

# LAB – 10

## ENHANCING THE PLAYER EXPERIENCE

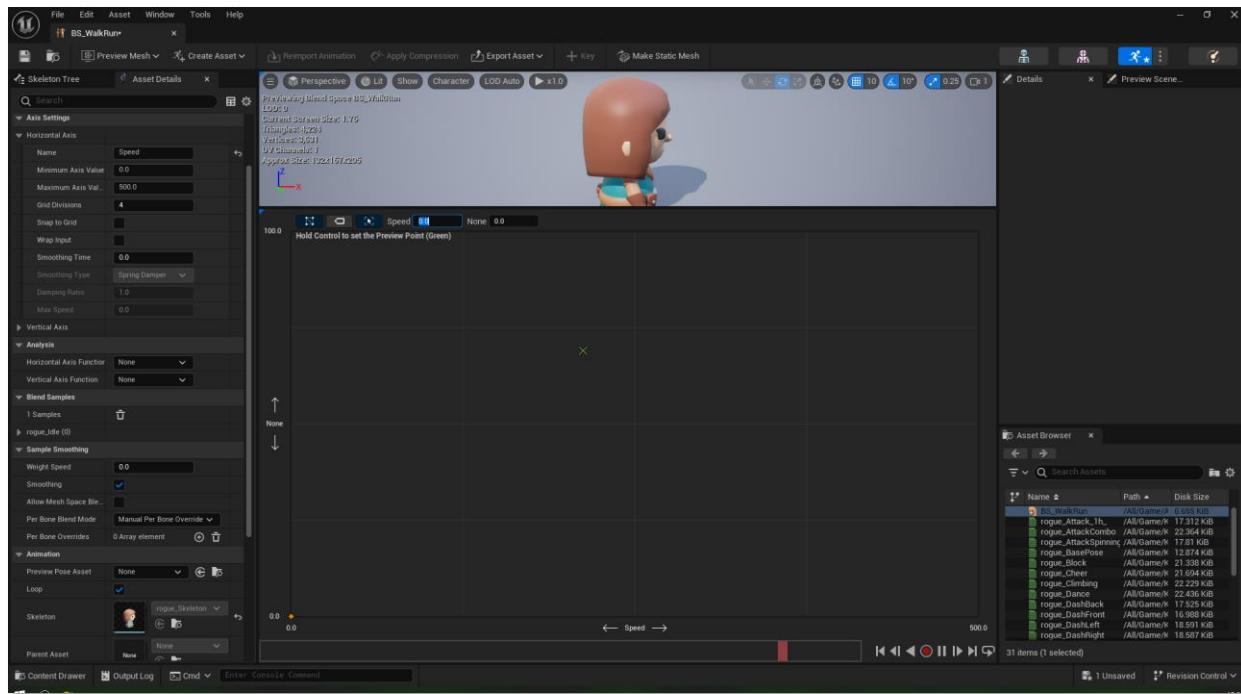


Fig 1.1: Screenshot showing coordinate set to 0 for respective attributes

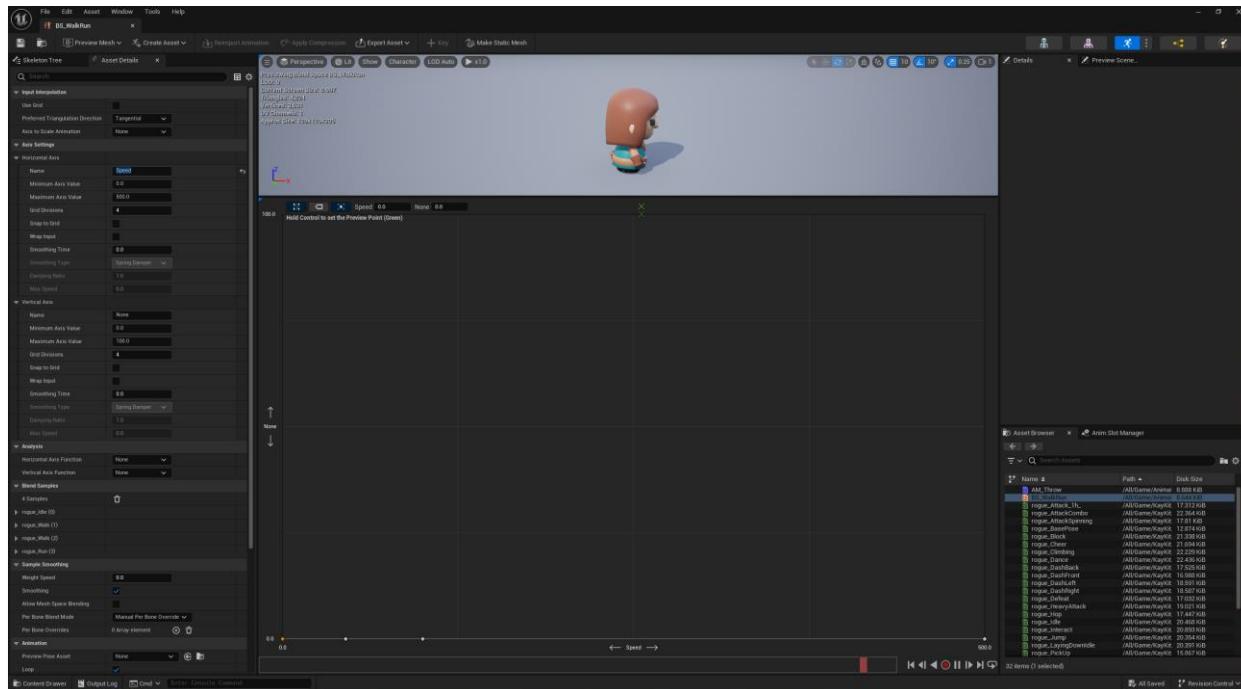


Fig 1.2: Screenshot showing coordinate set to 0 for respective attributes

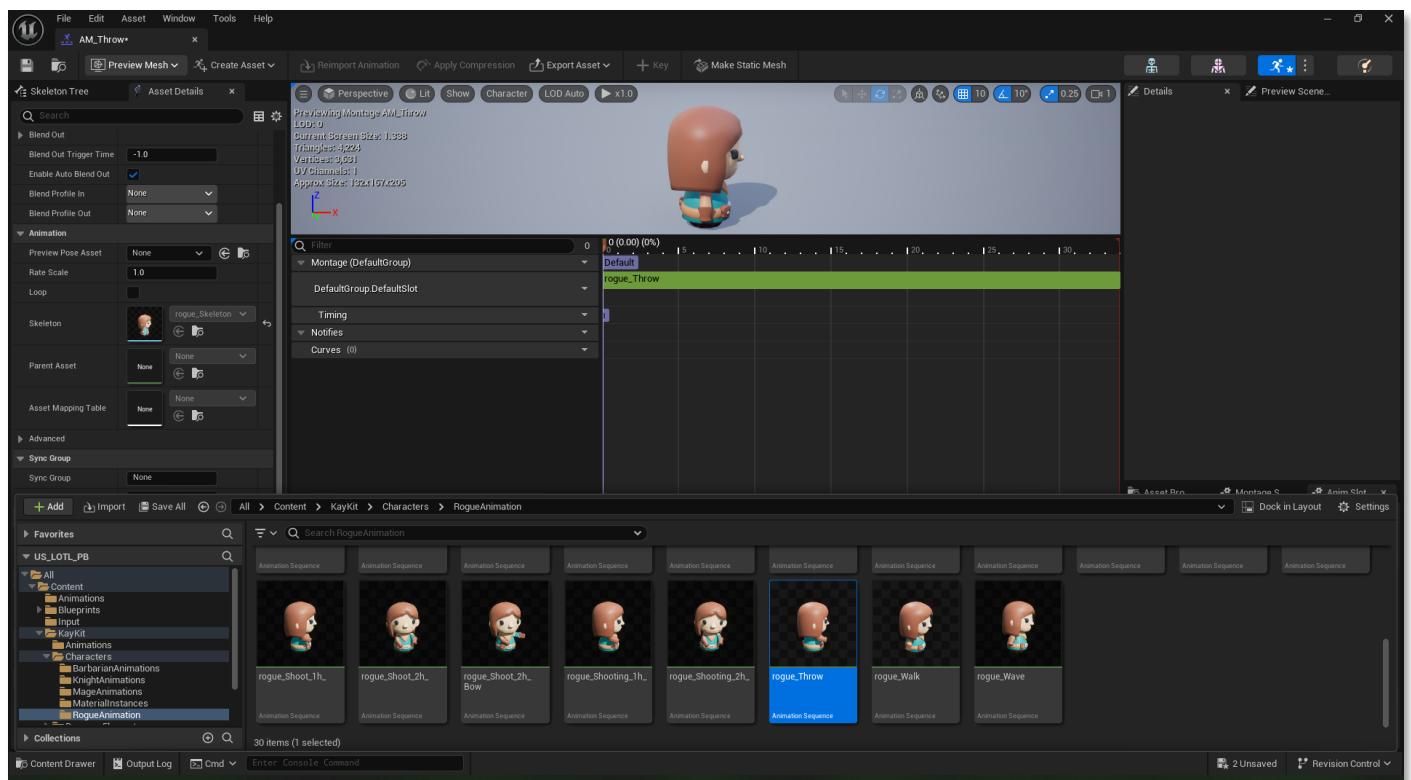


Fig 1.3: Screenshot showing Animation Montage for rogue\_throw

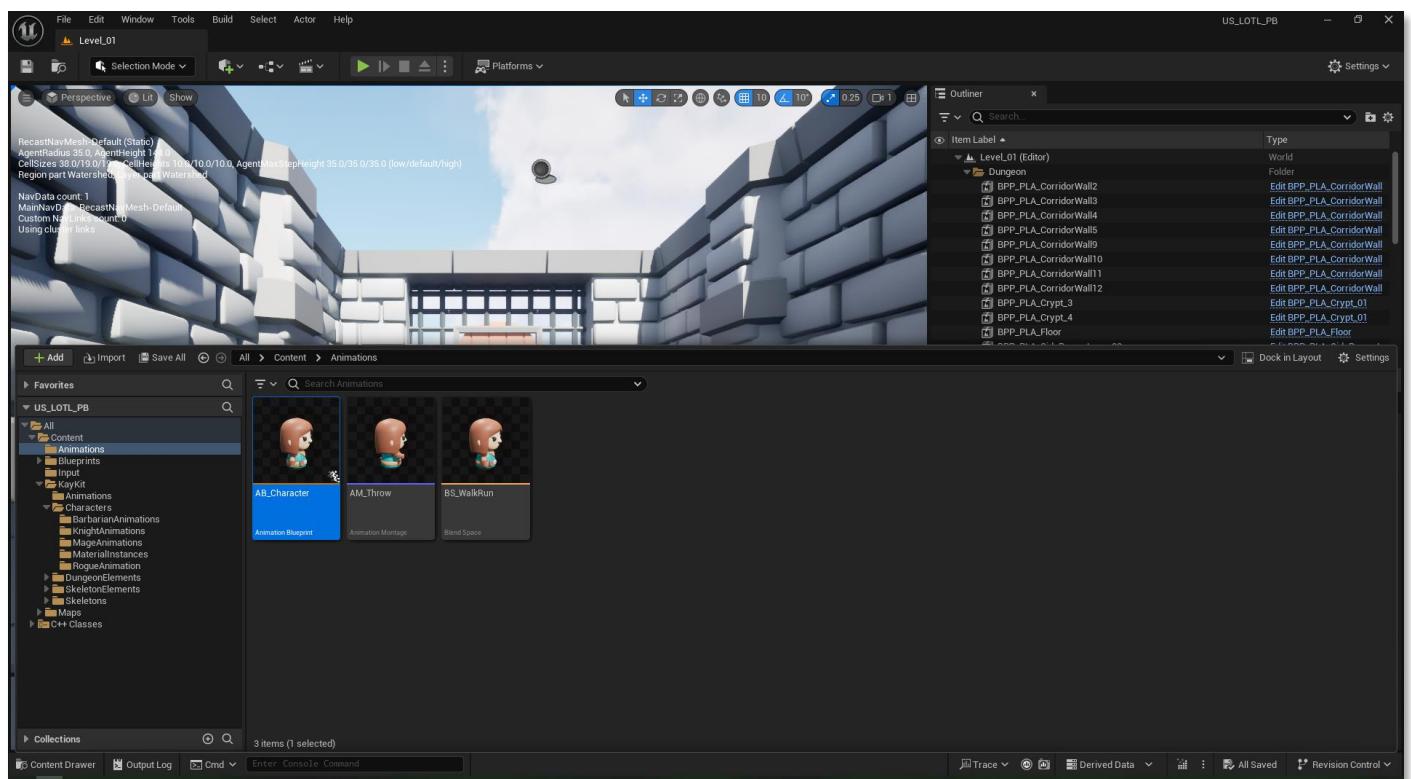


Fig 1.4: Screenshot showing Animation Blueprint for rogueSkeleton

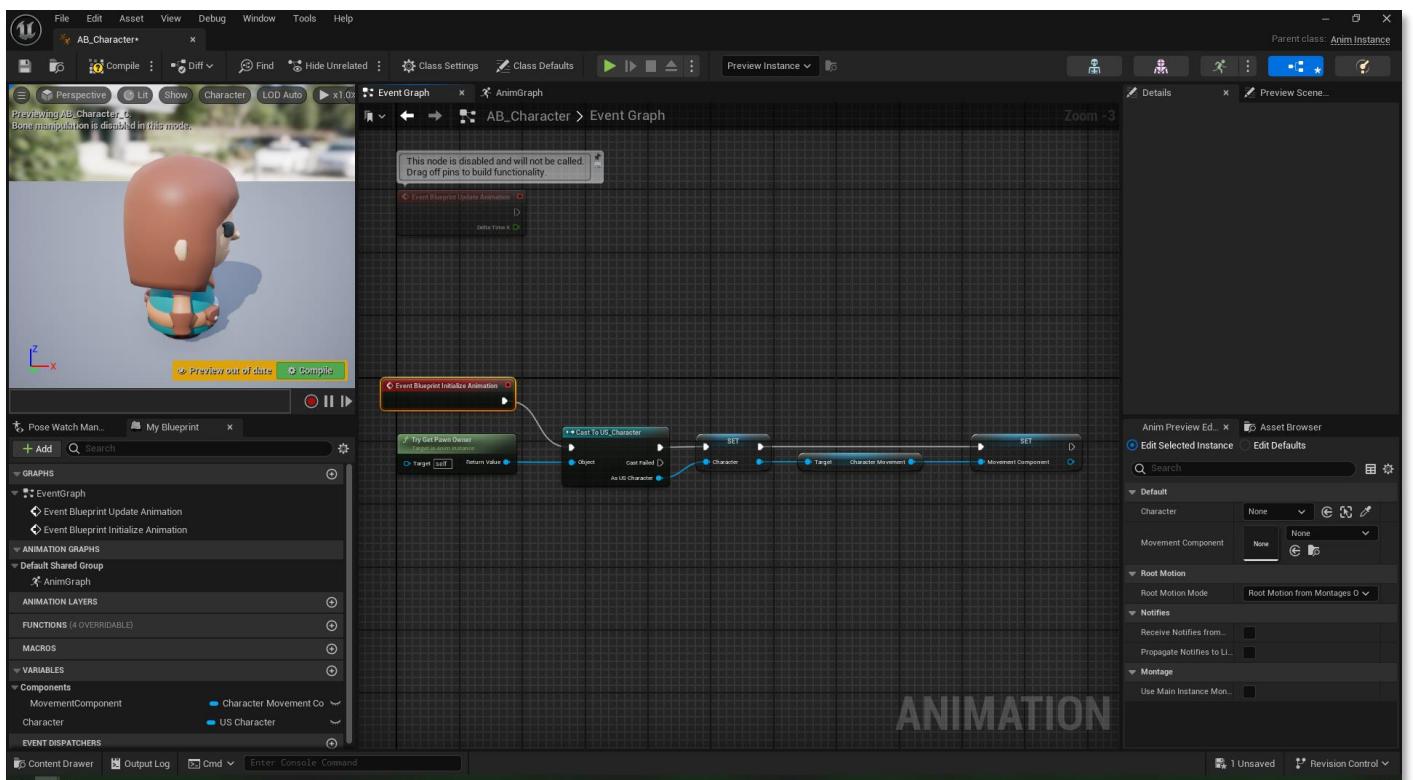


Fig 1.5: Screenshot showing Event Graph for AB\_Character (Initialize Animation)

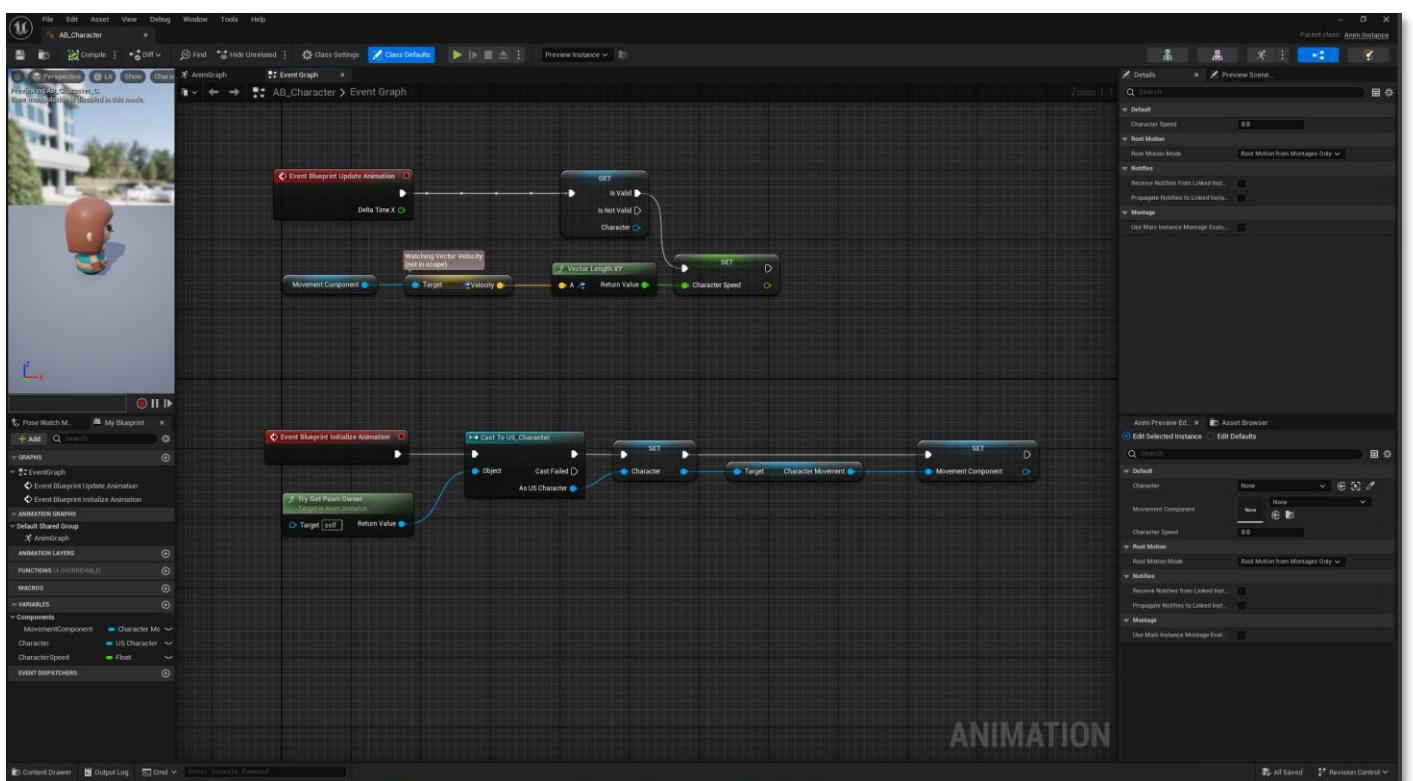


Fig 1.6: Screenshot showing Event Graph for AB\_Character (Update Animation)

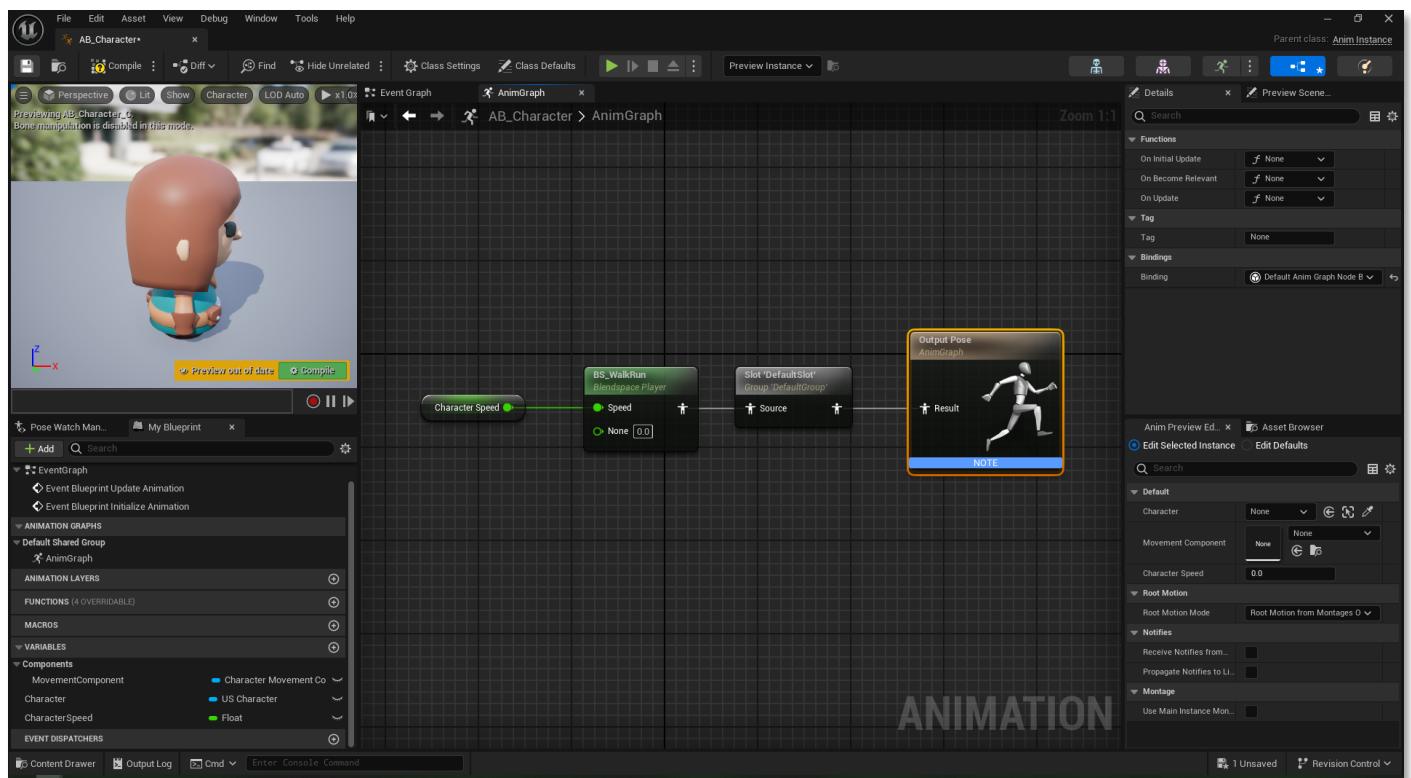


Fig 1.7: Screenshot showing AnimGraph for AB\_Character

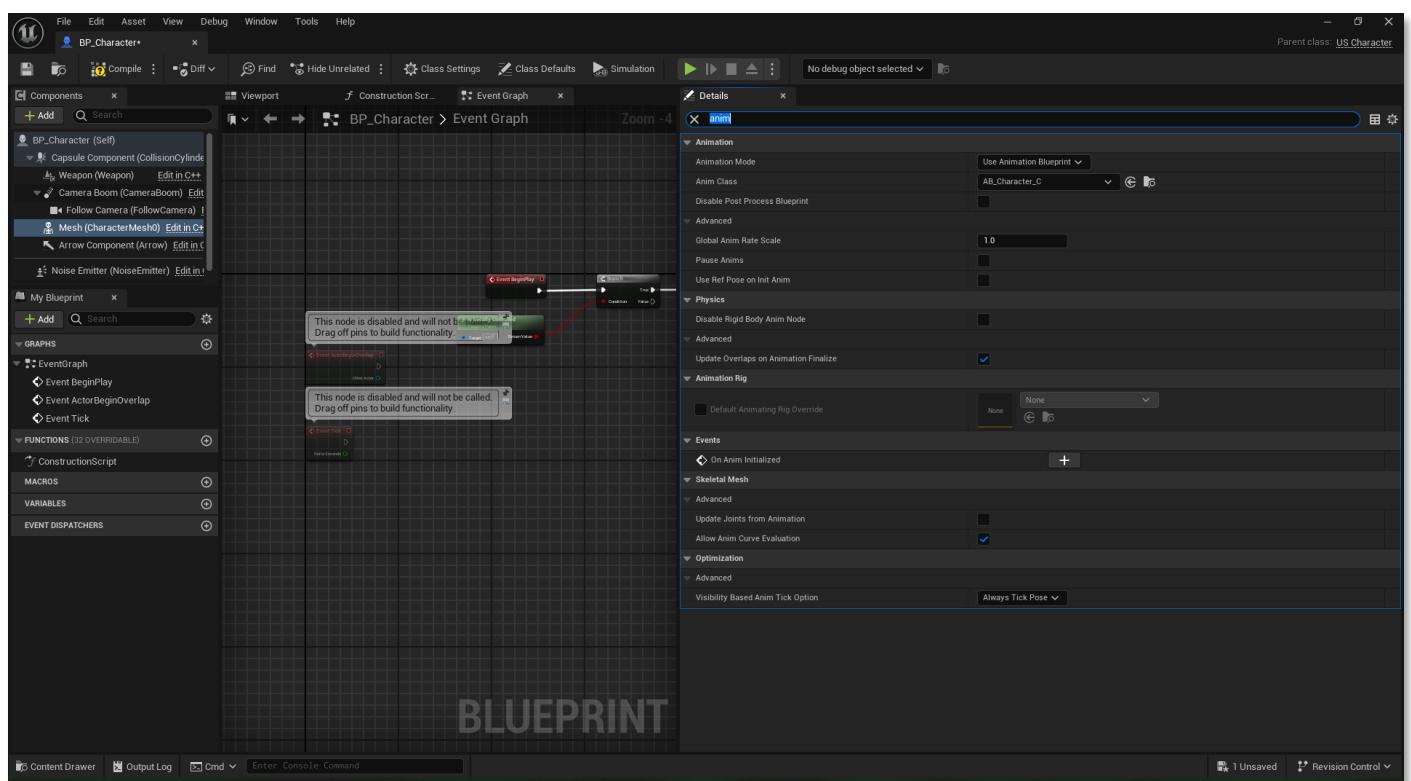


Fig 1.8: Screenshot showing setting of Anim Class for BP\_Character

## CHALLENGE

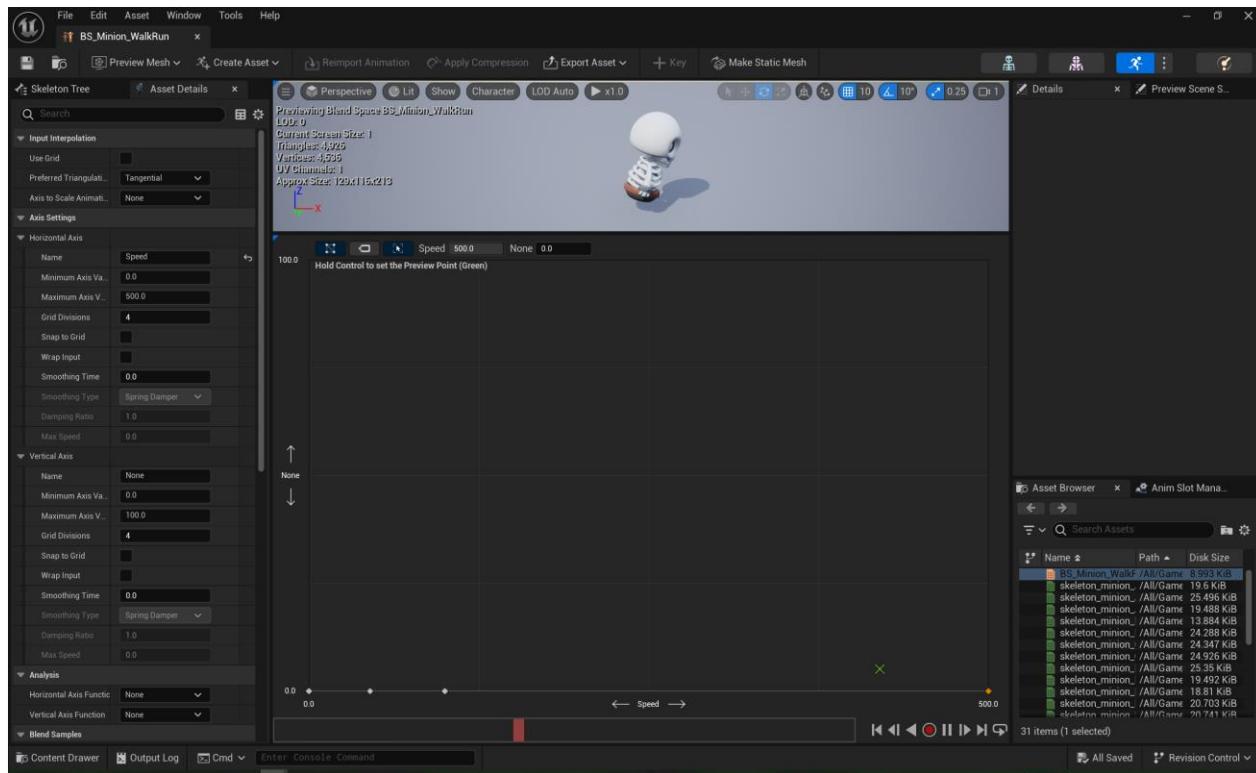


Fig 1.9: Screenshot showing Minion Animation Blending

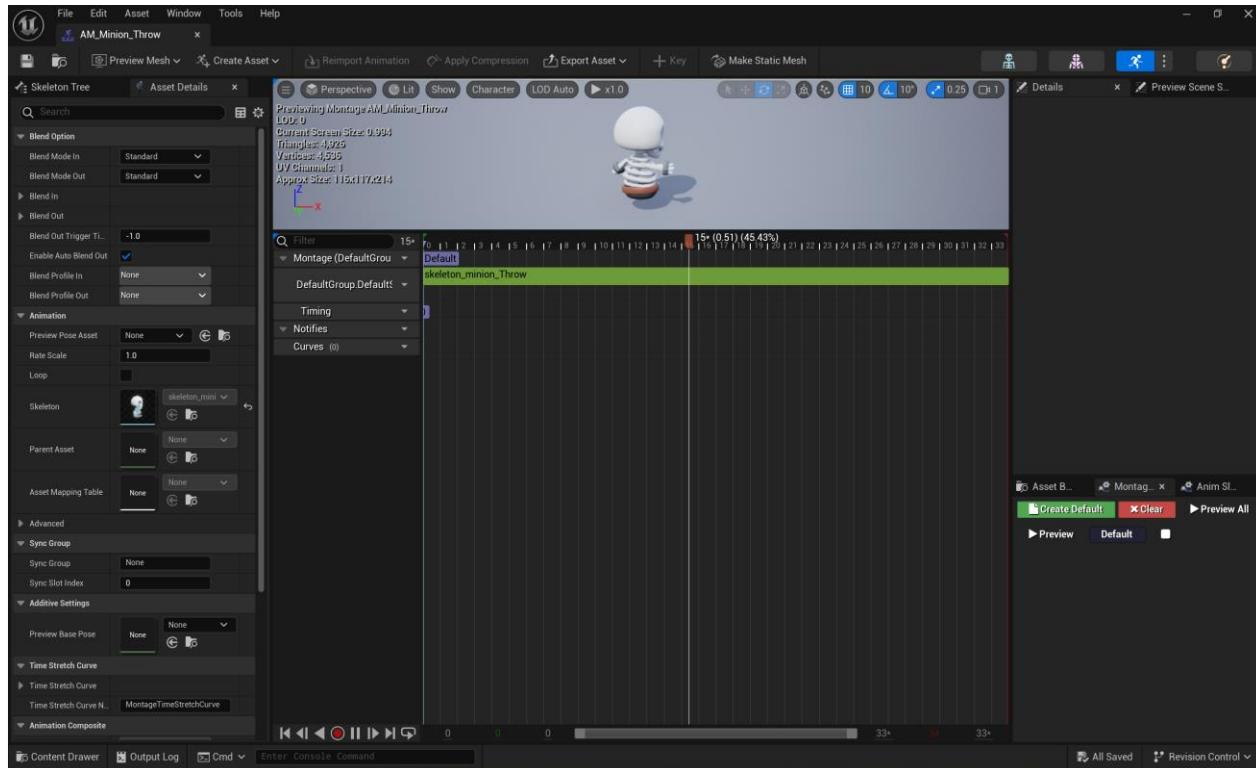


Fig 1.10: Screenshot showing Minion Animation Montage for Throw

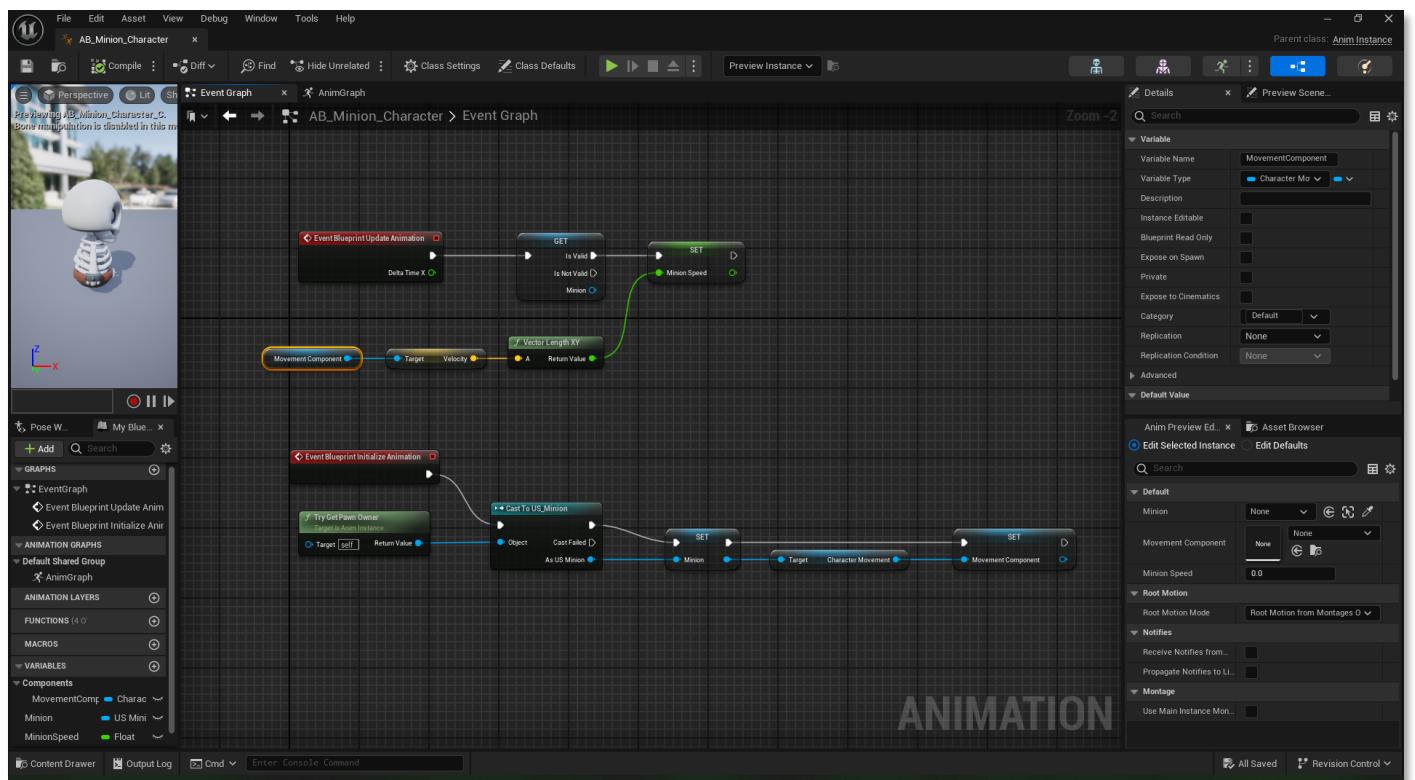


Fig 1.11: Screenshot showing Minion Animation Blueprint in Event Graph

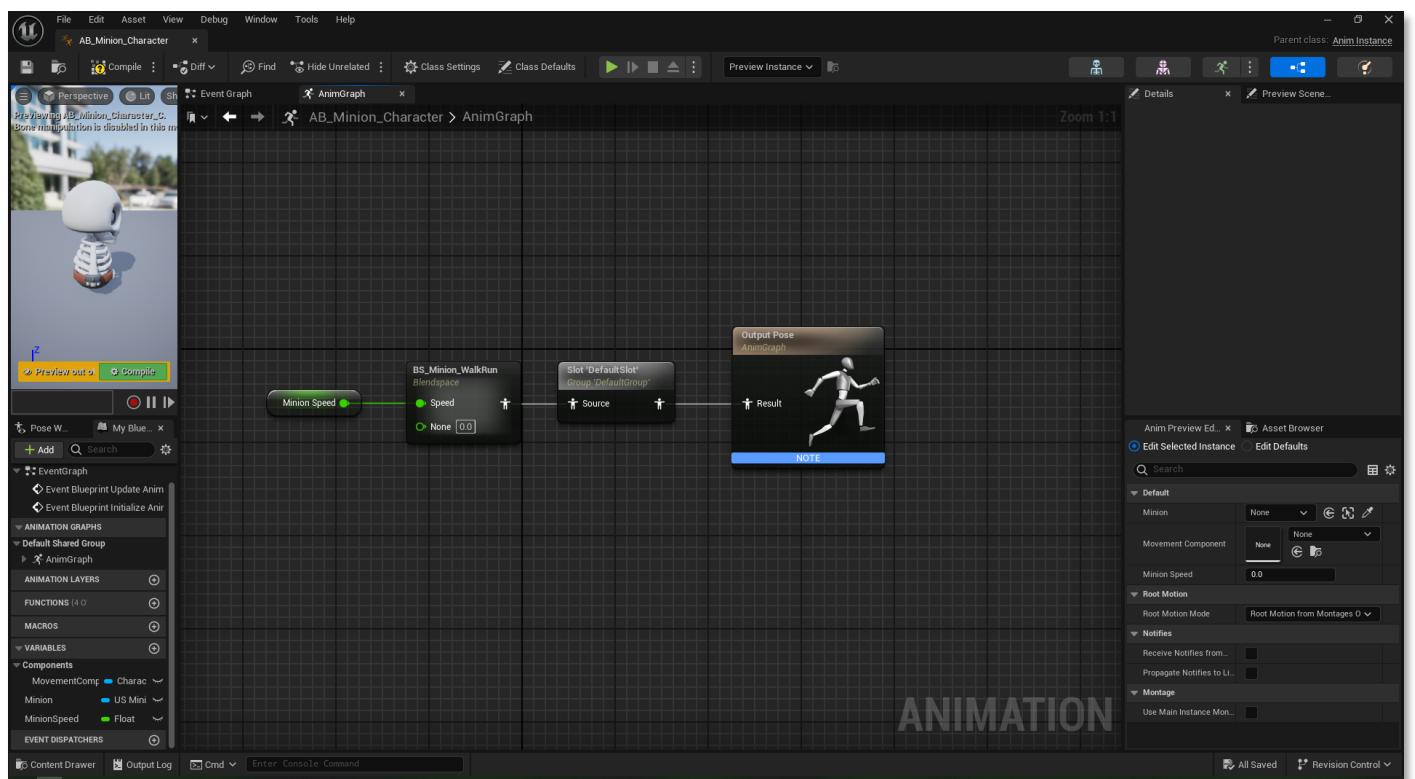


Fig 1.12: Screenshot showing Minion Animation Blueprint in Anim Graph

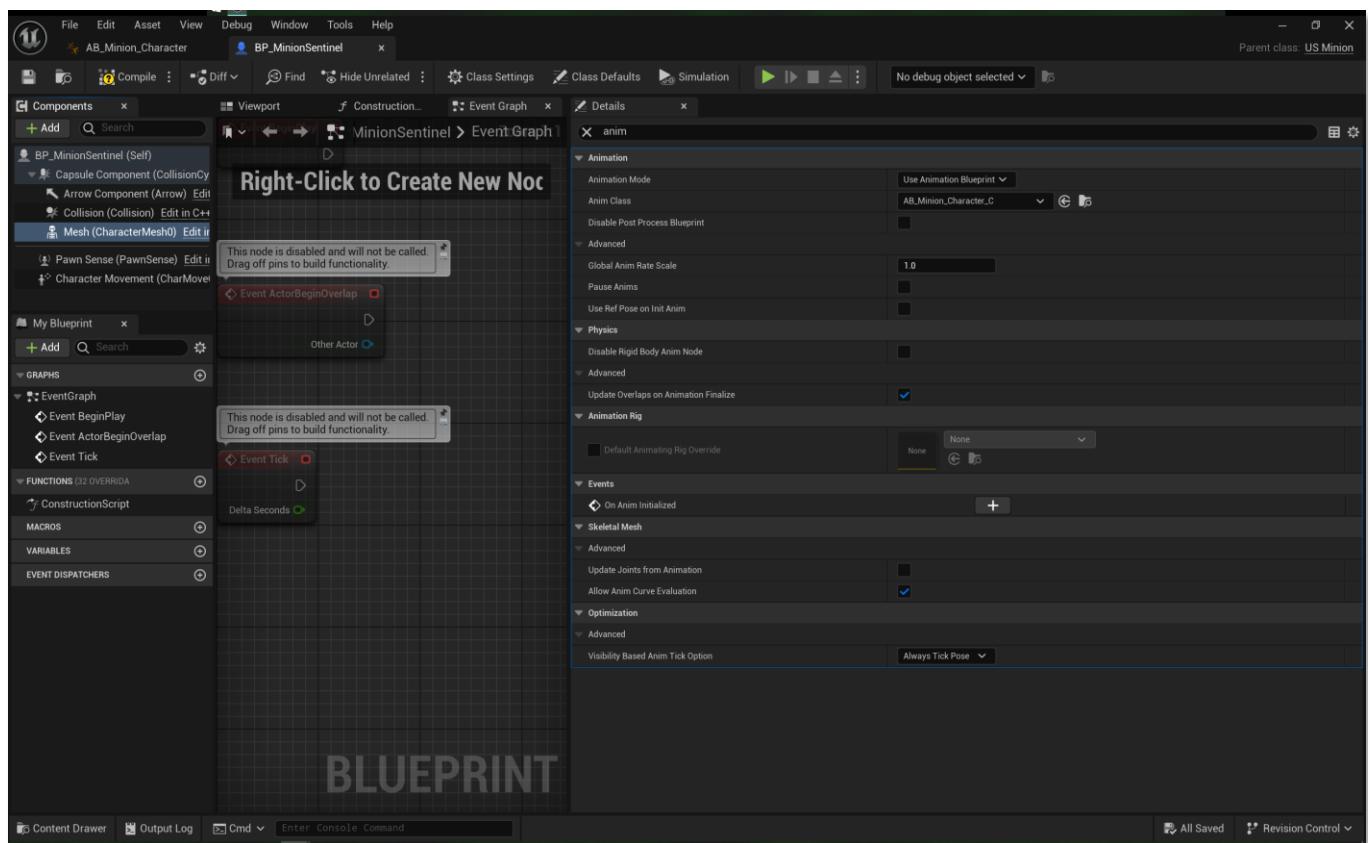


Fig 1.13: Screenshot showing addition of Anim Class for Minion Sentinel.

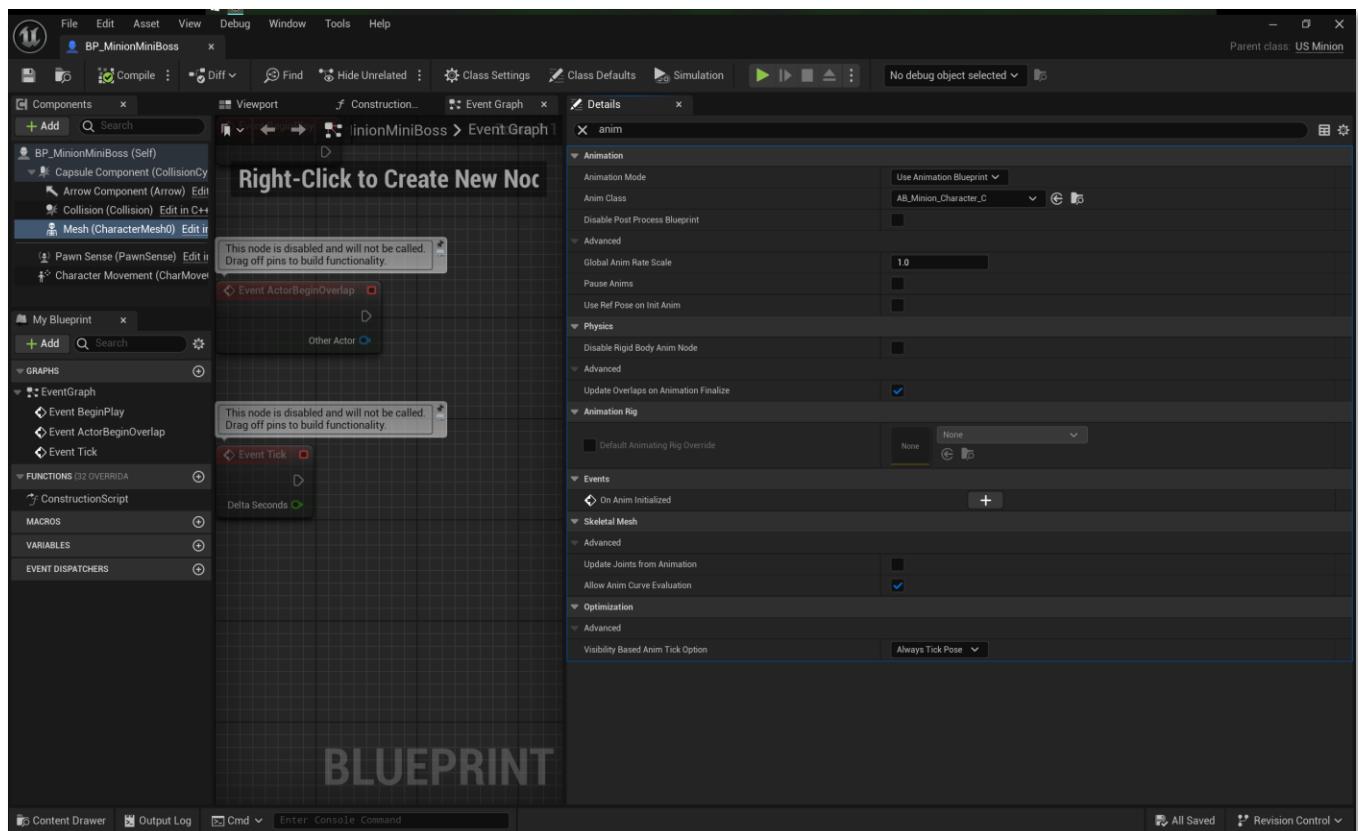


Fig 1.14: Screenshot showing addition of Anim Class for Minion MiniBoss.

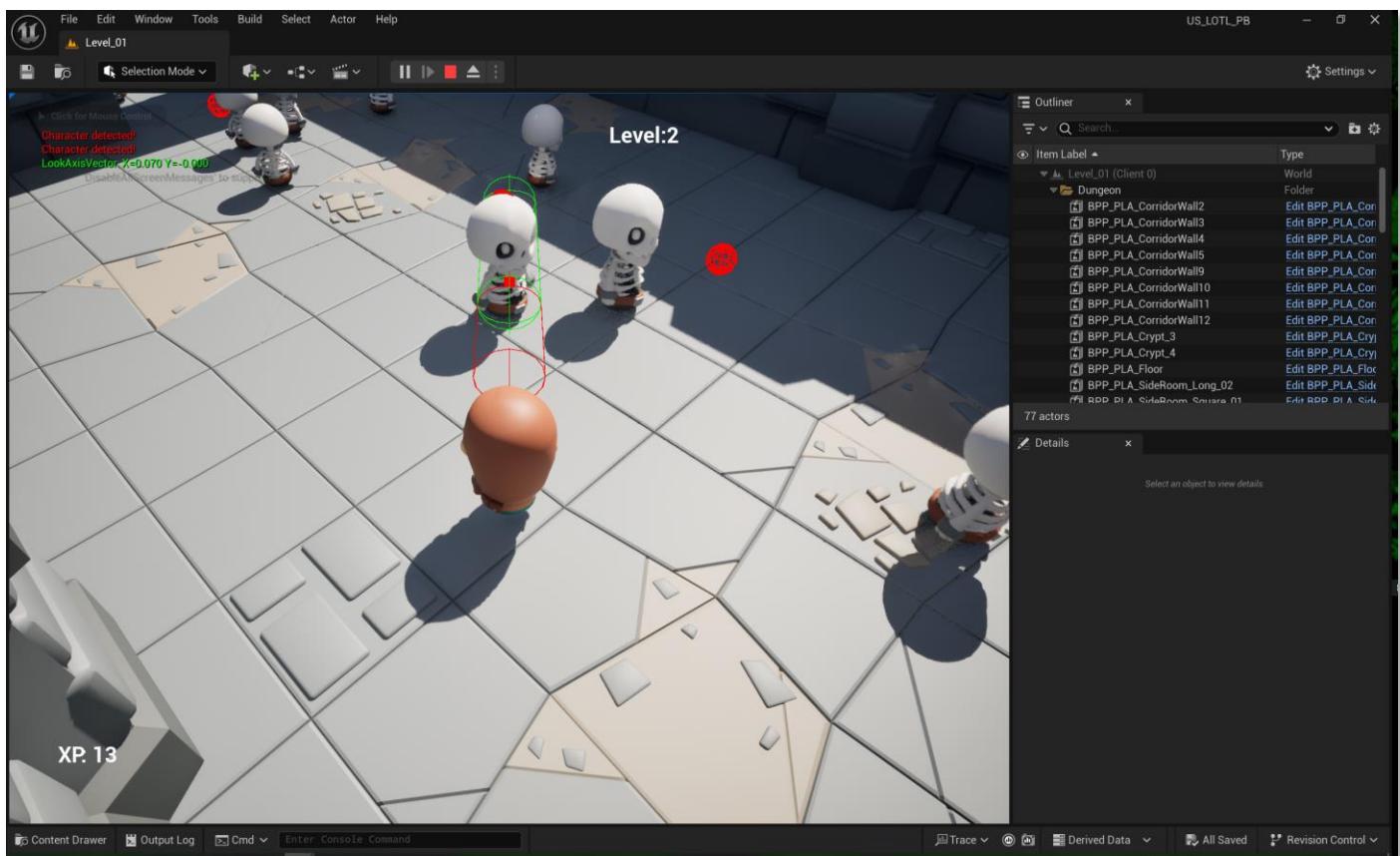


Fig 1.15: Screenshot showing testing (Parameters are set as per Mini Boss).

--- End of Challenge ---

```

US_WeaponProjectileComponent.h
18 UPROPERTY(EditAnywhere, BlueprintReadOnly, Category = "Input", meta = (AllowPrivateAccess))
19 class UInputMappingContext* WeaponMappingContext;
20 UPROPERTY(EditAnywhere, BlueprintReadOnly, Category = "Input", meta = (AllowPrivateAccess))
21 class UInputAction* ThrowAction;
22
23 UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = "Projectile", meta = (AllowPrivateAccess))
24 class UAnimMontage* ThrowAnimation;
25
26 public:
27     // Sets default values for this component's properties
28     UUSS_WeaponProjectileComponent();
29
30 protected:
31     // Called when the game starts
32     virtual void BeginPlay() override;
33
34     void Throw();
35     UFUNCTION(Server, Reliable)
36     void Throw_Server();
37
38     UFUNCTION(NetMulticast, Unreliable)
39     void Throw_Client();
40
41 public:
42     // Called every frame
43     virtual void TickComponent(float DeltaTime, ELevelTick TickType, FActorComponentTickFunction* ThisComponentTickFunction) override;
44
45     UFUNCTION(BlueprintCallable, Category = "Projectile")
46     void SetProjectileClass(TSubclassOf<class AUS_BaseWeaponProjectile> NewProjectileClass);
47
48 };
49

```

```

US_WeaponProjectileComponent.cpp
73     }
74
75     if (ProjectileClass)
76     {
77         Throw_Client();
78         TimerHandle TimerHandle;
79         GetWorld()->GetTimerManager().SetTimer(&InputHandle::TimerHandle, Callback, [&]()
80         {
81             const auto AUS_Character *const Character = Cast<AUS_Character>(Srcs GetOwner());
82             const auto const FVector ProjectileSpawnLocation = GetComponentLocation();
83             const auto const FRotator ProjectileSpawnRotation = GetComponentRotation();
84             auto FactorSpawnParams ProjectileSpawnParams = FactorSpawnParams();
85             ProjectileSpawnParams.Owner = GetOwner();
86             ProjectileSpawnParams.Instigator = Character;
87             GetWorld()->SpawnActor<AUSS_WeaponProjectile>(
88                 ProjectileClass, ProjectileSpawnLocation, ProjectileSpawnRotation,
89                 ), InLoop: false);
90         });
91     }
92
93     void UUSS_WeaponProjectileComponent::SetProjectileClass(TSubclassOf<AUSS_BaseWeaponProjectile> NewProjectileClass)
94     {
95         ProjectileClass = NewProjectileClass;
96     }
97
98     void UUSS_WeaponProjectileComponent::Throw_Client_Implementation()
99     {
100        const auto AUS_Character *const Character = Cast<AUS_Character>(Srcs GetOwner());
101        if (ThrowAnimation != nullptr)
102        {
103            if (const auto UAnimInstance *const AnimInstance = Character->GetMesh()->GetAnimInstance())
104            {
105                AnimInstance->Montage_Play(ThrowAnimation, 1.f);
106            }
107        }
108    }
109
110

```

Fig 1.16: Screenshot showing Weapon Configuration in US\_WeaponProjectileComponent.

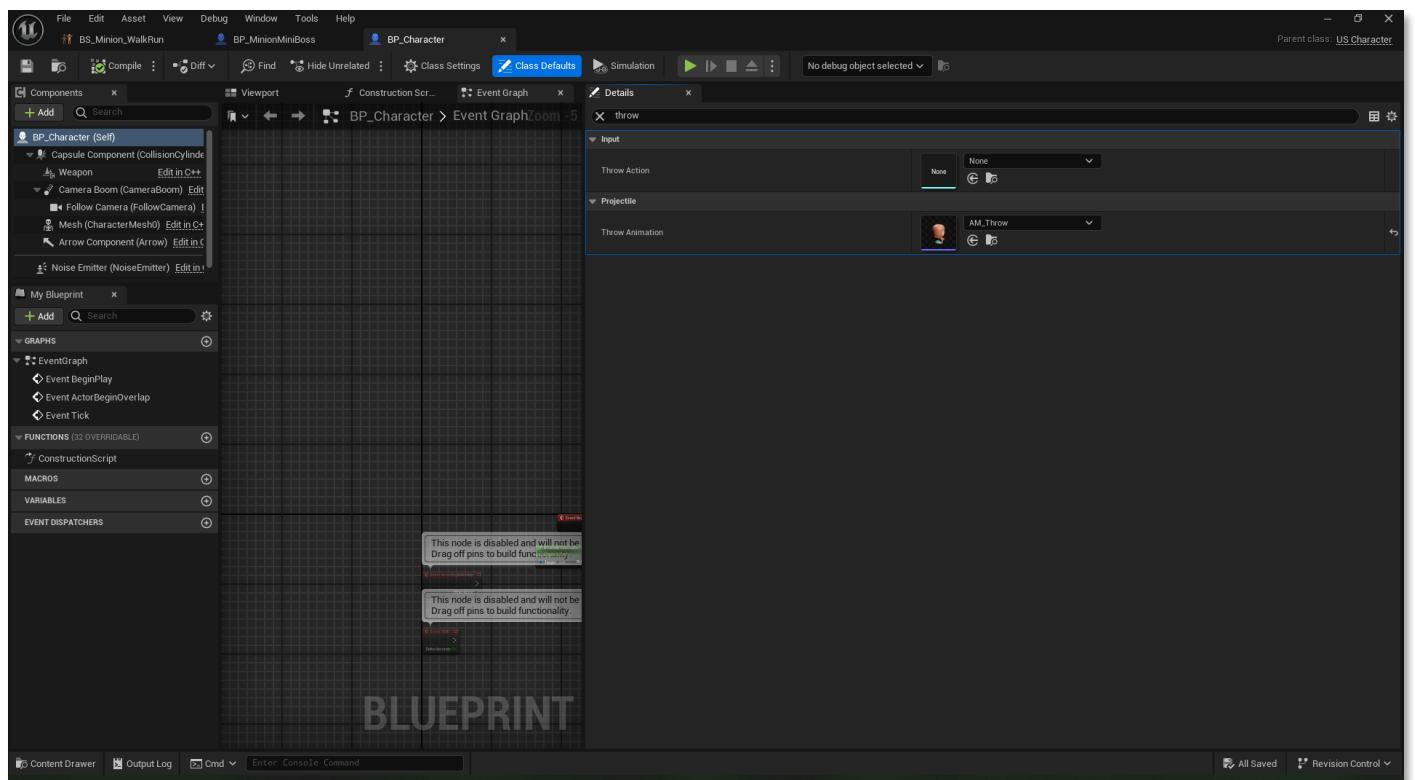


Fig 1.17: Screenshot showing Throw Animation assignment.

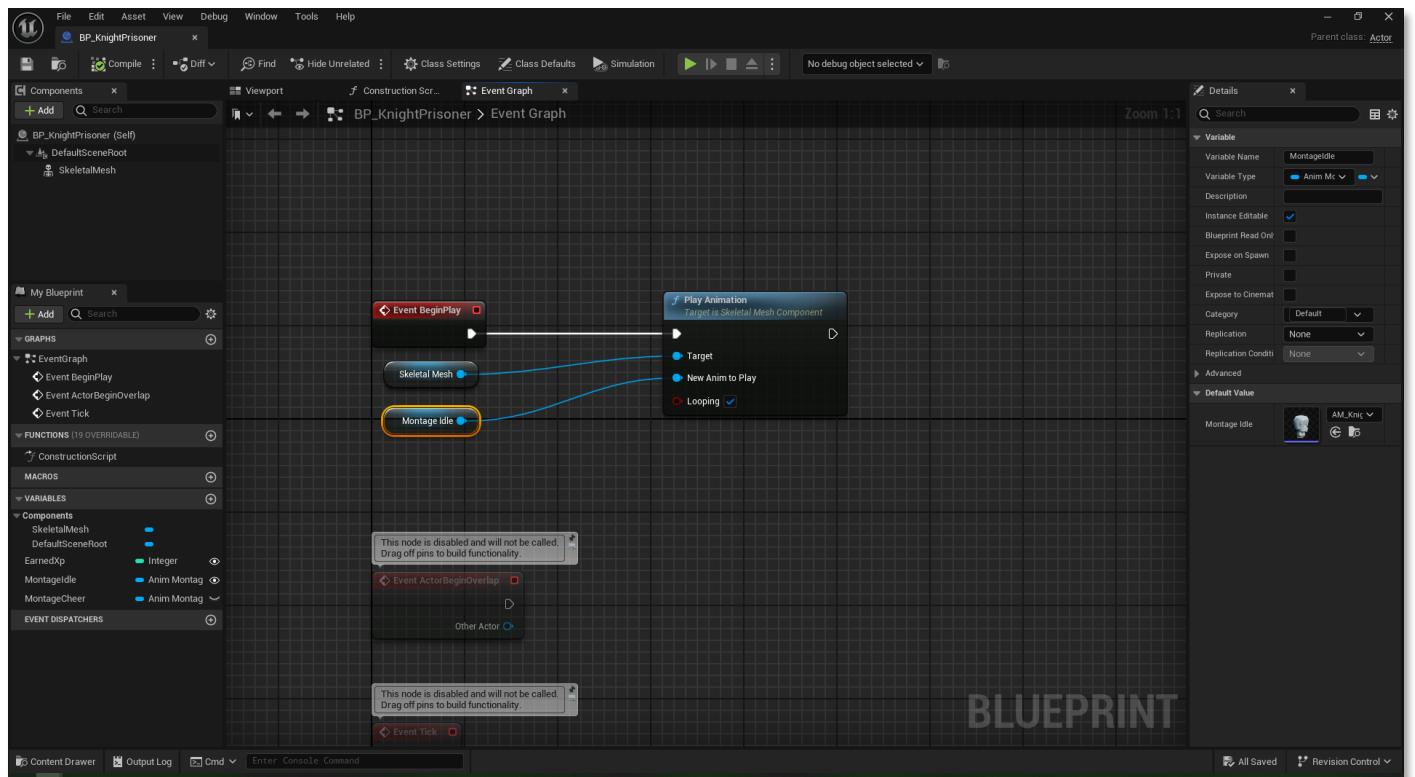


Fig 1.18: Screenshot showing BP\_KnightPrisoner with Animation Montages

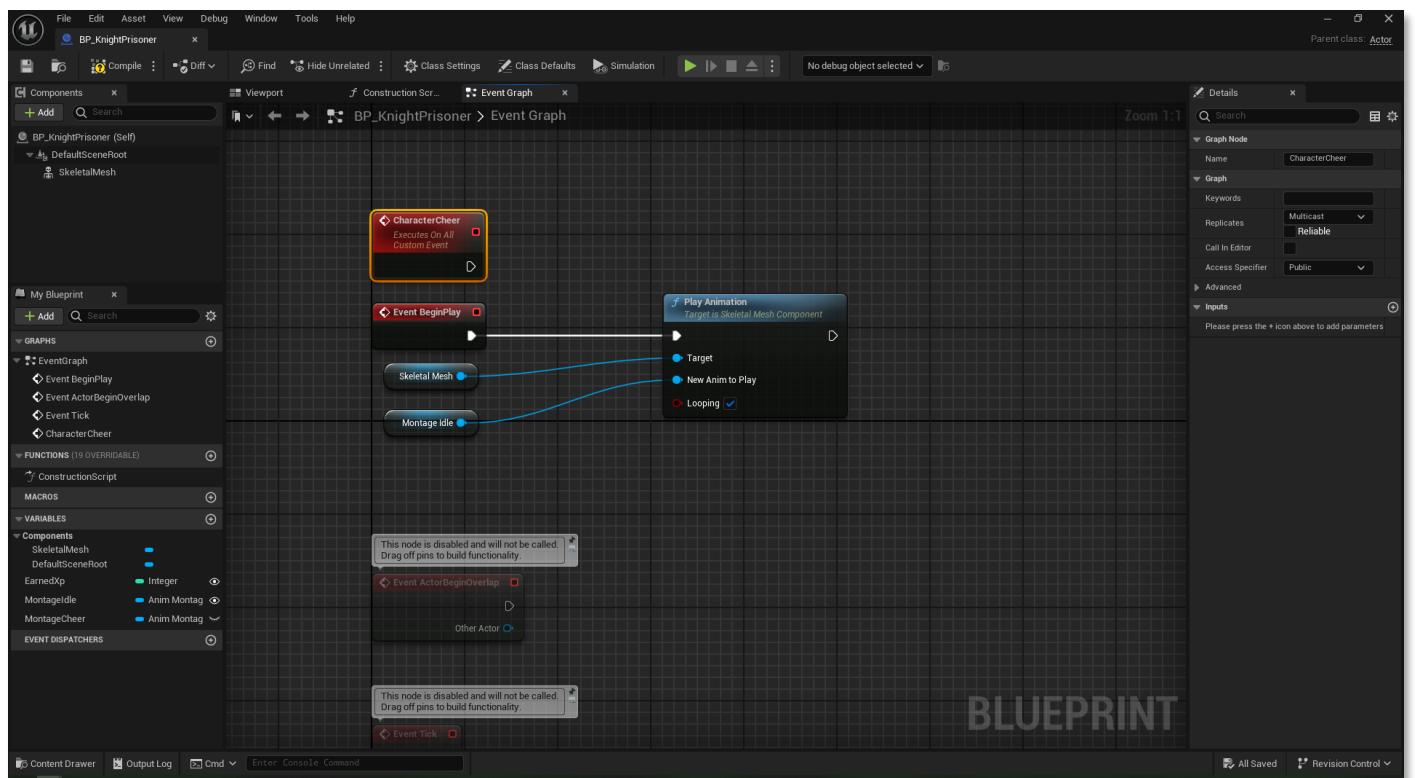


Fig 1.18: Screenshot showing BP\_KnightPrisoner event and setting for Multicast

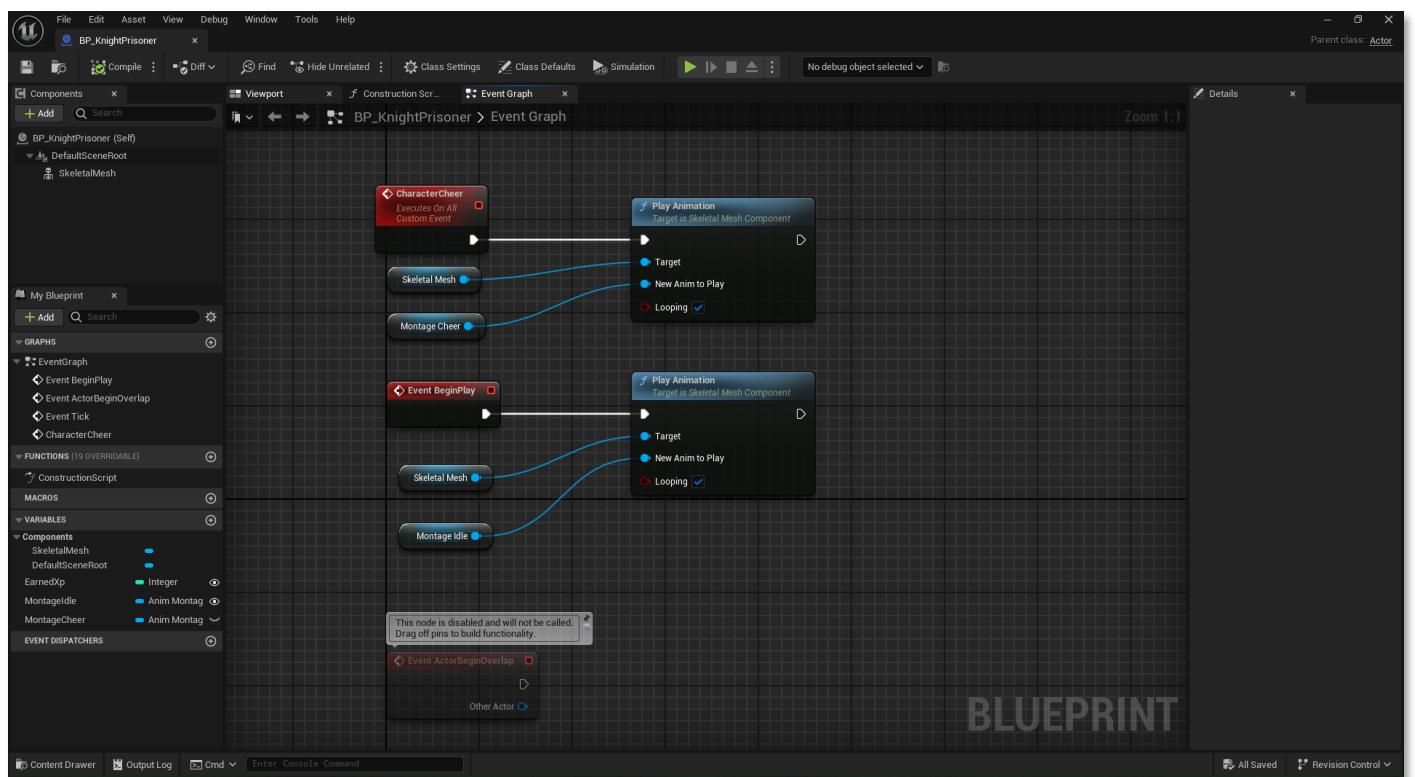


Fig 1.19: Screenshot showing BP\_KnightPrisoner custom event

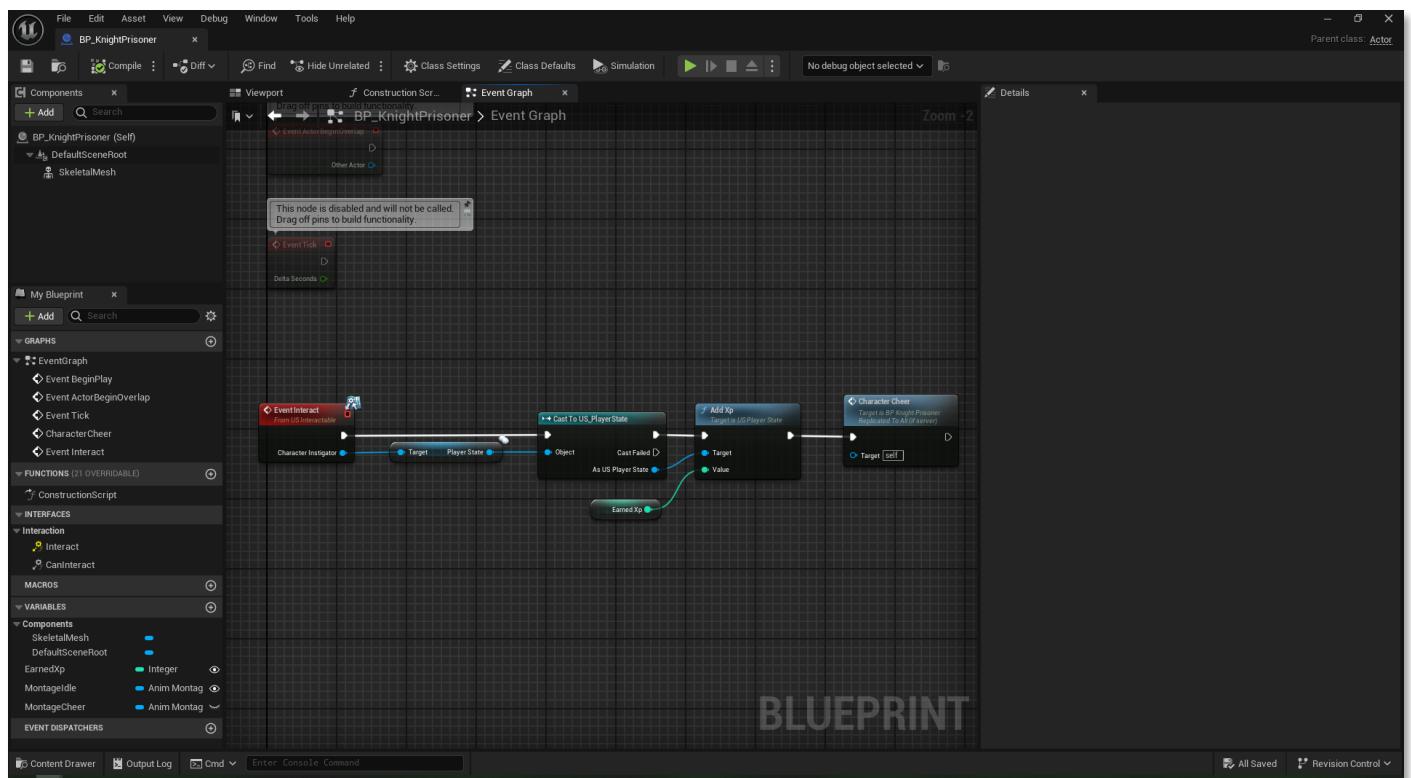


Fig 1.20: Screenshot showing addition of US\_Interactable Interface

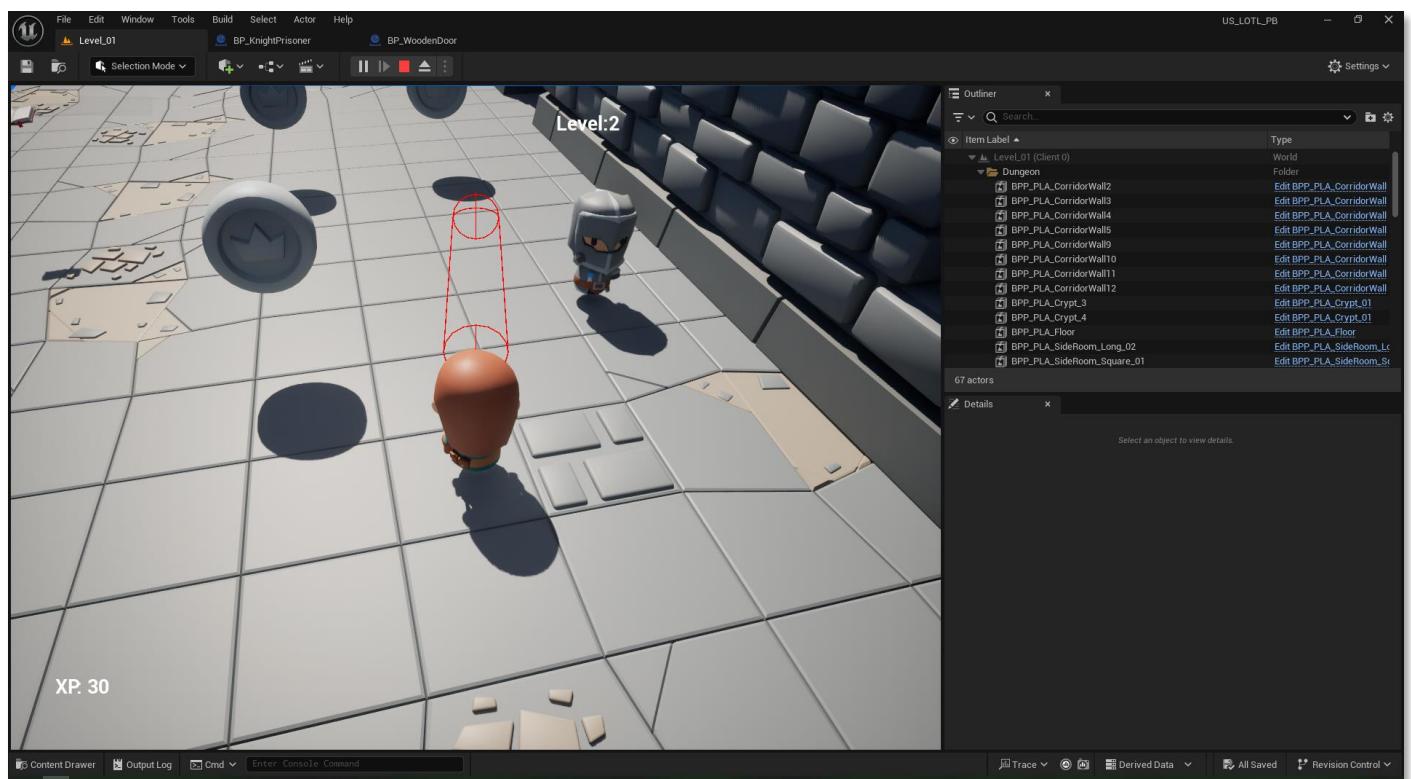


Fig 1.21: Screenshot showing cheer on interaction