Game Center Techniques, Part 2

Adding Multiplayer to Your Game

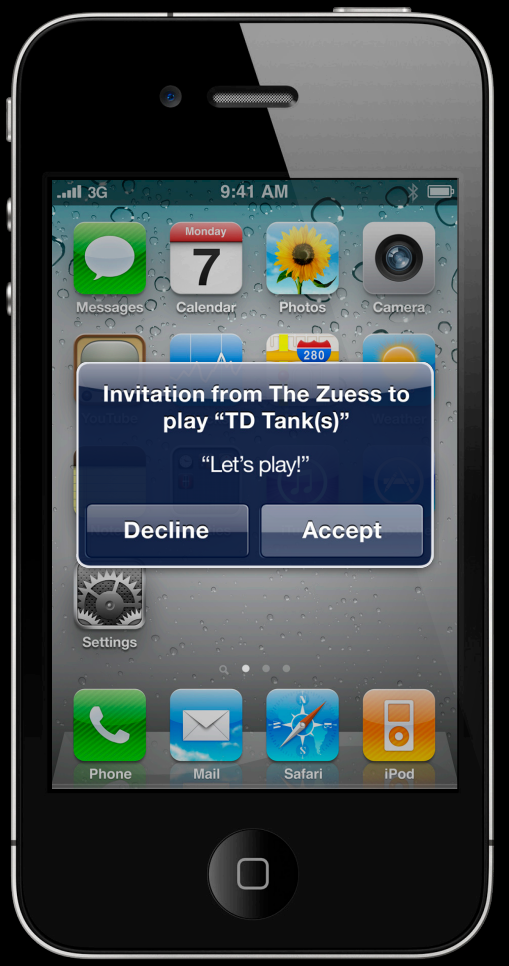
Nathan Taylor
iPhone Development Engineer

What Is Game Center?

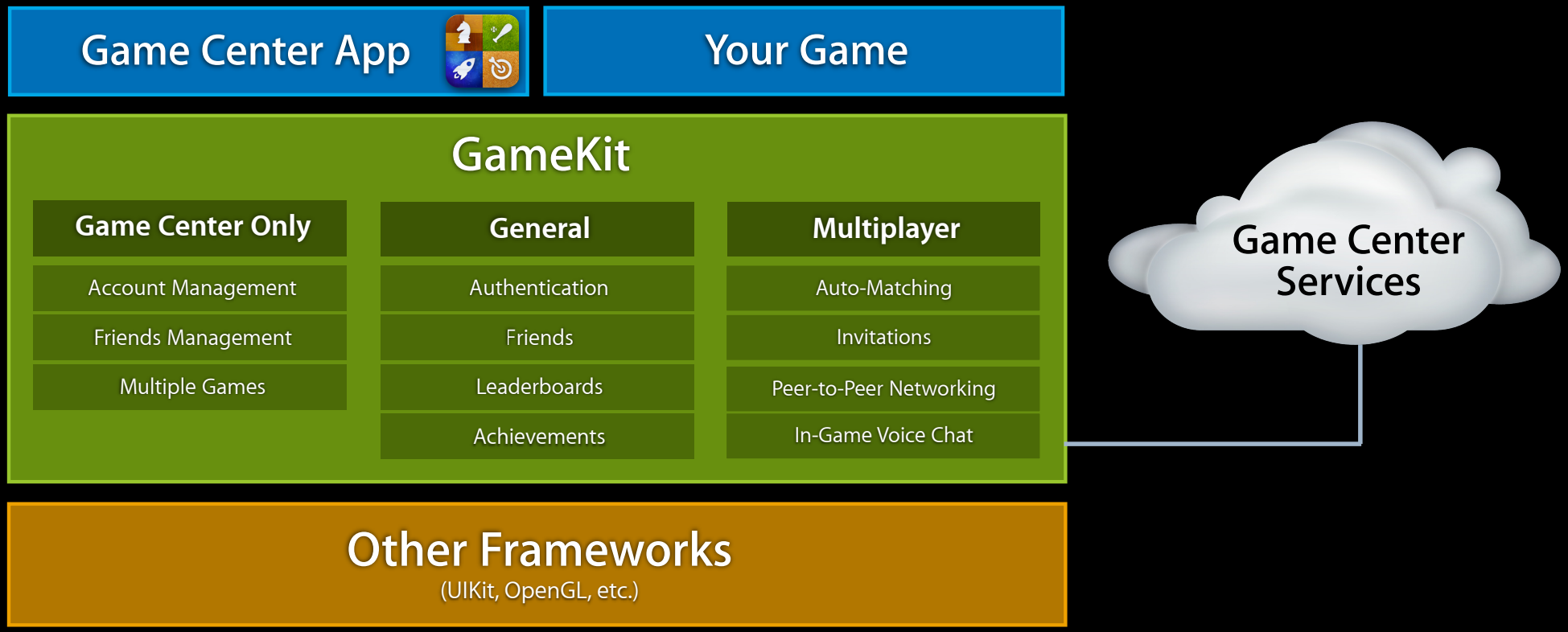
- Social gaming network
  - Built-in application
  - Framework
  - Online services
- Provides
  - Friend relationships
  - Leaderboards
  - Achievements
  - Multiplayer

What You'll Learn

- Brief overview of Game Center
- Quick look at Game Center Services
- In-depth discussion of multiplayer services
  - Authentication
  - Getting connected
  - Network communications
  - Player communications



Game Center Overview



Multiplayer

Auto-Matching

Invitations

Peer-to-Peer Networking

In-Game Voice Chat

Game Center Services

- Connect people
  - Route requests to devices
  - Establish global peer-to-peer connections
- Services available on WiFi and cellular
  - Great opportunity for social gaming
  - Lots of discovery through invites
  - People move around, connections come and go

Game Center Services

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

Game Center Services

Setup considerations

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

Getting Started

GKLocalPlayer

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

Getting Started

Authentication

- Authenticate as early as possible
- Other operations will return errors if not authenticated

```
GKLocalPlayer *localPlayer = [GKLocalPlayer localPlayer];

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

Multiplayer Services

- Auto-Matching
- Invitations
- Peer-to-Peer Networking
- In-Game Voice Chat

Multiplayer Services

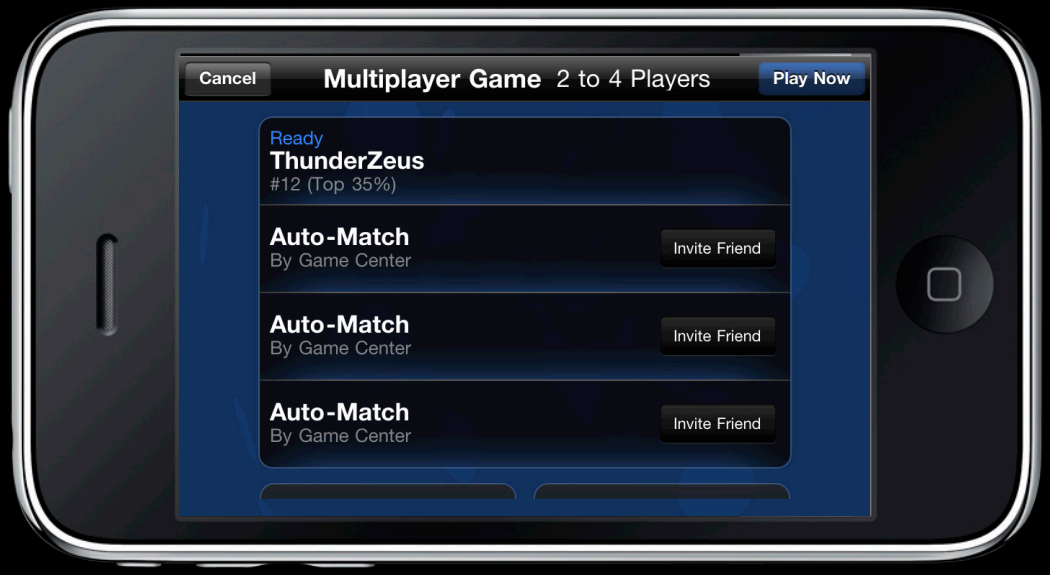
Auto-Matching

GKMatchRequest

GKMatchmaker

GKMatchmakerViewController

Auto-Matching



Auto-Matching

Process

Auto-Matching

Process

- Create a match request
- Send match request to server
- Server applies matchmaking logic
  - Player group
  - Player attributes
- Match returned
- Wait for players to connect
- Begin game!

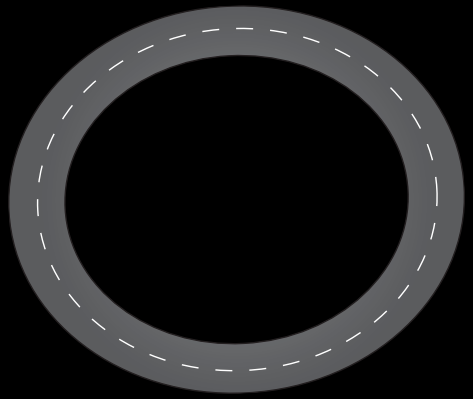
Auto-Matching

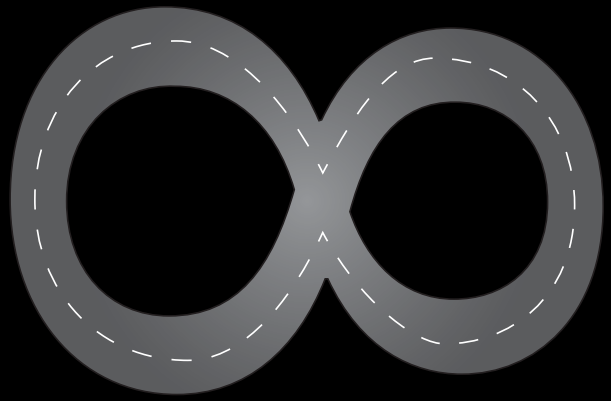
Match request

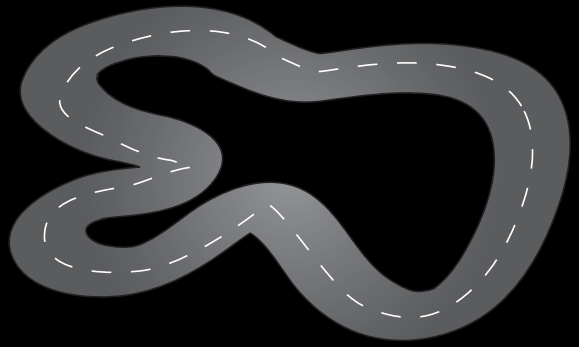
- Set minimum players
- Set maximum players
- Assign player group
- Assign player attributes

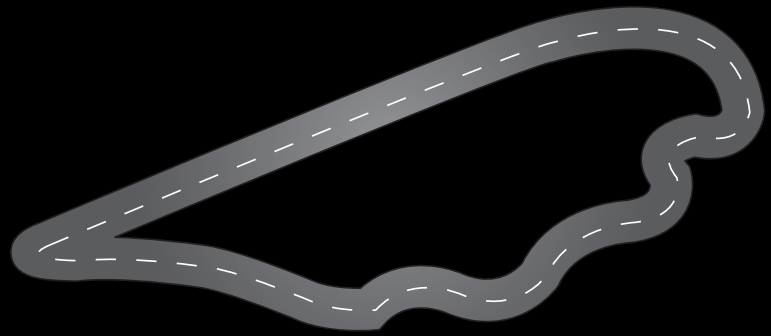
Auto-Matching

Player groups









Auto-Matching

Player groups

Auto-Matching

Player groups

- Arbitrary grouping based on in-game settings
- Used to match players with compatible in-game settings
- Ideas for player group assignment:
  - Difficulty setting (easy/normal/hard)
  - Game level or map
  - Game mode (capture the flag, deathmatch, etc.)
  - Region or realm
- Check activity for a player group to make sure there are others to match with

Auto-Matching

Querying a player group

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

[matchmaker queryPlayerGroupActivity:myPlayerGroup
                withCompletionHandler:^(NSInteger activity, NSError *error)
{
  if (!error) {
    // Indicate group activity to user

    if (activity < MY_THRESHOLD) {
      // Select a different player group
    }
  }
}];
```

Auto-Matching

Player attributes

Fighter
Mage
Cleric
Thief

Auto-Matching

Player attributes

Fighter
Mage
Cleric
Thief

Auto-Matching

Player attributes

- Optional 32-bit unsigned integer
- Logical OR operation
- Player group the AND operation
- Chosen based on player characteristics
  - Role-playing (fighter, cleric, mage, thief)
  - Band (guitar, bass, drums, vocals)
  - Sports (goalie, forward, defense)
- Match made with attributes combined to create 0xFFFFFFFF

Match Request

Example

- Four-player dungeon crawl game

```
GKMatchRequest *matchRequest = [[GKMatchRequest alloc] init];
matchRequest.minPlayers = 4;
matchRequest.maxPlayers = 4;
matchRequest.playerGroup = level4DungeonGroup;
```

- Fighter

```
matchRequest.playerAttributes = MY_FIGHTER; // 0xFF000000
```

- Mage

```
matchRequest.playerAttributes = MY_MAGE;    // 0x00FF0000
```

- Cleric

```
matchRequest.playerAttributes = MY_CLERIC;  // 0x0000FF00
```

- Thief

```
matchRequest.playerAttributes = MY_THIEF;   // 0x000000FF
```

Auto-Matching

Getting connected

- Peer-to-peer
  - Establishes communications directly between players
  - Send and receive data through API
- Server hosted
  - Separate server to host game
  - Player count less restricted
  - Custom network communications

Auto-Matching

Peer-to-peer or hosted

Auto-Matching

Auto-Matching

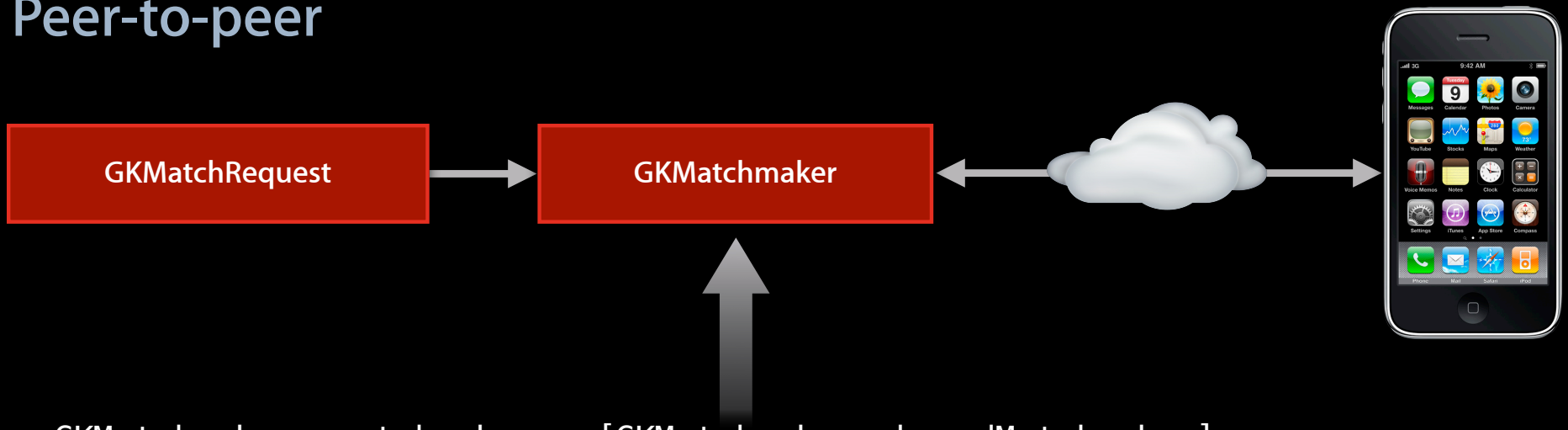
Auto-Matching

Match request


```
GKMatchRequest *matchRequest = [[GKMatchRequest alloc] init];
matchRequest.minPlayers = 2;
matchRequest.maxPlayers = 4;
matchRequest.playerGroup = level4Dungeon;
matchRequest.playerAttributes = MY_FIGHTER;
```

Auto-Matching

Peer-to-peer

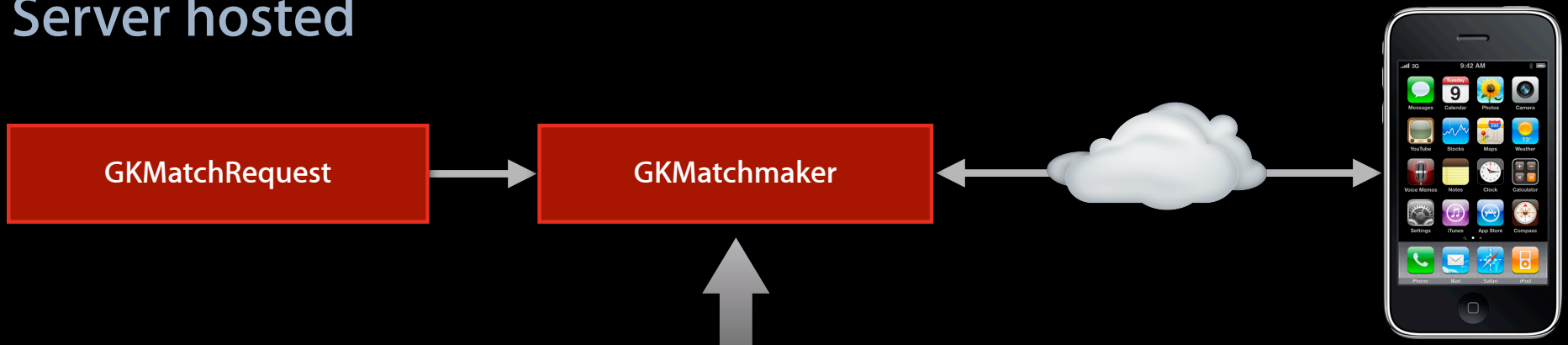



```
GKMatchmaker *matchmaker = [GKMatchmaker sharedMatchmaker];

[matchmaker createMatchForRequest:matchRequest
            withCompletionHandler:^(GKMatch *match, NSError *error) {
  if (error) {
    // Handle error
  }
  else {
    match.delegate = self;
  }
}];
```

Auto-Matching

Server hosted




```
[matchmaker findPlayersForRequest:matchRequest
            withCompletionHandler:^(NSArray *players, NSError *error) {
  if (error) {
    // Handle error
  }
  else {
    // Connect to the server and pass along player
  }
}];
```

Auto-Matching

Summary

- Create a match request
  - Assign player group
  - Assign player attributes
- Request match
- Handle player state changes
- Wait for players to connect

Multiplayer Services

Auto-Matching

Invitations

Peer-to-Peer Networking

In-Game Voice Chat

Multiplayer Services

Invitations

GKMatchRequest

GKMatchmaker

GKInvite

GKMatchmakerViewController

Invitations

Inviter

Invitee

Invitations

Inviter

Invitee

Invitations

Inviter

Invitee

Invitations

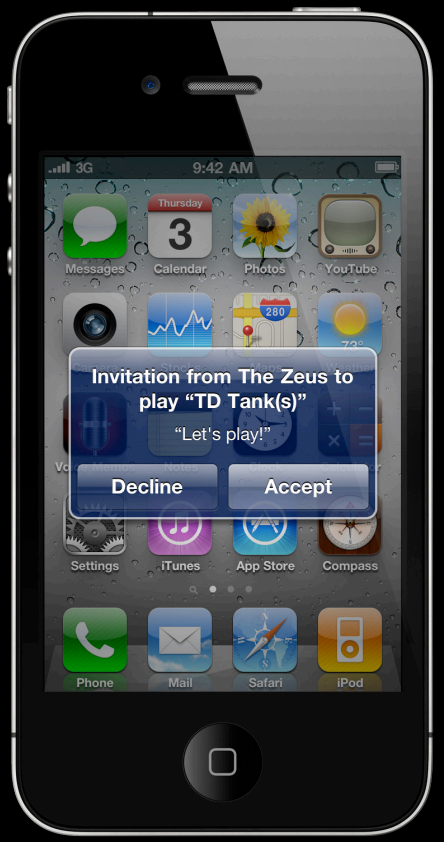
Inviter

Invitee

Invitations

Inviter

Invitee



Invitations

Inviter

Invitee

Invitations

Inviter

Invitee

Invitations

Inviter

Invitee

Invitations

- Invite friends to play game
  - Standard UI
  - Directly from Game Center
- Push notification sent to friend's device
  - Accept
  - Decline
  - Buy game
- Game launched

Invitations

Inviting friends

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

Invitations

Invitations

Inviting friends

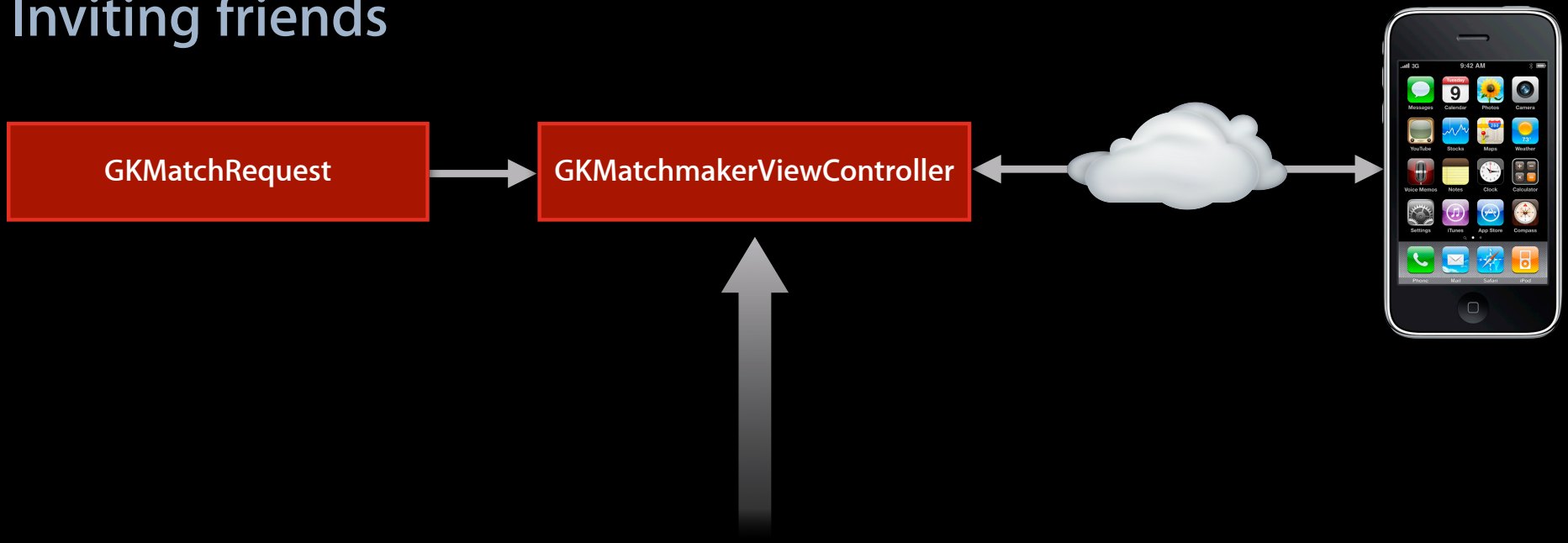
Invitations

Inviting friends


```
GKMatchRequest *matchRequest = [[GKMatchRequest alloc] init];
matchRequest.minPlayers = 2;
matchRequest.maxPlayers = 4;
matchRequest.playerGroup = level4Dungeon;
```

Invitations

Inviting friends



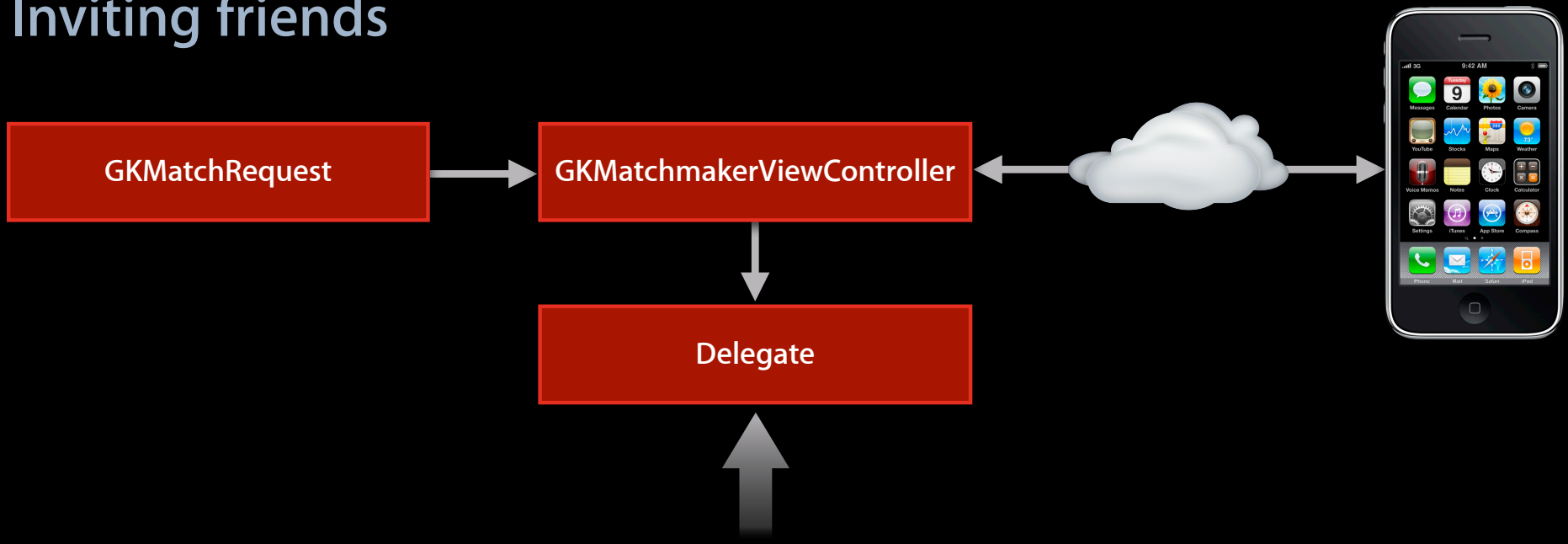

```
GKMatchmakerViewController *controller = [[GKMatchmakerViewController
alloc] initWithMatchRequest:matchRequest];

controller.delegate = self;

[controller show];
```

Invitations

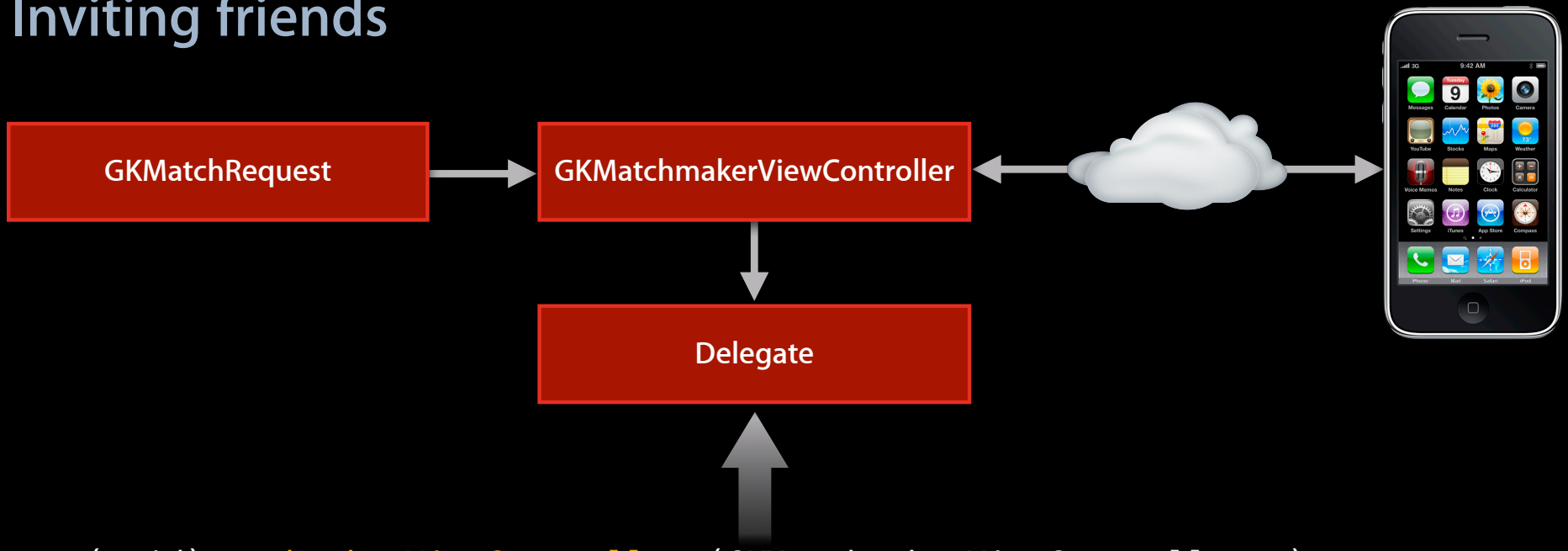
Inviting friends




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

Invitations

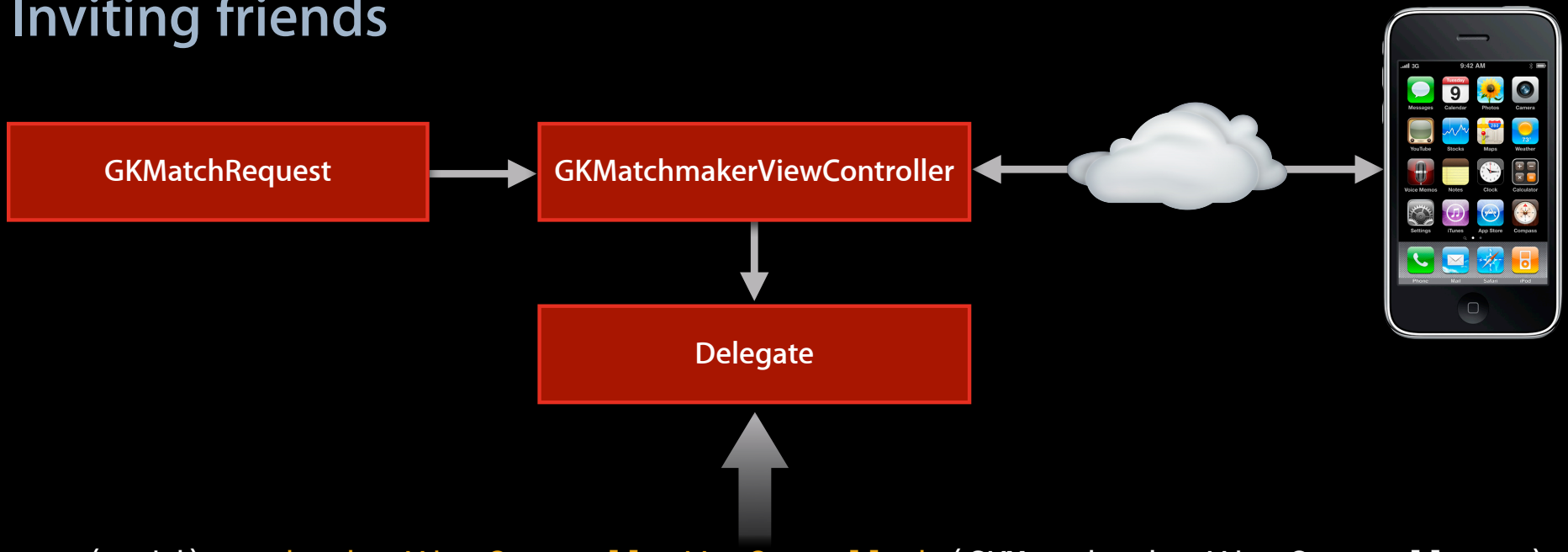
Inviting friends




```
- (void)matchmakerViewController:(GKMatchmakerViewController *)
viewController didFindPlayers:(NSArray *)players
{
  // Start communicating with server for hosted game
}
```

Invitations

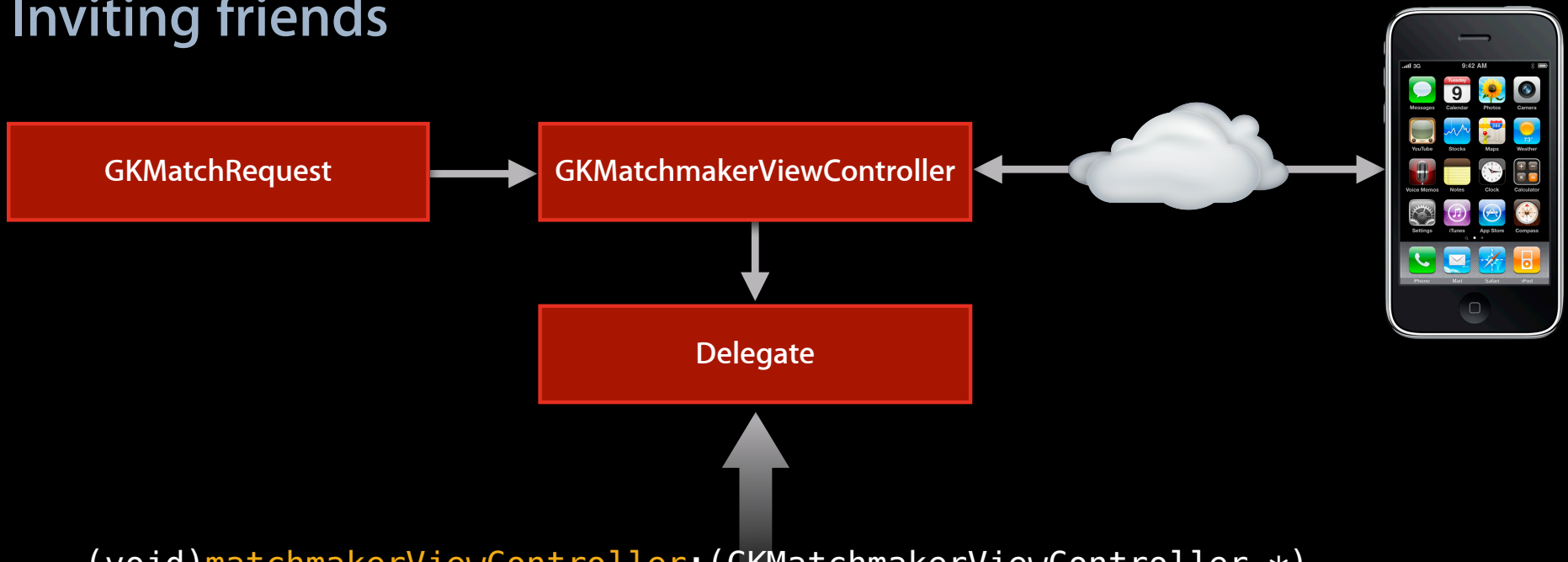
Inviting friends




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

Invitations

Inviting friends




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

Invitations

Handling invites

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

Invitations

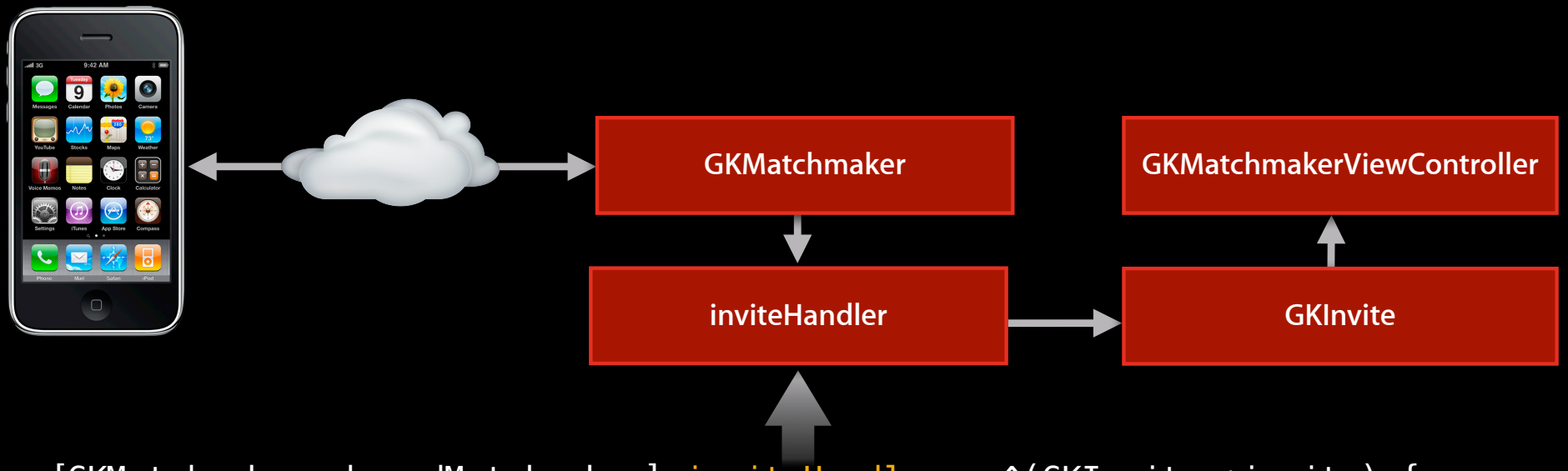
Handling invites

GKMatchmaker

GKMatchmakerViewController

Invitations

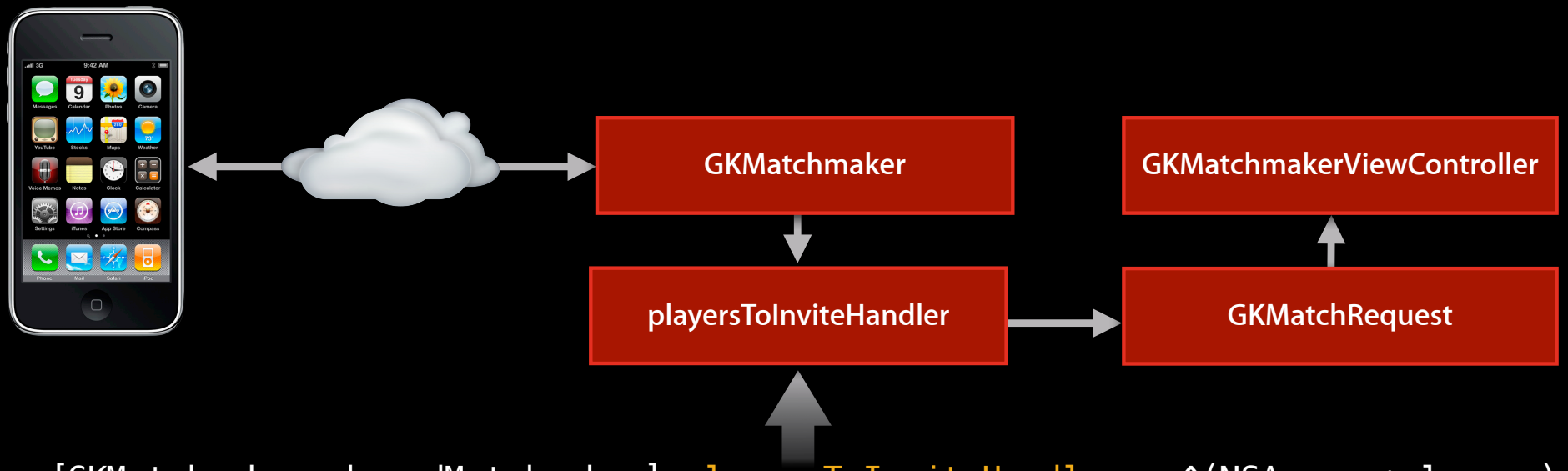
Handling invites




```
[GKMatchmaker sharedMatchmaker].inviteHandler = ^(GKInvite *invite) {
  GKMatchmakerViewController *controller = [[GKMatchmakerViewController alloc]
  initWithInvite:invite];
  controller.delegate = self;
  [controller show];
  [controller autorelease];
};
```

Invitations

Handling invites




```
[GKMatchmaker sharedMatchmaker].playersToInviteHandler = ^(NSArray *players) {
  GKMatchmakerViewController *controller = [[GKMatchmakerViewController alloc]
  initWithMatchRequest:self.matchRequest playersToInvite:players];
  controller.delegate = self;
  [controller show];
  [controller autorelease];
};
```

Invitations

Summary

- Create match request
- Present standard UI
- Handle invites
  - Called any time
  - May be called immediately

Multiplayer Services

Auto-Matching

Invitations

Peer-to-Peer Networking

In-Game Voice Chat

Multiplayer Services

Peer-to-Peer Networking

GKMatch

Peer-to-Peer Networking

- Game communications between players
  - Send data
    - Unreliable
    - Reliable
  - Receive data
- Player state changes
  - Wait for all players to connect
  - Handle disconnection mid-game

Peer-to-Peer Networking

Waiting for players to connect

```
- (void)match:(GKMatch *)match player:(GKPlayer *)player didChangeState:
(GKPlayerConnectionState)state
{
  // Handle connection state changes (eg. show connected players)
  switch (state) {
    case GKPlayerStateConnected:
      // Show that the player has connected
      break;
    case GKPlayerStateDisconnected:
      // Handle player disconnection
      break;
    default:
      break;
  }

  if (!self.gameStarted && match.expectedPlayers == 0) {
    // Begin game once all players are connected
  }
```

Peer-to-Peer Network

Sending data

```
NSArray *players = [NSArray arrayWithObject:destPlayer];

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

Peer-to-Peer Network

Sending data

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

Peer-to-Peer Network

Receiving data

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

Peer-to-Peer Network

Being a good network citizen

- Keep network traffic to minimum
  - Minimize size of data packets
  - Don’t send data for every frame
  - Don’t broadcast all data to all players
- Use common network strategies
  - Set up a client-hosted network
  - Set up a ring network
  - User a server-based network

Peer-to-Peer Networking

Client-hosted

- Nominate a host
  - Vote/coin-toss alogrithm
  - Compare playerID
- Send data to host
- Host maintains truth
- Host passes data to other clients

Auto-Matching

Full peer-to-peer mesh

Auto-Matching

Client-host topology

Auto-Matching

Ring topology

Hosting Your Own Server

- Choose correct API
- Use invitations and auto-matching
- Use playerID to track players
- Communicate matched players to server
- Implement your own networking

Hosting Your Own Server

Auto-Matching API

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

[matchmaker findPlayersForRequest:myMatchRequest
            withCompletionHandler:^(NSArray *players, NSError *error) {
  if (error) {
    // Handle error
  }
  else {
    // Connect to the server and pass along player
  }
}];
```

Hosting Your Own Server

Invitations API

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

viewController.hosted = YES;
viewController.delegate = self;

[viewController show];
[viewController release];


- (void)matchmakerViewController:(GKMatchmakerViewController *)
viewController didFindPlayers:(NSArray *)players
{
  // Start communicating with server for hosted game
}
```

Peer-to-Peer Networking

Summary

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

Multiplayer Services

Auto-Matching

Invitations

Peer-to-Peer Networking

In-Game Voice Chat

Multiplayer Services

In-Game Voice Chat

GKMatch

GKVoiceChat

In-Game Voice Chat

- Allows players to communicate 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 = ^(GKPlayer *player,
GKVoiceChatPlayerState state) {
  switch (state) {
    case GKVoiceChatPlayerConnected:
      // Indicate that the player has connected
      break;
    case GKVoiceChatPlayerDisconnected:
      // Indicate that the player has disconnected
      break;
    case GKVoiceChatPlayerSpeaking:
      // Indicate that the player has started speaking
      break;
    case GKVoiceChatPlayerSilent:
    default:
      // Indicate the the player has stopped speaking
      break;
  }
};
```

Notes on Testing

- Need to test on devices
  - Multiple devices
  - Multiple accounts
- Testing on simulator limited
  - Invitations are not available
  - In-game voice chat is disabled
  - No preemptive cache invalidation

Summary

- Authenticate the local player
- Define player group and attributes for match request
- Implement inviteHandler and playersToInviteHandler
- Use GKMatchmakerViewController
- Wait for all players to connect
- Integrate voice chat
- Preview now, available later this year

More Information

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

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

Related Sessions

Introduction to Game Center	Pacific Heights Tuesday 2:00PM
Game Center Techniques Part 1	Pacific Heights Tuesday 3:15PM
Game Design and Development for iPhone OS Part 1 (repeat)	Presidio Friday 9:00AM
Game Design and Development for iPhone OS Part 2 (repeat)	Presidio Friday 10:15AM

Labs

Game Center Lab	Graphics & Media Lab B Wednesday 2:00PM
Game Center Lab #2	Graphics & Media Lab D Friday 12:30PM
Game Design for iPhone OS Lab	Graphics & Media Lab A Wednesday 2:00PM
Game Design for iPhone OS Lab #2	Graphics & Media Lab A Friday 11:30AM


WWDC10