ARuinsRhapsodyState : AGameStateBase

AAlex* Alex
AWorldData* WorldData
AEndLevelTrigger* EndLevel
float LevelStartTime
float LevelCompletionTime
FText LevelName
uint8 EnemiesInLevel
uint8 EnemiesKilled
uint8 ItemsInLevel
uint8 ItemsPicked

ARuinsRhapsodyState()
void TryGetAlex()
void TryGetWorldData()
void TryGetEndLevel()
virtual void Tick(float DeltaSeconds) override
void StartTimer()
void EndTimer()
void OnAddItem(FItemStruct Item, int Quantity)
void OnLoadEnd()
void OnEnemyKilled()
FORCEINLINE float GetLevelCompletionTime() const

URuinsRhapsodyInstance : UGameInstance, public ISaveLoadSystemInterface

UMySaveGame* SaveGame
FString SaveName
UAudioComponent* MusicPlaying

virtual void NewGame_Implementation() override
virtual void SaveGame_Implementation() override
virtual void LoadGame_Implementation() override
virtual UMySaveGame* GetGame_Implementation() override
virtual void DeleteSave_Implementation() override
virtual void SaveGameData_Implementation(const FGameDataStruct GameDataStruct) override
virtual void SavePlayerData_Implementation(const FPlayerDataStruct PlayerDataStruct) override

SAVE SYSTEM

UMySaveGame : USaveGame

FGameDataStruct GameData
FWorldDataStruct WorldData
FCharacterDataStruct CharacterData
TArray<FInventoryPair> CharacterInventory
TArray<FInventoryPair> WorldInventory

UMySaveOptions : USaveGame

FOptions Options

struct FOptions

float Master
float Music
float Effects
float Dialogues
float UI
bool InvertPitch

struct FGameDataStruct

FDateTime DateTime
FString LastLeveMap
bool bisNewLevel
bool bisNewGame

struct FWorldDataStruct

UTexture2D* BackgroundImage
FTextLevelName
TArray<FDoorSaveData> SavedDoors
TArray<FMechanismSaveData> SavedMechanisms

AWorldData : AActor

DECLARE_DYNAMIC_MULTICAST_DELEGATE(FOnLoadEnd)
FOnLoadEnd OnLoadEnd
AAlex* Alex
UInventoryComponent* InventoryComponent
UMySaveGame* SaveGame
FWorldDataStruct WorldData
FDelays Delays
FVector CharacterOffsetLoad
FVector CharacterOffsetNewLevel

AWorldData()
virtual void BeginPlay() override
void CheckLoadEndData()
void LoadWorldItems()
void DestroyItems(TArray WorldActors)
void LoadDoors()
void LoadMechanisms()
void LoadCharacterData()
void GetInvertPitch() const
void SetCharacterInventory() const
void TeleportCharacter()
FItemStruct GetLastItemOfType(EItemType ItemType) const
void SaveData()
FGameDataStruct SetGameData() const
FCharacterDataStruct SetCharacterData()
void EndLoad() const
FText GetLevelName() const

INTERFACES

TriggerInterface : UInterface

UTriggerInterface
ITriggerInterface

void OnActivation()
void OnDeactivation()

InteractionInterface : UInterface

UInteractionInterface
IInteractionInterface

void OnCharacterOverlap(AAlex* Alex)
void OnCharacterEndOverlap(AAlex* Alex)
void Interact(AAlex* Alex)

MechanismInterface : UInterface

UMechanismInterface
IMechanismInterface

void Interaction()

SaveLoadSystemInterface : UInterface

USaveLoadSystemInterface
ISaveLoadSystemInterface

void NewGame()
void SaveGame()
void LoadGame()
void DeleteSave()
UMySaveGame* GetGame()

void SaveGameData(const FGameDataStruct GameData)
void SaveWorldData(const FWorldDataStruct WorldData)
void SaveCharacterData
(const FCharacterDataStruct CharacterData)
void SaveInventory
(const TArray& CharacterInventory, const TArray& WorldInventory)

void NewOptions()
void SaveOptions(FOptions Options)
void LoadOptions()
UMySaveOptions* GetOptions()

ACheckpoint : AItem

UArrowComponent* Direction
URotatingMovementComponent* RotatingMovement

ACheckpoint()
virtual void Interact_Implementation(AAlex* Alex) override
void SaveData(const AAlex* Character) const
void SaveDoors(AWorldData* WorldDataClass) const
void SaveMechanisms(AWorldData* WorldDataClass) const
FGameDataStruct GetGameData() const
void DestroyCheckpoint()

CHARACTER

UAlexInput : UDataAsset

UInputAction* Move
UInputAction* Camera
UInputAction* CameraReset
UInputAction* Walk
UInputAction* Sprint
UInputAction* Crouch
UInputAction* Jump
UInputAction* Climb
UInputAction* Flashlight
UInputAction* Interact
UInputAction* Weapon
UInputAction* Shoot
UInputAction* Reload
UInputAction* Inventory
UInputAction* Pause
UInputAction* SmallHealth
UInputAction* BigHealth
UInputAction* PreviousWeapon
UInputAction* NextWeapon

Alessandra

AAlex : ACharacter

AAlexState* AlexState
AWorldData* WorldData
UMotionWarpingComponent* MotionWarpingComponent
UClimbComponent* ClimbComponent
USlideComponent* SlideComponent
UInventoryComponent* InventoryComponent
UFallingDamageComponent* FallingComponent
UPawnNoiseEmitterComponent* NoiseEmitterComponent

USpringArmComponent* SpringArm
UCameraComponent* Camera
FRotator TargetCameraRotation
FRotator TargetSpringArmRotation
FRotator TargetControllerRotation
float CameraInterpTime
bool WantToResetCamera
bool CanResetCamera
FVector StandSpringArmPosition
FVector CrouchSpringArmPosition
float ResetCameraInterpolation
float CameraResetTolerance
float TurnRate
float LookUpRate
bool InvertPitch
float CameraDistanceThreshold
APlayerCameraManager* CameraManager

float JumpDelay
float JumpTimer
bool CanJump

bool CanMove
bool bisMovingBackward
uint8 InteractionType
bool bisSliding
FTimerHandle SlidingCheckDelay
bool bWantToSwimDown
FVector CheckBreathPosition
FTimerHandle CrouchHandle
bool bWantToUncrouch
float CrouchingEndCheck
USkeletalMeshComponent* Weapon
TSubclassOf<AWeapon> FirstWeapon
AWeapon* LastWeapon
USoundBase* HolsterWeaponSound
USoundBase* UnHolsterWeaponSound
bool bisShooting
FTimerHandle ShootingTimerHandle
IMaterialInterface* BulletHitDecal
FVector BulletHitDecalSize
UParticleSystem* BulletHitParticleSystem

AAlex()
virtual void BeginPlay() override
void InitializeCharacter()
virtual void Tick(float DeltaSeconds) override
void GenerateNoise(float Loudness)
UMotionWarpingComponent* GetMotionWarping() const
void CheckCameraDistance() const
void NormalRotationSetup()
void WeaponRotationSetup()
void Death() const

void SetSliding()
void ResetSlide()

void Move(const FInputActionValue& Value)
void GroundMotion(const FInputActionValue& Value)
void StopMove()

void RotateCamera(const FInputActionValue& Value)
void ResetCamera()
void ResettingCameraSmoothly(float DeltaTime)

void Run()

void Walk()
void EndWalk()

void Sprint()
void EndSprint()

void WantToCrouch()
void WantToUncrouch()
void NoCrouch()
void CheckUncrouch()
void Uncrouching()

void CheckJump(float DeltaTime)
virtual void Jump() override
void ResetJump()
virtual void StopJumping() override

void Interaction()
void ReInteractions()

void Climb() const
void ClimbMovement(const FInputActionValue& Value) const
void ClimbingRotation(FRotator Rotation)
void PlayClimbSoundEffects()
void StopClimbing() const
void StopClimbingRotation()
void PlayLedgeAnim()

void StartSwimming()
void StopSwimming()
void SwimmingMovement(FInputActionValue Value)
void SwimUp()
void SwimDown()

void ActivateFlashlight()

void DrawLastWeapon()
void DrawWeapon(AWeapon* WeaponToUse)
void PutAwayWeapon()
void Shoot()
void ContinuousShooting()
void StopShooting()

struct FCharacterDataStruct

float Health
float Stamina
float Breath
float Battery

AAlexStates : APlayerState

FCharacterDataStruct CurrentData

bool bisInGame
uint8 Weapon

float RotationSpeed
float WalkSpeed
float RunSpeed
float SprintSpeed
float SwimSpeed
float CrouchSpeed
float JumpZ
float WalkableAngle
ACameraHUD* CameraHUD
float GetHUDTimer
float GetHUDTimeSet

float MaxHealth
float MaxStamina
float MaxBreath
float MaxBattery

bool bisSwimming
bool bisUnderTheWater
float BreathDrainRate
float BreathRecoveryRate
float HealthDrainRate
TArray<AActor*> WaterBoxes

bool bisMoving
bool CanSprint
bool WantToSprint
bool bisSprinting
float StaminaDrainRate
float StaminaRecoveryRate

bool bisFlashlightActivated
float FlashlightReactivationValue
float BatteryDrainRate
float BatteryRecoveryRate

AAlexState()
virtual void Tick(float DeltaSeconds) override
void GetHUD()

bool bisDead
void UpdateHealth(float Health)
void UpdateHealthBar() const
void HideHealthBar() const

void DrainBreath(const float DeltaTime)
void RecoverBreath(const float DeltaTime)
void UpdateBreathBar() const
void HideBreathBar() const

void DrainStamina(const float DeltaTime)
void RecoverStamina(const float DeltaTime)
void UpdateStaminaBar() const

void HideStaminaBar() const

void DrainBattery(const float DeltaTime)
void RecoverBattery(const float DeltaTime)
void UpdateBatteryBar() const
void HideBatteryBar() const

void NewGame()

void SetCharacterData
(const FCharacterDataStruct& PlayerDataToSet)
void UpdateBarsAtStart()

AAlexController : APlayerController

UInputMappingContext* MappingContext
UAlexInput* AlexInput
AAlex* Alex

bool bisInventoryOpen = false
float WeaponDelay
FTimerHandle WeaponDelayHandle

void SetupInputComponent() override
void BeginPlay() override

void Move(const FInputActionValue& Value)
void StopMove()

void RotateCamera(const FInputActionValue& Value)
void ResetCamera

void Walk()
void EndWalk()

void Sprint()
void EndSprint()

void Crouch()
void UnCrouch()
void NoCrouch()

void Jump()
void StopJump()

void Interaction()
void AlexInteraction()

void Climb()

void EndClimb()

void Flashlight()

void Inventory()

void StartShooting()
void StopShooting()

void PauseGame()

ENEMY AI

AIEnemyCharacter : ACharacter

UPawnSensingComponent* PawnSensingComponent
UMaterialInterface* HitDecal
FVector HitDecalSize
UParticleSystem* HitParticleSystem
USoundBase* ShotSound
UParticleSystem* MuzzleFlashParticleSystem

FEnemyInfo EnemyInfo

FTimerHandle DestroyCorpseTimerHandle
float DestroyCorpseTimer
UWidgetComponent* WidgetComponent
UEnemyHealthBar* HealthBar
bool bisDead

AAlex* Alex
AAIEnemyController* AIController
UBlackboardComponent* BB
FOnEnemyKilled OnEnemyKilled
TSubclassOf ItemToSpawn
FVector SpawnOffset

AAIEnemyCharacter()
virtual void BeginPlay() override
void OnHearPawn(APawn* Pawn, const FVector& Location, float Volume)
void OnSeePawn(APawn* Pawn)

void EnemyHit(const UWorld* Context, const FHitResult& Hit, float Damage)
void AttackPlayer() const
void Death()
void DestroyCorpse()

struct FEnemyInfo

float Health
float Damage

UEnemyHealthBar : UUserWidget

UProgressBar* HealthBar
float MaxHealth

void UpdateHealthBar(float Health) const

AAIEnemyController : AAIController

UBehaviorTree* EnemyBehaviorTree

AAIEnemyCharacter* EnemyCharacter

AAIEnemyController()

virtual void OnPossess(APawn* InPawn) override

UBTTask_Attack : UBTTask_BlackboardBase

UBTTask_Attack()

virtual EBTNodeResult::Type ExecuteTask
(UBehaviorTreeComponent& OwnerComp, uint8* NodeMemory) override

UBTTask_ChasePlayer : UBTTask_BlackboardBase

UBTTask_ChasePlayer()

virtual EBTNodeResult::Type ExecuteTask
(UBehaviorTreeComponent& OwnerComp, uint8* NodeMemory) override

UBTTask_LookAround : UBTTask_BlackboardBase

float SearchRadius

UBTTask_LookAround()

virtual EBTNodeResult::Type ExecuteTask
(UBehaviorTreeComponent& OwnerComp, uint8* NodeMemory) override

COMPONENTS

UFallingDamageComponent : UActorComponent

AAlex* Character

float MinImpactSpeed
float DamagePerUnit
bool WasFalling
float FallStartZ
bool ToxicDamage
float CheckDelay

UFallingDamageComponent()

virtual void TickComponent(float DeltaTime, ...) override

UCheckObjectComponent : UCapsuleComponent

AActor* OverlapActor
FName TagToCheck

UCheckObjectComponent()

void OnBeginOverlap(..., AActor* OtherActor, ...)
void OnEndOverlap(...)
void Interact()

USlideComponent : UActorComponent

AAlex* Character

float SlopeAngle
float MaxSlopeAngle
float MinSlopeAngle
float SlidingDelay
bool CanCheckSliding
USoundBase* SlidingSound
bool bisSlidingSound
UAudioComponent* AudioComponent

USlideComponent()

virtual void TickComponent(float DeltaTime, ...) override
void CheckSlide()
void StopSlide()

UClimbComponent : UActorComponent

AAlex* Character

float CheckClimbOffset
float AttachDistanceClimb
float AttachDistanceMovement
bool bisClimbing
float ClimbLipDown
float ClimbLeftRight
float HeightLedgeOffset
float LedgeDistanceDetection
float FloorLedgeDetection
bool bHasExecuteOnce
float MotionWarpOffsetX
float MotionWarpOffsetY
float MotionWarpOffsetZ
float IdleToMovement
float IdleToMovementTimer
bool bCanMove
float CheckClimbCapsuleRadius
float CheckClimbCapsuleHalfHeight
USoundBase* SoundEffect
FVector ClimbDirection
float ClimbSpeed

UClimbComponent()

virtual void TickComponent(float DeltaTime, ...) override

void StartClimb()
FHitResult CheckClimb(float AttachDistance) const

void ClimbFHitResult HitResult)

void ClimbMovement(const FVector& WorldDirection, float ScaleValue)

void StopClimbing()
void CalculateLedge()
void ResetDoOnce()

UInventoryComponent: UActorComponent

FOnAddItem OnAddItem
FOnRemoveItem OnRemoveItem
TArray<TSubclassOf<AItem>> InitialInventory
TArray<FInventoryPair> Inventory

void AddItem(const FItemStruct& Item)

UInventoryComponent()
void SetCharacterInitialInventory()

void AddItem(AItem* Item, uint8 Quantity = 1)
void RemoveItem(const AItem* Item, uint8 Quantity)
bool ContainsStruct(const FItemStruct& ItemToCheck)

struct FInventoryPair

FItemStruct Key
int Value

bool operator==(const FInventoryPair& Other) const

ITEMS

enum class EItemType : uint8

Dev
Checkpoint
Weapon
Ammo
Utility
Secret

struct FItemStruct : public FTableRowBase

int UniqueID
FString DevName
FString LeveName
FVector Position
FRotator Rotation
bool operator==(const FItemStruct& Other) const

FString Name
UTexture2D* Icon
EItemType ItemType
TSubclassOf<AItem> ItemRef

FORCEINLINE uint32 GetTypeHash
(const FItemStruct& ItemStruct)

Item : AActor, public IInteractionInterface

UStaticMeshComponent* StaticMesh
UWidgetComponent* WarningWidget
FItemStruct ItemStruct
USoundBase* SoundToPlay
float SoundVolume
uint8 Quantity

AItem()
virtual void BeginPlay() override
void DestroyItem()
virtual void OnCharacterOverlap_Implementation(AActor* Alex) override
void BindEvents(AAlex* Alex)
void ShowWidget() const
void HideWidget() const
void PlaySound() const
virtual void Interact_Implementation() override
virtual void UseItem(UInventorySlot* Slot, UWorld* World, AAlex* Character) override
UStaticMeshComponent* GetStaticMesh() const

AKeyDoor : AItem

int32 KeyID
bool bCanBeUsed

AKeyDoor()

virtual void UseItem(UInventorySlot* Slot, UWorld* World, AAlex* Character) override

AMedPack : AItem

float Heal

AMedPack()

virtual void UseItem(UInventorySlot* Slot, UWorld* World, AAlex* Character) override

WEAPONS

struct FWeaponStruct

uint8 WeaponType
float Damage
float ShootingRate

AWeapon : AItem

USkeletalMeshComponent* SkeletalMeshComponent
FWeaponStruct WeaponStruct
FTimerHandle ShootingTimerHandle
USoundBase* ShotSound
UParticleSystem* MuzzleFlashParticleSystem
UMaterialInterface* BulletHitDecal
FVector BulletHitDecalSize
UParticleSystem* BulletHitParticleSystem

AWeapon()
virtual void UseItem(UInventorySlot* Slot, UWorld* World, AAlex* Character) override
void Shoot(USkeletalMeshComponent* WeaponMeshComponent, UWorld* Context)
void PerformShoot(USkeletalMeshComponent* WeaponMeshComponent, UWorld* Context) const
void SpawnImpact(const UWorld* Context, const FHitResult& Hit) const

MECHANISMS

AMechanism : AActor, public IInteractionInterface

UStaticMeshComponent* StaticMesh
USceneComponent* MovablePartPivot
UStaticMeshComponent* MovableMesh
FTransform StartTransform
FTransform EndTransform
float Speed
USoundBase* SoundToPlay
UWidgetComponent* WarningWidget
TArray<AActor*> ActorsAttached
bool bisTraverse
int KeyID
uint8 InteractionType
bool CanBeReused
AAlex* Alex

AMechanism()
virtual void BeginPlay() override
virtual void Tick(float DeltaTime) override
virtual void OnCharacterOverlap_Implementation(AActor* Character) override
virtual void Interact_Implementation() override

void MoveMechanism()
void ActorsInteractions()
bool CheckReuse() const
void BindEvents(AActor* Character)
void ShowWidget() const
void HideWidget() const

ADoor : AActor, public IMechanismInterface

UStaticMeshComponent* StaticMesh
USceneComponent* MovablePartPivot
UStaticMeshComponent* MovableMesh
FTransform StartTransform
FTransform EndTransform
float Speed
bool bisOpened
USoundBase* SoundBase

ADoor()

virtual void BeginPlay() override
virtual void Tick(float DeltaTime) override
virtual void Interaction_Implementation() override

TRIGGER

EndLevelTrigger : ATriggerBox

DECLARE_DYNAMIC_MULTICAST_DELEGATE(FOnEndLevel)

FOnEndLevel OnEndLevel

APlayerController* PlayerController
AGameHUD* GameHUD
FString NextLevelName

AEndLevelTrigger()

virtual void BeginPlay() override
void OnTrigger(AActor* OverlappedActor, AActor* OtherActor)

ATrapActivation : ATriggerBox

TArray<AActor*> LinkedTraps

virtual void BeginPlay() override
void OnBeginOverlap(AActor* OverlappedActor, AActor* OtherActor)
void OnEndOverlap(AActor* OverlappedActor, AActor* OtherActor)

TRAPS

ATrap : AActor, public ITriggerInterface

UStaticMeshComponent* StaticMeshComponent

bool bisActivated

ATrap()

virtual void OnActivation_Implementation() override
virtual void OnDeactivation_Implementation() override

AArrowShooterTrap : ATrap

TSubclassOf<AActor> ArrowToSpawn
UArrowComponent* ShootingPoint
UBoxComponent* BoxDestroyArrow
float TimeToShoot
float Timer
USoundBase* ShotSound
AActor* SpawnedArrow

AArrowShooterTrap()

virtual void Tick(float DeltaTime) override
void Shot()
void DestroyArrow(..., AActor* OtherActor, ...)

AArrow : AActor

UBoxComponent* CollisionComponent
UStaticMeshComponent* MeshComponent
UProjectileMovementComponent* ProjectileMovementComponent
float TimeToDestroy
float Timer
float Damage

AArrow()

void OnOverlapBegin(..., AActor* OtherActor, ...)

MAIN MENU

AMenu3DActor : AActor

UCameraComponent* CameraComponent
UWidgetComponent* WidgetComponent

float BlendTime
USoundBase* TransitionSound
TArray<AMenu2DActors> LinkedMenus

AMenu3DActor()

virtual void BeginPlay() override

void CheckTransition(const uint8 Index)
void TransitionTo(AMenu3DActor* MenuToTransition)

UMenu3DWidget : UUserWidget

DECLARE_DYNAMIC_MULTICAST_DELEGATE_OneParam(FTransitionToLinkedMenu, uint8, MenuIndex);

FTransitionToLinkedMenu TransitionToLinkedMenu

HUD

GAME UI

AGameHUD : AHUD

TSubclassOf<UGameWidget> GameWidgetSubclass

UGameWidget* GameWidget

TSubclassOf<UUserWidget> LoadingWidgetSubclass

UUserWidget* LoadingWidget

TSubclassOf<UInventoryWidget> InventoryWidgetSubclass

UInventoryWidget* InventoryWidget

TSubclassOf<UPauseWidget> PauseWidgetSubclass

UPauseWidget* PauseWidget

TSubclassOf<UEndLevelWidget> EndLevelSubclass

UEndLevelWidget* EndLevelWidget

TSubclassOf<UDeathWidget> DeathWidgetSubclass

UDeathWidget* DeathWidget

virtual void BeginPlay() override

void CreateLoadingScreen()

void RemoveLoadingScreen()

void CreateGameWidget()

void CreateInventoryWidget()

void CreatePauseWidget()

void CreateEndLevelWidget()

void ShowEndLevelWidget(APlayerController* PlayerController, FString NextLevel) const

void RemoveEndLevelWidget()

void PauseVisibility(bool Visibility) const

void SetMaxValues(const float MaxHealth, const float MaxBreath, const float MaxStamina, const float MaxBattery) const

void SetCurrentValues(const float CurrentHealth, const float CurrentBreath, const float CurrentStamina, const float CurrentBattery) const

void UpdateHealthBar(const float CurrentBreath) const

void UpdateBreathBar(const float CurrentBreath) const

void UpdateStaminaBar(const float CurrentStamina) const

void UpdateBatteryBar(const float CurrentBattery) const

void HideHealthBar() const
void HideBreathBar() const
void HideStaminaBar() const
void HideBatteryBar() const

void ShowHealthBar() const
void ShowBreathBar() const
void ShowStaminaBar() const
void ShowBatteryBar() const

void SwitchOffFlashlight()
void Inventory(bool WantToOpen)
void CreateDeathWidget()

UGameWidget : UUserWidget

UProgressBar* HealthBar
UProgressBar* BreathBar
UProgressBar* StaminaBar
UProgressBar* BatteryBar
UImage* BatteryImage

float MaxHealth
float MaxStamina
float MaxBreath
float MaxBattery

bool FirstHide
float FirstHideTimer
float HealthBarHideTimer
float HealthBarHideingTimer = 0
bool CanHide

void SetMaxValues(const float Health, const float Breath, const float Stamina, const float Battery)

void SetCurrentValues(const float Health, const float Breath, const float Stamina, const float Battery)

virtual void NativeTick(const FGeometry& MyGeometry, float InDeltaTime) override;

void CheckFirstHide(const float InDeltaTime)

void CheckHealthBarHide(const float InDeltaTime)

void UpdateHealthBar(const float CurrentHealth)

void UpdateBreathBar(const float CurrentBreath) const

void UpdateStaminaBar(const float CurrentStamina) const

void UpdateBatteryBar(const float CurrentBattery) const

void HideHealthBar() const
void HideBreathBar() const
void HideStaminaBar() const
void HideBatteryBar() const

void ShowHealthBar()
void ShowBreathBar() const
void ShowStaminaBar() const
void ShowBatteryBar() const

UDeathWidget : UUserWidget

TSubclassOf<UUserWidget> LoadingScreenWidget

UMenuButton* RetryButton
UMenuButton* MainMenuButton
UMenuButton* QuitButton
FName MainMenuName

virtual void NativeConstruct() override

void Retry()
void ShowLoading() const
void MainMenu()
void QuitGame()

UEndLevelWidget : UUserWidget

UTextBlock* LeveleName
UTextBlock* TimeTaken
UTextBlock* ItemsPickedUp
UTextBlock* ItemsInLevel
UTextBlock* EnemiesKilled
UTextBlock* EnemiesInLevel

FString NextLeveName

UMenuButton* NextLeveButton

virtual void NativeConstruct() override

void NextLevel()
void FillInfo(FString NextLevel)
static FText FormatTime(float TimeInSeconds)
static FText InToText(int Value)

UPauseWidget : UUserWidget

UMenuButton* ContinueButton
UMenuButton* OptionsButton
UMenuButton* MainMenuButton
UMenuButton* QuitButton
UWidgetSwitcher* WidgetSwitcher
UOptionsWidget* OptionsWidget
FName MainMenuLeveleName

virtual void NativeConstruct() override

void Continue()
void Options()
void MainMenu()
void Quit()

UMainMenuWidget : UMenu3DWidget

TSubclassOf<UUserWidget> LoadingScreenWidget

const UMySaveGame* SaveGame
FName FirstLeveleName

UMenuButton* NewGameButton
UOverwritingGameWidget* OverwritingGameBox
UMenuButton* ContinueButton
USavedGameInfoWidget* ContinueInfoBox
UMenuButton* MultiplayerButton
UMenuButton* OptionsButton
UMenuButton* ExtraButton
UMenuButton* CreditsButton
UMenuButton* QuitGameButton

virtual void NativeConstruct() override

void SetButtonsStyle()

void StartNewGame()

void OpenLevel(FName LeveToOpen) const
FGameDataStruct GetGameData() const

void HideOverwrite()
void ShowLoading() const

void NewGameButtonPressed()

void ContinueButtonHovered()
void ContinueButtonUnhovered()
void ContinueButtonPressed()

void MultiplayerButtonHovered()
void MultiplayerButtonPressed()

void OptionsButtonHovered()
void OptionsButtonPressed()

void ExtraButtonHovered()
void ExtraButtonPressed()

void CreditsButtonHovered()
void CreditsButtonPressed()

void QuitGameButtonHovered()
void QuitGameButtonPressed()

UMenuButton : UUserWidget

UButton* Button
UImage* Background
UTextBlock* TextBlock

FText Text
FLinearColor HoveredColor
FLinearColor UnhoveredColor
FLinearColor DisabledColor

virtual void NativePreConstruct() override
virtual void NativeConstruct() override
void HoveredButton()
void UnhoveredButton()
void DisabledButton()

USavedGameInfoWidget : UUserWidget

UTextBlock* TimeText
UTextBlock* LeveNameText
UImage* BackgroundImage

void FillInfoBox(const UMySaveGame* SaveGame) const

UOverwritingGameWidget : UUserWidget

UMenuButton* ConfirmNewGameButton
UMenuButton* CancelButton
USavedGameInfoWidget* InfoWidget

INVENTORY

UInventoryWidget : UUserWidget

UInventoryTab* WeaponsTab
UInventoryTab* UtilitiesTab
UInventoryTab* SecretsTab

virtual void NativeConstruct() override

void CheckItems(const FItemStruct& Item)

UInventoryTab : UUserWidget

USCrollBox* ScrollBox
UVerticalBox* Slots
TSubclassOf<UInventorySlot> InventorySlotClass

float SlotWidth
float SlotHeight

void AddItem(TSubclassOf<AItem> ItemSubclass, int Quantity)

void RemoveItem(const UInventorySlot* SlotToRemove) const

UInventorySlot* SearchSlot(const FItemStruct& Item) const

UInventorySlot : UUserWidget

UButton* SlotButton
UTextBlock* ItemNameText
FText ItemName
UImage* ItemIconImage
UTextBlock* ItemQuantityText
int ItemQuantity
TSubclassOf<AItem> ItemRef
UInventoryTab* InventoryTabRef

virtual void NativeConstruct() override

void OnCreateNewSlot(const FItemStruct& Item)

void AddItem(const AItem* Item, int Quantity)

void RemoveItem()
FText GetItemName() const

void ItemSelected()
UInventoryTab* GetInventoryTab() const

int GetItemQuantity() const

UOptionsWidget : UUserWidget

USlider* MasterSlider
USlider* MusicSlider
USlider* EffectsSlider
USlider* UISlider
USlider* DialoguesSlider

UTextBlock* UIText
UTextBlock* EffectsText
UTextBlock* MusicText
UTextBlock* DialoguesText
UTextBlock* MasterText

USoundMix* MasterMix
USoundMix* MusicMix
USoundMix* EffectsMix
USoundMix* UIMix
USoundMix* DialoguesMix

USoundClass* MasterClass
USoundClass* MusicClass
USoundClass* EffectsClass
USoundClass* UIClass
USoundClass* DialoguesClass

UCheckBox* CameraYAxisCheckBox

UMenuButton* SaveButton
UMenuButton* BackButton
FOptions Options
UMySaveOptions* SavedOptions

virtual void NativeConstruct() override

void Retry()
void ShowLoading() const
void MainMenu()
void QuitGame()