Article

Finding multiple players for a game

Discover and invite other players to participate in a real-time game.

Overview

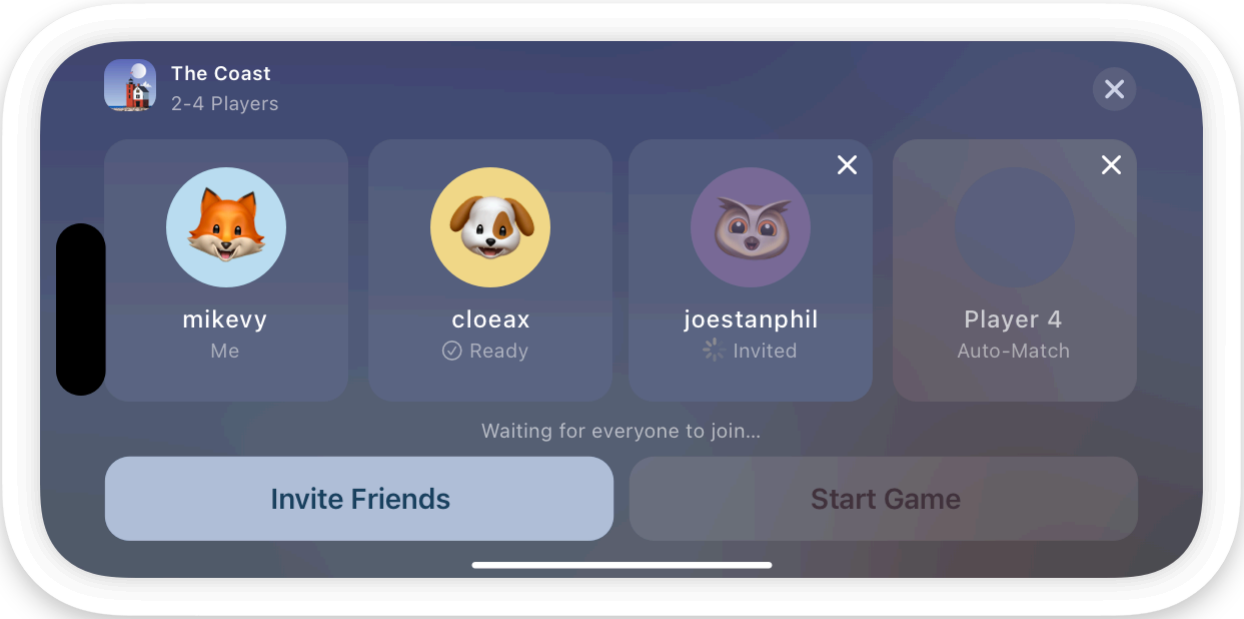
GameKit enables the user of your game, called the *local player*, to discover and connect with other players in real-time multiplayer games.

For design guidance, see Human Interface Guidelines > Technologies > Game Center > Multiplayer.

Choose whether to use the Game Center interface

You can present a Game Center interface that allows the local player to invite nearby or recent players, friends, contacts, Game Center or Message groups, and anyone with a phone number or email address.

The default interface allows the player to use automatic matches to fill any empty slots. The local player can see the status of players accepting or declining their invitations, as they invite additional players or remove players. Once the number of players that you specify accept their invitations, the local player can start gameplay. Optionally, your game can continue processing responses to invitations, and GameKit can continue automatching to fill empty slots until the maximum number of players that you specify join the game.



Alternatively, you can present your own custom interface to invite players, invite specific players, or let Game Center automatch players without presenting an interface.

In your code, you create a match request and optionally present the matchmaker view controller. Then you implement methods to present the appropriate interface to all players during the process of receiving and accepting invitations.

Create a match request

First, create a `GKMatchRequest` object and set the parameters of the game. You can set the number of players, apply groups and attributes to filter the matches, specify the game mode, and set other criteria.

```
let request = GKMatchRequest()
request.minPlayers = 2
request.maxPlayers = 10
```

Use the `maxPlayersAllowedForMatch(of:)` method to get the maximum number of players for your match type (`GKMatchType.peerToPeer`, `GKMatchType.hosted`, or `GKMatchType.turnBased`).

If you use game-specific matchmaking rules to find players, set the `queueName` and optionally, the `properties` properties of the `GKMatchRequest` object. Also, set the `minPlayers` and `maxPlayers` properties within the bounds of the rule set's `minPlayers` and `maxPlayers` fields. For more information, see Finding players using matchmaking rules.

Enable the local player to choose other players

The local player can choose opponents from the Game Center interface when you present a `GKMatchmakerViewController` object and handle the resulting invitation events.

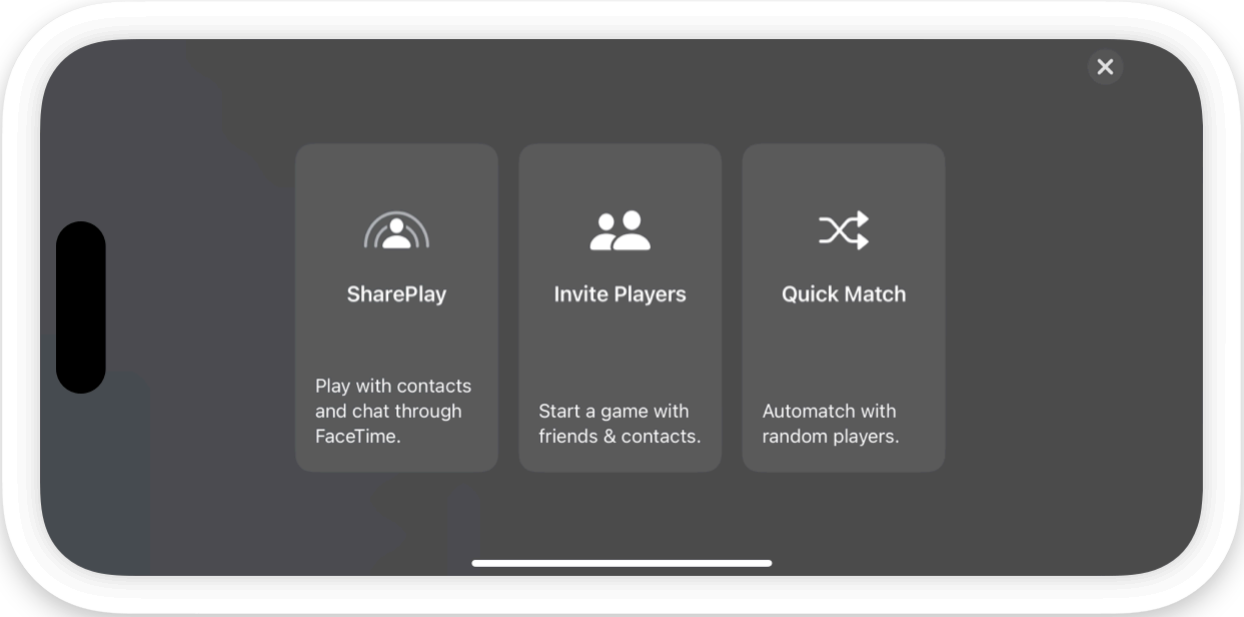
Pass the `GKMatchRequest` object to the `GKMatchmakerViewController` initializer and set the view controller's delegate to your game object before you present it. Use the `present(_:animated:completion:)` or equivalent method to present the matchmaker view controller, in which the local player can choose players or automatch to fill empty slots.

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

Optionally, configure the view controller before you present it:

- To show only nearby players for games where players need to be in the same room, as in augmented reality (ARKit) games, set the `matchmakingMode` property to `GKMatchmakingMode.nearbyOnly`.
- To use automatch only to connect with other players, set the mode to `GKMatchmakingMode.automatchOnly`, or to invite players without using automatch, set the mode to `GKMatchmakingMode.inviteOnly`.
- To allow the local player to start gameplay with a minimum number of players, while Game Center continues to fill empty slots in the background, set the `canStartWithMinimumPlayers` property to `true`. Then design your game to progressively add players as they accept their invitations.

Players automatically have the ability to chat with other players using FaceTime. The view controller shows a SharePlay button, in addition to the Invite Players and Quick Match buttons. To implement a custom group activities experience, see `startGroupActivity(playerHandler:)`. For macOS apps, add the Group Activities capability to your Xcode project to show the SharePlay button.

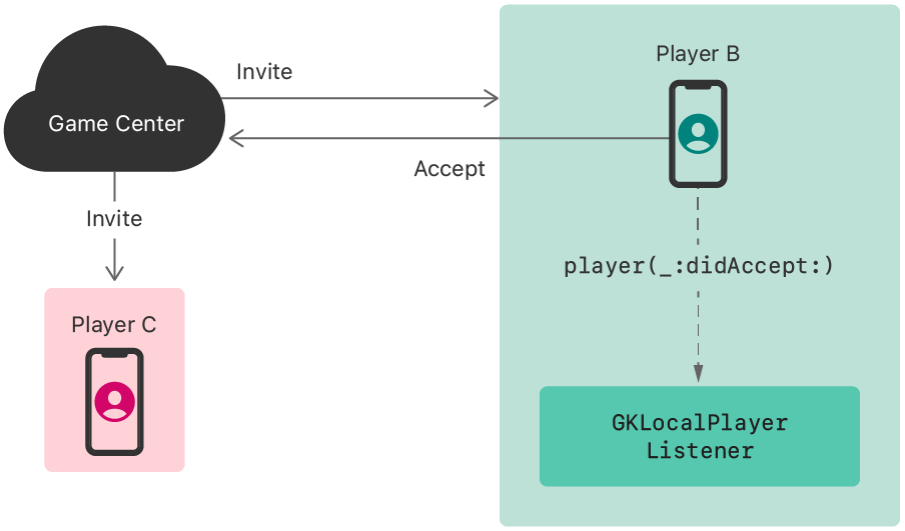


Then adopt the `GKLocalPlayerListener` and `GKMatchmakerViewControllerDelegate` protocols to implement methods that keep track of the invitations GameKit sends and the invitations players accept. To receive the `GKInviteEventListener` callbacks that the `GKLocalPlayerListener` protocol includes, register your game object with the local player object:

```
GKLocalPlayer.local.register(self)
```

Accept an invitation from another player

Game Center sends invitations to the players asynchronously. When a player accepts an invitation from another player, GameKit notifies Game Center and invokes the `GKInviteEventListener` `player(_:didAccept:)` method.



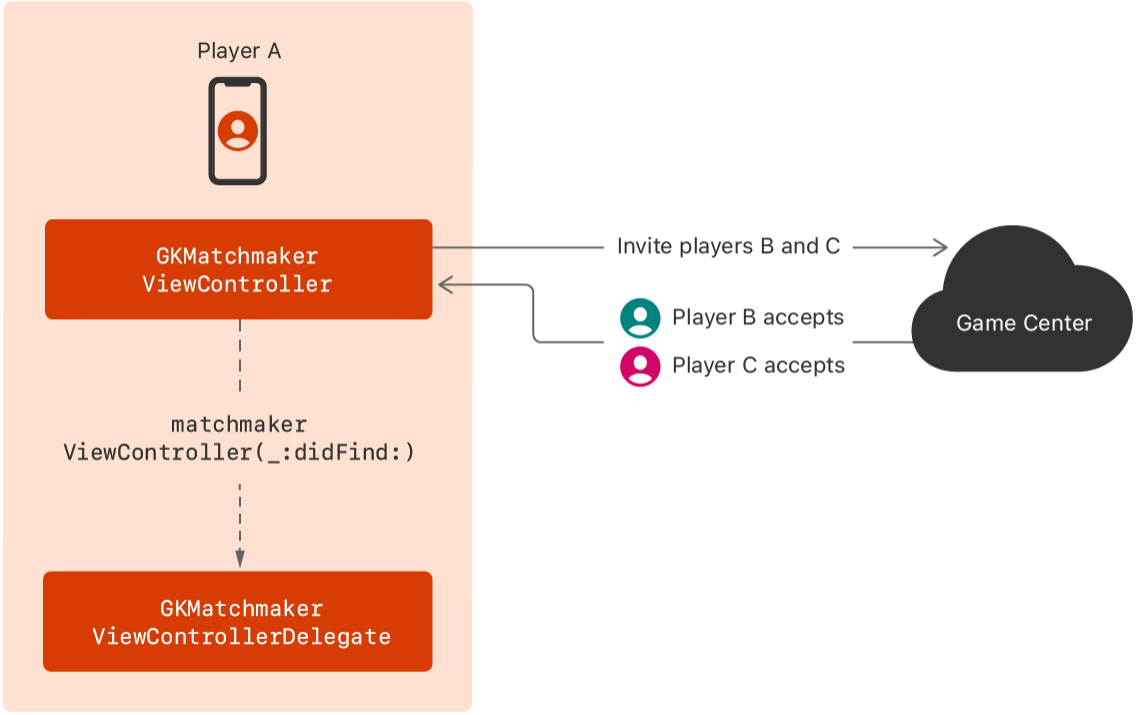
Implement the `player(_:didAccept:)` method to present the matchmaker view controller in the invitation state so the recipient of the invitation can see the progress as other players accept

their invitations. Create the matchmaker view controller by passing the invite to the initializer, and then present it.

```
func player(_ player: GKPlayer, didAccept invite: GKInvite) {
    // Present the matchmaker view controller in the invitation state.
    if let viewController = GKMatchmakerViewController(invite: invite) {
        viewController.matchmakerDelegate = self
        rootViewController?.present(viewController, animated: true) { }
    }
}
```

Start the game when players accept invitations

The matchmaker view controller shows the players who received or accepted invitations. When all the players accept their invitations, GameKit invokes the `GKMatchmakerViewControllerDelegate` `matchmakerViewController(_:didFind:)` method in the app instances for all players in the game, including the player who sent the invitations.



Implement the `matchmakerViewController(_:didFind:)` method to dismiss the view controller and start your game. If the player count is low or players accept invitations quickly, you can add a delay before you dismiss the view controller, so players see the outgoing invitations before you start the game.

```
func matchmakerViewController(_ viewController: GKMatchmakerViewController,
                              didFind match: GKMatch) {
    // Dismiss the view controller.
```

```
    viewController.dismiss(animated: true, completion: nil)

    // Set the match delegate.
    match.delegate = myGame

    // Start the game with the players in the match.
}
```

Then use the `GKMatch` object that GameKit passes to this delegate method to get the player list and send data from one player to another during the game.

Handle matchmaking cancellations and errors

When the local player closes the matchmaker interface without choosing players, or some other error occurs while choosing players, implement the `matchmakerViewControllerWasCancelled(_:)` delegate method to dismiss the view controller.

```
func matchmakerViewControllerWasCancelled(_ viewController: GKMatchmakerViewControl
{
    // Dismiss the view controller.
    viewController.dismiss(animated:true)
}
```

Also, implement the `matchmakerViewController(_:didFailWithError:)` method to handle any errors that might occur. For possible errors, see `GKError`.

Let Game Center match the player with others

GameKit can automatch the local player without presenting the Game Center matchmaking interface where players invite others. To start automatch, pass the match request to the shared `GKMatchmaker` object using the `findMatch(for:withCompletionHandler:)` method.

```
// Start automatch.
do {
    match = try await GKMatchmaker.shared().findMatch(for: request)
} catch {
    print("Error: \(error.localizedDescription).")
    return
}
```

If another game instance starts automatch at the same time, the `findMatch(for:withCompletionHandler:)` method creates a match with that player and returns the corresponding `GKMatch` object. To receive a callback when players join the match, set the match's delegate to an object that implements the `match(_:player:didChange:)` delegate method.

Implement the `match(_:player:didChange:)` method to check whether the `state` parameter is `GKPlayerConnectionState.connected` before starting the game and accessing the player's properties, such as the display name and avatar.

```
func match(_ match: GKMatch, player: GKPlayer, didChange state: GKPlayerConnectionSt
    switch state {
    case .connected:
        // Load the opponent's avatar.
        player.loadPhoto(for: GKPlayer.PhotoSize.small) { (image, error) in
            if let image {
                self.opponentAvatar = Image(uiImage: image)
            }
            else if let error {
                print("Error: \(error.localizedDescription).")
            }
        }
    case .disconnected:
        // Handle disconnected state.
    default:
        // Handle unknown state.
    }
}
```

After enough players join the match, or if the player cancels automatch, call the shared `GKMatchmaker` `finishMatchmaking(for:)` method to stop GameKit from looking for players.

Alternatively, to invite specific players without presenting an interface, set the match request's `recipients` property before invoking the `findMatch(for:withCompletionHandler:)` method.

See Also

Real-time games

{} Creating real-time games

Develop games where multiple players interact in real time.

Exchanging data between players in real-time games

Send data between players in a real-time multiplayer game.

Adding voice chat to multiplayer games

Enable players to voice chat with all, or groups of, players in a multiplayer game.

Matchmaking rules

Game Center applies different type of rules you create in a particular order to find the best matches.

`class` `GKMatchRequest`

An object that encapsulates the parameters to create a real-time or turn-based match.

`class` `GKMatchmaker`

An object that creates matches with other players without presenting an interface to the players.

`class` `GKMatchmakerViewController`

An interface that allows a player to invite other players to a real-time game and automatch to fill any empty slots.

`protocol` `GKInviteEventListener`

A protocol that handles invite events from Game Center.

`class` `GKInvite`

An invitation to join a match sent to the local player from another player.

`class` `GKMatch`

A peer-to-peer network between a group of players that sign into Game Center.