

# Proje Özeti

Bu sistem, **multiplayer uyumlu** olacak şekilde tasarlanmış bir karakter yetenek mekanizmasını içermektedir. Karakter, **sol tık** ile bir **fireball** fırlatarak menzilli saldırı gerçekleştirebilir, **sağ tık** ile ise yere vurarak **alan hasarı** veren bir yetenek kullanabilir. Her karakterin belirli bir **can değeri** bulunmakta olup, bu değer diğer oyuncular tarafından **widget component** aracılığıyla görülebilmektedir. Ayrıca, oyuncuların **can durumları** ve **yetenek bekleme süreleri**, kullanıcı arayüzü üzerinde anlık olarak görüntülenmektedir. Sistem, **GAS** kullanılarak, **çok oyunculu senaryolarda senkronize** çalışacak şekilde geliştirilmiş olup, ağ üzerinde veri tutarlılığını koruyacak optimizasyonlar içermektedir.

## C++ Sınıfları

### UPlayerAttributeSet

**UPlayerAttributeSet**, **UAttributeSet** sınıfından türetilmiş olup GAS ile entegre çalışan sağlık değerlerini içerir.

#### 1. Makro

Makro **ATTRIBUTE\_ACCESSORS** ile **Health** ve **MaxHealth** değişkenleri için **Getter**, **Setter** ve **Initializer** fonksiyonları oluşturulmuştur.

```
#define ATTRIBUTE_ACCESSORS(Classname, PropertyName)
GAMEPLAYATTRIBUTE_PROPERTY_GETTER(Classname, PropertyName)
GAMEPLAYATTRIBUTE_VALUE_GETTER(PropertyName)
GAMEPLAYATTRIBUTE_VALUE_SETTER(PropertyName)
GAMEPLAYATTRIBUTE_VALUE_INITTER(PropertyName)
```

Bu makro, **GAS** için gerekli **GetHealth**, **SetHealth**, **InitHealth** gibi fonksiyonları oluşturur.

#### 2. Delegate

**OnHealthChanged**: **Blueprint Assignable** olarak tanımlanan bir delegate olup, **Health** değiştiğinde event tetiklenmesini sağlar.

#### 3. Değişken

**Health**: Oyuncunun mevcut can değeridir. **OnRep\_Health** fonksiyonu ile replikasyonu yapılır.

**MaxHealth**: Oyuncunun maksimum can değeridir. **OnRep\_MaxHealth** fonksiyonu ile replikasyonu yapılır.

## 4. Fonksiyon

### 4.1 PostGameplayEffectExecute

**PostGameplayEffectExecute**, GAS tarafından **Health** değiştiğinde çağrılır. **Clamp** fonksiyonu ile **Health** değeri **0** ile **MaxHealth** arasında sınırlandırılır.

```
void UPlayerAttributeSet::PostGameplayEffectExecute(const
FGameplayEffectModCallbackData& Data) {
    Super::PostGameplayEffectExecute(Data);
    if (Data.EvaluatedData.Attribute == GetHealthAttribute())
    {
        SetHealth(FMath::Clamp(GetHealth(), 0.0f, GetMaxHealth()));
    }
}
```

### 4.2 Replikasyon

Unreal Engine'de C++ ile multiplayer senkronizasyonu için **GetLifetimeReplicatedProps** kullanılır.

```
void UPlayerAttributeSet::GetLifetimeReplicatedProps(TArray<FLifetimeProperty>&
OutLifetimeProps) const {
    Super::GetLifetimeReplicatedProps(OutLifetimeProps);
    DOREPLIFETIME_CONDITION_NOTIFY(UPlayerAttributeSet, Health, COND_None,
REPNOTIFY_Always);
    DOREPLIFETIME_CONDITION_NOTIFY(UPlayerAttributeSet, MaxHealth,
COND_None, REPNOTIFY_Always);
}
```

Bu fonksiyon, **Health** ve **MaxHealth** değişkenlerinin **replikasyonunu** sağlar.

### 4.3 Replikasyon Bildirimleri

```
void UPlayerAttributeSet::OnRep_Health(const FGameplayAttributeData& OldHealth) {
    GAMEPLAYATTRIBUTE_REPNOTIFY(UPlayerAttributeSet, Health, OldHealth);
    OnHealthChanged.Broadcast();
}
```

**OnRep\_Health**, **Health** değiştiğinde çalışır.

**OnHealthChanged.Broadcast()** çağrılarak, bağlı olan fonksiyonlar çalıştırılır..

```
void UPlayerAttributeSet::OnRep_MaxHealth(const FGameplayAttributeData&
OldMaxHealth) {
    GAMEPLAYATTRIBUTE_REPNOTIFY(UPlayerAttributeSet, MaxHealth,
OldMaxHealth);
}
```

**OnRep\_MaxHealth**, **MaxHealth** değiştiğinde çalışır.

## APlayerCharacter

**APlayerCharacter**, **ACharacter** sınıfından türetilmiş olup GAS ile uyumlu şekilde **AttributeSet** ve **AbilitySystemComponent** bileşenlerini içerir.

### 1. Değişken

```
UPROPERTY(VisibleAnywhere, BlueprintReadWrite, Category = "Initialize | GAS",  
Replicated) TSharedPtr<UPlayerAttributeSet> AttributeSet;
```

**AttributeSet**: Oyuncunun sağlık gibi özelliklerini tutan bileşen.

```
UPROPERTY(VisibleAnywhere, BlueprintReadWrite, Replicated)  
TSharedPtr<UAbilitySystemComponent> AbilitySystemComponent;
```

**AbilitySystemComponent**: GAS yetenek sistemini yöneten bileşen.

### 2. Fonksiyon

#### 2.1 Constructor

```
APlayerCharacter::APlayerCharacter() {  
    AttributeSet = CreateDefaultSubobject<UPlayerAttributeSet>(TEXT("AttributeSet"));  
    AbilitySystemComponent =  
    CreateDefaultSubobject<UAbilitySystemComponent>(TEXT("AbilitySystemComponent"));  
}
```

**Constructor**, **AttributeSet** ve **AbilitySystemComponent** bileşenlerini oluşturur.

#### 2.2 Replikasyon

```
void APlayerCharacter::GetLifetimeReplicatedProps(TArray<FLifetimeProperty>&  
OutLifetimeProps) const {  
    Super::GetLifetimeReplicatedProps(OutLifetimeProps);  
    DOREPLIFETIME(APlayerCharacter, AttributeSet);  
}
```

Bu fonksiyon, **AttributeSet** bileşeninin **replikasyonunu** sağlar.

#### 2.3 Getter Fonksiyonları

```
FORCEINLINE UAbilitySystemComponent* GetAbilitySystemComponent() const {  
    return AbilitySystemComponent;  
}
```

```
FORCEINLINE UPlayerAttributeSet* GetAttributeSet() const {  
    return AttributeSet;  
}
```

**GetAbilitySystemComponent:** Oyuncunun **AbilitySystemComponent** bileşenini döndüren bir fonksiyondur.

**GetAttributeSet:** Oyuncunun **AttributeSet** bileşenini döndüren bir fonksiyondur.

## UW\_Ability

**UW\_Ability**, oyuncunun seçili yeteneği ile ilgili **Cooldown (Bekleme Süresi)** takibini sağlayan bir kullanıcı arayüzü widget'ıdır.

### 1. Değişken

**CooldownTag:** Yeteneğin bekleme süresini takip eden **FGameplayTag** değişkenidir.

### 2. Fonksiyon

#### 2.1 NativeConstruct

```
void UW_Ability::NativeConstruct() {  
    Super::NativeConstruct();  
    BindOnCooldownTagChanged();  
}
```

**NativeConstruct**, widget oluşturulduğunda çağrılan bir fonksiyondur.

**BindOnCooldownTagChanged** fonksiyonunu çağırarak cooldown değişikliklerini dinlemeyi sağlar.

#### 2.2 BindOnCooldownTagChanged

```
void UW_Ability::BindOnCooldownTagChanged() {  
    APlayerCharacter* PlayerCharacter =  
    Cast<APlayerCharacter>(GetOwningPlayerPawn());  
    if (PlayerCharacter)  
    {
```

```
        PlayerCharacter->GetAbilitySystemComponent()->RegisterGameplayTagEvent(CooldownTag,  
        EGameplayTagEventType::NewOrRemoved).AddUObject(this,  
        &UW_Ability::OnCooldownTagChanged);  
    }  
}
```

**BindOnCooldownTagChanged**, **CooldownTag** değişikliklerini dinlemek için **GameplayTagEvent** kaydı yapar.

Oyuncunun **AbilitySystemComponent** bileşeni üzerinden **CooldownTag** değiştiğinde **OnCooldownTagChanged** fonksiyonu çağrılır.

## 2.3 OnCooldownTagChanged

```
void UW_Ability::OnCooldownTagChanged(const FGameplayTag Tag, int32 NewCount) {  
    if (NewCount > 0)  
    {  
        OnCooldownStarted();  
    }  
}
```

**OnCooldownTagChanged**, **CooldownTag** değiştiğinde çağrılır.

**NewCount > 0** olduğunda yani tag eklendiğinde **OnCooldownStarted** event'i tetiklenir.

## 2.4 OnCooldownStarted

UFUNCTION(BlueprintImplementableEvent) void OnCooldownStarted();

**OnCooldownStarted**, **BlueprintImplementableEvent** olarak tanımlanmıştır ve Blueprint üzerinden override edilir.

# Blueprint Sınıfları

## BP\_GameMode

BP\_GameMode **GameModeBase**'den türetilmiş bir sınıftır. Bu sınıfta seçili GameMode'a özgü çalışacak temel sınıfların ayarlanır.

▼ Classes	
Game Session Class	GameSession ▼ ↶ 📁 ✕
Game State Class	GameStateBase ▼ ↶ 📁 ⊕
Player Controller Class	BP_PlayerController ▼ ↶ 📁 ⊕
Player State Class	PlayerState ▼ ↶ 📁 ⊕
HUD Class	BP_HUD ▼ ↶ 📁 ⊕ ✕
Default Pawn Class	BP_PlayerCharacter ▼ ↶ 📁 ⊕ ✕
Spectator Class	SpectatorPawn ▼ ↶ 📁 ⊕
Replay Spectator Player Controller Class	PlayerController ▼ ↶ 📁 ⊕
Server Stat Replicator Class	ServerStatReplicator ▼ ↶ 📁

**BP\_PlayerController:** Oyuncu denetleyicisi, her oyuncunun oyundaki etkileşimlerini yöneten sınıftır.

**BP\_HUD:** Oyun içi kullanıcı arayüzü (HUD), oyuncuya oyunun durumu hakkında bilgi verir.

**BP\_PlayerCharacter:** Bu sınıf, oyuncunun oyun içindeki karakterini temsil eder.

## BP\_HUD

BP\_HUD **HUD** sınıfından türetilmiştir.

### 1. Event

#### 1. Event BeginPlay

**WBP\_HUD** türünde bir widget Blueprint'i oluşturulup, bu widget, HUD'a sahip karakterin **viewport**'una dinamik olarak eklenir.

## BP\_PlayerController

PlayerController sınıfından türetilmiştir.

## BP\_PlayerCharacter

### 1. Component

- 1.1 CapsuleComponent
- 1.2 SpringArmComponent
- 1.3 CameraComponent
- 1.4 SkeletalMeshComponent
- 1.5 WidgetComponent
- 1.6 ArrowComponent
- 1.7 CharacterMovementComponent
- 1.8 AbilitySystemComponent

### 2. Değişken

**Abilities:** **GameplayAbility** türünde değişkenler içeren bir dizidir. Karakterin sahip olacağı abilityler için kullanılır.

**WBPHealthWidgetComponent:** **WBP\_HealthWidgetComponent** türünde bir widget objesidir.

## 3. Event

### 3.1.1 EventGraph / Event Possessed

Server Only fonksiyondur. BeginPlay'den önce çalışır. SV\_AddAbilities event'i çağırılır.

### 3.1.2 EventGraph / Event BeginPlay

Karaktere MappingContext eklenir. SV\_UpdateHealth event'i çağırılır. SV\_Bind event'i çağırılır.

### 3.2.1 InputActions / EnhancedInputAction IA\_Look

Triggered anında **Look** fonksiyonu çağırılır.

### 3.2.2 InputActions / EnhancedInputAction IA\_Move

Triggered anında **Move** fonksiyonu çağırılır.

### 3.2.3 InputActions / EnhancedInputAction IA\_Jump

Started anında Character sınıfından **Jump** fonksiyonu çağırılır. Cancelled ve Completed anlarında Character sınıfından **StopJumping** fonksiyonu çağırılır.

### 3.2.4 InputActions / EnhancedInputAction IA\_Primary

Started anında **GA\_Primary** parametresi ile **SV\_ActivateAbility** çağırılır.

### 3.2.5 InputActions / EnhancedInputAction IA\_Secondary

Started anında **GA\_Secondary** parametresi ile **SV\_ActivateAbility** çağırılır.

### 3.3.1 GameplayAbilitySystem / Bind

AttributeSet içerisindeki **OnHealthChanged** delegate'ine **SV\_UpdateHealth** event'ini bağlar.

### 3.3.2 GameplayAbilitySystem / SV\_AddAbilities

ServerOnly RPC. **Abilities** dizisinde bulunan her bir yeteneği loop ile döndürerek karaktere ekler.

### 3.3.3 GameplayAbilitySystem / SV\_ActivateAbility

ServerOnly RPC. Parametre olarak gönderilen yeteneği **TryActivateabilityByClass** kullanarak aktifleştirir.

### 3.3.4 GameplayAbilitySystem / SV\_UpdateHealth

ServerOnly RPC. **MC\_UpdateHealth** event'ini çağırır.

### 3.3.5 GameplayAbilitySystem / MC\_UpdateHealth

Multicast RPC. WBP\_HealthWidgetComponent geçerli mi diye kontrol edilir. Geçerliyse Karakterin AbilitySet'inden Health ve MaxHealth değerleri çekilip WBP\_HealthWidgetComponent içinden SetProgressBarPercent event'i çağırılarak

güncellenir. Geçerli değilse Widget Component üzerinden GetUserWidgetObject kullanılarak bileşendeki Widget objesi çekilir ve WBP\_HealthWidgetComponent sınıfına cast yapılır. Sonrasında SetProgressBarPPercent kullanılır.

## GA\_Base

### 1. Değişken

**Duration:** Float türünde abilitynin uzunluğu için kullanılan değişkendir.

**CooldownTag:** GameplayTag türündeki değişkendir. Yeteneğe ait cooldown tag'ini çekebilmek için kullanılır.

### 2. Fonksiyon

#### 2.1 CanActivateAbility

Yeteneği tetikleyen karakterde **CooldownTag** olup olmadığı kontrol eder. Tag'e sahipse false dönerek yeteneğin çalışmasını engeller.

## GA\_Primary

**GA\_Base** sınıfından türetilmiştir. Karakterin birincil yeteneğidir.

### 1. Değişken

**Owner:** **BP\_PlayerCharacter** türündeki değişkendir. Yetenek başlatıldığında OwningActor'u bu değişkene atanır.

**SpawnedActor:** Yetenek içerisinde oluşturulan **BP\_PrimaryAbilityActor** değişkeni bu değişkende tutulur.

**Duration** ve **CooldownTag** **GA\_Base**'den gelmektedir. Uygun değerler atanır.

### 2. Event

#### 2.1 ActivateAbility

**PlayMontageAndWait** kullanarak montajı oynatmaya başlar. **WaitGameplayEvent** ile Montaj içerisindeki yeteneği tetikleyen **notify**'i dinler. **EventReceived** anında **HasAuthority** ile sadece serverda çalışacak şekilde devam eder. **SpawnActorforGameplayTask** ile **BP\_PrimaryAbilityActor** oluşturulur. **SpawnedActor**'e atanır. **CommitAbilityCooldown** çağırılır. **SpawnedActor** içindeki **bOverlapped** değişkeni true döndüyse **SpawnedActorOverlapped** çağırılır. False döndüyse **SpawnedActor** içindeki **SphereComponent OncomponentBeginOverlap**'e **SpawnedActorOverlapped** bağlanır. **SpawnedActor**'deki **OnManuallyDestroyed**'a **EndAbility** bağlanır. Montaj bitmeden yarıda kesilirse **OnInterrupted** ve **OnCancelled** anlarında **CancelAbility**'i çağırır.



## 2.2 OnEndAbility

SpawnedActor geçerliyse yokedilir.

## 2.3 SpawnedActorOverlapped

Overlap olan aktör **BP\_PlayerCharacter** türündeysse **DamageEffect** efekti uygulanır. Timer ile **EndAbility** çağırılır. Başka bir türdeyse direkt **EndAbility** çağırılır.

# GA\_Secondary

GA\_Base sınıfından türetilmiştir. Karakterin ikincil yeteneğidir.

## 1. Değişken

**Owner:** **BP\_PlayerCharacter** türündeki değişkendir. Yetenek başlatıldığında OwningActor'u bu değişkene atanır.

**SpawnedActor:** Yetenek içerisinde oluşturulan **BP\_SecondaryAbilityActor** değişkeni bu değişkende tutulur.

**blsAnimFinished:** Boolean türündeki bir değişkendir default olarak false döner. Montaj bittikten sonra true olarak değiştirilir.

**blsLoopCompleted:** Boolean türündeki bir değişkendir default olarak false döner.

OverlappedActors döngüsü bittikten sonra true olarak değiştirilir.

**ObjectTypes:** EObjectTypeQuery türünde değişkenler içeren bir dizidir. Default olarak tek elemanı "Pawn"dır.

**Duration** ve **CooldownTag** **GA\_Base**'den gelmektedir. Uygun değerler atanır.

## 2. Event

### 2.1 ActivateAbility

**PlayMontageAndWait** kullanarak montajı oynatmaya başlar. **WaitGameplayEvent** ile Montaj içerisindeki yeteneği tetikleyen **notify**'i dinler. **EventReceived** anında **HasAuthority** ile sadece serverda çalışacak şekilde devam eder. **SpawnActorforGameplayTask** ile **BP\_SecondaryAbilityActor** oluşturulur. **SpawnedActor**'e atanır. **CommitAbilityCooldown** çağırılır. **SphereOverlapActors** kullanılarak karakterin bulunduğu alan taranır. Alanın içerisindeki diğer karakterler dizisi döngüyle döndürülerek **DamageEffect** uygulanır. Döngü bittiğinde **blsLoopCompleted** true olarak değiştirilir ve **CanEndAbility** çağırılır. Montaj bitmeden yarıda kesilirse **OnInterrupted** ve **OnCancelled** anlarında **CancelAbility**'i çağırır. Montaj bittiğinde **OnCompleted** anında **blsAnimFinished** true olarak değiştirilir ve **CanEndAbility** çağırılır.

### 2.2 OnEndAbility

SpawnedActor geçerliyse yokedilir.

### 2.3 SpawnedActorOverlapped

**blsAnimFinished** ve **blsLoopCompleted** true ise **EndAbility** çağırılır.

# BP\_PrimaryAbilityActor

## 1. Component

- 1.1 DefaultSceneRoot
- 1.2 SphereComponent
- 1.3 NiagaraComponent
- 1.4 ProjectileMovement

## 2. Event'ler

### 2.1 Event BeginPlay

Optimizasyon amacıyla 5 saniyelik bir timer başlatır. 5 saniye boyunca aktör overlap olmazsa DestroyFunction çağırılır.

### 2.2 Event Destroyed

Aktör destroyed anında bOverlapped true döndüyse DestroyFunction çağırılır. False döndüyse OnManuallyDestroyed delegate i çağırılır.

### 2.3 SV\_StopMovement

Server Only RPC. **ProjectileMovement** için **StopMovementImmediately** çağırılır. Hareket durdurulur.

### 2.4 OnComponentBeginOverlap

OtherActor değişkene atanır. **bOverlapped** true olarak güncellenir. **SV\_StopMovement** çağırılır. **SetNiagaraSystemAsset** ve **ResetSystem** çağırılarak Niagara efekti değiştirilir patlama efekti gösterilir.

## 3. Fonksiyon

### 3.1 DestroyFunction

**DestroyActor** çağırılır.

# BP\_SecondaryAbilityActor

## 1. Component

- 1.1. DefaultSceneRoot
- 1.2. NiagaraComponent

## GE\_DamageEffect

**GameplayEffect** sınıfından türetilmiştir. Instant olarak efekt uygular. Modifier kullanarak karakterin **Health** attribute'unu 10 azaltır.

▼ Duration		
Duration Policy	Instant ▼	
▼ Stacking		
Stacking Type	None ▼	
▼ Status		
Editor Status Text	All Ok	
▼ Gameplay Effect		
Components	0 Array element	
▼ Modifiers	1 Array element	
▼ Index [ 0 ]	(AttributeName="Health",Attribute="/Script/CaseStudy.PlayerAttributeSet.Health",AttributeI ▼	
Attribute	PlayerAttributeSet.Health ▼	
Modifier Op	Add (Base) ▼	
▼ Modifier Magnitude		
Magnitude Calculation Type	Scalable Float ▼	
Scalable Float Magnitude	-10.0	Use CurveTable... ▼
▶ Source Tags		
▶ Target Tags		

## GE\_PrimaryCooldown

**GameplayEffect** sınıfından türetilmiştir. 5 saniye süreyle efekt uygular. Efekt süresince karaktere birincil yetenek için cooldown kontrolü sağlanan tagi ekler.

## GE\_SecondaryCooldown

**GameplayEffect** sınıfından türetilmiştir. 10 saniye süreyle efekt uygular. Efekt süresince karaktere ikincil yetenek için cooldown kontrolü sağlanan tagi ekler.

## WBP\_HUD

### 1. Hierarchy

- Overlay
  - SizeBox
    - WBP\_HealthBar
  - SizeBox
    - WBP\_Abilities

## WBP\_AttributeBarBase

UserWidget sınıfından türetilmiş bir sınıftır. HUD'a eklenecek Attribute bilgisini ProgressBar olarak gösterir.

## 1. Hierarchy

- ProgressBar

## 2. Değişken

**BarFillColor:** LinearColor değişken. Bar dolduğundaki rengi belirler.

**BarEmptyColor:** LinearColor değişken. Bar boşkenki rengi belirler.

**Attribute:** Gösterilecek Attribute.

**MaxAttribute:** Gösterilecek Max Attribute.

**CurrentHealth:** Float şuanki can miktarı.

**TargetHealth:** Float güncellenecek can miktarı.

**Alpha:** Lerp için kullanılan alpha değeri.

**ProgressBar:** Widget'ta bulunan ProgressBar.

## 3. Event

### 3.1 Event Construct

Seçilen değişkenlere göre **ProgressBar**'ın renklerini düzenler. Widget'ın sahibi olan karakterin **AttributeSet**'indeki OnHealthChanged delegate'ine

**UpdateProgressBar** fonksiyonunu bağlar. ProgressBar yüzdesini

AttributeSet'ten gelen değere göre günceller.

### 3.2 UpdateProgressBar

**Delegate** çağırıldığı andaki **Health** değerini ve **MaxHealth** değerini ilgili değişkenlerde günceller. Döngü ile 0.05 saniyede bir çağırılacak timer başlatır. Timer

**UpdateTimerFunction**'u çalıştırır.

## 4. Fonksiyon

### 4.1 UpdateTimerFunction

Lerp ile **CurrentHealth** ile **TargetHealth** arasında yumuşak bir geçiş olması için

ProgressBar yüzdesini günceller. Yüzde TargetHealth'a ulaştığında timer sıfırlanır ve Alpha 0 olarak güncellenir.

## WBP\_HealthBar

**WBP\_AttributeBarBase** sınıfından türetilmiştir. Widget değişkenleriyle kırmızı renk yapılmıştır. Event'ler AttributeBarBase'den inherit edilmiştir.

# WBP\_Ability

**W\_Ability** sınıfından türetilmiştir. Yetenek cooldown süresini görselleştirmek için kullanılır.

## 1. Hierarchy

- Overlay
  - SizeBox
    - ProgressBar

## 2. Değişken

**Duration:** Float değişken cooldown süre miktarını belirlemek için kullanılır.

**Alpha:** Lerp için kullanılan alpha değeri.

**ElapsedTime:** Timer çalıştığı süre boyunca geçen zaman.

**ProgressBar:** Widget'ta bulunan ProgressBar.

## 3. Event

### 3.1 Event OnCooldownStarted

**BlueprintImplementableEvent.** **ProgressBar** yüzdesini %100 yapıp **timer** başlatır. Döngü ile 0.05 saniyede bir çalışan timer **CooldownTimerFunction** fonksiyonunu çağırır.

### 3.2 SetDuration

Parametreyi **Duration** değişkenine atar.

### 3.3 SetCooldownTag

Parametreyi **CooldownTag** değişkenine atar.

## 4. Fonksiyon

### 4.1 CooldownTimerFunction

**Lerp** ile anlık kalan cooldown süresi ile toplam cooldown süresi arasında yumuşak bir geçiş olması için **ProgressBar** yüzdesini günceller. Yüzde sıfıra ulaştığında timer sıfırlanır ve **Alpha** ve **ElapsedTime** 0 olarak güncellenir.

# WBP\_Abilities

## 1. Hierarchy

- HorizontalBox

## 2. Değişken

**HorizontalBox:** Widget'ta bulunan HorizontalBox.

**OwnerASC:** Widget'a sahip olan karakterin AbilitySystemComponent'i.

## 3. Event

### 3.1 Event Construct

Karakterin sahip olduğu bütün yetenekleri çeker. Bir döngüyle hepsini tarar. Her biri için WBP\_Ability türünde widget oluşturur. Bu widget'ı HorizontalBox'a ekler. Eğer son yetenek değilse araya WBP\_Spacer oluşturup ekler. Yetenekten Duration ve CooldownTag bilgisini çekerek WBP\_Ability de SetDuration ve SetCooldownTag çağırır.

## WBP\_Spacer

Sadece Spacer içeren bir sınıftır. WBP\_Abilities içinde yetenek eklerken kullanılır.