1. Animasyonlar:
2. Çizim-UI-GUI ve Texturelar-Materialler
3. Parti Assetler
4. Ses ve Müzik
5. Model ve Prefablar
6. Sahne
7. Kodlar
8. Test ve Çıktı

Kodlar:

1. Audio Controller.cs

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class AudioController : MonoBehaviour

{

public static AudioController audioControllerInstance;

public sound[] sounds;

private PlayerController thePlayer;

void Awake()

{

if (audioControllerInstance == null)

{

audioControllerInstance = this;

}

foreach (var s in sounds)

{

thePlayer = FindObjectOfType<PlayerController>();

s.source = gameObject.AddComponent<AudioSource>();

s.source.clip = s.clip;

s.source.volume = s.volume;

s.source.pitch = s.pitch;

s.source.loop = s.loop;

}

}

public void Play(string name)

{

sound s = Array.Find(sounds, sound => sound.name == name);

s.source.Play();

}

public void Stop(string name)

{

sound s = Array.Find(sounds, sound => sound.name == name);

s.source.Stop();

}

}

1. BuyArea.cs

using DG.Tweening;

using System.Collections;

using System.Collections.Generic;

using TMPro;

using UnityEngine;

using UnityEngine.UI;

public class BuyArea : MonoBehaviour

{

[SerializeField] private BuyAreaSO buyAreaType = null; // Scriptable Objects erisir

[Space]

[Header("Shop Area Take")]

[SerializeField] private GameObject farmerAndShopObject, buyObject;

//[SerializeField] private int cost; // Satın alınacak öğenin maliyeti

//[SerializeField] private int currentMoney; // Ödenen para miktarı

[SerializeField] private float progress;

[SerializeField] private Image progressImage;

[SerializeField] private TextMeshProUGUI areaText;

public bool areaLocked;

[Space]

[Header("Shop Money Dotween")]

[SerializeField] private float duration; // Shake süresi

[SerializeField] private float strength; // Shake gücü

[SerializeField] private int vibrato; // Titreşim sayısı

[SerializeField] private float randomness; // Randomluk

private void Start()

{

areaText.text = "$ " + (buyAreaType.cost - buyAreaType.currentMoney);

this.areaLocked = buyAreaType.locked;

if (!buyAreaType.locked)

{

// Bina açık mı değil mi \*- Açık ise kodu kapat ve bina açık kalsın

this.buyObject.SetActive(false); // Satın alma triggerini pasif yapar

this.farmerAndShopObject.SetActive(true); // Satın alınan öğeyi aktif hale getir

this.enabled = false; // Satın alma işlemi tamamlanınca kodu kapat

}

}

public void Buy(int valueMoney)

{

if (buyAreaType.currentMoney < buyAreaType.cost)

{

buyAreaType.currentMoney += valueMoney; // Satın alma için ödenen para miktarını arttır

areaText.text = "$ " + (buyAreaType.cost - buyAreaType.currentMoney); // Kalan para mitarnı göster

progress = (buyAreaType.currentMoney / buyAreaType.cost);

progressImage.fillAmount = progress; // Ödenen miktarın bar gösterimi

SaveManager.savemanagerInstance.SaveGame(); // Verileri kaydet

if (buyAreaType.currentMoney == buyAreaType.cost)

{

// Eğer satın alma işlemi tamamlandıysa

this.buyAreaType.locked = false; // Bina satın alınma kilidi açar

this.areaLocked = buyAreaType.locked; // Bina satın alınma kilidi incpector'da gösterir

this.buyObject.SetActive(false); // Satın alma alanını pasif yapar

this.farmerAndShopObject.SetActive(true); // Satın alınan öğeyi aktif hale getir

this.farmerAndShopObject.transform.DOShakeScale(duration, strength, vibrato, randomness); // Dotween ile Açılan binanın Scale değerini büyütüp küçültür

AudioController.audioControllerInstance.Play("BuyAreaSound"); // Yeni alan satın alındığında ses çalışır

SaveManager.savemanagerInstance.SaveGame(); // Verileri kaydet

this.enabled = false; // Satın alma işlemi tamamlanınca kodu kapat

}

}

}

}

1. BuyAreaSO.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[CreateAssetMenu(fileName = "New BuyArea Type", menuName = "BuyAreaSO")]

public class BuyAreaSO : ScriptableObject

{

[SerializeField] private int \_cost; // Satın alınacak öğenin maliyeti

[SerializeField] private int \_currentMoney; // Ödenen para miktarı

[SerializeField] private bool \_locked = true;

public int currentMoney

{

get { return \_currentMoney; }

set { \_currentMoney = value; }

}

public int cost

{

get { return \_cost; }

set { \_cost = value; }

}

public bool locked

{

get { return \_locked; }

set { \_locked = value; }

}

}

1. BuyManager.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class BuyManager : MonoBehaviour

{

[SerializeField] private int totalMoney=0;

public int publicTotalMoney

{

get { return totalMoney; }

set { totalMoney = value; }

}

private void OnEnable()

{

TriggerEventManager.OnMoneyCollected += IncreaseMoney;

TriggerEventManager.OnBuyShopAndFarmer += BuyArea;

}

private void OnDisable()

{

TriggerEventManager.OnMoneyCollected -= IncreaseMoney;

TriggerEventManager.OnBuyShopAndFarmer -= BuyArea;

}

void BuyArea()

{

if (TriggerEventManager.buyArea != null)

{

if (totalMoney >= 1 && TriggerEventManager.buyArea.areaLocked)

{

// Eğer karakterin parası varsa ve satın alınacak alan açılmamışsa

TriggerEventManager.buyArea.Buy(1);

totalMoney -= 1;

}

}

}

private void IncreaseMoney()

{

if (TriggerEventManager.shopManager.moneyList.Count > 0)

{

totalMoney += 5; // Para değerini arttır

TriggerEventManager.shopManager.RemoveLastMoney(); // Her para kazanıldığında money listesindeki paralar çıkarılır

AudioController.audioControllerInstance.Play("MoneySound"); // Her para toplandığında ses çalışır

}

}

}

1. CameraController.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CameraController : MonoBehaviour

{

[SerializeField] private Transform \_target;

[SerializeField] private Vector3 \_offset;

[SerializeField] private float \_chaseSpeed = 5;

private void LateUpdate()

{

Vector3 desPos = \_target.position + \_offset; // Kamera ile takip edilen obje arasındaki mesafe

transform.position = Vector3.Lerp(transform.position, desPos, \_chaseSpeed); // Kamera pozisynu yumuþak geçiþ ile aradaki mesafe kader uzaktan takip eder

}

}

1. CollectManager.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CollectManager : MonoBehaviour

{

public List<GameObject> fruitList = new List<GameObject>(); // Toplanan meyvelerin tutulduğu liste

[SerializeField] private Transform collectPoint; // Meyvelerin toplanacağı pozisyon

[SerializeField] private float fruitBetween = 10f; //Meyveler arası mesafe

[SerializeField] private int fruitCollectLimit = 10; // Karakterin toplayabileceği meyve sayısı

[Space]

[Header("Object Pool")]

[SerializeField] private ObjectPool objectPool = null;

[SerializeField] private int poolValue = 0;

private void OnEnable()

{

TriggerEventManager.OnFruitCollet += GetFruit; //Eğer Meyve toplama alanın da ise GetFruit fonkisyonu çalışır

TriggerEventManager.OnFruitGive += GiveShopFruit; //Eğer Meyve bırakma alanın da ise GiveShopFruit fonkisyonu çalışır

}

private void OnDisable()

{

TriggerEventManager.OnFruitCollet -= GetFruit; // Eğer Meyve toplama alanından çıkmışsa GetFruit fonkisyonu durdurulur

TriggerEventManager.OnFruitGive -= GiveShopFruit; // Eğer Meyve bırakma alanından çıkmışsa GiveShopFruit fonkisyonu durdurulur

}

private void GetFruit()

{

// Meyve toplama fonksiyonu (Farmerden)

// #objectPool ile ekleme

if (fruitList.Count < fruitCollectLimit)

{

poolValue = TriggerEventManager.farmerManager.publicpoolValue+4; // poolValue değeri FarmerField alanında üretilen poolObjectin dizi numarısını alır ve o objeyi seçer

// Eğer karakterin topladğı meyve sayısı fruitCollectLimit sayısından az ise toplar

GameObject newCollectFruit = objectPool.GetPooledObject(poolValue); // "ObjectPool" scriptinden yeni nesne çeker ve aktif hale getirirr

newCollectFruit.transform.parent = collectPoint; // Aktif olan objeyi collectPoint çocuğu yapar

newCollectFruit.transform.position = new Vector3(collectPoint.position.x,

((float)fruitList.Count / fruitBetween) + collectPoint.position.y,

collectPoint.position.z); // Yeni Fruit objesini pozisyonu belirlenir

fruitList.Add(newCollectFruit); // Yeni oluşturulan meyveyi fruitList listesine ekle

AudioController.audioControllerInstance.Play("FruitSound"); // Her meyve toplandığında ses çalışır

if (TriggerEventManager.farmerManager != null)

{

// Temas ettği objenin içinde farmerManager varsa

// farmerManager objesinin RemoveLastFruit fonksiyonu ile son meyveleri siler

TriggerEventManager.farmerManager.RemoveLastFruit();

}

}

}

public void RemoveLastFruit()

{

// Karaker FarmerField alanında meyve topladığında son oluşturulan meyveyi toplar ve silinir

if (fruitList.Count > 0)

{

// Eğer toplanacak meyve var ise son meyveyi sil

// #objectPool ile silme

objectPool.SetPooledObject(fruitList[fruitList.Count - 1], poolValue); // objectPool ile aktif hale gelen objeyi pasif hale getirir

fruitList[fruitList.Count - 1].transform.parent = GameObject.Find("FruitObjects").gameObject.transform;

// Pasif olan objeyi tekrar FruitObjects çocuğu yapar - objecool döngü halinde çalışması için

fruitList.RemoveAt(fruitList.Count - 1); // fruitList listesinden siler

}

}

public void GiveShopFruit()

{

// Toplanan meyveleri Mağazaya bırakma fonksiyonu

if (fruitList.Count > 0)

{

// Eğer bırakılack meyve varsa shopManager objesinden GetFruit fonksiyonu çalışsın ve meyve bıraksın

if (TriggerEventManager.shopManager.fruitList.Count < TriggerEventManager.shopManager.maxFruit)

{

TriggerEventManager.shopManager.GetFruit();

RemoveLastFruit(); // Toplanan meyveyi listeden siler

}

}

}

}

1. FarmerManager.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using DG.Tweening;

public class FarmerManager : MonoBehaviour

{

public List<GameObject> fruitList = new List<GameObject>(); // Oluşturulan meyvelerin tutulduğu liste

[Space]

[Header("Fruit Spawner")]

[SerializeField] private float spawnerTime = 0.5f; //Meyve oluşturma süresi

[SerializeField] private float fruitBetween = 10f; //Meyveler arası mesafe

[SerializeField] private int maxFruit = 50; // Toplam çıkarilacak meyve sayısı

[SerializeField] private int stackCount = 10; // Bir sırada oluşacak meyve sayısı

[SerializeField] private Transform spawnPoint; // Meyvelerin Çıkarılacağı pozisyon

[SerializeField] private GameObject fullFruit;

public bool isWorking;

[Space]

[Header("Object Pool")]

[SerializeField] private ObjectPool objectPool = null;

[SerializeField] private int poolValue = 0;

public int publicpoolValue

{

get { return poolValue; }

set { poolValue = value; }

}

[Space]

[Header("Fruit Jump Dotween")]

[SerializeField] private GameObject jumpFruitObject;

[SerializeField] private Vector3 jumpFruitObjectStartPositoin;

[SerializeField] private float duration; // Düşme süresi

void Start()

{

StartCoroutine(nameof(FarmerFruitSpawner));

}

void Update()

{

}

IEnumerator FarmerFruitSpawner()

{

while (true)

{

float fruitCount = fruitList.Count;

int colCount = (int)fruitCount / stackCount; // Bir sırada oluşacak meyve sayısı

// #objectPool ile ekleme

if (isWorking) // Eğer farmer çalşıyorsa

{

GameObject newFruit = objectPool.GetPooledObject(poolValue); // "ObjectPool" scriptinden yeni nesne çeker ve aktif hale getirir

newFruit.transform.position = new Vector3(spawnPoint.position.x + ((fruitCount % stackCount) / fruitBetween),

spawnPoint.position.y + 0.1f,

spawnPoint.position.z + ((float)colCount / 3));

fruitList.Add(newFruit); // Yeni oluşturulan meyveyi fruitList listesine ekle

jumpFruitObject.transform.DOMove(new Vector3(newFruit.transform.position.x, 0f, newFruit.transform.position.z), duration);

jumpFruitObject.transform.localPosition = jumpFruitObjectStartPositoin;

if (fruitList.Count >= maxFruit)

{

isWorking = false; // Eğer toplam çıkarılan Fruit Sayısı maxFruit sayısına büyük eşit ise çalışma pasif olur

fullFruit.SetActive(true);

}

}

else if (fruitList.Count < maxFruit)

{

isWorking = true; // Eğer toplam çıkarılan Fruit Sayısı maxFruit sayısından az ise çalışma aktif olur

fullFruit.SetActive(false);

}

yield return new WaitForSeconds(spawnerTime); // Her spawnerTime süresinde bir Meyve oluştur

}

}

public void RemoveLastFruit()

{

// Karaket FarmerField alanında meyve topladığında son oluşturulan meyveyi toplar ve silinir

if (fruitList.Count > 0)

{

// Eğer toplanacak meyve var ise son meyveyi sil

// #objectPool ile silme

objectPool.SetPooledObject(fruitList[fruitList.Count - 1], poolValue); // objectPool ile aktif hale gelen objeyi pasif hale getirir

fruitList.RemoveAt(fruitList.Count - 1); // fruitList listesinden siler

}

}

}

1. Fps.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using TMPro;

public class Fps : MonoBehaviour

{

public float fps;

public TextMeshProUGUI fpsText;

public float fpsSecond=0.1f;

private void Awake()

{

QualitySettings.vSyncCount = 0;

Application.targetFrameRate = 60;

}

void Start()

{

StartCoroutine(nameof(GetFPS));

}

IEnumerator GetFPS()

{

while (true)

{

fps = (int)(1f / Time.unscaledDeltaTime);

fpsText.text = fps + " fps";

yield return new WaitForSeconds(fpsSecond);

}

}

}

1. GameManager.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class GameManager : MonoBehaviour

{

public void GameExit()

{

SaveManager.savemanagerInstance.SaveGame(); // Verileri kaydet

Application.Quit(); // Oyundan çık

}

}

1. ObjectPool.cs

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ObjectPool : MonoBehaviour

{

[Serializable]

public struct Pool

{ // Pool Birden fazla farklı nesne ekler

// Quene= Sıranın sonuna ekleyip, Başından çıkarır

public Queue<GameObject> pooledObjects; // Oluşturulacak nesneler bir sırada tutulur - Quene,Liste veya dizide tutulablir

public GameObject objectPrefab; // Oluşturulacak nesne prefab�

public int poolSize; // Oluşturulacak nesne sayısı

}

[SerializeField] private Pool[] pools = null;

private void Awake()

{ // --- Obje Havuz oluşturma ---

for (int j = 0; j < pools.Length; j++)

{

pools[j].pooledObjects = new Queue<GameObject>(); // Yeni bir sıra oluşturulur

for (int i = 0; i < pools[j].poolSize; i++)

{

// for dögüsü ile"poolSize" Oluşturulacak nesne sayısı kadar yeni nesne oluşturur

GameObject newObj = Instantiate(pools[j].objectPrefab); // Yeni oluşturulan nesneleri "newObj" ismi ile oluşturur( Hangi Obje (pool) se�ceçilmiş ise )

newObj.SetActive(false); // Başlangıçta tüm nesnenin aktifliği false yapar

newObj.transform.parent = GameObject.Find("FruitObjects").gameObject.transform; // Oluşturulan objeleri FruitObjects objesinin alt objesi yapar

newObj.name = pools[j].objectPrefab.gameObject.name;

pools[j].pooledObjects.Enqueue(newObj); // Oluşturulan yeni nesneler sıraya eklenir "poolSize" değeri kadar nesne eklenir

}

}

}

public GameObject GetPooledObject(int objectType)

{

// Aktif edilcek pool objeleri

if (objectType >= pools.Length) return null;

GameObject newObj = pools[objectType].pooledObjects.Dequeue(); // Sıranin başından ilk nesneyi çıkarır

newObj.SetActive(true); // Çıkarılan nesnenin aktifliğini true yapar

return newObj;

}

public void SetPooledObject(GameObject poolObject, int objectType)

{

// Pasif edilcek pool objeleri

if (objectType >= pools.Length) return;

pools[objectType].pooledObjects.Enqueue(poolObject); // Çıkarılan nesneyi tekrar sıranınn sonune ekler ve havuzda dongu oluşturur

poolObject.SetActive(false);

}

}

1. PlayerController.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PlayerController : MonoBehaviour

{

Rigidbody rb;

private Animator anim;

[Space]

[Header("Player Controller")]

[SerializeField] private float \_movementSpeed=300;

[SerializeField] private float \_rotationSpeed = 500;

private bool isRunnig;

[Space]

[Header("Joystick Controller")]

[SerializeField] Joystick joystick; // Joystick scripti

float vertical, horizontal; // Player yönü

void Start()

{

rb = GetComponent<Rigidbody>();

anim = GetComponent<Animator>();

}

void Update()

{

if (isRunnig)

{

anim.SetBool("isRunning", true);

}

else

{

anim.SetBool("isRunning", false);

}

}

private void FixedUpdate()

{

JoystickMove(); // Joystick kontrolü çalýþýr

}

public void JoystickMove()

{

// Joystick kontrolü

vertical = joystick.Vertical \* \_movementSpeed \* Time.fixedDeltaTime;

horizontal = joystick.Horizontal \* \_movementSpeed \* Time.fixedDeltaTime;

rb.velocity = new Vector3(horizontal, 0, vertical);

if (rb.velocity != Vector3.zero)

{

isRunnig = true;

Quaternion temp = Quaternion.LookRotation(rb.velocity, Vector3.up);

rb.rotation = Quaternion.RotateTowards(transform.rotation, temp, \_rotationSpeed \* Time.fixedDeltaTime);

}

else

{

isRunnig = false;

}

}

}

1. RotateMoney.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class RotateMoney : MonoBehaviour

{

[SerializeField] float turnSpeed = 100f; // Obejelerin dönme hızı

void Update()

{

transform.Rotate(0f, 0f, turnSpeed \* Time.fixedDeltaTime);

}

}

1. SaveManager.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System.Runtime.Serialization.Formatters.Binary;

using System.IO;

public class SaveManager : MonoBehaviour

{

public static SaveManager savemanagerInstance;

[Header("Meta")]

public string saveName;

[Header("Scriptable Objects")]

public List<ScriptableObject> objectsToSave;

private void Awake()

{

if (savemanagerInstance == null)

{

savemanagerInstance = this;

}

}

private void OnEnable()

{

// Veri Cekme

for (int i = 0; i < objectsToSave.Count; i++)

{

if (File.Exists(Application.persistentDataPath + string.Format("/{0}\_{1}.pso", saveName, i)))

{

Debug.Log("Veriler Alindi...");

BinaryFormatter bf = new BinaryFormatter();

FileStream file = File.Open(Application.persistentDataPath + string.Format("/{0}\_{1}.pso", saveName, i), FileMode.Open);

JsonUtility.FromJsonOverwrite((string)bf.Deserialize(file), objectsToSave[i]);

file.Close();

}

else

{

//Do Nothing

Debug.Log("Daha Once Veri Kaydedilmedi...");

}

}

}

public void SaveGame()

{

// Veri Kaydetme

Debug.Log("Veriler Kaydedildi...");

for (int i = 0; i < objectsToSave.Count; i++)

{

BinaryFormatter bf = new BinaryFormatter();

FileStream file = File.Create(Application.persistentDataPath + string.Format("/{0}\_{1}.pso", saveName, i));

var json = JsonUtility.ToJson(objectsToSave[i]);

bf.Serialize(file, json);

file.Close();

}

}

}

1. ShopManager.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using DG.Tweening;

public class ShopManager : MonoBehaviour

{

public List<GameObject> fruitList = new List<GameObject>(); // Oluşturulan meyvelerin tutulduğu liste

public List<GameObject> moneyList = new List<GameObject>(); // Oluşturulan paraların tutulduğu liste

[Space]

[Header("Shop Fruit Genarete")]

[SerializeField] private float fruitBetween=6;

[SerializeField] private int stackCount = 10; // X ekseninde oluşacak meyve sayısı

public int maxFruit = 150; // Max verilecek meyve sayısı

[SerializeField] private Transform givePoint; // Meyvelerin Çıkarılacağı pozisyon

[SerializeField] private GameObject fullFruit;

bool isWorking;

[Space]

[Header("Shop Money Genarete")]

[SerializeField] private GameObject moneyPrefab; // Oluşturulacak obje

[SerializeField] private GameObject rotateMoney; // Oluşturulacak obje

[SerializeField] private float moneySpawnerTime = 0.8f; //Meyve oluşturma süresi

[SerializeField] private float moneyBetween = 4f; //Meyveler arası mesafe

[SerializeField] private int moneyStackCountX = 5; // X ekseninde oluşacak para sayısı

[SerializeField] private int moneyStackCountY = 25; // Y ekseninde oluşacak para sayısı

[SerializeField] private int maxMoney = 100; // Max verilecek para sayısı

[SerializeField] private Transform moneySpawnPoint; // Paraların Çıkarılacağı pozisyon

bool isMoneySpawner;

[Space]

[Header("Object Pool")]

[SerializeField] private ObjectPool objectPool = null;

[SerializeField] private int poolValue = 0;

[Space]

[Header("Shop Money Dotween")]

[SerializeField] private float duration; // Shake süresi

[SerializeField] private float strength; // Shake gücü

[SerializeField] private int vibrato; // Titreşim sayısı

[SerializeField] private float randomness; // Randomluk

private void Start()

{

StartCoroutine(nameof(GenareteMoney));

}

IEnumerator GenareteMoney()

{

while (true)

{

if (moneyList.Count < maxMoney)

{

isMoneySpawner = true;

}

else

{

isMoneySpawner = false;

}

if (fruitList.Count > 0 && isMoneySpawner)

{

//Eğer Shop'ta meyve varsa para üret

float moneyCount = moneyList.Count;

int colCount = (int)moneyCount / moneyStackCountX; // Bir sırada oluşacak para sayısı

int rowCount = (int)moneyCount / moneyStackCountY; // Bir sırada oluşacak para sayısı

// #objectPool ile ekleme

GameObject newMoney = objectPool.GetPooledObject(0); // "ObjectPool" scriptinden yeni nesne çeker ve aktif hale getirir

newMoney.transform.position = new Vector3(moneySpawnPoint.position.x + ((moneyCount % moneyStackCountX) / moneyBetween),

moneySpawnPoint.position.y + ((float)rowCount / 10) + 0.05f,

moneySpawnPoint.position.z + ((float)colCount / 2) - ((2 \* rowCount) + ((float)rowCount / 2f)));

newMoney.transform.DOShakeScale(duration, strength, vibrato, randomness); // Dotween ile Paranın Scale değerini büyütüp küçültür

moneyList.Add(newMoney); // Yeni oluşturulan parayı moneyList listesine ekle

RemoveLastFruit();

}

yield return new WaitForSeconds(moneySpawnerTime);

}

}

private void Update()

{

if (moneyList.Count > 0)

{

// Eğer Shop para üretmiş ve moneyList en az 1 atne para varsa ise Shop üzerindeki para ikonu aktif hale gelir

rotateMoney.SetActive(true);

}

else

{

rotateMoney.SetActive(false);

}

if (fruitList.Count >= maxFruit)

{

fullFruit.SetActive(true);

}

else

{

fullFruit.SetActive(false);

}

}

public void GetFruit()

{

// Karakteren gelecek meyveler ile Meyve oluştur

float fruitCount = fruitList.Count;

int colCount = (int)fruitCount / stackCount; // Bir sırada oluşacak meyve sayısı

if (isWorking)

{

// #objectPool ile ekleme

FruitSelective(); // Karakterin topladığı meyveler listesinde en üstte bulunan meyveyinin ismine göre poolValue değeri belirlenir

GameObject newGiveFruit = objectPool.GetPooledObject(poolValue); // "ObjectPool" scriptinden yeni nesne çeker ve aktif hale getirir

newGiveFruit.transform.position = new Vector3(givePoint.position.x + ((fruitCount % stackCount) / fruitBetween),

givePoint.position.y + 0.1f,

givePoint.position.z + ((float)colCount / 6));

fruitList.Add(newGiveFruit); // Yeni oluşturulan meyveyi fruitList listesine ekle

AudioController.audioControllerInstance.Play("FruitSound"); // Her meyve toplandığında ses çalışır

if (fruitList.Count >= maxFruit)

{

isWorking = false; // Eğer toplam çıkarılan Fruit Sayısı maxFruit sayısına büyük eşit ise çalışma pasif olur

}

}

else if (fruitList.Count < maxFruit)

{

isWorking = true; // Eğer toplam çıkarılan Fruit Sayısı maxFruit sayısından az ise çalışma aktif olur

}

}

public void RemoveLastFruit()

{

// Karaket FarmerField alanında meyve topladığında son oluşturulan meyveyi toplar ve silinir

if (fruitList.Count > 0)

{

// Eğer toplanacak meyve var ise son meyveyi sil

// #objectPool ile silme

objectPool.SetPooledObject(fruitList[fruitList.Count - 1], poolValue); // objectPool ile aktif hale gelen objeyi pasif hale getirir

// Pasif olan objeyi tekrar FruitObjects çocuğu yapar döndü halinde çalışması için

fruitList.RemoveAt(fruitList.Count - 1); // fruitList listesinden siler

}

}

public void RemoveLastMoney()

{

// Karaket FarmerField alanında meyve topladığında son oluşturulan meyveyi toplar ve silinir

if (moneyList.Count > 0)

{

// Eğer toplanacak meyve var ise son meyveyi sil

// #objectPool ile silme

objectPool.SetPooledObject(moneyList[moneyList.Count - 1], 0); // objectPool ile aktif hale gelen objeyi pasif hale getirir

// Pasif olan objeyi tekrar FruitObjects çocuğu yapar döndü halinde çalışması için

moneyList.RemoveAt(moneyList.Count - 1); // fruitList listesinden siler

}

}

void FruitSelective()

{

if (TriggerEventManager.collectManager.fruitList[TriggerEventManager.collectManager.fruitList.Count - 1].gameObject.name == "Apple")

{

poolValue = 9;

}

if (TriggerEventManager.collectManager.fruitList[TriggerEventManager.collectManager.fruitList.Count - 1].gameObject.name == "Banana")

{

poolValue = 10;

}

if (TriggerEventManager.collectManager.fruitList[TriggerEventManager.collectManager.fruitList.Count - 1].gameObject.name == "Carrot")

{

poolValue = 11;

}

if (TriggerEventManager.collectManager.fruitList[TriggerEventManager.collectManager.fruitList.Count - 1].gameObject.name == "Mushroom")

{

poolValue = 12;

}

}

}

1. Sound.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

[System.Serializable]

public class sound

{

[HideInInspector]

public AudioSource source;

public string name;

public AudioClip clip;

[Range(0f, 1f)]

public float volume;

[Range(.1f, 3f)]

public float pitch;

public bool loop;

}

1. TreeController.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using DG.Tweening;

public class TreeController : MonoBehaviour

{

[Space]

[Header("Farmer Fruit Dotween")]

private GameObject farmer;

[SerializeField] private float duration; // Shake süresi

[SerializeField] private float strength; // Shake gücü

[SerializeField] private int vibrato; // Titreşim sayısı

[SerializeField] private float randomness; // Randomluk

[SerializeField] private float shakeTime = 0.5f;

private Animator anim;

private bool isWork;

void Start()

{

this.farmer = this.transform.parent.gameObject.transform.GetComponent<FarmerManager>().gameObject;

this.anim = this.farmer.gameObject.transform.GetChild(0).gameObject.GetComponent<Animator>();

StartCoroutine(nameof(ShakeTree));

}

void Update()

{

if (farmer.gameObject.GetComponent<FarmerManager>().isWorking)

{

// Eğer Farmer alanında meyve üretimi var ise Farmer karakterinin isWorking animsyonu çalışır

this.anim.SetBool("isWorking", true);

}

else

{

this.anim.SetBool("isWorking", false);

}

}

IEnumerator ShakeTree()

{

while (true)

{

if (farmer.gameObject.GetComponent<FarmerManager>().isWorking)

{

this.transform.DOShakeRotation(duration, strength, vibrato, randomness);

}

yield return new WaitForSeconds(shakeTime);

}

}

}

1. TriggerEvent.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class TriggerEventManager : MonoBehaviour

{

public delegate void OnFruitCollectArea(); // Meyve toplama alanı

public static event OnFruitCollectArea OnFruitCollet; // Meyve toplama Eventi

public delegate void OnFruitGiveArea(); // Meyve bırakma alanı

public static event OnFruitGiveArea OnFruitGive; // Meyve bırakma Eventi

public delegate void OnMoneyArea(); // Para toplama alanı

public static event OnMoneyArea OnMoneyCollected; // Para toplama Eventi

public delegate void OnBuyArea(); // Satın alma alanı

public static event OnBuyArea OnBuyShopAndFarmer; // Satın alma alanı Eventi

public static FarmerManager farmerManager; // FarmerManager erişim sağlar

public static ShopManager shopManager; // ShopManager erişim sağlar

public static CollectManager collectManager; // CollectManager erişim sağlar

public static BuyArea buyArea;

[SerializeField] private float fruitCollectTime = 0.5f; //Meyve toplama süresi

bool isCollecting,isGiving,isTakeMoney; // Topluyor mu - Bırakıyor mu

void Start()

{

StartCoroutine(nameof(CollectEnum)); //CollectEnum fonksiyonu başlangıçta çalıştır

}

IEnumerator CollectEnum()

{

// Toplama ve Verme işlemleri

while (true)

{

if (isCollecting)

{

// OnFruitCollet eventi ile birlikte istenilen yerden fonksiyon çalıştırıllır

OnFruitCollet(); // isCollecting True olduğu zamanda OnFruitCollet eventi çalışır

}

if (isGiving)

{

// OnFruitGive eventi ile birlikte istenilen yerden fonksiyon çalıştırıllır

OnFruitGive(); // isGiving True olduğu zamanda OnFruitGive eventi çalışır

}

if (isTakeMoney)

{

// OnMoneyCollected eventi ile birlikte istenilen yerden fonksiyon çalıştırıllır

OnMoneyCollected(); // isTakeMoney True olduğu zamanda OnMoneyCollected eventi çalışır

}

yield return new WaitForSeconds(fruitCollectTime);

}

}

private void OnTriggerStay(Collider other)

{

if (other.gameObject.CompareTag("FruitCollectArea"))

{

// Eğer Meyve toplama alanın da ise isCollecting True olur

isCollecting = true;

// Eğer Meyve toplama alanın da ise FarmerManager objesine erişir

farmerManager = other.gameObject.GetComponent<FarmerManager>();

}

if (other.gameObject.CompareTag("FruitGiveArea"))

{

// Eğer Meyve bırakma alanın da ise isGiving True olur

isGiving = true;

// Eğer Meyve bırakma alanın da ise ShopManager objesine erişir

shopManager = other.gameObject.GetComponent<ShopManager>();

// Eğer Meyve bırakma alanın da ise CollectManager objesine erişir

collectManager = gameObject.GetComponent<CollectManager>();

}

if (other.gameObject.CompareTag("MoneyArea"))

{

// Eğer Para toplama alanın da ise isTakeMoney True olur

isTakeMoney = true;

// Eğer Para bırakma alanın da ise ShopManager objesine erişir

shopManager = other.transform.parent.gameObject.GetComponent<ShopManager>();

}

if (other.gameObject.CompareTag("BuyArea"))

{

OnBuyShopAndFarmer();

// Eğer BuyArea alanın da ise BuyArea objesine erişir

buyArea = other.gameObject.GetComponent<BuyArea>();

}

}

private void OnTriggerExit(Collider other)

{

if (other.gameObject.CompareTag("FruitCollectArea"))

{

isCollecting = false;// Eğer Meyve toplama alanından çıkmışsa isCollecting pasif olur

farmerManager = null; // Eğer Meyve toplama alanından çıkmışsa erişimi keser

}

if (other.gameObject.CompareTag("FruitGiveArea"))

{

isGiving = false;// Eğer Meyve bırakma alanından çıkmışsa isGiving pasif olur

shopManager = null; // Eğer Meyve bırakma alanından çıkmışsa erişimi keser

collectManager = null; // Eğer Meyve bırakma alanından çıkmışsa erişimi keser

}

if (other.gameObject.CompareTag("MoneyArea"))

{

isTakeMoney = false; // Eğer Para toplama alanın da ise isTakeMoney pasif olur

shopManager = null; // Eğer Para bırakma alanından çıkmışsa erişimi keser

}

if (other.gameObject.CompareTag("BuyArea"))

{

buyArea = null; // Eğer BuyArea alanından çıkmışsa erişimi keser

}

}

}

1. UIManager.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using TMPro;

public class UIManager : MonoBehaviour

{

[Space]

[Header("Money Controller")]

[SerializeField] BuyManager buyManager;

[SerializeField] private TextMeshProUGUI totalMoneyTxt;

private void OnEnable()

{

TriggerEventManager.OnMoneyCollected += IncreaseMoneyUI;

TriggerEventManager.OnBuyShopAndFarmer += DecreaseMoneyUI;

}

private void OnDisable()

{

TriggerEventManager.OnMoneyCollected -= IncreaseMoneyUI;

TriggerEventManager.OnBuyShopAndFarmer -= DecreaseMoneyUI;

}

void Start()

{

totalMoneyTxt.text = "$ " + buyManager.publicTotalMoney.ToString();

}

void Update()

{

}

private void IncreaseMoneyUI()

{

// UI Money para arttır

totalMoneyTxt.text = "$ " + buyManager.publicTotalMoney.ToString();

}

private void DecreaseMoneyUI()

{

// UI Money para azalt

totalMoneyTxt.text = "$ " + buyManager.publicTotalMoney.ToString();

}

}