

Lecture 16

Debugging Strategies

There are Two Main Strategies

- **Confirmation**

- Confirm everything you believe to be true
- Find the thing that is not actually true
- In worse case, have to look at every line of code

- **Binary Search**

- Identify where the code is working properly
- Identify where the code is not working properly
- Limit confirmation to the space in between

There are Two Main Strategies

- **Confirmation**

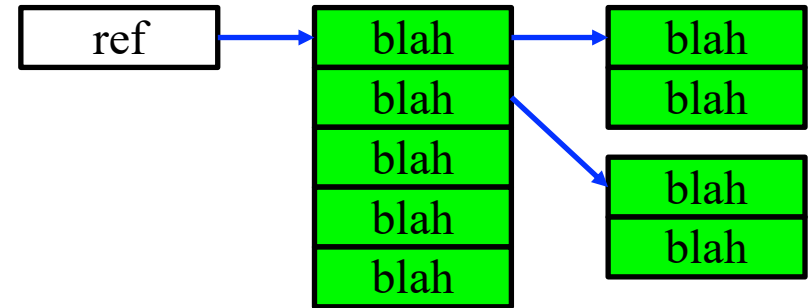
- Confirm everything you believe to be true
- Find the thing that is not actually true
- In worse case, have to look at

- Everything else is a fancy tool to do this

- Identify where the code is working properly
- Identify where the code is not working properly
- Limit confirmation to the space in between

The Challenge of Finding Errors

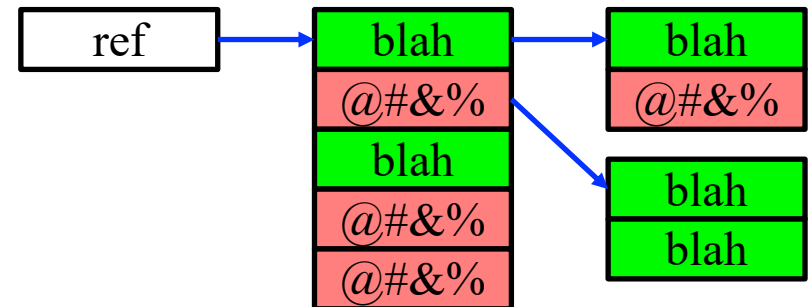
- *Access errors* are the hardest
 - Refer to object in memory
 - Object is deleted somehow
 - Refer to attribute of object
 - May/may not cause crash
- Remember the 1110 rule
 - Error found != error cause
 - Cause is somewhere before
- Must work up the *call stack*
 - Part of the **binary search**



The Challenge of Finding Errors

- *Access errors* are the hardest

- Refer to object in memory
- Object is deleted somehow
- Refer to attribute of object
- May/may not cause crash



- Remember the 1110 rule
 - Error found != error cause
 - Cause is somewhere before
- Must work up the *call stack*
 - Part of the **binary search**

- “Deletion” is not immediate
 - Marks it for deletion
 - Will be deleted later
- Can still access object
 - Data corrupted as recycled

Primitive Confirmation Tools

- **Logging** (CULog)
 - Print out a variable value to check it
 - Alternatively print out a trace of program flow
 - **Goal**: View the internal program state
- **Assertions** (CUAssert)
 - Check that your assumption is true
 - Crash the code if it is not
 - **Goal**: Make error closer to the crash

Primitive Confirmation Tools

- **CULog**(statement, v1, v2, v3...)
 - Uses same syntax as printf()
 - Need to use char* to display string names
 - **Ex:** CULog("Node is %s", node->getName().c_str())
- **CUAssert**(test, statement, v1, v2, v3...)
 - Test is any boolean statement
 - Remainder of arguments act like printf()
 - **Ex:** CUAssert(index > 0, "index is %d", index)

Problems with Logging

- **Verbose**

- Code with print every animation frame
- Way too much information to sort through
- Most game designers will log to a file

- **Distortionary**

- Logging and other I/O is a blocking operation
- Will change the thread behavior of your app
- Can cause errors to appear/disappear

Advanced Tools

- **Breakpoints**

- Stop the execution of the code
- Can continue running from that point
- Can continue one step at a time

- **Watches**

- Look at the value of an individual variable
- Can drill down into object attributes
- But only works when variable is in scope

Advanced Tools

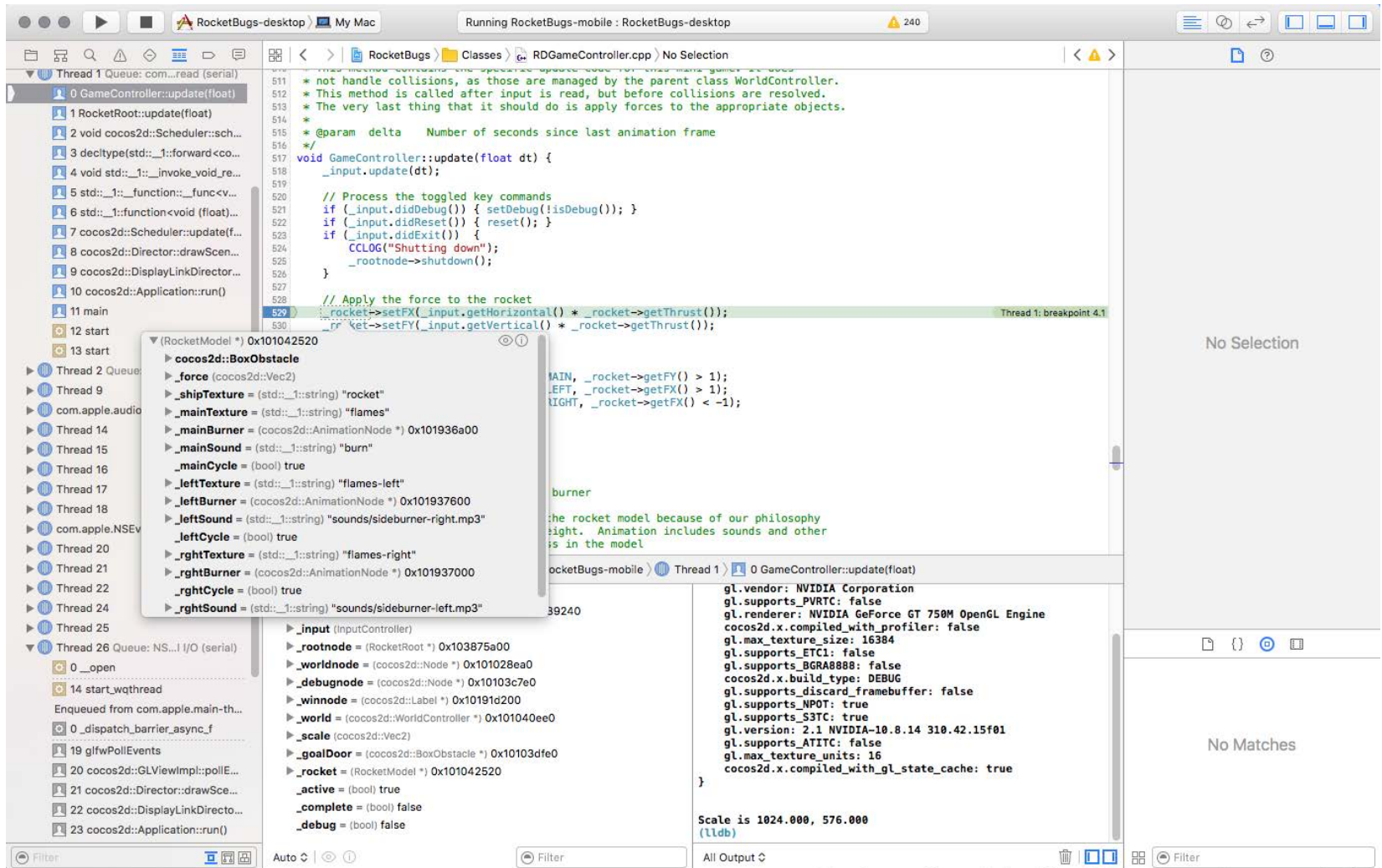
- **Memory Dumps**

- Look at a raw memory location
- Does not require a variable to be in scope
- Good way to look at heap for corruption

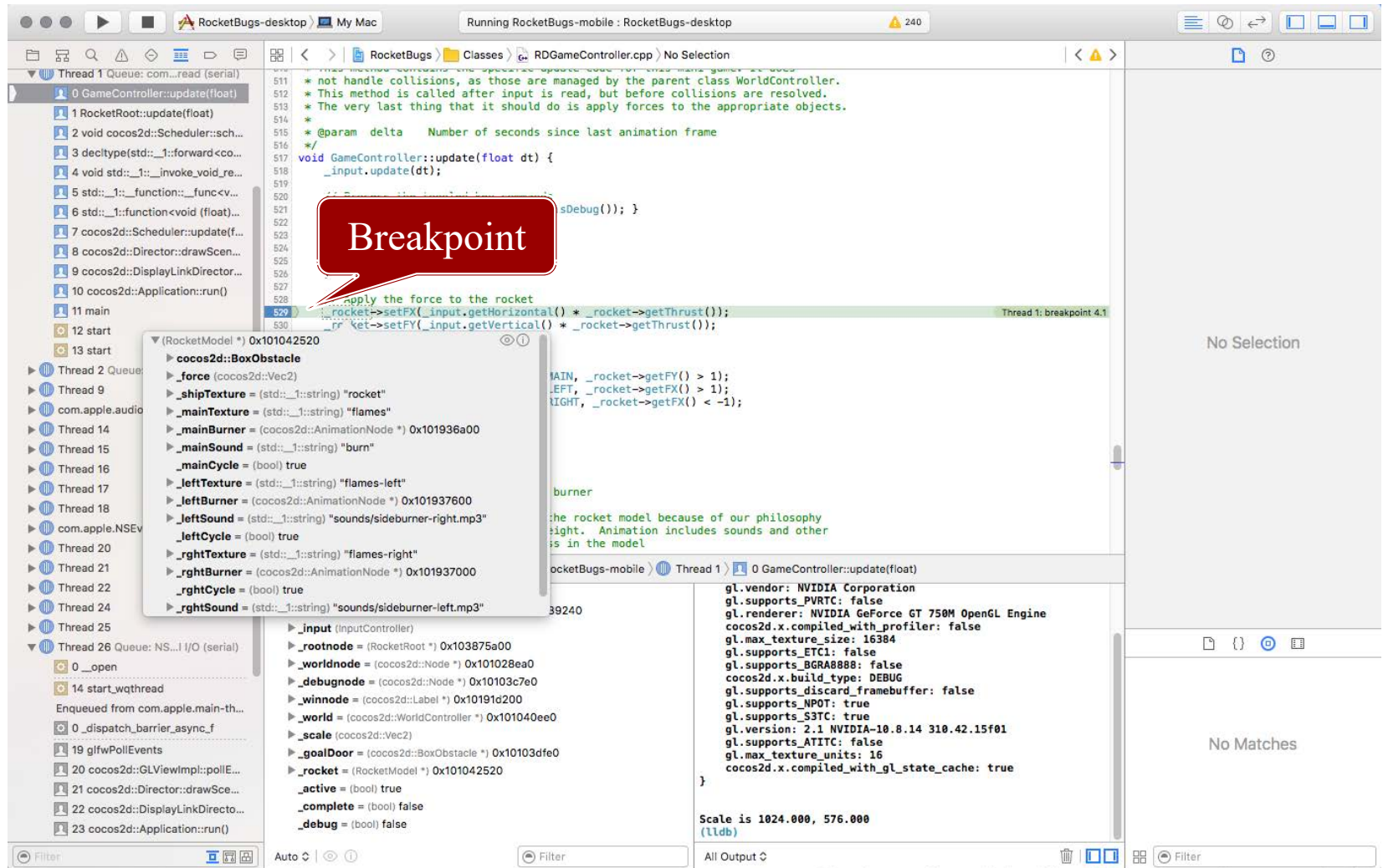
- **Thread Monitors**

- Stack traces for all running threads
- All threads are frozen by a breakpoint
- Allows you to compare state across threads

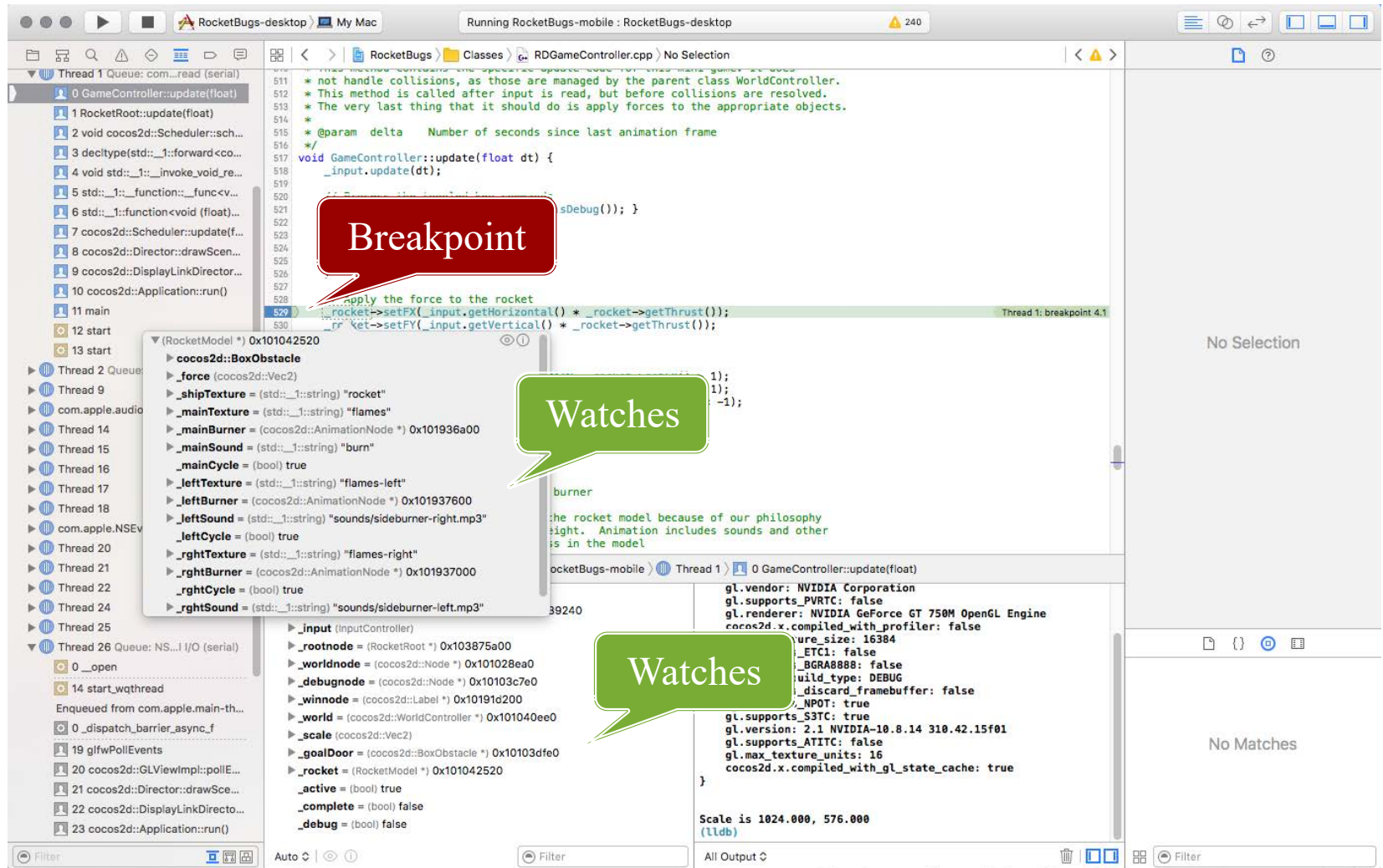
XCode Tools



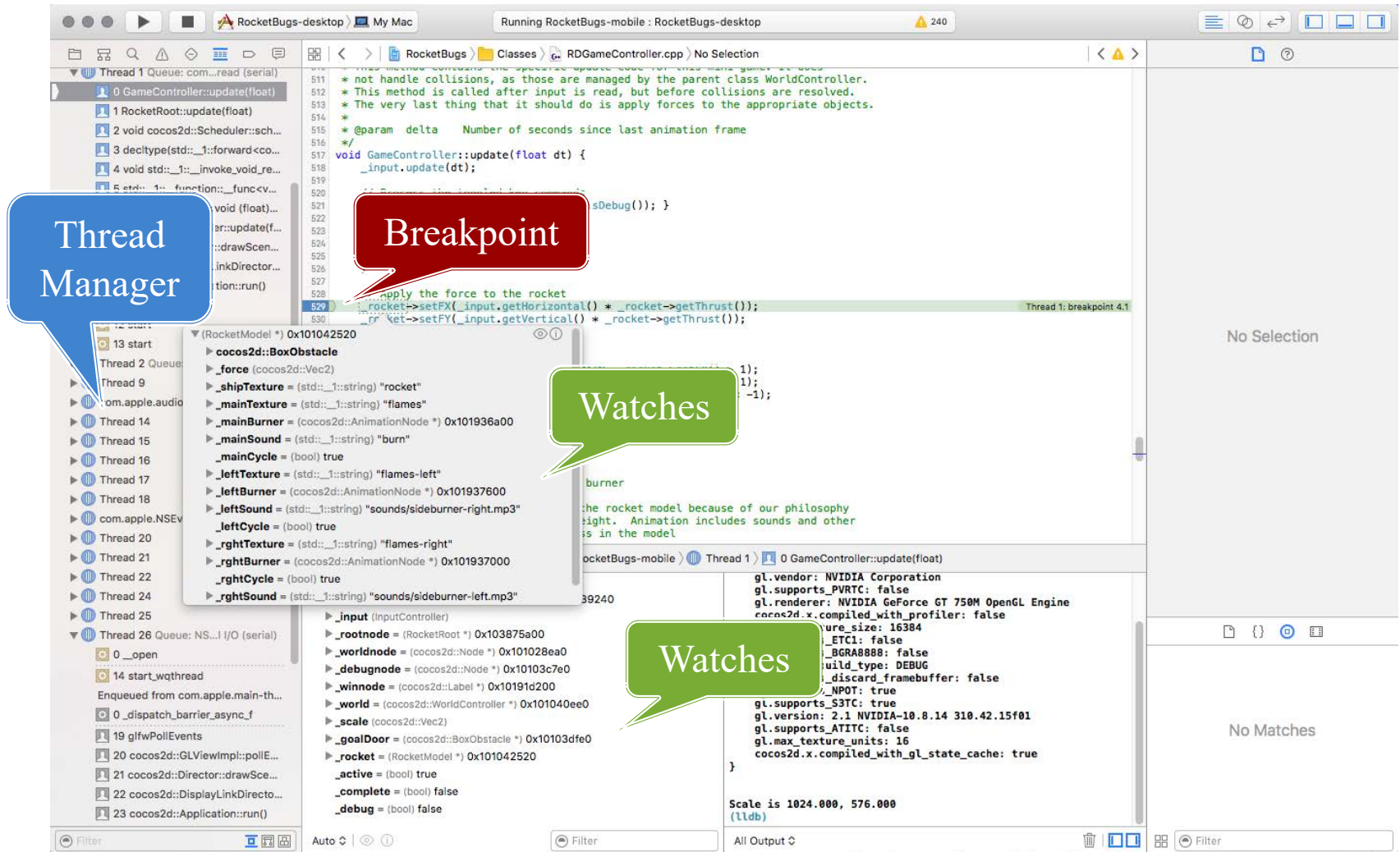
XCode Tools



XCode Tools



XCode Tools



XCode Tools

Memory Dump

Thread 1 Queue: com...read (serial)

- 0 GameController::update(float)
- 1 RocketRoot::update(float)
- 2 void cocos2d::Scheduler::sch...
- 3 decltype(std::_1::forward<co...
- 4 void std::_1::__invoke_void_re...
- 5 std::_1::__function::__func<v...
- 6 std::_1::function<void (float)...
- 7 cocos2d::Scheduler::update(f...
- 8 cocos2d::Director::drawScen...
- 9 cocos2d::DisplayLinkDirector...
- 10 cocos2d::Application::run()
- 11 main
- 12 start
- 13 start

Thread 2 Queue: com...ager (serial)

- Thread 9
- com.apple.audio.IOThread.client (...)
- Thread 14
- Thread 15
- Thread 16
- Thread 17
- Thread 18
- com.apple.NSEventThread (19)
- Thread 20
- Thread 21
- Thread 22
- Thread 24
- Thread 25
- Thread 26 Queue: NS...I I/O (serial)
- 0 _open
- 14 start_wqthread
- Enqueued from com.apple.main-th...
- 0_dispatch_barrier_async_f
- 19 glfwPollEvents
- 20 cocos2d::GLViewImpl::pollE...
- 21 cocos2d::Director::drawSce...
- 22 cocos2d::DisplayLinkDirecto...
- 23 cocos2d::Application::run()

Address: 0x101042520 Page: Lock: ...mber of Bytes: 512

Thread 1 0 GameController::update(float)

```
this = (GameController *) 0x103875e48
  _assets = (cocos2d::SceneManager *) 0x618000089240
  _input (InputController)
  _rootNode = (RocketRoot *) 0x103875a00
  _worldNode = (cocos2d::Node *) 0x101028ea0
  _debugNode = (cocos2d::Node *) 0x10103c7e0
  _winNode = (cocos2d::Label *) 0x10191d200
  _world = (cocos2d::WorldController *) 0x101040ee0
  _scale (cocos2d::Vec2)
  _goalDoor = (cocos2d::BoxObstacle *) 0x10103dfe0
  _rocket = (RocketModel *) 0x101042520
  _active = (bool) true
  _complete = (bool) false
  _debug = (bool) false

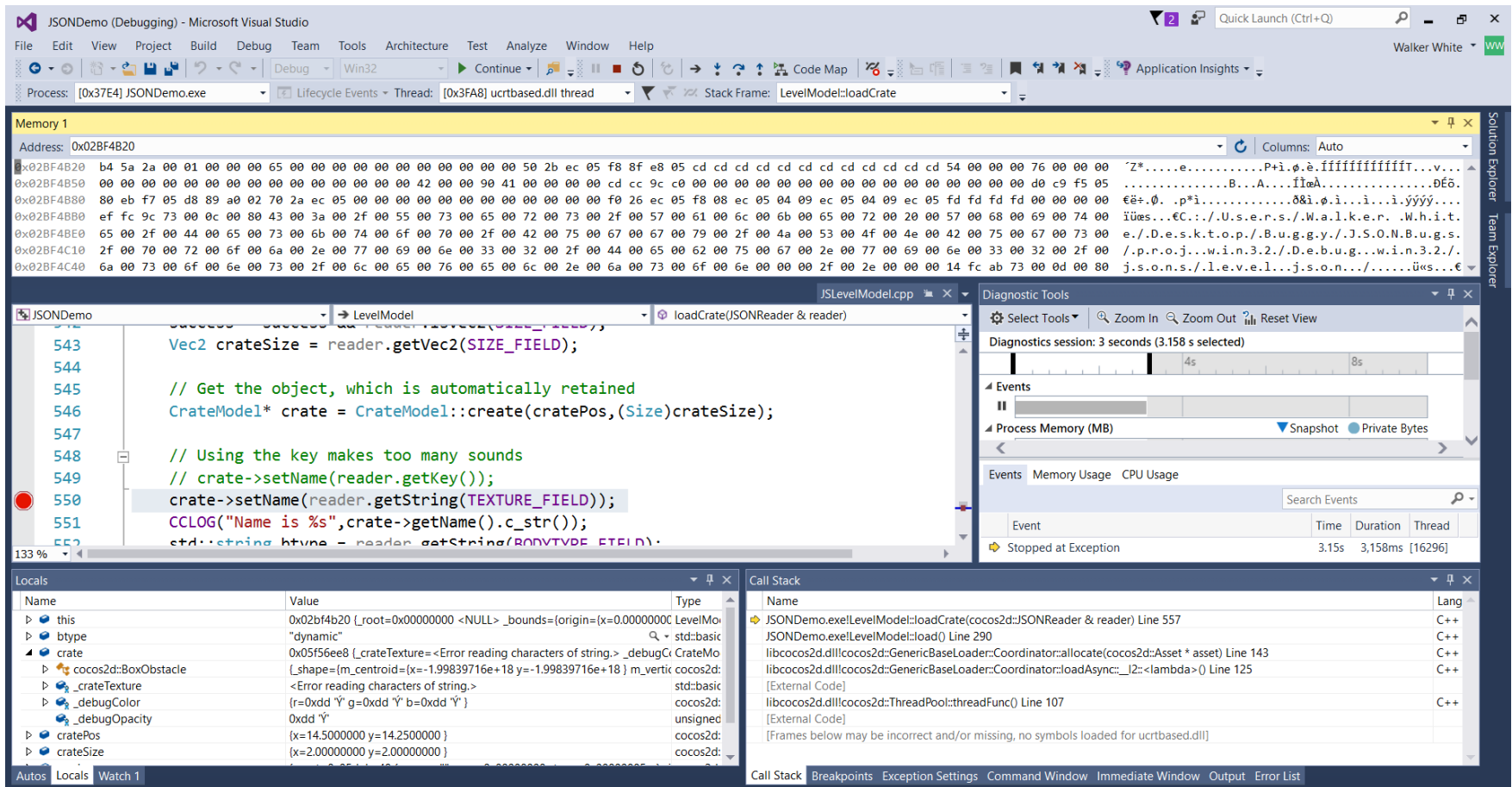
gl.vendor: NVIDIA Corporation
gl.supports_PVRTC: false
gl.renderer: NVIDIA GeForce GT 750M OpenGL Engine
cocos2d.x.compiled_with_profiler: false
gl.max_texture_size: 16384
gl.supports_ETC1: false
gl.supports_BGRA8888: false
cocos2d.x.build_type: DEBUG
gl.supports_discard_framebuffer: false
gl.supports_NPOT: true
gl.supports_S3TC: true
gl.version: 2.1 NVIDIA-10.8.14 310.42.15f01
gl.supports_ATITC: false
gl.max_texture_units: 16
cocos2d.x.compiled_with_gl_state_cache: true

Scale is 1024.000, 576.000
(1ldb)
```

No Selection

No Matches

Visual Studio Tools



Visual Studio Tools

JSONDemo (Debugging) - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help

Process: [0x37E4] JSONDemo.exe Thread: [0x3FA8] ucrtbased.dll thread Stack Frame: LevelModel::loadCrate

Memory 1

Address: 0x02BF4B20

Columns: Auto

JSLevelModel.cpp

loadCrate(JSONReader & reader)

543 Vec2 crateSize = reader.getVec2(SIZE_FIELD);

544

549 // Using the key makes too many sounds

550 // crate->setName(reader.getKey());

551 crate->setName(reader.getString(TEXTURE_FIELD));

552 CCLOG("Name is %s", crate->getName().c_str());

553 std::string btune = reader.getString(BONVTYPE_FIELD);

133 %

Locals

Name	Value	Type
this	0x02bf4b20 [_root=0x00000000 <NULL>] _bounds=(origin=(x=0.00000000 LevelMo	
btype	"dynamic"	std::basic
crate	0x05f56ee8 [_crateTexture=<Error reading characters of string>] _debugCr CrateMo	
cocos2d::BoxObstacle	[_shape=(m_centroid=(x=-1.99839716e+18 y=-1.99839716e+18) m_vertic cocos2d:	
_crateTexture	<Error reading characters of string>	std::basic
_debugColor	{r=0xdd 'Y' g=0xdd 'Y' b=0xdd 'Y'}	cocos2d:
_debugOpacity	0xdd 'Y'	unsigned
cratePos	{x=14.50000000 y=14.25000000}	cocos2d:
crateSize	{x=2.00000000 y=2.00000000}	cocos2d:

Call Stack

Name	Lang
JSONDemo.exe!LevelModel::loadCrate(cocos2d::JSONReader & reader) Line 557	C++
JSONDemo.exe!LevelModel::load() Line 290	C++
libcocos2d.dll!cocos2d::GenericBaseLoader::Coordinator::allocate(cocos2d::Asset * asset) Line 143	C++
libcocos2d.dll!cocos2d::GenericBaseLoader::Coordinator::loadAsync(_I2:<lambda>)() Line 125	C++
[External Code]	
libcocos2d.dll!cocos2d::ThreadPool::threadFunc() Line 107	C++
[External Code]	
[Frames below may be incorrect and/or missing, no symbols loaded for ucrtbased.dll]	

Diagnostic Tools

Select Tools Zoom In Zoom Out Reset View

Diagnostics session: 3 seconds (3.15s selected)

Events

Process Memory (MB) Snapshot Private Bytes

Events Memory Usage CPU Usage

Search Events

Event Time Duration Thread

Stopped at Exception 3.15s 3,158ms [16296]

Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List

The screenshot displays the Visual Studio 2019 IDE with the 'JSONDemo' project open. A breakpoint is set at line 550 in 'LevelModel.cpp'. The 'Watches' window shows the following variables and their values:

Name	Value
this	0x02bf4b20 [root=0x00000000 <NULL> <bound> "dynamic"]
btype	0x05f56ee8 [_crateTexture=<Error reading character>]
crate	{_shape=(m_centroid=(x=-1.99839716e+18 y=-1.99839716e+18), m_vertices=cocos2d::BoxObstacle), _crateTexture=<Error reading characters of string>, _debugColor=(r=0xdd 'y'=0xdd 'b'=0xdd 'y'), _debugOpacity=0xdd 'y', cratePos=(x=14.50000000 y=14.25000000), crateSize=(x=2.00000000 y=2.00000000)}

The 'Diagnostics Tools' window shows a 'Stopped at Exception' message at line 557 in 'LevelModel.cpp'. The 'Call Stack' window shows the following frames:

- JSONDemo.exe!LevelModel::loadCrate(cocos2d::JSONReader & reader) Line 557
- JSONDemo.exe!LevelModel::load() Line 290
- libcocos2d.dll!cocos2d::GenericBaseLoader::Coordinator::allocate(cocos2d::Asset * asset) Line 143
- libcocos2d.dll!cocos2d::GenericBaseLoader::Coordinator::loadAsync(_j2=<lambda>) Line 125
- [External Code]
- libcocos2d.dll!cocos2d::ThreadPool::threadFunc() Line 107
- [External Code]

The 'Diagnostics Tools' window also shows a 'Process Memory (MB)' graph and a 'Search Events' field.

Visual Studio Tools

The screenshot displays the Visual Studio IDE with the following components:

- Memory Dump:** A blue speech bubble points to the "Memory 1" window, which shows a hex dump of memory at address 0x02BF4B20. The dump includes ASCII text on the right side, such as "Z*.....e.....P+i.e.fffffffff...v...".
- Breakpoint:** A red speech bubble points to a breakpoint set at line 550 in the `JSLevelModel.cpp` file. The code at this line is `crate->setName(reader.getString(TEXTURE_FIELD));`. A comment above it says "ject, which is automatically retained".
- Watches:** A green speech bubble points to the "Watches" window, which shows the values of variables in the current scope. The variables and their values are:
 - `this`: 0x02bf4b20 [root=0x00000000 <NULL> _bound...
 - `btype`: "dynamic"
 - `crate`: 0x05f56ee8 [_crateTexture=<Error reading chara...
 - `cocos2d::BoxObstacle`: [shape=(m_centroid=(x=-1.99839716e+18 y=-1.9...
 - `_crateTexture`: <Error reading characters of string.>
 - `_debugColor`: (r=0xdd 'Y' g=0xdd 'Y' b=0xdd 'Y')
 - `_debugOpacity`: 0xdd 'Y'
 - `cratePos`: (x=14.50000000 y=14.25000000)
 - `crateSize`: (x=2.00000000 y=2.00000000)
- Call Stack:** The "Call Stack" window shows the sequence of function calls, including `JSONDemo.exe!LevelModel::loadCrate(cocos2d::JSONReader & reader) Line 557`.
- Other Windows:** The "Locals" window shows the current scope's variables. The "Diagnostics Tools" window shows the "Events" tab with a "Stopped at Exception" message.

Visual Studio Tools

The screenshot displays the Visual Studio IDE with several debugging tools open. A blue speech bubble labeled "Memory Dump" points to the Memory 1 window, which shows a hex dump of memory at address 0x02BF4B20. A red speech bubble labeled "Breakpoint" points to a breakpoint set on line 550 of JSLevelModel.cpp. A green speech bubble labeled "Watches" points to the Watches window, which shows the values of variables like 'this', 'btype', 'crate', 'cocos2d::BoxObstacle', 'crateTexture', '_debugColor', '_debugOpacity', 'cratePos', and 'crateSize'. A blue speech bubble labeled "Call Stack" points to the Call Stack window, which shows the sequence of function calls leading to the current state, including 'loadCrate' and 'load'.

Visual Studio (Debugging) - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help

Process: [0x37E4] JSONDemo.exe Thread: [0x3FA8] ucrtbased.dll thread Stack Frame: Level

Memory 1

Address: 0x02BF4B20

Columns: Auto

JSLevelModel.cpp

543 Vec2 crateSize = reader.getVec2(SIZE_FIELD);

544

549 // Using the key makes too many sounds

550 crate->setName(reader.getKey());

551 crate->setName(reader.getString(TEXTURE_FIELD));

552 CCLOG("Name is %s", crate->getName().c_str());

553 std::string btune = reader.getString(BONVTYPE_FIELD);

133 %

Locals

Name Value

this 0x02bf4b20 [_root=0x00000000 <NULL>] _bound

btype "dynamic"

crate 0x05f56ee8 [_crateTexture=<Error reading chara

cocos2d::BoxObstacle [_shape=(m_centroid=(x=-1.99839716e+18 y=-1.99839716e+18) m_vertices=cocos2d::Vec2(0,0) m_texture=crateTexture m_debugColor=(r=0xdd Y g=0xdd Y b=0xdd Y) m_debugOpacity=0xdd Y

crateTexture <Error reading characters of string.>

_debugColor (r=0xdd Y g=0xdd Y b=0xdd Y)

_debugOpacity 0xdd Y

cratePos (x=14.50000000 y=14.25000000)

crateSize (x=2.00000000 y=2.00000000)

Watches

Call Stack

Name Lang

JSONDemo.exe!LevelModel::loadCrate(cocos2d::JSONReader & reader) Line 557 C++

JSONDemo.exe!LevelModel::load() Line 290 C++

libcocos2d.dll!cocos2d::GenericBaseLoader::Coordinator::allocate(cocos2d::Asset * asset) Line 143 C++

libcocos2d.dll!cocos2d::GenericBaseLoader::Coordinator::loadAsync(_I2:<lambda>)() Line 125 C++

[External Code]

libcocos2d.dll!cocos2d::ThreadPool::threadFunc() Line 107 C++

[External Code]

[Frames below may be incorrect and/or missing, no symbols loaded for ucrtbased.dll]

Call Stack Breakpoints Exception Settings Command Window Immediate Window Output Error List

Visual Studio Tools

The screenshot displays the Visual Studio IDE with several debugging tools open:

- Memory Dump:** A window showing a memory dump at address 0x02BF4B20. It contains a grid of hexadecimal and ASCII data.
- Breakpoint:** A red dot on the left margin of the code editor, indicating a breakpoint set at line 550.
- Watches:** A window showing the values of variables in the current scope. It includes a table with columns for Name, Value, and a green callout bubble labeled "Watches".
- Call Stack:** A window showing the sequence of function calls. It includes a table with columns for Name, Lang, and a blue callout bubble labeled "Call Stack".
- Threads:** A window showing the list of threads. It includes a table with columns for Name, Lang, and a blue callout bubble labeled "Threads have a separate window".

The code editor shows the following code snippet:

```
543 Vec2 crateSize = reader.getVec2(SIZE_FIELD);
544
545 // Using the key makes too many sounds
546 // crate->setName(reader.getKey());
547 crate->setName(reader.getString(TEXTURE_FIELD));
548 CCLOG("Name is %s", crate->getName().c_str());
549 std::string btune = reader.getString(BONVTYPE_FIELD);
```

Breakpoint Strategies

- **Break early**

- Break before the error, to check everything is okay
- Step forward and watch how the code changes

- **Break infrequently**

- If you always break, cannot initialize or animate anything
- Design special conditionals for your breakpoint

- **Break on deletion**

- Put breakpoints inside of all your destructors
- Allows you to track accidental deletion

Problems with Code Stepping

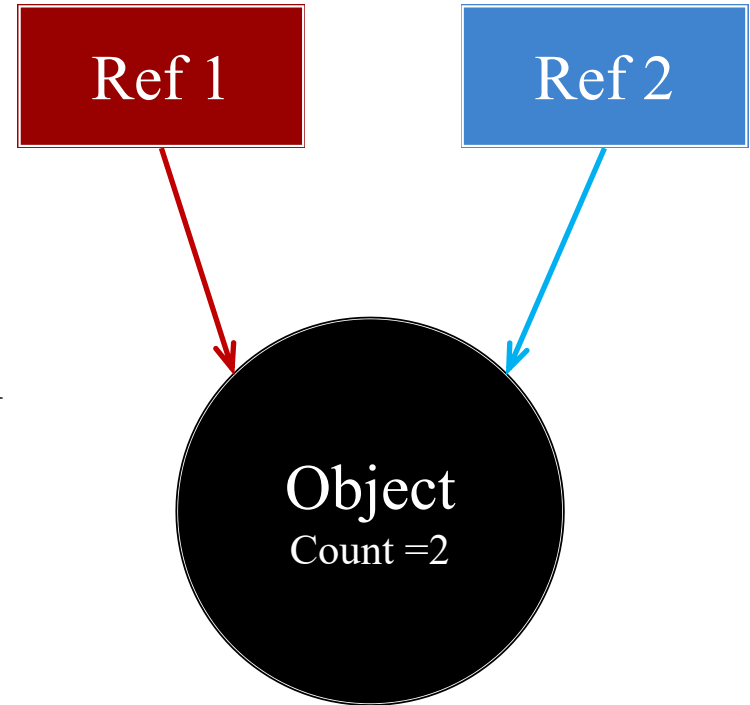
- Code stepping is not “thread safe”
 - Will never leave your current thread
 - Have to choose “continue” instead of “step”
- Makes it very difficult to find thread errors
 - May miss when a variable changes state
 - We had many problems in an old AudioEngine
- **Solution:** Rely heavily on assertions
 - Assert every variable shared across threads
 - Assert them everywhere they may change

Case Study: JSON Loading

- Problem in Cocos2d-x, an older engine
 - Not a C++11 compliant engine
 - Did not support smart pointers (or anything)
 - Instead all game objects had *reference counting*
- Manual reference counting leads to mistakes
 - Only slightly better than manual deletion
 - Even Apple has abandoned this in Objective-C
- But very instructive for **debugging memory**

Aside: Reference Counting

- Every object has a **counter**
 - Tracks number of “owners”
 - No owners = memory leak
 - Increment when get reference
- Often an explicit method call
 - Historically called `retain()`
- Decrement when reference lost
 - Method call is `release()`
 - If makes count 0, delete it



Scene Graphs the Old Way

// create a new instance

Node* node = Node::create();

Raw pointers!!

node->retain();

Manual refence counts!!

// Add the node to scene graph

scene->addChild(**node**);

// Release the local reference

node->release();

// Remove from scene graph

scene->removeChild(**node**);

Scene Graphs the Old Way

// create a new instance

Node* node = Node::create();

Custom allocator

node->retain();

Reference count 1

// Add the node to scene graph

scene->addChild(**node**);

Reference count 2

// Release the local reference

node->release();

Reference count 1

// Remove from scene graph

scene->removeChild(**node**);

Reference count 0

node is **deleted**

Scene Graphs the Old Way

// create a new instance

Node* node = Node::create();

Custom allocator

node->retain();

Reference count 1

// Add the node to scene graph

scene->addChild(**node**);

Reference count 2

// Do not release the local reference

// Remove from scene graph

scene->removeChild(**node**);

Reference count 1

Memory Leak!

Case Study: JSON Loading

- Problem was a thread *race condition*
 - Appeared on Windows, but not MacOS
 - Because of particular Windows thread schedule
 - But technically unsafe on all platforms
- Found by putting **breakpoints in destructors**
 - Models getting deleted immediately after creation
 - Watched the reference counts to find problem
 - There was a stray `release()` before `retain()`

Case Study: b2BlockAllocator

- *Memory address* problem in Box2D engine
 - Problem was because we put Box2D in a DLL
 - Required stepping through the allocation process
 - Required **memory dumps** to view the heap
- Problem with the *static global variables*
 - DLLs have a distinct global space
 - BlockAllocator was initialized inside of the DLL
 - When it was used outside the DLL, not initialized

Summary

- Two main strategies to debugging
 - **Confirmation:** Make sure code does what you think
 - **Binary Search:** Find where confirmation wrong
- Primitive tools in code on all platforms
 - Logging with CULog
 - Assertions with CUAssert
- Advanced tools in professional IDEs
 - Breakpoints and Watches
 - Thread Monitors (to see call stack)
 - Memory Dumps