Ventura Crest Studios FrameWork Documentation

Diogo Ventura

March 9, 2024

Contents

1 Introduction

This plugin is a base framework for AAA games , with everything needed to develop a game faster, This was developed by VCS. There are tutorials being recorded for freature requests join the Discord Server. The documentation is vague and there are some parts missing, but the tutorials cover everything so I recomend watching them

I RECOMEND LOOKING AT THE DEMO PROJECT, SINCE IT HAS THE MAJOR PART OF THE FUNCTIONS ETC IMPLEMENTED to get access to it, join the discord server, and verify your purchase.

2 Installation

Just add the plugin to the Plugins folder inside the project of the engine

3 Animations Notify Systems

Animations component is basically animmnotifies

3.1 Input notify

A function and a notify state, when you add the notifystate to the animation, you ll be asked for variables on the setup part, it s an array of a subclass of the function class in bp or c++ which you should create, two booleans , one for hold(if you want the input action to be held or just click), one for bShouldStop that variable is mainly for combos if the key is not pressed on the interval of the animstate it will stop the montage and blendout, the other option is an Action input.

3.2 Collision Check

There is an animation State for start collision check, which should be added, it has a variable of a struct with is swing direction, for hitreactions, it automatically checks the equiped weapon since you have multiple weapons support, works only with the combat component enabled! OnActorHit, it ll call a custom HitObject Function which should be defined on the ANS setup. The weapon should have two sockets setup with the names (WeaponStart and WeaponEnd) for modular weapons you can simple define the blade mesh has the default mesh and add the others on a child, only the blade mesh needs to have the sockets.

3.3 Effect Notify

Spawns a VFX Attached to the specific component, weapon or player, for dashes, trails. Setup it has 2 variables to be setup the niagara system, and the attach surface. Play a camera shake or a sfx too, sfx is attached to camera or player.

3.4 StateNottify

This notify is used to do a state change during an animation

3.5 Rotation Notify

This is an anim notify state that rotates the player or the AI to face one another, it should be used in finishers or attacks, so you don t have misses. It s configurable

3.6 Step Notify

This anim notify is related to the movement component you should add it on every step, so it spawns the VFX based on the surface type and plays the sfx at the location. Also adds a camera shake, if you want it.

3.7 Allow input Notify

This notify state adds input actions to a state that doesn t allow them, could be used for attack cancelation, and others

4 StatesComponent

States component basically defines the state you are at and the allowed inputs

4.1 StatesDataTable

This Datatable provides a row structure of the states, first there is the state-name, which should have State. prefix, and then there is other collumn that defines the allow inputs on that state, the wrtting should be ;INPUTNAME-ONTHEPLAYERINPUTFILE;ANOTHERINPUTNAMME; it should end in ; else the last input won t count and start in ; else the first input won t count. Without containing spaces.

It has a blendspace parameter which works with the movement component to use rootmotion if you leave the blendspace empty the blendspace will be the last blendspace available, used for states like attacking which you allow only custom input with the notify for example.

Function GetBlendSpace should be used in the AnimInstance, it s automatic and is inherited from the base class, that way there s litterally no work to implement movement or custom states.

4.2 StatesComponentMain

There is a function called GetStateNamesFromDataTable , you should call it to define the variables, or just write the name, but it needs to be the same as the datatable. There s another function called CanPerformAction returns true if it can false if it can t. INPUT NAME ON THE ENHANCED INPUT CONFIG NEEDS TO BE THE SAME AS THE FILENAME, ELSE EVERYTHING BREAKS

5 Movement Component

This component provides the movement part with a template on BS /underscore MovementTemplate, the movement should be defined on -2 or 2 for sprinting 1

or -1 for running/walking, 0 for no forward movement or backward. ROOT MOTION ONLY You have an Override due to the vaulting hit trace, OnNPCHit, you can use it to interact with NPCs, that way you use only one HitTrace instead of 2.

5.1 Setup

You should setup the input actions on the editor in the component selection tab, after adding it to the character. On you animation instance derived from our base class, you should get the blendspace and play with the inputs defined, you have the inputs has forwardinput and sideinput.

5.2 Event Override

For the predefined inputs there is an override in blueprints that you can use, it s called InputOverride. It gives you the Uactioninput called, the state and the inputaction value, so there s no need to track the input twice.

5.3 StepsDataTable

You need to create a datatable with 3 params, the structure is predefined, one is the physicalsurface material, the other one the camera shake, if you want, soundfx, and a niagara system, then you need to define the variable on the component called STEPSDATA, the rest is pretty much automatic.

5.4 VaultingComponent

Vault component is inside the movement component, on the movement component change the vaulting min and max height and lenght to your needs, animations are automatically adjusted while playing to the closest point possible to the vault surface, so you can add jump overs or vaults with for example with hand placement automatic, the animations should use rootmotion for better feel. You need to define a HeadSocket on the character skeleton

6 Camera

This component provides a physics based camera

6.1 Setup

Just add the camera component to your character, there are values you can change relatives to the ammount of torque, etc, you want, just select the component on the editor and they ll show up.

7 AILibrary

This component provides a full function library for your NPCS. How does it work? It has the normal NPC BEHAVIOR, you need to setup a datatable with AI data, with the states, Needs the STATECOMPONENT, and the StateActions, which is an UOBJECT like the animnotifies, there are 2 override functions, one ShouldPerform, and another FunctionDefinition, the names are self explanatory. You can always override the AIThinking by calling the function PerformAction(NAME OF THE ACTION); Tutorials explain everything better

7.1 Setup

Just add the component to your AI, this will enable rotatetoAI notify too.

7.2 Other info

There is an extensive library of action UOBJECT definitions ready for AIFunctions, the UOBJECT names reflect what the action does, for any extra request don t exitate to contact us on the Discord Server

8 Combat Component

This component provides a function library for your character, melee combat, may be changed in the future, this component also contains the stats component imbuted since I consider them to be related. You need to define the targetinput if you want the targetsystem. You need to define the Weapon variable if you want to use CollisionTracking

8.1 Setup

Just add the component to your player character.

8.2 Functions

Contains a targetting function to perform, health armor and stamina related functions, you can change the cameralockrange like the other variables on the combat component

9 Base Classes

This section has all the base classes

9.1 Base Character Class

Almost useless, used to ref from the MovementComponent

9.2 Base Weapon Class

Defines the weapon meshes you can call GetWeaponMesh function to get the blade or sword mesh, depending on your setup.

9.3 Function Object

This is used for you to create custom functions and maintain the project organized, create a child of this class it ll be called from the animnotifies. For the collisionNotify override Hitted Function, for normal calls override FunctionTriggered Function.

10 Level Manager v1.2

This is a basic Actor that handles level management, using levelstreaming, enabling you to add loadscreens preventing game stoppages when changing levels. Loading the new level and unloading the past one.

10.1 Setup

Just add the actor to the playercharacter or the scene

10.2 Functions

```
void LoadBackGroundLevel(TSoftObjectPtr<UWorld> LevelToLoad, float loadtime, FName LoadingSt
                         TSubclassOf<UUserWidget> LoadingScreen, UVCS_InputFunction_OBJ* Cal
```

Everything is option except the LevelToLoad and PlayerRef.

11 API

11.1 AILibrary

- `UFUNCTION(BlueprintCallable, Category="AILibrary")`
  `void SetAllowBehave(bool value);`
- `UFUNCTION(BlueprintCallable, Category="AILibrary")`
  `void PerformFunction(FName FunctionName);`
- `UFUNCTION(BlueprintPure, Category="AILibrary")`
  `static EHitDirection GetDirection(FVector HitNormal, FVector RightVector, FVector ForwardVector);;`
- `UPROPERTY(BlueprintReadWrite, EditAnywhere, Category="AILibrary")`
  `float TimerInterval;`

11.2 Camera Component

- `UPROPERTY(BlueprintReadWrite, Category="CameraComponent")`
  `float VelocityRange = 400.0f;`
- `UPROPERTY(BlueprintReadWrite, Category="CameraComponent")`
  `float TorqueAmount = 7.0f;`
- `UPROPERTY(BlueprintReadWrite, Category="CameraComponent")`
  `float MaxPitch = 25.0f;`
- `UPROPERTY(BlueprintReadWrite, Category="CameraComponent")`
  `float MaxRoll = 25.0f;`
- `UPROPERTY(BlueprintReadWrite, Category="CameraComponent")`
  `float MaxYaw = 25.0f;`
- `UPROPERTY(BlueprintReadWrite, Category="CameraComponent")`
  `float InterpSpeed = 5.0f;`
- `UPROPERTY(BlueprintReadWrite, Category="CameraComponent")`
  `float PanAmount = 50.0f;`
- `UPROPERTY(BlueprintReadWrite, Category="CameraComponent")`
  `float MaxPanX = 100.0f;`
- `UPROPERTY(BlueprintReadWrite, Category="CameraComponent")`
  `float MaxPanY = 100.0f;`
- `UPROPERTY(BlueprintReadWrite, Category="CameraComponent")`
  `float MaxPanZ = 100.0f;`
- `UPROPERTY(BlueprintReadWrite, Category="CameraComponent")`
  `float FieldOfViewAmount = 4.0f;`

- UPROPERTY(BlueprintReadWrite, Category="CameraComponent")
  float MaxFieldOfView = 120.0f;

11.3 Combat Component

- UPROPERTY(BlueprintReadWrite, Category="CombatComponent")
  AVCS$_{B}aseWeapon_{A}ctor * Weapon$;
- UFUNCTION(BlueprintPure, Category="CombatComponent")
  float GetHealth();
- UFUNCTION(BlueprintPure, Category="CombatComponent")
  float GetArmor();
- UFUNCTION(BlueprintPure, Category="CombatComponent")
  float GetStamina();
- UFUNCTION(BlueprintCallable, Category="CombatComponent")
  float TakeHealth(float Damage, bool bCountArmor);
- UFUNCTION(BlueprintCallable, Category="CombatComponent")
  void Target(AActor* actor);
- UFUNCTION(BlueprintCallable, Category="CombatComponent")
  void UnTarget();
- UFUNCTION(BlueprintCallable, Category="CombatComponent")
  void PerformAttack(FName AttackingState, UAnimMontage* Montage,
  float PlaySpeed);
- UFUNCTION(BlueprintCallable, Category="CombatComponent")
  ACharacter* GetTargetOrPossible();
- UFUNCTION(BlueprintCallable, Category="CombatComponent")
  void PerformFinisher(ACharacter* Killer, ACharacter* Victim, UAnimMontage*
  KillerMontage, UAnimMontage* VictimMontage, FName KillerState,
  FName VictimState, float DesiredDistance);
- UPROPERTY(EditDefaultsOnly, BlueprintReadWrite, Category="CombatComponent")
  class UInputAction* TargetInput;
- UPROPERTY(EditDefaultsOnly, BlueprintReadWrite, Category="CombatComponent")
  float DefaultHealth = 100.f;
- UPROPERTY(EditDefaultsOnly, BlueprintReadWrite, Category="CombatComponent")
  float DefaultArmor = 0.f;
- UPROPERTY(EditDefaultsOnly, BlueprintReadWrite, Category="CombatComponent")
  float DefaultArmor = 0.f;

- UPROPERTY(EditDefaultsOnly, BlueprintReadWrite, Category="CombatComponent")
  float DefaultStamina = 100.f;
- UPROPERTY(EditDefaultsOnly, BlueprintReadWrite, Category="CombatComponent")
  float CameraLockRange = 1500.f;

11.4 Movement Component

- UFUNCTION(BlueprintPure, Category="MovementComponent")
  FVector2D GetInputValues();
- UFUNCTION(BlueprintPure, Category="MovementComponent")
  bool GetSprint();;
- UFUNCTION(BlueprintNativeEvent, Category = "MovementComponent")
  void InputOverride(UInputAction* InputAction, ETriggerEvent State,
  const FInputActionValue value);
- UFUNCTION(BlueprintImplementableEvent, Category="MovementComponent")
  void OnNPCHit(AActor* actor);

11.5 State Component

- UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="StateManager")
  UDataTable* DStatesTable;
- UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="StateManager")
  bool IsAI;
- UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="StateManager")
  UDataTable* AITable;
- UFUNCTION(BlueprintCallable, Category="StateManager")
  TArray<FName> GetStateNamesFromDataTable();
- UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="StateManager")
  FName DefaultState;
- UPROPERTY(BlueprintReadWrite, Category="StateManager")
  FName CurrentState;
- UFUNCTION(BlueprintCallable, Category="StateManager")
  bool CanPerformAction(UInputAction* Input, FString Actionname)
- UFUNCTION(BlueprintPure, Category="StateManager")
  UBlendSpace* GetCurrentBlendSpace();
- UFUNCTION(BlueprintCallable, Category="StateManager") bool IsStateValid(FName
  StateName);

11.6 AllowCustomInput ANS

- `UPROPERTY(EditAnywhere, Category = "ActionSetup")`
  `TArray<UInputAction*> ActionList;`

11.7 AllowCustomInput ANS

- `UPROPERTY(BlueprintReadWrite, EditAnywhere, Category="StateNotify")`
  `FName DesiredState;`

11.8 CollisionTrace ANS

- `UPROPERTY(EditAnywhere, Category="ActionSetup")`
  `TSubclassOf<UVCS`$InputFunction_OBJ > sFunctionCall$;

11.9 InputCall ANS

- `UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="Notify")`
  `UInputAction* EInput;`
- `UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="Notify")`
  `UVCS`$InputFunction_OBJ * sFunctionCall$;
- `UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="Notify")`
  `bool bStopOnMiss;`
- `UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="Notify")`
  `bool bHold;`

11.10 Input Function Object

- `UFUNCTION(BlueprintImplementableEvent, Category = "Triggers")`
  `void FunctionTriggered(float fHoldTime);`
- `UFUNCTION(BlueprintImplementableEvent, Category = "Triggers")`
  `void Hitted(struct FHitResult hitresult);`
- `UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="Notify")`
  `bool bStopOnMiss;`
- `UFUNCTION(BlueprintCallable, Category="HitReact")`
  `void SpawnBloodVFX(UNiagaraSystem* NiagaraSystem, FVector location,`
  `AActor* HitActor);`

11.11 Play VFX ANS

- UPROPERTY(EditAnywhere, Category="Notify")
  UNiagaraSystem* VFXToPlay;
- UPROPERTY(EditAnywhere, Category="Notify")
  EAttachInstruction VFXInstruction;
- UPROPERTY(EditAnywhere, Category="Notify")
  USoundCue* SoundToPlay;
- UPROPERTY(EditAnywhere, Category="Notify")
  EAttachInstruction instruction;

11.12 RotateToAI AN

11.13 StepNotify AN

11.14 AIDataTable

- UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="StateComponent")
  FName StateName;
- UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="StateComponent")
  FName ActionName;
- UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="StateComponent")
  TSubclassOf<UVCS_AIFunctionObject_OBJ> ActionObject;
- UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="StateComponent")
  TSubclassOf<UVCS_AIFunctionObject_OBJ> ShouldPerfomObject;
- UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="StateComponent")
  bool bIsDefault = false;

11.15 States DataTable

- UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="StateComponent")
  FName StateName;
- UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="StateComponent")
  FString AllowedInputs;
- UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="StateComponent")
  UBlendSpace* BlendSpace;

11.16 Steps DataTable

- `UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="StepNotify")`
  `UPhysicalMaterial* PhysicalMaterial;`
- `UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="StepNotify")`
  `TSubclassOf<UCameraShakeBase> CameraShake;`
- `UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="StepNotify")`
  `USoundCue* SoundCue;`
- `UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="StepNotify")`
  `UNiagaraSystem* NiagaraSystem;`

11.17 AI Function Object

- `UFUNCTION(BlueprintImplementableEvent, Category = "Triggers")`
  `void FunctionDefinition(AActor* NPCRef);`
- `UFUNCTION(BlueprintImplementableEvent, Category = "Triggers")`
  `bool ShouldPerfom(AActor* NPCRef);`

11.18 Base weapon actor

- `class USkeletalMeshComponent* WeaponSkeletalMesh;`
- `class UStaticMeshComponent* WeaponStaticMesh;`

11.19 Character Base

11.20 DebugHelper NameSpace

- `static void Print(const FString MSG, const FColor Color = FColor::MakeRandomColor(),`
  `int32 inKey = -1)`