Protocol

EntityEquipment

A protocol for equipment in a game that you render using RealityKit.

TabletopKit | RealityKit | visionOS 2.0+

```
protocol EntityEquipment : Equipment
```

Overview

To render equipment using an entity, follow these steps:

1. Create a structure that conforms to this protocol.
2. Depending on the type of equipment, set the `State` type alias to either `BaseEquipmentState`, `DieState`, or `CardState`.
3. Declare the `id` property as a `EquipmentIdentifier` structure.
4. Declare the `initialState` property as a `State` structure.
5. Implement an initializer that sets these properties and the `entity` property.

```
struct Piece: EntityEquipment {
    typealias State = BaseEquipmentState

    var id: EquipmentIdentifier
    var entity: Entity
    var initialState: State

    @MainActor
    init(index: Int = 0, position: TableVisualState.Point2D) {
        id = .piece(index)
```

```
        entity = try! ModelEntity.load(named: "chess/queen", in: contentBundle)
        initialState = State(parentID: .table, pose: .init(position: position, rotat
    }
}
```

Optionally, implement the `layoutChildren(for:visualState:)` method for equipment that represents groups, and the `restingOrientation(state:)` method to provide a custom resting position. For example, implement the `restingOrientation(state:)` method to flip a card face down in a card game:

```
func restingOrientation(state: State) -> Rotation3D {
    state.faceUp ? .identity : .init(angle: .init(degrees: +180), axis: .init(x: 0,
}
```

Topics

Displaying the equipment

`var entity: Entity`

The entity associated with the equipment.

Required

Relationships

Inherits From

`Equipment`, `Identifiable`

See Also

Equipment

{} Implementing playing card overlap and physical characteristics

Add interactive card game behavior for a pile of playing cards with physically realistic stacking and overlapping.

`protocol` `Equipment`

A protocol for equipment that players directly interact with in a game.

`struct` `EquipmentCollection`

A collection of equipment whose state can be inspected and modified.

`struct` `EquipmentIdentifier`

A unique identifier for equipment.

`protocol` `EquipmentState`

A protocol for the equipment data that TabletopKit syncs between players.

`struct` `EquipmentStateCollection`

A collection of equipment states that can be inspected and modified.

`struct` `BaseEquipmentState`

A state for equipment that contains no equipment-specific data.

`protocol` `CustomEquipmentState`

A specialized protocol for the equipment state that allows to accommodate custom data that TabletopKit syncs between players.

`protocol` `MutableEquipmentState`

A protocol for equipment data that TabletopKit syncs between players, and that can be mutated.

`struct` `CardState`

A state for cards that contains face up and down information.

`struct` `DieState`

A state for dice that contains the current value.

`struct` `RawValueState`

A state for equipment that contains a game-specific value.

`enum` `ControllingSeats`

The seats that can manipulate or interact with the equipment.