

COLEGIUL NAȚIONAL „MIHAIL KOGĂLNICEANU”

LUCRARE PENTRU SUSȚINEREA
EXAMENULUI DE ATESTARE
PROFESIONALĂ ÎN INFORMATICĂ

Aplicatii ale interfețelor grafice

PROFESOR COORDONATOR:

NEAGU VIOLETA

ELEV: POPOIU DARIA

CLASA: a XII – a F

GALAȚI

2022

Cuprins

1. Tema proiectului
2. Considerații teoretice
3. Cerințele și specificațiile aplicației
4. Prezentarea aplicației
5. Concluzie
6. Bibliografie
7. Anexe

CAPITOLUL 1

→Tema proiectului

Să se realizeze o aplicație software implementată în limbajul C#, care utilizează platforma vizuală Unity, utilizând programarea orientată pe obiecte, care să permită interacționarea cu utilizatorul intr-un joc de tip arcade.

CAPITOLUL 2

→Consideratii teoretice

★ Mediul de programare Unity

Unity este un motor de joc multiplatform (smartphone , computer , console video și jocuri web) dezvoltat de Unity Technologies. Este unul dintre cele mai utilizate pe scară largă în industria jocurilor video, atât pentru studiourile mari, cât și pentru cei independenți, datorită rapidității sale în realizarea de prototipuri și faptului că permite lansarea jocurilor pe toate mediile.

Software - ul are particularitatea de a folosi codul (C #) pe „ NET “, o platformă cu Mono Develop . Editorul său se baza anterior pe MonoDevelop, prin MonoDevelop-Unity, dar începând cu versiunea 2018.1 se bazează pe Visual Studio Community .

★Programarea orientata pe obiecte

Programarea orientată pe obiecte (POO, în limba engleză, Object Oriented Programming (OOP)) este o paradigmă de programare, axată pe ideea grupării datelor și codului care operează asupra lor, într-o singură structură. Un alt concept important asociat programării orientate pe obiecte este polimorfismul, care permite abstractizări ce permit o descriere conceptuală și mai simplă a soluției.

Programarea orientată pe obiecte este unul din cei mai importanți pași făcuți în evoluția limbajelor de programare. Odată cu descoperirea programării pe obiecte, a fost descoperit și conceptul de clasă. O clasa grupează datele și unitățile de prelucrare a acestora într-un modul, unindu-le astfel într-o entitate mult mai naturală. Deși tehnica se numește "Programare Orientată Obiectual", conceptul de bază al ei este clasa. Clasa, pe lângă faptul că abstractizează foarte mult analiza/sinteza

problemei, are proprietatea de generalitate, ea desemnând o mulțime de obiecte care împart o serie de proprietăți.

★ Limbajul C#

C# este un limbaj de programare orientat-obiect conceput de Microsoft la sfârșitul anilor 90. A fost conceput ca un concurent pentru limbajul Java și este considerat un derivat al limbajului de programare C++.

C# este un limbaj simplu, cu circa 80 de cuvinte cheie și 12 tipuri de date predefinite. El permite programarea structurată, modulară și orientată obiectual, conform perceptelor moderne ale programării profesioniste.

Principiile de bază ale programării orientate pe obiecte sunt elemente fundamentale ale programării C#. În mare, limbajul moștenește sintaxa și principiile de programare din C++. Sunt o serie de tipuri noi de date sau funcțiuni diferite ale datelor din C++, iar în spiritul realizării unor secvențe de cod sigure, unele funcțiuni au fost adăugate (de exemplu, interfețe și delegări), diversificate (tipul struct), modificate (tipul string) sau chiar eliminate (moștenirea multiplă și pointerii către funcții). Unele funcțiuni (cum ar fi accesul direct la memorie folosind pointeri) au fost păstrate, dar secvențele de cod corespunzătoare se consideră „nesigure”.

CAPITOLUL 3

→Cerințele și specificațiile aplicatiei

Aplicația va permite utilizatorului să controleze acțiunile unui obiect 2D în timp real.

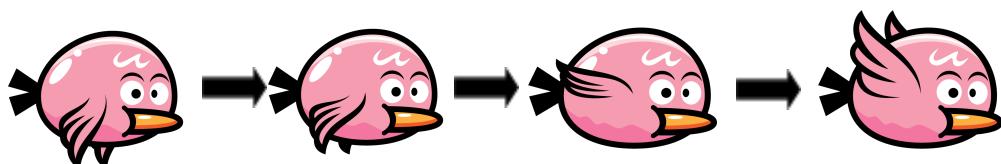
Utilizatorul are posibilitatea de a interacționa cu alte elemente 2D în funcție de acțiunile utilizatorului și dinamica obiectului controlat.

CAPITOLUL 4

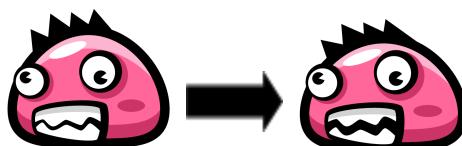
→ Prezentarea aplicatiei

Obiectul principal al aplicatiei este reprezentat de o pasare, ce are drept scop “uciderea” inamicilor.

Pe tot parcursul aplicatiei, pasarea este animata astfel incat sa se creeze impresia unor aripi in miscare :



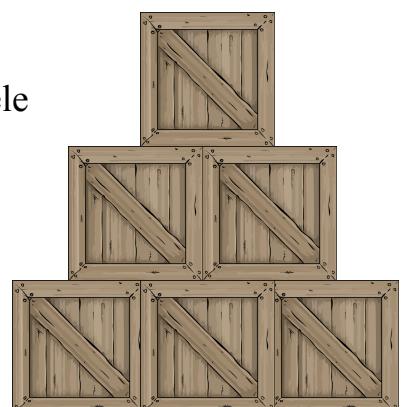
Inamicii din mini-game au fost animati pentru a imita comportamentul unui “monstru furios”.



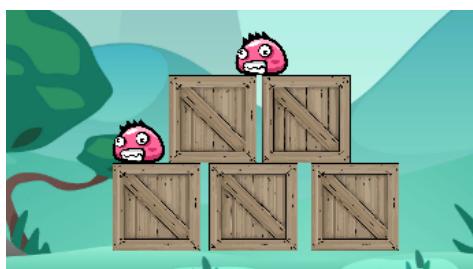
❖ Interactiunea personaj-obstacol

In aplicatie intalnim obstacole in forma de cutii, personajele interactionand cu acestea in urmatoarele moduri:

- ★ Cutiile sunt considerate obstacole de catre pasare, intrucat cutiile vor modifica traseul personajului Pasare sau pot il opri din a “ucide” inamicii



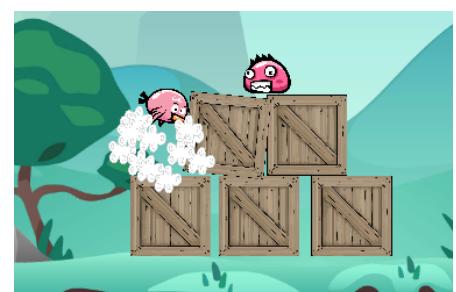
- ★ Monstruletii pot fi asezati pe cutii la inceputul fiecarui nivel si este posibila “uciderea” lor daca cutiile vor cadea deasupra personajului



❖ Uciderea inamicilor



Personajul Pasare isi va ucide inamicii printr-o simpla atingere, disparitia lor fiind evidențiată cu ajutorul unei animații de tip disperie de imagine.

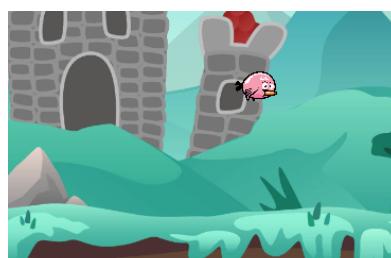


❖ Lansarea obiectului

Pasarea va fi controlată de către utilizator prin intermediul click-ului. Pentru a modifica direcția și viteza deplasării obiectului, utilizatorul trebuie să mențină click-ul deasupra acestuia.



In plus, cât timp pasarea este accesată, imaginea acesteia se va înroși, iar traiectoria obiectului va putea fi vizibilă pentru utilizator, fiind reprezentată de o serie de sageti albe.

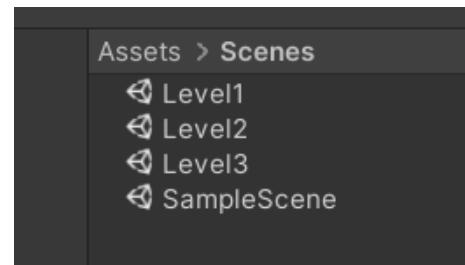


Odată ce cursorul eliberează pasarea, culoarea rosie va dispărea, iar aceasta va “zbura” către direcția dorită.

❖ Controlul nivelelor

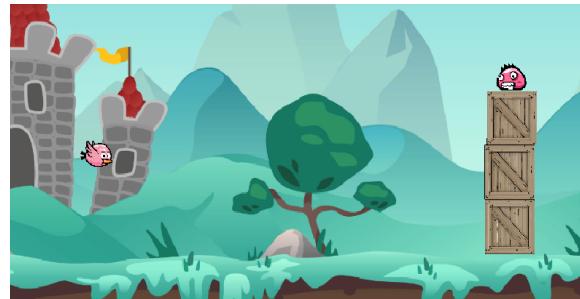
Pentru a trece de la un nivel la altul, trebuie ca toti inamicii din nivelul curent sa fie ucisi. Schimbarea nivelelor este posibila printr-un obiect numit “LevelController”, ce verifica daca mai sunt inamici in scena si va contoriza nivelele jucate.

Toate nivelele sunt salvate in asset-ul “Scenes”, unde fiecare nivel este considerat o “scena aranjata” de creatorul aplicatiei.



Aplicatia are doar 3 nivale prestabilite, fiecare reprezentand un grad de dificultate :

1. Usor



2. Mediu

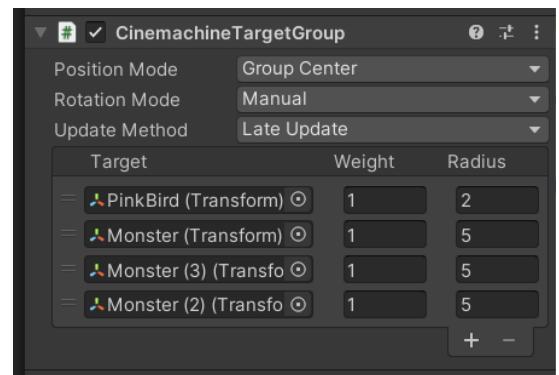


3. Dificil

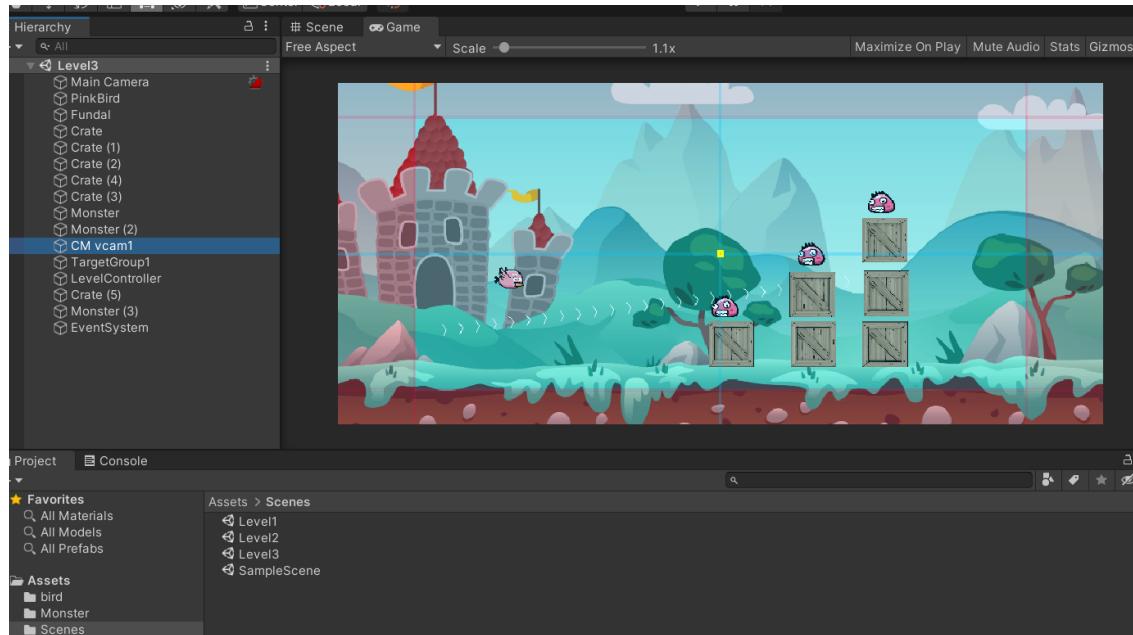


❖ Camera obiectului

Se utilizeaza o componenta a extensiei “Cinemachine” numita “Target Group” care urmareste in acelasi timp atat parcursul păsării, cat si a inamicilor, pentru a crea un efect de target.



Cu ajutorul acestui obiect, pe parcursul jocului, “camera” aplicatiei va pastra in campul vizual toate personajele din scena.



CAPITOLUL 5

→Concluzii

Aplicația prezentata este momentan in stadiul de versiune beta. Avand in vedere unele lipsuri ale jocului, in viitor se pot face urmatoarele imbunatatiri:

- ★ Crearea unui meniu cu butoane de tip “Restart” , “Pause” , “Resume”.
- ★ Modificarea obiectului “Target Group” astfel incat camera sa nu depaseasca cadranul aplicatiei.
- ★ Adaugarea unui numar mai mare de nivale.
- ★ Crearea mai multor tipuri de inamici.

CAPITOLUL 6

→Bibliografie

★<https://ro.wikipedia.org/>

★<https://stackoverflow.com/>

★<https://unity.com/>

★<https://opengameart.org/content/a-pixel-art-collection>

CAPITOLUL 7

→Anexe

★ Anexa 1 : Script-urile fiecarui obiect

1. Script-ul Bird.cs pentru obiectul Pasare

```
using UnityEngine;
using UnityEngine.SceneManagement;

/// Vector3 --> 3 dimensional vector ex (x,y,z)
/// Vector2 --> 2 dimensional vector ex (x,y)
public class Bird : MonoBehaviour
{
    private Vector3 _initialPosition;
    private bool _birdWasLaunched;
    private float _timeSittingAround;

    [SerializeField] private float _launchPower = 500;

    private void Awake()//poz initiala a pasarii pt launching
    {
        _initialPosition = transform.position;//pastram poz initiala
    }

    private void Update()
    {
        GetComponent<LineRenderer>().SetPosition(0, transform.position);
        GetComponent<LineRenderer>().SetPosition(1,_initialPosition);

        if ( _birdWasLaunched &&
            GetComponent<Rigidbody2D>().velocity.magnitude <= 0.1 )
        {
            _timeSittingAround += Time.deltaTime;//deltatime= cat dureaza
un frame in secunde

        }
        if (transform.position.y > 15 ||
            transform.position.y < -10 ||
            transform.position.x < -15 ||
            transform.position.x > 40 ||
            transform.position.z > 10 )
    }
}
```

```

        _timeSittingAround > 3 )
    {
        string currentSceneName = SceneManager.GetActiveScene().name;
        SceneManager.LoadScene(currentSceneName);
    }

}

private void OnMouseDown()///cat "tinem" pasarea ,este rosie
{
    GetComponent<SpriteRenderer>().color = Color.red;

    GetComponent<LineRenderer>().enabled = true;
}

private void OnMouseUp()//lansam pasarea
{
    GetComponent<SpriteRenderer>().color = Color.white;

    Vector2 directionToInitialPosition = -_initialPosition -
transform.position;

    GetComponent<Rigidbody2D>().AddForce(directionToInitialPosition *
_launchPower);
    GetComponent<Rigidbody2D>().gravityScale = 1;
    _birdWasLaunched = true;
    GetComponent<LineRenderer>().enabled = false;
}

private void OnMouseDrag()
{
    Vector3 newPosition =
Camera.main.ScreenToWorldPoint(Input.mousePosition);
    //luam pozitia cursorului in fct de camera
    transform.position =new Vector3(newPosition.x, newPosition.y);

}

```

2. Script-ul Enemy.cs pentru obiectul **Inamic**

```
using UnityEngine;
```

```

public class Enemy : MonoBehaviour
{
    [SerializeField] private GameObject _cloudParticlePrefab;///
sunt particulele de nori

    private void OnCollisionEnter2D(Collision2D collision)
    {
        Bird bird = collision.collider.GetComponent<Bird>() ;
        if (bird != null) ///daca dam de o pasare ( adica ce primim
e != NULL)
        {
            Instantiate(_cloudParticlePrefab, transform.position,
Quaternion.identity);///incepe particulele de nori
            ///Instantiate( obiect, pozitia enemy, rotatia
particulelor)
            Destroy(gameObject);///se distrugе dusmanul
            return;//gata!!
        }

        Enemy enemy =
collision.collider.GetComponent<Enemy>();//analog ca la bird
        if(enemy != null)
        {
            return;//atat!! gata!!
        }

        if(collision.contacts[0].normal.y < -0.5)///daca a fost
atins ON top
            ///acel contact
        normal ne da unghiul unde a fct contact
        {
            Instantiate(_cloudParticlePrefab, transform.position,
Quaternion.identity);
            Destroy(gameObject);
        }
    }
}

```

3. Script-ul **LevelController** pentru obiectul **care controleaza nivelurile**

```

using UnityEngine;
using UnityEngine.SceneManagement;
public class LevelController : MonoBehaviour
{
    private static int _nextLevelIndex = 1;

    private Enemy[] _enemies;

```

```

private void OnEnable()
{
    _enemies = FindObjectsOfType<Enemy>();
}

void Update()
{
    foreach (Enemy enemy in _enemies)/// pt fiecare dusman din
vectorul de dusmani
    {
        if (enemy != null)
            return;
    }

    Debug.Log("You killed all enemies");

    _nextLevelIndex++; //mergem la urm nivel
    string nextLevelName = "Level" + _nextLevelIndex;
    SceneManager.LoadScene(nextLevelName);
}
}

```

★ Anexa 2: Varianta finala a aplicatiei

