链滴

一款支持 markdown 和无限画布的自由嵌套图形化笔记软件 NodeNote

作者：babyQ033

原文链接：https://ld246.com/article/1642140398091

来源网站：链滴

许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

这是项目链接，喜欢的小伙伴可以帮忙点点star吗，最近校压力有点大哈哈哈，感谢了：

NodeNote

这是v2.19介绍视频，此时暂无markdown内容，主要介绍笔记思想

<iframe src="https://player.bilibili.com/player.html?aid=849841725&bvid=BV1PL4y1p7eb&cid=461446446&page=1" scrolling="no" border="0" frameborder="no" framespacing="0" allowullscreen="true"> </iframe>

这是最新版增加的markdown以及无限画布内容

<iframe src="https://player.bilibili.com/player.html?aid=550474675&bvid=BV18i4y197s9&cid=477922265&page=1" scrolling="no" border="0" frameborder="no" framespacing="0" allowullscreen="true"> </iframe>

1. 设计灵感

（一）当前背景

随着 GUI 框架的不断发展，纸质笔记逐渐有了它的弊端，比如说图片插入/书写代码/修改内容等繁杂，于是就涌现出了电子化笔记。目前市面上主流的笔记软件分为两个派系，一类是对应于 IOS 端的平书写类型笔记，这种笔记软件主要是优化纸质书写的弊端，提供了一定的便捷性；另一类是对应于 PC 端的笔记软件，其有两种类型：文字类笔记以及导图类笔记，对于大多数用户来说一般是用文字类笔进行记录内容，通过导图类笔记整理大纲，方便记忆与整理。

（二）灵感来源

在体验过众多的笔记软件之后，例如说 md 文档类型的思源笔记，Notion 笔记等等，以及 XMind 思维导图软件后，我们不难发现他们都有一个共同的弊端。

对于文字类笔记来说，逻辑表达完全取决于文字，举个例子来说：当我们在作操作系统这门课程的笔时，学到死锁这一节，要我们解释什么是死锁，用文字来表示就是如下：

是指两个或两个以上的进程在执行过程中，因争夺资源而造成的一种互相等待的现象，若无外力 作用它们都将无法推进下去。称此时系统处于死锁状态或系统产生了死锁。

我们不难发现，当我们阅读文字的时候，我们会将文字所表达的逻辑在我们的脑海中表示出来。这样有几个弊端：

1. 在涉及大规模知识结构的时候，我们只能通过文字列表或标题的形式去区分每个知识模块，但是知模块和知识模块之间是有联系的，这种表示方式割裂了它们之间存在的逻辑关系。
2. 我们在读取文字的时候，会多出一个“将文字转换成自己所理解的逻辑图像”的步骤。

对于思维导图类软件来说，虽然可以部分解决文字笔记的知识模块之间的联系，但是对于逻辑的表示不是很清晰：节点与节点之间的联系只有一条线，而对于这条线表示什么逻辑却没有办法说明，典型例子就是 XMind 这个软件，例如我们从 A 节点导出一条线连接到 B 节点，我们无法知道 A 和 B 是么关系，仅仅知道它们是有关系的而已。同时在市面上所有的思维导图软件中，都不支持节点内的复编辑：富文本/代码高亮/表格/图片/附件/子节点概念等等，对于复杂逻辑的表示很不友好。

因此为了解决上述弊端，我独自开发了一款基于节点的逻辑思维导图软件，它不仅支持节点的复杂编，同时对于 UI 的用户自定义也做到了极致，用户可以自定义其中任何一个部件的颜色/宽度；而对于关重要的复杂逻辑表达，我们引入了逻辑真/假的端口以及逻辑与或非门的支持。

2. 知识体系介绍

注: 此时的版本为 0.1版本, 在文章下侧会介绍最新版全部内容, 不过记笔记的思想应该是一样的.

针对于市面上思维导图的逻辑表示不完备所设计

对于一个节点与另一个节点的逻辑关系，我们需要有这两个节点的真值表示：同样以死锁为例子。

例如说，A 节点是不剥夺条件（产生死锁的其中一个必要条件），而 B 节点是死锁，那么当 A 节点值为真的时候，B 节点可能为真。而 A 节点真值为假的时候，B 节点一定为假。

因此我们采用了输入为真（左上角），输入为假（左下角），输出为真（右上角），输出为假（右下）四个端口去表示这个逻辑。当我们从 A 节点的输出为假端口引出这条逻辑线的时候表示，输出节点真值为假，即 A 节点真值为假。而 B 节点作为输入节点，也有两个输入，一个为真，一个为假，也表了自身的真值。这样子就表示了上述死锁的其中一种逻辑关系，具体见下图的详解：

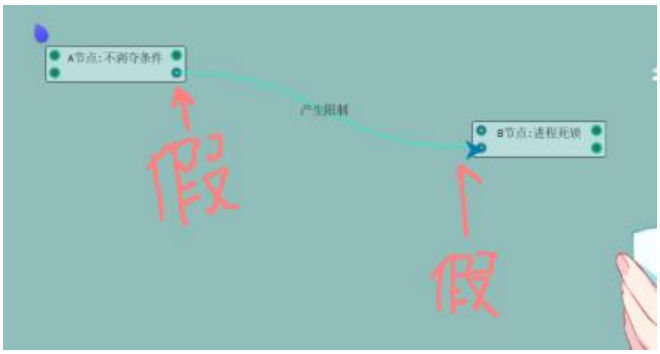


图 2：真值的引入

我们通过上面已经表示了什么时候 B 节点的进程死锁真值为假，但是我们还缺少令它真值为真的情况因此我们考虑什么时候它的真值为真。再看死锁的必要条件，一共有四个：不剥夺条件，请求和保持件，循环等待条件，互斥条件。只有当四个条件全部为真的时候，B 节点才能为真。

好，现在我们如果不引入任何东西，就使用现在的表示逻辑，我们只能够如此表示：

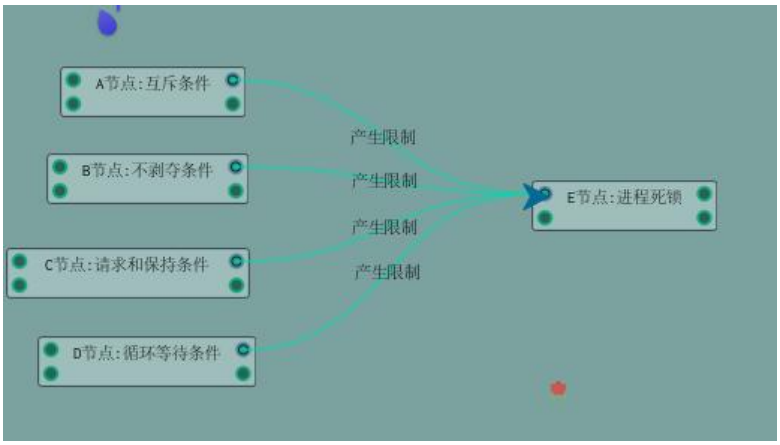


图 3：错误示范

当我们如此表示的时候，我们发现这样子的逻辑是说：A,B,C,D 中任意一个节点真值为真了，都能让 E

节点真值为真，这不是我们想要的逻辑表示。我们想要的表示应当是 A,B,C,D 全部为真的时候，E 才为真，因此我们引入了一个逻辑控制组件：与或非门，如图所示：逻辑控制组件对于输入和输出有三条件：分别是 And, Or, Not，当我们将 A,B,C,D 四个节点的输出放到输入与门上表示，只有四个节的真　值全部为真的时候，它们才能算作真值为真。而逻辑控制组件的输出为与门表示如果有多个节，那么所被输入的多个节点都能作数。

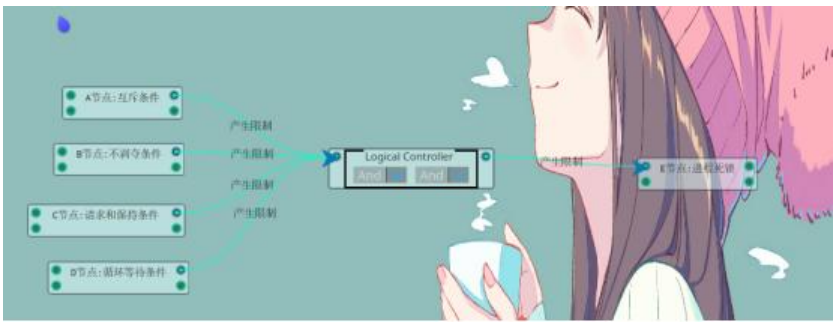


图 4： 逻辑控制组件的引入

我们再举个例子，对于逻辑控制组件的输入为或门（Or）的时候，表示只要其中一　个条件成立，么这条逻辑链才成立。同样用进程死锁的例子来表示：进程死锁产生的原因有很多，比如说对系统资的竞争，进程推进顺序非法，信号量使用不当等。这些原因只要出现一个，那么就会产生进程死锁。们用或门（Or）连接这几个条件，表示只要有一个为真，那么它们的组合才为真，逻辑链才能走通。图所示：

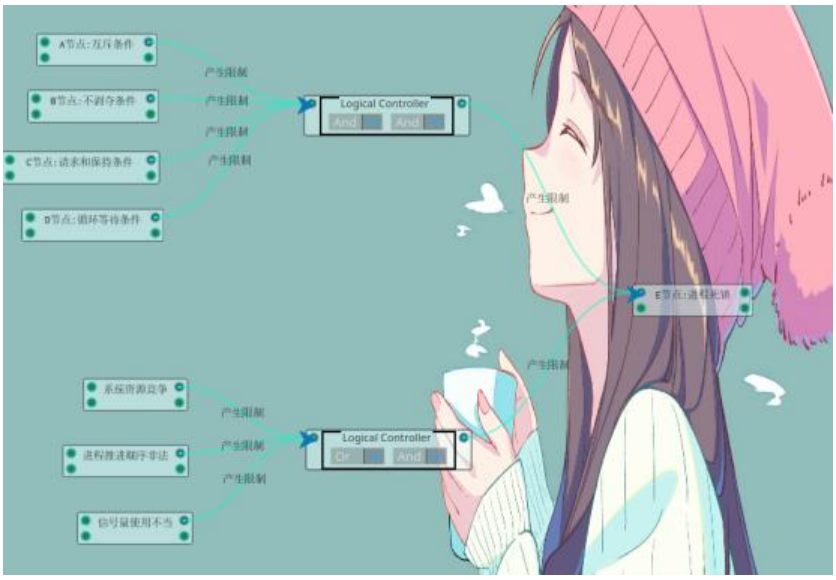


图 5： 逻辑控制控件的使用

好，目前我们已经处理了基本的节点与节点之间的相互限制的逻辑。但是我们考虑这种情况：如果我需要表示知识模块和知识模块之间的关系应该如何表示呢。因此，我们引入一个从属于的子节点概念即我们可以在节点内嵌套其他子节点，形成一个知识的小模块。如图所示：我们的子节点是网格布局允许子节点之间的位置调整。

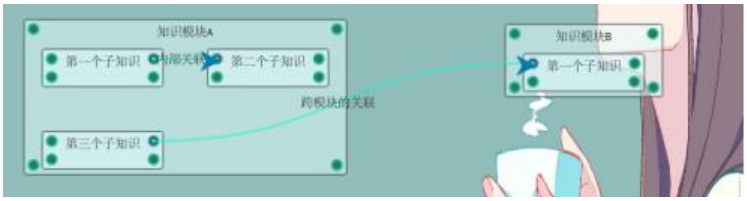
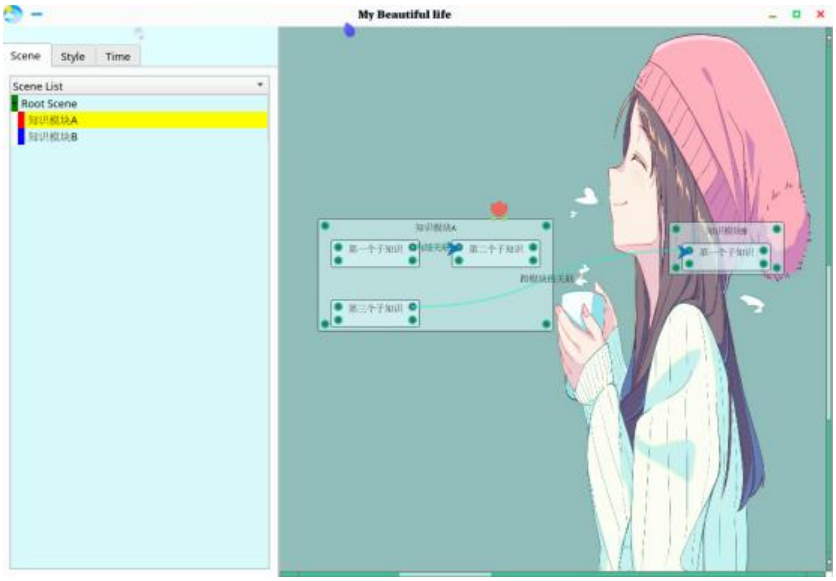


图 6：子节点的引入

好，接下来我们再考虑一种情况，如果我们的知识模块过大应当如何处理呢。如果还采用这种子节点限制的嵌套，一个知识模块有几百层这么长，虽然我们支持搜索功能，但是这样对于我们的记忆是不便的，我们的大脑喜欢一小块一小块的记忆，因此我们引入了一个子图的概念，对于每个节点，我们可以进入其子图，子图内又是一个新的场景，我们可以在子图里表示我们想要的复杂逻辑，从而优化节点过长带来的烦恼。如图所示，我们可以进入知识模块 A 的子图，进行复杂编辑，其内部是一个崭的场景，可以添加这些部件。而对于侧边的子图目录我们有升序排列和降序排列。



3. NodeNote功能全介绍

1. 如何运行

输入框架IME最好选择搜狗输入法

1. 不同平台的方式

1. 使用脚本方式运行

1. 安装　python: Python版本 >= 3.6
2. 安装依赖:　pip install -r requirements.txt
3. 运行脚本:　python example.py

2. 使用可执行文件运行(路径应为纯英文路径, 不要包含特殊字符)

- Windows: 运行　NodeNote.exe
- Mac: 运行　NodeNote.app
- Linux:
    1. 安装依赖:　sudo apt install libxcb-xinerama0 << (Unbuntu)
    2. 运行　NodeNote二进制文件

2. 打开后的工作区介绍

1. 进入工作区: 现版本采用工作区结构,

- 双击上次打开的目录可以直接打开目录
- 也可以选择打开工作区目录按钮, 选择你的文件夹

2. 工作区结构: 第一次打开空白工作区会生成以下文件

- .NOTENOTE: 记录你创建工作区的时间以及保存你上次打开过的文件
- Resources: 程序运行所需的资源文件
- Notes: 创建笔记所在的目录, 您的.note格式笔记最好都创建在该文件夹, 因为如果没有上次打开的件, 则在该目录检索, 如果没有检索到, 则在该目录新建一个.note格式文件
- History: 运行时的文件每隔3分钟会自动备份一份到该文件夹, 如果资源过大可以定时清理! 一个.note型的话大概只有几KB.
- Documents: 您的markdown文件备份
- Attachments: 当你使用节点的附件功能时, 会自动拉取该文件到这个文件夹
- Assets: 您的笔记所用到的图片都保存在这个文件夹

过去版本迁移

- 将根目录的 Assests移动到工作区目录
- 将您的 .note文件移动到Notes即可

2. 如何使用小部件

1. Alt+Q或者鼠标右键 创建属性控件: 支持富文本, markdown, 以及其他小部件的嵌套

支持的嵌套类型

- 嵌套自身
- 嵌套子图: 相当于一个存在于当前场景的自由的子场景
- 嵌套 todo控件:
  - 用于你的任务控制, 点开始按钮则开始计时, 左侧是总时间, 右侧是你开始计时之后的时间.
  - 结束计时后, 会将你的右侧时间加到左侧总时间上.
- 嵌套 附件控件:
  - cover是你的附件图片,可以自定义;
  - File是你要添加的附件.添加完成后, 会将你的附件拉到当前目录的Attachments中.
  - 添加完成后, 双击图片则会调用系统默认的打开程序打开你的附件.

支持的富文本操作:

您可以选中节点内的富文本拖拽到其他地方

Python高亮 :Ctrl+9 **清空对齐格式:Ctrl+P**
粗:Ctrl+W

左对齐 :Ctrl+[右对齐 :Ctrl+] 居
对齐 :Ctrl+ \

斜体 :Ctrl+Q 下划线 :Ctrl+R 删
线:Ctrl+/

增大字体:Ctrl+G 缩小字体:Ctrl+H
变字体颜色:Ctrl+N

超链接:Ctrl+M 数学公式格式参见: Ctrl+I
空所有格式:Ctrl + L

撤销上一步:Crtl+Z 恢复上一步:Ctrl+Y
建一个表格:Ctrl+1

增加一行表格:Ctrl+3 增加一列表格 :Ctrl+2
除一行表格 :Ctrl+5

删除一列表格 :Ctrl+4 选中表格后合并表格行列:Ctrl+6
并表格后拆分已合并内容:Ctrl+7

添加一个列表 :Ctrl+8 改变插入图片的大小 :Ctrl+U
字向后缩进或向前缩进:Tab or Ctrl+Tab

复制html内容:Ctrl+C 复制纯文本内容:Ctrl+Shift+C
贴:Ctrl+V

支持的Markdown操作:

- 选中小部件后在侧边栏进行编辑, 当退出侧边栏时, 会自动保存 markdown到数据库以及备份文件中
- 支持的Markdown类型有: 所有基本 markdown以及UML, 代码块高亮, LaTeX, 绘图 具体可以移该项目 >> tui.editor
- 插入图片请使用相对路径: ../Assets/您的图片

支持的节点无限画布:

- 其样式对应于Draw Widget的样式
- 您可以通过 W/A/S/D扩展画布的大小

支持的真值

- 可以注意到每个节点都有四个端口, 上面两个是真值为真的输入输出端口, 下面两个是真值为假的输输出端口
- 通过按照相应真值的端口进行连线, 结合下面的逻辑部件, 可以表示出你想要的逻辑.

支持的复制节点以及粘贴

- 您可以使用 Alt+R复制该节点以及其内部所有内容
- 然后可以通过 Alt+T粘贴其到任意一个场景中

支持的扩大与缩小: 用Shift+鼠标左键 扩大与缩小

2. Alt+W或鼠标右键创建逻辑控件: 使用两个与或非门进行

辑的控制

- 上面的是输入的 与或非门, 下面的是输出的与或非门
- 当你有多个输入的时候
  - 调整输入成 或门, 则表示所有输入, 仅需一个成立, 则输出成立
  - 调整输入成 与门, 则表示所有输入, 全部都得成立, 则输出成立
  - 调整输入成 非门, 则表示将输入结果逆反, 例如你从逻辑部件的真值为假的端口连到逻辑控件, 逆反后为真
- 当你有多个输出的时候
  - 调整输出成 或门, 则表示所有输出, 仅有部分成立
  - 调整输出成 与门, 则表示所有输出, 全部都成立
  - 调整输出成 非门, 则表示将输出结果逆反

3. Alt+E或鼠标右键创建绘画部件

- 请先连接数位板
- 支持橡皮擦功能, 现在数位板驱动中, 将你的笔其中一个按钮设置为橡皮擦.

4. 通过双击其他部件端口创建连线

- 连线选中后可以通过两个 控制点控制连线的位置
- 可以在 属性控件上按Ctrl+0生成与之相关所有连线的ui动画, 观察逻辑流向
- 也可以在 连线上单独使用Ctrl+0生成选中连线的逻辑动画

5. 结合上述所有部件的碰撞检测

- 您可以拖动 属性部件碰撞其他部件
  - 直接拖动到其他 属性部件: 将自身添加进其他属性部件的当前行
  - 拖动到其他 属性部件时, 按住Ctrl: 将自身添加进其他属性部件的下一行
  - 拖动到 连线: 自动插入该部件到连线的中间
- 您可以在 属性部件中进行碰撞检测
  - 删除 属性控件的子控件会生成一个雪花图案, 通过碰撞雪花图案, 可以替代其位置
- 您可以使用右键的上下文菜单调整自身在其他 属性控件内的位置

6. 综合上述的逻辑控件和属性控件的功能

- 您可以按 Ctrl+up/down/left/right实现辅助对齐功能
- 您可以按 Ctrl+1/2/3/4/5/6/7/8 移动当前鼠标位置对应左右上下的部件移动50px

3. 如何使用场景

1. 创建子场景

- 每个 子场景都与一个属性控件绑定, 您可以通过Alt+鼠标左键点击小部件, 创建其子场景.

- 创建子场景后, 会在侧边栏的目录中生成你的索引, 您可以通过索引找到相应的子场景
  - 可以通过 Alt+Z返回上一次的场景
  - 可以通过 Alt+X返回父场景

2. 属性控件与其他属性控件跨越场景的超链接跳转

- 通过 Alt+C复制该节点的id
- 在其他节点中输入该 id, 通过Ctrl+M将其变为超链接
- 变为超链接后, 您可以自由修改其 id字符串, 存储的跳转信息不会因为其字符串改变而改变了

3. 基本设置

- 扩大场景(左右上下): alt + 1/2/3/4
- 缩小场景(左右上下): alt + 5/6/7/8
- 打开辅助线功能(默认开启): F1
- 打开Undo&&Redo功能(默认开启): F2
- 打开飘落的图片特效(默认开启): F11
- 打开搜索栏: Ctrl+F
- 缩放视图: Ctrl +-
- 打开缩略图: Shift+B

4. 文件分享与导出操作

1. 导出该场景到.note文件

- 不导出该场景下的子场景, 仅导出该场景的内容: Shift+S
- 导出该场景下的所有递归子场景: Alt+S

2. 导出该场景到png图片

- 导出该场景所有内容: Ctrl+Alt+P
- 导出该场景选中部件的内容: Ctrl+Shift+P

5. UI的操作

- 您可以通过 Ctrl+B打开侧边栏
    - 侧边栏拥有文件夹视图: 您可以通过右键上下文菜单 创建或删除文件, 以及通过鼠标左键切换.note文件
    - 场景+Markdown视图: 这是您该 .note笔记的结构
    - 样式自定义视图: 您可以通过该视图自定义所有样式
        - 目前的窗口样式可以通过 qss文件进行修改, 我只做了一个, 感兴趣的朋友可以参照Resource/Stylesheets目录下的qss仿照一个, 然后可以通过样式按钮加载你写的qss文件.
        - 其他部件样式支持 所有场景, 当前场景, 选中部件样式的修改
- 您可以通过 Alt+G调整侧边栏到左边或上边
- 您可以通过拉动侧边栏以及内部的条进行调整大小

6. 最后

感谢您使用NodeNote, 如果遇到任何问题或想提建, 欢迎创建一个issue

如果有朋友想参与项目, 非常欢迎. 目前要是有人能多写几个qss样式表就好了哈哈哈哈.

1. fork该仓库
2. 修改代码
3. 创建新 brand
4. push到我的main分支
5. 我会根据实际情况进行合并分支

v2.36.21:

- 侧边栏画板
- 对齐功能
- 动画演示变成树的遍历
- 修复了序列化的一个错误
- 修复线条的编辑框下上文菜单的点击
- 所有场景的背景色更新sub view内的
- effect cutline 计算boundingRect()
- 修复搜索的错误
- 修复还没建立线条就删除的错误
- 修复了错误的majax解析闪退