Multiplayer Gaming with Game Center

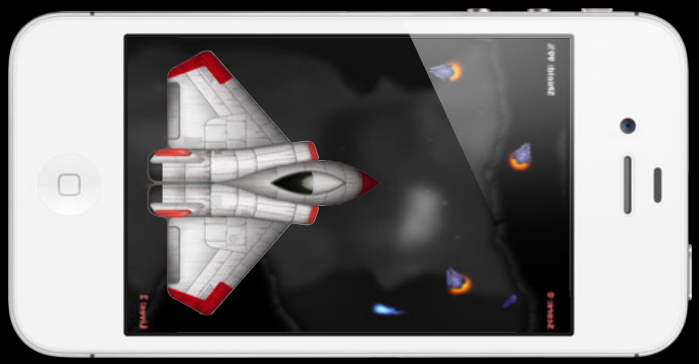
Session 519

Christy Warren
Senior iOS Development Engineer

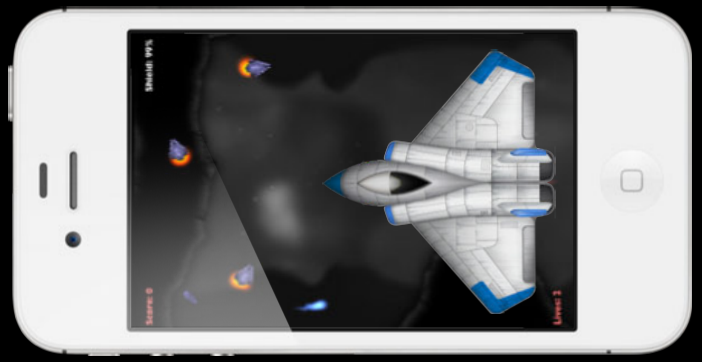
What You Will Learn

Multiplayer support

- Matchmaking UI
- Programmatic auto-match
- Peer-to-peer communications
- Turn-Based gaming



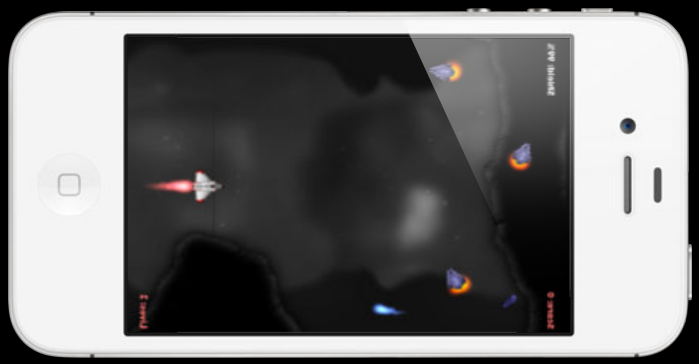
vs.

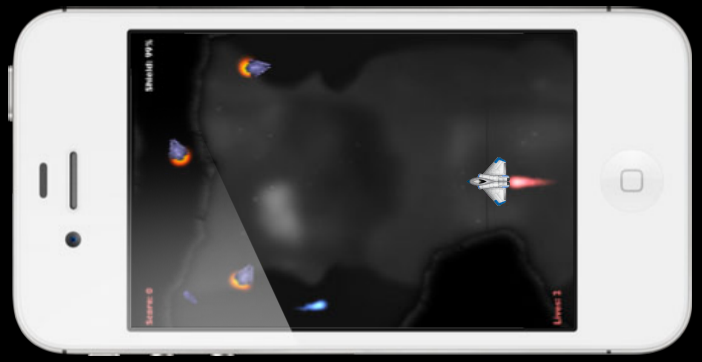


What You Will Learn

Multiplayer support

- Matchmaking UI
- Programmatic auto-match
- Peer-to-peer communications
- Turn-Based gaming





Why Add Multiplayer

- Discoverable
- Make it stand out
  - Players enjoy real opponents
  - Top games support multiplayer
- Increase longevity
  - Foster competition and engagement
  - Leaderboards and achievements
- Chance for immortality

Why Add Multiplayer

- Discoverable
- Make it stand out
  - Players enjoy real opponents
  - Top games support multiplayer
- Increase longevity
  - Foster competition and engagement
  - Leaderboards and achievements
- Chance for immortality

New in Game Center

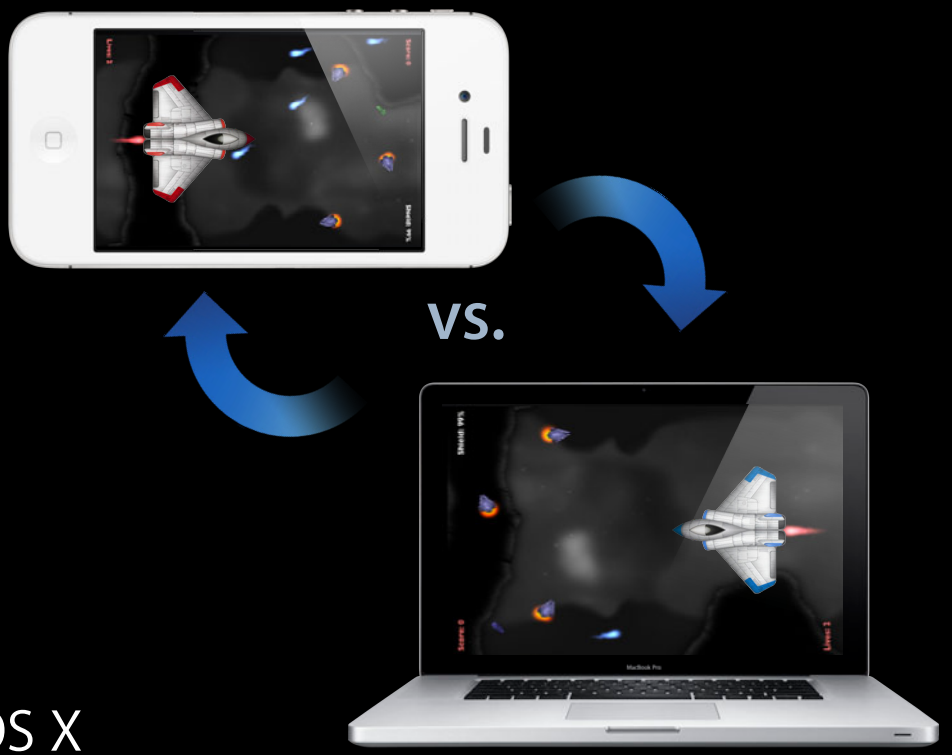
- New Matchmaking UI
- Discover players on nearby devices
- Programatic invites
- Re-match API
- Host election API
- Turn-Based improvements
  - Better handling of missed turns
  - Turn match data saving

New in Game Center

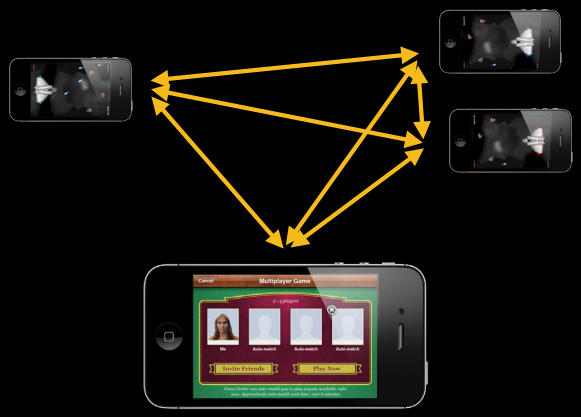
- New Matchmaking UI
- Discover players on nearby devices
- Programatic invites
- Re-match API
- Host election API
- Turn-Based improvements
  - Better handling of missed turns
  - Turn match data saving

New in Game Center

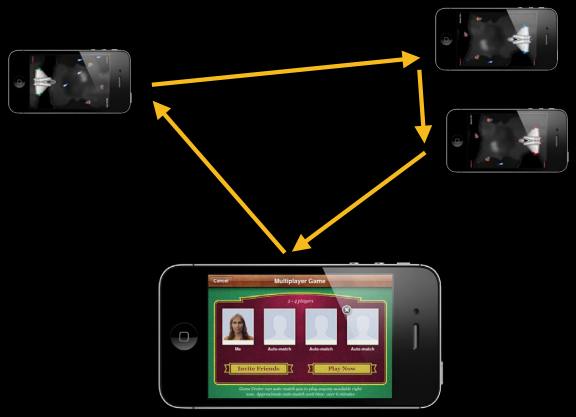
- New Matchmaking UI
- Discover players on nearby devices
- Programatic invites
- Re-match API
- Host election API
- Turn-Based improvements
  - Better handling of missed turns
  - Turn match data saving
- Interoperates with Game Center for OS X



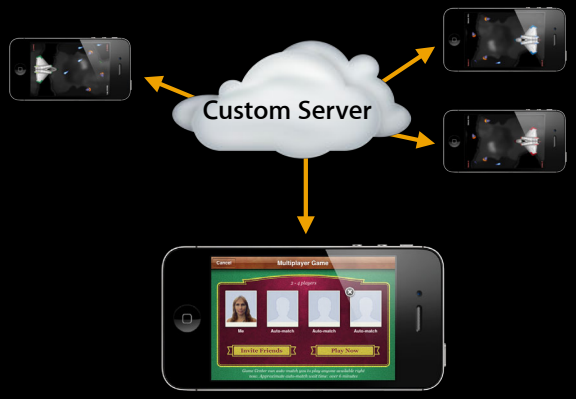
Styles of Multiplayer



Peer-to-Peer



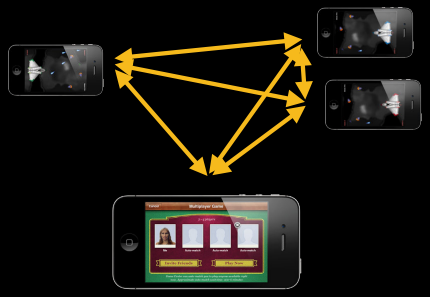
Turn-Based



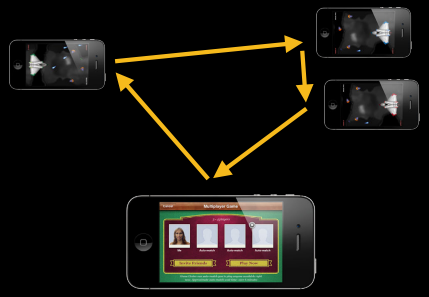
Server-Based

Styles of Multiplayer

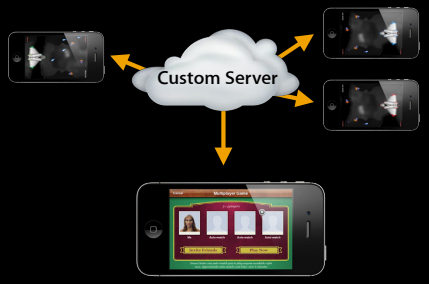
Comparison



Peer-to-Peer



Turn-Based

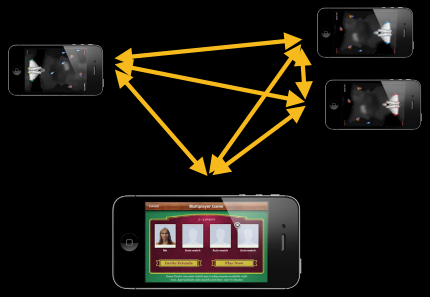


Server-Based

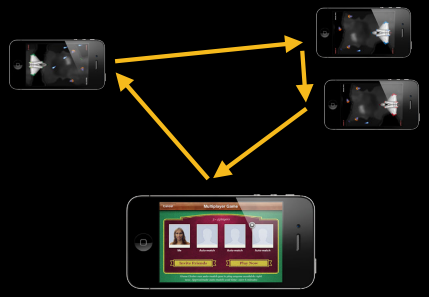
Players	2–4	2–16	2–16

Styles of Multiplayer

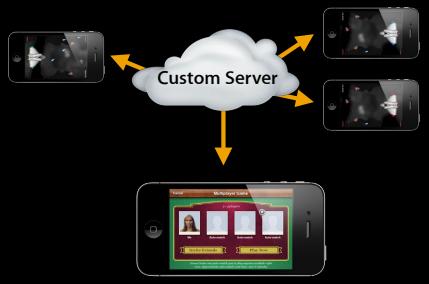
Comparison



Peer-to-Peer



Turn-Based

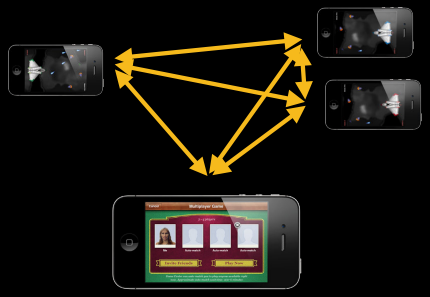


Server-Based

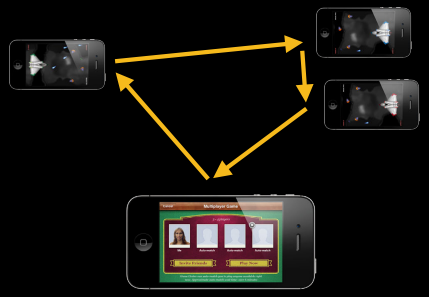
Players	2–4	2–16	2–16
Game Play	Simultaneous	Sequential	Simultaneous

Styles of Multiplayer

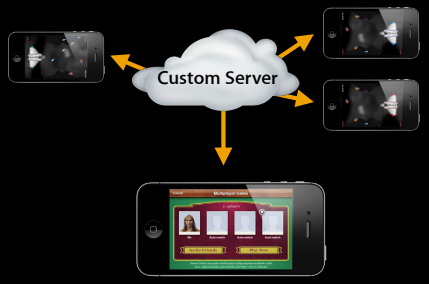
Comparison



Peer-to-Peer



Turn-Based

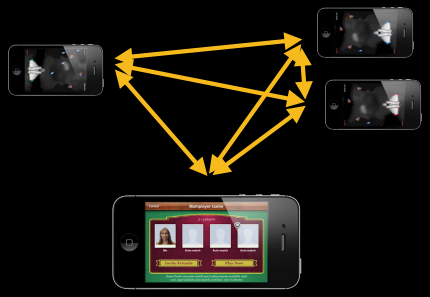


Server-Based

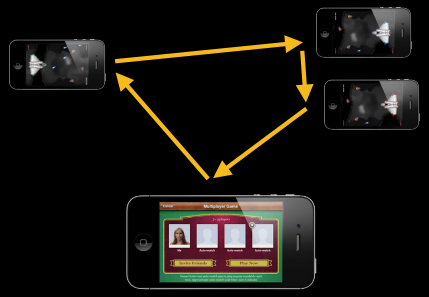
Players	2–4	2–16	2–16
Game Play	Simultaneous	Sequential	Simultaneous
Host	Device or Distributed	Distributed	Developer Server

Styles of Multiplayer

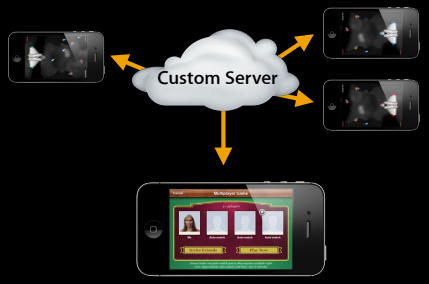
Comparison



Peer-to-Peer



Turn-Based

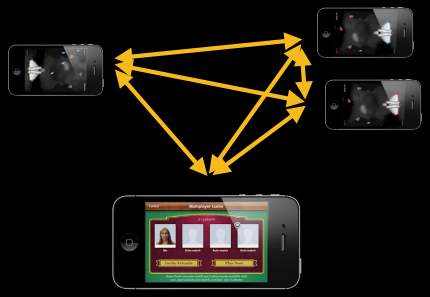


Server-Based

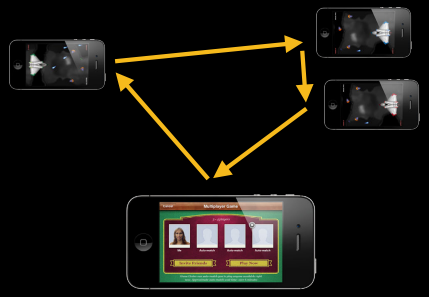
Players	2–4	2–16	2–16
Game Play	Simultaneous	Sequential	Simultaneous
Host	Device or Distributed	Distributed	Developer Server
Communications	Point-to-Point/Broadcast	Point-to-Point	Developer Defined

Styles of Multiplayer

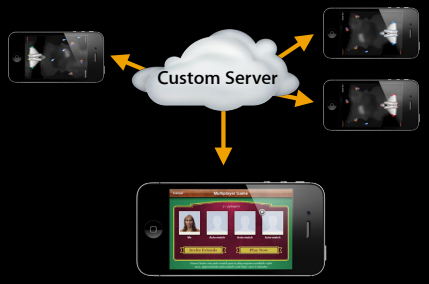
Comparison



Peer-to-Peer



Turn-Based



Server-Based

Players	2–4	2–16	2–16
Game Play	Simultaneous	Sequential	Simultaneous
Host	Device or Distributed	Distributed	Developer Server
Communications	Point-to-Point/Broadcast	Point-to-Point	Developer Defined
Data Transmission	**GameKit API**	**GameKit API**	Developer Defined

Peer-to-Peer Multiplayer

Peer-to-Peer Concepts

Match and play

- Invite or auto-match

Sue

You

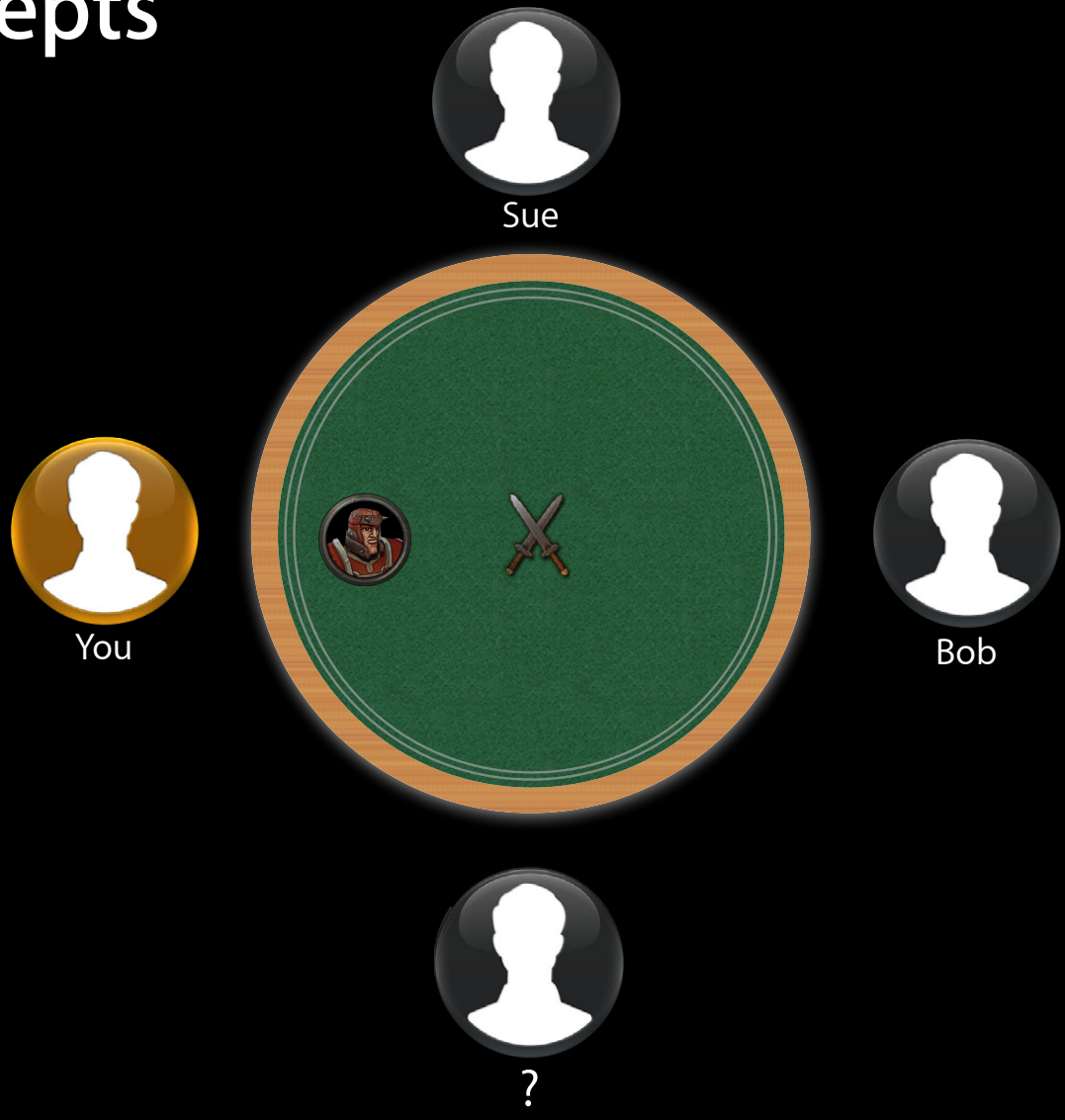
Bob

?

Peer-to-Peer Concepts

Match and play

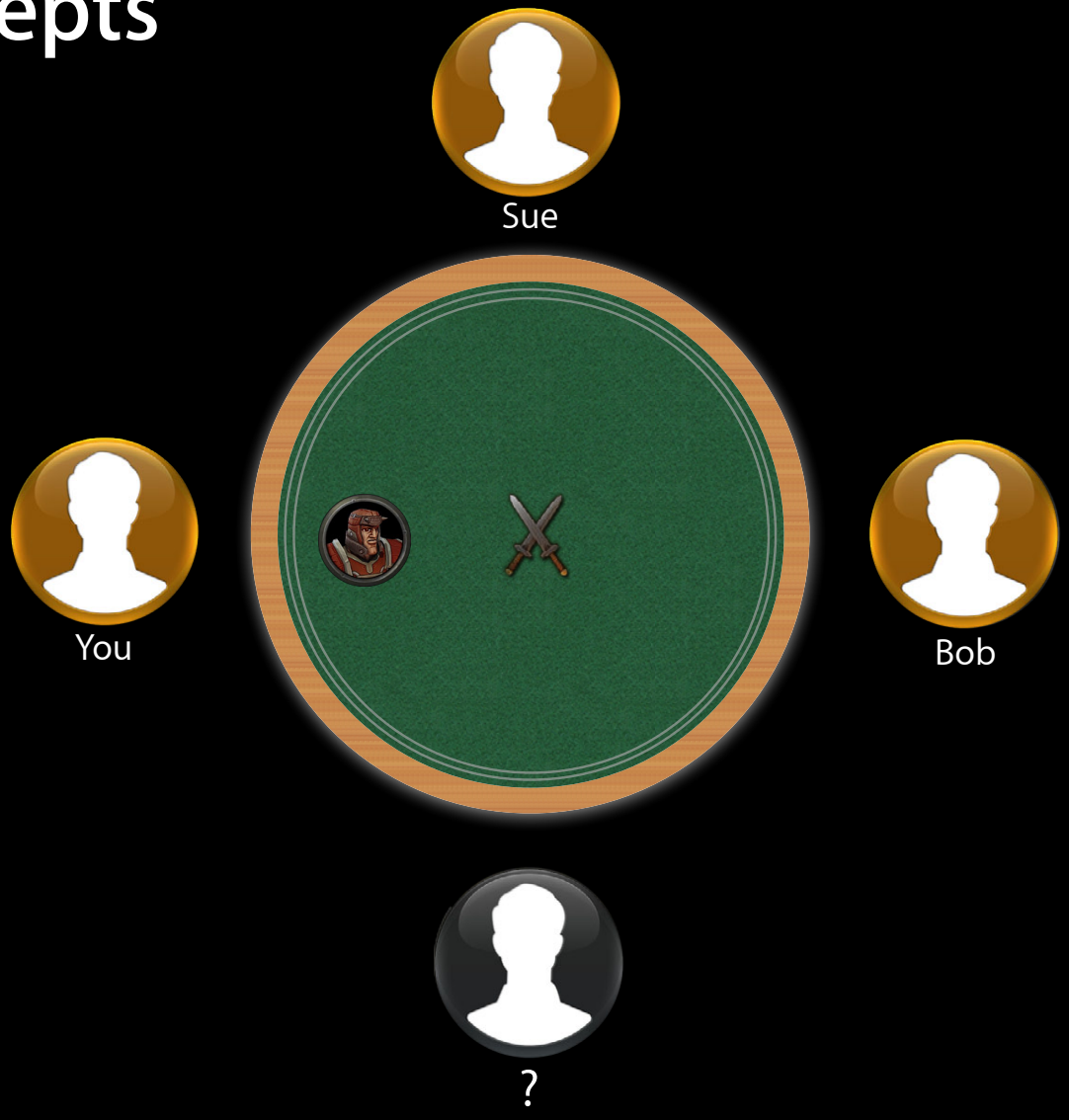
- Invite or auto-match
- Begin matchmaking



Peer-to-Peer Concepts

Match and play

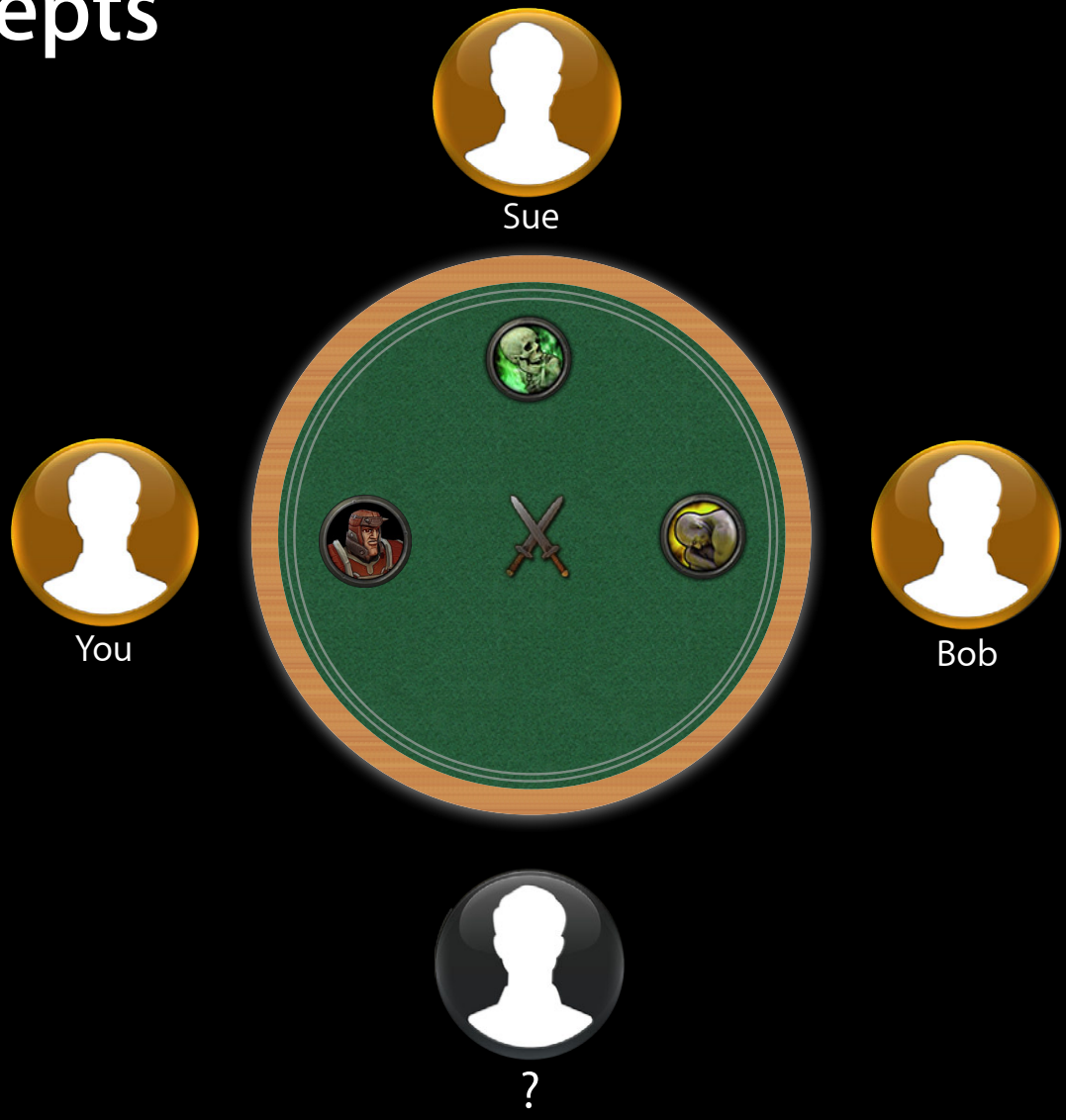
- Invite or auto-match
- Begin matchmaking
- Invite friends



Peer-to-Peer Concepts

Match and play

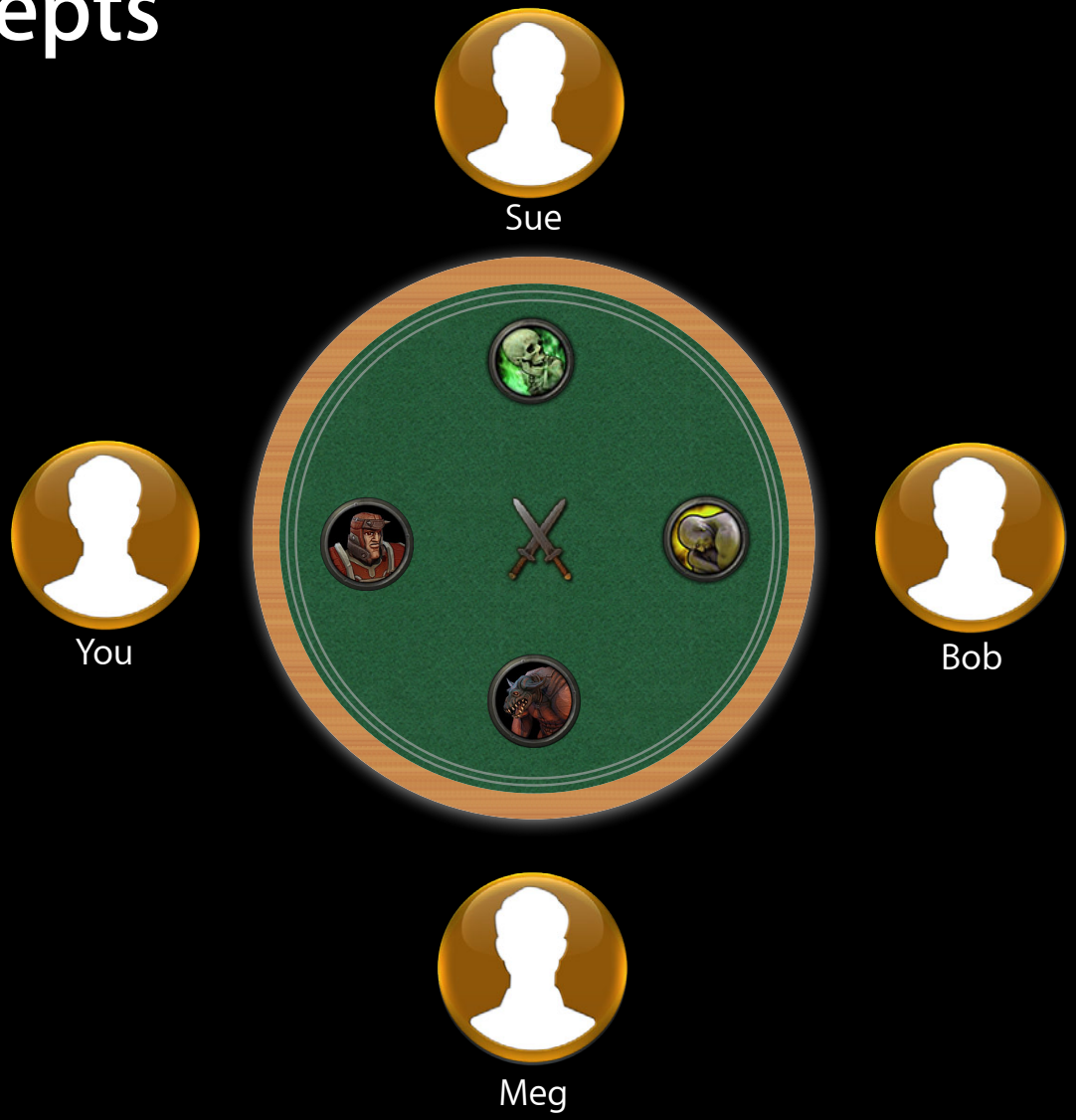
- Invite or auto-match
- Begin matchmaking
- Invite friends
- Friends accept



Peer-to-Peer Concepts

Match and play

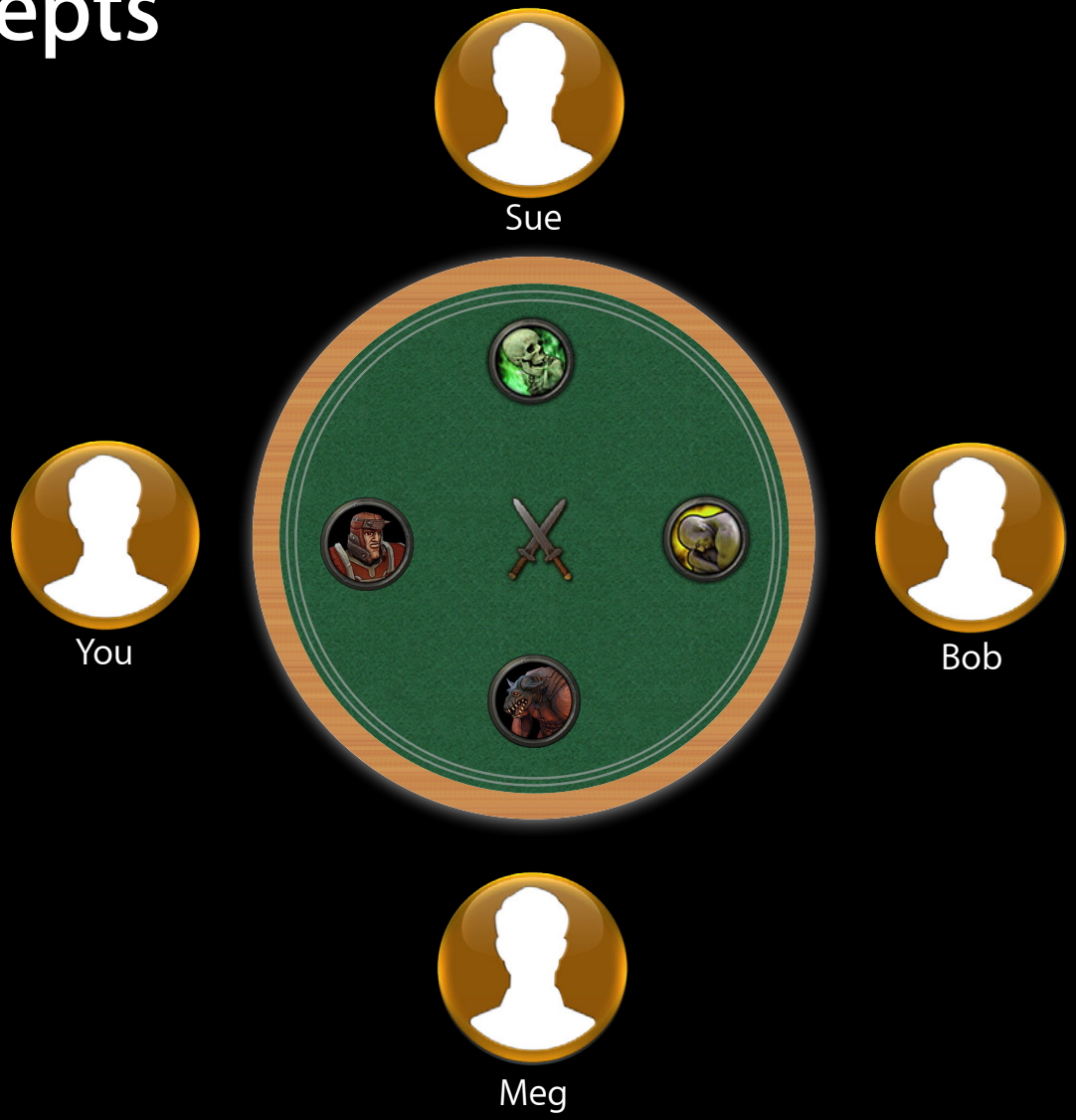
- Invite or auto-match
- Begin matchmaking
- Invite friends
- Friends accept
- Find additional players
- Matched a player



Peer-to-Peer Concepts

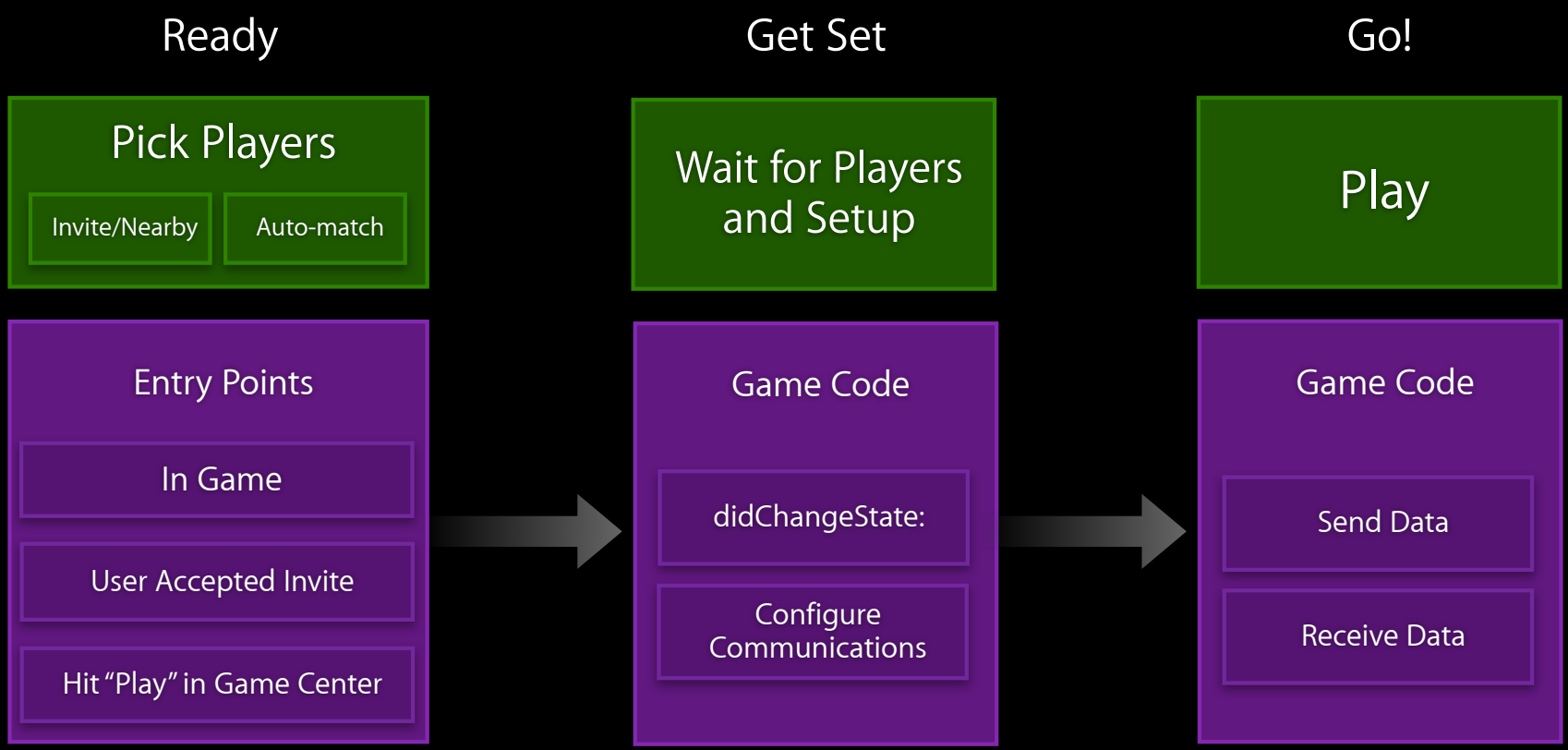
Match and play

- Invite or auto-match
- Begin matchmaking
- Invite friends
- Friends accept
- Find additional players
- Matched a player
- Go!



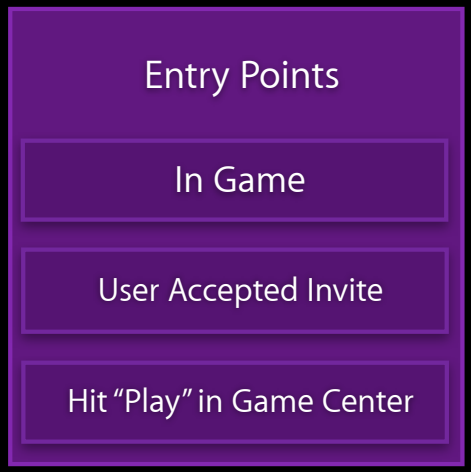
Peer-to-Peer Multiplayer

Tasks



Pick Players

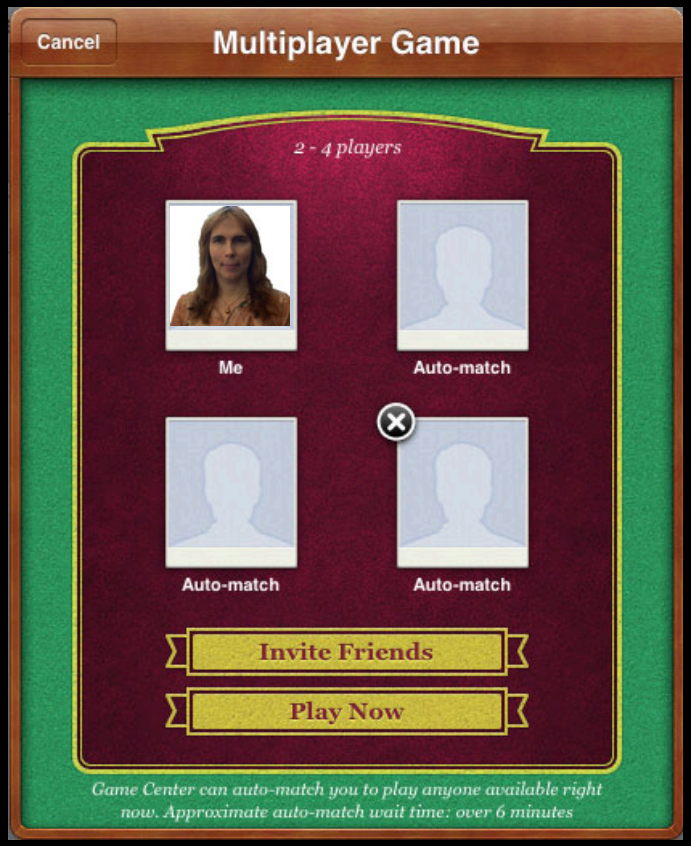
Multiplayer entry points



Pick Players

Multiplayer entry points

- Entry Points
  - In Game
  - User Accepted Invite
  - Hit “Play” in Game Center



Matchmaking UI

Classes

Matchmaking UI

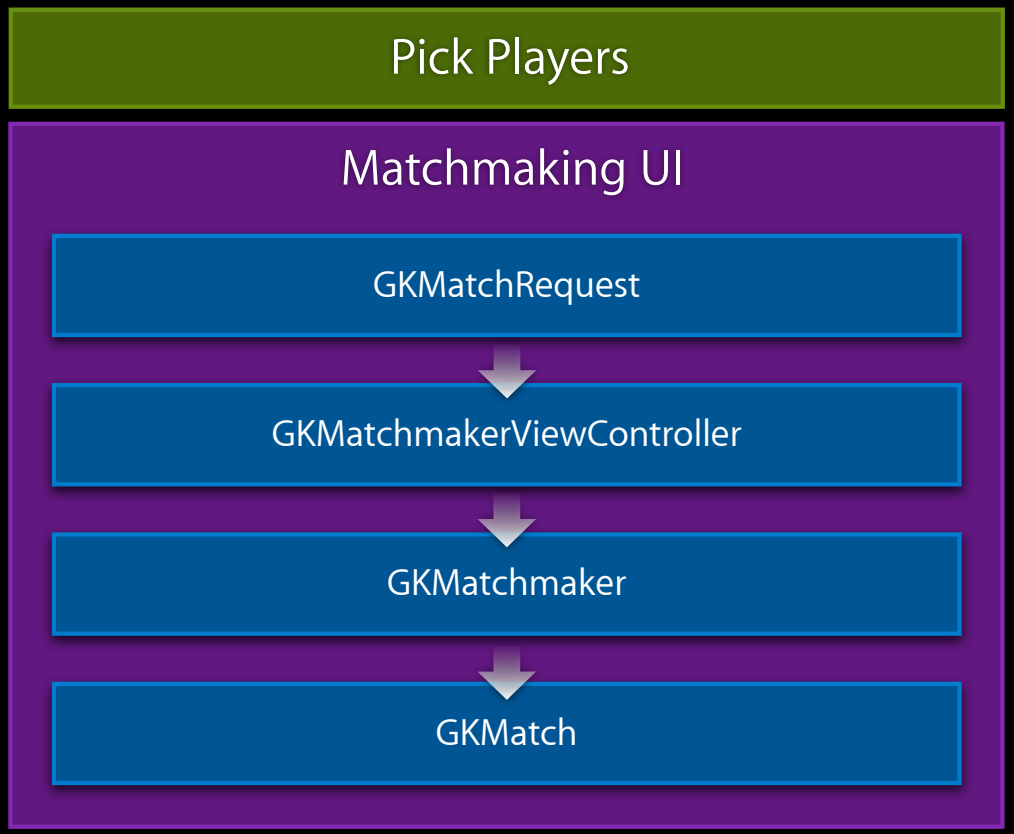
Classes

Matchmaking UI

Classes

Matchmaking UI

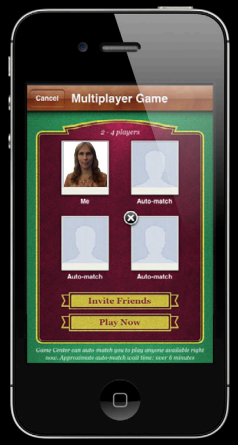
Classes



Matchmaking UI

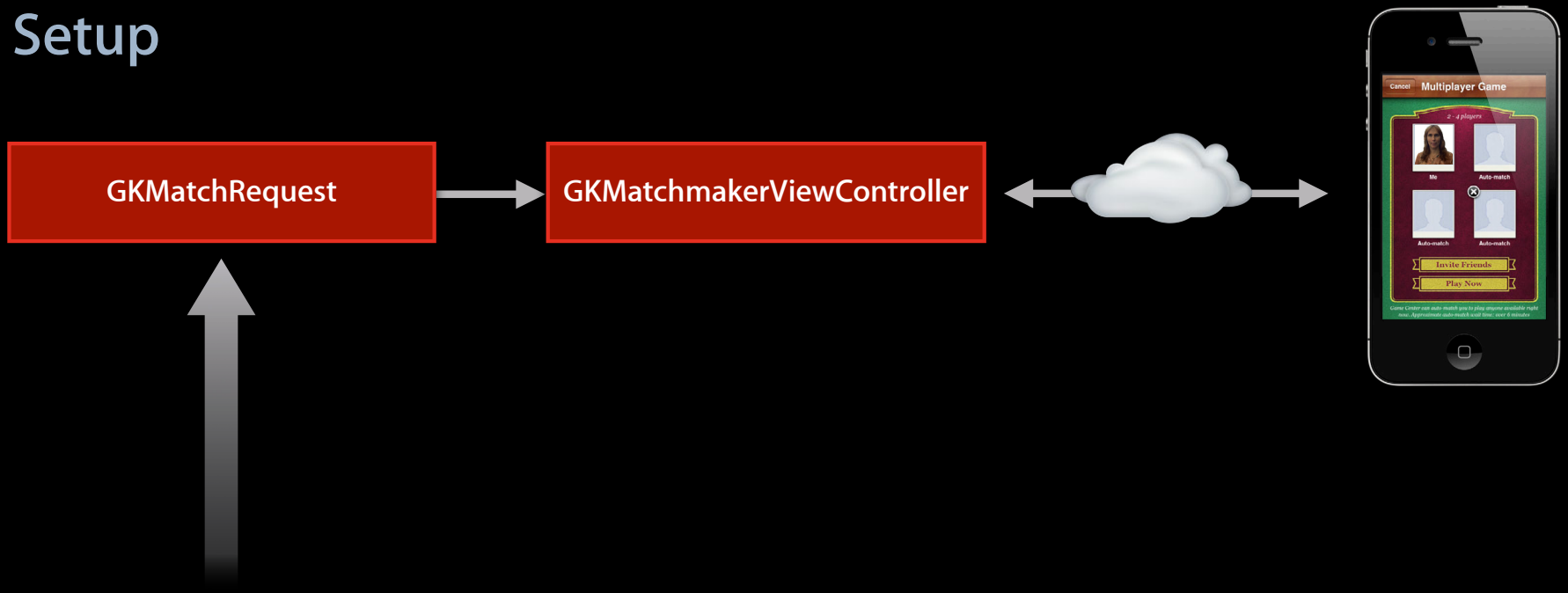
Setup

GKMatchRequest → GKMatchmakerViewController ↔



Matchmaking UI

Setup




```
GKMatchRequest *matchRequest = [[GKMatchRequest alloc] init];
matchRequest.minPlayers = 2;
matchRequest.maxPlayers = 4;
```

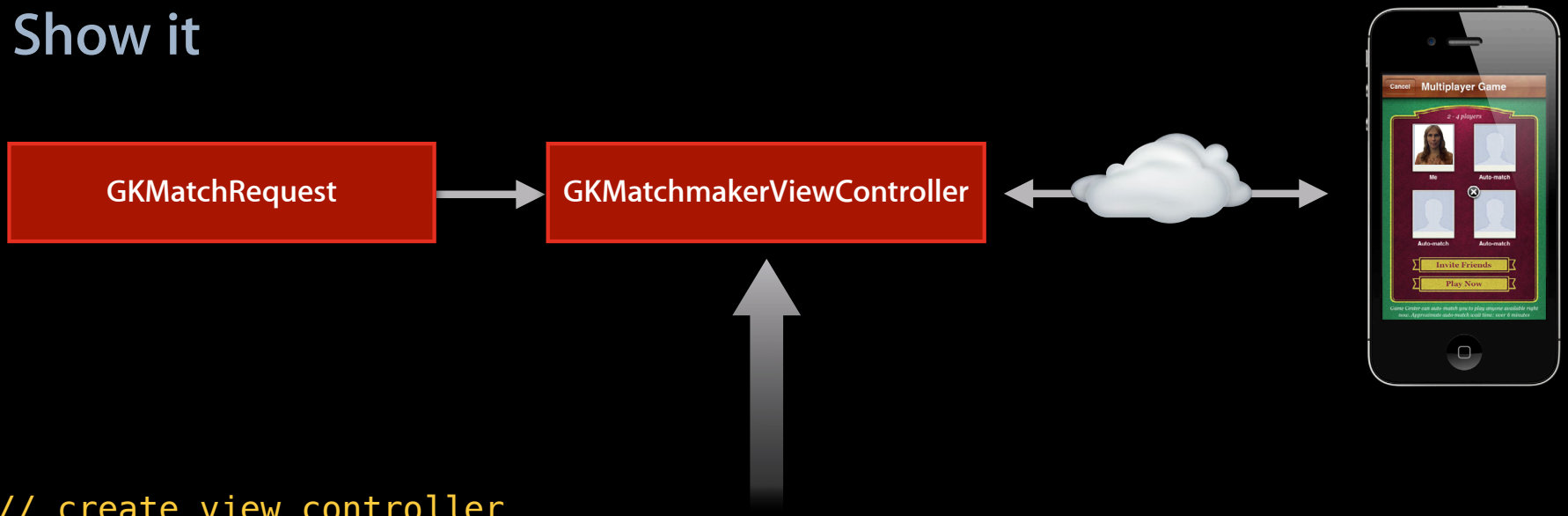
Matchmaking UI

Show it

GKMatchRequest → GKMatchmakerViewController

Matchmaking UI

Show it

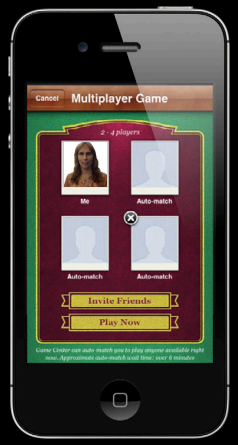



```
// create view controller
GKMatchmakerViewController *controller =
     [[GKMatchmakerViewController alloc] initWithMatchRequest:matchRequest];

// set delegate
controller.matchmakerDelegate = self;

// show it
[self.viewController presentViewController:viewController
                                  animated:YES
                                completion:nil];
```

Matchmaking UI

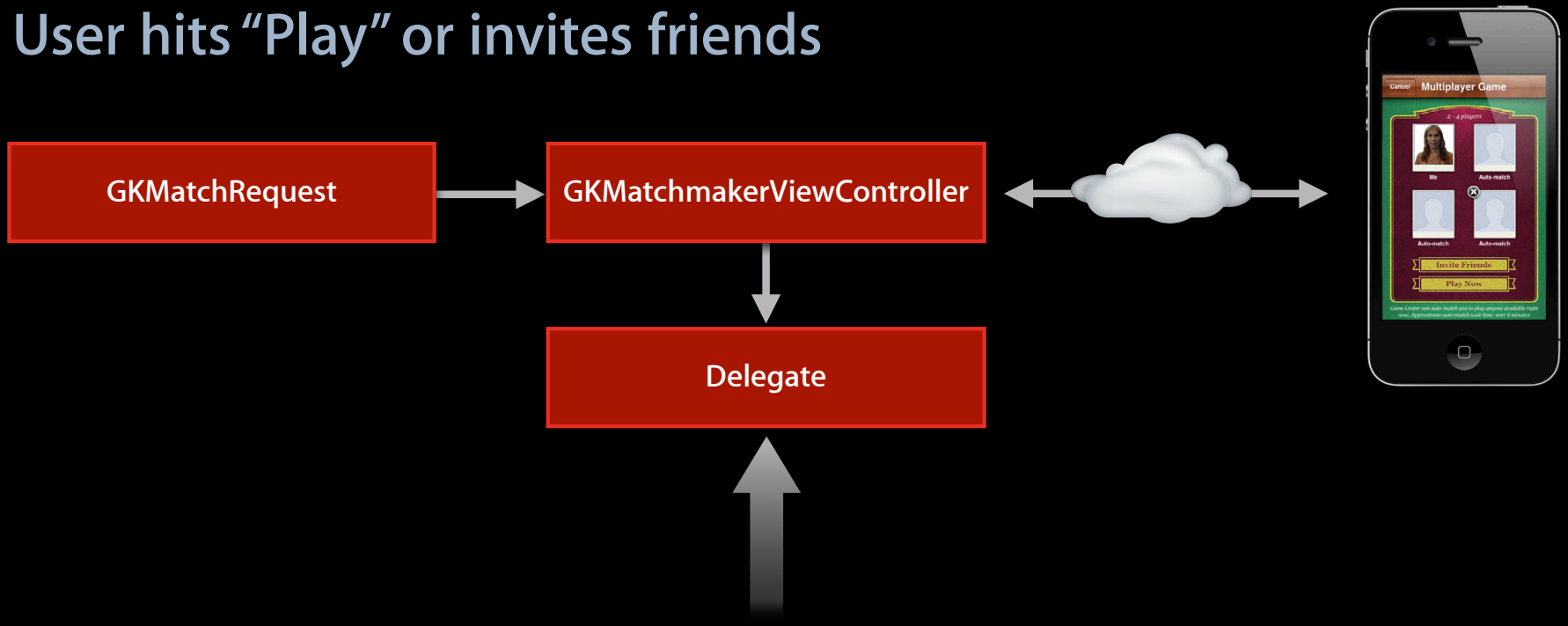
User hits “Play” or invites friends

GKMatchRequest → GKMatchmakerViewController



Matchmaking UI

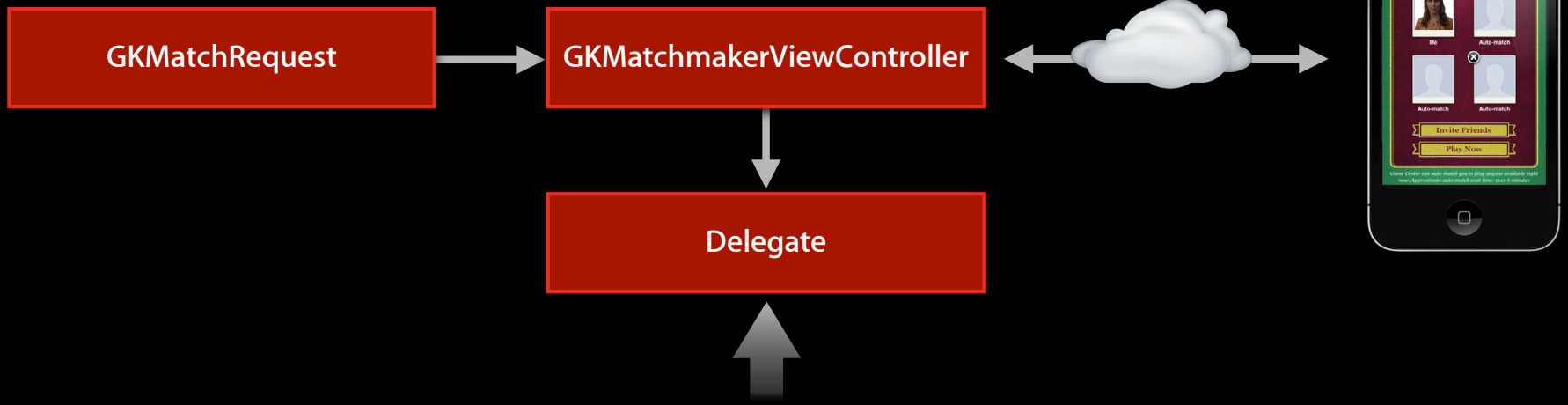
User hits “Play” or invites friends




```
- (void)matchmakerViewController:(GKMatchmakerViewController *)viewController
                    didFindMatch:(GKMatch *)match
{
    // set delegate
    match.delegate = self;

    // Setup match (your code here)
}
```

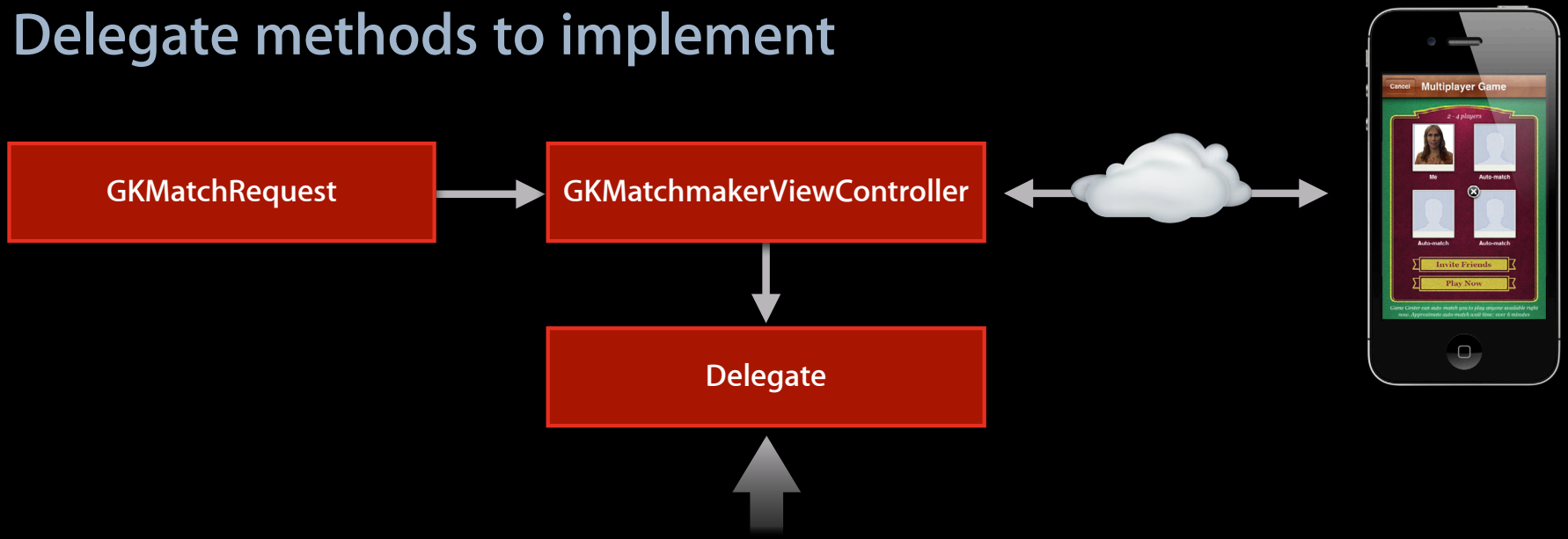
Matchmaking UI

Delegate methods to implement



Matchmaking UI

Delegate methods to implement

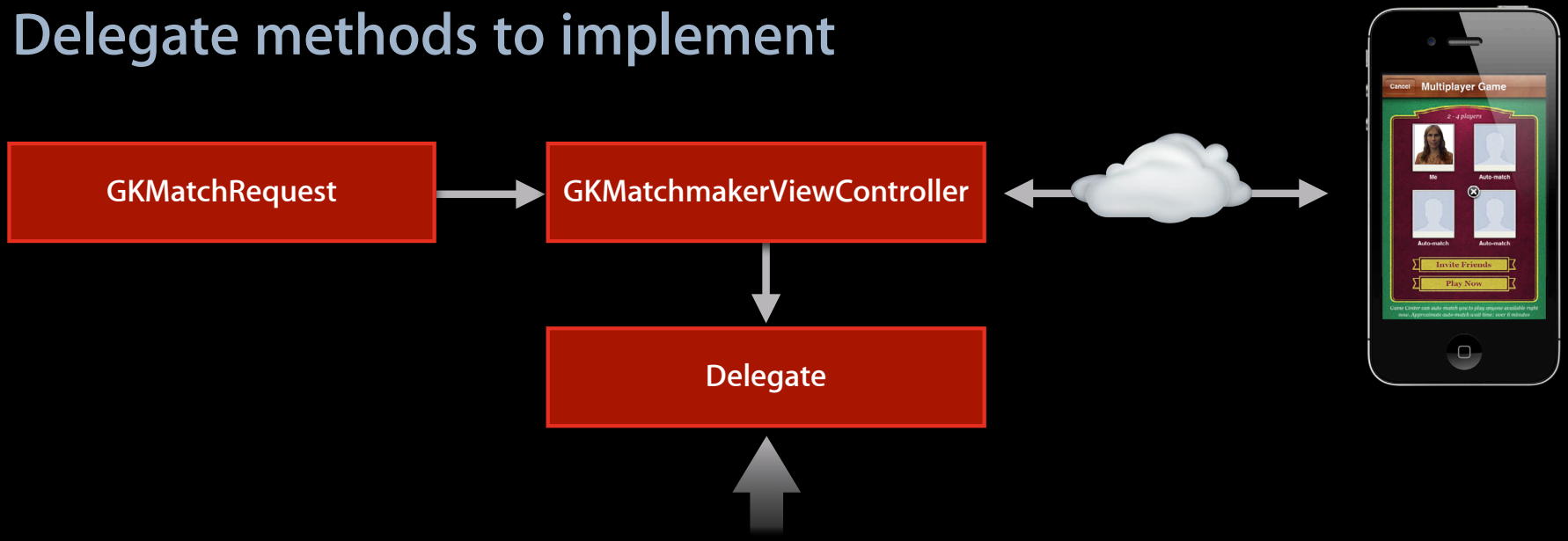


User hits “Play” or Invite Friends

```
- matchmakerViewController:didFindMatch:
```

Matchmaking UI

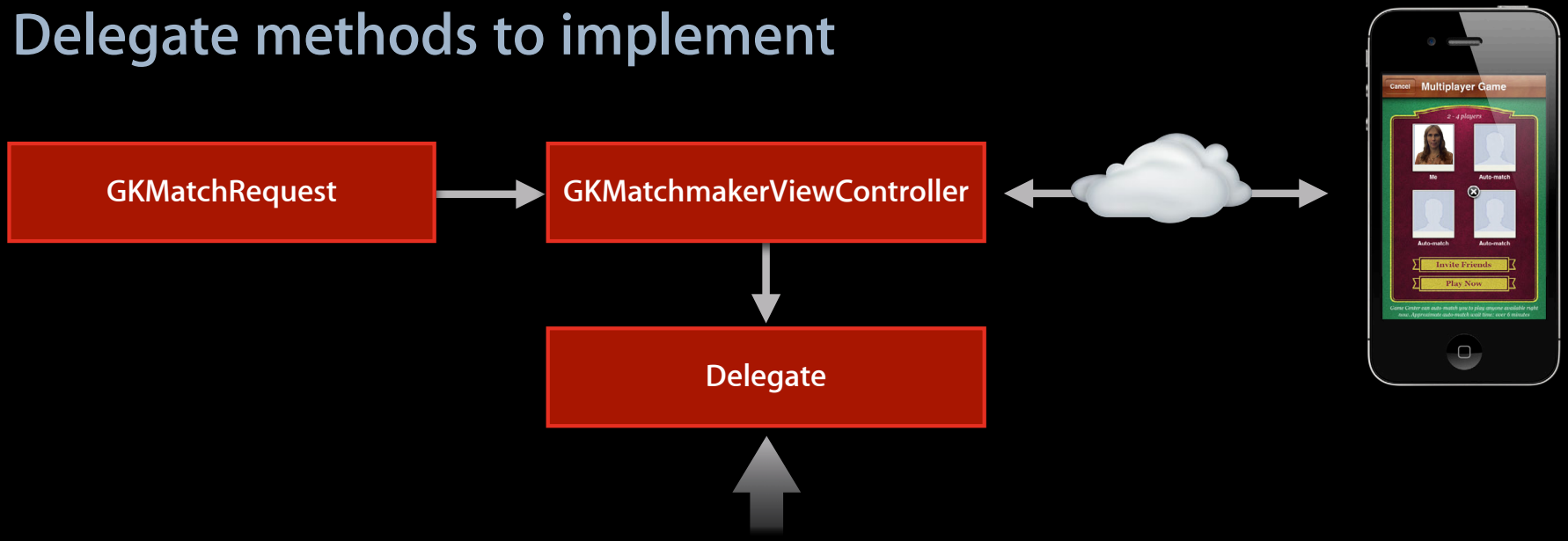
Delegate methods to implement



User hits “Play” or Invite Friends	`- matchmakerViewController:didFindMatch:`
User hits “Cancel”	`- matchmakerViewControllerWasCancelled:`

Matchmaking UI

Delegate methods to implement



User hits “Play” or Invite Friends	`- matchmakerViewController:didFindMatch:`
User hits “Cancel”	`- matchmakerViewControllerWasCancelled:`
Match Failed	`- matchmakerViewController:didFailWithError:`

Invitations

Handling invites

- Implement `inviteHandler` block
  - Times called
    - Recipient has accepted an invite
    - User launches your game from Game Center app
- Install inviteHandler early

Invite Notifications

```
-[GKMatchmaker sharedMatchmaker].inviteHandler = ^(GKInvite *invite, NSArray *players) {
 if (invite) {
  // Create view controller from invite
  GKMatchmakerViewController *controller =
              [[GKMatchmakerViewController alloc] initWithInvite:invite];
  controller.matchmakerDelegate = self;
  [self.viewController presentViewController:viewController animated:YES completion:nil];
  [controller autorelease];

 } else if (players ) {
  // Create view controller from players
        GKMatchmakerViewController *controller =
            [[GKMatchmakerViewController alloc] initWithMatchRequest:self.matchRequest
                                        playersToInvite:players];
  controller.matchmakerDelegate = self;
  [self.viewController presentViewController:viewController animated:YES completion:nil];
  [controller autorelease];
 }
}
```

Invite Notifications

```
-[GKMatchmaker sharedMatchmaker].inviteHandler = ^(GKInvite *invite, NSArray *players) {
 if (invite) {
  // Create view controller from invite
  GKMatchmakerViewController *controller =
              [[GKMatchmakerViewController alloc] initWithInvite:invite];
  controller.matchmakerDelegate = self;
  [self.viewController presentViewController:viewController animated:YES completion:nil];
  [controller autorelease];

 } else if (players ) {
  // Create view controller from players
        GKMatchmakerViewController *controller =
            [[GKMatchmakerViewController alloc] initWithMatchRequest:self.matchRequest
                                        playersToInvite:players];
  controller.matchmakerDelegate = self;
  [self.viewController presentViewController:viewController animated:YES completion:nil];
  [controller autorelease];
 }
}
```

Invite Notifications

```
-[GKMatchmaker sharedMatchmaker].inviteHandler = ^(GKInvite *invite, NSArray *players) {
 if (invite) {
  // Create view controller from invite
  GKMatchmakerViewController *controller =
              [[GKMatchmakerViewController alloc] initWithInvite:invite];
  controller.matchmakerDelegate = self;
  [self.viewController presentViewController:viewController animated:YES completion:nil];
  [controller autorelease];

 } else if (players ) {
  // Create view controller from players
        GKMatchmakerViewController *controller =
            [[GKMatchmakerViewController alloc] initWithMatchRequest:self.matchRequest
                                        playersToInvite:players];
  controller.matchmakerDelegate = self;
  [self.viewController presentViewController:viewController animated:YES completion:nil];
  [controller autorelease];
 }
}
```

Invite Notifications

```
-[GKMatchmaker sharedMatchmaker].inviteHandler = ^(GKInvite *invite, NSArray *players) {
 if (invite) {
  // Create view controller from invite
  GKMatchmakerViewController *controller =
              [[GKMatchmakerViewController alloc] initWithInvite:invite];
  controller.matchmakerDelegate = self;
  [self.viewController presentViewController:viewController animated:YES completion:nil];
  [controller autorelease];

 } else if (players ) {
  // Create view controller from players
        GKMatchmakerViewController *controller =
            [[GKMatchmakerViewController alloc] initWithMatchRequest:self.matchRequest
                                                playersToInvite:players];
  controller.matchmakerDelegate = self;
  [self.viewController presentViewController:viewController animated:YES completion:nil];
  [controller autorelease];
 }
}
```

Invite Notifications

```
-[GKMatchmaker sharedMatchmaker].inviteHandler = ^(GKInvite *invite, NSArray *players) {
 if (invite) {
  // Create view controller from invite
  GKMatchmakerViewController *controller =
              [[GKMatchmakerViewController alloc] initWithInvite:invite];
  controller.matchmakerDelegate = self;
  [self.viewController presentViewController:viewController animated:YES completion:nil];
  [controller autorelease];

 } else if (players ) {
  // Create view controller from players
        GKMatchmakerViewController *controller =
            [[GKMatchmakerViewController alloc] initWithMatchRequest:self.matchRequest
                                        playersToInvite:players];
  controller.matchmakerDelegate = self;
  [self.viewController presentViewController:viewController animated:YES completion:nil];
  [controller autorelease];
 }
}
```

Matchmaker UI

Summary

- Create match request
- Present standard UI
- Handle invites
  - May be called at app launch
  - Called any time even during the game
- Same UI works if you want to host yourself
- Programmatic auto-match is easy

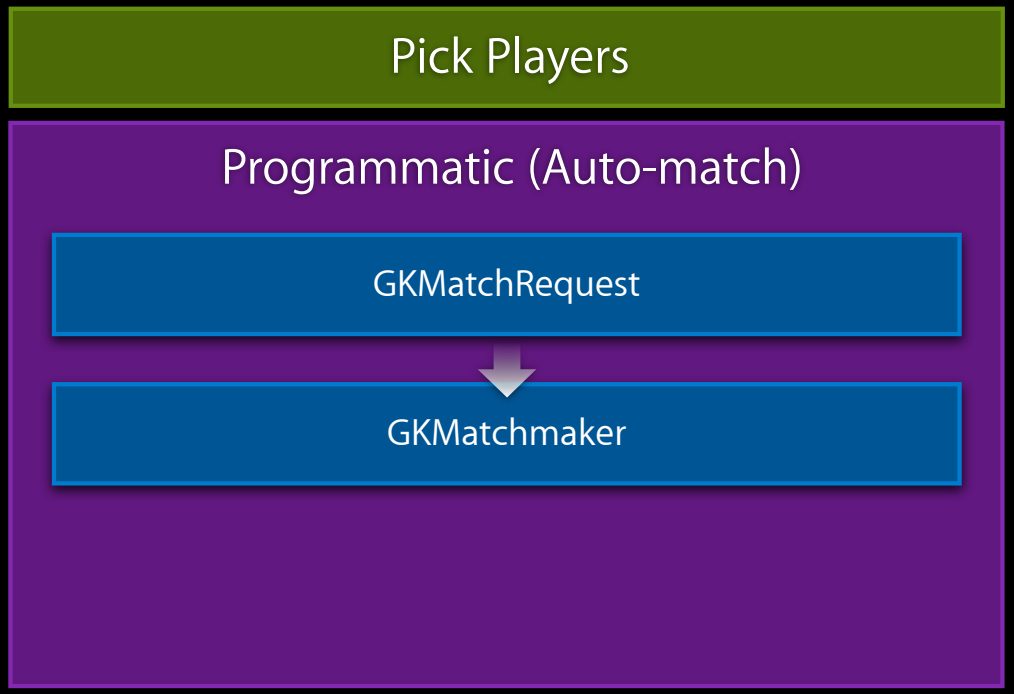
Programmatic Matchmaking

Programmatic Matchmaking

Quick way to play

Programmatic Matchmaking

Quick way to play



Programmatic Matchmaking

Quick way to play

Auto-Match

Quick and easy

Auto-Match

Quick and easy


```
GKMatchmaker *matchmaker = [GKMatchmaker sharedMatchmaker];

[matchmaker findMatchForRequest:myMatchRequest
          withCompletionHandler:^(GKMatch *match, NSError *error) {
  if (error) {
    // Handle error
  }
  else {
    // get ready to play
  }
}];
```

Local Multiplayer

Find nearby players running your game


```
- (void)startLookingForPlayers
    // get shared matchmaker
    GKMatchmaker *matchmaker = [GKMatchmaker sharedMatchmaker];

    // look for nearby players
    [matchMaker startBrowsingForNearbyPlayersWithReachableHandler:
            ^(NSString *playerID, BOOL reachable) {
        // add to your array
        [self.playersToInvite addObject:playerID];
    ];
}

- (void)stopLookingForPlayers
{
    // stop looking nearby players
    [[GKMatchmaker sharedMatchmaker] stopBrowsingForNearbyPlayers];
}
```

Local Multiplayer

Find nearby players running your game


```
- (void)startLookingForPlayers
    // get shared matchmaker
    GKMatchmaker *matchmaker = [GKMatchmaker sharedMatchmaker];

    // look for nearby players
    [matchMaker startBrowsingForNearbyPlayersWithReachableHandler:
            ^(NSString *playerID, BOOL reachable) {
        // add to your array
        [self.playersToInvite addObject:playerID];
    ];
}

- (void)stopLookingForPlayers
{
    // stop looking nearby players
    [[GKMatchmaker sharedMatchmaker] stopBrowsingForNearbyPlayers];
}
```

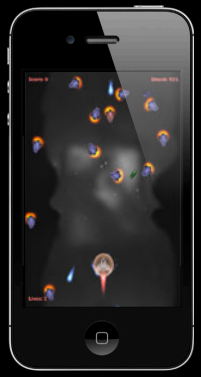
Local Multiplayer

Find nearby players running your game


```
- (void)startLookingForPlayers
    // get shared matchmaker
    GKMatchmaker *matchmaker = [GKMatchmaker sharedMatchmaker];

    // look for nearby players
    [matchMaker startBrowsingForNearbyPlayersWithReachableHandler:
            ^(NSString *playerID, BOOL reachable) {
        // add to your array
        [self.playersToInvite addObject:playerID];
    ];
}

- (void)stopLookingForPlayers
{
    // stop looking nearby players
    [[GKMatchmaker sharedMatchmaker] stopBrowsingForNearbyPlayers];
}
```

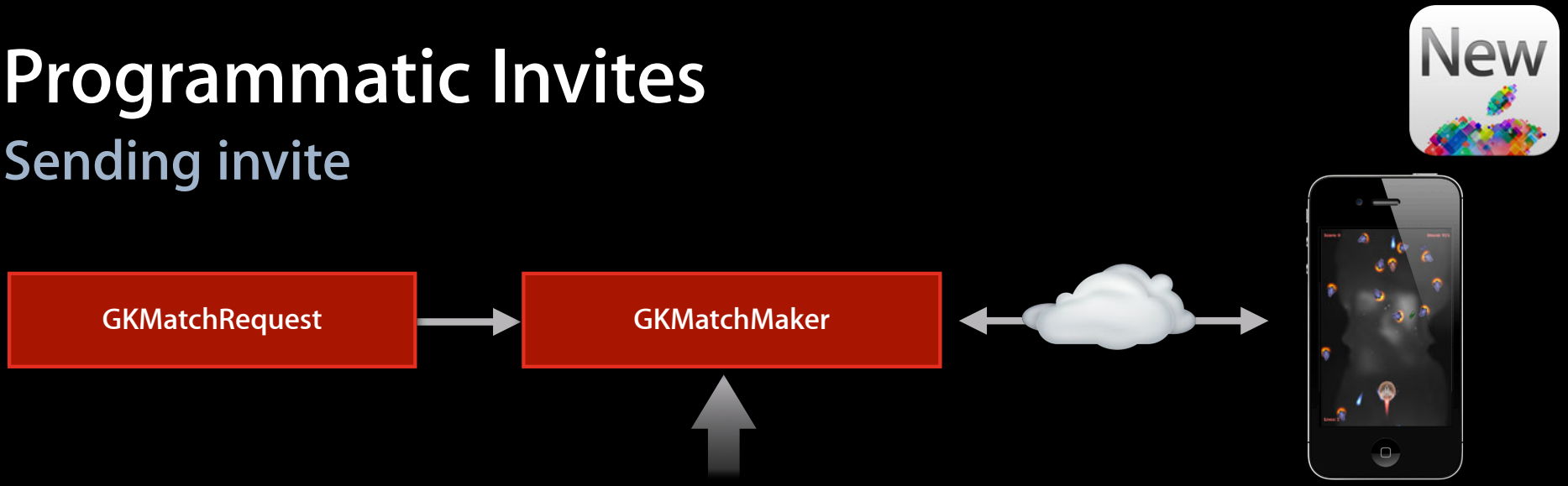
Programmatic Invites

Sending invite



Programmatic Invites

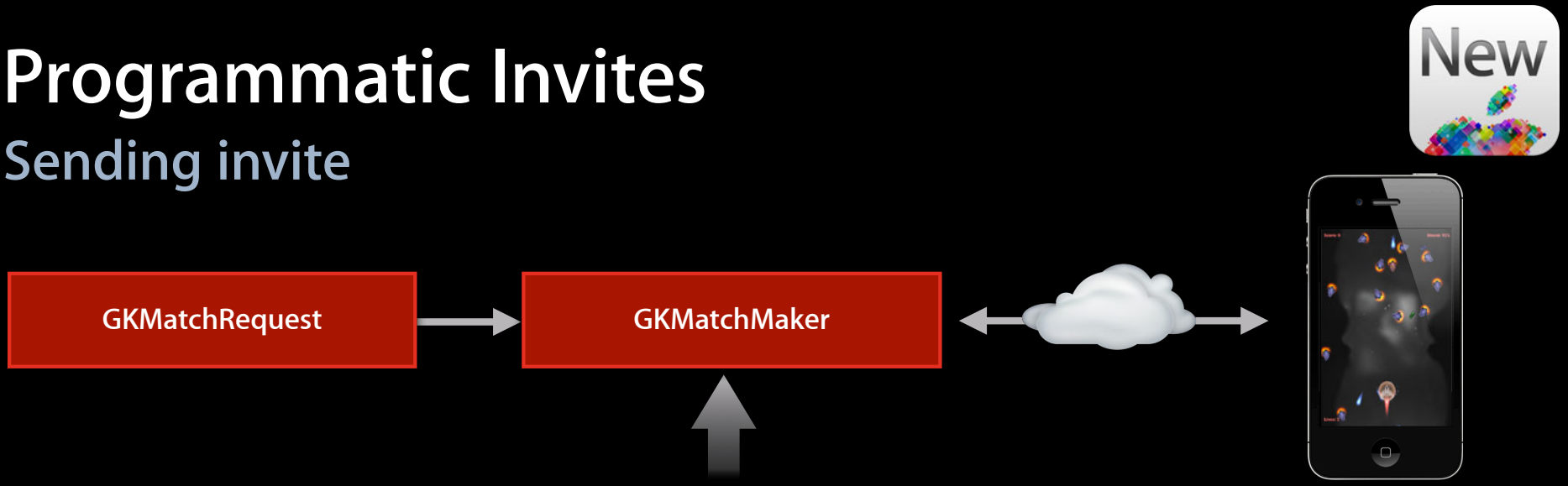
Sending invite




```
GKMatchmaker *matchmaker = [GKMatchmaker sharedMatchmaker];

// setup match request
myMatchRequest.playersToInvite = self.playersToInvite
myMatchRequest.inviteMessage = @"Let's play!"
myMatchRequest.responseHandler = self.responseHandler;

// find a match
[matchmaker findMatchForRequest:myMatchRequest
            withCompletionHandler:^(GKMatch *match, NSError *error) {
    match.delegate = self;
    self.match = match;

}];
```

Programmatic Invites

Sending invite




```
GKMatchmaker *matchmaker = [GKMatchmaker sharedMatchmaker];

// setup match request
myMatchRequest.playersToInvite = self.playersToInvite
myMatchRequest.inviteMessage = @"Let's play!"
myMatchRequest.responseHandler = self.responseHandler;
```

```
// find a match
[matchmaker findMatchForRequest:myMatchRequest
            withCompletionHandler:^(GKMatch *match, NSError *error) {
    match.delegate = self;
    self.match = match;

}];
```

Programmatic Invites

Handle response from invitee

Programmatic Invites

Handle response from invitee


```
// setup match request
myMatchRequest.playersToInvite = self.playersToInvite
myMatchRequest.inviteMessage = @”Let’s play!”

myMatchRequest.responseHandler = ^(NSString *playerID, GKInviteeResponse
response) {
    // Mark player as accepted in your UI
     if (self.haveEnoughPlayers)
          [matchmaker finishMatchmakingForMatch];
}];

// find a match
[matchmaker findMatchForRequest:myMatchRequest
```

Programmatic Invites

Handle response from invitee


```
// setup match request
myMatchRequest.playersToInvite = self.playersToInvite
myMatchRequest.inviteMessage = @"Let's play!"

myMatchRequest.responseHandler = ^(NSString *playerID, GKInviteeResponse
response) {
    // Mark player as accepted in your UI
     if (self.haveEnoughPlayers)
          [matchmaker finishMatchmakingForMatch];
}];

// find a match
[matchmaker findMatchForRequest:myMatchRequest
```

Player Groups

Pick a track

- Match players based on game defined groups

```
GKMatchRequest *matchRequest = [[GKMatchRequest alloc] init];
matchRequest.playerGroup = FigureEightTrack;
```

- Other ideas for player group assignment
  - Difficulty (easy, normal, hard)
  - Game type (death match, capture the flag, team-fortress)
- API to check player group activity

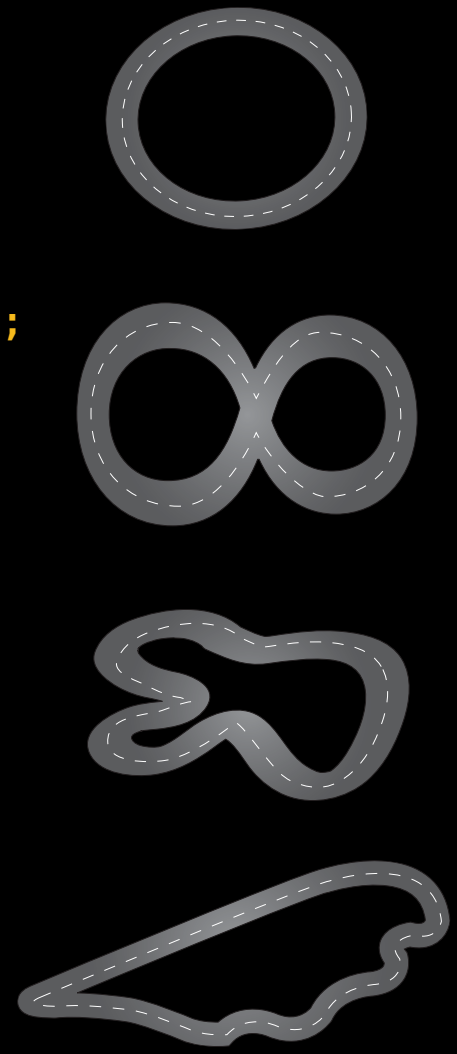
Player Groups

Pick a track

- Match players based on game defined groups

```
GKMatchRequest *matchRequest = [[GKMatchRequest alloc] init];
matchRequest.playerGroup = FigureEightTrack;
```

- Other ideas for player group assignment
  - Difficulty (easy, normal, hard)
  - Game type (death match, capture the flag, team-fortress)
- API to check player group activity



Player Attributes

Pick a side

- Specify the player's role
- 32-bit unsigned integer
- Logical OR operation
- Chosen based on player characteristics
  - Chess (white vs. black)
  - Role-playing (fighter, cleric, mage, thief)
  - Sports (goalie, forward, defense)

Player Attributes

Pick a side

- Specify the player's role
- 32-bit unsigned integer
- Logical OR operation
- Chosen based on player characteristics
  - Chess (white vs. black)
  - Role-playing (fighter, cleric, mage, thief)
  - Sports (goalie, forward, defense)

Player Attributes

Pick a side

- Specify the player's role
- 32-bit unsigned integer
- Logical OR operation
- Chosen based on player characteristics
  - Chess (white vs. black)
  - Role-playing (fighter, cleric, mage, thief)
  - Sports (goalie, forward, defense)

Black

0xFFFF0000

Player Attributes

Pick a side

- Specify the player's role
- 32-bit unsigned integer
- Logical OR operation
- Chosen based on player characteristics
  - Chess (white vs. black)
  - Role-playing (fighter, cleric, mage, thief)
  - Sports (goalie, forward, defense)

Black

0xFFFF0000

White

0x0000FFFF

Player Attributes

Pick a side

- Specify the player's role
- 32-bit unsigned integer
- Logical OR operation
- Chosen based on player characteristics
  - Chess (white vs. black)
  - Role-playing (fighter, cleric, mage, thief)
  - Sports (goalie, forward, defense)

Black	Don't Care	White
0xFFFF0000	0xFFFFFFFF	0x0000FFFF

Player Attributes

Pick a side

- Specify the player's role
- 32-bit unsigned integer
- Logical OR operation
- Chosen based on player characteristics
  - Chess (white vs. black)
  - Role-playing (fighter, cleric, mage, thief)
  - Sports (goalie, forward, defense)

Black	Don't Care	White
0xFFFF0000	0xFFFFFFFF	0x0000FFFF

Match players that OR to

0xFFFFFFFF

Player Attributes

Pick a side

- Specify the player's role
- 32-bit unsigned integer
- Logical OR operation
- Chosen based on player characteristics
  - Chess (white vs. black)
  - Role-playing (fighter, cleric, mage, thief)
  - Sports (goalie, forward, defense)

Black	Don't Care	White
0xFFFF0000	0xFFFFFFFF	0x0000FFFF

Match players that OR to

0xFFFFFFFF

Auto-Match

Summary

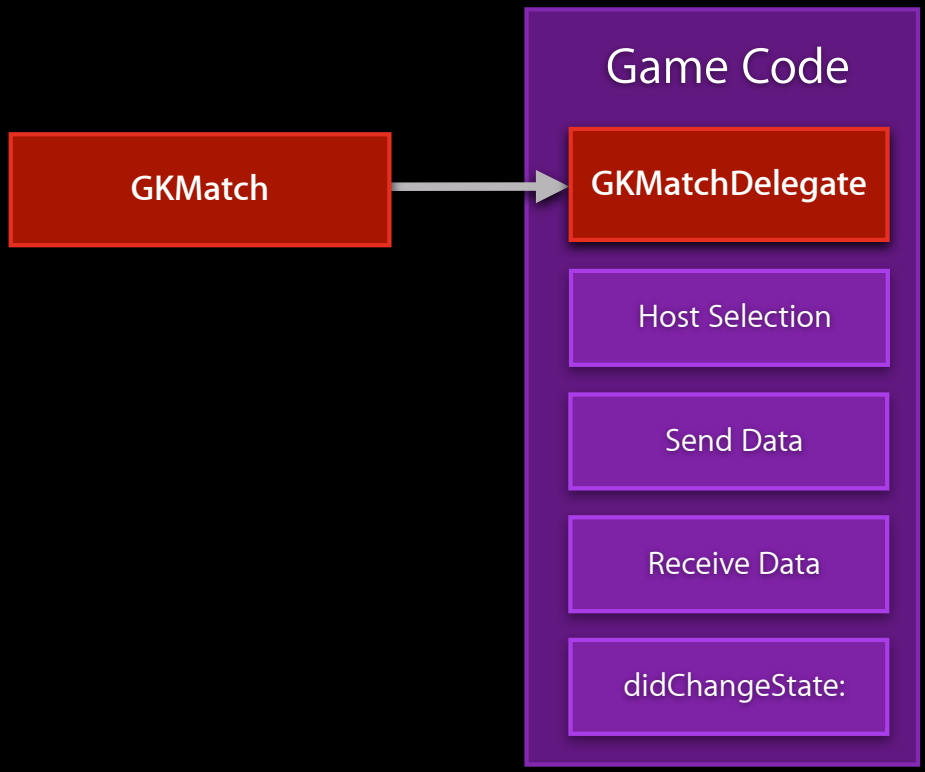
- Create a match request
  - Use player groups and attributes as desired
- Request match
- Wait for players to connect
- Play!

Peer-to-Peer Communications

Peer-to-Peer Communications

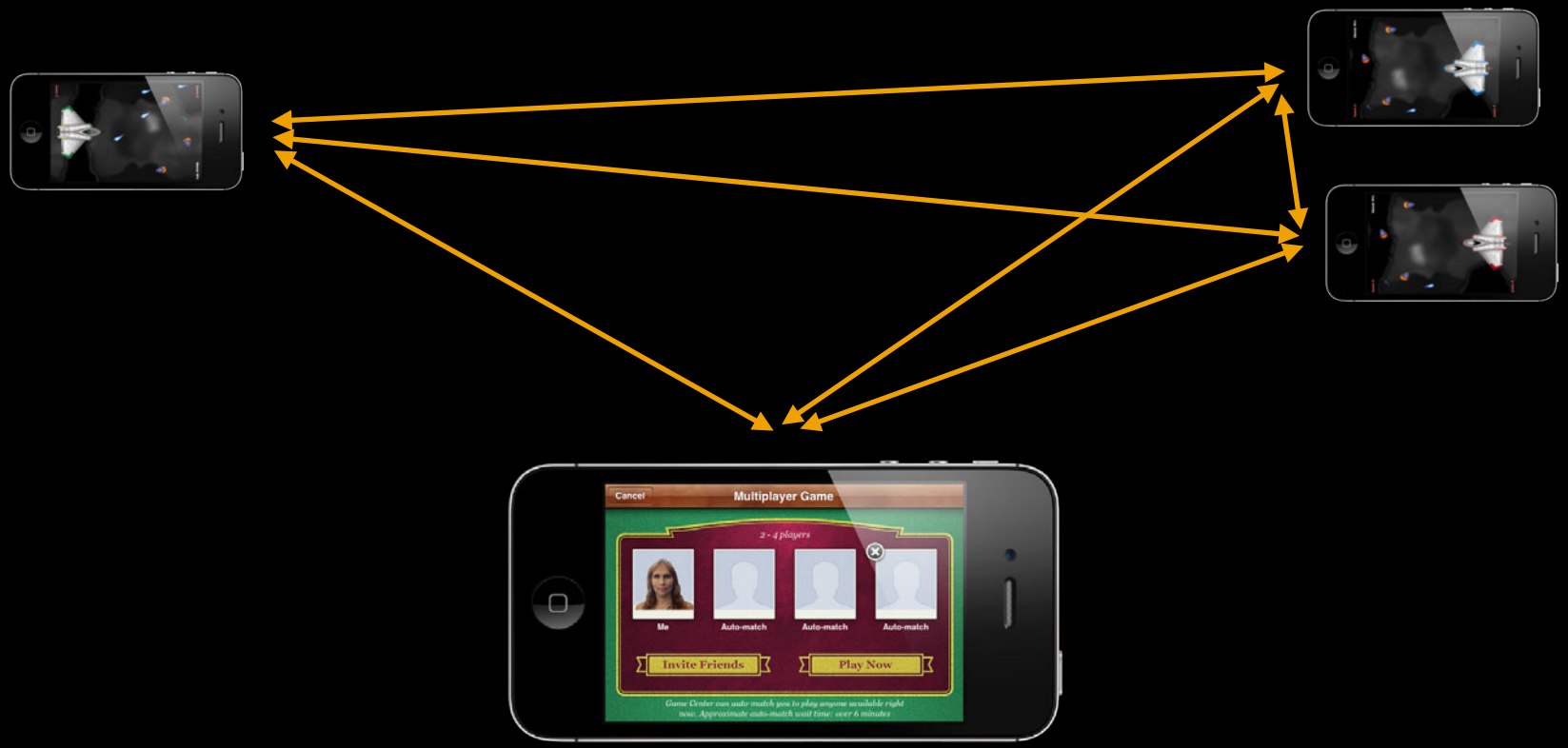
Overview

- Networking strategy
  - Host selection
- Send data
- Receive data
- State changes



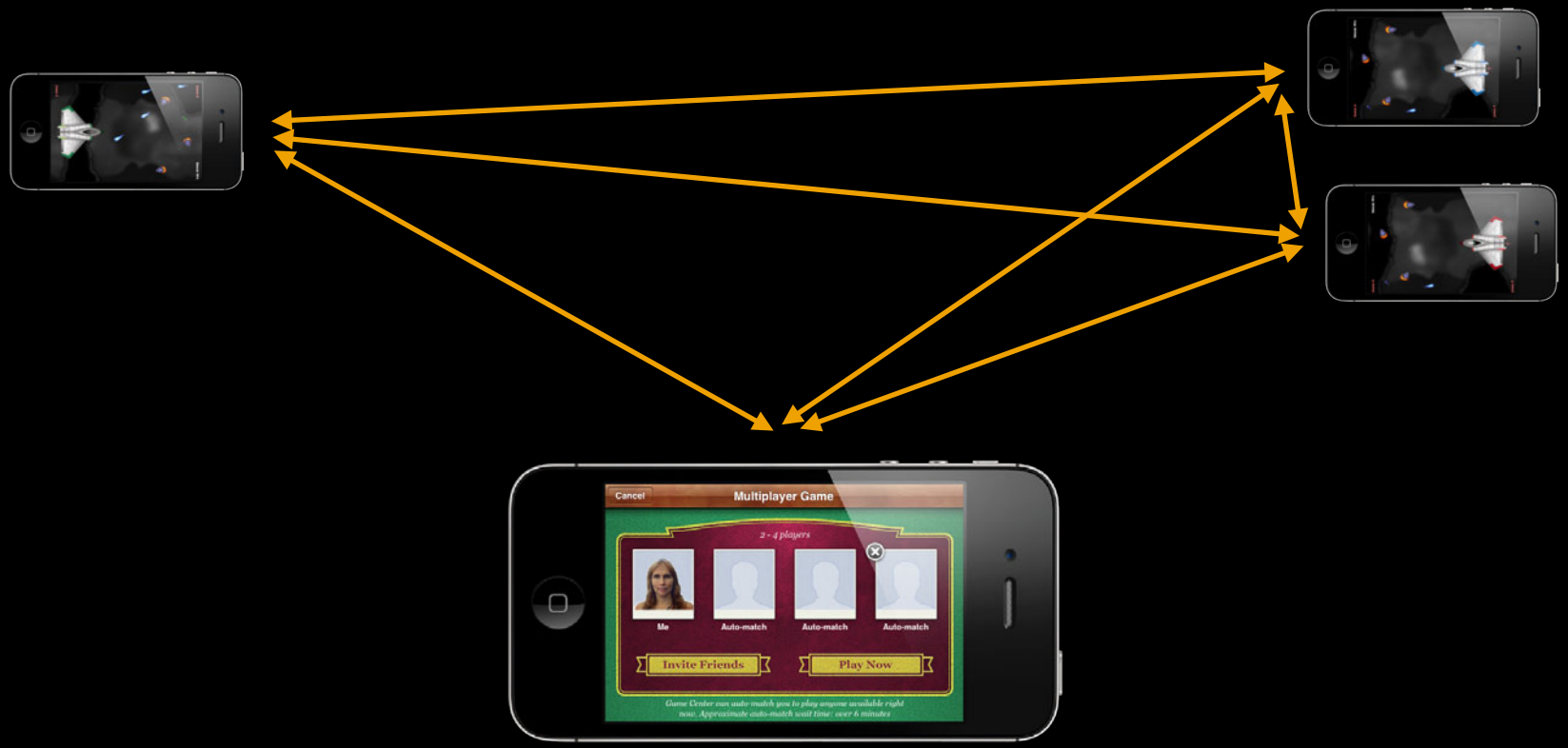
Network Strategy

Full mesh

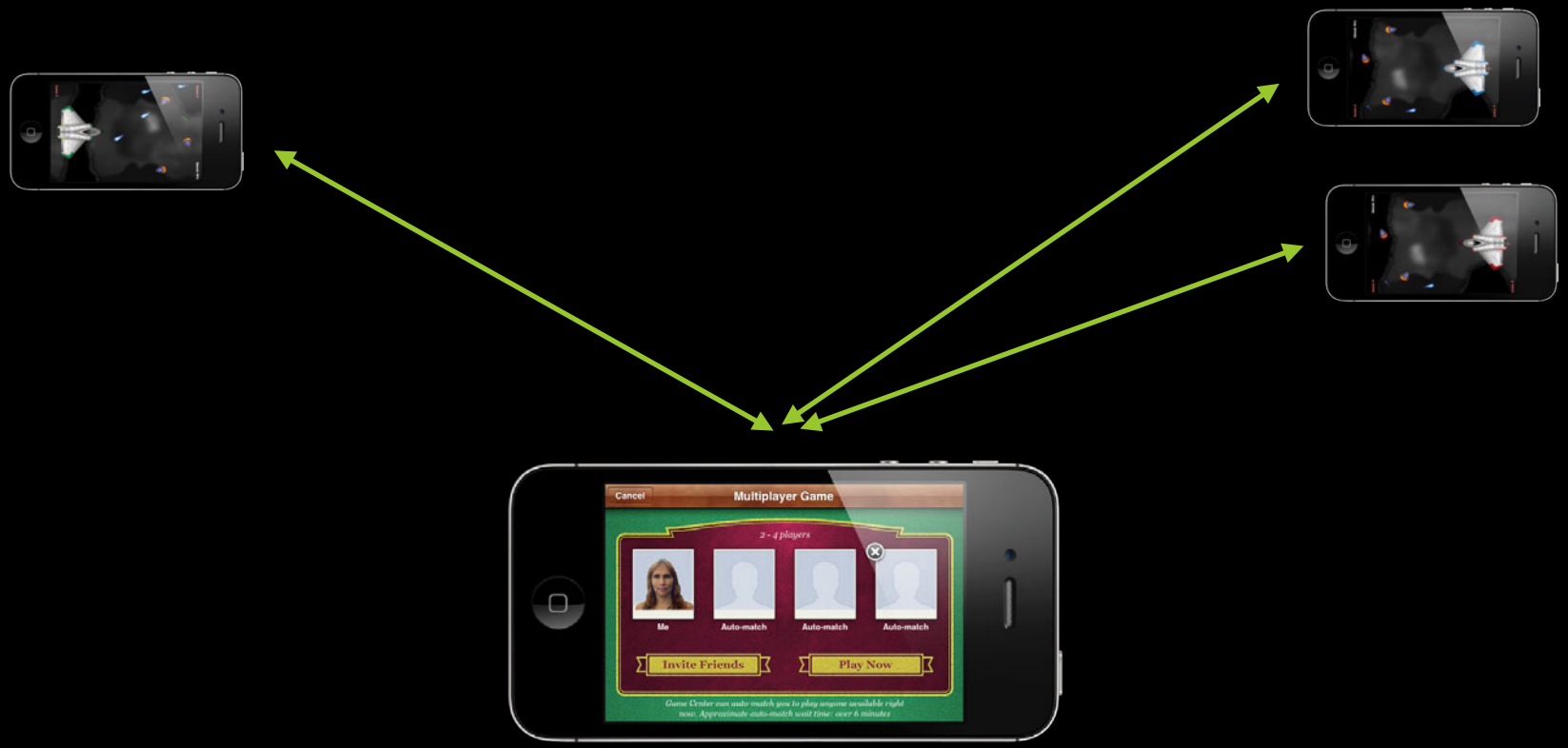


Network Strategy

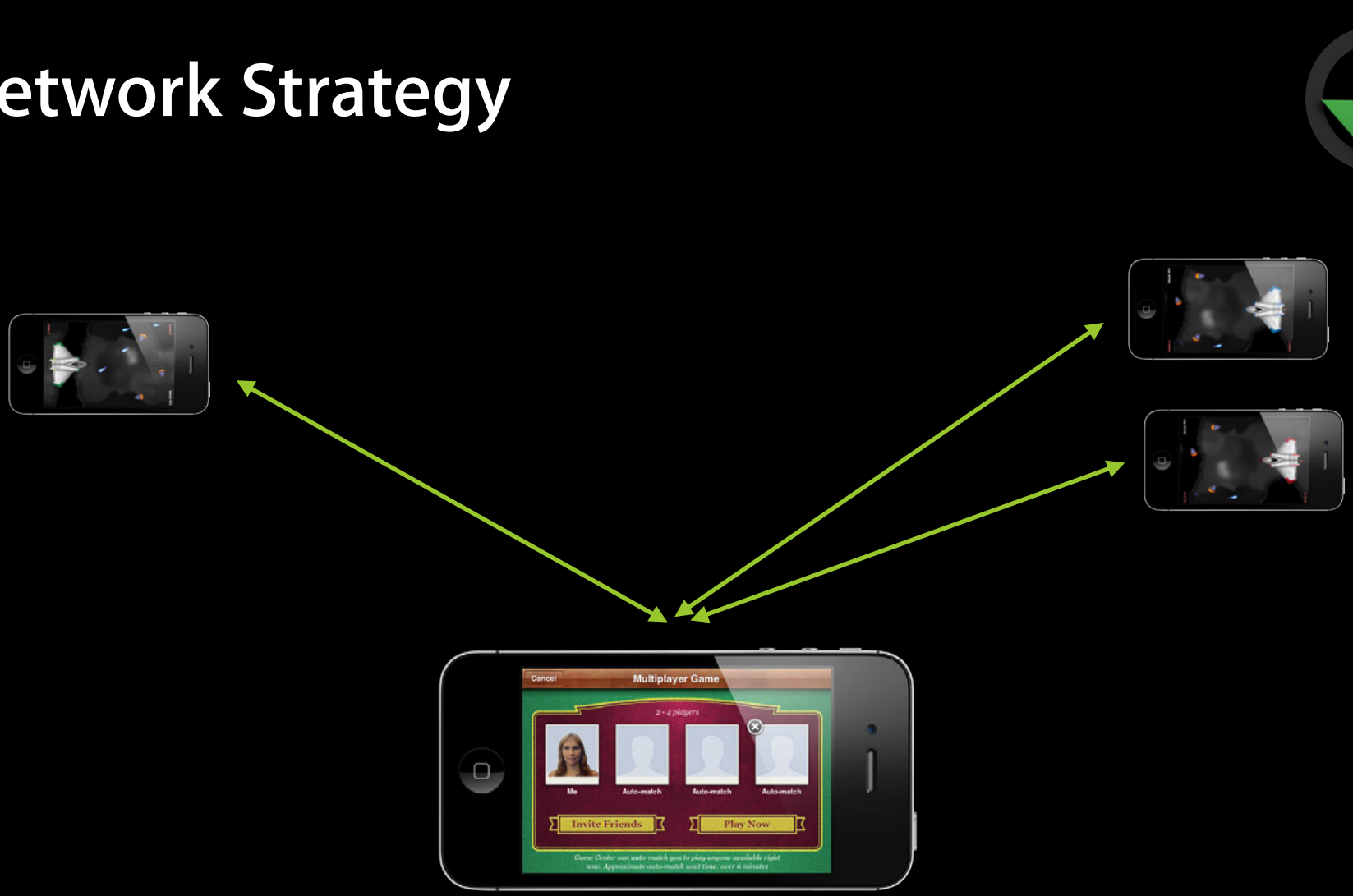
Full mesh



Network Strategy

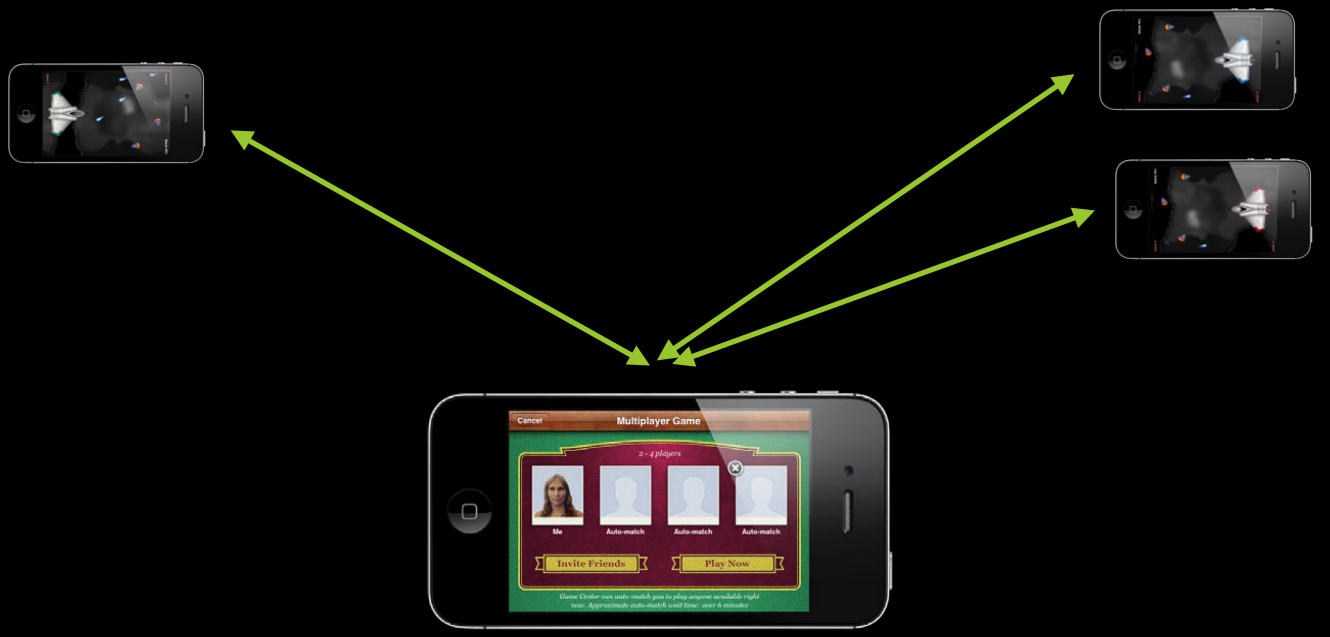


Network Strategy



Host Election API

Pick your host




```
[self.match  chooseBestHostPlayerWithCompletionHandler: ^(NSString *playerID) {
    self.hostingPlayer = playerID;
}];
```

Peer-to-Peer Network

Sending data

- Data size and frequency
- Choice of communication styles
  - Reliable vs. unreliable
- All players or specific

```
// unreliable send to all players
if (![self.match sendDataToAllPlayers:data
                         withDataMode:GKMatchSendDataUnreliable
                                error:&error]) {
    // Handle error
}
```

Peer-to-Peer Network

Send to specific player

```
// choose players
NSArray *playerIDs = [NSArray arrayWithObject:destPlayerID];

// unreliable send to specific players
if (![self.match sendData:data
                toPlayers:playerIDs
             withDataMode:GKMatchSendDataReliable
                    error:&error]) {
    // Handle error
}
```

Peer-to-Peer Network

Receiving data

```
// delegate callback to receive data
- (void)match:(GKMatch *)match didReceiveData:(NSData *)data
                                   fromPlayer:(NSString *)playerID
{
     // Parse data
}
```

Peer-to-Peer Networking

State changes

- Waiting for players to connect
  - Check expectedPlayers

```
- (void)match:(GKMatch *)match player:(NSString *)playerID
                        didChangeState:(GKPlayerConnectionState)state  {
  if (state == GKPlayerStateConnected)
      // Show that the player has connected
  else if (state == GKPlayerStateDisconnected)
      // Handle player disconnection
  if (!self.gameStarted && match.expectedPlayers == 0) {
    // Begin game once all players are connected
  }
}
```

Enable Reconnect for 1–1 Games

Works only on invite-based games

- Implement shouldReinvitePlayer on your GKMatchDelegate

```
- (BOOL)match:(GKMatch *)match shouldReinvitePlayer:(GKPlayer *)player
{
   return TRUE;
}
```

Add Player to Existing Game

Come join the fun

- Easy-to-use method on GKMatchMakerViewController

```
// modify existing match request
GKMatchRequest *matchRequest = self.matchRequest;
matchRequest.minPlayers = 2;
matchRequest.maxPlayers = 4;

// create view controller
GKMatchmakerViewController  *controller = [[GKMatchmakerViewController
alloc] initWithMatchRequest: matchRequest];
controller.delegate = self;

// add players to match
[controller addPlayersToMatch:self.currentMatch];
```

Add Player to Existing Game

Come join the fun

- Easy-to-use method on GKMatchMakerViewController

```
// modify existing match request
GKMatchRequest *matchRequest = self.matchRequest;
matchRequest.minPlayers = 2;
matchRequest.maxPlayers = 4;

// create view controller
GKMatchmakerViewController  *controller = [[GKMatchmakerViewController
alloc] initWithMatchRequest: matchRequest];
controller.delegate = self;

// add players to match
[controller addPlayersToMatch:self.currentMatch];
```

Add Player to Existing Game

Come join the fun

- Easy-to-use method on GKMatchMakerViewController

```
// modify existing match request
GKMatchRequest *matchRequest = self.matchRequest;
matchRequest.minPlayers = 2;
matchRequest.maxPlayers = 4;

// create view controller
GKMatchmakerViewController  *controller = [[GKMatchmakerViewController
alloc] initWithMatchRequest: matchRequest];
controller.delegate = self;

// add players to match
[controller addPlayersToMatch:self.currentMatch];
```

Add Player to Existing Game

Come join the fun

- Easy-to-use method on GKMatchMakerViewController

```
// modify existing match request
GKMatchRequest *matchRequest = self.matchRequest;
matchRequest.minPlayers = 2;
matchRequest.maxPlayers = 4;

// create view controller
GKMatchmakerViewController  *controller = [[GKMatchmakerViewController
alloc] initWithMatchRequest: matchRequest];
controller.delegate = self;

// add players to match
[controller addPlayersToMatch:self.currentMatch];
```

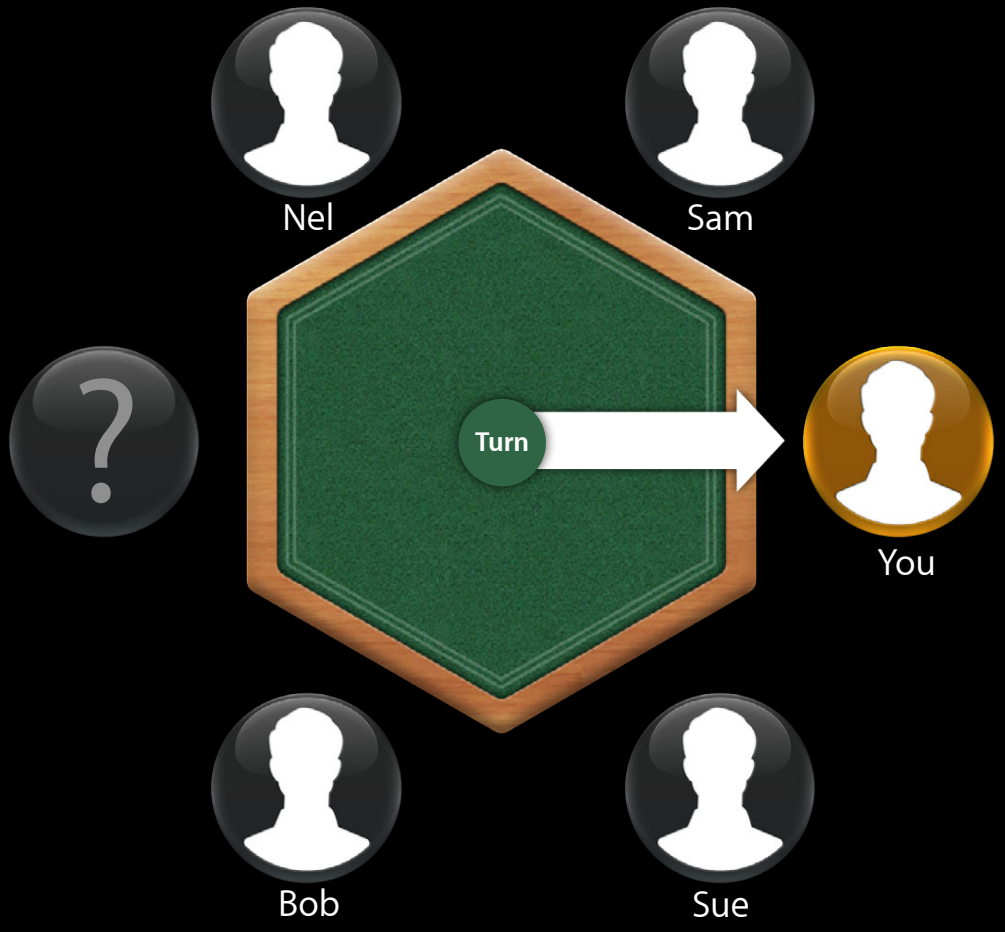
Peer-to-Peer Multiplayer

Summary

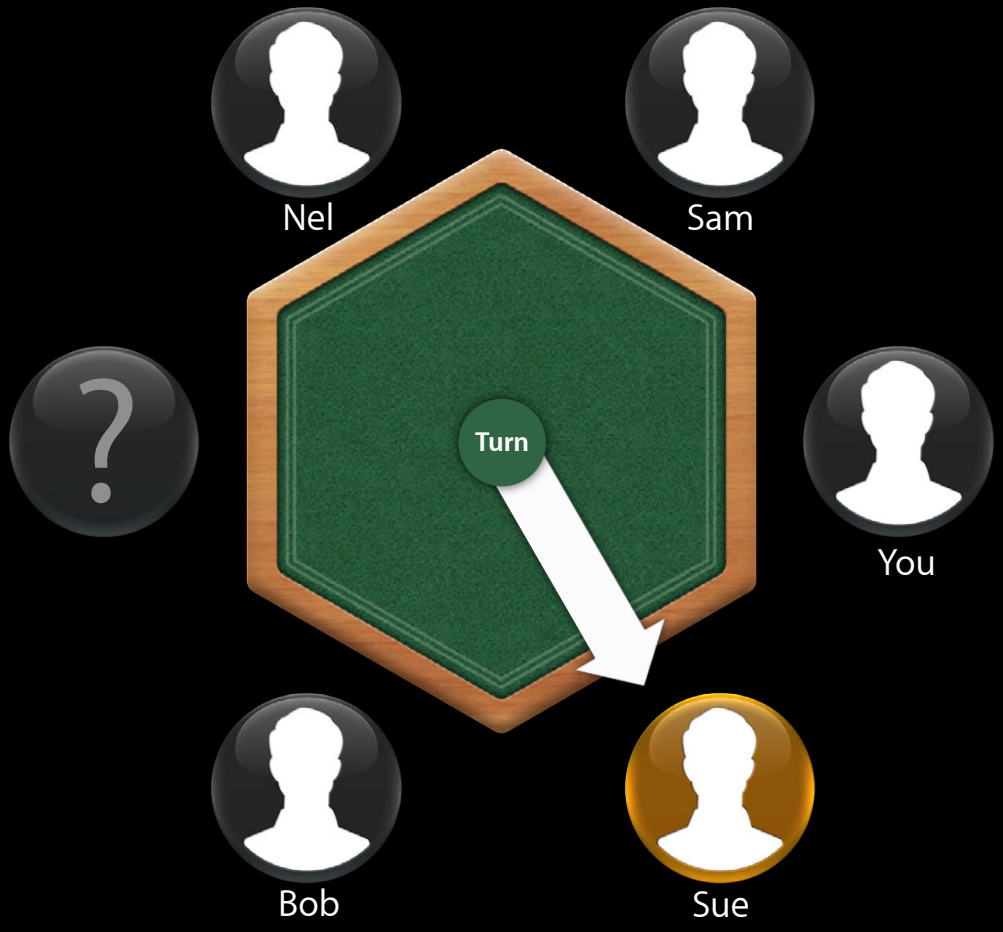
- Matchmaking UI
- Handling invites
- Programatic matchmaking
- Peer-to-peer communications

Turn-Based Multiplayer

Turn-Based Multiplayer



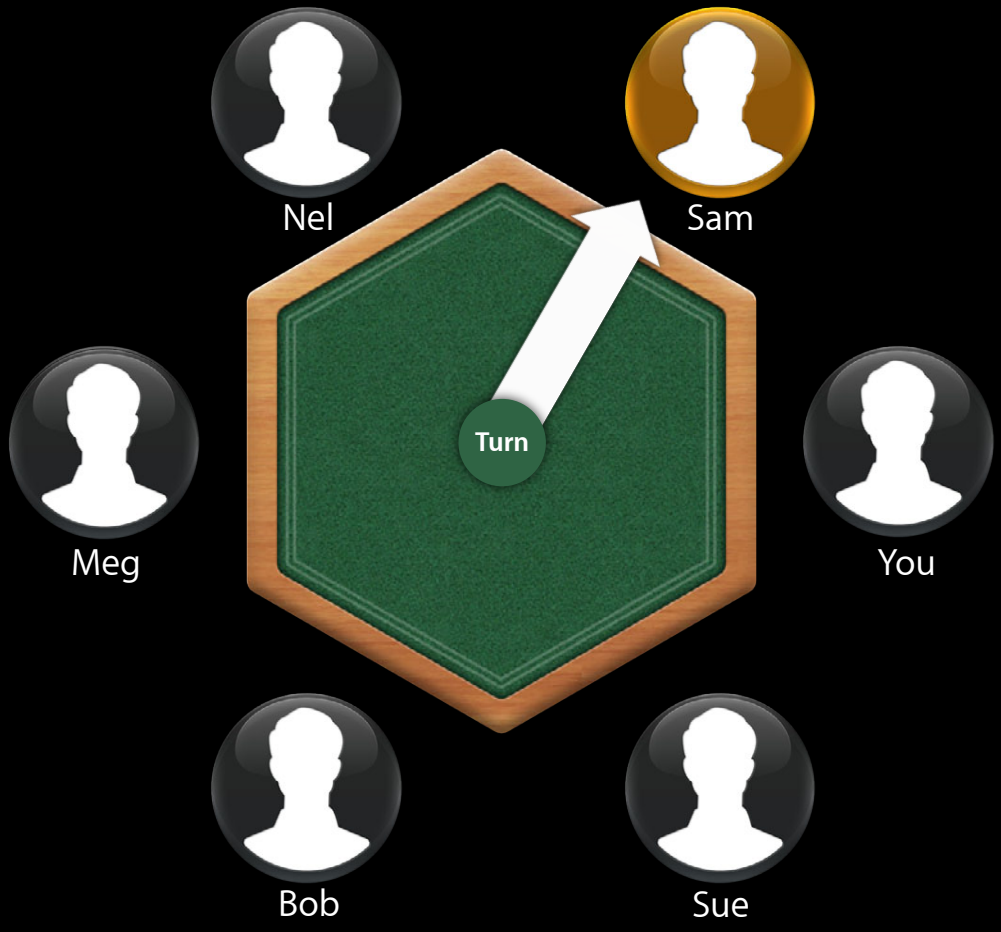
Turn-Based Multiplayer



Turn-Based Multiplayer

Turn-Based Multiplayer

Turn-Based Multiplayer

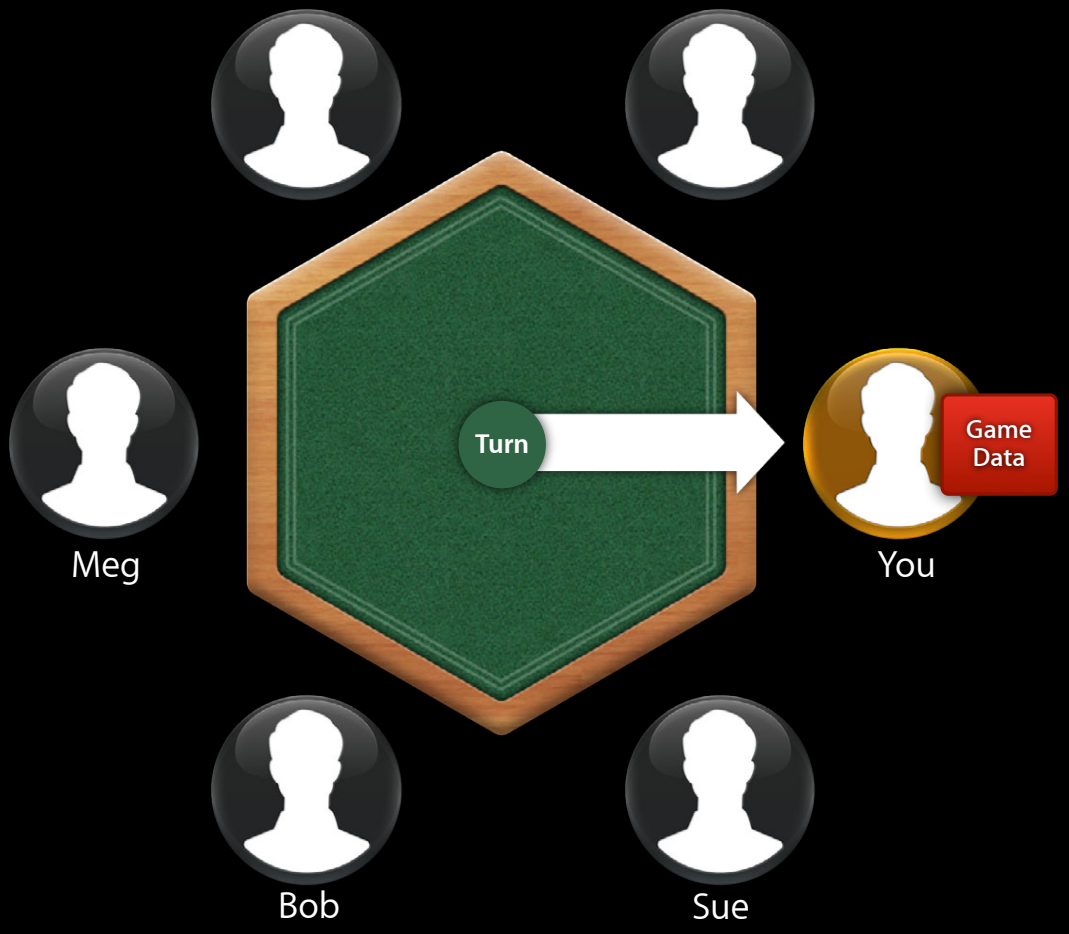


Turn-Based Multiplayer

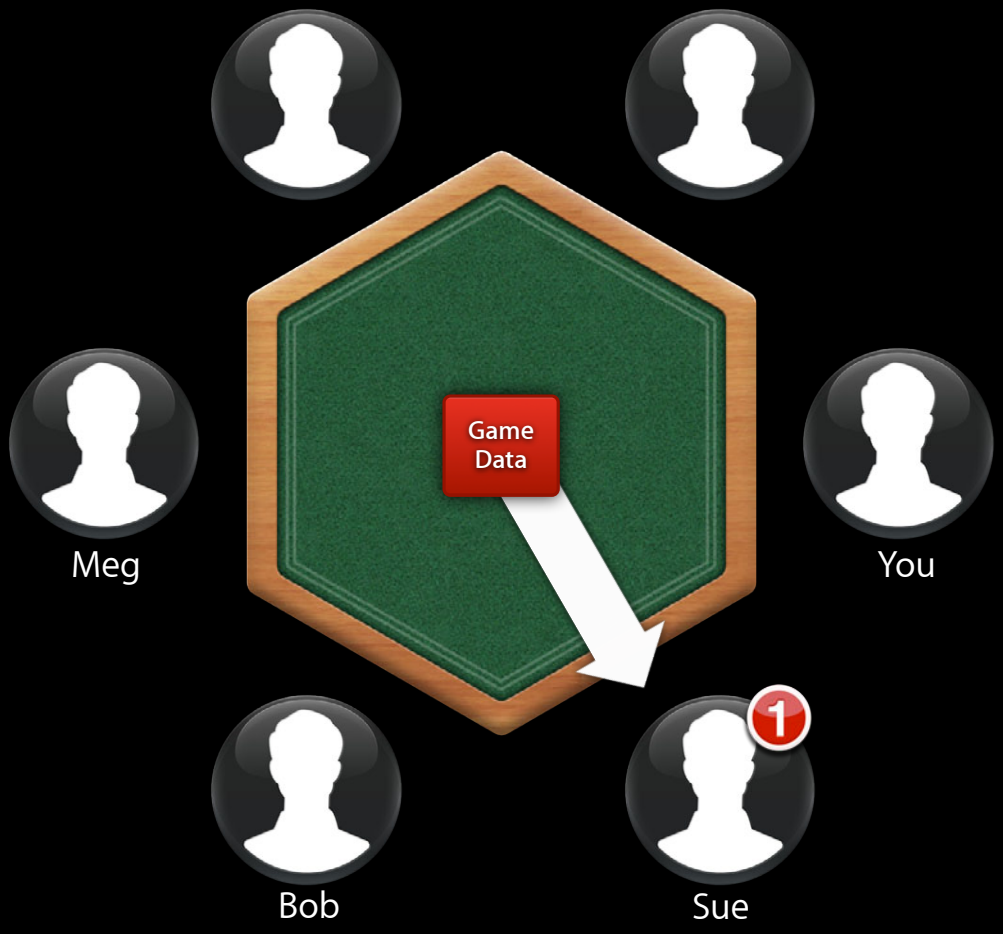
Turn-Based Multiplayer

Turn Flow Details

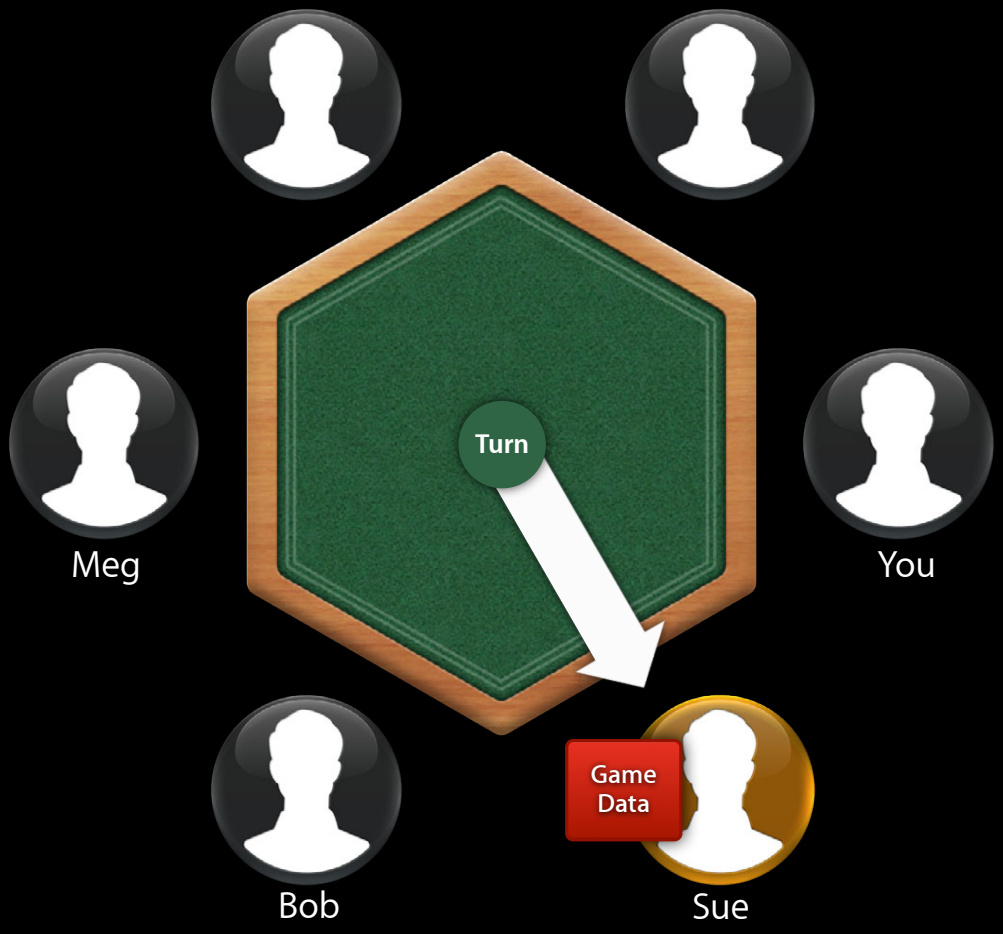
Turn Flow Details



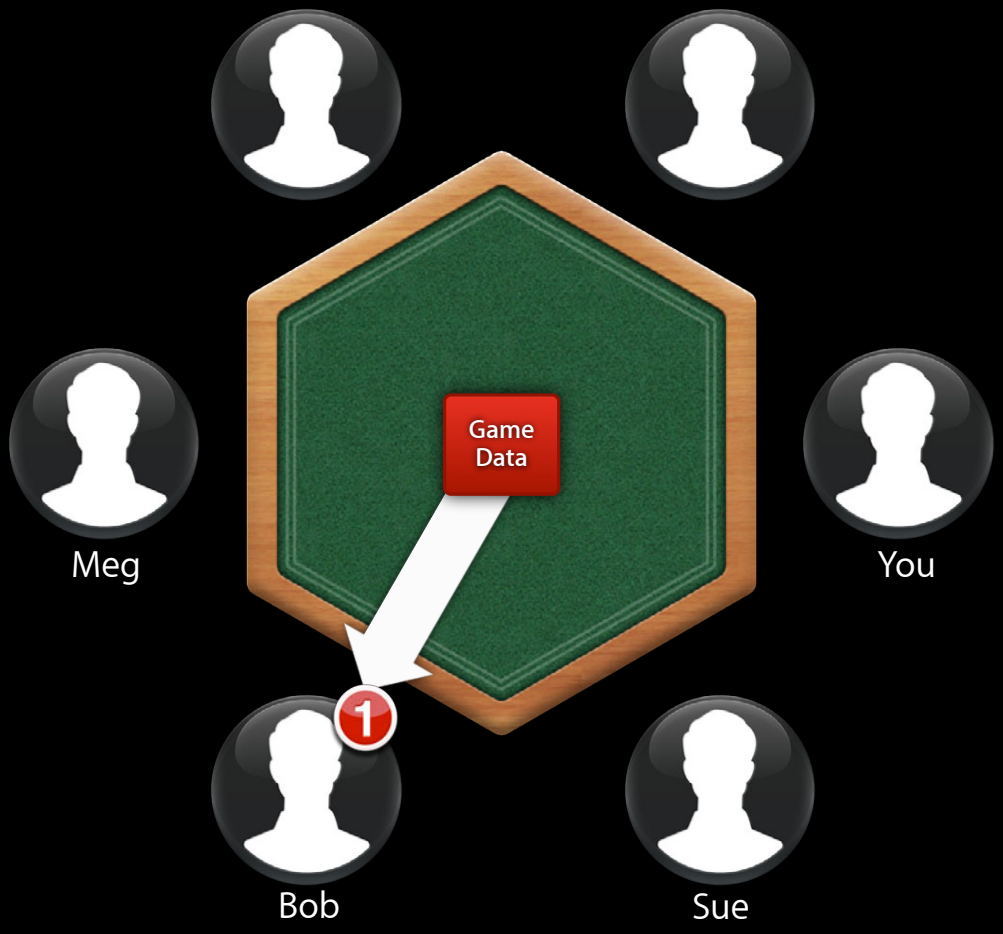
Turn Flow Details



Turn Flow Details

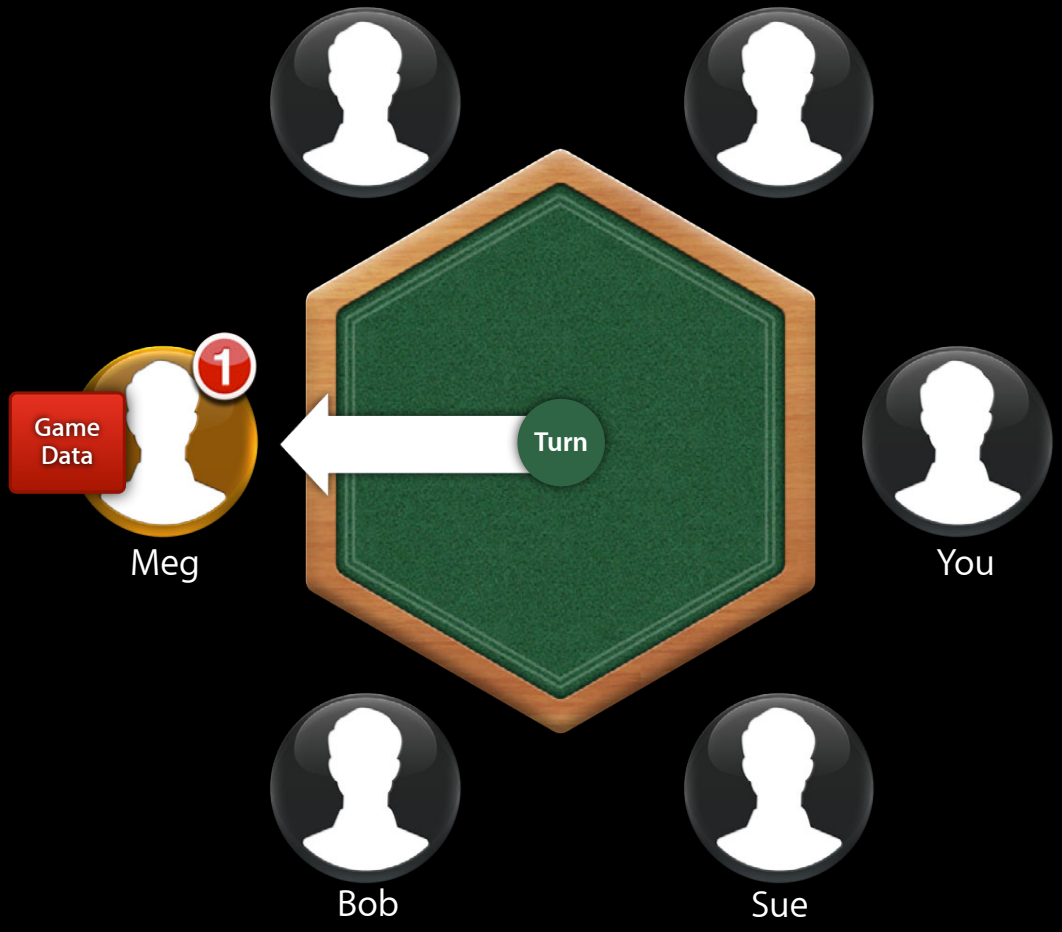


Turn Flow Details

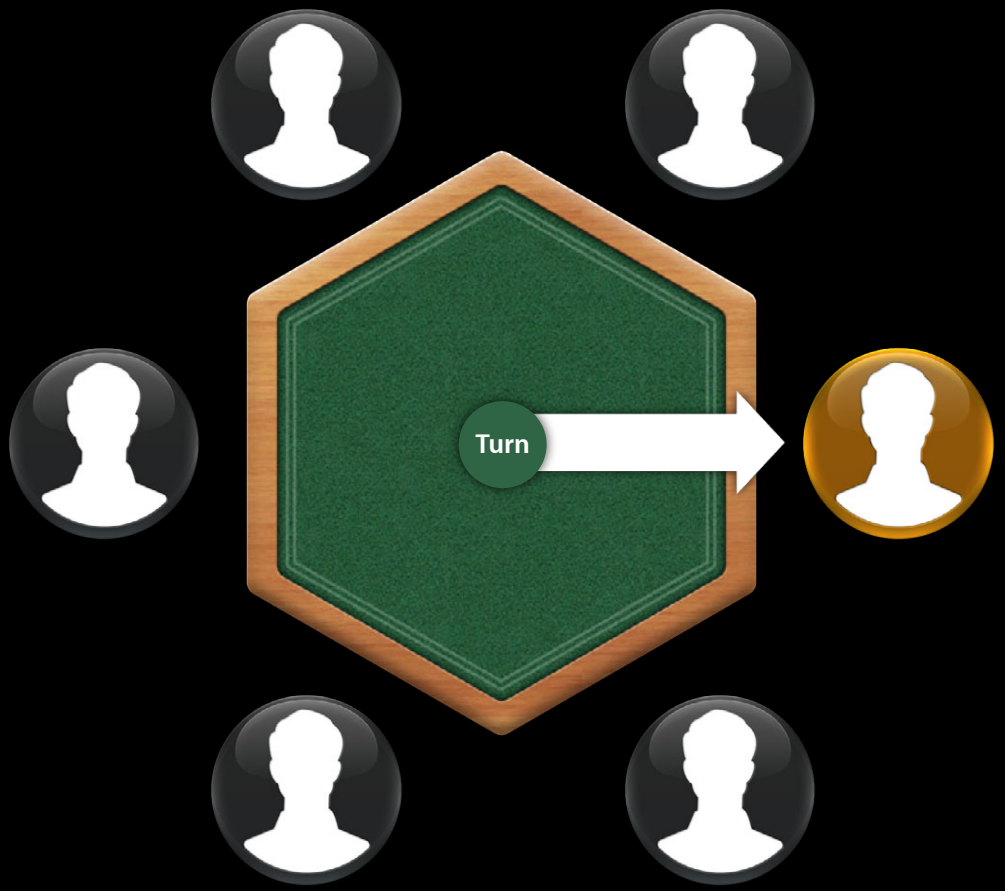


Turn Flow Details

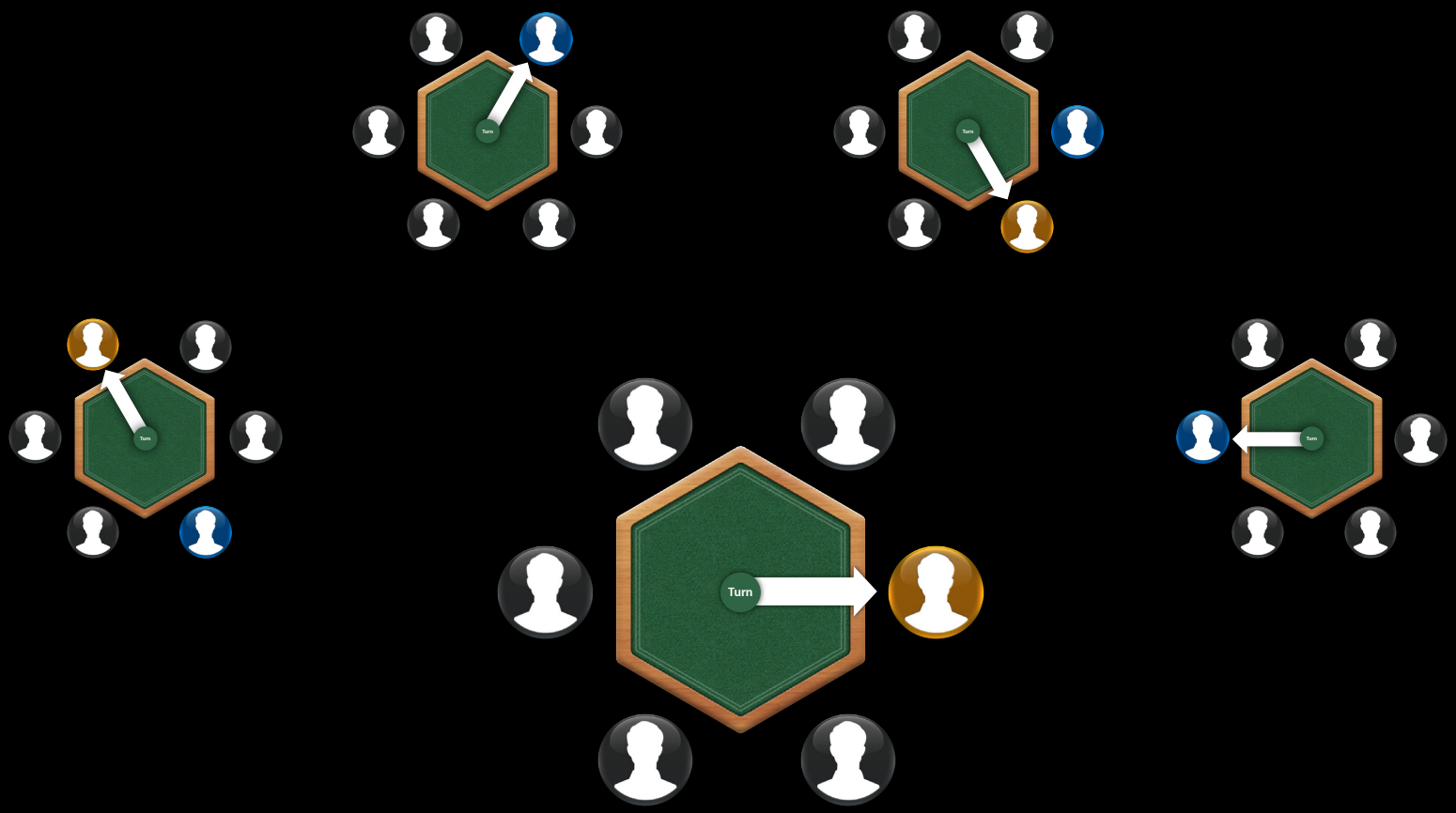
Turn Flow Details



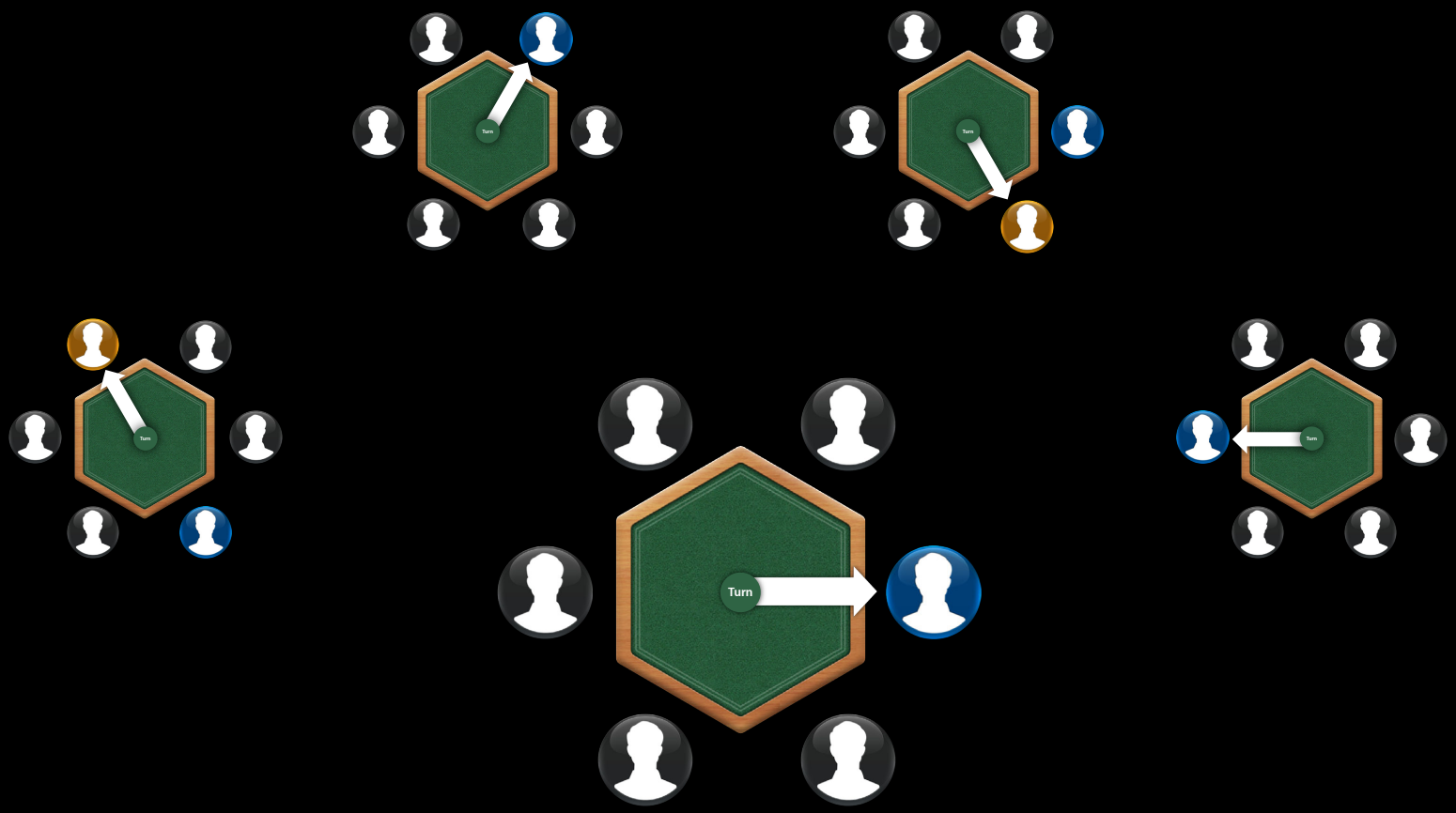
Multiple Matches



Multiple Matches



Multiple Matches



Turn-Based Games

Capabilities

Turn-Based Games

Capabilities

Simultaneous Matches	Up to 30

Turn-Based Games

Capabilities

Simultaneous Matches	Up to 30
Players per Match	Up to 16

Turn-Based Games

Capabilities

Simultaneous Matches	Up to 30
Players per Match	Up to 16
Gameplay	Asynchronous

Turn-Based Games

Capabilities

Simultaneous Matches	Up to 30
Players per Match	Up to 16
Gameplay	Asynchronous
Game Data	Up to 64K bytes

Turn-Based Games

Capabilities

Simultaneous Matches	Up to 30
Players per Match	Up to 16
Gameplay	Asynchronous
Game Data	Up to 64K bytes
Seats Filled	Invitations or Auto-Match

Turn-Based Games

Capabilities

Simultaneous Matches	Up to 30
Players per Match	Up to 16
Gameplay	Asynchronous
Game Data	Up to 64K bytes
Seats Filled	Invitations or Auto-Match
Turn Order	Developer defined

Turn-Based Games

Capabilities

Simultaneous Matches	Up to 30
Players per Match	Up to 16
Gameplay	Asynchronous
Game Data	Up to 64K bytes
Seats Filled	Invitations or Auto-Match
Turn Order	Developer defined
Missed Turns	Fallback list

Turn-Based Games

Capabilities

Simultaneous Matches	Up to 30
Players per Match	Up to 16
Gameplay	Asynchronous
Game Data	Up to 64K bytes
Seats Filled	Invitations or Auto-Match
Turn Order	Developer defined
Missed Turns	Fallback list
Turn Time Out	Default is 2 Weeks

Expectations of Turn-Based Games

Expectations of Turn-Based Games

- Multiple simultaneous matches
  - Each with its own state, players, outcome

Expectations of Turn-Based Games

- Multiple simultaneous matches
  - Each with its own state, players, outcome
- One player at a time
  - Other players just observe until it is their turn

Expectations of Turn-Based Games

- Multiple simultaneous matches
  - Each with its own state, players, outcome
- One player at a time
  - Other players just observe until it is their turn
- Not always running
  - Choose match, takes turn, and exits

Expectations of Turn-Based Games

- Multiple simultaneous matches
  - Each with its own state, players, outcome
- One player at a time
  - Other players just observe until it is their turn
- Not always running
  - Choose match, takes turn, and exits
- Anywhere in app
  - Making a move in match three when turn notification received for match six

Classes for Turn-Based Gaming

GKTurnBasedMatch

Main entry point to Turn-Based API

- Instance of the game
  - List of participants
  - The current game state
  - The player whose turn it is now

GKTurnBasedParticipant

Details about each participant in the match

- Player ID
  - May be a player or an open position
- Status
  - Invited, matching, active, done
- Outcome
  - Filled when game over, or player quits
  - Won, lost, tied, 1st place, 2nd place, etc.

GKTurnBasedEventHandler

Entry point for events

- Singleton for external events
- Called when:
  - The player has received an invite to join a new match
  - An invite has been initiated from the Game Center app
  - Someone has taken their turn in a match
  - The match has ended

GKTurnBasedMatchmakerViewController

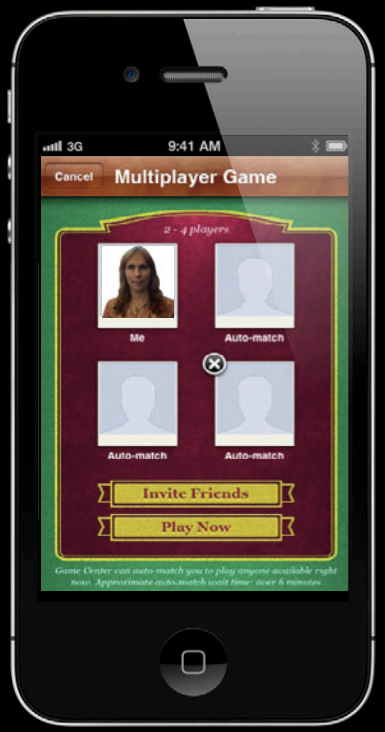
Focal point for player actions in game

- Manage matches
  - Choose a match to play
  - Quit from a match
- Create new matches
  - Invites
  - Auto-match

GKTurnBasedMatchmakerViewController

Focal point for player actions in game

- Manage matches
  - Choose a match to play
  - Quit from a match
- Create new matches
  - Invites
  - Auto-match



Creating the View Controller

```
// Set up match request
GKMatchRequest *request = [[GKMatchRequest alloc] init];
request.minPlayers = 2;
request.maxPlayers = 4;

// Create view controller & pass match request info
GKTurnBasedMatchmakerViewController *viewController =
    [[GKTurnBasedMatchmakerViewController alloc] initWithMatchRequest:request];

// Set options & delegate
viewController.showExistingMatches = YES;
viewController.turnBasedMatchmakerDelegate = self;

// Present to user
[rootViewController presentViewController:viewController
    animated:YES completion:nil];
```

Creating the View Controller

```
// Set up match request
GKMatchRequest *request = [[GKMatchRequest alloc] init];
request.minPlayers = 2;
request.maxPlayers = 4;

// Create view controller & pass match request info
GKTurnBasedMatchmakerViewController *viewController =
    [[GKTurnBasedMatchmakerViewController alloc] initWithMatchRequest:request];

// Set options & delegate
viewController.showExistingMatches = YES;
viewController.turnBasedMatchmakerDelegate = self;

// Present to user
[rootViewController presentViewController:viewController
    animated:YES completion:nil];
```

Creating the View Controller

```
// Set up match request
GKMatchRequest *request = [[GKMatchRequest alloc] init];
request.minPlayers = 2;
request.maxPlayers = 4;

// Create view controller & pass match request info
GKTurnBasedMatchmakerViewController *viewController =
    [[GKTurnBasedMatchmakerViewController alloc] initWithMatchRequest:request];

// Set options & delegate
viewController.showExistingMatches = YES;
viewController.turnBasedMatchmakerDelegate = self;

// Present to user
[rootViewController presentViewController:viewController
    animated:YES completion:nil];
```

Creating the View Controller

```
// Set up match request
GKMatchRequest *request = [[GKMatchRequest alloc] init];
request.minPlayers = 2;
request.maxPlayers = 4;

// Create view controller & pass match request info
GKTurnBasedMatchmakerViewController *viewController =
    [[GKTurnBasedMatchmakerViewController alloc] initWithMatchRequest:request];

// Set options & delegate
viewController.showExistingMatches = YES;
viewController.turnBasedMatchmakerDelegate = self;

// Present to user
[rootViewController presentViewController:viewController
    animated:YES completion:nil];
```

Creating the View Controller

```
// Set up match request
GKMatchRequest *request = [[GKMatchRequest alloc] init];
request.minPlayers = 2;
request.maxPlayers = 4;

// Create view controller & pass match request info
GKTurnBasedMatchmakerViewController *viewController =
    [[GKTurnBasedMatchmakerViewController alloc] initWithMatchRequest:request];

// Set options & delegate
viewController.showExistingMatches = YES;
viewController.turnBasedMatchmakerDelegate = self;

// Present to user
[rootViewController presentViewController:viewController
    animated:YES completion:nil];
```

View Controller Delegate

Delegate actions

View Controller Delegate

Delegate actions

Create New Match `– didFindMatch:`

View Controller Delegate

Delegate actions

Create New Match	`– didFindMatch:`
Select a Match	`– didFindMatch:`

View Controller Delegate

Delegate actions

Create New Match	`– didFindMatch:`
Select a Match	`– didFindMatch:`
Accept Invite	`– didFindMatch:`

View Controller Delegate

Delegate actions

Create New Match	`– didFindMatch:`
Select a Match	`– didFindMatch:`
Accept Invite	`– didFindMatch:`
User Hit “Cancel”	`– turnBasedMatchmakerViewControllerWasCancelled`

View Controller Delegate

Delegate actions

Create New Match	– didFindMatch:
Select a Match	– didFindMatch:
Accept Invite	– didFindMatch:
User Hit “Cancel”	– turnBasedMatchmakerViewControllerWasCancelled
Match Failed	– didFailWithError:

View Controller Delegate

Delegate actions

Create New Match	– didFindMatch:
Select a Match	– didFindMatch:
Accept Invite	– didFindMatch:
User Hit “Cancel”	– turnBasedMatchmakerViewControllerWasCancelled
Match Failed	– didFailWithError:
Remove a Match	– playerQuitForMatch:

View Controller Delegate

User selects a match

```
-(void)turnBasedMatchmakerViewController:(GKTurnBasedMatchmakerViewController *)tbmvc
                            didFindMatch:(GKTurnBasedMatch *)match
{
   // Dismiss view controller
   [rootViewController dismissViewControllerAnimated:YES completion:nil];

   // Download latest game state
   [match loadMatchDataWithCompletionHandler:^(NSData *data, NSError *err) {
      [self unpackMatchData:data forMatch:match];

      // Show match to user
      [self displayGameStateForMatch:match];

      // If my turn, allow player to take actions in-game
      if ([match.currentParticipant.playerID isEqualToString:localPlayer.playerID])
         [self takeMyTurnForMatch:match];
   }];
}
```

View Controller Delegate

User selects a match

```
-(void)turnBasedMatchmakerViewController:(GKTurnBasedMatchmakerViewController *)tbmvc
                                didFindMatch:(GKTurnBasedMatch *)match
{
   // Dismiss view controller
   [rootViewController dismissViewControllerAnimated:YES completion:nil];

   // Download latest game state
   [match loadMatchDataWithCompletionHandler:^(NSData *data, NSError *err) {
      [self unpackMatchData:data forMatch:match];

      // Show match to user
      [self displayGameStateForMatch:match];

      // If my turn, allow player to take actions in-game
      if ([match.currentParticipant.playerID isEqualToString:localPlayer.playerID])
         [self takeMyTurnForMatch:match];
   }];
}
```

View Controller Delegate

User selects a match

```
-(void)turnBasedMatchmakerViewController:(GKTurnBasedMatchmakerViewController *)tbmvc
                             didFindMatch:(GKTurnBasedMatch *)match
{
    // Dismiss view controller
    [rootViewController dismissViewControllerAnimated:YES completion:nil];

    // Download latest game state
    [match loadMatchDataWithCompletionHandler:^(NSData *data, NSError *err) {
        [self unpackMatchData:data forMatch:match];

        // Show match to user
        [self displayGameStateForMatch:match];

        // If my turn, allow player to take actions in-game
        if ([match.currentParticipant.playerID isEqualToString:localPlayer.playerID])
            [self takeMyTurnForMatch:match];
    }];
}
```

View Controller Delegate

User selects a match

```
-(void)turnBasedMatchmakerViewController:(GKTurnBasedMatchmakerViewController *)tbmvc
                             didFindMatch:(GKTurnBasedMatch *)match
{
   // Dismiss view controller
   [rootViewController dismissViewControllerAnimated:YES completion:nil];

   // Download latest game state
   [match loadMatchDataWithCompletionHandler:^(NSData *data, NSError *err) {
      [self unpackMatchData:data forMatch:match];

      // Show match to user
      [self displayGameStateForMatch:match];

      // If my turn, allow player to take actions in-game
      if ([match.currentParticipant.playerID isEqualToString:localPlayer.playerID])
         [self takeMyTurnForMatch:match];
   }];
}
```

View Controller Delegate

User selects a match

```
-(void)turnBasedMatchmakerViewController:(GKTurnBasedMatchmakerViewController *)tbmvc
                             didFindMatch:(GKTurnBasedMatch *)match
{
   // Dismiss view controller
   [rootViewController dismissViewControllerAnimated:YES completion:nil];

   // Download latest game state
   [match loadMatchDataWithCompletionHandler:^(NSData *data, NSError *err) {
      [self unpackMatchData:data forMatch:match];

      // Show match to user
      [self displayGameStateForMatch:match];

      // If my turn, allow player to take actions in-game
      if ([match.currentParticipant.playerID isEqualToString:localPlayer.playerID])
         [self takeMyTurnForMatch:match];
   }];
}
```

View Controller Delegate

User selects a match

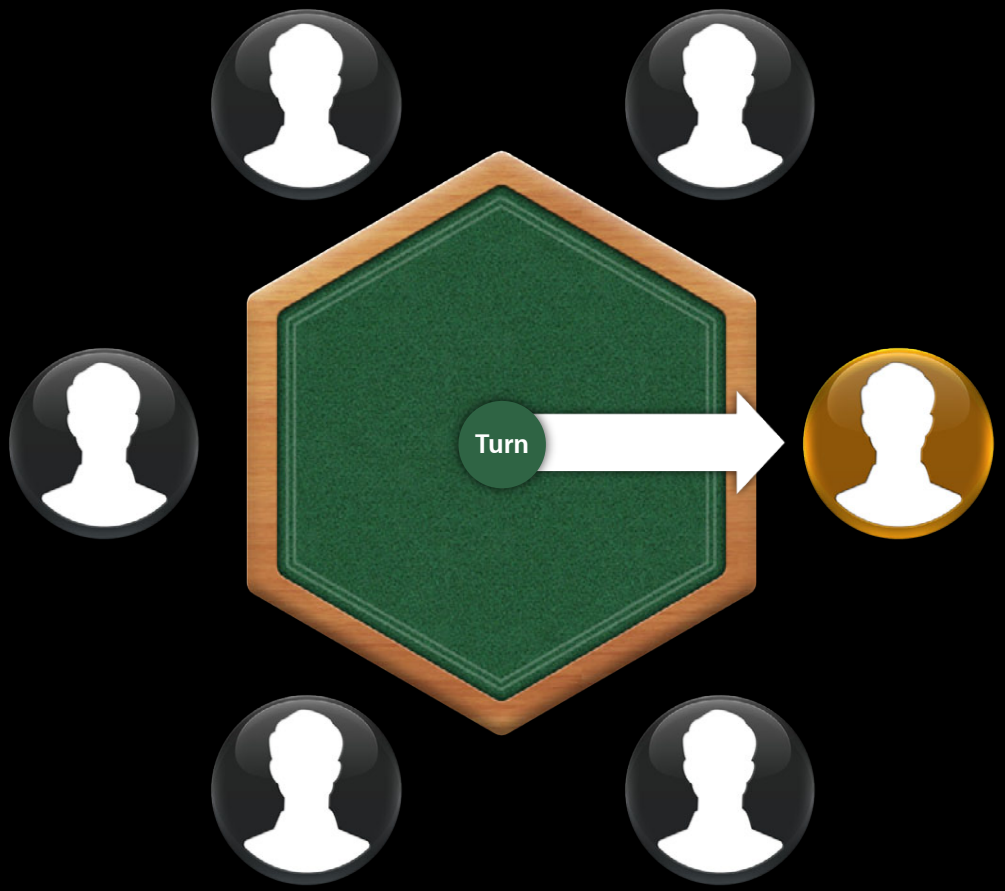
```
-(void)turnBasedMatchmakerViewController:(GKTurnBasedMatchmakerViewController *)tbmvc
                            didFindMatch:(GKTurnBasedMatch *)match
{
   // Dismiss view controller
   [rootViewController dismissViewControllerAnimated:YES completion:nil];

   // Download latest game state
   [match loadMatchDataWithCompletionHandler:^(NSData *data, NSError *err) {
      [self unpackMatchData:data forMatch:match];

      // Show match to user
      [self displayGameStateForMatch:match];

      // If my turn, allow player to take actions in-game
      if ([match.currentParticipant.playerID isEqualToString:localPlayer.playerID])
         [self takeMyTurnForMatch:match];
   }];
}
```

Match Data

Current state of the match

- NSData
  - Contents are developer-defined
- Stored online
  - Must have turn to update
  - Others can read
- Limited size: 64K bytes
  - Pack data wisely
  - Or point to server stored data

Taking a Turn



Taking a Turn

Taking My Turn

Overview

- Make move
  - Based on rules of your game
  - Make a move, resign, pass, etc.
- Choose the next player(s)
- Submit turn

Taking My Turn

Submitting a turn

```
// Update the state of the match
newMatchData = [self updateMatchData:match.matchData];

// Choose next participants
NSArray *nextParticipants = [self chooseNextParticipants:match];

// Set a message to be shown to other participant(s)
match.message = @"Made Kessel Run in 12 parsecs!";

// Send new game state to Game Center & pass turn to next participant
[match endTurnWithNextParticipants:nextParticipants
                       turnTimeout:GKTurnBasedTimeoutDefault
                         matchData:newMatchData
                 completionHandler:^(NSError *error) { ... } ];
```

Taking My Turn

Submitting a turn

```
// Update the state of the match
newMatchData = [self updateMatchData:match.matchData];

// Choose next participants
NSArray *nextParticipants = [self chooseNextParticipants:match];

// Set a message to be shown to other participant(s)
match.message = @"Made Kessel Run in 12 parsecs!";

// Send new game state to Game Center & pass turn to next participant
[match endTurnWithNextParticipants:nextParticipants
                       turnTimeout:GKTurnBasedTimeoutDefault
                         matchData:newMatchData
                 completionHandler:^(NSError *error) { ... } ];
```

Taking My Turn

Submitting a turn

```
// Update the state of the match
newMatchData = [self updateMatchData:match.matchData];

// Choose next participants
NSArray *nextParticipants = [self chooseNextParticipants:match];

// Set a message to be shown to other participant(s)
match.message = @"Made Kessel Run in 12 parsecs!";

// Send new game state to Game Center & pass turn to next participant
[match endTurnWithNextParticipants:nextParticipants
                       turnTimeout:GKTurnBasedTimeoutDefault
                         matchData:newMatchData
                 completionHandler:^(NSError *error) { ... } ];
```

Taking My Turn

Submitting a turn

```
// Update the state of the match
newMatchData = [self updateMatchData:match.matchData];

// Choose next participants
NSArray *nextParticipants = [self chooseNextParticipants:match];

// Set a message to be shown to other participant(s)
match.message = @"Made Kessel Run in 12 parsecs!";

// Send new game state to Game Center & pass turn to next participant
[match endTurnWithNextParticipants:nextParticipants
                       turnTimeout:GKTurnBasedTimeoutDefault
                         matchData:newMatchData
                 completionHandler:^(NSError *error) { ... } ];
```

Taking My Turn

Submitting a turn

```
// Update the state of the match
newMatchData = [self updateMatchData:match.matchData];

// Choose next participants
NSArray *nextParticipants = [self chooseNextParticipants:match];

// Set a message to be shown to other participant(s)
match.message = @"Made Kessel Run in 12 parsecs!";

// Send new game state to Game Center & pass turn to next participant
[match endTurnWithNextParticipants:nextParticipants
                       turnTimeout:GKTurnBasedTimeoutDefault
                         matchData:newMatchData
                 completionHandler:^(NSError *error) { ... } ];
```

Taking My Turn

Choosing next participants

- Based on game rules
- Choose active players
- Guard against missed turns
  - Provide a list of multiple next participants
  - Use time outs
  - Last participant on list does NOT time out
  - Include yourself last

Taking My Turn

Choosing the next participants

```
-(NSArray *)chooseNextParticipants:(GKTurnBasedMatch *)match
{
    NSMutableArray *nextParticipants = [NSMutableArray array];

    // Get my index in the player list
    NSUInteger i = [match.participants indexOfObject:match.currentParticipant];

    // Iterate through participants adding active players to array
    NSUInteger start = i;
    do {
        GKTurnBasedParticipant *participant =
            [match.participants objectAtIndex:((i + 1) % match.participants.count)];
        if (nextParticipant.status != GKTurnBasedParticipantStatusDone)
            [nextParticipants addObject: participant];

    } while ((i++ % match.participants.count) != start);

    if (nextParticipants.count > 0)
        return nextParticipants;
    else
        return nil;  // everyone else has quit, handle accordingly
}
```

Taking My Turn

Choosing the next participants

```
-(NSArray *)chooseNextParticipants:(GKTurnBasedMatch *)match
{
    NSMutableArray *nextParticipants = [NSMutableArray array];

    // Get my index in the player list
    NSUInteger i = [match.participants indexOfObject:match.currentParticipant];

    // Iterate through participants adding active players to array
    NSUInteger start = i;
    do {
        GKTurnBasedParticipant *participant =
            [match.participants objectAtIndex:((i + 1) % match.participants.count)];
        if (nextParticipant.status != GKTurnBasedParticipantStatusDone)
            [nextParticipants addObject: participant];

    } while ((i++ % match.participants.count) != start);

    if (nextParticipants.count > 0)
        return nextParticipants;
    else
        return nil;  // everyone else has quit, handle accordingly
}
```

Taking My Turn

Choosing the next participants

```
-(NSArray *)chooseNextParticipants:(GKTurnBasedMatch *)match
{
    NSMutableArray *nextParticipants = [NSMutableArray array];

    // Get my index in the player list
    NSUInteger i = [match.participants indexOfObject:match.currentParticipant];

    // Iterate through participants adding active players to array
    NSUInteger start = i;
    do {
        GKTurnBasedParticipant *participant =
            [match.participants objectAtIndex:((i + 1) % match.participants.count)];
        if (nextParticipant.status != GKTurnBasedParticipantStatusDone)
            [nextParticipants addObject: participant];

    } while ((i++ % match.participants.count) != start);

    if (nextParticipants.count > 0)
        return nextParticipants;
    else
        return nil;  // everyone else has quit, handle accordingly
}
```

Taking My Turn

Choosing the next participants

```
-(NSArray *)chooseNextParticipants:(GKTurnBasedMatch *)match
{
    NSMutableArray *nextParticipants = [NSMutableArray array];

    // Get my index in the player list
    NSUInteger i = [match.participants indexOfObject:match.currentParticipant];

    // Iterate through participants adding active players to array
    NSUInteger start = i;
    do {
        GKTurnBasedParticipant *participant =
            [match.participants objectAtIndex:((i + 1) % match.participants.count)];
        if (nextParticipant.status != GKTurnBasedParticipantStatusDone)
            [nextParticipants addObject: participant];

    } while ((i++ % match.participants.count) != start);

    if (nextParticipants.count > 0)
        return nextParticipants;
    else
        return nil;  // everyone else has quit, handle accordingly
}
```

Saving Match Data

Update turn progress


```
// turn lasts until wrong answer
while ([self askTriviaQuestion) {

    // update state
    self.numberCorrect += 1

    // notify others
    [self.match saveCurrentTurnWithMatchData:self.matchData
                           completionHandler:^(NSError *error) { }];
}
```

Saving Match Data

Update turn progress


```
// turn lasts until wrong answer
while ([self askTriviaQuestion) {

    // update state
    self.numberCorrect += 1

    // notify others
    [self.match saveCurrentTurnWithMatchData:self.matchData
                           completionHandler:^(NSError *error) { }];
}
```

Saving Match Data

Update turn progress


```
// turn lasts until wrong answer
while ([self askTriviaQuestion) {

    // update state
    self.numberCorrect += 1

    // notify others
    [self.match saveCurrentTurnWithMatchData:self.matchData
                           completionHandler:^(NSError *error) { }];
}
```

Saving Match Data

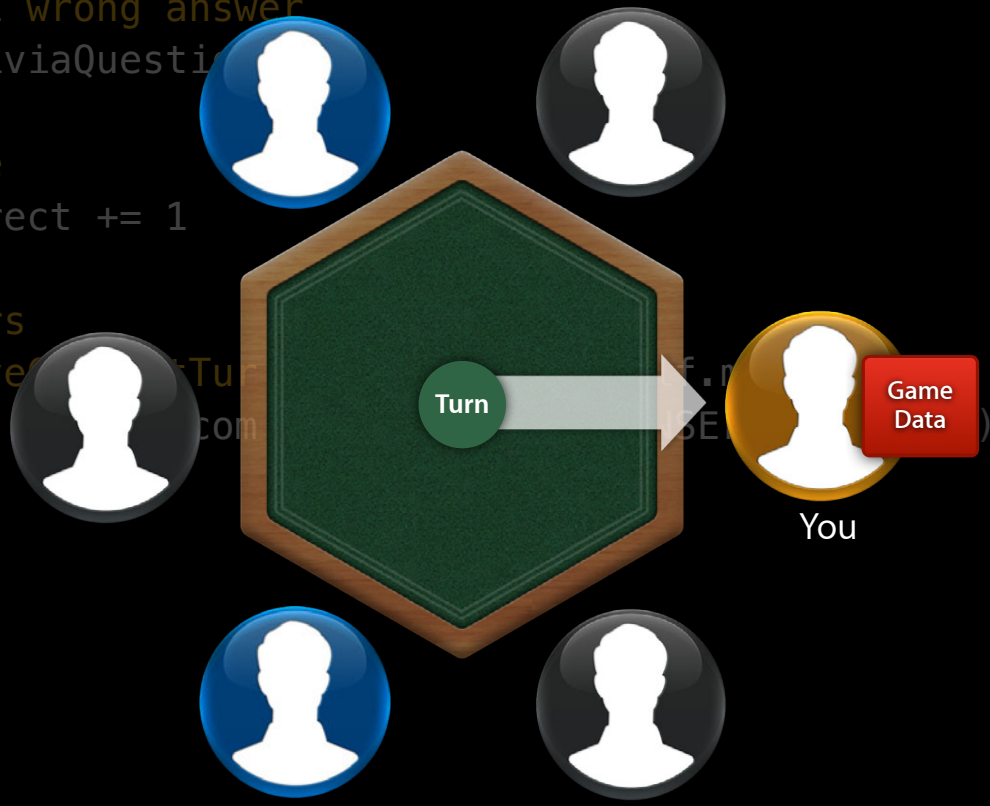
Update turn progress


```
// turn lasts until wrong answer
while ([self askTriviaQuestion) {

    // update state
    self.numberCorrect += 1

    // notify others
    [self.match saveCurrentTurnWithMatchData:self.matchData
                           completionHandler:^(NSError *error) { }];
}
```

Saving Match Data

Update turn progress

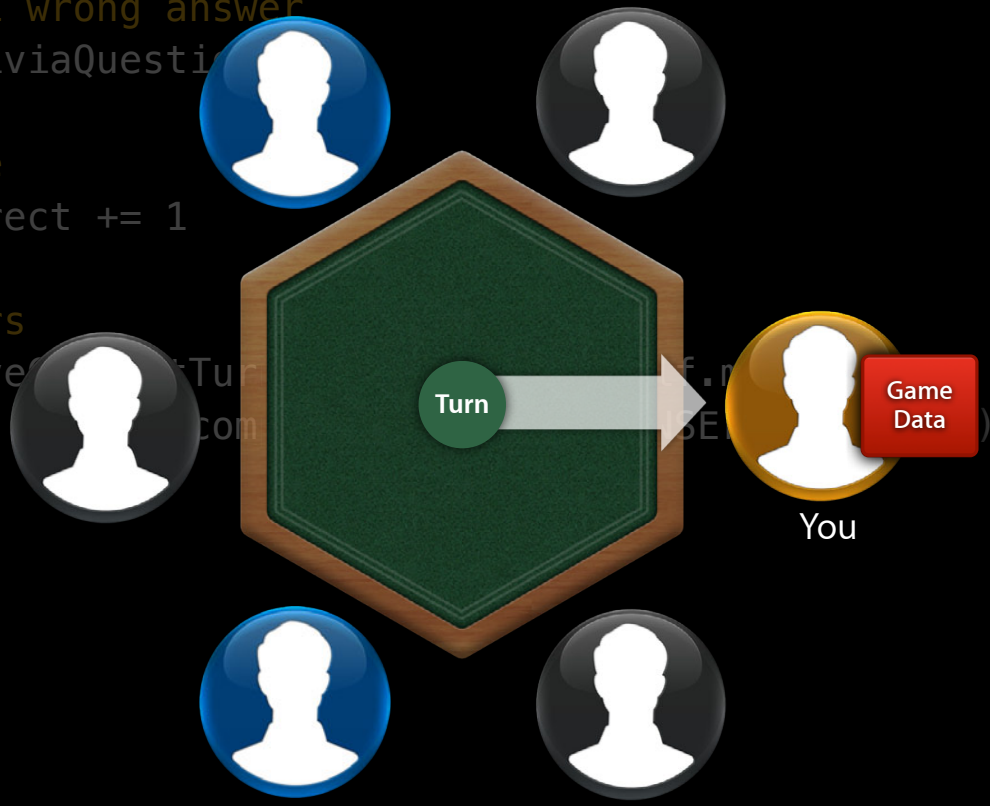

```
// turn lasts until wrong answer
while ([self askTriviaQuesti

    // update state
    self.numberCorrect += 1

    // notify others
    [self.match save
                    com                    ) { }];
}
```



Saving Match Data

Update turn progress

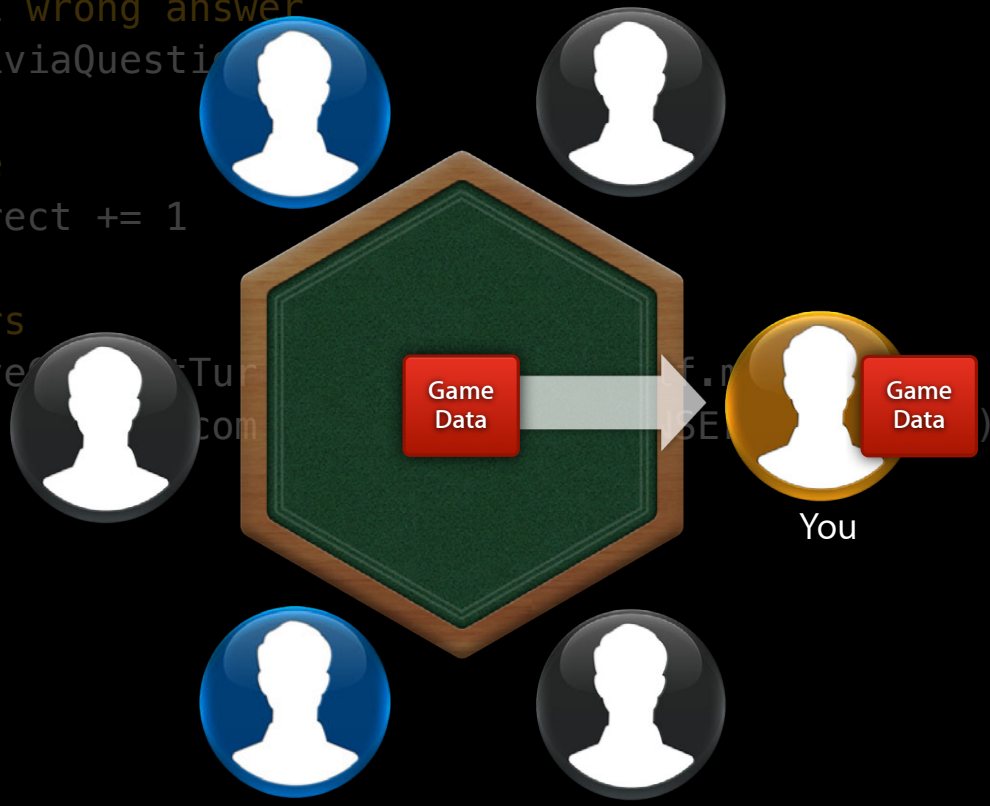

```
// turn lasts until wrong answer
while ([self askTriviaQuesti

    // update state
    self.numberCorrect += 1

    // notify others
    [self.match save
                                                    ) { }];
}
```



Saving Match Data

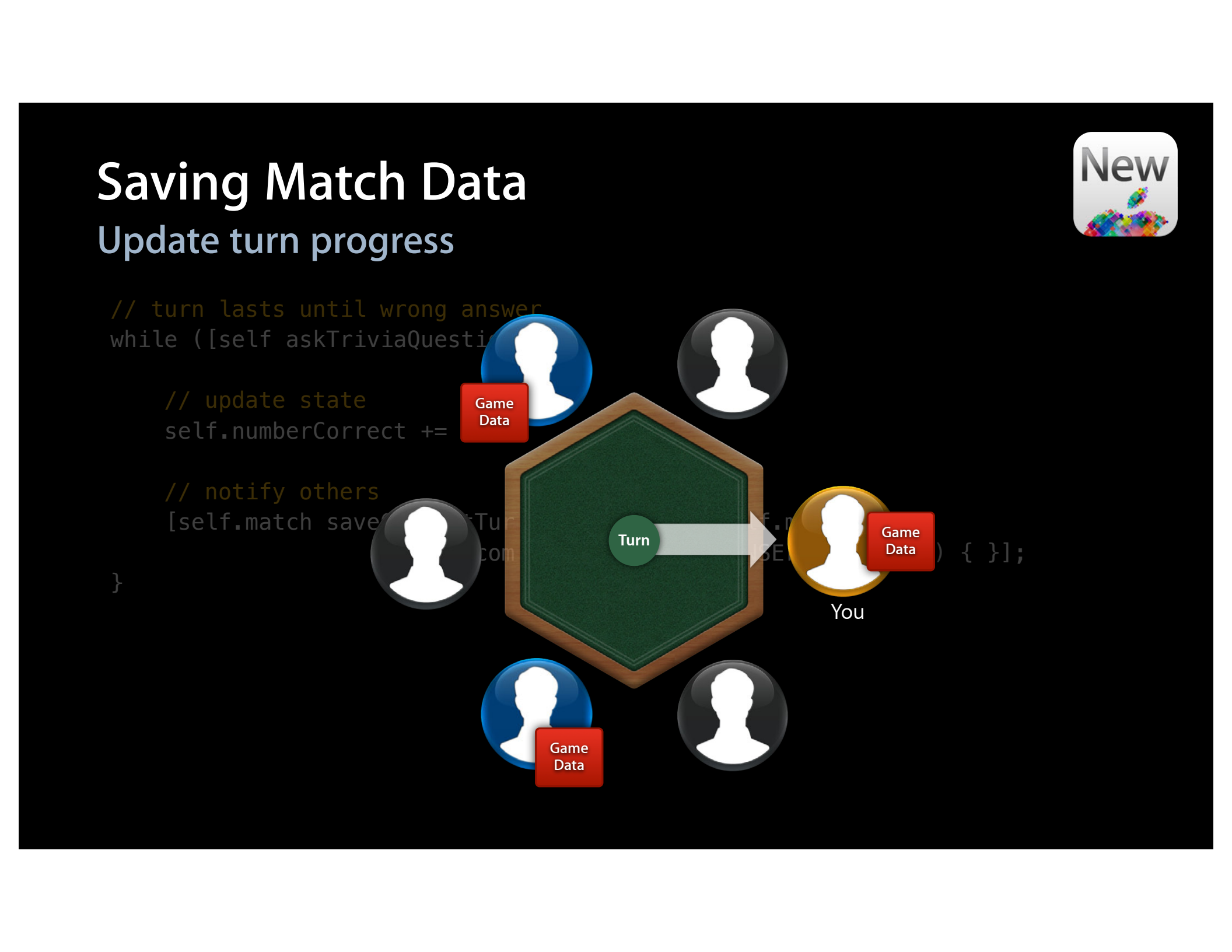
Update turn progress


```
// turn lasts until wrong answer
while ([self askTriviaQuesti

    // update state
    self.numberCorrect += 1

    // notify others
    [self.match save
                                    ) { }];
}
```



Saving Match Data

Update turn progress




```
// turn lasts until wrong answer
while ([self askTriviaQuesti

    // update state
    self.numberCorrect +=

    // notify others
    [self.match save
                                    ) { }];
}
```

Turn Notifications

GKTurnBasedEventHandler

Event	Notification	Method

Turn Notifications

GKTurnBasedEventHandler

Event	Notification	Method
Receive Invite	Push	`- handleTurnEventForMatch:`

Turn Notifications

GKTurnBasedEventHandler

Event	Notification	Method
Receive Invite	Push	`- handleTurnEventForMatch:`
Your Turn	Push	`- handleTurnEventForMatch:`

Turn Notifications

GKTurnBasedEventHandler

Event	Notification	Method
Receive Invite	Push	`- handleTurnEventForMatch:`
Your Turn	Push	`- handleTurnEventForMatch:`
Turn Passed	Running Instances	`- handleTurnEventForMatch:`

Turn Notifications

GKTurnBasedEventHandler

Event	Notification	Method
Receive Invite	Push	`- handleTurnEventForMatch:`
Your Turn	Push	`- handleTurnEventForMatch:`
Turn Passed	Running Instances	`- handleTurnEventForMatch:`
Save Data	Running Instances	`- handleTurnEventForMatch:`

Turn Notifications

GKTurnBasedEventHandler

Event	Notification	Method
Receive Invite	Push	`- handleTurnEventForMatch:`
Your Turn	Push	`- handleTurnEventForMatch:`
Turn Passed	Running Instances	`- handleTurnEventForMatch:`
Save Data	Running Instances	`- handleTurnEventForMatch:`
Match Ended	Push	`- handleMatchEnded:`

Turn Notifications

GKTurnBasedEventHandler Delegate

```
// Someone in some match has taken their turn
- (void)handleTurnEventForMatch:(GKTurnBasedMatch *)match
{
    // If it's now my turn, inform the user
    if ([match.currentParticipant.playerID isEqualToString:localPlayer.playerID])
       [GKNotificationBanner showBannerWithTitle:@"It's Your Turn!"
            message:match.message completionHandler:nil];

    // if I'm currently viewing this match, update my UI
    if ([match.matchID isEqualToString:self.currentMatch.matchID])
       [self updateUIWithMatch:match);

    ...

}
```

Turn Notifications

GKTurnBasedEventHandler Delegate

```
// Someone in some match has taken their turn
- (void)handleTurnEventForMatch:(GKTurnBasedMatch *)match
{
    // If it's now my turn, inform the user
    if ([match.currentParticipant.playerID isEqualToString:localPlayer.playerID])
       [GKNotificationBanner showBannerWithTitle:@"It's Your Turn!"
            message:match.message completionHandler:nil];

    // if I'm currently viewing this match, update my UI
    if ([match.matchID isEqualToString:self.currentMatch.matchID])
       [self updateUIWithMatch:match);

    ...

}
```

Turn Notifications

GKTurnBasedEventHandler Delegate

```
// Someone in some match has taken their turn
- (void)handleTurnEventForMatch:(GKTurnBasedMatch *)match
{
    // If it's now my turn, inform the user
    if ([match.currentParticipant.playerID isEqualToString:localPlayer.playerID])
       [GKNotificationBanner showBannerWithTitle:@"It's Your Turn!"
            message:match.message completionHandler:nil];

    // if I'm currently viewing this match, update my UI
    if ([match.matchID isEqualToString:self.currentMatch.matchID])
       [self updateUIWithMatch:match);

    ...

}
```

Turn Notifications

GKTurnBasedEventHandler Delegate

```
// Someone in some match has taken their turn
- (void)handleTurnEventForMatch:(GKTurnBasedMatch *)match
{
    // If it's now my turn, inform the user
    if ([match.currentParticipant.playerID isEqualToString:localPlayer.playerID])
       [GKNotificationBanner showBannerWithTitle:@"It's Your Turn!"
            message:match.message completionHandler:nil];

    // if I'm currently viewing this match, update my UI
    if ([match.matchID isEqualToString:self.currentMatch.matchID])
       [self updateUIWithMatch:match);

    ...

}
```

Taking Turns

Things to remember

- Current participant
  - Update data
    - Others are read-only
  - Pass the turn
    - Next participant gets notified
- Others may quit
- Turn events
  - Sent to all running instances
  - Add unobtrusive UI to inform player

Quitting a Match

Player resigns or is just leaving

- While my turn
  - Update game state
  - Set my outcome to Quit
  - Need to pass the turn
- Not my turn
  - Inform Game Center

Ending a Match

Game over

- Must have turn
- Set outcomes for all participants

```
GKTurnBasedMatchOutcomeWon
GKTurnBasedMatchOutcomeLost
GKTurnBasedMatchOutcomeTied
      ...
GKTurnBasedMatchOutcomeCustom(0–255)
```

- Optional message
- Inform Game Center
  - All participants gets notified

Ending a Match

-endMatchInTurnWithMatchData

```
// Set participant match outcomes
for (GKTurnBasedParticipant *participant in match.participants) {
     participant.matchOutcome = GKTurnBasedMatchOutcomeTied;
}

// Set final game state
NSData *finalMatchData = [self resolveGame];

// Set optional message
match.message = @"I really did win the Kessel Run!";

// End the match
[match endMatchInTurnWithMatchData:finalMatchData
     completionHandler:^(NSError *error) {
}];
```

Ending a Match

-endMatchInTurnWithMatchData

```
// Set participant match outcomes
for (GKTurnBasedParticipant *participant in match.participants) {
     participant.matchOutcome = GKTurnBasedMatchOutcomeTied;
}

// Set final game state
NSData *finalMatchData = [self resolveGame];

// Set optional message
match.message = @"I really did win the Kessel Run!";

// End the match
[match endMatchInTurnWithMatchData:finalMatchData
     completionHandler:^(NSError *error) {
}];
```

Ending a Match

-endMatchInTurnWithMatchData

```
// Set participant match outcomes
for (GKTurnBasedParticipant *participant in match.participants) {
     participant.matchOutcome = GKTurnBasedMatchOutcomeTied;
}

// Set final game state
NSData *finalMatchData = [self resolveGame];

// Set optional message
match.message = @"I really did win the Kessel Run!";

// End the match
[match endMatchInTurnWithMatchData:finalMatchData
     completionHandler:^(NSError *error) {
}];
```

Ending a Match

-endMatchInTurnWithMatchData

```
// Set participant match outcomes
for (GKTurnBasedParticipant *participant in match.participants) {
     participant.matchOutcome = GKTurnBasedMatchOutcomeTied;
}

// Set final game state
NSData *finalMatchData = [self resolveGame];

// Set optional message
match.message = @"I really did win the Kessel Run!";

// End the match
[match endMatchInTurnWithMatchData:finalMatchData
     completionHandler:^(NSError *error) {
}];
```

Ending a Match

-endMatchInTurnWithMatchData

```
// Set participant match outcomes
for (GKTurnBasedParticipant *participant in match.participants) {
     participant.matchOutcome = GKTurnBasedMatchOutcomeTied;
}

// Set final game state
NSData *finalMatchData = [self resolveGame];

// Set optional message
match.message = @"I really did win the Kessel Run!";

// End the match
[match endMatchInTurnWithMatchData:finalMatchData
     completionHandler:^(NSError *error) {
}];
```

Rematch API

Play it again

- GKTurnBasedMatch

```
[self.match rematchWithCompletionHandler:^(GKTurnBasedMatch *match,
NSError) {
    // start your game
}];
```

Rematch API

Play it again

- GKTurnBasedMatch

```
[self.match rematchWithCompletionHandler:^(GKTurnBasedMatch *match,
NSError) {
    // start your game
}];
```

- GKMatch

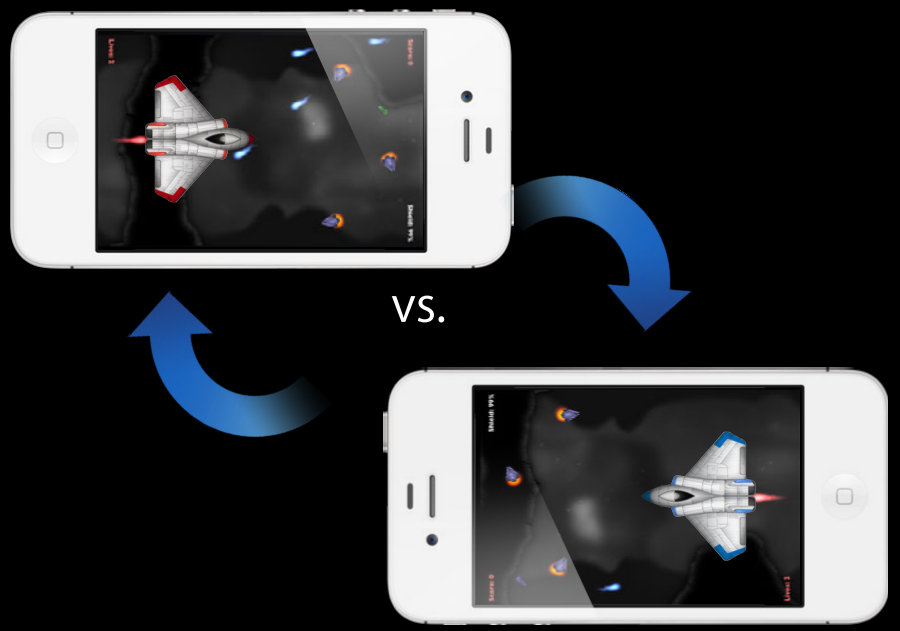
```
[self.match rematchWithCompletionHandler:^(GKMatch *match, NSError *error) {
    // start your game
}];
```

Multiplayer Summary

- Popular feature
  - Adds longevity to your app
- Styles of multiplayer
- MatchMaker UI
- Programmatic matchmaking
- Peer-to-peer communications
- Turn-Based gaming

Multiplayer Summary

- Popular feature
  - Adds longevity to your app
- Styles of multiplayer
- MatchMaker UI
- Programmatic matchmaking
- Peer-to-peer communications
- Turn-Based gaming



More Information

Allan Schaffer
Graphics and Game Technologies Evangelist
aschaffer@apple.com

Documentation
Game Kit Programming Guide
http://developer.apple.com/library/ios/

Apple Developer Forums
http://devforums.apple.com

Related Sessions

What's New in Game Center	Mission Tuesday 4:30PM
Integrating Your Games with Game Center	Pacific Heights Wednesday 4:30PM
What's New in iTunes Connect for App Developers	Nob Hill Thursday 9:00AM
Building Game Center Games for OS X	Pacific Heights Thursday 11:30AM

Labs

Game Center Lab	Graphics, Media & Games Lab B Thursday 2:00PM
Game Center Lab	Graphics, Media & Games Lab C Friday 9:00AM


WWDC2012