Integrating with Game Center

Bring your game to a higher level

Session 516

Dan Kurtz
iOS Engineer

Game Center

Three pillars

Game Center

Three pillars

Players

Game Center

Three pillars

Players

Scores

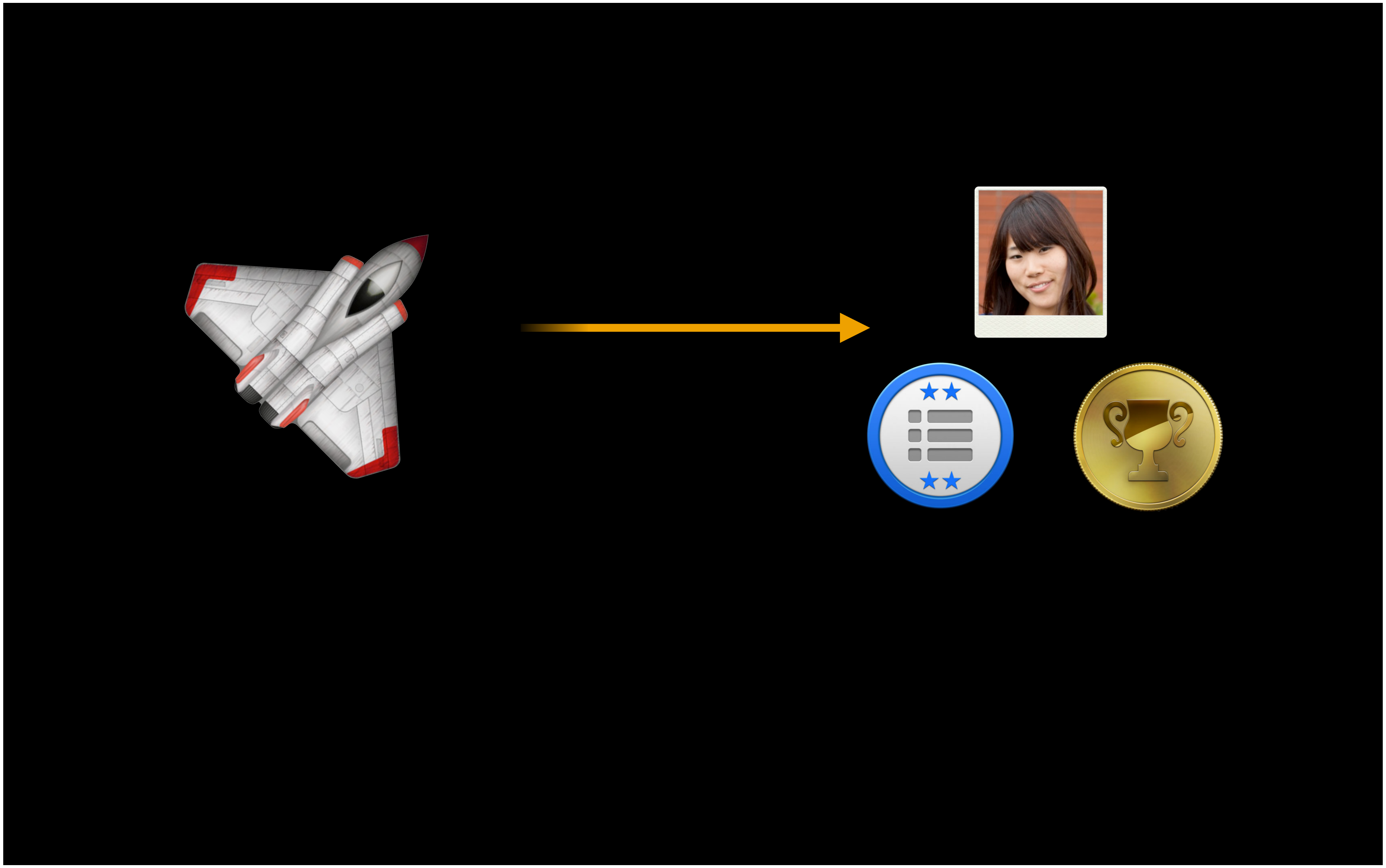
Game Center

Three pillars

Players

Scores

Achievements



Flip the script

From Here to There

- In-depth with Game Center pillars
- Use brand new features
- Give ideas for new possibilities

Players

[redacted]

Eunice

Eunice

Eunice

“Goned0”

Eunice

“Goned0”

Eunice

"Goned0"

Eunice

"Goned0"

`GKLocalPlayer`

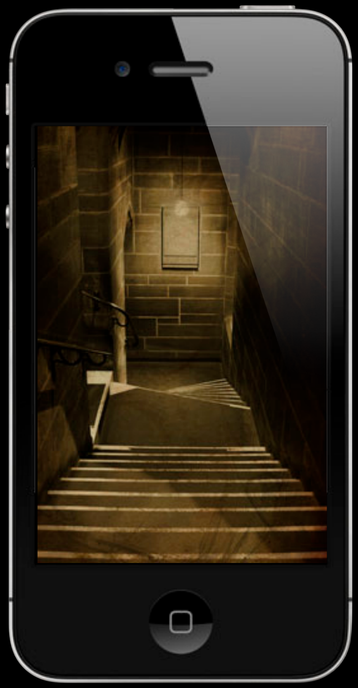
Eunice

"Goned0"

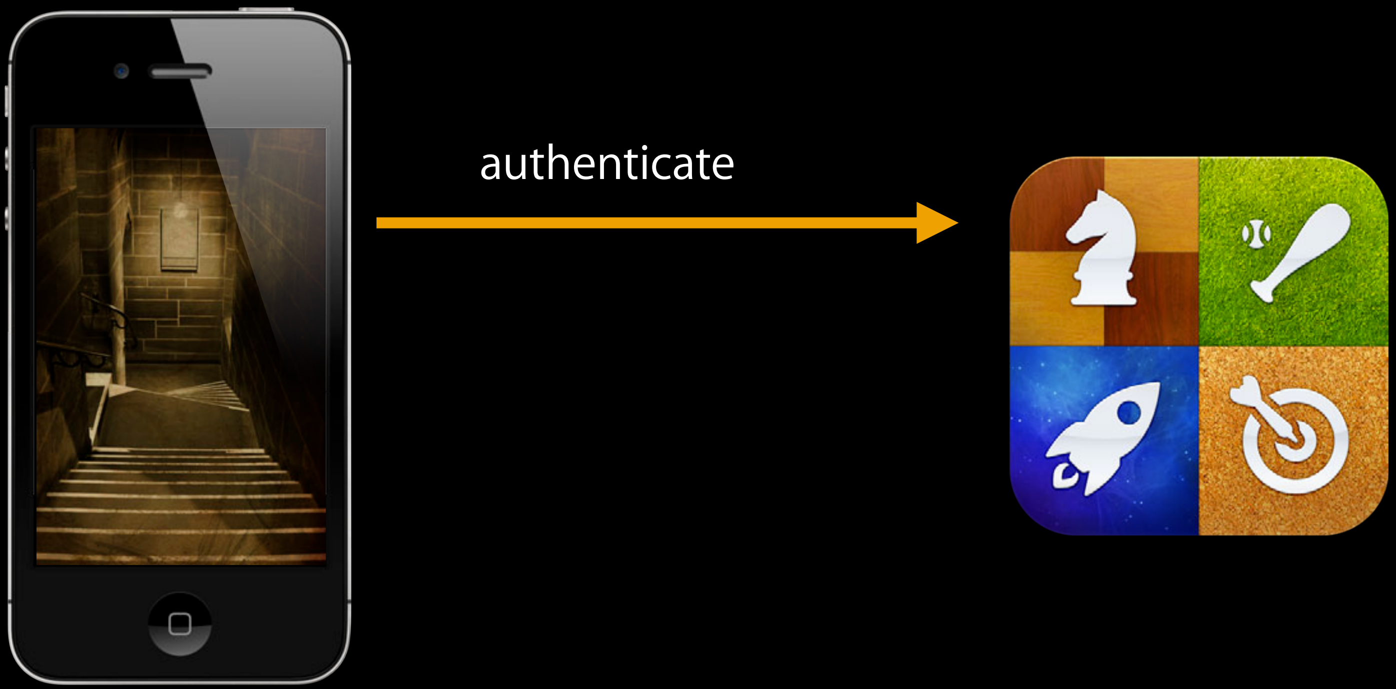
GKLocalPlayer

???

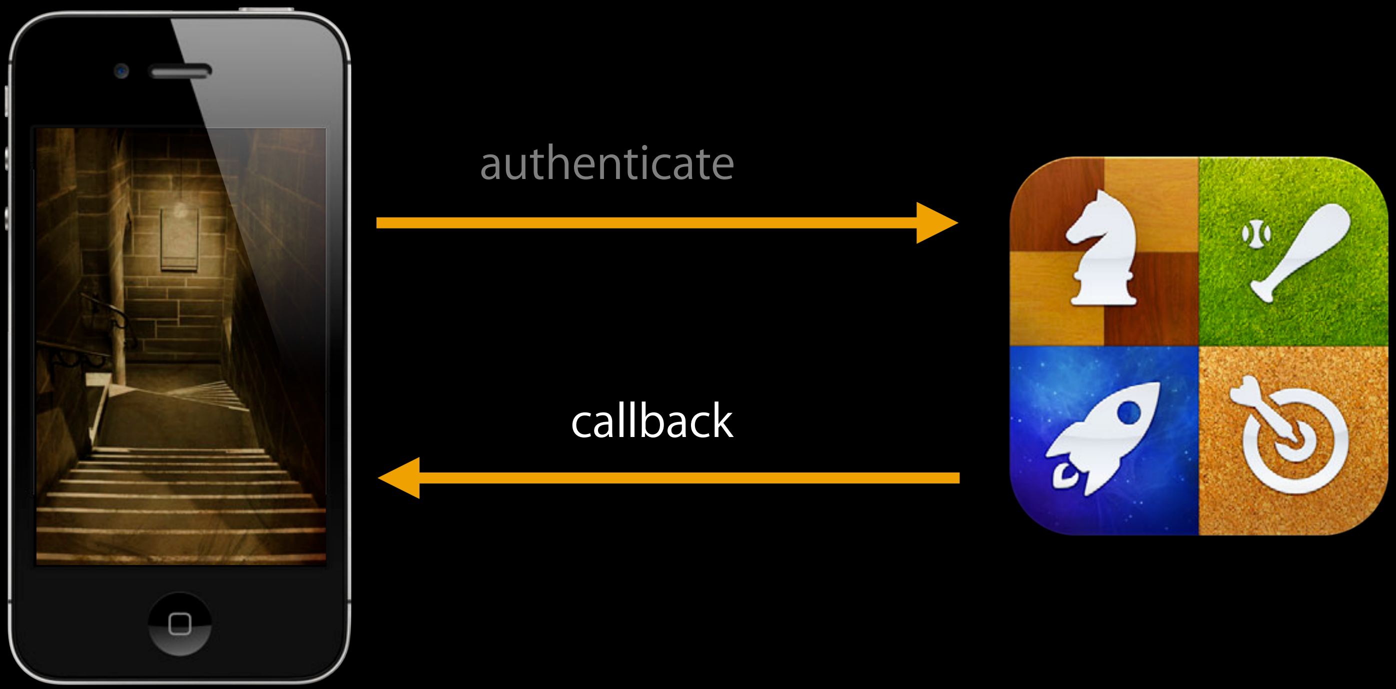
Authentication



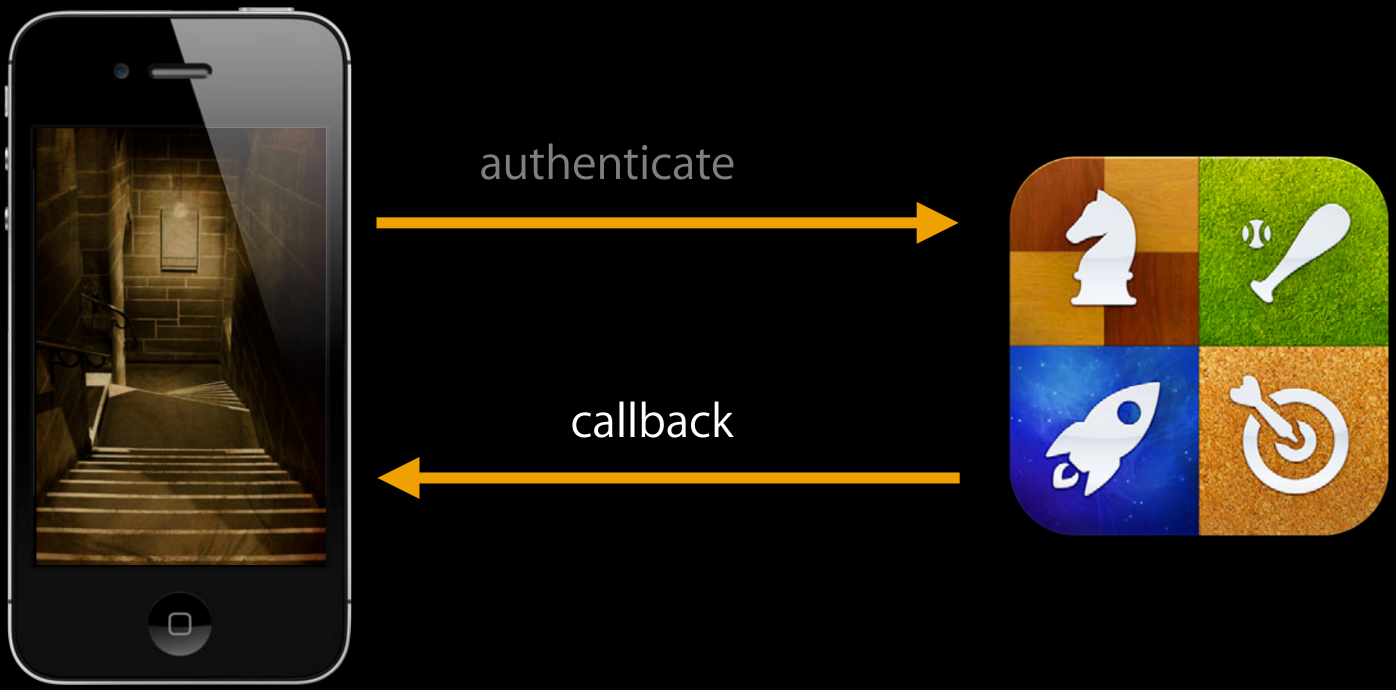
Authentication



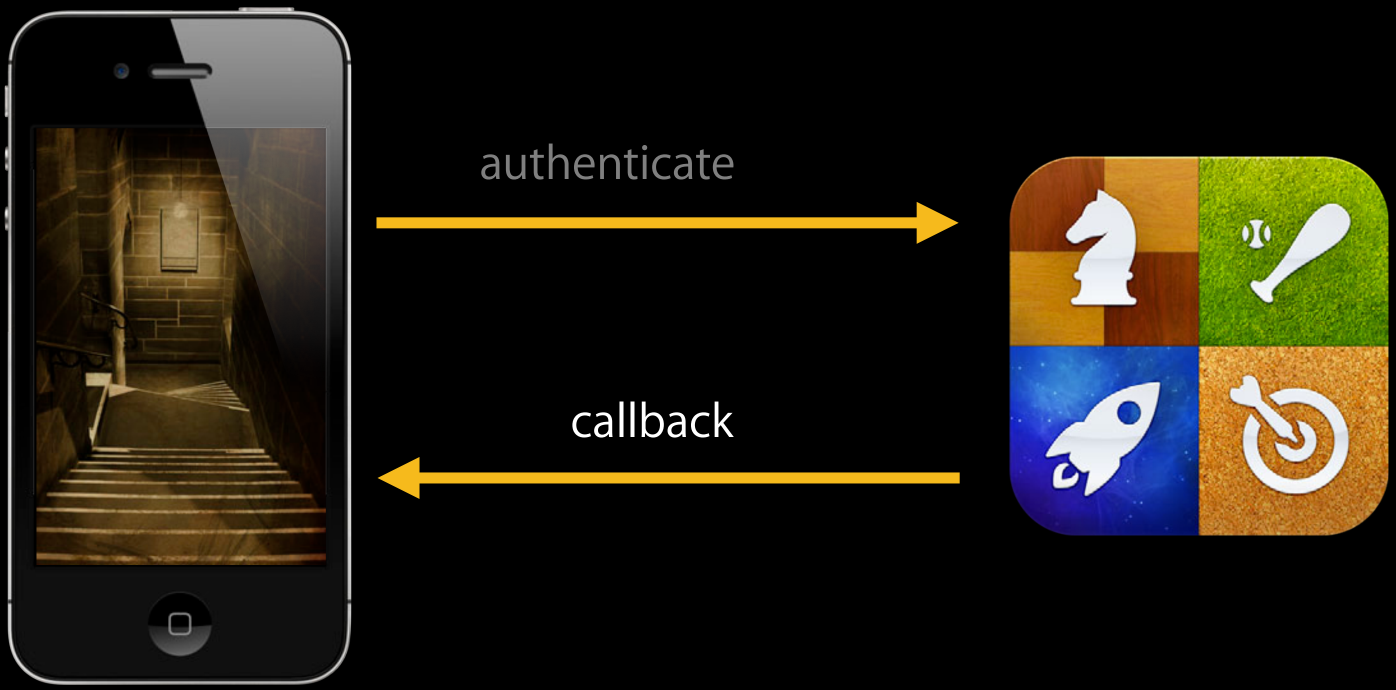
Authentication



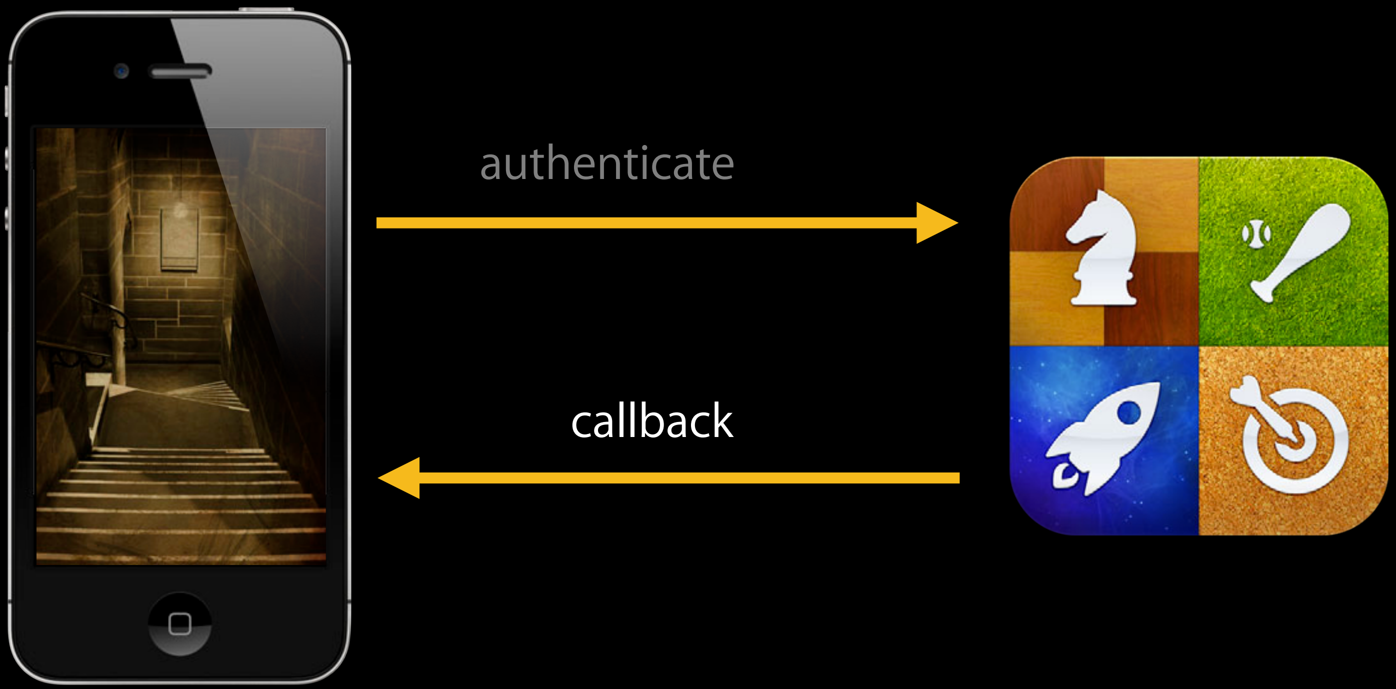
Authentication



Authentication

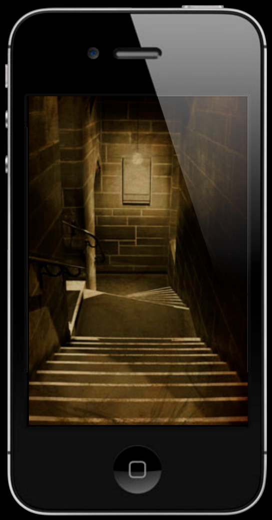


Authentication



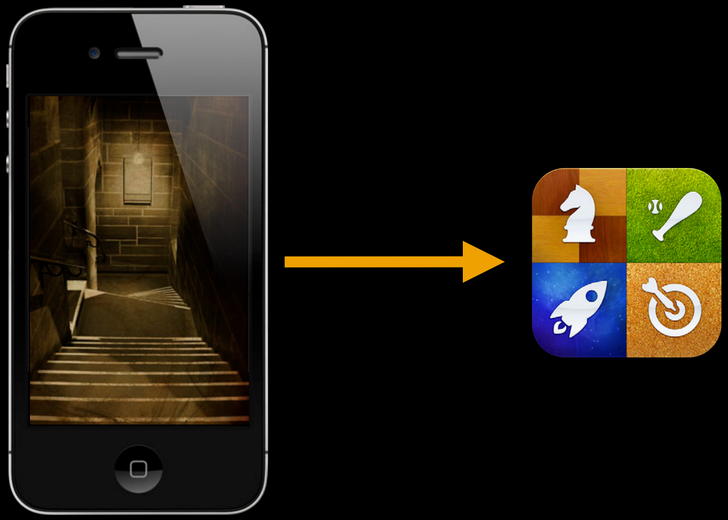
When We Call You Back

- Upon `authenticate`



When We Call You Back

- Upon `authenticate`



When We Call You Back

- Upon `authenticate`
- Coming back to foreground

When We Call You Back

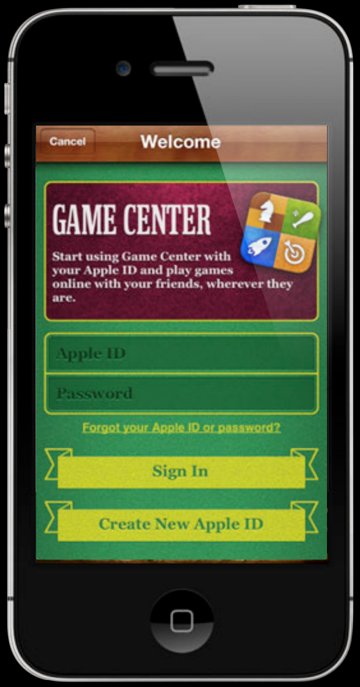
- Upon `authenticate`
- Coming back to foreground

When We Call You Back

- Upon `authenticate`
- Coming back to foreground
- Upon Sign In

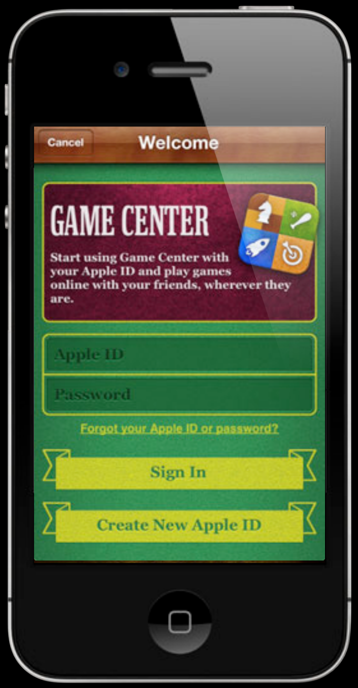
When We Call You Back

- Upon `authenticate`
- Coming back to foreground
- Upon Sign In


Cancel
Welcome
GAME CENTER
Start using Game Center with your Apple ID and play games online with your friends, wherever they are.
Apple ID
Password
Forgot your Apple ID or password?
Sign In
Create New Apple ID

Login View Controller

- We give you control
- Pause the game
- Present the controller



Start using Game Center with your Apple ID and play games online with your friends, wherever they are.

Making the Call

```
- (void)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    GKLocalPlayer *localPlayer = [GKLocalPlayer localPlayer];

    localPlayer.authenticationHandler = //handle the callback...

    [localPlayer authenticate];
}
```

Making the Call

```
- (void)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    GKLocalPlayer *localPlayer = [GKLocalPlayer localPlayer];

    localPlayer.authenticationHandler = //handle the callback...

    [localPlayer authenticate];
}
```

Making the Call

```
- (void)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    GKLocalPlayer *localPlayer = [GKLocalPlayer localPlayer];

    localPlayer.authenticationHandler = //handle the callback...

    [localPlayer authenticate];
}
```

Making the Call

```
- (void)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    GKLocalPlayer *localPlayer = [GKLocalPlayer localPlayer];

    localPlayer.authenticationHandler = //handle the callback...

    [localPlayer authenticate];
}
```

Making the Call

```
- (void)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    GKLocalPlayer *localPlayer = [GKLocalPlayer localPlayer];

    localPlayer.authenticationHandler = //handle the callback...

    [localPlayer authenticate];
}
```

Making the Call

```
- (void)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    GKLocalPlayer *localPlayer = [GKLocalPlayer localPlayer];

    localPlayer.authenticationHandler = //handle the callback...

    [localPlayer authenticate];
}
```

Making the Call

```
- (void)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    GKLocalPlayer *localPlayer = [GKLocalPlayer localPlayer];

    localPlayer.authenticationHandler = //handle the callback...

    [localPlayer authenticate];
}
```

Getting the Callback

```
localPlayer.authenticationHandler = ^(UIViewController *loginVC,
                                      NSError *error)
{
    if ([GKLocalPlayer localPlayer].authenticated) {
        // authentication successful
        [self enableGameCenterForPlayer:[GKLocalPlayer localPlayer]];
    }
    else if (loginVC) {
        // player not logged in yet, present the vc
        [self pauseGame];
        [self presentLoginVC:loginVC];
    }
    else {
        // authentication failed, provide graceful fallback
        [self disableGameCenter];
    }
};
```

Getting the Callback

```
localPlayer.authenticationHandler = ^(UIViewController *loginVC,
                                      NSError *error)
{
    if ([GKLocalPlayer localPlayer].authenticated) {
        // authentication successful
        [self enableGameCenterForPlayer:[GKLocalPlayer localPlayer]];
    }
    else if (loginVC) {
        // player not logged in yet, present the vc
        [self pauseGame];
        [self presentLoginVC:loginVC];
    }
    else {
        // authentication failed, provide graceful fallback
        [self disableGameCenter];
    }
};
```

Getting the Callback

```
localPlayer.authenticationHandler = ^(UIViewController *loginVC,
                                      NSError *error)
{
    if ([GKLocalPlayer localPlayer].authenticated) {
        // authentication successful
        [self enableGameCenterForPlayer:[GKLocalPlayer localPlayer]];
    }
    else if (loginVC) {
        // player not logged in yet, present the vc
        [self pauseGame];
        [self presentLoginVC:loginVC];
    }
    else {
        // authentication failed, provide graceful fallback
        [self disableGameCenter];
    }
};
```

Getting the Callback

```
localPlayer.authenticationHandler = ^(UIViewController *loginVC,
                                       NSError *error)
{
    if ([GKLocalPlayer localPlayer].authenticated) {
        // authentication successful
        [self enableGameCenterForPlayer:[GKLocalPlayer localPlayer]];
    }
    else if (loginVC) {
        // player not logged in yet, present the vc
        [self pauseGame];
        [self presentLoginVC:loginVC];
    }
    else {
        // authentication failed, provide graceful fallback
        [self disableGameCenter];
    }
};
```

Getting the Callback

```
localPlayer.authenticationHandler = ^(UIViewController *loginVC,
                                      NSError *error)
{
    if ([GKLocalPlayer localPlayer].authenticated) {
        // authentication successful
        [self enableGameCenterForPlayer:[GKLocalPlayer localPlayer]];
    }
    else if (loginVC) {
        // player not logged in yet, present the vc
        [self pauseGame];
        [self presentLoginVC:loginVC];
    }
    else {
        // authentication failed, provide graceful fallback
        [self disableGameCenter];
    }
};
```

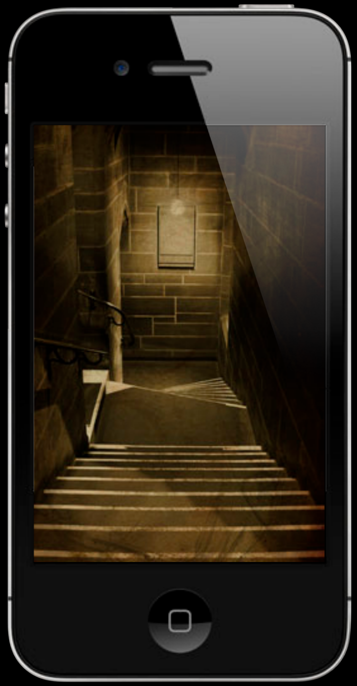
Getting the Callback

```
localPlayer.authenticationHandler = ^(UIViewController *loginVC,
                                       NSError *error)
{
    if ([GKLocalPlayer localPlayer].authenticated) {
        // authentication successful
        [self enableGameCenterForPlayer:[GKLocalPlayer localPlayer]];
    }
    else if (loginVC) {
        // player not logged in yet, present the vc
        [self pauseGame];
        [self presentLoginVC:loginVC];
    }
    else {
        // authentication failed, provide graceful fallback
        [self disableGameCenter];
    }
};
```

Getting the Callback

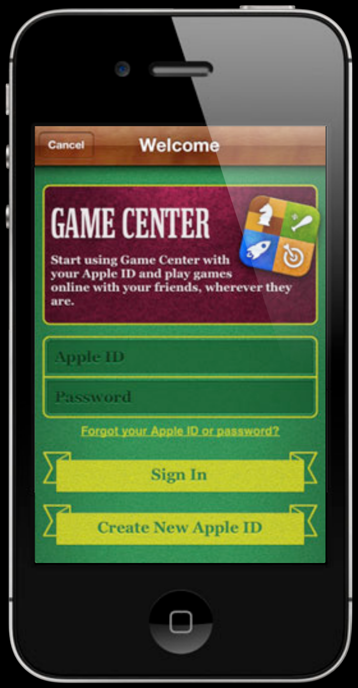
```
localPlayer.authenticationHandler = ^(UIViewController *loginVC,
                                       NSError *error)
{
    if ([GKLocalPlayer localPlayer].authenticated) {
        // authentication successful
        [self enableGameCenterForPlayer:[GKLocalPlayer localPlayer]];
    }
    else if (loginVC) {
        // player not logged in yet, present the vc
        [self pauseGame];
        [self presentLoginVC:loginVC];
    }
    else {
        // authentication failed, provide graceful fallback
        [self disableGameCenter];
    }
};
```

Handling Errors



`GKErrorCancelled`

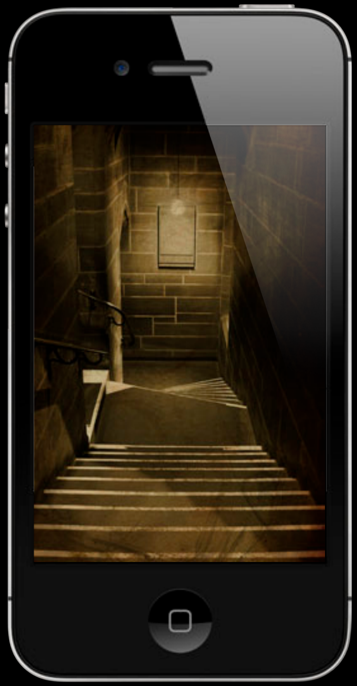
Handling Errors



Start using Game Center with your Apple ID and play games online with your friends, wherever they are.

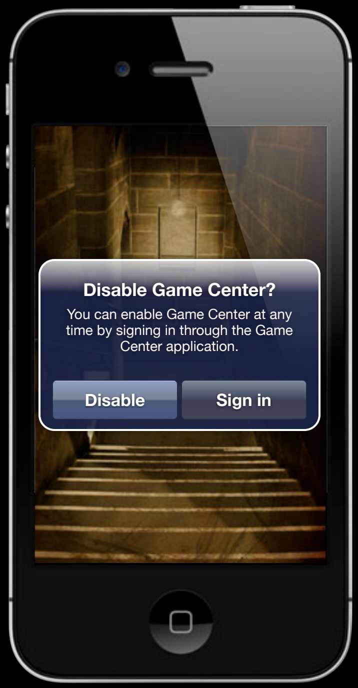
`GKErrorCancelled`

Handling Errors



`GKErrorCancelled`

Handling Errors

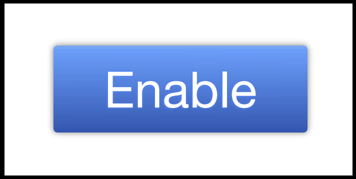

Disable Game Center?
You can enable Game Center at any time by signing in through the Game Center application.
Disable
Sign in

`GKErrorCancelled`

Handling Errors

```
GKErrorCancelled

GKErrorGameUnrecognized
```

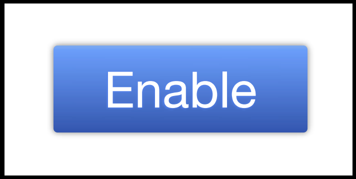
Handling Errors



`GKErrorCancelled`

`GKErrorGameUnrecognized`

Handling Errors

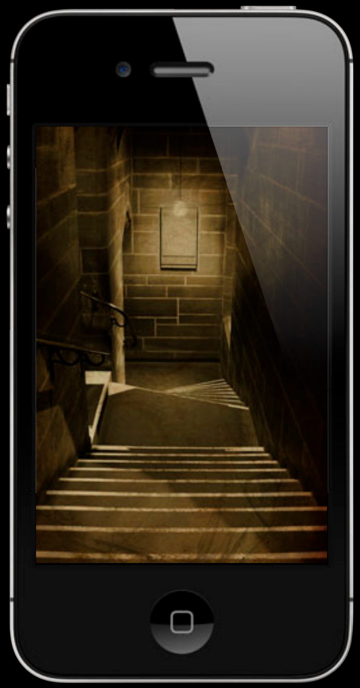


`GKErrorCancelled`

`GKErrorGameUnrecognized`

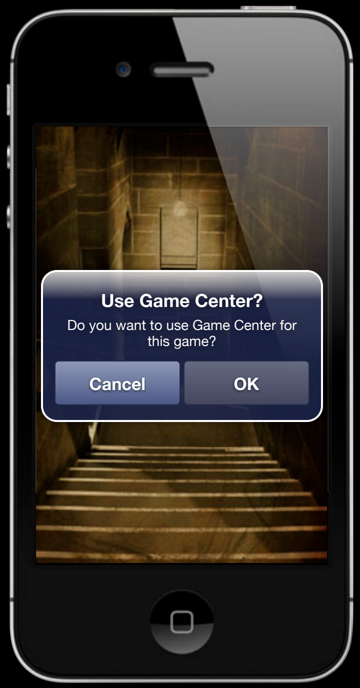
Best Practices

Avoid "Enable Game Center" dialogs



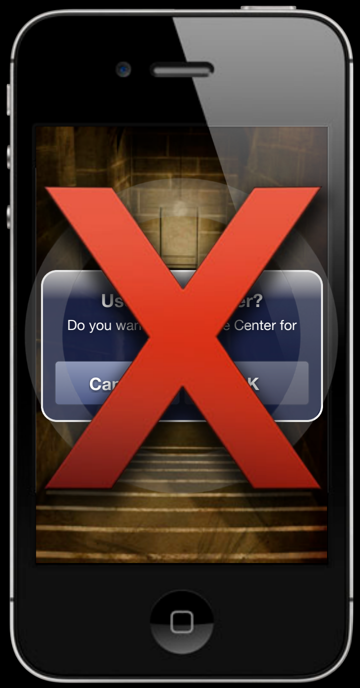
Best Practices

Avoid "Enable Game Center" dialogs



Best Practices

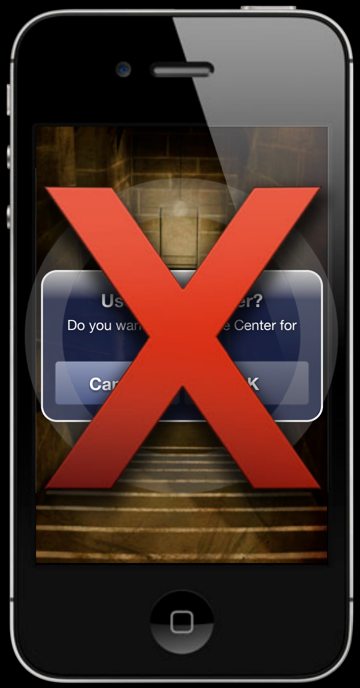
Avoid “Enable Game Center” dialogs



Best Practices

Avoid "Enable Game Center" dialogs

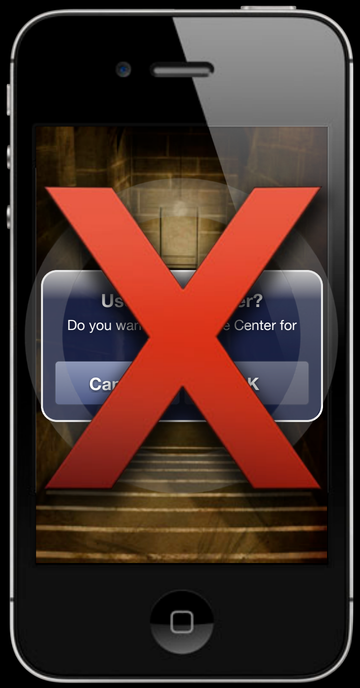
- May have already authenticated



Best Practices

Avoid "Enable Game Center" dialogs

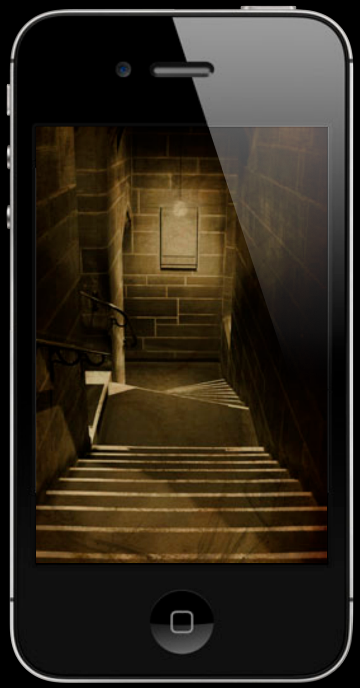
- May have already authenticated
- May have opted out



Best Practices

Avoid “Enable Game Center” dialogs

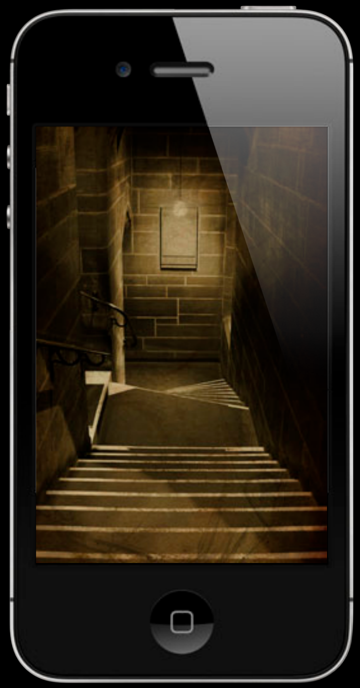
- May have already authenticated
- May have opted out



Best Practices

Avoid “Enable Game Center” dialogs

- May have already authenticated
- May have opted out



Summary

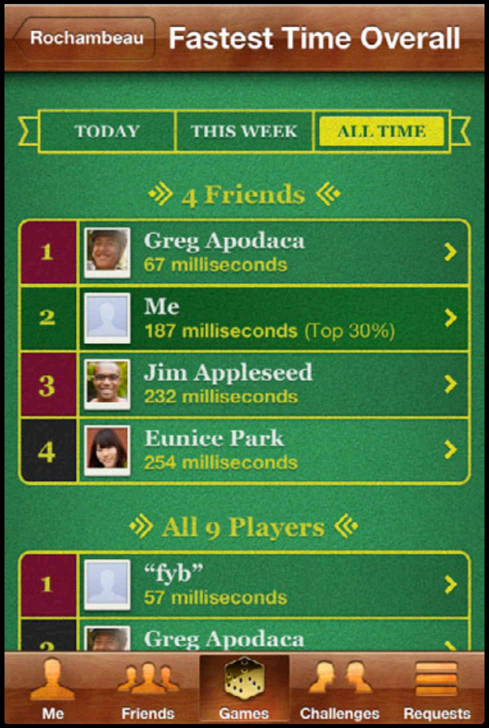
- Authentication comes first
- Asynchronous
- Fall back gracefully
- Avoid extra dialogs

Scores and Achievements

`GKScore` `GKAchievement`

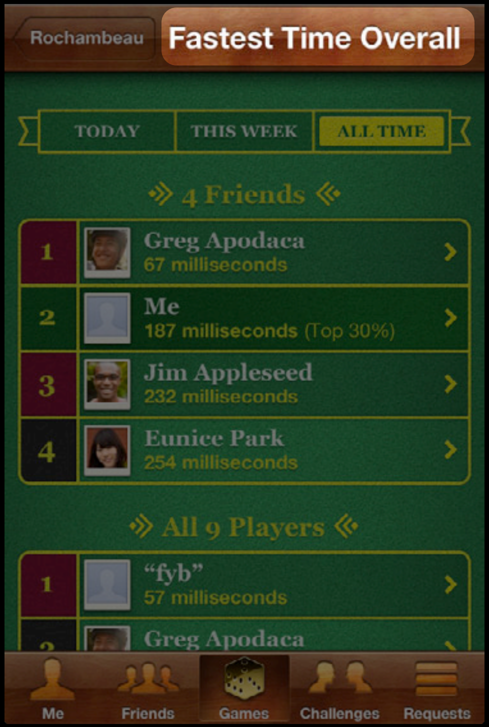
GKScore

```
NSString *category

int64_t value

uint64_t context
```



GKScore

```
NSString *category
int64_t value
uint64_t context
```


Rochambeau
Fastest Time Overall
TODAY | THIS WEEK | ALL TIME

4 Friends

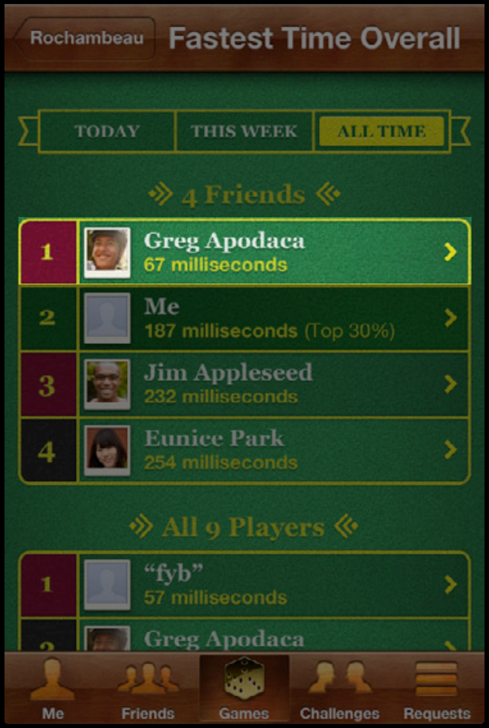
1	Greg Apodaca	67 milliseconds
2	Me	187 milliseconds (Top 30%)
3	Jim Appleseed	232 milliseconds
4	Eunice Park	254 milliseconds

All 9 Players

1	"fyb"	57 milliseconds
2	Greg Apodaca	

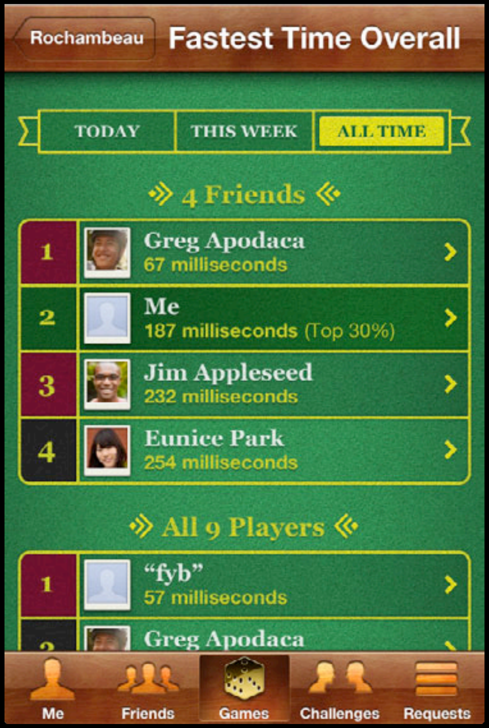
GKScore

```
NSString *category
int64_t value
uint64_t context
```



GKScore

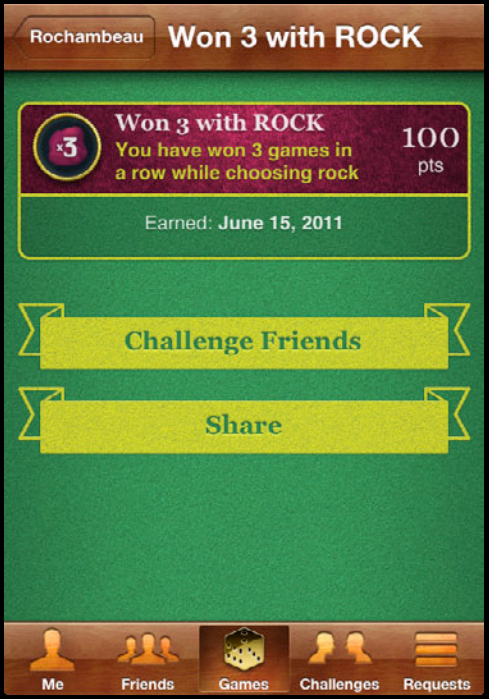
```
NSString *category

int64_t value

uint64_t context
```



1	Greg Apodaca	67 milliseconds
2	Me	187 milliseconds (Top 30%)
3	Jim Appleseed	232 milliseconds
4	Eunice Park	254 milliseconds
1	"fyb"	57 milliseconds
2	Greg Apodaca	

GKAchievement

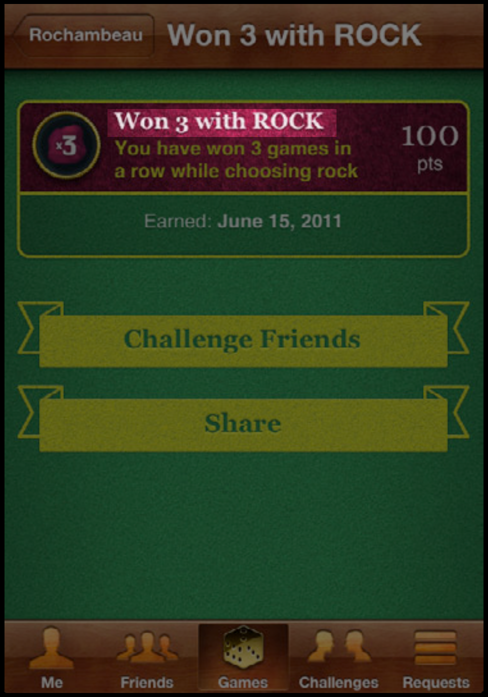
```
NSString *identifier

double percentComplete
```



GKAchievement

```
NSString *identifier
```

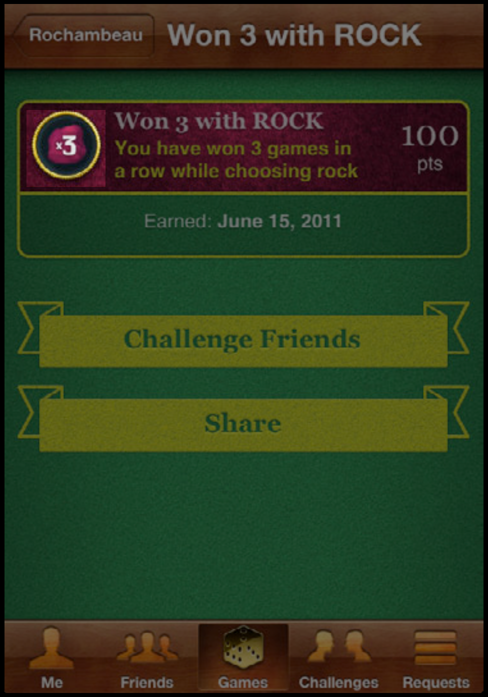
```
double percentComplete
```



Won 3 with ROCK
You have won 3 games in a row while choosing rock
100 pts
Earned: June 15, 2011

GKAchievement

```
NSString *identifier

double percentComplete
```



Submitting Achievements

```
GKAchievement *myAchievement = [[GKAchievement alloc]
   initWithIdentifier:@”com.mystudio.runandgun”];

myAchievement.percentComplete = 100.0;

[myAchievement reportAchievementWithCompletionHandler:^(NSError *error) {
    if (error) {
        // handle the error
    }
}];
```

Submitting Achievements

```
GKAchievement *myAchievement = [[GKAchievement alloc]
   initWithIdentifier:@”com.mystudio.runandgun”];

myAchievement.percentComplete = 100.0;

[myAchievement reportAchievementWithCompletionHandler:^(NSError *error) {
    if (error) {
        // handle the error
    }
}];
```

Submitting Achievements

```
GKAchievement *myAchievement = [[GKAchievement alloc]
   initWithIdentifier:@”com.mystudio.runandgun”];

myAchievement.percentComplete = 100.0;

[myAchievement reportAchievementWithCompletionHandler:^(NSError *error) {
    if (error) {
        // handle the error
    }
}];
```

Submitting Achievements

```
GKAchievement *myAchievement = [[GKAchievement alloc]
   initWithIdentifier:@”com.mystudio.runandgun”];

myAchievement.percentComplete = 100.0;

[myAchievement reportAchievementWithCompletionHandler:^(NSError *error) {
    if (error) {
        // handle the error
    }
}];
```

Submitting Achievements

```
GKAchievement *myAchievement = [[GKAchievement alloc]
   initWithIdentifier:@"com.mystudio.runandgun"];

myAchievement.percentComplete = 100.0;

[myAchievement reportAchievementWithCompletionHandler:^(NSError *error) {
    if (error) {
        // handle the error
    }
}];
```

Submitting Scores

```
GKScore *myScore = [[GKScore alloc]
    initWithCategory:@”com.mystudio.level1”];

myScore.value = 300;

myScore.context = gameContext;

[myScore reportScoreWithCompletionHandler:^(NSError *error) {
    if (error) {
        // handle the error
    }
}];
```

Submitting Scores

```
GKScore *myScore = [[GKScore alloc]
    initWithCategory:@"com.mystudio.level1"];

myScore.value = 300;

myScore.context = gameContext;

[myScore reportScoreWithCompletionHandler:^(NSError *error) {
    if (error) {
        // handle the error
    }
}];
```

Submitting Scores

```
GKScore *myScore = [[GKScore alloc]
    initWithCategory:@”com.mystudio.level1”];

myScore.value = 300;

myScore.context = gameContext;

[myScore reportScoreWithCompletionHandler:^(NSError *error) {
    if (error) {
        // handle the error
    }
}];
```

Submitting Scores

```
GKScore *myScore = [[GKScore alloc]
    initWithCategory:@”com.mystudio.level1”];

myScore.value = 300;

myScore.context = gameContext;

[myScore reportScoreWithCompletionHandler:^(NSError *error) {
    if (error) {
        // handle the error
    }
}];
```

Submitting Scores

```
GKScore *myScore = [[GKScore alloc]
    initWithCategory:@"com.mystudio.level1"];

myScore.value = 300;

myScore.context = gameContext;

[myScore reportScoreWithCompletionHandler:^(NSError *error) {
    if (error) {
        // handle the error
    }
}];
```

Submitting Scores

```
GKScore *myScore = [[GKScore alloc]
    initWithCategory:@”com.mystudio.level1”];

myScore.value = 300;

myScore.context = gameContext;

[myScore reportScoreWithCompletionHandler:^(NSError *error) {
    if (error) {
        // handle the error
    }
}];
```

Submitting Scores

```
GKScore *myScore = [[GKScore alloc]
    initWithCategory:@"com.mystudio.level1"];

myScore.value = 300;

myScore.context = gameContext;

[myScore reportScoreWithCompletionHandler:^(NSError *error) {
    if (error) {
        // handle the error
    }
}];
```

Submitting Scores

```
GKScore *myScore = [[GKScore alloc]
    initWithCategory:@”com.mystudio.level1”];

myScore.value = 300;

myScore.context = myShip.color;

[myScore reportScoreWithCompletionHandler:^(NSError *error) {
    if (error) {
        // handle the error
    }
}];
```

Submitting Scores

```
GKScore *myScore = [[GKScore alloc]
    initWithCategory:@”com.mystudio.level1”];

myScore.value = 300;

myScore.context = myShip.engine;

[myScore reportScoreWithCompletionHandler:^(NSError *error) {
    if (error) {
        // handle the error
    }
}];
```

Submitting Scores

```
GKScore *myScore = [[GKScore alloc]
    initWithCategory:@"com.mystudio.level1"];

myScore.value = 300;

myScore.context = myShip.weapons;

[myScore reportScoreWithCompletionHandler:^(NSError *error) {
    if (error) {
        // handle the error
    }
}];
```

Submitting Scores

```
GKScore *myScore = [[GKScore alloc]
    initWithCategory:@”com.mystudio.level1”];

myScore.value = 300;

myScore.context = [self encodedURLForLastPlaythrough];

[myScore reportScoreWithCompletionHandler:^(NSError *error) {
    if (error) {
        // handle the error
    }
}];
```

Submitting Scores

```
GKScore *myScore = [[GKScore alloc]
    initWithCategory:@”com.mystudio.level1”];

myScore.value = 300;

myScore.context = [self encodedURLForLastPlaythrough];

[myScore reportScoreWithCompletionHandler:^(NSError *error) {
    if (error) {
        // handle the error
    }
}];
```

Resubmission

- iOS 4: Recommended
  - Check for errors, resubmit
- iOS 5: Unnecessary
  - Handled by store and forward
- iOS 6: Incompatible
  - Incompatible with challenges
  - Remove resubmission code

Resubmission

- iOS 4: Recommended
  - Check for errors, resubmit
- iOS 5: Unnecessary
  - Handled by store and forward
- iOS 6: Incompatible
  - Incompatible with challenges
  - Remove resubmission code

```
[self syncGameCenterScoresAndAchievements];
```

Resubmission

- iOS 4: Recommended
  - Check for errors, resubmit
- iOS 5: Unnecessary
  - Handled by store and forward
- iOS 6: Incompatible
  - Incompatible with challenges
  - Remove resubmission code

```
[self syncGameCenterScoresAndAchievements];
```

Unified Experience

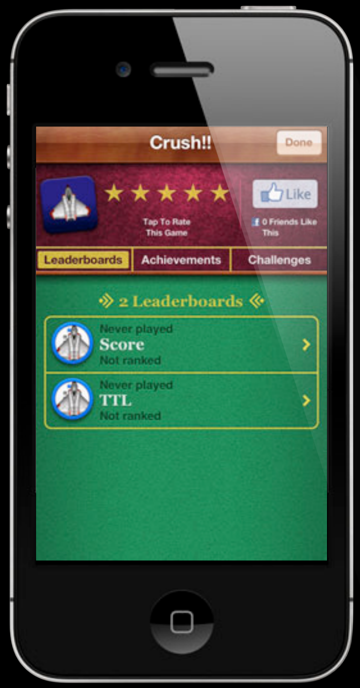
Unified Experience

GKGameCenterViewController

- Unified view of
  - Leaderboards
  - Achievements
  - Challenges
- App Rating
- Sharing

Unified Experience
GKGameCenterViewController

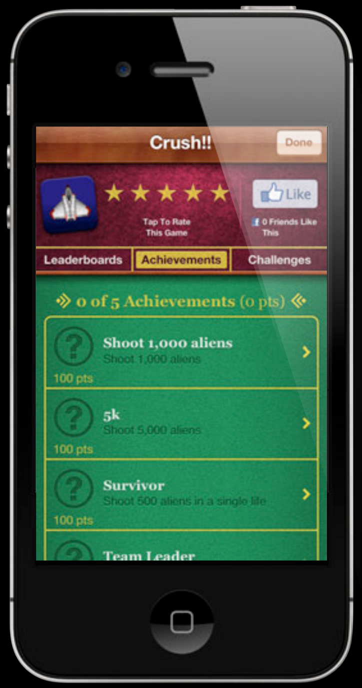
- Unified view of
  - Leaderboards
  - Achievements
  - Challenges
- App Rating
- Sharing



Unified Experience

GKGameCenterViewController

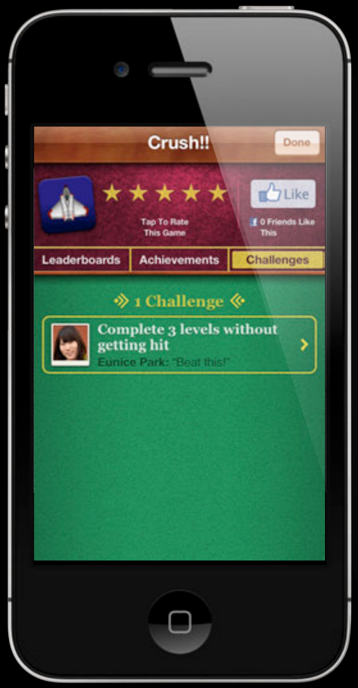
- Unified view of
  - Leaderboards
  - Achievements
  - Challenges
- App Rating
- Sharing



Unified Experience

GKGameCenterViewController

- Unified view of
  - Leaderboards
  - Achievements
  - Challenges
- App Rating
- Sharing



GKGameCenterViewController

```
@implementation MyViewController

- (void)showUIButtonPressed:(id)sender {
    // get the shared VC instance
    GKGameCenterViewController *gcvc = [GKGameCenterViewController
        sharedController];

    gcvc.gameCenterDelegate = self;

    // set the starting view state
    gcvc.viewState = GKGameCenterViewControllerStateDefault;

    [self presentViewController:gcvc animated:YES completion:nil];
}
```

GKGameCenterViewController

```
@implementation MyViewController

- (void)showUIButtonPressed:(id)sender {
    // get the shared VC instance
    GKGameCenterViewController *gcvc = [GKGameCenterViewController
        sharedController];

    gcvc.gameCenterDelegate = self;

    // set the starting view state
    gcvc.viewState = GKGameCenterViewControllerStateDefault;

    [self presentViewController:gcvc animated:YES completion:nil];
}
```

GKGameCenterViewController

```
@implementation MyViewController

- (void)showUIButtonPressed:(id)sender {
    // get the shared VC instance
    GKGameCenterViewController *gcvc = [GKGameCenterViewController
        sharedController];

    gcvc.gameCenterDelegate = self;

    // set the starting view state
    gcvc.viewState = GKGameCenterViewControllerStateDefault;

    [self presentViewController:gcvc animated:YES completion:nil];
}
```

GKGameCenterViewController

```
@implementation MyViewController

- (void)showUIButtonPressed:(id)sender {
    // get the shared VC instance
    GKGameCenterViewController *gcvc = [GKGameCenterViewController
        sharedController];

    gcvc.gameCenterDelegate = self;

    // set the starting view state
    gcvc.viewState = GKGameCenterViewControllerStateDefault;

    [self presentViewController:gcvc animated:YES completion:nil];
}
```

GKGameCenterViewController

```
@implementation MyViewController

- (void)showUIButtonPressed:(id)sender {
    // get the shared VC instance
    GKGameCenterViewController *gcvc = [GKGameCenterViewController
        sharedController];

    gcvc.gameCenterDelegate = self;

    // set the starting view state
    gcvc.viewState = GKGameCenterViewControllerStateDefault;

    [self presentViewController:gcvc animated:YES completion:nil];
}
```

GKGameCenterViewController

```
@implementation MyViewController

- (void)showUIButtonPressed:(id)sender {
    // get the shared VC instance
    GKGameCenterViewController *gcvc = [GKGameCenterViewController
        sharedController];

    gcvc.gameCenterDelegate = self;

    // set the starting view state
    gcvc.viewState = GKGameCenterViewControllerStateDefault;

    [self presentViewController:gcvc animated:YES completion:nil];
}
```

GKGameCenterViewController

```
@implementation MyViewController

- (void)showUIButtonPressed:(id)sender {
    // get the shared VC instance
    GKGameCenterViewController *gcvc = [GKGameCenterViewController
        sharedController];

    gcvc.gameCenterDelegate = self;

    // set the starting view state
    gcvc.viewState = GKGameCenterViewControllerStateDefault;

    [self presentViewController:gcvc animated:YES completion:nil];
}
```

GKGameCenterViewController

```
@implementation MyViewController

- (void)showUIButtonPressed:(id)sender {
    // get the shared VC instance
    GKGameCenterViewController *gcvc = [GKGameCenterViewController
        sharedController];

    gcvc.gameCenterDelegate = self;

    // set the starting view state
    gcvc.viewState = GKGameCenterViewControllerStateDefault;

    [self presentViewController:gcvc animated:YES completion:nil];
}
```

GKGameCenterViewController

```
@implementation MyViewController

- (void)showUIButtonPressed:(id)sender {
    // get the shared VC instance
    GKGameCenterViewController *gcvc = [GKGameCenterViewController
        sharedController];

    gcvc.gameCenterDelegate = self;

    // set the starting view state
    gcvc.viewState = GKGameCenterViewControllerStateDefault;

    [self presentViewController:gcvc animated:YES completion:nil];
}
```

GKGameCenterViewController

```
@implementation MyViewController

- (void)showUIButtonPressed:(id)sender {
    // get the shared VC instance
    GKGameCenterViewController *gcvc = [GKGameCenterViewController
        sharedController];

    gcvc.gameCenterDelegate = self;

    // set the starting view state
    gcvc.viewState = GKGameCenterViewControllerStateAchievements;

    [self presentViewController:gcvc animated:YES completion:nil];
}
```

GKGameCenterViewController

```
@implementation MyViewController

- (void)showUIButtonPressed:(id)sender {
    // get the shared VC instance
    GKGameCenterViewController *gcvc = [GKGameCenterViewController
        sharedController];

    gcvc.gameCenterDelegate = self;

    // set the starting view state
    gcvc.viewState = GKGameCenterViewControllerStateChallenges;

    [self presentViewController:gcvc animated:YES completion:nil];
}
```

GKGameCenterViewController

```
@implementation MyViewController

- (void)showUIButtonPressed:(id)sender {
    // get the shared VC instance
    GKGameCenterViewController *gcvc = [GKGameCenterViewController
        sharedController];

    gcvc.gameCenterDelegate = self;

    // set the starting view state
    gcvc.viewState = GKGameCenterViewControllerStateChallenges;

    [self presentViewController:gcvc animated:YES completion:nil];
}
```

GKGameCenterViewController

```
@implementation MyViewController

- (void)showUIButtonPressed:(id)sender {
    // get the shared VC instance
    GKGameCenterViewController *gcvc = [GKGameCenterViewController
        sharedController];

    gcvc.gameCenterDelegate = self;

    // set the starting view state
    gcvc.viewState = GKGameCenterViewControllerStateChallenges;



    [self presentViewController:gcvc animated:YES completion:nil];
}
```

GKGameCenterViewController

```
@implementation MyViewController

- (void)showUIButtonPressed:(id)sender {
    // get the shared VC instance
    GKGameCenterViewController *gcvc = [GKGameCenterViewController
        sharedController];

    gcvc.gameCenterDelegate = self;

    // set the starting view state
    gcvc.viewState = GKGameCenterViewControllerStateLeaderboards;
    gcvc.leaderboardTimeScope = GKLeaderboardTimeScopeToday;
    gcvc.leaderboardCategory = @"com.mystudio.level1";

    [self presentViewController:gcvc animated:YES completion:nil];
}
```

GKGameCenterViewController

```
@implementation MyViewController

- (void)showUIButtonPressed:(id)sender {
    // get the shared VC instance
    GKGameCenterViewController *gcvc = [GKGameCenterViewController
        sharedController];

    gcvc.gameCenterDelegate = self;

    // set the starting view state
    gcvc.viewState = GKGameCenterViewControllerStateLeaderboards;
    gcvc.leaderboardTimeScope = GKLeaderboardTimeScopeToday;
    gcvc.leaderboardCategory = @"com.mystudio.level1";

    [self presentViewController:gcvc animated:YES completion:nil];
}
```

GKGameCenterViewController

```
@implementation MyViewController

- (void)showUIButtonPressed:(id)sender {
    // get the shared VC instance
    GKGameCenterViewController *gcvc = [GKGameCenterViewController
        sharedController];

    gcvc.gameCenterDelegate = self;

    // set the starting view state
    gcvc.viewState = GKGameCenterViewControllerStateLeaderboards;
    gcvc.leaderboardTimeScope = GKLeaderboardTimeScopeToday;
    gcvc.leaderboardCategory = @"com.mystudio.level1";

    [self presentViewController:gcvc animated:YES completion:nil];
}
```

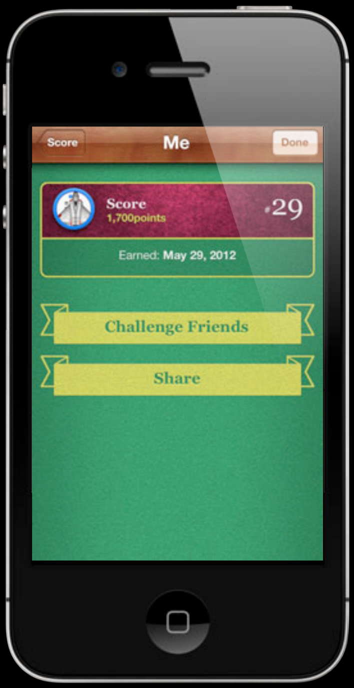
GKGameCenterViewController

```
@implementation MyViewController

- (void)showUIButtonPressed:(id)sender {
    // get the shared VC instance
    GKGameCenterViewController *gcvc = [GKGameCenterViewController
        sharedController];

    gcvc.gameCenterDelegate = self;

    // set the starting view state
    gcvc.viewState = GKGameCenterViewControllerStateLeaderboards;
    gcvc.leaderboardTimeScope = GKLeaderboardTimeScopeToday;
    gcvc.leaderboardCategory = @"com.mystudio.level1";

    [self presentViewController:gcvc animated:YES completion:nil];
}
```

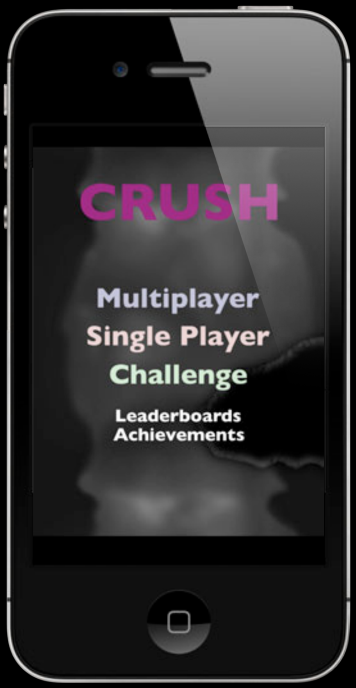
GKGameCenterViewController

```
@implementation MyViewController

- (void)showUIButtonPressed:(id)sender {
    // get the shared VC instance
    GKGameCenterViewController *gcvc = [GKGameCenterViewController
        sharedController];

    gcvc.gameCenterDelegate = self;

    // set the starting view state
    gcvc.viewState = GKGameCenterViewControllerStateLeaderboards;
    gcvc.leaderboardTimeScope = GKLeaderboardTimeScopeToday;
    gcvc.leaderboardCategory = @"com.mystudio.level1";

    [self presentViewController:gcvc animated:YES completion:nil];
}
```

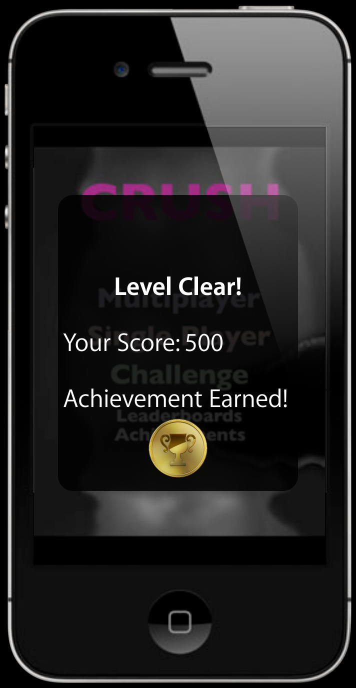
Sharing Scores and Achievements



Sharing Scores and Achievements



Sharing Scores and Achievements

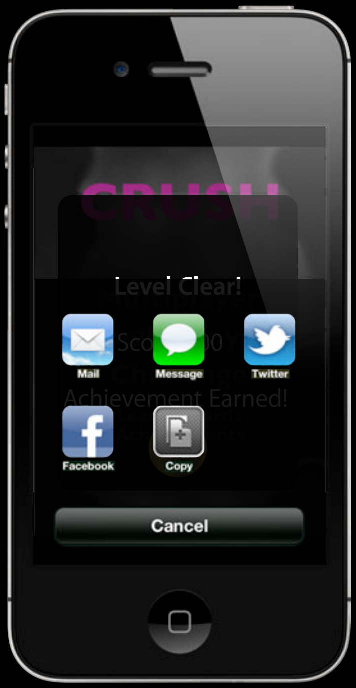


Level Clear!

Your Score: 500

Achievement Earned!

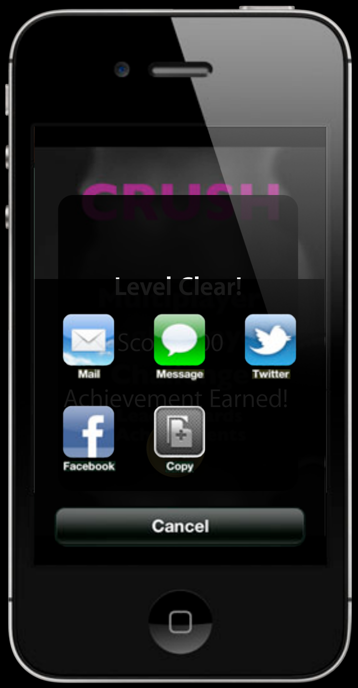
Sharing Scores and Achievements



Sharing Scores and Achievements

UIActivityViewController

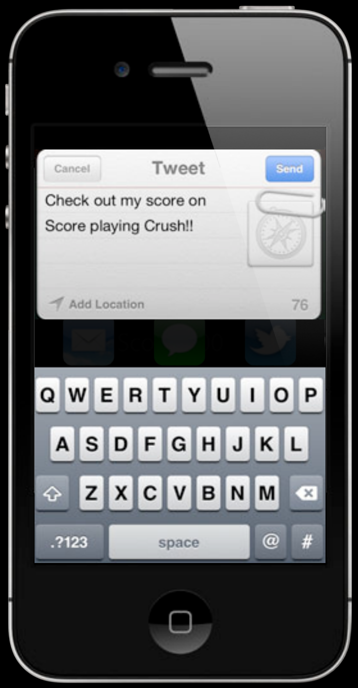
- Handles transport for different services
- Direct support for scores and achievements



Sharing Scores and Achievements

UIActivityViewController

- Handles transport for different services
- Direct support for scores and achievements



Check out my score on Score playing Crush!!

Sharing

UIActivityViewController and GKScore

```
// Get the score we want to share
GKScore *myScore = [self getScoreForSharing];

// Set up the VC with the score
UIActivityViewController *activityViewController = [[UIActivityViewController
    alloc] initWithActivityItems:@[myScore] applicationActivities:nil];

// Dismiss the VC when it’s done
activityViewController.completionHandler = ^(NSString *activityType, BOOL
completed) {
    if (completed)
        [self dismissViewControllerAnimated:YES];
};
// Present the VC
[self presentViewController:activityViewController animated:YES
completion:nil];
```

Sharing

UIActivityViewController and GKScore

```
// Get the score we want to share
GKScore *myScore = [self getScoreForSharing];

// Set up the VC with the score
UIActivityViewController *activityViewController = [[UIActivityViewController
    alloc] initWithActivityItems:@[myScore] applicationActivities:nil];

// Dismiss the VC when it's done
activityViewController.completionHandler = ^(NSString *activityType, BOOL
completed) {
    if (completed)
        [self dismissViewControllerAnimated:YES];
};
// Present the VC
[self presentViewController:activityViewController animated:YES
completion:nil];
```

Sharing

UIActivityViewController and GKScore

```
// Get the score we want to share
GKScore *myScore = [self getScoreForSharing];

// Set up the VC with the score
UIActivityViewController *activityViewController = [[UIActivityViewController
    alloc] initWithActivityItems:@[myScore] applicationActivities:nil];

// Dismiss the VC when it’s done
activityViewController.completionHandler = ^(NSString *activityType, BOOL
completed) {
    if (completed)
        [self dismissViewControllerAnimated:YES];
};
// Present the VC
[self presentViewController:activityViewController animated:YES
completion:nil];
```

Sharing

UIActivityViewController and GKScore

```
// Get the score we want to share
GKScore *myScore = [self getScoreForSharing];

// Set up the VC with the score
UIActivityViewController *activityViewController = [[UIActivityViewController
    alloc] initWithActivityItems:@[myScore] applicationActivities:nil];

// Dismiss the VC when it’s done
activityViewController.completionHandler = ^(NSString *activityType, BOOL
completed) {
    if (completed)
        [self dismissViewControllerAnimated:YES];
};
// Present the VC
[self presentViewController:activityViewController animated:YES
completion:nil];
```

Sharing

UIActivityViewController and GKScore

```
// Get the score we want to share
GKScore *myScore = [self getScoreForSharing];

// Set up the VC with the score
UIActivityViewController *activityViewController = [[UIActivityViewController
    alloc] initWithActivityItems:@[myScore] applicationActivities:nil];

// Dismiss the VC when it’s done
activityViewController.completionHandler = ^(NSString *activityType, BOOL
completed) {
    if (completed)
        [self dismissViewControllerAnimated:YES];
};
// Present the VC
[self presentViewController:activityViewController animated:YES
completion:nil];
```

Sharing

UIActivityViewController and GKScore

```
// Get the score we want to share
GKScore *myScore = [self getScoreForSharing];

// Set up the VC with the score
UIActivityViewController *activityViewController = [[UIActivityViewController
    alloc] initWithActivityItems:@[myScore] applicationActivities:nil];

// Dismiss the VC when it’s done
activityViewController.completionHandler = ^(NSString *activityType, BOOL
completed) {
    if (completed)
        [self dismissViewControllerAnimated:YES];
};
// Present the VC
[self presentViewController:activityViewController animated:YES
completion:nil];
```

Sharing

UIActivityViewController + GKAchievement

```
// Get the achievement we want to share
GKAchievement *myAchievement = [self getAchievementForSharing];

// Set up the VC with the achievement
UIActivityViewController *activityViewController = [[UIActivityViewController
    alloc] initWithActivityItems:@[myAchievement] applicationActivities:nil];

// Dismiss the VC when it’s done
activityViewController.completionHandler = ^(NSString *activityType, BOOL
completed) {
    if (completed)
        [self dismissViewControllerAnimated:YES];
};
// Present the VC
[self presentViewController:activityViewController animated:YES
completion:nil];
```

Sharing

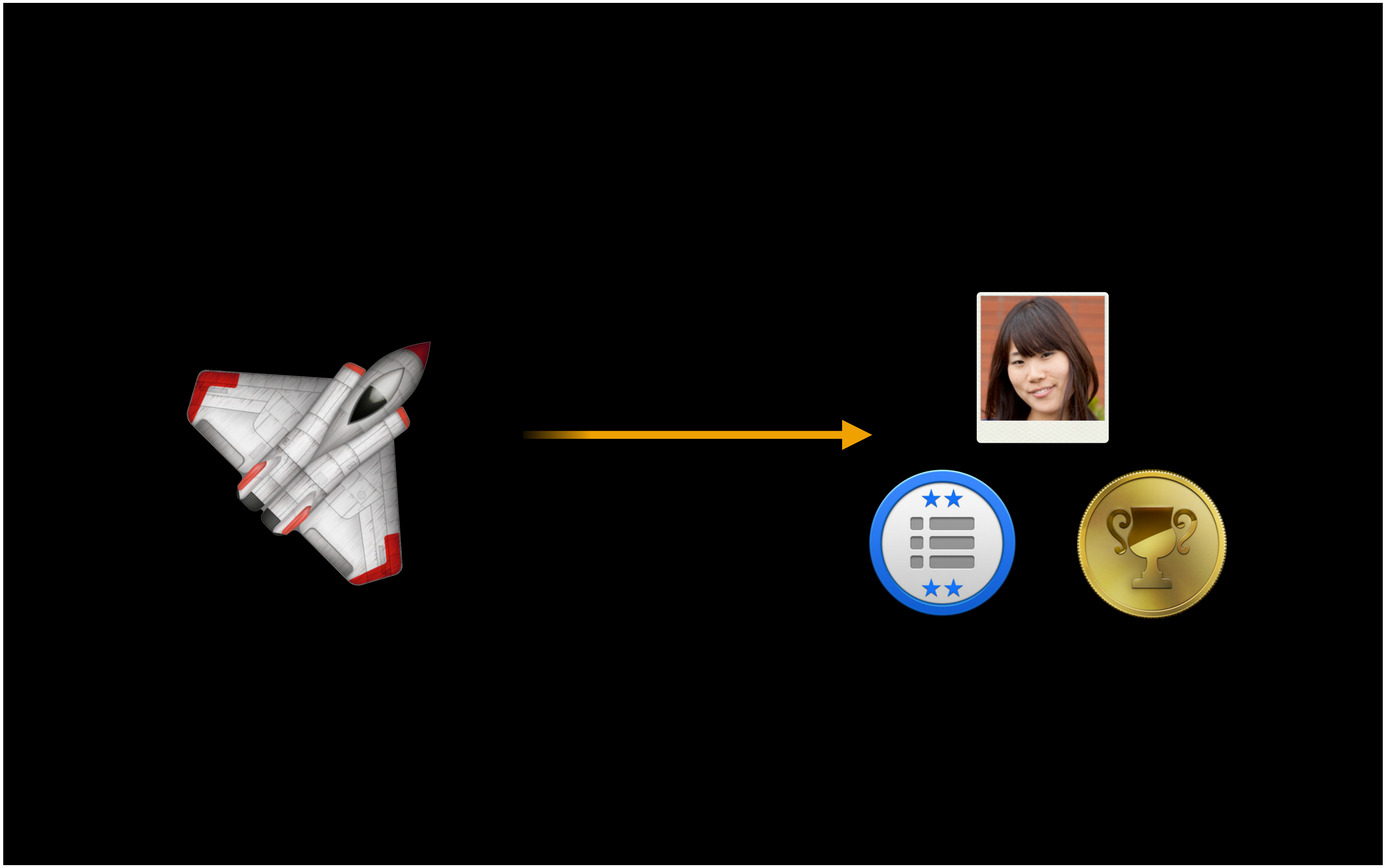
UIActivityViewController + GKAchievement

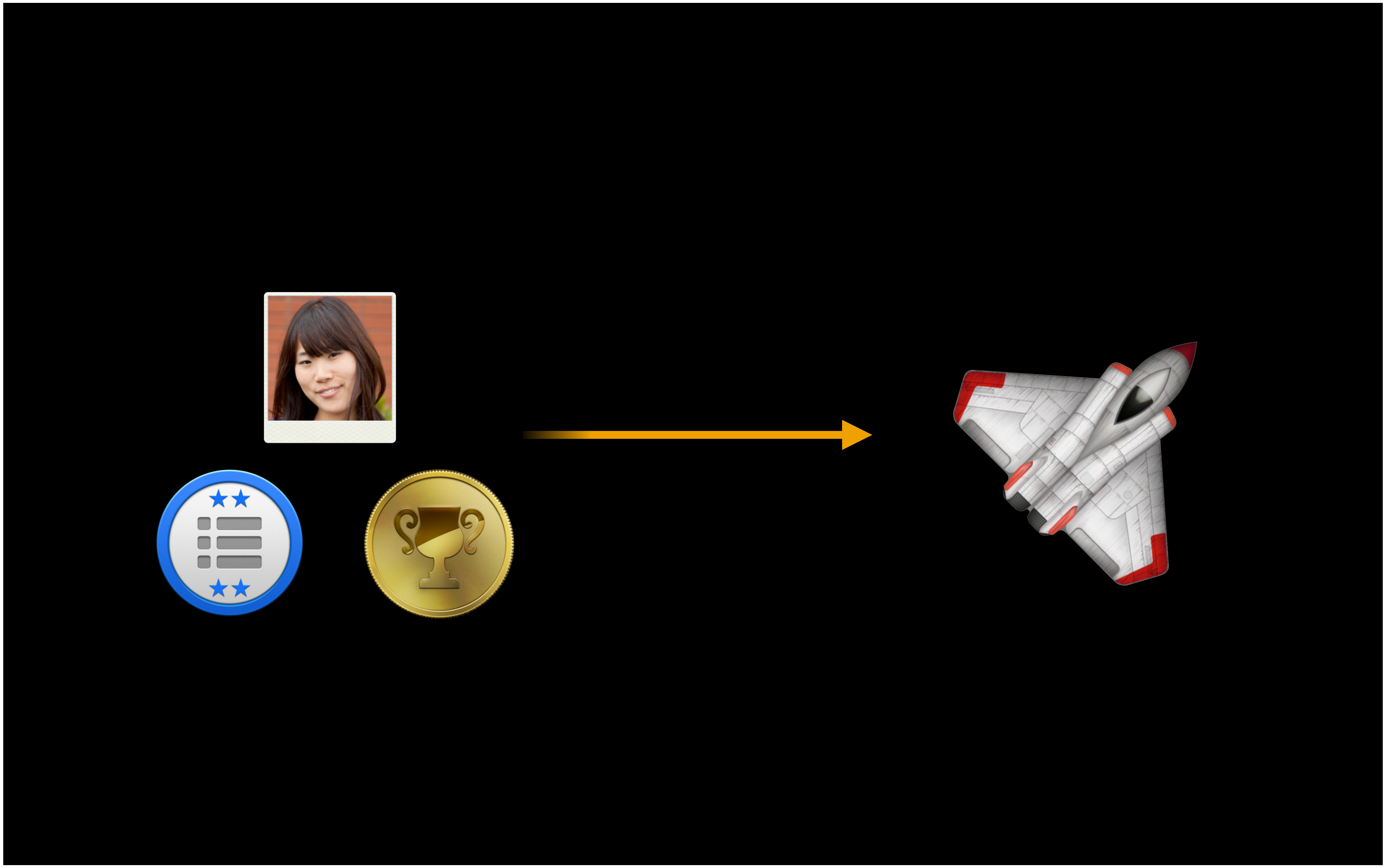
```
// Get the achievement we want to share
GKAchievement *myAchievement = [self getAchievementForSharing];

// Set up the VC with the achievement
UIActivityViewController *activityViewController = [[UIActivityViewController
    alloc] initWithActivityItems:@[myAchievement] applicationActivities:nil];

// Dismiss the VC when it’s done
activityViewController.completionHandler = ^(NSString *activityType, BOOL
completed) {
    if (completed)
        [self dismissViewControllerAnimated:YES];
};
// Present the VC
[self presentViewController:activityViewController animated:YES
completion:nil];
```

Scores and Achievements

Things to remember

- Think about context
- Don’t archive and resubmit
- Drop-in UI is a few lines away





Eunice

Eunice

Eunice

Eunice

Eunice

Greg

Jessica

Jim

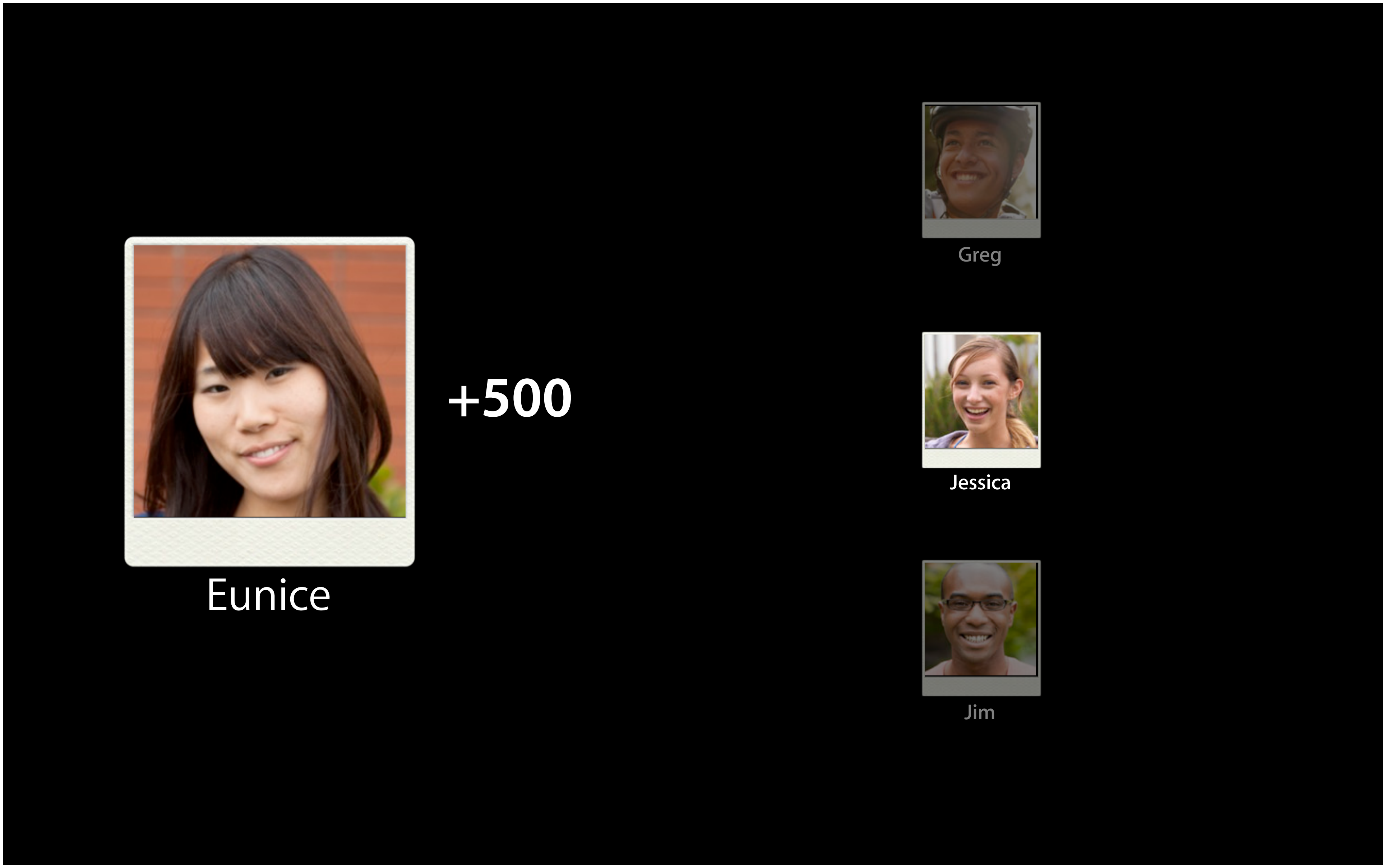
Eunice

+500

Greg

Jessica

Jim


Eunice
+500
Greg
Jessica
Jim

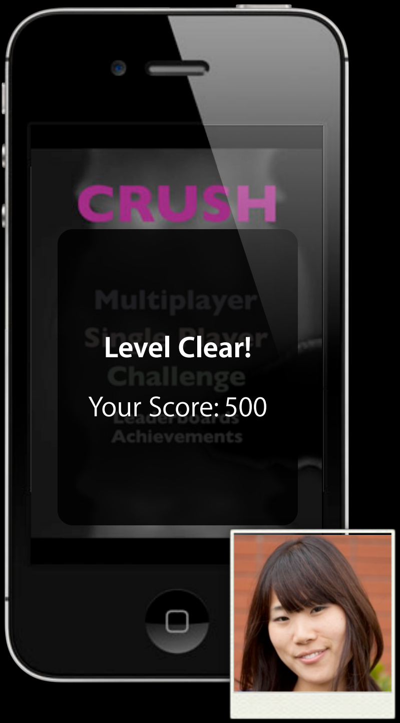
Challenges

New

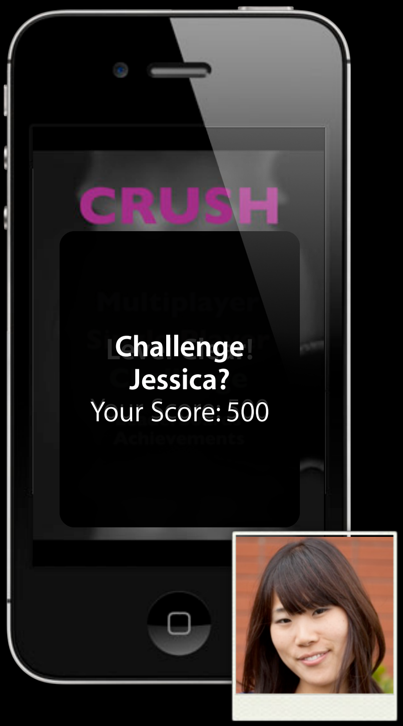
CRUSH
Multiplayer
Single Player
Challenge
Leaderboards
Achievements

iPod
9:41 AM
FaceTime
Calendar
Photos
Camera
Videos
YouTube
Maps
Weather
Notes
Reminders
Clock
Game Center
Newsstand
iTunes
App Store
Settings
Messages
Mail
Safari

New


CRUSH
Multiplayer
Single Player
Challenge
Leaderboards
Achievements
Level Clear!
Your Score: 500

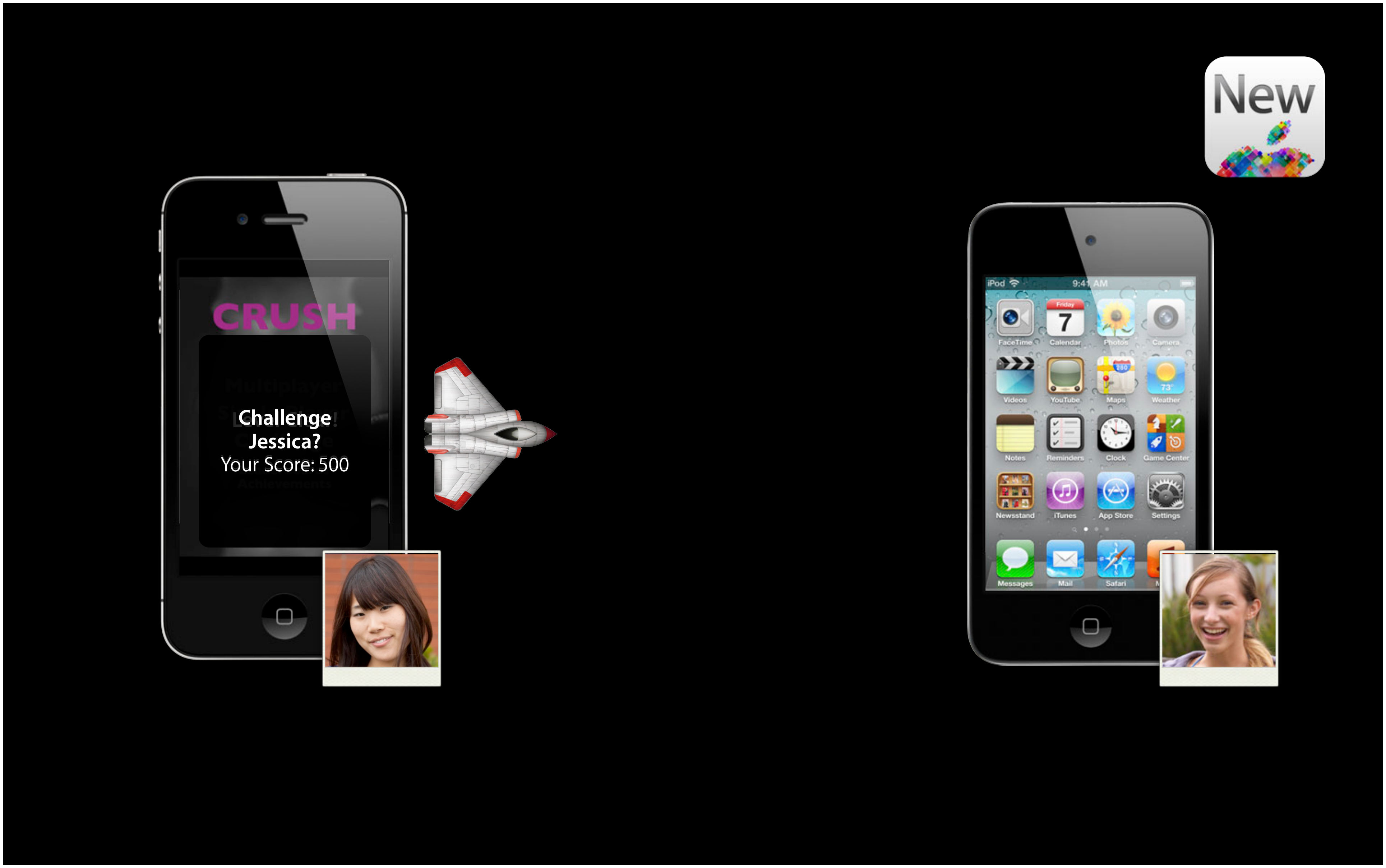
iPod
9:41 AM
Friday
7
FaceTime
Calendar
Photos
Camera
Videos
YouTube
Maps
Weather
Notes
Reminders
Clock
Game Center
Newsstand
iTunes
App Store
Settings
Messages
Mail
Safari



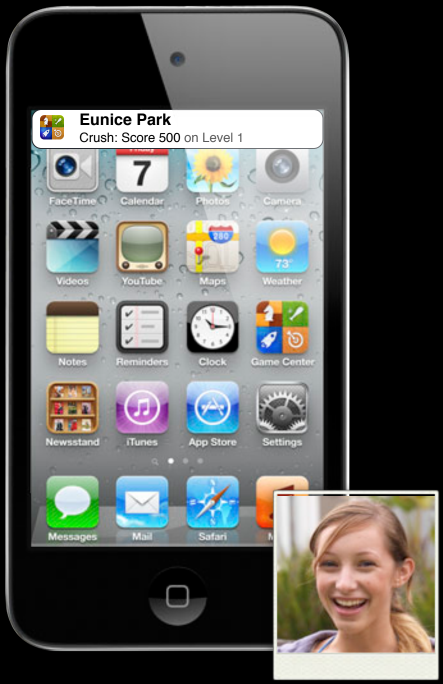
CRUSH

Challenge Jessica?
Your Score: 500

New


New
CRUSH
Challenge Jessica?
Your Score: 500
iPod
9:41 AM
FaceTime
Calendar
Photos
Camera
Videos
YouTube
Maps
Weather
Notes
Reminders
Clock
Game Center
Newsstand
iTunes
App Store
Settings
Messages
Mail
Safari

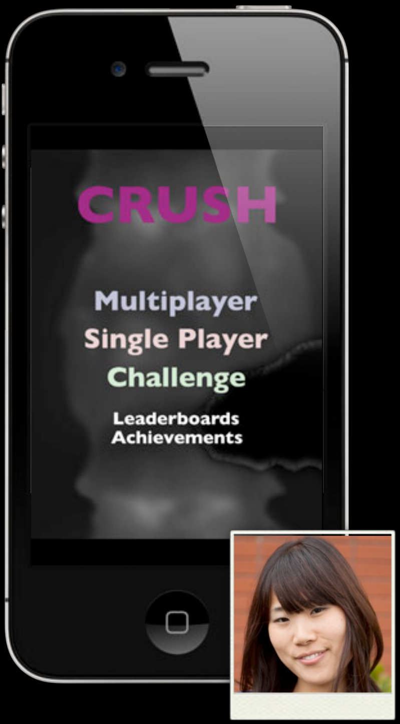
New

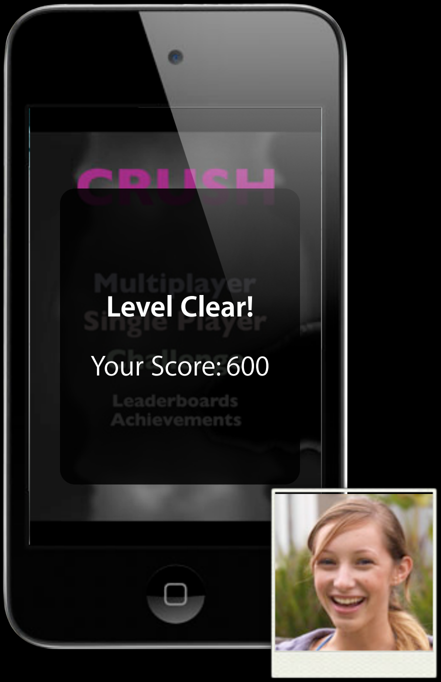

Eunice Park
Crush: Score 500 on Level 1

CRUSH
Multiplayer
Single Player
Challenge
Leaderboards
Achievements

CRUSH
Multiplayer
Single Player
Challenge
Leaderboards
Achievements

New

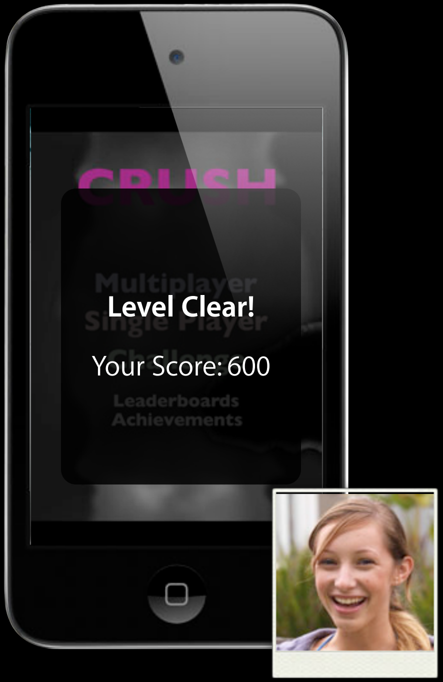

CRUSH
Multiplayer
Single Player
Challenge
Leaderboards
Achievements


CRUSH
Level Clear!
Your Score: 600
Multiplayer
Single Player
Challenge
Leaderboards
Achievements

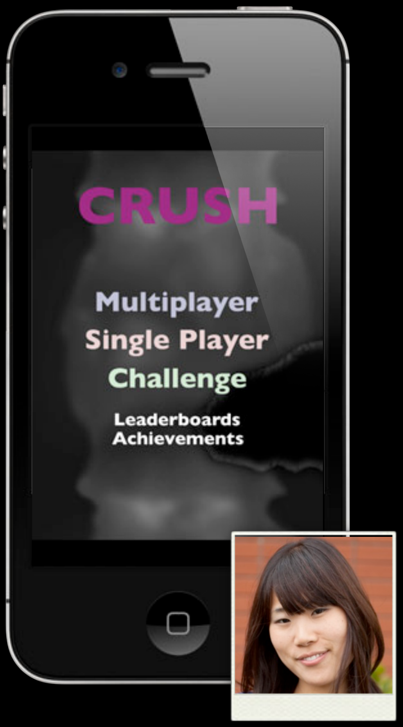
New

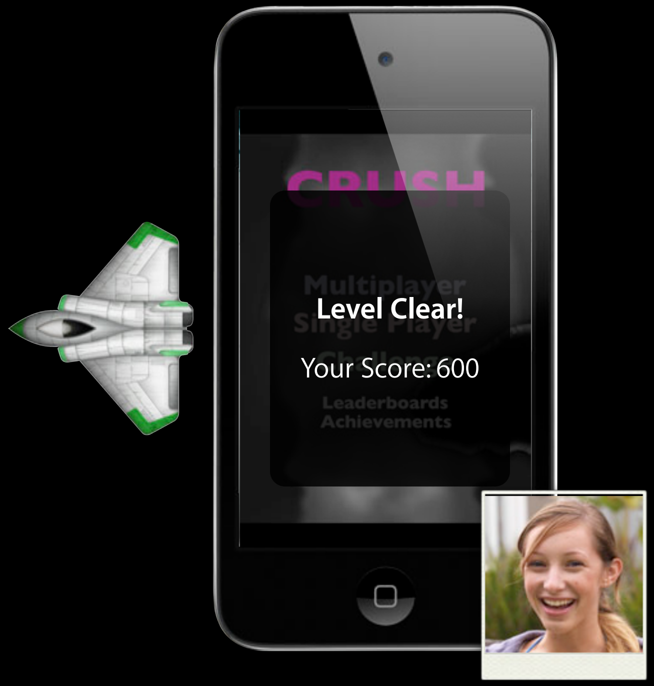
New

CRUSH
Multiplayer
Single Player
Challenge
Leaderboards
Achievements

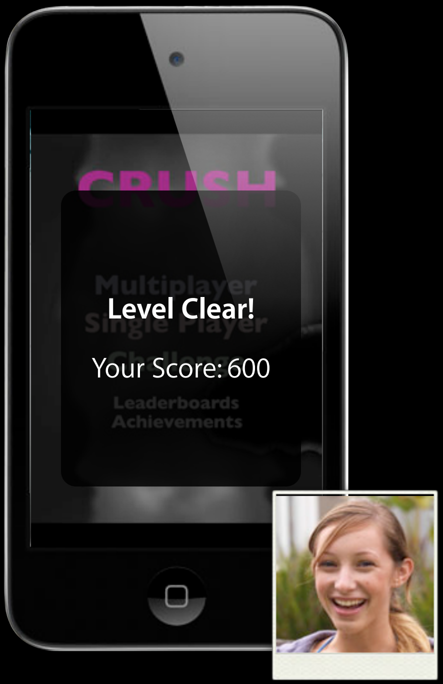

CRUSH
Level Clear!
Your Score: 600
Leaderboards
Achievements

New

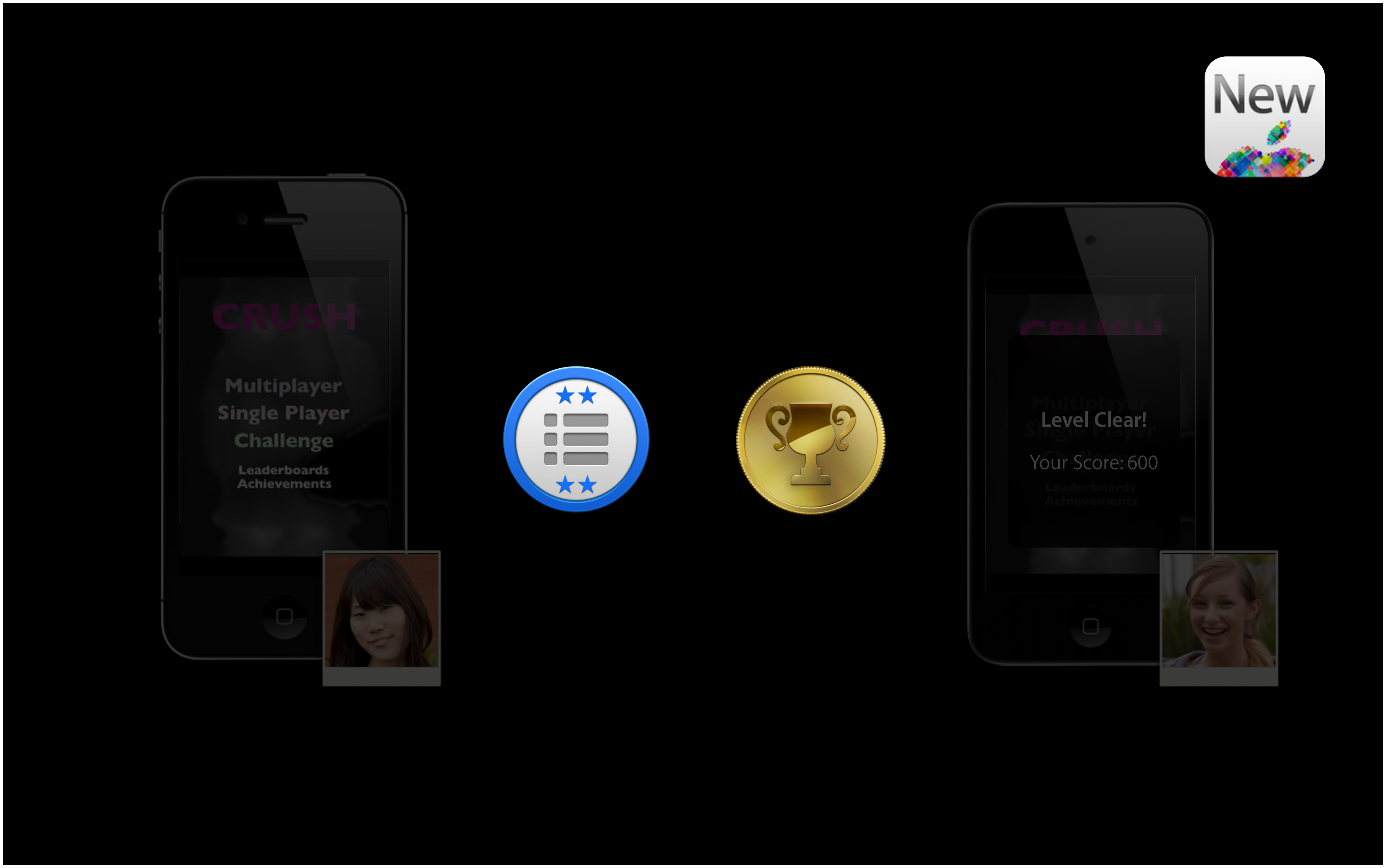



Level Clear!

Your Score: 600


Level Clear!

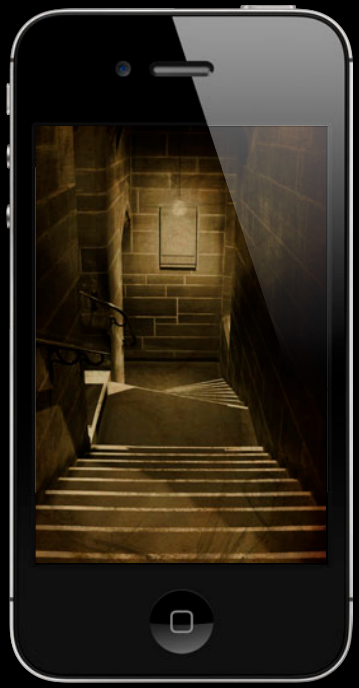
Your Score: 600


New
CRUSH
Multiplayer
Single Player
Challenge
Leaderboards
Achievements
Level Clear!
Your Score: 600

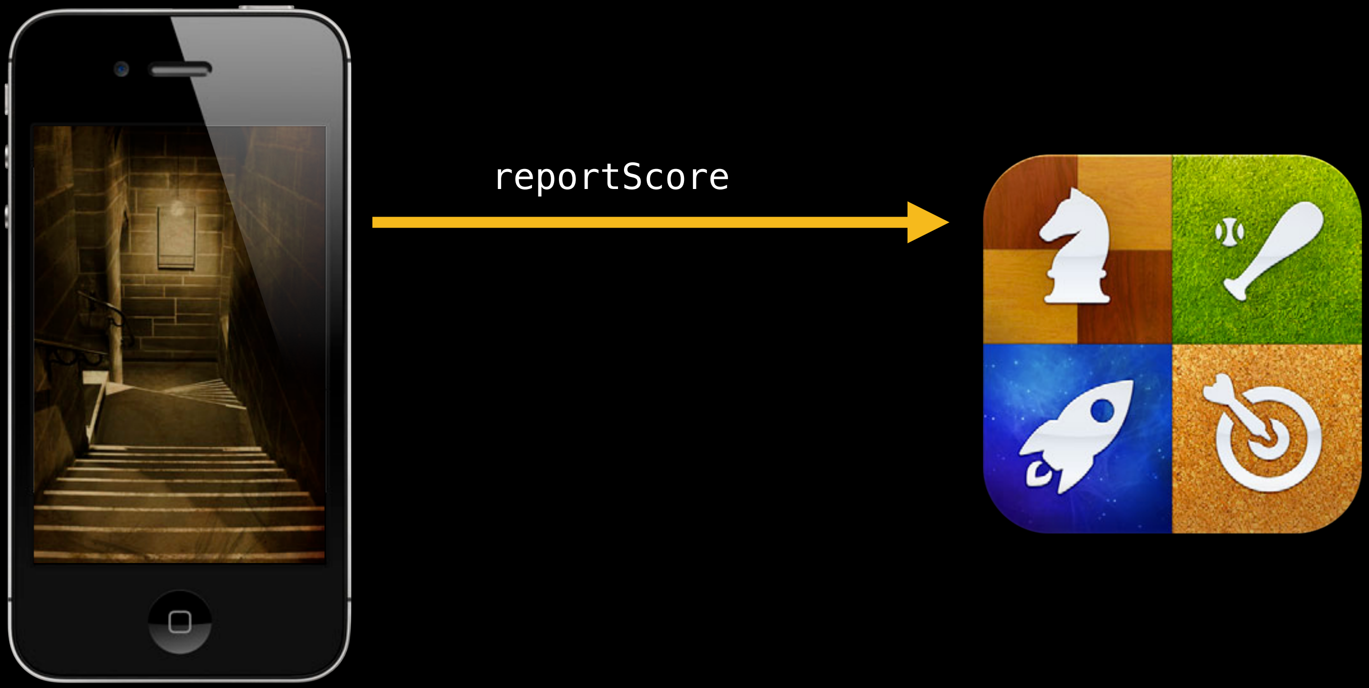
Demo

Megan Gardner
iOS Engineer

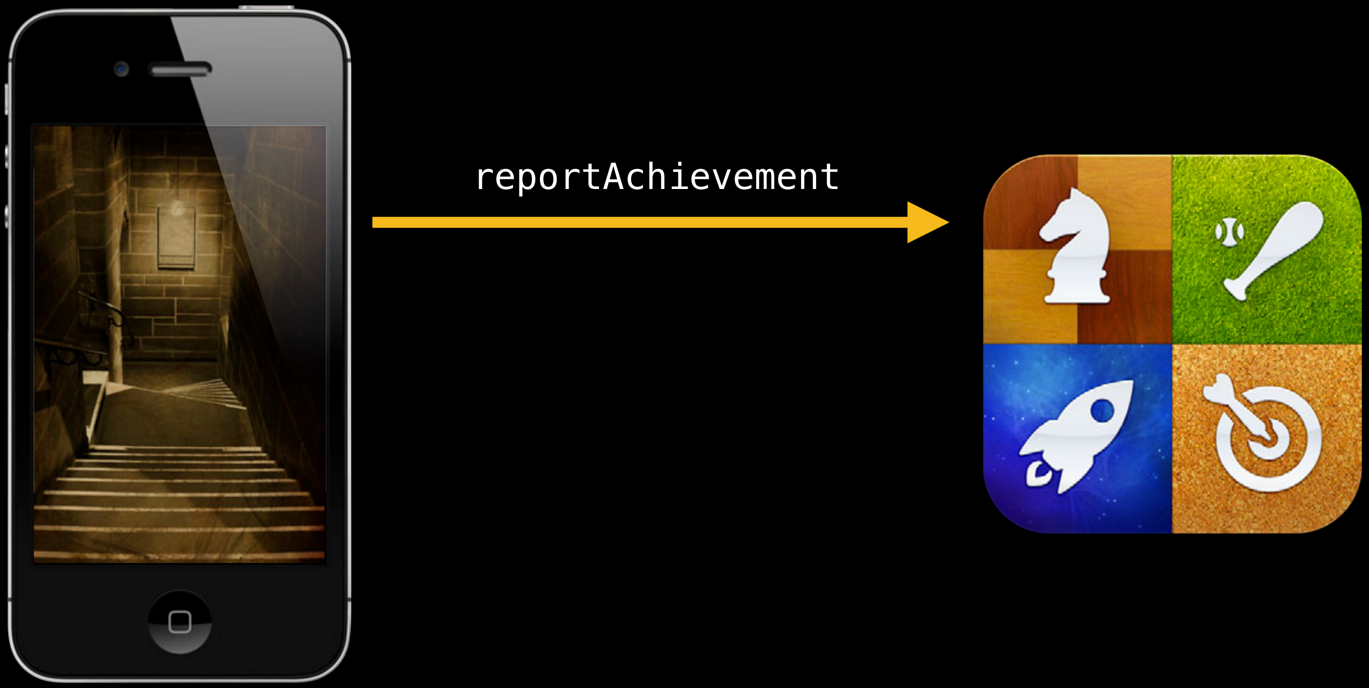
Challenges



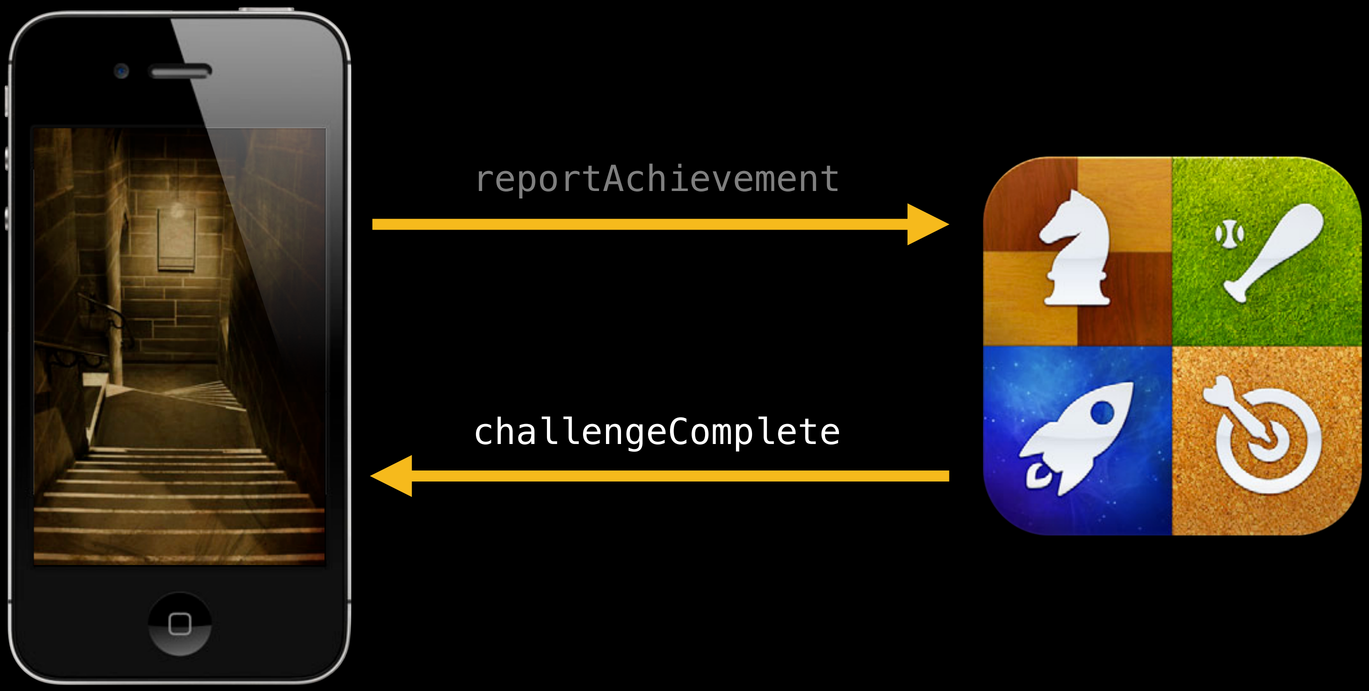
Challenges



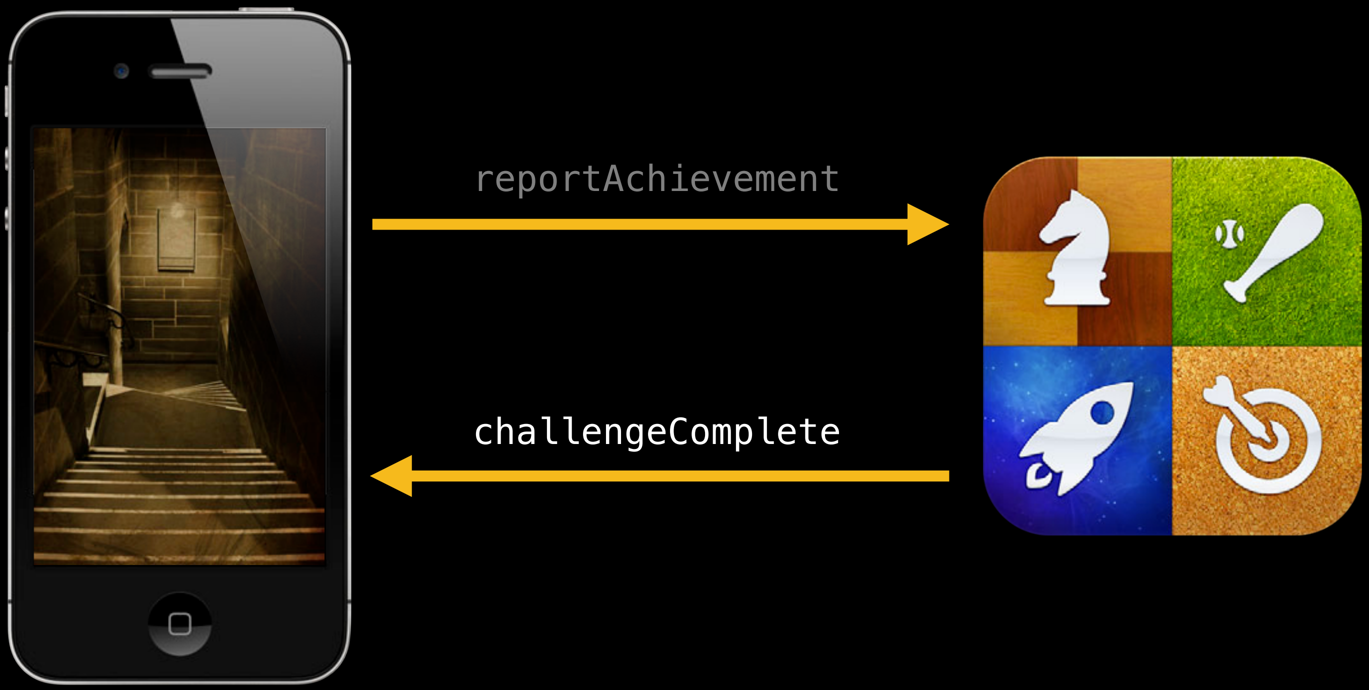
Challenges



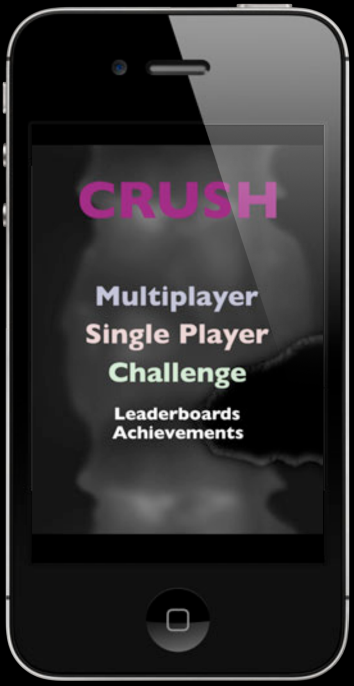
Challenges



Challenges

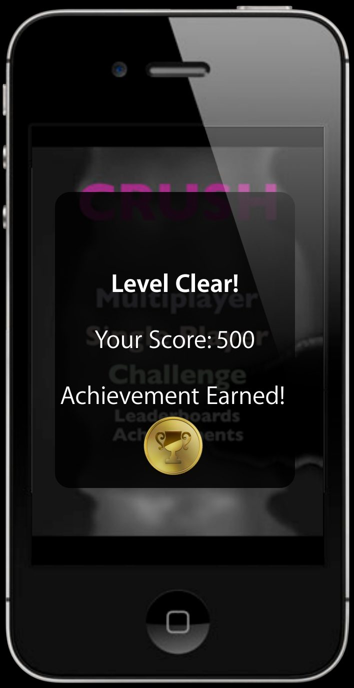


Issuing In-Game



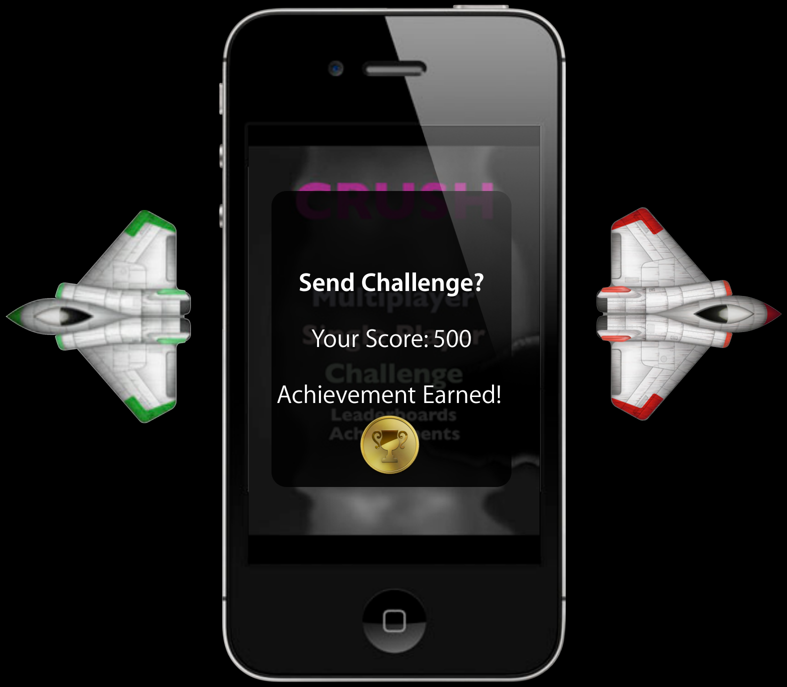
Issuing In-Game

New



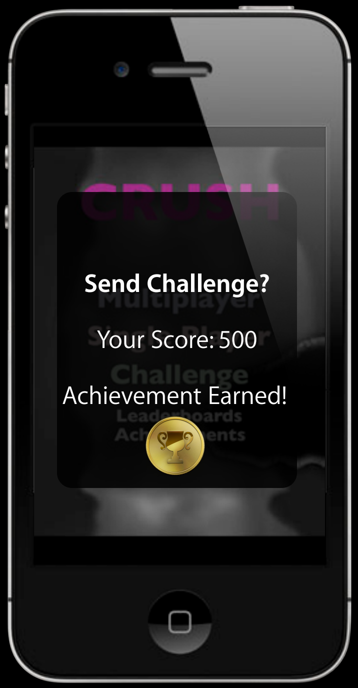
Issuing In-Game

New



Issuing In-Game

New



Send Challenge?

Your Score: 500

Achievement Earned!

Issuing Challenges

Achievement Challenges

```
// get friends and message from current player
NSArray *playersForChallenge = self.playersForChallenge;
NSString *message = self.messageForChallenge;

// set up the achievement
GKAchievement *myAchievement = [[GKAchievement alloc]
    initWithIdentifier:@”com.mystudio.runandgun”];
myAchievement.percentComplete = 100.0;

// issue the challenge
[myAchievement issueChallengeToPlayers:playersForChallenge
    message:message];
```

Issuing Challenges

Achievement Challenges

```
// get friends and message from current player
NSArray *playersForChallenge = self.playersForChallenge;
NSString *message = self.messageForChallenge;

// set up the achievement
GKAchievement *myAchievement = [[GKAchievement alloc]
    initWithIdentifier:@”com.mystudio.runandgun”];
myAchievement.percentComplete = 100.0;

// issue the challenge
[myAchievement issueChallengeToPlayers:playersForChallenge
    message:message];
```

Issuing Challenges

Achievement Challenges

```
// get friends and message from current player
NSArray *playersForChallenge = self.playersForChallenge;
NSString *message = self.messageForChallenge;

// set up the achievement
GKAchievement *myAchievement = [[GKAchievement alloc]
    initWithIdentifier:@”com.mystudio.runandgun”];
myAchievement.percentComplete = 100.0;

// issue the challenge
[myAchievement issueChallengeToPlayers:playersForChallenge
    message:message];
```

Issuing Challenges

Achievement Challenges

```
// get friends and message from current player
NSArray *playersForChallenge = self.playersForChallenge;
NSString *message = self.messageForChallenge;

// set up the achievement
GKAchievement *myAchievement = [[GKAchievement alloc]
    initWithIdentifier:@”com.mystudio.runandgun”];
myAchievement.percentComplete = 100.0;

// issue the challenge
[myAchievement issueChallengeToPlayers:playersForChallenge
    message:message];
```

Issuing Challenges

Achievement Challenges

```
// get friends and message from current player
NSArray *playersForChallenge = self.playersForChallenge;
NSString *message = self.messageForChallenge;

// set up the achievement
GKAchievement *myAchievement = [[GKAchievement alloc]
    initWithIdentifier:@”com.mystudio.runandgun”];
myAchievement.percentComplete = 100.0;

// issue the challenge
[myAchievement issueChallengeToPlayers:playersForChallenge
    message:message];
```

Issuing Challenges

Achievement Challenges

```
// get friends and message from current player
NSArray *playersForChallenge = self.playersForChallenge;
NSString *message = self.messageForChallenge;

// set up the achievement
GKAchievement *myAchievement = [[GKAchievement alloc]
    initWithIdentifier:@”com.mystudio.runandgun”];
myAchievement.percentComplete = 100.0;

// issue the challenge
[myAchievement issueChallengeToPlayers:playersForChallenge
    message:message];
```

Issuing Challenges

Score Challenges

```
// get friends and message from current player
NSArray *playersForChallenge = self.playersForChallenge;
NSString *message = self.messageForChallenge;

// set up the score
GKScore *myScore = [[GKScore alloc]
    initWithCategory:@”com.mystudio.level1”];
myScore.value = 300;
myScore.context = [self ghostDataURLForLastPlaythrough];

// issue the challenge
[myScore issueChallengeToPlayers:playersForChallenge
    message:message];
```

Issuing Challenges

Score Challenges

```
// get friends and message from current player
NSArray *playersForChallenge = self.playersForChallenge;
NSString *message = self.messageForChallenge;

// set up the score
GKScore *myScore = [[GKScore alloc]
    initWithCategory:@”com.mystudio.level1”];
myScore.value = 300;
myScore.context = [self ghostDataURLForLastPlaythrough];

// issue the challenge
[myScore issueChallengeToPlayers:playersForChallenge
    message:message];
```

Issuing Challenges

Score Challenges

```
// get friends and message from current player
NSArray *playersForChallenge = self.playersForChallenge;
NSString *message = self.messageForChallenge;

// set up the score
GKScore *myScore = [[GKScore alloc]
    initWithCategory:@”com.mystudio.level1”];
myScore.value = 300;
myScore.context = [self ghostDataURLForLastPlaythrough];

// issue the challenge
[myScore issueChallengeToPlayers:playersForChallenge
    message:message];
```

Issuing Challenges

Score Challenges

```
// get friends and message from current player
NSArray *playersForChallenge = self.playersForChallenge;
NSString *message = self.messageForChallenge;

// set up the score
GKScore *myScore = [[GKScore alloc]
    initWithCategory:@”com.mystudio.level1”];
myScore.value = 300;
myScore.context = [self ghostDataURLForLastPlaythrough];

// issue the challenge
[myScore issueChallengeToPlayers:playersForChallenge
    message:message];
```

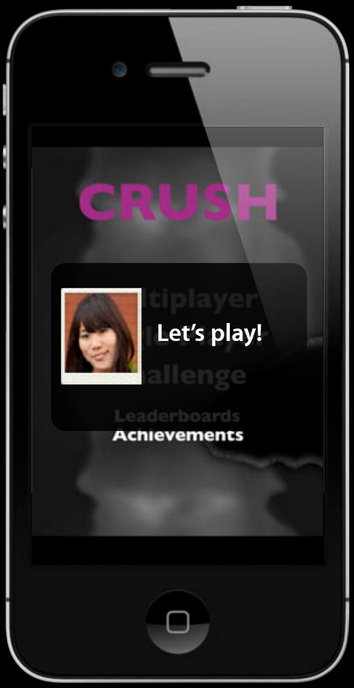
Receiving Challenges

Receiving Challenges

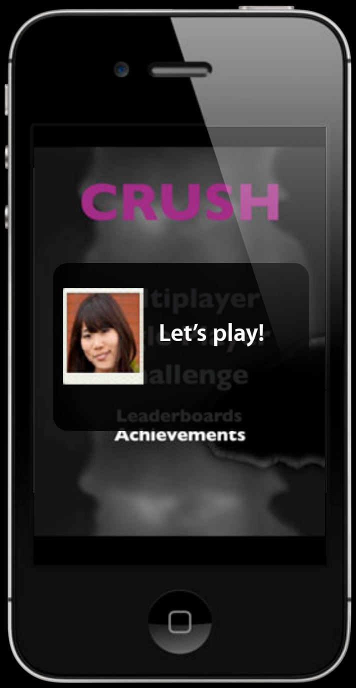
New

Receiving Challenges

New



Receiving Challenges



`GKChallengeEventHandler`

GKChallengeEventHandler

- Singleton for challenge events
- Delegate methods for
  - Reception, completion
  - Overriding banner behavior

Handling Selected Challenges

Steps for handling events

- Set up the delegate
- Override event methods
- Retrieve issuer info
  - Name
  - Photo
- Load any challenge data
  - GKScore, context
  - GKAchievement
- Present your custom UI

Handling Selected Challenges

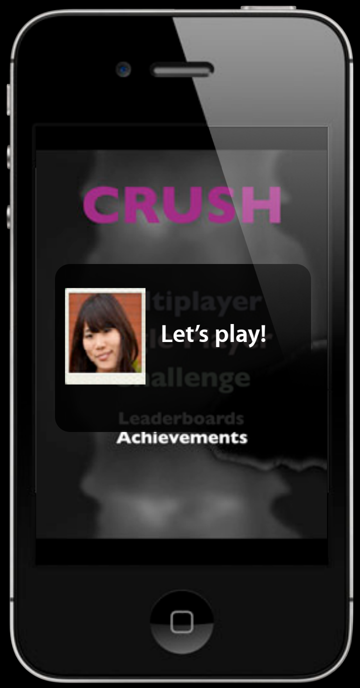
Steps for handling events

- Set up the delegate
- Override event methods
- Retrieve issuer info
  - Name
  - Photo
- Load any challenge data
  - GKScore, context
  - GKAchievement
- Present your custom UI

Handling Selected Challenges

Steps for handling events

- Set up the delegate
- Override event methods
- Retrieve issuer info
  - Name
  - Photo
- Load any challenge data
  - GKScore, context
  - GKAchievement
- Present your custom UI



Setting Up The Delegate

```
@interface MyAppDelegate <GKChallengeEventHandlerDelegate,
                          UIApplicationDelegate>
@end

@implementation MyAppDelegate

    - (void)application:(UIApplication *)application
            didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
    {
        // ...
        GKChallengeEventHandler *eventHandler = [GKChallengeEventHandler
            challengeEventHandler];
        eventHandler.delegate = self;
    }

@end
```

Setting Up The Delegate

```
@interface MyAppDelegate <GKChallengeEventHandlerDelegate,
                          UIApplicationDelegate>
@end

@implementation MyAppDelegate

    - (void)application:(UIApplication *)application
            didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
    {
        // ...
        GKChallengeEventHandler *eventHandler = [GKChallengeEventHandler
            challengeEventHandler];
        eventHandler.delegate = self;
    }

@end
```

Setting Up The Delegate

```
@interface MyAppDelegate <GKChallengeEventHandlerDelegate,
                          UIApplicationDelegate>
@end

@implementation MyAppDelegate

    - (void)application:(UIApplication *)application
            didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
    {
        // ...
        GKChallengeEventHandler *eventHandler = [GKChallengeEventHandler
            challengeEventHandler];
        eventHandler.delegate = self;
    }

@end
```

Setting Up The Delegate

```
@interface MyAppDelegate <GKChallengeEventHandlerDelegate,
                          UIApplicationDelegate>
@end

@implementation MyAppDelegate

    - (void)application:(UIApplication *)application
            didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
    {
        // ...
        GKChallengeEventHandler *eventHandler = [GKChallengeEventHandler
            challengeEventHandler];
        eventHandler.delegate = self;
    }

@end
```

Setting Up The Delegate

```
@interface MyAppDelegate <GKChallengeEventHandlerDelegate,
                          UIApplicationDelegate>
@end

@implementation MyAppDelegate

    - (void)application:(UIApplication *)application
            didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
    {
        // ...
        GKChallengeEventHandler *eventHandler = [GKChallengeEventHandler
            challengeEventHandler];
        eventHandler.delegate = self;
    }

@end
```

Setting Up The Delegate

```
@interface MyAppDelegate <GKChallengeEventHandlerDelegate,
                          UIApplicationDelegate>
@end

@implementation MyAppDelegate

    - (void)application:(UIApplication *)application
            didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
    {
        // ...
        GKChallengeEventHandler *eventHandler = [GKChallengeEventHandler
            challengeEventHandler];
        eventHandler.delegate = self;
    }

@end
```

Override Event Methods

```
- (void)localPlayerDidSelectChallenge:(GKChallenge *)challenge
{
    // load the issuing player
    NSString *issuerID = challenge.issuingPlayerID;
    [GKPlayer loadPlayersForIdentifiers:@[issuerID]
        withCompletionHandler:^(NSArray *players, NSError *error) {
        GKPlayer *player = [players lastObject];
        // load the photo
        [player loadPhotoForSize:GKPhotoSizeNormal
            withCompletionHandler:^(UIImage *photo, NSError *error) {
            // load any additional data and present
            [self presentChallenge:challenge photo:photo
                name:player.displayName];
            [self loadDataForChallenge:challenge];
        }];
    }];
}
```

Override Event Methods

```
- (void)localPlayerDidSelectChallenge:(GKChallenge *)challenge
{
    // load the issuing player
    NSString *issuerID = challenge.issuingPlayerID;
    [GKPlayer loadPlayersForIdentifiers:@[issuerID]
        withCompletionHandler:^(NSArray *players, NSError *error) {
        GKPlayer *player = [players lastObject];
        // load the photo
        [player loadPhotoForSize:GKPhotoSizeNormal
            withCompletionHandler:^(UIImage *photo, NSError *error) {
            // load any additional data and present
            [self presentChallenge:challenge photo:photo
                name:player.displayName];
            [self loadDataForChallenge:challenge];
        }];
    }];
}
```

Override Event Methods

```
- (void)localPlayerDidSelectChallenge:(GKChallenge *)challenge
{
    // load the issuing player
    NSString *issuerID = challenge.issuingPlayerID;
    [GKPlayer loadPlayersForIdentifiers:@[issuerID]
        withCompletionHandler:^(NSArray *players, NSError *error) {
        GKPlayer *player = [players lastObject];
        // load the photo
        [player loadPhotoForSize:GKPhotoSizeNormal
            withCompletionHandler:^(UIImage *photo, NSError *error) {
            // load any additional data and present
            [self presentChallenge:challenge photo:photo
                name:player.displayName];
            [self loadDataForChallenge:challenge];
        }];
    }];
}
```

Override Event Methods

```
- (void)localPlayerDidSelectChallenge:(GKChallenge *)challenge
{
    // load the issuing player
    NSString *issuerID = challenge.issuingPlayerID;
    [GKPlayer loadPlayersForIdentifiers:@[issuerID]
        withCompletionHandler:^(NSArray *players, NSError *error) {
        GKPlayer *player = [players lastObject];
        // load the photo
        [player loadPhotoForSize:GKPhotoSizeNormal
            withCompletionHandler:^(UIImage *photo, NSError *error) {
            // load any additional data and present
            [self presentChallenge:challenge photo:photo
                name:player.displayName];
            [self loadDataForChallenge:challenge];
        }];
    }];
}
```

Override Event Methods

```
- (void)localPlayerDidSelectChallenge:(GKChallenge *)challenge
{
    // load the issuing player
    NSString *issuerID = challenge.issuingPlayerID;
    [GKPlayer loadPlayersForIdentifiers:@[issuerID]
        withCompletionHandler:^(NSArray *players, NSError *error) {
        GKPlayer *player = [players lastObject];
        // load the photo
        [player loadPhotoForSize:GKPhotoSizeNormal
            withCompletionHandler:^(UIImage *photo, NSError *error) {
            // load any additional data and present
            [self presentChallenge:challenge photo:photo
                name:player.displayName];
            [self loadDataForChallenge:challenge];
        }];
    }];
}
```

Override Event Methods

```
- (void)localPlayerDidSelectChallenge:(GKChallenge *)challenge
{
    // load the issuing player
    NSString *issuerID = challenge.issuingPlayerID;
    [GKPlayer loadPlayersForIdentifiers:@[issuerID]
        withCompletionHandler:^(NSArray *players, NSError *error) {
        GKPlayer *player = [players lastObject];
        // load the photo
        [player loadPhotoForSize:GKPhotoSizeNormal
            withCompletionHandler:^(UIImage *photo, NSError *error) {
            // load any additional data and present
            [self presentChallenge:challenge photo:photo
                name:player.displayName];
            [self loadDataForChallenge:challenge];
        }];
    }];
}
```

Override Event Methods

```
- (void)localPlayerDidSelectChallenge:(GKChallenge *)challenge
{
    // load the issuing player
    NSString *issuerID = challenge.issuingPlayerID;
    [GKPlayer loadPlayersForIdentifiers:@[issuerID]
        withCompletionHandler:^(NSArray *players, NSError *error) {
        GKPlayer *player = [players lastObject];
        // load the photo
        [player loadPhotoForSize:GKPhotoSizeNormal
            withCompletionHandler:^(UIImage *photo, NSError *error) {
            // load any additional data and present
            [self presentChallenge:challenge photo:photo
                name:player.displayName];
            [self loadDataForChallenge:challenge];
        }];
    }];
}
```

Override Event Methods

```
- (void)localPlayerDidSelectChallenge:(GKChallenge *)challenge
{
    // load the issuing player
    NSString *issuerID = challenge.issuingPlayerID;
    [GKPlayer loadPlayersForIdentifiers:@[issuerID]
        withCompletionHandler:^(NSArray *players, NSError *error) {
        GKPlayer *player = [players lastObject];
        // load the photo
        [player loadPhotoForSize:GKPhotoSizeNormal
            withCompletionHandler:^(UIImage *photo, NSError *error) {
            // load any additional data and present
            [self presentChallenge:challenge photo:photo
                name:player.displayName];
            [self loadDataForChallenge:challenge];
        }];
    }];
}
```

Override Event Methods

```
- (void)localPlayerDidSelectChallenge:(GKChallenge *)challenge
{
    // load the issuing player
    NSString *issuerID = challenge.issuingPlayerID;
    [GKPlayer loadPlayersForIdentifiers:@[issuerID]
        withCompletionHandler:^(NSArray *players, NSError *error) {
        GKPlayer *player = [players lastObject];
        // load the photo
        [player loadPhotoForSize:GKPhotoSizeNormal
            withCompletionHandler:^(UIImage *photo, NSError *error) {
            // load any additional data and present
            [self presentChallenge:challenge photo:photo
                name:player.displayName];
            [self loadDataForChallenge:challenge];
        }];
    }];
}
```

Override Event Methods

```
- (void)localPlayerDidSelectChallenge:(GKChallenge *)challenge
{
    // load the issuing player
    NSString *issuerID = challenge.issuingPlayerID;
    [GKPlayer loadPlayersForIdentifiers:@[issuerID]
        withCompletionHandler:^(NSArray *players, NSError *error) {
        GKPlayer *player = [players lastObject];
        // load the photo
        [player loadPhotoForSize:GKPhotoSizeNormal
            withCompletionHandler:^(UIImage *photo, NSError *error) {
            // load any additional data and present
            [self presentChallenge:challenge photo:photo
                name:player.displayName];
            [self loadDataForChallenge:challenge];
        }];
    }];
}
```

Loading Challenge Data

```
- (void)loadDataForChallenge:(GKChallenge *)challenge
{









}
```

Loading Challenge Data

Score Challenge example

```
- (void)loadDataForChallenge:(GKChallenge *)challenge
{
    if ([challenge isKindOfClass:[GKScoreChallenge class]]) {
        // get the score from the score challenge
        GKScoreChallenge *scoreChallenge = (GKScoreChallenge *)challenge;
        GKScore *score = scoreChallenge.score;

        // load challenging player’s playthrough
        NSURL *ghostDataURL = [self decodeGhostURL:score.context];
        [self loadGhostDataForURL:ghostDataURL];
    }
}
```

Loading Challenge Data

Score Challenge example

```
- (void)loadDataForChallenge:(GKChallenge *)challenge
{
    if ([challenge isKindOfClass:[GKScoreChallenge class]]) {
        // get the score from the score challenge
        GKScoreChallenge *scoreChallenge = (GKScoreChallenge *)challenge;
        GKScore *score = scoreChallenge.score;

        // load challenging player's playthrough
        NSURL *ghostDataURL = [self decodeGhostURL:score.context];
        [self loadGhostDataForURL:ghostDataURL];
    }
}
```

Loading Challenge Data

Score Challenge example

```
- (void)loadDataForChallenge:(GKChallenge *)challenge
{
    if ([challenge isKindOfClass:[GKScoreChallenge class]]) {
        // get the score from the score challenge
        GKScoreChallenge *scoreChallenge = (GKScoreChallenge *)challenge;
        GKScore *score = scoreChallenge.score;

        // load challenging player's playthrough
        NSURL *ghostDataURL = [self decodeGhostURL:score.context];
        [self loadGhostDataForURL:ghostDataURL];
    }
}
```

Loading Challenge Data

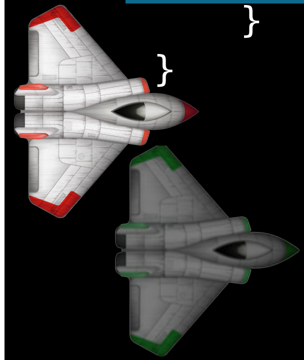
Score Challenge example

```
- (void)loadDataForChallenge:(GKChallenge *)challenge
{
    if ([challenge isKindOfClass:[GKScoreChallenge class]]) {
        // get the score from the score challenge
        GKScoreChallenge *scoreChallenge = (GKScoreChallenge *)challenge;
        GKScore *score = scoreChallenge.score;

        // load challenging player's playthrough
        NSURL *ghostDataURL = [self decodeGhostURL:score.context];
        [self loadGhostDataForURL:ghostDataURL];
    }
}
```

Loading Challenge Data

Score Challenge example

```
- (void)loadDataForChallenge:(GKChallenge *)challenge
{
    if ([challenge isKindOfClass:[GKScoreChallenge class]]) {
        // get the score from the score challenge
        GKScoreChallenge *scoreChallenge = (GKScoreChallenge *)challenge;
        GKScore *score = scoreChallenge.score;

        // load challenging player's playthrough
        NSURL *ghostDataURL = [self decodeGhostURL:score.context];
        [self loadGhostDataForURL:ghostDataURL];
    }
}
```



Loading Challenge Data

Score Challenge example

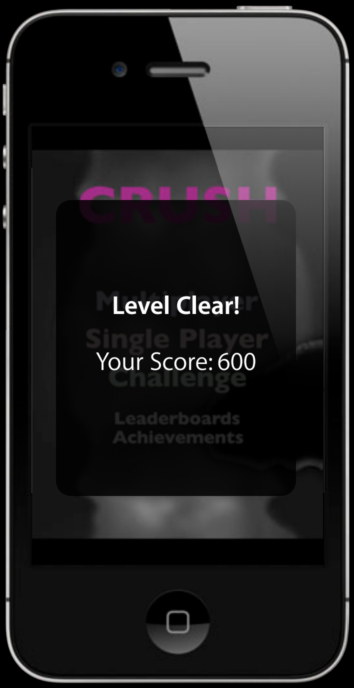
```
- (void)loadDataForChallenge:(GKChallenge *)challenge
{
    if ([challenge isKindOfClass:[GKScoreChallenge class]]) {
        // get the score from the score challenge
        GKScoreChallenge *scoreChallenge = (GKScoreChallenge *)challenge;
        GKScore *score = scoreChallenge.score;

        // load challenging player’s playthrough
        NSURL *ghostDataURL = [self decodeGhostURL:score.context];
        [self loadGhostDataForURL:ghostDataURL];
    }
}
```

Completed Challenges

New

Completed Challenges

New

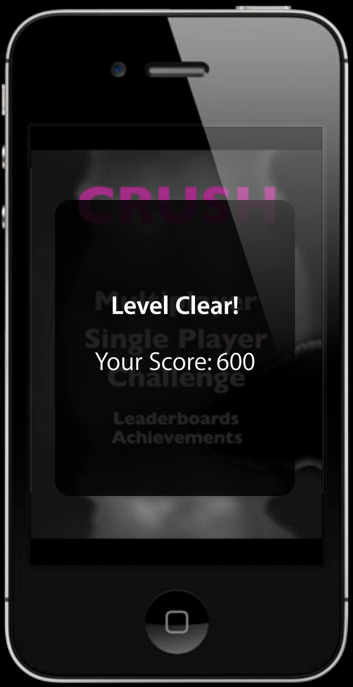


Level Clear!

Your Score: 600

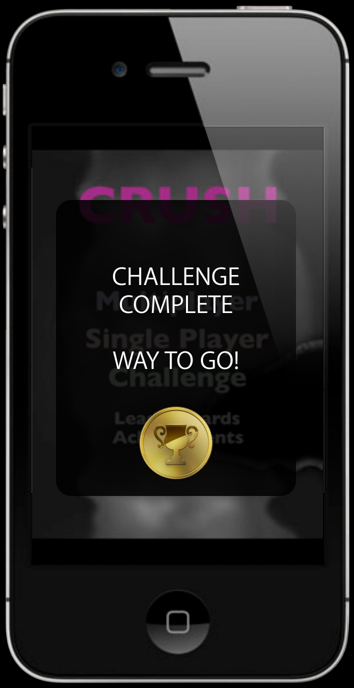
Completed Challenges

New



Completed Challenges

New



Handling Challenge Completion

```
- (void) localPlayerDidCompleteChallenge:(GKChallenge *)challenge
{
    UIImage *imageForChallenge;
    // lookup the leaderboard's icon
    if ([challenge isKindOfClass:[GKScoreChallenge class]]) {
        GKScoreChallenge *sc = (GKScoreChallenge *)challenge;
        imageForChallenge = [self getLeaderboardIconForScore:sc.score];
    }
   // lookup the achievement's icon
    else if ([challenge isKindOfClass:[GKAchievementChallenge class]]) {
        GKAchievementChallenge *ac = (GKAchievementChallenge *)challenge;
        imageForChallenge = [self getIconForAchievement:ac.achievement];
    }

    [self showCompletedChallengeUI:challenge image:imageForChallenge];
}
```

Handling Challenge Completion

```
- (void) localPlayerDidCompleteChallenge:(GKChallenge *)challenge
{
    UIImage *imageForChallenge;
    // lookup the leaderboard's icon
    if ([challenge isKindOfClass:[GKScoreChallenge class]]) {
        GKScoreChallenge *sc = (GKScoreChallenge *)challenge;
        imageForChallenge = [self getLeaderboardIconForScore:sc.score];
    }
   // lookup the achievement's icon
    else if ([challenge isKindOfClass:[GKAchievementChallenge class]]) {
        GKAchievementChallenge *ac = (GKAchievementChallenge *)challenge;
        imageForChallenge = [self getIconForAchievement:ac.achievement];
    }

    [self showCompletedChallengeUI:challenge image:imageForChallenge];
}
```

Handling Challenge Completion

```
- (void) localPlayerDidCompleteChallenge:(GKChallenge *)challenge
{
    UIImage *imageForChallenge;
    // lookup the leaderboard's icon
    if ([challenge isKindOfClass:[GKScoreChallenge class]]) {
        GKScoreChallenge *sc = (GKScoreChallenge *)challenge;
        imageForChallenge = [self getLeaderboardIconForScore:sc.score];
    }
   // lookup the achievement's icon
    else if ([challenge isKindOfClass:[GKAchievementChallenge class]]) {
        GKAchievementChallenge *ac = (GKAchievementChallenge *)challenge;
        imageForChallenge = [self getIconForAchievement:ac.achievement];
    }

    [self showCompletedChallengeUI:challenge image:imageForChallenge];
}
```

Handling Challenge Completion

```
- (void) localPlayerDidCompleteChallenge:(GKChallenge *)challenge
{
    UIImage *imageForChallenge;
    // lookup the leaderboard's icon
    if ([challenge isKindOfClass:[GKScoreChallenge class]]) {
        GKScoreChallenge *sc = (GKScoreChallenge *)challenge;
        imageForChallenge = [self getLeaderboardIconForScore:sc.score];
    }
   // lookup the achievement's icon
    else if ([challenge isKindOfClass:[GKAchievementChallenge class]]) {
        GKAchievementChallenge *ac = (GKAchievementChallenge *)challenge;
        imageForChallenge = [self getIconForAchievement:ac.achievement];
    }

    [self showCompletedChallengeUI:challenge image:imageForChallenge];
}
```

Handling Challenge Completion

```
- (void) localPlayerDidCompleteChallenge:(GKChallenge *)challenge
{
    UIImage *imageForChallenge;
    // lookup the leaderboard’s icon
    if ([challenge isKindOfClass:[GKScoreChallenge class]]) {
        GKScoreChallenge *sc = (GKScoreChallenge *)challenge;
        imageForChallenge = [self getLeaderboardIconForScore:sc.score];
    }
   // lookup the achievement’s icon
    else if ([challenge isKindOfClass:[GKAchievementChallenge class]]) {
        GKAchievementChallenge *ac = (GKAchievementChallenge *)challenge;
        imageForChallenge = [self getIconForAchievement:ac.achievement];
    }

    [self showCompletedChallengeUI:challenge image:imageForChallenge];
}
```

Handling Challenge Completion

```
- (void) localPlayerDidCompleteChallenge:(GKChallenge *)challenge
{
    UIImage *imageForChallenge;
    // lookup the leaderboard's icon
    if ([challenge isKindOfClass:[GKScoreChallenge class]]) {
        GKScoreChallenge *sc = (GKScoreChallenge *)challenge;
        imageForChallenge = [self getLeaderboardIconForScore:sc.score];
    }
   // lookup the achievement's icon
    else if ([challenge isKindOfClass:[GKAchievementChallenge class]]) {
        GKAchievementChallenge *ac = (GKAchievementChallenge *)challenge;
        imageForChallenge = [self getIconForAchievement:ac.achievement];
    }

    [self showCompletedChallengeUI:challenge image:imageForChallenge];
}
```

Handling Challenge Completion

```
- (void) localPlayerDidCompleteChallenge:(GKChallenge *)challenge
{
    UIImage *imageForChallenge;
    // lookup the leaderboard's icon
    if ([challenge isKindOfClass:[GKScoreChallenge class]]) {
        GKScoreChallenge *sc = (GKScoreChallenge *)challenge;
        imageForChallenge = [self getLeaderboardIconForScore:sc.score];
    }
   // lookup the achievement's icon
    else if ([challenge isKindOfClass:[GKAchievementChallenge class]]) {
        GKAchievementChallenge *ac = (GKAchievementChallenge *)challenge;
        imageForChallenge = [self getIconForAchievement:ac.achievement];
    }

    [self showCompletedChallengeUI:challenge image:imageForChallenge];
}
```

Handling Challenge Completion

```
- (void) localPlayerDidCompleteChallenge:(GKChallenge *)challenge
{
    UIImage *imageForChallenge;
    // lookup the leaderboard's icon
    if ([challenge isKindOfClass:[GKScoreChallenge class]]) {
        GKScoreChallenge *sc = (GKScoreChallenge *)challenge;
        imageForChallenge = [self getLeaderboardIconForScore:sc.score];
    }
   // lookup the achievement's icon
    else if ([challenge isKindOfClass:[GKAchievementChallenge class]]) {
        GKAchievementChallenge *ac = (GKAchievementChallenge *)challenge;
        imageForChallenge = [self getIconForAchievement:ac.achievement];
    }

    [self showCompletedChallengeUI:challenge image:imageForChallenge];
}
```

More Challenge Events

```
localPlayerDidSelectChallenge:
localPlayerDidCompleteChallenge:
```

More Challenge Events

```
localPlayerDidSelectChallenge:
localPlayerDidCompleteChallenge:
localPlayerDidReceiveChallenge:
```

More Challenge Events

```
localPlayerDidSelectChallenge:
localPlayerDidCompleteChallenge:
localPlayerDidReceiveChallenge:
remotePlayerDidCompleteChallenge:
```

More Challenge Events

```
localPlayerDidSelectChallenge:
localPlayerDidCompleteChallenge:
localPlayerDidReceiveChallenge:
remotePlayerDidCompleteChallenge:
shouldShowBannerForLocallyReceivedChallenge:
```

More Challenge Events

```
localPlayerDidSelectChallenge:
localPlayerDidCompleteChallenge:
localPlayerDidReceiveChallenge:
remotePlayerDidCompleteChallenge:
shouldShowBannerForLocallyReceivedChallenge:
shouldShowBannerForLocallyCompletedChallenge:
```

More Challenge Events

```
localPlayerDidSelectChallenge:
localPlayerDidCompleteChallenge:
localPlayerDidReceiveChallenge:
remotePlayerDidCompleteChallenge:
shouldShowBannerForLocallyReceivedChallenge:
shouldShowBannerForLocallyCompletedChallenge:
shouldShowBannerForRemotelyCompletedChallenge:
```

Best Practices

```
// get friends and message from current player
NSArray *playersForChallenge = self.playersForChallenge;
NSString *message = self.messageForChallenge;

// set up the achievement
GKAchievement *myAchievement = [[GKAchievement alloc]
    initWithIdentifier:@”com.mystudio.runandgun”];
myAchievement.percentComplete = 100.0;

// issue the challenge
[myAchievement issueChallengeToPlayers:playersForChallenge
    message:message];
```

Best Practices

```
// get friends and message from current player
NSArray *playersForChallenge = self.playersForChallenge;
NSString *message = self.messageForChallenge;

// set up the achievement
GKAchievement *myAchievement = [[GKAchievement alloc]
    initWithIdentifier:@”com.mystudio.runandgun”];
myAchievement.percentComplete = 100.0;

// issue the challenge
[myAchievement issueChallengeToPlayers:playersForChallenge
    message:message];
```

Picking Players

```
GKLeaderboard *leaderboard = [[GKLeaderboard alloc] init];
leaderboard.category = @”com.mystudio.level1”;

leaderboard.playerScope = GKLeaderboardPlayerScopeFriendsOnly;
leaderboard.range = NSMakeRange(1, 100);

[leaderboard loadScoresWithCompletionHander:^(NSArray *scores,
                                              NSError *error) {
    // get any scores lower than current player’s score
    NSPredicate *filter = [NSPredicate predicateWithFormat:@”value < %qi”,
        scoreForLevel1];
    NSArray *lesserScores = [scores filteredArrayUsingPredicate:filter];

    // allow player to choose which friends to challenge
    [self presentFriendPickerWithPreselectedScores:lesserScores];
}];
```

Picking Players

```
GKLeaderboard *leaderboard = [[GKLeaderboard alloc] init];
leaderboard.category = @”com.mystudio.level1”;

leaderboard.playerScope = GKLeaderboardPlayerScopeFriendsOnly;
leaderboard.range = NSMakeRange(1, 100);

[leaderboard loadScoresWithCompletionHander:^(NSArray *scores,
                                               NSError *error) {
    // get any scores lower than current player’s score
    NSPredicate *filter = [NSPredicate predicateWithFormat:@”value < %qi”,
        scoreForLevel1];
    NSArray *lesserScores = [scores filteredArrayUsingPredicate:filter];

    // allow player to choose which friends to challenge
    [self presentFriendPickerWithPreselectedScores:lesserScores];
}];
```

Picking Players

```
GKLeaderboard *leaderboard = [[GKLeaderboard alloc] init];
leaderboard.category = @”com.mystudio.level1”;

leaderboard.playerScope = GKLeaderboardPlayerScopeFriendsOnly;
leaderboard.range = NSMakeRange(1, 100);

[leaderboard loadScoresWithCompletionHander:^(NSArray *scores,
                                              NSError *error) {
    // get any scores lower than current player’s score
    NSPredicate *filter = [NSPredicate predicateWithFormat:@”value < %qi”,
        scoreForLevel1];
    NSArray *lesserScores = [scores filteredArrayUsingPredicate:filter];

    // allow player to choose which friends to challenge
    [self presentFriendPickerWithPreselectedScores:lesserScores];
}];
```

Picking Players

```
GKLeaderboard *leaderboard = [[GKLeaderboard alloc] init];
leaderboard.category = @”com.mystudio.level1”;

leaderboard.playerScope = GKLeaderboardPlayerScopeFriendsOnly;
leaderboard.range = NSMakeRange(1, 100);

[leaderboard loadScoresWithCompletionHander:^(NSArray *scores,
                                              NSError *error) {
    // get any scores lower than current player’s score
    NSPredicate *filter = [NSPredicate predicateWithFormat:@”value < %qi”,
        scoreForLevel1];
    NSArray *lesserScores = [scores filteredArrayUsingPredicate:filter];

    // allow player to choose which friends to challenge
    [self presentFriendPickerWithPreselectedScores:lesserScores];
}];
```

Picking Players

```
GKLeaderboard *leaderboard = [[GKLeaderboard alloc] init];
leaderboard.category = @"com.mystudio.level1";

leaderboard.playerScope = GKLeaderboardPlayerScopeFriendsOnly;
leaderboard.range = NSMakeRange(1, 100);

[leaderboard loadScoresWithCompletionHander:^(NSArray *scores,
                                              NSError *error) {
    // get any scores lower than current player's score
    NSPredicate *filter = [NSPredicate predicateWithFormat:@"value < %qi",
        scoreForLevel1];
    NSArray *lesserScores = [scores filteredArrayUsingPredicate:filter];

    // allow player to choose which friends to challenge
    [self presentFriendPickerWithPreselectedScores:lesserScores];
}];
```

Picking Players

```
GKLeaderboard *leaderboard = [[GKLeaderboard alloc] init];
leaderboard.category = @”com.mystudio.level1”;

leaderboard.playerScope = GKLeaderboardPlayerScopeFriendsOnly;
leaderboard.range = NSMakeRange(1, 100);

[leaderboard loadScoresWithCompletionHander:^(NSArray *scores,
                                               NSError *error) {
    // get any scores lower than current player’s score
    NSPredicate *filter = [NSPredicate predicateWithFormat:@”value < %qi”,
        scoreForLevel1];
    NSArray *lesserScores = [scores filteredArrayUsingPredicate:filter];

    // allow player to choose which friends to challenge
    [self presentFriendPickerWithPreselectedScores:lesserScores];
}];
```

Picking Players

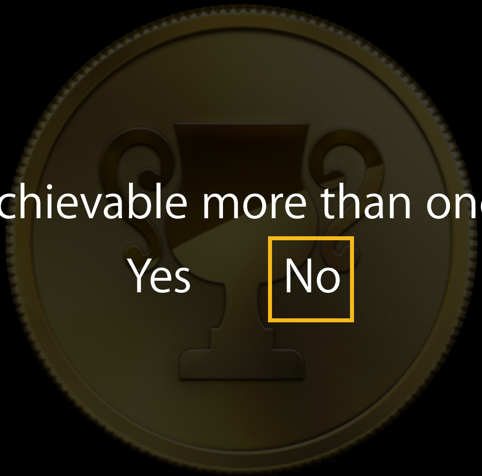
```
GKLeaderboard *leaderboard = [[GKLeaderboard alloc] init];
leaderboard.category = @"com.mystudio.level1";

leaderboard.playerScope = GKLeaderboardPlayerScopeFriendsOnly;
leaderboard.range = NSMakeRange(1, 100);

[leaderboard loadScoresWithCompletionHander:^(NSArray *scores,
                                              NSError *error) {
    // get any scores lower than current player's score
    NSPredicate *filter = [NSPredicate predicateWithFormat:@"value < %qi",
        scoreForLevel1];
    NSArray *lesserScores = [scores filteredArrayUsingPredicate:filter];

    // allow player to choose which friends to challenge
    [self presentFriendPickerWithPreselectedScores:lesserScores];
}];
```

Best Practices

Best Practices

Best Practices

Best Practices



Achievable more than once?

Yes [✓ No]

Best Practices

Achievable more than once?

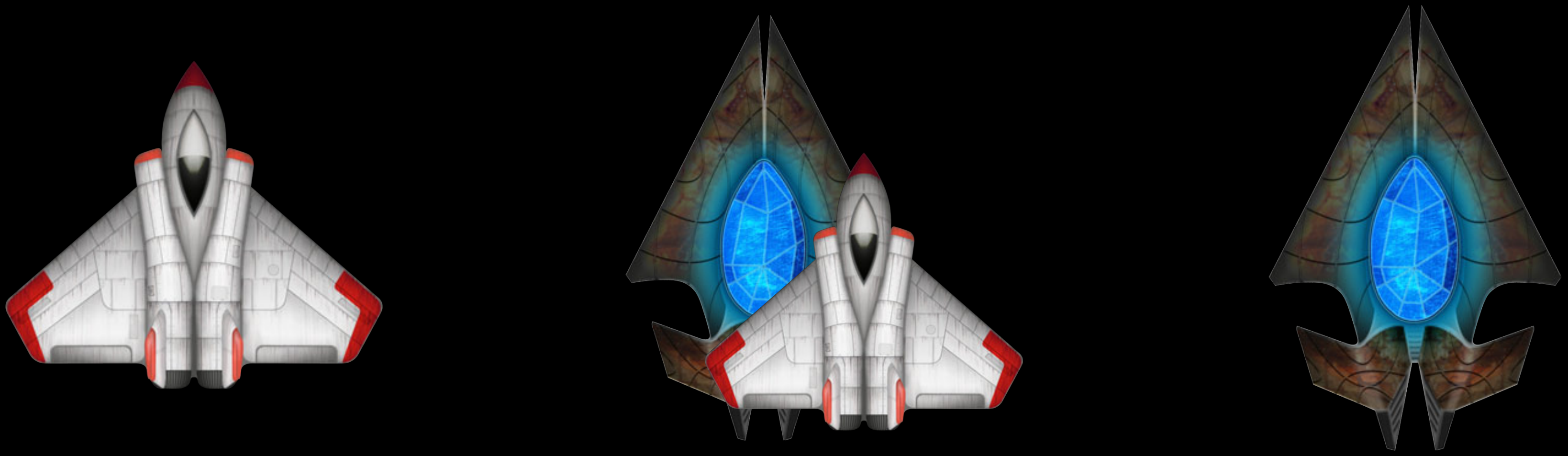
[✓ Yes] No

Summary

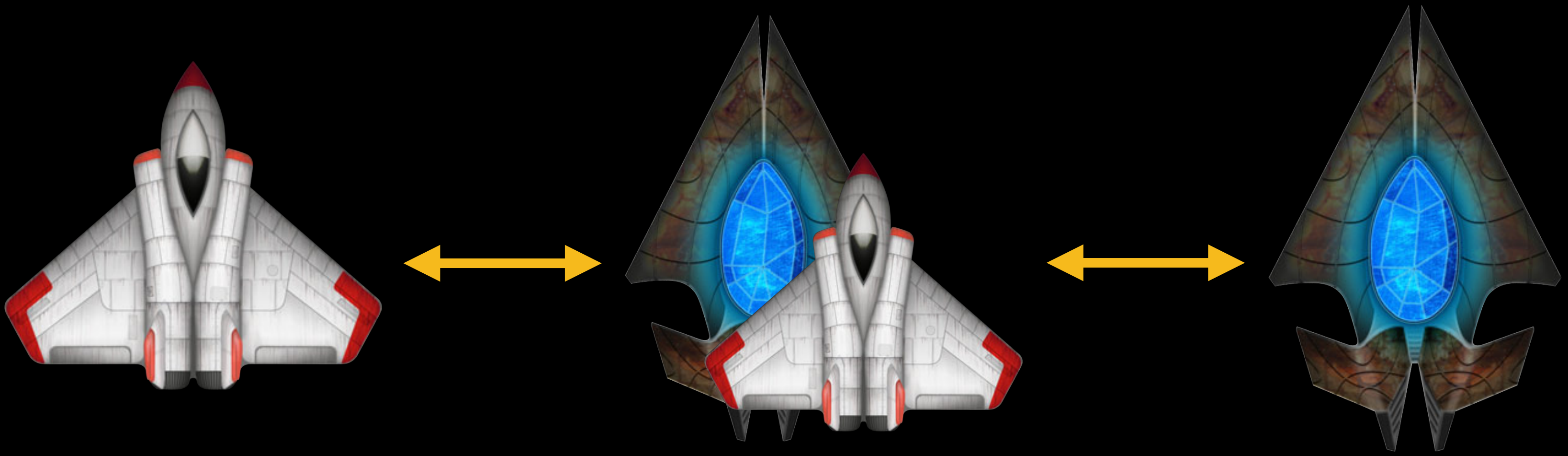
- Challenges: built-in
- API for extension
- GKScore context
- Give the issuer control
- Replayable = more fun

Different Games, Same Stats

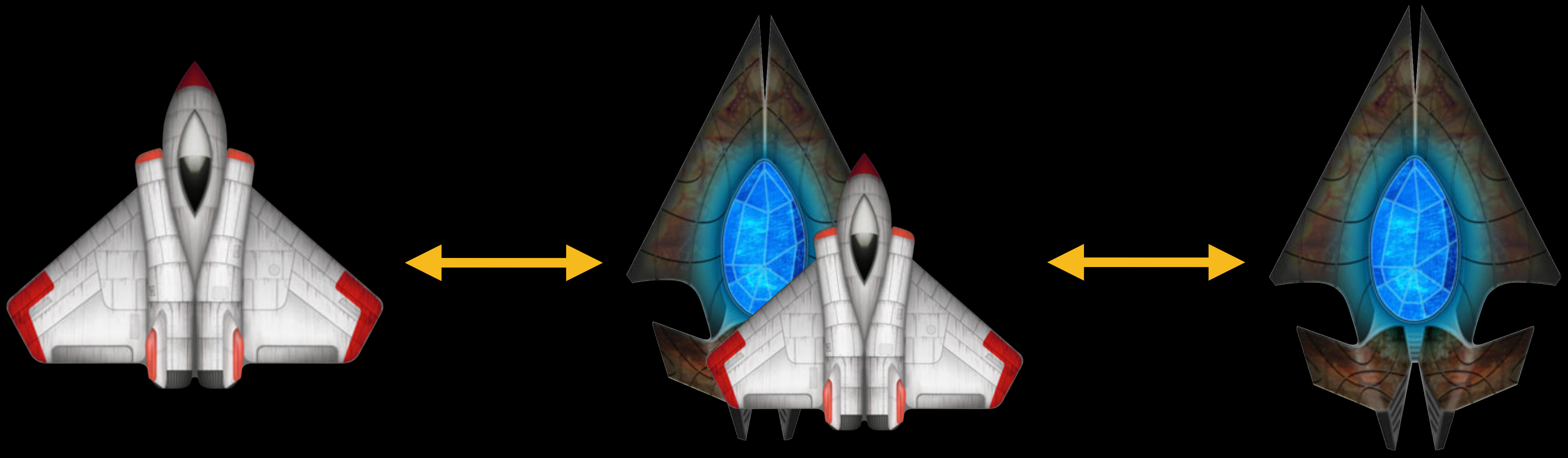
Different Games, Same Stats



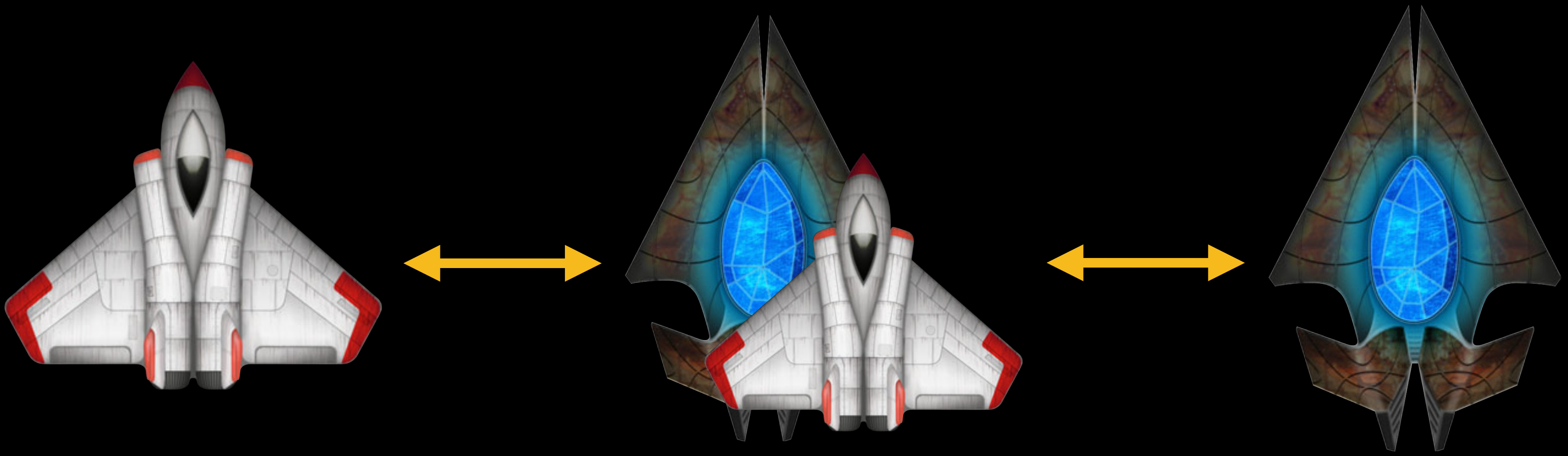
Different Games, Same Stats



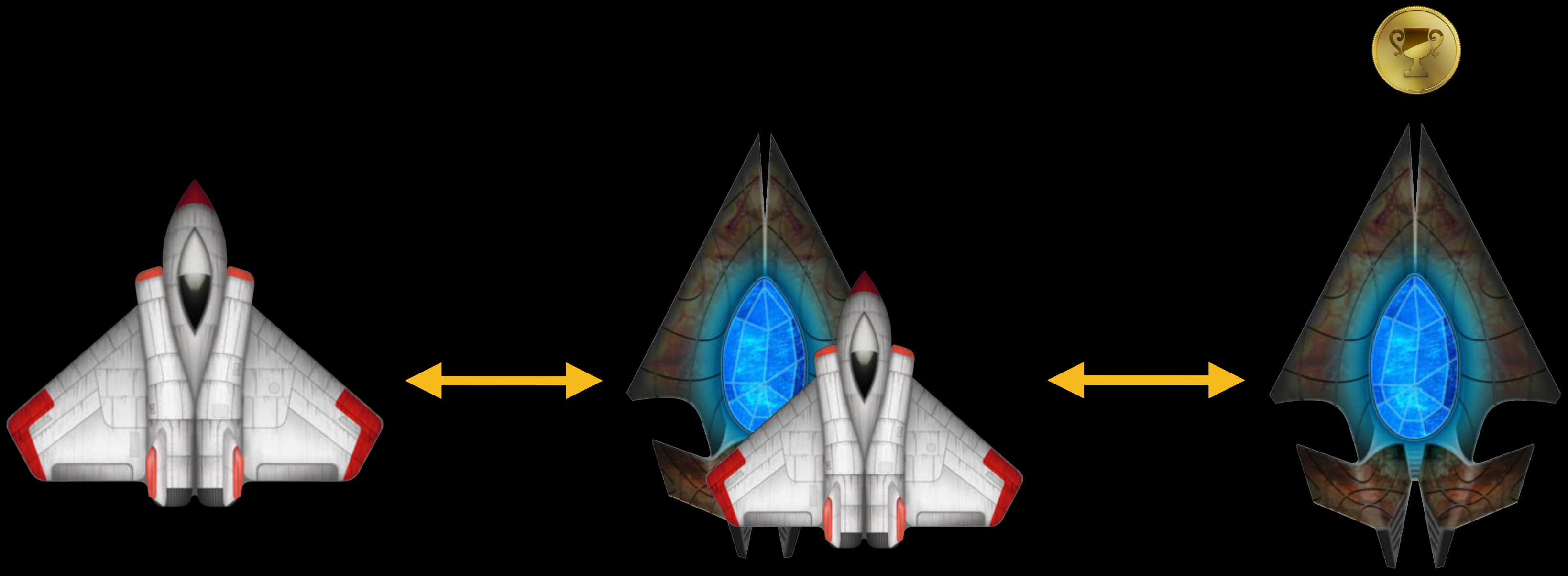
Different Games, Same Stats



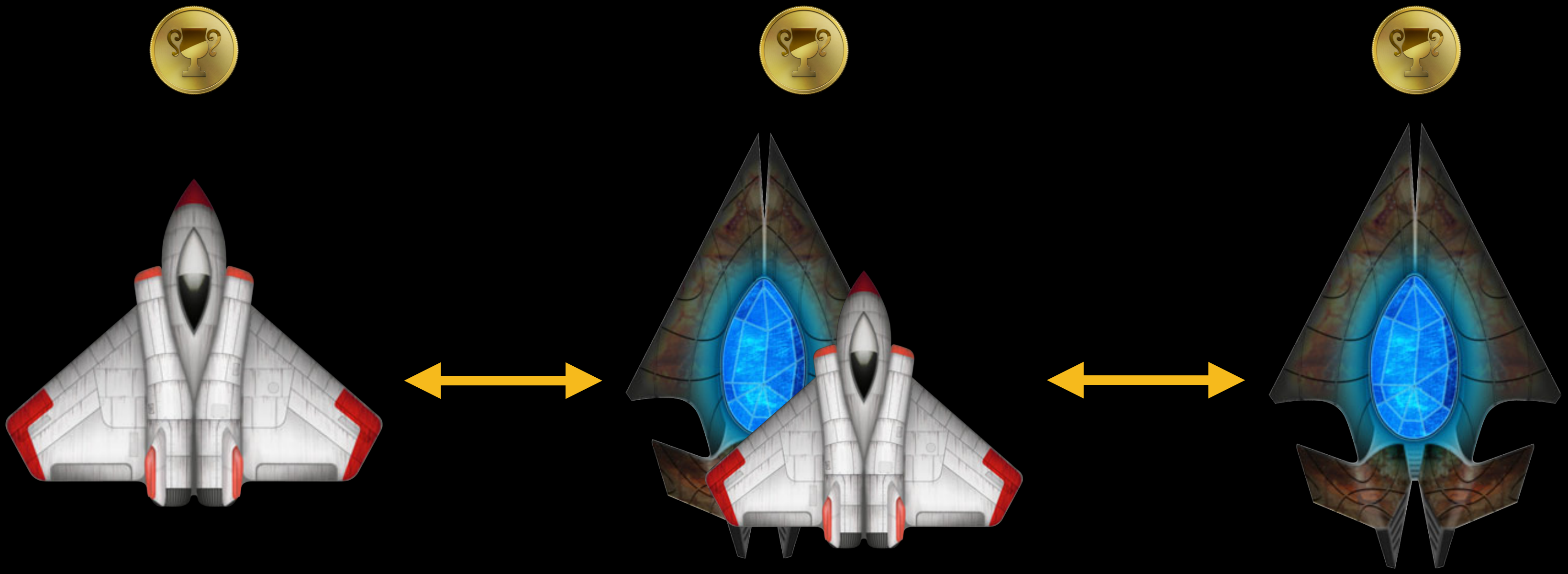
Different Games, Same Stats



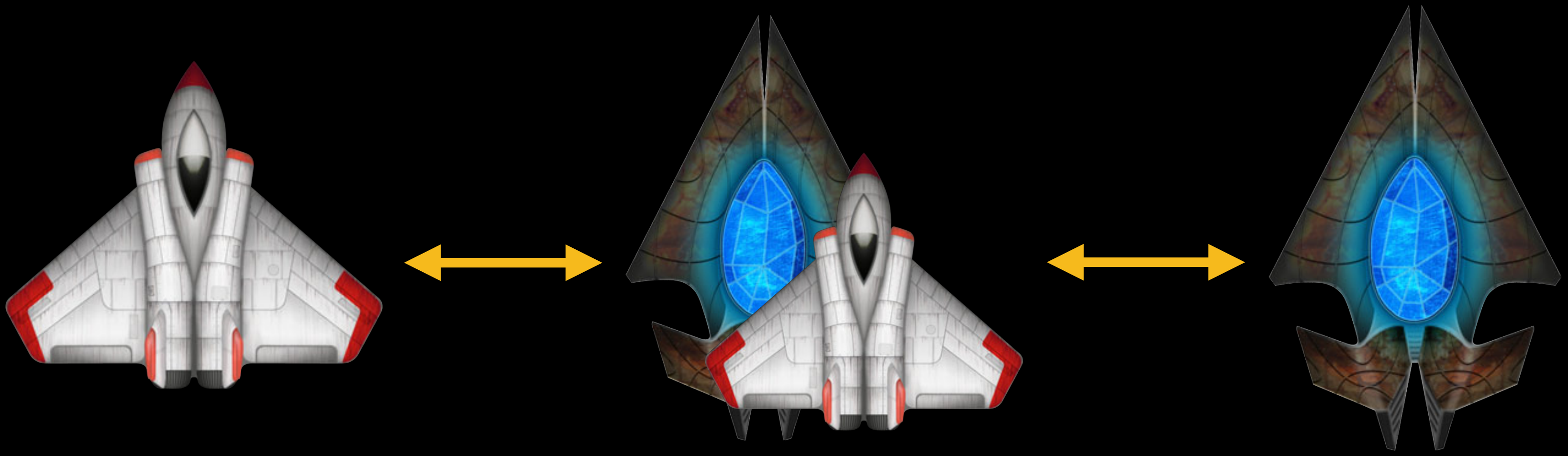
Different Games, Same Stats



Different Games, Same Stats



Different Games, Same Stats



Game Groups

Game Groups

New

- No code required

Game Groups

New

- No code required
- Group leaderboards

Game Groups

New

- No code required
- Group leaderboards
- Group achievements

Game Groups

New

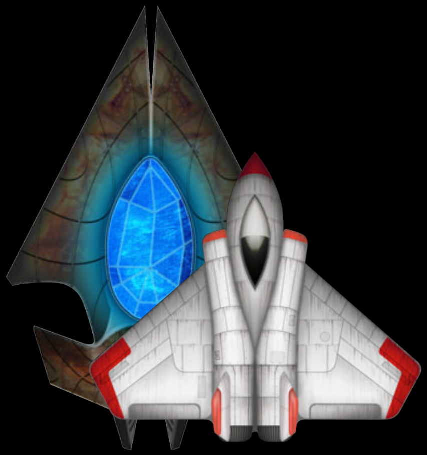
- No code required
- Group leaderboards
- Group achievements
- Beat challenges from any version

Game Groups

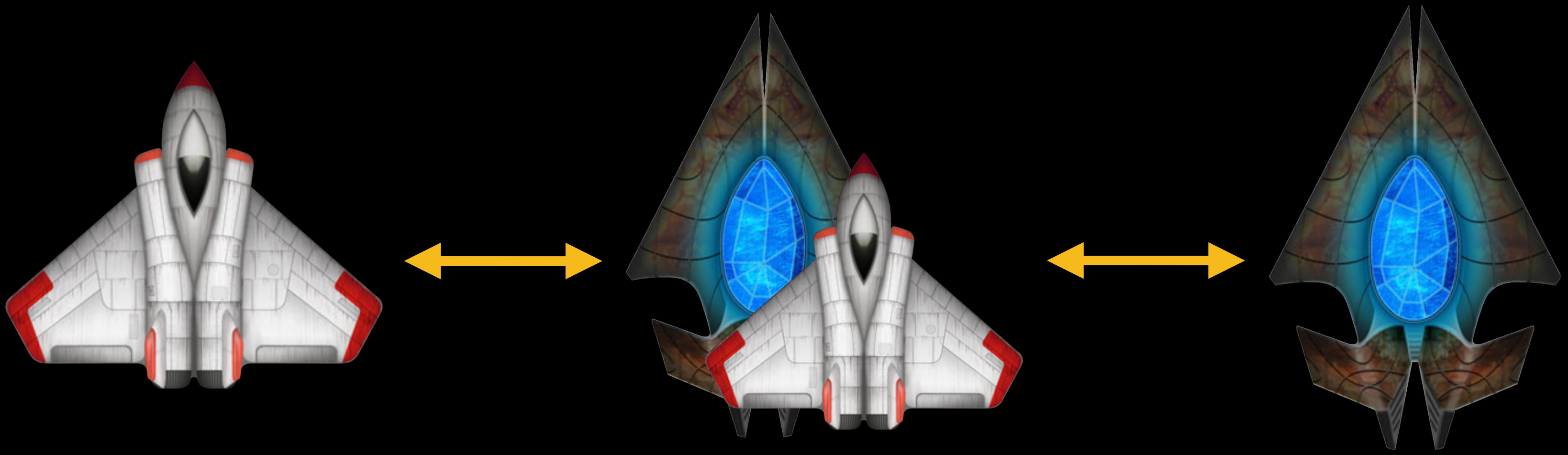
New

- No code required
- Group leaderboards
- Group achievements
- Beat challenges from any version
- Cross-version multiplayer

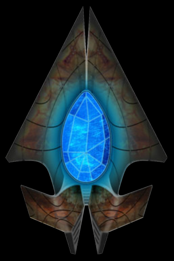
Grouping Games



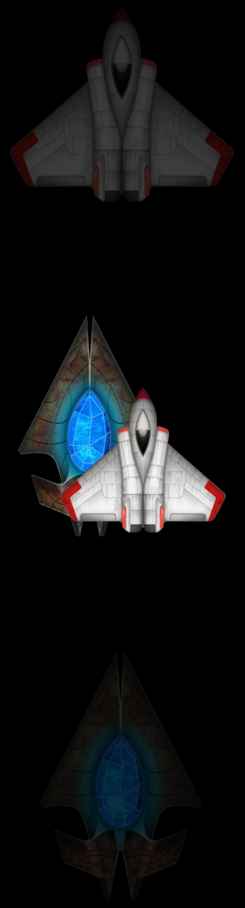
Grouping Games



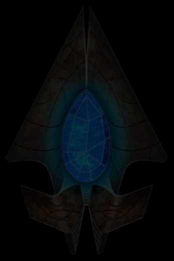
Grouping Games



Grouping Games



Grouping Games

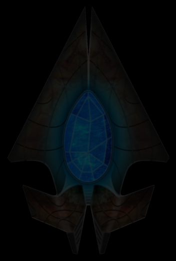


iTunes Connect

Grouping Games

iTunes Connect

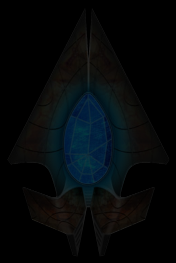
Move to a New Group:



Grouping Games

iTunes Connect

Move to a New Group:

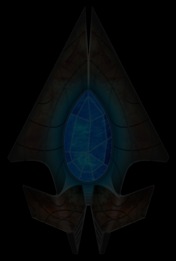


Grouping Games

iTunes Connect

Move to a New Group: Crush Group

Grouping Games

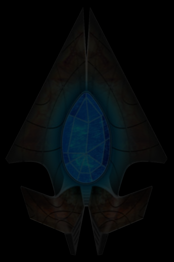


iTunes Connect

Grouping Games

iTunes Connect

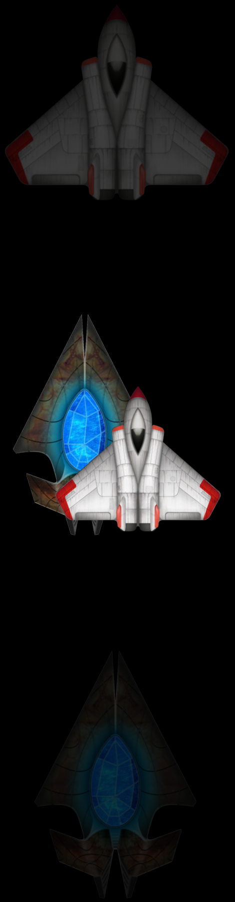
Grouping Games



iTunes Connect

	ID	New ID
	com.mystudio.level1	

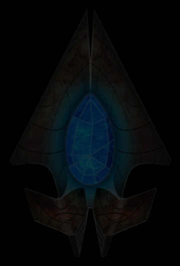
Grouping Games



iTunes Connect

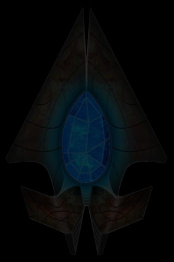
ID	New ID
com.mystudio.level1	grp.com.mystudio.level1

Grouping Games



iTunes Connect

Grouping Games



iTunes Connect

ID	New ID
com.mystudio.ach1	

Grouping Games

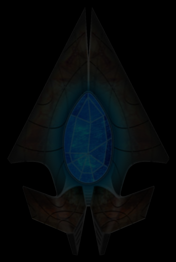
iTunes Connect

ID	New ID
com.mystudio.ach1	grp.com.mystudio.ach1

Grouping Games

iTunes Connect

Move to an Existing Group:



Grouping Games

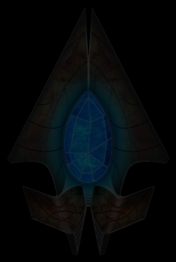
iTunes Connect

Move to an Existing Group: Crush Group

Grouping Games

iTunes Connect

Move to an Existing Group: **Crush Group**

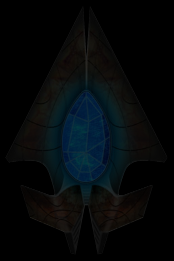


Grouping Games

iTunes Connect

ID	New ID
com.mystudio.level1	

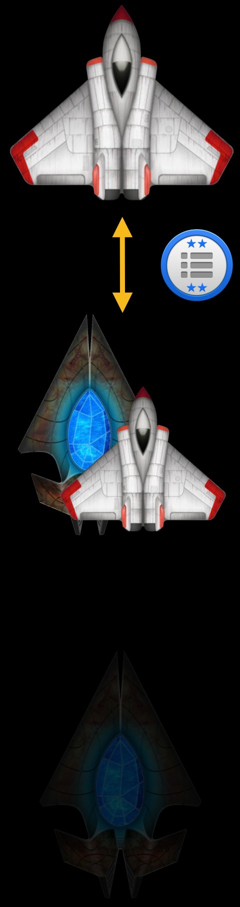
Grouping Games



iTunes Connect

	ID	New ID
☰	com.mystudio.level1	grp.com.mystudio.level1

Grouping Games



iTunes Connect

	ID	New ID
●	com.mystudio.level1	grp.com.mystudio.level1

Grouping Games

iTunes Connect

Move to an Existing Group:

Grouping Games

iTunes Connect

Move to an Existing Group: Crush Group

Grouping Games

iTunes Connect

Move to an Existing Group: **Crush Group**

Grouping Games

iTunes Connect

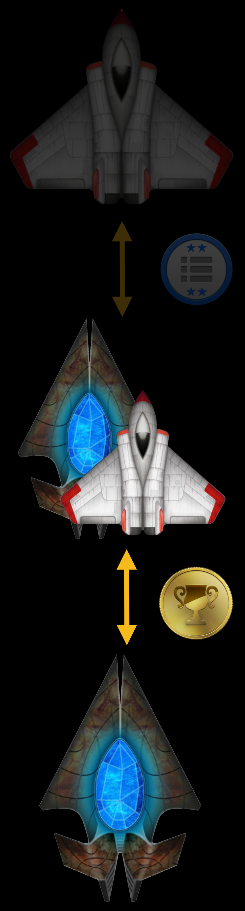
	ID	New ID
🏆	com.mystudio.ach1	

Grouping Games

iTunes Connect

ID	New ID
com.mystudio.ach1	grp.com.mystudio.ach1

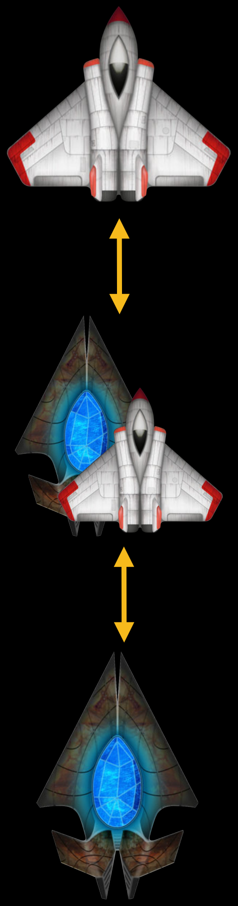
Grouping Games



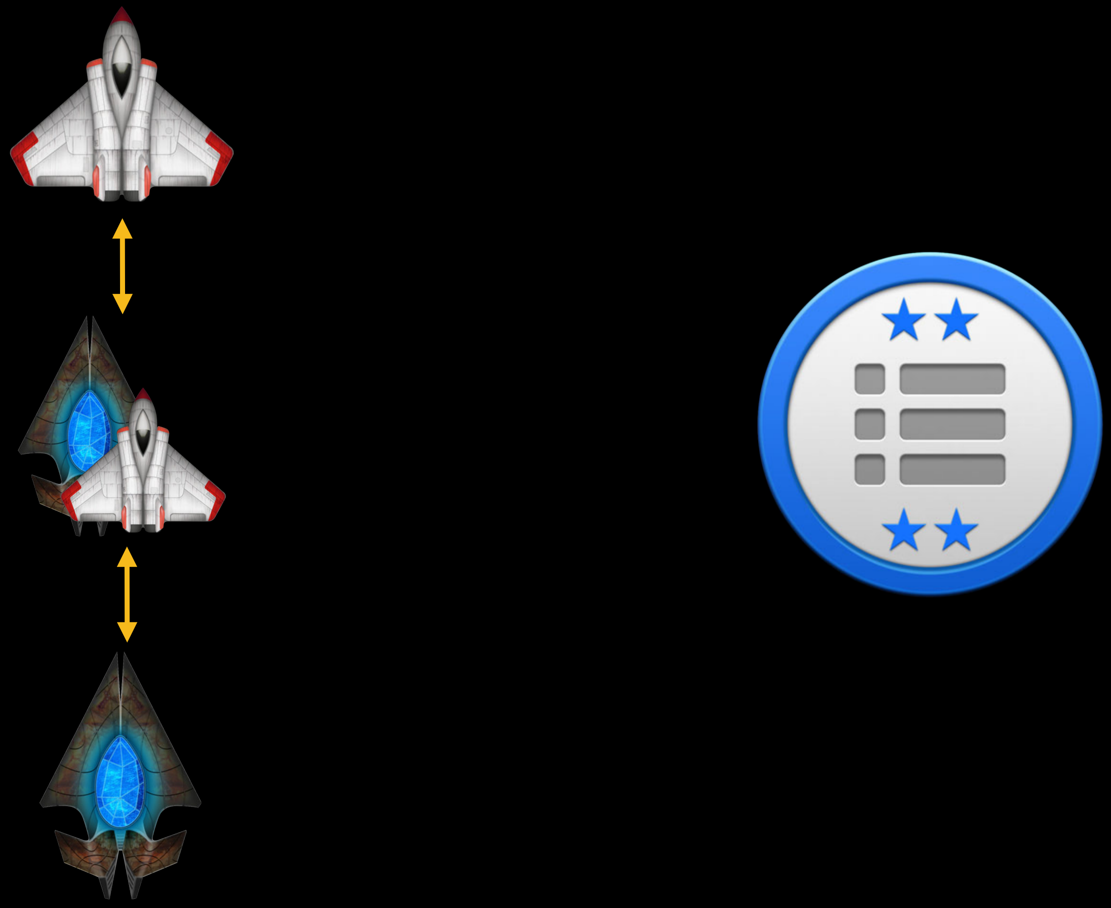
iTunes Connect

	ID	New ID
●	com.mystudio.ach1	grp.com.mystudio.ach1

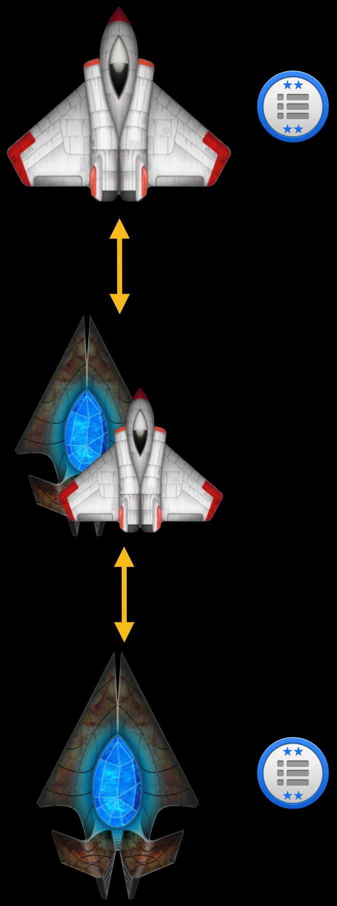
Grouping Games



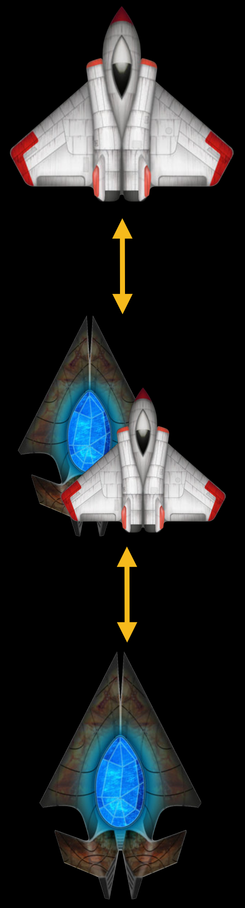
Grouping Games



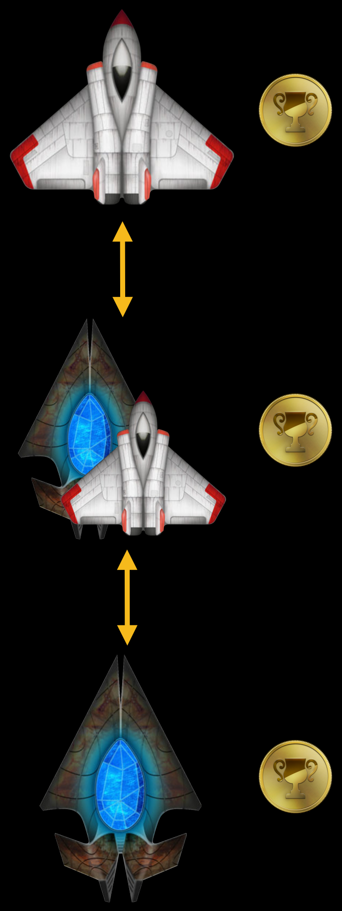
Grouping Games



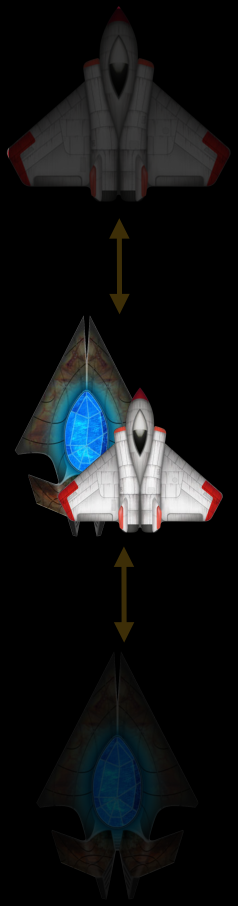
Grouping Games



Grouping Games

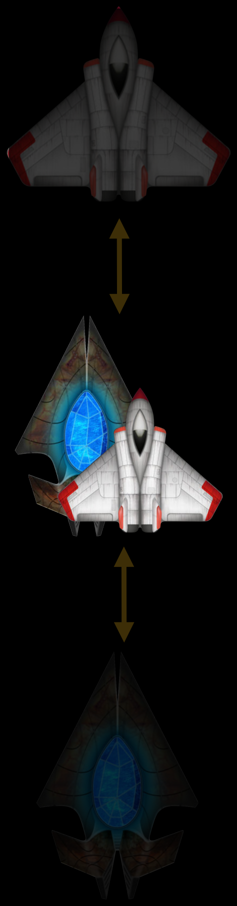


Cross-Version Multiplayer



iTunes Connect

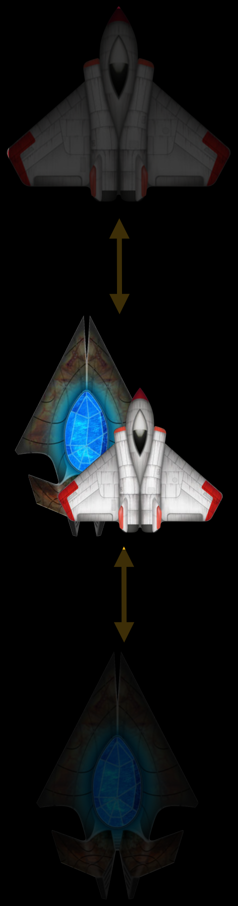
Cross-Version Multiplayer



iTunes Connect

View Details > Multiplayer Compatibility

Cross-Version Multiplayer



iTunes Connect

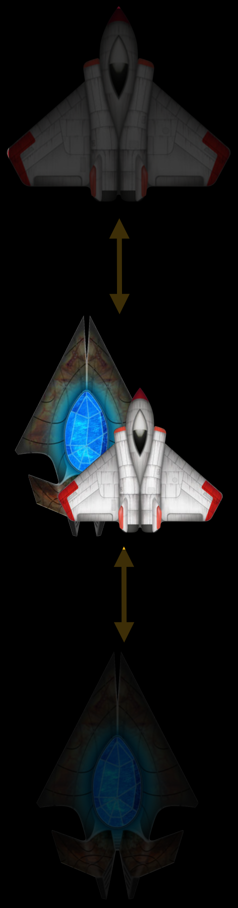
Crush! Lite	All versions	1.0	1.1	
Crush! HD	All versions	1.0	2.0	
Super Crush!	All versions	1.0	1.0.1	2.0

Cross-Version Multiplayer

iTunes Connect

Crush! Lite	All versions	1.0	$\boxed{1.1}$	
Crush! HD	All versions	1.0	2.0	
Super Crush!	All versions	1.0	1.0.1	2.0

Cross-Version Multiplayer



iTunes Connect

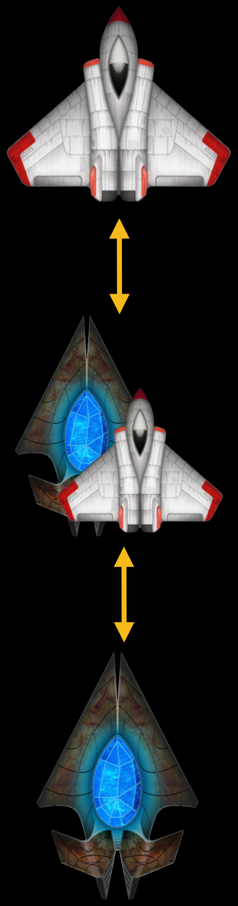
Crush! Lite	All versions	1.0	$\boxed{1.1}$	
Crush! HD	$\boxed{\text{All versions}}$	1.0	2.0	
Super Crush!	All versions	1.0	1.0.1	2.0

Cross-Version Multiplayer

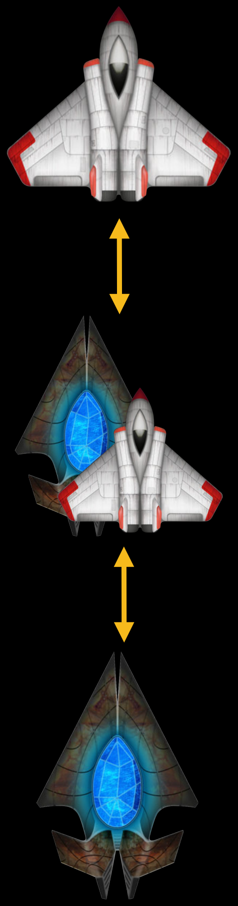
iTunes Connect

Crush! Lite	All versions	1.0	$\boxed{1.1}$	
Crush! HD	$\boxed{\text{All versions}}$	1.0	2.0	
Super Crush!	All versions	1.0	$\boxed{1.0.1}$	$\boxed{2.0}$

Summary

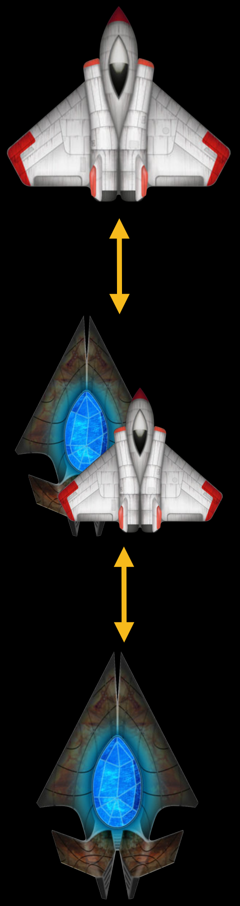


Summary



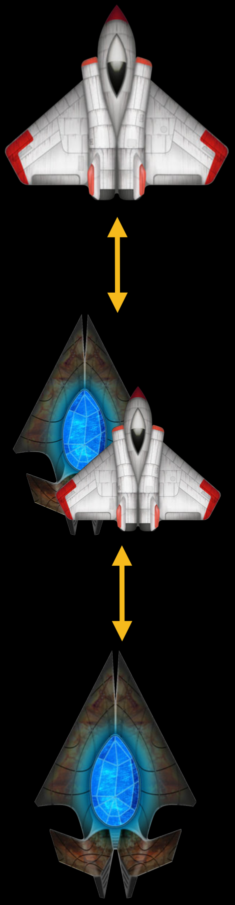
- Define a Group ID

Summary



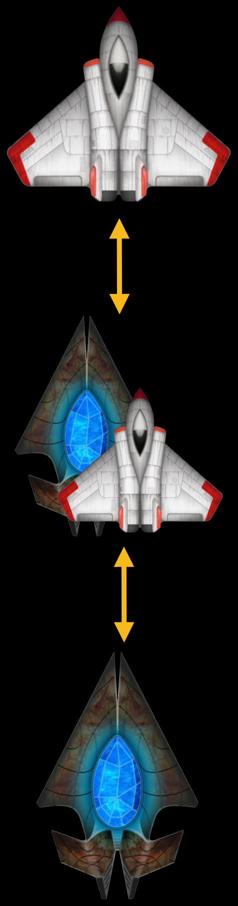
- Define a Group ID
- Convert leaderboards

Summary



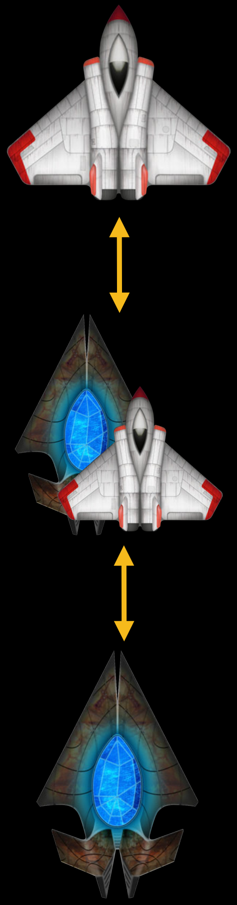
- Define a Group ID
- Convert leaderboards
- Convert achievements

Summary



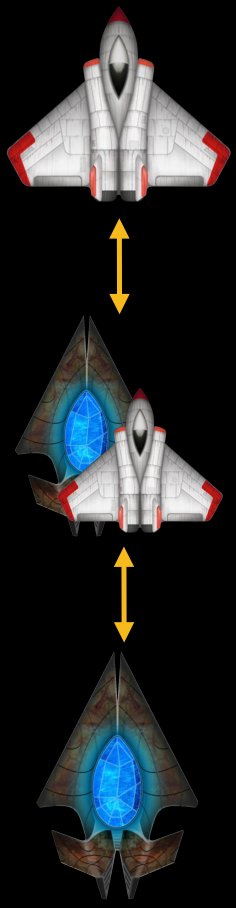
- Define a Group ID
- Convert leaderboards
- Convert achievements
- Add other games to group

Summary



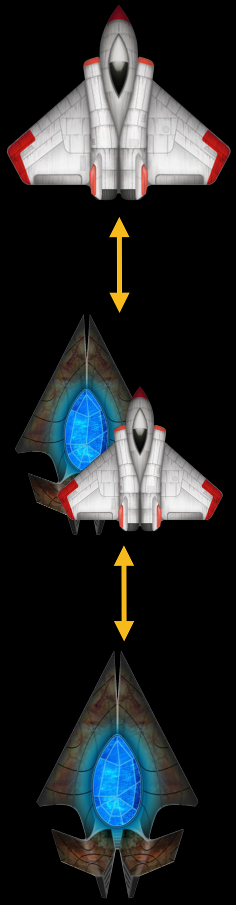
- Define a Group ID
- Convert leaderboards
- Convert achievements
- Add other games to group
- Merge

Summary



- Define a Group ID
- Convert leaderboards
- Convert achievements
- Add other games to group
- Merge
- Define new leaderboards and achievements

Summary



- Define a Group ID
- Convert leaderboards
- Convert achievements
- Add other games to group
- Merge
- Define new leaderboards and achievements
- Specify which games for multiplayer

What We’ve Learned

- Player authentication
- Scores
- Achievements
- Challenges
- Groups

More Information

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

Documentation
Game Center for Developers
http://developer.apple.com/devcenter/ios/gamecenter

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

Related Sessions

What’s New in iTunes Connect for App Developers	Nob Hill Thursday 9:00AM
Multiplayer Gaming with Game Center	Pacific Heights Thursday 10:15AM
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

[redacted]