

```
1 using Microsoft.Xna.Framework;
2 using Microsoft.Xna.Framework.Graphics;
3 using System;
4 using System.Collections.Generic;
5 using System.Text;
6
7 namespace GameDemo.Shared
8 {
9     public enum ENEMY_TYPES { SKELETON = 1, BOULDER = 2, AXE = 3, GHOST = 4 }
10
11     public class Enemy :Sprite
12     {
13
14         bool isVisible;
15         public float startingYpos;
16
17         ENEMY_TYPES type;
18
19
20         // shows whether the player is visible
21         public bool Visibility
22         { get { return isVisible; } }
23
24
25         public ENEMY_TYPES Type
26         {
27             get { return (ENEMY_TYPES) type; }
28             set { type = value; }
29         }
30
31         public Vector2 Speed
32         {
33             get { return speed; }
34         }
35
36         public Enemy(ENEMY_TYPES enemyType,Texture2D texture, Vector2 position, Point frameSize,
37                     int totalFrames, Vector2 collisionOffset, Point currentFrame,
38                     Point sheetSize,
39                     Vector2 speed,int secondsperframe,float scale)
40             : base(texture, position, frameSize, totalFrames, collisionOffset,
41                 currentFrame, sheetSize, speed,scale)
42         {
43             millisecondsPerFrame = secondsperframe;
44             startingYpos = position.Y;
45             type = enemyType;
46         }
47
48         // get the collision rectangle
49         public override Rectangle collisionRect()
50         {
51             if(this.type==ENEMY_TYPES.BOULDER)
52                 return new Rectangle((int)position.X-frameSize.X, (int)
53                     position.Y-frameSize.Y, (int)((float)this.frameSize.X *
54                         scale - collisionOffset.X), (int)((float)this.frameSize.Y
```

```
        * scale - this.collisonOffset.Y));
52     else
53         return new Rectangle((int)position.X, (int)position.Y, (int)
            ((float)this.frameSize.X*scale-collisonOffset.X),(int)
            ((float)this.frameSize.Y*scale- this.collisonOffset.Y));
54     }
55
56     // where the enemies spawn
57     public virtual TimeSpan Spawn(GameTime gameTime)
58     {
59         switch (type)
60         {
61             case ENEMY_TYPES.AXE:
62             {
63                 isVisible = true;
64                 position.X = 1900;
65                 //position.Y = 800;
66                 break;
67             }
68             case ENEMY_TYPES.SKELETON:
69             {
70                 isVisible = true;
71                 position.X = 1900;
72                 //position.Y = 800;
73                 break;
74             }
75             case ENEMY_TYPES.BOULDER:
76             {
77                 isVisible = true;
78                 position.X = 1900;
79
80                 position.Y = 400;
81                 break;
82             }
83             case ENEMY_TYPES.GHOST:
84             {
85                 isVisible = true;
86                 position.X = 1900;
87                 position.Y = 500;
88                 break;
89             }
90         }
91         return gameTime.TotalGameTime;
92     }
93     public void Behaviour(GameTime gameTime, ref float timeOnScreen) {
94
95         switch (type)
96         {
97             case ENEMY_TYPES.AXE:
98             {
99                 position.X -= Speed.X;
100                 if (position.X < -200 || position.X > 1920)
101                     isVisible = false;
102                 break;
103             }
104             case ENEMY_TYPES.SKELETON:
```

```
105         {
106             position.X -= Speed.X;
107             if (position.X < -200 || position.X > 1920)
108                 isVisible = false;
109             break;
110         }
111         case ENEMY_TYPES.BOULDER:
112         {
113             position.X -= Speed.X;
114             if (position.X < 0 || position.X > 1920)
115                 isVisible = false;
116             position = Movement.Bouncing(position, 3, gameTime, ref
117 timeOnScreen);
118             break;
119         }
120         case ENEMY_TYPES.GHOST:
121         {
122             position.X -= Speed.X;
123             if (position.X < 0 || position.X > 1920)
124                 isVisible = false;
125             position = Movement.SinWave(position, 20, 0.2,
126 gameTime, ref timeOnScreen);
127             break;
128         }
129     }
130     public override void Draw(GameTime gametime, SpriteBatch spriteBatch,
131 float scale, SpriteEffects spriteEffects)
132     {
133         if(isVisible)
134             base.Draw(gametime, spriteBatch, scale, spriteEffects);
135     }
136
137     public void DrawRotating(GameTime gametime, SpriteBatch spriteBatch,
138 float scale, SpriteEffects spriteEffects, float rotation)
139     {
140         if(isVisible)
141             spriteBatch.Draw(Texture, position, new Rectangle(currentFrame.X *
142 frameSize.X, currentFrame.Y * frameSize.Y, frameSize.X,
143 frameSize.Y), Color.White, rotation, new Vector2
144 (frameSize.X/2, frameSize.Y/2), scale, spriteEffects, 0);
145     }
146 }
```