```
        targetAux = target;
        target = new GameObject();
        target.AddComponent<Agent>();
    }
```

3. Also, implement the `OnDestroy` function to handle references and avoid memory issues:

```
void OnDestroy ()
{
    Destroy(target);
}
```

4. Finally, define the `GetSteering` function:

```
public override Steering GetSteering()
{
    Vector3 direction = targetAux.transform.position - transform.
position;
    if (direction.magnitude > 0.0f)
    {
        float targetOrientation = Mathf.Atan2(direction.x,
direction.z);
        targetOrientation *= Mathf.Rad2Deg;
        target.GetComponent<Agent>().orientation =
targetOrientation;
    }
    return base.GetSteering();
}
```

How it works...

The algorithm computes the internal target orientation according to the vector between the agent and the real target. Then, it just delegates the work to its parent class.

Wandering around

This technique works like a charm for random crowd simulations, animals, and almost any kind of NPC that requires random movement when idle.

Getting ready

We need to add another function to our `AgentBehaviour` class called `OriToVec` that converts an orientation value to a vector.

```
                  OriToVec
public Vector3 ~~GetOriAsVec~~ (float orientation) {
    Vector3 vector  = Vector3.zero;
    vector.x = Mathf.Sin(orientation * Mathf.Deg2Rad) * 1.0f;
    vector.z = Mathf.Cos(orientation * Mathf.Deg2Rad) * 1.0f;
    return vector.normalized;
}
```

How to do it...

Attach Wander.cs to game object Wander

We could see it as a big three-step process in which we manipulate the internal target position in a parameterized random way, face that position, and move accordingly:

1. Create the `Wander` class deriving from `Face`:

```
using UnityEngine;
using System.Collections;

public class Wander : Face
{
    public float offset;
    public float radius;
    public float rate;
}
```

2. Define the `Awake` function in order to set up the internal target:

```
public override void Awake()
{
    target = new GameObject();
    target.transform.position = transform.position;
    base.Awake();
}
```

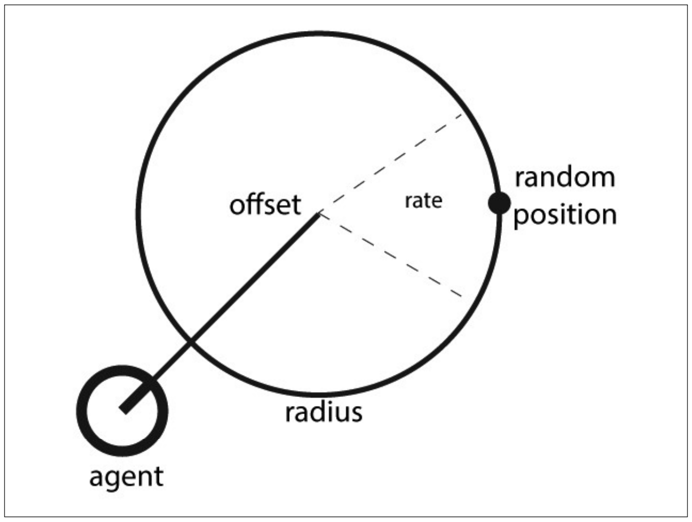
3. Define the `GetSteering` function:

```
public override Steering GetSteering()
{
    Steering steering = new Steering();
    float wanderOrientation = Random.Range(-1.0f, 1.0f) * rate;
    float targetOrientation = wanderOrientation + agent.
orientation;
    Vector3 orientationVec = OriToVec(agent.orientation);
```

```
        Vector3 targetPosition = (offset * orientationVec) +
    transform.position;
        targetPosition = targetPosition + (OriToVec(targetOrientation)
    * radius);
        targetAux.transform.position = targetPosition;
        steering = base.GetSteering();
        steering.linear = targetAux.transform.position - transform.
    position;
        steering.linear.Normalize();
        steering.linear *= agent.maxAccel;
        return steering;
    }
```

How it works...

The behavior takes into consideration two radii in order to get a random position to go to next, looks towards that random point, and converts the computed orientation into a direction vector in order to advance.



A visual description of the parameters for creating the Wander behavior