Turn-Based Gaming with Game Center

Session 412

Gabriel Belinsky
iOS Engineering

These are confidential sessions—please refrain from streaming, blogging, or taking pictures

GameKit Framework

Game Center App
GameKit Framework
Game Center Services

Introduction

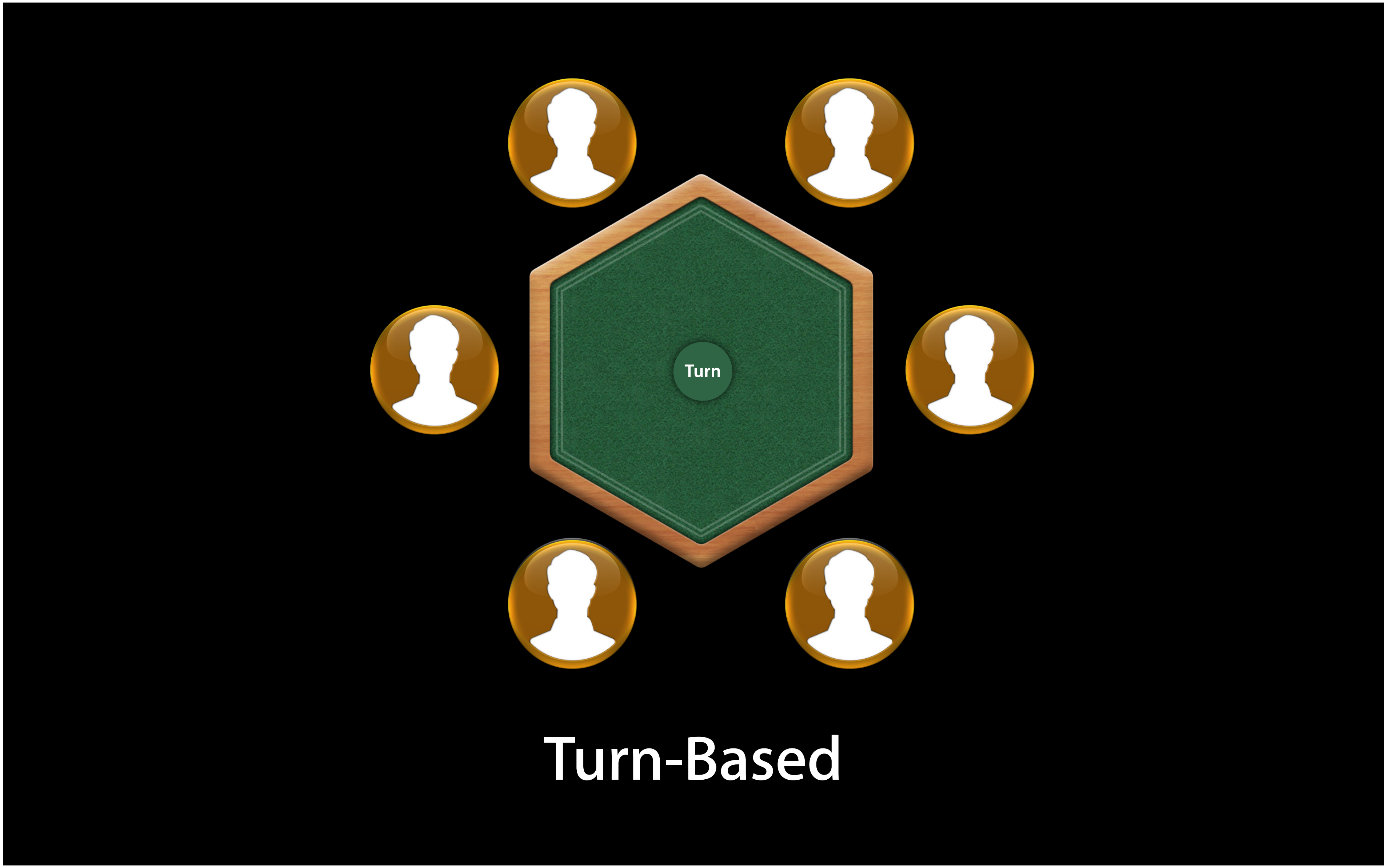
- Turn-based gaming is new for iOS 5
- Learn about the API and user interface
- See how to apply it in a variety of genres
- Gain insight into managing turn-based game state
- Get details about how to create your own game

What You Will Learn

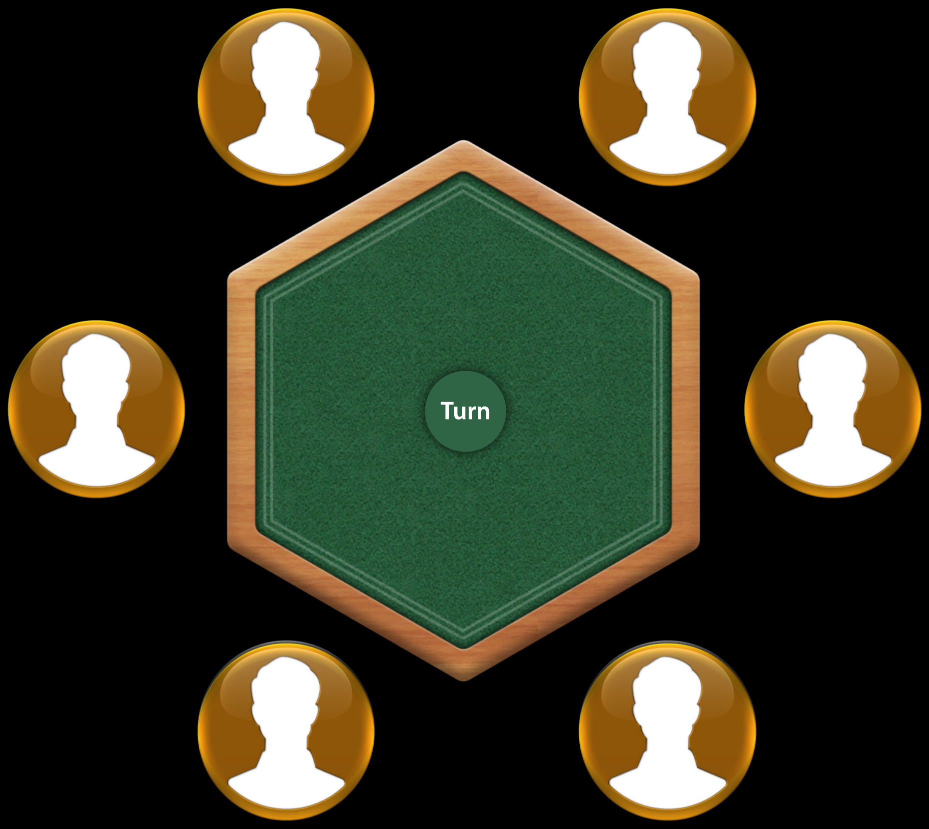
- Game Center turn-based API basics
- Starting a new match
- Making turns in match
- Handling notifications
- Ending a match

Turn-Based Gaming Examples

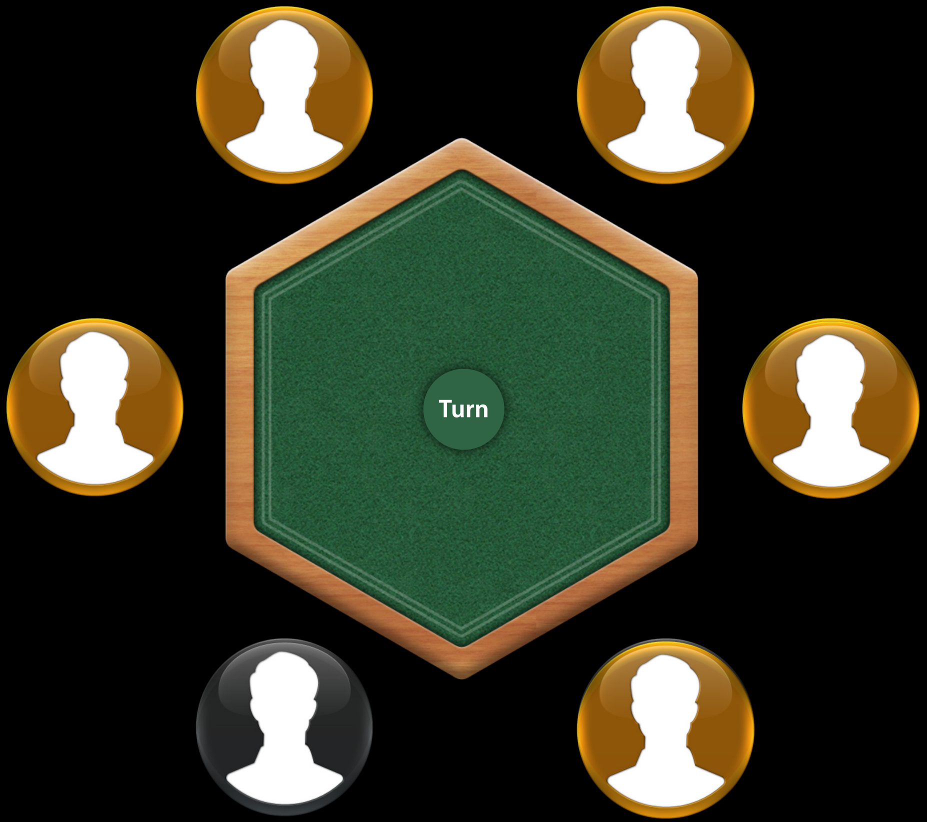
- Two player games
  - Chess/checkers/backgammon/go
- Multiplayer games
  - Bridge/cribbage
- More complex games
  - Coming soon in the app store



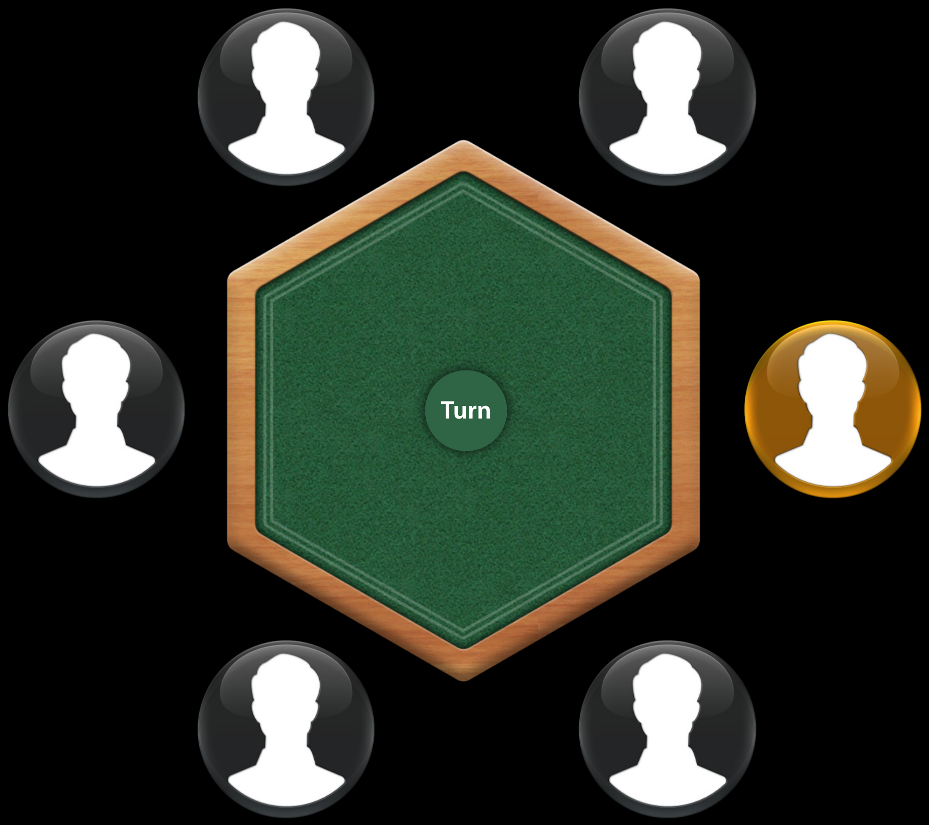
Turn-Based



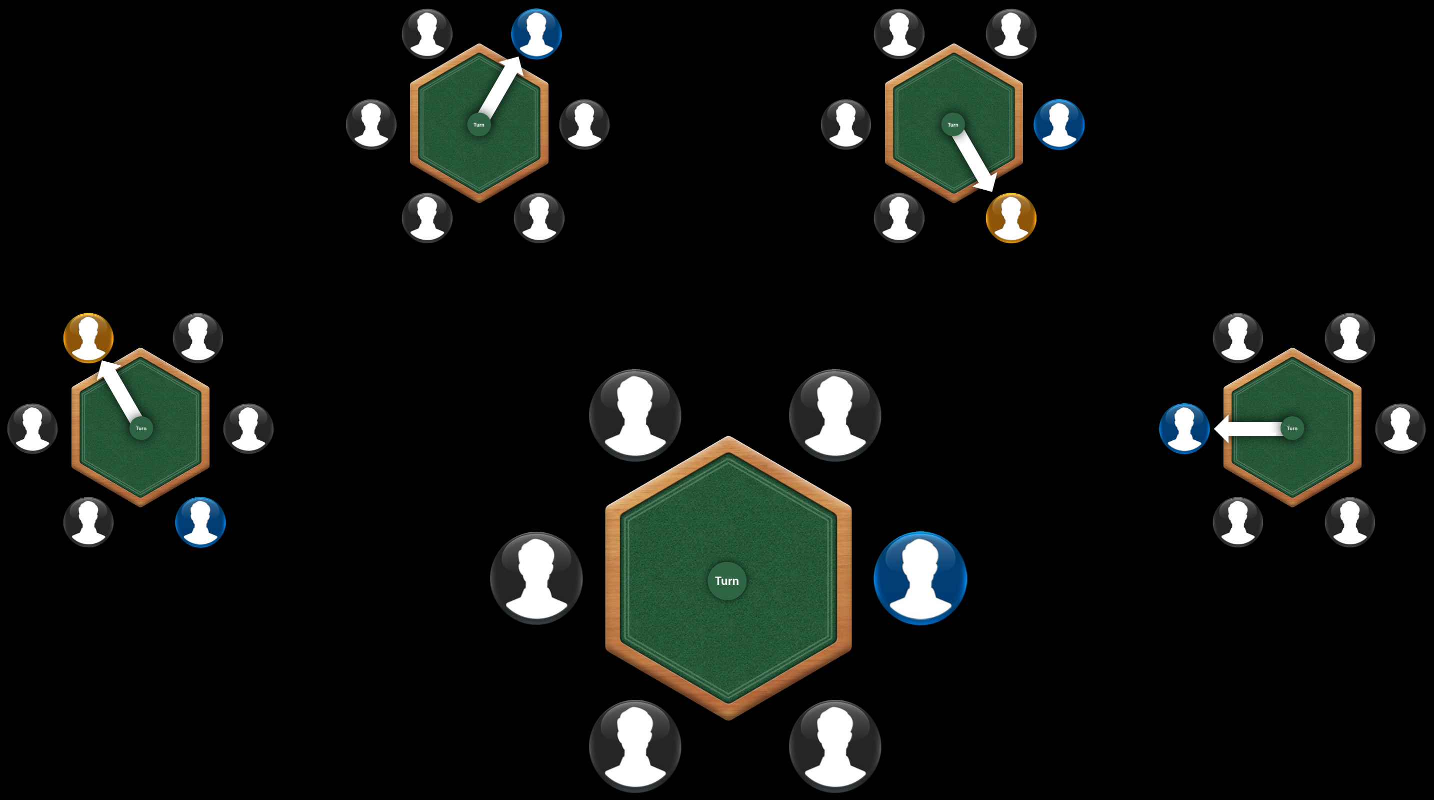
Auto-matching



Control of Next Turn



Multiple Matches

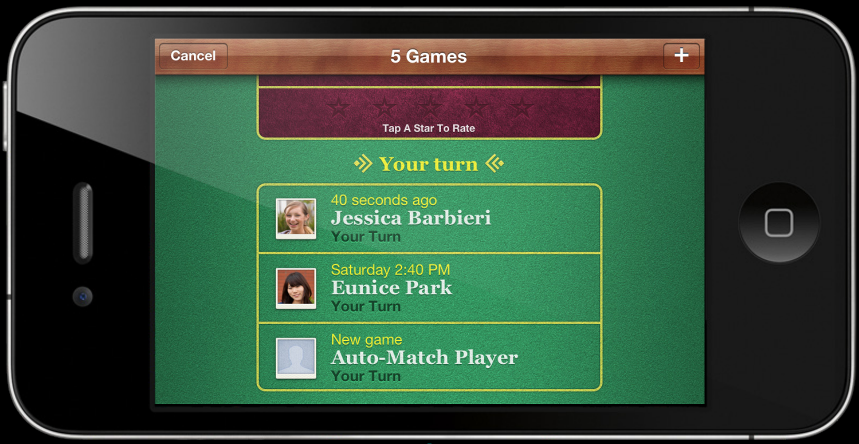


Multiple Matches

Match Walkthrough

Rochambeau multiplayer session

Start Game



40 seconds ago
Jessica Barbieri
Your Turn

Saturday 2:40 PM
Eunice Park
Your Turn

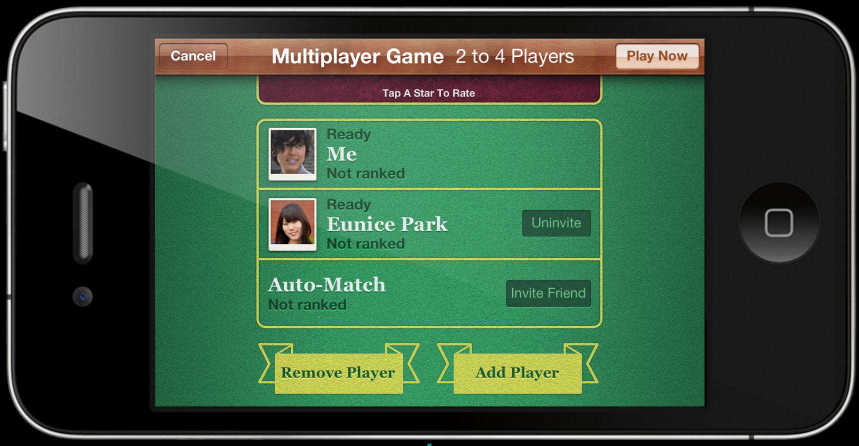
New game
Auto-Match Player
Your Turn

Game Outcome

Match Walkthrough

Rochambeau multiplayer session

Start Game



Cancel

Multiplayer Game 2 to 4 Players

Play Now

Tap A Star To Rate

Ready
Me
Not ranked

Ready
Eunice Park
Not ranked
Uninvite

Auto-Match
Not ranked
Invite Friend

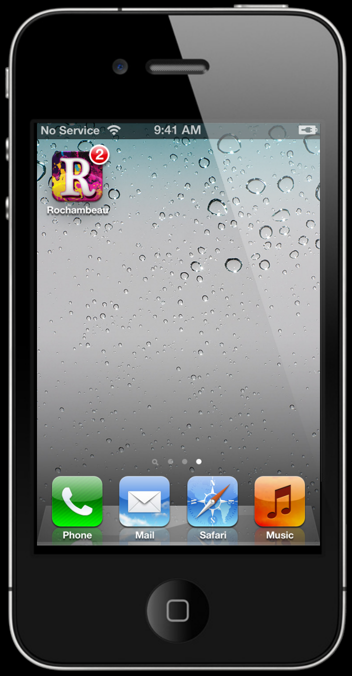
Remove Player

Add Player

Game Outcome

Match Walkthrough

Rochambeau multiplayer session



Demo

Jordan Ceccarelli
Ace Achievement Hunter

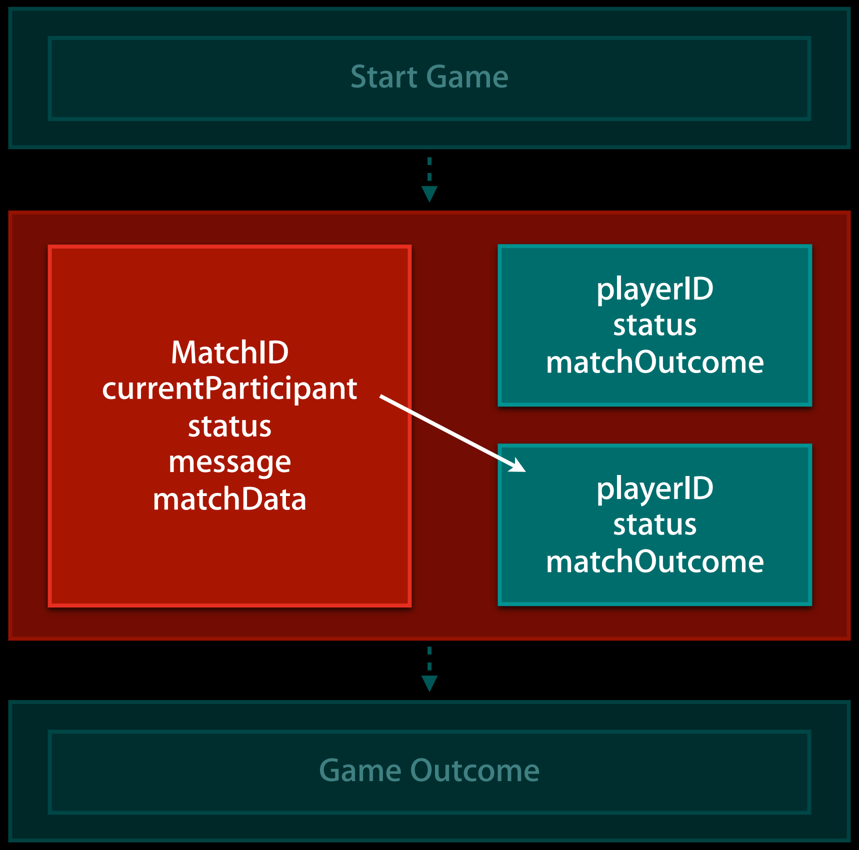
Match

- Each game instance is a match
- Allows multiple simultaneous matches
- Main entry point to API
  - Participants
  - End turns
  - Quit matches
  - End games
  - List games

Participants

- Details for each entity in the match
  - playerID
  - Statuses
  - Match outcomes
- May be a player or an open matching position
  - Matching position has no playerID

Match Properties



Start Game

MatchID
currentParticipant
status
message
matchData

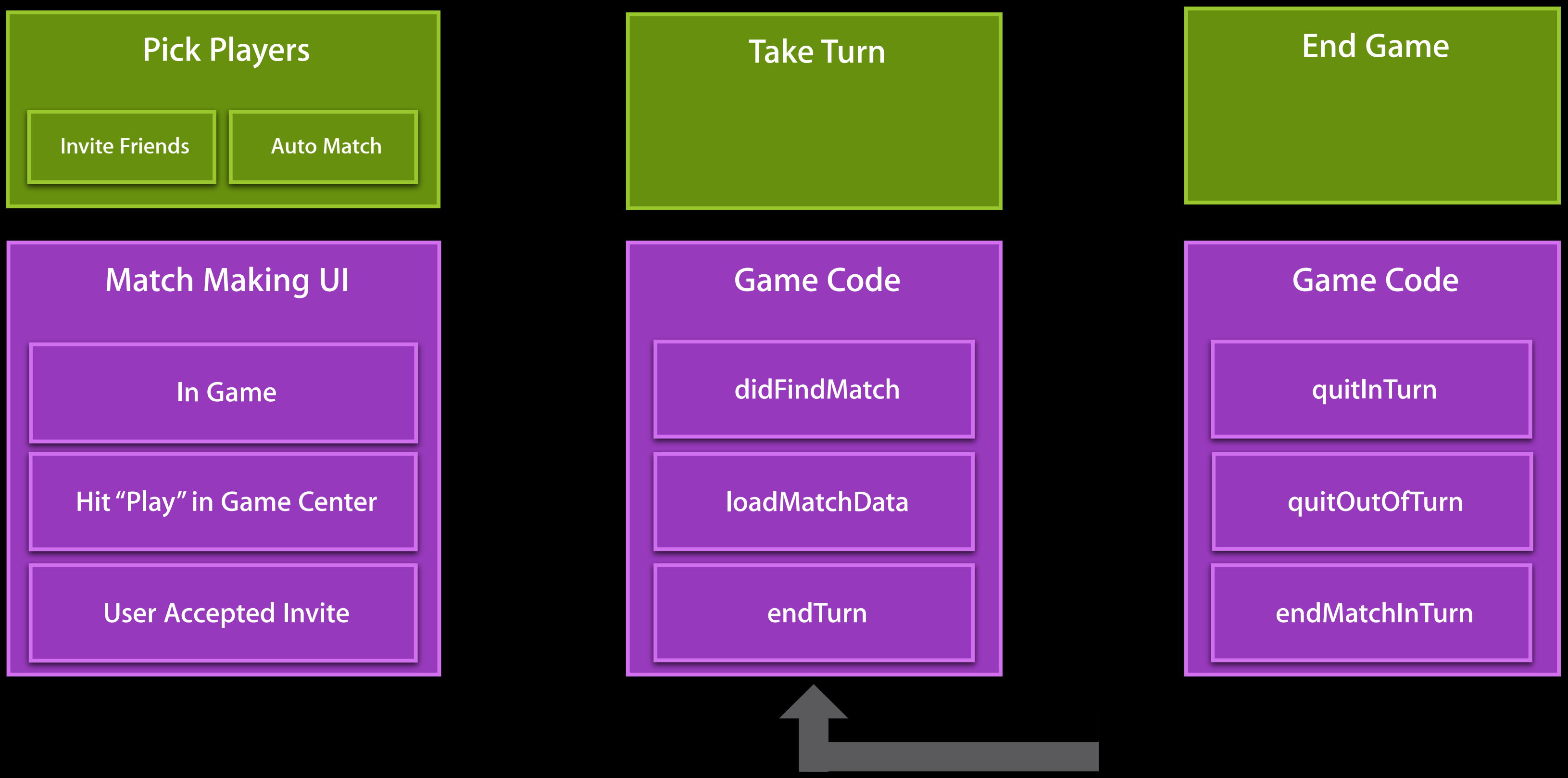
playerID
status
matchOutcome

playerID
status
matchOutcome

Game Outcome

Turn-Based Gaming

Tasks



Pick Players

Invite Friends

Auto Match

Match Making UI

In Game

Hit “Play” in Game Center

User Accepted Invite

Take Turn

Game Code

didFindMatch

loadMatchData

endTurn

End Game

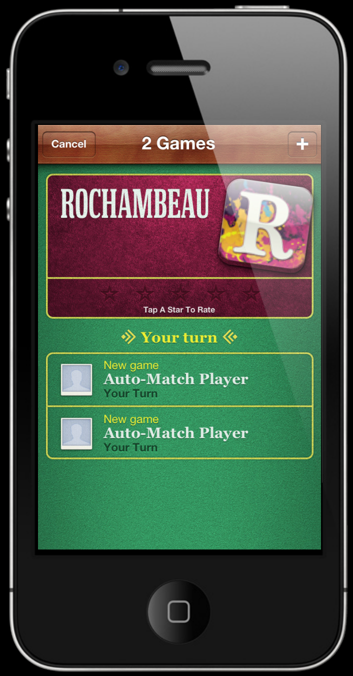
Game Code

quitInTurn

quitOutOfTurn

endMatchInTurn

View Controller



- Display and manage matches
  - Choose a match to play
  - Quit from a match
  - Remove a match from the list
- Create new matches
  - Invites
  - Auto-match

Using the View Controller

```
GKMatchRequest *request = [self matchRequest];

GKTurnBasedMatchmakerViewController *viewController =
     [[GKTurnBasedMatchmakerViewController alloc]
           initWithMatchRequest: request]
viewController.showExistingMatches = YES;
viewController.turnBasedMatchmakerDelegate = self;
[rootViewController presentViewController:viewController animated:YES
     completion:nil];
```

View Controller Delegate

```
- (void)turnBasedMatchmakerViewController:
     GKTurnBasedMatchmakerViewController *)viewController
     didFindMatch:(GKTurnBasedMatch *)match
{
     [rootViewController dismissViewControllerAnimated:YES completion:nil];
     // show the state of the game
     NSString *currentPlayerID = match.currentParticipant.playerID;
     if (currentPlayerID && [currentPlayerID isEqualToString:[GKLocalPlayer
localPlayer].playerID]) {
            // local player’s turn
            // allow user actions
    }
}
```

View Controller Delegate

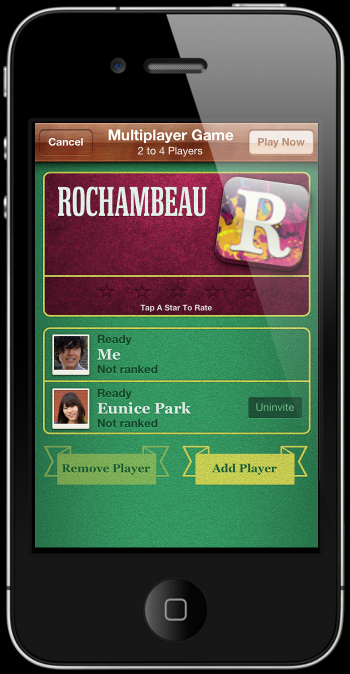
```
- (void)turnBasedMatchmakerViewController:
      GKTurnBasedMatchmakerViewController *)viewController
      playerQuitForMatch:(GKTurnBasedMatch *)match
{
      [match participantQuitInTurnWithOutcome:matchOutcome
              nextParticipant:nextParticipant
              matchData:matchData
              completionHandler:^(NSError *error) {
      }];
}
```

View Controller Delegate

```
- (void)turnBasedMatchmakerViewControllerWasCancelled:
     (GKTurnBasedMatchmakerViewController *)viewController
{
     [rootViewController dismissViewControllerAnimated:YES completion:nil];
}
```

```
- (void)turnBasedMatchmakerViewController:
     GKTurnBasedMatchmakerViewController *)viewController
     didFailWithError:(NSError *)error
{
     // handle error
     [rootViewController dismissViewControllerAnimated:YES completion:nil];
}
```

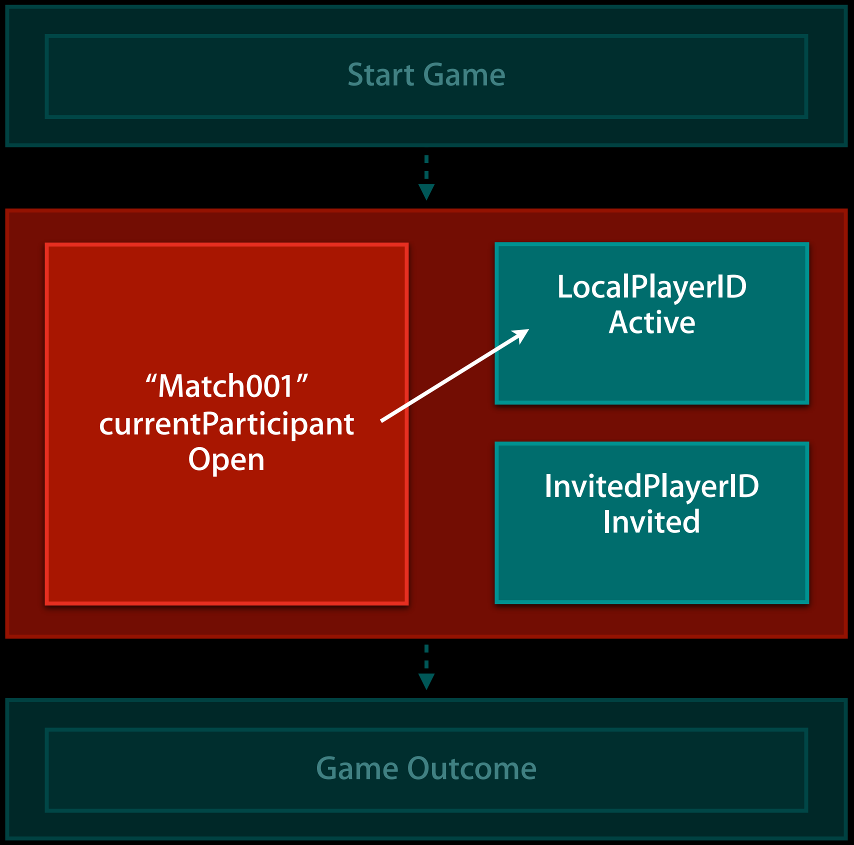
Invites



- Asynchronous invites
  - Play can begin immediately
  - Invites get sent with notification of first turn
- Declining an invite terminates the game

Match Properties

Invite



Start Game

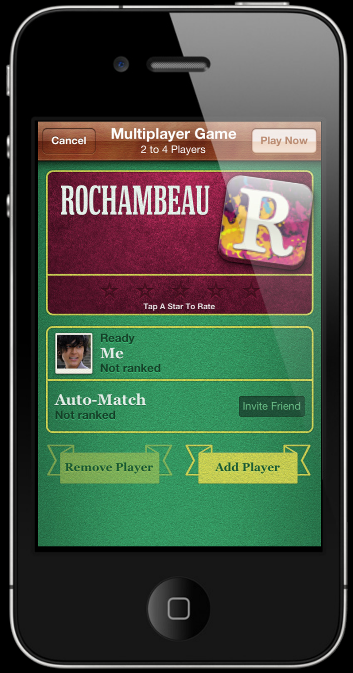
“Match001”
currentParticipant
Open

LocalPlayerID
Active

InvitedPlayerID
Invited

Game Outcome

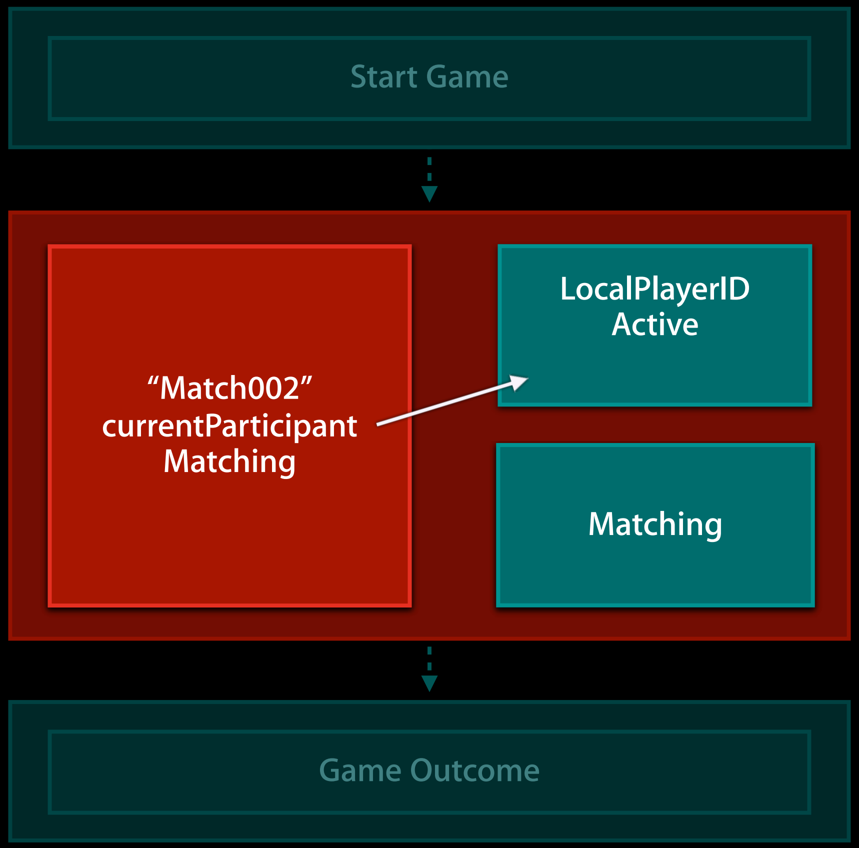
Matchmaking



- Asynchronous matchmaking
  - Play can begin immediately
  - Participants join as needed
  - May join an existing match
- Can be combined with invites
- Matched by setting nextParticipant

Match Properties

Matchmaking



Event Handling

- Invites from Game Center
- Turn notifications
- Game over notifications

Event Handler Delegate

```
- (void)handleInviteFromGameCenter:(NSArray *)playersToInvite
{
      GKMatchRequest *request = [self matchRequest];
      request.playersToInvite = playersToInvite;
      GKTurnBasedMatchmakerViewController *viewController =
             [[GKTurnBasedMatchmakerViewController alloc]
                    initWithMatchRequest:request]
      viewController.showExistingMatches = NO;
      viewController.turnBasedMatchmakerDelegate = self;
      [self presentModalViewController:viewController animated:YES];
}
```

Event Handler Delegate

```
- (void)handleTurnEventForMatch(GKTurnBasedMatch *)match
{
    if ([self.currentMatch.matchID isEqualToString:match.matchID) {
            self.currentMatch = match;
            // update ui and allow turn submission
    }
}
```

Event Handler Delegate

```
- (void)handleMatchEnded:(GKTurnBasedMatch *)match
{
     if ([self.currentMatch.matchID isEqualToString:match.matchID) {
            // show the match resolution
    }
}
```

GKTurnBasedMatch Properties

- matchID
- creationDate
- participants
- status
- currentParticipant
- matchData
- message

GKTurnBasedParticipant Properties

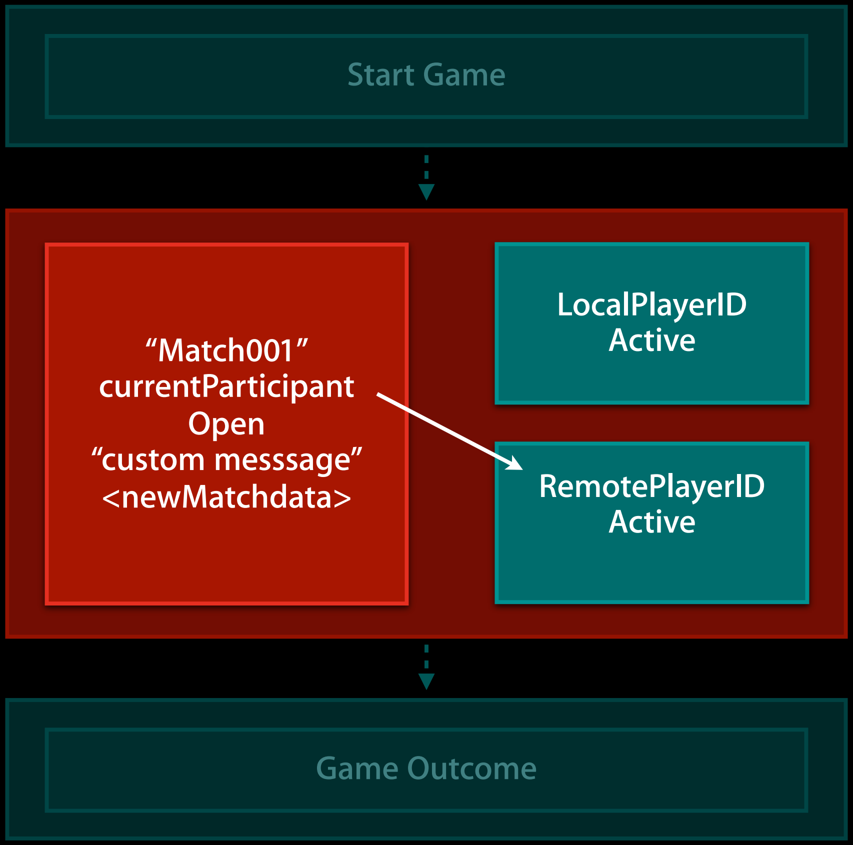
- playerID
- lastTurnDate
- status
- matchOutcome

Submitting a Turn

```
newMatchData = [self updateMatchData:match.matchData];
NSUInteger currentIndex = [match.participants
     indexOfObject:match.currentParticipant];
GKTurnBasedParticipant *nextParticipant = [match.participants
     objectAtIndex:(currentIndex + 1) % match.participants.count];
match.message = @”custom message”;
[match endTurnWithNextParticipant:nextParticipant
     matchData:newMatchData completionHandler:^(NSError *error) {
           if (error) {
                 // handle error
          }
}];
```

Match Properties

After a turn



Start Game

“Match001”
currentParticipant
Open
“custom messsage”
<newMatchdata>

LocalPlayerID
Active

RemotePlayerID
Active

Game Outcome

Turn Flow

- MatchData and nextParticipant can only be set by currentParticipant
- nextParticipant can be set to any participant in the match
- Setting nextParticipant to a one which is matching will find another player

Match Data

NSData stored on our servers

- Developer defined state of the match
- Not interpreted by us
- Limited size
- Player positions/last turns/pointer to server stored data
- Can be retrieved on multiple devices

Quitting a Match

- Quitting a match when it is your turn
- Quitting a match when it another player’s turn
- Quitting a match from the UI
    - Callback if it’s the local player’s turn

Quitting in Turn

```
newMatchData = [self updateMatchData:match.matchData];
nextParticipant = [self chooseNextParticipant:match];
participantQuitInTurnWithOutcome:GKTurnBasedMatchOutcomeQuit
     nextParticipant:nextParticipant
     matchData:newMatchData completionHandler:^(NSError *error) {
}];
```

Quitting Out of Turn

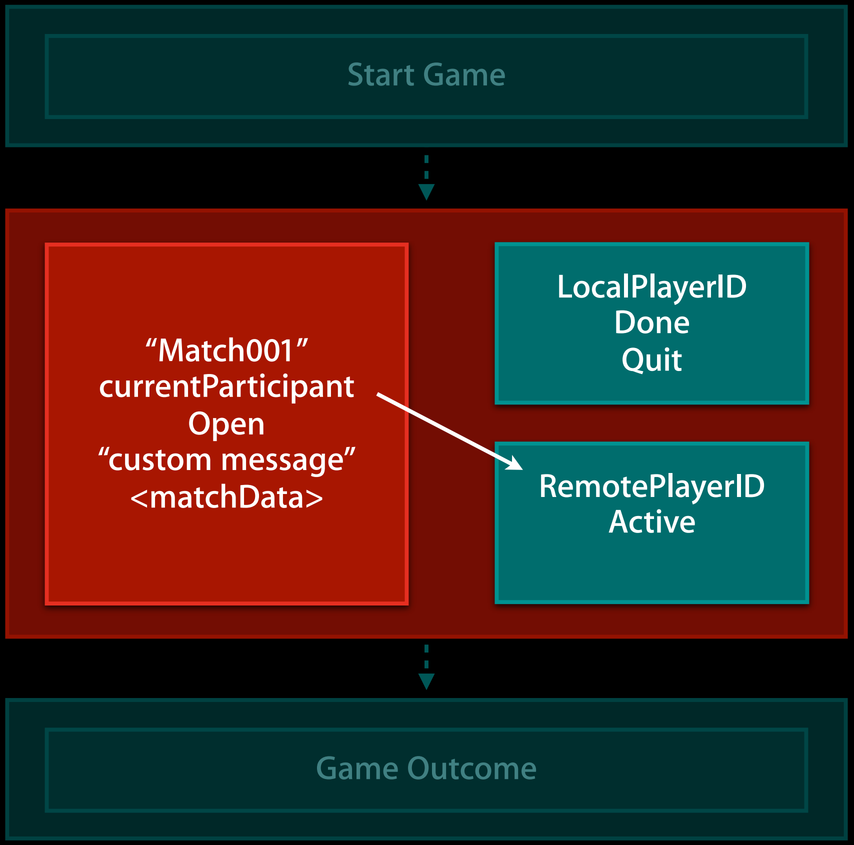
```
[match participantQuitOutOfTurnWithOutcome:GKTurnBasedMatchOutcomeQuit
     completionHandler:^(NSError *error) {
            // handle error
}];
```

User Quit from viewController

```
- (void)turnBasedMatchmakerViewController:
      GKTurnBasedMatchmakerViewController *)viewController
      playerQuitForMatch:(GKTurnBasedMatch *)match
{
      newMatchData = [self updateMatchData:match.matchData];
      nextParticipant = [self chooseNextParticipant:match];
      [match participantQuitInTurnWithOutcome:matchOutcome
             nextParticipant:nextParticipant
             matchData:newMatchData
             completionHandler:^(NSError *error) {
      }];
}
```

Match Properties

After quitting



Start Game

"Match001"
currentParticipant
Open
"custom message"
<matchData>

LocalPlayerID
Done
Quit

RemotePlayerID
Active

Game Outcome

Ending a Match

- Only the current participant can end a match
- Set outcomes for all participants
- Notification sent to all other participants

Ending a Match

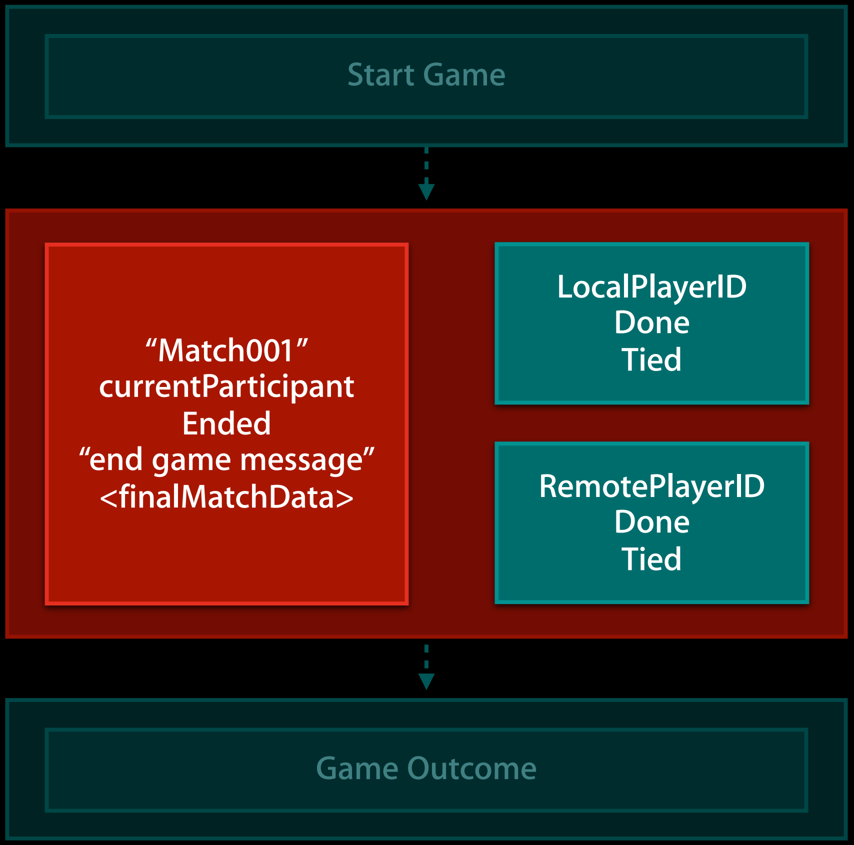
```
// set participant matchOutcomes
for (GKTurnBasedParticipant *participant in match.participants) {
     participant.matchOutcome = GKTurnBasedMatchOutcomeTied;
}
```

```
NSData *finalMatchData = [self resolveGame];
```

```
match.messasge = @”end game message”;
[match endMatchInTurnWithMatchData:finalMatchData
     completionHandler:^(NSError *error) {
}];
```

Match Properties

Match ended



Start Game

"Match001"
currentParticipant
Ended
"end game message"
<finalMatchData>

LocalPlayerID
Done
Tied

RemotePlayerID
Done
Tied

Game Outcome

Notifications and Badges

- Invite notification
  - UI to accept or decline
  - May send user to store
  - Launch app
- Turn notification
  - Option to launch app
  - Badge number on app shows outstanding turns
- End game notification
  - Option to launch app
- Launched from Game Center—list of players to invite

Customizations

- Using the API to list matches
- Determining the state of matches
- Creating custom UI
- Using GKTurnBasedMatchmakerViewController to create a new match

Summary

- Use provided UI
- Handle events
- Create matches
- Make turns
- Quit and end games

Related Sessions

Introduction to Game Center	Mission Tuesday 4:30 PM
Working with Game Center	Mission Wednesday 9:00 AM
Multiplayer Gaming with Game Center	Mission Wednesday 10:15 AM
Introduction to Game Center (Repeat)	Russian Hill Friday 9:00 AM

Labs

Game Center Lab	Graphics, Media & Games Lab A Wednesday 2:00PM
Game Center Lab	Graphics, Media & Games Lab A Friday 11:30AM

More Information

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

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

WWDC2011