前景（简化版）

# 简介

## 目的

本前景文档用于分析MY PC LOGO的用户需求，明确软件的产品定位和发展方向，以达到提高用户的软件使用体验、减少产品风险、抓住产品机遇的目的。

## 范围

MY PC LOGO旨在开发一个简单的Logo语言的开发环境，供少儿用户进行编程入门学习。用户通过命令行输入控制流指令，控制一个虚拟PC Logo小海龟在屏幕上移动、绘图。此外，我们还会创建线上协同绘图环境，提供脚本保存及上传功能。

## 定义、首字母缩写词和缩略语

|  |  |
| --- | --- |
| Command | 解释 |
| FD | 前进 |
| BK | 后退 |
| FT | 向左转 |
| RT | 向右转 |
| CLEAR | 清屏复位 |
| PU | 提笔 |
| PD | 落笔 |
| SETPC | 设置画笔颜色 |
| SETBG | 设置画布背景色 |
| STAMPOVAL | 画圆 |
| REPEAT | 重复命令 |
| HOME | 移到行首 |
| END | 移到行尾 |

## 参考资料

[1]沈备军&陈昊鹏&陈雨亭 《软件工程原理》 高等教育出版社

## 概述

本文档将分析产品的市场定位、明确用户与涉众，定义产品特性与产品的性能、环境等各方面需求，从而展现本软件的开发前景。

# 定位

## 商机

市场上缺少轻量级的少儿编程软件。现有的少儿编程软件（如scratch）功能虽丰富，但是有些冗杂，想要精通需要投入大量的时间。MY PC LOGO虽然你功能简单，可以让儿童在较短的时间内理解编程逻辑，并通过命令绘图获得即使反馈和成就感；通过在线多人协同绘图，儿童能够在与伙伴交互的过程中学习玩耍，让编程不再是一件枯燥的事。

此外，MY PC LOGO适合作为编程科普的载体，将编程的魅力融合在最简单的必要功能中，精通成本低，是科普载体的不二选择。

## 问题说明

|  |  |
| --- | --- |
| 问题是 | 少儿编程软件不够轻量 |
| 影响 | 想要感性认识编程的儿童 |
| 问题的后果 | 学习的时间成本高容易使儿童失去兴趣 |
| 成功的解决方案 | 在较短的时间内可以精通、即时获得变成成就感、可以与伙伴交互 |

## 产品定位说明

|  |  |
| --- | --- |
| 针对于 | 儿童、家长、教师 |
| 谁 | 学习、激发儿童的学习兴趣、科普编程的载体 |
| 该（产品名） | 儿童编程软件 |
| 功能 | 编程绘图，可以和伙伴交互 |
| 不同于 | Scratch |
| 我们的产品 | 轻量级，只保留最必要的编程功能，短时间内可以精通 |

# 涉众和用户说明

## 市场统计

• **团队希望获得的声誉**： 受欢迎的少儿编程软件开发团队。

• **后续开发**：构建一个线上展示平台，展示用户上传的绘图作品。

## 涉众概要

|  |  |  |
| --- | --- | --- |
| **名称** | **说明** | **角色** |
| 开发者 | 软件开发人员 | 负责软件开发质量 |
| 项目经理 | 项目经理 | 把握软件需求，负责软件开发进度 |
| 幼儿 | 儿童 | 使用软件，确保软件满足学生的要求 |
| 家长 | 家长 | 确保家长原以为软件付费 |
| 老师 | 老师 | 确保老师愿意使用MY PC LOGO进行儿童编程教学 |

## 用户概要

|  |  |  |
| --- | --- | --- |
| **名称** | **说明** | **涉众** |
| 幼儿 | 通过软件学习编程，同时获得绘画的乐趣 | 幼儿 |
| 老师 | 通过软件进行教学 | 老师 |

## 用户环境

单机绘图，也可以在线多人协同绘图。

一个课时内掌握软件功能；半天内绘制有观赏性的画面。

用户可以在windows、macOS上使用该软件。

## 关键的涉众/用户需要

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **需要** | **优先级** | **关注的要点** | **目前的解决方案** | **提议的解决方案** | |
| 流畅绘图，软件不崩溃 | 高 | 软件健壮性 | 提高代码质量，并进行测试 | | 准备第二套技术方案 |
| UI美观 | 高 | 符合用户习惯 | 模仿优秀的软件产品 | | 界面不断迭代 |
| 多人在线协同绘图 | 中 | 在线协同 | Client-server | | 准备第二套技术方案 |
| 代码语法检查 | 低 | 提示语法错误 | 语法分析 | |  |

## 备选方案和竞争

### Swift（ipad版）

优点：

* 有详细、有趣、由浅至深的新手教程，便于上手。
* 代码封装层级高，用户通过输入十余行代码，就能控制角色完成游戏，即时反馈感强。
* 代码简介，语法与自然语言相近，便于幼儿理解。

缺点：

* 使用平台单一，只能在ipad上使用。
* 对设备配置要求高，动画渲染消耗较多的计算资源。

### Scrach：

优点：

* 用户发布后的程序，在官网可以找到，并且在网页内运行的，作品可以通过网络被无数人看到。
* 社区发展得很成熟，便于用户交流学。
* 图形编程，易于理解

缺点：

* 不易表述复杂的逻辑。
* 数组下标从1开始，与主流编程语言不同。

# 产品概述

[此节高度概括产品的功能、与其他应用程序的接口以及系统配置。此节通常要包括以下三个小节：

• 产品总体效果

• 产品功能

• 假设与依赖关系]

## 产品总体效果

[**前景**文档的这一小节应将该产品放在其他相关产品环境和用户环境中进行介绍。如果该产品自成一体，应在此处说明。如果该产品是较大系统的构件，此小节则应说明这些系统如何进行交互，并确定系统之间的相关接口。要显示较大系统的主要构件、互连情况和外部接口，一种简单的方法就是通过框图来表示。]

## 功能摘要

[总结该产品将提供的主要优点和特性。例如，一个客户支持系统的**前景**文档可能会利用此部分来讨论存在问题的记录、消息传递和状态报告，而不必涉及每个功能的细节。

对功能加以组织，使客户或初次阅读该文档的其他人能够理解此功能列表。下面的简表列出了主要优点及支持的特性，该示例应足以说明问题。例如：]

**客户支持系统**

|  |  |
| --- | --- |
| **客户利益** | **支持特性** |
| 新的支持人员能够很快地步入正轨。 | 知识库可协助支持人员迅速地找到已知的解决方法和变通方法。 |
| 因为考虑周全而提高了客户满意度。 | 在整个解决过程中可将问题一一列出，并进行分类和跟踪。一出现老化问题就自动发出通知。 |
| 管理人员能够发现存在问题的领域并估计人员的工作量。 | 趋势及分布报告可从较高的角度来审查问题的状态。 |
| 分散的支持团队能够协同解决问题。 | 复制服务器使当前的数据库信息可以在整个企业的范围内共享 |
| 客户能够自行解决一些问题，从而降低了支持成本并缩短了答复时间。 | 可以通过 Internet 来访问知识库。包括超文本搜索功能和图形查询引擎 |

## 假设与依赖关系

[列出会影响**前景**文档中所述特性的所有因素。列出其变更将引起**前景**文档随之变化的假设。例如，有这样一项假设：将为该软件产品指定的硬件提供特定的操作系统。但如果没有提供该操作系统，就将需要更改**前景**文档。]

## 成本与定价

[对于向外部客户发售的产品和许多内部的应用程序，成本和定价问题会直接影响到应用程序的定义和施。在此节中，应记录任何相关的成本或定价约束。例如，分销成本（软盘的数量、光盘的数量、CD 制作）或其他商品销售的成本约束（手册、包装）可能对于项目的成功非常重要，也可能无关紧要，这取决于应用程序的性质。]

## 许可与安装

[许可和安装问题也可能直接影响到开发工作。例如，如果需要支持串行化、口令安全或网络许可，则会增加在开发工作中必须予以考虑的系统需求。

安装需求还可能会影响到编码，或需要单独安装的软件。]

# 产品特性

[列出并简述产品的特性。特性是为让用户获益而必须具备的高级系统功能。每一项特性都是外部所需的服务，它通常需要一系列输入来实现预期的结果。例如，问题跟踪系统的特性是能够提供趋势报告。当用例模型成型后，更新这里的说明以指代用例。

由于**前景**文档将由各种各样的相关人员来复审，所以不应太过详细，应让所有人对此都有大致的了解。但是，应该向团队提供他们创建用例模型所需的必要详细信息。

要有效地管理应用程序的复杂性，对于任何新系统或对现有系统的增量部分，我们建议将功能提炼到较高的程度，这样 25 到 99 项特性较为合理。这些特性为产品定义、规模管理和项目管理提供了基础。每项特性的详细程度都将在用例模型中得到较深入的扩展。

贯穿此节的始终，都应能让用户、操作人员或其他外部系统从外部觉察到每项特性。这些特性应包括功能性的说明以及必须考虑的任何相关的可用性问题。以下原则将会适用：

• 避免设计。使特性说明保持一定的概括程度。侧重于说明所需的功能以及为什么要（而不是如何） 实现这些功能。

• 如果您使用的是 Requisite 工具包，应将需求类型选择为“所有”，以便于引用和跟踪。]

## <一个特性>

## <另一个特性>

# 约束

[记录所有设计约束、外部约束或其他依赖关系。]

# 质量范围

[定义性能、强壮性、容错、可用性以及特性集内没有记录的类似特征的质量范围。]

# 优先级

[定义不同系统特性的优先级。]

# 其他产品需求

[在较高层次上列出适用的标准、硬件或平台需求、性能需求以及环境需求。]

## 适用的标准

[列出产品必须符合的所有标准。其中可能包括法律和法规（FDA、UCC）标准、通讯标准（TCP/IP、ISDN）、平台一致性标准（Windows、Unix 等）以及质量和安全标准（UL、ISO、CMM）。]

## 系统需求

[确定支持该应用程序所必需的任何系统需求。其中可能包括所支持的主机操作系统及网络平台、配置、内存、外围设备和配套软件。]

## 性能需求

[本节用于详细说明性能需求。性能问题可能包括在各种负载条件下的用户负载因素、带宽或通信容量、吞吐量、精确度以及可靠性或响应时间。]

## 环境需求

[根据需要详细说明环境需求。对于基于硬件的系统，环境因素可以包括温度、振荡、湿度、辐射等。对于软件应用系统，环境因素可以包括使用条件、用户环境、资源可用性、维护问题、错误处理和恢复。]

# 文档需求

[此节说明为支持成功部署应用程序而必须制作的文档。]

## 用户手册

[说明用户手册的目的和内容。讨论预期长度、详细程度，是否需要索引、词汇表、教程与参考手册策略等。还应确定格式和打印约束条件。]

## 联机帮助

[许多应用程序提供了联机帮助系统来协助用户。这些系统的性质对于应用程序开发来说独特的，因为它们综合了编程（如超链接）和技术写作（组织、演示）的各个方面。许多人发现联机帮助系统的开发本身就是一个受益于先期规模管理和计划活动的项目。]

## 安装指南、配置文件、自述文件

[在提供全套的解决方案时，提供包括安装说明和配置指南的文档是非常重要的。此外，自述文件通常也要作为一个标准构件包括在内。自述文件可以包括一个“本发布版中的新特性”部分，并讨论与以前发布版的兼容性问题。多数用户也希望在自述文件中列出任何已知的错误和变通方法。]

## 标签与包装

[目前最先进的应用程序从产品包装开始就提供了一致的外观，这种一致还体现在安装菜单、启动屏幕、帮助系统、GUI 对话框等等。此节定义对标签的需求和和标签类型，以便合并到代码中。举例来说，标签和包装涉及版权和专利声明、公司徽标、标准化的图标以及其他图形元素。]