Gameplay Ability System

Unreal Engine Documentation: The **Gameplay Ability System** is a highly flexible framework for building the types of abilities and attributes that you might find in an RPG or MOBA title. You can build actions or passive aggressive abilities for the characters in your games to use, and status effects that can build up or wear down various attributes as a result of these actions, additionally you can implement "cooldown" timers or resource costs to regulate the usage of these actions, change the level of the ability and its effects at each level, activate particle, sound effects, and more. The Gameplay Ability System can help you to design, implement, and efficiently network in-game abilities from as simple as jumping to as complex as your favorite character's ability set in any modern RPG or MOBA title.

Gameplay Ability System

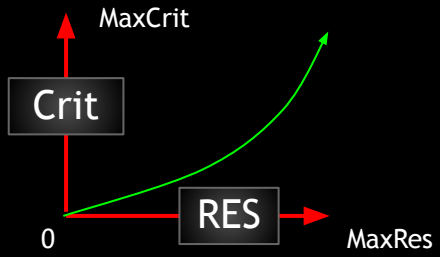
Aura

(Role-Playing Game)

Attributes

Life

Damage



Abilities

Play Animation

Cause Damage

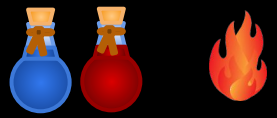
Fatal?

Gain Experience

XP Threshold?

Level Up!

Gameplay Ability System



Effects

Change Attribute Values

Cues

Particles

Sounds

(MOBA)

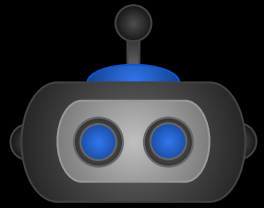
GAS

Gameplay Ability System

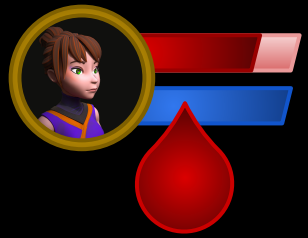
Ability System Component

Attribute Set

Gameplay Ability



Ability Task



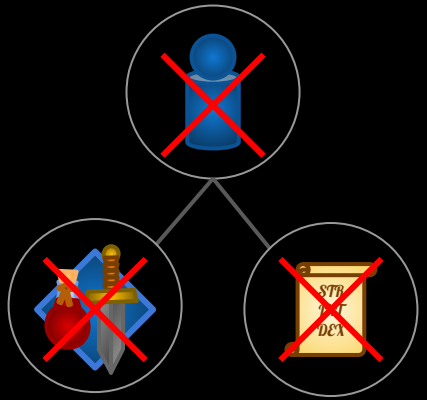
Gameplay Effect

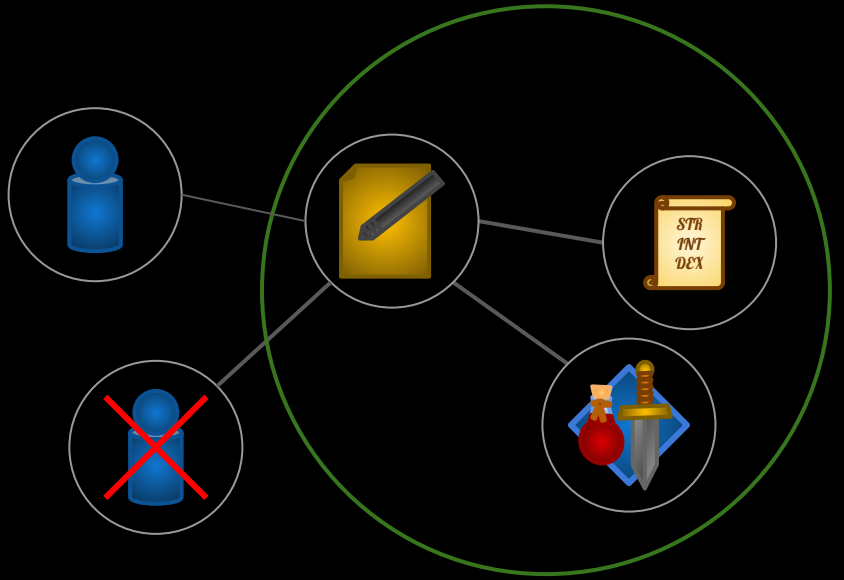
Gameplay Cue

Gameplay Tag

Gameplay Ability System

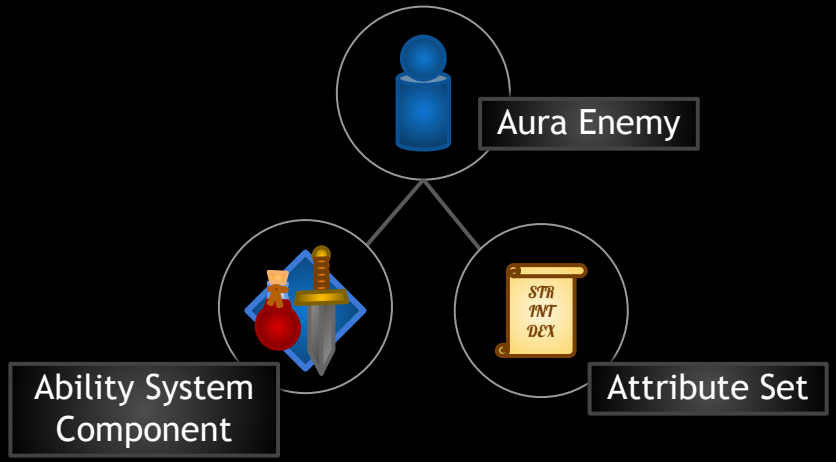
Gameplay Ability System



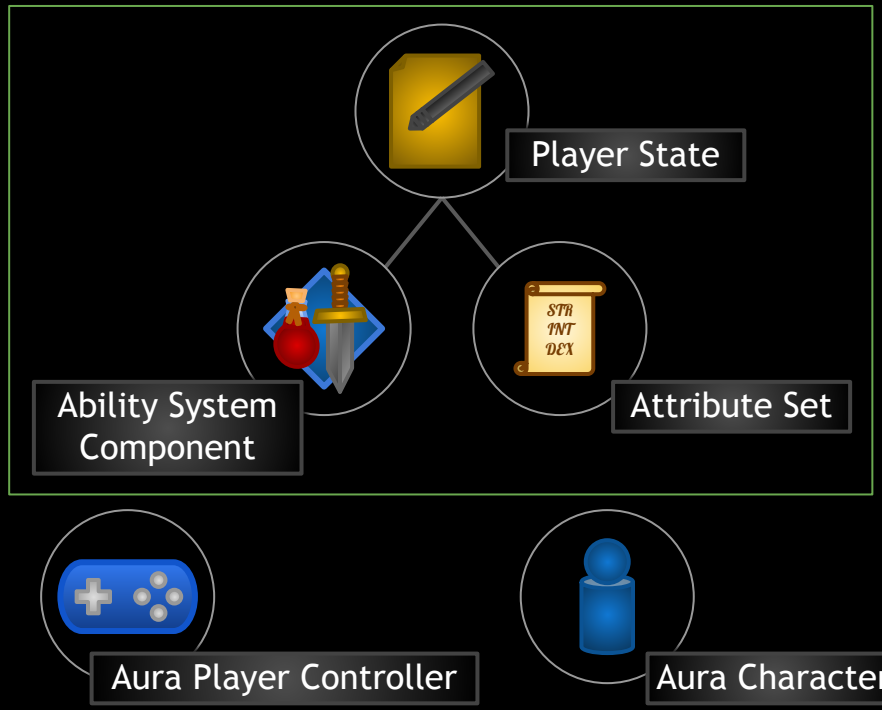


Gameplay Ability System

Enemy Character



Player Controlled Character

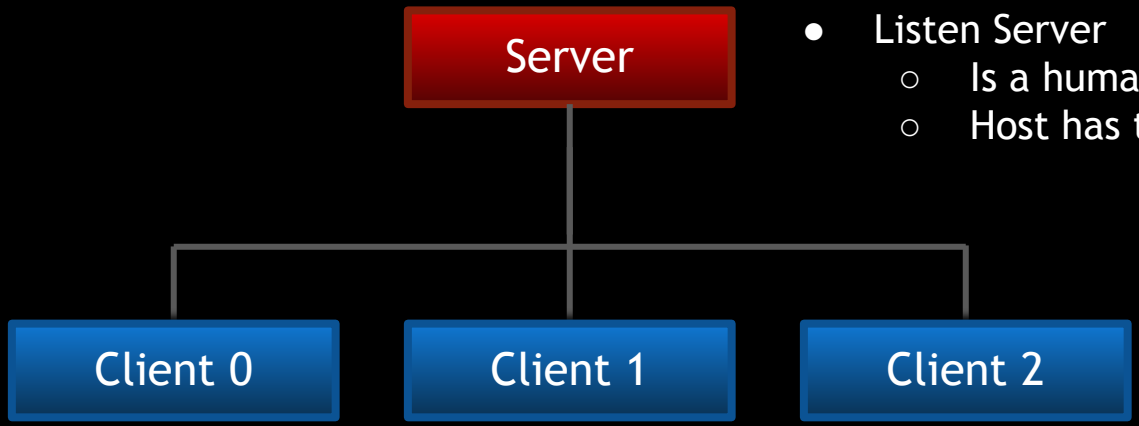


Gameplay Ability System

First Steps

1. Player State class
2. Ability System Component
3. Attribute Set

GAS in Multiplayer

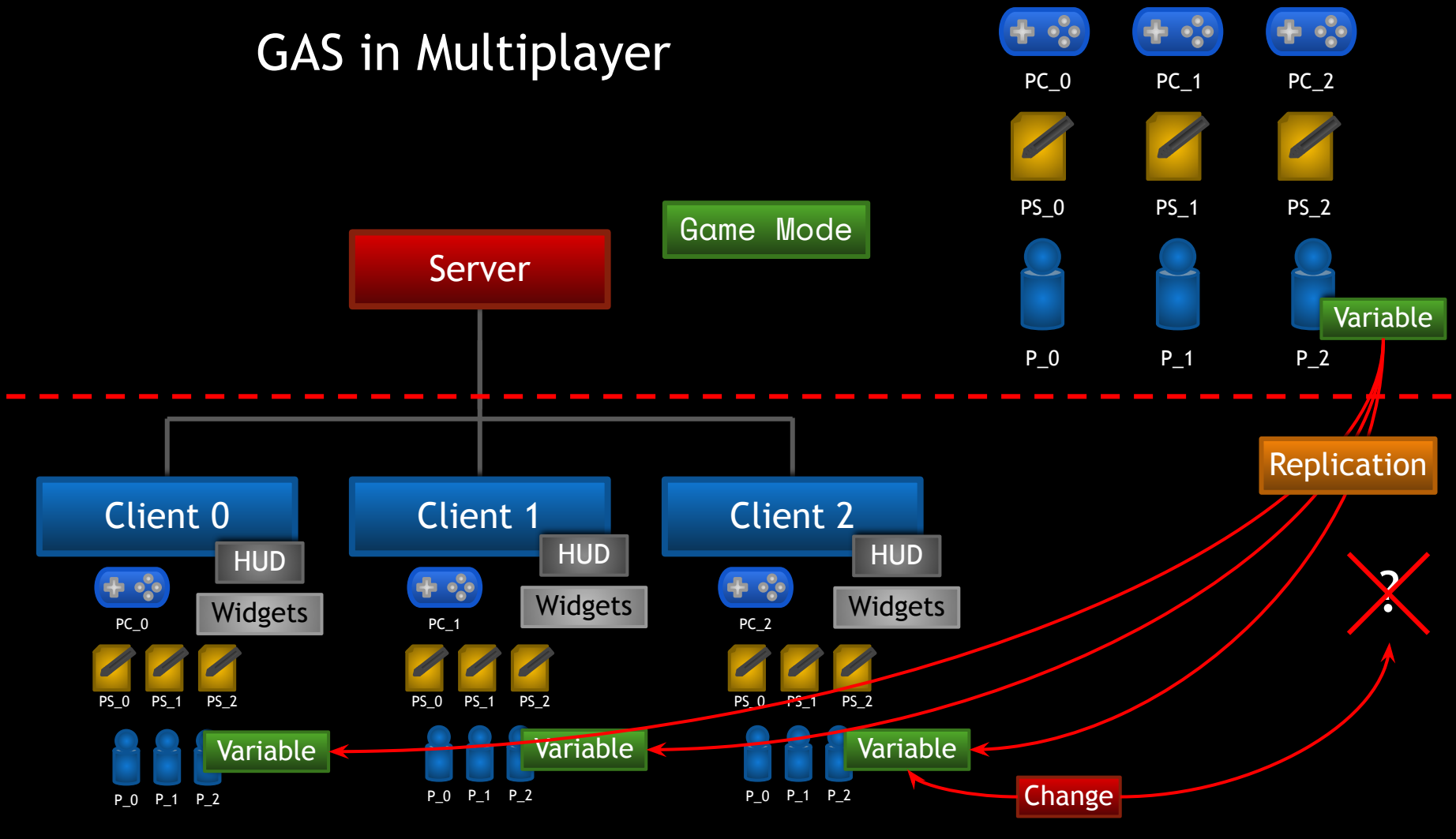


- Dedicated Server
  - No human player
  - No rendering to a screen
- Listen Server
  - Is a human player
  - Host has the advantage - no lag!

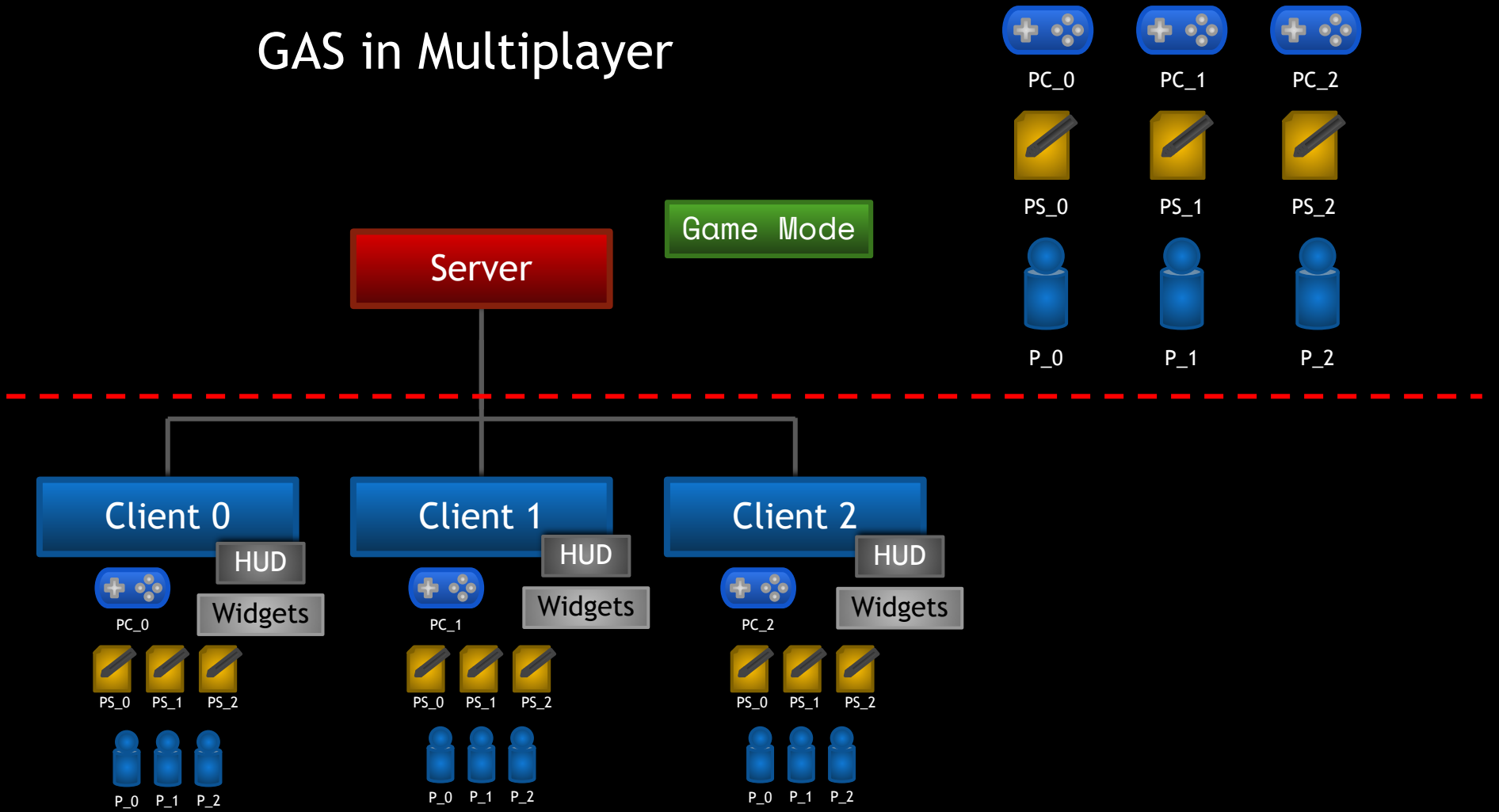
Server is the Authority

- “Correct” version of the game
- We do “important” things on the server
- We’ll learn what those are

GAS in Multiplayer



GAS in Multiplayer



Init Ability Actor Info

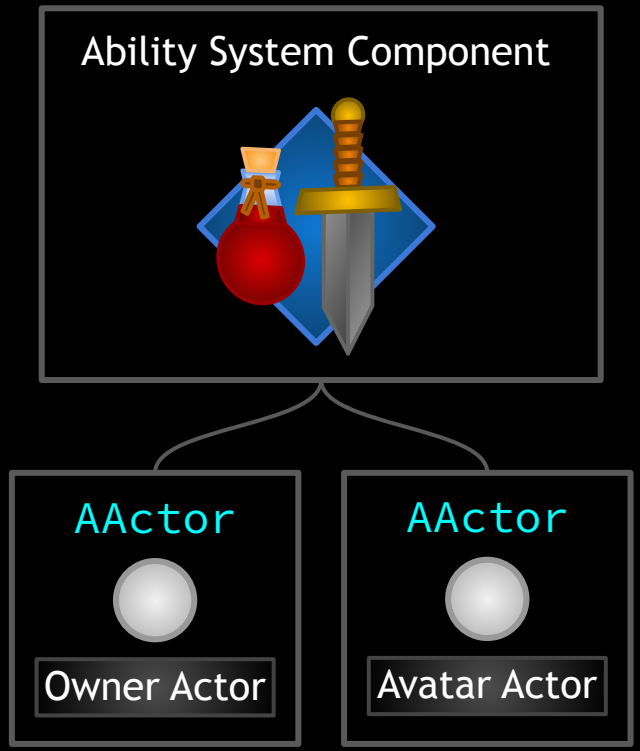
Replication Mode

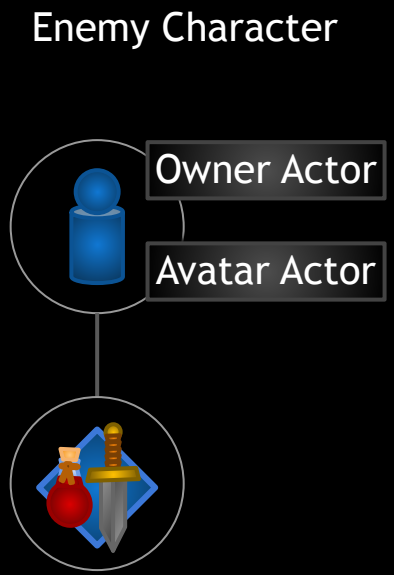
```
UAbilitySystemComponent::SetReplicationMode(EGameplayEffectReplicationMode::Mixed);
```

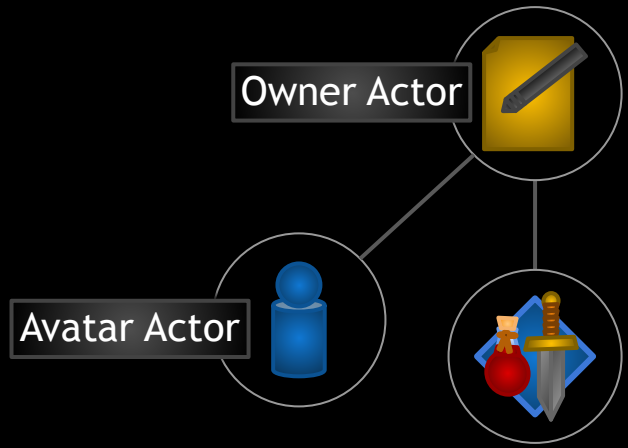
Replication Mode	Use Case	Description
`Full`	Single Player	Gameplay Effects are replicated to all clients
`Mixed`	Multiplayer, Player-Controlled	Gameplay Effects are replicated to the owning client only. Gameplay Cues and Gameplay Tags replicated to all clients.
`Minimal`	Multiplayer, AI-Controlled	Gameplay Effects are **not** replicated. Gameplay Cues and Gameplay Tags replicated to all clients.

Init Ability Actor Info

```
UAbilitySystemComponent::InitAbilityActorInfo(AActor* InOwnerActor, AActor* InAvatarActor);
```



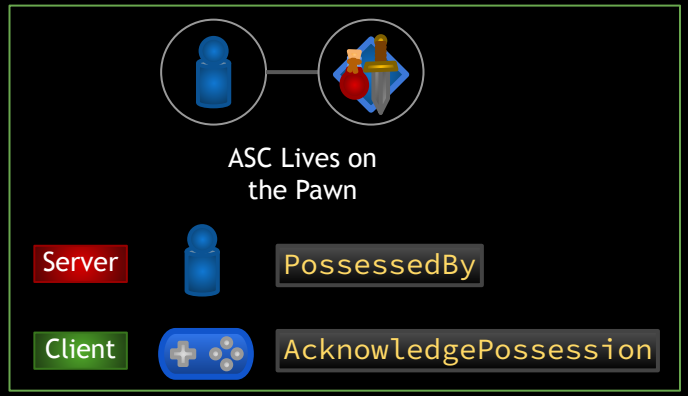




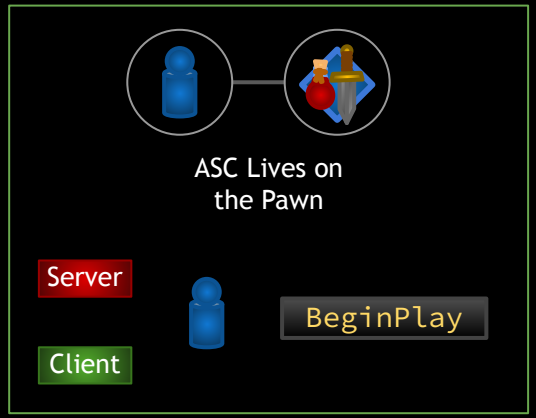
Init Ability Actor Info

- Must be done after possession (the Controller has been set for the Pawn)

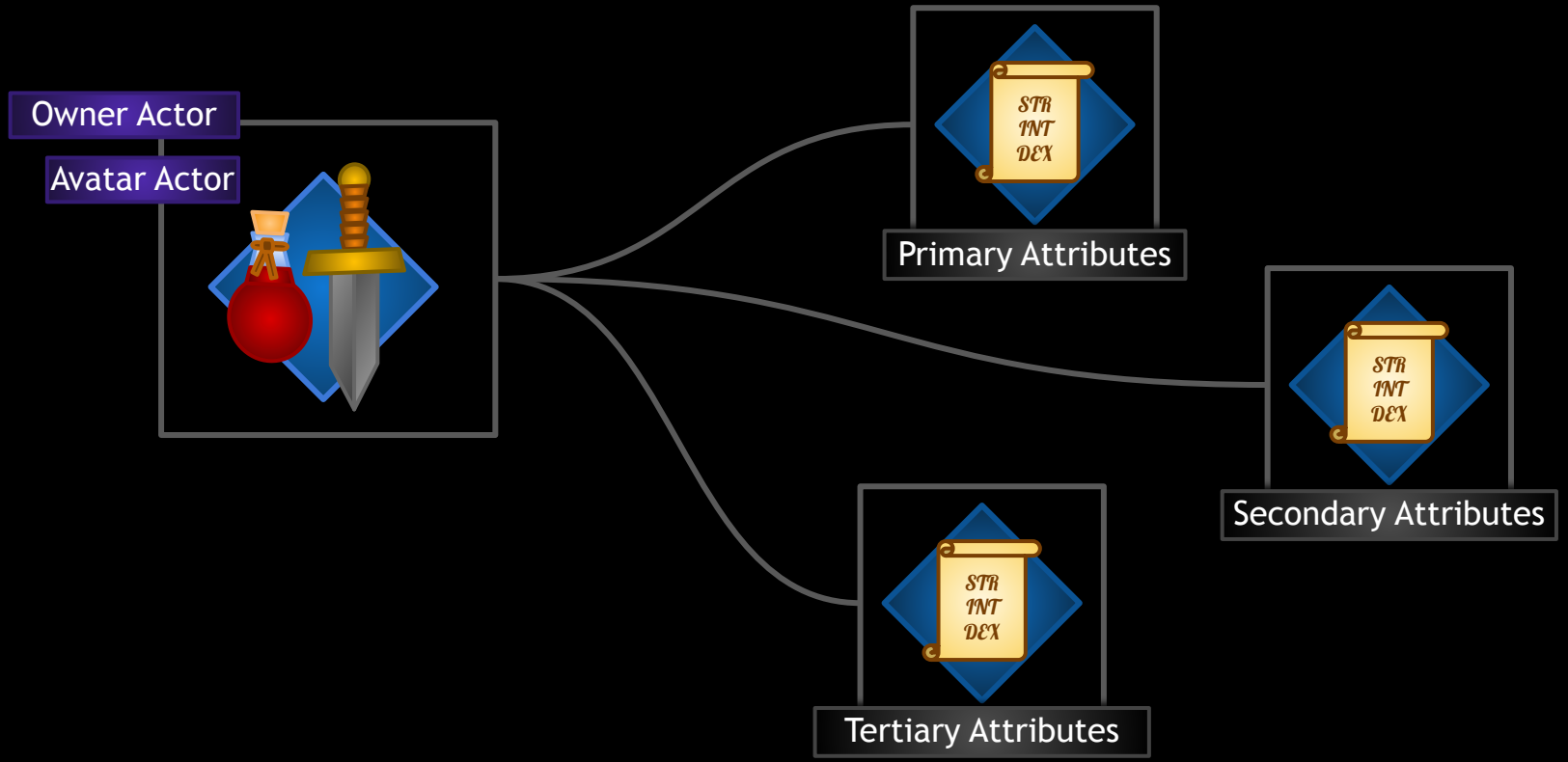
Player-Controlled Character



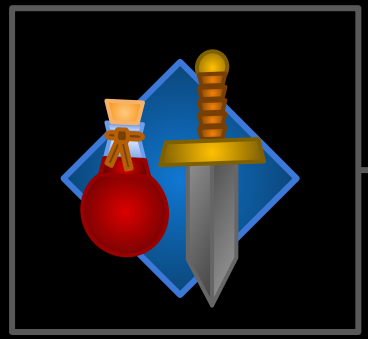
AI-Controlled Character

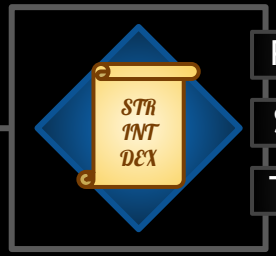


Attributes


Owner Actor
Avatar Actor
STR
INT
DEX
Primary Attributes
STR
INT
DEX
Secondary Attributes
STR
INT
DEX
Tertiary Attributes

Attributes



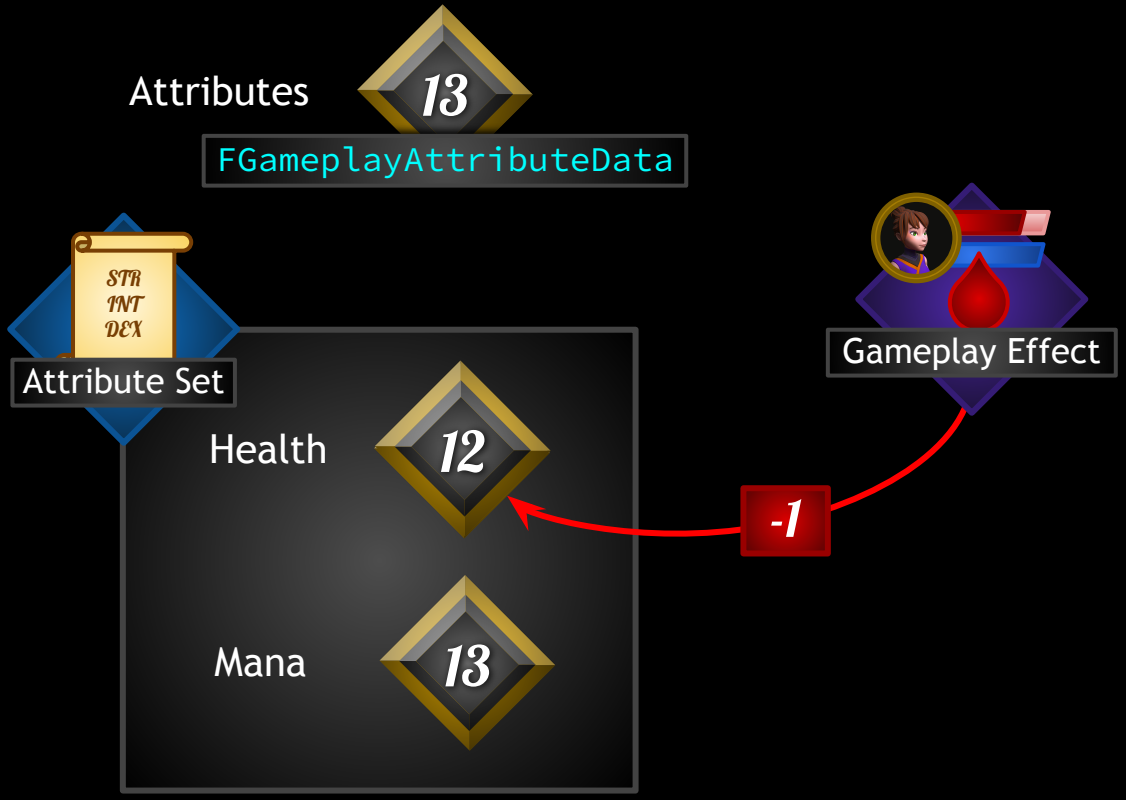


Primary Attributes

Secondary Attributes

Tertiary Attributes

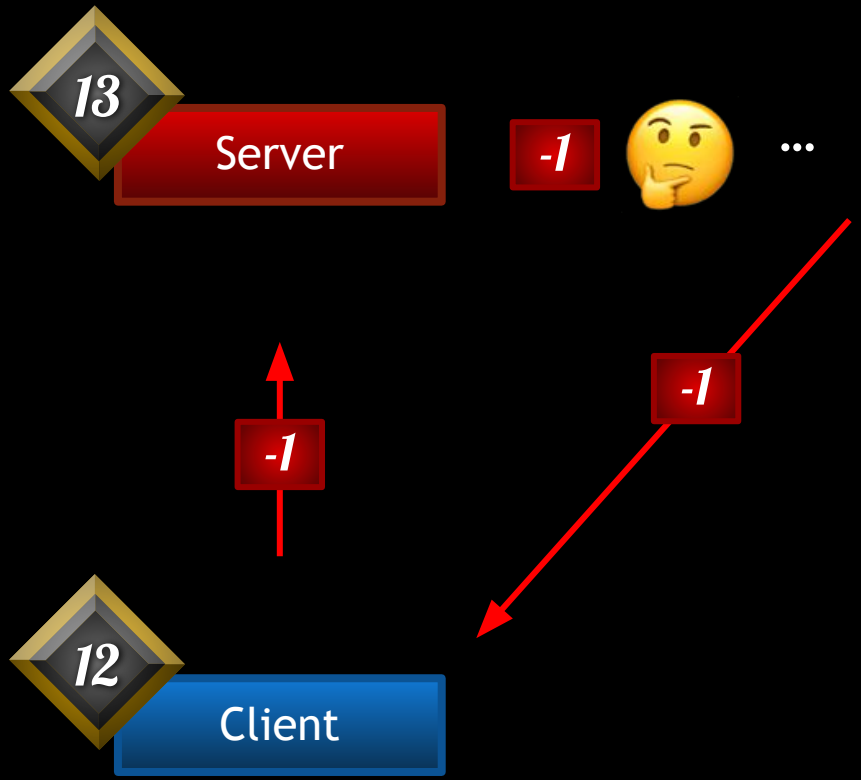
Attributes



Prediction: The client doesn't need to wait for the server's permission to change a value. The value can change immediately client-side and the server is informed of the change. The server can roll back changes that are invalid.

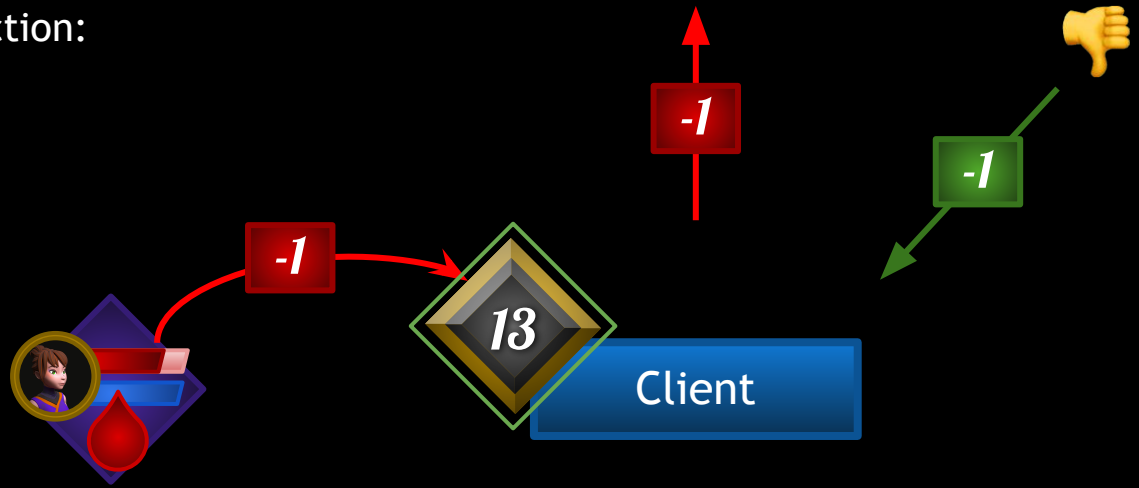
Attributes

Without Prediction:

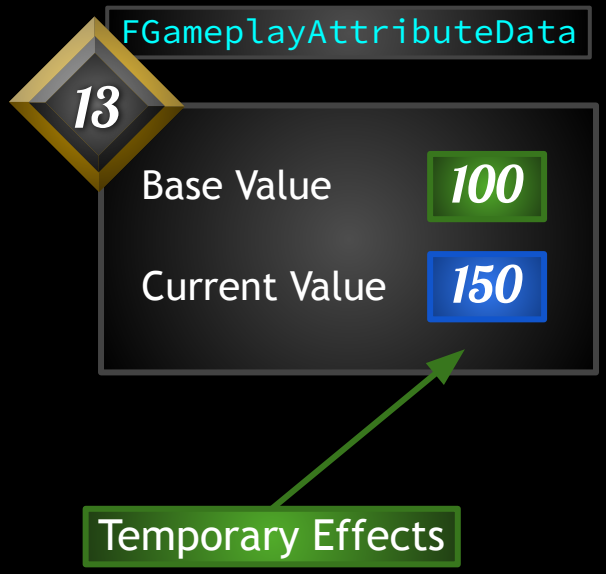


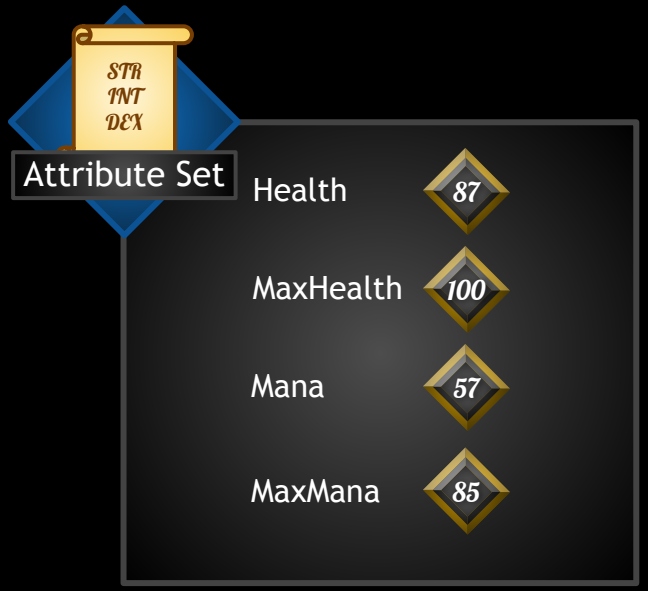
Attributes

With Prediction:



Attributes


FGameplayAttributeData
13
Base Value
100
Current Value
150
Temporary Effects
Buffs/Debuffs


STR
INT
DEX
Attribute Set
Health
87
MaxHealth
100
Mana
57
MaxMana
85


Fraction: Current Value / Base Value ?


Fraction: Health / MaxHealth

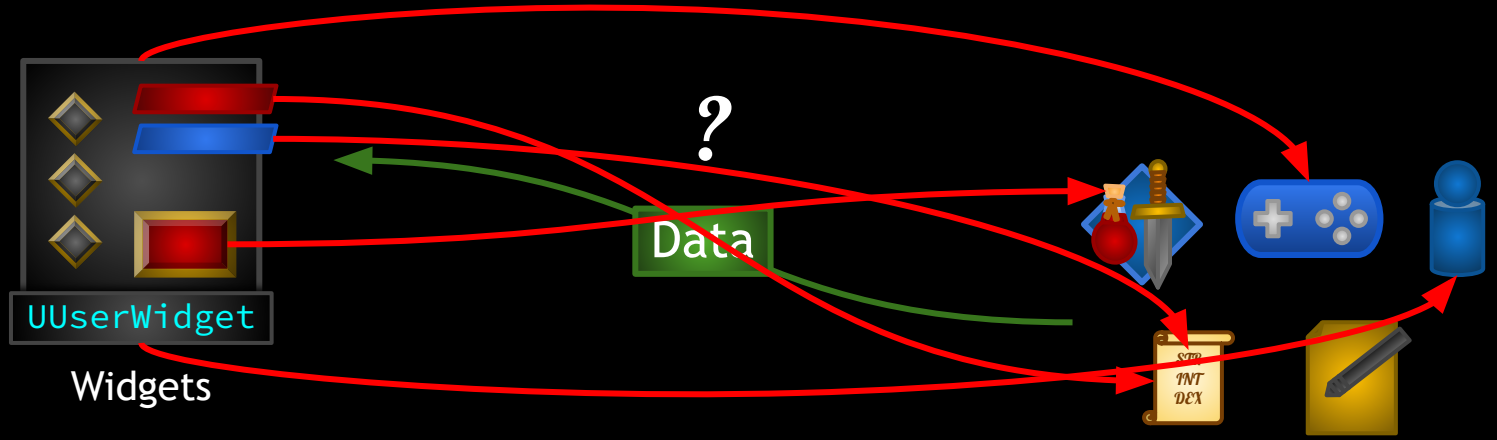
Mixed Replication Mode

For Mixed Replication Mode: The `OwnerActor`'s `Owner` must be the `Controller`. For Pawns, this is set automatically in `PossessedBy()`.

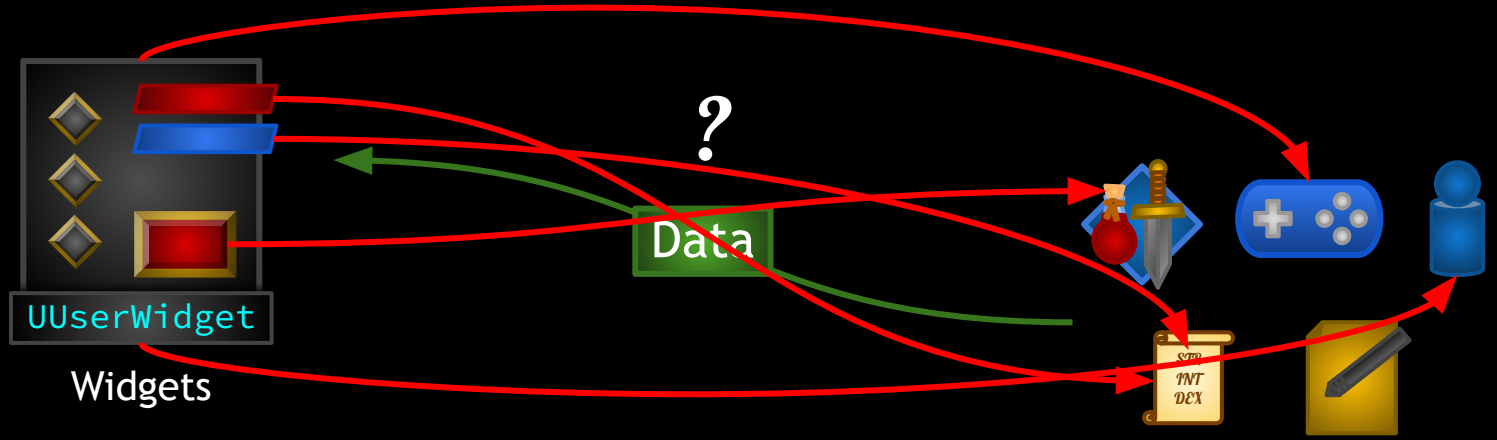
The `PlayerState`'s Owner is automatically set to the `Controller`.

Therefore, if your `OwnerActor` is not the `PlayerState`, and you use **Mixed Replication Mode**, you must call `SetOwner()` on the `OwnerActor` to set its owner to the `Controller.`

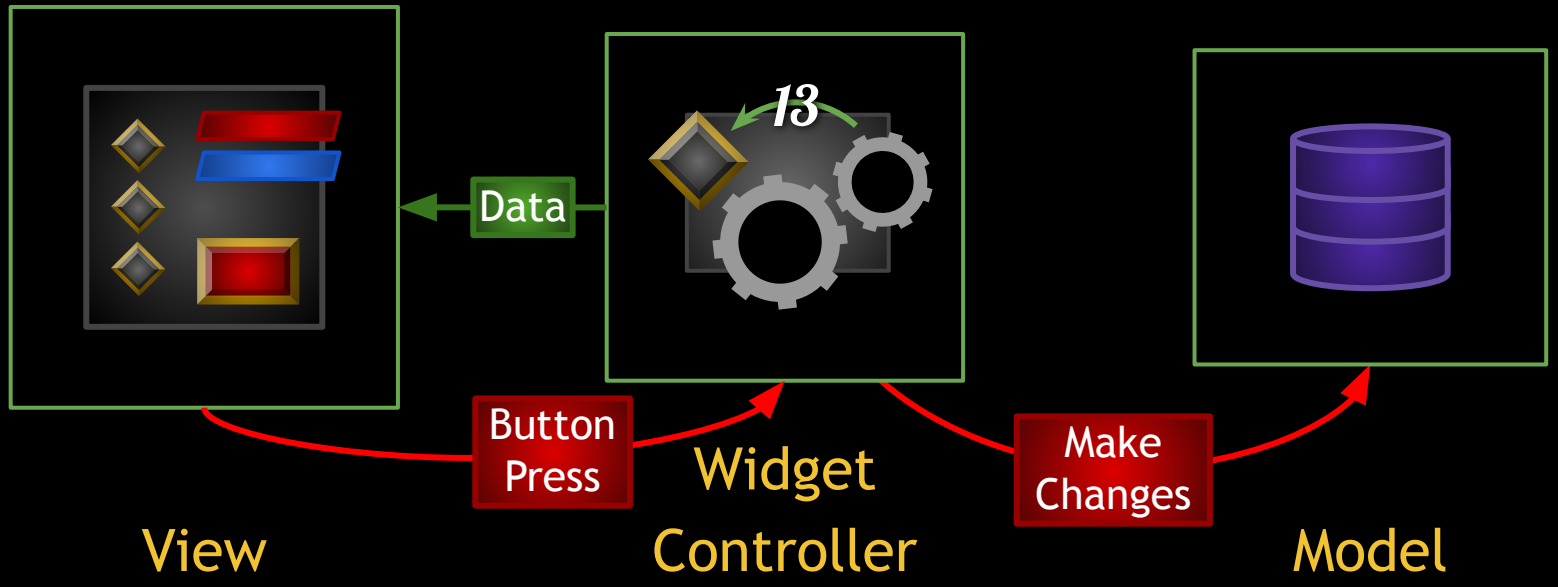
RPG Game UI



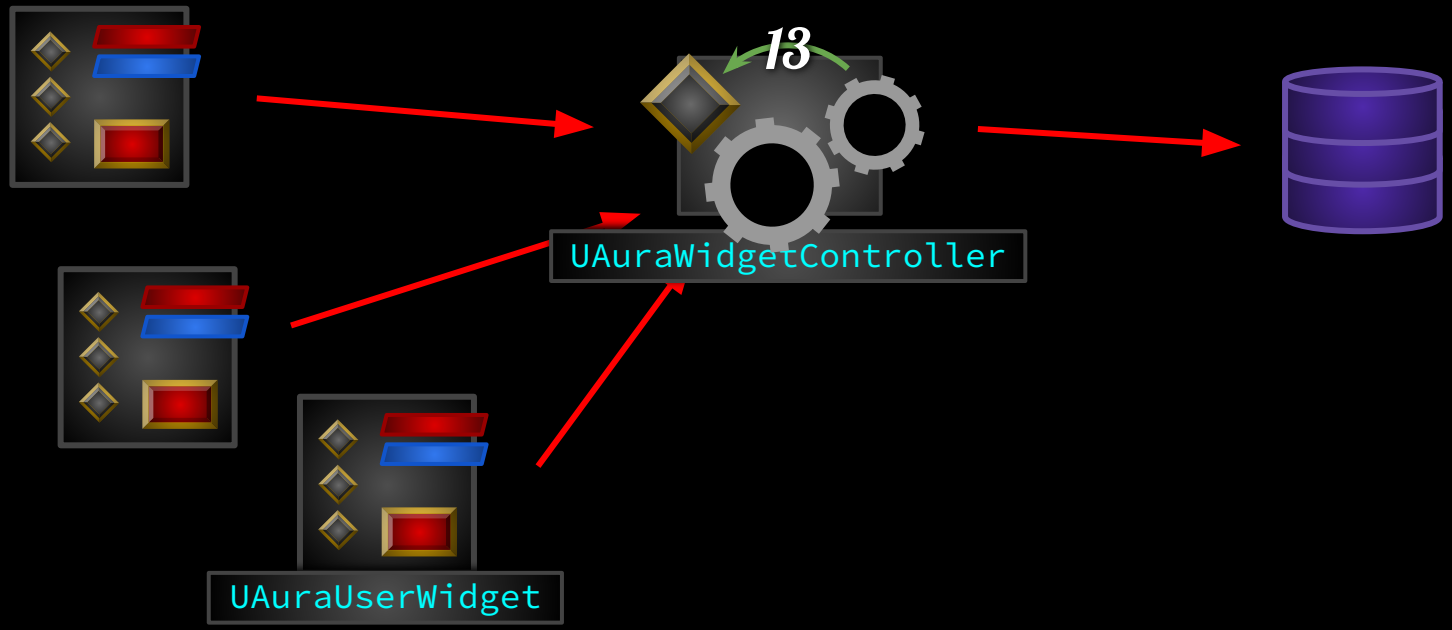
RPG Game UI



RPG Game UI


13
Data
Button Press
Make Changes
View
Widget Controller
Model

RPG Game UI

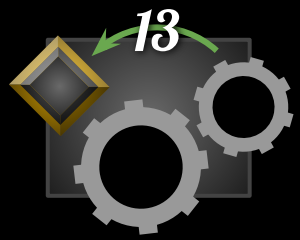


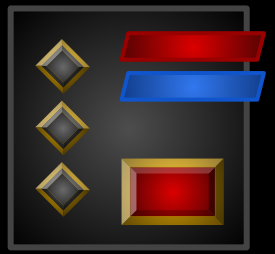
RPG Game UI

Next Steps

1. UAuraUserWidget
2. UAuraWidgetController
3. AAuraHUD
4. Display Health and Mana

RPG Game UI

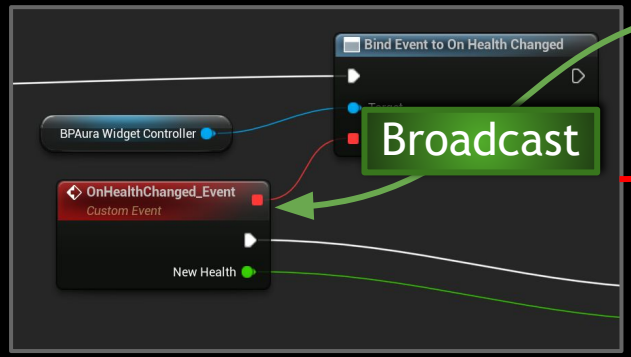




FOnHealthChangedSignature

OnHealthChanged

Bind



FOnGameplayAttributeValueChange

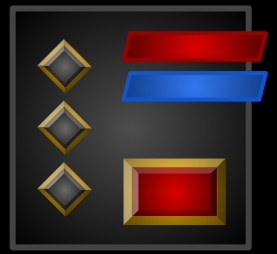
Delegate

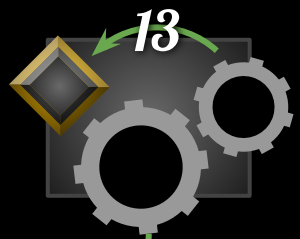
Broadcast

Bind

HealthChanged(const FOnAttributeChanged& Data)

RPG Game UI





Broadcast

Broadcast

```
HealthChanged(const FOnAttributeChanged& Data)
```

Gameplay Effects

- Data only
- Don’t subclass `UGameplayEffect`
- Change Attributes through:
  - Modifiers
  - Executions

Modifier Op

- Add
- Multiply
- Divide
- Override

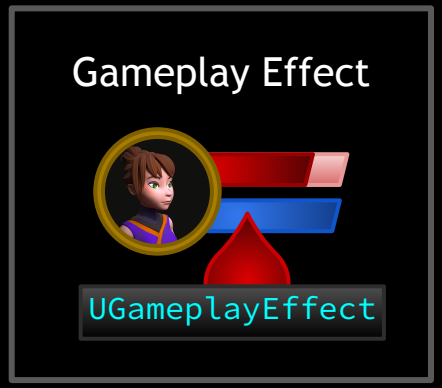
Magnitude Calculation Type

- Scalable Float
- Attribute Based
- Custom Calculation Class (MMC)
- Set by Caller

Executions

- Gameplay Effect Execution Calculation

Gameplay Effects



Duration Policy

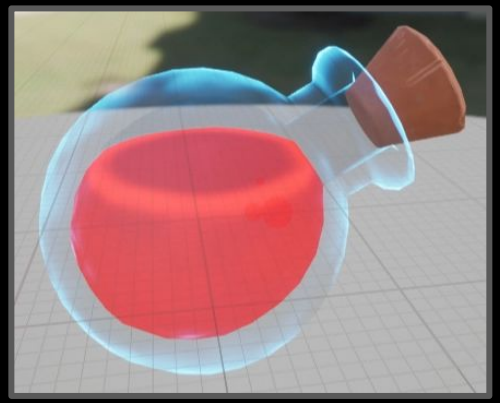
- Instant
- Has Duration
- Infinite

Stacking

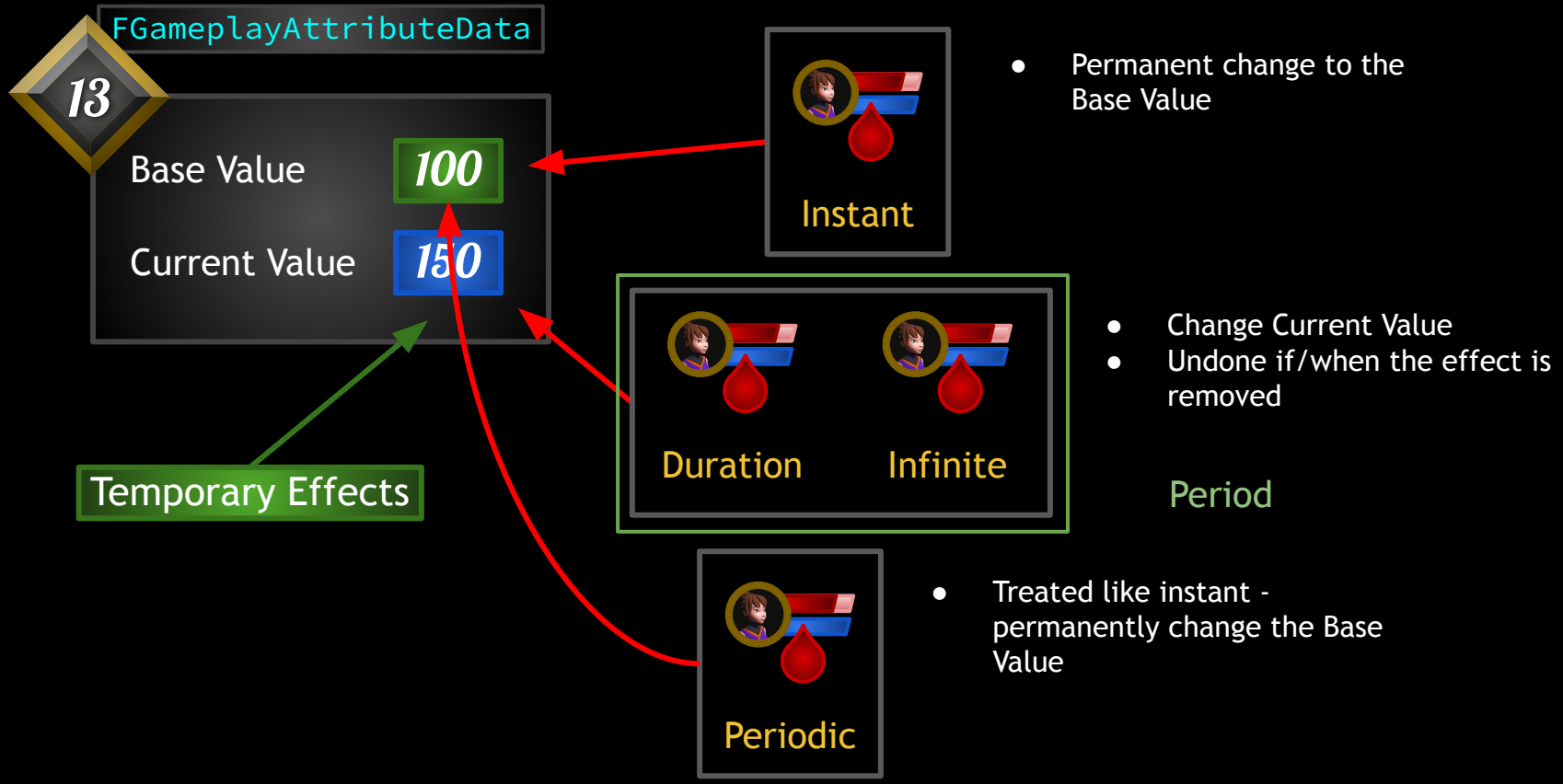
Add Gameplay Tags

Grant Abilities

Gameplay Effect Spec



Periodic Gameplay Effects



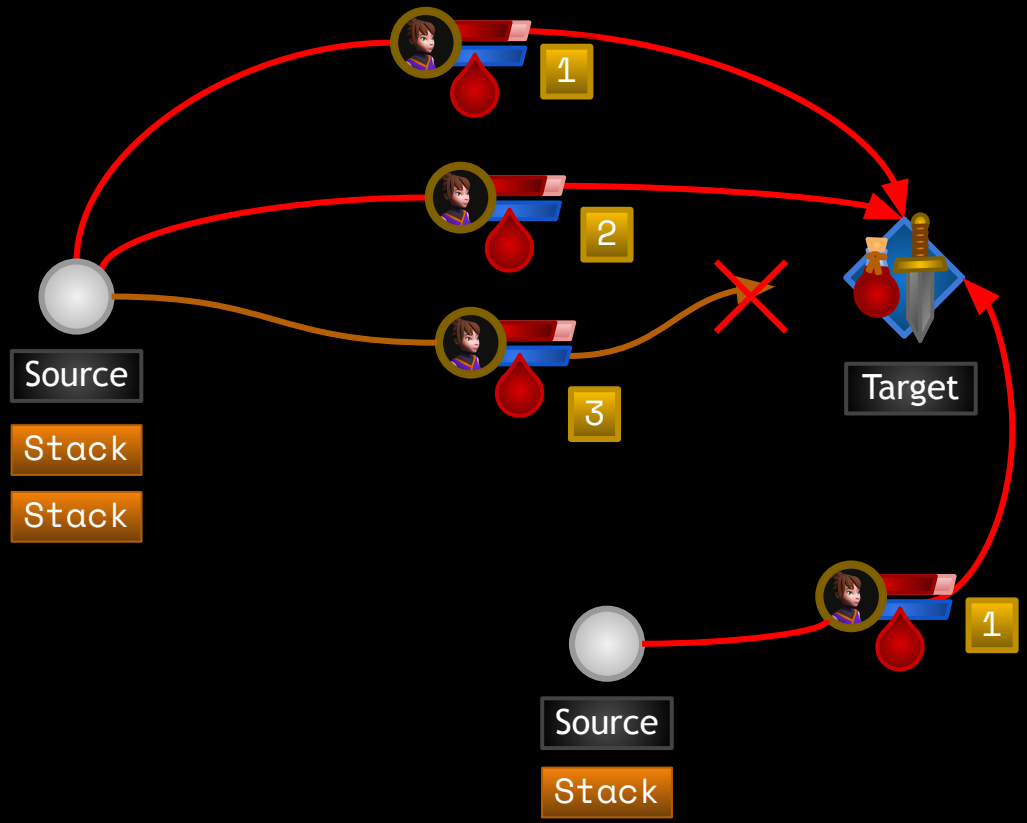
Stacking

Stacking Type

- Aggregate by Source

Stack Limit Count

2



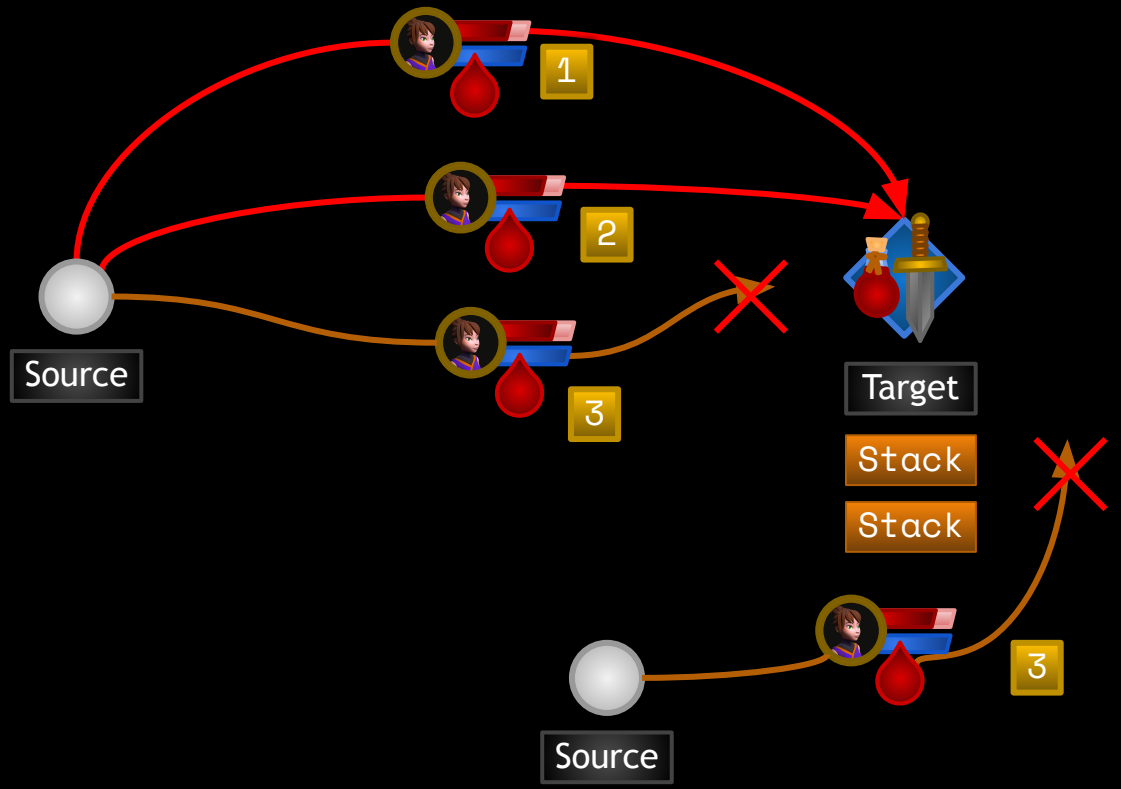
Stacking

Stacking Type

- Aggregate by Target

Stack Limit Count

2



PreAttributeChange

`PreAttributeChange`

- Changes to CurrentValue
  - before the change happens
- Triggered by changes to Attributes
  - Attribute Accessors
  - Gameplay Effects
- Does **not** permanently change the modifier, just the value returned from querying the modifier
- Later operations recalculate the Current Value from all modifiers
  - We need to clamp again

`PostGameplayEffectExecute`

Gameplay Tags

Parent.Child.GrandChild

FGameplayTag

Gameplay Tag

Door

Match?

Shares

Abilities

Abilities.Fire

Abilities.Ice

GameplayCue.Ability.FireBolt.Impact

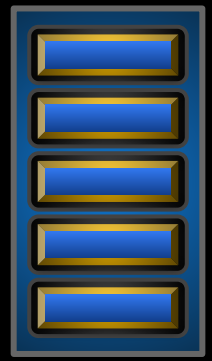
IGameplayTagAssetInterface

GetOwnedGameplayTags

HasMatchingGameplayTag

HasAllMatchingGameplayTags

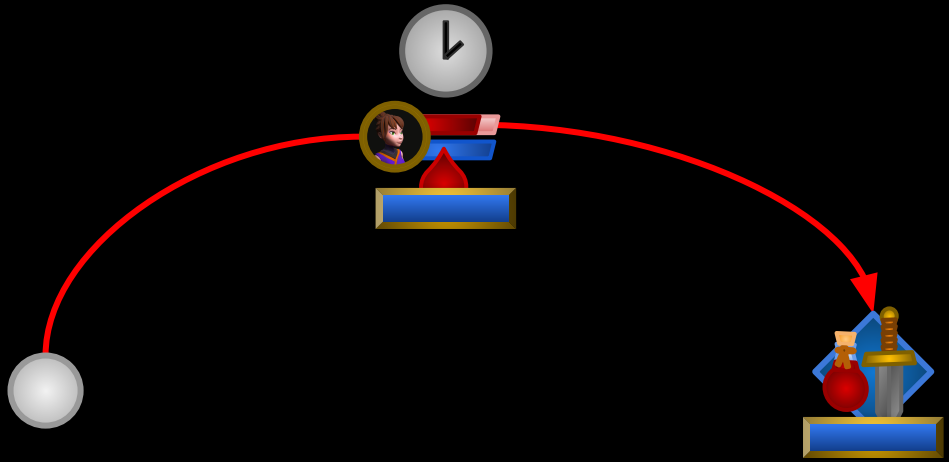
HasAnyMatchingGameplayTags

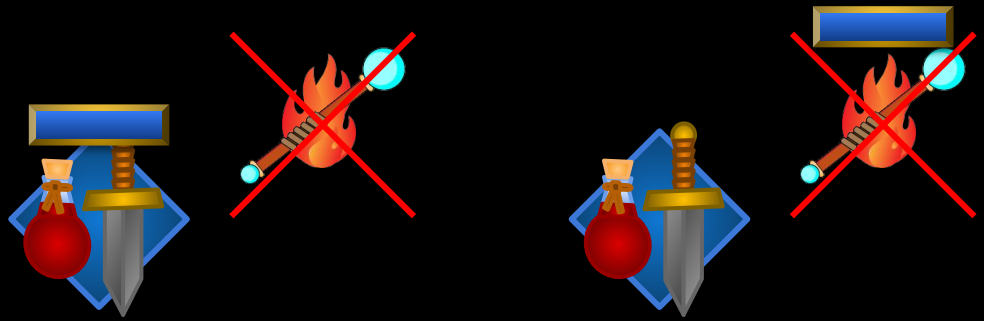


Gameplay Tag Container

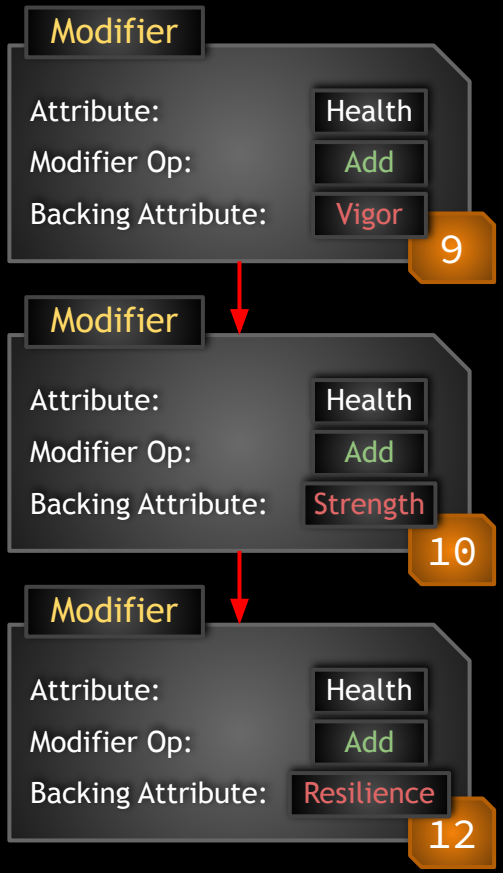
Gameplay Tag Manager

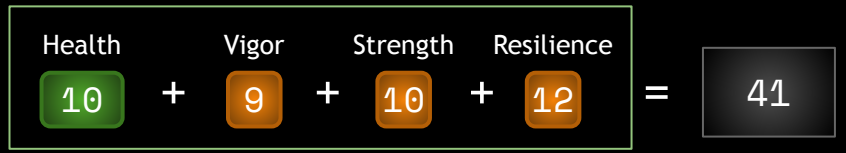
- Inputs
- Abilities
- Attributes
- Damage Types
- Buffs/Debuffs
- Messages
- Data
- Anything you want!



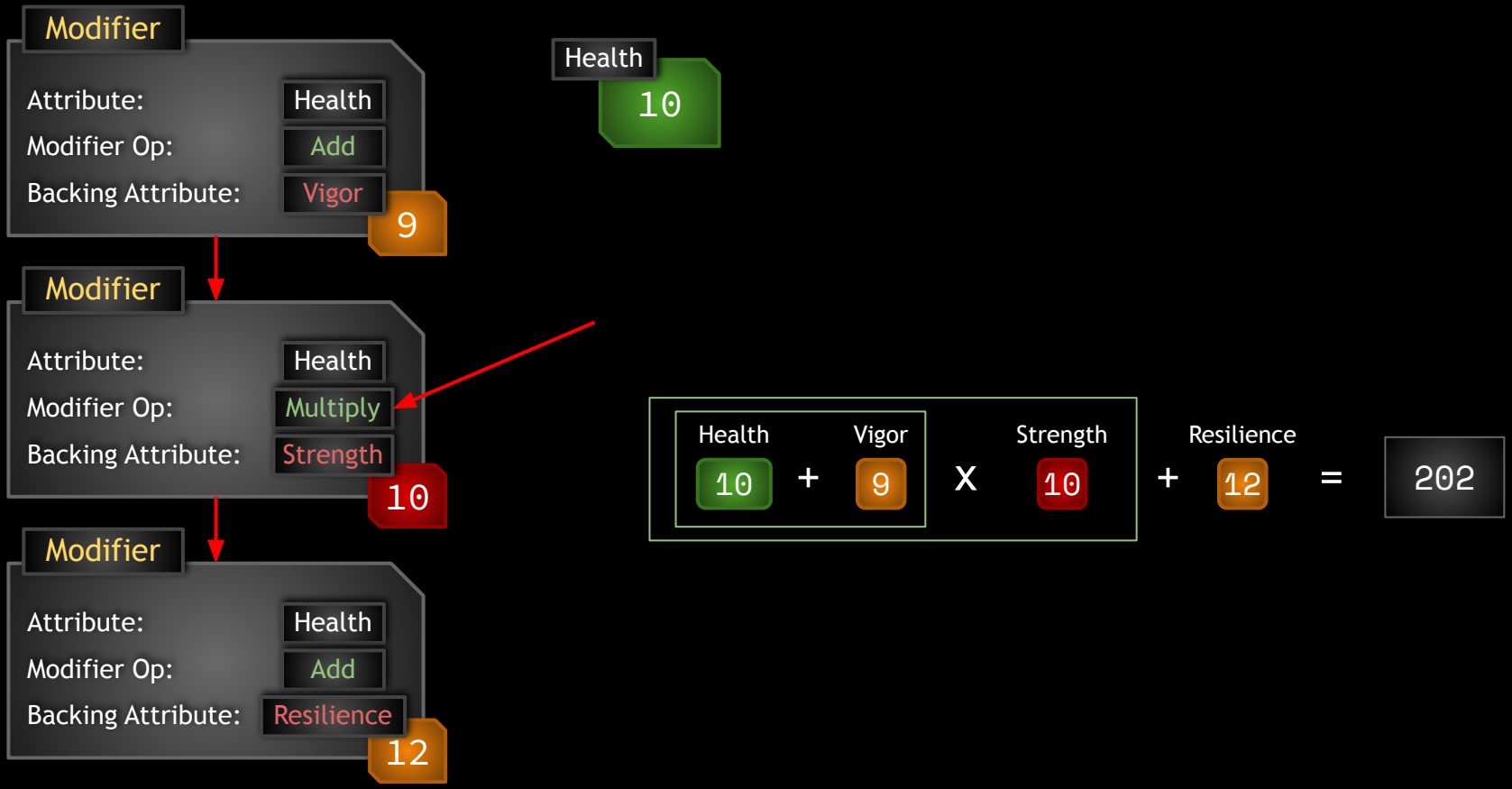


Attribute Based Modifiers

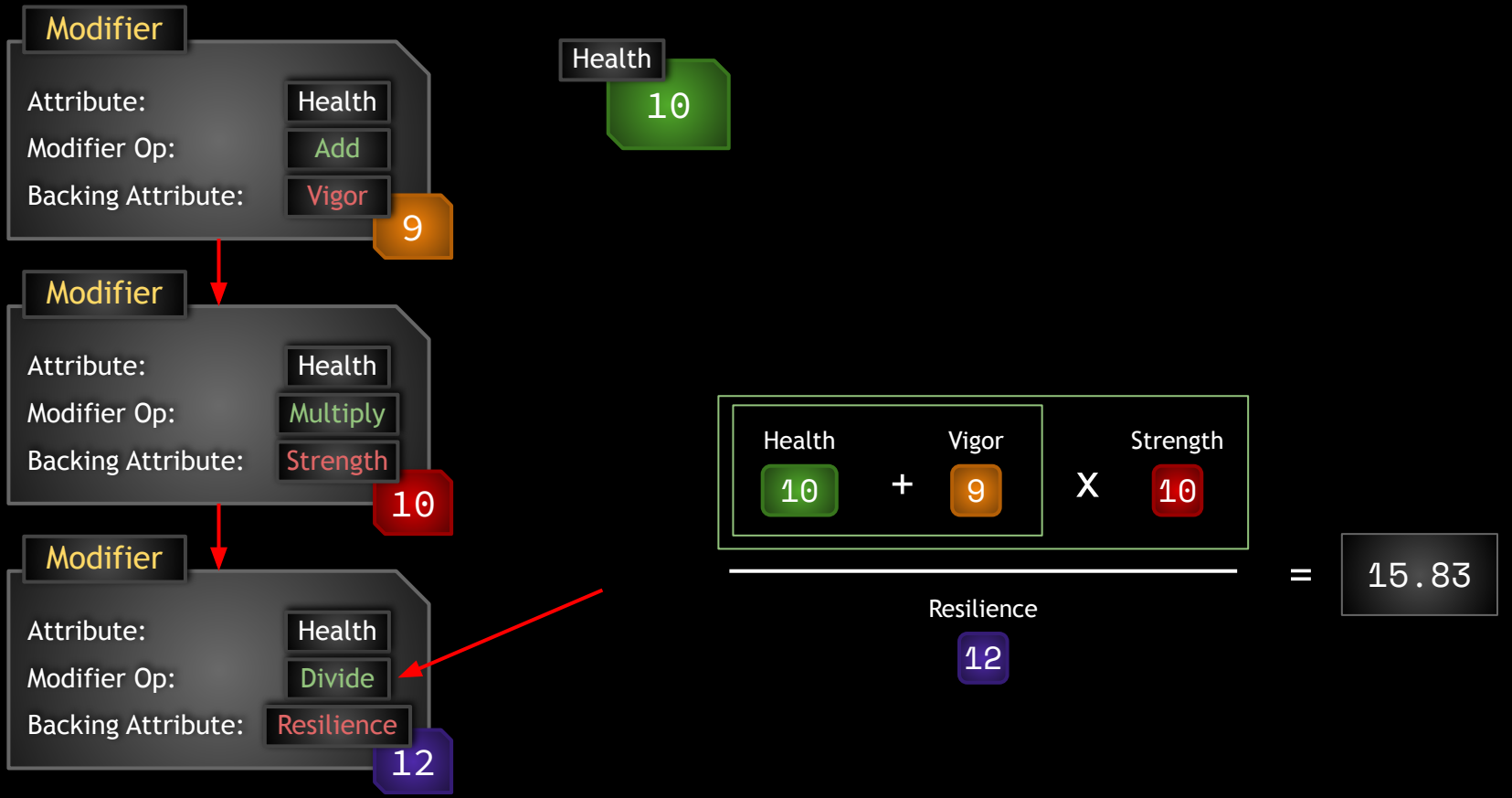




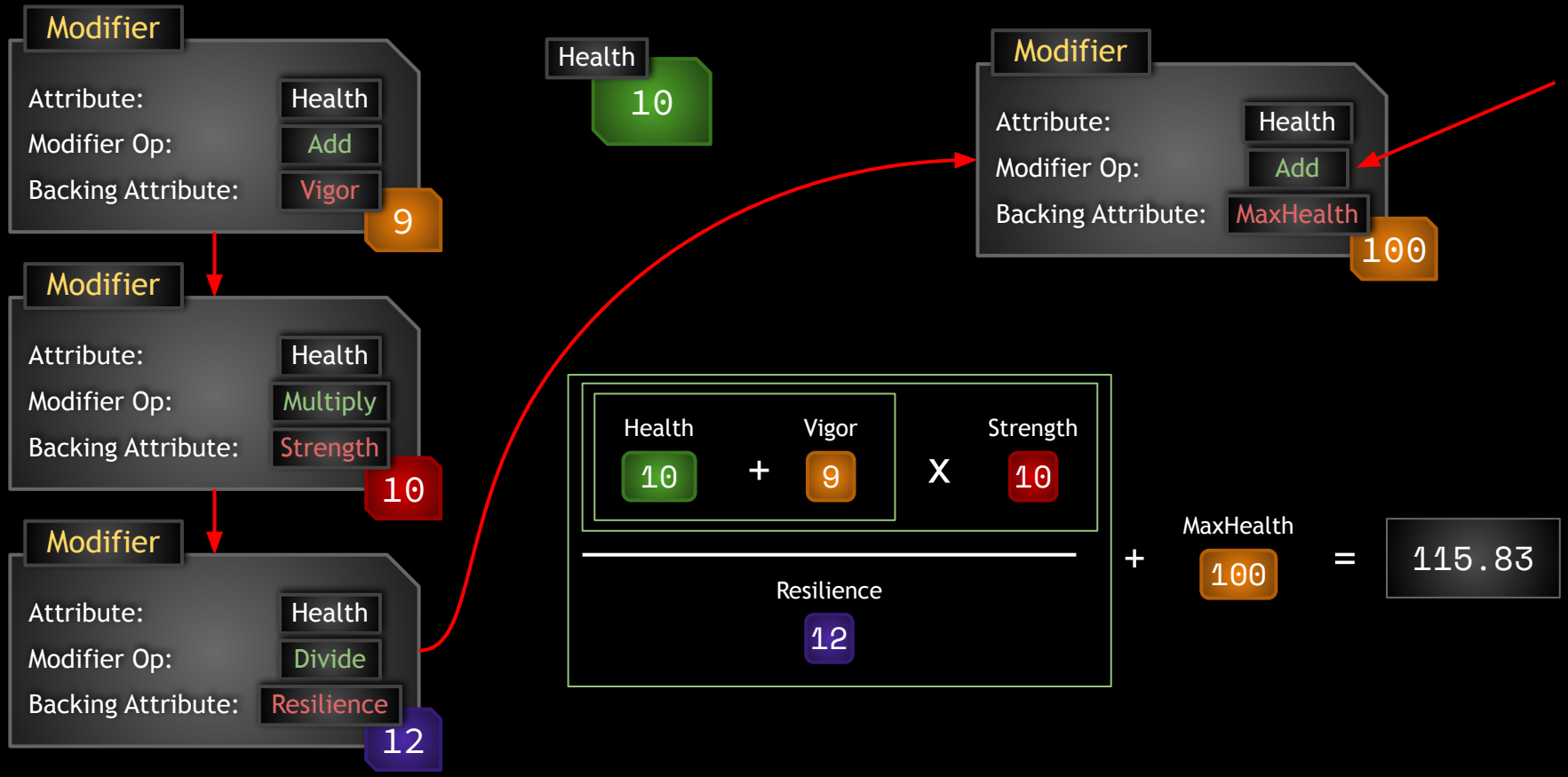
Attribute Based Modifiers



Attribute Based Modifiers



Attribute Based Modifiers



Attribute Based Modifiers

Coefficient

C

Pre Multiply Additive Value

Pre

Post Multiply Additive Value

Post

Modifier

Attribute: Health

Modifier Op: Add

Backing Attribute: Vigor

9

Coefficient C

Pre Multiply Additive Val Pre

Post Multiply Additive Val Post

C x 9 + Pre + Post

Attribute Based Modifiers

Coefficient

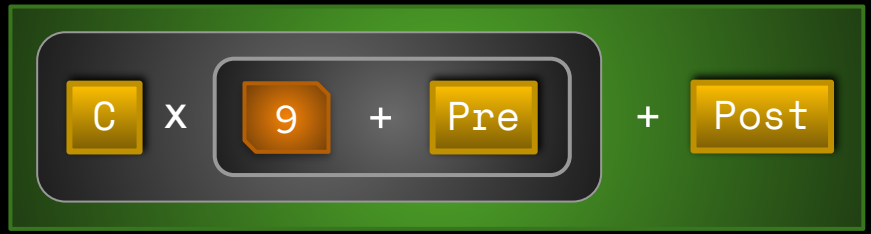
C

Pre Multiply Additive Value

Pre

Post Multiply Additive Value

Post



Modifier

Attribute: Health

Modifier Op: Add

Backing Attribute: Vigor

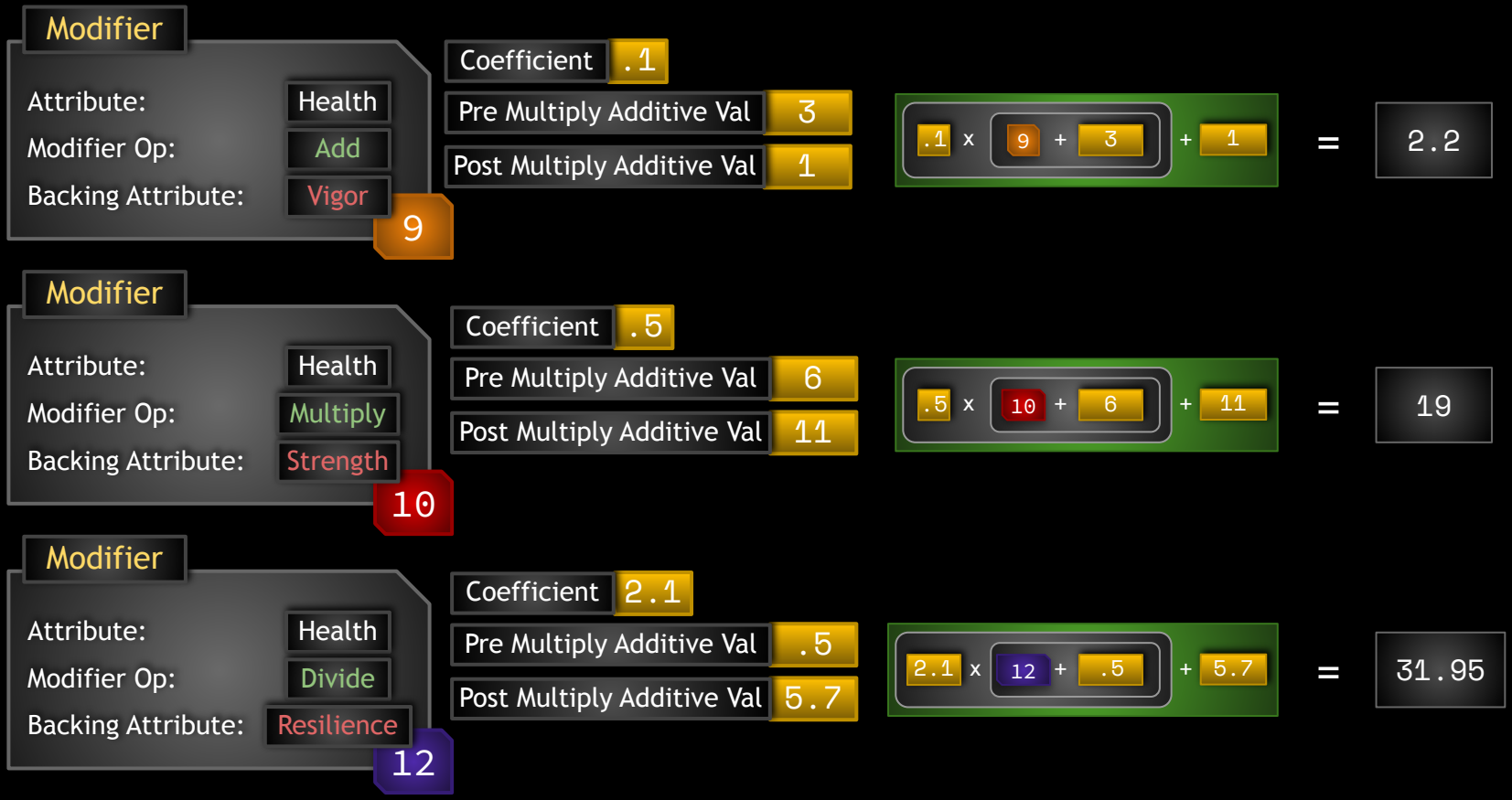
9

Coefficient .1

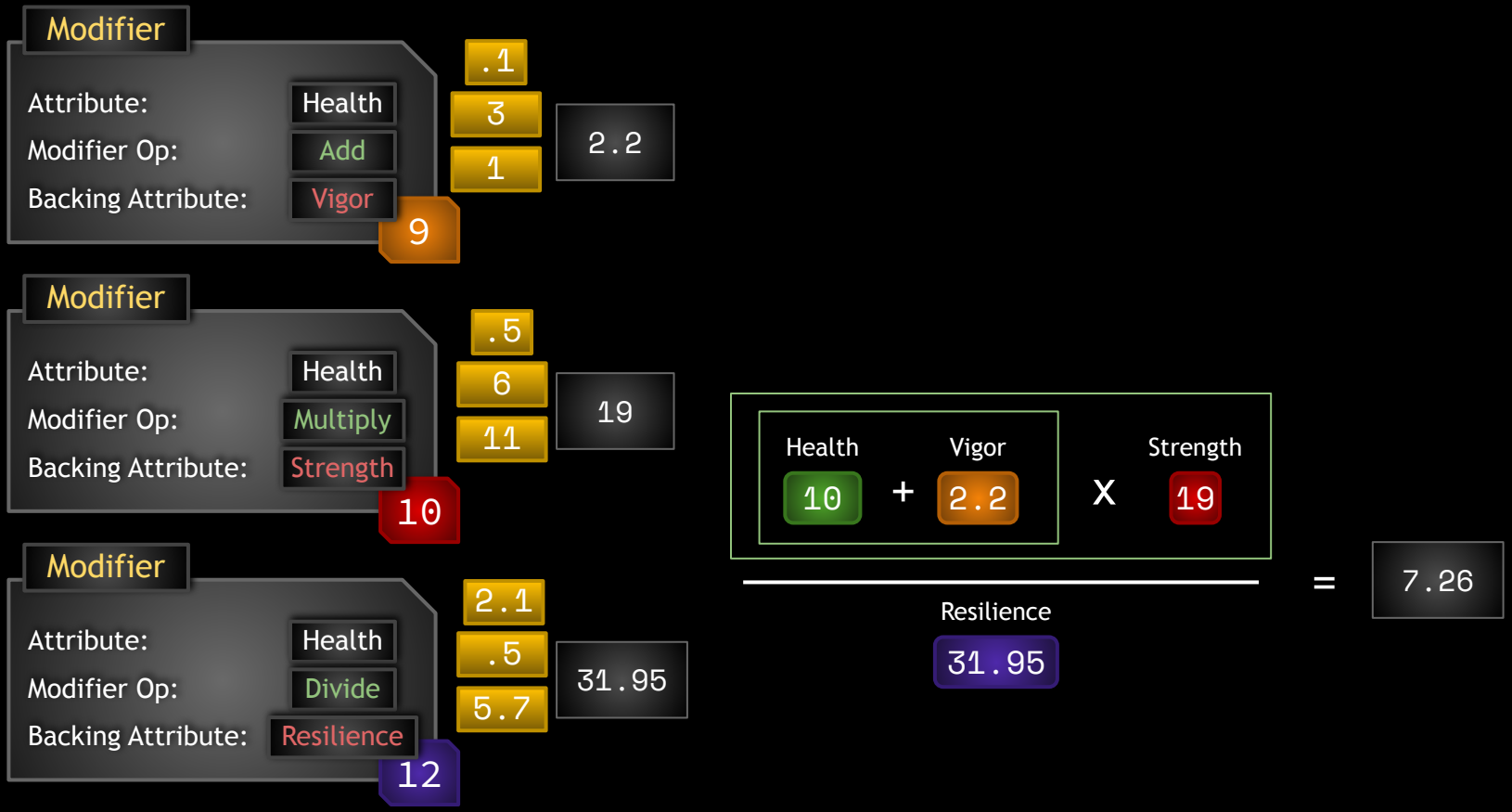
Pre Multiply Additive Val 2

Post Multiply Additive Val 5

Attribute Based Modifiers



Attribute Based Modifiers



Derived Attributes

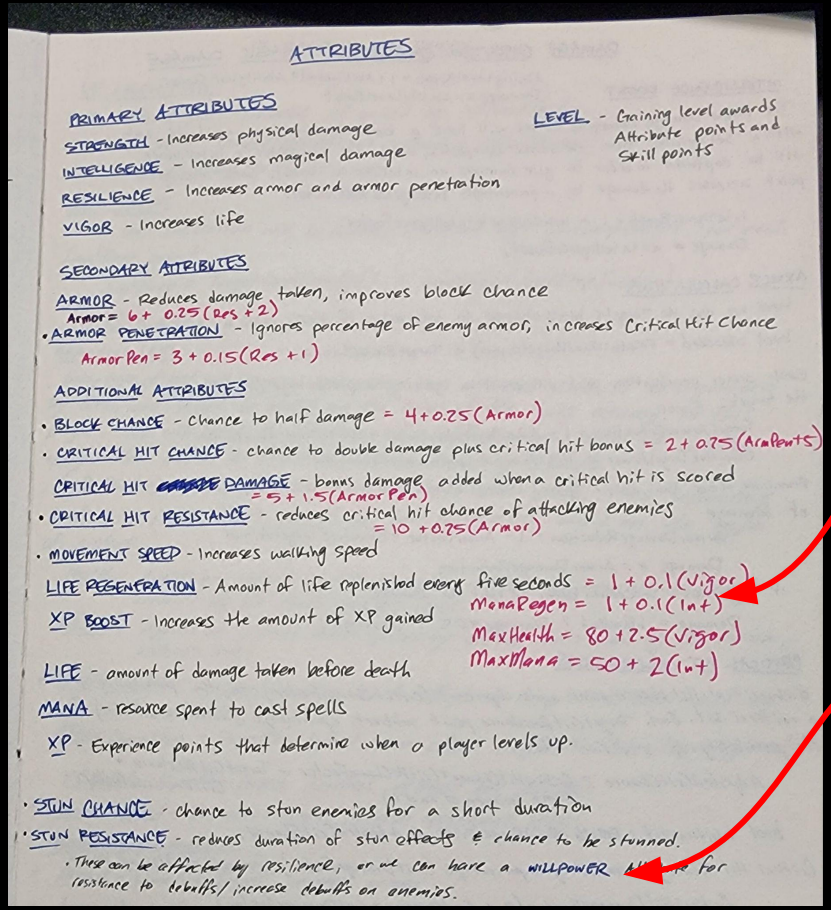
Primary Attributes

Str	Increases physical damage
Int	Increases magical damage
Res	Increases Armor and Armor Penetration
Vig	Increases Health

Secondary Attributes

Resilience	Armor	Reduces damage taken, improves Block Chance
Resilience	Armor Penetration	Ignores percentage of enemy Armor, increases Crit Hit Chance
Armor	Block Chance	Chance to cut incoming damage in half
Armor Penetration	Critical Hit Chance	Chance to double damage plus critical hit bonus
Armor Penetration	Critical Hit Damage	Bonus damage added when a critical hit is scored
Armor	Critical Hit Resistance	Reduces critical hit chance of attacking enemies
Vigor	Health Regeneration	Amount of Health regenerated every 1 second
Intelligence	Mana Regeneration	Amount of Mana regenerated every 1 second
Vigor	Max Health	Maximum amount of Health obtainable
Intelligence	Max Mana	Maximum amount of Mana obtainable

Derived Attributes



ATTRIBUTES

PRIMARY ATTRIBUTES

STRENGTH - Increases physical damage

INTELLIGENCE - Increases magical damage

RESILIENCE - Increases armor and armor penetration

VIGOR - Increases life

LEVEL - Gaining level awards Attribute points and Skill points

SECONDARY ATTRIBUTES

ARMOR - Reduces damage taken, improves block chance

Armor = 6 + 0.25(Res + 2)

- ARMOR PENETRATION - Ignores percentage of enemy armor, increases Critical Hit Chance

ArmorPen = 3 + 0.15(Res + 1)

ADDITIONAL ATTRIBUTES

- BLOCK CHANCE - Chance to half damage = 4 + 0.25(Armor)
- CRITICAL HIT CHANCE - chance to double damage plus critical hit bonus = 2 + 0.75(ArmPents)

CRITICAL HIT ~~CHANCE~~ DAMAGE - bonus damage added when a critical hit is scored = 5 + 1.5(ArmorPen)

- CRITICAL HIT RESISTANCE - reduces critical hit chance of attacking enemies = 10 + 0.75(Armor)
- MOVEMENT SPEED - Increases walking speed

LIFE REGENERATION - Amount of life replenished every five seconds = 1 + 0.1(Vigor)

XP BOOST - Increases the amount of XP gained

ManaRegen = 1 + 0.1(Int)

MaxHealth = 80 + 2.5(Vigor)

MaxMana = 50 + 2(Int)

LIFE - amount of damage taken before death

MANA - resource spent to cast spells

XP - Experience points that determine when a player levels up.

- STUN CHANCE - chance to stun enemies for a short duration
- STUN RESISTANCE - reduces duration of stun effects & chance to be stunned.
  - These can be affected by resilience, or we can have a WILLPOWER [illegible] for resistance to debuffs/increase debuffs on enemies.

- Brainstorming
- Messy
- Estimations

- Speculation/ideas

Derived Attributes

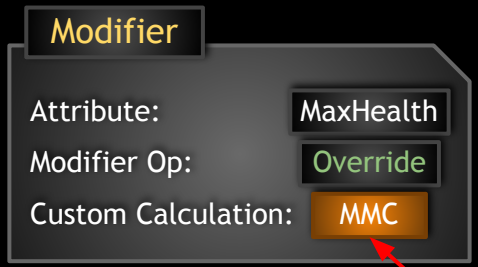
Primary Attributes

- **Str** Increases physical damage
- **Int** Increases magical damage
- **Res** Increases Armor and Armor Penetration
- **Vig** Increases Health

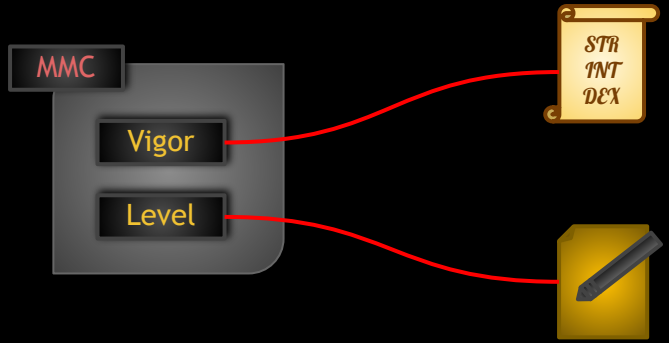
Secondary Attributes

Resilience		Armor
Resilience		Armor Penetration
Armor		Block Chance
Armor Penetration		Critical Hit Chance
Armor Penetration		Critical Hit Damage
Armor		Critical Hit Resistance
Vigor		Health Regeneration
Intelligence		Mana Regeneration
Vigor	Level	Max Health
Intelligence	Level	Max Mana

Custom Calculation



Modifier Magnitude Calculation (MMC)



MaxHealth = 80 + 2.5 x Vigor + 10 x Level

ICombatInterface

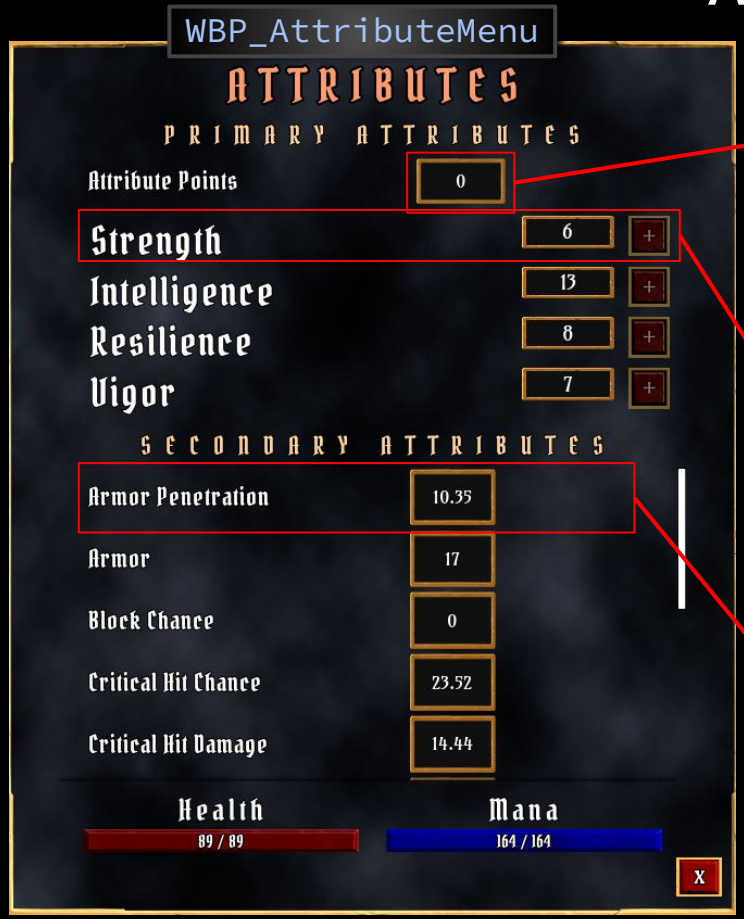
GetPlayerLevel

Custom Calculation

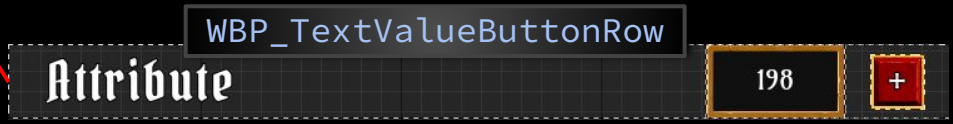
Next Steps

1. Add a Level variable to the Player State
2. Create a Combat Interface
3. Create MMCs for MaxHealth and MaxMana
    a. Make these dependent on Level as well as their backing Attributes

Attribute Menu



Framed Value



Text Value Row with Button

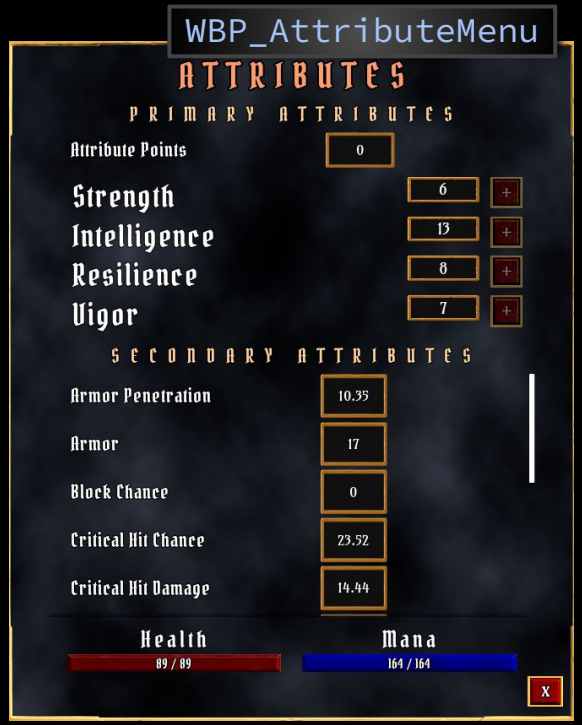
Attribute Row

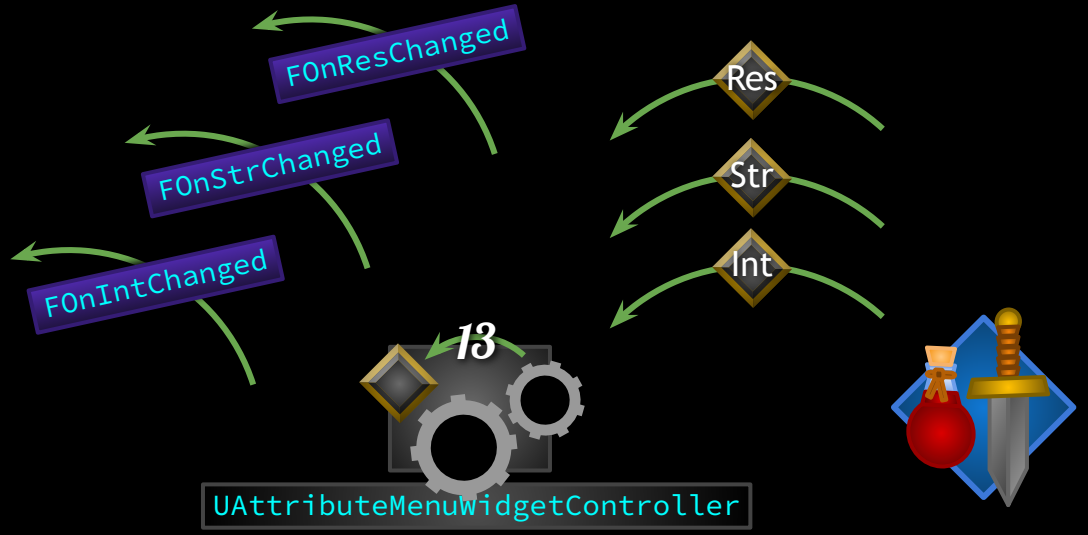
Attribute Menu

Next Steps

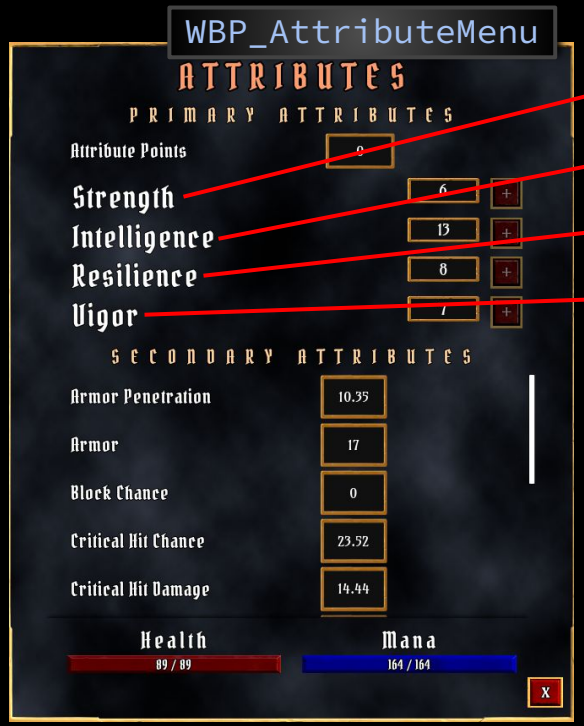
1. Create WBP_FramedValue widget
2. Create WBP_TextValueRow widget
3. Create WBP_TextValueButtonRow widget
4. Create WBP_AttributeMenu widget

Attribute Menu





Attribute Menu



Attributes.Primary.Strength

Attributes.Primary.Intelligence

Attributes.Primary.Resilience

Attributes.Primary.Vigor

FOnAttributeChanged

Attributes.Primary.Strength

Name

80

Description

Attributes.Primary.Strength

Name

80

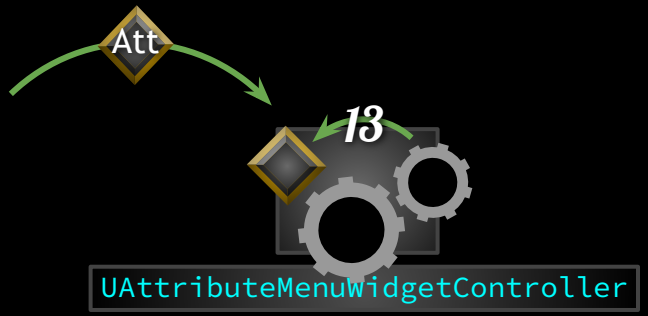
Description

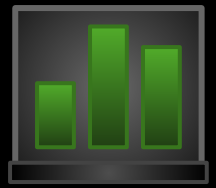
FAuraAttributeInfo

13

UAttributeMenuWidgetController

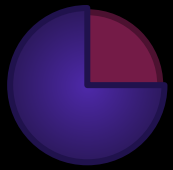
Attribute Menu



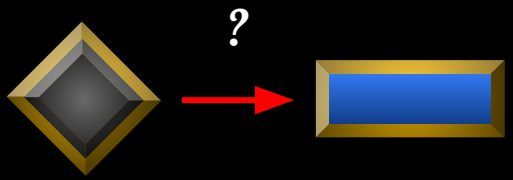


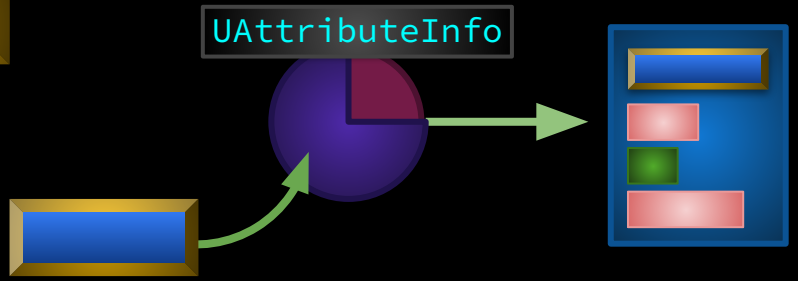
Data Table

or

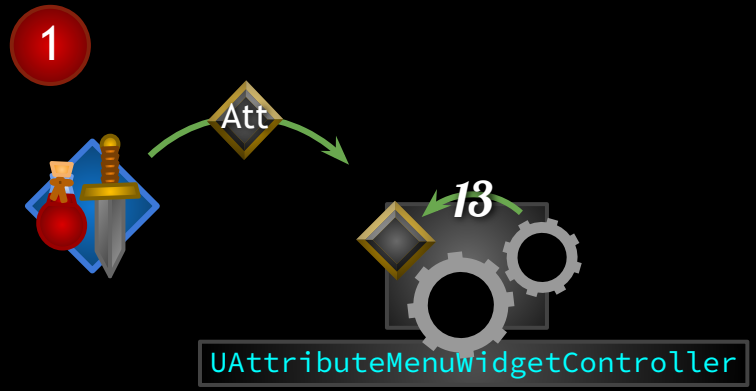


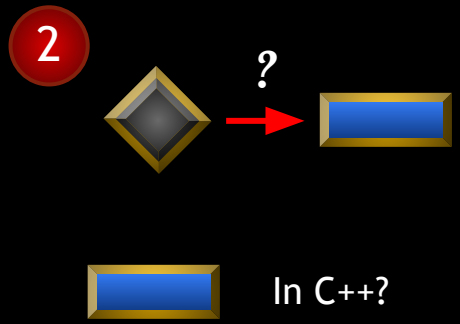
Data Asset

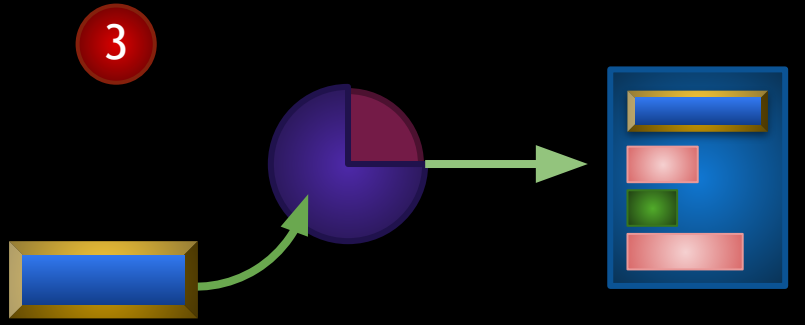


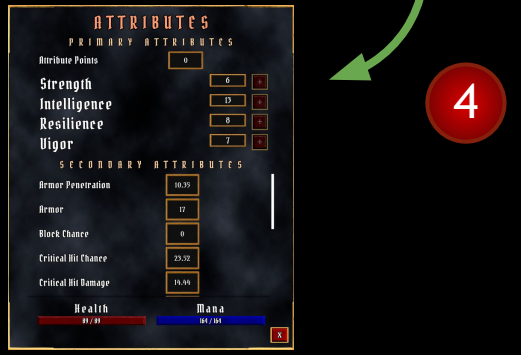


Attribute Menu









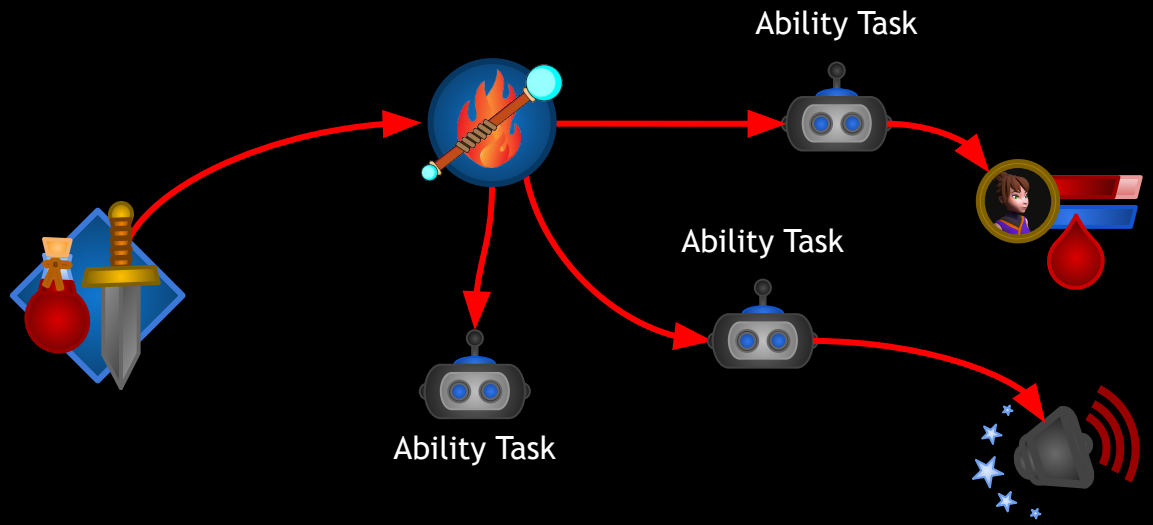
Attribute Menu

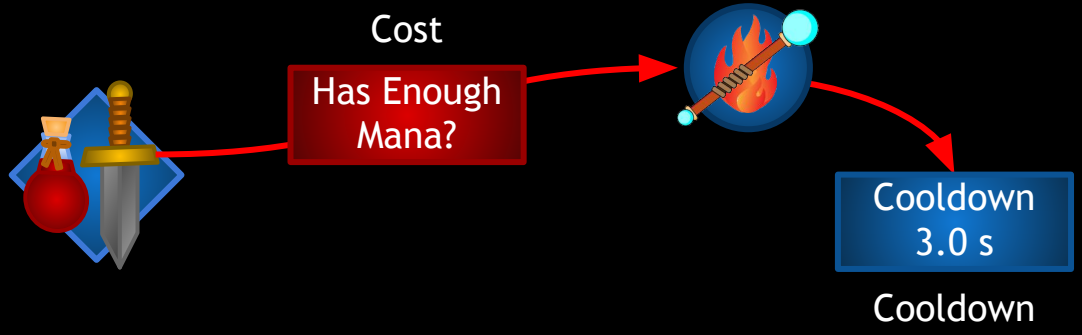
Next Steps

1. Create Secondary Attribute Gameplay Tags (handle tags better in C++)
2. Create UAttributeInfo Data Asset
3. Create FAuraAttributeInfo struct
4. Fill in the Data Asset for each Attribute
5. Create UAttributeMenuWidgetController

Gameplay Abilities

Gameplay Ability

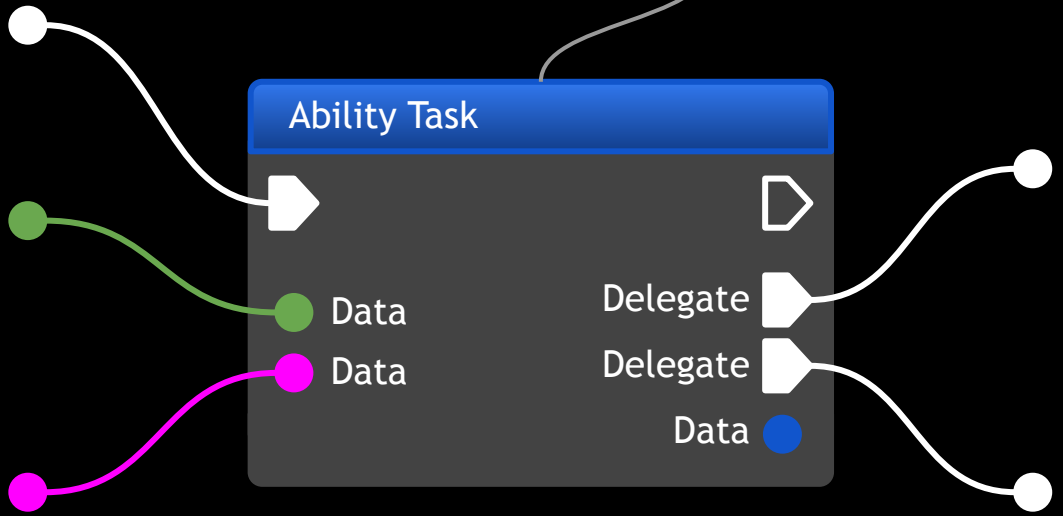




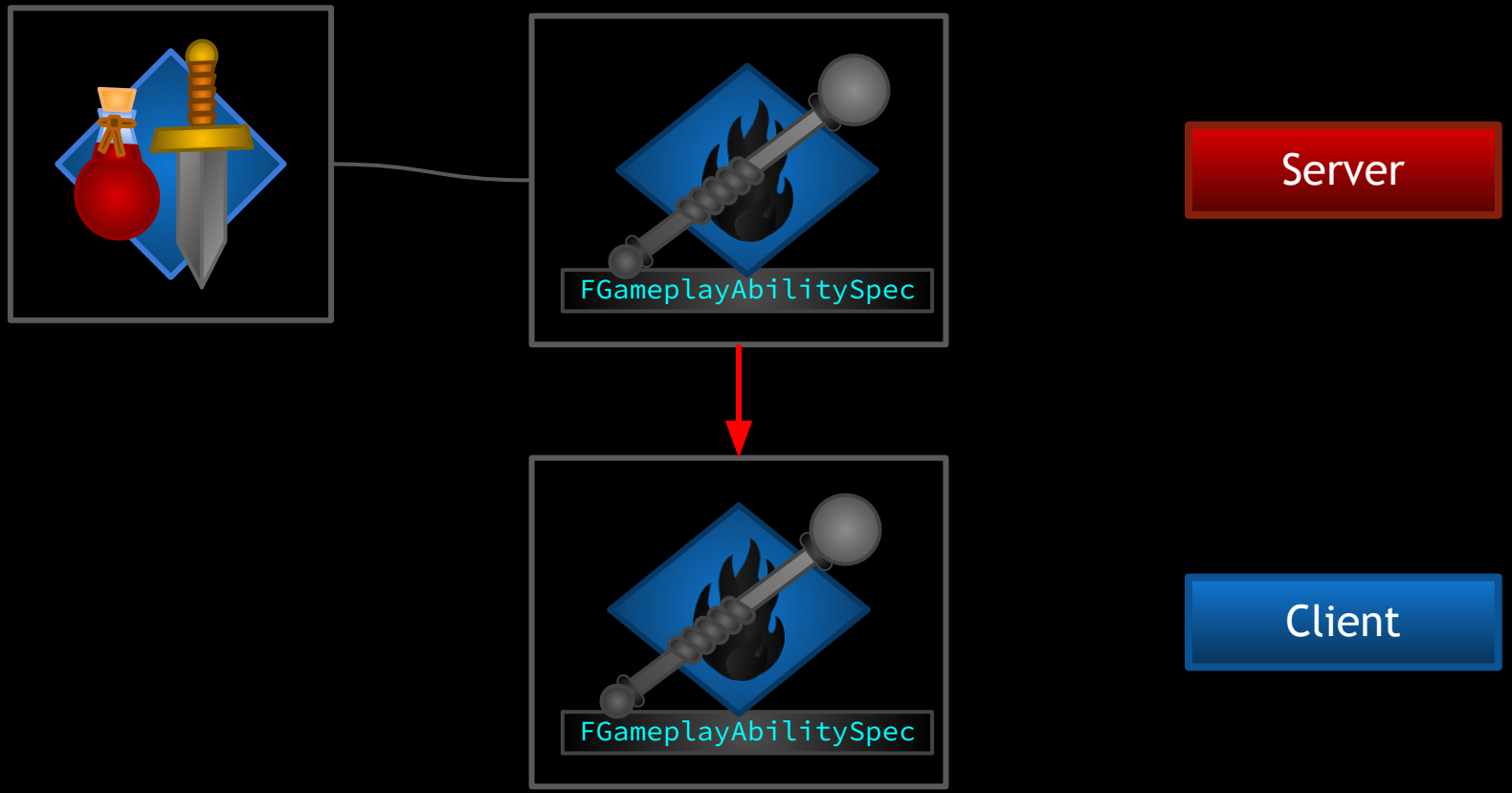
Ability Tasks

Ability Tasks

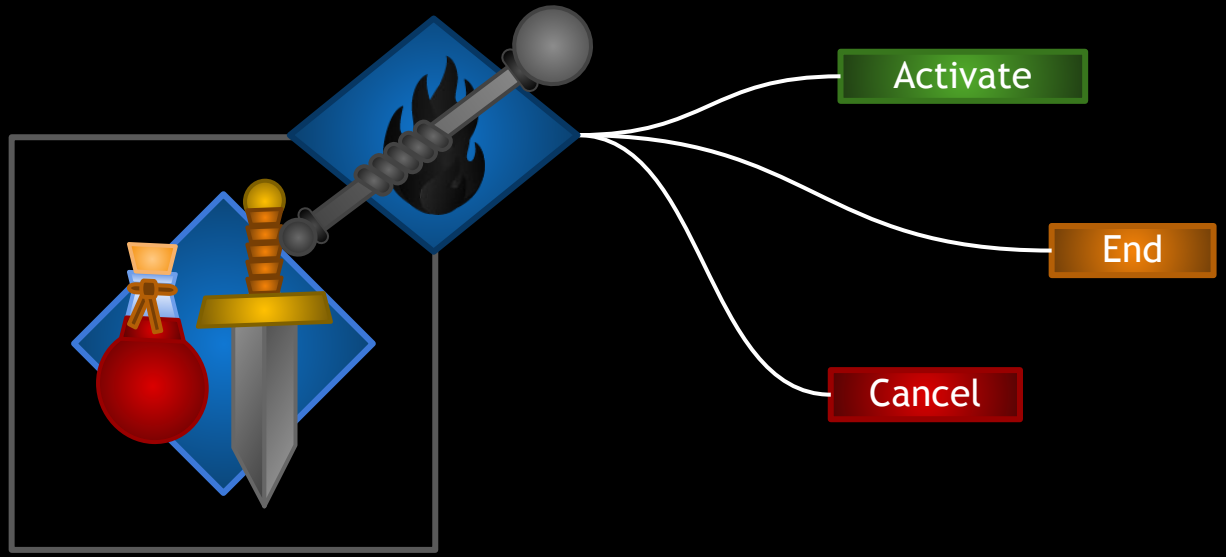

```
Activate()
{
  // Do great stuff…
}
```



Granting Abilities



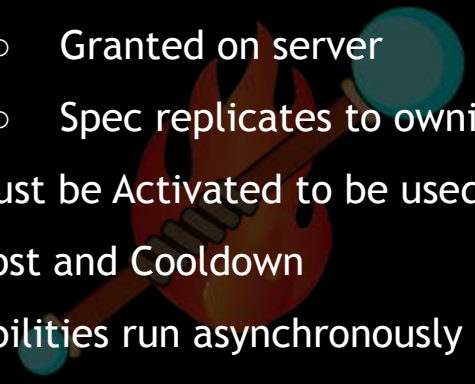
Granting Abilities



Summary

Gameplay Abilities

- Classes that define a skill or ability
- Must be granted
  - Granted on server
  - Spec replicates to owning client
- Must be Activated to be used
- Cost and Cooldown
- Abilities run asynchronously
  - Multiple active at a time
- Ability Tasks
  - Asynchronously perform operations



Tags

Gameplay Tag	Description
Ability Tags	*This ability has these tags*
Cancel Abilities with Tag	*Abilities with these tags are cancelled when this ability is executed*
Block Abilities with Tag	*Abilities with these tags are blocked while this ability is active*
Activation Owned Tags	*Tags to apply to activating owner while this ability is active. These are replicated if ReplicateActivationOwnedTags is enabled in AbilitySystemGlobals*
Activation Required Tags	*This ability can only be activated if the activating actor/component has all of these tags*
Activation Blocked Tags	*This ability is blocked if the activating actor/component has any of these tags*
Source Required Tags	*This ability can only be activated if the source actor/component has all of these tags*
Source Blocked Tags	*This ability is blocked if the source actor/component has any of these tags*
Target Required Tags	*This ability can only be activated if the target actor/component has all of these tags*
Target Blocked Tags	*This ability is blocked if the target actor/component has any of these tags*

Instancing Policy

Instancing Policy	Description	Details
`Instanced Per Actor`	A single instance is created for the ability. It is reused with each activation.	Can store persistent data. Variables must be manually reset each time.
`Instanced Per Execution`	New instance created with each activation	Does not store persistent data between activations. Less performant than Instanced per Actor
`Non-Instanced`	Only the Class Default Object is used, no instances are created.	Cannot store state, cannot bind to delegates on Ability Tasks. Best performance of the three options.

Net Execution Policy

Net Execution Policy	Description
`Local Only`	Only run on the local client. Server does not run the ability.
`Local Predicted`	Activate on the local client, and then on the Server. Makes use of prediction. Server can roll back invalid changes.
`Server Only`	Only run on the Server.
`Server Initiated`	Run on the Server first, then on the owning local client.

Things Not to Use:

Replication Policy

- Useless. Don’t use it. Refer to Epic’s Ability System Questions for an explanation from Epic.
- Gameplay Abilities are replicated from Server to owning Client already.
    - Note: Gameplay Abilities don’t run on Simulated Proxies (use GEs and GCs)

Server Respects Remote Ability Cancellation

- Means when the local Client’s ability ends, the server’s will end
    - Not typically a good idea; it’s the Server’s version that matters

Replicate Input Directly

- Always replicates input press/release events to the Server.
    - Epic discourages it

```
/** Direct Input state replication. These will be called if bReplicateInputDirectly is true on the ability and
is generally not a good thing to use. (Instead, prefer to use Generic Replicated Events). */
UAbilitySystemComponent::ServerSetInputPressed()
```

Input

Binding Input to the Ability System Component

- This was an option before Enhanced Input
- Inputs were bound directly to Abilities
- Enum with Ability Input constants
    - Rigid

Enhanced Input

- Input Actions are bound to inputs via the Input Mapping Context.
- We can decide how to activate abilities in response to inputs.
    - Lyra provides one example
    - We’ll use a similar approach (though less complicated)
- Data Driven
    - Change Input-to-Ability mappings at runtime

Click To Move

Top Down Template

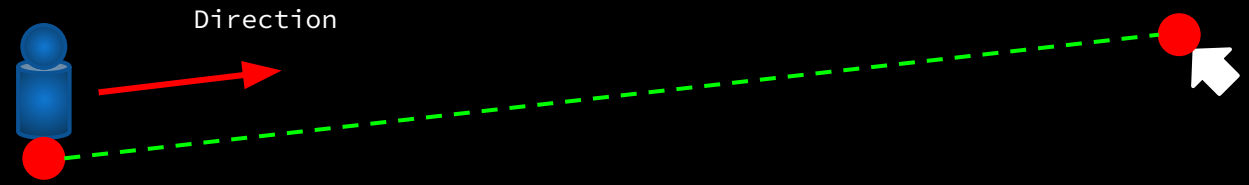
- Uses `SimpleMoveToLocation` if it was a short press
  - Does not work in multiplayer (only AI controlled on server)
- Uses `AddMovementInput` if input is held down
  - Works in multiplayer
  - Requires constant input (movement direction)

Our GAS Project

- We must use `AddMovementInput`
- Need a direction each frame

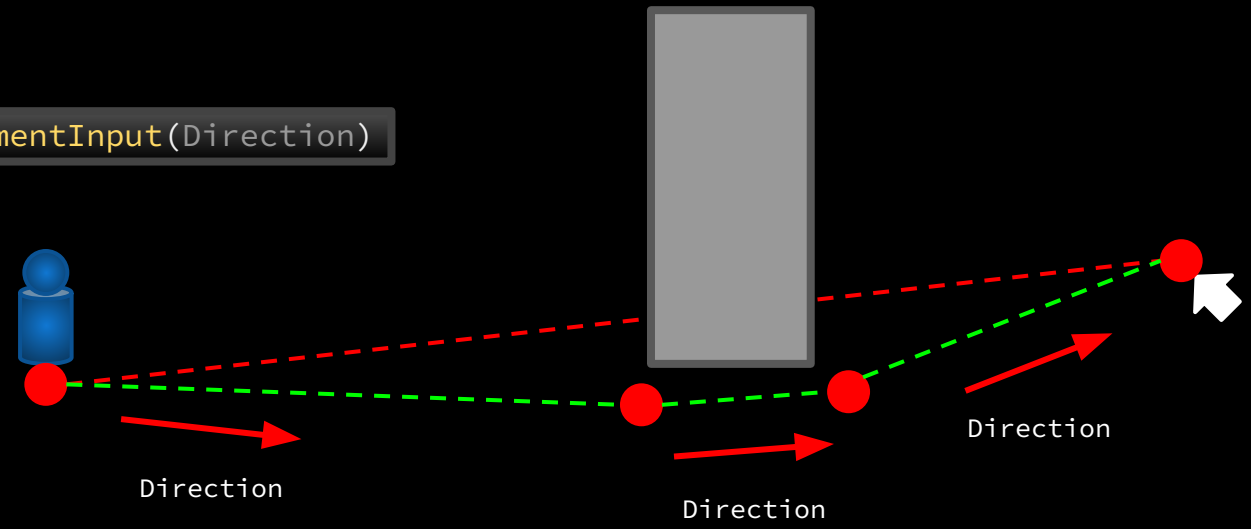
Click To Move

```
AddMovementInput(Direction)
```

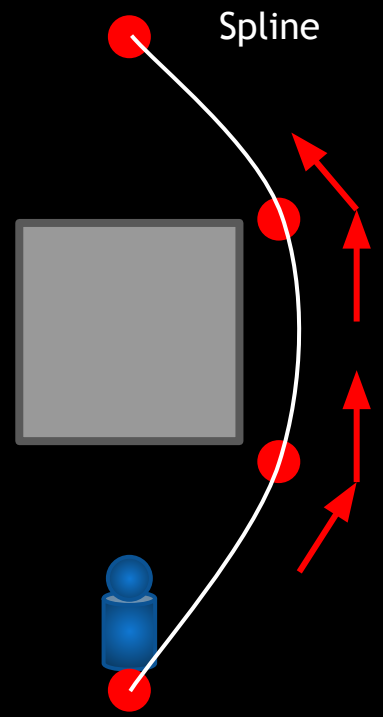


Click To Move

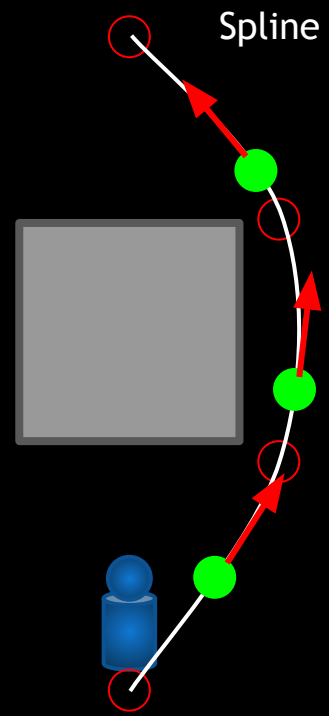
```
AddMovementInput(Direction)
```



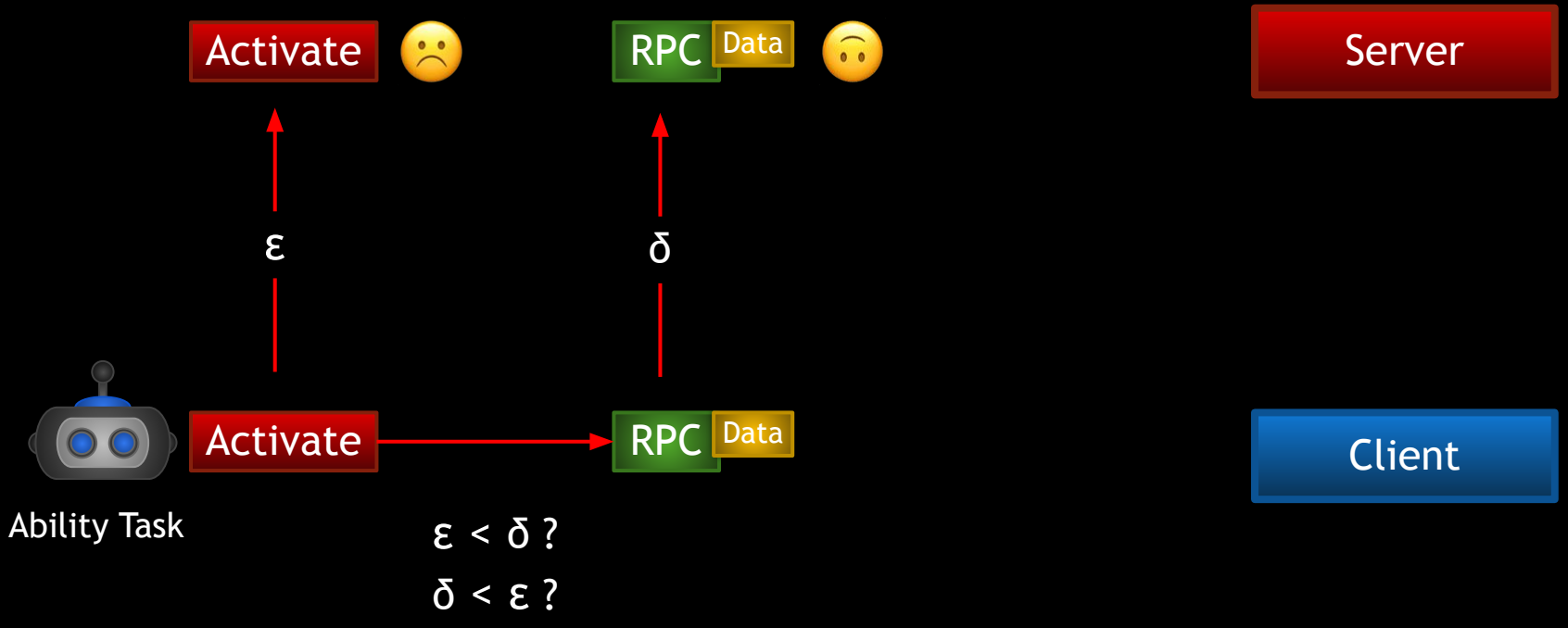
Click To Move

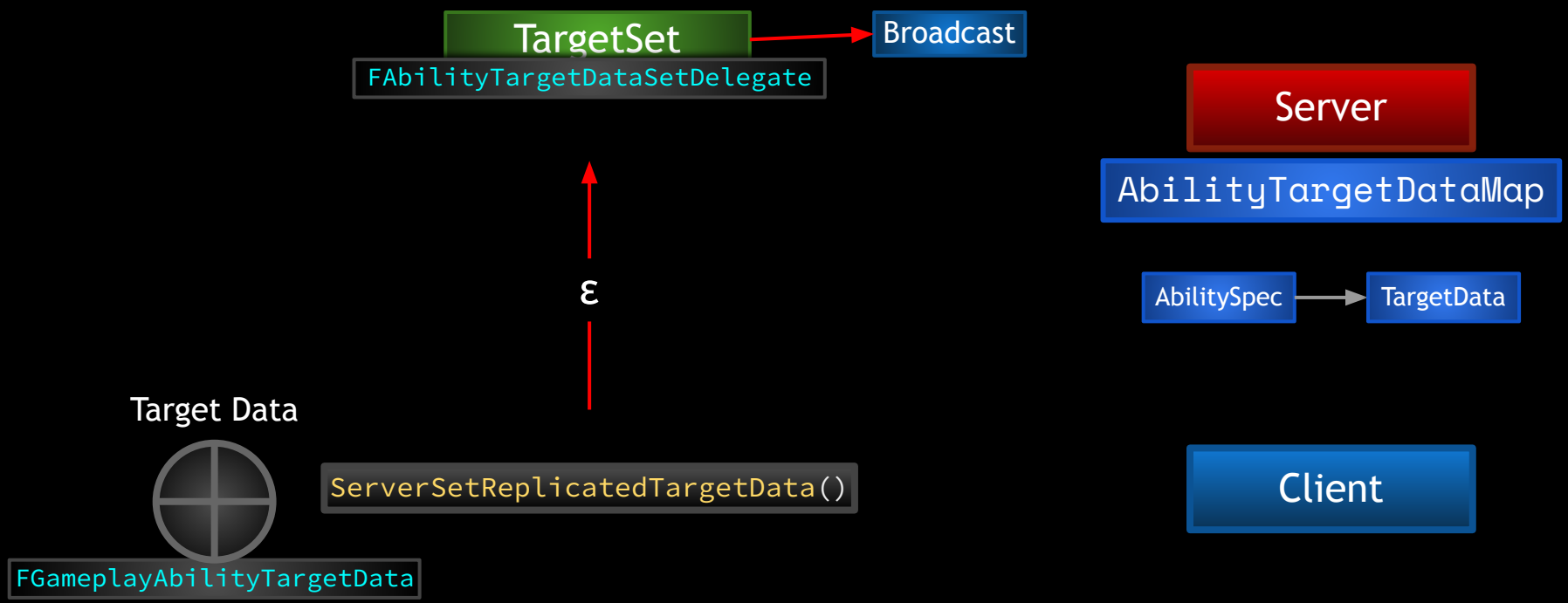


Click To Move

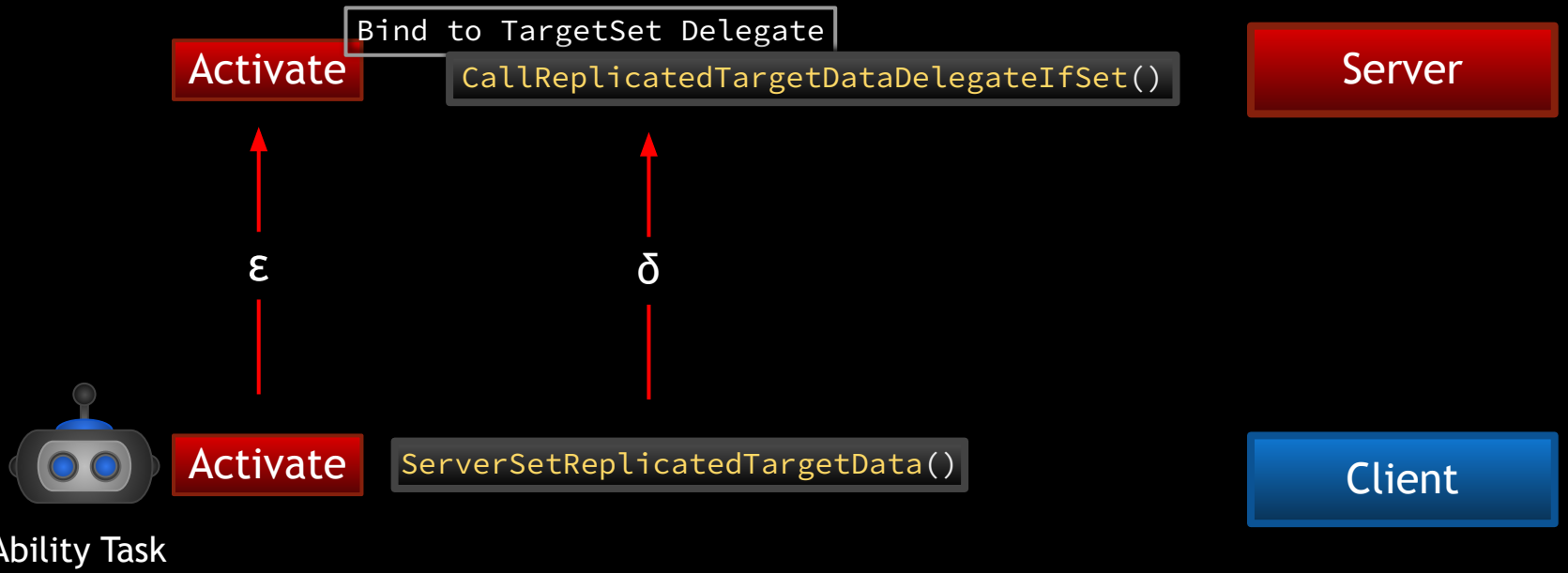


Target Data

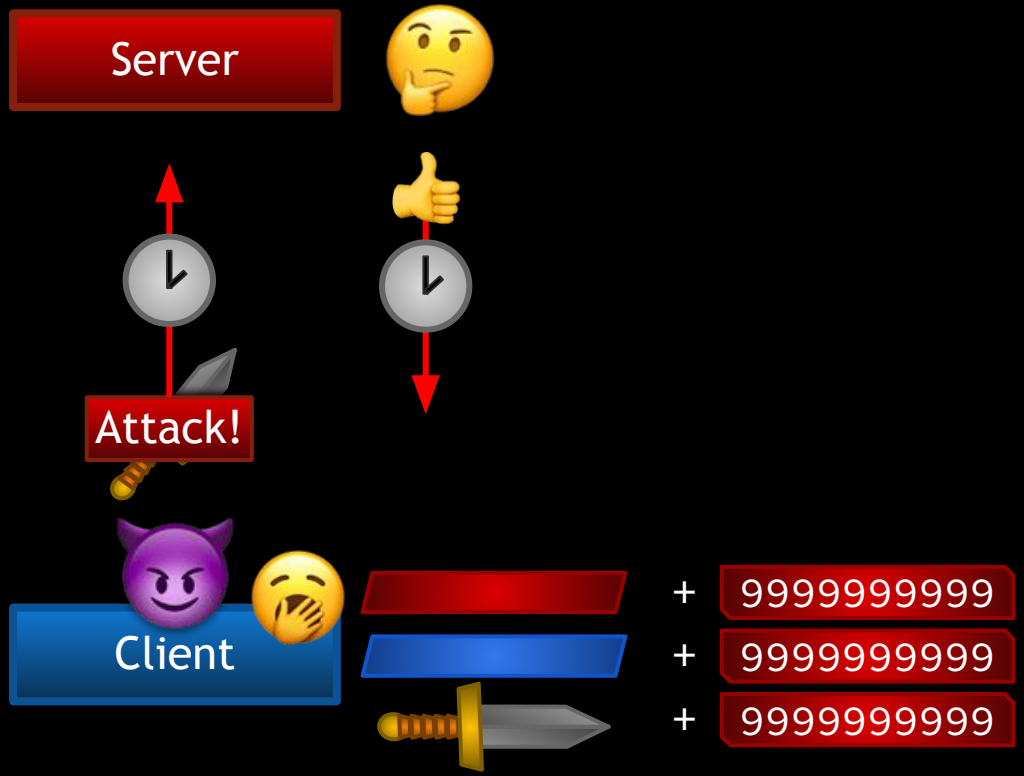

Activate
RPC
Data
Server
ε
δ
Activate
RPC
Data
Client
Ability Task
ε < δ ?
δ < ε ?


TargetSet
Broadcast
FAbilityTargetDataSetDelegate
Server
AbilityTargetDataMap
AbilitySpec
TargetData
ε
Target Data
ServerSetReplicatedTargetData()
Client
FGameplayAbilityTargetData

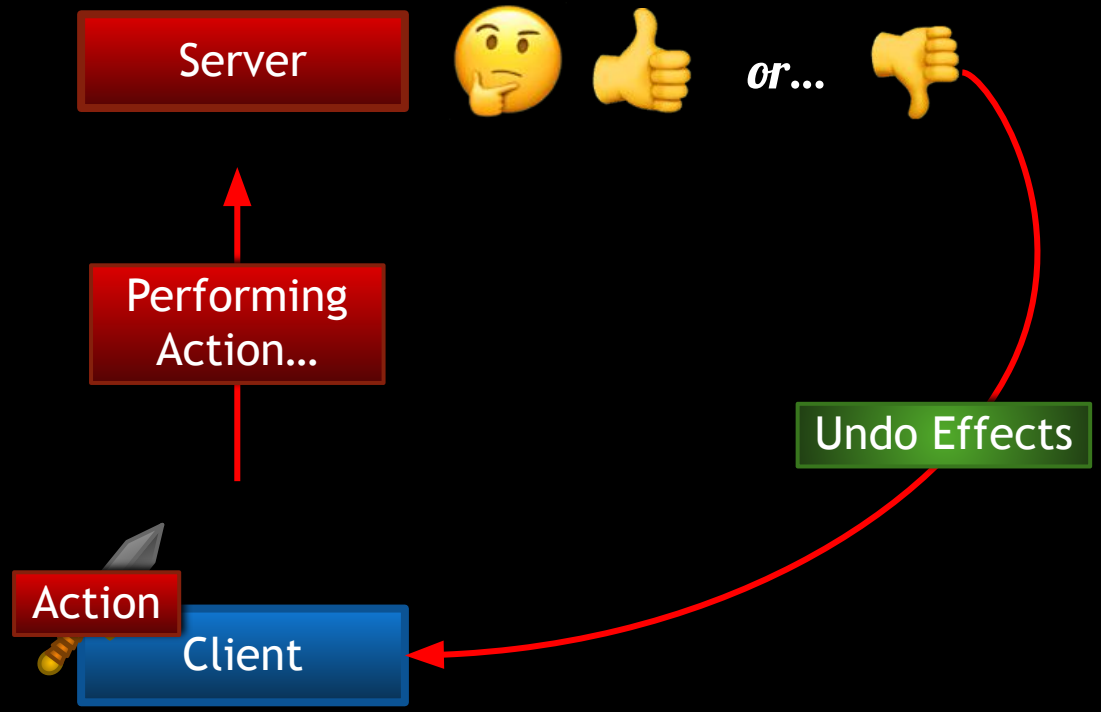
Target Data


Bind to TargetSet Delegate
Activate
CallReplicatedTargetDataDelegateIfSet()
Server
ε
δ
Activate
ServerSetReplicatedTargetData()
Client
Ability Task

Prediction



Prediction



GAS Prediction

Prediction in GAS

- Just works

Developers Didn’t Want:

- if (Authority) Do X
- else: Do predicted version…

Developers Did Want:

- Automatic
  - Not everything needs to be predicted
    - Footstep sounds, etc.
  - Use mana, heal, do damage, etc.

GAS Prediction

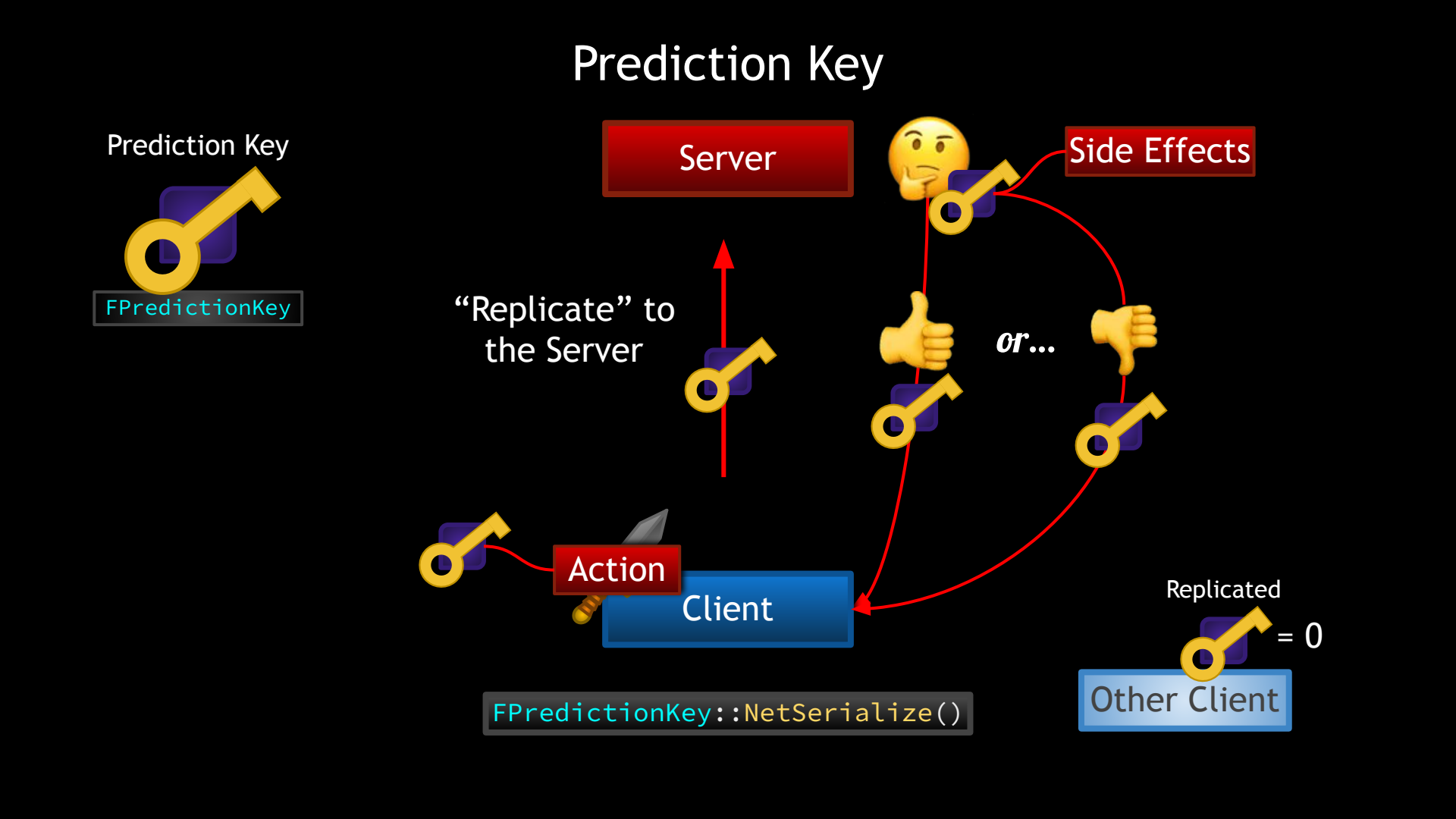
GAS Automatically Predicts:

- Gameplay Ability Activation
- Triggered Events
- Gameplay Effect Application
  - Attribute Modifiers (not Execution Calculations)
  - GameplayTag Modification
- Gameplay Cue Events
  - From within a predicted Gameplay Ability
  - Their own Events
- Montages
- Movement (UCharacterMovement)

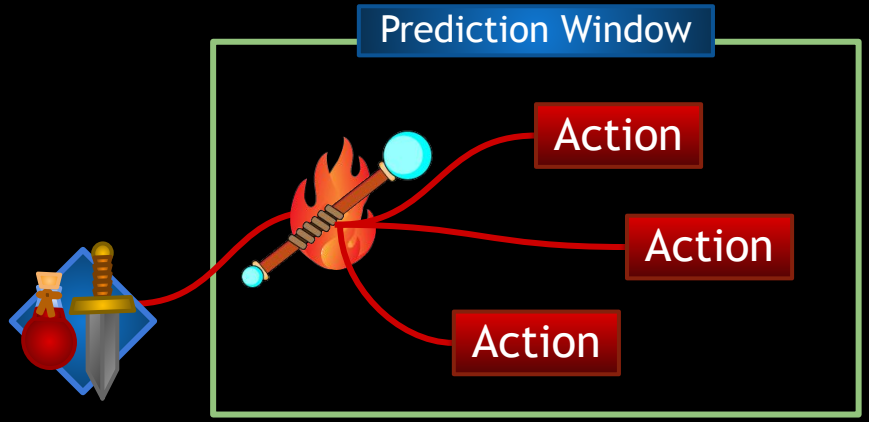
GAS Does NOT Predict:

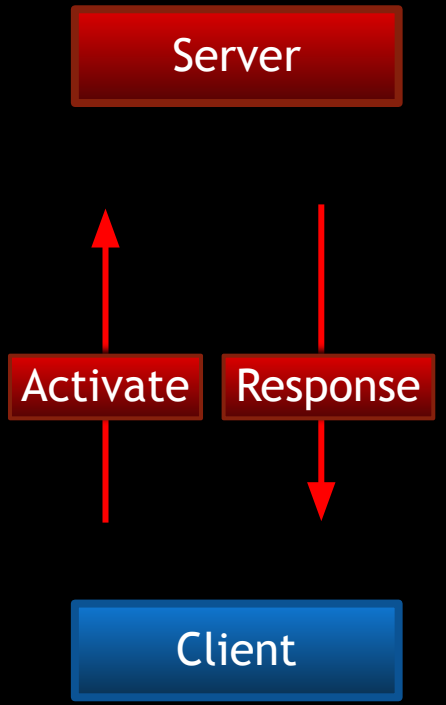
- Gameplay Effect Removal
- Gameplay Effect Periodic Effects

Prediction Key



Ability Activation

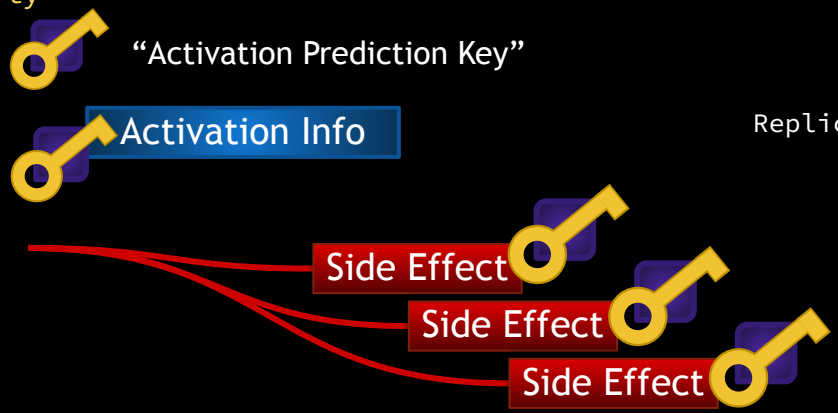




Ability Activation

TryActivateAbility:

- Client calls `TryActivateAbility`
  - New `FPredictionKey`
- Client continues
  - Calls `ActivateAbility`
- Client does things
  - Generates side effects
- `ServerTryActivateAbility`
  - Server decides if valid
  - calls `ClientActivateAbilityFailed`
  - or `ClientActivateAbilitySucceeded`
- Client receives the Server's response
  - If failure, kill the ability and undo side effects
  - If success, side effects are valid.
- ReplicatedPredictionKey replicates
  - OnRep_PredictionKey



Server

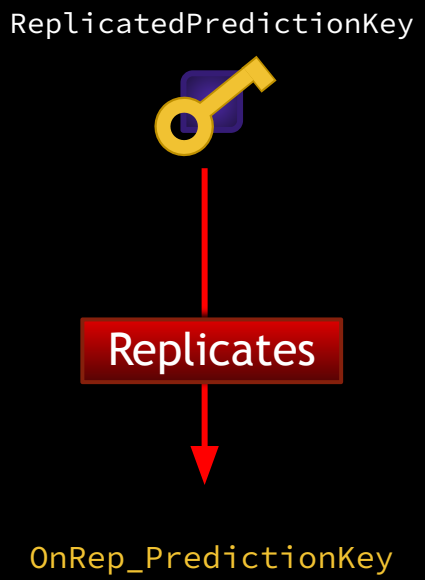
ReplicatedPredictionKey =

Client

Gameplay Effect Prediction

Gameplay Effects

- Side effects
- Only applied on Clients if:
  - There is a valid prediction key
- The following are predicted:
  - Attribute Modifications
  - Gameplay Tag Modifications
  - Gameplay Cues
- When the `FActiveGameplayEffect` is created
  - Stores the Prediction Key
- On the server, it gets the same key
- `FActiveGameplayEffect` is replicated
  - Client checks the key
  - If they match, then “OnApplied” logic doesn’t need to be done



GAS Prediction

For More Info:

- `GameplayPrediction.h`

RPG Character Classes

Warrior

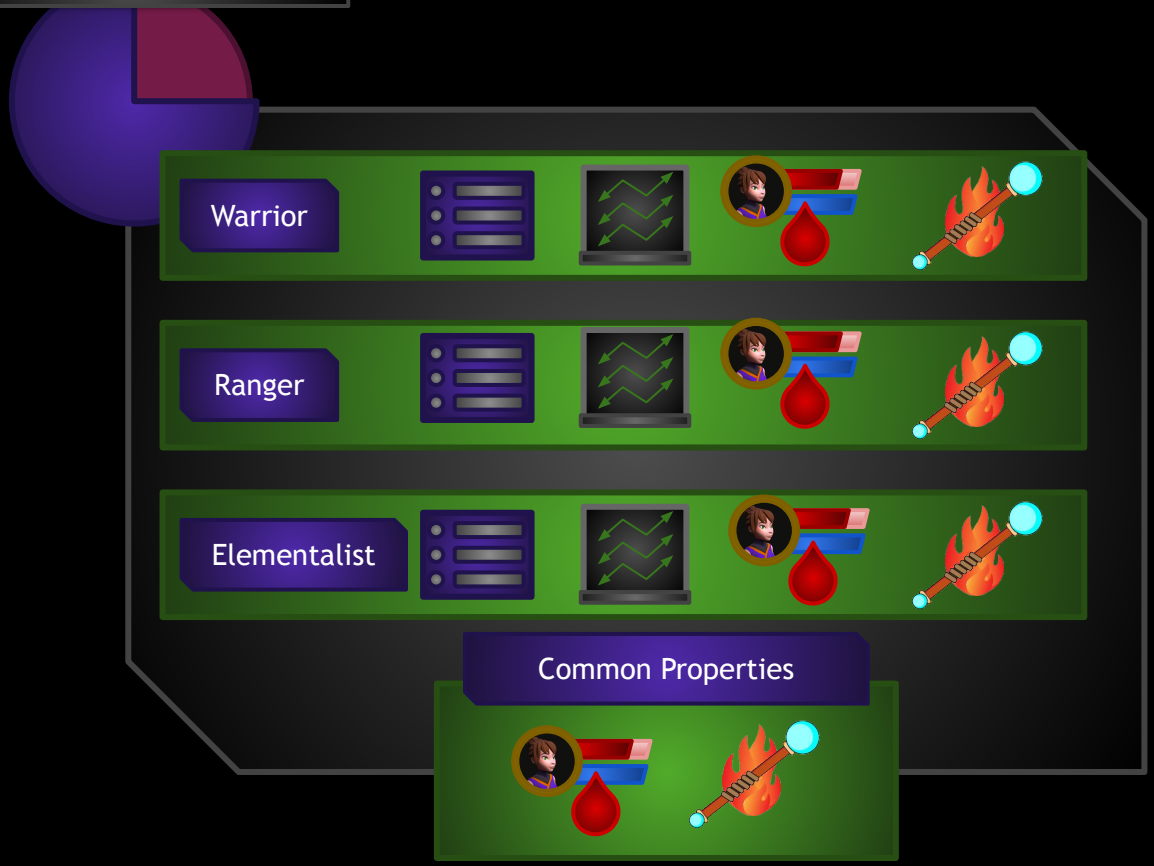
Ranger

Elementalist

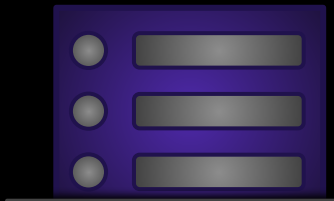
RPG Character Classes

Data Asset

UCharacterClassInfo



Enum



ECharacterClass

Curve Table

CT_PrimaryAttributes

RPG Character Classes

Next Steps

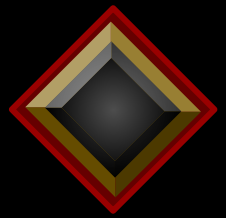
1. Create UCharacterClassInfo Data Asset
2. Create ECharacterClass enum
3. Curve Tables for Attributes
    a. CT_WarriorPrimaryAttributes
    b. CT_RangerPrimaryAttributes
    c. CT_ElementalistPrimaryAttributes
4. Create GEs for Primary, Secondary, and Vital Attributes
5. Shared Abilities and Effects
6. Function to initialize attributes using the Data Asset

Meta Attributes

Gameplay Attribute

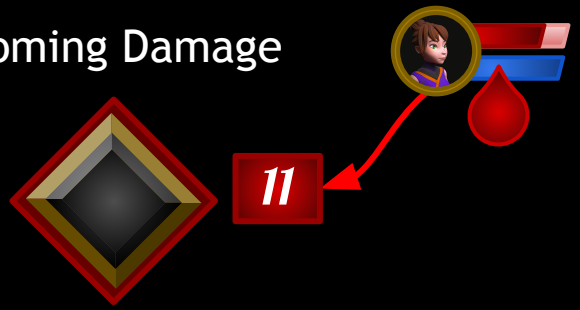
- Replicated

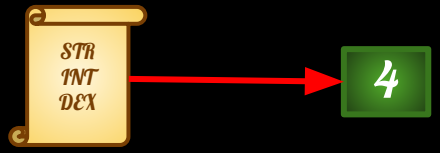
Meta Attribute



- Not Replicated
- Temporary Placeholder
- Allows for Calculations

Incoming Damage





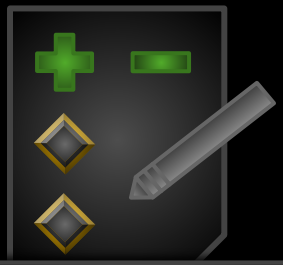
Attribute Set

- Block?
- Critical Hit?
- Intelligence Bonus?
- Strength Bonus?

Change Health

Execution Calculation

Execution Calculation



`UGameplayEffectExecutionCalculation`

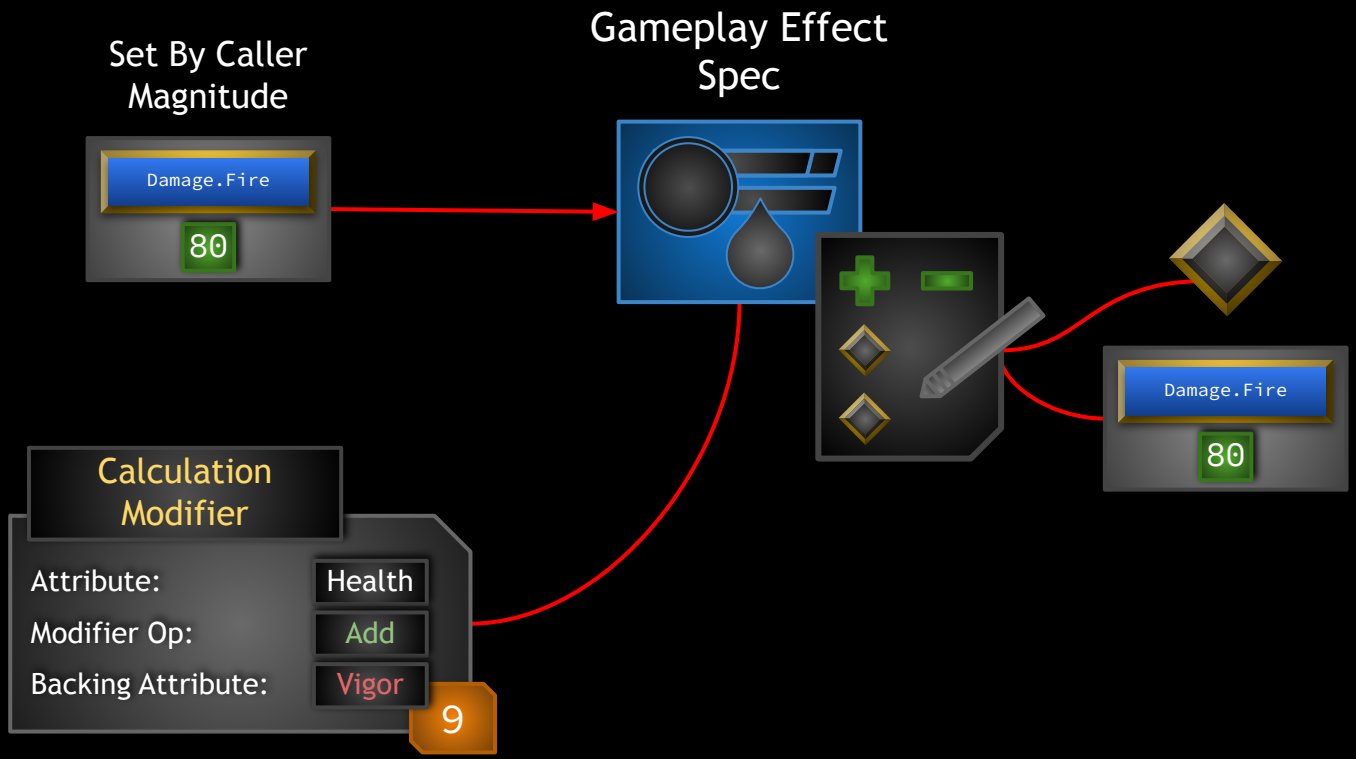
Snapshotting (Source)

- Snapshotting captures the Attribute value when the Gameplay Effect Spec is created
- Not snapshotting captures the Attribute value when the Gameplay Effect is applied
- From the Target, the value is captured on Effect Application only

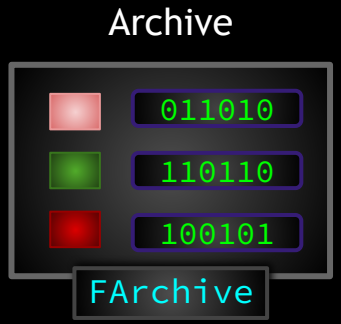
- Capture Attributes
- Can change more than one Attribute
- Can have programmer logic

- No prediction
- Only Instant or Periodic Gameplay Effects
- Capturing doesn't run `PreAttributeChange`; any clamping done there must be done again
- Only executed on the Server from Gameplay Abilities with Local Predicted, Server Initiated, and Server Only Net Execution Policies

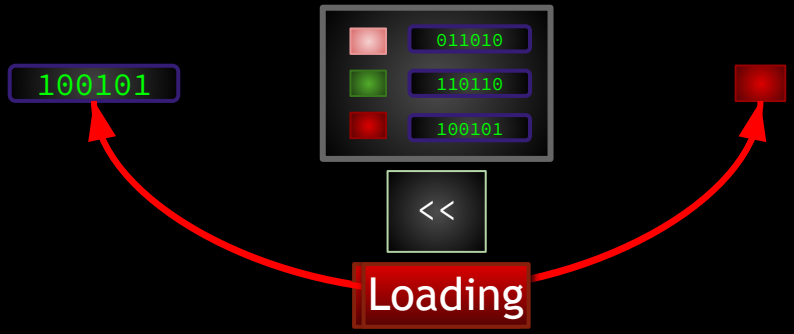
Execution Calculation



Net Serialize



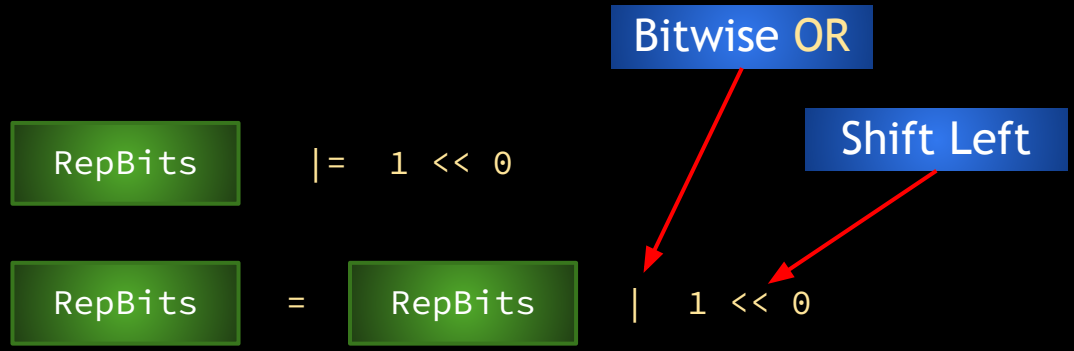
- Saving Data
- Loading Data
- Storing Data
- Serialized
- Overloads << operator
  - << works both ways depending on context (Saving/Loading)



Net Serialize

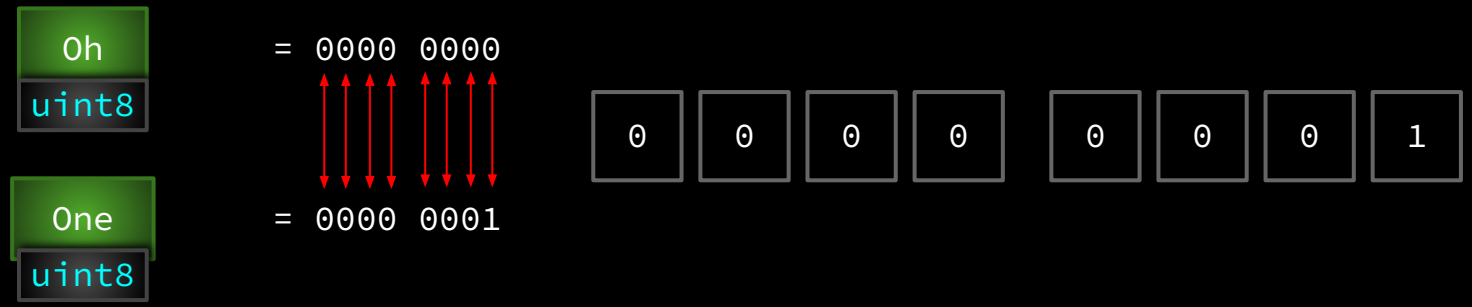
RepBits
uint8

= 0000 0000 8 bits



Net Serialize

Bitwise OR

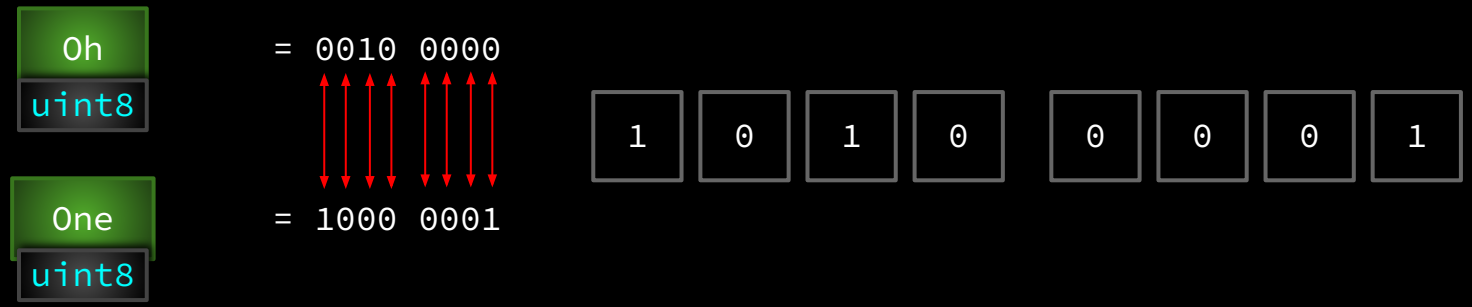


0h | One = ?

= 0000 0001

Net Serialize

Bitwise OR

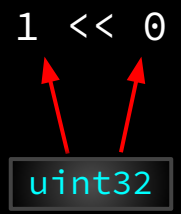


0h | One = ?

= 1010 0001

Net Serialize

Shift Left

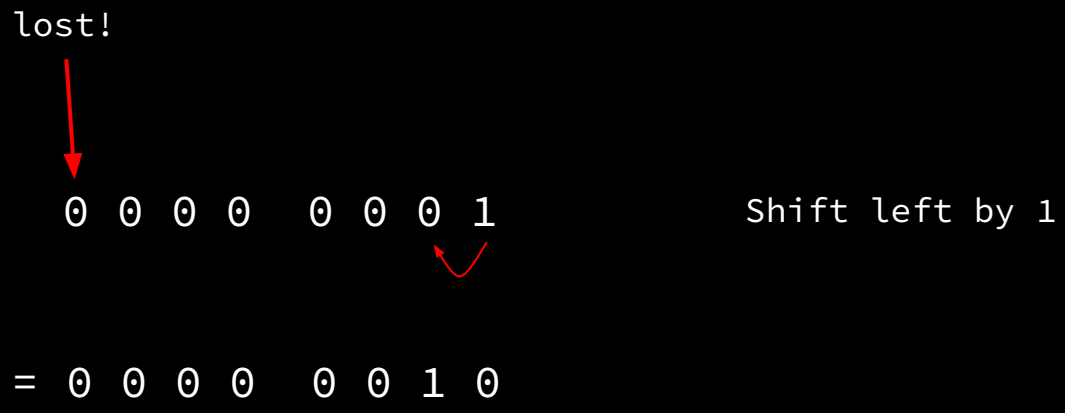


0 0 0 0 0 0 0 1 Shift left by 0

Net Serialize

Shift Left

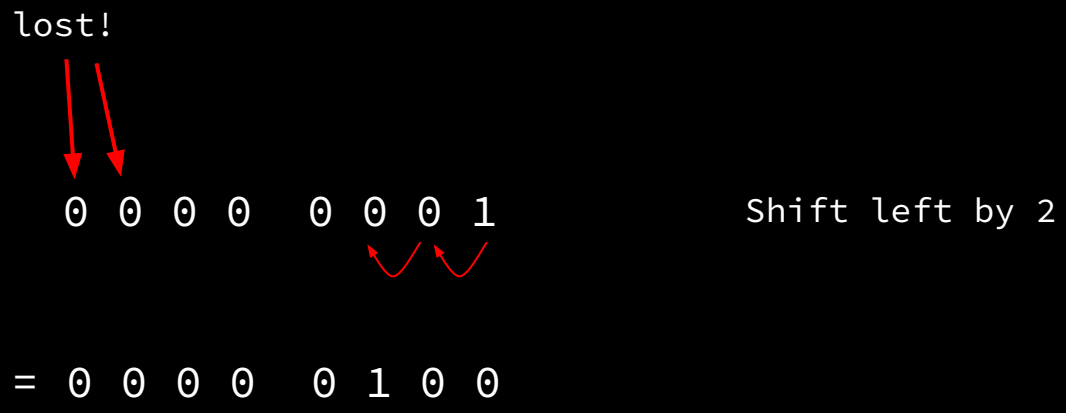
```
1 << 1
```



Net Serialize

Shift Left

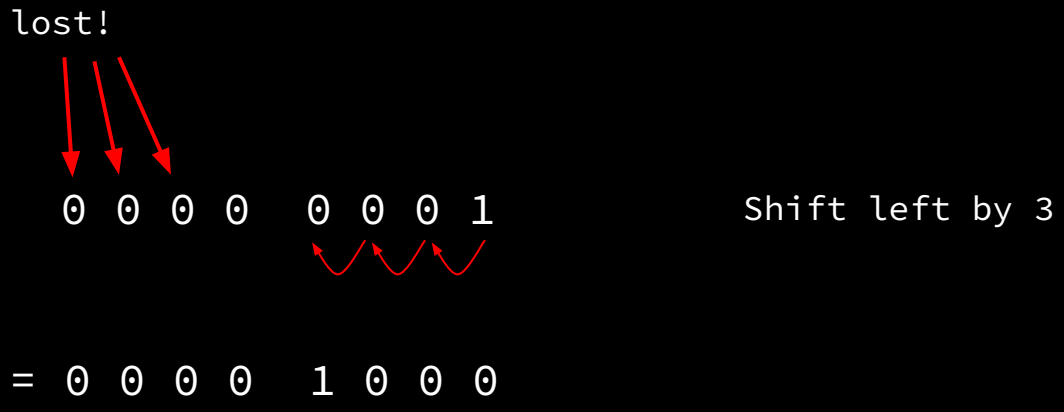
```
1 << 2
```



Net Serialize

Shift Left

```
1 << 3
```



Net Serialize

RepBits
uint32
= 0000 0000 0000 0000 0000 0000 0000 0000

RepBits |= 1 << 0

RepBits = RepBits | 1 << 0

1 << 0 = 0000 0000 0000 0000 0000 0000 0000 0001

Flipped the 0th Bit

RepBits | 1 << 0

0000 0000 0000 0000 0000 0000 0000 0000
| 0000 0000 0000 0000 0000 0000 0000 0001
= 0000 0000 0000 0000 0000 0000 0000 0001

Net Serialize

RepBits = 0000 0000 0000 0000 0000 0000 0000 0001

uint32

RepBits |= 1 << 1

1 << 1 = 0000 0000 0000 0000 0000 0000 0000 0010

Flipped the 1st Bit

RepBits | 1 << 1

0000 0000 0000 0000 0000 0000 0000 0001

| 0000 0000 0000 0000 0000 0000 0000 0010

= 0000 0000 0000 0000 0000 0000 0000 0011

Net Serialize

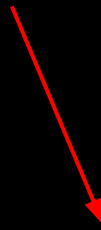
RepBits = 0000 0000 0000 0000 0000 0000 0000 0011

uint32

RepBits |= 1 << 2

1 << 2 = 0000 0000 0000 0000 0000 0000 0000 0100

Flipped the 2nd Bit



RepBits | 1 << 2

0000 0000 0000 0000 0000 0000 0000 0011

| 0000 0000 0000 0000 0000 0000 0000 0100

= 0000 0000 0000 0000 0000 0000 0000 0111

Net Serialize

RepBits = 0000 0000 0000 0000 0000 0000 0000 0111

uint32

RepBits & 1 << 0

1 << 0 = 0000 0000 0000 0000 0000 0000 0000 0001

RepBits & 1 << 0

true

```
  0000 0000 0000 0000 0000 0000 0000 0111
& 0000 0000 0000 0000 0000 0000 0000 0001
= 0000 0000 0000 0000 0000 0000 0000 0001
```

Net Serialize

RepBits = 0000 0000 0000 0000 0000 0000 0000 0111

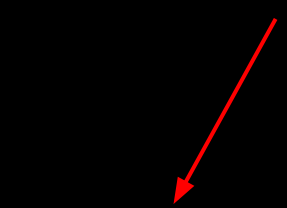
uint32

RepBits & 1 << 3

1 << 3 = 0000 0000 0000 0000 0000 0000 0000 1000

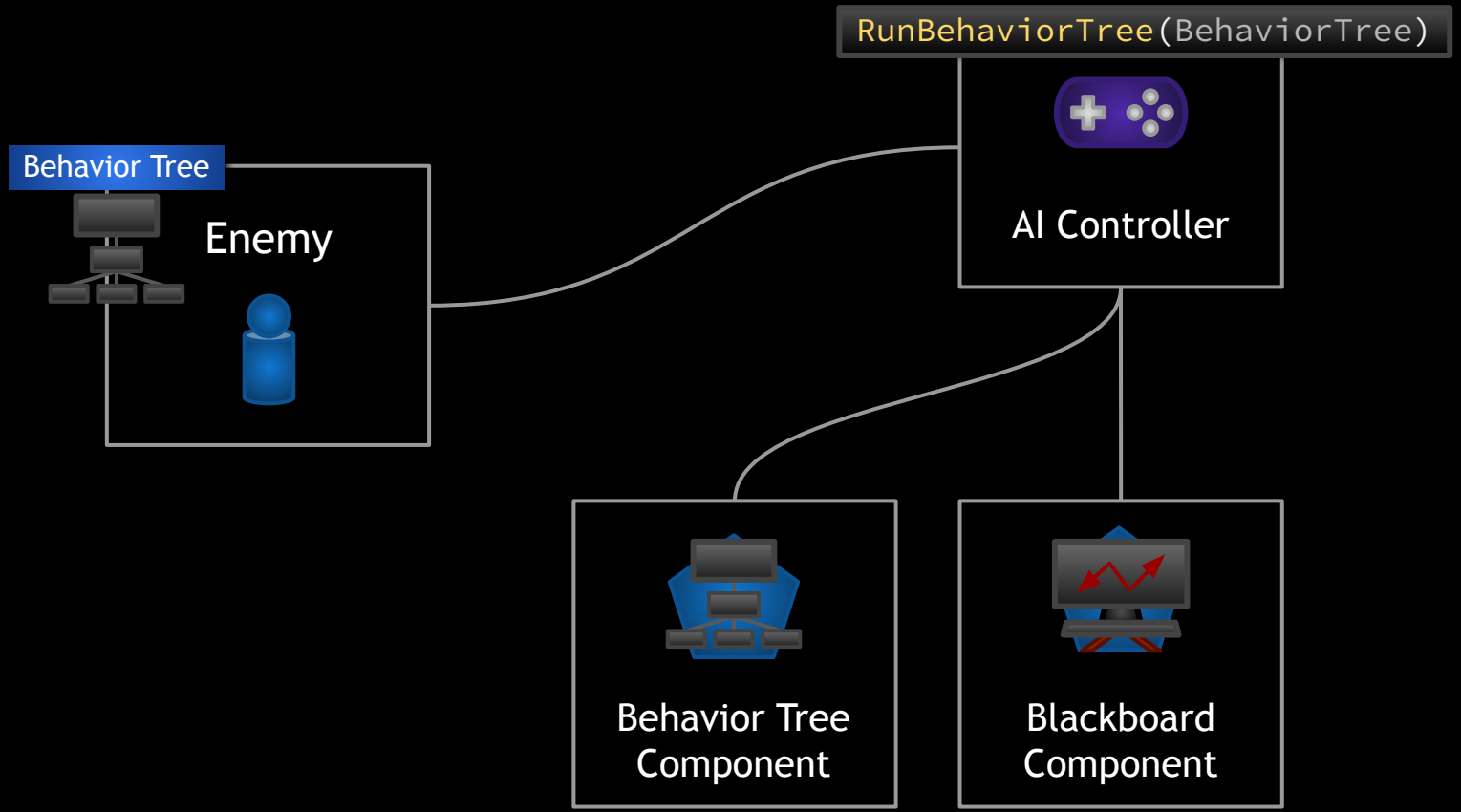
RepBits & 1 << 3

false



```
  0000 0000 0000 0000 0000 0000 0000 0111
& 0000 0000 0000 0000 0000 0000 0000 1000
= 0000 0000 0000 0000 0000 0000 0000 0000
```

Enemy AI

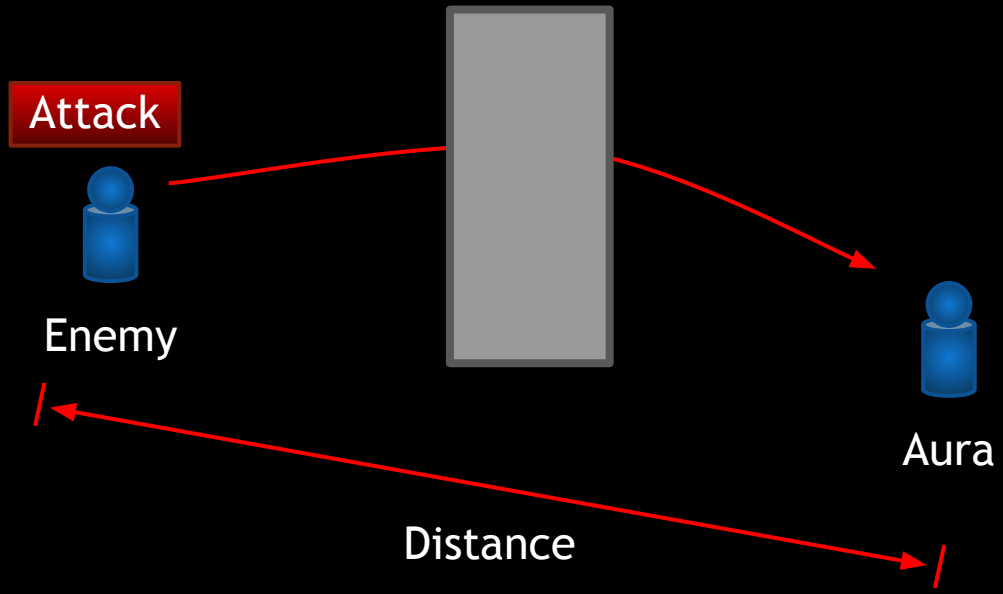

RunBehaviorTree(BehaviorTree)
Behavior Tree
Enemy
AI Controller
Behavior Tree Component
Blackboard Component

Enemy AI

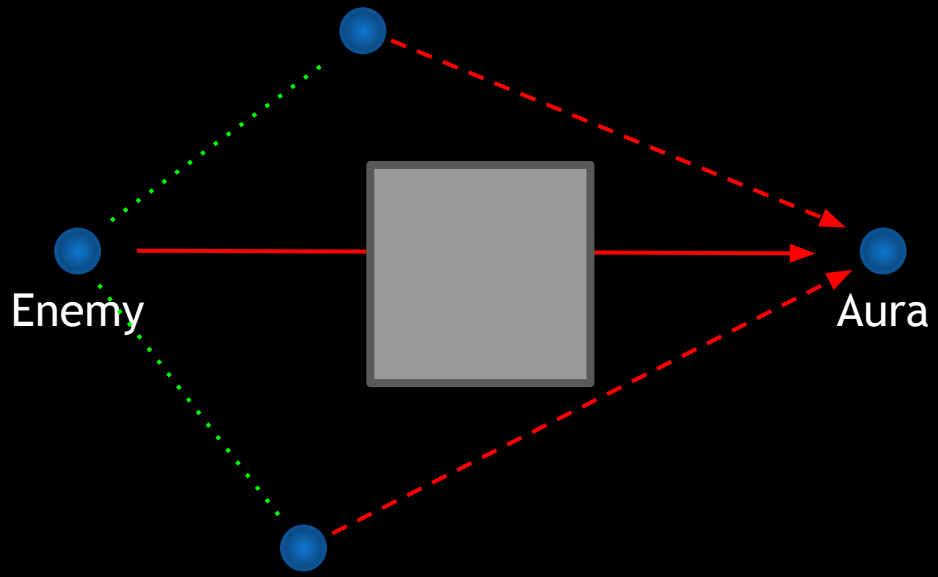
Next Steps

1. Create an AI Controller class
2. Create a Blackboard and Behavior Tree
3. Add a Blackboard Component and Behavior Tree Component to the AI Controller
4. Add a Behavior Tree to the Aura Enemy
5. Run the Behavior Tree

Environment Query System



Environment Query System

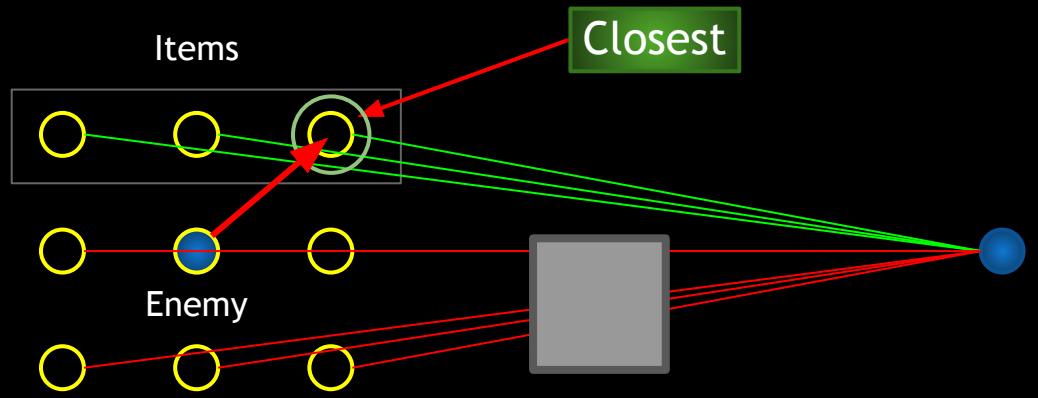


TOP VIEW:

Environment Query System

Tests

- Trace
- Distance



TOP VIEW:

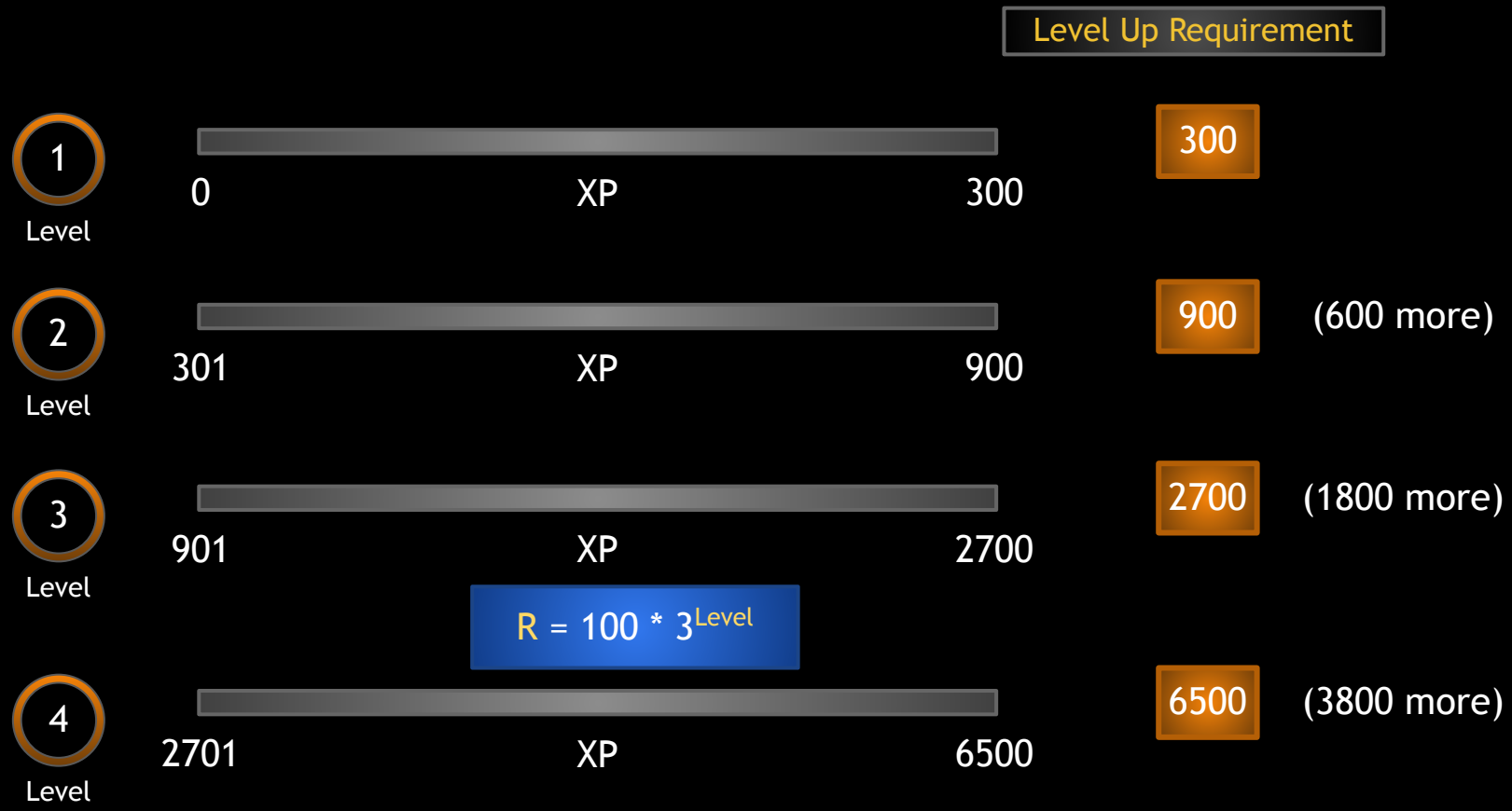
Experience (XP)

Level

Level Up

XP

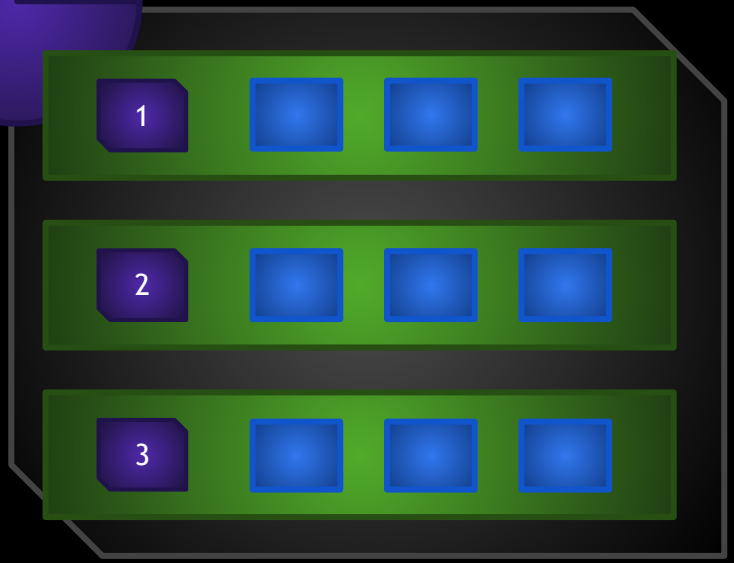
Experience (XP)



Experience (XP)

Data Asset

`ULevelUpInfo`



`FindLevelForXP(float XP)`

Level Up Requirement

`int32`

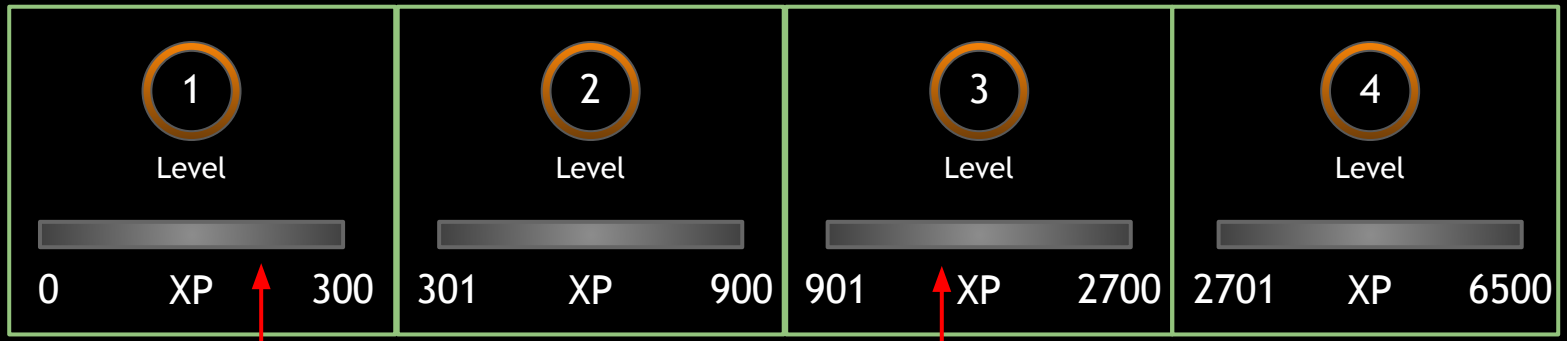
Attribute Point Reward

`int32`

Spell Point Reward

`int32`

Experience (XP)



XP

180 → 1180

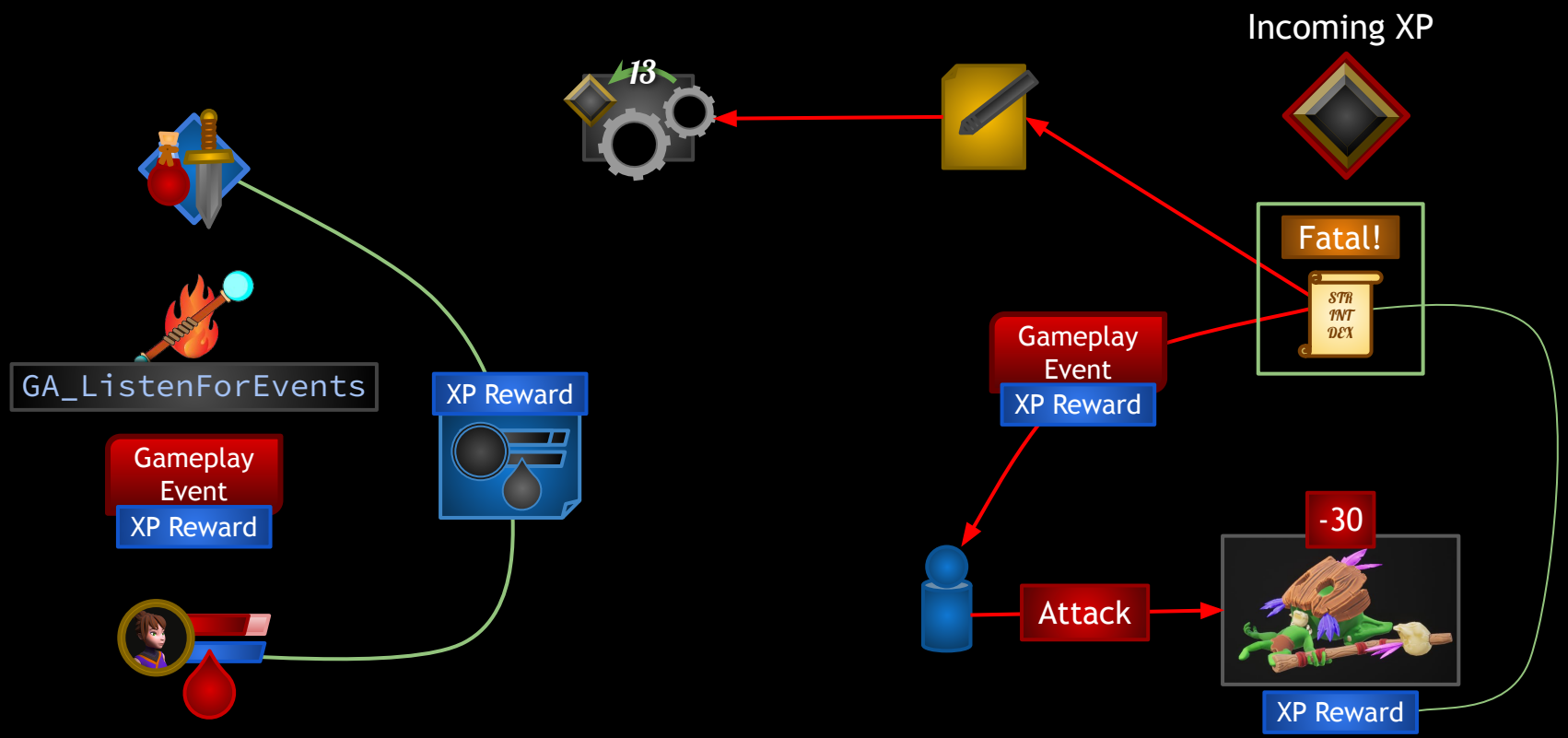
Double Level Up

Experience (XP)

Next Steps

1. Level Up Info Data Asset
2. Add XP to the Player State (with Rep Notify)
3. Delegate to broadcast when XP changes
4. Widget Controller response to delegate broadcast
5. XP Reward - amount of XP each enemy gives based on its level and class
6. Actually award XP when killing an enemy
7. Handle Level Up (handle potentially multiple Level Ups)
    a. Reward Attribute Points
    b. Reward Spell Points

Awarding XP


13
Incoming XP
Fatal!
STR
INT
DEX
GA_ListenForEvents
Gameplay
Event
XP Reward
XP Reward
Gameplay
Event
XP Reward
-30
Attack
XP Reward

Awarding XP

Next Steps

1. XP Reward on Enemies (and a way to get it)
2. Incoming XP Meta Attribute
3. Passive Gameplay Ability, GA_ListenForEvents (and grant it)
    a. Gameplay Effect to apply in response to Event
4. Award XP from Attribute Set when damage is fatal
5. Handle Incoming XP in Attribute Set and increase XP on Player State

Spell Menu

Spell Points

Spend Spell Point

Offensive Abilities

Spell Description

Passive Abilities

Spell Description (next Ability Level)

Equip Spell

Equipped Abilities

Spell Menu

Offensive Spell Tree

Spell Menu

Passive Spell Tree

Ability Status

Ability Status		Ability Status Tag
Locked		Abilities.Status.Locked
Eligible		Abilities.Status.Eligible
Unlocked		Abilities.Status.Unlocked
Equipped		Abilities.Status.Equipped

Ability Type Tag

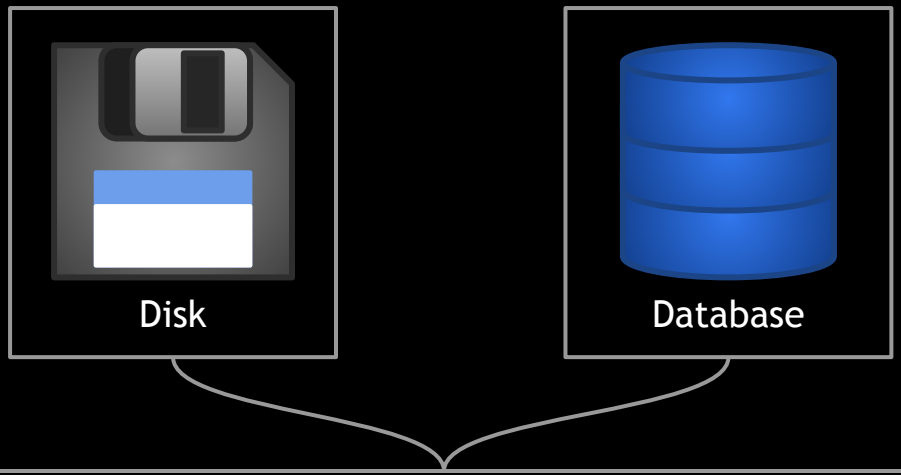
Abilities.Type.Offensive

Abilities.Type.Passive

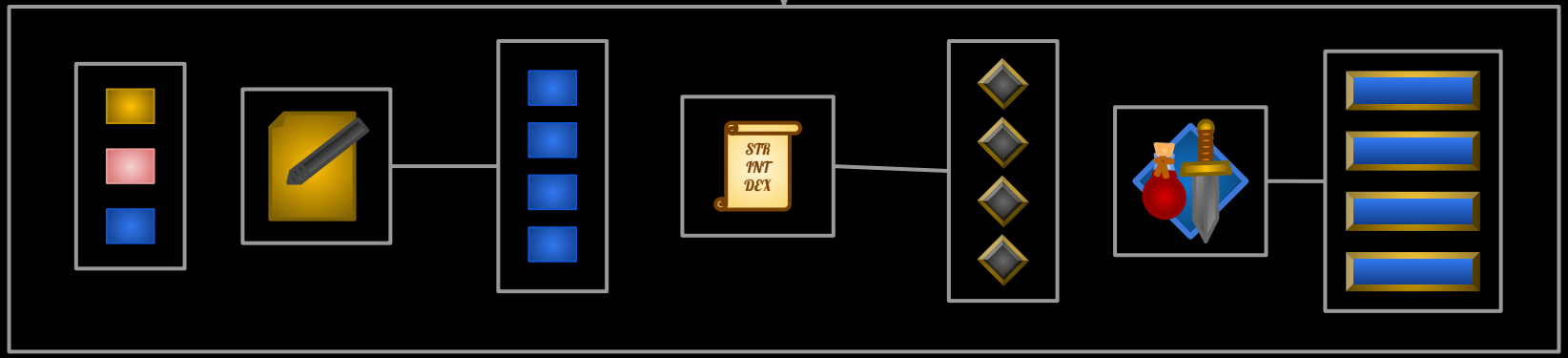
Abilities.Type.None

Saving Progress

- Simple
- Doesn't require external database
- Data is local



- Central Location
- Encrypted
- Dedicated Servers
- Authority
- Requires APIs
- Database knowledge



Model-View-ViewModel

If you put ten software architects into a room and have them discuss what the Model-View-Controller pattern is, you will end up with twelve different opinions.

-Russel East

Model-View-ViewModel