浙 江 大 學

本 科 生 毕 业 设 计 报 告

项目名称 可视化编程环境研究

姓　　名 _____

学　　号 _____

指导教师 _____

专　　业 计算机科学与技术

学　　院 计算机学院

A Dissertation Submitted to Zhejiang University for the Degree of Bachelor of Engineering

TITLE: The search of visual programming environment

Author: _____

StudentID _____

Mentor _____

Major: Computer Science

College: Zhejiang University

Submitted Date: _____

浙江大学本科生毕业论文（设计）诚信承诺书

1.本人郑重地承诺所呈交的毕业论文（设计），是在指导教师的指导下严格按照学校和学院有关规定完成的。

2.本人在毕业论文（设计）中引用他人的观点和参考资料均加以注释和说明。

3. 本人承诺在毕业论文（设计）选题和研究内容过程中没有抄袭他人研究成果和伪造相关数据等行为。

4. 在毕业论文（设计）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

毕业论文（设计）作者签名：

_____年_____月_____日

摘要

随着计算机行业的快速发展，目前专业的程序员的数量满足不了需求。但是由于学习常规性编程语言的成本很高，所以很多非专业的计算机人才并不能够很好的掌握编程的技能。所以我们希望能够降低编程的门槛，使得更多的非计算机专业的人能够以很小的成本学会编程。其中的一个解决问题的办法就是将编程可视化。将文本性的编程语言转为可视化的编程语言。通过对前人的研究进行分析，对目前一些非计算机专业的人的调查发现，创建一个可视化的编程工具是可行的。

本论文说明了，开发一个可视化编程工具前期的一些工作。使用 JavaScript 语言进行开发，并以此为基础采用 Nodewebkit 工具，解决了应用跨平台性的问题，提高了开发的效率。另外，我们采用 snap 的第三方库，绘制除了应用基本的界面元素。定义了按钮，图形块，调色盘等基础类和一些基础的方法。在此基础之上可以添加很多新的功能，使得应用有着很强的可扩展性。最后，是完成了图形块的拖动和拼接，用户可以通过这些拼接完成最基本的可视化语句编写。

关键词　可视化编程工具 Node-webkit 用户交互界面 图形块

Abstract

With the rapid development of the computer industry， the number of programmers can not meet our need.However, due to the difficulty of learning conventional programming language,many people who do not major in Computer Science can not master programming skills well.So we hope to reduce the difficulty of learning programming, more people can master programming skills easily.One of the solution is visual programming.According to the previous research and the survey to some people,visual programming is a good way to solve the problem.

This paper describes some work of the development of a visual programming tool .We use JavaScript to develop our project.And we use the tool of Node-webkit to solve the problem of Cross-platform,so that we can improve our efficient.And we use the library of snap to draw our GUI.We defined the Button class,the block class etc.Based on that,we can add some new function.At last, the user can draw and mosaic graphics block to finish programming.

Keywords visual programming tool ,Node-webkit ,GUI,graphics block

目录

摘要 I

第 1 章 项目背景 4

1.1 问题的提出 4
1.2 最新的研究状况 5
1.3 前期研究工作 13

第 2 章 项目实施方案 15

2.1 开发工具 15
2.1.1 JavaScript 15
2.1.2 Node.js 16
2.1.3 node-webkit 17
2.1.4 第三方库 17
2.2 界面设计 21
2.3 项目模块 23
2.4 本章小结 25

第 3 章 在项目中负责的具体工作 27

3.1 项目的构建 27
3.1.1 项目的配置 27
3.1.2 应用启动界面 28
3.2 用户交互界面 29
3.3 语句块的设计定义 34
3.4 控件的定义 37
3.5 程序的执行 40
3.6 本章小结 42

第 4 章 项目成果 43

参考文献 44

致谢 45

第 1 章　项目背景

1.1 问题的提出

随着计算机技术的飞速发展，对于专业的编程人才的需求也越来大。但是经过专业培训的程序员的数量很少，完全满足不了需求。另外，其他领域也与计算机编程的交叉越来越多，比如美术设计专业的人员，当他们在设计他们的产品的时候进行编程是越来越不可避免的事情。出于工作的需要，类似于这类人员都需要掌握一定的编程技能。我在美院当过一段时间的助教，发现美院的学生在的很多工作都是需要通过编程来完成的，他们有很多的点子，希望通过计算机实现很多效果，这些工作都是需要通过编程来实现。但是由于缺少编程技能，他们有很多的想法，不能够将其实现。他们不是很清楚循环，也不熟悉一些简单的编程逻辑，这使得他们很难实现自己的想法。所以，我们希望使得更多的非专业人士能够掌握编程技能。然而传统的文本式编程的进入门槛很高。诸如 C++，JAVA 等编程语言对一些非编程人员来说过于的复杂。编译错误问题，语句与函数的记忆对编程初学者很不友好。比如学习 c 语言的时候，我们先要掌握基本的语法，同时需要记住一些常用的函数名和库。另外在使用 c 语言进行简单的程序开发的时候，我们对于其中的一些运行的错误不能够很好的进行解决，使得大部分的人掌握编程的时间很长。所以很常见的一个情况就是一位初学者事先要经过很长时间的编程技能学习和训练，才可能开始独立的进行项目的开发。这使得专业编程人才的数量的增加很慢，于是编程人才的需要远远达不到需求。

为了能够应对这一问题，学者提出应该让更多地非计算机专业的人也能够参与到编程中来所以要尽可能的降低学习编程的门槛。于是学者们都将目光投向了可视化编程语言。学者认为可视化编程语言的语法相对比较简单，没有很多复杂的错误，比 C 语言，JAVA 等文本类的语言更易于接受。因为，对于复杂的文本来说，图形更加的容易被识别记忆。同时通过良好的图形设计，不仅可以做到语法上的提示，也可以提示编程者一些基本的逻辑语法结构，所以使用可视化编程语言进行编程的开发，并不用对语法非常的熟悉，图形化的编程是可以避免用户犯语法错误的。另外可视化的编程图形开发，可以做到即时的反映，即用户可以迅速的得知自己的编程运行结果，略去了其中很多运行问题。所以，编程的初学者在使用可视化编程工具进行编程学习的时候，可以不用再需要去考虑学习文本性语言编程中所面临的问题。这使得人可以更快的学习掌握编程的技能。

1.2 最新的研究状况

各方面的学者对这些问题也提出了相应的办法。比如波多黎各大学的 Jose A. Borges 和伊利诺伊大学的 Ralph E. Johnson 认为可视化编程通过提供直觉的易于理解的表示方式为非专业的编程人员打开了编程的大门。【1】墨西哥国家理工学院的 Sergio Sandoval-Reyes,Pedro Galicia-Galicia 和 Ivan Gutierrez-Sanchez 也提出了类似的看法：“那些非专业的编程人员总是希望能够使用那些简单易学的语言来表达自己的想法。这种语言具有简单的语法，并且没有复杂的编译错误或是运行错误。而可视化编程能满足这一要求。”【2】

所以我们急需一个简单易学的编程工具，既可以用来进行初步的编程学习。也方便其他专业的人员能够简单的编写一些程序。

但是仅仅是开发出一套简单的可视化编程工具是远远不够的。我们的最终的目的是使得，一些非计算机专业的人能够使用可视化编程工具来完成与计算机专业的人相同的工作。这意味着，可视化的编程工具需要具有与文本化的编程工具同样的解决一般性问题的能力。但这是目前很多的可视化编程工具所缺乏的。

美国日立公司中央研究实验室的研究员 Keiji Kojima, Yoshiki Matsuda, Seiji Futatsugi 指出尽管可视化编程简单易学，但是相对于文本语言编程来说，其缺乏精确性和普遍性。【3】

类似的观点在波多黎各大学的 Jose A. Borges 和伊利诺大学香槟分校的 Ralph E. Johnson 合作的论文中也有提到：可视化编程在通用目的的编程中并不成功。【1】

此外，工程师 Robert Schaefer 在他的论文 On the Limits of Visual Programming Languages 中详细的分享了他使用可视化编程编写代码的体验，同时指出了可视化编程一些很现实的缺点。【5】比如，可视化编程与文本语言的编程在设计中的思维方式没有什么本质的区别，在使用文本语言编程时遇到的设计问题在使用可视化编程依然会遇到。他认为设计实质是将人得想法翻译成逻辑，而编程则是将逻辑转化成一种规定的语法。在这一过程是无关你使用的时文本类型的语言还是可视化的语言的。他还提到，他使用的一款可视化编程工具中将每个函数都处理成一个块，通过将函数对应的块拖曳到编程区中来实现对函数的调用。但是线上的说明文档并不详细，导致他除了实际写一段小程序测试这个方法之外没有其他办法来得知这个块对应的函数的功能。这给编程带来了巨大的麻烦。另外一个可视化编程的问题就是“查找对象”，在文本类型的语言中，要查找一个特定的词，比如一个标签，就有很多的工具来帮助程序员在代码中找到这

个词。但是在可视化编程中，所有的对象都是可视化的块或者是图片，没有任何工具帮你来寻找你要寻找的对象。还有一些其他问题，比如可视化编程工具在显示终端资源上收到很大的限制，比如说对于嵌套函数模块，文本语言的编程可以很简单地表示每一个层次，但是对可视化的编程语言，就没有这么简单，一般可视化编程只能将代表一个函数的块放到另外代表一个函数的块里面。那么嵌套的层数如果超过 3 层，那么出现的图形将会非常的复杂。

为了解决这个问题。一般有两种思路，Jose A. Borges 与 Ralph E. Johnson 在论文 Multiparadigm Visual Programming Language 提供了第一种思路。【1】就是把各种具有不同特点的可视化编程工具结合起来，使他们拥有相同的用户接口。可视化的编程工具虽然不能解决一些普遍性的编程问题，但是单一的工具在一专门的领域有较好的表现。每一个可视化编程工具都拥有不同的特点，将这些特点结合起来，相互弥补自己的不足，就可以起到与文本类型的语言相同的效果。他们通过研究发现，当前有很多的可视化编程工具都是基于一种范式或是计算模型。一个可视化的编程工具在其对应的范式所强调的那个方面表现得很好，所以其在其他方面就会很难的使用。比如说，基于数据流的可视化编程工具很容易实现使用函数去传递数据，但是对于复杂的数据结构的表示就很困难。另外一种通过实例编程的系统，在复杂的数据结构上很容易表示，但是对于数据流或控制流就很难表示。研究者认为每一种语言基于特定范式的语言都能解决某一方面的编程问题，每一种语言都有相应地工具帮助其建立程序。将这些工具整合起来，使得所有的工具具有统一的用户接口，统一的运算符号，统一的命令，并且工具之间可以相互通信，比如在某个工具之下定义的对象可以在另一个工具中使用。研究者依照这几个目标，尝试整合了五种可视化编程的开发工具，开发了一种基于多范式的可视化编程工具，证明了这个方式是可行的。最后作者提到要成功的实现这个方法，开发一套可以使用的可视化编程开发工具需要考虑到以下几点：1.要分析清楚每一种可视化编程范式的优点。2.要将每一个编程范式表达能力比较强的方面结合起来。3.综合起来的可视化编程工具需要有很强的扩展性，能够不断的将新工具结合进来。4.新的编程工具需要具有易于学习的特点。

另外一种思路，就是借助文本类型的编程，将其中的一些优点结合到可视化的编程语言中。具体的作法有很多。

早期的方法，主要采用的是可视化编程的工具，但是在可视化编程语言表述含糊不清或是有歧义的地方，采用文本表述来弥补图形表达能力不足的缺点。其中作者论文中提到的可视化编程工具在表达数据结构时还是有一些歧义，比如说使用图形来表达一个具有 5 个元素的数组，当程序员选中了正中间的元素的时候，使用文字就会有两种表述方式，是选中了第 3 个元素，或是选中了正中间的

元素。这种歧义在文本类型的语言编程中是不会出现的。作者在这个问题上提出了两种的解决方法。第一种就是系统首先以文本的方式给出一个解释，如果程序员觉得这个解释不正确那么程序员也用文本的方式来修正这一个错误。第二种方法就是预先做一些规则（比如点击索引的图标两次意味着选中索引的元素）来规避歧义。最后作者认为第一种方法比第二种方法更好，第二种方法也能够解决歧义的问题但是由于要规定很多的规则，这样这个可视化编程工具就会难以学习。这样的编程方式，借助了两者的优点，使得程序员可以以可视化编程方式通过自己的直觉和常规的思维来编写程序，同时当可视化编程因为本身表达能力的不足而出现歧义的时候，程序员可以直接用文本语言来完善表达，消除歧义。这种方法结合了两者的优点。【3】

第二种方法，则是构建一些文本式语言的范式和图形的映射关系。从而加强可视化语言的对象的描述能力。在由澳大利亚莫道克大学的 Geoffrey G. Roy 与 Joel Kelso 以及埃迪斯科文大学的 Craig Standing 合作的论文中指出可以通过构建函数式编程范式到图形的映射关系，从而使用函数式编程范式的优点来弥补可视化编程的缺点。【6】他们认为图形的表述本身优于文本，这也是他们论文的前提：1.图形比文字更容易用来传递信息，因为图形能比一个单词带有更多地信息。2.图形能够帮助程序员理解与记忆。3.图形能够鼓励人去学习编程。4.图形编程更适合不同文化不同语言的程序员相互交流。与其他研究人员不同的是，作者更加的关注于如何选取合适的图形来表达信息，而不是说在可视化编程工具常规使用的图形中进行改进。函数式编程意味着编程人员使用最简单基本的函数形式（Value = f (Arg1, Arg2, Arg3.....ArgN)）能够表达出所有的编程对象。这个表达式说明了一个函数对于每组参数都有一个唯一的值。使用函数式编程的程序从最底层的函数到完整地程序都是遵循这个简单地式子。而作者就提出了一种简单地图形化方法来表达这个式子。这个图采用了三层点得结构，第一层每一个点代表一个参数，其中参数如果是一个函数的话则嵌套一个函数块。第二层则代表这个函数的操作。最后一层的点则包含了定义了函数输出结果类型的信息。这种方法的优势在于任何一个程序的每一层都可以采用统一的简单的图形表示。但是使用这样的表达方法还有一个问题需要解决，就是嵌套的层数太多以后，整个图形的表示会形成一颗很复杂的树。要解决这个问题，就需要对低层次的函数进行封装。

上述的方法确实能解决问题，但是实践发现基于上面两种方法的编程工具的使用很复杂。编程的效率并不是很高。

于是很多的学者则是着眼于一些可视化编程的工具，使用这些工具来发挥可视化编程语言易学且直观的特点。日本武藏大学的 Yoshiharu Kato，墨西哥国家理工学院的 Sergio Sandoval-Reyes,Pedro Galicia-Galicia 和 Ivan Gutierrez-Sanchez

还有奥地利格拉茨大学的 Wolfgang Slany 都在这方面做了相关的研究和叙述。【7】总结为对于编程的初学者来讲，可视化编程的工具比文本语言编程更加的直观，更加能够引起人编程的兴趣。在设计上，都是采用基于图标的设计，将运算、对象都抽象成为一个图标，简单易用。说明可视化编程比文本类的编程工具更加的易于接受。 而在设计上来说，大部分的可视化编程工具都具有以下的几个特点：1.将用户和他们的兴趣结合起来，基本上诸如 Scratch，Alice，Greenfoot 等都是使用户能够按照他们的兴趣来编程（如故事，游戏）。2.用户在编程的过程中，可以直接看到他们输入指令的执行结果。3.这些工具都将编程中那些复杂的思考都隐藏起来，总是引导用户先完成编程，在去思考为什么。

目前已经有很多可视化编程工具，如 Scratch，Ardublock，labview 等。

Scratch 就是一种以教育为目的的编程语言。用户通过一些可视化的编程块来控制图片，音乐，声音等多媒体的信息。Scratch 的应用界面如下图 1-1 所示。如上图中可以看出，Scratch 的语句就像一块块的积木，可以相互拼接。用户通过凭借这些语句块来完成编程。另外，也不是所有的语句块都能相互拼接在一起，只有符合语法的形状才能拼接成功。在 Scratch 中，语句图形的形状和颜色起到了提示用户语言语法的作用，比如图 1-2，图 1-3，图 1-4 显示了 Scratch 中的几个语句图形块。我们可以发现，动作的语句块是蓝色的，控制块则是黄色的，运算块是绿色的。不同类型的语句块是不同颜色的。所以用户只要辨认颜色就可以分辨语句的类型。另外从语句块的嵌入槽中，我们发现一个语句块的返回类型决定了语句块整体的形状，数字的返回值是圆形角，布尔值得返回值是三角形的。这样的设计使得，用户在嵌入槽的时候不会犯数据类型的语法错误。

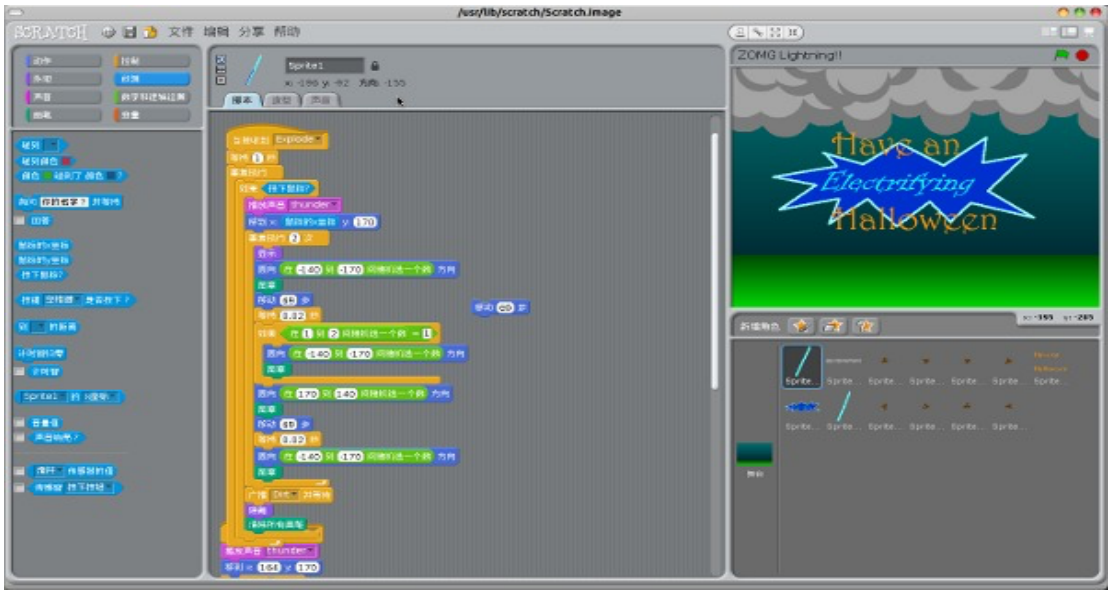

图 1-1 Scratch 应用界面

图 1-2 基本动作块

图 1-3 基本控制块

图 1-4 基本的运算块

另外在 Scratch 中，用户完成语句块的拼接后马上就可以直接运行，并且对用户隐藏了运行的过程细节。由于 Scratch 避免了运行的错误，所以使用 Scratch 编写的程序是不容易崩溃的，这很利于用户对编程的学习。因为 Scratch 可以独立的执行在栈内任何一段代码块，所以 Scratch 在调试上很方便，并且可以使用户直观地了解到程序运行的流程。另外我们从图 1-1 中还可以看出，使用 Scratch 完成代码的拼接之后，用户可以直接在最右侧的舞台上看到实时的代码运行反馈，这给了用户更好的编程体验。

但是 Scratch 依然有着不小的缺陷。一方面我们可以看到 Scratch 不支持代码

的封装。用户是不能够定义自己的函数，也就不能够创建自己的库函数。这使得 Scratch 只能实现一些简单的功能，用户使用 Scratch 只能写一些简单的程序。所以 Scratch 不能够解决很多的一般性。另外 Scratch 的网络功能不够强大，用户不能够分享自己编写的代码，在互联网的时代这是一个很严重的缺陷。这同时也反映出了 Scratch 的另外一个缺陷，扩展性不足。所以 Scratch 现在也仅仅是用于初级的编程教学。

Ardublock 是一款为 Arduino 设计的图形化编程软件。Ardublock 的应用界面如图 1-5 所示。从图中可以看到，不同于 Arduino 的文本式的编程环境，Ardublock 与 Scratch 一样是以图形化积木搭建的方式编程。这种方式也是现在很多可视化编程工具常用的方式。Ardublock 将 Arduino 中很多常用的函数，图形化了。并且将常用的语句，如结构语句，变量，运算符等等都图形化了。其中，对于 Arduino 来说，变量有两大类模拟量和数字量，所以在 Ardublock 中，变量有两种形状。同时很多的运算符模块中的凹槽的形状也不一样，以此来提示用户是输入模拟量还是数字量。此外，Ardublock 还图形化了很多输入输出设备的模块，很多复杂的设备被封装成一个图形块，这使得开发者只要通过拖动对应的图形块，就能驱动外部设备，而不用调用语句对外部进行复杂的初始化了。但是因为 Ardublock 的程序是要烧写到 Arduino 中，才能看到编程的效果，所以并没有舞台。

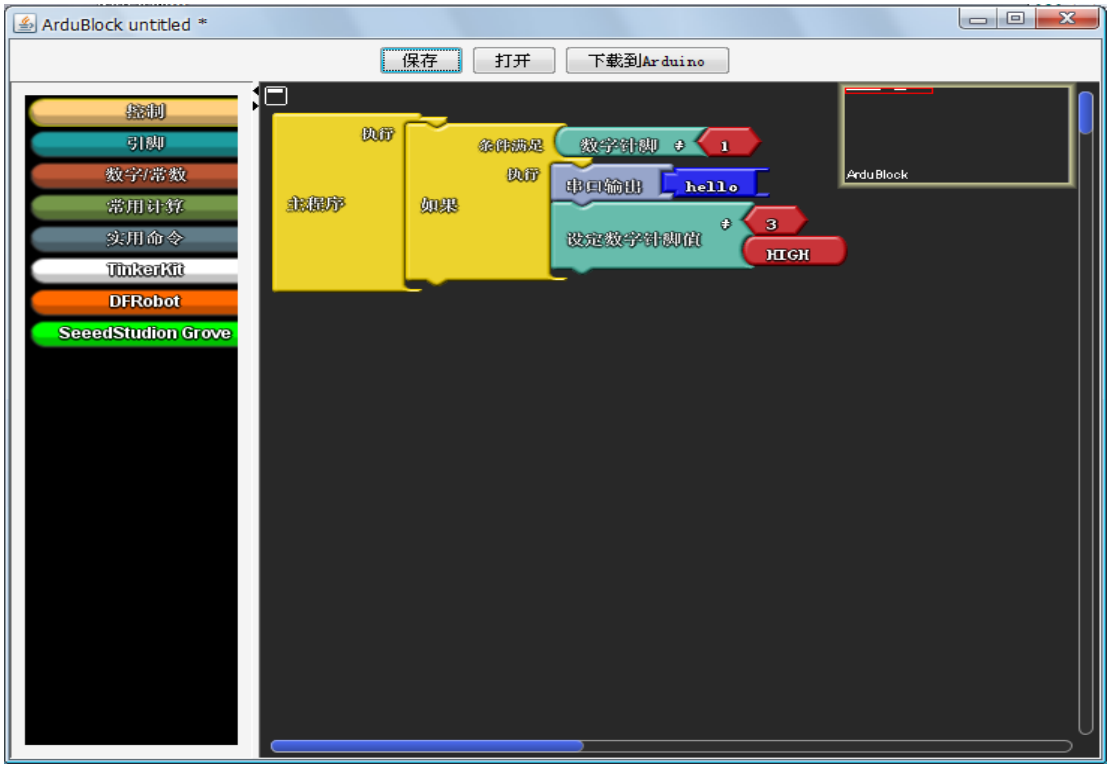

图 1-5 Ardublock 应用界面

Ardublock 的缺点与 Scratch 一样，就是不支持代码的封装。Ardublock 一样不支持用户定义自己的函数图形，在图形代码的复用性上很差。所以在某些项目中，使用 Ardublock 需要很庞大的一个图形块集合才能完成一个很简单的功能，这非常不利于编程者对代码的阅读和维护。最近的一个例子就是 7 段数码管的显示，由于不能够封装成函数，没显示一个数字，就需要很长的一个图形拼接快。这使得仅仅实现一个数字轮流显示的功能就需要 100 多块图形。此外，Ardublock 没有网络功能，用户完成自己的图形代码是不能够分享的。

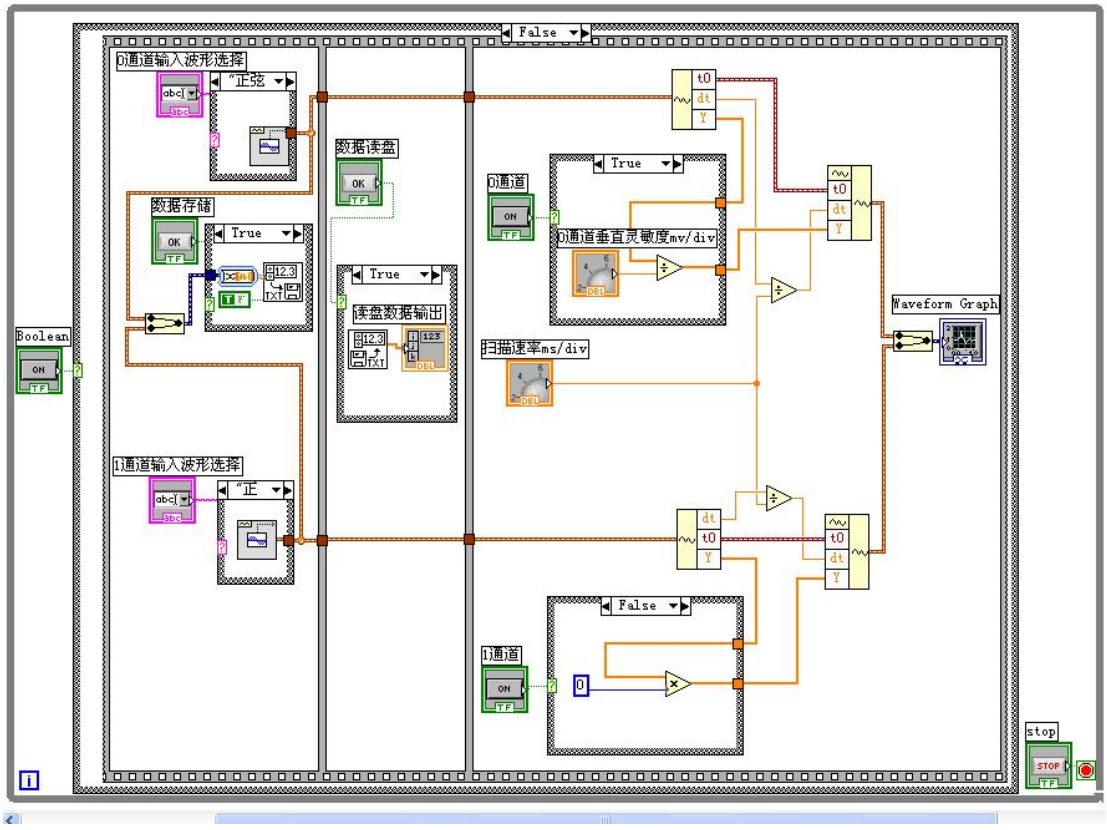

图 1-6 LabVIEW 的用户界面

LabVIEW 是一种用图标代替文本行创建应用程序的图形化编程语言。传统文本编程语言根据语句和指令的先后顺序决定程序执行顺序，而 LabVIEW 则采用数据流编程方式，程序框图中节点之间的数据流向决定了 VI 及函数的执行顺序。VI 指虚拟仪器，是 LabVIEW 的程序模块。LabVIEW 提供很多外观与传统仪器（如示波器、万用表）类似的控件，可用来方便地创建用户界面。用户界面在 LabVIEW 中被称为前面板。使用图标和连线，可以通过编程对前面板上的对象进行控制。这就是图形化源代码，又称 G 代码。LabVIEW 的图形化源代码在某种程度上类似于流程图，因此又被称作程序框图代码。

LabVIEW 在某些特定的领域中的特定特别的突出。其最初是为测试测量设计的。所以测试测量就是 LabVIEW 最广泛应用的领域。大多数主流的测试仪器、

数据采集设备都拥有专门的 LabVIEW 驱动程序，使用 LabVIEW 可以非常便捷的控制这些硬件设备。同时，用户也可以十分方便地找到各种适用于测试测量领域的 LabVIEW 工具包。这些工具包几乎覆盖了用户所需的所有功能，用户在这些工具包的基础上再开发程序就容易多了。有时甚至于只需简单地调用几个工具包中的函数，就可以组成一个完整的测试测量应用程序。

另外 LabVIEW 有着良好的跨平台性。LabVIEW 的代码不需任何修改就可以运行在常见的三大台式机操作系统上：Windows、Mac OS 及 Linux。

基于此，我们准备实现一个类似于 Scratch 但是功能更加强大的可视化编程的工具。在实现 Scratch 所用功能的基础之上。我们针对 Scratch 的缺陷加以改进。支持对代码的封装，用户可以定义函数，封装自己的库。拥有强大的网络功能，有很强的扩展性。可以在不同的平台上运行。

1.3 前期研究工作

我在开始项目之前进行了一些准备工作。对美术学院和一些工业设计专业的学生一共 78 人进行了相关的调查。这些学生在自己的项目中都需要接触到使用计算机编写程序，但是他们并不是计算机专业的学生。这些学生都有上过 c 语言的课程，但是并没有经过很长时间的培训。此外这些同学学习使用 Ardublock 可视化编程工具对 Arduino 进行过项目的开发。所以，我设计了一份问卷，了解他们对于 c 语言和 Ardublock 的一些看法。

通过问卷发现，虽然有上过 c 语言的课程。但是仍然有 62.5%的学生觉得自己对于 c 语言并不熟悉。在对 Arduino 的程序开发上，学生既可以通过直接编写 c 语言代码进行开发，也可以通过 Ardublock 进行开发，在课程学习中学生对这两种开发方式都有接触。通过调查发现，有 96.8%的学生更偏爱使用 Ardublock 进行开发，认为使用可视化编程工具开发更加的方便简单。并且大部分的学生都认为相比于 c 语言，Ardublock 更加的容易学习上手。从我在课堂上的情况反映上来看，也确实如此。同一个项目，比如驱动一个蜂鸣发声器，使用 Ardublock 开发，很多学生都能够有条理的一步步进行。但是使用 c 语言开发，则至少有一半的学生感到不知所措。而至少他们对 c 语言有过一段时间的学习，但是对 Ardublock 则仅仅接触了几个课时。

事实上，很多的人担心可视化编程语言并不能像传统的文本语言一样能够解决很多的问题。可能对于某类问题，可视化编程语言是无能为力的。特别是对于美术设计专业的学生来说，他们的设计逻辑可能一般的可视化编程工具并不能够满足。我对这个问题，作了相关的了解发现 87.1%的学生认为 Ardublock 能够很好的反映出他们的设计思路，事实上只有 64.5%的学生认为 c 语言能够反映出他们的设计思路。

在这个调查过程中，我发现目前的可视化编程工具已经能够达到预期的目的了。相比于传统的文本性编程语言 c 语言来说，可视化编程语言 Ardublock 更加的直观易学，容易上手。在普遍性上，可视化编程工具已经可以满足很多的设计领域类的一些问题。

下面两页的图 1-7 和表 1-1 是我设计的调查问卷和调查结果表。

可视化编程和文本类编程的调查表

1.你会 c 语言吗?
A.会 B不熟悉

2.编写 arduino 程序的时候，你更喜欢使用哪种方式?
A.ardublock B.c 语言

3.ardublock 和 c 语言编程，你觉得哪种更容易学习?
A.ardublock B.c 语言

4.当你的程序没有达成你想要实现的效果时，你能够很快的通过 ardublock 找到问题吗?
A.能 B不能

5.当抄写老师给出的 c 语言代码的时候，你们对这个代码每一段的功能有一个了解吗?
A.有 B没有

6.抄写老师给出的 ardublock 的程序时，你们队这个代码的每一段的功能有一个了解吗?
A.有 B没有

7.目前为止，你有没有碰到使用 ardublock 无法解决的问题?
A.有 B没有

8.如果第 7 题回答有，能简单描述下这个问题。

9.你觉得 ardublock 能否很好的表示出你编写程序时的思路?
A.能 B.不能

10.你觉得 c 语言能否很好的表示出你的设计思路吗?
A.能 B.不能

11.你熟悉 ardublock 中的每个程序块的功能吗?
A.常用块都熟悉 B.部分熟悉

图 1-7 可视化编程和文本编程调查问卷

表格 1-1 可视化编程和文本编程调查表

问题编号	选择 A 的数量	选择 B 的数量
1	29	49
2	76	2
3	70	8
4	13	65
5	8	70
6	35	43
7	50	28
9	68	10
10	50	28
11	8	70

第2章 项目实施方案

2.1 开发工具

2.1.1 JavaScript

项目采用 JavaScript 语言进行开发。JavaScript 语言是一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解释器被称为 JavaScript 引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在 HTML（标准通用标记语言下的一个应用）网页上使用，用来给 HTML 网页增加动态功能。

JavaScript 是通过嵌入或调入在标准的 HTML 语言中实现的，其出现弥补了 HTML 语言的缺陷，是 JAVA 和 HTML 折中的选择。JavaScript 具有以下的几个特点：1、JavaScript 是一种脚本编写语言，其采用小程序段的方式实现编程。同其他脚本语言一样，JavaScript 也是一种解释性语言，其提供了一个非常方便的开发过程。JavaScript 的语法基本结构形式与 C、C++、Java 十分类似。但在使用前，不像这些语言需要先编译，而是在程序运行过程中被逐行地解释。JavaScript 与 HTML 标识结合在一起，从而方便用户的使用操作。2、JavaScript 是一种基于对象的语言，同时其也可以被看作是一种面向对象的语言，这意味着 JavaScript 能运用其已经创建的对象。因此，许多功能可以来自于脚本环境中对象的方法与脚本的相互作用。3、JavaScript 具有简单性。其简单性主要体现在：首先，JavaScript 是一种基于 Java 基本语句和控制流之上的简单而紧凑的设计，从而对于使用者学习 Java 或其他 C 语系的编程语言是一种非常好的过渡，而对于具有 C 语系编程功底的程序员来说，JavaScript 上手也非常容易；其次，其变量类型是采用弱类型，并未使用严格的数据类型。4、JavaScript 具有非常高的安全性。JavaScript 作为一种安全性语言，不被允许访问本地的硬盘，且不能将数据存入服务器，不允许对网络文档进行修改和删除，只能通过浏览器实现信息浏览或动态交互。从而有效地防止数据的丢失或对系统的非法访问。5、JavaScript 是动态的，可以直接对用户或客户输入做出响应，无须经过 Web 服务程序。

JavaScript 对用户的响应，是以事件驱动的方式进行的。在网页（Web Page）中执行了某种操作所产生的动作，被称为“事件”（Event）。例如按下鼠标、移动窗口、选择菜单等都可以被视为事件。当事件发生后，可能会引起相应的事件响应，执行某些对应的脚本，这种机制被称为“事件驱动”。6、JavaScript 具有跨平台性。JavaScript 依赖于浏览器本身，与操作环境无关，只要计算机能运行浏览器，并支持 JavaScript 的浏览器，就可正确执行。

同时 JavaScript 也有一些局限性。各个浏览器厂商对 JavaScript 的支持程度是不同的。不同的浏览器在浏览一个带有 JavaScript 脚本的主页时，由于对 JavaScript 的支持不一样，其效果就会有一定的差距，有的时候甚至会完全不显示。

JavaScript 语言可以做到响应使用者的需求事件（例如表单的输入），而不需要任何的网络来回传输资料。所以当一位使用者输入一项资料时，此资料数据不用经过传给服务器（server）处理再传回来的过程，而直接可以被客户端（client）的应用程序所处理。

2.1.2 Node.js

项目是一个网络的应用。我们不仅仅需要客户端，还需要服务端。所以我们采用 Node.js 来作为 JavaScript 的运行环境。Node.js 是一个后端的 Js 运行环境，支持 windows 和 unix 的系统。这意味着我们可以可以编写系统级或者服务器端的 Javascript 代码，交给 Node.js 来解释执行。Node.js 提升了 Js 的伸缩性，使得我们的前端后端都可以用同一种语言实现。这意味着，我们在部署前端和后端的数据处理和逻辑处理量的时候，我们可以自由的将数据处理和逻辑运算来回的配置而不用耗费太多的工作量。这种功能对于复杂的前端的性能来说也有着很重要的意义，大量的数据运算和逻辑处理会极大的加重浏览器的负担，导致程序运行缓慢甚至崩溃。因为我们希望用户完成编程以后能够迅速的得到反馈，那么缓慢的运行速度将是致命的。而使用 Node.js 通过将大量的数据处理和逻辑运算移到后端去运行，可以大大减少前端浏览器的负担，提升整个应用的性能，就可以尽可能地避免运行缓慢的问题。这样就给用户更好的编程体验。

除此之外，Node.js 有着很高的性能。Node.js 采用 V8 引擎，选择了 C++进行编写，它以单进程、单线程模式运行，事件驱动机制是 Node.js 通过内部单线

程高效率地维护事件循环队列来实现的，没有多线程的资源占用和上下文切换。Node.js 通过更改连接到服务器的方式，每个连接发射（emit）一个在 NodeJS 引擎进程中运行的事件（Event），放进事件队列当中，而不是为每个连接生成一个新的 OS 线程（并为其分配一些配套内存）。这意味着面对大规模的 http 请求，Node.js 凭借事件驱动搞定一切。Node.js 很好的解决了并发连接的问题，有着极佳的性能，也是选择 Node.js 的原因之一。

Node.js 也有缺点，它的可靠性比较低。此外由于 Node.js 是单进程，单线程，只支持单核的 CPU，不能够充分的利用多核的 CPU 服务器。

2.1.3 node-webkit

其中关键的技术是，我们的开发的项目是既可以在桌面上运行也可以在浏览器上运行的，并且要具有跨平台的属性，程序在不同的平台上都是可以运行的。所以在上面的基础上，我们更进一步使用开发工具 node-webkit。node-webkit 是一个基于 chromium 和 node.js 实现的应用程序运行时环境，开发人员可运行通过 HTML(5)、CSS(3)、Javascript 来编写的本地应用程序。同时使用该工具开发的项目是跨平台的，所以很好的解决了上面的关键性问题。

2.1.4 第三方库

项目使用开源的 snap 所使用的图形库 Morphic.js 来进行界面的开发。Morphic.js 库可以允许开发者在一个 HTML 的 Canvas 元素中进行用户界面的设计。每一个 Canvas 元素用“world”来标记，其他的元素（可以看见的图形）就放在这个 world 中，用户可以操作这些元素。在这个库每一个图形就相当于一棵树的节点，可以包含任意数量的子节点。该库中的类型的继承关系如图 2-1 所示。

其中 Color 类，控制了图形的颜色，Node 类及其子类 Morph 则派生了其他所有的界面元素的类。Point 类定义了了一个元素所占的大小，Rectangle 类定义了最基本的矩形。

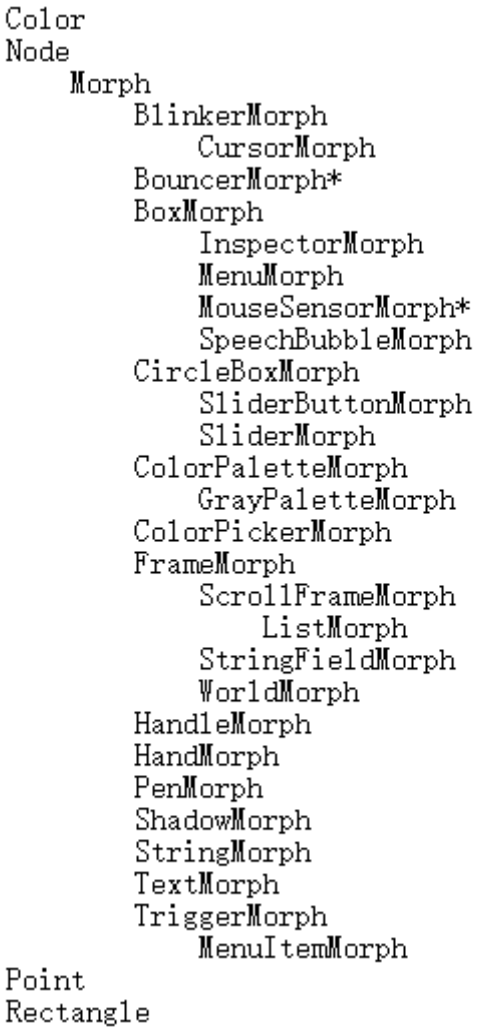

图 2-1 类继承关系图

Morphic.js 完全基于 JavaScript 和 Canvas，没有什么额外的功能。该库非常的基本，仅仅需要知道以下几个要点：

- 布进机制（一个时分复用器为每一个活跃的用户接口分配一个浏览器线程）
- 界面绘制的更新会不断的进行
- 树状结构
- 每一个 Canvas 的元素绘制出一个世界
- 每一个世界都有对应的一个控制器

Morphic.js 是一个用来在单个 HTML 的 Canvas 元素中绘制动态 GUI 的库。每一个 Canvas 元素对于其他的 GUI 中的形状元素来说就是一个世界。每一个

Morph 元素就像一个树节点并且可以包含任意数量的子 Morph。

每一个在“Morphic”世界中的可视的对象都是 Morph 对象本身。所有的对话框，菜单，菜单项目，按钮，光标等等的元素。都是用这个库中的类来绘制的，而不是通过 HTML DOM 元素来创建的。所以他们的行为，都是通过编程来设计的，而不是通过浏览器的一些属性方法来控制的。

每一个世界都有一只看不见的手（类似于鼠标光标）来控制鼠标的事件。

该库绘制界面的基本原理，就是不断的运行一个循环去绘制其中的元素，并且仅对世界中变“脏”的一些区域进行更新。在每一次的重复绘制之前，我们可以运行一些方法过程。这就可以形成一种是并行运行的错觉。

这些 Morph 类有很多的方法可以操作，其中最重要的并且最常使用的方法是如下表 2-1 所示的 9 种方法：

表格 2-1 Morph 类型基本方法表

方法名	作用
hide()	隐藏对象
show()	显示对象
setPosition(aPoint)	设定对象的位置
setExtent(aPoint)	设定对象的大小
setColor(aColor)	设定对象的颜色
add(submorph)	添加子对象到自己的顶部
addBack(submorph)	添加子对象到自己的底部
fullCopy()	复制
Destroy()	删除

所有用户的交互操作都是由事件来驱动触发的。这些触发事件都是从根节点，世界 morph 来一层一层的往下传递。世界中包含了所有的事件，这些事件都是如下方法中被初始化的：

initEventListeners()

其中有鼠标事件，键盘事件，窗口事件等。其中我们的项目中，主要处理了鼠

标的事件。下面的列表详细列出了库中需要处理的所有的鼠标事件：

- mouseDownLeft
- mouseDownRight
- mouseClickLeft
- mouseClickRight
- mouseDoubleClick
- mouseEnter
- mouseLeave
- mouseEnterDragging
- mouseLeaveDragging
- mouseMove
- mouseScroll

如果我们需要为我们的某一个图形的类添加所要的鼠标事件的时候，我们只要给该类定义相对应的函数就可以了。比如我们要添加一个鼠标的移动的事件，我们就需要定义如下函数：

```
MyMorph.prototype.mouseMove = function(pos) {};
```

需要传入的唯一的参数是一个 Point 对象，提供了目前“手”在世界中的位置信息。

最后一个提到的属性就是图形的拖动属性。当我们的鼠标左键点击对象的时候，长期按住不放并开始拖动的时候，就是调用拖动函数的时候了。

我们可以通过给一个布尔变量赋值的方式来控制我们的图形对象是不是需要可以拖动。给布尔变量名 isDraggable。如果这个属性是 false，那么图形对象就会不理会拖动这个事件。使用这个属性，我们就可以创建可以拖动的语句块了。

2.2 界面设计

项目最终是要实现一个可视化的编程应用，所以我们需要实现一个控制栏，用户通过控制栏，能够完成很多的功能。比如工程的创建保存功能，网络的登陆功能，还有一些基本的设置功能。

此外，因为最终我们希望能够通过拖动图形块的方式实现可视化的网络编程，所以我们参看 Scratch 的界面设计。将主要的界面分成 3 个部分。最左边称为调色盘，其中可以拖动的语句块都放在这个调色盘中，用户可以从调色盘中选出自己想要的语句块，并将其拖动到中间的区域内。中间的区域被称为画布，用户从调色盘中拖出的语句块就是在画布中完成拼接。最左边则是舞台，用于输出程序的运行效果。

界面设计如下图 2-2 中界面图所示：

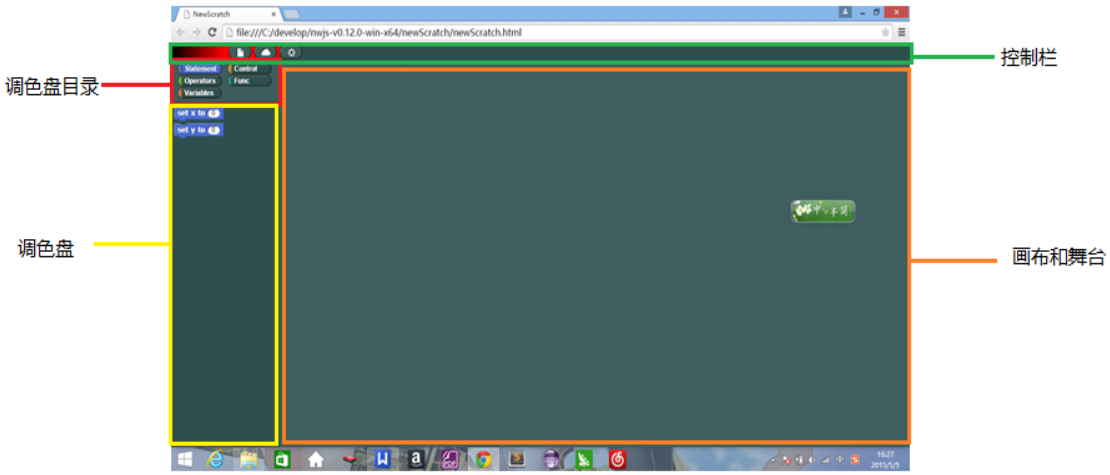

图 2-2 界面设计图

图中没有显示舞台。舞台是位于右侧的一个正方形的图形。其中最上方就是用户控制栏。第一个按钮是项目按钮，第二个按钮是网络功能按钮，第三个按钮是系统偏好设置按钮。另外控制栏，还向用户提供了控制代码运行的开始按钮，暂停按钮，这里图中没有显示。最左侧就是调色盘和调色盘目录。

其中调色盘目录和调色盘如图 2-2 和图 2-3 所示。图 2-2 是调色盘目录。调色盘的目录显示了所有的语句块的类型，每一种类型的语句块有统一的颜色。每一种类型整齐的排列在一起，使得用户不会产生找不到所需要语句块的问题。目前我们设计出来的语句块分类有 5 类，分别是动作语句，控制语句，计算语句，函数和变量。图 2-3 显示的是调色盘，我们可以发现，语句块整齐的排列下来，供用户选择需要的语句块。

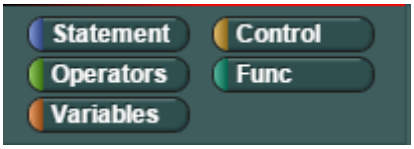

图 2-2 调色盘目录

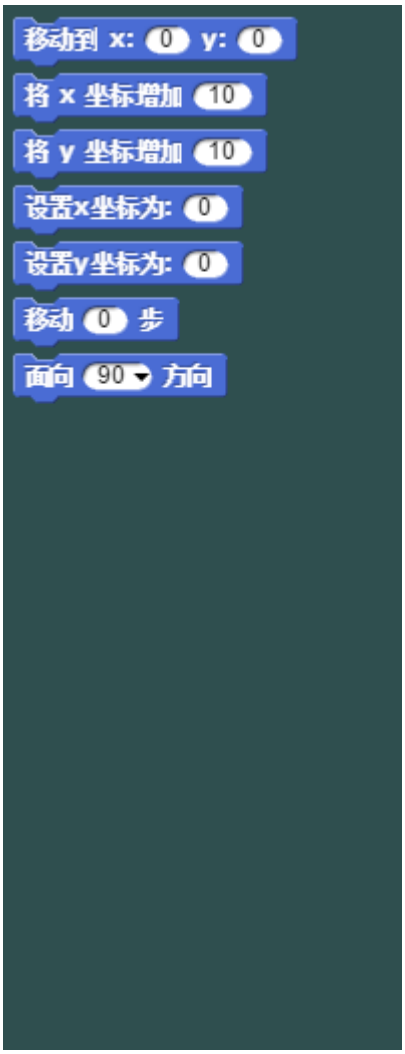

图 2-3 调色盘

2.3 项目模块

项目目前主要分为如下几个模块，用户界面模块，界面元素对象模块，语句块模块，和界面控件模块。各个模块的功能和文件名字如表 2-2 模块表所示：

表格 2-2 模块表

模块名	文件名	功能描述
用户界面模块	gui.js	生成用户界面，可以控制用户界面的样式。调用其他模块中的界面控件。
界面元素模块	object.js	操作界面的一些特别的元素的模块。里面的类管理一些界面的元素。如调色板，调色板的目录等的属性和方法
语句块模块	block.js	负责语句块的生成。所有的编程语句块都在该模块中定义。
界面控件模块	widget.js	按钮对象等界面的控件对象在该模块中定义，给其他模块调用
线程模块	thread.js	线程模块，完成一些语句的执行功能

各个模块的相互依赖关系如图 2-4 所示：

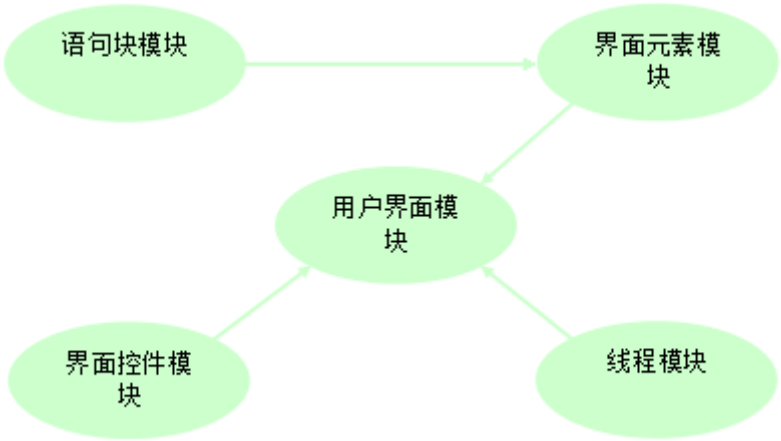

图 2-4 模块依赖图

其中最主要的模块就是用户界面模块。其中主要的类型 myIDE 是世界的唯

一的子节点，其他所有的元素如调色盘，画布，控制栏等都是 myIDE 的子节点。所有的元素都是在该类中实例化，并添加到界面上绘制出来的。

界面元素模块定义了用户界面中的一些布局的元素。比如调色盘类，舞台类。控制调色盘中绘制的标签和一些基本的样式。

语句块模块，则定义了所有的语句图形块类。

界面控件模块定义了需要用到的按钮。

线程模块，定义了线程管理类，过程类来实现语句的解释执行。

2.4 本章小结

这个项目是想要实现一个既能够作为桌面程序运行，也能够在浏览器上运行的应用。同时，我们计划实现的项目具有跨平台的特点，我们最后完成的应用的运行能够不依赖于操作系统，编译一次就能够在各个平台上运行。

所以项目实现的开始决定采用的语言是 JavaScript。JavaScript 是一种基于对象（Object）和事件驱动（Event Driven）并具有相对安全性的客户端脚本语言。同时也是一种广泛用于客户端 Web 开发的脚本语言，常用来给 HTML 网页添加动态功能，比如响应用户的各种操作。它最初由网景公司（Netscape）的 Brendan Eich 设计，是一种动态、弱类型、基于原型的语言，内置支持类。

JavaScript 是一种跨平台性的语言。JavaScript 依赖于浏览器的本身，与操作系统无关。只需要计算机中能运行支持 JavaScript 的浏览器，那么就能够正确执行。

同时我们的项目作为一款网络应用。希望能够加快网络传输的速度，也希望能够减少服务器的负担。而 JavaScript 的使用可以让一些传统的数据提交和验证的工作不用通过网络传输到服务端去执行，这些工作都能够在客户端完成。当用户很多数据量很大时，能够很好的节省网络和服务器端的资源。

在此基础之上，我们选择 Node.js 作为 JavaScript 的运行环境。Node.js 的设计目标就是“旨在提供一种简单的构建可伸缩网络程序的方法”。Node.js 有着如下的几个特点：

1.依赖于 Chrome V8 引擎进行代码解释

2.事件驱动

3.非阻塞 I＼O

4.轻量、可伸缩，适于实时数据交互应用

5.单进程，单线程

简而言之，我们可以使用 Node.js 实现实时的，双向连接的 web 应用，客户端和服务端都能够发起通信，能够自由的的交换数据。因为这些特点，使用 Node.js 开发的应用有着更好的性能。

更进一步，因为我们的应用不仅仅能够在浏览器上运行，还需要能够作为桌面程序运行。所以我们还需要工具 Node-webkit 来帮助我们进行相关的开发。Node-Webkit 是 NodeJS 与 WebKit 技术的融合，提供一个跨 Windows、Linux 平台的客户端应用开发的底层框架，利用流行的 Web 技术（Node.JS，JavaScript，HTML5）来编写应用程序的平台。这样使用该工具不仅能够开发桌面应用程序，还解决了跨平台的问题。

此外，在图形的绘制上，我们使用了开源项目 snap 的图形库，该图形库对 javascrpt 的绘图函数进行了封装。我们使用该图形库，能够很块的绘制出我们的图形。

界面设计我们使用了很多可视化编程工具采用的经典的界面布局。调色盘，画布和舞台构成了用户交互界面的主体。另外在页面的最顶部加了控制栏，辅助用户使用。

调色盘是语句模块的容器，调色盘分为两个部分。上部分是目录，下部分是语句块容器。用户通过点击目录中不同的类别，可以切换语句块的容器。语句块容器中是语句模块的集合。每一个语句块是一个对象，可以拖动拼接，有些语句块允许用户输入数据。用户如果想要编写一条语句，索要进行的操作就是将调色盘中的语句拖动出来。

语句从调色盘中拖动出来以后，就需要拖动到画布当中，其中画布是供用户进行语句拼接的地方。同时画布会记录语句块之间的拼接情况，和语句的数据情况。以便进行后续的执行。

对于界面的最终的实现，依赖于 5 个模块，用户界面模块，界面元素对象模块，语句块模块，界面控件模块和线程模块。用户界面模块，设计整个 gui 界面，为程序提供了入口。界面元素对象模块，提供了设置主界面元素的类，如调色板的类。语句块模块定义了所有的语句块。界面控件模块定义了界面设计所要用到的按钮。最后对于语句的解释执行，则由线程模块来完成。

第3章 在项目中负责的具体工作

3.1 项目的构建

3.1.1 项目的配置

首先要配置 Node-webkit 工具。开发 Node-webkit 工程，代码的目录结构如下图 3-1 基本程序结构图所示：

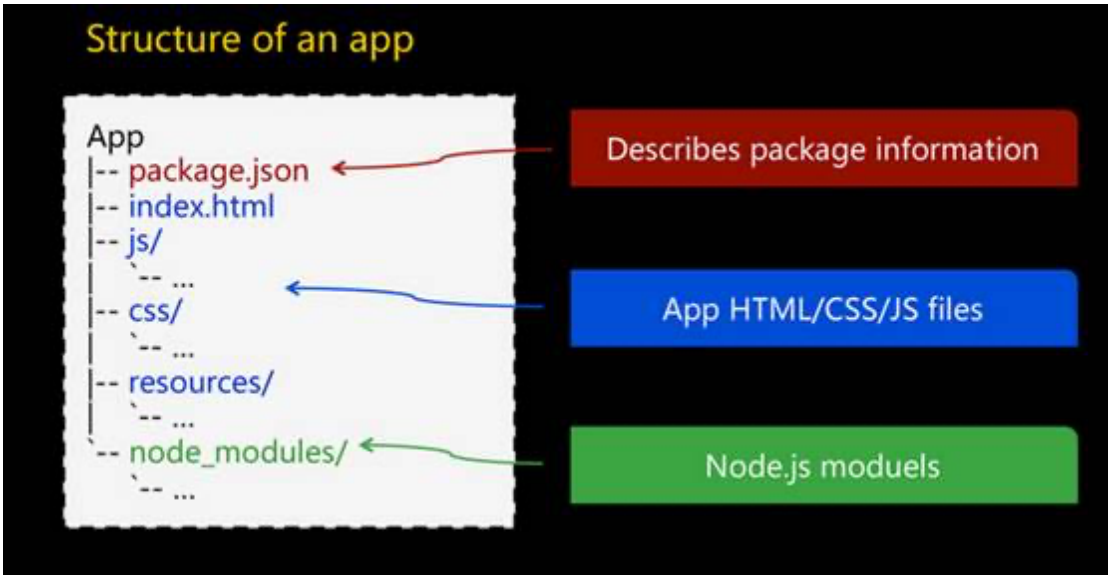

图 3-1 基本程序结构图

上图就是一个 nw 程序的基本组织结构，在根目录下的 package.json 是程序的配置文件；Index.html（可以是任意的文件名），是应用启动的界面；js/css/resources/，应用的样式、脚本、html、图片等资源文件；node_modules 存放 node.js 的扩展组件。

nw 在启动应用程序的时候，首先要读取 package.json 文件，初始化一些基本的属性。其中必须配置的属性有 main 属性和 name 属性，main 属性指定了程序的起始页面，name 属性则是项目的名字。

表 3-1 package.json 属性表详细说明了在本项目中，我在配置文件中设定的

各个属性，和其作用。其中 windows 属性中很多子属性，主要是定义项目作为桌面文件运行时窗口的长宽，有没有工具栏等一些基本的属性定义，较为繁琐，就不在赘述。

表格 3-1 package.json 属性表

属性名	属性值	属性描述
name	newScratch	应用的名称
main	newScratch.html	应用的启动界面
version	1.0.0	版本名
nodejs	true	启用 webkit 对 node 的支持
window		启动后的窗口属性设置

3.1.2 应用启动界面

应用启动界面是一个 html 文件。因为本项目使用了 snap 的第三方图形库，该图形库使得我们能够仅需要操作一个 html 元素，就能通过 js 完成对界面的绘制。所以启动页面的 html 文件中的 body 标签下只有一个元素：

```
<canvas id="world" tabindex="1" style="position: absolute;" />
```

之后插入 js，获得这个元素，并当做参数实例化 WorldMorph 类，如下所示：

```
world = new WorldMorph(document.getElementById('world'))
```

就可以使用类中 WorldMorph 的方法来操作这个元素。

我们获得 world 对象，之后以改对象作为参数，实例化 gui 对象，绘制出整个应用的用户界面，如下所示：

```
new myIDE().openIn(world)
```

并调用 doOneCycle()方法，使得程序会不断的循环执行。

3.2 用户交互界面

项目使用了 snap 的第三方图形库来绘制用户交互的界面。使用该库的原理在第二章中曾经提到过，所有的图形绘制的结构类似于一个树的结构，每一个树节点都是一个 morph。我们通过给树节点的 morph 添加子节点 morph 的方式，进行图形的绘制。我们的用户界面的 morph 树结构如下图 3-2 所示：

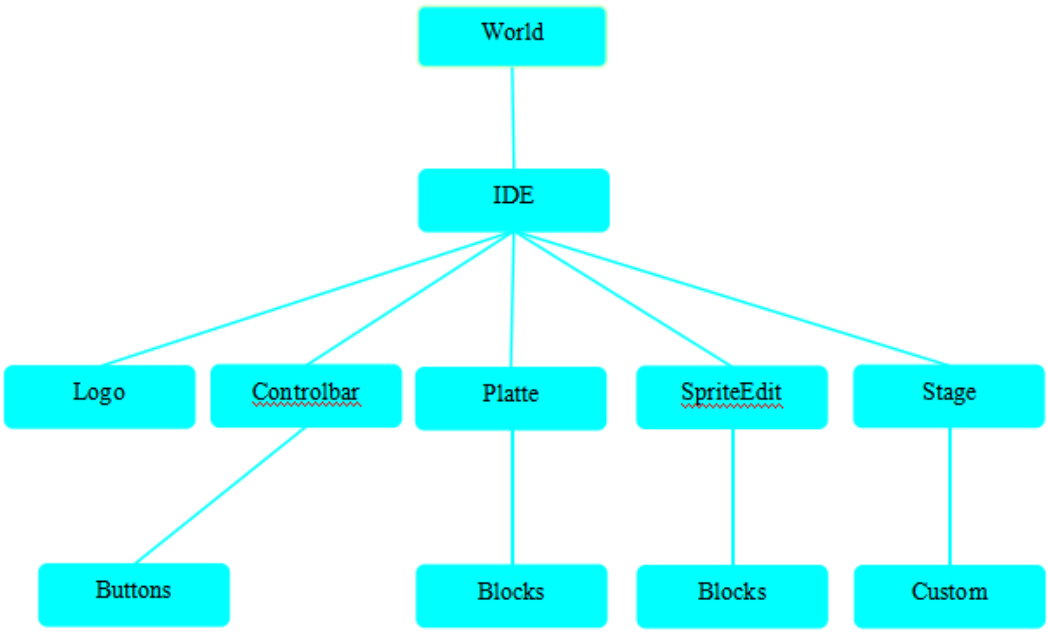

图 3-2

其中，根节点 world，在启动页面中创建，并作为参数传入到 IDE 的方法中。在 IDE 的方法中，world 调用 add 方法，将 IDE 添加都 world 中，如下所示：

```
myIDE.prototype.openIn = function (world) {

        this.buildPanes();
        world.add(this);
        this.reactToWorldResize(world.bounds);

};
```

其中 IDE 对象调用了 buildPanes 方法，为自己添加了 logo，controlbar，platte，

spriteEdit 等子图像。

buildPanes 方法绘制了整个 IDE 的用户交互界面，项目中如果要对用户交互界面进行一些修改，那么就只要在该方法中调用新的方法，就能够添加一些新的布局元素。目前本项目中有五个布局的元素。第一个布局元素是项目 logo，添加该布局元素，调用方法：

```
myIDE.prototype.createLogo()
```

改函数，在左上角显示一个矩形的图形。同时该元素为其他布局元素设立了一个长宽大小和所处位置的基准。以此为准，我们调用如下方法，为项目添加菜单栏布局：

```
myIDE.prototype.createControlBar()
```

菜单栏中包含了几个按钮，按照功能命名为项目按钮，云按钮，和设定按钮。在运行功能上有启动按钮，暂停按钮，停止按钮等。这个方法中主要负责以下几个功能。首先创建整个菜单栏的轮廓图形，餐单栏的高度于 logo 的高度相等。之后再调用创建几个按钮对象，通过对其属性的赋值来设定按钮的样式。最后通过方法：

```
myIDE.prototype.controlBar.fixLayout()
```

设置按钮的位置。之后则是创建调色盘和调色盘目录。分别是调用如下两个方法来添加这两个布局元素：

```
myIDE.prototype.createCategories()
myIDE.prototype.createPalette()
```

这两个方法，分别生成调色盘和调色盘目录的图形轮廓。同时访问界面元素对象模块中定义的目录数组。对每一个数组元素都生成相应的按钮。调色盘中语句块，则也依赖于界面元素对象模块中的方法来进行生成。下一个元素是画布元素。调用如下方法：

```
myIDE.prototype.createSpriteEditor()
```

该方法，创建了画布的布局元素并且设定了几个必要的属性。

最后则是舞台元素。当用户完成了编程之后，需要查看程序的效果。则查看的地方就是舞台。生成舞台需要调用如下的方法：

```
myIDE.prototype.createStage()
```

改方法，绘制出了舞台。并且在舞台中绘制了一个默认的人物，一个箭头。当我们不特别的为舞台添加人物的时候，箭头就是默认的人物。

各个方法的调用顺序如下页的图 3-3 所示。

图 3-3 界面绘制函数调用顺序图

此外，GUI 类还有其他的方法，。这些方法，是当用户点击按钮的时候，调用的方法，分别是暂停，开始，事件，设定等按钮的方法。下面图 3-4 显示了这些方法：

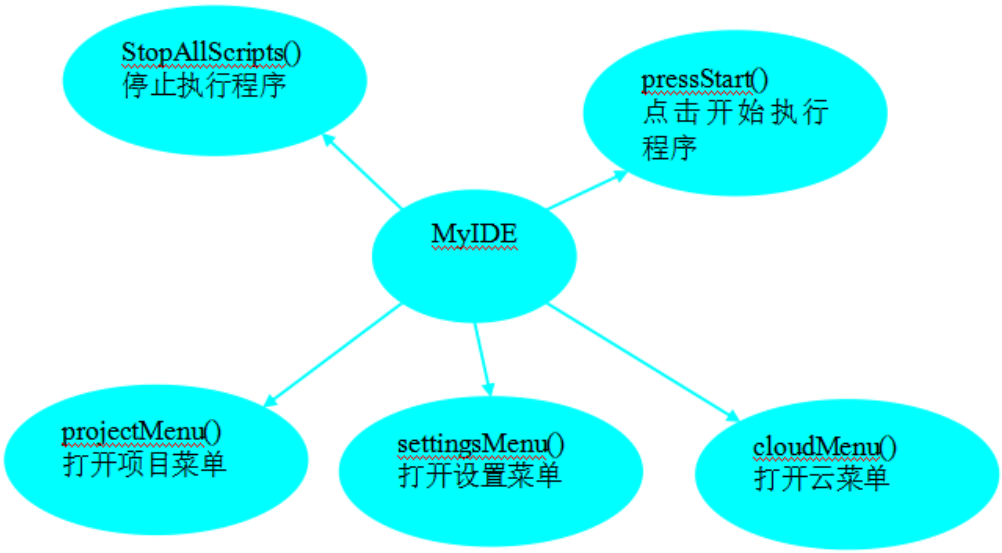

图 3-4 GUI 类的按钮触发方法

3.3 语句块的设计定义

首先设计的是语句块的目录。这由一个字符串数组所定义：

SpriteMorph.prototype.categories

其中 IDE 对象通过访问该数组获得语句块的分类的信息，并通过数组的字符串来生成按钮的标签。另外，每一个类别的按钮的颜色信息则在另一个颜色数组中定义：

SpriteMorph.prototype.blockColor

这是由颜色对象组成的数组，改变里面的颜色设置就可以改变用户界面中标签按钮的颜色。

其中，我们通过定义结构体的方式来定义新的语句块。每一个结构体的主要定义属性如下表 3-2 语句块结构体所示：

表格 3-2 语句块结构体表

属性名	例子	属性描述
type	‘command’ ‘reporter’	语句块功能的类型
category	‘control’ ‘statement’	语句所在的目录标签
spec	'设置 x 坐标为: %n'	语句的提示信息，其中%n 表示这里提供用户输入信息
defaults	[0]	数组，提供默认的数据

进入启动页面之后。在 myIDE.prototype.createPalette 方法中，传入目录数组作为参数调用了界面元素对象模块中的 SpriteMorph 的 palette 方法，进行调色板中语句图形块的生成。platte 方法中，首先查看 cache 中，有没有已经绘制好的调色盘，如果没有则调用 SpriteMorph.prototype.freshPalette 方法绘制调色盘。在该方法中，首先获得要绘制的语句块的对象数组，获得该数组后，遍历该数组，生成相应的图形块并排列显示。通过调用 SpriteMorph.prototype.blockTemplates 方法，来生成要绘制的图形块数组，需要传入目录名作为参数，返回改目录下

的语句块对象数组。该方法中实现了很多函数，用来生成语句块的对象。生成语句块对象的函数是调用了函数 SpriteMorph.prototype.blockForSelector 来生成语句块的对象，传入语句块结构体的名字作为参数，返回改语句块的对象。该方法，根据传入的结构体的名字，访问语句块结构体数组获得要生成对象的语句块的结构体。根据结构体的 type 属性，生成相应的图形对象，绘制出大致的图形。最后则更具 spec 属性和 default 属性，完成提示信息的书写和数据的输入。详细的调用过程如图 3-5 函数调用顺序图所示：

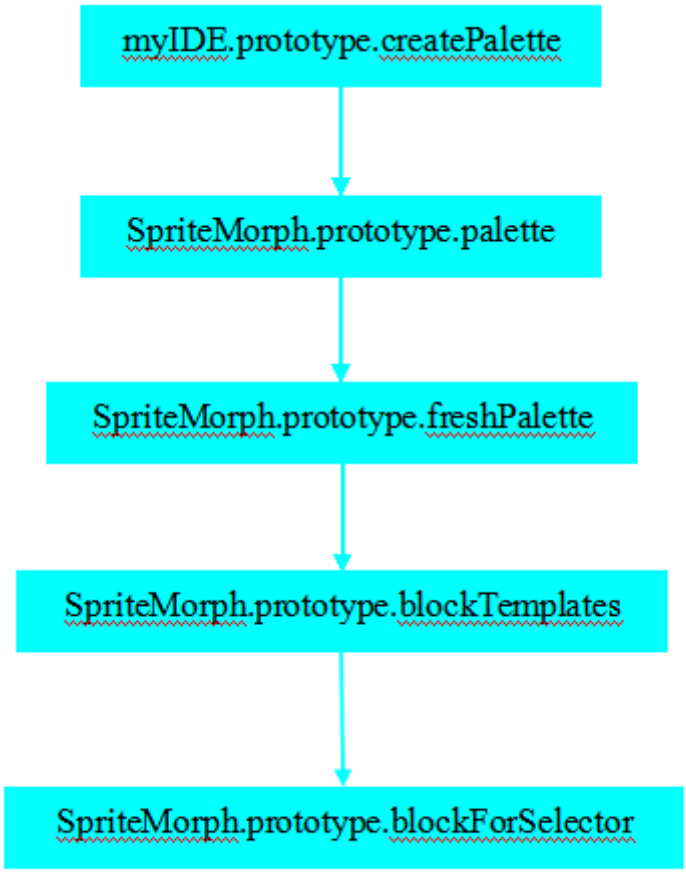

图 3-5 函数调用顺序图

其中方法 blockForSelector 的运行流程图如下图 3-6 所示：

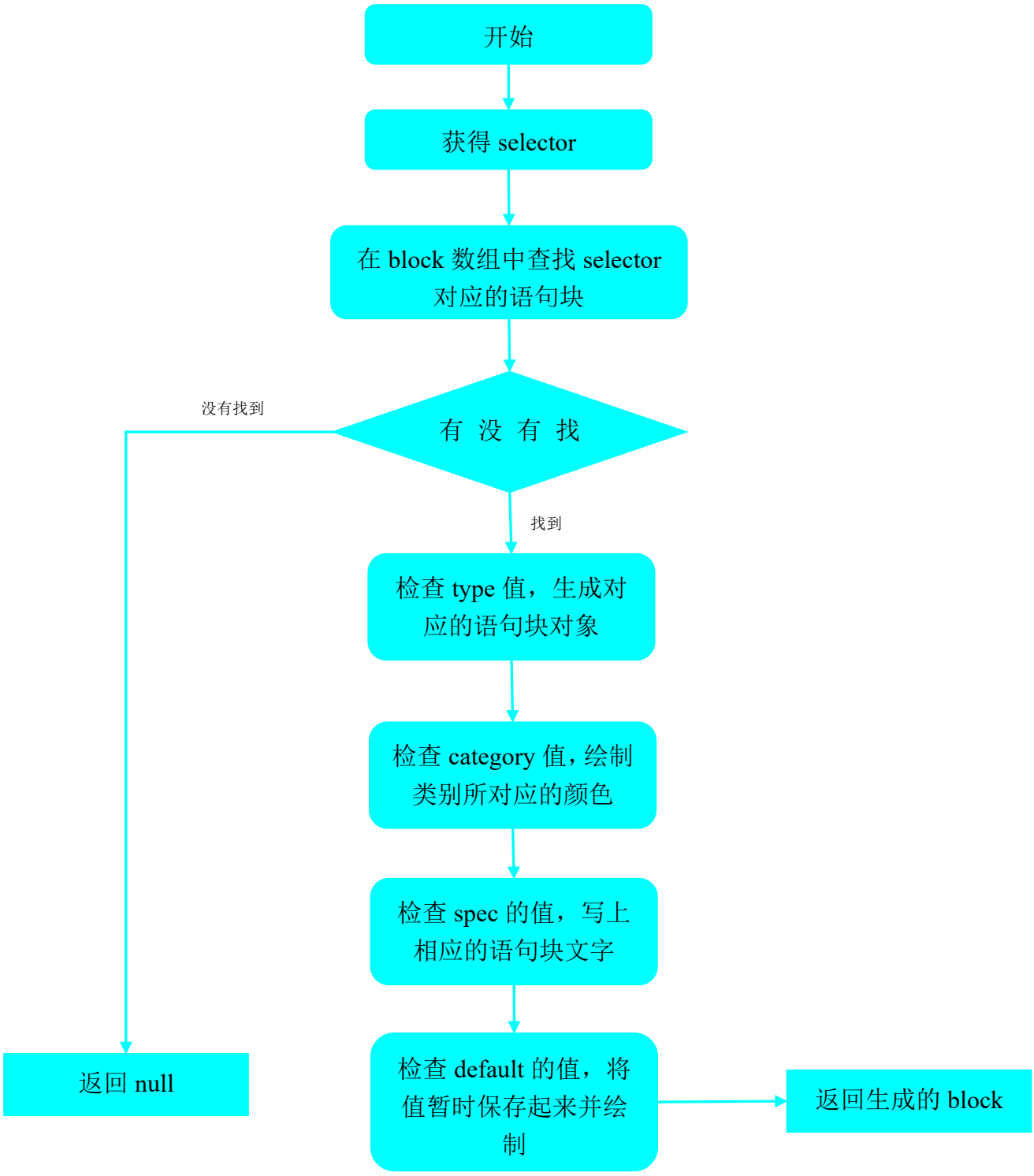

图 3-6 blockForSelector 运行流程图

3.4 控件的定义

控件模块中是对按钮的类进行定义。用户交互界面模块中通过实例化其中的类来完成。其中最基础的类为 PushButton，PushButton 是 morphic.js 中定义的 TriggerMorph 类的子类。控制 PushButton 的属性的几个变量，如下表 3-3 按钮属性表所示：

表格 3-3 按钮属性表

属性名	描述
fontSize	按钮上字体的大小
fontStyle	字体
labelColor	标签的颜色
labelShadowColor	标签阴影的颜色
color	按钮处于自然状态时的颜色
pressColor	按钮处于点击状态时的颜色
highlightColor	背景高亮时的颜色
outlineColor	边线的颜色

这些属性在定义的时候，具有一个默认的值。但是也可以在实例化一个按钮对象之后，对这些属性进行赋值。

实例化一个按钮对象，必须传入三个参数：按钮所在的上下文、点击按钮要调用的方法名，按钮的标签图形。

该模块一共定义了 4 种类型的按钮，继承关系如下图 3-7 所示。按钮中的方法一共有两类，一类方法是调用来绘制生成按钮的，另一类则是定义了按钮的鼠标事件。按钮的两类方法如图 3-8 按钮的绘制方法图和图 3-9 按钮的鼠标方法图所示。

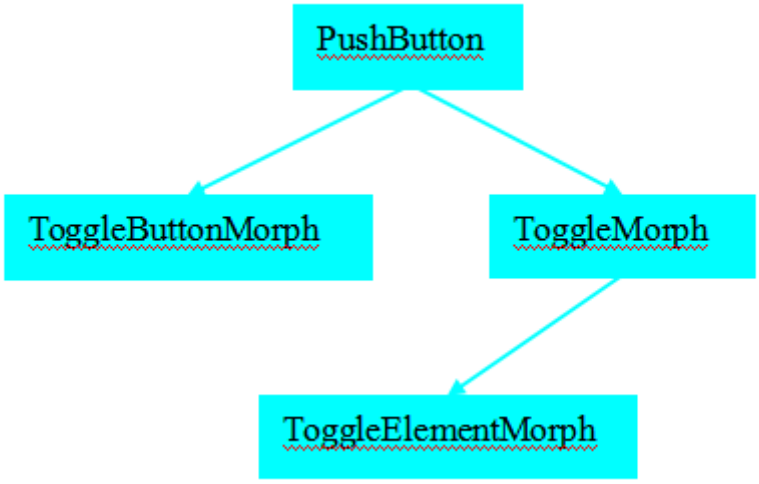

图 3-7 按钮对象继承关系图

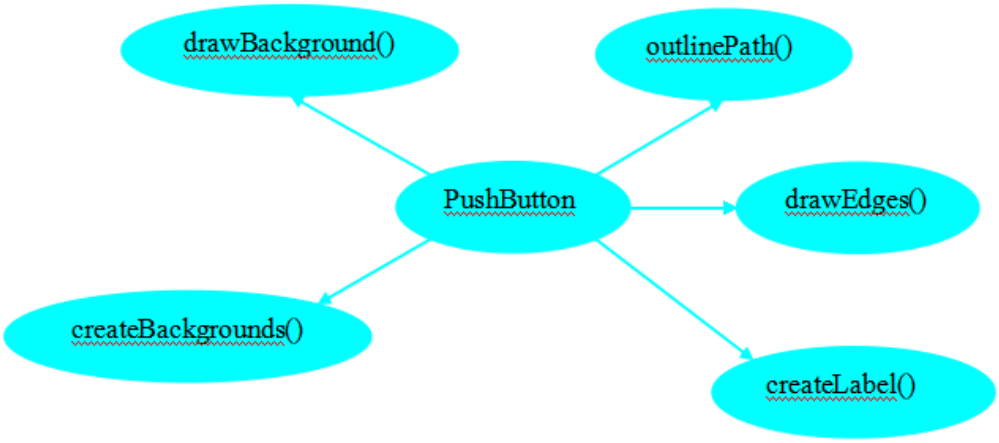

图 3-8 按钮的绘制方法图

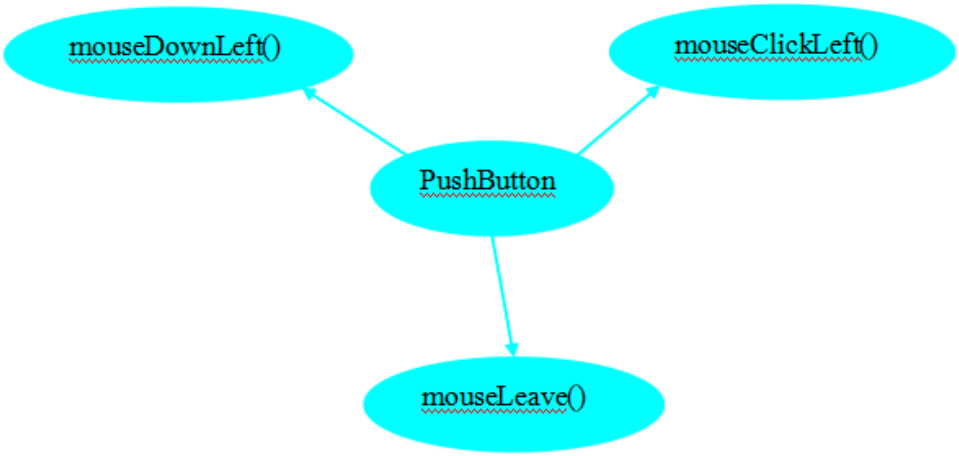

图 3-9 按钮的鼠标方法图

3.5 程序的执行

在应用的设计中，需要有一个起始块。改块规定了使得程序运行的事件。目前该事件默认为点击按钮执行。所以在控制类型块中，需要定义一个起始块，提示用户点击开始的按钮，这个按钮如图 3-10 所示：

图 3-10 开始按钮块

开始按钮位于控制栏右边，如图 3-11 所示：

图 3-11 按钮

上图中最左侧的按钮就是其实按钮。当点击该按钮的时候。则开始执行程序。

解释执行拼接而成的语句块的最基本的原理。就是当语句块拖入到画布中完成拼接的时候，我们记录拖入的语句块，每发现一个结构控制语句块就创建一个栈，之后在读到下一个结构控制语句块之前将每一个语句块其压到这个栈当中。当点击开始按钮的时候，我们按顺序读取栈。每读取一个栈顶的元素，就解释该语句块所要做的操作，并且执行。对于舞台来说，我们通过不断的刷新舞台。根据语句，将舞台的画面分成好几种执行情况，按照每种情况，对舞台进行绘制。之后重复绘制舞台，从而生成语句所执行的效果。这些工作，我们使用一些线程来完成。

当点击开始按钮的时候，我们执行如下的方法：

```
myIDE.prototype.pressStart();
```

该方法执行了一些按钮点击必需的操作后，执行下面的方法，开始刷新舞台实现执行的效果：

```
myIDE.prototype.runScripts();
```

该方法调用了舞台对象中的方法 fireGreenFlagEvent()，来实现执行效果。

方法 fireGreenFlagEvent()中的运行逻辑如下图 3-12 所示：

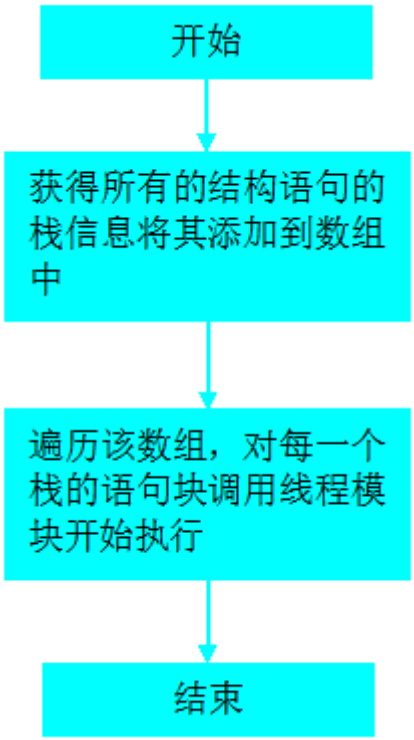

图 3-12 执行流程图

由于在拼接的时候，每一个语句块，都会获得其上一块语句块的信息。所以对于任何一个语句块，我们都可以找到该语句块所在栈的栈顶语句块的信息。从而可以获得整个语句块的排列。之后只要按照顺序读取语句块的信息，并对舞台中人物进行相应的绘制，就可以实现语句的解释执行。

3.6 本章小结

我在项目中，主要负责的工作，就是项目初始的结构构建和对界面进行绘制并设计添加了一些语句块。

对于项目结构的构建。因为我们要实现一个能够在浏览器和桌面上都能够运行的应用。所以最初的工作就是构建一个 Node-webkit 的项目，编写 package.json 文件。并创建应用的启动页面文件。

根据目前实现功能的划分，将项目暂时划分成 4 个模块。主要的模块是用户界面模块。该模块定义了整个应用的界面属性。每一个界面布局元素的位置都得到了确定，并通过函数调用的方式往整个应用添加。

另外，定义了语句块的类型。使用一个数组来控制语句体的分类。用定义结构体的方式来添加新的语句块。这样的数据结构的设计使得以后无论是增加新的语句类型或是在某一语句类型内增加新的语句都能很方便的执行。使得应用有很好的扩展性。另外定义了一些方法来对语句块进行了绘制，使其能够整齐的显示在界面的调色盘内。

最后则是相关的控件的定义。主要是按钮，设计按钮的偏好属性，和鼠标的点击事件。并且定义了一些方法来绘制按钮。

对于程序的解释执行。记录了拖入到画布中的语句块的自身信息和他们的排列顺序的信息。并规定了程序开始运行的事件——按钮的点击。当按钮点击事件触发后，重新读取这些语句块的信息。并且按照这些语句块的语义，重复的刷新舞台的绘制图形，来实现语句执行的结果。

第4章 项目成果

完成了界面的基本的设计，除舞台外其余部分的绘制与功能基本完成。

完成了用户界面类的基本属性，和目前需要用到的方法。接下去，可以通过定义新方法的方式对界面进行扩展。还可以通过对用户界面对象的属性的值进行更改的方式对用户界面的偏好进行修改更新。

用户界面中的元素，控制栏，调色板，画布，舞台，4 个必要元素中实现了控制栏，调色板，画布的绘制。

在控制栏上定义了所需要的按钮和菜单的样式。后续工作只需要完善定义函数的功能。

调色板完成了对于调色板目录的绘制和数据定义，也完成了语句块的设计和定义。语句块可以整齐的显示在调色板上，并且可以用鼠标对语句块进行拖动。

画布进行了简单的绘制。语句块可以在画布上进行拼接。用户可以输入数据。

绘制了白背景的舞台，以及默认的箭头。通过语句的编写，可以操作这个箭头的一些行为。

项目最基础的功能完成，用户可以通过拼接的手段，完成一些最简单的代码编写，来控制一个箭头。

参考文献

[1] Jose A. Borges，Ralph E. Johnson Multiparadigm Visual Programming Language

[2] Sergio Sandoval-Rey，,Pedro Galicia-Galicia，Ivan Gutierrez-Sanchez Visual Learning Environments for Computer Programming

[3] Keiji Kojima, Yoshiki Matsuda, Seiji Futatsugi LIVE - Integrating Visual and Textual Programming Paradigms

[4] Jose A. Borges，Ralph E. Johnson Multiparadigm Visual Programming Language

[5] Robert Schaefer On the Limits of Visual Programming Languages

[6] Geoffrey G. Roy，Joel Kelso，Craig Standing Towards a Visual Programming Environment for Software Development

[7] Yoshiharu Kato ，Musashi University Splish: A Visual Programming Environment for Arduino to Accelerate Physical Computing Experiences

[8] http://www.infoq.com/cn/articles/what-is-nodejs/

[9] 《深入浅出 Node.js》 朴灵 人民邮电出版社

[10] 《JavaScript 高级程序设计(第 3 版) 》 Zakas. Nicholas C. 人民邮电出版社

致谢

本论文的工作是在我的导师 XX 教授的悉心指导下完成的。XX 老师严谨的治学态度和科学的工作方法给了我极大的帮助和影响。同时，这一年来 XX 老师提供我多次担任助教的机会，让我能够有机会接触到很多非计算机专业的学生，让我能够得到他们对于编程的理解和看法。给我的项目提供了宝贵的意见。此外，XX 老师悉心的指导我的实验室工作，对我的项目的开发和论文都提出了许多的宝贵的意见，在此对 XX 老师表示衷心的谢意。

在实验室工作和撰写论文期间。XX、XX 等同学和 XX 先生对我论文中的研究工作给予了热情的帮助，在我的开发过程中给我提供了宝贵的建议，在此表示衷心的感谢。

最后也感谢我的家人，他们的理解和支持使我能够在学校专心的完成我的学业。

本科生毕业论文（设计）任务书

一、题目：

二、指导教师对毕业论文（设计）的进度安排及任务要求：

1.能够完成基本的类的设计
2.语句块可以拼接
3.能够实现简单语句的解释执行
4.界面主要元素能够设计完成
5.了解可视化编程工具的当前研究状况

起讫日期 200 年 月 日 至 200 年 月 日

指导教师（签名） 职称

三、系或研究所审核意见:

负责人（签名）

年　月　日

毕 业 论 文（设计） 考 核

一、指导教师对毕业论文（设计）的评语：

指导教师(签名）

年 月 日

二、答辩小组对毕业论文（设计）的答辩评语及总评成绩：

成绩比例	中期报告 占（10%）	开题报告 占（20%）	外文翻译 占（10%）	毕业论文（设计） 质量及答辩 占（60%）	总 评 成绩
分值					

答辩小组负责人（签名）_____

年　　月　　日