农田无人机喷洒农药模拟系统
课题需求分析报告

作者：自动化2406吴立锟、迟泰炎

农田无人机喷洒农药模拟系统
课题需求分析报告

作者：吴立锟，迟泰炎

1.引言

1.1编写目的

通过“无人机喷洒农药模拟系统”，我们可以通过图形化的页面让用户自由绘制自己的农田，或者将农田分块进行处理，来达到更加灵活的规划，在此同时我们支持不同的无人机型号，也具备各种灵活的规划，如采用A*算法，TSP算法等对不同的路径进行规划，针对这一项目，我们编写了这项报告，希望从项目概述，功能分析，代码实现以及一些核心算法解释这些方面来进行解答，旨在让用户通过阅读本篇报告来了解整个项目的原理和尽快熟练使用整个程序。

1.2项目背景

在当下科技的逐步发展，人们已经不满足于先前的机械化农业，即使用各种机械化设备配合人力进行工作，这样并没有真正的解放劳动力，再加上农村的人员外流，以及当下人们的愿景逐渐趋于可以采取一种完全解放劳动力的种植方式，这推动了自动化行业的发展，将自动化技术和智能技术应用于种植当中，可以智能监测农田的生长情况，并且针对不同的病害进行处理，同时它可以克服地形和气候对人的影响，大大提高劳作的灵活程度，从而提高产量。但并非所有农民都明白如何去对无人机进行喷洒规划，也不明白怎样的飞行是效率最高，工作时间最短，得到的产率最大的。

基于这些问题，编者编写了这个农田无人机喷洒农药模拟系统，旨在帮助更多的农民用户在图形化的界面了解自己农田可能出现的问题，了解如何对虫害进行更快地处理，从而提高产量。我们欢迎所有希望采用全自动技术进行劳作的劳动者们，希望通过此项技术可以模拟出真实环境下无人机喷洒的实际演示，来让劳动者们得到更高效的生产方式，便宜自身。

1.3编者的话

本农田无人机喷洒农药模拟系统课题早早就在我们的设计预案当中，主要原因是它涉及的路径规划所展现出的灵活程度让我们及其心仪，并且我们本身的专业是在人工智能与自动化学院中的自动化专业，无人机更加符合心意，并且当下正处于脱贫攻坚的持续阶段，很多土地的生产并不如人意，经过分析，我们发现其原因主要有以下几点：一是人的生产力是有限的并且是受很多环境因素制约的，二是虫害发现的不够及时，很多情况下是产生了坏的影响才被发现。所以我们选择了这个课题，希望用模拟的方式辅助开发无人机，希望用自己的专业知识并且结合一些常用算法对农民生产进行指导，并且最终在未来可以实现实际的无人机开发，真正投入生产的使用。

2.项目概述

随着农业科技的快速发展，无人机在农田管理中扮演着越来越重要的角色。特别是在精确农业领域，无人机可以用于高效、精确地喷洒农药，以保护作物免受病虫害的影响，同时减少农药的使用量和对环境的影响。为了支持这项技术的进步，本项目的目标是开发一套农田无人机喷洒农药模拟系统，通过使用bc编译器，创建一个模拟环境来测试和优化无人机喷洒策略。

2.1项目目的

1.模拟系统的开发： 利用bc编译器开发一套模拟系统，该系统能够模拟无人机在农田中喷洒农药的行为，包括飞行路线规划、喷洒覆盖率以及农药使用效率等。

2.优化喷洒策略： 通过模拟实验，分析不同的喷洒参数（如喷洒面积，载水量）和不同的飞行路径对喷洒效率和覆盖率的影响，从而确定最佳的喷洒策略。

2.2项目范围

1.bc编译器的应用： 研究和应用bc编译器在农业模拟系统中的使用，特别是在数值计算和算法逻辑方面的优化。

2.无人机操作模拟： 创建和优化无人机的飞行和操作模拟算法，确保模拟结果符合真实世界操作的精确性。

3.环境因素考虑： 在模拟系统中纳入多种环境因素，如气象条件、地形变化等，确保模拟环境的全面性和实用性。

4.用户界面开发： 设计直观的用户界面，使操作者能够轻松设定参数、运行模拟和解析结果

通过本项目的实施，预期能够极大地提高农田无人机喷洒农药的科学性和有效性，为实现更精确、高效和环保的农业生产做出贡献。

2.3编写规范

2.3.1命名规范：

1.变量命名，涉及用户以及股票信息的，应该尽量用英文表达其准确定义。其他类型变量名应给出详细注释以说明其主要功能。

2.函数命名应该用英文表达其确定含义

3.文件命名都用小写，并且表达出该文件所包含函数的主要功能。

4.涉及数据结构的命名应参考数据结构（C语言版），并进行适当修改。比如说沪深界面使用队列命名时可以用stoqueue。

2.3.2 注释:

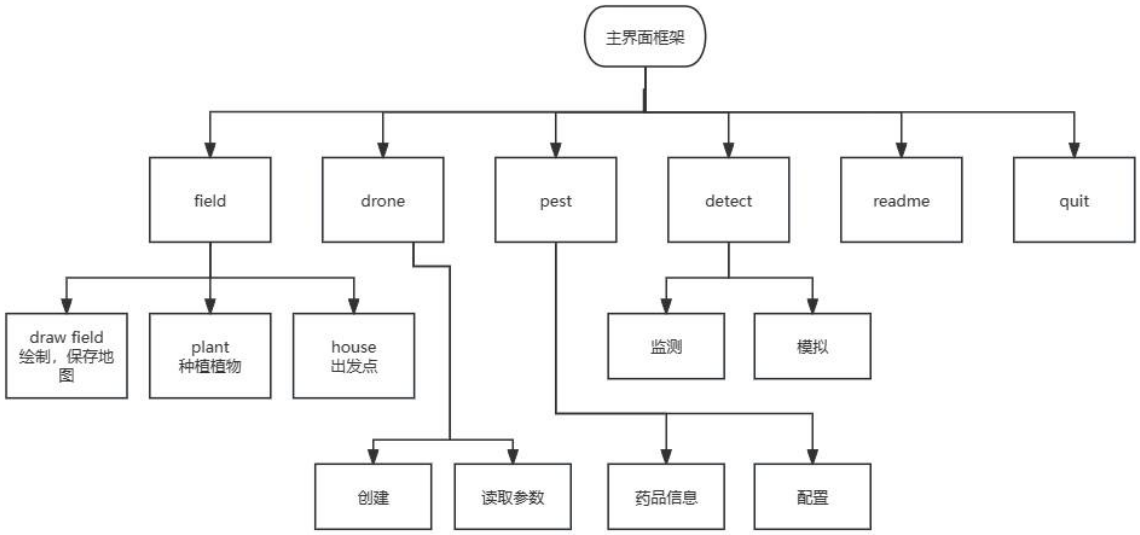
1.函数功能都要在函数原型后注明。

2.部分令测试者比较难以理解的算法和流程应该给出相应的注释。

3.功能需求分析

3.1系统概述

本系统旨在提供一个用于模拟无人机在农田进行农田喷洒操作的虚拟环境。系统将使用户能够准确模拟和调整无人机的飞行路径、喷洒参数和考虑环境等因素，从而优化策略和提高农药的使用效率。在此同时，本系统提高界面使得用户可以自定义农田图样，和无人机的各项参数，从而达到更加灵活的功能调试。



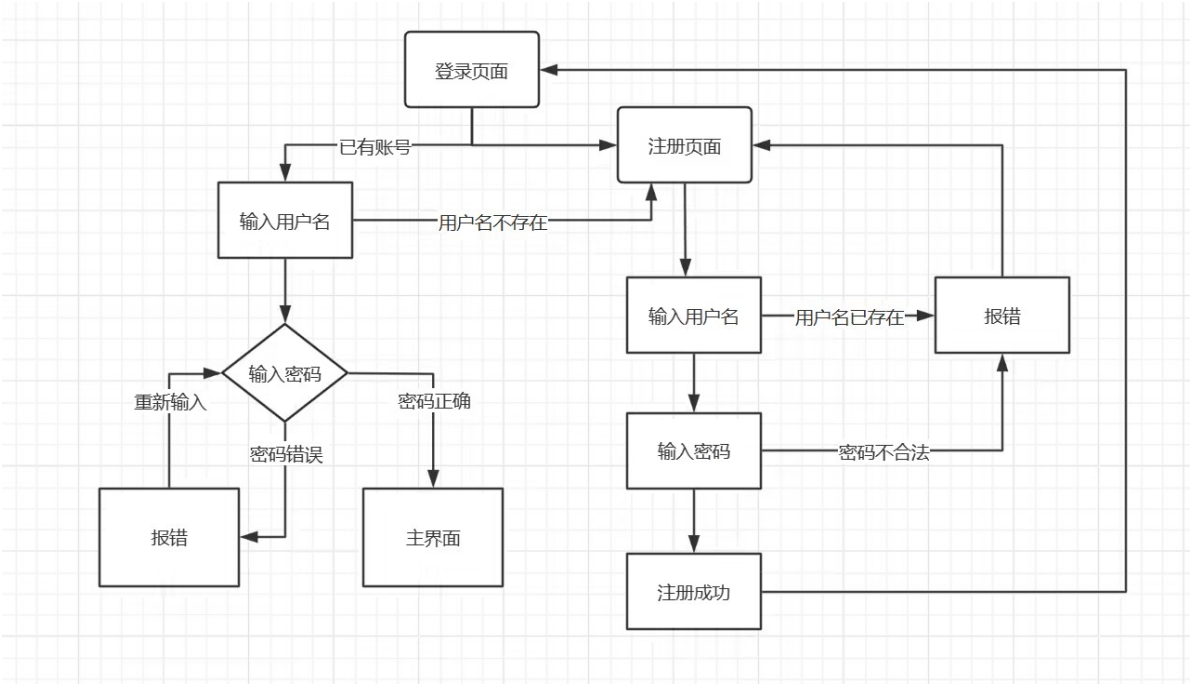
3.2功能需求分析

3.2.1 登录和注册页面

1.登录页面功能：在登录时需要检测用户的用户名和密码是否正确，这就需要从在注册页面中写入的账号密码存储的文件中读取并且一一比较，来达到正确进入用户的账号当中，以防发生账号信息的错误传递。

2.注册页面功能：在注册时需要检测用户注册的用户名和密码是否合法，比如必须包括大小写和位数限制之类的，在检测完之后需将其写入文档中保存，供给登录界面以及后续的信息读入和录入使用。

3.文本框输入功能：在图形化界面中无法智能地输入文字，这时候需要使用一个函数，对键盘传递出的数字信号进行读取，从而智能地实时输入文字，并且鼠标的光标可以随之移动。

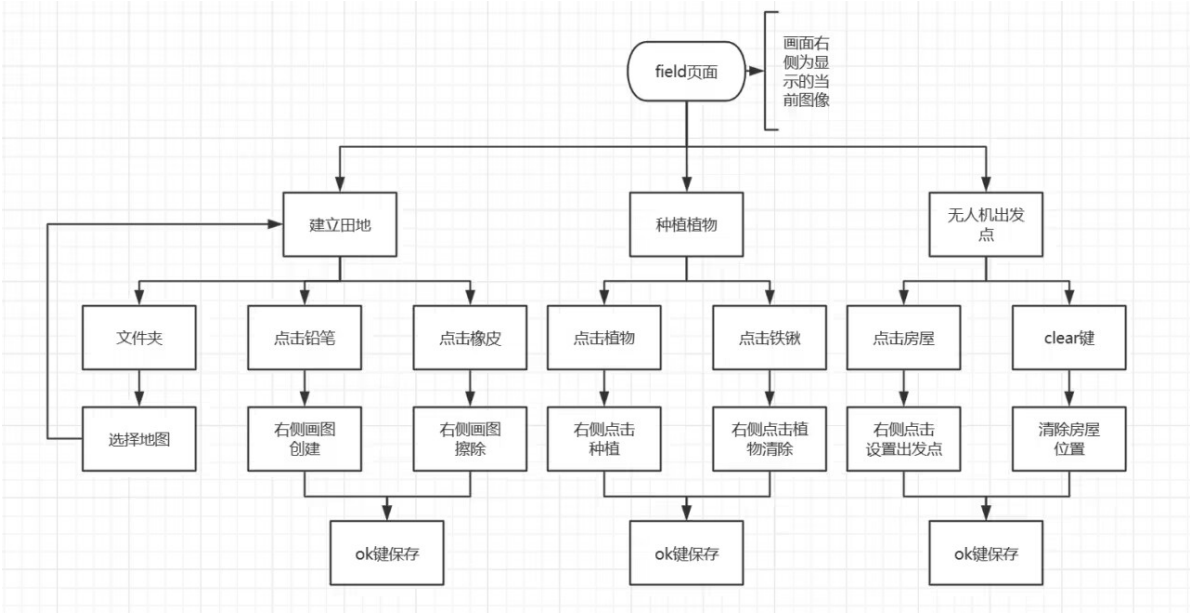


3.2.2 创建农田页面

1.农田绘制功能：在进行无人机农田喷洒模拟的时候，农田是必不可少的，所以我们写了一个农田绘制的功能，通过采用像素点集成的手段将其的单位像素扩大，从而抽象成一个几何规整的图样，便于进行模拟，在这需求下，农田绘制功能诞生了，它有绘制、清楚、保存三个功能，并且有坐标格，可以实时观看自己的农田的样貌，并且每次打开的时候都可以读取之前的农田绘制数据，不需要每次都重新绘制。

2.庄稼和无人机起点位置标注功能：在进行无人机模拟喷洒的时候，庄稼位置和无人机起点位置的信息是必不可少的，前者是我们要检测虫害和喷洒的对象，后者涉及到我们最终的路径规划和最高效路径的选取，所以我们想设计一个功能便于让用户自由标准自己的农田上的庄稼和无人机位置，从而传递到最终的模拟页面，使得其更加灵活多样。

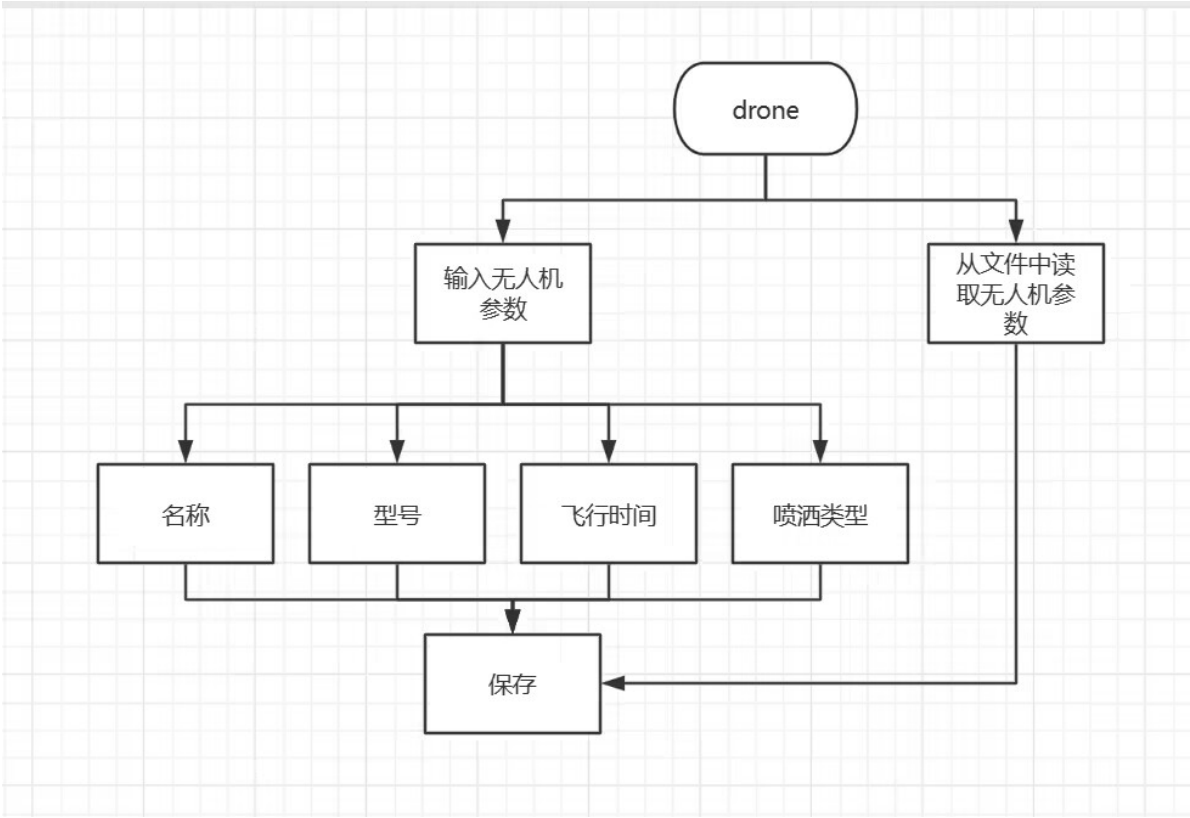
3.农田储存功能：由于最终目标是模拟无人机农田喷洒，农田的传递是必不可少的，所以我写了一个文件夹的功能，它可以遍历我们的储存数据，用户可以在自己名下的多个农田中选取任意农田，从而模拟出农民有多个农田想要同时模拟，或者将一个农田分割成不同的农田部分进行模拟的想法。并且将农田储存下来可以在模拟农田部分打开并且读取其中的数据，来识别农田上的庄稼位置，无人机起点位置，和一些房屋障碍位置。



3.2.3 无人机信息录入页面

1.创建新的无人机功能：由于在实际生产中，无人机型号并不是固定的，每个人的喜好不同，无人机本身的效率和适用的农田类型也不同，所以这个功能必不可少，我们系统本身会提供几种常见无人机，用户也可以自己创建自己手上拥有的无人机类型，主要包括型号、飞行时间和喷嘴大小，这个功能的设计会直接影响到最终模拟农田喷洒的结果，使得结果更为多样，同时也让用户体验到自己真实种植的环境和模拟结果。

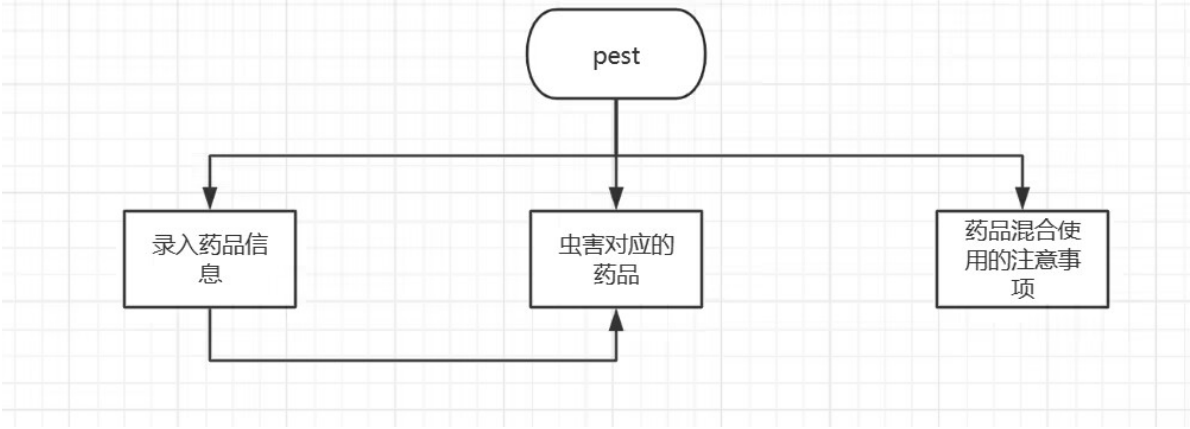
2.储存和读取无人机功能：由于无人机有多种，对其进行数据管理显得尤为重要，这个功能主要提供无人机的信息的储存和读取，我们预将其储存在自己名字的文件夹下，在需要使用的时候遍历自身的文件夹，从而可以选取自己名下的任意一台无人机并且对其进行数据的修改。



3.2.4 农药配置页面

1.农药类型录入功能：由于本项目的最终目的是对农田进行喷洒农药杀虫，并且不同农田不同生长时期的虫害都有所不同，所以农药也需要准备多种，这个功能则是将用户持有的不同农药录入系统，便于在后续模拟的同时调用不同的农药来针对不同的虫害，使得模拟更为真实，更有针对性，更加灵活。

2.农田混合比例录入功能：在很多情况下农作物并非只是患有一种虫害病，很多情况下是多种虫害同时作用，但就化学原理来看很多药物本身会产生反应，会导致农田因化学反应减产等等效应，所以用户可以在这个功能中备注比例的注意事项以及哪些药物不能相互作用，便于后续如何选择药品。

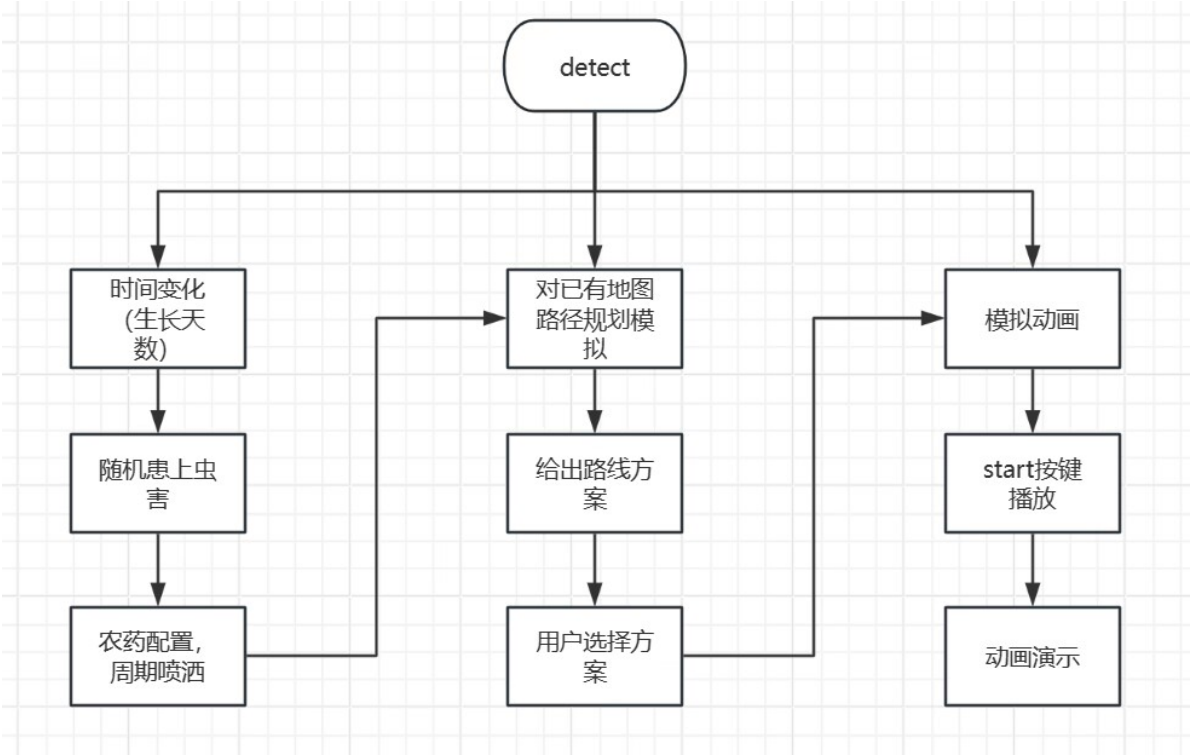


3.2.5 农田无人机喷洒模拟页面

1.实时模拟功能：在实际生产中，农作物的成长是连续的，并且不同时期所拥有的虫害是不一样的，所以我们准备将最终的喷洒演示设计成为动态演变的，主要目的是展示面对不同时期的动态变化的时候能够根据实际情况进行不同的路径规划，以此来达到每一次喷洒都是路径最优化的。

2.多条路径演示功能：由于主要生产中人的主观性占据主要地位，所以我们准备了多条路径的演示，以此来供给用户自由选择心仪的路径方向，来满足其个人需求。

3.动态演示功能：为了模拟无人机的实际飞行，我们设计了动态演示功能，将无人机的每一次飞行的位点都标注下来，从而达到实时观看演示，从中可以实时发现问题，并且可以随时解决。



3.2.6 其他功能

1.报错提醒功能：在一般的程序，存在一些问题的出现，但其结果是不影响代码的继续运行的，所以我们设计了报错提醒功能，用于提醒用户及时停止不恰当的操作，引导他们正常使用整个程序。并且将报错输出到日志当中，从而可以根据每次运行的结果更好的优化程序。

2.鼠标功能：采用鼠标库程序，我们采用手动拓充的方法，在绘制的时候将鼠标替换成各种图标，从而实现更加美观更加顺畅的鼠标移动和作用功能。

4.系统需求分析

4.1硬件接口

处理器：Intel Pentium 166 MX 或以上。
硬盘：空间500MB以上。
屏幕适配器：VGA接口。
系统运行内存：要求32MB以上。

4.2软件接口

开发软件工具：Borland C 3.0。
文字编辑工具：Visual Studio Code、Visual Studio 2022 Community
数据库：文本存储（记事本）或MySQL。
操作系统：DOS WINDOWS 9X/ME/2000/XP/WINDOWS 7。

4.3控制

该系统通过鼠标与键盘直接进行控制。用户将鼠标移至需要操作的功能区进行点击，通过键盘来完成登陆、注册的输入功能。操作完毕后下拉菜单点击相应位置退出系统。通过中断技术来获取鼠标的位置与键盘的输入功能。

5.软件代码需求分析

5.1函数说明·

public.c

```
/* **************
function:void printline(int x,int y,int len,int n,int flag,int COLOR,int wid,int
gap);
description:画像素风线函数
input:int x,int y,int len,int n,int flag,int COLOR,int wid,int gap
      x,y是第一个小方块左上角起始点，flag=0横着向右，flag=1竖着向下，len记录每个块有多长，n
记录有多少格
      wid每格宽度，每格长度为len倍的宽度,gap为每格之间间隙
output:void
************************ */
```

```
 /* ************************
 function:void printbox(int x1,int y1,int x2,int y2,int COLOR,int len,int wid,int
 gap);
 description: 绘制一个边缘由小方块组成的矩形
 input: int x1, int y1, int x2, int y2, int COLOR, int len, int wid, int gap
        x1,y1是矩形左上角点，x2,y2是矩形右下角点
        COLOR定义了方块的颜色
        len指定每个方块的长度，也就是几个宽度的倍数
        wid是每个小方块的宽度
        gap是小方块之间的间隙
 output: void
 ************************* */
```

```
/* ************************
function:void back_button(int flag);
description: 根据不同的状态绘制或删除一个返回按钮
input: int flag
       flag定义了按钮的状态，可以是PAINT, LIGHT, RECOVER, DELETE中的一个
       PAINT状态下绘制一个的按钮
       LIGHT状态下绘制一个的按钮
       RECOVER状态下将按钮恢复到PAINT状态
       DELETE状态下删除按钮
output: void
       如果flag参数不是有效值，将打印错误信息并退出程序
************************* */
```

· welcome.c

```
/************************
function: void welcome_screen(void);
description: 绘制欢迎屏幕，包括登录和退出按钮
input: none
output: void
************************/
```

```
/*************************
function: void put_title(int *color);
description: 在屏幕上输出标题文本，并周期性地改变标题颜色
input: int *color - 指向当前颜色编号的指针，颜色编号随着时间周期性改变
output: void
notes: 颜色编号会在每次调用该函数时递增，当增加到16时重置回1。这种颜色变换可以吸引用户注意。
*************************/
```

```
/*************************
function: int welcome_page(void);
description: 显示欢迎页面，并处理用户的交互，通过点击事件转换到不同的功能模块
input: none
output: int - 根据用户的点击事件返回对应的功能代码
*************************/
```

· signup.c

```
/*************************
function: void signup_bkpaint(void);
description: 绘制注册页面的背景和基本界面元素，包括文本输入框和按钮
input: none
output: void
*************************/
```

```
/*************************
function: int signup_page(void);
description: 处理注册页面的逻辑，包括用户输入处理、按钮点击响应等
input: none
output: int - 根据用户操作返回不同的代码，如转到登录页面
notes: 使用一个循环来不断检测鼠标点击事件，并根据点击位置调用相应的处理函数。这包括文本输入、按钮点击反馈和页面跳转等逻辑。
*************************/
```

· quit.c

```
/*************************
function: void quit_page(void);
description: 在屏幕上逐渐显示“THANK YOU”字样，颜色逐渐变化，作为程序退出前的感谢页面
input: none
output: void
notes: 通过一个循环逐个显示字符串数组中的字符串，每次显示后延迟一段时间，然后改变颜色并清屏，从而创建一个动态的视觉效果。当显示完整个感谢信息后关闭图形界面。
*************************/
```

· login.c

```
/* ************************************************************
function: void login_bkpaint(void)
description: 绘制登录页面的背景，包括文本和图形界面元素
input: none
output: void
************************************************************ */
```

```
/* ************************************************************
function: int login_page(INFO *temp)
description: 控制登录页面的主逻辑，处理用户输入和按钮事件。该函数主要功能包括初始化用户信息结构体，
             绘制登录界面，根据鼠标点击事件更新界面（如点亮按钮）、处理用户信息输入（用户名和密码）、
             验证输入的用户名和密码是否正确，以及根据用户交互跳转到相应的页面（如注册页面、欢迎页面或主页）。
             具体操作包括监听鼠标点击事件，判断鼠标位置以确定用户点击的按钮或输入框，并执行相应的逻辑。
input: INFO *temp - 指向用户信息的指针，用于存储登录后的用户信息，以便传递给其他页面。
output: int - 返回不同的整数值以指示后续操作，例如跳转至注册页面、返回欢迎页面或进入主页，具体操作取决于用户的交互和输入验证结果。
************************************************************ */
```

· logfunc.c

```
/* ************************************************************
function: int check(INFO *user)
description: 验证用户登录。首先检查指定路径下是否存在以用户名命名的文件夹，
             若不存在，提示用户未注册。若存在，尝试打开包含用户信息的文件，
             读取并比对密码。如果密码正确，返回1表示登录成功，否则提示密码错误
             或无法读取用户信息文件，并根据情况返回相应的错误代码或终止程序。
input: INFO *user - 包含要验证的用户名和密码的结构体指针
output: int - 返回1表示验证成功，返回0表示验证失败或指定用户不存在
************************************************************ */
```

```
/* ************************************************************
function: void temp_input(char *temp, int x, int y, int maxi, int w, int h, int COLOR, int size)
description: 在指定位置和大小的框内接收用户输入的文本。该函数处理键盘输入，
             包括普通字符和特殊键（如箭头键、回车、退格等），并在屏幕上实时显示
             输入的文本以及光标位置。支持光标移动，可以在文本中间插入字符。
             当用户输入完成（通过按下空格、换行、回车或ESC键）时，函数结束。
input: char *temp - 存储用户输入的字符串
       int x, y - 文本框的起始坐标
       int maxi - 最大字符数限制
       int w, h - 文本框的宽度和高度
       int COLOR - 文本框背景颜色
       int size - 字体大小
output: 无，但 *temp 会被更新为用户输入的文本
************************************************************ */
```

```
/* ************************************************************
function: void password_warning(char *s)
description: 显示密码警告消息。此函数用于在用户输入密码时，如果密码不符合要求（例如太短或包含非法字符），
             在屏幕特定位置显示警告消息。首先，它会清理鼠标轨迹，然后清空原有的密码输入区域，
             接着在指定位置以特定字体和颜色显示警告消息（如“too short!”或“illegal!”）。
             消息显示2秒后，再次清空密码输入框，以便用户重新输入。
input: char *s - 要显示的警告消息字符串
output: void - 无返回值，但会在屏幕上更改内容
************************************************************ */
```

```
/* ************************************************************
function: void title_warning(char *s, int PAGE)
description: 显示标题警告或提示消息。此函数用于在用户界面的标题位置显示特定的警告或提示消息。
             根据提供的页面标识（例如注册或登录页面），它会清除之前的标题信息，并显示新的提示信息。
             显示的消息因传入的PAGE值而异。消息显示2秒后，根据PAGE值刷新标题显示内容。
             这有助于引导用户了解他们当前应执行的操作（如注册或登录）。
input: char *s - 要显示的提示或警告消息字符串
       int PAGE - 页面标识符，决定了之后显示的默认提示信息类型
output: void - 无返回值，但会在屏幕上更改内容
************************************************************ */
```

```
/* ************************************************************
function: int password_check(const char *password)
description: 验证密码的复杂度。这个函数检查密码是否满足特定的安全要求，包括长度大于6个字符，
             至少包含一个大写字母、一个小写字母和一个数字。如果密码不满足这些要求，函数将返回一个特定的错误代码。
input: const char *password - 用户输入的密码字符串。
output: int - 返回密码状态码。0表示密码合法；1表示密码长度不足；2表示密码包含非法字符（既不是字母也不是数字）；
              3表示密码缺少必要的字符种类（大写字母、小写字母或数字中至少一种未出现）。
************************************************************ */
```

```
/* ************************************************************
function: int user_exist_check(const char *username)
description: 检查指定用户名的用户是否已经存在。通过检查系统中是否存在与用户名相对应的文件夹来确定用户是否存在。
             如果文件夹存在，意味着用户名已被占用。
input: const char *username - 待检查的用户名。
output: int - 如果用户名对应的文件夹存在返回1，否则返回0。
************************************************************ */
```

```
/* ************************************************************
function: void creat_user_directory(INFO *user)
description: 为新用户创建一个目录。该函数根据用户的用户名，在指定的根目录下创建一个新的文件夹。
             如果文件夹创建失败，函数将报告错误并退出程序。
input: INFO *user - 指向包含用户信息的结构体指针，其中包括用户名。
output: void - 无返回值，但如果创建文件夹时出错会通过 perror 显示错误信息，并退出程序。
************************************************************ */
```

```
/* ************************************************************
function: void creat_userinfo_file(INFO *user)
description: 在用户的目录中创建一个包含用户信息的文件。函数首先检查用户目录是否存在，如果不存在，则报告错误。
             如果目录存在，函数将在该目录下创建一个名为 "info" 的文件，并将用户的信息写入该文件。
input: INFO *user - 指向包含用户信息的结构体指针，其中包括用户名等信息。
output: void - 无返回值，但如果遇到错误会通过 perror 显示错误信息，并根据错误类型返回或关闭文件。
************************************************************ */
```

· home.c

```
/* ************************
function: void home_screen(void);
description: 绘制主页的图形界面。该界面包含六个功能区域（FIELD, DRONE, PEST, DETECT, README, QUIT）和一个退出按钮。每个功能区域都通过特定坐标的矩形表示，并在每个矩形内部显示相应的功能名称。此外，顶部中央显示“HOMEPAGE”标题。
input: void
output: void
************************* */
```

```
/* ************************
function: int home_page(INFO *temp);
description: 处理主页的逻辑和用户交互。该函数首先初始化图形界面和鼠标，然后循环等待用户的鼠标点击事件。根据用户点击的位置，函数决定用户选择的操作（例如FIELD, DRONE等）。如果用户点击一个功能区域，该区域会被高亮显示并根据点击类型（单击或双击）返回相应的操作常量值。该函数还处理退出按钮的点击事件，允许用户退出到登录页面。特定的视觉反馈（如按钮高亮和恢复正常显示）通过调用相关函数实现，以提升用户体验。
input: INFO *temp - 指向INFO结构的指针，用于存储或访问与主页状态相关的信息。本函数中未直接使用，可用于扩展功能。
output: int - 返回一个整数值，表示用户选择的操作。不同的返回值对应不同的操作（如FIELD, DRONE等）。
************************* */
```

· field.c

```
/* ************************
function:void field_screen();
description: 初始化并绘制场景界面
初始化并绘制场景界面，包含绘制标题栏、边框、网格线，以及农田、农作物和无人机位置的按钮。绘制的元素包括：
- 使用printline函数绘制顶部的线条
- 设置文本颜色和样式，并在指定位置输出“NAME:”标签和“SQUARE”
- 绘制一个包含X和Y轴坐标的大矩形边框，并在角落处绘制箭头
- 使用循环创建一个网格背景，通过竖线和横线的组合实现
- 调用back_button函数绘制返回按钮
调用put_field、put_sprout和put_house函数分别绘制农田、农作物和无人机位置的按钮
input: 无
output: void
************************* */
```

```
/* ************************
function: int field_page(INFO *temp);
description: 处理场景界面中的用户交互
该函数为场景界面提供交互功能，处理用户在界面上的各种鼠标点击事件。功能包括：

- 清除屏幕并设置背景色
- 调用field_screen函数以绘制界面
- 初始化鼠标并进入一个循环，不断检查鼠标状态和位置
- 根据鼠标位置和按钮区域的交互效果，决定是否改变按钮颜色或执行相应的动作
- 根据用户点击不同的按钮，返回不同的值以触发不同的操作，如绘制农田、放置农作物或无人机

当用户点击返回键时，清除屏幕并返回首页
input: INFO *temp - 指向INFO结构体的指针，用于存储和传递用户界面状态信息
output: int - 返回一个整数值来表示用户选择的不同操作或状态
************************* */
```

· house.c

```
/* ************************************************************
function: void house_screen(int record[21][26], char *nowfield);
description: 该函数用于清屏并重新绘制当前场景的屏幕。它首先清除屏幕上的所有内容，
             然后根据传入的记录数组和当前场景名称绘制新的屏幕。这包括绘制场景的背景、
             任何静态和动态对象以及鼠标光标。此函数是响应用户交互（如移动、点击等）
             和游戏状态变化时更新视觉呈现的核心。
input:
    - int record[21][26]: 二维数组，记录了房屋位置和状态信息。每个元素代表场景中的一个单元格，
                          数值表示不同的元素或物体状态。
    - char *nowfield: 当前场景的名称，一个字符串指针，指向表示当前地图的字符串。
output: void
************************************************************ */
```

```
/* ************************************************************
function: void clear_button(int flag);
description: 该函数负责根据传入的标志绘制清空按钮，并根据用户交互更改其状态。
             它可以绘制普通状态的按钮、当鼠标悬停时的高亮状态、以及响应点击的动作。
             根据传入的标志，该函数会选择性地绘制按钮的不同视觉状态，或者将按钮从屏幕上删除。
             这允许用户通过图形界面清空或重置特定的输入或场景元素，提高用户交云度和体验。
input:
    - int flag: 一个整数标志，用于指示按钮的绘制状态。不同的值对应不同的行为，例如绘制按钮、
                高亮显示、恢复默认状态，或者完全删除按钮。
output: void
************************************************************ */
```

```
/* ************************************************************
function: int house_page(char *username, char *nowfield);
description: 该函数是处理房屋场景中所有主要逻辑的函数，包括事件监听、场景绘制和状态管理。
             它根据用户的交互来更新场景状态，如移动、选择不同的房间等，并负责将这些变化反应到屏幕上。
             函数使用用户名来加载和保存用户特定的场景数据，确保每次用户回到游戏时都能恢复到他们离开时的状态。
             根据不同的用户操作和游戏逻辑，该函数可能返回不同的状态码，指示应用程序应采取的后续行动，
```

```
                如跳转到其他页面或显示特定的对话框。
input:
- char *username: 用户名，字符串指针，表示当前用户。这个信息用于个性化用户的游戏体验，
  包括加载和保存用户特定的场景设置和进度。
  - char *nowfield: 当前场景的名称，字符串指针，指向表示当前用户选择的房屋场景的字符串。
    这个信息用于确定哪些数据需要被加载和保存，以及如何绘制当前场景。
output:
    int: 函数返回一个整数值，通常作为状态码使用，表明函数执行的结果，或者是基于用户操作需要导向
的下一步动作。
************************************************************* */
```

· plant.c

```
/* ************************************************************
function: void paint_field(int record[21][26], char *nowfield);
description: 该函数用于绘制场景的画布和场景中的元素。它清除屏幕并绘制背景图以及根据记录数组
             绘制场景中的不同元素，如房屋、树苗等。此外，它也会绘制侧边栏和返回按钮，展示当前
场景
             的信息和用户可以进行的操作。
input:
    - int record[21][26]: 二维数组，记录了场景中元素的位置和状态。每个元素代表场景中的一个单
元格，
                          数值表示不同的元素或物体状态。
    - char *nowfield: 当前场景的名称，一个字符串指针，指向表示当前地图的字符串。
output: void
************************************************************* */
```

```
/* ************************************************************
function: void plant_screen(int record[21][26], char *nowfield);
description: 该函数用于清屏并重新绘制植物场景的屏幕。它首先清除屏幕上的所有内容，
             然后根据传入的记录数组和当前场景名称绘制新的屏幕。在右侧点击时可以种下植物，即点
击后在土地上出现植物幼苗的图标，并且在按下ok确认后将地图储存下来。按下铲子按钮后可以在右侧有植物
的地方将植物铲除，通过重绘地图的方式实现植物图标的清除，并且也在按下确认按键后将新的地图状态储存
下来（在用户名的文件夹下的field文件夹里面）
input:
- int record[21][26]: 二维数组，记录了植物位置和状态信息。每个元素代表场景中的一个单元格，
  数值表示不同的元素或物体状态。
char *nowfield: 当前场景的名称，一个字符串指针，指向表示当前场景的字符串。
output: void
************************************************************* */
```

```
/* ************************************************************
function: int plant_page(char *username, char *nowfield);
description: 该函数是处理植物场景中所有主要逻辑的函数，包括事件监听、场景绘制和状态管理。
             它根据用户的交互来更新场景状态，如种植树苗、使用铲子等，并负责将这些变化反应到屏幕上。
             函数使用用户名来加载和保存用户特定的场景数据，确保每次用户回到游戏时都能恢复到他们离开时的状态。
             根据不同的用户操作和游戏逻辑，该函数可能返回不同的状态码，指示应用程序应采取的后续行动，
             如跳转到其他页面或显示特定的对话框。
input:
- char *username: 用户名，字符串指针，表示当前用户。
  - char *nowfield: 当前场景的名称，字符串指针，指向表示当前用户选择的地图场景的字符串。
output:
    int: 函数返回一个整数值，通常作为状态码使用，表明基于用户操作需要导向的下一步动作。
************************************************************ */
```

· fieldfunc.c

```
/* ************************
function: void draw_field_screen();
description: 绘制绘图界面的主屏幕
- 调用back_button函数绘制返回按钮。
- 使用printline函数绘制顶部的灰色线条作为标题栏的背景。
- 设置文字颜色和样式，输出标题"NAME:"。
- 绘制一个包含X和Y轴的大矩形图框，并在其两端绘制箭头以指示坐标轴方向。
- 使用循环绘制内部的网格线，用于界定绘图区域的大小和范围。
调用put_pencil、put_rubber和put_file函数分别绘制铅笔、橡皮和文件图标作为操作按钮。
input: 无
output: void
************************* */
```

```
/* ************************
function: void open_file();
description: 执行打开文件操作的视觉反馈
该函数负责打开文件时的视觉反馈，其主要功能和步骤包括：
- 清除鼠标光标。
- 设置填充样式和颜色，绘制一个背景为大矩形，作为文件内容显示的背景。
- 使用printline函数绘制的边框和内部的分隔线，为文件内容的组织提供视觉结构。-
设置文字样式，输出提示信息"CREATE A FIELD."。
调用put_arrow函数绘制向左和向右的箭头按钮，用于翻阅文件内容或其他交云操作。
input: 无
output: void
************************* */
```

```
/* ************************
function: int draw_field_page(char *name,char *now_field);
description: 绘制场景界面的主要操作和逻辑处理
该函数用于绘制场景界面，并处理用户在界面上的各种交互操作。主要功能和步骤包括：
- 定义和初始化所需的变量和数据结构。
- 创建和初始化农田文件夹，并进行错误处理。
- 清除屏幕并绘制场景界面。
- 进入一个循环，不断检测用户的鼠标事件。
```

```
- 根据鼠标位置和按钮区域的交互效果，决定是否改变按钮颜色或执行相应的动作。
- 根据用户的点击不同按钮，执行相应的操作，如绘制、擦除或打开农田文件。
- 当用户点击返回键时，清除屏幕并返回上一个菜单。
input:
- char *name - 用户名字，用于创建农田文件夹的路径。
- char *now_field - 当前选择的农田文件名。
output: int - 返回一个整数值来表示用户选择的不同操作或状态。
************************* */
```

· plant.c

```
/* ************************************************************
function: void paint_field(int record[21][26], char *nowfield);
description: 该函数用于绘制场景的画布和场景中的元素。它清除屏幕并绘制背景图以及根据记录数组
             绘制场景中的不同元素。此外，它也会绘制侧边栏和返回按钮，展示当前场景的信息和用户
             可以进行的操作。
input:
    - int record[21][26]: 二维数组，记录了场景中元素的位置和状态。每个元素代表场景中的一个单
元格，
                          数值表示不同的元素或物体状态。
    - char *nowfield: 当前场景的名称，一个字符串指针，指向表示当前场景的字符串。这个名字用于
                      绘制场景特定的资源文件，如背景图、物体图案等，以便正确绘制当前场景。
output: void
************************************************************ */
```

```
/* ************************************************************
function: void plant_screen(int record[21][26], char *nowfield);
description: 该函数用于清屏并重新绘制植物场景的屏幕。它首先清除屏幕上的所有内容，
             然后根据传入的记录数组和当前场景名称绘制新的屏幕。这包括绘制场景的背景、
             任何静态和动态对象以及鼠标光标。此函数是响应用户交互（如移动、点击等）
             和游戏状态变化时更新视觉呈现的核心。
input:
    - int record[21][26]: 二维数组，记录了植物位置和状态信息。每个元素代表场景中的一个单元
格，
                          数值表示不同的元素或物体状态。
    - char *nowfield: 当前场景的名称，一个字符串指针，指向表示当前场景的字符串。这个名字用于
                      定位场景特定的资源文件，如背景图、物体图案等，以便正确绘制当前场景。
output: void
************************************************************ */
```

```
/* ************************************************************
function: int plant_page(char *username, char *nowfield);
description: 该函数是处理植物场景中所有主要逻辑的函数，包括事件监听、场景绘制和状态管理。
             它根据用户的交互来更新场景状态，如种植树苗、使用铲子等，并负责将这些变化反应到屏
幕上。
             函数使用用户名来加载和保存用户特定的场景数据，确保每次用户回到游戏时都能恢复到他
们离开时的状态。
             根据不同的用户操作和游戏逻辑，该函数可能返回不同的状态码，指示应用程序应采取的后
续行动，
             如跳转到其他页面或显示特定的对话框。
input:
   - char *username: 用户名，字符串指针，表示当前用户。这个信息用于个性化用户的游戏体验，
```

```
      包括加载和保存用户特定的场景设置和进度。
   -  char *nowfield: 当前场景的名称，字符串指针，指向表示当前用户选择的植物场景的字符串。
                      这个信息用于确定哪些数据需要被加载和保存，以及如何绘制当前场景。
output:
     int: 函数返回一个整数值，通常作为状态码使用，表明基于用户操作需要导向的下一步动作。
************************************************************* */
```

· detect.c

```
/* *************************************************************
function: void detect_screen(int record[21][26], char *nowfield);
description: 该函数用于清屏并重新绘制检测场景的屏幕。它首先清除屏幕上的所有内容，
             然后根据传入的记录数组和当前场景名称绘制新的屏幕。这包括绘制场景的背景、
             任何静态和动态对象以及鼠标光标。此函数是响应用户交互（如移动、点击等）
             和游戏状态变化时更新视觉呈现的核心。
input:
    - int record[21][26]: 二维数组，记录了检测场景中元素的位置和状态。每个元素代表场景中的一个单元格，
                          数值表示不同的元素或物体状态。
    - char *nowfield: 当前场景的名称，一个字符串指针，指向表示当前场景的字符串。这个名字用于
                      定位场景特定的资源文件，以便正确绘制当前场景。
output: void
************************************************************* */
```

```
/* *************************************************************
function: void start_button(int flag);
description: 根据不同的状态绘制开始按钮，并可能响应点击事件
input:
    - int flag: 控制按钮的状态，包括绘制、高亮、恢复默认和删除
output: void
************************************************************* */
```

```
/* *************************************************************
function: int detect_page(char *username, char *nowfield);
description: 处理检测场景中所有主要逻辑的函数，包括事件监听、绘制和状态管理。
             根据用户的交互来更新场景状态，如启动检测、暂停检测等，并将这些变化反应到屏幕上。
             函数使用用户名来加载和保存用户特定的场景数据，确保每次用户回到游戏时都能恢复到他们离开时的状态。
             根据不同的用户操作和游戏逻辑，该函数可能返回不同的状态码，指示应用程序应采取的后续行动，
             如跳转到其他页面或显示特定的对话框。
input:
    - char *username: 用户名，字符串指针，表示当前用户。这个信息用于个性化用户的游戏体验，
                      包括加载和保存用户特定的场景设置和进度。
    - char *nowfield: 当前场景的名称，字符串指针，指向表示当前用户选择的检测场景的字符串。
                      这个信息用于确定哪些数据需要被加载和保存，以及如何绘制当前场景。
output:
    - int: 返回操作结果，通常是状态码或导向其他页面的标识
************************************************************* */
```