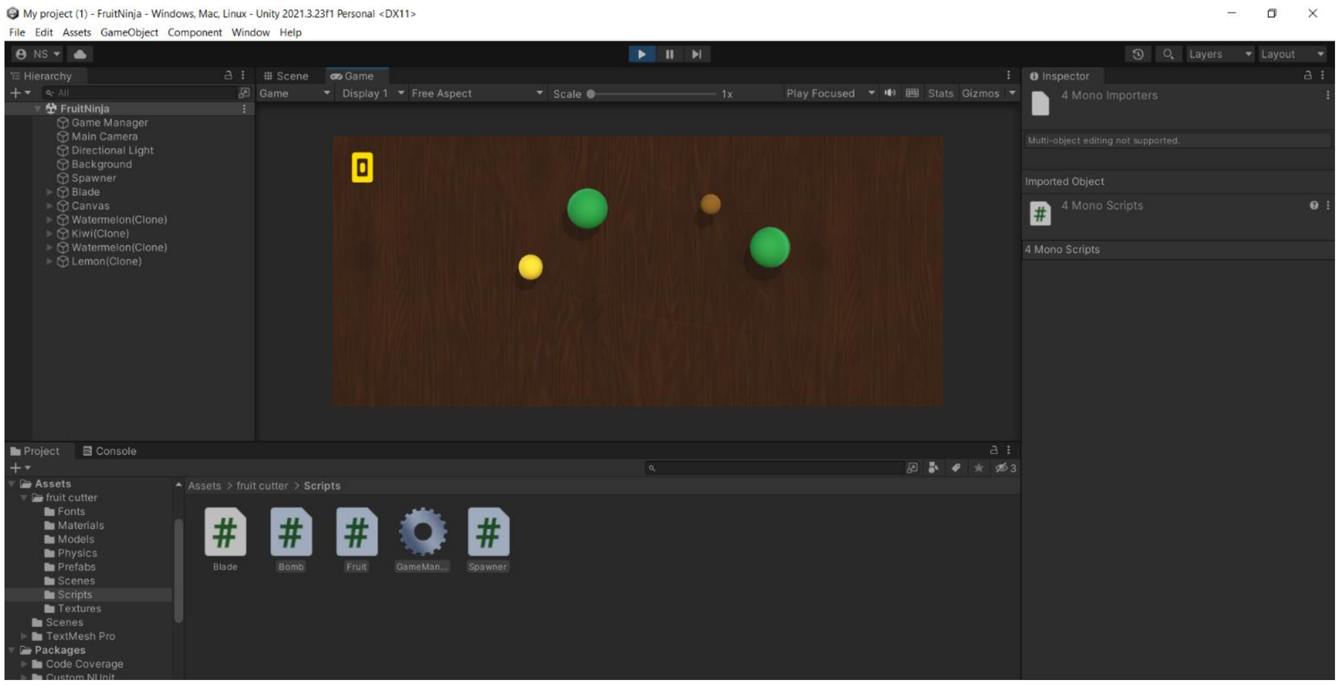
**Lovely Professional University**

# Game Design Document

**Course Code:** CAP819
**Section: KO230**
**Roll no.-** A26
**Reg. No. -**12019256

**Course Name:** Gaming
**Faculty Name:** Pallavi
**Name:** Nirbhay Singh
**Date of Submission**: 26/04/2023

---



## Table of Contents

# Overview

Fruits Ninjas is a fast-paced action game where players take on the role of a skilled ninja who has been tasked with slicing and dicing various types of fruit. The game is set in a lush fruit orchard, where the player must avoid obstacles and slice as many fruits as possible in a set amount of time. The game is easy to learn, but difficult to master, with increasing levels of difficulty and new types of fruit to slice as the player progresses.

## Introduction

Fruits Ninjas is a game that is designed to be fun and engaging for players of all ages. The game is easy to pick up, but difficult to master, with increasing levels of difficulty and new challenges as the player progresses through the game.

### Game Structure

The game is structured around a series of levels, each with increasing difficulty and new challenges for the player to overcome. The game is timed, with the player having a set amount of time to slice as many fruits as possible. The time limit decreases as the player progresses through the game.

## Control

The player controls the ninja using swipes and taps on the touchscreen of their device or mouse clicks and keyboard strokes on PC. The player must slice the fruit as it appears on the screen, using their finger or mouse to draw a line through the fruit.

## Scoring

The player earns points for each fruit they slice, with bonus points awarded for slicing multiple fruits in a single swipe.

## Task Points

The player must slice as many fruits as possible within the given time limit to earn points and progress to the next level.

## Others

**Touch:** The player uses touch to control the ninja and slice the fruit on the screen.

**Actions:** The player can perform the following actions in the game:

Movement: The player controls the ninja's movement by swiping and tapping on the screen or using mouse clicks and keyboard strokes.

**Perception:** The player's perception is used to identify and slice the fruits on the screen.

**Universe Simulation:** The game simulates a fruit orchard, with various types of fruit and obstacles for the player to avoid.

**Physical Universe:** The game simulates a 2D physical universe, with various types of fruit and bombs for the player to avoid.

**Surfaces:** The game simulates various types of surfaces, including wooden table top, fruits and bomb.

**Terrain - Vegetation - Materials:** The game simulates a fruit orchard with different types of.

**Trail Blade:** Its takes as 0.05 for blade slicing path degin shown in small form.

**Inanimate Objects:** The game includes various inanimate objects, including fruits and bombs.

**Landscape:** The game is set in a lush fruit orchard with various types of Fruits and obstacles.

## Reference:

- Unity's official tutorial on creating a fruit ninja game: This tutorial provides step-by-step instructions on how to create a basic fruit slicing game using Unity game engine. It covers aspects such as spawning fruits, slicing fruits with touch input, and adding scoring and game over mechanics. You can find the tutorial here:

  https://learn.unity.com/tutorial/fruit-slicer-game-unity-tutorial.

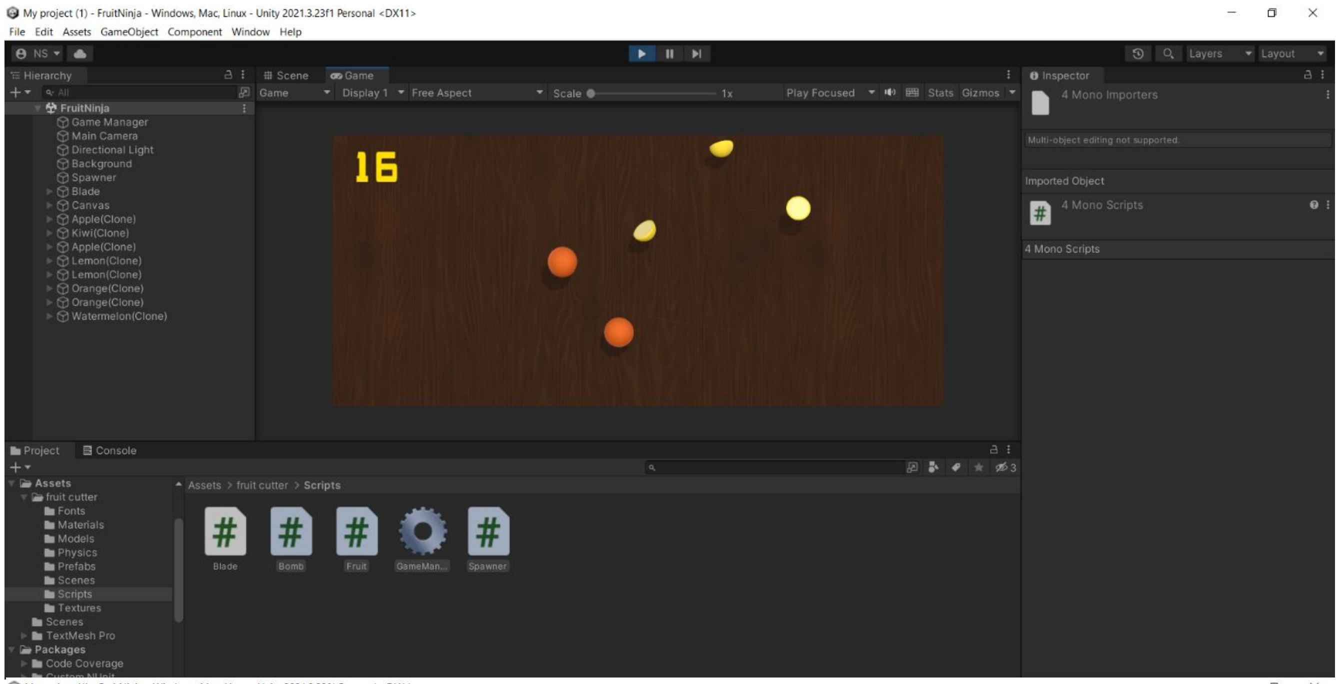
- Ray Wenderlich's tutorial on creating a fruit ninja game: This tutorial is focused on creating a Fruit Ninja clone using Swift and the SpriteKit framework. It covers topics such as slicing fruits, spawning them randomly, scoring points, and adding sound effects. You can find the tutorial here:

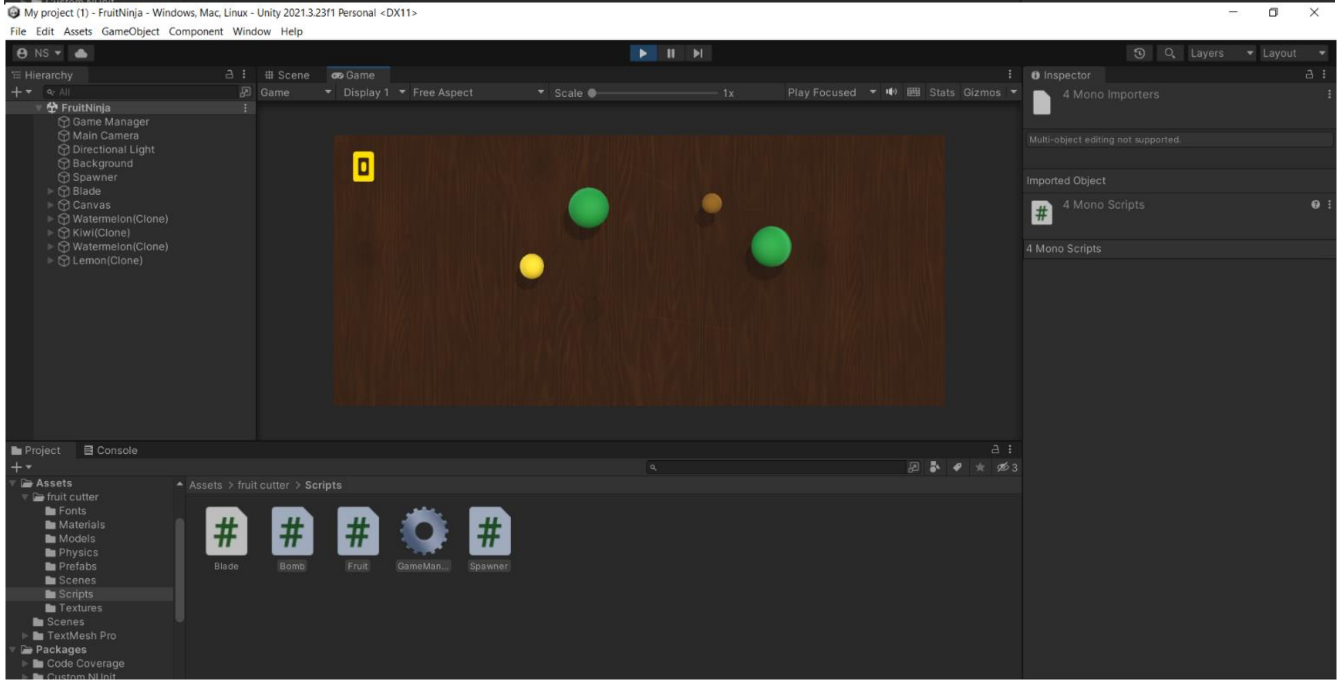
  https://www.raywenderlich.com/126-how-to-make-a-game-like-fruit-ninja-with-the-spritekit-framework.

- Udemy's course on creating a fruit ninja game: This course provides a comprehensive guide to creating a Fruit Ninja-style game using Unity. It covers topics such as creating game assets, slicing fruits, adding power-ups, creating game modes, and monetization strategies. You can find the course here:
  https://www.udemy.com/course/unity3d-fruits-slicing-game/.

# Simulation

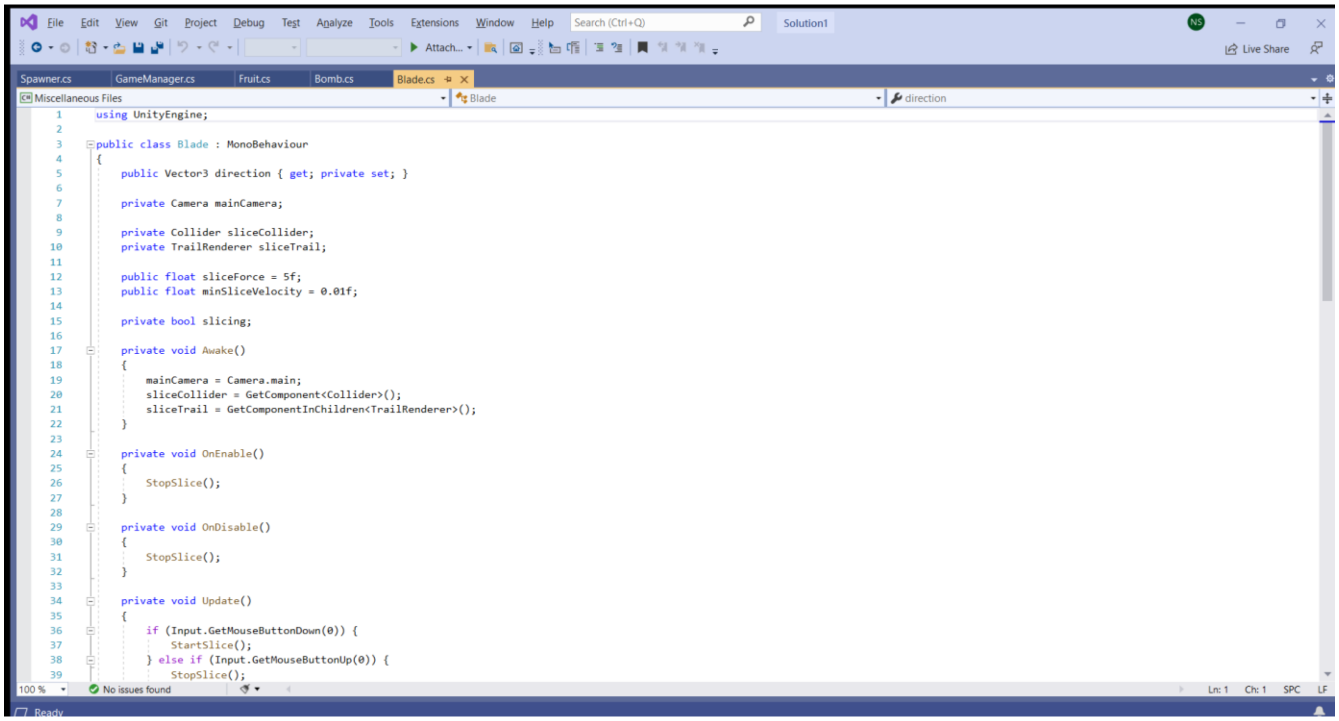
**Game Design Screenshot:**


My project (1) - FruitNinja - Windows, Mac, Linux - Unity 2021.3.23f1 Personal <DX11>
File Edit Assets GameObject Component Window Help
Hierarchy
FruitNinja
Game Manager
Main Camera
Directional Light
Background
Spawner
Blade
Canvas
Apple(Clone)
Kiwi(Clone)
Apple(Clone)
Lemon(Clone)
Lemon(Clone)
Orange(Clone)
Orange(Clone)
Watermelon(Clone)
Scene
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Stats
Gizmos
16
Inspector
4 Mono Importers
Multi-object editing not supported.
Imported Object
4 Mono Scripts
4 Mono Scripts
Project
Console
Assets
fruit cutter
Fonts
Materials
Models
Physics
Prefabs
Scenes
Scripts
Textures
Scenes
TextMesh Pro
Packages
Code Coverage
Assets > fruit cutter > Scripts
Blade
Bomb
Fruit
GameMan...
Spawner


My project (1) - FruitNinja - Windows, Mac, Linux - Unity 2021.3.23f1 Personal <DX11>
File Edit Assets GameObject Component Window Help
Hierarchy
FruitNinja
Game Manager
Main Camera
Directional Light
Background
Spawner
Blade
Canvas
Watermelon(Clone)
Kiwi(Clone)
Watermelon(Clone)
Lemon(Clone)
Scene
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Stats
Gizmos
0
Inspector
4 Mono Importers
Multi-object editing not supported.
Imported Object
4 Mono Scripts
4 Mono Scripts
Project
Console
Assets
fruit cutter
Fonts
Materials
Models
Physics
Prefabs
Scenes
Scripts
Textures
Scenes
TextMesh Pro
Packages
Code Coverage
Assets > fruit cutter > Scripts
Blade
Bomb
Fruit
GameMan...
Spawner

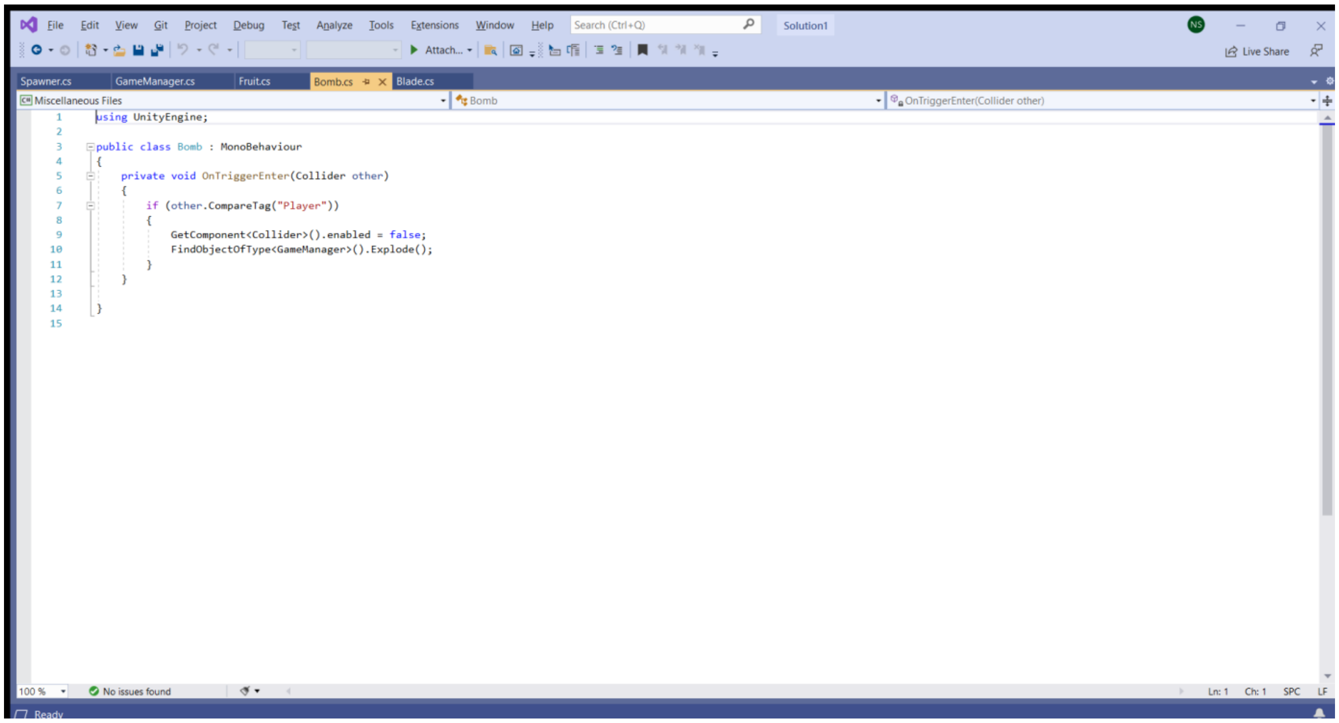
## Script

### a) Blade



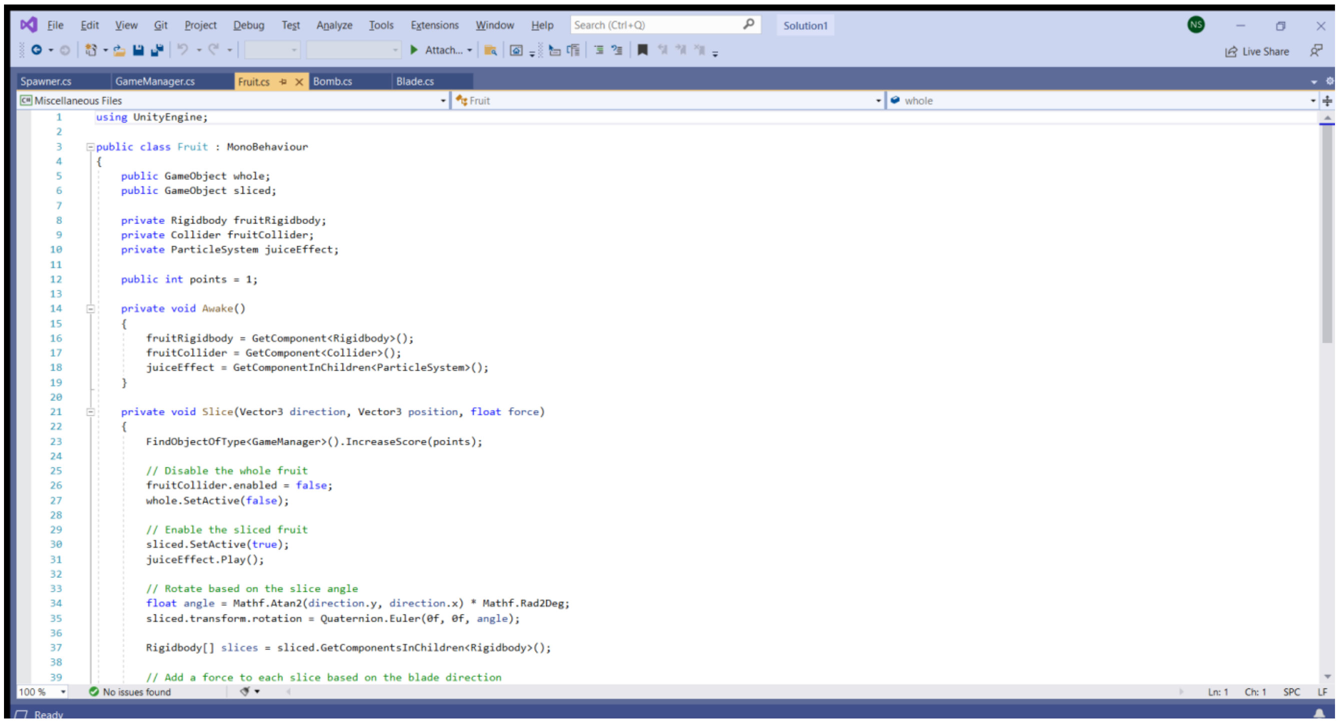
```
using UnityEngine;

public class Blade : MonoBehaviour
{
    public Vector3 direction { get; private set; }

    private Camera mainCamera;

    private Collider sliceCollider;
    private TrailRenderer sliceTrail;

    public float sliceForce = 5f;
    public float minSliceVelocity = 0.01f;

    private bool slicing;

    private void Awake()
    {
        mainCamera = Camera.main;
        sliceCollider = GetComponent<Collider>();
        sliceTrail = GetComponentInChildren<TrailRenderer>();
    }

    private void OnEnable()
    {
        StopSlice();
    }

    private void OnDisable()
    {
        StopSlice();
    }

    private void Update()
    {
        if (Input.GetMouseButtonDown(0)) {
            StartSlice();
        } else if (Input.GetMouseButtonUp(0)) {
            StopSlice();
```

### b) Bombs



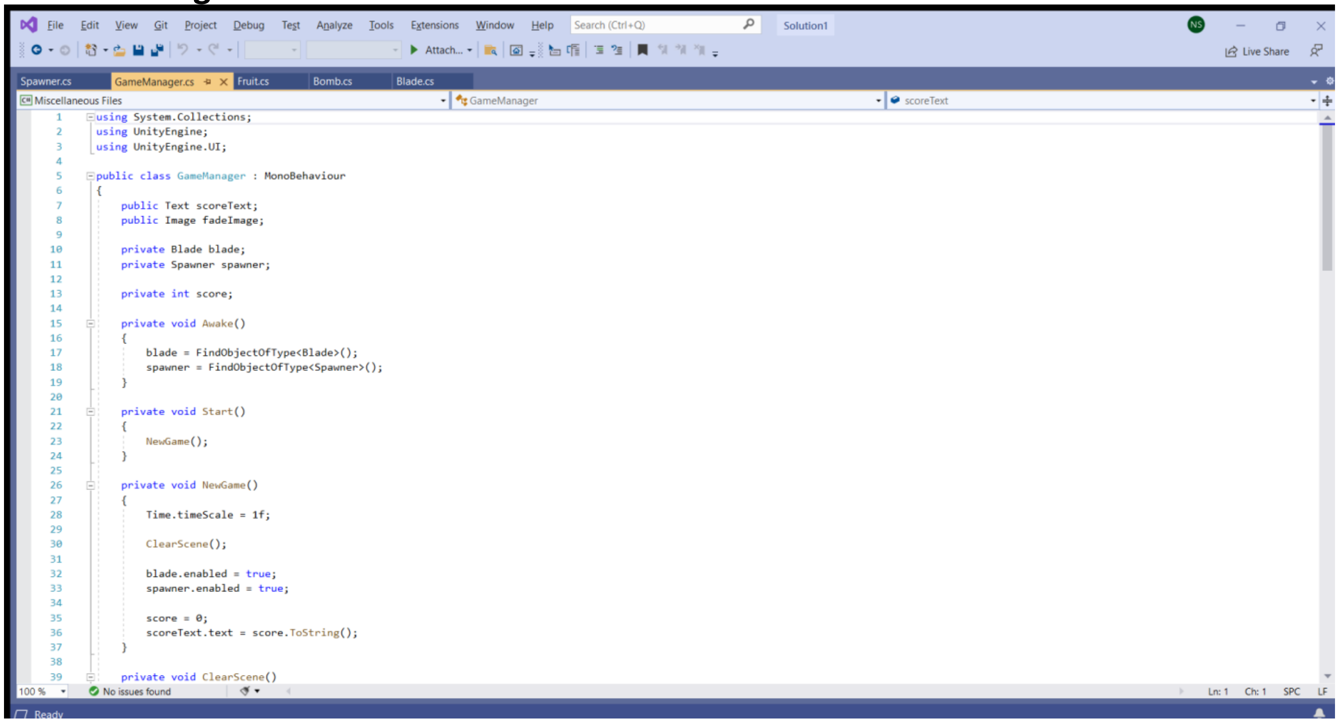
```
using UnityEngine;

public class Bomb : MonoBehaviour
{
    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            GetComponent<Collider>().enabled = false;
            FindObjectOfType<GameManager>().Explode();
        }
    }

}
```

## c) Fruits



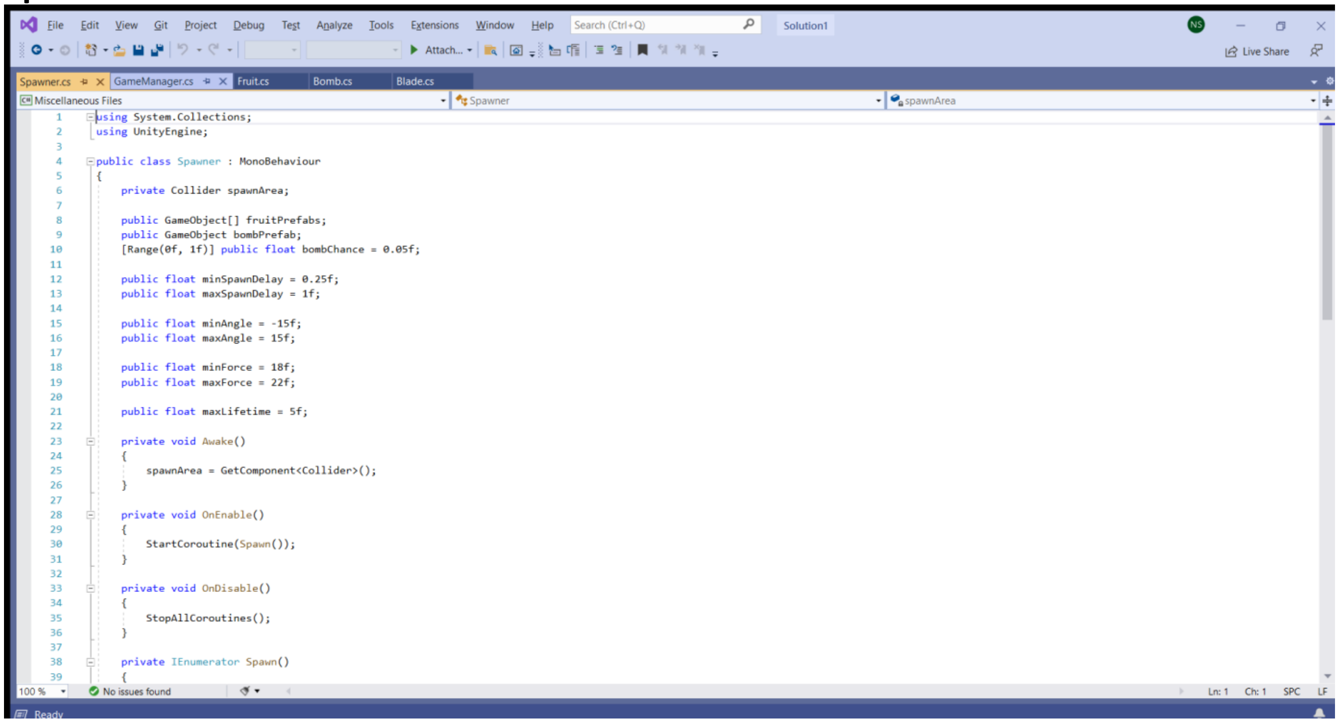
```
using UnityEngine;

public class Fruit : MonoBehaviour
{
    public GameObject whole;
    public GameObject sliced;

    private Rigidbody fruitRigidbody;
    private Collider fruitCollider;
    private ParticleSystem juiceEffect;

    public int points = 1;

    private void Awake()
    {
        fruitRigidbody = GetComponent<Rigidbody>();
        fruitCollider = GetComponent<Collider>();
        juiceEffect = GetComponentInChildren<ParticleSystem>();
    }

    private void Slice(Vector3 direction, Vector3 position, float force)
    {
        FindObjectOfType<GameManager>().IncreaseScore(points);

        // Disable the whole fruit
        fruitCollider.enabled = false;
        whole.SetActive(false);

        // Enable the sliced fruit
        sliced.SetActive(true);
        juiceEffect.Play();

        // Rotate based on the slice angle
        float angle = Mathf.Atan2(direction.y, direction.x) * Mathf.Rad2Deg;
        sliced.transform.rotation = Quaternion.Euler(0f, 0f, angle);

        Rigidbody[] slices = sliced.GetComponentsInChildren<Rigidbody>();

        // Add a force to each slice based on the blade direction
```

## d) Game Manager



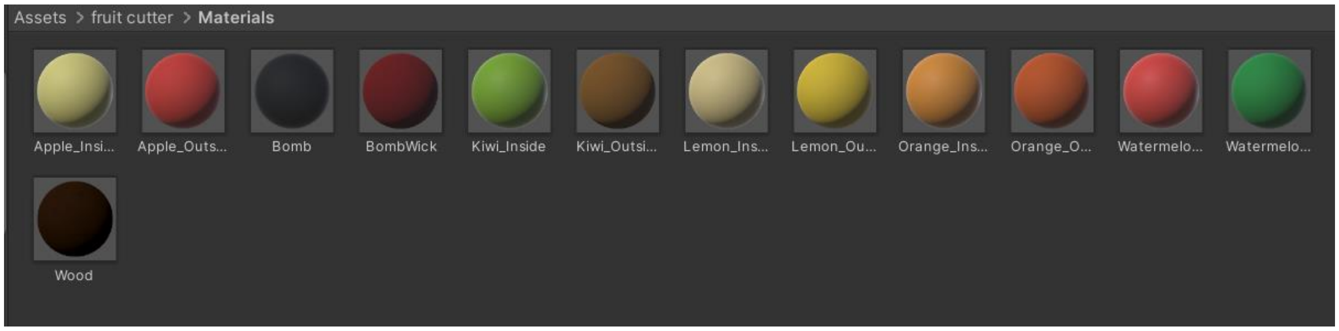
```
using System.Collections;
using UnityEngine;
using UnityEngine.UI;

public class GameManager : MonoBehaviour
{
    public Text scoreText;
    public Image fadeImage;

    private Blade blade;
    private Spawner spawner;

    private int score;

    private void Awake()
    {
        blade = FindObjectOfType<Blade>();
        spawner = FindObjectOfType<Spawner>();
    }

    private void Start()
    {
        NewGame();
    }

    private void NewGame()
    {
        Time.timeScale = 1f;

        ClearScene();

        blade.enabled = true;
        spawner.enabled = true;

        score = 0;
        scoreText.text = score.ToString();
    }

    private void ClearScene()
```

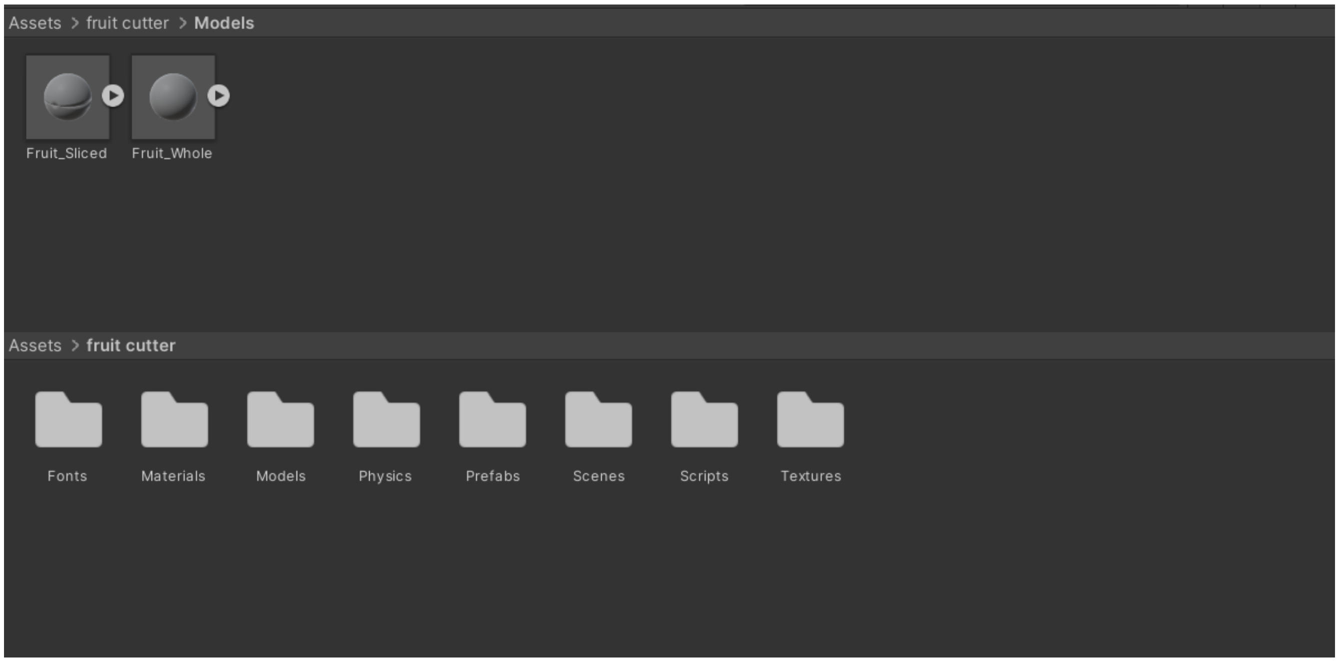
## e) Spawner



```csharp
using System.Collections;
using UnityEngine;

public class Spawner : MonoBehaviour
{
    private Collider spawnArea;

    public GameObject[] fruitPrefabs;
    public GameObject bombPrefab;
    [Range(0f, 1f)] public float bombChance = 0.05f;

    public float minSpawnDelay = 0.25f;
    public float maxSpawnDelay = 1f;

    public float minAngle = -15f;
    public float maxAngle = 15f;

    public float minForce = 18f;
    public float maxForce = 22f;

    public float maxLifetime = 5f;

    private void Awake()
    {
        spawnArea = GetComponent<Collider>();
    }

    private void OnEnable()
    {
        StartCoroutine(Spawn());
    }

    private void OnDisable()
    {
        StopAllCoroutines();
    }

    private IEnumerator Spawn()
    {
```

## Assets menu:




Assets > fruit cutter > Models
Fruit_Sliced
Fruit_Whole
Assets > fruit cutter
Fonts
Materials
Models
Physics
Prefabs
Scenes
Scripts
Textures