the gamedesigninitiative at cornell university

Debugging Strategies

There are Two Main Strategies

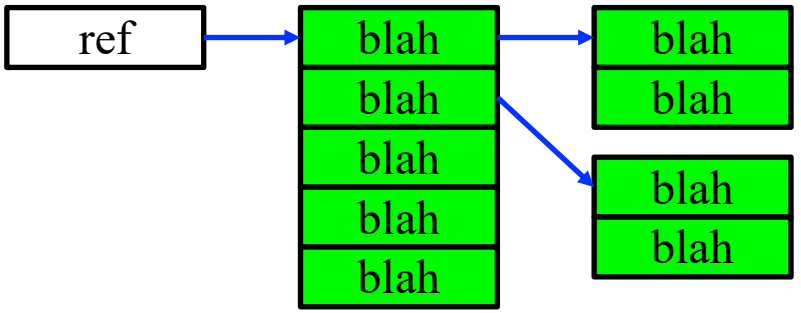
- **Confirmation**
  - Confirm everything you believe to be true
  - Find the thing that is not actually true
  - In worse case, have to look at every line of code
- **Binary Search**
  - Identify where the code is working properly
  - Identify where the code is not working properly
  - Limit confirmation 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

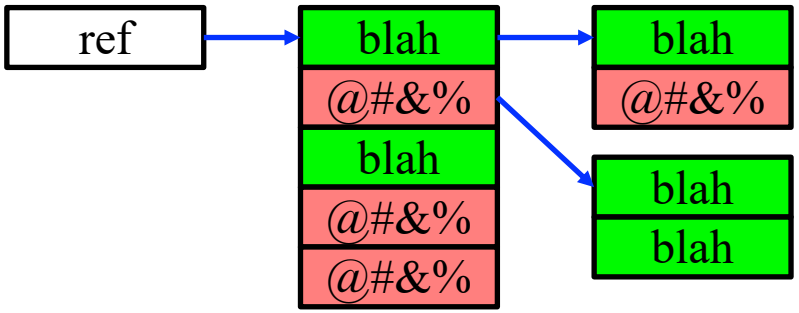
The Challenge of Finding Errors

- *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

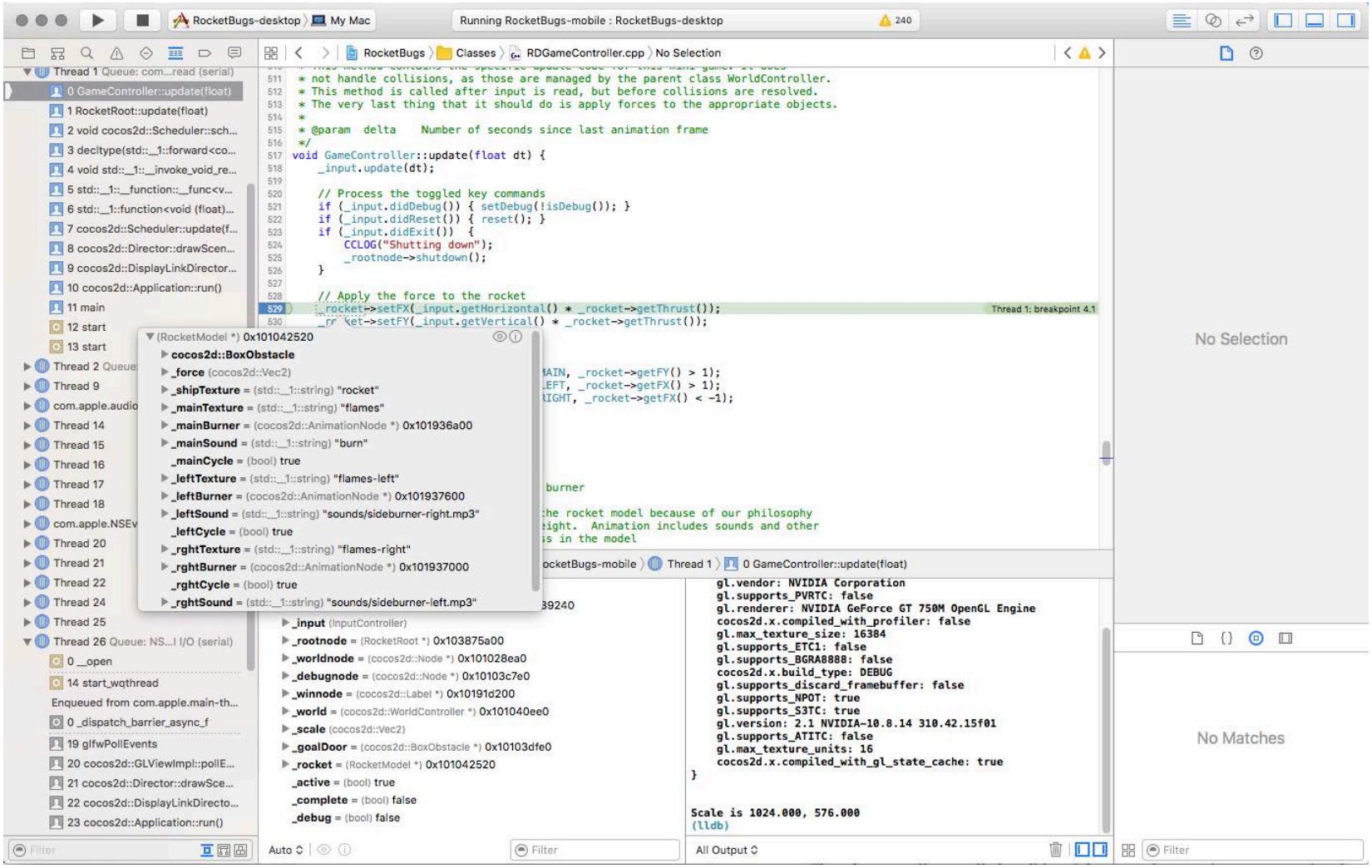
Advanced Tools

- **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

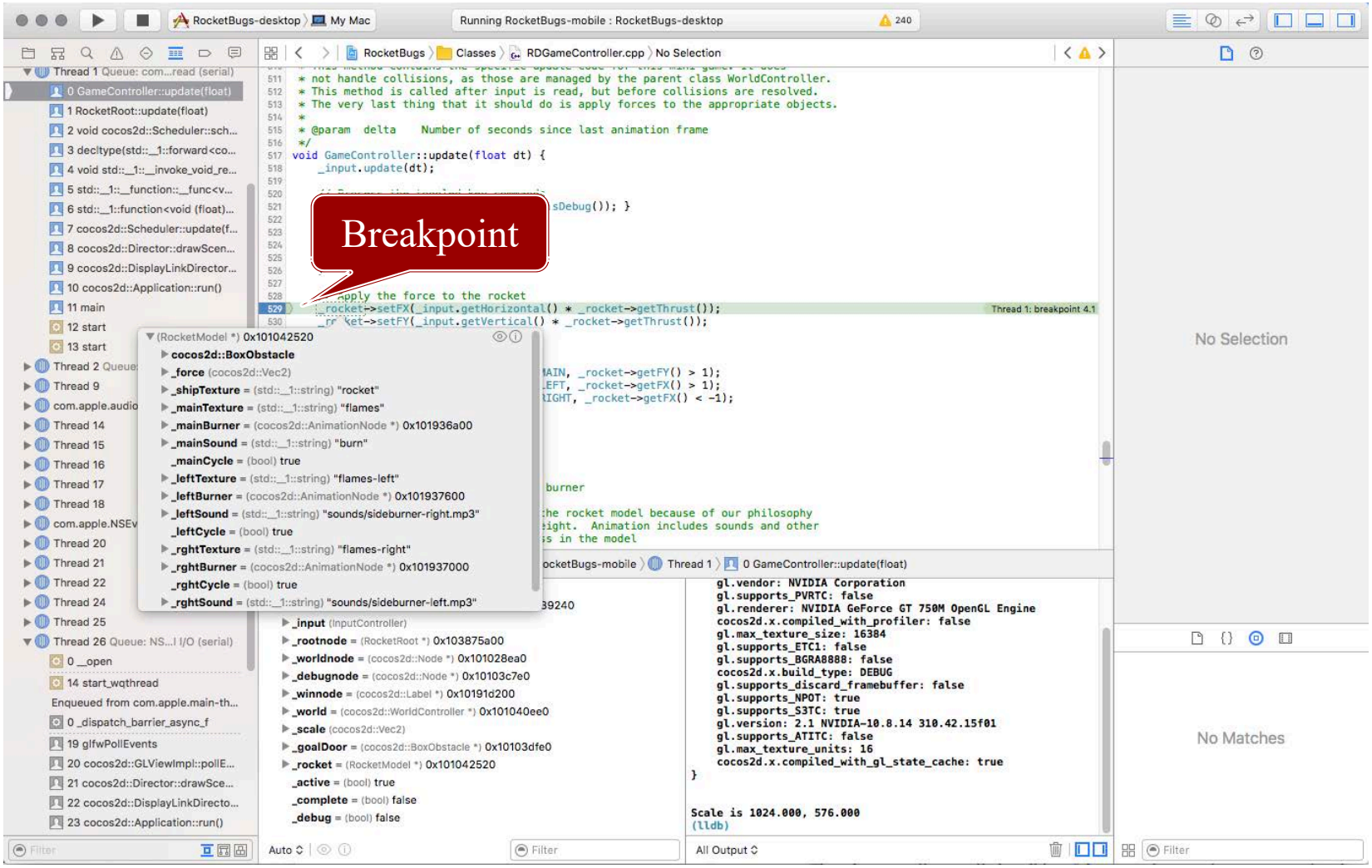
Advanced Tools

- **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

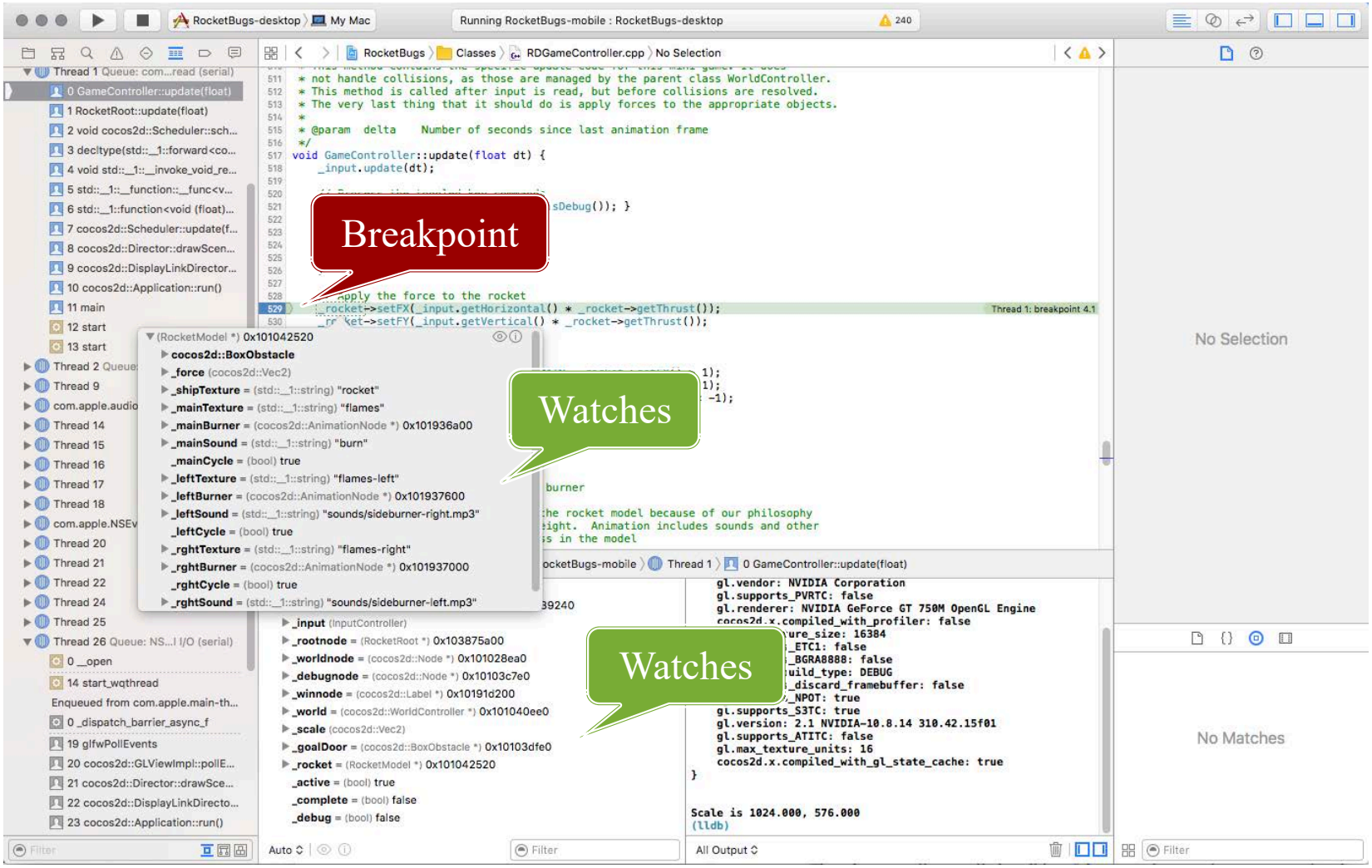
XCode Tools


RocketBugs-desktop
My Mac
Running RocketBugs-mobile : RocketBugs-desktop
RocketBugs
Classes
RDGameController.cpp
No Selection
* not handle collisions, as those are managed by the parent class WorldController.
* This method is called after input is read, but before collisions are resolved.
* The very last thing that it should do is apply forces to the appropriate objects.
* @param delta Number of seconds since last animation frame
void GameController::update(float dt) {
_input.update(dt);
// Process the toggled key commands
if (_input.didDebug()) { setDebug(!isDebug()); }
if (_input.didReset()) { reset(); }
if (_input.didExit()) {
CCLOG("Shutting down");
_rootnode->shutdown();
}
// Apply the force to the rocket
_rocket->setFX(_input.getHorizontal() * _rocket->getThrust());
_rocket->setFY(_input.getVertical() * _rocket->getThrust());
Thread 1: breakpoint 4.1
No Selection
No Matches
Thread 1 Queue: com...read (serial)
0 GameController::update(float)
1 RocketRoot::update(float)
(RocketModel *) 0x101042520
cocos2d::BoxObstacle
_force (cocos2d::Vec2)
_shipTexture = (std::__1::string) "rocket"
_mainTexture = (std::__1::string) "flames"
_mainBurner = (cocos2d::AnimationNode *) 0x101936a00
_mainSound = (std::__1::string) "burn"
_mainCycle = (bool) true
_leftTexture = (std::__1::string) "flames-left"
_leftBurner = (cocos2d::AnimationNode *) 0x101937600
_leftSound = (std::__1::string) "sounds/sideburner-right.mp3"
_leftCycle = (bool) true
_rghtTexture = (std::__1::string) "flames-right"
_rghtBurner = (cocos2d::AnimationNode *) 0x101937000
_rghtCycle = (bool) true
_rghtSound = (std::__1::string) "sounds/sideburner-left.mp3"
_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
(lldb)

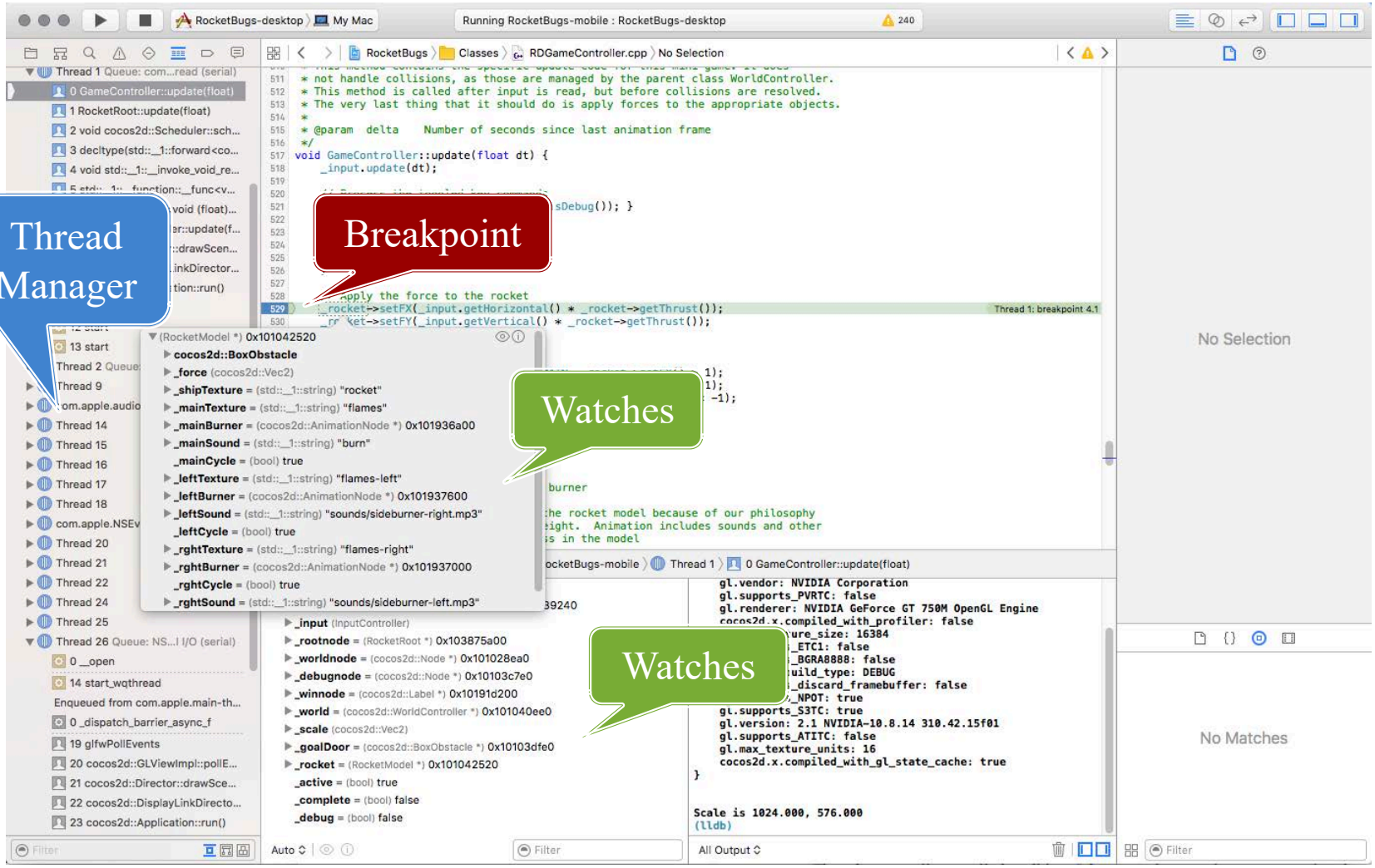
XCode Tools


Breakpoint
RocketBugs-desktop
My Mac
Running RocketBugs-mobile : RocketBugs-desktop
RocketBugs
Classes
RDGameController.cpp
No Selection
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 1 Queue: com...read (serial)
Thread 2 Queue
Thread 9
com.apple.audio
Thread 14
Thread 15
Thread 16
Thread 17
Thread 18
com.apple.NSEv
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()
511 * not handle collisions, as those are managed by the parent class WorldController.
512 * This method is called after input is read, but before collisions are resolved.
513 * The very last thing that it should do is apply forces to the appropriate objects.
514 *
515 * @param delta Number of seconds since last animation frame
516 */
517 void GameController::update(float dt) {
518 _input.update(dt);
519
520
521 sDebug()); }
522
523
524
525
526
527
528 Apply the force to the rocket
529 _rocket->setFX(_input.getHorizontal() * _rocket->getThrust());
Thread 1: breakpoint 4.1
530 _rocket->setFY(_input.getVertical() * _rocket->getThrust());
(RocketModel *) 0x101042520
cocos2d::BoxObstacle
_force (cocos2d::Vec2)
_shipTexture = (std::__1::string) "rocket"
_mainTexture = (std::__1::string) "flames"
_mainBurner = (cocos2d::AnimationNode *) 0x101936a00
_mainSound = (std::__1::string) "burn"
_mainCycle = (bool) true
_leftTexture = (std::__1::string) "flames-left"
_leftBurner = (cocos2d::AnimationNode *) 0x101937600
_leftSound = (std::__1::string) "sounds/sideburner-right.mp3"
_leftCycle = (bool) true
_rghtTexture = (std::__1::string) "flames-right"
_rghtBurner = (cocos2d::AnimationNode *) 0x101937000
_rghtCycle = (bool) true
_rghtSound = (std::__1::string) "sounds/sideburner-left.mp3"
MAIN, _rocket->getFY() > 1);
LEFT, _rocket->getFX() > 1);
RIGHT, _rocket->getFX() < -1);
burner
the rocket model because of our philosophy
eight. Animation includes sounds and other
s in the model
RocketBugs-mobile
Thread 1
0 GameController::update(float)
39240
_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
(lldb)
No Selection
No Matches
Auto
All Output
Filter

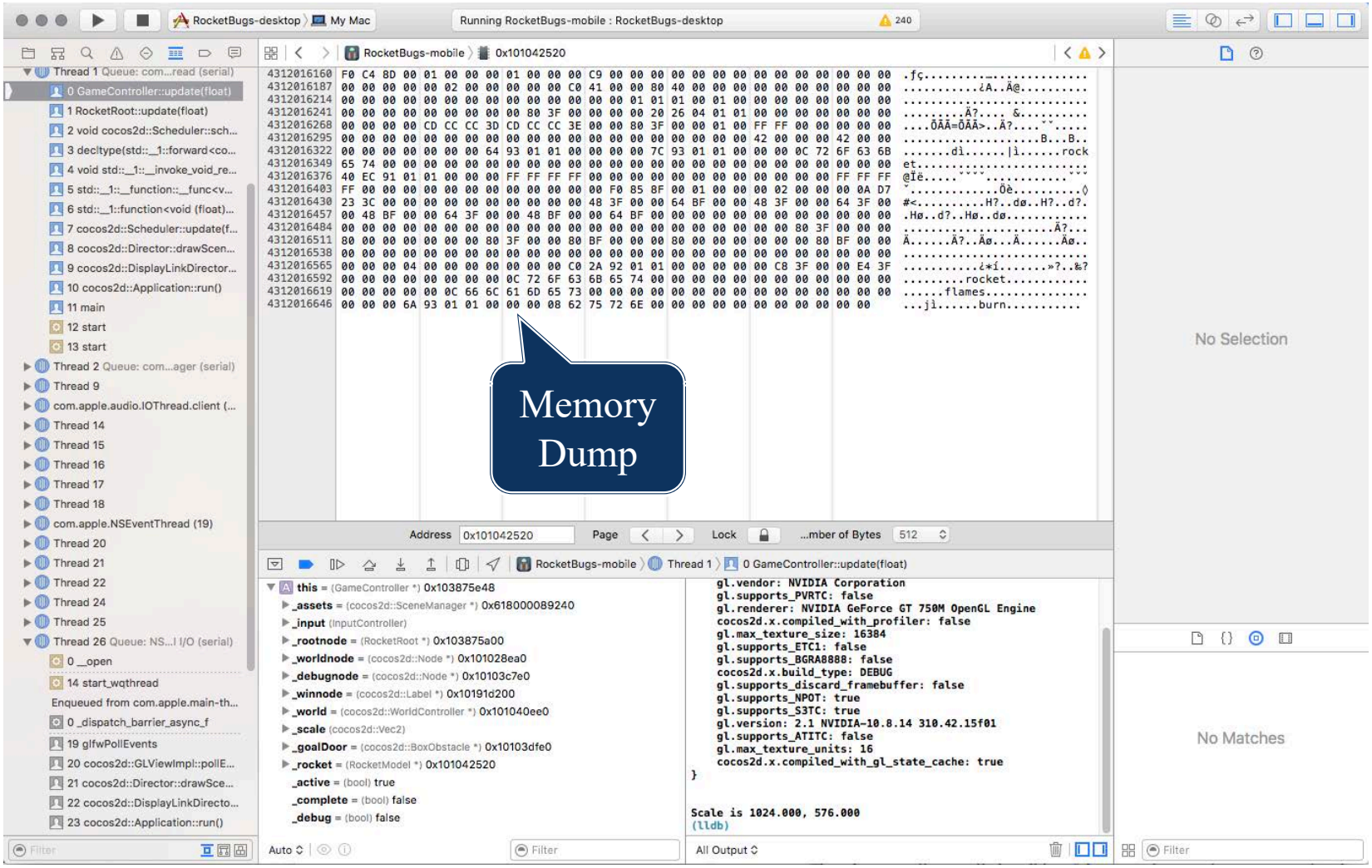
XCode Tools



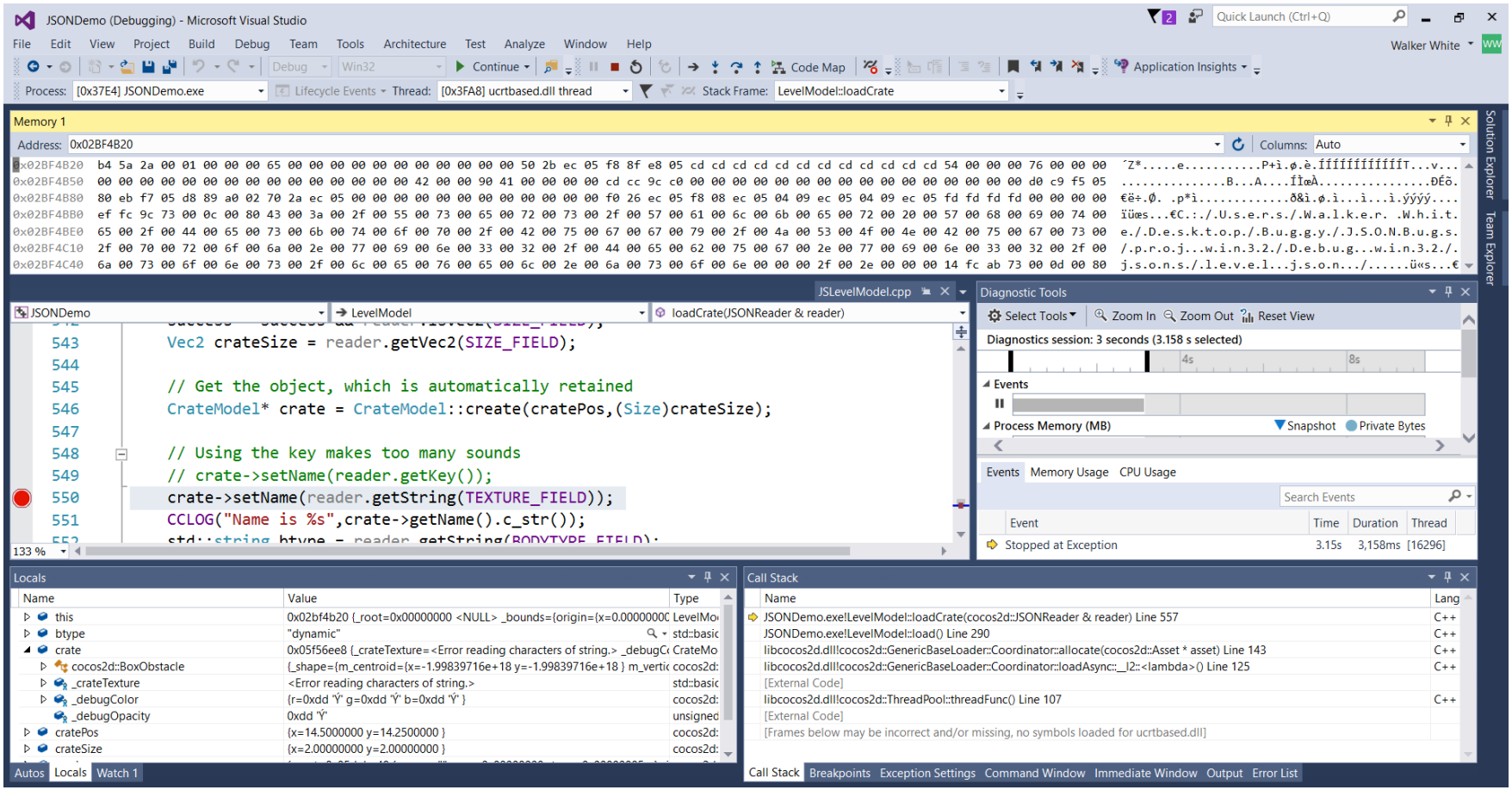
XCode Tools


Thread Manager
Breakpoint
Watches
Watches
No Selection
No Matches

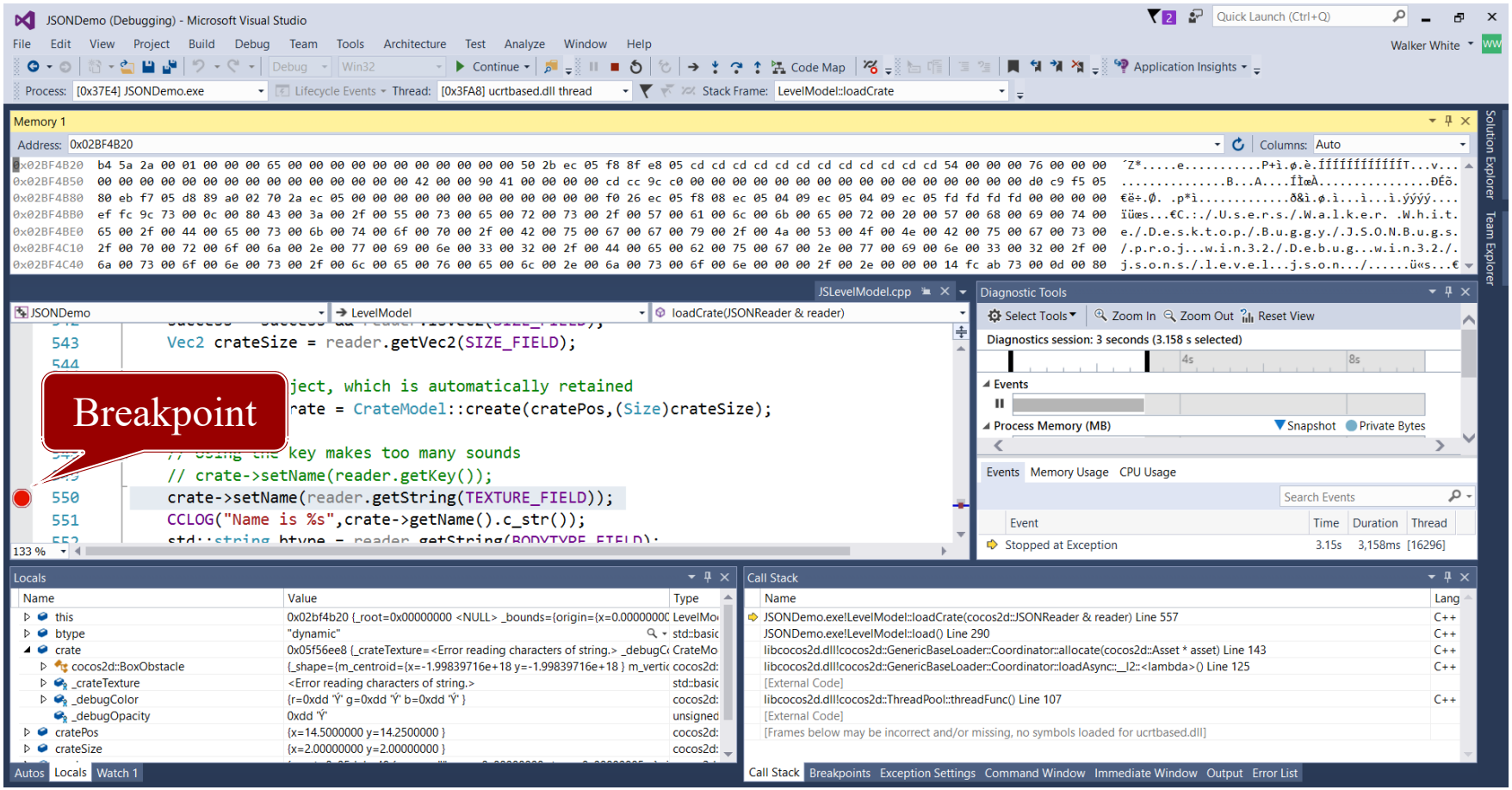
XCode Tools


RocketBugs-desktop
My Mac
Running RocketBugs-mobile : RocketBugs-desktop
RocketBugs-mobile
0x101042520
Memory Dump
No Selection
No Matches
Address
0x101042520
Page
Lock
Thread 1
0 GameController::update(float)
Scale is 1024.000, 576.000
(lldb)

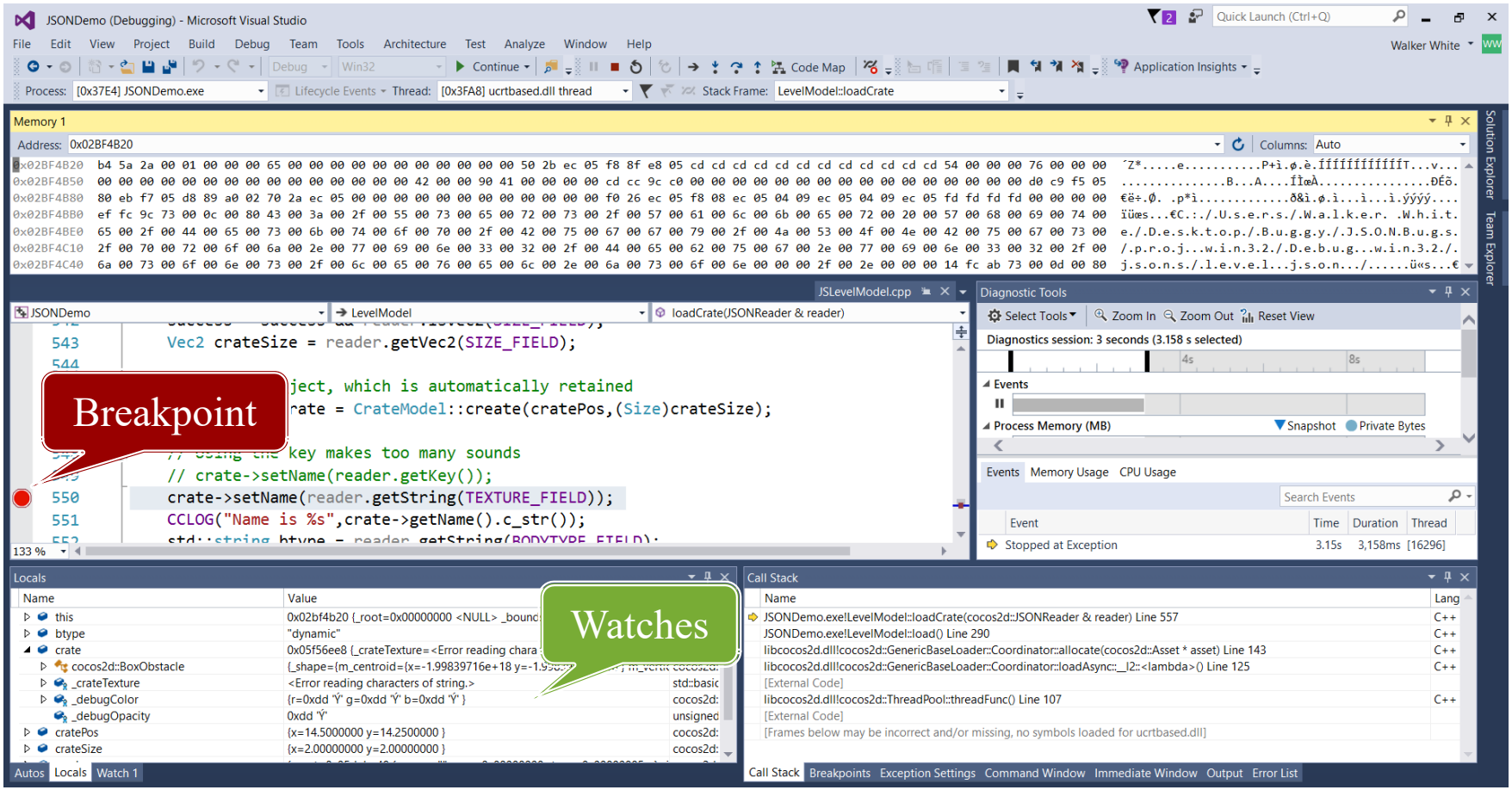
Visual Studio Tools



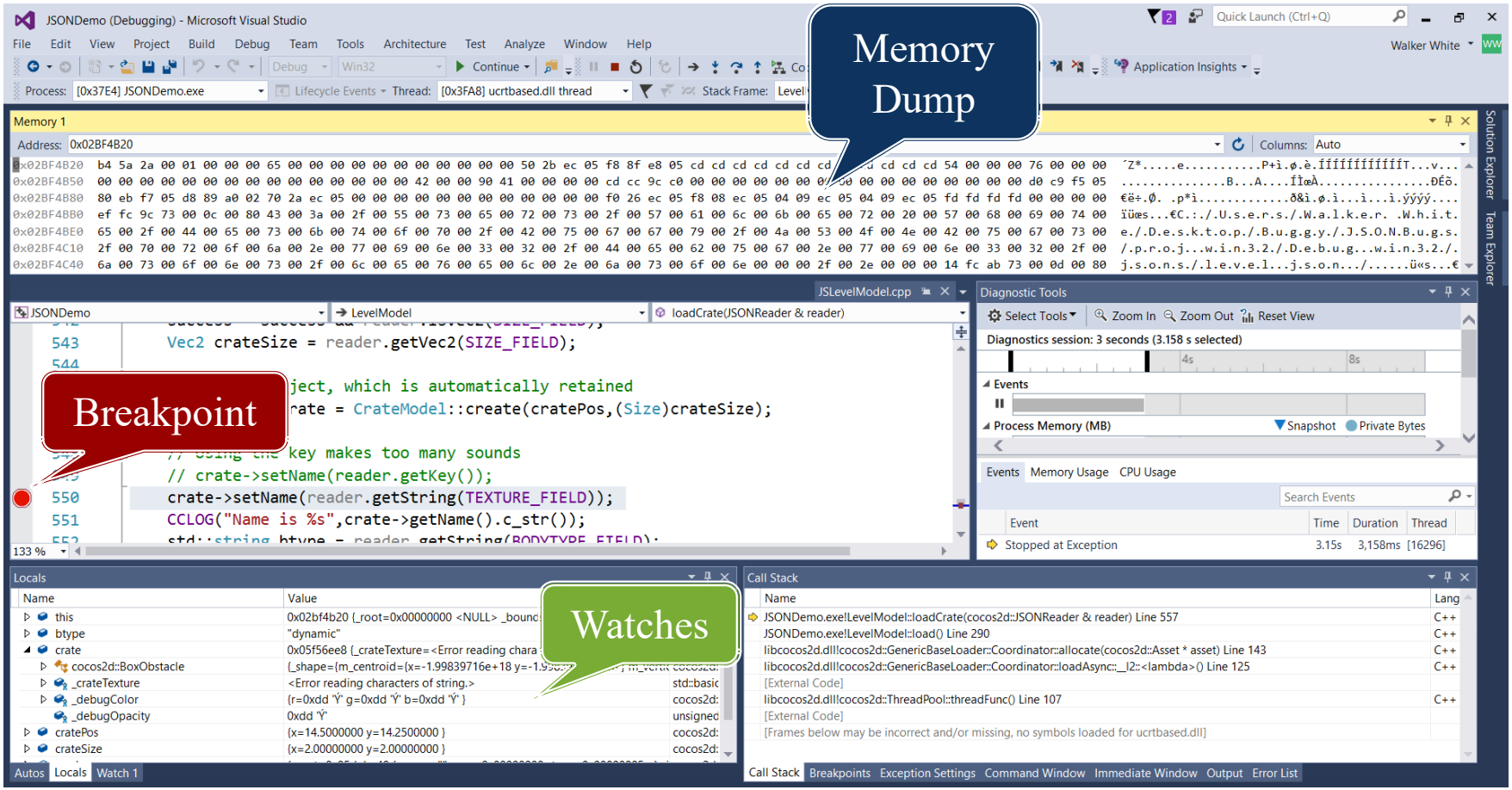
Visual Studio Tools



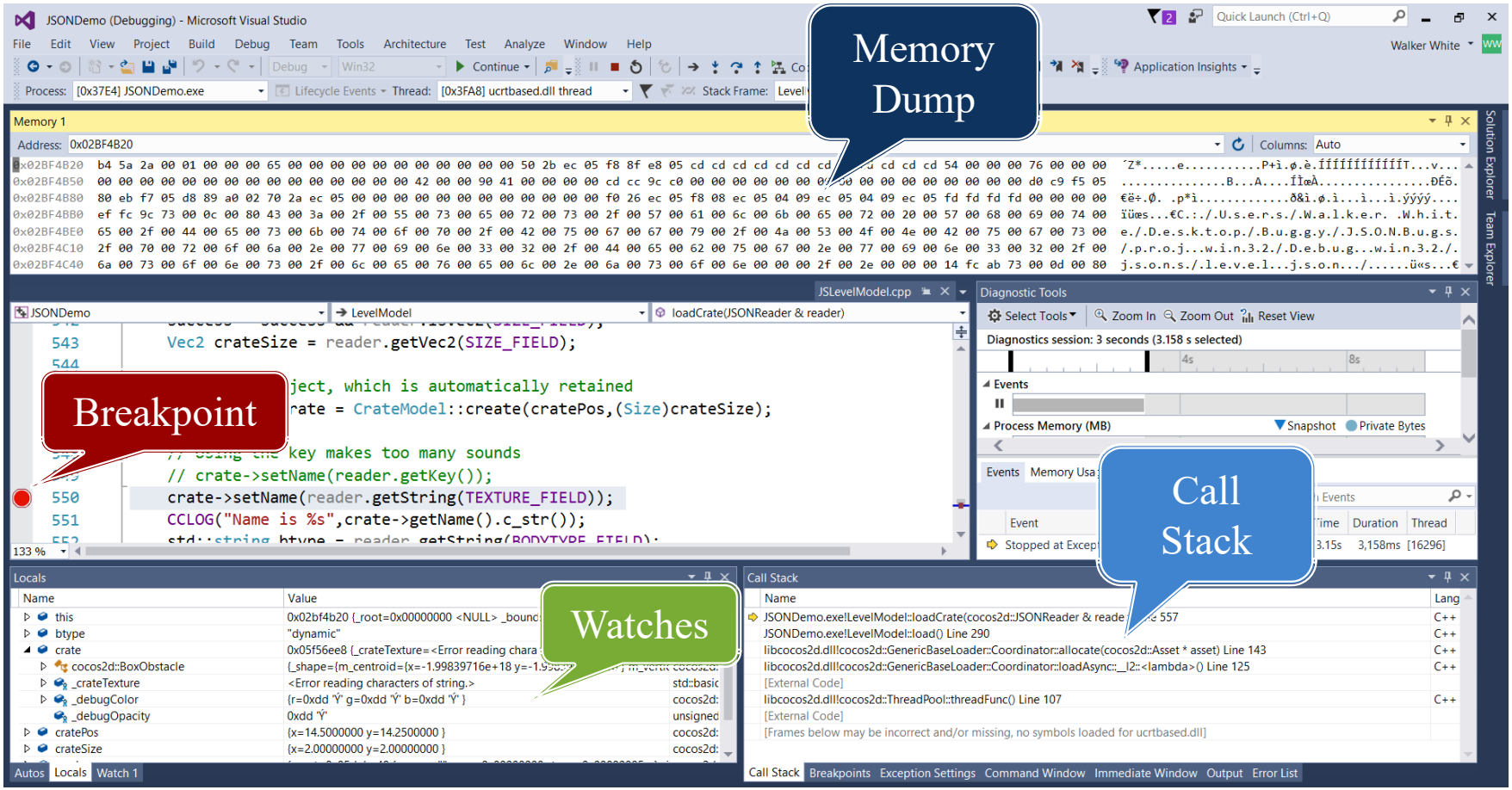
Visual Studio Tools



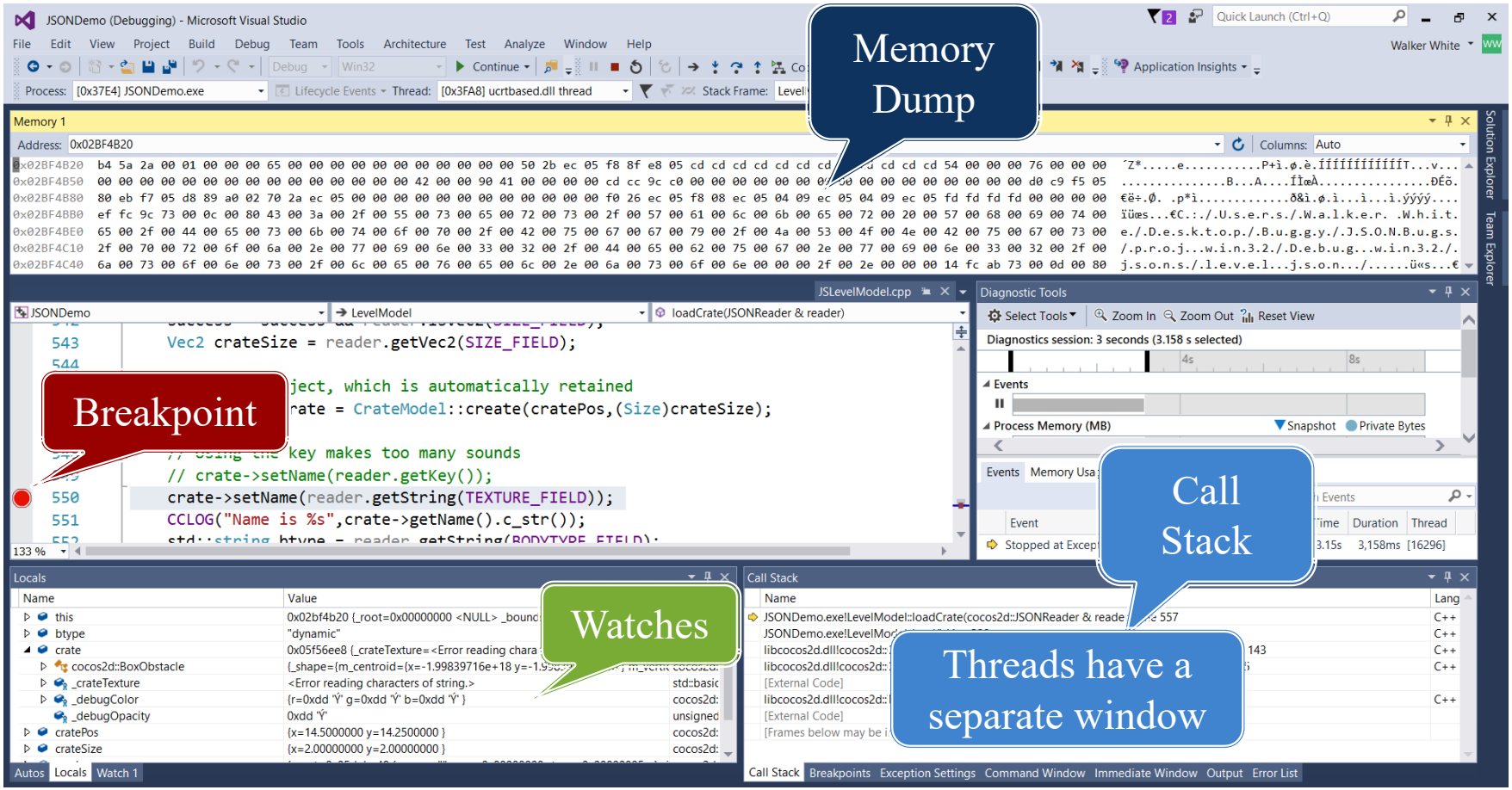
Visual Studio Tools



Visual Studio Tools



Visual Studio Tools



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

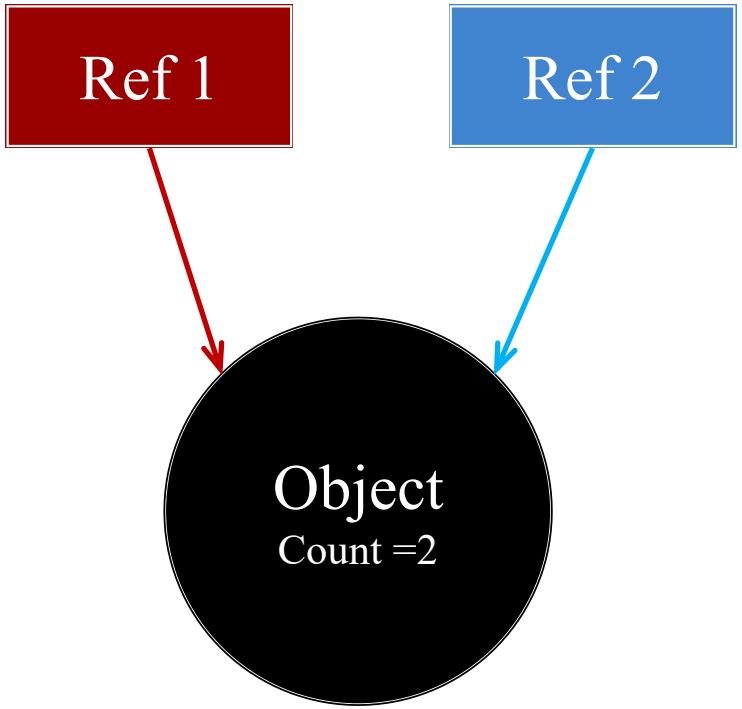
- Code stepping is not "thread safe"
  - Will never leave your current thread
  - Have to choose "continue" instead of "step"
- Makes it very difficult to find thread errors
  - May miss when a variable changes state
  - We had many problems in an old `AudioEngine`
- **Solution**: Rely heavily on assertions
  - 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();
node->retain();

// Add the node to scene graph
scene->addChild(node);

// Release the local reference
node->release();

// Remove from scene graph
scene->removeChild(node);
```

Raw pointers!!

Manual refence counts!!

Scene Graphs the Old Way

```
// create a new instance
Node* node = Node::create();
node->retain();

// Add the node to scene graph
scene->addChild(node);

// Release the local reference
node->release();

// Remove from scene graph
scene->removeChild(node);
```

Custom allocator

Reference count 1

Reference count 2

Reference count 1

Reference count 0

node is **deleted**

Scene Graphs the Old Way

```
// create a new instance
Node* node = Node::create();
node->retain();

// Add the node to scene graph
scene->addChild(node);

// Do not release the local reference

// Remove from scene graph
scene->removeChild(node);
```

Custom allocator

Reference count 1

Reference count 2

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