Article

Saving the player's game data to an iCloud account

Save game data during play or after a game in the player's iCloud account that's accessible from any device.

Overview

GameKit provides a convenient method to save game data that needs to persist after the player quits the game. You can save game data to record a player's progress, separate data for family members who use the same device, and support players who switch between devices. For example, let the player start your game on their iPhone, and then continue playing on their Mac when they arrive home. You can save just one game at a time using the same filename or multiple games by giving each game instance a unique filename.

To save a game, the player must have an iCloud account and enable iCloud Drive in their iCloud settings. To save space in their account and improve performance in your game, minimize the amount of data you save.

Provide a container identifier

For GameKit to store the game data in the player's iCloud account, you need to provide an identifier for the iCloud container that stores the data. Add the iCloud capability to your project and select the iCloud Documents checkbox.

For more information, see Configuring iCloud services.

Save a game

Decide what data to save to the file and encode it as a `Data` object. For example, create a structure containing the properties you want to save to the file that conforms to the `Codable` protocol. Then encode the structure using the `PropertyListEncoder` class.

Save the `Data` object to the file using the `GKLocalPlayer` `saveGameData(_:withName:completionHandler:)` method. GameKit overrides an existing file with the same filename. If you want to keep the previous file, pass a unique filename when you invoke this method.

```
// Encode and save game-specific data.
let gameData = encode(score: myScore)
do {
    try await GKLocalPlayer.local.saveGameData(gameData!, withName: matchName)
} catch {
    print("Error: \(error.localizedDescription).")
}
```

Fetch the saved games

Later, use the `fetchSavedGames(completionHandler:)` method to retrieve all the saved games. This method returns an array of `GKSavedGame` objects that contain file properties, such as the filename and modification date. For example, you can use the results to present a list of the saved games to the player.

```
// Fetch the current saved games.
let games = try await GKLocalPlayer.local.fetchSavedGames()
```

Use the `GKSavedGame` `loadData(completionHandler:)` method to load the contents of a file. For example, let the player choose the game that they want to continue playing from the list. Then restart the game with the contents of the file.

```
// Continue a game with the data from a file.
func load(game: GKSavedGame) async {
    let data: Data
    do {
        // Load the game data from the file.
        data = try await game.loadData()
    } catch {
        print("Error: \(error.localizedDescription).")
        return
    }
    // Decode the game data and start the game.
```

```
    if let gameData = decode(matchData: data) {
        startGame(score: gameData.score)
    }
}
```

Optionally, provide a similar interface for the player to remove saved games using the `GKLocalPlayer` `deleteSavedGames(withName:completionHandler:)` method.

Resolve conflicts

Occasionally, your game may need to resolve a conflict that occurs when the player saves game data from multiple devices using the same filename. For example, your game may present an interface that lets the player choose the correct game data or determine which data to keep based on some other criteria, such as the game with the highest score.

If GameKit notices two saved games with the same filename when you either fetch saved games, load game data, or save game data, it invokes the `GKSavedGameListener` `player(_:hasConflictingSavedGames:)` protocol method. To resolve a conflict in your code, adopt the `GKLocalPlayerListener` protocol and implement this method to save the correct data using the `GKLocalPlayer` `resolveConflictingSavedGames(_:with:completionHandler:)` method.

See Also

Players

- Connecting players with their friends in your game

  Give players the ability to connect and interact with friends in your game.

- Protecting the player's privacy using scoped identifiers

  Use the scoped identifiers that GameKit provides you as player IDs when transmitting or saving player data.

`class` `GKLocalPlayer`

The local player who signs in to Game Center on the device running the game.

`class` `GKPlayer`

A remote player who the local player running your game can invite and communicate with through Game Center.

`class` `GKBasePlayer`

A class that provides common data and methods for the different player objects.

```
protocol GKLocalPlayerListener
```

A protocol that handles events for Game Center players.

```
static let GKPlayerAuthenticationDidChangeNotificationName:
NSNotification.Name
```

A notification that posts after GameKit authenticates the local player.

```
static let GKPlayerDidChangeNotificationName: NSNotification.Name
```

A notification that posts when a player object's data changes.