目录

汇报部分

帮助

API Docs

Doodle Jump

项目主页 | 项目文档 | 下载游戏 | 国内镜像

用 C 语言在命令行下复刻经典手机游戏涂鸦跳跃。

介绍 Introduction

> 把红旗插上无穷的高峰！

向上跳跃，不要坠入虚空！

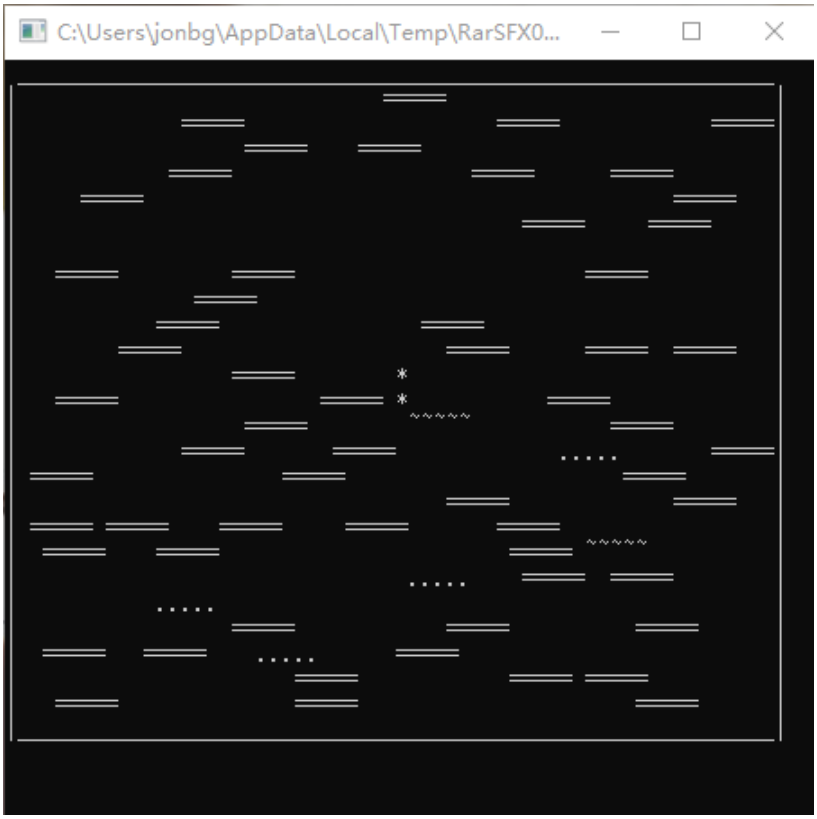
玩法 Howto

用方向键操纵小人左右移动，当小人落在跳板上时会向上弹起。

左移： A 或 ←

右移： D 或 →

截图 Screenshots



主要贡献者 Main Contributors

@lhy ： Backend Developer

@lz ： Frontend Developer

@Jonbgua ： Program Manager, Software Architect

需求分析

本节将叙述项目应实现的目标及需求。

功能分析

- 单机游戏
- 可高度定制

性能分析

- 画面流畅，不闪屏
- 随机生成的、无限长的地图

主要数据结构

本节将对本项目的主要数据存储方式作简要说明。

全局设置项

```
struct settings {
    int map_height;              //地图高度
    int map_width;               //地图宽度
    int map_board_length;        //跳板长度
    int player_height;           //玩家高度
    int player_width;            //玩家宽度
    int dp_tpf;                  //每多少tick刷新画面
    int dp_tpl;                  //每多少tick刷新一行
    int dp_tpm;                    //每多少tick玩家因重力下降/弹射上升一行
    int dp_line;                    //当前游戏最底端所在行的编号
};
```

命名规则

- `map_` 开头的变量与地图初始化有关
- `player_` 开头的变量与玩家有关
- `dp_` 开头的变量与显示（Display）有关

玩家

```
struct player {
    float x;
    int y;                          //玩家位置,以玩家最下端的中间位置为基准
    int prev_board_line;                 //上一次碰撞的板的所在行
    int remain_bounce_line;           //还能往上跳多少行
};
```

跳板

使用链表方式存储画面上所有跳板的数据。

```
struct board {
    int line_id;                 //板所在行
    int x;                       //左端点位置
    char type;                   //板的类型，默认为0
    struct board* next;          //下一个节点
};
```

跳板类型

- Type = 0：普通跳板

系统模块设计

本节将介绍该项目的整体架构和每个模块的功能。

整体设计

游戏分为前端渲染和后端控制两大模块，两者通过标准的数据接口连接。前后端分离将有助于协同开发、分模块调试以及模块单独升级优化。

游戏过程中每一帧都会进行如下操作：

前端按键捕获 -> 后端数据处理 -> 前端渲染 -> 前端打印输出

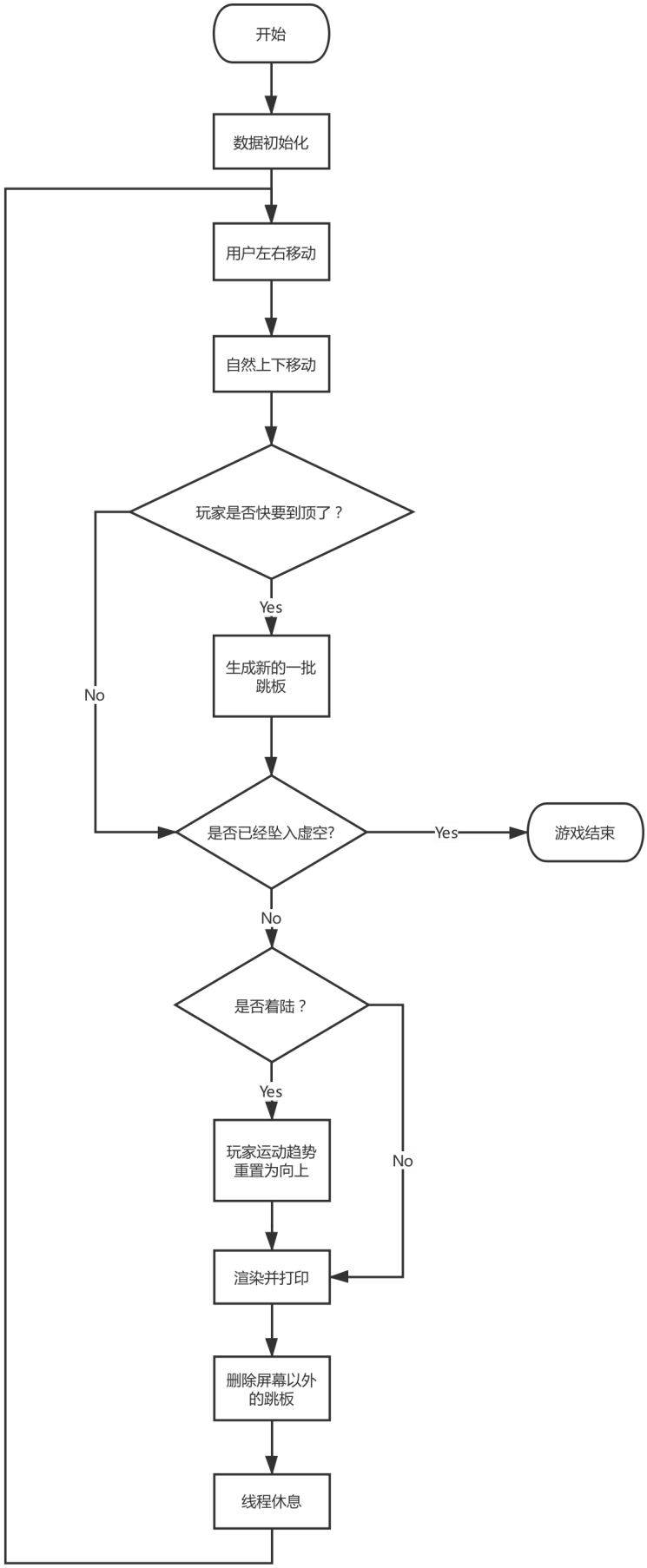
约定：

- 函数命名均为

```
type verb_noun(int, char, struct others*, struct setting*)
```

- 变量命名
  - 均使用单数

流程图


开始
数据初始化
用户左右移动
自然上下移动
玩家是否快要到顶了？
Yes
No
生成新的一批跳板
是否已经坠入虚空?
Yes
游戏结束
No
是否着陆？
Yes
No
玩家运动趋势重置为向上
渲染并打印
删除屏幕以外的跳板
线程休息

前端

前端负责直接与用户交互。

菜单类模块

菜单类模块主要用于显示各种菜单并实现各菜单之间的跳转，菜单包括：

- 主菜单
- 排行榜界面
- 游戏帮助界面

输入获取模块

此模块将获取用户的键盘输入，并返回对应的数值。

渲染类模块

`渲染` 是将后端处理好的数据以形象化的方式输出到二维字符数组中，为其后 `打印输出到屏幕` 服务。

跳板渲染

遍历跳板链表，在字符数组上将每个跳板画出。

玩家渲染

读取玩家数据，在字符数组上将玩家模型画出。

打印模块

`打印` 是将渲染完成的字符数组打印输出到屏幕上。

后端

后端负责数据处理（判断、修改等）。

跳板信息处理

跳板信息处理类模块负责游戏过程中跳板的生成、位置移动等数据处理。

核心模块

核心模块即对于所有类型的跳板均生效的模块，具体有：

- 链表操作（节点的增删改）
- 随机生成跳板

特殊类型跳板数据的处理

此分类下的模块只对特定类型的跳板生效。主要实现的功能如下：

- 横动跳板持续左右移动

玩家移动

玩家移动类模块处理游戏过程中玩家模型的各类移动。

操控移动

操控移动即通过键盘输入控制玩家模型作水平方向的移动。

自然移动

自然移动即玩家因受到重力作用或反冲作用而作的惯性运动。

游戏进程

此分类下的模块处理一些与游戏相关的非主干操作。

配置文件相关

此类模块主要负责：

- 读取配置文件
- 解析配置文件并将对应数值存储进 setting 结构体中
- 若配置文件不存在，将自动按照缺省设置新建一份配置文件

结束判断

此模块将判断玩家是否已掉出游戏区域外（游戏结束）。

分数记录

此模块记录玩家的本地高分排行。

网络通讯

网络通讯类模块负责进行互联网上传和下载操作。

上传本地最高分至远端

此模块将构造一个带参数的 HTTP GET 请求，将本地的最高分数上传至远程服务器。

从远端获取全球排行

此模块将向远程服务器发送一个 HTTP GET 请求，服务器将返回一个包含 Top N 玩家名和对应分数信息的字符串，模块负责解析与存储返回的排行榜信息。

技术难点及攻克

在本项目中遇到了若干技术难点，下文将列出所遇到的技术难点以及难点的解决方案。

前端

控制台双缓冲区解决闪屏问题

pass

参考资料

双缓冲控制台防闪屏技术 - CSDN

后端

与远端服务器建立连接

众所周知，C 对于网络请求的实现并不如新潮的语言（Python、Java 等）那般优雅，因此实现一个简单的 HTTP GET 请求也需要一番折腾。

参考资料

pass

与课程的联系

复习

1. 指针
    - 基本指针操作
    - 指针作为形参
2. 结构体
    - 链表操作
3. 文件操作
    - 配置文件的读写

初探

系统架构

1. 模块化
2. 前后端分离

前端

1. 双缓冲区：减少屏幕闪烁

后端

1. C 的网络编程

协作开发

1. 基于 Git 的多人协同开发

编译指南

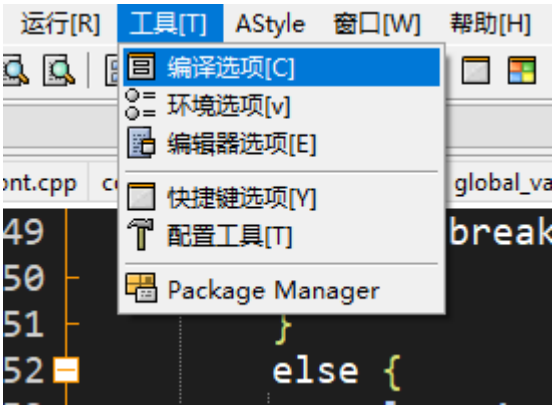
本节将介绍在各种流行的 IDE 下编译本项目的简要步骤。

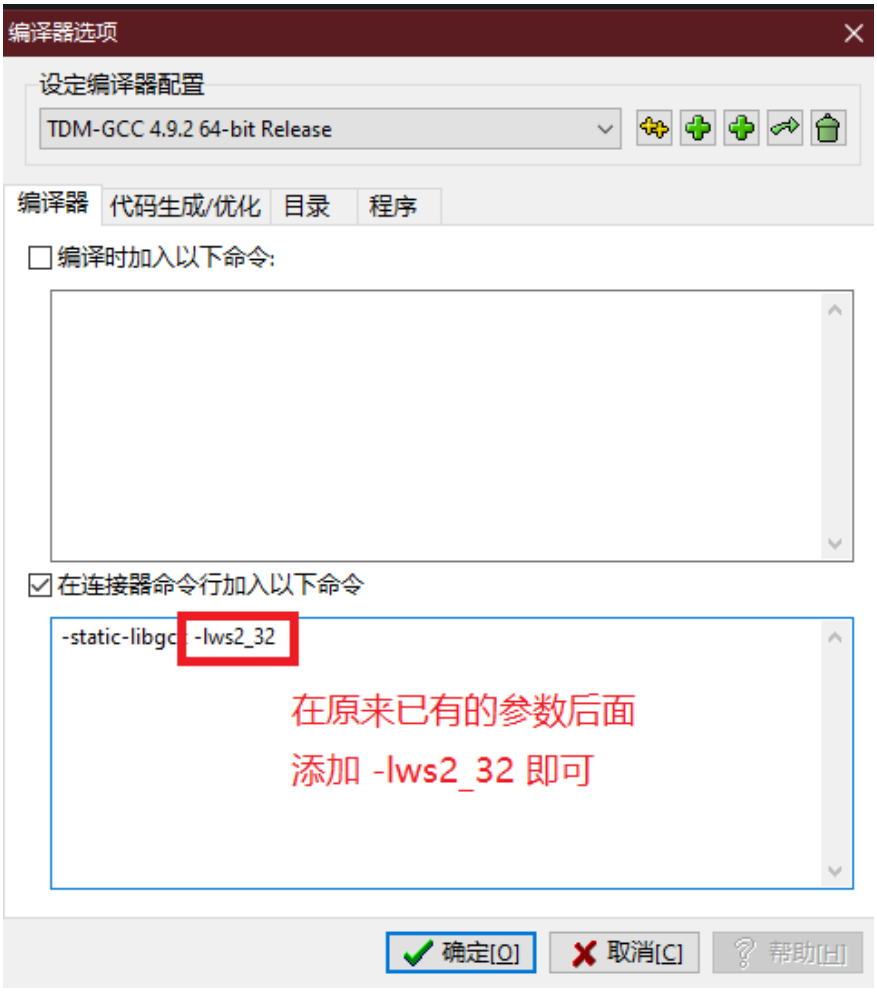
在 Dev-C++ 下编译

本节将介绍在 Dev-C++ IDE 下编译本项目的简要步骤。

Step 1. 修改编译器选项

在 `工具 - 编译器选项` 中勾选 `在连接器命令行加入以下命令` 并且在下面的文本框中追加 `-lws2_32` 选项即可。





Step 2. 进行编译

仅需打开 `base.cpp` 文件并对其进行编译即可。