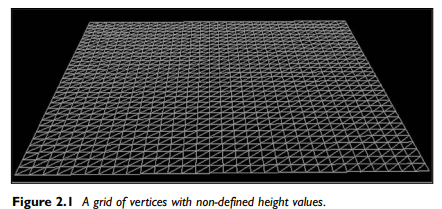
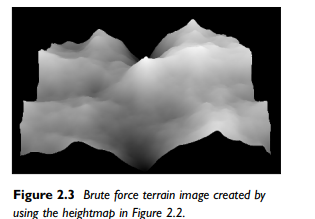
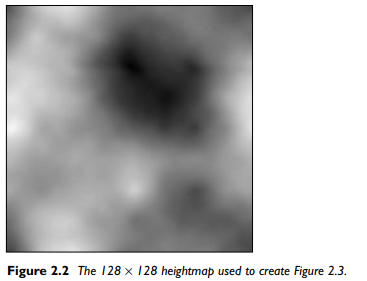
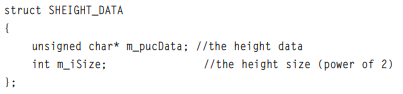
1. Chương 1: Height map
   1. Nội dung
      * Hightmap là gì, làm thế nào để tạo chúng, load chúng như thế nào.
      * Làm thế nào để kiết xuất địa hình sử dụng thuật toán brute force.
      * Làm thế nào để generate fractal địa hình sử dụng hai thuật toán: fault formation và midpoint displacement.
   2. Heightmaps
      * Ta có lưới đa giác như sau được tạo thành từ các đa giác(dọc theo phương X và Z) thông thường và không có giá trị chiều cao:



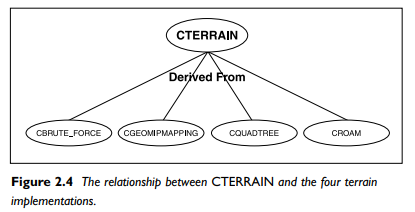
* + - Heightmap là một danh sách (series) các biến có kiểu unsigned char(giá trị từ 0-255) màu xám để gán cho từng điểm. Màu càng tối thì độ cao càng thấp hơn càng sáng thì độ cao càng cao hơn.



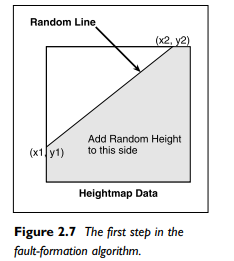
* + - Thông thường định dạng của định dạng heightmap có kiểu định dạng raw. Vì nó chứa dữ liệu thuần túy đơn giản dễ dàng hơn để tải sơ đồ chiều cao vào.
    - Cấu trúc đơn giản của một đối tượng heightmap như sau



* 1. Tạo class BaseTerrain
     + Chúng ta cần tạo một lớp BaseClass mà từ đó có tất cả các địa hinh của thể của chúng ta: brute force, geomipmapping, v.v. Đây là hình ảnh của ý tưởng đó.

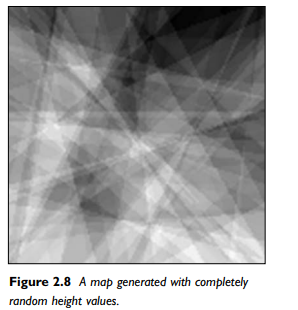


* 1. Fractal Terrain Generation
     + Có 2 thuật toán tạo địa hình Fractal terrain generation: fault formation và midpoint displacement.
     + Thuật toán Fault formation
       - Một phương pháp tạo địa hình fractal được gọi là sự hình thành đứt gãy(fault formation). Fault formation là quá trình tạo ra các “đứt gãy(faults)” trên địa hình; cho phần lớn, nó tạo ra địa hình khá bằng phẳng. Về cơ bản, tất cả những gì chúng tôi làm là thêm một dòng(line) ngẫu nhiên vào trường chiều cao trống, sau đó chúng tôi thêm một dòng ngẫu nhiên chiều cao sang một bên. Xem Hình 2.7 nếu bạn gặp sự cố hình dung điều này hoặc nếu bạn chỉ muốn xác nhận rằng hình ảnh trong đầu của bạn (hoặc, nếu bạn giống tôi, giọng nói trong đầu bạn—tôi lạ như vậy) là đúng.

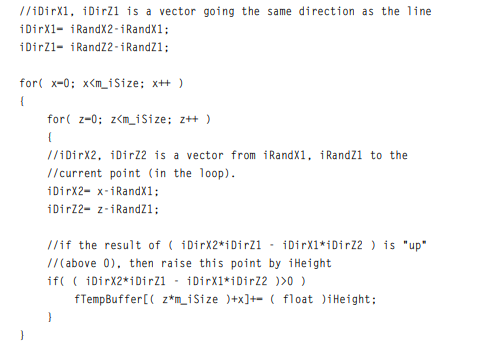


* + - * Tất nhiên, đó chỉ là bước đầu tiên trong toàn bộ quá trình. vẫn còn một số điều bạn cần biết về thuật toán trước khi tiếp cận các giai đoạn nâng cao của nó. Đầu tiên, giá trị chiều cao mà tôi đã nói đến trước đó cần phải được giảm với mỗi lần lặp lại. Tại sao, bạn có thể hỏi? Chà, nếu bạn không giảm chiều cao sau mỗi lần vượt qua, bạn sẽ kết thúc lên bằng sơ đồ chiều cao như Hình 2.8. Xem Hình 2.9 để biết ví dụ về bản đồ chiều cao sẽ trông như thế nào.
      * Lưu ý, trong Hình 2.8, các đốm sáng/tối không có vần điệu hoặc lý do như thế nào; họ chỉ trải ra khắp nơi. Điều này sẽ tốt cho địa hình hỗn loạn, nhưng chúng tôi muốn tạo ra những ngọn đồi thoai thoải. Không có nỗi sợ; sửa chữa vấn đề này là khá đơn giản. Chúng tôi muốn giảm tuyến tính giá trị chiều cao mà không làm nó giảm xuống 0. Để làm điều này, chúng tôi sử dụng các phương trình sau (lấy từ demo2\_2):

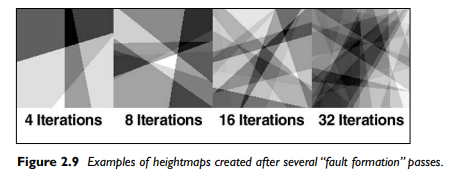




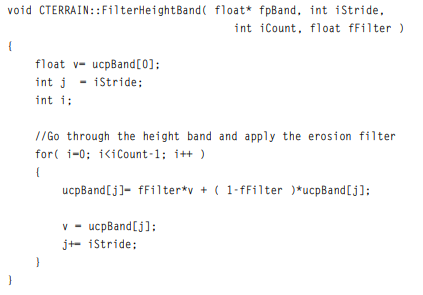
* + - * iMinDelta, iMaxDelta và iIterations đều được cung cấp dưới dạng function arguments. iMinDelta và iMaxDelta đại diện cho mức thấp nhất và cao nhất (tương ứng) các giá trị mà bạn muốn cho chiều cao khi tạo mới lỗi lầm. Tôi có xu hướng gắn bó với giá trị 0 cho iMinDelta và giá trị 255 cho iMaxDelta. iIterations, như tôi đã nói trước đây, đại diện cho số lượng lỗi vượt qua để thực hiện (số lần khác nhau mà sơ đồ chiều cao phải được chia). Và cuối cùng nhưng không kém phần quan trọng, iCurrentIteration đại diện cho số lần lặp hiện tại
      * Như tôi đã nói trước đó, chúng tôi chỉ muốn nâng cao một bên của ranh giới, và chúng tôi muốn tăng giá trị chiều cao của mọi điểm ở phía đó của đường thẳng. Do đó, chúng ta sẽ phải lặp qua tất cả các giá trị chiều cao cho toàn bộ sơ đồ chiều cao. Tất cả điều này là dễ dàng để thực hiện; chỉ là liên quan đến một số toán học đơn giản. Chúng tôi có một vectơ đi theo hướng của đường thẳng của chúng tôi (được xác định bởi hai điểm ngẫu nhiên mà chúng tôi được tạo trước đó) và hướng của nó được lưu trữ trong (iDirX1, iDirZ1). Các vectơ tiếp theo mà chúng tôi muốn tạo là một vectơ từ ngẫu nhiên ban đầu điểm (iRandX1, iRandZ1) ​​đến điểm hiện tại trong vòng lặp (x, z). Sau đó thế là xong, chúng ta cần tìm thành phần Z của tích chéo, và nếu nó lớn hơn 0, thì chúng ta cần nâng dòng điện điểm trong câu hỏi. Tất cả các giải thích trước đó được hiển thị tiếp theo trong mã từ bản demo.



* + - * Trong khi bạn đang nhìn vào fault formation and midpoint displacement code trong các hai phân đoạn và trong demo2\_2, bạn có thể lưu ý cách tôi tạo bộ đệm tạm thời, fTempBuffer, của các giá trị dấu phẩy động để dính tất cả các giá trị chiều cao trong. Nếu bạn còn nhớ, mặc dù vậy, tôi đã nói về bản đồ chiều cao của chúng tôi là một mảng biến unsigned char.Tại sao tôi sẽ sử dụng các biến dấu phẩy động trong này tình huống? Tôi đã làm điều này bởi vì các thuật toán cần một mức độ chính xác cao hơn so với của chúng tôi bộ đệm chiều cao char không dấu bình thường. Sau khi chúng tôi đã tạo và chuẩn hóa toàn bộ sơ đồ chiều cao, tôi chuyển tất cả thông tin từ fTempBuffer đến bộ đệm chiều cao trong Lớp CTERRAIN, m\_heightData.
      * Kiểm tra hình 2.9 để xem một số bản đồ chiều cao hình thành sử dụng sự hình thành lỗi và thay đổi số dòng lỗi lặp đi lặp lại. Gần như chúng ta, chúng tôi chưa hoàn thành với thuật toán này chưa! Trong trường hợp bạn không để ý, cái bản đồ trong hình trước nhìn phi địa hình (mới từ). Chúng ta cần phải vượt qua bộ lọc xói mòn trên toàn bộ bản đồ



* + - * sau khi chúng tôi hình thành một fault mới để làm phẳng các giá trị mà chúng tôi có. Cái này quá trình rất nhiều, nếu không muốn nói là chính xác, giống như truyền một bộ lọc mờ qua một hình ảnh trong chương trình sơn yêu thích của bạn. Nếu nó giúp bạn hiểu giải thích sau đây, chỉ cần nghĩ về nó như thế.
      * Những gì chúng ta sẽ làm là áp dụng một bộ lọc FIR đơn giản, như đề xuất của Jason Shankel.1 Bộ lọc này nhằm mô phỏng sự xói mòn địa hình, điều thường xuyên xảy ra trong tự nhiên. (Bạn đã bao giờ xem một bộ của những ngọn núi trong tự nhiên trông giống như các bản đồ chiều cao trong Hình 2.9?) Chúng tôi sẽ lấy dữ liệu theo dải, thay vì lọc toàn bộ sơ đồ chiều cao cùng một lúc. Chức năng lọc trông như thế này:



* + - * Hàm này lấy một dải giá trị chiều cao và đi qua chúng đánh giá theo giá trị, với iStride chỉ định mức tăng các giá trị theo cho mỗi lần lặp trong vòng lặp. iStride cũng ra lệnh mà hướng đi vì chúng tôi sẽ lọc toàn bộ trường chiều cao từ trên xuống dưới, từ dưới lên trên, từ trái sang phải và phải sang trái. nhiều nhất dòng quan trọng của toàn bộ chức năng là dòng này:



* + - * Đây là dòng làm mờ/xói mòn(blurs/erodes). Các giá trị khác nhau cho fFilter ảnh hưởng đến làm mờ(blurring). 0.0f hoàn toàn không bị mờ và 1.0f là độ mờ thực sự mạnh. Thông thường, chúng tôi muốn các giá trị nằm trong khoảng 0.3f đến 0.6f, tùy thuộc vào bạn muốn địa hình bằng phẳng như thế nào. Bây giờ, ví dụ, hãy nói chúng tôi có giá trị bộ lọc là 0.25f và giá trị dải hiện tại(current band value) là 0.9f. Các phương trình trước đó sẽ trông như thế này:



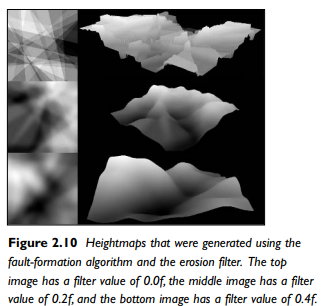
* + - * Sau khi chúng tôi thực hiện các tính toán ban đầu, phương trình trước đó sẽ đơn giản hóa xuống thế này:



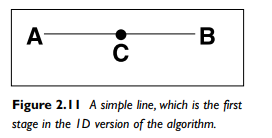
* + - * 0.675f là giá trị mới cho pixel heightmap mà chúng tôi đang làm mờ,nhưng bây giờ nó cần được nội suy với pixel trước nó. (Tốt cung cấp cho pixel đó giá trị là 0,87f.) Chúng tôi áp dụng bộ lọc làm mờ 0.25f giá trị cho pixel đó và thêm nó vào giá trị pixel không được nội suy cho pixel mà chúng tôi đang cố gắng tính toán:



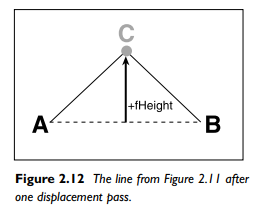
* + - * Thực hiện các phép tính cuối cùng, chúng ta có giá trị là 0,8925f. Vì thế, bạn thấy đấy, tất cả những gì chúng tôi đang thực sự làm ở đây là “pha trộn” một chút pixel với pixel hiện tại. Hãy xem Hình 2.10 để xem cách lọc trông như thế nào trên quy mô lớn hơn nhiều so với các hoạt động trên mỗi pixel mà chúng tôi trước đây đã nói về.
      * Chơi xung quanh với demo2\_2 một chút. Tôi đã tạo một khu vực mới của menu cho thao tác bản đồ chiều cao và bây giờ bạn có thể tạo bản đồ chiều cao mới một cách năng động. Nếu bạn tìm thấy một cái bạn thích, chỉ cần chọn Lưu hiện tại tùy chọn và sơ đồ chiều cao sẽ được lưu vào thư mục của chương trình. Khi bạn chọn tùy chọn Fault Formation, một hộp thoại sẽ mở ra và nhắc bạn nhập một giá trị để biết chi tiết. Giá trị này là một số nguyên giá trị, vì vậy hãy giữ các số trong phạm vi 0–100 cho việc này. Bây giơ là thời gian cho một số niềm vui dịch chuyển trung điểm



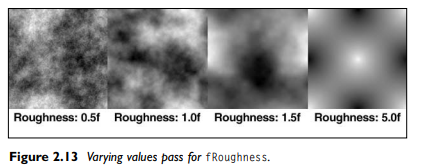
* + - Thuật toán midpoint displacement
      * Fault formation hoạt động tuyệt vời cho một cảnh nhỏ đẹp bao gồm một số những ngọn đồi nhỏ, nhưng nếu bạn muốn thứ gì đó hỗn loạn hơn thế thì sao, chẳng hạn như một dãy núi? Chà, đừng tìm đâu xa. Midpoint displacement là câu trả lời mà bạn đang tìm kiếm! Thuật toán này còn được gọi là plasma fractal và thuật toán hình vuông kim cương(The diamond-square algorithm). Tuy nhiên, midpoint displacement nghe hay hơn rất nhiều và nó mang lại cho người đọc (chính là bạn) một ý tưởng tốt hơn về những gì thực sự đang diễn ra trong toàn bộ quá trình này, vì vậy tôi sẽ dính vào thuật ngữ đó hầu hết thời gian.
      * Tất cả chúng tôi đang làm trong này về cơ bản, thuật toán đang lấy điểm giữa của một dòng và thay thế nó! Để tôi cung cấp cho bạn một chiều chạy xuyên qua. Nếu chúng ta có một đường thẳng đơn giản, chẳng hạn như AB trong Hình 2.11, chúng ta sẽ lấy trung điểm, biểu diễn là C trong hình, và di chuyển nó!



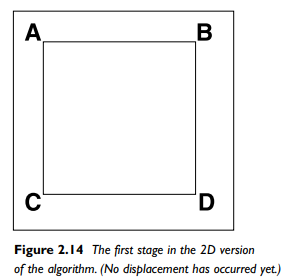
* + - * Bây giờ, chúng ta sẽ dịch chuyển điểm giữa của đường thẳng đó bằng một giá trị chiều cao, mà chúng ta sẽ gọi là fHeight (xem Hình 2.12). Chúng tôi sẽ làm cho nó tương đương với chiều dài của đoạn thẳng được đề cập và chúng tôi sẽ dịch chuyển điểm giữa bằng một phạm vi từ –fHeight/2 đến fHeight/2. (Chúng tôi muốn chia nhỏ dòng thành hai mỗi lần và chúng tôi muốn thay thế chiều cao của dòng ở đâu đó trong phạm vi đó.)



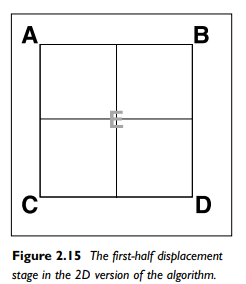
* + - * Sau lần vượt qua đầu tiên, chúng ta cần giảm giá trị của fHeight để đạt được độ nhám mà chúng ta mong muốn. Để làm điều này, chúng ta chỉ cần nhân fHeight với 2-fRoughness, trong đó fRoughness là một hằng số đại diện cho mong muốn độ gồ ghề của địa hình. Người dùng sẽ chỉ định giá trị cho fRoughness, vì vậy bạn cần biết một chút về các giá trị khác nhau mà bạn có thể đặt cho nó. Về mặt kỹ thuật, giá trị có thể là bất kỳ giá trị dấu phẩy động nào mà trái tim bạn mong muốn, nhưng kết quả tốt nhất là từ 0.25f đến 1.5f. Kiểm tra Hình 2.13 để có một chỉ báo trực quan về những mức độ nhám khác nhau có thể làm được.
      * Như bạn có thể thấy, giá trị bạn chuyển cho fRoughness ảnh hưởng lớn đến nhìn của bản đồ chiều cao. Các giá trị thấp hơn 1.0f tạo ra sự hỗn loạn địa hình, các giá trị 1.0f tạo ra giao diện khá “cân bằng” và các giá trị đều lớn hơn 1.0f tạo ra địa hình bằng phẳng. Bây giờ, hãy chuyển lời giải thích này sang chiều thứ hai.



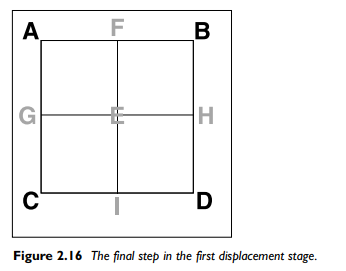
* + - * Giữ lời giải thích 1D trong đầu bạn liên tục khi chúng ta nói về cần thay đổi điều gì đối với phần giải thích 2D vì mọi khái niệm bạn chỉ cần đã học cho dòng đơn đó vẫn được áp dụng. Ngoại lệ là, thay vào đó tính toán điểm giữa cho một dòng, bây giờ chúng ta phải tính toán trung điểm của bốn dòng khác nhau, lấy trung bình cộng rồi cộng giá trị chiều cao ở giữa hình vuông. Hình 2.14 cho thấy chỗ trống hình vuông (ABCD) mà chúng ta bắt đầu với.



* + - * Như tôi đã nói một giây trước, chúng ta phải tính trung điểm cho cả bốn các đường thẳng (AB, BD, DC, CA). Điểm kết quả, E, phải trực tiếp trong chính giữa hình vuông. Sau đó, chúng tôi thay thế E bằng cách lấy trung bình của Các giá trị chiều cao của A, B, C và D, sau đó chúng tôi thêm một giá trị ngẫu nhiên vào phạm vi từ –fHeight/2 đến fHeight/2. Điều này dẫn đến hình ảnh hiển thị trong Hình 2.15.



* + - * Đó chỉ là nửa đầu của giai đoạn dịch chuyển đầu tiên. Bây giờ chúng ta phải tính các giá trị chiều cao cho từng trung điểm mà chúng ta được tìm thấy trước đó. Tuy nhiên, điều này tương tự như những gì chúng tôi đã làm trước đây; chúng tôi chỉ lấy trung bình các giá trị độ cao của các đỉnh xung quanh và thêm một giá trị độ cao ngẫu nhiên trong khoảng từ –fHeight/2 đến fHeight/2. bạn kết thúc với một hình vuông như trong Hình 2.16.
      * Sau đó, bạn quay lại tập hợp các hình chữ nhật tiếp theo và thực hiện cùng một quy trình. Tuy nhiên, nếu bạn hiểu lời giải thích 1D, thì bạn nhất định phải hiểu giải thích 2D và mã đi kèm, demo2\_2, được tìm thấy trên CD trong Code\Chương 2\demo2\_2. Biên dịch thông tin, như thường lệ, được cung cấp dưới dạng tệp văn bản trong bản trình diễn danh mục. Đi kiểm tra bản demo. Điều khiển giống như lần trước thời gian (xem Bảng 2.1 để biết lời nhắc), nhưng lần này, khi bạn bấm vào Độ dịch chuyển điểm giữa cho trường Chi tiết, bạn muốn các giá trị trong trường phạm vi từ 0 (địa hình thực sự hỗn loạn) đến 150 (địa hình đơn giản). Chúc vui vẻ!



1. Chương 3: Texturing Terrain
2. Chương 4: