**Hareket Kodlar**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class hareket : MonoBehaviour

{

Rigidbody2D rgb;

Vector3 velocity;

public float speedAmount; // veya( float speedAmount= 5f; )

// Start is called before the first frame update

void Start()

{

rgb = GetComponent<Rigidbody2D>();

}

// Update is called once per frame

void Update()

{

velocity = new Vector3(Input.GetAxis("Horizontal"), 0f);

transform.position += velocity \* speedAmount \* Time.deltaTime;

}

}

**Hareket Kodlar (Bakma-zıplama)**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class hareket : MonoBehaviour

{

Rigidbody2D rgb;

Vector3 velocity;

float speedAmount= 5f;

public float jumpAmount; // veya( float jumpAmount= 3f; )

// Start is called before the first frame update

void Start()

{

rgb = GetComponent<Rigidbody2D>();

}

// Update is called once per frame

void Update()

{

velocity = new Vector3(Input.GetAxis("Horizontal"), 0f);

transform.position += velocity \* speedAmount \* Time.deltaTime;

\*\*Zıplama

if(Input.GetButtonDown("Jump")&& Mathf.Approximately(rgb.velocity.y, 0))

{

rgb.AddForce(Vector3.up \* jumpAmount, ForceMode2D.Impulse);

}

\*\*Bakma karakterin saga sola gidişi

if (Input.GetAxisRaw("Horizontal") == -1)

{

transform.rotation = Quaternion.Euler(0f, 180f, 0f);

}

else if (Input.GetAxisRaw("Horizontal") == 1)

{

transform.rotation = Quaternion.Euler(0f, 0f, 0f);

}

}

}

**Durma ve koşma animayonu**

* Sunny\_land player içindeki idle klasöründeki resimlerin hepsini seçip sag paneldeki player içine atıyoruz. Kaydetmemizi istiyor animasyonu proje içine animation adında bir klasör oluşturup oraya kaydedebiliriz tüm animasyonları daha düzenli duracaktır. Player\_ıdle adıyla kayıt ediyoruz

Windows –animation – animatör sekmesini açıyoruz player ıdle gelmiş oluyor

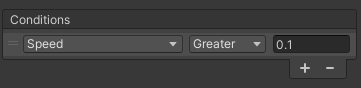
(yavaşlatmak istiyorsak animatör içindeki player ıdle kısmına tıklayıp speed değerini 1 den 0.5 e çekebilirsiniz)

* Sunny\_land player içindeki run klasöründeki resimlerin hepsini seçip sag paneldeki player içine atıyoruz. Player\_run adıyla kayıt ediyoruz

Karakterimiz ıdle pozisyonundan yürmüye başladığımızda run pozisyonuna girmesini istiyoruz onun için animatör sekmesi içinde bulunan Player\_ıdle üzerine sağ tuş yapıp “Make Transition” seçeneğini seçip Player\_run üzerine tıklıyoruz

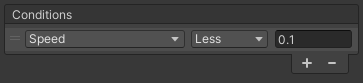
Aynı şekilde yürürken duracak olursak Player\_run dan yine Player\_ıdle kısmına girmesi için Player\_run üzerine tıklayıp “Make transition” seçeneğini seçerek Player\_ıdle üzerine tıklıyoruz.

* Sonraki adımda karakterin ne zaman ıdle dan runa geçeceğini belirtmek için bir parametre oluşturuyoruz. Animatör paneli içinde sol tarafta “Parameters” kısmına tıklayıp “+” işaretine basarak Float seçiyoruz. Adını “Speed” olarak değiştirip değerini 0.0 yapıyoruz.
* Daha sonra player\_ıdle kısmından player\_run kısmına giden oku tıklayerek sağ panelde İnspector kısmı açılıyor. Orada bulunan condition kısmından “+” basarak bir tane koşul ekleyecez.



Burda Greater seçili olacak değeri 0.1 yapıyoruz bunun anlamı biz hızı 0.0 olarak yazmıştık parametre kısmına eger hız 0.1 den büyük ise Player\_run animasyonuna geçiş yap diyoruz

* Player\_run kısmından geri Player\_ıdle kısmına girmek içinde aynı işlemi tekrarlıyoruz run dan idle giden oka tıklayarak inspector panelinceki condition kısmına tıklayarak “+” işaretine tıklıyoruz bu sefer Greater olan kısmı Less yapıyoruz 0.1 degerini giriyoruz eğer hız 0.1 den düşük ise ıdle animasyonuna girmesini istiyoruz.



Bu kısımdan sonra buraya yazdığımız speed kısmını artık kod olarakta playerımıza eklemiz gerekiyor

**Speed Animasyonu**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class hareket : MonoBehaviour

{

Rigidbody2D rgb;

Vector3 velocity;

public Animator animator;

public float speedAmount;

public float jumpAmount;

// Start is called before the first frame update

void Start()

{

rgb = GetComponent<Rigidbody2D>();

}

// Update is called once per frame

void Update()

{

velocity = new Vector3(Input.GetAxis("Horizontal"), 0f);

transform.position += velocity \* speedAmount \* Time.deltaTime;

animator.SetFloat("Speed", Mathf.Abs(Input.GetAxis("Horizontal")));

if(Input.GetButtonDown("Jump")&& Mathf.Approximately(rgb.velocity.y, 0))

{

rgb.AddForce(Vector3.up \* jumpAmount, ForceMode2D.Impulse);

}

if (Input.GetAxisRaw("Horizontal") == -1)

{

transform.rotation = Quaternion.Euler(0f, 180f, 0f);

}

else if (Input.GetAxisRaw("Horizontal") == 1)

{

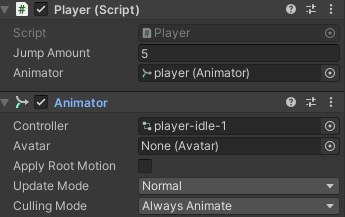
transform.rotation = Quaternion.Euler(0f, 0f, 0f);

}

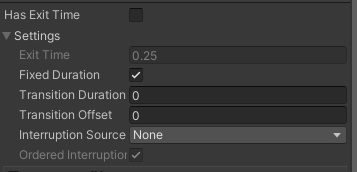
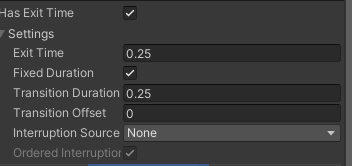
}

}

* Kodumuzu yazdıktan sonra program içinde playerımıza tıkladığımızda sag taraftaki inspector panelinde Player script içerisinde animatör diye bir kısım gelecektir içi boş bir şekilde altında bulunan Animatorü bu boş yere sürükleyip bırakıyoruz.



* Oyunu başlattığınızda animasyon geç devreye girdiğini göreceksiniz bunu ortadan kaldırmak için animatör sekmesine girip player\_ıdle kısmından player\_run kısmına giden ok işaretine bastığınızda sağ tarafta açılan inspector panelinde bulunan Has Exit Time yanındaki tik işaretini kaldırıp Transition Duration kısmına 0 a çekmeniz yeterli olacaktır aynı işlemi player run kısmından player ıdle giden ok işlemi içinde yapın. (aşağıdaki görseldeki gibi olacaktır.)



**Jump Animasyonu**

* Zıplama animasyonu için Sunny\_Lan player içinde jump klsörü içindeki iki görseli playerın içine atıyoruz ve Player\_Jump adı ile kayıt ediyoruz. animatör kısmına baktığımızda Player Jump butonu gelmiş olacaktır. Zıplama hareketi diğerlerinden bağımsız olduğu için içinde hazır olan “any State “ butonuna sağ tuş yapıp “Make Transition” seçerek Player\_Jumpa tıklıyoruz. Karakter hareketini bitirdikten sonra tekrar Idle pozisyonuna girmesi içinde player\_jump tan player\_ıdle kısmına yine make transition kısmını yapıyoruz.
* Zıplama içinde speed teki gibi bir değer belirtmemiz gerekmektedir. Animatör paneli içinde sol tarafta “Parameters” kısmına tıklayıp “+” işaretine basarak bool seçiyoruz. Adını “IsJumping” olarak değiştiriyoruz.
* Any state kısmından Player\_jump kısmına giden ok işaretine bastığımızda sağ tarafta açılan inspector panelinden Has exit time işaretli ise kaldırıp transition Duration kısmını 0 yapıyoruz. condition kısmından “+” basarak bir tane daha koşul eklememiz gerekiyor. IsJumping seçerek değerini true olarak bırakıyoruz. Aynı işlemi player\_jump kısmından player \_ıdle kısmına giden ok işeretini tıklayarak yapıyoruz. Burada sadece condition kımından koşulu eklerken Isjumping seçip değerini false yapıyoruz. Bu işlemlerden sonra aşağıdaki kodları ekliyoruz.

**Jump Animasyonu**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Player : MonoBehaviour

{

Rigidbody2D rgb;

Vector3 velocity;

float speedAmount = 5f;

public float jumpAmount;

public Animator animator;

// Start is called before the first frame update

void Start()

{

rgb = GetComponent<Rigidbody2D>();

}

// Update is called once per frame

void Update()

{

velocity = new Vector3(Input.GetAxis("Horizontal"), 0f);

transform.position += velocity \* speedAmount \* Time.deltaTime;

animator.SetFloat("Speed", Mathf.Abs(Input.GetAxis("Horizontal")));

if (Input.GetButtonDown("Jump") && Mathf.Approximately(rgb.velocity.y, 0))

{

rgb.AddForce(Vector3.up \* jumpAmount, ForceMode2D.Impulse);

animator.SetBool("IsJumping", true);

}

if (animator.GetBool("IsJumping") && Mathf.Approximately(rgb.velocity.y, 0))

{

animator.SetBool("IsJumping", false);

}

if (Input.GetAxisRaw("Horizontal") == -1)

{

transform.rotation = Quaternion.Euler(0f, 180f, 0f);

}

else if (Input.GetAxisRaw("Horizontal") == 1)

{

transform.rotation = Quaternion.Euler(0f, 0f, 0f);

}

}

}

**Can Sistemi**

* Can sistemi için sahnemize bir enemy ekliyoruz. (spikes) eklediğimiz bu nesnenin sağ tarafta bulunan inspecter içinde bulunan en üstte tag kısmı bulunmakta burada en altta bulunan at tag seçeneğini seçip yeni bir tag ekliyoruz adını enemy verelim daha sonra eklediğimiz nesnenin tagını enemy olarak seçelim. Oluşturduğumuz nesne Collider ekleyelim player ile etkileşime girebilmesi için

Sarı kodlar = Console kısmından bakıldığında aşağıdaki kodlar yazılırsa canın bir azaldığını göreceksiniz.

Mavi Kodlar= verilen can dolduktan sonra tekrar sahneye baştan başlamaya yarar.

Pembe Kodlar= player bir enemy değdikten sonra pozisyonunun biraz daha yukarıdan başlamasını sağlar

**Can Sistemi Kodlar**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Player : MonoBehaviour

{

Rigidbody2D rgb;

Vector3 velocity;

float speedAmount = 5f;

public float jumpAmount;

public Animator animator;

public int life;

// Start is called before the first frame update

void Start()

{

rgb = GetComponent<Rigidbody2D>();

}

// Update is called once per frame

void Update()

{

velocity = new Vector3(Input.GetAxis("Horizontal"), 0f);

transform.position += velocity \* speedAmount \* Time.deltaTime;

animator.SetFloat("Speed", Mathf.Abs(Input.GetAxis("Horizontal")));

if (Input.GetButtonDown("Jump") && Mathf.Approximately(rgb.velocity.y, 0))

{

rgb.AddForce(Vector3.up \* jumpAmount, ForceMode2D.Impulse);

}

if (Input.GetAxisRaw("Horizontal") == -1)

{

transform.rotation = Quaternion.Euler(0f, 180f, 0f);

}

else if (Input.GetAxisRaw("Horizontal") == 1)

{

transform.rotation = Quaternion.Euler(0f, 0f, 0f);

}

}

private void OnCollisionEnter2D(Collision2D collision)

{

if(collision.gameObject.tag=="enemy")

{

life--;

Debug.Log("Can : " + life);

gameObject.transform.position = new Vector3(gameObject.transform.position.x, gameObject.transform.position.y + 0.6f, gameObject.transform.position.z);

if(life<=0)

{

SceneManager.LoadScene("SampleScene");

}

}

}

}

**Can Sistemi DamageTime**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Player : MonoBehaviour

{

Rigidbody2D rgb;

Vector3 velocity;

float speedAmount = 5f;

public float jumpAmount;

public Animator animator;

public int life;

public float damageTime;

// Start is called before the first frame update

void Start()

{

rgb = GetComponent<Rigidbody2D>();

}

// Update is called once per frame

void Update()

{

velocity = new Vector3(Input.GetAxis("Horizontal"), 0f);

transform.position += velocity \* speedAmount \* Time.deltaTime;

animator.SetFloat("Speed", Mathf.Abs(Input.GetAxis("Horizontal")));

if (Input.GetButtonDown("Jump") && Mathf.Approximately(rgb.velocity.y, 0))

{

rgb.AddForce(Vector3.up \* jumpAmount, ForceMode2D.Impulse);

}

if (Input.GetAxisRaw("Horizontal") == -1)

{

transform.rotation = Quaternion.Euler(0f, 180f, 0f);

}

else if (Input.GetAxisRaw("Horizontal") == 1)

{

transform.rotation = Quaternion.Euler(0f, 0f, 0f);

}

if(damageTime>0)

{

damageTime -= Time.deltaTime;

}

}

private void OnCollisionEnter2D(Collision2D collision)

{

if(collision.gameObject.tag=="enemy" && damageTime<=0)

{

life--;

Debug.Log("Can : " + life);

damageTime = 0.7f;

gameObject.transform.position = new Vector3(gameObject.transform.position.x, gameObject.transform.position.y + 0.6f, gameObject.transform.position.z);

if(life<=0)

{

SceneManager.LoadScene("SampleScene");

}

}

}

private void OnCollisionStay2D(Collision2D collision)

{

if (collision.gameObject.tag == "enemy" && damageTime <= 0)

{

life--;

Debug.Log("Can : " + life);

damageTime = 0.7f;

gameObject.transform.position = new Vector3(gameObject.transform.position.x, gameObject.transform.position.y + 0.6f, gameObject.transform.position.z);

if (life <= 0)

{

SceneManager.LoadScene("SampleScene");

}

}

}

}

**Damage\_Color** **(hasar aldığında belirlenen süre kadar player rengi değişmesi)**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Player : MonoBehaviour

{

Rigidbody2D rgb;

Vector3 velocity;

float speedAmount = 5f;

public float jumpAmount;

public Animator animator;

public int life;

public float damageTime;

public Color firstColor;

public Color damageColor;

// Start is called before the first frame update

void Start()

{

rgb = GetComponent<Rigidbody2D>();

firstColor= gameObject.GetComponent<SpriteRenderer>().color;

}

// Update is called once per frame

void Update()

{

velocity = new Vector3(Input.GetAxis("Horizontal"), 0f);

transform.position += velocity \* speedAmount \* Time.deltaTime;

animator.SetFloat("Speed", Mathf.Abs(Input.GetAxis("Horizontal")));

if (Input.GetButtonDown("Jump") && Mathf.Approximately(rgb.velocity.y, 0))

{

rgb.AddForce(Vector3.up \* jumpAmount, ForceMode2D.Impulse);

animator.SetBool("IsJumping", true);

}

if (animator.GetBool("IsJumping") && Mathf.Approximately(rgb.velocity.y, 0))

{

animator.SetBool("IsJumping", false);

}

if (Input.GetAxisRaw("Horizontal") == -1)

{

transform.rotation = Quaternion.Euler(0f, 180f, 0f);

}

else if (Input.GetAxisRaw("Horizontal") == 1)

{

transform.rotation = Quaternion.Euler(0f, 0f, 0f);

}

if(damageTime>0)

{

damageTime -= Time.deltaTime;

if(gameObject.GetComponent<SpriteRenderer>().color == firstColor)

{

gameObject.GetComponent<SpriteRenderer>().color = damageColor;

}

else

{

if (gameObject.GetComponent<SpriteRenderer>().color == damageColor)

{

gameObject.GetComponent<SpriteRenderer>().color = firstColor;

}

}

}

}

private void OnCollisionEnter2D(Collision2D collision)

{

if(collision.gameObject.tag=="enemy" && damageTime<=0)

{

life--;

Debug.Log("Can : " + life);

damageTime = 0.9f;

gameObject.transform.position = new Vector3(gameObject.transform.position.x, gameObject.transform.position.y + 0.6f, gameObject.transform.position.z);

if(life<=0)

{

SceneManager.LoadScene("SampleScene");

}

}

}

private void OnCollisionStay2D(Collision2D collision)

{

if (collision.gameObject.tag == "enemy" && damageTime <= 0)

{

life--;

Debug.Log("Can : " + life);

damageTime = 0.9f;

gameObject.transform.position = new Vector3(gameObject.transform.position.x, gameObject.transform.position.y + 0.6f, gameObject.transform.position.z);

if (life <= 0)

{

SceneManager.LoadScene("SampleScene");

}

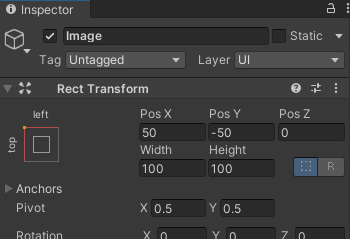
}

}

}

**Can Göstergesi**

* Can simgesi için <https://opengameart.org/content/heart-pixel-art> adresinden seçtiğimiz görseli kaydedip oyunumuzda asset klasörü içerisine atıyoruz.
* Sol panelden sağ tuş yapılır UI – Canvas seçeneği seçilir. Oluşuturulan canvasa sağ tuş yapılıp UI-İmage seçeneği seçilir.
* İmage tıklandığında sağ taraftaki inspector paneli içinde “source ımage” yazan yere indirdiğimiz kalp resmini sürükleyip bırakıyoruz.
* Koyduğumuz resim oyunumuzun ortasında görünmekte yine image’in inspector kısmından en üstte bulunan “rect transform” içerisinde solunda bulunan kare işaretine basın ve alt tuşuna basılı tuturak sol üst köşeyi seçiniz.



**Kamera Controller**

* Yeni bir script oluşturup adını cameraController veriyoruz. Kodları yazdıktan sonra yazdığımız scripti kullandığımız kameranın içine bırakıyoruz içinde public olarak player belirtmiştik o kısma da playerımızı sürükleyip bırakıyoruz.

**CameraController**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class cameraController : MonoBehaviour

{

public GameObject player;

// Start is called before the first frame update

void Start()

{

}

// Update is called once per frame

void Update()

{

Vector3 pos= new Vector3(player.transform.position.x, player.transform.position.y+1.5f, -10);

gameObject.transform.position = pos;

}

}

**Düşman kontrolü (yatay)**

* Collider ve rigidbody eklenir. Collider içerisinde ısTrigger aktif edilir. Rigidbody içerisinden constrains z konumu işaretlenir.
* Yeni bir script oluşturularak aşağıdaki kodlar yazılır.

**Düşman Kontrolü (YATAY)**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class deneme : MonoBehaviour

{

public float speed; programda içerisine 2 yazılır

public int turnDelay; programda içerisine 1 yazılır

bool r = false;

// Start is called before the first frame update

private void Start()

{

StartCoroutine(SwitchDirections());

}

// Update is called once per frame

private void Update()

{

transform.Translate(Vector3.right \* speed \* Time.deltaTime);

}

IEnumerator SwitchDirections()

{

yield return new WaitForSeconds(turnDelay);

Switch();

}

private void Switch()

{

r = !r;

Vector3 scaler =transform.localScale;

scaler.x \*= -1;

transform.localScale = scaler;

speed \*= -1;

StartCoroutine(SwitchDirections());

}

private void OnTriggerEnter2D(Collider2D collision)

{

if (collision.gameObject.CompareTag("Player"))

{

collision.GetComponent<Player>().Hit();

}

}

}

* Yukarıdaki kodlar yazıldıktan sonra player için oluşturduğumuz script dosyası içirisinede düzenleme yapabiliriz. Yukarıdaki kodda Hit() kontörlünü kullanmıştık.player içerisinde de Hit için ayrı bir yer oluşturulur.

**Player Kontrol içerisindeki düzenlemeler**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.UI;

using TMPro;

public class Player : MonoBehaviour

{

Rigidbody2D rgb;

Vector3 velocity;

float speedAmount = 5f;

public float jumpAmount;

public Animator animator;

public int life;

public float damageTime;

public Color firstColor;

public Color damageColor;

// Start is called before the first frame update

void Start()

{

rgb = GetComponent<Rigidbody2D>();

firstColor= gameObject.GetComponent<SpriteRenderer>().color;

}

// Update is called once per frame

void Update()

{

velocity = new Vector3(Input.GetAxis("Horizontal"), 0f);

transform.position += velocity \* speedAmount \* Time.deltaTime;

animator.SetFloat("Speed", Mathf.Abs(Input.GetAxis("Horizontal")));

if (Input.GetButtonDown("Jump") && Mathf.Approximately(rgb.velocity.y, 0))

{

rgb.AddForce(Vector3.up \* jumpAmount, ForceMode2D.Impulse);

animator.SetBool("IsJumping", true);

}

if (animator.GetBool("IsJumping") && Mathf.Approximately(rgb.velocity.y, 0))

{

animator.SetBool("IsJumping", false);

}

if (Input.GetAxisRaw("Horizontal") == -1)

{

transform.rotation = Quaternion.Euler(0f, 180f, 0f);

}

else if (Input.GetAxisRaw("Horizontal") == 1)

{

transform.rotation = Quaternion.Euler(0f, 0f, 0f);

}

if(damageTime>0)

{

damageTime -= Time.deltaTime;

if(gameObject.GetComponent<SpriteRenderer>().color == firstColor)

{

gameObject.GetComponent<SpriteRenderer>().color = damageColor;

}

else

{

if (gameObject.GetComponent<SpriteRenderer>().color == damageColor)

{

gameObject.GetComponent<SpriteRenderer>().color = firstColor;

}

}

}

}

private void OnCollisionEnter2D(Collision2D collision)

{

if(collision.gameObject.tag=="enemy" && damageTime<=0)

{

Hit();

}

}

private void OnCollisionStay2D(Collision2D collision)

{

if (collision.gameObject.tag == "enemy" && damageTime <= 0)

{

Hit();

}

}

public void Hit()

{

life--;

Debug.Log("Can : " + life);

damageTime = 0.9f;

gameObject.transform.position = new Vector3(gameObject.transform.position.x-2, gameObject.transform.position.y, gameObject.transform.position.z);

if (life <= 0)

{

SceneManager.LoadScene("SampleScene");

}

}

}

**Düşman Kontrolü (dikey)**

* Collider ve Rigidbody eklenir. Collider içindeki IsTrigger aktif hale getirilir. Rigidbody içerisinden constrains z konumu işaretlenir. Yine Rigidbody içerisinden Gravity Scale 0 yapılır ve yeni bir script açılarak aşağıdaki kodlar yazılır.
* Karakterin oyun içersinde gezdiği alanlar tag’ı ground olarak ayarlanır.

Düşman Kontrolü (DİKEY)

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Oposum : MonoBehaviour

{

public Rigidbody2D rb; programda rigidbody sürüklenir bırakılır

public float speed; programda içerisine 1.5 yazılır

public float minHeight, maxHeight; programda içerisine 1 – 4 yazılır

public LayerMask lm; programda içerisinde Ground seçilir.

// Start is called before the first frame update

void Start()

{

rb.velocity = new Vector2(0, -speed);

}

// Update is called once per frame

void Update()

{

Physics2D.queriesStartInColliders =false;

RaycastHit2D hit = Physics2D.Raycast(gameObject.transform.position, Vector2.down, 8, lm);

if (hit != null)

{

if(hit.distance<=minHeight)

{

rb.velocity = new Vector2(0, speed);

}

else if(hit.distance>=maxHeight)

{

rb.velocity = new Vector2(0, -speed);

}

}

}

private void OnTriggerEnter2D(Collider2D collision)

{

if(collision.gameObject.CompareTag("Player"))

{

collision.GetComponent<Player>().Hit();

}

}

}

**Coin ve life işlemleri**

* Coin= Yaptığımız oyuna altın veya elmas toplama gibi coin işlemleri yapmak için Sunnuy\_Land-artworks-ıtem-gem klsörü içerisinde bulunan resimlerin hepsini seçip oyun penceresinin sol tarafına bırakıyoruz animasyon olarak kaydetmemizi isteyecek animasyonları kaydetmek için bir klasör oluşturmuştuk onun için kayıt edebiliriz.
* Coin= Ekledğimiz objeye tıklayıp bir collider ekliyoruz. Istriger özelliğini aktif ediyoruz. Yeni bir script oluşturuyoruz coin adında ve aşağıdaki kodları yazıyoruz.
* Life= aynı şekilde işlemleri yapmak için Sunnuy\_Land-artworks-ıtem-cherry klsörü içerisinde bulunan resimlerin hepsini seçip oyun penceresinin sol tarafına bırakıyoruz animasyon olarak kaydetmemizi isteyecek animasyonları kaydetmek için bir klasör oluşturmuştuk onun için kayıt edebiliriz.
* Life= collier ekleyip IsTrigger aktif yapıyoruz yeni bir script oluşturup adını cherry olarak veriyoruz ve kodlarımızı yazıyoruz

**Coin (gem)**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class coin : MonoBehaviour

{

private void OnTriggerEnter2D(Collider2D collision)

{

if (collision.tag == "Player")

{

collision.gameObject.GetComponent<Player>().score++;

collision.gameObject.GetComponent<Player>().UpdateScoreText();

Destroy(gameObject);

}

}

}

**Life (cherry)**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class cherry : MonoBehaviour

{

private void OnTriggerEnter2D(Collider2D collision)

{

if (collision.tag=="Player")

{

collision.gameObject.GetComponent<Player>().life++;

collision.gameObject.GetComponent<Player>().UpdateLifeText();

Destroy(gameObject);

}

}

}

* Player scripti içerisine de üst tarafa “public int score; “ yazıp start kısmınada başlangıçta coin değerimiz 0 olacagı için “score = 0;” olarak belirtiyoruz.
* Yukarıdaki UpdateScoreText ve UpdateLifeText yazısı için de playera bu fonksiyonları eklememiz gerekektedir.

**Player içindeki değişiklikler**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

using UnityEngine.UI;

using TMPro;

public class Player : MonoBehaviour

{

Rigidbody2D rgb;

Vector3 velocity;

float speedAmount = 5f;

public float jumpAmount;

public Animator animator;

public int life;

public float damageTime;

public Color firstColor;

public Color damageColor;

public TextMeshProUGUI lifeText;

public int score;

public TextMeshProUGUI playerScoreText;

// Start is called before the first frame update

void Start()

{

rgb = GetComponent<Rigidbody2D>();

score = 0;

firstColor= gameObject.GetComponent<SpriteRenderer>().color;

}

// Update is called once per frame

void Update()

{

velocity = new Vector3(Input.GetAxis("Horizontal"), 0f);

transform.position += velocity \* speedAmount \* Time.deltaTime;

animator.SetFloat("Speed", Mathf.Abs(Input.GetAxis("Horizontal")));

if (Input.GetButtonDown("Jump") && Mathf.Approximately(rgb.velocity.y, 0))

{

rgb.AddForce(Vector3.up \* jumpAmount, ForceMode2D.Impulse);

animator.SetBool("IsJumping", true);

}

if (animator.GetBool("IsJumping") && Mathf.Approximately(rgb.velocity.y, 0))

{

animator.SetBool("IsJumping", false);

}

if (Input.GetAxisRaw("Horizontal") == -1)

{

transform.rotation = Quaternion.Euler(0f, 180f, 0f);

}

else if (Input.GetAxisRaw("Horizontal") == 1)

{

transform.rotation = Quaternion.Euler(0f, 0f, 0f);

}

if(damageTime>0)

{

damageTime -= Time.deltaTime;

if(gameObject.GetComponent<SpriteRenderer>().color == firstColor)

{

gameObject.GetComponent<SpriteRenderer>().color = damageColor;

}

else

{

if (gameObject.GetComponent<SpriteRenderer>().color == damageColor)

{

gameObject.GetComponent<SpriteRenderer>().color = firstColor;

}

}

}

}

private void OnCollisionEnter2D(Collision2D collision)

{

if(collision.gameObject.tag=="enemy" && damageTime<=0)

{

Hit();

}

}

private void OnCollisionStay2D(Collision2D collision)

{

if (collision.gameObject.tag == "enemy" && damageTime <= 0)

{

Hit();

}

}

public void Hit()

{

life--;

Debug.Log("Can : " + life);

lifeText.text = life.ToString();

damageTime = 0.9f;

gameObject.transform.position = new Vector3(gameObject.transform.position.x-2, gameObject.transform.position.y, gameObject.transform.position.z);

if (life <= 0)

{

SceneManager.LoadScene("SampleScene");

}

}

public void UpdateLifeText()

{

lifeText.text = life.ToString();

}

public void UpdateScoreText()

{

playerScoreText.text = score.ToString();

}

}

**Sahne Geçişi**

Bu kodu player scripti içine yazmanız gerekmektedir. Kapı simgesine gelince sahnenin değişmesini sağlayacaktır.

Bunun için kapının tagını ‘door’ olarak ayarlayın ve collider 2D ekleyin ve is triger özelliğini aktif hale getirin.

public void OnTriggerEnter2D(Collider2D collision)

{

if (collision.tag == "door")

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex+1);

}

}