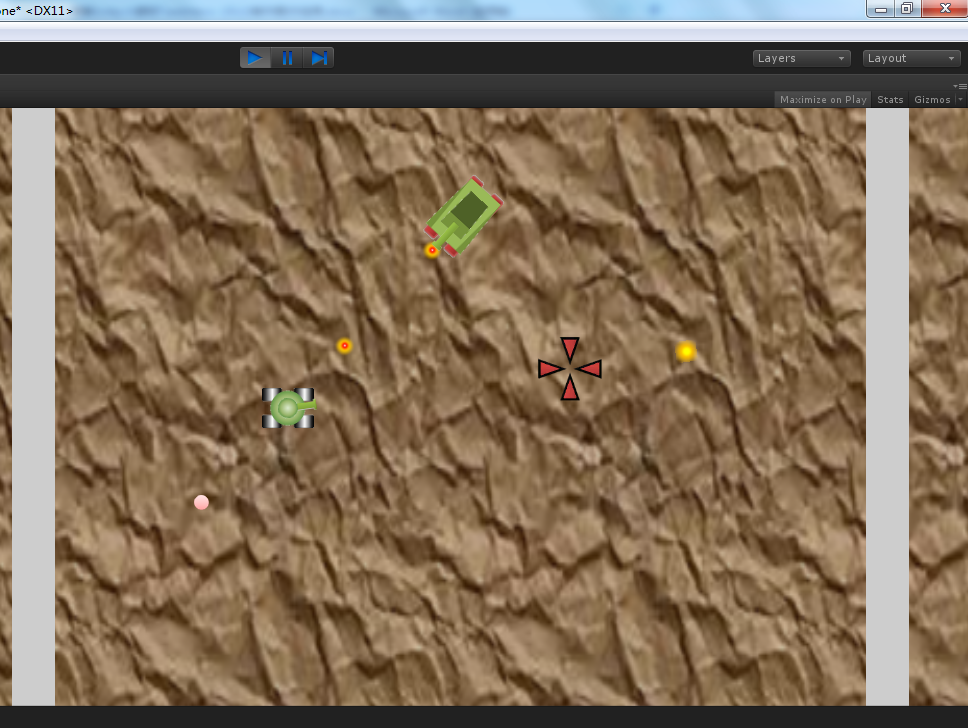
自制Unity小游戏TankHero-2D(2)制作敌方坦克

我在做这样一个坦克游戏，是仿照（<http://game.kid.qq.com/a/20140221/028931.htm>）这个游戏制作的。仅为学习Unity之用。图片大部分是自己画的，少数是从网上搜来的。您可以到我的github页面（<https://github.com/bitzhuwei/TankHero-2D>）上得到工程源码。

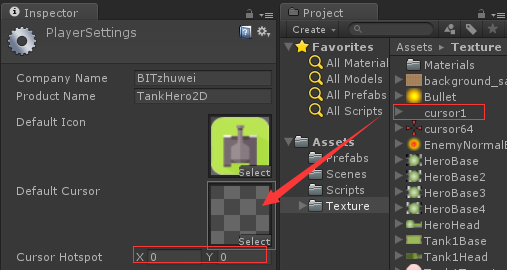


本篇主要记录制作敌方坦克（Tank1）的一些重点。

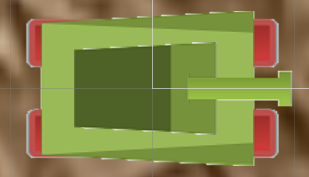
原本制作敌方坦克是很简单的，只要把TankHero复制一份，改改贴图就差不多了。不过考虑到代码的简洁和可重用，本篇花了些心思在重构上。

# 关于自定义鼠标箭头

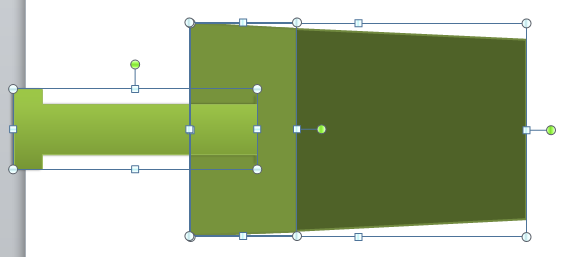
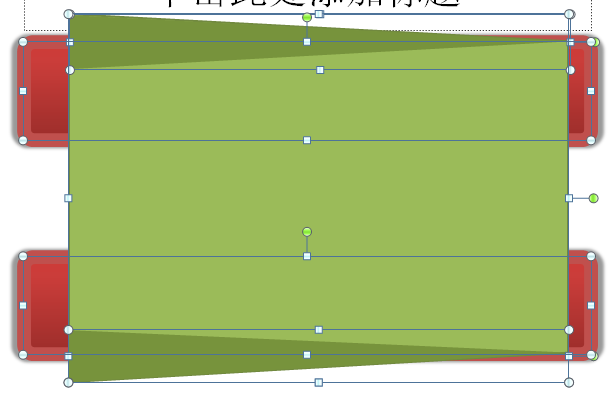
上一篇介绍了如何自定义鼠标箭头的事。这里补个漏。经过上一篇的研究，已经可以显示自定义的鼠标样式了，但是原有的鼠标箭头仍然存在，这怎么办？容易，只需制作一个1\*1像素的全透明的png图片，赋给Default Cursor即可。实际上就是让默认鼠标样式透明掉。



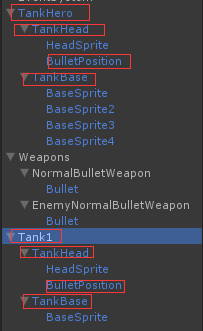
# 敌方坦克模型



这个模型依旧是用PPT做的。SmartArt+“形状”解决问题。具体技巧可参考[上一篇](http://www.cnblogs.com/bitzhuwei/p/tank-hero-2d-1-make-tank-hero.html)。



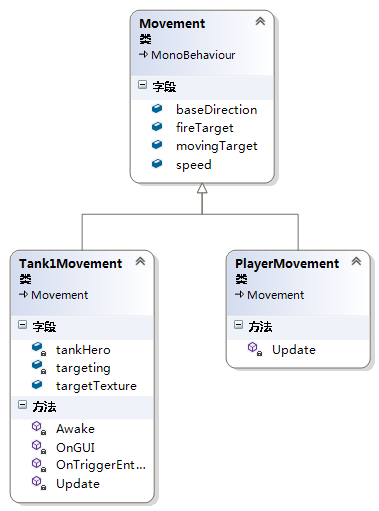
# 敌方坦克结构



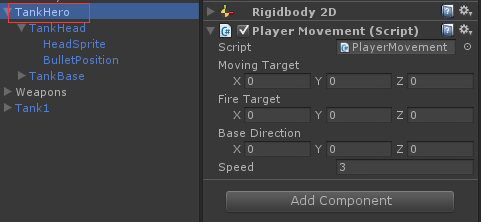
如上图所示，使用Duplicate从TankHero复制一个，重命名为Tank1。Tank1就是我们要做的敌方坦克了。其炮塔、底座、炮弹起始点这些结构都是一样的。

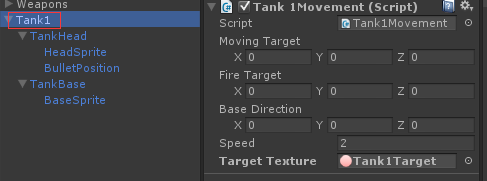
# 重构坦克运动代码

玩家坦克和敌方坦克有很多共同点，比如坦克对象的结构。也有一些特征（移动、旋转、开炮等）既相似又不同。具体来说，玩家坦克是由鼠标键盘指挥的，敌方坦克则要由AI指挥。指挥者不同，但是指挥的效果都是移动旋转开炮，是可以用同样的代码处理的。所以我在这里抽象出一个专门保存指挥信息的Movement类，这样就隔开了指挥者与执行者。



PlayerMovement接收用户输入的信息，存到基类的字段。Tank1Movement用AI获取指挥信息，存到基类的字段。



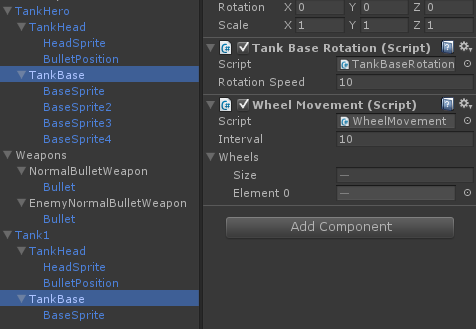


这样一来，在其它地方（不同类型的坦克的炮塔、底座）就都可以用 **Movement m = this.GetComponentXXX<PlayerMovement>();** 这样统一的方式获取平移、旋转、目标、目的地等信息了。

下面我们来详细介绍。

# 底座的旋转和轮子滚动

两种坦克的底座部分，只有轮子滚动部分是不同的。两者使用的脚本则都是TankBaseRotation和WheelMovement。



WheelMovement代码没有任何改变，只不过在Inspector里的Wheels数组元素不同而已。

在TankBaseRotation中则出现了这样的代码：

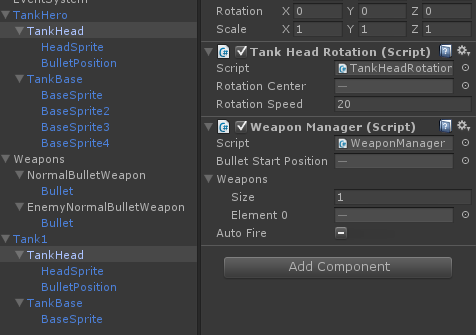
|  |
| --- |
| private Movement movementScript;  void Awake()  {  movementScript = this.GetComponentInParent<Movement> ();  } |

有了movementScript就可以得到坦克的移动方向movementScript.baseDirection，就可以更新坦克底座的旋转角度了。

|  |
| --- |
| void Update () {  if (movementScript == null) { return; }  var angle = Mathf.Atan2 (movementScript.baseDirection.y, movementScript.baseDirection.x) \* Mathf.Rad2Deg;  if (Mathf.Abs(angle - this.targetAngle) > 0.01f)  {  this.targetAngle = angle;  this.targetRotation = Quaternion.Euler (0, 0, angle);  }  this.transform.rotation = Quaternion.Slerp (  this.transform.rotation,  Quaternion.Euler (0, 0, angle),  rotationSpeed \* Time.deltaTime);  } |

这样就无需为两种坦克写两套旋转底座的脚本了。以后添加了新型坦克也仍然只需这一个脚本。

# 炮塔的旋转和武器管理



TankHero和Tank1的炮塔旋转中心(Rotation Center)和武器(Weapons)不同，但他们使用了相同的脚本（TankHeadRotation和WeaponManager）。这两个脚本中也都有如下的代码。

|  |
| --- |
| private Movement movementScript;  void Awake()  {  this.movementScript = this.GetComponentInParent<Movement> ();  //this.targetPosition = new Vector3 (1, 0, 0);  } |

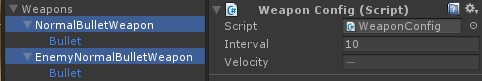
在movementScript中保存着目标的位置（fireTarget），旋转炮塔也很容易。

|  |
| --- |
| void Update () {  if (this.movementScript == null) { return; }  var y = this.movementScript.fireTarget.y - this.transform.position.y;  var x = this.movementScript.fireTarget.x - this.transform.position.x;  if (Mathf.Abs(y) > Quaternion.kEpsilon || Mathf.Abs(x) > Quaternion.kEpsilon)  {  this.targetAngle = Mathf.Atan2(y, x) \* Mathf.Rad2Deg;  var angle = this.targetAngle - this.transform.rotation.eulerAngles.z;  this.transform.RotateAround (this.rotationCenter.position, new Vector3 (0, 0, 1), angle);  }  } |

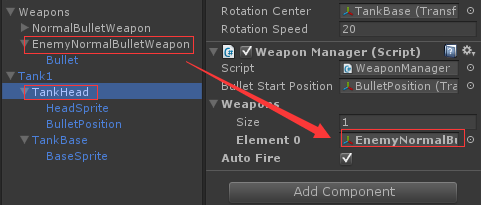
TankHero是玩家用鼠标开炮的，敌方坦克是自动开炮的。用autoFire标记坦克是否自动开炮即可。

|  |
| --- |
| void Update () {  passedInterval += Time.deltaTime \* 10;  if (passedInterval >= currentWeaponConfig.interval)  {  if (this.autoFire || Input.GetButton("Fire1"))  {  passedInterval = 0;  var bullet = Instantiate(currentBullet, bulletStartPosition.position, this.transform.rotation) as Transform;  bullet.renderer.enabled = true;  var bulletFly = bullet.GetComponent<BulletFly>();  bulletFly.undying = false;  bulletFly.velocity = currentWeaponConfig.velocity;  bulletFly.shooter = this.gameObject;  bulletFly.targetPosition = movementScript.fireTarget;  }  }  } |

# 武器系统



有了新的坦克，我们需要给它设计新的武器。只需Duplicate一下NormalBulletWeapon，再在WeaponConfig组件里调整一下敌方坦克武器的参数（炮弹速度调慢一点，不然敌人就太厉害了）。将新武器EnemyNormalBulletWeapon赋给Tank1的炮塔即可。



炮弹仍然是原有的那个，只需换一个贴图即可。

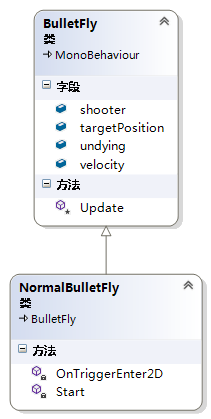


# 多种炮弹

玩过（<http://game.kid.qq.com/a/20140221/028931.htm>）的会发现有多种炮弹。其速度、攻击形式都不一样。



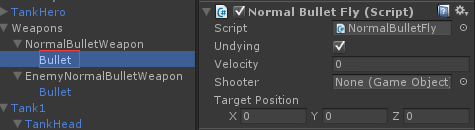
就是说，在不同类型的炮弹碰撞到某物时，会发生不同的事。因此我对控制炮弹飞行的脚本进行了抽象，在具体的子类里编写Trigger碰撞事件，用以处理不同的炮弹。



注意：如果在子类（NormalBulletFly）你添加了void Update();方法，那么Unity就不会调用父类（BulletFly）的Update方法了。这对Awake等都适用。就是说，Unity引擎只查找那些在继承层次上离MonoBehaviour最远的事件函数，找到之后就不再理会其它层上的同名函数了。

为什么是最远的？因为一个gameobject，其具有NormalBulletFly这个组件，意思是此gameobject拥有一个类型为NormalBulletFly的实例。很自然地，Unity会选中此类型的方法表中的Update方法。只有在NormalBulletFly中不存在时，才会轮到其父类的方法表。

当然目前为止只有1种炮弹，所以只有1个具体的NormalBulletFly脚本。这样，以后无论有多少种炮弹，只需一个Bullet的prefab即可。



# 总结

坦克的运动和炮弹的攻击，我都进行了重构。重构的目的是为了将重复的代码（平移、旋转、开炮、飞行）合并到一处，不同的代码（用户输入vs AI控制，不同的攻击方式）分别写如不同的脚本。重构的技术就是面向对象设计。

您可以到我的github页面（<https://github.com/bitzhuwei/TankHero-2D>）上得到工程源码。

请多多指教~