

VEP

基于Vaa3D-x的可视化编程模块

1. 依赖与安装

Pyside6

```
pip install pyside6
```

2. 项目文件组织

- src: Py交互界面源码
 - editor: 所有交互组件构造与功能
 - nodes: 用于存放自定义Node节点
 - qss: 存放样式表
 - structs: 存放处理py层级数据结构
 - widgets: 存在qt控件级别
 - tools: 内嵌项目工具
 - main.py: 项目运行入口
 - vep_xxx: nodes, ports, edge, 关注三个交互组件即可
 - model: json处理
 - viewModel: 建的玩

3. 结构简介

1. view: 试图
2. control: 流的控制
3. model: 对传统模型的封装

View

editor 编辑器

期内有载体, graph图, 用于存放可视化编程的界面

内有node节点, 用于承载特定的功能

对于每一个Node来说, 名为GraphNode, input,output, exec作具体功能执行处, NodePort, NodeTitle, Position, NodeEdge

技术选型

使用qt而非electron, 两者均为桌面应用开发框架, qt为c++,electron多为node-js。

- qt拥有Qt for python
- 可封装python, 对于算法使用更方便
- 美观程度, qt可使用QML or QSS进行样式调节, 对应在html的CSS, 可操作空间较大




4. 项目设置

Target: 实现每个节点的可视化

PySide6: qt6.4对python的GUI图形库

功能

基本功能

- ✓ 节点、边、端口、画布的绘制 
- ✓ editor的基本样式及布局
- ✓ Graph的执行
- ☐ 一套新的Graph的执行方法,可离线执行
 - **需要脱离UI执行 (View, Node, Edge, Port)**
 - 使用的方法是在四类基础上, 重新创了了running_graph running_node running_edge running_pin用于存储运行时的数据
 - 现有的Node自定义机制是在running_node的基础上自定义, 即RunningNode改名为Node完全替代Node, 原先的Node改名为TemplateNode用于创建GraphNode
 - 目前的对象关系 GraphNode > TemplateNode Node创建TemplateNode 自定义Node继承Node
 - 运行之前, 将GraphNode、port相关数据同步到Node内
 - 异步执行确实可以实现, 并且不会卡顿主线程。子线程使用sleep后主线程仍然可以操作。
 - 仍然存在的问题: 异步运行与变量相关的数据仍然会计算错误, 并且数据量大的循环, 主线程仍然卡顿。问题原因: 异步执行使用的是主线程的variableManager进行修改变量值, 通过信号改变变量值, 通过直接get获得值, 发现信号发送执行并非即时的, 甚至优先级会很低, 导致变量值更新不及时, 使用MoveToThread进行变量转移。
 - 变量转移之后, 出现新的问题, 因为自己定义的VariableManager没有定义copy方法, 导致自己将主线程的variableManager转移后, 主线程不再有variableManager, 从而使得主线程添加UI会抛出NoneType异常。
 - 运行方式改成三种: 正常运行 (目前的background运行), 离线运行 (脱离程序一样运行, 需要重开一个python进程, 类似需要根据graph文件进行运行), node运行 (单个node的运行)
 - **Session机制重新制定, 有些节点可以跳过Session**
 - 目前只是添加了一个refresh_along_session的参数用于判断节点是否会随着session进行刷新, 存在的问题, 当双重循环时, 节点是需要刷新的 
 - 没有连接节点的port对应的pin不会刷新
 - 仍然存在的问题, File节点需要特殊操作, 需要在离线运行之前运行。 
 - **异步执行, background执行, 线程交互, 显示每一个节点的状态**
 - 目前定义了几种简单的节点运行状态, 主线程与子线程交互主要是通过信号进行, 但是目前任然存在变量修改卡顿的问题。
 - 卡顿问题解决, 创建running_graph作为图运行的入口, 并且使用moveToThread将running-graph移动到thread线程内, 卡顿的原因是将variableManager直接复制, 但是view仍然与该manager存在关联导致GraphicsView卡顿。解决方法: 完全从最基础的变量类型进行创建variablemanager,或者实现variable的copy方法。

- 线程池及线程管理器

- editor具有一个线程池
- 每一个graph绑定一个线程

☒ PortWidget的定义，只定义输入PortWidget

☐ NodeWidget相关的框架定义

- 初步定义一下几种类型:

- TitleWidget 显示在node的title栏目，从右向左排列
- Port Widget 显示在node的input pin后面，接受一些简单的输入
- Preview Widget 主要是显示的功能，preview的是input的值
- ActionWidget: 一个按钮 以及对应的事件，例如动态添加一个pin
- 目前简单完成了TableWidget，需要将title widget以及node widget进行封装，以及联动
- 动态改变Node Port的数量

☒ 图文件的保存与加载

☐ Node的Signal的定义

- 目前只是简单的定义了一些运行状态的signal，需要进一步的细化和优化

☐ 不同节点连线时自动添加转换节点

☐ 配置文件功能，保存编辑器基本信息

☐ 工作区选择,新建project，保存项目基本信息

☐ 可拓展的插件框架

☒ 导出为EXE可执行文件

- 使用auto-py-to-exe对文件进行导出(本质上是对Pyinstaller的封装，只需要点击就行了，不需要写那么多参数)
- 需要注意的是需要使用--add-data将python的Site-pakcge/PySide6/plugins/添加为PySide6/plugins，参考https://blog.csdn.net/qg_41730930/article/details/112612864
- 其他需要用到的package例如pandas，需要使用--hidden-import进行引入。
- 自己项目内的package以及代码，需要使用--add-data映射为代码内的路径
- 资源文件根据程序内的路径，映射为对应路径,one folder的模式是可以直接寻址的
- onfile模式 需要使用虚拟路径解决,包括QSS内的路径可以使用正则表达式进行替换。

```
def resource_path(relative_path):  
    """ Get absolute path to resource, works for dev and for PyInstaller  
    """  
    try:  
        # PyInstaller creates a temp folder and stores path in _MEIPASS  
        base_path = sys._MEIPASS  
    except Exception:  
        base_path = os.path.abspath(".")  
  
    path = os.path.join(base_path, relative_path)  
    return path
```

- icon设置时需要注意，windows使用icon缓存的，想要实时出现需要导出时换个名字。图标ico格式尽量包含所有大小的图标。

扩展功能

- ☒ 添加侧边栏变量栏
- ☒ 变量详情页面
- ☒ 获得变量、设置变量的节点，拖动是get，alt+拖动是set
- ☒ Logger功能
 - 重新修改了Logger的右键菜单，目前的问题是快捷键问题。
 - 更多的选项功能，例如日志保存，日志筛选等 🕒
 - 添加一行操作栏，类似VScode。
 - 这个logger最大的问题就是超巨量输出时会卡主UI，目前还不知道怎么解决。 ❌
- ☐ Command Line功能，可以和Graph进行交互
 - 添加函数功能，引用graph函数的功能
 - 节点合并功能

节点的操作

- ☒ 节点选择 SelectAll
- ☒ 节点复制、剪切、粘贴
- ☒ 节点删除
- ☒ 节点的横向对齐以及纵向对齐（强迫症福音）
 - ☒ 水平居中对齐 (H)
 - ☒ 垂直居中对齐 (V)
 - ☒ 连接端口对齐 (Q)
 - ☒ 水平左对齐 (Shift+L)
 - ☒ 水平右对齐 (Shift+R)
 - ☒ 垂直向上对齐 (Shift+T)
 - ☒ 垂直向下对齐 (Shift+B)
 - ☒ 水平均匀分布 (Shift+H)
 - ☒ 垂直均匀分布 (Shift+V)
- ☒ 将选择的节点导出为图片
- ☐ 执行时的状态显示

自定义节点

- ☒ 使用Node类子类声明进行自定义
- ☒ 使用修饰器进行自定义
- ☐ 图形化自定义节点

节点库及扩充--基本节点

- ☒ Node库列表栏，并实现拖拽添加
- ☒ 右键菜单的库列表实现双击添加
- ☐ Preview Nodes 查看各种类型的操作
- ☐ Num 数值操作 加减乘除等 ✅

- ☐ String 操作的Nodes 基础操作 
- ☐ List 操作的Nodes 增删改查 
- ☐ Dict 操作的Nodes 增删改查 
- ☐ Table 操作的Nodes, 创建、合并等等 
- ☐ network 操作的Nodes

更好的编译器

画好网络之后，对graph进行编译执行。

- ☐ 生成指定python代码
- ☐ 更全功能的编译