MultiPlayer Gaming with Game Center

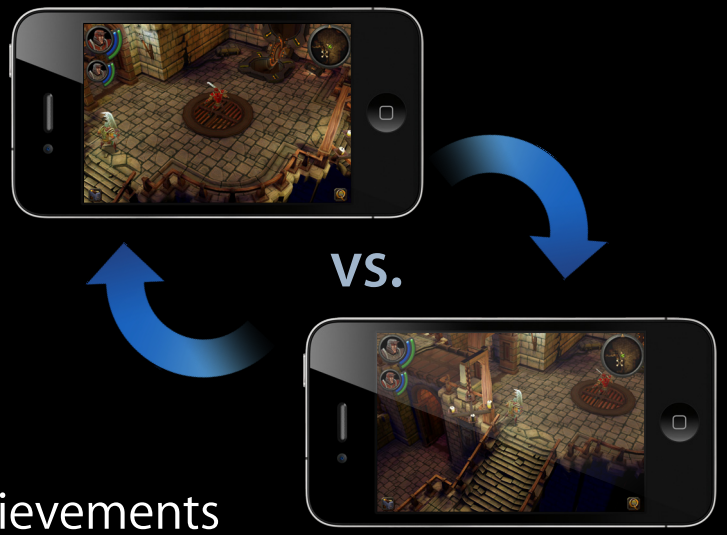
Get them to play it again and again

Session 410

Christy Warren
iPhone Development Engineer

What Is Multiplayer?

- Play with others over the network
  - Friends
  - Strangers
- Wi-Fi and cellular
- Great opportunity for social gaming
  - Foster competition and engagement
  - Increase impact of leaderboards and achievements



Why Add Multiplayer?

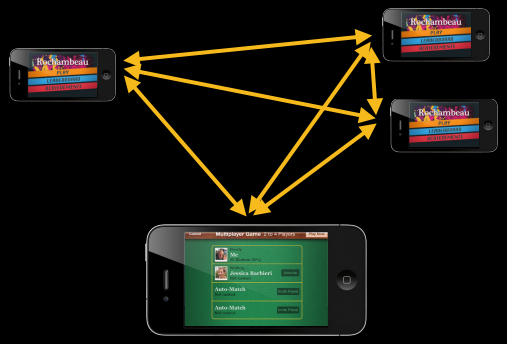
- Discover your game through invites
- Make your game stand out
  - People like to play against real opponents
  - Encourage cooperative team play
  - Many top games support multiplayer
  - Popular among players
- Increase the longevity of your game
  - Keep players coming back
- Chance for game immortality

What You Will Learn

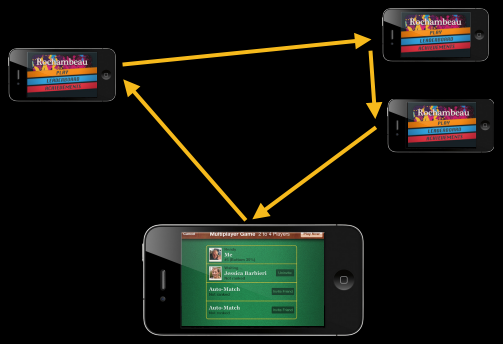
Multiplayer support

- Matchmaking UI
- Programmatic auto-match
- Peer-to-peer communications
- Server-based games
- In-game voice chat
- Setup on Game Center services

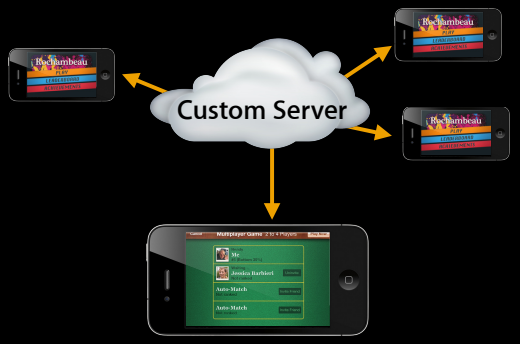
Styles of Multiplayer



Peer-to-Peer



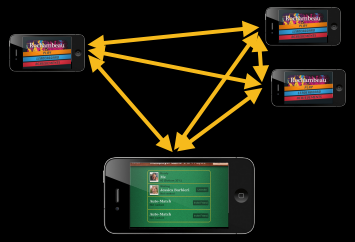
Turn-Based
(next session)



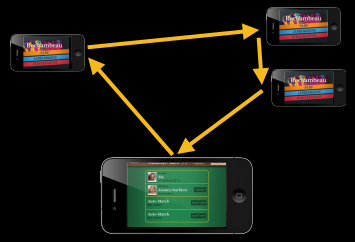
Server-Based

Styles of Multiplayer

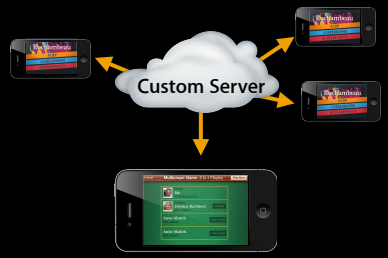
Comparison



Peer-to-Peer



Turn-Based

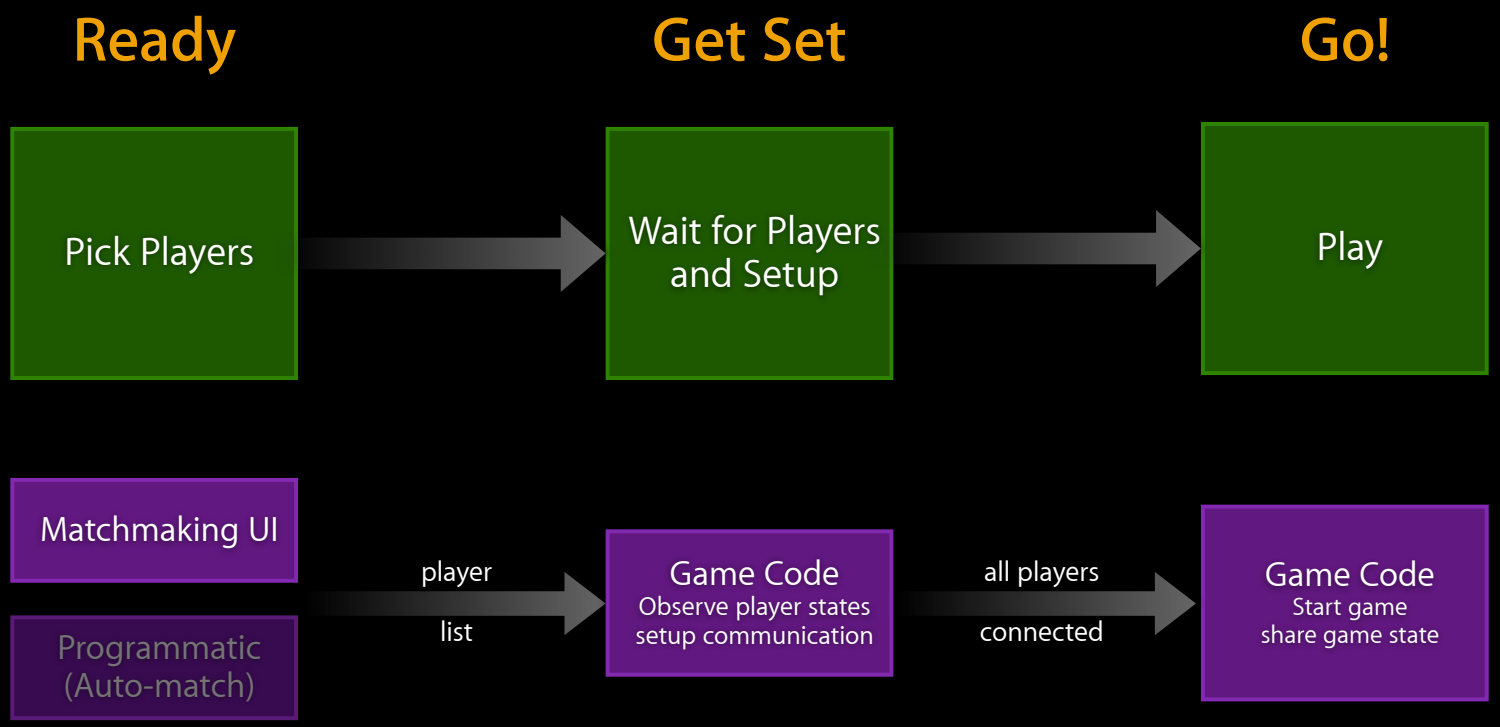


Server-Based

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

Basic Flow

All styles



Choose Players

Pick Players

Invite Friends

Auto-match

Multiplayer Entry Points

Entry Points

Hit “Play” in Game Center

User Accepted Invite

Basic Flow

Summary

Pick Players

Invite Friends | Auto-match

Matchmaking UI

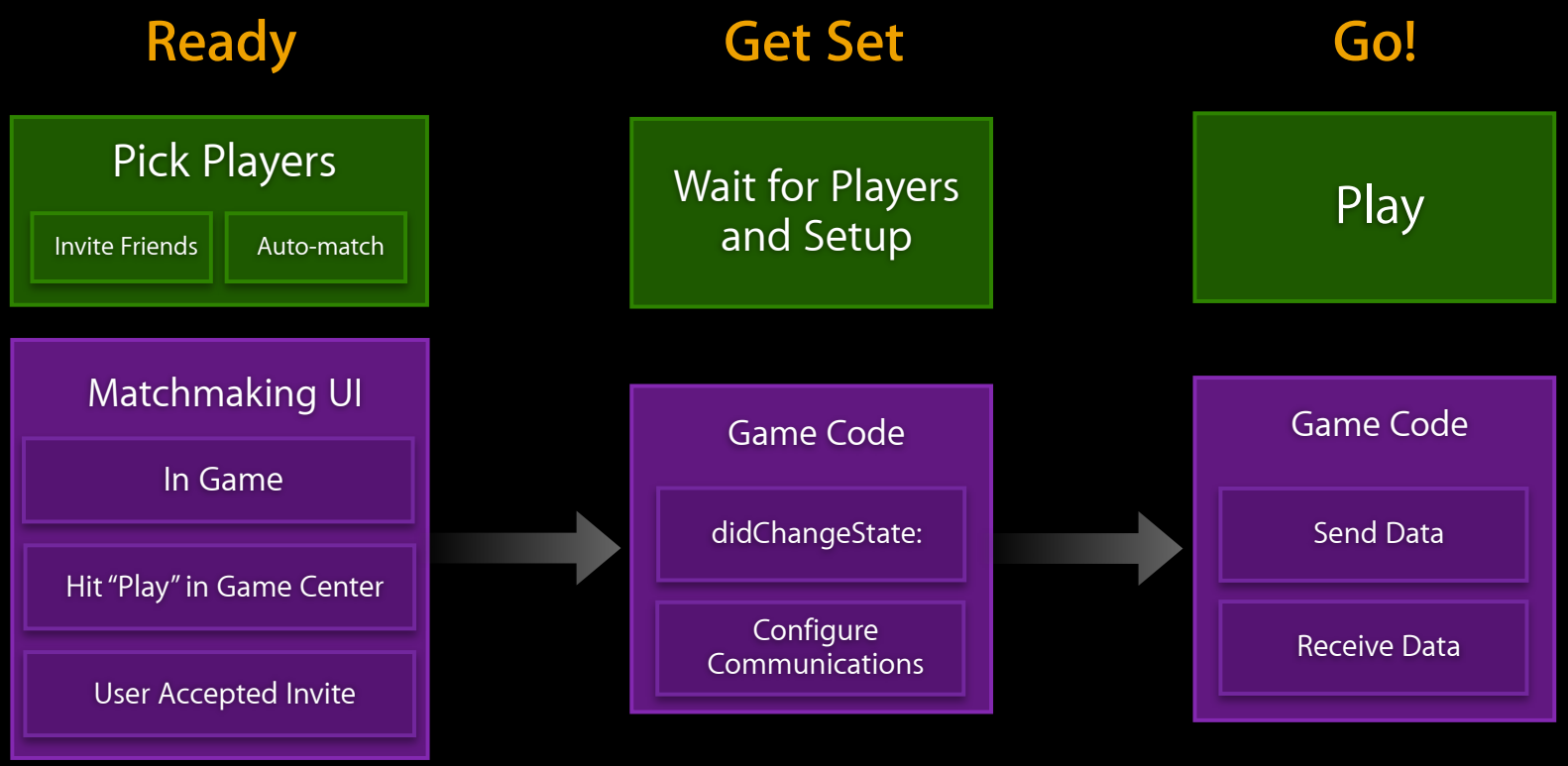
In Game

Hit "Play" in Game Center

User Accepted Invite

Multiplayer Walkthrough

Tasks



Multiplayer Preliminaries

GKLocalPlayer

- User of the device
- Responsible for authentication
- Provides friend list
- Invariant playerID
  - Save games
  - Cache data
  - Achievements
  - High scores

Multiplayer Preliminaries

Authentication

- Authenticate at launch
- Other operations will return errors if not authenticated
- May get called again later

```
GKLocalPlayer *localPlayer = [GKLocalPlayer localPlayer];
   // Authenticate and enable Game Center functionality
[localPlayer authenticateWithCompletionHandler:^(NSError *error) {
if (localPlayer.isAuthenticated) {
   // Enable Game Center features;
}
else {
   // Disable Game Center features
}
}];
```

Multiplayer Preliminaries

Thread safety

- Multiplayer APIs may not call back on the main thread
  - Delegate callbacks
  - Block-based callbacks
- Make sure you synchronize access to your data

Matchmaking UI

Matchmaking UI

Features

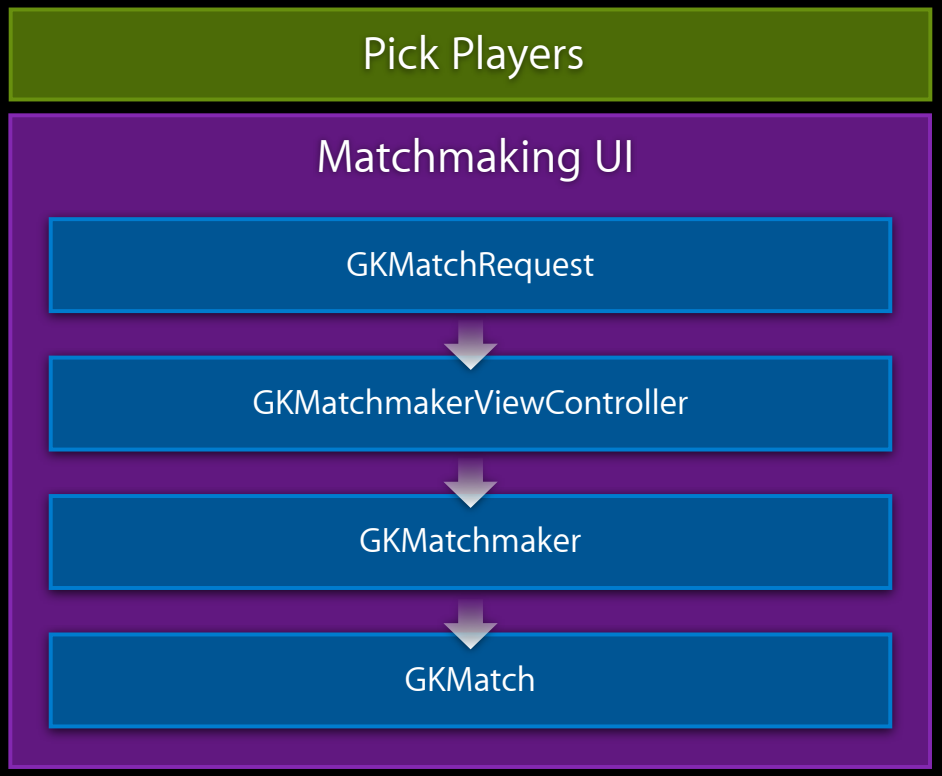
- Standard UI
  - Invite friends to play game
  - Auto-match
  - Users can rate your game
- Push notification sent to friend's device
  - Accept
  - Decline
  - Buy game

Matchmaking UI

Demo

Matchmaking UI

Classes



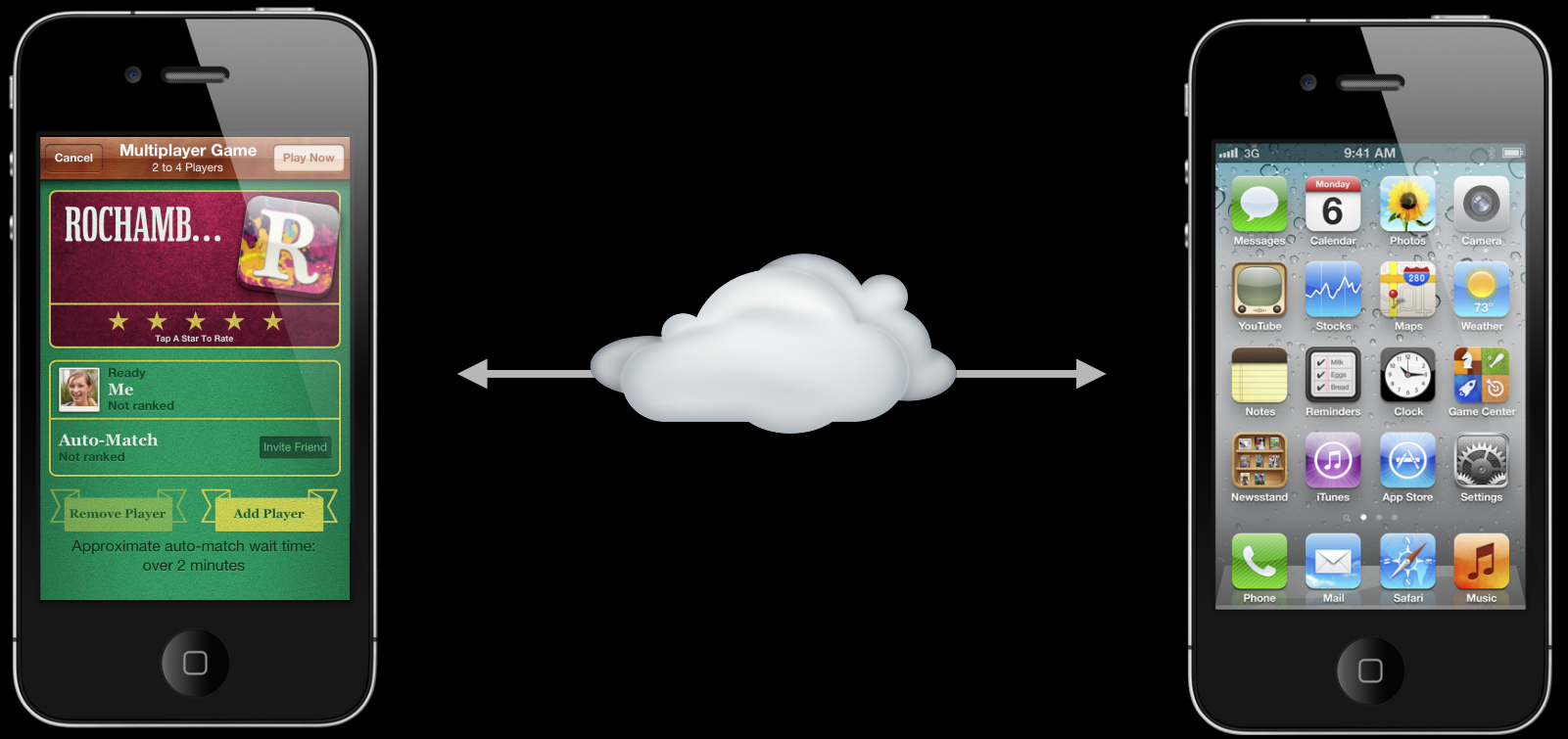
Steps to Make a Match

- Create match request
- Initialize GKMatchmakerViewController with request
- Show GKMatchmakerViewController
  - User will be able to invite players up to max players
  - Auto-match will fill in the rest
- Get match

GKMatchRequest

- Set minimum players
- Set maximum players
- Assign player group
- Assign player attributes

Matchmaking UI

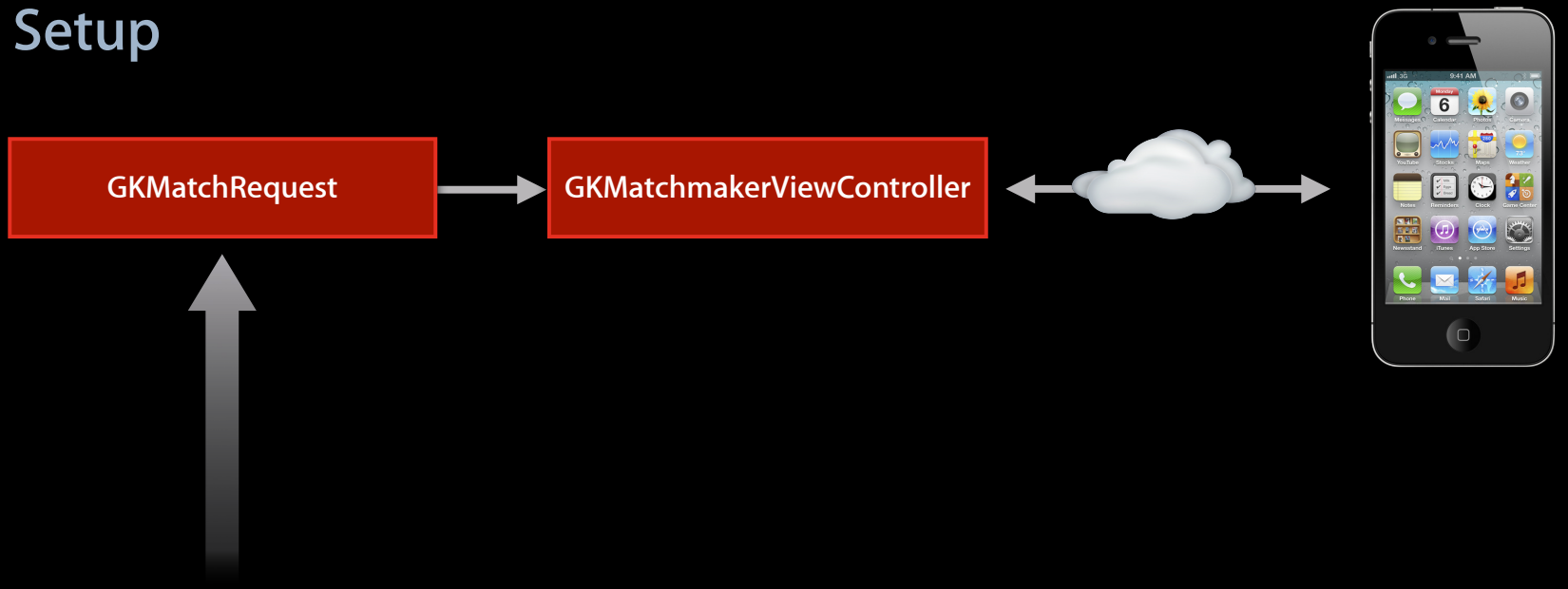


Matchmaking UI

Setup

Matchmaking UI

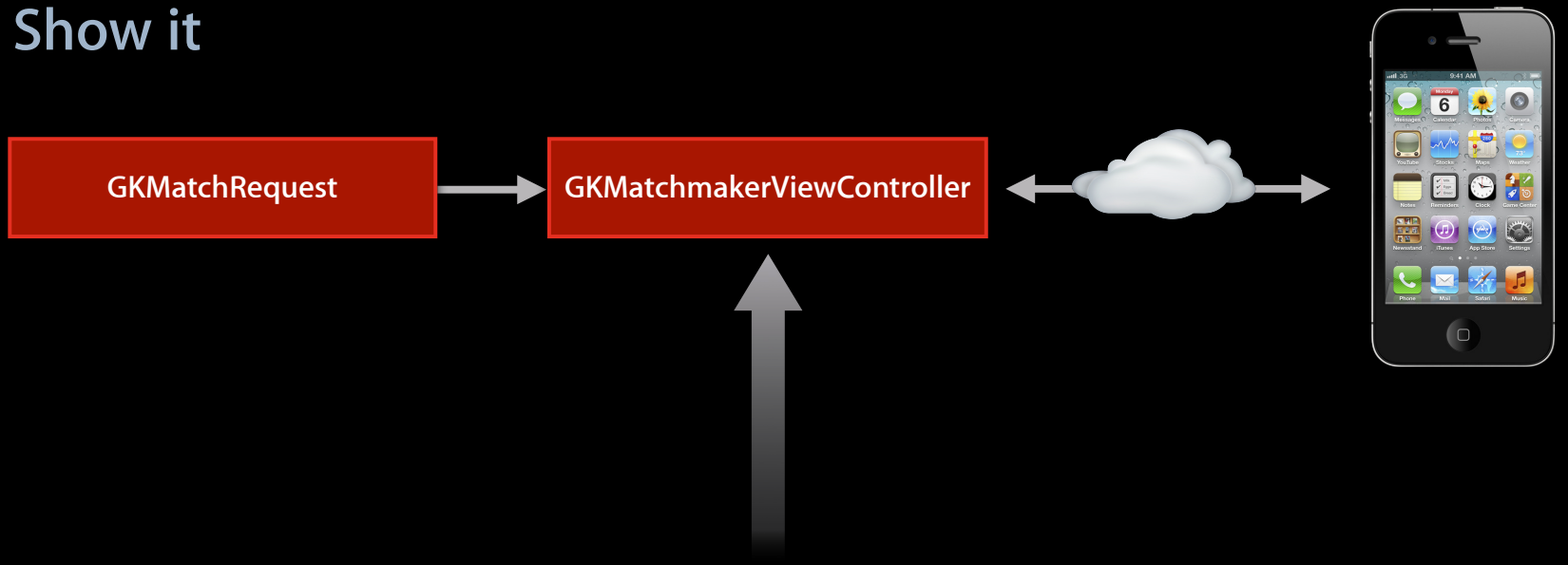
Setup




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

Matchmaking UI

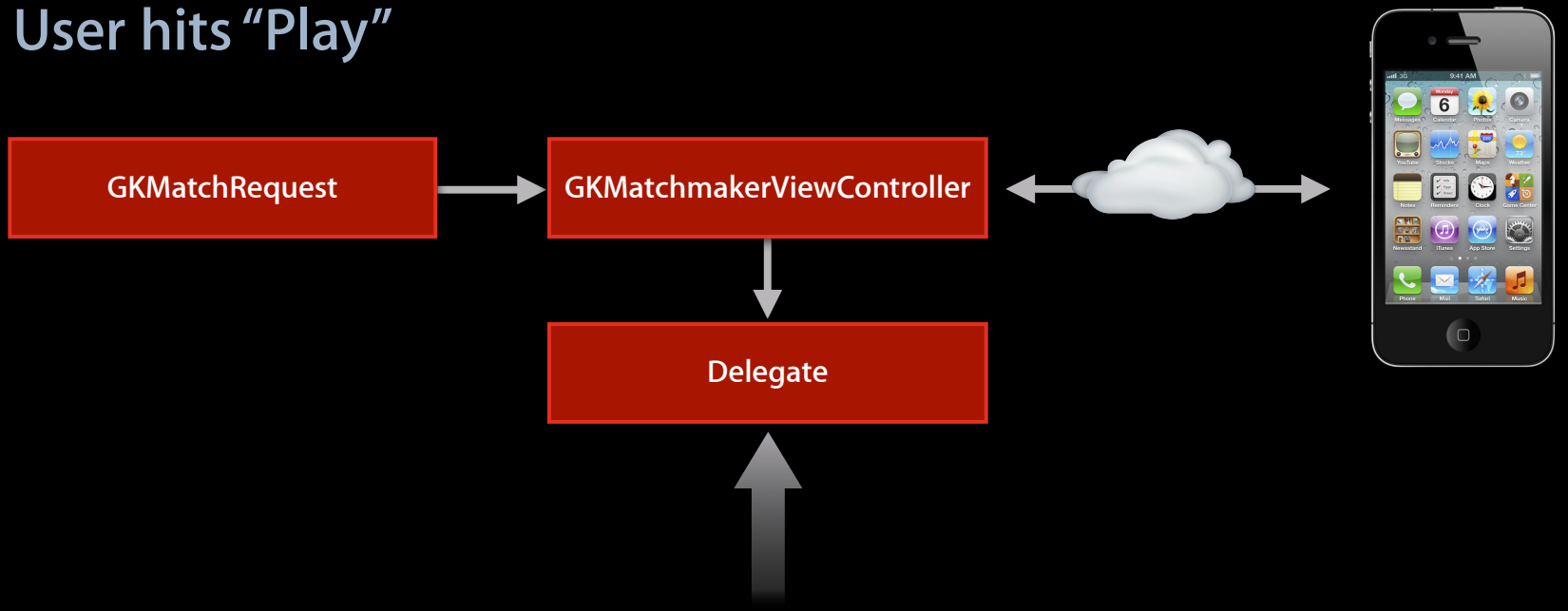
Show it




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

controller.matchmakerDelegate = self;

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

Matchmaking UI

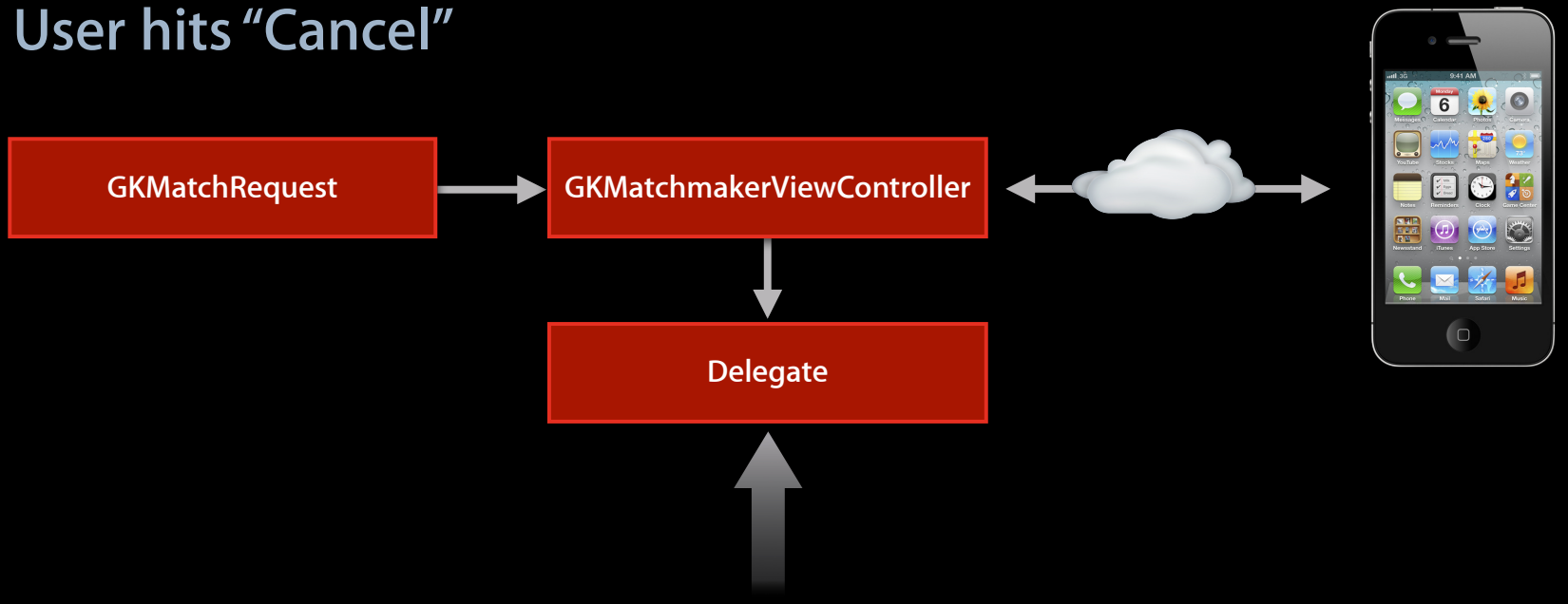
User hits “Play”




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

Matchmaking UI

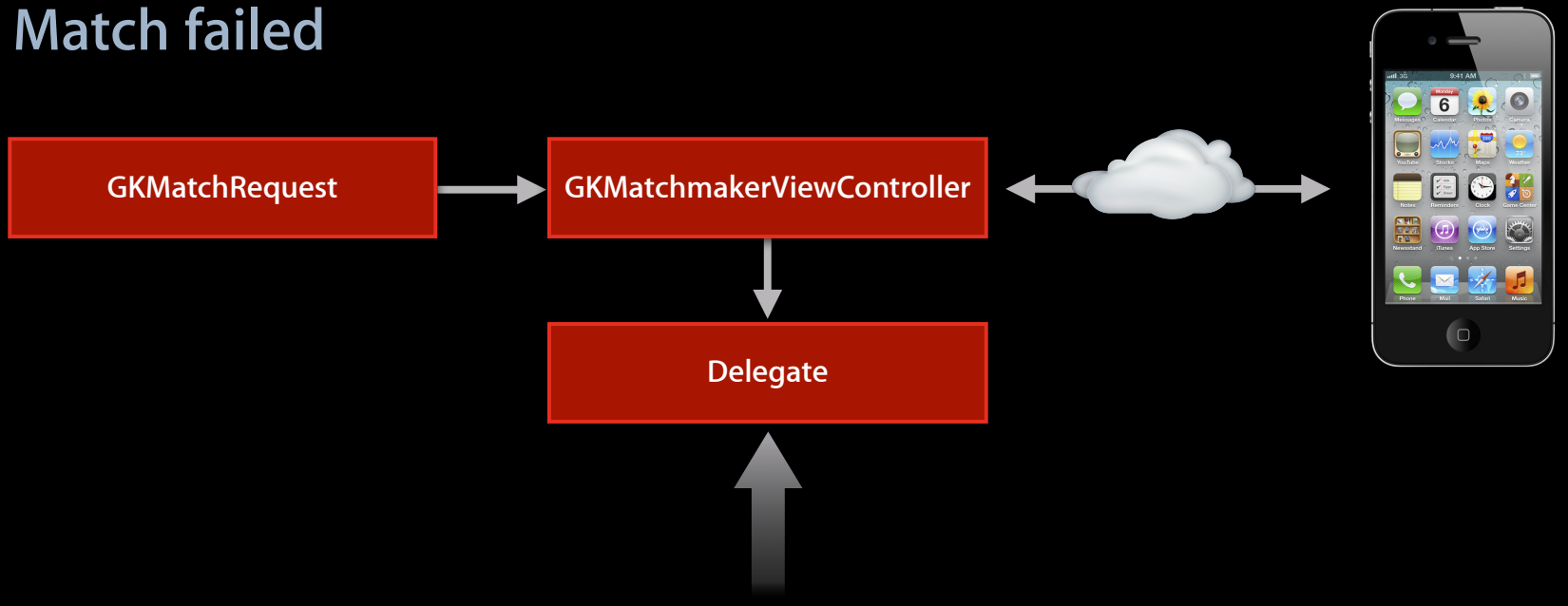
User hits “Cancel”




```
- (void)matchmakerViewControllerWasCancelled:(GKMatchmakerViewController *)
viewController
{
  // Handle cancellation
}
```

Matchmaking UI

Match failed



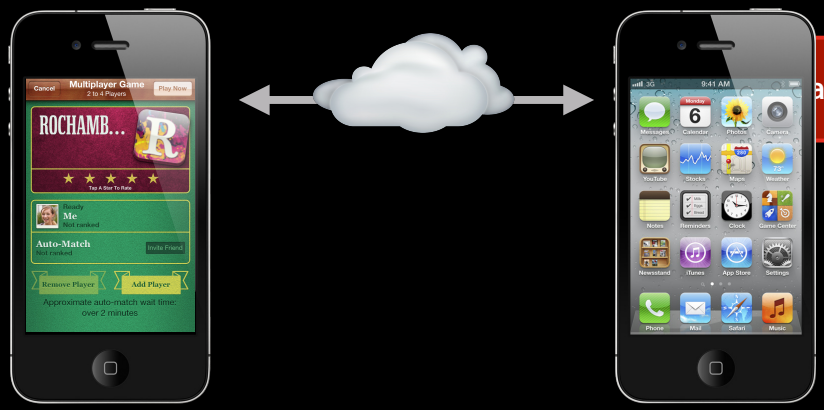

```
- (void)matchmakerViewController:(GKMatchmakerViewController *)viewController
didFailWithError:(NSError *)error
{
  // Handle error
}
```

Invitations

Handling invites

- Implement inviteHandler block
  - Called when user launches your game from Game Center app
    - Initialize GKMatchmakerViewController with match request and players
  - Called when recipient has accepted an invite
    - Initialize GKMatchmakerViewController with invite
    - May be called immediately if invite is already pending

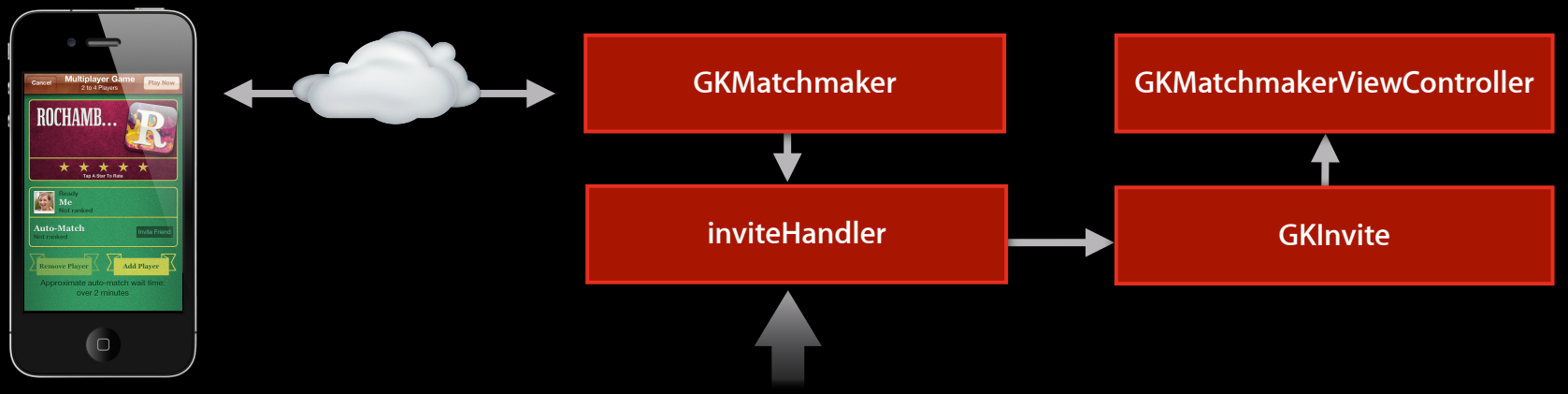
Classes



atchmaker

GKMatchmakerViewController

Invite Notifications




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

Programmatic Auto-match

Quick way to play

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
  }
}];
```

Auto-match

Match request redux

- Set minimum players
- Set maximum players
- Assign player group
- Assign player attributes

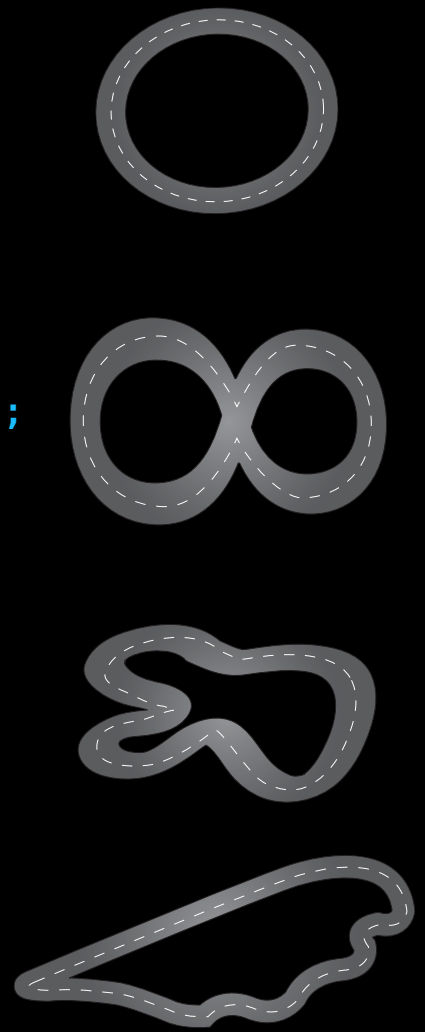
Player Groups

Pick a track…

- Match players based on game defined groups
  - Game level/map

```
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)
  - Location (North America, Europe, Asia)
- 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)
  - Band (guitar, bass, drums, vocals)
  - 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!

Hosting Your Own Server

Hosting Your Own Server

- Use UI or programmatic auto-match
- Use playerID to track players
- Communicate matched players to server
- Implement your own networking

Hosting Your Own Server

We will hook you up!

Hosting Your Own Server

Auto-matching API

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

GKMatchmaker *matchmaker = [GKMatchmaker sharedMatchmaker];

[matchmaker findPlayersForHostedMatchRequest:matchRequest
            withCompletionHandler:^(NSArray *playerIDs, NSError *error) {
  if (playerIDs.count > 0) {
     // Connect to the server and pass along player
  } else if (error) {
     // Handle error
  }
}];
```

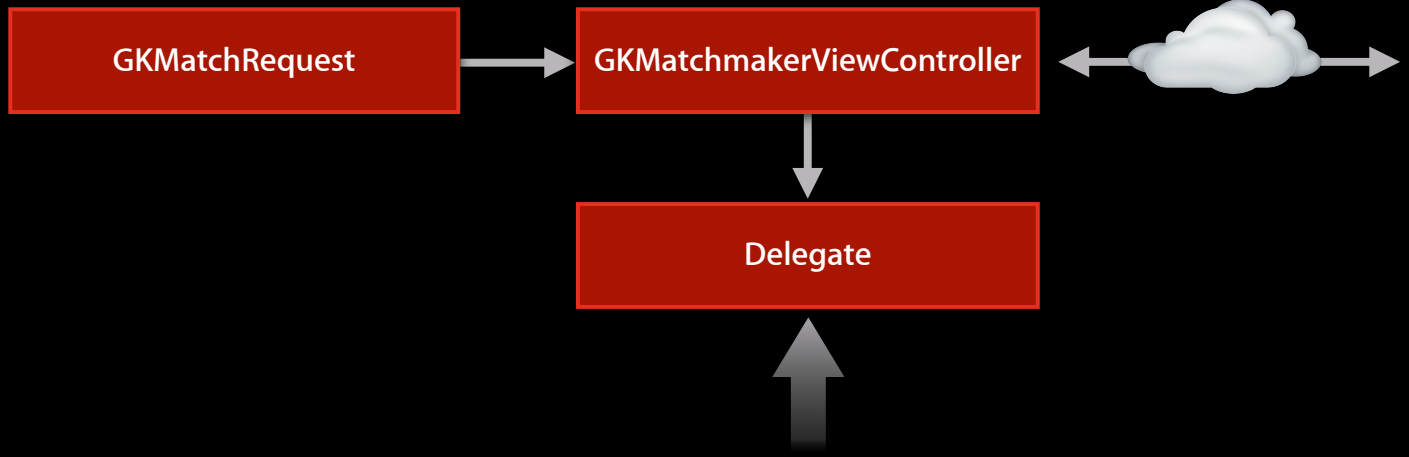
Hosting Your Own Server

MatchMaker UI—inviter side

```
GKMatchmakerViewController *viewController = [[GKMatchmakerViewController
alloc] initWithMatchRequest: matchRequest];

viewController.hosted = YES;
viewController.matchmakerDelegate = self;

[self.viewController presentViewController:viewController animated:YES
            completion:nil];


[viewController release];
```

Hosting Your Own Server

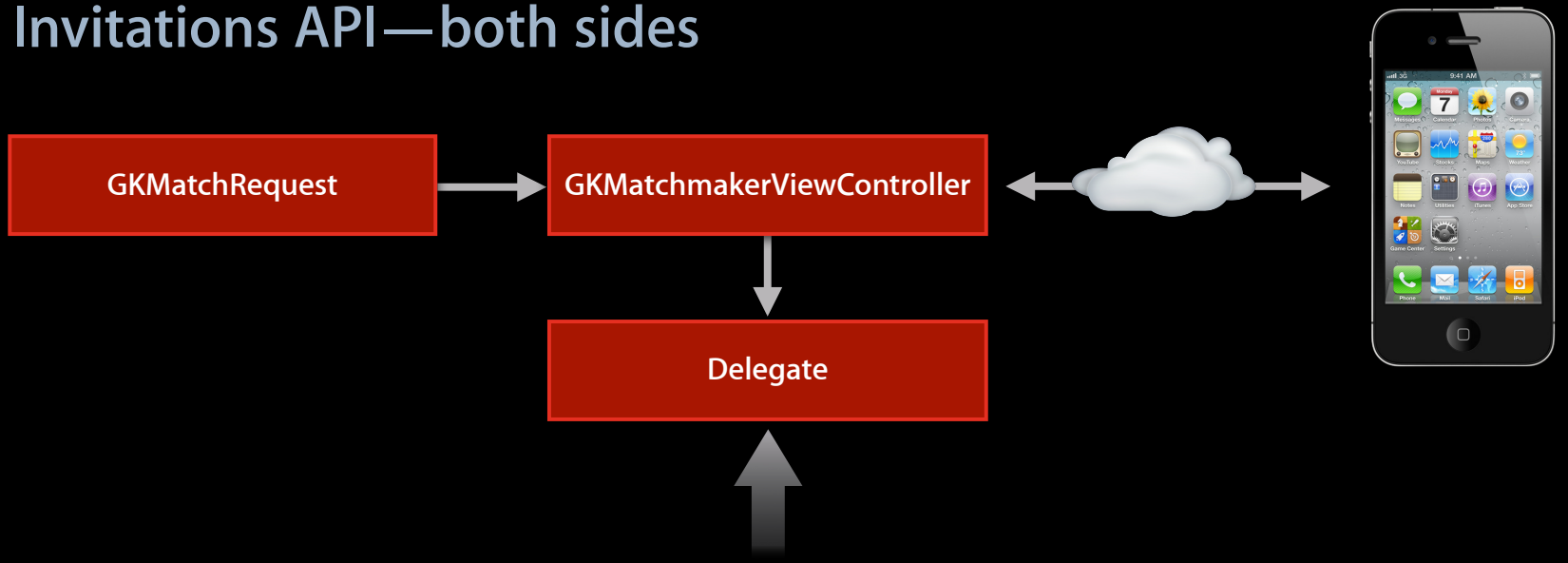
Invitations API—inviter side



```
- (void)matchmakerViewController:(GKMatchmakerViewController *)
viewController didReceiveAcceptFromHostedPlayer:(NSString *)playerID
{
  // talk to your server and make sure they are signed in

  [viewController setHostedPlayer:playerID connected:YES]; // update the UI
}
```

Hosting Your Own Server

Invitations API—both sides




```
- (void)matchmakerViewController:(GKMatchmakerViewController *)
  viewController didFindPlayers:(NSArray *)playerIDs
{
  [self.viewController dismissViewControllerAnimated: YES completion:^{
    // Start match
  }];
}
```

Peer-to-Peer Networking

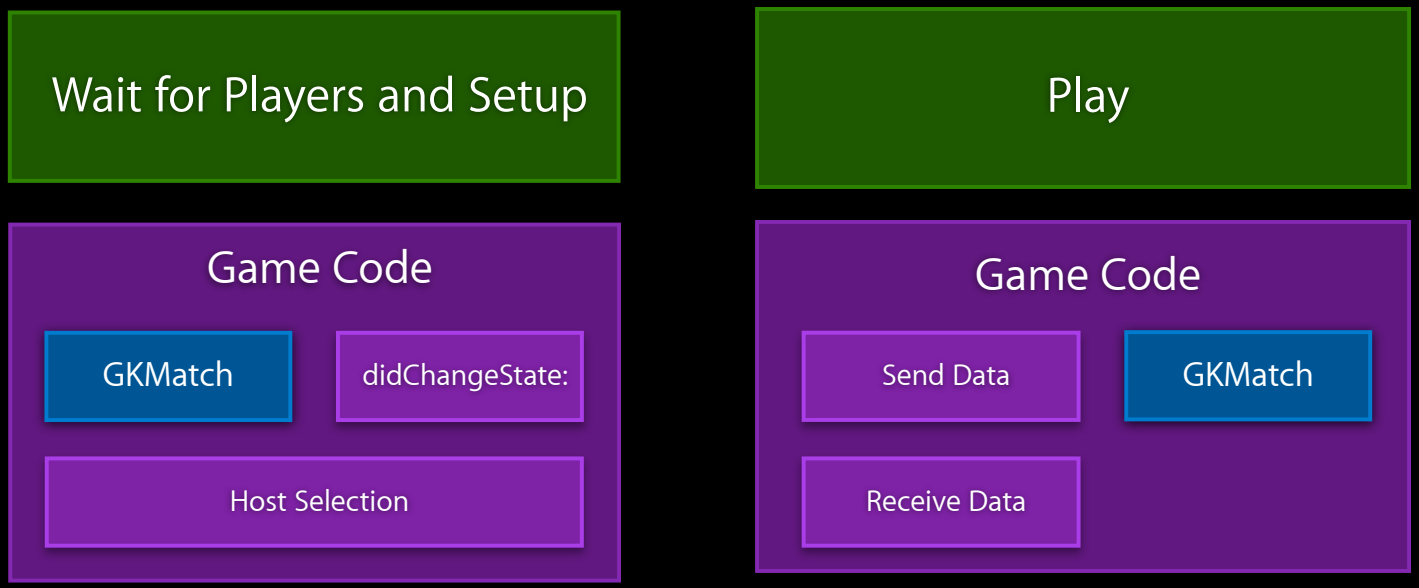
Peer-to-Peer Networking

Overview

- Game communications among players
  - Send data
  - Receive data
  - Option for reliable or unreliable communications
- Player state changes
  - Wait for all players to connect
  - Handle disconnection mid-game
  - Add players to existing game
- Use host selection to minimize network overhead

Peer-to-Peer Multiplayer

Communicating with your peers



Peer-to-Peer Network

Sending data

- Keep data sizes as small as possible
- Minimize update frequency
- Choice of communication styles
  - Reliable vs. unreliable

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

Peer-to-Peer Network

Sending data

```
NSArray *playerIDs = [NSArray arrayWithObject:destPlayerID];

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

Peer-to-Peer Network

Receiving data

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

Peer-to-Peer Networking

Waiting for players to connect

- Check expectedPlayers
  - Number of players you are waiting on

```
- (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
  }
}
```

Peer-to-Peer Networking

Offline considerations

- Players can come and go during game play
  - Take phone calls
  - Lose and regain connection
  - Switch game to background
- Important for game play to continue for others

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
  - Create a match request
  - Create matchMakerViewController
  - Call addPlayersToMatch:

```
GKMatchRequest *matchRequest = [[GKMatchRequest alloc] init];
matchRequest.minPlayers = 2;
matchRequest.maxPlayers = 4;
GKMatchmakerViewController  *controller = [[GKMatchmakerViewController
alloc] initWithMatchRequest: matchRequest];
controller.delegate = self;
[controller addPlayersToMatch:self.currentMatch];
```

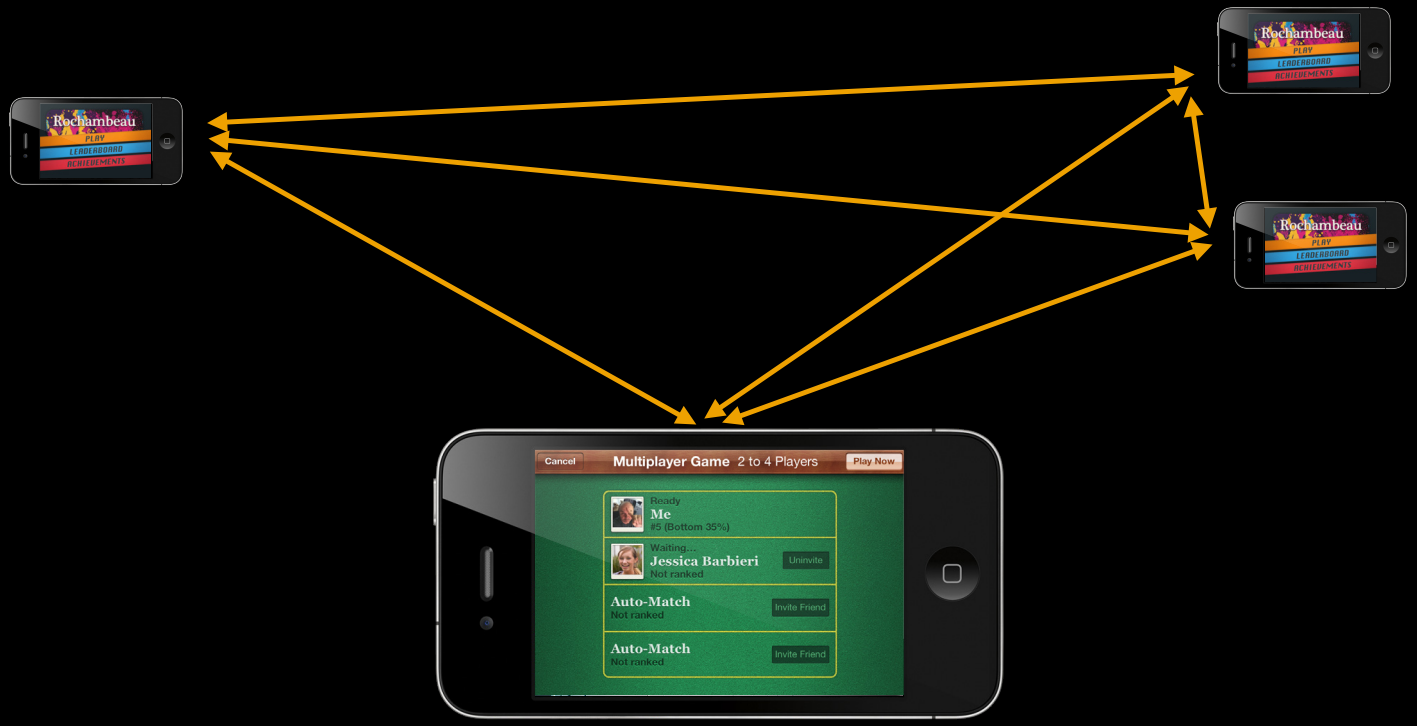
Peer-to-Peer Best Practices

Being a good network citizen

- Keep network traffic to minimum
  - Minimize size of data packets
  - Send data only when necessary
- Use common network strategies
  - Pick a peer to act as the host

Peer-to-Peer Best Practices

Full mesh



Peer-to-Peer Best Practices

Peer-hosted

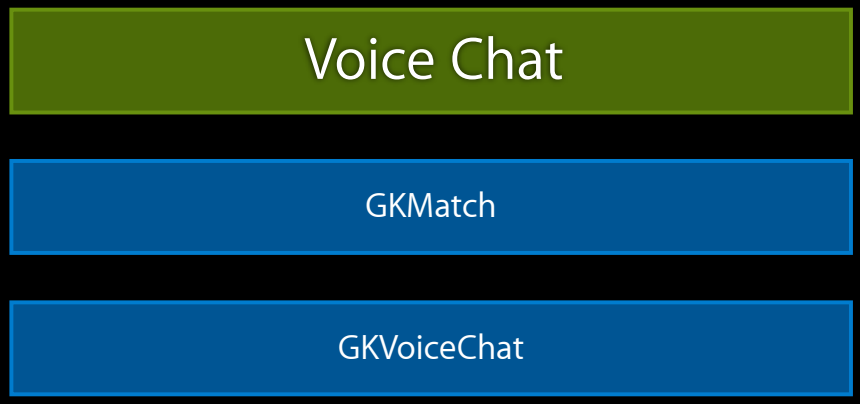
Peer-to-Peer Networking

Summary

- GKMatch provides API
- Handle player state changes
- Be a good network citizen
  - Keep network traffic to minimum
  - Consider hosting on your own servers

In-Game Voice Chat

In-Game Voice Chat



In-Game Voice Chat

- Allows players to talk with each other
- Keeps players involved
- Enhances competition
- Easy to integrate
- Networking handled for you

In-Game Voice Chat

Features

- Multiple named chats
- Hear audio from selected chats
- Microphone is routed to single chat
- Adjust the volume of a chat
- Mute player in a chat
- Player state feedback via playerStateUpdateHandler

In-Game Voice Chat

Pre-setup

- Set audio session to play and record
- Make audio session active

```
AVAudioSession *audioSession = [AVAudioSession sharedInstance];
[audioSession setCategory:AVAudioSessionCategoryPlayAndRecord error:&error];
[audioSession setActive:YES error:&error];
```

In-Game Voice Chat

Usage

```
// Get separate channels for the game and team
GKVoiceChat *mainChat = [self.match voiceChatWithName:@"main"];
GKVoiceChat *teamChat = [self.match voiceChatWithName:@"redTeam"];
```

```
// Stop main chat
[mainChat stop];
// Start team chat
[teamChat start];
```

```
// Make the team chat active to route microphone
teamChat.active = YES;

// Provide audio and visual indicator that the microphone is active
[self indicateMicrophoneActive];
```

In-Game Voice Chat

Handling player state changes

```
teamChat.playerStateUpdateHandler = ^(NSString *playerID,
            GKVoiceChatPlayerState state) {
  switch (state) {
    case GKVoiceChatPlayerConnected: { ... } break;
      // Indicate that the player has connected
    case GKVoiceChatPlayerDisconnected: { ... } break;
      // Indicate that the player has disconnected
    case GKVoiceChatPlayerSpeaking: { ... } break;
      // Indicate that the player has started speaking
    case GKVoiceChatPlayerSilent: { ... } break;
      // Indicate that the player has gone silent
    default: { ... } break;
      // Indicate the the player has stopped speaking
  }
};
```

Multiplayer Setup

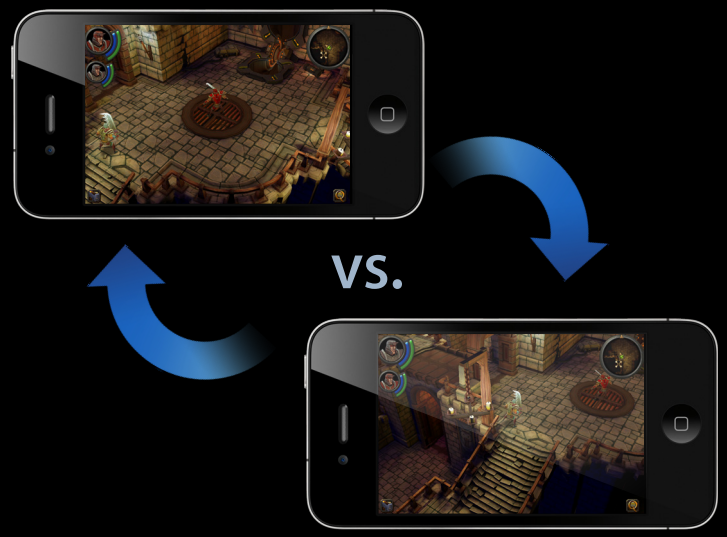
Things to consider

- Version compatibility
  - Set up in iTunes Connect
  - Invitee's device compares version to inviter's
- Upgrades offered if necessary, but only to current version

Notes on Testing

- Need to test on devices
  - Multiple devices
  - Multiple accounts
  - Multiple networks
  - Multiple carriers
- Testing on simulator limited
  - Invitations are not available
  - In-game voice chat is disabled
  - No push notifications

Multiplayer Summary



- Popular feature
- Adds longevity to your app
- Peer-to-peer/turn-based/hosted
- Basic flow
  - MatchMaker UI
  - Programmatic auto-match
  - Hosted server
- Peer-to-peer communications
- Setting up voice chat
- Game Center services
- Testing

More Information

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

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

Related Sessions

Essential Game Technologies for iOS, Part 1	Mission Tuesday 9:00 AM
Essential Game Technologies for iOS, Part 2	Mission Tuesday 10:15 AM
Introduction to Game Center	Mission Tuesday 4:30PM
Working with Game Center	Mission Wednesday 9:00AM
Turn-Based Gaming with Game Center	Mission Wednesday 11:30AM

Introduction to Game Center (Repeat)	Russian Hill Friday 9:00AM
Essential Game Technologies for iOS, Part 1 (Repeat)	Marina Friday 10:15AM
Essential Game Technologies for iOS, Part 2 (Repeat)	Marina Friday 11:30AM

Labs

Game Design for iOS Lab	Graphics & Media Lab A Tuesday 2:00PM-6:00PM
Game Center Lab	Graphics & Media Lab A Wednesday 2:00PM-6:00PM
Game Center Lab	Graphics & Media Lab A Friday 11:30AM-1:30PM


WWDC2011