实 验 报 告 单

**实验名称： 实验2：C#面向对象程序设计 1**

同组人：无 实验课时：4

实验室：计算机软件实验室 报告日期：2022年4月2日

**一、实验目的：**

1. 理解C#封装、继承、多态、接口、委托、事件等概念；

2. 能够选择合适的软件设计原则或设计模式进行开发；

3. 能够在程序设计中综合运用面向对象的多种开发技术，检查并验证程序的功能是否符合要求。

1. **实验内容：**

**主要截取三个类**

**TankFather,SingleObject,GameObject**

**1.GameObject**

internal abstract class tankFa:GameObject

{

//在父类中提供玩家坦克子类和敌人坦克子类的公共成员

private Image[] imgs = new Image[]{ };

public tankFa(int x,int y,Image[] imgs,int speed,int hp,Direction dir) : base(x, y, imgs[0].Width, imgs[0].Height, speed, hp, dir)

{//这个构造函数没法初始化图片，在父类中没有这个属性

this.imgs = imgs;//记得手动赋值

}

public abstract void Born();

public int bornTime = 0;

public bool isMove = false;

public abstract void Fire();

public abstract void isOver();

public override void Draw(Graphics g)

{

bornTime++;

if(bornTime % 20 == 0)

{

isMove = true; //

}

if (isMove)

{

switch (this.Dir) //不同方向就贴不同图片

{

case Direction.Up:

g.DrawImage(imgs[0], this.X, this.Y);

break;

case Direction.Down:

g.DrawImage(imgs[1], this.X, this.Y);

break;

case Direction.Left:

g.DrawImage(imgs[2], this.X, this.Y);

break;

case Direction.Right:

g.DrawImage(imgs[3], this.X, this.Y);

break;

}

}

}

}

**2.SingleObject**

internal class SingleObject

{

//这个单例类用来创造全局唯一的对象

private SingleObject() { }

public static SingleObject \_singleObject = null;

public static SingleObject GetSingleObject()

{

if(\_singleObject == null)

{

\_singleObject = new SingleObject();

}

return \_singleObject;

}

public void removeGameObject(GameObject go)

{

if(go is Boob)

{

listboob.Remove(go as Boob);

}

if(go is EnemyBullet)

{

listenemyBullet.Remove(go as EnemyBullet);

}

if(go is EnemyTank)

{

ListEnemyTank.Remove(go as EnemyTank);

}

if(go is PlayerBullet)

{

listplayerBullet.Remove(go as PlayerBullet);

}

}

public PlayerTank PT

{

get;

set;

}

//将敌人tank,以及子弹放到泛型中

List<EnemyTank> ListEnemyTank = new List<EnemyTank>();

List<PlayerBullet> listplayerBullet = new List<PlayerBullet>();

List<EnemyBullet> listenemyBullet = new List<EnemyBullet>();

List<Boob> listboob =new List<Boob>();

List<tankBirth> tb = new List<tankBirth>();

public void PZJC()

{

#region 判断玩家子弹是否打在敌人身上

for(int i = 0; i < listplayerBullet.Count; i++)

{

for(int j = 0; j < ListEnemyTank.Count; j++)

{

if (listplayerBullet[i].GetRectangle().IntersectsWith(ListEnemyTank[j].GetRectangle()))//玩家子弹达到了敌人身上

{

ListEnemyTank[j].hp-= 2;

ListEnemyTank[j].isOver();

//消除坦克

/\* SingleObject.GetSingleObject().removeGameObject(ListEnemyTank[j]);\*/

//消除坦克

SingleObject.GetSingleObject().removeGameObject(listplayerBullet[i]);

break;

}

}

}

#endregion

#region 判断敌人子弹是否打在玩家身上

for(int i = 0; i < listenemyBullet.Count; i++)

{

if (listenemyBullet[i].GetRectangle().IntersectsWith(PT.GetRectangle())){

PT.isOver();

listenemyBullet.Remove(listenemyBullet[i]);

}

}

for (int i = 0; i <listplayerBullet.Count; i++)

{

if (listplayerBullet[i].X <= 0 || listplayerBullet[i].Y <= 0 || listplayerBullet[i].X >= 710 || listplayerBullet[i].Y >= 590)

{

SingleObject.GetSingleObject().removeGameObject(listplayerBullet[i]);

break;

}

}

#endregion

}

//将父对象转化为子对象

public void AddGameObject(GameObject go)

{

if(go is PlayerTank)//IS 表示转换成功

{

PT = go as PlayerTank;//如果转换成功，则返回对应的对象，否则返回null

}

else if (go is EnemyTank)

{

ListEnemyTank.Add(go as EnemyTank);

}

else if(go is PlayerBullet)

{

listplayerBullet.Add(go as PlayerBullet);

}

else if(go is EnemyBullet)

{

listenemyBullet.Add(go as EnemyBullet);

}

else if(go is Boob)

{

listboob.Add(go as Boob);

}

else if(go is tankBirth)

{

tb.Add(go as tankBirth);

}

}

//绘制游戏对象

public void Draw(Graphics g)

{

PT.Draw(g);

for(int i = 0; i < ListEnemyTank.Count; i++)

{

ListEnemyTank[i].Draw(g);

}

for(int i = 0; i < listenemyBullet.Count; i++)

{

listenemyBullet[i].Draw(g);

}

for(int i = 0; i < listplayerBullet.Count; i++)

{

listplayerBullet[i].Draw(g);

}

for(int i =0; i<listboob.Count ; i++)

{

listboob[i].Draw(g);

}

/\* for(int i=0; i < tb.Count ; i++)

{

tb[i].Draw(g);

}\*/

}

}

1. **TankFather**

internal abstract class tankFa:GameObject

{

//在父类中提供玩家坦克子类和敌人坦克子类的公共成员

private Image[] imgs = new Image[]{ };

public tankFa(int x,int y,Image[] imgs,int speed,int hp,Direction dir) : base(x, y, imgs[0].Width, imgs[0].Height, speed, hp, dir)

{//这个构造函数没法初始化图片，在父类中没有这个属性

this.imgs = imgs;//记得手动赋值

}

public abstract void Born();

public int bornTime = 0;

public bool isMove = false;

public abstract void Fire();

public abstract void isOver();

public override void Draw(Graphics g)

{

bornTime++;

if(bornTime % 20 == 0)

{

isMove = true; //

}

if (isMove)

{

switch (this.Dir) //不同方向就贴不同图片

{

case Direction.Up:

g.DrawImage(imgs[0], this.X, this.Y);

break;

case Direction.Down:

g.DrawImage(imgs[1], this.X, this.Y);

break;

case Direction.Left:

g.DrawImage(imgs[2], this.X, this.Y);

break;

case Direction.Right:

g.DrawImage(imgs[3], this.X, this.Y);

break;

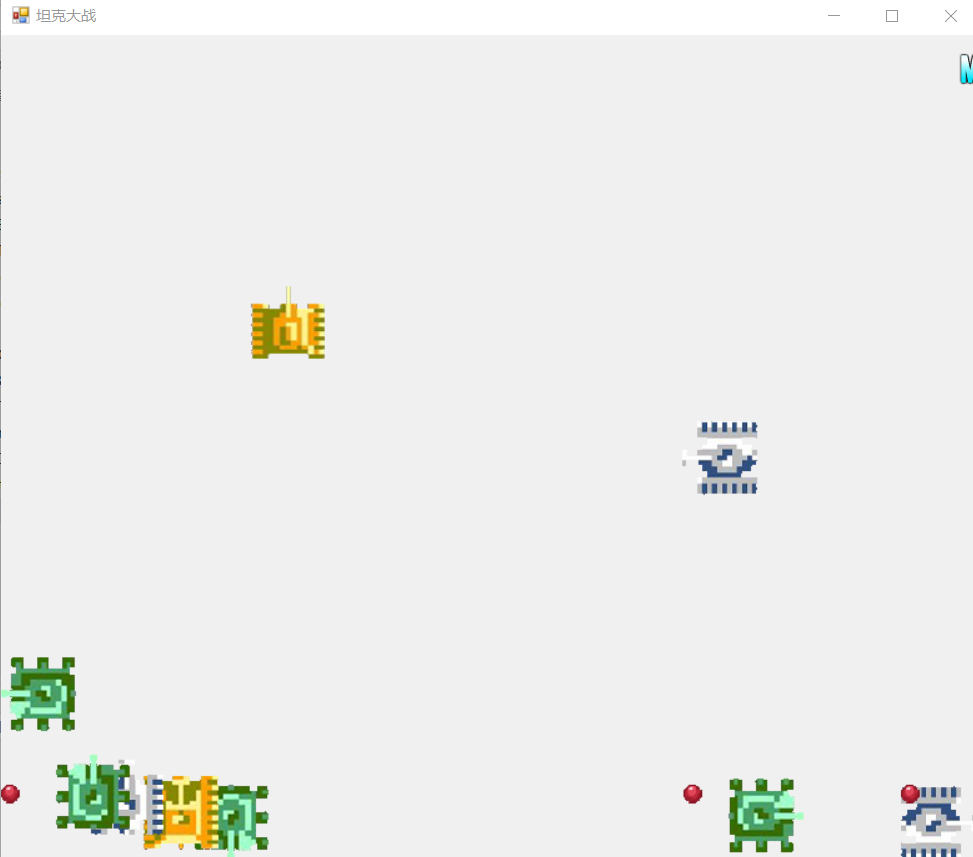
}

}

}

}

1. **实验结果：**



1. **实验总结：**

**主要有TankFather,SingleObject,GameObject类；其中，GameObject的子类下是所有物体，如坦克，子弹，出场动画，装备等等。而TankFather的子类有PlayerTank,EnemyTank，它们的属性又有坐标，rectangle,速度，图片等等；而SingleObject则是实现单例设计，对各种物体对象进行调用，如实现一个playerTank,实现多个EnemyTank,多个出场特效...;**

**除了基本结构，该游戏中的许多操作也值得学习；如，采用单例设计的方式创建对象，利用取模，random来调整延迟和调整创建对象的概率；还有一系列控件的使用，如timer，每隔一段时间调用；**

**又有获取事件的KeyDown,以此来实现移动和发射子弹。**

**在做这个实验时，一步一步地去做，很多地方觉得设计地很巧妙，也有很多没跟上的地方，导致过程有些曲折。**

———————————————————————————————

**成绩：**

批阅教师：

日 期：