Games!

What to expect

- No guide for games on Apple Watch
- Gaming on Apple Watch is different
- but programming is not

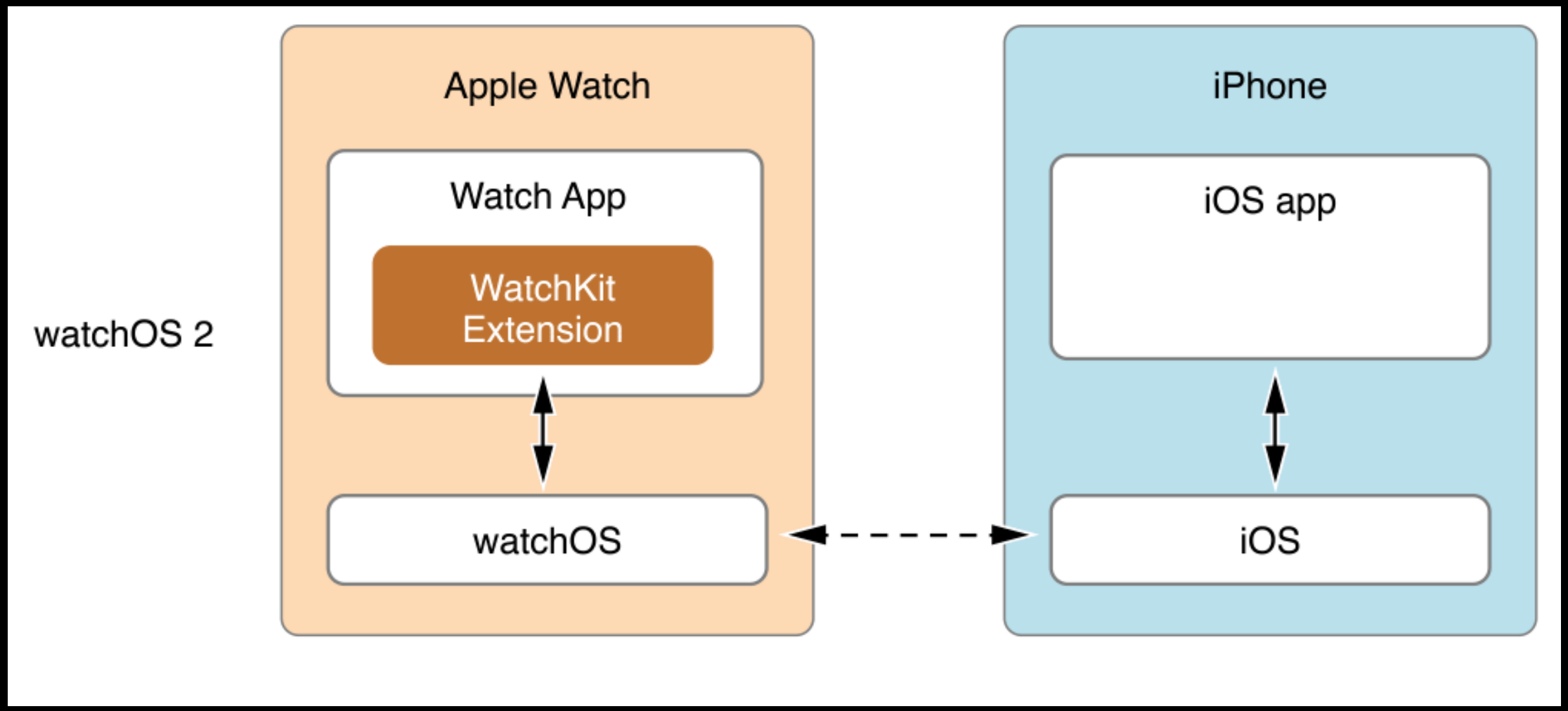
Agenda

- Why?
- Architecture
- Interface
- Frameworks
- An Example: Alien Ace
  - Storyboard
  - Game loop
  - Performance
  - Drawing

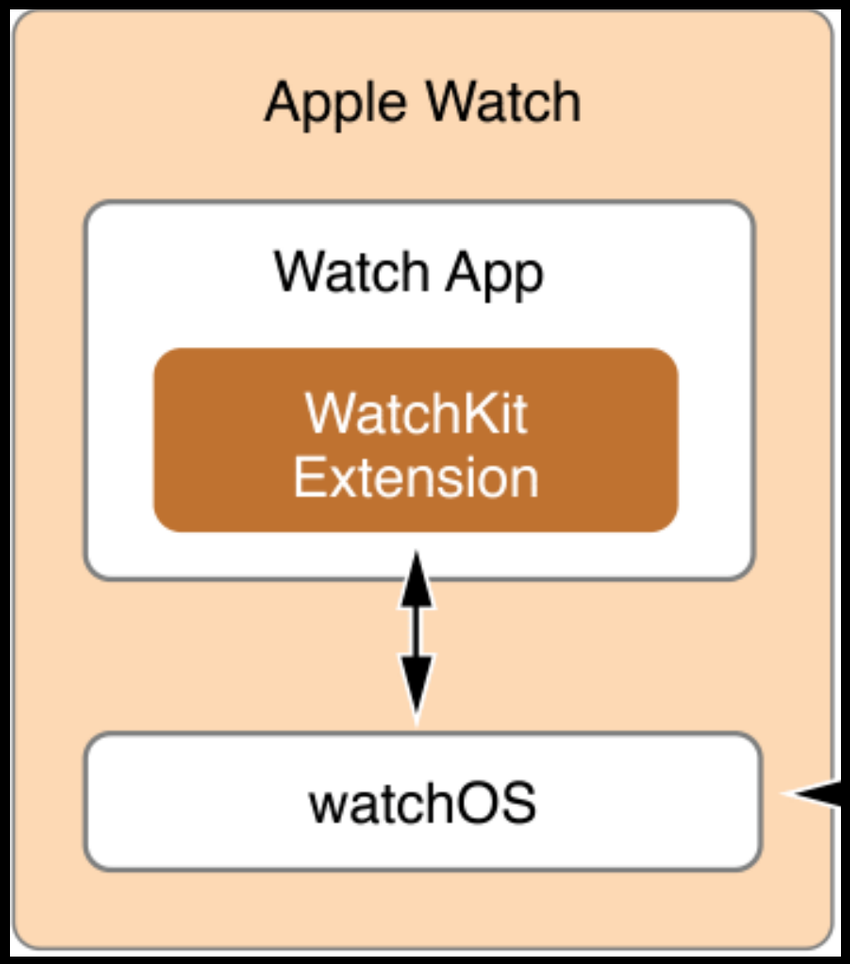
Why gaming on Apple Watch

- gaming is fun
- some years ago people asked „Why games on your phone?“
- small market, but less competitors
- people don’t know what to do with their Apple Watch :p
- add a small game to your existing iOS Game
  - daily quests
  - bonus points

Architecture



Architecture



- WatchKit Extension runs on device
- App and Extension are different targets running parallel
- they share resources and data needs to be *transferred*

Interface

- No Sprite Kit, no magic, just WKInterface
- let’s talk about some interesting classes:
  - WKInterfaceGroup
  - WKInterfaceButton
  - WKInterfaceLabel
  - WKInterfacePicker
  - WKInterfaceDevice

WKInterfaceGroup

- can embed other interface items
- nested groups
- can have a background image :)
  - func setBackgroundImage(_ *image*: UIImage?)
  - transparency
- can have any size, but no size getters

WKInterfaceButton

- button with target-action
  - no sender
- text or group as content
  - you *can't* remove the animation if you use groups
  - empty string is perfectly fine
- Color can be changed, e.g. UIColor.clearColor()

WKInterfaceLabel

- Can be used to render text
- but strings need to be transferred from extension to app -> maybe too slow

WKInterfacePicker

- Only way to access Digital Crown
- minimum size 2pt x 2pt
- can be placed outside of view
- can be initialized with „empty“ WKPickerItems, target-action

WKInterfaceDevice

- You can cache images up to 5MB in total
- Play haptics, there are different predefined haptics you can run

Additional frameworks

- Core Graphics
- Core Motion
- UIKit
- GameCenter (with a little help of your companion app)

Core Motion

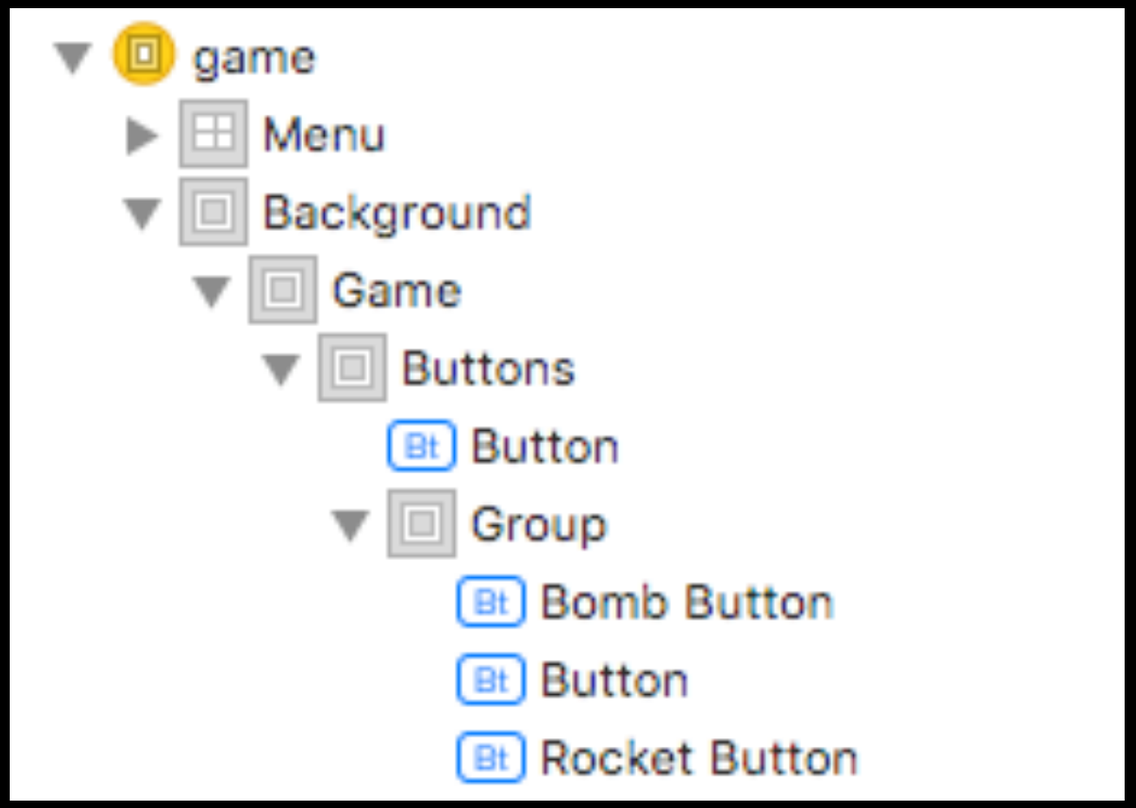
- full access to accelerometer data
- should be calibrated, because people hold their watch differently
- My Case: I use only y value, save beginning value and use *f(x)=4*(rawY - neutralY)* to get values between -1 and 1

That’s all you got


8:07
ALIEN
ACE

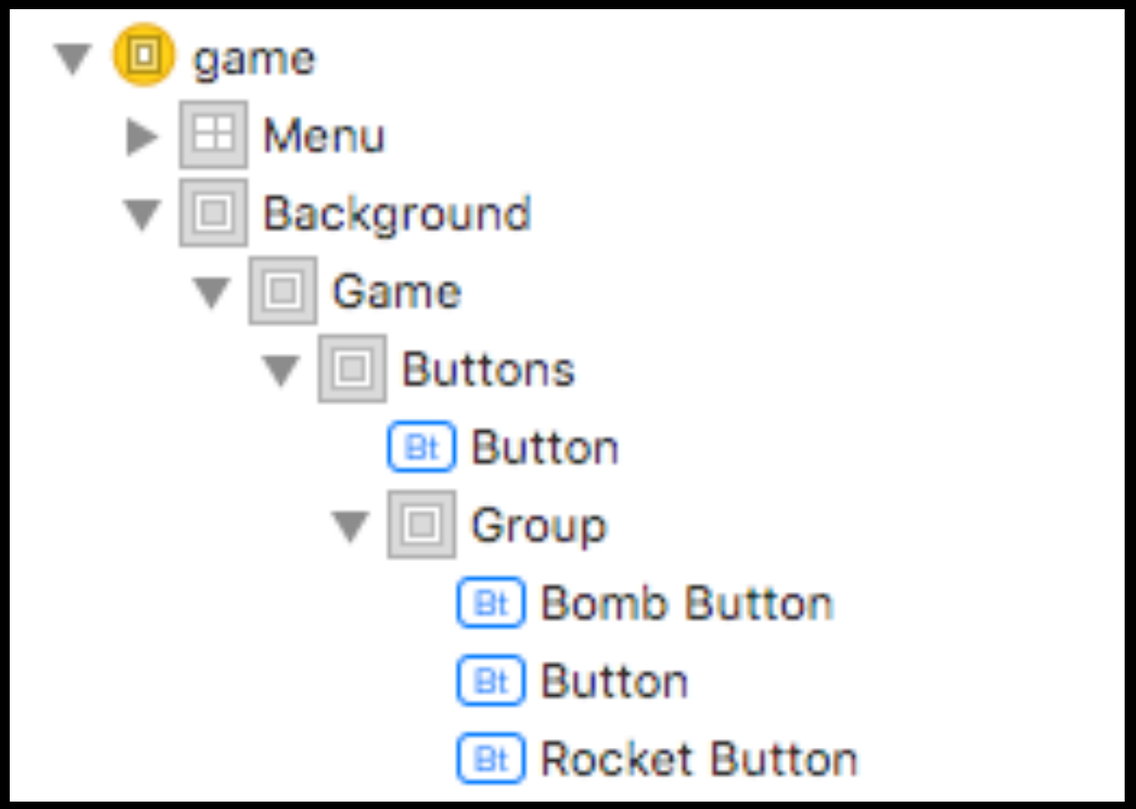
Storyboard

- a few nested groups and buttons
- Background group holds background image
- Game Group displays the transparent game as background Image
- transparent buttons are nested above



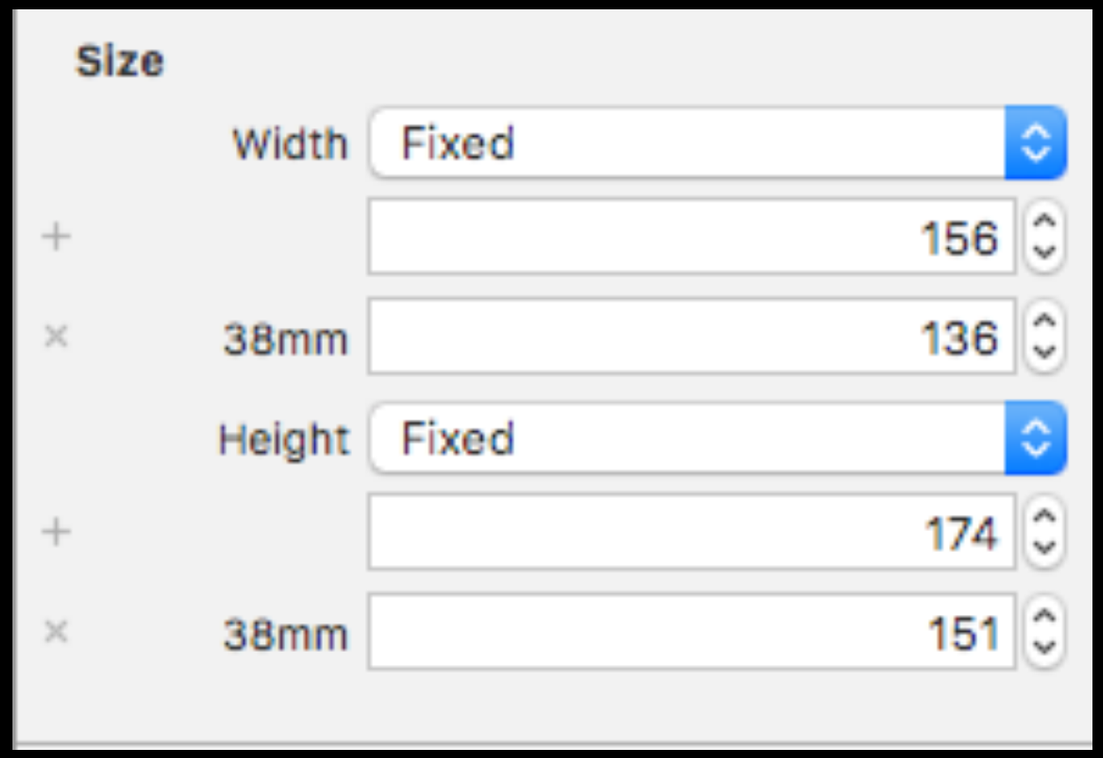
Storyboard

- note: All buttons are transparent
- there aren't any Labels
- try to reduce the calls needed to talk to the app
- multiple transparent layers need rendering time



Storyboard

- Size of game group is fixed, because there is no getter
- make sure to use same size in game to prevent resizing



The game loop

- game loop is controlled by NSTimer
- the app displays your game, you don’t know how fast, so you must chose your fps wisely
- on a smaller screen might lower fps suffice, I use 6 fps, up to about 10 might be possible
- If you set your fps higher you get lags

My game loop

```
public func updatePhysics(lastFrame: CFAbsoluteTime)

public func collisionDetection()

public func draw(lastFrame: CFAbsoluteTime)
```

- very simple:
  - updatePhysics() updates all Objects and runs all actions
  - collisionDetection() hitTests all objects, is surprisingly fast
  - draw() renders the scene in an UIImage, takes the majority of (extension) time

Memory

- memory on Apple Watch is limited
- Apps often get terminated at about 30MB
- share common data, use lightweight objects
- e.g. share sprites between objects and only save the internal state of the object
- Alien Ace runs with 2.5MB memory consumption

Some performance data

(Simulator data, as there are no symbols in Instruments using Device)

504.0ms	36.5%	0,0	▼Main Thread 0x93a3de
424.0ms	30.7%	0,0	▼start libdyld.dylib
424.0ms	30.7%	0,0	▼main WatchKit
424.0ms	30.7%	0,0	▼-[PKService run] PlugInKit
423.0ms	30.6%	0,0	▼-[NSXPCListener resume] Foundation
423.0ms	30.6%	0,0	▼xpc_main libxpc.dylib
423.0ms	30.6%	0,0	▼_xpc_objc_main libxpc.dylib
423.0ms	30.6%	0,0	▼-[NSRunLoop(NSRunLoop) run] Foundation
423.0ms	30.6%	0,0	▼-[NSRunLoop(NSRunLoop) runMode:beforeDate:] Foundation
416.0ms	30.1%	0,0	▼CFRunLoopRunInMode CoreFoundation
416.0ms	30.1%	0,0	▼CFRunLoopRunSpecific CoreFoundation
357.0ms	25.9%	0,0	▼__CFRunLoopRun CoreFoundation
324.0ms	23.5%	0,0	▶__CFRUNLOOP_IS_SERVICING_THE_MAIN_DISPATCH_QUEUE__ CoreFoundation
17.0ms	1.2%	0,0	▼__CFRunLoopDoTimer CoreFoundation
16.0ms	1.1%	0,0	▼__CFRUNLOOP_IS_CALLING_OUT_TO_A_TIMER_CALLBACK_FUNCTION__ CoreFoundation
16.0ms	1.1%	0,0	▼__NSFireTimer Foundation
15.0ms	1.0%	0,0	▼@objc Game.run() -> () GameKit
15.0ms	1.0%	0,0	▼Game.run() -> () GameKit
13.0ms	0.9%	0,0	▶AlienAce.updatePhysics(Double) -> () Space Impact WatchKit Extension
1.0ms	0.0%	0,0	▶AlienAce.collisionDetection() -> () Space Impact WatchKit Extension
1.0ms	0.0%	0,0	▼AlienAce.draw(Double) -> () Space Impact WatchKit Extension
1.0ms	0.0%	0,0	▶dispatch_async libdispatch.dylib
1.0ms	0.0%	0,0	▶_CFAutoreleasePoolPop CoreFoundation
1.0ms	0.0%	0,0	▶__CFRepositionTimerInMode CoreFoundation
8.0ms	0.5%	0,0	▶_CFAutoreleasePoolPop CoreFoundation
2.0ms	0.1%	1,0	▶CFAbsoluteTimeGetCurrent CoreFoundation
1.0ms	0.0%	1,0	voucher_mach_msg_revert libsystem_kernel.dylib
1.0ms	0.0%	1,0	dispatch_get_main_queue_port_4CF libdispatch.dylib

Some performance data

(Simulator data, as there are no symbols in Instruments using Device)

424.0ms	30.7%	0,0	start libdyld.dylib
424.0ms	30.7%	0,0	main WatchKit
424.0ms	30.7%	0,0	-[PKService run] PlugInKit
423.0ms	30.6%	0,0	-[NSXPCListener resume] Foundation
423.0ms	30.6%	0,0	xpc_main libxpc.dylib
423.0ms	30.6%	0,0	_xpc_objc_main libxpc.dylib
423.0ms	30.6%	0,0	-[NSRunLoop(NSRunLoop) run] Foundation
423.0ms	30.6%	0,0	-[NSRunLoop(NSRunLoop) runMode:beforeDate:] Foundation
416.0ms	30.1%	0,0	CFRunLoopRunInMode CoreFoundation
416.0ms	30.1%	0,0	CFRunLoopRunSpecific CoreFoundation
357.0ms	25.9%	0,0	__CFRunLoopRun CoreFoundation
324.0ms	23.5%	0,0	__CFRUNLOOP_IS_SERVICING_THE_MAIN_DISPATCH_QUEUE__ CoreFoundation
324.0ms	23.5%	0,0	_dispatch_main_queue_callback_4CF libdispatch.dylib
312.0ms	22.6%	0,0	_dispatch_client_callout libdispatch.dylib
308.0ms	22.3%	0,0	_dispatch_call_block_and_release libdispatch.dylib
268.0ms	19.4%	1,0	__54+[SPRemoteInterface setController:key:property:value:]_block_invoke WatchKit
266.0ms	19.3%	0,0	SerializablePropertyValue WatchKit
242.0ms	17.5%	0,0	-[NSKeyedArchiver encodeObject:forKey:] Foundation
242.0ms	17.5%	0,0	_encodeObject Foundation
238.0ms	17.2%	0,0	-[UIImage encodeWithCoder:] UIKit
231.0ms	16.7%	0,0	-[UIImage _encodeDataWithCoder:imageName:] UIKit
222.0ms	16.1%	0,0	UIImagePNGRepresentation UIKit
9.0ms	0.6%	0,0	-[NSKeyedArchiver encodeObject:forKey:] Foundation
6.0ms	0.4%	0,0	-[NSKeyedArchiver encodeObject:forKey:] Foundation
1.0ms	0.0%	0,0	-[UIImage _encodePropertiesWithCoder:] UIKit
1.0ms	0.0%	0,0	__NSKeyedArchiverUIDCreateCached Foundation
1.0ms	0.0%	0,0	-[SPRemoteInterfaceKeyedArchiverDelegate archiver:willEncodeObject:] WatchKit
1.0ms	0.0%	0,0	addValueToTopContainerE Foundation

Some performance data

(Simulator data, as there are no symbols in Instruments using Device)

0ms	22.6%	0,0	▼_dispatch_client_callout libdispatch.dylib
0ms	22.3%	0,0	▼_dispatch_call_block_and_release libdispatch.dylib
0ms	19.4%	1,0	▼__54+[SPRemoteInterface setController:key:property:value:]_block_invoke
0ms	19.3%	0,0	▼SerializablePropertyValue WatchKit
0ms	17.5%	0,0	▼-[NSKeyedArchiver encodeObject:forKey:] Foundation
0ms	17.5%	0,0	▼_encodeObject Foundation
0ms	17.2%	0,0	▼-[UIImage encodeWithCoder:] UIKit
0ms	16.7%	0,0	▼-[UIImage _encodeDataWithCoder:imageName:] UIKit
0ms	16.1%	0,0	▶UIImagePNGRepresentation UIKit
0ms	0.6%	0,0	▶-[NSKeyedArchiver encodeObject:forKey:] Foundation
0ms	0.4%	0,0	▶-[NSKeyedArchiver encodeObject:forKey:] Foundation
0ms	0.0%	0,0	▶-[UIImage _encodePropertiesWithCoder:] UIKit
0ms	0.0%	0,0	▶ NSKeyedArchiverUIDCreateCached Foundation

- image needs to be encoded as PNG to be transferred to the app
- the time to encode correlates to the number of (non-transparent) pixels

Some performance data

(Simulator data, as there are no symbols in Instruments using Device)

24.3%	1,0	AlienAce.(draw(AlienAce) -> (Double) -> ()).(closur
7.9%	0,0	GameObject.drawAtPosition() -> () GameKit
6.2%	0,0	AlienAce.drawEnemies() -> () Space Impact Wat
4.7%	0,0	AlienAce.drawPowerups() -> () Space Impact Wa
2.3%	0,0	AlienAce.drawButtons() -> () Space Impact Watc
1.1%	0,0	AlienAce.drawBullets() -> () Space Impact Watch
0.4%	0,0	LifeView.draw(CGRect, con : CGContext) -> () S
0.3%	0,0	UIGraphicsBeginImageContextWithOptions UIKi
0.3%	0,0	UIGraphicsGetImageFromCurrentImageContext
0.3%	0,0	Label.drawAtPosition() -> () GameKit
0.2%	0,0	SetGenerator.next<A where ...>() -> A? libswiftC
0.1%	0,0	objc_release libobjc.A.dylib
1.3%	0,0	_dispatch_apply_invoke libdispatch.dylib
0.0%	0,0	_dispatch_queue_override_invoke_owning libdispatch

- drawing takes a lot of time
- drawing time correlates to the number of pixels drawn, this is a serious problem
- trying to draw an entire screen takes way too much time

Too much pixels to draw fluent!

Solution?

Drawing non-retina

- non-retina drawing is much faster than retina drawing
- drawing is faster, encoding and decoding is faster
- set scale factor in UIGraphicsBeginImageContextWithOptions()
- Retina Graphics drawn as non-retina look ugly -> create optimized versions

Loading non-retina graphics

- There’s no way to load non-retina from your asset-catalogue
- I still use asset catalogue by setting the 1x Graphic as 2x Graphic
- To render correctly you need to change the scale factor to 1x, but it’s readonly
- place this in an extension:

```
return UIImage(data: UIImagePNGRepresentation(self)!, scale: CGFloat(scale))!
```

Labels

- drawing text takes a significant amount of time
- you can exchange CPU time for memory
- often game labels do only need numbers
- you can build a custom label with pre-rendered 0-9
- save them as UIImage, rendering will be much faster

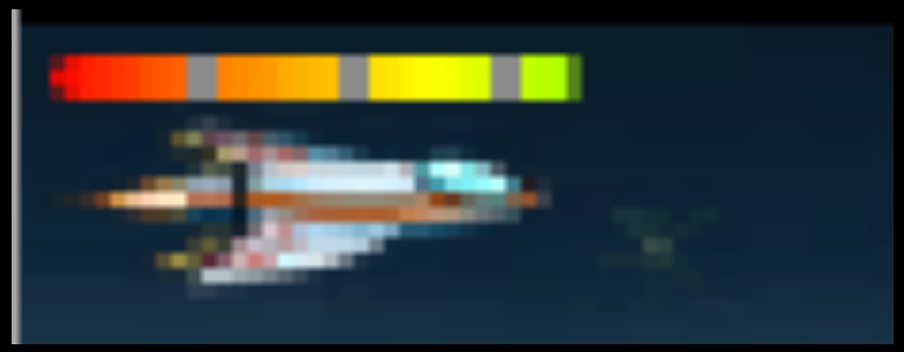
Labels

```
for x in 0 ... 9 {
    let string: NSString = "\(x)"
    let size = string.sizeWithAttributes(attr)
    UIGraphicsBeginImageContextWithOptions(size, false,
        CGFloat(GameSettings.scaleFactor))
    string.drawAtPoint(CGPointMake(0, 0), withAttributes: attr)
    self.digits.append(UIGraphicsGetImageFromCurrentImageContext())
    UIGraphicsEndImageContext()
}
```

Drawing

- Core Graphics
  - CGPath
- UIKit drawing methods
  - UIImage.drawAtPoint()
  - UIBezierPath
- watch out for expensive function calls

Drawing



- Alien Ace life gradient, implemented using Core Graphics
- created in background at beginning, saved as UIImage
- clipped to current life with CGContextClipToRect(…)

Multithreading

- GCD like on iOS
- create stuff in QOS_CLASS_BACKGROUND
- QOS_CLASS_USER_INITIATED for interactive drawing
  - INTERACTIVE might block your app

That’s it

- Questions and Discussion
- feel free to add me on Xing, Facebook and Twitter