The Gameplay Abilities Module

•••

Michael Chapman
(@woppin on Discord)

Gameplay Abilities

- A Common Plugin shared by Paragon and Fortnite
- Handle gameplay mechanics + their replication and prediction

Why?

- Allow engineers to handle the network prediction and replication in C++
- Allow designers to work in blueprint to tweak mechanics easily
- Generic enough to support anything the design team comes up with
- Optimised to reduce network load

Gameplay Abilities

Resources available:

- Sample from Epic dev Dave Ratti
  - https://github.com/daveratti/GameplayAbilitiesSample
- Forum post and wiki article from Kaz
  - https://wiki.unrealengine.com/GameplayAbilities_and_You
- #gameplay-abilities-plugin channel on Unreal Slackers discord community
- Code from this talk:
  - https://github.com/michaeltchapman/MCGameplayAbilities
- Sabre Dart Studios training:
  - https://www.youtube.com/watch?v=Ev2P6BTUxN0

Concepts

AbilitySystemComponent

AttributeSets

GameplayTags

GameplayEffects

GameplayCues

GameplayAbilities

GameplayTasks

GameplayEvents

Gameplay Abilities : AbilitySystemComponent

Central point for coordinating everything

Attach to any actor implementing `IAbilitySystemInterface`

```
AbilitySystem = CreateDefaultSubobject<UAbilitySystemComponent>(TEXT("AbilitySystem"));
```

On pawns after possession: `AbilitySystem->InitAbilityActorInfo(this, this);`

Relatively lightweight, attach to anything that should receive effects or activate abilities

Gameplay Abilities : AttributeSets

Store replicated floats wrapped in a struct

```
UPROPERTY(Category = "Attribute", EditAnywhere, ReplicatedUsing = OnRep_Health,
BlueprintReadWrite) FGameplayAttributeData Health;

UFUNCTION() void OnRep_Health() { GAMEPLAYATTRIBUTE_REPNOTIFY(UBaseAttributeSet,
Health); }  static FGameplayAttribute AttributeHealth();
```

Account for temporary changes without losing base

```
AbilitySystem->GetNumericAttribute(UBaseAttributeSet::AttributeHealth());

AbilitySystem->GetNumericAttributeBase(UBaseAttributeSet::AttributeHealth());
```

Gameplay Abilities : AttributeSets

Attribute Change Events

```
void PreAttributeBaseChange(const FGameplayAttribute& Attribute, float& NewValue) const;
```

```
void PreAttributeChange(const FGameplayAttribute& Attribute, float& NewValue);
```

Used to clamp values Eg. health <= max_health

```
void PostGameplayEffectExecute(const struct FGameplayEffectModCallbackData &Data);
```

Used to trigger actions as a result of attribute changes, eg health < 0 -> do_ragdoll

```
bool PreGameplayEffectExecute(struct FGameplayEffectModCallbackData &Data);
```

Used to modify or discard the values passed into an effect execution.

Gameplay Abilities : GameplayTags

If attributes are the ‘float’ part of describing the game state, tags are the ‘bool’ part. Can be applied locally and have state managed manually:

```
AbilitySystem->AddMinimalReplicationGameplayTags(InitialTags);
```

More commonly applied via effects and replicated. Can be applied multiple times and stack.

Triggers events on both client and server

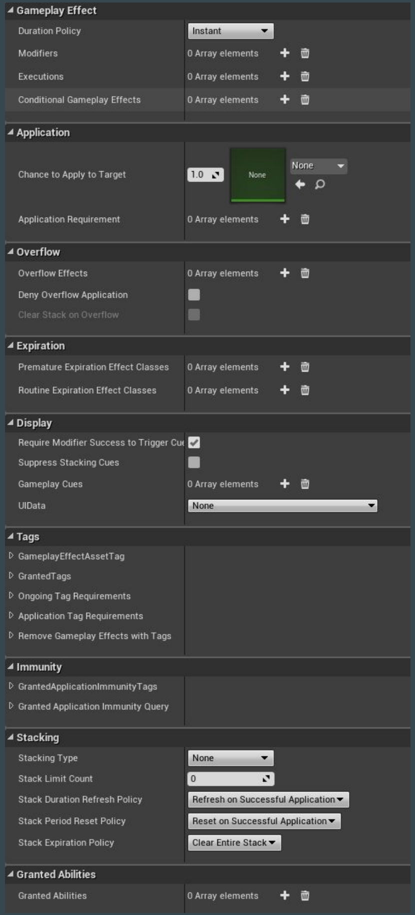
```
FOnGameplayEffectTagCountChanged& LitheCallback =
AbilitySystem->RegisterGameplayTagEvent(FGameplayTag::RequestGameplayTag(UGlobalData::Lit
heTag), EGameplayTagEventType::AnyCountChange);

LitheCallback.AddUObject(this, &ABaseCharacter::SetLithe);
```

Gameplay Abilities : GameplayEffects

Predictively applied when done via an ability

- Can modify attributes
- Can add/remove tags
- Can be instant, have set duration, or be infinite
- Can trigger a Cue
- Can have tags itself
- Can be blocked by tags on the source or target
- Can stack, and overflow the stack
- Can trigger other effects
- Can remove other effects
- Can trigger events
- Can have a context attached to it, including a FHitResult.



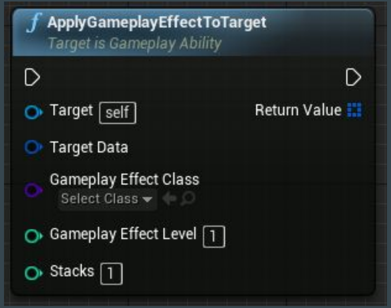
Gameplay Effect	
Duration Policy	Instant
Modifiers	0 Array elements
Executions	0 Array elements
Conditional Gameplay Effects	0 Array elements
Application	
Chance to Apply to Target	1.0 None None
Application Requirement	0 Array elements
Overflow	
Overflow Effects	0 Array elements
Deny Overflow Application	☐
Clear Stack on Overflow	☐
Expiration	
Premature Expiration Effect Classes	0 Array elements
Routine Expiration Effect Classes	0 Array elements
Display	
Require Modifier Success to Trigger Cue	✓
Suppress Stacking Cues	☐
Gameplay Cues	0 Array elements
UIData	None
Tags	
GameplayEffectAssetTag	
GrantedTags	
Ongoing Tag Requirements	
Application Tag Requirements	
Remove Gameplay Effects with Tags	
Immunity	
GrantedApplicationImmunityTags	
Granted Application Immunity Query	
Stacking	
Stacking Type	None
Stack Limit Count	0
Stack Duration Refresh Policy	Refresh on Successful Application
Stack Period Reset Policy	Reset on Successful Application
Stack Expiration Policy	Clear Entire Stack
Granted Abilities	
Granted Abilities	0 Array elements

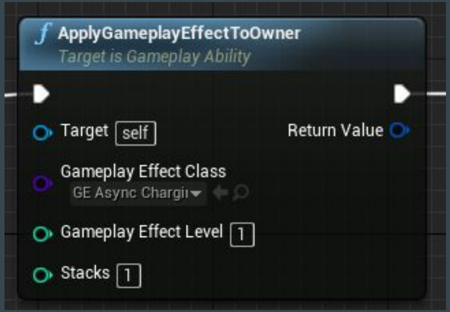
Gameplay Abilities : GameplayEffects

Can be applied via an ability, or using the ASC of the target. The latter won't be predicted

- Create GameplayEffectSpec from class
- Add or Remove tags
- SetByCallerMagnitude
- Add Effect Context
- Apply Effect Spec to Self/Target

The blueprint nodes for use within abilities omit the middle 3 steps.

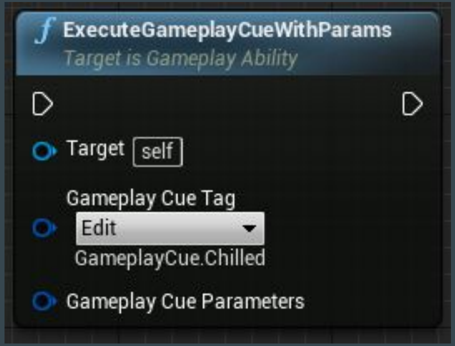

ApplyGameplayEffectToTarget
Target is Gameplay Ability
Target self
Return Value
Target Data
Gameplay Effect Class
Select Class
Gameplay Effect Level 1
Stacks 1


ApplyGameplayEffectToOwner
Target is Gameplay Ability
Target self
Return Value
Gameplay Effect Class
GE Async Chargi
Gameplay Effect Level 1
Stacks 1

Gameplay Abilities : GameplayCues

Cues encapsulate cosmetic effects

- Driven by a tag namespace ‘GameplayCue’
- Can be triggered by GameplayEffects
- Can be executed by a predicting client in ability and replicated out
- Can be executed locally and not replicated

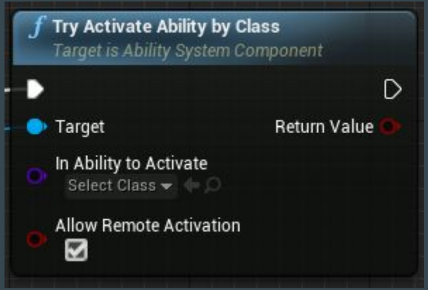


Cues hook into several events from the cue manager

- OnActive
  - Called when GameplayCue is activated.
- WhileActive
  - Called when GameplayCue is active, even if it wasn't actually just applied (Join in progress, etc)
- Executed
  - Called when a GameplayCue is executed: instant effects or periodic tick
- Removed
  - Called when GameplayCue is removed

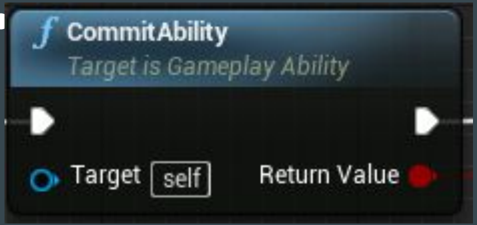
Gameplay Abilities : GameplayAbilities

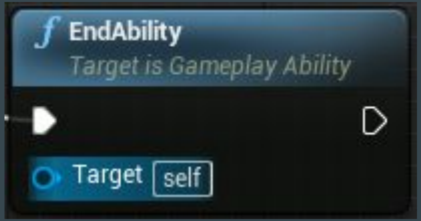
- Provide a (optionally) network-predicted way of interacting with all of these elements
- Lots of tag options:
    - Cancel other abilities
    - Block other abilities from activating
    - Require tags to activate
    - Block from activating
    - Apply to owner when activated
- Instancing:
    - Per Actor
    - Per Execution
    - Non Instanced
- Net Execution Policy
    - Local predicted
    - Local
    - Server Initiated
    - Server Only
- Cooldown effect class
- Cost Effect class



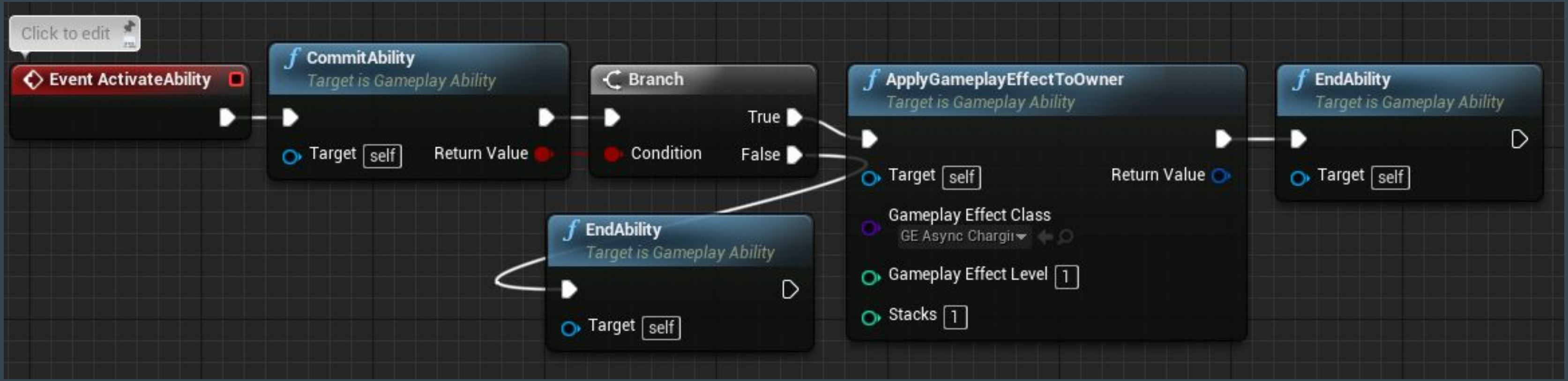
Gameplay Abilities : GameplayAbilities

- Most player abilities will be local predicted. This will have the ability graph run on both the client and the server.
- Often there will be a WaitTargetData node at the beginning. This allows the player to visualise aiming.
- The cooldown and cost will be applied when 'CommitAbility' is used.
- All paths must lead to EndAbility, or the UGameplayAbility instance will not be cleaned up.



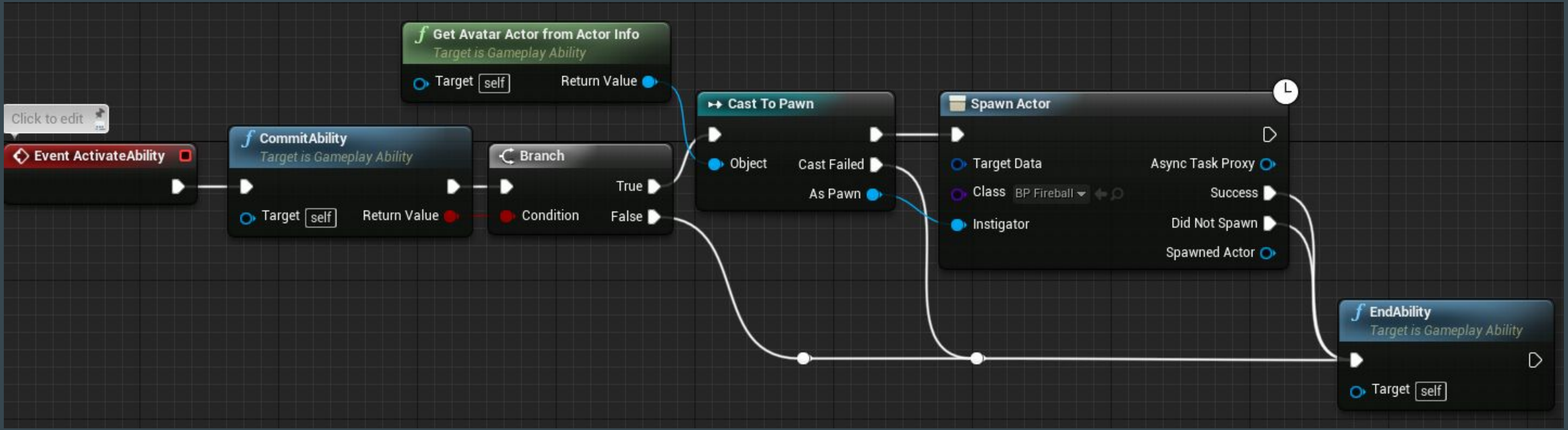


Gameplay Abilities : GameplayAbilities



Simple ability that applies an effect to the caster

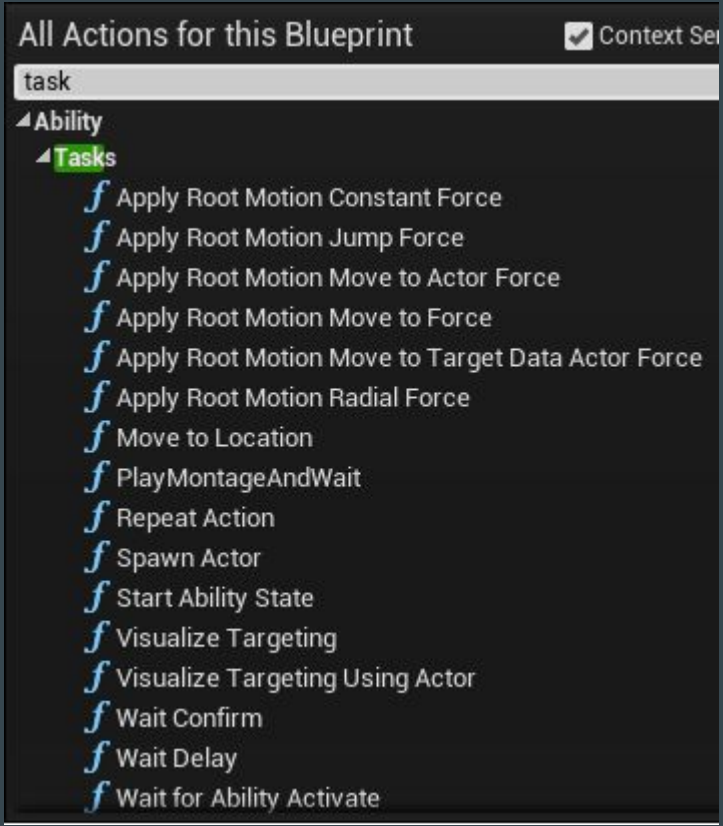
Gameplay Abilities : GameplayAbilities



Simple ability that spawns a projectile.

Gameplay Abilities : GameplayTasks

- Flexible task running framework, can be used separately from GameplayAbilities
- The GAS component inherits from the task management component
- bSimulatedTask
- bTickingTask
- AbilityTask adds some convenience pieces on top of GameplayTask
- Designer focused



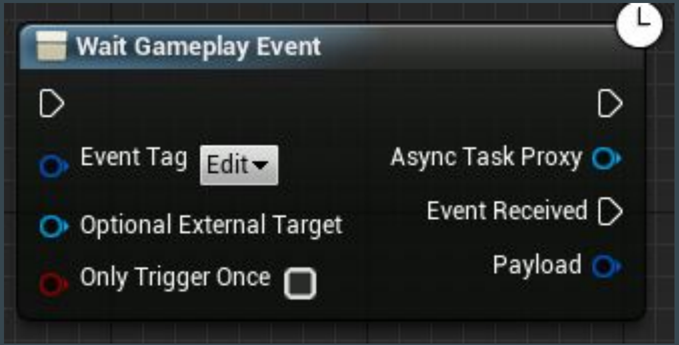
All Actions for this Blueprint ✓ Context Se

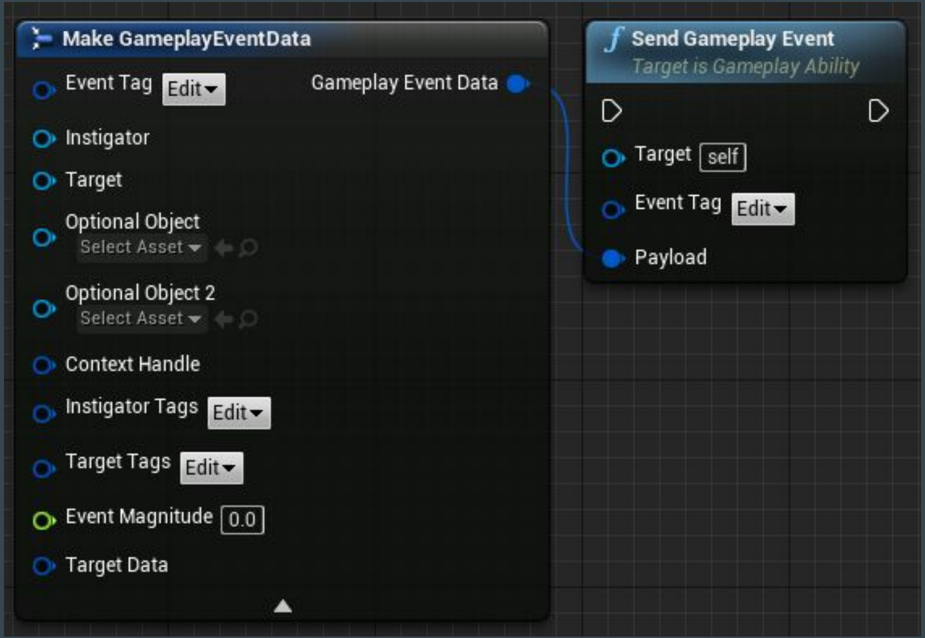
task

Ability
- Tasks
  - Apply Root Motion Constant Force
  - Apply Root Motion Jump Force
  - Apply Root Motion Move to Actor Force
  - Apply Root Motion Move to Force
  - Apply Root Motion Move to Target Data Actor Force
  - Apply Root Motion Radial Force
  - Move to Location
  - PlayMontageAndWait
  - Repeat Action
  - Spawn Actor
  - Start Ability State
  - Visualize Targeting
  - Visualize Targeting Using Actor
  - Wait Confirm
  - Wait Delay
  - Wait for Ability Activate

Gameplay Abilities : GameplayEvents

- Trigger things from other things.
- Associated with a tag and a UObject payload
- Trigger Abilities
- Wait for Event





Acknowledgements

- Thanks to the many friendly folks on the discord channel:
  - @Kaz for taking the time to write everything up initially
  - @jackblue for helping me through the early stages
  - @Roy Awesome, @Dartnalla, @Boolean Fiasco and everyone else who has helped to build up the collective knowledge around this plugin.
  - Finally, best of luck to @Acren and his colleagues, who are working on Hyper-Jam down in Melbourne, which is built on the GAS.