TabletopKit / TableSnapshot

Structure

TableSnapshot

A snapshot of the current state of the table.

visionOS 2.0+

```
struct TableSnapshot
```

Topics

Getting the table entity

```
var tableEntity: Entity?
```

Getting information on seats

```
var turn: Set<TableSeatIdentifier>
```

```
var seats: [any TableSeat]
```

```
var seatIDs: [TableSeatIdentifier]
```

```
func seat<S>(of: S.Type, for: Player) -> (S, S.State)?
```

```
func seat<S>(of: S.Type, matching: TableSeatIdentifier) -> (S, S.State)?
```

```
func seats<S>(of: S.Type) -> [(S, S.State)]
```

```
func state<E>(for: E) -> E.State
```

```
func state(matching: TableSeatIdentifier) -> (any SeatState)?
func entity(forSeat: some EntityTableSeat) -> Entity?
```

```
func entity(matching: TableSeatIdentifier) -> Entity?
```

Getting cursors

```
var cursors: [TableCursor]
```

```
func cursor(matching: TableCursorIdentifier) -> TableCursor?
```

```
func cursor(controlling: EquipmentIdentifier) -> TableCursor?
```

Returns the cursor corresponding to an interaction controlling the given equipment ID, or `nil` if no such cursors could be found.

~~func cursors(forPlayer: Player) -> [TableCursor]~~ Deprecated

```
func cursors(hovering: EquipmentIdentifier) -> [TableCursor]
```

```
func cursors(controlling: some Sequence<EquipmentIdentifier>) -> [TableCursor]
```

Finds and returns all the cursors corresponding to an interactions controlling any of the given equipment IDs. Duplicate equipment IDs are ignored.

```
func cursors(for: Player) -> [TableCursor]
```

Finds and returns all the cursors corresponding to an interactions owned by the given player.

```
func cursors(matching: TabletopInteraction.Identifier) -> [TableCursor]
```

Finds and returns all the cursors corresponding to a given interaction.

Getting information on equipment

```
func equipment<E>(of: E.Type) -> [(E, E.State)]
```

```
func equipment<E>(of: E.Type, childrenOf: some Equipment) -> [(E, E.State)]
```

```
func equipment<E>(of: E.Type, childrenOf: EquipmentIdentifier) -> [(E, E.State)]
```

```
func equipment<E>(of: E.Type, matching: some Sequence<EquipmentIdentifier>) -> [(E, E.State)]
```

```
func equipment<E>(of: E.Type, matching: EquipmentIdentifier) -> (E, E.State)?
```

```
func equipmentIDs() -> [EquipmentIdentifier]
func equipmentIDs(childrenOf: some Equipment) -> [EquipmentIdentifier]
```

```
func equipmentIDs(childrenOf: EquipmentIdentifier) -> [EquipmentIdentifier]
```

```
func state(matching: EquipmentIdentifier) -> (any EquipmentState)?
```

```
func entity(matching: EquipmentIdentifier) -> Entity?
```

```
func entity(forEquipment: some EntityEquipment) -> Entity?
```

Getting score counters

```
var counters: [ScoreCounter]
```

```
func counter(matching: ScoreCounter.ID) -> ScoreCounter?
```

Relationships

Conforms To

`Copyable`, `CustomDebugStringConvertible`

See Also

Interactions

{} Simulating dice rolls as a component for your game

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

`class TabletopInteraction`

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

`struct TossableRepresentation`

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

`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.