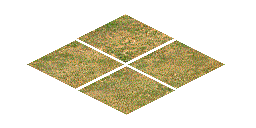
1. Xây dựng bản đồ vùng chiến sự

* Vấn đề :

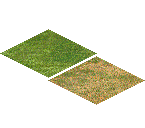
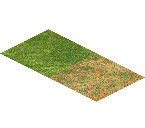
Bản đồ vùng chiến sự là nền để sắp đặt và bố trí các Sprite lên đó, bao gồm Terrain, Resource center, Unit và Structrure. Bản đồ có kích thước lớn hơn nhiều so với vùng nhìn của cửa sổ (Viewport) cho phép người dùng kéo vùng nhìn của cửa sổ trên khắp bản đồ để quan sát cục diện trận chiến trong Game, 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.

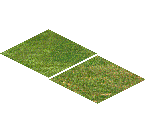
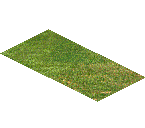
* Giải pháp:
  + Giải pháp thứ nhất là Load một ảnh thật lớn và vẽ ra như một bản đồ chiến sự trên màn hình Game. Nhưng khuyết điểm của giải pháp này là XNA framework chỉ cho phép load hình ảnh có kích thước tối đa là **2024 x 2024 px**, như vậy nếu load một ảnh thật lớn cũng chỉ có thể load được kích 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.



Như vậy, các bước thực hiện cho giải pháp này là

* + - 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. Vẽ bản đồ vùng chiến sự với ma trận Cell:

* 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

*i1 i2*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| *j1* | ***A*** |  |  | ***B*** |  |
|  |  |  |  |  |  |
| *j2* | ***C*** |  |  | ***D*** |  |
|  |  |  |  |  |  |

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

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | j1 | i1 |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  | **A** |  |  | **B** |  |  |  |
| j2 |  |  |  |  |  |  |  |  | i2 |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  | **C** |  |  | **D** |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

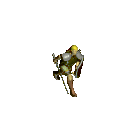
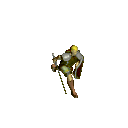
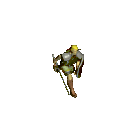
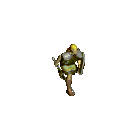
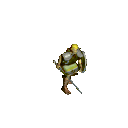
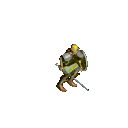
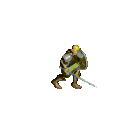
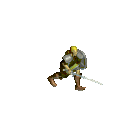
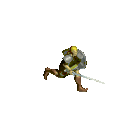
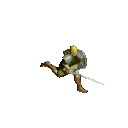
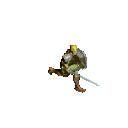
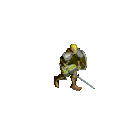
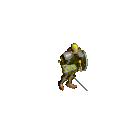
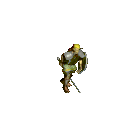
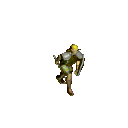
Thực hiện tương tự như dạng bản đồ hình vuông nhưng cách xác địnhi và j khác do cách biểu diễn hệ trục là khác

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 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 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 hình vượt quá ngưỡng.

1. Quản lý Structure và Unit trên map sao cho truy vấn 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)… vấn đề đặt ra là làm cách nào để quản lý và truy xuất các Sprite theo từng loại nhanh và dễ.

* Giải pháp

Chia các Sprite thành các mảng con mà các phần tử là một loại Sprite muốn quản lý

* Mảng các Unit hiện có trong BattleField
* Mảng các Structure hiện có trên BattleField
* Thực hiện vòng lặp trên 2 mảng này

1. Xác định 1 unit hoặc 1 structure trong phạm vi tấn công

* Vấn đề
  + - Trừ health
    - Xác định đối tượng tấn công có còn trong phạm vi tấn công hay ko(đối phương có thể đã bỏ chạy)
    - Đối phương đã die hay chưa(Nếu còn health thì còn đánh, nếu hết health thì ko tấn công nữa)
* Giải quyết:
  + - Một Unit cần biết nó sẽ attack ai 🡪 Whom I Hit

🡪 thêm thuộc tính **\_whomIHit**

1. Structure hoặc Unit đã hết health 🡪 không còn tồn tại trên map 🡪 dispose
2. Producer Unit
   * Xác định vị trí resource center để lấy

**Giải quyết:**

* + - * Duyệt mảng các resource center trên bản đồ
        + Phát hiện ra resource center mà nó có thể khai thác
        + Khai thác resoure center này nếu

Resource mà nó đang mang trên người số lượng chưa max

Nó hiện ko có mang resource nào

Resource center mà nó muốn khai thác khác resource mà nó mang trong người

* + - * Kiểm tra khi nó di chuyển, nó đã chạy ra ngoài phạm vi của resource center để có thể khai thác resource chưa. Nếu chưa ra khỏi phạm vi 🡪 khai thác tiếp, nếu đã ra khỏi phạm vi 🡪 đứng yên
  + Xác định vị trí nhà chính để cung cấp resource
  + Tự động bỏ chạy khi bị tấn công hoặc đứng yên để cho đối phương tấn công(power = 0)
    - Chạy hướng nào, xa bao nhiêu

**Giải quyết:** chưa có

* + - (Hoặc tiếp tục khai thác cho đến khi bị đánh chết 🡪 máu phải thật nhiều để chịu đựng cho tới khi người chơi di chuyển nó bỏ chạy)

1. Hiệu ứng âm thanh (sound effect)
   * Có 2 cách tiếp cận
     + Dùng XACT(nói sau)
     + Không dùng XACT 🡪 sẽ dùng class SoundEffect hoặc SoundEffectInstance, cách này đơn giãn hơn dùng XACT

(XACT không hoạt động trên Zune, nên cách 2 sẽ là 1 tùy chọn tốt cho các game trên Zune)