Wrangling Scotty3D

CMU 15-462

Goals Today

- Set up a C++ Environment
  - CLion, Xcode, Visual Studio
- Whirlwind crash course of C++, coming from C
- Debugging Multithreaded C++
  - Using Profilers to determine performance issues

But most of all to ***answer your questions!***

Getting Scotty3D Up and Running

The Basics

- Download the base code from
  `https://github.com/cmu462/Scotty3D`
- Click “Clone or Download”, then “Download Zip”
  - *Note: if you pulled the Scotty3D code before Monday evening, your base code is slightly out of date. More later…*
- On a Unix machine (Linux / Mac), you can compile from the command line directly. From Scotty3D root directory:

```
mkdir build
cd build
make .. && make
```

- Run with `./scotty3d ../media/<...>.dae`

Warning…

We will be grading your work on the Andrew Unix cluster.

If your code doesn't compile on Linux, or relies on undefined behavior that is different on the Linux machines, then you <u>will not</u> get credit!

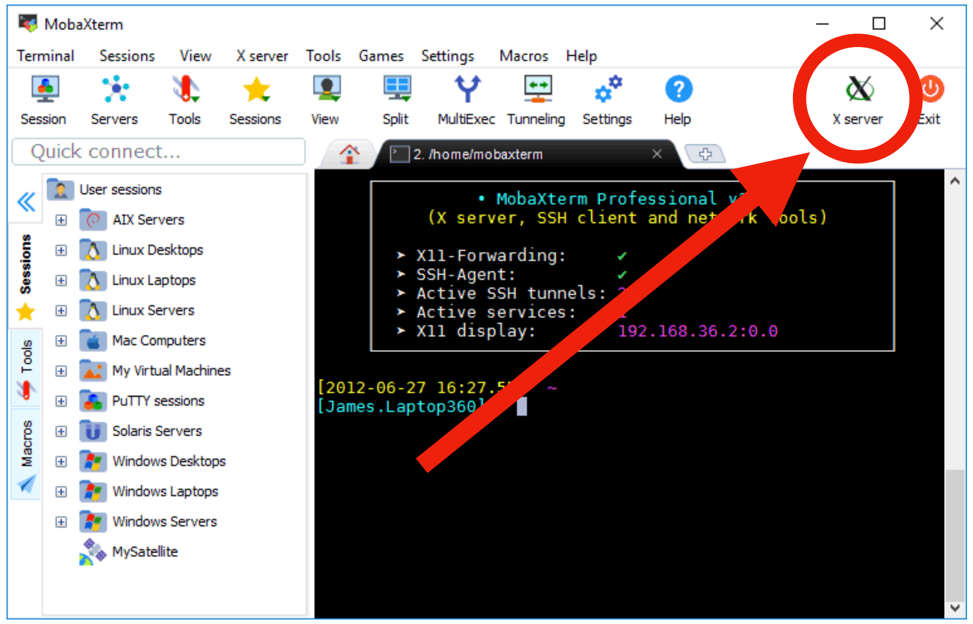
Make sure to test your code on the Linux machines well before the deadline. Word of Warning: the Visual Studio compiler is unusually lenient regarding non-standard C++

Using SSH with Andrew Unix Machines

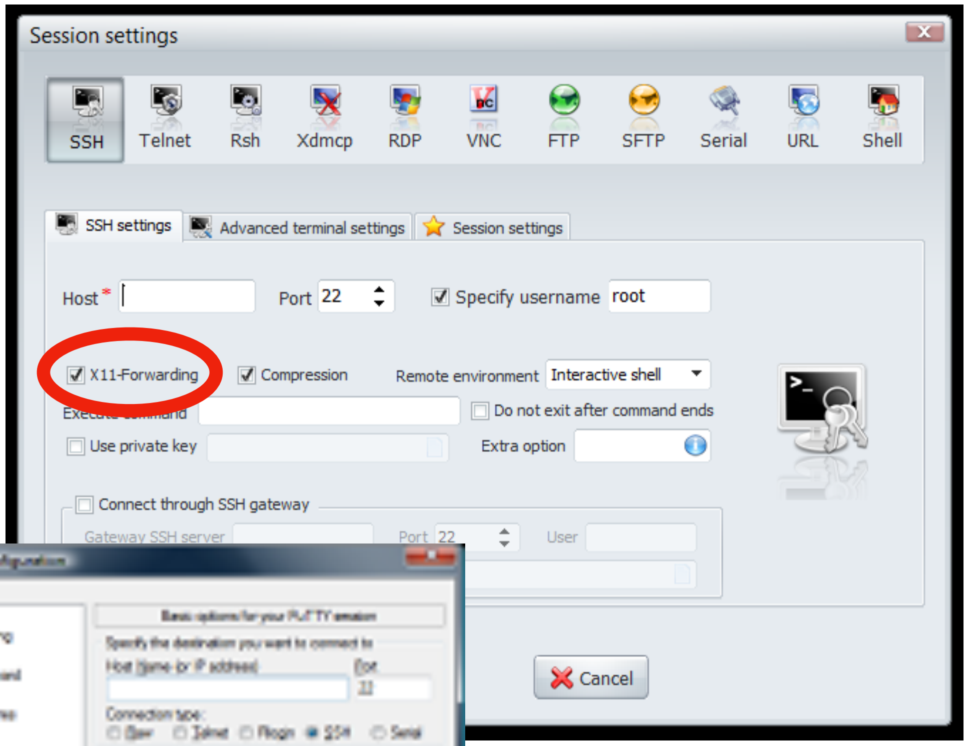
- If you try to test your code while connected to the Unix cluster via SSH, you will get an error. This is because by default SSH has no way of streaming Scotty3D to you.
- Solution: *X Forwarding*. ("X" is the window system common on Linux. One of X's features is that you can stream windows via SSH to other computers, with a bit of lag).
- MacOS/Linux: `ssh -Y andrewid@unix.andrew.cmu.edu`
  - On MacOS, you need to install XQuartz first (Mac implementation of X): https://www.xquartz.org

X Forwarding on Windows

- Windows has no native SSH client.
  - Option 1: Use MobaXterm to SSH in:
    - Download at https://mobaxterm.mobatek.net
    - Toggle on the X Server button and make sure "X11-Forwarding" is checked in your Session settings
  - Option 2: Use PuTTY to SSH in:
    - Download at https://putty.org
    - Guide: http://courses.cms.caltech.edu/cs11/misc/xwindows.html



```
  • MobaXterm Professional v
(X server, SSH client and net   k  ools)

➤ X11-Forwarding:     ✓
➤ SSH-Agent:          ✓
➤ Active SSH tunnels:
➤ Active services:
➤ X11 display:          192.168.36.2:0.0

[2012-06-27 16:27.5       ~
[James.Laptop360]
```


Session settings
SSH Telnet Rsh Xdmcp RDP VNC FTP SFTP Serial URL Shell
SSH settings
Advanced terminal settings
Session settings
Cancel

Host*

Port 22

☑ Specify username root

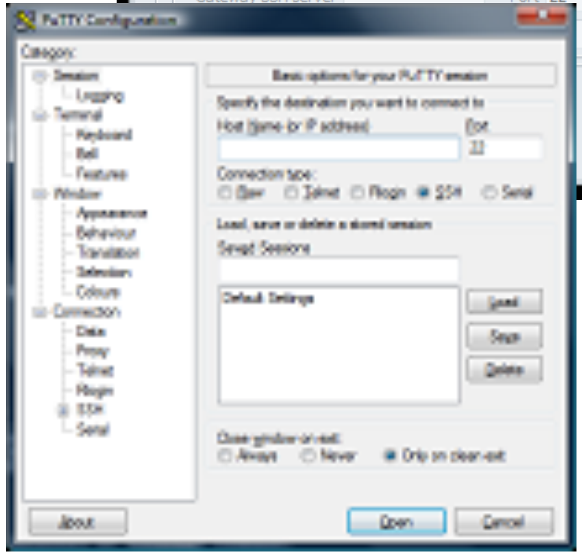
☑ X11-Forwarding ☑ Compression Remote environment Interactive shell

Execute command ☐ Do not exit after command ends

☐ Use private key Extra option

☐ Connect through SSH gateway

Gateway SSH server Port 22 User



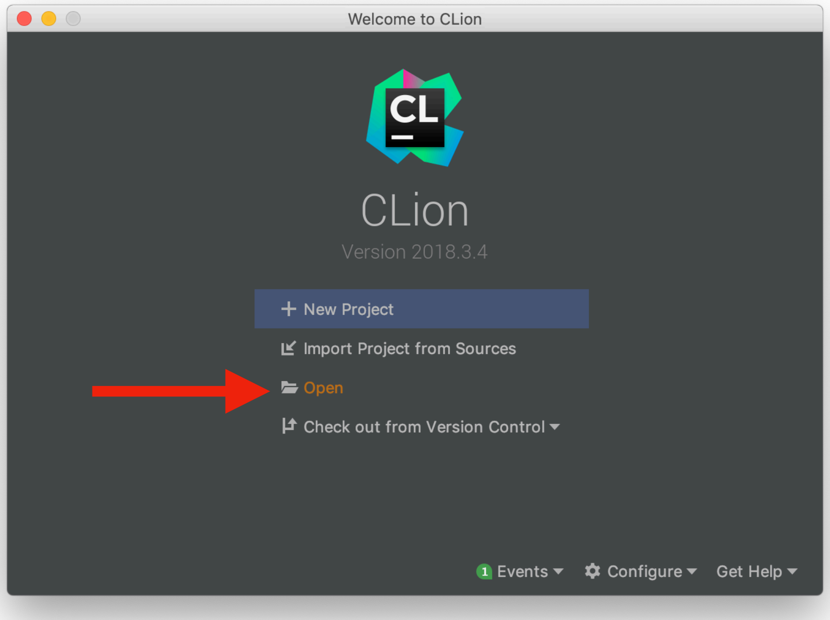
Setting up a C++ Development Environment

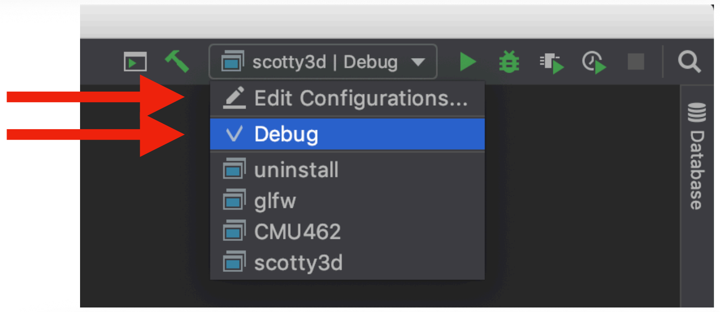
Why use an integrated development environment (IDE)?

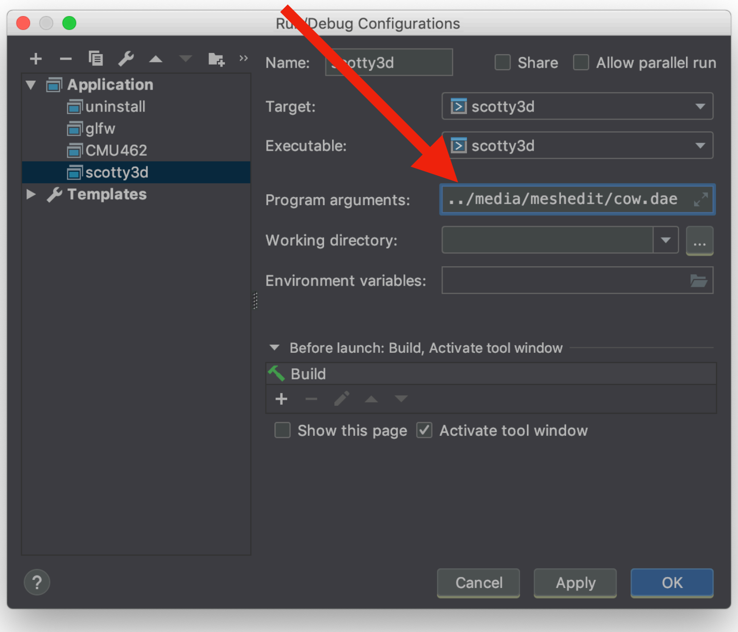
- **Autocomplete**: C++ code tends to have complicated class definitions with many member functions and variables. Autocompletion is essential so you don’t have to trawl through header files
- **Free Debugging**: In an IDE, when your code crashes the editor will jump to the line that had issues
- **Easier Navigation and Iteration**: Not having to jump in and out of vim, keyboard shortcuts, tabs, etc add up to speed your development time.
- Much more: Themes, intelligent syntax highlighting, linting (code formatting), tons of shortcuts to speed development

Windows/Mac/Linux: CLion

- Cross Platform IDE for C/C++ Development.
  - Free for students with a @andrew.cmu.edu email: https://www.jetbrains.com/student/
- Simply open the Scotty3D root directory, it should detect the CMake project out of the box
- To add command line arguments click on the configuration dropdown on the top right, then Click Debug. Then open it again and click "Edit Configurations..."







Name: scotty3d
Share
Allow parallel run
Target: scotty3d
Executable: scotty3d
Program arguments: ../media/meshedit/cow.dae
Working directory:
Environment variables:
Before launch: Build, Activate tool window
Build
Show this page ✓ Activate tool window

For Mac Users: XCode

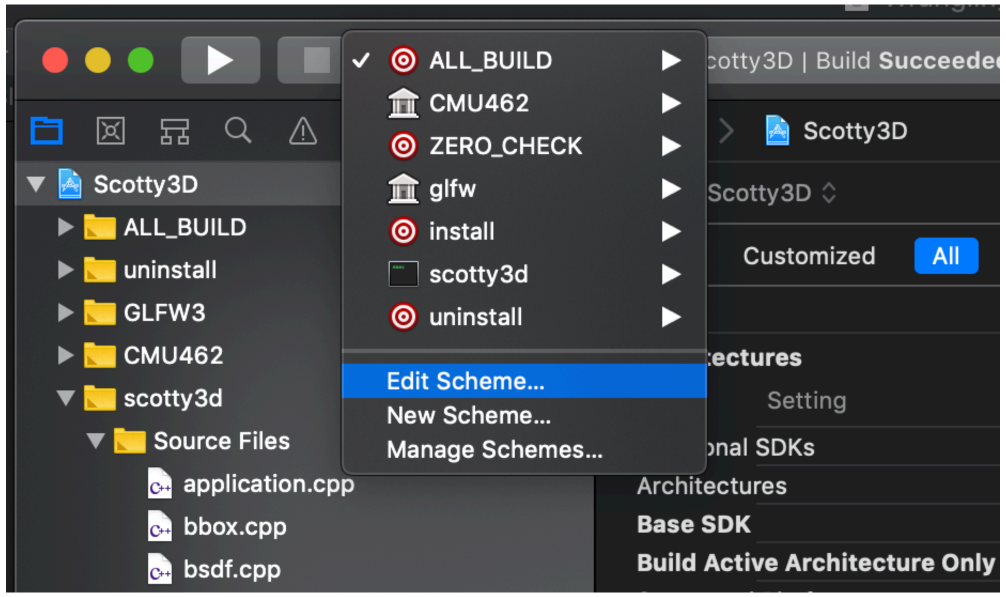
- Xcode is Apple’s IDE for the Mac. It supports C++, Objective-C, and Swift development, and has some editors specific to building Mac and iOS apps.
    - Download free from the Mac App Store (search “Xcode”)
- CMake can create Xcode projects for you out of the box.
    - From the root directory of Scotty3D:

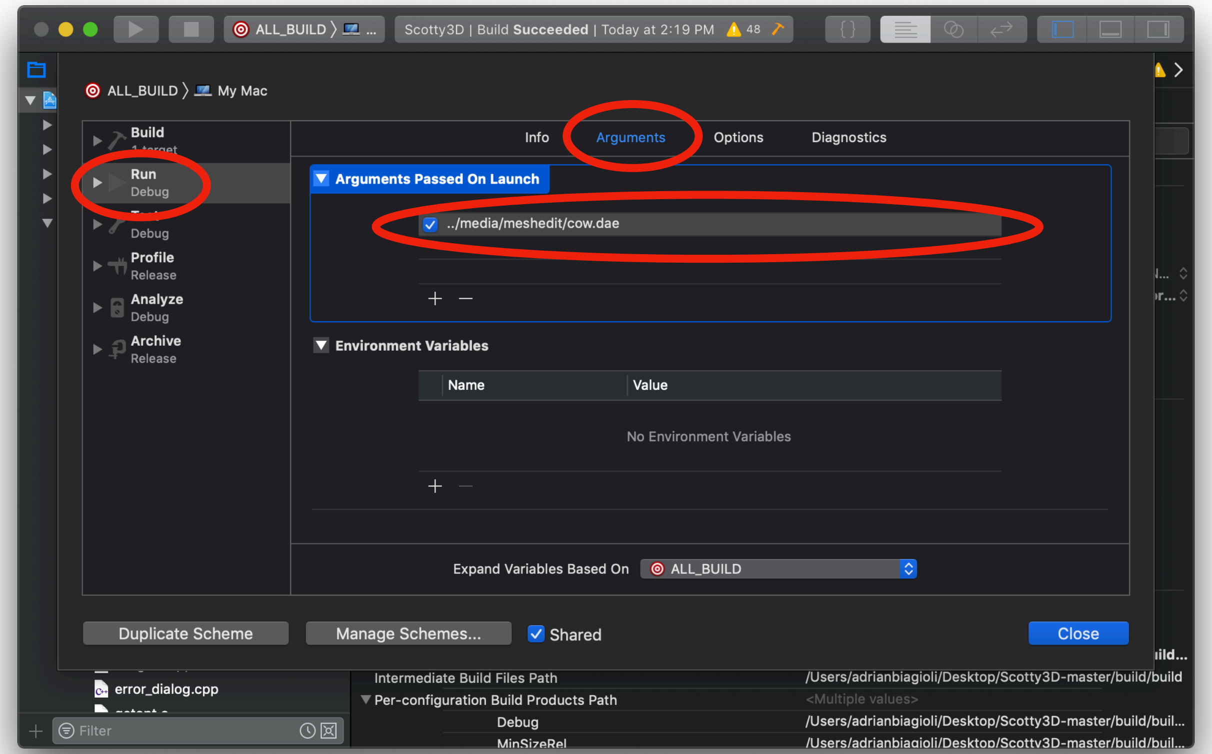
```
$ mkdir build
$ cd build
$ cmake -G Xcode ..
```

- Now double click on `Scotty3d.xcodeproj` to open in Xcode

For Mac Users: XCode

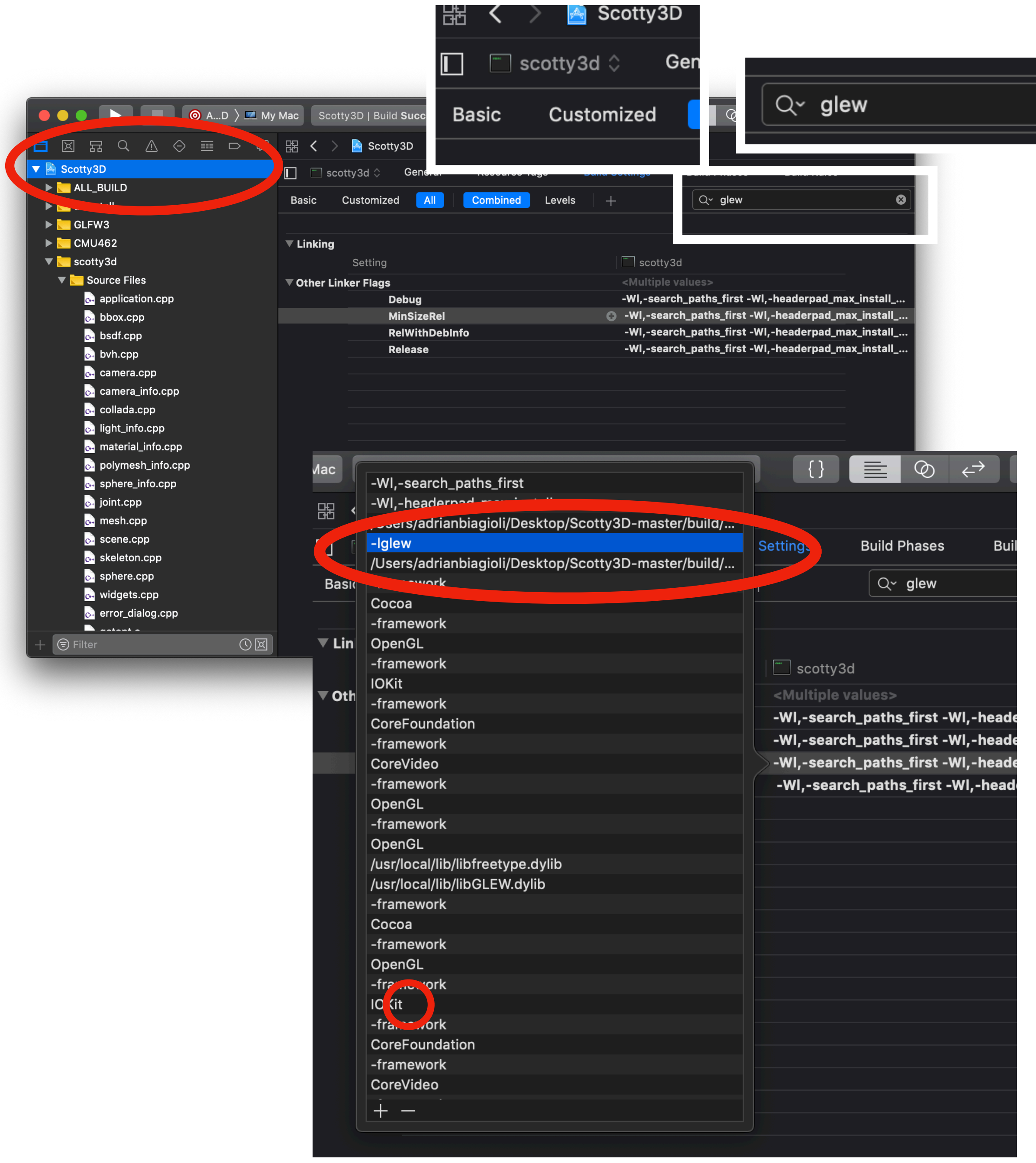
- To Set Command Line arguments:
  - Click on the "Scheme" ALL_BUILD at the top of the screen, then click "Edit Scheme"
  - Navigate to "Run" on the left, then "Arguments", then add the model you want to inspect, like "../media/meshedit/cow.dae"
  - Click "Close"
- To build, click on the Arrow on the top Left of the window.





For Mac Users: XCode

- There's a chance that Xcode can't link the project and gives you lip about the "GLEW" library. If this happens, remove the -lglew compiler flag from your build settings:
  - *Context*: GLEW is a support library for OpenGL that is built into MacOS as a system library.
  - Click on the project on the left side of the screen
  - Change the thing you are editing to "scotty3d" target
  - Search "glew"
  - Double click on each entry, find "-lglew", and delete it.



Linking

Setting	scotty3d
Other Linker Flags	<Multiple values>
Debug	-Wl,-search_paths_first -Wl,-headerpad_max_install_...
MinSizeRel	-Wl,-search_paths_first -Wl,-headerpad_max_install_...
RelWithDebInfo	-Wl,-search_paths_first -Wl,-headerpad_max_install_...
Release	-Wl,-search_paths_first -Wl,-headerpad_max_install_...

```
-Wl,-search_paths_first
-Wl,-headerpad_max_install...
/Users/adrianbiagioli/Desktop/Scotty3D-master/build/...
-lglew
/Users/adrianbiagioli/Desktop/Scotty3D-master/build/...
-framework
Cocoa
-framework
OpenGL
-framework
IOKit
-framework
CoreFoundation
-framework
CoreVideo
-framework
OpenGL
-framework
OpenGL
/usr/local/lib/libfreetype.dylib
/usr/local/lib/libGLEW.dylib
-framework
Cocoa
-framework
OpenGL
-framework
IOKit
-framework
CoreFoundation
-framework
CoreVideo
```

For Windows users: Visual Studio

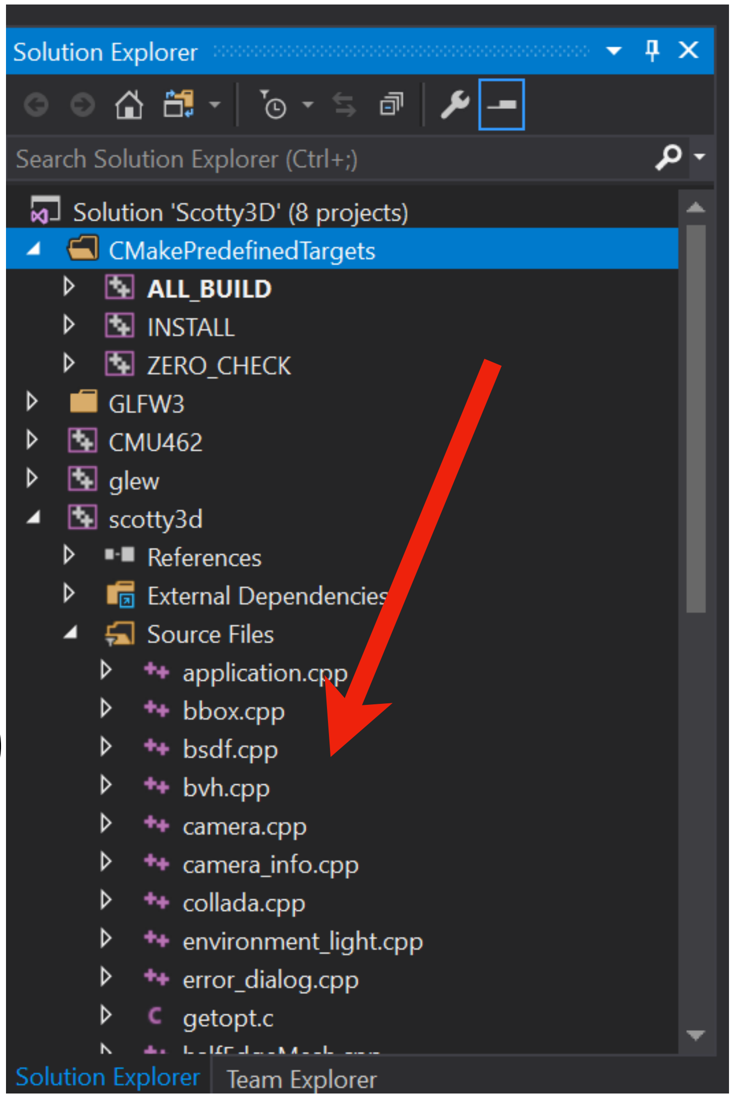
- Visual Studio is an industry-standard development environment for C++ and .NET (C# / F# / VB etc) code
  - “Windows version of XCode”
  - Download *Community Edition (Free)*: https://visualstudio.microsoft.com
    - Install "Desktop Development for C++" module
  - Note: *Visual Studio Code* is different — more similar to Sublime Text or Atom than an IDE. Don’t download this, it won’t work!
  - Don’t get the Mac version of VS—No C++ support on MacOS as of now

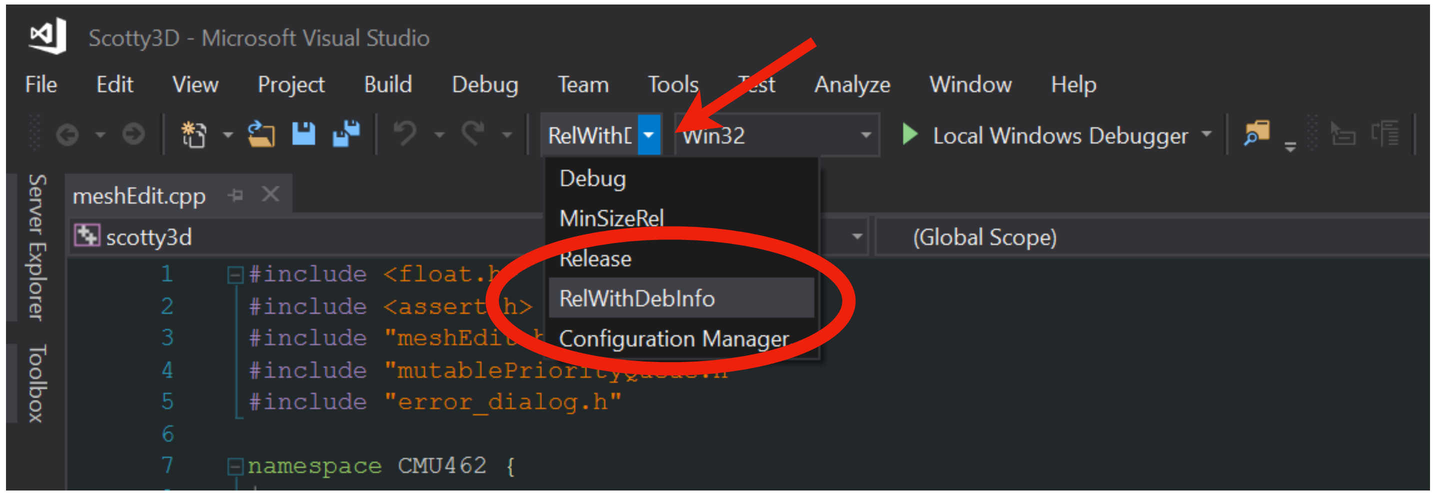
Setting up Visual Studio 2017 (Windows)

- Install the latest version of CMake on Windows: https://cmake.org/download/
  - Make sure you select "Add CMake to the System PATH for all users" during installation.
  - If you forgot to do this, see https://www.architectryan.com/2018/03/17/add-to-the-path-on-windows-10/
- Double click the file `runcmake_win.bat` at the root directory of Scotty3D
  - You should only need to do this once.
  - This will create a Visual Studio *solution* file for you

Setting up Visual Studio 2017 (Windows)

- Double click on `Scotty3D.sln` inside the `build` folder
    - Visual Studio will open with the project
- Navigate the files you need to edit via the Solution Explorer to the right
- Visual Studio should already have built-in autocomplete and debugging features! (Yay)
- Change the "Solution Configuration" from "Debug" to "RelWithDebInfo" or else Scotty3D will run slow as molasses

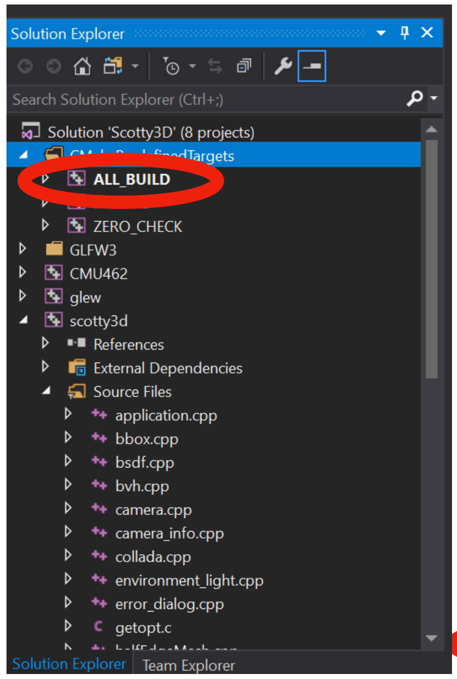


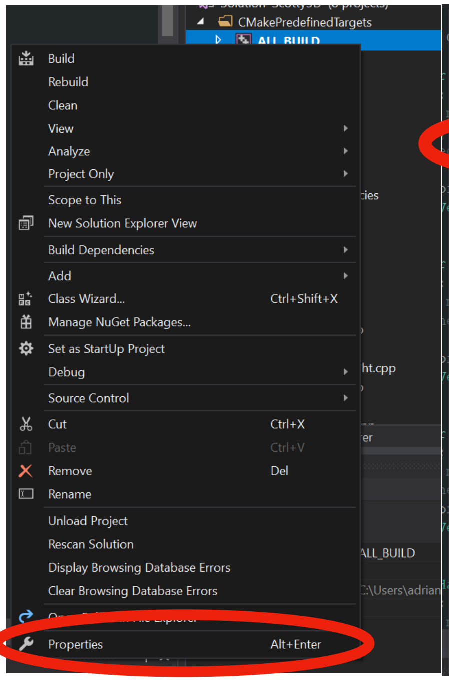


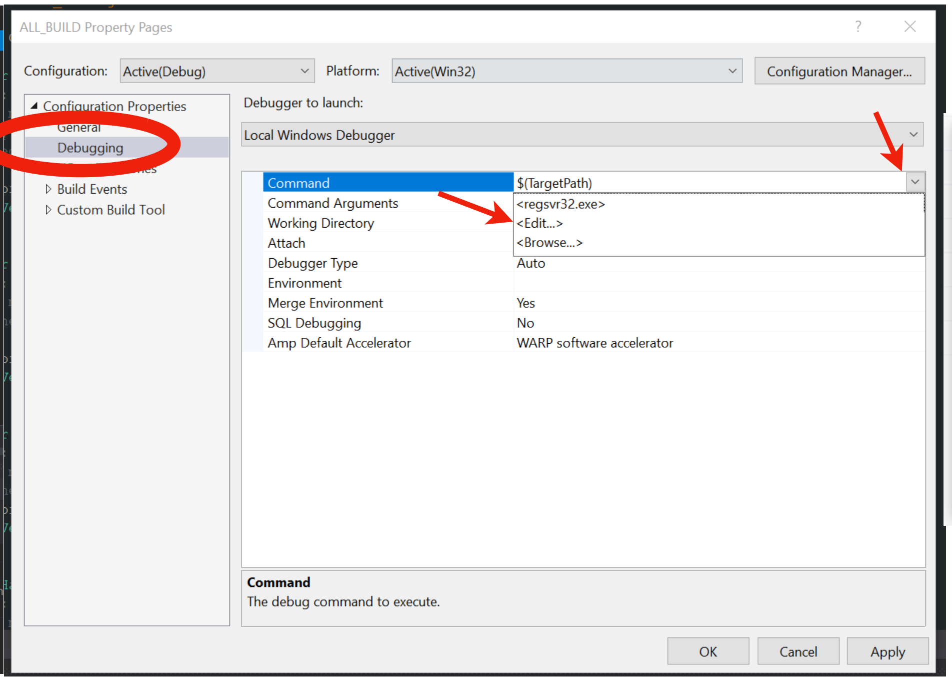
```
#include <float.h
#include <assert.h>
#include "meshEdit.h"
#include "mutablePriorityQueue.h"
#include "error_dialog.h"

namespace CMU462 {
```

Setting up Visual Studio 2017 (Windows)

- CMake workaround for VS2017:
  - Right click on the Target ALL_BUILD in the Solution explorer > Click Properties
  - Go to the Debugging tab
  - Open the "Command" Dropdown and click "<Edit...>"
  - Change from "`$(TargetPath)`" to "`$(TargetDir)\scotty3d.exe`"
  - Click OK > Click Apply
- There is a build error if you don't do this!





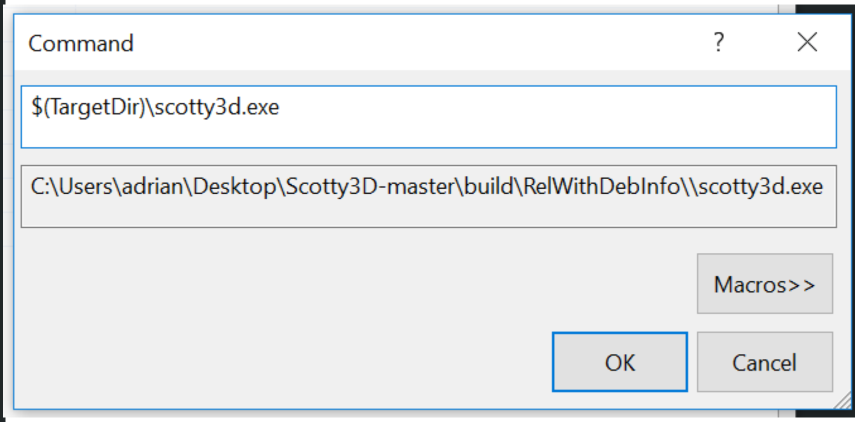


Configuration: Active(Debug) Platform: Active(Win32)

Debugger to launch: Local Windows Debugger

Command	$(TargetPath)
Command Arguments	<regsvr32.exe>
Working Directory	<Edit...>
Attach	<Browse...>
Debugger Type	Auto
Environment	
Merge Environment	Yes
SQL Debugging	No
Amp Default Accelerator	WARP software accelerator

Command
The debug command to execute.

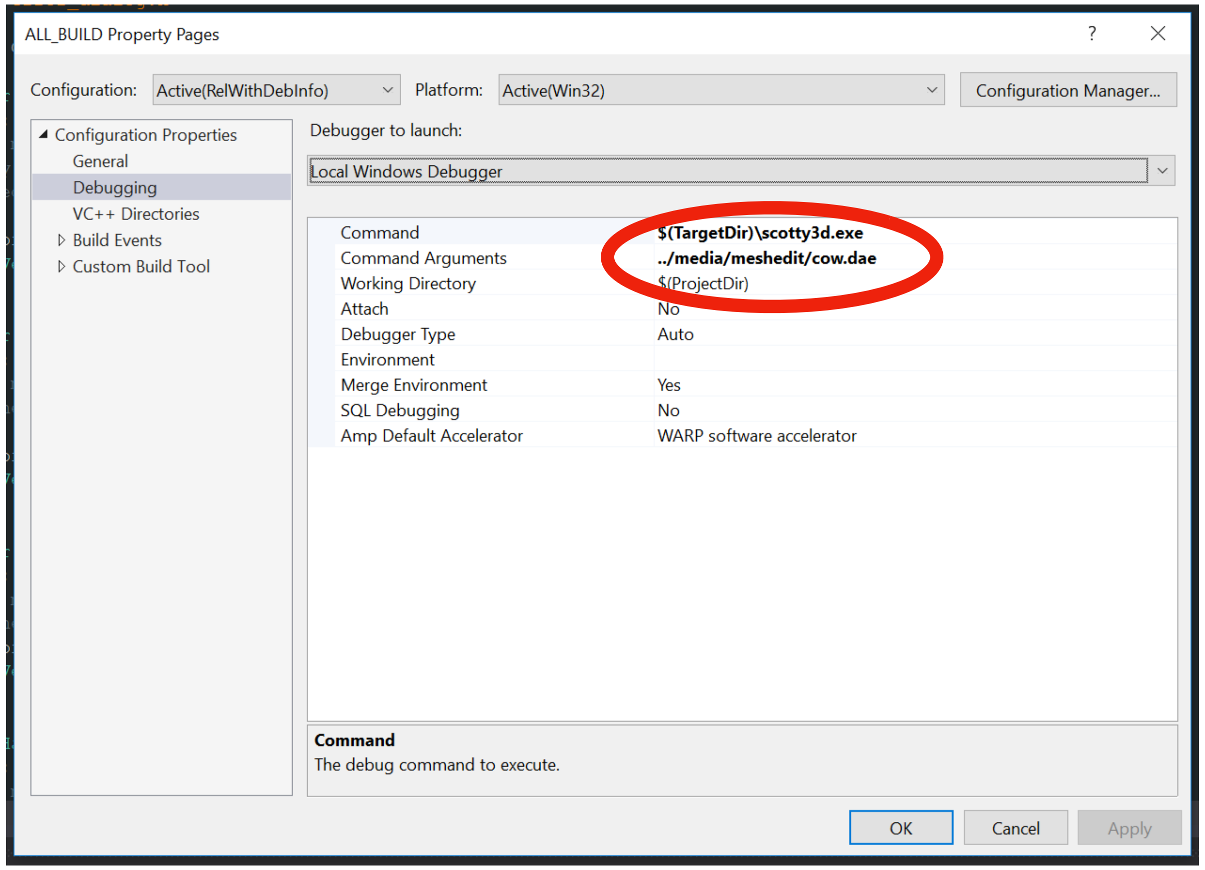


$(TargetDir)\scotty3d.exe

C:\Users\adrian\Desktop\Scotty3D-master\build\RelWithDebInfo\\scotty3d.exe

Setting up Visual Studio 2017 (Windows)

- To change command line arguments, right click on ALL_BUILD > Click Properties > Debugging tab
- Now you can edit "Command Line Arguments" in the form "`../media/meshedit/cow.dae`" (for example)
- To run hit F5 or navigate to Debug > Start Debugging



ALL_BUILD Property Pages

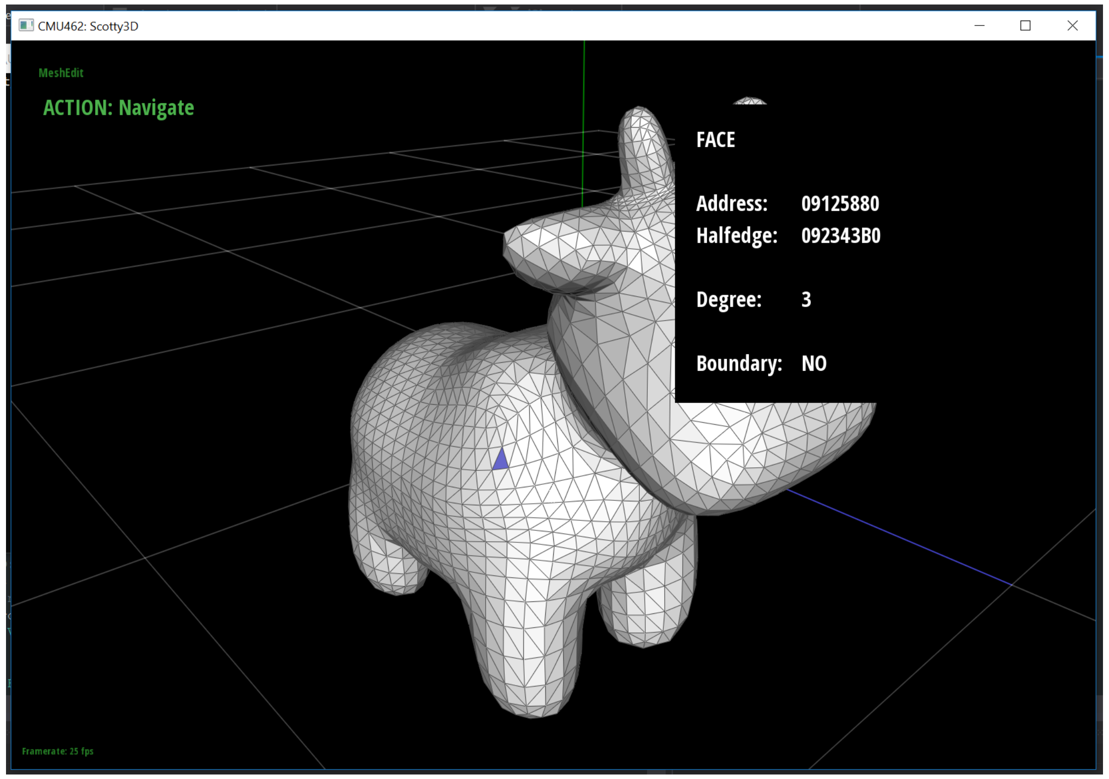
Configuration: Active(RelWithDebInfo) Platform: Active(Win32) Configuration Manager...

Configuration Properties
- General
- Debugging
- VC++ Directories
- Build Events
- Custom Build Tool

Debugger to launch: Local Windows Debugger

Command	$(TargetDir)\scotty3d.exe
Command Arguments	../media/meshedit/cow.dae
Working Directory	$(ProjectDir)
Attach	No
Debugger Type	Auto
Environment	
Merge Environment	Yes
SQL Debugging	No
Amp Default Accelerator	WARP software accelerator

Command
The debug command to execute.



C++ for C Programmers

The Basics

- C++ is a *superset* of C: <u>All C code is valid C++ Code</u>
    - (with few exceptions-but the basics all work correctly)
- Even though you *can* write C-style code, it is very hard to write scalable software without higher-level constructs
- Blog series about migrating from C to C++:
    - <u>https://ds9a.nl/articles/posts/c++-1/</u>

Major Differences

- Use new instead of malloc, delete instead of free:

```
// C style
int N = 42;
int *i = malloc(sizeof(int));
int *arr = malloc(
             sizeof(int) * N);
// ...
free(i);
free(arr);
```

```
// C++
int N = 42;
int *i = new int;
int *arr = new int[N];

// ...
delete i;
// delete[] for arrays
delete[] arr;
```

Major Differences

- C++ adds namespaces. Use the using keyword to bring a namespace "in-scope," or use the "scope resolution operator" (::) to zoom in to a namespace
  - Motivation: now you can have 2 functions (potentially from 2 different libraries) with the same name
  - Note, we are ignoring the different #includes here

```
// C style
int cmp(const void* a,
   const void* b) { /* ... */ }
int *arr; // initialized
int arrlen;

qsort(i, arrlen,
      sizeof(int), &cmp);
```

```
// C++
int *arr; // initialized
int arrlen;

std::sort(arr, arr + arrlen);

OR

using std;
sort(arr, arr + arrlen);
```

Major Differences

- Strings: use std::string instead of char * arrays.
- Console output: Use std::cout and the "stream operator" (<<) (new in C++). No more memorizing printf keywords!

```
// C style
int i = 15462;
char *str = "Hello, World.";
// segfault! read only memory
// str[12] = '!';

printf("%d It's %s.\n", i,
str);
// Prints:
// Hello, World. It's 15462.
```

```
// C++
int i = 15462;
std::string str =
            "Hello, World."
str[12] = '!'; // fine!

std::cout << str << " It's "
          << i << ‘\n';
// Prints:
// Hello, World! It's 15462
```

Major Differences

- **By the way: You can still use printf if you want! Convert to a C-style string using** `std::string::c_str()`

```
// C++
int i = 15462;
std::string str = "Hello, World."
str[12] = '!'; // fine!

printf("Hey! %s\n", str.c_str());
```

Major Differences

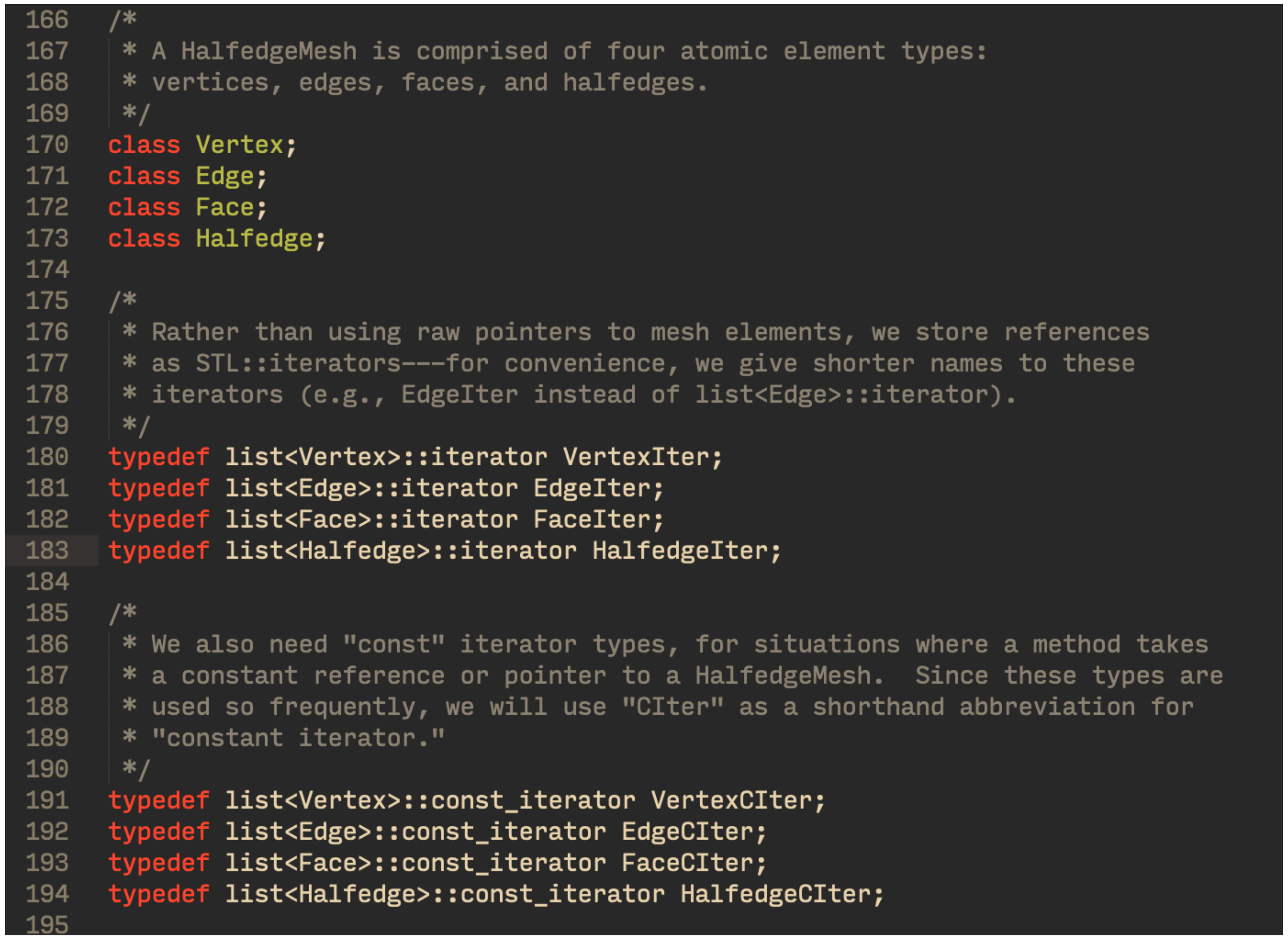
- *Classes* allow you to modularize your code much more effectively
  (source for below: *https://ds9a.nl/articles/posts/cpp-2/*)

```
// C style
struct Circle
{
     int x, y;
     int size;
     Canvas* canvas;
     ...
};

void setCanvas(Circle* circle, Canvas* canvas);
void positionCircle(Circle* circle,
                          int x, int y);
void paintCircle(Circle* circle);
```

```
// C++
class Circle
{
public:
     Circle(Canvas* canvas);  // "constructor"
     void position(int x, int y);
     void paint();
private:
     int d_x, d_y;
     int d_size;
     Canvas* d_canvas;
};

void Circle::paint()
{
     d_canvas->drawCircle(d_x, d_y, d_size);
}
```

Another Example: Iterators in Scotty3D

- From halfEdgeMesh.h:
    - Uses an std::list (a linked list) iterator (which itself is an std::iterator) to “simulate” pointers
    - https://en.cppreference.com/w/cpp/iterator/iterator
    - https://en.cppreference.com/w/cpp/container/list
    - Because of *operator overloading* in C++, “dereferencing” an iterator with * works as you would expect.

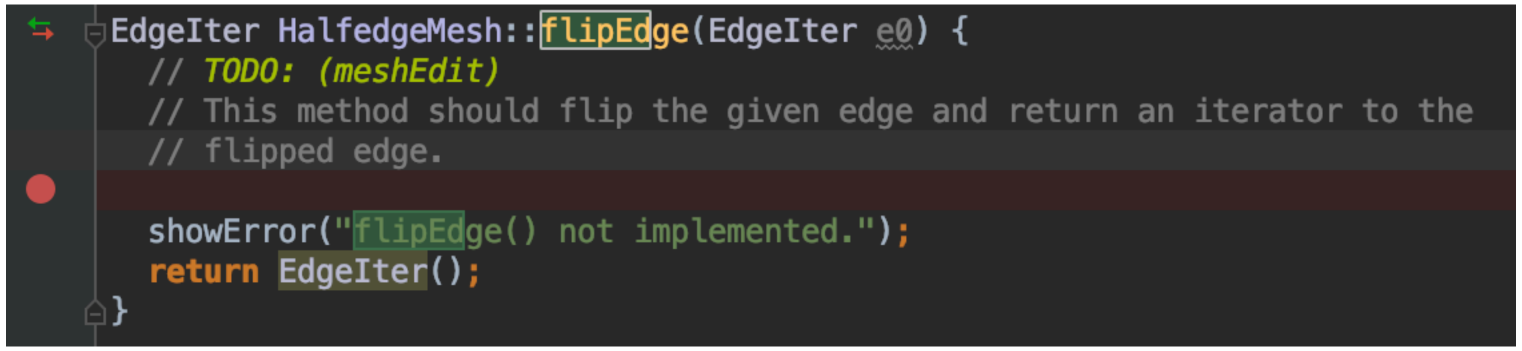


```
/*
 * A HalfedgeMesh is comprised of four atomic element types:
 * vertices, edges, faces, and halfedges.
 */
class Vertex;
class Edge;
class Face;
class Halfedge;

/*
 * Rather than using raw pointers to mesh elements, we store references
 * as STL::iterators---for convenience, we give shorter names to these
 * iterators (e.g., EdgeIter instead of list<Edge>::iterator).
 */
typedef list<Vertex>::iterator VertexIter;
typedef list<Edge>::iterator EdgeIter;
typedef list<Face>::iterator FaceIter;
typedef list<Halfedge>::iterator HalfedgeIter;

/*
 * We also need "const" iterator types, for situations where a method takes
 * a constant reference or pointer to a HalfedgeMesh.  Since these types are
 * used so frequently, we will use "CIter" as a shorthand abbreviation for
 * "constant iterator."
 */
typedef list<Vertex>::const_iterator VertexCIter;
typedef list<Edge>::const_iterator EdgeCIter;
typedef list<Face>::const_iterator FaceCIter;
typedef list<Halfedge>::const_iterator HalfedgeCIter;

```

Debugging / Profiling C++ Code

Setting Breakpoints

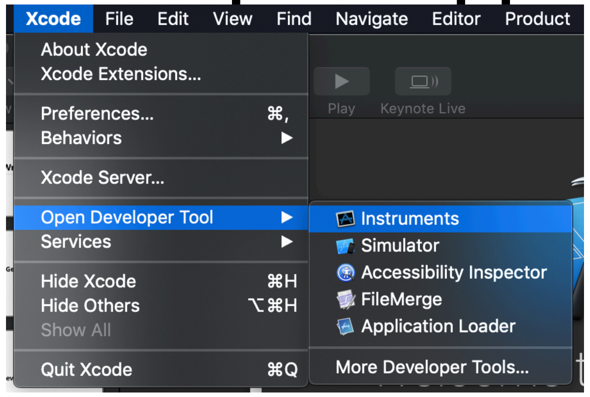
- In all three of the editors discussed today, you can set breakpoints in your code. When Scotty3D is run and the breakpoint is hit, the application will pause and you can step through the code / all variables in your IDE.
    - Not enough time today to go through every feature of every IDE — but there are plenty of resource online!
    - In all three editors we talked about, you can set a breakpoint by clicking to the left of each line.



```
EdgeIter HalfedgeMesh::flipEdge(EdgeIter e0) {
  // TODO: (meshEdit)
  // This method should flip the given edge and return an iterator to the
  // flipped edge.

  showError("flipEdge() not implemented.");
  return EdgeIter();
}
```

Profiling: Finding performance issues

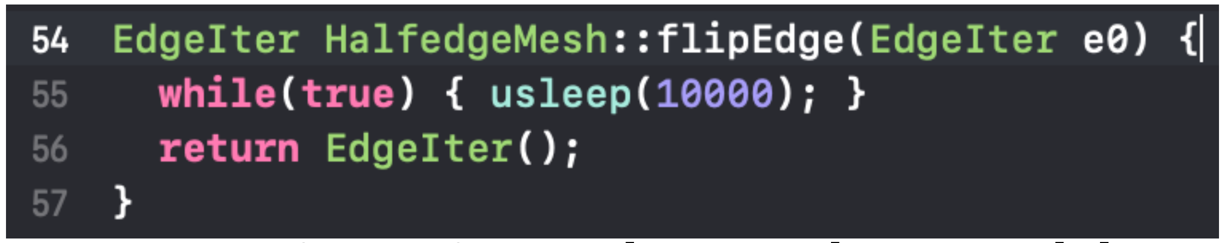
- A profiler is used to determine what functions are taking the most time to execute in your program.
  - We don't grade for performance in this class, but profilers can be very useful to determine where an infinite loop is happening.
- To profile with XCode, use the "Instruments" companion app (which should be installed alongside XCode)



  - Apple Talk "Using time profiler with Instruments:" https://developer.apple.com/videos/play/wwdc2016/418/
- Visual Studio Profiler Documentation: https://docs.microsoft.com/en-us/visualstudio/profiling
- CLion Profiler Documentation: https://www.jetbrains.com/help/clion/cpu-profiler.html

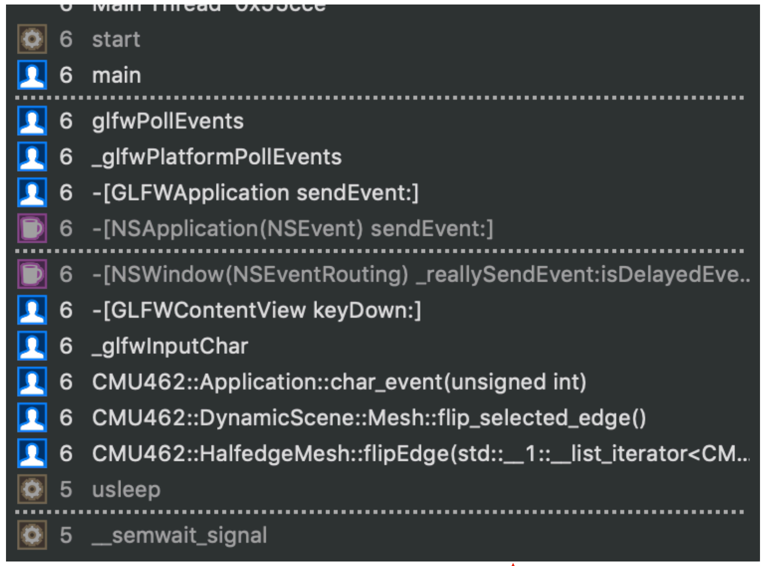
Example: Using MacOS Instruments to find an infinite loop

- I was naughty and added an infinite loop to my code:

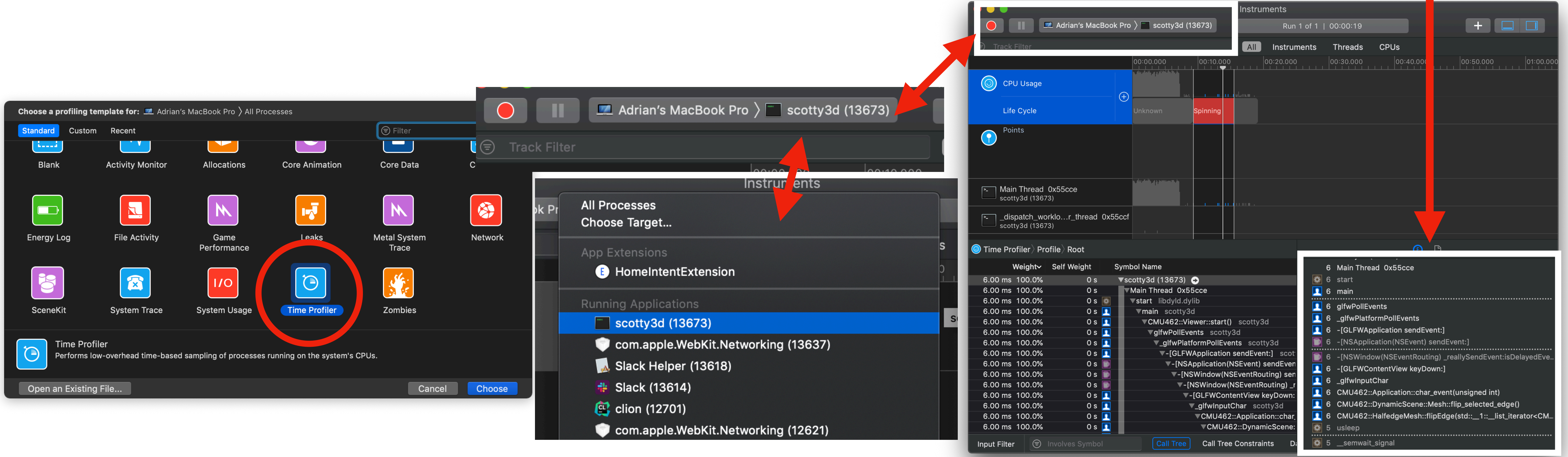


```
EdgeIter HalfedgeMesh::flipEdge(EdgeIter e0) {
  while(true) { usleep(10000); }
  return EdgeIter();
}
```

- I want to pinpoint where the problem is. While running the program, I'll open Instruments and select "Time Profiler". Then I can select scotty3d from the list of running programs, and I can clearly see the problem in the heaviest stack trace:



```
6  start
6  main
6  glfwPollEvents
6  _glfwPlatformPollEvents
6  -[GLFWApplication sendEvent:]
6  -[NSApplication(NSEvent) sendEvent:]
6  -[NSWindow(NSEventRouting) _reallySendEvent:isDelayedEve..
6  -[GLFWContentView keyDown:]
6  _glfwInputChar
6  CMU462::Application::char_event(unsigned int)
6  CMU462::DynamicScene::Mesh::flip_selected_edge()
6  CMU462::HalfedgeMesh::flipEdge(std::__1::__list_iterator<CM..
5  usleep
5  __semwait_signal
```



Time Profiler
Performs low-overhead time-based sampling of processes running on the system's CPUs.

```
Weight            Self Weight  Symbol Name
6.00 ms  100.0%   0 s          ▼scotty3d (13673)
6.00 ms  100.0%   0 s           ▼Main Thread  0x55cce
6.00 ms  100.0%   0 s            ▼start  libdyld.dylib
6.00 ms  100.0%   0 s             ▼main  scotty3d
6.00 ms  100.0%   0 s              ▼CMU462::Viewer::start()  scotty3d
6.00 ms  100.0%   0 s               ▼glfwPollEvents  scotty3d
6.00 ms  100.0%   0 s                ▼_glfwPlatformPollEvents  scotty3d
6.00 ms  100.0%   0 s                 ▼-[GLFWApplication sendEvent:]  scot
6.00 ms  100.0%   0 s                  ▼-[NSApplication(NSEvent) sendEven
6.00 ms  100.0%   0 s                   ▼-[NSWindow(NSEventRouting) sen
6.00 ms  100.0%   0 s                    ▼-[NSWindow(NSEventRouting) _r
6.00 ms  100.0%   0 s                     ▼-[GLFWContentView keyDown:
6.00 ms  100.0%   0 s                      ▼_glfwInputChar  scotty3d
6.00 ms  100.0%   0 s                       ▼CMU462::Application::char_
6.00 ms  100.0%   0 s                        ▼CMU462::DynamicScene:
```

```
6  Main Thread  0x55cce
6  start
6  main
6  glfwPollEvents
6  _glfwPlatformPollEvents
6  -[GLFWApplication sendEvent:]
6  -[NSApplication(NSEvent) sendEvent:]
6  -[NSWindow(NSEventRouting) _reallySendEvent:isDelayedEve..
6  -[GLFWContentView keyDown:]
6  _glfwInputChar
6  CMU462::Application::char_event(unsigned int)
6  CMU462::DynamicScene::Mesh::flip_selected_edge()
6  CMU462::HalfedgeMesh::flipEdge(std::__1::__list_iterator<CM..
5  usleep
5  __semwait_signal
```

Scotty3D Base Code Update

- If you pulled Scotty3D Base code before Monday evening, you are missing some important updates to Scotty3D's UI.
- meshEdit.cpp was not affected, so you can safely replace every other file with the updated ones on Github.
- See this link for details on what was changed: https://github.com/cmu462/Scotty3D/commit/e399f6098c6afd51512ca3a3f9452a9c28b250fa
- If you're having issues, make a Piazza post