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

<h2 id="这是项目链接--喜欢的小伙伴可以帮忙点点star吗--最近校招压力有点大哈哈哈--感谢了-">是项目链接, 喜欢的小伙伴可以帮忙点点 star 吗, 最近校招压力有点大哈哈哈, 感谢了.</h2>
NodeNote
<blockquote>
这是 v2.19 介绍视频，此时暂无 markdown 内容, 主要介绍笔记思想
</blockquote>
<iframe src="https://player.bilibili.com/player.html?aid=849841725&bvid=BV1PL4y1p7b&cid=461446446&page=1" border="0" sandbox="allow-scripts allow-same-origi" width="99%" height="500px"> </iframe>
<blockquote>
这是最新版增加的 markdown 以及无限画布内容
</blockquote>
<iframe src="https://player.bilibili.com/player.html?aid=550474675&bvid=BV18i4y197s&cid=477922265&page=1" border="0" sandbox="allow-scripts allow-same-origin" width="99%" height="500px"> </iframe>
<h2 id="1--设计灵感">1. 设计灵感</h2>
<h2 id="-一-当前背景">（一）当前背景</h2>
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/修改内容等杂性，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 端平板书写类型笔记，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笔记进行记录内容，通过导图类笔记整理大纲，方便记忆与整理。
<h2 id="-二-灵感来源">（二）灵感来源</h2>
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 XMin 等思维导图软件后，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笔记时，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。样会有几个弊端：

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，但是识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。
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，但是对于逻辑的示并不是很清晰：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，型的例子就是 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 是什么关系，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，都不支持节点内复杂编辑：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编辑，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；而于至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。
<h2 id="2--知识体系介绍">2. 知识体系介绍</h2>
<blockquote>
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.
</blockquote>
针对于市面上思维导图的逻辑表示不完备所设计
<blockquote>
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
</blockquote>

</blockquote>
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 点真值为真的时候，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（下角）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，输出点的真值为假，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，一个为假也代表了自身的真值。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：

/p>
图 2：真值的引入
<blockquote>
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，请求和持条件，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。
</blockquote>
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：

/p>
图 3：错误示范
<blockquote>
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 才能为真，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三种条件：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，只有四节点的真　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节点，那么所被输入的多个节点都能作数。
</blockquote>

图 4：　逻辑控制组件的引入
<blockquote>
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，那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，比如说对系资源的竞争，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。我们用或门（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。如图所示：
</blockquote>

图 5：　逻辑控制控件的使用
<blockquote>
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：如我们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局，允许子节点之间的位置调整。
</blockquote>

/p>
图 6：子节点的引入

<blockquote>
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。如果还采用这种子点无限制的嵌套，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不方便的，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，们都可以进入其子图，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，从而化子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，其内部是一崭新的场景，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。
</blockquote>

/p>
<h2 id="3--NodeNote功能全介绍">3. NodeNote 功能全介绍</h2>
<h2 id="1--如何运行">1. 如何运行</h2>
<blockquote>
输入框架 IME 最好选择 <code>搜狗输入法</code>
</blockquote>
<h2 id="1--不同平台的方式">1. 不同平台的方式</h2>
<h3 id="1--使用脚本方式运行">1. 使用脚本方式运行</h3>

安装 <code>python</code>: <code>Python版本</code> >= 3.6
安装依赖: <code>pip install -r requirements.txt</code>
运行脚本: <code>python example.py</code>

<h3 id="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-">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)</h3>

Windows: 运行 <code>NodeNote.exe</code>
Mac: 运行 <code>NodeNote.app</code>
Linux:

安装依赖: <code>sudo apt install libxcb-xinerama0</code> << <code>(Unbuntu)</ode>
运行 <code>NodeNote</code> 二进制文件

<h2 id="2--打开后的工作区介绍">2. 打开后的工作区介绍</h2>
<blockquote>

进入工作区: 现版本采用工作区结构,

ir_interface.png" alt="">

</blockquote>

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

<blockquote>
<ol start="2">
工作区结构: 第一次打开空白工作区会生成以下文件

231111754.png" alt="">

</blockquote>

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

<blockquote>
过去版本迁移
</blockquote>

将根目录的 <code>Assests</code> 移动到工作区目录
将您的 <code>.note</code> 文件移动到 <code>Notes</code> 即可

<h2 id="2--如何使用小部件">2. 如何使用小部件</h2>
<h2 id="1--或者-创建--支持富文本--markdown--以及其他小部件的嵌套">1. <code>Alt+Q</code> 或者 <code>鼠标右键</code> 创建 <code>属性控件</code>: 支持富文本, markdown, 以及他小部件的嵌套</h2>
<blockquote>
支持的嵌套类型

</blockquote>

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 <code>todo控件</code>:

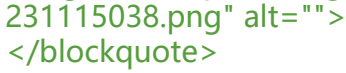
用于你的任务控制, 点开始按钮则开始计时, 左侧是总时间, 右侧是你开始计时之后的时间.
结束计时后, 会将你的右侧时间加到左侧总时间上.

嵌套 <code>附件控件</code>:

<code>cover</code> 是你的附件图片,可以自定义;
<code>File</code> 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 <code>Attchments</code> 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.

<blockquote>
支持的富文本操作:
</blockquote>
您可以选中节点内的富文本拖拽到其他地方
<table>
<thead>

<tr>
<th>Python 高亮 :<code>Ctrl+9</code></th>
<th>清空对齐格式:<code>Ctrl+P</code></th>
<th>加粗:<code>Ctrl+W</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>左对齐 :<code>Ctrl+[</code></td>
<td>右对齐 :<code>Ctrl+]</code></td>
<td>居中对齐 :<code>Ctrl+ \ </code></td>
</tr>
<tr>
<td>斜体 :<code>Ctrl+Q</code></td>
<td>下划线 :<code>Ctrl+R</code></td>
<td>删除线:<code>Ctrl+/</code></td>
</tr>
<tr>
<td>增大字体:<code>Ctrl+G</code></td>
<td>缩小字体:<code>Ctrl+H</code></td>
<td>改变字体颜色:<code>Ctrl+N</code></td>
</tr>
<tr>
<td>超链接:<code>Ctrl+M</code></td>
<td>数学公式格式参见: <code>Ctrl+I</code></td>
<td>清空所有格式:<code>Ctrl + L</code></td>
</tr>
<tr>
<td>撤销上一步:<code>Crtl+Z</code></td>
<td>恢复上一步:<code>Ctrl+Y</code></td>
<td>创建一个表格:<code>Ctrl+1</code></td>
</tr>
<tr>
<td>增加一行表格:<code>Ctrl+3</code></td>
<td>增加一列表格 :<code>Ctrl+2</code></td>
<td>删除一行表格 :<code>Ctrl+5</code></td>
</tr>
<tr>
<td>删除一列表格 :<code>Ctrl+4</code></td>
<td>选中表格后合并表格行列:<code>Ctrl+6</code></td>
<td>合并表格后拆分已合并内容:<code>Ctrl+7</code></td>
</tr>
<tr>
<td>添加一个列表 :<code>Ctrl+8</code></td>
<td>改变插入图片的大小 :<code>Ctrl+U</code></td>
<td>文字向后缩进或向前缩进:<code>Tab</code> or <code>Ctrl+Tab</code></td>
</tr>
<tr>
<td>复制 html 内容:<code>Ctrl+C</code></td>
<td>复制纯文本内容:<code>Ctrl+Shift+C</code></td>
<td>粘贴:<code>Ctrl+V</code></td>
</tr>

</tbody>
</table>
<blockquote>
支持的 Markdown 操作:


</blockquote>

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 <code>markdown</code> 到据库以及备份文件中
支持的 Markdown 类型有: 所有基本 <code>markdown</code> 以及 <code>UML</code>, <code>代码块高亮</code>, <code>LaTeX</code>, <code>绘图</code> 具体可以移步该项目 >> tui.editor
插入图片请使用相对路径: <code>../Assets/您的图片</code>

<blockquote>
支持的节点无限画布:


</blockquote>

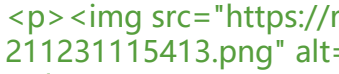
其样式对应于 Draw Widget 的样式
您可以通过 <code>W/A/S/D</code> 扩展画布的大小

<blockquote>
支持的真值
</blockquote>

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入输出端口
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.

<blockquote>
支持的复制节点以及粘贴
</blockquote>

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

<blockquote>
支持的扩大与缩小: 用 <code>Shift+鼠标左键</code> 扩大与缩小
</blockquote>
<h2 id="2--或创建--使用两个与或非门进行逻辑的控制">2. <code>Alt+W</code> 或 <code>标右键</code> 创建 <code>逻辑控件</code>: 使用两个与或非门进行逻辑的控制</h2>


上面的是输入的 <code>与或非门</code>, 下面的是输出的 <code>与或非门</code>
当你有多个输入的时候

调整输入成 <code>或门</code>, 则表示所有输入, 仅需一个成立, 则输出成立
调整输入成 <code>与门</code>, 则表示所有输入, 全部都得成立, 则输出成立
调整输入成 <code>非门</code>, 则表示将输入结果逆反, 例如你从 <code>逻辑部件</code>

的真值为假的端口连到 <code>逻辑控件</code>, 则逆反后为真

当你有多个输出的时候

调整输出成 <code>或门</code>, 则表示所有输出, 仅有部分成立
调整输出成 <code>与门</code>, 则表示所有输出, 全部都成立
调整输出成 <code>非门</code>, 则表示将输出结果逆反

<h2 id="3--或创建">3. <code>Alt+E</code> 或 <code>鼠标右键</code> 创建 <code>绘画件</code></h2>


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.

<h2 id="4--通过创建连线">4. 通过 <code>双击其他部件端口</code> 创建连线</h2>

连线选中后可以通过两个 <code>控制点</code> 控制连线的位置


可以在 <code>属性控件</code> 上按 <code>Ctrl+0</code> 生成与之相关所有连线的 <code>ui动画</code>, 观察逻辑流向


也可以在 <code>连线</code> 上单独使用 <code>Ctrl+0</code> 生成选中 <code>连线</code> 的逻辑动画

<h2 id="5--结合上述所有部件的碰撞检测">5. 结合上述所有部件的碰撞检测</h2>


您可以拖动 <code>属性部件</code> 碰撞其他部件

直接拖动到其他 <code>属性部件</code>: 将自身添加进其他 <code>属性部件</code> 的前行
拖动到其他 <code>属性部件</code> 时, 按住 <code>Ctrl</code>: 将自身添加进其他 <code>属性部件</code> 的下一行
拖动到 <code>连线</code>: 自动插入该部件到连线的中间

您可以在 <code>属性部件</code> 中进行碰撞检测

删除 <code>属性控件</code> 的子控件会生成一个雪花图案, 通过碰撞雪花图案, 可以替代其置

您可以使用右键的上下文菜单调整自身在其他 <code>属性控件</code> 内的位置

<h2 id="6--综合上述的逻辑控件和属性控件的功能">6. 综合上述的逻辑控件和属性控件的功能</h2

您可以按 <code>Ctrl+up/down/left/right</code> 实现辅助对齐功能

您可以按 <code>Ctrl+1/2/3/4/5/6/7/8</code> 移动当前鼠标位置对应左右上下的部件移动 5 px

<h2 id="3--如何使用场景">3. 如何使用场景</h2>
<h2 id="1--创建子场景">1. 创建子场景</h2>

每个 <code>子场景</code> 都与一个 <code>属性控件</code> 绑定, 您可以通过 <code>Al+鼠标左键</code> 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 <code>Alt+Z</code> 返回上一次的场景
可以通过 <code>Alt+X</code> 返回父场景

<h2 id="2--与其他跨越场景的超链接跳转">2. <code>属性控件</code> 与其他 <code>属性控件/code> 跨越场景的超链接跳转</h2>

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

<h2 id="3--基本设置">3. 基本设置</h2>

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

<h2 id="4--文件分享与导出操作">4. 文件分享与导出操作</h2>
<h2 id="1--导出该场景到文件">1. 导出该场景到 <code>.note</code> 文件</h2>

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

<h2 id="2--导出该场景到图片">2. 导出该场景到 <code>png</code> 图片</h2>

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

<h2 id="5--UI的操作">5. UI 的操作</h2>

您可以通过 <code>Ctrl+B</code> 打开 <code>侧边栏</code>

侧边栏拥有文件夹视图: 您可以通过右键上下文菜单 <code>创建</code> 或 <code>删除</code> 文件, 以及通过鼠标左键切换 <code>.note</code> 文件
场景 +Markdown 视图: 这是您该 <code>.note</code> 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 <code>qss</code> 文件进行修改, 我只做了一个, 感兴趣的朋友可以照 <code>Resources/Stylesheets</code> 目录下的 <code>qss</code> 仿照一个, 然后可以通样式按钮加载你写的 <code>qss</code> 文件.
其他部件样式支持 <code>所有场景</code>, <code>当前场景</code>, <code>选中部件</ode> 样式的修改

您可以通过 <code>Alt+G</code> 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

<h2 id="6--最后">6. 最后</h2>
<blockquote>
感谢您使用 <code>NodeNote</code>, 如果遇到任何问题或想提建, 欢迎创建一个 <code>issue</code>
</blockquote>
<blockquote>
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 <code>qss样式表</code> 就好哈哈哈哈.
</blockquote>

<code>fork</code> 该仓库
修改代码
创建新 <code>brand</code>
<code>push</code> 到我的 <code>main</code> 分支
我会根据实际情况进行合并分支

<h2 id="v2-36-21-">v2.36.21:</h2>

<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> 侧边画板
<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> 对齐能

<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> 动画
示变成树的遍历
<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> 修复
序列化的一个错误
<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> 修复
条的编辑框下上文菜单的点击
<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> 所有
景的背景色更新 sub view 内的
<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> effect
utline 计算 boundingRect()
<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> 修复
索的错误
<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> 修复
没建立线条就删除的错误
<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> 修复
错误的 majax 解析闪退
