《面向对象程序设计基础》

项目阅读报告

基本要求

- 阅读优秀的开源项目代码，了解面向对象思想在实际工程中的应用，在实践中学习。
- 要求每位同学**独立完成**，最终上交**最多15页研究报告PPT**和**其中涉及的代码**
- 除助教提供的参考选题外，允许同学们自己选题。目标选题应该是具有相当程度代码规模的C++项目，但需要提前和助教沟通，以确定选题的合适性以及报告的具体要求。
- 总体要求：读懂设计模型，梳理框架结构，测试、拓展代码功能

*对代码中可能含有的具体算法不做要求，重在面向对象的设计方法。

研究报告要求

- PPT格式，最多15页（包括一页标题），使用默认16:9大小
- 要求重点突出，文字精炼
- 推荐结合图片、代码说明（只放核心代码），字体不可过小
- 若报告中有参考（文字或图片），**请在相应位置标注引用**（如网址）。若与网上资料、同学报告有大范围雷同，可能会判为抄袭
- 报告分为两个部分：
  - 项目整体介绍及框架分析
  - 具体功能测试与拓展

项目整体介绍及框架分析

- 项目整体介绍
  - 简要介绍项目功能
  - 包含使用方法、达成效果等
  - 最好附上自己使用/运行例子
- 代码框架分析
  - 使用合适的方法展示代码整体框架（若项目过大，可以选择其中一个模块）
    - 例如 UML 图，调用关系图，类的层次结构等

具体功能测试与拓展

- 选取一至两个功能点
  - 编写例子对该功能进行测试
  - 说明功能的工作流程
  - 分析该实现的优越性或可改进空间
    - 分析可涉及具体应用场景的使用、扩展功能时的便利程度
    - 需要给出核心代码
    - 可以结合课上所学设计模式进行分析
- 质量>数量：推荐深入研究，不必攀比数量

提交代码要求

- 需包含研究报告中的所有例子
- 代码重点部分应该包括在研究报告PPT中，不能用提交的代码代替PPT中的核心代码展示（即不看代码也应能看懂报告）
- 代码本身不评分，但如不能复现报告中结果，将会有所减分

- 注意提交代码大小，注意不要包含可执行文件
- 需附说明文件，提供合适的安装、编译步骤，保证助教能够复现你的结果

评分占比

- 大作业满分10分，占课程总评的10%

评分	占比
项目整体理解	4
具体功能分析	6

- 考查：
  - 正确性：对项目、框架的理解是否正确
  - 分析价值：所选功能分析或拓展是否有价值，讨论是否深入
  - 展示效果：报告是否简单明晰，内容丰富

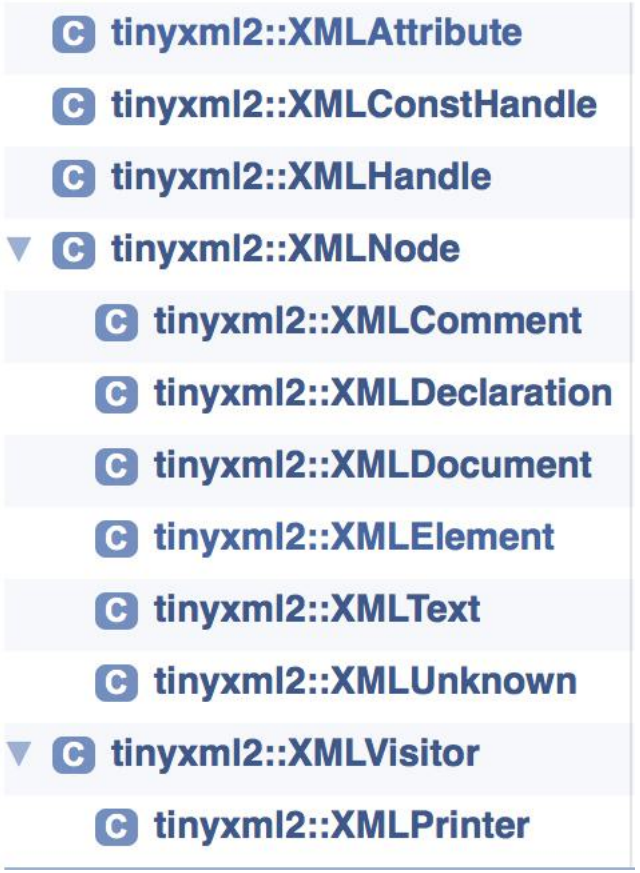
选题#1: TinyXML

- 对轻量级C++ XML parser： TinyXML-2 的分析。
- TinyXML-2 可以解析 XML 文档，使用 Document Object Model (DOM)，意味着能将 XML 转换为可操作的 C++ 的多个对象，也可以从 C++ 的多个对象构建 XML。
- github地址：https://github.com/leethomason/tinyxml2

TinyXML

- TinyXML-2 包含 tinyxml2.cpp，tinyxml2.h 两个文件。
- docs/index.html 中有简单的使用示例和类的说明文档

类的层次结构 ->

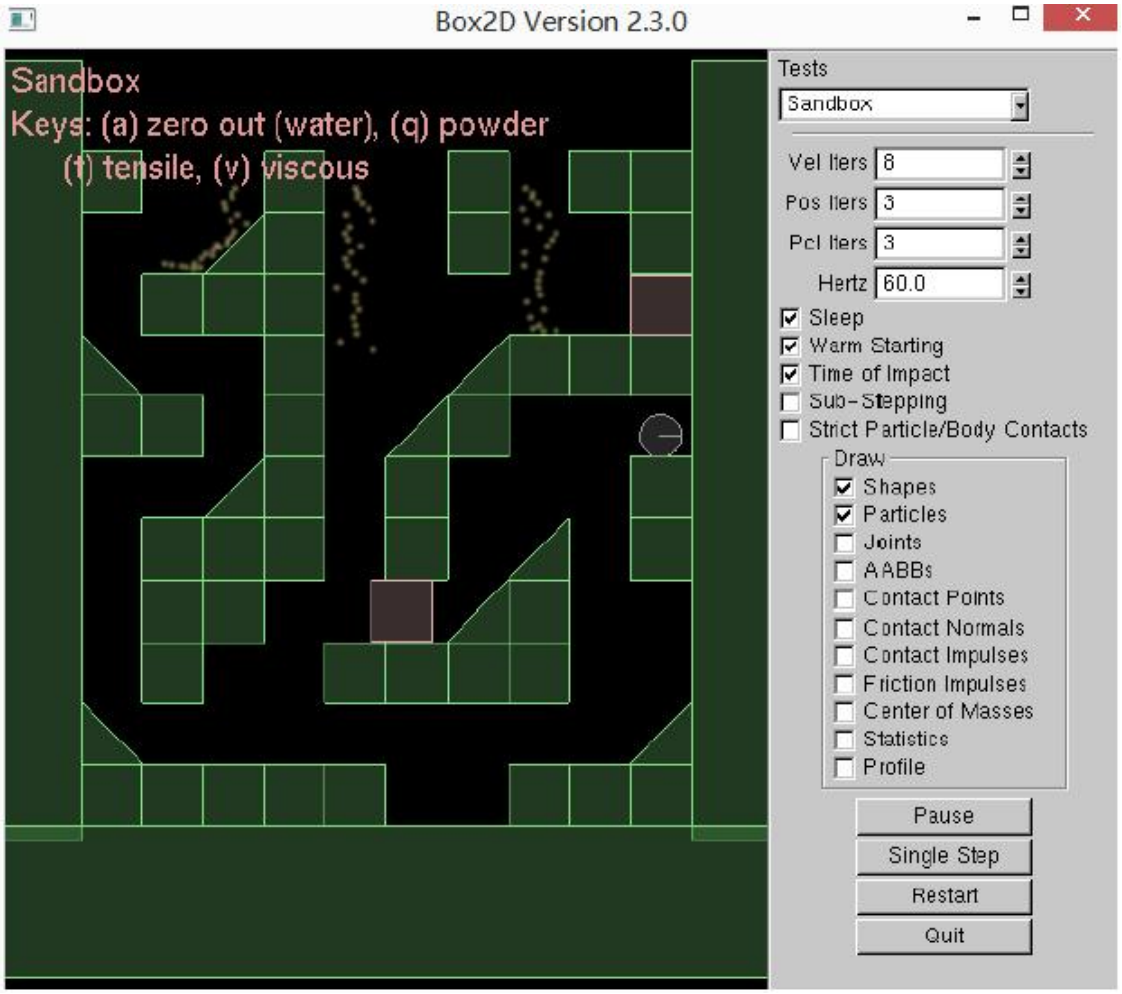


选题#2: LiquidFun

- liquidFun是一款基于Box2D的2D刚体模拟流体的C++库，主要用于游戏编程。它是Box2D引擎的扩展，它对Box2D的刚体功能扩展了基于粒子的流体模拟。
- 官网(打开这个网址可以看到example)：http://google.github.io/liquidfun/#Documentation
- 代码地址：https://github.com/google/liquidfun/

LiquidFun

- 粒子和物体的运行



选题#3: Eigen

- 基于C++模板的线性代数库
- 支持矩阵、向量有关的各种数值分析算法
- 配置简单、计算效率可观

Eigen

- Eigen的代码开源且可读性比较高，含有详细注释

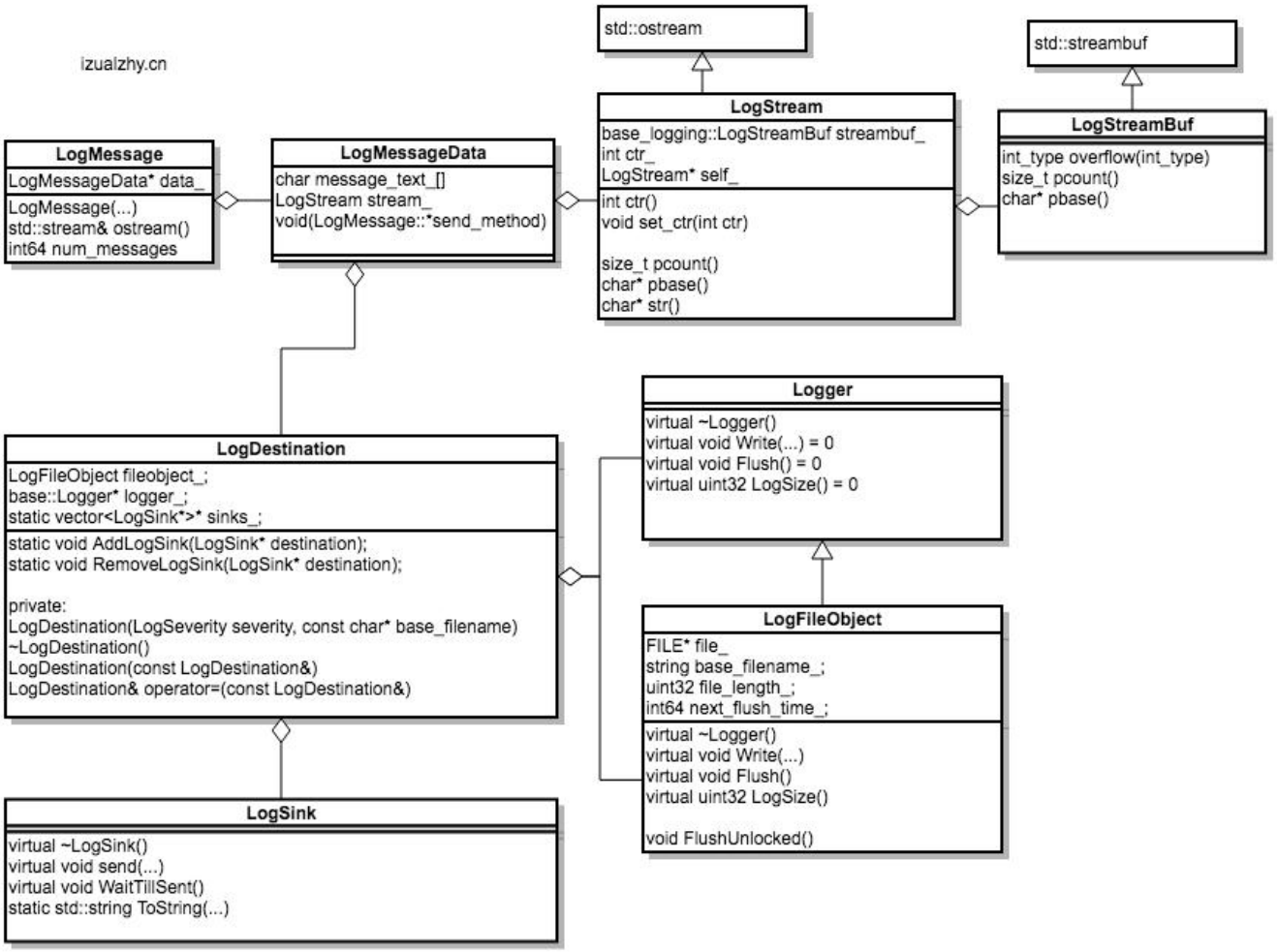
  https://gitlab.com/libeigen/eigen

- Eigen配有大量的示例代码，对库中函数的常见用法进行了详细说明，可在eigen/doc/examples目录下查看。

选题#4: glog

- Google glog是一个基于程序级记录日志信息的c++库，编程使用方式与c++的stream操作类似，
- 例：LOG(INFO) << "Found " << num_cookies << " cookies";
- 项目地址：https://github.com/google/glog

glog


izualzhy.cn
std::ostream
std::streambuf
LogStream
base_logging::LogStreamBuf streambuf_
int ctr_
LogStream* self_
int ctr()
void set_ctr(int ctr)
size_t pcount()
char* pbase()
char* str()
LogStreamBuf
int_type overflow(int_type)
size_t pcount()
char* pbase()
LogMessage
LogMessageData* data_
LogMessage(...)
std::stream& ostream()
int64 num_messages
LogMessageData
char message_text_[]
LogStream stream_
void(LogMessage::*send_method)
Logger
virtual ~Logger()
virtual void Write(...) = 0
virtual void Flush() = 0
virtual uint32 LogSize() = 0
LogDestination
LogFileObject fileobject_;
base::Logger* logger_;
static vector<LogSink*>* sinks_;
static void AddLogSink(LogSink* destination);
static void RemoveLogSink(LogSink* destination);
private:
LogDestination(LogSeverity severity, const char* base_filename)
~LogDestination()
LogDestination(const LogDestination&)
LogDestination& operator=(const LogDestination&)
LogFileObject
FILE* file_
string base_filename_;
uint32 file_length_;
int64 next_flush_time_;
virtual ~Logger()
virtual void Write(...)
virtual void Flush()
virtual uint32 LogSize()
void FlushUnlocked()
LogSink
virtual ~LogSink()
virtual void send(...)
virtual void WaitTillSent()
static std::string ToString(...)

选题#5: Nlohmann Json

- Nlohmann Json是一个用c++编写的json解析器
- Json是现有最为流行的数据格式之一，广泛应用在网络传输之中，而c++并没有自带的好用的json解析库
- Nlohmann Json为c++解析json提供了很好的方法
- 项目地址：
- https://github.com/nlohmann/json

Nlohmann Json

- Nlohmann Json最核心的代码：
- https://github.com/nlohmann/json/blob/develop/include/nlohmann/json.hpp
- 有非常详细的readme，包含使用的方法以及编译的方法

选题#6: Boost.MultiArray

- Boost是一个非常庞大的模块化的C++库，包含了各方面的工具。
- Boost的代码质量高、文档丰富，而且运用了许多C++的高级特性。
- Boost.MultiArray是一个高维数组的实现。它可以像普通的高维数组一样访问，此外还支持支持动态大小、快速选取子数组等方便的操作。
- *作为作业只需关注Boost.MultiArray，对Boost其他模块不作要求

Boost.MultiArray

- 代码及示例：
  https://github.com/boostorg/multi_array
- 官方文档：
  https://www.boost.org/doc/libs/1_61_0/libs/multi_array/doc/user.html

Boost.MultiArray

- 一个简单的示例：

```
#include "boost/multi_array.hpp"
#include <cassert>

int main() {
  // 定义 3 x 4 x 2 的三维数组
  const int N[3] = {3, 4, 2};
  boost::multi_array<int, 3> A(
      boost::extents[N[0]][N[1]][N[2]]);
  typedef decltype(A)::index index;
  // 随便写点数据
  for (index i = 0; i < N[0]; ++i)
    for (index j = 0; j < N[1]; ++j)
      for (index k = 0; k < N[2]; ++k)
        A[i][j][k] = static_cast<int>(i + j + k);
  // ...
```

Boost.MultiArray

- 一个简单的示例（续）：

```
  // ...
  // 抽出数组的一部分，得到2 x 2的子数组
  typedef decltype(A)::index_range range;
  typedef decltype(A)::array_view<2>::type myview;
  myview B = A[
      boost::indices[range(1, 3)][range(0, 3, 2)][1]];
  for (index i = 0; i < 2; ++i)
    for (index j = 0; j < 2; ++j)
      assert(B[i][j] == A[i + 1][j * 2][1]);

  return 0;
}
```