Mod de realizare

Mai intai am importat din “Asset store” diverse modele universale, toate potrivite pentru a crea o tematica abstracta, a spatiului cosmic, la care am facut diverse modificari proprii. Pe urma, am adaugat totul in scena, imaginea de fundal, platformele si jucatorul.

Pentru a crea miscarea jucatorului in mediul jocului, am adaugat componenta rigidbody2D pentru a adauga o fizica implicita caracterului, apoi am creat si adaugat character controller-ul, acesta fiind un script care defineste miscarea propriu zisa.

Pentru a defini miscarea orizontala a caracterului am folosit secventa:

Vector3 targetVelocity = new Vector2(move \* 10f, m\_Rigidbody2D.velocity.y);

m\_Rigidbody2D.velocity = Vector3.SmoothDamp(m\_Rigidbody2D.velocity, targetVelocity, ref m\_Velocity, m\_MovementSmoothing);

Pentru a define saritura caracterului am folosit secventa:

if (m\_Grounded && jump)

{

// Adauga o forta verticala caracterului daca acesta este pe pamant

m\_Grounded = false;

m\_Rigidbody2D.AddForce(new Vector2(0f, m\_JumpForce));

animator.SetBool("Sare", true);

}

Pentru a define ghemuirea caracterului am folosit secventa:

// Pentru a verifica daca caracterul se poate ridica din ghemuire

if (!crouch)

{

// Daca caracterul are un tavan deasupra capului, nu se poate ridica din ghemuire

if (Physics2D.OverlapCircle(m\_CeilingCheck.position, k\_CeilingRadius, m\_WhatIsGround))

{

crouch = true;

}

}

//numai daca playerul este pe pamant, sau daca controlul in aer este activat

if (m\_Grounded || m\_AirControl)

{

// daca era ghemuit

if (crouch)

{

if (!m\_wasCrouching)

{

m\_wasCrouching = true;

OnCrouchEvent.Invoke(true);

}

// reduce viteza in pozitia ghemuit

move \*= m\_CrouchSpeed;

// dezactiveaza collider-ul partii superioare a corpului cand playerul e sta ghemuit

if (m\_CrouchDisableCollider != null)

{

m\_CrouchDisableCollider.enabled = false;

m\_CrouchDisableCollider2.enabled = false;

}

} else

{

// reactiveaza collider-ul partii superioare a corpului cand playerul se ridica in picioare

if (m\_CrouchDisableCollider != null)

{

m\_CrouchDisableCollider.enabled = true;

m\_CrouchDisableCollider2.enabled = true;

}

if (m\_wasCrouching)

{

m\_wasCrouching = false;

OnCrouchEvent.Invoke(false);

}

}

Pentru a folosi miscarile in cadrul caracterului, i-am atribuit acestuia un script care contine secventele:

Pentru miscarea orizontala:

miscareOrizontala = Input.GetAxisRaw("Horizontal") \* vitezaMers;

si

controller.Move(miscareOrizontala \* Time.fixedDeltaTime, ghemuire, saritura);

Pentru saritura:

if (Input.GetButtonDown("Jump"))

{

saritura = true;

}

Si

saritura = false;

Pentru ghemuire:

if (Input.GetButtonDown("Crouch"))

{

ghemuire = true;

}

else if (Input.GetButtonUp("Crouch"))

{

ghemuire = false;

}

Pentru crearea animatiilor am pus caracterul manual in pozitii diferite si progresive, pentru a produce o miscari formata din cateva frame-uri, accentuand caracterul “pixel art” pe care l-am urmarit, dupa care am legat totul de caracter cu un animator controller pe care l-am creat, care conditiona in ce moment se va derula fiecare animatie. Pe urma am legat conditiile folosite in animator controller la joc adaugand niste secvente de cod la scripturile de miscare ale jucatorului:

Pentru a-i indica animatorului daca caracterul trebuie sa se miste sau sa stea pe loc am folosit:

animator.SetFloat("Viteza", Mathf.Abs(miscareOrizontala));

Pentru a-i indica animatorului saritura caracterului, am adaugat o secventa langa codul care aplica forta verticala caracterului:

animator.SetBool("Sare", true);

Pentru a-i indica animatorului sfarsitul sariturii am adaugat un eveniment care defineste aterizarea pe pamant a jucatorului, si apoi l-am folosit intr-o secventa:

if (OnLandEvent == null)

OnLandEvent = new UnityEvent();

bool wasGrounded = m\_Grounded;

m\_Grounded = false;

// Playerul atinge pamantul daca collider-ul pe care l-am adaugat la nivelul picoarelor si l-am numit GroundCheck atinge cea ce am definit ca a fi pamant

Collider2D[] colliders = Physics2D.OverlapCircleAll(m\_GroundCheck.position, k\_GroundedRadius, m\_WhatIsGround);

for (int i = 0; i < colliders.Length; i++)

{

if (colliders[i].gameObject != gameObject)

{

m\_Grounded = true;

if (!wasGrounded)

OnLandEvent.Invoke();

}

}

Si

public void OnLanding()

{

animator.SetBool("Sare", false);}

Pentru a-i indica animatorului ghemuirea am adaugat un eveniment care defineste ghemuirea jucatorului, si apoi l-am folosit intr-o secventa:

if (OnCrouchEvent == null)

OnCrouchEvent = new BoolEvent();

if (!m\_wasCrouching)

{

m\_wasCrouching = true;

OnCrouchEvent.Invoke(true);

}

public void OnCrouch(bool ghemuieste)

{

animator.SetBool("Ghemuieste", ghemuieste);

}

Pentru a defini moartea jucatorului am pus intr-un script separat, pe care l-am numit “game manager”, secventa:

public void Moartea()

{

if (mort == false)

{

mort = true;

Restart();

}

}

void Restart()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().name);

}

Pentru a face ca coliziunea cu un obstacol de tip teapa sa ii scada o viata jucatorului am adaugat intr-un script legat de player secventa:

void OnCollisionEnter2D(Collision2D collisionInfo)

{

if(collisionInfo.collider.tag == "Teapa")

FindObjectOfType<GameManager>().Moartea();

}

Pentru trecerea la urmatorul nivel am adaugat un obiect de tip trigger, asupra caruia am aplicat un script:

void OnTriggerEnter2D()

{

gameManager.NivelulUrmator();

}

public void NivelulUrmator()

{

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

}