Magnetic Blocks

Requirements

- Magnetic blocks must be able to connect 2 entities through physical constraints.
- Magnetic blocks must detect surfaces
- Magnetic blocks must detach if the surface is no longer detected
- Multiple constraints may be present between two bodies.

Feature

The magnetic block represents landing gear and magnetic plates which allow a grid to attach to another grid or voxel surface (or any physical body).

Components

MagneticBlockComponent

Main component for the magnetic block. It is in charge of handling inputs, states and collecting detections to lock and unlock entities.

Signals

The magnetic block component provides signals to notify other components of changing states.

- **LockStateChangedSignal** provides a simple state enum for the lock. It can be “Unlocked”, “Locked” or “ReadyToLock”. This is invoked during the “UpdateConnections” job.
- **UpdateReadyToLockSignal / UpdateLockedEntitySignal** can be used to get the entity being detected or locked, respectively. Those signals can be invoked at any time.

Commands

This component listens to parking brakes input through **IParkingBrake**.

Locking / Unlocking can be requested by calling **RequestLock**, **RequestUnlock** or **RequestSwitchLock**.

The commands change the lock state only when the block is functional. The state does not change when the block loses power or is disabled but it unlocks if the block is broken or if the locked entity is no longer detected.

Auto lock

When auto lock is enabled, the magnetic block will instantaneously attach to the first detected entity. To prevent instantaneous relocking, a cooldown starts when the magnetic block unlocks and prevents the autolock from activating again until the cooldown is finished. The cooldown method is performed by the ICooldown.

Default state can be set from the object builder.

Detections / Constraints ?

This component does not detect nor create the physical constraints by itself but delegate those tasks to other components through IPhysicConstraintCreator and IEntityDetectorProvider. This allows customizing the way detection is performed and constraints are created.

The IEntityDetectorProvider is optional so it can be added only on client, server or both. Note that if this component is not present on the entity, the magnetic block will never be able to lock on anything.

MagneticBlockIndicatorComponent

This client component will override some of the model materials to indicate the state of the block.

Indicator states

- NotFunctional
- NotPowered
- TurnedOff
- Idle
- AutoLock (idle with autolock enabled)
- ReadyToLock
- Locked

MagneticBlockDetectionComponent

This server component is in charge of filtering the detections. It will ignore detected entities where the surface is not aligned with the magnetic block. It also prioritizes the currently locked entity if multiple entities are detected.

Entities must have the **MagneticBlockDetectionComponent.MagneticTag** to be seen by the magnetic block.

It does implement the IEntityDetectorProvider interface referenced by the main component but delegates the actual detection to 2 other IEntityDetectorProvider. One is used to detect a new entity when the block is not yet locked. The other is used to detect if the locked entity is still in range.

This separation allows using a different range to ensure the locked entity doesn't get out of range if their physical motions are not perfectly attached while the entity was already at the limit of the detection range.

The second detection is optional, if not provided, the first detector will be used instead.

IPhysicConstraintCreator

This interface is provided by the VRage.Physics and allows to create constraints between entities.

Magnetic block requires a strong link between the entities. **FixedPhysicsConstraintComponent** should be used to ensure the entities keep the same relative positions.

IEntityDetectorProvider

Many implementations are available:

- **RaycastEntityDetectorComponent** to detect entities using a simple raycast.
- **DummyShapedEntityDetectorComponent** to detect entities inside the dummy box.

ICooldown

- **TimeBasedCooldownComponent** to prevent autolock a fixed duration after unlocking.

- **DetectionBasedCooldownComponent** to prevent autolock while entities are still in range.

Screens

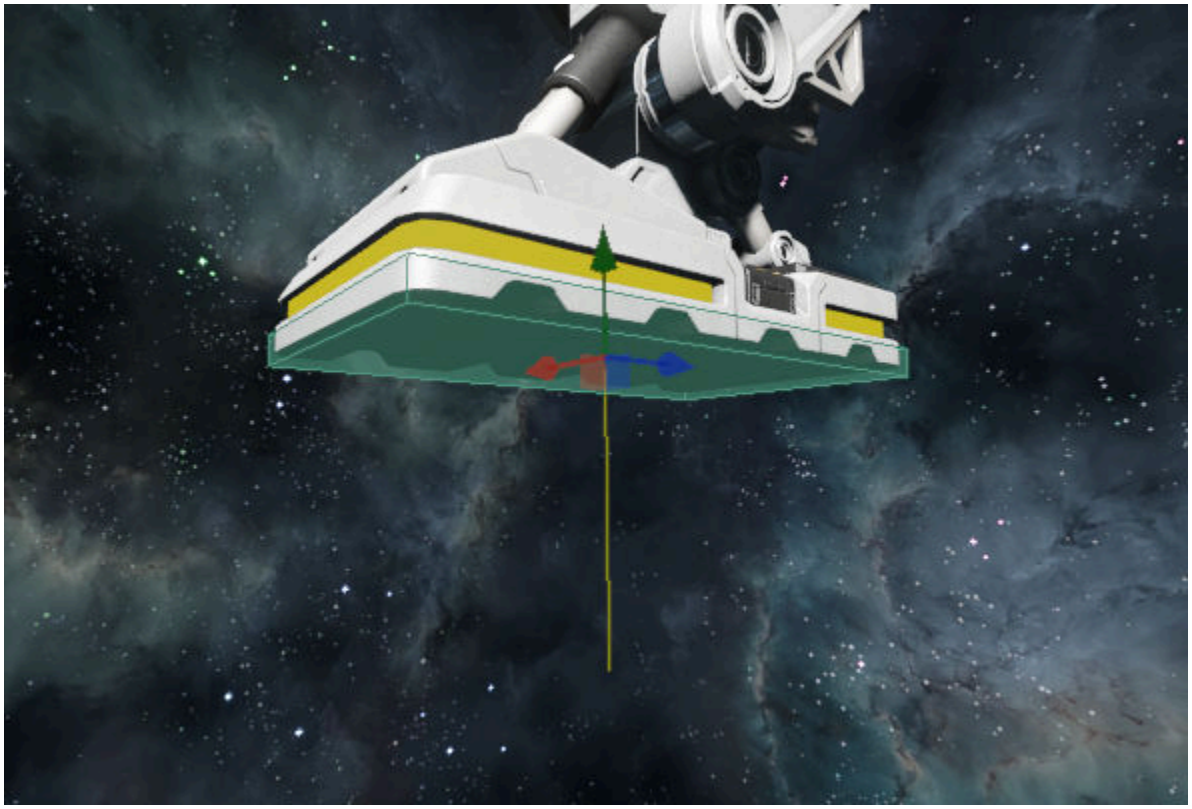
Terminal

The terminal screen is implemented by **MagneticBlockDetailScreen** which is linked to the simulation through **MagneticBlockDetailModel**.

Setup

Model dummy

The first step is to set up the model dummies. The forward vector (yellow line) of the dummy should point away from the block. This alignment is important as it is used to know if a surface is aligned with the magnetic block.



Add IPhysicConstraintCreator

The second step is to add the component that implements IPhysicConstraintCreator (FixedPhysicsConstraintComponent).

The component's tag should be set to "IPhysicConstraintCreator".

Add IEntityDetectorProvider(s)

For locking detection

Next the block will require at least one IEntityDetectorProvider:

- Tag = "LockingEntityDetectorProvider" to detect new entities.
- Tag = "UnLockingEntityDetectorProvider" to detect currently locked entities. (optional)

To have a bigger unlocking area a single dummy is required but the 2 detector must be setup differently:

- If using DummyShapedEntityDetectorComponent, "MaxSeparatingDistance" definition property can be used to artificially increase the dummy box and so increase the detection range.
- If using RaycastEntityDetectorComponent, Adjusting "MaxLength" is enough to increase the range.

For autolock cooldown

The next step will add a cooldown component, the **DetectionBasedCooldownComponent** will require aIEntityDetectorProvider with tag "DetectionBasedCooldownDetector".

The provider for the locking detection can be used or a new one can be added for different ranges.

Add ICooldown

Then add one of the ICooldown implementations (DetectionBasedCooldownComponent).

The tag should be "MagneticAutoLockCooldown"

Add Magnetic Block Components

The following step is to add the MagneticBlockDetectionComponent, adding it will also add the MagneticBlockComponent.

If all tags were set up correctly, all dependencies should be automatically linked by the editor.

Then add the MagneticBlockIndicatorComponent.

Definitions setup

Most definitions are straightforward but some may need additional explanations:

MagneticBlockDefinition

The “DetectionArgs” property will be passed to the IEntityDetectorProvider

- “DetectTopMostEntities” must be true
- “Frequency” can be increased to not scan entities every frame.
  This can also be used to delay the auto lock feature.

MagneticBlockIndicatorDefinition

ReplacementModels

Collection of all model assets used by the block.

DefaultMaterials

Indicate the material to use for a specific state.

- This acts as the default material to use by all mesh parts.
- Each state except NotFunctional should have a material.
  - If not provided, no material will be applied on the model indicator parts.

IndicatorMeshParts

Name of all mesh parts that correspond to the indicators.

- Most uses “EmissiveOff”

MeshPartMaterialOverrides

This is a special collection to override the default material for a specific part at a specific state.

This collection can be used if some part of the model should have a different material than the other parts.

- For example, if model need a green blinking light lamp and a static red light on a unlock button, the definition would be setup like this: