Designing for Game Controllers

Session 611
JJ Cwik
Software Engineer

Game Controllers

MFi Specification

For third-party controller vendors

Defines hardware requirements

Gives confidence controllers work for all games which support controllers

Game Controller Framework

Add controller support to your game

iOS and OS X

Simple API

- Detect controllers
- Read inputs

One API for all controllers

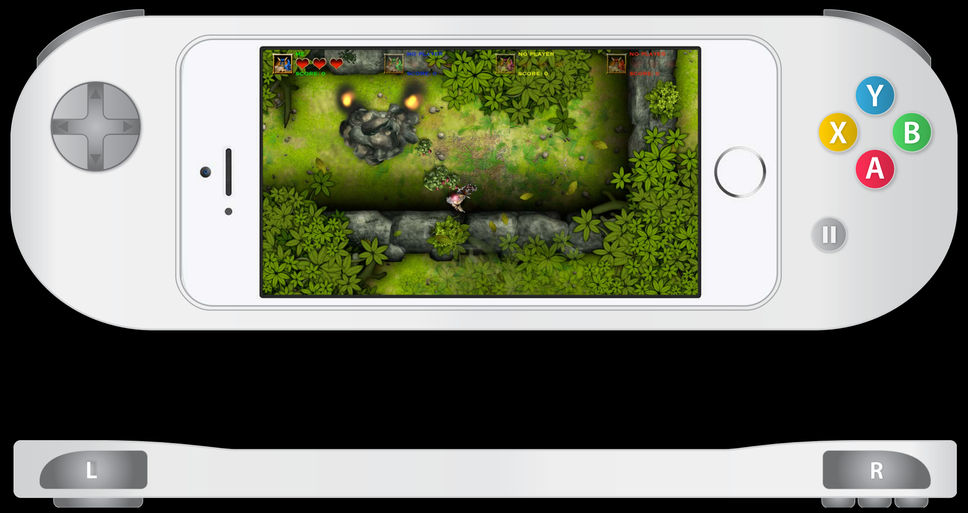
Form-Fitting Standard Controller

Form-fitting

- Encases device
- Touch, motion

Standard control layout

- D-pad
- Buttons—A, B, X, Y
- Shoulder buttons—L, R



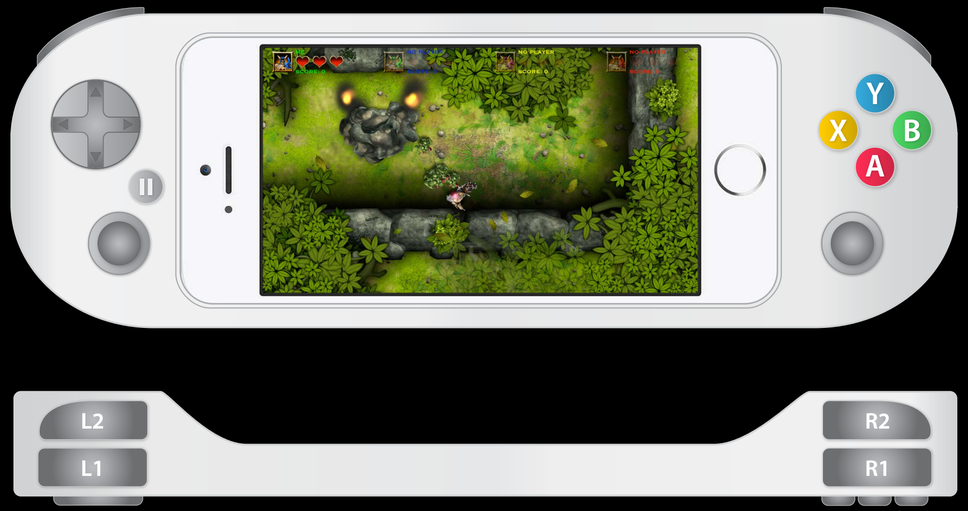
Form-Fitting Extended Controller

Form-fitting

- Encases device
- Touch, motion

Extended control layout

- D-pad
- Buttons—A, B, X, Y
- Shoulder buttons—L1, R1
- Thumbsticks
- Triggers—L2, R2



Form-Fitting Extended Controller

Form-fitting

- Encases device
- Touch, motion

Extended control layout

- D-pad
- Buttons—A, B, X, Y
- Shoulder buttons—L1, R1
- Thumbsticks
- Triggers—L2, R2

Standalone Extended Controller

Standalone

- Not attached to device
- Input only from controller

Extended control layout

- D-pad
- Buttons—A, B, X, Y
- Shoulder buttons—L1, R1
- Thumbsticks
- Triggers—L2, R2

Standalone Extended Controller

Standalone

- Not attached to device
- Input only from controller

Extended control layout

- D-pad
- Buttons—A, B, X, Y
- Shoulder buttons—L1, R1
- Thumbsticks
- Triggers—L2, R2

Agenda

Overview

Finding controllers

Reading controller input

What's new

Design guidance

Finding Controllers

GCController

Main class

Same class for all controllers

Provides

- Finding controllers
- Reading input
- Controller information

Finding Controllers

```
@interface GCController : NSObject
+ (NSArray *)controllers;
...
```

Currently-connected controllers

Array of GCController instances (empty if none)

Updated when controllers connect and disconnect

Finding Controllers

```
-application:didFinishLaunchingWithOptions:
```

```
-setupControllers:
```

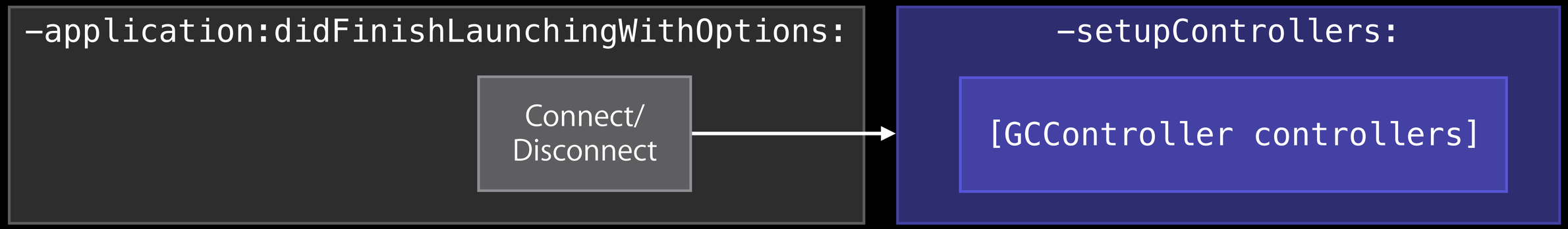
Finding Controllers

```
-application:didFinishLaunchingWithOptions:
```

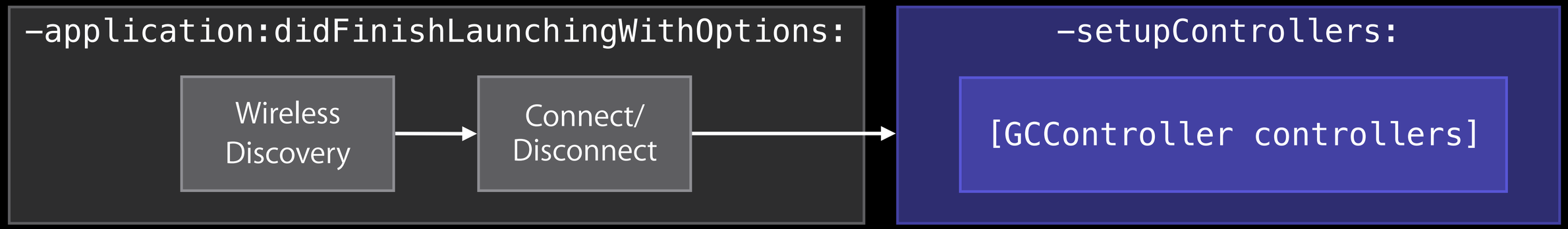
```
-setupControllers:

    [GCController controllers]
```

Finding Controllers



Finding Controllers



Finding Controllers

```
-(BOOL)application:(UIApplication*)app didFinishLaunchingWithOptions:
(NSDictionary*)dict
{
    if ([GCController class])
    {
        NSNotificationCenter* center = [NSNotificationCenter defaultCenter];

        [center addObserver:self selector:@selector(setupControllers:)
            name:GCControllerDidConnectNotification object:nil];
        [center addObserver:self selector:@selector(setupControllers:)
            name:GCControllerDidDisconnectNotification object:nil];

        [GCController startWirelessControllerDiscoveryWithCompletionHandler:^{...}];
    }
}
```

Finding Controllers

```
-(BOOL)application:(UIApplication*)app didFinishLaunchingWithOptions:
(NSDictionary*)dict
{
    if ([GCController class])
    {
        NSNotificationCenter* center = [NSNotificationCenter defaultCenter];

        [center addObserver:self selector:@selector(setupControllers:)
            name:GCControllerDidConnectNotification object:nil];
        [center addObserver:self selector:@selector(setupControllers:)
            name:GCControllerDidDisconnectNotification object:nil];

        [GCController startWirelessControllerDiscoveryWithCompletionHandler:^{...}];
    }
}
```

Finding Controllers

```
-(BOOL)application:(UIApplication*)app didFinishLaunchingWithOptions:
(NSDictionary*)dict
{
    if ([GCController class])
    {
        NSNotificationCenter* center = [NSNotificationCenter defaultCenter];

        [center addObserver:self selector:@selector(setupControllers:)
            name:GCControllerDidConnectNotification object:nil];
        [center addObserver:self selector:@selector(setupControllers:)
            name:GCControllerDidDisconnectNotification object:nil];

        [GCController startWirelessControllerDiscoveryWithCompletionHandler:^{...}];
    }
}
```

Finding Controllers

```
-(BOOL)application:(UIApplication*)app didFinishLaunchingWithOptions:
(NSDictionary*)dict
{
    if ([GCController class])
    {
        NSNotificationCenter* center = [NSNotificationCenter defaultCenter];

        [center addObserver:self selector:@selector(setupControllers:)
            name:GCControllerDidConnectNotification object:nil];
        [center addObserver:self selector:@selector(setupControllers:)
            name:GCControllerDidDisconnectNotification object:nil];

        [GCController startWirelessControllerDiscoveryWithCompletionHandler:^{...}];
    }
}
```

Finding Controllers

```
- (void)setupControllers:(NSNotification *)notification
{
    // Get array of connected controllers
    self.controllerArray = [GCController controllers];

    if ([self.controllerArray count] > 0)
        // Found controllers
    ...
}
```

Finding Controllers

```
- (void)setupControllers:(NSNotification *)notification
{
    // Get array of connected controllers
    self.controllerArray = [GCController controllers];

    if ([self.controllerArray count] > 0)
        // Found controllers
    ...
}
```

Finding Controllers

```
- (void)setupControllers:(NSNotification *)notification
{
    // Get array of connected controllers
    self.controllerArray = [GCController controllers];

    if ([self.controllerArray count] > 0)
        // Found controllers
    ...
}
```

Finding Controllers

```
- (void)setupControllers:(NSNotification *)notification
{
    // Get array of connected controllers
    self.controllerArray = [GCController controllers];

    if ([self.controllerArray count] > 0)
        // Found controllers
    ...
}
```

Design Guidance

Graceful connecting and disconnecting

When connected

- Move to controller-based input
- Remove onscreen virtual controls (including pause button)
- Set playerIndex to illuminate player indicator LEDs

When disconnecting

- Pause gameplay
- Return to regular controls

Design Guidance

Treat connected controller as intent to use

Design Guidance

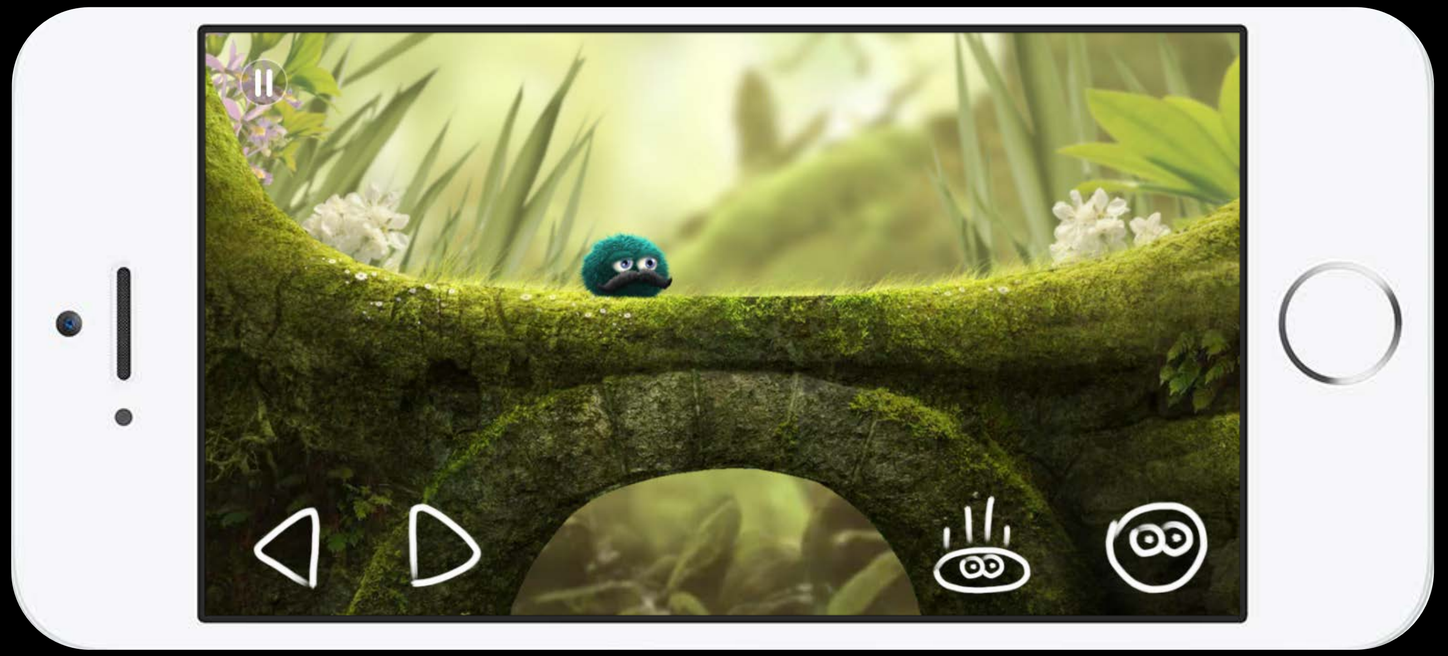
Treat connected controller as intent to use

Design Guidance

Treat connected controller as intent to use

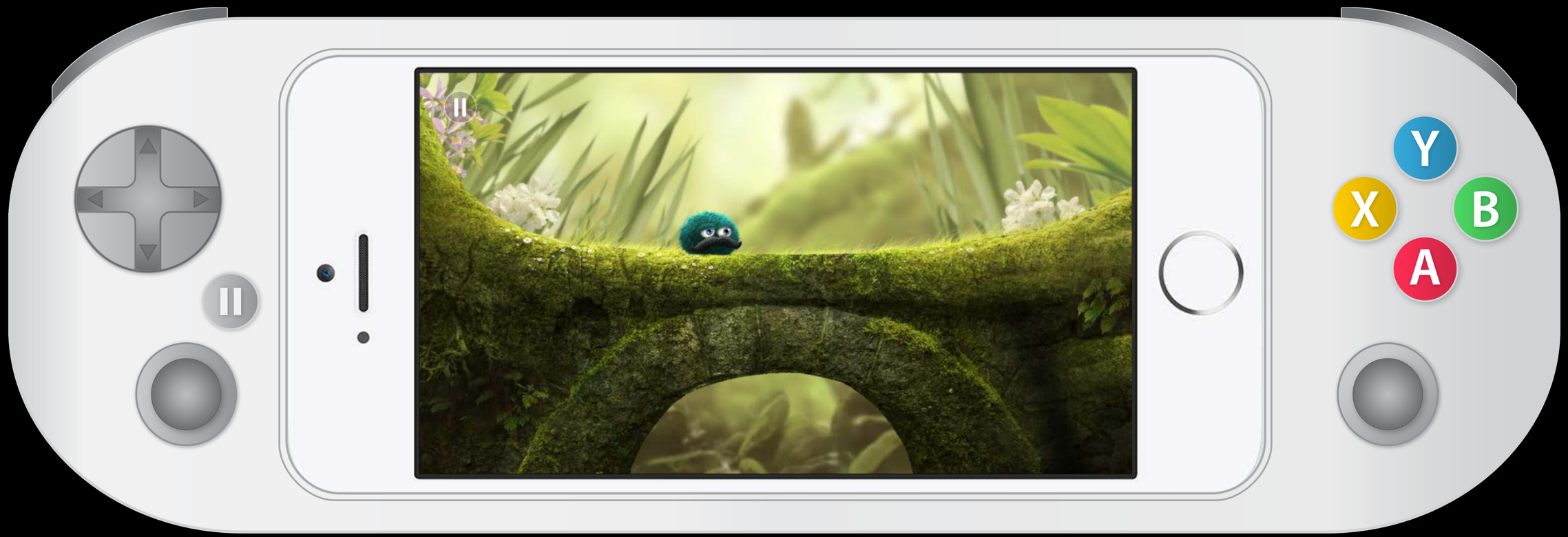
Design Guidance

Treat connected controller as intent to use



Design Guidance

Treat connected controller as intent to use



Reading Controller Input

Design Guidance

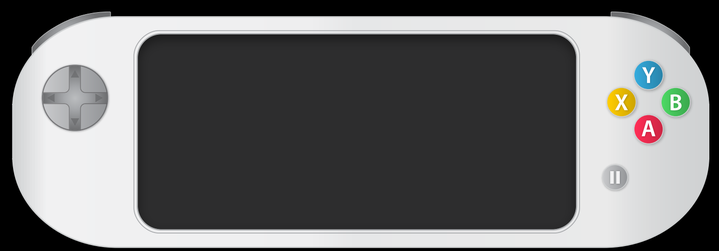
First design for touch

Multi-touch and motion

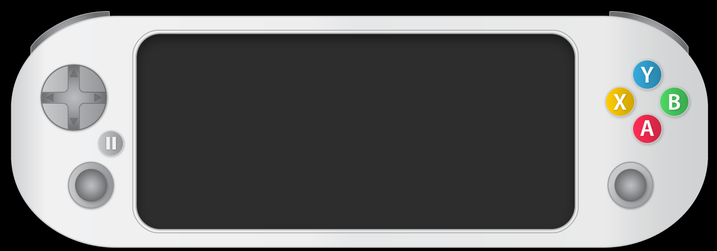
Game controllers

Profiles

Profiles



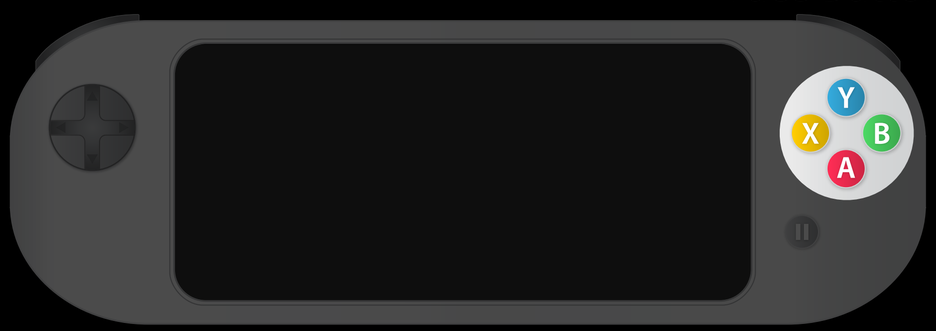

```
myController.gamepad
```




```
myController.gamepad
```

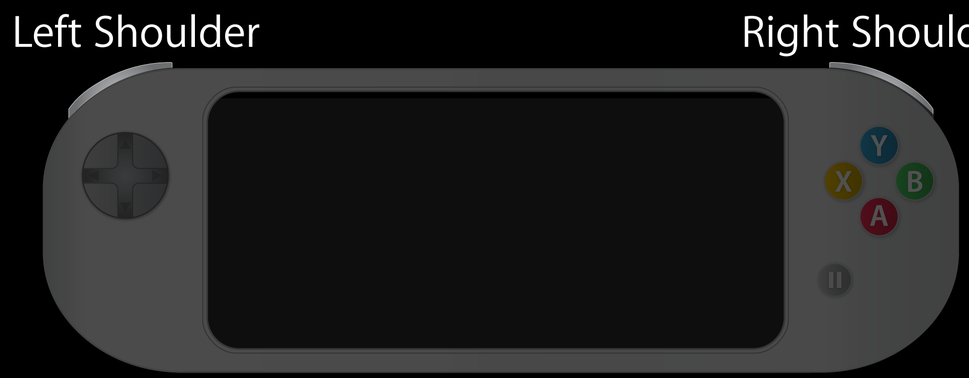
```
myController.extendedGamepad
```

Gamepad Profile



Property Name	Type
`buttonA`	`GCControllerButtonInput`
`buttonB`	
`buttonX`	
`buttonY`	

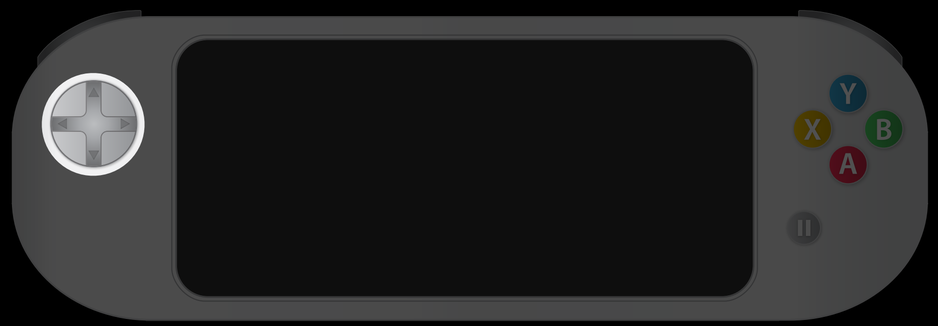
Gamepad Profile



Property Name	Type
buttonA buttonB buttonX buttonY	GCControllerButtonInput
leftShoulder rightShoulder	GCControllerButtonInput

Gamepad Profile

D-Pad

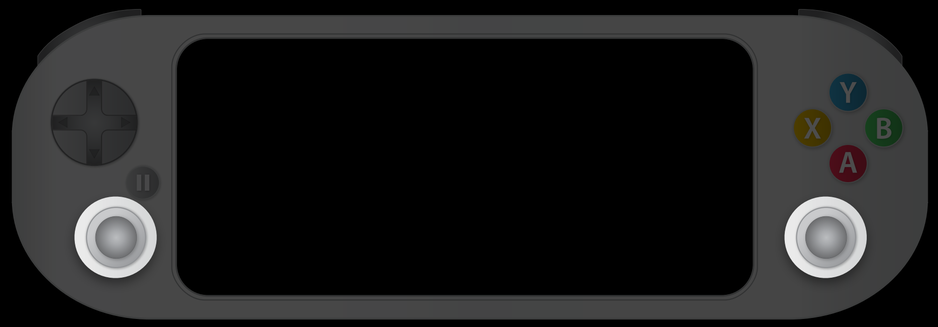


Property Name	Type
buttonA buttonB buttonX buttonY	GCControllerButtonInput
leftShoulder rightShoulder	GCControllerButtonInput
dpad	GCControllerDirectionPad

ExtendedGamepad Profile

Property Name	Type
buttonA buttonB buttonX buttonY	GCControllerButtonInput
leftShoulder rightShoulder	GCControllerButtonInput
dpad	GCControllerDirectionPad

ExtendedGamepad Profile



Left Thumbstick

Right Thumbstick

Property Name	Type
buttonA buttonB buttonX buttonY	GCControllerButtonInput
leftShoulder rightShoulder	GCControllerButtonInput
dpad	GCControllerDirectionPad
leftThumbstick rightThumbstick	GCControllerDirectionPad

ExtendedGamepad Profile

Left Trigger

Right Trigger

Property Name	Type
buttonA buttonB buttonX buttonY	GCControllerButtonInput
leftShoulder rightShoulder	GCControllerButtonInput
dpad	GCControllerDirectionPad
leftThumbstick rightThumbstick	GCControllerDirectionPad
leftTrigger rightTrigger	GCControllerButtonInput

ExtendedGamepad Profile

Property Name	Type
buttonA buttonB buttonX buttonY	GCControllerButtonInput
leftShoulder rightShoulder	GCControllerButtonInput
dpad	GCControllerDirectionPad
leftThumbstick rightThumbstick	GCControllerDirectionPad
leftTrigger rightTrigger	GCControllerButtonInput

ExtendedGamepad Profile

Property Name	Type
buttonA buttonB buttonX buttonY	GCControllerButtonInput
leftShoulder rightShoulder	GCControllerButtonInput
dpad	GCControllerDirectionPad
leftThumbstick rightThumbstick	GCControllerDirectionPad
leftTrigger rightTrigger	GCControllerButtonInput

GCControllerButtonInput

Classic button state

```
BOOL pressed;   // digital
```

Buttons are also pressure-sensitive

```
float value;   // analog, 0.0 to 1.0
```

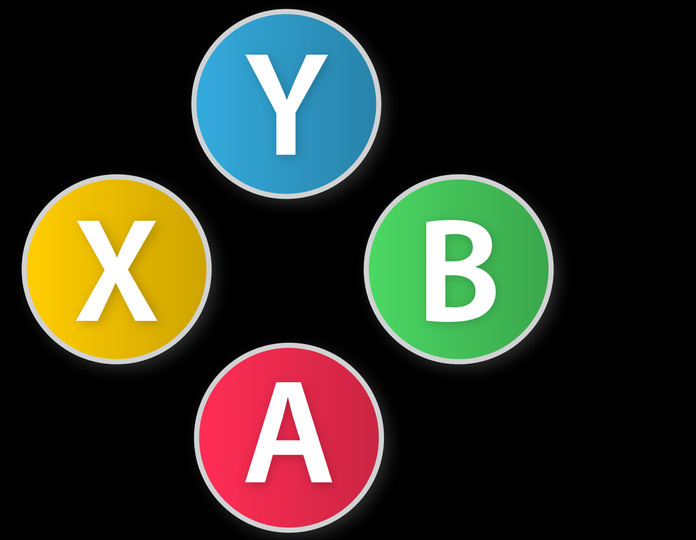
GCControllerButtonInput

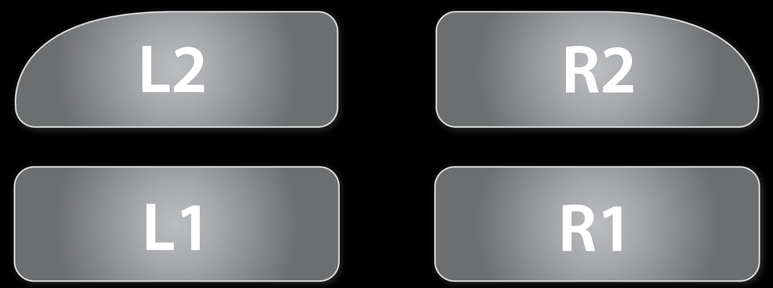
Classic button state

```
BOOL pressed;   // digital
```

Buttons are also pressure-sensitive

```
float value;   // analog, 0.0 to 1.0
```





Design Guidance

Observe button conventions

A button—Primary action

B button—Secondary action

What are your game's primary and secondary actions?

- In-game (jump? shoot?)
- UI (accept, cancel)

Design Guidance

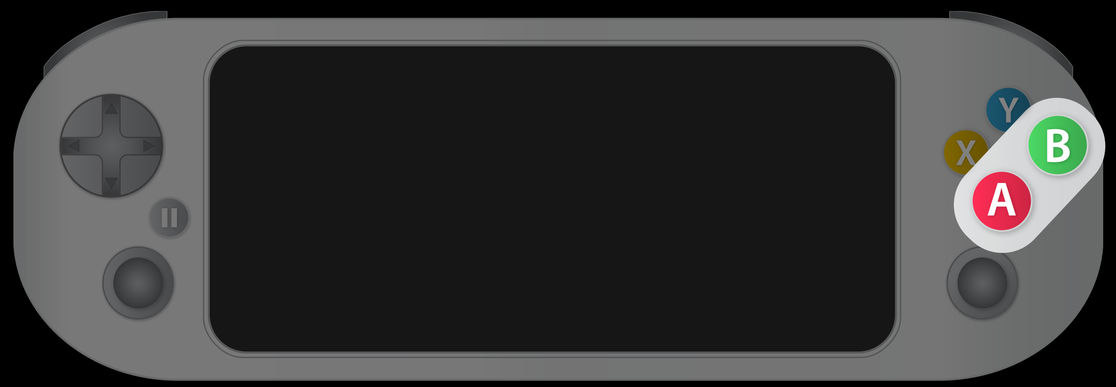
Observe button conventions

A button—Primary action

B button—Secondary action

What are your game's primary and secondary actions?

- In-game (jump? shoot?)
- UI (accept, cancel)



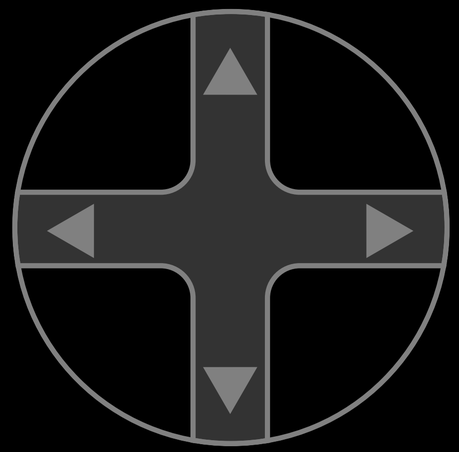
GCControllerDirectionPad

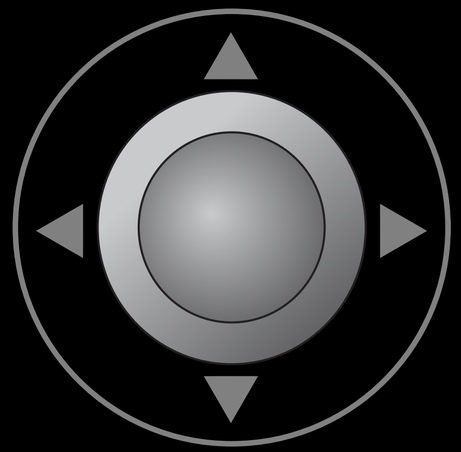
Treated as four buttons

```
GCControllerButtonInput *up, *down, *left, *right;
```

Or as two axes

```
GCControllerAxisInput *xAxis, *yAxis;
```





GCControllerAxisInput

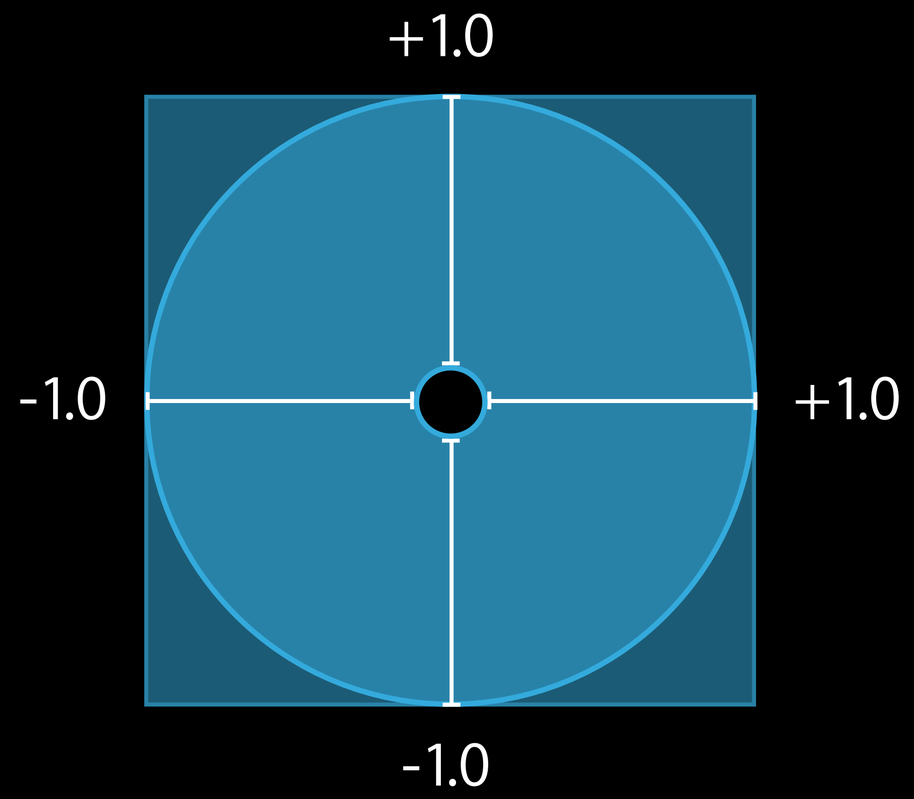
Measures movement along an axis

```
float value;   // −1.0 to +1.0, neutral is 0.0
```

Minimum range is unit circle

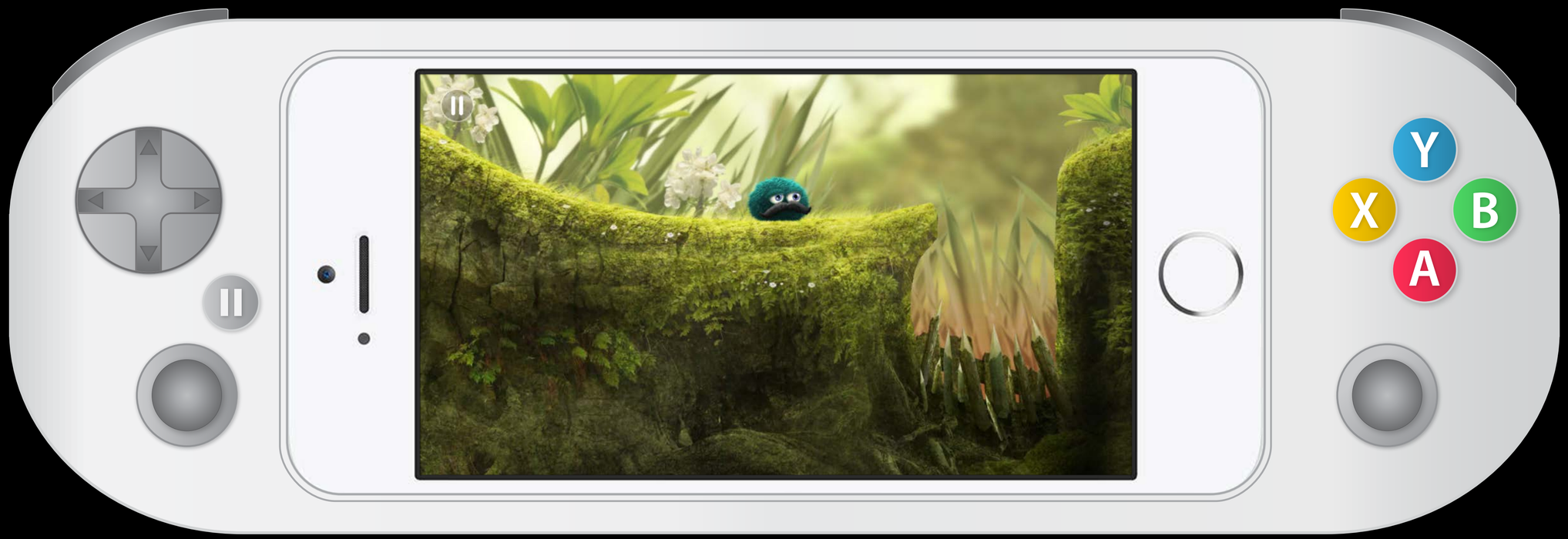
Automatic calibration

Automatic dead-zone handling



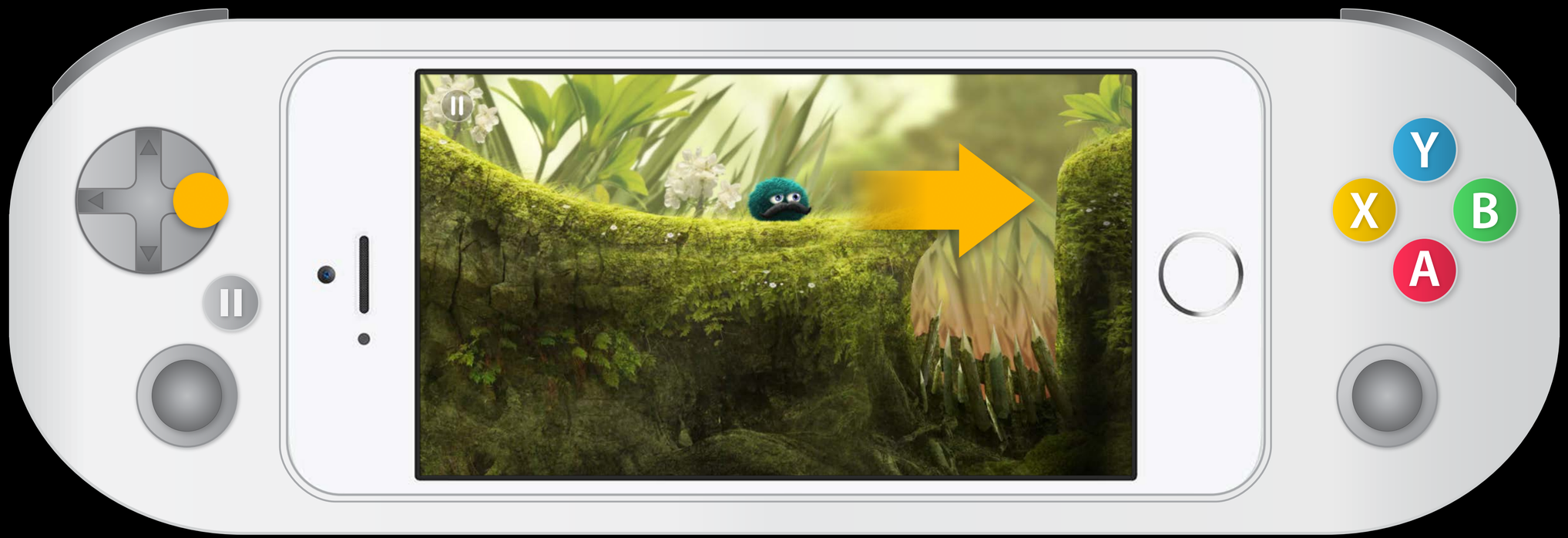
Design Guidance

Group logically-similar actions



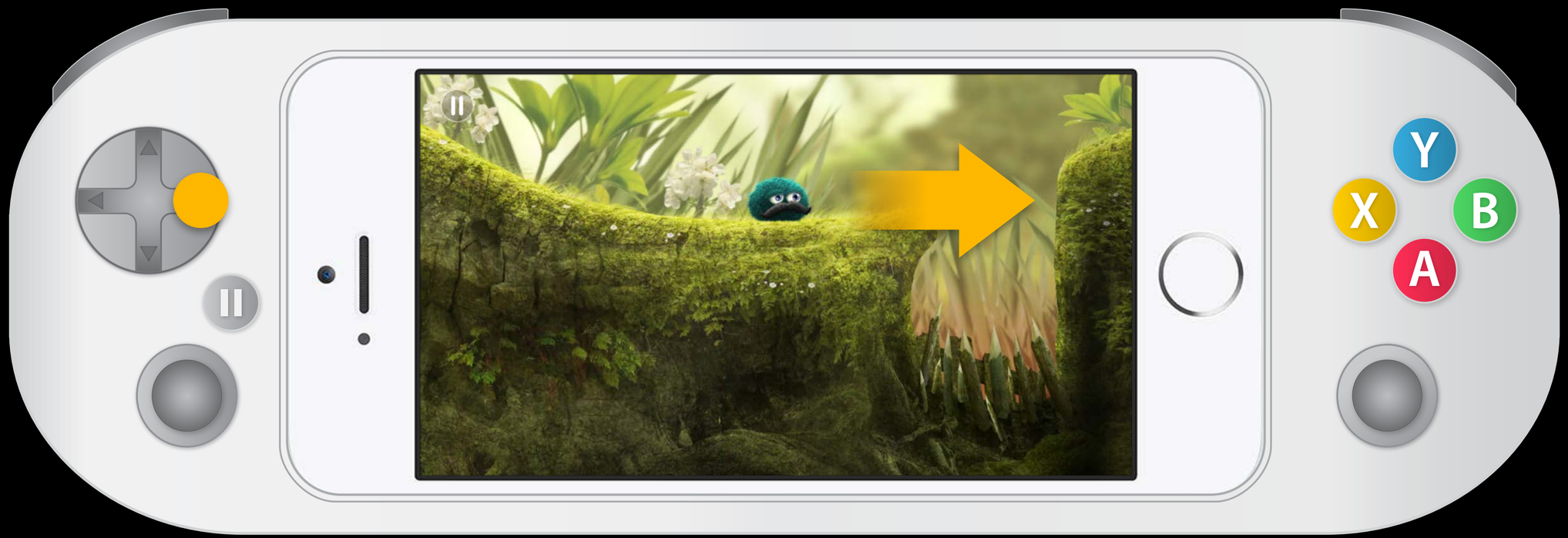
Design Guidance

Group logically-similar actions



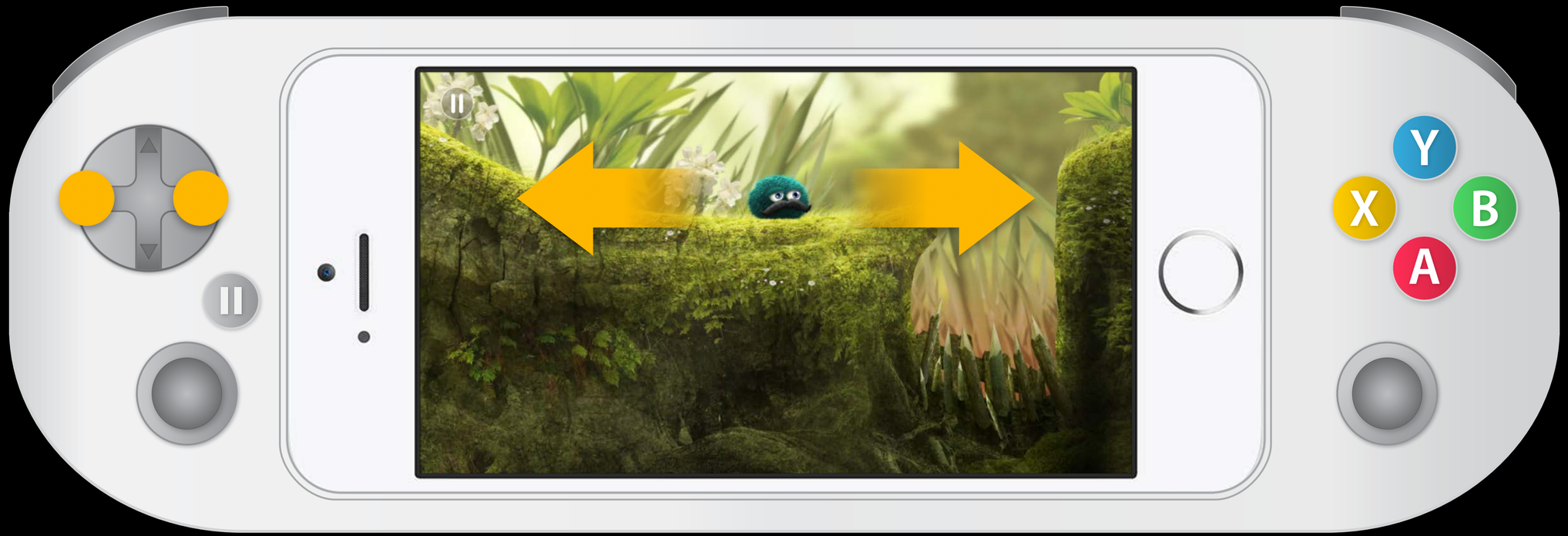
Design Guidance

Group logically-similar actions



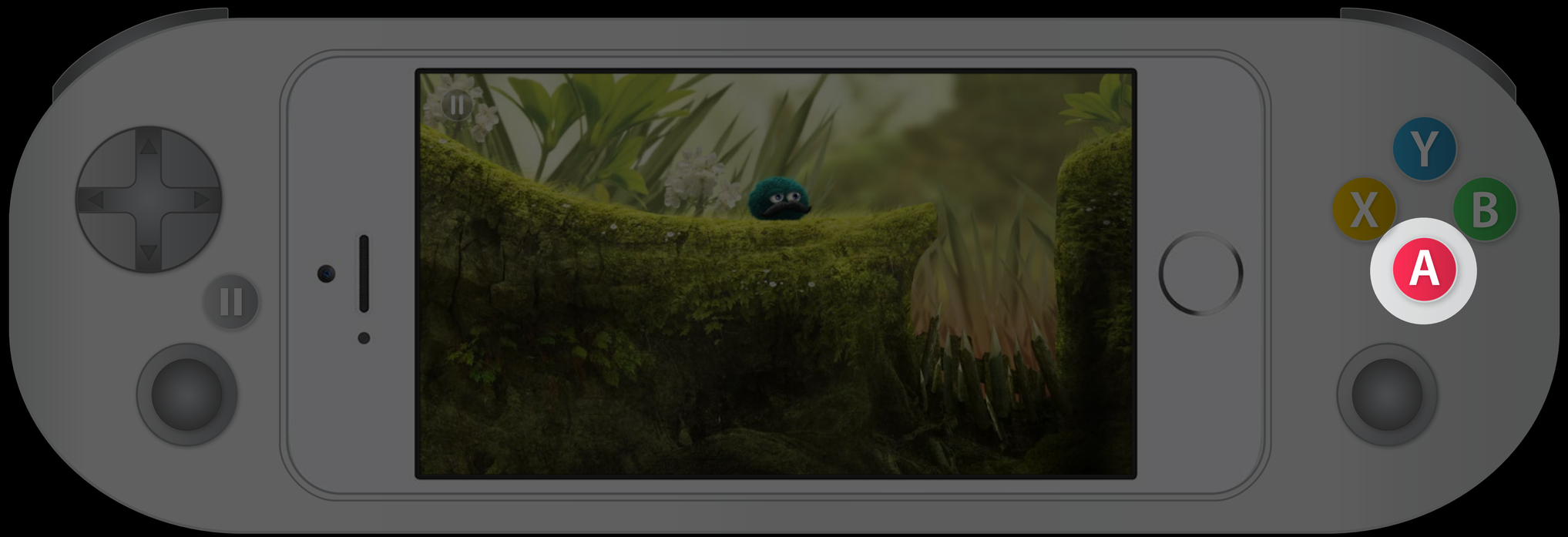
Design Guidance

Group logically-similar actions



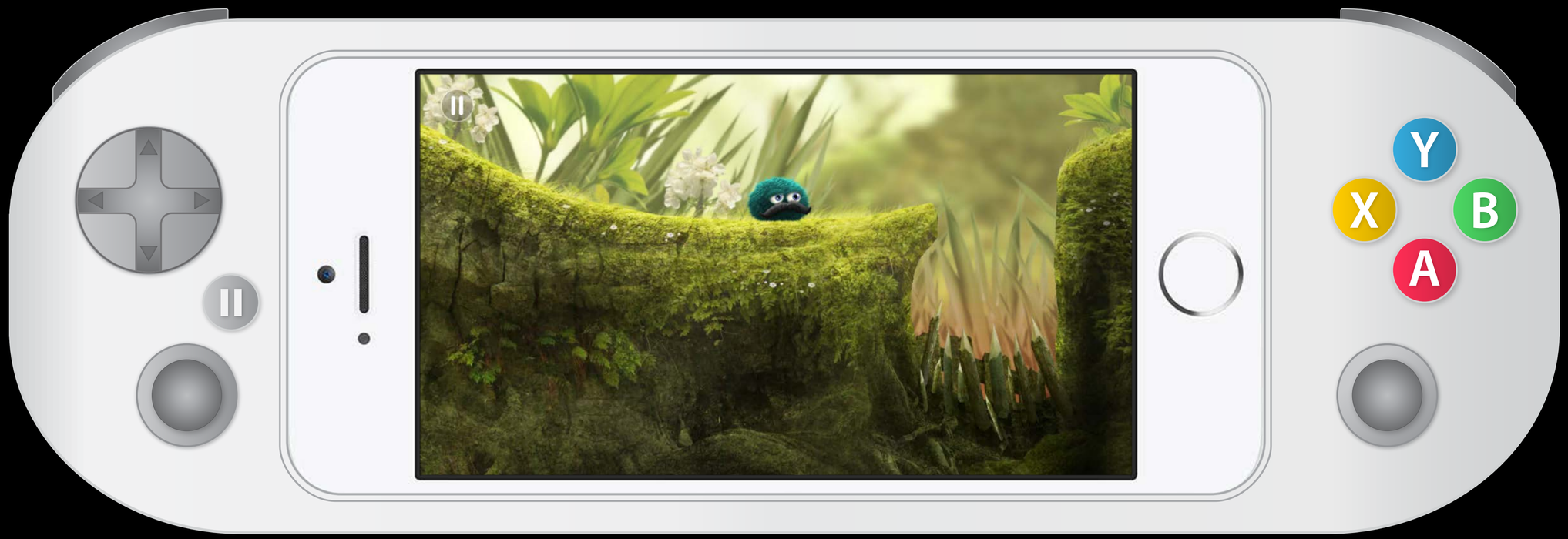
Design Guidance

Group logically-similar actions



Design Guidance

Group logically-similar actions



Design Guidance

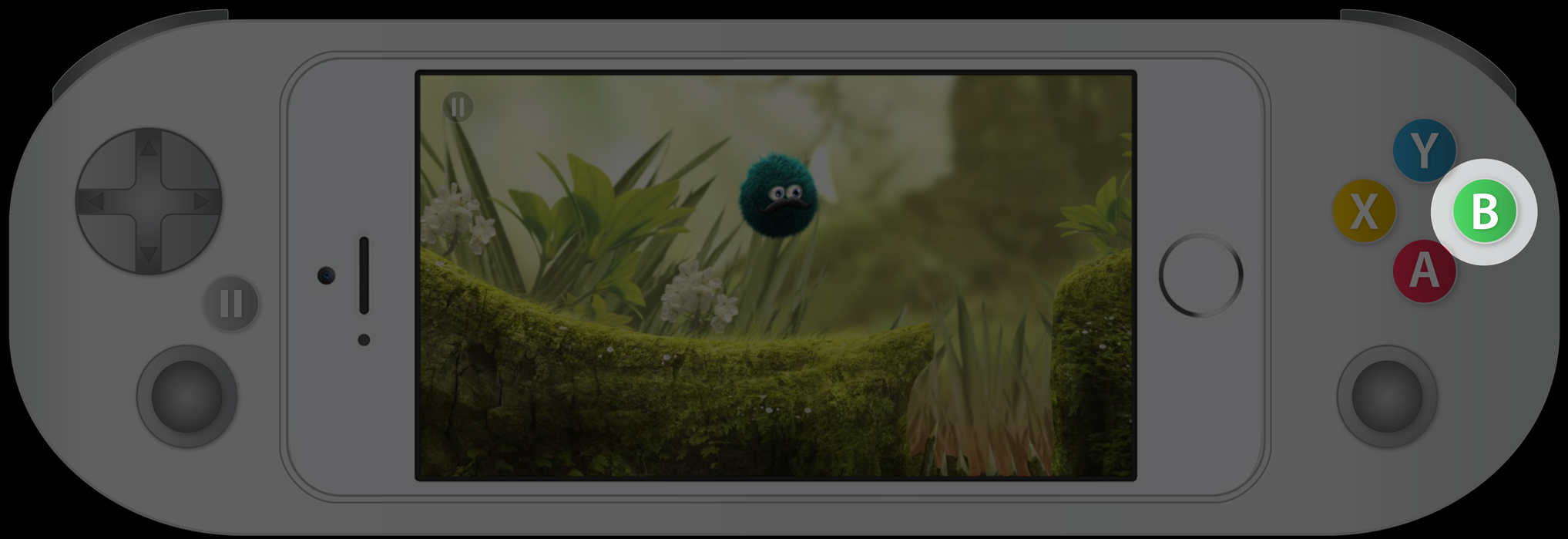
Group logically-similar actions

Design Guidance

Group logically-similar actions

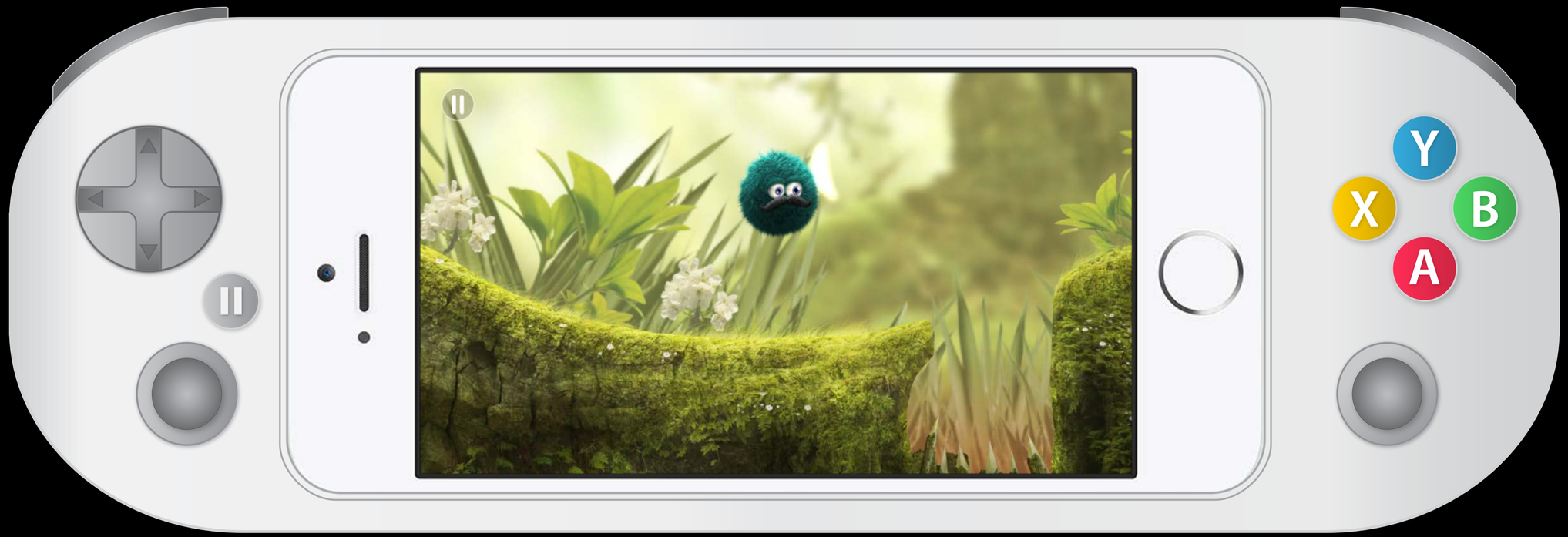
Design Guidance

Group logically-similar actions



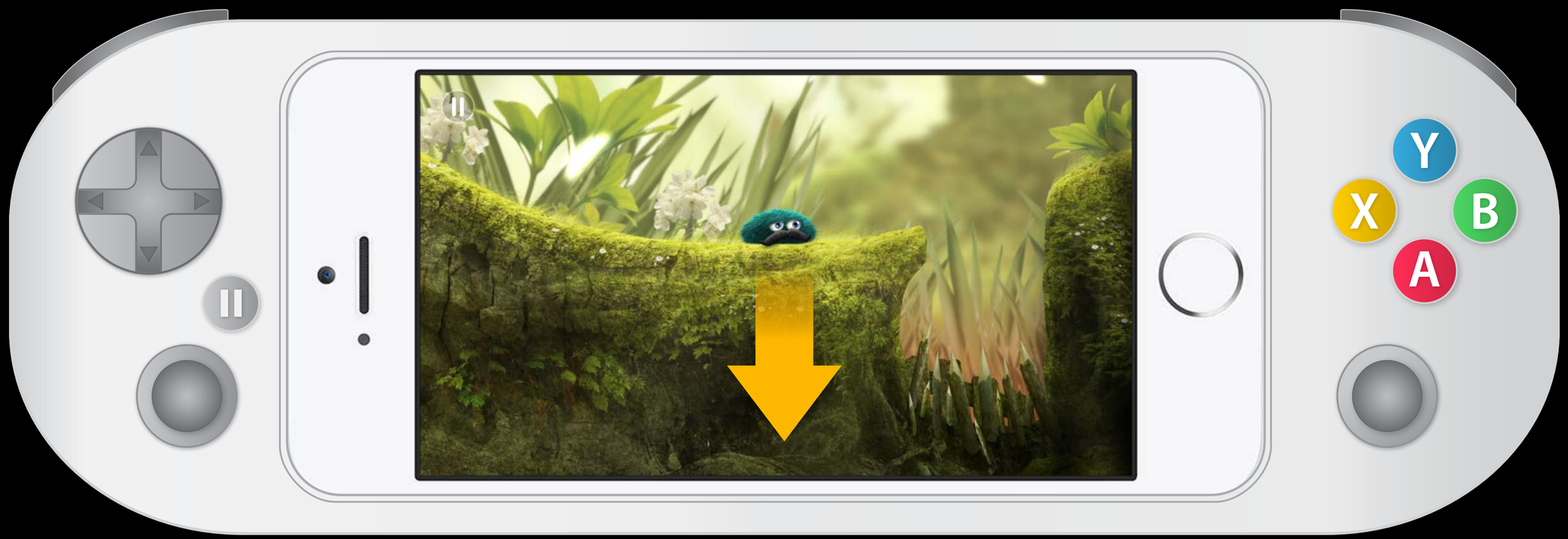
Design Guidance

Group logically-similar actions



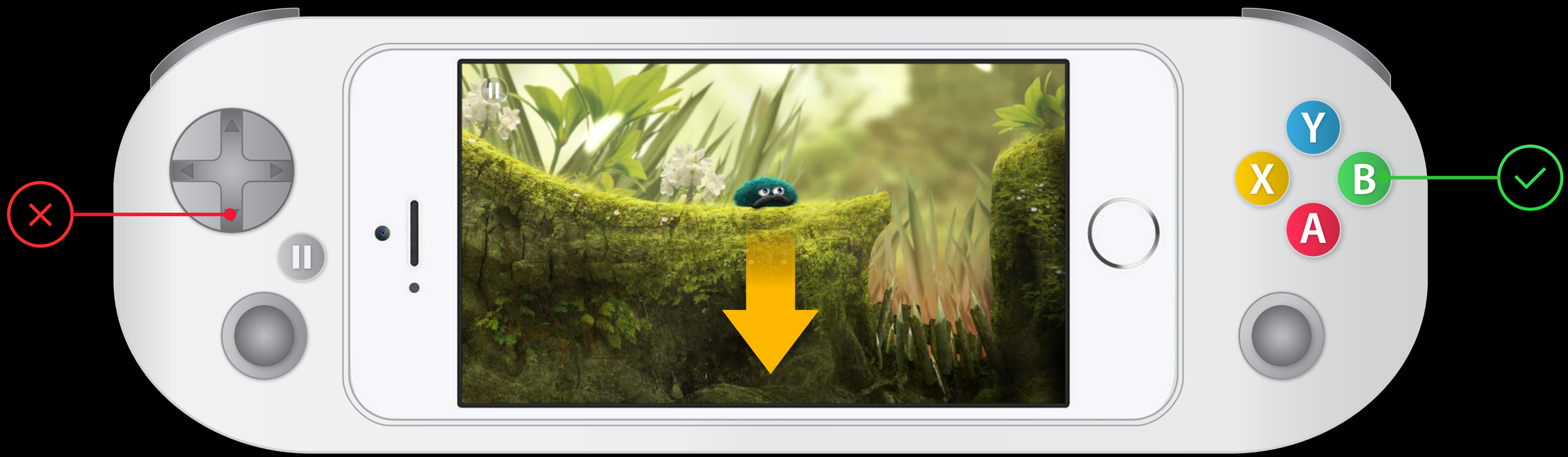
Design Guidance

Group logically-similar actions



Design Guidance

Group logically-similar actions



Reading Input—Polling

```
-(void)readControls
{
    // Weapons
    if (myController.gamepad.buttonA.pressed)
        [self fireLasers];
    if (myController.gamepad.buttonB.pressed)
        [self fireMissilesAtRate:myController.gamepad.buttonB.value];

    // Thrust
    [self applyThrust:myController.gamepad.dpad.yAxis.value];

    // Camera
    if (myController.extendedGamepad) {
        // this is an Extended controller
        GCControllerDirectionPad *r = myController.extendedGamepad.rightThumbstick;
        [self freeLookX:r.xAxis.value Y:r.yAxis.value];
    ...
```

Reading Input—Polling

```
-(void)readControls
{
    // Weapons
    if (myController.gamepad.buttonA.pressed)
        [self fireLasers];
    if (myController.gamepad.buttonB.pressed)
        [self fireMissilesAtRate:myController.gamepad.buttonB.value];

    // Thrust
    [self applyThrust:myController.gamepad.dpad.yAxis.value];

    // Camera
    if (myController.extendedGamepad) {
        // this is an Extended controller
        GCControllerDirectionPad *r = myController.extendedGamepad.rightThumbstick;
        [self freeLookX:r.xAxis.value Y:r.yAxis.value];
    ...
```

Reading Input—Polling

```
-(void)readControls
{
    // Weapons
    if (myController.gamepad.buttonA.pressed)
        [self fireLasers];
    if (myController.gamepad.buttonB.pressed)
        [self fireMissilesAtRate:myController.gamepad.buttonB.value];

    // Thrust
    [self applyThrust:myController.gamepad.dpad.yAxis.value];

    // Camera
    if (myController.extendedGamepad) {
        // this is an Extended controller
        GCControllerDirectionPad *r = myController.extendedGamepad.rightThumbstick;
        [self freeLookX:r.xAxis.value Y:r.yAxis.value];
    ...
```

Reading Input—Polling

```
-(void)readControls
{
    // Weapons
    if (myController.gamepad.buttonA.pressed)
        [self fireLasers];
    if (myController.gamepad.buttonB.pressed)
        [self fireMissilesAtRate:myController.gamepad.buttonB.value];

    // Thrust
    [self applyThrust:myController.gamepad.dpad.yAxis.value];

    // Camera
    if (myController.extendedGamepad) {
        // this is an Extended controller
        GCControllerDirectionPad *r = myController.extendedGamepad.rightThumbstick;
        [self freeLookX:r.xAxis.value Y:r.yAxis.value];
    ...
```

Reading Input—Polling

```
-(void)readControls
{
    // Weapons
    if (myController.gamepad.buttonA.pressed)
        [self fireLasers];
    if (myController.gamepad.buttonB.pressed)
        [self fireMissilesAtRate:myController.gamepad.buttonB.value];

    // Thrust
    [self applyThrust:myController.gamepad.dpad.yAxis.value];

    // Camera
    if (myController.extendedGamepad) {
        // this is an Extended controller
        GCControllerDirectionPad *r = myController.extendedGamepad.rightThumbstick;
        [self freeLookX:r.xAxis.value Y:r.yAxis.value];
    ...
```

Reading Input—Polling

```
-(void)readControls
{
    // Weapons
    if (myController.gamepad.buttonA.pressed)
        [self fireLasers];
    if (myController.gamepad.buttonB.pressed)
        [self fireMissilesAtRate:myController.gamepad.buttonB.value];

    // Thrust
    [self applyThrust:myController.gamepad.dpad.yAxis.value];

    // Camera
    if (myController.extendedGamepad) {
        // this is an Extended controller
        GCControllerDirectionPad *r = myController.extendedGamepad.rightThumbstick;
        [self freeLookX:r.xAxis.value Y:r.yAxis.value];
    ...
```

Reading Input—Polling

```
-(void)readControls
{
    // Weapons
    if (myController.gamepad.buttonA.pressed)
        [self fireLasers];
    if (myController.gamepad.buttonB.pressed)
        [self fireMissilesAtRate:myController.gamepad.buttonB.value];

    // Thrust
    [self applyThrust:myController.gamepad.dpad.yAxis.value];

    // Camera
    if (myController.extendedGamepad) {
        // this is an Extended controller
        GCControllerDirectionPad *r = myController.extendedGamepad.rightThumbstick;
        [self freeLookX:r.xAxis.value Y:r.yAxis.value];
    ...
```

Reading Input—Events

Be notified when inputs change

- Buttons
- Axes
- D-pad/thumbsticks
- Profiles

Register block as change handler

Reading Input—Events

Value changed handlers

- For analog changes

```
myController.gamepad.buttonY.valueChangedHandler
```

Reading Input—Events

Value changed handlers

- For analog changes

```
myController.gamepad.buttonY.valueChangedHandler
```

Pressed changed handlers

- For digital changes

```
myController.gamepad.buttonY.pressedChangedHandler
```

Reading Inputs—Events

NEW

```
myController.gamepad.buttonY.pressedChangedHandler =
    ^(GCControllerButtonInput *button, float value, BOOL pressed)
{
    if (pressed)
        // button pressed
        [self chargeSpeedBurst];
    else
        // button released
        [self doSpeedBurst];
};
```

Reading Inputs—Events

NEW

```
myController.gamepad.buttonY.pressedChangedHandler =
    ^(GCControllerButtonInput *button, float value, BOOL pressed)
{
    if (pressed)
        // button pressed
        [self chargeSpeedBurst];
    else
        // button released
        [self doSpeedBurst];
};
```

Reading Inputs—Events

NEW

```
myController.gamepad.buttonY.pressedChangedHandler =
    ^(GCControllerButtonInput *button, float value, BOOL pressed)
{
    if (pressed)
        // button pressed
        [self chargeSpeedBurst];
    else
        // button released
        [self doSpeedBurst];
};
```

Reading Inputs—Events

NEW

```
myController.gamepad.buttonY.pressedChangedHandler =
    ^(GCControllerButtonInput *button, float value, BOOL pressed)
{
    if (pressed)
        // button pressed
        [self chargeSpeedBurst];
    else
        // button released
        [self doSpeedBurst];
};
```

Design Guidance

Use pressure-sensitivity with discernment

Don’t use “value” if merely pressing an input is what’s needed

- Use “pressed” instead

Take advantage of pressure-sensitivity

- D-pad—360 degree movement
- Face Buttons—Pass and shoot with varying speed in sports games

Tell player when using pressure-sensitivity

Design Guidance

Be a good teacher

Design Guidance

Be a good teacher

Design Guidance

Be a good teacher

Agenda

Overview

Finding controllers

Reading controller input

What's new

Design guidance

What’s New

Forwarding

NEW

Use iPhone as wireless controller for another device

Works over Bluetooth or Wi-Fi

Both devices signed in to same iCloud account

Forwarding

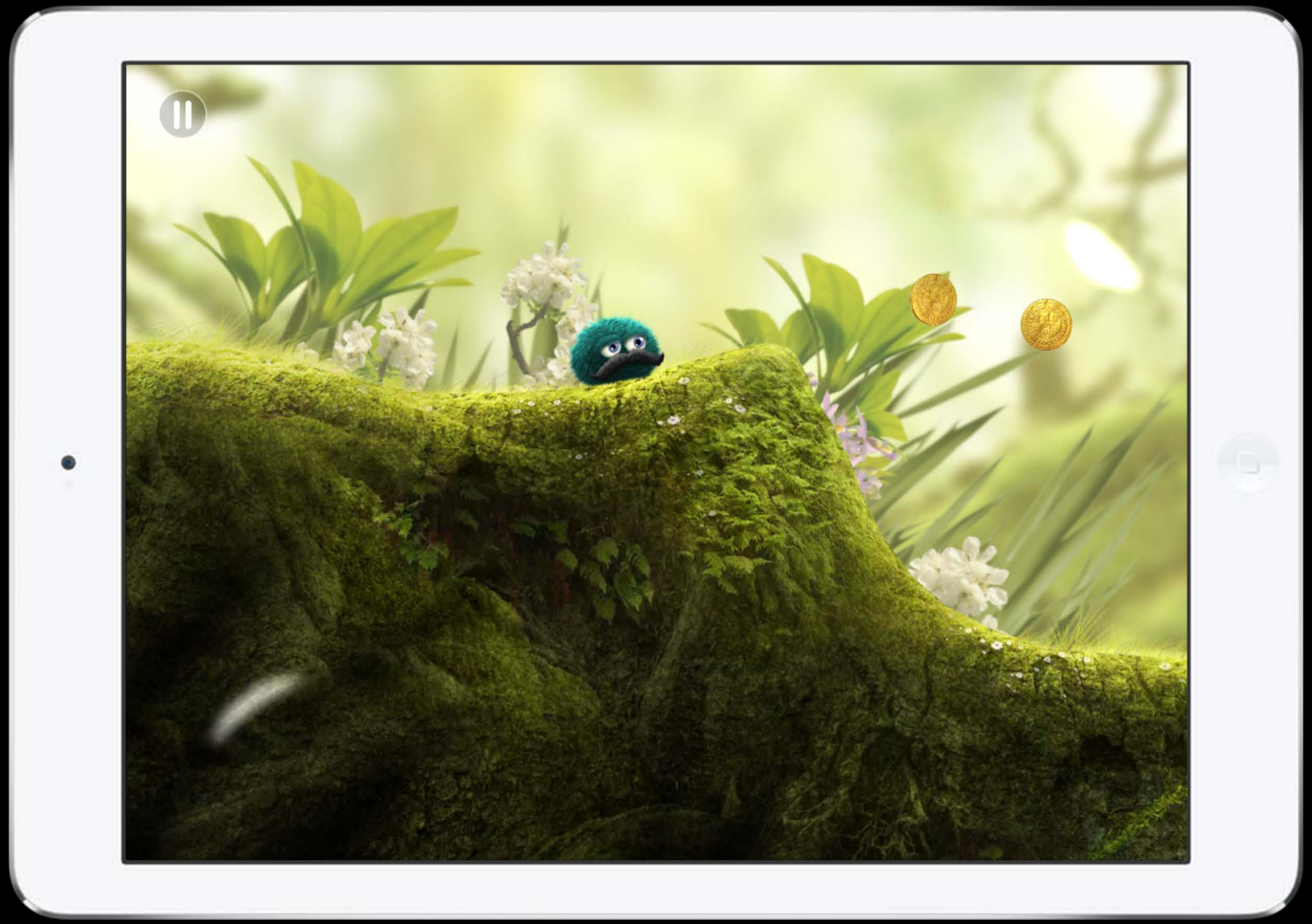
NEW

Use iPhone as wireless controller for another device

Works over Bluetooth or Wi-Fi

Both devices signed in to same iCloud account

Forwarding



Forwarding

Forwarding

Forwarding

Forwarding

Forwarding

iPhone discovered automatically in

```
[GCController startWirelessControllerDiscoveryWithCompletionHandler:^{...}]
```

Appears as wireless controller to your game

Motion Forwarding

NEW

Accelerometer and gyroscope data automatically forwarded

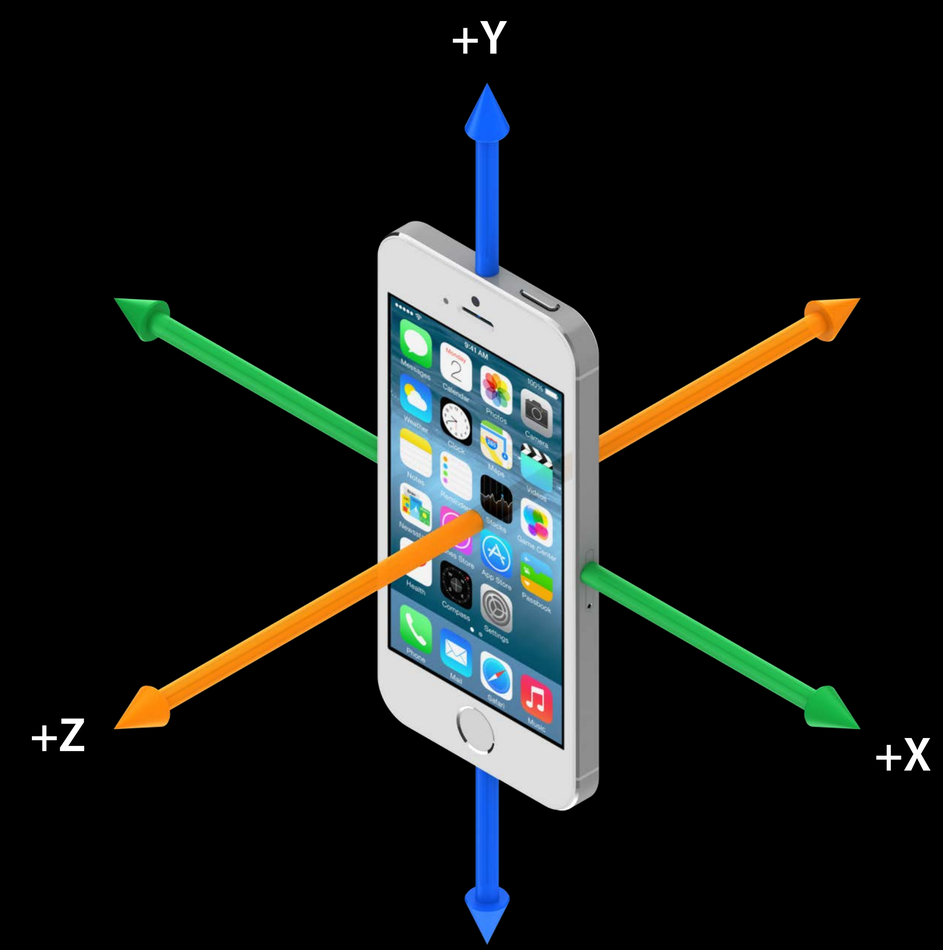
New profile

```
@interface GCController : NSObject
GCMotion *motion;
...
```

Motion Forwarding

GCMotion

```
@interface GCMotion : NSObject
GCAcceleration gravity;
GCAcceleration userAcceleration;
GCQuaternion attitude;
GCRotationRate rotationRate;
...
```



Motion Forwarding

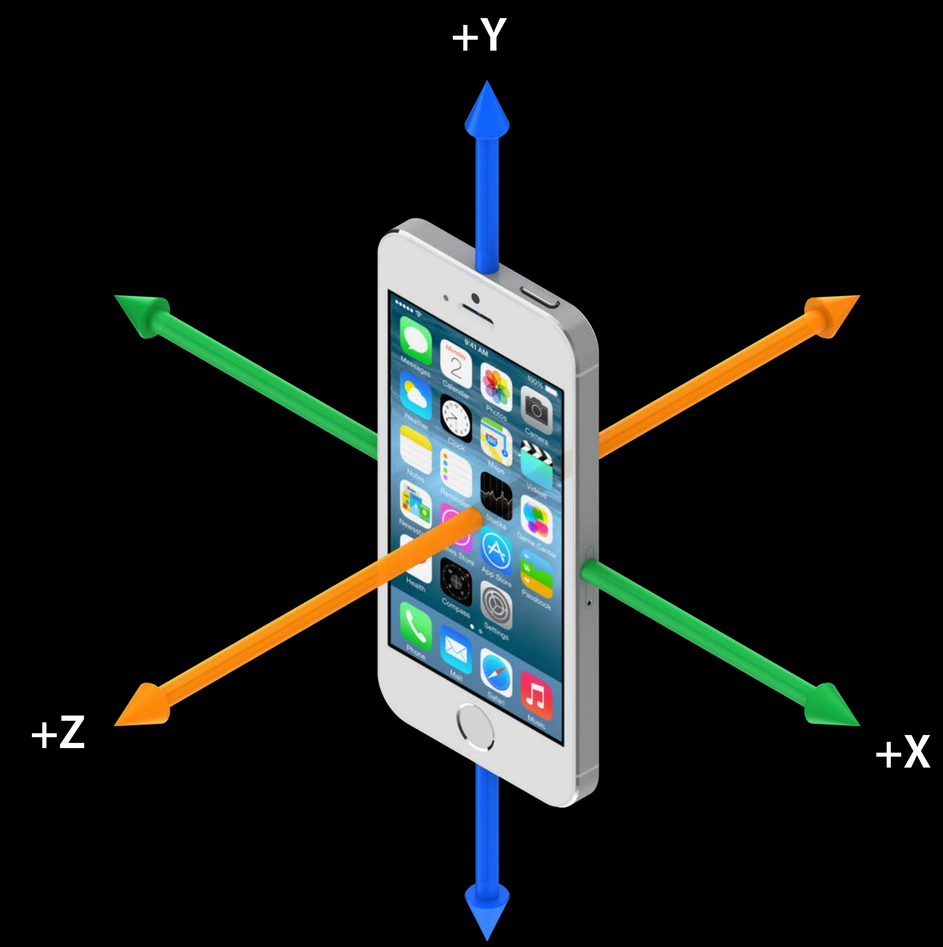
GCAcceleration gravity

Vector which gravity is applying to device

Units are in G's

Flat on table with screen up—(0, 0, -1)

```
typedef struct {
    double x, y, z;
} GCAcceleration;
```



Motion Forwarding

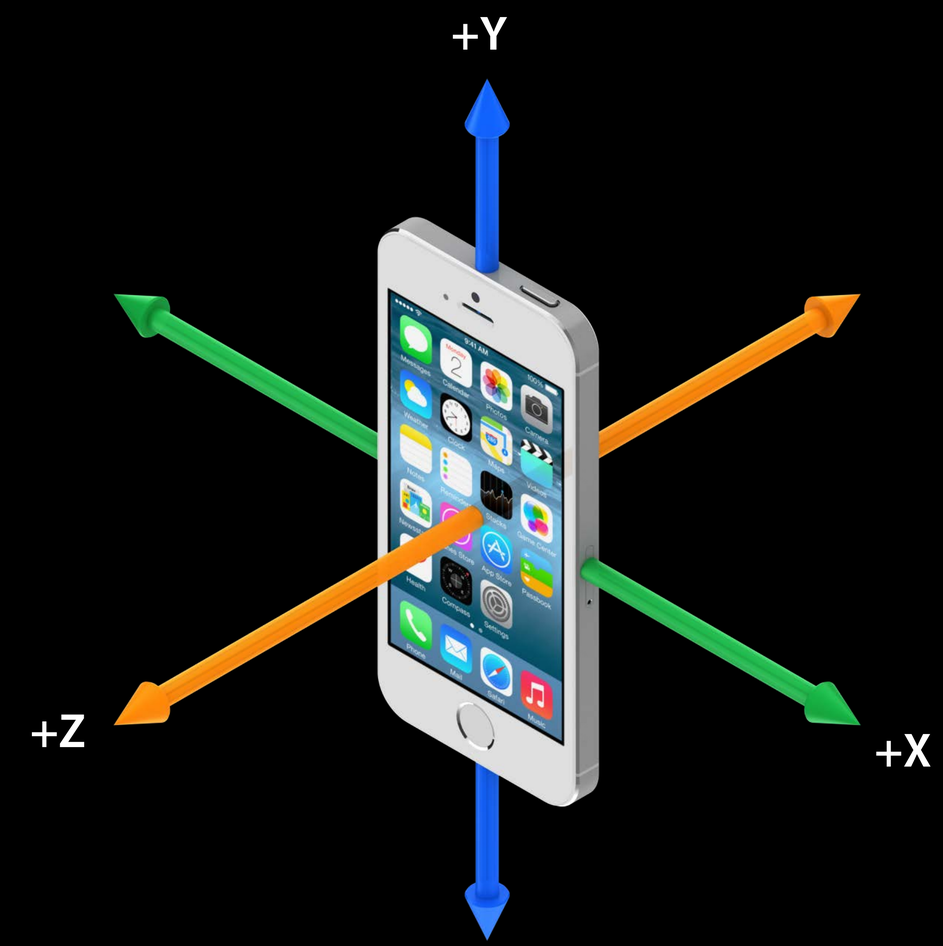
GCAcceleration userAcceleration

Inertial acceleration seen by device

Excludes gravity

Units are in G's

At rest: (0, 0, 0)



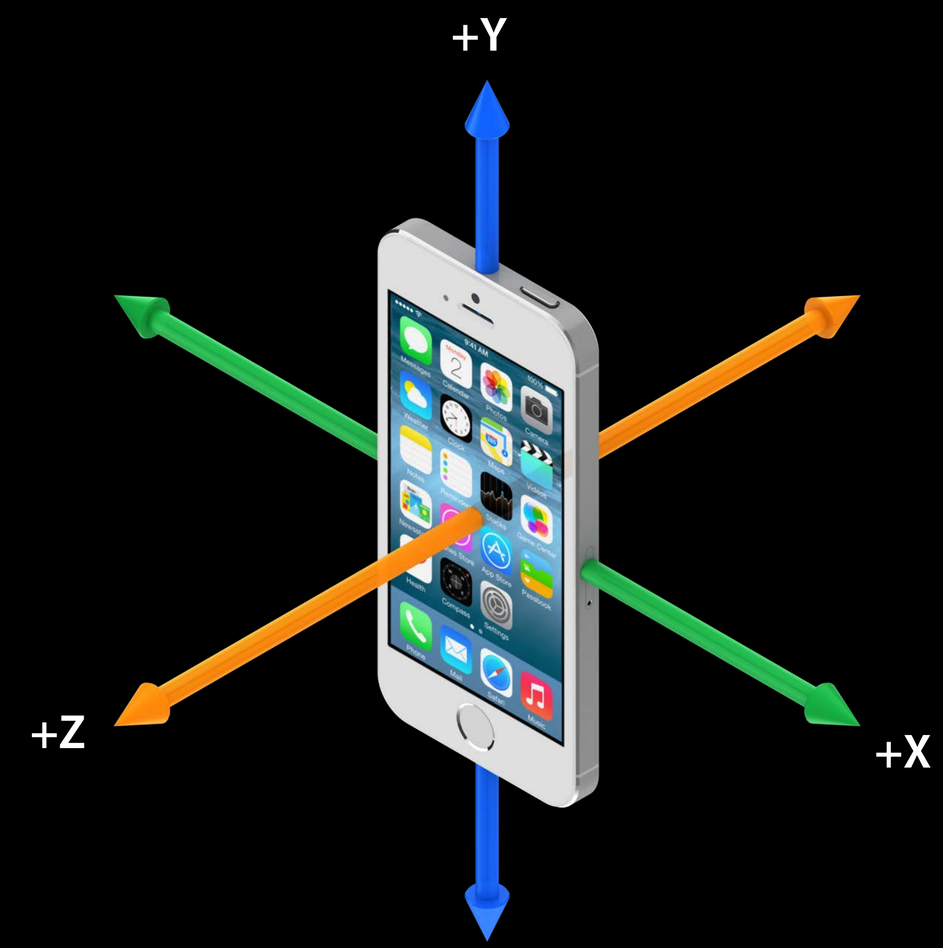
Motion Forwarding

GCQuaternion attitude

3D orientation of device

- Yaw
- Pitch
- Roll

```
typedef struct {
    double x, y, z, w;
} GCQuaternion;
```



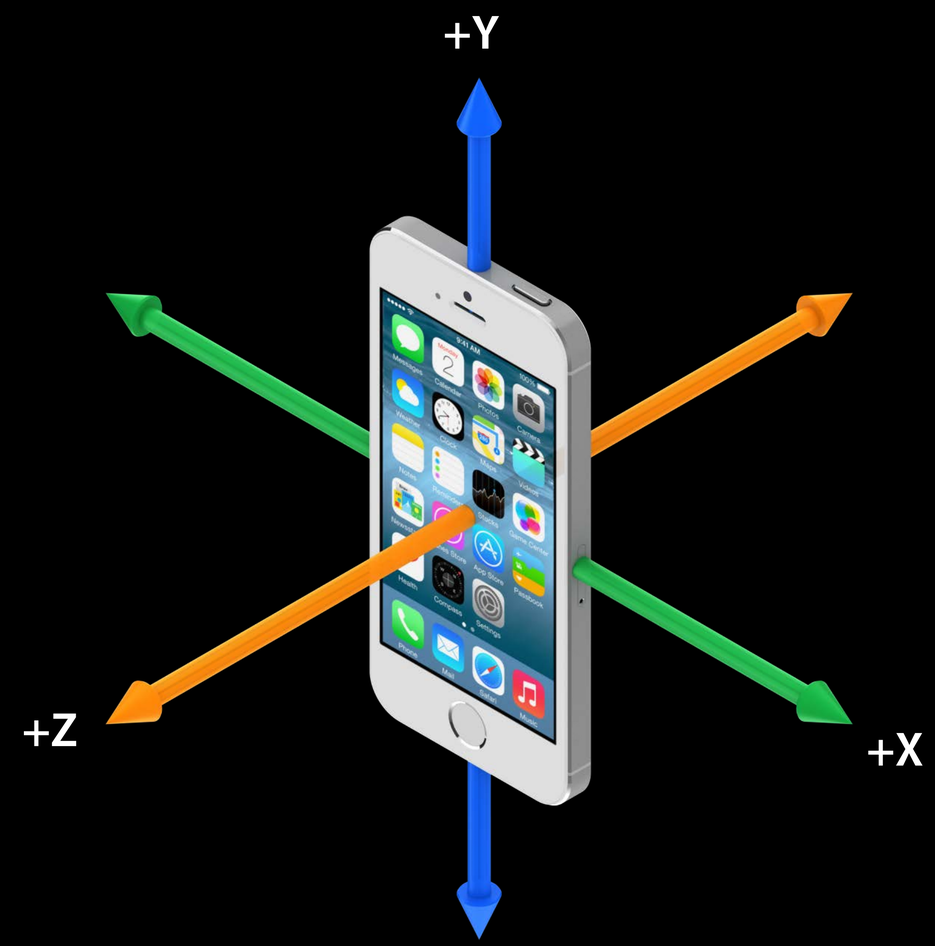
Motion Forwarding

GCRotationRate rotationRate

Rate at which device is spinning

Units are in radians/second

```
typedef struct {
    double x, y, z;
} GCRotationRate;
```



Motion Forwarding

```
GCMotion *motionProfile = self.myController.motion;
if (motionProfile) {
    // use controller’s motion profile
}
else {
    // use motion from device
}
```

Motion Forwarding

```
GCMotion *motionProfile = self.myController.motion;
if (motionProfile) {
    // use controller’s motion profile
}
else {
    // use motion from device
}
```

Motion Forwarding

```
GCMotion *motionProfile = self.myController.motion;
if (motionProfile) {
    // use controller’s motion profile
}
else {
    // use motion from device
}
```

Motion Forwarding

```
GCMotion *motionProfile = self.myController.motion;
if (motionProfile) {
    // use controller’s motion profile
}
else {
    // use motion from device
}
```

Motion Forwarding

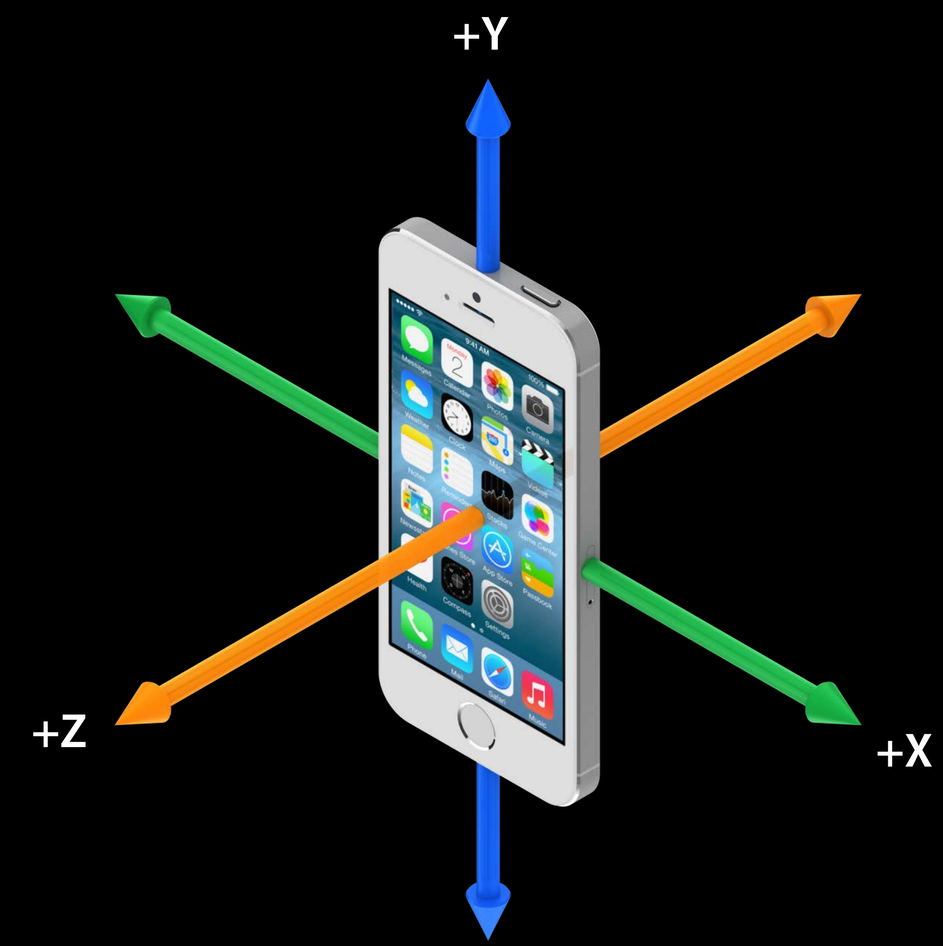
```
GCMotion *motionProfile = self.myController.motion;
if (motionProfile) {
    // use controller’s motion profile
}
else {
    // use motion from device
}
```

Motion Forwarding

Motion axes move with device

Motion data will be jittery

- Apply filter to smooth data

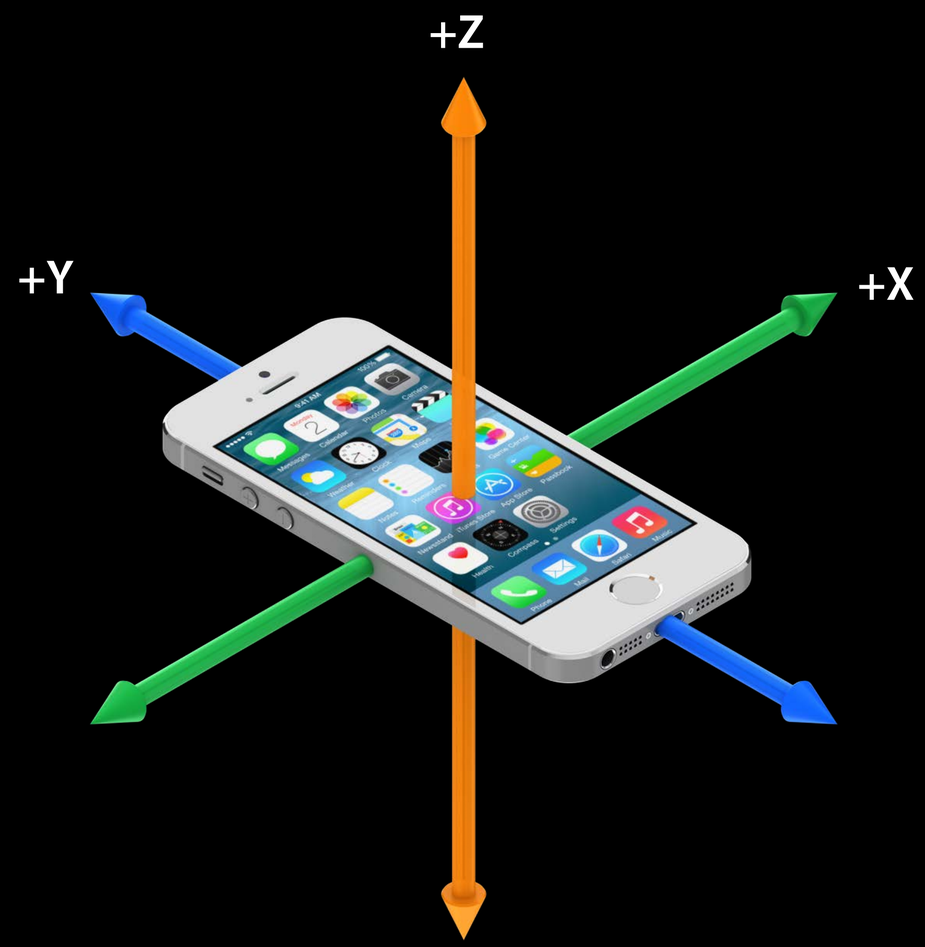


Motion Forwarding

Motion axes move with device

Motion data will be jittery

- Apply filter to smooth data



Idle Timer

Prevents screen from turning off

iOS 8—Handled automatically

iOS 7—Handle yourself

```
[[UIApplication sharedApplication] setIdleTimerDisabled:(BOOL)];
```

- Be careful

Motion apps should also manage their idle timer

Additional Guidance

Pause Button

Every controller has pause button

Treat as toggle

Handling required if support game controllers

```
myController.controllerPausedHandler =
^(GCController *controller)  {
    // Pause button pressed
    [self togglePauseResumeState];
};
```

Pause Button

Every controller has pause button

Treat as toggle

Handling required if support game controllers

```
myController.controllerPausedHandler =
^(GCController *controller)  {
    // Pause button pressed
    [self togglePauseResumeState];
};
```

Player Indicator LEDs

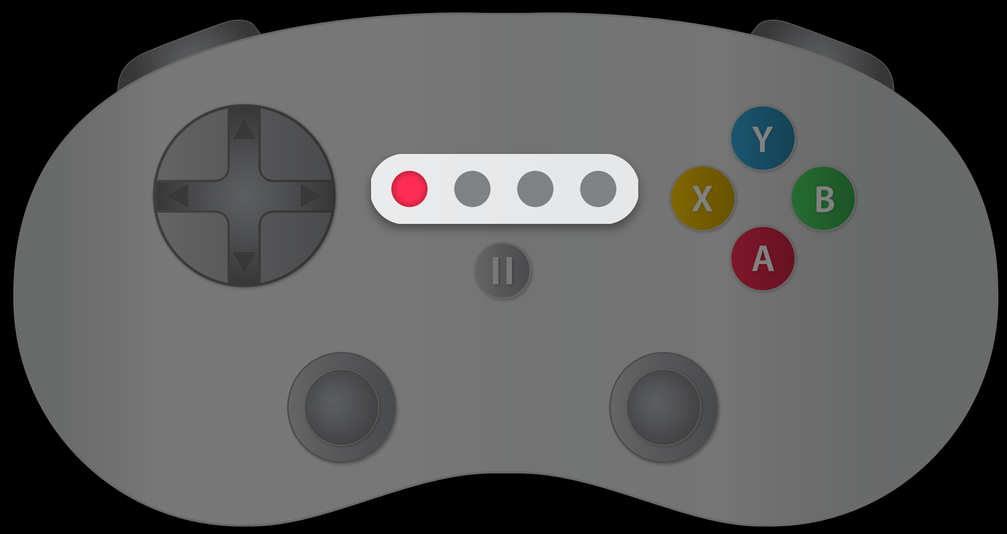
Always set for each controller your game uses

```
if (self.myController.playerIndex == GCControllerPlayerIndexUnset)
{
    self.myController.playerIndex = 0;  // Zero-based index
}
```

Player Indicator LEDs

Always set for each controller your game uses

```
if (self.myController.playerIndex == GCControllerPlayerIndexUnset)
{
    self.myController.playerIndex = 0;  // Zero-based index
}
```



Design Guidance

Multiple-controller games

Keep track of array of controllers

- Set playerIndex for all controllers used

Consider which controller(s) can navigate menus

Determine whether game can proceed after a controller disconnects

Design Guidance

Multiple-controller games

```
- (void)setupControllers:(NSNotification *)notification
{
    ...
    for (GCController *c in self.myControllers)
    {
        if (c.playerIndex == GCControllerPlayerIndexUnset)
        {
            [self launchControllerPicker];
            break;
        }
    }
    ...
}
```

Design Guidance

Multiple-controller games

```
- (void)setupControllers:(NSNotification *)notification
{
    ...
    for (GCController *c in self.myControllers)
    {
        if (c.playerIndex == GCControllerPlayerIndexUnset)
        {
            [self launchControllerPicker];
            break;
        }
    }
    ...
}
```

Design Guidance

Multiple-controller games

```
- (void)setupControllers:(NSNotification *)notification
{
    ...
    for (GCController *c in self.myControllers)
    {
        if (c.playerIndex == GCControllerPlayerIndexUnset)
        {
            [self launchControllerPicker];
            break;
        }
    }
    ...
}
```

Design Guidance

Multiple-controller games

```
- (void)setupControllers:(NSNotification *)notification
{
    ...
    for (GCController *c in self.myControllers)
    {
        if (c.playerIndex == GCControllerPlayerIndexUnset)
        {
            [self launchControllerPicker];
            break;
        }
    }
    ...
}
```

Design Guidance

Respond to game controller inputs early

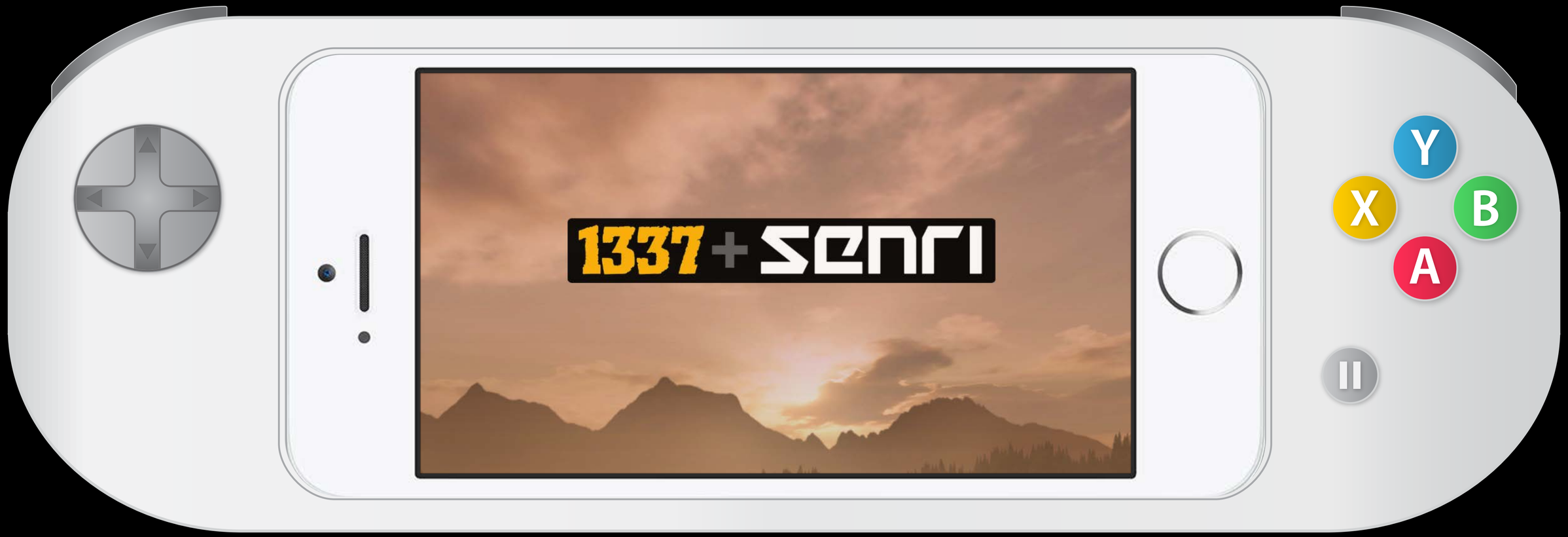
Players instinctively press buttons upon game launch

Responding early tells players your game supports controllers

- Splash screens
- Introductory cinematics
- Main menu—at the latest

Design Guidance

Respond to game controller inputs early



Design Guidance

Respond to game controller inputs early

Design Guidance

Respond to game controller inputs early

Design Guidance

Mechanize UI for controllers

Support standalone controllers

- Screen (touch) may not be easily reachable
- Game input from controller only

Summary

Overview

Finding controllers

Reading controller input

What's new

Design guidance

More Information

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

Filip Iliescu
Graphics and Game Technologies Evangelist
filiescu@apple.com

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

Related Sessions

• What's New in SpriteKit	Pacific Heights	Wednesday 2:00PM
• Best Practices for Building SpriteKit Games	Pacific Heights	Wednesday 3:15PM
• What's New in SceneKit	Pacific Heights	Thursday 10:15AM
• Building a Game with SceneKit	Pacific Heights	Thursday 11:30AM

Labs

• Game Controllers Lab	Graphics and Games Lab B	Friday 11:30AM

WWDC14