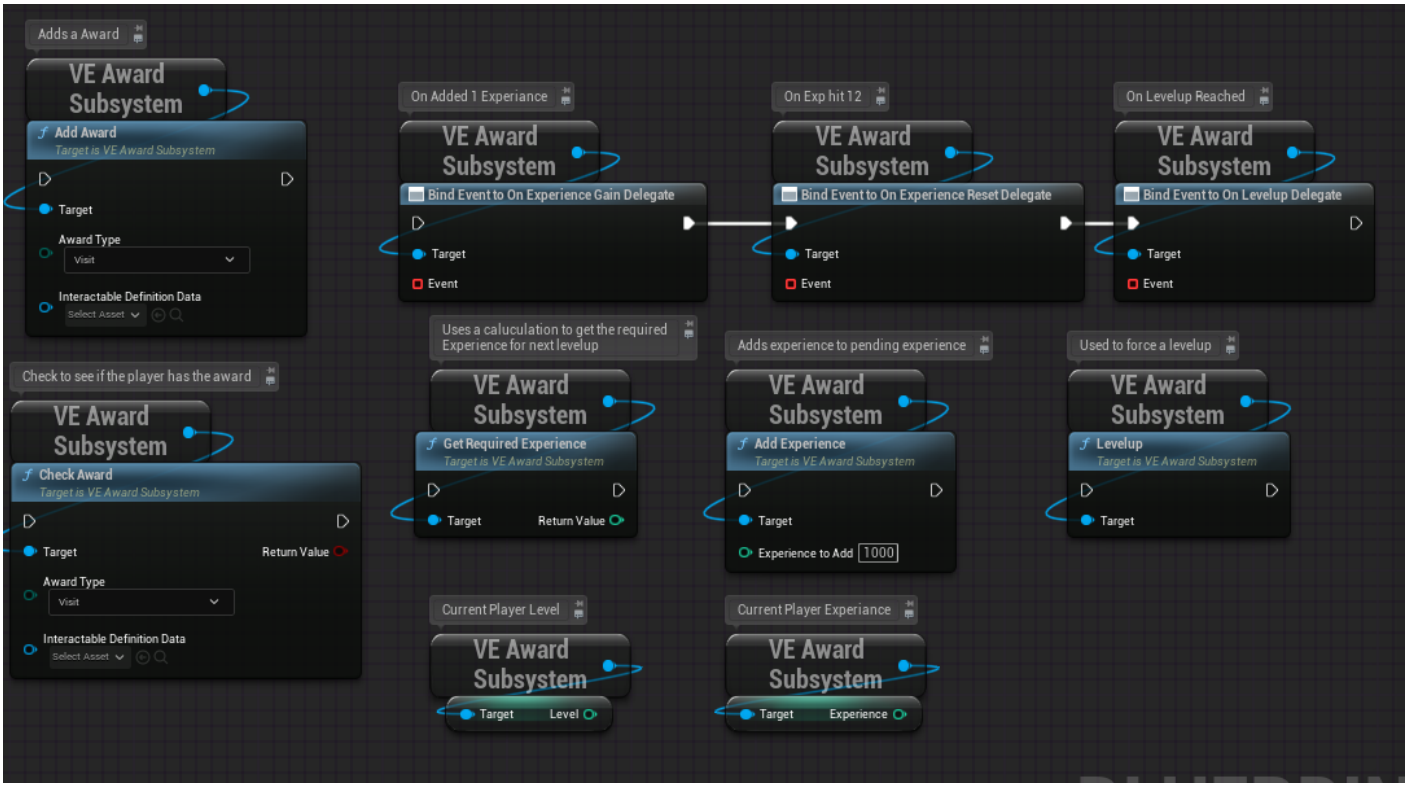
Award Subsytem

The controller for the experience and levelups, also has award storage

- Blueprint functions
- Adding Experience
- Adding/Checking Awards

Blueprint functions



Adding Experience

```cpp
void UVE_AwardSubsystem::AddExperience(int ExperienceToAdd)
{
    PendingExperience += ExperienceToAdd;
}
```

```cpp
bool UVE_AwardSubsystem::ExperienceCheck()
{
    float FLevel = 12 * floor(Level - 1 / 10) + 12;

    if (Experience >= FLevel)
    {
        return true;
    }
    else
    {
        return false;
    }

}
```

```cpp
void UVE_AwardSubsystem::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);
    if (Paused) { return; }
    TimeSinceLastTick += DeltaTime;

    if (TimeSinceLastTick >= TickInterval)
    {
        if (PendingExperience > 0){
            Experience += 1;
            PendingExperience -= 1;
            AddedExperience += 1;
            if (ExperienceCheck()) {
                Experience = 0;
                Levelup();
            }
            if(AddedExperience == 12) {
                TickInterval = 1.1f; //Delay experience gain when leveling up
                AddedExperience = 0;
                OnExperienceResetDelegate.Broadcast();
            }
            else {
                TickInterval = 0.3f; //Reset tick interval so Experience can be gained at a normal rate
                OnExperienceGainDelegate.Broadcast();
            }
        }

        TimeSinceLastTick = 0.0f;
    }

}
```

The add experience function adds experience to the pending experience variable, the tick then moves one point at a time to the experience variable leveling up along the way, every 12 the clock will need to reset

Adding/Checking Awards

```cpp
//Add an award to the completed awards array
void UVE_AwardSubsystem::AddAward(EAwardType AwardType, UInteractableDefinitionData* InteractableDefinitionData)
{
    FAwardData AwardData;
    AwardData.AwardType = AwardType;
    AwardData.InteractableDefinitionData = InteractableDefinitionData;
    CompletedAwards.Add(AwardData);
}

//Check if the award has been completed
bool UVE_AwardSubsystem::CheckAward(EAwardType AwardType, UInteractableDefinitionData* InteractableDefinitionData)
{
    for (int i = 0; i < CompletedAwards.Num(); i++)
    {
        if (CompletedAwards[i].AwardType == AwardType && CompletedAwards[i].InteractableDefinitionData == InteractableDefinitionData)
        {
            return true;
        }
    }
    return false;
}
```