Course - Lập trình AI cho Unreal Engine

Thực tập AI tuần tra, rượt đuổi và tấn công

Kết hợp một số thành phần mà chúng ta đã sử dụng trong các chương trước, bao gồm giác quan (Sensor) và chuyển động (Movement) của AI, để nhân vật AI của chúng ta điều hướng. Sau đó, chúng tôi sẽ áp dụng tính ngẫu nhiên cho thời gian mà nhân vật AI sẽ dành để đuổi theo các nhân vật mà nó phát hiện.

Tags: Thực tập AI tuần tra, rượt đuổi và tấn công

Mục tiêu của chương này là tạo ra một AI sẽ tấn công bạn bằng cách sử dụng Behavior Tree; component AI cuối cùng này sẽ có lợi cho bạn vì không có giải pháp hoàn hảo. Vì vậy, hiểu từng công cụ có sẵn cho phép bạn khai thác lợi thế của từng component để sử dụng trong AI. Những công cụ này cho phép bạn tạo AI phản ứng nhanh và thuyết phục.

Tạo ra một Blackboard và Behavior Tree

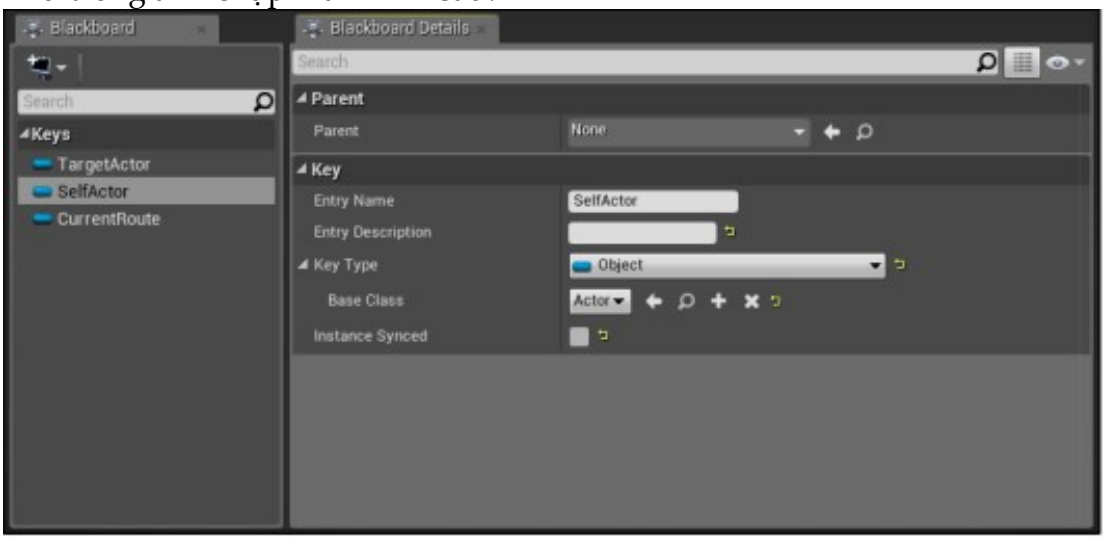
Blackboard xác định không gian biến cục bộ cho Behavior Tree. Các Blackboard này cũng có thể đồng bộ hóa với các phiên bản khác của cùng một Blackboard. Blackboard có thể được tạo trước vì bạn sẽ thấy mình thường xuyên sửa đổi chúng cho đến khi khám phá ra mọi thứ bạn cần cho Behavior Tree của mình.

Trước tiên, chúng ta sẽ tạo dữ liệu Blackboard của mình và sau đó cung cấp dữ liệu đó cho Behavior Tree của chúng ta. Bây giờ, chúng ta hãy bắt đầu! Dưới đây là các bước:

1. Nhấp chuột phải vào thư mục *Content* và tạo một thư mục mới có tên là *AI*.
2. Bây giờ, nhấp chuột phải vào bên trong thư mục *AI* và cuộn xuống để tìm **Artificial Intelligent**; sau đó, nhấp vào **Blackboard**. Hãy đặt tên cho nó là *EnemyData*.
3. Chúng ta sẽ mở *EnemyData* và sau đó tạo hai biến kiểu **Object** sẽ được Behavior Tree sử dụng. Đặt tên cho cái đầu tiên là *TargetActor* và cái tiếp theo là *CurrentRoute*.

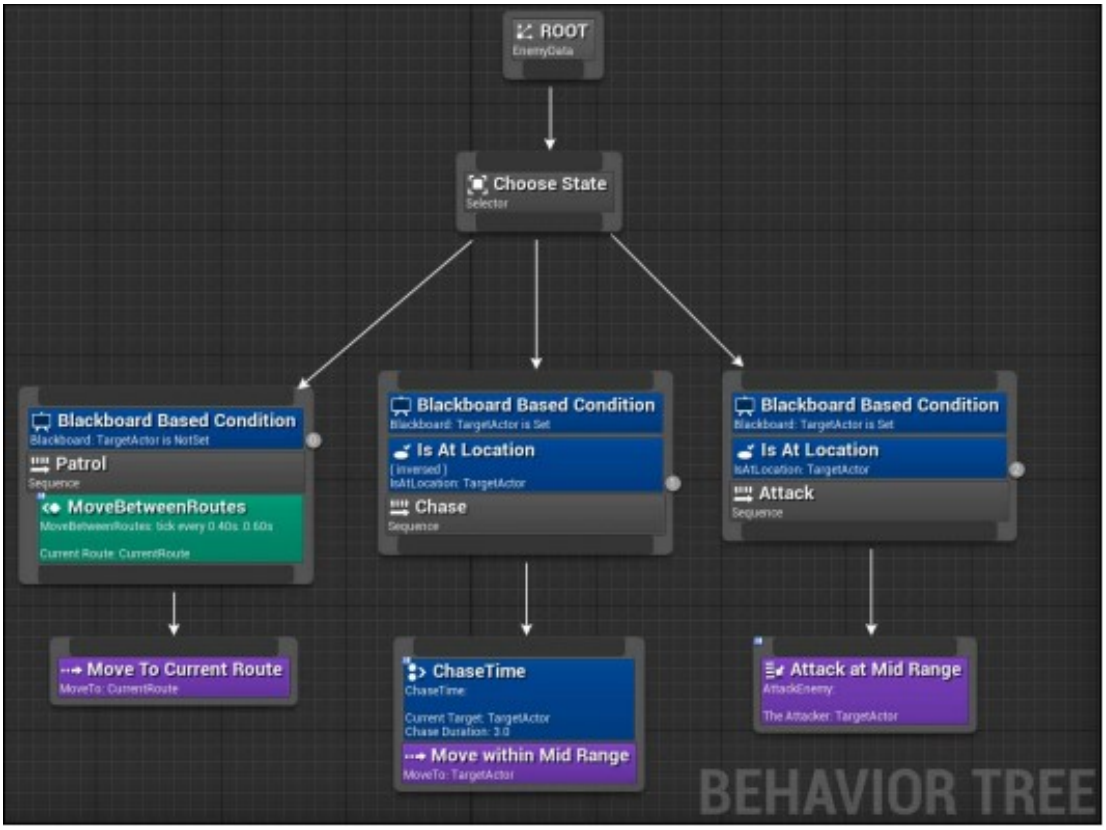
   > **Chú ý:** Trong trường hợp bạn đang sử dụng **Enumeration**, bạn có thể xác định **Enumeration** giống như cách chúng ta sẽ xác định actor cho các đối tượng này.

4. Nhấp vào *TargetActor* và chọn mở tùy chọn **Key Type** trong phần **Key**. Sau đó, thay đổi giá trị **Base Class** thành **Actor**. Chúng ta cũng sẽ làm điều tương tự với *CurrentRoute*, như trong ảnh chụp màn hình sau:



Behavior Tree là một cây gồm các node có sơ đồ cấu trúc quy định việc kiểm soát luồng, mỗi node trong đó đại diện cho mã thực tế để AI thực thi. Điều này dẫn đến việc AI đưa ra một

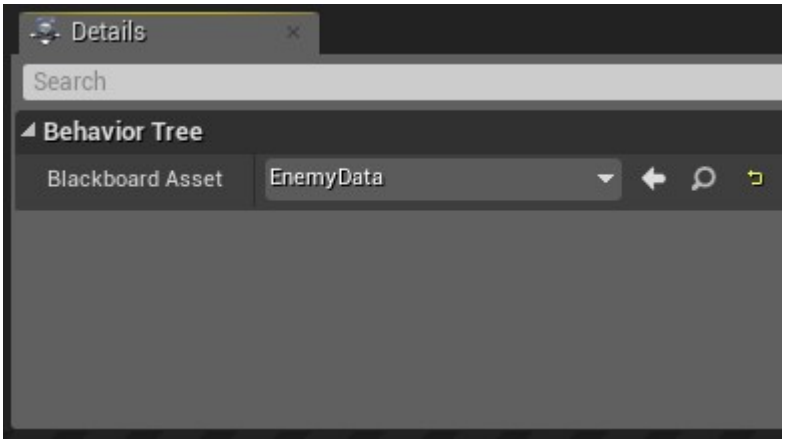
chuỗi các quyết định phù hợp với thông tin đầu vào mà nó nhận được. Để xem ví dụ về những gì chúng ta sẽ thực hiện ngày hôm nay, đây là giao diện của Behavior Tree trong Unreal Engine 4:



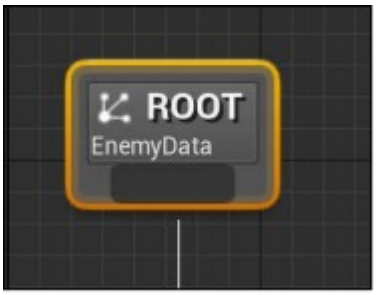
Có nhiều dạng state machine (bộ máy trạng thái) có sẵn trên thế giới. Bahavior Tree là một dạng cây với một cây con gồm các node cành lá. Nó có các tài nguyên composite cho phép bạn kiểm soát luồng, dẫn đến việc thực hiện các node cành lá, cũng như kiểm soát luồng bổ sung. Mức độ kiểm soát này cho phép bạn tạo các Behavior Tree sâu với rất nhiều quyền kiểm soát.

Hãy bắt đầu tạo và xem nhanh những gì có được trong khóa học này:

1. Chúng ta sẽ nhấp chuột phải vào **Content Browser** và trỏ đến **Artificial Intelligent**. Bây giờ, hãy nhấp vào **Behavior Tree** và đặt tên cho cây này là *EnemyAI* này.
2. Mở *EnemyAI* và bạn sẽ thấy *EnemyData* được điền trong node *ROOT*. Nếu không, hãy nhấp vào node *ROOT* và đặt tài nguyên Blackboard thành *EnemyData*:



3. Sau khi đã thiết lập được Blackboard vào Behavior Tree, chúng ta sẽ có được sơ đồ như hình bên dưới:



Tấn công trung tầm

Nó sẽ là một đường kẻ từ tâm mắt của chúng ta. Nó sẽ là cách bạn tấn công AI và cũng là cách AI sẽ tấn công bạn. Chúng ta sẽ chỉ cần chạy một đường vạch và sau đó vẽ một đường thẳng gở rối (debug) lên trên, tạo ra một chùm tia màu đỏ trên trán người chơi của chúng ta! Chúng ta sẽ kiểm tra xem nó có hoạt động không và sau đó, chúng ta sẽ tích hợp nó cho cả hai player.

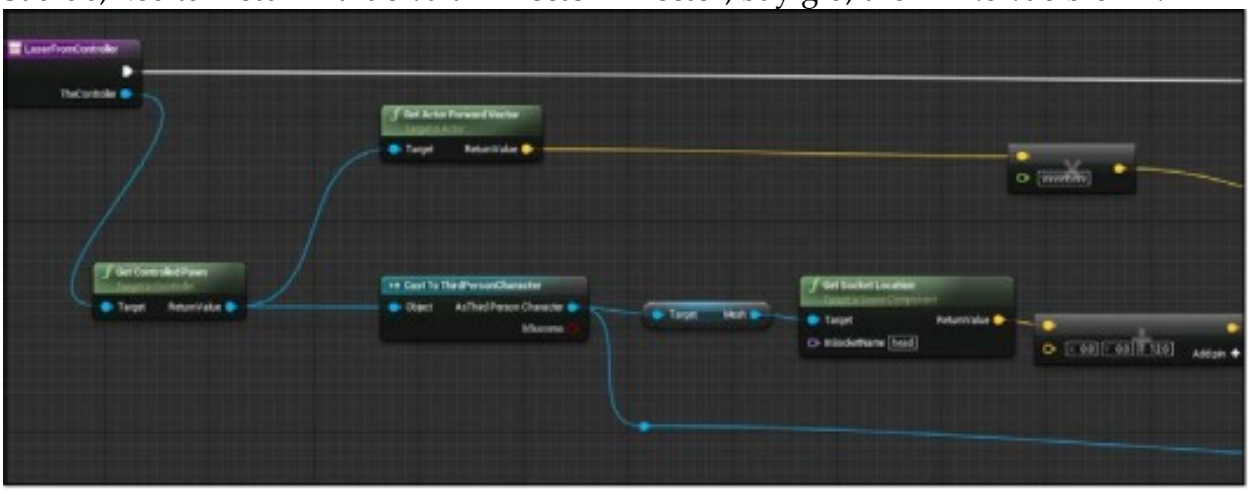
Bây giờ, để bắt đầu tạo chức năng này, chúng ta sẽ làm điều gì đó khác biệt để tiết kiệm thời gian. Chúng ta sẽ đưa chức năng vào **Blueprint Function Library** để chia sẻ trên mọi sơ đồ Blueprint.

Để giữ chức năng có ích cho cả AIController và PlayerController, chúng ta sẽ cung cấp controller làm đầu vào và sau đó xuất ra một actor. Chức năng này cũng sẽ xử lý việc vẽ chùm tia màu đỏ khi cần thiết:

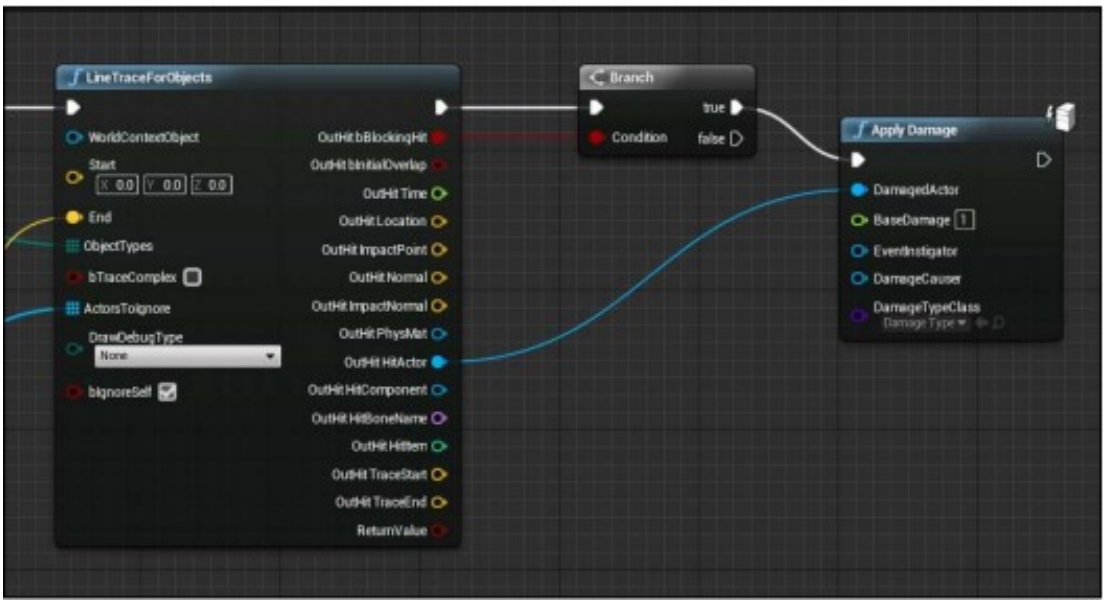
1. Hãy nhấp chuột phải, vào menu **Blueprints** và sau đó trỏ đến **Blueprint Function Library**. Đặt tên cho nó là *EnemyLibrary* này và mở nó lên.
2. Trong panel **My Blueprint**, hãy nhấn vào nút **Add New** để thêm +**Function** mới. Đặt tên cho nó là *LaserFromController*. Đặt đầu vào đầu tiên là controller có tên *theController* và sau đó thiết lập ngõ xuất là một actor có tên là *Hit Actor*. Cuối cùng, tạo một **Local**

Variable có tên là *foundActor* có kiểu dữ liệu **Actor**.

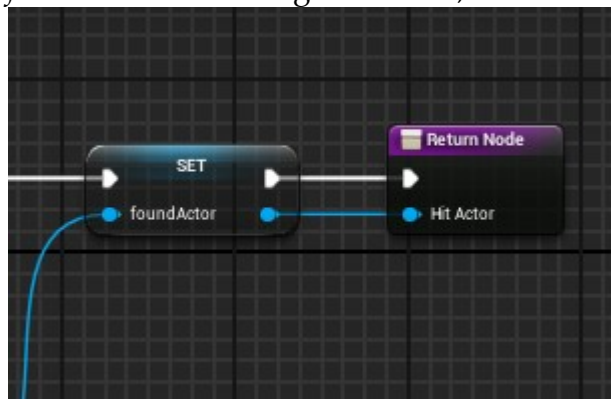
3. Kéo từ controller và tìm kiếm **Get Controlled Pawn**. Sau đó, từ đây, chúng tôi sẽ tạo một đường thẳng dựa trên góc quay hiện tại của nhân vật. Kéo từ **Get Controlled Pawn** và tìm kiếm **Get Actor Forward Vector**.
4. Chúng ta sẽ nhân **Get Actor Forward Vector** với **9999999**.
5. Quay lại node **Get Controlled Pawn** từ Bước 3. Sau đó, chiếu nó sang *ThirdPersonCharacter* bằng node **Cast to ThirdPersonCharacter** và bây giờ chúng ta có đã thể truy cập biến **Mesh**. Chúng ta sẽ định vị xương đầu của chúng ta với node **Get Socket Location**. Thiết lập giá trị **InSocketName** trong node **Get Socket Location** là *head*.
6. Sau đó, kéo từ **ReturnValue** và tìm **Vector + Vector**; bây giờ, thêm **12.0** vào biến **Z**:



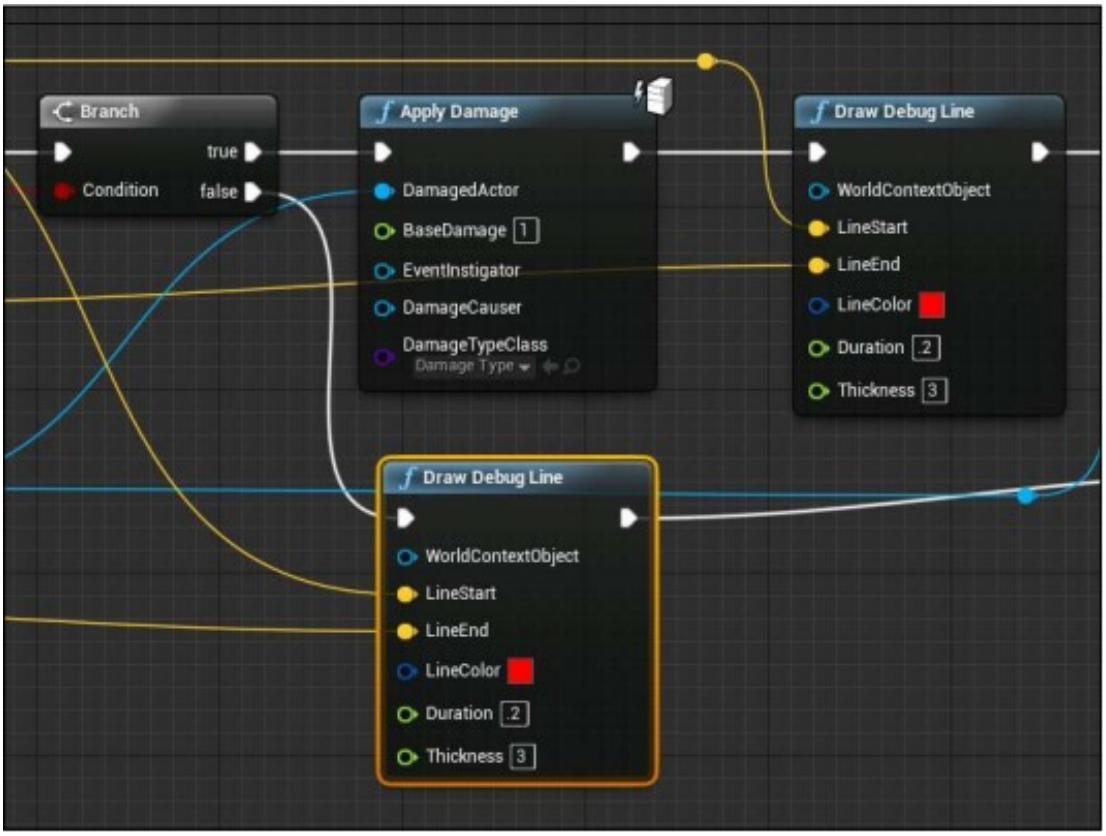
7. Bây giờ, chúng ta phải thêm kết quả của Bước 4 vào kết quả của Bước 6 bằng node **Vector + Vector**. Sau đó, chúng ta sẽ chuyển sang vẽ đường thẳng laser cho nhiều đối tượng. Hãy tìm kiếm **LineTraceForObjects** rồi cắm kết quả đó vào chân **End** của node **LineTraceForObjects**.
8. Bây giờ, hãy đặt giá trị **Start** của node **LineTraceForObjects** ở bước trước với ngỏ ra từ node **Vector + Vector** của bước 6. Sau đó, chúng ta sẽ có một chùm tia bắt đầu từ đầu của nhân vật và kéo nó về phía trước nhân vật trong không gian 3 chiều.
9. Kéo từ **ObjectTypes** và tạo một mảng với node **Make Array**; sau đó điền **Pawn** vào đầu vào.
10. Kéo từ **ActorsToIgnore** và tạo một mảng; sau đó, nối node **Cast to ThirdPersonCharacter** vào phần tử **0**.
11. Từ **LineTraceForObjects**, hãy tách chân **OutHit** bằng cách click chuột phải sau đó chọn **Split Struct Pin**. Bây giờ, kéo từ **OutHit bBlockingHit** để tìm kiếm và tạo node **Branch**.
12. Nhấp chuột phải lên sơ đồ để tạo và tìm kiếm node **Apply Damage**; sau đó, kết nối chân thực thi của node **Branch** với node này. Kéo từ **OutHit HitActor** và cắm nó vào chân **DamagedActor** của node **Apply Damage**, sau đó điền tùy chọn **DamageTypeClass** với **Damage Type**:



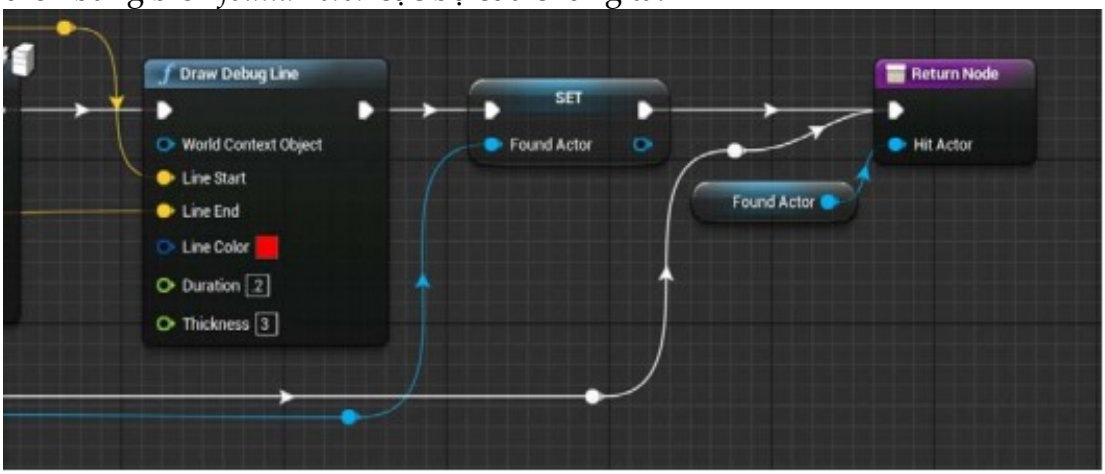
13. Tiếp theo, nhấp chuột phải và tìm kiếm node **Draw Debug Line**. Chúng ta sẽ cắm vào chân **LineStart** kết quả từ Bước 6, đây là điểm bắt đầu của đường thẳng của chúng ta. Sau đó, kéo chân **OutHit Location** của node **LineTraceForObjects** sẽ được cắm vào **LineEnd** của node **Draw Debug Line**. Chọn các giá trị cho **LineColor** là màu đỏ, cho **Duration** giá trị **2** và cho **Thickness** là **3**.
14. Cuối cùng, kéo biến *foundActor* cục bộ mà chúng thiết lập lúc đầu vào sơ đồ. Kéo từ chân **OutHit HitActor** của node **LineTraceForObjects** và đặt nó vào biến cục bộ *foundActor* của chúng ta. Sau đó, bơm cái này vào **Return Node**.



15. Chúng ta sẽ sao chép **Draw Debug Line** để kết nối với chân **False** của node **Branch**:



16. Thay vì lấy **OutHit Location** cho **LineEnd**, chúng ta sẽ lấy kết quả từ Bước 7.
17. Sau đó, chúng ta sẽ cắm vào **Return Node**. Chân **Hit Actor** trong **Return Node** được điền bằng biến *foundActor* cục bộ của chúng ta:



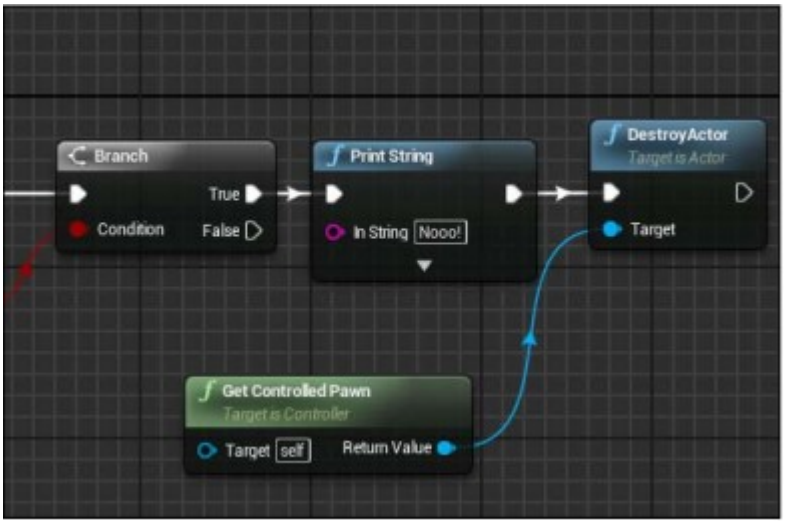
Controllers

Trong ví dụ này, chúng ta sẽ cung cấp cho PlayerController khả năng nhận và gây sát thương. Ngoài ra, chúng ta cũng sẽ sử dụng thêm AI Perception, chúng ta sẽ đăng ký nguồn Stimuli cho controller của mình. Tiếp theo, chúng ta sẽ thiết lập AIController. Chúng ta sẽ cung cấp cho nó khả năng cảm nhận các phản ứng. Sau đó, chúng ta sẽ tạo một component

Actor, component này sẽ chứa chức năng cần thiết để cập nhật Behavior Tree.

Hãy tiếp tục và bắt đầu tạo các component cơ sở này, như sau:

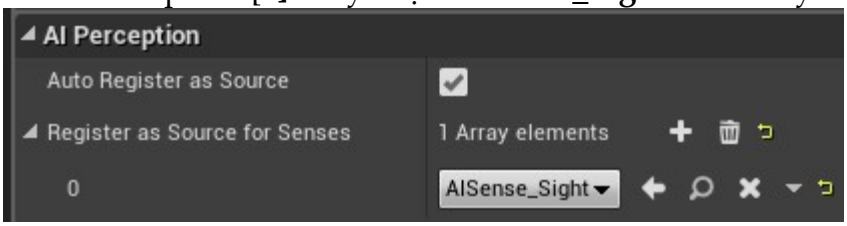
1. Trong thư mục *AI*, nhấp chuột phải và chọn **Blueprint Class**. Sau đó, từ tùy chọn **All Class**, chọn **PlayerController**. Sau đó, chúng ta muốn đặt tên cho cái này là *OutController*.
2. Mở **Event Graph** và tìm **Event BeginPlay**.
3. Nhấp chuột phải vào gần đó để tìm kiếm và tạo **Get Controlled Pawn**. Sau đó, kéo từ **Return Value** để tìm kiếm và tạo **Assign OnTakeAnyDamage**.
4. Thao tác này sẽ tạo event **OnTakeAnyDamage** và tạo ra **Damage**.
5. Chúng ta cần tạo một biến để giữ biến *Health* của mình. Nhấp vào nút **Add New** trong **My Blueprint** và chọn **+Variable** để tạo một biến **Float** có tên là *Health*. Đặt giá trị mặc định là **5**.
6. Nhìn vào **OnTakeAnyDamage**, rút ra **Damage** và trừ nó khỏi *Health*. Sau đó, đặt kết quả trong biến *Health* của chúng ta. Tiếp theo, chúng ta sẽ kiểm tra xem *Health* có nhỏ hơn hoặc bằng **0** hay không.
7. Tạo một node **Branch** từ các kết quả. Kéo từ **True** và nhấp chuột phải để tìm **Print String**. Điền thông tin này với chuỗi "**Player Died**", nhấp chuột phải, sau đó tìm kiếm **DestroyActor**.
8. Nhấp chuột phải và tìm kiếm **Get Controlled Pawn**; sau đó, cắm cái này vào chân **Target** trên **DestroyActor** từ bước trước. Bây giờ, chúng ta có thể nhận sát thương từ đối tác AI của mình:



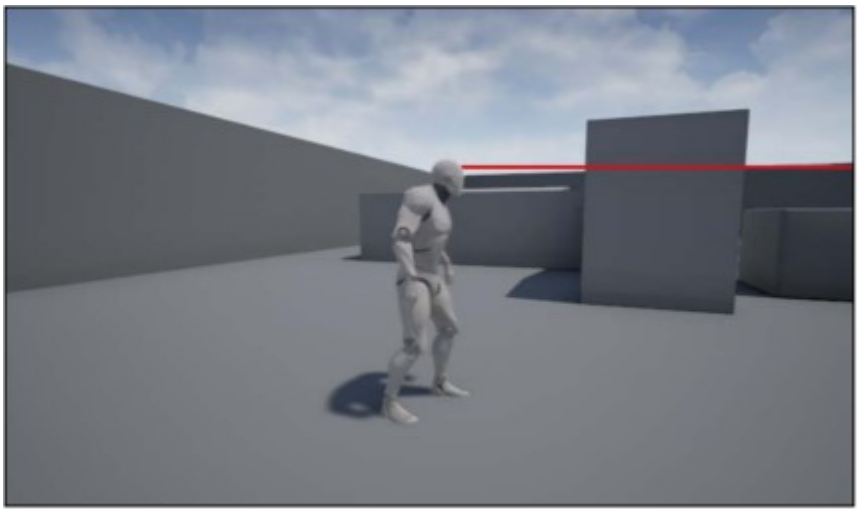
9. Nhấp chuột phải và tìm kiếm event **E Input**. Kéo chân **Pressed** và nhả chuột để tìm kiếm **Sequence**. Kéo từ **Then 0** và thả ra để tìm kiếm **DoOnce**. Kéo từ **Then 1**, sau đó thả ra để tìm kiếm **Delay**. Bây giờ, đặt **Duration** thành **0,75** và cắm **Completed** vào chân **Reset** trong **DoOnce**.
10. Quay về node **DoOnce**, kéo từ **Completed** và nhả chuột để tìm kiếm **LaserfromController**. Sau đó, kéo từ **Hit Actor** và nhả chuột ra để tìm kiếm **Apply Damage**. Đặt **BaseDamage** thành **1.0**, **Damage Causer** thành **Self**, và

DamageTypeClass thành **Damage Type**.

11. Bây giờ, chúng ta muốn thêm controller này làm **StimuliSource**. Trong panel **Components**, nhấp vào **Add Component** và tìm **AIPerceptionStimuliSource**.
12. Nhấp chọn **Auto Register as Source** và sau đó trong phần **Register as Source for Senses** nhấp vào **[+]**. Hãy chọn **AISense_Sight** từ các tùy chọn có sẵn.

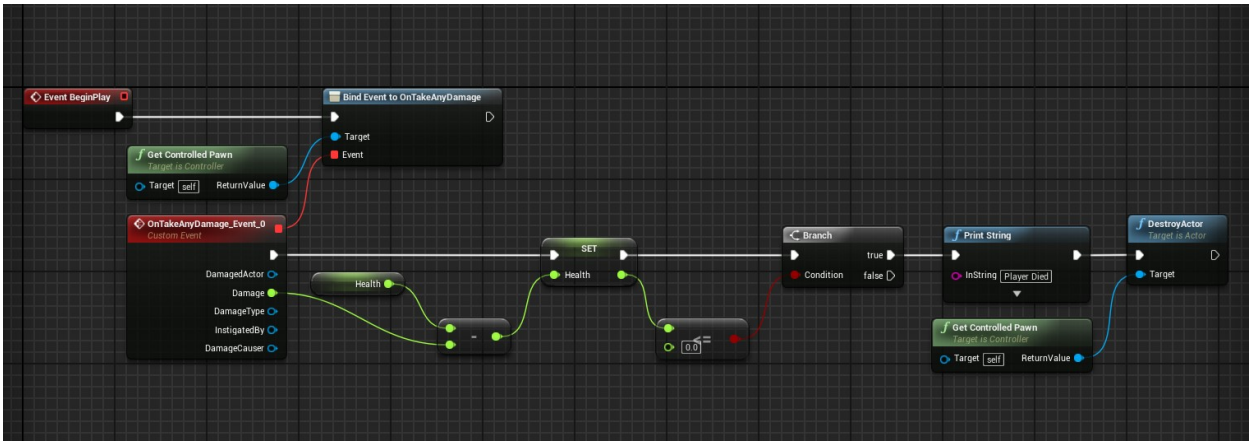


Bây giờ, hãy biên dịch blueprint này. Chúng ta đã hoàn tất thiết lập PlayerController của mình. Bây giờ chúng ta cần tạo một thiết lập AIController để AI của chúng ta có thể bắn trả chúng ta:

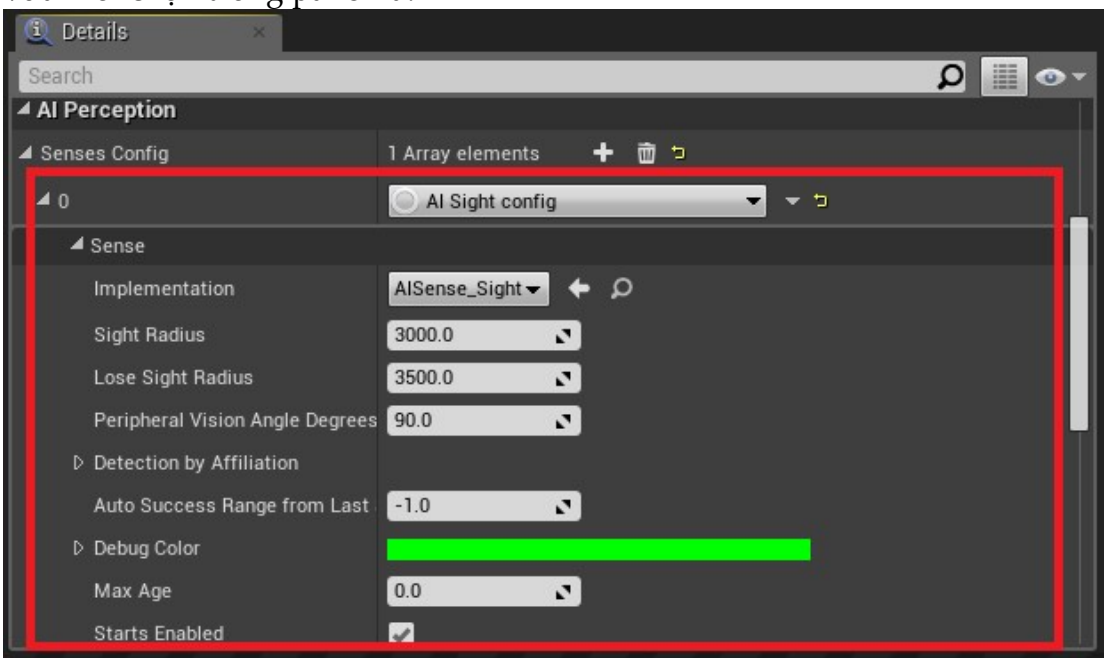


Hãy thiết lập AIController của chúng tôi như sau:

1. Nhấp chuột phải và nhấp vào **Blueprint Class**. Trong phần **All Classes**, tìm kiếm **AIController**. Chọn nó và sau đó nhấp vào **Select** để tạo nó. Đặt tên cho nó là *EnemyController* và mở nó trong EventGraph.
2. Hãy tạo một biến **Float** mới có tên là *Health*. Sau đó, đặt giá trị mặc định thành **5.0**.
3. Phần đầu của blueprint này giống nhau ở cả hai nhân vật ngoại trừ một vài thay đổi nhỏ. Vì vậy, chúng ta có thể sao chép mã từ *OurController* hoặc bắt đầu từ Bước 2 trong phần trước. Chúng ta sẽ sao chép phần được hiển thị trong ảnh chụp màn hình sau vào *EnemyController* từ *OurController*:



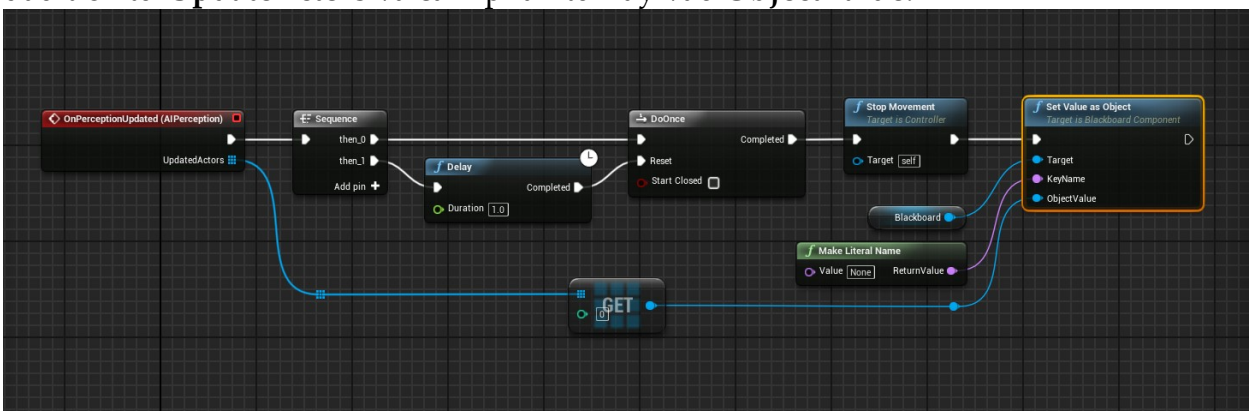
4. Khi kết thúc event **Bind Event to OnTakeAnyDamage**, kéo và nhả chuột để tìm kiếm **Run Behavior Tree**. Đặt giá trị **BTAsset** thành *EnemyAI*, là nội dung mà chúng ta đã tạo trước đó trong chương. Bây giờ, mã code trong cây BT sẽ chạy trên AIController này.
5. Bây giờ AI có thể nhận sát thương, chúng tôi cần nó cảm nhận người chơi ở gần nó. Chúng ta sẽ làm điều này với component AI Perception.
6. Trong phần **Components**, nhấp vào **Add Component** và tìm kiếm **AIPerception**. Sau khi thêm component này, trong panel **Details** bên dưới phần **AI Perception** ở phía bên tay phải, tìm tùy chọn **Senses Config** từ menu thả xuống và nhấn **+**. Sau đó, đặt nó thành tùy chọn **AI Sight config** từ menu thả xuống. Sau đó mở chi tiết của tùy chọn vừa mới chọn trong panel ra.



7. Sau đó mở tiếp **Detection by Affiliation** và check vào tùy chọn **Detect Neutrals**.
8. Đi tới phần **Events** và nhấp vào nút [+] trên tùy chọn **OnPerceptionUpdated** để nhận các actor bị AI phát hiện.
9. Từ node **OnPerceptionUpdated (AIPerception)**, kéo và thả để tìm kiếm **Sequence**. Chúng ta muốn với **then_0** đường thẳng sẽ được vẽ, vì thế từ chân **Then 0**, hãy tìm kiếm **DoOnce**. Sau đó, chúng ta sẽ cắm chân **then_1** vào node **Delay**. Hãy thiết lập

Duration 1. Bây giờ, hãy kết nối **Completed** vào chân **Reset** của node **DoOnce**.

10. Tập trung vào **DoOnce**. Chúng ta sẽ kéo từ chân **Completed** và sau đó nhả ra để tìm kiếm **Stop Movement**. Nhấp chuột phải và tìm kiếm "Get **Blackboard**" để đưa biến **Blackboard** vào sơ đồ. Kéo từ **Blackboard** và thả ra để tìm kiếm node **Set Value as Object**. Nhấp chuột phải, tìm kiếm node **Make Literal Name**, thiết lập Value với giá trị "**TargetActor**" và bơm nó vào **Key Name** của node **Set Value as Object**. Lấy phần tử đầu tiên từ **UpdateActors** và cắm phần tử này vào **ObjectValue**:



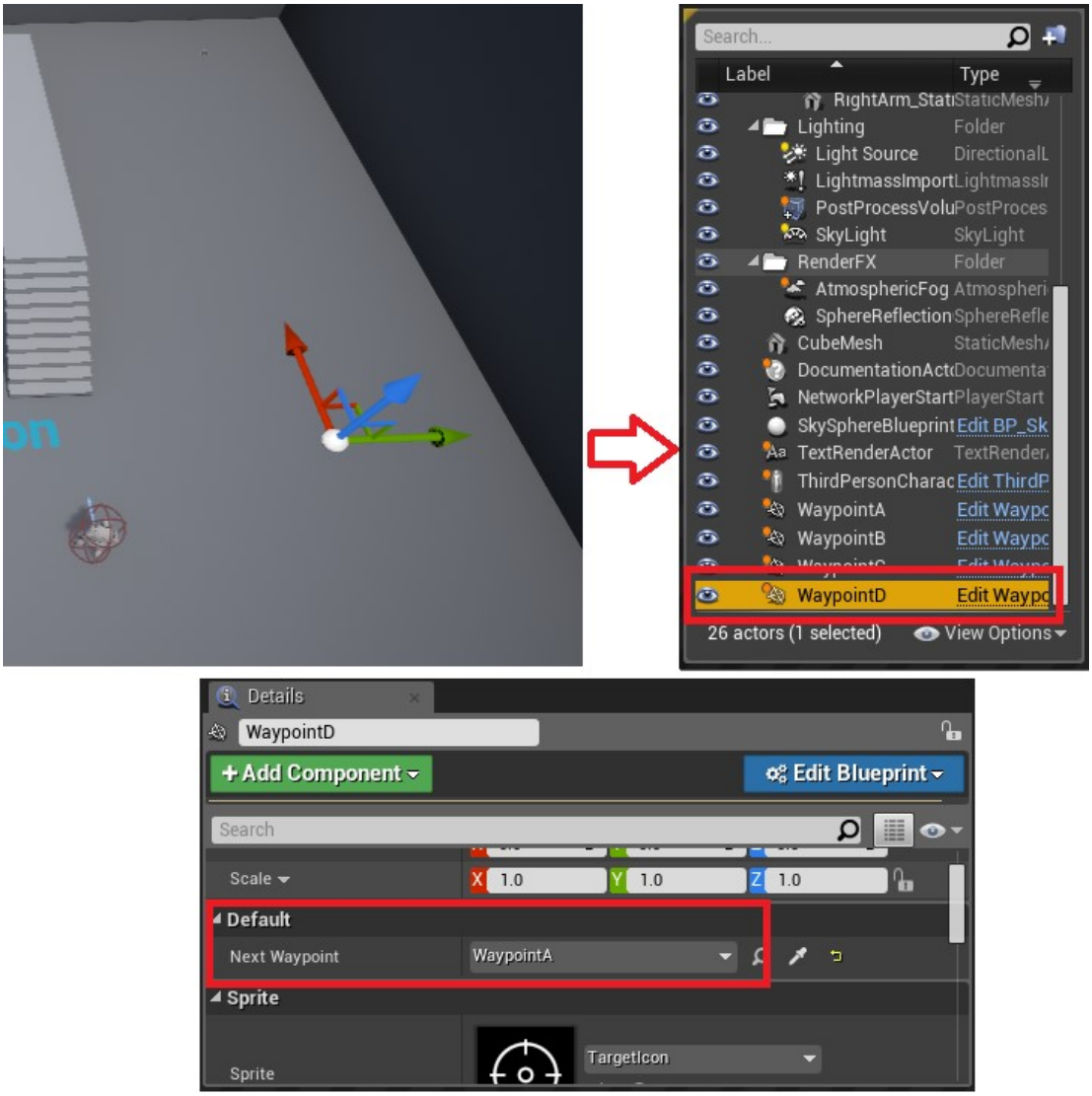
Bây giờ, AI sẽ ngay lập tức báo cáo các actor được cảm nhận cho cây Behavior Tree của chúng ta, và sau đó nó có thể được sử dụng ngay lập tức trong quyết định tiếp theo của nhân vật AI của chúng ta.

Điểm đến (Waypoint)

Tương tự như dự án AI Sense, các điểm đến này sẽ có tham chiếu đến điểm hẹn tiếp theo. Liên kết đơn giản này giữa hai điểm hẹn cho phép chúng ta tạo đường dẫn để AI của chúng ta điều hướng đến.

Bây giờ, chúng ta nên tiếp tục tạo các điểm hẹn này; thực hiện các bước sau:

1. Để bắt đầu, hãy nhấp chuột phải vào khoảng trống bên trong thư mục và nhấp chọn vào **Blueprint Class**. Ở dưới cùng, tìm kiếm trong **All Classes** cho **TargetPoint**. Chọn nó và sau đó nhấn **Select** để tạo nó. Sau đó, đặt tên cho nó là *Waypoint* và mở cái này trong EventGraph.
2. Ở phía bên tay trái bên dưới panel **My Blueprint**, hãy thêm một điểm tham chiếu biến đổi có tên là *NextWaypoint* và check **Editable** để cho nó có khả năng chỉnh sửa. Chúng ta sẽ sử dụng nó để tìm điểm tham chiếu tiếp theo để đi qua.
3. Thêm bốn điểm hẹn waypoint vào màn chơi và liên kết chúng với nhau bằng cách sử dụng biến *NextWaypoint*. Kết nối sẽ giống như **A->B->C->D**. D sau đó sẽ kết nối với A (**D->A**), sao cho nó tạo ra một vòng lặp.



BT Composites, Task, Decorator, and Service

Task được thực thi bởi composite. Composite rất quan trọng vì chúng ảnh hưởng trực tiếp đến luồng trong Behavior Tree.

Hiện tại, các composite có ba loại: **Sequence**, **Selector** và **Simple Parallel**. Dưới đây là một mô tả của mỗi:

- **Sequence**: Nó thực thi từng node, trả lại success (trạng thái thành công) trên node cuối cùng; tuy nhiên, nếu bất kỳ node nào không thành công, nó sẽ ngay lập tức trả về fail (trạng thái thất bại) và hủy bỏ phần luồng còn lại.
- **Selector**: Nó thực thi từng node, trả về success (trạng thái thành công) ngay lập tức và hủy bỏ phần luồng còn lại. Nếu một node trả về fail (trạng thái thất bại), nó sẽ tiếp tục cho đến khi nhận success hoặt kết thúc luồng còn lại.
- **Simple Paralled**: Điều này thực thi một tác vụ và một cây con cùng một lúc, cho phép

bạn đi bộ và cho phép một cây ra quyết định khác ở trên cùng của tác vụ đi bộ chẳng hạn.

Tasks thường là node cuối cùng trong thay đổi được gọi vì chúng chứa mã code ảnh hưởng trực tiếp đến các hành động AI. Chúng ta sẽ thực hiện task của riêng mình và tìm hiểu cách nó giao tiếp với Behavior Tree.

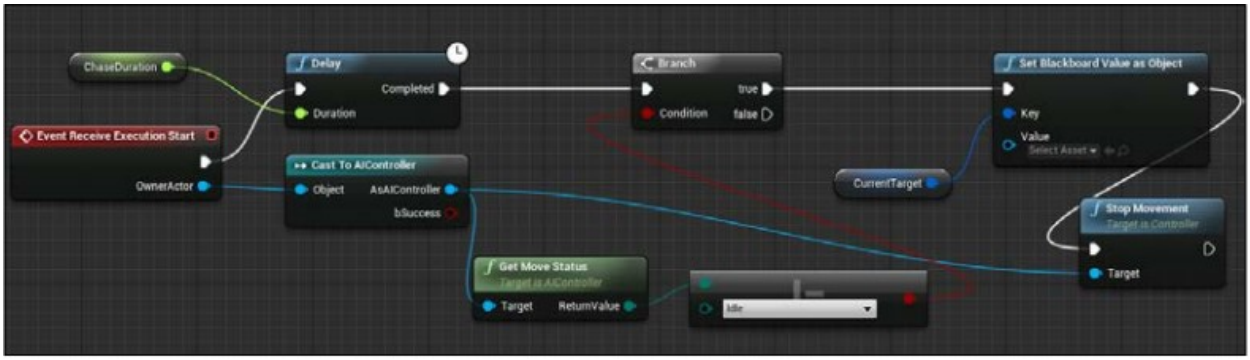
Decorators thực thi khởi điểm của composite hoặc task, có thể xác định xem composite hoặc node có nên được thực thi hay không. Đây là những điều tuyệt vời để tạo bài kiểm tra tùy biến cho task cụ thể này. Ví dụ, nó chỉ thực hiện *Break Door* nếu nó có chế độ *Door* và *Beast*.

Services được thực hiện trong khi luồng con đang hoạt động. Điều này có nghĩa là nó có một sự kiện tick riêng. Nó cho phép chúng ta thu thập hoặc cập nhật thông tin để thực hiện các thay đổi ngay lập tức trong trò chơi. Ví dụ, service bắn súng sẽ chịu trách nhiệm bắn bất cứ khi nào người chơi ở trong tầm nhìn. Điều này sau đó cũng cho phép bạn kiểm soát quá trình chụp và tránh bị gián đoạn khi bạn không muốn.

Trong phần này, chúng ta sẽ sử dụng mọi thứ ngoại trừ **Simple Parallel**. Hãy bắt đầu nào!

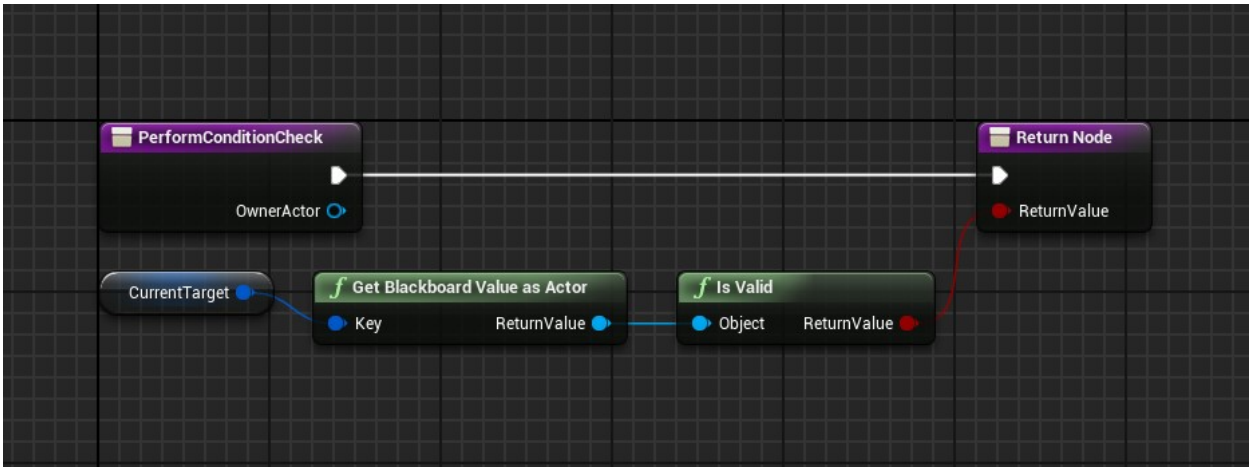
Đầu tiên, chúng ta sẽ tạo một **Decorator**, và nó sẽ chịu trách nhiệm thoát khỏi luồng nếu vượt quá thời gian cho phép. Vì vậy, trong trường hợp của chúng ta, quá trình này sẽ diễn ra sau 3 giây và nếu trạng thái di chuyển của chúng ta không ở chế độ chờ, chúng ta sẽ xóa **CurrentTarget**, như sau:

1. Nhấp chuột phải vào **Content Browser** và nhấp chọn vào **Blueprint Class**. Sau đó, đi xuống **All Classes** và tìm kiếm **BTD**. Click ngay chuột vào **BTDecorator_BlueprintBase** và sau đó nhấn nút **Select** để tạo một điều kiện BT mới. Đặt tên cho nó là *ChaseTime*.
2. Bây giờ, hãy mở *ChaseTime* và truy cập EventGraph. Nhấp chuột phải và tìm kiếm **Event Receive Execution Start**.
3. Chúng ta nên tạo hai biến. Loại đầu tiên sẽ là loại **Blackboard Key Selector** có tên là *CurrentTarget*. Loại thứ hai sẽ có kiểu **Float** có tên là *ChaseDuration*. Thiết lập giá trị mặc định cho nó là **3.0**. Cho cả hai chọn tùy chọn **Editable**.
4. Kéo từ **Event Receive Execution Start** và sau đó tìm kiếm **Delay**. Cắm *ChaseDuration* vào chân **Duration** của node **Delay**.
5. Kéo từ **Owner Actor** trên node **Event Receive Execution Start** và chuyển nó sang AIController bằng node **Cast to AIController**. Sau đó, lấy **AsAIController** và tìm kiếm **Get Move Status**. Kéo từ **Return Value** và tìm kiếm **Not Equal To**. Lấy từ kết quả và tạo **Branch**. Cắm chân **Completed** của node **Delay** vào node **Branch**.
6. Lấy **CurrentTarget** và đặt nó vào trong sơ đồ EventGraph. Kéo từ biến và tìm kiếm **Set Blackboard Value as Object**. Kéo **true** từ node **Branch** vào node này. Sau đó, kéo từ **AsAIController** và tìm kiếm **Stop Movement**. Xếp nó chạy sau **Set Blackboard Value as Object**:



Sau khi đã hoàn thành các xử lý trong Decorator, bây giờ chúng ta cần phải thêm các xử lý điều kiền cho Decorator, Trong trường hợp này, chúng ta muốn chắc chắn là CurrentTarget phải là đã được xác định bằng một mục tiêu cụ thể, chúng ta sẽ thực hiện việc xác nhận nó như sau:

1. Click chuột vào nút **Override** trên phần **Function** trong panel **My Blueprint**, chọn Perform Condition Check.
2. Sau khi nó mở ra sơ đồ mới, Hãy kéo biến CurrentTarget ra sơ đồ, sau đó kéo và nhả chuột để tìm **Get Blackboard Value as Actor**, Từ chân **Return Value**, hãy kéo và tìm node **Is Valid**.
3. Sau đó từ chân **Return Value** trên node **Is Valid**, hãy bơm nó vào chân **Return Value** của node **Return Node**.



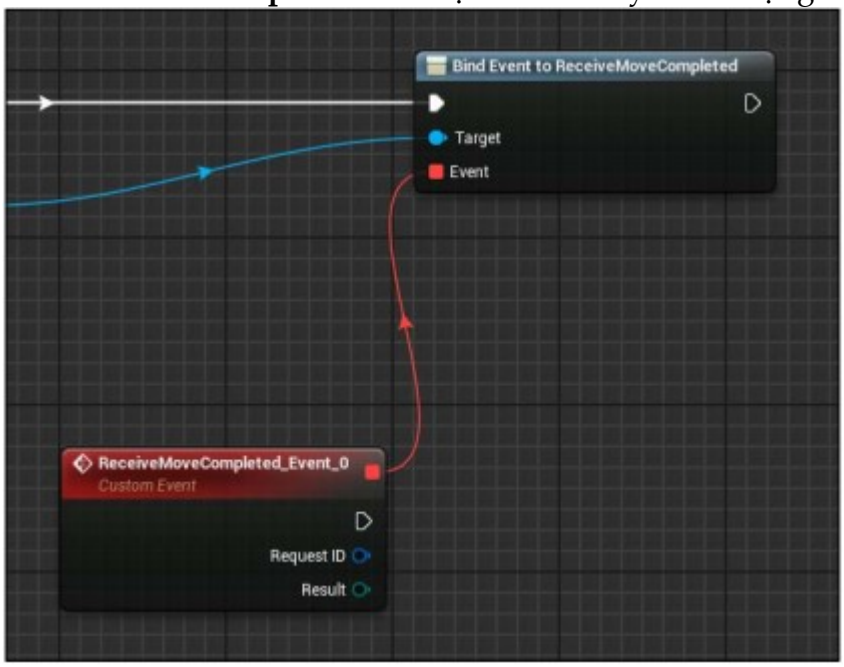
Bây giờ nó đã được thực hiện, hãy biên dịch nó. Sau đó, chúng ta sẽ chuyển sang component tiếp theo.

Thứ hai, chúng ta sẽ tạo một service sẽ đặt tuyến đường đầu tiên của chúng ta một cách ngẫu nhiên. Sau đó, khi chúng ta đến tuyến đường hiện tại, chúng ta sẽ chuyển sang tuyến đường tiếp theo trong danh sách. Điều này có thể xảy ra bất cứ lúc nào và các service cho phép bạn có một chức năng có thể chạy liên tục như sau:

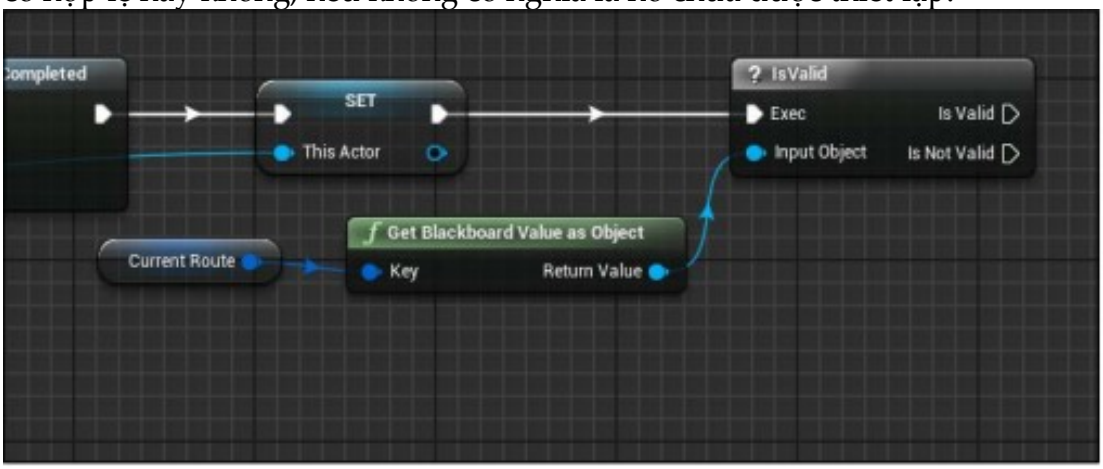
1. Nhấp chuột phải vào **Content Browser** và nhấp chuột vào **Blueprint Class**. Sau đó, đi

xuống **All Classes** và sau đó tìm kiếm **BTS**. Nhấp ngay chuột vào **BTService_BlueprintBase** và nhấn **Select** để tạo BT service mới. Đặt tên cho nó là *MoveBetweenRoutes*.

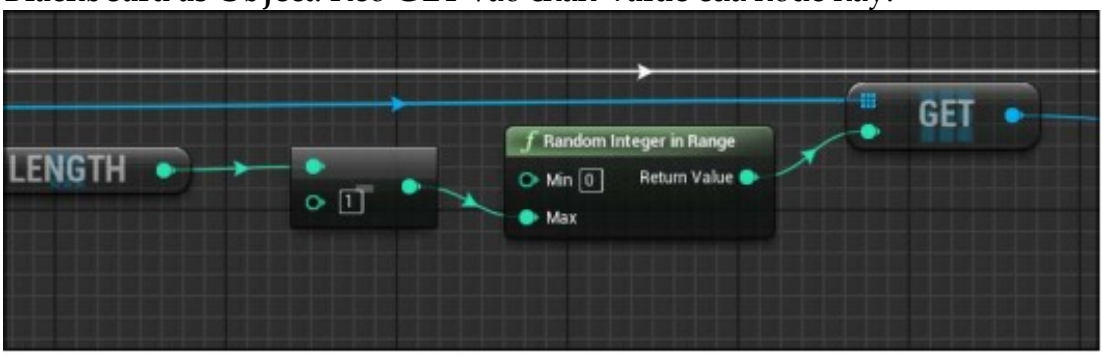
2. Mở *MoveBetweenRoutes* và điều hướng đến EventGraph.
3. Nhấp chuột phải và tìm kiếm **Event Receive Activation**. Hãy lấy từ **Owner Actor** và tạo một *pure cast* cho AIController. Tiếp theo, kéo từ chân **AsAIController** và tìm kiếm **ReceiveMoveCompleted**. Chỉ định event này để sử dụng sau:



4. Hãy tạo hai biến mới. Đầu tiên sẽ là **Blackboard Key Selector** có tên là *CurrentRoute*. Cái thứ hai sẽ là **AIController** có tên là *thisActor*. Cả hai biến đều chọn tùy chọn **Editable** trong panel **Details**.
5. Bây giờ, chúng ta sẽ thiết lập node **SET** *thisActor* và kéo từ chân **AsAIController** vào *thisActor*.
6. Tiếp theo, chúng ta sẽ lấy *CurrentRoute* và tạo node **Get Blackboard Value as Object**; sau đó, chúng ta sẽ tạo một node **IsValid** để kiểm tra xem đối tượng giá trị **Blackboard** có hợp lệ hay không, nếu không có nghĩa là nó chưa được thiết lập:

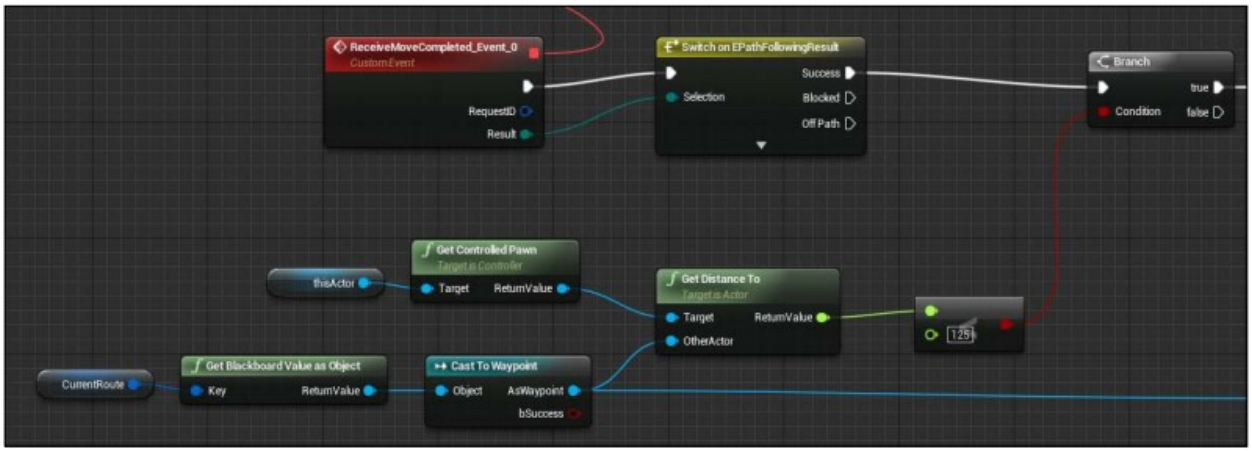


7. Kéo chân **Is Not Valid** từ node **Is Valid** và thả ra để tìm kiếm **Get All Actors of Class**. Thiết lập **Actor Class** thành **Waypoint**.
8. Kéo từ **Out Actors** và thả chuột ra để tìm kiếm **Length**; sau đó, trừ đi **1**. Nhấp chuột phải và tìm kiếm **Random Integer in Range**. Sau đó, cắm hiệu số vào chân giá trị **Max**. Tiếp theo, kéo từ **Out Actors**, thả chuột ra và tìm kiếm **GET**.
9. Cắm **Return Value** từ **Random Integer in Range** vào trong **GET**. Bây giờ, hãy kéo **CurrentRoute** vào sơ đồ. Kéo từ biến này, sau đó nhả chuột ra, rồi tìm kiếm **Set Blackboard as Object**. Kéo **GET** vào chân **Value** của node này:

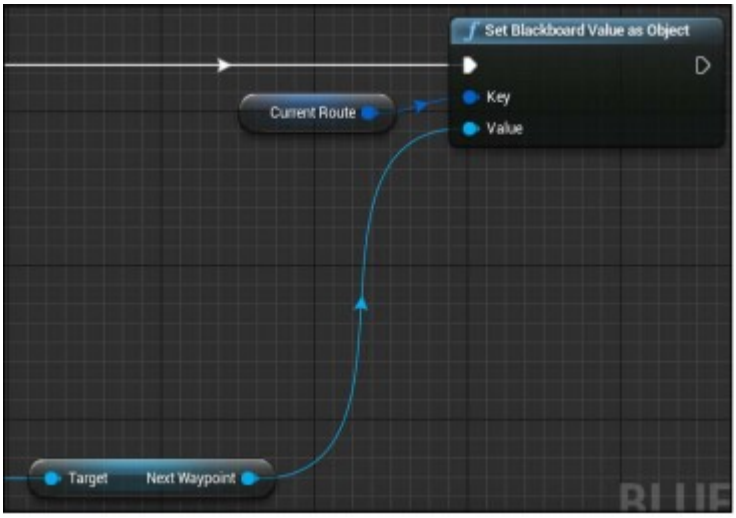


Bây giờ, một tuyến đường ngẫu nhiên được chọn khi bắt đầu.

10. Hãy tập trung trở lại event **ReceiveMoveCompleted** mà chúng ta đã tạo trước đó. Chuyển đổi **Result** với node **Switch on EPathFollowingResult** và từ **Success**, thả chuột ra để tìm kiếm **Branch**.
11. Kéo biến *thisActor* vào sơ đồ và sau đó kéo từ đó để tìm kiếm **Get Controlled Pawn**. Sau đó, từ **ReturnValue**, hãy kéo chân và tìm kiếm **Get Distance To**.
12. Kéo biến *CurrentRoute* vào sơ đồ và tìm kiếm **Get Blackboard Value as Object**. Tiếp theo, chuyển nó sang Waypoint và cắm **AsWaypoint** vào **OtherActor**.
13. Kiểm tra xem **ReturnValue** có nhỏ hơn **125** hay không. Cắm kết quả vào **Branch** đã tạo trước đó:



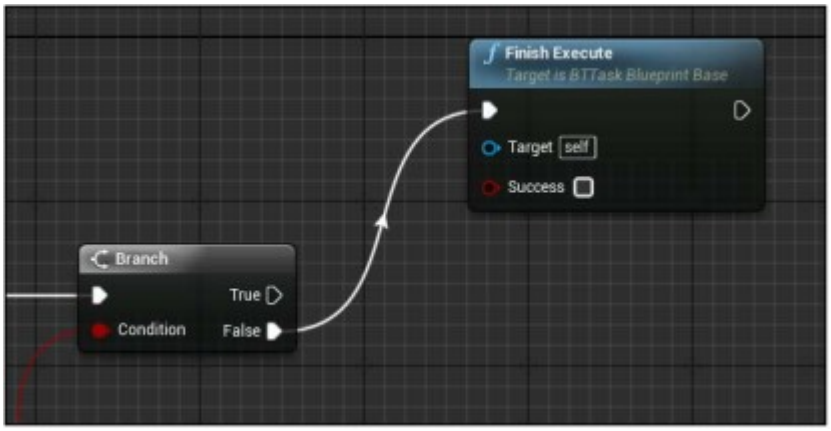
14. Kéo từ **AsWaypoint** và thả ra để tìm kiếm *NextWaypoint*.
15. Sau đó, kéo *CurrentRoute* và thả nó vào sơ đồ. Kéo chân và thả chuột ra để tìm kiếm **Set Blackboard Value as Object**. Sau đó, kéo chân từ *NextWaypoint* vào **Value** của node này:



Bây giờ, chúng ta có thể di chuyển đến tuyến đường tiếp theo khi đã di chuyển đủ gần tuyến đường hiện tại.

Điều thứ ba, chúng ta sẽ tạo là **Task** và nó được gọi là một chiếc lá trên cây. node này sẽ đặc biệt giúp AI xoay vòng và tấn công người chơi khi nó ở trong phạm vi. Thực hiện các bước sau:

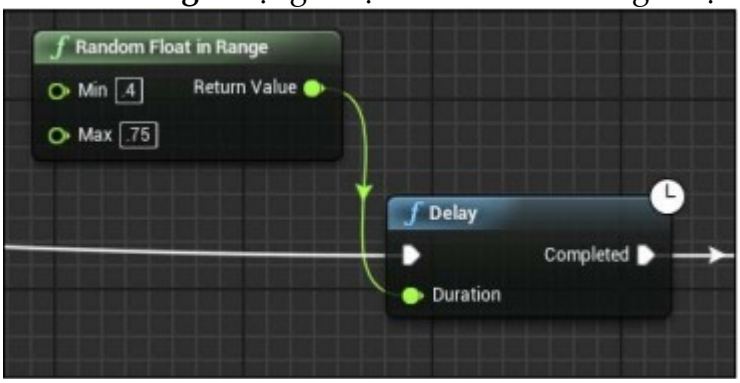
1. Nhấp chuột phải vào **Content Browser** và nhấp vào **Blueprint Class**. Sau đó, đi xuống **All Classes** và tìm kiếm **BTT**. Nhấp vào **BTTask_BlueprintBase** và nhấn nút **Select** để tạo **BTTask** mới. Đặt tên cho nó là *AttackEnemy*.
2. Mở *AttackEnemy* để mở sơ đồ EventGraph.
3. Nhấp chuột phải và nhả để tìm kiếm **Event Receive Execute** và tạo node này. Bây giờ, chuyển từ **Owner Actor** sang **AIController**.
4. Từ node **Cast To AIController** vừa mới được tạo, hãy tạo node **Branch**. **False** sẽ đi đến node **Finish Execute**. Nó cũng sẽ trả về **False** cho **Success**, bỏ chọn cho tham số **Success**:



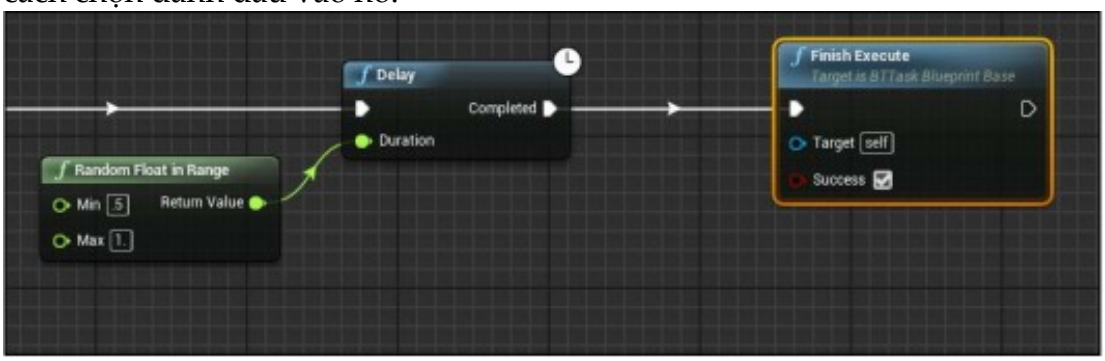
5. Tạo **Blackboard Key Selector** có tên *TheAttacker*, sau đó kéo *TheAttacker* vào sơ đồ, kéo chân và thả chuột ra để tìm kiếm **Get Blackboard Value as Object**. Sau đó, chúng ta sẽ kéo chân và thả chuột để chuyển sang ThirdPersonCharacter. Sau đó, **Success** sẽ được

cắm vào **Condition** của node **Branch** mà chúng ta đã tạo ở Bước 4.

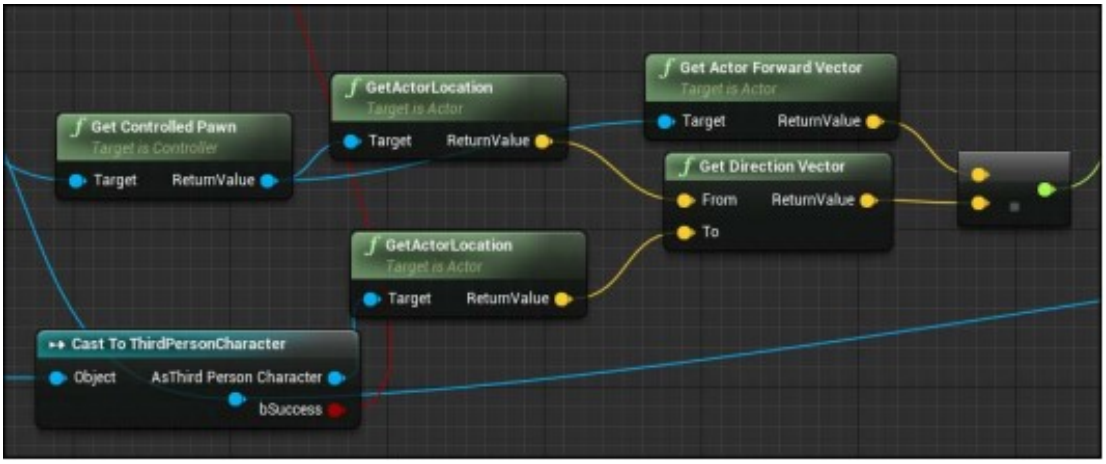
6. **True** sau đó được kéo và thả chuột ra để tìm kiếm node **Delay**. Trong tham số **Duration**, chúng ta điền một số ngẫu nhiên vì vậy chúng ta tìm kiếm node **Random Float in Range**. Đặt giá trị **Min** thành **0.4** và giá trị **Max** thành **0.75** trên node:



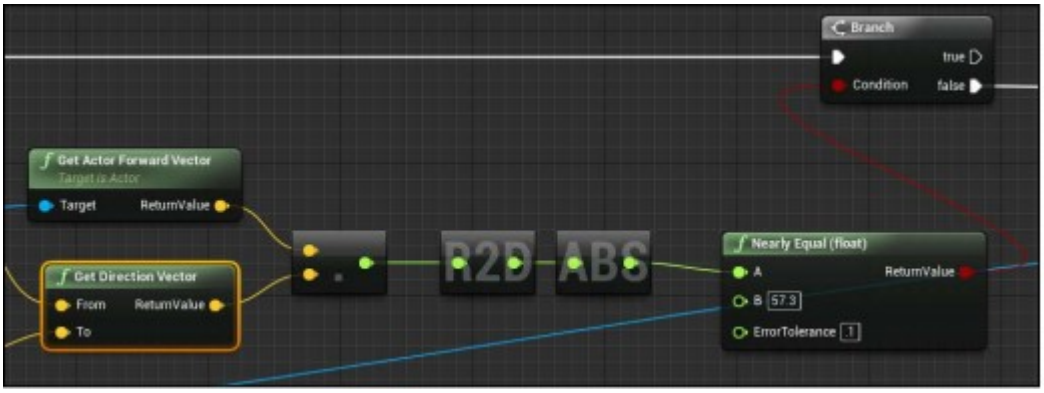
7. Kéo từ **AsAIController**, thả chuột ra và sau đó tìm kiếm **Laser from Controller**. Bây giờ, kéo từ **Hit Actor**, thả và tìm kiếm **Is Valid**. Kéo **Is Not Valid** và thả chuột ra để tìm kiếm **Finish Execute**.
8. Kéo từ **Is Valid** và tìm kiếm **Apply Damage**. Sau đó, cắm **Hit Actor** vào **Damaged Actor**. Cắm node **AsAIController** của chúng ta vào **Damage Causer**.
9. Kéo từ **Apply Damage** và tìm kiếm **Delay**. Nhấp chuột phải, tìm kiếm **Random Float in Range**, sau đó cắm **Return Value** vào **Duration** của node mà chúng ta vừa tạo. Đặt giá trị **Min** thành **0.5** và giá trị **Max** thành **1.0**.
10. Sau node **Delay** kéo và tìm kiếm node **Finish Execute**. Đánh dấu **Success** là true bằng cách chọn đánh dấu vào nó:



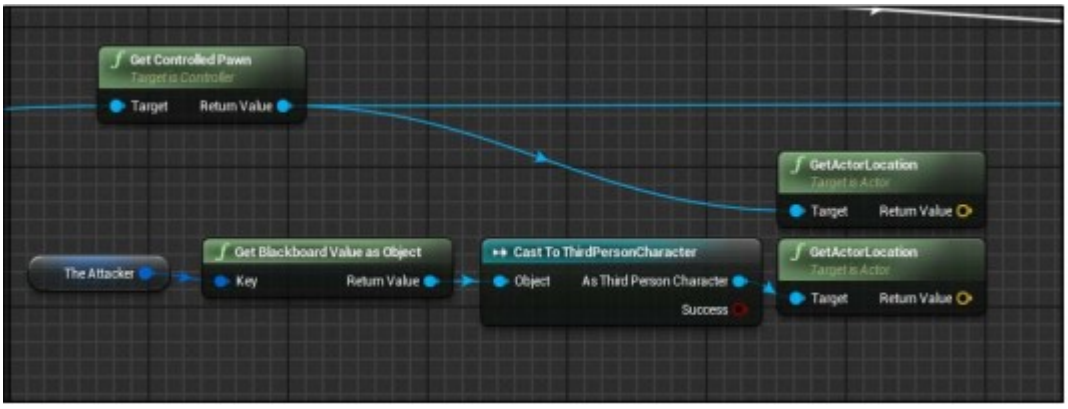
11. Bây giờ, nhấp chuột phải vào biểu đồ và tìm kiếm **Event Receive Tick**. Sau đó, chuyển **Owner Actor** sang **AIController**. Từ đây, chúng ta có thể lấy từ chân **AsAIController** và thả chuột ra để tìm kiếm **Get Controlled Pawn**. Sau đó, tạo node **Get Actor Location** và node **Get Direction Vector** liên kết với nhau.
12. Kéo từ **AsThirdPerson Character** và nhả chuột ra để tìm kiếm **GetActorLocation**. Sau đó, cắm **ReturnValue** vào chân **To** của **Get Unit Direction Vector** đã được tạo ở bước trước.
13. Kéo từ **ReturnValue** của **Get Controlled Pawn** và thả chuột ra để tìm kiếm **Get Actor Forward Vector**. Sau đó, từ **ReturnValue**, hãy phát hành và tìm kiếm node tích vô hướng **dot product** (**.**).
14. Cắm **ReturnValue** từ node **Get Unit Direction Vector** vào chân **B**:



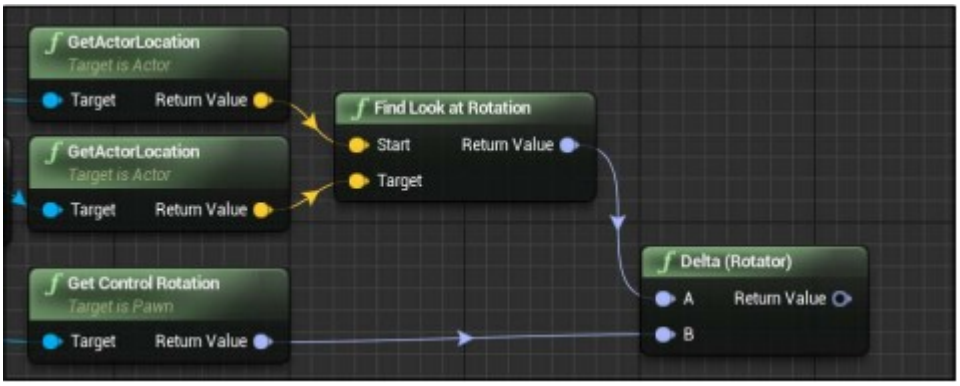
15. Lấy kết quả của tích vô hướng và nhả chuột ra để tìm kiếm node **Radians to Degress**. Sau đó, kéo và thả chuột để tìm kiếm **Absolute**. Cuối cùng, kéo kết quả và thả ra để tìm kiếm **Nearly Equal (float)**. Cắm đầu ra từ node **Absolute** vào chân **A**.
16. Giá trị **B** là **57,3** và giá trị **ErrorTolerance** là **0.1**. Điều này có nghĩa là phạm vi của chế độ xem gấp hai lần giá trị của **B**. Sau đó, kéo **ReturnValue** và thả chuột ra để tìm kiếm node **Branch**:



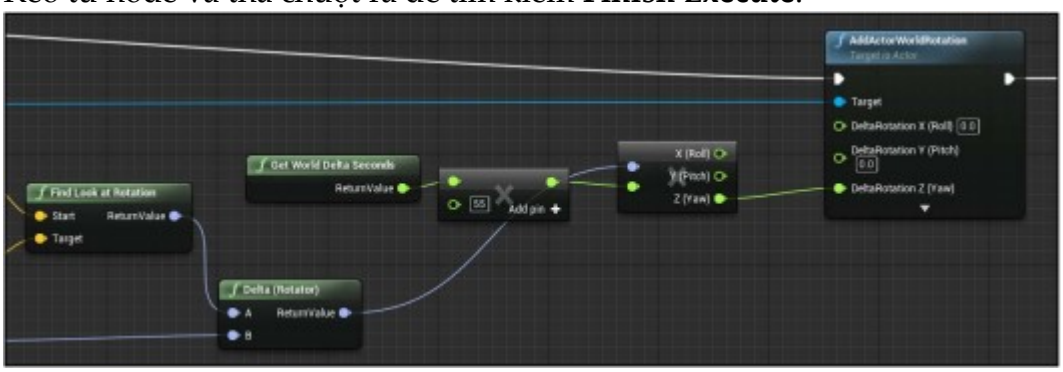
17. Bây giờ, kéo từ **AsAIController** và thả chuột, sau đó tìm kiếm **Get Controlled Pawn**. Sau đó, từ **Return Value**, kéo và thả để tìm kiếm **AddActorWorldRotation**. Nhấp chuột phải và tách cấu trúc.
18. Từ node **Return Value** của node **Get Controlled Pawn**, chúng ta sẽ kéo và thả chuột để tìm kiếm **GetActorLocation**.
19. Kéo biến **The Attacker** xuống và thả ra để tìm kiếm node **Get Blackboard Value as Object**, và cuối cùng chuyển nó thành **ThirdPerson Character**. Sau đó, từ **As ThirdPersonCharacter**, kéo chân và thả chuột ra để tìm kiếm **GetActorLocation**:



20. Từ **Return Value** của node **Get Controlled Pawn** và cắm vào chân **Target** của **GetActorLocation**; sau đó, tìm kiếm **Find Look in Rotation**.
21. Cắm **Return Value** từ Bước 19 vào **Target** của node **Find Look in Rotation**.
22. Kéo từ **Get Controlled Pawn** và thả chuột ra để tìm kiếm **Get Control Rotation**.
23. Kéo từ **Return Value** của node **Find Look in Rotation** và thả chuột ra để tìm kiếm node **Delta (Rotator)**.
24. Cắm **Return Value** của node **Get Control Rotation** vào chân **B** của **Delta (Rotator)**:



25. Kéo từ **Return Value** của node **Delta** và **ScaleRotator** với giá trị kết quả của **Get World Delta Seconds** nhân 55; Sau đó, tách đầu ra của **Scale Rotator** và cắm **Z (Yaw)** vào **DeltaRotation Z (Yaw)** của node **AddActorWorldRotation**.
26. Kéo từ node và thả chuột ra để tìm kiếm **Finish Execute**:



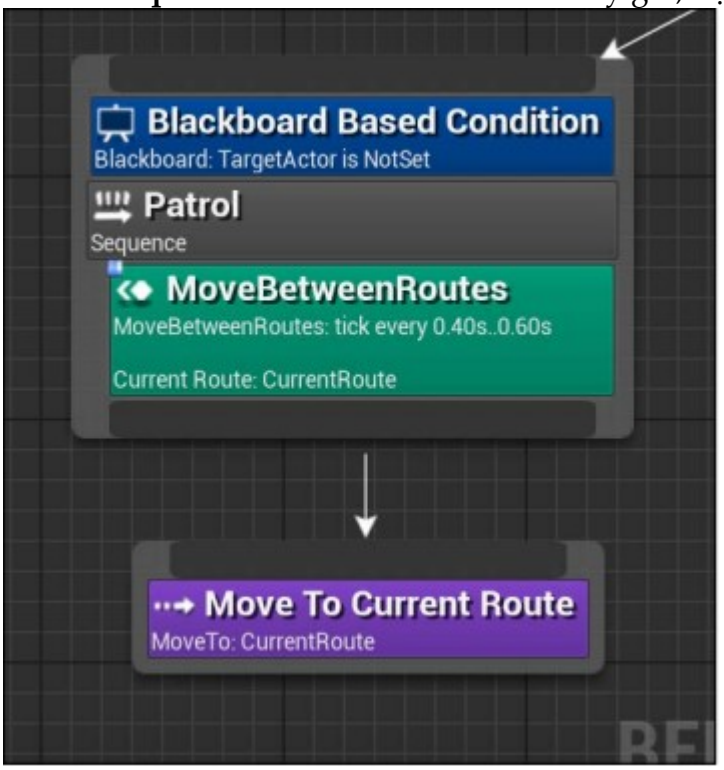
Bây giờ chúng ta đã hoàn thành, AI sẽ xoay về phía chúng ta bất cứ khi nào chúng ta ở ngoài tầm nhìn của nó.

Xử lý luống thực thi

Bây giờ, chúng ta đã hoàn thành tất cả các component của mình. Chúng ta đã cấu hình ba node tùy chỉnh của mình. Bây giờ, chúng ta chỉ cần quay lại BT của mình. Sau đó, chúng ta phải thiết lập ba trạng thái để AI hoạt động. Đầu tiên là *Patrol* (Tuần tra), chịu trách nhiệm chuyển AI sang tuyến đường tiếp theo. Thứ hai là *Chase* (Rượt), chịu trách nhiệm di chuyển AI trong bán kính của người chơi. Trạng thái cuối cùng là *Attack* (Tấn công), và trạng thái này sẽ bắn và xoay AI cho đến khi Target không còn trong khoảng cách.

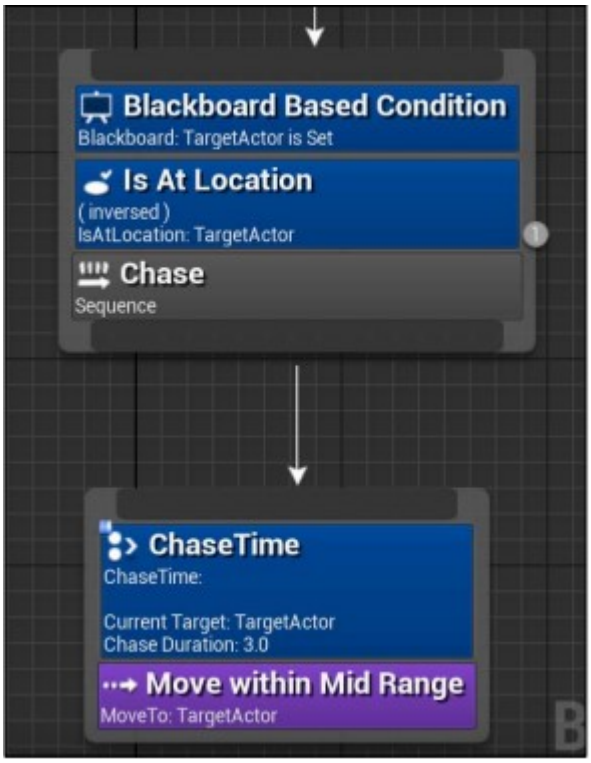
Bây giờ, hãy mở BT của *EnemyAI* theo các bước sau:

1. Kéo xuống từ **Root**, tìm **Selector**, và thiết lập **Node Name** thành *ChooseState*. Chúng ta sẽ xác định ba trạng thái riêng biệt.
2. Kéo **Choose State** và tìm **Sequence**. Bây giờ, nhấp chuột phải, chọn **Add Decorator** và tìm **Blackboard**.
3. Nhấp vào **Blackboard** và thiết lập **Key Query** thành **Is Not Set** và **Blackboard Key** thành **TargetActor**.
4. Nhấp chuột phải, nhấp vào **Add Service** và tìm **MoveBetweenRoutes**.
5. Nhấp vào **MoveBetweenRoutes** và đặt **CurrentRoute** thành *CurrentRoute*.
6. Kéo từ **Sequence** và tìm kiếm **MoveTo**. Bây giờ, đặt **MoveTo** thành *CurrentRoute*:

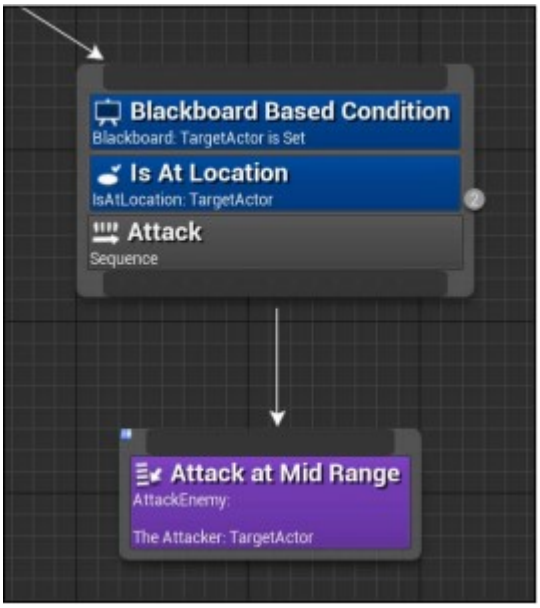


7. Kéo **Choose State** và tìm **Sequence**. Bây giờ, nhấp chuột phải và chọn **Add Decorator** để tìm **Blackboard**.
8. Nhấp vào **Blackboard** và đặt **Key Query** thành **Is Set** và **Blackboard Key** thành *TargetActor*.

9. Nhấp chuột phải và chọn **Add Decorator**; sau đó, tìm kiếm **Is At Location**.
10. Bấm vào **Is At Location** và đặt **Acceptable Radius** thành **600.0**, **Inverse Condition** thành **True** và **Blackboard Key** thành *TargetActor*. **Inverse Condition** sẽ trả về **True** nếu chúng ta nằm ngoài giá trị **Acceptable Radius**.
11. Kéo từ **Sequence** và tìm kiếm **MoveTo**. Nhấp vào **MoveTo** và đặt **MoveTo** thành *TargetActor*.
12. Nhấp chuột phải vào nút **MoveTo**, chọn **Add Decorator** và tìm kiếm *ChaseTime*.
13. Nhấp vào *ChaseTime* và đặt **Current Target** thành *TargetActor* và **Chase Duration** thành **3.0**:



14. Kéo từ **Choose State** và tìm kiếm **Sequence**.
15. Nhấp chuột phải, chọn **Add Decorator** và tìm kiếm **Blackboard**. Nhấp vào **Blackboard** và đặt **Key Query** thành **Is Set** và **Blackboard Key** thành *TargetActor*.
16. Nhấp chuột phải, chọn **Add Decorator** và tìm kiếm **Is at Location**. Nhấp vào **Is at Location** và đặt **Acceptable Radius** thành **599,99** và **Blackboard Key** thành *TargetActor*.
17. Kéo từ **Sequence** và tìm kiếm **Attack Enemy**:



Bây giờ, với tất cả các nút này ở đây, bạn có một kẻ thù AI hoàn chỉnh, giờ đây bạn có thể chiến đấu và tự mình chiến đấu! Chúc mừng bạn đã đi xa đến thế này!

Tóm tắt

Chương này chứa mọi thứ cần thiết để bắt đầu với một số AI ấn tượng. Chúng ta đã nói về Behavior Tree, giúp đưa ra quyết định cho AI của bạn dựa trên các tình huống mà nó gặp phải. Sau đó, chúng ta đã nói về việc tạo ra một cuộc tấn công tầm trung cho bạn, người chơi và AI sử dụng. Chúng ta thiết lập controller và bắt đầu cài đặt controlller cho nhiều chức năng hơn. Điều cuối cùng cần làm trước khi bắt mọi thứ chạy trong Bahavior Tree là thiết lập các điểm hẹn để AI tuần tra. Cuối cùng, chúng ta đã tích hợp mọi thứ vào Behavior Tree.

Bạn vừa học cách tạo dạng trò chơi AI cơ bản bằng cách sử dụng các công cụ có sẵn trong Unreal Engine 4.

Chúng ta đã hoàn thành rồi! Chúng ta đã tạo ra một số AI thú vị và đầy thử thách. Sau khi tìm hiểu tất cả tài liệu này về AI và Unreal Engine 4 cũng như cách chúng kết hợp với nhau, điều quan trọng là chúng ta dành một chương để xem lại mọi thứ đã được đề cập. Tôi cũng sẽ chỉ ra những điều lẽ ra chúng ta có thể làm khác đi. Sau đó, chúng ta sẽ tiến gần hơn đến một khởi đầu mới trong thế giới trò chơi AI!