**Gold Miner(黄金矿工)实验报告**

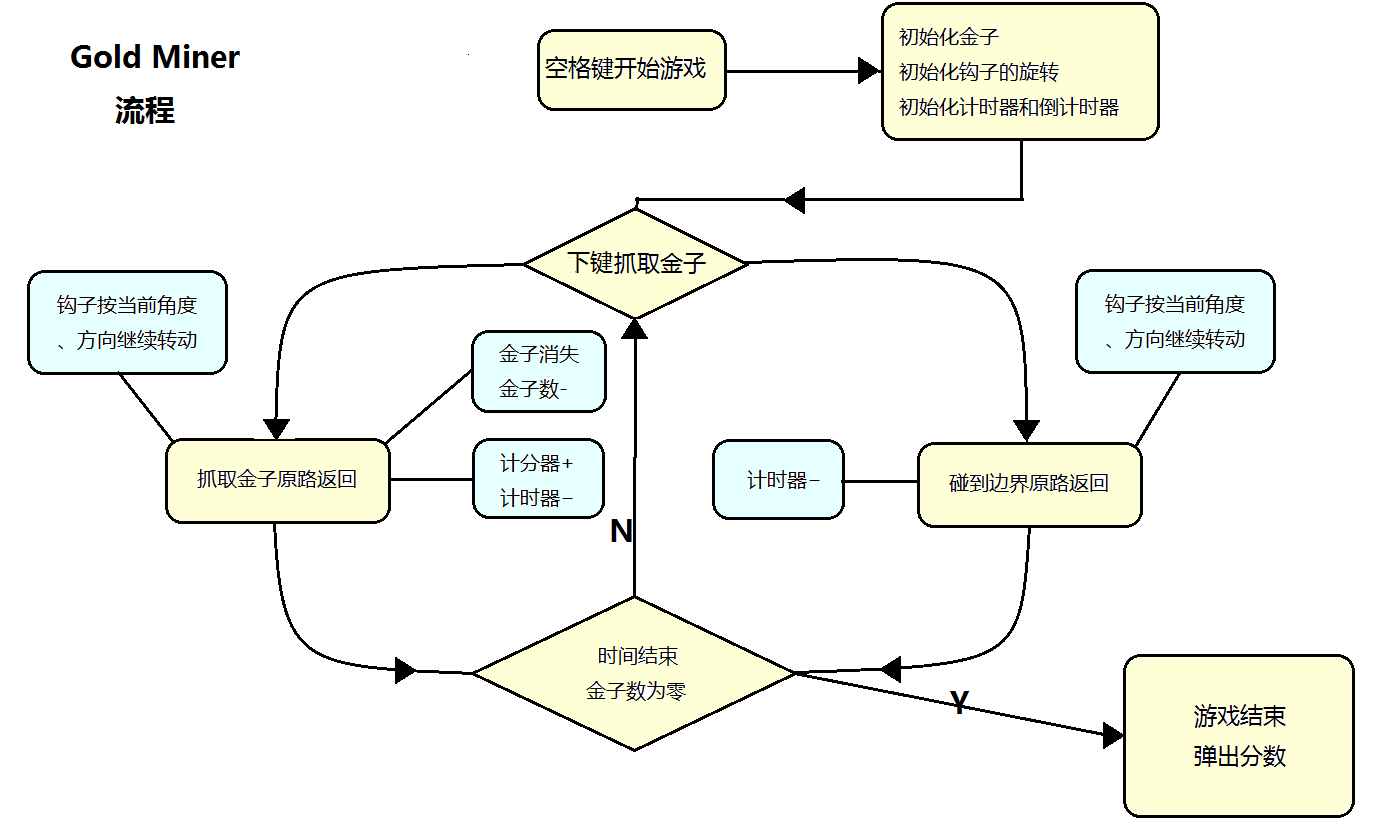
**一、游戏规则与思路设计**

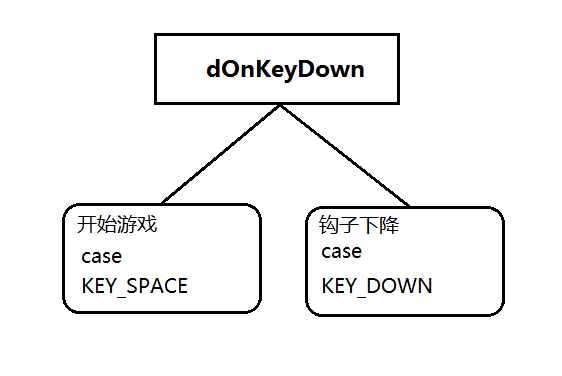
**游戏界面：**游戏界面有背景、矿工人物、钩绳、金块（均贴图）、计分器和计时器，按金块大小不同有不同的分值，定为500、100、20三种规格的金块，大小不同。

**游戏规则：**点击空格键开始游戏，计时器开始倒计时，倒计时设置为120秒，计分器初始化为0，矿工在一个定点，钩子按一定周期旋转180°，初始为由右向左旋转，180°后变为由左向右旋转，玩家点击键盘方向“下”键放钩，勾到金块则钩子原直线返回定点，金子消失，计分器按金子规格加上不等的分数；若未勾到金块则钩子触到边框返回，计时结束或金子被抓取完则游戏结束，弹出窗口，内容为”Time Over”。

**模块划分：**Gold Miner工程分为六个文件，hook.h存储变量及函数声明，gold\_and\_hook.c为函数定义，game\_operation.c为游戏主逻辑和操作，ui.c为窗口函数及界面GDI绘图，LinkList.h和 LinkList.c为用到的链表。

**二、流程图与游戏逻辑**





**三、数据结构与变量设置**

//---------宏定义---------

//定义常量

#define size1 12//2//1//小金块尺寸

#define size2 25//3//2//中金块尺寸

#define size3 60//5//4//大金块尺寸

#define num 20//金块总数

#define num1 10//小金块数量

#define num2 6//中金块数量

#define num3 4//大金块数量

#define score1 20//小金块分数

#define score2 100//中金块分数

#define score3 500//大金块分数

#define hookstartlen 50//初始钩子的绳长

#define rotation 0.0314159/3.7//每次旋转的角度（弧度制）

#define hspeed 3.6//钩子下放时的速度

#define pi 3.1//180°的弧度

//计时器ID

#define TIME\_ID1 10

#define TIME\_ID3 30

//游戏坐标

typedef struct \_GAME\_COORD

{

int x;

int y;

}GAME\_COORD, \*PGAME\_COORD;

//一个金块

typedef struct \_Gold

{

PGAME\_COORD coord;

int size;

}\*PGold;

//--------全局变量--------

//计分器，初始为零

int score;

//钩子

PGAME\_COORD hook;

//钩子移动前的位置

PGAME\_COORD exhook;

//金块（链表）

PLIST gold\_list;

//游戏边界

PGAME\_COORD boundary;

//金块边界

PGAME\_COORD goldbounary;

//钩子旋转状态 右→左为1 左→右为0 初始为1

int hooktoleft;

//钩子此时旋转与定点左右两侧边界夹角 初始为零

float angle;

//矿工所在的定点

GAME\_COORD point;

//倒计时器

int second;

//钩子移动的状态

int hookback;

int hookmove;

**四、函数与关键算法描述**

**函数：**

//---------函数声明--------

//---------边界--------

//设置边界坐标

void SetBoundary(int x, int y);

// 获得边界坐标

PGAME\_COORD GetBoundary();

//设置金块边界坐标

void SetGoldBoundary();

//---------金块--------

//生成一个新的金块 随机生成坐标

PGold CreatGold(int size);

//获得金块

PGold GetGold(int n);

//获得金块坐标

PGAME\_COORD GetGoldAt(int n);

//创建所有金块的链表

void CreatAllGolds();

//---------钩子--------

//创建钩子即初始坐标

void CreatHook();

//获得钩子坐标

PGAME\_COORD GetHook();

//钩子转动

void HookRoll();

//钩子移动

void HookMove();

//钩子回到初始点

void HookBack();

//是否抓取到金子

int CoordEqual(PGAME\_COORD hook, PGold g);

//抓取金子

int CatchGold();

//碰到边界

int CatchBoundary();

// game\_opration.c 中的接口函数

//获取得分

int GetScore();

//获取时间

int GetSecond();

//创造游戏

void CreateGame(HWND hwnd,

DWORD dwInitTimerElapse1,

DWORD dwInitTimerElapse3,

int boundary\_x,

int boundary\_y//边界

);

//钩子旋转函数

void OnRollTimer(HWND hwnd);

//倒计时函数

void OnSecondTimer(HWND hwnd);

//按键控制状态转换

void OnKeyDown(DWORD vk, DWORD hwnd);

//---------内部函数声明---------

//判断是否捞到金块

int CoordEqual(PGAME\_COORD hook, PGold g);

**关键算法：**

**1、计时器逻辑：**初始化两个计时器，第一个为控制钩子运动，第二个为控制倒计时器，第二个计时间隔设置为一秒，每秒全局变量second--；第一个计时器隔50毫秒传递一次消息，改变钩子位置并重绘界面。

**2、生成金子：**每块金子定义为结构体，包含坐标和尺寸，所有金子存储在gold\_list的链表中，游戏开始，创建链表，金子位置随机，按尺寸生成定量的每种金子，分别生成每个金子在压入链表。

**3、钩子旋转：**设置状态存储变量hooktoleft，若钩子由右向左转动为1，钩子由左向右转动为0，设置角度存储变量angle，初始化为零，初始hooktoleft为1，每获取一次计时器消息，angle增加rotation度，再计算钩子的坐标（利用三角函数），若angle大于或等于180°，则angle=0，hooktoleft=0，钩子由左向右旋转，每次增加ratation度，直至大于或等于180°，则angle=1，hooktoleft=0.

void HookRoll()

{

if (hooktoleft == 1 && angle < pi)

{

angle += rotation;

hook->x = point.x + hookstartlen\*cos(angle);

hook->y = point.y + hookstartlen\*sin(angle);

}

else if (hooktoleft == 1 && angle >= pi)

{

angle = 0;

hooktoleft = 0;

hook->x = point.x - hookstartlen;

hook->x = point.y;

}

else if (hooktoleft == 0 && angle < pi)

{

angle += rotation;

hook->x = point.x - hookstartlen\*cos(angle);

hook->y = point.y + hookstartlen\*sin(angle);

}

else if (hooktoleft == 0 && angle >=pi)

{

angle = 0;

hooktoleft = 1;

hook->x = point.x + hookstartlen;

hook->x = point.y;

}

}

**4、钩子伸长：**根据hooktoleft的状态，计时器每发送一次消息，钩子伸长hspeed长度，再次计算钩子xy坐标（利用三角函数）。

//钩子移动

void HookMove()

{

if (hooktoleft == 1 && angle < pi)

{

hook->x = (hook->x + hspeed\*cos(angle));

hook->y = (hook->y + hspeed\*sin(angle));

}

else if (hooktoleft == 1 && angle >= pi)

{

angle = 0;

hooktoleft = 0;

hook->x = point.x - hookstartlen;

hook->x = point.y;

}

else if (hooktoleft == 0 && angle < pi)

{

hook->x = (hook->x - hspeed\*cos(angle));

hook->y = (hook->y + hspeed\*sin(angle));

}

else if (hooktoleft == 0 && angle >= pi)

{

angle = 0;

hooktoleft = 1;

hook->x = point.x + hookstartlen;

hook->x = point.y;

}

}

**5、是否钩到金块：**遍历链表，根据每块金子的尺寸，判断钩子到金块中心的长度与尺寸的关系，若距离小于或等于尺寸，则钩到金块，若距离大于尺寸，则未钩到金块。

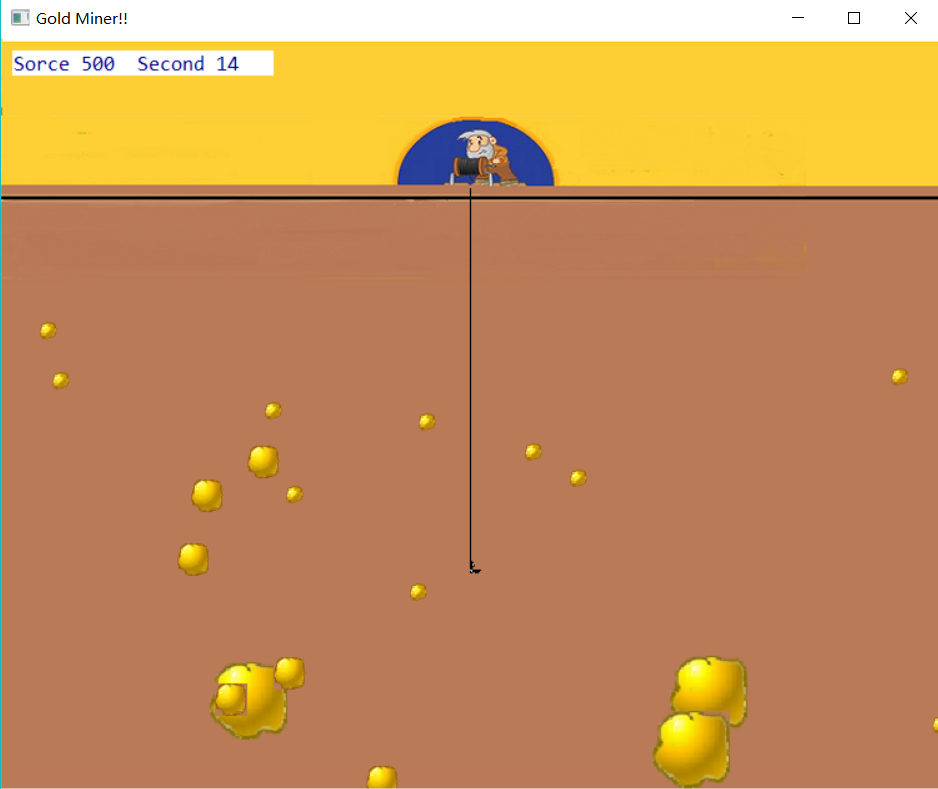
**6、金子消失：**若金块被钩到，则xy坐标均置为0，在绘图时，若xy均为零则不画该坐标的金块。

**7、钩子原路返回：**计算过程与钩子伸长同理，只是方向相反。

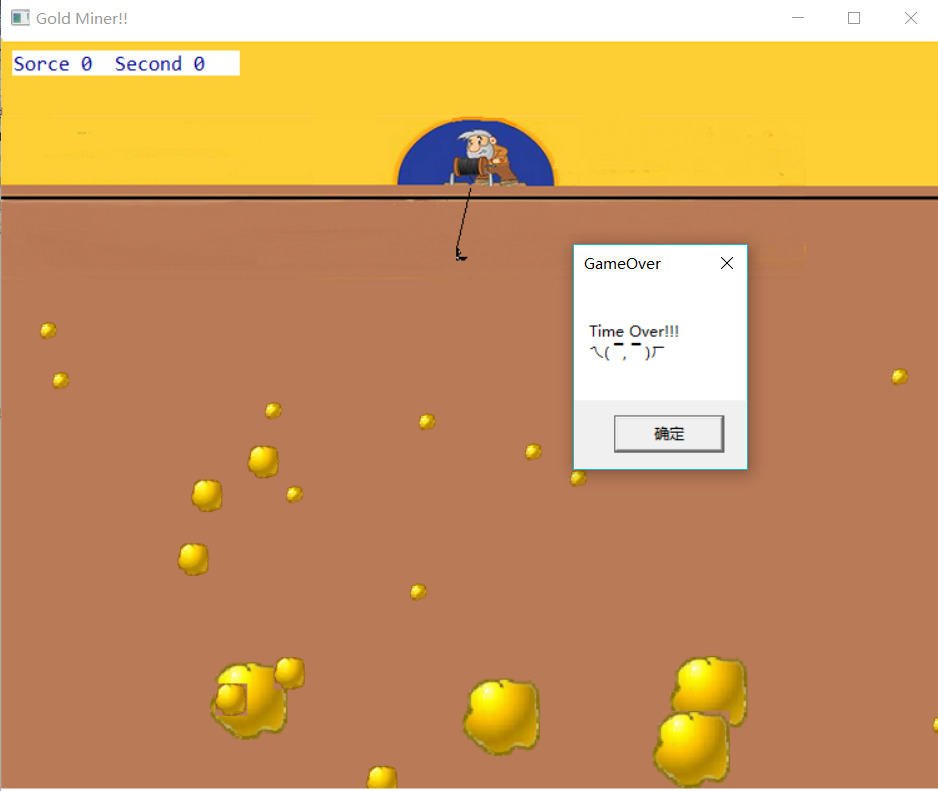
**五、程序实现效果**

钩子伸长抓取金块

****

抓取到金块，金块消失，分数增加****

钩子持续转动直至倒计时结束，弹出“Timer Over”窗口

****