你好玉米

你好玉米设计实现

版本 <1.0>

[注：以下提供的模板用于程序设计课程设计。其中包括用方括号括起来并以蓝色斜体（样式=InfoBlue）显示的文本，它们用于向作者提供指导，在发布此文档之前应该将其删除。按此样式输入的段落将被自动设置为普通样式（样式=Body Text）。]

[要定制 Microsoft Word 中的自动字段（选中时显示灰色背景），请选择 File>Properties，然后将 Title、Subject 和 Company 等字段替换为此文档的相应信息。关闭该对话框后，通过选择 Edit>Select All（或 Ctrl-A）并按 F9，或只是在字段上单击并按 F9，可以在整个文档中更新自动字段。对于页眉和页脚，这一操作必须单独进行。按 Alt-F9，将在显示字段名称和字段内容之间切换。有关字段处理的详细信息，请参见 Word 帮助。]

文档信息

|  |  |
| --- | --- |
| 标题： | 你好玉米 |
| 文件位置： |  |
| 版本： | 2.0 |
| 提交人： | 张云翔 |
| 提交日期： | <13/7/2014> |
| 状态： | 半成品，bug多 |

修订历史记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** | **审核人** |
| <13/7/2014> | <1.0> | <bug多，下落会出bug> | <张云翔> | <> |
| <7/24/2014> | <2.0> | <> | <张云翔> |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 范围 4

1.3 定义、首字母缩写词和缩略语 4

1.4 参考资料 4

2. 软件架构 4

3. 静态设计 4

4. 动态设计 4

你好玉米设计实现

# 简介

[简介应提供整个文档的概述。它应包括此文档的目的、范围、定义、首字母缩写词、缩略语、参考资料和概述。]

## 目的

你好玉米味一款通过鼠标连线实现消除的游戏，此文档的目的在于通过介绍实现方法，来让读者对游戏的实现有一个基本的认识

## 范围

此文档的对象为你好玉米这个游戏以及hge引擎

## 定义、首字母缩写词和缩略语

enum COLOR {PURPLE, RED, YELLOW, BLUE, GREEN}; // 用来表示玉米颜色

enum STATE {SLEEP, AWAKE}; // 用来表示玉米是否被点击

enum KIND {KNORMAL, FROZEN};

enum EFFECT {ENORMAL, BOMB, HORIZONTAL, VERTICAL}; // 用来表示玉米种类

class Corn : public hgeSprite {

private:

string name; // 玉米图片名字

COLOR color; // 玉米颜色

STATE state; // 玉米状态，是否被点击

KIND kind; // 玉米种类，是否冰冻

EFFECT effect;// 玉米的效果

pair<double,double> location;

public:

Corn(); // 默认复制构造函数

Corn(HTEXTURE tex,string s); // corn纹理为tex，名字为s，坐标为x，y

string Get\_name(); // 取corn的名字

COLOR Get\_color(); // 取corn的颜色

STATE Get\_state(); // 取corn的状态，是否被点击

KIND Get\_kind(); // 取corn的种类，是否冰冻

EFFECT Get\_effect();// 取corn的效果

pair<double,double> Get\_location(); // 取corn的地址，first为x（第x列），second为y（第y行）

void Set\_color(COLOR c); // 设置corn的颜色为c

void Set\_state(STATE s); // 设置corn的状态为s

void Set\_kind(KIND k); // 设置corn的种类为k

void Set\_effect(EFFECT e);// 设置corn的效果为e

void Set\_location(pair<double,double> p); // 设置corn的地址为p

};

// GUI类用来表示整个图形环境

class GUI

{

private:

STAGE stage; // stage用来记录是在菜单还是游戏中

Line\* line; // line用来记录连线以及消除的信息

Map\* map; // map用来记录6\*6corn的所有信息

HGE\* hge; // hge框架

Count\* count; // count用来时间分数等信息

HEFFECT music;

hgeSprite background; // background为背景

Menu\* menu; // menu即为菜单信息

static void draw\_corn(); // 绘制玉米

static void draw\_background(); // 绘制背景

static void draw\_count(); // 绘制计分板

static void draw\_menu(); // 绘制菜单

static void initial\_menu(); // 初始化菜单

static void initial\_corn(); // 初始化玉米

static void initial\_background();// 初始化背景

static void initial\_hge(); // 初始化hge框架

static void initial\_count(); // 初始化计分板

static void initial\_music();

static bool MenuRender(); // 菜单时的RenderFunc

static bool GameRender(); // 游戏时的RenderFunc

public:

GUI(); // GUI构造器

HGE\* get\_hge(); // 返回hge

Line\* get\_line(); // 取得连线信息

Map\* get\_map(); // 取得地图（玉米）信息

Count\* get\_count(); // 取得计分信息

Menu\* get\_menu(); // 取得菜单信息

STAGE get\_stage(); // 取得是在游戏或菜单

void set\_stage(STAGE s); // 设置stage

static bool RenderFunc(); // hge每一帧调用的绘图函数

static void initial(); // 初始化

static void reinitial(); // 重新初始化

~GUI(); // 析构

};

class Count

{

private:

hgeFont\* fnt; // fnt用来加载字体，显示计分板

int score; // score用来记录分数

int time; // time用来记录总帧数

double game\_time; // game\_time用来记录游戏时间

int best\_score; // best\_score用来记录最佳分数

int step; // step用来记录消除次数

public:

Count(); // 构造器

void add\_score(int s); // 加分，分数上涨s

void add\_time(); // 加帧数1

void add\_game\_time(); // 加游戏时间（每一帧调用之间的δt）

void add\_step(); // 加消除次数1

int get\_time(); // 取得帧数

double get\_game\_time(); // 取得游戏时间

int get\_score(); // 取得分数

int get\_best\_score(); // 取得最佳分数

int get\_step(); // 取得消除次数

void set\_fnt(string s); // 设置字体fnt为s

void set\_game\_time(double g); // 设置游戏时间为g

void set\_score(int s); // 设置分数为s

void set\_best\_score(int b); // 设置最佳分数为b

void set\_step(int s); // 设置消除次数为s

void Render(); // 显示

void initial(string s); // 初始化

~Count(); // 析构器

};

class File

{

private:

ofstream save; // 用来存储的流

ifstream load; // 用来读取的流

ofstream record; // 用来记录视频的流

ifstream review; // 用来回放视频的流

bool is\_recording; // 表示是否在记录

bool is\_reviewing; // 表示是否在回放

public:

File(); // 构造器

void read\_map(int i); // 读取地图数据

void read\_best\_score(); // 读取最好成绩

void open\_record(); // 打开记录视频文件

void close\_record(); // 关闭记录视频文件

void open\_review(); // 打开回放视频文件

void close\_review(); // 关闭回放视频文件

void write\_file(string filemane); // 向filename文件中写入玉米信息

void read\_file(string filename); // 从filename文件中读取玉米信息

void record\_file(); // 记录当前视频信息

void review\_file(); // 回放当前视频信息

void write\_best\_score();// 写入最佳成绩

bool get\_is\_recording();// 取得是否在记录

bool get\_is\_reviewing();// 取得是否在回放

void set\_is\_recording(bool b); // 设置是否在记录状态为b

void set\_is\_reviewing(bool b); // 设置是否在回放状态为b

};

class hgeGUIMenuItem : public hgeGUIObject

{

public:

hgeGUIMenuItem(int id, hgeFont \*fnt, HEFFECT snd, float x, float y, float delay, char \*title);

virtual void Render();

virtual void Update(float dt);

virtual void Enter();

virtual void Leave();

virtual bool IsDone();

virtual void Focus(bool bFocused);

virtual void MouseOver(bool bOver);

virtual bool MouseLButton(bool bDown);

virtual bool KeyClick(int key, int chr);

private:

hgeFont \*fnt;

HEFFECT snd;

float delay;

char \*title;

hgeColor scolor, dcolor, scolor2, dcolor2, sshadow, dshadow;

hgeColor color, shadow;

float soffset, doffset, offset;

float timer, timer2;

};

class Line

{

private:

vector<pair<int,int>> dot; // dot记录被连起的点

vector<pair<int,int>> vanish; // vanish记录被消除的点（由于有特殊效果，被消除的点不一定是被连起的）

public:

void add\_dot(int x, int y); // 加入被连起的点x，y

void add\_vanish(int x,int y); // 加入被消除的点x，y

int dot\_number(); // 返回被连起的点的数量

int vanish\_number(); // 返回被消除的点的数量

pair<int, int> last\_dot(); // 返回最后一个被连起的点的坐标

pair<int, int> get\_dot(int i); // 取得第i个被连起的点的坐标

pair<int, int> get\_vanish(int i); // 取得第i个被消除的点的坐标

void delete\_last\_dot(); // 删除最后一个被连起的点

void delete\_last\_vanish(); // 删除最后一个被消除的点

void clean\_dot(); // 清除所有的被连起的点

void clean\_vanish(); // 清除所有的被消除的点

bool compare\_last\_dot(int x, int y); // 与最后一个被连起点的位置比较，如果一样，返回true

bool compare\_dot\_i(int i,int x,int y); // 与第i个被连起点的位置比较，如果一样，返回true

bool last\_round(int x,int y); // 如果为最后一个点的周围8个，返回true

bool dot\_is\_empty(); // 检测连线是否为空

bool vanish\_is\_empty(); // 检测消除是否为空

};

// 逻辑类，调用所有逻辑动作

class Logic {

private:

static bool esc(); // 检查是否调用了esc退出

static bool is\_vanished(int i, int j); // 检测第i列，第j行玉米是否消失

static bool is\_recording(); // 检测是否在记录

static bool is\_reviewing(); // 检测是否在回放

static bool can\_vanish(); // 检测是否消失

static bool can\_add\_special(); // 检测是否能满足加入玉米的条件（为连线点大于5个）

static bool MenuFrame(); // menu时调用的逻辑

static bool GameFrame(); // game时调用的逻辑

static bool same\_color(int i1,int j1,int i2,int j2); // 检测i1,j1玉米与i2,j2玉米是否同色

public:

static void connect(); // 检查鼠标连接玉米

static void associate(); // 检查所有被连带消除的玉米

static void save(string filename); // 存储玉米信息到filename中

static void load(string filename); // 从filename中读取玉米信息

static void record(); // 记录这一帧的视频

static void review(); // 回放这一帧的视频

//static void previous();

static void vanish(); // 玉米消失

static void fall(); // 玉米掉落

static void add\_special(); // 加入特殊玉米

static void statistic(); // 统计分数，时间等信息

static void fresh(); // 刷新图片为全部未点击状态

static void clean(); // 清除连线及消除数据

static void create(int i,int j); // 在第i列,j行随机生成一个玉米

static void start\_review(); // 开始回放

static void stop\_review(); // 结束回放

static void start\_record(); // 开始记录

static void stop\_record(); // 结束记录

static void return\_to\_menu(); // 返回到menu stage

static void return\_to\_game(); // 返回到game stage

static bool FrameFunc(); // hge每一帧调用的逻辑函数

};

class Map // 用来存储地图信息

{

private:

Corn corn[6][6]; // 6\*6的玉米矩阵

public:

Map();

void Set\_color(int i,int j, COLOR c); // 设置第i列，第j行的颜色为c

void Set\_kind(int i,int j, KIND k); // 设置第i列，第j行的种类为k

void Set\_state(int i,int j,STATE s); // 设置第i列，第j行的状态为s

void Set\_effect(int i,int j,EFFECT e); // 设置第i列，第j行的效果为e

void Set\_location(int i,int j,pair<double,double> location); // 设置第i列，第j行的位置为location

void Set\_corn(int i1,int j1,int i2,int j2); // 将第j1行第i1列的玉米赋给第j2行第i2列的玉米

void Set\_corn(int i,int j,Corn& c); // 设置第i列，第j行的玉米为c

void Set\_texture(int i,int j,HTEXTURE h); // 设置第i列，第j行的纹理为h

void Render(); // 显示

void initial(); // 初始化

string Get\_name(int i,int j); //

COLOR Get\_color(int i,int j); //

KIND Get\_kind(int i, int j);

STATE Get\_state(int i,int j);

EFFECT Get\_effect(int i, int j);

pair<double,double> Get\_location(int i,int j);

HTEXTURE Get\_texture(int i,int j);

~Map();

};

class Menu

{

private:

bool enter; // enter判定是否进入游戏

HEFFECT snd; // snd为按钮的音效

HTEXTURE tex; // tex为鼠标的纹理

hgeQuad background; // background为背景

hgeQuad touch; // touch为enter的图片

hgeGUI\* hgui; // hgui为button的连接界面

hgeFont \*mfnt; // mfnt为按钮的字体

hgeSprite \*spr; // spr为鼠标精灵

public:

bool Frame(); // 逻辑函数

bool Render(); // 显示函数

void initial(); // 初始化

~Menu(); // 析构器

};

## 参考资料

[本小节应完整列出此文档中其他部分所引用的任何文档。每个文档应标有标题、文件名和位置。这些信息可以通过引用附录或其他文档来提供。]

|  |  |
| --- | --- |
| 文件名和位置 | 标题 |
|  |  |

# 软件架构

[本小节应提供架构的概述，包括软件高层分解图等]

底层为Corn类以及hgeGUIMenuItem类，Corn类通过Map类包装，hgeGUIMenuItem类被map类使用。

次底层为Map,Count,Line,Menu类，为基本元素

高层为GUI,Logic,File类

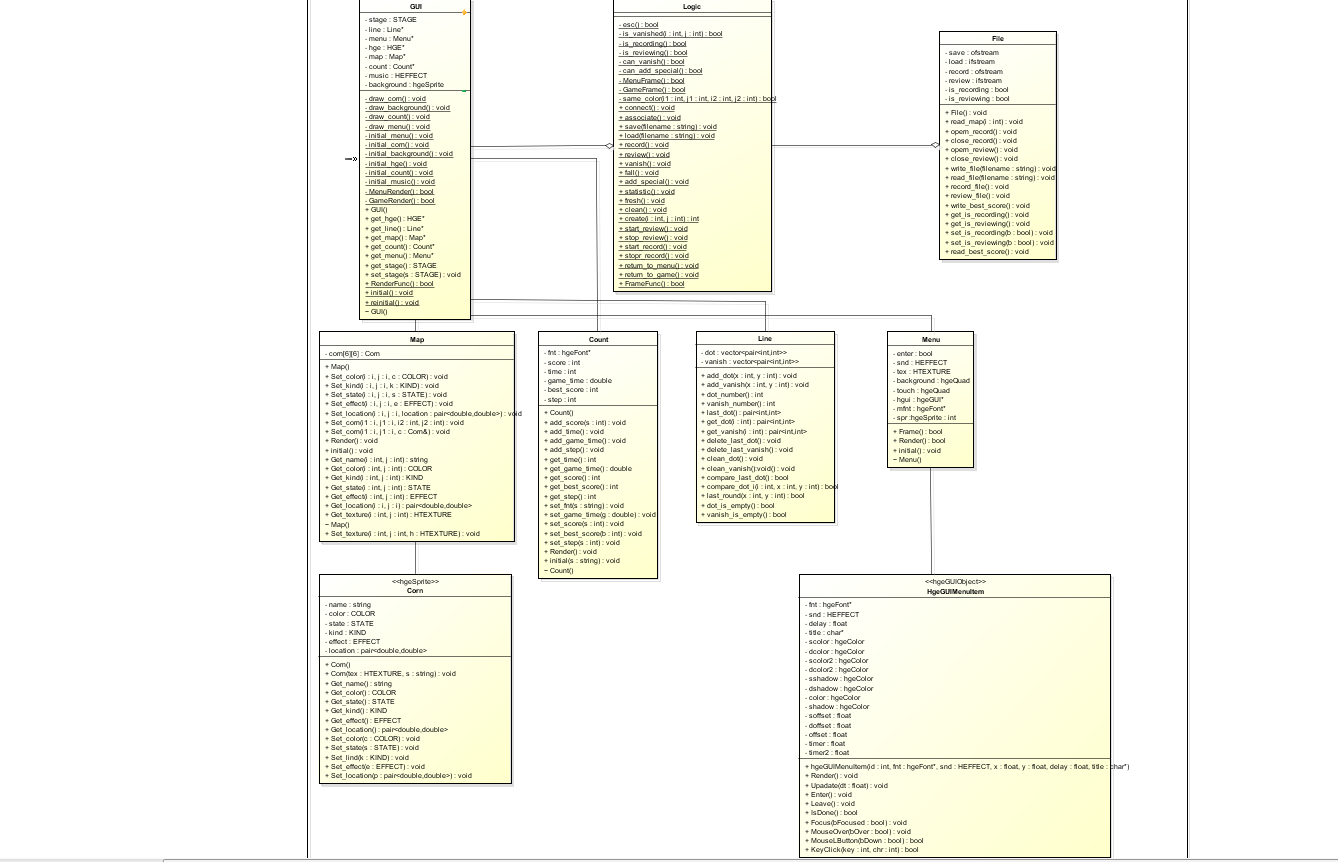
GUI类包含了Map类，Count类，Menu类，为图形化界面，通过全局变量gui

Logic类控制逻辑

File类控制文件读写，通过全局变量file

用户，文件与Logic的交互通过Logic来实现

# 静态设计



# 动态设计

[本小节应对系统中各个类之间的交互机制进行描述，可以采用UML顺序图或状态图来描述]

