JAVA语言程序设计A卷参考答案

（**86141-2**）

一、名词解释**（每题2分，共5小题，总分10分）**

1. 面向对象：面向[对象](http://baike.baidu.com/view/2387.htm)(Object Oriented,OO)是[软件开发方法](http://baike.baidu.com/view/1471861.htm" \t "_blank)，是一种对现实世界理解和抽象的方法，是计算机编程技术发展到一定阶段后的产物。

2. 类属性：在Java中，类的属性并非指的是类中的成员变量，属性是一个或两个代码块，表示一个get访问器和/或一个[set](http://baike.baidu.com/view/127838.htm" \t "_blank)访问器。当读取属性时，执行 get 访问器的代码块；当向属性分配一个新值时，执行 set 访问器的代码块。不具有 set 访问器的属性被视为只读属性。不具有 get 访问器的属性被视为只写属性。同时具有这两个访问器的属性是读写属性。

3.抽象方法： java中的抽象方法就是以abstract修饰的方法，这种方法只声明返回的数据类型、方法名称和所需的参数，没有方法体，也就是说抽象方法只需要声明而不需要实现，并且抽象方法只能定义在抽象类中。

4.端口： 端口可以认为是设备与外界通讯交流的出口。端口可分为虚拟端口和物理端口，其中虚拟端口指计算机内部或交换机路由器内的端口，不可见。物理端口又称为接口，是可见端口。

5.向量(Vector)是java.util类包提供的一个工具类。它是允许不同类型元素共存的变长数组,Vector实现了变长数组。

二、单选**（每题1分，共20小题，总分20分）**

01-05： AADBC 06-10：ADCAD

11-15： CCDBA 16-20：BDBCA

三、多选**（每题2分，共10小题，总分20分）**

01 BCD 02 BCD 03 ACD 04 BC 05 ACD

1. ACD 07 ACD 08 ABC 09 BCD 10 ABD
2. 写程序的运行结果**（每题5分，共4小题，总分20分）**

1. two

two

one

one

2.14

3.x=3

y=2

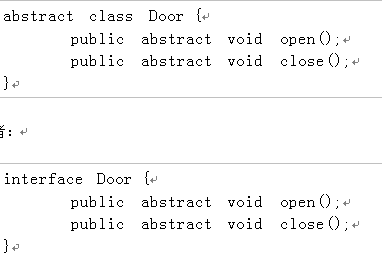
y=6

4. 20182222

五、编程题（略）**（共3大题，每题10分，总分30分）**

1.略

2.



但是现在如果我们需要门具有报警alarm( )的功能，那么该如何实现？下面提供两种思路：

1）将这三个功能都放在抽象类里面，但是这样一来所有继承于这个抽象类的子类都具备了报警功能，但是有的门并不一定具备报警功能；

2）将这三个功能都放在接口里面，需要用到报警功能的类就需要实现这个接口中的open( )和close( )，也许这个类根本就不具备open( )和close( )这两个功能，比如火灾报警器。

从这里可以看出， Door的open() 、close()和alarm()根本就属于两个不同范畴内的行为，open()和close()属于门本身固有的行为特性，而alarm()属于延伸的附加行为。因此最好的解决办法是单独将报警设计为一个接口，包含alarm()行为,Door设计为单独的一个抽象类，包含open和close两种行为。再设计一个报警门继承Door类和实现Alarm接口。

interface Alram

{    void alarm();}

 abstract class Door

{    void open();

     void close();}

class AlarmDoor extends Door implements Alarm {

    void oepn() {      //....    }

    void close() {      //....    }

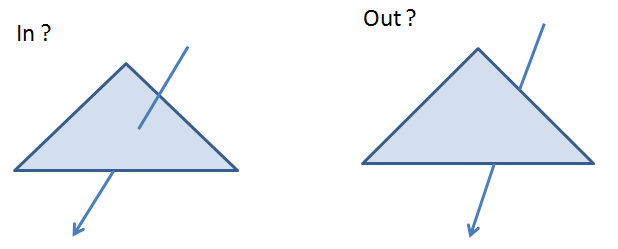
    void alarm() {      //....    }

* **}**

3.

# 概述

射线和三角形的相交检测是游戏程序设计中一个常见的问题，最典型的应用就是拾取(Picking)，本文介绍一个最常见的方法，这个方法也是DirectX中采用的方法，该方法速度快，而且存储空间少。先讲述理论，然后给出对应的代码实现。



# 理论部分

## 一个直观的方法

我想大多数人在看到这个问题时，可能都会想到一个简单而直观的方法：首先判断射线是否与三角形所在的平面相交，如果相交，再判断交点是否在三角形内。

[判断射线是否与平面相交](http://www.cnblogs.com/graphics/archive/2009/10/17/1585281.html)

[判断点是否在三角形内](http://www.cnblogs.com/graphics/archive/2010/08/05/1793393.html)

但是，上面的方法效率并不很高，因为需要一个额外的计算，那就是计算出三角形所在的平面，而下面要介绍的方法则可以省去这个计算。

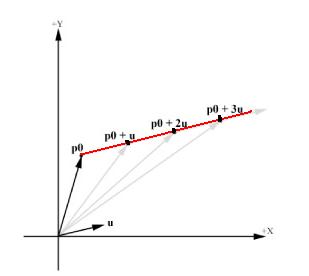
## 本文的方法

接下来会涉及到一些数学知识，不过没关系，我会详细解释每一个步骤，不至于太晦涩，只要您不觉得烦就行了，好了开始！

射线的参数方程如下，其中O是射线的起点，D是射线的方向。

http://pic002.cnblogs.com/img/zdd/201008/2010080823241736.jpg

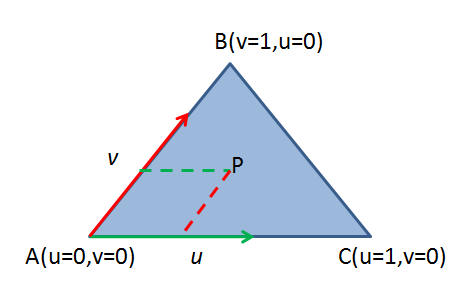
我们可以这样理解射线，一个点从起点O开始，沿着方向D移动任意长度，得到终点R，根据t值的不同，得到的R值也不同，所有这些不同的R值便构成了整条射线，比如下面的射线，起点是P0，方向是u，p0 + tu也就构成了整条射线。



三角形的参数方程如下，其中V0，V1和V2是三角形的三个点，u, v是V1和V2的权重，1-u-v是V0的权重，并且满足u>=0, v >= 0,u+v<=1。

http://pic002.cnblogs.com/img/zdd/201008/2010080823243118.jpg

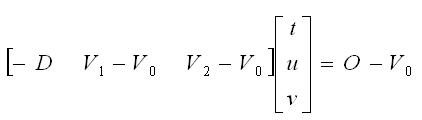
确切的说，上面的方程是三角形及其内部所有点的方程，因为三角形内任意一点都可以理解为从顶点V0开始，沿着边V0V1移动一段距离，然后再沿着边V0V2移动一段距离，然后求他们的和向量。至于移动多大距离，就是由参数u和v控制的。



于是，求射线与三角形的交点也就变成了解下面这个方程-其中t,u,v是未知数，其他都是已知的

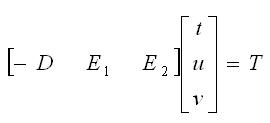
http://pic002.cnblogs.com/img/zdd/201008/2010080823245595.jpg

移项并整理，将t,u,v提取出来作为未知数，得到下面的线性方程组

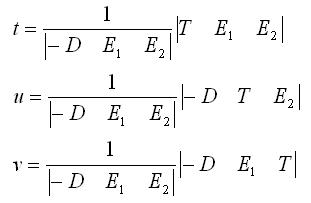


现在开始解这个方程组，这里要用到两个知识点，一是克莱姆法则，二是向量的混合积。

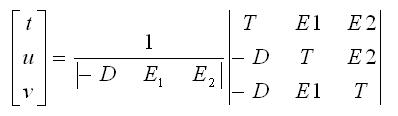
令E1 = V1 - V0，E2 = V2 - V0，T = O - V0上式可以改写成



根据克莱姆法则，可得到t,u,v的解分别是



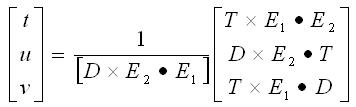
将这三个解联合起来写就是



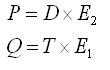
根据混合积公式

http://pic002.cnblogs.com/img/zdd/201008/2010080823262395.jpg

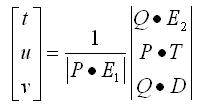
上式可以改写成



令



得到最终的公式，这便是下面代码中用到的最终公式了，之所以提炼出P和Q是为了避免重复计算



**代码部分**

理论部分阐述完毕，开始上代码，这份代码来自DirectX SDK中的Demo，名字叫做Picking(拾取)，该函数位于文件Pick.cpp的最末尾。这个函数有一个特点，就是判断语句特别多，因为对于一个频繁被调用的函数来说，效率是最重要的，这么多判断就是为了在某个条件不满足时，及时返回，避免后续不必要的计算。

复制代码

1 // Determine whether a ray intersect with a triangle

2 // Parameters

3 // orig: origin of the ray

4 // dir: direction of the ray

5 // v0, v1, v2: vertices of triangle

6 // t(out): weight of the intersection for the ray

7 // u(out), v(out): barycentric coordinate of intersection

8

9 bool IntersectTriangle(const Vector3& orig, const Vector3& dir,

10 Vector3& v0, Vector3& v1, Vector3& v2,

11 float\* t, float\* u, float\* v)

12 {

13 // E1

14 Vector3 E1 = v1 - v0;

15

16 // E2

17 Vector3 E2 = v2 - v0;

18

19 // P

20 Vector3 P = dir.Cross(E2);

21

22 // determinant

23 float det = E1.Dot(P);

24

25 // keep det > 0, modify T accordingly

26 Vector3 T;

27 if( det >0 )

28 {

29 T = orig - v0;

30 }

31 else

32 {

33 T = v0 - orig;

34 det = -det;

35 }

36

37 // If determinant is near zero, ray lies in plane of triangle

38 if( det < 0.0001f )

39 return false;

40

41 // Calculate u and make sure u <= 1

42 \*u = T.Dot(P);

43 if( \*u < 0.0f || \*u > det )

44 return false;

45

46 // Q

47 Vector3 Q = T.Cross(E1);

48

49 // Calculate v and make sure u + v <= 1

50 \*v = dir.Dot(Q);

51 if( \*v < 0.0f || \*u + \*v > det )

52 return false;

53

54 // Calculate t, scale parameters, ray intersects triangle

55 \*t = E2.Dot(Q);

56

57 float fInvDet = 1.0f / det;

58 \*t \*= fInvDet;

59 \*u \*= fInvDet;

60 \*v \*= fInvDet;

61

62 return true;

63 }

复制代码

**参数说明**

输入参数：前两个参数orig和dir是射线的起点和方向，中间三个参数v0，v1和v2是三角形的三个顶点。

输出参数：t是交点对应的射线方程中的t值，u，v则是交点的纹理坐标值