TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

LÊ VŨ THÁI BẢO – NGUYỄN KHẮC HUY

NGHIÊN CỨU VÀ PHÁT TRIỂN

ỨNG DỤNG GAME

VỚI CÔNG NGHỆ XNA

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT

TP.HCM, 2009

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

LÊ VŨ THÁI BẢO 0512072

NGUYỄN KHẮC HUY 0512132

NGHIÊN CỨU VÀ PHÁT TRIỂN

ỨNG DỤNG GAME

VỚI CÔNG NGHỆ XNA

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN TIN HỌC

GIÁO VIÊN HƯỚNG DẪN

T.S TRẦN MINH TRIẾT

NIÊN KHÓA 2005 – 2009

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Khóa luận đáp ứng yêu cầu của LV cử nhân tin học.

TpHCM, ngày …… tháng …… năm 2009

Giáo viên hướng dẫn

NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

Khóa luận đáp ứng yêu cầu của LV cử nhân tin học.

TpHCM, ngày …… tháng …… năm 2009

Giáo viên phản biện

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn Khoa Công Nghệ Thông Tin, trường Đại Học Khoa Học Tự Nhiên, Tp.HCM đã tạo điều kiện tốt cho chúng em thực hiện đề tài này.

Chúng em xin chân thành cảm ơn Thầy Trần Minh Triết, là người đã tận tình hướng dẫn, chỉ bảo chúng em trong suốt thời gian thực hiện đề tài.

Chúng em cũng xin gửi lời cảm ơn sâu sắc đến quý Thầy Cô trong Khoa đã tận tình giảng dạy, trang bị cho chúng em những kiến thức quí báu trong những năm học vừa qua.

Chúng em xin gửi lòng biết ơn sâu sắc đến ba, mẹ, các anh chị và bạn bè đã ủng hộ, giúp đỡ và động viên chúng em trong những lúc khó khăn cũng như trong suốt thời gian học tập và nghiên cứu.

Mặc dù chúng em đã cố gắng hoàn thành luận văn trong phạm vi và khả năng cho phép, nhưng chắc chắn sẽ không tránh khỏi những thiếu sót, kính mong sự cảm thông và tận tình chỉ bảo của quý Thầy Cô và các bạn.

Nhóm thực hiện

Lê Vũ Thái Bảo & Nguyễn Khắc Huy



**Mở đầu**

*✍ Nội dung chương này giới thiệu tổng quan về nhu cầu và xu hướng phát triển game và Productline game dựa trên các Game framework và Game engine. Giới thiệu về công nghệ XNA của Microsoft, đồng thời nêu lên mục tiêu và nội dung của đề tài.*

## Tổng quan

Sức mạnh cốt lõi của công nghệ giải trí Game chính là do cỗ máy “game engine” và Game Framework cung cấp.

Game Framework là một tập hợp thư viện chứa các hàm và đối tượng hỗ trợ, cho phép vẽ, xử lý hoạt cảnh, xây hiệu ứng âm thanh từ các tài nguyên có sẵn, được xây dựng với mục đích có tính tái sử dụng cao nhất có thế, từ đó các Game engine được xây dựng. Game engine sẽ tạo ra các đối tượng Game, các logic game cho một thể loại game cụ thể. Một số Game framework.

* + - **Microsoft XNA Framework**
    - **Direct X**

Game engine có thể được hiểu như một middleware (phần mềm lớp giữa). Nói theo cách khác, Game Engine là phần mềm kiểm soát mọi hoạt động, tương tác của một trò chơi, từ kỹ thuật đến cách chơi, logic game.Về cơ bản, đó là một thực thể bao gồm những công việc liên quan đến phát triển game như vẽ, mô phỏng vật lý, các nguồn tương tác với game để những nhà phát triển tập trung vào việc phát triển chi tiết nội dung của game. Game engine cung cấp những thành phần cho phép những nhà phát triển có thể tương tác nhằm tạo nên game như giao diện, animation, sự tương tác giữa các đối tượng, vật lý và ngay cả trí thông minh nhân tạo. Mỗi một nhà cung cấp giải pháp game engine có cách tiếp cận riêng và các nhà phát triển hiện có khá nhiều engine thương mại để lựa chọn.

* **BIGWORLD** - Là một trong những engine tươm tất, được thiết kế đặc biệt để phục vụ thị trường game trực tuyến nhiều người chơi.
* **CRYENGINE** - được biết đến với tựa game bắn súng CRYSIS năm 2007.
* **UNREAL ENGINE** của EPIC - dành cho các hãng phát triển game toàn cầu.
* **QUBESOFT** - Q ENGINE là được phát triển bởi hai cựu nhân viên của Microsoft sáng lập ra DirectX là Servan Keondjian và Doug Rabson.
* **SIMUTRONICS** - được xây dựng dựa trên kinh nghiệm từ thời game text.

Dựa trên các tiền đề này, các nhà sản xuất đã xây dựng ra các Product line nổi trội trong làn giải trí Game, với nội dung phong phú như dòng Game nhập vai Diablo, Prince Of Persia…, dòng Game dàn trận Age Of Wonder, Age Of Empire…với máy PC và dòng Game nhập vai Final Fantasy, Kingdom Heart .. với hệ máy Nintendo, Play Station… Các phiên bản sản phẩm trong Product line này được các nhà sản xuất xây dựng từng bước, từ giao diện 2D thô sơ ban đầu được nâng cấp lên giao diện đồ họa 3D, với các hiệu ứng particle ngày càng đặc sắc, cùng với hiệu ứng âm thanh hoành tráng làm cho Product line của họ ngày càng được ưa chuộng. Nhưng có một điều mà các Product line Game này không thay đổi hoặc thay đổi rất ít qua mỗi phiên bản của chúng là nội dung chính của trò chơi, cốt lõi của các Product line, việc thay đổi chủ yếu được thực hiện trong mỗi phiên bản ra đời của một Product line chính là cách thức thể hiện nội dung đó là ra bên ngoài giao diện, thiệt bị output và tương tác với người chơi sẽ khác nhau hoặc có thể, tuy sẽ có một số cải tiến nào đó như số lượng pháp thuật, số lượng vũ khí, số lượng nhân vật, cách thức giao tiếp giữa các nhân vật, hoạt động của nhân vật trong Game nhưng chung quy lại, mọi hoạt động chính vẫn tuân theo một nội dung nhất định. Đó là con đường phát triển chủ yếu đã làm nên các Product line Game nổi tiếng trong làng giải trí Game, gắn liền với các hãng sản xuất hàng đầu như Square Soft hiện nay được biết với cái tên Square Enix, Ubisoft,… Không nằm ngoài con đường đó, đề tài này xây dựng những bước ban đầu cho một Product line của thể loại Game dàn trận dựa trên nền Microsoft XNA Framework.

Mục tiêu của đề tài bao gồm các nội dung sau:

* + - Tìm hiểu Microsoft XNA framework.
    - Xây dựng một Product line thể loại game dàn trận dựa trên Microsoft XNA framework.
    - Xây dựng Game editor.

## Giới thiệu XNA

XNA framework được thiệt kế như một tập các lớp thư viện dùng cho công nghệ phát triển game, nhằm phát huy cao nhất tính tài sử dụng của mã trên nhiều nền tản. Nó cũng bao gồm một phiên bản của Common Language Runtime, được tối ưu hóa cho yêu cầu lập trình game, nhằm cung cấp một môi trường thực thi có quản lý cho các game XNA. Các Common Language Runtime này đều có sẵn trong môi trường Window XP, Window Vista, Zune và Xbox. Từ đó Game XNA có thể chạy trên bất kì nền tản nào có hổ trợ XNA Framework, với những sửa đổi tối thiểu hoặc thậm chí không cần sửa đổi. Như vậy, XNA framework gói gọn những chi tiết kĩ thuật ở mức độ thấp trong mã lập trình của một game và đảm bảo rằng chính bản thân framework sẽ hỗ trợ các nền tản khác nhau khi các game được tạo ra từ XNA framework được mang đi thực thi ở một nền tản tương đương. Và cũng do đó, đã cho phép các nhà phát triển game tập trung hơn vào nội dung và kinh nghiệm làm game mà không cần phải bận tâm về nền tản để thực thi game. Ngoài ra, XNA framework còn tích hợp thêm một số công cụ, như công cụ tạo hiểu ứng âm thanh XACT, nhằm mục đích hổ trợ khả năng sáng tạo nội dung ngày càng tốt hơn. Những công cụ này sẽ giúp cho tác giả hình thành những ý tưởng mới về nội dung hình ảnh, âm thanh, sự kết hợp giữa chúng và xây dựng các hình tượng nhân vật thật năng động. XNA framework còn cung cấp các giải pháp, kỹ thuật giúp cho các nhà phát triển game độc lập và chuyên nghiệp, tạo ra các sản phẩm game chạy trên nền tản Microsoft.



Hiện nay XNA framework đã được nâng cấp đến phiên bản 3.0. So với phiên bản 2.0, XNA framework 3.0 là sự mở rộng lớn cung cấp những tính năng dễ sử dụng, tiến đến gần mục tiêu sản xuất ra các dạng game chạy trên nhiều nền tản, có khả năng chia sẽ với thế giới game trong Xbox360, Window và bây giờ là trên Zune.

XNA Game Studio là bộ plug-in Visual Studio, với các tính năng triển khai được sử dụng với XNA framework. XNA Game Studio là một công cụ được sử dụng để xây dựng một game trong XNA Framework với Visual Studio và cung cấp khả năng chuyển đổi game đã được xây dựng sang nền tản Xbox hoặc Zune thay vì trên hệ điều hành Window.

## Mục tiêu đề tài

Đề tài này thuộc về hướng nghiên cứu và tìm hiểu công nghệ, từ đó xây dựng và phát triển ứng dụng. Mục tiêu chính mà để tài hướng vào là tìm hiểu công nghệ XNA và xây dựng Productline game dựa trên công nghệ XNA. Đề tài bao gồm các các phần sau:

* Nghiên cứu và tìm hiểu Microsoft XNA Framework 3.0.
* Mô tả nội dung của Productline Game sẽ xây dựng.
* Xác định và phân tích các vấn đề khi xây dựng và phát triển game với Productline này trên nền tản Microsoft XNA framework.
* Đề ra các giải pháp kỹ thuật và công nghệ.
* Trên các cơ sở tìm hiểu và phân tích, nhóm sẽ xây dựng Productline game cho dòng game dàn trận trên nền tản XNA Framework 3.0 với nội dung thể hiện dưới dạng 2D. Như vậy, nội dung đề tài không phải xây dựng một ứng dụng game cụ thể mà xây dựng một Productline game với thể loại dàn trận trên nền tản XNA Framework 3.0

## Nội dung của luận văn

**Chương 1:** Mở đầu

Nội dung chương này giới thiệu tổng quan về nhu cầu và xu hướng phát triển game và Productline game dựa trên các Game framework và Game engine. Giới thiệu về công nghệ XNA của Microsoft, đồng thời nêu lên mục tiêu và nội dung của đề tài.

**Chương 2:** XNA framework và XNA game Studio

Nội dung chương này giới thiệu và XNA framework và XNA game studio,cách thức thực thi game và thể hiện nội dung game bằng hình ảnh, input và âm thanh.

**Chương 3:** Danh sách và vấn đề và giải pháp thực hiện.

Nội dung chương này đề cập và phân tích các vấn đề khi xây dựng Productline Game thể loại dàn trận. Với mới vấn đề, các giải pháp sẽ được đưa ra để phân tích và áp dụng.

**Chương 4:** Mô tả nội dung Productline game dàn trận

Nội dung chương này sẽ mô tả nội dung của Productline game sẽ được xây dựng trong đề tài, với thể loại game được lựa chọn là game dàn trận bao gồm các thành phần trong game, logic thể hiện trong game giữa các thành phần.

**XNA framework và XNA Game Studio**

*✍ Nội dung chương này giới thiệu và XNA framework và XNA game studio,cách thức thực thi game và thể hiện nội dung game bằng hình ảnh, input và âm thanh trong công nghệ XNA.*

## Giới thiệu:

XNA framework là một tập các lớp thư viện hỗ trợ cho việc phát triển game trên các nền tản Microsoft như Window, Xbox 360, Zune.

XNA Game studio là một bộ công cụ dựa trên các phiên bản hỗ trợ của Microsoft Visual Studio , cho phép các nhà phát triển game tạo các ứng dụng dựa trên công nghệ XNA triển khai trên các môi trường thuộc nền tản Microsoft như Microsoft Window, Xbox 360 và bây giờ có cả Zune.

## Kiến trúc XNA và các trạng thái xử lý trong game:

Bước đầu tiên trong việc xây dựng một game là tạo ta một lớp kế thừa từ Microsoft.Xna.Framework.Game , lớp này phải khai báo override các phương thức **Update**, **Draw, LoadContent** và **Initialize**.Phương thức **Update** có trách nhiệm xử lý logic của game và phương thức **Draw** có trách nhiệm vẽ các thành phần của game ra màn hình. Phương thức **Initialize** cùng với hàm contructor có trách nhiệm khởi tạo các giá trị ban đầu cho các thành phần đồ họa, âm thanh và input trong game. Phương thức **LoadContent** load tài nguyên đồ họa bào gồm các hình ảnh sẽ vẽ trong game và **UnloadContent** có trách nhiệm giải phóng tài nguyên đồ họa.

public class Game1 : Microsoft.Xna.Framework.Game

{

public Game1()

protected override void Initialize()

protected override void LoadContent()

protected override void UnloadContent()

protected override void Update(GameTime gameTime)

protected override void Draw(GameTime gameTime)

}

Quá trình thực thi các trạng thái xử lý của một game đơn giản có thể được trình bày như sau :

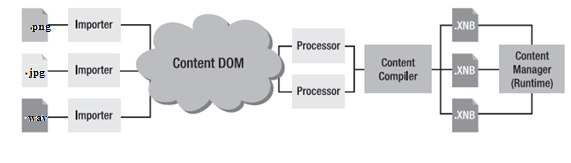
* Khởi tạo tài nguyên xử lý đồ họa, input, âm thanh
* Load các tài nguyên hình ảnh, âm thanh …
* Bắt đầu thực thi vòng lặp của game :
  + Lấy input từ người dùng
  + Thực thi các tính toán như AI, xử lý va chạm …
  + Kiểm tra yêu cầu kết thúc game(người dùng muốn kết thúc game) 🡪 kết thúc vòng lặp game
  + Vẽ hình ảnh, phát sinh âm thanh và trả lời các điều khiển input.
* Giải phóng các tài nguyên

Trong game XNA, Update và Draw được thực hiện trong vòng lặp của game để thực hiện các thao tác hình ảnh và logic game trong suốt quá trình game được thực thi, Initialize và constructor cùng với LoadContent sẽ khởi tạo các giá trị đồ họa, âm thanh … cho game, Unload sẽ giải phóng các tài nguyên khi quá trình thực thi game kết thúc. Quá trình thực thi này có thể được ánh xạ vào solution của XNA game như sau:

* Game1() –Khởi tạo các thành phần chung(Game1.cs)
  + Initialize() – Khởi tạo game (Game1.cs)
  + LoadContent() – Load các tài nguyên hình ảnh đồ họa (Game1.cs)
  + Run() – chạy vòng lặp của game (Program.cs):
* Update() – Đọc các Input từ người dùng, thực hiện tính toàn cho logic game, kiểm tra điều kiện ngừng thực thi game(Game1.cs)
* Draw() – Thực hiện các thao tác vẽ ra màn hình nội dung cùa game thông qua các tài nguyên đã load và thực thi xử lý(Game1.cs)
  + UnloadContent() – Free graphics resources (Game1.cs)

## Content Pipeline

XNA Game Studio Content Pipeline là một tính năng quan trọng do XNA cung cấp bởi vì nó đơn giản hóa việc người phát triển game load các tài nguyên hình ảnh, âm thanh từ các định dạng khác nhau một cách dễ dàng. Do đó, người lập trình không phải bận tâm đến việc làm cách nào để load các hình ảnh, âm thanh và làm cách nào để đọc nội dung của chúng. Tử các định dạng khác nhau, thông qua quá trình xử lý với XNA Game Studio Content PipeLine sẽ được chuyển thành định dạng dạng chung cho tất cả là .xnb, nội dung tập tin này sẽ được loacd và đọc ra trong game XNA nhờ Content manager.



## Input control

Input là thuật ngữ chung để nói đến quá trình nhận các thao tác từ người dùng. Trong Microsoft XNA framework, namspace [Microsoft.Xna.Framework.Input](http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.input.aspx) đã cho phép hỗ trợ các thiết bị input khác nhau bao gôm gamepad(Xbox360), gamestick, mouse và keyboard.

* Gamepad(Xbox360) cung cấp một sự kết hợp tốt giữa các button digital và các stick analog, vì vậy có nhiều loại game có thể được chơi với nó. Các gamepad này có thể được sử dụng trên hệ thống Window hoặc trên máy Xbox 360, nhưng chỉ tối đa 4 cái được sử dụng.

Cách sử dụng:

Lấy trạng thái của gamepad bằng phương thức GetState của lớp GamePad. Nhận lấy trạng thái từ GamePadState và đưa ra cac thông tin xử lý tương ứng trong phương thức Update()

* Keyboard chỉ bao gồm các button digital và không có bất kì điều khiển analog nào cả. Nhưng không phải keyboard nào cũng hỗ trợ đầu đủ các phím được liệt kê trong lớp Keys, ví dụ như những bàn phím cổ điển không có các phím tăng giảm âm lượng.Thông thường, keyboard được dùng với như một thiết bị Input với máy tính, nhưng nếu một keyboard được sử dụng với cổng USB thì có thể sử dụng được với cả Window và Xbox 360

Cách sử dụng:

Tương tự như dùng gamepad, Keypad sẽ được lấy trạng thái bằng GetState() của lớp KeyBoard và nhận thông tin trả về từ KeyboardState để thực thi các yêu cầu xử lý từ người dùng.

* Mouse là sự lựa chọn khá tốt cho nhiều đối tượng thể hiện trong game, giả sử trong một game có nội dung thuộc dòng game dàn trận, các thao tác chuột nhằm xử lý việc scroll vùng viewport qua khắp bản đồ hay việc điều khiển một số lượng nhân vật lớn trong game đều cần thiết phải thực hiện qua mouse. Cũng như Keyboard, không phải tất cả các mouse đều hỗ trợ đủ các button.
  + Cách sử dụng:
    - Dùng GetState để lấy trạng thái mouse từ lớp Mouse, thông qua cac thông tin của MouseState để trả lời bằng các tính toán với người dùng.

Như vậy XNA hỗ trợ hầu hết các thiết bị input thông dụng, nhưng việc xử lý các thiết bị input đề dựa và các trang thái của thiết bị. Như vậy, khi sử dụng XNA trên nền Window, bất kì hoạt động nào của mouse và key board thực hiện đều bị ghi lại trạng thái và game cũng thực thi luôn trạng thái đó dù cửa sổ game không được forcus.

## XNA audio system

Trong XNA framework 2.0, các file âm thanh phải được add vào Content dưới định dạng đặc biệt, được tạo ra từ Microsoft Cross-PlatformAudio Creation Tool, được biết với cái tên XACT, một tool đi kèm khi cài đặt XNA framework. XACT giúp tạo ra Sound bank và Wave bank từ nhiều file wav, AIFF, XMA và biên dịch thành file XAP. File XAP này chứa các âm thanh đã đưa vào bằng XACT. File XAP sẽ được load từ Content và được phát thông qua các đối tượng AudioEngine, WaveBank , SoundBank.

Nhưng trong XNA framework 3.0 đã cung cấp lớp SoundEffect hoặc (SoundEffectInstance). Việc sử dụng hai lớp này đơn giãn hơn dùng XACT, và XACT không hoạt động trên Zune, nên giải pháp của phiên bản 3.0 sẽ là một tùy chọn tốt cho các game trên Zune.

**Danh sách các vấn đề và giải pháp thực hiện**

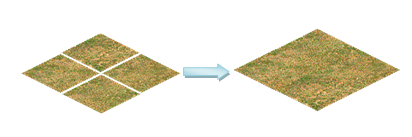
*✍ Nội dung của chương này đề cập và phân tích các vấn đề khi xây dựng Productline Game thể loại dàn trận. Với mới vấn đề, các giải pháp sẽ được đưa ra để phân tích và áp dụng.*

## Xây dựng bản đồ vùng chiến sự.

* Vấn đề :

Bản đồ vùng chiến sự là nền để sắp đặt và bố trí các Sprite lên đó, bao gồm Terrain, Resource center, Unit và Structrure. Bản đồ có kích thước lớn hơn nhiều so với vùng nhìn của cửa sổ (Viewport) cho phép người dùng kéo vùng nhìn của cửa sổ trên khắp bản đồ để quan sát cục diện trận chiến trong Game, đồng thời bản đồ cần phải có tính linh động cao trong việc thay đổi kích thước. **Vậy, vấn đề ở đây là xây dựng một bản đồ chiến sự lớn, có khả năng tùy biến về kích thước**.Có hai giải pháp được nêu ra trong việc xây dựng bản đồ vùng chiến sự.

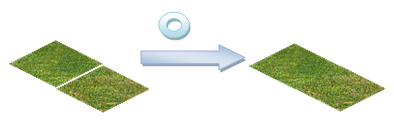
* Giải pháp:
  + Giải pháp thứ nhất được đề ra : Load một ảnh thật lớn và vẽ ra như một bản đồ chiến sự trên màn hình Game. Nhưng khuyết điểm của giải pháp này là XNA framework chỉ cho phép load hình ảnh có kích thước tối đa là **2024 x 2024 px**, như vậy nếu load một ảnh thật lớn cũng chỉ có thể load được kích thước 2024px cho mỗi cạnh, điều này đã làm giảm tính tùy biến về kích thước của bản đồ chiến sự. Tuy vậy nếu chấp nhận giải pháp này có thể làm giảm ảnh hưởng của khuyết điểm trên bằng cách vẽ hình đã load với kích thước khác nhau, tuy nhiên cách này sẽ làm giảm chất lượng của ảnh được vẽ, thử tưởng tượng ta có một ảnh kích thước 128 x 128px thử resize nó thành 256 x 256 px, hiển nhiên hình ảnh sẽ không còn đẹp nữa. Cần thiết phải có một giải pháp tối ưu hơn.
  + Giải pháp thứ hai được nêu ra ở đây, thử xem bản đồ vùng chiến sự là một tập hợp nhiều ảnh nền nhỏ ghép lại(được gọi là các **Cell**), kích thước bản đồ tùy thuộc vào số lượng các mảnh ghép vì thế, khi tăng số lượng các mảnh ghép, ta được một bản đồ có kích thước to hơn. Việc vẽ bản đồ được thực hiện bằng phép vẽ nối tiếp các mãnh ghép trên Viewport.



Các bước thực hiện cho giải pháp

* + - Xác định tính đúng đắn của các mãnh ghép(sự chuyển tiếp giữa các vùng đất trên bản đồ).





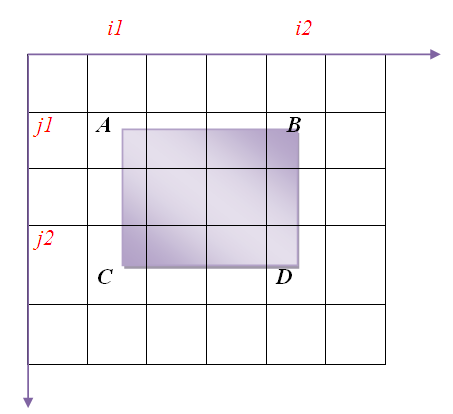
* + - Từ đó xây dựng được ma trận các mảnh ghép.
    - Xác định vị trí của mỗi mãnh ghép bản đồ, sao cho khi vẽ ra, các mãnh ghép nối tiếp nhau.
    - Vẽ mảnh ghép: thực hiện thao tác vẽ chỉ cho các mãnh ghép nằm trong vùng Viewport và không thực hiện vẽ các mãnh ghép nằm ngoài cùng Viewport vì điều này là không cần thiết.

1. Xác định cách vẽ bản đồ vùng chiến sự với ma trận Cell, sao cho chỉ giới hạn vẽ trong vùng Viewport nhằm làm giảm chi phí.

* Vấn đề:

Giả sử đã có ma trận các Cell với tính đúng đắn và vị trí vẽ thích hợp vào bản đồ, bây giờ phải vẽ các Cell. Ta không thể lặp toàn bộ ma trận và vẽ ra tất cả các Cell trong ma trận, điều này là không cần thiết, vì các Cell không nằm trong Viewport không cần phải được vẽ ra. Không những thế, chi phí để thực hiện vòng lặp trên ma trận Cell sẽ rất lớn, vì nếu bản đồ chiến sự có kích thước rất lớn thì đồng nghĩa với việc số lượng Cell cần lặp và vẽ trên bản đồ chiến sự là rất nhiều. **Như vậy, vấn đề ở đây là làm sao giới hạn lại những Cell nằm trong Viewport để vẽ ra hay nói cách khác là tìm ra index của các Cell trong ma trận sao cho chúng đảm bảo Cell nằm trong Viewport.**

* Giảipháp:
  + Để giải quyết vấn đề trên ta chia thành hai trường hợp
    - Với bản đồ có các Cell dạng hình chữ nhật hoặc hình vuông

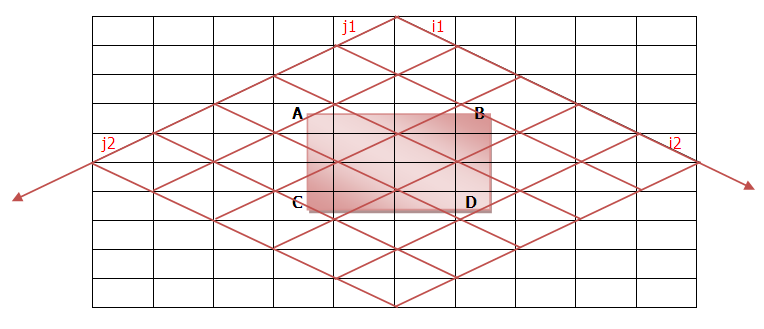
**

Giả sử có vùng bản đồ chiến sự như trên, bao gồm nhiều Cell hình vuông ghép lại. Viewport là hình chữ ABCD.

i,j là index của ma trận các Cell

Xác định lần lượt các đỉnh A,B,C,D nằm ở Cell nào trong ma trận. A(i1,j1) B(i2,j1) C(i1,j2) D(i2,j2) như vậy các Cell cần vẽ có **i1<=i<= i2, j1<=j<=j2** (việc xác định D là không cần thiết, với A,B,C đã đủ xác định được i1,i2,j1,j2)

* + - Với bản đồ có các Cell dạng hình thoi



Bản đồ là một tập các Cell dạng hình thoi ghép lại.

Thực hiện tương tự như dạng bản đồ hình vuông nhưng cách xác định i và j khác do cách biểu diễn hệ trục là khác

Tóm lại giải pháp chủ yếu dựa trên việc xác định index giới hạn trong ma trận Cell để chọn ra các Cell nằm trong Viewport và vẽ.

1. Thể hiện bản đồ vào vùng Viewport

* Vấn đề:

Sau khi xác định được các Cell cần vẽ, việc tiếp theo phảo làm là thể hiện chúng vào Viewport.

* Giải pháp.:
  + Giải pháp được thực hiện
    - Lớp Map được cài đặt là một lớp kế thừa từ lớp DrawableGameComponents của XNA framework, DrawableGameComponents là lớp tương tự lớp GameComponents nhưng sẽ thực hiện chức năng vẽ vào màn hình bằng cách override Draw(). Lớp này chứa ma trận các Cell.
    - Cài đặt hàm Draw() của Map chỉ vẽ các Cell được giới hạn bởi index đã xác định trước.
    - Add đối tượng Map vào Game.Components và vẽ ra các Cell trong giới hạn index đã xác định trước.
    - Tùy vào loại Map mà thiết kế các lớp kế thừa cụ thể.

1. Thể hiện các thành phần của Game và vẽ hình ảnh các thành phần này.

* Vấn đề:

Các thành phần trong Game bao gồm Unit(quân lính), Structure(công trình), Terrain(địa vật), Resource Center(mỏ tài nguyên)... Sau khi đã vẽ từ ma trận Cell ra màn hình một khung cảnh nền cho bản đồ chiến sự, các thành này sẽ được vẽ lên màn hình để thể hiện trực quan cho người chơi cục diện chiến sự.

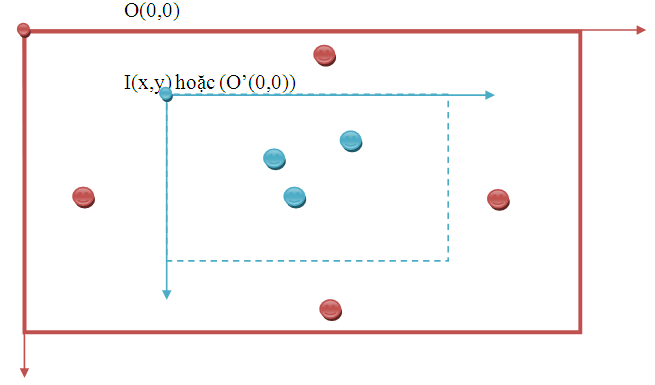
* Giải pháp:
  + Giải pháp như sau:
    - Đầu tiên, cài đặt lớp Sprite kế thừa từ lớp DrawableGameComponents của XNA framework, DrawableGameComponents là lớp tương tự lớp GameComponents nhưng sẽ thực hiện chức năng vẽ vào màn hình bằng cách override Draw(). Các thành phần của Game đều kế thừa từ lớp Sprite và tùy vào từng thành phần cụ thể, sẽ được cài đặt thành các lớp Unit, Structure … nhưng chung quy lại, hàm Draw() của các thành phần này đều override trên hàm Draw() của Sprite. Để thực hiện vẽ, các thành phần được add vào thuộc tính Components của đối tượng Game dưới dạng Sprite và vẽ ra theo thứ tự trước sau như khi add vào.

+Sprite(kế thừa DrawableGameComponents): cài đặt lại về Draw()

+Unit,Structure… : kế thừa từ Sprite

+Add Unit,Structure… vào Components của lớp Game dưới dạng Sprite

* + - Vấn đề đặt ra ở đây là làm sao vẽ các thành phần này tối ưu nhất(nghĩa là không tốn nhiều tài nguyên phần cứng quá nhiều). Cụ thể, khi được vẽ ra màn hình, hiển nhiên các thành phần đã nằm trong Components của đối tượng Game sẽ được vẽ hết ra ngoài thông qua cài đặt hàm override Draw() của lớp Sprite. Nhưng vấn đề nãy sinh là đâu cần phải vẽ hết các thành phần nếu chúng không cần thiết phải thể hiện, đó là các thành phần không nằm trong Viewport sẽ không cần thiết phải vẽ. Như thế trong cài đặt hàm Draw() của Sprite, cần thiết phải kiểm tra Sprite có nằm trong Viewport hay không.
    - Để thực hiện điều này, thực hiện như sau, xem toàn bộ bản đồ là một hình chữ nhật lớn dù là loại hình thoi hoặc hình vuông. Gốc tọa độ nằm ở gốc trái trên của bản đồ, trục tọa độ là cạnh trên (Ox) và cạnh trái (Oy), ta được một hệ trục tọa độ toàn cục. Vị trí của góc trái trên Viewport chỉ là một điểm nằm trên hệ trục đó, nhưng Game component trong XNA frame work lại xem hệ trục tọa độ khi vẽ luôn có gốc là góc trái trên Viewport, trục tọa độ là hai cạnh trên và trái, như vậy Viewport là một hệ trục tọa độ nhỏ trong hệ trục tọa độ toàn cục. Lớp Sprite được cài đặt thuộc tính Position, thể hiện vị trí của đối tượng Sprite trong hệ trục tọa độ toàn cục, để kiểm tra có vẽ đối tượng Sprite nào đó ra màn hình Viewport không, ta chỉ cầsn xét trong hệ tọa độ Viewport, Position của Sprite có nằm trong vùng Viewport hay không.



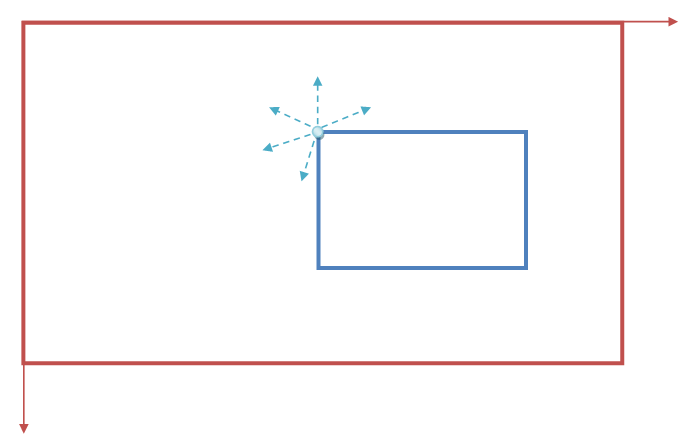
* + - Hiển nhiên thuộc tính Position phải đảm bảo cho Sprite không được nằm ngoài vùng lát nền của bản đồ dạng hình thoi.

1. Scroll Viewport trên bản đồ chiến sự để thể hiện trạng thái của các vùng khác nhau trên bản đồ:

* Vấn đề:

Viewport là màn hình của Game, chỉ thể hiện một phần nhỏ cũng một bản đồ rộng lớn. Người chơi muốn thao tác trên bản đồ cần phải chuyển vùng màn hình này trên khắp bản đồ bằng các thiết bị Input. **Vấn đề là làm cách nào dùng các thiết bị Input để thay đổi vị trí của Viewport trên bản đồ**.

* Giải pháp:
  + Giải pháp đưa ra dựa trên các giải pháp đã dùng trong việc hiển thị bản đồ và hiển thị các thành phần Game trong Viewport. Bằng cách dựa vào tọa độ của góc trái trên Viewport trong hệ tọa độ toàn cục bản đồ vùng chiến sự, các cell của bản đồ được giới hạn index lại và vẽ ra Viewport, đồng thời, với cách sử dụng tọa độ này, các thành phần Game nằm trong vùng Viewport cũng được vẽ ra màn hình Viewport. Như vậy chỉ cần thay đổi tọa độ của góc trái trên của Viewport trong hệ tọa độ toàn cục của toàn bản đồ bằng các thiết bị Input là có thể thực hiện Scroll Viewport trên khắp bản đồ.



Tọa độ của góc trái trên vùng Viewport được thay đổi bằng thiết bị Input là keyboardvà mouse.

* Với keyboard, sử dụng các button để thay đổi tọa độ bằng cách cộng trừ tọa độ theo pixel.
* Với mouse, xác định vị trí trỏ chuột dựa vào vùng Viewport để cộng trừ tọa độ góc trái trên Viewport. Cụ thể, XNA framework cung cấp khả năng xác định trạng thái của mouse, dựa vào trạng thái này, xác định vị trí của con trỏ mouse so với Viewport, nếu mouse nằm ngoài Viewport thì cộng trừ tọa độ góc trái trên Viewport để có hướng Scroll Viewport thích hơn trên toàn bộ vùng bản đồ.

1. Thể hiện trạng thái hoạt động của các Sprite trong Game.

* Vấn đề:

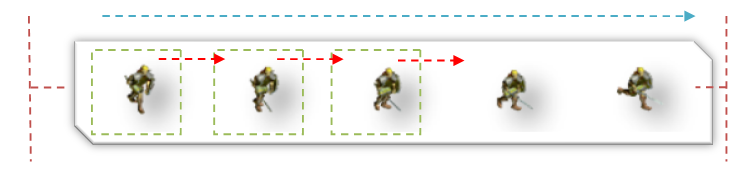
Mỗi Sprite trong Game có một tập những trạng thái hoạt động nhất định bao gồm những hoạt động cơ bản sau : IDLE, WALK, ATTACK, DIE… được gọi tắt là trạng thái. Mỗi trạng thái, cùng với hướng hoạt động của trạng thái đó, bao gồm một loạt các hình ảnh chuyển đổi liên tiếp, mô phỏng từng bước hoạt động của trạng thái đó. Hình ảnh sau là 15 ảnh mô tả một Sprite cụ thể là một quân lính đang di chuyển(WALK)



**Như vậy, để một tả một hoạt động, cần phải liên tục chuyển đổi hình ảnh theo thứ tự cho trước theo một khoảng thời gian trì hoãn giữa mỗi lần chuyển đổi**.

* Giảipháp:
  + Giải pháp thứ nhất là lưu tập hình trên thành một Frame hình(một ảnh có chiều rộng lớn và trên đó là các hình mô tả chuyển động đã xếp thứ tự). Bao gồm 2 cách sau:
    - Tất cả hình của tất cả các trạng thái nằm trong một frame hình.
    - Mỗi trạng thái có một frame hình.

Sự chuyển đổi liên tiếp được thực hiện bằng cách liên tục khoanh vùng hình ảnh trên frame và vẽ ra màn hình. Cách làm như thế có ưu điểm là chỉ cần load Frame một lần, rồi chuyển khoanh vùng liên tục trên Frame hình và vẽ ra, nhưng khuyết điểm của nó là phải xây dựng được Frame hình phù hợp, đồng thời, xác định khoanh vùng phải chính xác và đủ nhanh, mặt khác, XNA chỉ có thể load hình ảnh với size 2024x2024 px, vì thế, nếu số lượng hình ảnh trên một Frame là quá nhiều có lẽ sẽ làm chiều rộng của Frame vượt ngưỡng.



* + Giải pháp thứ 2 được sử dụng là lưu các hình ảnh một cách rời rạc, và mô tả đường dẫn đến mỗi hình trong file XML sao cho mỗi tập hình cho một trạng thái được lưu cùng chỗ. Từ file XML, mỗi lẫn muốn thể hiện trạng thái của Sprite, load toàn bộ tập hình vào mảng và việc chuyển đổi sẽ thực hiện bằng cách chuyển index của mảng. Khuyết điểm của giải pháp này là thời gian xử lý hơi chậm nếu tập hình cho một trạng thái quá nhiều, nhưng ưu điểm của nó là cho phép chuyển đổi hình nhanh bằng index sau khi đã load đủ tập hình vào mảng vì không phải thực hiện tính toán khoanh vùng như Frame, mặt khác sẽ giảm thiểu rủi ro phải load một ảnh vượt quá ngưỡng kích thước cho phép.

Cả hai giải pháp trên đều thực hiện chuyển đổi hình ảnh theo cách thức khác nhau nhưng đều có điểm chung là cách làm này dựa trên cách thức làm phim hoạt hình, sau một khoảng thời gian ngắn, vài mili second, mới thực hiện chuyển hình một lần để thể hiện hoạt động. Khoảng thời gian ngắn đó là delay time để trì hoản giữa hai lần chuyển hình kế tiếp.

1. Quản lý Structure và Unit trên sao cho truy cập thật nhanh.

* Vấn đề:

Các Sprite trên vùng chiến sự có số lượng rất nhiều, và chia thành nhiều loại bao gồm quân lính(Unit), công trình(Structure)… mặt khác, với các Sprite là Unit, Structure, chúng thuộc sở hữu của một Player cụ thể. Vấn đề đặt ra là làm cách nào để quản lý và truy xuất các Sprite theo từng loại nhanh và dễ dàng.

* Giải pháp
  + Giải pháp thứ nhất : duyệt trong sách Components của đối tượng Game mà XNA cung cấp và kiểm tra xem Components lấy được là Sprite cụ thể loại gì, thuộc sở hữu của Player nào... Cách này hoàn toàn không khả thi với một danh sách Components quá nhiều phần tử.
  + Giải pháp thứ hai là
    - Trên toàn bộ Game, chia các Sprite thành các danh sách con mà các phần tử là một loại Sprite muốn quản lý
* Danh sách các Unit hiện có trong vùng chiến sự
* Mảng các Structure hiện có trên BattleField
* Thực hiện vòng lặp trên hai danh sách này để tìm Sprite cần thiết
  + - Đối với mỗi Player lưu hai danh sách bao gồm danh sách các unit và danh sách các Structure
    - Trong Structure, lưu danh sách các Unit do nó sinh ra.
    - Mỗi Unit có thuộc tính là lớp Structure cho biết nó do Structure nào sinh ra.
    - Mỗi Unit và Structure có một thuộc tính là lớp Player cho biết nó thuộc Player nào.

Nhờ vào các mối quan hệ này, việc truy cập thông tin các Sprite trên map sẽ thực hiện nhanh chóng hơn.

1. Với Unit có khả năng tấn công, nhiệm vụ phát hiện và tấn công đối phương là nhiệm vụ chủ yếu.Trong suốt quá trinh tồn tại, Unit sẽ chết nếu sinh lực đã cạn do bị tấn công.

* Vấn đề

Một Unit có khả năng tấn công sẽ tiêu diệt đối phương khi đối phương tiến lại gần hoặc do chính player chủ động điều khiển Unit đánh quân địch. Unit cần phải biết khi nào có kẻ địch ở gần và thu hẹp khoảng cách nếu Unit có ý định tấn công. Unit kiểm tra khoảng cách giữa Unit và địch có nằm trong phạm vi cho phép Unit ra đòn tấn công. Dùng sức mạnh của mình, Unit sẽ làm tiêu hao sinh lực và tiêu diệt địch. Trong qua trình tấn công, Unit cần xác định đối tượng mình tấn công có còn trong phạm vi tấn công hay không(đối phương có thể đã bỏ chạy) để có hành động phù hợp(không tấn công nữa, chuyển sang trang thái IDLE hoặc WALK để đuổi theo). Trong trường hợp đối phương đã hết sinh lực, Unit ngừng tấn công đối phương, nhưng trong trường Unit hết sinh lực, Unit sẽ chuyển sang trang thái DIE. **Vấn đề phải giải quyết là xử lý toàn bộ quá trình phát hiện, tấn công và chết của một Unit**.

* Giải pháp:
  + Giải pháp đưa ra như sau
    - Các Unit có khả năng tấn công có thuộc tính mô tả phạm vi tấn công và phạm vi phát hiện, được thể hiện bằng khoảng cách từ Unit tới đối phương và phạm vi tấn công lớn hơn phạm vi phát hiện, bất kì đối phương nào rơi vào phạm vi phát hiện sẽ bị Unit áp sát và rút ngắn khoảng cách, khi khoảng cách nằm trong phạm vi tấn công Unit sẽ ra đòn tấn công lên đối phương để tiêu hao sinh lực địch.
    - Unit cần phải biết đối tượng hiện tại bị nó tấn công để kiểm tra rằng
      * Đội tượng đang bị Unit tấn công có còn sinh lực không, nếu đã hết, Unit sẽ ngừng tấn công và chờ đợi một đối tượng tấn công khác
      * Nếu đối tượng bị tấn công nếu đã bỏ chạy ra ngoài phạm vi tấn công, Unit sẽ ngừng tấn công và đuổi theo, nếu đối tượng bị tấn công nằm ngoài phạm vi phát hiện, Unit dừng tấn công và đứng yên hoặc chờ lệnh từ Player.

Đối tượng bị tấn công này được lưu thành thuộc tính **WhoIHit** trong lớp Unit. Mọi thao tác kiểm tra như trên(kiểm tra sinh lực, kiểm tra còn nằm trong phạm vi), được thực hiện thông qua nó.

* + - Trong trường hợp Unit đã cạn kiệt sinh lực, DIE là trạng thái Unit sẽ phải nhận lấy và được dispose ra khỏi chương trình.
    - Tất cả các hoạt động của Unit đều được thể hiện qua trạng thái cùng với tập hình ảnh thể hiện thao tác đi kèm đã được nêu trong “**Thể hiện trạng thái hoạt động của các Sprite trong Game”**

1. Vấn đề đi lấy tài nguyên từ các mỏ bởi Unit khai thác tài nguyên:
   * Vấn đề:
     + Tài nguyên là thành phần quan trọng, từ đó, Player có thể xây dựng lực lượng, duy trì cuộc chơi. Trong suốt quá trình diễn ra trò chơi, việc khai thác tài nguyên là không thể thiếu, từ các tài nguyên khai thác được, Player thực hiện quá trình “quy đổi” thành các Structure và Unit. Tài nguyên sẽ được khai thác từ các mỏ tài nguyên tùy loại. **Player tham gia trận đánh cần phải khai thác tài nguyên là điều cần thiết.**
   * Giải pháp:
   * Giải pháp đề ra:

Các tài nguyên sẽ được cung cấp từ các mỏ tài nguyên. Ta dùng một danh sách lưu các mỏ tài nguyên có trong bản đồ chiến sự, và sử dụng đối tượng ProducerUnit để khai thác các mỏ bằng cách:

* Duyệt qua danh sách các mỏ tài nguyên có trên bản đồ trong quá trình đối tượng này di chuyển, khi đã phát hiện mỏ tài nguyên có thể khai thác, thực hiện các bước kiểm tra
  + Kiểm tra tình trạng khai thác của mỏ tài nguyên đã phát hiện được. Nếu mỏ đã không còn gì để khai thác sẽ không khai thác được gì.
  + Kiểm tra tài nguyên mang trên người cùng loại hay khác loại với tài nguyên mà mỏ sắp khai thác cung cấp. Nếu tài nguyên đang mang khác loại với tài nguyên mà mỏ cung cấp, sẽ xóa bỏ hết tài nguyên đang mang và khai thác loại tài nguyên từ mỏ này. Nếu tài nguyên mang trên người là cùng loại với mỏ này, cần thiết phải kiểm tra lượng tài nguyên đang mang đã max hay chưa max để khai thác tiếp tục loại tài nguyên này.
* Có thể trong quá trình khai thác, Player bằng cách nào đó đã khiến ProducerUnit di chuyển ra khỏi phạm vi khai thác của mỏ, như thế điều cần thiết là phải liên tục kiểm tra khoảng cách từ ProducerUnit tới mỏ có còn nằm trong phạm vi khai thác.

Sau khi khai thác tài nguyên, Producer sẽ mang trong mình số lượng tại nguyên mà nó khai thác(duy nhất một loại tại một thời điểm). Để nhận được số lượng tài nguyên này, ProducerUnit sẽ phải quay lại nhà chính để dở bỏ tài nguyên và tiếp tục cho một chuyến khai thác mới

* Mỗi Player có một danh sách các Structure chứa bao gồm cả Structure chính, từ danh sach này ProducerUnit sẽ xác vị trí nhà chính để trở về tháo dở tài nguyên.
* Khi tài nguyên được tháo dở ở nhà chính, tài nguyên trong tài khoản của Player sẽ tăng theo chỉ số mà ProducerUnit lấy về.

1. Hiệu ứng âm thanh (sound effect)

* Vấn đề:

Bất kì một thể loại Game nào, chơi trên hệ máy nào thì âm thanh cũng đã trở thành một phần không thể thiếu. Âm thanh kết hợp hài hòa với hình ảnh trên màn hình, cùng với các hiệu ứng đặc biệt góp phần gây cảm giác hứng thú với người chơi. **Như vậy, vấn đề đặt ra là phải sử dụng hiệu ứng âm thanh trong Game**. Các hiệu ứng âm thanh rất đa dạng và phong phú, từ nhạc nền, âm thanh thể hiện hành động nhân vật(tấn công, chạy)… đến các hiệu ứng đặt biệt như … Ở đây chỉ đưa ra giải pháp cho vấn đề phát âm thanh bằng XNA framework với mức độ đơn giản để làm nền tản, những hiệu ứng đặc biệt được thử nghiệm trong phần demo Game cũng được thực hiện dựa trên nền tản này.

* Giải pháp:
  + Giải pháp một : dùng XACT. Trong XNA framework 2.0, các file âm thanh phải được add vào Content dưới định dạng đặc biệt, được tạo ra từ Microsoft Cross-PlatformAudio Creation Tool, được biết với cái tên XACT, một tool đi kèm khi cài đặt XNA framework. XACT giúp tạo ra Sound bank và Wave bank từ nhiều file wav, AIFF, XMA và biên dịch thành file XAP. File XAP này chứa các âm thanh đã đưa vào bằng XACT. XAP sẽ được load từ Content và được phát thông qua các đối tượng AudioEngine, WaveBank , SoundBank. Các bước thực hiện được tóm tắt như sau.
    - Dùng XACT tạo file XAP từ nhiều file âm thanh muốn dùng làm hiệu ứng.
    - Dùng các đối tượng AudioEngine, WaveBank, SoundBank để phát các âm thanh trong file XAP.

*(ebook Beginning XNA 2.0 Game Programming From Novice to Professional Alexandre Lobão, Bruno Evangelista, and José Antonio Leal de Farias)*

* + Để thể hiện âm thanh trong Game, XNA framework 3.0 cũng cung cấp lớp SoundEffect hoặc (SoundEffectInstance). Việc sử dụng hai lớp này đơn giãn hơn dùng XACT, và XACT không hoạt động trên Zune, nên giải pháp này sẽ là một tùy chọn tốt cho các game trên Zune. Các hàm trong SoundEffect cài đặt cho phép phát âm thanh từ file nguồn với âm lượng khác nhau và vị trí loa khác nhau(loa trái, phải) tùy theo các tham số đầu vào. SoundEffect

Trong phần demo của Game, đã cài đặt hiệu ứng âm thanh thực hiện dựa trên sự tùy biến về âm lượng và vị trí loa này. Vấn đề này được mô tả như sau : các Unit trong Game thường xuyên tham gia vào các trận đánh, âm thanh đi kèm với chúng là những tiếng binh khí va chạm, điều này đã được thực hiện bằng các dùng lớp SoundEffect để phát file âm thanh nguồn có định dạng .wav. Nhưng, trên vùng bản đồ rộng lớn, Viewport chỉ có thể mô tả một phần của bản đồ, vậy thế để giúp người chơi phát hiện có xung đột giữa quân lính bằng âm thanh, một cài đặt nhỏ đã được thực hiện dựa vào sự điều tiết âm lượng và vị trí loa. Bằng cách sử dụng các tham số âm lượng và vị trí loa, quá trình cài đặt được thực hiện như sau:

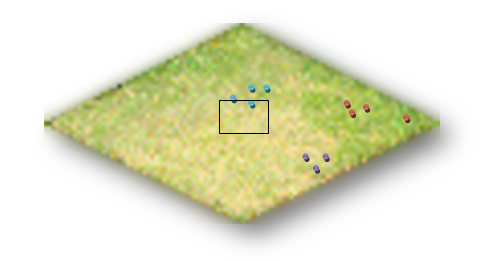
* Cài đặt lớp AudioGame.cs để phát âm thanh với các tham số đầu vào bao gồm âm lượng và âm lượng loa trái phải.
* Trong lớp Unit, trong hàm chuyển ảnh để mô tả trạng thái, kiểm tra Unit đã đi đến ảnh cuối trong tập ảnh của trạng thái ATTACK để phát âm thanh theo các yêu cầu sau:
  + Trong trường hợp xung đột xảy ra tại phần bản đồ nằm trong vùng nhìn của Viewport, âm thanh được phát đồng thời cả hai loa và âm lượng lớn.
  + Trong trường hợp xung đột xảy ra tại phần bản đồ nằm ngoài vùng Viewport, âm thanh sẽ được điều chỉnh nhỏ dần và loa phát tùy theo vị trí của xung đột so với Viewport(loa trái nếu bên trái, loa phải nếu bên phải).

1. Mini Map:

* Vấn đề:

Bản đồ chiến trận có kích thước rất lớn, Viewport chỉ thể hiện một phần trên bản đồ đó. **Như thế, để người chơi có một cách nhìn tổng quát về toàn cảnh trận chiến, cần phải có một hình ảnh thu nhỏ của bản đồ, thể hiện toàn bộ các đối tượng Unit, Structure của các Player, đồng thời hình ảnh thu nhỏ này có thể tương tác với bản đồ thật**.

* Giải pháp:
  + Giải pháp đề ra là:
    - Xây dựng một hình ảnh mô phỏng bản đồ thật với các Unit và Structure của các Player trên đó gọi là mini map.
    - Dùng một hình ảnh thu nhỏ của bản đồ làm nền cho mini map. Hình ảnh này có hình dạng mô tả lại hình dạng của bản đồ thật.
    - Từ vị trí của các Unit và Player trên bản đồ thật, tính ra vị trí của Unit và Structure trên mini map dựa vào tỷ lệ của kích thược bản đồ thật và kích thước ảnh nền mini map.
    - Thể hiện lên mini map các điểm tượng trưng với màu sắc khác nhau tại các vị trí của Unit và Structure đã tính trên mini map nhằm biểu thị Unit và Structure của các Player.
    - Với mini map này, có thể xây dựng nó thành một công cụ hữu ích cho việc quan sát bản đồ thật. Bằng cách thể hiện Viewport trên minimap là một hình chữ nhật nhỏ, mô tả lại kích thước và vị trí của Viewport trên bản đồ thật bằng cách sử dụng phương pháp tỷ lệ như đã dùng với Unit và Structure. Sự tương tác qua lại giữa mini map và bản đồ thật được thể hiện như sau:
      * Khi Viewport được thay đổi vị trí trên map, đồng thời hình chữ nhật thể hiện cũng thay đổi vị trí như vậy.
      * Khi click chọn vị trí mới cho hình chữ nhật trên mini map, Viewport cũng phải thay đổi vị trí trên bản đồ thật đúng với vị trí thể hiện của hình chữ nhật trên mini map.



Như vậy, giải pháp đơn giản chỉ là tính lại vị trí của các thành phần muốn thể hiện trong mini map bằng vị trí các thành phần trên bản đồ thực thông qua phép tỷ lệ.

1. Hiển thị thông tin Structure, Unit, Player trên menu:

* Vấn đề:

Chơi Game là quá trình người chơi thao tác trên các thành phần của Game, bao gồm Unit, Structure thuộc sở hữu của Player đóng vai trò người chơi. Để có thể thực hiện các thao tác, người chơi cần nắm được các thông tin về Unit, Structure đồng thời cả thông tin của Player mình đang điều khiển để ra các quyết định trong công việc xây dựng Structure và mua Unit phù hợp. **Hiển thị thông tin Structure, Unit và Player là vấn đề cần giải quyết.**

* Giải pháp:
  + Giải pháp đề ra:
    - Xây dựng menu hiển thị trong Viewport là nơi sẽ chứa các thông tin cần hiển thị của Game.
    - Hiển thị các thông tin của Player cần được hiện thị bao gồm thông số của các tài nguyên mà Player hiện có. Xây dựng lớp Player.cs với các thành phần mô tả thông tin về các tài nguyên: tên tài nguyên, số lượng. Các thông tin này sẽ luôn luôn được hiển thị trong phần menu của màn hình nhằm mô tả hiện trang tài nguyên của Player.
    - Biểu diễn thông tin Unit: thông tin Unit chỉ được biểu diễn khi người chơi chọn ít nhất một Unit để xem thông tin. Thông tin Unit bao gồm sinh lực, sức mạnh, phạm vi tấn công … và hình ảnh của Unit để thể hiện trạng thái Unit đang làm trên bản đồ.
    - Biểu diễn thông tin của Structure tương tự như thông tin của Unit và cũng chỉ được biểu diễn khi người chơi chọn một Structure để xem thông tin. Ngoài ra, trong việc hiển thị thông tin Structure, bên cạnh các thông tin về sinh lực, hình ảnh trạng thái Structure, mỗi Structure sẽ có một tập các Unit cho người chơi chọn mua, như vậy trong phần thông tin của Structure sẽ hiển thị các thông tin về Unit mà Structure cho phép mua cùng với các yêu cầu về tài nguyên được đòi hỏi để mua các Unit đó.
    - Ngoài ra, trong trạng thái không chọn bất kì Unit hoặc Structure hoặc Unit nào trên bản đồ, Menu sẽ luôn luôn hiển thị thông tin về các Structure mà Player có thể xây dựng cùng với các yêu cầu về tài nguyên để xây dựng.

Như vậy các thông tin hiển thị đều được ghi vào vùng menu của màn hình Viewport. Menu sẽ bao gồm 2 phần, một phần luôn hiển thị thông tin của Player, một phần tùy thuộc trạng thái của thao tác hiện tại từ người chơi, bao gồm, hiển thị thông ti Unit nếu người chơi chọn Unit, thông tin Structure có kèm các Unit có thể mua nếu người chơi chọn Structure, và thông tin các Structure trong tập các Structure mà hiện tại Player có thể xây dựng.

1. Xử lý xây dựng Structure với người chơi:

* Vấn đề:

Trong Game có nhiều loại Structure với các thuộc tính và các loại Unit để mua khác nhau. Tùy vào tài nguyên và các điều kiện khác, cho phép player xây dựng các Structure để phát triển đội quân của mình. **Vấn đề là cần thực hiện các bước xử lý để xây dựng một Structure hoàn chỉnh và đưa vào sử dụng được trong trận đấu.**

* Giải pháp:
  + Giải pháp thực hiện

Với trạng thái hiện tại của người chơi là không chọn Unit hoặc Structure nào trên màn hình, menu sẽ hiển thị các Structure mà hiện tại Player được phép xây dựng. Nếu người chơi chọn một Structure để xây dựng từ menu này, việc xử lý xây dựng Structure sẽ được thực hiện, với các bước cụ thể sau:

* + - Sau khi người chơi click chọn một Structure từ Menu, chuyển đổi màu sắc của hình ảnh hiển thị Structure trong Menu để cho biết người chơi đã chọn xây dựng Structure này. Người chơi click vào vùng bản đồ nằm trong Viewport để chọn vị trí xây Structure, vị trí xây dựng này không được đè lên vị trí của một Structure đã có trước hoặc một Unit đang đứng đó, vì vậy, việc kiểm tra vị trí xây Structure mới được tiến hành để đảm bảo các Structure và Unit đã có trước đó không bị Structure mới đè lên.
    - Kiểm tra số lượng tài nguyên yêu cầu cho Structure, nếu đã có đủ tài nguyên, tiếp tục kiểm tra hiện tại có Unit xây dựng đang không bận xây hay sửa chữa Structure khác thì công việc xây dựng sẽ bắt đầu được tiến hành. Trong trường hợp các bước kiểm tra này không được thỏa đáng, Game sẽ không cho phép người chơi xây Structure này.
    - Trong quá trình xây dựng, Structure không thể đi vào hoạt động. Structure chỉ đi vào hoạt động sau khi đã hoàn tất, việc này được kiểm tra bằng cách dựa vào tập hình ảnh mô tả hoạt động xây dựng của Structure. Mỗi Structure có một khoảng thời gian cần cho việc xây dựng, các hình ảnh được trì hoãn trong quá trình xây dựng để hiển thị từng bước xây dựng cho đến khi thời gian xây dựng kết thúc cũng là hình ảnh cuối trong tập hình được vẽ ra, khi đó Structure được xem là hoàn tất, bằng cách kiểm tra index của tập hình để đến max chưa là có thể biết Structure đã hoàn thành xong.

1. Xử lý mua Unit với người chơi:

* Vấn đề:

Trái ngược với các Structure chỉ đứng yên, Unit là thành phần năng động trong Game, đóng vai trò nhiều mặt tùy vào loại Unit : tấn công, bảo vệ, khai thác tài nguyên, xây dựng… Các Unit sẽ được mua từ các Structure đã được xây dựng nếu Player đáp ứng đủ nhu cầu cấn thiết về tài nguyên, quyền hạn mua Unit … **Vấn đề cần giải quyết là xây dựng các bước để thực hiện chức năng mua Unit của Game từ các Structure, bao gồm cả việc các Unit này xuất hiện thế nào và ở đâu sau khi đã mua thành công.**

* Giải pháp:
  + Giải pháp đề ra:
    - Khi người chơi đang ở trạng thái chọn một Structure của đội mình trên bản đồ, Menu sẽ hiễn thị thông tin Structure đó, đồng thời sẽ hiển thị danh sách các Unit mà đội chơi này được phép mua mà Structure cung cấp được. Danh sách Unit này sẽ bao gồm các hình ảnh đại diện cho Unit cùng với các thông yêu cầu đi kèm. Muốn mua Unit, người chơi nhấn chuột để chọn vào Unit muốn mua từ menu.
    - Chương trình kiểm tra tài nguyên hiện có đủ để mua Unit hay không, khi đó có hai trường hợp xảy ra : đủ tài nguyên để mua Unit, không đủ tài nguyên để mua Unit.
      * Nếu người chơi có đủ tài nguyên để mua Unit, số lượng Unit của loại chọn mua trong hàng đợi được mua sẽ tăng lên, và tài nguyên của Player sẽ giảm với số lượng đã yêu cầu trong menu. Sỡ dĩ các Unit đặt mua phải nằm trong hàng đợi vì một Unit muốn mua xong cần phải có một khoảng thời gian nhất định tùy mỗi loại, vì thế các Unit được lưu trong hàng đợi để chờ đợi thời gian hoàn thành trôi qua. Trong quá trình nằm trong hàng đợi, Player có thể hủy mua Unit bằng cách giảm số lượng Unit trong hàng đợi lại, khi đó tài nguyên đã bỏ ra sẽ phục hồi. Tuy nhiên hàng đợi có giới hạn số lượng, vì thế cùng một loại Unit, chỉ có thể mua với một số lượng nhất định.
      * Trong trường hợp kiểm tra và thấy không đủ tài nguyên để mua Unit, hiển nhiên Unit sẽ không được đưa vào hàng đợi để đợi mua xong.
    - Mặt khác, các Unit sau khi mua thành công cần phải có một chỗ dừng chân, như vậy, mỗi Structure cần có một thuộc tính mô tả về điểm tập trung quân lính sau khi mua. Điểm này sẽ hướng dẫn cho quân lính sau khi hình thành sẽ đi đến và đứng đợi lệnh từ Player. Điểm tập trung quân này là có tính tùy biến, có thể được thay đổi nhằm tăng tính tiện lợi trong quá trình hành quân của quân lính. Vậy, đề giải quyết vấn đề hậu cần cho chỗ dừng chân của quân lính, thuộc tính điểm tập trung quân sẽ được đưa vào lớp Structure.

*Cách* các *giải pháp Xây dựng structure và mua Unit được thực hiện tương tự đối với các Player được điều khiển bởi máy tính, chỉ khác là các thao tác không cần dùng các thiệt bị Input như đối với Player của người chơi.*

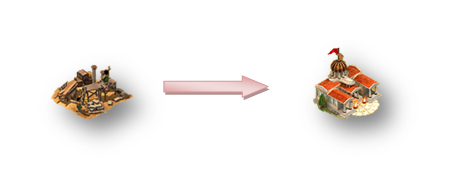
1. :

**Mô tả nội dung Productline Game dàn trận**

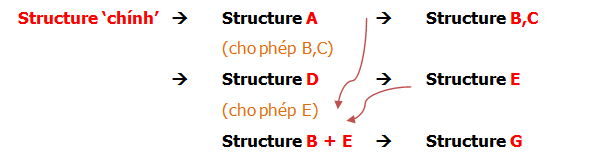
1. Công trình(Structure ):



* 1. Structure bao gồm nhiều loại khác nhau. Mỗi loại có sự khác nhau về thuộc tính, chức năng, số lượng các loại Unit cho Player mua, các loại Unit có thể mở rộng.
  2. Structure khác nhau cũng có sự yêu cầu khác nhau về tài nguyên, về điều kiện trong chuỗi trình tự mua Structure sẽ trình bày dưới đây.
  3. Mỗi player mặc định có một Structure ‘chính’ đã được xây khi game bắt đầu. Structure ‘chính’ là Structure cơ bản nhất, cần có đầu tiên để làm tiền đề cho phép xây dựng các Structure khác.
  4. Structure ‘chính’ có thể mua thêm nếu player có đủ tài nguyên và điều kiện yêu cầu. Số lượng Structure ‘chính’ bị giới hạn. Nếu player đã xây dựng Structure ‘chính’ với số lượng bằng với số lượng tối đa cho phép thì không được phép xây thêm bất kì Structure ‘chính’ nào nữa.
  5. Nếu tất cả các Structure ‘chính’ bị phá hủy, player không thể xây bất kì Structure nào kể cả việc xây một Structure ‘chính’ khác cũng không thể. Vì thế trong một lần vào trận, player không được để mất hết tất cả Structure ‘chính’ cùng một lúc, nghĩa là có ít nhất một Structure ‘chính’ còn tồn tại mới đảm bảo tốt cho hoạt động xây dựng Structure của player.
  6. Structure được xây dựng từng bước(dựng lên từ từ), nếu Structure vẫn trong quá trình đang xây dựng, Structure không thể tham gia vào hoạt động mà phải đợi đến lúc đã hoàn thành. Sau một khoảng thời gian quy định, Structure mới được hoàn thành và sử dụng. Tốc độ xây dựng Structure phụ thuộc vào số lượng Unit xây dựng, số lượng Unit xây dựng càng đông thì Structure được hoàn thành càng nhanh và ngược lại.



* 1. Structure được xây theo trình tự quy định.
     1. Player phải sỡ hữu ít nhất một Structure ‘chính’ mới được xây thêm các Structure khác, mà hiện tại player được phép xây.
     2. Có Structure A và Structure A cho phép xây các Structure B,C… thì player mới được phép xây các Structure B,C… này; tương tự, để xây được E thì cần có D (nếu Structure ‘chính’ có ít nhất một cái còn tồn tại). Ngoài ra, trình tự này còn có sự kết hợp trong quá trình xây dựng, giả sử để xây Structure G thì cần có Structure B + E.



* 1. Mỗi Structrure sỡ hữu một tập các Unit cho phép player mua. Tập Unit này sẽ mở rộng thêm (tăng số lượng loại Unit có thể mua, tăng số lượng tối đa các Unit cùng loại có thể mua so với ban đầu) nếu Player xây dựng một Structure cho phép mở rộng thêm các Unit đó hoặc player có một technology mới.
  2. Structure có các thông số yêu cầu về các loại tài nguyên, player muốn xây dựng Structure phải khai thác đủ số lượng tài nguyên yêu cầu cho mỗi loại (cùng một số điều kiện yêu cầu nếu có: đã có Structure tiên quyết, đã có đủ số Unit …).
  3. Khi Structure bị đối phương tấn công, Structure sẽ bị thiệt hại hay bị phá hủy hoàn toàn.
  4. Nếu Structure bị tấn công và thiệt hại, Player có thể sửa chữa Structure. Việc sửa chữa yêu cầu tốn kém tài nguyên, nhưng tài nguyên bị tốn kém trong quá trình sửa chữa không lớn hơn số tài nguyên bỏ ra cho việc xây dựng Structure đó.
  5. Khi Structure bị phá hủy bởi Player sở hữu, Player sở hữu sẽ thu hồi được một số tài nguyên từ việc phá hủy Structure, nhưng số tài nguyên này không lớn hơn số tài nguyên mà Player đã bỏ ra để xây dựng trước đó.

1. Mỏ tài nguyên(Resource center)
   1. Tài nguyên không thuộc sở hữu của bất kì player nào và có thể nằm bất cứ vị trí nào trên bản đồ. Mỗi mỏ tài nguyên chỉ cung cấp 1 loại tài nguyên cụ thể, không cung cấp cùng một lúc nhiều loại tài nguyên(VD: không thể vừa cung cấp vàng, vừa cung cấp đá).
   2. Mỏ tài nguyên có một số lượng tài nguyên nhất định ban đầu và có thể tăng lên dần tùy vào loại mỏ tài nguyên như sẽ trình bày dưới đây. Mỏ sẽ bị giảm số lượng tài nguyên mà mỏ có nếu bị Unit khai thác mỏ.
   3. Nhiều player có thể khai thác chung một mỏ tài nguyên, để chiếm mỏ tài nguyên và ngăn cho đối phương khai thác, player có thể cho quân bao vây mỏ tài nguyên, như thế sẽ phát sinh khả năng xung đột khi nhiều player mún khai thác cùng một mỏ tài nguyên.
   4. Mỏ tài nguyên được chia thành hai loại, loại có thể phục hồi và loại không thể phục hồi:
      1. Mỏ phục hồi sẽ tăng dần số lượng tài nguyên mà nó có một cách liên tục theo thời gian với số lượng thấp. Dù khai thác hết số lượng tài nguyên của mỏ, thì các player có thể đợi cho tới khi mỏ tăng dần tài nguyên và có thể khai thác tiếp.
      2. Mỏ tài nguyên không thể phục hồi sẽ bị mất đi liên tục nếu khai thác và khi hết sẽ không còn dùng được nữa.
   5. Mỏ tài nguyên có giới hạn về số lượng, nếu khai thác hết số lượng của mỏ, không thể khai thác thêm, phải chuyển qua mỏ khác hoặc đợi mỏ phục hồi.
   6. Mỏ tài nguyên bao gồm nhiều loại mỏ, mỗi loại mỏ cung cấp một loại tài nguyên. Player cần có nhiều loại tài nguyên phù hợp với yêu cầu của Structure và Unit sẽ được xây dựng Structure và mua Unit.
2. Quân lính(Unit)



* 1. Unit bao gồm nhiều loại. Mỗi loại Unit có sự khác nhau về thuộc tính, bao gồm sức mạnh, thể lực, phạm vi tấn công … hoặc khác nhau về chức năng như Unit tấn công, Unit khai thác, Unit xây dựng…
  2. Mỗi loại Unit cũng khác nhau về số lượng yêu cầu cho mỗi loại tài nguyên, cùng với những điều kiện khác(Structure, technology,…) mỗi khi mua loại Unit đó.
  3. Mỗi loại Unit sẽ được Player mua, nếu hiện tại Player được phép mua và có đủ các yêu cầu cần thiết về tài nguyên để mua.
  4. Unit được mở rộng tùy theo Structure xây dựng hoặc technology, bao gồm các loại mở rộng sau:
     1. Mở rộng số lượng loại Unit được mua của Player.
     2. Mở rộng số lượng tối đa các Unit được mua cho mỗi loại Unit.
  5. Unit được tăng điểm kinh nghiệm sau mỗi lần chiến đấu (nếu còn sống), khi đạt đến ngưỡng cao nhất cần thiết cho level kế tiếp, level của Unit sẽ tăng và điểm kinh nghiệm lại tiếp tục bắt đầu cho một lần tăng level khác.
  6. Các thuộc tính của Unit sẽ được cải tiến cùng với level của Unit, các Unit cùng loại có level khác nhau thì thuộc tính cũng khác nhau.
  7. Các thuộc tính của Unit sẽ được cải tiến cùng với các technology mà Player có, các Unit khác loại thì thuộc tính cũng khác nhau tùy vào các technology mà Player sở hữu.
  8. Đối với Unit có khả năng tấn công:
     1. Unit tự động phát hiện đối phương (Structure hoặc Unit không phải là cùng đội chơi) trong phạm vi cho phép và áp sát vào đối phương để thu hẹp phạm vi tấn công.
     2. Đến khi khoảng cách tấn công giữa Unit và đối phương được thu hẹp đến mức Unit có thể tấn công đối phương được, Unit sẽ thực hiện tấn công đối phương nếu là phe đối kháng, tiêu hủy sinh lực đến khi một trong hai không còn sinh lực và chết.



* + 1. Đối phương mà Unit phát hiện được là đồng minh, Unit không cần phát động tấn công(không cần áp sát, không cần tấn công).
  1. Đối với Unit có khả năng khai thác:
     1. Unit có thể khai thác nhiều loại tài nguyên, nhưng chỉ được phép mang trên người một loại tài nguyên trong cùng một thời điểm. Nếu tại thời điểm này, Unit mang lượng tài nguyên **a** khai thác được từ mỏ **A**, nhưng chưa về lại căn cứ để cung cấp tài nguyên, mà lại chuyển hướng qua mỏ **B** để khai thác **b**, thì toàn bộ tài nguyên a đang mang trên người xem như mất hết.
     2. Với tài nguyên đang mang trên người trong quá trình khai thác mà chưa trở về căn cứ để cung cấp, số lượng phải không vượt quá lượng tài nguyên tối đa mà Unit được mang theo. Nghĩa là, Unit có một giới hạn số lượng tài nguyên mang trên người cho mỗi lần khai thác.
     3. Unit khai thác, ban đầu, không có khà năng tấn công hoặc khả năng tấn công rất yếu. Khả năng và thuộc tính của Unit sẽ được mở rộng theo việc mở rộng Structure và Technology.
  2. Đối với Unit xây dựng:
     1. Unit xây dựng, ban đầu, không có khà năng tấn công hoặc khả năng tấn công rất yếu. Khả năng và thuộc tính của Unit sẽ được mở rộng theo việc mở rộng Structure và Technology.
     2. Unit xây dựng tại một thời điểm chỉ tham gia xây dựng hoặc sửa chữa một Structure.

1. Đội chơi(player)
   1. Một player có thể là đồng minh của các player khác. Các đội là đồng minh sẽ không đụng độ quân. Các đội không là đồng minh là đối kháng, quân lính sẽ tấn công nhau nếu gặp.
   2. Các Player làm đồng minh của nhau không được chia sẽ tài nguyên đã khai thác được, mà chỉ cùng nhau khai thác chung một mỏ tài nguyên, không tấn công lẫn nhau và hợp lực cùng tấn công phe đối kháng.
   3. Player có một tập các tài nguyên(resource) , ngoài một số lượng nhỏ cho mỗi loại tài nguyên mà player có sẵn khi bắt đầu cuộc chơi, player phải dùng người khai thác các mỏ tài nguyên để có đủ số lượng tài nguyên cho mỗi loại.
   4. Player phải có đủ số lượng tài nguyên (cùng một số điều kiện yêu cầu nếu có: đã có Structure tiên quyết, đã có đủ số Unit …) cho mỗi loại mà Structure hoặc Unit đó yêu cầu mới được xây dựng Structure hoặc mua Unit.
   5. Việc xây Structure và mua Unit được player điều khiển
      1. Player sẽ chọn Structure để xây dựng, bỏ ra một số lượng tài nguyên tương ứng với lượng tài nguyên mà Structure khi chấp nhận xây Structure đó, chọn vị trí sẽ xây Structure trên bản đồ.
      2. Player sẽ chọn Unit sẽ mua, bỏ ra một số lượng tài nguyên bằng với lượng tài nguyên mà Unit yêu cầu trên mỗi đơn vị Unit được mua.
   6. Nếu hủy hay giảm bớt số Unit đang đợi mua trong hàng đợi, tài nguyên bỏ ra để mua Unit đó được phục hồi trong tập tài nguyên của player. Nếu chấp nhận phá hủy Structure để thu hồi tài nguyên, player chấp nhận thu hồi lại chỉ một phần tài nguyên, nhỏ hơn số tài nguyên đã bỏ ra để xây dựng Structure đó.
   7. Player có thể phá hủy để thu hồi tài nguyên cả Structure ‘chính’ nhưng phải đảm bảo còn ít nhất một Structure chính của mình trên bản đồ điều đó có nghĩa là không thể phá hủy Structure ‘chính’ nếu nó là cái duy nhất còn tồn tại.
   8. Player điều khiển Unit, di chuyển Unit trên bản đồ, tấn công Unit hoặc phá hủy Structure của phe đối kháng, điều khiển Unit lấy tài nguyên từ các mỏ tùy loại Unit.
   9. Player sẽ thua trận trong cuộc chơi, nếu không còn bất kì Unit hoặc Structure nào thuộc sở hữu của mình có mặt trên bản đồ(tất cả các Unit đều bị giết, tất cả các Structure đều bị phá huỷ) hoặc chấp nhận đầu hàng.