Colegiul Național „Mihai Viteazul” Turda
Matematică-informatică, intensiv informatică
Atestat informatică

LabPaint

Autor
Oltean Dragoş-Paul

Profesori îndrumători
prof. Miron Florin
prof. Pop Daniela

2020

Cuprins

Descriere generală

Aplicaţia **LabPaint** este un joc 2D în care jucătorul controlează un om de ştiinţă. Scopul jocului este să omori slime-urile care au fost create după un experiment eşuat, colorând podeaua în culorile corespunzătoare slime-urilor. Acestea apar în locaţii diferite. La început vor veni slime-uri de culoare verde şi galbenă, dar în momentul în care unul dintre ele va face contact cu o culoare diferită, se va transforma într-un slime roşu, mult mai puternic. Jucătorul primeşte un anumit punctaj în momentul în care omoara un slime. Jocul se termină când jucătorul moare, avand posibilitatea să înscrie scorul în clasament.

Motivul alegerii temei

Am ales această temă, deoarece de mic am fost pasionat de jocuri, având mereu în minte sa imi creez propriul joc. Astfel, atestatul mi-a oferit această şansă.

Resurse Hardware şi Software

- ## Hardware

  Procesor 1.6GHz sau mai rapid
  256 MB RAM
  62 MB spaţiu pe disk

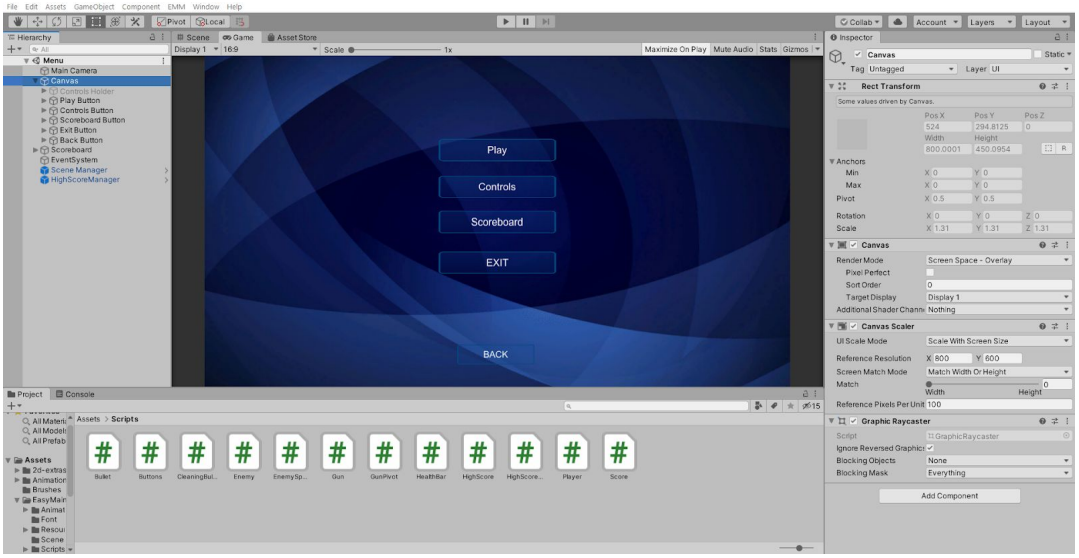
- ## Software

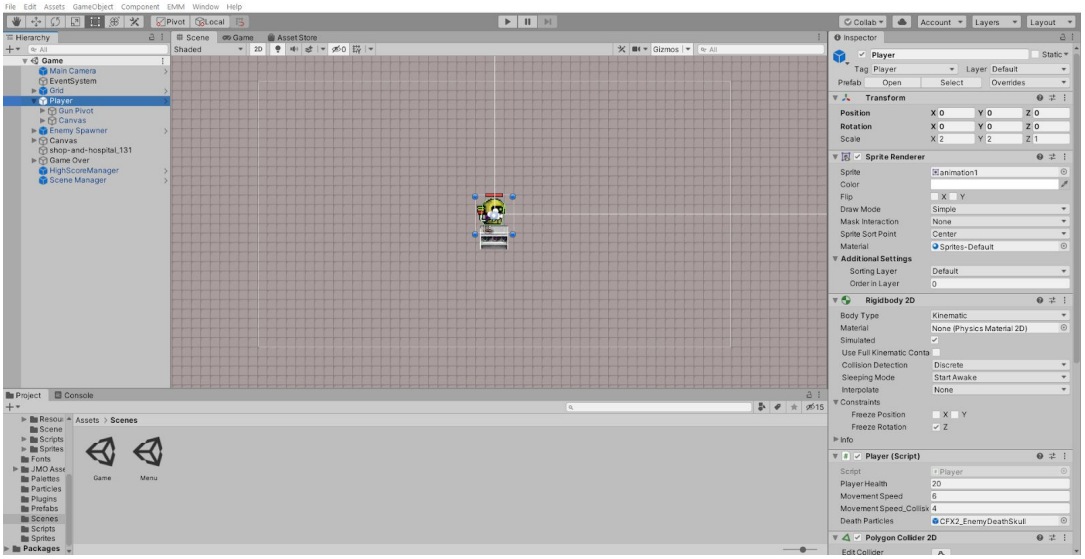
Windows 8/8.1(x86 sau x64)
Windows 10(x86 sau x64)

Realizarea aplicaţiei

Construcţia scenelor

Aplicaţia conţine 2 scene: scena meniului şi scena jocului.





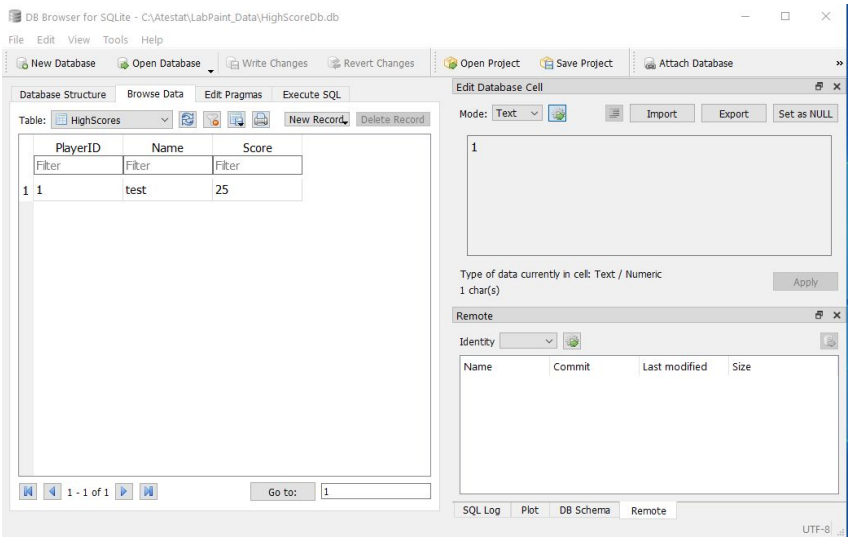
Scena cu jocul conţine 2 tipuri de obiecte: obiecte de joc (jucătorul, inamicii) şi obiecte de interfaţă(scor, bara de viaţă).

În partea stânga sus se află ierarhia jocului. Acolo se aşază obiectele în scenă şi sunt aplicate diferite relaţii între ele, de tipul părinte-copil.

În partea dreaptă este inspectorul de obiecte. Aici pot fi schimbate proprietăţile obiectelor aflate în scenă. Caracterul pe care jucătorul îl controlează are o proprietate de tip "Rigidbody 2D" care permite sa i se atribuie diferite fizici. Acesta are şi o proprietate numită "Polygon Collider 2D" care permite sa existe coliziune între caracter şi inamici. De asemenea, obiectul are şi un script numit "Player" unde sunt realizate toate mecanicile legate de acesta.

Baza de date

Baza de date a fost realizata utilizand SQLite. SQLite este o bibliotecă C care implementează un motor de baze de date SQL încapsulat, oferă posibilitatea de a-l introduce în diverse sisteme şi necesită zero configurare. Baza de date folosită în aplicaţii contine un singur tabel numit Highscores care reţine: "Player ID" (câmp de tip INTEGER, UNIC), "Name" (câmp de tip TEXT) şi "Score" (câmp de tip INTEGER). Pentru a crea şi vizualiza baza de date a fost folosit 'DB Browser for SQLite'.



Secvenţe reprezentative de cod

Secvenţa urmatoare realizează mişcarea caracterului. Funcţia Input.getAxisRaw returnează -1, 0, sau 1, în funcţie de tasta de mişcare pe care o apasam. Funcţia Move Player(), apelată în funcţia FixedUpdate(), realizează mişcarea efectiva în funcţie de rezultatul obtinut. De asemenea, avem şi proprietatea eulerAngles care realizeaza intoarcerea caracterului în funcţie de direcţia de mişcare.

```
1 reference
private void PlayerMovementSetup()
{
    movement.x = Input.GetAxisRaw("Horizontal");
    movement.y = Input.GetAxisRaw("Vertical");

    if (movement.x == 1)
        transform.eulerAngles = new Vector3(0, 0, 0);
    if (movement.x == -1)
        transform.eulerAngles = new Vector3(0, 180, 0);
}

1 reference
private void MovePlayer()
{
    if (!isSlowed)
        currentMovementSpeed = movementSpeed;
    else
        currentMovementSpeed = movementSpeed_Collision_Slime;

    rb.MovePosition(rb.transform.position + movement * currentMovementSpeed * Time.fixedDeltaTime);

    //Keeping the player within screen boundaries
    Vector3 screenBounds = Camera.main.ScreenToWorldPoint(new Vector3(Screen.width, Screen.height, 0));
    float objectWidth = transform.GetComponent<SpriteRenderer>().bounds.extents.x;
    float objectHeight = transform.GetComponent<SpriteRenderer>().bounds.extents.y;

    Vector3 pos = transform.position;
    pos.x = Mathf.Clamp(pos.x, screenBounds.x * -1 + objectWidth, screenBounds.x - objectWidth);
    pos.y = Mathf.Clamp(pos.y, screenBounds.y * -1 + objectHeight, screenBounds.y - objectHeight);
    transform.position = pos;
}
```

Funcţia Paint Selection() realizează selecţia de vopsea în funcţie de tasta apasata de utilizator. Mecanică de ‘shooting’ este realizata de funcţia Player Shooting(). De asemenea, a fost creat şi un ‘Coroutine’ care ajuta la ‘shooting-ul’ continuu.

```
private void PaintSelection()
{
    if (Input.GetButtonDown("Paint1"))
        selectedPaintIndex = 0;
    if (Input.GetButtonDown("Paint2"))
        selectedPaintIndex = 1;
    if (Input.GetButtonDown("Paint3"))
        selectedPaintIndex = 2;
}

1 reference
private void PlayerShooting()
{
    if (Input.GetButtonDown("Fire1") && cleaningActive == false)
    {
        firingCoroutine = StartCoroutine(FireContinously());
        firingActive = true;
    }
    if (Input.GetButtonUp("Fire1"))
    {
        StopCoroutine(firingCoroutine);
        firingActive = false;
    }

    if (Input.GetButtonDown("Fire2") && firingActive == false)
    {
        cleaningCoroutine = StartCoroutine(CleaningContinously());
        cleaningActive = true;
    }
    if (Input.GetButtonUp("Fire2"))
    {
        StopCoroutine(cleaningCoroutine);
        cleaningActive = false;
    }
```

Funcţia OnTriggerEnter2D realizeaza coliziunea dintre vopsea şi podea pentru a o colora în culoarea corespunzătoare. Funcţia MoveBullet() ajuta vopseaua sa se deplaseze la poziţia cursorului, reţinută în variabila targetPos.

```
private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.gameObject.tag.Equals("Tile"))
    {
        collision.gameObject.GetComponent<SpriteRenderer>().color = GetComponent<SpriteRenderer>().color;
        Destroy(gameObject);
    }
}

1 reference
private void MoveBullet()
{
    transform.position = Vector3.MoveTowards(transform.position, targetPos, bulletSpeed * Time.deltaTime);

    if (transform.position == targetPos)
        gameObject.layer = LayerMask.NameToLayer("Tiles");
}
```

Funcţia EnemyMovement() realizează mişcarea inamicilor. Aceştia sunt programaţi să se îndrepte spre jucător prin funcţia MoveTowards().

Verificarea culorii se face în funcţia checkColorMatch() unde se compară culoarea podelei cu cea a inamicului. De asemenea, în momentul în care culorile nu se potrivesc, inamicul devine mai puternic, mecanică realizata în funcţia checkEmpoweredNeeded().

```
private void EnemyMovement()
{
    playerPos = GameObject.FindGameObjectWithTag("Player").GetComponent<Transform>().position;

    transform.position = Vector3.MoveTowards(transform.position, playerPos, enemyMoveSpeed * Time.deltaTime);

    //Handle Enemy Direction
    if (transform.position.x < playerPos.x)
        transform.eulerAngles = new Vector3(0, 180, 0);
    if (transform.position.x > playerPos.x)
        transform.eulerAngles = new Vector3(0, 0, 0);
}

1 reference
private bool checkColorMatch(Collider2D tileCollider)
{
    Color enemy = GetComponent<SpriteRenderer>().color;
    Color tile = tileCollider.gameObject.GetComponent<SpriteRenderer>().color;

    Color32 enemy32 = new Color32((byte)(enemy.r * 255), (byte)(enemy.g * 255), (byte)(enemy.b * 255), (byte)(enemy.a * 255));
    Color32 tile32 = new Color32((byte) (tile.r*255), (byte)(tile.g * 255), (byte)(tile.b * 255), (byte)(tile.a * 255));

    return enemy32.Equals(tile32);
}

1 reference
private bool checkEmpoweredNeeded(Collider2D tileCollider)
{
    Color tile = tileCollider.gameObject.GetComponent<SpriteRenderer>().color;
    Color32 tile32 = new Color32((byte)(tile.r * 255), (byte)(tile.g * 255), (byte)(tile.b * 255), (byte)(tile.a * 255));

    return !tile32.Equals(tileColor);
}
```

Partea de baze de date este gestionată cu ajutorul următoarelor funcţii. Acestea au rolul de a introduce date in tabela, de a prelua toate randurile din tabela şi de a completa primele 10 randuri de elemente dintr-un obiect de tip UI numit GridLayoutGroup.

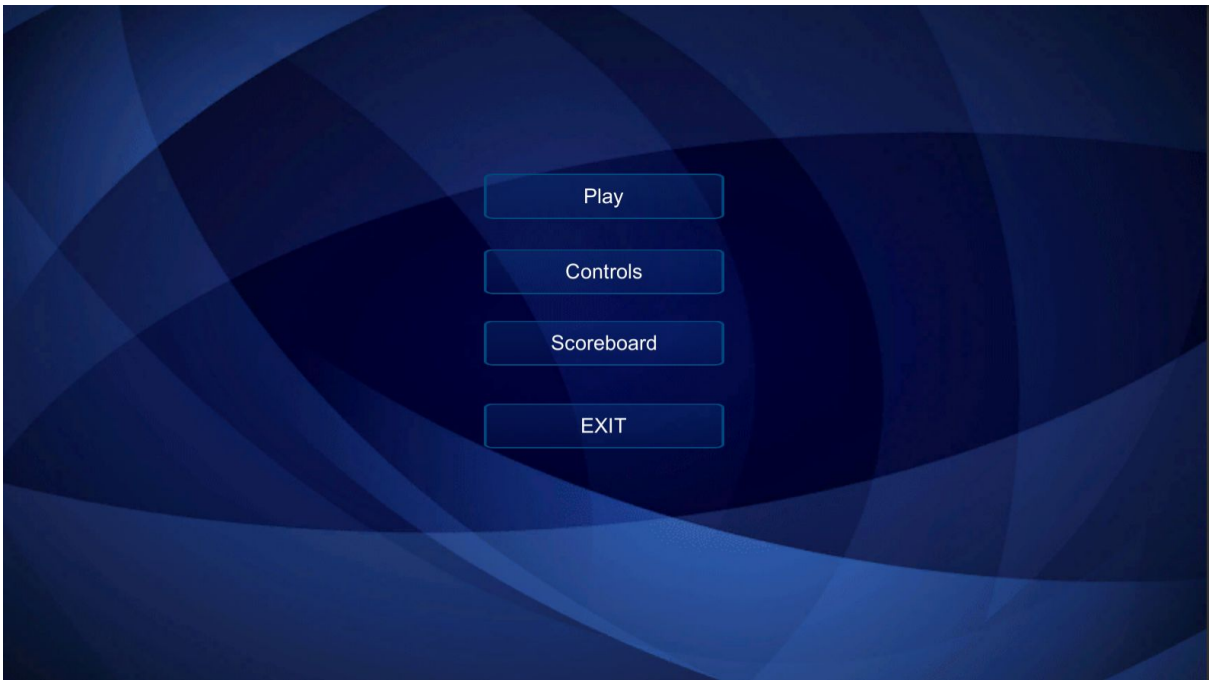
```
private void InsertScore(string name, int score)
{
    using (IDbConnection dbConnection = new SqliteConnection(connectionString))
    {
        dbConnection.Open();

        using (IDbCommand dbCmd = dbConnection.CreateCommand())
        {
            string sqlQuery = String.Format("INSERT INTO HighScores(Name, Score) VALUES (\"{0}\", \"{1}\")", name, score);

            dbCmd.CommandText = sqlQuery;
            dbCmd.ExecuteScalar();
            dbConnection.Close();
        }
    }
}
```

```
private void GetScores()
{
    highScores.Clear();

    using (IDbConnection dbConnection = new SqliteConnection(connectionString))
    {
        dbConnection.Open();

        using (IDbCommand dbCmd = dbConnection.CreateCommand())
        {
            string sqlQuery = "SELECT * FROM HighScores";

            dbCmd.CommandText = sqlQuery;

            using (IDataReader reader = dbCmd.ExecuteReader())
            {
                while (reader.Read())
                {
                    highScores.Add(new HighScore(reader.GetInt32(0), reader.GetInt32(2), reader.GetString(1)));
                }

                dbConnection.Close();
                reader.Close();
            }
        }
    }

    highScores.Sort();
}
```

```
private void ShowScores()
{
    GetScores();
    for (int i = 0; i < 10; i++)
    {
        if (i < highScores.Count)
        {
            GameObject tmpObject = Instantiate(scorePrefab);

            HighScore tmpScore = highScores[i];

            tmpObject.GetComponent<HighScoreScript>().SetScore(tmpScore.Name, tmpScore.Score.ToString(), "#" + (i + 1).ToString());

            tmpObject.transform.SetParent(scoreParent);

            tmpObject.GetComponent<RectTransform>().localScale = new Vector3(1, 1, 1);
        }
    }
}
```

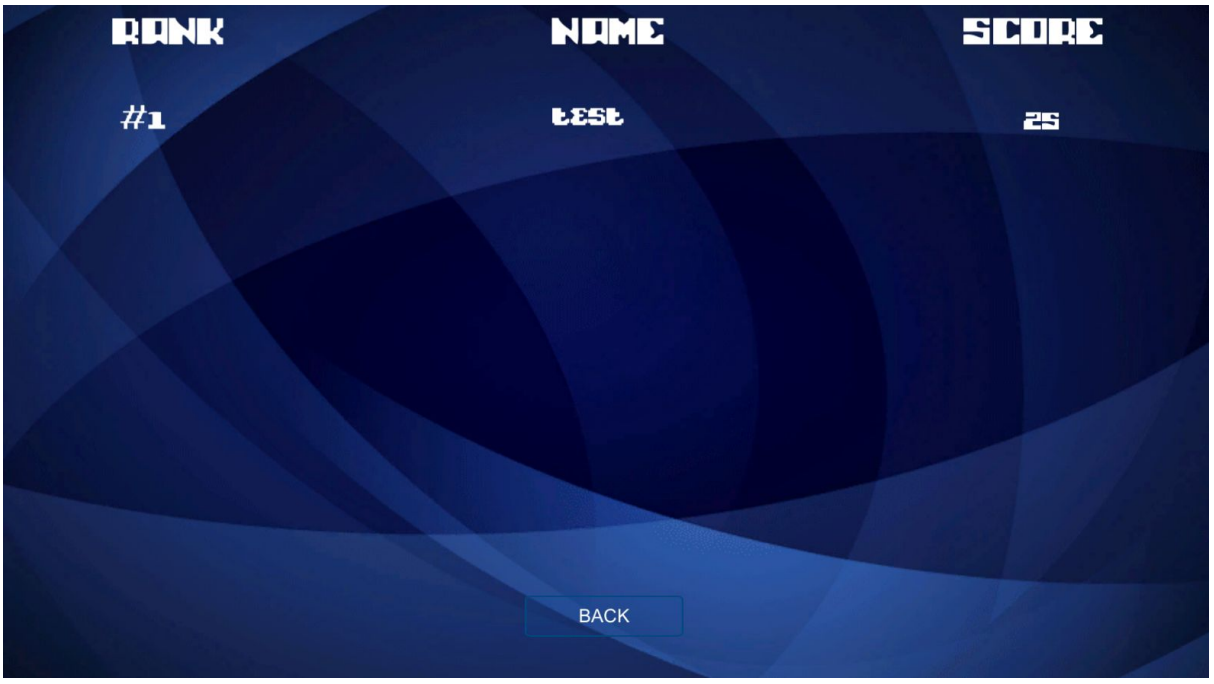
Manual de utilizare

La deschiderea jocului va apărea meniul, de unde se poate alege diferite opţiuni.
Play - Începerea jocului
Exit - Inchiderea aplicatiei



Controls - Se deschide meniul cu controale

Scoreboard - Se deschide tabela de scor



Ecranul de Game Over

Jucătorul va avea posibilitatea de a adăuga scorul obţinut in clasament, sa restarteze jocul, sau sa revina la meniul principal.

Posibile îmbunătăţiri

- adaugarea unor noi tipuri de inamici
- adaugarea de nivele noi
- adaugarea muzicii, efectelor
- elemente de UI îmbunătăţite
- grafică îmbunătăţită

Bibliografie

https://www.youtube.com/user/Brackeys
https://www.youtube.com/channel/UC9Z1XWw1kmnvOOFsj6Bzy2g
https://www.youtube.com/channel/UCFK6NCbuCIVzA6Yj1G_ZqCg
https://answers.unity.com/index.html
https://stackoverflow.com/
https://www.udemy.com/