Міністерство освіти і науки України

Прикарпатський національний університет

імені Василя Стефаника

*Факультет математики та інформатики*

*Кафедра інформаційних технологій*

Лабораторна робота № 3з дисципліни   
"Програмування ігрових застосувань"

Тема: Вступ до Unity 3D. Гра «Смуга перешкод».

Виконав: Совтус А.А.

Група ІПЗ-33

Дата: 10 квітня 2024р.

Викладач: Горєлов В.О.

Івано-Франківськ – 2024

**Завдання.**

Завдання гравця: переміщуватися уздовж прямої, вибираючи вдалі моменти для початку руху.

Перпендикулярно до напрямку руху гравця здійснюють коливальні рухи предмети (наприклад, мішки з піском, що гойдаються на мотузці) або зворотно-поступальні рухи (управо-уліво чи вгору-униз). Предмети коливаються чи переміщуються із різним періодом. Таким чином, пройшовши повз одну перешкоду, гравцеві потрібно буде подолати іншу, адаптувавшись до характеру її коливання чи переміщення.

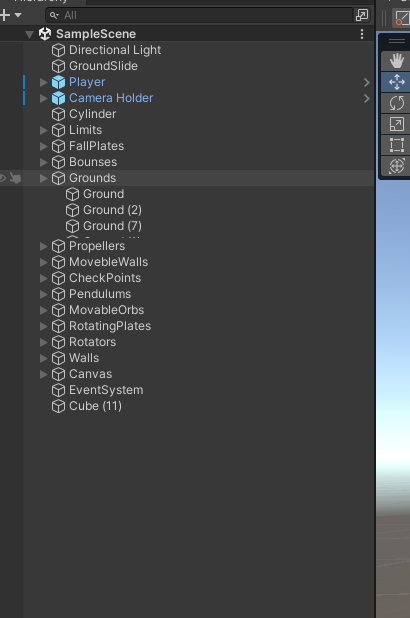
Після проходження усіх перешкод у даному напрямі гра продовжується у зворотному.

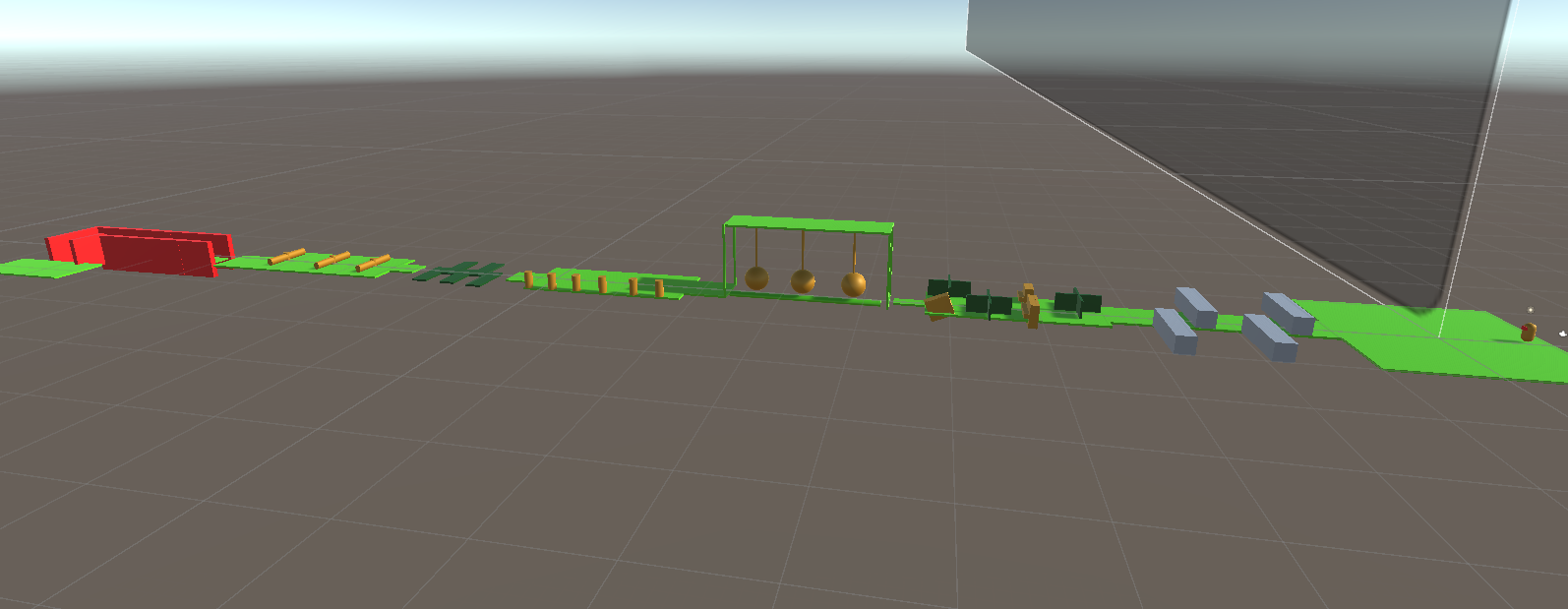
Тип перешкод та характер їхнього руху виберіть самостійно. Гравець не повинен залишатися нерухомим більше 5 секунд після завершення переміщення. Якщо цей час перевищено – гра завершується.

Передбачте виведення на екран рахунку (кількості успішних кроків).

Переміщення персонажа може бути дискретним або неперервним (поки гравець утримує кнопку курсора).

**GAME**





Main logic |

using UnityEngine;

using System.Collections;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

[RequireComponent (typeof (Rigidbody))]

[RequireComponent (typeof (CapsuleCollider))]

public class CharacterControls : MonoBehaviour {

[SerializeField] private GameObject endPanel;

public float speed = 10.0f;

public float airVelocity = 8f;

public float gravity = 10.0f;

public float maxVelocityChange = 10.0f;

public float jumpHeight = 2.0f;

public float maxFallSpeed = 20.0f;

public float rotateSpeed = 25f; //Speed the player rotate

private Vector3 moveDir;

public GameObject cam;

private Rigidbody rb;

private float distToGround;

private bool canMove = true; //If player is not hitted

private bool isStuned = false;

private bool wasStuned = false; //If player was stunned before get stunned another time

private float pushForce;

private Vector3 pushDir;

public Vector3 checkPoint;

private bool slide = false;

private float idleTimer = 0f;

private bool isIdle = false;

void Start (){

endPanel.SetActive (false);

// get the distance to ground

distToGround = GetComponent<Collider>().bounds.extents.y;

}

bool IsGrounded (){

return Physics.Raycast(transform.position, -Vector3.up, distToGround + 0.1f);

}

void Awake () {

rb = GetComponent<Rigidbody>();

rb.freezeRotation = true;

rb.useGravity = false;

checkPoint = transform.position;

Cursor.visible = false;

}

void FixedUpdate () {

if (canMove)

{

if (moveDir.x != 0 || moveDir.z != 0)

{

Vector3 targetDir = moveDir; //Direction of the character

targetDir.y = 0;

if (targetDir == Vector3.zero)

targetDir = transform.forward;

Quaternion tr = Quaternion.LookRotation(targetDir); //Rotation of the character to where it moves

Quaternion targetRotation = Quaternion.Slerp(transform.rotation, tr, Time.deltaTime \* rotateSpeed); //Rotate the character little by little

transform.rotation = targetRotation;

}

if (IsGrounded())

{

// Calculate how fast we should be moving

Vector3 targetVelocity = moveDir;

targetVelocity \*= speed;

// Apply a force that attempts to reach our target velocity

Vector3 velocity = rb.velocity;

if (targetVelocity.magnitude < velocity.magnitude) //If I'm slowing down the character

{

targetVelocity = velocity;

rb.velocity /= 1.1f;

}

Vector3 velocityChange = (targetVelocity - velocity);

velocityChange.x = Mathf.Clamp(velocityChange.x, -maxVelocityChange, maxVelocityChange);

velocityChange.z = Mathf.Clamp(velocityChange.z, -maxVelocityChange, maxVelocityChange);

velocityChange.y = 0;

if (!slide)

{

if (Mathf.Abs(rb.velocity.magnitude) < speed \* 1.0f)

rb.AddForce(velocityChange, ForceMode.VelocityChange);

}

else if (Mathf.Abs(rb.velocity.magnitude) < speed \* 1.0f)

{

rb.AddForce(moveDir \* 0.15f, ForceMode.VelocityChange);

//Debug.Log(rb.velocity.magnitude);

}

// Jump

if (IsGrounded() && Input.GetButton("Jump"))

{

rb.velocity = new Vector3(velocity.x, CalculateJumpVerticalSpeed(), velocity.z);

}

}

else

{

if (!slide)

{

Vector3 targetVelocity = new Vector3(moveDir.x \* airVelocity, rb.velocity.y, moveDir.z \* airVelocity);

Vector3 velocity = rb.velocity;

Vector3 velocityChange = (targetVelocity - velocity);

velocityChange.x = Mathf.Clamp(velocityChange.x, -maxVelocityChange, maxVelocityChange);

velocityChange.z = Mathf.Clamp(velocityChange.z, -maxVelocityChange, maxVelocityChange);

rb.AddForce(velocityChange, ForceMode.VelocityChange);

if (velocity.y < -maxFallSpeed)

rb.velocity = new Vector3(velocity.x, -maxFallSpeed, velocity.z);

}

else if (Mathf.Abs(rb.velocity.magnitude) < speed \* 1.0f)

{

rb.AddForce(moveDir \* 0.15f, ForceMode.VelocityChange);

}

}

}

else

{

rb.velocity = pushDir \* pushForce;

}

if (rb.velocity.magnitude <= 0.1f && IsGrounded())

{

idleTimer += Time.deltaTime;

if (idleTimer >= 5f)

{

EndGame();

}

}

else

{

idleTimer = 0f;

}

// We apply gravity manually for more tuning control

rb.AddForce(new Vector3(0, -gravity \* GetComponent<Rigidbody>().mass, 0));

}

private void Update()

{

float h = Input.GetAxis("Horizontal");

float v = Input.GetAxis("Vertical");

Vector3 v2 = v \* cam.transform.forward; //Vertical axis to which I want to move with respect to the camera

Vector3 h2 = h \* cam.transform.right; //Horizontal axis to which I want to move with respect to the camera

moveDir = (v2 + h2).normalized; //Global position to which I want to move in magnitude 1

RaycastHit hit;

if (Physics.Raycast(transform.position, -Vector3.up, out hit, distToGround + 0.1f))

{

if (hit.transform.tag == "Slide")

{

slide = true;

}

else

{

slide = false;

}

}

}

float CalculateJumpVerticalSpeed () {

// From the jump height and gravity we deduce the upwards speed

// for the character to reach at the apex.

return Mathf.Sqrt(2 \* jumpHeight \* gravity);

}

public void HitPlayer(Vector3 velocityF, float time)

{

rb.velocity = velocityF;

pushForce = velocityF.magnitude;

pushDir = Vector3.Normalize(velocityF);

StartCoroutine(Decrease(velocityF.magnitude, time));

}

public void LoadCheckPoint()

{

transform.position = checkPoint;

}

private IEnumerator Decrease(float value, float duration)

{

if (isStuned)

wasStuned = true;

isStuned = true;

canMove = false;

float delta = 0;

delta = value / duration;

for (float t = 0; t < duration; t += Time.deltaTime)

{

yield return null;

if (!slide) //Reduce the force if the ground isnt slide

{

pushForce = pushForce - Time.deltaTime \* delta;

pushForce = pushForce < 0 ? 0 : pushForce;

//Debug.Log(pushForce);

}

rb.AddForce(new Vector3(0, -gravity \* GetComponent<Rigidbody>().mass, 0)); //Add gravity

}

if (wasStuned)

{

wasStuned = false;

}

else

{

isStuned = false;

canMove = true;

}

}

public void PlayAgain()

{

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

}

void EndGame()

{

Debug.Log("Гра завершена.");

endPanel.SetActive(true);

Cursor.visible = true;

}

}

}