Open Sesame

Hello and welcome to the manual of Open Sesame asset pack. It allows you to quickly create, set up and easily maintain all kinds of functional (and **lockable**) **doors**, **windows**, trapdoors, hatches, or to provide opening/closing ability to **furniture** or **props**. An Overview level bundled within the asset pack demonstrates main features and displays some tips and ideas where you can utilize these assets in your projects.

Open sesame is created fully in blueprints with generous amount of properties, such as **type** of opening (rotating, sliding, custom), **direction**, movement **range**, **duration** and many others. Tooltips for these properties will provide directly in the UE Editor quick reference and help you to set up assets without reaching manual all the time.

Features and abilities:

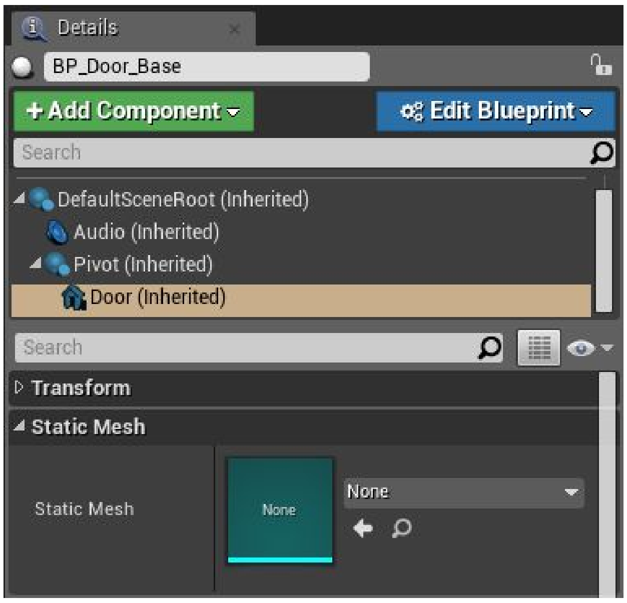
- Create common rotating doors on hinges, sliding doors or more complex custom opening with movement and rotation together
- Double (or multi) wing doors
- Adjustable behavior - how opening looks like - constant speed, rapid start and smooth finish or special effect (angry slam, ...)
- Locking system - doors can be locked and unlocked with keys (or keycards, chips, whatever)
- Basic inventory system for collecting keys
- Sounds - assign sounds to play on events opening, closing, locking, unlocking or failed unlocking. Over 40 demo sounds are included.
- Manual opening on player input or automatic on actor overlapping trigger
- Easy to set up - visual identification of opening direction and range
- Easy to extend and modify - access events from other actors or create custom blueprints that fits into the system thanks to provided interfaces

Basic door functionality (BP_Door_Base)

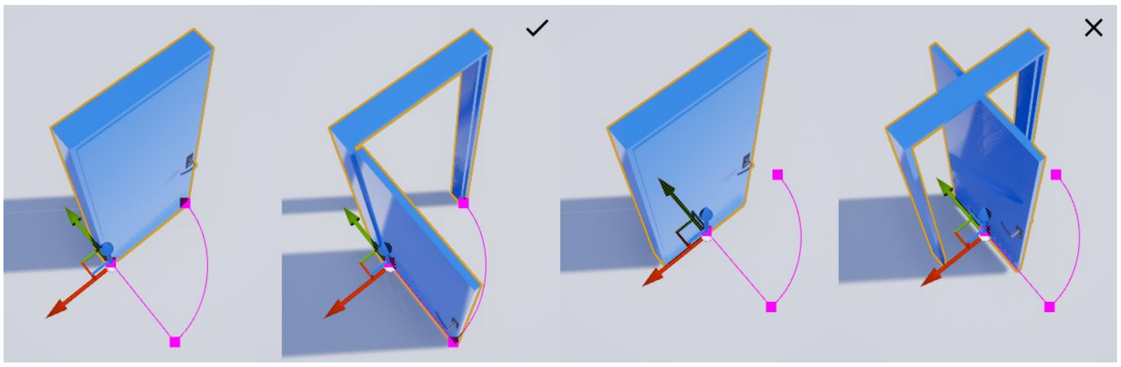
This is base blueprint for all doors. It performs actual door movement, playing sounds and visualizing door opening.

To set up a door:

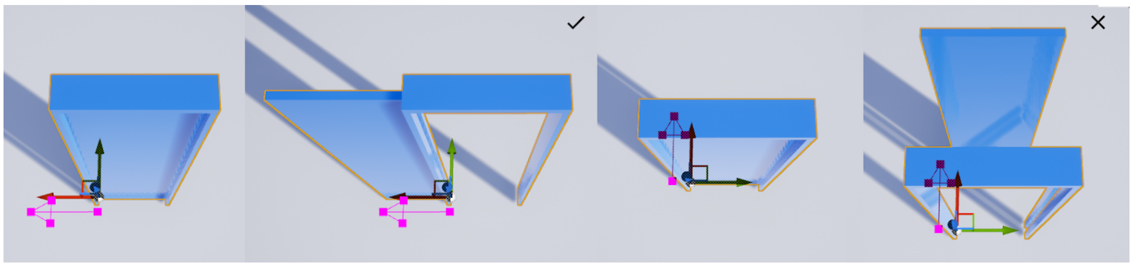
1. Start by dropping a BP_Door_Base into your level or create a Child Blueprint Class based on it.
2. Choose which type of door you are creating by setting Type Of Opening property - Rotate, Slide or Custom
3. Click on **Door** component and set a door mesh to its Static Mesh property.



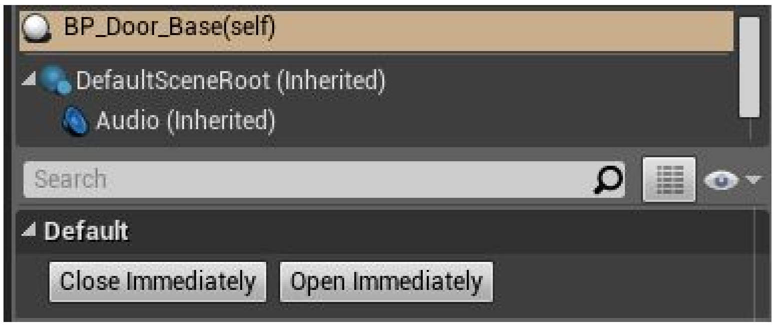
4. Door movement is performed by moving or rotating the **Pivot** scene component, so **Door** and **Pivot** need to be in correct position and rotation with respect to each other. In case of rotating door, Pivot should be on the edge with hinges. Use arc at the bottom to guide you.



For sliding door, rotation is more important. Opening and closing happens in local +X and -X axis of Pivot, so ensure that your door won't drift sideways as result of wrong orientation. Arrow at the bottom shows direction of opening movement.



If necessary, move the Door component as needed to make adjustments. You can also use **Close Immediately** and **Open Immediately** buttons to check whether the door opens correctly.



5. Set other properties - **Duration**, **Rotate/Slide amount**, reverse opening direction with **Open Anticlockwise** or **Reverse Slide Opening** checkboxes if necessary.

Total time of opening or closing movement is controlled by Duration parameter. However, you may need to control relative speed across the movement to create a different feeling of movement. It can be plain simple linear movement, or you can make a door slow down just before hitting a door frame to make smooth closing. More elaborate special effects such as a little swing back and swift shut to express "angry door slam" are also possible.

To achieve this, you can use **Behavior** or **Secondary behavior** properties, that take Float Curve values. X-axis of curve represents time and Y-axis represents movement whether it is angle for rotating door, or distance for sliding door. Curve should start at value 0 and end at value 1 on both axis. Steepness of the curve determines speed of movement - steep slope means fast, shallow means slow.

Primary behavior property (Rotating Behavior, Sliding Behavior) influence opening and also closing if Secondary behavior curve is not assigned. Use

Secondary behavior property (Secondary Rotating Behavior, Secondary Sliding Behavior) to use different curve for closing.

Custom movement has only three parameters - Duration property shared with other movement types, and **Custom Location** and **Custom Rotation** properties that take Vector Curve values. X-axis of curves represent time and Y-axis represent movement or rotation. Curves should start at value 0 and end at value 1 on X-axis. Y-axis represent actual movement values, so for example set blue Z-curve to 1 on X-axis and to 120 on Y-axis to rotate door 120 degrees at the end of opening.

To actually open a BP_Door_Base in game, use **AC_Automatic_Door**, **AC_Door_Handler** or call door's functions Open or Close from your own blueprint.

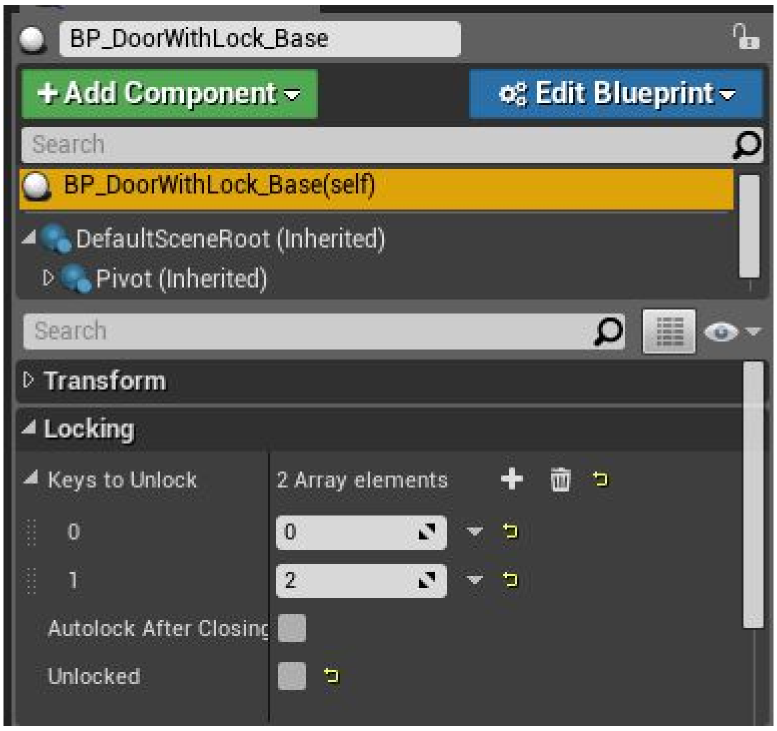
Adding lock to a door (BP_DoorWithLock_Base)

BP_DoorWithLock_Base extends basic door functionality by adding locking mechanism. Door must be unlocked with matching key types before opening, locked door will fail to open.

For the locking system, keys are just integer numbers - for example, in your project number 0 can represent gold key, 1 can be silver key and number 2 can be considered as biometric ID card. What matters is that a key provider must be able to provide all key types that a locked door requires to be able to unlock it.

To set up a locked door:

1. Set up a door according to instructions for Basic door functionality, only in the first step add a BP_DoorWithLock_Base blueprint into your level instead of BP_Door_Base.
2. In the details panel, under header Locking, find an array "Keys to Unlock" and set it to any key types that you need for door to require.
3. You may need to uncheck property "Unlocked", as default value is set to true.
4. Door with lock has also 3 more sound properties for locking, unlocking and failed unlocking. You may want to fill them.

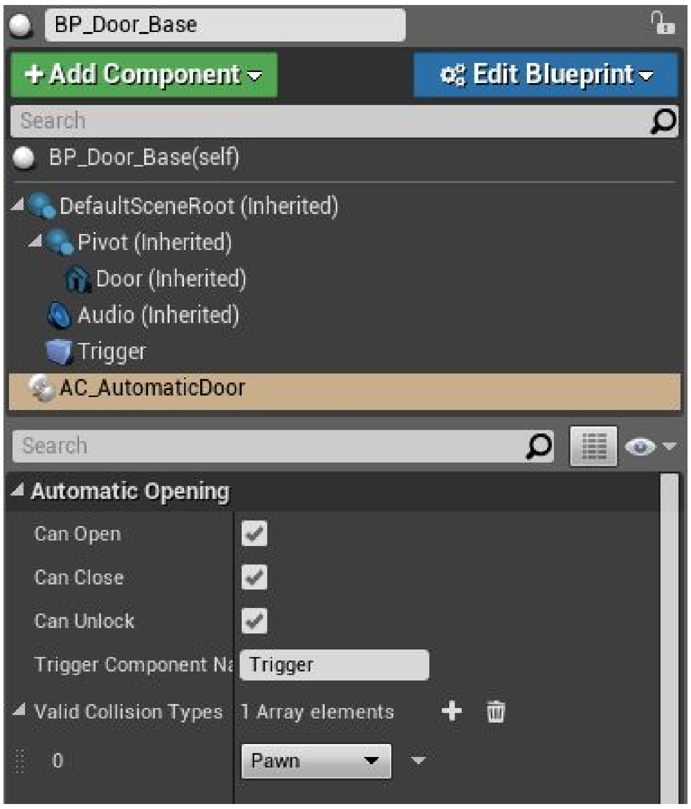


Keys to Unlock: 2 Array elements
0: 0
1: 2
Autolock After Closing
Unlocked

Automatic opening (AC_Automatic_Door)

Actor component for automatic door opening. It will open door when an actor enters its trigger volume and close when all actors leave it.

1. Add **AC_Automatic_Door** component to a door
2. Add a Box, Capsule or Sphere Collision component to the door and set its name to "**Trigger**". The system will look for any component of such type containing this keyword in its name and will identify it as the trigger volume. You can change the keyword in the Automatic door component's options if you want (property Trigger Component Name).


BP_Door_Base

BP_Door_Base(self)

DefaultSceneRoot (Inherited)

Pivot (Inherited)

Door (Inherited)

Audio (Inherited)

Trigger

AC_AutomaticDoor

Automatic Opening

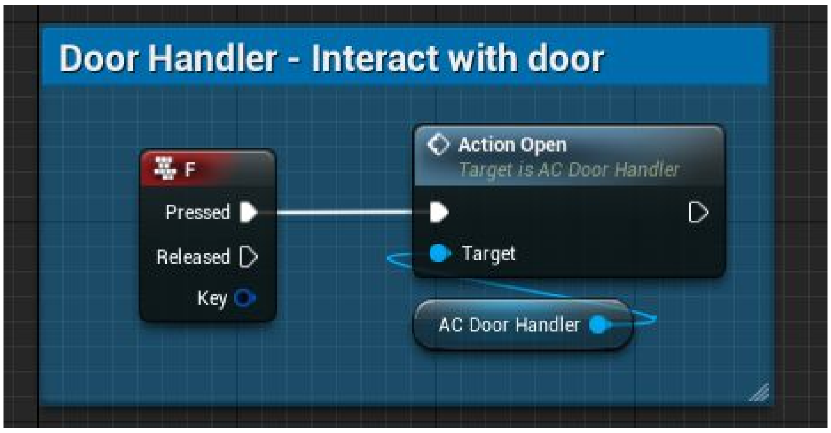
Can Open	✓
Can Close	✓
Can Unlock	✓
Trigger Component Na	Trigger
Valid Collision Types	1 Array elements
0	Pawn

3. Adjust trigger's size, location and rotation as you need.
4. By default, automatic door component will only react to actors that have collision type set to "Pawn". This ensures, that it won't be triggered by everything moving around, only by player. If you need for other object types to open and close door, edit Valid Collision Types in Automatic door component's options.

Open door with pawn (AC_DoorHandler)

Actor component for manual door opening. It detects door in vicinity of its owner and opens it when called (e.g. on player input). It can also show hints to guide player what to do to open, close or unlock the door or which keys are needed.

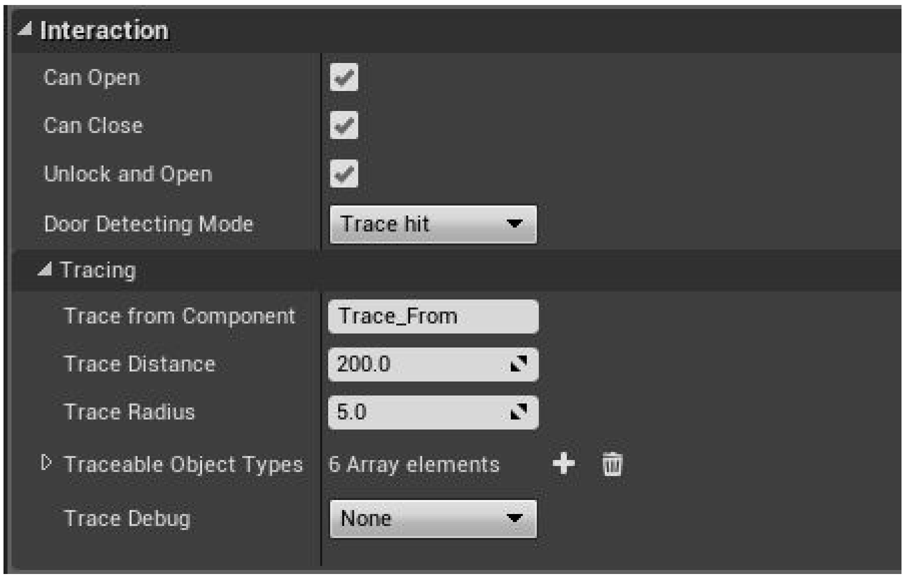
1. Add an AC_DoorHandler component to your pawn or character.
2. From SesameFirstPersonCharacter demo blueprint copy blue "Door Handler - Interact with door" part to your blueprint.



3. In AC_DoorHandler options choose **Door Detecting Mode**. It can be either **Trace hit**, when the pawn will send traces, or **Actor overlap**.

   3.1. Door detecting mode: **Trace hit**

   - Add an Arrow component to pawn and name it "Trace_From". Door handler will use this as "aiming tool" to send traces and detect doors. In fact, you can use any scene component, not just arrow, but I find it as the most suitable for the job.
   - Set it to some appropriate location. For third person character it can be chest, for first person character set it as child of camera.
   - Name "Trace_From" is not mandatory, you can use different name, but then you will need to set variable "Trace from Component" in option group Interaction->Tracing to match the new name.
   - You can visualize tracing ray by turning on “Trace Debug” property



Interaction

Can Open	✓
Can Close	✓
Unlock and Open	✓
Door Detecting Mode	Trace hit

Tracing

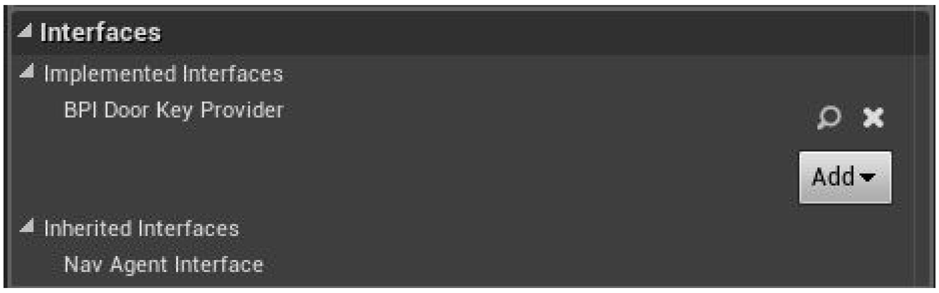
Trace from Component	Trace_From
Trace Distance	200.0
Trace Radius	5.0
Traceable Object Types	6 Array elements
Trace Debug	None

3.2. Door detecting mode: Actor overlap
- With this setting, something must be able to overlap something else, so either door, pawn or both must have some collision volume set up.

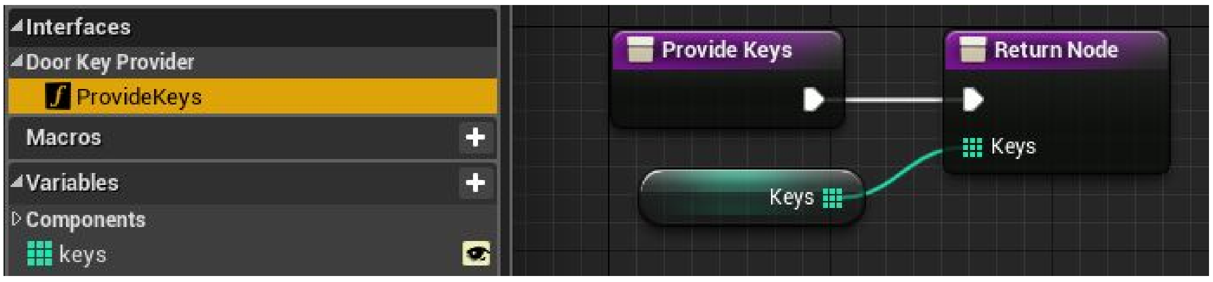
Unlock door with pawn (AC_DoorHandler)

Pawn that has been set up according to previous chapter can open and close door, but so far can not unlock it if needed. To set up a locked door, see chapter [Adding lock to a door (BP_DoorWithLock_Base)]()

To provide pawn with unlocking functionality:
1. Open your pawn's blueprint. Go to Class Settings (normally on the bar at the top of the screen) and then look at the details on the right side after clicking it. Add a BPI_DoorKeyProvider to Implemented Interfaces and compile.

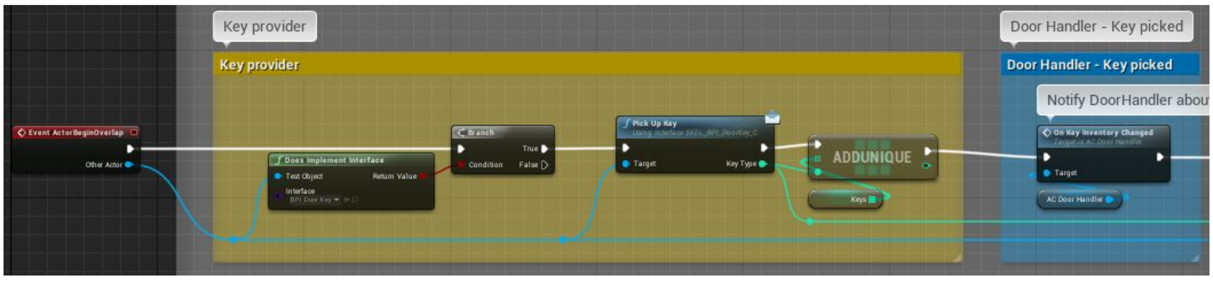


2. Create variable of type array of integers, that will store character's keys. For the locking system, keys are just integer numbers - for example, in your project number 0 can represent gold key, 1 can be silver key and number 2 can be considered as biometric ID card. What matters is that pawn must be able to provide all key types that a locked door requires to be able to unlock it.
3. Set interface function ProvideKeys to return this array.



4. From SesameFirstPersonCharacter demo blueprint copy yellow "KeyProvider" part and blue "Door Handler - Key picked" part and connect

it to event ActorBeginOverlap. This will enable pawn to collect keys and will notify Door handler about new keys in inventory.



If keys owned by pawn can't open a door, door handler will show hint telling which additional keys are needed. In order to show user proper key name and not just key type number, you can assign names to key types in data table "**KeyNamesTable**".

Keys (BP_Key_Base)

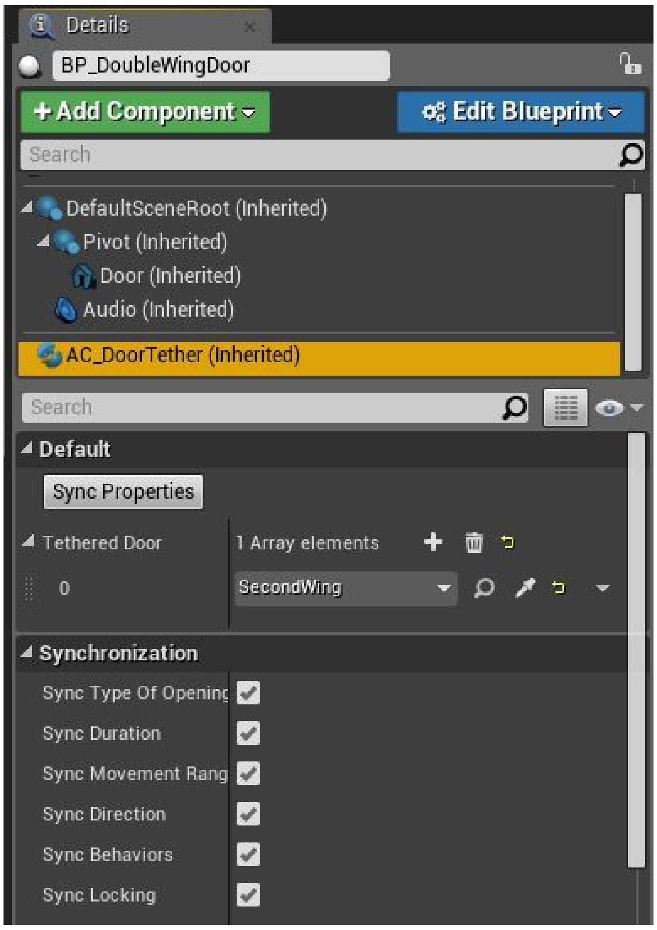
Base blueprint for representing keys. You can place it in level, pick it up with pawn and use to unlock a door.

1. Place a BP_Key_Base into your level or create a Child Blueprint Class based on it.
2. Set **Key Type** number. Key Icon is used by inventory system to show on screen after the key is picked up by player. Property "Key Name" is not used (key names are set in data table KeyNamesTable), but it's available to use for your custom blueprints that would interact with keys.
3. Click on **KeyMesh** component and set your mesh to its Static Mesh property.
4. Adjust **BoxTrigger** component's size, location and rotation as you need. Pawn's AC_Door_Handler will pick up the key when it overlaps the trigger.

Two wing door (BP_DoubleWingDoor, AC_DoorTether)

AC_DoorTether joins multiple doors, so opening or closing one door will do the same for all others. Therefore it can be used for making two wing doors.

1. Set up a door - mesh, materials, pivot, movement type etc. It will act as one (master) wing. You can use any blueprint for it - BP_Door_Base, BP_DoorWithLock_Base, their child blueprint, however the best choice would be **BP_DoubleWingDoor** template as it will save you a little work.
2. If you <u>did not</u> use BP_DoubleWingDoor, add a **AC_DoorTether** component to your door.
3. Set up a second door, that will act as second (slave) wing. Now it's necessary to set only mesh, materials and pivot, other properties can be copied from the master door. Important thing here is, that if you used lockable door as master, you need to use lockable door for slave as well.
4. Select **AC_DoorTether** component on master door and add slave door actor to **Tethered door** property.
5. Copy settings from master to slave by clicking on **Sync Properties** button in door tether component. Below, you can choose which properties will be copied. If you used BP_DoubleWingDoor, Sync settings button is also available in blueprint details.


Details
BP_DoubleWingDoor
+ Add Component
Edit Blueprint
Search
DefaultSceneRoot (Inherited)
Pivot (Inherited)
Door (Inherited)
Audio (Inherited)
AC_DoorTether (Inherited)
Search
Default
Sync Properties
Tethered Door
1 Array elements
0
SecondWing
Synchronization
Sync Type Of Opening
Sync Duration
Sync Movement Rang
Sync Direction
Sync Behaviors
Sync Locking