Function Go

张冠杰 201730684502

陈　亮 201764681188

夏锐航 201736684186

目录：

项目标题：Function Go（Visual 3D）

项目摘要：通过输入一个函数表达式将该函数式的函数图像绘制出来。

Github 地址：https://github.com/SCUTSSE/Visual3D

依赖：

opengl32.lib

glfw3.lib

glew32s.lib

开发流程：

出于团队项目开发周期以及规模的考虑，我们小组决定按照 Phased Prototype Model 进行开发，其中囿于实际情景的限制由我们三位开发者扮演用户的角色向互相向其他人提出改进的建议。

首先我们分析了整个函数绘图应用的总体需求与定义，并将其分为函数解析，2D 绘制以及 3D 绘制三个组成模块，之后每人负责一个模块进行深入的分析，包括设计算法以及设计测试等。开发过程中首先分析了每个模块的特点以及难点，在草做纲要之后商量互相之间的封装接口，之后着手代码开始实现相应的功能。

在各个模块基本功能初步实现并完成单元测试之后进行了一次联调，将此次联调中发现的一些问题向相应的组员反馈。每人在拿到反馈之后对代码进行重构以达到提高效率或优化体验的目的。

在完成以上步骤之后将项目精简生成 release 版本，作为 ver1.0 提交。

项目设计:

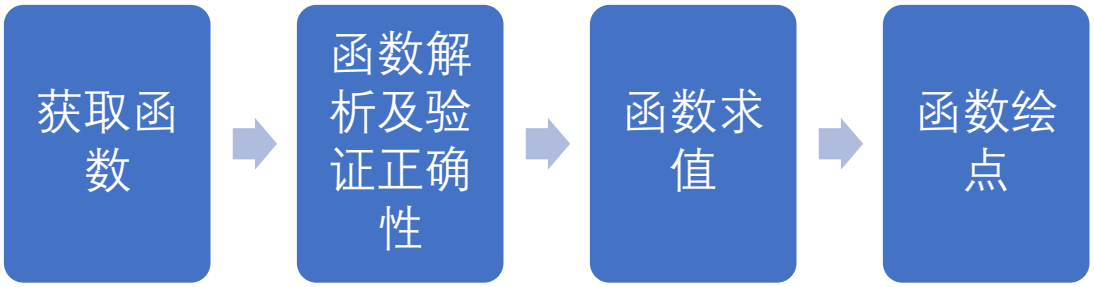


Figure 1 **项目执行流程**

项目亮点:

1、 设计了一门简洁直观的函数描述语言规范与用户的交互操作。(见 https://github.com/SCUTSSE/FunctionGo/blob/master/help/FunctionDescribeLanguage's_syntax.md)

2、 采用树结构对函数进行解析。

3、 针对不同类型的函数进行定向优化。

4、 采用多线程充分利用系统资源提升运行速度。

5、 灵活应用 opengl 进行绘制函数，轴长等于定义域宽度，直观表达。

6、 给用户提供了直接方便并且有效的操作方式以便更好地观察函数。(见 https://github.com/SCUTSSE/FunctionGo/blob/master/help/operation_of_Grapher.md)

功能概览:

1、 绘制任意给定二维函数图像，并支持缩放，旋转操作

2、 绘制三维函数图像，并支持缩放，旋转操作

代码说明:

解析模块:

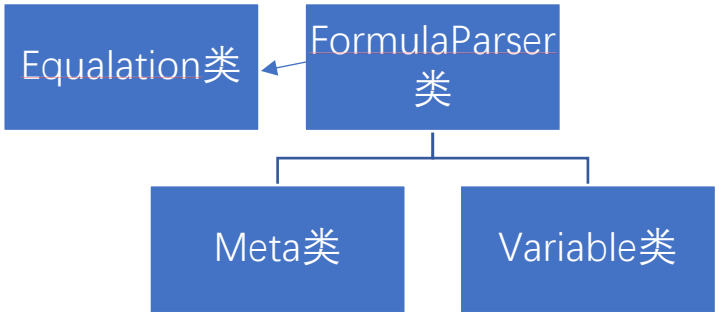


Figure 2 **解析模块类图**

类图如上所示，其中从下往上为 Meta 及 Variable 类与 FormulaParser 类为 has-a 关系。FormulaParser 类表示函数解析类，将输入的表达式转化为便于计算求值的树结构。其中树的内部节点与叶子节点即分别为 Meta 类和 variable 类。Meta 类表示每一个基础运算元，Variable 表示运算过程中所涉及到的变量，封装了工具函数便于求值。

最后 Equalation 类为 FormulaParser 的 proxy 类，隐藏了函数解析的内部细节，对外开放方便使用的求值接口。

求值时先将所有的变量 feed 给 variable 类，然后从 FormulaParser 所维护的表达式树的根出发，递归求出每个运算元的表达式值。

每次求值的时候提供两种方法予上层类调用，一个为一次性求出所有的点返回给主调函数，这种情形便于绘制 2D 的函数，点与点之间没有复杂的空间关系，并且采用多线程充分利用 cpu 的运算能力，平衡了精度和运算速度。另外一种为单点求值，便于主调函数按照自己的逻辑顺序来绘制较为复杂的空间函数图像。一次求一个非常有序，而且可以节省不用渲染的点的运算开销。

绘制模块:

本代码封装了 Grapher 类充当 OpenGL 的深度自定义的 shader。其主要函数有场景搭建函数 initScene()和 drawAxis()、标准化函数 standardize()、着色函数 drawFunction()和 drawTri3D()、构造函数 Grapher()。以下依次简略说明:

·场景搭建函数

·initScene() 用于初始化 OpenGL 的 glfw、glut 库，并且采用阴影平滑等个性设置。

·drawAxis() 用于描绘坐标轴与箭头，长度与输入函数定义域宽度一致

·标准化函数

·standardize() 用于对离散的点作适当的排序 可补充三角阵列分组

·着色函数

·drawFunction() 是一个汇合接口的函数，方便调用函数解析、2D 绘图、3D 绘图

·drawTri3D() 用于 3D 绘图

·构造函数

·Grapher() 有初始化窗口、设置、使用交互回调函数、构建轮询平台等功能

另外类外函数有:

·comp(): 制定 sort 中的排序规则

·key_callback():制定交互回调函数的规则

Main 模块：

通过 WindowsAPI 搭建了一个文本框，功能包括输入，删除，左右键，HOME，END 键或鼠标点击移动光标位置等。

通过弹窗指引用户完成输入，最后调用 Grapher 完成绘图。

测试流程：

单元测试：

- 函数解析模块
    - 基本的常数函数
    - 带有多项式的函数
    - 含有括号改变运算优先级的函数
    - 带超越函数的函数
    - 带复合多项式的函数
    - 带复合超越函数的函数
    - 隐函数
    - 不正确的函数
- 绘制模块
    - 引用 OpenGL3d 模型，检测 openGl 环境搭建是否正确
    - 回调函数测试，测试交互过程是否正确
    - 输入离散的点，检验连线是否正确
    - 输入离散的点，检测三角剖分是否正确
    - 输入简单二维表达式，检验二维绘图是否正确

    - 输入复杂二维表达式，检验二维绘图是否正确
    - 输入简单三维表达式，检验三维绘图是否正确
    - 输入复杂三维表达式，检验三维绘图是否正确
    - 搭建坐标轴场景，代入函数，定点检验场景是否正确
    - 构建不同的函数检验 3d 感官增强模块的效果，并且选出最佳的颜色以及透明度
    - 以上测试均添加放缩测试，检验放缩功能是否无误
    - 以上测试均添加旋转测试，检验旋转功能是否无误
- 联合调试
    - 输入已知图像的函数并将其与输出对比。

以下补充测试过程测试图

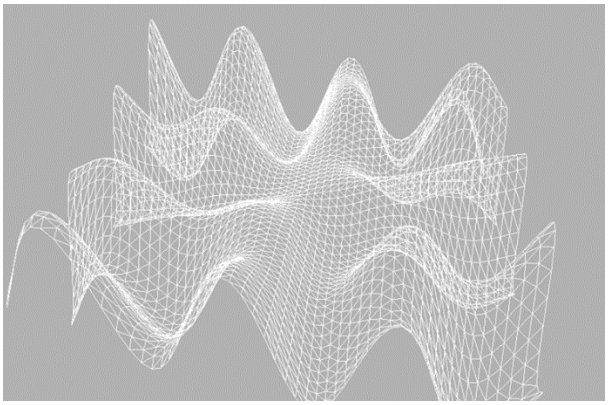


Figure3.三角剖分

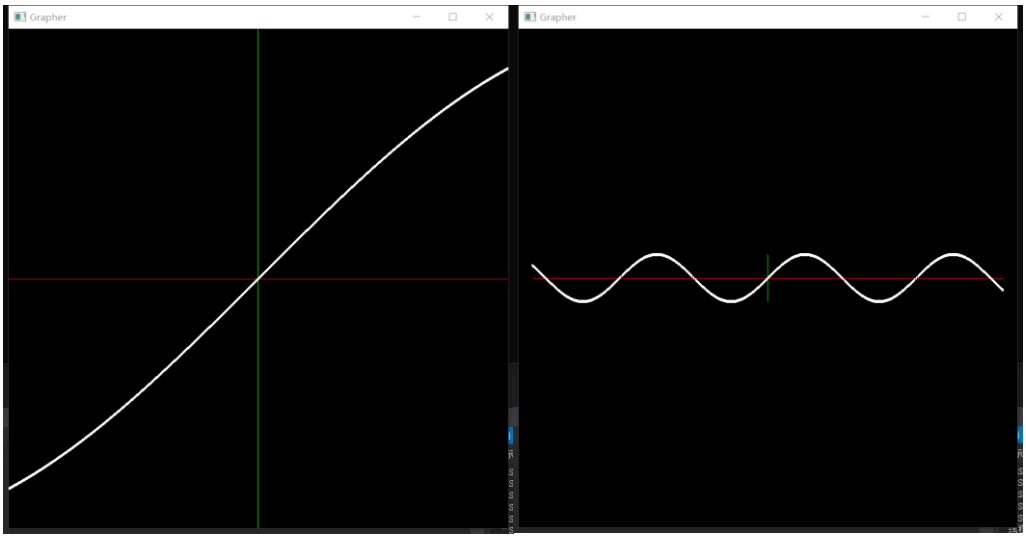


Figure4. 简单二维：sinx 在[-10.0,10.0] 左为放缩前，右为放缩后

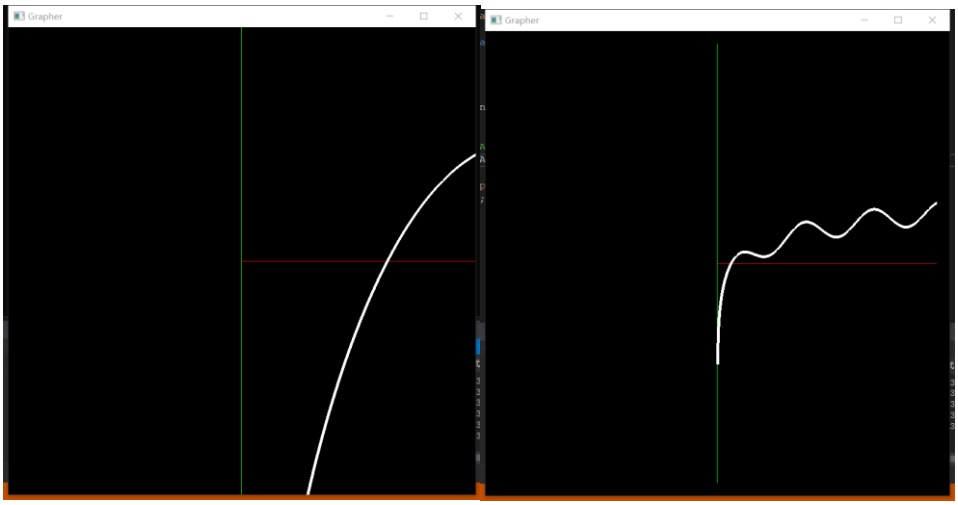
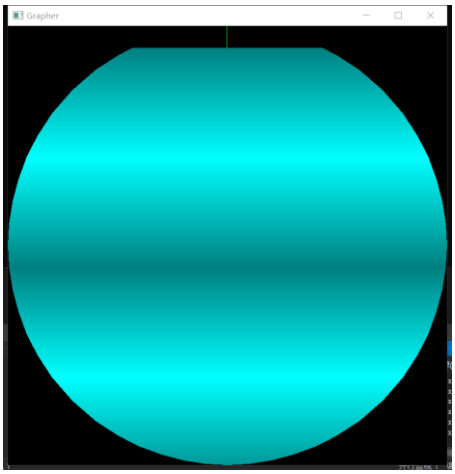


Figure6. 复杂二维：y=sinx * cosx + lnx 在[0,10.0] 左为放缩前，右为放缩后



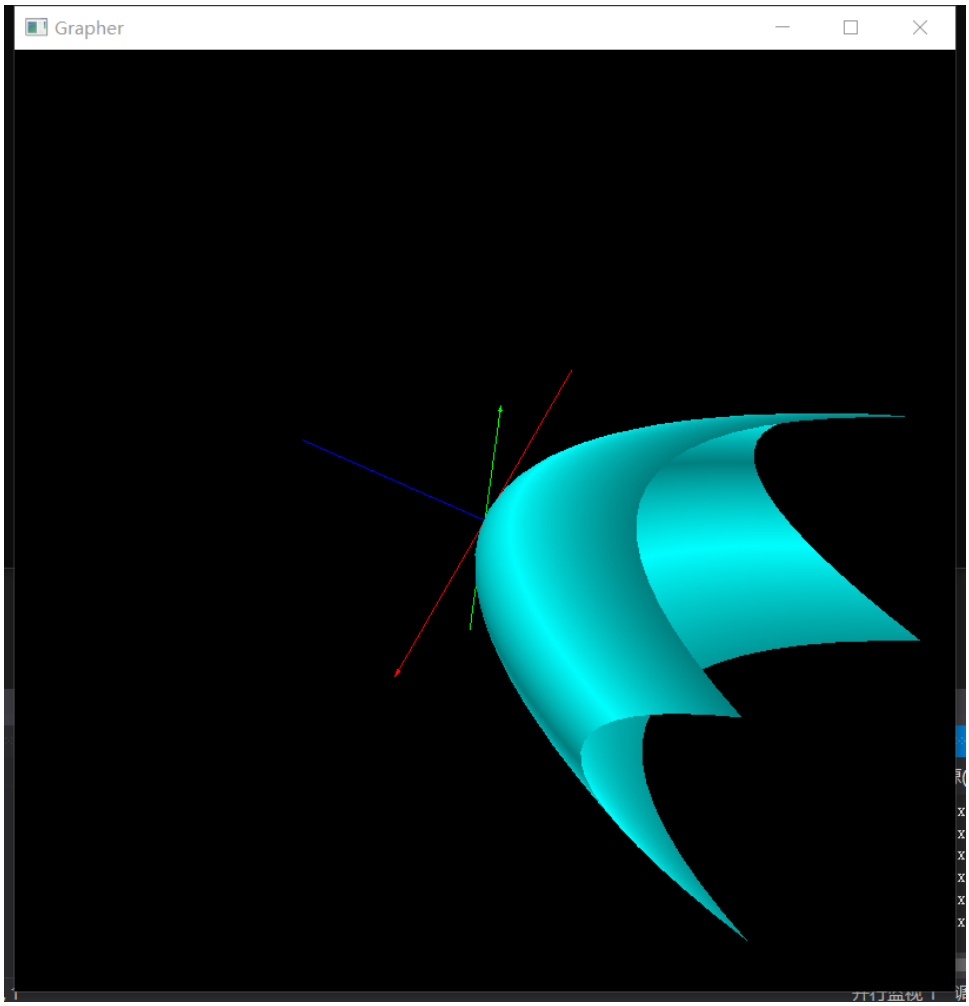


Figure6. ： 简单三维 $z=x^2+y^2$在[-1.0,1.0]*[-1.0,1.0] 上为放缩旋转前，右为放缩旋转后，测试支持观察模式

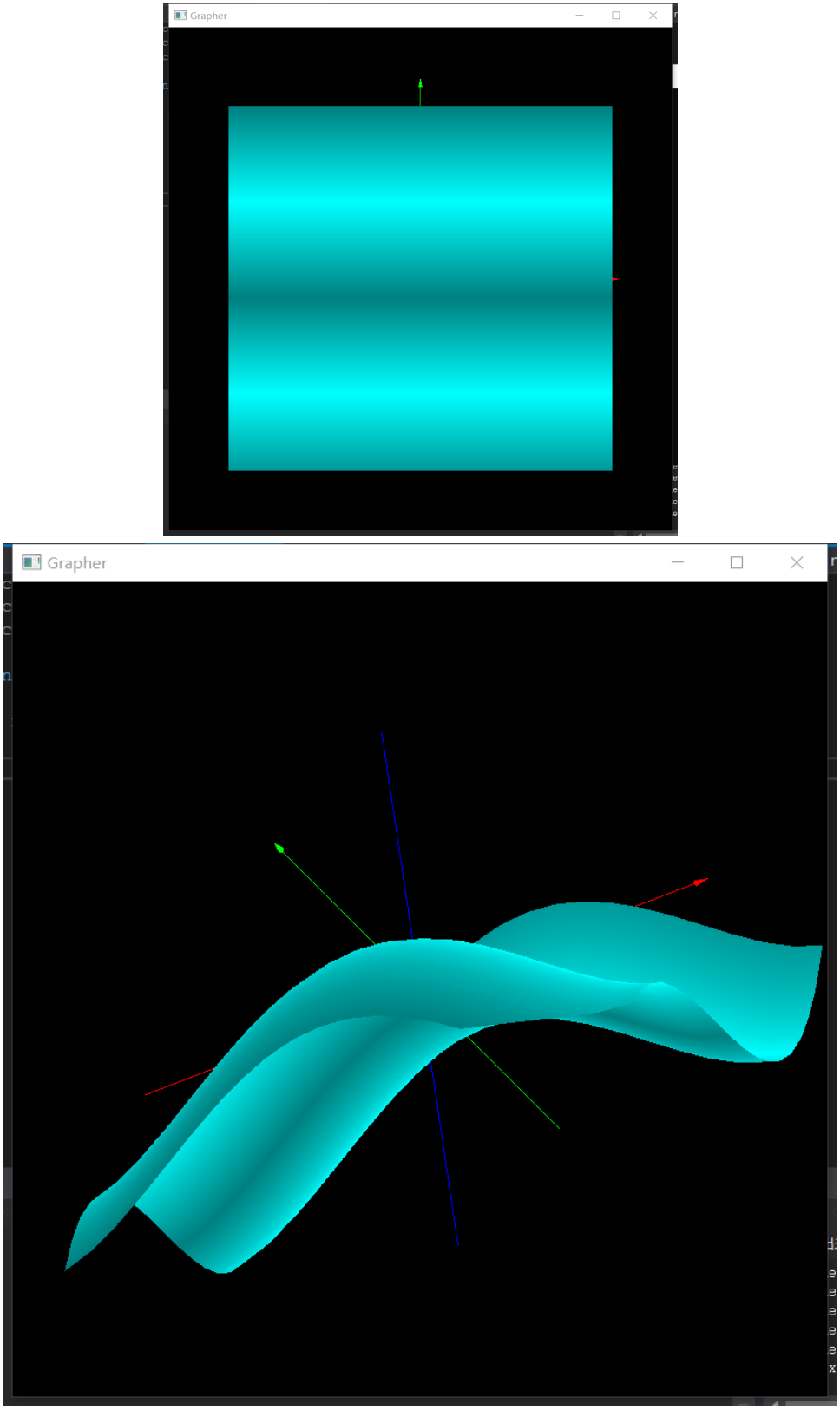


Figure7. ： 复杂三维 $z=x^2*\cos(x)+y^2*\sin(y)$在[-1.0,1.0]*[-1.0,1.0] 上为放缩旋转前，右为放缩旋转后，测试支持观察模式

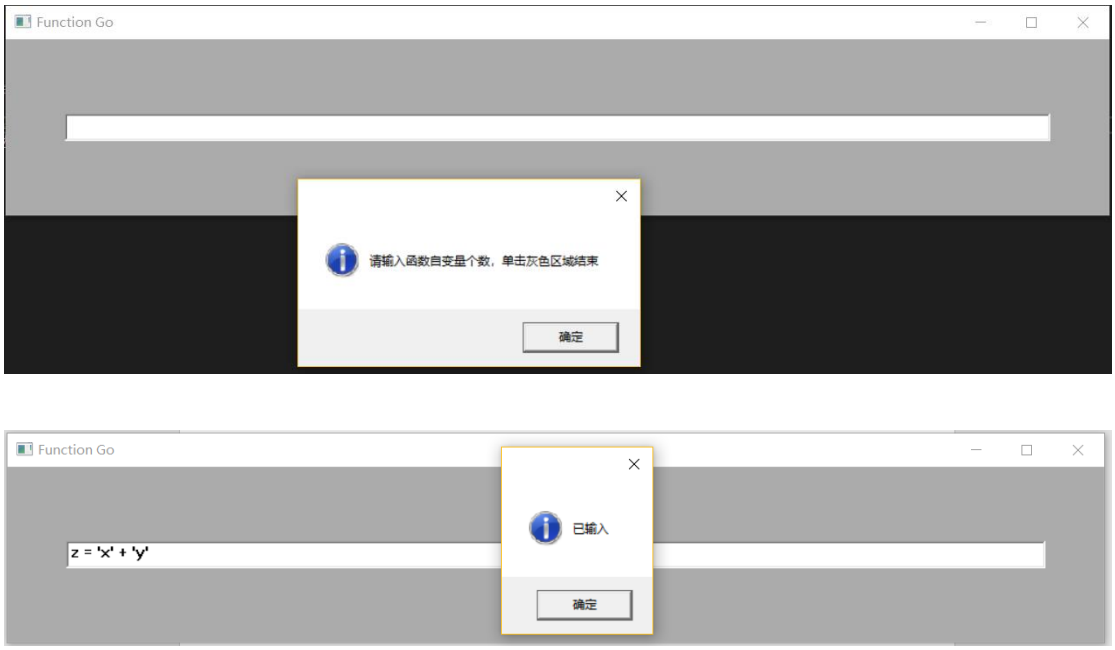


Figure8：Main 模块的测试

小组信息

项目成员：

张冠杰 201730684502

陈　亮 201764681188

夏锐航 201736684186

成员分工：

张冠杰：尝试使用 DirectX 开发无果。完成了 Main 模块的开发。

陈亮：使用 OpenGl 完成绘制模块的开发。并实现了绘图部分的交互功能。

夏锐航：完成了函数解析模块的开发。并应用多线程提高整体性能。