Welcome to CS 106L!

Reminder: class starts at 3:30 PM

Game Plan

- Welcome
- Why C++ and CS 106L?
- Logistics
- History and Philosophy of C++
- C++ Basics

Introduction

Instructors

Why C++?

C++ is still a very popular language.

Sep 2019	Sep 2018	Change	Programming Language	Ratings	Change
1	1		Java	16.661%	-0.78%
2	2		C	15.205%	-0.24%
3	3		Python	9.874%	+2.22%
4	4		C++	5.635%	-1.76%
5	6		C#	3.399%	+0.10%

Take that, Python!

Programming language popularity: C++ bounces back at Python's expense

Broader compiler support is driving a resurgence in interest in the nearly 35-year-old C++ programming language, which replaces Python in Tiobe's top 3.

By Liam Tung | April 8, 2019 -- 12:43 GMT (20:43 GMT+08:00) | Topic: Enterprise Software

5

Python has seen the largest rise of any

MORE FROM LIAM TUNG

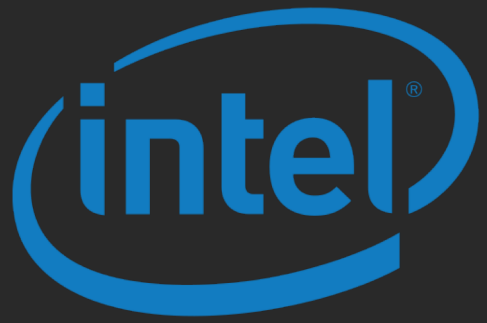
Google

Google: We've changed search rankings to reward 'original news reporting'

Classes that use C++

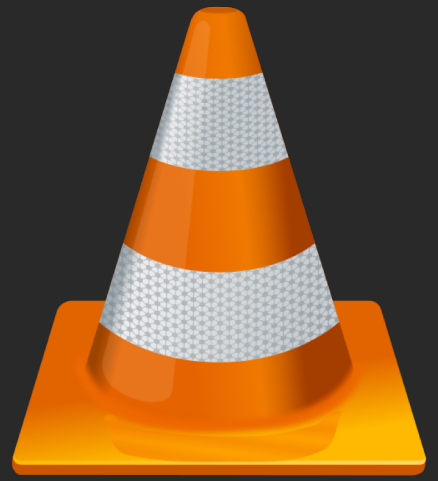
BIOE 215: Physics-Based Simulation of Biological Structure
CME 213: Introduction to parallel computing using MPI
CS 144: Introduction to Computer Networking
CS 231N: Convolutional Neural Networks for Visual Recognition
GENE 222: Parallel Computing for Healthcare
ME 328: Medical Robotics
MUSIC 256A: Music, Computing, Design I
MUSIC 420A: Signal Processing Models in Musical Acoustics

Companies that use C++



Browsers written in C++

Software written in C++

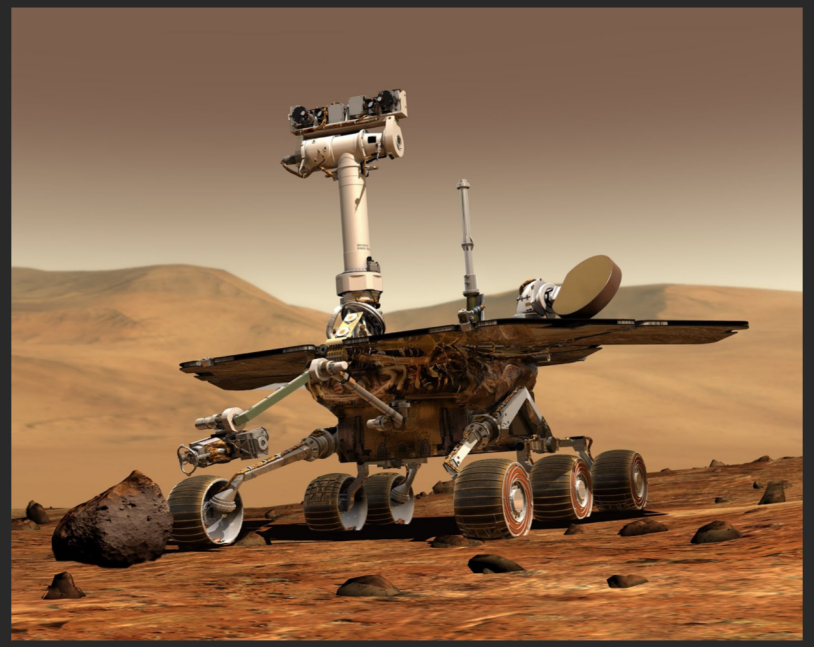


Games written in C++

Cool stuff written in C++

The F-35 Lightning II (Joint Strike Fighter) relies extensively on C++

The Spirit rover was operational for over 6 years when the mission was only planned to run for around 3 months



“One of the things I really like about programming languages is that it's the perfect excuse to stick your nose into any field. So if you're interested in high energy physics and the structure of the universe, being a programmer is one of the best ways to get in there. It's probably easier than becoming a theoretical physicist”

-Bjarne Stroustrup

Why CS 106L?

Comparison of 106B vs. 106L

CS 106B Programming Abstractions

CS 106L Standard C++ Programming

Comparison of 106B vs. 106L

CS 106B	Applicable to all programming languages
CS 106L	Applicable mostly to C++ (perhaps others)

Comparison of 106B vs. 106L

CS 106B

Barely enough C++ to learn programming abstractions.

CS 106L

Enough C++ for a job, internship, or research.

Comparison of 106B vs. 106L

CS 106B C++98*

CS 106L C++17
(sneak peek into C++20)

*exception: range-based for loop.

Comparison of 106B vs. 106L

CS 106B Stanford libraries abstract away messy details.

CS 106L Deep dive into messy C++ details.

Comparison of 106B vs. 106L

CS 106B "Just use getInteger"

CS 106L "Let's tame cin"

Goals of CS 106L

1. Learn what features are out there in C++ and why they exist.

2. Become comfortable with reading C++ documentation.

3. Become familiar with the design philosophy of modern C++.

NOT: memorize the syntax of C++.

C++ documentation is not beginner friendly.

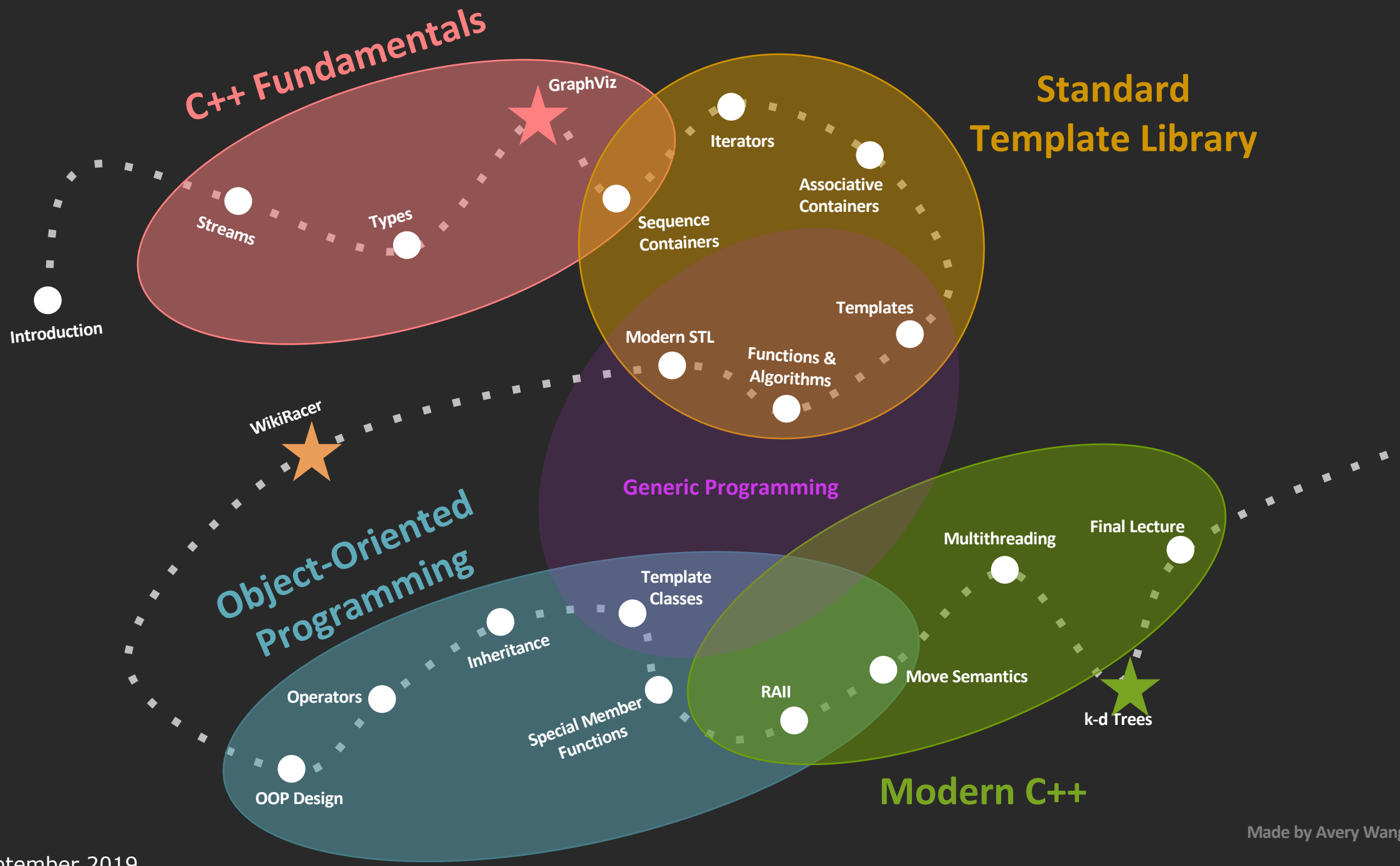
```
vector<int> nums; // the first default constructor
```

default (1)	`vector();` `explicit vector (const allocator_type& alloc);`

fill (2)	`explicit vector (size_type n, const allocator_type& alloc = allocator_type());`
	`vector (size_type n, const value_type& val,`
	`const allocator_type& alloc = allocator_type());`

range (3)	`template <class InputIterator>`
	`vector (InputIterator first, InputIterator last,`
	`const allocator_type& alloc = allocator_type());`

copy (4)	`vector (const vector& x);` `vector (const vector& x, const allocator_type& alloc);`
move (5)	`vector (vector&& x);` `vector (vector&& x, const allocator_type& alloc);`
initializer list (6)	`vector (initializer_list<value_type> il,` `const allocator_type& alloc = allocator_type());`



Made by Avery Wang

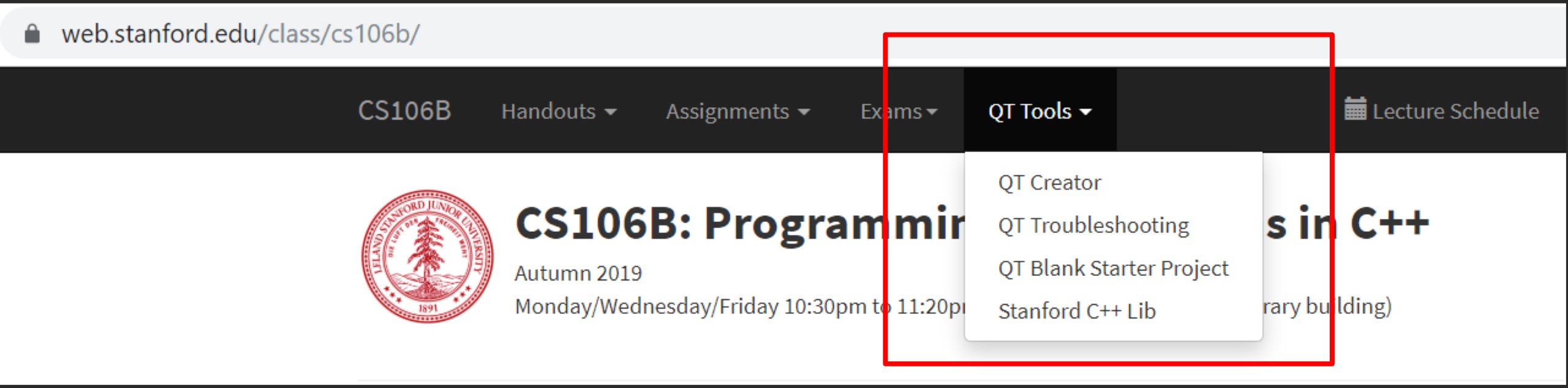
Logistics

Logistics

Lecture:	T/Th 3:30-4:20 in 380-380C, weeks 1-9
Website:	https://cs106l.stanford.edu
Getting Help:	Office Hours, Piazza, do not use LaIR
Assignments:	3 total, complete 2 satisfactorily for credit
Late Days:	Two 24-hour late days
Development:	Qt Creator (from CS 106B)
Honor Code:	Don't cheat. Same rules as CS 106B.

Piazza: https://piazza.com/stanford/fall2019/cs106l

QT Creator Setup



Troubleshooting session:
Thursday, 9/26, 7:30 - 9:30 pm at LaIR

Comparison of 106B vs. 106L

CS 106B	3 lectures + section / week
CS 106L	2 lectures / week

Comparison of 106B vs. 106L

CS 106B

undergrad 5, grad 3-5 units
letter grade or C/NC

CS 106L

1 unit S/NC
auditors welcome!

Comparison of 106B vs. 106L

CS 106B	7 assignments + section + exams
CS 106L	3 assignments, choose 2

History of C++

Some C++ Code

```cpp
#include <iostream>

int main() {
    std::cout << "Hello, world!" << std::endl;
    return 0;
}
```

Also Some C++ Code

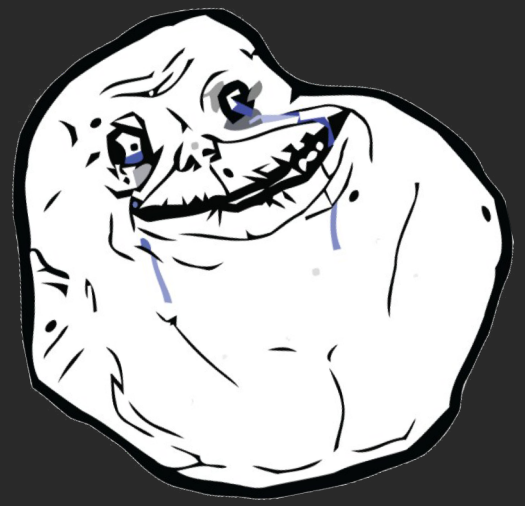
```c
#include "stdio.h"
#include "stdlib.h"

int main(int argc, char *argv) {
    printf("%s", "Hello, world!\n");
    return EXIT_SUCCESS;
}
```

...Also (Technically) Some C++ Code

```
#include "stdio.h"
#include "stdlib.h"

int main(int argc, char *argv) {
   asm(  "sub    $0x20,%rsp\n\t"
         "movabs $0x77202c6f6c6c6548,%rax\n\t"
         "mov    %rax,(%rsp)\n\t"
         "movl   $0x646c726f, 0x8(%rsp)\n\t"
         "movw   $0x21, 0xc(%rsp)\n\t"
         "movb   $0x0,0xd(%rsp)\n\t"
         "leaq    (%rsp),%rax\n\t"
         "mov    %rax,%rdi\n\t"
         "call  __Z6myputsPc\n\t"
         "add    $0x20, %rsp\n\t"
     );
    return EXIT_SUCCESS;
}
```



C++ History: Assembly

C++ History: Assembly

```
section     .text
global      _start                          ;must be declared for linker (ld)

_start:                                             ;tell linker entry point

    mov     edx,len                         ;message length
    mov     ecx,msg                         ;message to write
    mov     ebx,1                           ;file descriptor (stdout)
    mov     eax,4                           ;system call number (sys_write)
    int     0x80                                    ;call kernel
    mov     eax,1                           ;system call number (sys_exit)
    int     0x80                                    ;call kernel

section     .data
msg     db  'Hello, world!',0xa             ;our dear string
len     equ $ - msg                         ;length of our dear string
```

C++ History: Assembly

Benefits:

- Unbelievably simple instructions
- Extremely fast (when well-written)
- Complete control over your program

Why don’t we always use assembly?

C++ History: Assembly

```
section     .text
global      _start                          ;must be declared for linker (ld)

_start:                                             ;tell linker entry point

    mov     edx,len                         ;message length
    mov     ecx,msg                         ;message to write
    mov     ebx,1                           ;file descriptor (stdout)
    mov     eax,4                           ;system call number (sys_write)
    int     0x80                                    ;call kernel
    mov     eax,1                           ;system call number (sys_exit)
    int     0x80                                    ;call kernel

section     .data
msg     db  'Hello, world!',0xa             ;our dear string
len     equ $ - msg                         ;length of our dear string
```

C++ History: Assembly

FAST, but...

Simple instructions?

→ Requires lots of code to do simple tasks

→ Hard to understand other people's code

Complete control over program?

→ Extremely unportable

C++ History: Invention of C

C++ History: Invention of C

Writing assembly was too difficult but computers only understood assembly.

Idea:

- Source code can be written in a more intuitive language
- An additional program can convert it into assembly

This is called a compiler!

C++ History: Invention of C

K&R created C in 1972, to much praise.

C made it easy to write code that was

- Fast
- Simple
- Cross-platform

Learn to love it in CS107!

Ken Thompson and Dennis Ritchie, creators of the C language.

C++ History: Invention of C

C was popular since it was simple.

This was also its weakness:

- No objects or classes
- Difficult to write code that worked generically
- Tedious when writing large programs

C++ History: Welcome to C++!

C++ History: Welcome to C++!

In 1983, the first vestiges of C++ were created by Bjarne Stroustrup.

He wanted a language that was:

- Fast
- Simple to Use
- Cross-platform
- Had high level features

C++ History: Welcome to C++!

In 1983, the first vestiges of C++ were created by Bjarne Stroustrup.

He wanted a language that was:

- Fast
- Simple to Use
- Cross-platform
- Had high level features

C++ History: Welcome to C++!

In 1983, the first vestiges of C++ were created by Bjarne Stroustrup.

He wanted a language that was:

- Fast
- Simple to Use
- Cross-platform
- Had high level features

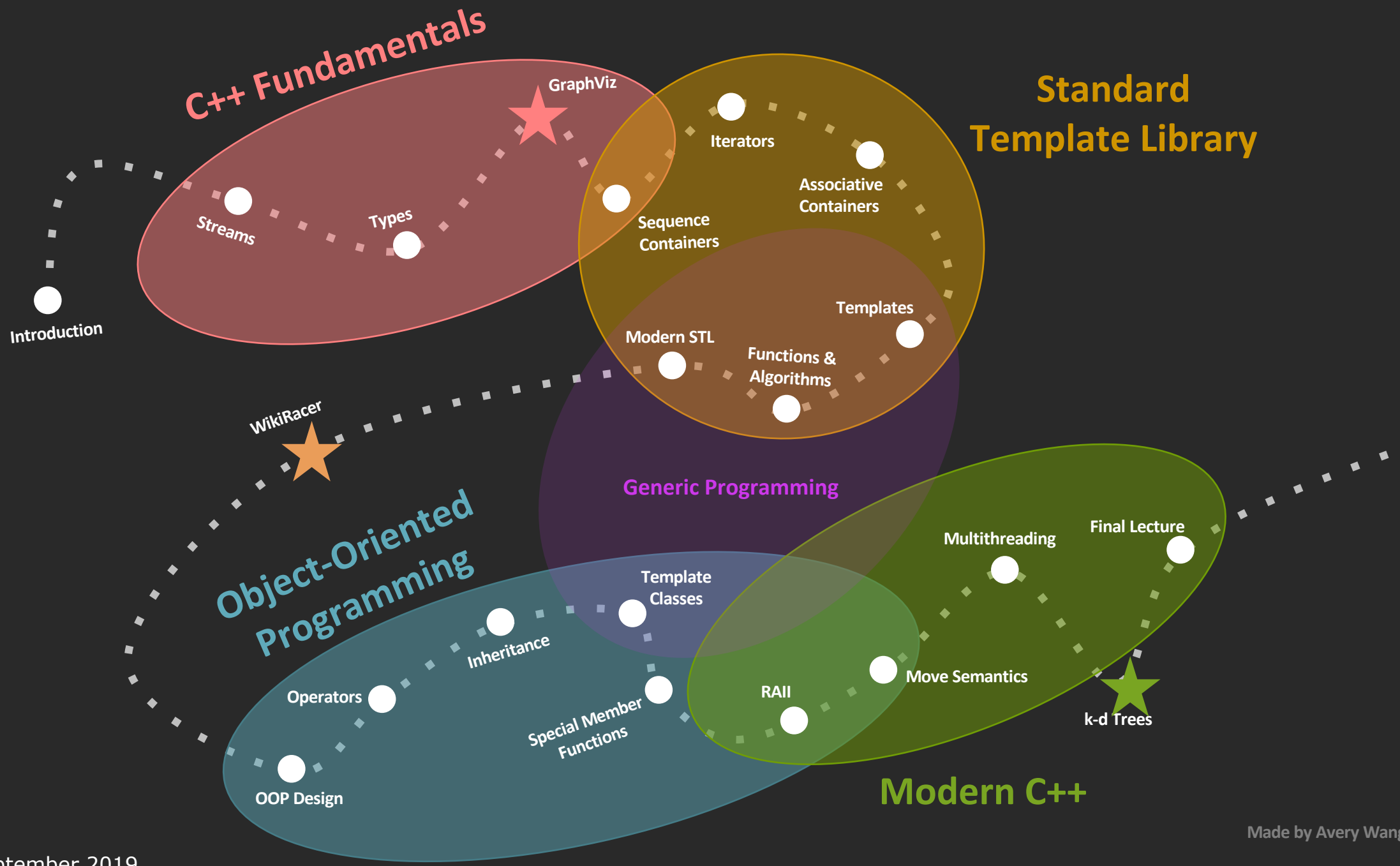
C++ History: Evolution of C++

Design Philosophy of C++

Design Philosophy of C++

Design Philosophy of C++

- Allow the programmer full control, responsibility, and choice if they want it.
-
-
-
-



Made by Avery Wang

Design Philosophy of C++

- Allow the programmer full control, responsibility, and choice if they want it.
- Express ideas and intent directly in code.
-
-
-

Finding the sum of a vector of ints.

```
vector<int> vec = {1, 2, 3};
int sum = 0;
for (auto val : vec) {
  sum += val;
}
```

```
vector<int> vec = {1, 2, 3};
int sum = 0;
for (const auto& val : vec) {
  sum += val;
}
```

```
vector<int> vec = {1, 2, 3};
int sum = std::accumulate(vec.begin(), vec.end(), 0);
```

Design Philosophy of C++

- Allow the programmer full control, responsibility, and choice if they want it.
- Express ideas and intent directly in code.
- Enforce safety at compile time whenever possible.
-
-

Design Philosophy of C++

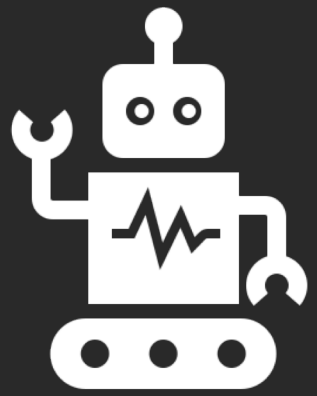
- Allow the programmer full control, responsibility, and choice if they want it.
- Express ideas and intent directly in code.
- Enforce safety at compile time whenever possible.
- Do not waste time or space.
-

Design Philosophy of C++

- Allow the programmer full control, responsibility, and choice if they want it.
- Express ideas and intent directly in code.
- Enforce safety at compile time whenever possible.
- Do not waste time or space.
- Compartmentalize messy constructs.

Design Philosophy of C++

- Allow the programmer full control, responsibility, and choice if they want it.
- Express ideas and intent directly in code.
- Enforce safety at compile time whenever possible.
- Do not waste time or space.
- Compartmentalize messy constructs.



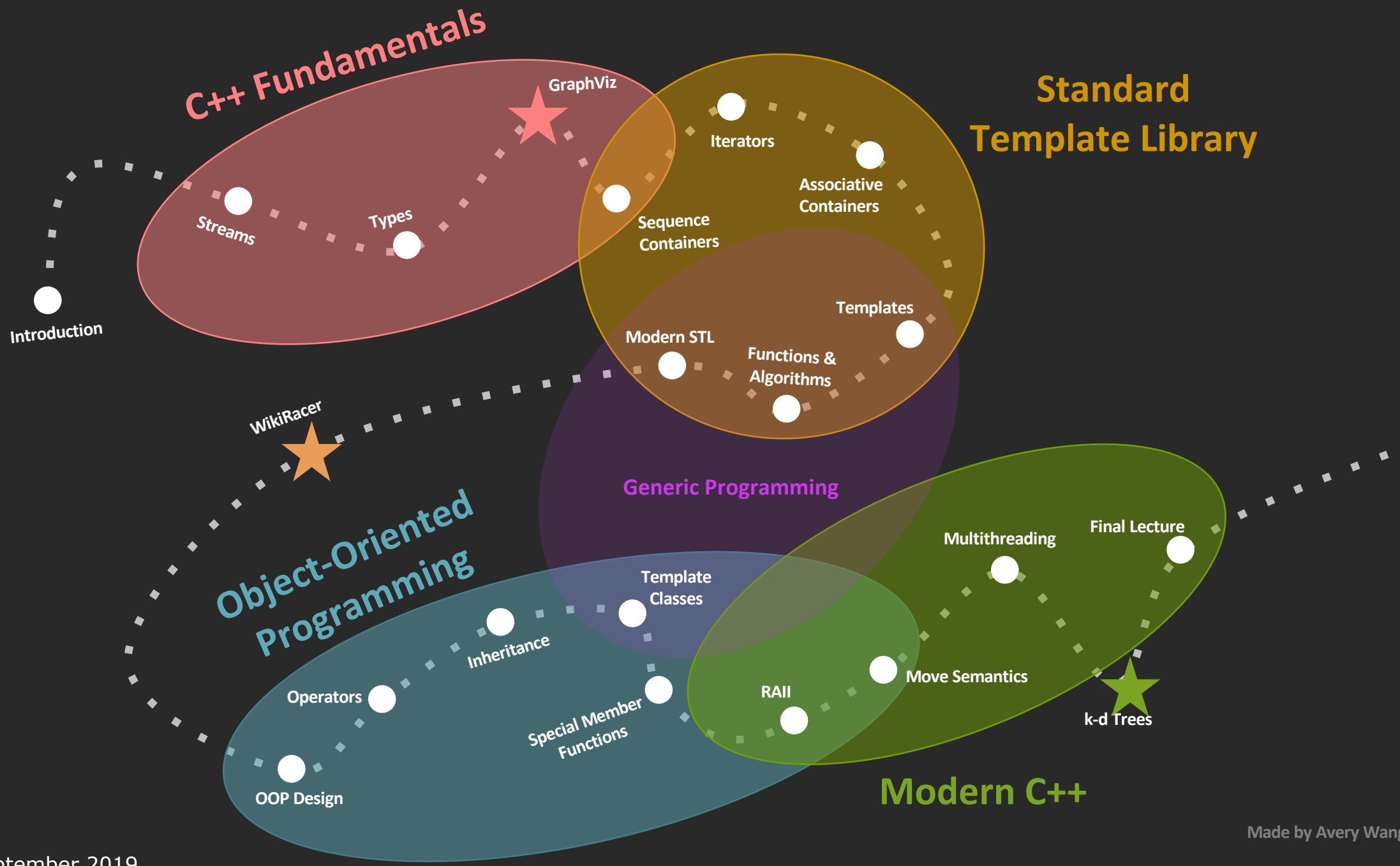
Example

Our first C++ program

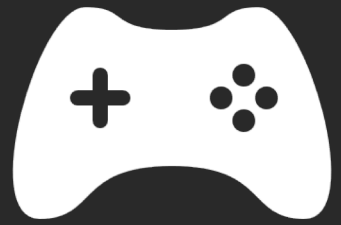
Survey

https://bit.ly/2mLb60Y

= +1 late day!



Made by Avery Wang



Next time

Streams

Looking Ahead

- streams abstraction
- stringstream
- state bits
- i/o streams and buffering
- [CS 106B - Friday] file streams and the Stanford library
- types: type inference, structures, initialization
- error-checking and implementing simpio.h
- manipulators
- overloading stream << and >>
- sequence containers