

```

TurretGun ▶ InitiateStats ()
1 using UnityEngine;
2 using System.Collections;
3
4 public class TurretGun : MonoBehaviour
5 {
6     private int m_damage;
7
8     private float m_attackSpeed;
9     private float s_attackSpeed;
10    private float m_aimSpeed;
11    private float m_step;
12
13    private Vector3 m_forward;
14    private Vector3 m_targetDotPos;
15    private Vector3 m_lookDirection;
16
17    private Quaternion m_lookRotation;
18
19    private bool m_fired = false;
20    private bool m_inLOS = false;
21
22    private GameObject target;
23
24    private TurretBody Turret;
25
26    // Use this for initialization
27 void Start()
28 {
29     InitiateStats();
30 }
31
32 // Update is called once per frame
33 void Update()
34 {
35     GetTarget();
36
37     CheckTargetState();
38     CheckAttackStatus();
39 }
40
41 void InitiateStats()
42 {
43     Turret = GameObject.FindWithTag("Turret").GetComponent<TurretBody>();
44
45     m_damage = Turret.damage;
46     m_attackSpeed = Turret.attackSpeed;
47     s_attackSpeed = m_attackSpeed;
48     m_aimSpeed = Turret.aimSpeed;
49 }
50
51 void GetTarget()
52 {
53     target = Turret.target;
54 }
55
56 void LookAtTarget()
57 {
58     m_lookDirection = (target.transform.position - transform.position).normalized;
59     m_lookRotation = Quaternion.LookRotation(m_lookDirection);
60
61     m_step = m_aimSpeed * Time.deltaTime;
62
63     transform.rotation = Quaternion.Slerp(transform.rotation, m_lookRotation, m_step);
64 }
65
66 void CheckTargetState()
67 {
68     if(target)
69     {
70         LookAtTarget();
71         TargetLockCheck();
72     }
73     else
74     {
75         m_inLOS = false;
76         m_fired = false;
77     }
78 }
79
80 void TargetLockCheck()
81 {
82     m_forward = transform.TransformDirection(Vector3.forward);
83     m_targetDotPos = (target.transform.position - transform.position).normalized;
84
85     if(Vector3.Dot(m_forward, m_targetDotPos) >= 0.9f)
86     {
87         m_inLOS = true;
88     }
89     else
90     {
91         m_inLOS = false;
92     }
93 }
94
95 void CheckAttackStatus()
96 {
97     if(target)
98     {
99         if(m_inLOS && !m_fired)
100         {
101             m_fired = true;
102             Attack();
103             print("boom");
104         }
105
106         if(m_fired)
107         {
108             AttackCooldown();
109         }
110     }
111 }
112
113 void AttackCooldown()
114 {
115     m_attackSpeed -= Time.deltaTime;
116
117     if(m_attackSpeed <= 0)
118     {
119         m_fired = false;
120         m_attackSpeed = s_attackSpeed;
121     }
122 }
123
124 void Attack()
125 {
126     if(target)
127     {
128         target.GetComponent<EnemyStats>().health -= m_damage;
129     }
130 }
131 }

```