**“Seeing” obstacles with a raycast**

As you saw in the introduction to this chapter, raycasting is a technique that’s useful for a number of tasks within 3D simulations. One easily grasped task was shooting, but another place raycasting can be useful is for scanning around the scene. Given that scanning around the scene is a step in AI code, that means raycasting is used in AI code.

Earlier you created a ray that originated at the camera, because that’s where the player was looking from; this time you’ll create a ray that originates at the enemy. The first ray shot out through the center of the screen, but this time the ray will shoot forward in front of the character; figure 3.6 illustrates this. Then just like the shooting code used RaycastHit information to determine whether anything was hit and where, the AI code will use RaycastHit information to determine whether anything is in front of the enemy and, if so, how far away

One difference between raycasting for shooting and raycasting for AI is the radius of the ray detected against. For shooting the ray was treated as infinitely thin, but for AI the ray will be treated as having a large cross-section; in terms of the code, this means using the method SphereCast() instead of Raycast(). The reason for this difference is that bullets are tiny, whereas to check for obstacles in front of the character we need to account for the width of the character.

Create a new script called WanderingAI, attach that to the target object (alongside the ReactiveTarget script), and write the code from the next listing. Play the scene now and you should see the enemy wandering around the room; you can still shoot the target and it reacts the same way as before.

**«Увидеть» препятствия с помощью raycast**

Как вы видели во введении к этой главе, raycasting - это метод, который полезен для ряда задач в рамках трехмерного моделирования. Одной из легко выполнимых задач была съемка, но еще одно место, где может быть полезно raycasting, - это сканирование сцены. Учитывая, что сканирование сцены является шагом в коде AI, это означает, что в коде AI используется raycasting.

Ранее вы создали луч, исходящий от камеры, потому что именно оттуда смотрел игрок; на этот раз вы создадите луч, исходящий от врага. Первый луч вылетал через центр экрана, но на этот раз луч выйдет вперед перед персонажем; Рисунок 3.6 иллюстрирует это. Затем, точно так же, как код стрельбы использовал информацию RaycastHit, чтобы определить, было ли что-то поражено и где, код AI будет использовать информацию RaycastHit, чтобы определить, находится ли что-либо перед противником и, если да, то как далеко

Одно из различий между raycasting для стрельбы и raycasting для AI - это радиус обнаруженного луча. Для съемки луч считался бесконечно тонким, но для AI луч будет рассматриваться как имеющий большое поперечное сечение; с точки зрения кода это означает использование метода SphereCast () вместо Raycast (). Причина этой разницы в том, что пули крошечные, тогда как для проверки препятствий перед персонажем нам нужно учитывать ширину персонажа.

Создайте новый сценарий с именем WanderingAI, прикрепите его к целевому объекту (вместе со сценарием ReactiveTarget) и напишите код из следующего листинга. Сыграйте сцену сейчас, и вы должны увидеть врага, блуждающего по комнате; вы все еще можете стрелять в цель, и она реагирует так же, как и раньше.