GameMove(), 在游戏模式中控制蛇身移动; 在 GameMove() 中会调用 ReDrawBody() 重绘蛇身; ReDrawBody() 发送系统消息进行窗口重绘, 调用 CSnake 类中的 OnPaint() 方法进行游戏画面的重绘.

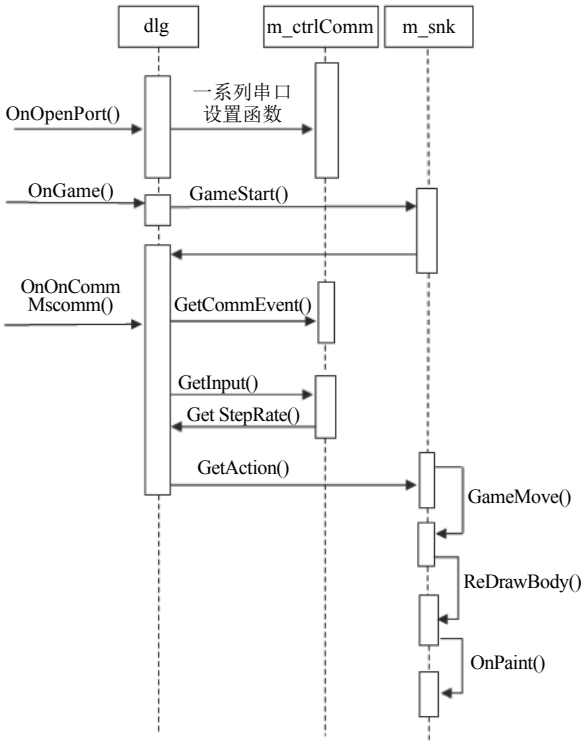

图 6 踏步机贪吃蛇游戏时序图

2.3 踏步机贪吃蛇游戏主要类的设计

2.3.1 游戏规则类

游戏规则类的判断游戏状态模块主要算法设计如下:

(1) 当蛇身每移动一步时, 就对蛇头部的坐标进行判断.

(2) 如果已经与游戏界面的边界或者和蛇身向量中的坐标重合, 说明贪吃蛇已经碰到了墙壁或者自己身体, 提示游戏结束.

(3) 设置游戏为结束状态.

2.3.2 串口类

利用 VC++6.0 中所提供的串口控件, 在工程中加载 CMSComm 类, 调用类中函数可实现串口通信. 蛇身动作函数应放在 OnOnCommMscomm1() 中调用, 即可实现踏板动作与蛇身移动的一致. 显示平均步频、显示得分语句也可放在此函数中.

2.3.3 蛇类

蛇类的设计是整个软件的关键环节, 蛇类具体包含数据、函数如下:

蛇类的头文件 CSnake.h

```
#include "Afxtempl.h" //插入模板头文件
#include "Rule.h" //插入规则类头文件
class CSnake: public CWnd
{
public:
CSnake();
public:
int m_avgRate; //平均步频
int m_nDifRate; //步频差
int m_SnkRRate; //蛇的右步频
int m_SnkLRate; //蛇的左步频
bool flagR; //右步频标志
bool flagL; //左步频标志
int m_nPreDir; //前一次动作方向
int m_nScore; //分数
bool m_bIsOver; //游戏结束
public:
//游戏模式蛇身移动
void GameMove(CPoint pt1);
//游戏模式蛇身动作
void GameAction();
BOOL GameStart(); //游戏模式开始
void InitGame2(); //游戏模式初始化
void InitFoods(); //初始化果实
void ReDrawBody(CPoint pt); //重绘蛇身
private:
CArray<CPoint, CPoint>m_body; //蛇身向量
CArray<CPoint, CPoint>m_body1; //目标蛇身向量
CPoint m_psFood; //果实
CRule rule; //规则类对象
};
```

蛇类主要完成以下功能:

(1) 果实的随机生成和显示

果实必须随机生成, 果实出现的范围不要在游戏窗口边缘, 这样可以降低游戏难度. 果实不能与蛇身重合, 即果实的坐标不能与蛇身的坐标相同. Snake.cpp 中 void CSnake::InitFoods() 初始化果实, 如图 7.

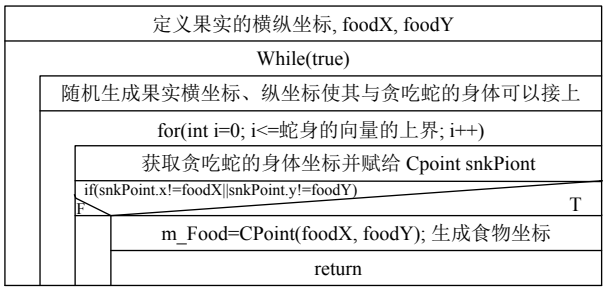

图 7 果实生成程序 N-S 图

(2) 贪吃蛇游戏功能

游戏功能是让玩家感受到健身与游戏的结合, 从游戏中得到有效的锻炼. 先在游戏中探索到自己的有氧运动强度范围, 可记录下心率和体力感知度[8](对照 RPE 表获得) 对应的上下限步频, 然后在游戏中设定上下限步频, 平均步频数值决定蛇身颜色的变化, 玩家低于下限步频踏板时, 蛇身变蓝, 高于上限步频, 蛇身变红, 在上下步频之间蛇身是绿色. 玩家在游戏中让蛇身保持绿色就是在一定步频范围内踏板, 也就是在做有氧踏板运动, 如图 8 所示. 当玩家经过一段时间的游戏, 如果感到踏板比较轻松, 而且比较容易得到高分, 就可以设定更高的上下限步频, 这样通过再一轮的踏步游戏, 便可以提高玩家的有氧运动能力.

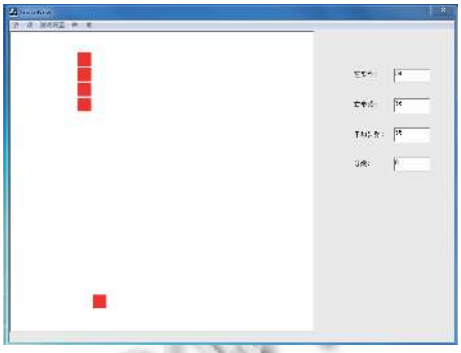

图 8 游戏界面图

蛇身的方向由左、右步频差来控制, 例如可以规定步频差等于右步频减去左步频:

$$m\_nDifRate = m\_SnkRRate - m\_SnkLRate \quad (1)$$

步频差阈值可以自由设定, 数值小, 拐弯就容易. 例如可以规定阈值为 10, 当步频差 (*m_nDifRate*) 大于 10, 设定蛇身向左拐, *m_nDifRate*<–10 时, 蛇身向右拐, –10 <= *m_nDifRate*<=10 时蛇身保持原方向前进. 在编程时要增加一个变量保存先前蛇身移动方向. 蛇身移动的快慢由平均步频决定, 如式 (1), 具体移动步数由式 (2) 计算得出, 蛇身每步即移动一个像素, 移动多少步数由平均步频、*m*、*n* 决定, *m* 和 *n* 可根据蛇身大小、游戏窗口的大小、游戏的节奏自主设定.

$$m\_avgRate=[(m\_SnkRRate+m\_SnkLRate)/2] \quad (2)$$

$$steps=[m\_avgRate/m]+n \quad (3)$$

Snake.cpp 中 void CSnake::GameAction() 是蛇身动作函数, 它控制游戏模式中蛇身移动, 详细设计过程如图 9 所示. 其中 for 循环语句为 for(int i=0; i<移动步数公式所得出值; i++), 表示蛇身移动步数.

3 结语

踏步机游戏设计是否成功关键在于它的交互性, 实现良好交互性关键是游戏的操控性, 这是从玩家操控踏步机器进而控制贪吃蛇的移动体现出来. 操控性是指游戏的易用性和操作的准确性, 准确性是指, 按照传感器以及游戏软件所设定的规格所能达到的操作结果与玩家实际操作游戏时所预判的结果, 在程度上的差异. 操控性能的好坏评价, 一方面在于设计, 另一方面在于玩家熟练掌握技巧所需要的时间以及他们的主观体验. 本游戏易用性方面主要表现在玩家在做拐弯动作时有点难度, 玩家必须一只脚踏板快, 另一只踏板慢, 这样才能产生步频差较大, 形成蛇的转向. 本文的建议是, 在做拐弯动作时把一只踏板停住, 快速踏另一只踏板, 这样就容易形成一个步频数值就会远大于另一个步频. 测试者要经过一些练习才能掌握好.

基于运动器材作为游戏的输入设备具有稳定性好、精确度高、成本低, 从而设计的电子游戏具有操作感、精准感较强, 模拟运动体验好等特点. 由于它们本身就是健身器材, 因此更容易设计出健身游戏软件与之配套. 基于体感游戏手柄设计体育类游戏, 或者基于 Kinect 摄像头, 通过深度识别和人体骨骼追踪技术, 实现人机交互, 设计的游戏, 易产生误差, 游戏操作起来有时会产生延迟和操作失败, 玩家的游戏体验与该体育项目的实际体验相差较大. 本文所用的踏步机为市场上普通款式, 传感器的安装并不需要改变对原踏步机结构, 具有设计简单, 安装方便, 成本低廉等优点, 有较好的市场前景. 就普适性而言, 本文的传感器装置的设计思想也可以运用在其他运动器械上, 例如具有双桨的划船机、椭圆机等, 只要有两个运动部件, 类似于踏步机, 都适用以上原理, 即计算左右运动频率, 利用左右频率差和平均频率来驱动游戏主角运动.

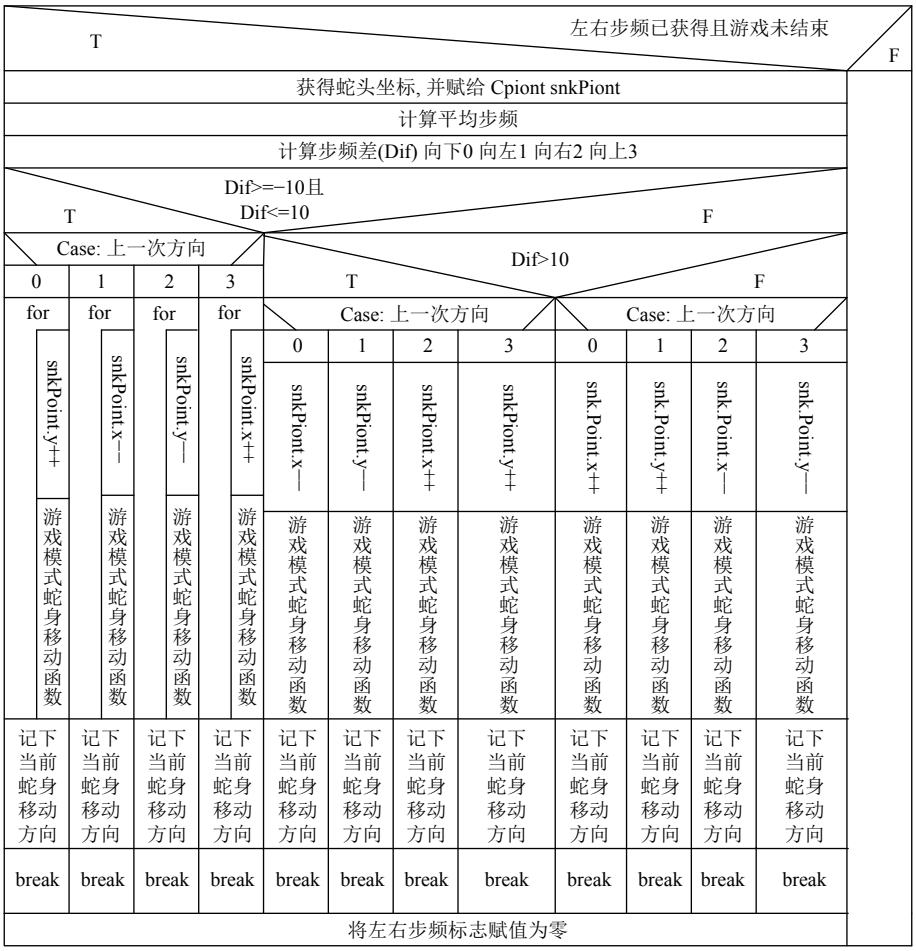

图 9 蛇身动作 N-S 图

参考文献

1 耿彬. 跑步机体感游戏的设计与实现[硕士学位论文]. 长沙: 湖南大学, 2010.

2 李波. 健身单车体感游戏设计与实现[硕士学位论文]. 长沙: 湖南大学, 2010.

3 张诗潮, 钱冬明. 体感技术现状和发展研究. 华东师范大学学报 (自然科学版), 2014, (2): 41–49, 126.

4 黄静, 郎波, 张志稳, 等. 基于 Kinect 跑步机系统. 计算机系统应用, 2016, 25(9): 113–118.

5 郭天祥. 新概念 51 单片机 C 语言教程-入门、提高、开发拓展全攻略. 北京: 电子工业出版社, 2009. 74–76.

6 何青. 游戏程序设计教程. 北京: 人民邮电出版社, 2011. 17–20.

7 王浩. Visaul C++游戏开发经典案例详解. 北京: 清华大学出版社, 2010. 263–302.

8 曲绵域, 于长隆. 实用运动医学. 北京: 北京大学医学出版社, 2003: 78.