2016 年秋季学期

《嵌入式系统》&&《人机交互理论与技术》综合大作业

manuduino

实验报告

作者：

梁泽宇（2014011381）

谭思楠（2013011720）

1　选题背景

本项目的设计灵感来源于一款自动机编程游戏——manufactoria。在此游戏中，玩家需要在网格中放置各种元件（箭头、颜色选择器、输出装置等），来对以红蓝两种颜色组成的纸带进行一系列操作从而完成指定功能（该游戏设置了 31 个关卡，每关需要实现一个功能）。

manufactoria 游戏的设计中处处体现了程序设计思想，并引入算法概念，其中的某些关卡需要相当有技巧性的算法设计才能通过，对脑力是一个巨大的挑战。同时，该游戏通过图形界面（UI）与用户（玩家）交互，改变了往常程序设计以文本（命令行）界面交互的枯燥无味的缺点，大大增加了编程的趣味性。

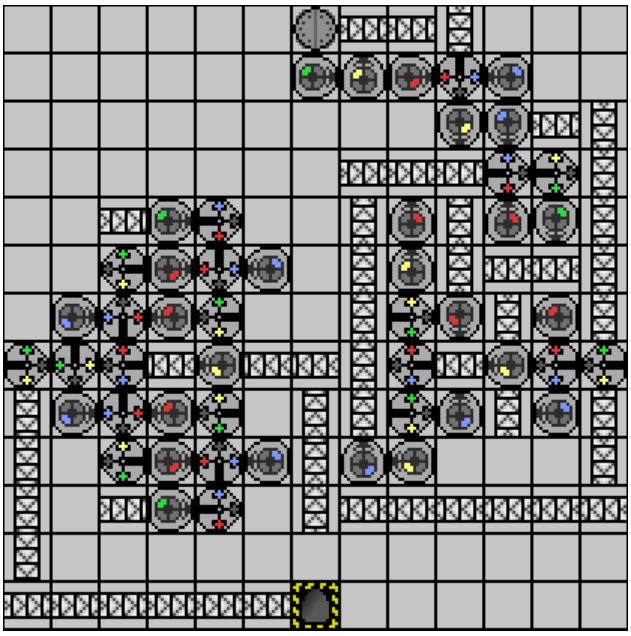

图 1: manufactoria 游戏界面截图

manufactoria 游戏为我们提供了一个很好的图形化编程平台的范例。本学期，在学习了《人机交互理论与技术》、《嵌入式系统》等课程后，我们想到了以 manufactoria 为原型设计一个图形化编程平台的设计方案，将编程语言中的常见语句与模块抽象为图形元件放入网格中，并通过网格中的箭头指示程序执行流程，来构成完整程序。同时，编写的程序可以由用户在

图形界面编译成嵌入式开发板支持的代码，并在嵌入式开发板上执行，在显示屏等输出设备上输出结果。

我们最初的设计中选用的嵌入式开发板为 Arduino Uno，所以本项目命名为 `manuduino`。但设计过程中由于硬件兼容性等问题，我们最终改为以 Raspberry Pi 3（树莓派 3）作为使用的嵌入式开发板，并以液晶显示器作为输出设备。

2 功能介绍

2.1 图形界面（GUI）

2.1.1 制造模式（Manu Mode）

制造模式是本项目的主要模式，其图形界面如下图所示：

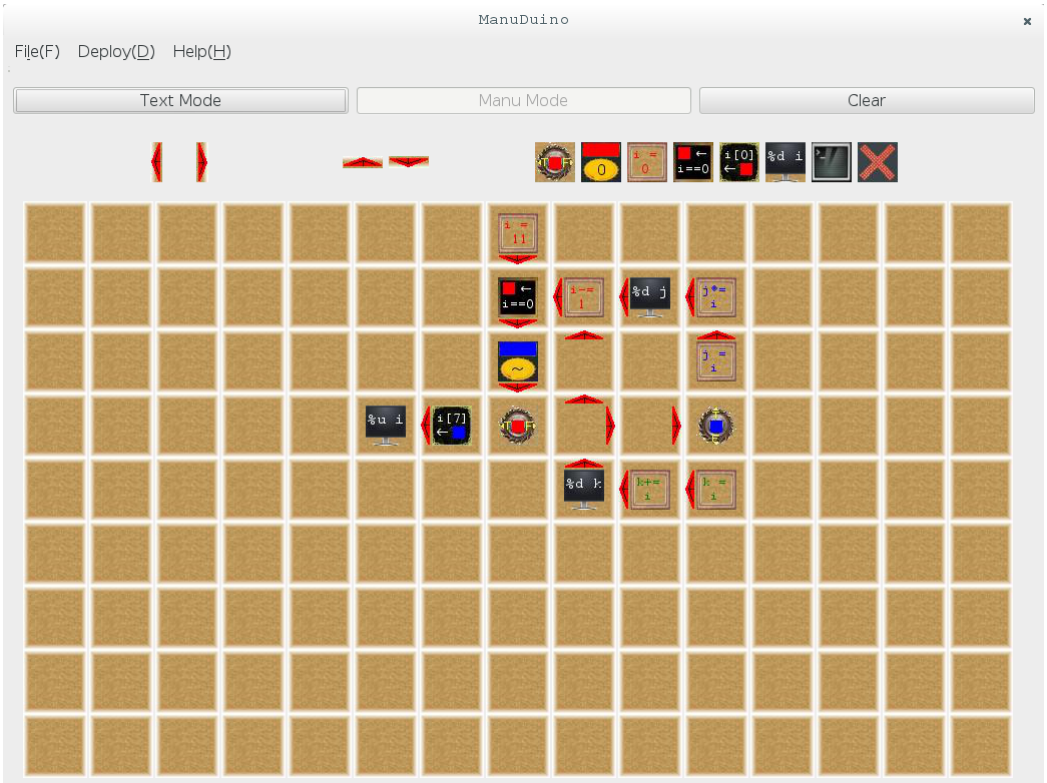

图 2: `manuduino` 制造模式图形界面

主界面为 9 行 15 列的网格，其中每个格子中心处可以放置至多一个中央元件（Central Entity），四周可以放置上/下/左/右四个方向的箭头（Arrow），每个格子中，上下箭头至多一个，左右箭头至多一个。

主界面以上为操作菜单栏（Operation Menu），包含上下左右箭头、各种中央元件以及删除操作。用户可以通过鼠标左键单击这些元件的图标并用鼠标拖动到任意一个格子里来放置这些元件。各个中央元件以及删除操作的说明如下表（对于变量的说明见 2.3 节）：

名称	图标	说明	备注
选择器（Selector）		根据 bool 变量的值决定走向，为 True 则走 T 方向，为 False 则走 F 方向。	双击菜单栏图标可选择具体的 bool 变量与 T、F 方向。该元件所在格无箭头。
布尔控制器（Bool Controller）		给 bool 变量赋值，可赋为 0（False）、1（True）或 ~（取反）。	双击菜单栏图标可选择具体的 bool 变量与所赋的值。
整数控制器（Int Controller）		给 int 变量赋值，支持 =、+=、-=、*=、/=、%= 等多种赋值方式，所赋的值可以为 0~15 的常量或其它 int 变量。	双击菜单栏图标可选择具体的 int 变量、赋值方式与所赋的值。
比较器（Comparator）		将 int 变量的值与 0 比较，并将比较结果赋给 bool 变量，比较运算有 =、>、>=、<、<=、!= 等。	双击菜单栏图标可选择具体的 int 变量、比较方式与赋给的 bool 变量。
置位器（Bit Setter）		将 int 变量的某一位置为某个 bool 变量的值。	双击菜单栏图标可选择具体的 int 变量、所置的位与 bool 变量。
输出装置（Output）		输出 int 变量的值，输出格式有有符号整型、无符号整型和字符型。	双击菜单栏图标可选择具体的 int 变量与输出格式。
终端（Shell）		插入自定义代码。	拖动插入终端后，双击可打开对话框，输入自定义代码。
删除操作（Clear Operation）		将其拖动到某个格子内，可删除该格所有元件（包括箭头）。	

在制造模式中，点击界面上的 “Clear” 按钮，可清空所有网格。

2.1.2 文字模式（Text Mode）

本项目除制造模式外，还有一个用于用户测试的文字模式，其界面如下图所示：

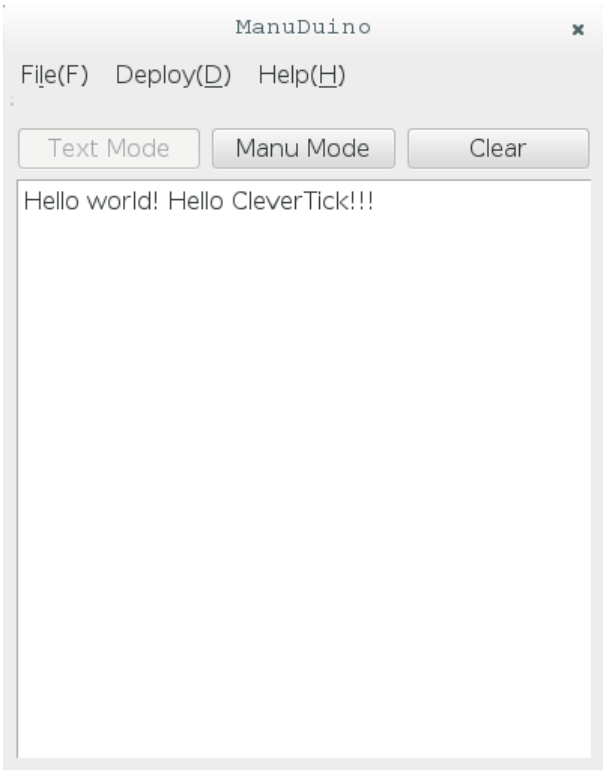

图 3: `manuduino` 文字模式界面

在文字模式中，用户可直接在文本框内输入文字，执行效果为在显示器上输出文本框内的文字。

此模式中点击 “Clear” 按钮，可清空文本框（不会影响制造模式中的程序）。

点击 “Manu Mode” 与 “Text Mode” 按钮，可在两种模式间切换。由于文字模式仅用于用户测试，因此本文以下部分，均默认在制造模式下进行。

2.2 程序编译与执行

在图形界面设计程序之后，需要将程序进行编译，并在嵌入式开发板（Raspberry Pi 3）上执行，在显示屏上输出结果。

在 Raspberry Pi 3 一端上，需要预先安装 JRE 和 processing 的 ARM 版本一边执行，并且需要配置 ssh 远程登录。在开发端完成图形设计、编译后，之后的部署命令会通过 ssh 和 scp 命令自动完成远程的部署和执行。

2.3 运行规则

`manuduino` 程序中可使用 8 个 bool 变量（用 8 种不同颜色的色块表示）和 3 个 8 位整型（int8）变量（命名为 `i`、`j`、`k`）。

程序运行时，将有一个虚拟的 NPC 在主界面网格内移动，初始时，其位于第 1 行第 8 列（即最上一行中间的格子）上方，初始方向向下（也就是其最先进入的格子为第 1 行第 8 列）。所有变量初始值均为 False（bool 变量）/0（int 变量）。

程序运行过程中，该虚拟 NPC 每到达一个格子，就会根据该格的中央元件指示，执行相应操作（如某一格中央元件为写有 `i=11` 的整数控制器，则 NPC 到达该格后，会执行将变量 `i` 的值赋为 11 的操作），若该格没有任何中央元件，则什么也不做。之后，NPC 会根据该格含有的箭头方向，决定其下一步的走向（对于中央元件是选择器的格子，则根据选择器的结果决定下一步走向）：

- 若该格含有两个箭头（上下箭头、左右箭头各一个），则当 NPC 上一步走进该格时的方向为上/下方向，则下一步走上/下箭头指示的方向；当 NPC 上一步走进该格时的方向为左/右方向，则下一步走左/右箭头指示的方向；
- 若该格只有一个箭头，则不论 NPC 上一步走进该格时是什么方向，下一步均走这个箭头指示的方向；
- 若该格没有箭头（中央元件是选择器的格子除外），或者 NPC 走出了网格区，则程序立即终止（除此以外，没有其它办法能让程序终止）。

3 设计详情

3.1 模块划分

本项目主要在 Qt 4 开发环境下编写，以 Raspberry Pi 3 作为嵌入式开发板（用于执行程序）。整个项目划分为以下模块：

- 元件（Entity 类）：定义图形界面中用到的各种图形元件，其派生类有 Arrow（定义箭头）、EmptyEntity（定义空元件）、OperationEntity（定义菜单栏操作元件）等；
- 中央元件对话框（*Dialog 类）：定义各种中央元件双击后弹出的对话框内容；
- 主界面网格（BoardGrid 类）：定义主界面的网格，每个格子由 3 行 3 列的 9 个元件（Entity）构成，其中左上、右上、左下、右下元件均固定为空，上中、左中、右中、下中元件留给箭头使用（无箭头时为空），最中间的为中央元件。网格类提供用户操作的主要接口（放置元件、删除等），也为编译器调取主界面局面信息提供接口。
- 图形界面主窗口（MainWindow 类）：定义图形界面的主窗口布局，以及读取用户鼠标操作事件（单击、双击、释放、拖动等）并处理；
- 编译器（Compiler 类）：定义了编译制造模式和文字模式到 Java 代码的算法。细节见后文描述；
- 远程部署（RemoteManager 类）：自动通过 SSH，向连接无线网络的 Raspberry Pi 3 远程进行部署和执行；
- 服务器配置（ServerConfigurationDialog 类）：用于通过图形界面调整配置远程服务器的相关参数。

3.2 一些设计细节和技巧

3.2.1 鼠标拖动效果的设计

为了增强交互效果，本项目实现了一个重要的功能——将元件用鼠标左键拖动到网格内。

Qt 中鼠标拖动的事件处理集成在鼠标移动（`mouseMoveEvent()`）中，这样就无法判断鼠标指针移动操作是否在点击操作元件之后发生。为了解决这个问题，我们利用 Qt 的“信号——槽”机制，设计了“激活”机制——在 OperationEntity 类中进行鼠标按下事件（`mousePressEvent()`）处理，当用户按下了操作菜单栏上的某个元件图标时，就发射“该元件被激活”的信号，主窗口类 MainWindow 的一个槽接收该信号，判定这个元件被激活。此后在鼠标移动时，鼠标指针（用 QCursor 控制）会被置为目前被激活的元件（如果有的话），以实现“拖动”效果。

3.2.2 鼠标释放事件处理机制

在主界面中，当用户拖着某个元件至某处释放鼠标左键时，需要判断该释放处是否为某个网格内部，若是则在该网格中加入元件，若否则什么也不做。因此，在处理鼠标释放事件时，我们需要读取该释放事件发生的全局坐标（即鼠标指针当前在主窗口中的位置），并调用网格（BoardGrid）类的 `mapToGlobal()` 接口，获取每个网格的坐标范围，以确定释放处是否位于及位于哪个网格当中。

3.2.3 编译、执行的实现细节

编译过程的具体细节是这样的：

- 首先，“终端”模块允许用户插入任意 processing（Java）代码，但是全局变量的定义显然必须脱离控制流，因此，定义扩展关键字“___global;”，以此行开头的终端代码会被调整到整个 processing 程序的变量定义段；
- 接下来程序生成其运行时代码。利用 processing 的变量定义个 setup 函数，整个网格中所有的箭头信息都被转化为一个 Java 数组存储。而程序的真正执行是通过在 processing 的 draw 函数中以状态机模式来运行。每一次经过自动循环的 draw 函数，都会执行为坐标匹配的 manuduino 模块编译出的代码。除此之外，编译器还创建了一段运行时环境代码，用于根据之间存储的网格信息和选择器信息决定网格位置的移动，以及在执行到没有箭头的位置之后自动停机；
- 编译完成之后，程序通过 ssh 和 scp 命令，配合一个图形化的 SSH AskPass 程序，像远程服务器传输创建目录和复制文件的命令。最后，

当用户要求执行代码时，则通过 ssh 命令配置 X 服务器连接信息并执行编译好的 processing 程序。

4 项目测试及其结果

参见本报告所在目录中的视频附件。

5 代码仓库

在开发本项目的过程中，我们使用 Git 进行版本控制与代码的维护工作。代码仓库位于 Github 上，地址为：

`https://github.com/MatoNo1/manuduino.git`