中期报告

1 技术难点

1.1 DPI适配

DPI适配问题是一个比较难以处理的问题，在本项目中，一位队友的电脑分辨率为4K，放大倍率为250%，在测试桌面程序时，出现了界面显示不正常（缩在一起，且缩放比例异常）以及点击“导入”之后，整体布局发生变化的问题。经过排查，问题产生的原因有2：

1. 测试者的DPI和代码编写者的DPI（2K 125%）有巨大差异，而界面初始化设置的是限定最小大小（setMinimumSize），所以在程序初始化时，界面显示会不正常。
2. 为了达成更好的交互体验，客户端在未导入图片时，在对应位置用和“没有导入图片”意思相近的字符串提示，在点击导入后，提示自动消失，由于该QLabel控件没有字符串填充，所以受到了其余控件的挤压，导致了界面的变形。（这个问题只会在高DPI情况下发生）。

解决方案：调用系统API，获取当前环境的屏幕大小（经过放大倍率计算），对界面的最小大小设置进行调整。代码如下：

```
#include <Windows.h>
inline QSize getAdaptedSize(int width, int height)
{
// sysWidth: resolution_width / expanding rate
// sysHeight: resolution_height / expanding rate
// The dafault size of screen is 1920*1080 125%
// In this case, the size of the software is 960*600.
// In case of different solutionn, change the width and height in
proportion.
// Windows.h MUST be included!!!
    int sysWidth = ::GetSystemMetrics(SM_CXSCREEN);
    int sysHeight = ::GetSystemMetrics(SM_CYSCREEN);
    int adaptedWidth = (width * sysWidth) / 1536;
    int adaptedHeight = (height * sysHeight) / 864;
    return QSize(adaptedWidth, adaptedHeight);
}
void View::initQLayout()
{
    screenSize = getAdaptedSize(960, 600);
    ...
    setMinimumSize(screenSize);
    ...
}
```

1.2 控件动态显示

为了保证用户体验，软件计划像Markdown一样，将Latex渲染结果展示框和Latex公式编辑框组合起来，点击渲染图片即可编辑，编辑结束自动渲染。

为了实现这样的效果，有三件事比较重要：

1. QPlainTextEdit（编辑）和QLabel（渲染）两个控件至少有一个得由分配内存，不能通过ui获得（因为在QGridLayout布局下，同一个grid不能放置两个控件）。
2. 需要重载这两个控件的事件过滤器eventFilter，实现控件切换。
3. 需要利用计时器，记录用户停止编辑的时间，完成自动切换，进行渲染。

实现方法：抓取QPlainTextEdit的QEvent::Leave事件，开始计时器，超时后触发槽，其余时刻刷新计时器。

```
bool View::eventFilter(QObject* watched, QEvent* event)
{
    connect(timer.get(), SIGNAL(timeout()), this,
SLOT(onChangeLatexDisplay()));
    if (watched == latexLabel.get())
    {
        ...
    }
    else if (watched == latexEditor.get())
    {
        qDebug() << event->type();
        if (event->type() == QEvent::Leave)
        {
            timer->start(5000);
        }
        else
        {
            timer->stop();
        }
        return false;
    }
    else if(watched == engineSelectionInterface.get())
    {
        ...
    }
    else
    {
        return QWidget::eventFilter(watched, event);
    }
}
```

1.3 渲染源返回图片无法满足桌面显示要求

本软件Latex公式的渲染调用的网站latex.codecogs.com的API，但是初步实现时，返回的图片分辨率过低，显示不清晰。

解决方案：

1. 使用svg文件来展示渲染结果。
2. 通过向提交dpi要求，来获得足够精度的图片。

其中方案1的难点在于将svg数据显示在QLabel中。实现方法是利用QPainter，代码如下：

```
QSvgRenderer* svg = new QSvgRenderer;
QImage* img = new QImage;
```

```
int width, height;

int cnt = 0;
do {
    if (++cnt >= 2)
    {
        displayErrorMsg(imageData->constData());
        return;
    }
    renderLatexString(imgType);

} while (!(imgType == "svg" && svg->load(*imageData)) && !(imgType !=
"svg" && img->loadFromData(*imageData)));

if (imgType == "svg")
{
    // judge height and width
    ...

    QPixmap* pixmap = new QPixmap(QSize(width,height));
    //img->loadFromData(img_bytes);
    pixmap->fill(Qt::white);
    QPainter painter(pixmap);
    svg->render(&painter);

    latexLabel->setPixmap(*pixmap);
    latexLabel->setAlignment(Qt::AlignCenter);
}
```

方案2的实现在于，在对应网址中加入`\dpi{600}`字段。

1.4 token加密

由于后端调用了一些API，这些API由小组成员注册，为了不泄露小组成员的隐私信息，所以决定对token进行加密，在代码中只留下token的密文和经过加密的`curl_link`函数声明和lib文件。

通过XXX算法（略去不表）和lib文件，一定程度上保护了小组成员的隐私安全。技术难点在于加密算法的实现。

1.5 HTTP请求发送

调用网络API遇到的第一个问题就是HTTP请求的发送。我们选择了广泛使用的libcurl作为实现这一需要的库。在实际使用中，我们遇到了许多棘手的问题。

2020年的今天，HTTPS协议已经广泛使用，但libcurl并不原始支持HTTPS协议，而是需要编译者提供相应SSL库（且SSL库大多是高度平台相关的）。然而官方网站提供的预编译库并没有与SSL库共同编译，导致我们必须自己编译libcurl源码。最初我决定使用OpenSSL库以求全平台支持，然而OpenSSL库的编译异常困难，最终我们选择了与Windows内置的Windows SSPI共同编译以支持HTTPS协议，虽然解决了问题，但也导致了平台相关依赖项的产生。对于其他系统，目前我们只能要求用户提供自己的libcurl动态库。

libcurl提供的HTTP POST方法在使用HTTPS协议时出现了意外情况。我们参考了百度的相关文档，使用`curl_formadd`或是`curl_mime_init`时无法设置POST内容，发送过去的数据似乎已经损坏。最后我们使用了`curl_easy_setopt(curl,CURLOPT_POSTFIELDS,)`函数并手动修改POST内容才完成POST请求。这一问题难以定位，也未在网上找到相关的内容，我们小组目前仍然没有得出问题产生的原因。

此外，libcurl的回调函数接口是纯C形式（其实libcurl整体就是纯C库），使用了老旧的...arg，导致将lambda表达式作为参数传入时会出现回调函数参数不匹配的问题。引起这一问题的原因可能是lambda表达式生成临时匿名类的实现方式，导致函数存在隐含的this指针参数进入了不检查类型的...arg参数列表中。

1.6 API返回值解析

工程使用到的API返回值格式不尽相同。公式渲染使用的API`latex.codecogs.com/`返回的是图片数据，公式识别使用的`https://aip.baidubce.com/rest/2.0/ocr/v1/formula`或`https://api.mathpix.com/v3/latex`返回的是json，公式计算使用的`http://api.wolframalpha.com/v2/query`返回的是xml。

对于传输返回数据的问题，我们在后续迭代中不断优化，最终使得数据从下载到被model使用，只有从libcurl的buffer中构造相关数据结构时进行了一次复制。

对于不同格式的返回值的问题，我们选择了rapidjson用于解析json，tinyxml2用于解析xml，并在model和libcurl之间新增了parser模块进行数据交换的处理。

1.7 Latex解析

将Latex解析为表达式树的难点有很多。首当其冲的便是Latex本身。Latex公式是描述性而非定义性的，它非常适合用于描述公式，但它并不适用于程序的输入。例如Latex中没有明确定义变量，像`\sum_{i=0}^{n}{i^2}`这种带有变量的嵌套结构非常难解析。使用正则表达式难以处理数学表达式内含的优先度问题，而Latex也没有给出任何优先度相关的限制（因为它自始至终只是描述性的，仅仅负责将用户编写的公式渲染出来而不会考虑公式的合法性）。

一般的字符串处理方法难以解决这个非常复杂的解析问题，我们最终选择了`boost::spirit`作为解析器。`boost::spirit`提供了类似EBNF的语法。然而，在后续的算法设计中，我们发现普通的解析+即时计算的算法无法满足要求，必须建立抽象语法树。再举上一例提到的sum，sum导致表达式的计算无法简单地后序遍历，而需要构造AST并前序遍历寻找迭代性质的节点，对这个节点的子树迭代计算。当这些迭代性的节点形成嵌套结构时情况会更加复杂。尤其当这类节点处于AST上层时，减少计算量是极其必要的（例如AST的根节点就是SUM，那么整棵树都要被计算SUM的迭代次数次，而其中相当一部分子项可能可以通过数学方法优化），因而AST还需要进行等价变换，将迭代性节点尽量移动到树的下方。

在技术分析的环节中我们已经意识到，这个解析器的构建已经超出了我们当前的能力范围，我们也没有找到Latex解析相关的开源项目，最终只能使用商用性质的wolframalpha API进行相关计算。

1.8 表达式树构建

对于表达式树的建立，由于在Latex解析为表达式树的环节遇到了困难，本部分也暂时搁置了。由于鞠紫盈同学能力有限，构建出的表达式树虽然可以完成简单的计算功能，但是结构不合理，构建表达式树时必须对每一个节点分别处理，对于节点是符号还是变量还是常量需要人工确认并赋值，很难满足要求，因此最终未能实现。

2 协作情况

2.1 第一轮迭代

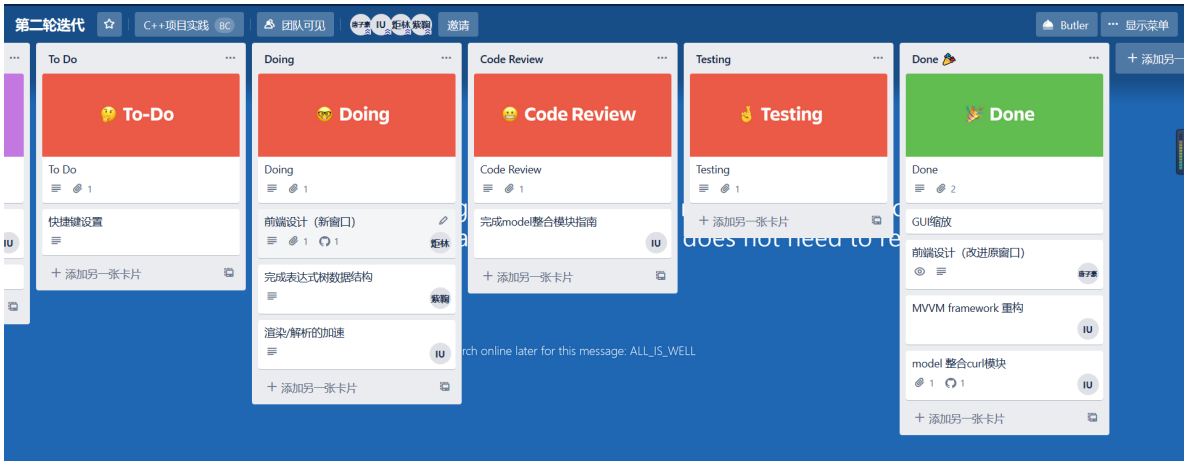
- MVVM框架搭建：徐江雨
- GUI框架搭建：唐子豪
- 表达式树设计：鞠紫盈、林炬乙
- 公式识别接口设计：徐江雨
- 公式渲染接口设计：唐子豪
- 公式识别令牌加密：唐子豪

2.2 第二轮迭代

- 表达式树设计（Algorithm层）：鞠紫盈
- GUI优化（View层）：唐子豪
- 计算界面设计（View层）：林炬乙
- CppCurl封装测试，json解析（Model层）：徐江雨
- 整体结构优化：徐江雨
- ViewModel层：徐江雨

2.3 Trello协作展示

小组协作使用Trello进行协作，下面展示Trello协作情况。

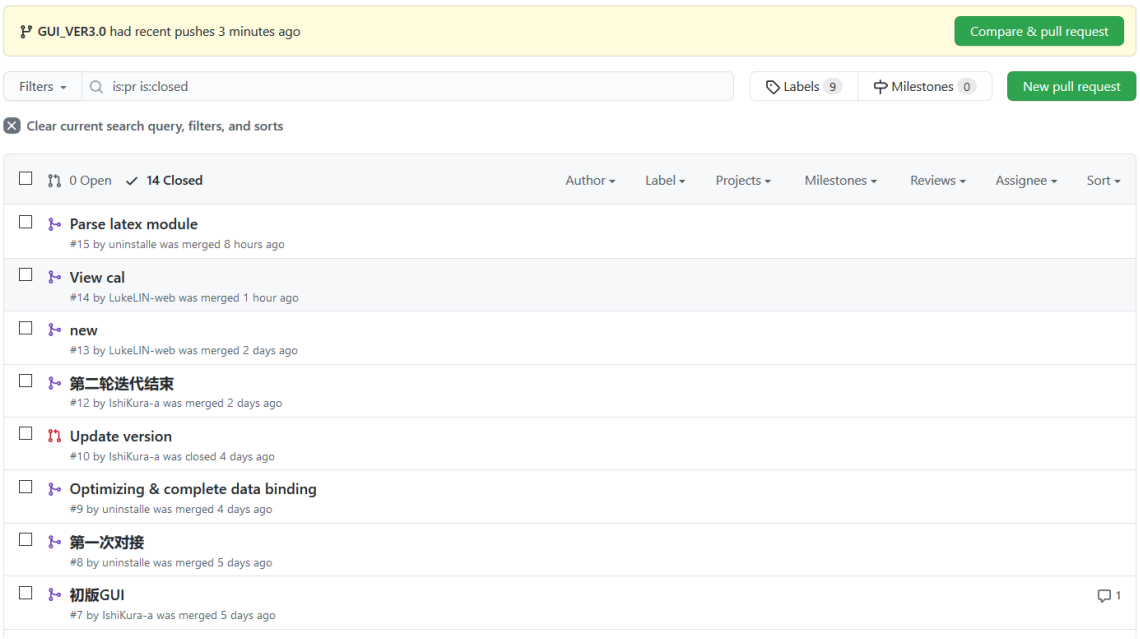


2.4 GitHub协作情况

GitHub链接：

https://github.com/IshiKura-a/CPP_PROJECT_2020

Pull Request使用情况：



2.5 目前进度

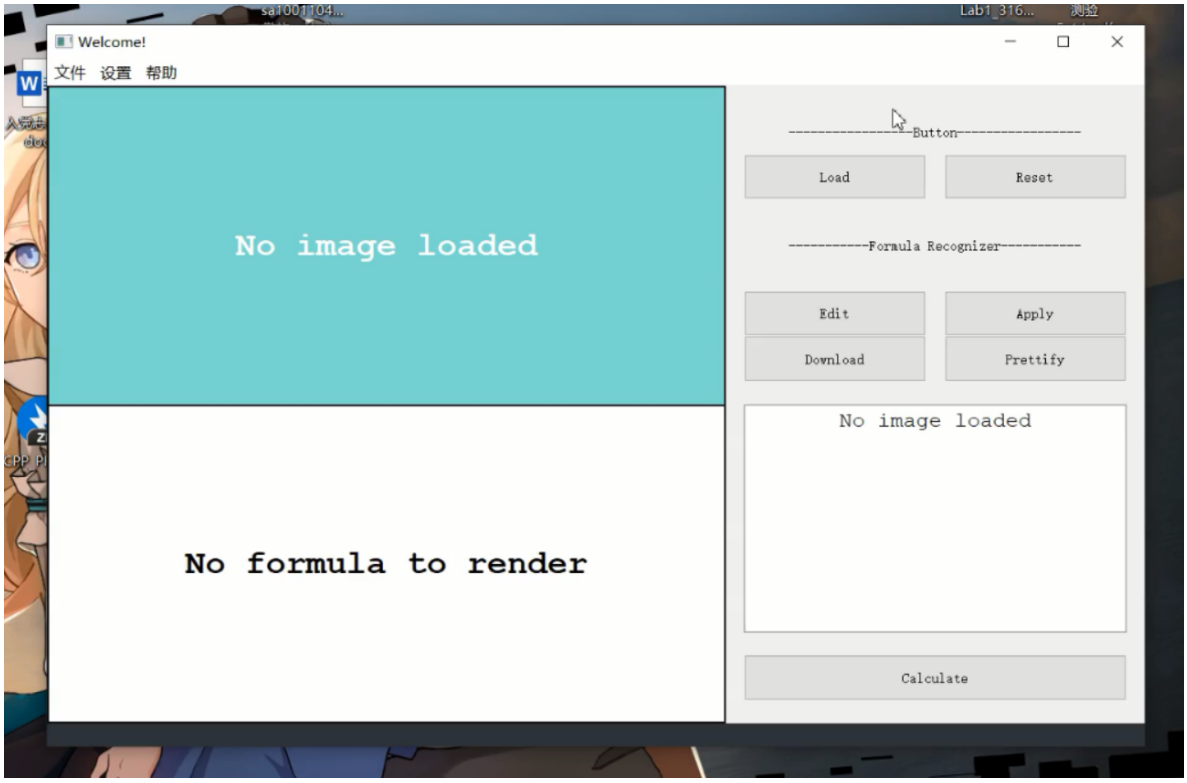
第二轮迭代基本实现

3 部分效果图

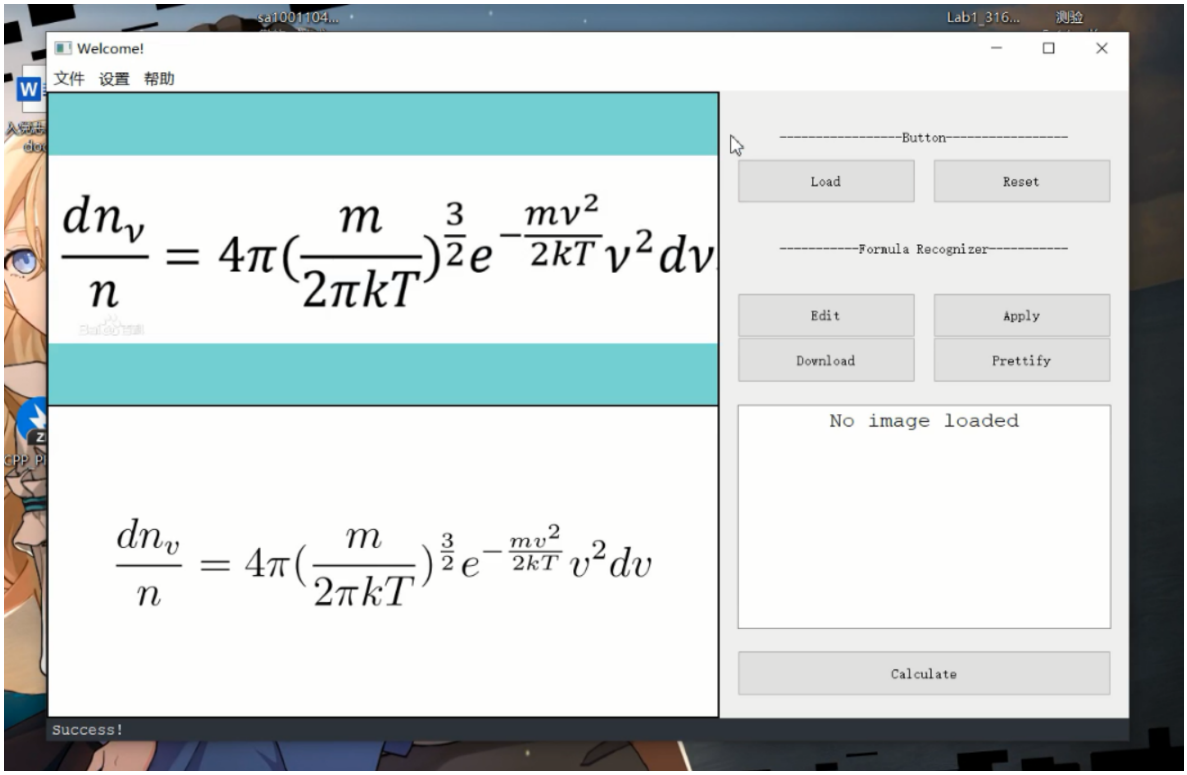
整体测试视频链接：

https://raw.githubusercontent.com/IshiKura-a/CPP_PROJECT_2020/master/doc/video/001.mp4

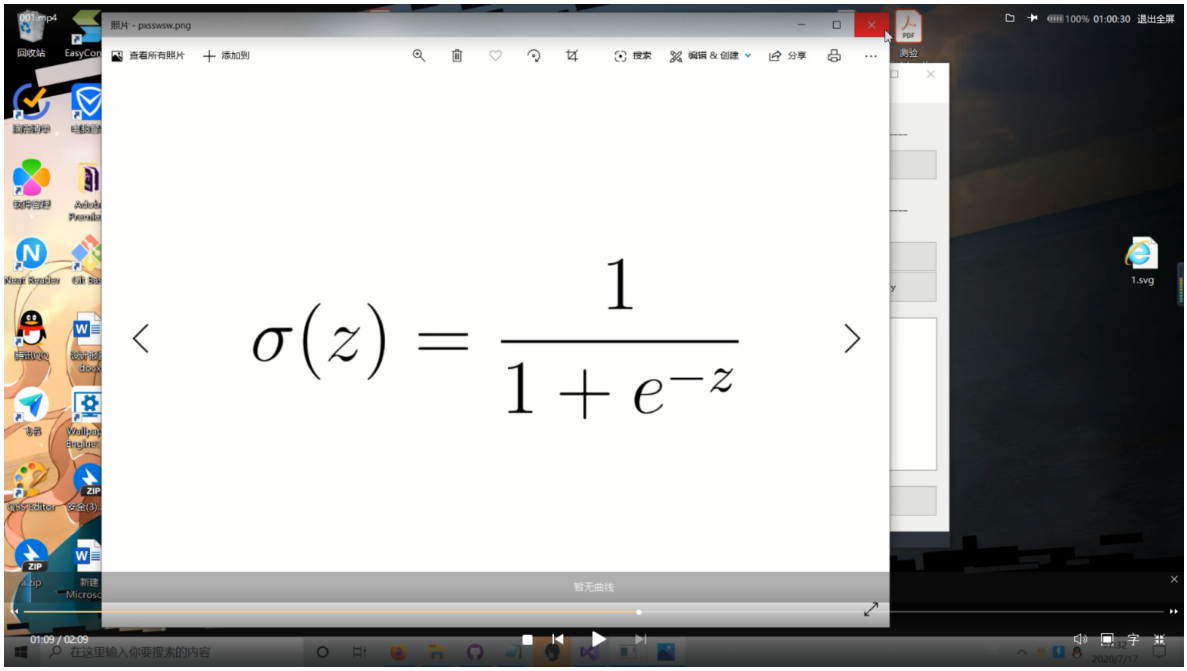
整体界面如图所示：



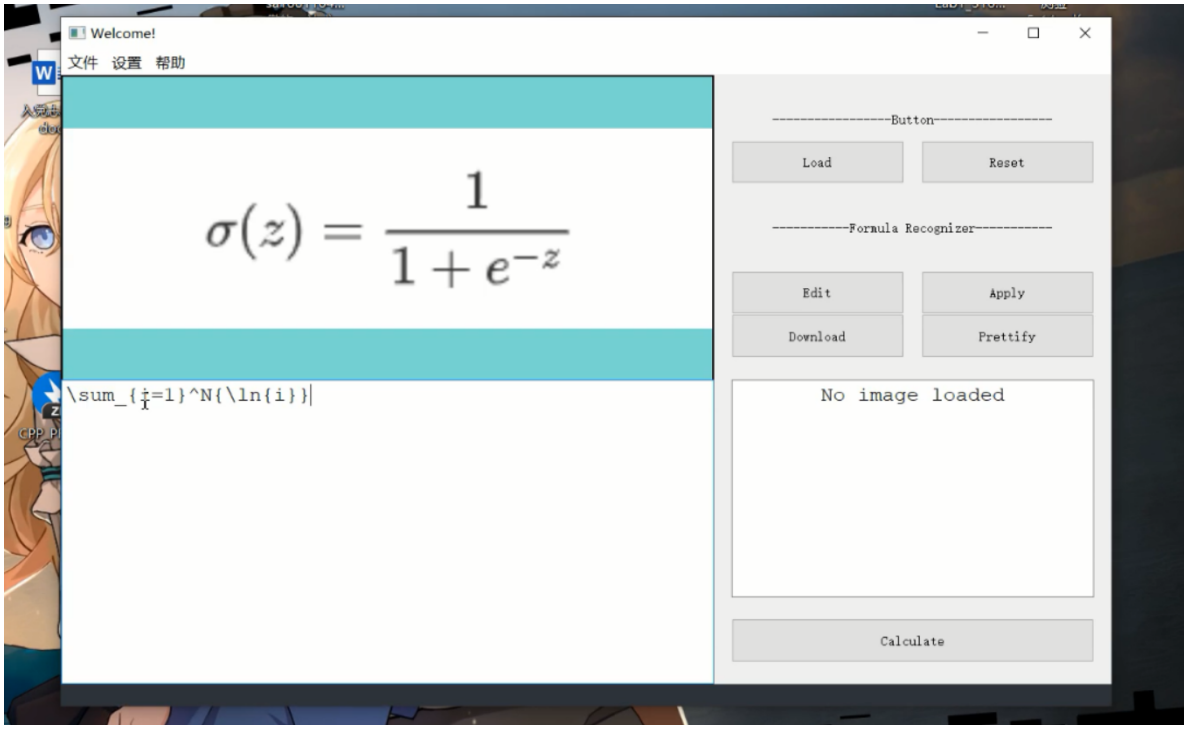
导入Latex公式后，界面如下图所示，注意下方有状态提示。

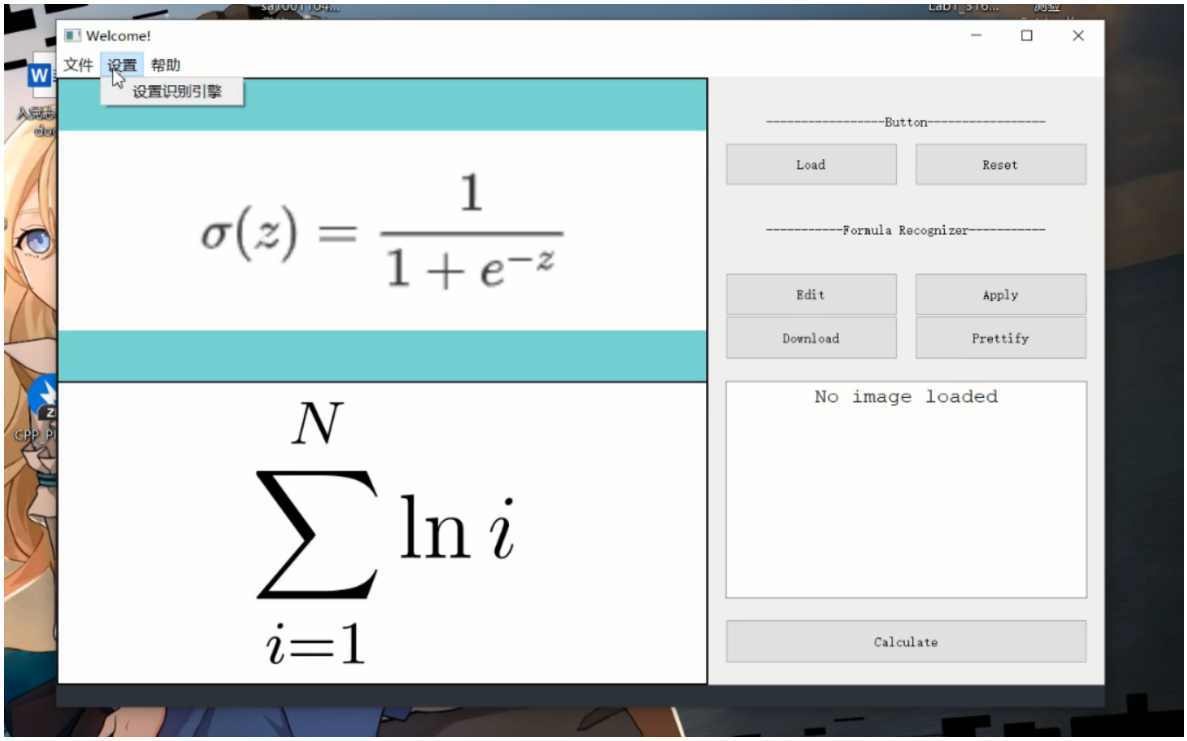


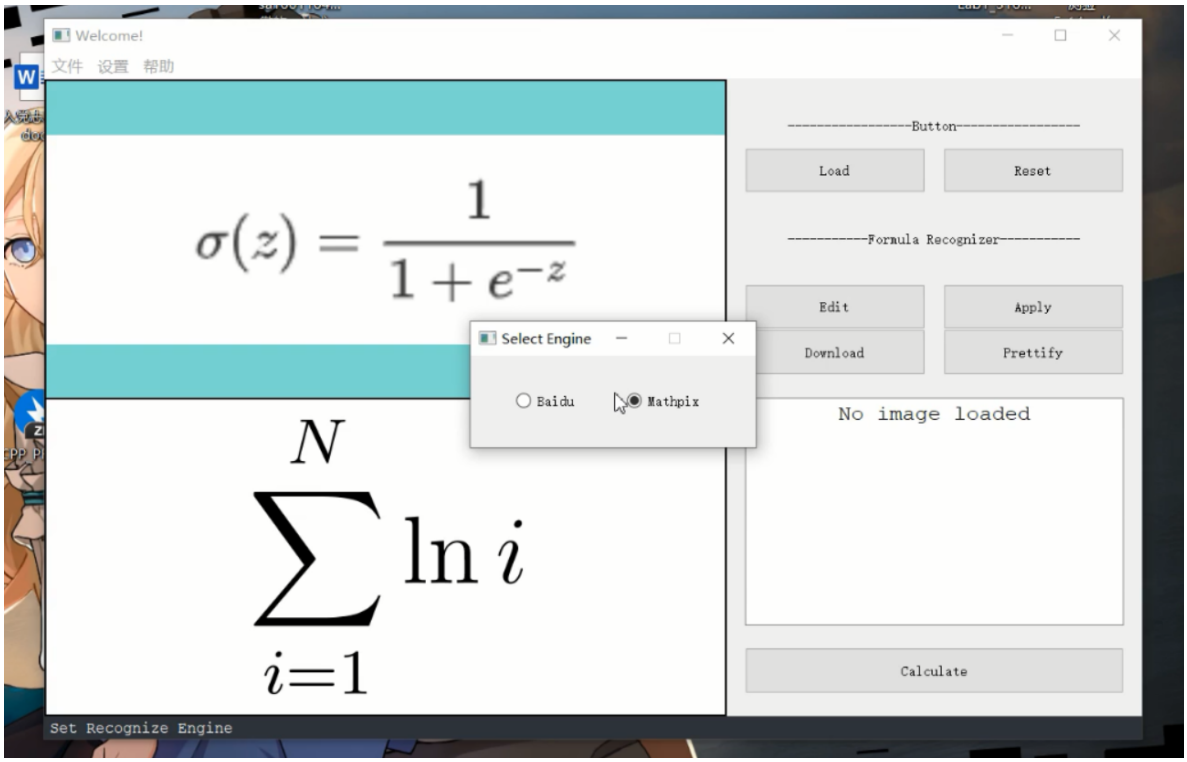
点击Download，可以下载渲染结果。

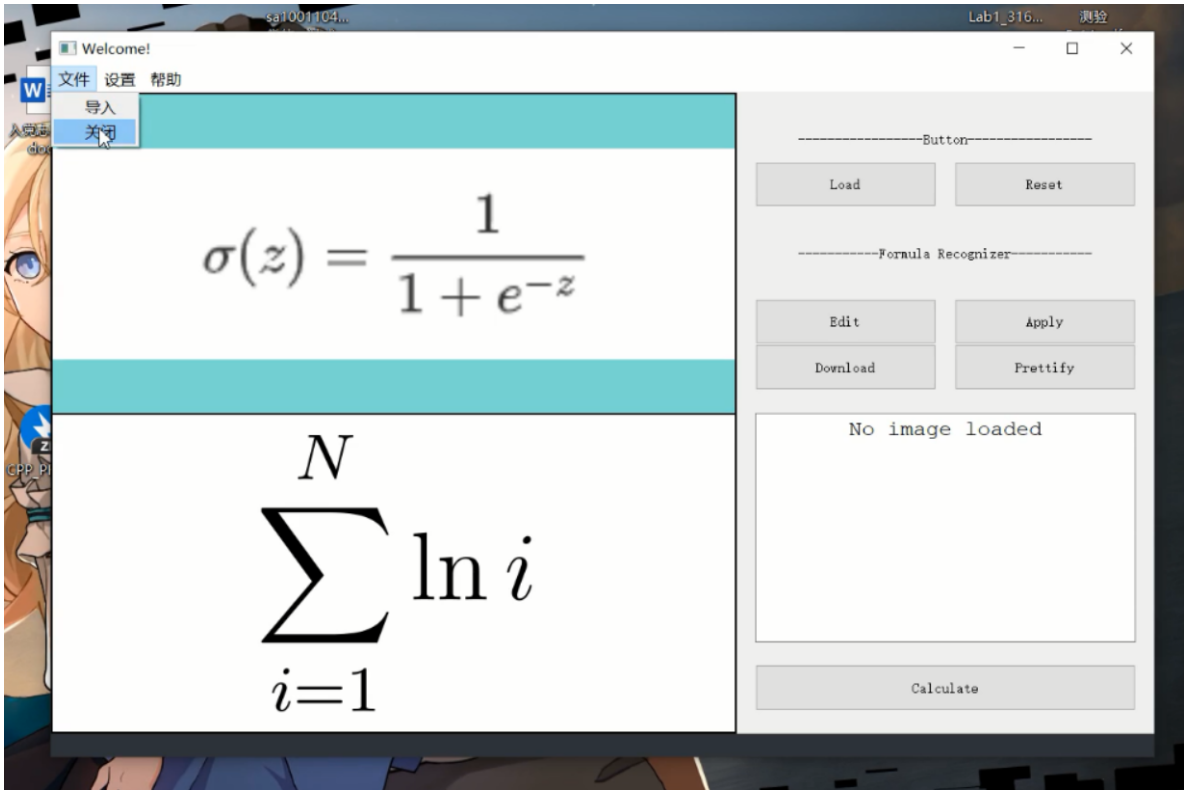

照片 - pxsswsw.png
$$\sigma(z) = \frac{1}{1+e^{-z}}$$
暂无曲线

菜单栏：


Welcome!
文件 设置 帮助
$$\sigma(z) = \frac{1}{1+e^{-z}}$$
\sum_{i=1}^N{\ln{i}}
----------Button----------
Load
Reset
----------Formula Recognizer----------
Edit
Apply
Download
Prettify
No image loaded
Calculate


Welcome!
文件 设置 帮助
设置识别引擎
$$\sigma(z) = \frac{1}{1+e^{-z}}$$
$$\sum_{i=1}^{N} \ln i$$
----------Button----------
Load
Reset
----------Formula Recognizer----------
Edit
Apply
Download
Prettify
No image loaded
Calculate


Welcome!
文件 设置 帮助
$$\sigma(z) = \frac{1}{1+e^{-z}}$$
$$\sum_{i=1}^{N} \ln i$$
Select Engine
Baidu
Mathpix
----------Button----------
Load
Reset
----------Formula Recognizer----------
Edit
Apply
Download
Prettify
No image loaded
Calculate
Set Recognize Engine



4 心得

4.1 总体心得

从两对互不相识的人，到现在C++项目的队友，这样的转变对所有人来说都是一个巨大的机遇与挑战。一开始，我们闷头开发，各自耕自己的一亩三分地，因为对MVVM框架不够熟悉，做出来的东西虽有其形，但无其神，小组协作在一定程度上存在着等待的情况。在第一轮迭代展示之后，我们小组决心改头换面，重构了代码，通过Trello明确分工和迭代计划，分配工作，各司其职。虽然，小组还存在着拖DDL的现象，但是比起几天前那个闷声各干各的小组，已经有了大幅地改善。如今，我们的项目进度已经过半，逐渐变漂亮的GUI，逐渐丰富的功能，逐渐改善的用户体验.....我们看着自己的产品不断地更新迭代，不断地向完美前进，所有人都感到自豪与满足。

在这次项目实践中，我们收获了许许多多的新知识，熟悉了GitHub, Trello, Markdown, Drawio, AppVeyor等各种工具，我们互相学习，互相交流心得，互相取长补短，在这短短的十几天里，取得了长足的进步。

4.2 个人心得-唐子豪

小学期项目来到中期，感觉自己收获颇丰。在这十几天里，我对MVVM框架有了一定的认识，也对C++有了新的看法。C++项目实践这门小学期，从工程的角度，给我们展示了实际应用中的C++，可谓是刷新了我的C++三观。在这里，我学到了C++的新特性，包括lambda表达式和std::any等等。在我实际编写代码的过程中，我也学会了很多，首先是对Qt和C++的了解更加深入了，自己的码力提升了；其次，我分析问题、解决问题的能力也在一次次debug中得到了提升，虽然每次遇到自己不能复现的bug，头疼得要死，但是解决之后的如释重负和愉悦更为珍贵。我在这次小学期中也学会了许多的工具，使用GitHub，Trello等更加得心应手了。最后，在小组协作中，我也做到了学他人之长，摒自己之短，从队友身上学到了很多知识，非常有收获。

4.3 个人心得-徐江雨

在本次项目中我负责了MVVM框架的编写和后端模块的实现。在编写MVVM框架时，我对demo的代码实际上颇有不满，尤其是Command接口的笨拙，最终选择自己从零开始实现框架。因而在MVVM框架的实现中，也存在不少因我个人的喜好而改变的地方，例如由于notification必须被硬编码在类定义中限制了扩展性，所以采用了静态方法注册事件回调方法等设计。对于Command，则是采用了`std::function<void()>`和`std::function<void(std::any)>`作为替代。作为编写者，虽然我想自夸已经完全理解了MVVM框架的设计，但当前经过两次重构的MVVM框架在第二次迭代末期仍然发现了数据同步BUG，使我不得不收声。尤其遗憾于没有挖掘这一框架异步的潜力，当前的同步的实现使得等待API返回结果时会根据用户的网络情况产生或长或短的无响应时间，实在有损用户体验。

作为框架和后端几个功能模块的编写者，我经常需要与前端设计者一起debug。在这一过程中，我碰到了不少次我能复现而别人不能的问题（我就是那个4K 250%缩放的用户），发现了许多曾经从未思考过的影响因素，也因此积累了不少debug的经验。由于后端有模块使用了DLL形式，为了简化繁琐的编译过程，我也去学习了一下VS工程的结构和设置，成功让编译->复制粘贴->编译简化为一步到位。在对功能模块进行技术分析的过程中，我也了解了许多其他课程的知识，尽管技术分析的结果一般总是滑落到寻求已有API的支持，但在放弃之前的思考与尝试设计带来的收获并没有消失。

4.4 个人心得-林炬乙

小学期项目来到中期,首先感谢老师和队友给了我很多指导,让我学会了怎么解决编译问题,我用的vs2017平台,队友想办法让我自己编译了.lib文件,成功解决了libcurl库的无法解析问题.也让我了解了很多新颖的工具如Trello这样的分工工具.课程一开始几天是学习了GitHub,了解冲突发生的原因,进行push和merge.然后学习MVVM框架,每一天感觉都学到了非常多的东西,也对工程上持续部署等有了概念,知道实际应用中不是文件传来传去,而是可以同时编写,可以方便地push,部署.然后从对Qt框架一无所知,到几天学了很多Qt的框架,属性,槽函数信号传递等内容.也对前端的布局工作有了一定了解.
在程序学习中,还了解了很多原来了解的代码之外的计算机问题,比如从GitHub上可以方便地下载腾讯开源的代码库, libcurl库的编译需要openssl, json的返回需要rapidjson,对vs项目的依赖项和配置了解深入了很多.

4.5 个人心得-鞠紫盈

小学期来到中期了，首先感谢队友对我的包容和指导，让我了解了很多之前我几乎没怎么接触的东西，例如大型项目的协作机制，还有GitHub和Trello的使用方法，头一次打破了我过去团队协作中互相传文件，功能设计很难整合等的问题。同时这次工程也让我初步接触了Qt这个对我来说较为新鲜的概念，也对使用Qt和VS结合开发的工程有了初步的了解，同时也初步接触了MVVM框架。同时深刻的意识到自己对于C++的了解过于浅薄，很多新的、优秀的C++功能自己都不了解。在表达式树的设计中我遇到了很多问题，也感谢组内队友给了我很多建议和指导。