БАЛЛИСТИКА СТРЕЛЬБЫ

Янченко О.О
Науковий керівник – Ст викл. Новіков Ю.С.
Харківський національний університет радіоелектроніки
(61166, Харків, просп. Науки,14, каф. Програмна інженерія,
e-mail: oleksandr.yanchenko@nure.ua,
моб. номер (050) 979-68-36

Ballistics in the game is carried out by two productions: 1) visual; 2) mechanical. When creating a shooting in a visual way, objects are used (for example, a sphere). In this case, standard physics shoots Unity3D, which does not allow fast because of speed, and flies through various other objects about high speeds. Because of this problem, developers use the second method. The way out of this situation, the proposed script with which the shooting becomes more realistic. This script should answer the following questions: How is the shooting? Is there a collision? How is the calculation of the damage coefficient.

Баллистика в играх осуществляется двумя способами :

1)визуальная;

2)механическая.

При создании стрельбы визуальным способом используются предметы(например, сфера). В этом случае стандартная физика стрельбы в Unity3D не успевает срабатывать из-за скорости, и предмет на высоких скоростях пролетает сквозь различные другие предметы. Из-за такой проблемы разработчики используют второй способ – с помощью лучей.

Выходом из данной ситуации предложен скрипт, с помощью которого стрельба становится более реалистичной. Данный скрипт должен отвечать на следующие вопросы : Как происходит стрельба? Есть ли коллизия? Как идёт расчёт коэффициента урона.

Приведенный ниже скрипт (рис.1) отвечает на выше поставленные вопросы. Выпущенный объект летит по навесной траектории. Независимо от скорости полёта объекта происходит коллизия. Для расчёта коэффициента урона используются кривые, что зависят от расстояния, что пролетел объект, к примеру, объект который выпущен и сразу же столкнулся с другим объектом нанесёт больше урона, чем объект, который был выпущен и пролетел до иного объекта n-ое расстояние. Для реализации полёта требуется во время полёта объекта считать текущую позицию объекта в пространстве, а также считывается предыдущая позиция объекта. Из предыдущей позиции объекта считывается попал ли выпущенный объект в другой. Данный скрипт наиболее схож со стрельбой в реальной жизни.

```
    void Awake ()
    {
        Invoke ("DestroyNow", TimeToDestruct);


 rigidbody.velocity = transform.TransformDirection(Vector3.forward * StartSpeed);
        PreviousStep = gameObject.transform.position;
        StartTime = Time.time;
        CurrentDamage = Damage;
        if(RandomDamage)
            CurrentDamage += Random.Range(minRandLimit, maxRandLimit)
        Keyframe[] ks;
        ks = new Keyframe [3];
        ks [0] = new Keyframe (0, 1);
        ks [1] = new Keyframe (StartPoinOfDamageReduction / 100, 1);
        ks [2] = new Keyframe (1, FinalDamageInPercent / 100);
        DamageReductionGraph = new AnimationCurve (ks);
    }
void FixedUpdate()
    {
        Quaternion CurrentStep = gameObject.transform.rotation;
        transform.LookAt (PreviousStep, transform.up);
        RaycastHit hit = new RaycastHit();
        float Distance = Vector3.Distance (PreviousStep, transform.position);
        if (Distance == 0.0f)
            Distance = 1e-05f;
        Debug.Log(Distance);

 if(Physics.Raycast(PreviousStep, transform.TransformDirection(Vector3.back), out
hit, Distance * 0.9999f) && (hit.transform.gameObject !=gameObject))
        {
            Instantiate(particleHit, hit.point,
Quaternion.FromToRotation(Vector3.up, hit.normal));
            SendDamage(hit.transform.gameObject);
        }
        gameObject.transform.rotation = CurrentStep;

        PreviousStep = gameObject.transform.position;
    }
    void DestroyNow ()
    {
        DestroyObject (gameObject);
    }
    void SendDamage(GameObject Hit)
    {
        Hit.SendMessage ("ApplyDamage",
CurrentDamage * GetDamageCoefficient(),
SendMessageOptions.DontRequireReceiver);
        Destroy(gameObject);
    }
    float GetDamageCoefficient()
    {
        float Value = 1.0f;
        float CurrentTime = Time.time - StartTime;

Value = DamageReductionGraph.Evaluate(CurrentTime / TimeToDestruct);
        return Value;
    }
}
```

Рис.1 – баллистика стрельбы

Вывод. Из-за некачественной физики стрельбы в unity3D методом создания с помощью визуального способа, наиболее лучшими вариантами будут использование лучей (что имеет некоторые недостатки), а также приведенный выше скрипт.