

**实验报告**

|  |  |
| --- | --- |
| **课程名称**: | 设计模式 |
| **开课学期**: | 2020-2021学年 第2学期 |
| **专业**: | 软件工程 |
| **班级年级**: | 2018级 |
| **学生姓名**: | 宋行健 |
| **学号:** | 222018321062006 |
| **实验教师:** | 王晓蒙 |

**计算机与信息科学学院 软件学院**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | | 实验4结构型模式1 | | | |
| 实验时间 | | 2021/4/6 | Type | \*验证性□设计性 □综合性 | |
| 1. 思路  首先对老师发的代码进行逐句研读，了解代码架构，并绘制类图。通过类图的展示，程序的逻辑逐渐明显。程序主要被封装在GIFApp的类中，在这个类中进行了页面的布局和各种控件响应函数的创建。第二重要的类是Render，在这个类中，进行了字幕和GIF的渲染。其中很多渲染的参数都是写死的，比如字幕的位置、字号、颜色等，这个可以在改进中进行优化。第三个比较重要的类是Frame，这个类将GIF进行的每一帧了封装。  2. 类图    3. 代码分析——适配器模式  在程序中两处地方使用到了适配器模式，分别是GIF的读取和GIF的写入。在程序中我们操作的都是Frame类型的帧数据，而数据源（计算机中储存的）是GIF格式的数据，这里我们就用到了适配器的思想。GIFReader适配器将GIF转换成Frame，GIFWriter适配器将Frame数组转换成GIF。    这里的读写两个适配器均使用了对象适配器的结构。这里我以GIFReader适配器为例进行说明，GIFWriter同理。  IGIFReader定义了一个接口，GIFReader作为一个适配器去实现这个接口，在适配器中声明一个GifDecoder对象，通过这个对象调用GIF源数据格式的方法，并进行逐帧读取，并转换成Frame类数组。    4. 改进  **（一）导出操作弹窗**  原程序在导出操作后没有反馈，用户并不知道文件是否导出成功，也不知道文件导出到了哪里。因此，我添加了导出时的路径信息弹窗，并同时在命令行输出。在导出之前添加了一个判断，如果未导入GIF则弹框警告。    **（二）原尺寸导出GIF**  原程序中导出的GIF是固定宽高的，这个宽高是写死在程序中的，只能是和canvas一样。我在Frame类中加入了两个参数，分别是GIF的宽和高。通过在读入GIF时同步调整canvas的宽高来调整GIF宽高，使导出的GIF保持原比例。      **（三）“选择文件”的默认路径为根目录**  原程序中点击“选择文件”打开的文件选择窗口默认为系统根目录，每次都要层层进入到实验的文件夹去选择，十分麻烦。经过查资料，我发现了FileChooser类中有一个设置默认打开位置的方法，setInitialDirectory，设置路径为“.”，即为当前程序路径。    **（四）实现字幕位置拖拽**  原程序中字幕的位置是固定的，其主要是因为在Render类中，将字幕的位置进行了固定的渲染。我的修改是在渲染的方法，rendering()中加入了两个参数，textX和textY，分别表示字幕位置的横纵坐标。通过在GIFApp主类中对渲染GIF的画布canvas对象进行鼠标拖拽事件的监听，并将鼠标的位置坐标缓存后传入渲染函数，对字幕进行渲染。      **（五）超出GIF帧范围提示错误框**  原程序中的帧范围是没有检验的，当用户填写的帧范围是不合理的情况下，程序会报错，我在渲染字幕之前，添加了一步检验的工作，如果帧范围超出了GIF的范围，则不进行渲染，并弹出警告提示框，如果帧范围合理，则弹出成功提示框，并显示添加字幕的信息。    **（六）添加字幕颜色选择器**  原程序中的字幕是默认蓝色的，我这里在GIFApp中添加了一个ColorPicker颜色选择器，并将选择的结果传入Render进行渲染。但是这里美中不足的是，只能选择所有字幕的颜色，不能每条字幕单独选不同颜色。      **（七）自定义字幕大小**  原程序中的字幕大小是固定32px，我在GIFApp中添加了一个输入框让用后可以输入字号，并将用户输入的字号读取后传入Render进行渲染。    5. 运行结果          6. 总结  该项目是适配器模式的实现。其中GIF的读取和写入都使用到了对象适配器模式。都是通过在适配器中创建一个需要被适配对象，并通过这个对象调用需要被适配的类中的方法，进而将其转化为适配后的类。在程序中就是，GIFReader适配器将GifDecoder转换成了Frame。  在本次实验中，我对原程序进行了较多的修改，其中主要是从Render类入手，因为这个类中的rendering函数中对GIF的渲染时有很多都是写死的数据，例如字幕的位置、大小、颜色等，这些都是可以通过前端传参进行修改的。另一方面是增加了一些人性化的弹窗提示等。其中我认为最困难的是对字幕拖拽的处理，因为这里的字幕不是一个实例，而是直接渲染在GraphicsContext上面的，所以不能用网上常规的方式进行鼠标拖拽。这里我采用了监听鼠标按下时的位置，传入Render，在鼠标的位置直接渲染字幕。看似是一个拖拽的功能，实则是一个根据鼠标位置动态渲染的过程。  我在实验中也遇到了一些问题，例如导出的图片总是找不到位置，后来我发现是因为我的文件目录中含有不止一个“gif”，所以在替换的时候全部都替换掉了。针对这个问题我暂时还没有很好的解决方案，只好把文件路径重命名了，但是这样毕竟不是长久之计，以后还是要想办法把这个问题解决掉。 | | | | | |
| 教师评阅 | 实验内容（60%）： | | | |  |
| 实验思路（20%）： | | | |  |
| 实验总结（20%）： | | | |  |
| 实验成绩： | | | | |