Ministerul Educației al Republicii Moldova
Universitatea Tehnică a Moldovei

RAPORT

Lucrarea de laborator nr. 5
Medii Interactive de dezvoltare a produselor software
Tema: Dezvoltarea unei aplicatii mobile

A efectuat:
st. gr. TI-144 Gorduz Daniel

A verificat:
lect. univ. Irina Cojan

Chișinău 2016

Tema: Dezvoltarea unei aplicatii mobile

Obiectivele lucrării:

- Cunostinte de baza privina arhitectura unei aplicatii mobile;
- Cunostinte de baza ale platformei SDK.

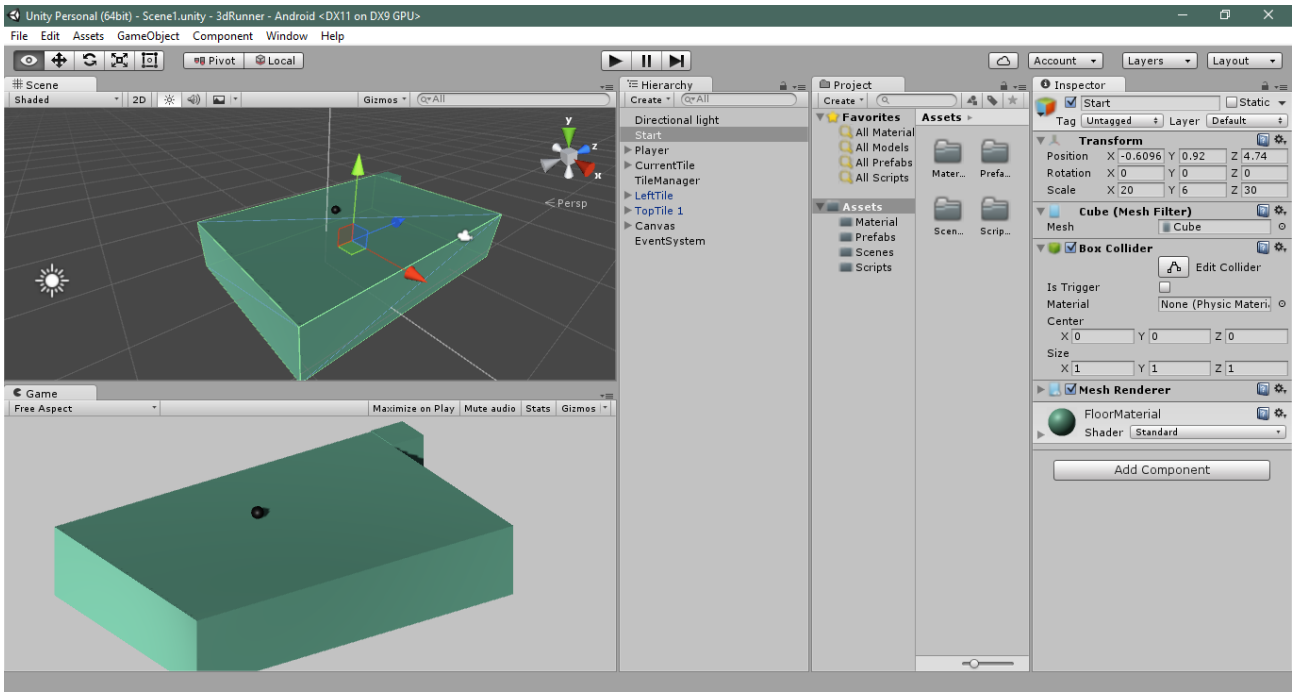
Modul de lucru:

In cadrul acestui laborator a fost propus de creat o aplicatie mobila si anume pe Android. Aplicatia mea este o aplicatie de gen:

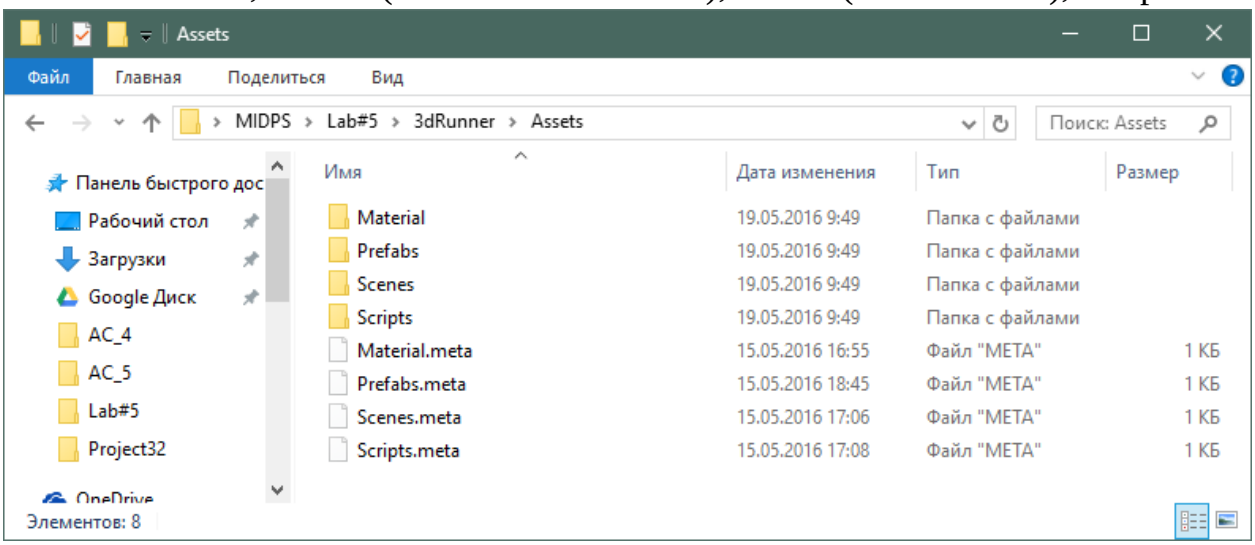
- Platformer 2.5D;
- EndlessRunner.

Pentru acesta aplicatie a fost folosita mediu Unity3D & C#.

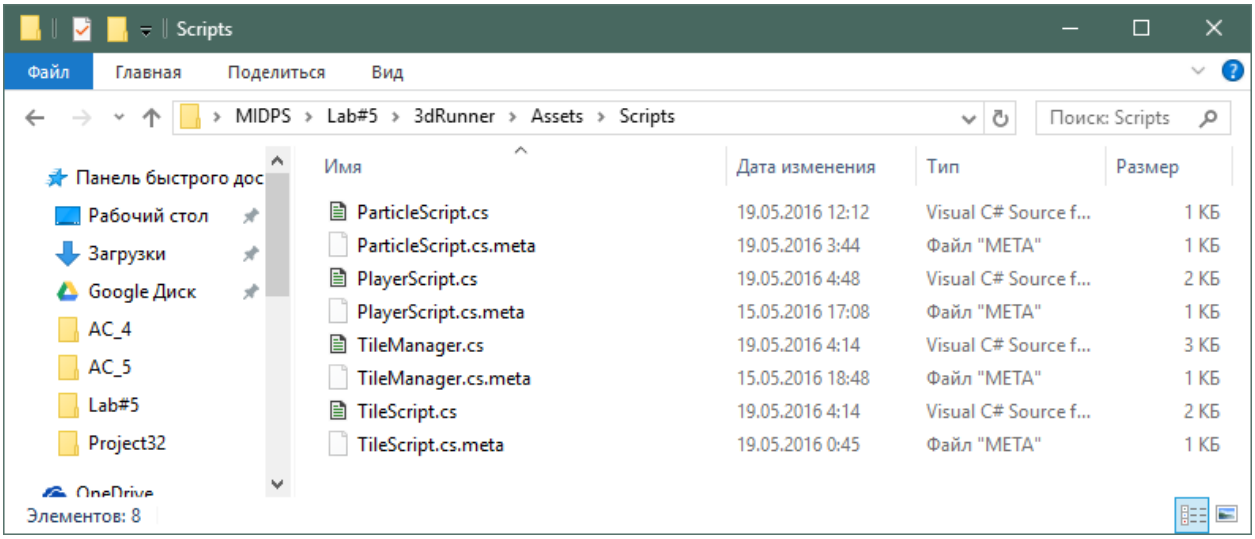
1. Astfel cu ajutorul componentelor integrate din Unity au fost adaugate: platforma, player(sfera), platformele temporare create random si butoanele cu text de afisat, a fizicii si a materialelor acestor obiecte.



2. Mapa Assets contine resursele necesare de a modifica si dezvolta aplicatia. Si anume: Material, Prefabs(Carcase si structurile), Scenes(scena salvata), Scripts.



3. Script-urile create folosind C# au permis modificarea proprietatilor si functiilor de interactiune dintre aceste obiecte create.



3.1. **Particle Script** contine script-ul pentru distrugerea acestor particule odata dupa ce Player a trecut peste Collider-ul al Tile.

```
ublic class ParticalScript : MonoBehaviour {
    private ParticleSystem ps;
   // Use this for initialization
   void Start () {
        ps = GetComponent<ParticleSystem>();
   }

   // Update is called once per frame
   void Update () {
        if (!ps.isPlaying){
            Destroy(gameObject);
        }
   }
}
```

3.2. Player Script contine script-ul pentru determinarea daca player-ul a iesit de pe platforma, pentru determinarea directiei da deplasare, determinarea scorului si aplicarea fizicii asupra player-ului.

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
public class PlayerScript : MonoBehaviour {
    public float speed;

    private Vector3 dir;

    public GameObject ps;

    private bool isDead;

    public GameObject resetBtn;

    private int score = 0;

    public Text scoreText;
       // Use this for initialization
       void Start () {
        isDead = false;
        dir = Vector3.zero;
       }

       // Update is called once per frame
       void Update () {
           if (Input.GetMouseButtonDown(0) && !isDead){
            score++;
            scoreText.text = score.ToString();
            if (dir == Vector3.forward)
                dir = Vector3.left;
            else
                dir = Vector3.forward;
        }
        float amountToMove = speed * Time.deltaTime;
        transform.Translate(dir * amountToMove);
       }

    void OnTriggerEnter(Collider other){
        if (other.tag == "Pickup")
        {
            other.gameObject.SetActive(false);
            Instantiate(ps, transform.position, Quaternion.identity);
            score+=3 ;
            scoreText.text = score.ToString();
        }
    }
    void OnTriggerExit(Collider other){
        if (other.tag == "Tile"){
            RaycastHit hit;
            Ray downRay = new Ray(transform.position, -Vector3.up);
            if(!Physics.Raycast(downRay, out hit)){
                //DEATH!!
                isDead = true;
                resetBtn.SetActive(true);
                transform.GetChild(0).transform.parent = null;
            }
        }
    }
}
```

3.3. Tile Manager Script contine toate functiile ce tin de spawn-ul platformelor si distrugerea lor indata dupa ce s-a deplasat pe el Player-ul vor avea animatia caderii si distrugerii pentru a nu incarca la maxim memoria.

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class TileManager : MonoBehaviour {

    public GameObject[] tilePrefabs;

    public GameObject currentTile;

    private static TileManager instance;

    private Stack<GameObject> leftTiles = new Stack<GameObject>();

    private Stack<GameObject> topTiles = new Stack<GameObject>();

    public static TileManager Instance{
        get{
            if (instance == null)
                instance = GameObject.FindObjectOfType<TileManager>();
            return instance;
        }
    }

    public Stack<GameObject> LeftTiles{
        get{
            return leftTiles;
        }
        set{
            leftTiles = value;
        }
    }

    public Stack<GameObject> TopTiles{
        get{
            return topTiles;
        }
        set{
            topTiles = value;
        }
    }

    // Use this for initialization
    void Start () {
        CreateTiles(100);

        for (int i = 0; i < 50; i++){
            SpawnTile();
        }
       }

       // Update is called once per frame
       void Update () {

       }

    public void CreateTiles(int amount)
    {
```

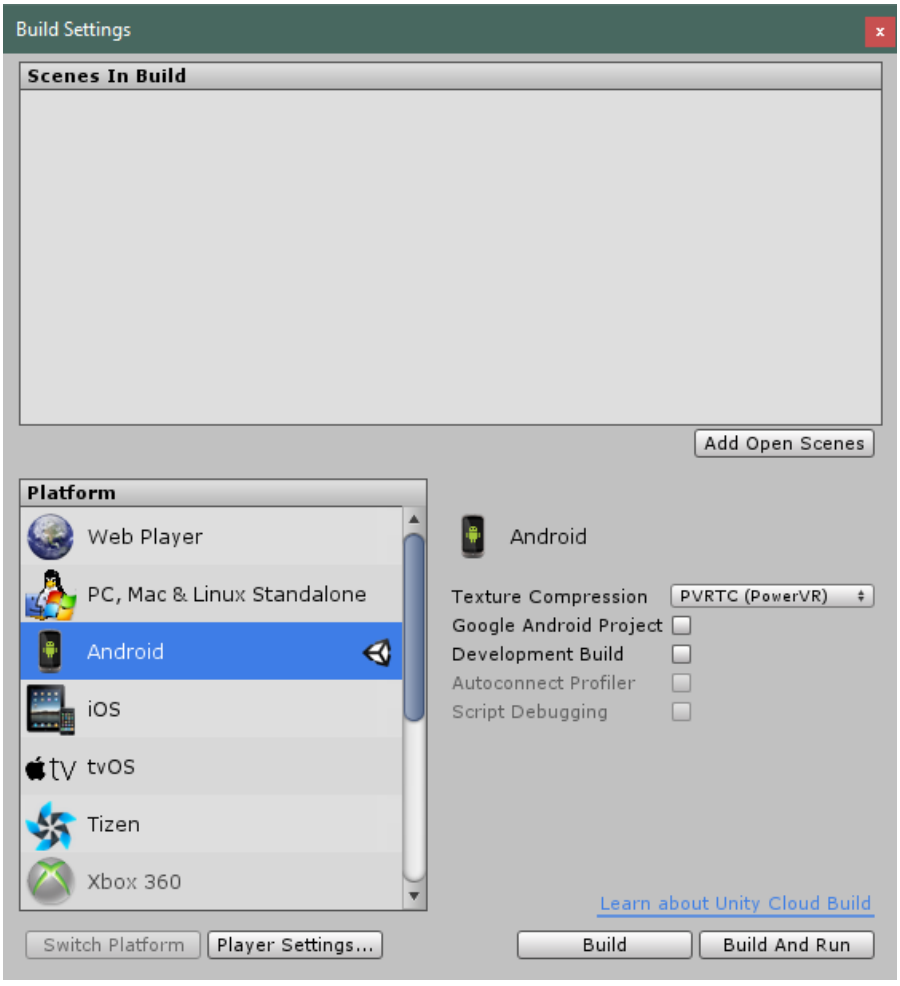
```
        for(int i = 0; i < amount; i++)
        {
            LeftTiles.Push(Instantiate(tilePrefabs[0]));
            TopTiles.Push(Instantiate(tilePrefabs[1]));
            TopTiles.Peek().name = "topTiles";
            TopTiles.Peek().SetActive(false);
            TopTiles.Peek().name = "topTiles";
            LeftTiles.Peek().SetActive(false);
        }
    }

    public void SpawnTile()
    {
        if(LeftTiles.Count==0 || TopTiles.Count == 0)
        {
            CreateTiles(10);
        }
        //Generam un nr random  intre 0 si 1
        int randomIndex = Random.Range(0, 2);
        if (randomIndex == 0)
        {
            GameObject tmp = LeftTiles.Pop();
            tmp.SetActive(true);
            tmp.transform.position =
currentTile.transform.GetChild(0).transform.GetChild(randomIndex).position;
            currentTile = tmp;
        }
        else if (randomIndex == 1)
        {
            GameObject tmp = TopTiles.Pop();
            tmp.SetActive(true);
            tmp.transform.position =
currentTile.transform.GetChild(0).transform.GetChild(randomIndex).position;
            currentTile = tmp;
        }
        int spawnPickup = Random.Range(0, 10);
        if (spawnPickup == 0)
        {
            currentTile.transform.GetChild(1).gameObject.SetActive(true);
        }
}
    public void ResetGame()
    {
        Application.LoadLevel(Application.loadedLevel);
    }
}
```

3.4. **Tile Script** ține de instatierea platformelor generate si precum delay-ul de cadere de dupa ce Player.tag a actionat cu collider-ul platformei.

```
using UnityEngine;
using System.Collections;

public class TileScript : MonoBehaviour {
    private float fallDelay = 0.8f;

    // Use this for initialization
    void Start () {

       }
       // Update is called once per frame
       void Update () {
```

```
    }
    void OnTriggerExit(Collider other) {
        if (other.tag == "Player")
            TileManager.Instance.SpawnTile();
        StartCoroutine(FallDown());
    }
    IEnumerator FallDown() {
        yield return new WaitForSeconds(fallDelay);
        GetComponent<Rigidbody>().isKinematic = false;
        yield return new WaitForSeconds(2);
        switch (gameObject.name){
            case "leftTile":
                TileManager.Instance.LeftTiles.Push(gameObject);
                gameObject.GetComponent<Rigidbody>().isKinematic = true;
                gameObject.SetActive(false);
                break;
            case "topTile":
                TileManager.Instance.TopTiles.Push(gameObject);
                gameObject.GetComponent<Rigidbody>().isKinematic = true;
                gameObject.SetActive(false);
                break;
        }
    }
}
```

4. Aplicatia a fost transferata pe Android prin optiunea „Build and Run”, ce permite crearea buil-ului si rularea lui pe orice plaforma din optiuni si tranferarea aplicatiei deodata pe dispozitivul conectat prin cablu USB.



Rezultatul final:

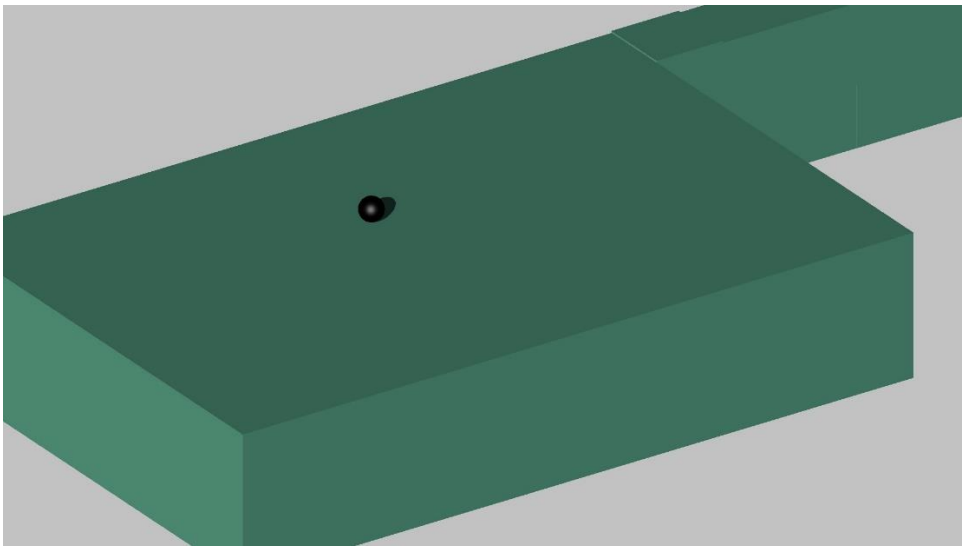


Fig. 1.1 Platforma de baza

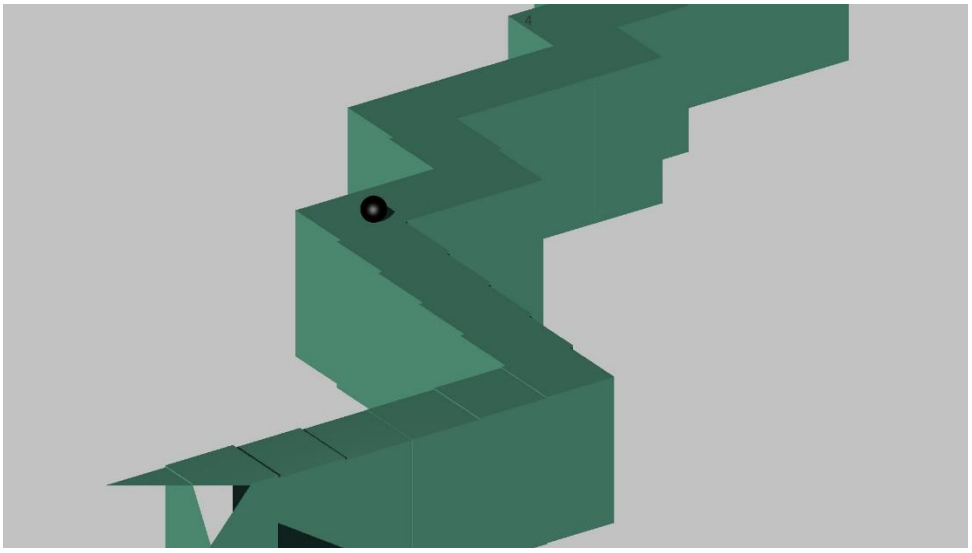


Fig. 1.2 Aplicatia in rulare

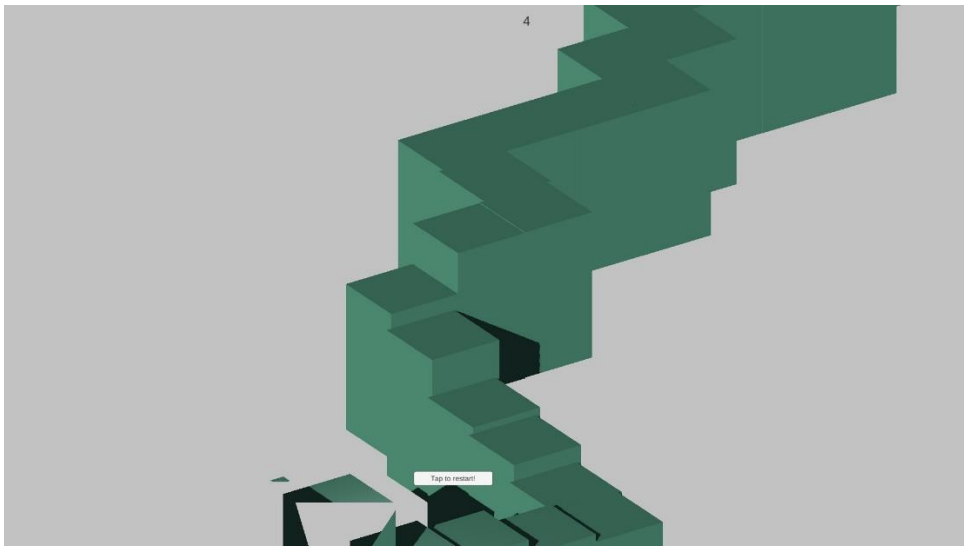


Fig. 1.3. Optiunea de arestarta si rezultatul afisat

Concluzie

In urma efectuarii acestui laborator am facut cunostinta cu procesul de creare a unei aplicatii, si in cazul dat pe Android. Ne-am familiarizat cu mediul Unity3D si C#. Putem spuna ca Unity3D este mai comod decit AndroidStudio, deoarece acesta permite crearea fisierul executabil pe multe platforme (IOS, Android, Windows, Tizen, Xbox si altele) precum si usurarea utilizarii datorita preseturilor de fizica, materiale s. a.

Bibliografie

http://docs.unity3d.com/Manual/index.html
http://www.unity3dstudent.com/