Sample Code

Creating turn-based games

Develop games where multiple players take turns and can exchange data while waiting for their turn.

Download

iOS 15.0+ | iPadOS 15.0+ | Xcode 15.0+

Overview

This sample code project uses the GameKit framework to create a simple turn-based game where two players take turns incrementing a counter until a player stops playing or forfeits the match. The game uses the turn-based matchmaker interface for starting and managing multiple turn-based matches. This game also uses the GameKit exchanges feature for participants in a turn-based game to message and exchange data.

To create your own turn-based game, replace the SwiftUI files with your gameplay interface, and modify the `TurnBasedGame` class to support your game.

Configure the sample code project

To configure the sample code project, perform the following steps in Xcode:

1. If necessary, change the Bundle Identifier to a unique ID on the Signing & Capabilities pane.
2. Add your Apple ID account and assign the target to a team so Xcode can enable Game Center.
3. In App Store Connect, create an app record that matches the bundle ID (see Add a new app in App Store Connect Help).
4. Connect two iOS devices to your Mac.

5. If necessary, click Register Device in the Signing & Capabilities pane to create the provisioning profile.
6. Build and run the sample on the two iOS devices.
7. If the Welcome to Game Center sheet appears, sign in using a different Apple ID on each device.

Initialize the player and register for turn-based events

Before using any GameKit APIs, the game needs to initialize the local player by presenting an interface for them to sign in to Game Center on their device. The `TurnBasedGame.authenticatePlayer()` method handles the initialization flow and, when complete, registers for turn-based game events.

```
// Register for turn-based invitations and other events.
GKLocalPlayer.local.register(self)
```

Start a turn-based match

To start or join an existing match, the player taps the Start Match button on the content view. Then the `TurnBasedGame.startMatch()` action method creates a match request and presents a `GKTurnBasedMatchmakerViewController` interface where the player can invite friends or automatch to fill empty slots.

```
// Create a match request.
let request = GKMatchRequest()
request.minPlayers = minPlayers
request.maxPlayers = maxPlayers
if playersToInvite != nil {
    request.recipients = playersToInvite
}

// Present the interface where the player selects opponents and starts the game.
let viewController = GKTurnBasedMatchmakerViewController(matchRequest: request)
viewController.turnBasedMatchmakerDelegate = self
rootViewController?.present(viewController, animated: true) { }
```

After the player selects opponents in the Game Center interface, GameKit dismisses the view controller and invokes the `GKTurnBasedEventListener.player(_:receivedTurnEventFor:didBecomeActive:)` protocol method, passing a new `GKTurnBasedMatch` object. This method handles a variety of turn-based events throughout the match.

The first time GameKit invokes the `player(_:receivedTurnEventFor:didBecomeActive:)` method, the local player is the current participant. This method sets the `myTurn` property to `true`, which enables the Take Turn button in the game view interface.

```
// Update the interface depending on whether it's the local player's turn.
myTurn = GKLocalPlayer.local == match.currentParticipant?.player ? true : false
```

GameKit doesn't send invitations to the match until the participant who starts the match takes the first turn. Therefore, this method displays a placeholder name and avatar for the opponent until the opponent joins the match.

Instead of retaining the `GKTurnBasedMatch` object, which can change during the course of the match, this method retains the match ID so the game can fetch the current match object as necessary later.

```
// Retain the match ID so action methods can load the current match object later.
currentMatchID = match.matchID
```

Pass the turn to the next participant

When the current participant taps the Take Turn button, the `TurnBasedGame.takeTurn()` action method increments the count and passes the turn to the opponent.

First the `takeTurn()` method loads the current match object using the match ID it retains in the `player(_:receivedTurnEventFor:didBecomeActive:)` method.

```
// Load the most recent match object from the match ID.
let match = try await GKTurnBasedMatch.load(withID: currentMatchID!)
```

This method passes the turn to the opponent using the `endTurn(withNextParticipants:turnTimeout:match:completionHandler:)` method. It passes the next participant an array containing just the opponent, and a `Data` representation of the game state that contains the current count.

```
// Create the game data to store in Game Center.
let gameData = (encodeGameData() ?? match.matchData)!

// Remove the current participant from the match participants.
let nextParticipants = activeParticipants.filter {
    $0 != match.currentParticipant
```

```
}
```

Then pass the next participants and the game data, specifying a timeout in case the receipients don't respond, to the `endTurn(withNextParticipants:turnTimeout:match:completionHandler:)` method.

```
// Pass the turn to the next participant.
try await match.endTurn(withNextParticipants: nextParticipants, turnTimeout: GKTurnT
                        match: gameData)
```

> **Note**
>
> In games with more than two participants, best practice is to add multiple participants to the array for the next participants. Then if communication fails, or a participant doesn't finish their turn within the time limit, Game Center passes the turn to the next participant in the array to keep the match going.

The first time the game passes the turn, GameKit sends invitations to all the participants.

Accept turn-based match invitations

When the opponent taps the Accept button in the dialog that GameKit displays on their device, GameKit invokes the `GKTurnBasedEventListener.player(_:receivedTurnEventFor:didBecomeActive:)` method.

When the local player accepts the invitation, the other participant's status is `GKTurnBasedParticipant.Status.active` and its player property is non-`nil`, allowing this method to get the opponent's name and load their avatar from the `GKPlayer` object.

```
// If the player starts the match, the opponent hasn't accepted the invitation and h
let participant = participants.first
if (participant != nil) && (participant?.status != .matching) && (participant?.playe
    if opponent == nil {
        // Load the opponent's avatar and create the opponent object.
        let image = try await participant?.player?.loadPhoto(for: GKPlayer.PhotoSize
        opponent = Participant(player: (participant?.player)!,
                               avatar: Image(uiImage: image!))
    }

    // Restore the current game data from the match object.
    decodeGameData(matchData: match.matchData!)
```

```
}
```

The match object also contains data, such as the current count, that this method encodes to update the game view interface. The `TurnBasedGame.encodeGameData()` and `TurnBasedGame.decodeGameData()` methods store just the game properties that you need to continue playing when GameKit sends turn-based events between participants.

Forfeit a turn-based match

If a participant taps the Forfeit button, the `TurnBasedGame.forfeitMatch()` action method quits the match whether it's the participant's turn or not. Because only the current participant can update the match data, GameKit provides two different methods to leave a match.

When it's the local player's turn, the `forfeitMatch()` method creates a `Data` representation of the game data, selects the next participants, and invokes the `participantQuitInTurn(with:nextParticipants:turnTimeout:match:completionHandler:)` method, passing `GKTurnBasedMatch.Outcome.quit` as the outcome.

```
// Create the game data to store in Game Center.
let gameData = (encodeGameData() ?? match.matchData)!

// Remove the participants who quit and the current participant.
let nextParticipants = match.participants.filter {
  ($0.status != .done) && ($0 != match.currentParticipant)
}

// Forfeit the match.
try await match.participantQuitInTurn(
    with: GKTurnBasedMatch.Outcome.quit,
    nextParticipants: nextParticipants,
    turnTimeout: GKTurnTimeoutDefault,
    match: gameData)
```

When it's not the local player's turn, the `forfeitMatch()` method calls the `participantQuitOutOfTurn(with:withCompletionHandler:)` method.

```
// Forfeit the match while it's not the local player's turn.
try await match.participantQuitOutOfTurn(with: GKTurnBasedMatch.Outcome.quit)
```

Both of these methods change the status of the participant to `GKTurnBasedParticipant.Status.done` and generate a turn-based event that invokes the `player(_:receivedTurn`

EventFor:didBecomeActive:) method. If there aren't enough participants to continue (when there's only one participant remaining in the match), this method sets the outcome of the recipient to GKTurnBasedMatch.Outcome.won and ends the game.

```
// Remove participants who quit or otherwise aren't in the match.
let nextParticipants = match.participants.filter {
    $0.status != .done
}

// End the match if active participants drop below the minimum.
if nextParticipants.count < minPlayers {
    // Set the match outcomes for the active participants.
    for participant in nextParticipants {
        participant.matchOutcome = .won
    }

    // End the match in turn.
    try await match.endMatchInTurn(withMatch: match.matchData!)

    // Notify the local player when the match ends.
    youWon = true
}
```

The `takeTurn()` action method also ends the match with a win if there aren't enough participants to continue.

Exchange data between participants

Participants can message other participants and exchange items while they're waiting for the current participant to take their turn. When a participant taps the message bubble in the game view, the chat view sheet appears so the participant can send a text message to the opponent.

The `TurnBasedGame.sendMessage()` method sends the text message as the data in an exchange request. It passes the data, a localizable message, and a response timeout to the GKTurnBasedMatch.sendExchange(to:data:localizableMessageKey:arguments:timeout:completionHandler:) method.

```
// Create the exchange data.
guard let data = content.data(using: .utf8) else { return }

// Load the most recent match object from the match ID.
```

```
let match = try await GKTurnBasedMatch.load(withID: currentMatchID!)

// Remove the local player (the sender) from the recipients;
// otherwise, GameKit doesn't send the exchange request.
let participants = match.participants.filter {
    localParticipant?.player.displayName != $0.player?.displayName
}

// Send the exchange request with the message.
try await match.sendExchange(to: participants, data: data,
                             localizableMessageKey: "This is my text message.",
```

In the recipient's game instance, GameKit invokes the `GKTurnBasedEventListener.player(_:receivedExchangeRequest:for:)` protocol method, passing the player, the exchange object, and the match object. This method displays the message to the recipient when they have the chat view sheet open. Otherwise, the message appears in the chat view thread the next time the participant opens it.

```
// Unpack the exchange data and display the message in the chat view.
let content = String(decoding: exchange.data!, as: UTF8.self)
let message = Message(content: content, playerName: exchange.sender.player?.displayN
```

For expedience, this method immediately accepts the exchange request when the exchange status is `GKTurnBasedExchangeStatus.active`.

```
try await exchange.reply(withLocalizableMessageKey: "I accept the exchange request."
```

Save completed exchanges

In the sample, each participant starts with 50 items next to their names. To exchange items, a participant clicks the Exchange Item button below the Send Reminder button. The same code that accepts the chat messages also automatically accepts the exchange item request.

When participants reply to exchange requests, GameKit invokes the `player(_:receivedExchangeReplies:forCompletedExchange:for:)` protocol method in the current participant and the sender's game instance. In this sample, the `player(_:receivedExchangeReplies:forCompletedExchange:for:)` method invokes the `TurnBasedGame.saveExchanges()` method.

Because GameKit requires that the current participant save completed exchanges before ending a match, the `takeTurn()` method also invokes the `saveExchanges(for:)` method before

ending a match.

Because only the current participant can save exchanges, the `saveExchanges(for:)` method first checks whether the local player is the current participant before continuing.

```
// Check whether the local player is the current participant who can save exchanges.
guard myTurn else { return }
```

This method resolves the exchange by transferring one item from the recipient to the sender of the exchange request. Then it adds the participant's exchange items to the game data, and passes it with the completed exchanges to the `GKTurnBasedMatch.saveMergedMatch(_:withResolvedExchanges:completionHandler:)` method.

```
// Resolve the game data to pass to all participants.
let gameData = (encodeGameData() ?? match.matchData)!

// Save and forward the game data with the latest items.
Task {
    try await match.saveMergedMatch(gameData, withResolvedExchanges: completedExchar
}
```

The `saveMergedMatch(_:withResolvedExchanges:completionHandler:)` method removes the completed exchange objects from the `GKTurnBasedMatch` object's `completedExchanges` and `exchanges` properties.

Because the match data changes, GameKit invokes the `player(_:receivedTurnEventFor:didBecomeActive:)` method in the other participants' game instances. The `player(_:receivedTurnEventFor:didBecomeActive:)` method unpacks the match data and displays the exchanged items.

See Also

Turn-based games

- Starting turn-based matches and passing turns between players

  Let Game Center store and forward match data between players in a turn-based game.

- Sending messages to players in turn-based games

  Notify players of match events by sending messages and game data.

Exchanging data between players in turn-based games

Add the ability for players to exchange game data and send messages while waiting for their turns.

`class` `GKTurnBasedMatchmakerViewController`

An interface that allows a player to invite other players to a turn-based match and automatch to fill any empty slots.

`class` `GKTurnBasedMatch`

An object that encapsulates the match data for games where players take turns.

`class` `GKTurnBasedParticipant`

A participant in a turn-based match.

`protocol` `GKTurnBasedEventListener`

The protocol that handles turn-based and data-exchange events between participants in a match.

`class` `GKTurnBasedExchange`

Exchange request information that participants send in a turn-based match.

`class` `GKTurnBasedExchangeReply`

Details about a recipient's response to an exchange request.

`GKGameCenterBadgingDisabled`

A Boolean value indicating whether GameKit can add badges to a turn-based game icon.