau đó, phương thức End trả về những thiết lập ban đầu nếu ta chỉ định SaveStateMode.SaveState.
- Nếu gọi phương thức Begin, xác định SpriteSortMode.Immediate, nó tạo chế độ Immediate. Trong chế độ Immediate, ngay lập tức SpriteBatch thay đổi trạng thái thiết bị đồ họa để bắt đầu vẽ sprite. Sau đó, mỗi lần gọi phương thức Draw, nó vẽ sprite bằng cách sử dụng các thiết bị cài đặt hiện tại. Phương thức End trả về những thiết lập thiết bị ban đầu nếu ta chỉ định SaveStateMode.SaveState.
- Ở chế độ Immediate, một khi gọi phương thức Begin trên một thể hiện (instance) SpriteBatch, chúng ta không được gọi nó ở thể hiện SpriteBatch khác, cho đến khi gọi phương thức End cho SpriteBatch đầu tiên.
- Chế độ Deferred chậm hơn chế độ Immediate, nhưng nó cho phép nhiều thể hiện của SpriteBatch chấp nhận phương thức Begin và Draw mà không cản trở lẫn nhau.

Sprite Blend Modes:

- Chế độ trộn có thể tạo ra AlphaBlend, Additive, None. Chế độ trộn (blend) mặc định là AlphaBlend.

Sprite Sort Mode:

- Chế độ sắp xếp quyết định các ảnh (sprite) được vẽ hiển thị trên hình ảnh. Trước đây, các nhà phát triển trò chơi phải cẩn thận để đảm bảo việc các hình ảnh hiển thị trên màn hình phải được sắp xếp, pha trộn màu nền và bề mặt một cách chính xác. Ngày nay, với sự hiểu biết về XNA sẽ giúp chúng ta làm công việc này đơn giản hơn.
- Các chế độ sắp xếp có thể thiết lập: BackToFront, Deferred, FrontToBack, Immediate và Texture.
- Chế độ sắp xếp mặc định là Deferred khi chúng ta gọi phương thức Begin không có tham số.
- Immediate cập nhật việc thiết lập thiết bị đồ họa ngay lập tức khi phương thức Begin được gọi.
- Chế độ BackToFront và FrontToBack. Khi chúng ta vẽ ảnh, chúng ta có thể thiết lập độ sâu của ảnh. Giá trị này là một số thực giữa 0.0 và 1.
  - BackToFront dùng cho hình ảnh trong suốt.
  - FrontToBack dùng cho hình ảnh không trong suốt.
  - Chế độ Texture sẽ kiểm tra tất cả hình ảnh được yêu cầu và sắp xếp chúng.

Sprite Transformation Matrices

- XNA Game Studio bao gồm một tính năng cho hàng loạt sprite, khả năng xác định chuyển đổi ma trận có thể áp dụng cho mỗi sprite trước khi vẽ. Việc chuyển đổi ma trận có thể được kết hợp với phép tịnh tiến (translation), phép xoay (rotation), hoặc phép tỉ lệ (scaling) ma trận nhân với nhau thành một ma trận đơn. Đây là ma trận kết hợp với các tham số vị trí, phép xoay, phép tỉ lệ và chiều sâu của sprite để cung cấp cho phương thức Draw. Bởi vì ma trận cũng áp dụng chiều sâu, sự chuyển đổi tọa độ để làm cho độ sâu sprite lớn hơn 1.0 hoặc nhỏ hơn 0.0 sẽ làm sprite biến mất.

Sprite Fonts

- XNA Game Studio cho phép vẽ văn bản bằng cách sử dụng SpriteBatch. Phương thức DrawString sẽ vẽ văn bản trên màn hình với vị trí, màu sắc, góc quay, gốc, và hệ số tỉ lệ. Phương thức DrawString cũng yêu cầu loại texture của lớp SpriteFont. SpriteFont được tạo ra bởi Content Pipeline khi ta thêm một tập tin Sprite Font vào ứng dụng của mình. Các tập tin Sprite Font có các thông tin như tên và kích cỡ của phông chữ, và trong đó bao gồm các ký tự Unicode trong texture SpriteFont. Khi chạy ứng dụng, một SpriteFont được nạp với ContentManager.Load giống như một đối tượng Texture2D.

3.3.3 Các bước tạo và vẽ đối tượng 2D trong XNA

3.3.3.1 Tạo class cho một đối tượng trong Game:

Câu lệnh using cần có trong một class của XNA:

```
using Microsoft.Xna.Framework.Graphics; // để sử dụng Texture2D
using Microsoft.Xna.Framework; // để sử dụng Vector2
```

Khai báo biến:

```
publicTexture2D texture { get; set; } // sprite 2D
publicVector2 position { get; set; } // vị trí sprite trên nền
publicVector2 size { get; set; } // kích thước của sprite
```

Sử dụng Vector2 cho biến size và position (X, Y), Texture2D cho biến texture.

Hàm khởi tạo:

```
Public clsSprite(Texture2D newTexture, Vector2newPosition, Vector2 newSize,int ScreenWidth, int ScreenHeight)
{
texture = newTexture;
position = newPosition;
size = newSize;
}
```

Hàm đồ họa (Draw()):

```
// vẽ các sprite lên màn hình
publicvoid Draw(SpriteBatch spriteBatch)
{
spriteBatch.Draw(texture, position, Color.White);
}
```

3.3.3.2Đưa đối tượng vào Game:

Bổ sung biến trong phần khai báo biến:

```
clsSprite mySprite1;
```

Bổ sung chiều cao và chiều rộng cho cửa sổ game:

```
public Game1()
 {
graphics = newGraphicsDeviceManager(this);
graphics.PreferredBackBufferWidth = 800;
 graphics.PreferredBackBufferHeight = 480;
 graphics.IsFullScreen = false;
 Content.RootDirectory = "Content";
 statusPlayer = false;
}
```

Bổ sung trong hàm LoadContent():

```
// Load một sprite với “ball” là file ảnh đã được add vào content
mySprite1 = newclsSprite(Content.Load<Texture2D>("ball"),
newVector2(0f, 0f), newVector2(64f, 64f));
```

Bổ sung trong hàm UnloadContent():

```
mySprite1.texture.Dispose();
```

Vì chưa có xử lý nên hàm Update() giữ nguyên.

Bổ sung trong hàm Draw() (Thực hiện đưa Sprite lên bằng một texture tại vị trí nhất định):

```
spriteBatch.Begin(SpriteSortMode.FrontToBack, BlendState.AlphaBlend);
mySprite1.Draw(spriteBatch);
spriteBatch.End();
```

3.4 Đồ họa 3D trong XNA

3.4.1 Hệ tọa độ 3D:

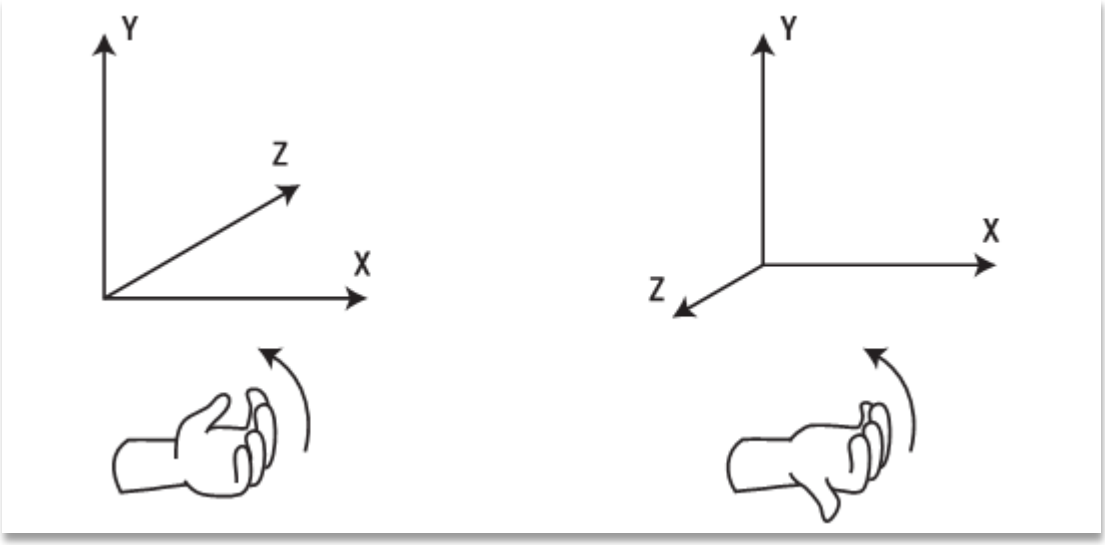
Khi nói về hệ tọa độ 3 chiều ta cần phải tìm hiểu thêm nhiều quan niệm để hiểu rõ các vấn đề về việc tạo ra một mô hình 3D, và các phép biến đổi đối tượng 3D sang dạng hiển thị hai chiều để có thể hiển thị lên màn hình.

Khi thao tác với hệ tọa độ Đề-các 3D, cần chú ý hai loại hệ thống tọa độ sau: quy tắc bàn tay trái và quy tắc bàn tay phải. Hai loại này khác nhau ở chỗ vị trí của trục Z so với hai trục còn lại X và Y.

Để quyết định vị trí này, ta chỉ các ngón tay của một bàn tay theo hướng dương của trục X, và di chuyển chúng theo hướng ngược chiều kim đồng hồ đến vị trí dương của trục Y. Hướng của trục Z là hướng của ngón tay cái.

Trong hệ tọa độ bàn tay trái, giá trị của Z (theo hướng dương) sẽ lớn hơn khi ta đi từ màn hình đến điểm xa ta (ta xem trục X và Y đang nằm trên màn hình). Tức là phần Z (+) hướng vào trong (Z: càng đi vào thì càng lớn).

Trong hệ tọa độ bàn tay phải thì ngược lại, giá trị của Z sẽ tăng dần hướng về ta tính từ màn hình. Tức là phần Z(+) hướng ra ngoài (Z: càng đi ra thì càng lớn.)



Hình 3.2

Theo mặc định, XNA Frameword chọn hệ tọa độ bàn tay phải để làm việc. Điều này khác với DirectX (chọn hệ tọa độ bàn tay trái). Điều này có nghĩa là các giá trị âm Z (–) là hữu hình và càng âm thì càng xa màn hình. Các giá trị dương Z (+) lớn thì không được hiển thị trừ khi chúng ta "thay đổi vị trí camera".

3.4.2 Ánh xạ đối tượng 3D lên màn hình 2D:

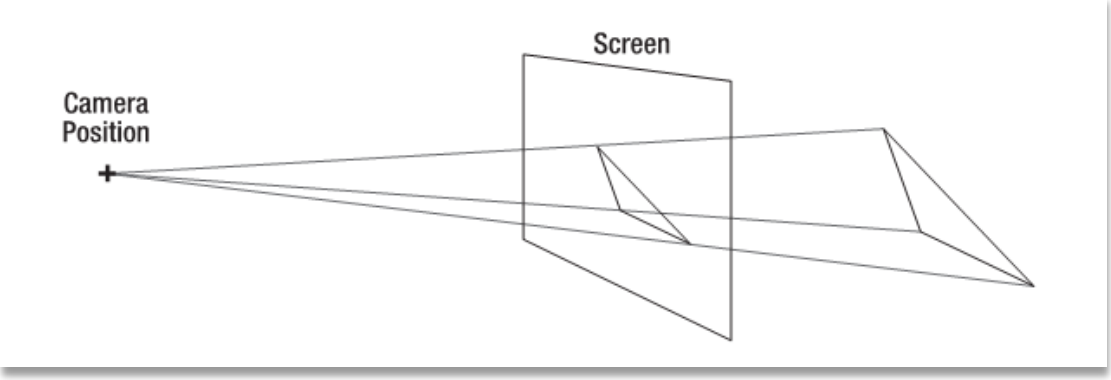
XNA đã giúp ta thực hiện tất cả các công việc tính toán cho việc ánh xạ này, nhưng ta vẫn cần phải hiểu định nghĩa "Phép chiếu" và cách để áp dụng vào XNA để góp phần hiển thị các đối tượng lên màn hình.

Tương tự như các thư viện lập trình trò chơi khác, XNA hỗ trợ hai loại phép chiếu như sau:

- Phép chiếu bối cảnh (Perspective Projection): Đây là kiểu phép chiếu phổ biến nhất, nó sử dụng khoảng cách Z và điều chỉnh các đối tượng cho phù hợp.
  - Phép chiếu này làm cho các đối tượng xuất hiện nhỏ hơn khi xa dần từ màn hình.

- Tùy vào vị trí mà các đối tượng cũng có thể bị biến dạng giống như thế giới thực (realworld).
- Ví dụ, đối với một hình lập phương, các mặt nào gần màn hình hơn thì được xem như là lớn hơn so với các mặt xa màn hình hơn.

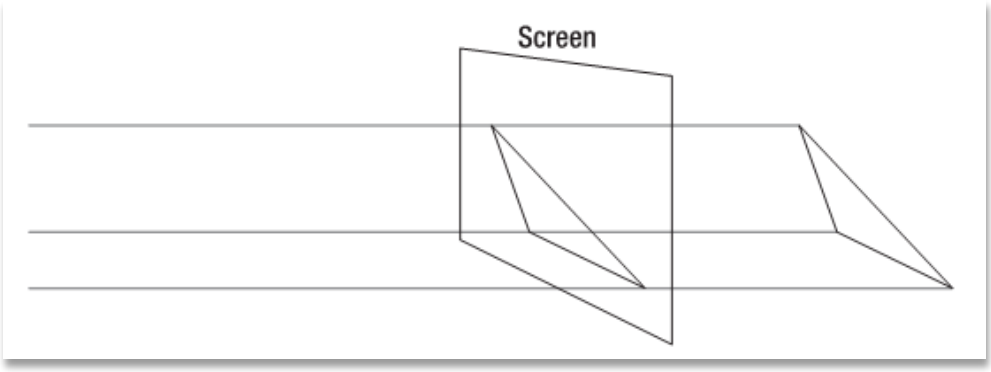
Sau đây là hình ảnh mô tả phép chiếu phối cảnh:



Hình 3.3

- Phép chiếu trực giao (Orthogonal Projection): Đối với kiểu chiếu này, thành phần Z được bỏ qua, và các đối tượng không lớn hơn khi gần màn hình hoặc không nhỏ hơn khi xa màn hình. Phép chiếu này hầu hết được dùng trong trò chơi 2-D (trò chơi giả 3-D, chỉ để đặt một số sprite lên trên các sprite khác) hoặc một trò chơi 3-D đơn giản.

Sau đây là hình ảnh mô tả phép chiếu trực giao:



Hình 3.4

3.4.3 Đỉnh và các hình cơn bản:

Phần cơ bản nhất của một đối tượng 3-D là một đỉnh (vertex). Về mặt toán học, các đỉnh được hiển thị độc lập bằng hệ tọa độ 3-D (được ánh xạ thành kiểu dữ liệu Vector3 trong XNA), nhưng trong XNA, chúng còn bao gồm thêm các thông tin phụ như màu sắc, texture hoặc vector pháp tuyến, tùy vào định dạng đỉnh được sử dụng.

Sau đây là một số định nghĩa mà XNA đã cung cấp:

Bảng 3.1

Định dạng đỉnh	**Mô tả**
VertexPositionColor	Định nghĩa một đỉnh kèm theo thông tin vị trí và màu sắc.
VertexpositionTexture	Định nghĩa một đỉnh mang thông tin vị trí, các tọa độ texture, xác định cách để ánh xạ một texture cho trước lên đỉnh này, với (0,0) là tọa độ phía trên bên trái và (1,1) là giới hạn phía dưới bên phải của texture.

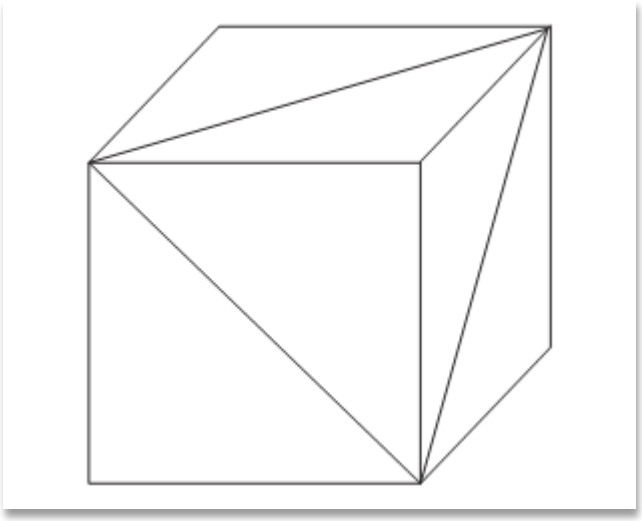
VertexPositionColorTexture	Định nghĩa một đỉnh mang thông tin vị trí, màu sắc và các hệ tọa độ texture.
VertexPositionNormalTexture	Định nghĩa một đỉnh mang thông tin vị trí, màu sắc, các tọa độ texture và vector pháp tuyến.

Ngoài vị trí và các thông tin phụ ra, khi tạo một đối tượng 3-D ta cũng cần phải xác định cách mà XNA sẽ kết nối các đỉnh này lại tùy theo các hình cơ bản khác nhau.

Để vẽ các hình cơ bản hoặc các đối tượng 3-D cơ bản, ta phải định nghĩa cho XNA biết danh sách các đỉnh (XNA gọi là VertexBuffer) sẽ được vẽ khi các chức năng vẽ được gọi. Các đỉnh này có thể được vẽ như là một tập hợp cá điểm rời rạc hoặc dưới dạng nhiều đường thẳng hoặc là dạng nhiều tam giác.

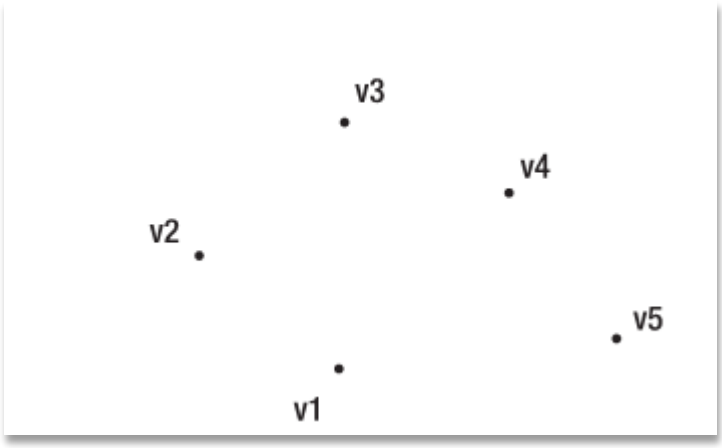
Tam giác được xem là cơ sở để tạo ra bất kỳ các đối tượng 2-D hoặc 3-D nào. Đó là do một hình cơ bản (primitive) được định nghĩa chỉ với ba điểm thì được đảm bảo rằng đó là một mặt đơn (single plane), và mang tính chất lồi (convex, một đoạn thẳng nối bất kỳ hai điểm nào trong một tam giác thì hoàn toàn nằm trong tam giác đó, điều này không xảy ra đối với một vài hình có 4 đỉnh). Những đặc tính này rất quan trọng để thực hiện việc vẽ (rendering) nhanh hơn nếu có thể bằng card đồ họa (card đồ họa luôn xem tam giác như là đối tượng vẽ cơ bản nhất).

Ví dụ: Nếu muốn vẽ một hình vuông lên màn hình, ta sẽ phải dùng 2 tam giác. Nếu muốn tạo một hình khối, ta sẽ dùng 12 tam giác (6 mặt x 2 tam giác).



Hình 3.5

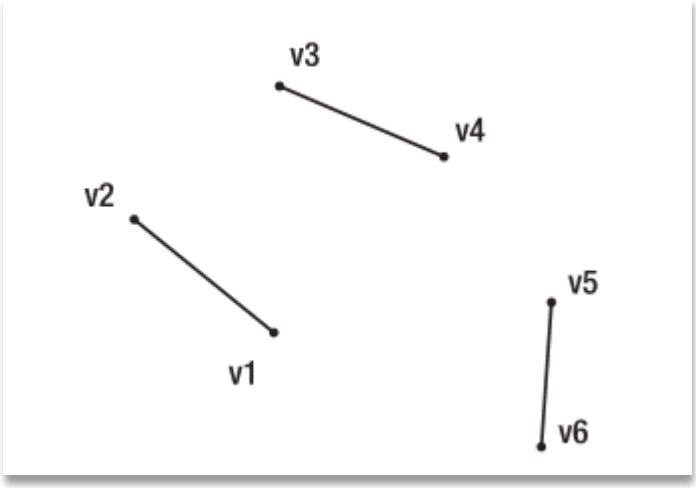
Trong XNA, đối tượng *GraphicsDevice* có một phương thức tên *DrawPrimitives*. Phương thức này dùng để vẽ một vertex buffer tùy vào kiểu hình chỉ ra, được định nghĩa bởi kiểu liệt kê PrimitiveType, như sau:



Hình 3.6

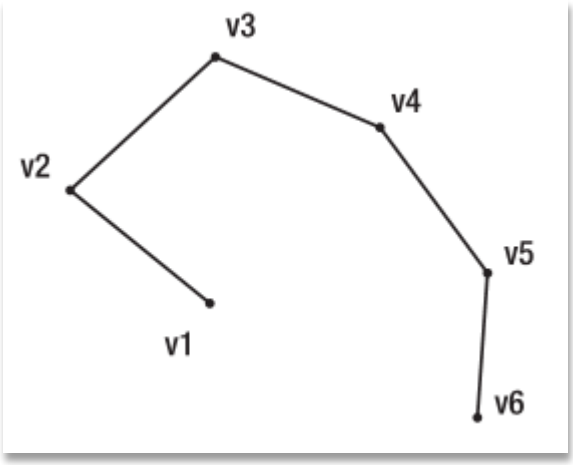
- *PointList:* mỗi đỉnh được vẽ độc lập với những đỉnh khác. Vì thế ta có thể thấy một danh sách các điểm trôi nổi (floating points).

- *LineList:* các đỉnh được vẽ theo cặp (pairs), với các đoạn thẳng nối với mỗi cặp khác. Việc gọi phương thức này thất bại nếu ta thất bại khi truyền vào một vertex buffer với một số chẵn các đỉnh.



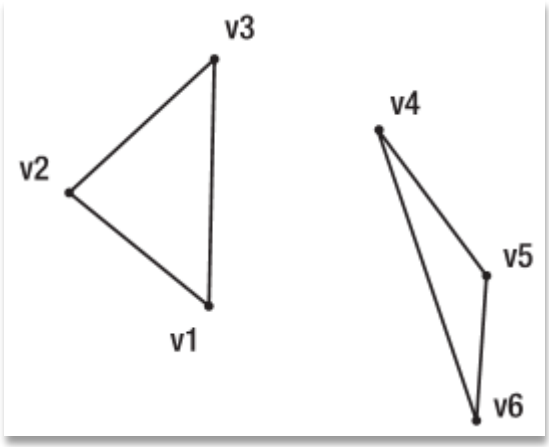
Hình 3.6

- *LineStrip:* tất cả các đỉnh trong vùng đệm (buffer) được vẽ thành một đường thẳng đơn và nối nhau. Nó rất hữu ích khi debug, bởi vì loại hình cơ bản này cho phép ta thấy được một ảnh khung sườn (wireframe image) của đối tượng mà không quan tâm đến số đỉnh.



Hình 3.7

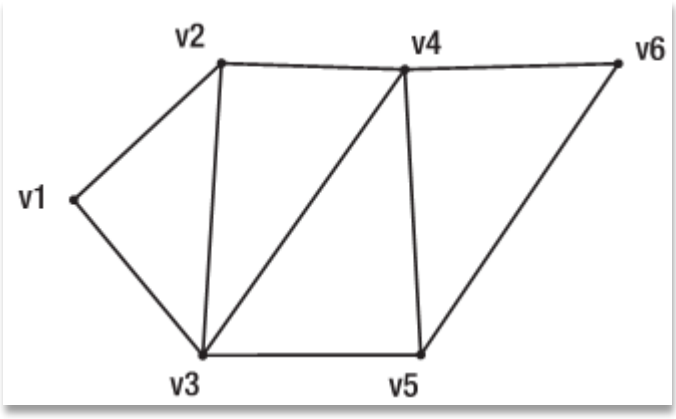
- *TriangleList:* các đỉnh được vẽ theo nhóm 3, thành nhiều tam giác độc lập. Nó tạo ra độ linh hoạt cao nhất khi vẽ các cảnh (scene) phức tạp, nhưng có một hạn chế khi có nhiều đỉnh trùng nhau nếu ta muốn vẽ các tam giác nối nhau.



Hình 3.8

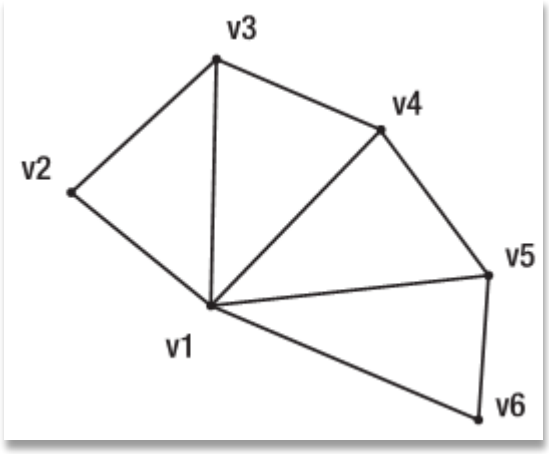
- *TriangleStrip:* ta dùng loại này khi vẽ các tam giác nối tiếp nhau. Kiểu này khá hiệu quả để vẽ cảnh, bởi vì ta không cần phải lặp lại các đỉnh

trùng nhau. Mỗi đỉnh mới (sau hai đỉnh đầu tiên) được thêm vào vùng đệm tạo nên một tam giác mới sử dụng hai đỉnh trước đó.



Hình 3.9

- *TriangleFan:* với loại hình này, tất cả các tam giác có một đỉnh chung, đỉnh đầu tiên của buffer, và mỗi đỉnh mới được thêm vào sẽ tạo thành một tam giác mới dùng đỉnh đầu tiên và đỉnh trước đó.



Hình 3.10

3.4.4Vector, ma trận và các phép biến đổi 3D

3.4.4.1Vector:

Vector: ngoài việc lưu trữ các giá trị vị trí, nó còn cung cấp nhiều phương thức hỗ trợ (sẽ có ích vào một lúc nào đó khi tao tạo trò chơi). Vector3 là vector được dùng phổ biến nhất trong trò chơi 3-D. Sau đây là một số phương thức quan trọng nhất của Vector3:

- Vector3.Distance: nhận vào hai điểm để tính toán và cho ra khoảng cách giữa chúng.
- Vector3.Add và Vector3.Subtract: cộng và trừ hai vector
- Vector3.Multiply và Vector3.Divide: nhân và chia hai vector, hoặc một vector với một số.
- Vector3.Clamp: ràng buộc các thành phần của vector theo một vùng giá trị cho trước (rất hữu ích khi định nghĩa các giá trị của ma trận hoặc độ sáng).
- Vector3.Lerp: thực hiện nội suy tuyến tính giữa hai vector
- Vector3.SmoothStep: nội suy hai vector tùy theo giá trị cho trước đóng vai trò làm trọng số.
- Ngoài các phương thức này, Vector3 cũng cung cấp một loạt các định nghĩa các vector đặc biệt như Vector3.Zero (0, 0, 0), Vector3.Up (0,1,0), Vector3.Right (1,0,0) và nhiều vector khác nữa.
- Vector2 và Vector4 cũng có một số phương thức và định nghĩa tương tự.

3.4.4.2Ma trận:

Ma trận (Matrix): là cơ sở cho việc định nghĩa phép quay (rotation), tỉ lệ (scaling) và tịnh tiến (translation) của một đối tượng trong thế gới 3-D. Bởi vì các ma trận được dùng để định nghĩa bất kỳ phép biến đổi 3-D nào nên chúng cũng được dùng để định nghĩa các phép toán cần cho việc giả lập các phép chiếu và biến đổi cảnh 3-D tùy theo vị trí của camera và hướng đối diện.

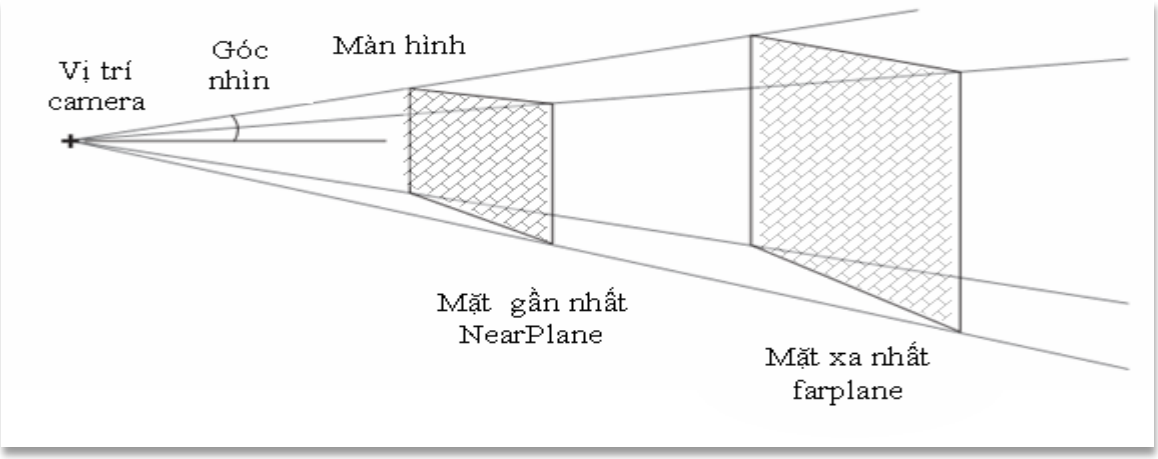
Một trong những ích lợi lớn nhất đó là ta có thể thực hiện các phép tính phức tạp bằng cách nhân các ma trận biến đổi tương ứng của chúng lại với nhau. Sau đó, ta có thể áp dụng ma trận kết quả cho mỗi đỉnh của mô hình 3D, vì thế ta có thể thực hiện tất cả các thao tác trên mô hình bằng việc nhân các đỉnh của nó cho một ma trận, thay vì phải tính toán mỗi phép biến đổi cho mỗi đỉnh.

Nếu ta xem rằng các đối tượng 3D phức tạp có hàng ngàn đỉnh, thì việc thực hiện các phép biến đổi (với một chi phí xử lý thấp nếu có thể) là cần thiết, và ma trận là cách để thực hiện việc này.

3.4.4.3Các phép biến đổi trong 3D:

Trong XNA thông qua lớp *Matrix*, nhiều phép toán trên ma trận đã được cung cấp sẵn, ta không cần hiểu tất cả các chi tiết toán học khi sử dụng ma trận và khi thực thi các phép biến đổi 3D trong ứng dụng của mình.

- *Matrix.CreateRotationX,Matrix.CreateRotationY* và *Matrix.CreateRotationZ*: dùng để tạo ma trận quay cho mỗi trục
- *Matrix.Translation*: tạo một ma trận tịnh tiến theo một hoặc nhiều trục.
- *Matrix.Scale*: tạo 1 ma trận tỉ lệ theo một hoặc nhiều trục.
- *Matrix.CreateLookAt*: tạo một ma trận nhìn (view matrix) dùng để định vị camera, bằng cách thiết lập vị trí 3-D của camera, và hướng ở trên cho camera.
- *Matrix.CreatePerspectiveFieldOfView*: tạo ra một ma trận chiếu, dùng một khung nhìn phối cảnh (perspective view), và các mặt gần và xa (farplane và nearplane: 2 mặt này sẽ giới hạn phần khung cảnh 3-D sẽ được vẽ) xem ảnh minh họa sau.



Hình 3.11

- *Matrix.CreateOrthographic*: tạo một ma trận dùng trong phép chiếu trực giao. Phương thức này nhận vào độ rộng, độ cao, mặt gần, mặt xa để tạo thành một ma trận chiếu trực giao.

3.4.5 Hiệu ứng ánh sáng, Camera:

XNA đã cung cấp các lớp hiệu ứng giúp ta xử lý hiệu ứng được dễ dàng hơn. Điển hình là lớp *BasicEffect*. Lớp *BasicEffect* đáp ứng tất cả các yêu cầu không những cho trò chơi đơn giản mà còn cho các trò chơi phức tạp. Lớp này cung cấp nhiều thuộc tính và phương thức giúp ta định nghĩa các thuộc tính để vẽ cảnh 3D. Sau đây là một số thuộc tính quan trọng của lớp *BasicEffect*:

- *View*: là ma trận nhìn (view matrix) dùng để định nghĩa vị trí đặt và hướng của camera. Thường được tạo ra bằng phương thức *Matrix.CreateLookAt.*
- *Projection*: ma trận chiếu (projection matrix), được dùng để ánh xạ hệ tọa độ cảnh 3D thành hệ tọa độ màn hình. Thường dùng phương thức *Matrix.CreatePerspective, Matrix.CreateOrthographic* hoặc các phương thức tương tự.
- *World*: ma trận thế giới thực (world matrix), được dùng để áp dụng các phép biến đổi cho tất cả các các đối tượng trong cảnh 3D.

- *LightingEnabled*: nếu là false, thì cảnh được vẽ bằng cách dùng một ánh sáng cơ sở để chiếu sáng tới tất cả các mặt của các đối tượng. Nếu là true thì thuộc tính ánh sáng của *BasicEffect* sẽ được dùng để chiếu sáng cảnh.
- *EnableDefaultLighting()*: phương thức này bật hiệu ứng ánh sáng đơn trắng (simple white light) chiếu hướng vào đối tượng mà không cần thêm bất kỳ cấu hình ánh sáng nào.
- *DirectionalLight0, DirectionalLight1*và *DirectionalLight2*: định nghĩa ba loại ánh sáng định hướng (directional light) được sử dụng bởi hiệu ứng khi vẽ. Mỗi ánh sáng được định nghĩa bởi màu sắc phản chiếu (specular color) của nó (màu sắc của ánh sáng sẽ tương phản hoàn hảo, giống như gương), màu khuếch tán (diffuse color, màu sắc của ánh sáng sẽ được phản xạ một cách khuếch tán), và hướng chiếu sáng (light direction). Những thuộc tính này chỉ được dùng khi thuộc tính *LightingEnabled* là *true*.
- *FogColor, FogStart* và *FogEnd*: giúp ta định nghĩa "làn sương mù" (fog) cho khung cảnh. Vì thế các đối tượng trong vùng sương mù khi xuất hiện sẽ được nhìn thấy thông qua một làn khói dày đặc (dense smoke). Ta có thể chỉ định màu sắc của làn sương, cùng với khoảng cách bắt đầu và kết thúc cho làn sương đó.

Cùng với những thuộc tính này, lớp BasicEffect cũng cung cấp tính năng giúp ta vẽ các cảnh 3D một cách chính xác. Ví dụ, đoạn mã sau là một bản thảo cho những gì mà chương trình của ta cần làm để vẽ cảnh 3D một cách chính xác.

```
// phải được khai báo và khởi tạo một cách hợp lý
BasicEffect bfx;
bfx.Begin();
foreach(EffectPass pas in fx.CurrentTechnique.Passes)
```

```
  {
  pass.Begin();
  // code để vẽ cảnh 3-D dùng hiệu ứng này
pass.End();
  }
  bfx.End();
```

Giải thích:

- Ta thông báo cho hiệu ứng bắt đầu việc xử lý
- Sau đó, là một vòng lặp duyệt qua tất cả các EffectPass của kỹ thuật hiện tại (CurrentTechnique) đang dùng.
- Ta cũng cần bắt đầu và kết thúc cho mỗi pass.
- Cuối cùng, báo cho hiệu ứng biết rằng đã kết thúc xử lý.

3.4.6 Mô hình (Models) và mảnh (Meshes):

Model là một đối tượng 3D. Một mô hình đơn giản là một cây phân cấp của các mảnh. Một mảnh có thể được vẽ độc lập. Một mảnh là tập hợp của nhiều đỉnh có quan hệ với nhau, cùng với một vài thông tin vẽ. XNA cung cấp một số lớp đặc biệt để thao tác với các mô hình và mảnh. Đó là lớp *Model* và lớp *ModelMesh*.

Để tạo chương trình có thao tác với các mô hình, trước hết ta phải tải (load) mô hình vào chương trình. Các tập tin định nghĩa mô hình thường có định dạng là (*.X) hoặc (*.FBX).

```
Model myModel;
```

Trong hàm LoadContent, ta tiến hành nạp mô hình như sau:

```
myModel = Content.Load<Model>("ModelName");
```

Cuối cùng, ta đặt một vòng lặp trong hàm *Draw* để duyệt hết các mảnh của mô hình. Đối với mỗi mảnh, ta cũng lặp để duyệt tất cả các hiệu ứng rồi áp dụng hiệu ứng cho mảnh đó. Giả sử đã có các ma trận *matView, matWorld, matProjection*, đoạn mã để vẽ mô hình có dạng như sau:

```
//lặp để vẽ tất cả các mesh của model
foreach (ModelMesh mesh in myModel.Meshes)
{
// lặp để áp dụng tất cả các hiệu ứng của mesh hiện tại
foreach (BasicEffect bfx in mesh.Effects)
 {
 bfx.EnableDefaultLighting();
//thiết lập các ma trận cho hiệu ứng.
 bfx.World = matWorld;
 bfx.View = matView;
 bx.Projection = matProjection;
 }
// tiến hành vẽ mesh
 mesh.Draw();
}
```

Chúng ta không cần chú ý đến việc nạp các *texture* nếu mô hình có sử dụng chúng các tập tin mô hình đã bao gồm sẵn các thông tin về các texture chúng sẽ dùng. Bởi vì thông tin này bao gồm đường dẫn của texture, ta chỉ cần biết đường dẫn này rồi chép các tập tin vào đường dẫn tương ứng. Ta có thể dễ dàng tìm ra các đường dẫn đó bằng cách đọc nội dung của tập tin (bằng chương trình soạn thảo văn bản chẳng hạn), hoặc bằng cách gộp mô hình trong dự án và biên dịch nó. XNA sẽ luôn thông báo lỗi khi không tìm được texture tương ứng cho mô hình. Lỗi này chính là thông báo tập tin texture không tìm thấy tại đường dẫn nào đó. Lúc này ta chỉ cần chép tập tin *texture* vào đúng theo đường dẫn đó là xong.

3.4.7 Độ tạo bóng (Shader)

XNA sử dụng bộ tạo bóng (shader) để chuyển đổi dữ liệu đỉnh (vertex) sang điểm ảnh (pixel). Phương pháp này sẽ giúp thể hiện đẹp hơn bởi vì shader xử lý đồ họa trên card đồ họa. Shader cung cấp khả năng để tạo các đỉnh hiển thị theo ý muốn. Shader được dùng để thao tác với tất cả thuộc tính của đỉnh như màu sắc, vị trí, bề mặt… Ngoài ra, shader cho phép thực hiện việc tạo hiệu ứng ánh sáng, pha trộn hiệu ứng như chế độ trong suốt, đa bề mặt...

- ***Vertex shader:*** Là một phần của shader biểu diễn hoạt động trên mỗi đỉnh nhận từ mã nguồn XNA. Chúng ta có thể dùng vertex shader để chỉnh sửa và vẽ vài thuộc tính của đỉnh. Loại shader này áp dụng các phép biến đổi cho dữ liệu đầu vào dạng đỉnh. Khi các đỉnh đã biến đổi được truyền qua cho shader, dữ liệu đầu ra (sẽ không được nhìn thấy đối với người xem) sẽ được cắt xén và các mặt sau được loại bỏ. Thao tác này gọi là Chọn lọc (*Culling*). Quá trình quét được thực hiện để chuyển đổi dữ liệu của vector sang dạng ảnh đầu ra. Việc nội suy cũng được thực thi giữa các đỉnh để phân phối đồng đều dữ liệu của đỉnh giữa các toạ độ. Trong bộ tạo bóng

điểm ảnh, việc tô màu và tạo bề mặt được áp dụng trước khi xuất các điểm ảnh lên màn hình.

- ***Pixel shaders:*** Chuyển dữ liệu đỉnh từ vertex shader sang dữ liệu điểm ảnh có màu. Pixel shader không thể chỉnh sửa vị trí hoặc thông tin vector pháp tuyến nhưng nó có thể biểu diễn hoạt động trên mỗi điểm ảnh để hiện thực hiệu ứng ánh sáng, màu sắc, bề mặt, pha trộn.

XNA hỗ trợ các ngôn ngữ lập trình shader High Level Shading Language (HLSL) thông qua HLSL của microsoft. HLSL có một vài hàm xây dựng sẵn, nó sẽ bao gồm các phép tính toán học, các truy suất texture và điều khiển luồng. Các kiểu dữ liệu mà HLSL hỗ trợ tương tự với C, ngoại trừ các vector, ma trận (matrix) và bộ lấy mẫu (samplers).

3.4.8 Các bước tạo và vẽ một đối tượng 3D trong XNA

3.4.8.1 Load một Model:

Trước tên ta cần add một model vào dự án Content của game bằng cách add vào "MyGameContent".

Ta load model vào game bằng cách sử dụng lớp "ContentManager", trước tiên ta cần khai báo một model và một mảng ma trận biểu diễn cho sự biến đổi của từng mesh bên trong model.

```
Model Model;
Matrix[] Transforms;
```

Hoàn thiện phương thức LoadContent();

3.5.1 High Level Shading Language (HLSL)

```
Model = Content.Load<Model>("ship");

Transforms = new Matrix[model.Bones.Count];

Model.CopyAbcoluteBoneTransformsTo(transform);
```

3.5.1.1 Giới thiệu:

XNA hỗ trợ ngôn ngữ lập trình shader thông qua HLSL của Microsoft. HLSL có một số hàm xây dựng sẵn, nó bao gồm các phép tính toán học, các truy suất texture và điều khiển luồng. Các kiểu dữ liệu mà HLSL hỗ trợ tương tự với C nhưng ngoại trừ các vector, matrix và sampler.

3.5.1.2 Kiểu dữ liệu

HLSL hỗ trợ nhiều kiểu dữ liệu khác nhau, bao gồm kiểu vô hướng, vector và ma trận.

Bảng 3.2

Type	**Values**
bool	True hoặc false
int	Kiểu int 32 bit
half	Kiểu float 16 bit
float	Kiểu float 32 bit
double	Kiểu float 64 bit

Một kiểu dữ liệu khác trong HLSL là kiểu sampler. Nó dùng để lấy mẫu dữ liệu từ texture. Nhiều loạisampler khác nhau (VD: sampler1D, sampler2D

và sampler3D) được dùng để lấy mẫu các texture 1D, 2D và 3D. Cùng với các kiểu sampler có thêm một vài công đoạn, để xác định chính xác vùng texture sẽ đc lấy mẫu, các bộ lọc có thể sử dụng.

3.5.1.3 Input cố định và Input thay đổi

Kiểu input cố định: kiểu dữ liệu này là dạng hằng số cho tất cả các vertex cũng như pixel của shader trong khi đang xử lý toàn bộ dữ liệu vào. Ví dụ như, khi tạo hình cái cây, texture của nó, ma trận world vàcác điều kiện ánh sáng là hằng số. dữ liệu kiểu này sẽ được thiết lập bên trong ứng dụng XNA.

Kiểu input thay đổi: là các dữ liệu sẽ thay đổi do bị shader biến đổi thông qua quá trình chạy shader. Như khi tạo hình cái cây, vertex shader cần xử lý toàn bộ các vertex của cây, nghĩa là những thông tin về vertex trên cái cây sẽ bị biến đổi bởi mỗi chu kỳ xử lý của vertex shader.

3.5.1.4 Ngữ nghĩa (Semantics)

Semantic là từ đã được định nghĩa từ trước (của HLSL), nó dùng để sắp xếp các dữ liệu input và outputvào các biến trong code HLSL. Ví dụ, mỗi vertex của một đối tượng 3D có thể chứa 1 biến float4 chứa một vị trí trong không gian 3D, một biến float4 khác chứa 2 tọa độ lấy mẫu texture. Làm sao HLSL có thể biết đc cái nào trong số chúng đc sử dụng như là một vị trí trong không gian. Có một giải pháp là ta thêm semantic POSITION0 Vào trong quá trình xử lý vertex để chỉ ra các thuộc tính vị trí của mỗi vertex trên các kiểu dữ liệu thay đổi. Các semantic bắt buộc phải có trong mọi giá trị input thay đổi.

Một số Semantic của Vertex Shader:

- Input Vertex Shader Semantic

Bảng 3.3

Input	**Description**	**Type**
BINORMAL[n]	Binormal	Float4
BLENDINDICES[n]	Blend indices	Uint
BLENDWEIGHT[n]	Blend weights	Float
COLOR[n]	Diffuse and specular color	Float4
NORMAL[n]	Normal vector	Float4
POSITION[n]	Vertex position in object space	Float4
POSITION	Transformed vertex position	Float4
PSIZE[n]	Point size	Float
TANGENT[n]	Tangent	Float4
TEXCOORD[n]	Texture coordinates	Float4

- Output Vertex Shader Semantic

Bảng 3.4

Output	**Description**	**Type**
COLOR[n]	Diffuse or specular color	Float4
FOR	Vertex fog	Float
POSITION[n]	Position of a vertex in homogenous space. Compute position in screen-space by dividing (x,y,z) by w. Every	Float4

	vertex shader must write out a parameter with this semantic.	
PSIZE	Point size	Float
TESSFACTOR[n]	Tessellation factor	Float
TEXCOORD[n]	Texture coordinates	Float4

- Pixel Shader Semantic

Bảng 3.5

Input	**Description**	**Type**
COLOR[n]	Diffuse or specular color	Float4
TEXCOORD[n]	Texture coordinates	Float4
VFACE	Floating-point scalar that indicates a back-facing primitive. A negative value faces backwards, while a positive value faces the camera.	Float
VPOS	The pixel location (x,y) in screen space.	Float2

Bảng 3.6

Output	**Description**	**Type**
COLOR[n]	Output color	Float4

DEPTH	Output depth	Float

3.5.1.5 Hàm (Functions)

HLSL cho phép tạo ra nhiều hàm với cú pháp tương tự C. Mỗi hàm cần có phần khai báo và phần thân. Khai báo hàm chứa tên hàm và kiểu trả về, có thể có một list tham số. Và kiểu trả về của hàm có thể có cả semantic có liên quan với nó.

Dưới đây là một hàm được dùng như là hàm bắt đầu thực hiện công việc của pixel shader:

```
float4 simplePS(float4 inputColor: COLOR0): COLOR0
{
return inputColor * 0.5f;
}
```

3.5.1.6 Techniques, Passes, và Effects

Technique là sự kết hợp của vertex shader và pixel shader. Dưới đây là một technique sử dụng vertex shader và pixel shader:

```
technique basicTechnique
{
 pass p0
 {
 VertexShader = compile vs_2_0 SimpleVS();
```

```
 PixelShader = compile ps_2_0 SimplePS();
 }
}
```

Technique được định nghĩa để xác định phiên bản shader (shader model) cần được biên dịch. Ở đây dùng shader model 2.0 cho cảhai shader. Phiên bản cao hơn cho phép sử dụng shader phức tạp hơn và có nhiều hàm hơn, nhưng yêu cầu GPU phải hỗ trợ chúng.

GPU (Graphic Processor Unit) là đơn vị xử lý đồ họa, độ mạnh của GPU tùy thuộc vào card đồ họa.

Có thể thấy, cả hai shader đều được gói lại trong pass. Một pass sẽ nhận toàn bộ vetrex cần được draw kết hợp với technique, pass xử lý trên từng pixel, và tạo hình nhưng pixel này và đưa nó vào vùng đệm, vùng đệm là nơi mà các pixel sẽ chờ để được draw lên màn hình. Để xử lý tất cả vertex trong frame hai lần technique sẽ có hai pass, mỗi pass đều có vertex shader và pixel shader riêng của nó.

Khi sử dụng nhiều technique, technique đầu tiên dùng để biến đổi không gian 3D sang dạng 2D, và một technique khác sẽ dùng kết quả của technique đầu tiên như là input của nó.

Sự kết hợp của các shader và technique gọi là hiệu ứng. Một hiệu ứng đơn giản sẽ chứa vertex shader, pixel shader và technique. Những hiệu ứng nâng cao sẽ chứa rất nhiều vertex shader, pixel shader và technique.

Tất cả shader và technique của một hiệu ứng được giữ trong một file duy nhất. XNA gọi file mã HLSL là file hiệu ứng (*.fx). HLSL cho phép XNA tùy chỉnh các hiệu ứng như các tài nguyên của game, cũng giống như model. Tất cả hiệu ứng sẽ được xử lý thông qua trình xử lý tài nguyên của XNA. HLSL

là những đối tượng đã được quản lý sẵn để ContentManager có thể load trong quá trình chạy game.

3.5.1.7 Effect Class

Trong XNA, hiệu ứng cần được load trong đối tượng lớp Effect (Giống một bức ảnh load trong đối tượng texture2D) lớp hiệu ứng cho phép cấu hình các tham số cố định của HLSL. Lựa chọn technique để sử dụng, và dùng effect để tạo hình.

Đoạn mã sau minh họa việc load và cấu hình hiệu ứng với XNA:

```
// XNA Effect object
Effect effect;
// Load the effect
effect = content.Load<Effect>("/effects/simpleEffect");
// Set the technique
effect.CurrentTechnique = lightEffect.Techniques["basicTechnique"];
// Configure uniform effect parameters – Đưa giá trị ma trận cho tham số của hiệu ứng
```

```
effect.Parameters["matWVP "].SetValue(worldViewProjectionMatrix);
```

Đoạn mã trên thiết lập việc load một hiệu ứng từ mã HLSL sử dụng phương thức LoadContent() của ContentManager. Sau đó xác định technique sẽ sử dụng, ở đây technique chính là basicTechnique đã định nghĩa trước đó. Cuối cùng đưa giá trị ma trận cho tham số cố định của hiệu ứng: matWVP.

Đoạn mã sau cho thấy cách draw đối tượng bằng cách sử dụng file hiệu ứng đã load ở trên:

```
// First begin the effect
effect.Begin();
// Remember that the effect can have many passes
foreach (EffectPass pass in effect.CurrentTechnique.Passes)
{
 pass.Begin();
 // PUT YOUR DRAWING CODE HERE
 pass.End();
}
// Finally, end the effect
effect.End();
```

Để draw một đối tượng 3D, đầu tiên cần begin hiệu ứng muốn dùng, lặp tất cả các pass trong technique đã chọn, Begin pass, draw đối tượng, sau đó End pass vừa Begin. Cuối cùng, phần kết thúc hiệu ứng, pass của hiệu ứng được biểu diễn bởi lớp EffectPass của XNA, có thể lựa chọn technique cần sử dụng bằng thuộc tính CurrentTechnique của lớp Effect. Nếu muốn thay đổi tham số hiệu ứng sau khi Begin pass, cần gọi phương thức CommitChange của lớp hiệu ứng để cập nhật lại những sự thay đổi.

3.5.1.8 Materials Class

Materials là lớp cần tạo để lưu các tham số dùng để cấu hình lên hiệu ứng. Ví dụ, có thể tạo hình 2 bề mặt sử dụng một hiệu ứng chung áp dụng lên mỗi texture trên mỗi bề mặt. Lúc này, chất liệu của mỗi bề mặt là texture. Do đó, hai bề mặt sẽ có cùng 1 lớp chất liệu, mà vẫn tránh được việc phải thay đổi hoàn toàn hiệu ứng đã được được đặt hoặc phải thay đổi tham số của mô hình. Material gồm 2 lớp cơ bản:

- LightMaterial: là lớp lưu các thuộc tính bề mặt dùng cho ánh sáng bao gồm: diffuse color, specular color và specular power.
- TextureMaterial: lớp này lưu trữ cách lấy mẫu texture và tile. Tile là những texture rất nhỏ, được nhân bản ra dùng để draw một bề mặt lớn, để áp dụng texture đó lên một bề mặt nào đó.

Sau đây là đoạn mã cho lớp TextureMaterial

```
public class TextureMaterial
{
 // Texture
 Texture2D texture;
 // Texture UV tile
 Vector2 uvTile;
 // Properties
 public Texture2D Texture
 {
 get { return texture; }
```

```
set { texture = value; }
}
public Vector2 UVTile
{
get { return uvTile; }
set { uvTile = value; }
}
public TextureMaterial(Texture2D texture, Vector2 uvTile)
{
this.texture = texture;
this.uvTile = uvTile;
}
}
```

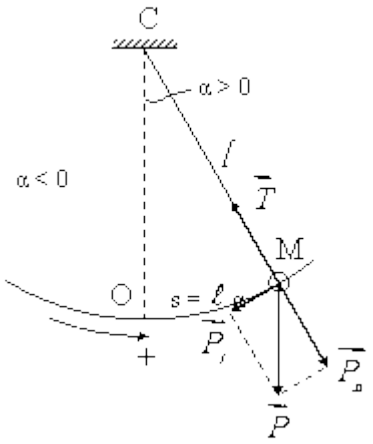
Trong đoạn mã trên trong đoạn mã trên các thuộc tính của lớp TextureMaterial được lưu trữ giống như một Texture 2D trong XNA, còn về Vector2 UVTile được dùng để làm bump Mapping (hiệu ứng cho địa hình).

CHƯƠNG 4: KIẾN THỨC VẬT LÝ

4.1 Chuyển động của con lắc đơn:

Cấu tạo

- Gồm một sợi dây không giãn có độ dài l, khối lượng không đáng kể, một đầu cố định, đầu còn lại được gắng vào một vật có khối lượng m. Con lắc dao động với biên độ góc nhỏ (α < 10).
- Điều kiện dao động điều hoà:Bỏ qua ma sát, lực cản và $\alpha_0 \ll 10^0$hay $S_0 \ll l$



Hình 4.1

Các đại lượng đặc trưng:

- Phương trình dao động:
  - PT li độ cung**:** s = $s_0\cos(\omega t + \varphi)$
  - PT li độ góc: $a = a_0\cos(wt + j)$ Với s = a .l ; $s_0 = a_0$.l
  - PT vận tốc: v = s' = $-\omega s_0 \sin(\omega t + \varphi) = - wl a_0\sin(wt + j)$
  - PT gia tốc: a = s" = $-w^2 s_0 \cos(\omega t + \varphi) = - w^2 l a_0\cos(wt + j) = - w^2 s = - w^2 a l$

- Tần số góc, chu kỳ và tần số: $\omega = \sqrt{\frac{g}{l}}$; $T = 2\pi\sqrt{\frac{l}{g}}$ và $f = \frac{1}{2\pi}\sqrt{\frac{g}{l}}$.
- Lực kéo về (hồi phục): (xét với dao động với biên độ góc nhỏ)
  - $F = Pt = - mg\sin\alpha \approx - mg\alpha \approx - mgs/l = -m w^2 s$
- Hệ thức độc lập:
  - $a = - w^2 s = - w^2 a1$
  - $S_0 = \sqrt{s^2 + \left(\frac{v}{\omega}\right)^2} = \sqrt{\frac{v^2}{\omega^2} + \frac{a^2}{\omega^4}}$
  - $v^2 = w^2 (s_0^2 - s^2)$
- Cơ năng:
  - Thế năng: $W_t = mgl(1 - \cos\alpha)$
  - Động năng: $W_đ = \frac{1}{2} mv^2 = mgl(\cos\alpha - \cos\alpha_0)$
  - Cơ năng: $W = W_t + W_đ = mgl(1 - \cos\alpha_0)$
  - Nếu $\alpha_0 \leq 10^0$ thì:

    $W_t = \frac{1}{2} mgl\alpha^2$;

    $W_đ = \frac{1}{2} mgl(\alpha_0^2 - \alpha^2)$;

    $W = \frac{1}{2} mgl \alpha_0^2$; α và α_0 tính ra rad.
  - Thế năng và động năng của con lắc đơn biến thiên tuần hoàn với

    $\omega' = 2\omega$;

    $f' = 2f$;

    $T' = \frac{T}{2}$.
- Vận tốc:

- Vận tốc khi đi qua li độ góc α:

$$v = \sqrt{2gl(\cos\alpha - \cos\alpha_0)}.$$

- Vận tốc khi đi qua vị trí cân bằng ($\alpha = 0$):

$$|v| = v_{max} = \sqrt{2gl(1-\cos\alpha_0)}.$$

- Nếu $\alpha_0 \leq 10^0$ thì:

$$v = \sqrt{gl(\alpha_0^2 - \alpha^2)};$$

$v_{max} = \alpha_0\sqrt{gl}$ α, α_0 tính ra rad.

- Sức căng của sợi dây:
  - Sức căng của sợi dây khi đi qua li độ góc α là:

$$T_\alpha = mg\cos\alpha + \frac{mv^2}{l} = mg(3\cos\alpha - 2\cos\alpha_0).$$

  - Tại VTCB là: $T_{VTCB} = T_{max} = mg(3 - 2\cos\alpha_0)$
  - Tại vị trí biên: $T_{biên} = T_{min} = mg\cos\alpha_0$.
  - Với $\alpha_0 \leq 10^0$:

$$T = 1 + \alpha_0^2 - \frac{3}{2}\alpha^2;$$

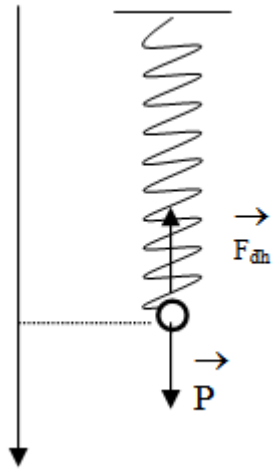
$$T_{max} = mg(1 + \alpha_0^2);$$

$$T_{min} = mg(1 - \frac{\alpha_0^2}{2}).$$

4.2 Chuyển động của con lắc lò xo

Cấu tạo:

- Gồm một lò xo co giãn được giữ cố định một đầu, một đầu được nối với một vật A có khối lượng m, khi tác dụng một lực vào vật A hướng xuống hoặc hường lên vật dao động theo phương thẳng đứng quanh gốc tọa độ.



Hình 4.2

Các đại lượng đặc trưng:

- Phương trình dao động:
  - Phương trình ly độ: $x = A\cos(\omega t + \mu)$
  - Phương trình vận tốc: $v = -A\omega\sin(\omega t + \mu)$
  - Phương trình gia tốc: $a = -A\omega^2\cos(\omega t + \mu)$
- Các đại lượng đặc trưng tương ứng trong chương trình:

  A: biên độ dao động

  ω: tần số góc

  μ: pha ban đầu
  - Chu kỳ dao động: $T = \frac{2\pi}{\omega}$
  - Tần số dao động: $f = \frac{1}{T}$
  - Pha dao động: $\mu = \omega t + \mu_0$
- Năng lượng trong dao động con lắc lò xo:
  - Động năng: $W_d = \frac{1}{2}mv^2 = \frac{1}{2}m\omega^2A^2\sin(\omega t + \varphi)^2$
  - Thế năng: $W_t = \frac{1}{2}kx^2 = \frac{1}{2}kA^2\cos(\omega t + \varphi)^2$

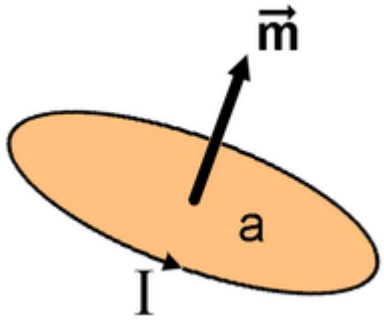
- Cơ năng: $W = W_d + W_t = \frac{1}{2}kA^2 = \frac{1}{2}m\omega^2A^2 = const$

$$W = W_{0d} + W_{0t} = \frac{1}{2}kA^2 = \frac{1}{2}m\omega^2A^2$$

- Mối liên hệ giữa A, ω, v, x: $A^2 = x^2 + \frac{v^2}{\omega^2}$

4.3 Chuyển động momen

Cấu tạo gồm một vật A có khối lượng m, có trục cố định, vật chuyển động tròn quanh trục cố định khi tác dụng một lực F vào điểm cách trục một đoạn R.



Hình 4.3

Một số đại lượng quan trọng trong chuyển động quay momen:

- Momen quán tính:
  - Tổng quát: $I = \sum_i m_i r_i^2$
  - Với vành tròn đồng chất: $I = \frac{1}{2}mr^2$
  - Với thanh thẳng đồng chất: $I = \frac{1}{12}ml^2$
  - Với hình cầu đặc: $I = \frac{2}{5}mr^2$
  - Với hình cầu rỗng: $I = \frac{2}{3}mr^2$
- Momen động lượng:
  - Momen động lượng tỉ lệ thuận với động lượng của vật.

$$\mathbf{L} = \mathbf{r} \times \mathbf{p} = \mathbf{r} \times m\mathbf{v}$$

- Với vật thể rắn:

$$\mathbf{L} = I\boldsymbol{\omega}$$

- Định luật bảo toàn động lượng: momen động lượng của một hệ thống sẽ không đổi khi hệ chịu tổng cộng các momen ngoại lực bằng 0.

▪ Momen lực:

- Xét một vật nằm trong mặt phẳng ngang quanh trục Oz momen lực F quanh trục quay là đại lượng đặc trưng cho tác dụng làm quay của trục.

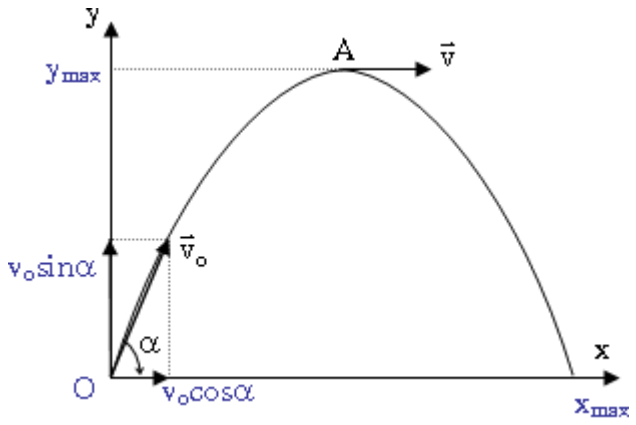
  M = Fd

- Muốn một vật rắn có trục quay cố định thì tổng momen của các lực bằng 0;

  $M_1+M_2+...+M_n=0$;

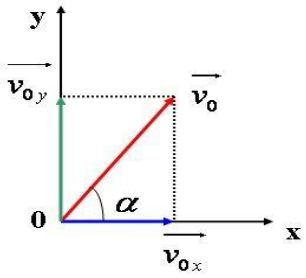
4.4 Chuyển động của vật bị ném xiên

Khảo sát chuyển động của một vật bị ném xiên từ một điểm O trên mặt đất. Sau khi được truyền một vận tốc đầu V_0 làm với mặt phẳng ngang một góc α vật chỉ còn chịu tác dụng của trọng lực (bỏ qua sức cản của không khí).



Hình 4.4

Xét một vật được ném ra theo phương xiên (có vận tốc ban đầu v_0 hợp với phương ngang một góc α). Với dạng quỹ đạo là một parabol, chuyển động ném xiên có thể phân tích thành hai chuyển động là chuyển động theo phương ngang Ox và theo phương thẳng đứng Oy.



Hình 4.5

Chọn hệ trục tọa độ như hình vẽ. Gốc tọa độ O là vị trí ném vật. Gốc thời gian là lúc ném vật ($t_0 = 0$).

Ta xét chuyển động của vật trên trục Ox và Oy.

- Theo OY: ta có các phương trình mô tả chuyển động của vật là:
  - Gia tốc: $a_{y=} - g$
  - Vận tốc: $v_y = v_0.\sin a - g.t$
  - Tọa độ:$y = v_0 \sin(a)\, t - g t^2 \frac{1}{2}$
- Theo OX: ta có các phương trình mô tả chuyển động của vật:
  - Gia tốc: $a_x = 0$
  - Vận tốc: $v_x = V_{0x} = v_0.\cos a = Const$
  - Tọa độ: $x = v_0 \cos a\,.\,t$

Phương trình quỹ đạo của chuyển động ném xiên: $y = \frac{-gX^2}{2v_0^2 cos(a)^2} + \tan a.X$

Độ cao cực đại của vật bị ném: $Y_{max} = -\frac{\Delta}{4a} = \frac{\tan a^2}{4\frac{g}{2v_0^2 cos\, a^2}} = \frac{v_0^2 sin\, a^2}{2g}$

Tầm bay xa của vật bị ném: $D = 2X_0 = 2\frac{v_0^2 \sin a \cos a}{g}$

Vận tốc của vật bị ném xiên:

$$V = \sqrt{V_x^2 + V_y^2} = \sqrt{v_0^2 \cos a^2 + (v_0 \sin a - gt)^2} = \sqrt{g^2 t^2 - 2v_0 g \sin a\, t v_0^2}$$

CHƯƠNG 5: MÔ PHỎNG VẬT LÝ

5.1 Phân tích yêu cầu bài toán

5.1.1 Yêu cầu chung của chương trình:

- Áp dụng kiến thức vật lý theo các chuyên đề đã nêu.
- Sử dụng công cụ XNA vào mô phỏng 2D.

5.1.2 Chức năng chính của chương trình:

Chức năng chung:

- Chức năng thay đổi thông số đầu vào.
- Chức năng Save/Load.
- Chức năng thay đổi tốc độ mô phỏng.

Chức năng riêng theo từng mô phỏng:

- Đối với con lắc đơn:
  - Mô phỏng trong môi trường 2D.
  - Mô phỏng như trong thực tế một con lắc lò xo có một đầu cố định, một đầu gắn vào một quả lắc.
  - Thể hiện các thông số lúc chuyển động.
  - Vẽ đồ thị dao động.
  - Tương tác với quả lắc bằng tay.
- Đối với con lắc lò xo:
  - Mô phỏng trong môi trường 2D.
  - Mô phỏng như trong thực tế một con lắc đơn có một đầu cố định, một đầu gắn vào một quả lắc.
  - Thể hiện các thông số trong lúc chuyển động.
  - Vẽ đồ thị chuyển động.
  - Tương tác với quả lắc bằng cảm ứng tay trên màn hình cảm ứng.
- Đối với chuyển động ném xiên:

- Mô phỏng trong môi trường 2D.
- Cấu tạo gồm một vật ở một độ cao xác định được tác dụng một lực phương của lực tác dụng tạo thành một góc cố định theo phương ngang.
- Thể hiện các thông số trong quá trình chuyển động.
- Vẽ đồ thị chuyển động.

- Đối với Momen:
  - Mô phỏng trong môi trường 2D.
  - Cấu tạo gồm một bàn quay có khối lượng và bán kính ban đầu cùng các vật được dung để đặt lên bàn quay để xem quán tính của từng vật, một vật hãm tốc để giảm dần vận tốc của bàn quay.
  - Thể hiện các thông số trong quá trình chuyển động.
  - Vẽ đồ thị chuyển động.

5.1.3 Mô hình Use Case:

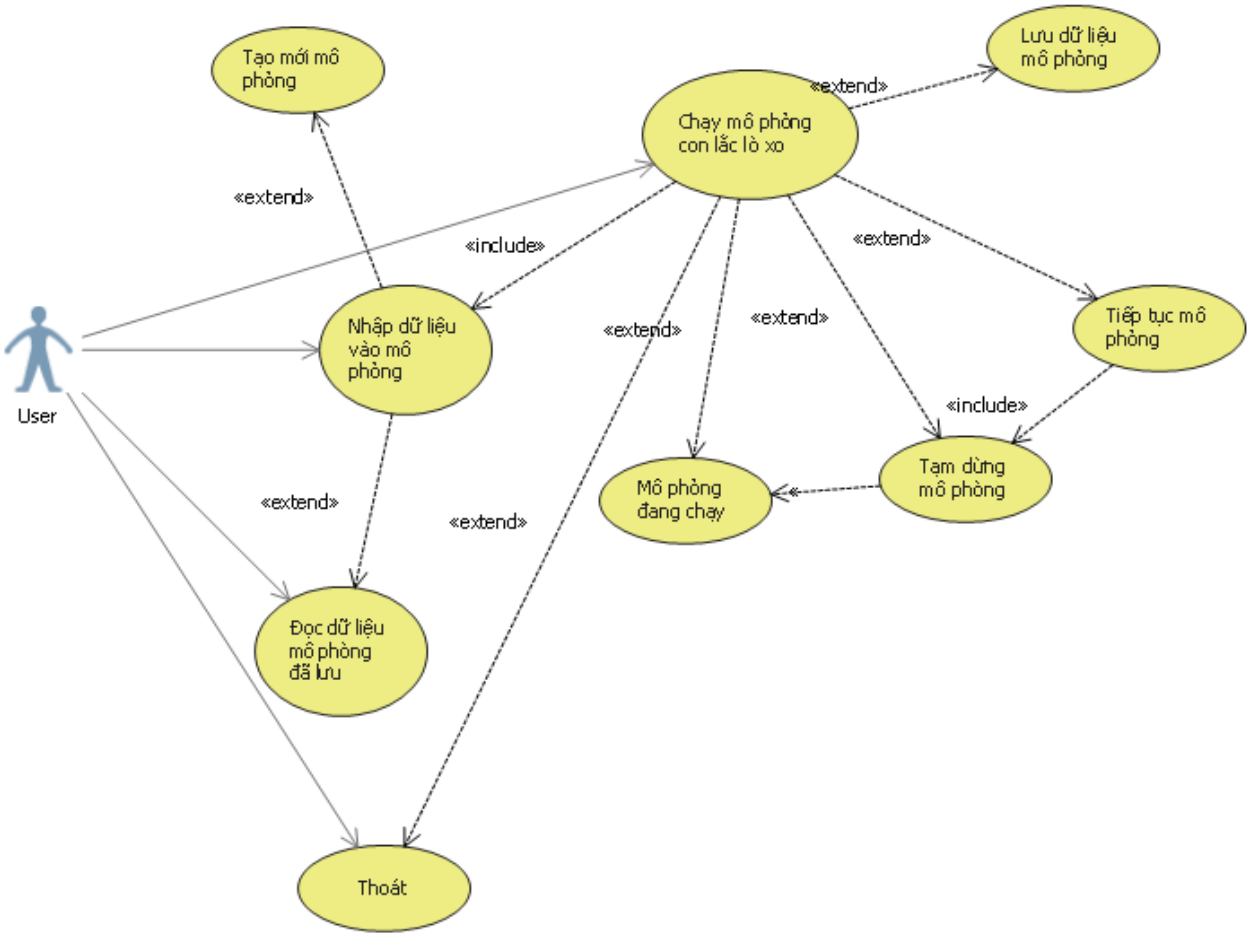
❖ **Mô hình use case cho con lắc lò xo:**

Đặc tả Use case:

Bảng 5.1

Use case ID:	UC_01		
Tên use case:	Mô phỏng con lắc lò xo		
Người tạo:	Trần Thanh Sơn	**Người cập nhật cuối:**	Trần Thanh Sơn
Ngày tạo:	15/10/2013	**Ngày cập nhật cuối:**	15/10/2013
Nhân vật:	User		
Mô tả:	Khi người dùng vào chức năng mô phỏng của con lắc lò xo thì ban đầu sẽ tạo những giá trị mặc định. Người dùng có thể thay đổi các giá trị ban đầu với chức năng nhập thông số hoặc đọc từ danh sách các thông số đã được lưu		

Điều kiện kích hoạt:	Người dùng chọn chức năng mô phỏng con lắc lò xo
Điều kiện đầu:	1. Vào ứng dụng 2. Chọn mô phỏng con lắc lò xo
Điều kiện sau:	1. Trở lại màn hình menu chính.
Tiến trình thường:	1. Chọn mô phỏng con lắc lò xo 2. Chọn "nhập thông số" hoặc "dữ liệu đã lưu" dưới thanh Application bar để thiết lập thông số cho mô phỏng 3. Chọn "Chạy" để chạy mô phỏng 4. Chọn "Dừng" để dừng mô phỏng tạm thời 5. Chọn "Lưu" để lưu thông số mô phỏng lại cho lần sau có thể sử dụng lại mà không cần phải nhập lại. 6. Chọn "Bắt đầu lại" để trở về trạng thái ban đầu khi mô phỏng chưa được chạy
Tiến trình phụ:	1. Bấm nút "Back" để thoát mô phỏng và trở lại màn hình chính
Các ngoại lệ:	
Các use case liên quan:	
Yêu cầu riêng biệt:	
Các giả định:	
Ghi chú và các vấn đề:	1. Các thông số trong quá trình mô phỏng thay đổi theo thời gian thực 2. Độ sai số chỉ khoảng 1/10000



Hình 5.1

Nội dung mô hình:

- User: người sử dụng chương trình.
- Nhập dữ liệu vào mô phỏng: đây là quá trình người dùng nhập những thông số cần thiết cho quá trình chuyển động. Đối với con lắc lò xo thì dữ liệu vào mô phỏng gồm: A (biên độ dao động), X_0 (vị trí ban đầu của lò xo), θ (pha ban đầu), K (độ cứng của lò xo), M (khối lượng con lắc đơn).

- Đọc file có sẵn: người dùng chọn những file có sẵn đã được tạo ra bởi hệ thống. Các file này chính là những file lưu sẵn trước đó của mô phỏng.
- Nhập dữ liệu vào mô phỏng: đây chính là bước tổng hợp từ hai bước: nhập dữ liệu vào mô phỏng và đọc dữ liệu từ file. Bước này sẽ khởi tạo các thông số cho mô phỏng được chạy.
- Lưu quá trình mô phỏng: sau khi thiết lập những thông số cơ bản cho quá trình mô phỏng người dùng có thể lưu lại quá trình mô phỏng.
- Tạm dừng quá trình mô phỏng: khi quá trình mô phỏng đang diễn ra, có thể tạm dừng chương trình để đọc những thông số tại thời điểm đó. Điều kiện cần để dừng quá trình mô phỏng là mô phỏng đang được chạy.
- Tiếp tục quá trình mô phỏng: sau khi đã tạm dừng mô phỏng để đọc những thông số cần thiết người dùng có thể tiếp tục lại quá trình mô phỏng. Điều kiện cần để tiếp tục mô phỏng là mô phỏng đang được dừng lại.
- Thoát: người chơi muốn kết thúc mô phỏng.

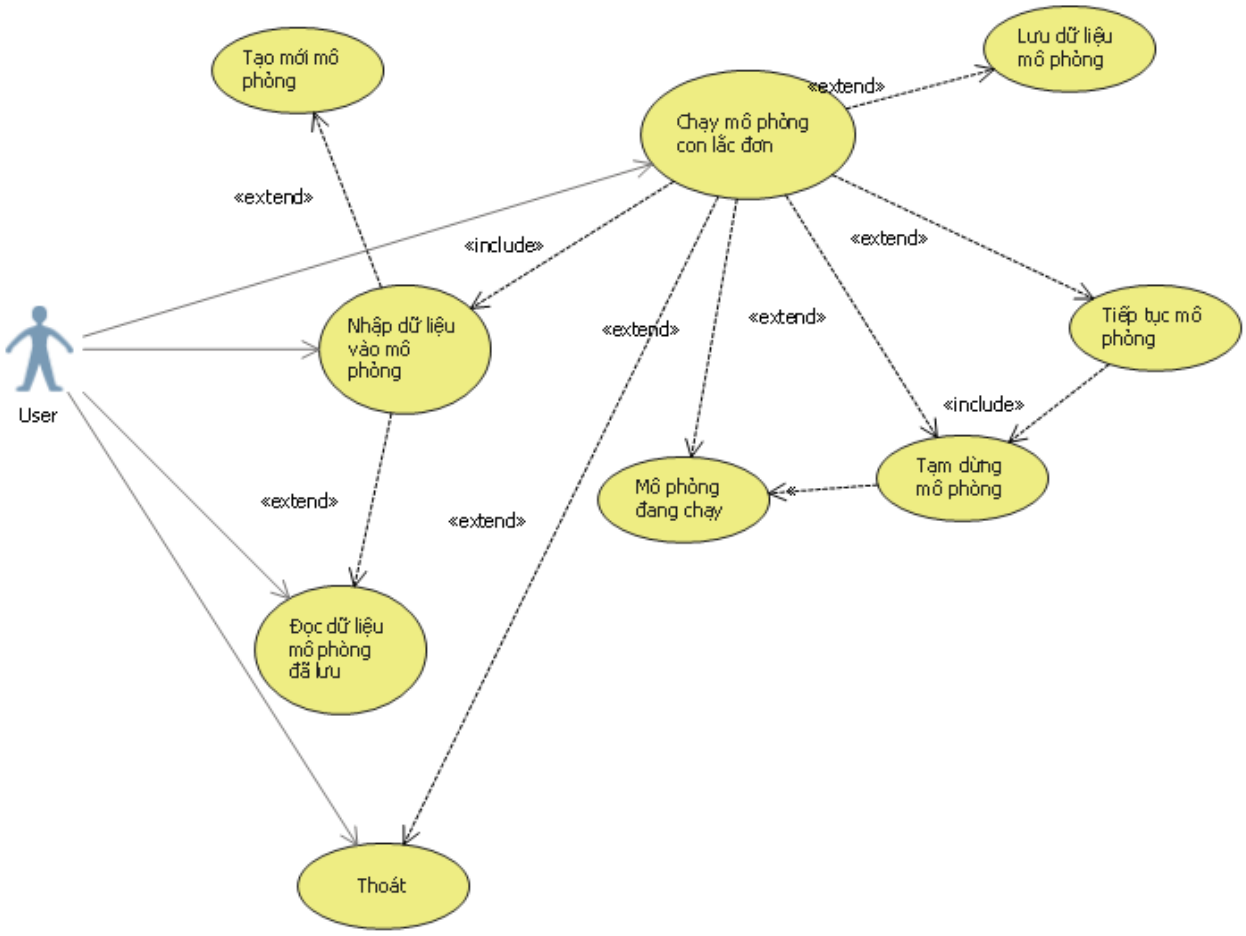
❖ **Mô hình use case cho con lắc đơn:**

Đặc tả Use case:

Bảng 5.2

Use case ID:	UC_02		
Tên use case:	Mô phỏng con lắc đơn		
Người tạo:	Trần Thanh Sơn	**Người cập nhật cuối:**	Trần Thanh Sơn
Ngày tạo:	18/10/2013	**Ngày cập nhật cuối:**	18/10/2013
Nhân vật:	User		
Mô tả:	Khi người dùng vào chức năng mô phỏng của con lắc đơn thì ban đầu sẽ tạo những giá trị mặc định. Người dùng có		

	thể thay đổi các giá trị ban đầu với chức năng nhập thông số hoặc đọc từ danh sách các thông số đã được lưu
Điều kiện kích hoạt:	Người dùng chọn chức năng mô phỏng con lắc đơn
Điều kiện đầu:	3. Vào ứng dụng 4. Chọn mô phỏng con lắc đơn
Điều kiện sau:	2. Trở lại màn hình menu chính.
Tiến trình thường:	7. Chọn mô phỏng con lắc đơn 8. Chọn "nhập thông số" hoặc "dữ liệu đã lưu" dưới thanh Application bar để thiết lập thông số cho mô phỏng 9. Chọn "Chạy" để chạy mô phỏng 10. Chọn "Dừng" để dừng mô phỏng tạm thời 11. Chọn "Lưu" để lưu thông số mô phỏng lại cho lần sau có thể sử dụng lại mà không cần phải nhập lại. 12. Chọn "Bắt đầu lại" để trở về trạng thái ban đầu khi mô phỏng chưa được chạy
Tiến trình phụ:	2. Bấm nút "Back" để thoát mô phỏng và trở lại màn hình chính
Các ngoại lệ:	
Các use case liên quan:	
Yêu cầu riêng biệt:	
Các giả định:	
Ghi chú và các vấn đề:	3. Các thông số trong quá trình mô phỏng thay đổi theo thời gian thực 4. Độ sai số chỉ khoảng 1/10000



Hình 5.2

Nội dung mô hình:

- User: người sử dụng chương trình.
- Nhập dữ liệu vào mô phỏng: đây là quá trình người dùng nhập những thông số cần thiết cho quá trình chuyển động. Đối với con lắc đơn thì dữ liệu mô phỏng bao gồm: A (biên độ dao động), L (chiều dài con lắc đơn), X_0: vị trí của con lắc đơn, θ (pha ban đầu).
- Đọc file có sẵn: người dùng chọn những file có sẵn đã được tạo ra bởi hệ thống. Các file này chính là những file lưu sẵn trước đó của mô phỏng.

- Nhập dữ liệu vào mô phỏng: đây chính là bước tổng hợp từ hai bước: nhập dữ liệu vào mô phỏng và đọc dữ liệu từ file. Bước này sẽ khởi tạo các thông số cho mô phỏng được chạy.
- Lưu quá trình mô phỏng: sau khi thiết lập những thông số cơ bản cho quá trình mô phỏng người dùng có thể lưu lại quá trình mô phỏng.
- Tạm dừng quá trình mô phỏng: khi quá trình mô phỏng đang diễn ra, có thể tạm dừng chương trình để đọc những thông số tại thời điểm đó. Điều kiện cần để dừng quá trình mô phỏng là mô phỏng đang được chạy.
- Tiếp tục quá trình mô phỏng: sau khi đã tạm dừng mô phỏng để đọc những thông số cần thiết người dùng có thể tiếp tục lại quá trình mô phỏng. Điều kiện cần để tiếp tục mô phỏng là mô phỏng đang được dừng lại.
- Thoát: người chơi muốn kết thúc mô phỏng.

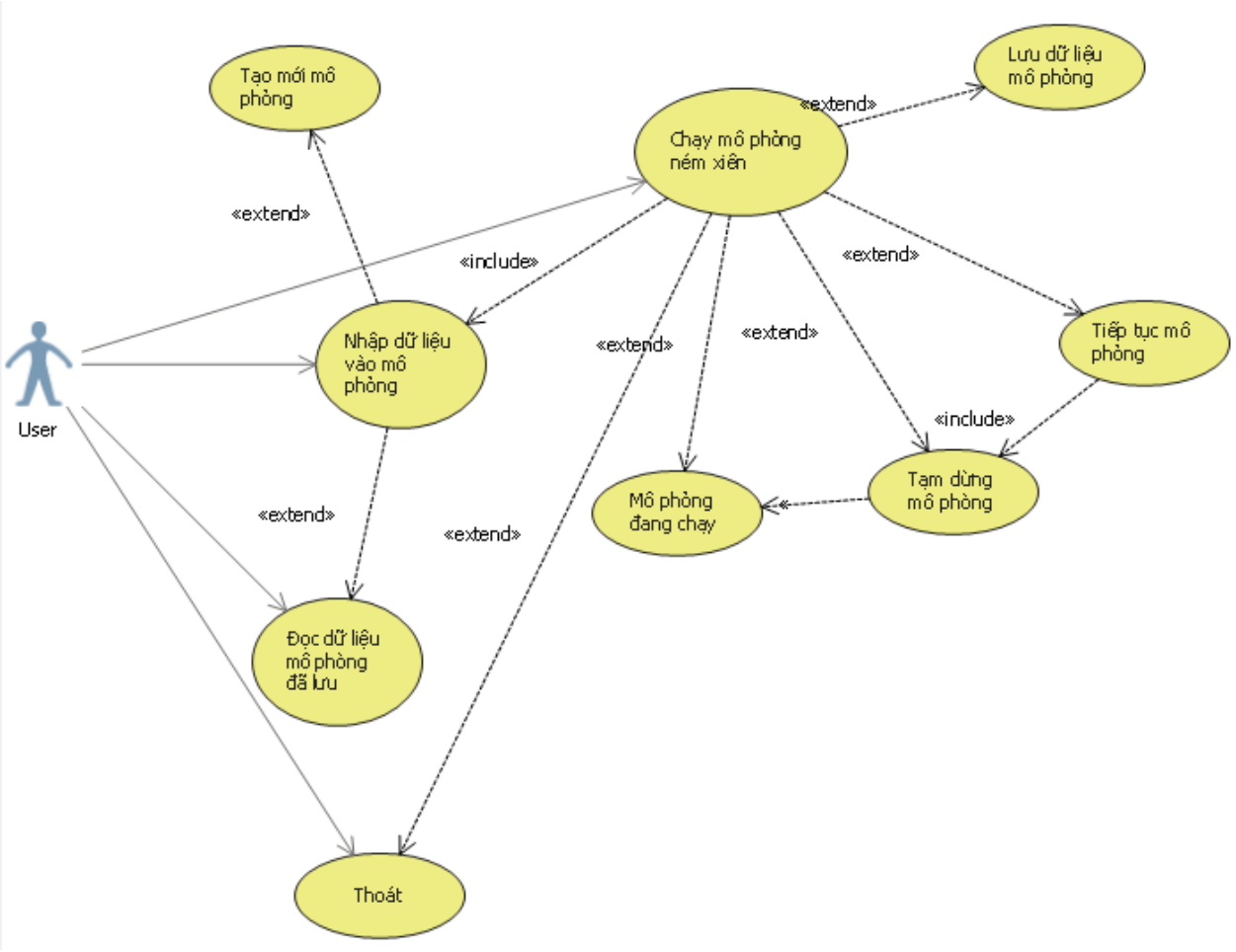
❖ **Mô hình use case cho chuyển động ném xiên:**

Đặc tả Use case:

Bảng 5.3

Use case ID:	UC_02		
Tên use case:	Mô phỏng vật bị ném xiên		
Người tạo:	Trần Thanh Sơn	**Người cập nhật cuối:**	Trần Thanh Sơn
Ngày tạo:	20/10/2013	**Ngày cập nhật cuối:**	20/10/2013
Nhân vật:	User		
Mô tả:	Khi người dùng vào chức năng mô phỏng của vật bị ném xiên thì ban đầu sẽ tạo những giá trị mặc định. Người dùng có thể thay đổi các giá trị ban đầu với chức năng nhập thông số hoặc đọc từ danh sách các thông số đã được lưu		

Điều kiện kích hoạt:	Người dùng chọn chức năng mô phỏng vật bị ném xiên
Điều kiện đầu:	5. Vào ứng dụng 6. Chọn mô phỏng vật bị ném xiên
Điều kiện sau:	3. Trở lại màn hình menu chính.
Tiến trình thường:	13. Chọn mô phỏng vật bị ném xiên 14. Chọn "nhập thông số" hoặc "dữ liệu đã lưu" dưới thanh Application bar để thiết lập thông số cho mô phỏng 15. Chọn "Chạy" để chạy mô phỏng 16. Chọn "Dừng" để dừng mô phỏng tạm thời 17. Chọn "Lưu" để lưu thông số mô phỏng lại cho lần sau có thể sử dụng lại mà không cần phải nhập lại. 18. Chọn "Bắt đầu lại" để trở về trạng thái ban đầu khi mô phỏng chưa được chạy
Tiến trình phụ:	3. Bấm nút "Back" để thoát mô phỏng và trở lại màn hình chính
Các ngoại lệ:	
Các use case liên quan:	
Yêu cầu riêng biệt:	
Các giả định:	
Ghi chú và các vấn đề:	5. Các thông số trong quá trình mô phỏng thay đổi theo thời gian thực 6. Độ sai số chỉ khoảng 1/10000



Hình 5.3

Nội dung mô hình:

- User: người sử dụng chương trình.
- Nhập dữ liệu vào mô phỏng: đây là quá trình người dùng nhập những thông số cần thiết cho quá trình chuyển động. Đối với ném xiên người dùng nhập những thông số như trọng lượng vật, độ cao ban đầu, góc ném.
- Đọc file có sẵn: người dùng chọn những file có sẵn đã được tạo ra bởi hệ thống. Các file này chính là những file lưu sẵn trước đó của mô phỏng.

- Nhập dữ liệu vào mô phỏng: đây chính là bước tổng hợp từ hai bước: nhập dữ liệu vào mô phỏng và đọc dữ liệu từ file. Bước này sẽ khởi tạo các thông số cho mô phỏng được chạy.
- Lưu quá trình mô phỏng: sau khi thiết lập những thông số cơ bản cho quá trình mô phỏng người dùng có thể lưu lại quá trình mô phỏng.
- Tạm dừng quá trình mô phỏng: khi quá trình mô phỏng đang diễn ra, có thể tạm dừng chương trình để đọc những thông số tại thời điểm đó. Điều kiện cần để dừng quá trình mô phỏng là mô phỏng đang được chạy.
- Tiếp tục quá trình mô phỏng: sau khi đã tạm dừng mô phỏng để đọc những thông số cần thiết người dùng có thể tiếp tục lại quá trình mô phỏng. Điều kiện cần để tiếp tục mô phỏng là mô phỏng đang được dừng lại.
- Thoát: người chơi muốn kết thúc mô phỏng.

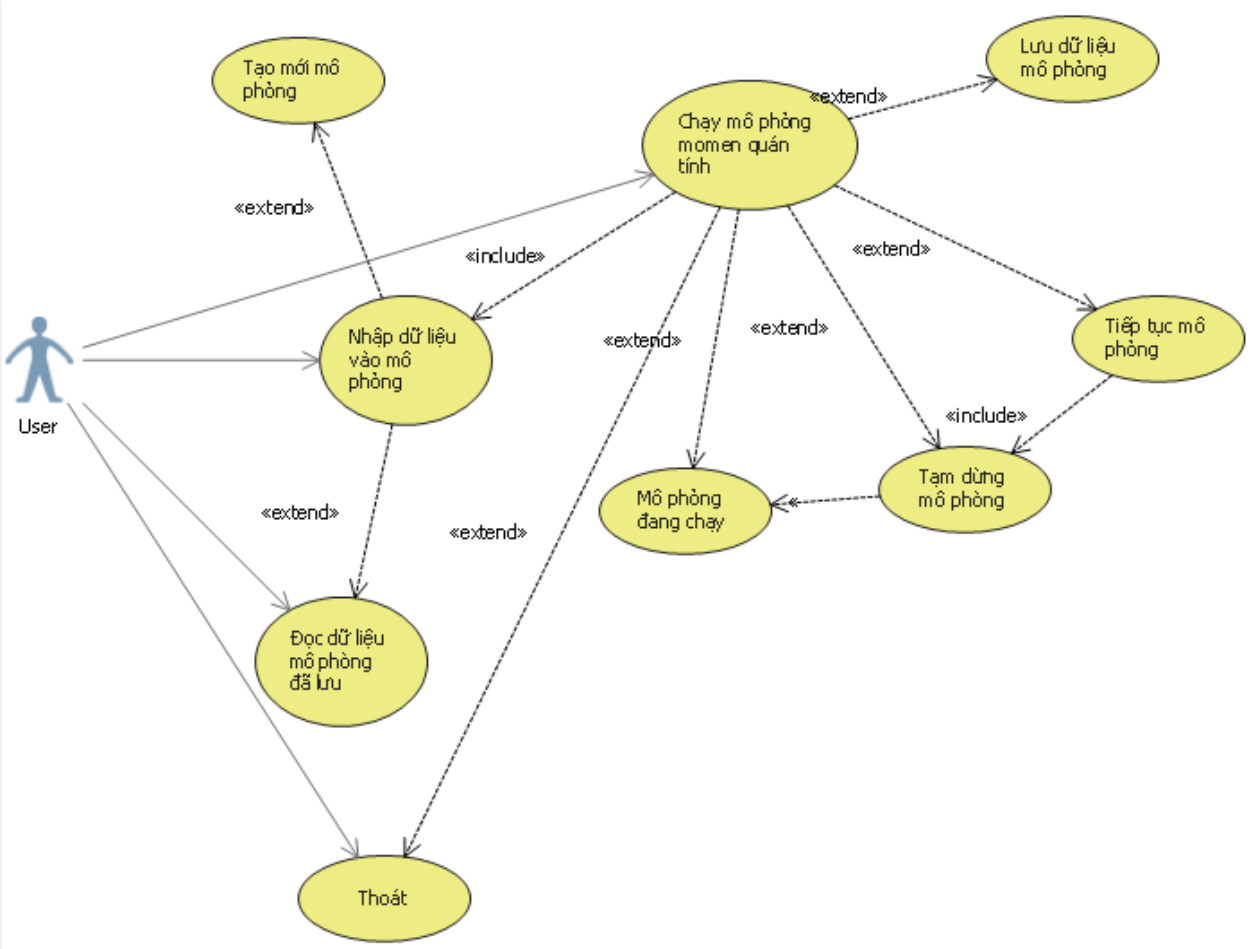
❖ **Mô hình use case cho con momen quán tính:**

Đặc tả Use case:

Bảng 5.4

Use case ID:	UC_02		
Tên use case:	Mô phỏng momen quánh tính		
Người tạo:	Trần Thanh Sơn	**Người cập nhật cuối:**	Trần Thanh Sơn
Ngày tạo:	23/10/2013	**Ngày cập nhật cuối:**	23/10/2013
Nhân vật:	User		
Mô tả:	Khi người dùng vào chức năng mô phỏng của momen quánh tính thì ban đầu sẽ tạo những giá trị mặc định. Người dùng có thể thay đổi các giá trị ban đầu với chức năng nhập thông số hoặc đọc từ danh sách các thông số đã được lưu		

Điều kiện kích hoạt:	Người dùng chọn chức năng mô phỏng con lắc đơn
Điều kiện đầu:	7. Vào ứng dụng 8. Chọn mô phỏng momen quánh tính
Điều kiện sau:	4. Trở lại màn hình menu chính.
Tiến trình thường:	19. Chọn mô phỏng con lắc đơn 20. Chọn "nhập thông số" hoặc "dữ liệu đã lưu" dưới thanh Application bar để thiết lập thông số cho mô phỏng 21. Chọn "Chạy" để chạy mô phỏng 22. Chọn "Dừng" để dừng mô phỏng tạm thời 23. Chọn "Lưu" để lưu thông số mô phỏng lại cho lần sau có thể sử dụng lại mà không cần phải nhập lại. 24. Chọn "Bắt đầu lại" để trở về trạng thái ban đầu khi mô phỏng chưa được chạy
Tiến trình phụ:	4. Bấm nút "Back" để thoát mô phỏng và trở lại màn hình chính
Các ngoại lệ:	
Các use case liên quan:	
Yêu cầu riêng biệt:	
Các giả định:	
Ghi chú và các vấn đề:	7. Các thông số trong quá trình mô phỏng thay đổi theo thời gian thực 8. Độ sai số chỉ khoảng 1/10000



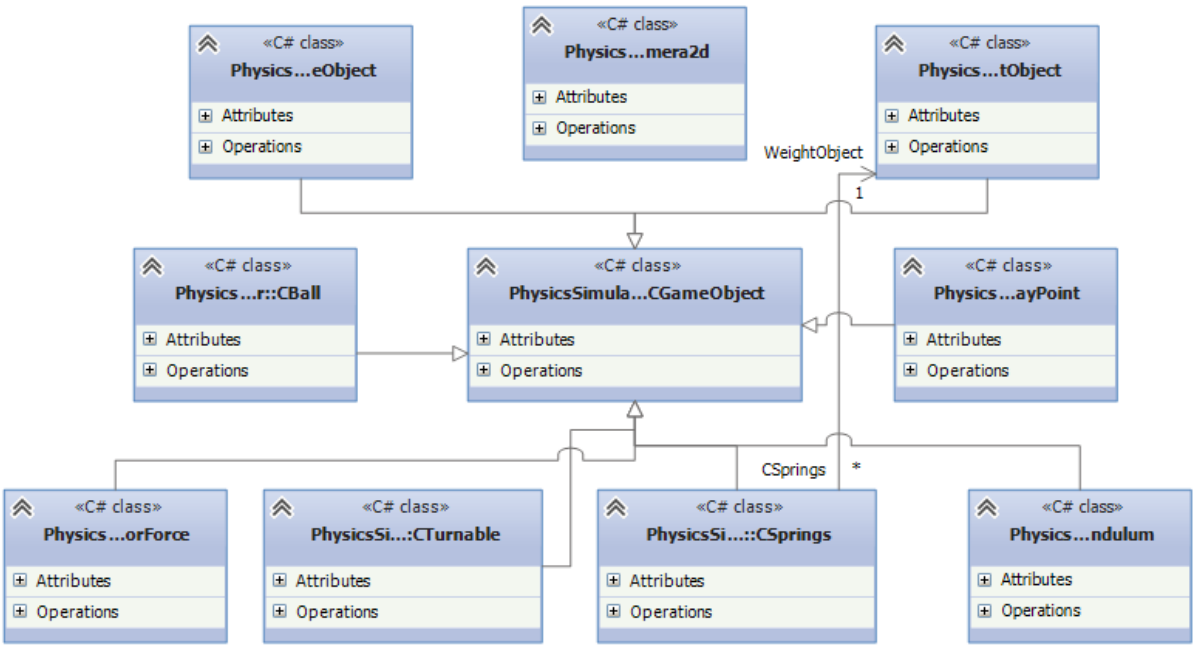
Hình 5.4

Nội dung mô hình:

- User: người sử dụng chương trình.
- Nhập dữ liệu vào mô phỏng: đây là quá trình người dùng nhập những thông số cần thiết cho quá trình chuyển động. Đối với momen động lượng người dùng nhập những thông số như chiều dài thanh, khối lượng thanh, vận tốc góc hoặc lực tác dụng để thanh có thể quay quanh trục.
- Đọc file có sẵn: người dùng chọn những file có sẵn đã được tạo ra bởi hệ thống. Các file này chính là những file lưu sẵn trước đó của mô phỏng.

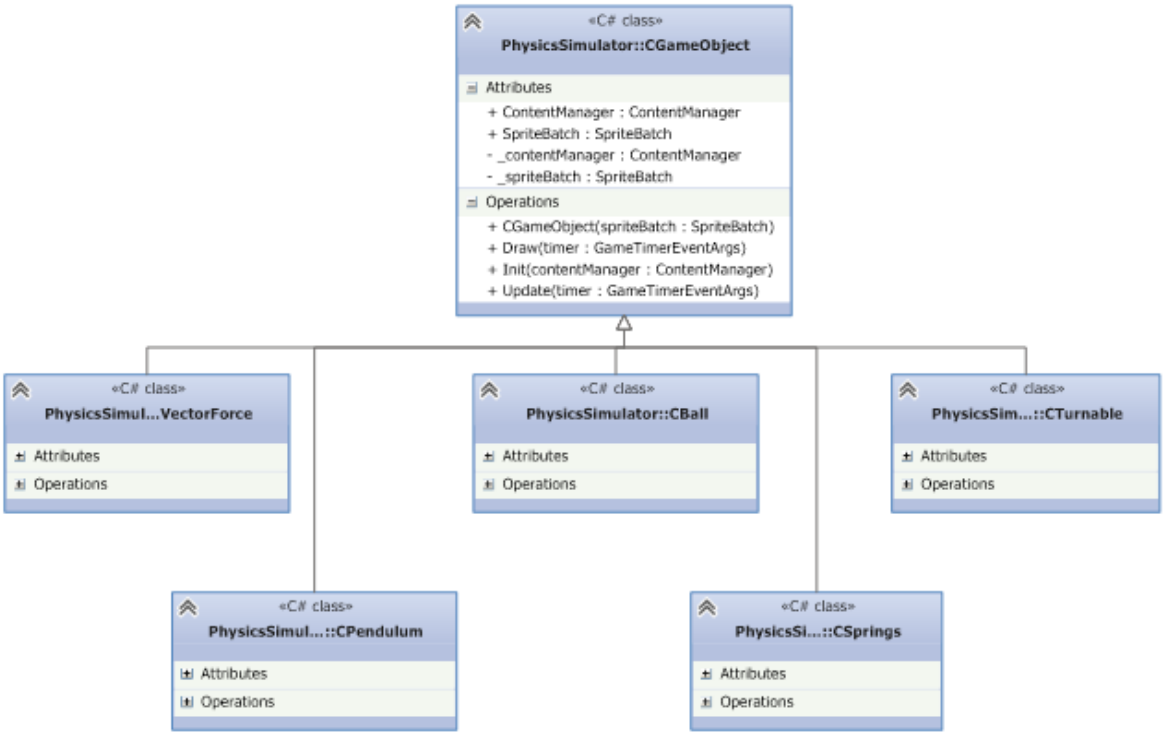
- Nhập dữ liệu vào mô phỏng: đây chính là bước tổng hợp từ hai bước: nhập dữ liệu vào mô phỏng và đọc dữ liệu từ file. Bước này sẽ khởi tạo các thông số cho mô phỏng được chạy.
- Lưu quá trình mô phỏng: sau khi thiết lập những thông số cơ bản cho quá trình mô phỏng người dùng có thể lưu lại quá trình mô phỏng.
- Tạm dừng quá trình mô phỏng: khi quá trình mô phỏng đang diễn ra, có thể tạm dừng chương trình để đọc những thông số tại thời điểm đó. Điều kiện cần để dừng quá trình mô phỏng là mô phỏng đang được chạy.
- Tiếp tục quá trình mô phỏng: sau khi đã tạm dừng mô phỏng để đọc những thông số cần thiết người dùng có thể tiếp tục lại quá trình mô phỏng. Điều kiện cần để tiếp tục mô phỏng là mô phỏng đang được dừng lại.
- Thoát: người chơi muốn kết thúc mô phỏng.

5.2 Thiết kế



Hình 5.5

5.2.1 Lớp CGameObject:



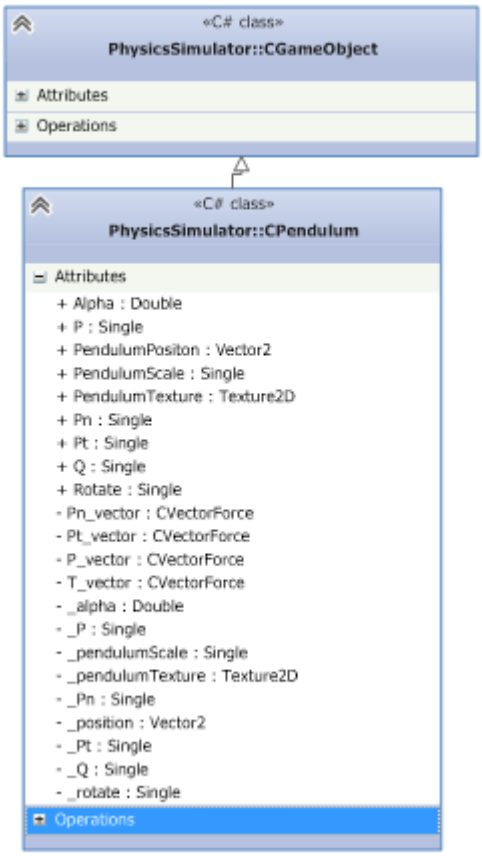
Hình 5.6

Lớp CGameObject là lớp đối tượng chung trong mô phỏng. Nó là cơ sở cho các lớp khác kế thừa như các lớp CVectorForce, Cpendulum, CBall, Cturnable, CSpring,…

Lớp CGameObject bao gồm các phương thức ảo (Visual):

- Init: Lớp này là lớp khởi tạo các thể hiện của đối tượng trên màn hình.
- Update: Cập nhật thông số của đối tượng theo thời gian.
- Draw: Lớp này được dung để vẽ đối tượng lên màn hình khi gọi.

5.2.2 Lớp CPendulum (Con lắc đơn):

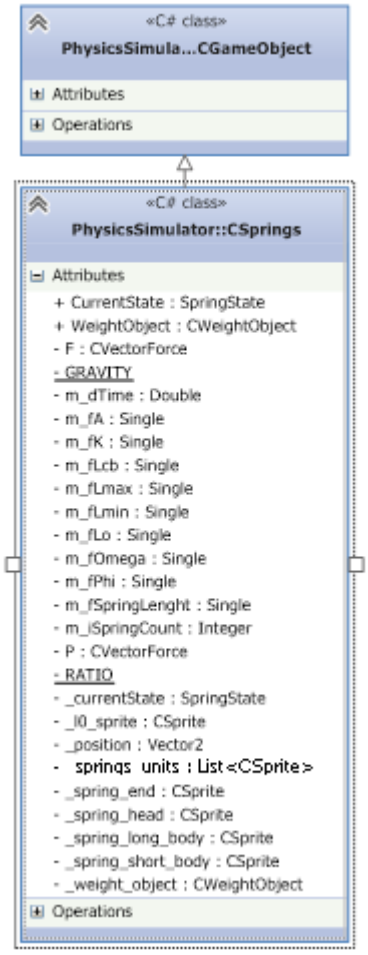


Hình 5.7

Lớp CPendulum kế thừa các phương thức ảo từ CGameObject. Ngoài ra nó còn chứa các thành phần trong tính toán các giải trị Vật lý dao động của con lắc đơn. Chi tiết các object Vật lý trong CPendulum như sau:

- Alpha: Góc dao động của con lắc đơn theo thời gian
- P: trọng lực tác dụng lên con lắc
- Các lực thành phần: Pn, Pt
- Lực căng dây Q

5.2.3 Lớp CSprings (Con lắc lò xo):

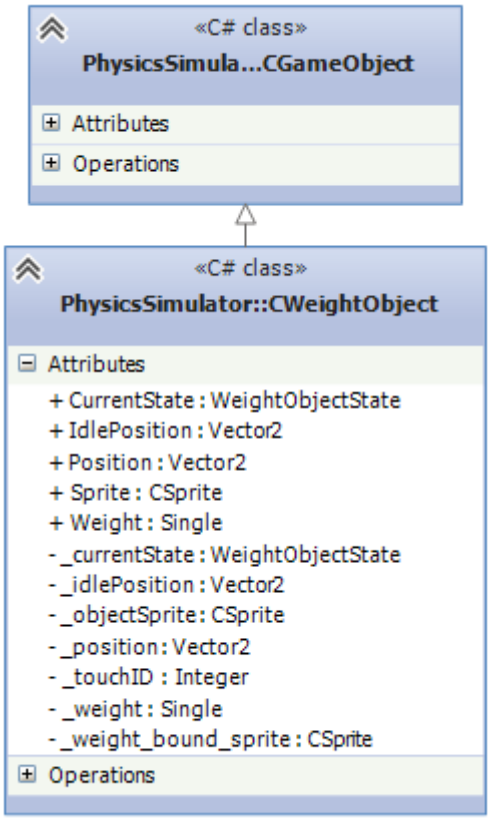


Hình 5.8

Lớp CSprings kế thừa các phương thức ảo từ CGameObject. Ngoài ra nó còn chứa các thành phần trong tính toán các giái trị Vật lý dao động của con lắc lò xo. Chi tiết các object Vật lý trong CSprings như sau:

- Gravity
- m_fA: biên độ dao động lớn nhất của con lắc lò xo
- Chiều dài cực tiểu và cực đại của lò xo:
- mf_Omega: giá trị Omega trong công thức của con lắc lò xo
- mf_Phi: Pha ban đầu của con lắc lò xo khi dao động

5.2.4 Lớp CWeightObject (Con lắc lò xo):

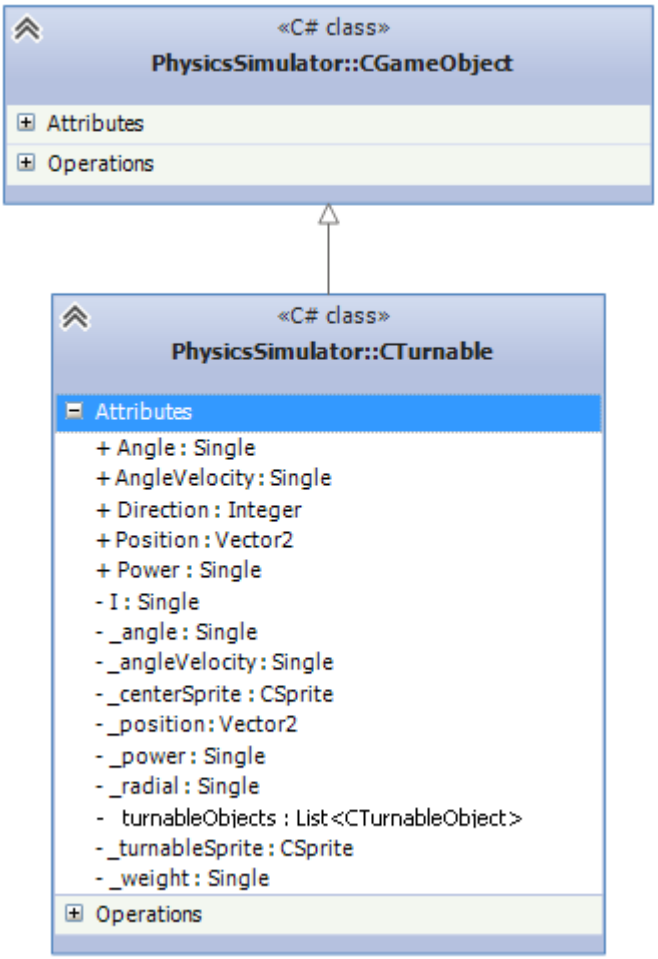


Hình 5.9

Lớp CWeightObject kế thừa các phương thức ảo từ CGameObject. Đây là lớp mô tả đối tượng là các quả cân trọng lượng trong mô phỏng dao động của con lắc lò xo. Nó có các thành phần chính như:

- IdlePosition: Vị trí trong lúc quả cân ở trang thái nhà rỗi
- Position: Vị trí có thể thai đỗi trong lúc dao động dựa và công thức dao động của con lắc lò xo
- Weight: Khối lượng của quả cân treo trên lò xo (Kg)

5.2.5 Lớp CTurnable (Momen quán tính):

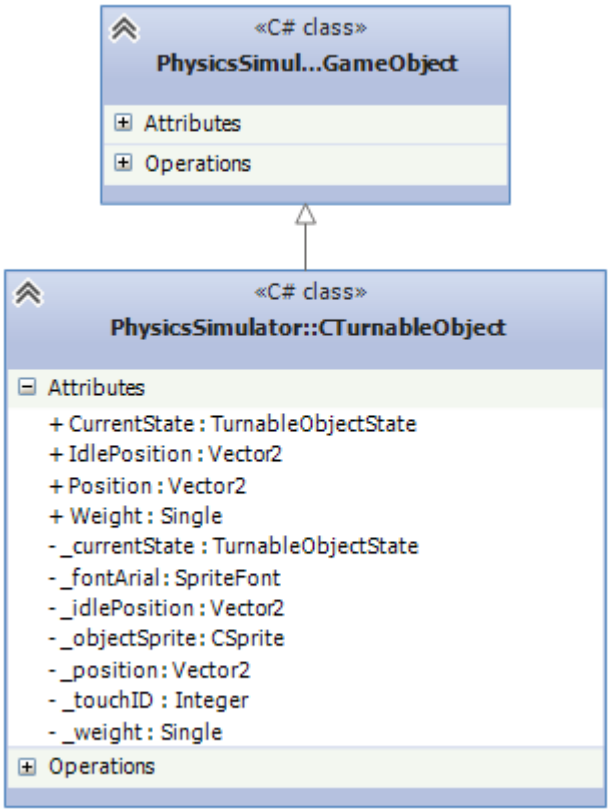


Hình 5.10

Lớp CTurnable kế thừa các phương thức ảo từ CGameObject. Ngoài ra nó còn chứa các thành phần trong tính toán các giải trị Vật lý dao động của momen quán tính. Chi tiết các object Vật lý trong CTurnable như sau:

- Angle: góc quay ban đầu của đĩa quay
- AngleVelocity: Vận tốc góc của đĩa quay
- Direction: hướng quay đĩa
- Power: lực tác dụng làm quay đĩa

5.2.6 Lớp CTurnableObject (Momen quán tính):

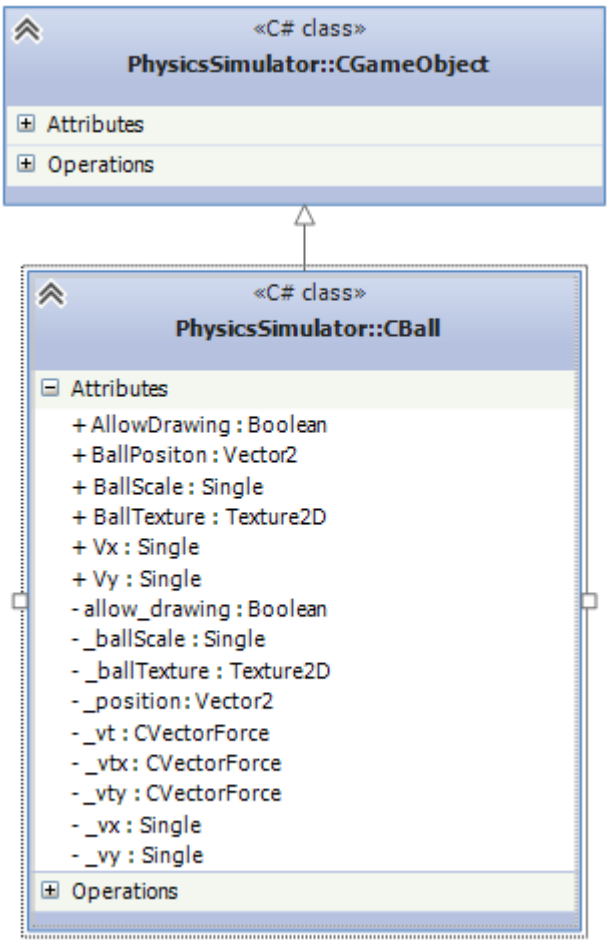


Hình 5.11

Lớp CTurnableObject kế thừa các phương thức ảo từ CGameObject. Đây là lớp mô tả đối tượng là các quả cầu trọng lượng trong mô phỏng dao động của momen quán tính. Nó có các thành phần chính như:

- CurrentState: Trạng thái của quả cầu
- IdlePosition: Vị trí khi đang nhàn rỗi
- Position: Vị trí dao động khi trạng thái doa động
- Weight: Khối lượng quả cầu (Kg)

5.2.7 Lớp CBall (Vật bị ném xiên):

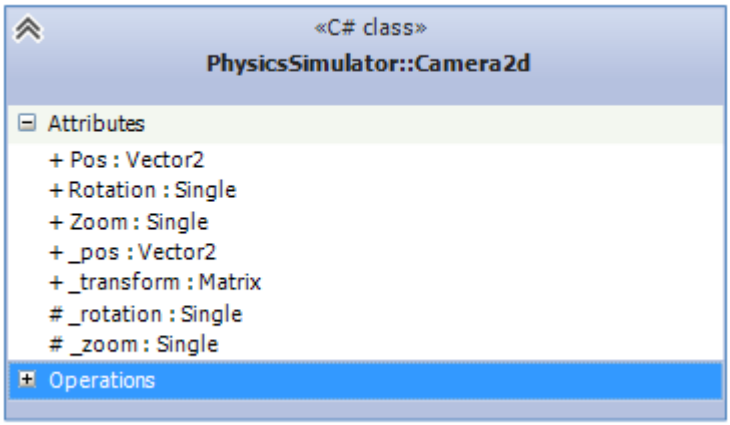


Hình 5.12

Lớp CBall kế thừa các phương thức ảo từ CGameObject. Ngoài ra nó còn chứa các thành phần trong tính toán các giải trị Vật lý dao động của vật bị ném xiên. Chi tiết các object Vật lý trong CBall như sau:

- BallPosition: Vị trí của vật trong quá trình doa động
- Vx: Vận tốc dao động theo phương ngang tại thời gian t
- Vy: Vận tốc dao động theo phương thẳng đứng tại thời gian t

5.2.8 Lớp Camera2d



Hình 5.13

Lớp này được dung để thay đổi các góc nhìn trong quá trình mô phỏng. Do kích thước màn hình của điện thoại là có giới hạn cho nên cần phải có lớp này để có thể thay đổi góc nhìn phù hợp để có thể thấy được toàn bộ quá trình mô phỏng. Người sử dụng có thể thay đổi góc nhìn tùy biến.

5.3 Xử lí

5.3.1 Con lắc lò xo:

Trong thực tế, con lắc lò xo có nhiều dạng như con lắc nằm ngang, con lắc thẳng đứng, con lắc nằm xiên… Các dạng thường thấy nhất của con lắc lò xo là dạng nằm ngang và dạng thẳng đứng. Nhưng thực tế, khi thí nghiệm, người ta thường chọn con lắc lò xo thẳng đứng vì nó chịu ít ma sát với môi trường hơn con lắc nằm ngang. Con lắc lò xo thẳng đứng chịu tác dụng của lực lò xo, trọng trường và lực ma sát. Ở đây, trong phần mềm mô phỏng này, ta sẽ chọn con lắc thẳng đứng để mô phỏng.

Độ giãn ban đầu của con lắc lò xo khi treo quả cân:

- Như ta đã biết, khi treo một quả cân có khối lượng ***m*** lên con lắc lò xo có độ cứng là ***k***. Dưới tác dụng của lực trọng trường của trái đất ***g***, lò xo sẽ bị giãn ra một khoảng Δl_0theo công thức:

$$\Delta l_0 = \frac{m.g}{k}$$

- Phần mềm mô phỏng cũng phải tính tới yếu tố này. Khi ta thay đổi khối lượng ***m*** của quả cân thì độ giãn ban đầu $\Delta \boldsymbol{l_0}$ của nó cũng phải khác nhau để đảm bảo đúng tính thực tế.

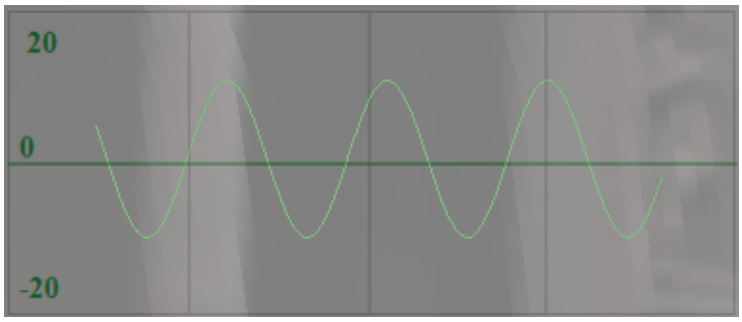
Xử lý ma sát của con lắc lò xo:

- Con lắc lò xo chịu rất nhiều lực cản làm nó dừng lại như: ma sát không khí, lực bị hao phí ở đoạn nối giữa con lắc lò xo và đinh treo, lực hao phí khi lò xo chuyển động tạo thành nhiệt năng,… Vì vậy, không có một công thức chính xác nào để tính lực ma sát tổng hợp lên con lắc lò xo làm cho nó đứng yên. Ta chỉ xét nó ở một dạng năng lượng hao phí chung, năng lượng hao phí này làm cho biên độ ***A*** của dao động giảm lại theo thời gian. Biên độ ***A*** giảm làm cho động năng $\boldsymbol{W_đ}$ và thế năng $\boldsymbol{W_t}$ giảm theo, từ đó ta tính được năng lượng hao phí này thông qua công thức:

$$W_{hao\ phí} = W - (W_t + W_đ)$$

Vẽ đồ thi cho dao động:

- Đồ thị của con lắc lò xo bao gồm: đồ thị về lực tác dụng lên vật, đồ thị gia tốc của vật, đồ thị cho các dạng năng lượng: động năng, thế năng, năng lượng hao phí và cơ năng.
- Đồ thị đặt trưng của dao động điều hòa đó là dạng đồ thị hình ***sin*** biến đổi theo thời gian. Đồ thị phải cho thấy được sự thay đổi điều hòa đó đúng với thời gian thực tế.



Hình 5.14

Lực lò xo tác dụng:

- Theo công thức tính lực đàn hồi của Hooke thì:

$$F_{đh} = -k.x$$

- Ở đây x chính là độ giãn của lò xo. Trong phương trình dao động điều hòa thì ta cũng có một đại lượng là x (vị trí của hòn bi). Nhưng ta nên nhớ rằng vị trí x của hòn bi khác với x độ giãn của lò xo, vì ban đầu, lò xo đã bị giãn một đoạn là Δl_0 do sức năng của hòn bi kéo xuống. Nên công thức tính lực đàn hồi sẽ được viết lại như sau:

$$F_{đh} = -k.(x+\Delta l_0)$$

5.3.2 Con lắc đơn:

Môi trường lý tưởng cho con lắc đơn:

- Điều kiện dao động điều hoà:Bỏ qua ma sát, lực cản và $\alpha_0 \ll 10^0$hay $S_0 \ll l$.
- Nhưng khi mô phỏng thì khi quan sát góc quay $\alpha_0 \ll 10^0$ rất khó vì gốc α_0 quá nhỏ.

- Vì vậy ta phải giả thuyết nó trong điều kiện lý tưởng, có nghĩa là với $\alpha_0 \geq 10^0$và $\alpha_0 \ll 90^0$ thì dao động con lắc đơn vẫn là dao động điều hòa. Nó cũng giống như là ta phóng đại góc quay của con lắc đơn lên 9 lần vậy.

Kéo thả trong con lắc đơn:

- Việc xử lý cũng giống như kéo thả trong con lắc lò xo, như đã nói ở trên, ta cũng xác định đường thẳng từ ngón tay tiến thẳng vào màn hình. Ở đây chỉ khác là việc kéo thả trong con lắc đơn theo một phương hình cung, để con lắc có thể quay theo.
- Và việc xác định pha ban đầu khi kéo thả cũng tương tự như con lắc lò xo. Có nghĩa là khi ta kéo hòn bi mà hòn bi đã ở một vị trí có góc α lớn hơn phương thẳng đứng (khi vật cân bằng) thì lúc này ta cho dao động mới có $\alpha_0 = \alpha$ và pha ban đầu là: θ = 0 *(vì* $cos\, 0 = 1$*)*.
- Còn nếu α có tạo độ bé hơn vị trí cân bằng thì lúc này α_0 của dao động vẫn là $\alpha_0 = \alpha$ nhưng khi đó pha ban đầu sẽ là: θ = π *vì* $cos\, \pi = -1$*)*

Xử lý ma sát:

- Con lắc đơn cũng chịu nhiều lực tác dụng làm hao phí năng lượng và làm con lắc dừng lại. Năng lượng hao phí này cũng sẽ được tính theo công thức:

$$W_{hao\ phí} = W - (W_t + W_đ)$$

- Khi năng lượng hao phí càng cao thì dao động sẽ dừng lại càng nhanh có nghĩa là góc quay ban đầu α_0 sẽ giảm càng nhanh.

Vấn đề chiều dài của con lắc đơn:

- Để mô phỏng đầy đủ các dạng của con lắc đơn thì ta cần phải cho chiều dài của con lắc có nhiều kích thước. Độ chênh lệch giữa kích thước lớn nhất và kích thước nhỏ nhất là rất lớn, ta không thể scale theo đúng tỉ lệ được.

- Để giải quyết điều này thì trong phần mềm mô phỏng, ta sẽ giữ nguyên chiều dài của con lắc và đưa ra cho người dùng biết một tỉ lệ giữa kích thước mô phỏng và kích thước thực tế.

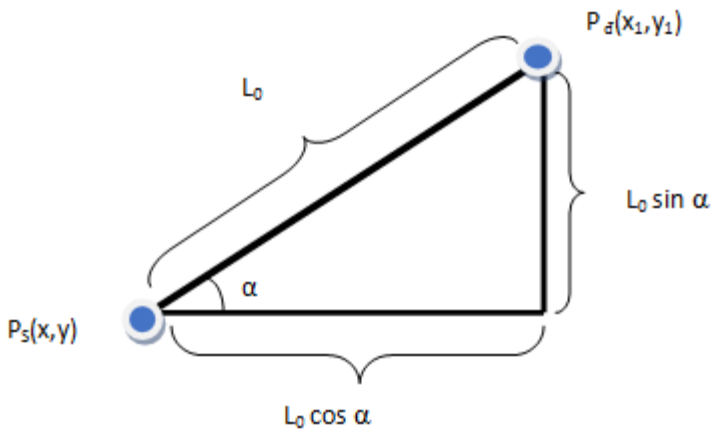
5.3.3 Ném xiên:

Chọn bối cảnh ném xiên:

- Ta lấy bối cảnh ném xiên là một khẩu súng đại bác bắn ra viên đạn, chuyển động của viên đạn là chuyển động ném xiên trong không gian 2D.

Xử lý vị trí ban đầu của khẩu súng:

- Trong chuyển động ném xiên có 3 thông số quan trọng là: Độ cao ban đầu H_0, vận tốc khi ném V_0 và góc ném α_0. Vị trí ban đầu của viên đạn có tọa độ trong không gian là x và z không thay đổi, ta chỉ thay đổi độ cao của viên đạn. Vấn đề ở đây là ta phải canh cho vị trí của viên đạn trùng với đầu của nòng súng, vị trí này sẽ thay đổi khi ta thay đổi độ cao và góc ném của vật. Để xử lý vấn đề này ta dùng công thức sau:



- Với L_0 là chiều dài của súng, α là góc ném, P_S là vị trí của súng, $P_đ$ là vị trí của viên đạn cũng chính là vị trí đầu nòng súng.
- Ở đây ta sẽ có vị trí Ps của súng sẽ được tính theo công thức:

$$x = x_1 - L_0 \cos\alpha$$

$$y = y_1 - L_0 \sin\alpha$$

- Như vậy ta đã cập nhật lại vị trí mới của súng khi độ cao và góc nghiêng thay đổi, làm cho viên đạn ở chính xác vị trí của đầu nòng súng.

Vẽ lại đường đi cho chuyển động:

- Việc vẽ lại đường đi cho chuyển động ném xiên sẽ giúp ta có được cái nhìn dễ hơn, sâu hơn về chuyển động ném xiên. Phần mềm sẽ vẽ lại đường đi của viên đạn, phân tích và hiển thị ra độ cao lớn nhất mà viên đạn đạt được, cũng như khoảng cách xa nhất mà viên đạn di chuyển được trước khi chạm đất.

5.3.4 Momen:

5.3.4.1 Vấn đề đặt ra:

- Xác định loại momen.
- Xác định cấu tạo của mô phỏng.
- Xác định bài toán mô phỏng.
- Hiển thị mô phỏng 2D lên màn hình.

5.3.4.2 Giải quyết vấn đề:

Chương trình chọn mô phỏng momen quán tính

Trong thực tế chuyển động quay momen xuất hiện rất nhiều VD: chuyển động quay của bánh xe, bóng lăn,.... Ở đây chương trình chọn chuyển động quay quanh một trục cố định của thanh nằm ngang.

Cấu tạo của mô phỏng gồm một bàn quay với một thanh ma sát để giảm tốc độ của bàn quay chậm dần cùng với các vật là các quả cầu có khối lượng khác nhau được dung để đặt lên bàn quay và xem thể hiện của từng vật.

Hiển thị mô phỏng Momen lên màn hình:

- Chương trình mô phỏng 2 hệ vật có cấu tạo như trên.
- Các đại lượng ban đầu:

- Vận tốc góc ban đầu chính là số vòng quay trong 1 phút: "ω"
- Khối lượng bàn quay: "m"

- Đổi vận tốc góc qua rad/s. Sử dụng công thức: $\omega.\frac{\pi}{60}$
- Từ những thông số ban đầu tính được momen quán tính: $I = \frac{1}{12}ml^2$
- Tính được momen động lượng: $L = I.\omega$
- Lúc sau khi thay đổi chiều dài thanh thì vận tốc góc sẽ được tính lại theo công thức: $\omega = \frac{L}{l}$ do động lượng trong quá trình chuyển động không thay đổi.
- Thông số được tính sẽ được vẽ ra trên màn hình.
- Để hiển thị quá trình chyển động sử dụng thuộc tính "Rotation" khi thay đổi vận tốc góc thì "Rotation" của model thay đổi, nhưng không thay đổi toàn model mà chỉ thay đổi theo từng bone.
- Để mô phỏng chiều dài thanh thay đổi sử dụng thuộc tính "Scale" khi thanh thay đổi chiều dài thì "Scale" phù hợp với chiều dài. Nhưng không thay đổi toàn model mà chỉ thay đổi theo từng bone.

Hiển thị momen lực lên màn hình:

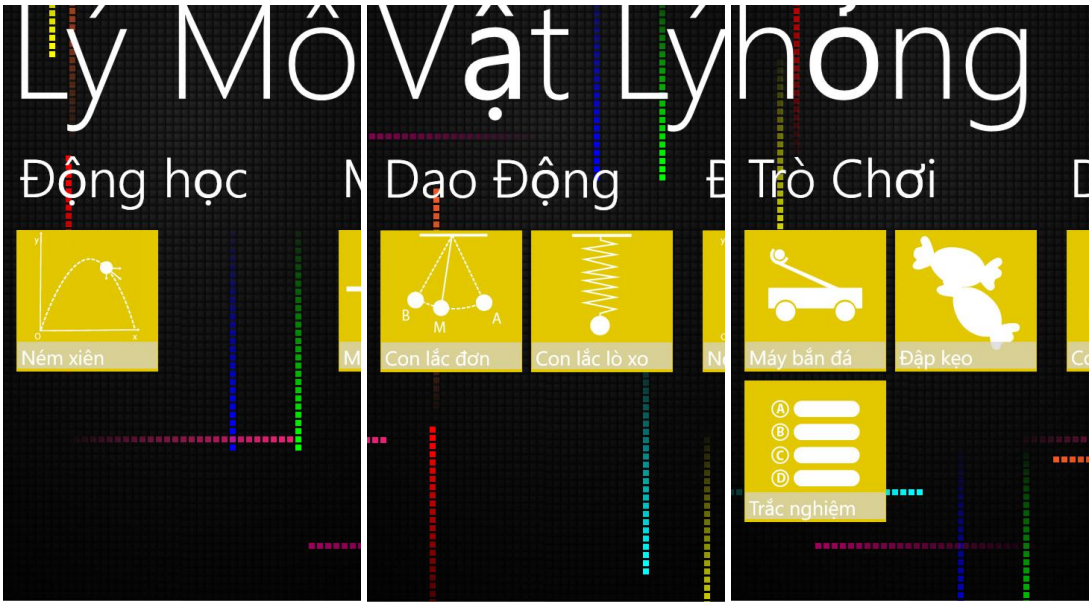
- Chương trình mô phỏng một hệ vật có cấu tạo như trên ban đầu vật chuyển động được nhờ một lực tác dụng vật sẽ chuyển động nhanh dần đều sau một khoảng thời gian tác dụng một lực Fms vào khi đó vật sẽ chuyển động chậm dần và dừng lại sau khoảng thời gian t.
- Các đại lượng ban đầu:
  - Bán kính bàn quay "R".
  - Khối lượng bàn quay "m".
  - Vận tốc góc của bàn quay "V".
- Từ các đại lượng ban đầu tính được momen lực: $M = F\frac{l}{2}$

- Momen quán tính: $I = \frac{1}{12}ml^2$
- Gia tốc góc: $\gamma = \frac{M}{I}$
- Phương trình vận tốc góc: $\omega_t = \omega_0 + \gamma t$
- Khi tác dụng một lực vật sẽ chuyển động với một vận tốc góc mới: $\omega_{t'} = \omega_t + \gamma_h t$.
  - $\omega_{t'}$ vận tốc góc lúc sau.
  - ω_t vận tốc góc khi vật vừa bị tác dụng một lực hãm.
  - γ_h gia tốc góc sau khi tác dụng lực hãm.
- Khi vật dừng quá trình quay thì vận tốc góc của vật bằng 0 khi đó tính được thời gian chuyển động của vật sau khi tác dụng lực hãm.

Để hiển thị quá trình chyển động sử dụng thuộc tính "Rotation" khi thay đổi vận tốc góc thì "Rotation" của model thay đổi, nhưng không thay đổi toàn model mà chỉ thay đổi theo từng bone xác định trong những khoảng thời gian xác định.

5.4 Giao diện chương trình:

5.4.1 Giao diện chính:



Hình 5.15

Hiển thị các chức năng của ứng dụng

Mỗi chức năng tương ứng mỗi item

Chức năng được phân chia theo từng mục cụ thể, để thao tác

5.4.2 Giao diện chung của mô phỏng:

Giao diện chung của các mô phỏng bao gồm các chức năng:

- Play/Pause
- Xem các thông số trong quá trình mô phỏng
- Nhập các thông số đầu vào cho mô phỏng
- Lưu lại thông số mô phỏng
- Load các thông số mô phỏng đã lưu

5.4.3 Giao diện mô phỏng Con Lắc Đơn:

Hình 5.16

Có các textbox nhập các thông số của chuyển động:

- Alpha: Góc dao động.
- L: Chiều dài của sợi dây.
- m: Khối lượng của vật
- θ: Pha ban đầu.
- g: Gia tốc trọng trường nơi đặt con lắc

Hiển thị thông tin khi chuyển động:

- t: thời gian dao động
- T: chu kì dao động
- s: Biên độ dao động theo thời gian
- V: Vận tốc của con lắc tại một thời điểm t.
- Vmax: Vận tốc tối đa của con lắc.
- P: Trọng lực tác dụng lên co lắc
- Pn, Pt: Các lực thành phần tác dụng lên con lắc
- Q: Lực căng dây theo thời gian

- Qmax: Lực căng dây tối đa
- Qmin: Lực căng dây tối thiểu
- Wđ: Động năng con lắc.
- Wt: Thế năng của con lắc
- W: Năng lượng toàn phần của con lắc.

5.4.4 Giao diện mô phỏng con lắc lò xo

Hình 5.17

Có các textbox nhập các thông số của chuyển động:

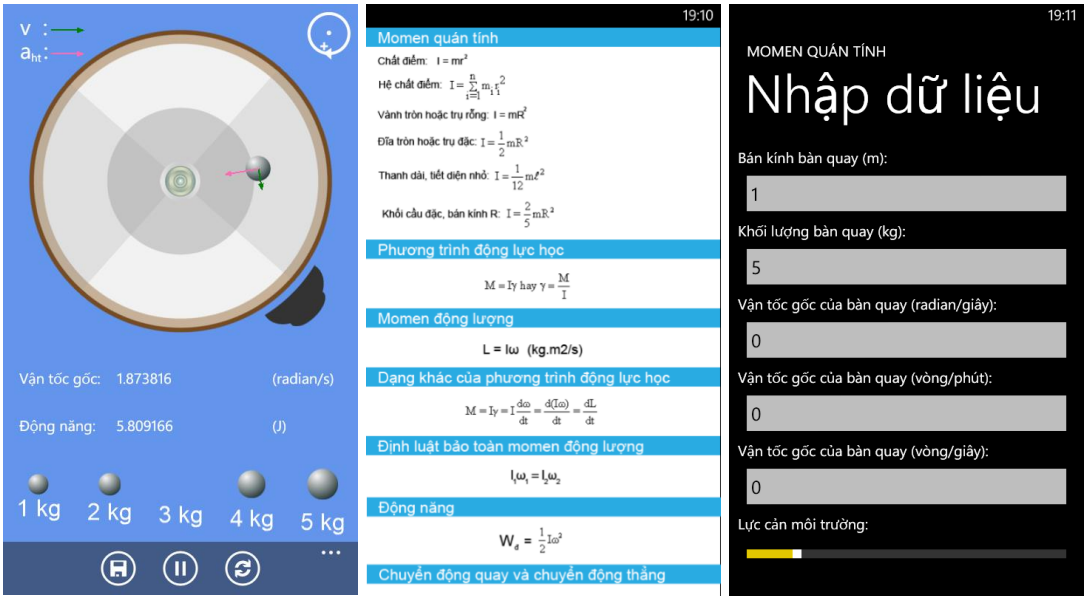
- Alpha: Góc dao động.
- K: Độ cứng của lò xo
- m: Khối lượng của vật
- θ: Pha ban đầu.
- g: Gia tốc trọng trường tại nơi đặt lò xo
- Lực cản môi trường

Hiển thị thông tin khi chuyển động:

- t: thời gian dao động

- s: Biên độ dao động theo thời gian
- V: Vận tốc biến thiên doa động
- Vmax: vận tốc tối đa
- Wt: Thế năng
- Wđ: động năng
- W: Năng lượng toàn phần

5.4.5 Giao diện dao động của monen quán tính:



Hình 5.18

Có các textbox nhập các thông số của chuyển động:

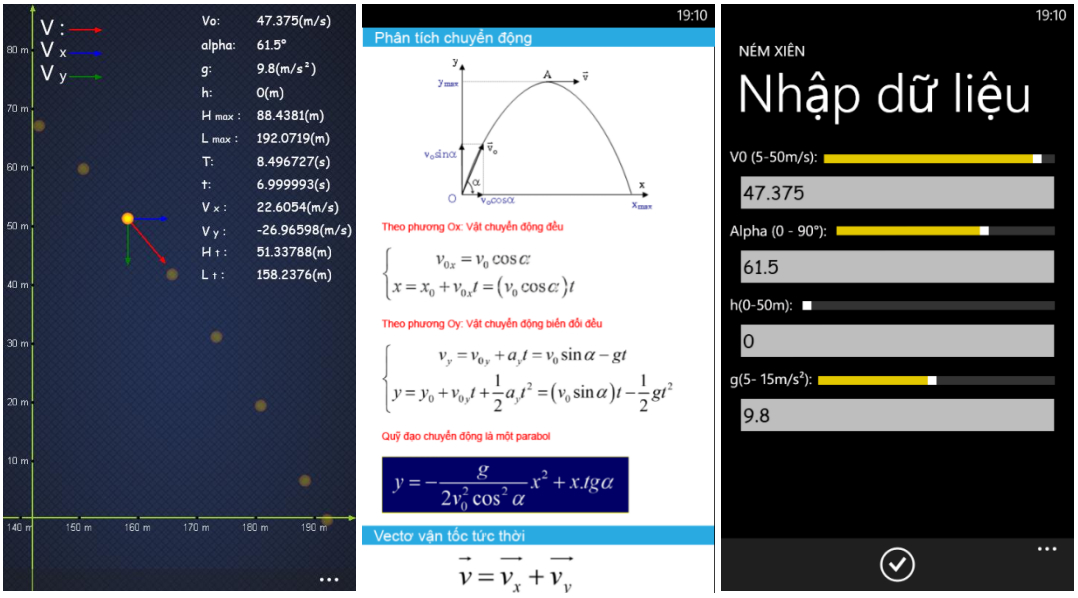
- R: Bán kính bàn quay
- m: Khối lượng bàn quay
- V: Vận tốc góc của bàn quay
- Lực cản môi trường

Hiển thị thông tin khi chuyển động:

- t: thời gian quay
- V: Vận tốc góc biến thiên

- Wđ: động năng

5.4.6 Giao diện mô phỏng Ném Xiên:



Hình 5.19

Có các textbox nhập các thông số của chuyển động:

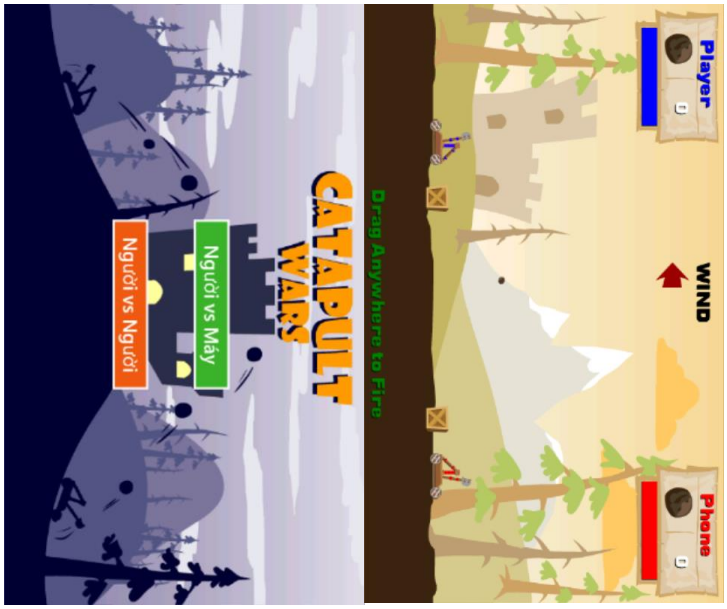
- V0: Vận tốc ban đầu
- Alpha: Góc hợp với phương ngang
- h: Khoảng cách với mặt đất
- g: Gia tốc trọng trường tại nơi tiến hành mô phỏng

Hiển thị thông tin khi chuyển động:

- t: thời gian thực hiện mô phỏng
- T: Tổng thời gian từ khi ở vị trí ban đầu cho đến khi chạm đất
- Vx: Vận tốc theo phương ngang biến thiên theo thời gian t
- Vy: Vận tốc theo phương thẳng đứng biến thiên theo thời gian t
- Hmax: Chiều cao tối đa mà vật đạt được
- Lmax: Quãng đường xa nhất mà vật đạt được trong khoảng thời gian T
- H: Chiều cao biến thiên theo thời gian t

- L: Quảng đường biến thiên theo thời gian t

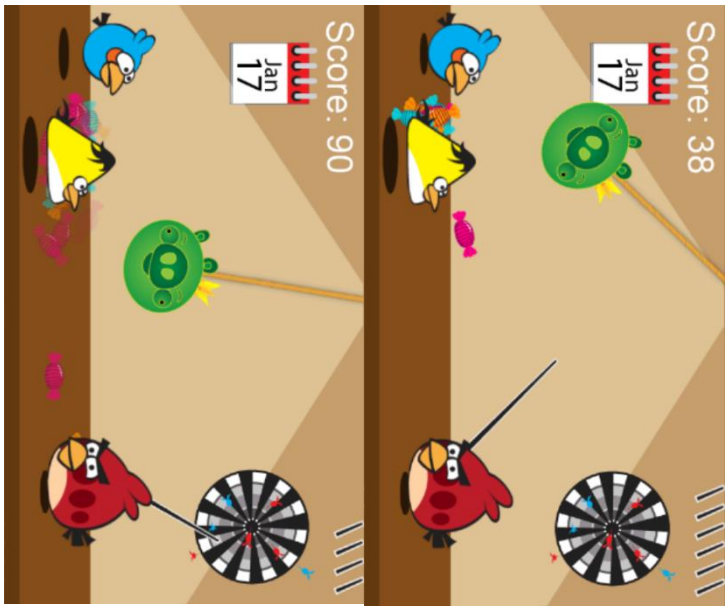
5.4.7 Giao diện game Bắn đá



Hình 5.20

Game bắn đá được xây dựng dựa theo mô phỏng ném xiên của vật. Trong đó các viên đá là vật được bắn đi và mục tiêu là trúng đối tượng là chiếc máy bắn đá khác có thể là máy hoặc người chơi thứ hai. Trong lúc bắn viên đá có sụ ảnh hưởng của thời tiết là gió. Bắn đến khi nào một bên máy bắn đá bị phá hủy là kết thúc trò chơi và có thể chơi lại.

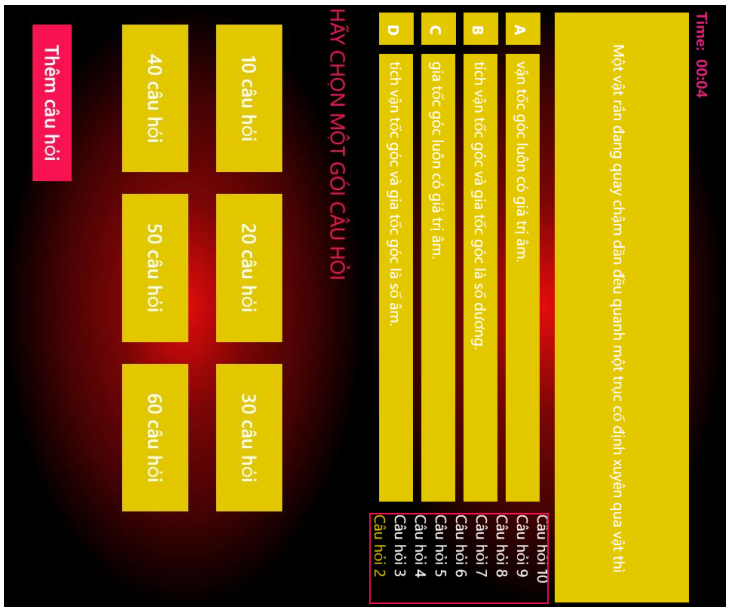
5.4.8 Giao diện game Đập kẹo



Hình 5.21

Game đập kẹo mô phỏng dựa theo bài toán do động của con lắc đơn. Mục tiêu là đặt được số kẹo nhiều nhất trong tổng toàn bộ số lần đập. Nếu lực tác dụng từ gậy lên bịt kẹo được treo là lớn hay nhỏ mà số lượng kẹo rơi ra sẽ khác nhau. Đồng thời đó độ dao động của bịt kẹo sẽ mạnh hay yếu. Game chơi dựa theo việc tính điểm highscore và người chơi có thể xem điểm highscore đó.

5.4.9 Giao diện game Trắc nghiệm Vật lý



Hình 5.22

Game Trắc nghiệm Vật lý xây dựng nhằm giúp các đối tượng là sinh viên, học sinh kiểm tra kiến thức về Vật lý. Game chơi dễ dàng dựa vào việc tìm ra câu hỏi chính xác trong các câu hỏi, dựa vào quỹ thời gian hiện có mà có thể chọn lựa gói câu hỏi phù hợp, có các gói câu hỏi như: gói 10 câu hỏi, 20 câu hỏi, 30 câu hỏi, 40 câu hỏi, 50 câu hỏi và 60 câu hỏi. Sau khi hoàn thành hết các câu hỏi trả lời đúng sẽ thực hiện tính điểm cho người chơi.

5.5 Cài đặt:

Cấu trúc code chương trình:

Để xử lý những vấn đề đã đặt ra, quá trình code được chia làm các phần:

- Camera2D gồm class điều khiển xử lý Camera2d trong chương trình:
- GameEngine gồm các class:
  - Const: thiết lập các thông số mặc định cho chương trình.
  - Game: khởi tạo và cấp phát các thông số ban đầu cho các đối tượng trong chương trình.

- Program: class hệ thống.

- Objects gồm các class:
  - Momen: xử lý mô phỏng chuyển động của momen động lượng.
  - MomenForce: xử lý mô phỏng chuyển động của momen lực.
  - Pendulum: xử lý mô phỏng con lắc đơn.
  - SpringPendulum: xử lý mô phỏng con lắc lò xo.
  - ThrowOblique: xử lý mô phỏng ném xiên.

Trong Content được chia làm các phần:

- Fonts: thư mục chứa các file Font trong chương trình
- Images: chứa các texture sử dụng trong chương trình.
- Sounds: Chứa các âm thanh được sử dụng trong chương trình.

5.6 Thử nghiệm và đánh giá

Mục đích của chương trình "Mô Phỏng Vật Lý" là hướng tới các đối tượng giáo viên, sinh viên, học sinh có nghiên cứu những chuyên đề liên quan. Nên trong quá trình đưa vào thử nghiệm nhóm đã chọn nhóm sinh viên năm nhất của trường Đại học Công nghệ thông tin hiện đang sống ở khu nhà A9 Kí Túc Xá ĐHQG làm đối tượng thử nghiệm chương trình.

Sau khoảng thời gian thử nghiệm là một tuần nhóm đã nhận được những phản hồi tích cực về chương trình cũng như một số phản hồi về những điểm thiếu sót trong quá trình mô phỏng cũng như giải bài tập.

Từ những đánh giá thực tế, chương trình của nhóm đã được chỉnh sửa để hoàn thiện hơn về chất lượng.

CHƯƠNG 6: KẾT LUẬN

Qua quá trình tìm hiểu về công nghệ XNA 2D và 2D và những kiến thức vật lý cơ bản nhóm đã đạt được những thành quả nhất định:

Nhóm đã tìm hiểu đặc điểm, cấu tạo, cơ chế hoạt động của một số hiện tượng vật lý thường gặp hàng ngày, ứng dụng được công nghệ XNA vào mô phỏng những hiện tượng Vật Lý này như hiện tượng dao động của Con Lắc Đơn, Con Lắc Lò Xo, Ném Xiên, Momen Lực, Momen Động Lượng và một Mô Phỏng Tổng Hợp dựa theo việc xây dựng một game bắn máy bay.

Chương trình đã đáp ứng được những yêu cầu và mục tiêu ban đầu đặt ra:

- Tìm hiểu, nghiên cứu các hiện tượng vật lý, công nghệ XNA 2D, ứng dụng XNA 2D xây dựng chương trình mô phỏng chuyển động của Con Lắc Đơn, Con Lắc Lò Xo, Ném Xiên, Momen quán tính và các game Mô phỏng.
- Chương trình gồm những chức năng:
  - Hiển thị mô phỏng và các thông số cần thiết lên màn hình.
  - Tương tác với hệ thống bằng kéo thả hoặc nhập thông số từ bàn phím.
  - Đọc các thông số tại thời điểm nhất định.
  - Tạm dừng mô phỏng khi mô phỏng đang hoạt động.
  - Tiếp tục quá trình mô phỏng sau khi tạm dừng.
  - Lưu lại thông số chương trình vào một file.
  - Đọc file đã lưu và hiển thị mô phỏng.
  - Giải bài tập theo các chuyên đề.
  - Vừa học vừa giải trí với game tổng hợp các mô phỏng.
  - Giao diện chương trình thân thiện với người dùng
- Mở rộng sáng tạo thêm mục Game Tổng Hợp để tập hợp những gì đã được nghiên cứu. Ý tưởng về Game Tổng Hợp chịu ảnh hưởng từ ý tưởng ban đầu

định hình cho chương trình, và tiến hóa theo suốt thời gian chương trình được xây dựng.

- Quá trình đưa chương trình vào thử nghiệm thực tế đã đạt được những đánh giá tích cực từ người dùng, đáp ứng được mục đích ban đầu đặt ra là ứng dụng phần mềm vào việc giạy và học của giáo viên cũng như sinh viên học sinh có chuyên đề nghiên cứu liên quan.

Hướng phát triển:

- Tích hợp thêm các hiện tượng vật lý khác và xa hơn nữa là các phản ứng hóa học cùng với tích hợp nhiều hiện tượng vật lý trong một mô phỏng.
- Thiết kế lại các mô hình, giao diện phù hợp hơn với người dùng.
- Phát triển game mô phỏng với quy mô lớn hơn.

TÀI LIỆU THAM KHẢO

[1] Nguyễn Trọng (Chủ biên) - Tống Danh Đạo - Lê Thị Hoàng Yến – **Cơ học lý thuyết** - Tập 2. Phần động lực học - NXB: Khoa học kỹ thuật - Xuất bản: 8/2006.

[2] Khoa sư phạm kỹ thuật - Bộ môn Cơ kỹ thuật - **Giáo trình Cơ học lý thuyết** - ĐH Bách Khoa Đà Nẵng.

[3] http://www.codeproject.com/

[4] http://www.hocvi.com

[5] http://www.msdn.microsoft.com/en-us/library/bb200104.aspx

[6] http://www.phet.colorado.edu/en/

[7] http://www.stackoverflow.com/

[8] http://thuvienvatly.com/

[9] http://vatlyphothong.com/