TRƯỜNG ĐẠI HỌC XÂY DỰNG
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN

TỐT NGHIỆP ĐẠI HỌC

NGÀNH CÔNG NGHỆ PHẦN MỀM

PHÁT TRIỂN GAME 3D VỚI OGRE3D TRÊN MÔI TRƯỜNG WINDOWS

Sinh viên thực hiện: **Nguyễn Anh Tuấn**
Mã sinh viên: 183855
Lớp: **55PM1**
Giảng viên hướng dẫn: ThS. Lê Thị Hoàng Anh
& Đoàn Như Tùng

HÀ NỘI 01/2017

LỜI NÓI ĐẦU

Trước hết, em xin chân thành cảm ơn cô, thầy giáo - ThS. Lê Thị Hoàng Anh & Đoàn Như Tùng, giảng viên Khoa Công nghệ thông tin - Trường Đại học Xây Dựng Hà Nội, người đã dành cho em rất nhiều thời gian quý báu, trực tiếp hướng dẫn tận tình giúp đỡ, chỉ bảo em trong suốt quá trình làm đồ án tốt nghiệp.

Em xin chân thành cảm ơn tất cả các thầy cô giáo trong khoa Công nghệ thông tin - Trường Đại học Xây Dựng Hà Nội, chân thành cảm ơn các thầy giáo, cô giáo tham gia giảng dạy và truyền đạt những kiến thức quý báu trong suốt thời gian em học tập tại trường, đã đọc và phản biện đồ án của em giúp em hiểu rõ hơn các vấn đề mình nghiên cứu, để em có thể hoàn thành đồ án này.

Em xin chân thành cảm ơn PGS.TS.NGƯT Phạm Duy Hòa - Bí thư Đảng ủy, Hiệu trưởng Trường Đại học Xây dựng, ban giám hiệu nhà trường, khoa Công nghệ thông tin, các phòng ban nhà trường đã tạo điều kiện tốt nhất trong suốt thời gian em học tập và làm tốt nghiệp. Tuy có nhiều cố gắng trong quá trình học tập và làm đồ án tốt nghiệp nhưng không thể tránh khỏi những thiếu sót nhất định, em rất mong được sự góp ý quý báu của tất cả các thầy cô giáo cũng như tất cả các bạn để đồ án của em ngày càng hoàn thiện hơn. Em xin chân thành cảm ơn!

Hà Nội, Ngày 07/01/2017

Nguyễn Anh Tuấn.

PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

1. Thông tin về sinh viên

Họ và tên sinh viên: Nguyễn Anh Tuấn

Điện thoại liên lạc: 0938187924 Email: bahao247@gmail.com

Lớp: 55PM1 Hệ đào tạo: Chính qui Năm tốt nghiệp: 2017

Đồ án tốt nghiệp được thực hiện tại: Hà Nội

Thời gian làm ĐATN: Từ ngày 26/09/2016 đến 07/01/2017

2. Mục đích nội dung của ĐATN

Tìm hiểu tổng quan về Game engine. Ứng dụng phát triển game 3D với engine Ogre3D trên nền Windows.

3. Các nhiệm vụ cụ thể của ĐATN

1. Tổng quan về Game engine: Tại sao cần phát triển Game engine? Phân loại Game engine.
2. Tìm hiểu Kỹ thuật quản lý khung cảnh LOD trong Ogre3D
3. Cấu trúc phân vùng không gian và các Scene Graphic trong Ogre3D.
4. Kỹ thuật render trong Ogre3D.
5. Ứng dụng phát triển game 3D với engine Ogre3D.

4. Lời cam đoan của sinh viên:

Tôi Nguyễn Anh Tuấn cam kết ĐATN là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của ThS. Lê Thị Hoàng Anh & ThS. Đoàn Như Tùng.

Các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

Hà Nội, ngày 7 tháng 1 năm 2017

Tác giả ĐATN

Nguyễn Anh Tuấn

5. Xác nhận của giảng viên hướng dẫn về mức độ hoàn thành ĐATN và cho phép bảo vệ:

Hà Nội, ngày tháng năm

Giảng viên hướng dẫn

TÓM TẮT
NỘI DUNG ĐỒ ÁN TỐT NGHIỆP

Lập trình game đang là xu hướng phát triển ngày nay của ngành truyền thông và công nghệ thông tin. Với việc dân số tăng không ngừng thì khoảng không để con người vui chơi cũng hẹp dần. Chính vì vậy, game 3D như một lựa chọn giải trí thiết yếu với con người khi muốn tìm một nơi thư giãn, giải trí. Nhất là khi công nghệ phần cứng xử lý hình ảnh ngày càng nâng cao, con người có thể mang theo một thiết bị chơi game ở mọi nơi, mọi chỗ. Cộng với công nghệ Thực tế ảo và Thực tế ảo tăng cường đang được phát triển mạnh trong thời gian gần đây thì thế giới trong game đang rất gần với thực tại và mang lại nhiều lợi ích như giáo dục, du lịch,... không chỉ là giải trí.

Là ngành đem lại lợi nhuận lớn, thúc đẩy nhiều ngành công nghiệp khác phát triển nhưng đây lại là công việc yêu cầu đổ nhiều chất xám, sự sáng tạo và nắm bắt thị trường khiến nhiều lập trình viên khá e ngại khi quyết định theo học. Chính vì vậy, khi được chọn đề tài Đồ án tốt nghiệp em đã nghĩ ngay tới việc xây dựng một nền tảng cơ bản dễ hiểu nhưng đầy đủ về ngành làm game 3D, cũng như cơ hội nghề nghiệp để những lập trình viên hiểu hơn trước khi quyết định theo đuổi công việc yêu cầu nhiều đam mê này.

Cụ thể, đề tài Đồ án tốt nghiệp của em là "Phát triển game 3D với Ogre3D trên môi trường Windows". Đồ án tốt nghiệp này được trình bày theo cấu trúc sau:
- Lời cảm ơn
- Tóm tắt
- Nội dung đồ án
- Phần 1: Đặt vấn đề và đưa ra các giải pháp.
- Phần 2. Các kết quả đạt được
- Phần 3. Kết luận.
- Tài liệu tham khảo.

Từ khóa: Engine game, Lập trình game, Ogre3D, C++.

MỤC LỤC

LỜI NÓI ĐẦU 2
PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP 3
TÓM TẮT 4
NỘI DUNG ĐỒ ÁN TỐT NGHIỆP 4
MỤC LỤC 5
MỤC LỤC HÌNH ẢNH 7
PHẦN NỘI DUNG ĐỒ ÁN 8
Phần 1. Đặt vấn đề và định hướng giải pháp 8
1, Trình bày chi tiết về mục tiêu, nhiệm vụ 8
1.1, Giới thiệu về game 3D 8
1.2, Mục tiêu và nhiệm vụ 9
2, Trình bày định hướng giải quyết các vấn đề đặt ra 9
3, Giới thiệu tóm tắt về cơ sở lý thuyết và công cụ được lựa chọn 9
3.1, Lý thuyết về Game engine 9
3.1.a, Khái niệm 9
3.1.b, Mục đích 9
3.1.c, Lịch sử 10
3.1.d, Ảo hóa phần cứng (trừu tượng hóa phần cứng-hardware abstraction) 11
3.1.e, Xu hướng hiện tại 11
3.1.f, Phần mềm trung gian 11
3.1.g. Game engine cho Game bắn súng góc nhìn người thứ nhất (FPS) 12
3.2, Phân loại Game engine theo mức độ chuyên biệt 12
3.2.a. Roll-your-own Game engine 12
3.2.b. Mostly-ready Game engine 12
3.2.c. Point-and-click Game engine 13
3.3, Game engine và không gian 3d 13
3.3.a, Vec-tơ (Vectors) 13
3.3.b, Camera (Cameras) 13
3.3.c, Đa giác (Polygon), cạnh (Edges), đỉnh (Vertices) và mắt lưới (Meshes) 14
3.3.d, Vật liệu (Materials), kết cấu (Textures) và đổ bóng (Shaders) 14
3.4, Các tính năng Game engine 14
3.4.a, Graphics Egngine 14
3.4.b, Sound Engine 15
3.4.c, Input Engine 15
3.4.d, Physic Engine 15
3.4.e, Góc nhìn trong game 15
3.4.e .1. Góc nhìn thứ nhất 15
3.4.e .2. Góc nhìn thứ 3 16
3.4.e .3. Góc nhìn từ trên xuống 16
3.4.e .4. Góc nhìn ngang 17
3.4.e .5. Góc nhìn kết hợp 17
3.4.f, Others 18
3.4.g, Tools 18
3.5, Công cụ được lựa chọn để giải quyết các vấn đề đặt ra 18
3.5.a, Graphic Game engine: Object-Oriented Graphics Rendering Engine (OGRE) 18
3.5.b, Sound Game engine: irrKlang 19
3.5.c, Object Oriented Input System: OIS 19
3.5.d, GUI System: SdkTrays 19
3.5.e, Other: Boost, TinyXML 20

3.5.f, Tool: Maya3D, 3DMax, Sketchup, Ogre Meshy *20*
Phần 2. Các kết quả đạt được 21
1, Xây dựng Game engine với Ogre3D *21*
1.1, Kỹ thuật quản lý khung cảnh LOD trong OGRE3D *21*
1.1.a, Khái niệm *21*
1.1.b, Ứng dụng thực tế *23*
1.2, Quản lý khung cảnh đồ hoạ (BSP-tree và Octree) *24*
1.2.a, Quản lý khung cảnh đồ hoạ *24*
1.2.b, BSP-tree *24*
1.2.c, Octree/Quad tree *25*
1.2.d, CSG tree (Constructive Solid Geometry tree) *26*
1.2.e, Lồng ghép BSP-tree và Octree trong Scene Graph Management *26*
1.3, Các API chính trong Modules của Core *26*
1.3.a, Animation *26*
1.3.b, Resources *28*
1.3.c, RenderSystem *32*
1.3.d, Scene *35*
1.4, Build SDK *38*
1.4.a, Ogre3D *38*
1.4.b, Cài đặt thư viện âm thanh irrKlang *49*
1.4.c, Cài đặt tool 3ds Max *49*
1.4.d, Cài đặt tool Maya *57*
1.4.e, Cài đặt tool SketchUp *58*
1.4.f, Cài đặt tool Ogre Meshy *65*
1.4.g, Cài đặt tool Scape *65*
1.4.g, Cài đặt tool Visual Studio 2010 ..... *67*
2, Xây dựng kịch bản cho game *68*
2.1, Tổng quan về game *68*
2.1.a, Chủ đề game *68*
2.1.a.1, Tóm tắt cốt truyện *68*
2.1.a.2, Nội dụng game *69*
2.2, Kịch bản game *69*
2.2.a, Game Map *70*
2.2.b, Nhân vật *71*
2.2.c, Vũ khí *74*
2.2.d, Âm thanh *75*
2.2.e, Thiết bị nhận điều khiển *75*
2.2.f, Hiệu ứng *76*
3, Xây dựng game theo kịch bản *78*
3.1, Viết Class load âm thanh (MyFileFactoryCommon.h) *78*
3.2, Tạo bảng Common (SdkTableCommon.h) *80*
3.3, Viết Class MaterialControls cho địa hình dưới nước (MaterialControlsCommon.h) . *82*
3.4, Xây dựng Content cho game (Terrain.h) *89*
3.5, Viết Class SinbadCharacterController tạo nhân vật (SinbadCharacterControllerCommon.h) *104*
4, Cài đặt và trình bày về sản phẩm game *142*
4.1, Cài đặt game *142*
5, Đánh giá ưu, nhược điểm của sản phẩm game *153*
5.1, Ưu điểm *153*
5.2, Nhược điểm *153*
Phần 3. Kết luận 154
TÀI LIỆU THAM KHẢO 155

MỤC LỤC HÌNH ẢNH

Hình 1: Graphic Game engine có ứng dụng nhiều trong game 18
Hình 2: irrKlang là thư viện hỗ trợ phát âm thanh 2D và 3D 19
Hình 3: LOD sẽ giảm mức chi tiết cho các đối tượng ở xa, quá nhỏ 21
Hình 4: LOD giúp nâng cao hiệu suất render hình ảnh trong đồ hoạ 3D 23
Hình 5: Hiệu suất reder hình ảnh vượt trội so với BRUTE 23
Hình 6: Octree quản lý các nút theo phân vùng 25
Hình 7: Sự kết hợp giữa BSP-tree và Octree trong Scene Graph Management giúp nâng cao hiệu suất hiển thị hình ảnh 26
Hình 8: Level 1- Thach Sanh Banian-tree 70
Hình 9: Level 2- Cut python 70
Hình 10: Level 3- Shoot eagle 71
Hình 11: Level 4- Tich tich tinh tang 71
Hình 12: Level 5- Rhythm of the justice 71
Hình 13: Thạch Sanh 72
Hình 14: Công chúa Quỳnh Nga 72
Hình 15: Bot nhỏ 1 72
Hình 16: Bot nhỏ 2 73
Hình 17: Bot nhỏ 3 73
Hình 18: Đại bàng tinh 73
Hình 19: Chằn tinh 74
Hình 20: Song đao 74
Hình 21: Nồi cơm 74
Hình 22: Cung tên vàng 74
Hình 23: Đàn thần 75
Hình 24: Rìu đốn củi 75
Hình 25: Bàn phím và chuột 76
Hình 26: Hào quang của vũ khí 76
Hình 27: Hào quang nhân vật 77
Hình 28: Khói lửa 77
Hình 29: Pháo hoa 78
Hình 30: Chọn công nghệ Render hình ảnh 147
Hình 31: Vào giao diện menu game 148
Hình 32: Bắt va chạm trong game 153

PHẦN NỘI DUNG ĐỒ ÁN

Phần 1. Đặt vấn đề và định hướng giải pháp

1, Trình bày chi tiết về mục tiêu, nhiệm vụ

1.1, Giới thiệu về game 3D

Đơn giản có thể hiểu xây dựng một game 3D với dây chuyền chế tạo một chiếc xe ôtô và động cơ chính là Game engine.

Động cơ giúp ôtô chuyển hóa năng lượng động lực vào trục xe khiến bánh lăn giúp xe chuyển động. Người lái xe giống như người chơi game sẽ chỉ thấy bên trong chiếc xe là ghế ngồi, đồng hồ hiển thị, các nút bấm điều khiển, vô lăng,... chứ không quan tâm tới bên trong chiếc xe hoạt động ra sao.

Vậy việc xây dựng game 3D có thể liên tưởng một cách thú vị với dây chuyền sản xuất ôtô như sau:

- Phân tích thị trường, chọn cho mình việc sản xuất sản phẩm thế mạnh: Với ôtô là dòng xe thể thao, du lịch, sang trọng, bình dân... từ đó tính toán để chọn bản thiết kế thích hợp cộng với chi phí, thời gian sản xuất. Làm game cũng vậy ta cần tính toán tới thị hiếu của người chơi, đối tượng hướng tới để chọn thể loại game sẽ làm như giải đố, game run, game phiêu lưu, RPG..., giá thành và thời gian sản xuất để chọn bản thiết kế thích hợp cho game.

- Chọn động cơ và các module theo kèm: Động cơ chính là quả tim của chiếc xe, nó quyết định sức mạnh và những module theo kèm. Đó là hệ thống truyền lực, hệ thống điều khiển với các mạch nhúng, hệ thống âm thanh, điều hòa, giải trí... Làm game cũng tương tự Game engine chiếm vị trí quan trọng tương tự với trung tâm là thư viện chuyển hóa hình ảnh 3D quyết định sự chân thực của hình ảnh. Cũng như động cơ xe thì Game engine cũng phép gắn kết các module ngoài hay gắn trực tiếp vào engine đó là thư viện xử lý va chạm, hiệu ứng vật lý, âm thanh, hiệu ứng kỹ xảo, thư viện hỗ trợ điều khiển, thư viện hỗ trợ giao diện sử dụng...

- Thiết kế và sản xuất: Sau khi chọn được động cơ thứ quyết định của sức mạnh của chiếc xe và những chức năng mà xe có thể hỗ trợ thì việc thiết kế hình dáng, lập trình các hệ thống điều khiển chức năng trên game sẽ được bám sát vào bản phân tích thị trường và những chức năng chiếc xe có thể hỗ trợ để đưa ra bản thiết kế thích hợp. Cuối cùng, lắp ghép vỏ xe, bánh xe, động cơ, các module khác, bộ điều khiển mạch nhúng đã được lập trình điều khiển các chức năng trên xe. Cuối cùng, gắn bộ điều khiển vô lăng và các nút điều khiển các chức năng của xe. Làm game cũng tương tự, đó là việc chọn ra kiểu chơi cho game, viết kịch bản game, thiết kế đồ họa, âm thanh, effect, giao diện sử dụng... Sau khi sản xuất xong lắp ghép các module nhỏ vào game thông qua các engine hỗ trợ. Tiếp đó, viết các hàm chức năng theo kịch bản game.

- Quảng cáo và đưa ra thị trường: Sau khi sản xuất xong một dòng xe việc chọn ra một slogan và chiến dịch PR được coi là điều tối quan trọng tới sự thành công của sản phẩm. Việc ra mắt game cũng vậy thường kéo dài cả tháng trời với những chiến dịch ưu đãi quà tặng cực lớn hay quảng cáo trên các kênh truyền thông để thu hút game thủ.

1.2, Mục tiêu và nhiệm vụ

- Nắm được căn bản về Game engine.
- Tìm hiểu sâu về graphic Game engine Ogre3D và các thư viện hỗ trợ làm để phát triển game.
- Ứng dụng kiến thức tìm hiểu được phát triển game 3D chạy trên nền Windows.

2, **Trình bày định hướng giải quyết các vấn đề đặt ra**

- Tìm hiểu lý thuyết về Game engine.
- Nghiên cứu các thư viện của Game engine Ogre3D và những thư viện bổ trợ.
- Thực hành xây dựng game theo các hướng dẫn của trang Ogre3D và nguồn Internet.

3, **Giới thiệu tóm tắt về cơ sở lý thuyết và công cụ được lựa chọn**

3.1, Lý thuyết về Game engine

3.1.a, Khái niệm

Một Game engine là một phần mềm được viết để thiết kế và phát triển video game, hiểu đơn giản nó là loại phần mềm trung gian kết nối tương tác của nhiều ứng dụng trong cùng 1 hệ thống với nhau. Có rất nhiều loại Game engine dùng để thiết kế game cho các hệ máy như hệ consoles hay máy tính cá nhân (PC). Chức năng cốt lõi của Game engine phần lớn nằm trong công cụ dựng hình (kết xuất đồ họa) cho các hình ảnh 2 chiều (2D) hay 3 chiều(3D), công cụ vật lý (hay công cụ tính toán và phát hiện va chạm), âm thanh, mã nguồn, hình ảnh động (animation), trí tuệ nhân tạo, phân luồng, tạo dòng dữ liệu xử lý, quản lý bộ nhớ, dựng ảnh đồ thị, và kết nối mạng. Quá trình phát triển game tiết kiệm được rất nhiều thời gian và kinh phí vào việc tái sử dụng và tái thích ứng một engine để tạo nhiều game khác nhau.

3.1.b, Mục đích

Game engine cung cấp một bộ các công cụ phát triển trực quan và có thể tái sử dụng từng thành phần trong đó. Nói chung các bộ công cụ này cung cấp một môi trường phát triển tích hợp được đơn giản hóa. Phát triển ứng dụng nhanh (tiếng Anh:rapid application development) cho game theo cách lập trình hướng dữ liệu. Những Game engine này đôi khi còn được gọi là các "phần mềm trung gian cho game"(game middleware), như ý nghĩa của thật ngữ, chúng cung cấp một nền tảng phần mềm linh hoạt và dễ dàng sử dụng lại với mọi chức năng cốt lõi cần thiết ngay trong nó để có thể phát triển một ứng dụng game đồng thời giảm giá thành, độ phức tạp, và

kịp thời hạn phát hành- tất cả các yếu tố quan trọng trong ngành công nghiệp game đầy cạnh tranh.
Giống như các phần mềm trung gian khác, Game engine thường cung cấp một nền tảng trừu tượng hóa, cho phép một game có thể chạy trên nhiều hệ máy bao gồm các hệ console hoặc máy tính cá nhân với một vài, nếu có, thay đổi trong mã nguồn của game đó. Thông thường, phần mềm trung gian cho game được thiết kế với một nền tảng kiến trúc dựa trên các thành phần khác, cho phép các hệ thống khác nhau trong engine có thể thay thế hoặc mở rộng với các phần mềm trung gian khác chuyên biệt hơn (và cũng đắt đỏ hơn) như là Havok cho hệ thống vật lý trong game, Miles Sound System cho âm thanh, hay Bink cho các đoạn video. Một số Game engine như RenderWare thậm chí còn thiết kế như một loạt các kết nối lỏng lẻo các phần mềm trung gian khác mà từ đó có thể chọn lọc và kết hợp để tạo ra một Game engine theo yêu cầu, thay vì cách tiếp cận thông thường là mở rộng và tùy biến một cách linh hoạt để tạo ra giải pháp tích hợp. Tuy vậy, sự mở rộng vẫn cần thiết, đó vẫn là một ưu tiên cao do yêu cầu ứng dụng đa dạng trong Game engine. Mặc dù có sự phức tạp trong tên gọi, các Game engine vẫn thường được sử dụng cho các loại khác của ứng dụng tương tác với các yêu cầu đồ họa thời gian thực như giới thiệu các bản demo, dựng hình kiến trúc, đào tạo mô phỏng, và mô hình hóa môi trường. Một số Game engine chỉ cung cấp khả năng dựng hình (kết xuất) 3D thời gian thực thay vì rất nhiều chức năng trong phạm vi rộng mà game yêu cầu. Loại engine này tùy thuộc vào các nhà phát triển, sử dụng để thực hiện phần chức năng hoặc kết nối phần hình ảnh còn lại từ các Game engine khác nhằm mục đích cải thiện hình ảnh cho game. Loại engine này thường được gọi là: "graphics engine", "rendering engine," hay "3D engine" thay vì thuật ngữ bao quát hơn là "Game engine". Thuật ngữ không nhất quán này sử dụng với nghĩa một bộ công cụ đầy đủ chức năng để tạo hình ảnh 3 chiều trong game chứ không đơn giản chỉ là "3D engine". Một vài ví dụ cho các engine đồ họa là: RealmForge, Truevision3D, OGRE, Crystal Space, Genesis3D, Vision Engine, Irrlicht và JMonkey Engine. Game hoặc engine đồ họa ngày nay thường cung cấp bởi một trường đồ thị dựng sẵn, khiến việc biểu diễn hướng đối tượng của thế giới game 3D trong thiết kế được đơn giản hóa và có thể làm cho việc kết xuất thế giới ảo rộng lớn một cách hiệu quả hơn.

3.1.c, Lịch sử

Thuật ngữ "Game engine" xuất hiện vào giữa những năm 90, đặc biệt là trong mối quan hệ giữa Game 3D và Game bắn súng góc nhìn người thứ nhất (FPS). Như các thương hiệu nổi tiếng của id Sortware: Doom và Quake, thay vì phải làm việc từ đầu, các nhà phát triển khác (nếu được cấp phép) sẽ có quyền truy nhập vào phần lõi (mã nguồn) của Game và thiết kế các hình ảnh, nhân vật, vũ khí, và các màn chơi của riêng họ - gọi là Game Content (nội dung Game) hay "Game Assets" (tài sản Game). Các Game sau này, như Quake III Arena và sản phẩm năm 1998 của Epic Games: Unreal được thiết kế với cách tiếp cận mới này, Game engine và nội dung Game được chia thành các phần riêng biệt để phát triển. Engine có thể tái sử dụng khiến việc phát triển Game tiếp theo nhanh hơn và dễ dàng hơn, một thuận lợi to lớn trong ngành công nghiệp cạnh tranh này. Game engine hiện đại là một trong những ứng dụng được viết ra (bằng các ngôn ngữ lập trình) phức tạp nhất, thường

xuyên phải có rất nhiều tinh chỉnh trong hệ thống để đảm bảo kiểm soát chính xác trải nghiệm người dùng. Sự phát triển liên tục của Game engine đã tạo ra một sự phân chia mạnh mẽ giữa các công việc dựng hình, viết kịch bản, thiết kế hình ảnh, và thiết kế màn chơi. Hiện nay thông thường một đội ngũ phát triển Game điển hình phải có số lượng họa sĩ gấp vài lần số lượng lập trình viên.

3.1.d, Ảo hóa phần cứng (trừu tượng hóa phần cứng-hardware abstraction)

Thông thường, engine 3D hay hệ thống kết xuất đồ họa trong Game engine thường được xây dựng trên một nền tảng giao diện lập trình ứng dụng đồ họa như Direct3D hay OpenGL cung cấp sẵn một hệ ảo hóa cho GPU hay card đồ họa. Thư viện mã nguồn cấp thấp (cấp cơ bản) như DirectX, SDL, và OpenAL (thư viện âm thanh mã nguồn mở) cũng thường được sử dụng trong các game khi chúng cung cấp khả năng truy cập phần cứng độc lập với phần cứng máy tính khác như các thiết bị đầu vào (chuột, bàn phím, và tay cầm điều khiển), card mạng, và card âm thanh, trước cả bộ điều hợp đồ họa 3D, phần mềm dựng hình được sử dụng. Phần mềm dựng hình vẫn được sử dụng trong một số trường hợp như là công cụ mô hình hóa hoặc kết xuất ảnh liên tục khi cần hình ảnh chính xác hơn là biểu diễn theo thời gian thực (đo bằng số khung hình/giây) hoặc trong trường hợp phần cứng máy tính không đủ mạnh để hỗ trợ hình ảnh như hỗ trợ tô bóng(đổ bóng-shader)
Với sự ra đời của các phần cứng tăng tốc xử lý vật lý, các giao diện lập trình ứng dụng vật lý như PAL và hệ vật lý mở rộng như COLLADA (một định dạng chuyển đổi cho các loại ảnh 3D) trở nên sẵn có để cung cấp một phần mềm ảo hóa cho vi xử lý vật lý của các nhà cung cấp phần mềm trung gian và hệ máy platform.

3.1.e, Xu hướng hiện tại

Với công nghệ tạo Game engine càng phát triển và trở nên thân thiện hơn với người sử dụng, ứng dụng của nó càng được mở rộng, và giờ đây được sử dụng để tạo các Game mang mục đích khác với giải trí đơn thuần như: mô phỏng, huấn luyện ảo, y tế ảo, và mô phỏng các ứng dụng quân sự. Nhiều Game engine đang được tạo ra bằng các ngôn ngữ lập trình cấp cao như Java và C# hay .NET (ví dụ: TorqueX, và Visual3D.NET) hay Python (Panda3D). Vì hầu hết các Game 3D hiện nay đều có giới hạn cho GPU (giới hạn bởi sức mạnh của card đồ họa), khả năng gây chậm máy của các ngôn ngữ lập trình cấp cao trở nên không đáng kể, trong khi việc tăng năng suất được cung cấp bởi các ngôn ngữ này lại có lợi cho các nhà phát triển Game engine.

3.1.f, Phần mềm trung gian

Một số hãng hiện nay chuyên tập trung vào phát triển các bộ phần mềm gọi là phần mềm trung gian (Middleware). Các nhà phát triển Middleware cố gắng phát triển một bộ phần mềm mạnh mẽ bao gồm hầu hết các yếu tố mà một nhà phát triển Game cần đến để xây dựng một Game. Phần lớn các Middleware cung cấp các cơ sở để phát triển game một cách dễ dàng, như hình ảnh, âm thanh, hệ thống vật lý và chức năng AI. Gamebryo và RenderWare là hai phần mềm trung gian được sử dụng khá rộng rãi. Một số Middleware chỉ làm một chức năng nhưng nó còn hiệu quả hơn cả toàn bộ Engine.

Ví dụ, SpeedTree được sử dụng để dựng hình ảnh cây cối thêm thực tế trong Game nhập vai The Elder Scrolls IV: Oblivion. Bốn gói phần mềm trung gian được sử dụng rộng rãi nhất, thường được cung cấp dưới dạng một chức năng bổ sung trong một hệ thống Engine gồm có Bink của hãng RAD Game Tools, hãng Firelight với FMOD, Havok, và GFx của Scaleform. RAD Game Tools phát triển Bink cho dựng hình video cơ bản, đi cùng với âm thanh Miles, và dựng hình 3D Granny . FMOD của Firelight là một thư viện và bộ công cụ phần mềm âm thanh mạnh mẽ giá rẻ. Havok cung cấp một hệ thống mô phỏng vật lý mạnh, cùng với các bộ phần mềm về hình ảnh động và mô phỏng hành vi. Scaleform đem lại GFx dùng cho các biểu diễn cao cấp trên nền giao diện Flash, cùng với các giải pháp trình chiếu video chất lượng cao, và một Input Method Editor (IME) phần bổ sung nhằm hỗ trợ trò chuyện trong Game bằng ngôn ngữ Châu Á.

3.1.g. Game engine cho Game bắn súng góc nhìn người thứ nhất (FPS)

Một trong các loại Game engine được biết đến nhiều nhất là Game engine cho các Game FPS. Là bước đột phá của các nhà phát triển về chất lượng hình ảnh khi thực hiện các Game FPS trên góc nhìn của con người. Trong khi các Game mô phỏng như điều khiển máy bay hay xe cộ và Game chiến thuật thời gian thực (RTS) tăng cường tính thực tế trong hình ảnh ở mức độ vĩ mô, thì Game bắn súng góc nhìn thứ nhất lại đi đầu về xử lý hình ảnh thực tế ở cấp độ nhỏ hơn.

3.2, Phân loại Game engine theo mức độ chuyên biệt

3.2.a. Roll-your-own Game engine

Roll-your-own Engine có thể được xem như những Engine ở mức thấp nhất. Ngày nay rất nhiều công ty tự tạo ra Engine cho riêng họ với những thứ có thể. Họ tận dụng những gói ứng dụng phổ biến, các API như XNA, DirectX, OpenGL, các API và SDL của Windows và Linux. Ngoài ra, họ có thể dùng những thư viện từ nhiều nguồn khác nhau, được mua hoặc Open-Source. Những thư viện đó có thể bao gồm cả những thư viện vật lý như Havok và ODE, những thư viện Scene-Graph như OpenSceneGraph và các thư viện GUI như AntTweakBar. Thực tế, những hệ thống "tự sản xuất" đó cho phép những lập trình viên dễ tương tác hơn rất nhiều vì họ biết cần gì và chọn những thành phần phù hợp từ đó tạo nên chính xác thứ cần thiết. Nhưng cũng vì vậy mà tốn rất nhiều thời gian. Thêm vào đó, những lập trình viên thường gặp rất nhiều lỗi khi các thư viện từ nhiều nguồn không tương thích với nhau. Điều này đã góp phần làm cho những Engine dạng này kém hiệu quả và không được ưa chuộng ngay cả với Engine được thiết kế một cách chuyên nghiệp.

3.2.b. Mostly-ready Game engine

Mostly-ready Game engine là Engine nằm ở tầng trung. Các nhà thiết kế Game nghiệp dư rất thích những Engine dạng này. Chúng được thiết kế rất đầy đủ với mọi tính năng cần thiết, vẽ, tương tác, GUI, vật lý… ngoài ra chúng còn kèm theo rất nhiều tool mạnh. Những Engine tầng này khá phong phú từ ORGE và Genesis3D (open-source), rẻ tiền như Torge hoặc đắt tiền như Unreal, id Tech và Gamebryo. Về mức độ, những Engine kể trên đều cần lập trình thêm để gắn kết mọi thứ với

nhau nhằm tạo nên một Game hoàn chỉnh. Hầu như mọi Mostly-ready Game engine đều có phần hạn chế một ít so với roll-your-own Engine. Vì mặc dù nó là sản phẩm của rất nhiều người tham gia phát triển trải qua thời gian dài nhằm mang đến cho khách hàng những "tiện nghi" tốt nhất nhưng không phải khách hàng nào cũng giống nhau và phát triển những Game tương tự nhau.

3.2.c. Point-and-click Game engine

Point-and-click Engine được xếp ở tầng cao nhất. Những Engine này ngày càng trở nên phổ biến. Nó có tất cả mọi tool với đầy đủ chức năng cho phép chúng ta chỉ việc chọn, click và chọn để tạo ra Game. Những Engine như vậy, kể cả GameMaker, Torque Game Builder, và Unity3D được thiết kế để càng thân thiện với người dùng càng tốt. Nó cũng yêu cầu ít lập trình hơn. Điều này không có nghĩa việc lập trình không cần, nhưng nếu so với 2 loại Engine kể trên trên thì loại Engine này cần ít nhất. Nhược điểm rất lớn của những Engine point-and-click này đến từ việc những nhà phát triển bị hạn chế rất nhiều, vì với loại Engine này, chúng chỉ có thể cho ra một loại giao diện như tương tự nhau hoặc khác nhau một ít. Nhưng điều này không có nghĩa chúng vô dụng. Vì khi đối mặt với những hạn chế đó, những sáng tạo của con người sẽ càng được kích thích cao. Ngoài ra, Engine này cho phép chúng ta có thể làm việc nhanh hơn, làm ra Game nhanh hơn mà không phải làm quá nhiều.

Ngoài ra, các Engine có thể phân loại theo giấy phép bản quyền: Unreal Engine, Unity- Proprietary (Bản quyền), Torque3D, Ogre3D- MIT (Miễn phí),...; theo ngôn ngữ sử dụng: C++, Java, C#, Python, Javascript, HTML5,...; theo có hỗ trợ 2D/3D hay ứng dụng có chạy Cross-platform hay không.

3.3, Game engine và không gian 3d

3.3.a, Vec-tơ (Vectors)

Chúng ta cũng sẽ có khái niệm vector 3D trong hệ trục toạ độ Cartesian. Cũng giống như vector trong 2D, vector 3D đơn giản là một đường thẳng trong không gian 3D, có hướng và độ dài. Vector có thể di chuyển trong không gian 3D. Vector rất hữu ích trong Game engine, vì chúng cho phép chúng ta tính toán khoảng cách, góc giữa các đối tượng và hướng của đối tượng.

3.3.b, Camera (Cameras)

Camera là một đối tượng rất cần thiết trong thế giới 3D, chức năng của chúng hoạt động như một khung nhìn cho màn hình. Có một tầm nhìn theo hình kim tự tháp, camera có thể được đặt ở bất kỳ điểm nào trong thế giới 3D, được diễn hoạt, hoặc gắn vào nhân vật hay các đối tượng như là một phần của kịch bản trò chơi. Với khả năng điều chỉnh tầm nhìn (Field of Vision - FOV), 3D Camera là tầm nhìn của chúng ta trong không gian 3D. Trong Game engine, chúng ta nhận thấy những hiệu ứng như ánh sáng, chuyển động mờ (Motion Blurs), và những hiệu ứng khác được áp dụng cho Camera để giúp Game mô phỏng đôi mắt của người xem trong thế giới 3D đó, chúng ta thậm chí có thể thêm một vài hiệu ứng điện ảnh mà mắt người thật không bao giờ nhận thấy được như tia sáng (Lens Flares) khi nhìn vào mặt trời. Hầu

hết các 3D Game hiện đại sử dụng nhiều Camera để hiển thị các phần của thế giới trong trò chơi, Unity thực hiện điều này một cách dễ dàng bằng cách cho phép có nhiều Camera trong một cảnh duy nhất, điều này có thể được viết mã điều khiển Camera tại bất kỳ điểm nào trong suốt thời gian chạy. Nhiều camera có thể được sử dụng trong trò chơi để điều khiển việc kết xuất những đơn vị hình ảnh 2D và 3D riêng biệt như là một phần của quá trình tối ưu. Ví dụ, đối tượng có thể được gom nhóm trong trong một lớp, và Camera được chỉ định để kết xuất hình ảnh của đối tượng trong một lớp riêng biệt. Việc này cho phép chúng ta kiểm soát nhiều hơn đối với từng cá nhân của một đối tượng nhất định trong Game.

3.3.c, Đa giác (Polygon), cạnh (Edges), đỉnh (Vertices) và mắt lưới (Meshes)

Trong dựng hình 3D, tất cả các đối tượng cuối cùng đều được nối với nhau bởi những hình ảnh 2D được biết với các tên "đa giác" (Polygons). Khi nhập mô hình (Model) từ những ứng dụng dựng hình (Modelling Application), Unity chuyển đổi tất cả các đa giác thành tam giác. Tam giác (còn được gọi là mặt) được tạo nên từ 3 cạnh kết nối với nhau. Vị trí mà các cạnh gặp nhau gọi là điểm hay đỉnh. Khi xác định được các vị trí này, Game engine có thể thực hiện tính toán sự va chạm (Collisions), … Trong các dự án Game, một điều rất quan trọng đó là các nhà phát triển phải hiểu được tầm quan trọng của số lượng đa giác. Số lượng đa giác là tổng số lượng các đa giác tạo nên Model, càng nhiều đa giác thì càng có nhiều công việc mà máy tính phải làm để kết xuất hình ảnh của đối tượng lên màn hình.

3.3.d, Vật liệu (Materials), kết cấu (Textures) và đổ bóng (Shaders)

Vật liệu là một khái niệm rất quen thuộc trong tất cả các ứng dụng 3D, chúng cung cấp phương tiện để thiết lập sự xuất hiện (Visual Appearence) của một mô hình 3D. Từ những màu sắc cơ bản đến các bề mặt phản chiếu hình ảnh, vật liệu xử lí tất cả mọi thứ. Vật liệu có thể là một màu sắc cơ bản hay bao gồm một hoặc nhiều hình ảnh được gọi là Textures (kết cấu). Shaders là những mã kịch bản phụ trách phong cách kết xuất hình ảnh.

3.4, Các tính năng Game engine

Các tính năng chính của Game engine bao gồm Graphics Egngine, Sound Engine, Input Engine, Physic Engine, Others, Model Editor.

3.4.a, Graphics Egngine

- Khái niệm:
+ Là thành phần quan trọng nhất của Game engine.
+ Xử lý quá trình render trong game.
+ Cung cấp các API giúp việc lập trình trở nên đơn giản và có cấu trúc chặt chẽ hơn.
- Các graphics engine đôi khi được coi là Game engine.
– Các tính năng:
+ Quản lý khung cảnh: Quản lý khung cảnh đồ hoạ- (SceneGraph, BSP-tree, Octree, CSG tree, Quad tree); Kỹ thuật quản lý khung cảnh đồ hoạ- LOD.

+ Đổ bóng và ánh sáng: Per - vertex (Quản lý ánh sáng render lên bề mặt đồ vật theo theo vector) và per - pixel lighting(Phong- Quản lý ánh sáng render lên bề mặt đồ vật theo từng pixel với độ mịn hơn so với Per - vertex cũng như hiển thị rõ độ nông- sâu của ánh sáng chiếu lên); Shadow mapping (xử lý hình ảnh bóng đổ trên bề mặt map); Radiosity (quản lý hiển thị tại bề mặt phản xạ ánh sáng và đổ bóng); Volumetric (Hiển thị hình ảnh 3D thông qua sự phát xạ, tán xạ, hoặc chuyển tiếp của ánh sáng từ các khu vực được xác định rõ trong không gian);
+ Render: GUI Rendering (Tạo giao diện GUI để tương tác với game); Render to texture (Cập nhật textures theo thời gian thực); Ray tracing (Kỹ thuật để tạo ra một hình ảnh bằng cách lần theo đường đi của ánh sáng qua các điểm ảnh trong một mặt phẳng ảnh và mô phỏng hiệu ứng như phản xạ và khúc xạ, tán xạ và hiện tượng tán sắc của ảnh lên mặt phẳng đó).
+ Một số các tính năng hỗ trợ khác như load mô hình, animation, địa hình và một số các hiệu ứng.

3.4.b, Sound Engine

– Các tính năng:
+ Hỗ trợ trong việc phát các nhạc nền và
các hiệu ứng âm thanh
+ Hỗ trợ âm thanh 2D và 3D
+ Tối ưu hóa dữ liệu và đa luồng
Ví dụ về sound engine: OpenAL, FMOD, IrrKlang...

3.4.c, Input Engine

– Khi chơi game, người chơi thường sử dụng bàn phím khác khi sử dụng thông thường
– Bên cạnh bàn phím, người chơi còn có thể sử dụng các thiết bị khác như gamepad, joystick, Wii mote, Kinect.

3.4.d, Physic Engine

+ Hầu hết các game dù đơn giản đều sử dụng mô phỏng vật lý.
+ Lập trình game càng phát triển thì càng nhiều các mô phỏng vật lý phức tạp
Các ví dụ:
+ Collision detection
+ Rigid body
+ Rag doll
+ Vehicle
+ Soft body
+ Collision

3.4.e, Góc nhìn trong game

3.4.e .1. Góc nhìn thứ nhất

Đây là góc nhìn được áp dụng từ rất sớm, hầu hết được các nhà phát triển sử dụng chính cho các tựa game bắn súng (Hay còn gọi là FPS). Dưới con mắt của góc nhìn

của người chơi, bạn sẽ có cảm giác như được chính mình trực tiếp trải nghiệm và tham gia trực tiếp vào những trận chiến khốc liệt trong các trò chơi bắn súng.

Trải qua một thời gian khá dài, góc nhìn thứ nhất này còn được nhiều Studio sử dụng rộng rãi hơn khi áp dụng vào các tựa game Stealth Action (Thief, Dishonored,...) hay RPG (Skyrim), và nó nhanh chóng trở thành cơn sốt khi hầu hết các tựa game này đều được game thủ đón nhận một cách hết sức nhiệt tình bởi góc nhìn và lối chơi mới lạ.

Tuy nhiên, tuy mang tính đến tính trải nghiệm và thực tế cao hơn, nhưng góc nhìn người chơi ở FPC này rất hạn chế, điều này sẽ khiến bạn phải quay chuột, màn hình nhiều hơn để làm quen với môi trường trong game. Với nhiều game được gắn nhiều hình ảnh hay hiệu ứng phức tạp, một số người chơi sẽ cảm thấy khó chịu và buồn nôn sau khi trải nghiệm một thời gian dài.

Trải nghiệm game ở góc nhìn thứ nhất quả thật khó và thú vị hơn rất nhiều

Một số tựa game góc nhìn thứ nhất đình đám trên thị trường : Dishonored, Far Cry 4, Call of Duty, Đột kích, Warface, Skyrim,...

3.4.e .2. Góc nhìn thứ 3

Không phải là mới, nhưng có thể nói khi đồ họa Game đã có sự cải tiến vượt bậc hơn trước, góc nhìn thứ 3 mới chính thức được nhiều game thủ thật sự ưa chuộng. Dĩ nhiên, chả ai lại muốn ngắm nhìn bối cảnh, môi trường xung quanh mình toàn là một đống Pixel được sắp xếp loạn xạ xấu xí. Góc nhìn thứ 3, bạn có thể hiểu đơn thuần là đặt Camera ở vị trí trước hoặc sau nhân vật chính trong game, cùng nhìn nhân vật của mình đi đến mọi nẻo đường và lần lượt hoàn thành các nhiệm vụ.

So với góc nhìn thứ nhất, góc nhìn thứ 3 có độ rộng nhiều hơn, người chơi có thể dễ dàng chiêm ngưỡng được hết những kỹ năng của nhân vật mình, trang bị vũ khí, Skin để tăng sức hấp dẫn, cũng như biết trước được bối cảnh xung quanh để dễ dành tránh né, đưa ra các tình huống xử trí kịp thời. Tuy nhiên, nó cũng khiến rất nhiều nhà phát triển phải đau đầu vì buộc phải đặt Camera sao cho hợp lý, khiến người chơi thoải mái và dễ dàng quan sát nhất có thể vì chỉ một vài sai lầm nhỏ về góc nhìn cũng khiến game nhận được một số điểm tồi tệ.

Góc nhìn thứ 3 này được sử dụng rộng rãi ở các tựa game phiêu lưu, bắn súng, RPG,...Một trong những nhà phát triển thành công nhất trong việc sử dụng góc nhìn này không ai khác ngoài Ubisoft, họ đã sử dụng rất tinh tế, biến những góc quay của nhân vật mình thành hoàn hảo nhất khiến nhiều người chơi cực kỳ hài lòng.

Một số tựa game góc nhìn thứ 3 dình đám trên thị trường: Assassin's Creed, Tomb Raider, Resident Evil,...

3.4.e .3. Góc nhìn từ trên xuống

Hay còn gọi là Top-Down Perspective, một khái niệm chắc hẳn cực kỳ lạ lẫm với nhiều người chơi, kể cả những game thủ lâu năm. Nhưng thực chất, đây có thể nói là một trong những "ông tổ" của các Camera hiện tại khi hầu hết những tựa game từ xưa đã sử dụng nó. Để giải thích một cách đơn giản nhất có thể, nó chỉ đơn thuần là một góc nhìn từ trên xuống xuống, bạn có thể quan sát toàn bộ một bề mặt hoặc đội quân của mình và điều khiển chúng một cách cực kỳ dễ dàng.

Nói đến đây bạn sẽ nghĩ đến ngay những tựa game dàn trận, chiến thuật trên thị trường hiện tại. Song khái niệm này đến thời điểm hiện tại đã không còn được sử dụng rộng rãi do bị biến thể quá nhiều, hầu hết các tựa game có góc nhìn từ trên xuống bây giờ không còn bị gò bó như trước, bạn có thể dễ dàng kéo màn hình để thay đổi sang một góc nhìn khác dễ dàng hơn rất nhiều.

Nhìn chung thì Top-Down Perspective chỉ phù hợp với những ai muốn trải nghiệm sử dụng nhiều đầu óc hơn, cần nhiều góc nhìn để đưa ra chiến thuật hợp lý hơn thôi chứ không hẳn được hầu hết các game thủ ưa chuộng.

Một số tựa game có góc nhìn từ trên xuống đình đám trên thị trường hiện tại tuy có bị biến thể ít nhiều nhưng không thể không kể đến Liên minh huyền thoại, Dota 2, Age of Empires, Total War, FIFA, PES...

3.4.e .4. Góc nhìn ngang

Hay còn gọi với cái tên quen thuộc là Side Scrolling, cũng tương tự như Top-Down, game màn hình ngang đã gắn bó với nhiều game thủ Việt ngay từ khi còn mò mẫm trên hệ máy NES hay phổ biến hơn là Arcade (Điện tử cắc). Game màn hình ngang dễ chơi, phù hợp với nhiều đối tượng hơn từ nhỏ tuổi đến cả những người đã đứng tuổi, hầu hết đều được thiết kế khá đơn giản và không yêu cầu nhiều kỹ thuật như góc nhìn thứ 3 hay thứ 1.

Ở thị trường hiện tại, game màn hình ngang đang ngày càng trở lại và khẳng định về thế của mình hơn, chứng minh sức hấp dẫn không bao giờ dập tắt và khơi lại nhiều kỷ niệm cho người chơi khiến nhiều họ cực kỳ thích thú. Khỏi phải lấy ví dụ minh họa, khi nhắc đến game màn hình ngang chắc hẳn ai cũng nghĩ tới những tựa game song đấu hấp dẫn hay những trò chơi phiêu lưu nhỏ mà hay được thiết kế đơn giản nhưng ẩn chứa một cốt truyện sâu sắc. Ubisoft tuy không phải mạnh về mảng này, nhưng những tựa game gần đây của hãng đều gặt hái không ít thành công.

Một số tựa game có góc nhìn màn hình ngang đình đám trên thị trường hiện tại chắc chắn là những cái tên quen thuộc như Valiant Hearts: The Great War, Shank, Child of Light, Limbo, Dota Truyền Kỳ,...

3.4.e .5. Góc nhìn kết hợp

Đây thực chất không phải là một khái niệm, nhiều tựa game hiện đại cho phép bạn có thể trải nghiệm game dưới một lúc 2 đến 3 góc nhìn khác nhau tùy theo sở thích

của người chơi, điều này thực sự không mới mẻ nhưng rất thú vị và được nhiều game thủ yêu thích.

Một số tựa game có góc nhìn kết hợp: Hitman, Grand Theft Auto V, Skyrim, Deus Ex, Đột kích (Chế độ đua xe bắn súng)...

3.4.f, Others

- Network: Đặc biệt quan trọng đối với các MMO
Ví dụ: NetDog, RakNet, GNE.
- Scripting: Giúp việc lập trình game trở nên mềm dẻo
Có thể sử dụng các ngôn ngữ scripting có sẵn (Python, Lua, Ruby...) hoặc viết ra các ngôn ngữ mới.
- Artifical Intelligent: Một số giải thuật Trí tuệ nhân tạo được sử dụng rất nhiều, do đó có thể xây dựng thành các framework để sử dụng trong Game engine.
– Hỗ trợ cross-platform: Giúp lập trình viên có thể viết game và build ở các HĐH khác nhau một cách dễ dàng.

3.4.g, Tools

- Model Editor: Maya3D, 3DMax, Blender, Sketchup...

3.5, Công cụ được lựa chọn để giải quyết các vấn đề đặt ra

3.5.a, Graphic Game engine: Object-Oriented Graphics Rendering Engine (OGRE)

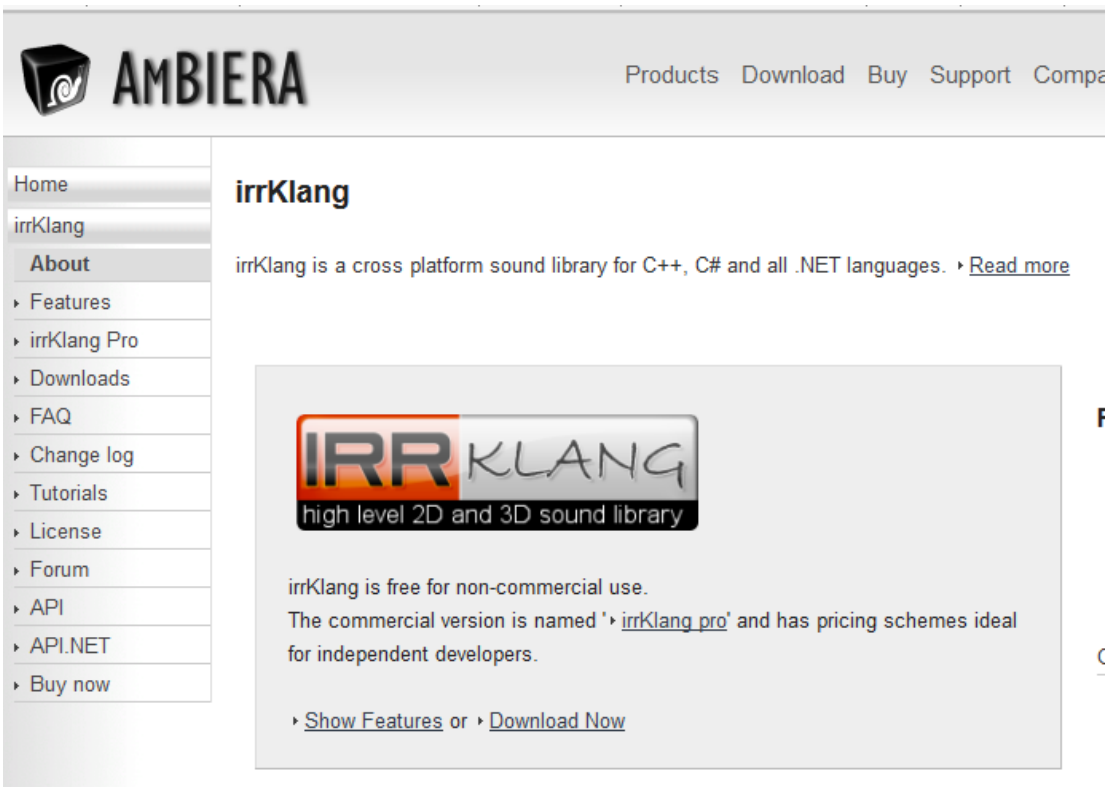
- Thông tin chung

Hình 1: Graphic Game engine có ứng dụng nhiều trong game

Nhà phát triển: OGRE Team
Phiên bản ổn định : 1.9.0 (Ghadamon) / 22/11/2013; 2 năm trước
Phiên bản xem trước: 2.1 / Công bố vào ngày 09/02/2015; 20 tháng trước
Thực trạng phát triển: Đang hoạt động
Ngôn ngữ: C ++
Nền tảng: Đa nền tảng

Kiểu: Graphics rendering engine
Giấy phép sử dụng: MIT (Massachusetts Institute of Technology)
Web chính thức: http://www.ogre3d.org/
- Các game ứng dụng
Zombie Driver (Hành động); Rigs of Rods (Đua xe), Skyscraper Simulator (Xây dựng), WorldForge (Phiêu lưu), Stunt Rally (Đua xe)...

3.5.b, Sound Game engine: irrKlang



Hình 2: irrKlang là thư viện hỗ trợ phát âm thanh 2D và 3D

irrKlang là thư viện hỗ trợ phát âm thanh 2D và 3D miễn phí với các định dạng WAV, MP3, OGG, MOD, XM, IT, S3M… với ngôn ngữ sử dụng là C ++ và tất cả các các ngôn ngữ .NET (C #, VisualBasic.NET , vv). Nó được phát triển bởi đội ngũ Irrlicht (đồng thời phát triển Game engine Irrlicht).

3.5.c, Object Oriented Input System: OIS

OIS là một thư viện nhận tín hiệu đầu vào nhỏ gọn, miễn phí được tích hợp vào Ogre3D (từ bản 1.4 (Eihort)). Thư viện OIS hỗ trợ cả 3 thiết bị chơi game như bàn phím, chuột và cần điều khiển.

3.5.d, GUI System: SdkTrays

SdkTrays hỗ trợ xây dựng giao diện điều khiển ở mức đơn gian dựa trên hệ thống OGRE Overlay thay vì phải sử dụng CEGUI hay MyGUI- những thư viện hỗ trợ chuyên nghiệp.

3.5.e, Other: Boost, TinyXML…

- Boost là một tập hợp các thư viện cho các ngôn ngữ lập trình C ++ cung cấp hỗ trợ tính toán, đa luồng, xử lý hình ảnh và kiểm tra đơn vị. Nó chứa hơn tám mươi thư viện cá nhân.
- TinyXML là một thư viện đơn giản, nhỏ cho C ++ dùng để phân tích cú pháp các file XML.

3.5.f, Tool: Maya3D, 3DMax, Sketchup, Ogre Meshy…

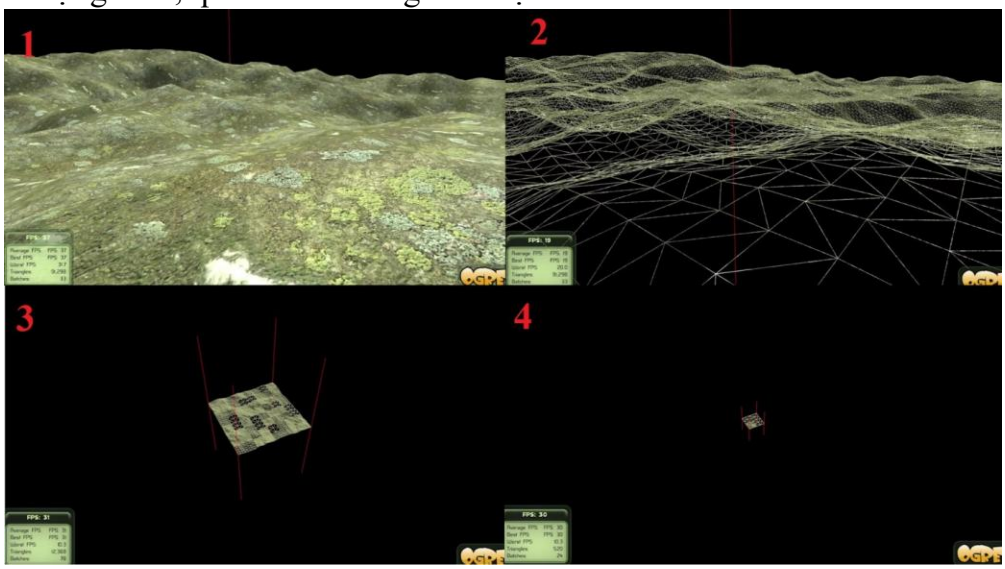
Phần 2. Các kết quả đạt được

1, Xây dựng Game engine với Ogre3D

1.1, Kỹ thuật quản lý khung cảnh LOD trong OGRE3D

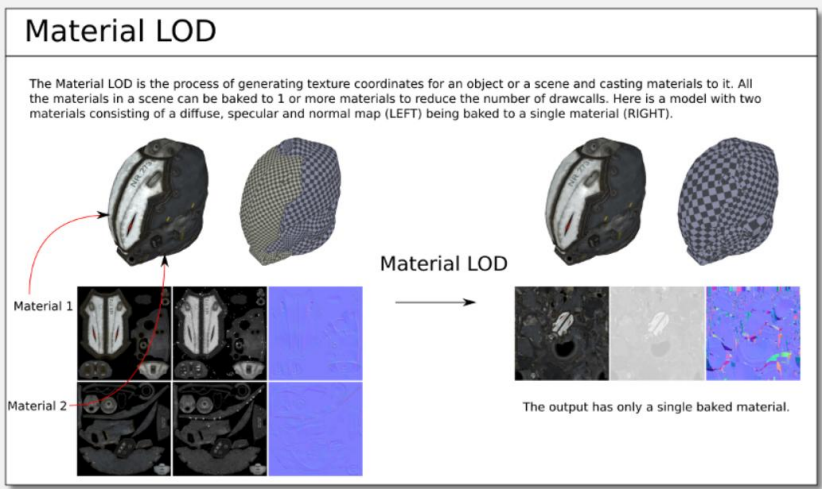
1.1.a, Khái niệm

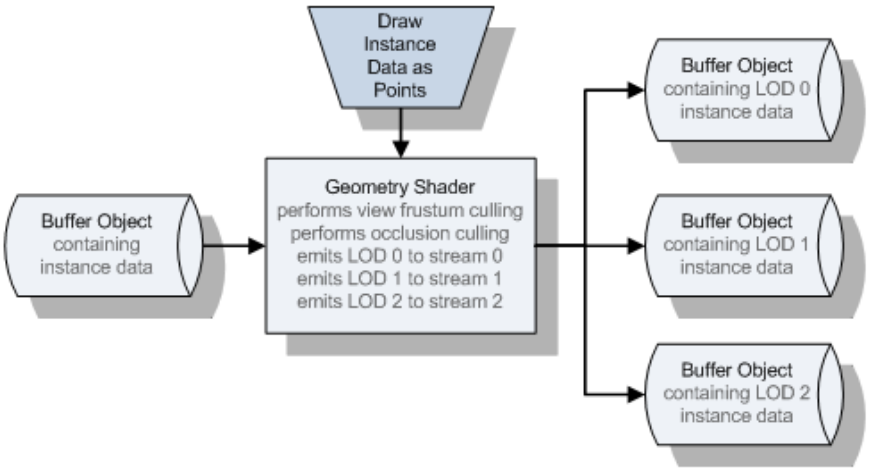
LOD (Level of detail): LOD cung cấp một hệ thống quản lý toạ độ để ta có thể gắn khung cảnh và các vật liệu đồ hoạ lên một đối tượng. Ngoài ra, LOD còn được phát triển để render các địa hình từ các file terrain meshes bao gồm quản lý đồ hoạ và cách bố trí các đối tượng meshes trong không gian. LOD sẽ giảm mức chi tiết cho các đối tượng ở xa, quá nhỏ để nâng cao hiệu suất render.

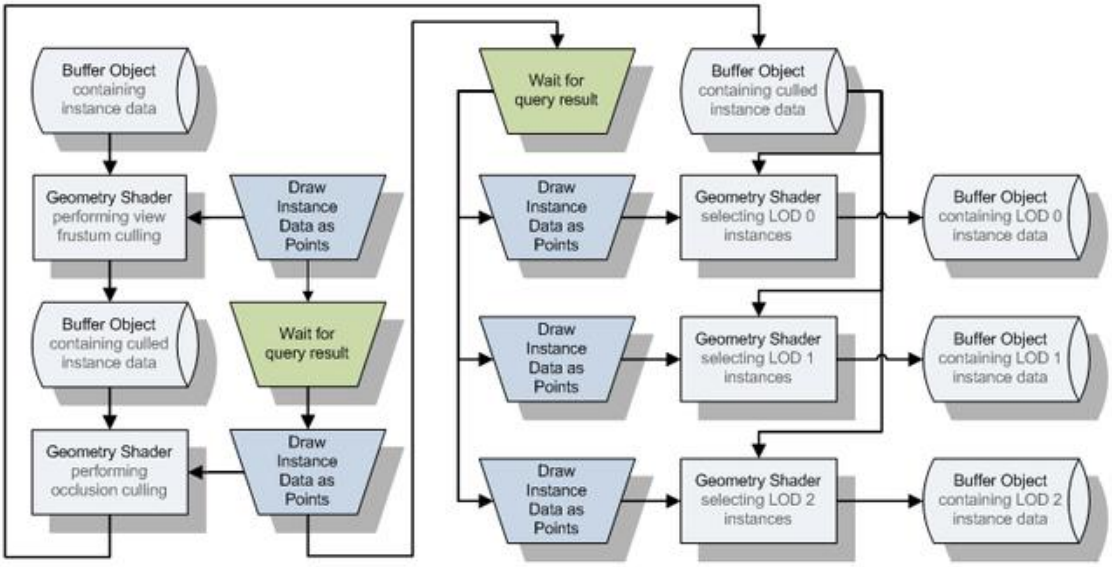


Hình 3: LOD sẽ giảm mức chi tiết cho các đối tượng ở xa, quá nhỏ

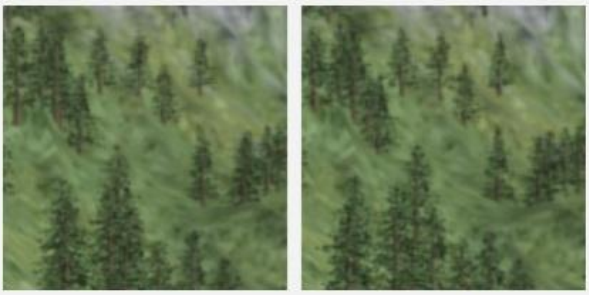
Dưới đây là một ví dụ bên trái là một đối tượng được gắn đầy đủ vật liệu đồ hoạ trong khi hành ảnh bên phải chỉ bao gồm thô khối đối tượng.



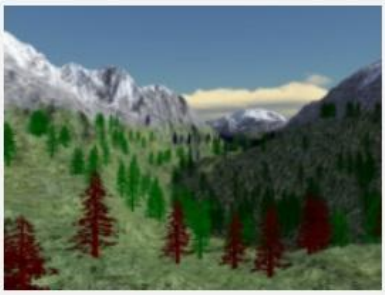




Các thuật toán của LOD dành cho đồ họa máy tính 3D được đề cập lần đầu trong một bài viết của James H. Clark trong sự kiện truyền thông ACM diễn ra vào tháng 10 năm 1976. Các thuật toán điển hình được giới thiệu là DLOD, Brute, HLOD và LOD-ing...



Close-up view of distant objects to compare the image quality without (left) and with (right) dynamic LOD.



Geometry LOD visualization: LOD 0 (red), LOD 1 (green), LOD 2 (blue).

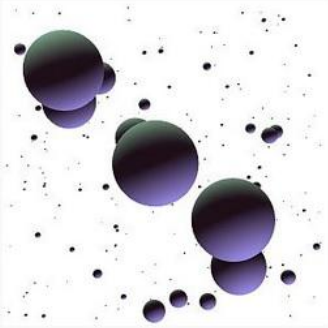
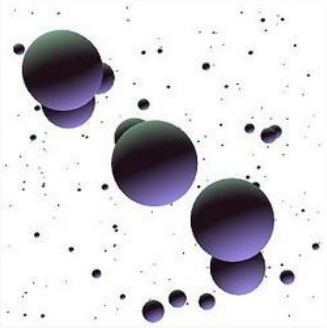
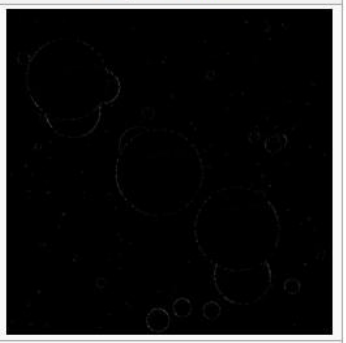
Hình 4: LOD giúp nâng cao hiệu suất render hình ảnh trong đồ hoạ 3D

1.1.b, *Ứng dụng thực tế*

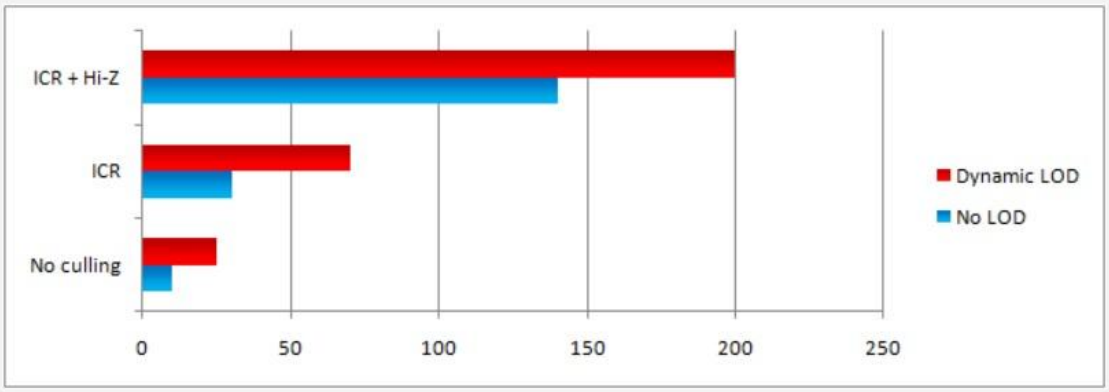
so sánh tác động thị giác và các phép đo

hình ảnh					
đỉnh	~ 5500	~ 2880	~ 1580	~ 670	140
Ghi chú	Chi tiết tối đa, cho closeups.				Chi tiết tối thiểu, đối tượng rất xa.

so sánh tác động thị giác và các phép đo

	brute	DLOD	sự so sánh
render hình ảnh			
Render thời gian	27,27 ms	1,29 ms	21 × giảm
đỉnh scene	2.328.480	109.440	21 × giảm

Hình 5: Hiệu suất reder hình ảnh vượt trội so với BRUTE



So sánh hiệu suất của các bản demo trong khung hình mỗi giây trên một HD5770 Radeon (cao hơn là tốt hơn): không có ICR + Hi-Z bản đồ tắc dựa trên tiêu huỷ (trên cùng) tiêu huỷ (phía dưới), giảm dụ đám mây (giữa),, không có hình học LOD (màu xanh), hình học động LOD (màu đỏ).

Ứng dụng LOD đặc biệt hữu ích trong các ứng dụng game video 3D khi làm giảm sức nặng về phần cứng cho người lập trình. Các game đầu tiên ứng dụng LOD có thể kể đến như Spyro the Dragon , Crash Bandicoot: Warped , Unreal Tournament and the Serious Sam engine. Ngoài ra, các ứng dụng sử dụng quản lý đồ hoạ cũng áp dụng LOD như MeshLab (Công cụ xử lý nguồn file mesh), Polygon Cruncher, Simplygon, Vizup, Rational Reducer, Pro Optimizer, GPure...

1.2, Quản lý khung cảnh đồ hoạ (BSP-tree và Octree)

1.2.a, Quản lý khung cảnh đồ hoạ

Quản lý khung cảnh đồ hoạ(Scene Graph Management) được xây dựng trên nền OpenGL 3D Graphics Engine. Scene Graph quản lý toàn bộ thế giới 3D, có thể được sử dụng để quản lý các đối tượng trong một công cụ đồ họa 3D.

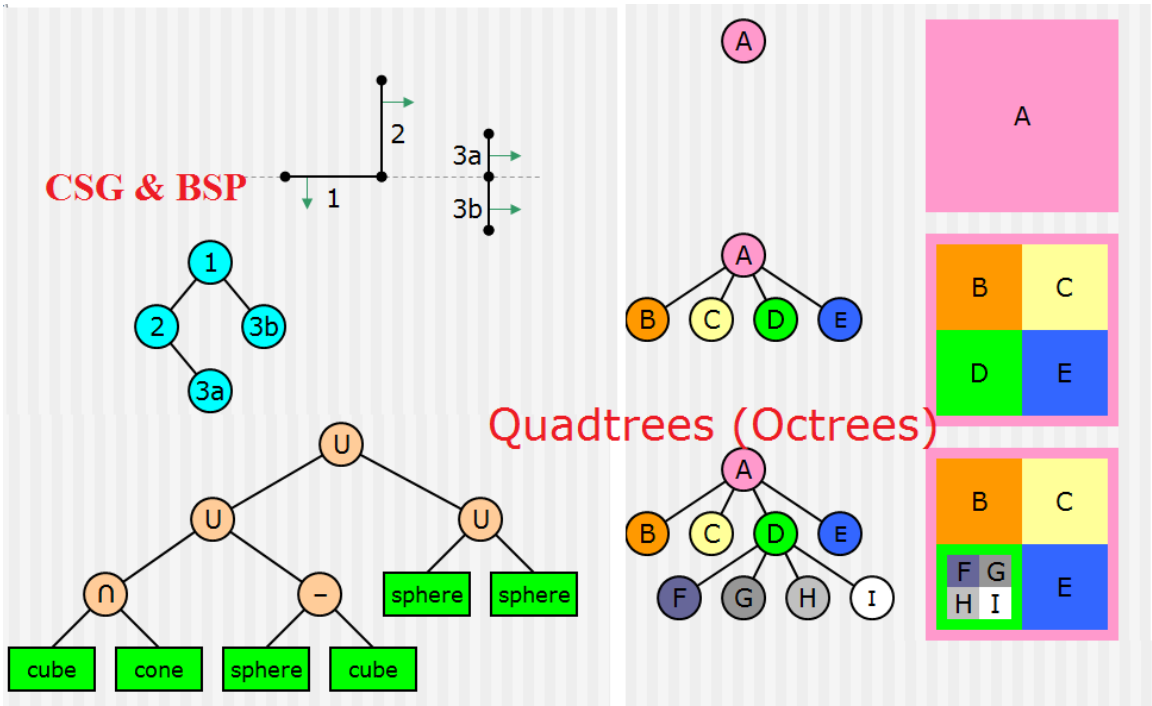
Quản lý khung cảnh đồ hoạ bao gồm các nút (Node) và mảng (Arc). Các nút này được phân cấp, trong khi các Arcs được định hướng với các nút con và nút cha.

Có 4 kiểu quản lý khung cảnh đồ hoạ là Scene Graphs(Các nút giữ các đối tượng); CSG Trees; BSP Trees (Các nút giữ các đối tượng); Quadtrees & Octrees (Các nút giữ các vùng không gian và các nhánh quản lý các đối tượng).

1.2.b, BSP-tree

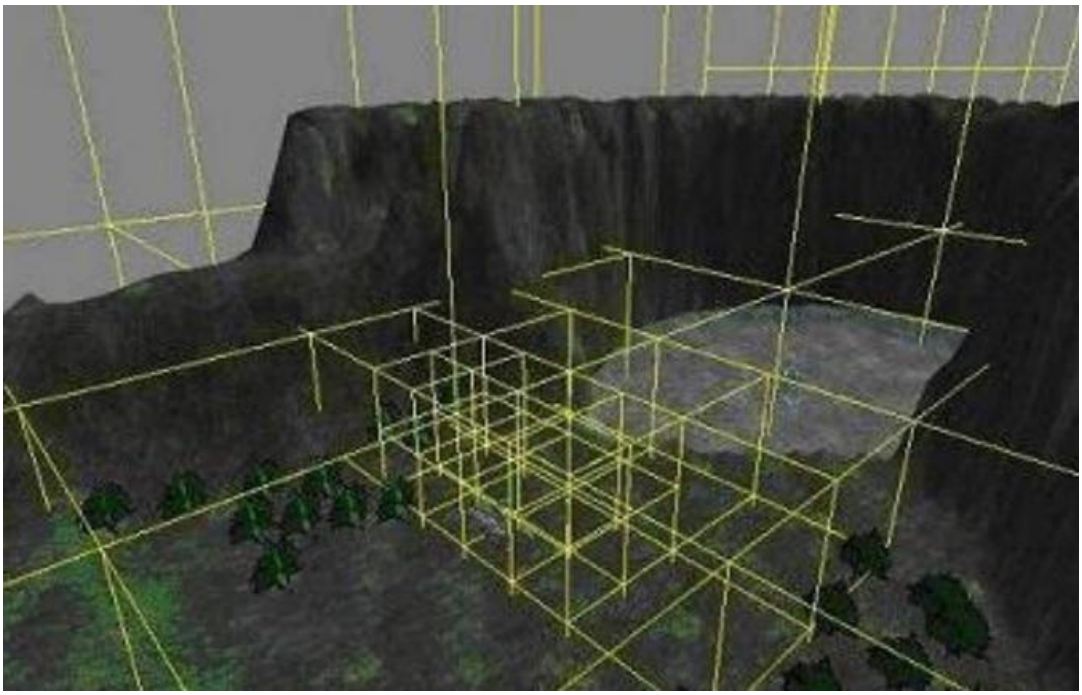
BSP-tree là cấu trúc dữ liệu theo dạng cây nhị phân được sử dụng xây dựng các đối tượng tĩnh có cấu trúc phức tạp, đặc biệt là map trong nhà. Cấu trúc BSP phải được xây dựng hoàn thiện trước khi load vào game. Như thường lệ, trong BSP-tree mỗi vùng được quản lý bởi các nút.

Ưu điểm:
BSP-tree quy hoạch quy hoạch tốt các map phức tạp và xử lý các va chạm hiệu quả.



1.2.c, Octree/Quad tree

Octree là một loại đặc biệt của cấu trúc dữ liệu mà mỗi nút có 8 đối tượng. Quad tree cũng tương tự nhưng với 4 đối tượng cho một nút. Phương pháp Octree cũng có thể được sử dụng song song với Quad tree.



Hình 6: Octree quản lý các nút theo phân vùng

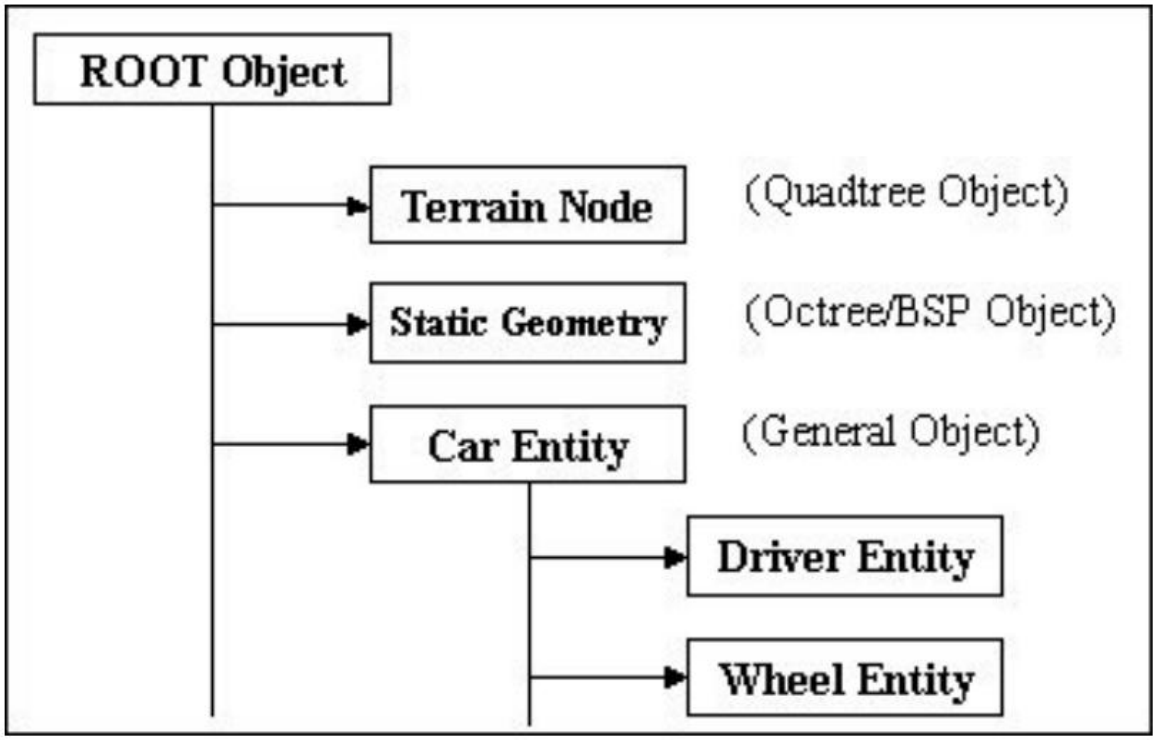
Toàn bộ dữ liệu thế giới được chia thành tám nút khởi tạo. Nút gốc quản lý khối lập phương tạo lên cả thế giới trong 3D, trong khi các nút con còn lại tiếp tục đệ quy quản lý các nút con khác.

Ưu điểm: Octree cung cấp giải pháp cân bằng cho quản lý khung cảnh ngoài trời và những cảnh trong nhà. Đối với địa hình lớn nó mang lại cho hiệu suất tốt hơn so với các kiểu cấu trúc dữ liệu khác.

1.2.d, CSG tree (Constructive Solid Geometry tree)

Trong CSG tree, các cảnh vật được xây dựng dưới dạng khối ngay từ đầu với các nút riêng quản lý theo dạng cây nhị phân.

Ưu điểm: CSG tree thích hợp với các cảnh vật đơn giản không quá lớn. Nó giúp dễ dàng bắt được các va chạm trong map.



Hình 7: Sự kết hợp giữa BSP-tree và Octree trong Scene Graph Management giúp nâng cao hiệu suất hiển thị hình ảnh

1.2.e, Lồng ghép BSP-tree và Octree trong Scene Graph Management

Có thể lồng ghép sử dụng quản lý đối tượng bằng Octree/Quad tree và BSP/Portals. BSP-tree sử dụng các nút quản lý cảnh vật trong khi Octree/Quad tree quản lý các đối tượng. Sự kết hợp này sẽ giúp việc render hình ảnh đạt hiệu suất tối ưu hơn.

1.3, Các API chính trong Modules của Core

1.3.a, Animation

Class Ogre :: AnimableObject
Định nghĩa một interface tới các Class AnimableValue được khởi tạo.

Class Ogre :: AnimableValue
Định nghĩa các thuộc tính animable cho một đối tượng.

Class Ogre :: Animation
Lưu trữ các chuỗi Animation của đối tượng.

Class Ogre :: AnimationContainer
Tạo một interface chứa các Animation.

Class Ogre :: AnimationState
Quy định thời hạn của một Animation trong một state nhất định.

Class Ogre :: AnimationStateControllerValue
Class ControllerValue cho AnimationState.

Class Ogre :: AnimationStateSet
Class đóng gói một tập các đối tượng AnimationState.

Class Ogre :: AnimationTrack
Kiểm soát các điểm track của một Animation.

Class Ogre :: Bone
Kiểm soát một xương trong bộ xương khung của một Animation.

Class Ogre :: Keyframe
Tạo khung chính chính từ các AnimationTrack.

struct Ogre :: LinkedSkeletonAnimationSource
Liên kết với bộ xương khung khác để tạo các Animation.

Class Ogre :: NodeAnimationTrack
AnimationTrack chuyên để giữ các nút khi nó bị biến đổi.

Class Ogre :: NumericAnimationTrack
AnimationTrack chuyên để đối phó với các giá trị animable chung chung.

Class Ogre :: NumericKeyFrame
Keyframe đặc hiệu mà các cửa hàng bất kỳ giá trị số.

Class Ogre :: Pose
Một tư thế là một tập hợp liên kết của offsets đỉnh áp dụng cho một tập hợp các dữ liệu đỉnh.

Class Ogre :: Skeleton
Một bộ sưu tập của các đối tượng với khung xương di chuyển.

Class Ogre :: SkeletonManager
Xử lý việc quản lý các nguồn tài nguyên Skeleton load vào Animation.

Class Ogre :: SkeletonSerializer
Class liên kết dữ liệu bộ xương load từ fiel .skeleton trong folder media.

Class Ogre :: TagPoint
Một điểm được gắn thẻ vào một bộ xương, có thể được sử dụng để gắn các đối tượng để các thực thể khác cụ thể.

Class Ogre :: TimeIndex
Hiện chỉ số đối tượng sử dụng để tìm kiếm keyframe vào vị trí nhất định.

Class Ogre :: TransformKeyFrame
Keyframe đặc hiệu mà các cửa hàng toàn biến đổi.

Class Ogre :: VertexAnimationTrack
Quản lý các AnimationTrack khi đối tượng thay đổi thông tin vị trí node.

struct Ogre :: VertexBoneAssignment_s
Ghi lại animation của khung xương.

Class Ogre :: VertexMorphKeyFrame
Keyframe dành cho bộ nhớ đệm.

Class Ogre :: VertexPoseKeyFrame
Keyframe quản lý các tập hợp các bộ đệm để cung cấp một tư thế blendable.

1.3.b, Resources

Class Ogre :: Archive
Class lưu trữ xử lý.

Class Ogre :: ArchiveFactory
Class quản lý lưu trữ các plugin.

Class Ogre :: ArchiveManager
Class này quản lý các plugin ArchiveFactory có sẵn.

struct Ogre :: BackgroundProcessResult
Lữu trữ các kết quả của một yêu cầu hàng đợi nền.

Class Ogre :: DataStream
Class mục đích chung được sử dụng cho việc đóng gói đọc và viết dữ liệu.

Class Ogre :: EmbeddedZipArchiveFactory
ZipArchiveFactory Lữu trữ tập tin dưới dạng Zip.

Class Ogre :: FileHandleDataStream
Class con của DataStream để xử lý dữ liệu từ tập tin xử lý.

struct Ogre :: FileInfo
Thông tin về một tập tin / thư mục trong thư lưu trữ sẽ được trả về bằng một struct FileInfo.

Class Ogre :: FileStreamDataStream
Class con của DataStream để xử lý dữ liệu từ std :: basic_istream.

Class Ogre :: FileSystemArchive
FileSystemArchive Lưu trữ để cho phép đọc các tập tin từ thư mục hệ thống tập tin / thư mục.

Class Ogre :: FileSystemArchiveFactory
ArchiveFactory quản lý các tập tin hệ thống.

Class Ogre :: Font
Class quản lý phông chữ trong hệ thống.

Class Ogre :: FontManager
Quản lý các mẫu Font, phân tích các file .fontdef và đọc chúng.

Class Ogre :: GpuProgram
Định nghĩa một chương trình chạy trên GPU.

Class Ogre :: GpuProgramManager

Class Ogre :: HighLevelGpuProgram
Class cơ sở trừu tượng đại diện cho một chương trình cấp cao (một đỉnh hoặc đoạn chương trình).

Class Ogre :: HighLevelGpuProgramFactory

Class Ogre :: HighLevelGpuProgramManager
ResourceManager quản lý các chương trình và fragment- ở mức cao.

Class Ogre :: ManualResourceLoader
Định nghĩa một bộ nạp tài nguyên.

Class Ogre :: MemoryDataStream
Class con của DataStream để xử lý dữ liệu từ bộ nhớ.

Class Ogre :: Mesh
Resource tái tạo lại ở dạng lưới 3D.

struct Ogre :: MeshLodUsage

Class Ogre :: MeshManager
Xử lý việc quản lý các nguồn tài nguyên lưới.

Class Ogre :: MeshSerializer
Class quản lý dữ liệu lưới từ một tập tin .mesh trong media.

Class Ogre :: MeshSerializerImpl
Bộ đọc các file định dạng .mesh.

Class Ogre :: MeshSerializerImpl_v1_1
Class để cung cấp khả năng tương thích ngược để tải phiên bản 1.1 của định dạng .mesh.

Class Ogre :: MeshSerializerImpl_v1_2
Class để cung cấp khả năng tương thích ngược để tải phiên bản 1.2 của định dạng .mesh.

Class Ogre :: MeshSerializerImpl_v1_3
Class để cung cấp khả năng tương thích ngược để tải phiên bản 1.3 của định dạng .mesh.

Class Ogre :: MeshSerializerImpl_v1_4
Class để cung cấp khả năng tương thích ngược để tải phiên bản 1.4 của định dạng .mesh.

Class Ogre :: MeshSerializerImpl_v1_41
Class để cung cấp khả năng tương thích ngược để tải phiên bản 1.41 của định dạng .mesh.

Class Ogre :: MeshSerializerListener

Class Ogre :: PrefabFactory
Class tạo ra prefabs cho mesh.

Class Ogre :: Resource
Tóm tắt Class đại diện cho một nguồn tài nguyên có thể nạp được.

Class Ogre :: ResourceBackgroundQueue
Class này được sử dụng để thực hiện các hoạt động tài nguyên trong một thread nền.

Class Ogre :: ResourceGroupListener
Class trừu tượng này định nghĩa một giao diện được gọi trở lại trong nhóm tài nguyên tải để chỉ ra sự tiến bộ của phụ tải.

Class Ogre :: ResourceGroupManager
Class Singleton này quản lý các danh sách của các nhóm tài nguyên, và thông báo cho các nhà quản lý tài nguyên khác nhau của các nghĩa vụ của họ để tải / dỡ bỏ các nguồn lực trong một nhóm.

Class Ogre :: ResourceLoadingListener

Class Ogre :: ResourceManager
Định nghĩa một hàm xử lý nguồn tài nguyên chung.

Class Ogre :: StreamSerialiser
Class tiện ích cung cấp các phương thức trợ giúp cho việc đọc / ghi dữ liệu có cấu trúc tổ chức tại một DataStream.

Class Ogre :: SubMesh
Định nghĩa một phần của một mạng lưới hoàn chỉnh.

Class Ogre :: Texture
Tóm tắt Class đại diện cho một nguồn tài nguyên Texture.

Class Ogre :: TextureManager
Class để nạp và kết cấu quản lý.

Class Ogre :: UnifiedHighLevelGpuProgram
Chức năng của HighLevelGpuProgram mà chỉ các đại biểu thực hiện của mình cho một chương trình cấp cao khác.

Class Ogre :: UnifiedHighLevelGpuProgramFactory
Class quản lý các tiến trình chạy.

Class Ogre :: ZipArchive
Chức năng của Class ZipArchive để cho phép đọc các tập tin từ một kho lưu trữ dưới định dạng zip.

Class Ogre :: ZipArchiveFactory
ArchiveFactory quản lý các tập tin Zip.

Class Ogre :: ZipDataStream
Quản lý DataStream để xử lý luồng dữ liệu từ kho lưu trữ zip.

1.3.c, RenderSystem

Class Ogre :: DefaultHardwareBufferManager
DefaultHardwareBufferManager như một Singleton.

Class Ogre :: DefaultHardwareBufferManagerBase

Class Ogre :: DefaultHardwareCounterBuffer

Class Ogre :: DefaultHardwareIndexBuffer

Class Ogre :: DefaultHardwareUniformBuffer

Class Ogre :: DefaultHardwareVertexBuffer

Class Ogre :: DepthBuffer

struct Ogre :: DriverVersion
DriverVersion được sử dụng bởi RenderSystemCapabilities cho cả GL và D3D9

Class Ogre :: HardwareBuffer
Định nghĩa bộ đệm cứng.

Class Ogre :: HardwareBufferLicensee

Class Ogre :: HardwareBufferManager
Quản lý bộ đệm cứng cho Singleton.

Class Ogre :: HardwareBufferManagerBase

Class Ogre :: HardwareCounterBuffer

Class Ogre :: HardwareCounterBufferSharedPtr

Class Ogre :: HardwareIndexBuffer

Class Ogre :: HardwareIndexBufferSharedPtr

Class Ogre :: HardwareOcclusionQuery

Class Ogre :: HardwarePixelBuffer

Class Ogre :: HardwarePixelBufferSharedPtr

Class Ogre :: HardwareUniformBuffer

Class Ogre :: HardwareUniformBufferSharedPtr

Class Ogre :: HardwareVertexBuffer

Class Ogre :: HardwareVertexBufferSharedPtr

Class Ogre :: IndexData

Class Ogre :: MultiRenderTarget
Quản lý chạy cùng lúc nhiều RenderTextures.

Class Ogre :: QueuedRenderableCollection

Class Ogre :: QueuedRenderableVisitor

struct Ogre :: RenderablePass

Class Ogre :: RenderOperation

Class Ogre :: RenderPriorityGroup

Class Ogre :: RenderQueue
Lớp để quản lý hàng đợi khung hình sẽ render.

Class Ogre :: RenderQueueGroup

Class Ogre :: RenderQueueInvocation

Class Ogre :: RenderQueueInvocationSequence

Class Ogre :: RenderQueueListener

Class Ogre :: RenderSystem
Xác định các chức năng của một API 3D.

Class Ogre :: RenderSystemCapabilities
Lớp singleton để lưu trữ các khả năng của card đồ họa.

Class Ogre :: RenderSystemCapabilitiesManager
Lớp để quản lý cơ sở dữ liệu cho RenderSystemCapabilities.

Class Ogre :: RenderSystemCapabilitiesSerializer
Class cho quản lý chuỗi sự kiện RenderSystemCapabilities đến trong file kịch bản *.rendercaps.

Class Ogre :: RenderTarget

struct Ogre :: RenderTargetEvent
Struct có chứa thông tin về một sự kiện RenderTarget.

Class Ogre :: RenderTargetListener
Một lớp giao diện xác định một người nghe có thể được sử dụng để nhận thông báo về các sự kiện RenderTarget.

struct Ogre :: RenderTargetViewportEvent
Struct có chứa thông tin về một sự kiện RenderTarget Viewport cụ thể.

Class Ogre :: RenderTexture
Lớp này đại diện cho một RenderTarget mà gắn vào một Texture.

Class Ogre :: RenderToVertexBuffer
Một đối tượng dùng để làm hình học với một đỉnh.

Class Ogre :: RenderWindow
Quản lý các mục tiêu render cửa sổ.

Class Ogre :: TempBlendedBufferInfo
Cấu trúc để ghi lại việc sử dụng các bộ đệm pha trộn tạm thời.

Class Ogre :: VertexBufferBinding
Ghi lại trạng thái của tất cả các ràng buộc đệm đỉnh yêu cầu cung cấp một tuyên bố đỉnh với các dữ liệu đầu vào cần thiết cho các yếu tố đỉnh.

Class Ogre :: VertexCacheProfiler
Vertex bộ nhớ cache hồ sơ.

Class Ogre :: VertexData

Class Ogre :: VertexDeclaration

Class Ogre :: VertexElement

Class Ogre :: Viewport

Class Ogre :: WindowEventListener

Class Ogre :: WindowEventUtilities

1.3.d, Scene

Class Ogre :: AxisAlignedBoxSceneQuery

Class Ogre :: BaseInstanceBatchVTF

Class Ogre :: Camera
Quản lý một viewpoint của cảnh sẽ được hiển thị.

Class Ogre :: DefaultAxisAlignedBoxSceneQuery

Class Ogre :: DefaultIntersectionSceneQuery

Class Ogre :: DefaultPlaneBoundedVolumeListSceneQuery

Class Ogre :: DefaultRaySceneQuery

Class Ogre :: DefaultSceneManager

Class Ogre :: DefaultSceneManagerFactory

Class Ogre :: DefaultShadowCameraSetup

Class Ogre :: DefaultSphereSceneQuery

Class Ogre :: Entity

Class Ogre :: EntityFactory

Class Ogre :: FocusedShadowCameraSetup

Class Ogre :: InstanceBatch

Class Ogre :: InstanceBatchHW

Class Ogre :: InstanceBatchHW_VTF

Class Ogre :: InstanceBatchShader

Class Ogre :: InstanceBatchVTF

Class Ogre :: InstancedEntity

Class Ogre :: InstancedGeometry

Class Ogre :: InstanceManager

struct Ogre :: InstanceManager :: BatchSettings

Class Ogre :: IntersectionSceneQuery

Class Ogre :: IntersectionSceneQueryListener

struct Ogre :: IntersectionSceneQueryResult

Class Ogre :: Light
Quản lý nguồn ánh sáng trong scene.

Class Ogre :: LightFactory

Class Ogre :: LiSPSMShadowCameraSetup
Quản lý các thuật toán Mapping đổ bóng ánh sáng với camera.

Class Ogre :: ManualObject

Class Ogre :: ManualObjectFactory

Class Ogre :: MovableObject
Lớp trừu tượng định nghĩa một đối tượng di chuyển trong scene.

Class Ogre :: MovableObjectFactory

Class Ogre :: MovablePlane

Class Ogre :: Node

Class Ogre :: PlaneBoundedVolumeListSceneQuery

Class Ogre :: PlaneOptimalShadowCameraSetup
Tối ưu các thuật toán đổ bóng.

Class Ogre :: PSSMShadowCameraSetup

Class Ogre :: RaySceneQuery

Class Ogre :: RaySceneQueryListener

struct Ogre :: RaySceneQueryResultEntry

Class Ogre :: Rectangle2D
Cho phép vẽ một hình chữ nhật 2D đơn giản.

Class Ogre :: RegionSceneQuery
Lớp trừu tượng xác định một truy vấn mà trả về kết quả duy nhất từ một khu vực trong scene.

Class Ogre :: Renderable
Lớp trừu tượng định nghĩa giao diện tất cả các đối tượng renderable phải thực hiện.

Class Ogre :: SceneManager
Quản lý các tổ chức và cung cấp một scene.

Class Ogre :: SceneManagerEnumerator
Liệt kê các lớp SceneManager có sẵn.

Class Ogre :: SceneManagerFactory
Lớp sẽ tạo ra một SceneManager nhất định.

struct Ogre :: SceneManagerMetaData

Class Ogre :: SceneNode
Lớp khai báo một nút trong quản lý khung cảnh.

Class Ogre :: SceneQuery
Một lớp học để thực hiện các truy vấn trong một scene.

Class Ogre :: SceneQueryListener
Lớp tùy chọn này cho phép bạn nhận được các callbacks mỗi kết quả từ SceneQuery.

struct Ogre :: SceneQueryResult
Giữ các kết quả của một truy vấn trong scene ở trên.

Class Ogre :: ShadowCameraSetup
Lớp này cho phép câú hình các thiết lập máy ảnh khi dựng hình và tạo hiệu ứng bóng đổ.

Class Ogre :: ShadowCaster

Class Ogre :: ShadowRenderable

struct Ogre :: ShadowTextureConfig

Class Ogre :: ShadowTextureManager

Class Ogre :: ShadowVolumeExtrudeProgram

Class Ogre :: SimpleRenderable

Class Ogre :: SkeletonInstance

Class Ogre :: SphereSceneQuery
Lớp SceneQuery cho các truy vấn các đối tượng trong một hình cầu.

struct Ogre :: StaticFaceGroup

Class Ogre :: StaticGeometry

Class Ogre :: SubEntity
Lớp tiện ích cho xác định các đối tượng trong scene.

Class Ogre :: UserObjectBindings
Lớp học mà cung cấp giao diện thuận tiện để thiết lập một mối liên kết giữa các đối tượng ứng dụng người dùng tùy chỉnh và lớp lõi Ogre.

struct Ogre :: ViewPoint
Cơ cấu tổ chức một vị trí và định hướng cặp.

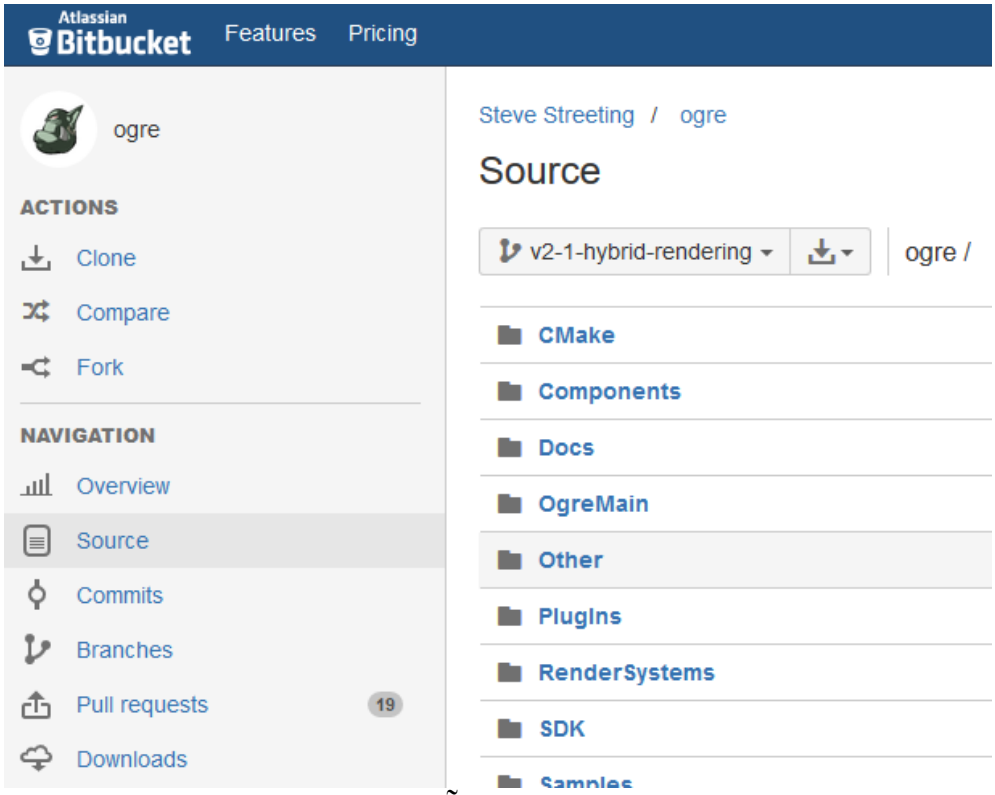
struct Ogre :: VisibleObjectsBoundsInfo
Cơ cấu thu hợp thông tin về các đối tượng có thể nhìn thấy mà đã được phát hiện trong một cảnh.

Class Ogre :: WireBoundingBox
Cho phép dựng hình trong một bao wireframe.

1.4, Build SDK

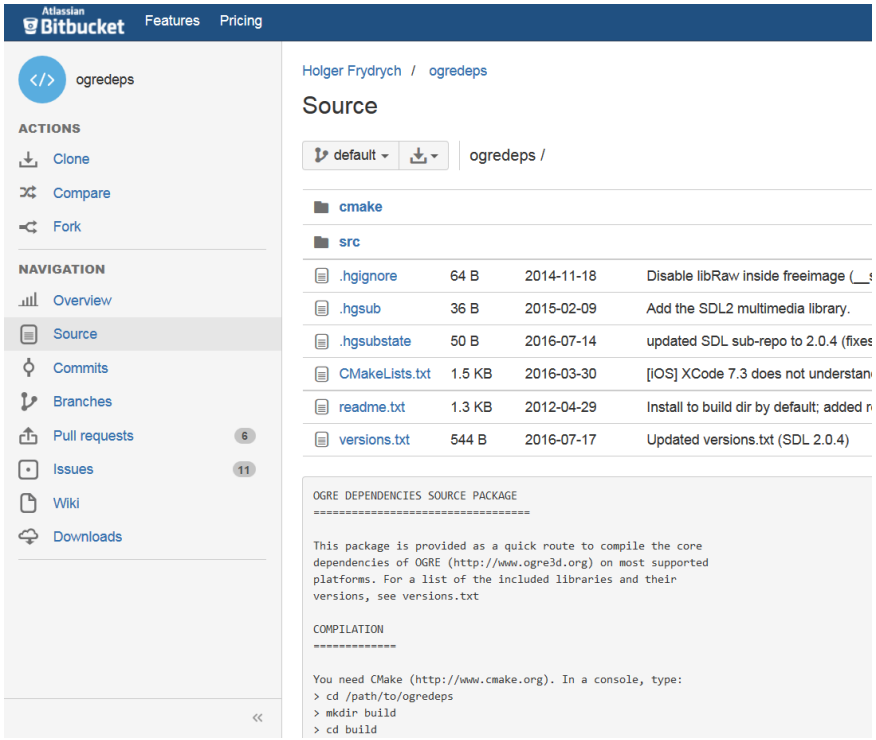
1.4.a, Ogre3D

+ Bước 1: Dùng tool git control version Clone SDK Ogre3D về máy tính theo đường dẫn: https://bitbucket.org/sinbad/ogre (Đặt vào E:\TNA\OgreSDK\OgreSDK_vc10_v2-0-0\Source). Có thể chọn version tùy ý (tính tới tháng 1/2017 mới nhất là v2-1-hybrid-rendering).



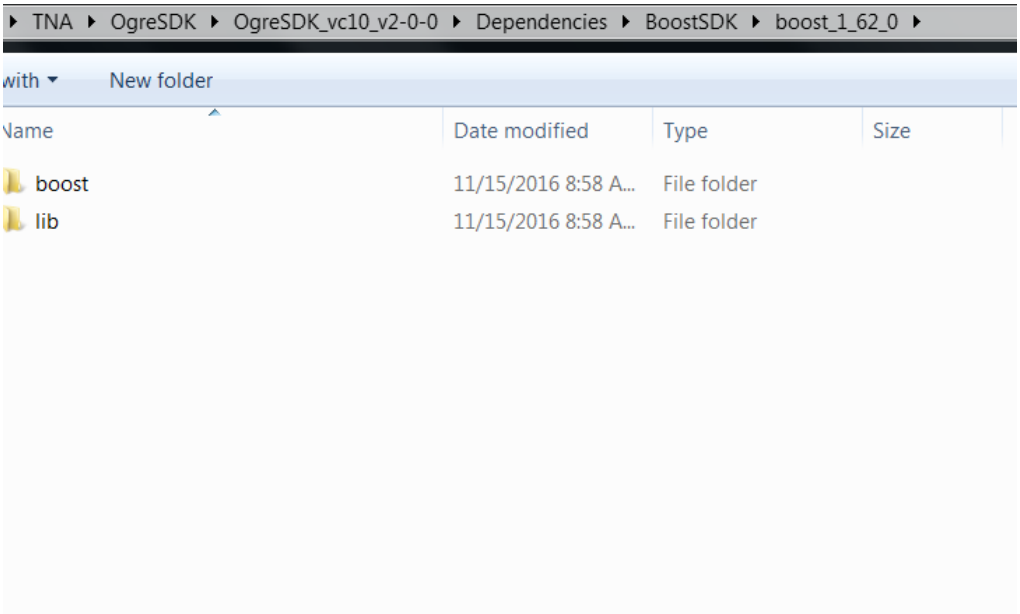
+ Bước 2: Cài đặt Cmake với đường dẫn download file cài đặt là https://cmake.org/download/.

+ Bước 3: Dùng tool git control version Clone Ogredeps về máy tính theo đường dẫn: https://bitbucket.org/cabalistic/ogredeps. Project này sẽ bao gồm các module phụ mà OgreSDK cần thiết để build.

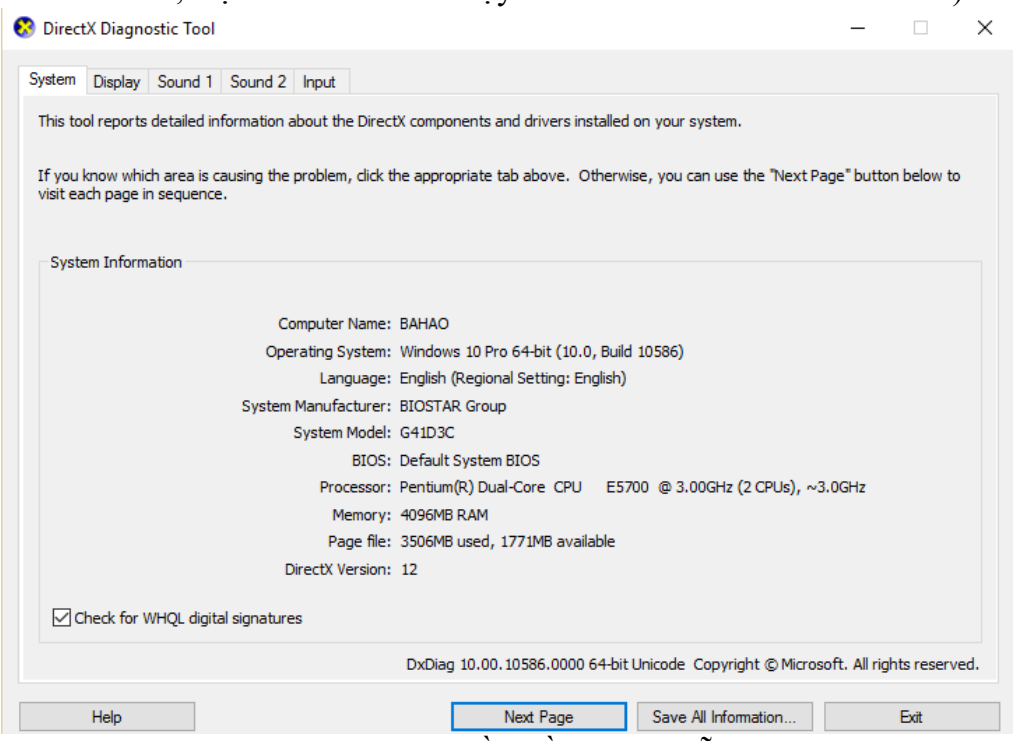


+ Bước 4: Download Boost SDK theo đường dẫn là
https://sourceforge.net/projects/boost/files/boost/1.62.0/boost_1_62_0.zip/download

Hướng dẫn build Boost với Visual Studio x64 Win64 Command Prompt (2010) (Tương tự với phiên bản khác):
1, Ấn Window + Search(từ khóa " Visual Studio x64 Win64 Command Prompt (2010) ") và chạy theo chế độ Administrator.
2, Nhảy tới thư mục "C:\Program Files\boost\boost_1_62_0"(đường dẫn tới BoostSDK vừa download về) bằng lệnh CD, sau đó gõ "run bootstrap.bat".
3, Tiếp tục gõ "Run .\b2"
4, Tiếp tục gõ "Run .\b2 variant=debug,release link=static runtime-link=static address-model=64"
5, Sau đó, Cut thư mục C:\Program Files\boost\boost_1_62_0\stage\lib vào C:\Program Files\boost\boost_1_62_0\lib.
6, Ta có đường dẫn C:\Program Files\boost\boost_1_62_0\lib là thư viện hoàn chỉnh. (Copy vào E:\TNA\OgreSDK\OgreSDK_vc10_v2-0-0\Dependencies\BoostSDK\boost_1_62_0)

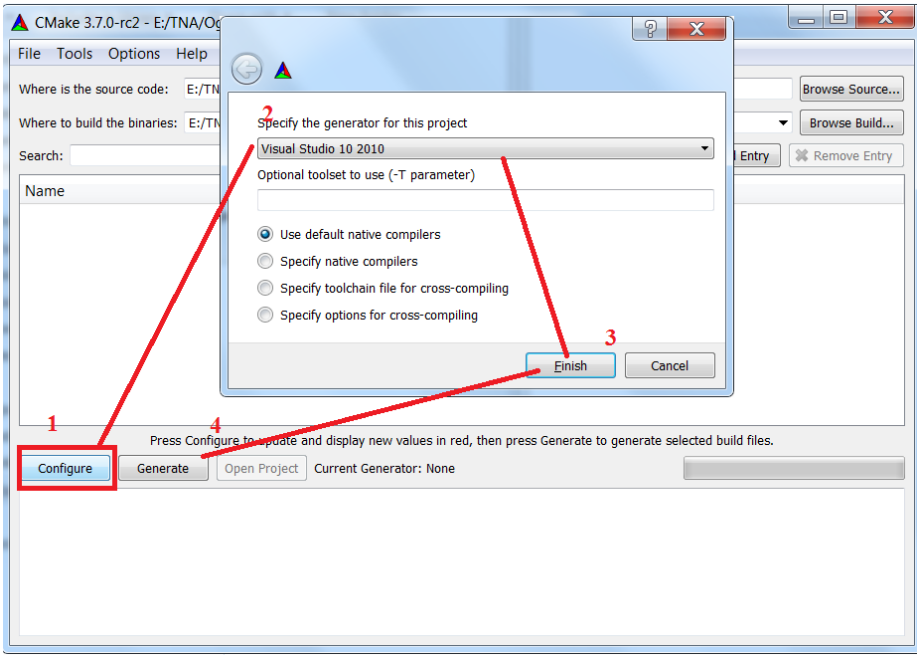


Bước 5: Download DirectX Software Development Kit theo đường dẫn là https://www.microsoft.com/en-us/download/confirmation.aspx?id=6812. Sau đó cài đặt bình thường, chọn đường dẫn mặc định, có thể bỏ bớt tùy chọn cài đặt tool và Sample. Ngoài ra, có thể không cài nếu như dùng công nghệ OpenGL thay vì DirectX9 và DirectX11. Nếu như vậy khi Cmake cần bỏ tích các thư viện liên quan DirectX9 và DirectX11 tới DirectX9 và DirectX11 và config lại file plugin trong bin để SDK không load RenderSystem_Direct3D9_d.dll và RenderSystem_Direct3D11_d.dll (DirectX – Microsoft là công nghệ render hình ảnh của Microsoft, hiện bản mới nhất chạy trên Windows 10 là DirectX12).

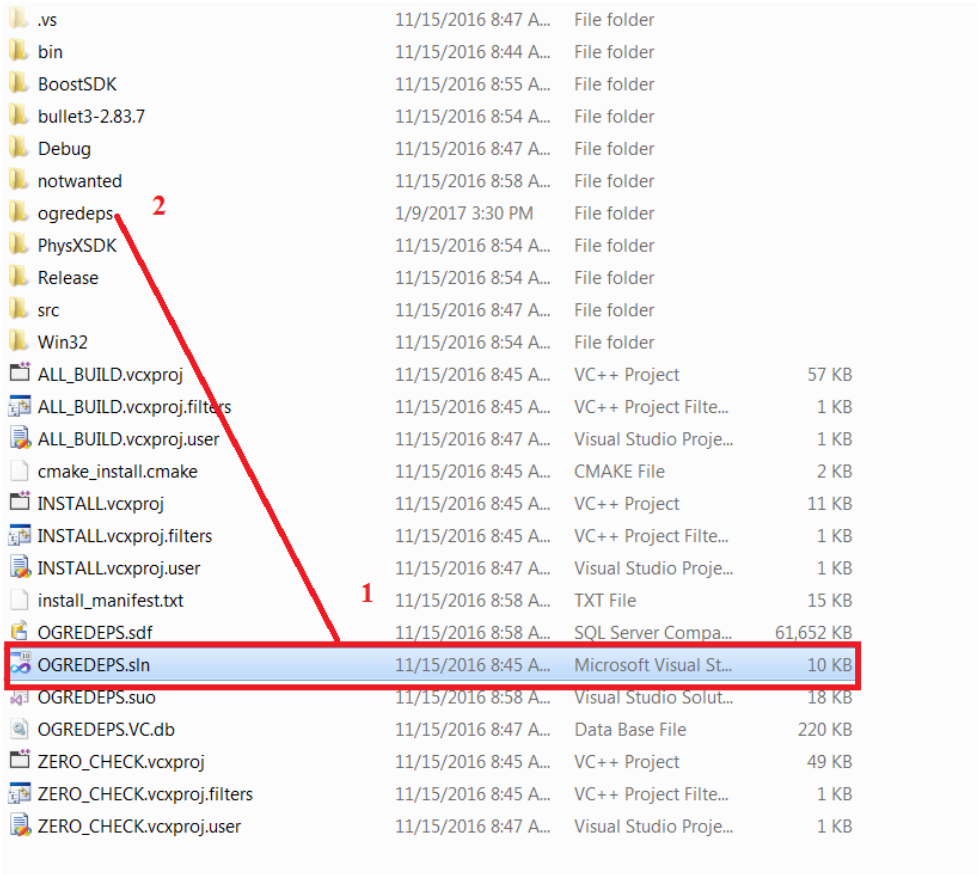


+ Bước 6: Build Ogredeps. Bật Cmake rồi điền đường dẫn của Ogredeps (E:\TNA\OgreSDK\OgreSDK_vc10_v2-0-0\Dependencies) vừa clone về vào ô

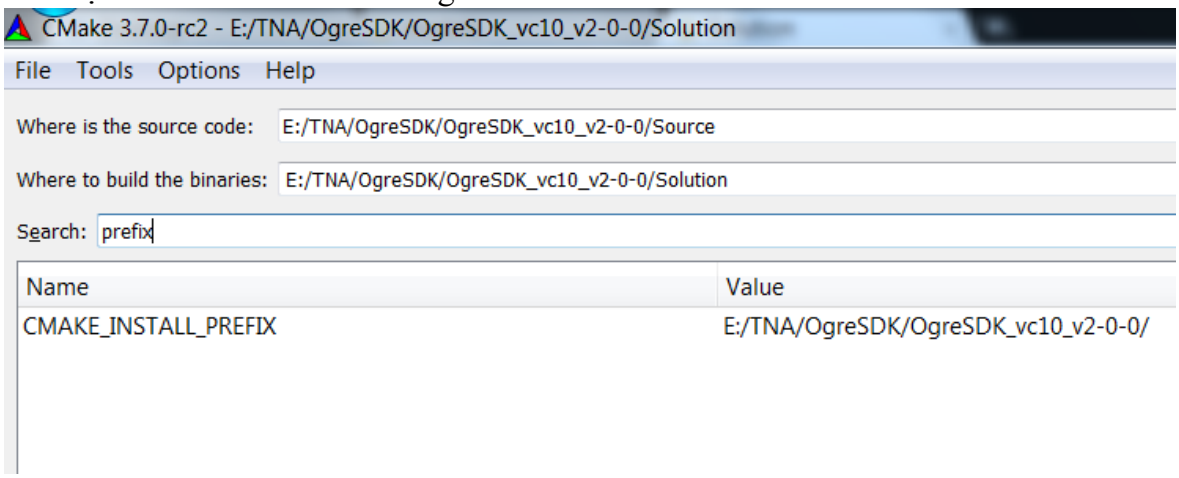
source code và đường dẫn build vào thư mục chứa SDK Ogre3D để tiện link thư viện khi build SDK Ogre3D (E:\TNA\OgreSDK\OgreSDK_vc10_v2-0-0\Dependencies\ogredeps). Tương tự, BoostSDK cũng có thể đặt vào thư mục đó. (VD: E:\TNA\OgreSDK\OgreSDK_vc10_v2-0-0\Dependencies). Configure báo lỗi thiếu BoostSDK thì chỉ cần link đường dẫn tới BoostSDK vừa build là ok. Ngoài ra, chỉnh thư mục PREFIX_FOLDER tới E:\TNA\OgreSDK\OgreSDK_vc10_v2-0-0\Dependencies\ogredeps. Đây là đường dẫn link build ra thư viện để import tiếp vào build OgreSDK.



Sau khi Greateor thành công ta build Solution OGREDEPS.sln vừa được tạo và nhận được ogredeps ở thu mục số 2.



Bước 7: Điền đường dẫn của SDK Ogre3D vừa clone về vào ô source code (VD: E:\TNA\OgreSDK\OgreSDK_vc10_v2-0-0\Source) và đường dẫn build ra Solution build SDK (E:\TNA\OgreSDK\OgreSDK_vc10_v2-0-0\Solution). Làm tương tự với hình ảnh hướng dẫn ở dưới. Lưu ý: Thư mục CMAKE_INSTALL_PREFIX sẽ thư mục build ra SDK cuối cùng.



CMake 3.7.0-rc2 - E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Solution

File Tools Options Help

Where is the source code: E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Source

Where to build the binaries: E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Solution

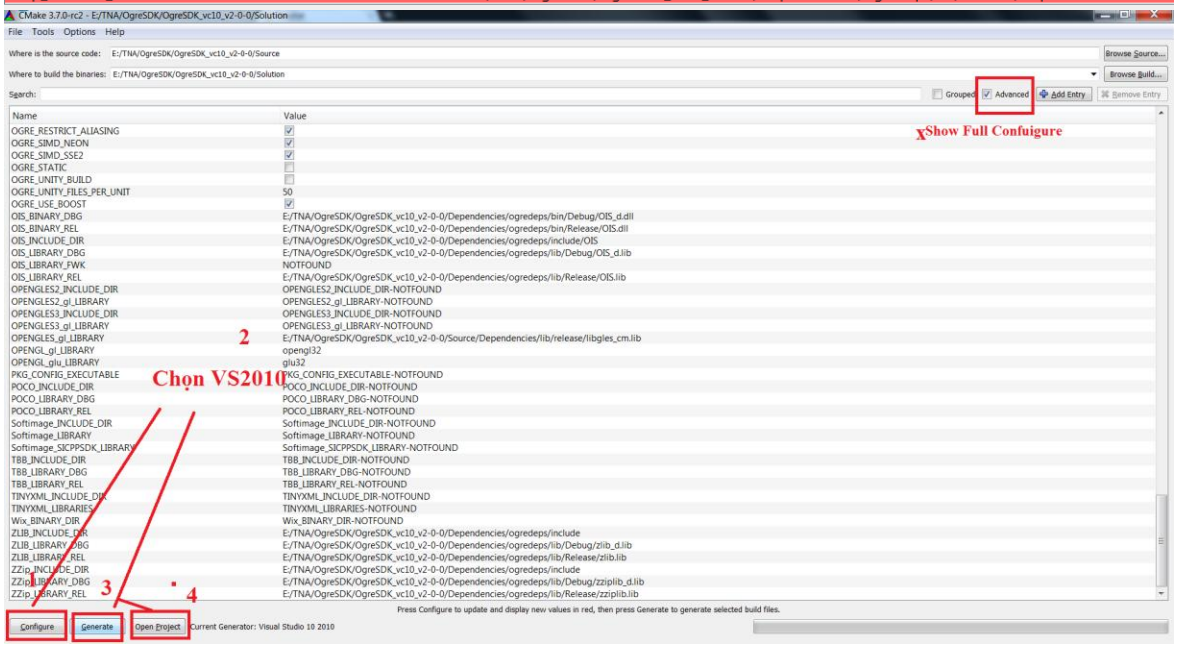
Search:

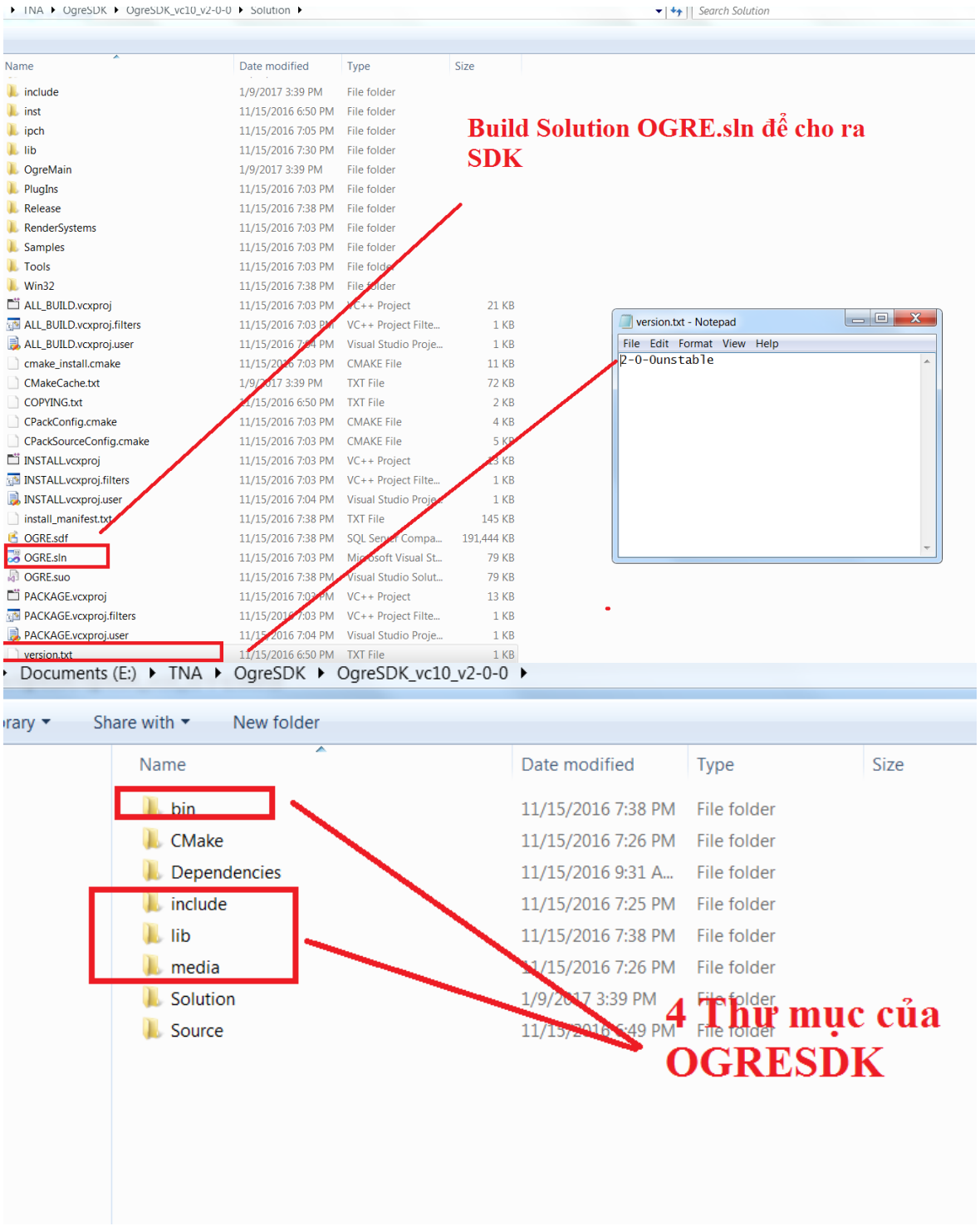
Name	Value
Boost_ATOMIC_LIBRARY_DEBUG	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/BoostSDK/boost_1_62_0/lib/libboost_atomic-vc100-mt-gd-1_62.lib
Boost_ATOMIC_LIBRARY_RELEASE	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/BoostSDK/boost_1_62_0/lib/libboost_atomic-vc100-mt-1_62.lib
Boost_CHRONO_LIBRARY_DEBUG	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/BoostSDK/boost_1_62_0/lib/libboost_chrono-vc100-mt-gd-1_62.lib
Boost_CHRONO_LIBRARY_RELEASE	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/BoostSDK/boost_1_62_0/lib/libboost_chrono-vc100-mt-1_62.lib
Boost_DATE_TIME_LIBRARY_DEBUG	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/BoostSDK/boost_1_62_0/lib/libboost_date_time-vc100-mt-gd-1_62.lib
Boost_DATE_TIME_LIBRARY_RELEASE	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/BoostSDK/boost_1_62_0/lib/libboost_date_time-vc100-mt-1_62.lib
Boost_DIR	Boost_DIR-NOTFOUND
Boost_INCLUDE_DIR	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/BoostSDK/boost_1_62_0
Boost_LIBRARY_DIR_DEBUG	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/BoostSDK/boost_1_62_0/lib
Boost_LIBRARY_DIR_RELEASE	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/BoostSDK/boost_1_62_0/lib
Boost_SYSTEM_LIBRARY_DEBUG	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/BoostSDK/boost_1_62_0/lib/libboost_system-vc100-mt-gd-1_62.lib
Boost_SYSTEM_LIBRARY_RELEASE	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/BoostSDK/boost_1_62_0/lib/libboost_system-vc100-mt-1_62.lib
Boost_THREAD_LIBRARY_DEBUG	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/BoostSDK/boost_1_62_0/lib/libboost_thread-vc100-mt-gd-1_62.lib
Boost_THREAD_LIBRARY_RELEASE	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/BoostSDK/boost_1_62_0/lib/libboost_thread-vc100-mt-1_62.lib
CMAKE_CONFIGURATION_TYPES	Debug;Release;MinSizeRel;RelWithDebInfo
CMAKE_CXX_FLAGS	/DWIN32 /D_WINDOWS /W3 /GR /EHsc
CMAKE_CXX_FLAGS_DEBUG	/D_DEBUG /MDd /Zi /Ob0 /Od /RTC1
CMAKE_CXX_FLAGS_MINSIZEREL	/MD /O1 /Ob1 /DNDEBUG
CMAKE_CXX_FLAGS_RELEASE	/MD /O2 /Ob2 /DNDEBUG
CMAKE_CXX_FLAGS_RELWITHDEBINFO	/MD /Zi /O2 /Ob1 /DNDEBUG
CMAKE_CXX_STANDARD_LIBRARIES	kernel32.lib user32.lib gdi32.lib winspool.lib shell32.lib ole32.lib oleaut32.lib uuid.lib comdlg32.lib advapi32.lib
CMAKE_C_FLAGS	/DWIN32 /D_WINDOWS /W3
CMAKE_C_FLAGS_DEBUG	/D_DEBUG /MDd /Zi /Ob0 /Od /RTC1
CMAKE_C_FLAGS_MINSIZEREL	/MD /O1 /Ob1 /DNDEBUG
CMAKE_C_FLAGS_RELEASE	/MD /O2 /Ob2 /DNDEBUG
CMAKE_C_FLAGS_RELWITHDEBINFO	/MD /Zi /O2 /Ob1 /DNDEBUG
CMAKE_C_STANDARD_LIBRARIES	kernel32.lib user32.lib gdi32.lib winspool.lib shell32.lib ole32.lib oleaut32.lib uuid.lib comdlg32.lib advapi32.lib
CMAKE_EXE_LINKER_FLAGS	/machine:X86
CMAKE_EXE_LINKER_FLAGS_DEBUG	/debug /INCREMENTAL
CMAKE_EXE_LINKER_FLAGS_MINSIZEREL	/INCREMENTAL:NO
CMAKE_EXE_LINKER_FLAGS_RELEASE	/INCREMENTAL:NO
CMAKE_EXE_LINKER_FLAGS_RELWITHDEBINFO	/debug /INCREMENTAL
CMAKE_INSTALL_PREFIX	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/
CMAKE_LINKER	C:/Program Files (x86)/Microsoft Visual Studio 10.0/VC/bin/link.exe
CMAKE_MODULE_LINKER_FLAGS	/machine:X86
CMAKE_MODULE_LINKER_FLAGS_DEBUG	/debug /INCREMENTAL
CMAKE_MODULE_LINKER_FLAGS_MINSIZEREL	/INCREMENTAL:NO
CMAKE_MODULE_LINKER_FLAGS_RELEASE	/INCREMENTAL:NO
CMAKE_MODULE_LINKER_FLAGS_RELWITHDEBINFO	/debug /INCREMENTAL
CMAKE_RC_COMPILER	rc
CMAKE_RC_FLAGS	/DWIN32
CMAKE_SHARED_LINKER_FLAGS	/machine:X86
CMAKE_SHARED_LINKER_FLAGS_DEBUG	/debug /INCREMENTAL
CMAKE_SHARED_LINKER_FLAGS_MINSIZEREL	/INCREMENTAL:NO
CMAKE_SHARED_LINKER_FLAGS_RELEASE	/INCREMENTAL:NO
CMAKE_SHARED_LINKER_FLAGS_RELWITHDEBINFO	/debug /INCREMENTAL
CMAKE_SKIP_INSTALL_RPATH	☐
CMAKE_SKIP_RPATH	☐
CMAKE_STATIC_LINKER_FLAGS	/machine:X86
CMAKE_STATIC_LINKER_FLAGS_DEBUG	
CMAKE_STATIC_LINKER_FLAGS_MINSIZEREL	
CMAKE_STATIC_LINKER_FLAGS_RELEASE	
CMAKE_STATIC_LINKER_FLAGS_RELWITHDEBINFO	
CMAKE_VERBOSE_MAKEFILE	☐
CPACK_BINARY_7Z	☐
CPACK_BINARY_IFW	☐
CPACK_BINARY_NSIS	☑
CPACK_BINARY_WIX	☐
CPACK_BINARY_ZIP	☐
CPACK_SOURCE_7Z	☑
CPACK_SOURCE_ZIP	☑
Cg_BINARY_DBG	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/bin/Debug/cg.dll
Cg_BINARY_REL	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/bin/Release/cg.dll
Cg_INCLUDE_DIR	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/include/Cg
Cg_LIBRARY_DBG	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/lib/Debug/cg.lib
Cg_LIBRARY_FWK	
Cg_LIBRARY_REL	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/lib/Release/cg.lib
CppUnit_INCLUDE_DIR	CppUnit_INCLUDE_DIR-NOTFOUND
CppUnit_LIBRARY_DBG	CppUnit_LIBRARY_DBG-NOTFOUND
CppUnit_LIBRARY_REL	CppUnit_LIBRARY_REL-NOTFOUND
DOXYGEN_DOT_EXECUTABLE	DOXYGEN_DOT_EXECUTABLE-NOTFOUND
DOXYGEN_EXECUTABLE	DOXYGEN_EXECUTABLE-NOTFOUND
DirectX11_D3DCOMPILER_LIBRARY	C:/Program Files (x86)/Microsoft DirectX SDK (June 2010)/Lib/x86/d3dcompiler.lib
DirectX11_D3DX11_LIBRARY	C:/Program Files (x86)/Microsoft DirectX SDK (June 2010)/Lib/x86/d3dx11.lib
DirectX11_DXERR_LIBRARY	C:/Program Files (x86)/Microsoft DirectX SDK (June 2010)/Lib/x86/DxErr.lib
DirectX11_DXGI_LIBRARY	C:/Program Files (x86)/Microsoft DirectX SDK (June 2010)/Lib/x86/dxgi.lib
DirectX11_DXGUID_LIBRARY	C:/Program Files (x86)/Microsoft DirectX SDK (June 2010)/Lib/x86/dxguid.lib
DirectX11_INCLUDE_DIR	C:/Program Files (x86)/Microsoft DirectX SDK (June 2010)/Include

Name	Value
DirectX11_LIBRARY	C:/Program Files (x86)/Microsoft DirectX SDK (June 2010)/Lib/x86/d3d11.lib
DirectX9_D3DX9_LIBRARY	C:/Program Files (x86)/Microsoft DirectX SDK (June 2010)/Lib/x86/d3dx9.lib
DirectX9_DXERR_LIBRARY	C:/Program Files (x86)/Microsoft DirectX SDK (June 2010)/Lib/x86/DxErr.lib
DirectX9_DXGUID_LIBRARY	C:/Program Files (x86)/Microsoft DirectX SDK (June 2010)/Lib/x86/dxguid.lib
DirectX9_INCLUDE_DIR	C:/Program Files (x86)/Microsoft DirectX SDK (June 2010)/Include
DirectX9_LIBRARY	C:/Program Files (x86)/Microsoft DirectX SDK (June 2010)/Lib/x86/d3d9.lib
EGL_INCLUDE_DIR	EGL_INCLUDE_DIR-NOTFOUND
EGL_egl_LIBRARY	EGL_egl_LIBRARY-NOTFOUND
FREETYPE_FT2BUILD_INCLUDE_DIR	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/include
FREETYPE_INCLUDE_DIR	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/include/freetype
FREETYPE_LIBRARY_DBG	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/lib/Debug/freetype_d.lib
FREETYPE_LIBRARY_REL	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/lib/Release/freetype.lib
FreeImage_INCLUDE_DIR	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/include/freetype
FreeImage_LIBRARY_DBG	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/lib/Debug/FreeImage_d.lib
FreeImage_LIBRARY_REL	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/lib/Release/FreeImage.lib
GLSL_Optimizer_INCLUDE_DIR	GLSL_Optimizer_INCLUDE_DIR-NOTFOUND
GLSL_Optimizer_LIBRARY_DBG	GLSL_Optimizer_LIBRARY_DBG-NOTFOUND
GLSL_Optimizer_LIBRARY_FWK	NOTFOUND
GLSL_Optimizer_LIBRARY_REL	GLSL_Optimizer_LIBRARY_REL-NOTFOUND
HLSL2GLSL_INCLUDE_DIR	HLSL2GLSL_INCLUDE_DIR-NOTFOUND
HLSL2GLSL_LIBRARY_DBG	HLSL2GLSL_LIBRARY_DBG-NOTFOUND
HLSL2GLSL_LIBRARY_FWK	NOTFOUND
HLSL2GLSL_LIBRARY_REL	HLSL2GLSL_LIBRARY_REL-NOTFOUND
OGRE_ASSERT_MODE	0
OGRE_BUILD_COMPONENT_MESHLODGENERATOR	☑
OGRE_BUILD_COMPONENT_OVERLAY	☑
OGRE_BUILD_COMPONENT_PAGING	☑
OGRE_BUILD_COMPONENT_PROPERTY	☑
OGRE_BUILD_COMPONENT_RTSHADERSYSTEM	☑
OGRE_BUILD_COMPONENT_TERRAIN	☑
OGRE_BUILD_COMPONENT_VOLUME	☑
OGRE_BUILD_MSVC_MP	☑
OGRE_BUILD_MSVC_ZM	☑
OGRE_BUILD_PLATFORM_NACL	☐
OGRE_BUILD_PLATFORM_WINDOWS_PHONE	☐
OGRE_BUILD_PLATFORM_WINRT	☐
OGRE_BUILD_PLUGIN_CG	☑
OGRE_BUILD_PLUGIN_PFX	☑
OGRE_BUILD_RENDERSYSTEM_D3D11	☑

Name	Value
OGRE_BUILD_RENDERSYSTEM_D3D11	☑
OGRE_BUILD_RENDERSYSTEM_D3D9	☑
OGRE_BUILD_RENDERSYSTEM_GL	☑
OGRE_BUILD_RENDERSYSTEM_GL3PLUS	☐
OGRE_BUILD_RENDERSYSTEM_GLES	☐
OGRE_BUILD_RENDERSYSTEM_GLES2	☐
OGRE_BUILD_RTSHADERSYSTEM_CORE_SHADERS	☑
OGRE_BUILD_RTSHADERSYSTEM_EXT_SHADERS	☑
OGRE_BUILD_SAMPLES	☑
OGRE_BUILD_TESTS	☐
OGRE_BUILD_TOOLS	☑
OGRE_CONFIG_ALLOCATOR	4
OGRE_CONFIG_CONTAINERS_USE_CUSTOM_ALLOCATOR	☑
OGRE_CONFIG_DOUBLE	☐
OGRE_CONFIG_ENABLE_DDS	☑
OGRE_CONFIG_ENABLE_ETC	☐
OGRE_CONFIG_ENABLE_FREEIMAGE	☑
OGRE_CONFIG_ENABLE_GL_STATE_CACHE_SUPPORT	☐
OGRE_CONFIG_ENABLE_MESHLOD	☑
OGRE_CONFIG_ENABLE_PVRTC	☐
OGRE_CONFIG_ENABLE_VIEWPORT_ORIENTATIONMODE	☐
OGRE_CONFIG_ENABLE_ZIP	☑
OGRE_CONFIG_MEMTRACK_DEBUG	☐
OGRE_CONFIG_MEMTRACK_RELEASE	☐
OGRE_CONFIG_STATIC_LINK_CRT	☐
OGRE_CONFIG_STRING_USE_CUSTOM_ALLOCATOR	☐
OGRE_CONFIG_THREADS	2
OGRE_CONFIG_THREAD_PROVIDER	boost
OGRE_COPY_DEPENDENCIES	☑
OGRE_DEPENDENCIES_DIR	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps
OGRE_INSTALL_DEPENDENCIES	☑
OGRE_INSTALL_DOCS	☑
OGRE_INSTALL_PDB	☑
OGRE_INSTALL_SAMPLES	☑
OGRE_INSTALL_SAMPLES_SOURCE	☐
OGRE_INSTALL_TOOLS	☑
OGRE_INSTALL_VSPROPS	☐
OGRE_LEGACY_ANIMATIONS	☑
OGRE_LIB_DIRECTORY	lib

Name	Value
OGRE_RESTRICT_ALIASING	☑
OGRE_SIMD_NEON	☑
OGRE_SIMD_SSE2	☑
OGRE_STATIC	☐
OGRE_UNITY_BUILD	☐
OGRE_UNITY_FILES_PER_UNIT	50
OGRE_USE_BOOST	☑
OIS_BINARY_DBG	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/bin/Debug/OIS_d.dll
OIS_BINARY_REL	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/bin/Release/OIS.dll
OIS_INCLUDE_DIR	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/include/OIS
OIS_LIBRARY_DBG	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/lib/Debug/OIS_d.lib
OIS_LIBRARY_FWK	NOTFOUND
OIS_LIBRARY_REL	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/lib/Release/OIS.lib
OPENGLES2_INCLUDE_DIR	OPENGLES2_INCLUDE_DIR-NOTFOUND
OPENGLES2_gl_LIBRARY	OPENGLES2_gl_LIBRARY-NOTFOUND
OPENGLES3_INCLUDE_DIR	OPENGLES3_INCLUDE_DIR-NOTFOUND
OPENGLES3_gl_LIBRARY	OPENGLES3_gl_LIBRARY-NOTFOUND
OPENGLES_gl_LIBRARY	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Source/Dependencies/lib/release/libgles_cm.lib
OPENGL_gl_LIBRARY	opengl32
OPENGL_glu_LIBRARY	glu32
PKG_CONFIG_EXECUTABLE	PKG_CONFIG_EXECUTABLE-NOTFOUND
POCO_INCLUDE_DIR	POCO_INCLUDE_DIR-NOTFOUND
POCO_LIBRARY_DBG	POCO_LIBRARY_DBG-NOTFOUND
POCO_LIBRARY_REL	POCO_LIBRARY_REL-NOTFOUND
Softimage_INCLUDE_DIR	Softimage_INCLUDE_DIR-NOTFOUND
Softimage_LIBRARY	Softimage_LIBRARY-NOTFOUND
Softimage_SICPPSDK_LIBRARY	Softimage_SICPPSDK_LIBRARY-NOTFOUND
TBB_INCLUDE_DIR	TBB_INCLUDE_DIR-NOTFOUND
TBB_LIBRARY_DBG	TBB_LIBRARY_DBG-NOTFOUND
TBB_LIBRARY_REL	TBB_LIBRARY_REL-NOTFOUND
TINYXML_INCLUDE_DIR	TINYXML_INCLUDE_DIR-NOTFOUND
TINYXML_LIBRARIES	TINYXML_LIBRARIES-NOTFOUND
Wix_BINARY_DIR	Wix_BINARY_DIR-NOTFOUND
ZLIB_INCLUDE_DIR	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/include
ZLIB_LIBRARY_DBG	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/lib/Debug/zlib_d.lib
ZLIB_LIBRARY_REL	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/lib/Release/zlib.lib
ZZip_INCLUDE_DIR	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/include
ZZip_LIBRARY_DBG	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/lib/Debug/zziplib_d.lib
ZZip_LIBRARY_REL	E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/Dependencies/ogredeps/lib/Release/zziplib.lib





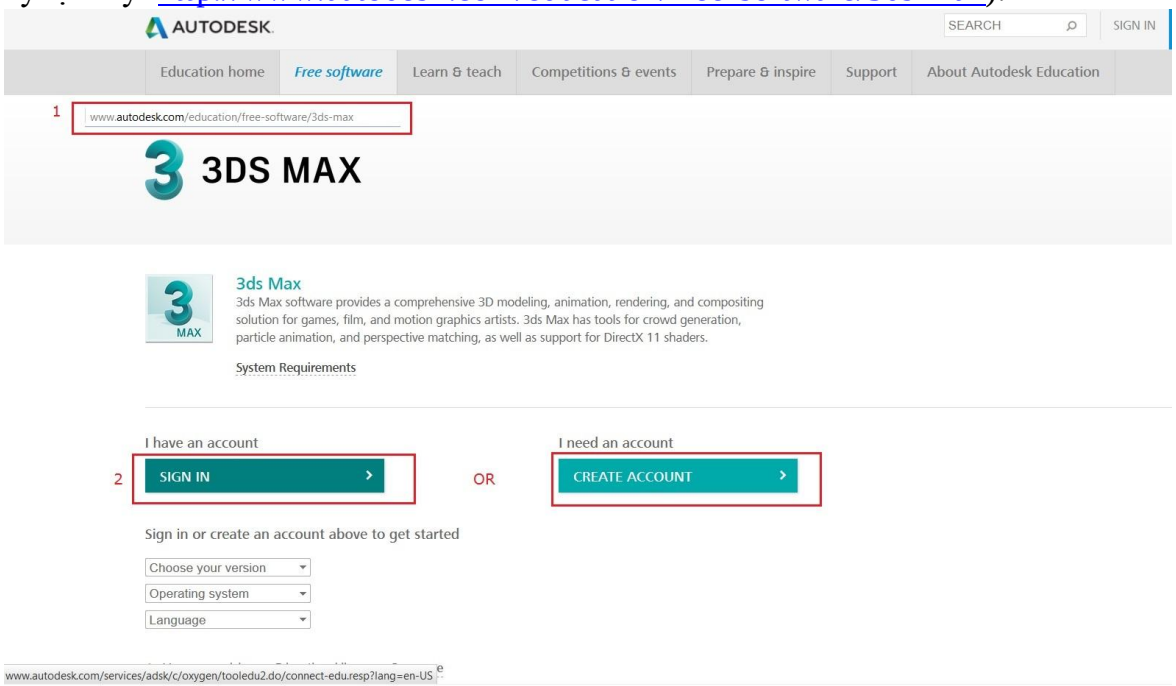
Cuối cùng, ta có thư viện OgreSDK_vc10_v2-0-0 version 2.0 tương thích với compiler Microsoft Visual C++ 2010 Service Pack 1 đã bao gồm thư viện Boost, OIS, SDKTray, OpenGL (Mặc định theo kèm), DirectX9 và DirectX11.

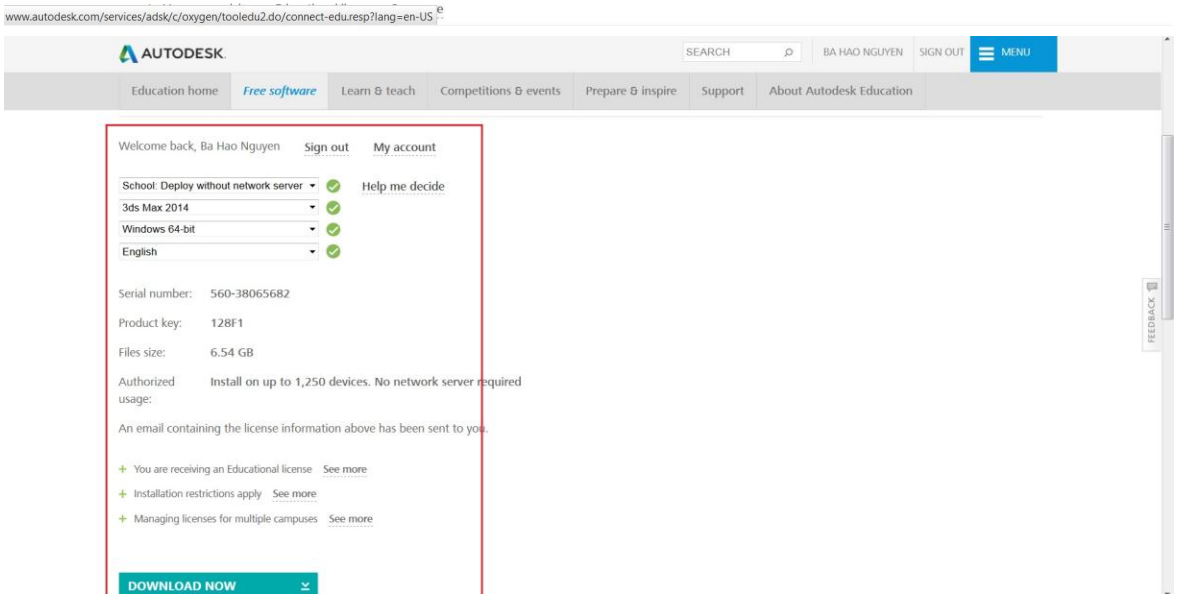
1.4.b, Cài đặt thư viện âm thanh irrKlang

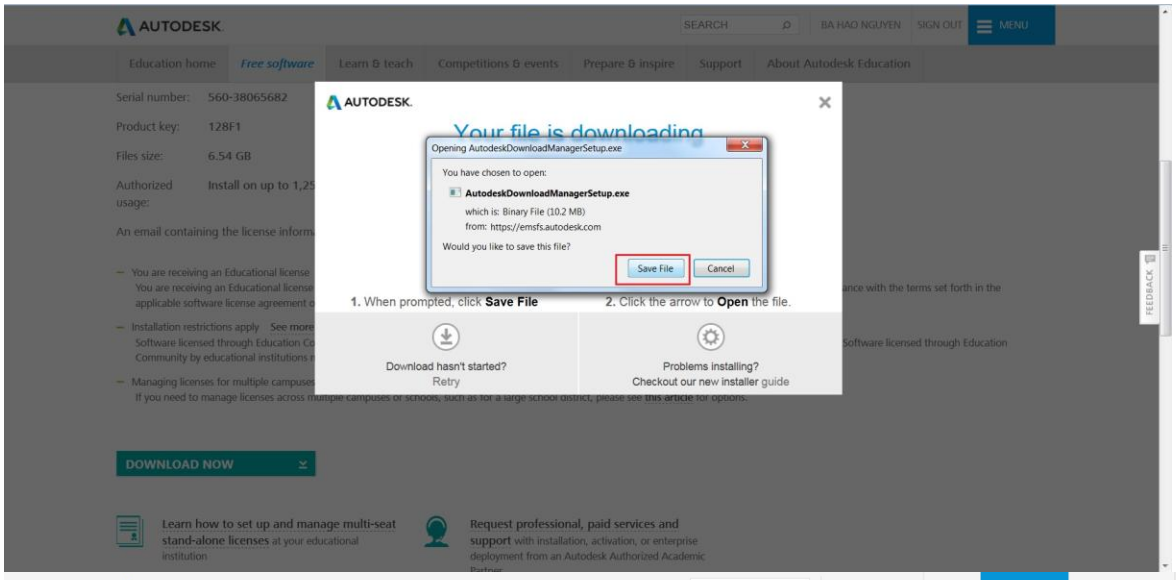
Bước 1, Download thư viện mới nhất tại đường dẫn: http://www.ambiera.com/irrklang/downloads.html. Sau đó, giải nén vào thư mục E:\TNA\OgreSDK\OgreSDK_vc10_v2-0-0\Dependencies\IrrKlangSDK.
Bước 2: Copy các thư viện *.dll của IrrKlang vào ổ E:\TNA\OgreSDK\OgreSDK_vc10_v2-0-0/bin/Debug (cả Release). Sau đó, link thư viện include và lib của IrrKlang vào dự án là có thể sử dụng thư viện IrrKlang như các thư viện khác trong OgreSDK.

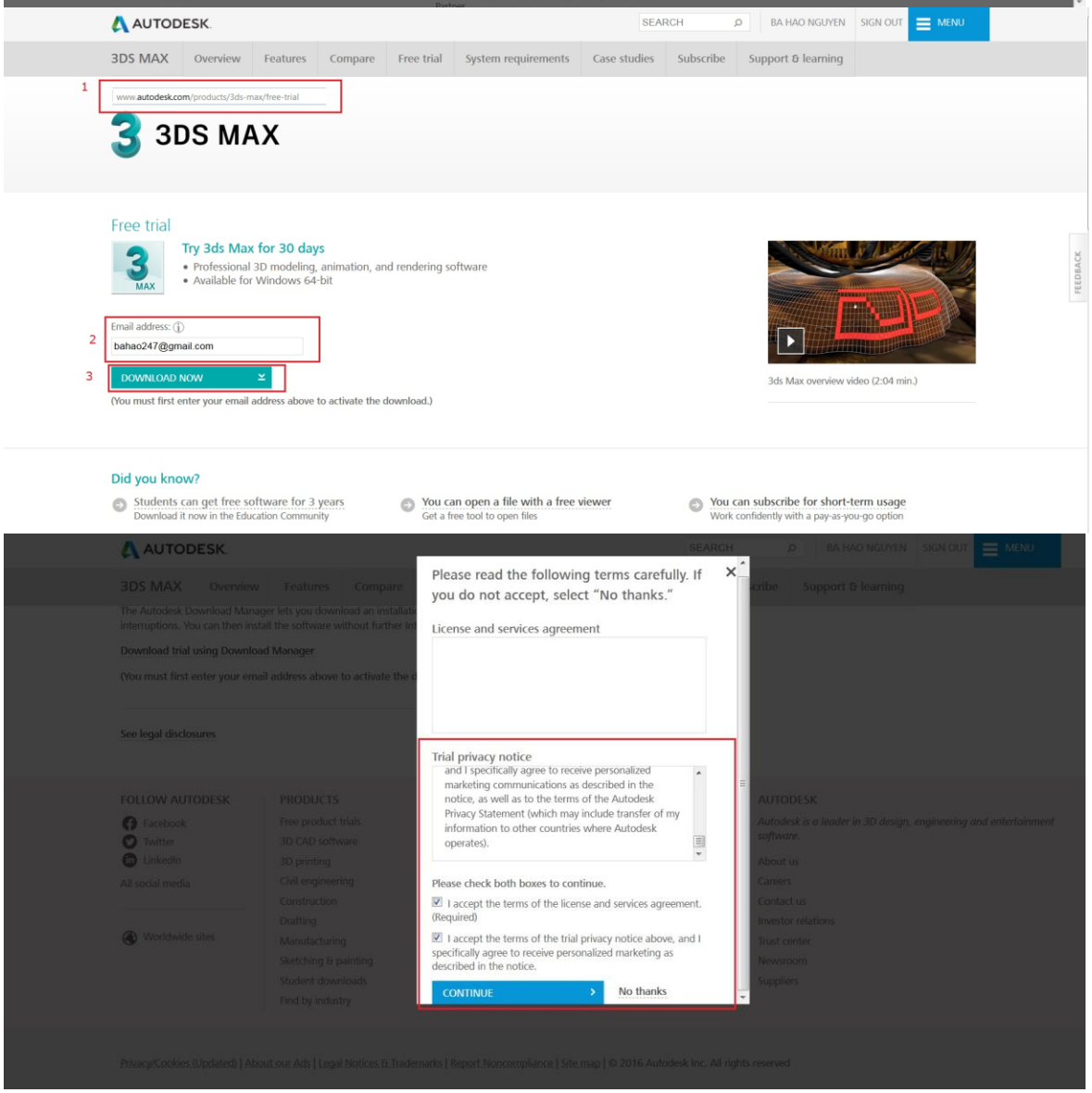
1.4.c, Cài đặt tool 3ds Max

Bước 1: Đăng ký 3ds Max, để dùng miễn phí bản dành cho sinh viên học tập (Đăng ký tại đây: http://www.autodesk.com/education/free-software/3ds-max).

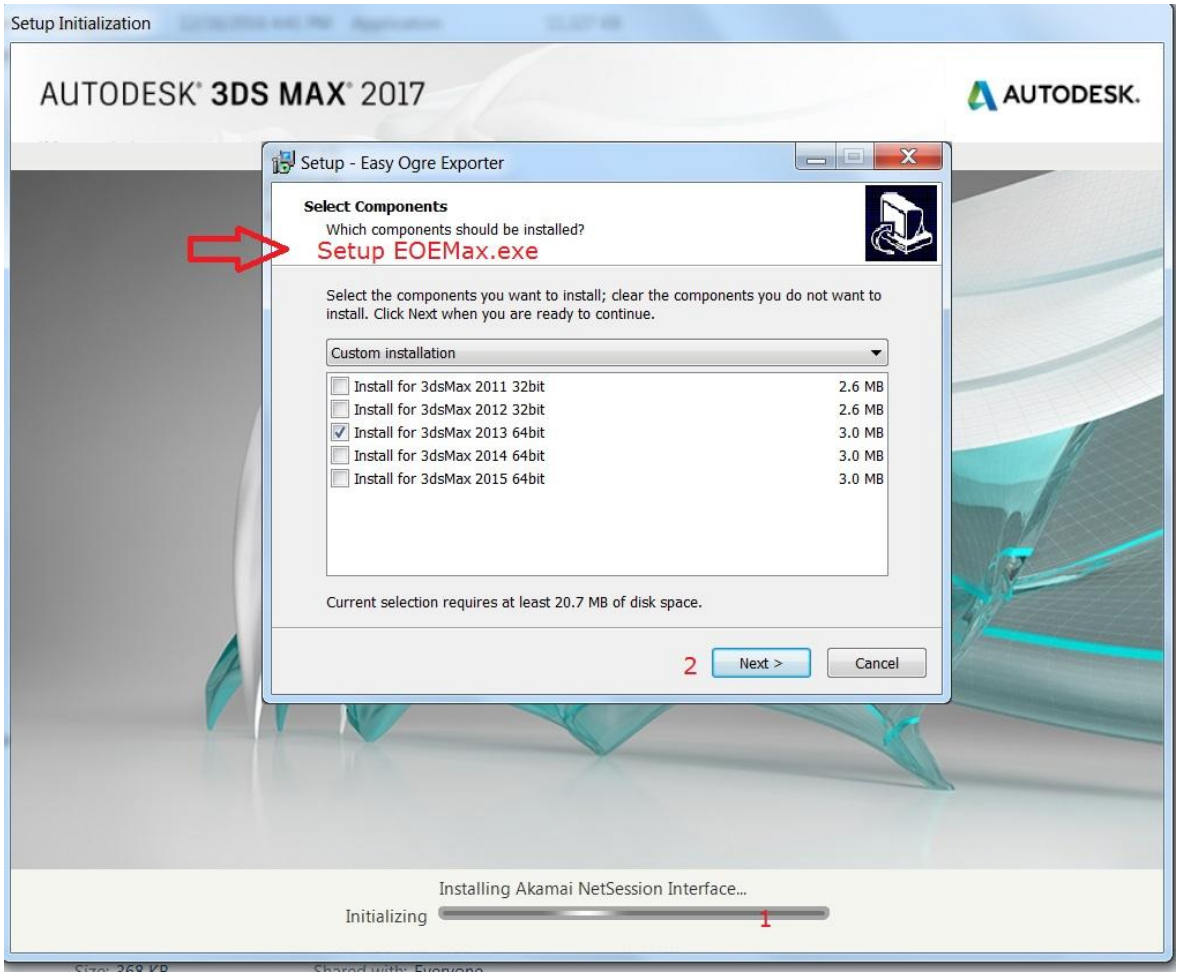


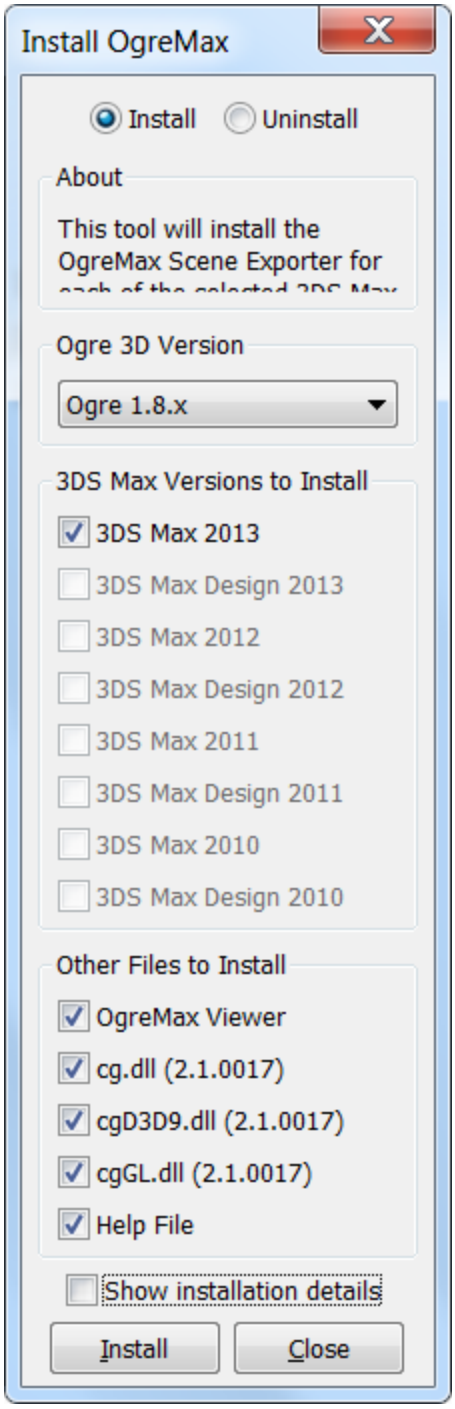



AUTODESK.
Education home
Free software
Learn & teach
Competitions & events
Prepare & inspire
Support
About Autodesk Education
Serial number: 560-38065682
Product key: 128F1
Files size: 6.54 GB
Authorized usage: Install on up to 1,25
AUTODESK.
Your file is downloading
Opening AutodeskDownloadManagerSetup.exe
You have chosen to open:
AutodeskDownloadManagerSetup.exe
which is: Binary File (10.2 MB)
from: https://emsfs.autodesk.com
Would you like to save this file?
Save File
Cancel
1. When prompted, click Save File
2. Click the arrow to Open the file.
Download hasn't started?
Retry
Problems installing?
Checkout our new installer guide
DOWNLOAD NOW
Learn how to set up and manage multi-seat stand-alone licenses at your educational institution
Request professional, paid services and support with installation, activation, or enterprise deployment from an Autodesk Authorized Academic

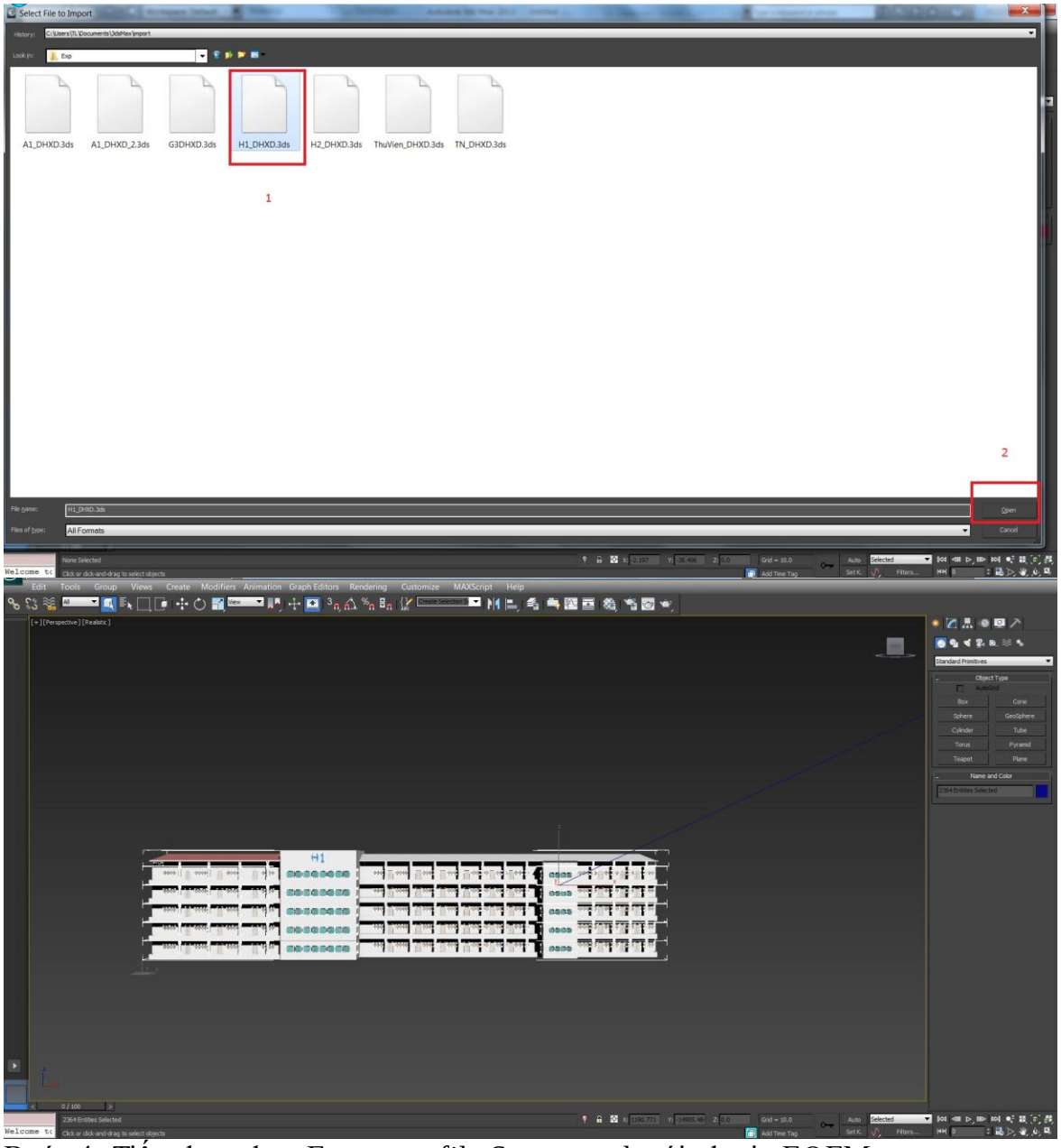

AUTODESK.
SEARCH
BA HAO NGUYEN
SIGN OUT
MENU
3DS MAX
Overview
Features
Compare
Free trial
System requirements
Case studies
Subscribe
Support & learning
1
www.autodesk.com/products/3ds-max/free-trial
3DS MAX
Free trial
Try 3ds Max for 30 days
Professional 3D modeling, animation, and rendering software
Available for Windows 64-bit
Email address:
2
bahao247@gmail.com
3
DOWNLOAD NOW
(You must first enter your email address above to activate the download.)
3ds Max overview video (2:04 min.)
Did you know?
Students can get free software for 3 years
Download it now in the Education Community
You can open a file with a free viewer
Get a free tool to open files
You can subscribe for short-term usage
Work confidently with a pay-as-you-go option
Please read the following terms carefully. If you do not accept, select "No thanks."
License and services agreement
Trial privacy notice
and I specifically agree to receive personalized marketing communications as described in the notice, as well as to the terms of the Autodesk Privacy Statement (which may include transfer of my information to other countries where Autodesk operates).
Please check both boxes to continue.
I accept the terms of the license and services agreement. (Required)
I accept the terms of the trial privacy notice above, and I specifically agree to receive personalized marketing as described in the notice.
CONTINUE
No thanks

Bước 2: Sau khi cài đặt 3DS Max (hỗ trợ bản 2011-2015) xong thì cài thêm plugin EOEMax.exe (Link download: https://goo.gl/au13Xv) hoặc OgreMaxSceneExporterInstaller (Link download: https://goo.gl/mIygW7) để export ra được file *.mesh load vào trong game.

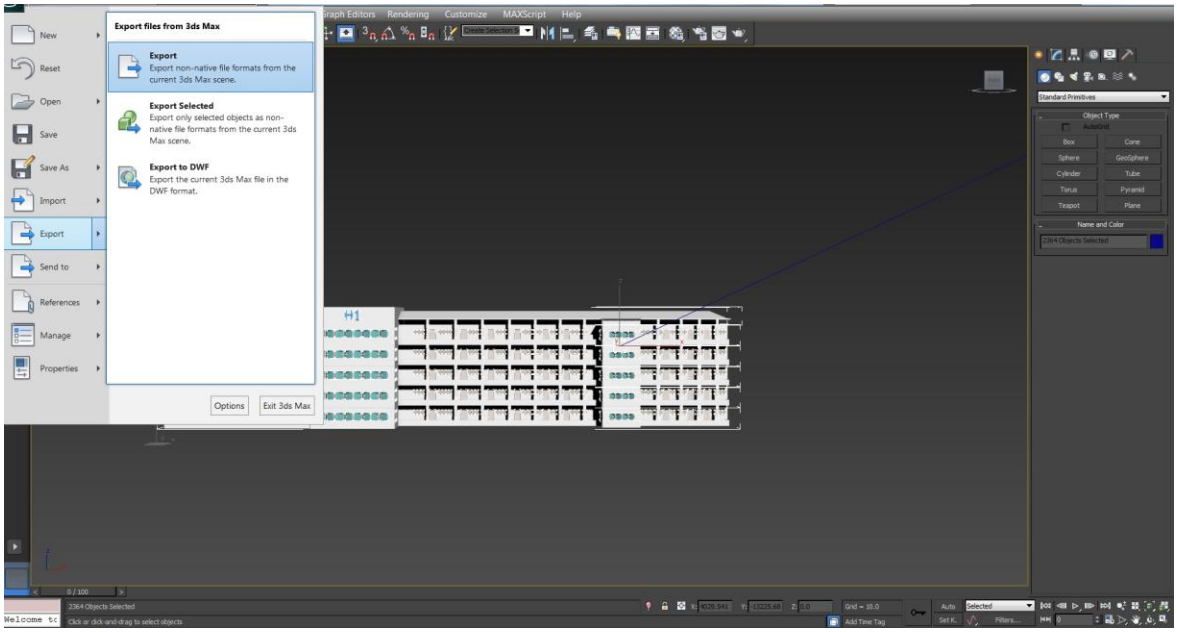


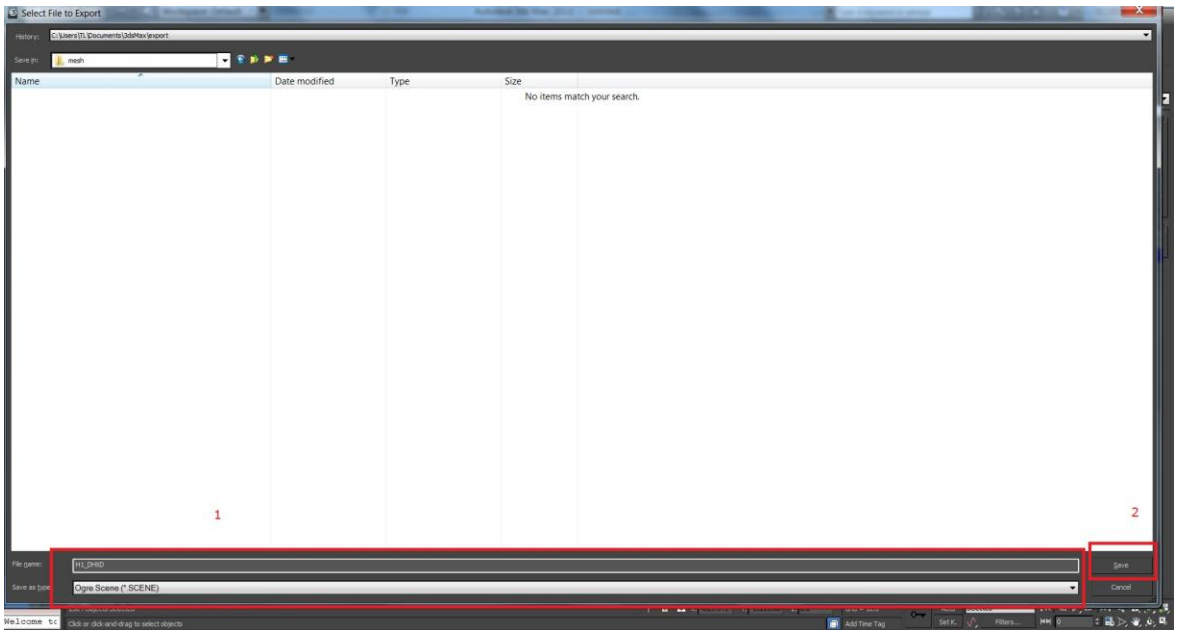


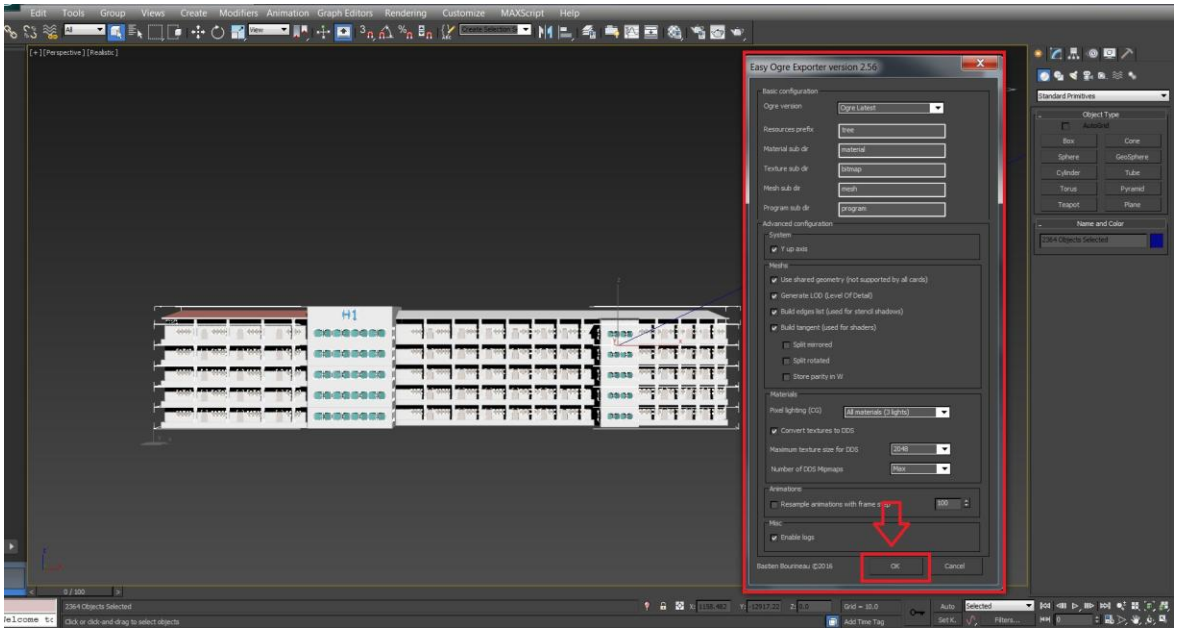
Bước 3: Import thử một file 3DS



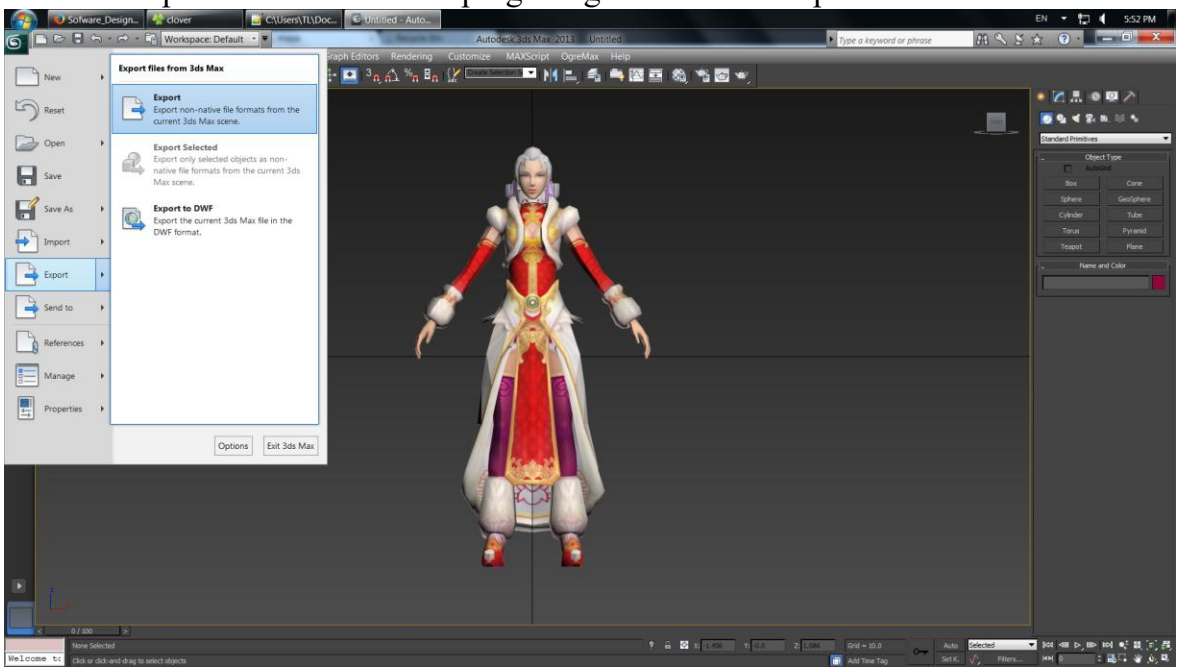
Bước 4: Tiếp theo chọn Export ra file Scene mesh với plugin EOEMax.exe.

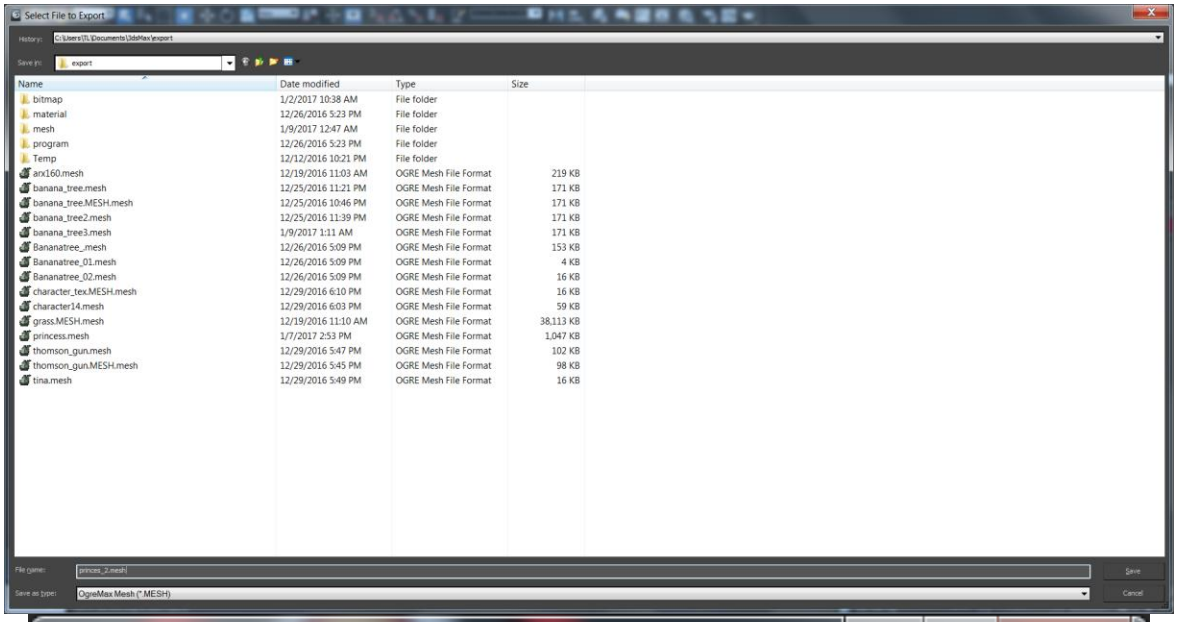

Export files from 3ds Max
Export
Export non-native file formats from the current 3ds Max scene.
Export Selected
Export only selected objects as non-native file formats from the current 3ds Max scene.
Export to DWF
Export the current 3ds Max file in the DWF format.
New
Reset
Open
Save
Save As
Import
Export
Send to
References
Manage
Properties
Options
Exit 3ds Max
Graph Editors
Rendering
Customize
MAXScript
Help
H1

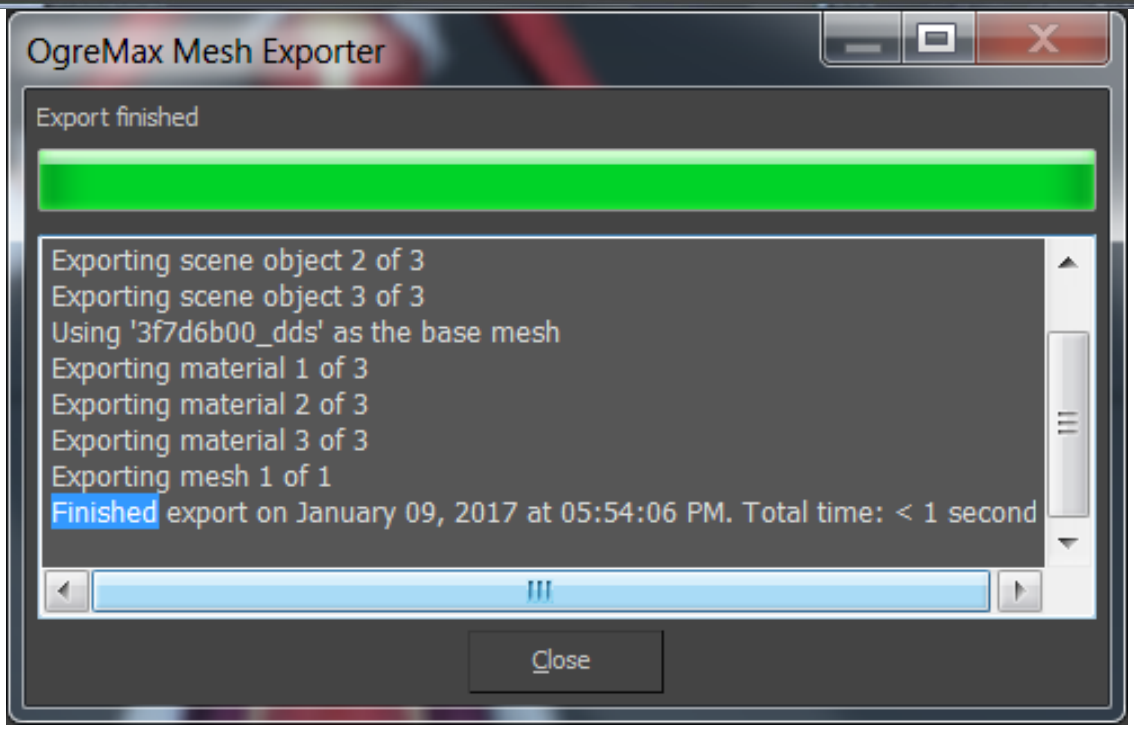

Select File to Export
Name
Date modified
Type
Size
No items match your search.
1
2
File name:
H1_0H0
Save as type:
Ogre Scene (*.SCENE)
Save
Cancel

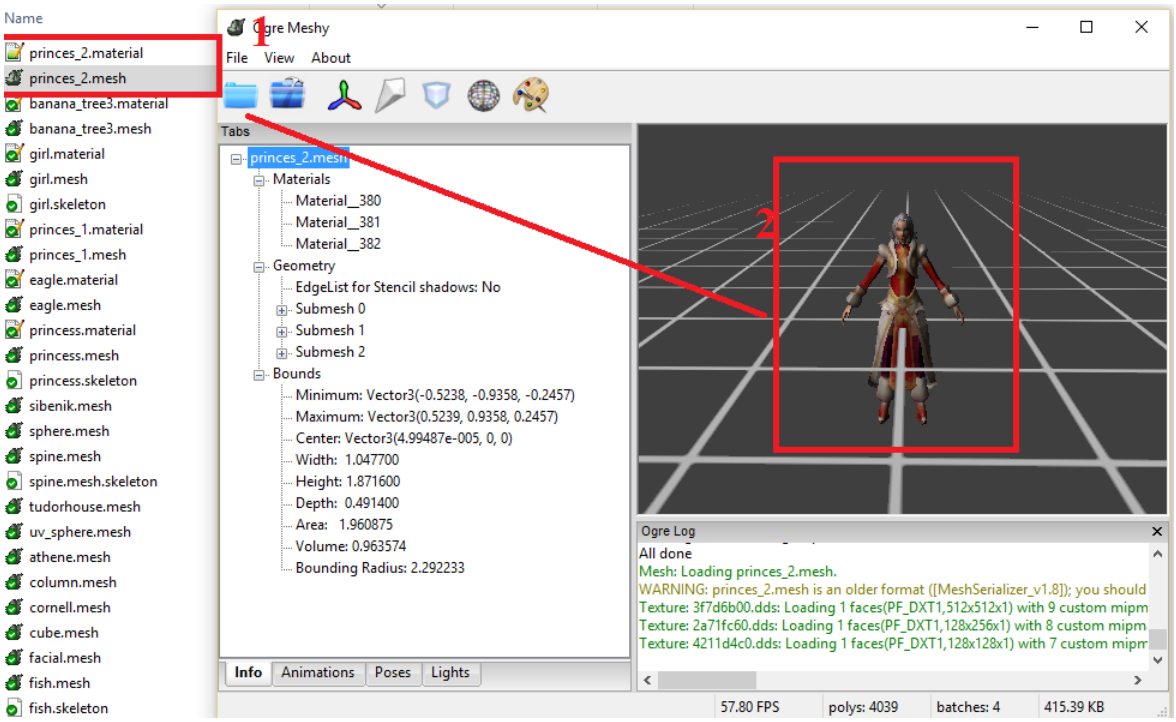


Bước 4: Export ra file mesh với plugin OgreMaxSceneExporterInstaller.




Select File to Export
History: C:\Users\TL\Documents\3dsMax\export
Save in: export
Name
Date modified
Type
Size
bitmap 1/2/2017 10:38 AM File folder
material 12/26/2016 5:23 PM File folder
mesh 1/9/2017 12:47 AM File folder
program 12/26/2016 5:23 PM File folder
Temp 12/12/2016 10:21 PM File folder
arx160.mesh 12/19/2016 11:03 AM OGRE Mesh File Format 219 KB
banana_tree.mesh 12/25/2016 11:21 PM OGRE Mesh File Format 171 KB
banana_tree.MESH.mesh 12/25/2016 10:46 PM OGRE Mesh File Format 171 KB
banana_tree2.mesh 12/25/2016 11:39 PM OGRE Mesh File Format 171 KB
banana_tree3.mesh 1/9/2017 1:11 AM OGRE Mesh File Format 171 KB
Bananatree_mesh 12/26/2016 5:09 PM OGRE Mesh File Format 153 KB
Bananatree_01.mesh 12/26/2016 5:09 PM OGRE Mesh File Format 4 KB
Bananatree_02.mesh 12/26/2016 5:09 PM OGRE Mesh File Format 16 KB
character_tex.MESH.mesh 12/29/2016 6:10 PM OGRE Mesh File Format 16 KB
character14.mesh 12/29/2016 6:03 PM OGRE Mesh File Format 59 KB
grass.MESH.mesh 12/19/2016 11:10 AM OGRE Mesh File Format 38,113 KB
princess.mesh 1/7/2017 2:53 PM OGRE Mesh File Format 1,047 KB
thomson_gun.mesh 12/29/2016 5:47 PM OGRE Mesh File Format 102 KB
thomson_gun.MESH.mesh 12/29/2016 5:45 PM OGRE Mesh File Format 98 KB
tina.mesh 12/29/2016 5:49 PM OGRE Mesh File Format 16 KB
File name: princes_2.mesh
Save as type: OgreMax Mesh (*.MESH)
Save
Cancel

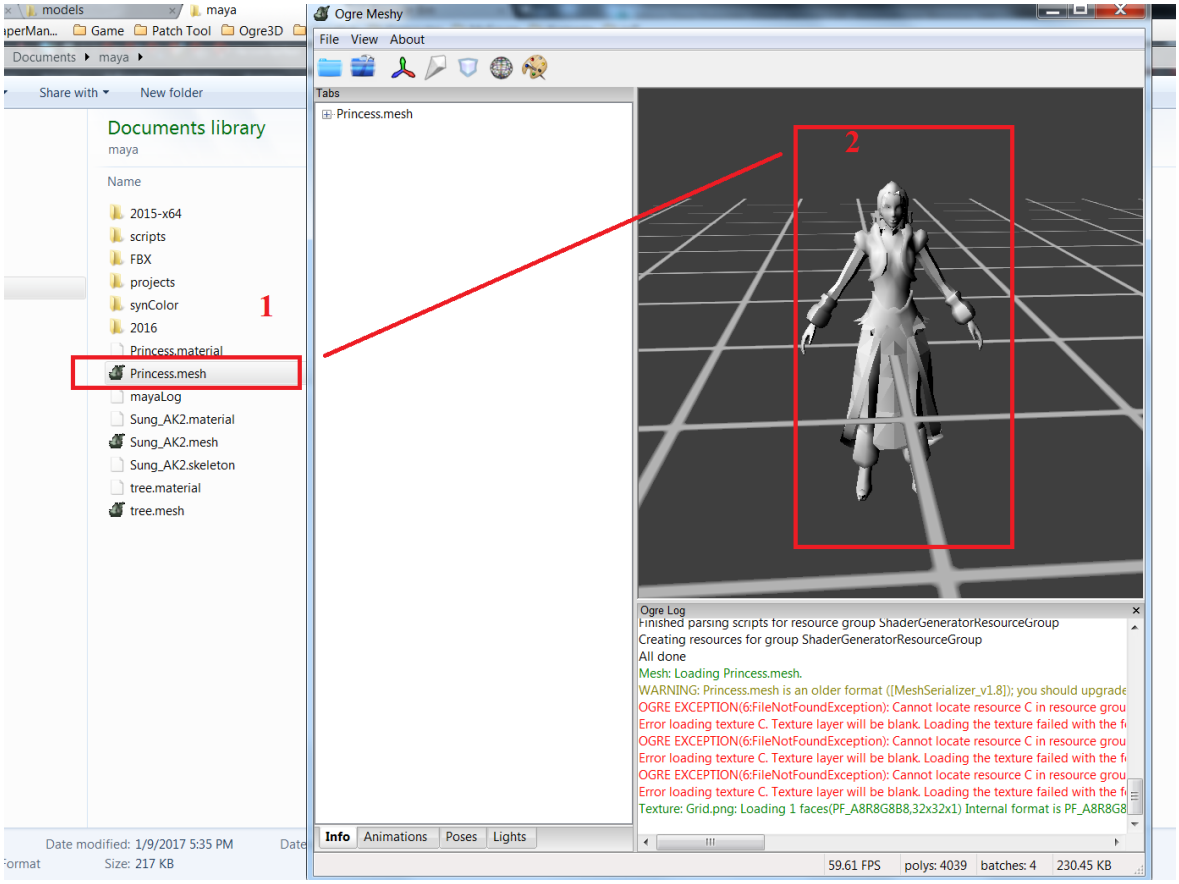

OgreMax Mesh Exporter
Export finished
Exporting scene object 2 of 3
Exporting scene object 3 of 3
Using '3f7d6b00_dds' as the base mesh
Exporting material 1 of 3
Exporting material 2 of 3
Exporting material 3 of 3
Exporting mesh 1 of 1
Finished export on January 09, 2017 at 05:54:06 PM. Total time: < 1 second
Close



1.4.d, Cài đặt tool Maya

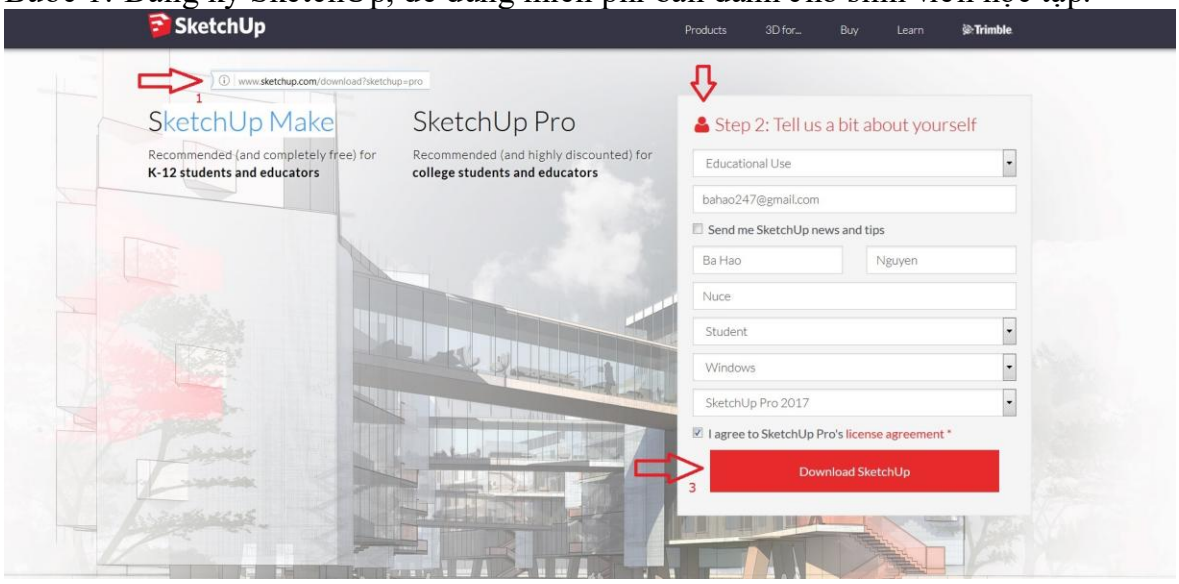
Bước 1: Đăng ký Maya, để dùng miễn phí bản dành cho sinh viên học tập (Đăng ký tại: http://www.autodesk.com/education/free-software/maya).

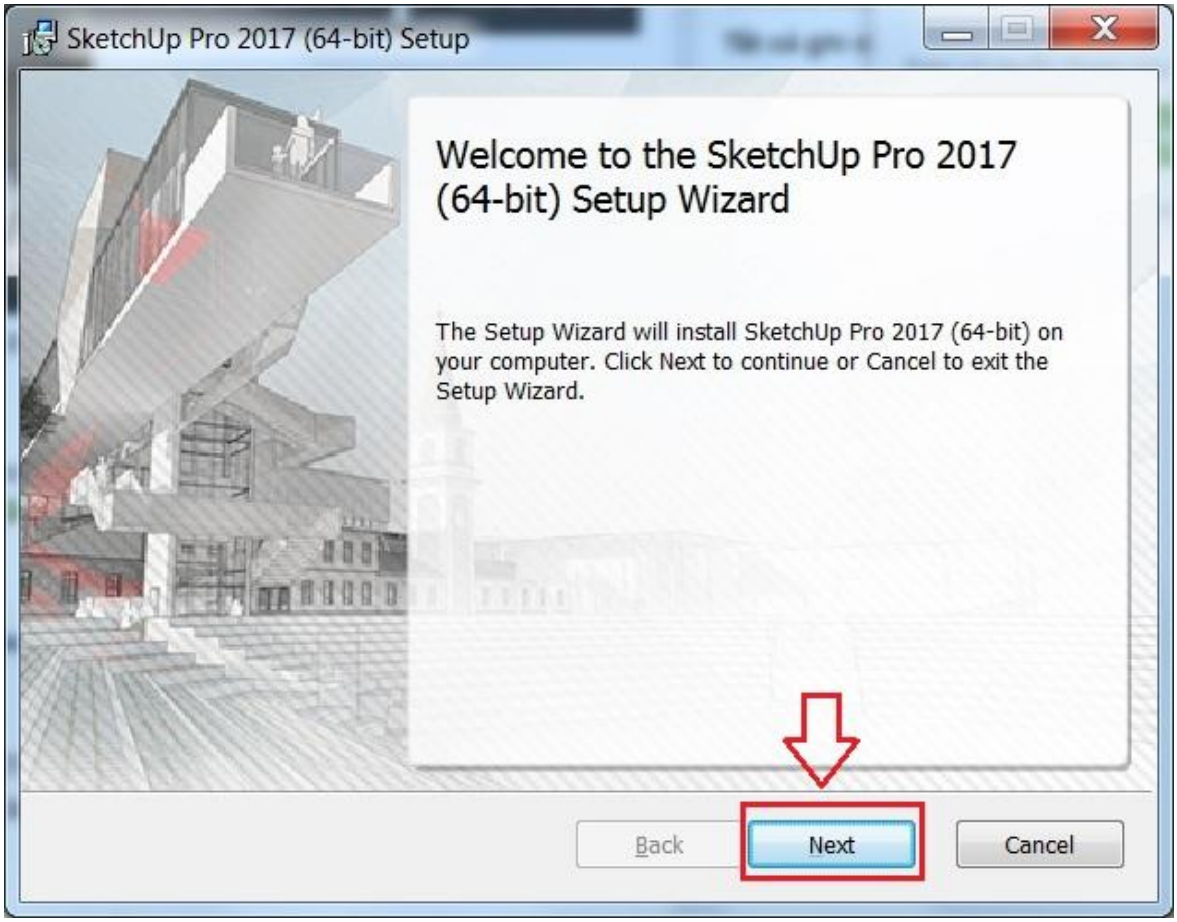
Bước 2: Cài đặt plugin Maya Ogre Exporter (Download tại: https://goo.gl/TxVzss) cho Maya để export ra file *.mesh.

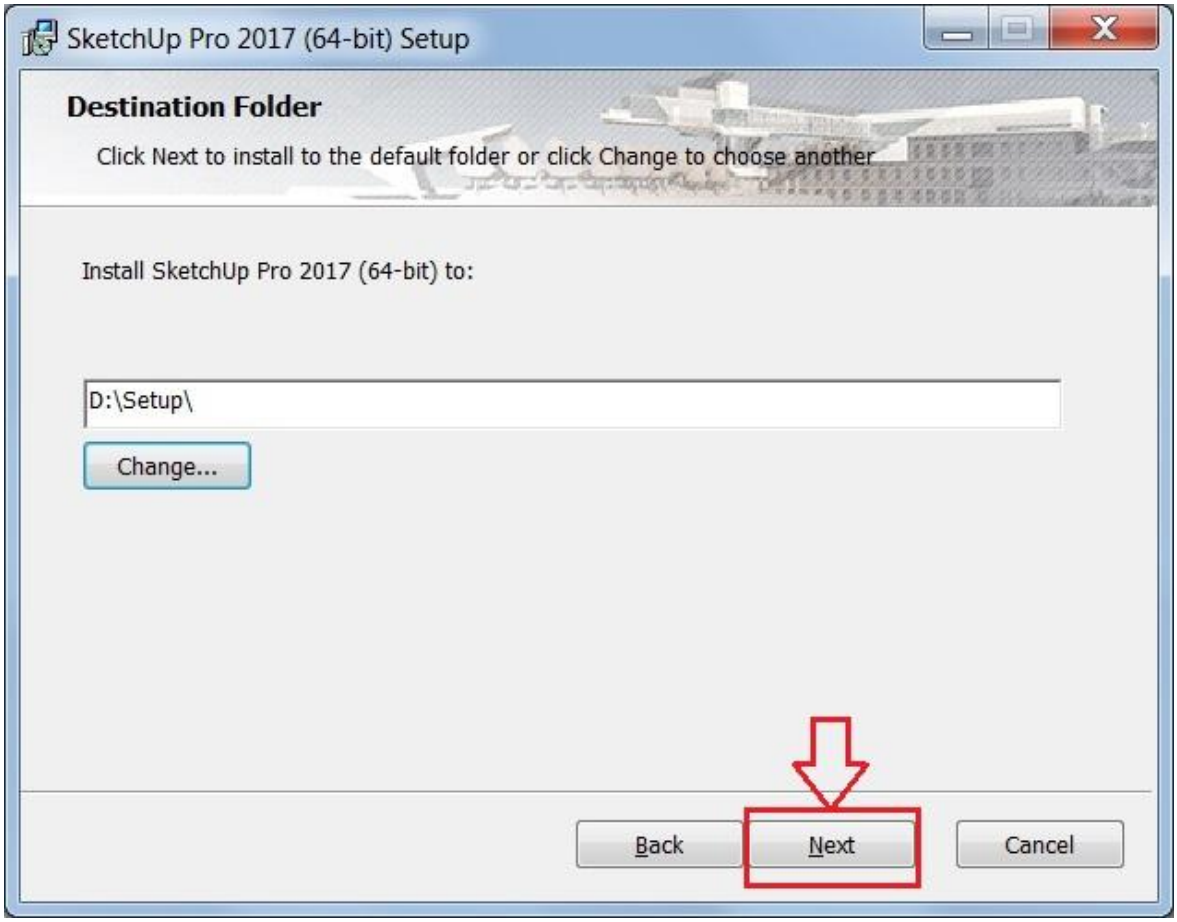


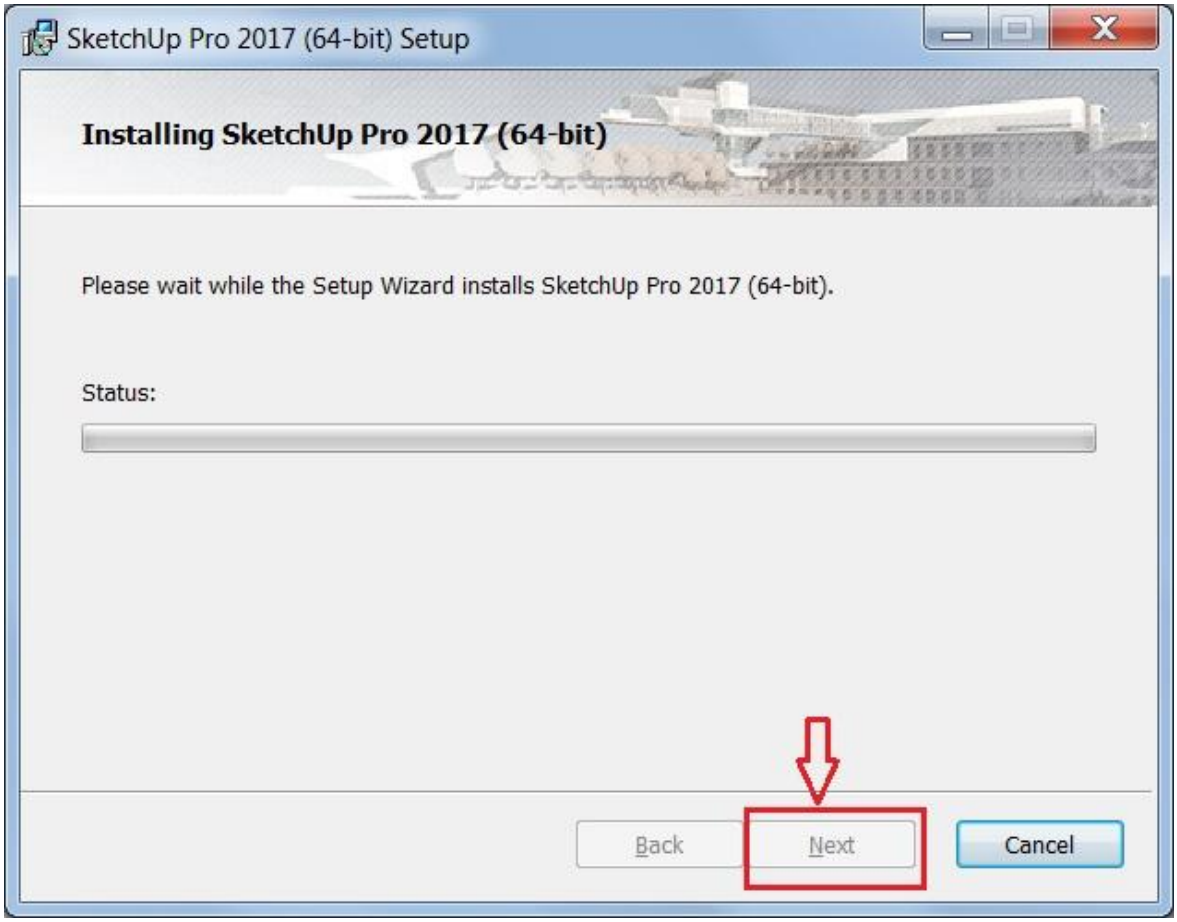
1.4.e, Cài đặt tool SketchUp

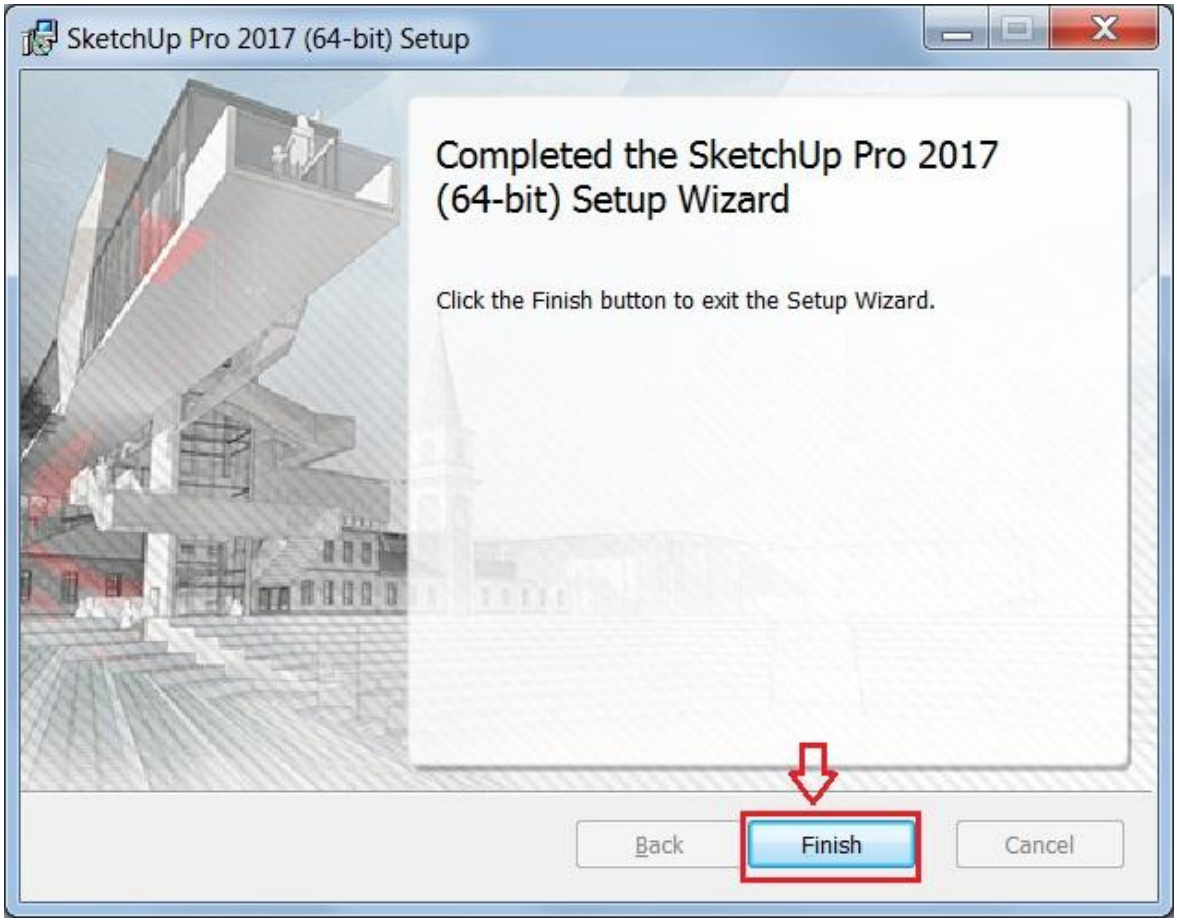
Bước 1: Đăng ký SketchUp, để dùng miễn phí bản dành cho sinh viên học tập.

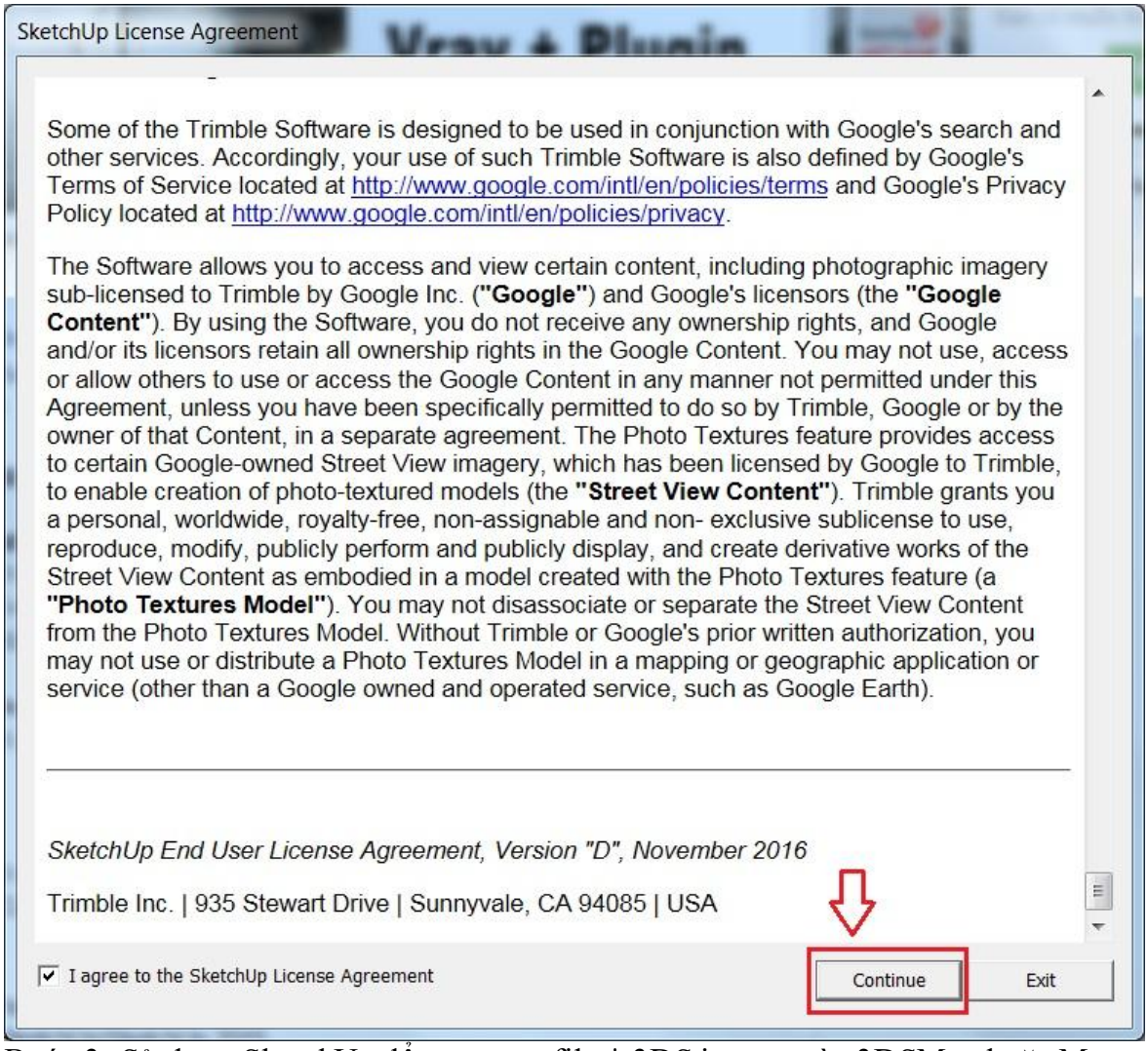



SketchUp Pro 2017 (64-bit) Setup
Welcome to the SketchUp Pro 2017 (64-bit) Setup Wizard
The Setup Wizard will install SketchUp Pro 2017 (64-bit) on your computer. Click Next to continue or Cancel to exit the Setup Wizard.
Back
Next
Cancel

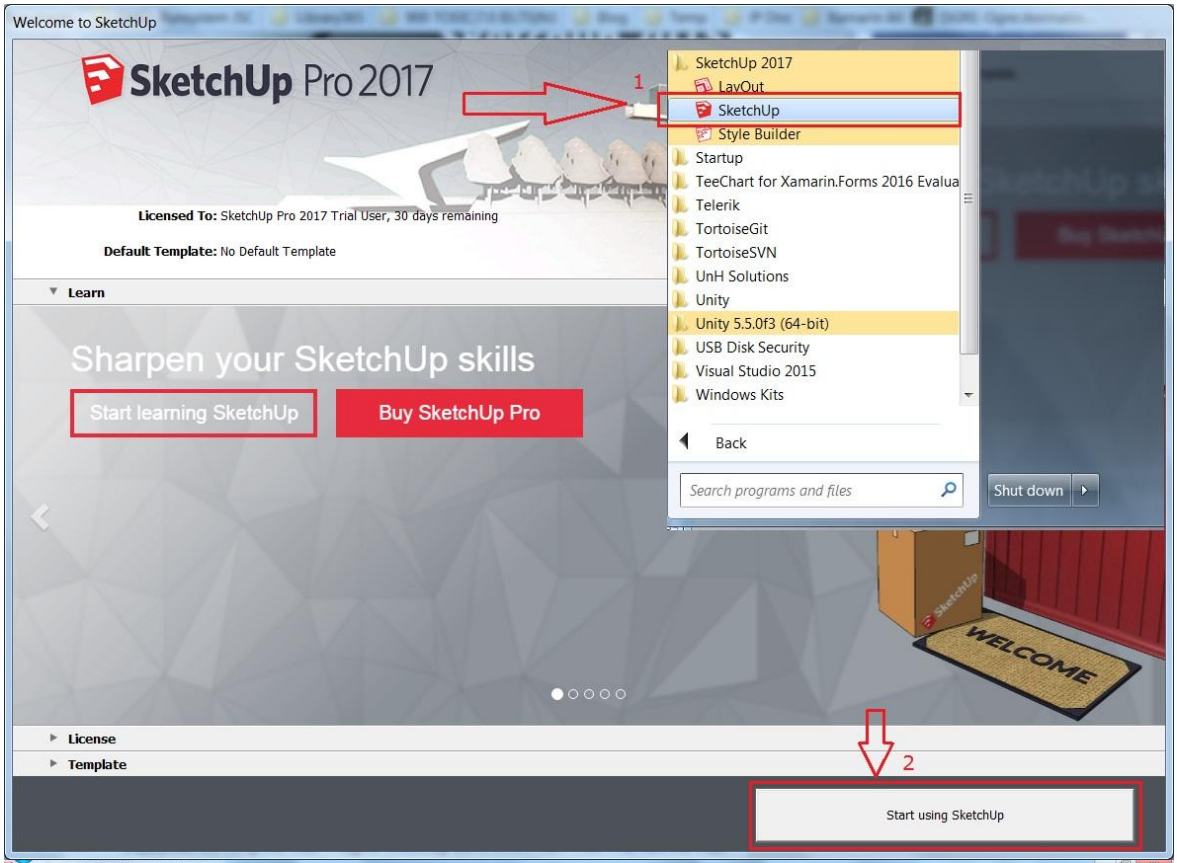

SketchUp Pro 2017 (64-bit) Setup
Destination Folder
Click Next to install to the default folder or click Change to choose another.
Install SketchUp Pro 2017 (64-bit) to:
D:\Setup\
Change...
Back
Next
Cancel

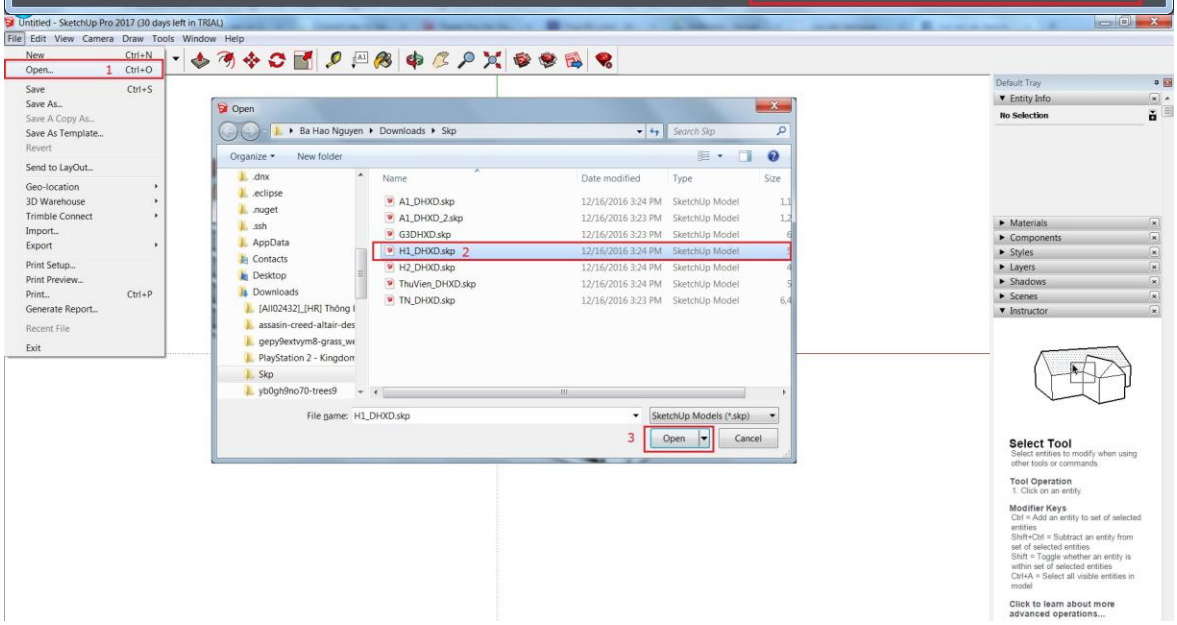

SketchUp Pro 2017 (64-bit) Setup
Installing SketchUp Pro 2017 (64-bit)
Please wait while the Setup Wizard installs SketchUp Pro 2017 (64-bit).
Status:
Back
Next
Cancel

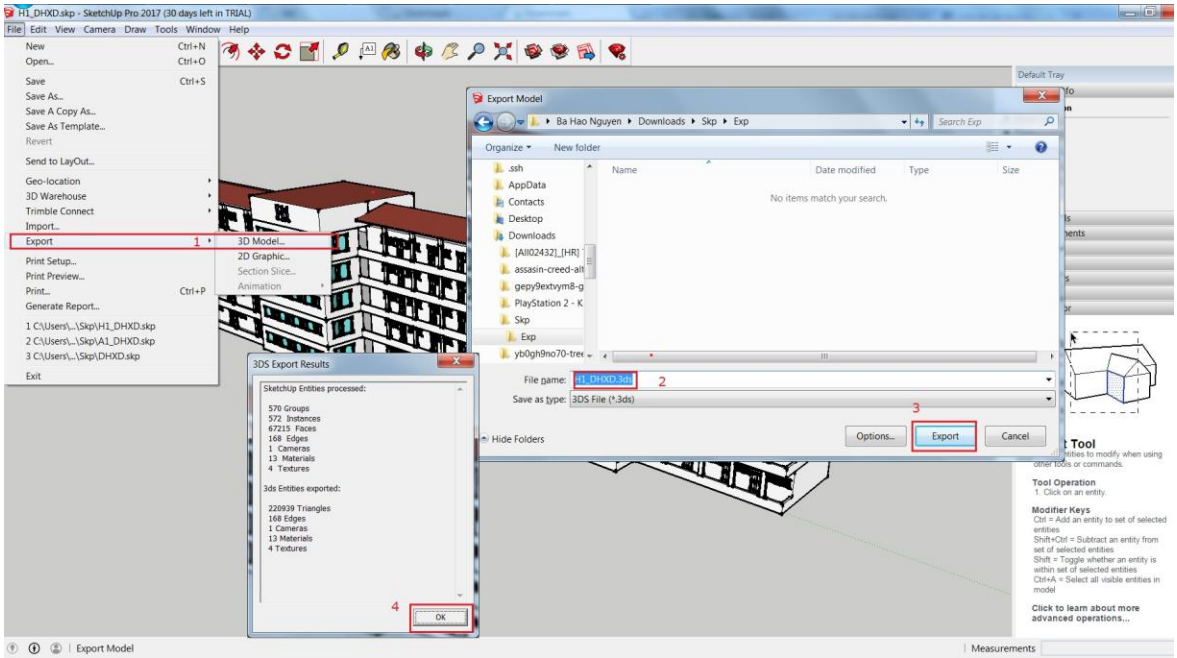

SketchUp Pro 2017 (64-bit) Setup
Completed the SketchUp Pro 2017 (64-bit) Setup Wizard
Click the Finish button to exit the Setup Wizard.
Back
Finish
Cancel



Bước 3: Sử dụng SketchUp để export ra file *.3DS import vào 3DSMax hoặc Maya để export ra file *.mesh. Ngoài ra, có thể cài đặt plugin trực tiếp cho SketchUp để export ra file *.mesh.

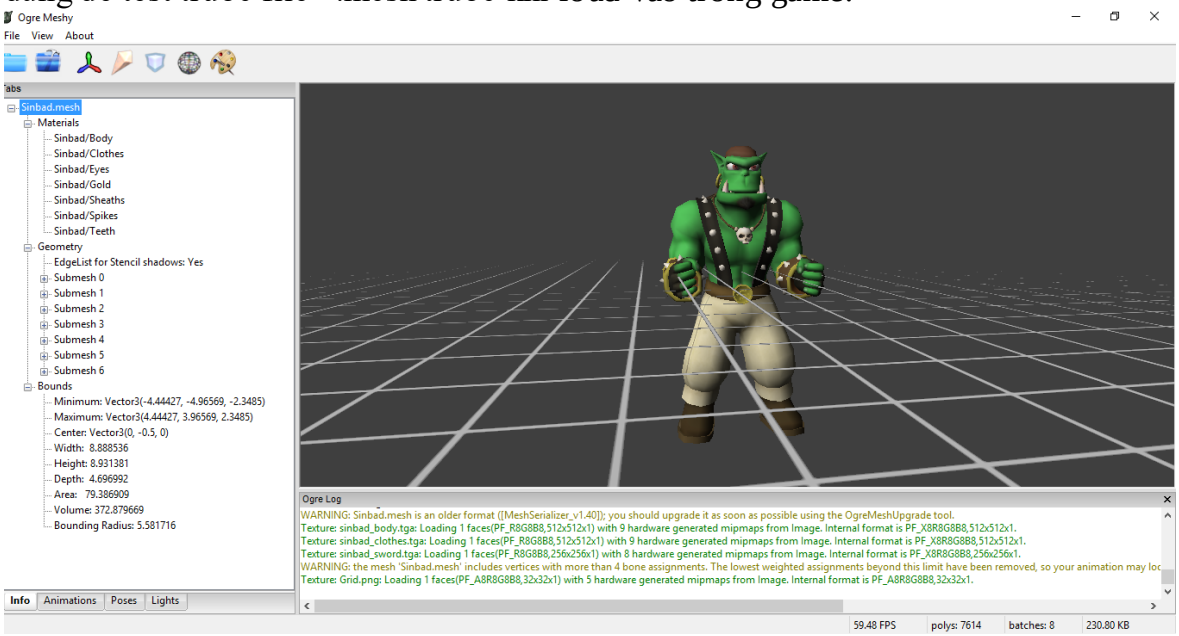

Welcome to SketchUp
SketchUp Pro 2017
Licensed To: SketchUp Pro 2017 Trial User, 30 days remaining
Default Template: No Default Template
Learn
Sharpen your SketchUp skills
Start learning SketchUp
Buy SketchUp Pro
License
Template
Start using SketchUp
1
2
SketchUp 2017
LayOut
SketchUp
Style Builder
Startup
TeeChart for Xamarin.Forms 2016 Evalua
Telerik
TortoiseGit
TortoiseSVN
UnH Solutions
Unity
Unity 5.5.0f3 (64-bit)
USB Disk Security
Visual Studio 2015
Windows Kits
Back
Search programs and files
Shut down


Untitled - SketchUp Pro 2017 (30 days left in TRIAL)
File Edit View Camera Draw Tools Window Help
Open... 1 Ctrl+O
Open
Ba Hao Nguyen ▸ Downloads ▸ Skp
A1_DHXD.skp
A1_DHXD_2.skp
G3DHXD.skp
H1_DHXD.skp 2
H2_DHXD.skp
ThuVien_DHXD.skp
TN_DHXD.skp
File name: H1_DHXD.skp
SketchUp Models (*.skp)
3 Open
Cancel
Entity Info
Materials
Components
Styles
Layers
Shadows
Scenes
Instructor
Select Tool



1.4.f, Cài đặt tool Ogre Meshy

Bước 1: Download Ogre Meshy tại đường dẫn https://sourceforge.net/projects/ogremeshy/files/latest/download. Phần mềm này dùng để test trước file *.mesh trước khi load vào trong game.



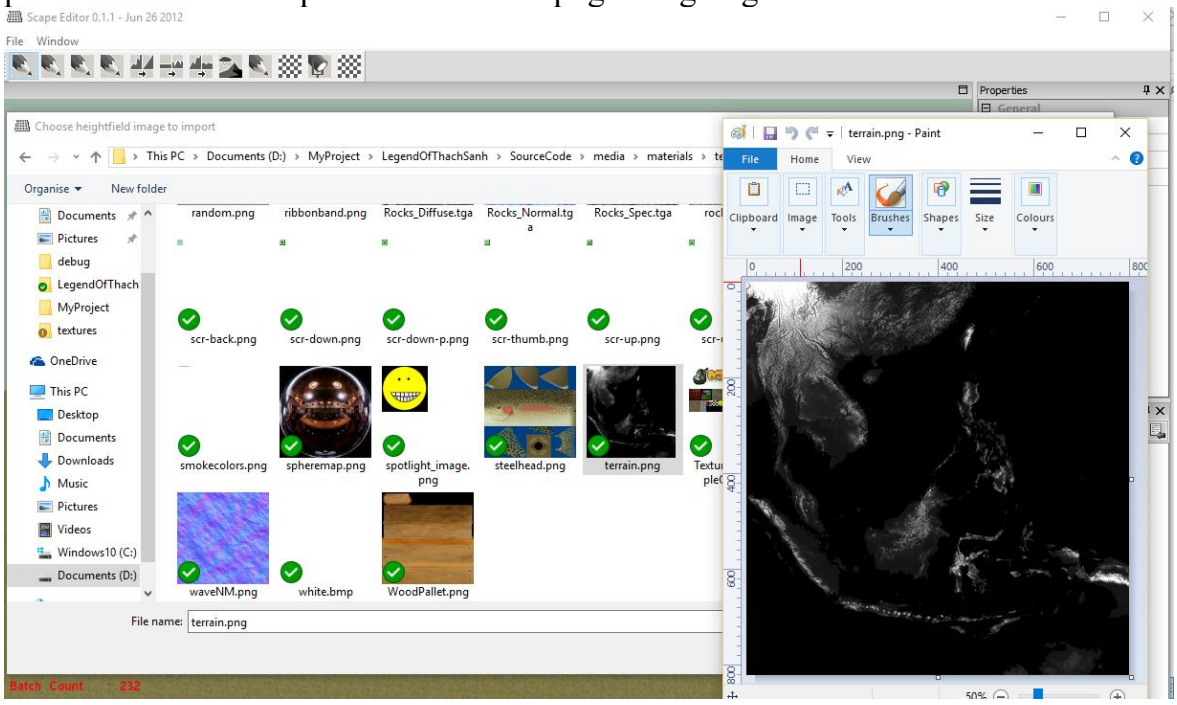
1.4.g, Cài đặt tool Scape

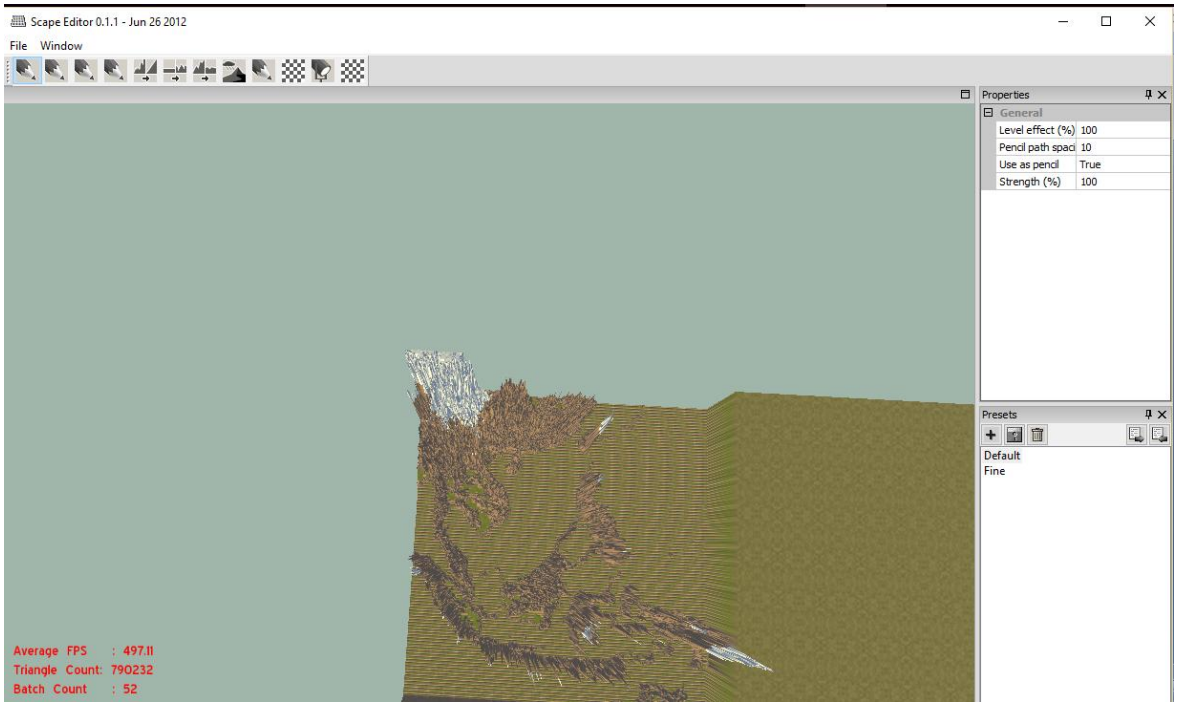
Bước 1: Download scape0.1.1-bin-win32.zip tại đường dẫn sau: https://goo.gl/ZLrI9M. Tải về giải nén và chạy với file Scape.exe. Phần mềm này có chức năng export file qui định độ cao của Terrain được tạo ra.

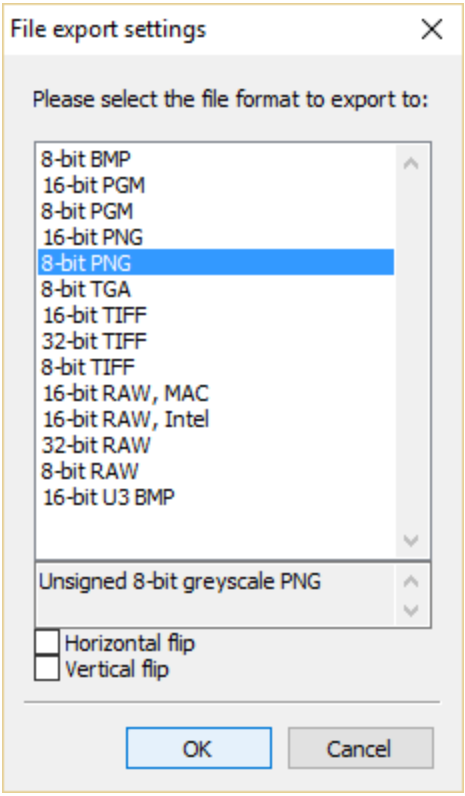
Materials	2012-06-29 6:38 PM	File folder	
Models	2016-12-11 2:11 PM	File folder	
Overlays	2016-12-11 2:11 PM	File folder	
Packs	2012-06-29 6:38 PM	File folder	
Settings	2016-12-11 2:11 PM	File folder	
cg.dll	2012-05-26 4:53 AM	Application extens...	8,331 KB
OgreMain.dll	2012-05-26 6:56 AM	Application extens...	7,420 KB
OIS.dll	2012-05-26 6:46 AM	Application extens...	107 KB
Plugin_CgProgramManager.dll	2012-05-26 6:58 AM	Application extens...	151 KB
Plugin_OctreeSceneManager.dll	2012-05-26 7:01 AM	Application extens...	154 KB
RenderSystem_Direct3D9.dll	2012-05-26 6:59 AM	Application extens...	578 KB
Resources.cfg	2008-03-05 3:16 PM	CFG File	1 KB
Scape.exe	2012-06-26 1:28 AM	Application	1,390 KB
Scape.log	016-12-11 2:37 PM	Text Document	20 KB
Settings.cf	008-03-05 3:16 PM	CFG File	1 KB
Settings_d.cfg	2008-03-05 3:16 PM	CFG File	1 KB
wxbase28_vc_custom.dll	2012-06-26 12:33 ...	Application extens...	1,024 KB
wxmsw28_aui_vc_custom.dll	2012-06-26 12:35 ...	Application extens...	241 KB
wxmsw28_core_vc_custom.dll	2012-06-26 12:34 ...	Application extens...	2,655 KB
wxmsw28_propgrid_vc_custom.dll	2012-06-26 12:51 ...	Application extens...	498 KB

Date created: 2016-12-06 10:47 PM
Size: 1.35 MB

Bước 2: Mở file terrain.png để edit. Có thể tăng/giảm độ cao, tăng giảm độ mịn và phủ texture sau đó export ra file terrain.png tương ứng.





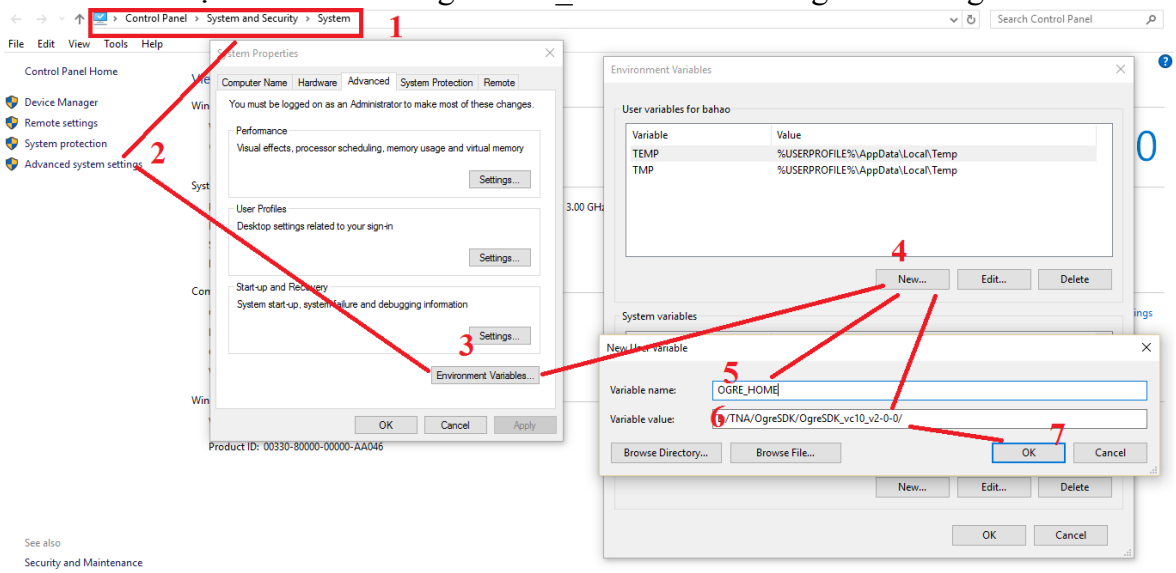


1.4.g, Cài đặt tool Visual Studio 2010

Bước 1: Download file cài đặt (Download tại đây: https://goo.gl/1pd2Ft) và chạy file setup với tùy chọn gói cài đặt mặc định.

Bước 2: Sau đó, cài đặt bản patch lên VS10sp1-KB983509 (Download tại đây: https://goo.gl/PPkVh8).

Bước 3: Cài đặt Ogre_VC10_AppWizard_1.8.0 giúp tạo nhanh ứng dụng Ogre3D (Link download: https://goo.gl/XU8Nw2). Tuy nhiên, từ OgreSDK 1.9 trở lên sẽ phải thêm thư viện Ogre Overlay để ứng dụng không báo lỗi thiếu thư viện.
Bước 4: Cài đặt biến môi trường OGRE_HOME với đường dẫn tới OgreSDK:



- Ngoài ra, có thể dụng Command Prompt window để set biến môi trường: Bấm Cửa sổ+R > cmd > enter. Tiếp đó, nhập lệnh: setx OGRE_HOME "< E:/TNA/OgreSDK/OgreSDK_vc10_v2-0-0/>".

2, Xây dựng kịch bản cho game

2.1, Tổng quan về game

2.1.a, Chủ đề game

2.1.a.1, Tóm tắt cốt truyện

Phần 1: Thạch Sanh vốn con trời đầu thai làm con một đôi vợ chồng già. Sớm mồ côi cha mẹ, Thạch Sanh sống một mình ở gốc đa với tài sản quý nhất là cây rìu và cái niêu cơm mãi không bao giờ cạn.

Phần 2: Vào một ngày nọ, Thạch Sanh bị Lý Thông giàn bẫy phải đi nộp mạng thay tới trông miếu Chằn Tinh. Kết cục, chàng không chết mà còn nhận được một cây cung vàng sau khi đốt xác Chằn Tinh.

Phần 3: Bị anh kết nghĩa Lý Thông lừa gạt cướp công, Thạch Sanh trở lại gốc đa sống và vô tình dùng cây cung vàng bắn trúng Đại bàng cắp công chúa để lần theo vết máu cứu công chúa Quỳnh Nga. Kết cục, chàng một lần nữa lập công giúp Lý Thông và bị nhốt dưới hang động đá.

Phần 4: Sau khi giải cứu con vua Thuỷ Tề cùng bị nhốt dưới hang, Thạch Sanh được tặng cây đàn thần. Và nhờ chính cây đàn, Thạch Sanh được công chúa Quỳnh Nga giải oan khi đang ngồi trong ngục vì bị hồn Chằn Tinh và Đại bàng trả thù.

Phần 5: Sau đó, việc Thạch Sanh được vua gả công chúa Quỳnh Nga cho khiến thái tử 18 chư hầu quay lại tấn công và Thạch Sanh một lần nữa dùng thành công tiếng đàn thần để đẩy lui. Chàng cũng đã hào phóng cho họ ăn cơm đựng trong niêu cơm thần.

2.1.a.2, Nội dụng game

Level 1- Thach Sanh Banian-tree:
+ Bối cảnh: Ngoài trời, cây đa...
+ Nội dung: Thể loại game run. Thạch Sanh chạy trong thời gian nhất định ăn các niêu cơm để tăng điểm và kích thước.
Level 2- Cut python:
+ Bối cảnh: Ngoài trời, Miếu thờ, Chằn tinh...
+ Nội dung: Đánh nhau với chằn tinh. Nếu thắng hiện thêm vật phẩm cung tên vàng.
Level 3- Shoot eagle:
+ Bối cảnh: Ngoài trời, cây đa, công chúa Quỳnh Nga...
+ Nội dung: Bắn Đại bàng. Nếu chiến thắng nhận được cây đàn thần.
Level 4- Tich tich tinh tang:
+ Bối cảnh: Trong nhà...
+ Nội dung: Đánh đàn. Gõ đàn theo đúng kịch bản để được gặp công chúa Quỳnh Nga.
Level 5- Rhythm of the justice:
+ Bối cảnh: Ngoài trời, cung thành, quân lính dàn hàng ngoài thành...
+ Nội dung: Đánh đàn. Gõ đàn theo đúng kịch bản và đưa niêu cơm cho quân địch lui quân. Cuối game cảnh trời tối và pháo hoa đêm giao thừa.

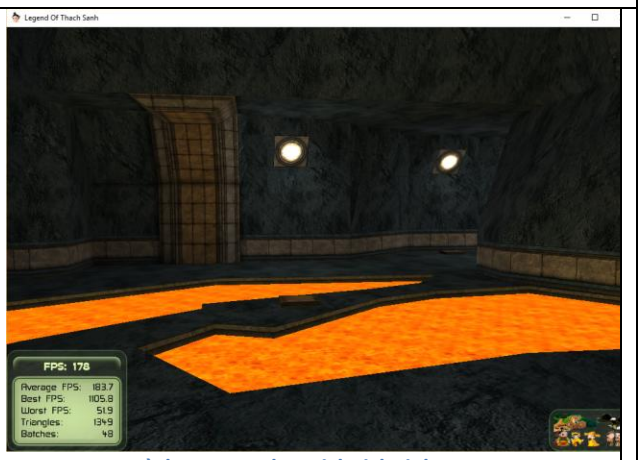
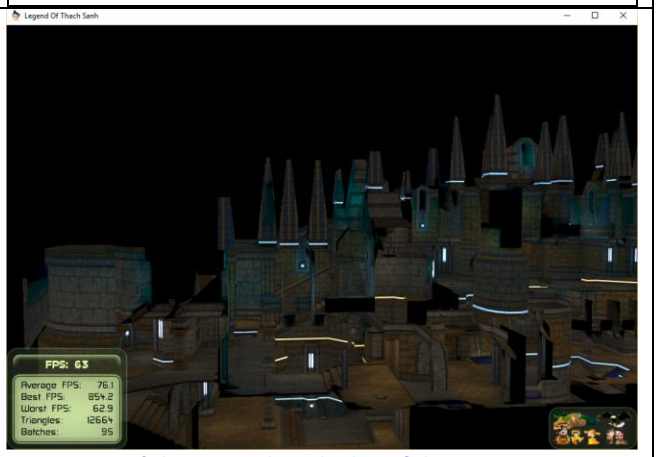
2.2, Kịch bản game

Tên	Mô tả
Bản đồ game	Game bao gồm 5 Level (version 1). Player phải tiêu diệt hết tất cả bot để chuyển map kế tiếp.
	Trong phần select map gồm map đã qua và map đang chơi và map đang bị lock.
Nhân vật	Nhân vật sẽ có các chỉ số: Max HP = 100- Máu; Max Speed = 17- Tốc độ; Max Jump = 30- Tăng tốc khi nhảy, Gravity = 90- Trọng lực, Max Height = 5- Độ cao.
Vũ khí	Vũ khí sẽ có các loại: Đao cơ bản; Rìu; Cung tên vàng; Đàn thần; Nồi cơm. Các chỉ số: Max Attack = 1- Tấn công (Tuỳ theo vũ khí); Max Visibility- Độ xa Max ảnh hưởng, Stability- Độ bền Max = 1.

Game Play	Khi player hết máu, nhân vật sẽ chết và state kết thúc.
	Người chơi phải giết hết bot nếu muốn qua state khi đó item chìa khoá sẽ bật ra, người chơi ăn chìa khoá để qua màn.
	Nhạc hiệu và hiệu ứng thay đổi qua màn khác.
Bàn phím và chuột	Sử dụng thư viện OIS để bắt sự kiện gõ phím và thao chuột trong.
Âm thanh	Backgroud map
	Nhân vật
	Vũ khí
	Hiệu ứng
Lưu dữ liệu	Lưu dữ liệu trên file xml
Hiệu ứng	Vũ khí
	Map
	Nhân vật
Camera	Góc nhìn thứ 3

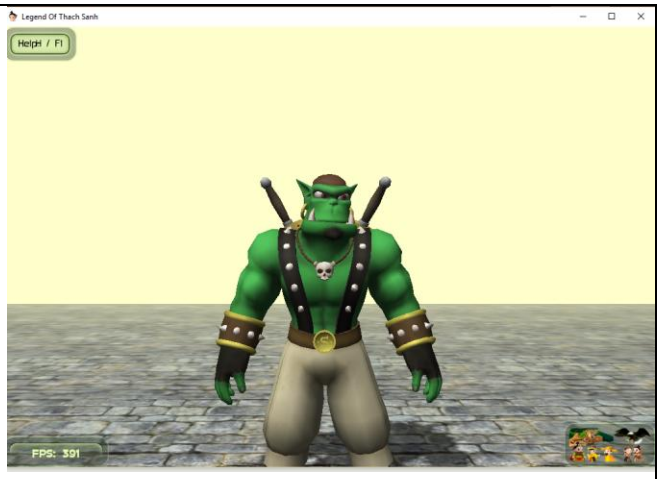
2.2.a, Game Map

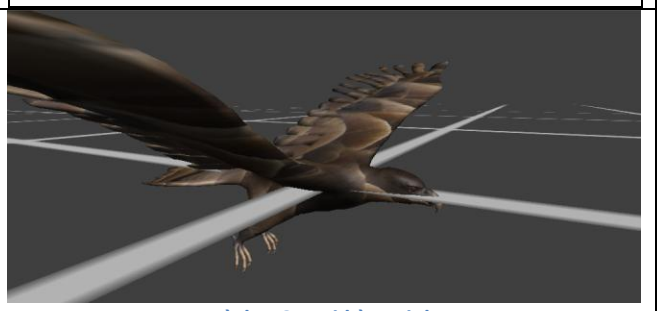
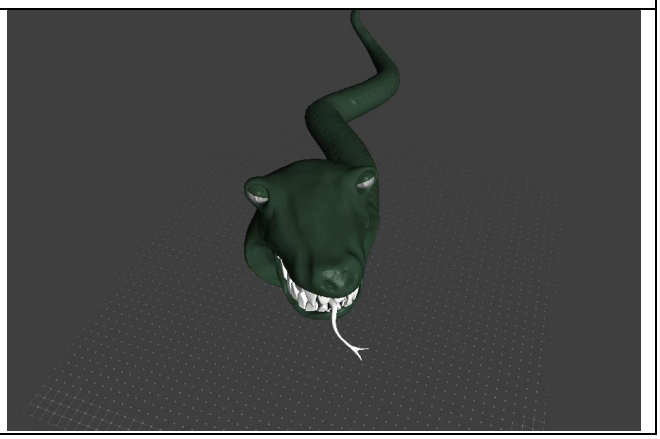
Tên	Mô tả
Level 1- Thach Sanh Banian-tree: + Bối cảnh: Ngoài trời, cây đa...	 Hình 8: Level 1- Thach Sanh Banian-tree
Level 2- Cut python: + Bối cảnh: Ngoài trời, Miếu thờ, Chằn tinh...	 Hình 9: Level 2- Cut python

Level 3- Shoot eagle: + Bối cảnh: Ngoài trời, cây đa, công chúa Quỳnh Nga...	 Hình 10: Level 3- Shoot eagle
Level 4- Tich tich tinh tang: + Bối cảnh: Trong nhà...	 Hình 11: Level 4- Tich tich tinh tang
Level 5- Rhythm of the justice: + Bối cảnh: Ngoài trời, cung thành, quân lính dàn hàng ngoài thành...	 Hình 12: Level 5- Rhythm of the justice

2.2.b, Nhân vật

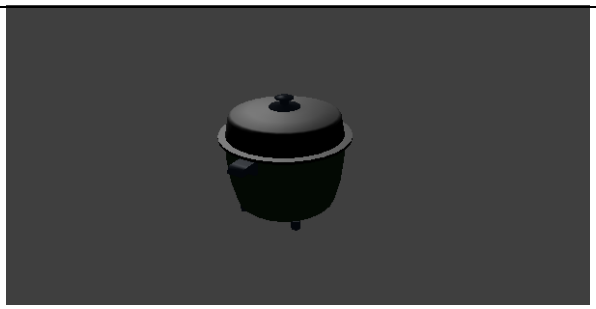
Tên nhân vật	Mô tả

Thạch Sanh	 Hình 13: Thạch Sanh
Công chúa Quỳnh Nga	 Hình 14: Công chúa Quỳnh Nga
Bot nhỏ 1(Trang phục màu đỏ)	 Hình 15: Bot nhỏ 1

Bot nhỏ 2 (Robot)	 Hình 16: Bot nhỏ 2
Bot nhỏ 3 (Ninja)	 Hình 17: Bot nhỏ 3
Đại bàng tinh	 Hình 18: Đại bàng tinh
Chằn tinh	

	Hình 19: Chẳn tinh

2.2.c, Vũ khí

Tên vũ khí	Mô tả
Song đao	 Hình 20: Song đao
Nồi cơm	 Hình 21: Nồi cơm
Cung tên vàng	 Hình 22: Cung tên vàng

Đàn thần	 Hình 23: Đàn thần
Rìu đốn củi	 Hình 24: Rìu đốn củi

2.2.d, Âm thanh

Tên âm thanh	Mô tả
Nhạc nền	XinChaoVietNam_Jmi.flac: Hòa tấu Hello Viet Nam
Tiếng nhân vật chết	death1.wav
Tiếng nhân vật bị tấn công	damage1.wav
Tiếng bước chân	footsteps-running.wav
Tiếng vung đao và kiếm	knife_hit1.wav
Tiếng tên bắn	elite-1.wav
Pháo hoa	fireworks.mp3

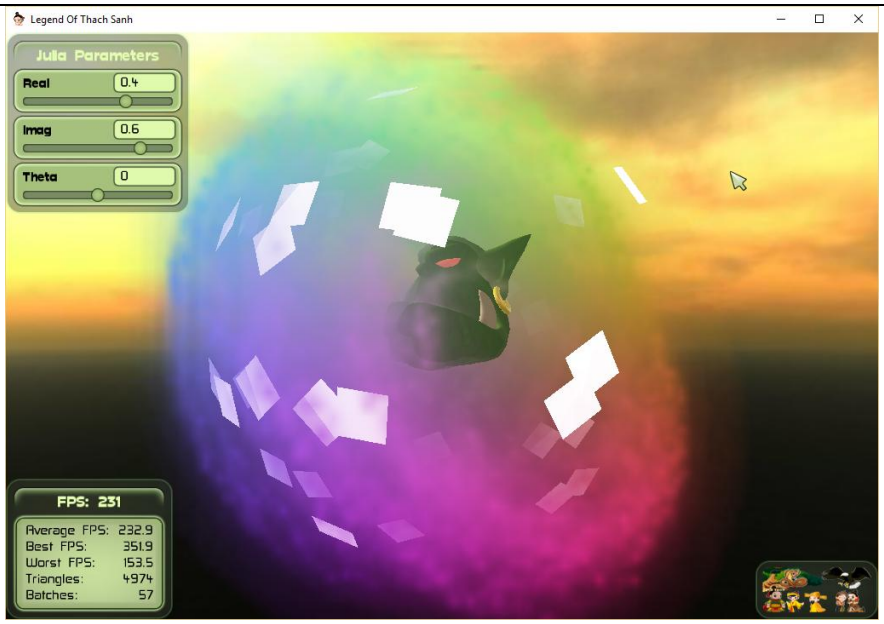
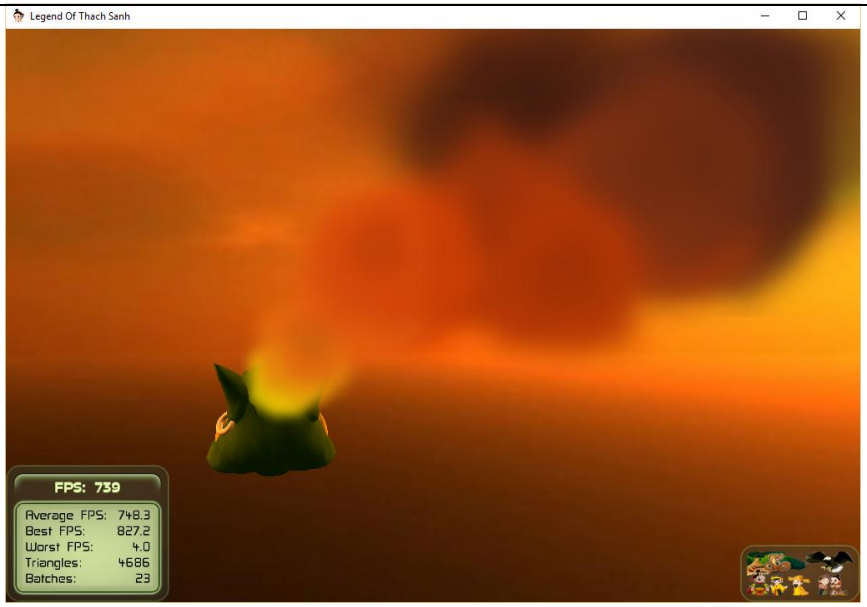
2.2.e, Thiết bị nhận điều khiển

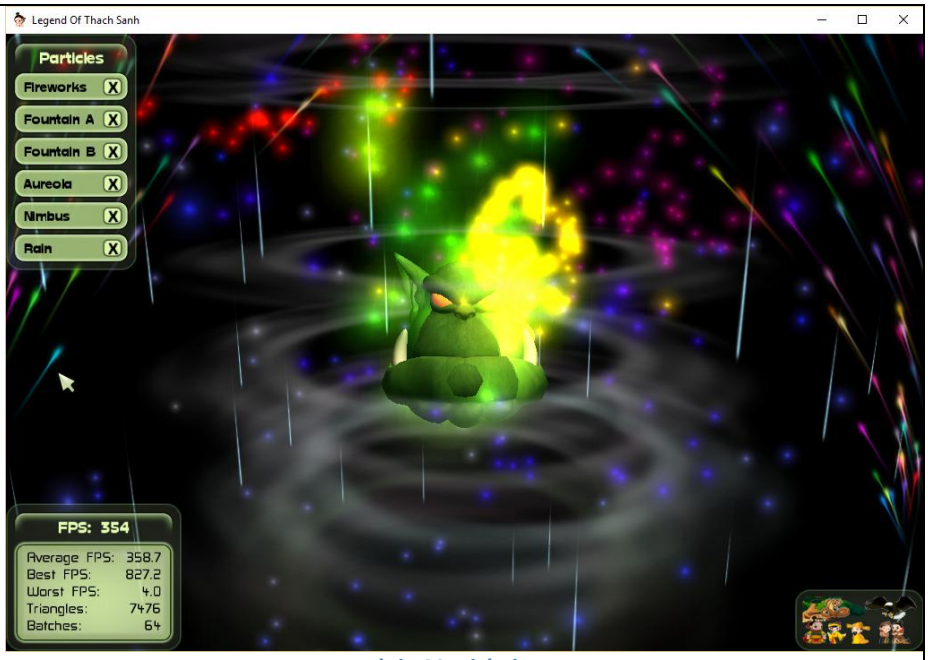
Tên thiết bị	Mô tả

Bàn phím và chuột	 Hình 25: Bàn phím và chuột

2.2.f, Hiệu ứng

Tên hiệu ứng	Mô tả
Hào quang của vũ khí	 Hình 26: Hào quang của vũ khí

Hào quang nhân vật	 Hình 27: Hào quang nhân vật
Khói lửa	 Hình 28: Khói lửa

Pháo hoa	 Hình 29: Pháo hoa

3, Xây dựng game theo kịch bản

3.1, Viết Class load âm thanh (MyFileFactoryCommon.h)

```
#if defined(WIN32)
#include <conio.h>
#else
#include "../common/conio.h"
#endif

#include <string.h>
#include <stdio.h>
#include "irrKlang.h"
using namespace irrklang;

#pragma comment(lib, "irrKlang.lib") // link with irrKlang.dll

// To start, we need to implement the class IFileFactory, which irrKlang uses
// to open files. The interface consists only of one single method named
// createFileReader(const ik_c8* filename). In this method, we create return
// our own file access class and return it:

// a class implementing the IFileFactory interface to override irrklang file access
class CMyFileFactory : public IFileFactory
{
public:
```

```
        //! Opens a file for read access. Simply return 0 if file not found.
        virtual IFileReader* createFileReader(const ik_c8* filename)
        {
                printf("MyFileFactory: open file %s\n", filename);

                FILE* file = fopen(filename, "rb");
                if (!file)
                        return 0;

                return new CMyReadFile(file, filename);
        }

protected:

        // To write our own file access methods returned in the method above,
        // we only need to implement the IFileReader interface, which has some
        // standard methods like read(), seek(), getPos() etc. In this example
        // we simply use fopen, fread, fseek etc and print to the console
        // when we are reading or seeking:

        // an own implementation of IReadFile to overwrite read access to files
        class CMyReadFile : public IFileReader
        {
        public:

                // constructor, store size of file and filename
                CMyReadFile(FILE* openedFile, const ik_c8* filename)
                {
                        File = openedFile;
                        strcpy(Filename, filename);

                        // get file size
                        fseek(File, 0, SEEK_END);
                        FileSize = ftell(File);
                        fseek(File, 0, SEEK_SET);
                }

                ~CMyReadFile()
                {
                        fclose(File);
                }

                //! reads data, returns how much was read
                ik_s32 read(void* buffer, ik_u32 sizeToRead)
```

```
		{
			printf("CMyReadFile: read %d bytes\n", sizeToRead);
			return (ik_s32)fread(buffer, 1, sizeToRead, File);
		}

		//! changes position in file, returns true if successful
		bool seek(ik_s32 finalPos, bool relativeMovement)
		{
			printf("CMyReadFile: seek to position %d\n", finalPos);
			return fseek(File, finalPos, relativeMovement ? SEEK_CUR
: SEEK_SET) == 0;
		}

		//! returns size of file
		ik_s32 getSize()
		{
			return FileSize;
		}

		//! returns where in the file we are.
		ik_s32 getPos()
		{
			return ftell(File);
		}

		//! returns name of file
		const ik_c8* getFileName()
		{
			return Filename;
		}

		FILE* File;
		char Filename[1024];
		ik_s32 FileSize;

	}; // end class CMyReadFile

}; // end class CMyFileFactory
```

3.2, Tạo bảng Common (SdkTableCommon.h)

```
#pragma once
#ifndef __SdkTableCommon_H__
#define __SdkTableCommon_H__
```

```
#pragma region [Define Terrain]
#define TERRAIN_PAGE_MIN_X 0
#define TERRAIN_PAGE_MIN_Y 0
#define TERRAIN_PAGE_MAX_X 0
#define TERRAIN_PAGE_MAX_Y 0
#define TERRAIN_FILE_PREFIX String("terrain_level3")
#define TERRAIN_FILE_SUFFIX String("dat")
#define TERRAIN_WORLD_SIZE 12000.0f
#define TERRAIN_SIZE 513
#define MAX_GRASS 11
#define MAX_GROUP_GRASS 3

enum Mode
{
	MODE_NORMAL = 0,
	MODE_EDIT_HEIGHT = 1,
	MODE_EDIT_BLEND = 2,
	MODE_COUNT = 3
};
enum ShadowMode
{
	SHADOWS_NONE = 0,
	SHADOWS_COLOUR = 1,
	SHADOWS_DEPTH = 2,
	SHADOWS_COUNT = 3
};
enum GrassType
{
	GRASS_NONE = 0,
	GRASS_RED_FLOWER = 1,
	GRASS_YELLOW_FLOWER = 2,
	GRASS_ALOE_VERA = 3,
	GRASS_DIEP = 4,
	GRASS_DAO_FLOWER = 5,
	GRASS_ANH_TUC = 6,
	GRASS_MAI = 7,
	GRASS_GANG = 8,
	GRASS_LAU = 9,
	GRASS_LAU_2 = 10,
	GRASS_SAN_HO = 11
};
#pragma endregion [Define Terrain]
//////////////////////////////////////////////////////////////////////////
#pragma region [Define Character]
#define NUM_ANIMS 13		// number of animations the character has
```

```
#define CHAR_HEIGHT 5          // height of character's center of mass above
ground
#define CAM_HEIGHT 2           // height of camera above character's center of
mass
#define RUN_SPEED 17           // character running speed in units per second
#define TURN_SPEED 500.0f      // character turning in degrees per second
#define ANIM_FADE_SPEED 7.5f   // animation crossfade speed in % of full
weight per second
#define JUMP_ACCEL 30.0f       // character jump acceleration in upward units
per squared second
#define GRAVITY 90.0f          // gravity in downward units per squared second
#define HP 100.0f //HP of character

// all the animations our character has, and a null ID
// some of these affect separate body parts and will be blended together
enum AnimID
{
        ANIM_IDLE_BASE,
        ANIM_IDLE_TOP,
        ANIM_RUN_BASE,
        ANIM_RUN_TOP,
        ANIM_HANDS_CLOSED,
        ANIM_HANDS_RELAXED,
        ANIM_DRAW_SWORDS,
        ANIM_SLICE_VERTICAL,
        ANIM_SLICE_HORIZONTAL,
        ANIM_DANCE,
        ANIM_JUMP_START,
        ANIM_JUMP_LOOP,
        ANIM_JUMP_END,
        ANIM_NONE
};
#pragma endregion [Define Character]
//////////////////////////////////////////////////////////////////////////////
#endif
```

3.3, Viết Class MaterialControls cho địa hình dưới nước (MaterialControlsCommon.h)

```
/*
-----------------------------------------------------------------------------------------
  ___                _

 /___\__ _ _ __ ___    /_\  _ __  _ __
 //  // _` |'__/ _ \  //_\\| '_ \| '_ \
```

```
/ \_// (_| | | |  __/ /  _  \ |_) | |_) \
\____/ \__, |_|  \____| \_/ \_/ .__/| .____/
     |____/                |_|   |_|
     Ogre 1.9.x Application for VC10
     https://plus.google.com/+bahaonguyen/
-----------------------------------------------------------------------------
| Filename   : MaterialControls.h
-----------------------------------------------------------------------------
| Written in : C++
-----------------------------------------------------------------------------
| Compiler   : Microsoft Visual C++ 2010 Service Pack 1.
-----------------------------------------------------------------------------
| Engine     : Ogre3D; ...............................
-----------------------------------------------------------------------------
| Written by : Ba Hao Nguyen – Sports editors, IT.
-----------------------------------------------------------------------------
| Note       : ...............................
-----------------------------------------------------------------------------
| Create File: 21:32:47 15/10/2016 by bahao
-----------------------------------------------------------------------------
| Last Update: 21:32:47 15/10/2016 by bahao
-----------------------------------------------------------------------------
*/

#ifndef __MaterialControls_H__
#define __MaterialControls_H__

#include "OgreString.h"

enum ShaderValType
{
        GPU_VERTEX, GPU_FRAGMENT, MAT_SPECULAR,
MAT_DIFFUSE, MAT_AMBIENT, MAT_SHININESS, MAT_EMISSIVE
};

//-----------------------------------------------------------------------
struct ShaderControl
{
   Ogre::String Name;
        Ogre::String ParamName;
        ShaderValType ValType;
        float MinVal;
        float MaxVal;
        size_t ElementIndex;
```

```
        mutable size_t PhysicalIndex;

        float getRange(void) const { return MaxVal - MinVal; }
        float convertParamToScrollPosition(const float val) const { return val -
MinVal; }
        float convertScrollPositionToParam(const float val) const { return val +
MinVal; }
};

typedef Ogre::vector<ShaderControl>::type ShaderControlsContainer;
typedef ShaderControlsContainer::iterator ShaderControlIterator;
// used for materials that have user controls

//-----------------------------------------------------------------------
class MaterialControls
{
public:
   MaterialControls(const Ogre::String& displayName, const Ogre::String&
materialName)
      : mDisplayName(displayName)
      , mMaterialName(materialName)
   {
   };

   ~MaterialControls(void){}

   const Ogre::String& getDisplayName(void) const { return mDisplayName; }
   const Ogre::String& getMaterialName(void) const { return mMaterialName; }
   size_t getShaderControlCount(void) const { return
mShaderControlsContainer.size(); }
   const ShaderControl& getShaderControl(const size_t idx) const
   {
      assert( idx < mShaderControlsContainer.size() );
      return mShaderControlsContainer[idx];
   }
   /** add a new control by passing a string parameter

   @param
     params is a string using the following format:
       "<Control Name>, <Shader parameter name>, <Parameter Type>, <Min
Val>, <Max Val>, <Parameter Sub Index>"

       <Control Name> is the string displayed for the control name on screen
       <Shader parameter name> is the name of the variable in the shader
       <Parameter Type> can be GPU_VERTEX, GPU_FRAGMENT
```

```
        <Min Val> minimum value that parameter can be
        <Max Val> maximum value that parameter can be
        <Parameter Sub Index> index into the the float array of the parameter.  All
GPU parameters are assumed to be float[4].

    */
    void addControl(const Ogre::String& params);

protected:

    Ogre::String mDisplayName;
    Ogre::String mMaterialName;

    ShaderControlsContainer mShaderControlsContainer;
};

typedef Ogre::vector<MaterialControls>::type MaterialControlsContainer;
typedef MaterialControlsContainer::iterator MaterialControlsIterator;

//-----------------------------------------------------------------------------
/** loads material shader controls from a configuration file
    A .controls file is made up of the following:

    [<material display name>]
    material = <material name>
    control = <Control Name>, <Shader parameter name>, <Parameter Type>,
<Min Val>, <Max Val>, <Parameter Sub Index>

    <material display name> is what is displayed in the material combo box.
    <material name> is the name of the material in the material script.
    control is the shader control associated with the material. The order
    of the contol definitions in the .controls file determines their order
    when displayed in the controls window.

    you can have multiple .controls files or put them all in one.
*/
void loadMaterialControlsFile(MaterialControlsContainer& controlsContainer,
const Ogre::String& filename);
/** load all control files found in resource paths
*/
void loadAllMaterialControlFiles(MaterialControlsContainer&
controlsContainer);

#endif // __MaterialControls_H__
```

```
/*
-----------------------------------------------------------------------------
  ___                 _

 /___\__ _ _ __ ___    /_\  _ __  _ __
// // _` | '__/ _ \ //_\\| '_ \| '_ \

/ \_// (_| | | |  __/ / _  \ |_) | |_) \
\___/ \__, |_|  \___| \_/ \_/ .__/| .___/
     |___/                  |_|   |_|
     Ogre 1.9.x Application for VC10
     https://plus.google.com/+bahaonguyen/
-----------------------------------------------------------------------------
| Filename   : MaterialControls.cpp
-----------------------------------------------------------------------------
| Written in : C++
-----------------------------------------------------------------------------
| Compiler   : Microsoft Visual C++ 2010 Service Pack 1.
-----------------------------------------------------------------------------
| Engine     : Ogre3D; ...............................
-----------------------------------------------------------------------------
| Written by : Ba Hao Nguyen – Sports editors, IT.
-----------------------------------------------------------------------------
| Note       : ...............................
-----------------------------------------------------------------------------
| Create File: 21:32:34 15/10/2016 by bahao
-----------------------------------------------------------------------------
| Last Update: 21:32:34 07/01/2017 by bahao
-----------------------------------------------------------------------------
*/

#include "MaterialControls.h"
#include "OgreLogManager.h"
#include "OgreStringVector.h"
#include "OgreStringConverter.h"
#include "OgreConfigFile.h"
#include "OgreResourceGroupManager.h"
#include "OgreException.h"
#include "OgreMaterial.h"
#include "OgreTechnique.h"
#include "OgreMaterialManager.h"

/*****************************************************************
***************
        MaterialControls Methods
```

```
*****************************************************************
****************/
void MaterialControls::addControl(const Ogre::String& params)
{
   // params is a string containing using the following format:
   //  "<Control Name>, <Shader parameter name>, <Parameter Type>, <Min
Val>, <Max Val>, <Parameter Sub Index>"

   // break up long string into components
   Ogre::StringVector vecparams = Ogre::StringUtil::split(params, ",");

   // if there are not five elements then log error and move on
   if (vecparams.size() != 6)
   {
      Ogre::LogManager::getSingleton().logMessage(
         "Incorrect number of parameters passed in params string for
MaterialControls::addControl()", Ogre::LML_CRITICAL);

      return;
   }

   ShaderControl newControl;
   Ogre::StringUtil::trim(vecparams[0]);
   newControl.Name = vecparams[0];

   Ogre::StringUtil::trim(vecparams[1]);
   newControl.ParamName = vecparams[1];

   Ogre::StringUtil::trim(vecparams[2]);
   if (vecparams[2] == "GPU_VERTEX")
      newControl.ValType = GPU_VERTEX;
   else if (vecparams[2] == "GPU_FRAGMENT")
      newControl.ValType = GPU_FRAGMENT;

   newControl.MinVal = Ogre::StringConverter::parseReal(vecparams[3]);
   newControl.MaxVal = Ogre::StringConverter::parseReal(vecparams[4]);
   newControl.ElementIndex = Ogre::StringConverter::parseInt(vecparams[5]);

   mShaderControlsContainer.push_back(newControl);

}

void loadMaterialControlsFile(MaterialControlsContainer& controlsContainer,
const Ogre::String& filename)
{
```

```
    // Load material controls from config file
    Ogre::ConfigFile cf;

    try
    {

        cf.loadFromResourceSystem(filename,
Ogre::ResourceGroupManager::DEFAULT_RESOURCE_GROUP_NAME,
"\t;=", true);

        // Go through all sections & controls in the file
        Ogre::ConfigFile::SectionIterator seci = cf.getSectionIterator();

        Ogre::String secName, typeName, materialName, dataString;

        while (seci.hasMoreElements())
        {
            secName = seci.peekNextKey();
            Ogre::ConfigFile::SettingsMultiMap* settings = seci.getNext();
            if (!secName.empty() && settings)
            {
                materialName = cf.getSetting("material", secName);

                                Ogre::MaterialPtr curMat =
Ogre::MaterialManager::getSingleton().getByName(materialName);
                                curMat->load();
                                Ogre::Technique * curTec = curMat-
>getBestTechnique();
                                if (!curTec || !curTec->isSupported())
                                {
                                        continue;
                                }

                MaterialControls newMaaterialControls(secName, materialName);
                controlsContainer.push_back(newMaaterialControls);

                size_t idx = controlsContainer.size() - 1;

                Ogre::ConfigFile::SettingsMultiMap::iterator i;

                for (i = settings->begin(); i != settings->end(); ++i)
                {
                    typeName = i->first;
                    dataString = i->second;
                    if (typeName == "control")
```

```
                    controlsContainer[idx].addControl(dataString);
                }
            }
        }

            Ogre::LogManager::getSingleton().logMessage( "Material Controls
setup" );
    }
    catch (Ogre::Exception e)
    {
        // Guess the file didn't exist
    }
}


void loadAllMaterialControlFiles(MaterialControlsContainer&
controlsContainer)
{
    Ogre::StringVectorPtr fileStringVector =
Ogre::ResourceGroupManager::getSingleton().findResourceNames( "Popular",
"*.controls");
        Ogre::StringVector::iterator controlsFileNameIterator =
fileStringVector->begin();

    while ( controlsFileNameIterator != fileStringVector->end() )
        {
        loadMaterialControlsFile(controlsContainer, *controlsFileNameIterator);
        ++controlsFileNameIterator;
        }
}
```

3.4, Xây dựng Content cho game (Terrain.h)

```
/*
-----------------------------------------------------------------------------
  ___                 _

 /___\__ _ _ __ ___    /_\  _ __  _ __
 //  // _` | '__/ _ \  //_\\| '_ \| '_ \

/ \_// (_| | | |  __/ /  _  \ |_) | |_) \
\____/ \__, |_|  \___| \_/ \_/ .__/| .___/
     |___/              |_|   |_|
```

```
    Ogre 1.9.x Application for VC10
    https://plus.google.com/+bahaonguyen/
-----------------------------------------------------------------------------------------
| Filename   : SinbadCharacterController.h
-----------------------------------------------------------------------------------------
| Written in : C++
-----------------------------------------------------------------------------------------
| Compiler   : Microsoft Visual C++ 2010 Service Pack 1.
-----------------------------------------------------------------------------------------
| Engine     : Ogre3D; ...............................
-----------------------------------------------------------------------------------------
| Written by : Ba Hao Nguyen – Sports editors, IT.
-----------------------------------------------------------------------------------------
| Note       : ...............................
-----------------------------------------------------------------------------------------
| Create File: 21:33:23 15/10/2016 by bahao
-----------------------------------------------------------------------------------------
| Last Update: 21:33:23 15/10/2016 by bahao
-----------------------------------------------------------------------------------------
*/
#ifndef __Sinbad_H__
#define __Sinbad_H__

#include "Ogre.h"
#include "OIS.h"
#include "SdkGame.h"
#include "SdkTableCommon.h"

using namespace Ogre;

class SinbadCharacterController
{
private:

public:

        SinbadCharacterController(Camera* cam)
        {
                setupBody(cam->getSceneManager(), cam->getPosition());
                setupCamera(cam);
                setupAnimations();
        }

        Vector3 getGoalDirection()
        {
```

```
		return (mSaveGoalDirection);
	}

	Real getLeng()
	{
		return mMechLength;
	}
	void setCharacterPos(Vector3 characterPos)
	{
		mBodyNode->setPosition(characterPos);
	}

	Vector3 getCharacterPos()
	{
		Vector3 characterPos = mBodyNode->getPosition();
		return characterPos;
	}

	void setMinPos(bool MinPos)
	{
		mMinPos = MinPos;
	}

	int getAnime()
	{
		return mBaseAnimID;
	}

	bool checkJump()
	{
		return (mVerticalVelocity < 0);
	}

	void addTime(Real deltaTime)
	{
		updateBody(deltaTime);
		updateAnimations(deltaTime);
		updateCamera(deltaTime);
	}

	void injectKeyDown(const OIS::KeyEvent& evt)
	{
		if (evt.key == OIS::KC_Q && (mTopAnimID ==
ANIM_IDLE_TOP || mTopAnimID == ANIM_RUN_TOP))
		{
```

```
                        // take swords out (or put them back, since it's the same
animation but reversed)
                        setTopAnimation(ANIM_DRAW_SWORDS, true);
                        mTimer = 0;
                }
                else if (evt.key == OIS::KC_E && !mSwordsDrawn)
                {
                        if (mTopAnimID == ANIM_IDLE_TOP || mTopAnimID ==
ANIM_RUN_TOP)
                        {
                                // start dancing
                                setBaseAnimation(ANIM_DANCE, true);
                                setTopAnimation(ANIM_NONE);
                                // disable hand animation because the dance controls
hands
                                mAnims[ANIM_HANDS_RELAXED]-
>setEnabled(false);
                        }
                        else if (mBaseAnimID == ANIM_DANCE)
                        {
                                // stop dancing
                                setBaseAnimation(ANIM_IDLE_BASE);
                                setTopAnimation(ANIM_IDLE_TOP);
                                // re-enable hand animation
                                mAnims[ANIM_HANDS_RELAXED]-
>setEnabled(true);
                        }
                }

                // keep track of the player's intended direction
                else if (evt.key == OIS::KC_W) mKeyDirection.z = -1;
                else if (evt.key == OIS::KC_A) mKeyDirection.x = -1;
                else if (evt.key == OIS::KC_S) mKeyDirection.z = 1;
                else if (evt.key == OIS::KC_D) mKeyDirection.x = 1;

                else if (evt.key == OIS::KC_SPACE && (mTopAnimID ==
ANIM_IDLE_TOP || mTopAnimID == ANIM_RUN_TOP))
                {
                        // jump if on ground
                        setBaseAnimation(ANIM_JUMP_START, true);
                        setTopAnimation(ANIM_NONE);
                        mTimer = 0;
                }

                if (!mKeyDirection.isZeroLength() && mBaseAnimID ==
```

```
ANIM_IDLE_BASE)
		{
			// start running if not already moving and the player wants to
move
			setBaseAnimation(ANIM_RUN_BASE, true);
			if (mTopAnimID == ANIM_IDLE_TOP)
setTopAnimation(ANIM_RUN_TOP, true);
		}
	}

	void injectKeyUp(const OIS::KeyEvent& evt)
	{
		// keep track of the player's intended direction
		if (evt.key == OIS::KC_W && mKeyDirection.z == -1)
mKeyDirection.z = 0;
		else if (evt.key == OIS::KC_A && mKeyDirection.x == -1)
mKeyDirection.x = 0;
		else if (evt.key == OIS::KC_S && mKeyDirection.z == 1)
mKeyDirection.z = 0;
		else if (evt.key == OIS::KC_D && mKeyDirection.x == 1)
mKeyDirection.x = 0;

		if (mKeyDirection.isZeroLength() && mBaseAnimID ==
ANIM_RUN_BASE)
		{
			// stop running if already moving and the player doesn't want
to move
			setBaseAnimation(ANIM_IDLE_BASE);
			if (mTopAnimID == ANIM_RUN_TOP)
setTopAnimation(ANIM_IDLE_TOP);
		}
	}

#if (OGRE_PLATFORM == OGRE_PLATFORM_APPLE_IOS) ||
(OGRE_PLATFORM == OGRE_PLATFORM_ANDROID)
	void injectMouseMove(const OIS::MultiTouchEvent& evt)
	{
		// update camera goal based on mouse movement
		updateCameraGoal(-0.05f * evt.state.X.rel, -0.05f * evt.state.Y.rel,
-0.0005f * evt.state.Z.rel);
	}

	void injectMouseDown(const OIS::MultiTouchEvent& evt)
	{
		if (mSwordsDrawn && (mTopAnimID == ANIM_IDLE_TOP ||
```

```
mTopAnimID == ANIM_RUN_TOP))
		{
			// if swords are out, and character's not doing something
weird, then SLICE!
	setTopAnimation(ANIM_SLICE_VERTICAL, true);
			mTimer = 0;
		}
	}
#else
	void injectMouseMove(const OIS::MouseEvent& evt)
	{
		// update camera goal based on mouse movement
		updateCameraGoal(-0.05f * evt.state.X.rel, -0.05f * evt.state.Y.rel,
-0.0005f * evt.state.Z.rel);
	}

	void injectMouseDown(const OIS::MouseEvent& evt,
OIS::MouseButtonID id)
	{
		if (mSwordsDrawn && (mTopAnimID == ANIM_IDLE_TOP ||
mTopAnimID == ANIM_RUN_TOP))
		{
			// if swords are out, and character's not doing something
weird, then SLICE!
			if (id == OIS::MB_Left)
setTopAnimation(ANIM_SLICE_VERTICAL, true);
			else if (id == OIS::MB_Right)
setTopAnimation(ANIM_SLICE_HORIZONTAL, true);
			mTimer = 0;
		}
	}
#endif

private:

	void setupBody(SceneManager* sceneMgr, Vector3 camPost)
	{
		// create main model
		mBodyNode = sceneMgr->getRootSceneNode()-
>createChildSceneNode();
		mBodyEnt = sceneMgr->createEntity("SinbadBody",
"Sinbad.mesh");
		mBodyNode->attachObject(mBodyEnt);

		Ogre::AxisAlignedBox box = mBodyEnt->getBoundingBox();
```

```
		mMechLength = box.getSize().x;

		// create swords and attach to sheath
		LogManager::getSingleton().logMessage("Creating swords");
		mSword1 = sceneMgr->createEntity("SinbadSword1",
"Sword.mesh");
		mSword2 = sceneMgr->createEntity("SinbadSword2",
"Sword.mesh");
		mBodyEnt->attachObjectToBone("Sheath.L", mSword1);
		mBodyEnt->attachObjectToBone("Sheath.R", mSword2);

		LogManager::getSingleton().logMessage("Creating the chains");
		// create a couple of ribbon trails for the swords, just for fun
		NameValuePairList params;
		params["numberOfChains"] = "2";
		params["maxElements"] = "80";
		mSwordTrail = (RibbonTrail*)sceneMgr-
>createMovableObject("RibbonTrail", &params);
		mSwordTrail->setMaterialName("Examples/LightRibbonTrail");
		mSwordTrail->setTrailLength(20);
		mSwordTrail->setVisible(false);
		sceneMgr->getRootSceneNode()->attachObject(mSwordTrail);


		for (int i = 0; i < 2; i++)
		{
			mSwordTrail->setInitialColour(i, 1, 0.8, 0);
			mSwordTrail->setColourChange(i, 0.75, 1.25, 1.25, 1.25);
			mSwordTrail->setWidthChange(i, 1);
			mSwordTrail->setInitialWidth(i, 0.5);
		}

		mKeyDirection = Vector3::ZERO;
		mVerticalVelocity = 0;
	}

	void setupAnimations()
	{
		// this is very important due to the nature of the exported animations
		mBodyEnt->getSkeleton()-
>setBlendMode(ANIMBLEND_CUMULATIVE);

		String animNames[] =
		{"IdleBase", "IdleTop", "RunBase", "RunTop", "HandsClosed",
```

```
"HandsRelaxed", "DrawSwords",
		"SliceVertical", "SliceHorizontal", "Dance", "JumpStart",
"JumpLoop", "JumpEnd"};

		// populate our animation list
		for (int i = 0; i < NUM_ANIMS; i++)
		{
			mAnims[i] = mBodyEnt-
>getAnimationState(animNames[i]);
			mAnims[i]->setLoop(true);
			mFadingIn[i] = false;
			mFadingOut[i] = false;
		}

		// start off in the idle state (top and bottom together)
		setBaseAnimation(ANIM_IDLE_BASE);
		setTopAnimation(ANIM_IDLE_TOP);

		// relax the hands since we're not holding anything
		mAnims[ANIM_HANDS_RELAXED]->setEnabled(true);

		mSwordsDrawn = false;
	}

	void setupCamera(Camera* cam)
	{
		// create a pivot at roughly the character's shoulder
		mCameraPivot = cam->getSceneManager()->getRootSceneNode()-
>createChildSceneNode();
		// this is where the camera should be soon, and it spins around the
pivot
		mCameraGoal = mCameraPivot-
>createChildSceneNode(Vector3(0, 0, 15));
		// this is where the camera actually is
		mCameraNode = cam->getSceneManager()->getRootSceneNode()-
>createChildSceneNode();
		mCameraNode->setPosition(mCameraPivot->getPosition() +
mCameraGoal->getPosition());

		mCameraPivot->setFixedYawAxis(true);
		mCameraGoal->setFixedYawAxis(true);
		mCameraNode->setFixedYawAxis(true);

		// our model is quite small, so reduce the clipping planes
		//cam->setNearClipDistance(0.1);
```

```
		cam->setFarClipDistance(100);
		cam->setFarClipDistance(50000);
		mCameraNode->attachObject(cam);

		mPivotPitch = 0;
	}

	void updateBody(Real deltaTime)
	{
		if (mGoalDirection != Vector3::ZERO)
		{
			mSaveGoalDirection = mGoalDirection;
		}

		mGoalDirection = Vector3::ZERO;  // we will calculate this

		if (mKeyDirection != Vector3::ZERO && mBaseAnimID !=
ANIM_DANCE)
		{
			// calculate actually goal direction in world based on player's
key directions
			mGoalDirection += mKeyDirection.z * mCameraNode-
>getOrientation().zAxis();
			mGoalDirection += mKeyDirection.x * mCameraNode-
>getOrientation().xAxis();
			mGoalDirection.y = 0;
			mGoalDirection.normalise();

			Quaternion toGoal = mBodyNode-
>getOrientation().zAxis().getRotationTo(mGoalDirection);

			// calculate how much the character has to turn to face goal
direction
			Real yawToGoal = toGoal.getYaw().valueDegrees();
			// this is how much the character CAN turn this frame
			Real yawAtSpeed = yawToGoal / Math::Abs(yawToGoal)
* deltaTime * TURN_SPEED;
			// reduce "turnability" if we're in midair
			if (mBaseAnimID == ANIM_JUMP_LOOP) yawAtSpeed
*= 0.2f;

			// turn as much as we can, but not more than we need to
			if (yawToGoal < 0) yawToGoal = std::min<Real>(0,
std::max<Real>(yawToGoal, yawAtSpeed)); //yawToGoal =
Math::Clamp<Real>(yawToGoal, yawAtSpeed, 0);
```

```
                        else if (yawToGoal > 0) yawToGoal = std::max<Real>(0,
std::min<Real>(yawToGoal, yawAtSpeed)); //yawToGoal =
Math::Clamp<Real>(yawToGoal, 0, yawAtSpeed);

                        mBodyNode->yaw(Degree(yawToGoal));

                        // move in current body direction (not the goal direction)
                        mBodyNode->translate(0, 0, deltaTime * RUN_SPEED *
mAnims[mBaseAnimID]->getWeight(),
                                Node::TS_LOCAL);
                }

                if (mBaseAnimID == ANIM_JUMP_LOOP)
                {
                        // if we're jumping, add a vertical offset too, and apply
gravity
                        mBodyNode->translate(0, mVerticalVelocity * deltaTime,
0, Node::TS_LOCAL);
                        mVerticalVelocity -= GRAVITY * deltaTime;

                        Vector3 pos = mBodyNode->getPosition();
                        if (mMinPos)
                        {
                                // if we've hit the ground, change to landing state
                                //pos.y = CHAR_HEIGHT;
                                //mBodyNode->setPosition(pos);
                                setBaseAnimation(ANIM_JUMP_END, true);
                                mTimer = 0;
                                mMinPos = false;
                                mVerticalVelocity = 0;
                        }
                }
        }

        void updateAnimations(Real deltaTime)
        {
                Real baseAnimSpeed = 1;
                Real topAnimSpeed = 1;

                mTimer += deltaTime;

                if (mTopAnimID == ANIM_DRAW_SWORDS)
                {
                        // flip the draw swords animation if we need to put it back
                        topAnimSpeed = mSwordsDrawn ? -1 : 1;
```

```
                        // half-way through the animation is when the hand grasps
the handles...
                        if (mTimer >= mAnims[mTopAnimID]->getLength() / 2 &&
                                mTimer - deltaTime < mAnims[mTopAnimID]-
>getLength() / 2)
                        {
                                // so transfer the swords from the sheaths to the hands
                                mBodyEnt->detachAllObjectsFromBone();
                                mBodyEnt->attachObjectToBone(mSwordsDrawn ?
"Sheath.L" : "Handle.L", mSword1);
                                mBodyEnt->attachObjectToBone(mSwordsDrawn ?
"Sheath.R" : "Handle.R", mSword2);
                                // change the hand state to grab or let go
                                mAnims[ANIM_HANDS_CLOSED]-
>setEnabled(!mSwordsDrawn);
                                mAnims[ANIM_HANDS_RELAXED]-
>setEnabled(mSwordsDrawn);

                                // toggle sword trails
                                if (mSwordsDrawn)
                                {
                                        mSwordTrail->setVisible(false);
                                        mSwordTrail->removeNode(mSword1-
>getParentNode());
                                        mSwordTrail->removeNode(mSword2-
>getParentNode());
                                }
                                else
                                {
                                        mSwordTrail->setVisible(true);
                                        mSwordTrail->addNode(mSword1-
>getParentNode());
                                        mSwordTrail->addNode(mSword2-
>getParentNode());
                                }
                        }

                        if (mTimer >= mAnims[mTopAnimID]->getLength())
                        {
                                // animation is finished, so return to what we were
doing before
                                if (mBaseAnimID == ANIM_IDLE_BASE)
setTopAnimation(ANIM_IDLE_TOP);
                                else
```

```
                        {
                                setTopAnimation(ANIM_RUN_TOP);
                                mAnims[ANIM_RUN_TOP]-
>setTimePosition(mAnims[ANIM_RUN_BASE]->getTimePosition());
                        }
                        mSwordsDrawn = !mSwordsDrawn;
                }
        }
        else if (mTopAnimID == ANIM_SLICE_VERTICAL ||
mTopAnimID == ANIM_SLICE_HORIZONTAL)
        {
                if (mTimer >= mAnims[mTopAnimID]->getLength())
                {
                        // animation is finished, so return to what we were
doing before
                        if (mBaseAnimID == ANIM_IDLE_BASE)
setTopAnimation(ANIM_IDLE_TOP);
                        else
                        {
                                setTopAnimation(ANIM_RUN_TOP);
                                mAnims[ANIM_RUN_TOP]-
>setTimePosition(mAnims[ANIM_RUN_BASE]->getTimePosition());
                        }
                }

                // don't sway hips from side to side when slicing. that's just
embarrassing.
                if (mBaseAnimID == ANIM_IDLE_BASE)
baseAnimSpeed = 0;
        }
        else if (mBaseAnimID == ANIM_JUMP_START)
        {
                if (mTimer >= mAnims[mBaseAnimID]->getLength())
                {
                        // takeoff animation finished, so time to leave the
ground!
                        setBaseAnimation(ANIM_JUMP_LOOP, true);
                        // apply a jump acceleration to the character
                        mVerticalVelocity = JUMP_ACCEL;
                }
        }
        else if (mBaseAnimID == ANIM_JUMP_END)
        {
                if (mTimer >= mAnims[mBaseAnimID]->getLength())
                {
```

```
                        // safely landed, so go back to running or idling
                        if (mKeyDirection == Vector3::ZERO)
                        {
                                setBaseAnimation(ANIM_IDLE_BASE);
                                setTopAnimation(ANIM_IDLE_TOP);
                        }
                        else
                        {
                                setBaseAnimation(ANIM_RUN_BASE, true);
                                setTopAnimation(ANIM_RUN_TOP, true);
                        }
                }
        }

        // increment the current base and top animation times
        if (mBaseAnimID != ANIM_NONE) mAnims[mBaseAnimID]-
>addTime(deltaTime * baseAnimSpeed);
        if (mTopAnimID != ANIM_NONE) mAnims[mTopAnimID]-
>addTime(deltaTime * topAnimSpeed);

        // apply smooth transitioning between our animations
        fadeAnimations(deltaTime);
    }

    void fadeAnimations(Real deltaTime)
    {
        for (int i = 0; i < NUM_ANIMS; i++)
        {
                if (mFadingIn[i])
                {
                        // slowly fade this animation in until it has full weight
                        Real newWeight = mAnims[i]->getWeight() +
deltaTime * ANIM_FADE_SPEED;
                        mAnims[i]-
>setWeight(Math::Clamp<Real>(newWeight, 0, 1));
                        if (newWeight >= 1) mFadingIn[i] = false;
                }
                else if (mFadingOut[i])
                {
                        // slowly fade this animation out until it has no weight,
and then disable it
                        Real newWeight = mAnims[i]->getWeight() -
deltaTime * ANIM_FADE_SPEED;
                        mAnims[i]-
>setWeight(Math::Clamp<Real>(newWeight, 0, 1));
```

```
                        if (newWeight <= 0)
                        {
                                mAnims[i]->setEnabled(false);
                                mFadingOut[i] = false;
                        }
                }
        }
}

void updateCamera(Real deltaTime)
{
        // place the camera pivot roughly at the character's shoulder
        mCameraPivot->setPosition(mBodyNode->getPosition() +
Vector3::UNIT_Y * CAM_HEIGHT);
        // move the camera smoothly to the goal
        Vector3 goalOffset = mCameraGoal->_getDerivedPosition() -
mCameraNode->getPosition();
        mCameraNode->translate(goalOffset * deltaTime * 9.0f);
        // always look at the pivot
        mCameraNode->lookAt(mCameraPivot->_getDerivedPosition(),
Node::TS_WORLD);
}

void updateCameraGoal(Real deltaYaw, Real deltaPitch, Real
deltaZoom)
{
        mCameraPivot->yaw(Degree(deltaYaw), Node::TS_WORLD);

        // bound the pitch
        if (!(mPivotPitch + deltaPitch > 25 && deltaPitch > 0) &&
                !(mPivotPitch + deltaPitch < -60 && deltaPitch < 0))
        {
                mCameraPivot->pitch(Degree(deltaPitch),
Node::TS_LOCAL);
                mPivotPitch += deltaPitch;
        }

        Real dist = mCameraGoal-
>_getDerivedPosition().distance(mCameraPivot->_getDerivedPosition());
        Real distChange = deltaZoom * dist;

        // bound the zoom
        if (!(dist + distChange < 8 && distChange < 0) &&
                !(dist + distChange > 25 && distChange > 0))
        {
```

```
                        mCameraGoal->translate(0, 0, distChange,
Node::TS_LOCAL);
                }
        }

        void setBaseAnimation(AnimID id, bool reset = false)
        {
                if (mBaseAnimID >= 0 && mBaseAnimID < NUM_ANIMS)
                {
                        // if we have an old animation, fade it out
                        mFadingIn[mBaseAnimID] = false;
                        mFadingOut[mBaseAnimID] = true;
                }

                mBaseAnimID = id;

                if (id != ANIM_NONE)
                {
                        // if we have a new animation, enable it and fade it in
                        mAnims[id]->setEnabled(true);
                        mAnims[id]->setWeight(0);
                        mFadingOut[id] = false;
                        mFadingIn[id] = true;
                        if (reset) mAnims[id]->setTimePosition(0);
                }
        }

        void setTopAnimation(AnimID id, bool reset = false)
        {
                if (mTopAnimID >= 0 && mTopAnimID < NUM_ANIMS)
                {
                        // if we have an old animation, fade it out
                        mFadingIn[mTopAnimID] = false;
                        mFadingOut[mTopAnimID] = true;
                }

                mTopAnimID = id;

                if (id != ANIM_NONE)
                {
                        // if we have a new animation, enable it and fade it in
                        mAnims[id]->setEnabled(true);
                        mAnims[id]->setWeight(0);
                        mFadingOut[id] = false;
                        mFadingIn[id] = true;
```

```
                        if (reset) mAnims[id]->setTimePosition(0);
                }
        }

        SceneNode* mBodyNode;
        SceneNode* mCameraPivot;
        SceneNode* mCameraGoal;
        SceneNode* mCameraNode;
        Real mPivotPitch;
        Entity* mBodyEnt;
        Entity* mSword1;
        Entity* mSword2;
        RibbonTrail* mSwordTrail;
        AnimationState* mAnims[NUM_ANIMS];   // master animation list
        AnimID mBaseAnimID;                  // current base (full- or lower-body)
animation
        AnimID mTopAnimID;                   // current top (upper-body) animation
        bool mFadingIn[NUM_ANIMS];           // which animations are fading in
        bool mFadingOut[NUM_ANIMS];          // which animations are fading
out
        bool mSwordsDrawn;
        Vector3 mKeyDirection;     // player's local intended direction based on
WASD keys
        Vector3 mGoalDirection;    // actual intended direction in world-space
        Vector3 mSaveGoalDirection;    // save actual intended direction in world-
space
        Real mVerticalVelocity;    // for jumping
        Real mTimer;               // general timer to see how long animations have
been playing
        bool mMinPos; //Min height Terrain at pos

        Ogre::Real mMechLength;
};

#endif
```

3.5, Viết Class SinbadCharacterController tạo nhân vật (SinbadCharacterControllerCommon.h)

```
/*
-----------------------------------------------------------------------------
  ___                 _
```

```
  /____\__ _ _ __ ____     /_\  _ __  _ __
 //  // _` | '__/ _ \  //_\\| '_ \| '_ \

/ \_// (_| | | |  __/ /  _  \ |_) | |_) \
\____/ \__, |_|  \____| \_/ \_/ .__/| .____/
      |____/                 |_|   |_|
      Ogre 1.9.x Application for VC10
      https://plus.google.com/+bahaonguyen/
-----------------------------------------------------------------------------------------
| Filename   : Terrain.h
-----------------------------------------------------------------------------------------
| Written in : C++
-----------------------------------------------------------------------------------------
| Compiler   : Microsoft Visual C++ 2010 Service Pack 1.
-----------------------------------------------------------------------------------------
| Engine     : Ogre3D; ...............................
-----------------------------------------------------------------------------------------
| Written by : Ba Hao Nguyen ESports editors, IT.
-----------------------------------------------------------------------------------------
| Note       : Chức năng xây dựng Content: Terrain, Ocean, Tree, Character,...bắt
sự kiện trong game như ấn nút bàn phím, chuột, Va chạm
-----------------------------------------------------------------------------------------
| Create File: 21:31:47 15/10/2016 by bahao
-----------------------------------------------------------------------------------------
| Last Update: 21:31:47 07/01/2017 by bahao
-----------------------------------------------------------------------------------------
*/
#ifndef __Terrain_H__
#define __Terrain_H__
//////////////////////////////////////////////////////////////////////////////
#pragma region [Include lib]
#include "GamePlugin.h"
#include "MaterialControlsCommon.h"
#include "SdkGame.h"
#include "OgreTerrain.h"
#include "OgreTerrainGroup.h"
#include "OgreTerrainQuadTreeNode.h"
#include "OgreTerrainMaterialGeneratorA.h"
#include "OgreTerrainPaging.h"
#include "SinbadCharacterControllerCommon.h"
#include "MyFileFactoryCommon.h"
#pragma endregion [Include lib]
//////////////////////////////////////////////////////////////////////////////
#if OGRE_PLATFORM == OGRE_PLATFORM_APPLE_IOS ||
```

```
OGRE_PLATFORM == OGRE_PLATFORM_APPLE
#include "macUtils.h"
#endif
//////////////////////////////////////////////////////////////////////////////
using namespace Ogre;
using namespace OgreBites;

class _OgreGameClassExport Game_Terrain : public SdkGame
{
public:

	Game_Terrain()
		: mTerrainGroup(0)
		, mTerrainPaging(0)
		, mPageManager(0)
		, mFly(false)
		, mFallVelocity(0)
		, mMode(MODE_NORMAL)
		, mLayerEdit(1)
		, mBrushSizeTerrainSpace(0.02)
		, mHeightUpdateCountDown(0)
		, mTerrainPos(1000,0,5000)
		, mTerrainsImported(false)
		,GRASS_WIDTH(40)
		,GRASS_HEIGHT(40)

	{
		mInfo["Title"] = "Level 3";
		mInfo["Description"] = "Shoot eagle.";
		mInfo["Thumbnail"] = "thumb_level3.png";
		mInfo["Category"] = "Easy";
		mInfo["Help"] = "Use the WASD keys to move Sinbad, and the
space bar to jump. "
				"Use mouse to look around and mouse wheel to zoom. Press
Q to take out or put back "
				"Sinbad's swords. With the swords equipped, you can left
click to slice vertically or "
				"right click to slice horizontally. When the swords are not
equipped, press E to "
				"start/stop a silly dance routine.";
	}
//////////////////////////////////////////////////////////////////////////////
   void testCapabilities(const RenderSystemCapabilities* caps)
	{
      if (!caps->hasCapability(RSC_VERTEX_PROGRAM) || !caps-
```

```
>hasCapability(RSC_FRAGMENT_PROGRAM))
	{
				OGRE_EXCEPT(Exception::ERR_NOT_IMPLEMENTED,
"Your graphics card does not support vertex or fragment shaders, "
			"so you cannot run this Game. Sorry!",
"Game_Terrain::testCapabilities");
	}
	}
 ////////////////////////////////////////////////////////////////////
	StringVector getRequiredPlugins()
	{
		StringVector names;
	if (!GpuProgramManager::getSingleton().isSyntaxSupported("glsles") &&
!GpuProgramManager::getSingleton().isSyntaxSupported("glsl150"))
	 names.push_back("Cg Program Manager");
		return names;
	}
////////////////////////////////////////////////////////////////////
	void doTerrainModify(Terrain* terrain, const Vector3& centrepos, Real
timeElapsed)
	{
		Vector3 tsPos;
		terrain->getTerrainPosition(centrepos, &tsPos);
#if OGRE_PLATFORM != OGRE_PLATFORM_APPLE_IOS
		if (mInputContext.isKeyDown(OIS::KC_EQUALS) ||
mInputContext.isKeyDown(OIS::KC_ADD) ||
				mInputContext.isKeyDown(OIS::KC_MINUS) ||
mInputContext.isKeyDown(OIS::KC_SUBTRACT))
		{
			switch(mMode)
			{
			case MODE_EDIT_HEIGHT:
				{
					// we need point coords
					Real terrainSize = (terrain->getSize() - 1);
					long startx = (tsPos.x -
mBrushSizeTerrainSpace) * terrainSize;
					long starty = (tsPos.y -
mBrushSizeTerrainSpace) * terrainSize;
					long endx = (tsPos.x +
mBrushSizeTerrainSpace) * terrainSize;
					long endy= (tsPos.y +
mBrushSizeTerrainSpace) * terrainSize;
					startx = std::max(startx, 0L);
					starty = std::max(starty, 0L);
```

```
                                        endx = std::min(endx, (long)terrainSize);
                                        endy = std::min(endy, (long)terrainSize);
                                        for (long y = starty; y <= endy; ++y)
                                        {
                                                for (long x = startx; x <= endx; ++x)
                                                {
                                                        Real tsXdist = (x / terrainSize) -
tsPos.x;
                                                        Real tsYdist = (y / terrainSize)
- tsPos.y;

                                                        Real weight = std::min((Real)1.0,
                                                                Math::Sqrt(tsYdist *
tsYdist + tsXdist * tsXdist) / Real(0.5 * mBrushSizeTerrainSpace));
                                                        weight = 1.0 - (weight *
weight);

                                                        float addedHeight = weight *
250.0 * timeElapsed;
                                                        float newheight;
                                                        if
(mInputContext.isKeyDown(OIS::KC_EQUALS) ||
mInputContext.isKeyDown(OIS::KC_ADD))
                                                                newheight = terrain-
>getHeightAtPoint(x, y) + addedHeight;
                                                        else
                                                                newheight = terrain-
>getHeightAtPoint(x, y) - addedHeight;
                                                        terrain->setHeightAtPoint(x, y,
newheight);

                                                }
                                        }
                                        if (mHeightUpdateCountDown == 0)
                                                mHeightUpdateCountDown =
mHeightUpdateRate;
                                }
                                break;
                        case MODE_EDIT_BLEND:
                                {
                                        TerrainLayerBlendMap* layer = terrain-
>getLayerBlendMap(mLayerEdit);
                                        // we need image coords
                                        Real imgSize = terrain-
>getLayerBlendMapSize();
```

```
                                long startx = (tsPos.x -
mBrushSizeTerrainSpace) * imgSize;
                                long starty = (tsPos.y -
mBrushSizeTerrainSpace) * imgSize;
                                long endx = (tsPos.x +
mBrushSizeTerrainSpace) * imgSize;
                                long endy= (tsPos.y +
mBrushSizeTerrainSpace) * imgSize;
                                startx = std::max(startx, 0L);
                                starty = std::max(starty, 0L);
                                endx = std::min(endx, (long)imgSize);
                                endy = std::min(endy, (long)imgSize);
                                for (long y = starty; y <= endy; ++y)
                                {
                                        for (long x = startx; x <= endx; ++x)
                                        {
                                                Real tsXdist = (x / imgSize) -
tsPos.x;
                                                Real tsYdist = (y / imgSize)  -
tsPos.y;

                                                Real weight = std::min((Real)1.0,
                                                        Math::Sqrt(tsYdist *
tsYdist + tsXdist * tsXdist) / Real(0.5 * mBrushSizeTerrainSpace));
                                                weight = 1.0 - (weight *
weight);

                                                float paint = weight *
timeElapsed;
                                                size_t imgY = imgSize - y;
                                                float val;
                                                if
(mInputContext.isKeyDown(OIS::KC_EQUALS) ||
mInputContext.isKeyDown(OIS::KC_ADD))
                                                        val = layer-
>getBlendValue(x, imgY) + paint;
                                                else
                                                        val = layer-
>getBlendValue(x, imgY) - paint;
                                                val = Math::Clamp(val, 0.0f,
1.0f);
                                                layer->setBlendValue(x, imgY,
val);

                                        }
```

```
                                        }

                                        layer->update();
                                }
                                break;
        case MODE_NORMAL:
        case MODE_COUNT:
            break;
                        };
                }
#endif

        }
//////////////////////////////////////////////////////////////////////////////
        bool frameRenderingQueued(const FrameEvent& evt)
    {
#pragma region [Funct mMode]
                if (mMode != MODE_NORMAL)
                {
                        // fire ray
                        Ray ray;
                        //ray = mCamera->getCameraToViewportRay(0.5, 0.5);
                        ray = mTrayMgr->getCursorRay(mCamera);

                        TerrainGroup::RayResult rayResult = mTerrainGroup-
>rayIntersects(ray);
                        if (rayResult.hit)
                        {
                                mEditMarker->setVisible(true);
                                mEditNode->setPosition(rayResult.position);

                                // figure out which terrains this affects
                                TerrainGroup::TerrainList terrainList;
                                Real brushSizeWorldSpace =
TERRAIN_WORLD_SIZE * mBrushSizeTerrainSpace;
                                Sphere sphere(rayResult.position,
brushSizeWorldSpace);
                                mTerrainGroup->sphereIntersects(sphere,
&terrainList);

                                for (TerrainGroup::TerrainList::iterator ti =
terrainList.begin();
                                        ti != terrainList.end(); ++ti)
                                        doTerrainModify(*ti, rayResult.position,
evt.timeSinceLastFrame);
```

```
                        }
                        else
                        {
                                mEditMarker->setVisible(false);
                        }
                }
#pragma endregion [Funct mMode]
//////////////////////////////////////////////////////////////////////////
#pragma region [Funct mFly]
                if (!mFly)
                {
                        // clamp to terrain
                        Vector3 characterPos = mChara->getCharacterPos();
                        Ray ray;
                        ray.setOrigin(Vector3(characterPos.x, mTerrainPos.y +
10000, characterPos.z));
                        ray.setDirection(Vector3::NEGATIVE_UNIT_Y);

                        TerrainGroup::RayResult rayResult = mTerrainGroup-
>rayIntersects(ray);
                        Real distanceAboveTerrain = 5;
                        Real fallSpeed = 300;
                        Real newy = characterPos.y;
                        if (rayResult.hit)
                        {
                                if (characterPos.y > rayResult.position.y +
distanceAboveTerrain)
                                {
                                        if (!(mChara->getAnime() == 11))
                                        {
                                                mFallVelocity +=
evt.timeSinceLastFrame * 20;
                                                mFallVelocity =
std::min(mFallVelocity, fallSpeed);
                                                newy = characterPos.y - mFallVelocity
* evt.timeSinceLastFrame;
                                        }
                                }
                                else if(mChara->checkJump())
                                {
                                        mChara->setMinPos(true);
                                }

                                newy = std::max(rayResult.position.y +
distanceAboveTerrain, newy);
```

```
                                mChara->setCharacterPos(Vector3(characterPos.x,
newy, characterPos.z));
                        }

                        if(isCollision())
                        {
                                LogManager::getSingleton().logMessage("True
Collision!!!!!!");
                        }
                        else
                        {
                                LogManager::getSingleton().logMessage("Not
Collision!!!!!!");
                        }
                }
#pragma endregion [Funct mFly]
//////////////////////////////////////////////////////////////////////
#pragma region [Funct mHeight]
                if (mHeightUpdateCountDown > 0)
                {
                        mHeightUpdateCountDown -= evt.timeSinceLastFrame;
                        if (mHeightUpdateCountDown <= 0)
                        {
                                mTerrainGroup->update();
                                mHeightUpdateCountDown = 0;

                        }

                }
#pragma endregion [Funct mHeight]
//////////////////////////////////////////////////////////////////////
#pragma region [Funct build terrain isDerivedDataUpdateInProgress]
                if (mTerrainGroup->isDerivedDataUpdateInProgress())
                {
                        mTrayMgr->moveWidgetToTray(mInfoLabel, TL_TOP, 0);
                        mInfoLabel->show();
                        if (mTerrainsImported)
                        {
                                mInfoLabel->setCaption("Building terrain, please
wait...");
                        }
                        else
                        {
                                mInfoLabel->setCaption("Updating textures,
patience...");
```

```
                        }
                }
                else
                {
                        mTrayMgr->removeWidgetFromTray(mInfoLabel);
                        mInfoLabel->hide();
                        if (mTerrainsImported)
                        {
                                saveTerrains(true);
                                mTerrainsImported = false;
                        }
                }
#pragma endregion [Funct build terrain isDerivedDataUpdateInProgress]
//////////////////////////////////////////////////////////////////////////
                //PTR TuanNA [Add Character- 3/1/2017]
                mChara->addTime(evt.timeSinceLastFrame);

                //PTR TuanNA [Wave Grass- 3/1/2017]
                waveGrass(evt.timeSinceLastFrame);

                return SdkGame::frameRenderingQueued(evt);  // don't forget the
parent updates!
    }
//////////////////////////////////////////////////////////////////////////
        void saveTerrains(bool onlyIfModified)
        {
                mTerrainGroup->saveAllTerrains(onlyIfModified);
        }
//////////////////////////////////////////////////////////////////////////
        bool keyPressed (const OIS::KeyEvent &e)
        {
#if OGRE_PLATFORM != OGRE_PLATFORM_APPLE_IOS
                // relay input events to character controller
                if (!mTrayMgr->isDialogVisible()) mChara->injectKeyDown(e);
                switch (e.key)
                {
                case OIS::KC_S:
                        // CTRL-S to save
                        if (mInputContext.isKeyDown(OIS::KC_LCONTROL) ||
mInputContext.isKeyDown(OIS::KC_RCONTROL))
                        {
                                saveTerrains(true);
                        }
                        else
                                return SdkGame::keyPressed(e);
```

```
                        break;
                case OIS::KC_F10:
                        // dump
                        {
                                TerrainGroup::TerrainIterator ti = mTerrainGroup-
>getTerrainIterator();
                                while (ti.hasMoreElements())
                                {
                                        Ogre::uint32 tkey = ti.peekNextKey();
                                        TerrainGroup::TerrainSlot* ts = ti.getNext();
                                        if (ts->instance && ts->instance->isLoaded())
                                        {
                                                ts->instance->_dumpTextures("terrain_"
+ StringConverter::toString(tkey), ".png");
                                        }
                                }
                        }
                        break;
                        /*
                case OIS::KC_F7:
                        // change terrain size
                        if (mTerrainGroup->getTerrainSize() == 513)
                                mTerrainGroup->setTerrainSize(1025);
                        else
                                mTerrainGroup->setTerrainSize(513);
                        break;
                case OIS::KC_F8:
                        // change terrain world size
                        if (mTerrainGroup->getTerrainWorldSize() ==
TERRAIN_WORLD_SIZE)
                                mTerrainGroup-
>setTerrainWorldSize(TERRAIN_WORLD_SIZE * 2);
                        else
                                mTerrainGroup-
>setTerrainWorldSize(TERRAIN_WORLD_SIZE);
                        break;
                        */
                default:
                        return SdkGame::keyPressed(e);
                }
#endif

                return true;
        }
//////////////////////////////////////////////////////////////////////////////
```

```
	bool keyReleased(const OIS::KeyEvent& e)
	{
		// relay input events to character controller
		if (!mTrayMgr->isDialogVisible()) mChara->injectKeyUp(e);
		return SdkGame::keyReleased(e);
	}
//////////////////////////////////////////////////////////////////////
	void itemSelected(SelectMenu* menu)
	{
		if (menu == mEditMenu)
		{
			mMode = (Mode)mEditMenu->getSelectionIndex();
		}
		else if (menu == mShadowsMenu)
		{
			mShadowMode = (ShadowMode)mShadowsMenu-
>getSelectionIndex();
			changeShadows();
		}
	}
//////////////////////////////////////////////////////////////////////
	void checkBoxToggled(CheckBox* box)
	{
		if (box == mFlyBox)
		{
			mFly = mFlyBox->isChecked();
		}
	}
//////////////////////////////////////////////////////////////////////
#if (OGRE_PLATFORM == OGRE_PLATFORM_APPLE_IOS) ||
(OGRE_PLATFORM == OGRE_PLATFORM_ANDROID)
	bool touchPressed(const OIS::MultiTouchEvent& evt)
	{
		// relay input events to character controller
		if (!mTrayMgr->isDialogVisible()) mChara-
>injectMouseDown(evt);
		return SdkGame::touchPressed(evt);
	}
//////////////////////////////////////////////////////////////////////
	bool touchMoved(const OIS::MultiTouchEvent& evt)
	{
		// relay input events to character controller
		if (!mTrayMgr->isDialogVisible()) mChara-
>injectMouseMove(evt);
		return SdkGame::touchMoved(evt);
```

```
	}
//////////////////////////////////////////////////////////////////////////////
#else
	bool mouseMoved(const OIS::MouseEvent& evt)
	{
		// relay input events to character controller
		if (!mTrayMgr->isDialogVisible()) mChara-
>injectMouseMove(evt);
		return SdkGame::mouseMoved(evt);
	}
//////////////////////////////////////////////////////////////////////////////
	bool mousePressed(const OIS::MouseEvent& evt, OIS::MouseButtonID
id)
	{
		// relay input events to character controller
		if (!mTrayMgr->isDialogVisible()) mChara-
>injectMouseDown(evt, id);
		// play a single sound
		engine->play3D("../../media/sound/explosion.wav",
			vec3df(0,0,0), false, false, true);
		return SdkGame::mousePressed(evt, id);
	}
#endif
//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
protected:
	MaterialControlsContainer mMaterialControlsContainer;
	TerrainGlobalOptions* mTerrainGlobals;
	TerrainGroup* mTerrainGroup;
	bool mPaging;
	TerrainPaging* mTerrainPaging;
	PageManager* mPageManager;
#ifdef PAGING
	/// This class just pretends to provide prcedural page content to avoid page
loading
	class DummyPageProvider : public PageProvider
	{
	public:
		bool preparePROceduralPage(Page* page, PagedWorldSection*
section) { return true; }
		bool loadProceduralPage(Page* page, PagedWorldSection*
section) { return true; }
		bool unloadProceduralPage(Page* page, PagedWorldSection*
section) { return true; }
		bool unpreparePROceduralPage(Page* page, PagedWorldSection*
```

```
section) { return true; }
        };
        DummyPageProvider mDummyPageProvider;
#endif
        bool mFly;
        Real mFallVelocity;

        Mode mMode;
        ShadowMode mShadowMode;
        Ogre::uint8 mLayerEdit;
        Real mBrushSizeTerrainSpace;
        SceneNode* mEditNode;
        Entity* mEditMarker;
        Real mHeightUpdateCountDown;
        Real mHeightUpdateRate;
        Vector3 mTerrainPos;
        SelectMenu* mEditMenu;
        SelectMenu* mShadowsMenu;
        CheckBox* mFlyBox;
        OgreBites::Label* mInfoLabel;
        bool mTerrainsImported;
        ShadowCameraSetupPtr mPSSMSetup;
        SceneNode* mBodyNode1;
        Entity* mBodyEnt1;
        Ogre::Real mMechLength;

        typedef std::list<Entity*> EntityList;
        EntityList mHouseList;

        //PTR TuanNA [Add grass into map- 11/12/2016]
        const Real GRASS_WIDTH;
        const Real GRASS_HEIGHT;
        StaticGeometry* mField;
        // start the sound engine with default parameters
        ISoundEngine* engine;
        ISound* music;
//////////////////////////////////////////////////////////////////////////
        void defineTerrain(long x, long y, bool flat = false)
        {
                // if a file is available, use it
                // if not, generate file from import

                // Usually in a real project you'll know whether the compact terrain
data is
                // available or not; I'm doing it this way to save distribution size
```

```
		if (flat)
		{
			mTerrainGroup->defineTerrain(x, y, 0.0f);
		}
		else
		{
			String filename = mTerrainGroup->generateFilename(x, y);
			if
(ResourceGroupManager::getSingleton().resourceExists(mTerrainGroup-
>getResourceGroup(), filename))
			{
				mTerrainGroup->defineTerrain(x, y);
			}
			else
			{
				Image img;
				getTerrainImage(x % 2 != 0, y % 2 != 0, img);
				mTerrainGroup->defineTerrain(x, y, &img);
				mTerrainsImported = true;
			}

		}
	}
//////////////////////////////////////////////////////////////////////////
	void getTerrainImage(bool flipX, bool flipY, Image& img)
	{
		img.load("terrain.png",
ResourceGroupManager::DEFAULT_RESOURCE_GROUP_NAME);
		if (flipX)
			img.flipAroundY();
		if (flipY)
			img.flipAroundX();

	}
//////////////////////////////////////////////////////////////////////////
	void initBlendMaps(Terrain* terrain)
	{
		TerrainLayerBlendMap* blendMap0 = terrain-
>getLayerBlendMap(1);
		TerrainLayerBlendMap* blendMap1 = terrain-
>getLayerBlendMap(2);
		Real minHeight0 = 70;
		Real fadeDist0 = 40;
		Real minHeight1 = 70;
```

```
		Real fadeDist1 = 15;
		float* pBlend1 = blendMap1->getBlendPointer();
		float* pBlend0 = blendMap0->getBlendPointer();
		for (Ogre::uint16 y = 0; y < terrain->getLayerBlendMapSize();
++y)
		{
			for (Ogre::uint16 x = 0; x < terrain-
>getLayerBlendMapSize(); ++x)
			{
				Real tx, ty;

				blendMap0->convertImageToTerrainSpace(x, y,
&tx, &ty);
				Real height = terrain-
>getHeightAtTerrainPosition(tx, ty);
				Real val = (height - minHeight0) / fadeDist0;
				val = Math::Clamp(val, (Real)0, (Real)1);
				*pBlend0++ = val;

				val = (height - minHeight1) / fadeDist1;
				val = Math::Clamp(val, (Real)0, (Real)1);
				*pBlend1++ = val;


			}
		}
		blendMap0->dirty();
		blendMap1->dirty();
		//blendMap0->loadImage("blendmap1.png",
ResourceGroupManager::DEFAULT_RESOURCE_GROUP_NAME);
		blendMap0->update();
		blendMap1->update();

		// set up a colour map
		/*
		if (!terrain->getGlobalColourMapEnabled())
		{
			terrain->setGlobalColourMapEnabled(true);
			Image colourMap;
			colourMap.load("testcolourmap.jpg",
ResourceGroupManager::DEFAULT_RESOURCE_GROUP_NAME);
			terrain->getGlobalColourMap()->loadImage(colourMap);
		}
		*/
```

```
        }
//////////////////////////////////////////////////////////////////////////
        void configureTerrainDefaults(Light* l)
        {
                // Configure global
                mTerrainGlobals->setMaxPixelError(8);
                // testing composite map
                mTerrainGlobals->setCompositeMapDistance(3000);
                //mTerrainGlobals->setUseRayBoxDistanceCalculation(true);
                //mTerrainGlobals->getDefaultMaterialGenerator()-
>setDebugLevel(1);
                //mTerrainGlobals->setLightMapSize(256);

#if OGRE_NO_GLES3_SUPPORT == 1
      // Disable the lightmap for OpenGL ES 2.0. The minimum number of
Gamers allowed is 8(as opposed to 16 on desktop).
      // Otherwise we will run over the limit by just one. The minimum was raised
to 16 in GL ES 3.0.
      if (Ogre::Root::getSingletonPtr()->getRenderSystem()-
>getName().find("OpenGL ES 2") != String::npos)
      {
         TerrainMaterialGeneratorA::SM2Profile* matProfile =

static_cast<TerrainMaterialGeneratorA::SM2Profile*>(mTerrainGlobals-
>getDefaultMaterialGenerator()->getActiveProfile());
         matProfile->setLightmapEnabled(false);
      }
#endif
                // Important to set these so that the terrain knows what to use for
derived (non-realtime) data
                mTerrainGlobals->setLightMapDirection(l-
>getDerivedDirection());
                mTerrainGlobals->setCompositeMapAmbient(mSceneMgr-
>getAmbientLight());
                //mTerrainGlobals->setCompositeMapAmbient(ColourValue::Red);
                mTerrainGlobals->setCompositeMapDiffuse(l-
>getDiffuseColour());

                // Configure default import settings for if we use imported image
                Terrain::ImportData& defaultimp = mTerrainGroup-
>getDefaultImportSettings();
                defaultimp.terrainSize = TERRAIN_SIZE;
                defaultimp.worldSize = TERRAIN_WORLD_SIZE;
                defaultimp.inputScale = 600;
                defaultimp.minBatchSize = 33;
```

```
		defaultimp.maxBatchSize = 65;
		// textures
		defaultimp.layerList.resize(3);
		defaultimp.layerList[0].worldSize = 100;

	defaultimp.layerList[0].textureNames.push_back("dirt_grayrocky_diffuse
specular.dds");

	defaultimp.layerList[0].textureNames.push_back("dirt_grayrocky_normal
height.dds");
		defaultimp.layerList[1].worldSize = 30;
		defaultimp.layerList[1].textureNames.push_back("grass_green-
01_diffusespecular.dds");
		defaultimp.layerList[1].textureNames.push_back("grass_green-
01_normalheight.dds");
		defaultimp.layerList[2].worldSize = 200;

	defaultimp.layerList[2].textureNames.push_back("growth_weirdfungus-
03_diffusespecular.dds");

	defaultimp.layerList[2].textureNames.push_back("growth_weirdfungus-
03_normalheight.dds");


	}
//////////////////////////////////////////////////////////////////////////
	void addTextureDebugOverlay(TrayLocation loc, TexturePtr tex, size_t i)
	{
		addTextureDebugOverlay(loc, tex->getName(), i);
	}
//////////////////////////////////////////////////////////////////////////
	void addTextureDebugOverlay(TrayLocation loc, const String& texname,
size_t i)
	{
		// Create material
		String matName = "Ogre/DebugTexture" +
StringConverter::toString(i);
		MaterialPtr debugMat =
MaterialManager::getSingleton().getByName(matName);
		if (debugMat.isNull())
		{
			debugMat =
MaterialManager::getSingleton().create(matName,

	ResourceGroupManager::DEFAULT_RESOURCE_GROUP_NAME);
```

```
		}
		Pass* p = debugMat->getTechnique(0)->getPass(0);
		p->removeAllTextureUnitStates();
		p->setLightingEnabled(false);
		TextureUnitState *t = p->createTextureUnitState(texname);
		t->setTextureAddressingMode(TextureUnitState::TAM_CLAMP);

		// create template
		if
(!OverlayManager::getSingleton().hasOverlayElement("Ogre/DebugTexOverlay",
true))
		{
			OverlayElement* e =
OverlayManager::getSingleton().createOverlayElement("Panel",
"Ogre/DebugTexOverlay", true);
			e->setMetricsMode(GMM_PIXELS);
			e->setWidth(128);
			e->setHeight(128);
		}

		// add widget
		String widgetName = "DebugTex"+ StringConverter::toString(i);
		Widget* w = mTrayMgr->getWidget(widgetName);
		if (!w)
		{
			w = mTrayMgr->createDecorWidget(loc, widgetName,
"Ogre/DebugTexOverlay");
		}
		w->getOverlayElement()->setMaterialName(matName);

	}
//////////////////////////////////////////////////////////////////////////
	void addTextureShadowDebugOverlay(TrayLocation loc, size_t num)
	{
		for (size_t i = 0; i < num; ++i)
		{
			TexturePtr shadowTex = mSceneMgr-
>getShadowTexture(i);
			addTextureDebugOverlay(loc, shadowTex, i);

		}

	}
//////////////////////////////////////////////////////////////////////////
	MaterialPtr buildDepthShadowMaterial(const String& textureName)
```

```
	{
		String matName = "DepthShadows/" + textureName;

		MaterialPtr ret =
MaterialManager::getSingleton().getByName(matName);
		if (ret.isNull())
		{
			MaterialPtr baseMat =
MaterialManager::getSingleton().getByName("Ogre/shadow/depth/integrated/pss
m");
			ret = baseMat->clone(matName);
			Pass* p = ret->getTechnique(0)->getPass(0);
			p->getTextureUnitState("diffuse")-
>setTextureName(textureName);

			Vector4 splitPoints;
			const PSSMShadowCameraSetup::SplitPointList&
splitPointList =

	static_cast<PSSMShadowCameraSetup*>(mPSSMSetup.get())-
>getSplitPoints();
			for (int i = 0; i < 3; ++i)
			{
				splitPoints[i] = splitPointList[i];
			}
			p->getFragmentProgramParameters()-
>setNamedConstant("pssmSplitPoints", splitPoints);


		}

		return ret;
	}
////////////////////////////////////////////////////////////////////////
	void changeShadows()
	{
		configureShadows(mShadowMode != SHADOWS_NONE,
mShadowMode == SHADOWS_DEPTH);
	}
////////////////////////////////////////////////////////////////////////
	void configureShadows(bool enabled, bool depthShadows)
	{
		TerrainMaterialGeneratorA::SM2Profile* matProfile =

	static_cast<TerrainMaterialGeneratorA::SM2Profile*>(mTerrainGlobals-
```

```
>getDefaultMaterialGenerator()->getActiveProfile());
		matProfile->setReceiveDynamicShadowsEnabled(enabled);
#ifdef SHADOWS_IN_LOW_LOD_MATERIAL
		matProfile->setReceiveDynamicShadowsLowLod(true);
#else
		matProfile->setReceiveDynamicShadowsLowLod(false);
#endif

		// Default materials
		for (EntityList::iterator i = mHouseList.begin(); i !=
mHouseList.end(); ++i)
		{
			(*i)->setMaterialName("Examples/TudorHouse");
		}

		if (enabled)
		{
			// General scene setup
			mSceneMgr-
>setShadowTechnique(SHADOWTYPE_TEXTURE_ADDITIVE_INTEGRATE
D);
			mSceneMgr->setShadowFarDistance(3000);

			// 3 textures per directional light (PSSM)
			mSceneMgr-
>setShadowTextureCountPerLightType(Ogre::Light::LT_DIRECTIONAL, 3);

			if (mPSSMSetup.isNull())
			{
				// shadow camera setup
				PSSMShadowCameraSetup* pssmSetup = new
PSSMShadowCameraSetup();
				pssmSetup->setSplitPadding(mCamera-
>getNearClipDistance());
				pssmSetup->calculateSplitPoints(3, mCamera-
>getNearClipDistance(), mSceneMgr->getShadowFarDistance());
				pssmSetup->setOptimalAdjustFactor(0, 2);
				pssmSetup->setOptimalAdjustFactor(1, 1);
				pssmSetup->setOptimalAdjustFactor(2, 0.5);

				mPSSMSetup.bind(pssmSetup);

			}
			mSceneMgr->setShadowCameraSetup(mPSSMSetup);
```

```
		if (depthShadows)
		{
			mSceneMgr->setShadowTextureCount(3);
			mSceneMgr->setShadowTextureConfig(0, 2048,
2048, PF_FLOAT32_R);
			mSceneMgr->setShadowTextureConfig(1, 1024,
1024, PF_FLOAT32_R);
			mSceneMgr->setShadowTextureConfig(2, 1024,
1024, PF_FLOAT32_R);
			mSceneMgr->setShadowTextureSelfShadow(true);
			mSceneMgr-
>setShadowCasterRenderBackFaces(true);

			MaterialPtr houseMat =
buildDepthShadowMaterial("fw12b.jpg");
			for (EntityList::iterator i = mHouseList.begin(); i !=
mHouseList.end(); ++i)
			{
				(*i)->setMaterial(houseMat);
			}

		}
		else
		{
			mSceneMgr->setShadowTextureCount(3);
			mSceneMgr->setShadowTextureConfig(0, 2048,
2048, PF_X8B8G8R8);
			mSceneMgr->setShadowTextureConfig(1, 1024,
1024, PF_X8B8G8R8);
			mSceneMgr->setShadowTextureConfig(2, 1024,
1024, PF_X8B8G8R8);
			mSceneMgr->setShadowTextureSelfShadow(false);
			mSceneMgr-
>setShadowCasterRenderBackFaces(false);
			mSceneMgr-
>setShadowTextureCasterMaterial(StringUtil::BLANK);
		}

		matProfile-
>setReceiveDynamicShadowsDepth(depthShadows);
		matProfile-
>setReceiveDynamicShadowsPSSM(static_cast<PSSMShadowCameraSetup*>(
mPSSMSetup.get()));

		//addTextureShadowDebugOverlay(TL_RIGHT, 3);
```

```
		}
		else
		{
			mSceneMgr-
>setShadowTechnique(SHADOWTYPE_NONE);
		}


	}
//////////////////////////////////////////////////////////////////////////
	/*-----------------------------------------------------------------------------
	| Extends setupView to change some initial camera settings for this Game.
	-----------------------------------------------------------------------------*/
//	void setupView()
//	{
//		//SdkGame::setupView();
//
//		mCamera->setPosition(mTerrainPos + Vector3(1683, 50, 2116));
//		mCamera->lookAt(Vector3(1963, 50, 1660));
//		mCamera->setNearClipDistance(0.1);
//		mCamera->setFarClipDistance(50000);
//
//		if (mRoot->getRenderSystem()->getCapabilities()-
>hasCapability(RSC_INFINITE_FAR_PLANE))
//	{
//	   mCamera->setFarClipDistance(0);   // enable infinite far clip distance if
we can
//	}
//	}
//////////////////////////////////////////////////////////////////////////
	void setupControls()
	{
		mTrayMgr->showCursor();

		// make room for the controls
		mTrayMgr->showLogo(TL_TOPRIGHT);
		mTrayMgr->showFrameStats(TL_TOPRIGHT);
		mTrayMgr->toggleAdvancedFrameStats();

		mInfoLabel = mTrayMgr->createLabel(TL_TOP, "TInfo", "", 350);

		mEditMenu = mTrayMgr->createLongSelectMenu(TL_BOTTOM,
"EditMode", "Edit Mode", 370, 250, 3);
```

```
		mEditMenu->addItem("None");
		mEditMenu->addItem("Elevation");
		mEditMenu->addItem("Blend");
		mEditMenu->selectItem(0);  // no edit mode

		mFlyBox = mTrayMgr->createCheckBox(TL_BOTTOM, "Fly",
"Fly");
		mFlyBox->setChecked(false, false);

		mShadowsMenu = mTrayMgr-
>createLongSelectMenu(TL_BOTTOM, "Shadows", "Shadows", 370, 250, 3);
		mShadowsMenu->addItem("None");
		mShadowsMenu->addItem("Colour Shadows");
		mShadowsMenu->addItem("Depth Shadows");
		mShadowsMenu->selectItem(0);  // no edit mode

		// a friendly reminder
		StringVector names;
		names.push_back("Help");
		mTrayMgr->createParamsPanel(TL_TOPLEFT, "Help", 100,
names)->setParamValue(0, "H/F1");
	}
//////////////////////////////////////////////////////////////////////
	//PTR TuanNA begin comment
	//[Init GrassVertex- 11/12/2016]
#if OGRE_COMPILER == OGRE_COMPILER_MSVC
#	pragma pack(push, 1)
#endif
	struct GrassVertex
	{
		float x, y, z;
		float nx, ny, nz;
		float u, v;
	};
#if OGRE_COMPILER == OGRE_COMPILER_MSVC
#	pragma pack(pop)
#endif
	//PTR TuanNA end comment
//////////////////////////////////////////////////////////////////////
////////////-----------------------------------//////////////////////
//////////////////////////////////////////////////////////////////////
//PTR TuanNA begin comment
//[SETTUP CONTENT APP- 3/1/2017]
	void setupContent()
	{
```

```
                //PTR TuanNA [Create Map game- 3/1/2017]
                createscene();
                createCharacter();
                createSound();
        }
//PTR TuanNA end comment
//////////////////////////////////////////////////////////////////////////
        void createSound()
        {
                // Update terrain at max 20fps
                mHeightUpdateRate = 1.0 / 20.0;

                engine = createIrrKlangDevice();

                CMyFileFactory* factory = new CMyFileFactory();
                engine->addFileFactory(factory);
                factory->drop(); // we don't need it anymore, delete it

                // play a single sound
                engine->play3D("../../media/sound/XinChaoVietNam_Jmi.flac",
                        vec3df(0,0,0), true, false, true);
        }
//////////////////////////////////////////////////////////////////////////
        bool isCollision()
        {
                // fire ray
                // execute the query, returns a vector of hits
                RaySceneQuery* mRaySceneQuery = mSceneMgr-
>createRayQuery(Ray());

                Ray ray;

                ray.setOrigin(mChara->getCharacterPos());
                ray.setDirection(mChara->getGoalDirection());

                // create a query object
                mRaySceneQuery->setRay(ray);

                RaySceneQueryResult &result = mRaySceneQuery->execute();
                RaySceneQueryResult::iterator itr;
                for (itr = result.begin(); itr != result.end(); itr++) {

                        if ((itr->movable->getName() != "") && itr->distance <
100) {
                                LogManager::getSingleton().logMessage("That is " +
```

```
itr->movable->getName());
                        }

                        if (itr->movable->getName().compare("SinbadBody1")==0
&& itr->distance<mMechLength/2) {
                                return true;
                        }
                }
                return false;
        }
//////////////////////////////////////////////////////////////////////
//////////////----------------------------------------//////////////////////
//////////////////////////////////////////////////////////////////////
        void createCharacter()
        {
                // disable default camera control so the character can do its own
                mCameraMan->setStyle(CS_MANUAL);

                mChara = new SinbadCharacterController(mCamera);

                mChara->setCharacterPos(mTerrainPos);

                // create main model
                mBodyNode1 = mSceneMgr->getRootSceneNode()-
>createChildSceneNode(mTerrainPos + Vector3(100, mTerrainGroup-
>getHeightAtWorldPosition(mTerrainPos) + 5, 0));
                mBodyEnt1 = mSceneMgr->createEntity("SinbadBody1",
"Sinbad.mesh");

                SubEntity* subEntity = mBodyEnt1->getSubEntity(6);
                subEntity->setMaterialName("Sinbad/RedClothes");

                mBodyNode1->attachObject(mBodyEnt1);
                Ogre::AxisAlignedBox box = mBodyEnt1->getBoundingBox();
                mBodyNode1->showBoundingBox(true);
                mBodyNode1->setScale(0.5,0.5,0.5);
                mMechLength = box.getSize().x;
        }
//////////////////////////////////////////////////////////////////////
        void createscene()
        {
#pragma region [Create Terrain]
                bool blankTerrain = false;
                //blankTerrain = true;
```

```
                mTerrainGlobals = OGRE_NEW TerrainGlobalOptions();


        ResourceGroupManager::getSingleton().createResourceGroup("Terrain");

        ResourceGroupManager::getSingleton().addResourceLocation(mFSLayer-
>getWritablePath(""), "FileSystem", "Terrain", false, false);

                mEditMarker = mSceneMgr->createEntity("editMarker",
"sphere.mesh");
                mEditNode = mSceneMgr->getRootSceneNode()-
>createChildSceneNode();
                mEditNode->attachObject(mEditMarker);
                mEditNode->setScale(0.05, 0.05, 0.05);

                setupControls();

                mCameraMan->setTopSpeed(50);

                setDragLook(false);


        MaterialManager::getSingleton().setDefaultTextureFiltering(TFO_ANISO
TROPIC);
                MaterialManager::getSingleton().setDefaultAnisotropy(7);

                mSceneMgr->setFog(FOG_LINEAR, ColourValue(0.7, 0.7, 0.8), 0,
10000, 25000);

                LogManager::getSingleton().setLogDetail(LL_BOREME);

                Vector3 lightdir(0.55, -0.3, 0.75);
                lightdir.normalise();


                Light* l = mSceneMgr->createLight("tstLight");
                l->setType(Light::LT_DIRECTIONAL);
                l->setDirection(lightdir);
                l->setDiffuseColour(ColourValue::White);
                l->setSpecularColour(ColourValue(0.4, 0.4, 0.4));

                mSceneMgr->setAmbientLight(ColourValue(0.2, 0.2, 0.2));


                mTerrainGroup = OGRE_NEW TerrainGroup(mSceneMgr,
```

```
Terrain::ALIGN_X_Z, TERRAIN_SIZE, TERRAIN_WORLD_SIZE);
		mTerrainGroup-
>setFilenameConvention(TERRAIN_FILE_PREFIX,
TERRAIN_FILE_SUFFIX);
		mTerrainGroup->setOrigin(mTerrainPos);
		mTerrainGroup->setResourceGroup("Terrain");

		configureTerrainDefaults(l);
#ifdef PAGING
		// Paging setup
		mPageManager = OGRE_NEW PageManager();
		// Since we're not loading any pages from .page files, we need a way
just
		// to say we've loaded them without them actually being loaded
		mPageManager->setPageProvider(&mDummyPageProvider);
		mPageManager->addCamera(mCamera);
		mTerrainPaging = OGRE_NEW TerrainPaging(mPageManager);
		PagedWorld* world = mPageManager->createWorld();
		mTerrainPaging->createWorldSection(world, mTerrainGroup,
2000, 3000,
			TERRAIN_PAGE_MIN_X, TERRAIN_PAGE_MIN_Y,
			TERRAIN_PAGE_MAX_X, TERRAIN_PAGE_MAX_Y);
#else
		for (long x = TERRAIN_PAGE_MIN_X; x <=
TERRAIN_PAGE_MAX_X; ++x)
			for (long y = TERRAIN_PAGE_MIN_Y; y <=
TERRAIN_PAGE_MAX_Y; ++y)
				defineTerrain(x, y, blankTerrain);
		// sync load since we want everything in place when we start
		mTerrainGroup->loadAllTerrains(true);
#endif

		if (mTerrainsImported)
		{
			TerrainGroup::TerrainIterator ti = mTerrainGroup-
>getTerrainIterator();
			while(ti.hasMoreElements())
			{
				Terrain* t = ti.getNext()->instance;
				initBlendMaps(t);
			}
		}

		mTerrainGroup->freeTemporaryResources();
#pragma endregion [Create Terrain]
```

```
		//PTR TuanNA begin comment
		//[Add grass into map- 11/12/2016]
		//createGrassMesh(String grassName, String grassMaterialName)
#pragma region [Create Grass]
		// type grass in enum GrassType
		int id = 1;
		for (id; id <= MAX_GRASS; id++)
		{
			String grassMaterialName = "Examples/GrassBlades" +
StringConverter::toString(id);
			String grassName = "grass" + StringConverter::toString(id);
			createGrassMesh(grassName, grassMaterialName);

		}

		//PTR TuanNA [Create Entity- 12/12/2016]
		Entity* grass1 = mSceneMgr->createEntity("Grass1", "grass1");
		Entity* grass2 = mSceneMgr->createEntity("Grass2", "grass2");
		Entity* grass3 = mSceneMgr->createEntity("Grass3", "grass3");
		Entity* grass4 = mSceneMgr->createEntity("Grass4", "grass4");
		Entity* grass5 = mSceneMgr->createEntity("Grass5", "grass5");
		Entity* grass6 = mSceneMgr->createEntity("Grass6", "grass6");
		Entity* grass7 = mSceneMgr->createEntity("Grass7", "grass7");
		Entity* grass8 = mSceneMgr->createEntity("Grass8", "grass8");
		Entity* grass9 = mSceneMgr->createEntity("Grass9", "grass9");
		Entity* grass10 = mSceneMgr->createEntity("Grass10",
"grass10");
		Entity* grass11 = mSceneMgr->createEntity("Grass11",
"grass11");

		// create a static geometry field, which we will populate with grass
		mField = mSceneMgr->createStaticGeometry("Field");
		mField->setRegionDimensions(Vector3(140, 140, 140));
		mField->setOrigin(Vector3(70, 70, 70));

		// add grass uniformly throughout the field, with some random
variations
		for (int x = -280; x < 280; x += 50)
		{
			for (int z = -280; z < 280; z += 50)
			{
				Vector3 pos(mTerrainPos.x + 2043 + x, 0,
mTerrainPos.z + 1715 + z);
				//Set heigh for grass always on surface
				pos.y = mTerrainGroup-
```

```
>getHeightAtWorldPosition(pos) + mTerrainPos.y;
                                Quaternion ori(Degree(Math::RangeRandom(0, 359)),
Vector3::UNIT_Y);
                                Vector3 scale(0.5, Math::RangeRandom(0.42, 0.6),
0.5);

                                mField->addEntity(grass1, pos, ori, scale);
                        }
                }

                //Max grass in group
                int groupGrass = MAX_GROUP_GRASS;

                // add grass uniformly throughout the field, with some random
variations
                for (int x = -TERRAIN_WORLD_SIZE/2; x <
TERRAIN_WORLD_SIZE/2; x += Math::RangeRandom(200, 500))
                {
                        for (int z = -TERRAIN_WORLD_SIZE/2; z <
TERRAIN_WORLD_SIZE/2; z += Math::RangeRandom(200, 500))
                        {
                                Vector3 pos(mTerrainPos.x + Math::RangeRandom(-
50, 50) + x + 50, 0, mTerrainPos.z + Math::RangeRandom(-50, 50) + z + 50);
                                //Set heigh for grass always on surface
                                pos.y = mTerrainGroup-
>getHeightAtWorldPosition(pos) + mTerrainPos.y;
                                Quaternion ori(Degree(Math::RangeRandom(0, 359)),
Vector3::UNIT_Y);
                                int scaleGrass = Math::RangeRandom(0.5, 2);
                                Vector3 scale(1*scaleGrass,
Math::RangeRandom(0.85, 1.15)*scaleGrass, 1*scaleGrass);

                                if (groupGrass > 0)
                                {
                                        switch (id)
                                        {
                                        case GRASS_RED_FLOWER:
                                                mField->addEntity(grass1, pos, ori,
scale);
                                                break;
                                        case GRASS_YELLOW_FLOWER:
                                                mField->addEntity(grass2, pos, ori,
scale);
                                                break;
                                        case GRASS_ALOE_VERA:
```

```
                                        mField->addEntity(grass3, pos, ori,
scale);
                                        break;
                                case GRASS_DIEP:
                                        mField->addEntity(grass4, pos, ori,
scale);
                                        break;
                                case GRASS_DAO_FLOWER:
                                        mField->addEntity(grass5, pos, ori,
scale);
                                        break;
                                case GRASS_ANH_TUC:
                                        mField->addEntity(grass6, pos, ori,
scale);
                                        break;
                                case GRASS_MAI:
                                        mField->addEntity(grass7, pos, ori,
scale);
                                        break;
                                case GRASS_GANG:
                                        mField->addEntity(grass8, pos, ori,
scale);
                                        break;
                                case GRASS_LAU:
                                        mField->addEntity(grass9, pos, ori,
scale);
                                        break;
                                case GRASS_LAU_2:
                                        mField->addEntity(grass10, pos, ori,
scale);
                                        break;
                                case GRASS_SAN_HO:
                                        mField->addEntity(grass11, pos, ori,
scale);
                                        break;
                                }
                                groupGrass--;
                        }
                        else
                        {
                                id = Math::RangeRandom(0, 11);
                                groupGrass = MAX_GROUP_GRASS;
                        }
                }
        }
```

```
		mField->build();  // build our static geometry (bake the grass into it)
#pragma endregion [Create Grass]
		//PTR TuanNA end comment

#pragma region [Create House]
		// create a few entities on the terrain
		Entity* e = mSceneMgr->createEntity("tudorhouse.mesh");
		Vector3 entPos(-2000, 0, 2000);
		Quaternion rot;
		entPos.y = mTerrainGroup->getHeightAtWorldPosition(entPos) +
65.5 + mTerrainPos.y;
		rot.FromAngleAxis(Degree(Math::RangeRandom(-180, 180)),
Vector3::UNIT_Y);
		SceneNode* sn = mSceneMgr->getRootSceneNode()-
>createChildSceneNode(entPos, rot);
		sn->setScale(Vector3(0.12, 0.12, 0.12));
		sn->attachObject(e);
		mHouseList.push_back(e);

		e = mSceneMgr->createEntity("tudorhouse.mesh");
		entPos = Vector3(mTerrainPos.x + 1850, 0, mTerrainPos.z +
1478);
		entPos.y = mTerrainGroup->getHeightAtWorldPosition(entPos) +
65.5 + mTerrainPos.y;
		rot.FromAngleAxis(Degree(Math::RangeRandom(-180, 180)),
Vector3::UNIT_Y);
		sn = mSceneMgr->getRootSceneNode()-
>createChildSceneNode(entPos, rot);
		sn->setScale(Vector3(0.12, 0.12, 0.12));
		sn->attachObject(e);
		mHouseList.push_back(e);

		e = mSceneMgr->createEntity("tudorhouse.mesh");
		entPos = Vector3(mTerrainPos.x + 1970, 0, mTerrainPos.z +
2180);
		entPos.y = mTerrainGroup->getHeightAtWorldPosition(entPos) +
65.5 + mTerrainPos.y;
		rot.FromAngleAxis(Degree(Math::RangeRandom(-180, 180)),
Vector3::UNIT_Y);
		sn = mSceneMgr->getRootSceneNode()-
>createChildSceneNode(entPos, rot);
		sn->setScale(Vector3(0.12, 0.12, 0.12));
		sn->attachObject(e);
		mHouseList.push_back(e);
```

```
#pragma endregion [Create House]
		mSceneMgr->setSkyBox(true, "Examples/CloudyNoonSkyBox");
#pragma region [Create Ocean]
		loadAllMaterialControlFiles(mMaterialControlsContainer);
		//Define a plane mesh that will be used for the ocean surface
		Ogre::Plane oceanSurface;
		oceanSurface.normal = Ogre::Vector3::UNIT_Y;
		oceanSurface.d = 20;
		Ogre::MeshManager::getSingleton().createPlane("OceanSurface",

	Ogre::ResourceGroupManager::DEFAULT_RESOURCE_GROUP_NAM
E,
			oceanSurface,
			TERRAIN_WORLD_SIZE*2,
TERRAIN_WORLD_SIZE*2, 50, 50, true, 1, 1, 1, Ogre::Vector3::UNIT_Z);

		Ogre::Entity*	mOceanSurfaceEnt;
		mOceanSurfaceEnt = mSceneMgr->createEntity( "OceanSurface",
"OceanSurface" );
		mOceanSurfaceEnt-
>setMaterialName(mMaterialControlsContainer[0].getMaterialName());
		entPos = Vector3(1000, 0, 5000);
		entPos.y = mTerrainGroup->getHeightAtWorldPosition(entPos) +
17;
		sn = mSceneMgr->getRootSceneNode()-
>createChildSceneNode(entPos);
		sn->attachObject(mOceanSurfaceEnt);
#pragma endregion [Create Ocean]
	}
//////////////////////////////////////////////////////////////////////////
	void createGrassMesh(String grassName, String grassMaterialName)
	{
		MeshPtr mesh =
MeshManager::getSingleton().createManual(grassName,
ResourceGroupManager::DEFAULT_RESOURCE_GROUP_NAME);

		// create a submesh with the grass material
		SubMesh* sm = mesh->createSubMesh();
		sm->setMaterialName(grassMaterialName);//PTR TuanNA
[grassMaterialName = "Examples/GrassBlades"- 11/12/2016]
		sm->useSharedVertices = false;
		sm->vertexData = OGRE_NEW VertexData();
		sm->vertexData->vertexStart = 0;
		sm->vertexData->vertexCount = 12;
		sm->indexData->indexCount = 18;
```

```
#if defined(INCLUDE_RTSHADER_SYSTEM)
		MaterialPtr grassMat =
MaterialManager::getSingleton().getByName(grassMaterialName);
		grassMat->getTechnique(0)-
>setSchemeName(Ogre::RTShader::ShaderGenerator::DEFAULT_SCHEME_N
AME);
#endif

		// specify a vertex format declaration for our mesh: 3 floats for
position, 3 floats for normal, 2 floats for UV
		VertexDeclaration* decl = sm->vertexData->vertexDeclaration;
		decl->addElement(0, 0, VET_FLOAT3, VES_POSITION);
		decl->addElement(0, sizeof(float) * 3, VET_FLOAT3,
VES_NORMAL);
		decl->addElement(0, sizeof(float) * 6, VET_FLOAT2,
VES_TEXTURE_COORDINATES, 0);

		// create a vertex buffer
		HardwareVertexBufferSharedPtr vb =
HardwareBufferManager::getSingleton().createVertexBuffer
			(decl->getVertexSize(0), sm->vertexData->vertexCount,
HardwareBuffer::HBU_STATIC_WRITE_ONLY);

		GrassVertex* verts = (GrassVertex*)vb-
>lock(HardwareBuffer::HBL_DISCARD);  // start filling in vertex data

		for (unsigned int i = 0; i < 3; i++)  // each grass mesh consists of 3
planes
		{
			// planes intersect along the Y axis with 60 degrees between
them
			Real x = Math::Cos(Degree(i * 60)) * GRASS_WIDTH / 2;
			Real z = Math::Sin(Degree(i * 60)) * GRASS_WIDTH / 2;

			for (unsigned int j = 0; j < 4; j++)  // each plane has 4
vertices
			{
				GrassVertex& vert = verts[i * 4 + j];

				vert.x = j < 2 ? -x : x;
				vert.y = j % 2 ? 0 : GRASS_HEIGHT;
				vert.z = j < 2 ? -z : z;

				// all normals point straight up
```

```
                        vert.nx = 0;
                        vert.ny = 1;
                        vert.nz = 0;

                        vert.u = j < 2 ? 0 : 1;
                        vert.v = j % 2;
                    }
                }

                vb->unlock();  // commit vertex changes

                sm->vertexData->vertexBufferBinding->setBinding(0, vb);  // bind
vertex buffer to our submesh

                // create an index buffer
                sm->indexData->indexBuffer =
HardwareBufferManager::getSingleton().createIndexBuffer
                        (HardwareIndexBuffer::IT_16BIT, sm->indexData-
>indexCount, HardwareBuffer::HBU_STATIC_WRITE_ONLY);

                // start filling in index data
                Ogre::uint16* indices = (Ogre::uint16*)sm->indexData-
>indexBuffer->lock(HardwareBuffer::HBL_DISCARD);

                for (unsigned int i = 0; i < 3; i++)  // each grass mesh consists of 3
planes
                {
                        unsigned int off = i * 4;  // each plane consists of 2 triangles

                        *indices++ = 0 + off;
                        *indices++ = 3 + off;
                        *indices++ = 1 + off;

                        *indices++ = 0 + off;
                        *indices++ = 2 + off;
                        *indices++ = 3 + off;
                }

                sm->indexData->indexBuffer->unlock();  // commit index changes
        }

        //PTR TuanNA begin comment
        //[Add grass into map- 11/12/2016]
        void waveGrass(Real timeElapsed)
        {
```

```
		static Real xinc = Math::PI * 0.3;
		static Real zinc = Math::PI * 0.44;
		static Real xpos = Math::RangeRandom(-Math::PI, Math::PI);
		static Real zpos = Math::RangeRandom(-Math::PI, Math::PI);
		static Vector4 offset(0, 0, 0, 0);

		xpos += xinc * timeElapsed;
		zpos += zinc * timeElapsed;

		// update vertex program parameters by binding a value to each
renderable
		StaticGeometry::RegionIterator regs =  mField-
>getRegionIterator();
		while (regs.hasMoreElements())
		{
			StaticGeometry::Region* reg = regs.getNext();

			// a little randomness
			xpos += reg->getCentre().x * 0.001;
			zpos += reg->getCentre().z * 0.001;
			offset.x = Math::Sin(xpos) * 4;
			offset.z = Math::Sin(zpos) * 4;

			StaticGeometry::Region::LODIterator lods = reg-
>getLODIterator();
			while (lods.hasMoreElements())
			{
				StaticGeometry::LODBucket::MaterialIterator mats =
lods.getNext()->getMaterialIterator();
				while (mats.hasMoreElements())
				{

	StaticGeometry::MaterialBucket::GeometryIterator geoms =
mats.getNext()->getGeometryIterator();
					while (geoms.hasMoreElements())
geoms.getNext()->setCustomParameter(999, offset);
				}
			}
		}
	}
	//PTR TuanNA end comment
//////////////////////////////////////////////////////////////////////////
	void _shutdown()
	{
		if (mTerrainPaging)
```

```
		{
			OGRE_DELETE mTerrainPaging;
			mTerrainPaging = 0;
			OGRE_DELETE mPageManager;
			mPageManager = 0;
		}
		else if(mTerrainGroup)
		{
			OGRE_DELETE mTerrainGroup;
			mTerrainGroup = 0;
		}

		OGRE_DELETE mTerrainGlobals;
		mTerrainGlobals = 0;


	ResourceGroupManager::getSingleton().destroyResourceGroup("Terrain")
;

		mHouseList.clear();

		SdkGame::_shutdown();
	}
//////////////////////////////////////////////////////////////////////////////
	//PTR TuanNA begin comment
	//[Clean up Grass and Ocean Surface- 11/12/2016]
	void cleanupContent()
	{
		//PTR TuanNA [Remove OceanSurface- 3/1/2017]
		MeshManager::getSingleton().remove("OceanSurface");

		//PTR TuanNA [Remove Grass- 3/1/2017]
		for (int id = 1; id <= MAX_GRASS; id++)
		{
			String grassName = "grass" + StringConverter::toString(id);
			MeshManager::getSingleton().remove(grassName);
		}

		// clean up character controller and the floor mesh
		if (mChara)
		{
			delete mChara;
			mChara = 0;
		}
```

```
                engine->drop(); // delete engine
        }
        //PTR TuanNA end comment

        //Init Main Character
        SinbadCharacterController* mChara;
        SinbadCharacterController* mChara2;
};

#endif
```

```
/*
-----------------------------------------------------------------------------
  ___                 _

 /___\__ _ _ __ ___    /_\  _ __  _ __
 //  // _` | '__/ _ \ //_\\| '_ \| '_ \

/ \_// (_| | | |  __/ / _  \ |_) | |_) \
\___/ \__, |_|  \___| \_/ \_/ .__/| .___/
     |___/              |_|   |_|
     Ogre 1.9.x Application for VC10
     https://plus.google.com/+bahaonguyen/
-----------------------------------------------------------------------------
| Filename   : Terrain.cpp
-----------------------------------------------------------------------------
| Written in : C++
-----------------------------------------------------------------------------
| Compiler   : Microsoft Visual C++ 2010 Service Pack 1.
-----------------------------------------------------------------------------
| Engine     : Ogre3D; ...............................
-----------------------------------------------------------------------------
| Written by : Ba Hao Nguyen – Sports editors, IT.
-----------------------------------------------------------------------------
| Note       :Include file Content Terrain.h và build ra file thư viện *.dll.
-----------------------------------------------------------------------------
| Create File: 21:31:19 15/10/2016 by bahao
-----------------------------------------------------------------------------
| Last Update: 21:31:19 15/10/2016 by bahao
-----------------------------------------------------------------------------
*/
#include "GamePlugin.h"
```

```
#include "Terrain.h"

using namespace Ogre;
using namespace OgreBites;

#ifndef OGRE_STATIC_LIB

GamePlugin* sp;
Game* s;

extern "C" _OgreGameExport void dllStartPlugin()
{
	s = new Game_Terrain;
	sp = OGRE_NEW GamePlugin(s->getInfo()["Title"] + " Game");
	sp->addGame(s);
	Root::getSingleton().installPlugin(sp);
}

extern "C" _OgreGameExport void dllStopPlugin()
{
	Root::getSingleton().uninstallPlugin(sp);
	OGRE_DELETE sp;
	delete s;
}
#endif
```

4, Cài đặt và trình bày về sản phẩm game

4.1, Cài đặt game

Bước 1: Sau khi build xong dự án vào thư mục D:\MyProject\LegendOfThachSanh\SourceCode\bin\Release. Vào kiểm tra 3 file config bao gồm games.cfg; plugin.cfg và resources.cfg.

Name	Date modified	Type	Size
cg.dll	2016-11-19 4:56 PM	Application extens...	8,331 KB
Game_Character.dll	2017-01-09 2:48 AM	Application extens...	136 KB
Game_DeferredShading.dll	2017-01-09 2:49 AM	Application extens...	255 KB
Game_Ocean.dll	2017-01-09 2:48 AM	Application extens...	161 KB
Game_ParticleFX.dll	2017-01-09 2:48 AM	Application extens...	129 KB
Game_Terrain.dll	2017-01-09 2:58 AM	Application extens...	204 KB
games.cfg	2016-11-19 4:56 PM	CFG File	1 KB
ikpFlac.dll	2017-01-04 10:19 ...	Application extens...	156 KB
ikpMP3.dll	2017-01-04 10:19 ...	Application extens...	160 KB
irrKlang.dll	2017-01-04 10:19 ...	Application extens...	524 KB
LegendOfThachSanh.exe	2017-01-09 2:49 AM	Application	205 KB
OgreMain.dll	2016-11-19 4:56 PM	Application extens...	6,576 KB
OgreMeshUpgrader.exe	2016-11-19 4:56 PM	Application	101 KB
OgreOverlay.dll	2016-11-19 4:56 PM	Application extens...	648 KB
OgrePaging.dll	2016-11-19 4:56 PM	Application extens...	166 KB
OgreProperty.dll	2016-11-19 4:56 PM	Application extens...	92 KB
OgreRTShaderSystem.dll	2016-11-19 4:56 PM	Application extens...	839 KB
OgreTerrain.dll	2016-11-19 4:56 PM	Application extens...	428 KB
OgreVolume.dll	2016-11-19 4:56 PM	Application extens...	225 KB
OgreXMLConverter.exe	2016-11-19 4:56 PM	Application	242 KB
OIS.dll	2016-11-19 4:56 PM	Application extens...	103 KB
Plugin_BSPSceneManager.dll	2016-11-19 4:56 PM	Application extens...	226 KB
Plugin_CgProgramManager.dll	2016-11-19 4:56 PM	Application extens...	202 KB
Plugin_OctreeSceneManager.dll	2016-11-19 4:56 PM	Application extens...	159 KB
Plugin_OctreeZone.dll	2016-11-19 4:56 PM	Application extens...	78 KB
Plugin_ParticleFX.dll	2016-11-19 4:56 PM	Application extens...	134 KB
Plugin_BSPSceneManager.dll	2016-11-19 4:56 PM	Application extens...	226 KB
Plugin_CgProgramManager.dll	2016-11-19 4:56 PM	Application extens...	202 KB
Plugin_OctreeSceneManager.dll	2016-11-19 4:56 PM	Application extens...	159 KB
Plugin_OctreeZone.dll	2016-11-19 4:56 PM	Application extens...	78 KB
Plugin_ParticleFX.dll	2016-11-19 4:56 PM	Application extens...	134 KB
Plugin_PCZSceneManager.dll	2016-11-19 4:56 PM	Application extens...	278 KB
plugins.cfg	2016-11-19 4:56 PM	CFG File	1 KB
quakemap.cfg	2016-11-19 4:56 PM	CFG File	1 KB
RenderSystem_Direct3D9.dll	2016-11-19 4:56 PM	Application extens...	572 KB

- Trong đó, games.cfg: Qui định các file thư viện game *.dll được load vào game.

```
GameFolder=.
GamePlugin=Game_Character
GamePlugin=Game_DeferredShading
GamePlugin=Game_ParticleFX
GamePlugin=Game_Terrain
GamePlugin=Game_Ocean
```

- plugin.cfg: Đặt đường dẫn tới các plugin sẽ load

```
# Defines plugins to load

# Define plugin folder
PluginFolder=.

# Define plugins
 Plugin=RenderSystem_Direct3D9
 Plugin=RenderSystem_Direct3D11
 Plugin=RenderSystem_GL
# Plugin=RenderSystem_GL3Plus
# Plugin=RenderSystem_GLES
# Plugin=RenderSystem_GLES2
 Plugin=Plugin_ParticleFX
 Plugin=Plugin_BSPSceneManager
 Plugin=Plugin_CgProgramManager
 Plugin=Plugin_PCZSceneManager
 Plugin=Plugin_OctreeZone
 Plugin=Plugin_OctreeSceneManager
```

- Và resources.cfg: Đặt đường dẫn tới các resources sẽ được Game_Brower load vào game như file mesh, âm thanh, texture,…

```
# Resources required by the sample browser and most samples.
[Essential]
Zip=../../Media/packs/SdkTrays.zip
Zip=../../Media/packs/profiler.zip
FileSystem=../../Media/thumbnails

# Common sample resources needed by many of the samples.
# Rarely used resources should be separately loaded by the
# samples which require them.
[Popular]
FileSystem=../../Media/fonts
FileSystem=../../Media/materials/programs
FileSystem=../../Media/materials/scripts
FileSystem=../../Media/materials/textures
FileSystem=../../Media/materials/textures/nvidia
FileSystem=../../Media/materials/textures/Grass
FileSystem=../../Media/materials/textures/Character
FileSystem=../../Media/models
FileSystem=../../Media/particle
FileSystem=../../Media/DeferredShadingMedia
FileSystem=../../Media/PCZAppMedia
```

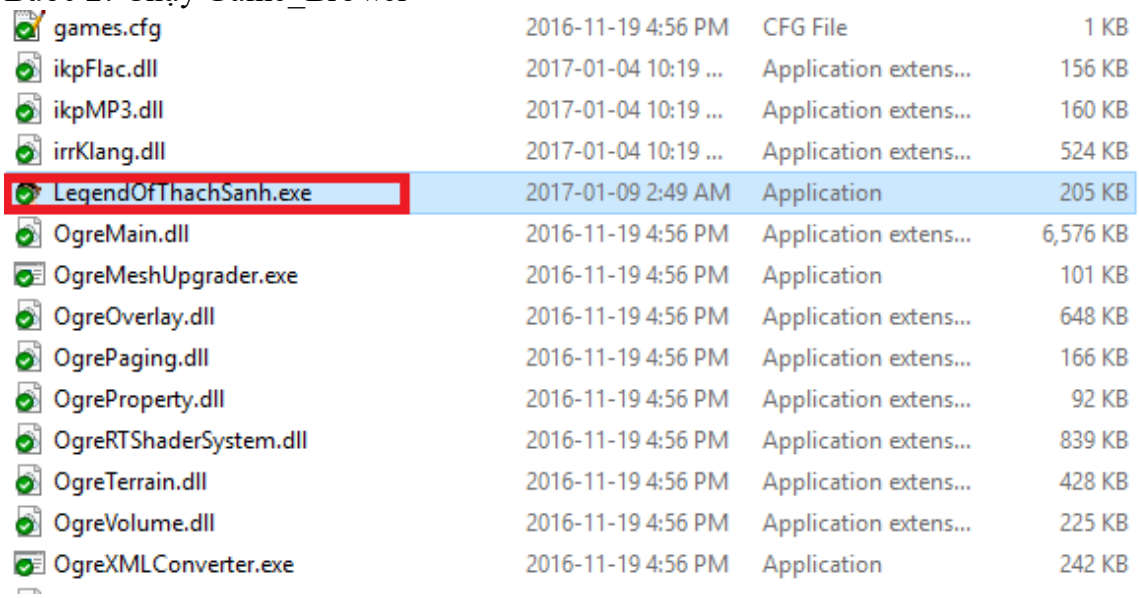
```
FileSystem=../../Media/RTShaderLib
FileSystem=../../Media/RTShaderLib/materials
FileSystem=../../Media/materials/scripts/SSAO
FileSystem=../../Media/materials/textures/SSAO
FileSystem=../../Media/volumeTerrain
Zip=../../Media/packs/cubemap.zip
Zip=../../Media/packs/cubemapsJS.zip
Zip=../../Media/packs/dragon.zip
Zip=../../Media/packs/fresneldemo.zip
Zip=../../Media/packs/ogretestmap.zip
Zip=../../Media/packs/ogredance.zip
Zip=../../Media/packs/Sinbad.zip
Zip=../../Media/packs/skybox.zip
Zip=../../Media/volumeTerrain/volumeTerrainBig.zip
Zip=../../Media/packs/snake.zip

[General]
FileSystem=../../Media

# Materials for visual tests
[Tests]
FileSystem=../../Tests/Media
```

Ngoài ra, file ogre.cfg (tự sinh ra nếu không tồn tại) trong thư mục C:\Users\<"Tên User">\Documents\Ogre\Ghadamon sẽ quyết định công nghệ render hình ảnh và cấu hình khác.

```
Render System=OpenGL Rendering Subsystem

[Direct3D9 Rendering Subsystem]
Allow DirectX9Ex=No
Allow NVPerfHUD=No
FSAA=0
Fixed Pipeline Enabled=Yes
Floating-point mode=Fastest
Full Screen=Yes
Multi device memory hint=Use minimum system memory
Rendering Device=Monitor-1-AMD Radeon HD 7700 Series
Resource Creation Policy=Create on all devices
Use Multihead=Auto
VSync=No
VSync Interval=1
Video Mode=800 x 600 @ 32-bit colour
sRGB Gamma Conversion=No
```

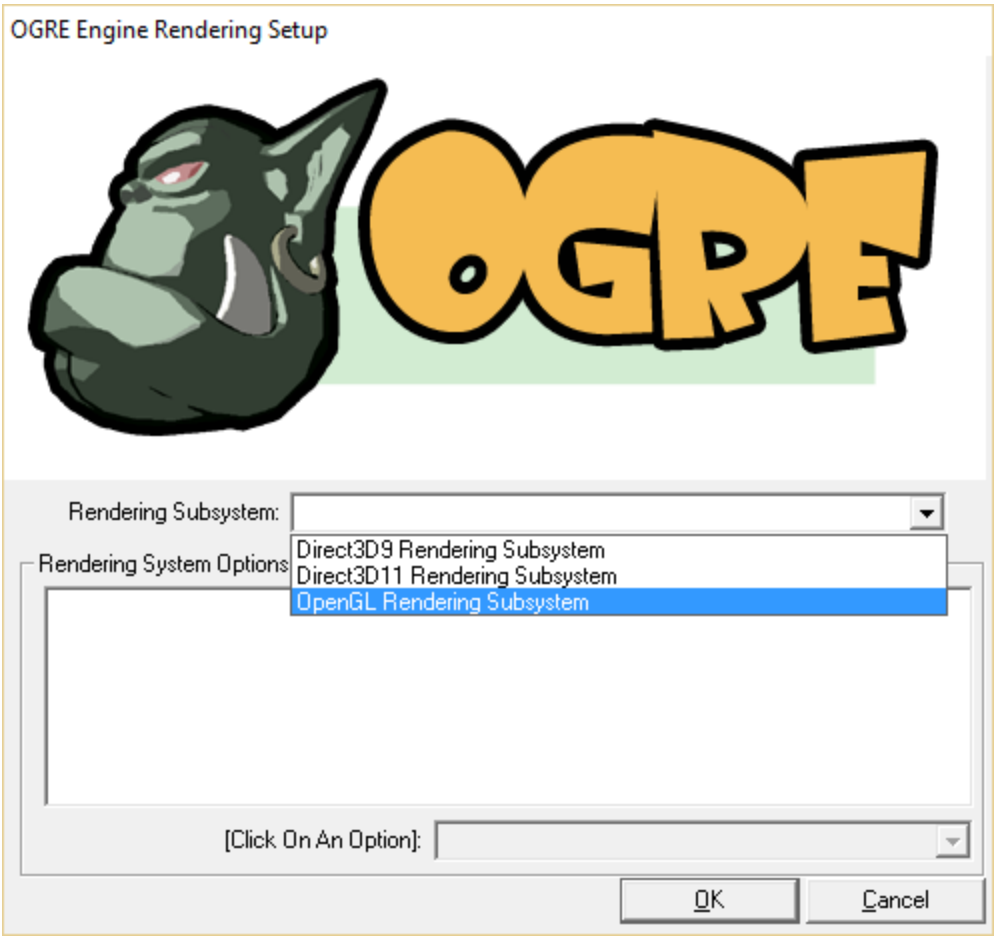
```
[Direct3D11 Rendering Subsystem]
Allow NVPerfHUD=No
Driver type=Hardware
FSAA=0
Floating-point mode=Fastest
Full Screen=Yes
Information Queue Exceptions Bottom Level=Info (exception on any message)
Max Requested Feature Levels=11.0
Min Requested Feature Levels=9.1
Rendering Device=AMD Radeon HD 7700 Series
VSync=No
VSync Interval=1
Video Mode=800 x 600 @ 32-bit colour
sRGB Gamma Conversion=No

[OpenGL Rendering Subsystem]
Colour Depth=32
Display Frequency=N/A
FSAA=0
Fixed Pipeline Enabled=Yes
Full Screen=No
RTT Preferred Mode=FBO
VSync=No
VSync Interval=1
Video Mode=1024 x 768
sRGB Gamma Conversion=No
```

Bước 2: Chạy Game_Brower



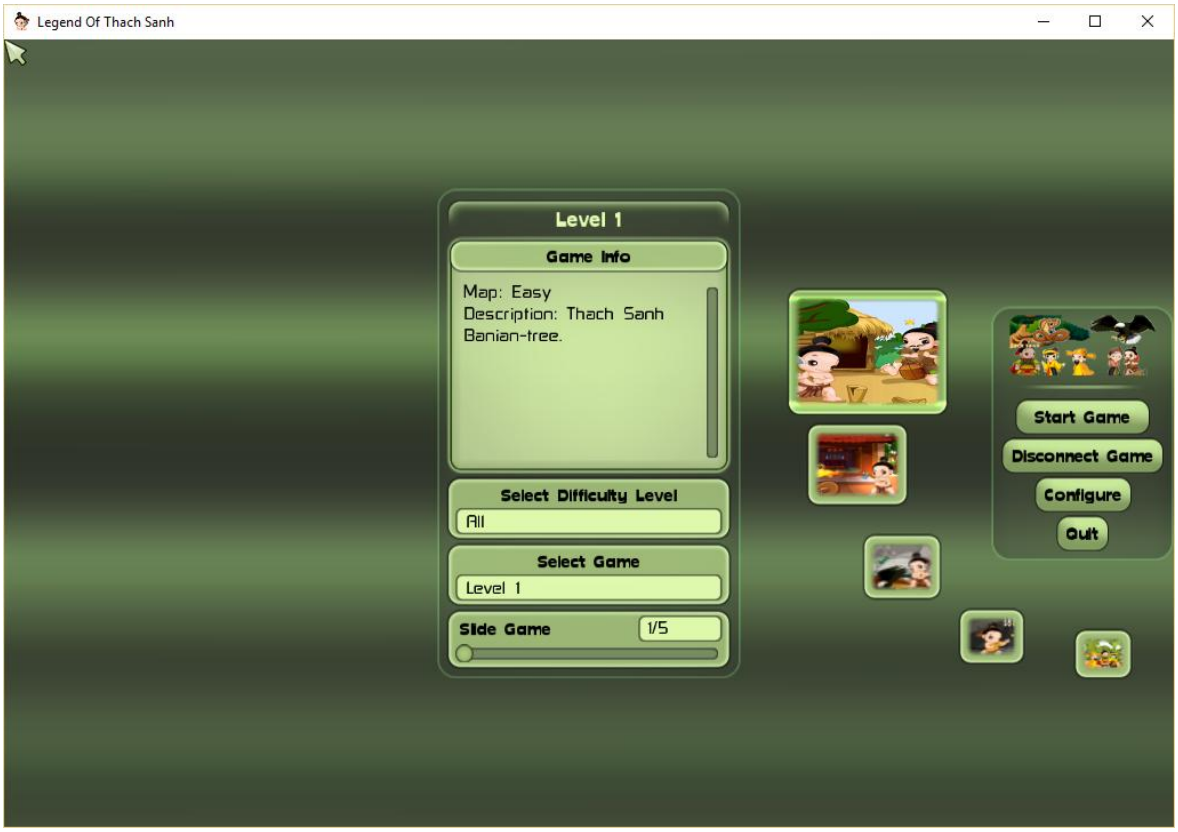
games.cfg	2016-11-19 4:56 PM	CFG File	1 KB
ikpFlac.dll	2017-01-04 10:19 ...	Application extens...	156 KB
ikpMP3.dll	2017-01-04 10:19 ...	Application extens...	160 KB
irrKlang.dll	2017-01-04 10:19 ...	Application extens...	524 KB
LegendOfThachSanh.exe	2017-01-09 2:49 AM	Application	205 KB
OgreMain.dll	2016-11-19 4:56 PM	Application extens...	6,576 KB
OgreMeshUpgrader.exe	2016-11-19 4:56 PM	Application	101 KB
OgreOverlay.dll	2016-11-19 4:56 PM	Application extens...	648 KB
OgrePaging.dll	2016-11-19 4:56 PM	Application extens...	166 KB
OgreProperty.dll	2016-11-19 4:56 PM	Application extens...	92 KB
OgreRTShaderSystem.dll	2016-11-19 4:56 PM	Application extens...	839 KB
OgreTerrain.dll	2016-11-19 4:56 PM	Application extens...	428 KB
OgreVolume.dll	2016-11-19 4:56 PM	Application extens...	225 KB
OgreXMLConverter.exe	2016-11-19 4:56 PM	Application	242 KB

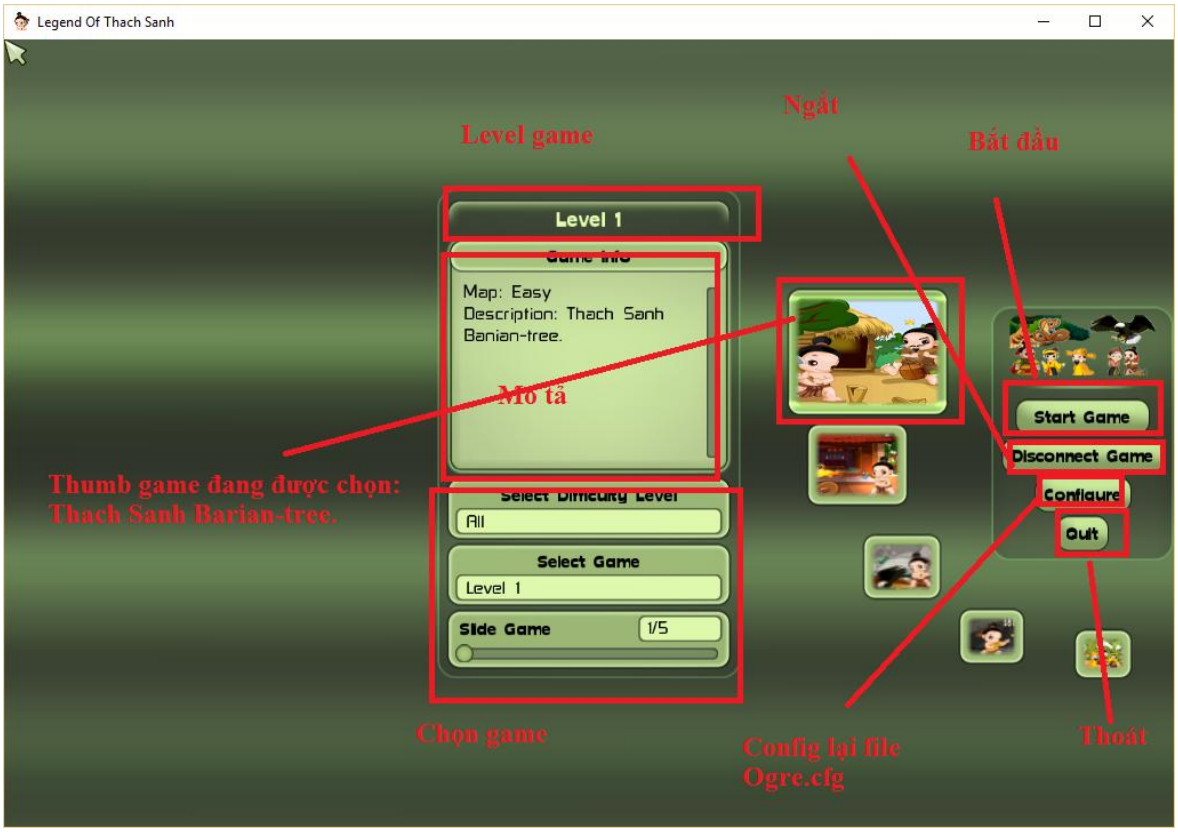
Bước 3: Chọn công nghệ Render hình ảnh



Hình 30: Chọn công nghệ Render hình ảnh

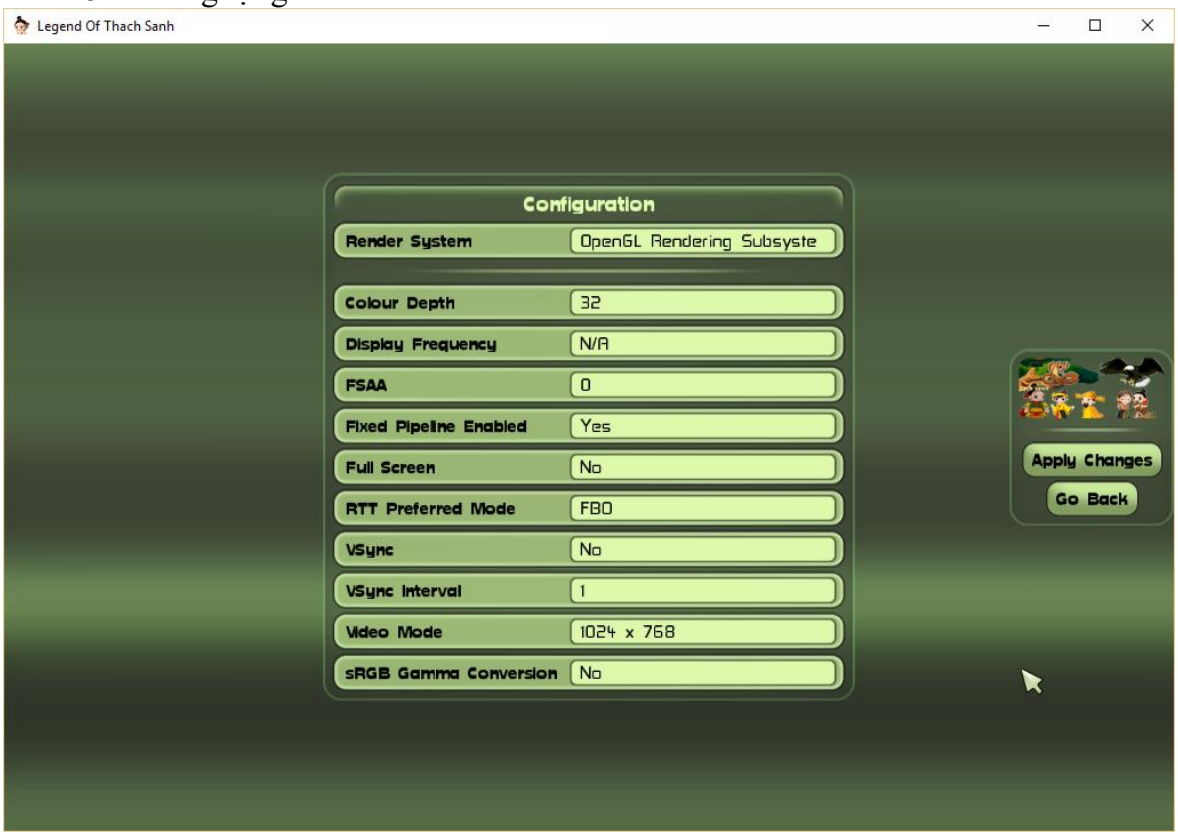
Bước 4: Vào giao diện menu game

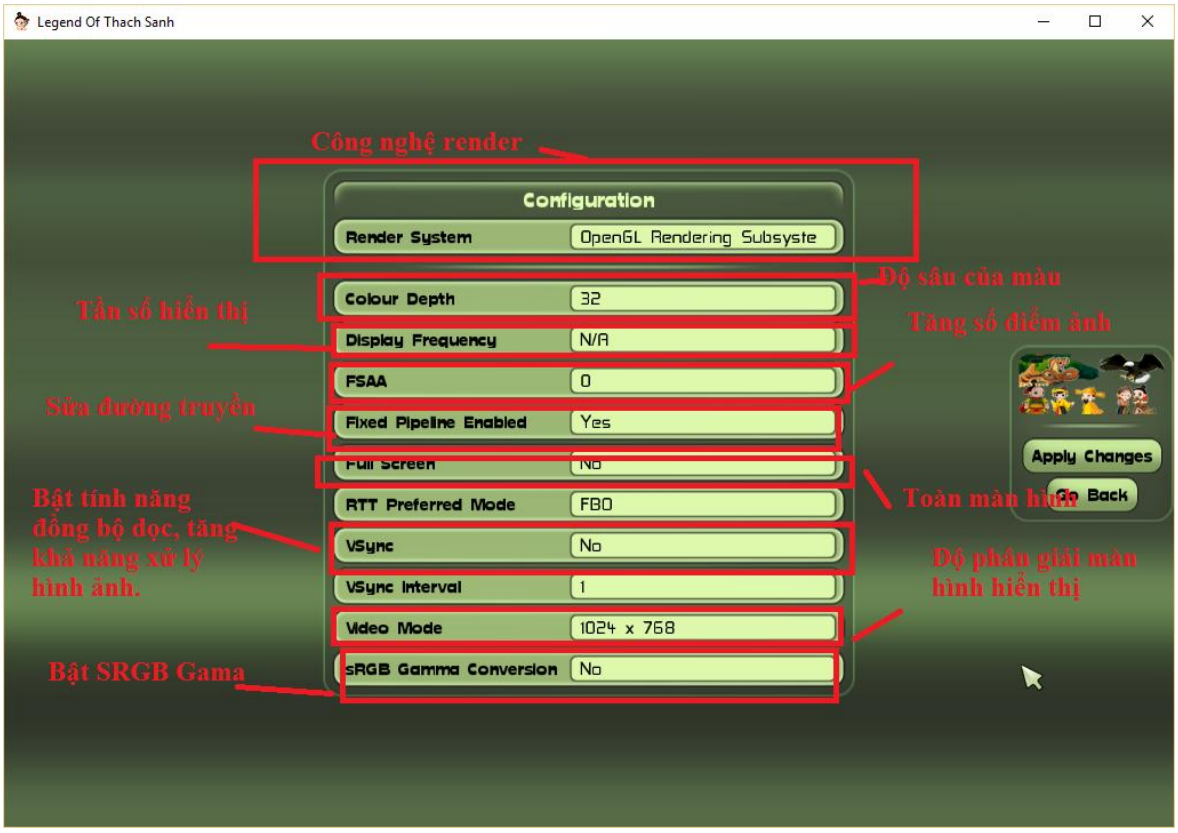




Hình 31: Vào giao diện menu game

Bước 5: Config lại game

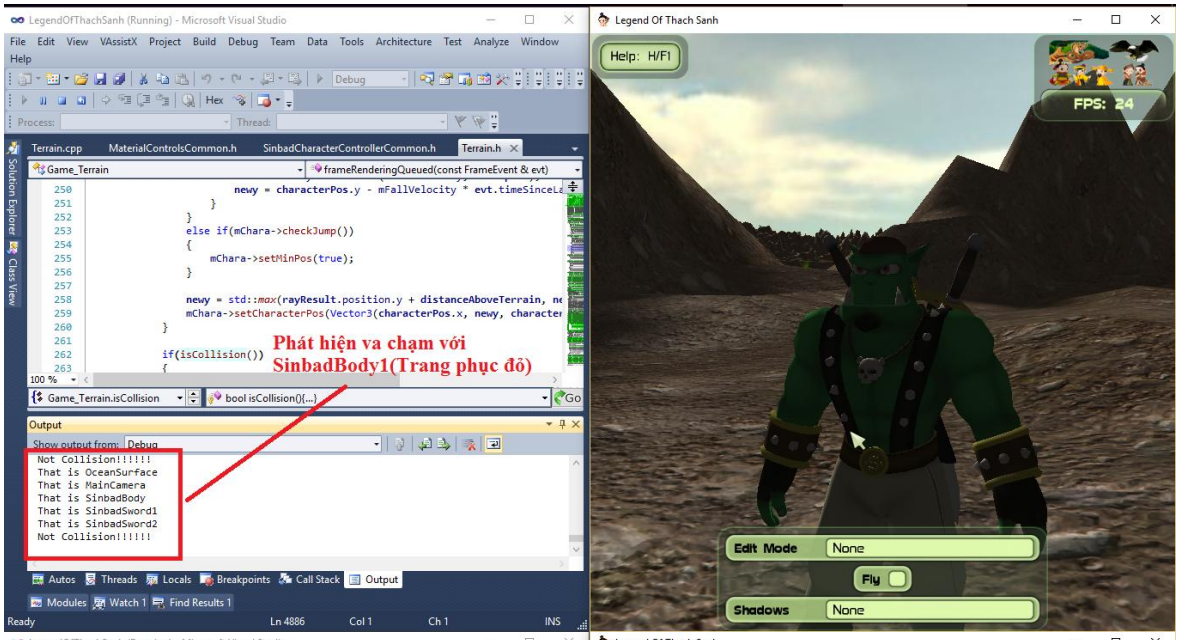


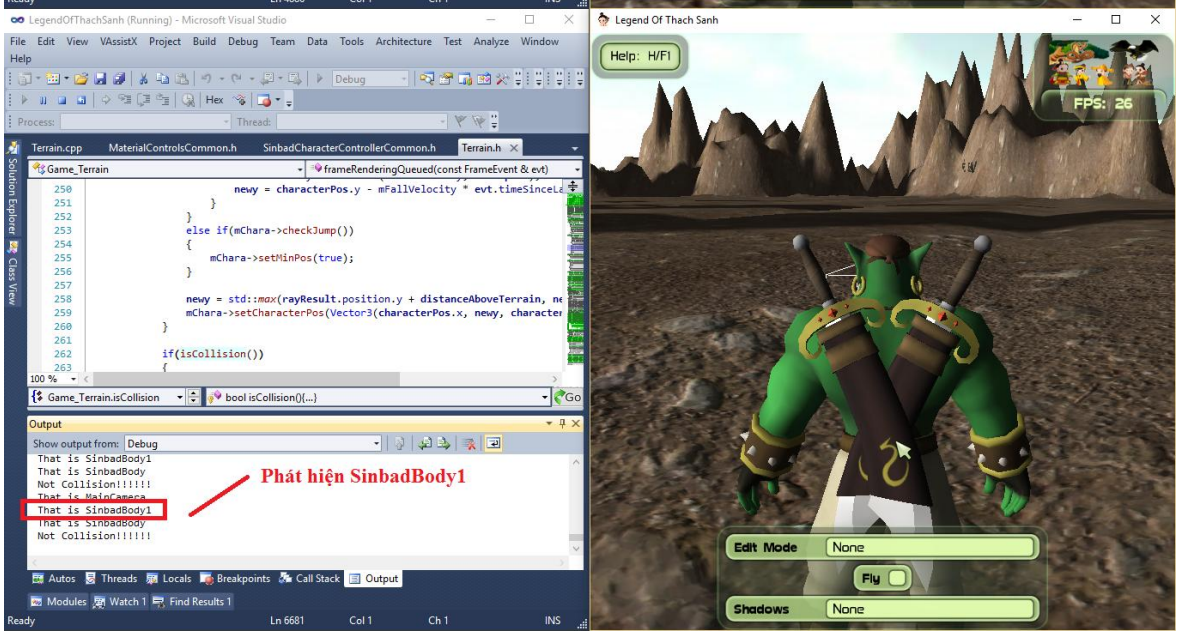


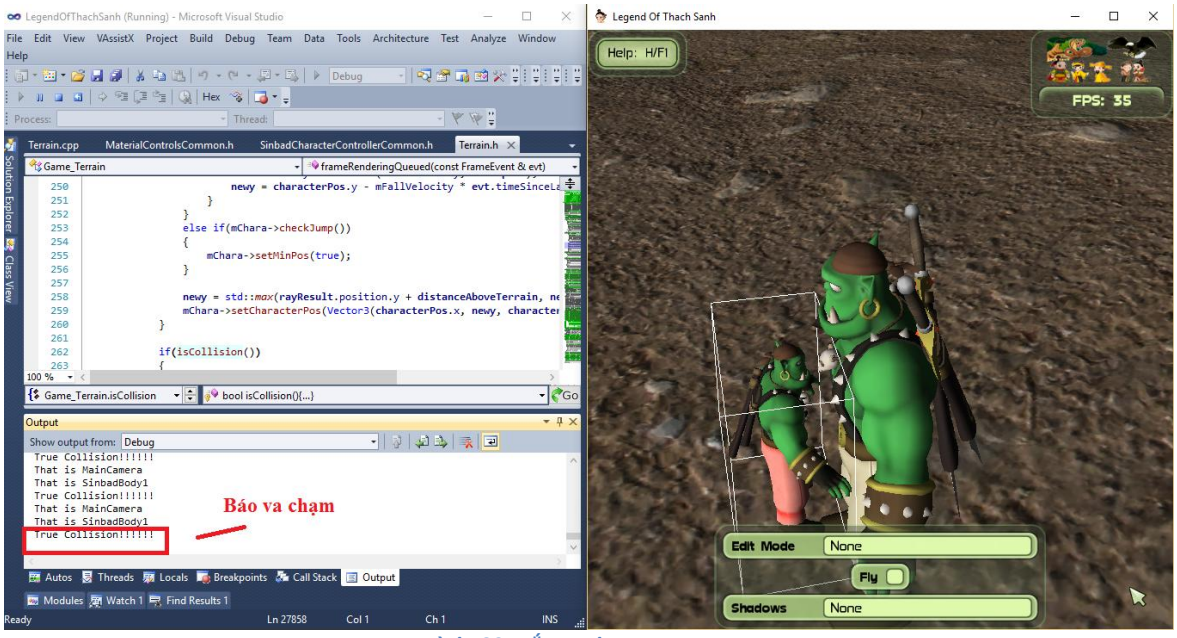
Bước 6: Chạy game với nút Start Game.

Legend Of Thach Sanh
Help: H/F1
FPS: 145
Edit Mode
None
Fly
Shadows
None

Legend Of Thach Sanh
Help: H/F1
FPS: 130
Edit Mode
None
Fly
Shadows
None


Phát hiện va chạm với
SinbadBody1(Trang phục đỏ)
Not Collision!!!!!!
That is OceanSurface
That is MainCamera
That is SinbadBody
That is SinbadSword1
That is SinbadSword2
Not Collision!!!!!!
Help: H/F1
FPS: 24
Edit Mode
None
Fly
Shadows
None


Phát hiện SinbadBody1
That is SinbadBody1
That is SinbadBody
Not Collision!!!!!!
That is SinbadBody1
That is SinbadBody
Not Collision!!!!!!
Help: H/F1
FPS: 26
Edit Mode
None
Fly
Shadows
None



Hình 32: Bắt va chạm trong game

5, Đánh giá ưu, nhược điểm của sản phẩm game

5.1, Ưu điểm

- Game cho hình ảnh chân thực, đẹp, khá sát với file design.
- FPS (Số khung hình xử lý/giây) khá nhẹ.
- Mang giá trị truyền thống, giúp người chơi hiểu hơn về chuyện cổ tích Việt Nam.

5.2, Nhược điểm

- Sản phẩm game vẫn chưa thể hoàn tất phần nội dụng game.
- Design chủ yếu sử dụng từ nguồn trên mạng chia sẻ chưa thể tự design mang bản sắc của dân tộc.

Phần 3. Kết luận

Đề tài "Phát triển game 3D" không là chủ đề mới hiện nay nhưng những kiến thức cụ thể về 3D thì lại còn khá lạ lẫm với những sinh viên CNTT nói chung, và là một trong những hướng phát triển đem lại nhiều thành quả trong lĩnh vực công nghệ thông tin khi công nghệ Thực tế ảo và Thực tế ảo tăng cường đang ngày càng phát triển. Sau khi kết thúc khóa luận lần này em nhận thấy mình đã đạt được một số kết quả như sau:

1, Sinh viên hiểu biết thêm về Game engine, các kiến thức chuyên sâu về xử lý hình ảnh trong game và dây chuyền sản xuất game.

2, Tự tay xây dựng từ A-Z phát triển một game 3D với engine Ogre3D.

3, Nâng cao khả năng tư duy, lập trình hướng đối tượng cho sinh viên.

Ngoài ra, cũng có những điều chưa kịp làm được như tìm hiểu về AI trong game và hoàn thiện sản phẩm game LegendOfThachSanh. Trong tương lai, sinh viên dự định sẽ tiếp tục theo đuổi ngành làm game với vị trí lập trình viên.

TÀI LIỆU THAM KHẢO

Sách:

1. Gregory Junker, Pro_OGRE_3D_Programming, 1nd ed., Apress, 2006.
2. Deitel & Associates, Game Programing with OGRE, 1nd ed, Deitel & Associates, 2008

Tài liệu tham khảo từ Internet :

3. Các bài giảng trên trang chủ Ogre3D và forum: http://www.ogre3d.org/, last visited January, 2017.
4. Game engine – Wikipedia tiếng Việt, last visited January, 2017.
5. Góc nhìn trong game và những điều bạn nên biết: http://infogame.vn/game-offline/goc-nhin-trong-game-va-nhung-dieu-ban-nen-biet-12959.html, last visited January, 2017.