Framework

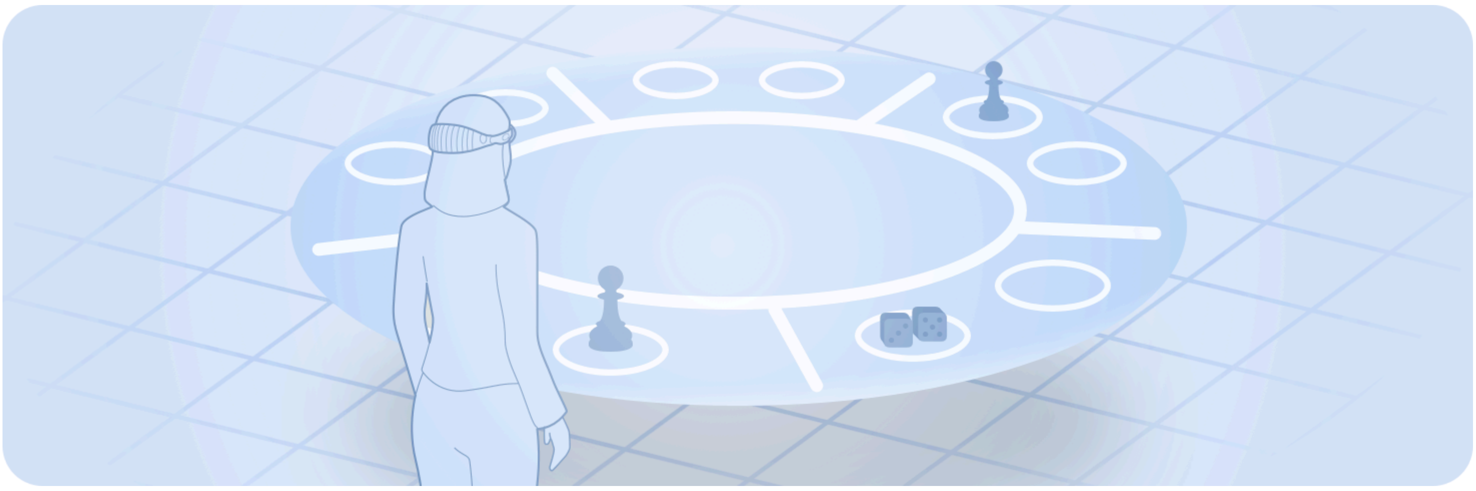
TabletopKit

Create multiplayer spatial games on a virtual table surface and use FaceTime to invite players.

visionOS 2.0+

Overview

TabletopKit helps you create a spatial multiplayer game on a table surface for visionOS, where players join your game using SharePlay. TabletopKit provides support for designing your game, implementing rules, rendering effects, and syncing multiplayer game state.



Follow these steps to implement your TabletopKit game:

- Configure your game on the tabletop and create the game pieces or equipment that players interact with. You provide the renderer that draws your game and its pieces.
- Implement the game rules and player interactions with the equipment. TabletopKit processes and represents player gestures as interactions. You observe the interactions and append your game-specific actions.
- Add effects to the RealityKit entities of renderable equipment and trigger them during interactions. For example, play a sound effect when a player throws a piece or an animation

when a player achieves a goal.

- Set up multiplayer using SharePlay. Start a Group Activities session and provide it to TabletopKit. Then customize the spatial experience of your game. For example, place the players in their seats and spectators around the room.

To get started, create a `TabletopGame` object that represents your game instance and a `TableSetup` object that represents your game layout and equipment.

Topics

Essentials

{} Creating tabletop games

Develop a spatial board game where multiple players interact with pieces on a table.

{} Synchronizing group gameplay with TabletopKit

Maintain game state across multiple players in a race to capture all the coins.

`class TabletopGame`

An object that manages the setup and gameplay of a tabletop game.

`struct TableSetup`

An object that represents the arrangement of seats, equipment, and counters around the game table.

`protocol Tabletop`

A protocol for the table surface in your game.

`protocol EntityTabletop`

A protocol for the table surface in your game when you render it using RealityKit.

`struct TabletopShape`

An object that represents the physical properties of the table.

Seats

`struct TableState`

The state of the table that can be queried and modified.

`protocol TableSeat`

A protocol for seats at the table that players occupy.

`protocol EntityTableSeat`

A protocol for seats at the table that you render using RealityKit.

`struct TableSeatIdentifier`

A unique identifier for seats.

`struct TableSeatState`

The data associated with a seat that a player occupies.

`protocol SeatState`

A protocol for seat data that TabletopKit syncs between players.

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.

`protocol EntityEquipment`

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

`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.

Equipment layout

`protocol EquipmentLayout`

A protocol for objects that describe the layout of equipment.

`struct DefaultEquipmentLayout`

An object that provides a standard configuration for equipment layout.

`struct EquipmentPose2D`

An object that represents the position and rotation of equipment on the XZ plane.

`struct EquipmentPose3D`

An object that represents the 3D position and orientation of equipment on the table.

Score counters

`struct ScoreCounter`

An object that keeps a score in a tabletop game.

`struct CounterCollection`

A collection of score counters that can be inspected and modified.

Players

`struct Player`

A player in a tabletop game.

`struct PlayerIdentifier`

A unique identifier for players.

Actions

`protocol TabletopAction`

A protocol for objects that describe an action in a tabletop game.

`struct MoveEquipmentAction`

An action that moves a piece of equipment on the table or changes the grouping.

`struct UpdateEquipmentAction`

An action that updates properties of equipment on the table.

`struct SetTurnAction`

An action that sets the current seats participating in the current turn.

`struct UpdateCounterAction`

An action that updates the game counter.

`struct CreateBookmarkAction`

An action that takes a snapshot of the game.

`protocol CustomAction`

A protocol that represents an action whose behavior is implemented outside of TabletopKit. A custom action that can be applied to a `TableState`.

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 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.

Bookmarks

`struct StateBookmark`

A snapshot of the game state at a point in time.

`struct StateBookmarkIdentifier`

A unique identifier for bookmarks.

Multiplayer network session

`class TabletopNetworkSession`

An object that coordinates network-related tasks in multiplayer games.

`protocol TabletopNetworkSessionCoordinator`

A protocol for objects that manage network sessions between peers.

`enum TabletopSendMessageResult`

The possible results of sending messages in a network session.

Debugging

`struct DebugDrawOptions`

Types of items in a rendering that you want to debug.