```
 1  using System.Collections;
 2  using System.Collections.Generic;
 3  using UnityEditor;
 4  using UnityEngine;
 5  using UnityEngine.Events;
 6  using Random = UnityEngine.Random;
 7
 8  public class SheepBehavior : MonoBehaviour
 9  {
10      [SerializeField]
11      float sheepSpeedMin, sheepSpeedMax;
12      float sheepSpeed;
13
14      [SerializeField]
15      float grazeTimerMin = 3, grazeTimerMax = 7;
16
17      [SerializeField]
18      float obedienceTimerMin = 3, obedienceTimerMax = 7;
19
20      [SerializeField]
21      float barkDistance;
22
23      [SerializeField]
24      float minFenceDistance = 5f;
25
26      [SerializeField]
27      float rayLength = 1f;
28
29      float fenceDistance = float.MaxValue;
30
31      bool hasRunAway = false;
32      bool nearFence = false;
33      bool isGrounded;
34
35      Rigidbody rigBod;
36      Rigidbody dogBod;
37
38      GameObject[] Fences;
39      GameObject closestFence;
40
41      float distance;
42
43      private Vector3 gizmoSpherePosition;
44      private bool drawGizmo = false;
45
46
47      private void OnEnable()
48      {
49          FindObjectOfType<EventManager>().OnBark += RunAway;
```

```
50      }
51      private void OnDisable()
52      {
53          FindObjectOfType<EventManager>().OnBark -= RunAway;
54      }
55
56
57      // Start is called before the first frame update
58      void Start()
59      {
60          // Give the sheep a random Y rotation value on start
61          float randomRotationY = Random.Range(0f, 360f);
62          transform.rotation = Quaternion.Euler(0f, randomRotationY, 0f);
63
64          //Get Sheep RigidBody
65          rigBod = GetComponent<Rigidbody>();
66
67          //Get the Dog
68          dogBod = GameObject.FindGameObjectWithTag
              ("Player").GetComponent<Rigidbody>();
69
70          //Start the Graze() Coroutine
71          StartCoroutine(Graze());
72
73          Fences = GameObject.FindGameObjectsWithTag("Fence");
74
75          gizmoSpherePosition = transform.position;
76          drawGizmo = true;
77      }
78
79      void Update()
80      {
81          if (Fences != null && nearFence == false)
82          {
83
84              //Check to see how close the Fences are
85              foreach (GameObject f in Fences)
86              {
87                  distance = Vector3.Distance(transform.position,
                      f.transform.position);
88                  if (distance <= fenceDistance)
89                  {
90                      fenceDistance = distance;
91                      closestFence = f;
92                  }
93                  if (Vector3.Distance(transform.position,
                      closestFence.transform.position) < minFenceDistance)
94                  {
95                      nearFence = true;
```

```
96                      MoveFromFence(closestFence);
97
98                  }
99              }
100         }
101         isGrounded = GroundCheck();
102     }
103
104     void FixedUpdate()
105     {
106
107         if (!isGrounded)
108         {
109             Vector3 newVelocity = new Vector3(rigBod.velocity.x,
                  rigBod.velocity.y - 2f, rigBod.velocity.z);
110             rigBod.velocity = newVelocity;
111         }
112     }
113
114     IEnumerator Graze()
115     {
116         //Choose one of three actions
117         int actionChoice = Random.Range(1, 4);
118         //Set countDown timer
119         float countDown = Random.Range(grazeTimerMin, grazeTimerMax);
120         switch (actionChoice)
121         {
122             case 1:
123                 //Move the sheep forward over a period of time
124                 while (countDown > 0)
125                 {
126                     //rigbod.AddForce(transform.forward * sheepSpeed *
                    Time.smoothDeltaTime, ForceMode.Acceleration);
127                     sheepSpeed = Random.Range(sheepSpeedMin,
                    sheepSpeedMax);
128                     rigBod.velocity = transform.forward * sheepSpeed;
129                     countDown -= Time.smoothDeltaTime;
130                     yield return null;
131                 }
132                 //Stop Sheep from moving when movement is completed
133                 rigBod.velocity = Vector3.zero;
134                 break;
135             case 2:
136                 //Rotate the sheep
137                 float degreesMoved = 0;
138                 //Set the random degrees the sheep will rotate
139                 float degreesToMove = Random.Range(1, 180);
140
141                 //Randomly choose negative or positive movement for
```

```
                    clockwise and counterclockwise rotation
                int chooseRotation = Random.Range(1, 3);
                int rotationDirection;
                if (chooseRotation == 1) rotationDirection = 1;
                else rotationDirection = -1;

                Vector3 rotate = new Vector3(0, rotationDirection, 0);
                //Rotate the sheep over a period of time
                while (degreesMoved < degreesToMove)
                {
                    transform.Rotate(rotate);
                    degreesMoved++;
                    yield return new WaitForSeconds(0.02f);
                }
                yield return new WaitForSeconds(countDown);
                break;
            case 3:
                //Pause the sheep for a period of time
                rigBod.velocity = Vector3.zero;
                while (countDown > 0)
                {
                    countDown -= Time.smoothDeltaTime;
                    yield return null;
                }
                break;
            default:
                break;
        }
        //Restart the Graze() Coroutine
        RestartGraze();

    }

    void RestartGraze()
    {
        StopAllCoroutines();
        StartCoroutine(Graze());
    }

    public void RunAway()
    {
        float dogDistance = Vector3.Distance(dogBod.position,
          rigBod.position);
        if (dogDistance <= barkDistance)
        {
            StopAllCoroutines();
            StartCoroutine(RunAwayCoroutine());
        }
    }
```

```
189
190     void MoveFromFence(GameObject fence)
191     {
192         StopAllCoroutines();
193         StartCoroutine(MoveFromFenceCoroutine(fence));
194     }
195
196     IEnumerator RunAwayCoroutine()
197     {
198         //Get the direction of the Dog in relation to the Sheep
199         Vector3 direction = dogBod.position - rigBod.position;
200         direction.Normalize();
201         // Calculate the rotation to face the opposite direction
202         Quaternion rotation = Quaternion.LookRotation(-direction);
203         //Rotate the sheep over a period of time
204         while (Quaternion.Angle(transform.rotation, rotation) > 0.1f)
205         {
206             transform.rotation = Quaternion.RotateTowards
                  (transform.rotation, rotation, 180f * Time.deltaTime);
207             yield return new WaitForSeconds(0.02f); // Adjust speed here
                  as well
208         }
209         //Set countDown timer
210         float countDown = Random.Range(obedienceTimerMin,
              obedienceTimerMax);
211         //Run Away
212         while (countDown > 0)
213         {
214             //rigbod.AddForce(transform.forward * sheepSpeed *
                  Time.smoothDeltaTime, ForceMode.Acceleration);
215             sheepSpeed = Random.Range(sheepSpeedMin, sheepSpeedMax);
216             rigBod.velocity = transform.forward * sheepSpeed;
217             countDown -= Time.smoothDeltaTime;
218             yield return null;
219         }
220         rigBod.velocity = Vector3.zero;
221         //Return to Grazing once obedience timer runs out
222         yield return new WaitForSeconds(countDown);
223         hasRunAway = true;
224         RestartGraze();
225
226
227         yield return null;
228     }
229
230     IEnumerator MoveFromFenceCoroutine(GameObject fence)
231     {
232         //Set the sheep velocity to zero
233         rigBod.velocity = Vector3.zero;
```

```
234
235         //pause for a second
236         yield return new WaitForSeconds(1.0f);
237
238         //Get the direction of the Fence in relation to the sheep
239         Vector3 direction = fence.transform.position - rigBod.position;
240         direction.Normalize();
241         direction = new Vector3(direction.x, 0f, direction.z);
242
243         //Calculate the rotation to face the opposite direction
244         Quaternion rotation = Quaternion.LookRotation(-direction);
245
246         //Rotate the sheep over a period of time
247         while (Quaternion.Angle(transform.rotation, rotation) > 0.1f)
248         {
249             transform.rotation = Quaternion.RotateTowards
                  (transform.rotation, rotation, 180f * Time.deltaTime);
250             yield return new WaitForSeconds(0.02f); // Adjust speed here
                  as well
251         }
252
253
254         //Set countDown timer
255         float countDown = Random.Range(obedienceTimerMin,
              obedienceTimerMax);
256
257         //Move away
258         while (countDown > 0)
259         {
260             sheepSpeed = Random.Range(sheepSpeedMin, sheepSpeedMax);
261             rigBod.velocity = transform.forward * sheepSpeed;
262             countDown -= Time.smoothDeltaTime;
263             yield return null;
264         }
265         nearFence = false;
266
267         //Return to Grazing
268         RestartGraze();
269
270         yield return null;
271     }
272
273     bool GroundCheck()
274     {
275         RaycastHit hit;
276
277         if (Physics.Raycast(transform.position, Vector3.down, out hit,
              rayLength))
278         {
```

```
279             return true;
280         }
281         Debug.Log(transform.position.y);
282         return false;
283     }
284 }
285
```