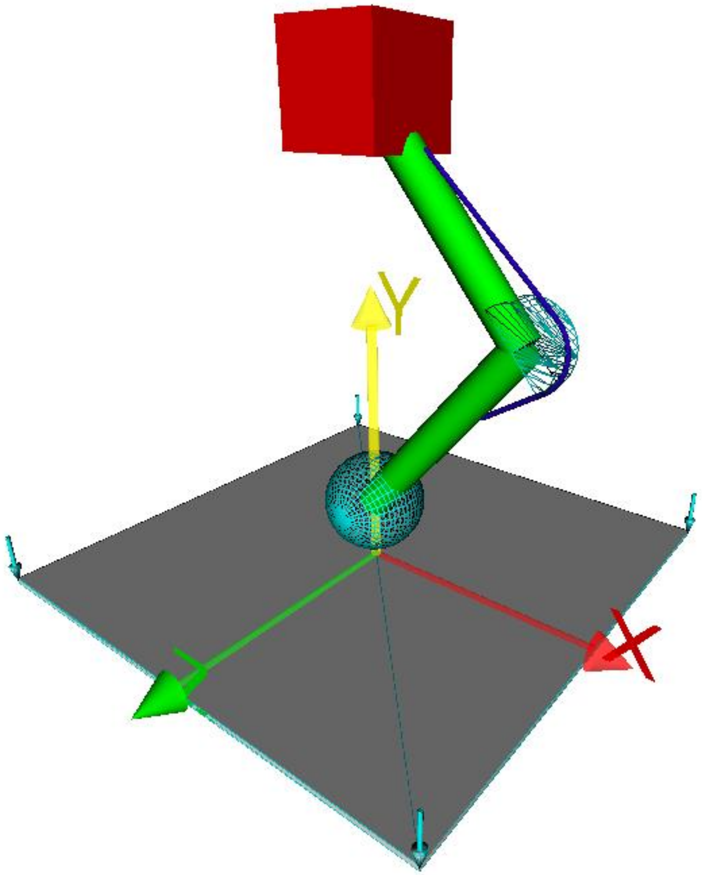
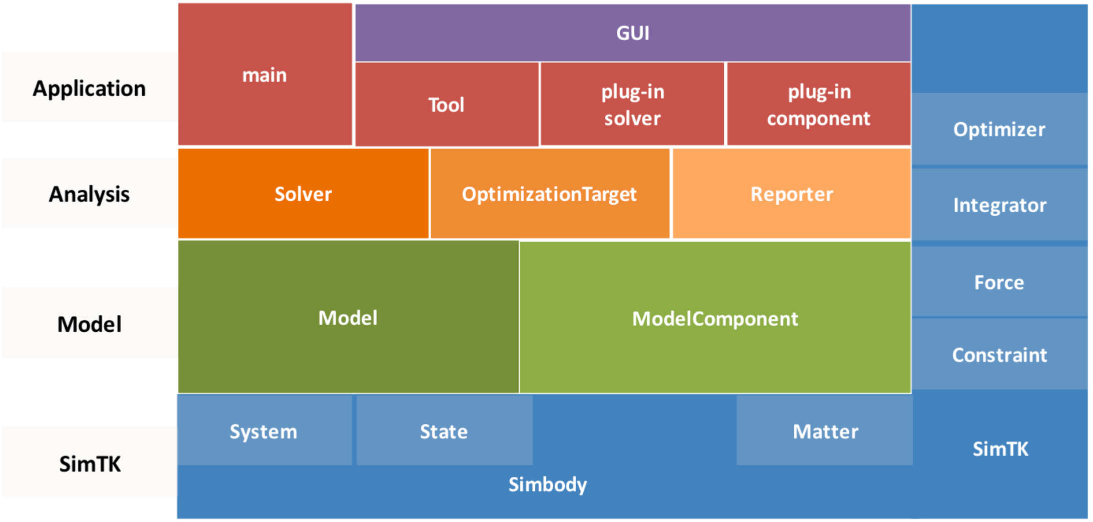
Working with OpenSim API

Modeling and Simulation of a Single-Legged Hopping Mechanism

- Part A
  - Bottom up modeling
- Part B
  - Simulation
  - Optimization
  - Plugins



OpenSim API



Application

main

GUI

Tool

plug-in solver

plug-in component

Analysis

Solver

OptimizationTarget

Reporter

Model

Model

ModelComponent

SimTK

System

State

Matter

Simbody

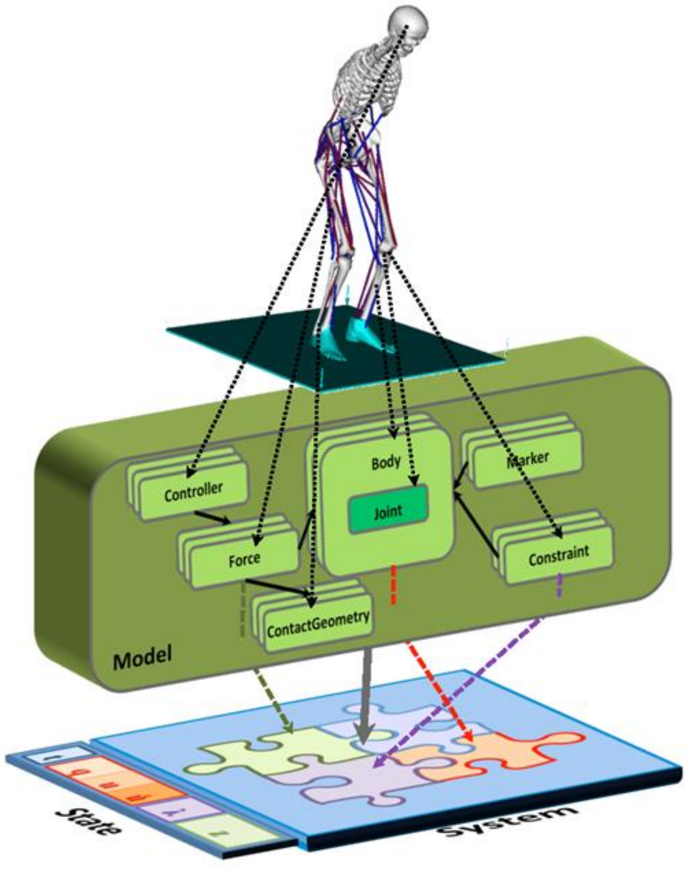
Optimizer

Integrator

Force

Constraint

SimTK



C++ development process

- A simple program
- Control structures
- Constants, variables, pointers
- Functions, pass by value, pass by reference
- Separation between header and source files
- Classes, encapsulation, inheritance, message passing, instantiation
- Compilation, linking, dynamic linking
- CMake
- Visual Sutdio

Simple Program

```
// c style
#include <stdio.h>

// c++ style
#include <iostream>
using namespace std;

int main()
{
    // c style
    printf("Hello\n");

    // c++ style
    cout << "Hello" << std::endl;

    return 1;
}
```

Control Structures

```
bool isPrime = false;
if (isPrime == true) {
    cout << "Is prime" << endl;
}else {
    cout << "Not prime" << endl;
}

for (int i = 0; i < 10; i++){
    cout << i << endl;
}

while(true){
    cout << "Infinite loop" << endl;
}
```

Constants, Variables and Pointers

```
// c style
#define PI 3.1415926

// c++ style
const double PI = 3.1415926;

// pointer
int a = 3; // variable of type int with value of 3
int* p = nullptr; // p is a pointer to an int (or = NULL, = 0)
int* p1 = &a; // p1 is a pointer to an int (and points to the address of a)

cout << p1 << endl; // address of a e.g. 0xFFFFFFFA
cout << *p1 << endl; // pointer dereference 3

int foo[5] = {16, 2, 77, 40, 12071};
int foo[] = {16, 2, 77, 40, 12071};

cout << foo[0] << endl; // zero based indexing, 0
cout << &foo[0] << endl; // the address of the first element, e.g. 0xFFFFFFFA
```

http://www.cplusplus.com/doc/tutorial/pointers/

Functions

```
// ------- simple function call
double distSquared(double x, double y){ // x, y are copied to stack (pass by value)
    return (x - y) * (x - y);
}
cout << distSquared(3, 0) << endl; // 9

// ------- example pass by reference
double distSquared(double* x, double& y){ // pass by reference
   *x += 1; // x = x + 1
   return (*x - y) * (*x - y);
}

int x = 3, y = 0;
cout << distSquared(3, 0) << endl; // 9
cout << x << endl; // 4 (changed!!!)

// ------- return multiple outputs and pass by reference to improve execution time
void calcSomething(const std::vector<double>& in_array,
    std::vector<double>& out_array1, std::vector<double>& out_array2){
    in_array[3] = 5.0; // compile time error because in_array is const
    ...
}
std::vector<double> in_array, result1, result2;
in_array = ... // set values
calcSomething(in_array, result1, result2);
```

Header and Source

util.h

```
#ifndef UTIL_H
#define UTIL_H

void f1(double x);
int f2(double x, double y);

#endif
```

main.cpp

```
#include <iostream>
using namespace std;
#include "util.h"

int main(){
    cout << f1(3) << endl;
    cout << f2(1, 2) << endl;
    return 1;
}
```

util.cpp

```
#include "util.h"

void f1(double x)
{
    return x + 1;
}
int f2(double x, double y)
{
    return atan2(x, y);
}
```

Classes 1/3 Encapsulation

person.h

```
#ifndef PERSON_H
#define PERSON_H

#include <string>

class Person{
public:
    Person(std::string name, int age);
    int getAge();
private:
    std::string name;
    int age;
}

#endif
```

person.cpp

```
#include "person.h"

Person::Person(std::string name, int age) : name(name) {
    age = age; // two ways to initialize private
variables
}

int Person::getAge(){
    return age - 5; // make someone appear younger
}
```

Classes 2/3 Inheritance

worker.h

```
#ifndef WORKER_H
#define WORKER_H

#include <string>
#include "person.h"

class Worker : public Person{
public:
    Worker(std::string name, int age, double salary);
    double tellMeYourSalary();
private:
    double salary;
}

#endif
```

worker.cpp

```
#include "worker.h"

Worker::Worker(std::string name, int age, double salary)
         : Person(name, age), salary(salary) {

}

double Worker::tellMeYourSalary() {
    return salary;
}
```

Classes 3/3 (Instantiation – Message Passing)

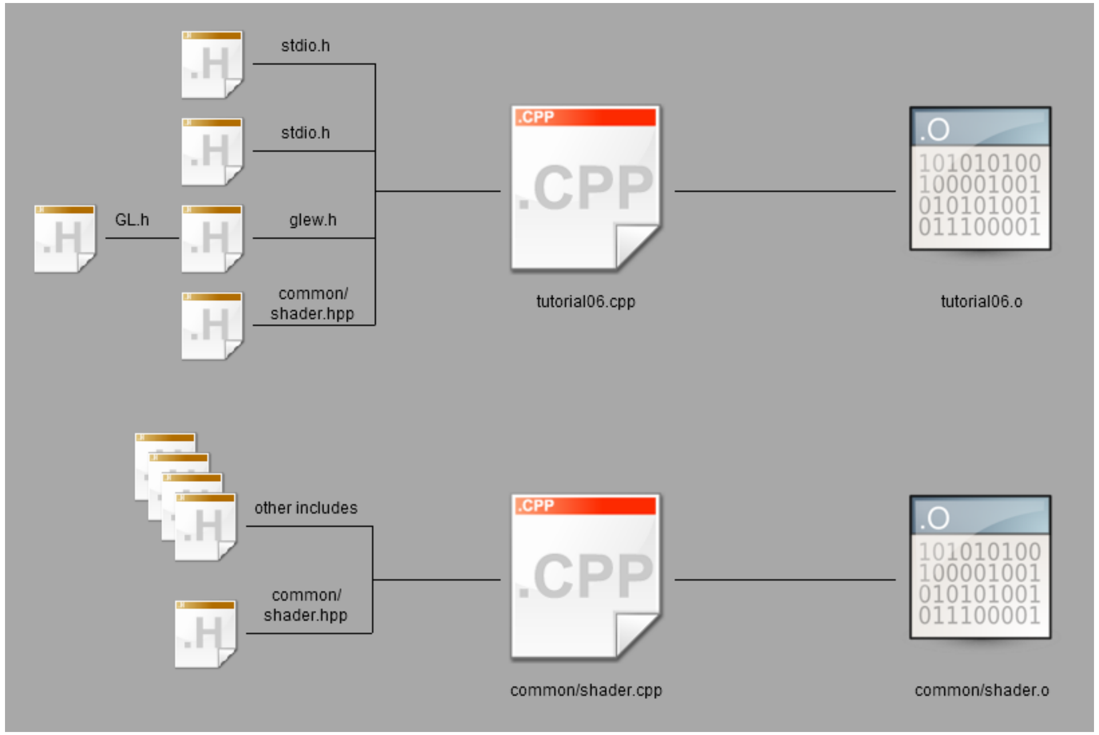
main.cpp

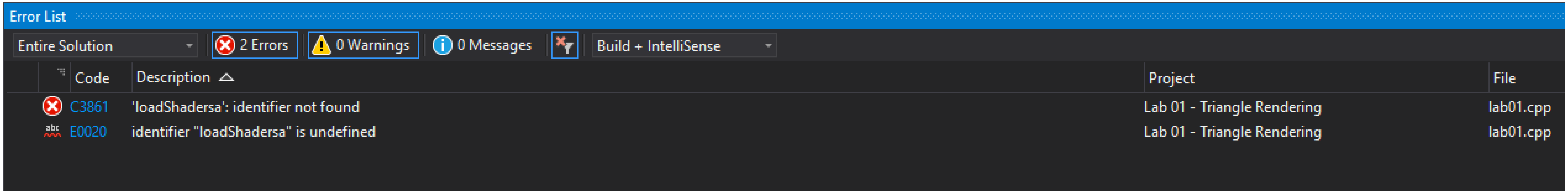
```cpp
#include <iostream>
using namespace std;
#include "person.h"
#include "worker.h"

int main(){
    Person kostas("kostas", 20); // allocation on stack
    cout << kostas.getAge() << endl;
    cout << kostas.name << endl; // compile time error (private member)

    Person alexis; // compile time error no default constructor?

    Person* dimitris; // contractor is not called
    // allocation on heap + polymorphism
    dimitris = new Worker("dimitris", 13, 100);
    cout << dimitris.getAge() << endl; // inherited from Person
    cout << dimitris.tellMeYourSalary() << endl; // new property of Worker

    return 1;
}
```

Compilation

- Syntax error
- Unable to find <GL/glew.h>

- Include directory must be known at compile time (e.g. where are <stdio.h>, <iostream> etc.)



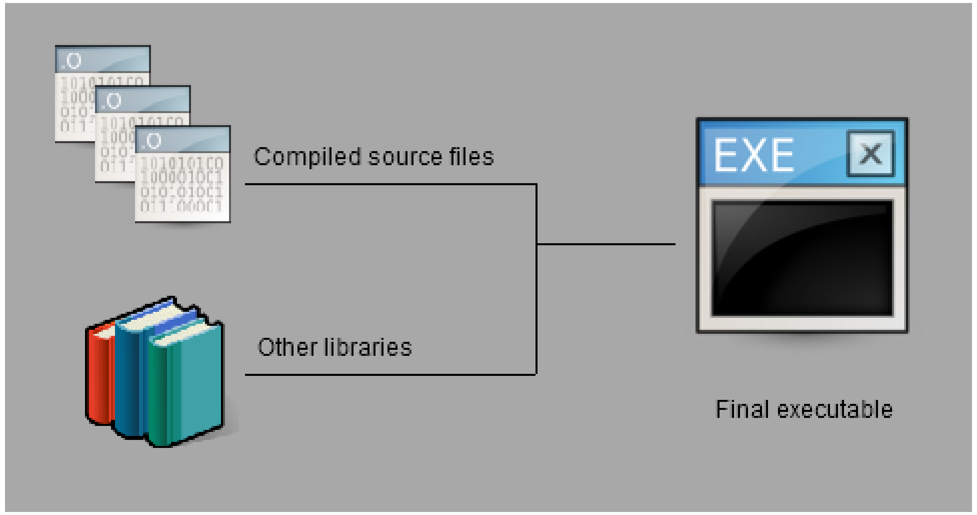


Code	Description	Project	File
C3861	'loadShadersa': identifier not found	Lab 01 - Triangle Rendering	lab01.cpp
E0020	identifier "loadShadersa" is undefined	Lab 01 - Triangle Rendering	lab01.cpp

Linking

- Linking error: unable to find static libraries

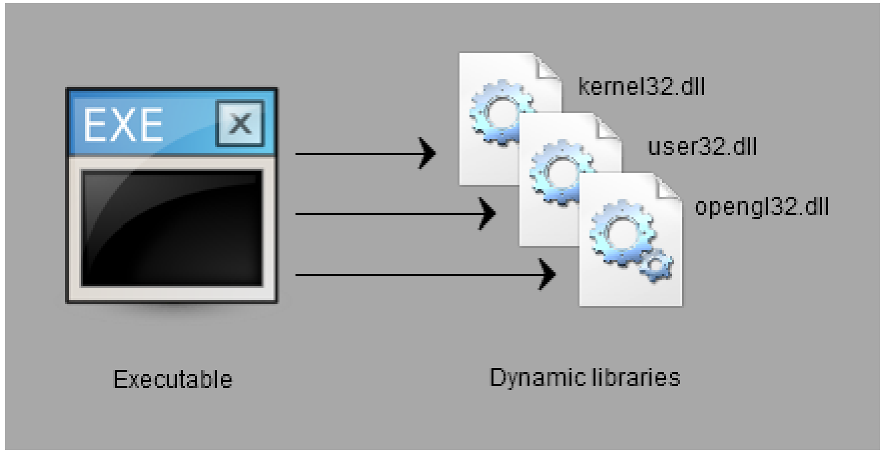
- The linker must know the location of the static libraries (Windows -> .lib, Linux -> .a)

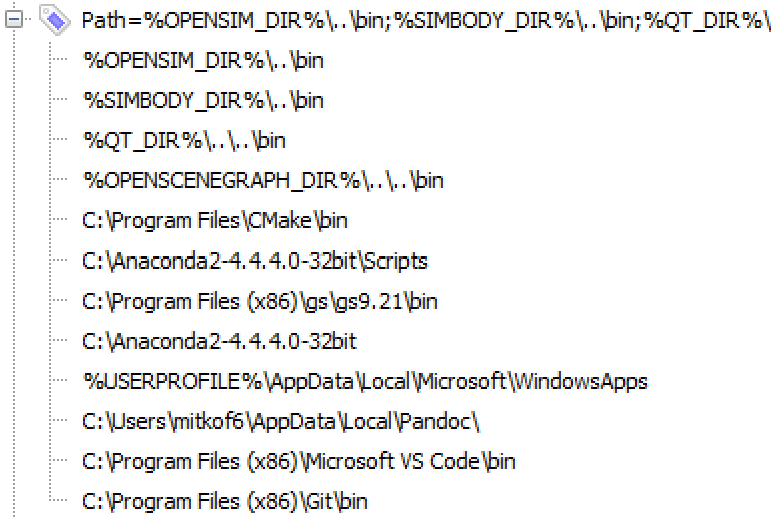


Dynamic Linking

- The application was unable to find opengl32.dll

- Dynamic libraries (Windows -> .dll, Linux -> .so) need to be located from the PATH variable or copied along with the application (.exe)

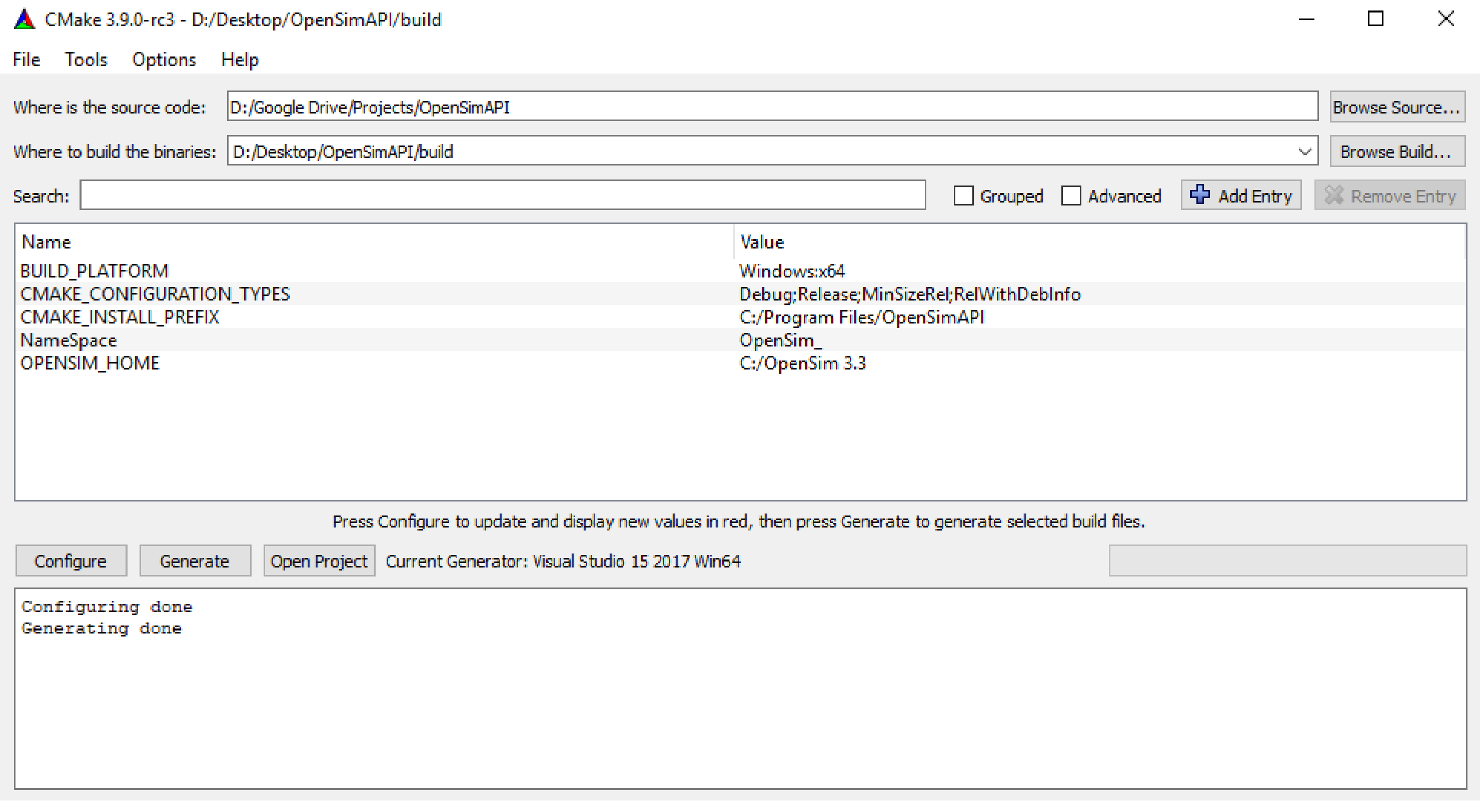




```
Path=%OPENSIM_DIR%\..\bin;%SIMBODY_DIR%\..\bin;%QT_DIR%\
    %OPENSIM_DIR%\..\bin
    %SIMBODY_DIR%\..\bin
    %QT_DIR%\..\..\bin
    %OPENSCENEGRAPH_DIR%\..\..\bin
    C:\Program Files\CMake\bin
    C:\Anaconda2-4.4.4.0-32bit\Scripts
    C:\Program Files (x86)\gs\gs9.21\bin
    C:\Anaconda2-4.4.4.0-32bit
    %USERPROFILE%\AppData\Local\Microsoft\WindowsApps
    C:\Users\mitkof6\AppData\Local\Pandoc\
    C:\Program Files (x86)\Microsoft VS Code\bin
    C:\Program Files (x86)\Git\bin
```

CMake

```
cmake_minimum_required (VERSION 2.6)
project(your_project_name)
# Creating a new project
find_package(OpenSim REQUIRED)
set(target lab01)
# Adding a source files in a project
add_executable(${target}
    lab01.cpp
)
# Adding include directories
include_directories(
    ${OpenSim_INCLUDE_DIRS}
)
# Link with libraries
set(ALL_LIBS
    ${OpenSim_LIBRARIES}
)
# Create target
target_link_libraries(${target} ${ALL_LIBS})
```



CMake 3.9.0-rc3 - D:/Desktop/OpenSimAPI/build

Where is the source code: D:/Google Drive/Projects/OpenSimAPI

Where to build the binaries: D:/Desktop/OpenSimAPI/build

Search:

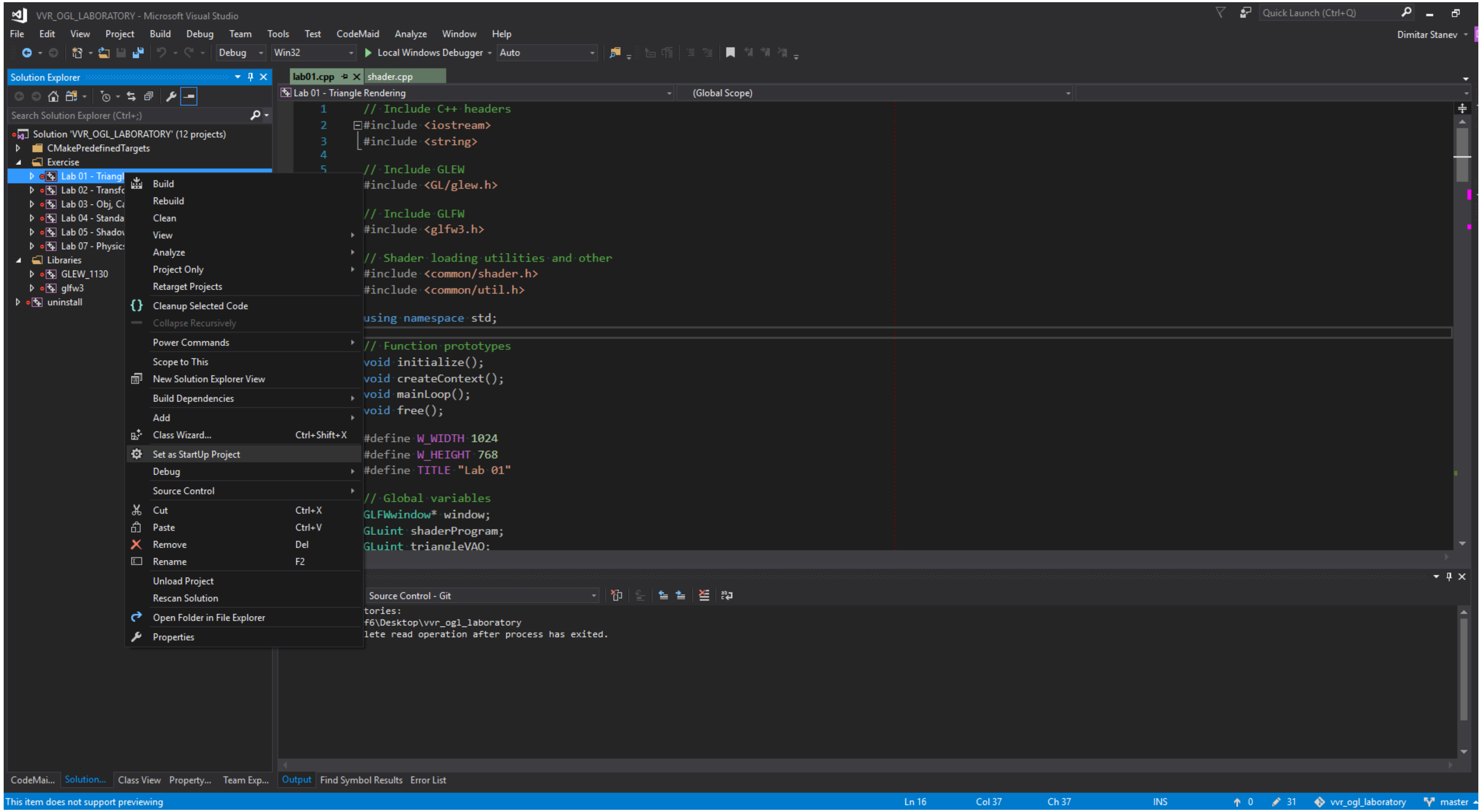
Name	Value
BUILD_PLATFORM	Windows:x64
CMAKE_CONFIGURATION_TYPES	Debug;Release;MinSizeRel;RelWithDebInfo
CMAKE_INSTALL_PREFIX	C:/Program Files/OpenSimAPI
NameSpace	OpenSim_
OPENSIM_HOME	C:/OpenSim 3.3

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 15 2017 Win64

```
Configuring done
Generating done
```

Visual Studio



```
// Include C++ headers
#include <iostream>
#include <string>

// Include GLEW
#include <GL/glew.h>

// Include GLFW
#include <glfw3.h>

// Shader loading utilities and other
#include <common/shader.h>
#include <common/util.h>

using namespace std;

// Function prototypes
void initialize();
void createContext();
void mainLoop();
void free();

#define W_WIDTH 1024
#define W_HEIGHT 768
#define TITLE "Lab 01"

// Global variables
GLFWwindow* window;
GLuint shaderProgram;
GLuint triangleVAO;
```

```
tories:
f6\Desktop\vvr_ogl_laboratory
lete read operation after process has exited.
```

Links

- https://simtk-confluence.stanford.edu:8443/display/OpenSim/API+Examples#
- https://simtk-confluence.stanford.edu:8443/display/OpenSim/Developer%27s+Guide