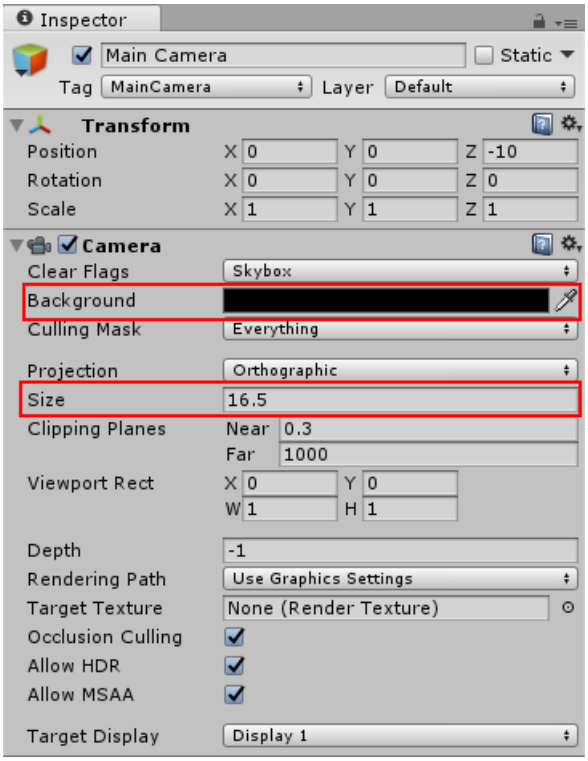
Snake

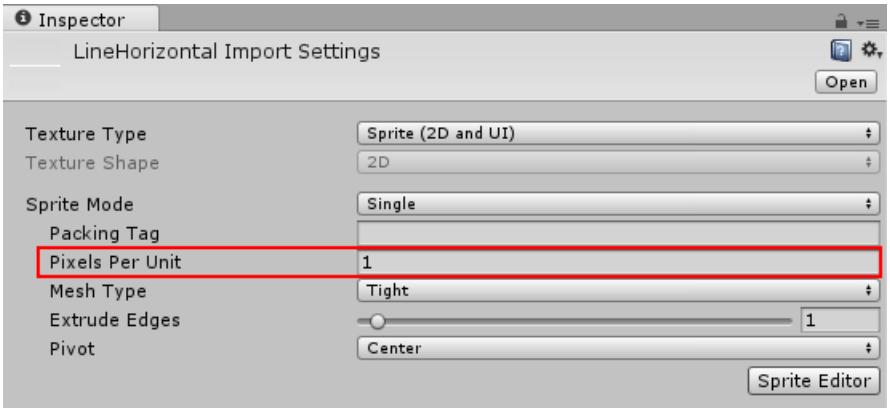
Setting up the Project

If we select the **Main Camera** in the **Hierarchy** then we can set the Background Color to black, adjust the **Size** and the **Position** like shown in the following image:



*Note: **Size** is the Camera's zoom factor.*

For all *.png files, set the **Pixels per Unit** to 1. There are 3 .png files. Be sure to hit Apply!

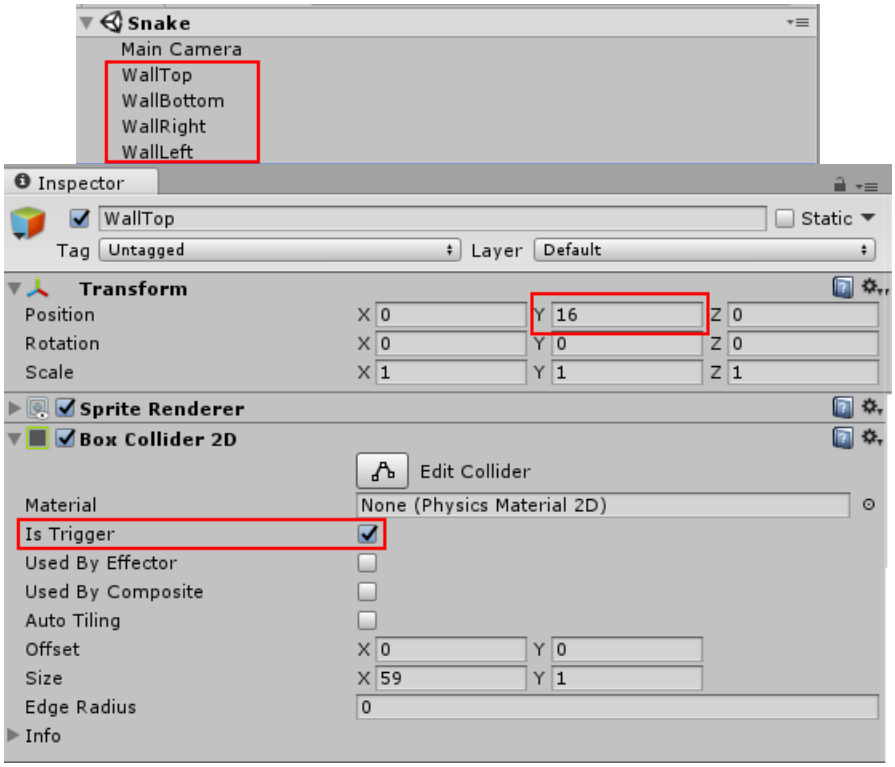


*Note: **Pixels Per Unit** is the ratio between one pixel in the image and one unit in the world. The Snake will have a size of 1x1 pixel, which should be 1 unit in the game world. This is why we will use a **Pixels Per Unit** value of **1** for all our textures.*

Creating the Walls

Place two **HoriontalLine.png** and two **VerticalLine.png** and rename them accordingly as show below. For the four walls, adjust the positions to create a box, attach a **Box Collider 2D** to each of them, and check the **Is Trigger** property of the **Box Colllider 2D** . The positions used are shown below along with an image of the **WallTop**.

WallTop: Transform Position **Y = 16**
WallBottom: Transform Position **Y = -16**
WallRight: Transform Position **X = 29**
WallLeft: Transform Position **X = -29.**



*Note: These **Transform Position** values must be integers! These positions will be used later on.*

Creating a Food Prefab

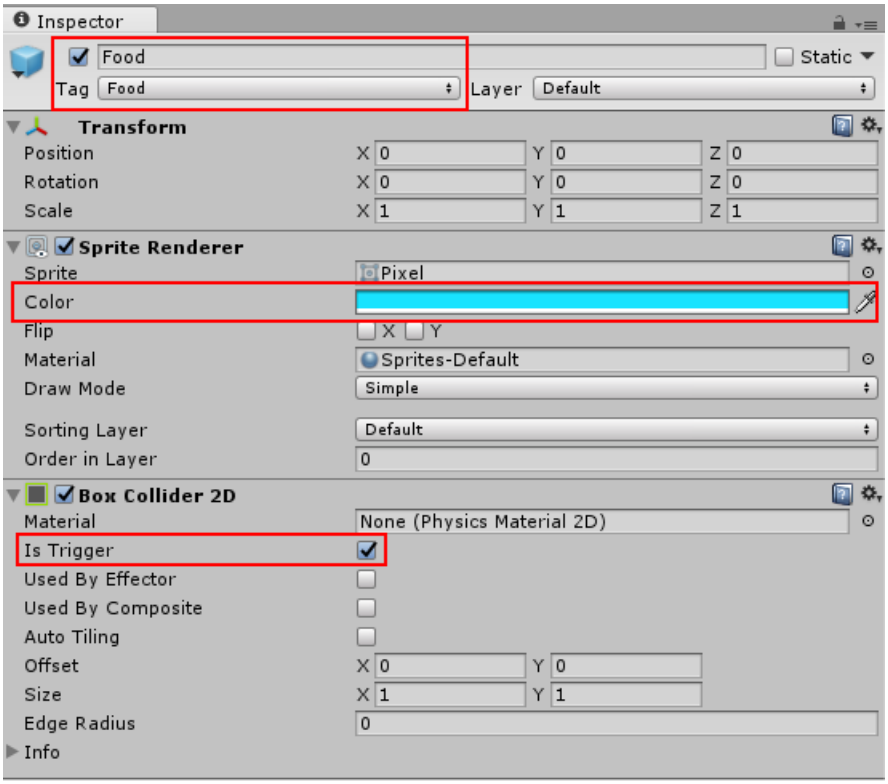
We will randomly spawn food for the Snake as time passes. Alright, let's drag the pixel image into the Scene to create a **GameObject** and rename the **GameObject** to **Food**.

The Snake should receive some kind of information whenever it collides with food. This means that the food has to be part of the physics world, which can be done with a **Collider**.

A **GameObject** without a **Collider** is just a visual thing, its not part of the physics world. A **GameObject** with a **Collider** is part of the physics world, just like a wall. It will cause other things to collide with it, and it will trigger the **OnCollisionEnter2D** event. A **GameObject** with a **Collider** that has **Is Trigger** checked will **not** cause other things to collide with it, but it will still trigger the **OnTriggerEnter2D** event.

The Snake should get notified when it walks through food, but it's not supposed to collide with it like if the food was a wall. So let's select the food in the **Hierarchy** and then choose **Add Component**->**Physics 2D**->**Box Collider 2D** in the **Inspector** and enable **Is Trigger**.

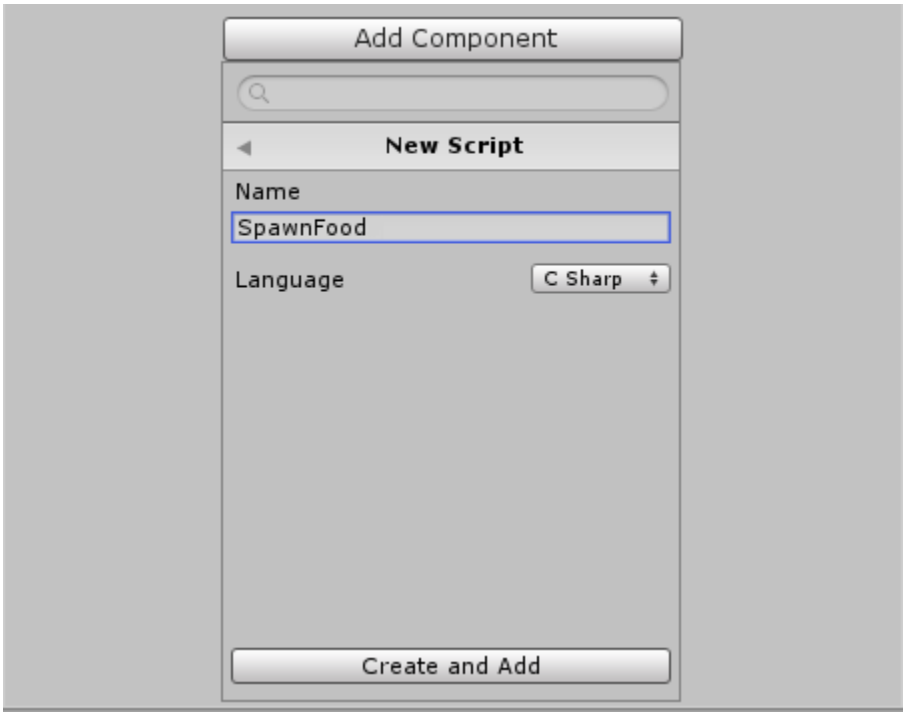
Choosing **Tag->Add Tag...->Food** will be useful later on for checking if the Snake collided with Food or something else.



Once these are done, move the **Food** from the **Hierarchy** to the **Project** area to create a **Prefab** of the **Food**.

Spawning Food

Let's spawn new food at some random position every few seconds. This kind of behavior can be implemented with a Script, so let's create a **SpawnFood** Script. The Script should be in the Scene all the time, so we will keep it simple and add it to the **Main Camera** *(because it will be in the Scene all the time, too)*. Let's select the **Main Camera** in the **Hierarchy** and then click on **Add Component->New Script**, name it **SpawnFood** and select **CSharp** for the language.



Double click on the **Script** in the **Project Area** to open it. Remove the **Update()** function since we will not be using it.

In order to spawn food, our **Script** needs to know about the **Food Prefab** so we will add it as a public variable of type **GameObject.**

The food should be spawned within the walls, not outside. So we will also need a variable for each of the walls so that our Script knows their positions.

```
using UnityEngine;
using System.Collections;

public class SpawnFood : MonoBehaviour {
    // Food Prefab to spawn
    public GameObject food;

    // Walls for spawning area
    public Transform wallTop;
    public Transform wallBottom;
    public Transform wallLeft;
    public Transform wallRight;

    // Use this for initialization
    void Start () {
```

```
    }
}
```

*Note: they are already of type **Transform** so we don't have to write **wallTop.transform.position** all the time. Instead we will be able to access their positions like **wallTop.position**.*

Let's create the **Spawn** function that spawns one piece of food within the walls. At first we will choose the **x** position somewhere randomly between the left and right wall. Then we will choose the **y** position randomly between the top and bottom wall. Afterwards we will instantiate the food Prefab at that position:

```
// Spawning one food
void Spawn() {
    // random x position between left & right wall
    int x = (int)Random.Range(borderLeft.position.x, borderRight.position.x);

    // random y position between top & bottom wall
    int y = (int)Random.Range(borderBottom.position.y, borderTop.position.y);

    // create food at (x,y) with a default rotation (no rotation)
    Instantiate(foodPrefab, new Vector2(x, y),Quaternion.identity);
}
```

*Note: **x** and **y** are rounded via **(int)** to make sure that the food is always spawned at a position like **(1, 2)** but never at something like **(1.234, 2.74565)**.*

Now let's make sure that our Script calls the **Spawn** function every few seconds. We can do so by using InvokeRepeating:

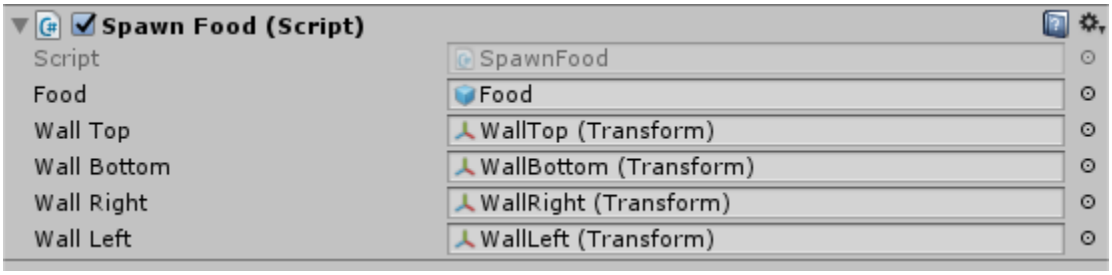
```
// Use this for initialization
void Start () {
    // after 3 seconds, Food is spawned every 4 seconds by calling Spawn()
    InvokeRepeating("Spawn", 3, 4);
}
```

Full script:

```
using UnityEngine;
using System.Collections;

public class SpawnFood : MonoBehaviour {
    // Food Prefab to spawn
    public GameObject food;

    // Walls for spawning area
    public Transform wallTop;
    public Transform wallBottom;
    public Transform wallLeft;
    public Transform wallRight;

    // Use this for initialization
    void Start () {
        // after 3 seconds, Food is spawned every 4 seconds by calling Spawn()
        InvokeRepeating("Spawn", 3, 4);
    }

    // Spawning one food
    void Spawn() {
        // random x position between left & right wall
```

```
        int x = (int)Random.Range(borderLeft.position.x, borderRight.position.x);

        // random y position between top & bottom wall
        int y = (int)Random.Range(borderBottom.position.y, borderTop.position.y);

        // create food at (x,y) with a default rotation (no rotation)
        Instantiate(foodPrefab, new Vector2(x, y),Quaternion.identity);
    }
}
```

Drag and drop the components needed by the Script.



Creating the Snake Head

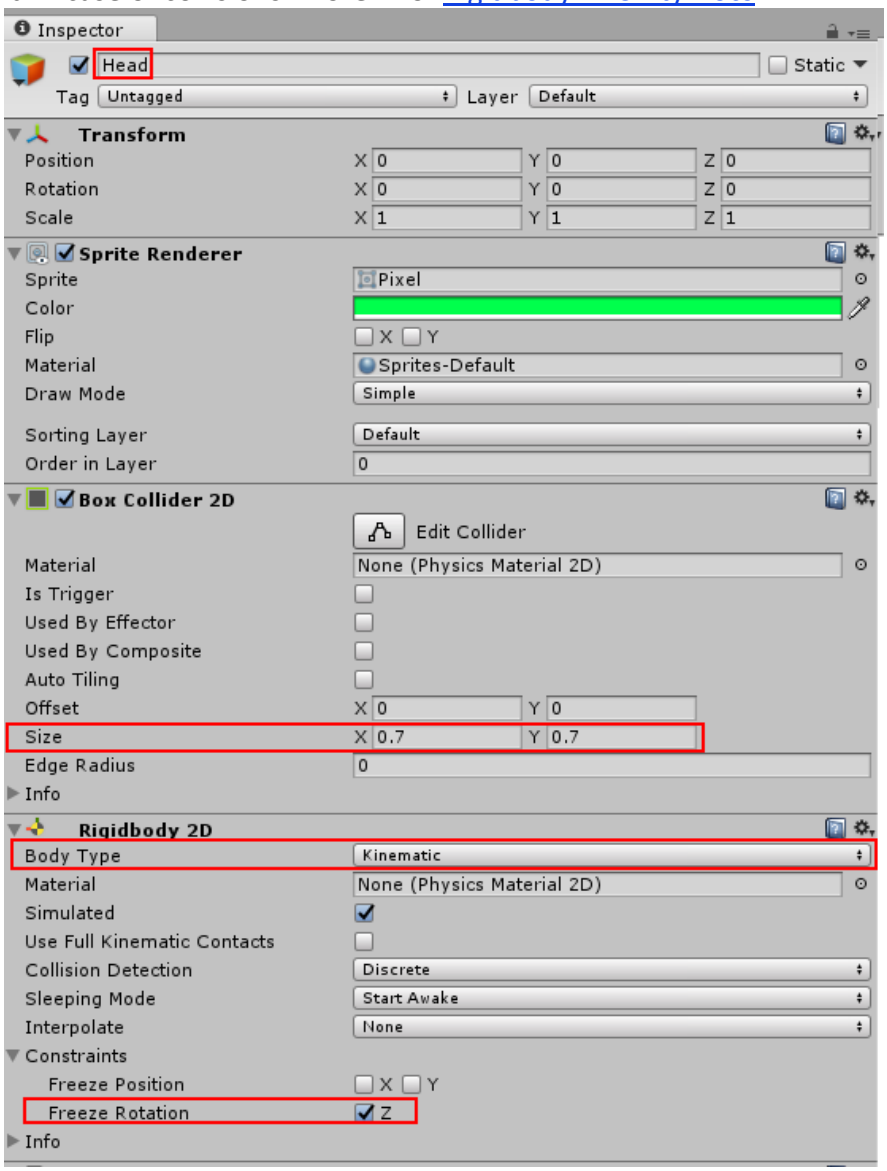
So far it's just the Snake's head, so let's rename it to **Head** to keep things clean. The snake should be part of the physics world, which means that we need to add a Collider to it again, so let's select **Add Component->Physics 2D->Box Collider 2D**.

*Note: the Collider has the size **(0.7, 0.7)** instead of **(1, 1)** so that it doesn't collide with other parts of the snake that are right next to it. We simply want to give Unity some space.*

Now the snake is also supposed to move around. As a rule of thumb, everything in the physics world that is supposed to move, needs a **Rigidbody**. A Rigidbody takes care of things like gravity, velocity and movement forces. We can add one by selecting **Add Component->Physics 2D->Rigidbody 2D**. We will use the following settings for it:

Notes:

- The Rigidbody's **Gravity Scale** is **0** because we don't want the snake to fall towards the bottom of the screen all the time.
- The **Is Kinematic** option disables the physical behavior of the Rigidbody, so it doesn't react to gravity or collisions. We only need to know if the snake collided with something. We don't need Unity's physics to push things around in case of collisions. More info: [Rigidbody2D Unity Docs].



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Linq;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class Snake : MonoBehaviour {

    // direction of Snake's movement
    private Vector2 direction;

    // keeping track of the tail
    private List<Transform> tailList = new List<Transform>();

    // flag for checking if the Snake ate Food
    private bool ate = false;

    // Snake's Tail Prefab for eating Food
    public GameObject tailPrefab;

    public Text gameOverText;
    public Text restartText;

    private bool gameOver;
    private bool restart;

    // Use this for initialization
    void Start () {
        gameOver = false;
        restart = false;
        gameOverText.text = "";
        restartText.text = "";

        // Snake starts moving to the right
        direction = Vector2.right;

        // Snake moves every 100ms by calling the Move() function
        InvokeRepeating("Move", 0.1f, 0.1f);
    }

    // Update is called once per frame
    void Update () {
        // Snake direction changes based on pressed key and NO TURNING BACK
        if (Input.GetKey(KeyCode.RightArrow) && (direction != Vector2.left))
        {
            direction = Vector2.right;
        }
        else if (Input.GetKey(KeyCode.LeftArrow) && (direction != Vector2.right))
        {
            direction = Vector2.left;
        }
        else if (Input.GetKey(KeyCode.UpArrow) && (direction != Vector2.down))
        {
            direction = Vector2.up;
```

```
        }
        else if (Input.GetKey(KeyCode.DownArrow) && (direction != Vector2.up))
        {
            direction = Vector2.down;
        }

        if (restart)
        {
            if (Input.GetKeyDown (KeyCode.Space))
            {
                SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex);
            }
        }
        if (gameOver)
        {
            restartText.text = "Press Space for Restart";
            restart = true;
        }
    }

    // Move used to move the Snake
    void Move()
    {
        // save current position of the Snake's Head
        Vector2 v = transform.position;

        // move Snake's Head one space based on the new direction (gap)
        transform.Translate(direction);

        // check if the ate flag is true
        if (ate) {
            // loads the Tail Prefab to be placed where the Snake's Head was (fills gap)
            GameObject gameObject = (GameObject)Instantiate (tailPrefab, v, Quaternion.id
entity);

            // adds the loaded Tail Prefab to the Tail list
            tailList.Insert (0, gameObject.transform);

            // Snake finished eating setting ate flag back to false
            ate = false;
        }
        // check if the Snake have a Tail
        else if (tailList.Count > 0)
        {
            // move last Tail element to where the Head was
            tailList.Last ().position = v;

            // add Tail to the front of the list
             tailList.Insert (0, tailList.Last ());

             // remove Tail from the back of the list
             tailList.RemoveAt (tailList.Count - 1);
         }

    }
```

```
    // Trigger event when another object collides with this object
    void OnTriggerEnter2D(Collider2D other)
    {
        // checks to see if collided object has the Food tag
        if (other.CompareTag("Food"))
        {
            // set ate flag to true
            ate = true;
            // destroy the Food
            Destroy (other.gameObject);
        }
        // game ends if collided with any other object
        else
        {
            gameOver = true;
        }
    }
}
```