Performance improvements in cocos2d-x v3.0: Lessons learned

Ricardo Quesada

The Architecture for the Digital World®

Who am I?

- Ricardo Quesada
- Co-founder and main author of cocos2d
- Chief Architect at Chukong

What is Cocos2d-x?

ARM

Cocos2d-x

- 2D Game engine + 3D extensions
- Fast and robust
- Easy to use
  - Familiar API
- Open Source
  - MIT License
- Multiplatform
  - Mobile, desktop and web

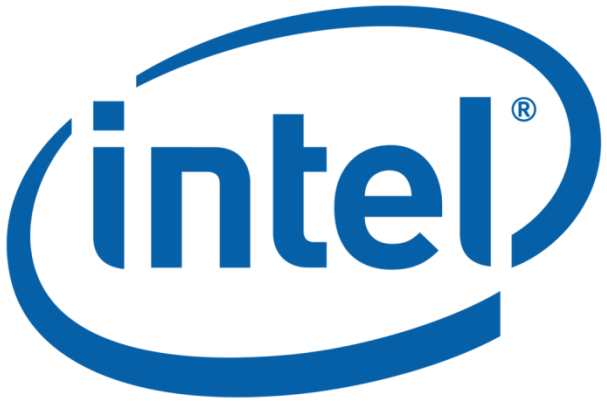
Cocos2d-x Basic Features

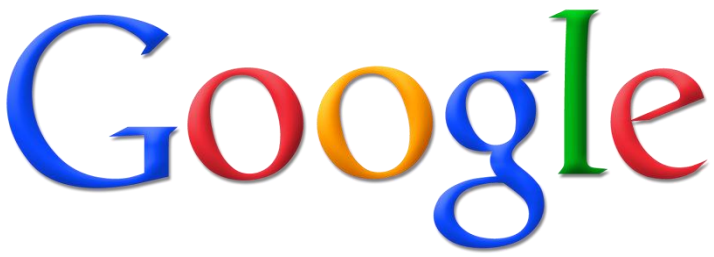
- Workflow
- Scene graph
- Sprites
- Particles
- Labels
- Tile Maps
- Parallax Scrolling
- Actions
- Mesh effects
- Physics integration
- 3D extensions
- Animation support

Who is Using Cocos2d-x ?

- Many companies
  - From big companies like Zynga
  - To long-tail / indie companies
- Used by thousands mobile games
  - Many Top #10 games on AppStore and PlayStore
  - 70% of Chinese games are using Cocos2d-x

Partners



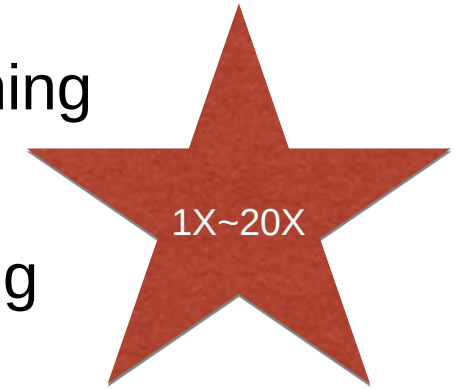


Performance Improvements: Renderer

ARM

Renderer: Performance improvements

- Auto Batching
- Auto Culling
- Caching Transform

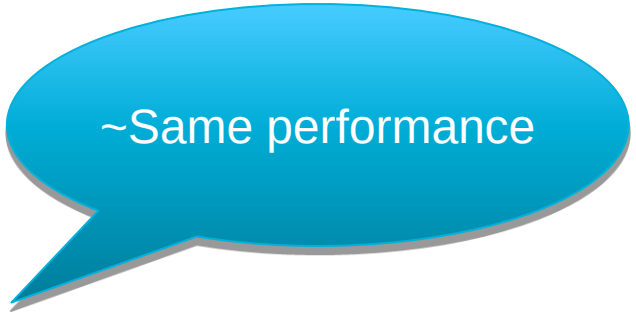


Auto Batching

- v3.0-beta
  - Fast for sprites
  - We profiled GPU. It was very fast.
  - We did basic CPU profiling. It was fast.
  - But some users were reporting it was very slow
- We did more CPU profiling
  - We were copying the Quads 2 times
  - For big Tile Maps it was 5x slower
- Basically our tests suit was incomplete

Auto Culling

- Different strategies:
  - Frustum culling
  - Bounding box testing
  - No culling



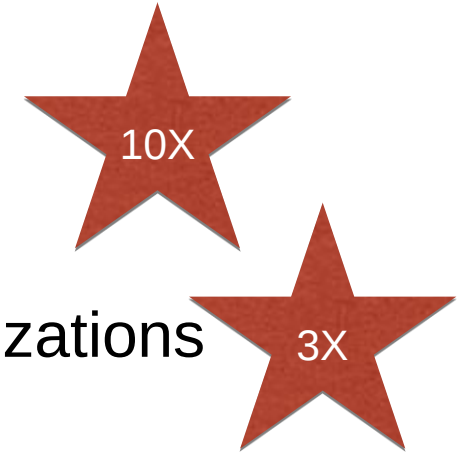
- We realized:
  - We weren’t doing the correct tests
  - Our culling algorithm was not fast enough

Performance Improvements: visit()

ARM

Visit(): Performance improvements

- Profiled key parts of the code
  - Node::sortAllChildren(): std::sort()
  - Node::visit(): caching + other optimizations



Performance improvements: Other examples

ARM

GL_TRIANGLES vs GL_TRIANGLE_STRIP

- A few years ago Apple recommended GL_TRIANGLE_STRIP
  - But in cocos2d, GL_TRIANGLES was much faster
- Certain best practices might not be valid for your game
  - Or not valid for a certain GPU / OpenGL driver
- Profile everything…
  - …including recommend best practices

ARM

Xcode® 5.0 vs 5.1beta5

- Xcode 5.0 was about 5% faster than v5.1 beta5
  - Fixed in final v5.1

- Profile everything in the same environment
  - Same device
  - Same operating system version
  - Same toolchain

Performance Improvements: The ones that didn't work

ARM

Optimizing vec3 * mat4

- (vec4 * mat4) is super fast with Neon instructions
- (vec3 * mat4) is slower

- Tried:
  - Convert vec3 to vec4
  - vec4 * mat4 using Neon
  - Convert output from vec4 to vec3

- Result: a bit slower than vec3 * mat4 in C due to the conversion to/from vec3/vec4

Improvements for v3.1
Work in progress

ARM

Trying vec3[] * mat4

- (vec3 * mat4) good speed using Neon
- (vec3[] * mat4) much faster using Neon

- Unfortunately cocos2d-x can't use it because it uses an interleaved array:
  - Array = [vec3,vec2,vec4; vec3,vec2,vec4; vec3,vec2,vec4]

- We are working with ARM on an interleaved vec3[] * mat4

Auto culling in Tile Maps

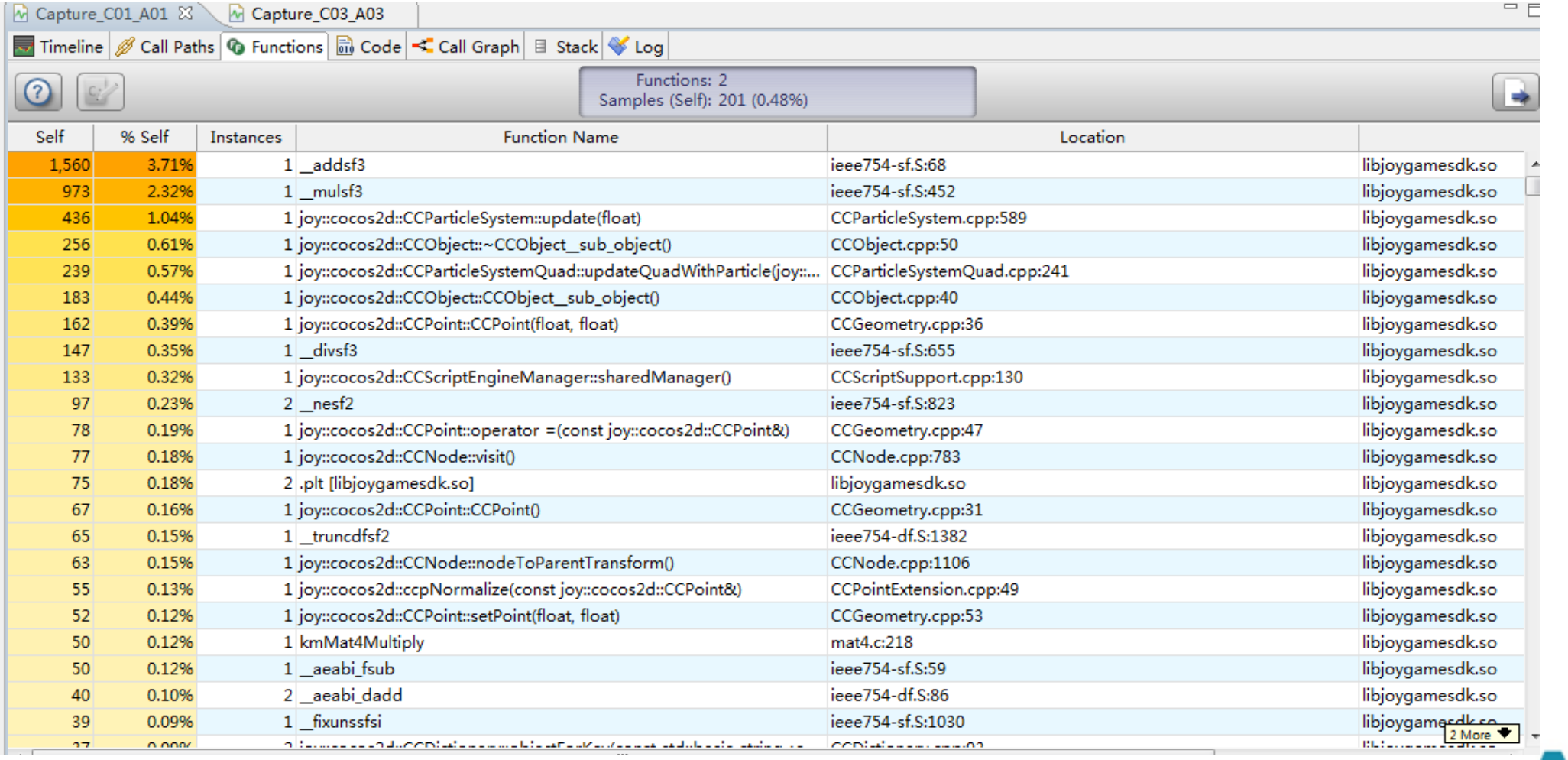
- v3.0:
  - Sends the whole map all the time
  - Super fast for small maps
  - Super slow for big maps
- v3.1:
  - Only sends visible tiles
  - Fast for both small and big maps
  - Prototype working:
    - 10x faster for big maps
    - about 5% ~ 10% slower for small maps on old devices

Profiling Tools Used for Cocos2d-x v2.2 and v3.0

ARM

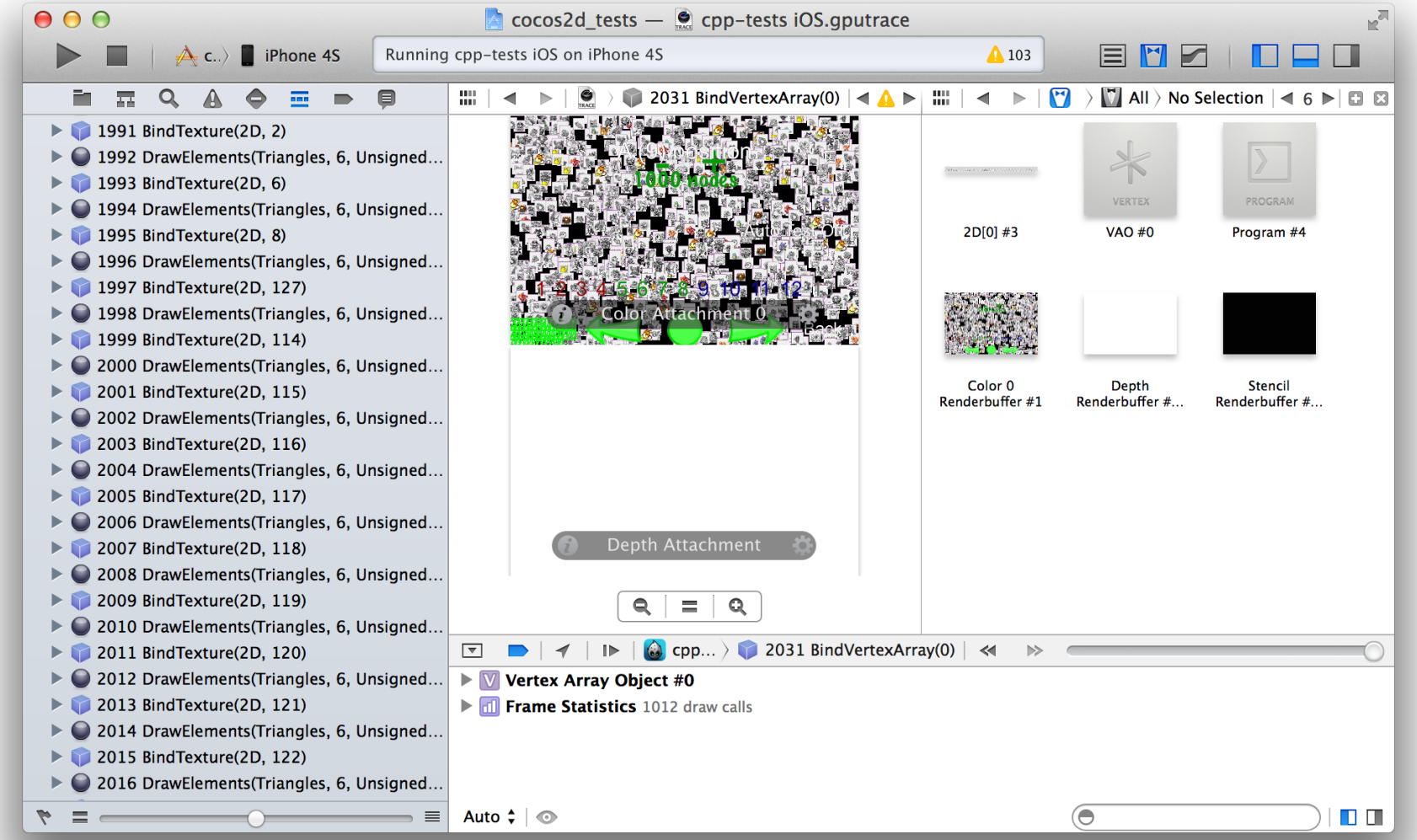
Android™ OS: ARM® DS-5 Toolchain

CPU profiling

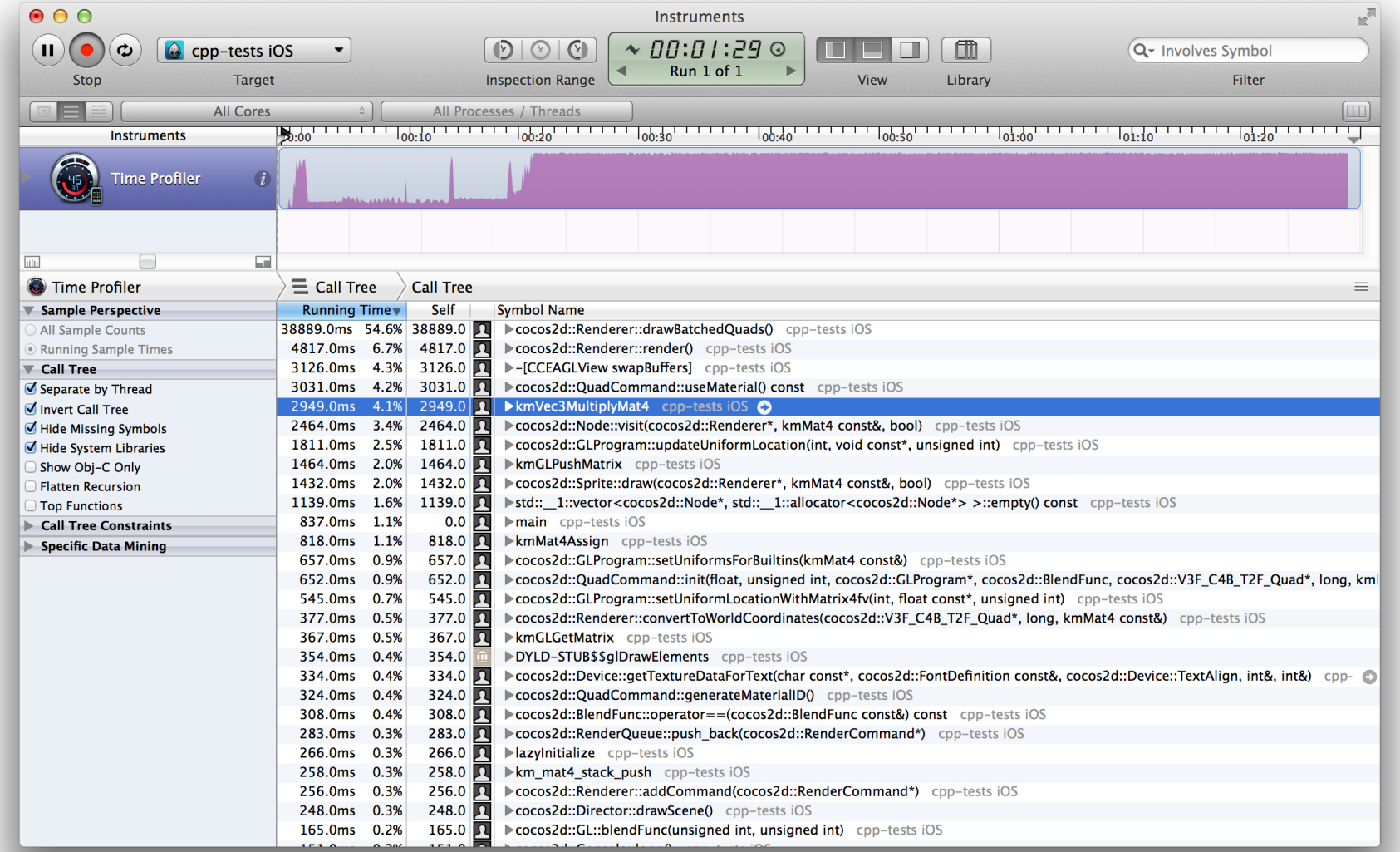


Self	% Self	Instances	Function Name	Location	
1,560	3.71%	1	__addsf3	ieee754-sf.S:68	libjoygamesdk.so
973	2.32%	1	__mulsf3	ieee754-sf.S:452	libjoygamesdk.so
436	1.04%	1	joy::cocos2d::CCParticleSystem::update(float)	CCParticleSystem.cpp:589	libjoygamesdk.so
256	0.61%	1	joy::cocos2d::CCObject::~CCObject_sub_object()	CCObject.cpp:50	libjoygamesdk.so
239	0.57%	1	joy::cocos2d::CCParticleSystemQuad::updateQuadWithParticle(joy::...	CCParticleSystemQuad.cpp:241	libjoygamesdk.so
183	0.44%	1	joy::cocos2d::CCObject::CCObject_sub_object()	CCObject.cpp:40	libjoygamesdk.so
162	0.39%	1	joy::cocos2d::CCPoint::CCPoint(float, float)	CCGeometry.cpp:36	libjoygamesdk.so
147	0.35%	1	__divsf3	ieee754-sf.S:655	libjoygamesdk.so
133	0.32%	1	joy::cocos2d::CCScriptEngineManager::sharedManager()	CCScriptSupport.cpp:130	libjoygamesdk.so
97	0.23%	2	__nesf2	ieee754-sf.S:823	libjoygamesdk.so
78	0.19%	1	joy::cocos2d::CCPoint::operator =(const joy::cocos2d::CCPoint&)	CCGeometry.cpp:47	libjoygamesdk.so
77	0.18%	1	joy::cocos2d::CCNode::visit()	CCNode.cpp:783	libjoygamesdk.so
75	0.18%	2	.plt [libjoygamesdk.so]	libjoygamesdk.so	libjoygamesdk.so
67	0.16%	1	joy::cocos2d::CCPoint::CCPoint()	CCGeometry.cpp:31	libjoygamesdk.so
65	0.15%	1	__truncdfsf2	ieee754-df.S:1382	libjoygamesdk.so
63	0.15%	1	joy::cocos2d::CCNode::nodeToParentTransform()	CCNode.cpp:1106	libjoygamesdk.so
55	0.13%	1	joy::cocos2d::ccpNormalize(const joy::cocos2d::CCPoint&)	CCPointExtension.cpp:49	libjoygamesdk.so
52	0.12%	1	joy::cocos2d::CCPoint::setPoint(float, float)	CCGeometry.cpp:53	libjoygamesdk.so
50	0.12%	1	kmMat4Multiply	mat4.c:218	libjoygamesdk.so
50	0.12%	1	__aeabi_fsub	ieee754-sf.S:59	libjoygamesdk.so
40	0.10%	2	__aeabi_dadd	ieee754-df.S:86	libjoygamesdk.so
39	0.09%	1	__fixunssfsi	ieee754-sf.S:1030	libjoygamesdk.so

iOS®: Xcode® OpenGL® Frame Capture



iOS®: Xcode® Profiler



Lessons Learned

ARM

Lessons Learned

- Profile everything… including recommended best practices
- Profile everything… in different GPUs / GL drivers
- Profile everything… use the same environment when comparing results
- Profile everything… have a comprehensive test suite

If you can't measure it, you can't tell if it is getting better… or worse.

For further information…

- http://www.cocos2d-x.org

Thank You

Questions?

@ricardoquesada