COMP3016

Introduction

Immersive Game Technologies

Dr Ji-Jian Chin

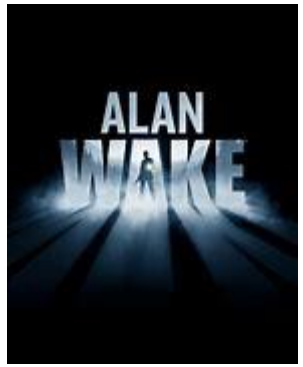
University of Plymouth 2024/2025

Welcome!

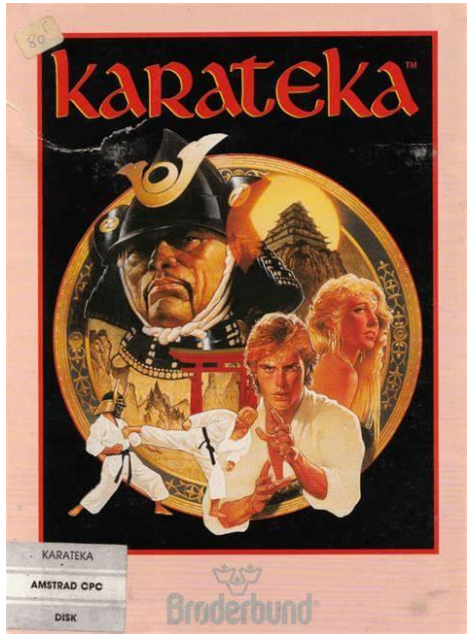
COMP3016
Immersive Game Technologies

- Stable changes each year
- setting up a project, basic C++ introduction *(you will have to do self-directed learning to keep up)*, OpenGL starter, required maths, file formats, lighting and texturing, design principles, maybe a bit more advanced stuff like physics and procedural generation
- one short project on C++, one longer project on OpenGL
- Teaching staff:
  - Dr. Ji-Jian Chin ji-jian.chin@plymouth.ac.uk
  - Sebastian Outram sebastian@outram.org

About Me?



- Dr. Ji-Jian Chin
- PhD in Cryptography
- Why am I teaching this module? I know how to code, I play games
- Gamer since 1988
- Archon 5 Dota Season 2, SC2 SEA Silver League
- Favourite games: immersive story (WC2,KOTOR, Alan Wake, RDR2)
- Doesn't like: platformers



About You

- What skills do you need?
- Where do you want to go?

- What we provide?
  - platform independent development
  - close to hardware programming
  - Sufficient skills for a career pivot if you wish
  - a software piece for your job interview

COMP3016

Agenda:

- expectation
- introduction/organisation
- Coursework
- C++ fundamentals
- quickstart live code

Where to find course information?

- DLE page:
  - Will post all lectures and practicals

- GitHub
  - Host your own Git Repo
  - Show off your work

The lectures

- Thursday 9.00am – 11.00am
  - Emdeck 209
- Theory and Concepts presented and discussed (attendance highly recommended)
  - You will need to follow up on the concepts yourself!
- Guest lecture(s) by experts (still under arrangement)

The lectures

- Thursday 9.00am – 11.00am
  - Emdeck 209
- Theory and Concepts presented and discussed (attendance highly recommended)
  - You will need to follow up on the concepts yourself!
- Guest lecture(s) by experts (still under arrangement)

The labs

- Thursdays 11:00am – 13:00am
  - SMB 201
  - SMB 100 (spillover venue)
- dev time
- time for experimentation
- working on your project
- Q&A time

Planned Schedule

Week	Lab	Lecture	Dates	Coursework Dates
1	GDD Exercise	Intro and C++ Fundamentals	25 Sept	
2	Lab 0+1: C++ Fundamentals	OOP Coding	3 Oct	
3	Lab 2: OOP 1	OOP Coding 2	10 Oct	
4	Lab 3: OOP 2	Game Programming Patterns	17 Oct	
5	Lab 4: Tictactoe using SDL	Immersive Games Technology	24 Oct	
6	Reading week: CW1 polish	Introduction to OpenGL - Hello Triangle	31 Oct	
7	Lab 5: OpenGL - hello triangle. CW2 pitch.	OpenGL Details and Textures	7 Nov	
8	Lab 6: Quads, Colours, Uniforms and Textures	3D Math	14 Nov	
9	Lab 7: Moving around 3D space	Transforms, MVP and 3D Cube	21 Nov	CW1: 4 Nov
10	Lab 8: PCG	PCG and models	28 Nov	
11	Lab 9: Model loading	Lighting Physics and Optimisation	5 Dec	
12	Reading week: CW2 polish	Vortex Guest Lecture: Immersive Games	12 Dec	
				CW2: 7 January
		Vivas	8-16 January	

Assessed Learning Outcomes (LO)

1. Apply core programming principles (including mathematical concepts) within a high performance real-time environments such as graphical programming
2. Design, conceptualise and implement a working prototype with clearly defined features and optimised performance
3. Articulate method of approach and rationale for solution

Summative assessments

- First Component Development (30%)
  - Get your hands dirty with C++ (LO1,LO2)
  - Understand the un-managed code for building software on the command line

- Full Interactive Prototype (70%)
  - Build a FULLY WORKING prototype in an area of your choosing; requirement is that a focus should be on programming and OpenGL 4.X (LO1-LO2)
  - Articulate your design decisions and pitch them in a short video; Communicate Complex Concepts Concisely (LO3)

COMP3016

- Some examples of CW1 and CW2

COMP3016

Short 5min break?

COMP3016

Agenda:

- introduction/organisation

Questions so far?
Menti

- NEXT: C++ Fundamentals
- write a quick start game in C++

Visual Studio

1. What you'll need:
    1. when using VS2022 select:
        1. Desktop development C++ (C++ core packages)
        2. Universal Windows Platform Development (nuget)
        3. Linux Development with C++ (CMake)

COMP3016

Intro to C++

Immersive Games Technologies

Dr Ji-Jian Chin

University of Plymouth 2024/2025

Topics for this lecture

- Comparing Languages
- language specific elements
- alternative setups

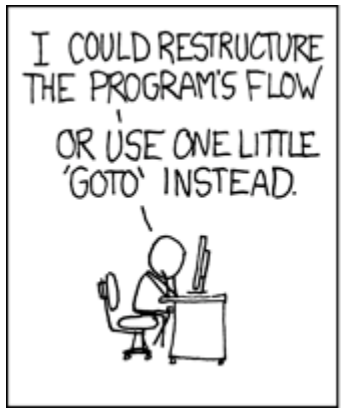
Comparison: C++ to Java/C#

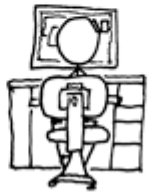
- Platform-independent
- Design Goal
- Support Unconditional Jumps
- Multiple inheritance
- Operator Overloading
- Pointers
- Compiler / Interpreter

- Compiler / Interpreter
- Call by Value and Call by reference
- Thread Support
- Documentation comment
- Inheritance Tree
- Hardware
- Object-oriented/ Procedural

Comparison: C++ to Java/C#

	C++	Java	C#
Platform-independent	N	Y	Y
Design Goal	C++ was designed for systems and applications programming. It was an extension of C	Java was designed and created as an interpreter for printing systems but later extended as a support network computing. It was designed with a goal of being easy to use and accessible to a broader audience.	Modern replacement for Java
Goto	Y	N	Y





Comparison: C++ to Java/C#

	C++	Java	C#
Multiple inheritance	Y	N (interfaces)	N
Operator Overloading	Y	N	Y
Pointers	Y	N (simulate using container objects)	Y (unsafe mode)
Compiler / Interpreter	Compiles to platform-dependent machine code	Compiles to java bytecode, interpreted at runtime	Compiles to CLR using the .NET VM to execute or the machine code
Call by Value and Call by reference	C++ supports both call by value and call by reference.	Java supports call by value only. There is no call by reference in java.	
Thread Support	C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support.	Java has built-in thread support.	C# has built-in thread support. caveat Unity performance vs co-routines
Documentation comment	C++ doesn't support documentation comment.	Java supports documentation comment (/** ... */) to create documentation for java source code.	C# supports documentation comments(/// … ///) using a tag system.
Inheritance Tree	C++ always creates a new inheritance tree.	Java uses a single inheritance tree always because all classes are the child of Object class in java. The object class is the root	Uses single inheritance tree; all children have root System.Object as root
Hardware	C++ is nearer to hardware.	Java is not so interactive with hardware.	C# is in between C++ and Java
Object-oriented	Y However, in C language, single root hierarchy is not possible.	Y However, everything (except fundamental types) is an object in Java. It is a single root hierarchy as everything gets derived from java.lang.Object.	Y

How about C++ vs Python?

Aspect	C++	Python
Performance	High performance, compiled language	Slower execution speed, interpreted language
Syntax	Complex and verbose syntax	Clean, concise syntax
Ease of Learning	More challenging, steeper learning curve	Easier to learn, beginner-friendly
Typing System	Statically-typed	Dynamically-typed
Memory Control	Manual memory management (can be error-prone)	Automatic memory management
Use Cases	Systems programming, game development, real-time applications	Web development, scripting, automation
Libraries	Rich libraries and frameworks available	Extensive standard library, many third-party libraries
Community	Large and active community	Large and active community

C++ From a C#/Java Perspective

- Syntax

```
#include <iostream>


int main()
{
  bool doneBefore;
  std::cout << "Hello students,\n"
       << "Welcome to COMP3016!\n";
  std::cout << "Have you used C++ before?"
  std::cin >> doneBefore;

  if (!doneBefore)
    std::cout << "Don't get bored!";
  else
    std::cout << "Let's fix this then!";
  return 0;
}
```

Using Namespaces

- Syntax

Making it neater by using namespaces!

```
#include <iostream>
using namespace std;

int main()
{
  bool doneBefore;
  cout << "Hello students,\n"
       << "Welcome to COMP3016!\n";
  cout << "Have you used C++ before?"
  cin >> doneBefore;

  if (!doneBefore)
    cout << "Don't get bored!";
  else
    cout << "Let's fix this then!";
  return 0;
}
```

Pointers

Symbol	Name	Description
& (ampersand sign)	Address operator	Determine the address of a variable.
* (asterisk sign)	Indirection operator	Access the value of an address.

Declaring a pointer
int * a; //pointer to int
char * c; //pointer to char

Pointers

Symbol	Name	Description
& (ampersand sign)	Address operator	Determine the address of a variable.
* (asterisk sign)	Indirection operator	Access the value of an address.

Declaring a pointer

```
int     *   a; //pointer to int
char *  c; //pointer to char
```

Pointer Example

```
#include <iostream>

int main()
{
}
```

Pointers

Symbol	Name	Description
& (ampersand sign)	Address operator	Determine the address of a variable.
* (asterisk sign)	Indirection operator	Access the value of an address.

Declaring a pointer

```
int     *   a; //pointer to int
char *  c; //pointer to char
```

Pointer Example

```
#include <iostream>

int main()
{
  int number=30;
  int *   p;
  p=&number;   //stores the address
  std::cout<<"Address of number variable is:"<<&number<<std::endl;
  std::cout<<"Address of p variable is:"<<p<<std::endl;
  std::cout<<"Value of p variable is:"<<*p<<std::endl;
  return 0;
}
```

&number = 2

p → 2

*p

Pointers

Symbol	Name	Description
& (ampersand sign)	Address operator	Determine the address of a variable.
* (asterisk sign)	Indirection operator	Access the value of an address.

Declaring a pointer

```
int * a; //pointer to int
char * c; //pointer to char
```

Output

```
Address of number variable is:0x7ffccc8724c4
Address of p variable is:0x7ffccc8724c4 Value of p variable is:30
```

Pointer Example

```
#include <iostream>

int main()
{
  int number=30;
  int * p;
  p=&number;  //stores the address
  std::cout<<"Address of number variable is:"<<&number<<std::endl;
  std::cout<<"Address of p variable is:"<<p<<std::endl;
  std::cout<<"Value of p variable is:"<<*p<<std::endl;
  return 0;
}
```

https://beginnersbook.com/2014/01/c-pointers/

Member Access

Symbol	Description
:: as in a::b	b is member of class or namespace for a
. as in a.b	b is member of the object a, that means a always must be an object
-> as in a->b	b is a member of the referenced object for a; a is a pointer; a->b = (*a).b

Pointer Example

```
#include <iostream>
int main()
{
  int number=30;
  int * p;
  p=&number;  //stores the address
  std::cout<<"Address of number variable is:"<<&number<<std::endl;
  std::cout<<"Address of p variable is:"<<p<<std::endl;
  std::cout<<"Value of p variable is:"<<*p<<std::endl;
  return 0;
}
```

Procedures

- **Free Functions VS Member Functions**

```
#include <iostream>
using namespace std;

struct Soft
{
  void attendLecture(string lect){}
};

void attendLecture(string lect){}
int main()
{
  attendLecture("COMP3016");
  Soft COMP3016;
  COMP3016.attendLecture("COMP3016");
  return 0;
}
```

member fct

free fct

Procedures

- **Free Functions VS Member Functions**

```
#include <iostream>
using namespace std;

struct Soft
{
  void attendLecture(string lect){}
};

void attendLecture(string lect){}
int main()
{
  attendLecture("COMP3016");
  Soft COMP3016;
  COMP3016.attendLecture("COMP3016");
  return 0;
}
```

```
#include <iostream>
using namespace std;

class Soft
{
  void attendLecture(string lect){}
};

void attendLecture(string lect){}
int main()
{
  attendLecture("COMP3016");
  Soft COMP3016;
  COMP3016.attendLecture("COMP3016");
  return 0;
}
```

Access Modifier

- public
- protected/internal
- private

	C++	Java	C#
Structs/ classes	Public	Default/protected	internal
Class members	Private	Default/protected	private
Struct Members	Public	-	Internal

Access Modifier

	C++	Java	C#
Structs/ classes	Public	Default/protected	internal
Class members	Private	Default/protected	private
Struct Members	Public	-	Internal

```
#include <iostream>
using namespace std;

class Soft
{
  public:
   void attendLecture(string lect){}
}
int main()
{
  Soft COMP3016;
  COMP3016.attendLecture("COMP3016");
  return 0;
}
```

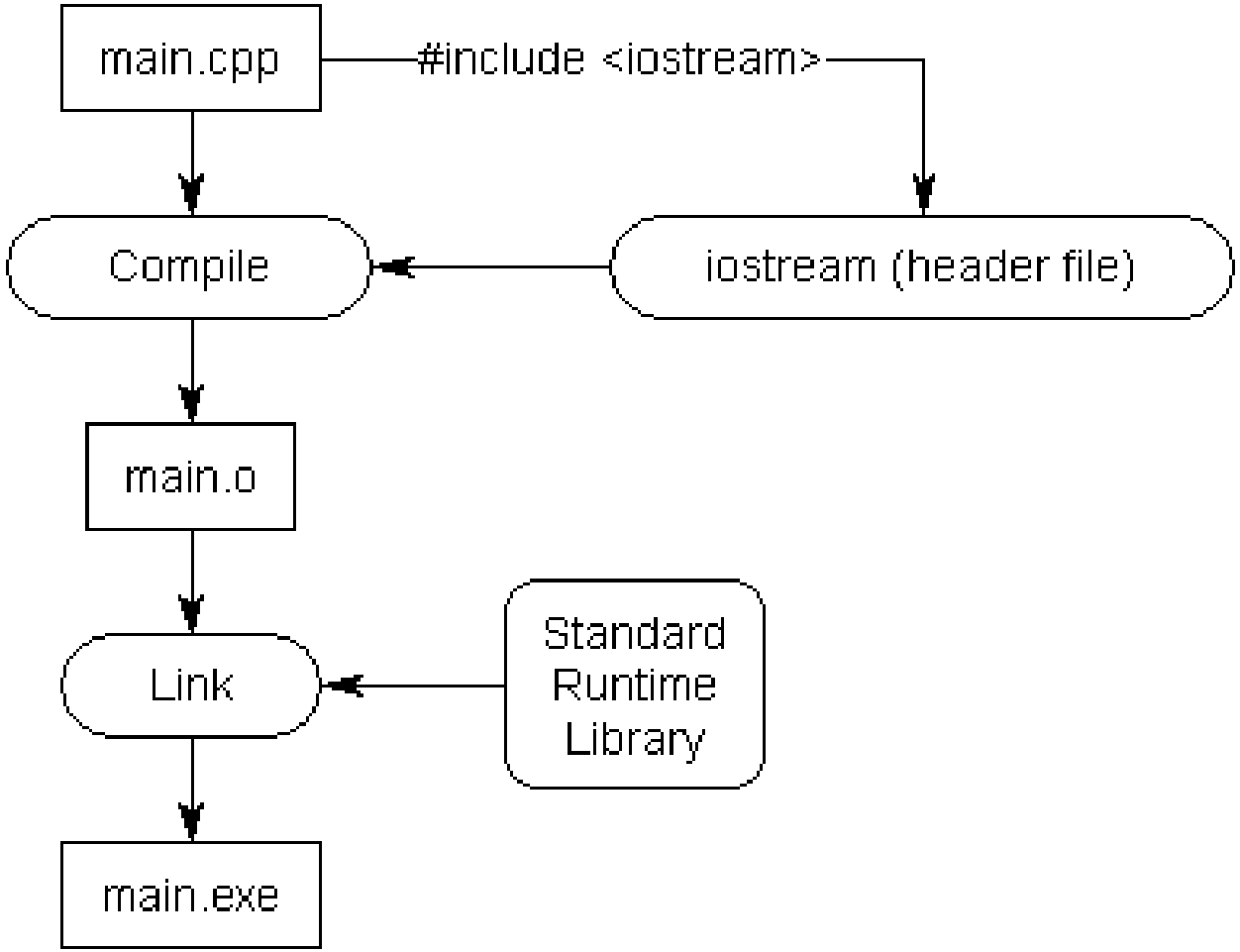
Header Files

```
#include <iostream>
using namespace std;

class Soft : public Module
{
  public:
   void attendLecture(string lect){}
}

int main()
{
  Soft COMP3016;
  COMP3016.attendLecture("COMP3016");
  return 0;
}
```

- not compiled but included/ copied in place by compiler
- similar to interfaces in Java/C#
- provides link to function declaration

cplusplus.com article
learncpp.com header details

Header Files



- not compiled but included/ copied in place by compiler
- similar to interfaces in Java/C#
- provides link to function declaration

cplusplus.com article
learncpp.com header details

Include Directive

```
#include <iostream>
using namespace std;

class Soft
{
  public:
   void attendLecture(string lect){}
}

int main()
{
  Soft COMP3016;
  COMP3016.attendLecture("COMP3016");
  return 0;
}
```

Include Directive

```
#include <iostream>
using namespace std;

class Soft
{
  public:
   void attendLecture(string lect){}
}

int main()
{
  Soft COMP3016;
  COMP3016.attendLecture("COMP3016");
  return 0;
}
```

#include <filename>

- search directories pre-designated by the compiler/IDE
- used to include standard library header files

#include "filename"

- first searches the same directory as the file containing the directive
- search directories pre-designated by the compiler/IDE
- used to include programmer-defined header files

Include Directive

```
#include “Module.h”
using namespace std;

class Soft : public Module
{
  public:
   void attendLecture(string lect){}
}

int main()
{
  Soft COMP3016;
  COMP3016.attendLecture(“COMP3016”);
  return 0;
}
```

#include <filename>

- search directories pre-designated by the compiler/IDE
- used to include standard library header files

#include "filename“

- first searches the same directory as the file containing the directive
- search directories pre-designated by the compiler/IDE
- used to include programmer-defined header files

Reading, Writing

- R/W from/to prompt

  Input:
  cin

  Output:
  cout
  cerr
  clog

```
#include <iostream>
using namespace std;

int main( int argc,char** argv)
{
  char correct;
  cout << "I thick you are awake!" << "/n";
  cout << "Is this correct? (y/n)";
  cin >> correct;
  if (correct == 'y')
    cout << "Thank you!" << endl;
  return 0;
}
```

Reading, Writing

- reading app inputs

  Input:
  $ g++ Source.cpp -o main
  $ ./main Soft 356 Module

  Output: …

```
#include <iostream>
using namespace std;

int main( int argc,char** argv)
{
  cout << "Number of arguments:" << argc << "/n";
  for (int i=0;i<argc;i++)
    cout <<argv[i] << endl;
  return 0;
}
```

Reading, Writing

- Reading app inputs

  Input:
  $ g++ Source.cpp -o main
  $ ./main Comp 1000 Module

  Output:
  Number of arguments:4
  ./main
  Comp
  1000
  Module

```
#include <iostream>
using namespace std;

int main( int argc,char** argv)
{
  cout << "Number of arguments:" << argc << "/n";
  for (int i=0;i<argc;i++)
    cout <<argv[i] << endl;
  return 0;
}
```

Reading, Writing

- R/W from/to file

  using stdio.h

```
#include <stdio.h>



int main( int argc,char** argv)
{
  FILE * pFile;
  fopen_s(&pFile, "/media/map.text","rb");
  if (!pFile)
  {
   /// error
  }
  fseek(pFile, 0, SEEK_END);
  int len = ftell(pFile);
  fseek(pFile, 0, SEEK_SET);
  GLchar* source = new GLchar[len + 1];
  fread(source, 1, len, pFile);
  return 0;
}
```

Reading, Writing

- R/W from/to file

  using fstream

```
#include <iostream>
#include <fstream>

using namespace std;

int main( int argc,char** argv)
{
  string line;
  ifstream myFile ("/media/map.text");
  if (myFile.is_open())
  {
   getline(myFile,line);
   cout << line << endl;
   myFile.close();
  }

  return 0;
}
```

Reading, Writing

- R/W from/to file

  using fstream

```
#include <iostream>
#include <fstream>

using namespace std;

int main( int argc,char** argv)
{
  string line;
  ofstream myFile ("/media/map.text");
  if (myFile.is_open())
  {
   myFile << "Some Text to fill the time" << endl;

   myFile.close();
  }

  return 0;
}
```

Basic Data Types

	Integer DataTypes
Short, int, long, float, double	Varying length depending on machine (32/64)
int8_t, int16_t	Fixed width (1byte,2bytes)

```
#include <iostream>
using namespace std;

int main()
{
  uint8_t shortCounter = 254; /// 1 byte of mem
  cout << "Your max range is" << shortCounter;
  return 0;
}
```

Basic Data Types

	String DataTypes
string	Modern string std::string
const char * str	C-style string

```
#include <iostream>
using namespace std;

int main()
{
  const char * str = "This is a c-string";
  string name("JJChin"); ///declare & init
  string module;        /// declare
  module = "COMP3016";  ///init
  return 0;
}
```

Basic Data Types

	Array DataTypes
[]	Static length, perfect for optimization
vector<T>	Variable length, template based

```
#include <vector>
using namespace std;

int main
{
  int array[20];

  cout << "Static Size" << sizeof(array[0]) << endl;
  vector<int> vec;
  vec.push_back(1);
  vec.push_back(2);
  cout << "Variable Size" << vec.size() << endl;
  return 0;
}
```

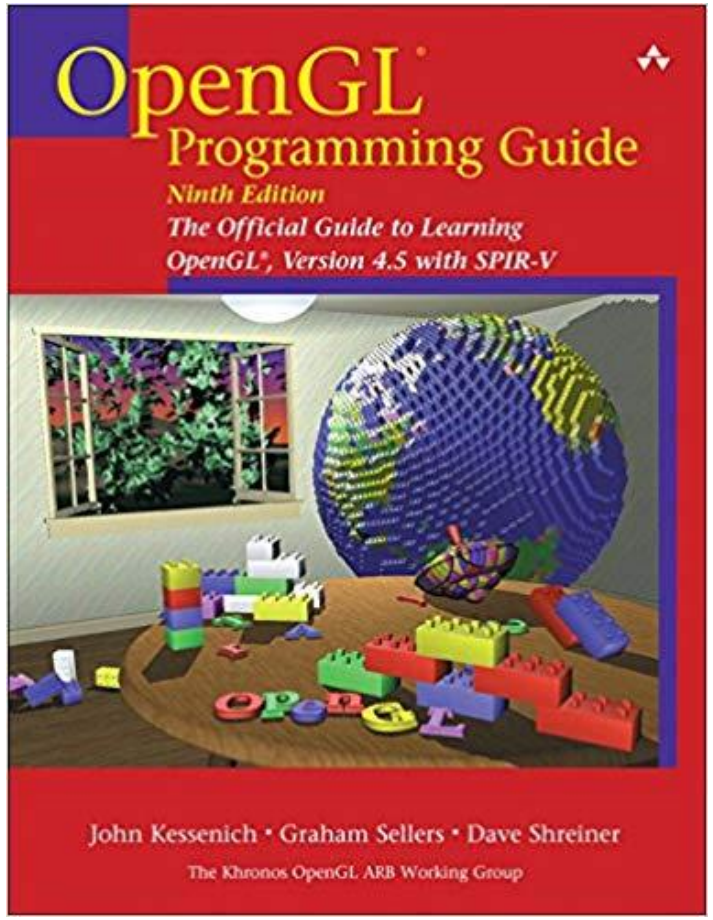
Thanks so far!

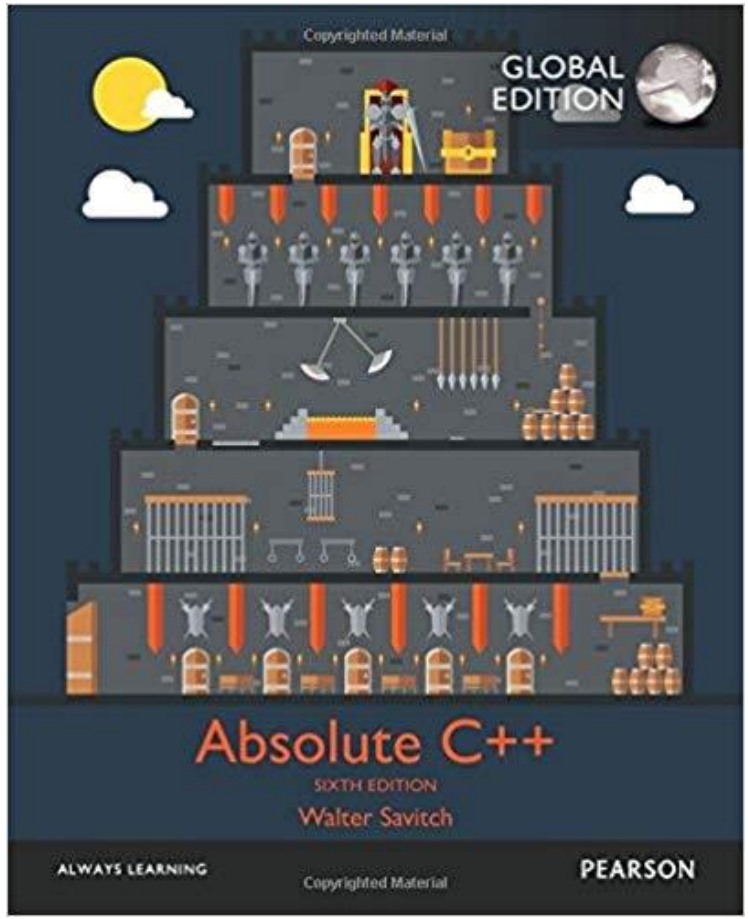
Any Questions?

Lets see some C++ in action

Reading List

LearnOpenGL.com





Kungfu practice:

- https://www.codewars.com/
- https://leetcode.com
- https://www.codingame.com