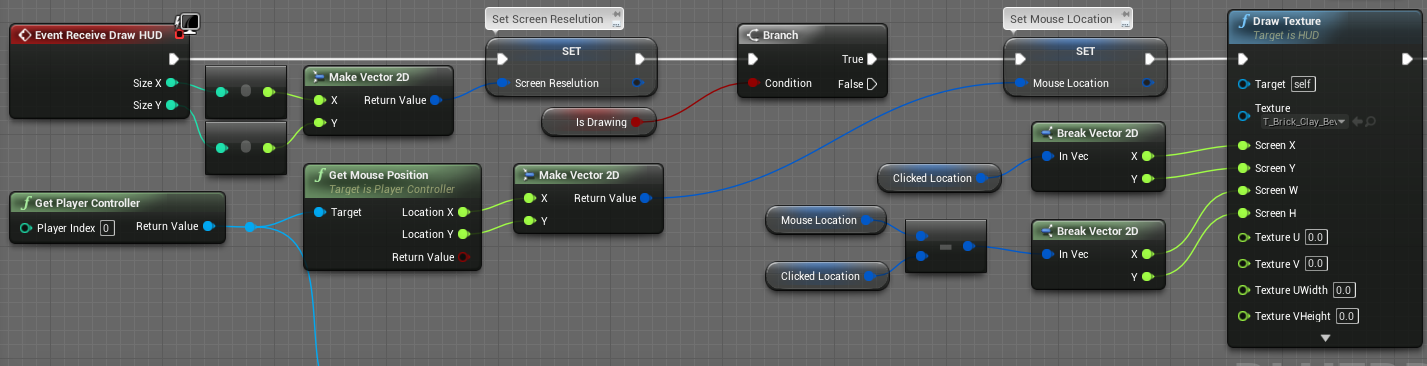
# Spil Ide

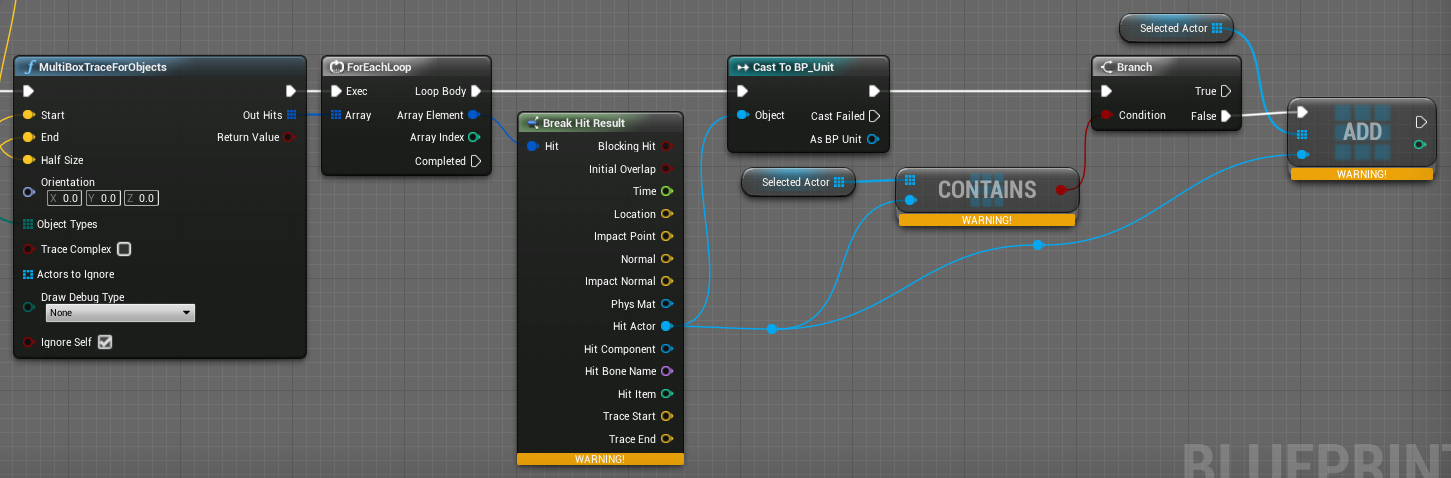
Vi vil under dette forløb forsøge om med at lavet en multiplayer real time strategi, målet er at 2 spiller skal være i stand til forbinde sig til et spil og kæmpe mod hinanden. Hver spiller skal være i stand til at bygge en eller flere bygninger samt to eller flere enheder. For at en spiller skal kunne vinde over den anden så skal de ødelægge modstanderens central bygning som give med i starten af spillet til begge spiller. Vi vil også forsøge os med at lave passive enheder hvis opgave vil være at samle ressourcer og bygge bygninger, dog vil denne del være lav prioriter. Der er umiddelbart ingen planer for a lave tech træ som vil gør det muligt at opgrader eller låse op for nye enheder.

# Multi og Single-enheds markering

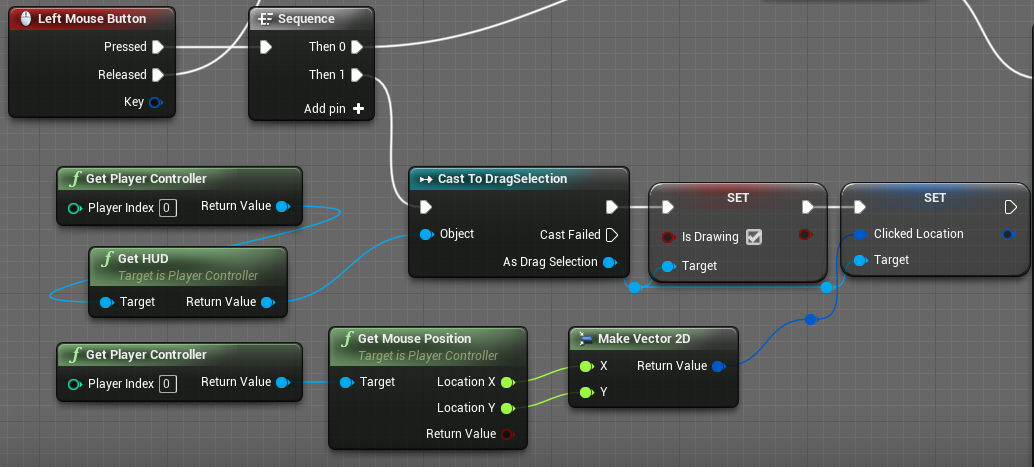
Dette er første del af at marker flere enheder, for at spilleren skal være i stand til at se hvilke enheder der bliver markeret tegnes en boks på skæremen. Det første der gøres er at kalde et nyt event som tillader os at tegne på skærmen, denne event har to integrer en x og en y som er spil skærmens størrelse. Dette konventer man om til en Vektor2D som sørger for at der tegnes inde for spil skærmen, herefter bruges en branch som spørger efter et nyt variable IsDrawing, på den måde kan man sørger for kun at tegne mens der trækket i musen. Hvis der tegnes skal der bruge en lokation for hvor man starte med at tegne, dette lave også om til en vector2D. Vi skal endvidere også bruge den nuværende lokation for at opdater den tegnet firkant. Screen x og y på metoden Draw Texture er der hvor texturen som skal lave boksen startes, det er her vi skal bruge vores Clicked location, Screen H og W er højden og bredden på vore texture, denne udregnes ved at subtraher den nuværende lokation som er mouse location med clicked location.



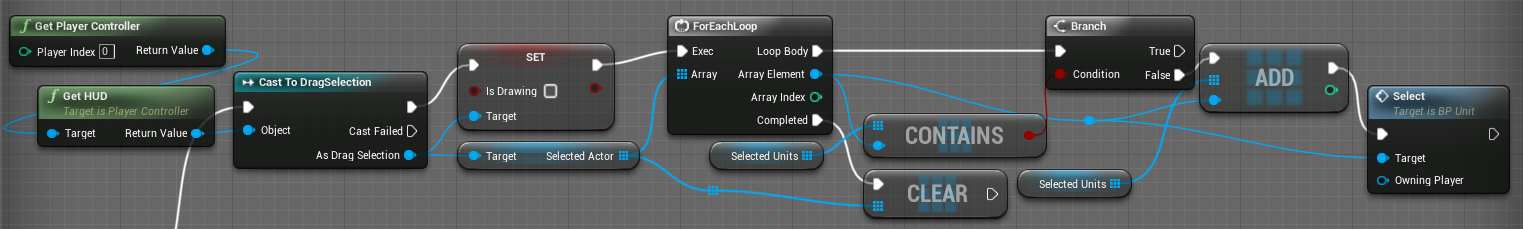
Når man så har tegnet bokse skal de enheder som er inde for boksen grænser identificeres, dette gøre vi ved at bruge metoden multi box trace for objects, denne finde alle objekter inden for dens grænser og via den tilføjer man den til et array, hvis kun noget specifikt ønskes at markers caster man det til den ønsket type.



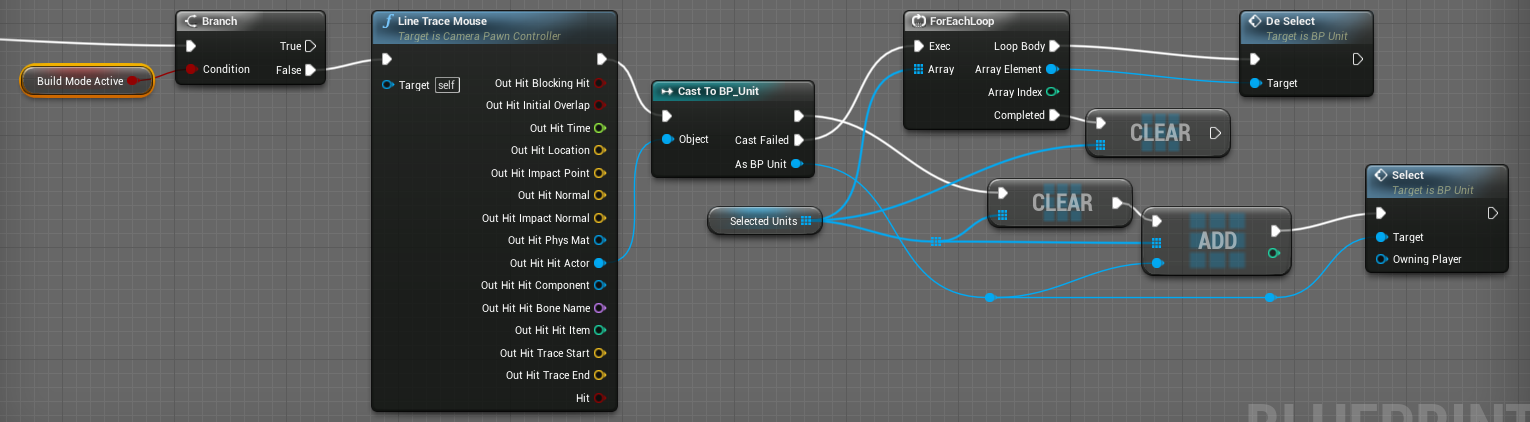
Når man så har de forrige metoder klar, kan man begynde at binde det til ønske input fra spilleren. Når der her trykker på venstre museknap vil den kalde den nødvendige metode, heri sætter vi også de nødvendig variabler som IsDrawing og clicked location.



Når spillerne endelig giver slip på venstre museknap sker anden del af den forrige metode, IsDrawing bliver sat til at være falsk hvilket betyder at boksen forsvinder. Sammen tid med at boksen forsvinder vil de enheder som er inden for den grænse også blive tilføjet til et nyt array af enheder og den gamle array som tilføjer vores markerings boks vil blive tømt.

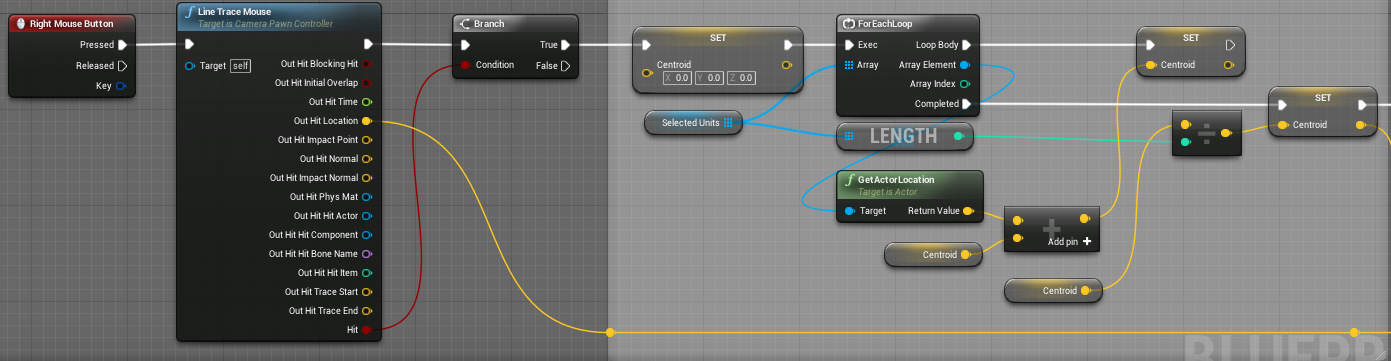


Det skal også være muligt for spilleren at vælge enkelte enheder at gangen, dette gøre vi ved at se på hvor musen er og laver en linje ned gennem verden og ser på om en enhed blivr ramt af linjen. Når vi marker en enkelt enhed ønskes der også at alle andre enhed ikke længer er markeret, derfor tømme vores array af enheder inden den enkelte enhed tilføjes. Der skal også gøres muligt at ikke have enheder markeret, så nar man klikker et sted på skærmen og det ikke ramme en unit vil castet fejl og arrayet tømmes for enheder.

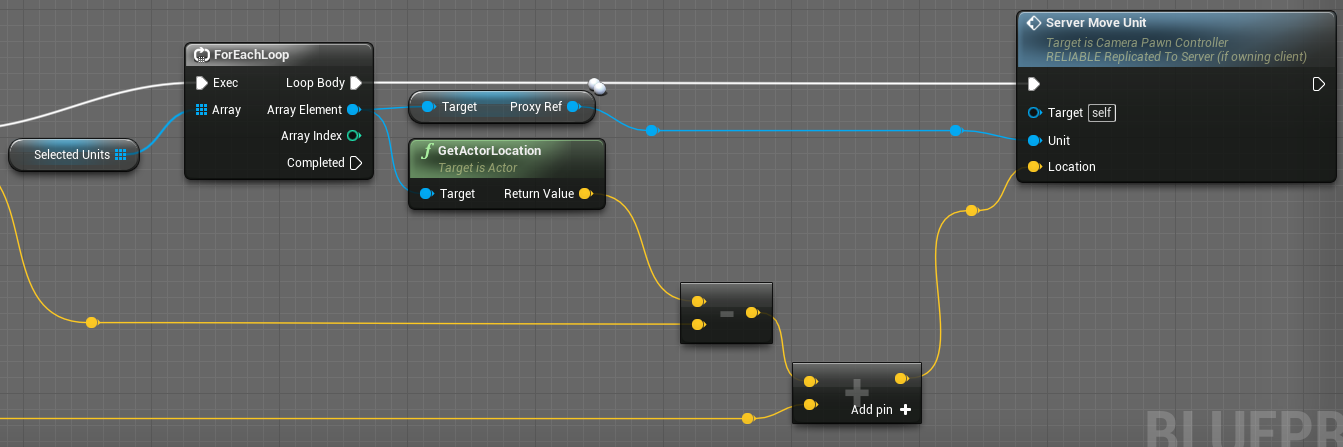


# Bevægelse

For at får spilleren til at bevæg sig bruges højre museknap, når knappen trykkes bliver der laver en linje fra cameraets position ned til det sted hvor der trykke på skærmen. Hvis så en eller flere enheder er markeret vil der blive sendt en besked til alle enheder om at de har fået en ny location de skal bevæg sig til. Dette giver dog umiddelbart et problem da alle enheder så vil prøve at stiller sig på den præcise samme location, derfor laves der et midtpunkt mellem alle de markeret enheder og dette gøres til der sammenligs punkt. Dette gøre ved at oprette et nyt variable centroid, denne sætte ved at loop igennem alle de markeret enheders location og lægge dem sammen, når den er færdig divider man det samlet resultat med antal enheder.



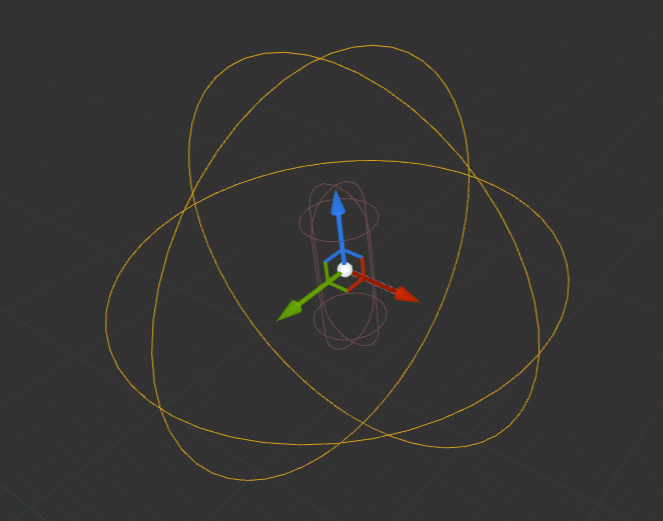
For så at få enheden til at bevæg sig i den korrekte retning subtraheres den enkelte enhedens nuværende location med centroid, derefter lægges den til den location som man får når man trykker på skærmen. Når vi endelig har dette kan man sende det til *ServerMoveUnit* som skal bruge denne information, der sende også en reference fra den proxy enhed som sørger for at serveren opdaterede position for begge spiller, dette bliver uddybet i afsnittet server meddelelser.



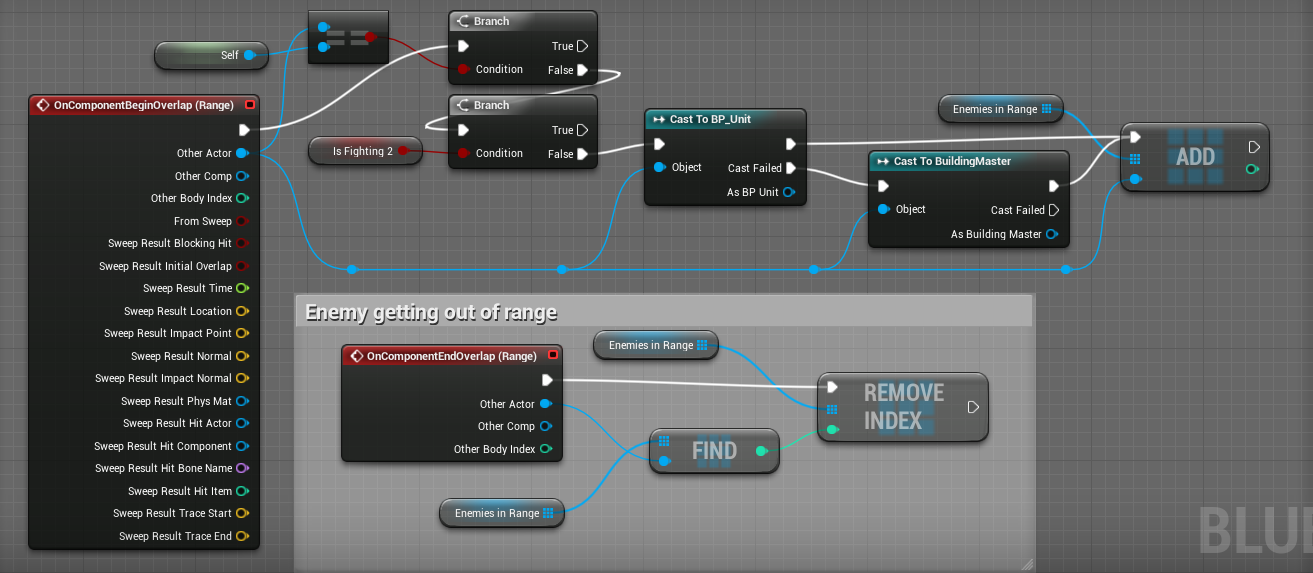
# Kamp mellem enheder

Når der laves kamp mellem enheder er der mange ting der skal tages i betragtning og er derfor også en af de mere komplekse del af spille. Der skal for eksempel tags højde for enheden skudvidde, hvilken modstander den skal angribes og hvornår, finde en ny fjende når den gamle dør samt en række af andre ting.

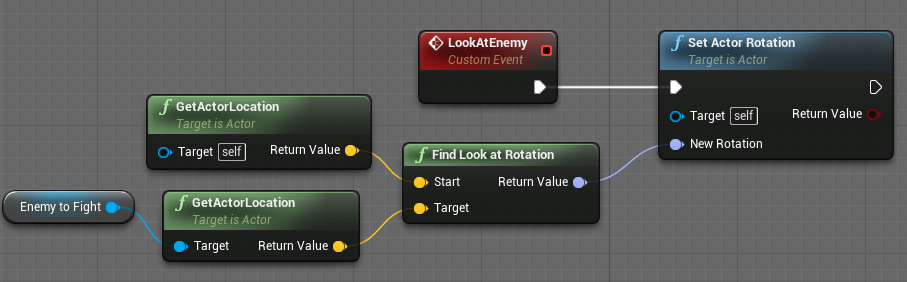
Det første der skal gøres er at tjekke på om noget er inden for rækkevidde dette gøres ved at lave en kollisions sfære på vores enheder, man kan nu bruge denne sfære til at se på om den anden enhed kollider hvis den gør der kan den angribes.

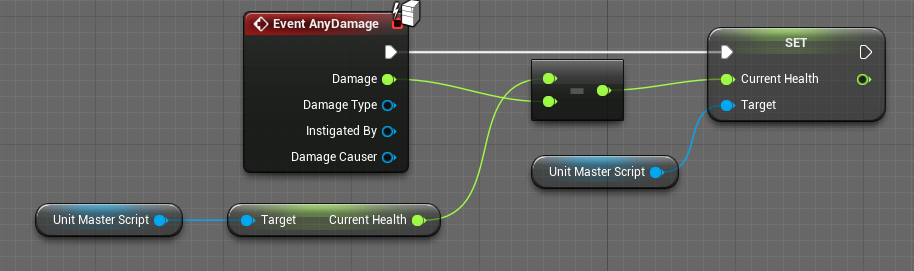


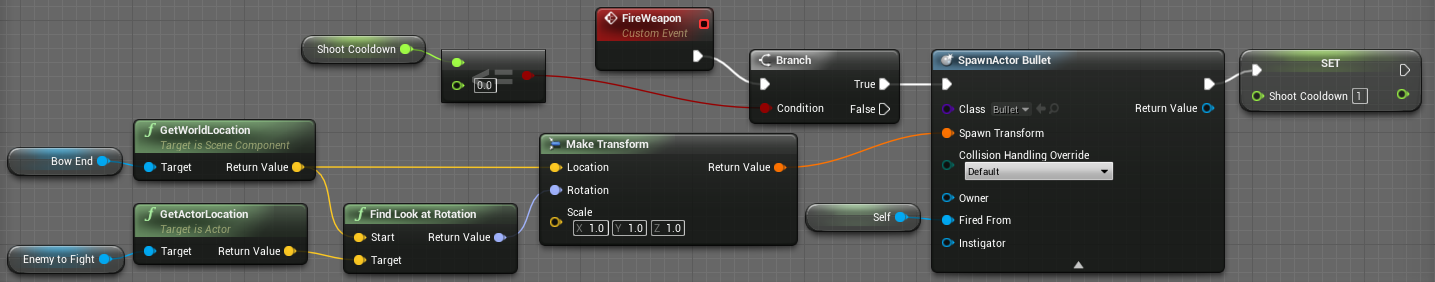
For at fjerne og tilføje enheder bruges en begivenhed som er tilførende kollisions komponentet som hedder *On Component Begin Overlap* og *On Component End Overlap*. Nå kollisionen begynder tjekkers der først om den kollider med sig selv, da dette er muligt, hvis kollisionen ikke sker med sig selv skal den kontroller om det den kollider med er den del af dens egen hold. Når begge af disse statements er falske kan en enhed eller bygning tilføjes til er array af enheder i rækkevidde. Hvis *On Component End Overlap* bliver aktiveret findes den specifikke enhed og fjernes fra arrayet.



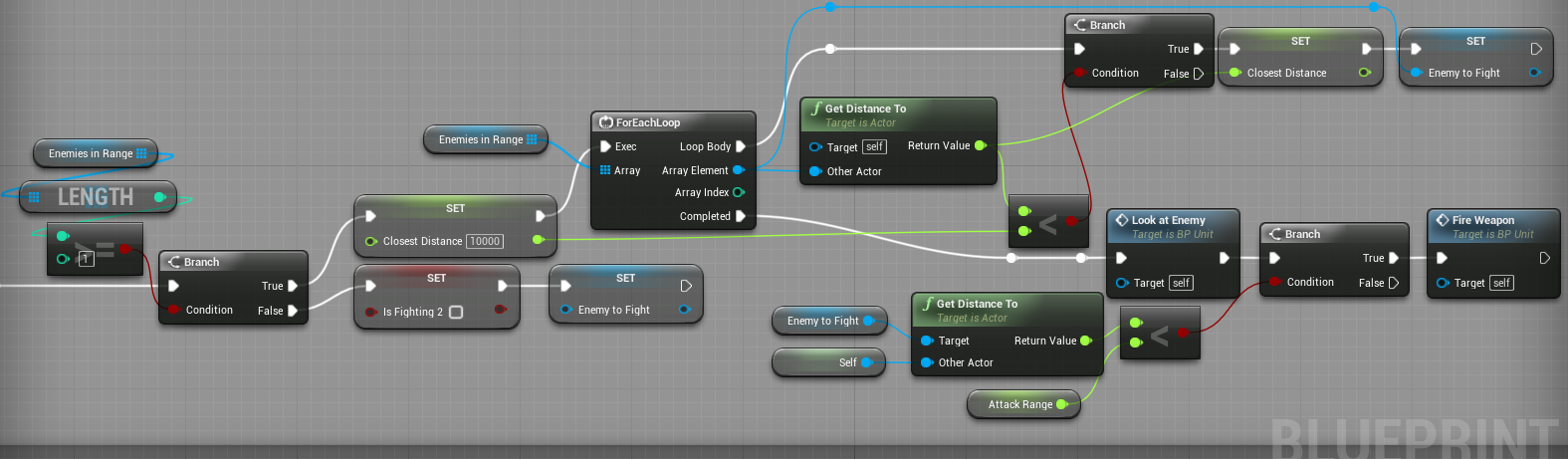
*Look at Enemy* er et event som sørger for at den kigger på den enhed som den skyder på, hertil bruger den *set Actor Rotation*.



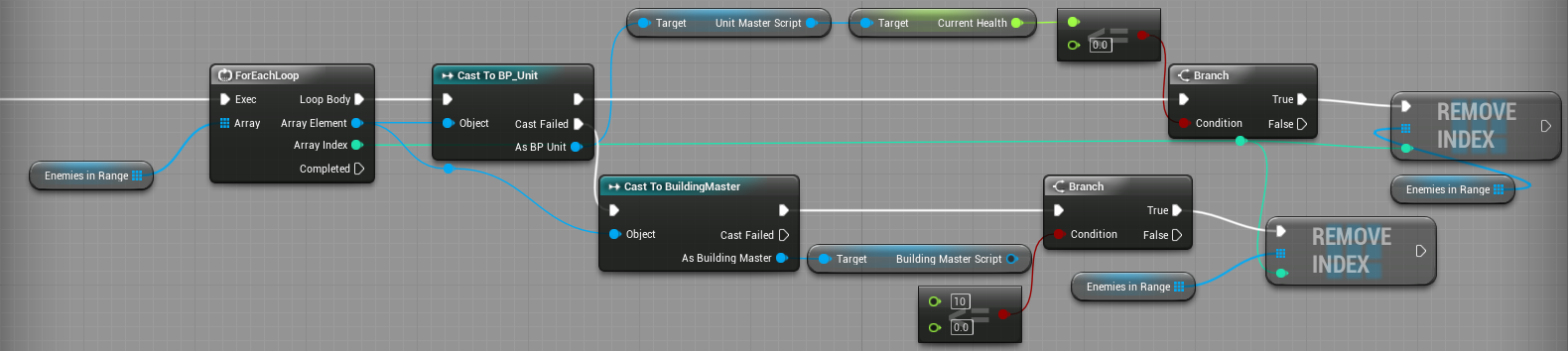
*Event anyDamage* bruges til at ændre vores health når den bliver ramt af noget som giver damage, her hentes en reference fra *unit master scrips*.som indeholder disse variabler.

Den sidste event som skal bruges inden det hele skal slås sammen er *fire Weapon*, denne bruges til at lave et ny instans af en kugle som skal affyres mod fjenden. Når den gøre dette tjekke den om dens nedkøling er færdig, dette gøres så der ikke bliver lave nye kugler konstant. Når kuglen bliver lavet vil den automatisk bevæg sig i den retning den blev skud imod, derfor skal der bruges et *spawn transform* som kan bestem hvorfra og i hvilken retning den bliver lavet. Lokationen bestemmes af en komponent som ligge på enheden og retning af den fjende den nu kigger på.

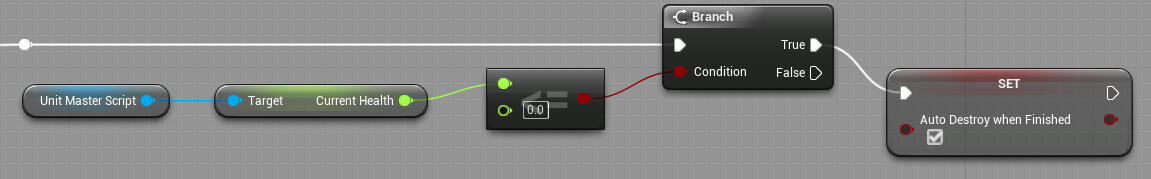
Nu hvor vi har de nødvendige metoder, kan der endelige lave noget funktionalitet for kamp. Det første der gøre er at tjekke på om der er noget at skyde på, så man tjekker igennem Arrayet af fjendtlige enheder i rækkevidde og vis dens længde er stører end eller lige med 1 så må der være noget at skyde på. Når vi gøre dette vil vi finde den enhed der er tættes på, ved at sammenligne afstanden fra den selv og alle enheder i rækkevidde. Når den endelige har fundet enheden der er tættest på bruger den *look at enemy* til at vende sig mod fjenden og begynder at skyde ved at bruge *Fire Weapon.*



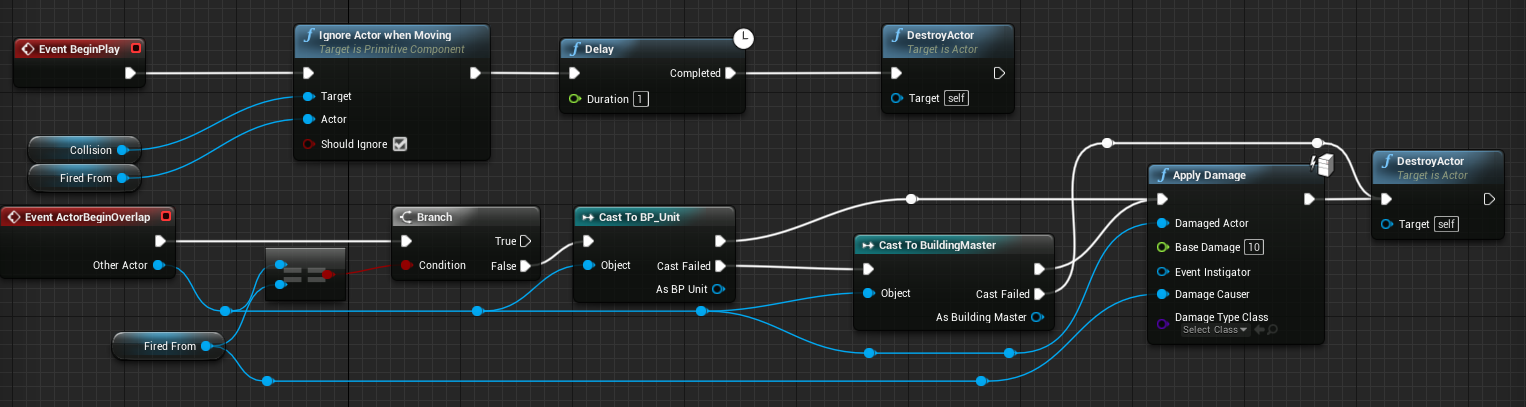
Nu mangles der bare at gøres to ting, at tjekke på om fjenden er dø eller en bygning er ødelagt og om man selv er død. Det der skal gøres er at tjekke på om fjenden er død, dette gøre vi ved at se alle enhed i vores array igennem og se om der liv er under eller lige med nul hvis dette argument er sand vil den specifikke enhed bliver fjerne fra lisen af sete fjender, det samme gøres på bygninger.



Til sidst er der en funktion som tjekker på enheden egen liv hvis de kommer under 0 eller er lige med vil enheden bliver fjernet fra spillet når den ikke længer har reference eller kald liggende, dette sørger *auto Destroy when Finished,* for som er en del af unreal. Grunde til at man ikke bare kan buge destroy er at der så vil være null referencer til den fjernet enhed.



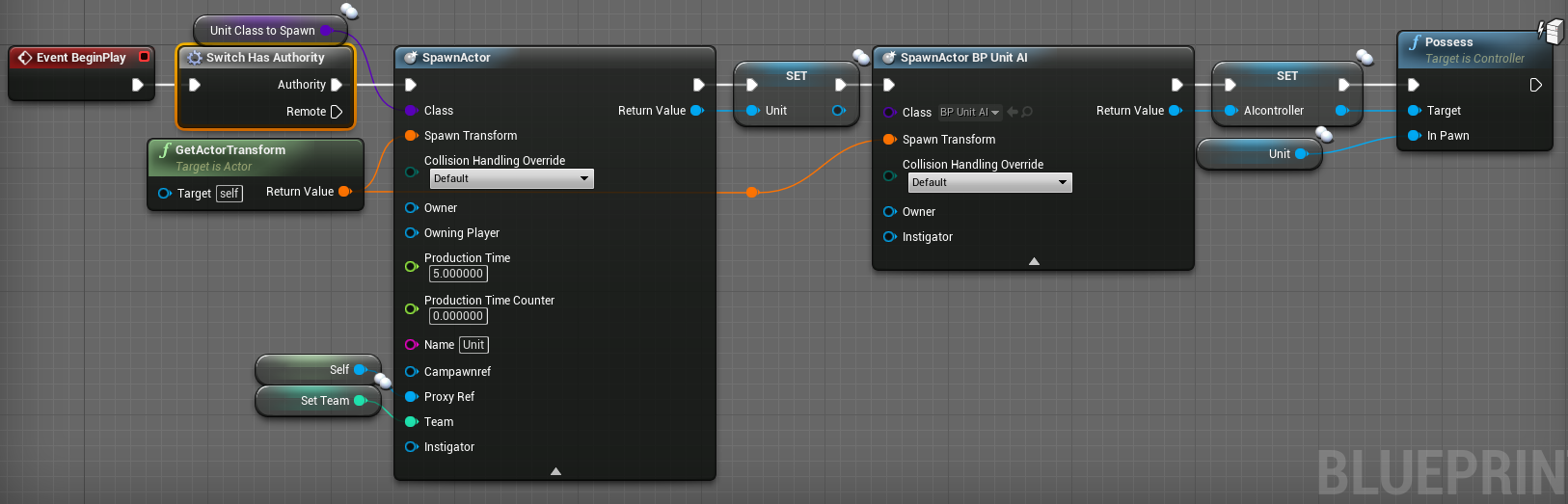
Bullet er også en esential del af kamp og det object som betemmer hvor meget skade en enhed tage når de bliver ramt. Det første der skal sørget for er at kugle ignorer den enhed den er affyre fra da dette er en mulighed, der efter skal kugle destrueres så den ikke bare hænger i spilverdnen og optager hukommelse, dette gøres via et delay så den ikke destruer sig selv med de samme. Det forrige sker dog kun hvis kglen ikke rammer en enhed, for at tjekke om kuglen har ramt en enhed har den en kollisions sfære på sig og med den kan man bruge *Actor Begin Overlap.* Igen skal der tjekke om den kollider med den enhed den er skudt fra, hvis dette argument er falsk, vil den give skade til enten en enhed eller en bygnign alt efter hvad den rammer, hvis den rammer andet vil den også forsvinde.



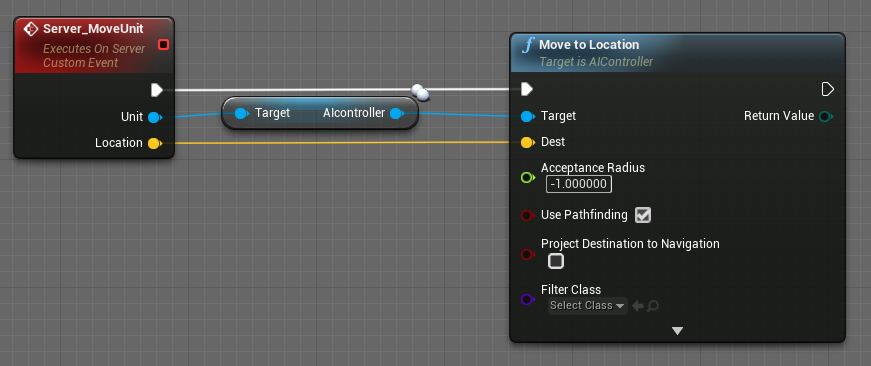
# Serveside Meddelelser

I det at spillet der er online multiplayer spil er det også vigtig at sørger for at information som enhed positioner, antal, hvilke bygninger og hvem der er på hvilket det bliver sendt frem og tilbage mellem spilleren. For at gøre dette bruges nogle *proxys* som kan replikker data fra klienterne og sende det vider.

Denne proxy har to opgaver, at tilføje vores enhed til verden og give den en AI controller som skal sørger for at enhed kan bevæg sig rundt i banen. Dette event begynder når spilleren bygger en enhed, den starter med først et bestem at den kun er serveren der kan have kontrol over den ved at bruge *switch has Authority*. Der efter spawner den en Actor og bruger *Unit Class to spawn* som er er variable af *BP\_Unit*, her får den også tildelt et team og en reference til sig selv, denne reference kaldes når man bruger trykke for at bevæg enheden. For at proxyen sener kan finde den unit den tilhøre sætte en variable unit til at være den nu spawned actor her efter. Nu spawne en *BP\_Unit\_Ai* som er der blueprint som indeholder Ai controlleren og bliver igen sat over i en variable, dette variable bliver sat til at overtage kontrollen af enheden.



Når proxyen er lavet kan det kaldes i *CameraPawnController*, den kan man bruge i en indbygget *Move to location*, som tager den tidligere AiContoller variable og destinationen som bliver sat når man klikker.



Server Spawn unit kaldes når der laves en unit, som før sætte serveren til at være den eneste der må gøre dette. Efter den har gjort dette spawner den en BP\_unit\_proxy som er beskrevet tidligere i afsnittet, efter dette castes det hele til en BP\_unit og tilføjes til et array af ens egen enheder.

# 