Course - Lập trình AI cho Unreal Engine

Tạo AI Căn Bản

Giúp bạn từng bước tạo AI đầu tiên của mình và nói về các kỹ thuật mà chúng tôi sẽ trình bày trong quá trình thực hiện. Chúng ta sẽ đi sâu vào Unreal Engine 4, sử dụng các thành phần cơ bản cần thiết để tạo một trạng thái duy nhất với chuyển động ngẫu nhiên cho AI của bạn.

Tags: Tạo AI Căn Bản

Trong chương này, chúng ta sẽ từng bước tạo AI đầu tiên của mình và nói về các kỹ thuật mà chúng ta sẽ chứng minh trong quá trình thực hiện. Vì vậy, hôm nay, chúng ta sẽ đi sâu vào Unreal Engine 4 bằng cách sử dụng các thành phần cơ bản cần thiết để tạo một trạng thái duy nhất với chuyển động ngẫu nhiên cho AI của bạn. Sau đó, chúng ta sẽ xem xét những gì chúng ta đã thực hiện, những thay đổi chúng ta có thể thực hiện và nhược điểm của các kỹ thuật đã trình bày. Chương này sẽ bao gồm:

- Thiết lập dự án của chúng ta
- Tạo AIController
- Gửi hướng dẫn vào quân tốt Pawn bằng AIController
- Tạo các lệnh script blueprint nhỏ để hỗ trợ điều hướng

Bạn có thể download dự án mẫu full source, tại đây.

Mục tiêu

Mục tiêu của chúng ta cho chương này là đặt một nhân vật AI vào trong màn chơi có blueprint hướng dẫn nó di chuyển ngẫu nhiên và vô thời hạn. Chúng ta sẽ trình bày nhiều kỹ thuật trong suốt chương này để hiểu rõ một số kỹ thuật AI cơ bản thường có trong các tài liệu. Những kỹ thuật này được liệt kê như sau:

- Đầu tiên, chúng tôi muốn đặt một nhân vật AI, Hero, vào trong màn chơi có blueprint hướng dẫn nhân vật đó di chuyển ngẫu nhiên và vô thời hạn. Chúng ta sẽ đạt được điều này bằng cách trước tiên tạo một dự án ThirdPerson mới và đặt tên cho nó một cách thích hợp. Sau đó, chúng ta sẽ sử dụng Pawn mặc định được cung cấp từ nội dung mẫu làm bot. Chúng ta sẽ tạo một AIController để điều khiển con tốt của chúng ta. Sau đó, chúng ta sẽ cung cấp cho AIController các hướng dẫn để di chuyển bot của chúng tôi một cách ngẫu nhiên và vô thời hạn.
- Thứ hai, chúng ta muốn làm cho nhân vật AI đi theo một số đường dẫn cơ bản. Ví dụ: chúng ta sẽ để AI di chuyển dọc theo các bức tường theo một hướng. Chúng ta có thể lấy dự án hiện tại của mình và sửa đổi AIController bằng các hướng dẫn mới. Từ đó, AI của chúng ta sẽ di chuyển dọc theo các bức tường theo một hướng vô thời hạn.
- Thứ ba, chúng ta muốn tạo một nhân vật AI mới là kẻ thù và sẽ đuổi theo nhân vật AI đầu tiên mà chúng ta đã tạo ra—tức là Hero.

> Chúng ta sẽ phải thực hiện các thay đổi bổ sung đối với Hero để cung cấp cho nó khả năng chạy trực tiếp khỏi Enemy.

Enemy sẽ chỉ được hướng dẫn di chuyển về phía Hero mỗi giây.

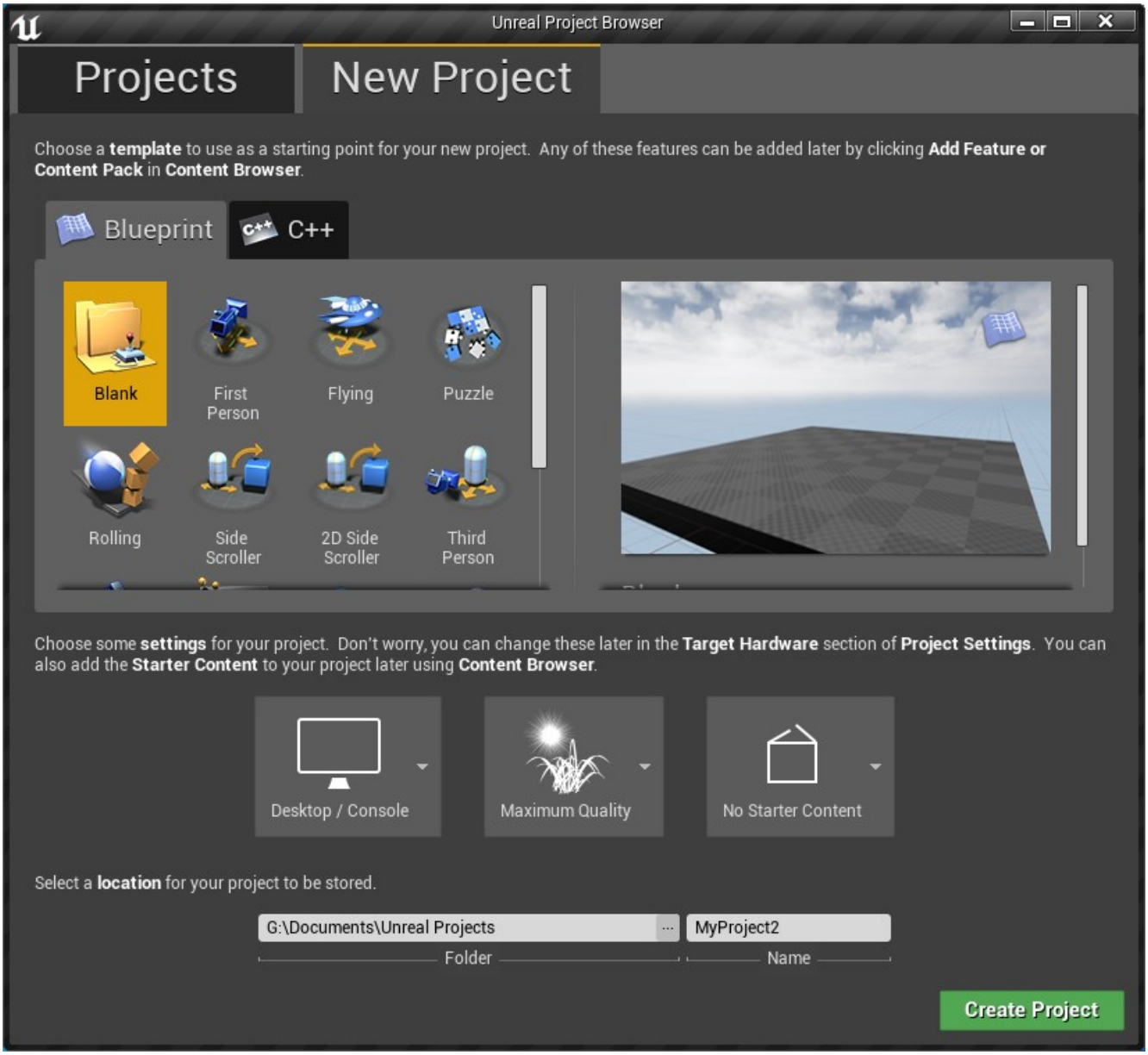
Thiết lập dự án

Hãy mở Unreal Engine 4! Chúng ta sẽ bắt đầu với quy trình đầu tiên để tạo một dự án mới.

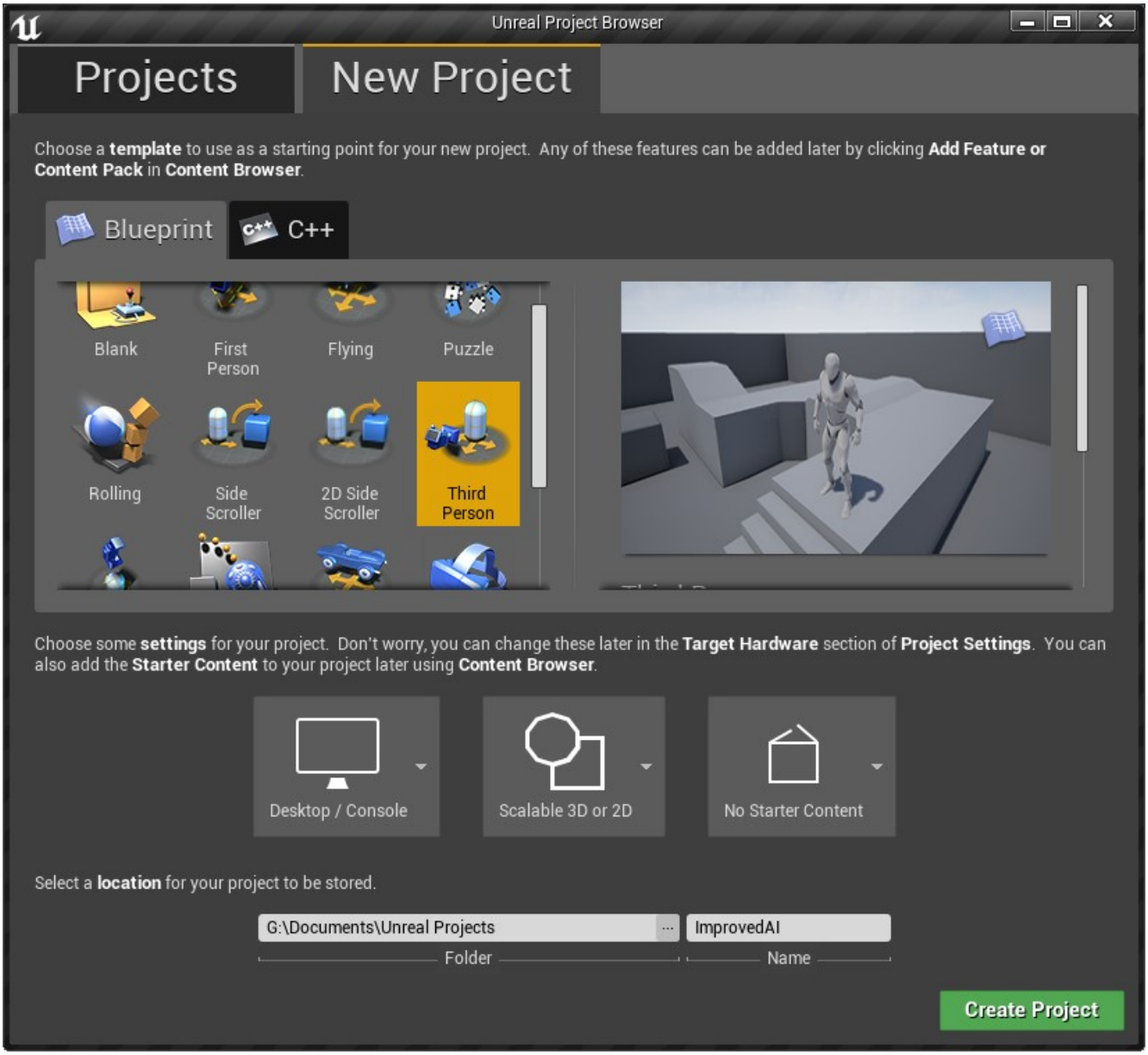
> **Chú ý:** Chúng tôi sẽ sử dụng Unreal Engine 4.6.0 trong suốt cuốn sách này. Các hướng dẫn có thể khác nhau cho mỗi phiên bản. Chúng tôi sẽ trình bày ý tưởng đằng sau các hành động của mình khi chúng tôi chứng minh chúng bằng Unreal Engine 4; vì vậy, hy vọng rằng bạn sẽ có thể dịch các hướng dẫn khi bạn thấy phù hợp.

Ở đây, chúng ta sẽ sử dụng dự án mẫu Third Person, mẫu này cho phép chúng ta dễ dàng quan sát cách nhân vật di chuyển trong môi trường. Thực hiện các bước sau:

Hãy đi đến cửa sổ **New Project**.



Hãy chọn dự án **Third Person** Blueprint.



Hãy đặt tên dự án của bạn theo ý muốn, ở đây tôi sẽ đặt tên nó là *ImprovedAI*. Sau đó nhấn nút **Create Project** nằm ở vị trí góc phải dưới cùng của cửa sổ.

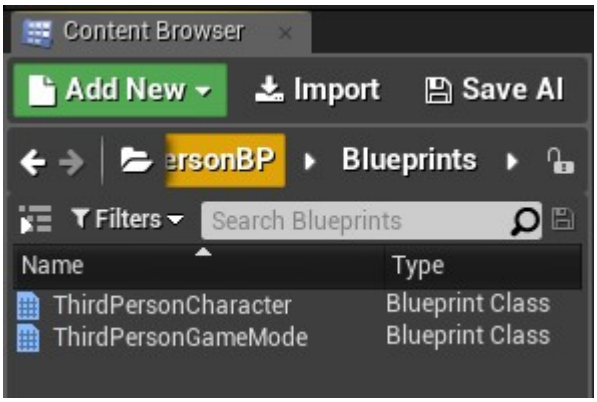
Môi trường

Mặc dù chúng tôi đang sử dụng mẫu Third Person Blueprint, những kỹ thuật này cũng có thể được sử dụng trên các mẫu khác. Bạn phải điều chỉnh những gì bạn học được ở đây. Nó đang nói, điều bạn bắt đầu hiểu là những kỹ thuật này là công cụ. Việc hiểu cách tạo trạng thái, component cảm quan, component điều hướng, v.v. thường được coi là giống nhau, nhưng cái gì được sử dụng và cách sử dụng nó là do môi trường AI quyết định.

Yêu cầu cần thiết

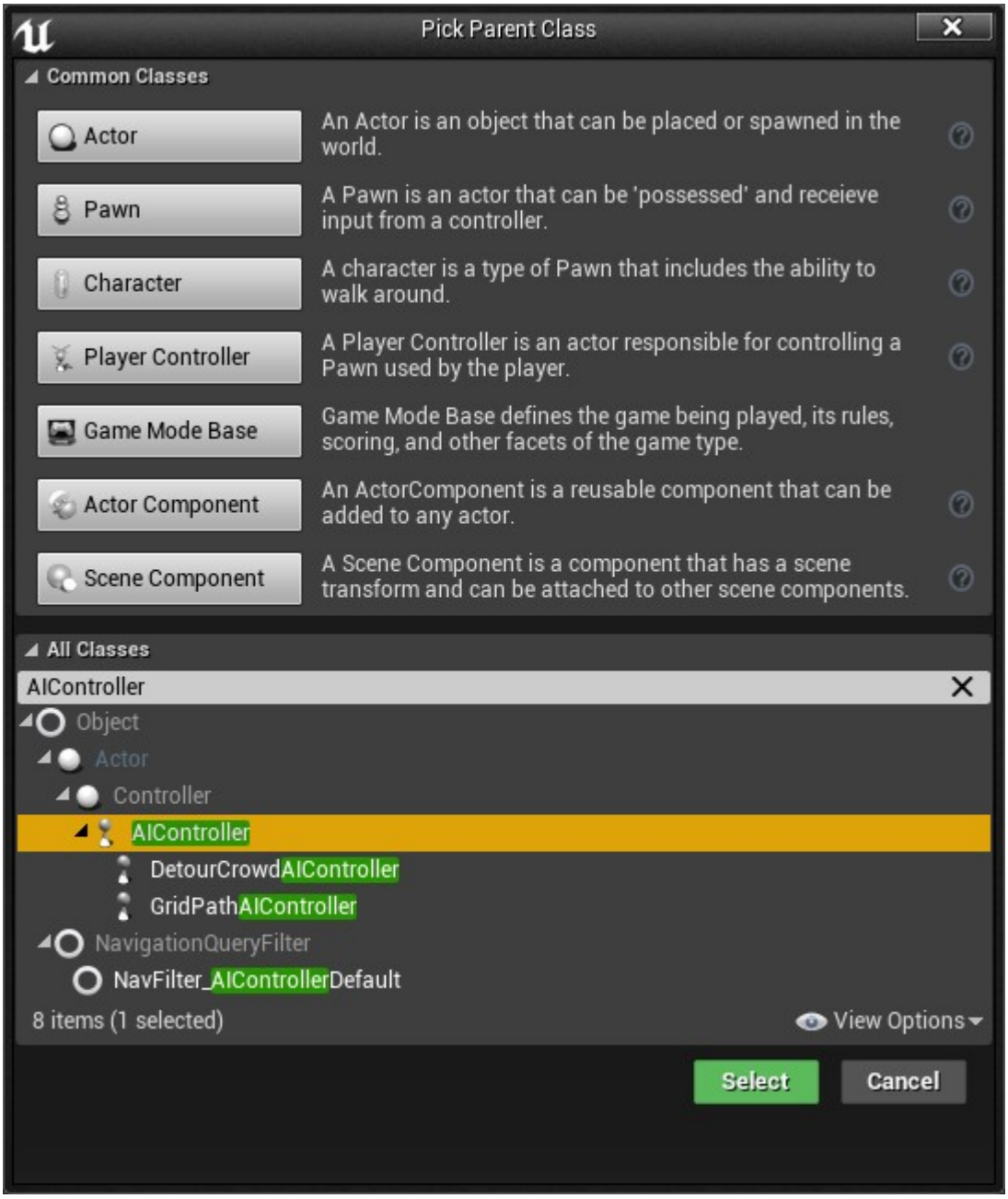
Lưu ý rằng các cửa sổ và tên chức năng của tôi là khác nhau. Tôi sẽ hướng dẫn bạn qua các cài đặt của tôi để bạn có thể có cùng một thiết lập như tôi. Dưới đây là các bước để thực hiện:

Chuyển đến thư mục *Blueprints* bên trong dự án.

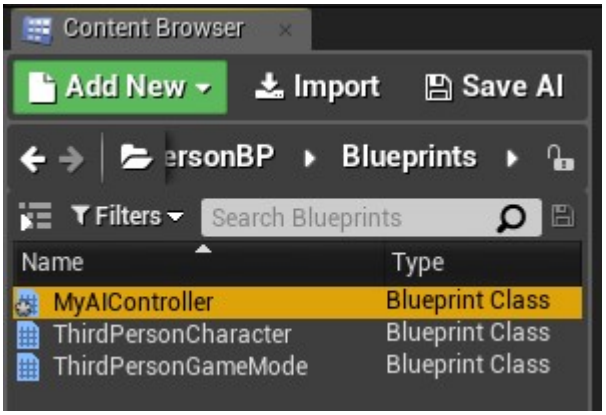


Bây giờ chúng ta sẽ chiếm lấy Pawn của chúng ta. Click chuột phải lên Content Browser để mở context-menu và nhấp vào tùy chọn **Blueprint Class** để tạo một blueprint trong từ popup **Pick Parent Class**.

Trong popup được hiện lên, chúng ta sẽ tạo class **AIController** của mình. Chuyển đến **Custom Class** và nhập *AIController*. Chọn nó và sau đó bấm vào Select, như thể hiện trong ảnh chụp màn hình sau:



Nó sẽ tạo ra một blueprint, và chúng ta sẽ đặt tên nó là *MyAIController*.

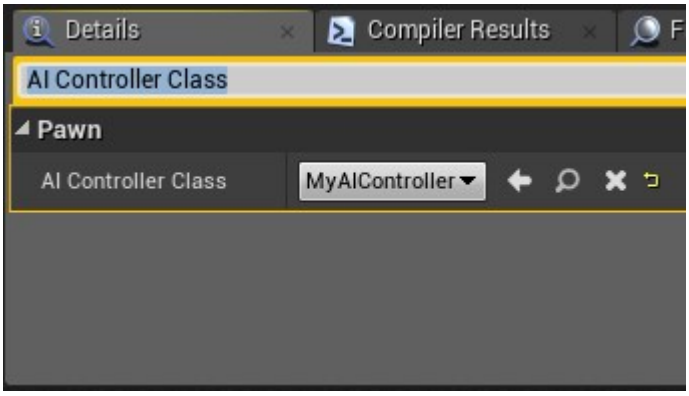


Sử dụng tài nguyên AIController mới tạo

Bạn có bao giờ để ý cách một người chơi có thể trở thành bất kỳ nhân vật nào họ muốn không? Đây là hệ thống phân cấp tạo ra con tốt Pawn và Controller. Controller là thứ mà người chơi thừa hưởng sau khi chờ đợi một thời gian trong sảnh trò chơi. Nó được sử dụng để quản lý đầu vào và kết nối từ đầu phát. Class này đi kèm với các chức năng bổ sung để giúp điều hướng bot và khả năng gán một Bahavior Tree cho Controller. Trong phần trình diễn demo này, chúng ta sẽ đề cập đến một số điều cơ bản của class AIController.

Gán nhượng class AIController

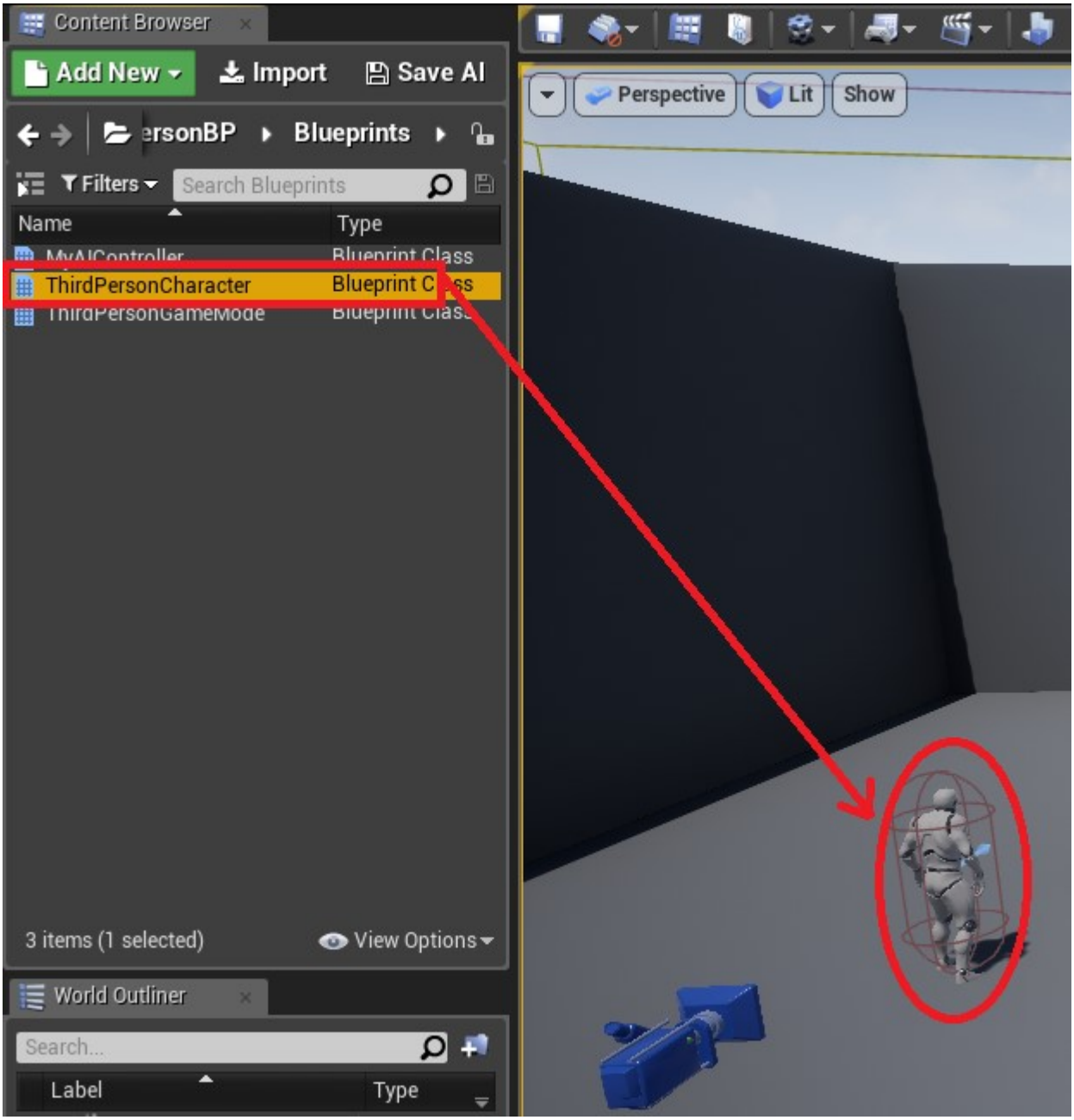
Bây giờ chúng ta có những thứ cần thiết để tạo AI, chúng ta sẽ gán class *MyAIController* cho *ThirdPersonCharacter* cơ sở. Để làm được như vậy, hãy chuyển đến panel **Details** trong blueprint của *ThirdPersonCharacter*. Tìm kiếm *AIController Class* và đặt nó thành *MyAIController*, như thể hiện trong ảnh chụp màn hình sau:



Khi một character không bị chiếm hữu, nó sẽ tự động bị chiếm hữu bởi *MyAIController*. Vì vậy, với thay đổi mà chúng ta vừa thực hiện, class **AIController** mặc định sở hữu blueprint *ThirdPersonCharacter* của chúng ta sẽ là *MyAIController*.

Đặt quân tốt Pawn

Giả sử bạn hiểu một việc làm đơn giản này! Hãy đặt con tốt mới được thiết lập của chúng ta vào thế giới tươi đẹp tươi sáng bằng cách kéo và thả nó từ Content Browser:



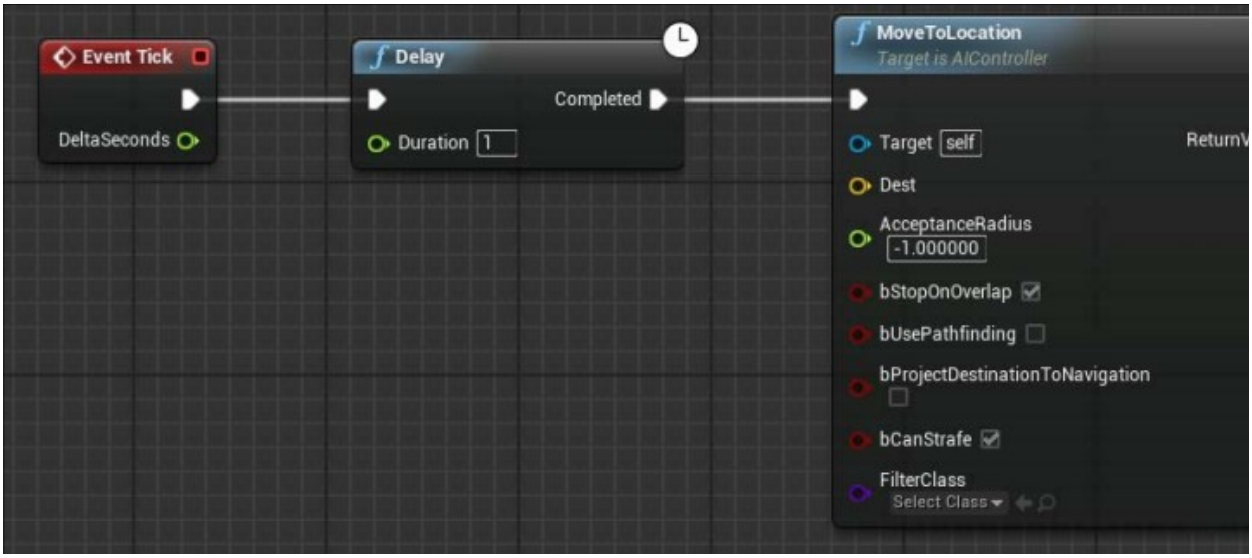
Hãy kéo thả blueprint ThirdPersonCharacter vào trong màn chơi để tạo ra một quân tốt Pawn.

> **Chú ý:** Có một hàng dài những người bị mất việc vì tai nạn và những rủi ro khác; nếu bạn muốn tham gia cùng họ, vui lòng bỏ qua bước này. Nếu không, hãy tiến đến **File**

| **Save All** bất cứ khi nào bạn thực hiện một số thay đổi mới quan trọng đối với màn chơi.

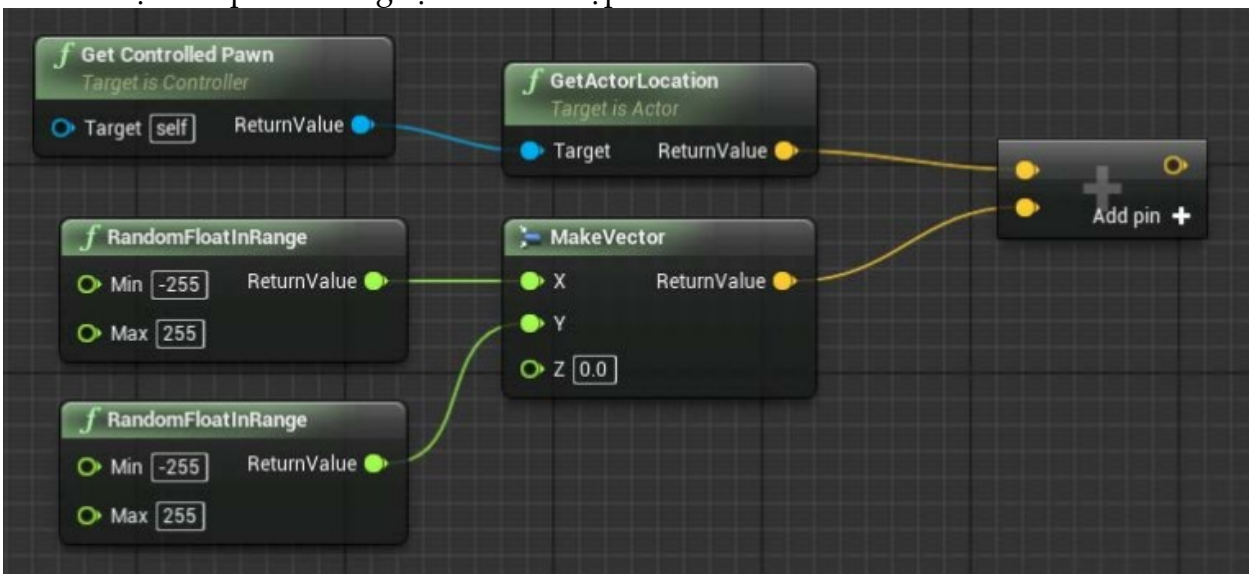
Gửi đi các hướng dẫn

Bây giờ chúng ta đã thiết lập và lưu dự án của mình—hoặc chưa lưu cho những người bạn dũng cảm của chúng ta—hãy chuyển sang phần thú vị: Blueprint! Chúng ta hãy xem các bước sau để gửi hướng dẫn:

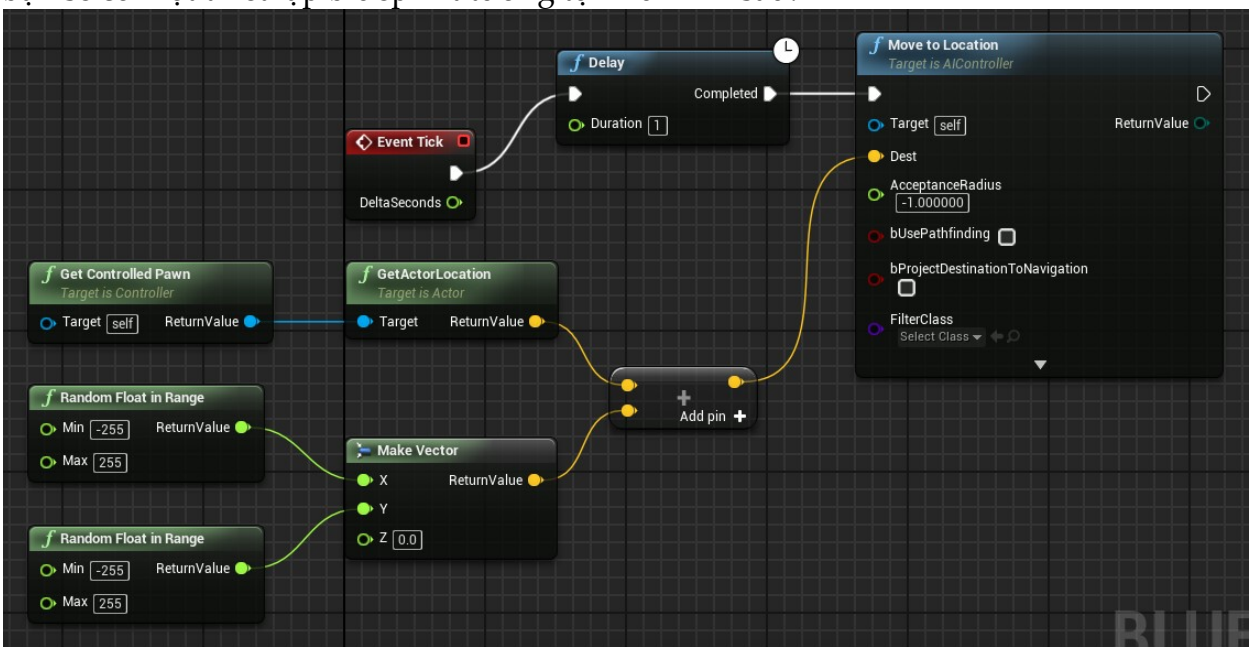
1. Mở blueprint *MyAIController* của bạn đang nằm trong **Content Browser** và mở qua tab **EventGraph**: kế hoạch sẽ là để bot của chúng ta di chuyển ngẫu nhiên. Nó sẽ được xây dựng theo kiểu đơn giản, vì vậy chúng ta sẽ thực hiện tính toán bằng tay.
2. Trước tiên, hãy tạo một node **Event Tick** sẽ được kích hoạt trong mọi khung hình khi trò chơi chạy.
3. Chúng tôi sẽ thêm node **Delay** để nhận tín hiệu từ **Event Tick** để đặt chân **Duration** thành **1**.
4. Tiếp theo, chúng ta sẽ thêm một node **Move To Location**, node này sẽ báo hiệu cho class *MyAIController* nhằm yêu cầu con tốt Pawn của nó di chuyển đến điểm đích đã chỉ định.
5. Vì chúng ta không sử dụng tính năng **Path Finding** vào thời điểm này, nó là chủ đề mà chúng ta sẽ đề cập sau, hãy chuyển đến node **MoveToLocation** và đặt chân tham số **bUsePathFinding** thành *false* hoặc không chọn nó.
   EventGraph cho class *MyAIController* của bạn sẽ trông giống như ảnh chụp màn hình sau:
   

6. Với node **MoveToLocation** đã sẵn sàng, bây giờ chúng ta cần cung cấp cho nó một vị trí ngẫu nhiên. Chúng tôi sẽ lấy vị trí hiện tại của con tốt được kiểm soát, tạo một vector bổ sung có giá trị ngẫu nhiên từ -255 đến 255 cho các biến **X** và **Y**, để **Z** bằng **0,0**. Sau đó, chúng ta sẽ thêm vị trí từ con tốt được kiểm soát vào vector mà chúng ta vừa tạo. Bạn

nên có một blueprint tương tự như thiết lập sau:



7. Hãy hoàn thiện nó và di chuyển blueprint vị trí ngẫu nhiên mới này sang node **MoveToLocation** mà chúng ta đã thiết lập trước đó. Bây giờ, hãy kết nối kết quả của phép cộng hai vector với đích của node **MoveToLocation**. Khi điều này được thực hiện, bạn sẽ có một thiết lập blueprint tương tự như hình sau:



Compile và **Save** tất cả blueprint *MyAIController*.

Mẹo nhỏ về MoveToLocation

Dưới đây là một số mẹo cơ bản về **MoveToLocation**:

- **AcceptanceRadius**: Điều này cho phép bạn tăng bán kính chấp nhận được cho một

nước đi đã hoàn thành. Giả sử rằng có một kẻ thù đang cầm kiếm muốn tấn công người chơi. Tùy chọn **AcceptanceRadius** sẽ giúp bạn xác định khoảng cách mà kẻ thù này nên đứng cách xa mục tiêu của hắn—lý tưởng là 1 mét—và sau đó thực hiện hoạt ảnh tấn công để vung kiếm.
- **bStopOnOverlap**: Nó yêu cầu bot của bạn dừng lại nếu nó trùng với điểm thay vì đi chính xác vào điểm đó. Nó sẽ xem xét bán kính của lưới va chạm được gắn vào bot.
- **bUsePathFinding**: Nếu tùy chọn này được chọn, bot sẽ sử dụng tùy chọn **NavMesh** để tìm đích đến của nó. Nếu không được chọn, bot sẽ chỉ di chuyển theo đường thẳng đến đích mà không tính đến bất kỳ chướng ngại vật nào. Điều này giúp tiết kiệm hiệu suất trong một số tình huống.
- **bProjectDestinationToNavigation**: Nó sẽ chiếu vị trí trên dữ liệu điều hướng trước khi sử dụng. Điều này giúp xác thực vị trí của actor mục tiêu—nghĩa là vị trí đó có tồn tại trên khu vực có thể chơi được hay không.
- **bCanStrafe**: Nó sẽ xác định liệu AI có thể di chuyển theo đường chéo trên **NavMesh** hay không.
- **FilterClass**: Nó cho phép bạn sử dụng **AreaClass**, đây là một thành phần điều hướng khác ảnh hưởng đến điều hướng của AI. Điều này ảnh hưởng đến những thay đổi như loại trừ hoặc độc quyền đi vào các khu vực của **NavMesh** và làm thay đổi chi phí điều hướng.

Xem xét lại tiến trình hiện tại

Bạn có thể lau mồ hôi trên trán; công việc khó khăn vẫn chưa bắt đầu. Vì vậy, những gì chúng ta đã làm cho đến nay?

- Chúng ta đã thiết lập dự án AI của mình
- Chúng ta đã thiết lập Pawn với AIController mới của mình
- Chúng ta đã gửi hướng dẫn cho Pawn của mình bằng AIController

Chúng ta đã đi được nửa chặng đường. Thiết lập đơn giản này cho phép chúng ta đưa tất cả các hướng dẫn của mình vào AIController, mà đang chiếm hữu Pawn mà chúng ta đã tạo từ tài nguyên trong **Content Browser**. AIController đã được gán cho những con tốt Pawn, có nghĩa là nhiều con tốt có thể chia sẻ cùng một AIController.

Như chúng ta có thể thấy, AI của chúng ta hiện chạy vô thời hạn. Hoàn hảo! Hãy chuyển sang phần thứ hai của chương này!

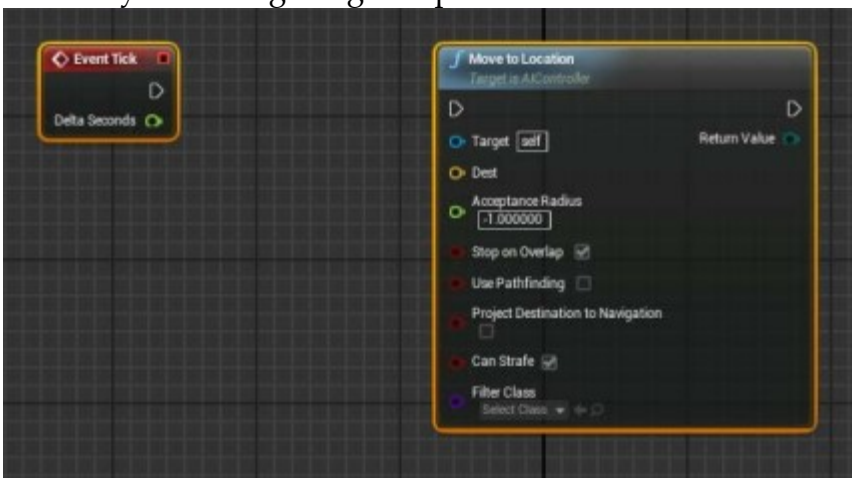
Thêm vài thử thách khác

Bây giờ, chúng tôi sẽ thêm các tia thẳng dò tìm vào nhân vật AI. Trong phần trình diễn demo của chúng ta, chúng ta sẽ sử dụng việc dò tìm để phát hiện bức tường phía trước quân tốt

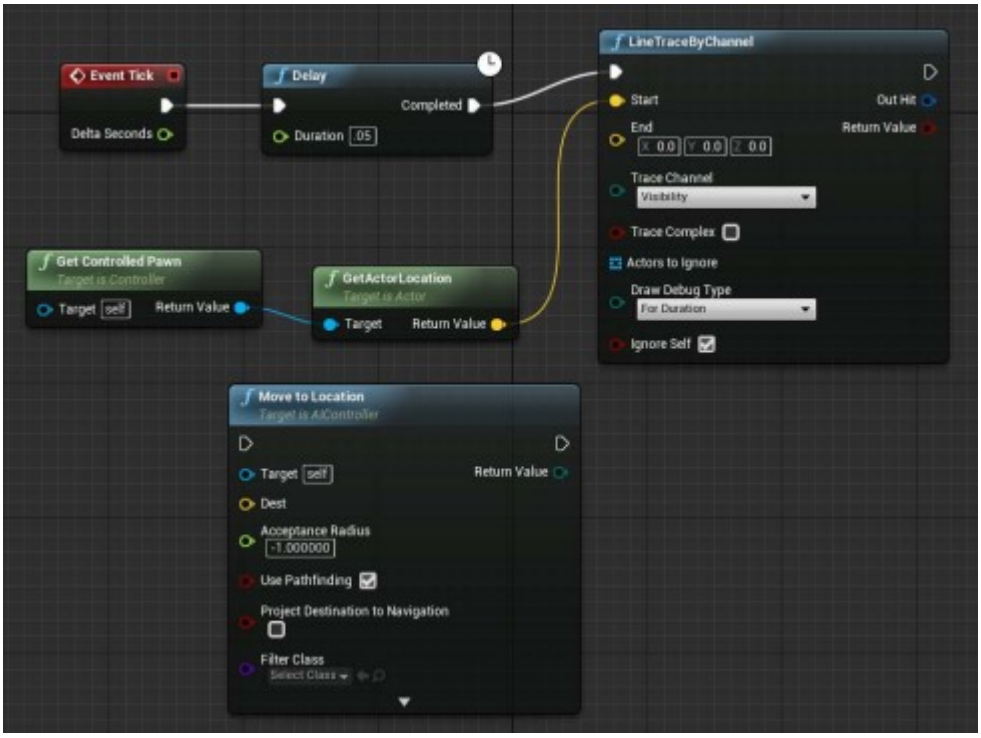
Pawn. Các ví dụ khác về việc sử dụng dấu vết trong AI bao gồm kiểm tra Đường ngắm (Line of Sight), xoay bề mặt và nhận các actor ở gần đó.

Hãy quay lại Level Editor của Unreal Engine và tập trung vào trong **Content Browser**. Thực hiện các bước sau:

- Đổi tên blueprint *MyAIController* thành *Hero* của chúng ta; Nó sẽ đóng vai trò là người chơi trong kịch bản này.
- Mở blueprint *Hero* của chúng tôi và chuyển qua tab **EventGraph**.
- Bây giờ, hãy xóa mọi node ngoại trừ các node **Event Tick** và **Move to Location**. Chúng ta sẽ thay thế chúng bằng blueprint mới khác:



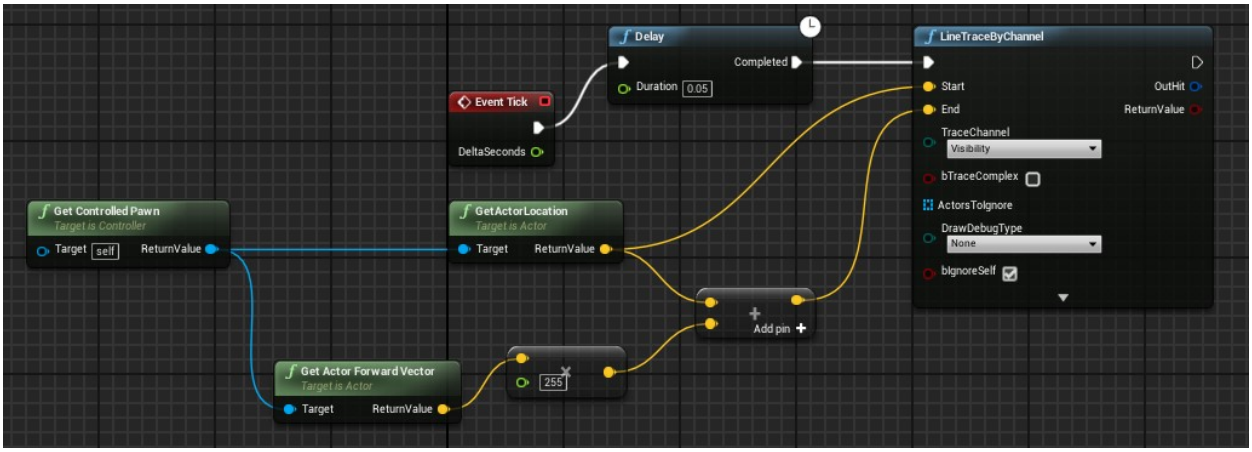
- Kéo từ chân thực thi trên node **Event Tick** và phát biểu node **Delay**.
- Đặt giá trị **Duration** thành **0,05** để nó cập nhật nhanh lên một chút.
- Bây giờ, chúng ta phải lấy vị trí từ con tốt Pawn để tạo tia thẳng dò tìm. Chúng ta cũng sẽ sử dụng vector đối mặt ngay với con tốt Pawn ở ngay chính vòng quay hiện tại của con tốt Pawn khi phát hiện va chạm phía trước.
- Nhấp chuột phải vào **EventGraph** và tìm kiếm **Get Controlled Pawn**.
- Từ chân **Return Value** của node **Get Controlled Pawn**, hãy kéo node **Get Actor Location**.
- Từ chân **Return Value** của node **GetActorLocation**, hãy kéo một vector rồi thả nó vào một vùng trống.
- Tìm kiếm **LineTraceByChannel**, nằm trong danh mục **Collision**, như minh họa trong ảnh chụp màn hình sau:



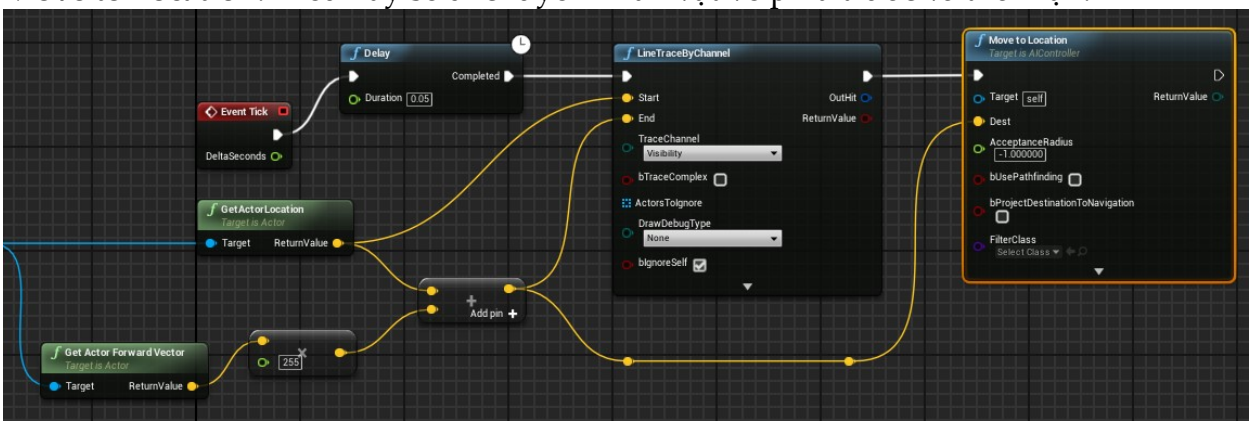
Dò tìm

Chúng ta sẽ sử dụng cái này để theo dõi từ vị trí của người chơi đến **255** đơn vị phía trước nhân vật. Nếu bất cứ thứ gì va chạm với tia dò, chúng ta sẽ đối mặt với con tốt Pawn dựa ngay trên vòng quay của con tốt Pawn. Kiểm tra đơn giản này sẽ đủ để làm cho bot của chúng ta chạy dọc theo các bức tường vô thời hạn; vì vậy, hãy thực hiện các bước sau:

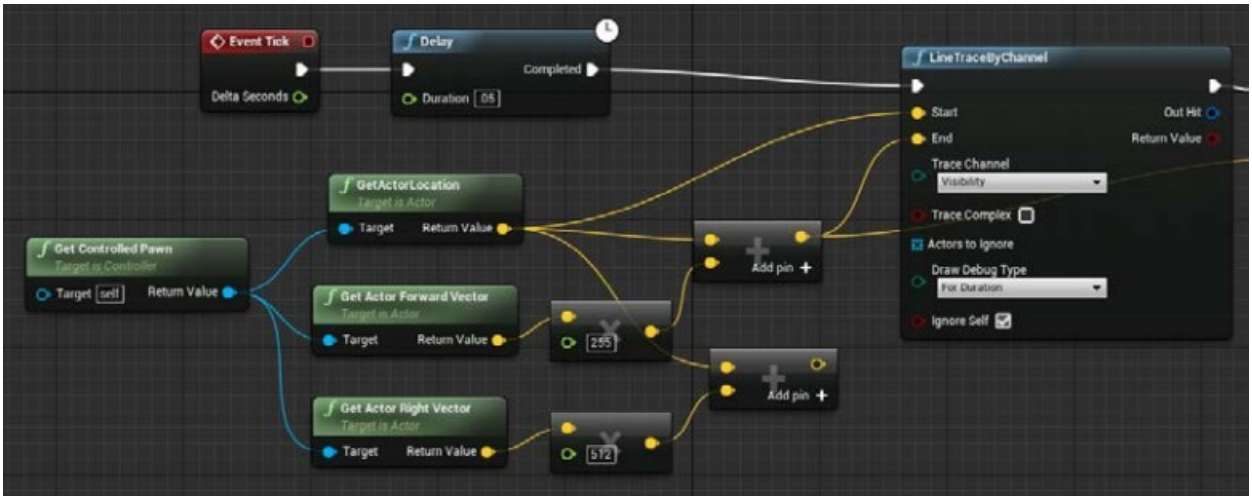
1. Kéo chân **Return Value** từ **GetActorLocation** và thả nó xuống. Sau đó, tìm kiếm **Vector + Vector**.
2. Bây giờ chúng ta cần chọn **Get Controlled Pawn** và kéo ra node mới **Get Actor Forward Vector** từ đó. Cái này chứa thông tin vector đi theo hướng ở phía trước con tốt của chúng ta.
3. Tiếp đó, chúng ta sẽ nhân **Return Value** với **255**. Đây là vector chúng ta muốn thêm vào vị trí của actor. Điều này dẫn đến việc thêm 255 đơn vị theo hướng phía trước vị trí hiện tại của quân tốt Pawn.
4. Bây giờ, chúng ta cần lấy kết quả vào việc bổ sung chân **End** cho node **LineTraceByChannel**. Nó sẽ dò tìm một cách trực tiếp từ phía trước quân tốt Pawn:



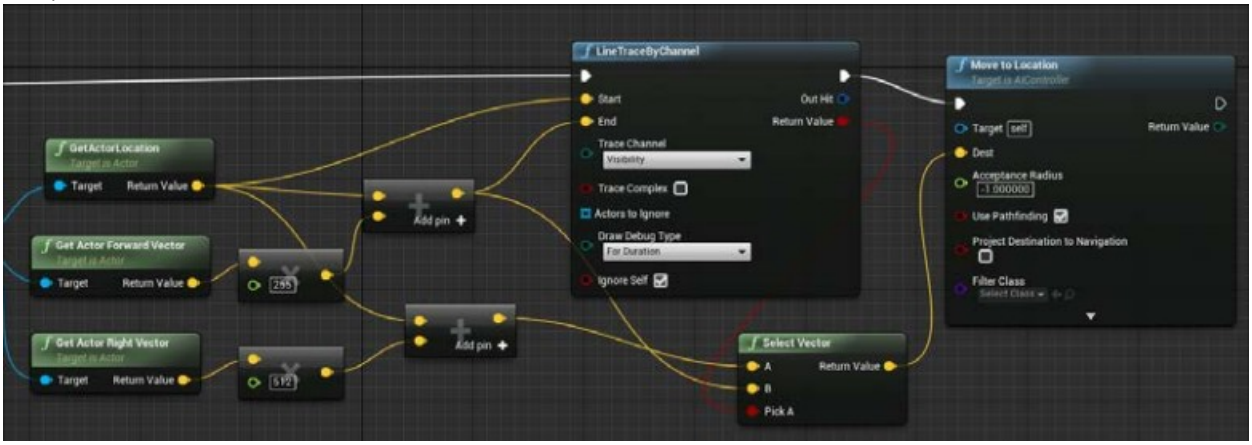
5. Kéo lại từ nút **Vector + Vector** và lần này chúng ta sẽ cắm nó vào chân **Dest** cho node **Move to Location**. Điều này sẽ di chuyển nhân vật về phía trước vô thời hạn:



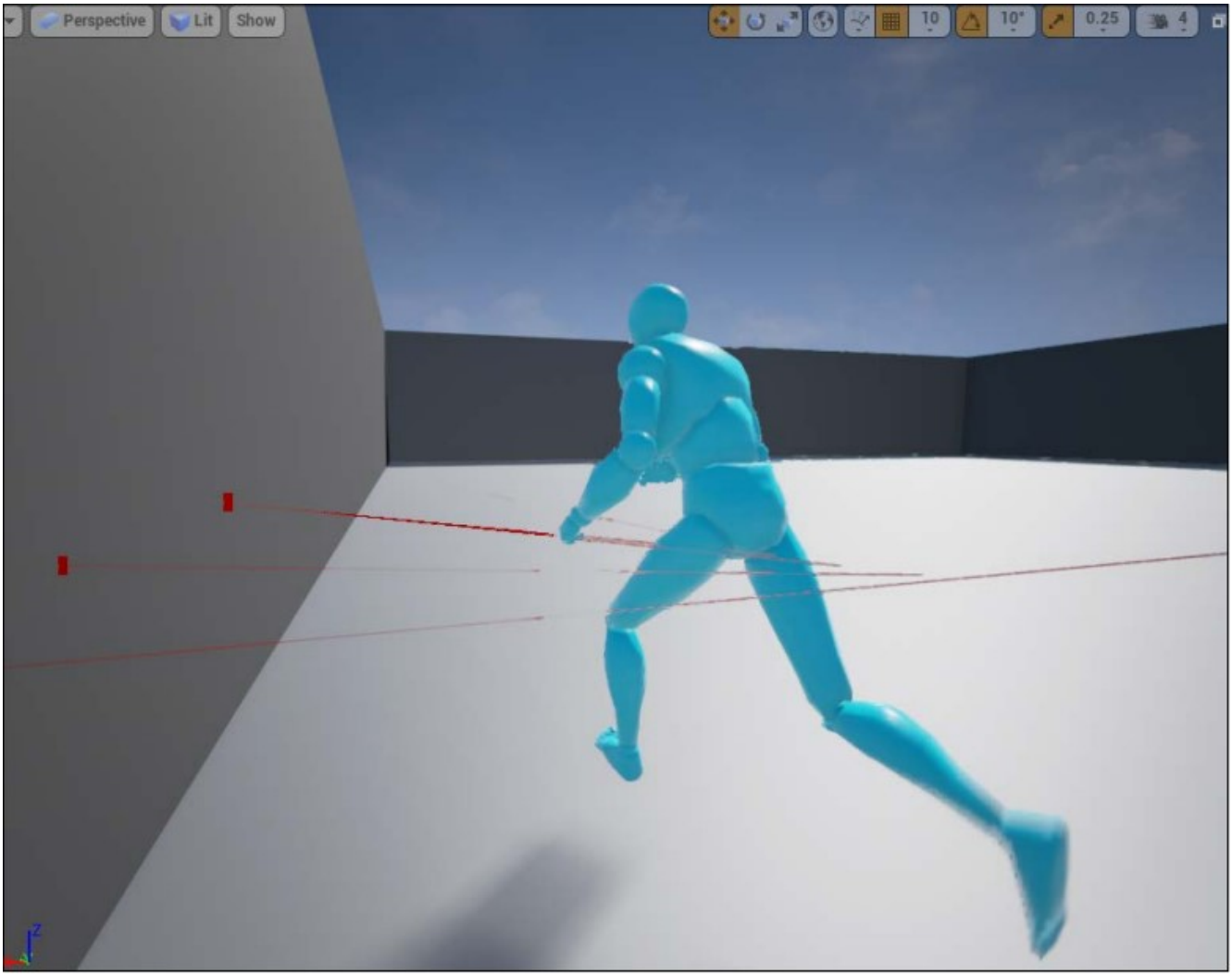
6. Hãy thử trải nghiệm lệnh script mới xem sao! Nhấn **Simulate** và nhìn nhân vật của bạn tiến về phía trước vô tận! Vấn đề duy nhất là nó chưa thể tránh được các bức tường. Chúng tôi sẽ thay đổi điều này bằng cách giới thiệu một node **Select Vector** sẽ chọn một hướng khác khi phát hiện va chạm bởi **Line Trace**:
7. Một lần nữa, chúng ta sẽ lấy từ node **Get Controlled Pawn** và tìm kiếm node **Get Actor Right Vector**.
8. Từ nút này, chúng ta sẽ kéo vector và nhân nó với **512**.
9. Chúng ta sẽ thêm nó vào vị trí hiện tại của quân tốt Pawn. Vì vậy, tương tự như những gì chúng ta đã làm trước khi tiếp tục, chúng ta sẽ thêm **512** đơn vị vào ngay vị trí hiện tại của quân tốt Pawn:



10. Nhấp chuột phải vào phần **EventGraph** và tìm kiếm node **Select Vector**.
11. Từ nút **LineTraceByChannel**, hãy kéo **Return Value** và kết nối nó với node **Select Vector**. Sau đó, chọn **A**.
12. Bây giờ, kéo từ đúng vị trí đến **A** của node **Select Vector**.
13. Sau đó, kéo từ vị trí chuyển tiếp đến **B** của node **Select Vector**.
14. Chọn node **Select Vector** và kéo từ **Return Value** vào chân **Dest** của **Move to Location**.



Bây giờ, hãy xem nó trông như thế nào trong trò chơi:



Nó dường như hoạt động hoàn hảo! Điều này sẽ tạo ra sự săn đuổi cho phần bổ sung tiếp theo cho dự án này.

Đánh giá lại tiến trình hiện tại

Vậy bây giờ, chúng ta đã tạo ra nhân vật *Hero* của chúng ta mà trong trường hợp người không phải là anh hùng thì anh ta sẽ chạy vô thời hạn, nhưng đó là một câu chuyện khác. Đây là sự thay đổi hiện tại:

- Chúng ta đã cập nhật hướng dẫn cho AI.
- Chúng ta đã minh họa một component cảm quan cơ bản.

Bây giờ, chúng ta tiếp tục chuyển sang tạo AI cho kẻ địch.

Logic của quân địch

Kẻ địch cần có khả năng tìm và chạy về phía *Hero*. Chúng ta sẽ thực hiện điều này bằng cách tìm kiếm nhân vật *Hero*, qua việc tính toán các hướng khác nhau và làm kẻ địch đối mặt về phía *Hero*.

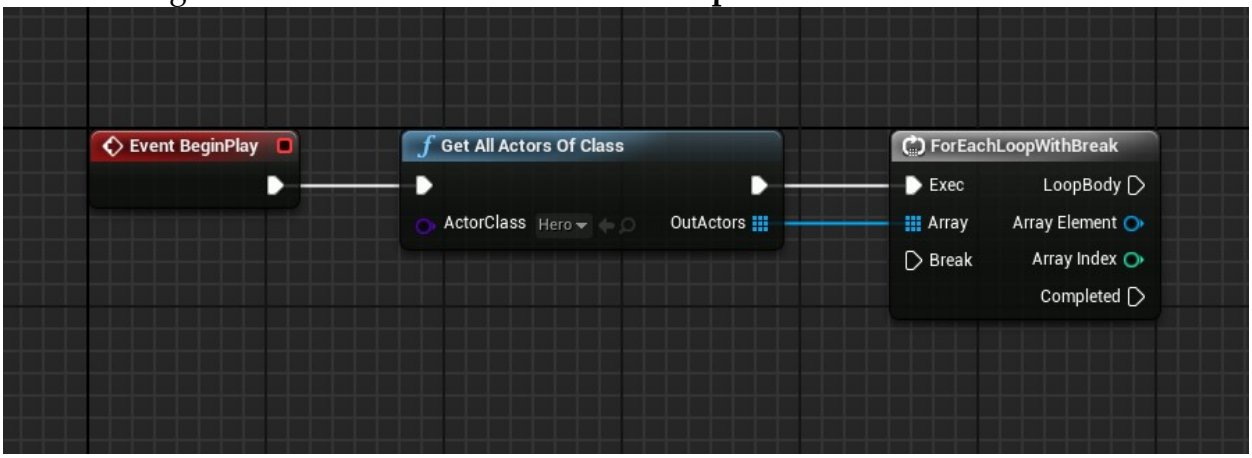
Thêm trí tuệ nhân tạo AI cho quân địch

Hãy quay lại Unreal Engine và tập trung vào **Content Browser**. Bây giờ chúng ta cần một đối thủ. Để thêm phần này vào blueprint của chúng ta, hãy làm theo các bước sau:

1. Nhấp chuột phải và chọn **Blueprint Class**.
2. Ở cuối cửa sổ, hãy bỏ tất cả các class bằng cách tìm kiếm **AIController**.
3. Chọn **AIController** trong **Controller** và nhấn nút **Select** ở góc dưới cùng bên phải.
4. Chúng ta sẽ đặt tên cho **AIController** là *Enemy*.
5. Mở **AIController** của *Enemy* và đi đến tab **EventGraph**.

Đầu tiên, chúng ta phải tìm *Hero* và sau đó lưu trữ nó trong một biến cục bộ để có thể sử dụng bất cứ lúc nào. Để làm đươc như vậy, hãy làm theo những bước sau:

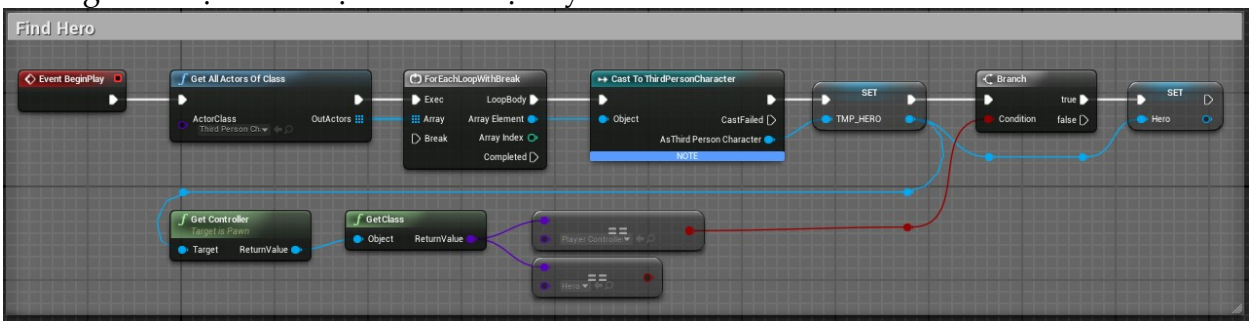
1. Nhấp chuột phải vào vùng trống trong **EventGraph** và tìm kiếm **Event Begin Play**.
2. Kéo từ chân thực thi và tìm kiếm để tạo node **Get All Actors Of Class**.
3. Đặt chân **Actor Class** thành **Hero**.
4. Kéo từ mảng **Out Actors** và tìm kiếm **ForEachLoopWithBreak**:



Chúng ta muốn một bộ lọc cho class *Hero* trong *ThirdPersonCharacter* được trả về. Để làm đươc như vậy, hãy đi theo những bước sau:

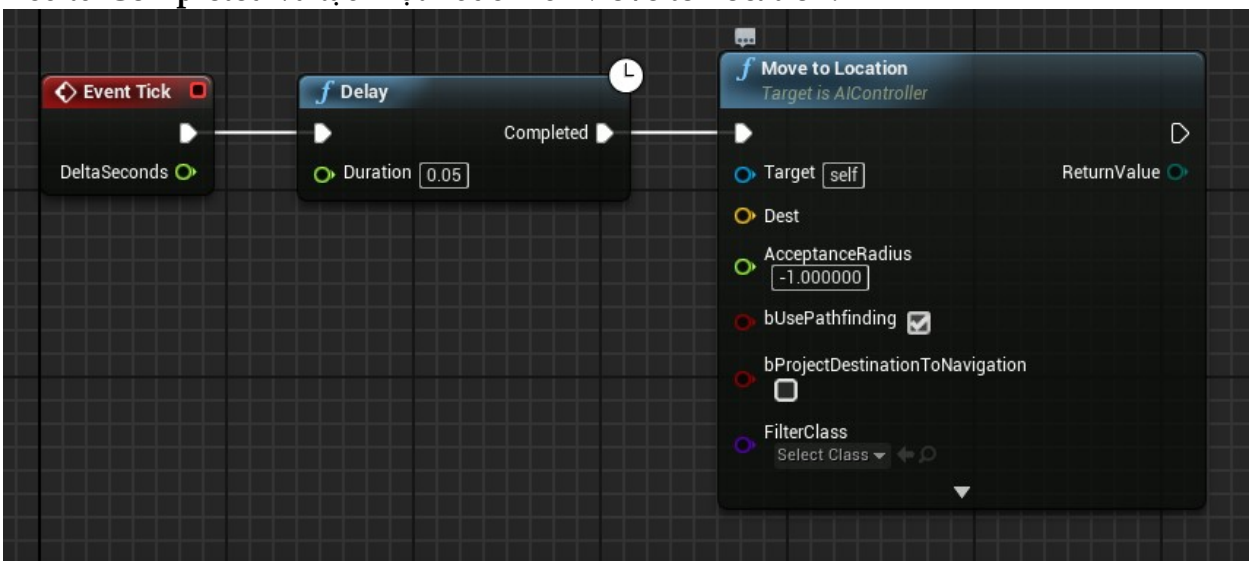
1. Kéo từ chân **Array Element** và tìm kiếm **Cast to ThirdPersonCharacter**.
2. Sau đó, kéo từ chân **As ThirdPersonCharacter** và tìm kiếm **Get Controller**.
3. Nhận lại class controller bằng cách sử dụng node **Get Class**.
4. So sánh class *Hero* với node **Class = Class**.
5. Kéo từ chân trả về của node **Equal** và tạo ra node **Branch**.

6. Từ chân **Loop Body**, liên kết với **Branch** vừa tạo.
7. Từ chân thực thi **True** của node **Branch**, tạo node **SET**.
8. Sau đó, kéo từ chân pure cast **As ThirdPersonCharacter** sang node **SET** mà chúng ta vừa tạo.
9. Chúng ta sẽ đặt bình luận cho tác vụ này là **Find Hero**:



Bây giờ chúng ta phải liên tục cập nhật các hướng dẫn trong *Enemy* để di chuyển về phía nhân vật *Hero* đang chạy trốn. Lưu ý rằng có những node có thể trực tiếp đạt được điều này, chẳng hạn như **Simple Move to Actor**, **AI MoveTo**, v.v. Chúng ta sẽ thực hiện hành vi tương tự để cho bạn một cái nhìn về cách thực hiện điều này một cách bí mật. Thực hiện các bước sau:

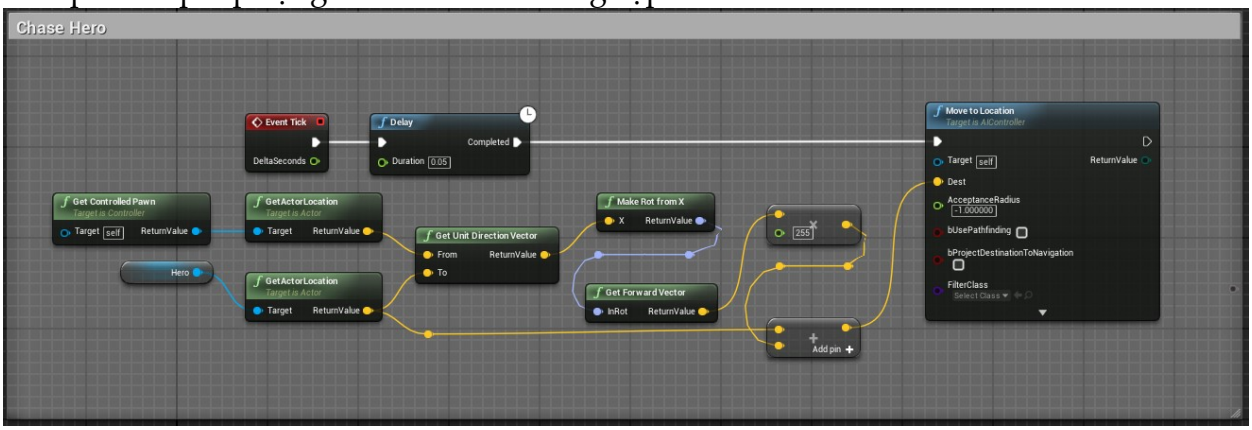
1. Nhấp chuột phải vào vùng trống trong **EventGraph** và tìm kiếm **Event Tick**.
2. Từ chân thực thi, thả và tìm kiếm **Delay**.
3. Đặt chân **Duration** thành **0,05** giây.
4. Kéo từ **Completed** và tạo một node mới **Move to Location**:



Bây giờ, chúng ta cần chuyển hướng từ *Enemy* sang *Hero* và di chuyển *Enemy* theo hướng này. Chúng ta có thể làm điều này như sau:

1. Lấy biến *Hero* và thả nó gần nút **Delay**
2. Từ biến *Hero*, lấy vị trí của actor với node **GetActorLocation**.

3. Kéo từ chân **Return Value** của node **GetActorLocation**, thả chuột vào vùng trống của sơ đồ và tìm kiếm **Get Direction Vector**.
4. Bây giờ, nhấp chuột phải và tìm kiếm **Get Controlled Pawn**.
5. Kéo từ chân **Return Value** của **Get Controlled Pawn** và **GetActorLocation**.
6. Sau đó, kéo từ chân **Return Value** của node **GetActorLocation** và cắm nó vào chân **From** trong node **Get Direction Vector**.
7. Kéo từ chân **Return Value** của node **Get Direction Vector** và **Make Rot from X**.
8. Kéo từ chân **Make Rot From X** và **Get Forward Vector**.
9. Bây giờ, từ **Get Forward Vector**, chúng ta sẽ nhân nó với **255** để có 255 đơn vị theo hướng vector tiến thẳng phía trước.
10. Cuối cùng, chúng ta sẽ thêm phần này vào node **GetActorLocation** của *Hero* pawn.
11. Kết quả của phép cộng là đích đến sẽ cung cập cho node **Move to Location**:



12. Bây giờ, hãy bình luận xung quanh nó và gọi bình luận là *Chase Hero*.

Lưu tất cả chúng lại!

Bây giờ, quay lại phần **Viewport** và nhấn **Simulate**. Bạn sẽ thấy nhân vật *Hero* của chúng ta đang chạy về phía trước và rẽ phải khi phát hiện chướng ngại vật cản đường. Nhân vật *Enemy* của chúng ta đang đuổi theo rất nhanh phía sau nhân vật *Hero* của chúng ta. Bây giờ, hãy tưởng tượng nếu bạn ở vị trí của nhân vật *Hero*; bạn sẽ hoàn toàn sợ hãi!

Tóm tắt

Những gì chúng tôi đã trình bày ở đây là cách bạn có thể tạo AI cho *Enemy* truy đuổi người chơi hoặc AI khác. Loại hành vi này có thể được sử dụng để tạo thử thách và cung cấp phản hồi cho người chơi, giúp mang lại trải nghiệm chơi trò chơi tốt hơn. AI chắc chắn có thể phức tạp hơn, nhưng điều này chỉ nên được thực hiện nếu cần để cải thiện trải nghiệm chơi. Trong chương tiếp theo, chúng ta sẽ thêm tính ngẫu nhiên và xác suất vào nhân vật AI của mình để làm cho nó hoạt động theo cách thú vị hơn.