

```
1 using Final_Project;
2 using Final_Project.Properties;
3 using System;
4 using System.Collections.Generic;
5 using System.Linq;
6 using System.Security.Cryptography.X509Certificates;
7 using System.Text;
8 using System.Timers;
9 using static Final_Project.Form1;
10
11 namespace BasicAI
12 {
13     public class Enemy
14     {
15         //amalgamates most things required to make an enemy so most things ↗
16         //are self contained,
17         //same thing as gunner but doesnt shoot and just tries to follow ↗
18         //player
19         public PictureBox picEnemy;
20         PictureBox player;
21         Label[] projectiles;
22         Boolean isActive = true;
23         int internalHealth;
24         public int healthcap;
25         public int internalDeaths = 0;
26         public double speed;
27         public WeaponSelected currentWeapon;
28
29         Form1 form = Form1.GetInstance();
30
31         System.Random r = new System.Random((int) ↗
32             System.DateTime.Now.Ticks);
33
34         public Enemy(PictureBox picEnemy, PictureBox player, Label[] ↗
35             projectiles, double speed, int healthcap)
36         {
37             this.picEnemy = picEnemy;
38             this.player = player;
39             this.projectiles = projectiles;
40
41             internalHealth = healthcap;
42             this.healthcap = healthcap;
43             this.speed = speed;
44             currentWeapon = WeaponSelected.SWORD;
45         }
46
47         public void Update()
48         {
49             if (internalHealth > 0)
```

```
46         {
47             pursuit();
48         }
49
50         else
51         {
52             internalDeaths++;
53             respawn(r.Next(0, 1000), r.Next(0, 1000));
54
55         }
56     }
57
58     private void pursuit()
59     {
60         double posX = picEnemy.Left + (speed * getDeltaX());
61         double posY = picEnemy.Top + (speed * getDeltaY());
62
63         if (enemyTouch() || swordTouch())
64         {
65             posX -= (1 * getDeltaX());
66             posY -= (1 * getDeltaY());
67
68             picEnemy.Left = (int)posX;
69             picEnemy.Top = (int)posY;
70         }
71
72
73
74         if (!shieldTouch())
75         {
76             picEnemy.Left = (int)posX;
77             picEnemy.Top = (int)posY;
78         }
79         else
80         {
81             posX -= (0.1 * getDeltaX());
82             posY -= (0.1 * getDeltaY());
83
84             picEnemy.Left = (int)posX;
85             picEnemy.Top = (int)posY;
86         }
87
88     }
89
90     public void reset()
91     {
92         internalDeaths = 0;
93         internalHealth = healthcap;
94     }
```

```
95
96     public void move(int x, int y)
97     {
98         picEnemy.Left = x;
99         picEnemy.Top = y;
100     }
101
102     public void respawn(int x, int y)
103     {
104         internalHealth = healthcap;
105         move(x, y);
106     }
107
108     public Boolean shieldTouch()
109     {
110         if (currentWeapon == WeaponSelected.SHIELD)
111         {
112             for (int i = 0; i < projectiles.Length; i++)
113             {
114                 if (picEnemy.Bounds.Intersects(projectiles
115                 [i].Bounds))
116                 {
117                     return true;
118                 }
119             }
120             return false;
121         }
122         else
123             return false;
124     }
125
126     public Boolean enemyTouch()
127     {
128         return picEnemy.Bounds.Intersects(player.Bounds);
129     }
130
131     public Boolean swordTouch()
132     {
133
134         for(int i = 0; i < projectiles.Length; i++)
135         {
136             if (currentWeapon == WeaponSelected.SWORD &&
137             picEnemy.Bounds.Intersects(projectiles[i].Bounds))
138             {
139                 //form.spawnHeart();
140                 internalHealth--;
141                 return true;
142             }
143         }
144     }
145 }
```

```
142     }
143
144     return false;
145 }
146
147 public int getHealth()
148 {
149     return internalHealth;
150 }
151
152 public int getDeaths()
153 {
154     return internalDeaths;
155 }
156
157 private double getDeltaY()
158 {
159     int playerX = player.Left; // - (player.Width / 2);
160     int playerY = player.Top; // + (player.Height / 2);
161
162     int enemyX = picEnemy.Left; // - (picEnemy.Width / 2);
163     int enemyY = picEnemy.Top; // + (picEnemy.Height / 2);
164
165     double deltaX = playerX - enemyX;
166     double deltaY = playerY - enemyY;
167
168     return deltaY;
169 }
170
171
172 private double getDeltaX()
173 {
174     int playerX = player.Left; // - (player.Width / 2);
175     int playerY = player.Top; // + (player.Height / 2);
176
177     int enemyX = picEnemy.Left; // - (picEnemy.Width / 2);
178     int enemyY = picEnemy.Top; // + (picEnemy.Height / 2);
179
180     double deltaX = playerX - enemyX;
181     double deltaY = playerY - enemyY;
182
183     return deltaX;
184 }
185 }
186 }
187 }
188
```