Nekatus Doors / Buttons / Indicators (DBI) Project.

- There are three object types… Doors, Buttons and Indicators.
- The door and indicator are directly derived from the “Actor” class.
- The button is derived from the Ark “PrimalItemStructure” class, as that class provides the functionality for interactivity, which is only required on the button.
- The **three** main blueprints are:
    - “**BP_Nekatus_Door_01**” for the door.
    - “**BP_Nekatus_Button_01**” for the button.
    - “**BP_Nekatus_Indicator_01**” for the indicator.
- There other relevant files:
    - “**Interface_Nekatus_BDI**” which contains the blueprint interfaces for the project. It’s located in the “_Interfaces” folder.
    - “**Functions_Nekatus_BDI**” which contains the function library for the project. It’s located in the “_Scripts” folder.
    - “**NK_Buttons_Struct**” and “**NK_Indicator_Struct**” in the “_Structs” folder, which contain data structures used in the blueprints.
- These will be grouped together into a “**Network**”, which would include all the connected objects inside one cave, for example.
- Each network can only have a single door, but can have as many buttons or indicators as required.
- There can be as many networks as required on a map.
- The network is configured by setting the “**NetworkTag**” variable on the Door, and by configuring that same variable on each instance of the Buttons and Indicators. For example, you might want a network called “IceCave”. You would set the “NetworkTag” variable on the door to “IceCave”, then on each button and indicator you place, you set that same variable to the value “IceCave”. This allows the door to find the connected buttons and indicators for that network.
- Buttons and Indicators can also have an assigned group using the “**GroupID**” variable. If a button and an indicator have the same GroupID, then that indicator will activate when the button is pressed, then reset itself after the number of seconds it’s configured for. Any number of buttons and indicators can be in the same group.
- There are no hard links required when placing the objects. Each door will automatically find its connected buttons and indicators using the network tag. It does this once each time the server starts. There will be a single, small time penalty for each network at startup while the server finds the connected buttons and indicators. Only a fraction of a second. This should not cause any noticeable impact.

BP_Nekatus_Door_01

Function: The door acts as both the controller of the network, and as an actual door, with a moveable mesh. It receives interface calls from buttons, and sends interface calls to indicators.

Config Variables:

- **NetworkTag (Name)** - Used to configure the name of the network the door belongs to. This is used to search for buttons and indicators with a matching Actor Tag. Using Actor Tags allows for slightly more efficient searching.
- **OpenLocation (Vector)** - This is the relative location of the door mesh when it's in the open position.
- **ClosedLocation (Vector)** - This is the relative location of the door mesh when it's in the closed position.
- **AutoCloseTime (Float)** - The number of seconds the door will wait before automatically closing.
- **DoorOpenSpeed (Float)** - The number of seconds the door will take to move from closed to open.
- **DoorCloseSpeed (Float)** - How many seconds the door will take to move from open to closed.
- **ButtonSequence (Byte Array)** - This is an array which specifies the order buttons must be pushed in order to trigger the opening of the door. Add as many buttons as required.

BP_Nekatus_Button_01

Function: The button acts as the only input to the system, allowing a user to complete an action. That action is sent to and processed by the door. When pressed, the button can move part of its mesh to a "pressed" position, and then auto-reset itself after a configurable amount of time. Of course this can be changed as required.

Config Variables:

- **NetworkTag (Name)** - Used to configure the name of the network the door belongs to. This is used to search for buttons and indicators with a matching Actor Tag. Using Actor Tags allows for slightly more efficient searching.
- **DeviceID (Byte)** - Used by the door to determine which button has been pressed. Valid values are between 1 and 254. All buttons and indicators on a network should be unique.
- **GroupID (Byte)** - Used to tie an indicator or indicators to a particular button press. Valid values are between 1 and 254. As many buttons and indicators can share a group as required.
- **RestPosition (Vector)** - This is the relative location of the button mesh when it's in the rest position.
- **PressedPosition (Vector)** - This is the relative location of the button mesh when it's in the pressed position
- **ButtonResetTime (Float)** - This is the amount of time the button mesh will remain in the "pressed" position before returning to the rest position.

BP_Nekatus_Indicator_01

Function: The indicator acts and an output for the system. The door controls the indicator state, except for an auto-reset timer that will deactivate the indicator after a configured number of seconds.The button indicates its "active" state by changing a material parameter

to set an emissive variable on the indicator material. Of course this can be changed as required.

Config Variables:

- **NetworkTag (Name)** - Used to configure the name of the network the door belongs to. This is used to search for buttons and indicators with a matching Actor Tag. Using Actor Tags allows for slightly more efficient searching.
- **DeviceID (Byte)** - Used by the door to determine which button has been pressed. Valid values are between 1 and 254. All buttons and indicators on a network should be unique.
- **GroupID (Byte)** - Used to tie an indicator or indicators to a particular button press. Valid values are between 1 and 254. As many buttons and indicators can share a group as required.
- **DefaultResetTime (Float)** - This is the amount of time the indicator will remain active before resetting itself to the inactive state.