Class

TabletopInteraction

A protocol for objects that manage the entire flow of players interacting with equipment.

visionOS 2.0+

```
class TabletopInteraction
```

Overview

Conform to the `TabletopInteraction.Delegate` protocol to take an appropriate action, depending on the equipment and the phase of the interaction. For example, move equipment or toss a die when a gesture ends.

```
struct MoveInteraction: TabletopInteraction.Delegate {
    func update(interaction: TabletopKit.TabletopInteraction) {
        let equipment = interaction.value.controlledEquipmentID
        guard let destination = interaction.value.proposedDestination else {
            return
        }

        if interaction.value.phase == .ended {
            interaction.addAction(.moveEquipment(matching: equipment, childOf: desti
        }
    }
}
```

To get information about the equipment that the interaction applies to, use the `value` property. To get the phase of the interaction or gesture, use the `Value` `gesturePhase` or `phase` properties.

Then execute actions — for example, move equipment when the phase ends — using the `addAction(_:)` or `addActions(_:)` method.

To start an interaction programmatically, use the `TabletopGame` `startInteraction(onEquipmentID:)` method.

Topics

Performing actions

`protocol Delegate`

A protocol for objects that manage the entire flow of players interacting with equipment.

`struct TossOutcome`

An object representing the final outcome of tossing one equipment, as it appears at the end of its simulation.

`func addAction(some TabletopAction)`

Submit an action tied to this interaction. If the interaction gets canceled, all the associated actions will be automatically rolled back.

`func addAction(some CustomAction)`

Submit a custom action tied to this interaction. If the interaction gets canceled, all the associated actions will be automatically rolled back.

`func addActions(some Sequence<any TabletopAction>)`

Submit a collection of actions tied to this interaction. If the interaction gets canceled, all the associated actions will be automatically rolled back.

```
func toss(equipmentID: EquipmentIdentifier, as: TossableRepresentation,
linearVelocity: Vector3D?, angularVelocity: Vector3D?)
```

Begins a simulation of a toss of the equipment with the specificied parameters. Equipment that begins a toss in the same TabletopInteraction may interact with each other as well as the game's boundary.

`func end()`

Ends the current interaction.

`func cancel()`

Cancels the current interaction. All actions added to the interaction will also be cancelled.

Getting the value of the interaction

`var value: TabletopInteraction.Value`

The current value belonging to this interaction.

`struct Value`

A structure that provides the details about an interaction, such as the phase of the gesture and position of the equipment.

Setting information about the equipment and pose

`func setControlledEquipment(matching: EquipmentIdentifier)`

Replace the current controlled equipment with a different one

`func setPose(Pose3D)`

Sets the pose of the controlled `Equipment`.

Managing the interaction destination

`func setConfiguration(TabletopInteraction.Configuration)`

Sets the configuration of this interaction.

`struct Configuration`

`enum AllowedDestinations`

The possible destinations of equipment in an interaction.

`struct Destination`

An object that represents the destination position and orientation of equipment in an interaction.

~~`func setAllowedDestinations(TabletopInteraction.AllowedDestinations)`~~

Sets which equipment the interaction can target.

Deprecated

Getting the interaction identifier

`struct Identifier`

A unique identifier for interactions.

Determining the dead zone

`enum DeadZone`

The dead zone allows to specify how much the input device should move or rotate from its initial pose to start moving the object.

Handling collision behavior

`enum Constants`

`struct CollisionTargets`

A set of targets for collision behaviors during an interaction

`struct DirectPickupBehavior`

An object that represents the behavior of the pickup phase of the direct interaction. The pickup phase describes how the object moves from its initial pose to the pose it will have when moving rigidly with the input device.

`struct DirectInteractionConstants`

An object that represents the parameters of a direct interaction that cannot be changed while the interaction is active.

`enum IndirectRotationAlignmentBehavior`

An object that represents how the equipment’s orientation should be automatically aligned during the course of the interaction.

`struct IndirectInteractionConstants`

An object that represents the parameters of an indirect interaction that cannot be changed while the interaction is active.

`enum HoverAlignmentBehavior`

An object that describes how the controlled equipment should behave when approaching a target.

`enum HoverAlignmentSource`

An object representing the types of features that can be auto aligned by the `HoverAlignmentBehavior`

`struct ProgrammaticInteractionConstants`

An object that represents the parameters of a programmatic interaction that cannot be changed while the interaction is active.

Handling interaction intents

`enum NewInteractionIntent`

`struct NewDirectInteractionIntent`

An object that represent the developer’s intent when a new direct interaction is proposed by the system

`struct NewIndirectInteractionIntent`

An object that represent the developer’s intent when a new indirect interaction is proposed by the system

---

See Also

Interactions

{} Simulating dice rolls as a component for your game

Create a physically realistic dice game by adding interactive rolling and scoring.

`struct TossableRepresentation`

An object that represents geometric shapes that the player can throw during gameplay, such as dice.

`struct TableSnapshot`

A snapshot of the current state of the table.

`struct TableVisualState`

A structure that represents the appearance of an object on the table.

`struct TableCursor`

A cursor conveys information about one equipment that is currently being controlled by an interaction.

`struct TableCursorIdentifier`

A unique identifier for cursors.