# Module User Experience | Gaming

## Onderdeel Unity Practicum 3: Gevaar

## Auteurs

Paul Bergervoet (Gameplay en teamleiding)

Monique Dewanchand (C#)

Lars Tijsma (Unity)

Anton Visser (Gameplay en testen)



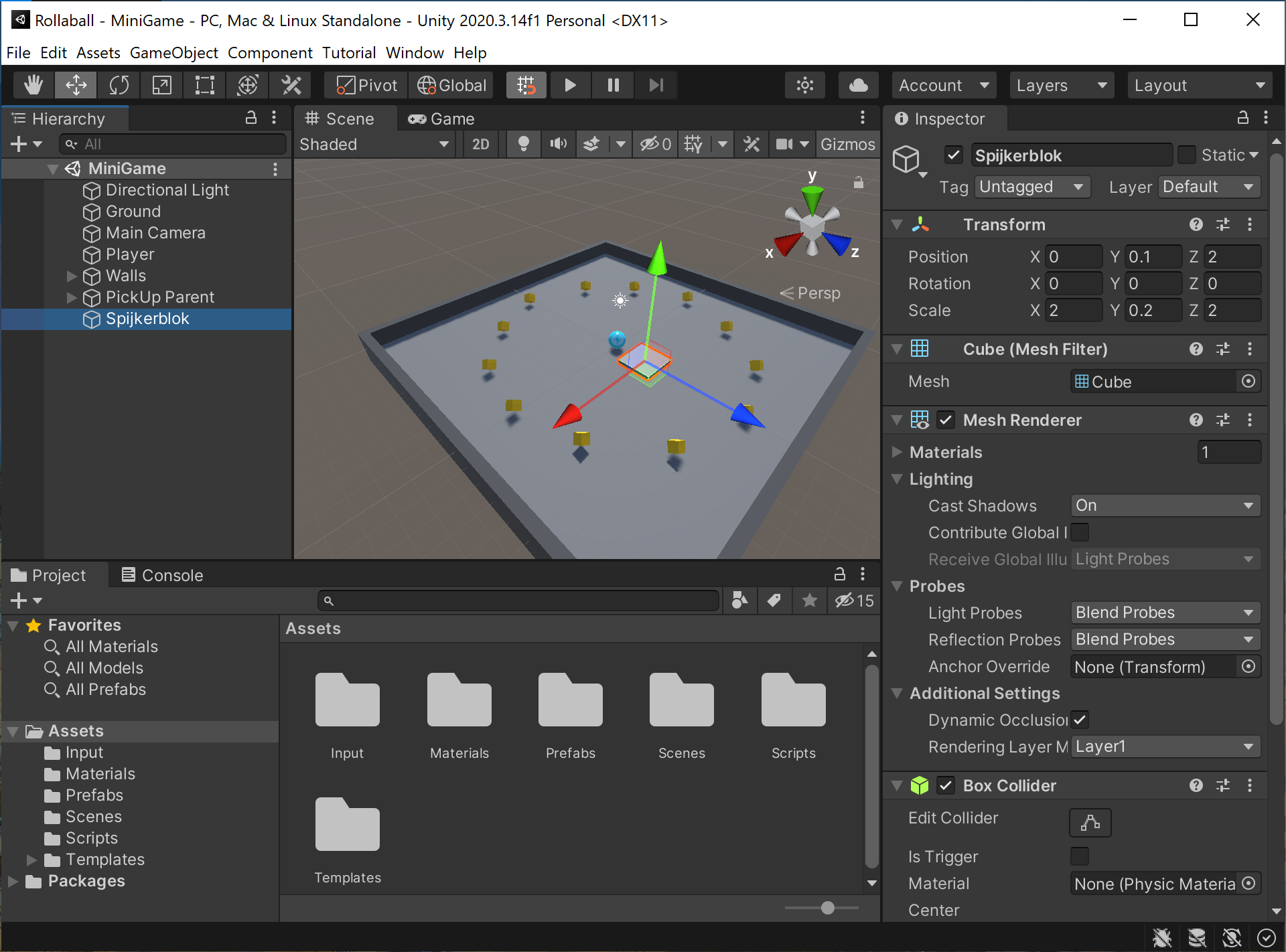
Unity-opdracht bij GD3: Gevaar

In deze tutorial maak je spijkerbed waar de bal niet op terecht kan komen. Als de bal op het spijkerbed terecht komt schiet de bal omhoog (waarmee we simuleren dat er een beetje lucht uit de bal schiet). De spijkers bevinden zich alleen aan de bovenkant.



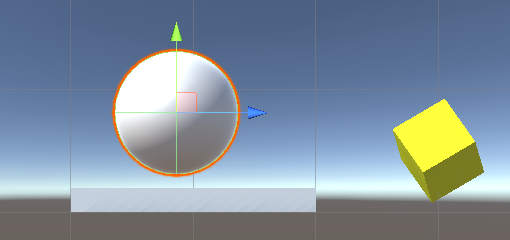
## Start

* Open de versie die je eerder hebt gemaakt, of open de versie in de map 03-gevaar en open de scene MiniGame en selecteer Player.
* Maak een nieuw Cube Game object, plaats deze voor de speler, verander de naam in Spijkerblok en wijzig de Scale in X: 2, Y: 0.2 en Z: 2 zoals in onderstaande afbeelding te zien is:



# Spijkerblok

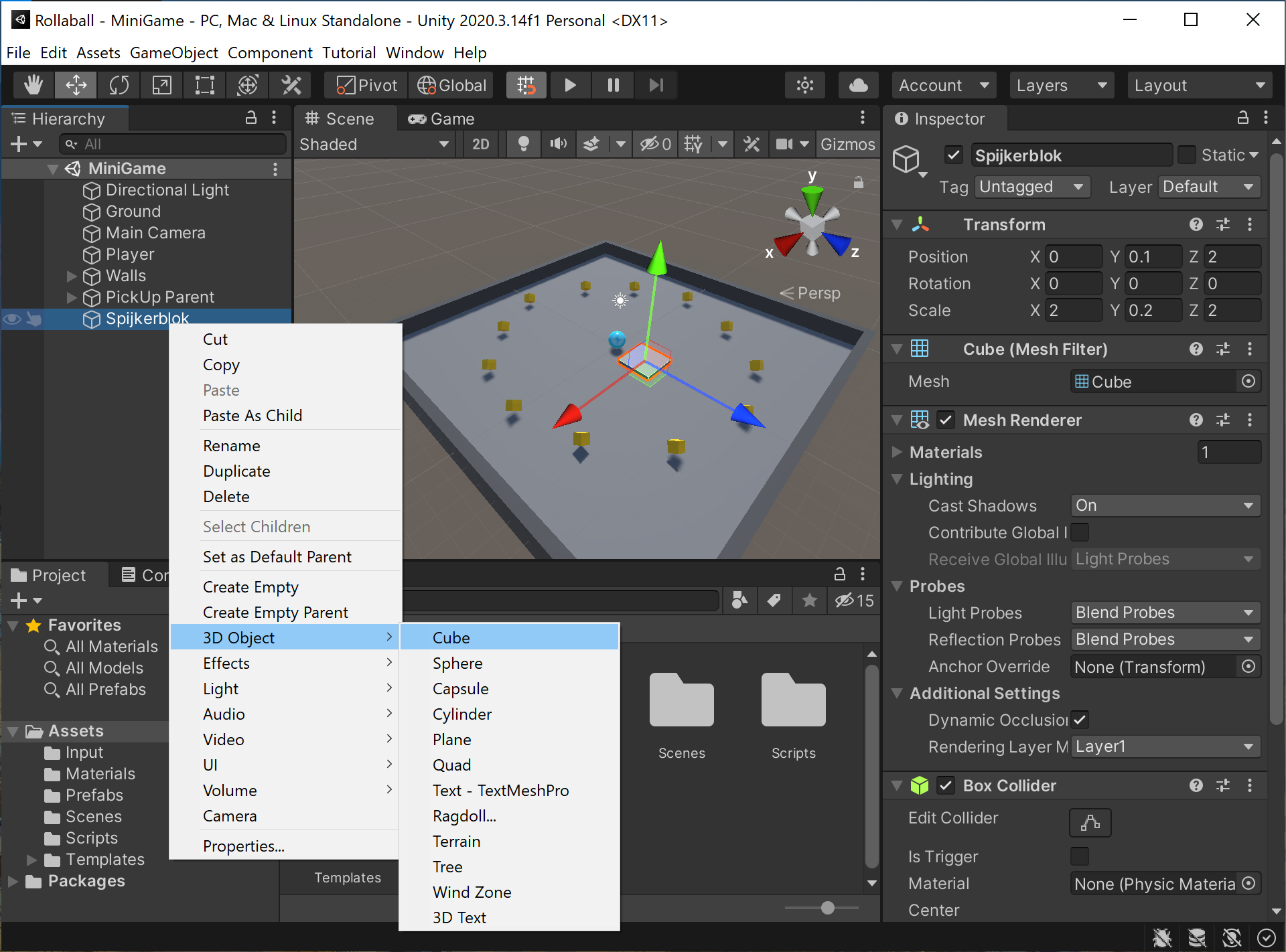
Het witte blokje tussen de bal en de pick up, *Spijkerblok*, wordt het blok met gevaarlijke spijkers. Het idee is dat de spijkers aan de bovenkant van het blok zitten en dat de bal alleen wordt geraakt als de bal zich aan de bovenkant van het blok bevindt zoals hieronder te zien is.



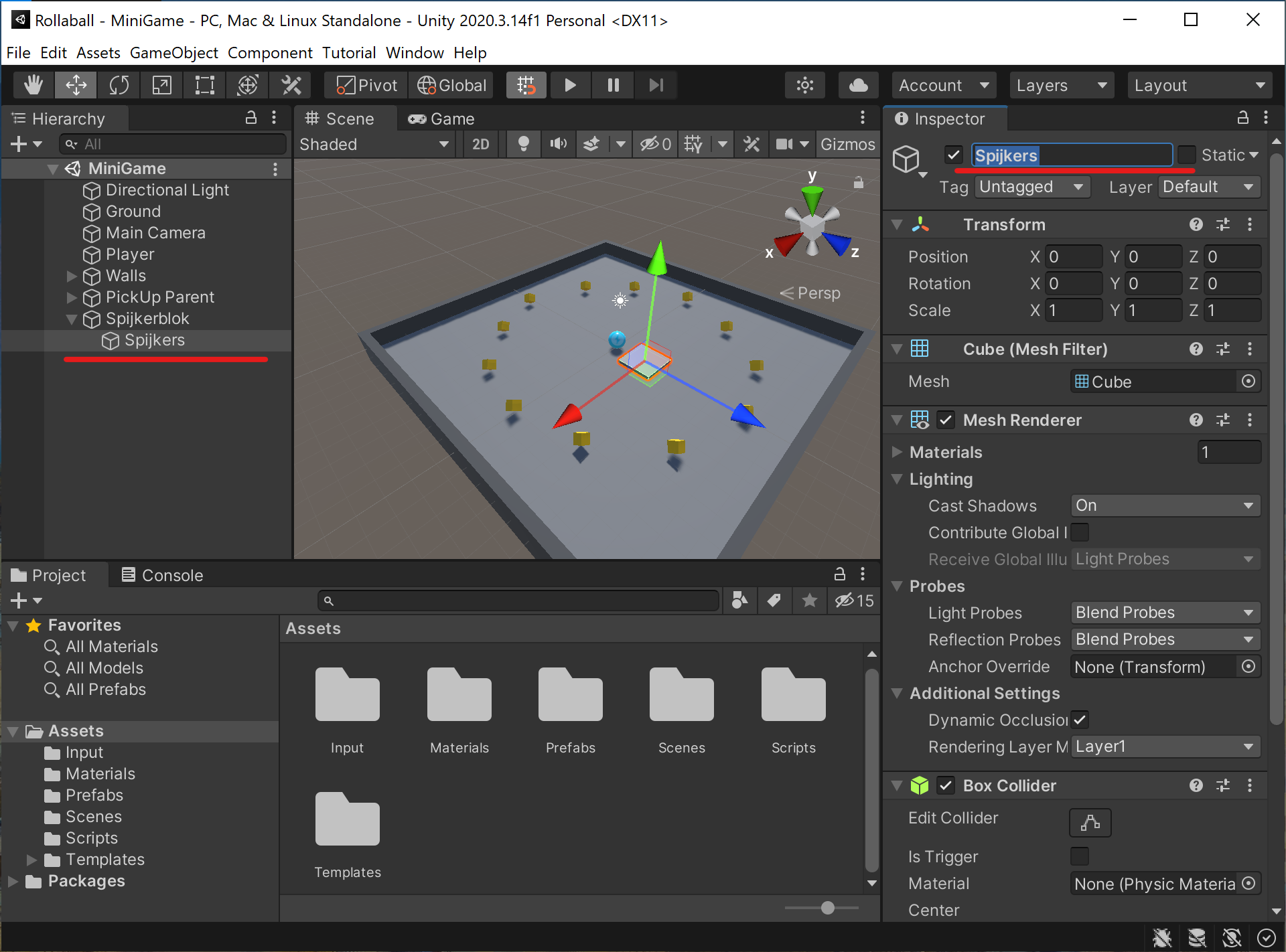
### Spijkers toevoegen

Het spijkerblok zelf heeft een collider waardoor de bal er niet doorheen gaat. Je kunt met code achterhalen of de bal de bovenkant raakt van deze collider, maar dit is vrij complex. De meest eenvoudige manier om goed botsingen te detecteren is een nieuw gameobject aan de bovenkant van het blok toe te voegen met een nieuwe collider.

* Voeg een nieuw gameobject toe aan *Spijkerblok* door in de *Hierarchy* met de rechtermuisknop te klikken op 3D Object > Cube zoals hieronder te zien is.

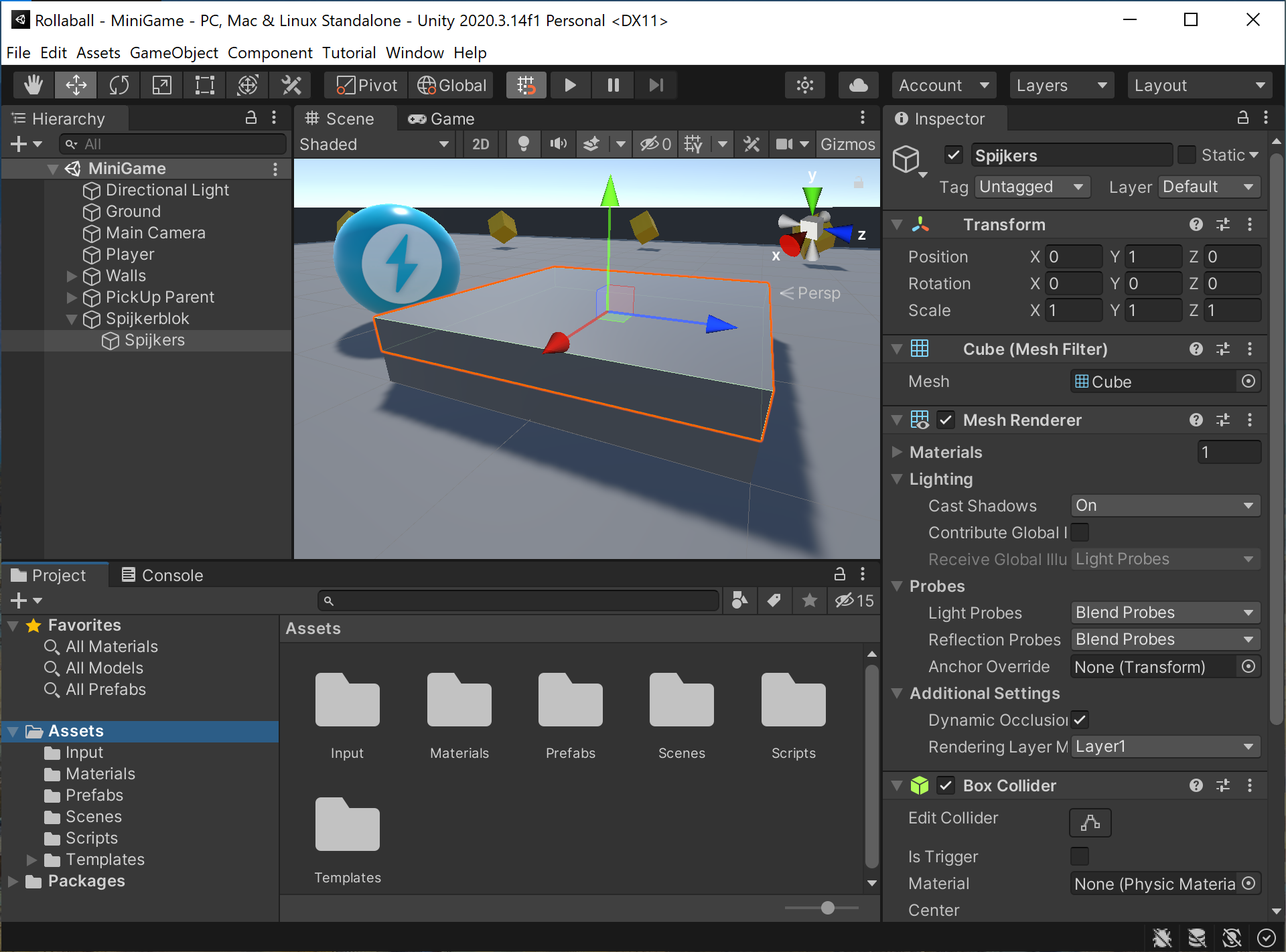


* Verander de naam van dit nieuwe object in *Spijkers*:



In de Scene View lijkt er niets veranderd te zijn. Dit komt doordat het nieuwe gameobject automatisch op dezelfde plek gezet wordt als de parent en dezelfde afmetingen en kleur krijgt.

* Verander de *Y-Position* van de Spijkers in 1, zodat de spijkers precies op het spijkerblok liggen:



Hoewel de spijkers nu aan de bovenkant van het blok staan, vallen ze nog niet op.

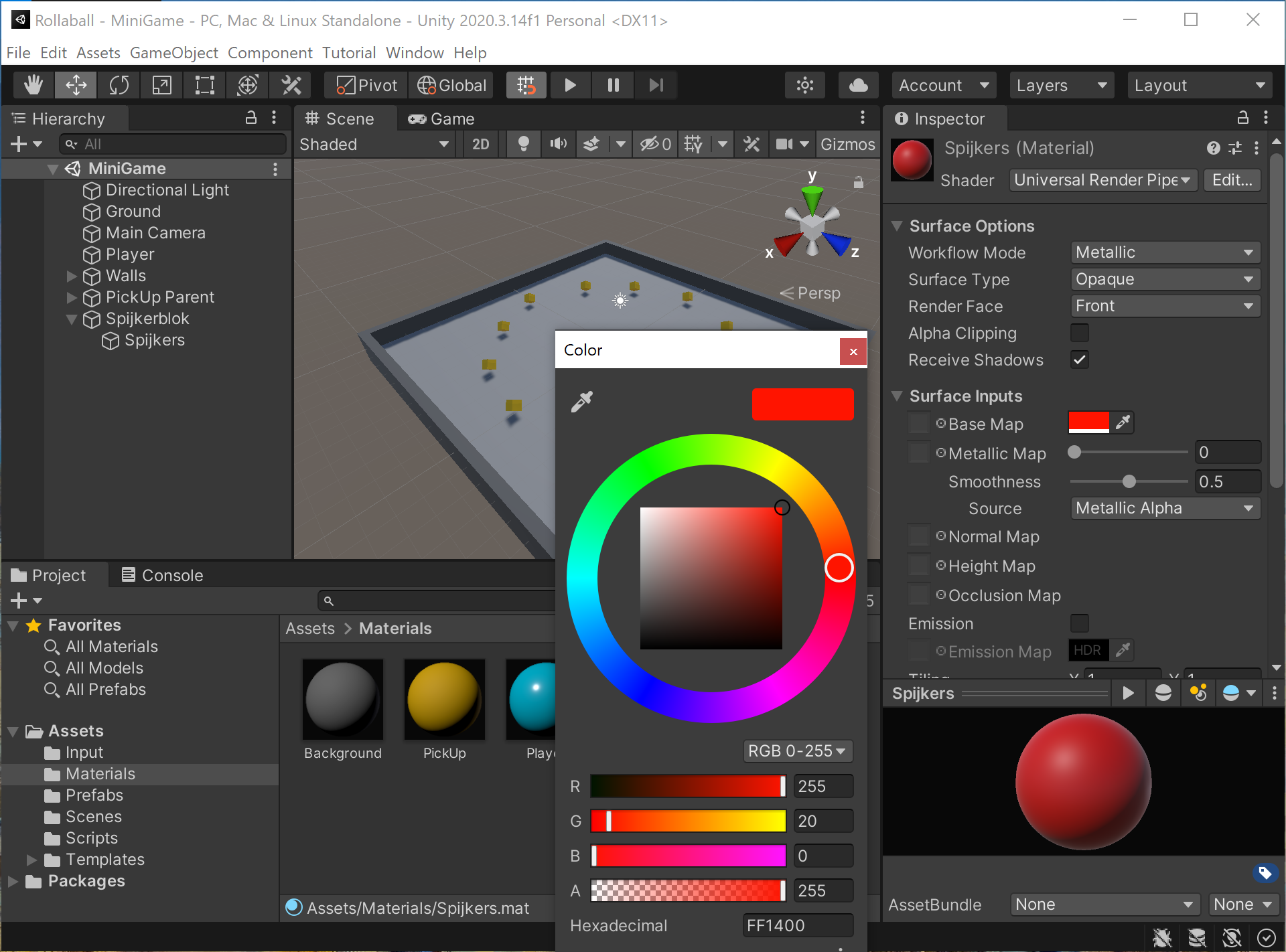
### 

### Kleur veranderen

* Voeg een nieuw materiaal door in de Project View met de rechtermuis te klikken op *Materials > Create > Material* zoals hieronder te zien is.



* Verander de naam van het Material in *Spijkers* en verander de kleur (in bijvoorbeeld rood) zodat de spijkers goed te herkennen zijn.



* Sleep het nieuwe *Spijkers-Material* naar het *Spijkers* gameobject (*niet naar het Spijkerblok*) in de Hierarchy View. Als het goed is zie je in de Scene View dat de Spijkers nu rood zijn.



## Player wordt geraakt door spijkers

Als de de bal tegen de Spikes aankomt, dan moet de speler duidelijk te zien krijgen dat dit niet de bedoeling is. Dat gaan we doen door de bal omhoog te laten stuiteren en een leven af te trekken van het totaal aantal levens.

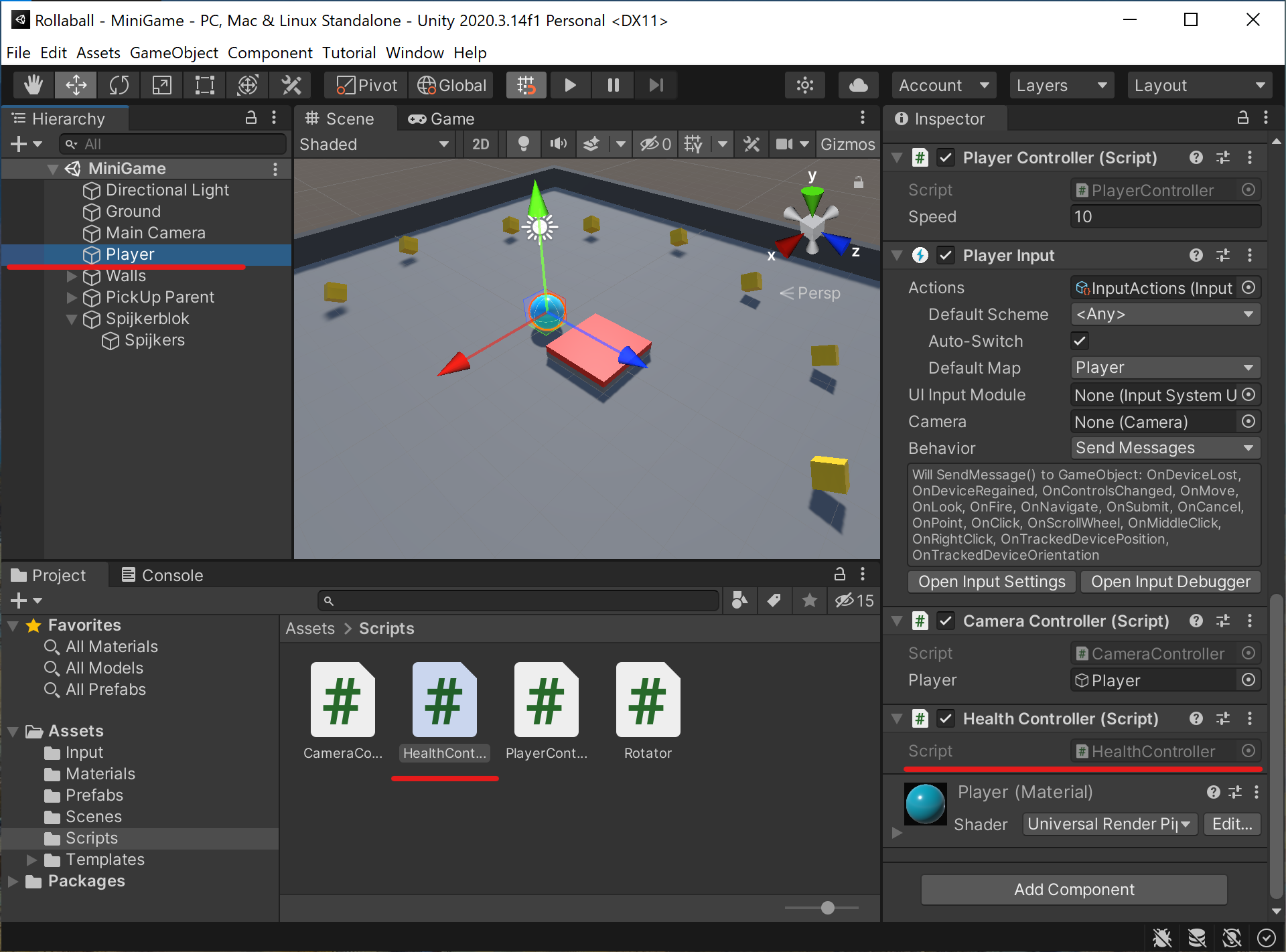
Je gaat deze functionaliteit op zo’n manier implementeren dat het later makkelijker wordt om andere gevaarlijke objecten toe te voegen.

1. We geven de Player een nieuw script: HealthController, met de methode Geraakt die ervoor zorgt dat de bal weg stuitert van het object waardoor de bal werd geraakt.
2. De Spijkers krijgen ook een component die de methode Geraakt van de Player aanroept zodra de bal tegen de Spijker botst.

Hieronder werk je deze twee functionaliteiten stap voor stap uit.

### HealthController maken

* Maak een nieuw script met de naam HealthController in de Scripts-map van het project en voeg dit script toe aan de Player.



* Vervang de gegenereerde code in HealthController door de onderstaande code waarmee de Player omhoog kan weg stuiteren.

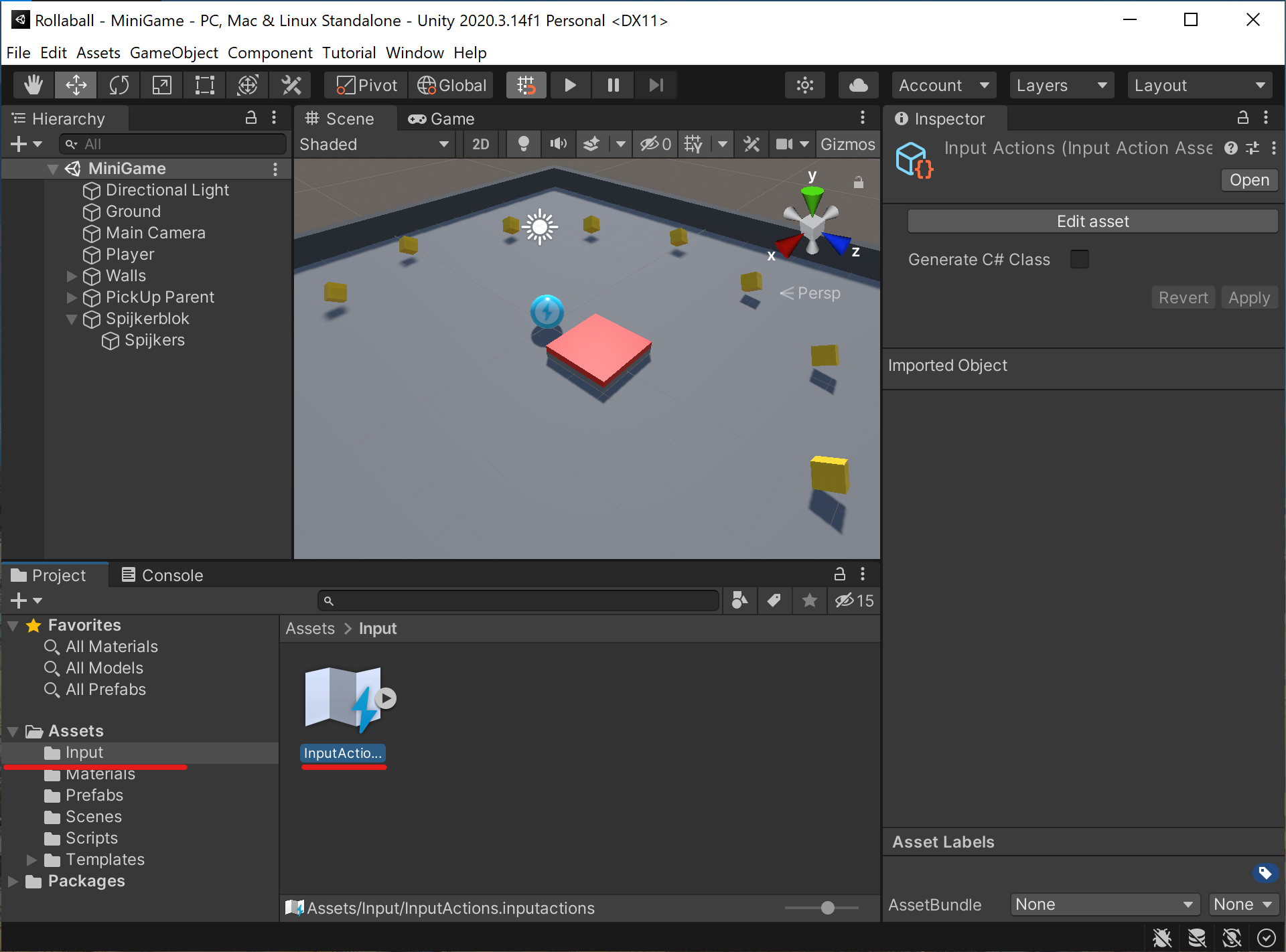
| using System.Collections; using System.Collections.Generic; using UnityEngine;  public class HealthController : MonoBehaviour {  private Rigidbody rb;   void Start()   {  rb = GetComponent<Rigidbody>();  }   public void Geraakt()   {  rb.AddForce(Vector3.up \* 5, ForceMode.Impulse);  } } |
| --- |

Een opmerking bij de code.Omdat we een kracht gebruiken om de Player omhoog te laten stuiteren, hebben we het *RigidBody*-component van de Player nodig die we ophalen in de methode *Start*. De code in het PlayerController-script haalt dit component ook op, maar de code in een ander script kan daar niet bij. Daarom moet je in elk script elk component dat je nodig hebt apart ophalen.

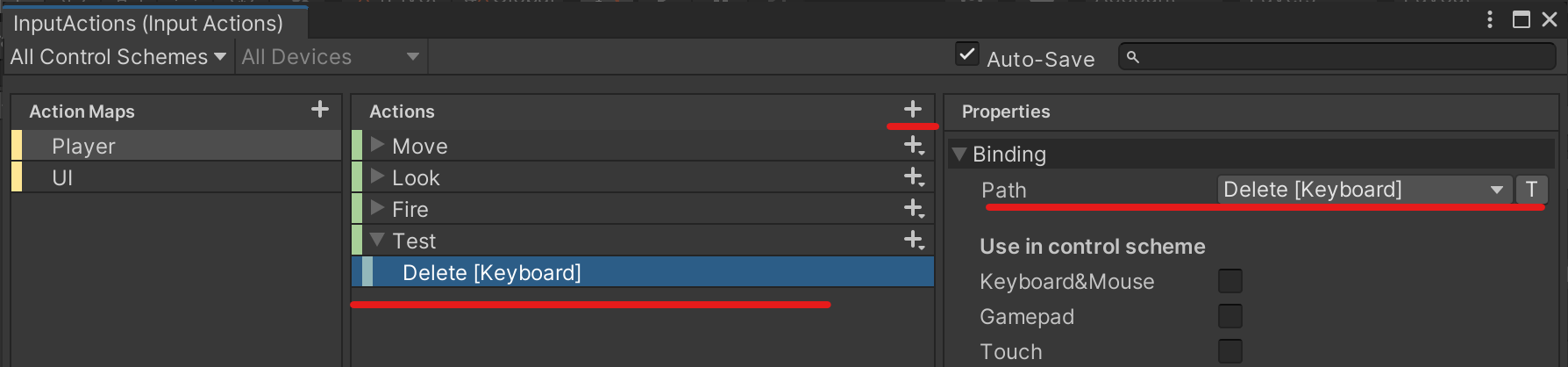
### Geraakt Testen

Voordat je deze methode aan gaat roepen als de Player de Spikes raakt, is het handig om te controleren of deze methode werkt zoals je verwacht. Om dit te bereiken maak je een nieuwe Action in de InputActions.

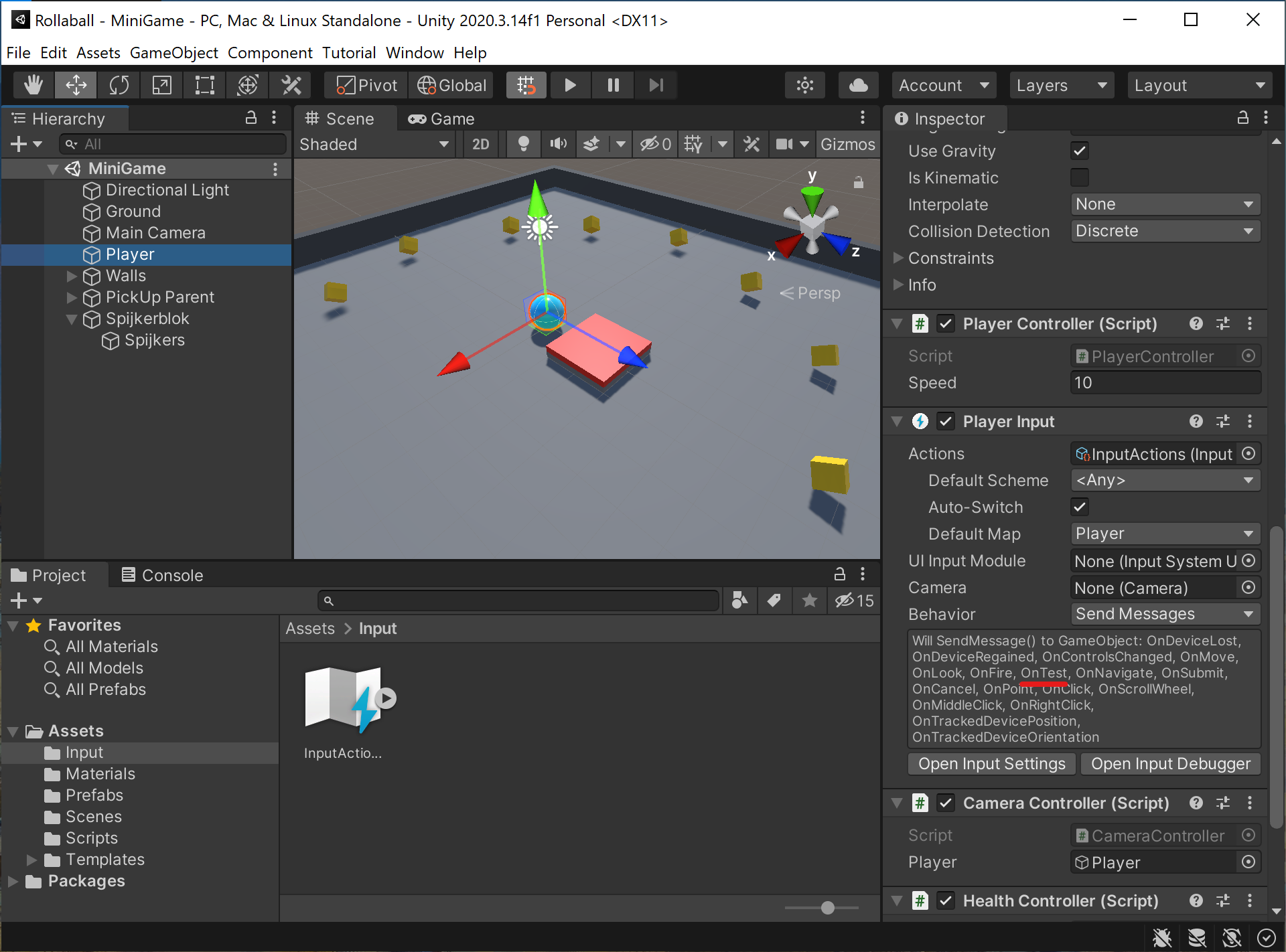
* Dubbelklik op InputActions (in de map Assets/Input)



* Voeg een Test-Action toe en koppel deze aan de Delete knop van het toetsenbord (of een andere knop waarvan je verwacht dat deze normaal gesproken niet in het spel gebruikt wordt).



* Open het Player gameobject en controleer dat *OnTest* te zien is bij het Player Input Component



* Open het script HealthController en voeg de methode OnTest toe waarin je de methode Geraakt aanroept zoals in onderstaande code te zien is.

| // Testfunctie om de speler op een willekeurig moment te kunnen raken    void OnTest()  {  Geraakt();  } |
| --- |

* Test het spel. Als het goed is springt de Player nu omhoog als je op Delete drukt.

### TakeDamage flexibeler maken

Het is handig als we Geraakt ook kunnen gebruiken voor andere gevaren dan Spikes die omhoog gericht staan. Daarom maken we de richting van de kracht flexibel door er een parameter van te maken

* Verander de methode Geraakt zoals in onderstaande code te zien is:

| public void Geraakt(Vector3 wegschietRichting)   {  rb.AddForce(wegschietRichting \* 5, ForceMode.Impulse);  } |
| --- |

* Verander de aanroep van Geraakt in OnTest door een willekeurige richting mee te geven

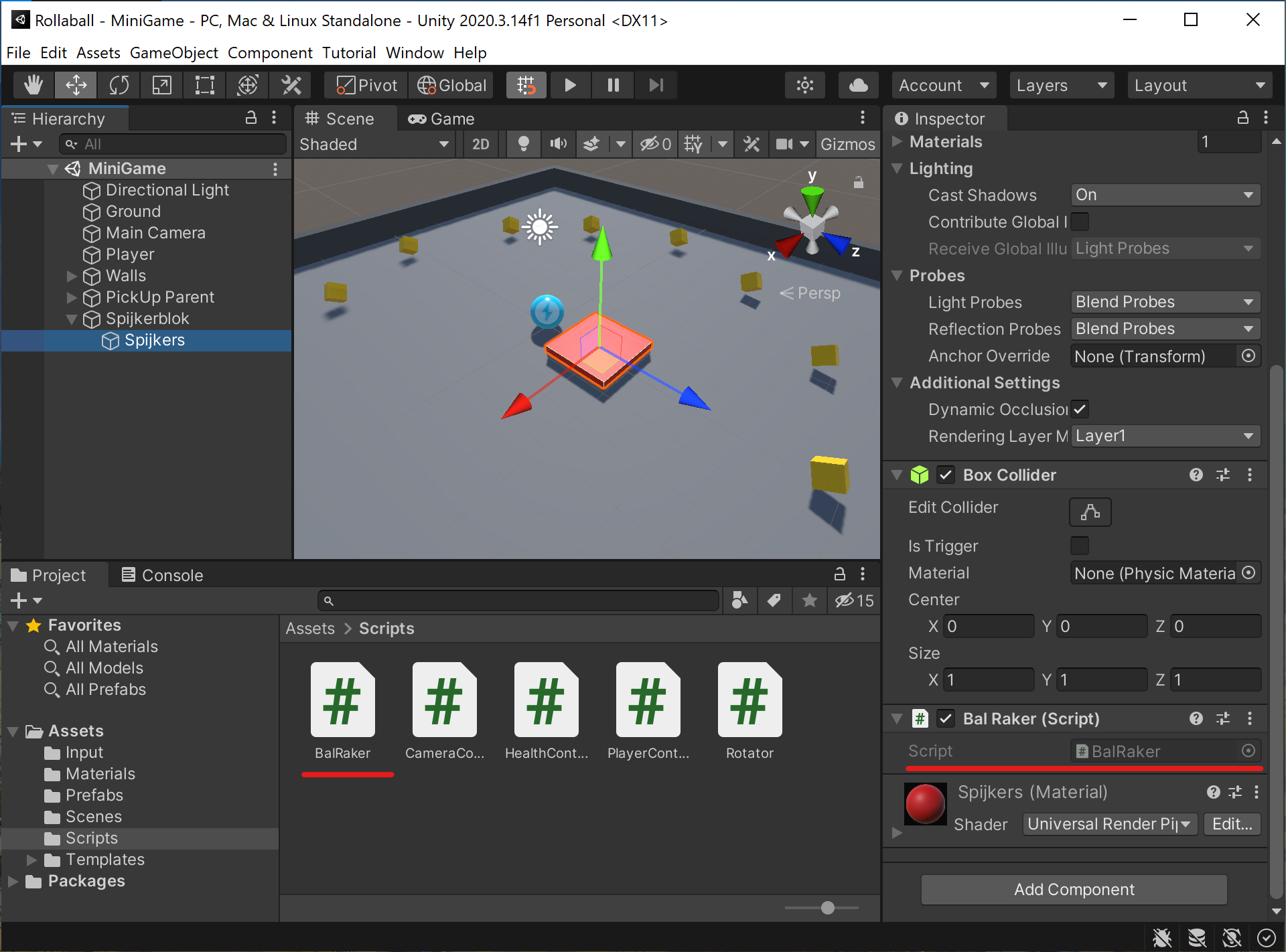
| // Testfunctie om de speler op een willekeurig moment te kunnen raken  void OnTest()  {  Geraakt(Vector3.right);  } |
| --- |

* Test het spel. Als het goed is, springt de bal nu in de richting die je hebt meegegeven zodra je op delete drukt.
* Verander de richting in de aanroep van Geraakt en controleer of het spel nog steeds werkt.
* Voorlopig hebben we Update niet meer nodig, dus zet de methode OnTest in commentaar.

### Spijkers gevaarlijk maken

Nu je zeker weet dat de Player geraakt kan worden ga je ervoor zorgen dat de bal door de spijkers geraakt wordt.

* Maak een nieuw script BalRaker toe in de Scripts map en voeg dit script toe aan het Spijkers Game Object:



* Open dit script en vervang de code door onderstaande code waarmee de Spijkers kunnen detecteren dat ze een ander object raken.

| using System.Collections; using System.Collections.Generic; using UnityEngine;  public class BalRaker : MonoBehaviour {  void OnCollisionEnter(Collision other)   {  Debug.Log("We prikken een slachtoffer");  } } |
| --- |

* Test het spel door met de bal tegen het spijkerblok te botsen en controleren dat je de “We prikken een slachtoffer” in de console ziet verschijnen.

De methode OnCollisionEnter wordt automatisch door Unity aangeroepen zodra er een botsing optreedt. In de variabele other zit informatie over het object dat tegen de Spijkers botst. In theorie kan dit elk ander gameobject zijn, maar op dit moment kan dit alleen Player zijn, omdat andere gameobjecten zich niet verplaatsen.

In de volgende stap ga je de naam van het slachtoffer dat in de spijkers valt ophalen en tonen in de console.

* Verander de code in deze methode door onderstaande code

void OnCollisionEnter(Collision other)

{

string naamSlachtoffer = other.gameObject.name;

Debug.Log("We prikken: ");

Debug.Log(naamSlachtoffer);

}

* Test het spel en controleer dat je “We prikken” en “Player” in de console ziet verschijnen.

### Methode uit een ander object aanroepen

Het script BalRaker moet de methode Geraakt uit het Script HealthController van het Player gameobject kunnen aanroepen. In de vorige stap heb je gezien hoe je de naam van de Player kunt ophalen als deze met de spijkers botst en in deze stap leer je hoe je het script HealthController ophaalt.

* Voeg onderstaande code toe

void OnCollisionEnter(Collision other)

{

string naamSlachtoffer = other.gameObject.name;

Debug.Log("We prikken: ");

Debug.Log(naamSlachtoffer);

HealthController healthController =

other.gameObject.GetComponent<HealthController>();

healthController.Geraakt(Vector3.up);

}

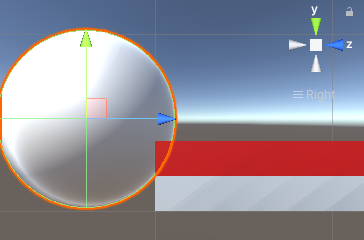
}

* Test het spel en controleer dat de Player omhoog stuitert als de spijkers geraakt worden.

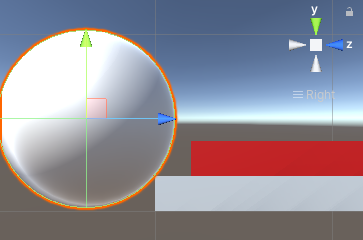
## 

## Spijkers kleiner maken

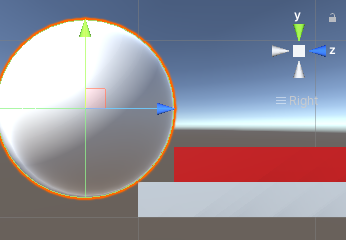
Wanneer de bal tegen het spijkerblok aanrolt van de zijkant, wordt het al geraakt door de spijkers terwijl dat waarschijnlijk niet de bedoeling is. In de afbeelding kun je zien waardoor dat komt:



Om dit te voorkomen veranderen we de grootte van de spijkers zodat een Speler die voorzichtig tegen een spijkerblok botst niet geraakt wordt:

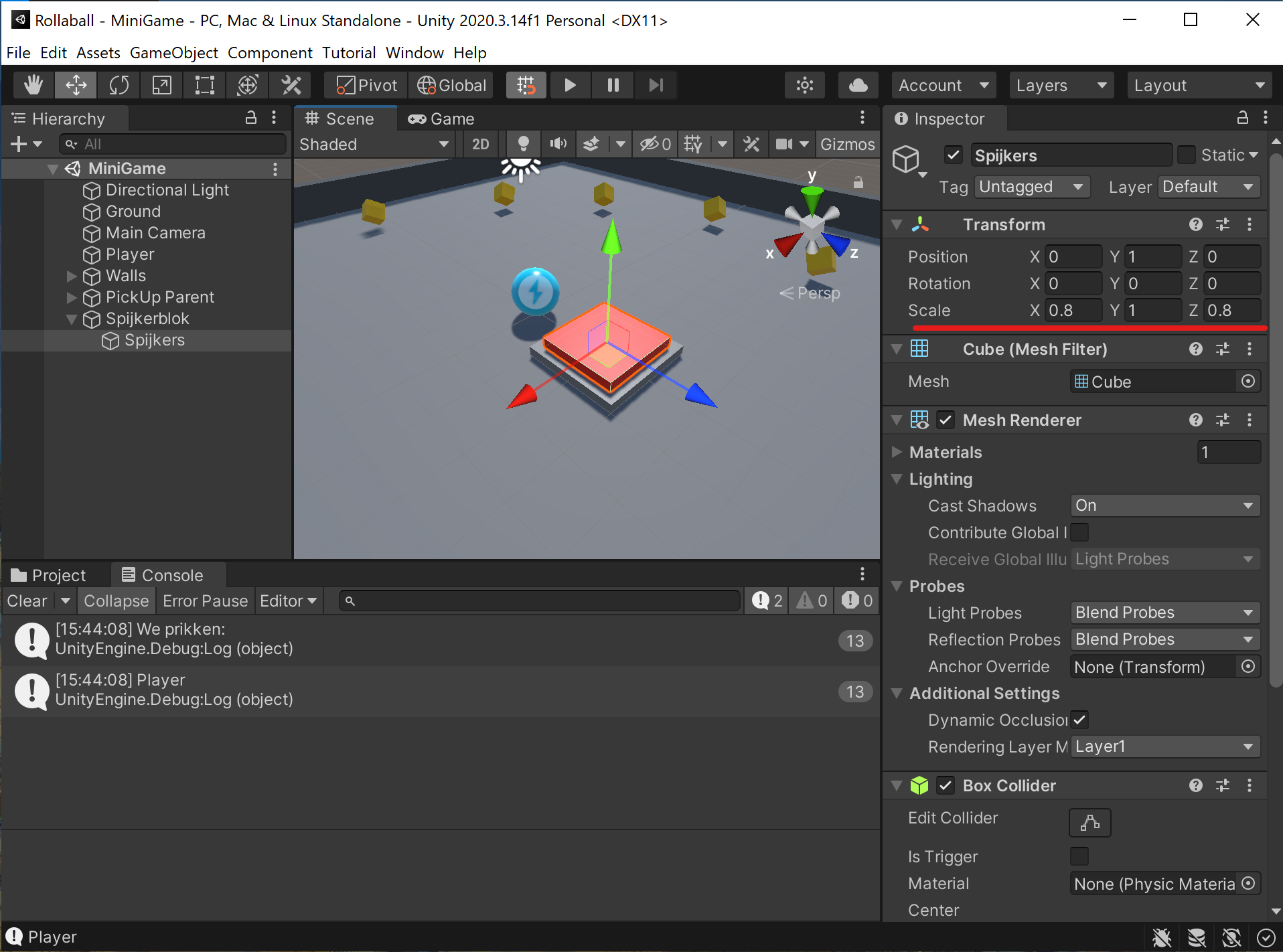


Als de bal wel de bovenkant van het blok raakt, maar de Spikes net niet, dan wordt dit niet bestraft.



Als de bal echter veel vaart heeft, stuitert de bal een beetje omhoog en worden de spijkers wel geraakt.

* Verander de X-Scale en de Z-Scale van het Spijkers gameobject in 0.8. Als het goed is ziet de Transform Component van de Spikes er als volgt uit.



* Test het spel. Als je vindt dat de speler te voorzichtig moet zijn, kun je de spijkers nog wat kleiner maken, of het blok zelf hoger. Wanneer je tevreden bent, kun je de waarden laten voor wat ze zijn.
* Maak een prefab van het Spijkerblok door dit gameobject naar de map Prefabs te slepen

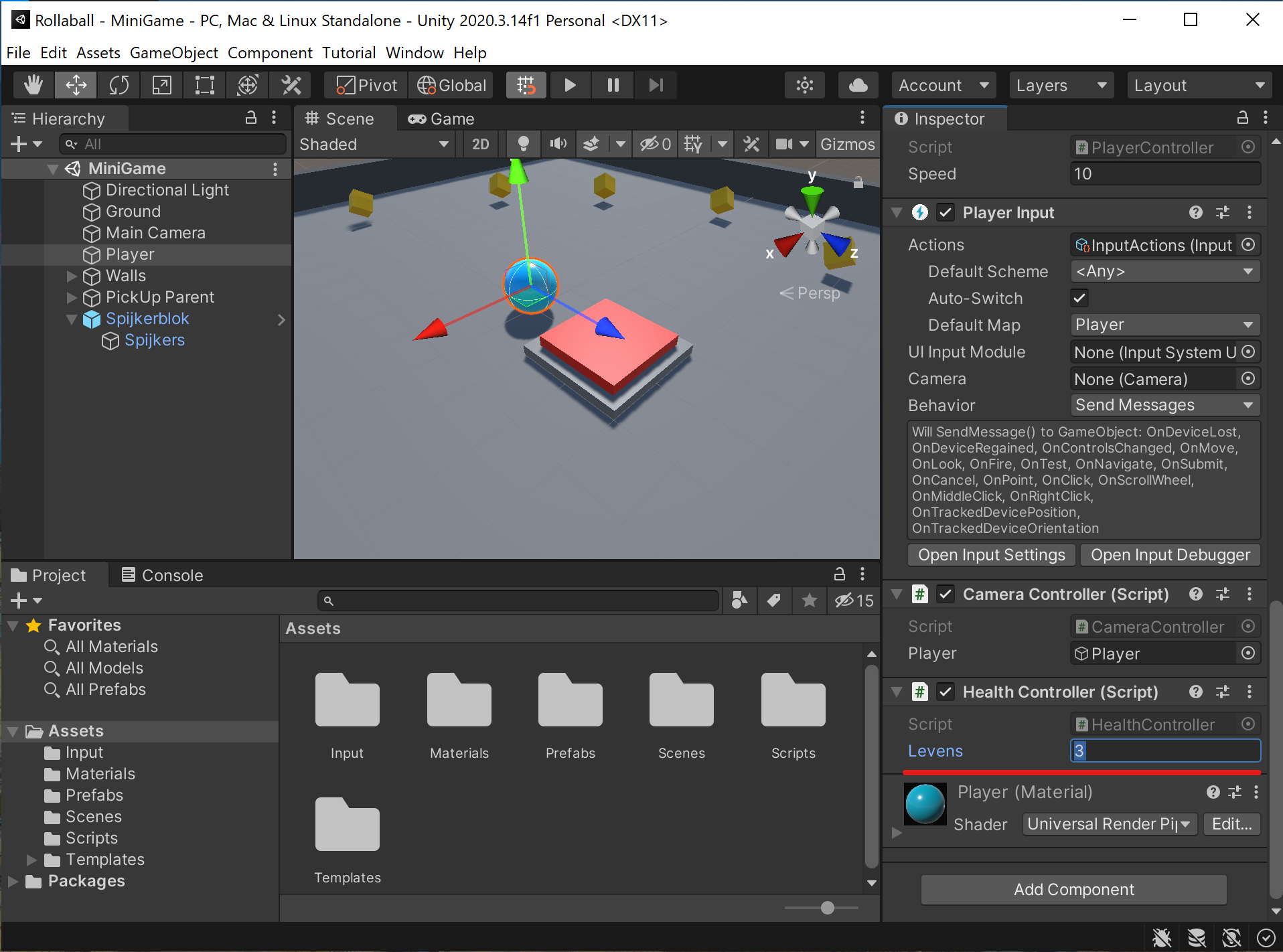
## Gezondheid toevoegen

De Player kan wel geraakt worden, maar het spel bestraft dit nog niet. In deze laatste stap voegen we een aantal levens aan de speler toe en herstarten we het level als de speler minder dan 0 levens heeft.

* Voeg de publieke variabele levens toe aan het script HealthController en trek 1 van deze variabele af als de speler wordt geraakt.

| public class HealthController : MonoBehaviour  {  private Rigidbody rb;  //De waarde van levens kan bepaald worden in de editor  public int levens;  void Start()  {  rb = GetComponent<Rigidbody>();  }  public void Geraakt(Vector3 wegschietRichting)  {  rb.AddForce(wegschietRichting \* 5, ForceMode.Impulse);  levens--;  } |
| --- |

* Zet de waarde van levens op 3 (of een andere waarde) in de Unity editor.



* Speel het spel en test dat de levens lager wordt als de spijkers worden geraakt.

### Level herstarten

Je kunt dit level herstarten met de ingebouwde methode:

SceneManager.LoadScene("MiniGame");

Dit werkt alleen als je de package UnityEngine.SceneManagement in het script importeert.

* Voeg onderstaande regel toe aan het script HealthController

| using System.Collections; using System.Collections.Generic; using UnityEngine; using UnityEngine.SceneManagement; *//Deze regel moet je toevoegen*  public class HealthController : MonoBehaviour { |
| --- |

* Voeg de code toe om het level te herstarten in de methode Geraakt zodra levens kleiner is geworden dan 0.
* Test het spel.

# Extra Opgaven

## Level maken

Maak een interessant level met meerdere spijkerblokken en moeilijk te bereiken pickups.

Als toch blijkt dat de spijkers wat kleiner moeten worden, of het blok wat hoger, pas dan de prefab aan zodat alle spijkerblokken automatisch veranderen.

## Spijkerblokken draaien

Maak een nieuw type spijkerblok dat gedraaid is en de bal in een andere richting duwt.

## 