电子科技大学 计算机 学院

**实 验 报 告**

课程名称 C++程序设计

实验题目  精灵游戏

难度级别 4

提交时间 2021.10.14

姓 名 孟巳茗

学 号 2020080601010

# 软件说明

## 游戏使用说明

基本功能：方向键←↑↓→移动

打开方式：打开Sprite.exe文件开始游戏

使用方法：操作猫追捕鸭子和老鼠，躲避狗。

老鼠们虽然跑得很快，但它们很笨，只能直线移动，遇到边界也只会走回头路，抓住1只可得1分。在它们的必经之路上等待它们！

鸭子们沿直线移动......直到它们发现猫为止。当鸭子接近猫时，它们会加速躲避猫，逃到一定距离外才会慢下来向回路跑去。注意！鸭子会穿过边界从另一边出现，不过别担心，猫也可以，而且猫跑得更快。抓住它们，1只鸭子值3分！

小心，猫虽然很快，但是不太灵敏，别撞到狗身上了。狗速度较慢，但它们会直直地向猫追去。猫打不过狗，但是狗并非毫无弱点，多只狗会相互排斥，把它们引到一起，它们就顾不上追你了。总之，别碰上狗，它们会吃掉你一半的分。

## 源码编译说明

方式1：GCC

cmd中输入：

gcc -c \*.c

g++ -c \*.cpp

g++ \*.o -lgdi32 -lole32 -loleaut32 -luuid -lwinmm -lmsimg32 -o Sprite.exe

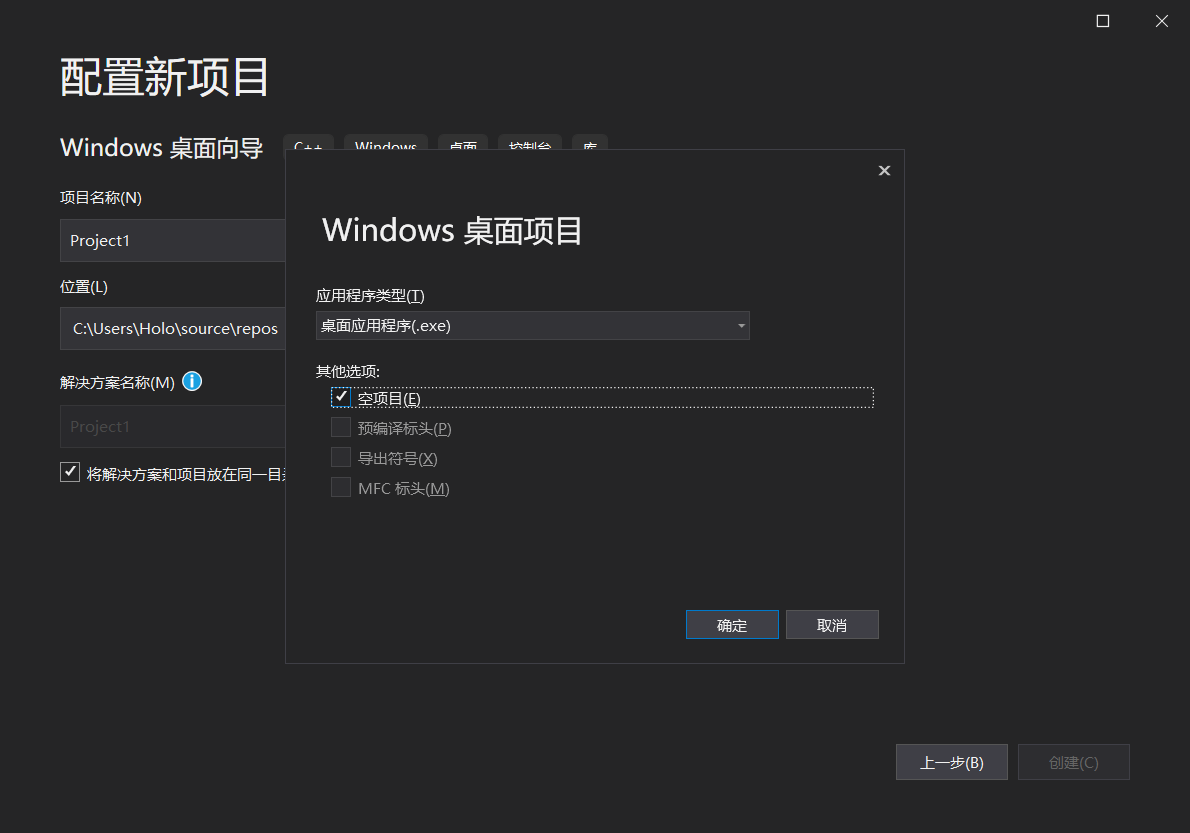
点击文件目录下的Sprite.exe开始游戏

方式2：Visual Studio

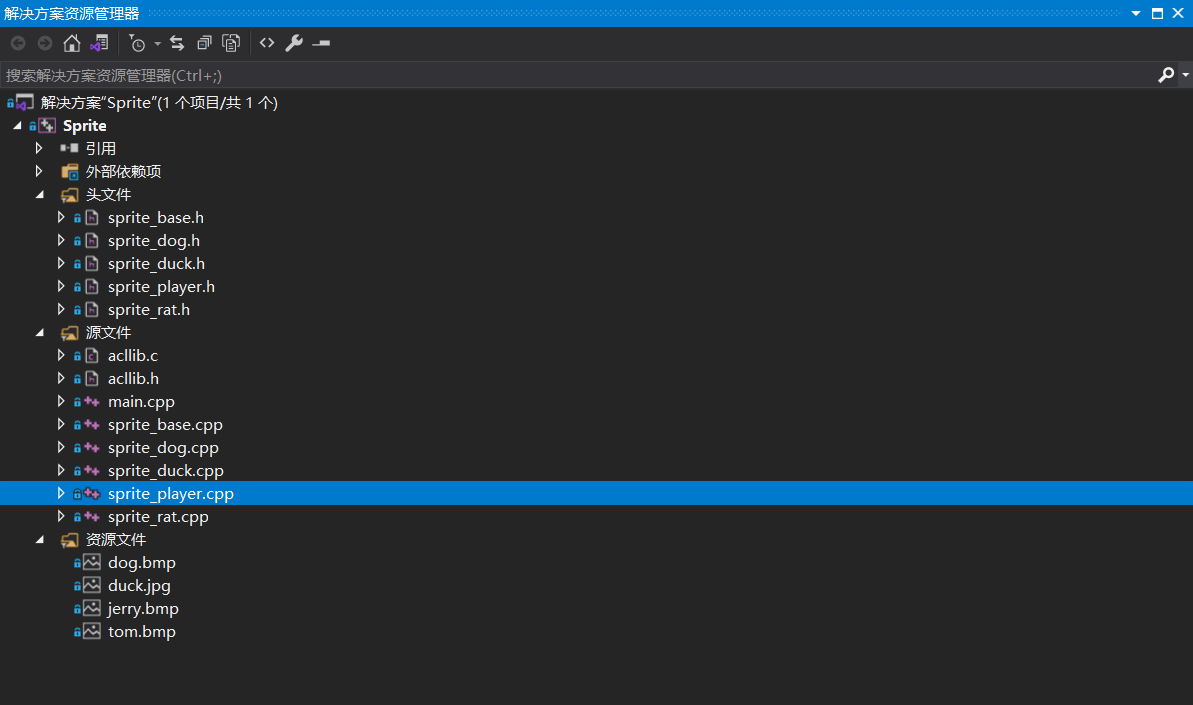
1.选择Windows桌面向导



2.选择桌面应用程序(.exe) 点击空项目(E)



3.将文件如图导入



4.选择Release X64 点击本地Windows调试器，等待编译完成



5.进入.\x64\Release，将Sprite.exe复制到.\(资源文件目录)，点击开始游戏

# 系统设计

每种动物都有相似的行为属性，所以用基类

类 SpriteBase

属性survive width height speed img pos

行为move drawSprite initilize

抽象动物共同的行为和属性

游戏人物有猫，老鼠，鸭子，狗。故构造4个SpriteBase的派生类：

SpritePlayer(Cat) SpriteRat SpriteDuck SpriteDog

分别依照4种人物的移动逻辑，编写4个类自己的move函数，同时，不同人物有自己的特点，不属于基类行为属性，故分别编写成员和函数描述其特点：

>类SpritePlayer为玩家操纵人物，涉及碰撞行为，编写描述行为。同时玩家有积分，设置整形成员记录分数

>类SpriteRat被抓有积分，设置整型成员记录分数

>类SpriteDuck涉及逃跑行为，检查是否在危险范围内并做出相应反应。同时鸭子被抓有积分，设置整型成员表示分数

>类SpriteDog描述人物狗会相互排斥，检查范围并作出反应

不同类的声明和实现分别编写在不同文件中

# 程序实现

bool SpritePlayer::collision(const SpriteBase& sprite):

玩家碰撞逻辑

X重叠：

x < sprite.x 且 x + width > sprite.x

或

x > sprite.x 且x < sprite.x + sprite.width

Y重叠：

y < sprite.y 且y + width > sprite.y

或

y > sprite.y 且y < sprite.y + sprite.width

X重叠且Y重叠，判断为true，否则为false

void SpriteDuck或SpriteDog::danger(const SpriteBase\* sprite):

鸭子逃跑逻辑 和 狗躲避逻辑

dist = (x1 - x2)^2 + (y1 - y2)^2;

limit = (h1 + h2)^2 + (w1 + w2)^2;

if dist < limit / 2 且 not escape :

escape = true

speed \*= 2 加速

if dist > limit \* 2 且 escape :

escape = false

speed /= -2 变向减速

move逻辑：

直线：

追/避：

main.cpp：

参数：

#define HEIGHT 800 窗口高度

#define WIDTH 1200 窗口宽度

const int baseSpeed = 10 基础速度

const int spriteRatNum = 9 老鼠数量

const int spriteDuckNum = 6 鸭子数量

const int spriteDogNum = 3 狗数量

函数：

void createSprite():

检测人物个体是否survive

是则continue，不做更改

否则initialize(…)，重新生成

void paintSprite():

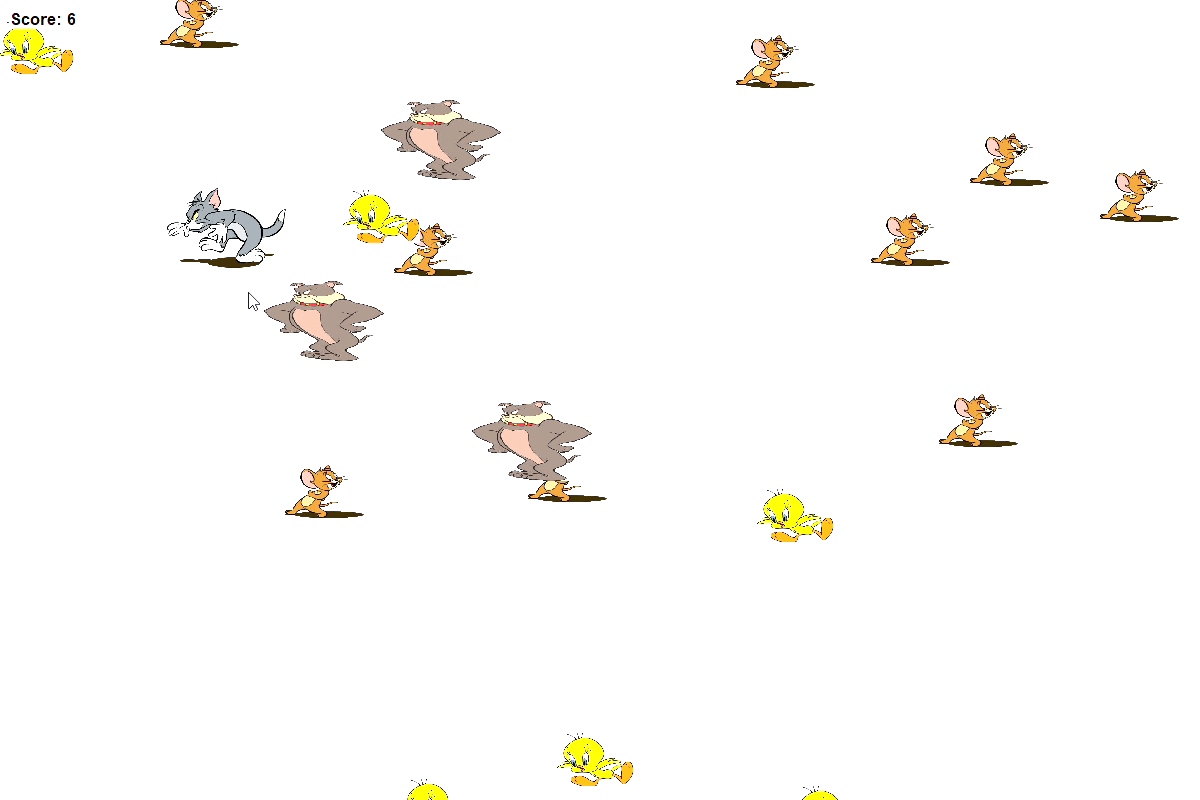
遍历所有人物个体调用基类drawSprite()行为

void spriteMove(int key)：

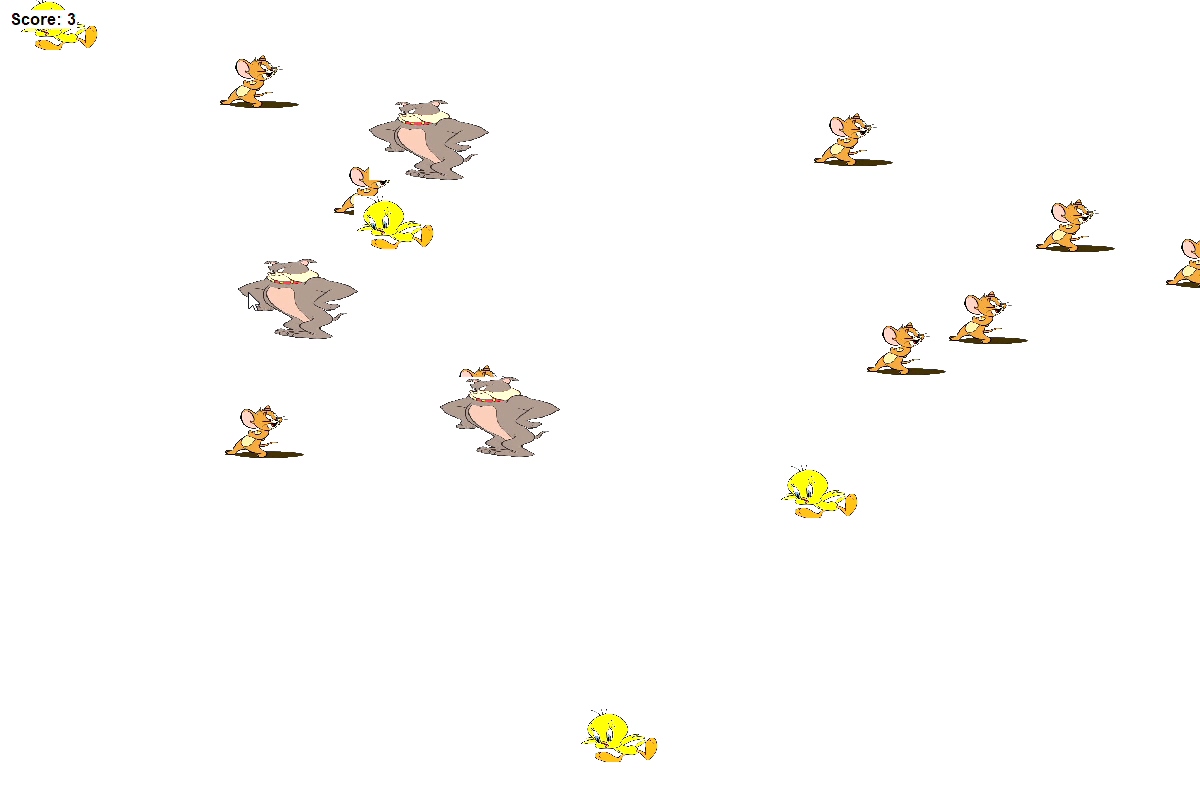
遍历所有人物个体调用move(…)行为

# 测试报告

1.狗靠近猫



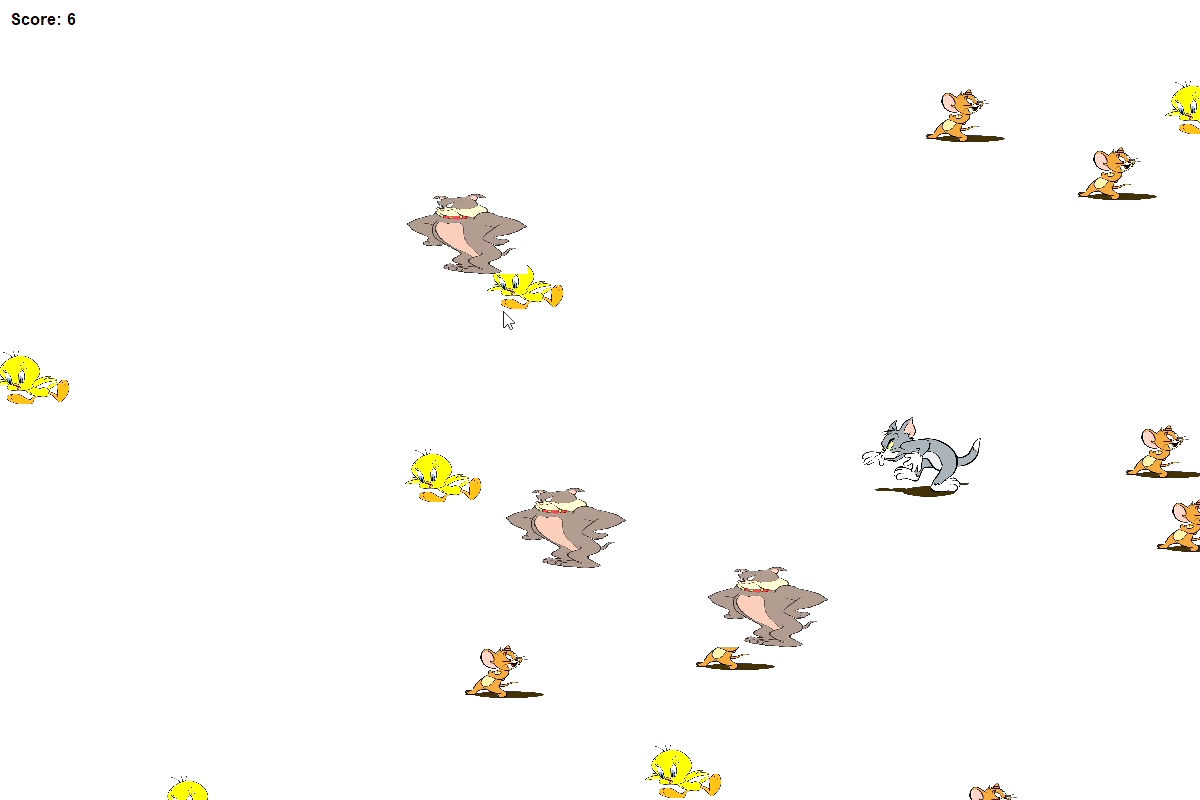
2.狗捕捉猫，猫分数减半



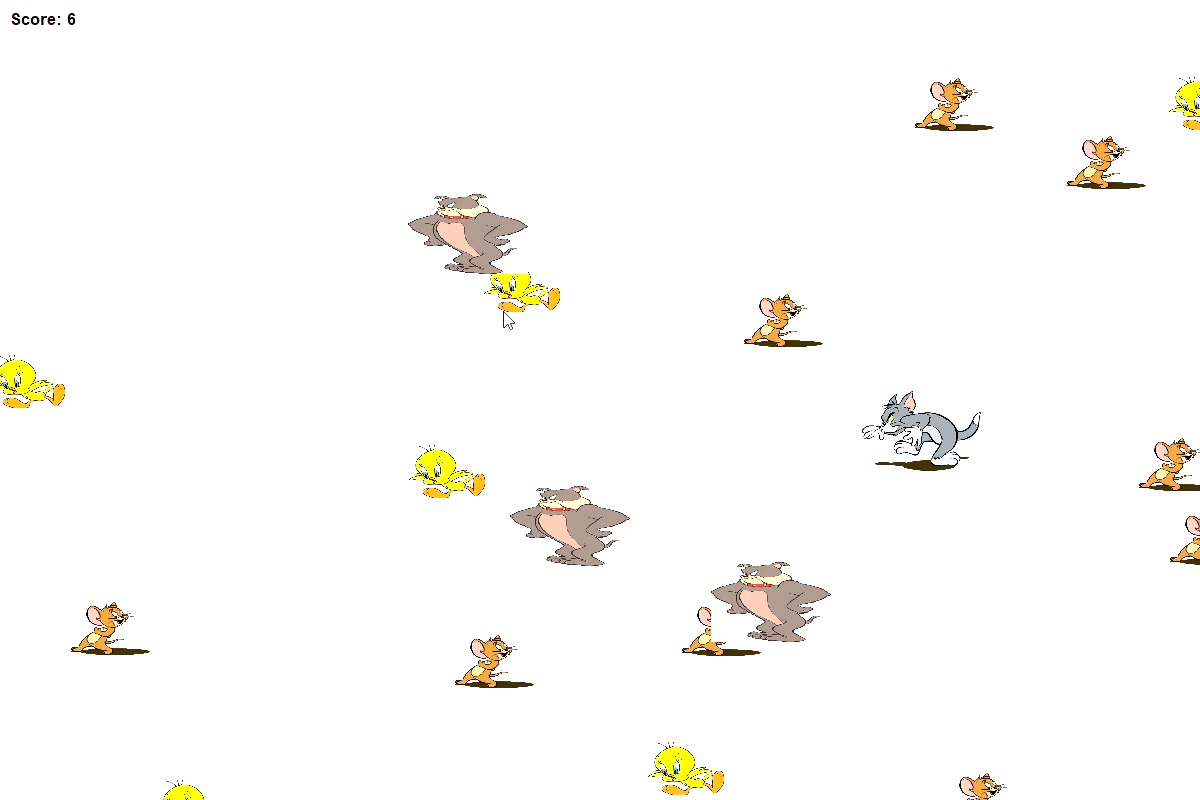
3.猫靠近老鼠



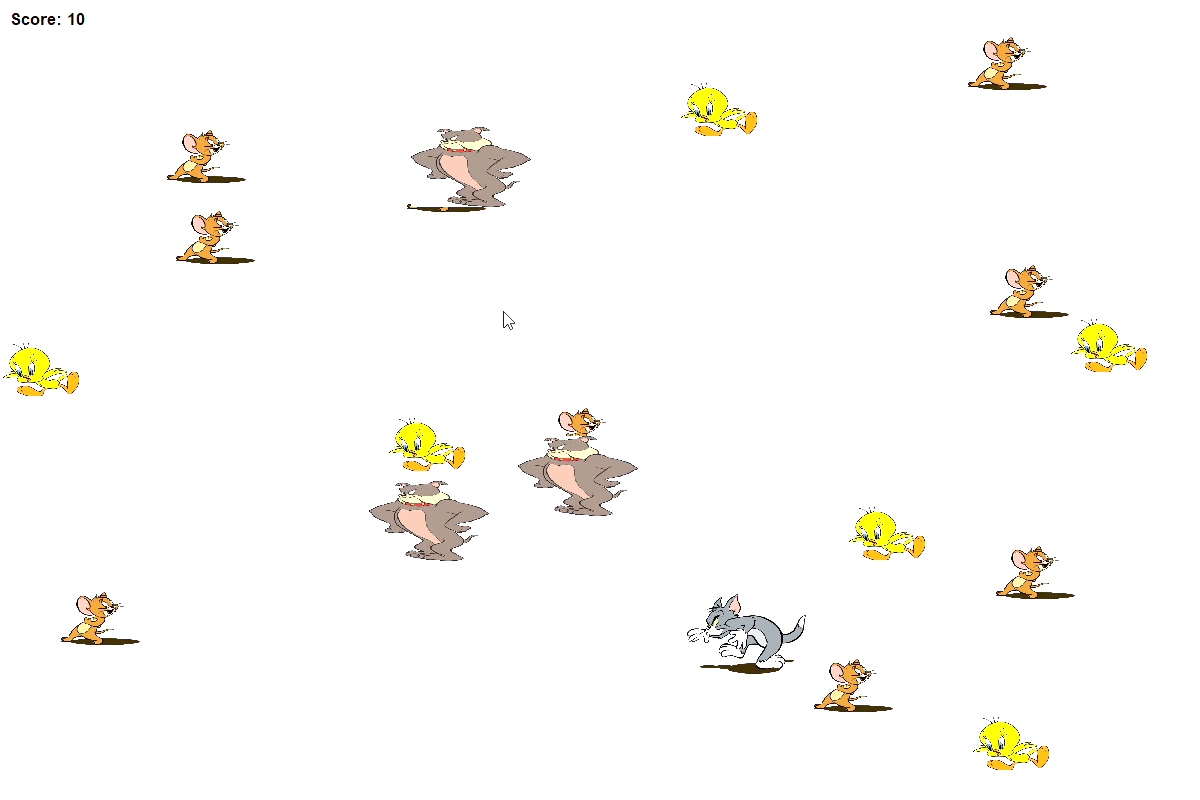
4.猫抓捕到2只老鼠，分数加2



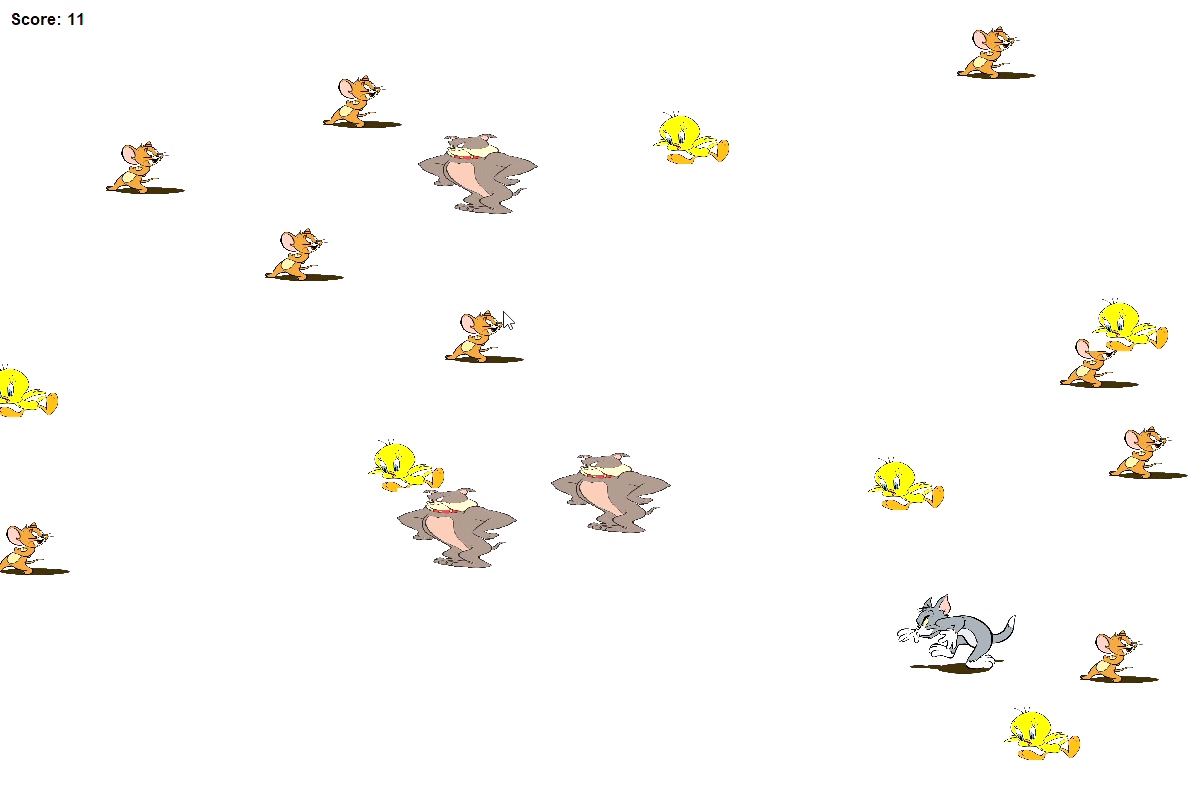
5.生成了2只新的老鼠



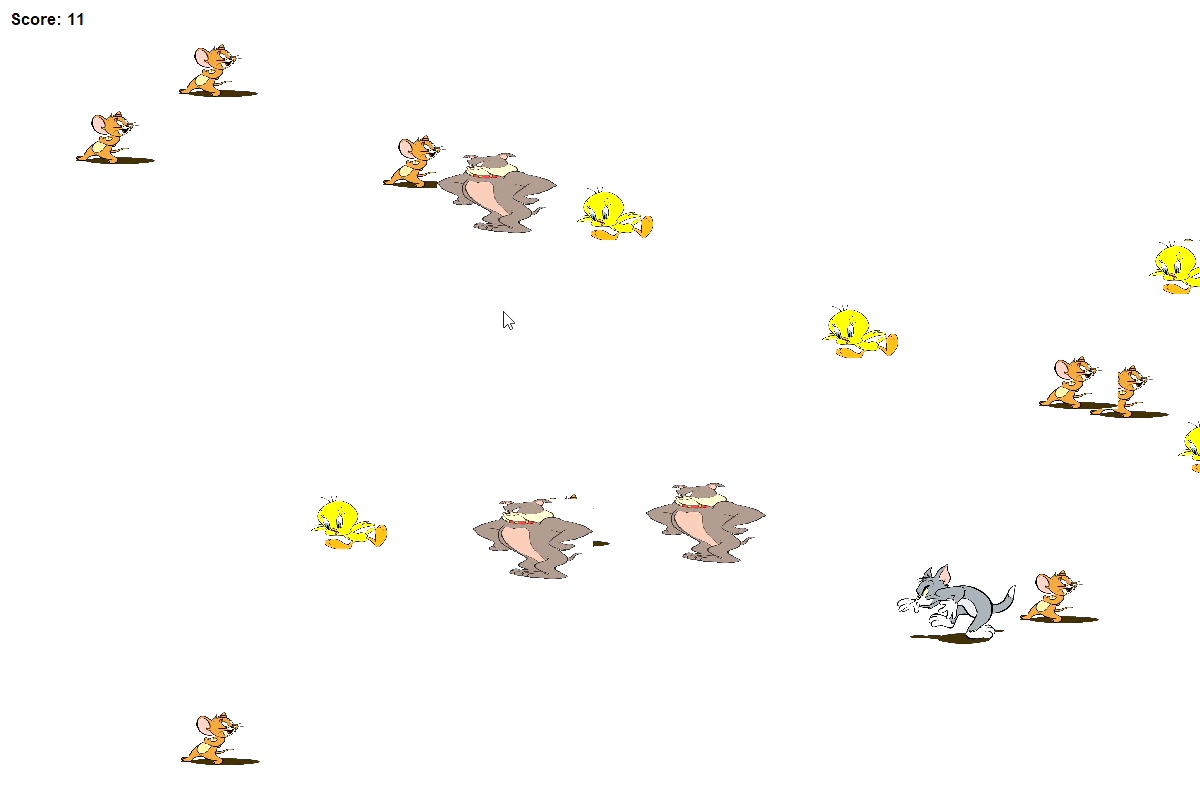
6.猫靠近鸭子



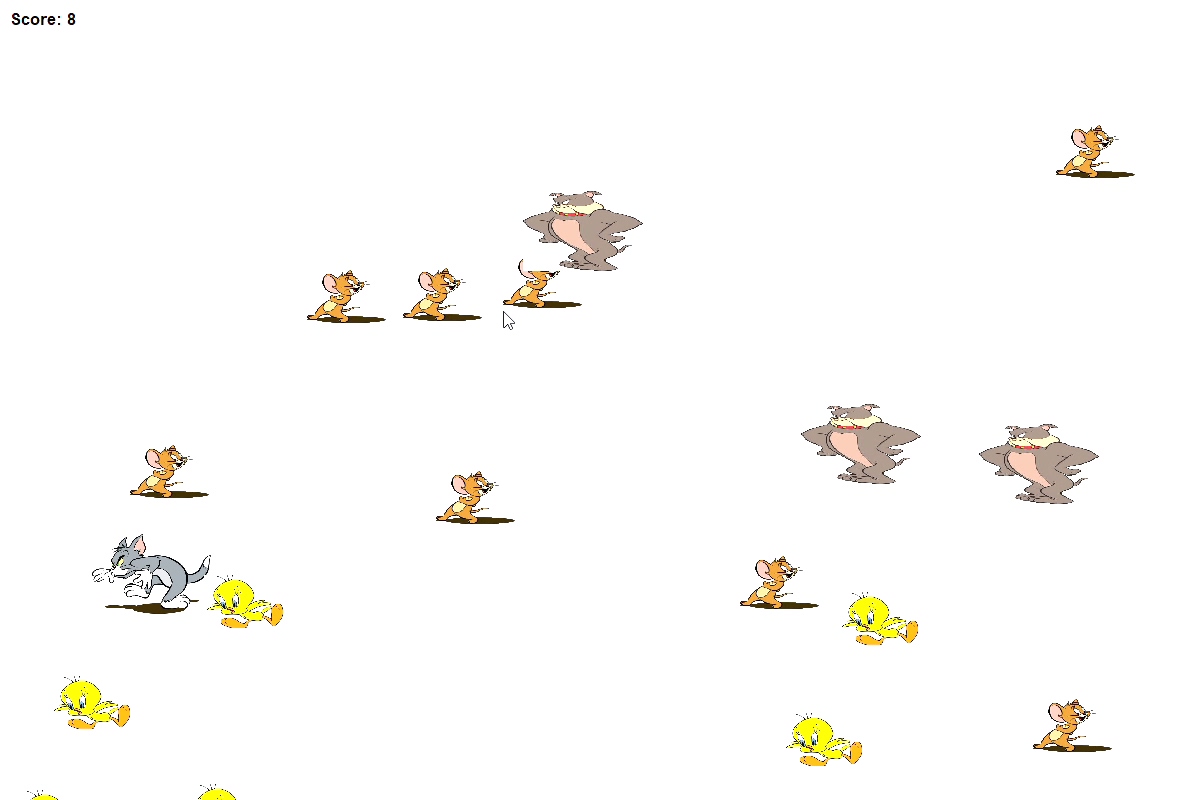
7.鸭子逃跑



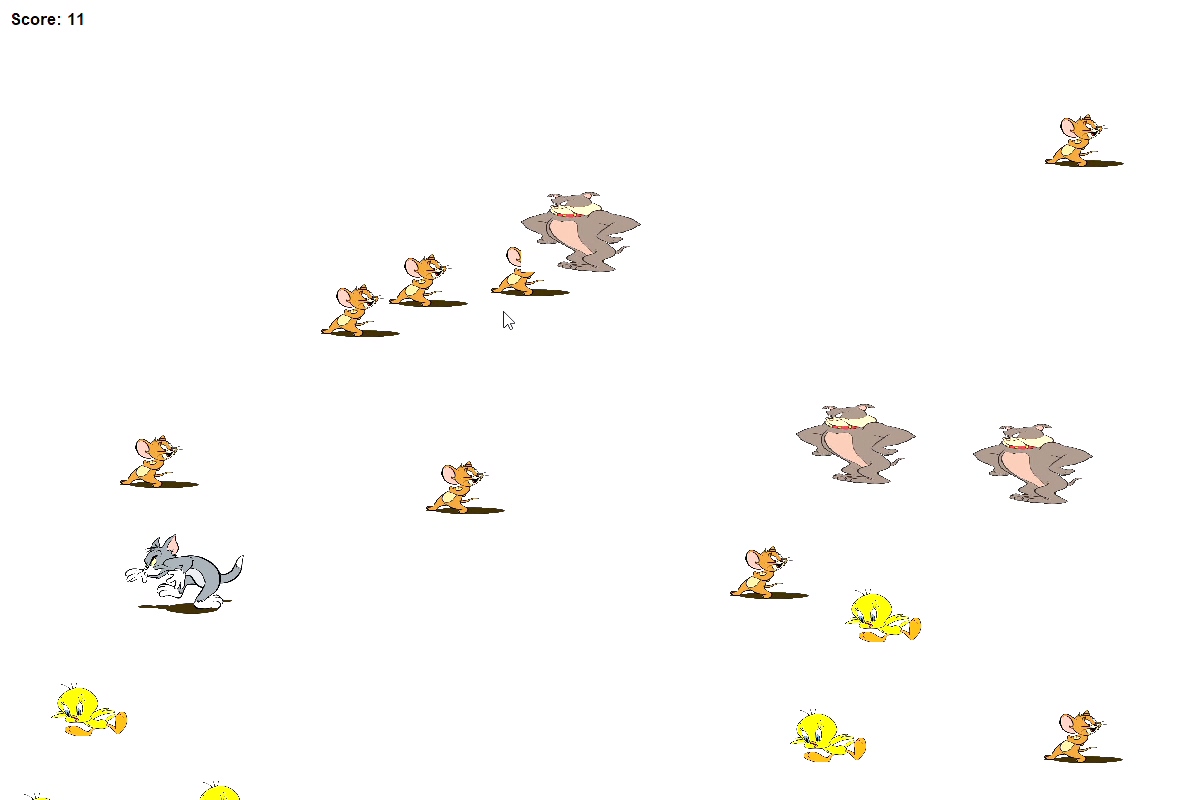
8.鸭子逃到安全范围，改变方向前行



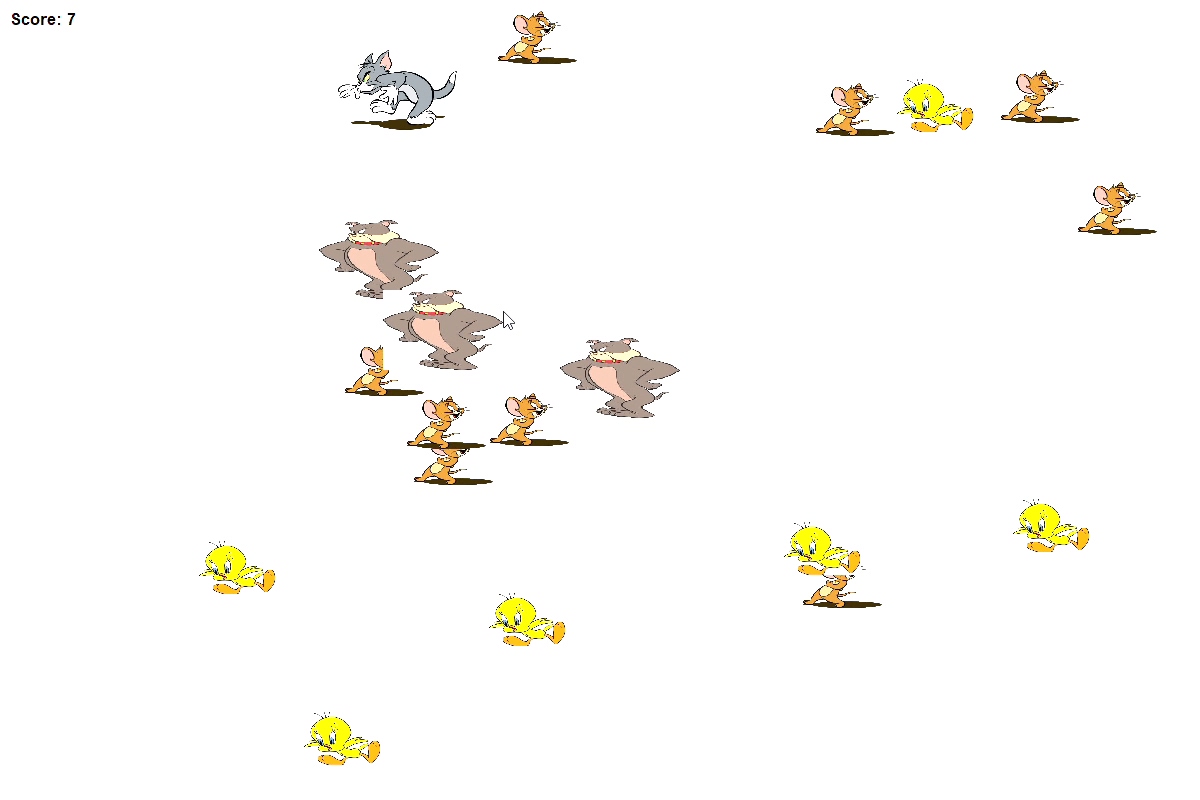
9.猫追捕鸭子



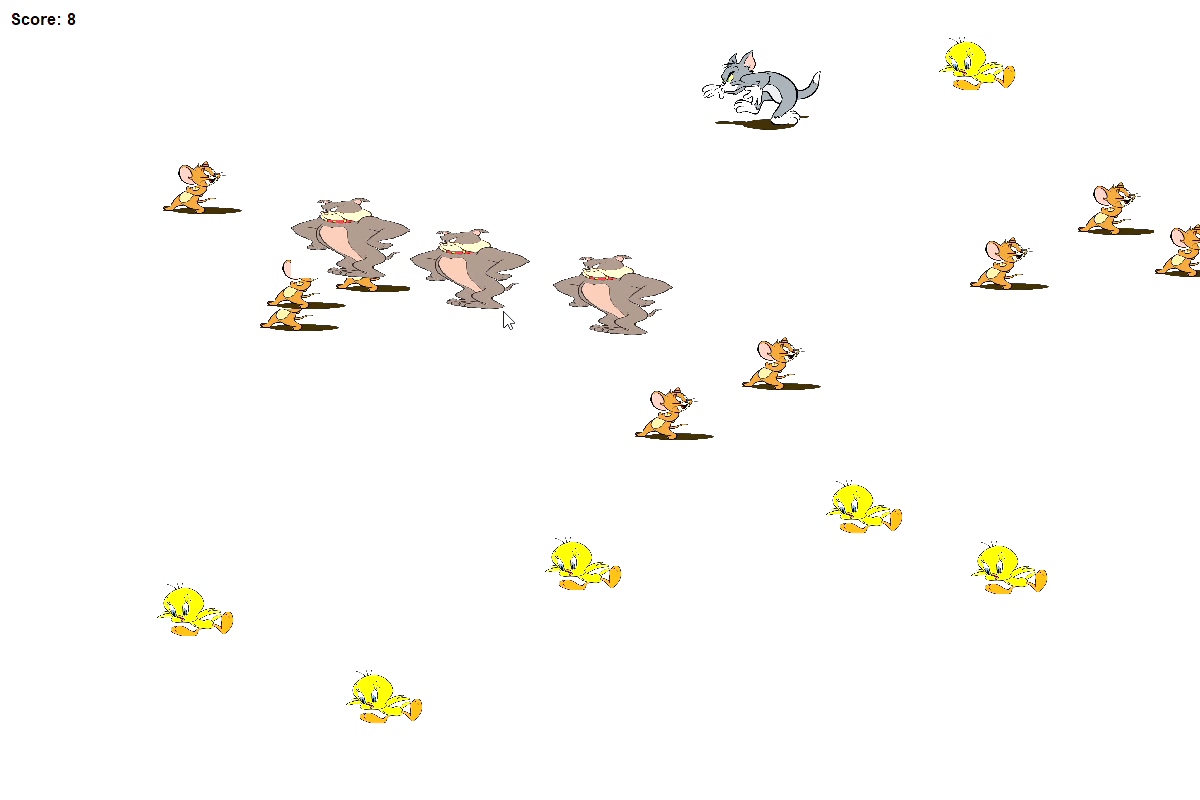
10.猫捕捉鸭子，加3分



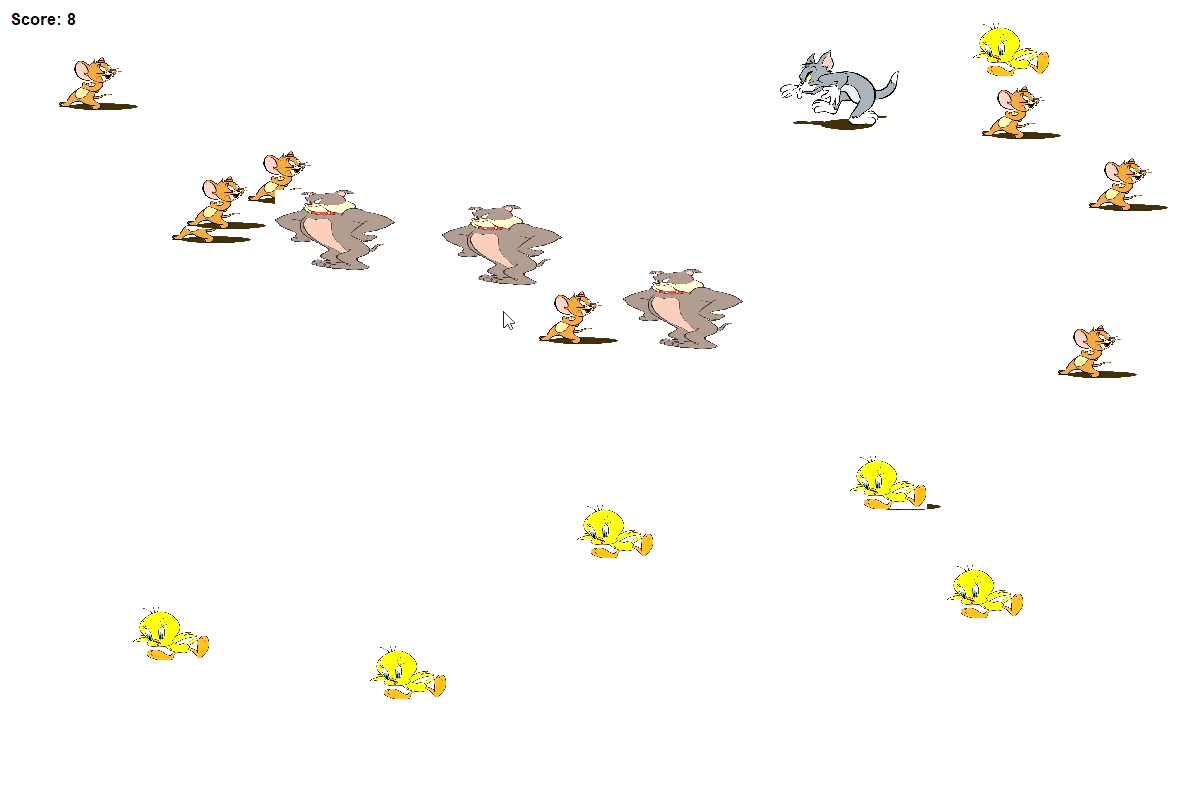
11.狗追逐猫



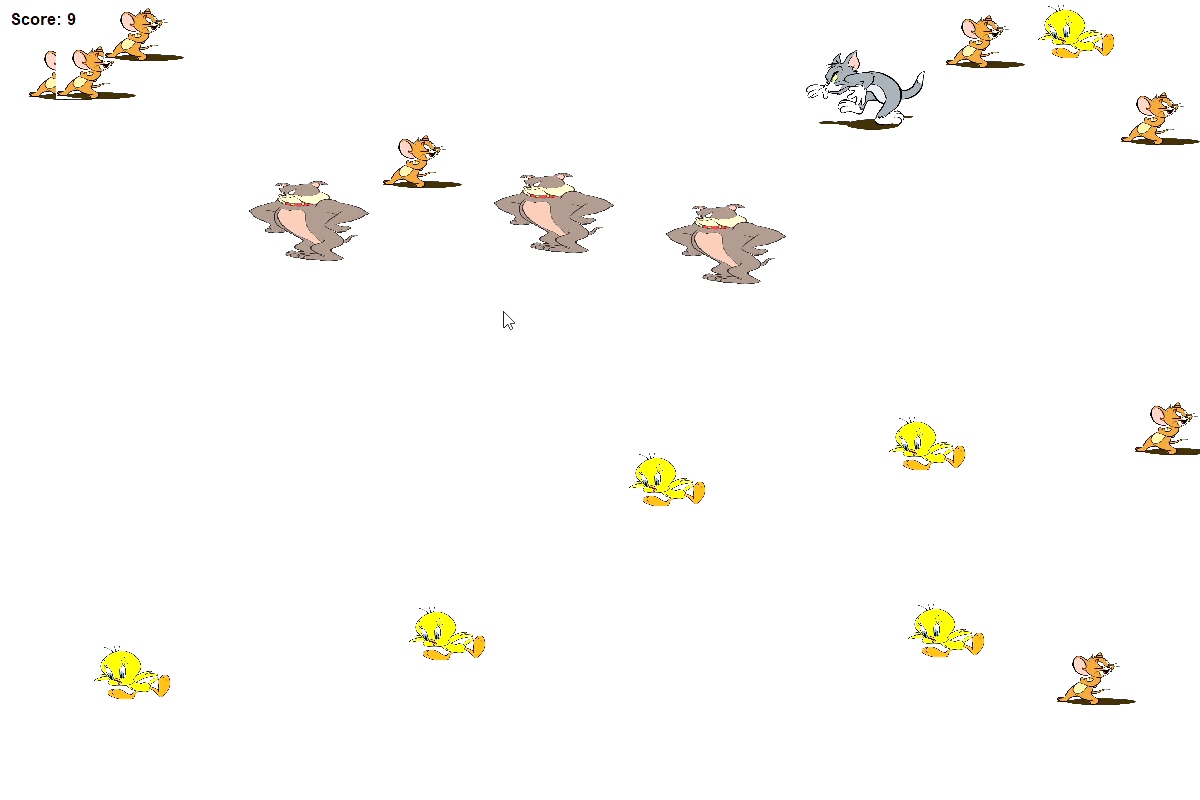
12.狗相互靠近



13.狗相互躲避



14.狗间隔达到安全范围，继续追逐猫



# 实验心得

问题及解决措施：

1. 不熟悉acllib库的使用——查看acllib库的文档，观察分析示范代码
2. 无法将各个派生类一起存放管理——尝试建立数组存放spriteBase类，即基类，但后续派生类函数调用上会出现问题，如spriteRat类数组存放成员除开[0]成员外，其余成员存放的move函数指针在initiize()后会改变并指向非法内存区域，导致move函数调用时程序崩溃——后续放弃使用数组存放不同派生类，该将具体实现全部写在单一函数中，保证后续添加游戏人物时候只需要改变特定几个函数，降低了程序耦合性
3. 编写鸭子逃逸和狗躲避的实现时，由于库精度，使用“==”检测条件判断距离容易出错，导致人物运行紊乱——改用“>”，“<”作为距离判断条件，同时为spriteDuck类和spriteDog类添加bool型成员变量escape描述当前行为，并以此作为另一判断条件
4. 人物碰撞模型已经接触但SpritePlayer::collision(const SpriteBase& sprite)常常判断为false——在先x后y判断基础上和加上先y后x重复判断人物是否接触
5. 游戏人物速度随机性较低，但设置随机值过大会导致移动过快或过慢——添加baseSpeed保证基础移速，生成speed时利用上一代的speed

存在问题：

1. 狗的移动逻辑(追逐猫或者躲避狗)需要优化。当狗躲避狗到一定范围外后向猫方向前进时候如果仍存在之前的狗，该狗容易在两个条件边界之间容易不断来回——暂时解决思路：为狗设置追逐猫的范围，并添加躲避和追逐以外的行动逻辑
2. 猫的移动操作较为僵硬，只能按住单一移动方向按键，无法同时按住多方向键做出斜向移动，同时长按和切换按键反应缓慢——应该是acllib库本身缺陷，难以解决

学习收获：

加深了面向对象编程的思想，熟悉了封装、继承、抽象等特性的使用，了解了更多有关基类，派生类，以及虚函数，虚表等的相关知识。