Airlock System

Unreal Engine Plugin

Github repo

Description:

The Airlock System plugin for Unreal Engine is a purely code based module that adds a simple and modular Airlock System for you to customize or use as is in your project. This documentation explains what the code does and why and how to use it.

Note:

This plugin was made by a Junior Game Programmer contributing in the Hiraeth project. Hiraeth is a Sci-fi survival game where the player is stuck alone on an unknown planet, made on the Unreal Engine 5.1, by a group of volunteers wanting to gain experience and create something they like in their freetime. This document is specific about the C++ version of the plugin, although the Blueprint version is practically the same at 99%, only class names are different and really minor changes have been made during the C++ development. Also, the code is fairly commented and should be easy to know what is happening by just looking at the header files.

Overview:

The Airlock System is pretty simple to grasp. There are 12 classes: 2 Actors, and 10 Widgets. 3 classes are the most important because they implement and manage the others, the AirlockModuleController is the central one, it manages the AirlockDoors that are linked to it and sends its informations to the ModulePanel widget that will display the relevant informations to the player.

How to add the plugin:

To interact with widgets in the 3D environment, you will need to add a WidgetInteractionComponent to your PlayerCharacter and bind a new Input Action to it. Also, you may need to change the AIRLOCSYSTEM_API between the class keyword and the class name by your project name _api (example: if you project is named SuperDeathRPG, replace AIRLOCKSYSTEM_API by SUPERDEATHRPG_API).
Check the Youtube tutorials for more in depth

Legend:

Type Property Function Delegate Specifier Class/Struct/Enum

Enums

EDoorType : uint8

BlueprintType
EDoorType represents the type of door the door is. There are 3 types that you can extend if you need your own custom types. EDoorType is used to know what animations to play when opening and closing the door. If the door is ERotatingDT, the door will rotate on its Z axis when opening and closing. If it's EElevatingDT, the door will translate its Z position, and Y position if it's ESlidingDT.

- ERotatingDT
- ESlidingDT
- EElevatingDT

EModuleUnlockMethod : uint8

BlueprintType
EModuleUnlockMethod represents how the module is supposed to be unlocked. There are 3 methods that you can extend if you need your own custom method. ETimeUM means the module will be unlocked automatically at the end of a timer, EButtonUM means the module is unlocked by pressing a button and ECodeUM needs a code to be entered to unlock it.

- ETimeMUM
- EButtonMUM
- ECodeMUM

EModuleState : uint8

BlueprintType
This enum is not used, it was to simplify the different states possible of the module but scrapped.

Classes

AirlockDoorBase : The door

Abstract, Blueprintable, BlueprintType

AirlockDoorBase is the base abstract Actor class that implements a StaticMeshComponent for showing the door in the world, a BoxCollisionComponent that detects the player where overlapping and a OpenButton widget placed on its knob.
This class declares and implements a dynamic multicast delegate: FDoorDelegate.
The door uses 8 delegates, for each event that occurs during its existence: 4 for opening and 4 for closing: Try, Start, Update and End.
Each delegate is broadcasted by a BlueprintNativeEvent function, overridable in C++ and Blueprint. Bind your custom events to these delegates to manufacture your custom system.
At last, the door needs a CurveFloat to provide the Timeline animation,this float is meant to go from 0 to 1, it will interpolate the starting position/rotation to the ending position/rotation.

- **Public:**
  - UStaticMeshComponent* DoorMesh *EditDefaultsOnly, BlueprintReadWrite*
  - UWidgetComponent* OpenButton *EditDefaultsOnly, BlueprintReadWrite*
  - UBoxComponent* Collider *EditDefaultsOnly, BlueprintReadOnly*
  - AirlockModuleController* ModuleOwner *BlueprintReadWrite*
  - FDoorDelegate TryOpenDoor_Delegate *BlueprintCallable, BlueprintAssignable*
  - FDoorDelegate StartOpenDoor_Delegate *BlueprintCallable, BlueprintAssignable*
  - FDoorDelegate UpdateOpenDoor_Delegate *BlueprintCallable, BlueprintAssignable*
  - FDoorDelegate EndOpenDoor_Delegate *BlueprintCallable, BlueprintAssignable*
  - FDoorDelegate TryCloseDoor_Delegate *BlueprintCallable, BlueprintAssignable*
  - FDoorDelegate StartCloseDoor_Delegate *BlueprintCallable, BlueprintAssignable*
  - FDoorDelegate UpdateCloseDoor_Delegate *BlueprintCallable, BlueprintAssignable*
  - FDoorDelegate EndCloseDoor_Delegate *BlueprintCallable, BlueprintAssignable*
  - void DestroyOpenButton *BlueprintCallable*
  - EDoorType GetDoorType *BlueprintCallable, BlueprintPure*
  - bool WantToOpen *BlueprintCallable, BlueprintPure*
  - bool IsOpen *BlueprintCallable, BlueprintPure*
- **Protected:**
  - FTimeline DoorTL
  - float AnimationValueStart *EditAnywhere*

    - float AnimationValueGoal *EditAnywhere, BlueprintReadWrite*
    - void OnColliderBeginOverlap *BlueprintNativeEvent*
    - void OnColliderEndOverlap *BlueprintNativeEvent*
    - virtual void CloseDoorOnEndOverlap
    - void OnTryOpenDoor_Event *BlueprintNativeEvent, BlueprintCallable*
    - void OnStartOpenDoor_Event *BlueprintNativeEvent, BlueprintCallable*
    - void OnUpdateOpenDoor_Event *BlueprintNativeEvent, BlueprintCallable*
    - void OnEndOpenDoor_Event *BlueprintNativeEvent, BlueprintCallable*
    - void OnTryCloseDoor_Event *BlueprintNativeEvent, BlueprintCallable*
    - void OnStartCloseDoor_Event *BlueprintNativeEvent, BlueprintCallable*
    - void OnUpdateCloseDoor_Event *BlueprintNativeEvent, BlueprintCallable*
    - void OnEndCloseDoor_Event *BlueprintNativeEvent, BlueprintCallable*
- **Private:**
    - bool wantToOpen *VisibleAnywhere*
    - bool isOpen *VisibleAnywhere*
    - EDoorType DoorType *EditAnywhere*
    - UCurveFloat* AnimCurve *EditAnywhere*
    - ETimelineDirection TLDirection *BlueprintReadWrite*
    - void InitDoorButton
    - void UpdateTimeline *BlueprintCallable*
    - void RotateDoor (float)
    - void SlideDoor (float)
    - void ElevateDoor (float)
    - void EndTimeline *BlueprintCallable*
    - void InitTL

AirlockModuleControllerBase : The central controller

Abstract, Blueprintable, BlueprintType

AirlockModuleControllerBase is the base abstract Actor class for the controller entity that manages the doors linked to it, and transmits its state to the panel widget.
It declares and implements a dynamic multicast delegate: FModuleDelegate. The Module controller uses 9 delegates for different events that occur during its existence: Door open and close, module locked, try to unlock, fail or success of the unlock and start, update then end the module pressurization. Each of these delegates is broadcasted by a BlueprintNativeEvent, overridable in C++ and Blueprint. Bind your custom events to these delegates to manufacture your custom system.

- **Public:**
  - bool canBeOpen *VisibleAnywhere, BlueprintReadWrite*
  - bool ShowPressurizationProgressBar *EditAnywhere, BlueprintReadWrite*
  - bool ShowPressurzationTimer *EditAnywhere, BlueprintReadWrite*
  - bool codeAutoValidate *EditAnywhere, BlueprintReadOnly*
  - TArray<AirlockDoorBase*> Doors *EditInstanceOnly, BlueprintReadWrite*
  - EModuleUnlockMethod UnlockMethod *EditAnywhere, BlueprintReadWrite*
  - float PressurizationTime *EditAnywhere, BlueprintReadWrite, meta =(min= 0))*
  - bool AutoPressurize *EditAnywhere, BlueprintReadWrite*
  - FModuleDelegate DooOpen_Delegate *BlueprintCallable, BlueprintAssignable*
  - FModuleDelegate DooClose_Delegate *BlueprintCallable, BlueprintAssignable*
  - FModuleDelegate Locked_Delegate *BlueprintCallable, BlueprintAssignable*
  - FModuleDelegate StartPressurization_Delegate *BlueprintCallable, BlueprintAssignable*
  - FModuleDelegate UpdatePressurization_Delegate *BlueprintCallable, BlueprintAssignable*
  - FModuleDelegate EndPressurization_Delegate *BlueprintCallable, BlueprintAssignable*
  - FModuleDelegate TryUnlock_Delegate *BlueprintCallable, BlueprintAssignable*
  - FModuleDelegate UnlockFailed_Delegate *BlueprintCallable, BlueprintAssignable*
  - FModuleDelegate Unlocked_Delegate *BlueprintCallable, BlueprintAssignable*
  - float GetLockTime *BlueprintCallable, BlueprintPure*
  - FText GetCode *BlueprintCallable, BlueprintPure*
  - bool IsLocked *BlueprintCallable, BlueprintPure*
  - void TriggerUnlockedEvent *BlueprintCallable*
  - void TriggerTryUnlockEvent *BlueprintCallable*
  - void TriggerFailUnlockEvent *BlueprintCallable*

- **Protected:**
  - UStaticMeshComponent* PanelMesh *VisibleAnywhere, BlueprintReadWrite*
  - USceneComponent* LightsPivot *VisibleAnywhere, BlueprintReadOnly*
  - USpotLightComponent* Light1 *VisibleAnywhere, BlueprintReadOnly*
  - USpotLightComponent* Light2 *VisibleAnywhere, BlueprintReadOnly*
  - UWidgetComponent* PanelWidget *VisibleAnywhere, BlueprintReadOnly*
  - bool isLocked *VisibleAnywhere*
  - bool isOpen *VisibleAnywhere*
  - bool canBeUnlocked *VisibleAnywhere*
  - bool isPressurizing *VisibleAnywhere*
  - float PressurizationRemainingTime *VisibleAnywhere, BlueprintReadWrite*
  - FText code *EditAnywhere, BlueprintReadOnly*
  - float lightRotationSpeed *EditAnywhere, BlueprintReadOnly*
  - void OnDoorOpen_Event *BlueprintNativeEvent, BlueprintCallable*
  - void OnDoorClosed_Event *BlueprintNativeEvent, BlueprintCallable*
  - void OnLockeed_Event *BlueprintNativeEvent, BlueprintCallable*
  - void OnStartPressurization_Event *BlueprintNativeEvent, BlueprintCallable*
  - void OnUpdatePressurization_Event *BlueprintNativeEvent, BlueprintCallable*
  - void OnEndPressurization_Event *BlueprintNativeEvent, BlueprintCallable*
  - void OnTryUnlock_Event *BlueprintNativeEvent, BlueprintCallable*
  - void OnFailUnclock_Event *BlueprintNativeEvent, BlueprintCallable*
  - void OnUnlocked_Event *BlueprintNativeEvent, BlueprintCallable*
- **Private:**
  - void BindOnDoorOpen
  - void BindOnDoorClosed
  - void TriggerStartPressurizationEvent *BlueprintCallable*

UWAirlockModulePanel : The interactive widget

Blueprintable, BlueprintType

AirlockModulePanel is the user widget class that displays relevant information about the state of the module. It's the widget used by the AirlockModuleController WidgetComponent that is interactable with the player for pressing buttons. It's in charge of binding differents events and show differents widgets according to some properties of the AirlockModuleController.

- **Public:**
  - void InitPanel *BlueprintCallable, BlueprintNativeEvent*
  - void ModuleTriggerTryUnlock
  - void ModuletriggerFailUnlock
  - bool GetShowPressurizationBar *BlueprintCallable, BlueprintPure*
  - bool GetShowPressurizationTimer *BlueprintCallable, BlueprintPure*
  - FText GetCode *BlueprintCallable, BlueprintPure*

    - AirlockModuleControllerBase* GetModuleOwner *BlueprintCallable, BlueprintPure*
    - AirlockModuleUnlockButton* GetUnlockButton *BlueprintCallable, BlueprintPure*
    - StartPressurizationButton* GetPressurizeButton *BlueprintCallable, BlueprintPure*
- **Protected:**
    - FText code *VisibleAnywhere, BlueprintReadWrite*
    - EModuleUnlockMethod ModuleUnlockMethod *VisibleAnywhere, BlueprintReadWrite*
    - bool ShowPressurizationBar *VisibleAnywhere, BlueprintReadWrite*
    - bool ShowPressurizationTimer *VisibleAnywhere, BlueprintReadWrite*
    - UTextBlock* DisplayText *EditAnywhere, BlueprintReadOnly, meta = (BindWidget, DesignerRebuild)*
    - UUWPressurizationProgress* PressurizationProgress *EditAnywhere, BlueprintReadOnly, meta = (BindWidget, DesignerRebuild)*
    - UWidgetSwitcher* PanelSwitcher *EditAnywhere, BlueprintReadOnly*
    - UUWStartPressurizationButton* PressurizeButton *EditAnywhere, BlueprintReadOnly, meta = (BindWidget, DesignerRebuild)*
    - UUWAirlockModuleUnlockButton* UnlockButton *EditAnywhere, BlueprintReadOnly, meta = (BindWidget)*
    - UUWNumPad* NumPad *EditAnywhere, BlueprintReadOnly, meta = (BindWidget)*
    - UVerticalBox* VerticalBox *EditAnywhere, BlueprintReadOnly, meta = (BindWidget, DesignerRebuild)*
    - AirlockModuleControllerBase* ModuleOwner *EditAnywhere, BlueprintReadWrite*
    - void OnDoorOpen *BlueprintCallable, BlueprintNativeEvent*
    - void OnDoorClosed *BlueprintCallable, BlueprintNativeEvent*
    - void OnModuleLocked *BlueprintCallable, BlueprintNativeEvent*
    - void OnStartPressurize *BlueprintCallable, BlueprintNativeEvent*
    - void OnPressurizationUpdate *BlueprintCallable, BlueprintNativeEvent*
    - void OnEndPressurization *BlueprintCallable, BlueprintNativeEvent*
    - void UnlockByTime *BlueprintCallable, BlueprintNativeEvent*
    - void DisplayUnlockButton *BlueprintCallable, BlueprintNativeEvent*
    - void DisplayNumPad *BlueprintCallable, BlueprintNativeEvent*
    - void OnTryUnlock *BlueprintCallable, BlueprintNativeEvent*
    - void TimerFailure *BlueprintCallable, BlueprintNativeEvent*
    - void ButtonFailure *BlueprintCallable, BlueprintNativeEvent*
    - void CodeFailure *BlueprintCallable, BlueprintNativeEvent*
    - void OnUnlockSuccess *BlueprintCallable, BlueprintNativeEvent*
- **Private:**
    - UCanvasPanel* CanvasPanel
    - void BindDelegates
    - void InitProperties

UWAirlockModuleUnlockButton

Blueprintable, BlueprintType

Widget of the button that will Unlock the module when pressed. It's used when the module unlock method is Button and appears at the end of the pressurization.

- **Public:**
  - UButton* GetButton *BlueprintCallable, BlueprintPure*
- **Protected:**
  - UButton* Button *EditAnywhere, BlueprintReadOnly, meta = (BindWidget, DesignerRebuild)*
  - UTextBlock* Text *EditAnywhere, BlueprintReadOnly, meta = (BindWidget, DesignerRebuild)*

UWNumPad

BlueprintType, Blueprintable

The Numpad assembles a multitude of buttons for the player to enter a code in order to unlock the module. The numpad is shown at the end of the pressurization on the AirlockModulePanel when the unlock method is Code. The NumPad could be extended to be a full keyboard if you want other symbols than numbers.

- **Public:**
  - bool AutoValidate *VisibleAnywhere, BlueprintReadWrite*
  - FText GetCode *BlueprintCallable, BlueprintPure*
  - void InitPad (UUWAirlockModulePanel*) *BlueprintCallable, BlueprintNativeEvent*
  - void DeleteCode *BlueprintCallable, BlueprintNativeEvent*
  - void EnterCode (const FText&) *BlueprintCallable, BlueprintNativeEvent*
  - void ValidateCode *BlueprintCallable, BlueprintReadWrite*
- **Protected:**
  - ESlateVisibility GetValidateButtonVisibility
- **Private:**
  - FText currentCode
  - FText correctCode
  - UUWAirlockModulePanel* PanelOwner
  - UUWNumPadEntryButton* Button0 *EditAnywhere, BlueprintReadOnly, meta= (BindWidget, DesignerRebuild, AllowPrivateAccess = true)*
  - UUWNumPadEntryButton* Button1 *EditAnywhere, BlueprintReadOnly, meta= (BindWidget, DesignerRebuild, AllowPrivateAccess = true)*
  - UUWNumPadEntryButton* Button2 *EditAnywhere, BlueprintReadOnly, meta= (BindWidget, DesignerRebuild, AllowPrivateAccess = true)*
  - UUWNumPadEntryButton* Button3 *EditAnywhere, BlueprintReadOnly, meta= (BindWidget, DesignerRebuild, AllowPrivateAccess = true)*
  - UUWNumPadEntryButton* Button4 *EditAnywhere, BlueprintReadOnly, meta= (BindWidget, DesignerRebuild, AllowPrivateAccess = true)*
  - UUWNumPadEntryButton* Button5 *EditAnywhere, BlueprintReadOnly, meta= (BindWidget, DesignerRebuild, AllowPrivateAccess = true)*
  - UUWNumPadEntryButton* Button6 *EditAnywhere, BlueprintReadOnly, meta= (BindWidget, DesignerRebuild, AllowPrivateAccess = true)*
  - UUWNumPadEntryButton* Button7 *EditAnywhere, BlueprintReadOnly, meta= (BindWidget, DesignerRebuild, AllowPrivateAccess = true)*

- UUWNumPadEntryButton* Button8 *EditAnywhere, BlueprintReadOnly, meta= (BindWidget, DesignerRebuild, AllowPrivateAccess = true)*
- UUWNumPadEntryButton* Button9 *EditAnywhere, BlueprintReadOnly, meta= (BindWidget, DesignerRebuild, AllowPrivateAccess = true)*
- TArray<UUWNumPadEntryButton*> EntryButtons *EditAnywhere, BlueprintReadWrite, meta = (BindWidget, DesignerRebuild, AllowPrivateAccess = true)*
- UUWNumPadDeleteButton* DeleteButton *EditAnywhere, BlueprintReadWrite, meta = (BindWidget, DesignerRebuild, AllowPrivateAccess = true)*
- UUWNumPadValidateButton* ValidateButton *meta = (BindWidget, DesignerRebuild, AllowPrivateAccess = true)*
- UTextBlock* DisplayText *EditAnywhere, BlueprintReadWrite, meta = (BindWidget, DesignerRebuild, AllowPrivateAccess = true)*
- UUniformGridPanel* Grid *meta = (BindWidget)*
- UVerticalBox* VerticalBox *meta = (BindWidget)*

UWNumPadButtonBase

Abstract, Blueprintable, BlueprintType

The NumPadButtonBase is the base abstract user widget class for the derived NumPadButton that will be assembled in the NumPad to create a keyboard of number for the player to use to enter, delete and validate a code to unlock the module.

- **Public:**
  - FText TextToDisplay *EditAnywhere, BlueprintReadWrite*
  - UUWNumPad* NumPadOwner *VisibleAnywhere, BlueprintReadWrite*
- **Protected:**
  - UButton* Button *VisibleAnywhere, BlueprintReadOnly, meta = (BindWidget, DesignerRebuild)*
  - UTextBlock* DisplayText *EditAnywhere, BlueprintReadOnly, meta = (BindWidget, DesignerRebuild)*
  - void OnButtonClicked *BlueprintCallable, BlueprintNativeEvent*

UWNumPadDeleteButton

Blueprintable, BlueprintType

Derived class of UWNumPadButtonBase that implements the OnButtonClicked method to call the DeleteCode method of its owning NumPad.

- **Protected:**
  - virtual void OnButtonClicked_Implementation override

UWNumPadEntryButton

Blueprintable, BlueprintType

Derived class of UWNumPadButtonBase that implements the OnButtonClicked method to call the EnterCode method of its owning NumPad with it's TextToDisplay's text as the parameter.

- **Protected:**
  - virtual void OnButtonClicked_Implementation override

UWNumPadValidateButton

Blueprintable, BlueprintType

Derived class of UWNumPadButtonBase that implements the OnButtonClicked method to call the ValidateCode method of its owning NumPad.

- **Protected:**
  - virtual void OnButtonClicked_Implementation override

UWOpenDoorButton

Blueprintable, BlueprintType

Userwidget class of a Button that is attached to the AirlockDoorBase's knob. If the owning AirlockModuleController is AutoPressurize, this button is deleted from the door. The UWOpenDoorButton is used to prevent the module to stay locked forever if the player steps out just before the module locks when it needs to be manually pressurized from the inside. It could use the button to unlock the module then pressurize when it locks again.

- **Public:**
  - AirlockDoorBase* DoorOwner *EditAnywhere, BlueprintReadWrite*
- **Protected:**
  - void OnButtonClicked *BlueprintCallable, BlueprintNativeEvent*
- **Private:**
  - UButton* Button *EditAnywhere, BlueprintReadOnly, meta = (BindWidget, DesignerRebuild, AllowPrivateAcces = true)*
  - UTextBlock* DisplayText *EditAnywhere, BlueprintReadOnly, meta = (BindWidget, DesignerRebuild, AllowPrivateAcces = true)*

UWPressurizationProgress

Blueprintable, BlueprintType.

UserWidget that will display the progress of the pressurization from when it starts to finish.

- **Public:**
  - float timer *VisibleAnywhere, BlueprintReadWrite*
  - float GetCurrentTime*BlueprintCallable, BlueprintPure*
  - void InitPressurizationProgress (UWAirlockModulePanel*) *BlueprintCallable, BlueprintNativeEvent*
  - void UpdateTimer (float) *BlueprintCallable*
- **Protected:**
  - UWAirlockModulePanel* PanelOwner *VisibleAnywhere*
  - float GetPercent *BlueprintCallable, BlueprintPure*
  - FText GetCurrentTimerText *BlueprintCallable, BlueprintPure*
  - FLinerColor GetColor *BlueprintCallable, BlueprintNativeEvent, BlueprintPure*
  - ESlateVisibility GetBarVisibility *BlueprintCallable, BlueprintPure*
  - ESlateVisibility GetTextVisibility *BlueprintCallable, BlueprintPure*
- **Private:**
  - UProgressBar* ProgressBar *EditAnywhere, meta = (BindWidget, DesignerRebuild)*
  - UTextBlock* TimerText *EditAnywhere, meta = (BindWidget, DesignerRebuild)*
  - bool showProgressBar *VisibleAnywhere*
  - bool showProgressTimer *VisibleAnywhere*
  - float PresurrizationRemainingTime *VisibleAnywhere*
  - UCanvasPanel* CanvasPanel *VisibleAnywhere, meta = (BindWidget, DesignerRebuild)*

UWStartPressurizationButton

Blueprintable, BlueprintType

UserWidget that will show if the owning AirlockModuleController does not autopressurize, that means that the player has to manually trigger the pressurization by pressing this button.

- **Public:**
  - UButton* GetButton *BlueprintCallable, BlueprintPure*
  - UTextBlock* GetDisplayText *BlueprintCallable, BlueprintPure*
- **Private:**
  - UButton* Button *EditAnywhere, meta = (BindWidget, DesignerRebuild)*
  - UTextBlock* DisplayText *EditAnywhere, meta = (BindWidget, DesignerRebuild)*
  - void OnButtonClicked *BlueprintCallable*