Going Social with ReplayKit and Game Center

What's new in social gaming

Session 605

Edwin Iskandar Software Engineer
Megan Gardner Software Engineer

Agenda

What's new in Game Center

Agenda

What's new in Game Center

- Guest players

Agenda

What's new in Game Center

- Guest players
- Unified server environment

Agenda

What's new in Game Center

- Guest players
- Unified server environment

Introducing ReplayKit

Agenda

What's new in Game Center

- Guest players
- Unified server environment

Introducing ReplayKit

- New way to add social to your games

Agenda

What's new in Game Center

- Guest players
- Unified server environment

Introducing ReplayKit

- New way to add social to your games
- Share game experiences

What's New in Game Center

Game Center

Social gaming

Friends

Leaderboards

Achievements

Real time multiplayer

Turn based multiplayer

Challenges

What's New?

Guest players

Unified environment

Guest Players

Adding more to your multiplayer

Players

Current usage

GKLocalPlayer

- Only one per device
- Must be an authenticated Game Center user

Players

Current usage

GKLocalPlayer

- Only one per device
- Must be an authenticated Game Center user

GKPlayer

- Friend of the GKLocalPlayer
- Participant in a match

Players

Current usage

GKLocalPlayer

- Only one per device
- Must be an authenticated Game Center user

GKPlayer

- Friend of the GKLocalPlayer
- Participant in a match

Multiplayer

- Real time
- Turn based

Guest Players

What are they?

NEW

GKPlayer instances

- Do not require authentication
- First class participants in multiplayer games

Can fill up all the slots in a game

- Up to three for real time
- Up to fifteen for turn based

Guest Players

What you need to know

Great for pass and play and handling AI players

You define the identifier—needs to be unique across the game

Cannot earn achievements or post scores

Sessions with guest players only compatible with other players running iOS 9

- Game still compatible with players on older systems

Game Center Sandbox Environment

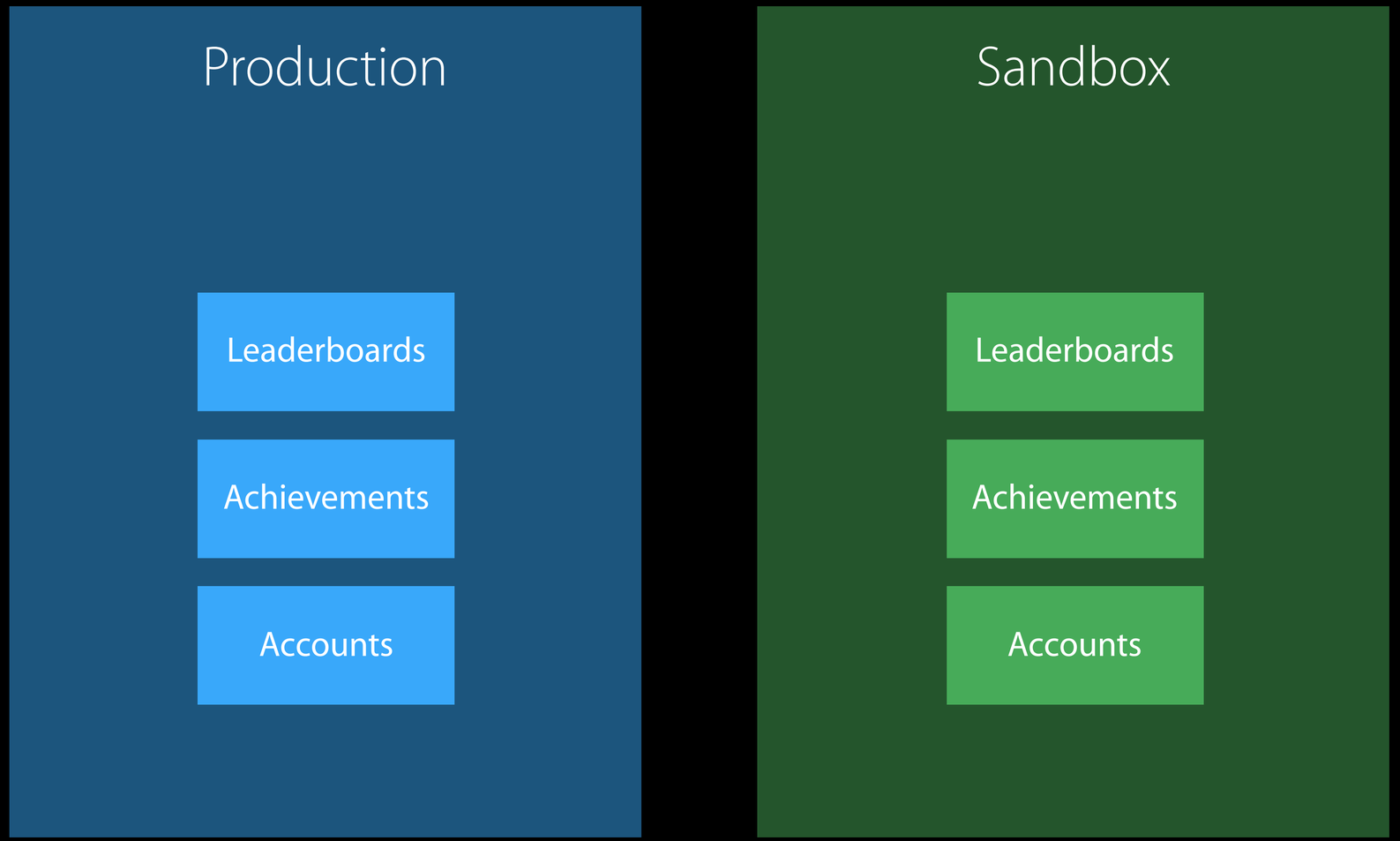
Unification with production

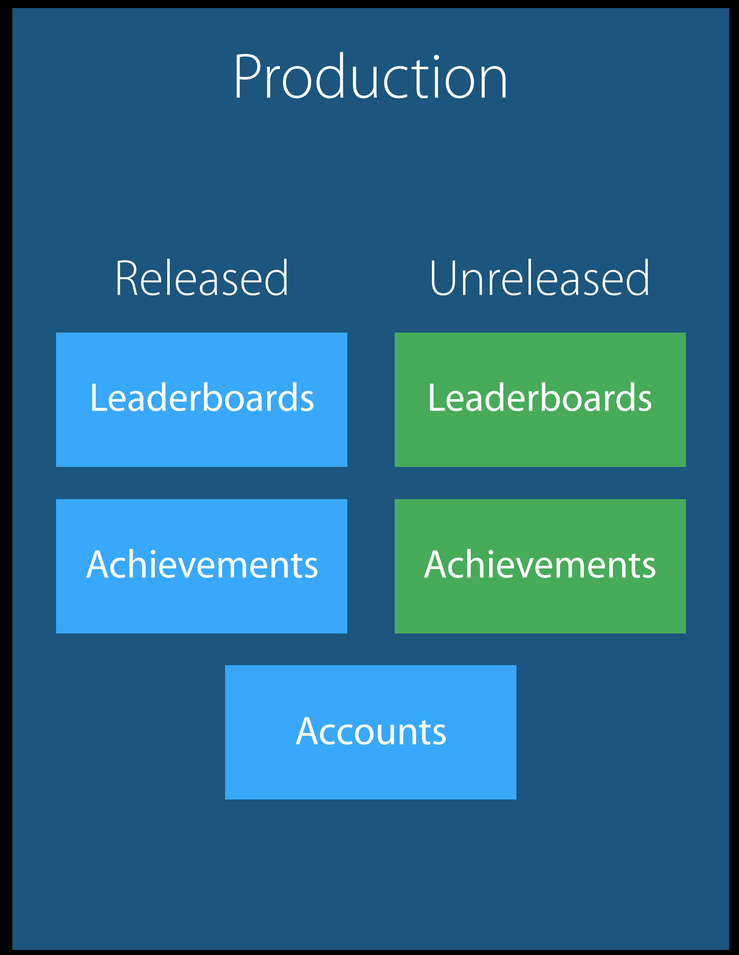
Sandbox

Original purpose

Environment for pre-release testing with Game Center

- Duplicated production Game Center functionality
- Same iTunes Connect metadata
- Different servers
- Different accounts


Production
Leaderboards
Achievements
Accounts
Sandbox
Leaderboards
Achievements
Accounts


Production
Released
Unreleased
Leaderboards
Leaderboards
Achievements
Achievements
Accounts

Unified Environment

NEW

Benefits

Simplifies accounts

Compatible with TestFlight

Multiplayer works across versions (if enabled)

Automatic

Unified Environment

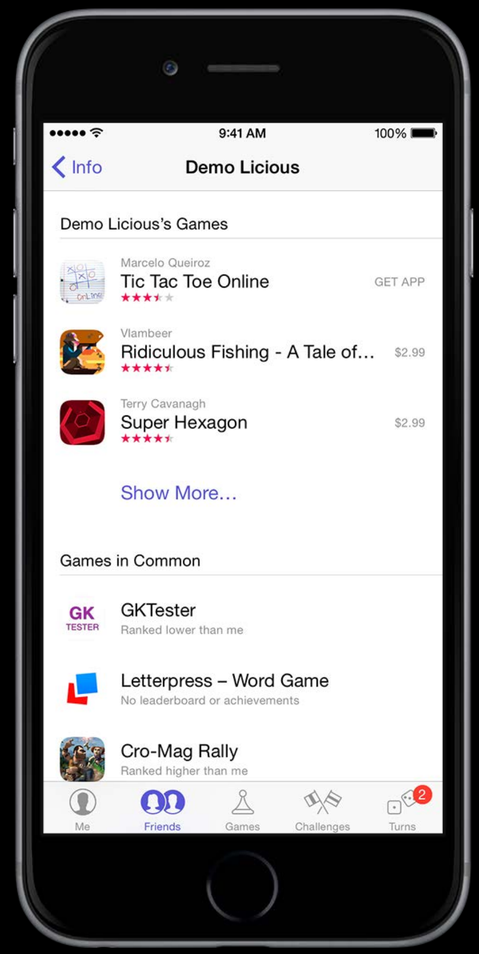
Considerations

New games appear in friend's games list

- Including unreleased

Scores post to existing leaderboards

iOS 8 users still need to flip the switch



Leaderboards

All versions will post to the same leaderboards

New leaderboard only visible to those who have that version

Remove test scores via iTunes Connect

Multiplayer

Play against any other version, specified in the compatibility matrix

Release versions can match against unreleased versions

iTunes Connect

Up to two sets of metadata will be maintained

- Currently released version
- Unreleased version

iTunes Connect

Up to two sets of metadata will be maintained

- Currently released version
- Unreleased version

Which data you are vended as a user will depend on your CFBundleVersion

- CFBundleVersion > Released Bundle Version—unreleased data
- CFBundleVersion <= Released Bundle Version—currently released data

Post-Sandbox

Compatibility and visibility

CFBundleVersion	Who Can See?	Who Can Play?	Where Will I Post?
0.9 “Beta”	Game Center Friends	Other beta versions	Beta leaderboards
1.0 “Released”	Game Center Friends	All versions, including future updates, if specified	Released leaderboards
2.0 “Update”	Game Center Friends	All specified versions	Released leaderboards and update leaderboards, not viewable by Game Center friends

Recap

What's new

Guest players

- Expands multiplayer opportunity

Recap

What's new

Guest players

- Expands multiplayer opportunity

Unified environment

- Fulfills developers requests
- Simple and automatic
- Works great with TestFlight

ReplayKit

Sharing game experiences

Edwin Iskandar Software Engineer

ReplayKit

ReplayKit

ReplayKit

Record your running app

ReplayKit

Record your running app

Add voice commentary

ReplayKit

Record your running app

Add voice commentary

Playback, scrub, and trim

ReplayKit

Record your running app

Add voice commentary

Playback, scrub, and trim

Share

- Social networks
- Video destination sites

ReplayKit

ReplayKit

HD quality

- Low performance impact
- Minimal power usage

ReplayKit

HD quality

- Low performance impact
- Minimal power usage

Privacy safeguards

ReplayKit

HD quality

- Low performance impact
- Minimal power usage

Privacy safeguards

Available in iOS 9

ReplayKit

HD quality

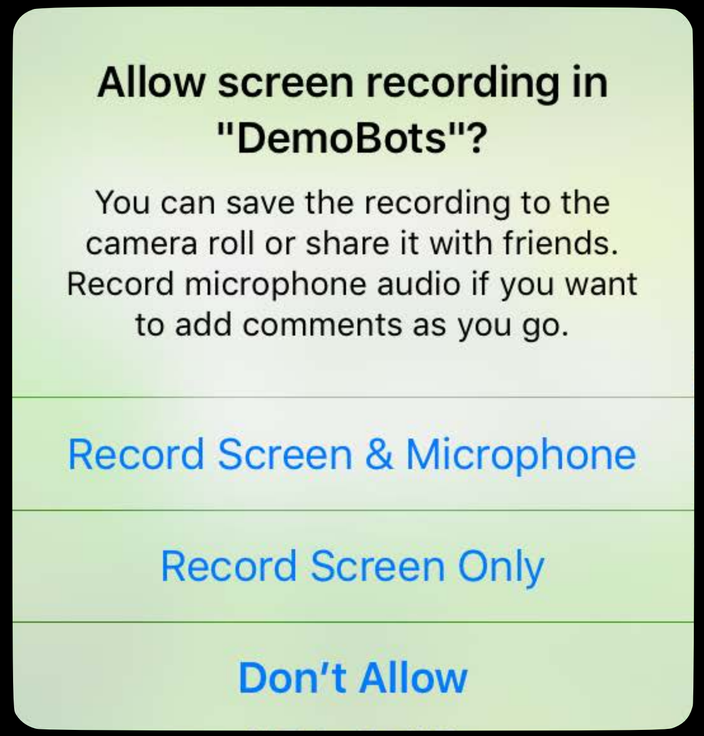
- Low performance impact
- Minimal power usage

Privacy safeguards

Available in iOS 9

A7 and A8 based devices

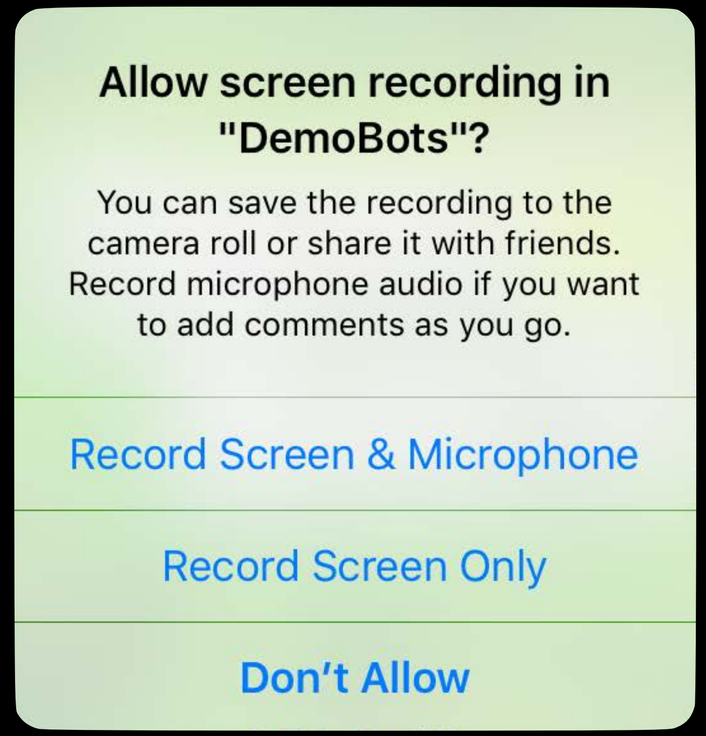
Privacy Safeguards



Privacy Safeguards

Permission required

- User consent prompt
- Parental controls



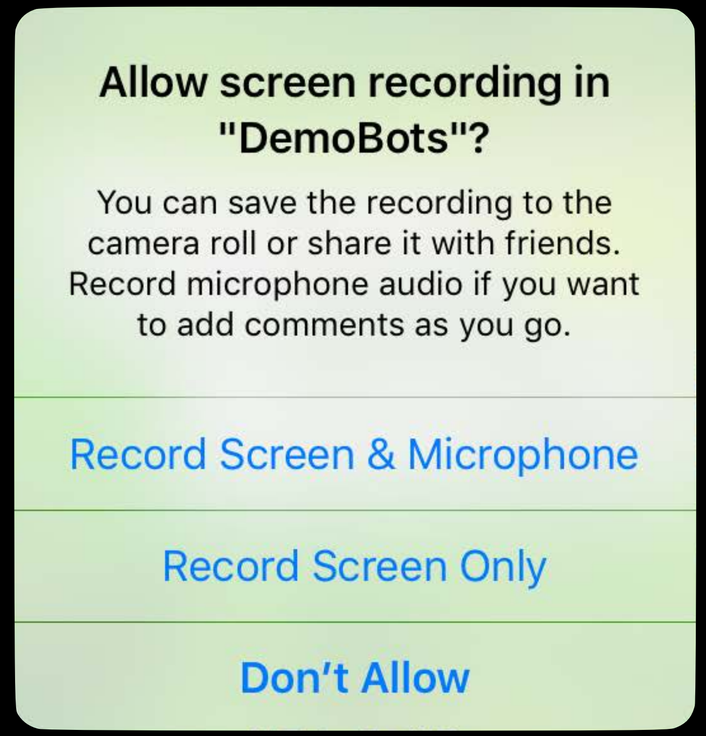
Privacy Safeguards

Permission required

- User consent prompt
- Parental controls

Recording excludes system UI

- Notifications
- Keyboard entry



Privacy Safeguards

Permission required

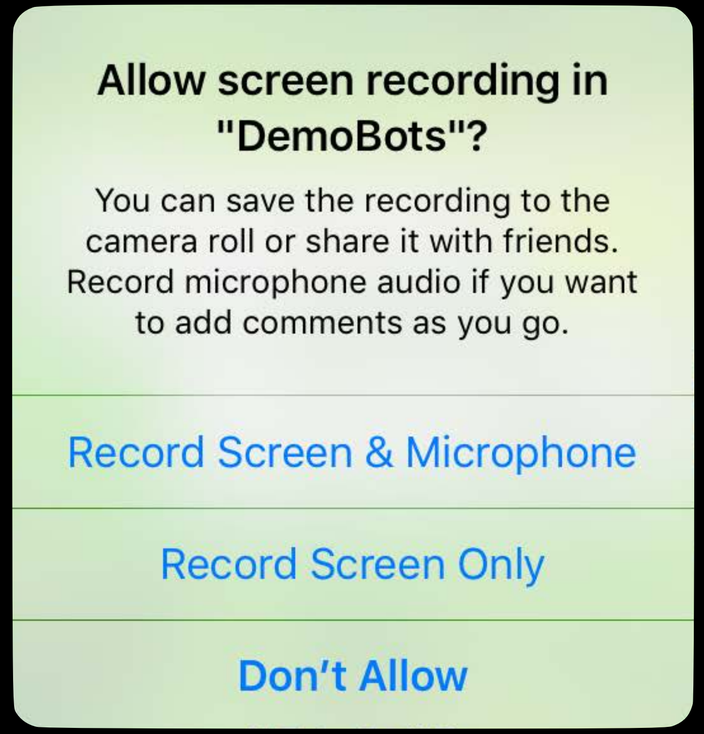
- User consent prompt
- Parental controls

Recording excludes system UI

- Notifications
- Keyboard entry

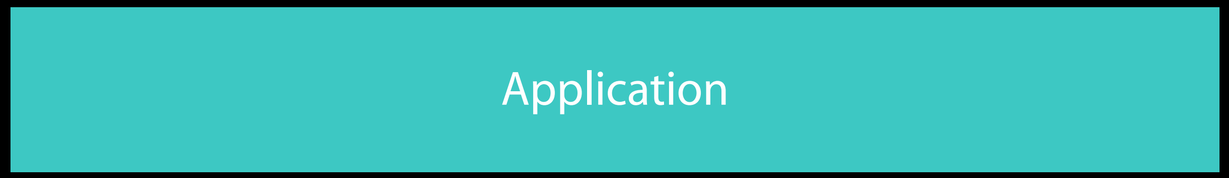
No direct access to recordings

- Share sheet only

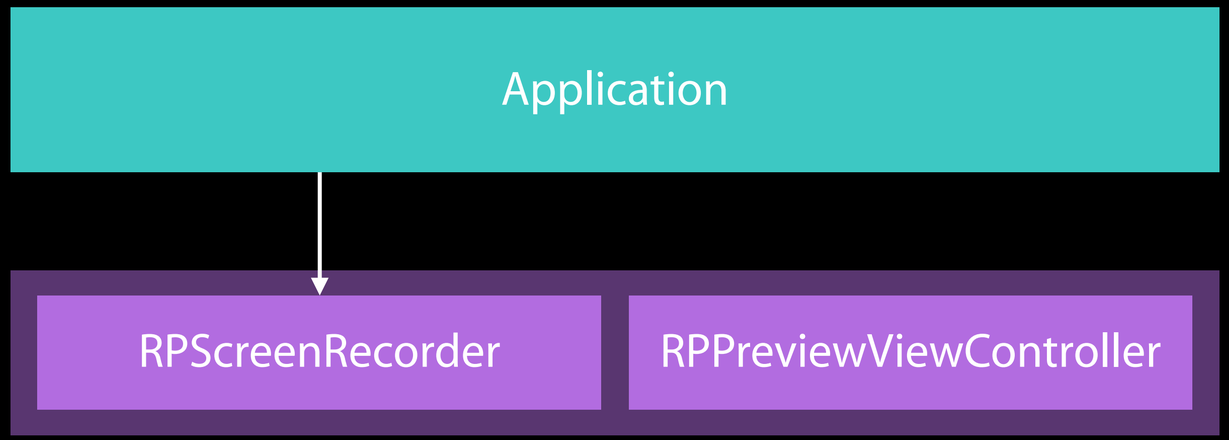


Architecture

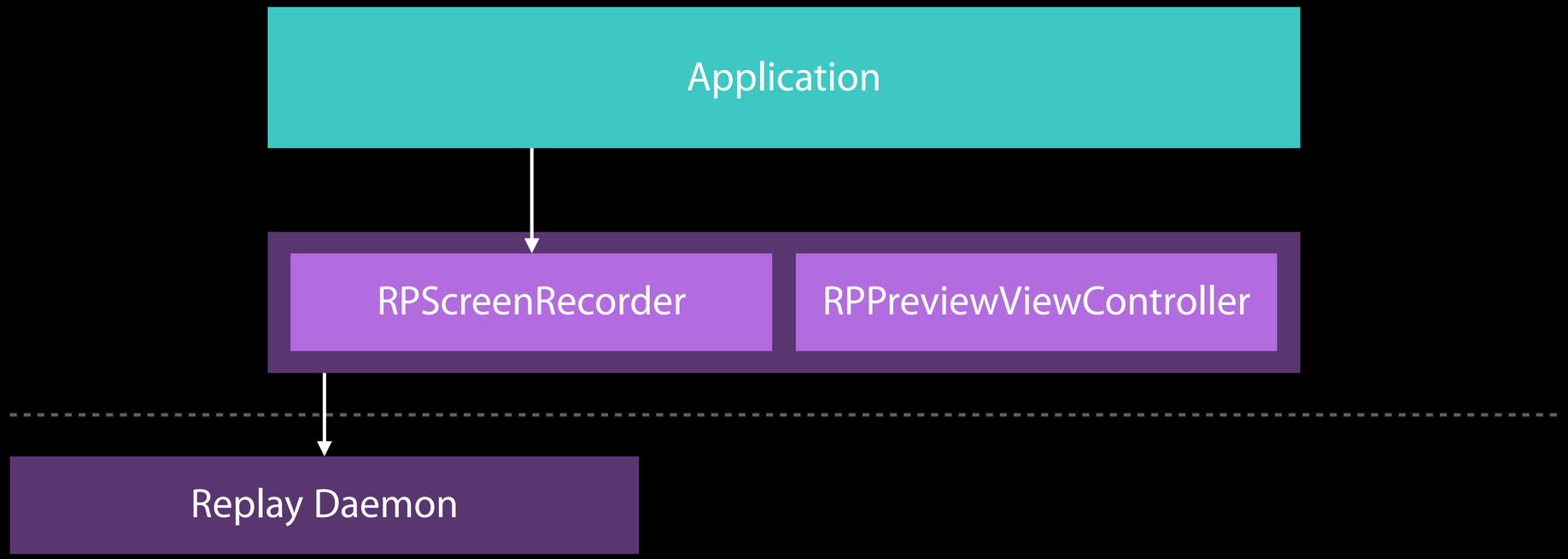
Application



Application

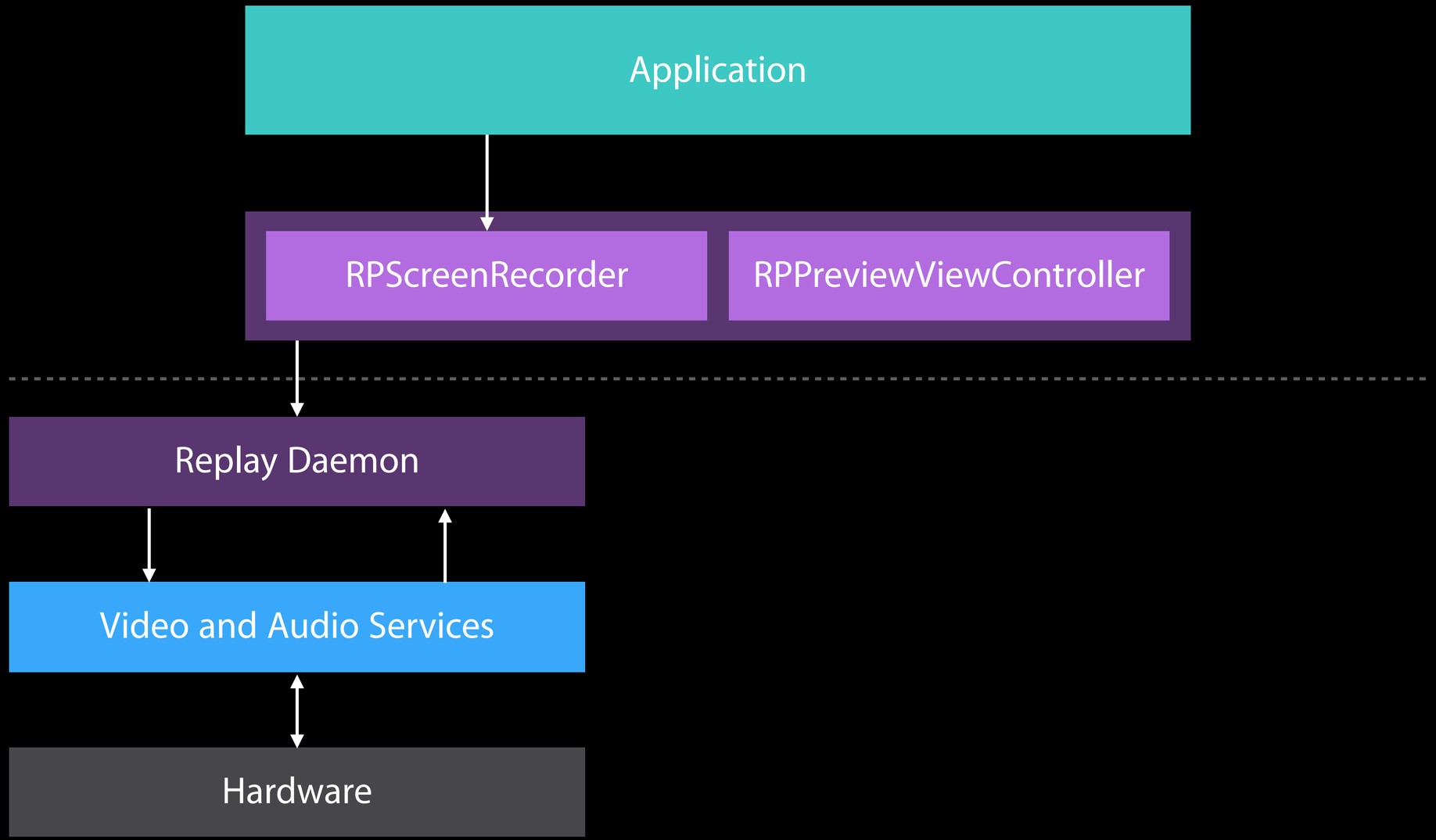


Application



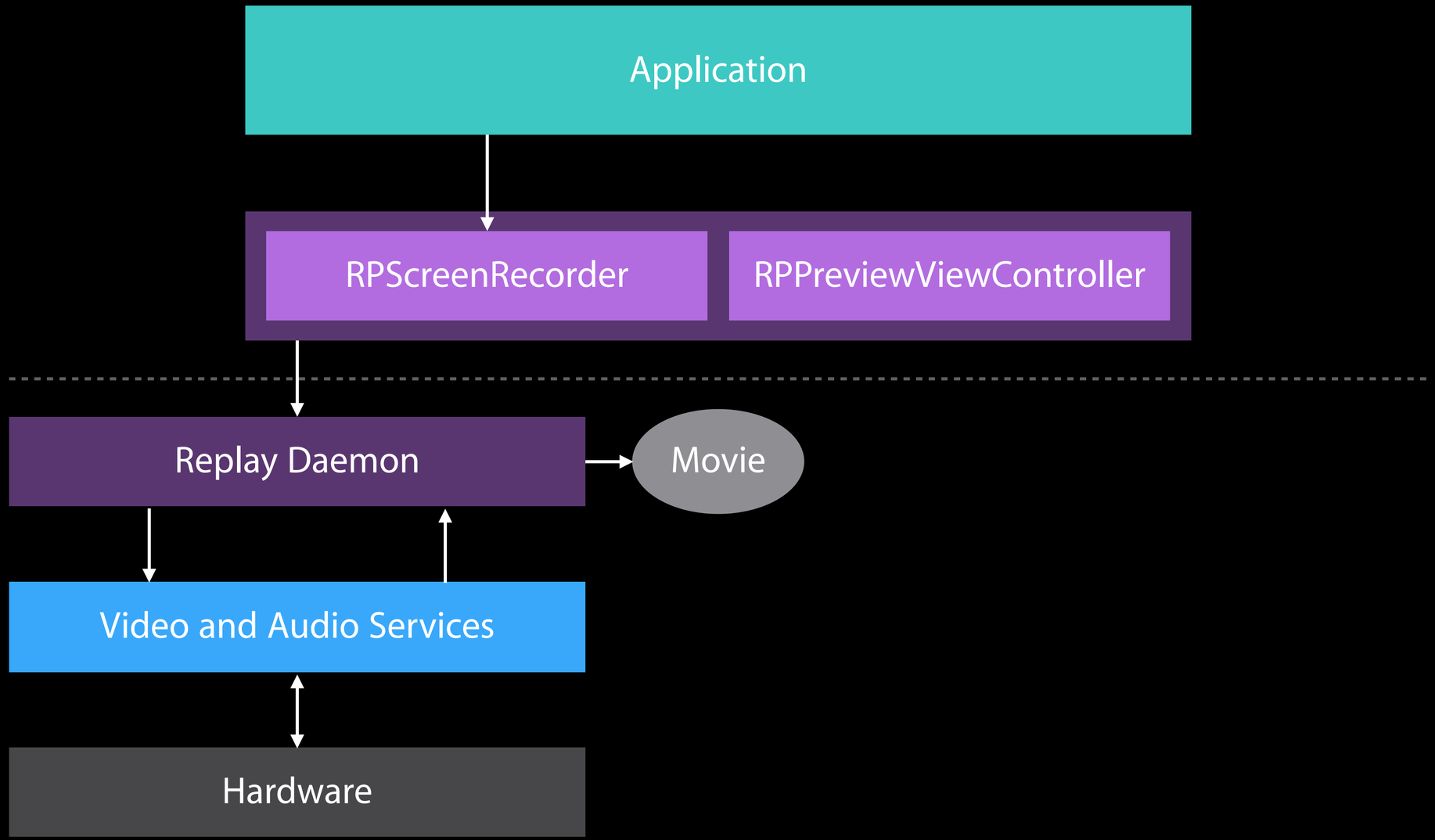
System

Application



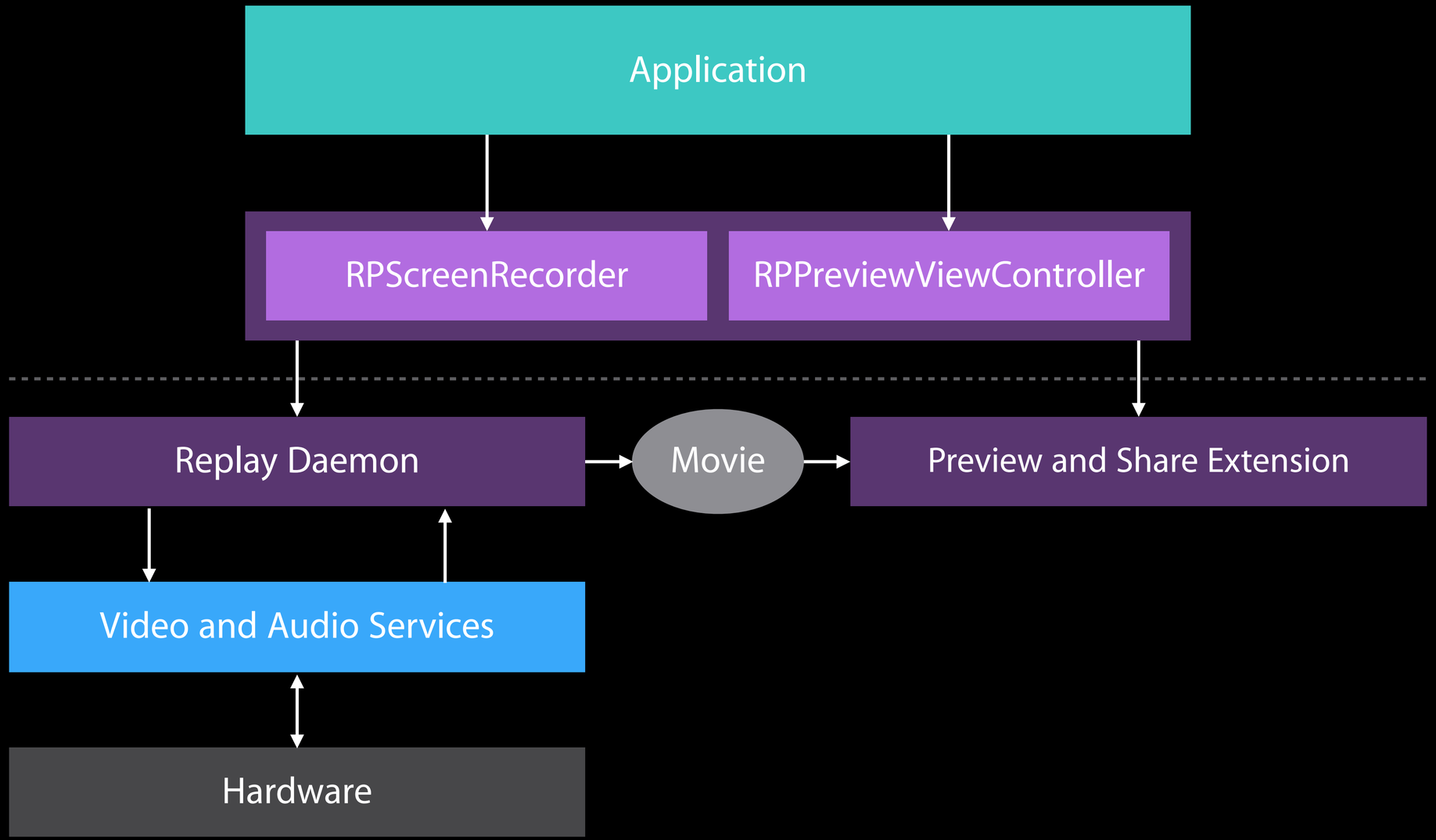
System

Application

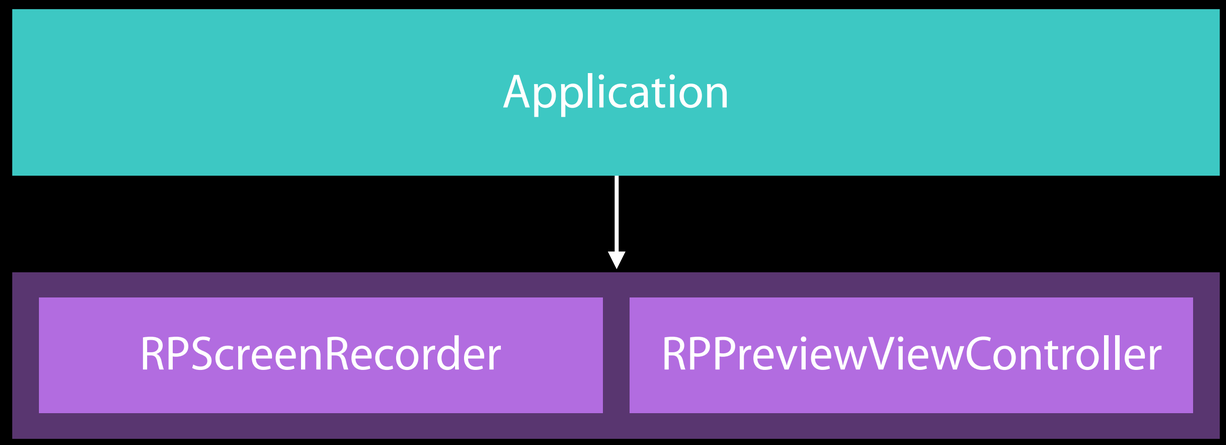


System

Application



System


Application
RPScreenRecorder
RPPreviewViewController

Using ReplayKit

Getting started

Classes and Protocols

Classes and Protocols

`RPScreenRecorder`

- Start, stop, discard recording
- Check ability to record
- Enable microphone for commentary

Classes and Protocols

`RPScreenRecorder`

- Start, stop, discard recording
- Check ability to record
- Enable microphone for commentary

`RPScreenRecorderDelegate`

- If availability changes
- If recording stops (due to error)

Classes and Protocols

`RPScreenRecorder`

- Start, stop, discard recording
- Check ability to record
- Enable microphone for commentary

`RPScreenRecorderDelegate`

- If availability changes
- If recording stops (due to error)

`RPPreviewViewController`

- Preview the recording
- Edit and trim
- Share

Classes and Protocols

`RPScreenRecorder`

- Start, stop, discard recording
- Check ability to record
- Enable microphone for commentary

`RPScreenRecorderDelegate`

- If availability changes
- If recording stops (due to error)

`RPPreviewViewController`

- Preview the recording
- Edit and trim
- Share

`RPPreviewViewControllerDelegate`

- After view controller dismissal

Example

DemoBots

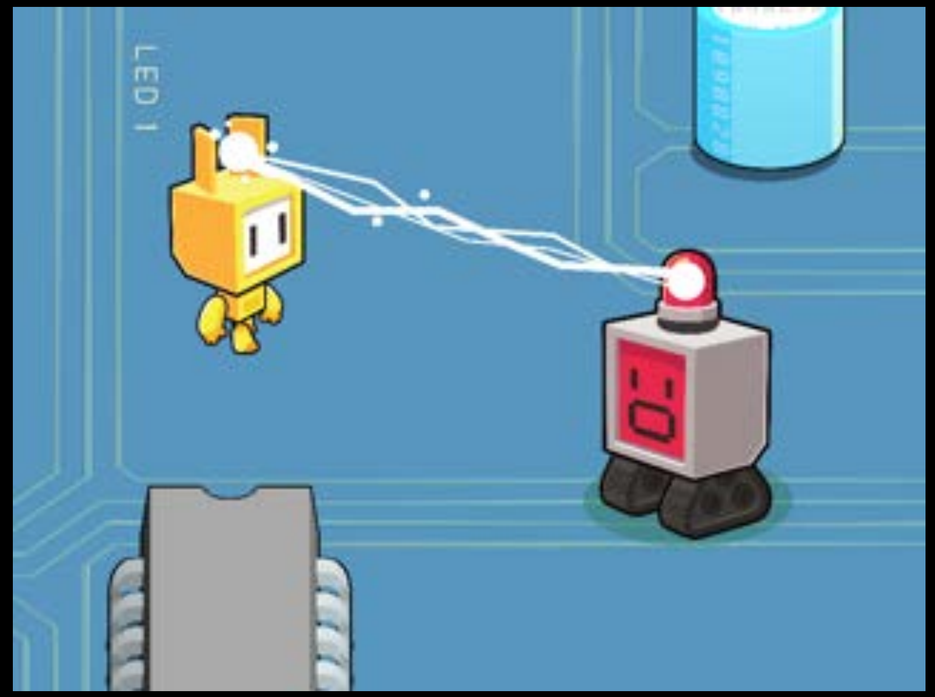
DemoBots

Taskbots keep the circuit board running

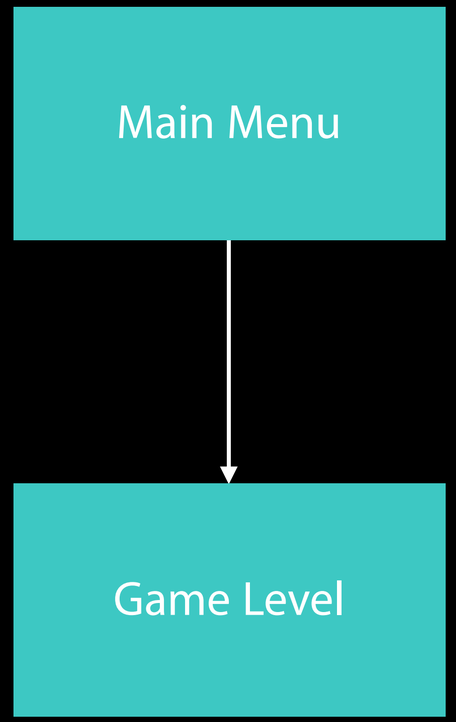
Bots are buggy, debug them

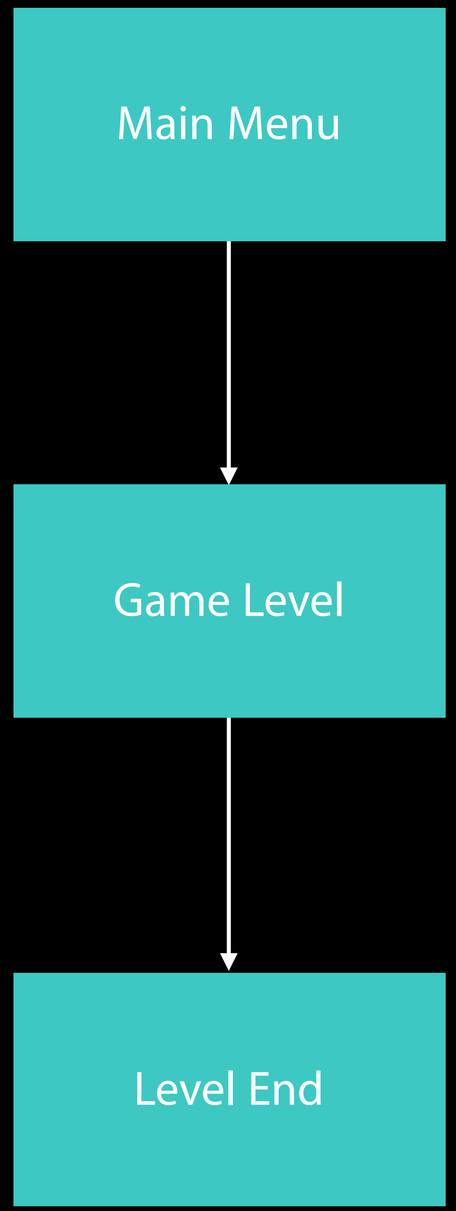
Bite size 2–3 minute levels

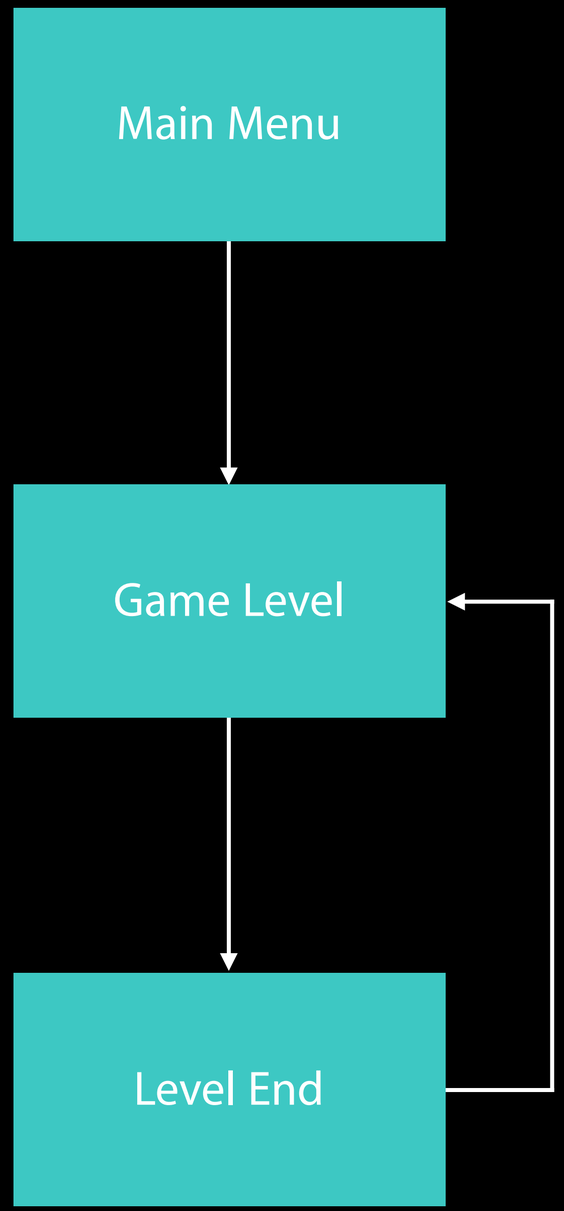
Source available at developer.apple.com

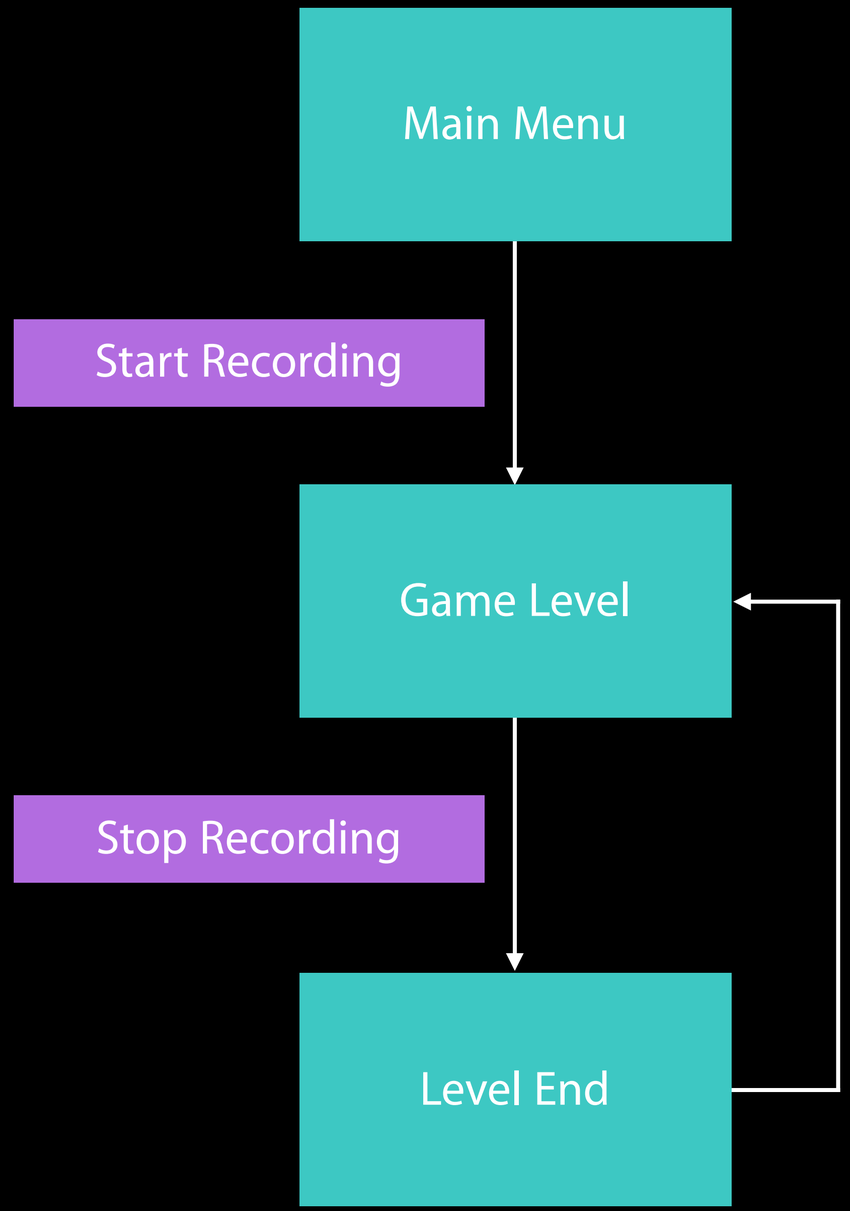


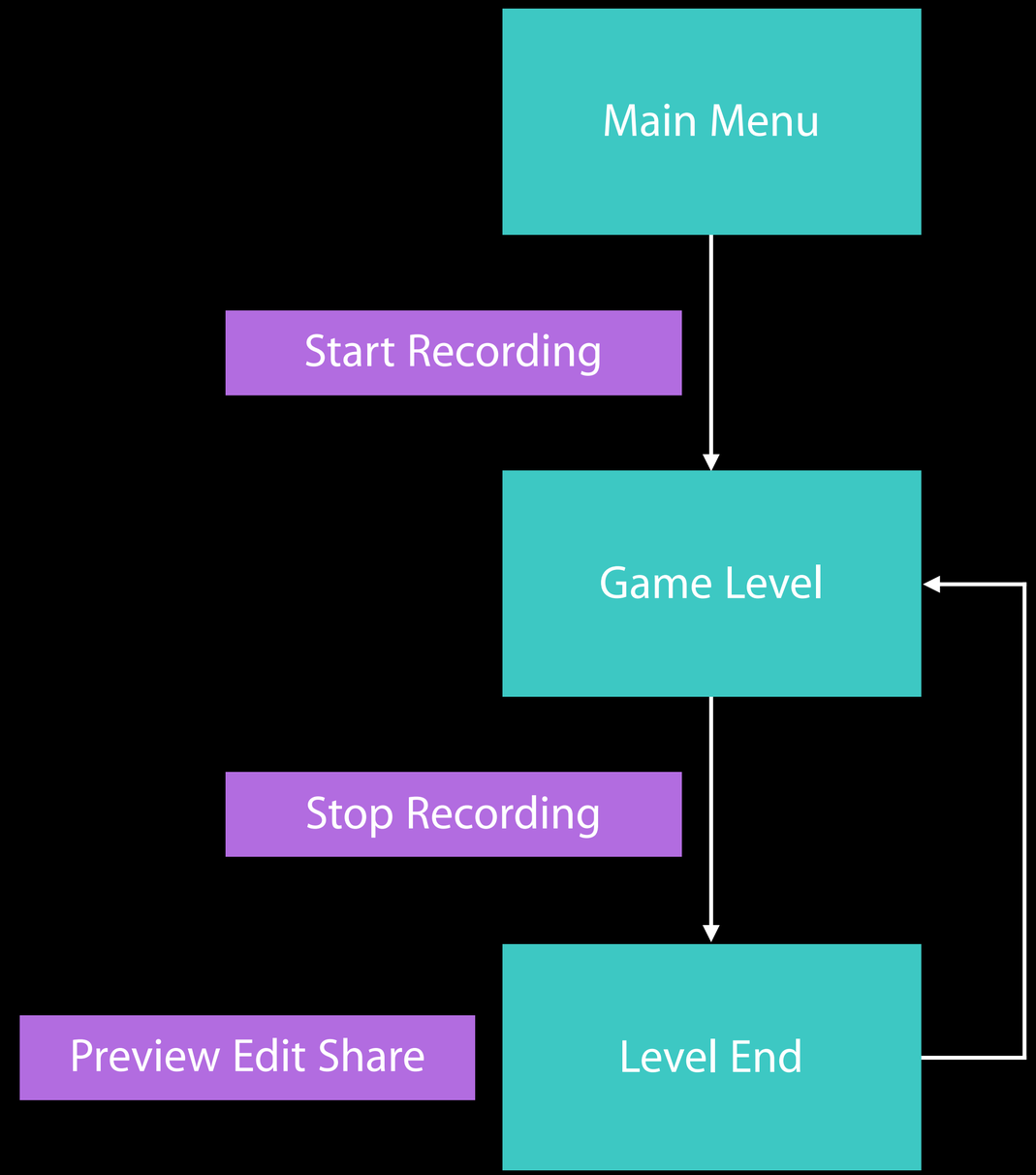
Main Menu

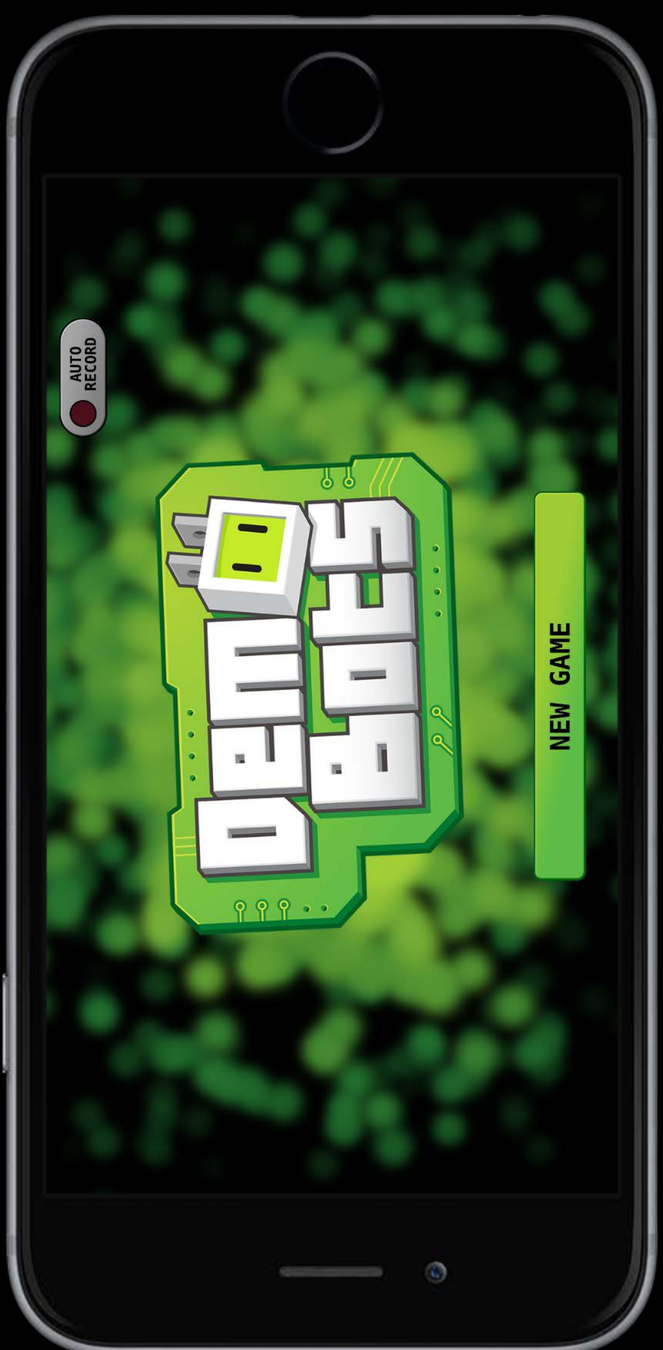

Main Menu
Game Level


Main Menu
Game Level
Level End


Main Menu
Game Level
Level End


Main Menu
Start Recording
Game Level
Stop Recording
Level End


Main Menu
Start Recording
Game Level
Stop Recording
Preview Edit Share
Level End


AUTO
RECORD
Demo Bots
NEW GAME

AUTO
RECORD
Demo
Bots
NEW GAME

AUTO
RECORD
Demo
Bots
NEW GAME

Shared Recorder

```
class RPScreenRecorder : NSObject {
    class func sharedRecorder() -> RPScreenRecorder
```

```
let sharedRecorder = RPScreenRecorder.sharedRecorder()
```

Shared Recorder

```
class RPScreenRecorder : NSObject {
    class func sharedRecorder() -> RPScreenRecorder
```

```
let sharedRecorder = RPScreenRecorder.sharedRecorder()
```

Start Recording

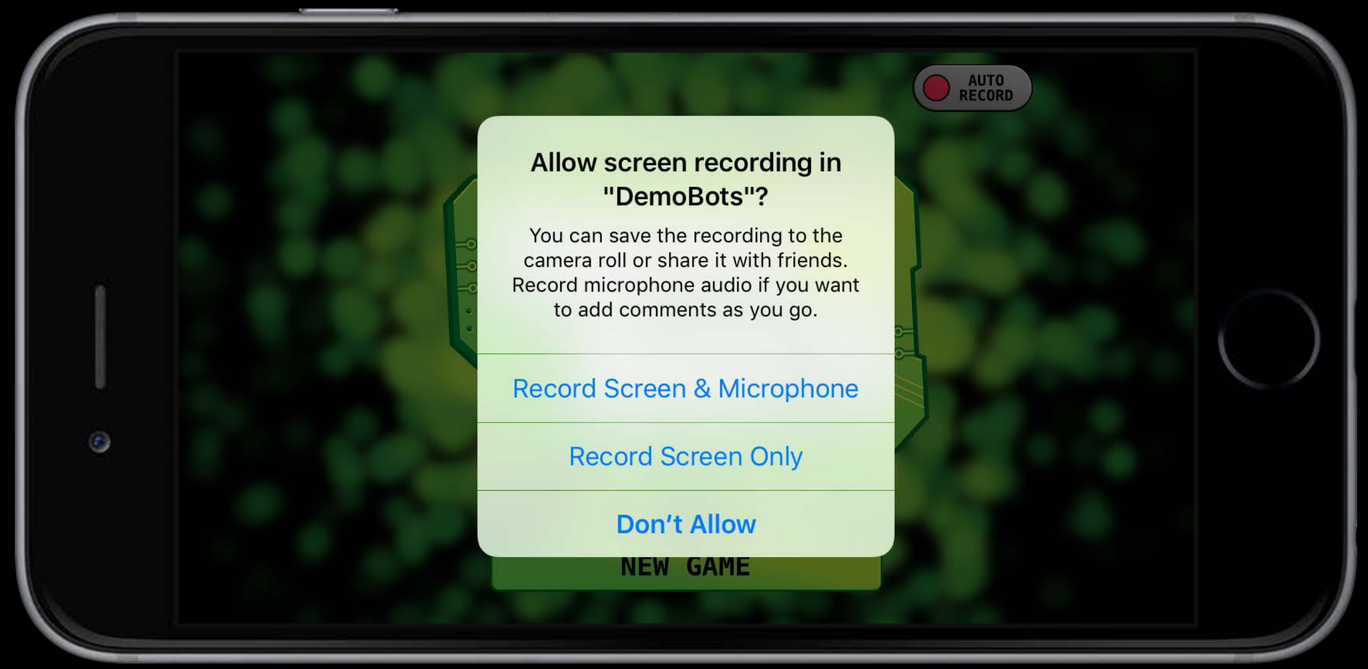
```
class RPScreenRecorder : NSObject {
    func startRecordingWithMicrophoneEnabled(microphoneEnabled: Bool,
            handler: ((NSError?) -> Void)?)
```

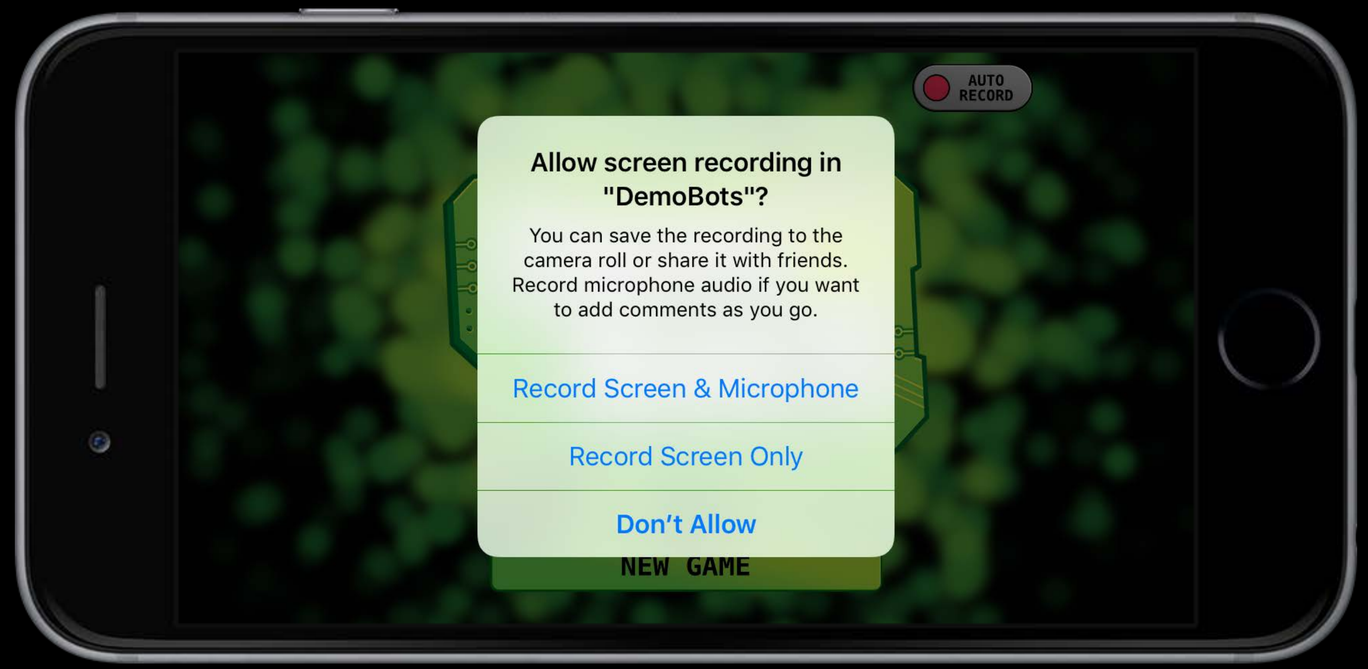
```
func levelDidStart() {
    sharedRecorder.startRecordingWithMicrophoneEnabled(true) { (error:) in
        if error != nil {
            // pause game and show error
        }
    }
}
```

Start Recording

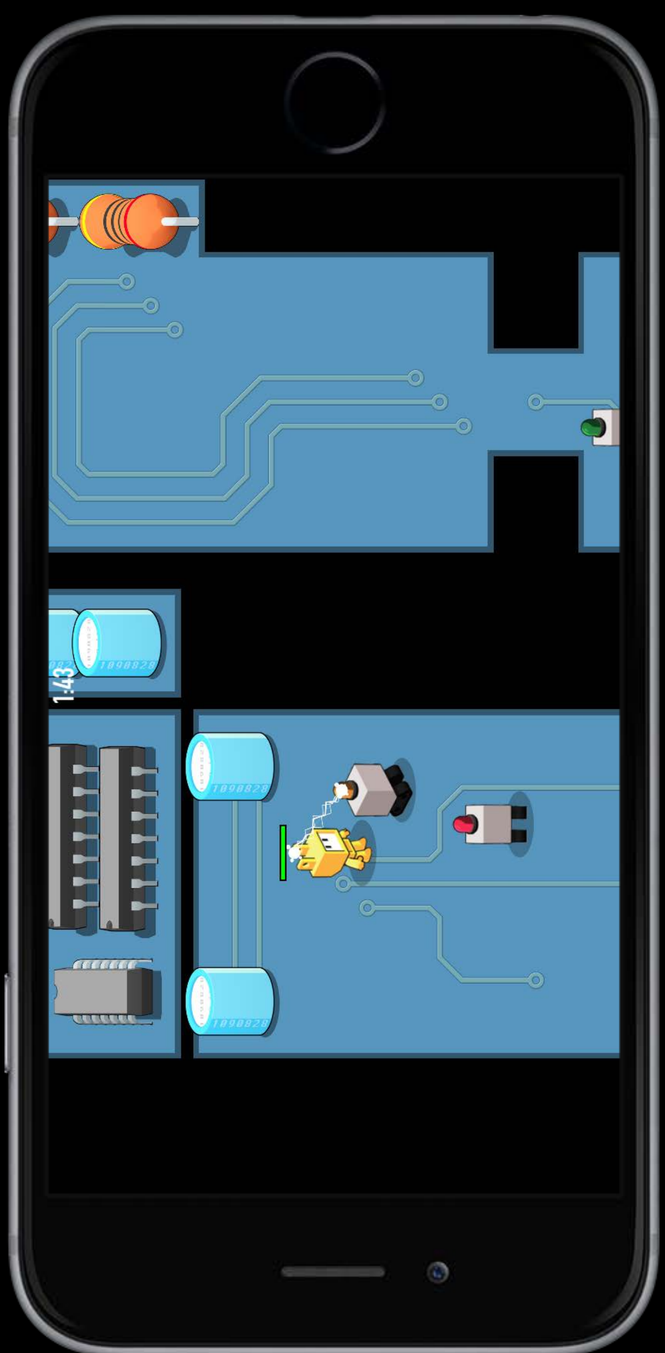
```
class RPScreenRecorder : NSObject {
    func startRecordingWithMicrophoneEnabled(microphoneEnabled: Bool,
            handler: ((NSError?) -> Void)?)
```

```
func levelDidStart() {
    sharedRecorder.startRecordingWithMicrophoneEnabled(true) { (error:) in
        if error != nil {
            // pause game and show error
        }
    }
}
```


AUTO RECORD
Allow screen recording in "DemoBots"?
You can save the recording to the camera roll or share it with friends. Record microphone audio if you want to add comments as you go.
Record Screen & Microphone
Record Screen Only
Don’t Allow
NEW GAME


AUTO RECORD
Allow screen recording in "DemoBots"?
You can save the recording to the camera roll or share it with friends. Record microphone audio if you want to add comments as you go.
Record Screen & Microphone
Record Screen Only
Don’t Allow
NEW GAME


2:50


1:43

Stop Recording

```
class RPScreenRecorder : NSObject {
    func stopRecordingWithHandler(
            handler: ((RPPreviewViewController?, NSError?) -> Void)?)
```

```
func levelWillEnd() {
    sharedRecorder.stopRecordingWithHandler {
        (previewViewController, error) -> Void in

        // Handle error
        if previewViewController != nil {
            // Keep a reference for later use
            self.previewViewController = previewViewController
        }
    }
```

Stop Recording

```
class RPScreenRecorder : NSObject {
    func stopRecordingWithHandler(
            handler: ((RPPreviewViewController?, NSError?) -> Void)?)
```

```
func levelWillEnd() {
    sharedRecorder.stopRecordingWithHandler {
            (previewViewController, error) -> Void in

        // Handle error
        if previewViewController != nil {
            // Keep a reference for later use
            self.previewViewController = previewViewController
        }
    }
```


2:33
AUTO RECORD
Demo Bots
PLAY RECORDING
LEVEL COMPLETE!
NEXT LEVEL
REPLAY LEVEL
QUIT


2:33
AUTO RECORD
Demo Bots
PLAY RECORDING
LEVEL COMPLETE!
NEXT LEVEL
REPLAY LEVEL
QUIT

Present Preview UI

```
class UIViewController : … {
    func presentViewController(viewControllerToPresent: UIViewController,
            animated flag: Bool, completion: (() -> Void)?)
```

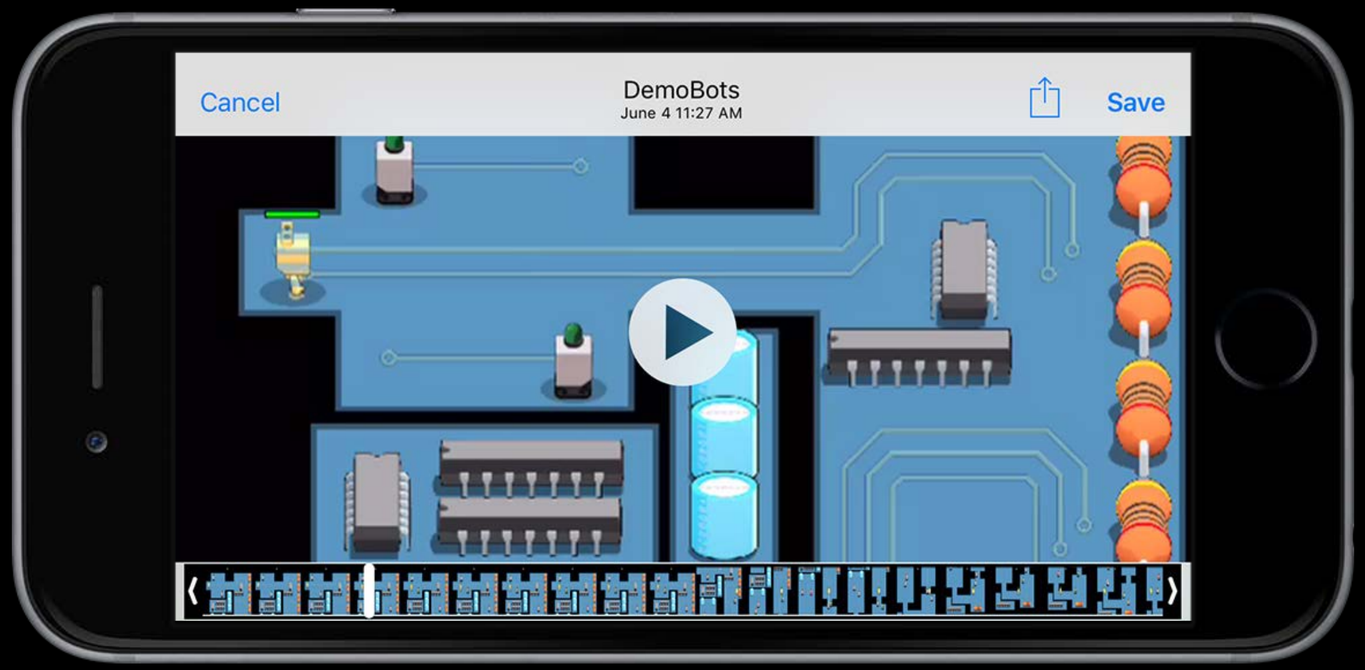
```
@IBAction func didPressViewRecordingButton() {
    // Present the preview view controller we made a reference to before
    self.presentViewController(previewViewController, animated: true,
            completion:nil)
}
```

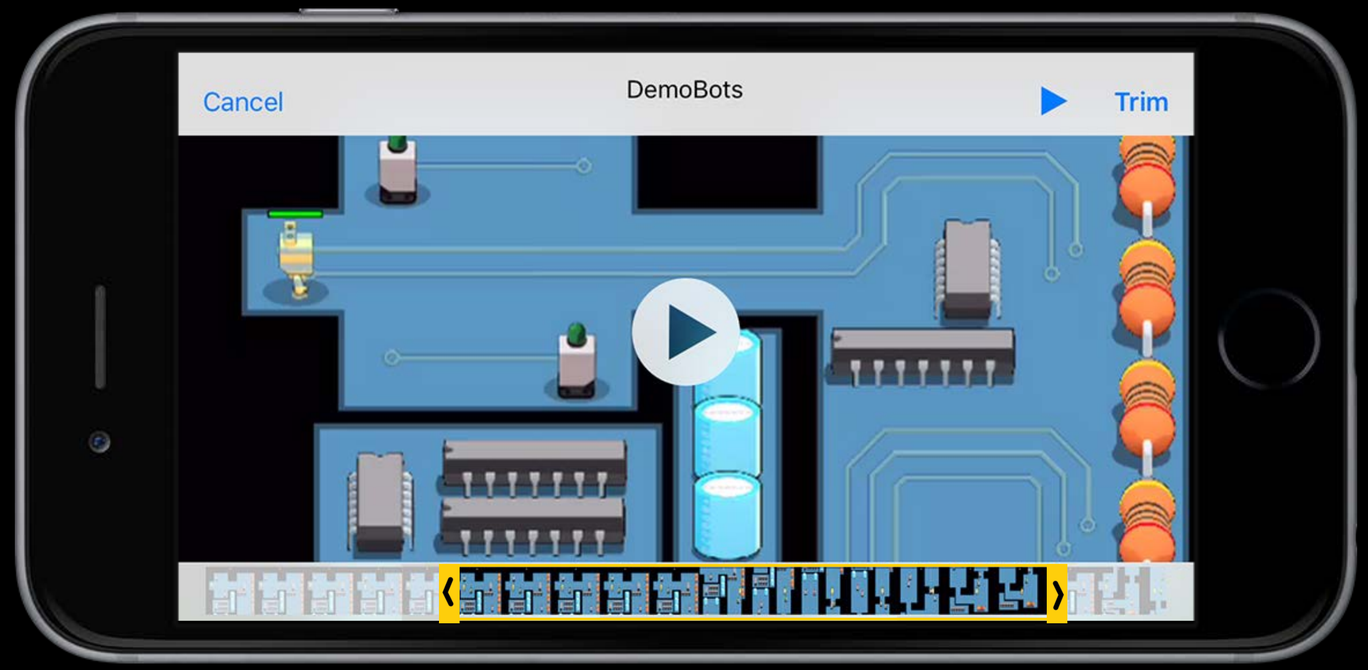
Present Preview UI

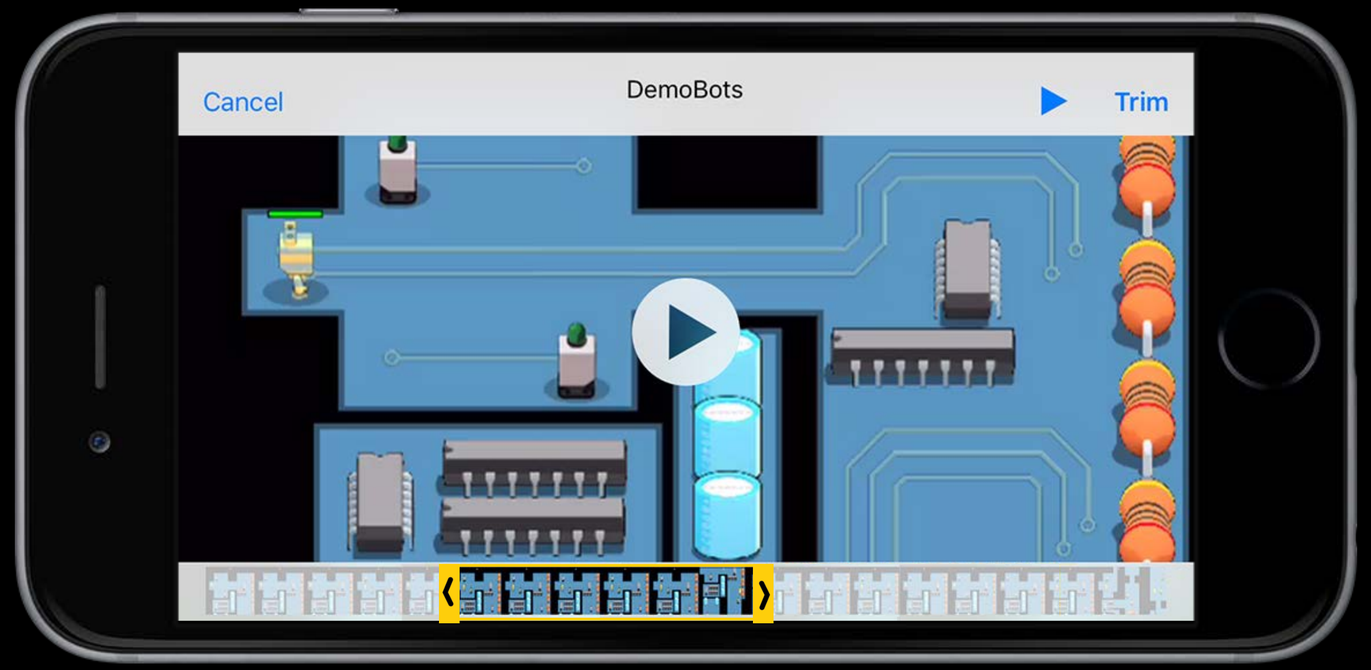
```
class UIViewController : … {
    func presentViewController(viewControllerToPresent: UIViewController,
            animated flag: Bool, completion: (() -> Void)?)
```

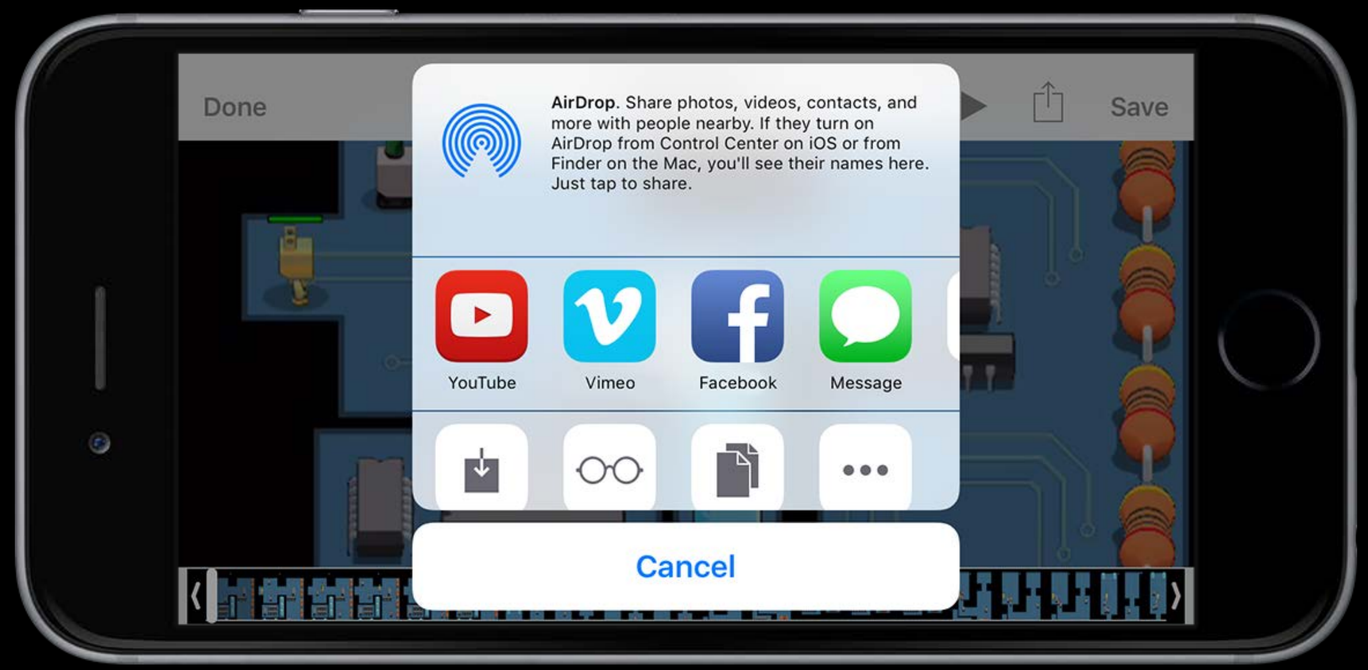
```
@IBAction func didPressViewRecordingButton() {

    // Present the preview view controller we made a reference to before
    self.presentViewController(previewViewController, animated: true,
            completion:nil)
}
```

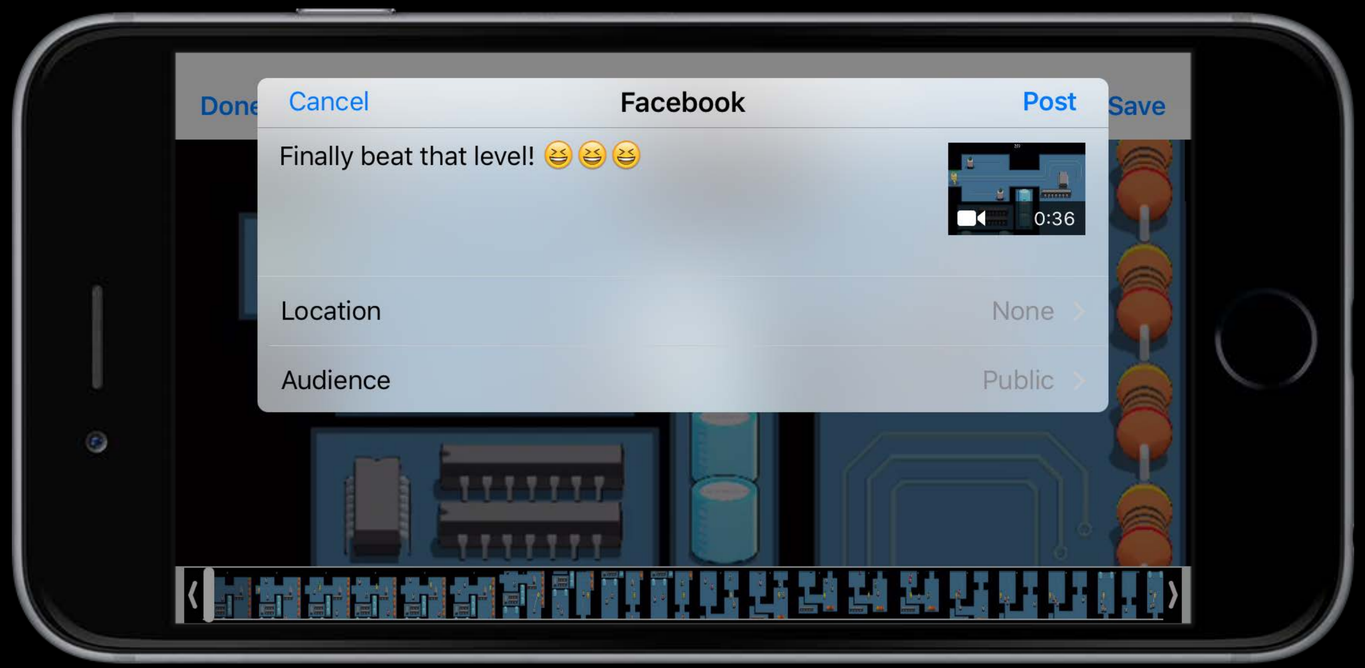

AUTO RECORD
Demo Bots
PLAY RECORDING
LEVEL COMPLETE!
NEXT LEVEL
REPLAY LEVEL
QUIT


Cancel
DemoBots
June 4 11:27 AM
Save


Cancel
DemoBots
Trim


Cancel
DemoBots
Trim


Done
Save
AirDrop. Share photos, videos, contacts, and more with people nearby. If they turn on AirDrop from Control Center on iOS or from Finder on the Mac, you'll see their names here. Just tap to share.
YouTube
Vimeo
Facebook
Message
Cancel


Done
Cancel
Facebook
Post
Save
Finally beat that level! 😆😆😆
0:36
Location
None
Audience
Public


Cancel
DemoBots
June 4 11:27 AM
Save


Cancel
DemoBots
June 4 11:27 AM
Save

Dismissing Preview UI

```
protocol RPPreviewViewControllerDelegate : NSObjectProtocol {

    optional func previewControllerDidFinish(
            previewController: RPPreviewViewController)
```

```
@IBAction func didPressViewRecordingButton() {
    previewViewController.previewViewControllerDelegate = self
}

func previewControllerDidFinish(previewController: RPPreviewViewController) {
    previewViewController.dismissViewControllerAnimated(true, completion: nil)
```

Dismissing Preview UI

```
protocol RPPreviewViewControllerDelegate : NSObjectProtocol {

    optional func previewControllerDidFinish(
            previewController: RPPreviewViewController)
```

```
@IBAction func didPressViewRecordingButton() {
    previewViewController.previewViewControllerDelegate = self
}

func previewControllerDidFinish(previewController: RPPreviewViewController) {
    previewViewController.dismissViewControllerAnimated(true, completion: nil)
```

Using ReplayKit

Getting started

Getting a shared instance of the recorder

Starting and stopping the recorder

Presenting and dismissing the Preview UI

Using ReplayKit

Fine tuning

Verifying Availability

Verifying Availability

Recording may be unavailable

- AirPlay in use
- TV-out in use
- Unsupported device

Verifying Availability

Recording may be unavailable

- AirPlay in use
- TV-out in use
- Unsupported device

Guidance

- Use `available` property to check for availability
- Disable recording UI if false
- Use `screenRecorderDidChangeAvailability` to listen for changes

Verifying Availability

Example

```
class RPScreenRecorder : NSObject {
    var available: Bool { get }
```

```
func updateButtonUI() {
    recordingToggleButton.hidden = !sharedRecorder.available
}
```

Verifying Availability

Example

```
class RPScreenRecorder : NSObject {
    var available: Bool { get }
```


```
func updateButtonUI() {
    recordingToggleButton.hidden = !sharedRecorder.available
}
```

Discarding the Recording

Automatically discarded when new recording starts

- Only one recording at a time per app

Discard when preview no longer accessible

- Use `discardRecordingWithHandler`

Discarding the Recording

Automatically discarded when new recording starts

- Only one recording at a time per app

Discard when preview no longer accessible

- Use `discardRecordingWithHandler`

Discarding the Recording

Example

```
class RPScreenRecorder : NSObject {
    func discardRecordingWithHandler(handler: () -> Void)
```

```
func willTransitionToNextLevel() {
    sharedRecorder.discardRecordingWithHandler {
        // start next level
        self.transitionToNextLevel()
    }
}
```

Discarding the Recording

Example

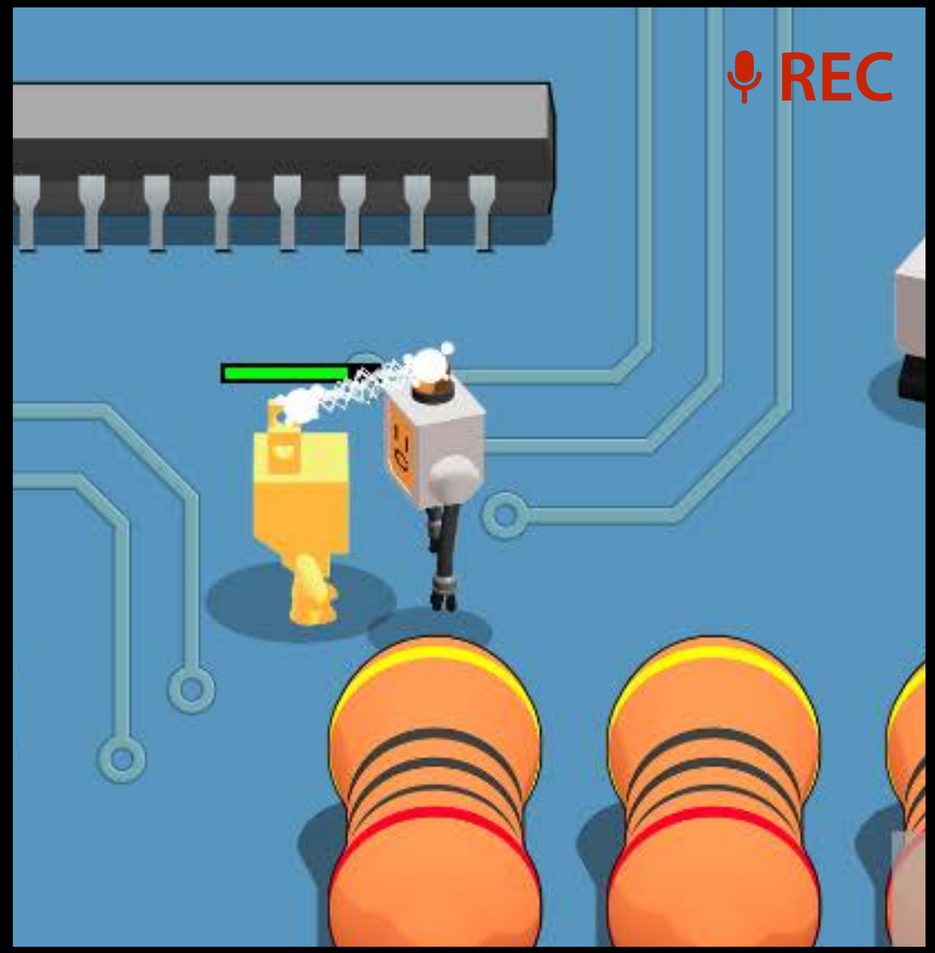
```
class RPScreenRecorder : NSObject {
    func discardRecordingWithHandler(handler: () -> Void)
```

```
func willTransitionToNextLevel() {
    sharedRecorder.discardRecordingWithHandler {
        // start next level
        self.transitionToNextLevel()
    }
}
```

Recording Indicator

Indicate that recording is on

Indicate that microphone is on



Recording Indicator

Example

```
class RPScreenRecorder : NSObject {

    var recording: Bool { get }
    var microphoneEnabled: Bool { get }
```

```
func recordingDidStart() {
    recordingIndicator.hidden = !sharedRecorder.recording
    microphoneIndicator.hidden = !sharedRecorder.microphoneEnabled
}
```

Recording Indicator

Example

```
class RPScreenRecorder : NSObject {

    var recording: Bool { get }
    var microphoneEnabled: Bool { get }
```

```
func recordingDidStart() {
    recordingIndicator.hidden = !sharedRecorder.recording
    microphoneIndicator.hidden = !sharedRecorder.microphoneEnabled
}
```

Excluding UI

Excluding UI

Hide elements that are uninteresting to a spectator

- Recording indicators
- Virtual controls
- Pause and menu buttons

Excluding UI

Hide elements that are uninteresting to a spectator

- Recording indicators
- Virtual controls
- Pause and menu buttons

ReplayKit only records your applications main UIWindow

- Use a separate UIWindow to hide UI

When to Record

Automatic vs. user-initiated

When to Record

Automatic vs. user-initiated

App controlled (automatic)

- Short gameplay sessions

When to Record

Automatic vs. user-initiated

App controlled (automatic)

- Short gameplay sessions

User-initiated

- Longer gameplay sessions
- Spread out interesting events

When to Record

Automatic vs. user-initiated

App controlled (automatic)

- Short gameplay sessions

User-initiated

- Longer gameplay sessions
- Spread out interesting events

Choose what is appropriate

Using ReplayKit

Getting Started

- Getting a shared instance of the recorder
- Starting and stopping the recorder
- Presenting and dismissing the Preview UI

Fine Tuning

- Verifying availability
- Discarding the recording
- Showing indicator
- Excluding UI
- When to record

Demo

Call of Champions

Cinco Barnes
Chief Vision Officer, Spacetime Studios


SPACETIME STUDIOS

CALL OF CHAMPIONS

PLAY CHAMPIONS

100
21,442
24,649

JustG
Adept

JustG
1 COBAL

Global Players: 1
Queued Players:0

Add Friend

Add Friend

PLAY

Record Match

Role: All

JustG: Stand by... going to Record this match.

PLAY
CHAMPIONS
2,304
445
8,519
Cinco
Adept
Cinco
1 MACKENNA
PhantomLOrd
1 CINDER
Snoopeh
1 AVALANCHE
Voyboy
1 BJORN
Cinco
GOBI 1
Jakles
MODUS 1
JustG
AVALANCHE 1
PLAY
CHANGE MODE
Record Match

PLAY
CHAMPIONS
2,304
445
8,519
Cinco
Adept
Cinco
1 MACKENNA
PhantomLOrd
1 CINDER
Snoopeh
1 AVALANCHE
Voyboy
1 BJORN
Cinco
GOBI 1
Jakles
MODUS 1
JustG
AVALANCHE 1
PLAY
CHANGE MODE
Record Match

LEAVE

00:19:04

0 / 0 / 0

TIMELINE

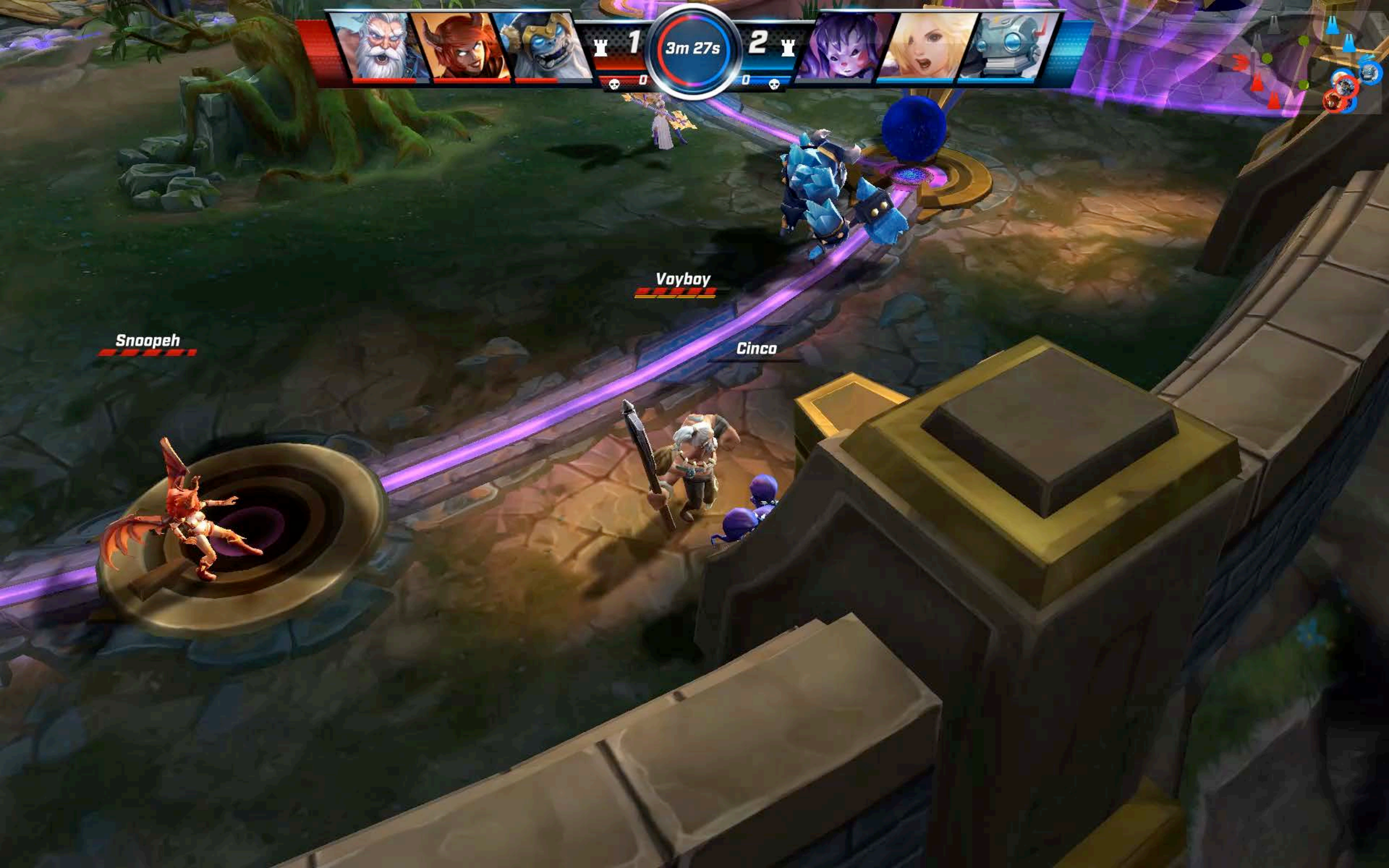
LIVE

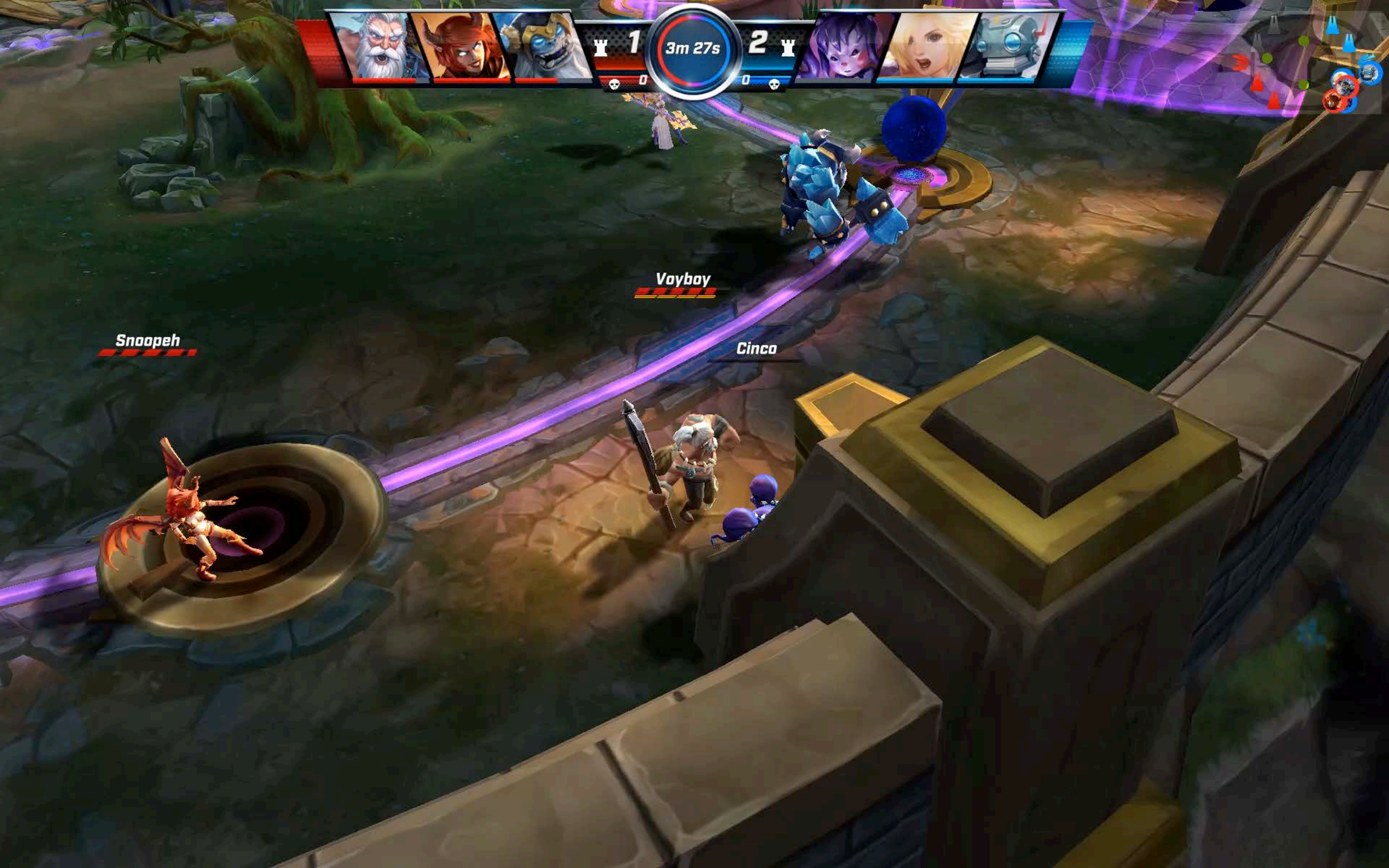
PLAY SPEED

1/4x | 1/2x | 1x | 2x | 4x

RED	TEAM POWER-UP	POWER-UP	TIME ON ORB	DAMAGE TAKEN	DAMAGE DONE	ACC.	ASSISTS	DEATHS	KILLS
	0	0	0s	0	0	0	0	0	0
	22	22	0s	0	22	22	0	0	0
	86	86	0s	0	86	86	0	0	0

BLUE	TEAM POWER-UP	POWER-UP	TIME ON ORB	DAMAGE TAKEN	DAMAGE DONE	ACC.	ASSISTS	DEATHS	KILLS
	0	0	0s	0	0	0	0	0	0
	0	0	6s	0	0	0	0	0	0
	113	113	0s	0	113	113	0	0	0


1
0
3m 27s
2
0
Voyboy
Snoopeh
Cinco


1
0
3m 27s
2
0
Voyboy
Snoopeh
Cinco


SPACETIME STUDIOS

What's New in Social Gaming

What's New in Social Gaming

New in Game Center

- Guest players allow for new modes in multiplayer
- Unified server environment to streamline development and testing

What's New in Social Gaming

New in Game Center

- Guest players allow for new modes in multiplayer
- Unified server environment to streamline development and testing

Introducing ReplayKit

- Records audio and visuals of running application
- Efficient with privacy safeguards
- Compact framework with simple adoption

More Information

Documentation and Videos
http://developer.apple.com

Apple Developer Forums
http://developer.apple.com/forums

Developer Technical Support
http://developer.apple.com/support/technical

General Inquiries
Allan Schaffer, Game Technologies Evangelist
aschaffer@apple.com

Related Labs

ReplayKit Lab	Graphics, Games, and Media Lab D	Wednesday 2:30PM
ReplayKit Lab	Graphics, Games, and Media Lab D	Thursday 9:00AM

WWDC15