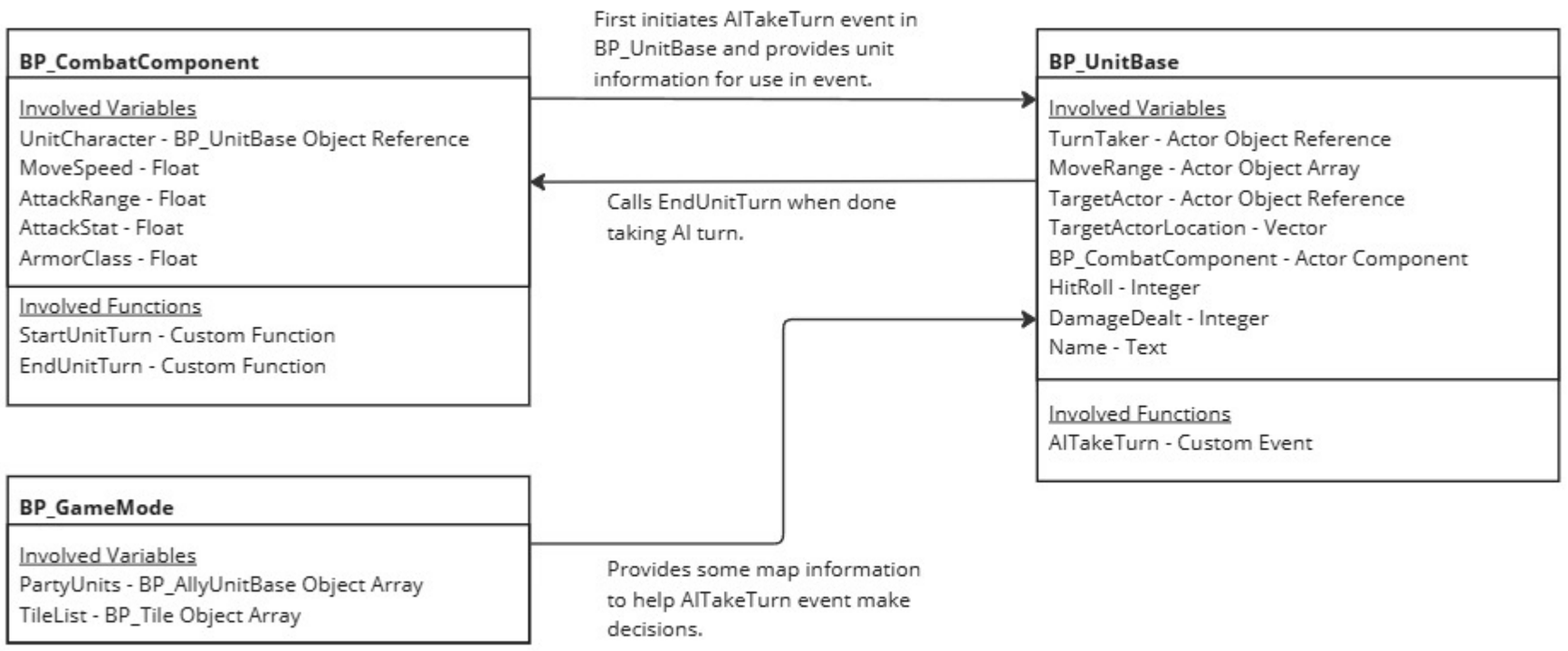
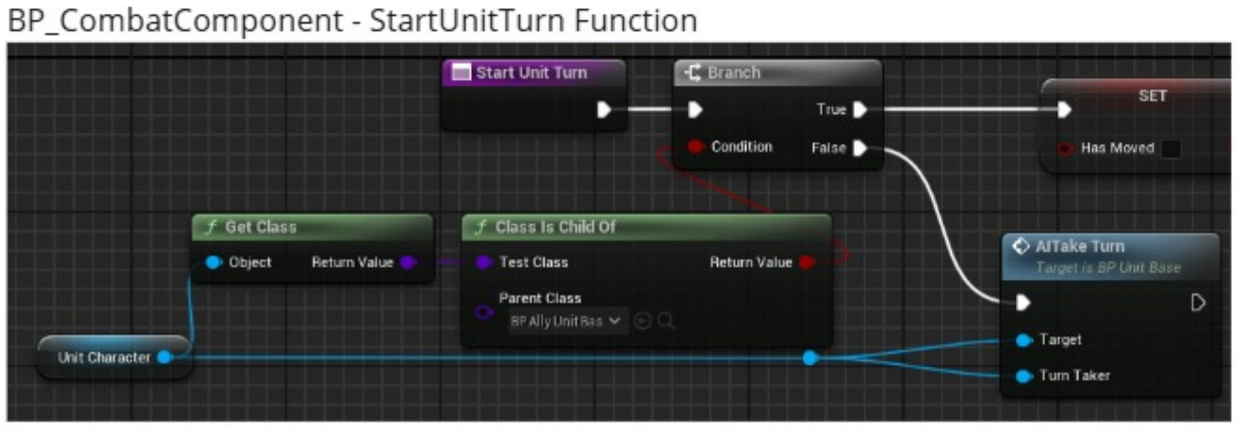
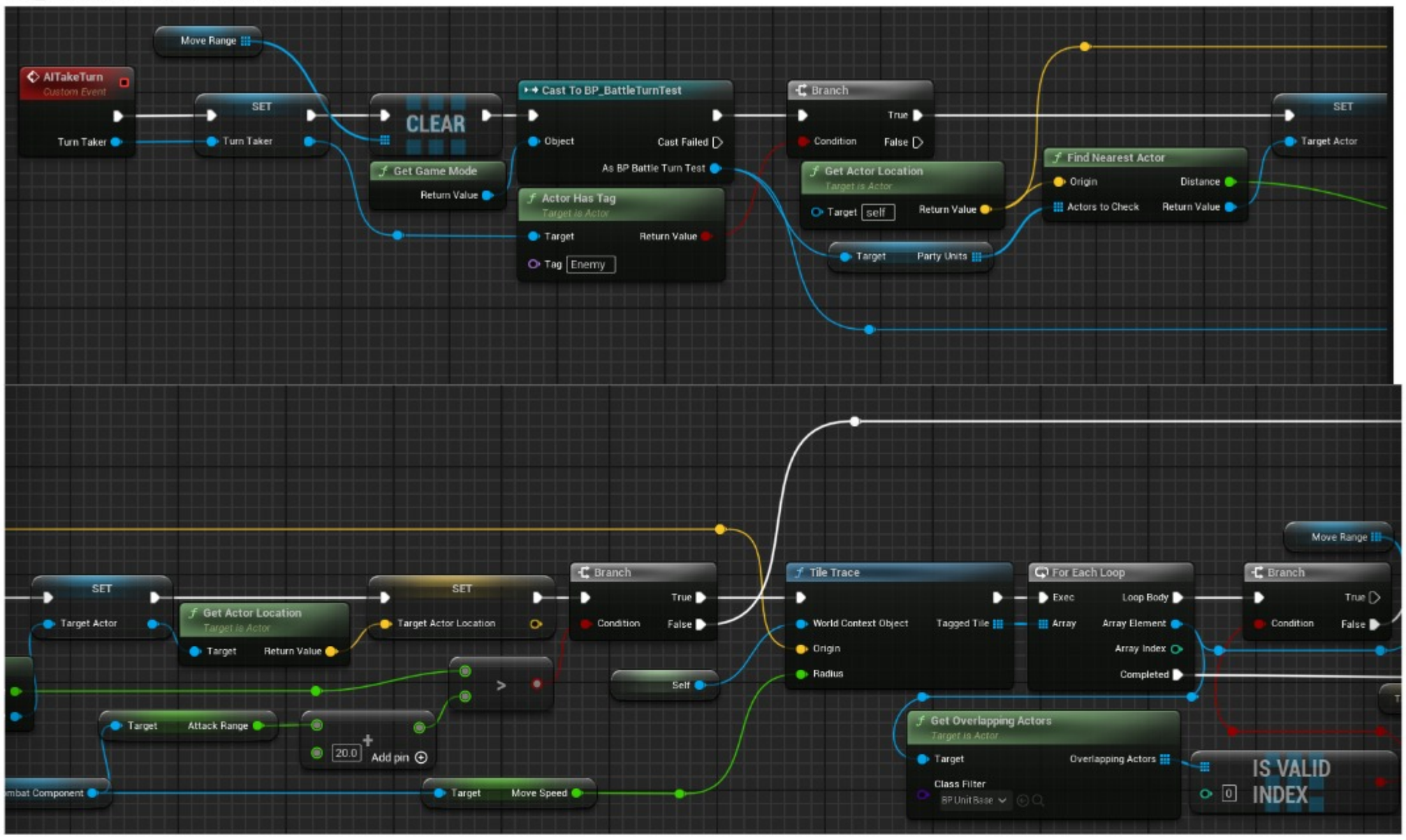
Feature - Enemy Turn AI
BP_UnitBase - Actor
BP_CombatComponent - Actor Component
BP_GameMode - Game Mode



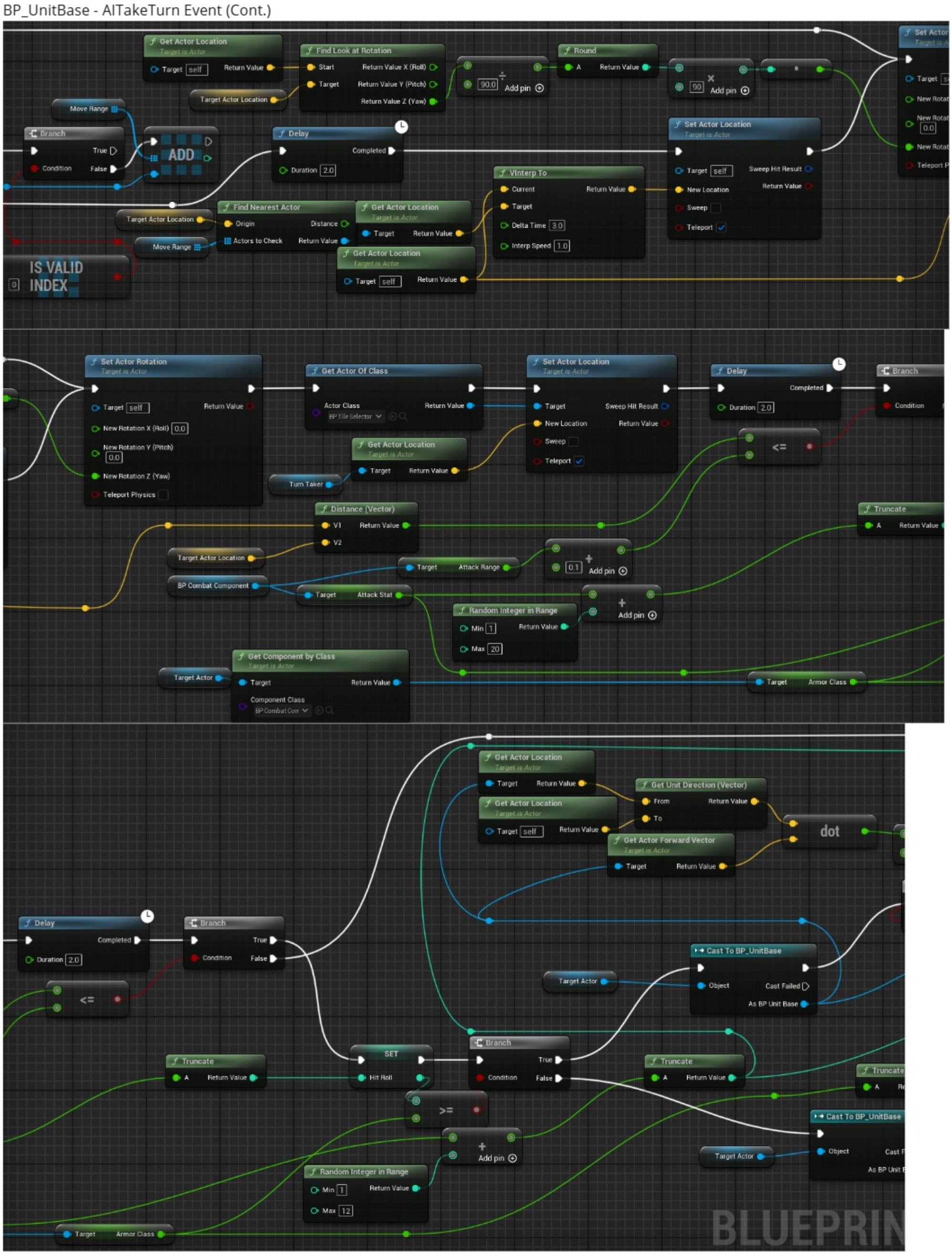
BP_CombatComponent - StartUnitTurn Function



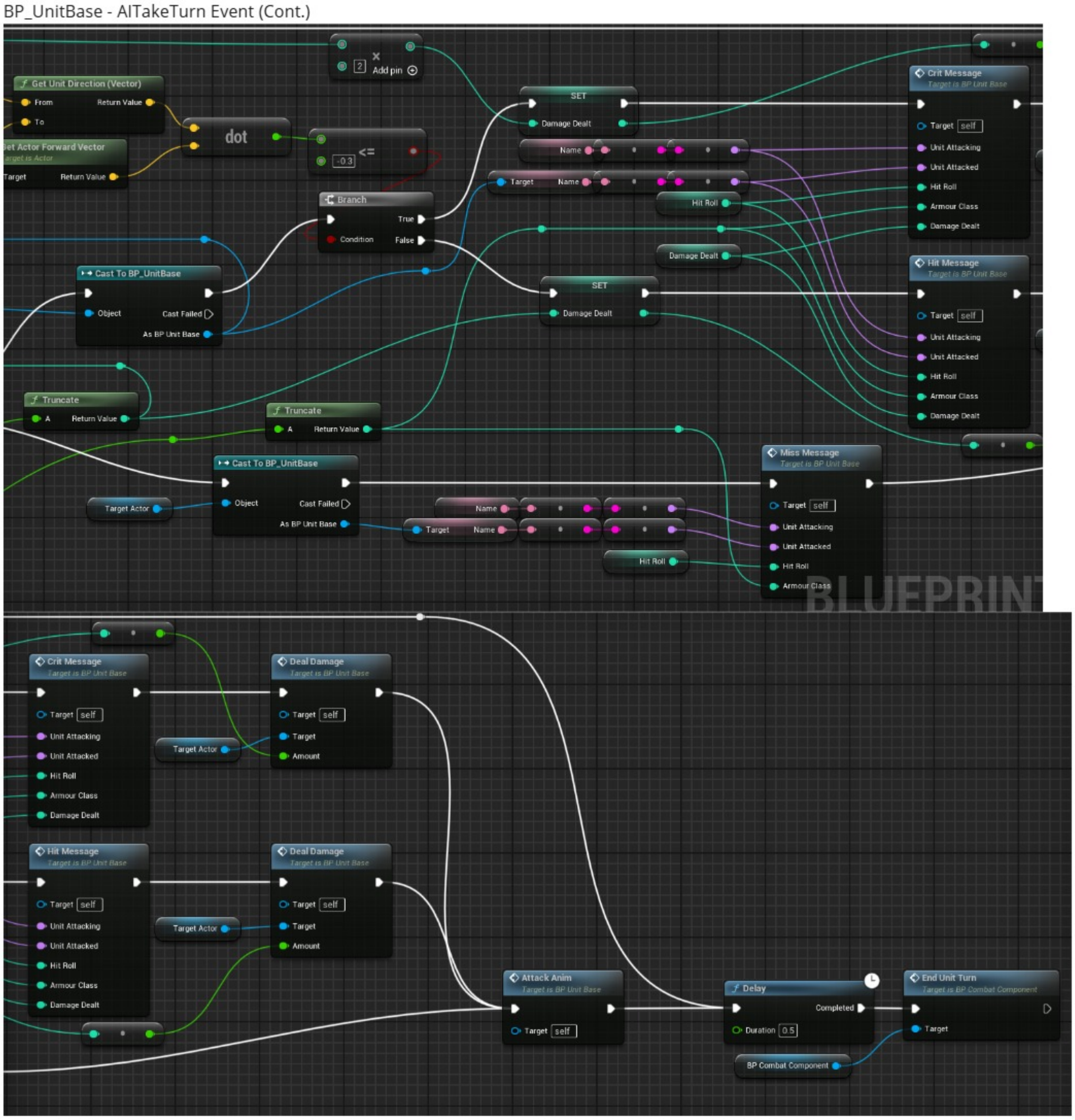
BP_UnitBase - AITakeTurn Event



BP_UnitBase - AITakeTurn Event (Cont.)



BP_UnitBase - AITakeTurn Event (Cont.)



Pseudo Code

BP_CombatComponent - StartUnitTurn Function

On Function called

If UnitCharacter Class is Child of BP_AllyUnitBase is false

Call AITakeTurn event with Target and TurnTaker input as UnitCharacter

BP_UnitBase - AITakeTurn Function

On Event called

Set Input variable TurnTaker as TurnTaker Actor Reference

Clear contents of MoveRange Actor Array

Cast to BP_GameMode as Game Mode

If TurnTaker has tag Enemy

Find nearest Actor to self of PartyUnits Array from BP_GameMode and Set as TargetActor

Get Location of TargetActor and Set as TargetActorLocation

If Distance of TargetActor is greater than AttackRange Float of BP_CombatComponent plus 20 is true

Call TileTrace function with input WorldContextObject as Self, Origin of Actor Location and Radius of MoveSpeed of BP_CombatComponent.

For Each element of returned TaggedTiles Array

If array of Overlapping Actors with Class BP_UnitBase does not have valid Index of 0.

Add element to MoveRange Array

On Completed, Delay 2 seconds

Set Actor Location of Self equal to VInterp To return value with Current input as Actor Location and Target as Actor Location of Nearest Actor to

. TargetActorLocation of MoveRange Array.

BP_UnitBase - AITakeTurn Function (Cont.)

Set Actor Rotation of Self with new Z Rotation equal to returned Z value of Find Look At Rotation from Actor Location to TargetActorLocation, divided by 90, rounded, and then multiplied by 90.
Get Actor of Class BP_TileSelector
Set Actor Location of returned Actor equal to Actor Location of TurnTaker.
Delay 2 seconds
If Distance between Actor Location and TargetActorLocation is less than or equal to the AttackRange Float of BP_CombatComponent plus 0.1 is true.
Set HitRoll Integer equal to AttackStat of BP_CombatComponent plus Random Integer between 1 and 20.
If HitRoll is greater than or equal to ArmorClass Float of TargetActor's BP_CombatComponent is true
Cast to BP_UnitBase as TargetActor
If Dot value of Unit Direction from Actor Location of returned BP_UnitBase reference to Actor Location of Self and Actor Forward Vector of returned BP_UnitBase reference is less than or equal to -0.3 is true.
Set DamageDealt Integer equal to AttackStat of BP_CombatComponent plus a Random Integer between 1 and 12, multiplied by 2.
Call CritMessage event with inputs UnitAttacking as Name, UnitAttacked as Name of returned BP_UnitBase from cast, HitRoll as HitRoll, ArmourClass as ArmorClass, and DamageDealt as DamageDealt.
Call DealDamage event with Target as TargetActor and Amount DamageDealt.
If Dot value of Unit Direction from Actor Location of returned BP_UnitBase reference to Actor Location of Self and Actor Forward Vector of returned BP_UnitBase reference is less than or equal to -0.3 is false.
Set DamageDealt Integer equal to AttackStat of BP_CombatComponent plus a Random Integer between 1 and 12
Call HitMessage event with inputs UnitAttacking as Name, UnitAttacked as Name of returned BP_UnitBase from cast, HitRoll as HitRoll, ArmourClass as ArmorClass, and DamageDealt as DamageDealt.
Call DealDamage event with Target as TargetActor and Amount DamageDealt.
If HitRoll is greater than or equal to ArmorClass Float of TargetActor's BP_CombatComponent is false
Cast to BP_UnitBase as TargetActor
Call MissMessage event with inputs UnitAttacking as Name, UnitAttacked as Name of returned BP_UnitBase from cast, HitRoll as HitRoll, and ArmourClass as ArmorClass.
Call AttackAnim event
Delay 0.5 seconds
Call EndUnitTurn Function of BP_CombatComponent.

Summary and Explanation

BP_CombatComponent - StartUnitTurn Function

This little excerpt from the StartUnitTurn Function serves to detect whether the Unit currently starting its turn is player controlled or not. If it were, it would proceed to enable the turn taking interface for the player, but for the purpose of explaining the AI Turn Taking feature, it is not. It checks this by seeing if the UnitCharacter is a child of BP_AllyUnitBase, a parent to all playable units. When it detects that it is not a child, it calls the AITakeTurn function within BP_UnitBase with the UnitCharacter as an input.

BP_UnitBase - AITakeTurn Function

This function serves to logically move step by step through the thought process behind a turn a player might take, translating that into predefined mechanics for an enemy unit to follow. A turn consists of two main actions; movement and attack. Therefore the AI will first move towards a potential target and then try to hit them with an attack.

The function opens by saving the UnitCharacter input as TurnTaker Actor Reference. It then clears any existing content with the MoveRange Array, clearing it up to be filled this turn. It casts to the BP_GameMode, as this holds some global arrays that will be necessary for the AI to make its decisions. The function has a check here to confirm if the TurnTaker has the 'Enemy' tag. This check is here solely in case this might be expanded to include allied units not controlled by the player, if perhaps the player teams up with a NPC character. This branch could then open up to a similar set of behaviour geared towards attacking enemy units instead of player units. For the purpose of this feature, we assume an enemy unit.

Next, the function searches within the PartyUnits Array from the BP_GameMode cast to earlier, and finds the Nearest Actor within that Array to the TurnTaker Actor Location, saving it as TargetActor. It gets the Actor Location of the TargetActor and predictably calls that TargetActorLocation. it then checks if the distance between the TurnTaker and the TargetActorLocation is greater than the AttackRange of the TurnTaker's BP_CombatComponent plus 20 for a little leeway. If the distance is greater, the TurnTaker should make an attempt to move towards the player, but if it is less than this value, the TurnTaker must already be within attacking range and can skip having to move. If the TurnTaker needs to move, the function calls the TileTrace function, a custom function that casts a Multi Box Trace for Objects around a certain point with a specified radius. A Self reference is given as the WorldContextObject and the TurnTaker Actor Location as the Origin, while the MoveSpeed becomes the Radius. This returns an Array of Tiles tagged by the box trace, which then enters a For Each Loop to check if these tagged Tiles are overlapping an Actor with Class BP_UnitBase, which means any Unit. If there is no Unit occupying that space, that Tile is then added to the MoveRange Array as a viable move location. When the Loop is Completed, there is a Delay here for 2 seconds to give the player a moment to follow what is happening, as their cursor will move to the active TurnTaker as part of the StartTurn function that originally calls the StartUnitTurn, so the cursor moves to the unit taking its turn and gives the player a moment while the above takes place before advancing. The Actor's location is then set to be a VInterp from the Actor's current location to the location of the Nearest Actor to the TargetActorLocation in the MoveRange Array, so moving to the tagged tile closest to the TargetActor, as close as the TurnTaker can get within its MoveSpeed limit, and the movement stage is mostly completed.

Regardless of whether the TurnTaker moves or not, the function then sets its Rotation to face the TargetActor. This happens even if the Unit doesn't move as it is possible the player moved a Unit behind the enemy to attack, so the TurnTaker will then spin around to face the player Unit. The function accomplishes this by changing only the Z Rotation value in Set Actor Rotation, setting it to the Look At Rotation Z value from the TurnTaker Actor Location to the TargetActorLocation, divided by 90, rounded, and then multiplied by 90. These final steps are to ensure it is always facing either up, down, left, or right, and not at angles, as the result will always be a multiple of 90. Once the Rotation is set, the function grabs the BP_TileSelector cursor that has the camera attached and sets its location equal to the TurnTaker Location so that the cursor follows the AI movements, making it clear to the player what is happening. Lastly we have another 2 second delay before the function enters into the combat portion of its turn.

BP_UnitBase - AITakeTurn Function (Cont.)

At the start of the combat portion, the function checks that the TurnTaker is actually within attacking range of the TargetActor. It does this by checking that the Distance between the TurnTaker Actor Location and the TargetActorLocation is less than or equal to the AttackRange Float of the TurnTaker's BP_CombatComponent plus 0.1 for a little leeway. If this is false, then the function will skip over combat since there is nothing it can do, but if true, the function determines the TurnTaker's HitRoll Integer value. In Dungeons & Dragons, when first deciding to attack, the player will roll a 20-sided die and add their own modifiers to that roll. This is emulated here as the function will grab the AttackStat from the BP_CombatComponent and add it to a random integer between 1 and 20, before setting that as the HitRoll Integer so that it is saved and only calculated once. If the HitRoll value is greater than or equal to the TargetActor's ArmorClass Float from its BP_CombatComponent, it is considered a hit. If a miss, the function starts by casting to the BP_UnitBase of the TargetActor. It then triggers the MissMessage event, inputting information from the TurnTaker and the returned cast of the TargetActor, printing a message like 'EnemyName attacked PlayerUnit and rolled a 6 against their 12 armour. Miss!'.

If the attack is a hit, the function will similarly cast to the BP_UnitBase of TargetActor, but will then enter another branch. Here the function will check if the dot value between the Unit Direction Vector from the TargetActor Location to the TurnTaker Location and the Actor Forward Vector is less than or equal to -0.3. The dot value returns a value between -1 and 1, with 1 meaning the TurnTaker is directly in front of the TargetActor, and -1 meaning directly behind, so -0.3 suggests enough of an angle so as to be out of sight and with a clear view of the TargetActor's back. If this is true, the attack will be considered a critical hit and damage will be doubled. The DamageDealt value is set as the AttackStat Float added to a random integer between 1 and 12, and then multiplied by 2 as a critical hit. The function then calls the CritMessage event, inputting information like names, HitRoll, DamageDealt, and ArmorClass. This event will print a message like 'EnemyName attacked PlayerUnit and rolled a 14 against their 12 armour. Critical hit! 20 damage dealt.'. It will then trigger the DealDamage event with the TargetActor as the targer, inputting how much damage should be dealt with the DamageDealt Integer. This event will handle actually applying the damage and checking whether the target has been killed or not, and updating health values.

If the attack is not from behind and therefore not a critical hit, the function will set the DamageDealt in the same way, just without multiplying the end result by 2. It will then call the HitMessage event similar to the CritMessage event, returning a message like 'EnemyName attacked PlayerUnit and rolled a 14 against their 12 armour. 10 damage dealt.'. It will call the DealDamage event in the same way to ensure the damage is applied. The final step of combat, regardless of whether the TurnTaker hits, crits, or misses, is to trigger the AttackAnim event, which will make the Unit player their attack animation before returning to Idle. Lastly, to wrap up the AI turn, there is a small 0.5 second delay for the player to process what has happened, and then the Unit will call the EndUnitTurn function, ending their turn and allowing the next turn to take place.

This function is quite flexible in that it allows for a wide variety of AI movement and attack types. It does not matter if the enemy has to charge and attack from up close, or wants to stop and shoot from range. Additionally, as mentioned earlier, this can be opened up to accommodate more than simply enemy turns, allowing the implementation of npc allied units. One could even implement abilities by inserting another branch just before the combat portion to check for abilities and triggering them within the Unit's own class, bypassing the rest of the combat portion as this would take place instead of an attack.