Course - Lập trình AI cho Unreal Engine

Chúng ta đã học được gì?

Lướt qua các chương trước. Chúng tôi cũng sẽ nói về các ví dụ bổ sung về những gì chúng ta có thể đạt được với những bài học kết hợp này.

Tags: Chúng ta đã học được gì

Trong chương này, chúng ta sẽ lướt qua một số chương. Chương này rất quan trọng vì chúng ta sẽ khám phá ra những cảnh báo trước về những gì chúng ta đã làm. Cuối cùng, chúng ta sẽ nói về những ví dụ khác về những gì chúng ta có thể đạt được với những bài học này.

Trong mỗi chương, mục tiêu là bắt đầu với một mục tiêu. Sau đó, chúng ta tiếp cận mục tiêu bằng cách trình diễn kiến thức thực tế. Chúng ta đã nói về ý nghĩa của AI đối với trò chơi AI. Sau đó, chúng ta đã trình bày những kiến thức cơ bản về cách tạo AI trong trò chơi trong Unreal Engine 4. Những gì chúng ta chưa đề cập đến là những vấn đề mà chúng ta có thể gặp phải. Những điểm yếu và lợi ích của các kỹ thuật chúng ta đã trình diễn là gì? Trong chương này, chúng ta sẽ trả lời những câu hỏi này.

Tạo AI căn bản

Có nhiều hình thức AI khác nhau. Trong cuốn sách này, chúng ta đã tạo ra AI cơ bản. Trong hệ thống phân cấp của các component, có một controller. Controller sau đó quyết định Behavior Tree nào sẽ chạy. Sau đó, các tác vụ được chọn dựa trên chuỗi các quyết định trong cây.

Trong controller, chúng ta có thể có một loạt các Behavior Tree khác nhau phù hợp với các tác vụ khác nhau. Sau đó, khi chúng ta tiếp cận tác vụ, cây Behavior Tree chứa nhiệm vụ con sẽ hỗ trợ đưa ra giải pháp. Việc này đòi hỏi phải biết trạng thái nào là cần thiết để tiếp cập nhiệm vụ.

Chúng tôi đã trình bày cách tạo một AI đủ cơ bản để chạy vô thời hạn. Trong đó khi sử dụng toán học, chúng ta có thể giúp AI tránh tiếp xúc với các vật cản chẳng hạn như tường. Có lẽ, đây là tất cả những gì bạn cần cho một số tình huống. Nếu bạn cần tất cả các đơn vị của mình chạy độc lập với nhau và tránh các bức tường khi cần thiết, thì những hướng dẫn script kịch bản đó sẽ hoàn hảo cho bạn. Tuy nhiên, nếu bạn cần kiểm soát nhiều hơn đối với các đơn vị của mình, thì bạn không thể đạt được điều đó chỉ bằng kỹ thuật này.

Giải pháp thay thế là sử dụng một tiện ích điều khiển (control widget), chẳng hạn như một Spline, để kiểm soát chính xác vị trí của AI. Nếu bạn nắm lấy hướng đến vị trí trên đường dẫn, AI có thể được yêu cầu di chuyển đến đó thay vì hướng về anh hùng, như chúng ta đã trình bày trong *Chương 2, Tạo AI cơ bản*.

Dưới đây là những ưu và nhược điểm của việc sử dụng các công cụ điều khiển để chỉ định vị trí của AI.

Ưu và nhược điểm của việc sử dụng công cụ điều khiển

Có ưu điểm sau:

- Di chuyển AI đến vị trí chính xác

Có khuyết điểm sau:

- Bạn phải tạo đường dẫn bằng tay
- Các đường dẫn được giới hạn để kiểm soát

Bạn có thể nhanh chóng phát hiện ra những lợi thế của một trong hai cách sử dụng. Trong một tình huống được kiểm soát, bạn sẽ sử dụng các điểm đến (waypoint). Những thứ này sẽ chỉ đạo AI chính xác nơi di chuyển trong không gian 3D. Điều này là hoàn hảo để định tuyến các vị trí AI trong màn chơi theo ý bạn muốn. Vì vậy, nếu đó là một đơn vị Secret Security trong trò chơi, nó sẽ tuần tra các khu vực bạn chỉ định. Sau đó, bạn có thể loại bỏ người chơi khi tiếp xúc.

Thêm ngẫu nhiên và xác suất

Trong chương này, chúng ta sẽ tập trung vào việc điều chỉnh ví dụ trước để thực hiện tính ngẫu nhiên và xác suất. Vì vậy, chúng ta sẽ tập trung vào những ưu và nhược điểm của tính ngẫu nhiên và xác suất. Tính ngẫu nhiên có thể trùng lặp với xác suất nếu bạn xác định trước kết quả đầu ra. Ví dụ: nếu bạn muốn phát các hoạt ảnh khác nhau mỗi khi người chơi tấn công bằng nắm đấm, bạn có thể có một loạt các hoạt ảnh khác nhau mà bạn chọn ngẫu nhiên. Tuy nhiên, giả sử rằng có năm tùy chọn khả dụng; với điều này, bạn có 20% cơ hội chọn bất kỳ một trong các lựa chọn có sẵn.

Ưu và nhược điểm của tính ngẫu nhiên

Sau đây là các ưu điểm:

- Hầu như không có giới hạn cho đầu ra
- Nó có được sự cân bằng cho các hệ thống

Sau đây là các nhược điểm:

- Bạn phải tạo một bảng đầu ra lớn để tăng độ lệch
- Các đường dẫn được giới hạn để kiểm soát

Ưu và nhược điểm của tính xác xuất

Sau đây là các ưu điểm:

- Bạn có thể xác định trước đầu ra
- Bạn có thể kiểm soát được tần suất đầu ra
- Bạn có thể đưa ra sự cân bằng cho các hệ thống

Sau đây là các nhược điểm:

- Nó không phải là ngẫu nhiên
- Bạn phải xác định bảng đầu ra

Xác suất rất hữu ích khi bạn muốn xác định trước đầu ra. Xác suất cho phép bạn chọn tần suất một đầu ra được đưa ra. Tính ngẫu nhiên không cung cấp cho bạn bất kỳ quyền kiểm soát nào đối với đầu ra. Việc giới thiệu những thứ này có thể thêm một abstract class khác vào trải nghiệm của người chơi.

Giới thiệu sự di chuyển

Chúng ta chỉ đề cập đến chuyển động của những con tốt (pawn) vì cuốn sách này chỉ là phần giới thiệu. Tuy nhiên, nếu bạn quan tâm đến việc di chuyển các loại tốt khác, bạn phải tạo lại function box, nó là component chuyển động. Component chuyển động cho một số loại quân tốt sẽ chịu trách nhiệm lấy cự ly đầu vào và chuyển đổi nó thành gia tốc.

Điều này có ý nghĩa gì đối với AI của ô tô trước tiên là nhận hướng từ điểm này sang điểm khác, dẫn đến hướng vị trí. Sau đó, chúng ta cần tạo hướng vị trí này so với AI của ô tô bằng cách trừ nó khỏi hướng đi của AI ô tô. Điều này dẫn đến một hướng trên một mặt phẳng hai chiều, có thể được áp dụng cho hướng đi của AI trên ô tô.

Bạn cũng có thể làm điều gì đó tương tự trong không gian 3D cho tàu vũ trụ. Điều này có nghĩa là nếu bạn muốn tạo nhện AI không thể di chuyển bằng component chuyển động mặc định, thì bằng cách tạo component chuyển động tùy chỉnh, bạn có thể yêu cầu nhện AI di chuyển đến bất kỳ đâu bạn muốn mà vẫn có thể tích hợp nó vào Behavior Tree.

Cho AI các lựa chọn

Trong chương này, chúng tôi tập trung vào việc tạo AI với nhiều trạng thái bằng cách sử dụng Behavior Tree. Chúng ta cũng đã nói ngắn gọn về việc sử dụng EQS trên chú chó để chọn ngẫu nhiên một khu vực trong vị trí của nó. Ví dụ cụ thể này rất đơn giản, nhưng khi bạn bắt đầu sử dụng các bộ lọc, sức mạnh của EQS sẽ tỏa sáng.

Ví dụ: bạn muốn tiêu diệt tất cả kẻ thù trong phạm vi 500 đơn vị tính từ vị trí của mình và loại bỏ những kẻ không ở trong tầm nhìn. Khi đó, chúng ta sẽ tiêu diệt bất kỳ kẻ thù nào không nằm trong góc nhìn 60 độ. Cuối cùng, chúng ta đã có một nhân vật gần nhất.

Trong Blueprint, điều này có thể được thực hiện, nhưng sau đó nó yêu cầu bạn đặt nó vào Blueprint Library hoặc một node nào đó. Nếu chúng ta thực hiện trong EQS, chúng ta chỉ cần tạo các điều kiện trong giao diện người dùng. Tuy nhiên, việc chia sẻ dễ dàng hơn và không yêu cầu bất kỳ tập lệnh script nào.

Hãy so sánh việc sử dụng EQS và Blueprint để cảm nhận môi trường.

Ưu và nhược điểm khi dùng EQS

Sau đây là ưu điểm:

- Không yêu cầu blueprint và việc chỉ định các điều kiện sẽ trực quan hơn.

Sau đây là khuyết điểm:

- Nó giới hạn các tùy chọn trong giao diện người dùng

Ưu và nhược điểm khi dùng Blueprint

Sau đây là ưu điểm:

- Có sự linh hoạt tối đa để thực hiện

Sau đây là nhược điểm:

- Bạn phải tạo nó từ đầu
- Khó tái sử dụng hơn

EQS là một công cụ rất mạnh khi mục đích của công cụ được hiểu đầy đủ. Nó có khả năng truy vấn nhanh chóng và chính xác một cảnh mà không cần phải xử lý bất kỳ blueprint nào. Khi được ghép nối với blueprint, bạn có thể mở ra mục đích thực sự của nó. Bạn có thể tạo các giao diện, giao diện này có thể được sử dụng trong tính toán EQS và nhận được kết quả mong muốn, chẳng hạn như nơi trống an toàn nhất để AI chạy ẩn nấp.

Cách mà AI nhìn và cảm nhận thế giới?

Chúng ta đã giải thích nên sử dụng những component nào để AI của chúng ta cảm nhận được những con tốt (pawn) khác trên thế giới. Chúng ta có thể tùy chỉnh cài đặt cho hình ảnh và âm thanh. Sau đó, hệ thống nhận thức sẽ trả về bất kỳ con tốt (pawn) nào phù hợp với các tiêu chí này với mỗi lần cập nhật.

Chúng ta có khả năng có thể sử dụng những con tốt (pawn) mà chúng ta cảm nhận được nhiều hơn là tấn công. Nếu chúng ta cũng sử dụng nó để tính toán nơi trú ẩn tốt nhất, AI của chúng ta sẽ phản ứng với vị trí của chúng ta. Sau đó, việc đặt các tiêu chí nhất định cho phép chúng ta báo cho AI biết khi nào là thời điểm tốt nhất để vào hoặc rời khỏi chỗ nấp. Sau đó, thêm các tiêu chí bổ sung sẽ cho phép kiểm soát hành vi di chuyển vào và ra.

Nâng cấp thêm cho việc di chuyển của AI

Chuyển động nâng cao giới thiệu các hành vi đổ xô. Loại hành vi di chuyển này thường thấy ở các đàn chim hoặc đàn cá. Có nhiều hành vi chuyển động hơn, chẳng hạn như xếp hàng, chẳng hạn chỉ yêu cầu một đơn vị đi qua một cửa tại một thời điểm. Hành vi của đội là tuyệt vời khi bạn có các đơn vị bảo vệ lẫn nhau.

Ưu điểm của việc sử dụng hành vi chuyển động là nó không yêu cầu AI của bạn đưa ra quyết định bổ sung. Điều này cũng cho phép bạn lấy một số tính toán ra khỏi cây và tính toán nó trên con tốt.

Ưu và nhược điểm của việc dùng Behavior Tree

Sau đây là ưu điểm:

- Nó dễ dàng mở rộng
- Nó đồng bộ hóa các đối tượng Blackboard

Sau đây là nhược điểm:

- Thiết kế chỉ ra luồng thực thi

Ưu và nhược điểm dùng blueprint cho AI

Sau đây là ưu điểm:

- Có ít công việc hơn cho nhiều kết quả hơn
- Không yêu cầu AIController

Sau đây là nhược điểm:

- Không truy cập vào EQS

Sử dụng các tập lệnh blueprint bên ngoài cây Behavior Tree để cung cấp thêm thông tin cho cây BT, bạn có thể khắc phục một số trở ngại của luồng thực thi trong Behavior Tree. Ví dụ:

nếu bạn yêu cầu AI di chuyển đến một địa điểm, bạn sẽ không thể kiểm tra xem có kẻ thù nào được nhìn thấy trong khi di chuyển hay không trừ khi bạn sử dụng dịch vụ. Sau đó, bạn phải hỏi liệu dịch vụ này sẽ hữu ích cho nhánh này cụ thể hay cho toàn bộ cây Behavior Tree.

Tóm tắt

Tôi hy vọng tôi có thể trình bày mọi thứ để giúp bạn tự tin bắt đầu tiếp cận AI trong trò chơi trong Unreal Engine 4. Khám phá các ý tưởng khác nhau ở đây và khơi dậy trí tò mò của bạn khi khám phá AI trong trò chơi. Chủ đề này có vẻ đáng sợ, nhưng nếu bạn đã làm theo mọi thứ trong cuốn sách này, thì bạn đã sẵn sàng để bắt đầu!

Tôi đã rất vui khi chứng minh được các component AI khác nhau có sẵn trong Unreal Engine 4. Ngoài ra bạn có thể tham khảo thêm các chỉ dẫn có sẵn trên mạng để tăng thêm kiến thức của bạn đối với công cụ Unreal Engine tuyệt vời.