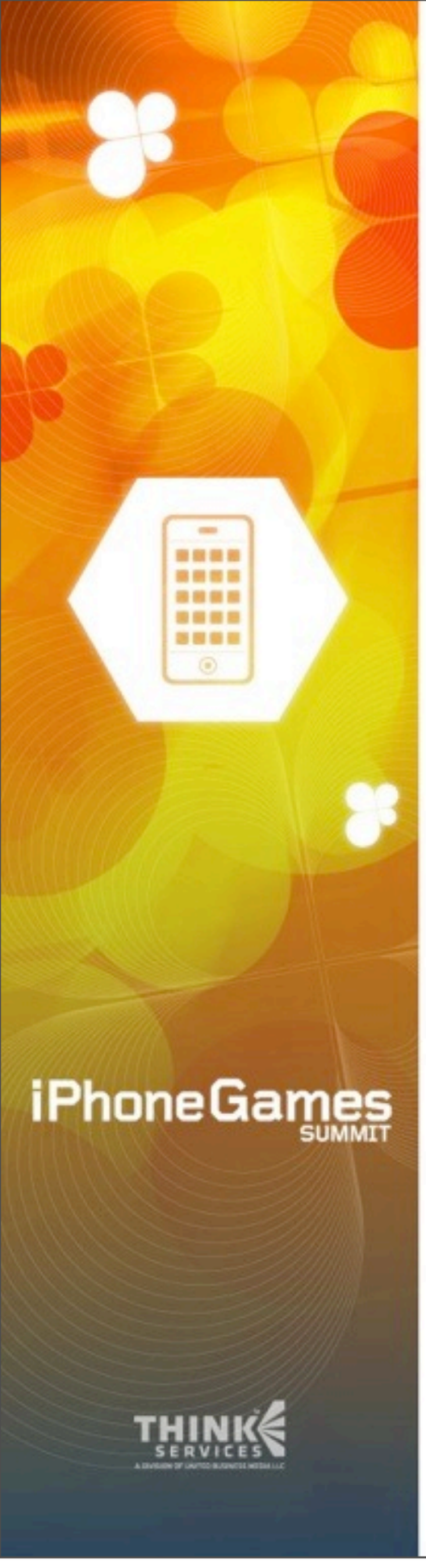


Squeezing Every Drop Of Performance Out Of The iPhone

Noel Llopis
Snappy Touch

http://twitter.com/snappytouch
noel@snappytouch.com
http://gamesfromwithin.com

In the last 11 years I've made games for almost every major platform out there

In the last 11 years I've made games for almost every major platform out there

During that time, working on engine technology and trying to achieve maximum performance
Performance always very important

snappy touch

snappy touch

Almost a year ago I started Snappy Touch

snappy
touch

snappy
touch

Almost a year ago I started Snappy Touch

When you’re on the engine team of a AAA game, you can spend a lot of time on performance.
On the iPhone, I didn’t think I had the time
In a few days I had the prototype ported over with one small problem

Running at 40 FPS
on first attempt...

When you’re on the engine team of a AAA game, you can spend a lot of time on performance.
On the iPhone, I didn’t think I had the time
In a few days I had the prototype ported over with one small problem

Running at 40 FPS
on first attempt...

... on the simulator!

When you're on the engine team of a AAA game, you can spend a lot of time on performance.
On the iPhone, I didn't think I had the time
In a few days I had the prototype ported over with one small problem

Running at 40 FPS
on first attempt...

... on the simulator!

4 FPS!

snappy touch

When you're on the engine team of a AAA game, you can spend a lot of time on performance.
On the iPhone, I didn't think I had the time
In a few days I had the prototype ported over with one small problem

Lessons learned getting frame rate from 4 fps to ~30 fps

- Performance analysis tools

- Performance analysis tools
- CPU

- Performance analysis tools
- CPU
- Game loop architecture

- Performance analysis tools
- CPU
- Game loop architecture
- Rendering

snappy touch

- Performance analysis tools
- CPU
- Game loop architecture
- Rendering
- Memory

Performance Analysis Tools

Exploratory

snappy touch

Exploratory

- Turning features on and off in real time

snappy touch

Exploratory

- Turning features on and off in real time
- Works very well to get very rough estimates

snappy touch

Exploratory

- Turning features on and off in real time
- Works very well to get very rough estimates
- Simulation vs. geometry creation vs. rendering

snappy touch

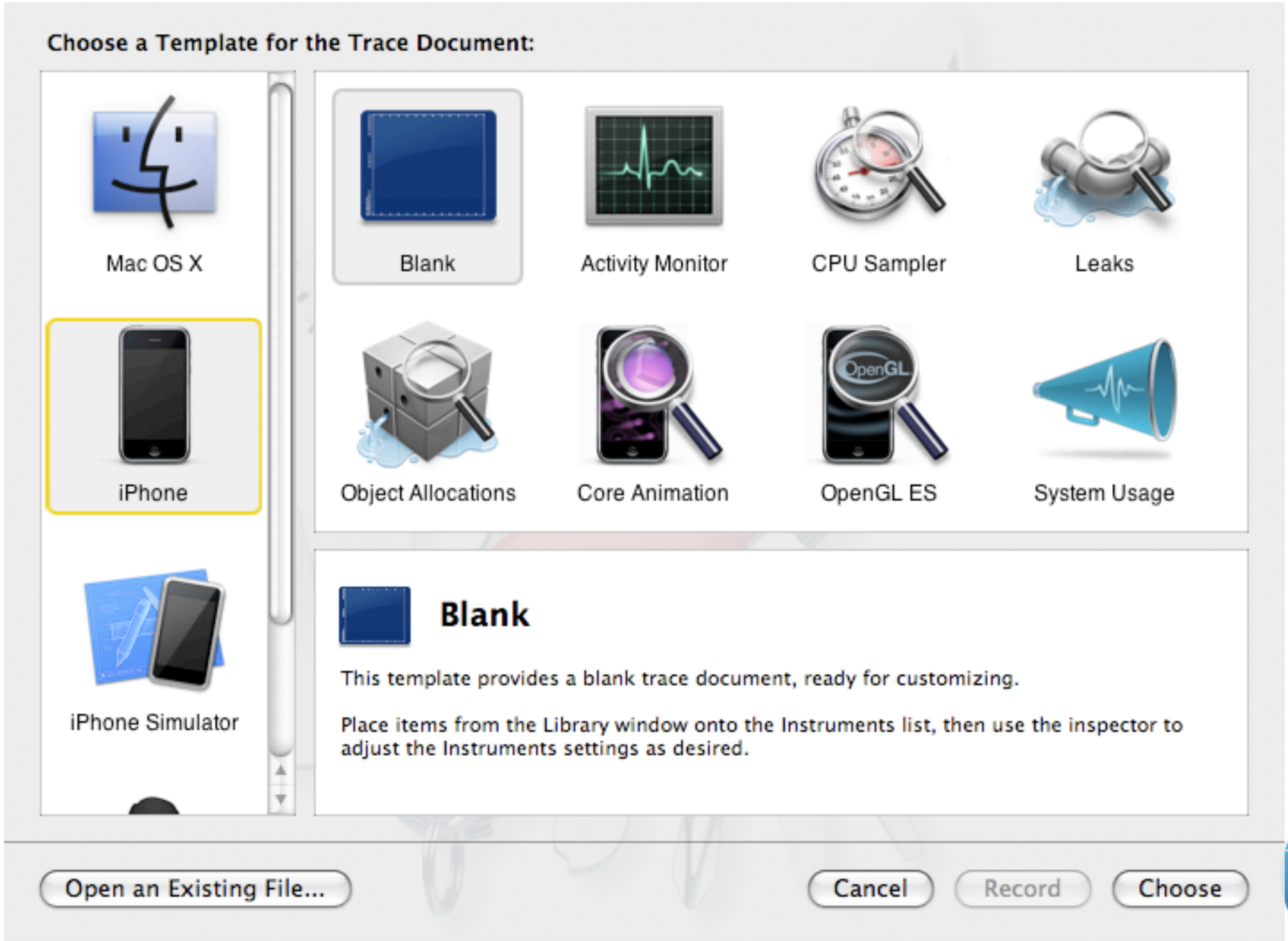
Instruments

snappy touch

Instruments

snappy touch

Instruments



snappy touch

Instruments

snappy touch

Instruments

- Versatile

snappy touch

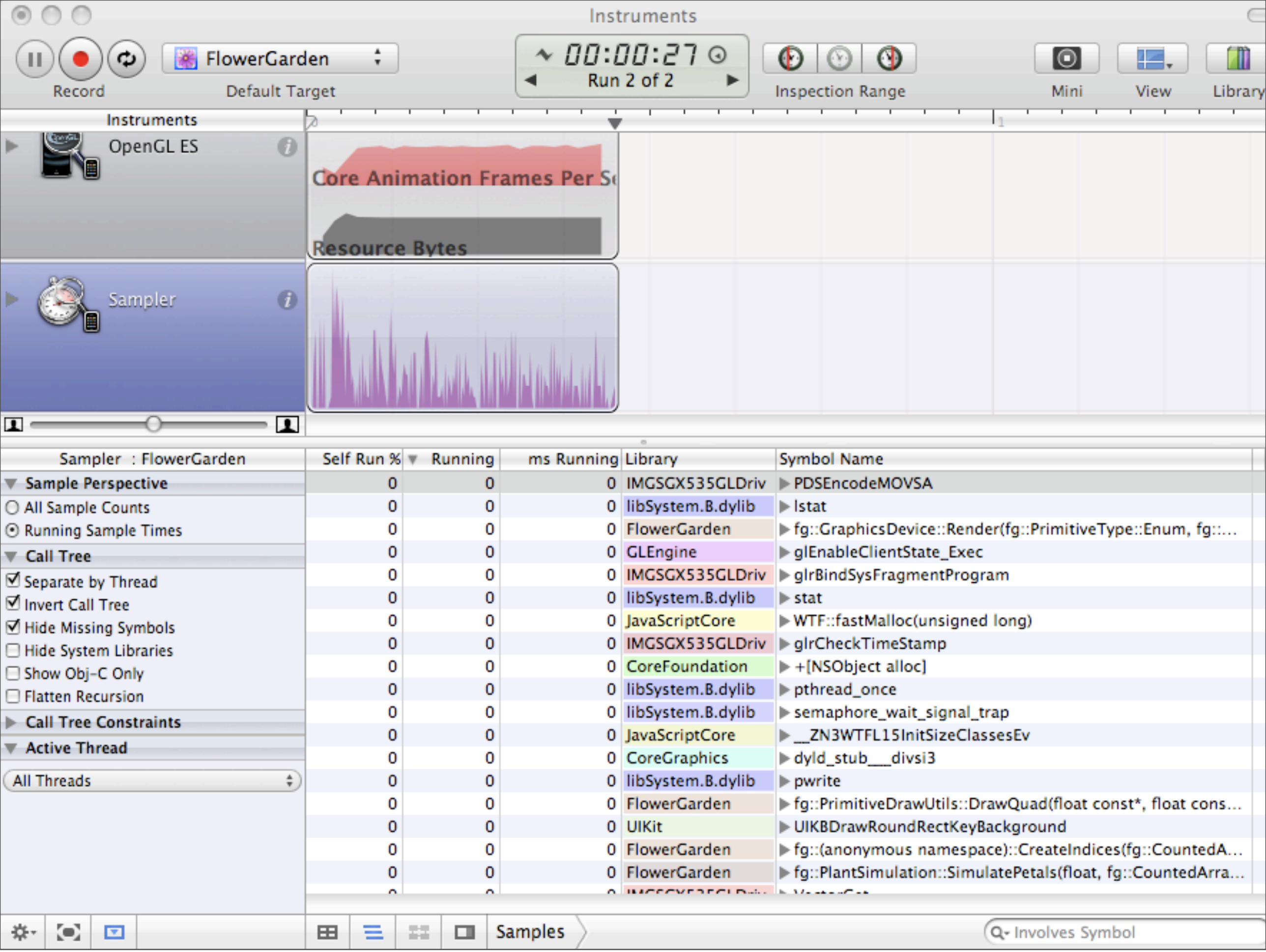
Instruments

- Versatile
- Can display frame rate, object allocations, memory usage, or even OpenGL calls.

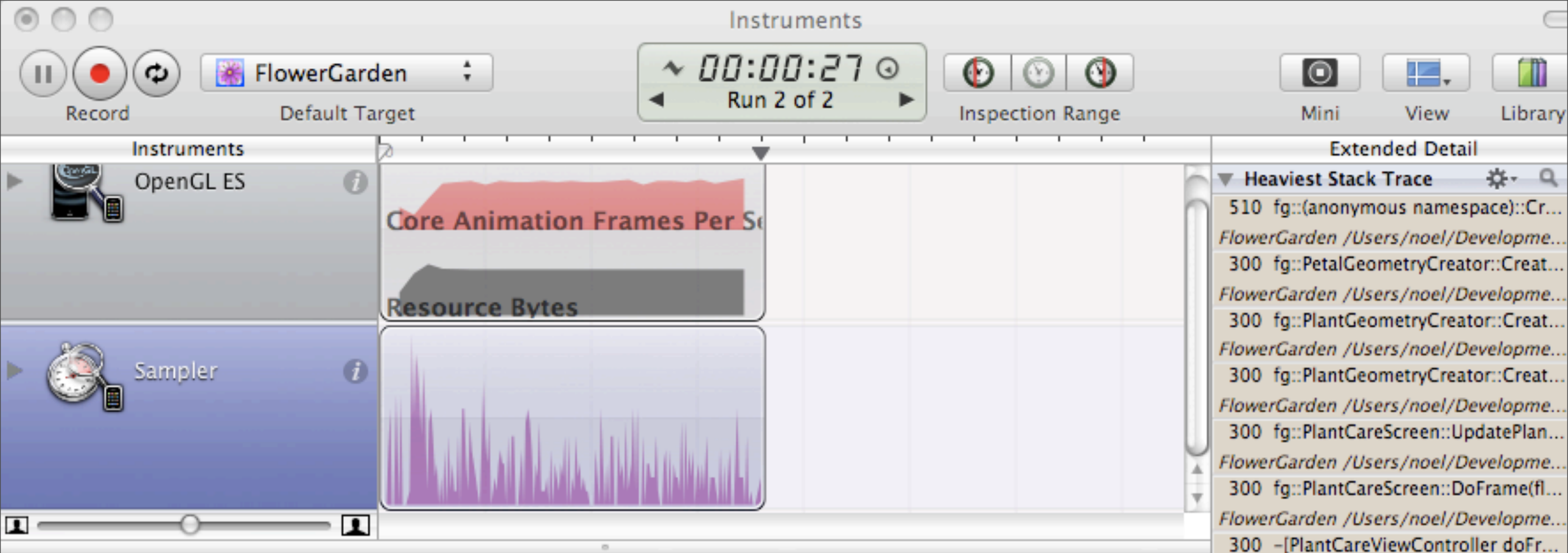
Instruments

- Versatile
- Can display frame rate, object allocations, memory usage, or even OpenGL calls.
- Rut it on the real device, not the simulator!

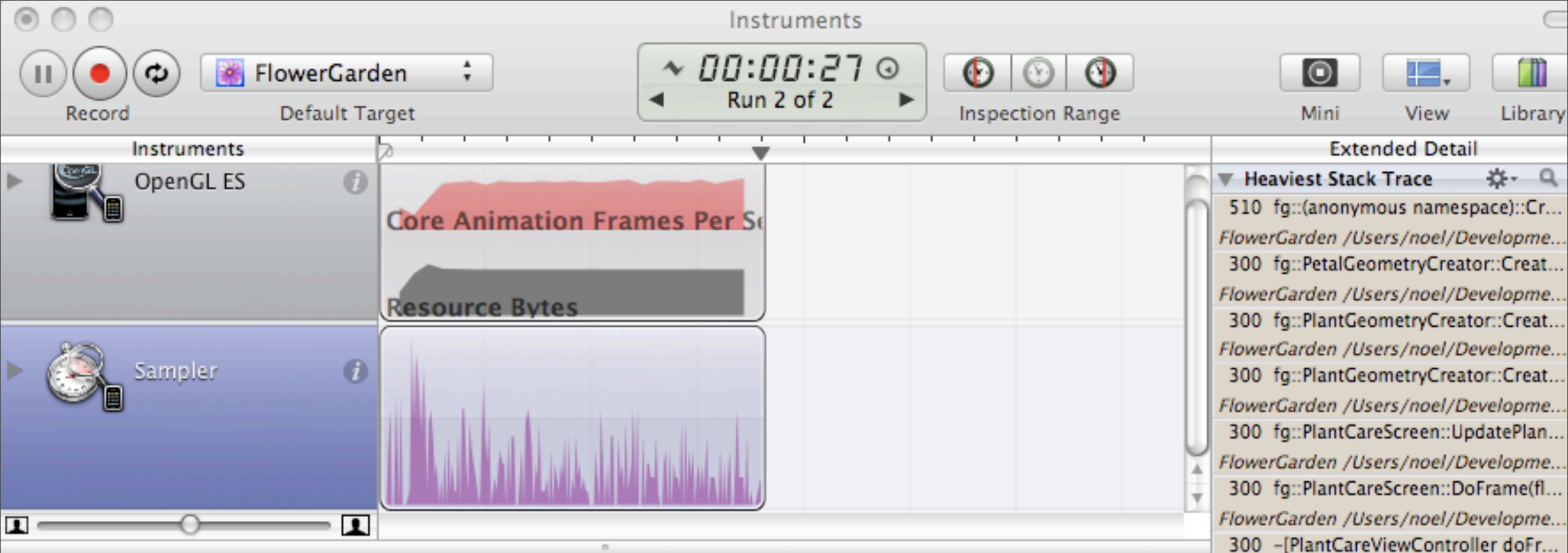
snappy touch



Best way to start.
Frame rate over time plus CPU sampler



Digging deeper



Digging deeper

Instruments

Record | Default Target: FlowerGarden | 00:00:27, Run 1 of 1 | Inspection Range | Mini | View

Instruments: ObjectAlloc, Leaks

ObjectAlloc	Graph	Category	Net Bytes	# Net	Overall Bytes	# Overall	# Allocations (Net / Overall)
Allocation Lifespan	☑	* All Allocations *	5050088	14608	17093052	82245	+++
All Objects Created	☐	GeneralBlock-32	75488	2359	702080	21970	
Created & Still Living	☐	GeneralBlock-48	26304	547	747120	15587	
Call Tree	☐	GeneralBlock-16	61616	3850	108464	6907	
Separate by Thread	☐	CFString	75168	2108	236880	6858	
Invert Call Tree	☐	GeneralBlock-80	13136	164	227424	2839	
Hide Missing Symbols	☐	GeneralBlock-416	4992	12	1131520	2720	
Hide System Libraries	☐	GeneralBlock-208	4992	24	511056	2457	
Show Obj-C Only	☐	GeneralBlock-96	28128	293	180864	1886	
Flatten Recursion	☐	CFArray	14192	380	54336	1604	
Call Tree Constraints	☐	CFArray (store-deque)	16720	213	69328	1324	
	☐	CFNumber	12928	808	17968	1123	
	☐	CFDictionary (key-store)	26304	239	81328	1061	
	☐	CFDictionary (value-store)	26304	239	81328	1061	
	☐	GeneralBlock-224	2016	9	234976	1049	
	☐	GeneralBlock-128	8192	64	102400	799	
	☐	GeneralBlock-176	6336	36	131648	752	
	☐	CFDictionary	19712	298	50048	746	
	☐	GeneralBlock-160	13120	82	118832	743	
	☐	CFString (store)	1104	10	446704	727	
	☐	GeneralBlock-192	3456	18	103296	539	

Object Summary | Category

Also good for tracking memory leaks (wish for more automated way though)

Instruments

Record | Default Target: FlowerGarden | 00:00:27, Run 1 of 1 | Inspection Range | Mini | View

Instruments: ObjectAlloc, Leaks

ObjectAlloc	Graph	Category	Net Bytes	# Net	Overall Bytes	# Overall	# Allocations (Net / Overall)
Allocation Lifespan	☑	* All Allocations *	5050088	14608	17093052	82245	+++
All Objects Created	☐	GeneralBlock-32	75488	2359	702080	21970	
Created & Still Living	☐	GeneralBlock-48	26304	547	747120	15587	
Call Tree	☐	GeneralBlock-16	61616	3850	108464	6907	
Separate by Thread	☐	CFString	75168	2108	236880	6858	
Invert Call Tree	☐	GeneralBlock-80	13136	164	227424	2839	
Hide Missing Symbols	☐	GeneralBlock-416	4992	12	1131520	2720	
Hide System Libraries	☐	GeneralBlock-208	4992	24	511056	2457	
Show Obj-C Only	☐	GeneralBlock-96	28128	293	180864	1886	
Flatten Recursion	☐	CFArray	14192	380	54336	1604	
Call Tree Constraints	☐	CFArray (store-deque)	16720	213	69328	1324	
	☐	CFNumber	12928	808	17968	1123	
	☐	CFDictionary (key-store)	26304	239	81328	1061	
	☐	CFDictionary (value-store)	26304	239	81328	1061	
	☐	GeneralBlock-224	2016	9	234976	1049	
	☐	GeneralBlock-128	8192	64	102400	799	
	☐	GeneralBlock-176	6336	36	131648	752	
	☐	CFDictionary	19712	298	50048	746	
	☐	GeneralBlock-160	13120	82	118832	743	
	☐	CFString (store)	1104	10	446704	727	
	☐	GeneralBlock-192	3456	18	103296	539	

Object Summary | Category

Also good for tracking memory leaks (wish for more automated way though)

Instruments

snappy touch

Instruments

- First pass with Instruments confirmed that Flower Garden was simulation bound.

Instruments

- First pass with Instruments confirmed that Flower Garden was simulation bound.
- Especially the matrix multiplies and geometry creation sections.

Shark

Shark

- CPU usage

snappy touch

Shark

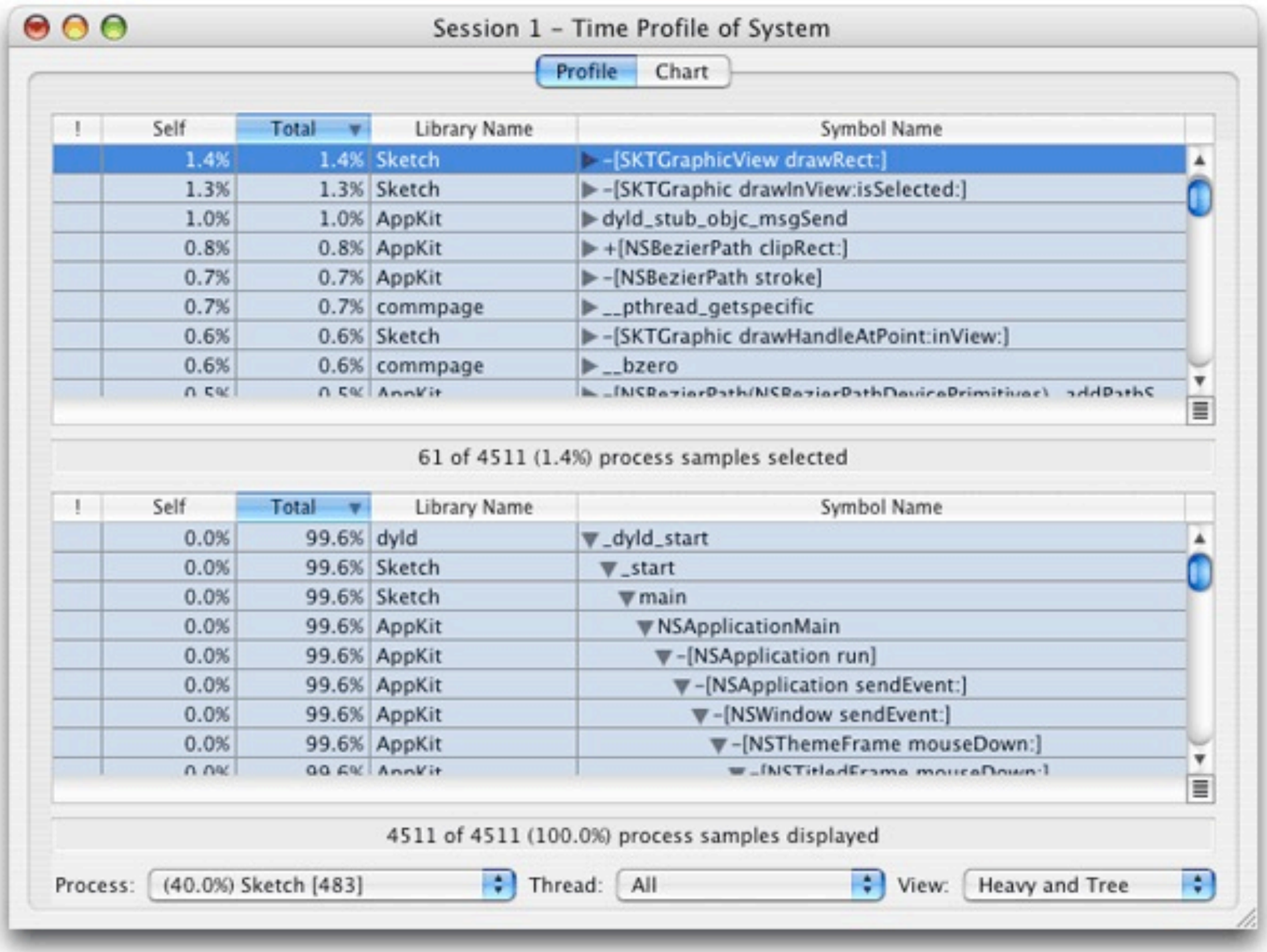
- CPU usage
- More in depth than Instruments

snappy touch

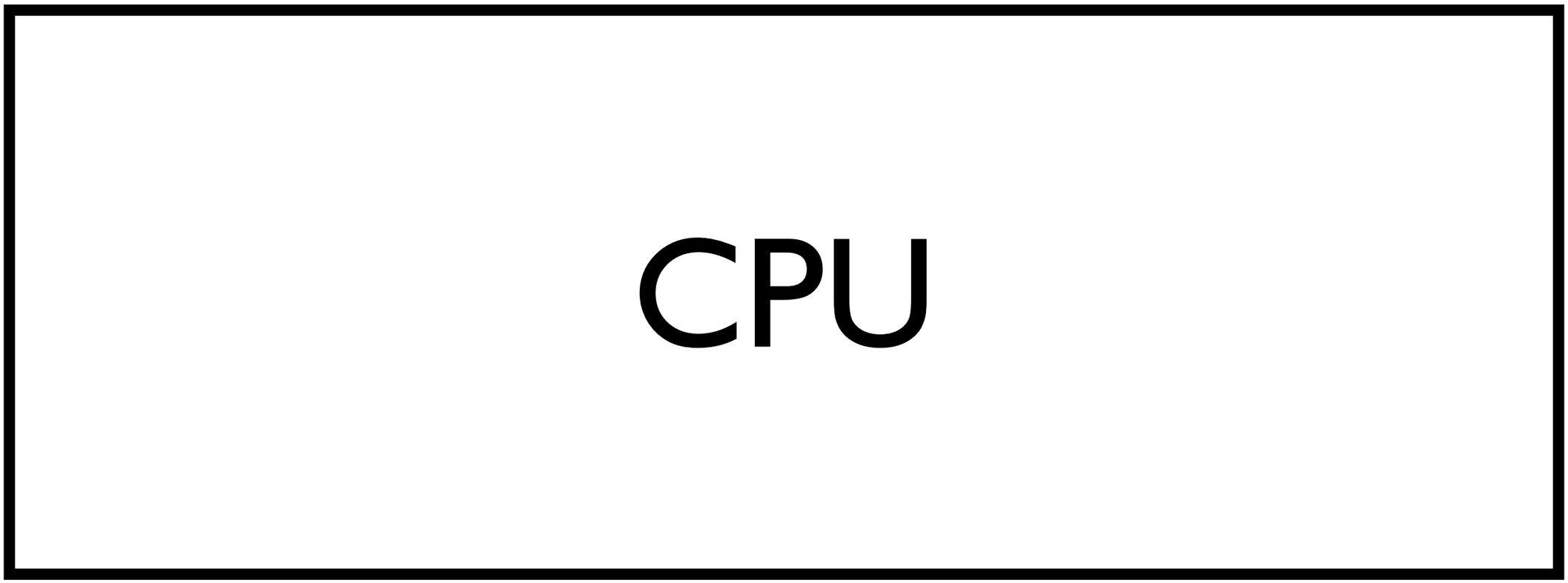
Shark

- CPU usage
- More in depth than Instruments
- Can even report good info on cache misses

Shark

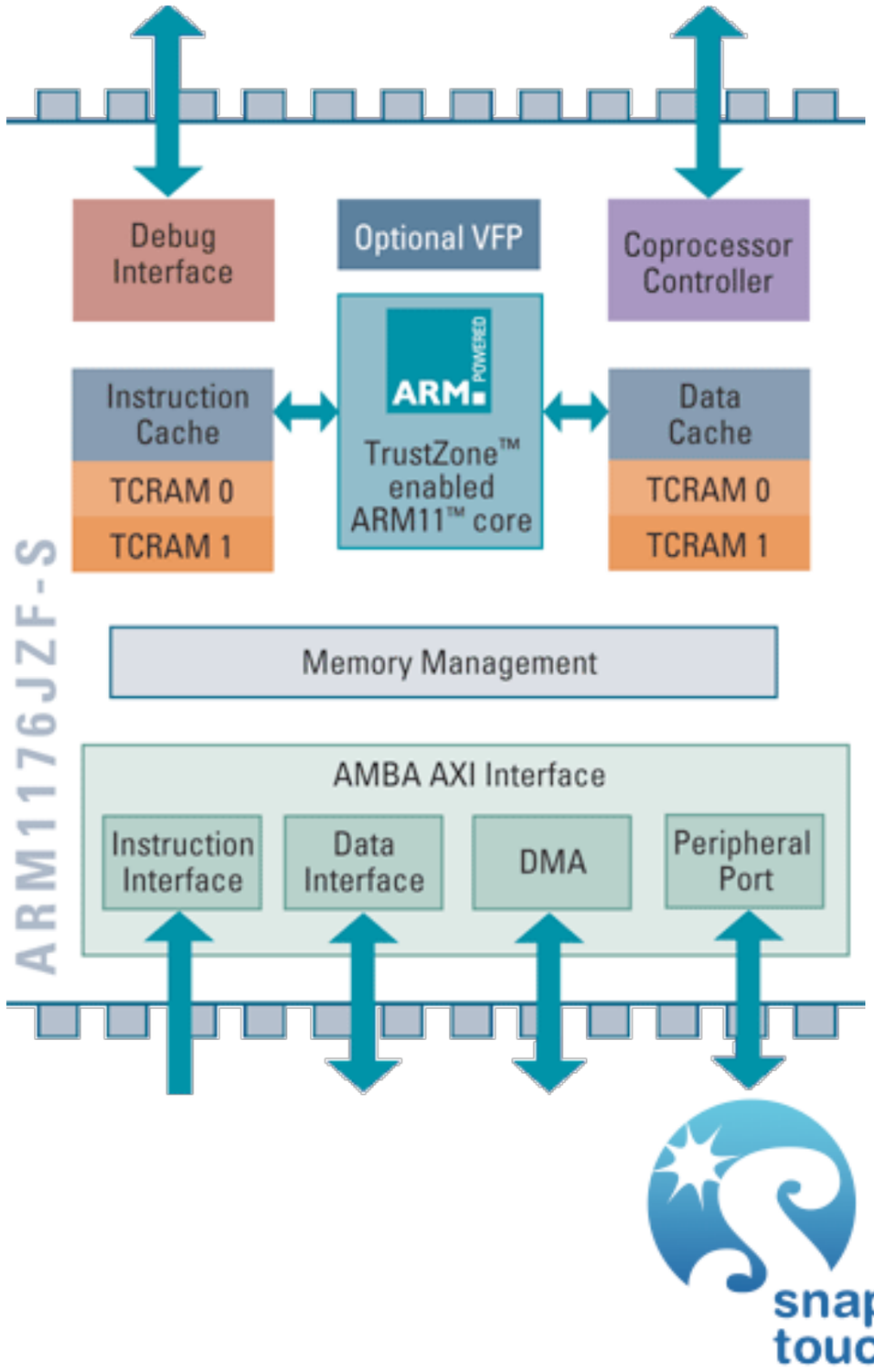


snappy touch



snappy touch

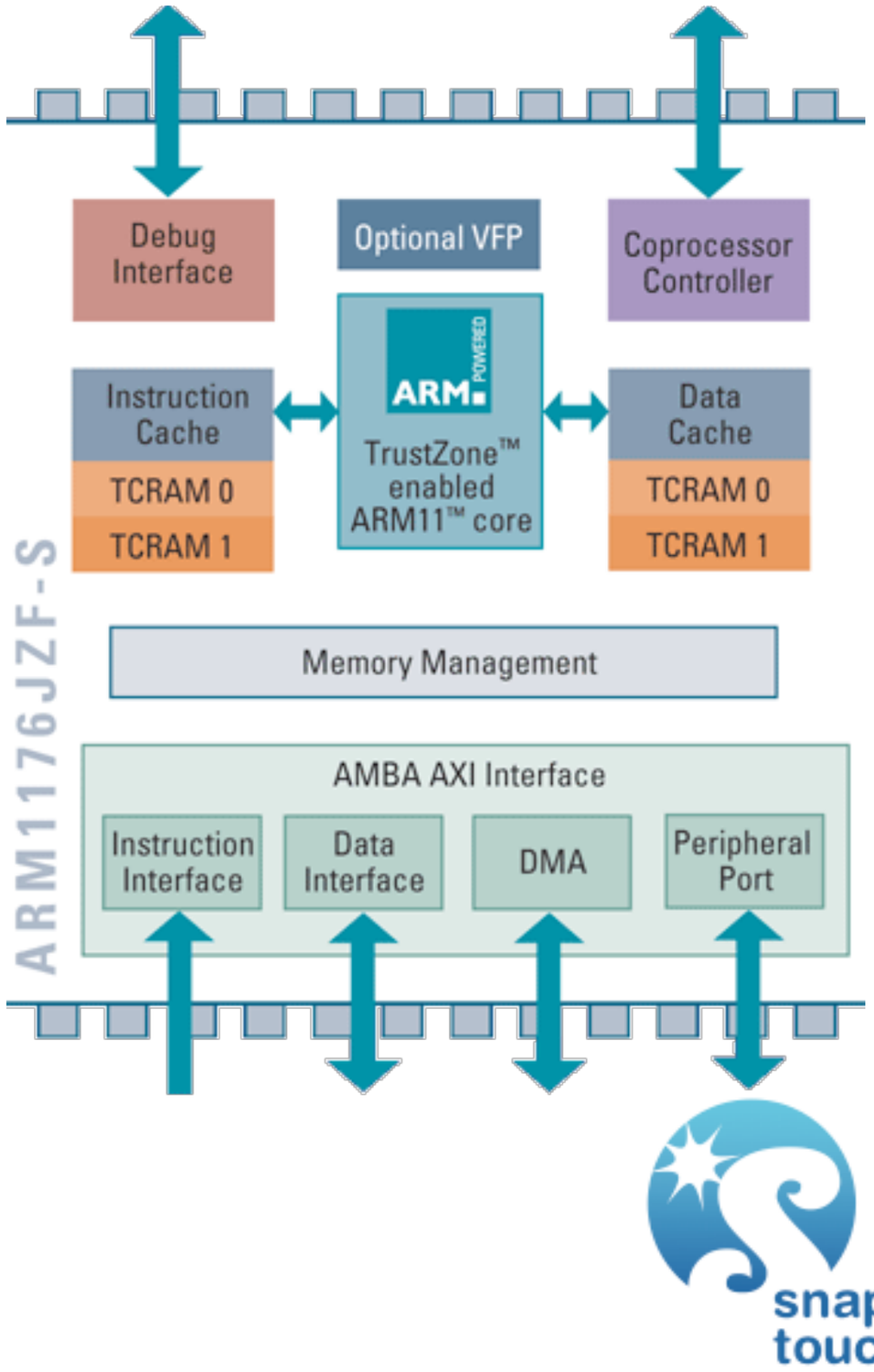
CPU



This is where the biggest bottlenecks were for Flower Garden at first

CPU

- 32-bit RISC ARM 11

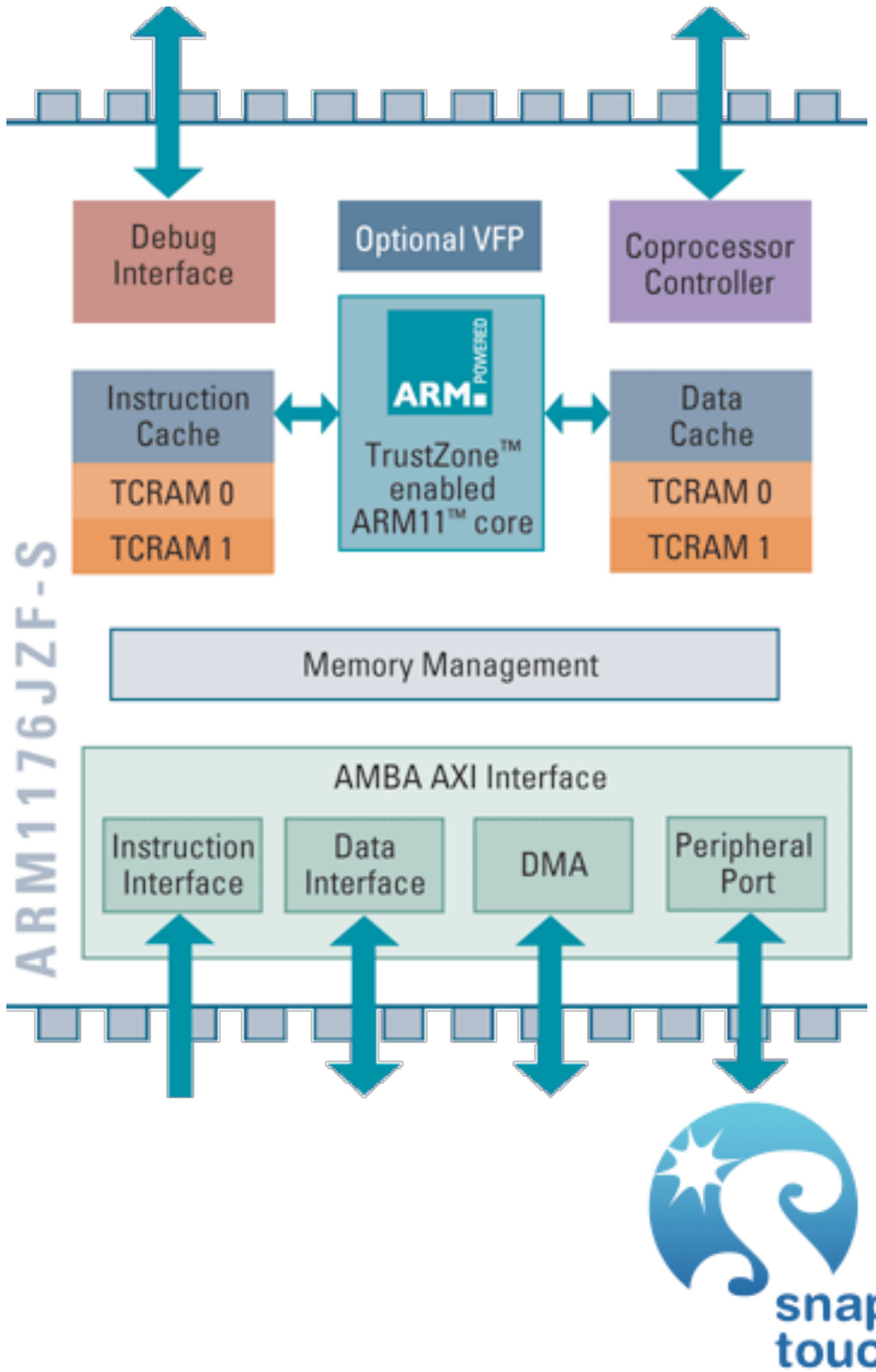


snappy touch

This is where the biggest bottlenecks were for Flower Garden at first

CPU

- 32-bit RISC ARM 11
- 400-535Mhz

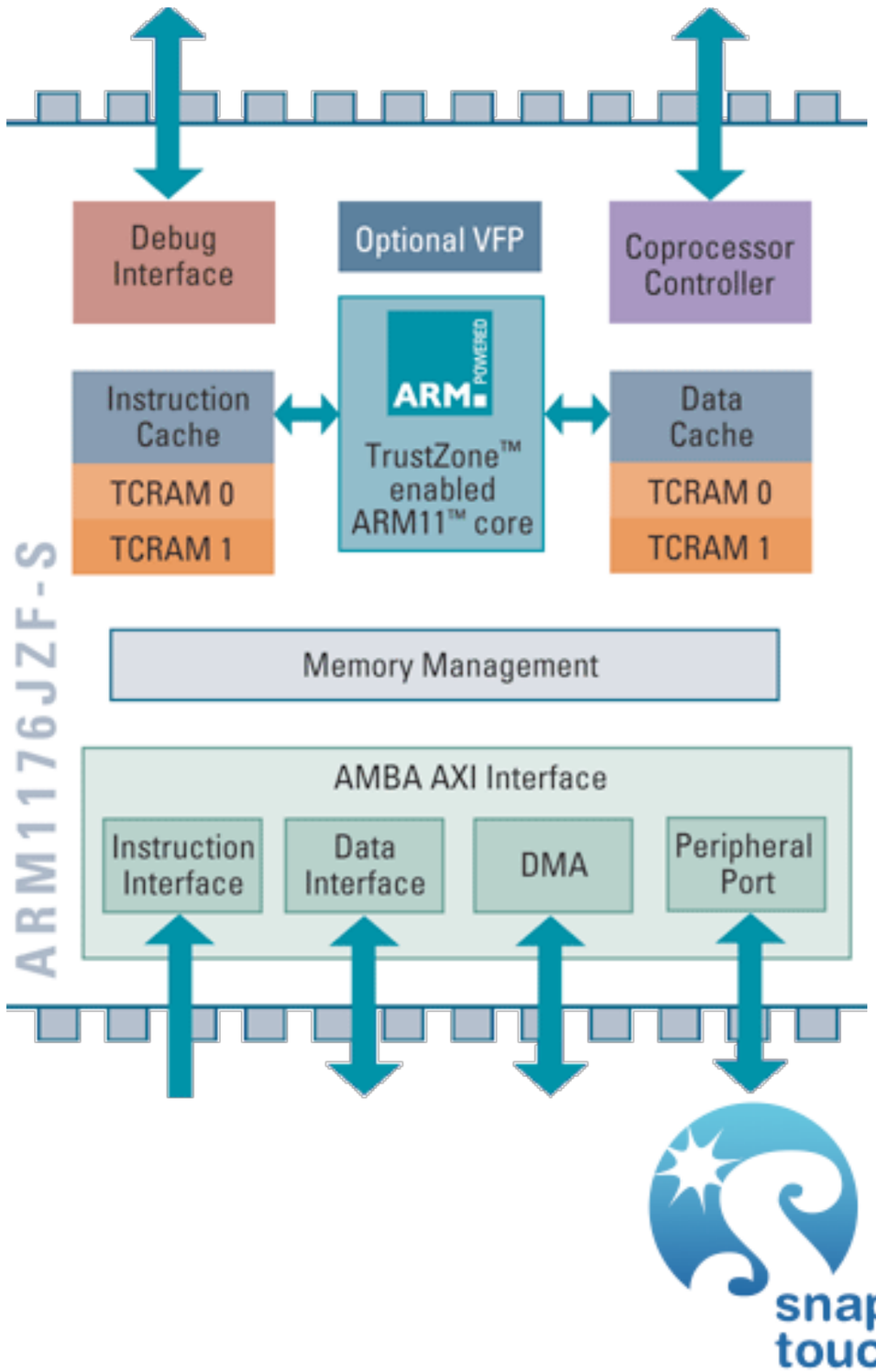


snappy touch

This is where the biggest bottlenecks were for Flower Garden at first

CPU

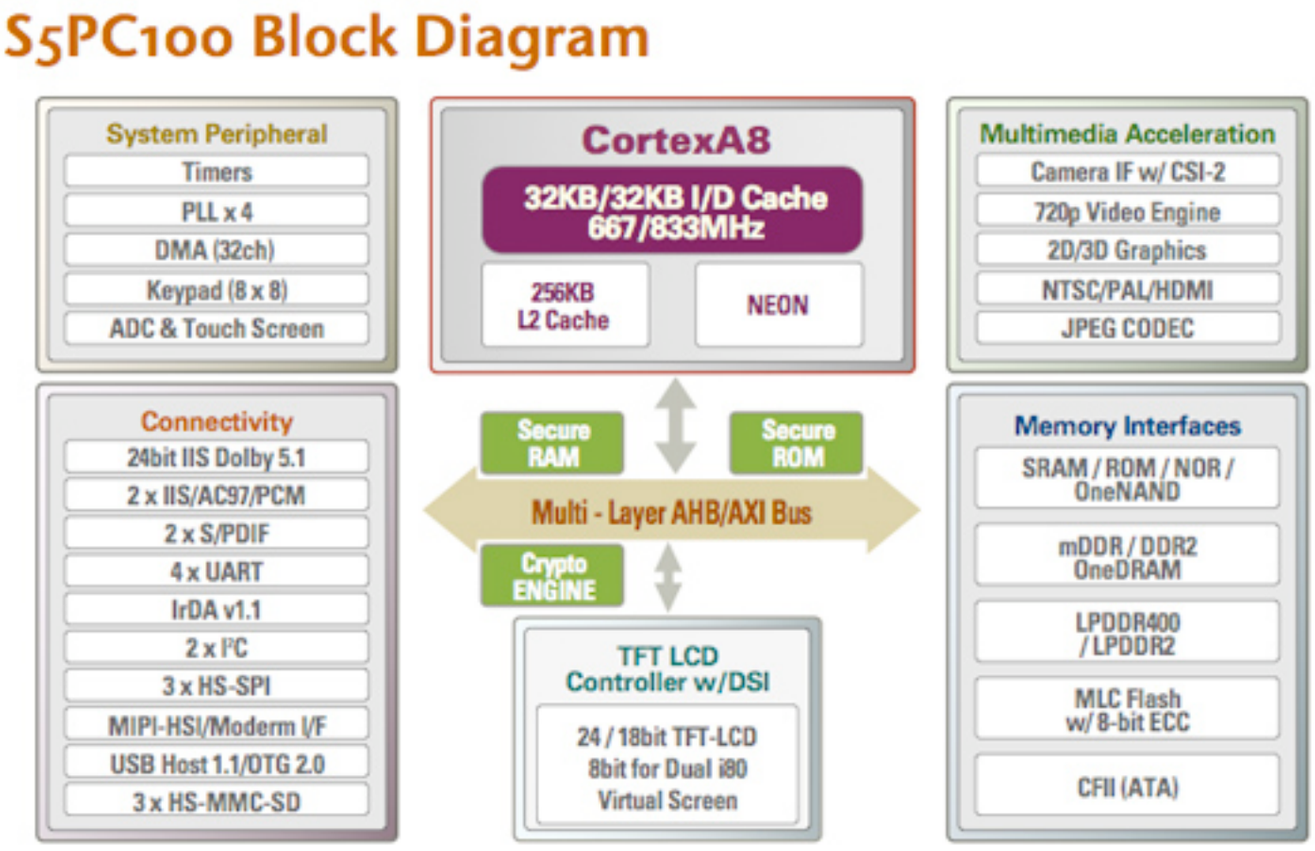
- 32-bit RISC ARM 11
- 400-535Mhz
- iPhone 2G/3G and iPod Touch 1st and 2nd gen



snappy touch

This is where the biggest bottlenecks were for Flower Garden at first

CPU (iPhone 3GS)

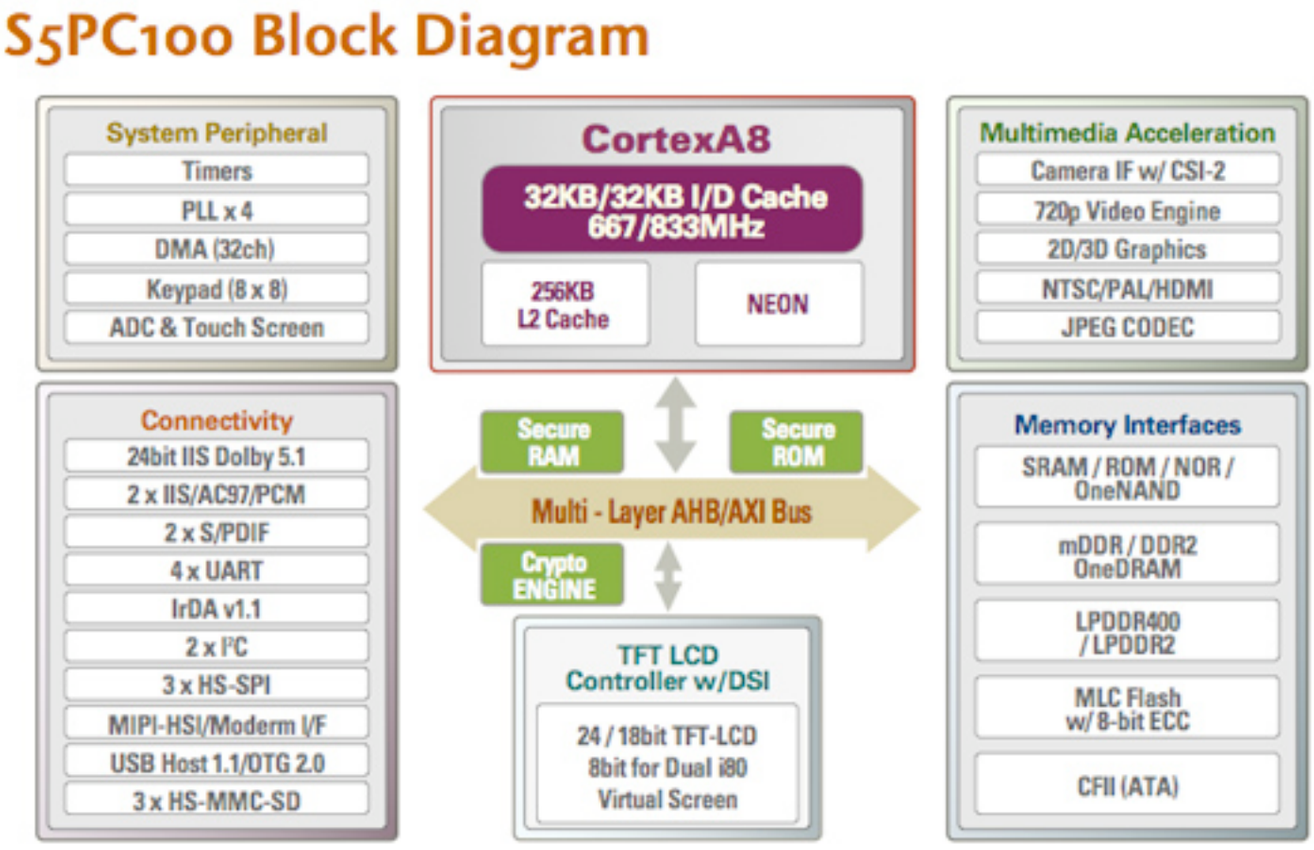


snappy touch

Still single core

CPU (iPhone 3GS)

- Cortex-A8 600MHz

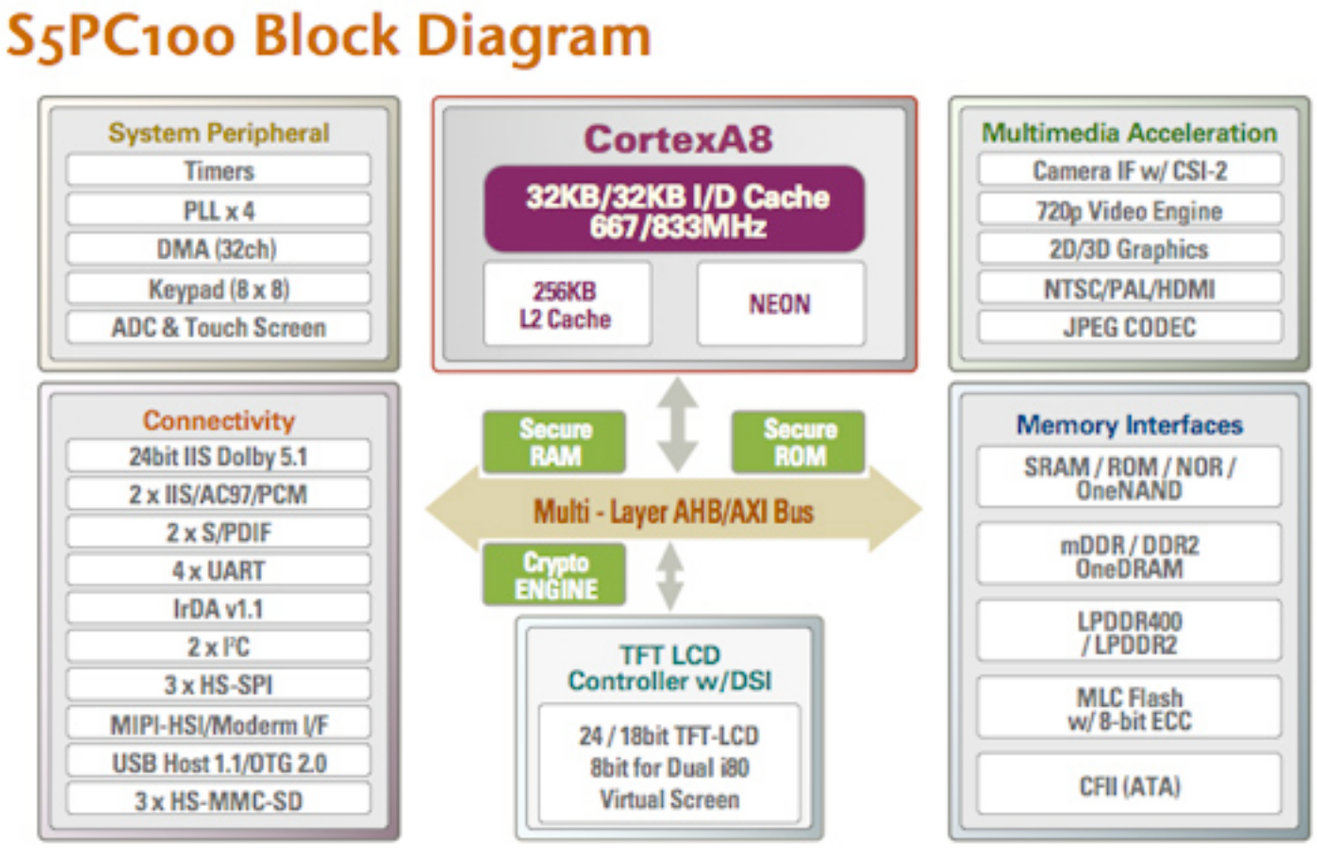


snappy touch

Still single core

CPU (iPhone 3GS)

- Cortex-A8 600MHz
- More advanced architecture



snappy touch

Still single core

Thumb Mode

Thumb Mode

snappy touch

Thumb Mode

- CPU has a special thumb mode.

snappy touch

Thumb Mode

- CPU has a special thumb mode.
- Less memory, maybe better performance.

snappy touch

Thumb Mode

- CPU has a special thumb mode.
- Less memory, maybe better performance.
- No floating point support.

snappy touch

Thumb Mode

- CPU has a special thumb mode.
- Less memory, maybe better performance.
- No floating point support.
- It's on by default!

snappy touch

Thumb Mode

- CPU has a special thumb mode.
- Less memory, maybe better performance.
- No floating point support.
- It's on by default!
- Potentially HUGE wins turning it off.

snappy touch

Thumb Mode

- CPU has a special thumb mode.
- Less memory, maybe better performance.
- No floating point support.
- It's on by default!
- Potentially HUGE wins turning it off.

Show Warnings	☑
ser-Defined	
GCC_C_LANGUAGE_STANDARD	c99
GCC_DYNAMIC_NO_PIC	**NO**
GCC_OBJC_CALL_CXX_CDTORS	**YES**
GCC_OPTIMIZATION_LEVEL	**0**
GCC_PRECOMPILE_PREFIX_HEADER	**YES**
GCC_PREFIX_HEADER	**FlowerGarden_Prefix.pch**
GCC_THUMB_SUPPORT	**NO**
GCC_WARN_ABOUT_MISSING_NEWLINE	YES
GCC_WARN_ABOUT_RETURN_TYPE	YES
GCC_WARN_CHECK_SWITCH_STATEMENTS	YES

snappy touch

Thumb Mode

Thumb Mode

- Turning off Thumb mode increased performance in Flower Garden by over 2x

Thumb Mode

- Turning off Thumb mode increased performance in Flower Garden by over 2x
- Heavy usage of floating point operations though

snappy touch

Thumb Mode

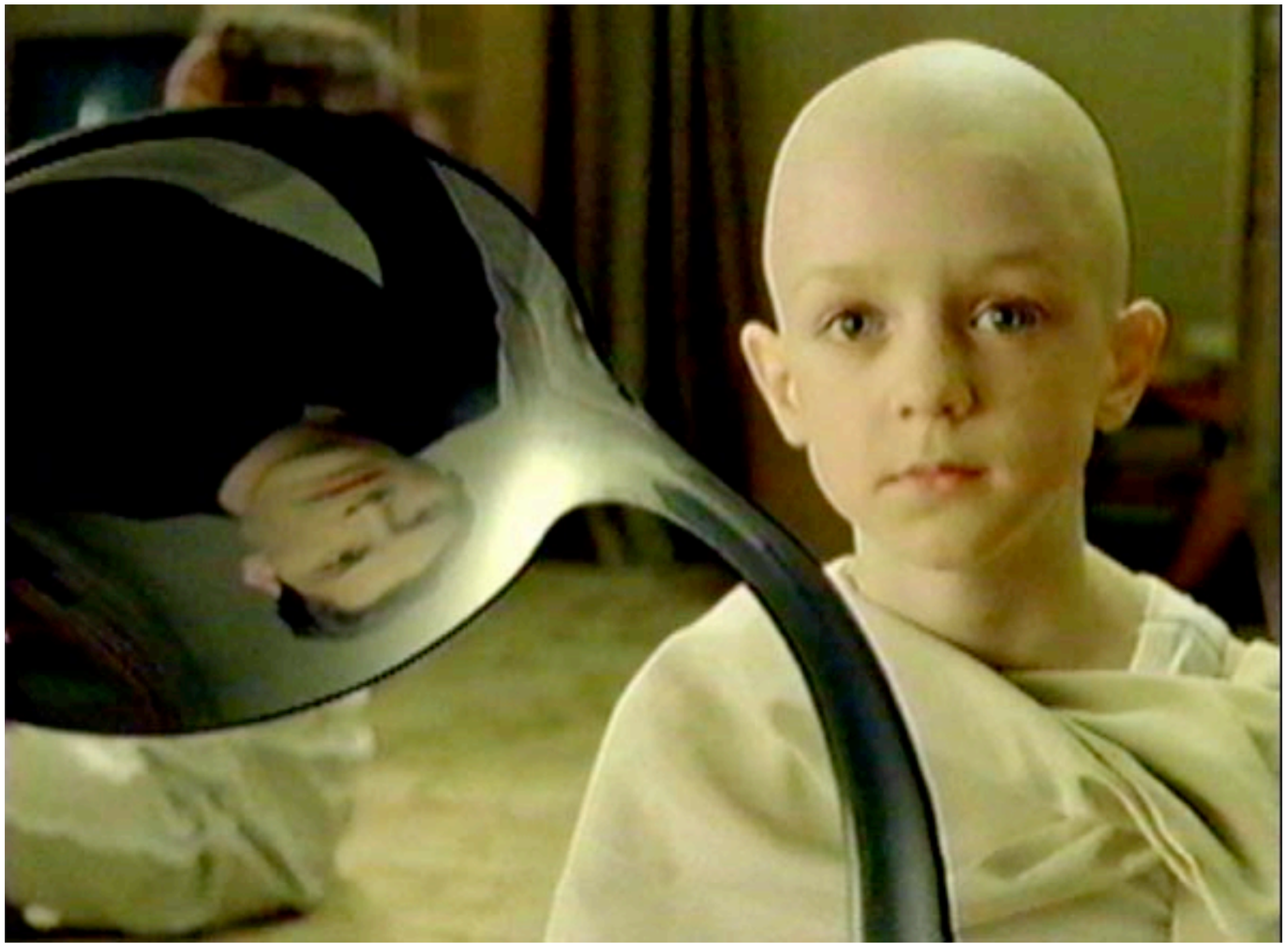
- Turning off Thumb mode increased performance in Flower Garden by over 2x
- Heavy usage of floating point operations though
- Most games will probably benefit from turning it off (especially 3D games)

snappy touch

Integer Divide

So plan accordingly!

Integer Divide

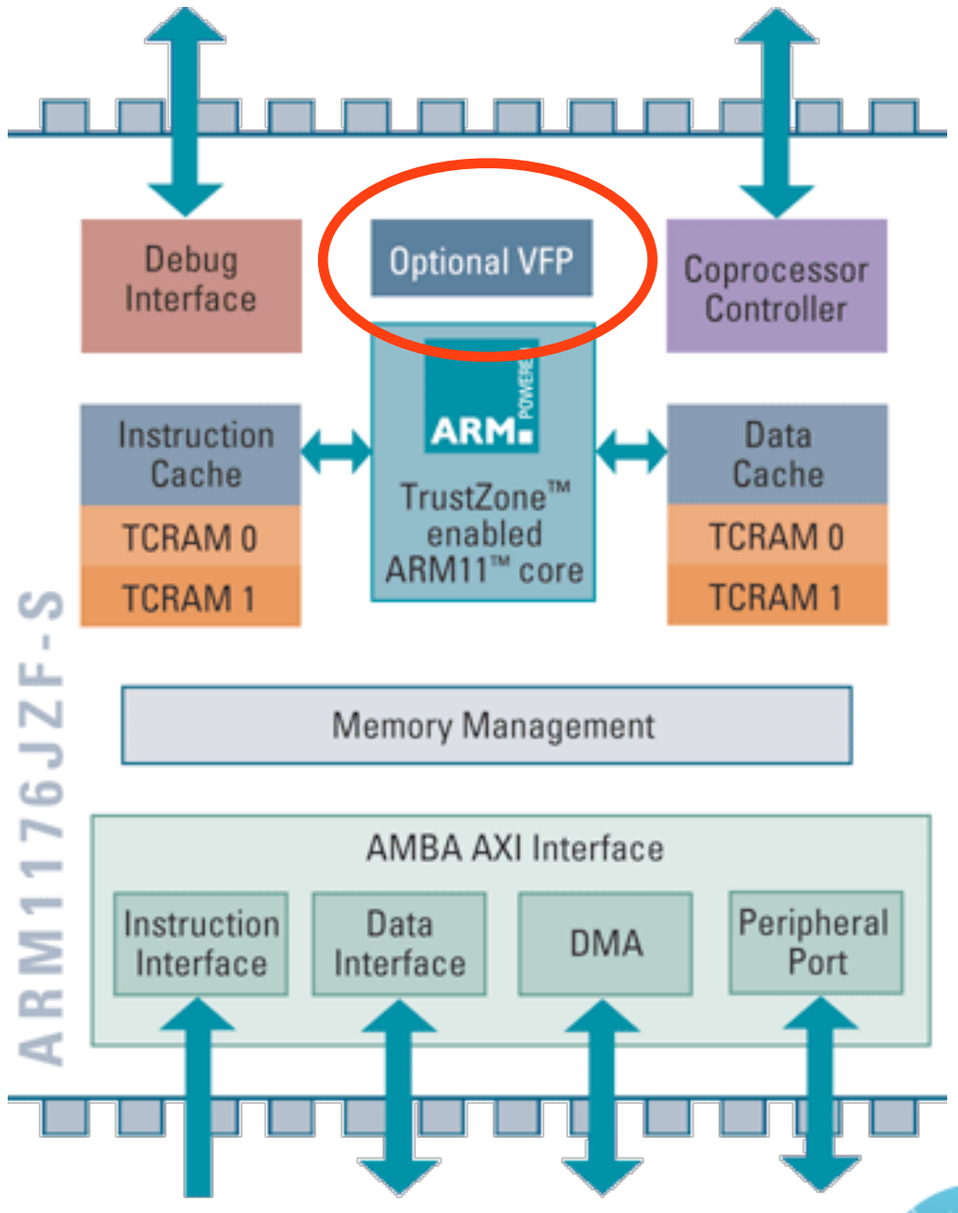


There is no integer divide

snappy touch

So plan accordingly!

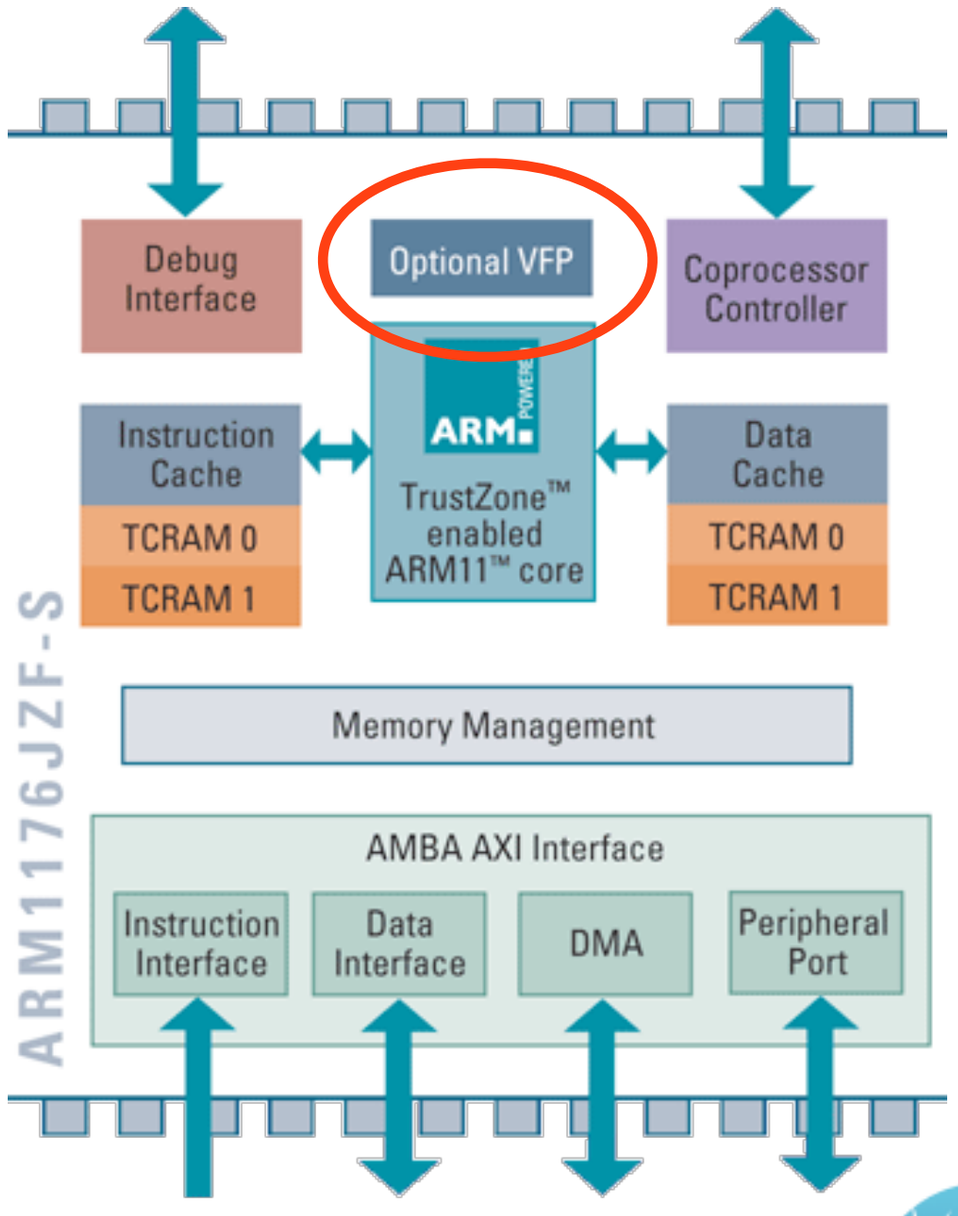
Vector FP Unit



snappy touch

Vector FP Unit

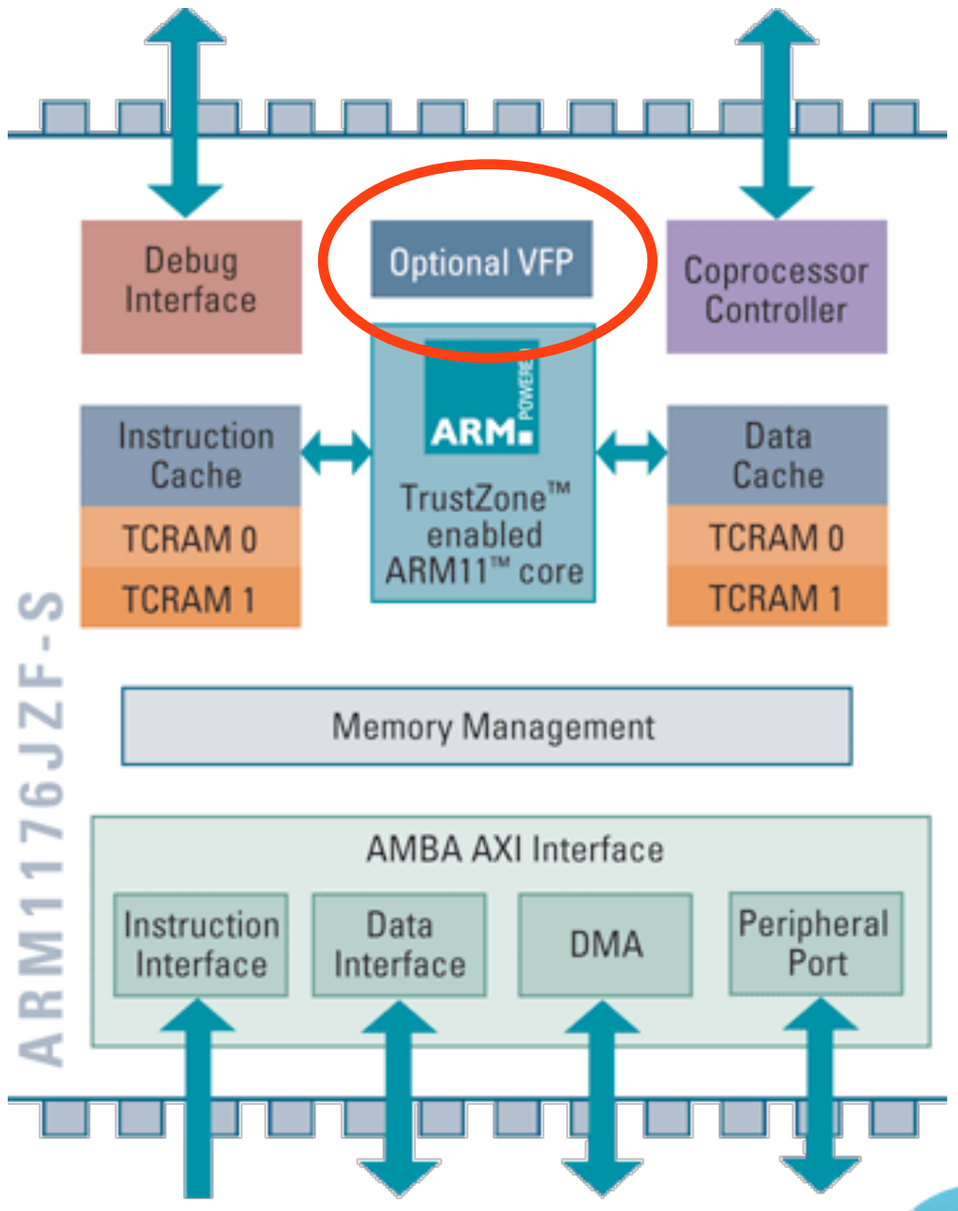
- The main CPU has no floating point support.



snappy touch

Vector FP Unit

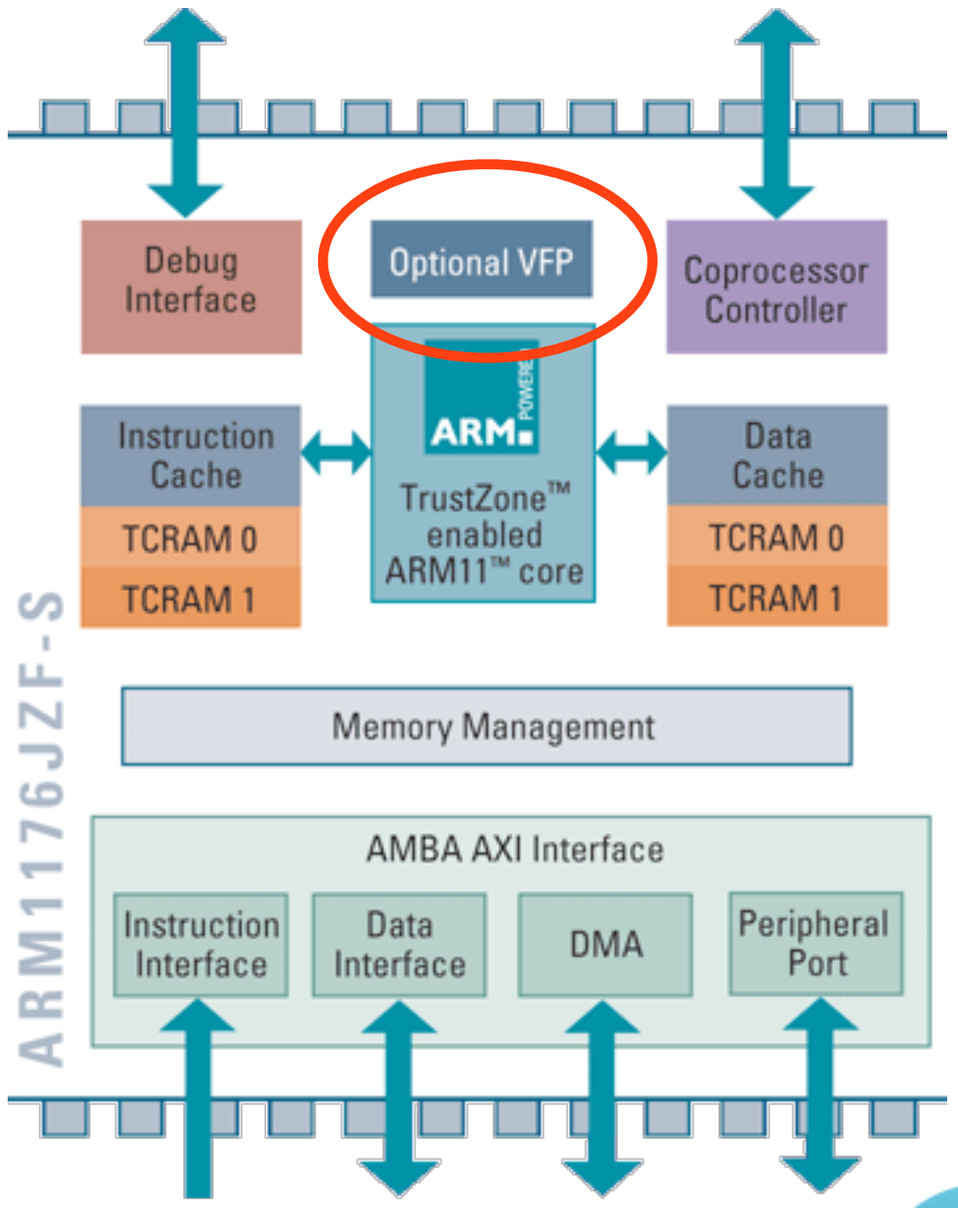
- The main CPU has no floating point support.
- Compiled C/C++/ObjC code uses the vector floating point unit for any floating point operations.



snappy touch

Vector FP Unit

- The main CPU has no floating point support.
- Compiled C/C++/ObjC code uses the vector floating point unit for any floating point operations.
- Can program the VFP in assembly for max performance.



snappy touch

Vector FP Unit

Vector FP Unit

- Big win when many floating point operations can be vectorized.

Vector FP Unit

- Big win when many floating point operations can be vectorized.
- Matrix multiplies, skinning, etc

snappy touch

Vector FP Unit

- Big win when many floating point operations can be vectorized.
- Matrix multiplies, skinning, etc
- vfpmath project in Google Code

Vector FP Unit

```
void Matrix4Mul(const float* src_mat_1, const float* src_mat_2,
float* dst_mat)
{
  asm volatile (VFP_SWITCH_TO_ARM
                VFP_VECTOR_LENGTH(3)

                "fldmias  %2, {s8-s23}     \n\t"
                "fldmias  %1!, {s0-s3}     \n\t"
                "fmuls s24, s8, s0         \n\t"
                "fmacs s24, s12, s1        \n\t"
//...
```

Game Loop Architecture

Game loop

Main difference between games and apps is the constant amount of “stuff” happening on screen.

Event-driven architecture is not a very good match for games

Game loop

Gather input

snappy touch

Main difference between games and apps is the constant amount of “stuff” happening on screen.

Event-driven architecture is not a very good match for games

Game loop

Gather input

Update state

snappy touch

Main difference between games and apps is the constant amount of “stuff” happening on screen.

Event-driven architecture is not a very good match for games

Game loop

Gather input

Update state

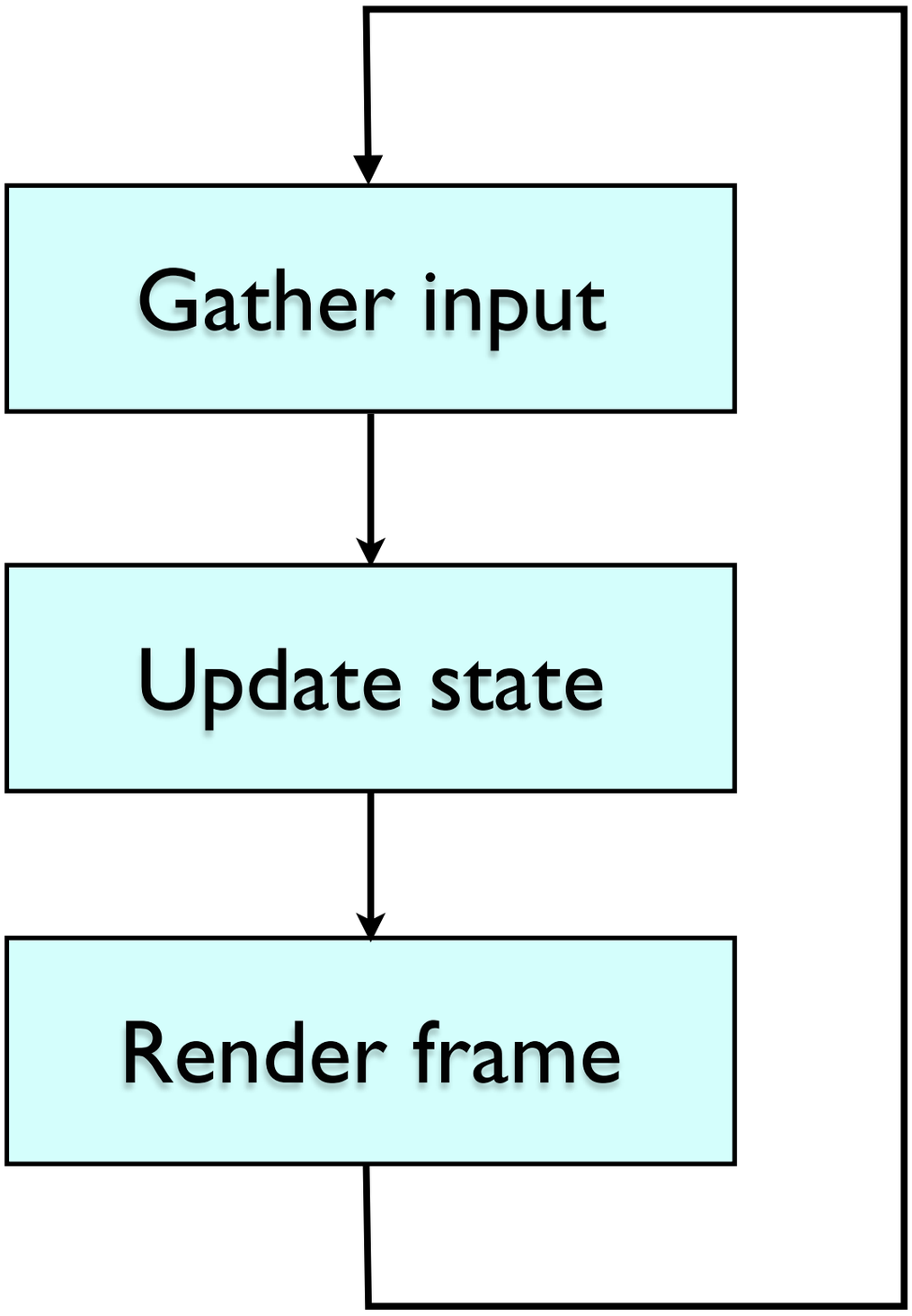
Render frame

snappy touch

Main difference between games and apps is the constant amount of “stuff” happening on screen.

Event-driven architecture is not a very good match for games

Game loop



snappy touch

Main difference between games and apps is the constant amount of “stuff” happening on screen.

Event-driven architecture is not a very good match for games

NSTimer

OK for games that don’t require a high/steady frame rate

NSTimer

- Easiest way to drive the main loop

OK for games that don’t require a high/steady frame rate

NSTimer

- Easiest way to drive the main loop
- All the samples from Apple

OK for games that don’t require a high/steady frame rate

NSTimer

- Easiest way to drive the main loop
- All the samples from Apple

```
m_gameLoopTimer = [NSTimer
      scheduledTimerWithTimeInterval:animationInterval
      target:self
      selector:@selector(doFrame)
      userInfo:nil
      repeats:YES];
```

OK for games that don’t require a high/steady frame rate

NSTimer

NSTimer

- Not very accurate.
  Frames can vary by as much as 5-10 ms!

NSTimer

- Not very accurate.
  Frames can vary by as much as 5-10 ms!

Event

snappy touch

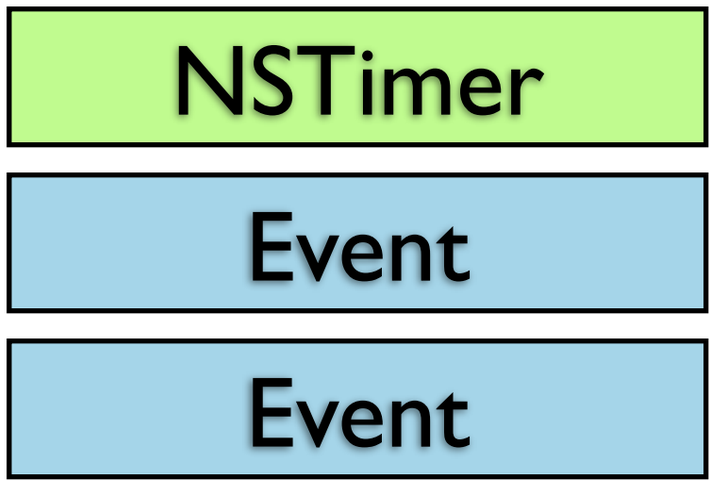
NSTimer

- Not very accurate.
  Frames can vary by as much as 5-10 ms!

snappy touch

NSTimer

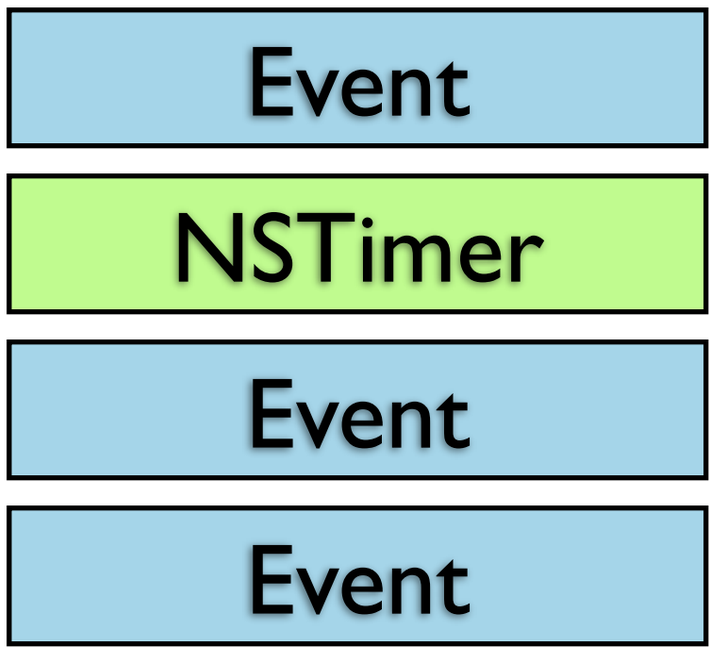
- Not very accurate.
  Frames can vary by as much as 5-10 ms!



snappy touch

NSTimer

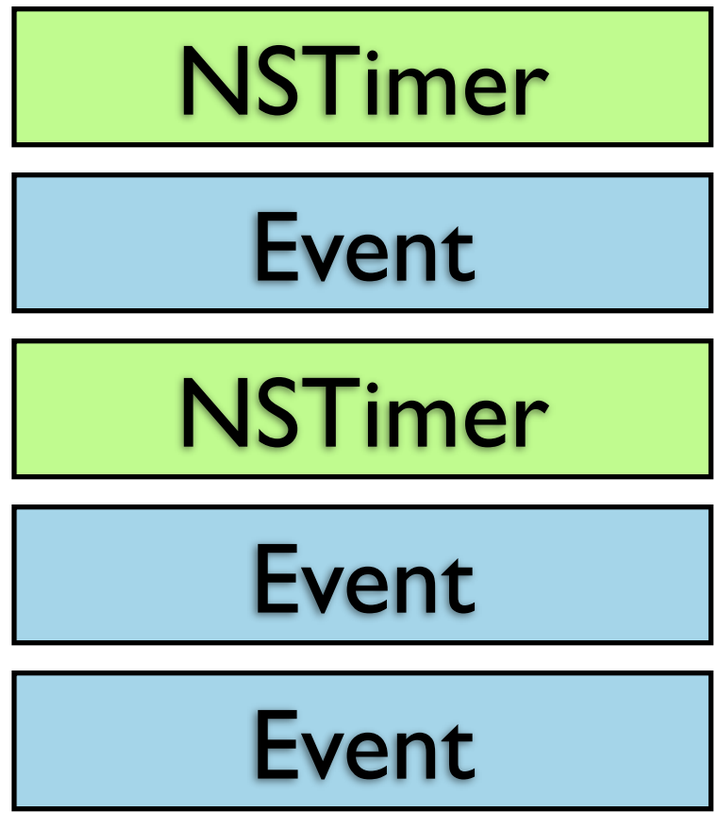
- Not very accurate. Frames can vary by as much as 5-10 ms!



snappy touch

NSTimer

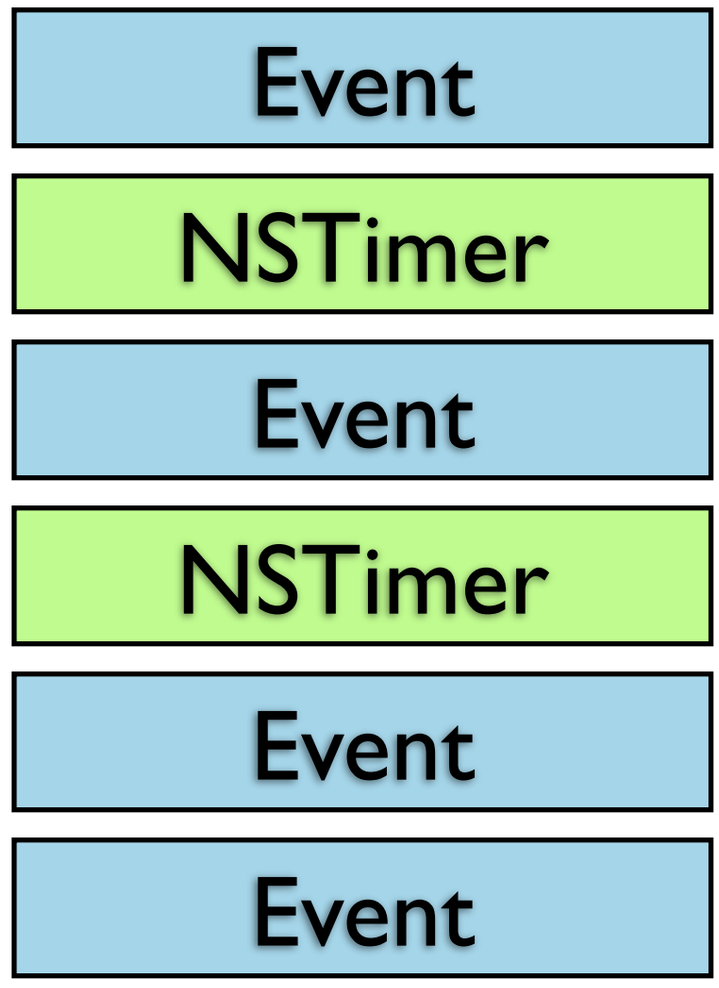
- Not very accurate. Frames can vary by as much as 5-10 ms!



snappy touch

NSTimer

- Not very accurate. Frames can vary by as much as 5-10 ms!



snappy touch

NSTimer

The app becomes even less responsive

NSTimer

- Call it at a higher frequency so main loop is called right away.

The app becomes even less responsive

NSTimer

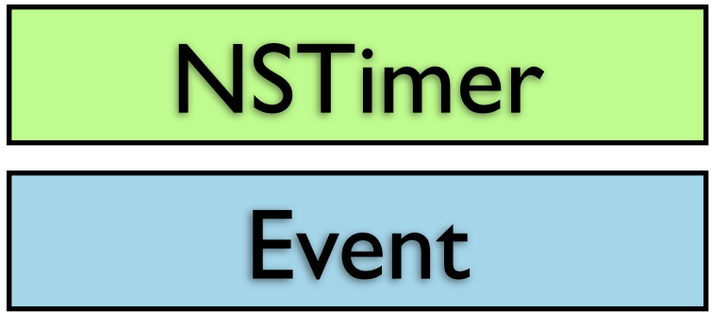
- Call it at a higher frequency so main loop is called right away.

Event

The app becomes even less responsive

NSTimer

- Call it at a higher frequency so main loop is called right away.

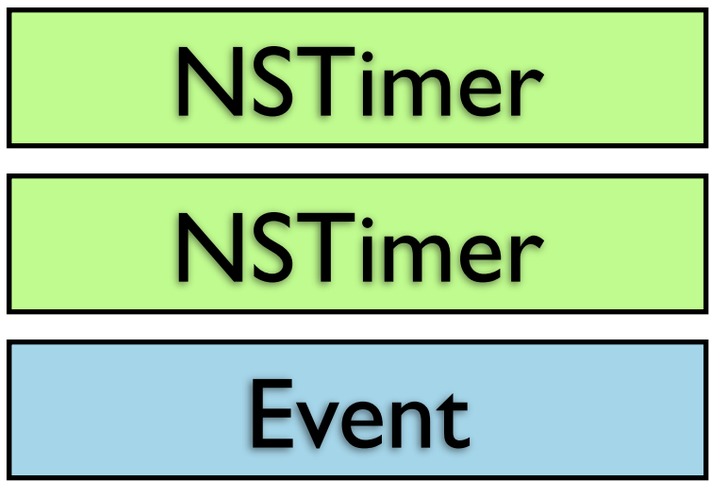


snappy touch

The app becomes even less responsive

NSTimer

- Call it at a higher frequency so main loop is called right away.

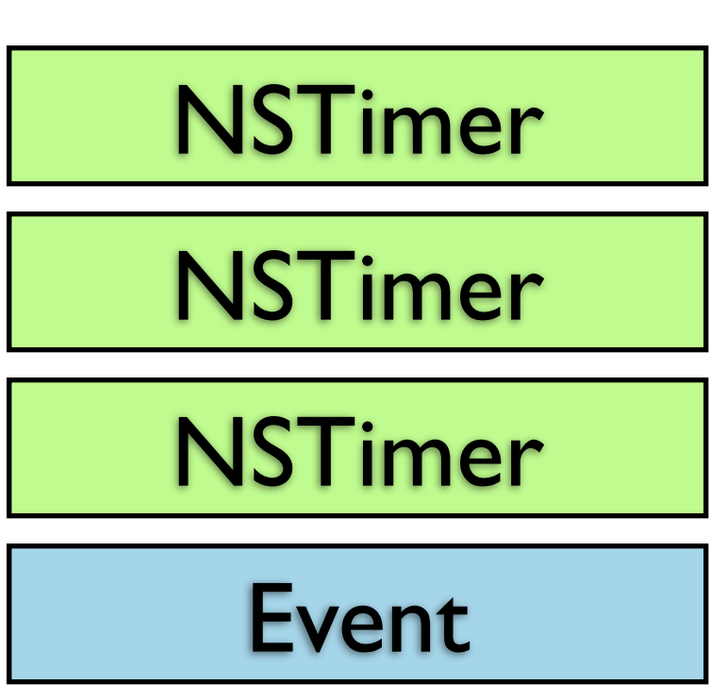


snappy touch

The app becomes even less responsive

NSTimer

- Call it at a higher frequency so main loop is called right away.

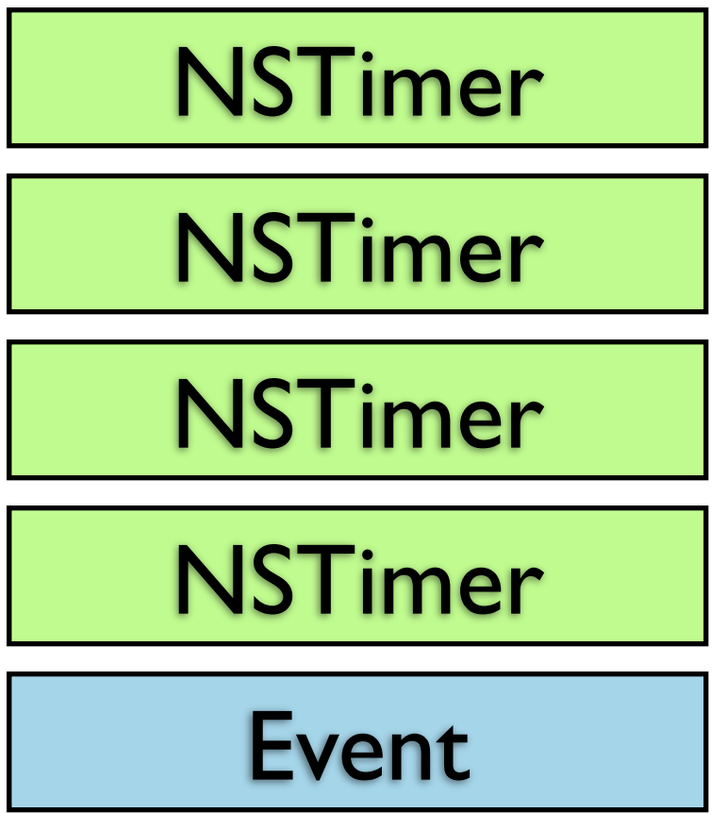


snappy touch

The app becomes even less responsive

NSTimer

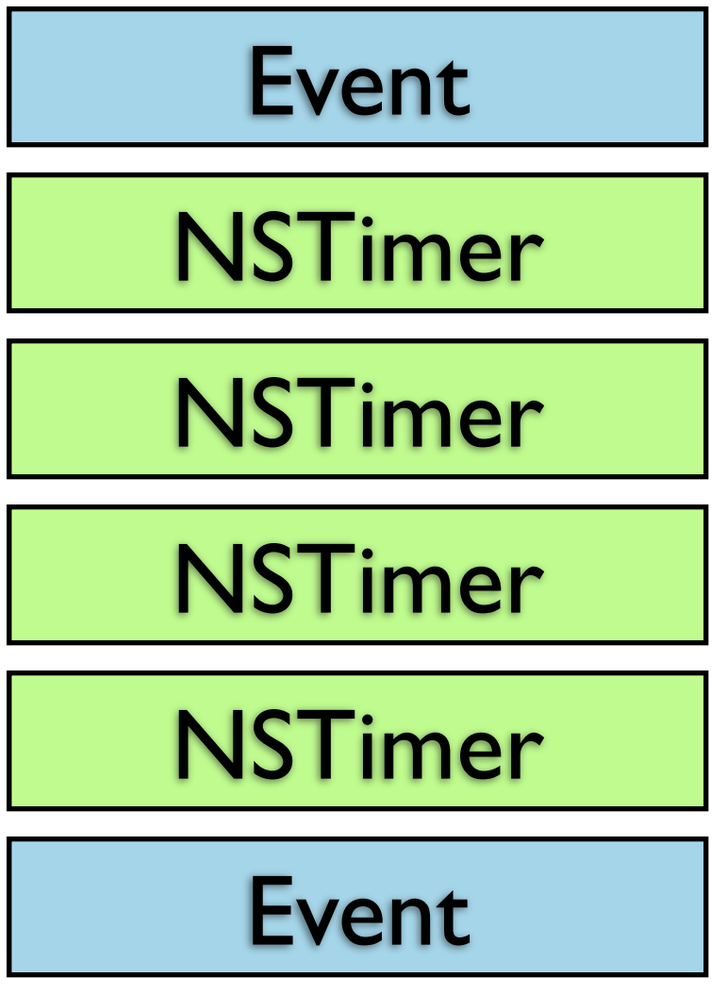
- Call it at a higher frequency so main loop is called right away.



The app becomes even less responsive

NSTimer

- Call it at a higher frequency so main loop is called right away.



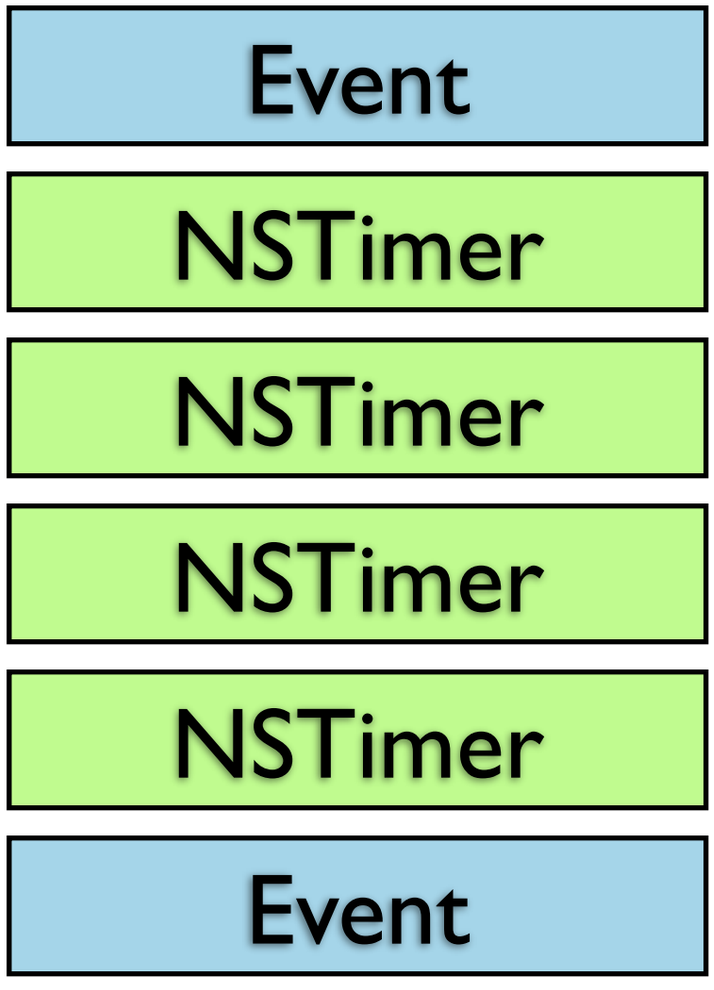
snappy touch

The app becomes even less responsive

NSTimer

- Call it at a higher frequency so main loop is called right away.

- Unfortunately that floods the message queue and causes delay on touch events



snappy touch

The app becomes even less responsive

Threads

Threads

- Put the game on a separate thread from the UI one.

Threads

- Put the game on a separate thread from the UI one.
- Runs as fast as possible without delays

snappy touch

Threads

Threads

- Careful what you do when parsing input events from main thread

Threads

- Careful what you do when parsing input events from main thread
- Can be more complex to debug

Threads

- Careful what you do when parsing input events from main thread
- Can be more complex to debug
- Running as fast as possible might not always be desirable (draining battery life)

Threads

Threads

- Can also split game update and rendering in two different threads.

Threads

- Can also split game update and rendering in two different threads.
- Syncing the two is much more difficult

Threads

- Can also split game update and rendering in two different threads.
- Syncing the two is much more difficult
- Can only use one thread for each OpenGL context

snappy touch

Threads

- Can also split game update and rendering in two different threads.
- Syncing the two is much more difficult
- Can only use one thread for each OpenGL context
- Theoretically best results, but maybe not much difference

snappy touch

Thread Driving Loop

That’s what I’m using in Flower Garden

Thread Driving Loop

```
m_mainLoop = [[NSThread alloc] initWithTarget:self
                selector:@selector(mainLoopTimer) object:nil];
[m_mainLoop start];
```

That’s what I’m using in Flower Garden

Thread Driving Loop

```
m_mainLoop = [[NSThread alloc] initWithTarget:self
                selector:@selector(mainLoopTimer) object:nil];
[m_mainLoop start];
```

```
- (void)mainLoopTimer
{
  while (![[NSThread currentThread] isCancelled])
  {
      [self performSelectorOnMainThread:@selector(doFrame)
                withObject:nil waitUntilDone:YES];
      usleep(2000);
  }
}
```

That’s what I’m using in Flower Garden

Thread driving loop

Thread driving loop

- Can be best of both worlds

Thread driving loop

- Can be best of both worlds
- Consistent call to main loop

Thread driving loop

- Can be best of both worlds
- Consistent call to main loop
- No flooding of events (cuts in line)

snappy touch

CADisplayLink

CADisplayLink

- Triggered by display refresh

CADisplayLink

- Triggered by display refresh
- Can choose to get a call every X updates

CADisplayLink

- Triggered by display refresh
- Can choose to get a call every X updates
- Consistent update

CADisplayLink

- Triggered by display refresh
- Can choose to get a call every X updates
- Consistent update
- Perfect for games

CADisplayLink

- Triggered by display refresh
- Can choose to get a call every X updates
- Consistent update
- Perfect for games

```
m_displayLink  = [CADisplayLink displayLinkWithTarget: selfselector:@selector(mainLoop:)];
m_displayLink.frameInterval = 2;
[m_displayLink addToRunLoop :[NSRunLoopcurrentRunLoop] forMode:NSDefaultRunLoopMode];
```

CADisplayLink

CADisplayLink

- Only available on SDK 3.1

CADisplayLink

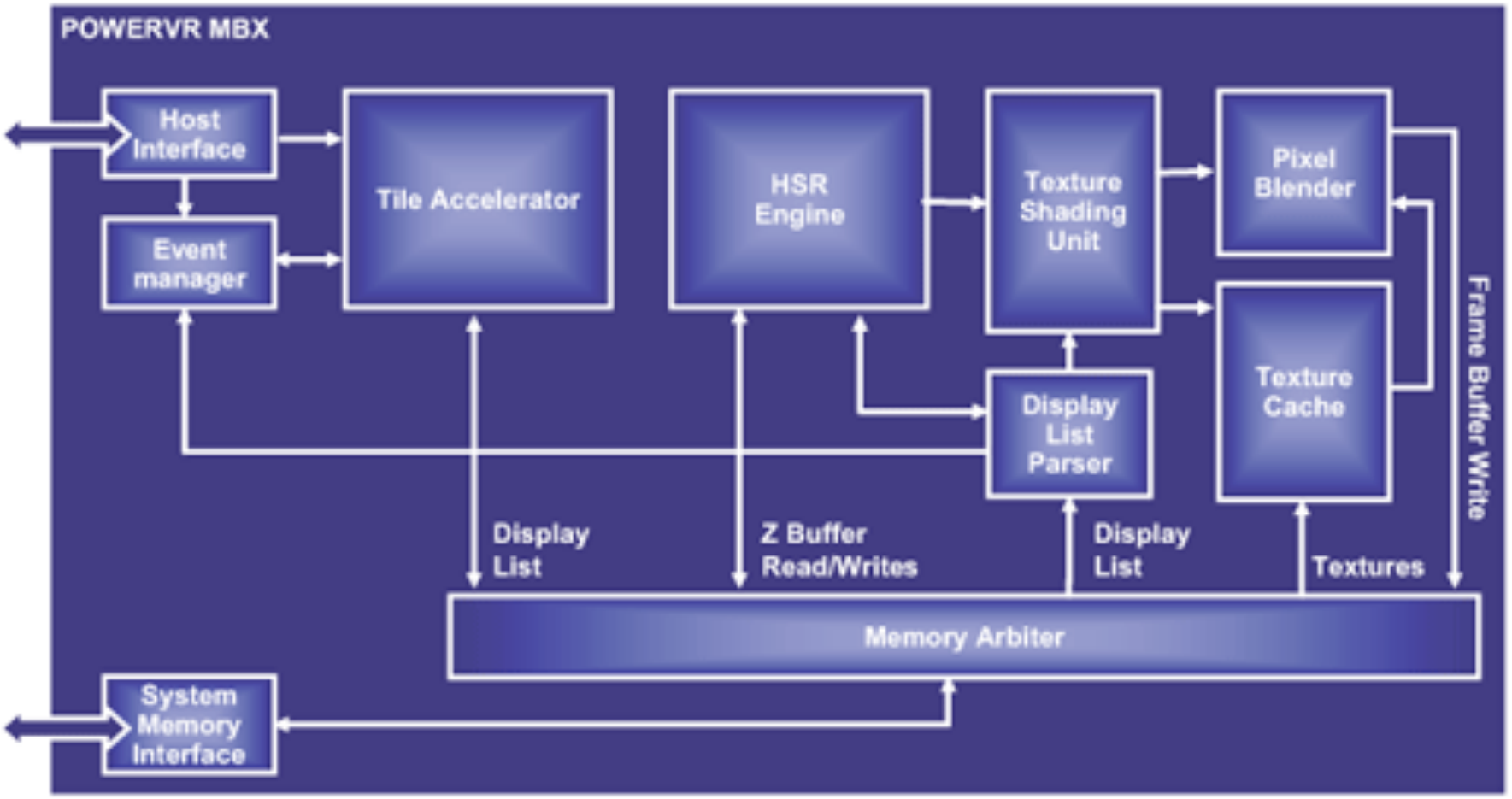
- Only available on SDK 3.1
- Definitely the way to go for future games

snappy touch

Rendering

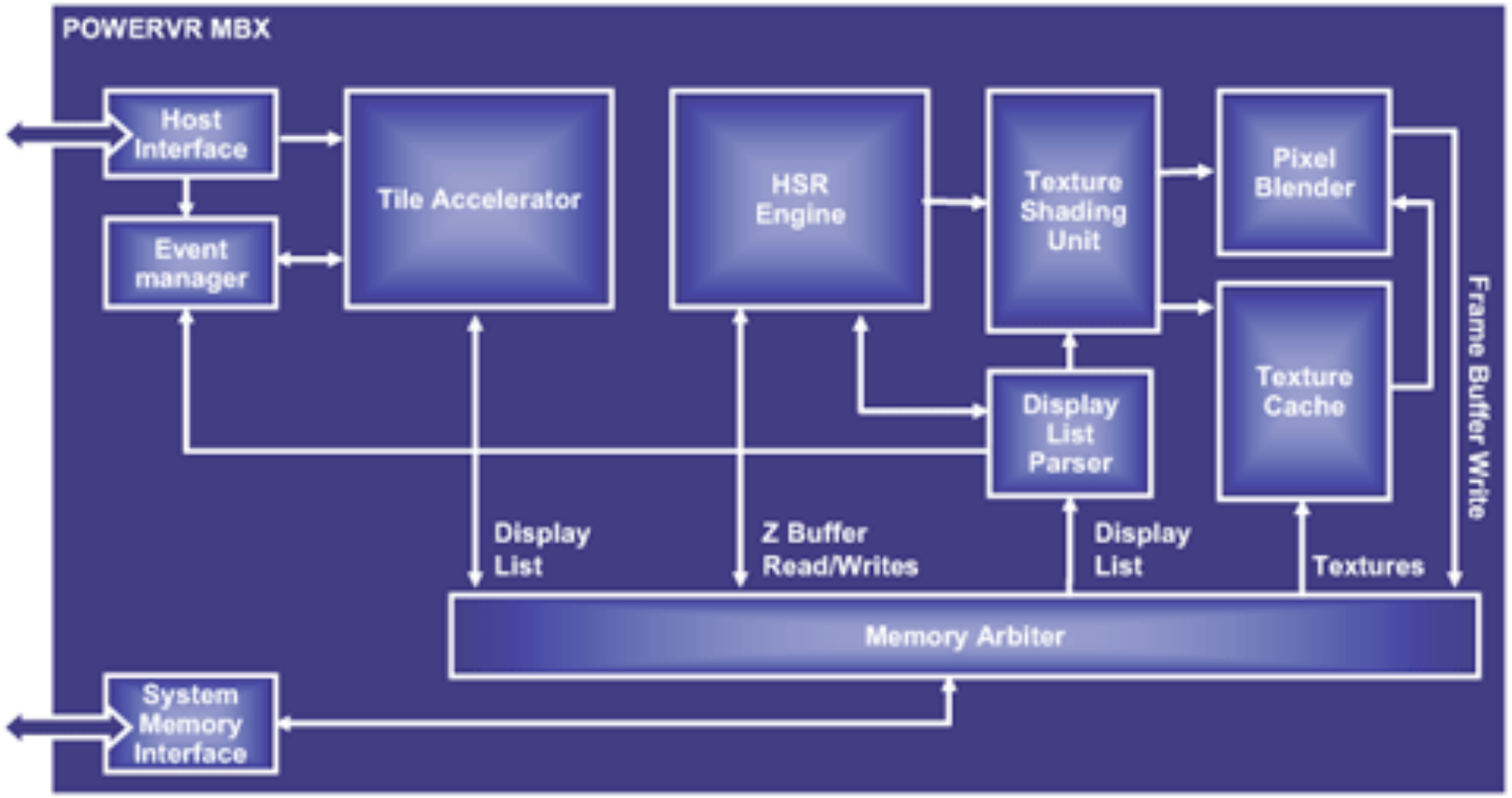
snappy touch

snappy
touch



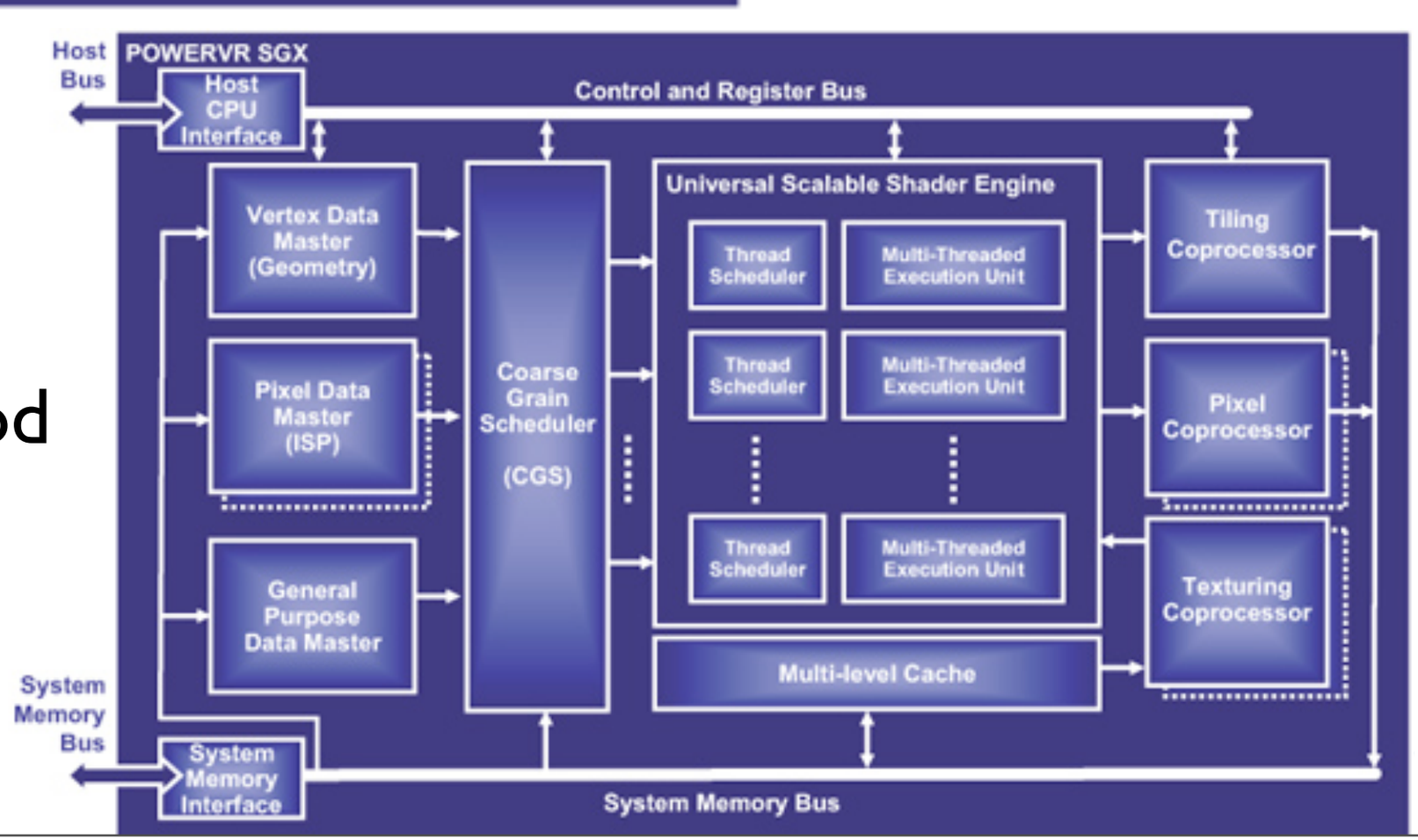
iPhone 2G
iPhone 3G
iPod Touch 1st, 2nd,
and some 3rd gen

snappy touch

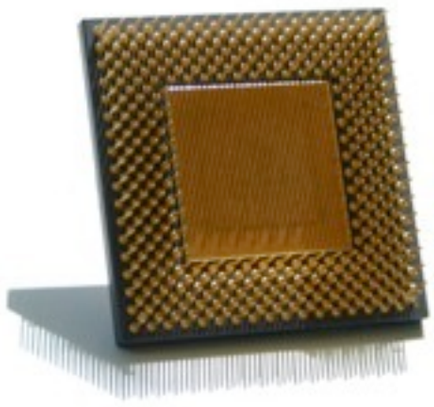


iPhone 2G
iPhone 3G
iPod Touch 1st, 2nd, and some 3rd gen

iPhone 3GS
High-end 3rd gen iPod Touch

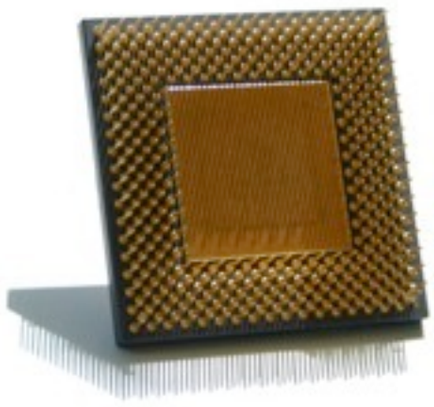


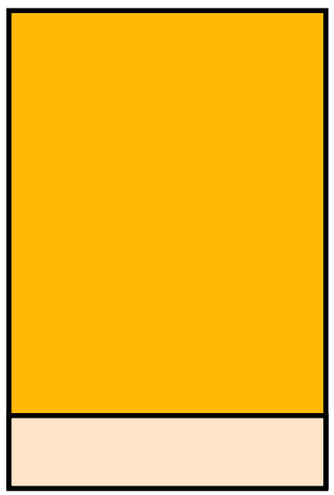
Game loop



No multiple cores, but we have a GPU
How to parallelize the two?

Game loop

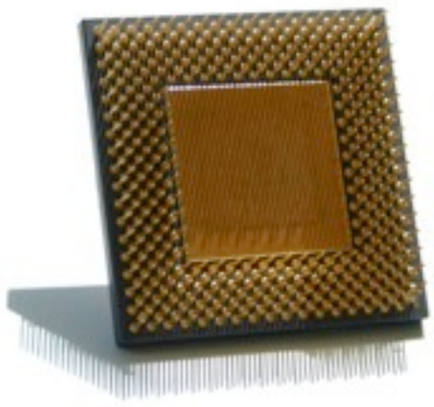


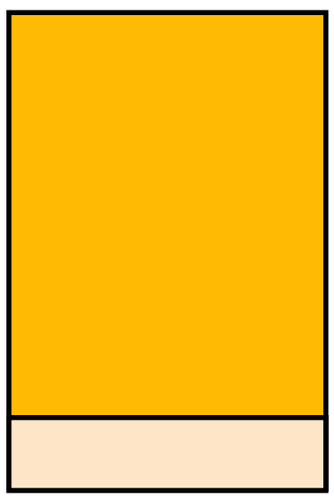


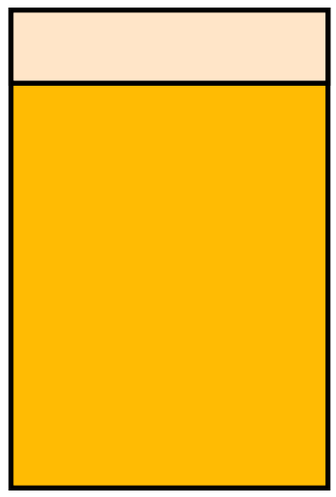
snappy touch

No multiple cores, but we have a GPU
How to parallelize the two?

Game loop

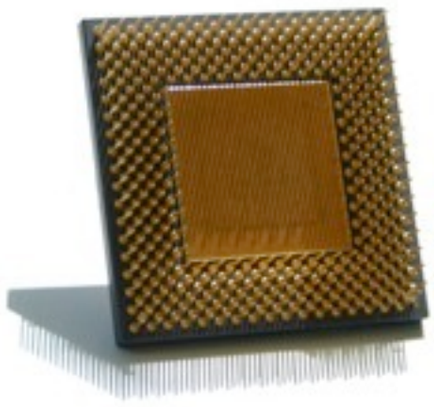


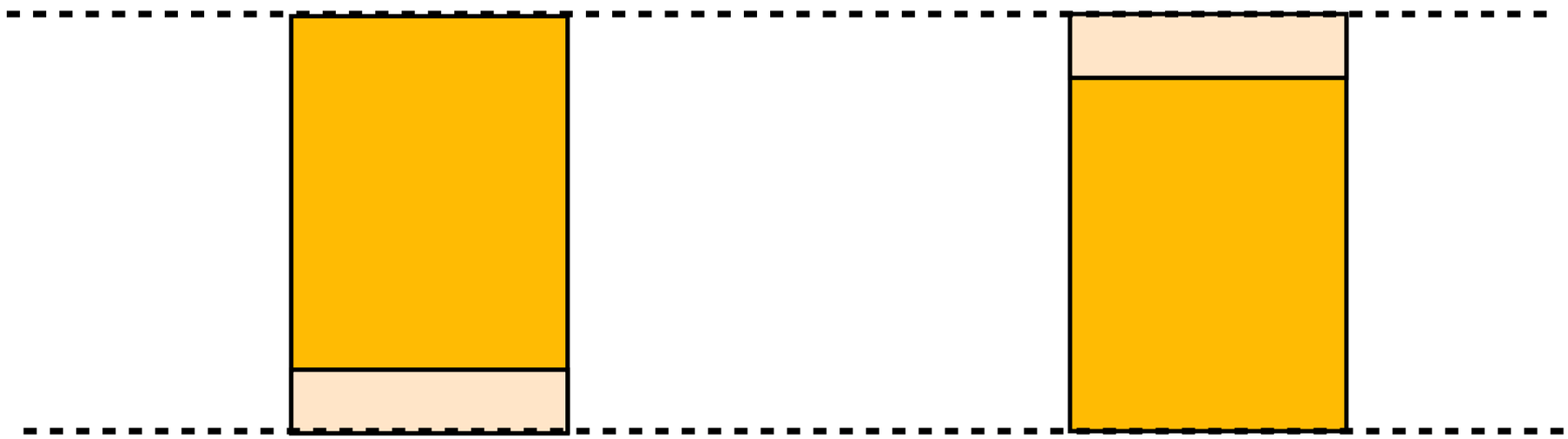




No multiple cores, but we have a GPU
How to parallelize the two?

Game loop

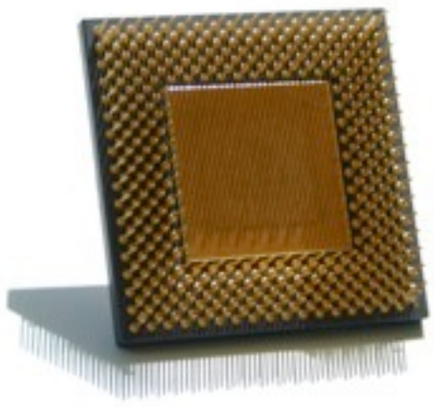


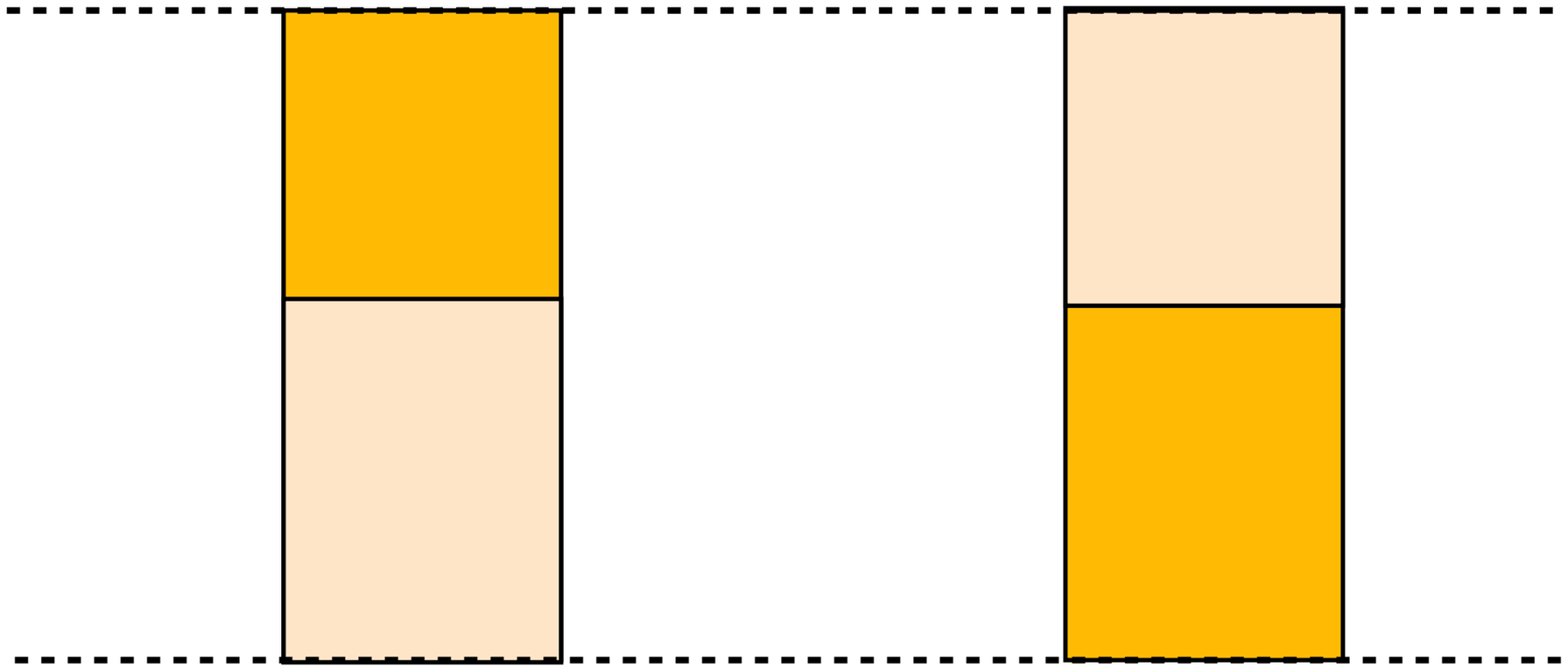


snappy touch

No multiple cores, but we have a GPU
How to parallelize the two?

Game loop

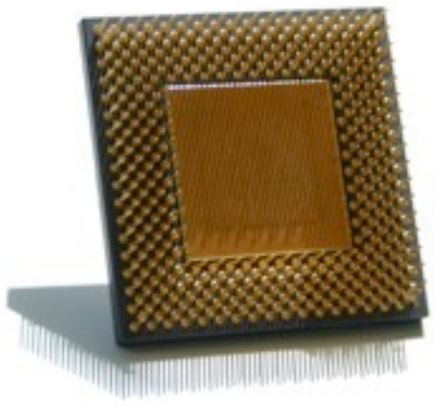




snappy touch

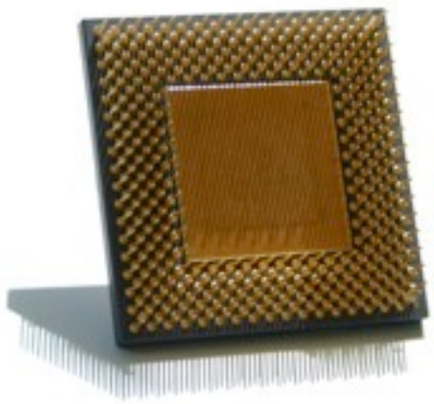
No multiple cores, but we have a GPU
How to parallelize the two?

Game loop



snappy touch

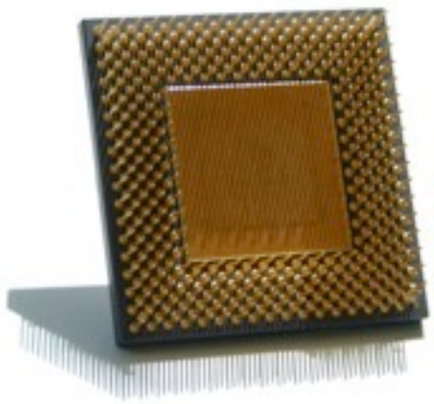
Game loop



- Avoid locking some resource while the GPU is rendering.

snappy touch

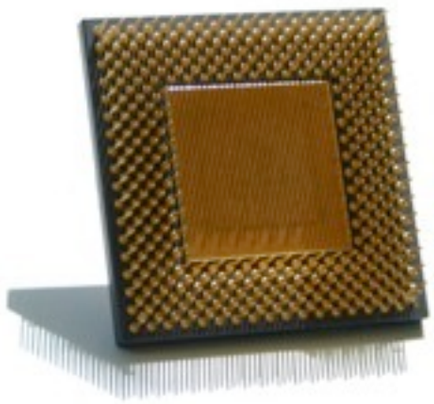
Game loop



- Avoid locking some resource while the GPU is rendering.
- Might get a small speed up by first presenting the frame, then updating game state, and finally rendering.

snappy touch

Game loop



- Avoid locking some resource while the GPU is rendering.
- Might get a small speed up by first presenting the frame, then updating game state, and finally rendering.
- Almost no difference in Flower Garden

snappy touch

Rendering

Rendering

- A lot of the usual advice for GPUs:

Rendering

- A lot of the usual advice for GPUs:
- Minimize draw primitive calls

snappy touch

Rendering

- A lot of the usual advice for GPUs:
- Minimize draw primitive calls
- Minimize state changes

snappy touch

Rendering

- A lot of the usual advice for GPUs:
- Minimize draw primitive calls
- Minimize state changes
- ...

snappy touch

Vertices

Vertices

- Rendering vertices can become a bottleneck

Vertices

- Rendering vertices can become a bottleneck
- No true Vertex Buffer Objects on the MBX, so CPU involved in every draw call

snappy touch

Vertices

- Rendering vertices can become a bottleneck
- No true Vertex Buffer Objects on the MBX, so CPU involved in every draw call
- Fixed on the 3GS though

snappy touch

Vertices

Need to set correct scaling/biasing matrix on the other end
Textures jumped around when made uvs a byte

Vertices

- Make vertices as small as possible

Need to set correct scaling/biasing matrix on the other end
Textures jumped around when made uvs a byte

Vertices

- Make vertices as small as possible

```
struct PetalVertex
{
#ifdef PETAL_SMALL_VERTEX
  int16_t x, y, z;
  int16_t u, v;
  int16_t u1, v1;
#else
  float x, y, z;
  float u, v;
  float u1, v1;
#endif
};
```

Need to set correct scaling/biasing matrix on the other end
Textures jumped around when made uvs a byte

Vertices

- Make vertices as small as possible

```
struct PetalVertex
{
#ifdef PETAL_SMALL_VERTEX
  int16_t x, y, z;
  int16_t u, v;
  int16_t u1, v1;
#else
  float x, y, z;
  float u, v;
  float u1, v1;
#endif
};
```

snappy touch

Need to set correct scaling/biasing matrix on the other end
Textures jumped around when made uvs a byte

Vertices

Vertices

- Align vertices on at least 4-byte boundaries

snappy touch

Vertices

- Align vertices on at least 4-byte boundaries
- Experiment with larger alignments

snappy touch

Vertices

You get the best performance without the degenerate verts of a strip

Vertices

- Use indexed lists...

You get the best performance without the degenerate verts of a strip

Vertices

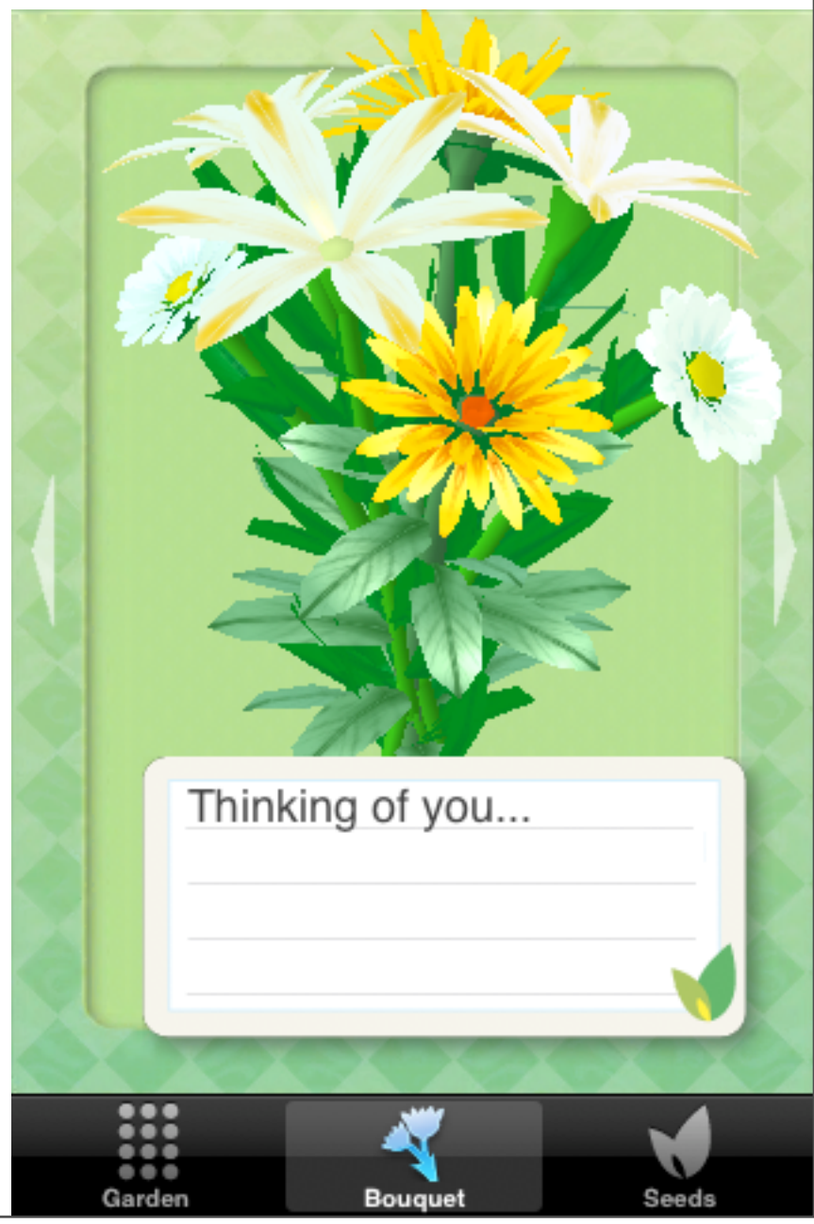
- Use indexed lists...
- ... ordered as if they were strips.

You get the best performance without the degenerate verts of a strip

Mixing OpenGL and UIKit

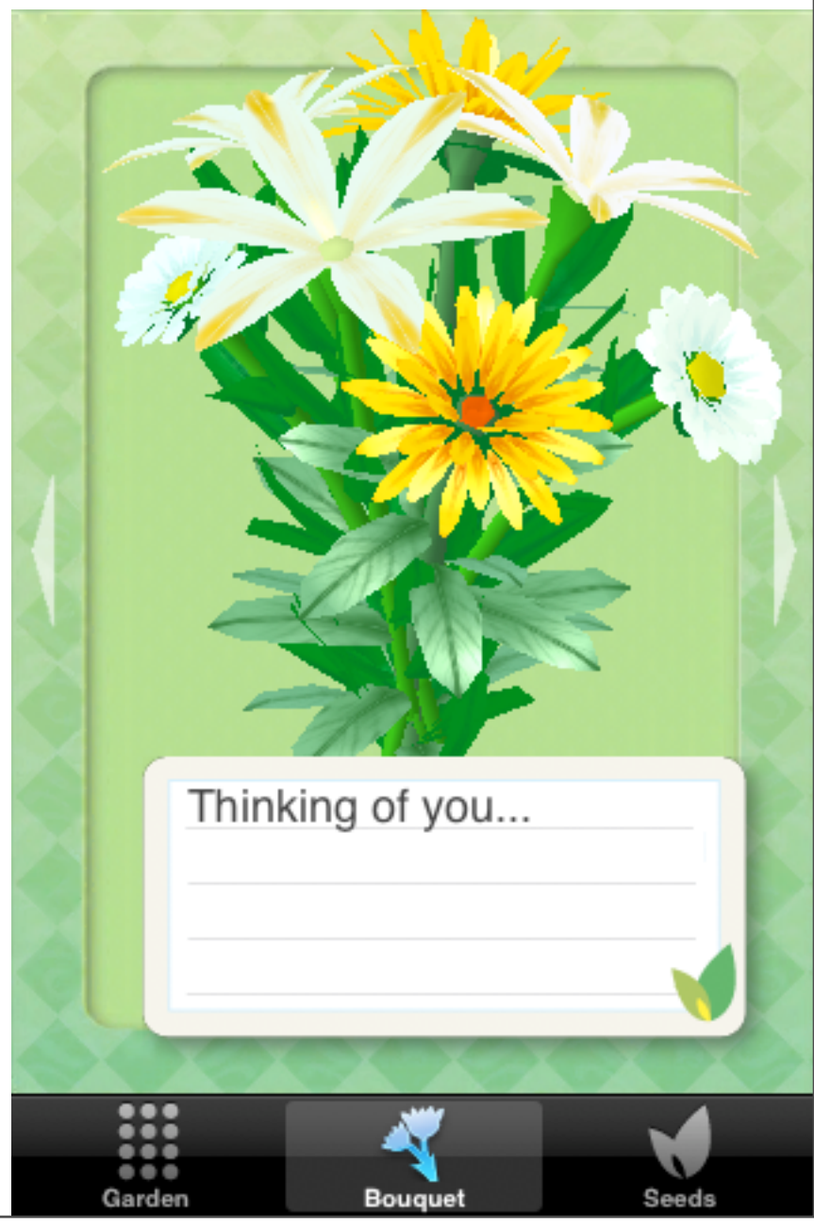
Mixing OpenGL and UIKit

- You can mix and match OpenGL and UIKit



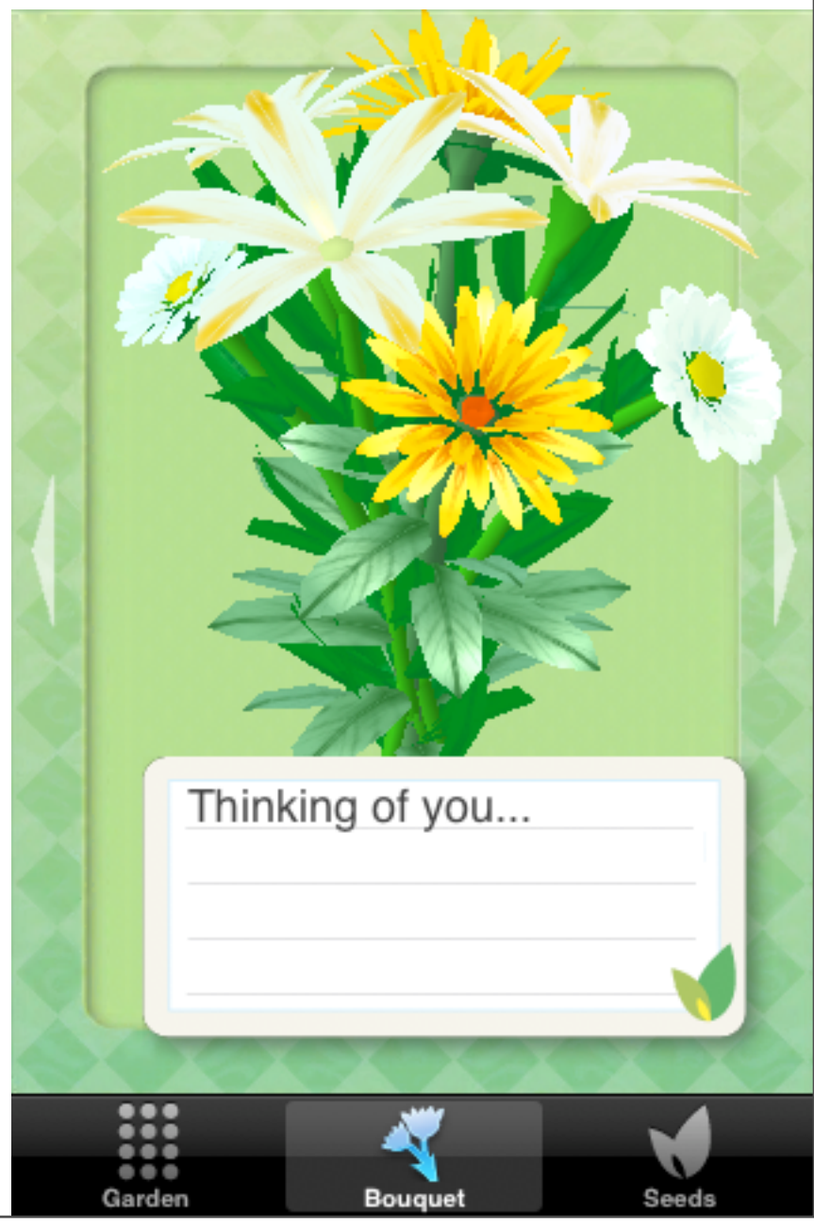
Mixing OpenGL and UIKit

- You can mix and match OpenGL and UIKit
- But you have to be very careful



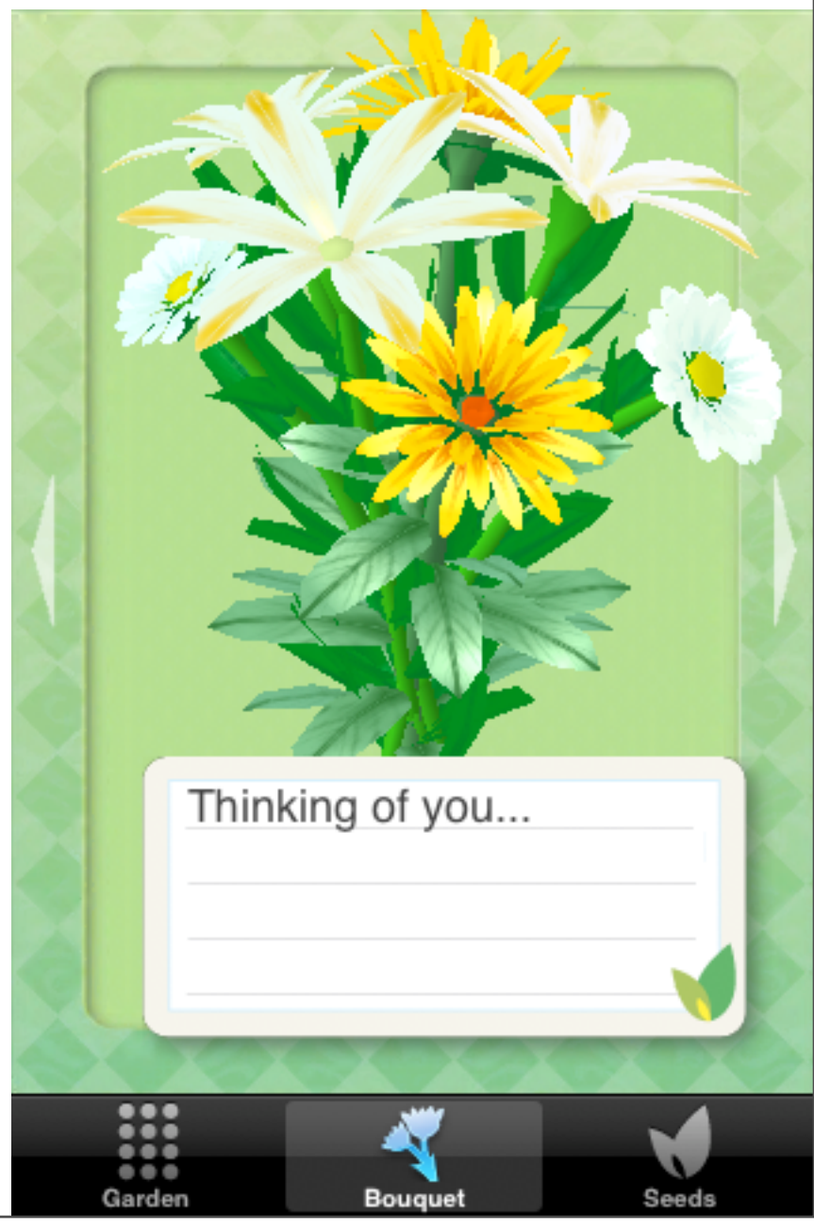
Mixing OpenGL and UIKit

- You can mix and match OpenGL and UIKit
- But you have to be very careful
- Minimize UIKit elements on top of EAGLView



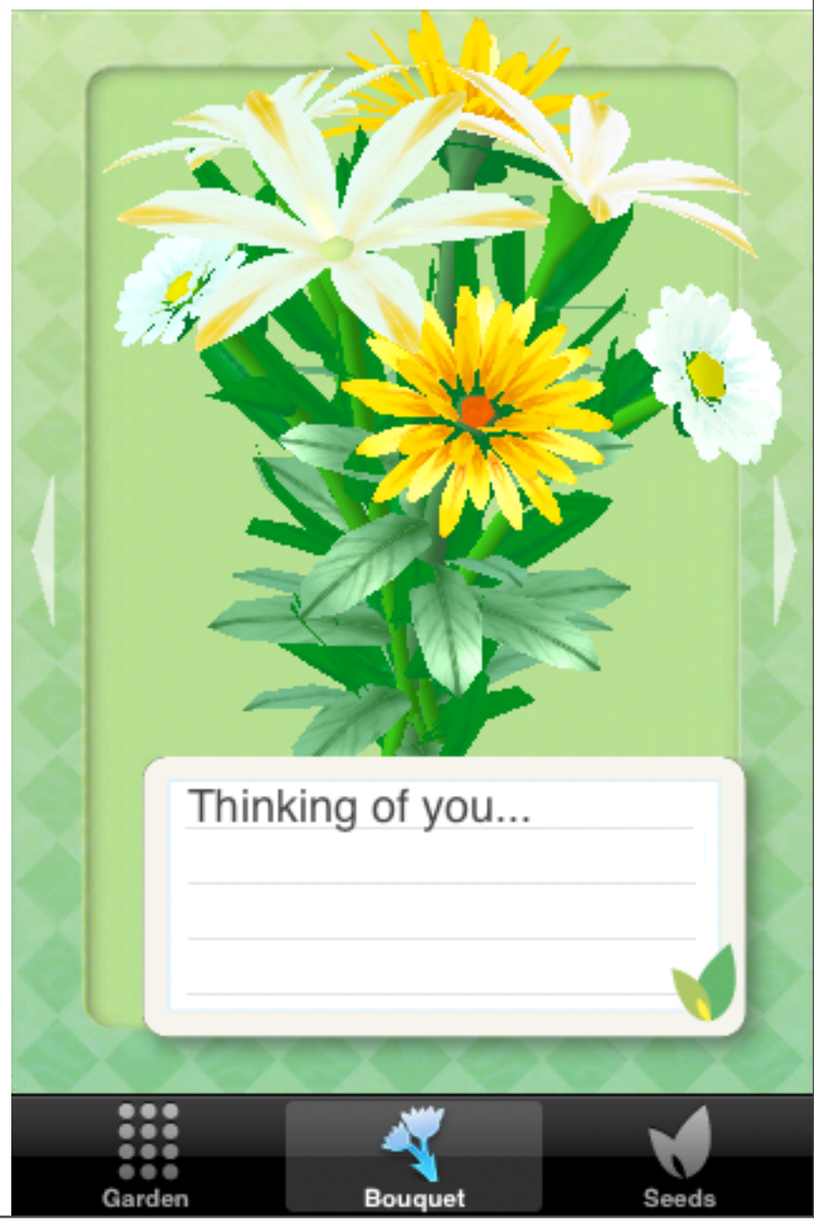
Mixing OpenGL and UIKit

- You can mix and match OpenGL and UIKit
- But you have to be very careful
- Minimize UIKit elements on top of EAGLView
- Keep them opaque



Mixing OpenGL and UIKit

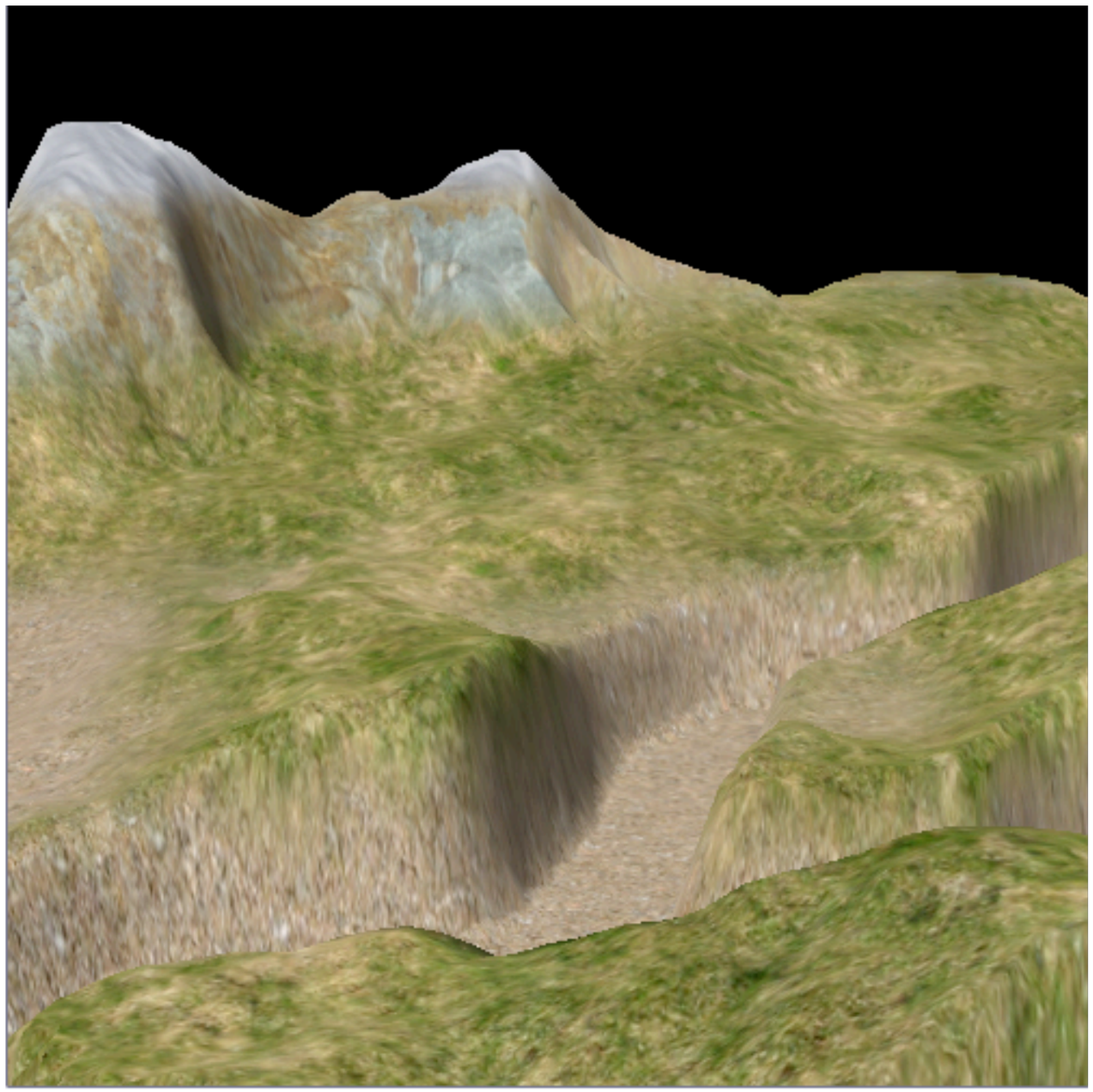
- You can mix and match OpenGL and UIKit
- But you have to be very careful
- Minimize UIKit elements on top of EAGLView
- Keep them opaque
- OK to have smaller EAGLView



Use iPhone Features

Use multitexturing (2 texture combiners)
Point sprites

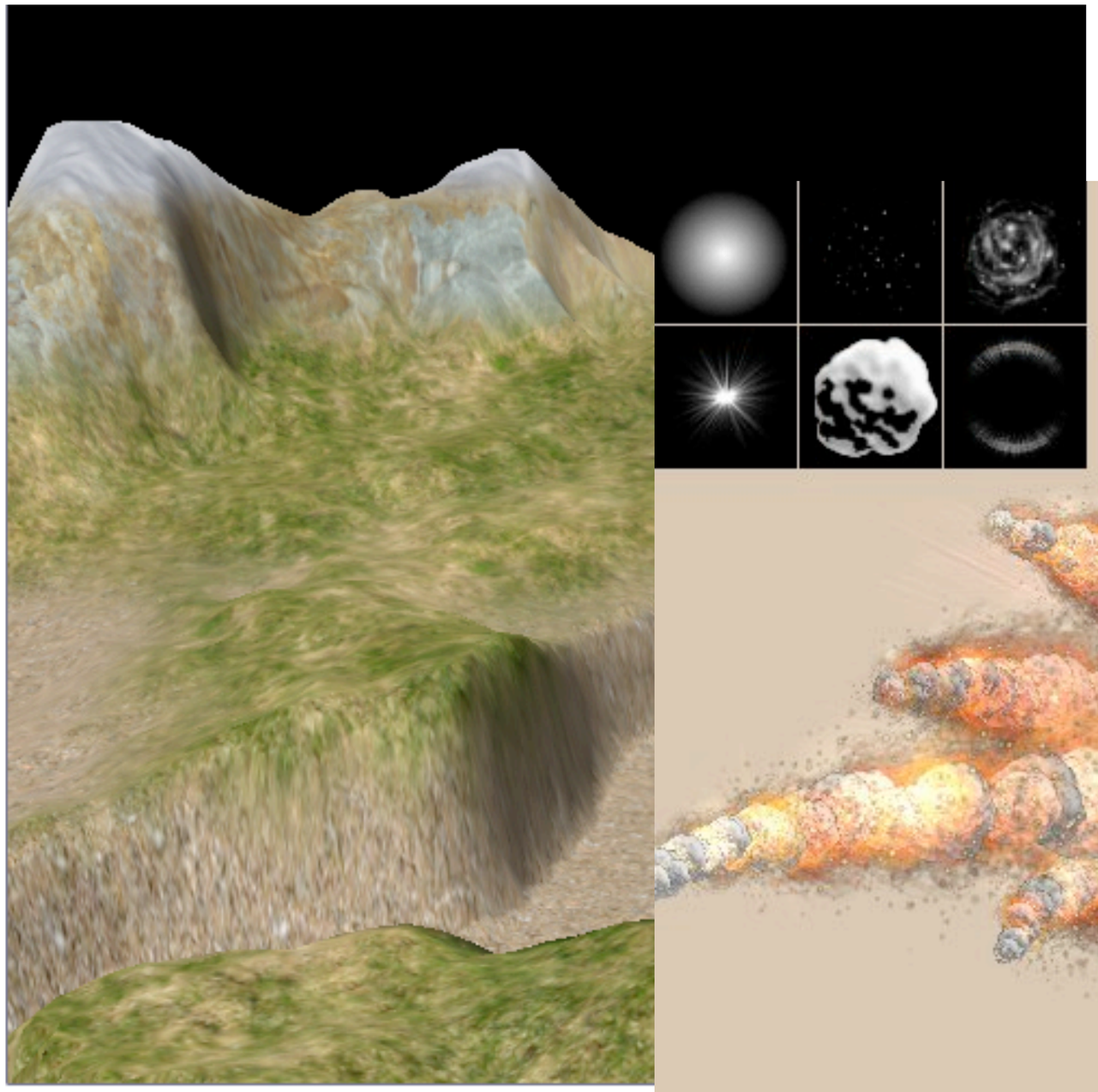
Use iPhone Features

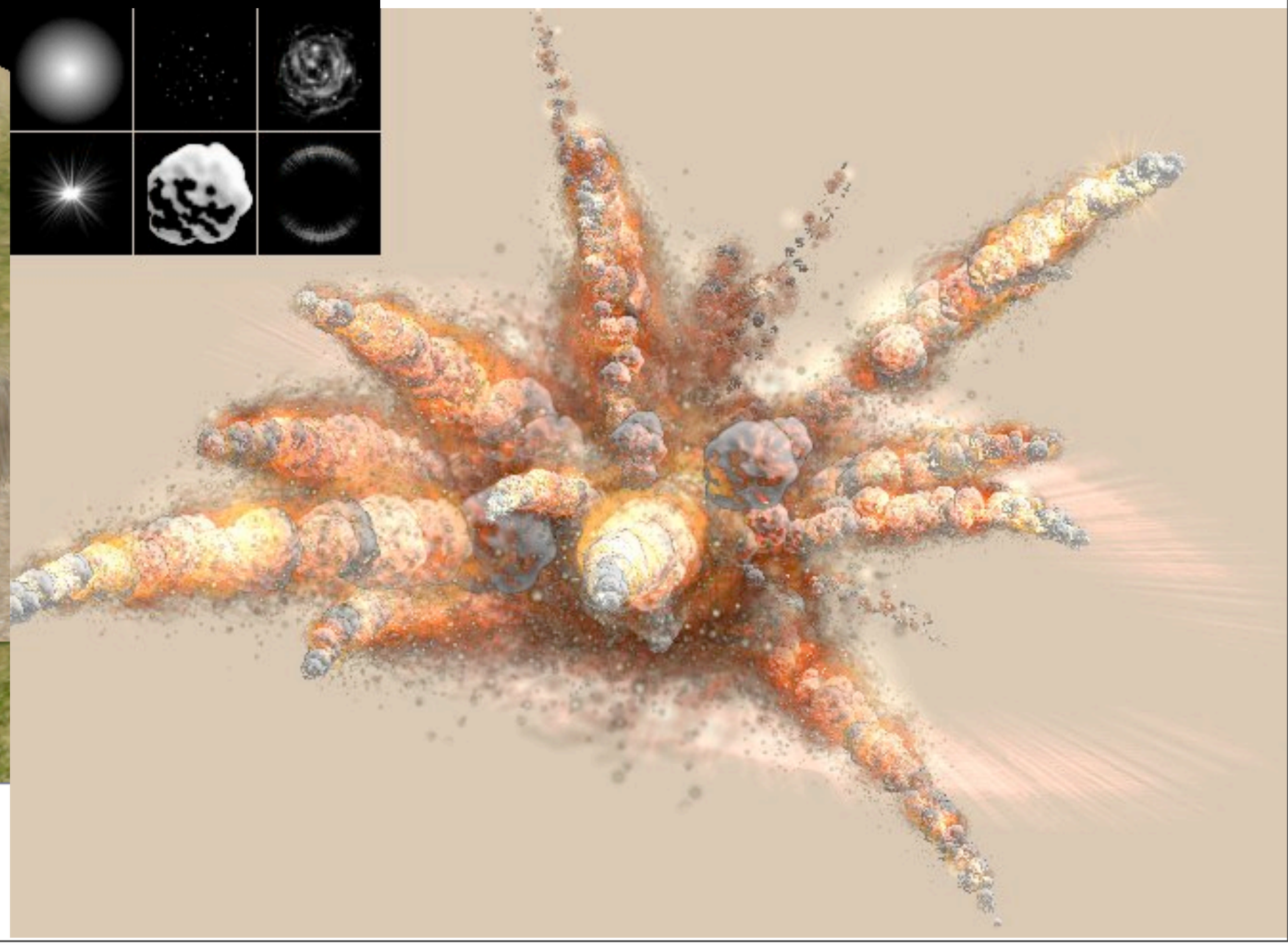


snappy touch

Use multitexturing (2 texture combiners)
Point sprites

Use iPhone Features





Use multitexturing (2 texture combiners)
Point sprites

Memory

This makes it clear that the iPhone is NOT a game console (unfortunately)

snappy touch

This makes it clear that the iPhone is NOT a game console (unfortunately)

!=

snappy touch

This makes it clear that the iPhone is NOT a game console (unfortunately)

!=

snappy touch

This makes it clear that the iPhone is NOT a game console (unfortunately)

Memory rant

Apple wants you to play nice and work with any amount of memory
Game developers want to count on the available memory and use it all
Clash!

Memory rant

- No memory guarantees

Apple wants you to play nice and work with any amount of memory
Game developers want to count on the available memory and use it all
Clash!

Memory rant

- No memory guarantees
- No idea of how much memory will be available

Apple wants you to play nice and work with any amount of memory
Game developers want to count on the available memory and use it all
Clash!

Memory rant

- No memory guarantees
- No idea of how much memory will be available
- Not even how much memory at startup!

Apple wants you to play nice and work with any amount of memory
Game developers want to count on the available memory and use it all
Clash!

Memory rant

- No memory guarantees
- No idea of how much memory will be available
- Not even how much memory at startup!
- Sometimes I've seen the game start with as low as 3MB free memory!!!

Apple wants you to play nice and work with any amount of memory
Game developers want to count on the available memory and use it all
Clash!

Memory

Memory

- System notifies apps when memory is running low

Memory

- System notifies apps when memory is running low
- Apps are supposed to free as much memory as they can

Memory

- System notifies apps when memory is running low
- Apps are supposed to free as much memory as they can
- Horrible situation for games!

snappy touch

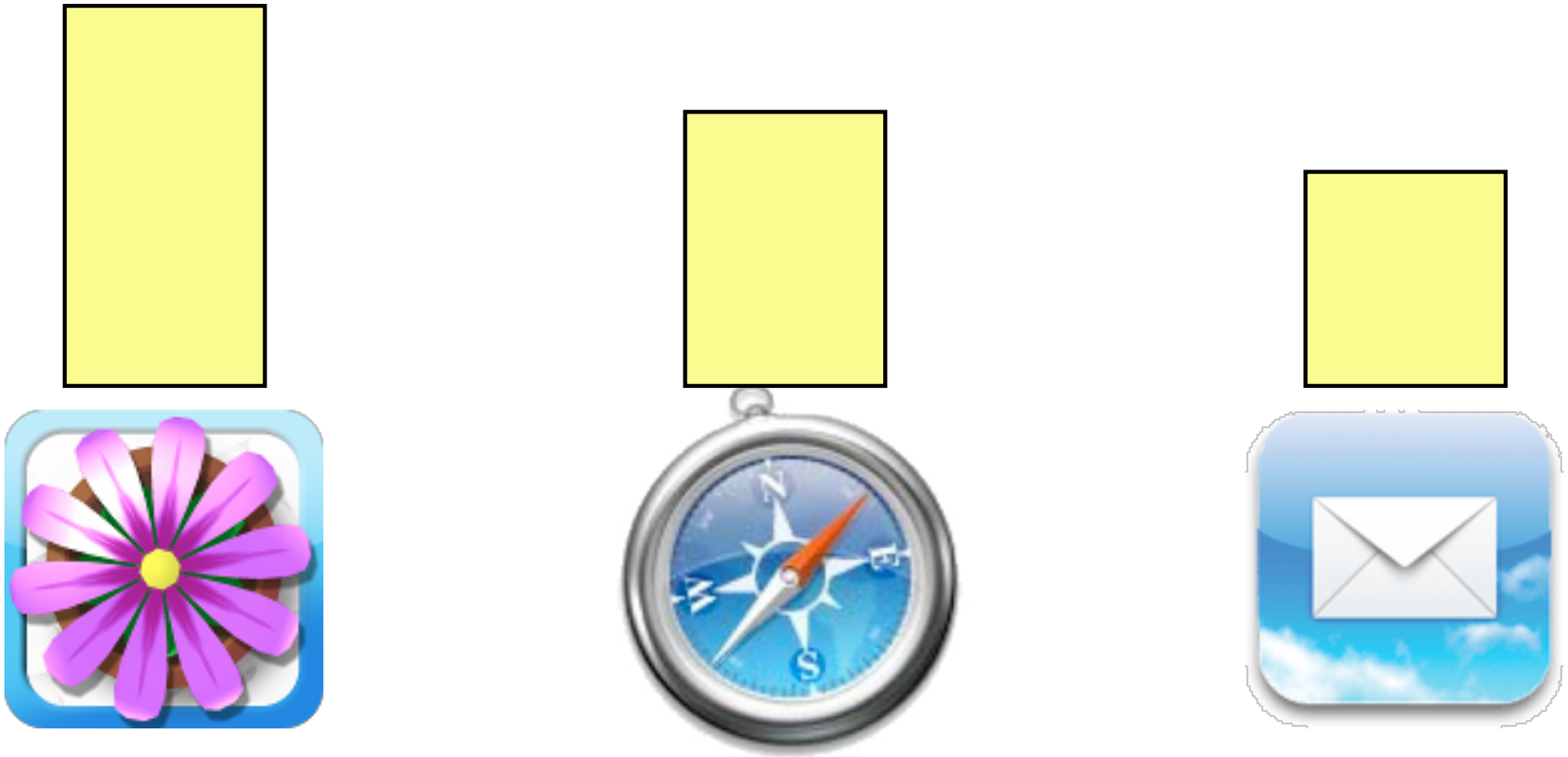
Memory

snappy touch

Memory

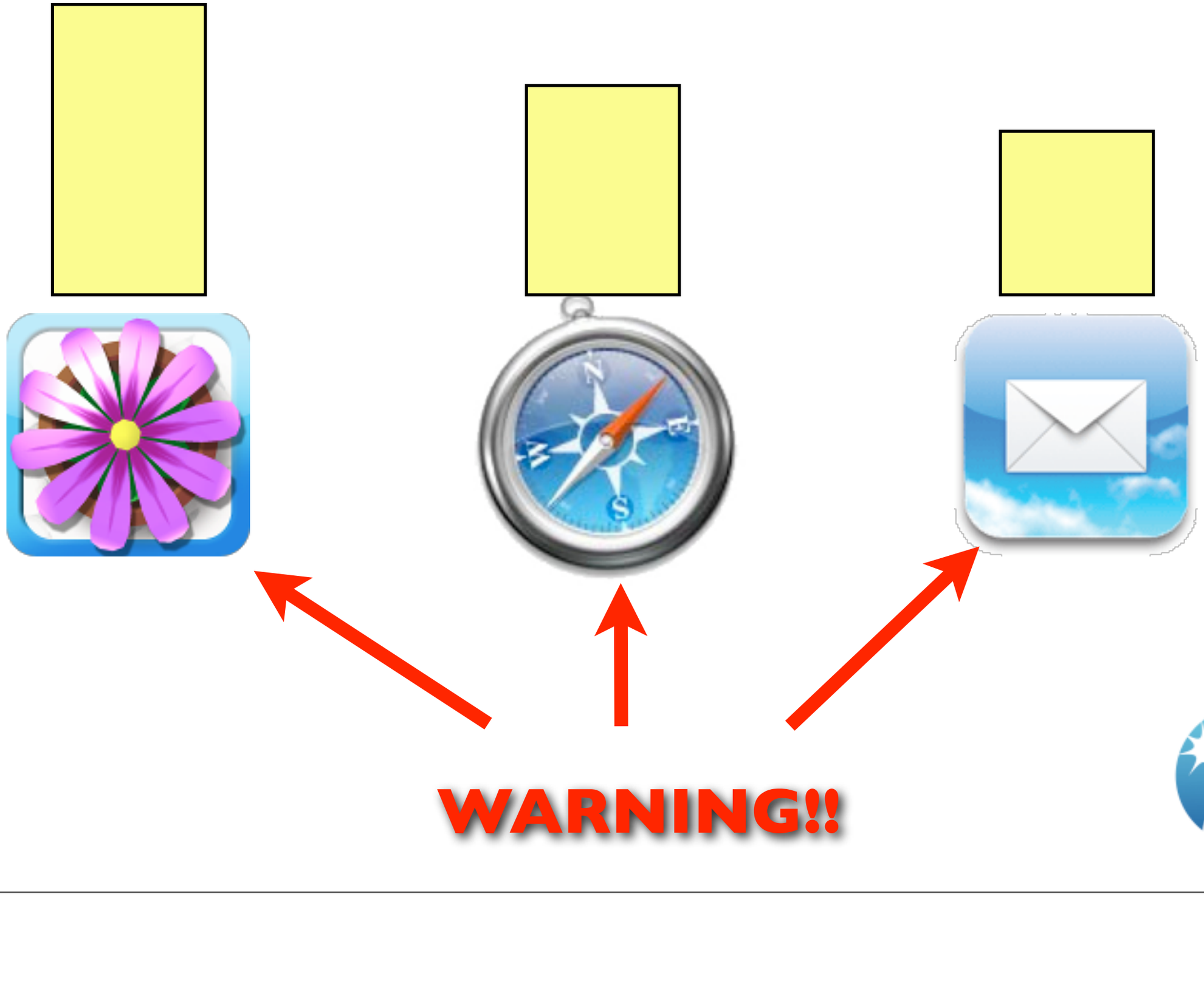
snappy touch

Memory



snappy touch

Memory

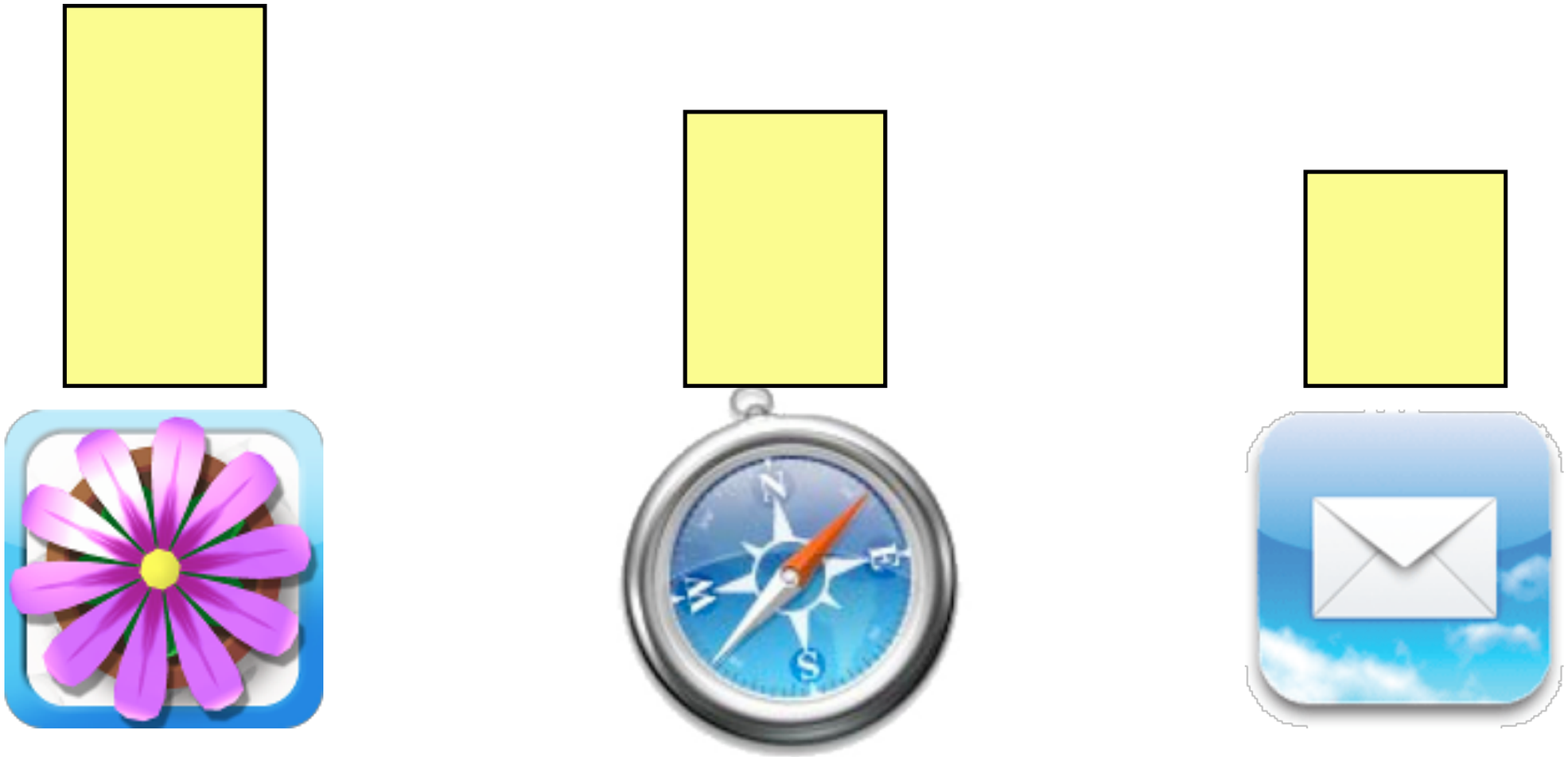


snappy touch

Memory

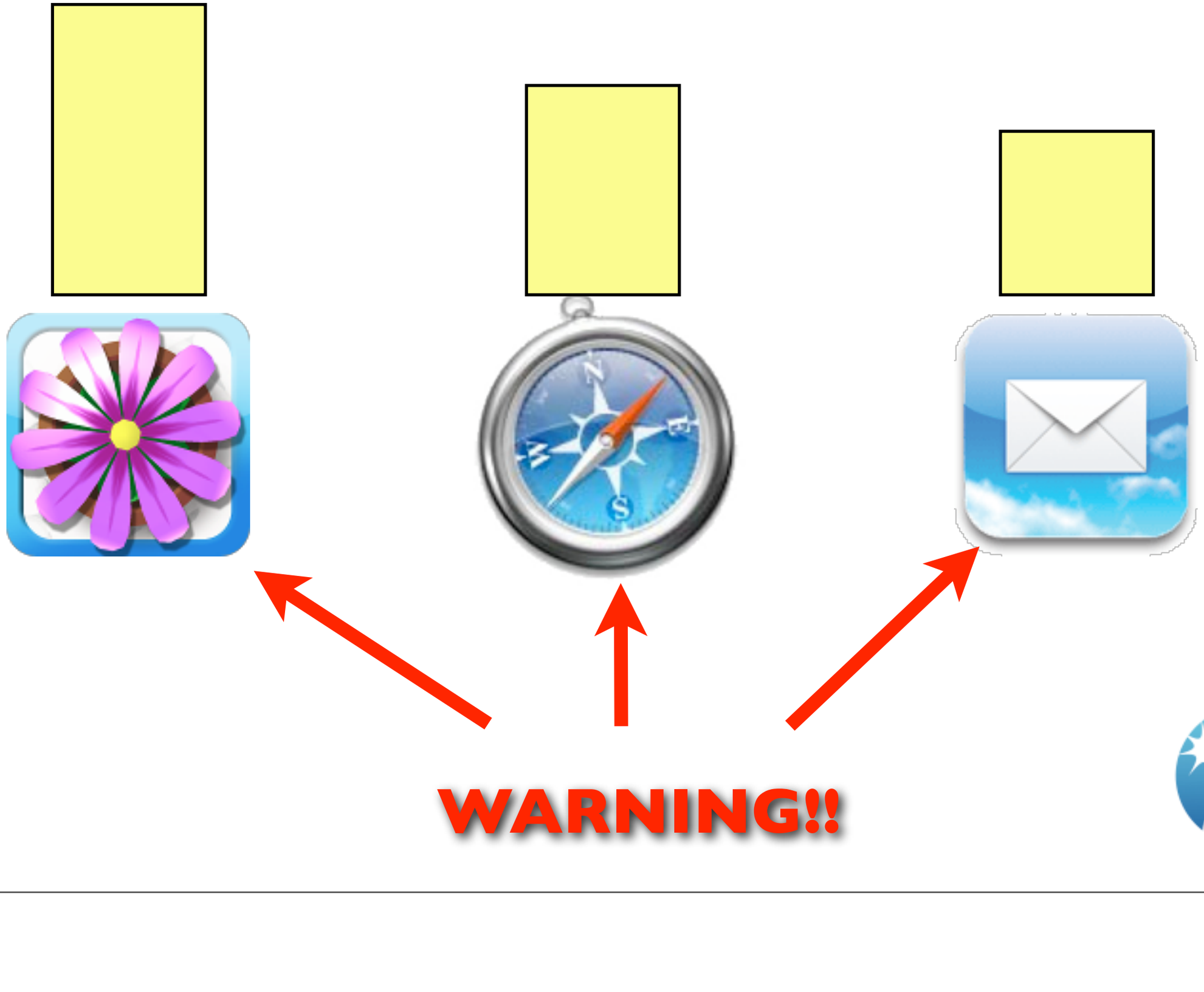
snappy touch

Memory



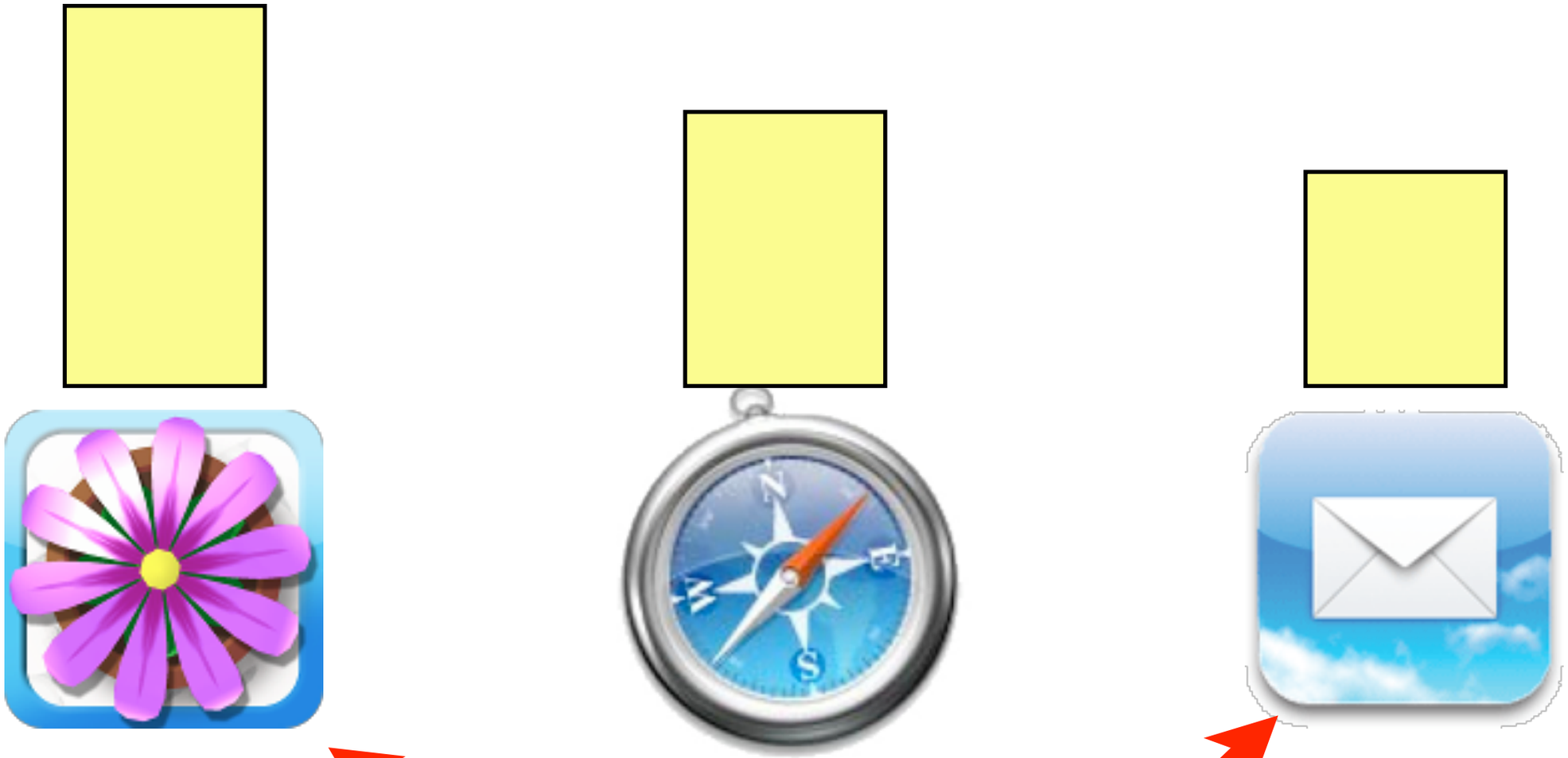
snappy touch

Memory



snappy touch

Memory

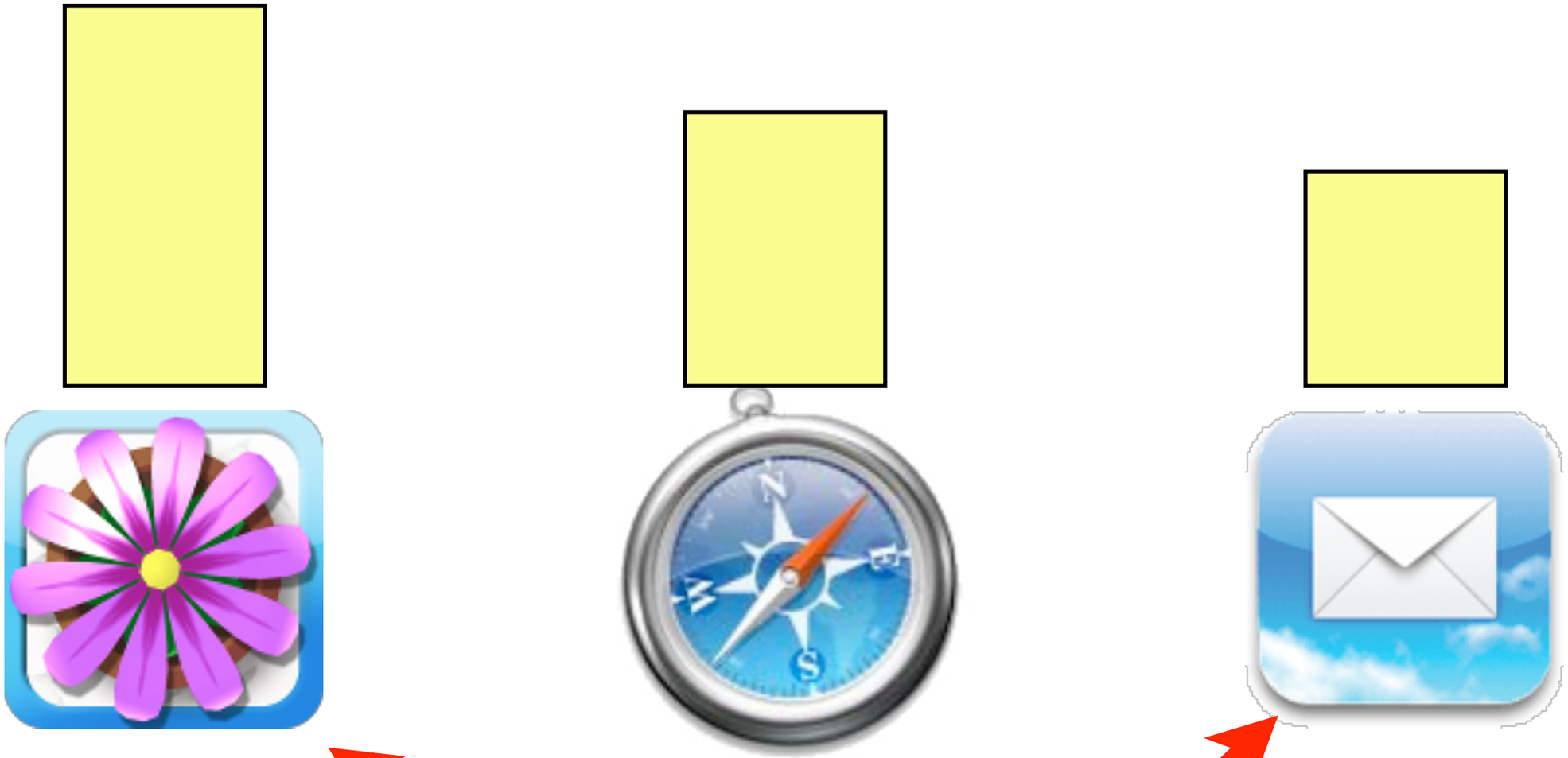


Wait...

WARNING!!

snappy touch

Memory



Wait...

... wait for it ...

WARNING!!

snappy touch

Memory

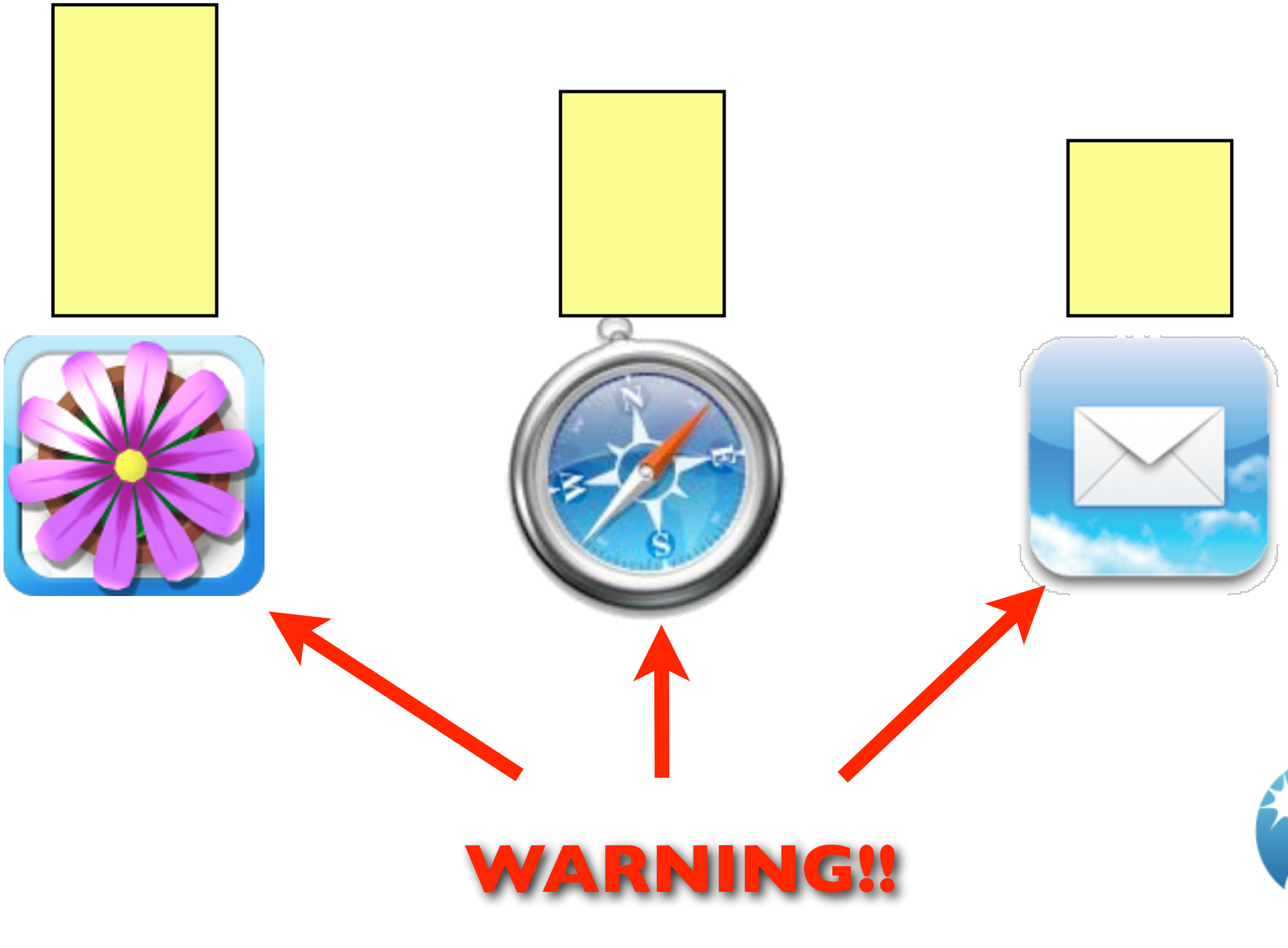
Wait...

... wait for it ...

WARNING!!

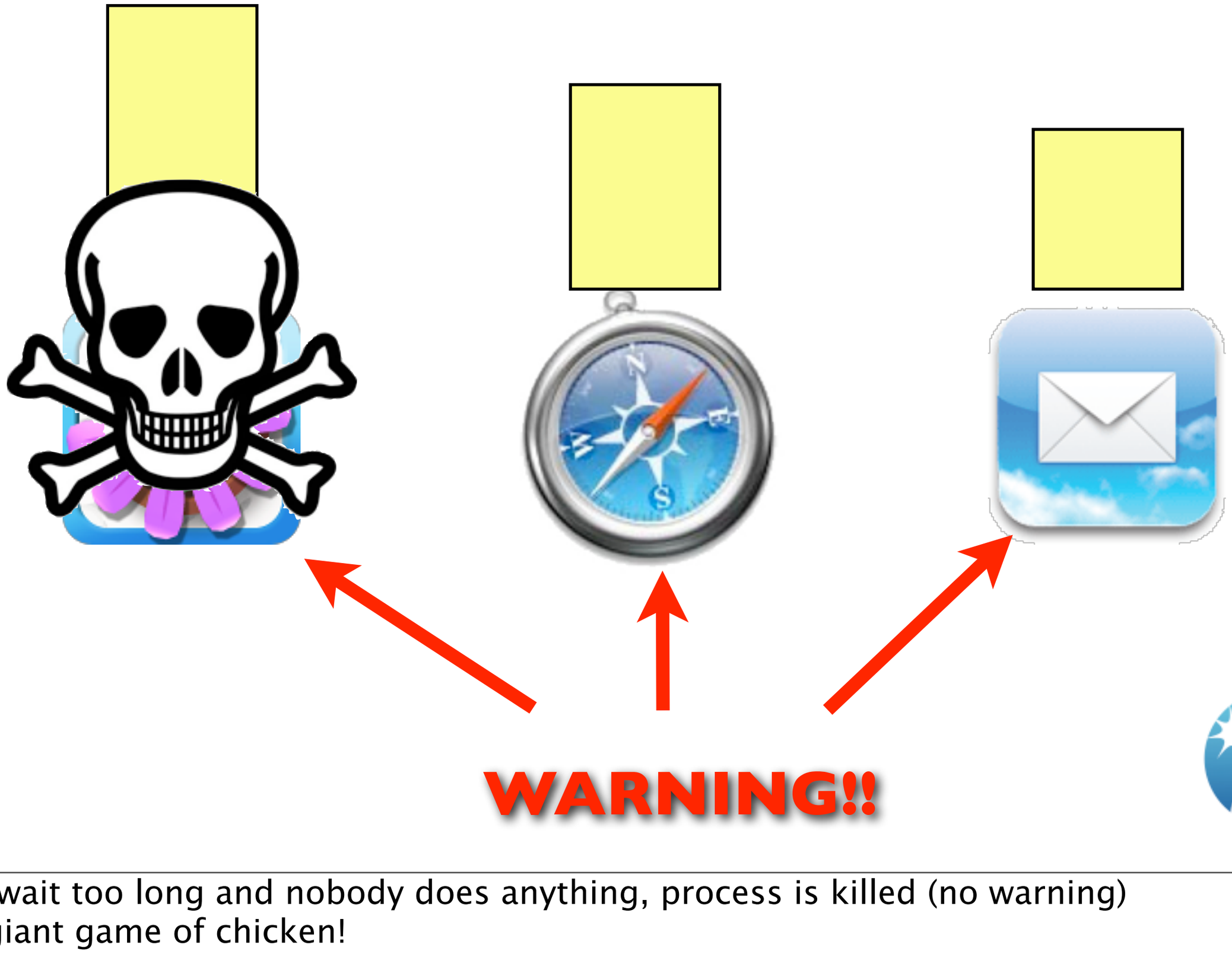
snappy touch

Memory



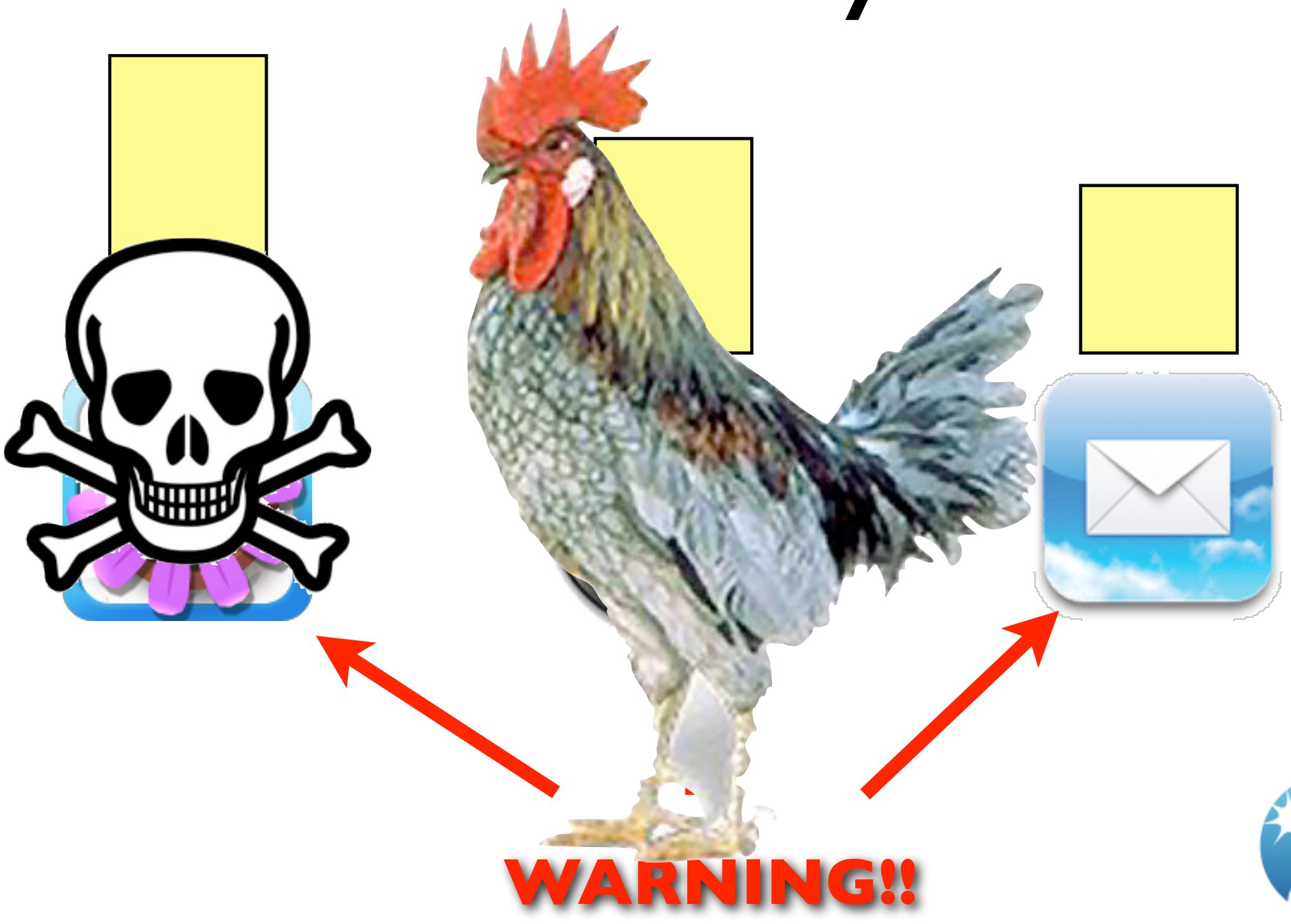
If you wait too long and nobody does anything, process is killed (no warning)
It’s a giant game of chicken!

Memory



If you wait too long and nobody does anything, process is killed (no warning)
It’s a giant game of chicken!

Memory



snappy touch

If you wait too long and nobody does anything, process is killed (no warning)
It’s a giant game of chicken!

Memory trick

Be extremely careful with that! Potential for crashes in some phones if not enough memory.

Memory trick

- Do you prefer to allocate your memory all at once at initialization time?

Be extremely careful with that! Potential for crashes in some phones if not enough memory.

Memory trick

- Do you prefer to allocate your memory all at once at initialization time?
- Allocate memory slowly.

Be extremely careful with that! Potential for crashes in some phones if not enough memory.

Memory trick

- Do you prefer to allocate your memory all at once at initialization time?
- Allocate memory slowly.
- When you get a warning, wait a little.

Be extremely careful with that! Potential for crashes in some phones if not enough memory.

Memory trick

- Do you prefer to allocate your memory all at once at initialization time?
- Allocate memory slowly.
- When you get a warning, wait a little.
- If free memory doesn't increase, release some.

Be extremely careful with that! Potential for crashes in some phones if not enough memory.

Memory trick

- Do you prefer to allocate your memory all at once at initialization time?
- Allocate memory slowly.
- When you get a warning, wait a little.
- If free memory doesn't increase, release some.
- Repeat until enough memory

Be extremely careful with that! Potential for crashes in some phones if not enough memory.

Memory trick

- Do you prefer to allocate your memory all at once at initia
- Allocate memo
- When you get e.
- If free memory ease some.
- Repeat until enough memory

snappy touch

Be extremely careful with that! Potential for crashes in some phones if not enough memory.

Thank you!

Twitter: @snappytouch

Email: noel@snappytouch.com

Blog: http://gamesfromwithin.com