

Liceul de Informatică "Tiberiu Popoviciu", Cluj-Napoca

PROIECT PENTRU OBTINEREA ATESTĂRII PROFESIONALE ÎN  
INFORMATICĂ

Titlul lucrării :

# Incorruptible

Cluj-Napoca,  
Mai 2021

Suciu Ioan-Mihail  
Clasa XII C  
Coordonatori: prof. Andreea Racolța

# Cuprins

1. Prezentare generală
2. Resurse hardware și software necesare
3. Realizarea aplicației
  - a. Prezentarea jocului
  - b. Proiectarea tehnica
4. Extinderi posibile ale aplicației

## 1. Prezentarea generală

Aplicația este un joc de tip „platformer” în care jucătorul controlează un caracter și trebuie să parcurgă o temniță , pe parcurs acesta va putea colecta diamante.

Jocul are un sistem de logare și unul de autentificare, astfel acesta reține numărul de diamante colectate de la crearea contului și numărul maxim de diamante colectate pentru fiecare nivel.

Aplicația poate să :

- Adauge noi conturi
- Logheze utilizatorii la conturile lor
- Delogheze utilizatorul logat curent
- Rețină pentru fiecare utilizator cele mai mari punctaje pentru fiecare nivel și cel mai mare punctaj de la crearea contului


## 2. Resurse hardware și software necesare

Resurse necesare :

- Procesor minim 2 GHz
- Memorie minim 2 GB RAM
- Memorie minima pe Hard Disk 1 GB
- Microsoft Windows 7
- Intel HD Graphics 2500

## 3. Realizarea aplicației

- Prezentarea jocului

Incorruptible te pune în ipostaza unui erou care navighează printr-o temniță. Temnița este alcătuită din mai multe nivele fiecare având scopul de a face pe jucător să găsească metode ingenioase de a le parcurge. Te poți mișca stânga , dreapta si să sari pentru a traversa fiecare nivel si pentru a evita să cazi. Pe parcursul fiecărui nivel jucătorul poate să colecteze diamante  ,acestea fiind adunate la sfârșitul fiecărui nivel , formând un scor. Fiecare jucător poate să își vadă punctajul maxim pentru fiecare nivel și numărul de diamante adunate de la începerea jocului.

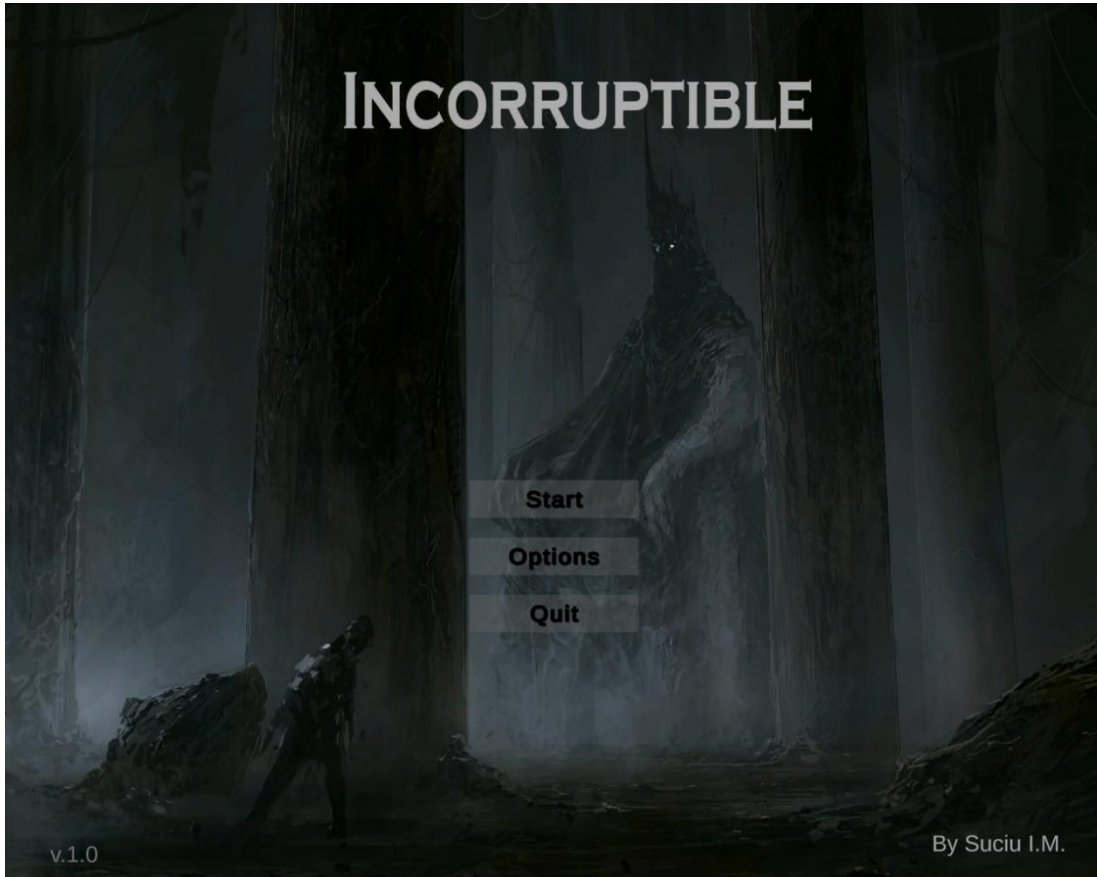
- Proiectarea tehnică

Pentru proiectarea jocului am folosit Unity. Jocul este structurat pe mai multe scene care comunică între ele. În același timp există Scripturi (fișiere cod) care se ocupă cu logica din spatele jocului.

Scenele sunt :

## 1. Scena de meniu.

Scena aceasta este responsabilă pentru accesarea scenei de logare/creare cont cât și pentru a accesa scena de setări.

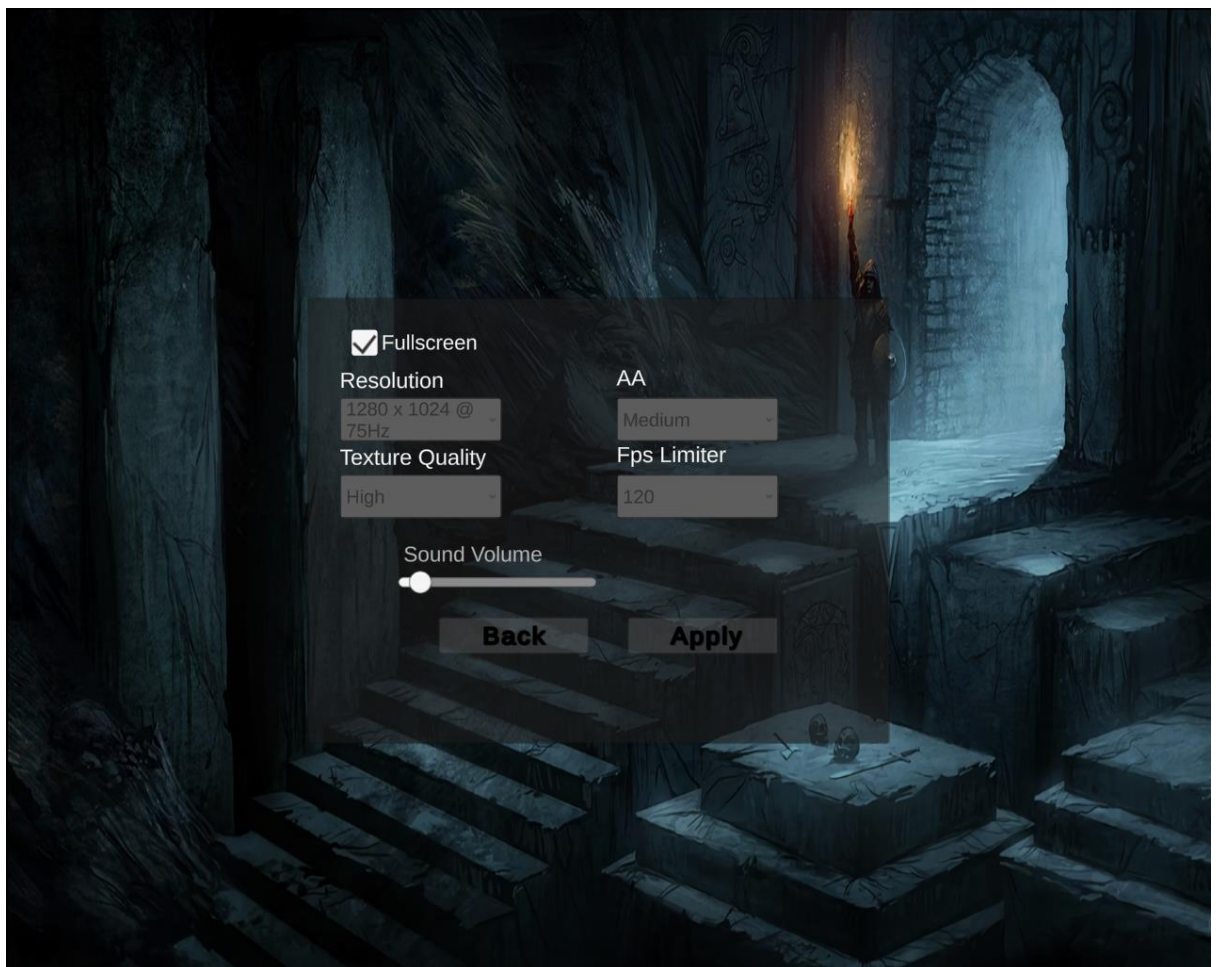


Aici se poate vedea codul din spatele fiecărui buton.

```
public void PlayGame()
{
    SceneManager.LoadScene(6);
}
0 references
public void OptionsGame()
{
    SceneManager.LoadScene(1);
}
0 references
public void QuitGame()
{
    Application.Quit();
}
```

## 2. Scena de Opțiuni

Scena aceasta este responsabilă cu schimbare setărilor grafici (Ecran complet , Rezoluția ,Calitatea Texturilor ,Filtru Anisotropic, Intensitatea Volumului , Numărul de cadre pe secundă la care va rula programul)



Aici se poate vedea codul responsabil pentru fiecare componentă a meniului.

```
public void OnFullscreenToggle()
{
    gameSettings.fullscreen = Screen.fullScreen = fullscreenToggle.isOn;
}

1 reference
public void OnResolutionChange()
{
    Screen.SetResolution(resolutions[resolutionDropdown.value].width, resolutions[resolutionDropdown.value].height, Screen.fullScreen);
    gameSettings.resolutionIndex = resolutionDropdown.value;
}

1 reference
public void OnTextureQualityChange()
{
    QualitySettings.masterTextureLimit = gameSettings.textureQuality = textureQualityDropdown.value;
}

1 reference
public void OnAntialiasingChange()
{
    if (antialiasingDropdown.value == 0)
    {
        QualitySettings.antiAliasing = 0;
        gameSettings.antialiasing = 0;
    }
    else if (antialiasingDropdown.value == 1)
    {
        QualitySettings.antiAliasing = 2;
        gameSettings.antialiasing = 1;
    }
    else if (antialiasingDropdown.value == 2)
    {
        QualitySettings.antiAliasing = 4;
        gameSettings.antialiasing = 2;
    }
}

public void OnFpsDropdown()
{
    QualitySettings.vSyncCount = 0;
    if (fpsDropdown.value == 0)
    {
        Application.targetFrameRate = 30;
        gameSettings.fps = fpsDropdown.value;
    }
    else if (fpsDropdown.value == 1)
    {
        Application.targetFrameRate = 60;
        gameSettings.fps = fpsDropdown.value;
    }
    else if (fpsDropdown.value == 2)
    {
        Application.targetFrameRate = 120;
        gameSettings.fps = fpsDropdown.value;
    }
}

1 reference
public void OnMusicVolumeChange()
{
    ObjectMusic = GameObject.FindWithTag("music");

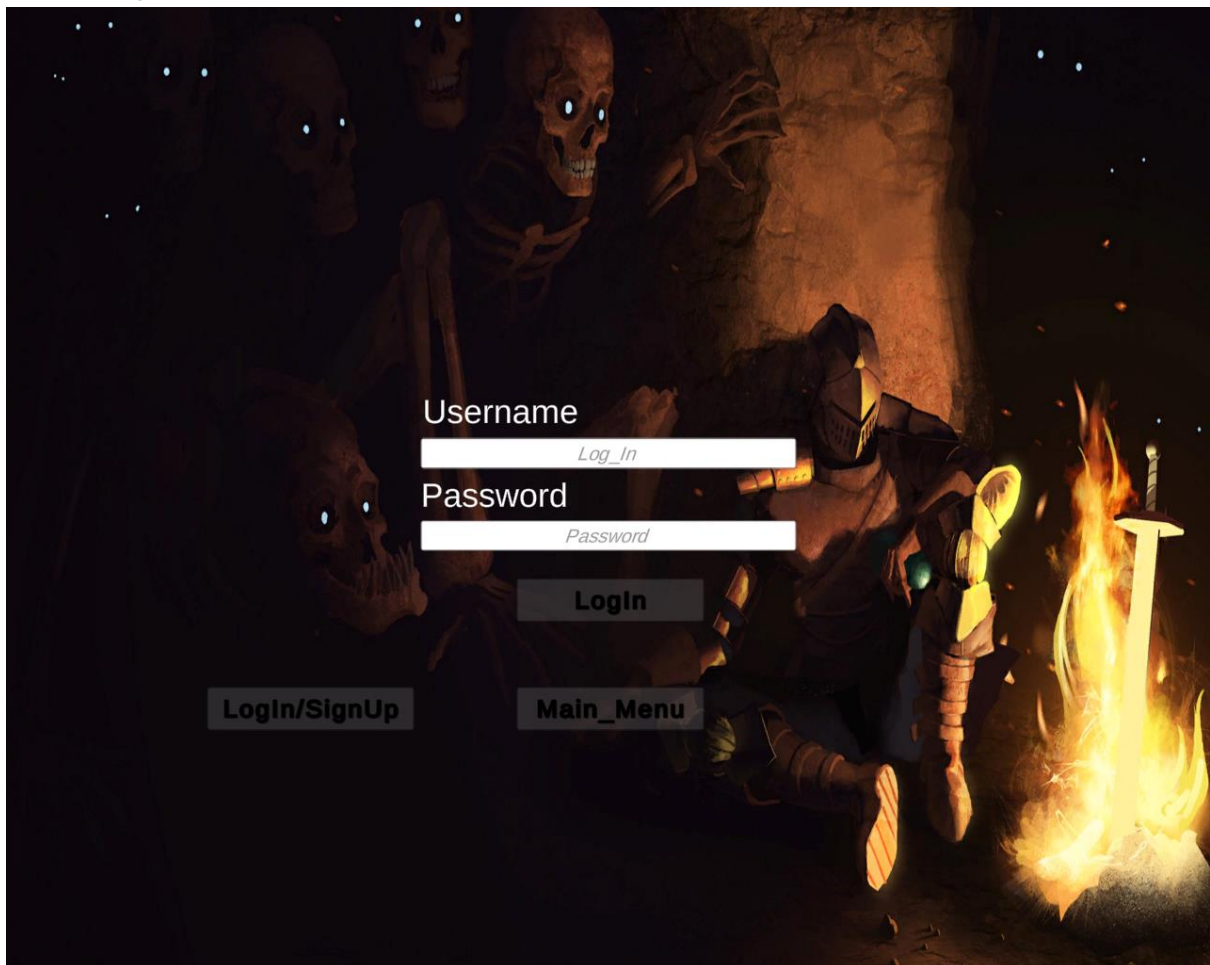
    musicSource = ObjectMusic.GetComponent<AudioSource>();

    musicSource.volume = gameSettings.musicVolume = musicVolumeSlider.value;

    string jsonData = JsonUtility.ToJson(gameSettings, true);
    File.WriteAllText(Application.persistentDataPath + "/gamesettings.json", jsonData);
}
```

### 3. Scena de Login

Scena aceasta este responsabilă cu logarea utilizatorului în joc, pentru a fi posibilă salvarea scorului. Utilizatorul trebuie să introducă username-ul și parola contului la care dorește să se logheze.



Aici se poate vedea codul din spatele meniului de logare.

```
public void Login_Button()
{
    if (Username != "" && Password != "")
    {
        if (verif(Username) == false)
            warning.text = "Username can only contain letters or numbers";
        else
        {
            if (File.Exists(Application.persistentDataPath + "/" + Username + ".json") == true)
            {
                data = JsonUtility.FromJson<Saved_Data>(File.ReadAllText(Application.persistentDataPath + "/" + Username + ".json"));
                if (data.password != Password)
                    warning.text = "Invalid password";
                else if (data.password == Password && data.username == Username)
                {
                    PlayerPrefs.SetString("User", Username);
                    SceneManager.LoadScene(4);
                }
            }
            else
            {
                warning.text = "Username doesn't exist.";
            }
        }
    }
    else
    {
        warning.text = "Please complete all fields";
    }
}

private bool verif(string x)
{
    for (int i = 0; i < x.Length; i++)
    {
        if (!((x[i] >= 'a' && x[i] <= 'z') || (x[i] >= 'A' && x[i] <= 'Z') || (x[i] >= '0' && x[i] <= '9'))))
            return false;
    }
    return true;
}

⊞ Unity Message | 0 references
private void Awake()
{
    warning.text = " ";
}

⊞ Unity Message | 0 references
private void Update()
{
    if (Input.GetKeyDown(KeyCode.Tab))
    {
        if (username.GetComponent<TMP_InputField>().isFocused)
        {
            password.GetComponent<TMP_InputField>().Select();
        }
        if (password.GetComponent<TMP_InputField>().isFocused)
        {
            username.GetComponent<TMP_InputField>().Select();
        }
    }

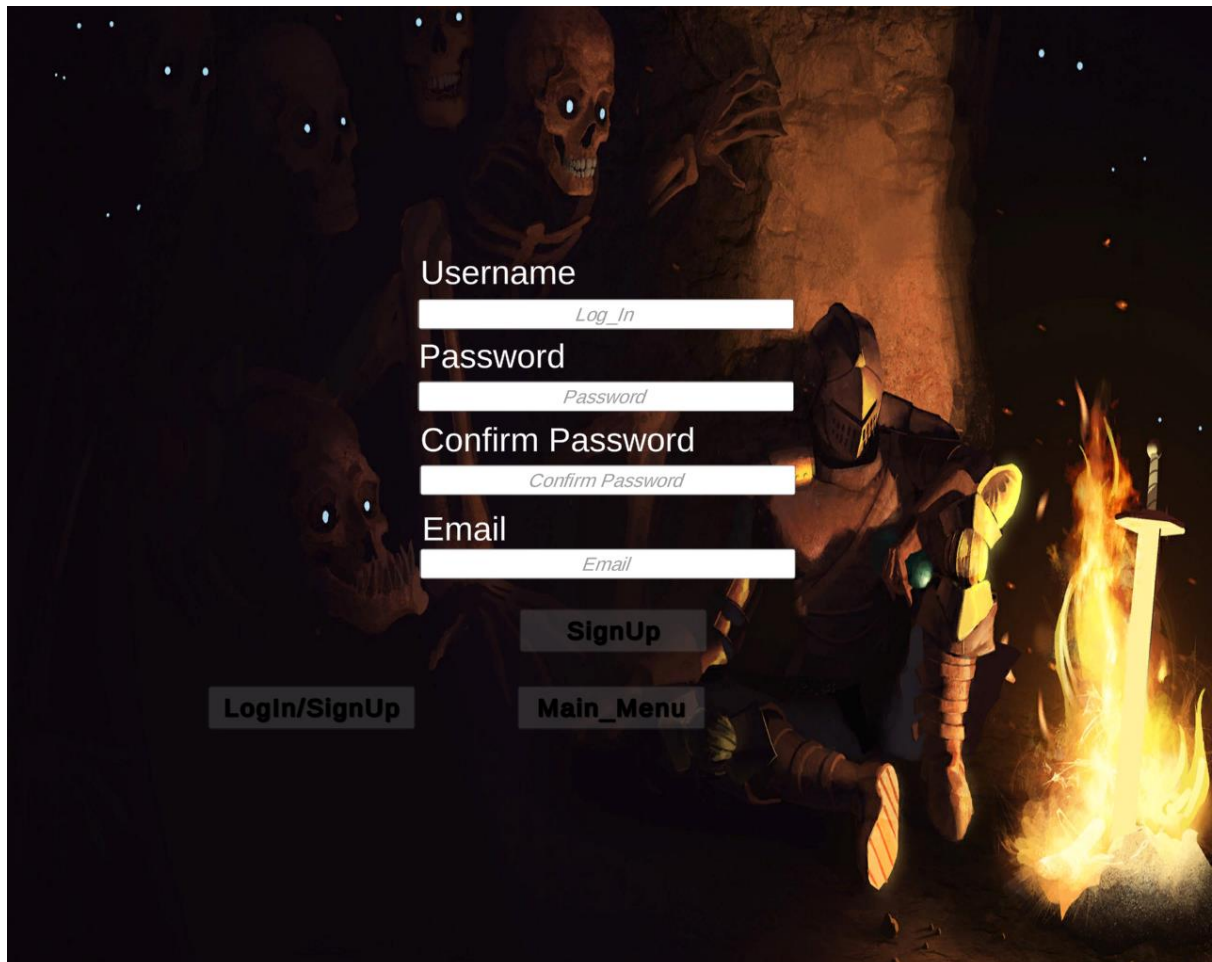
    Username = username.GetComponent<TMP_InputField>().text;
    Password = password.GetComponent<TMP_InputField>().text;
}
}
```



#### 4. Scena de Creare Cont

Aici utilizatorul poate crea un nou cont care este înregistrat. Utilizatorul trebuie să introducă un username, o parolă, apoi să confirme parola și un email.

După înregistrare utilizatorul va fi trimis la meniul pentru nivele și logat direct cu contul tocmai creat.



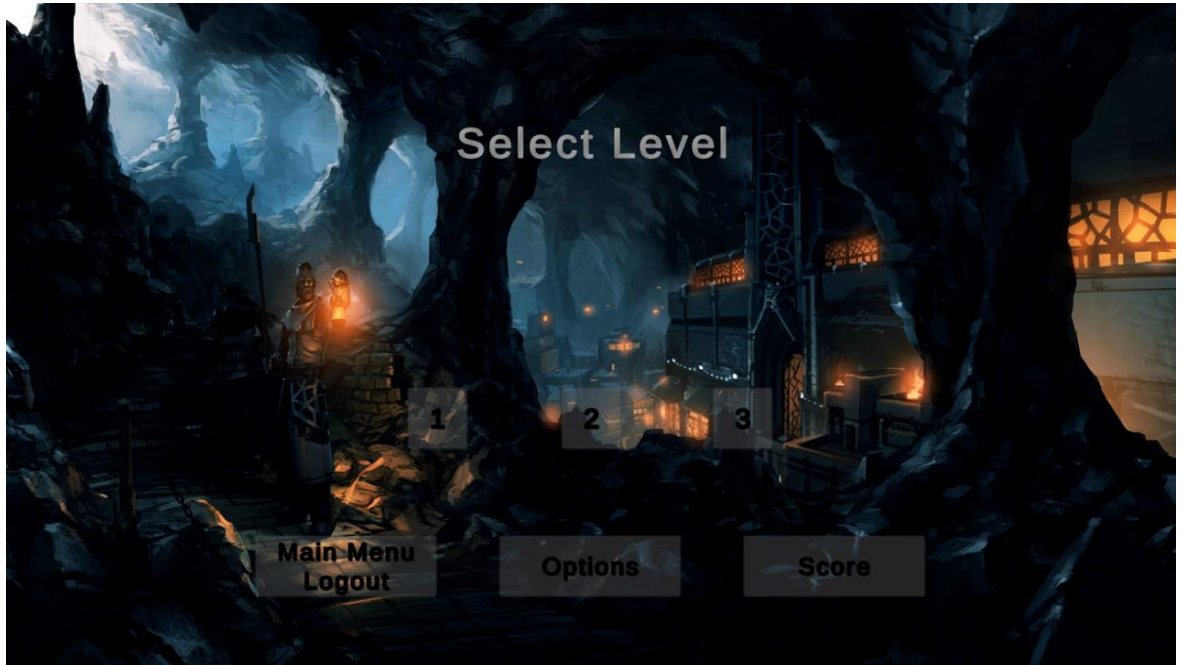
Aici putem vedea codul din spatele meniului de creare cont.

```
public void Register_Button()
{
    data = new Saved_Data();
    if (Username != "" && Password != "" && Confirmpassword != "" && Email != "")
    {
        bool Un = false;
        bool Em = false;
        bool Pw = false;
        bool ConfPw = false;

        if (verif(Username) == false)
            warning.text = "Username can only contain letters or numbers";
        else
        {
            if (File.Exists(Application.persistentDataPath + "/" + Username + ".json") == true)
            {
                warning.text = "Username taken.";
            }
            else
            {
                Un = true;
            }
            if (Email.Contains("@"))
            {
                if (Email.Contains("."))
                {
                    Em = true;
                }
                else
                {
                    warning.text = "Email is incorrect.";
                }
            }
            else
            {
                warning.text = "Email is incorrect.";
            }
            if (Password.Length > 5)
            {
                Pw = true;
            }
            else
            {
                warning.text = "Password must be atleast 6 characters long.";
            }
            if (Confirmpassword == Password)
            {
                ConfPw = true;
            }
            else
            {
                warning.text = "Passwords don't match.";
            }
            if (Un == true && Em == true && Pw == true && ConfPw == true)
            {
                data.username = Username;
                data.password = Password;
                data.email = Email;
                string jsonData = JsonUtility.ToJson(data, true);
                File.WriteAllText(Application.persistentDataPath + "/" + Username + ".json", jsonData);
                SceneManager.LoadScene(4);
                PlayerPrefs.SetString("User", Username);
            }
        }
    }
    else
    {
        warning.text = "Please complete all fields";
    }
}
```

## 5. Scena de selectare a nivelurilor

Scena aceasta este responsabilă cu selectarea : nivelurilor , opțiunilor și a scorului.

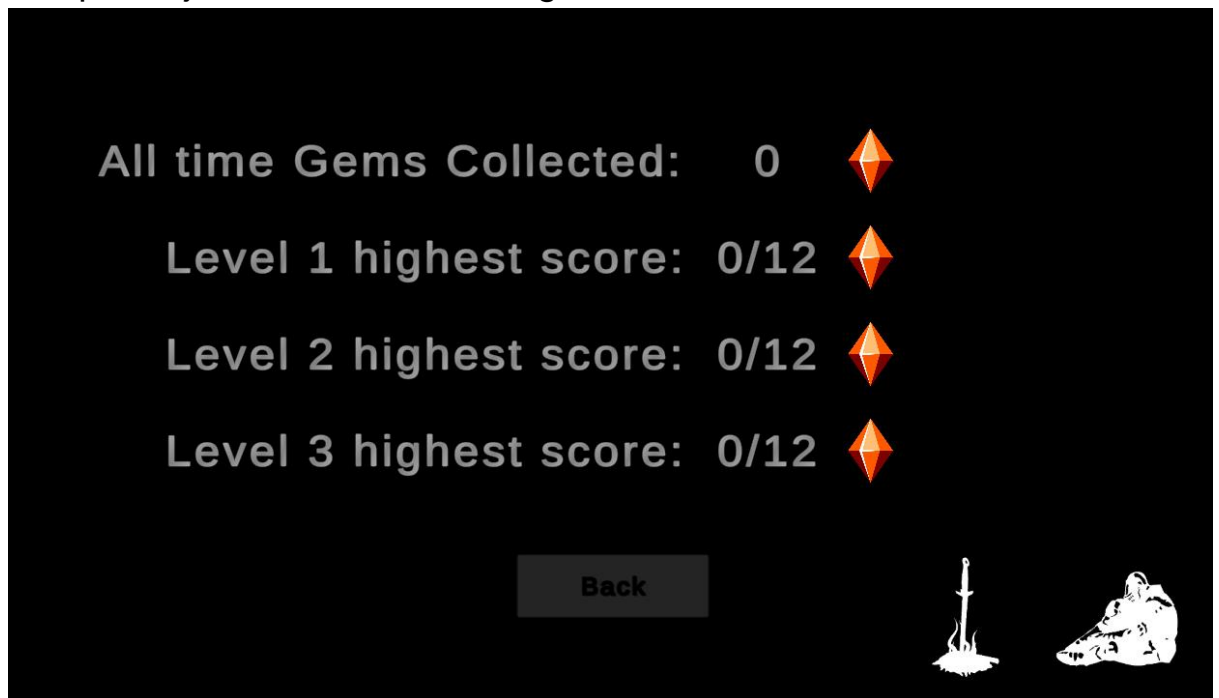


Aici putem vedea codul meniului.

```
public void Level_0()
{
    SceneManager.LoadScene(2);
}
0 references
public void Level_1()
{
    SceneManager.LoadScene(7);
}
0 references
public void Level_2()
{
    SceneManager.LoadScene(8);
}
0 references
public void Main_Menu()
{
    SceneManager.LoadScene(0);
}
0 references
public void Options()
{
    SceneManager.LoadScene(1);
}
0 references
public void Score()
{
    SceneManager.LoadScene(9);
}
```

## 6. Scena scorului

Scena aceasta este responsabilă cu afișarea punctajul maxim pentru fiecare nivel și numărul de diamante adunate de la începerea jocului utilizatorului logat.



Aici putem vedea codul din spatele meniului de scor.

```
public void Awake()
{
    data = JsonUtility.FromJson<Saved_Data>(File.ReadAllText(Application.persistentDataPath + "/" + PlayerPrefs.GetString("User") + ".json"));

    textGems0.text = data.total_score.ToString();
    textGems1.text = data.level1_max_score.ToString()+"/12";
    textGems2.text = data.level2_max_score.ToString()+"/12";
    textGems3.text = data.level3_max_score.ToString()+"/12";

    QualitySettings.vSyncCount = 0;
    if (File.Exists(Application.persistentDataPath + "/gamesettings.json") == true)
    {
        gameSettings = JsonUtility.FromJson<GameSettings>(File.ReadAllText(Application.persistentDataPath + "/gamesettings.json"));
        if (gameSettings.fps == 0)
            Application.targetFrameRate = 30;
        if (gameSettings.fps == 1)
            Application.targetFrameRate = 60;
        if (gameSettings.fps == 2)
            Application.targetFrameRate = 120;
    }
    else
        Application.targetFrameRate = 60;
}

0 references
public void Back_Button()
{
    SceneManager.LoadScene(4);
}
```

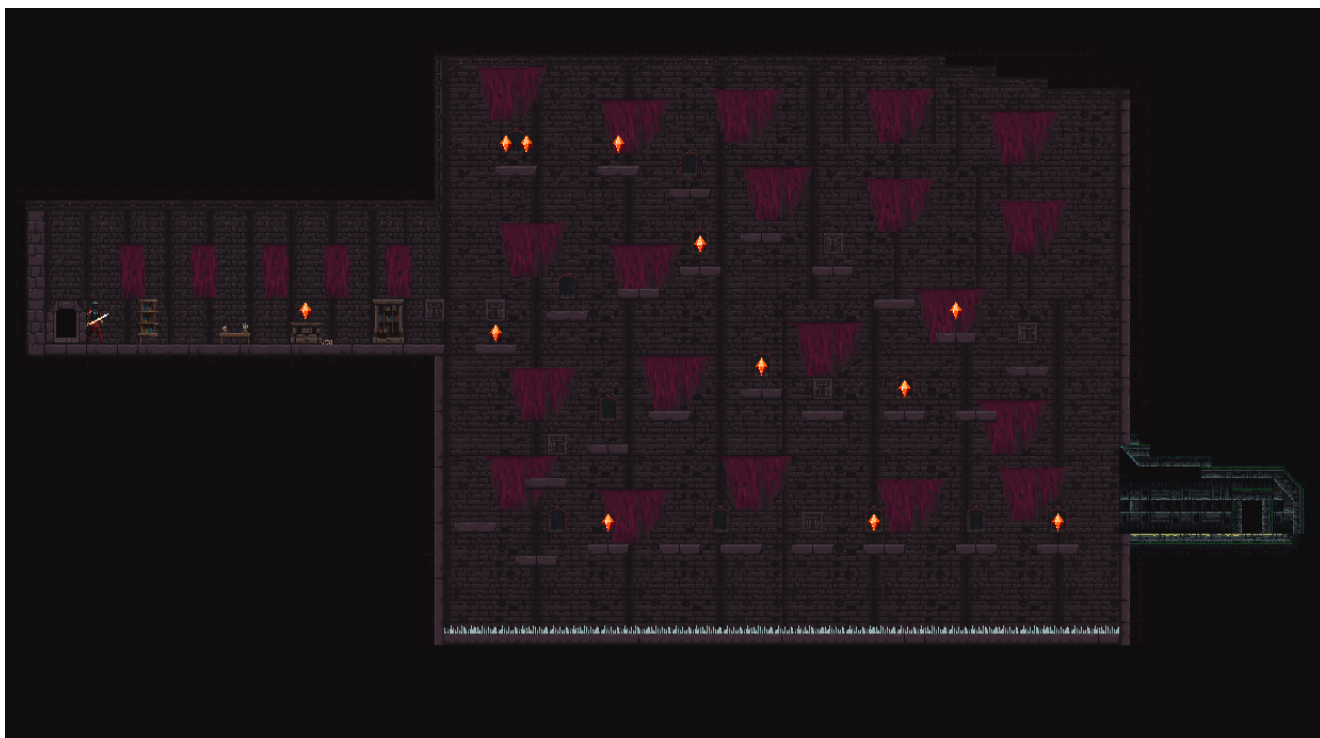
## 7. Scena Nivelului 1

Această scenă conține primul nivel al jocului.



## 8. Scena Nivelului 2

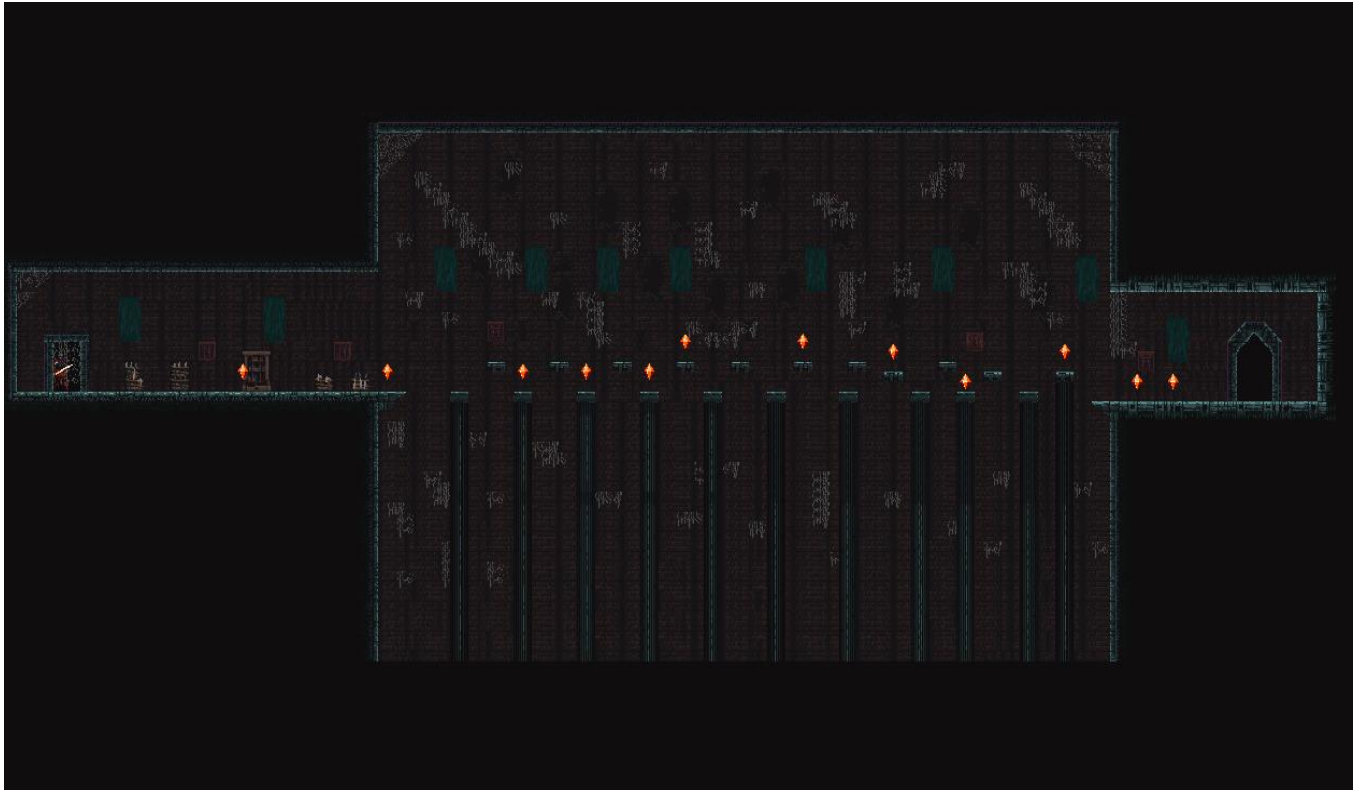
Această scenă conține nivelul 2 al jocului.





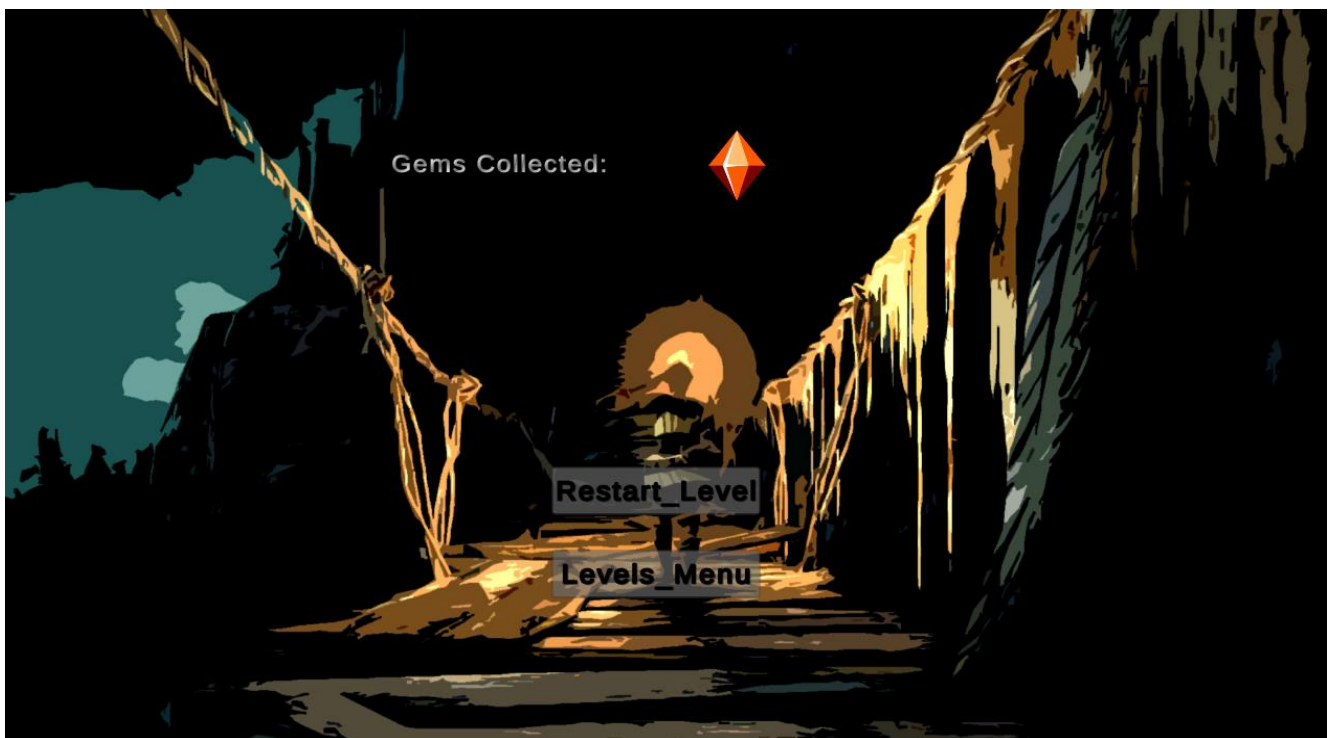
### 9. Scena Nivelului 3

Această scenă conține nivelul 3 al jocului.



### 10.Scena de sfârșit a nivelelor

Această scenă afișează diamantele colectate de pe parcursul nivelului cât si trimite jucătorul la meniul principal sau la următorul nivel.In același timp acesta adaugă scorul obținut la scorul total și calculează maximul dintre numărul de diamante colectate în acest nivel cu numărul de diamante maxim colectate ale utilizatorului curent pentru nivelul parcurs.



Aici putem vedea codul din spatele acestui meniu.

```
void save()
{
    data = JsonUtility.FromJson<Stored_data>(File.ReadAllText(Application.persistentDataPath + "/stored_data.json"));
    gemsnumber = data.gems;
    data2 = JsonUtility.FromJson<Saved_Data>(File.ReadAllText(Application.persistentDataPath + "/" + PlayerPrefs.GetString("User") + ".json"));
    if (PlayerPrefs.GetInt("LastLevel") == 2)
    {
        if (gemsnumber > data2.level1_max_score)
            data2.level1_max_score = gemsnumber;
    }
    else if (PlayerPrefs.GetInt("LastLevel") == 7)
    {
        if (gemsnumber > data2.level2_max_score)
            data2.level2_max_score = gemsnumber;
    }
    else if (PlayerPrefs.GetInt("LastLevel") == 8)
    {
        if (gemsnumber > data2.level3_max_score)
            data2.level3_max_score = gemsnumber;
    }
    data2.total_score += gemsnumber;
    string jsonData = JsonUtility.ToJson(data2, true);
    File.WriteAllText(Application.persistentDataPath + "/" + PlayerPrefs.GetString("User") + ".json", jsonData);
}
0 references
public void Next_Level_Button()
{
    save();

    if (PlayerPrefs.GetInt("LastLevel") == 2)
    {
        SceneManager.LoadScene(7);
    }
    else if (PlayerPrefs.GetInt("LastLevel") == 7)
    {
        SceneManager.LoadScene(8);
    }
}
public void Restart_Level()
{
    save();
    SceneManager.LoadScene(PlayerPrefs.GetInt("LastLevel"));
}
0 references
public void Main_Menu()
{
    save();
    SceneManager.LoadScene(4);
}
```

## 11.Scena de Respawn

Această scenă apare doar atunci când caracterul cade de pe platforme.



```

public GameSettings gameSettings;
0 references
public void Awake()
{
    QualitySettings.vSyncCount = 0;
    if (File.Exists(Application.persistentDataPath + "/gamesettings.json") == true)
    {
        gameSettings = JsonUtility.FromJson<GameSettings>(File.ReadAllText(Application.persistentDataPath + "/gamesettings.json"));
        if (gameSettings.fps == 0)
            Application.targetFrameRate = 30;
        if (gameSettings.fps == 1)
            Application.targetFrameRate = 60;
        if (gameSettings.fps == 2)
            Application.targetFrameRate = 120;
    }
    else
        Application.targetFrameRate = 60;
}

0 references
public void Restart_Level()
{
    SceneManager.LoadScene(PlayerPrefs.GetInt("LastLevel"));
}

0 references
public void Main_Menu()
{
    SceneManager.LoadScene(0);
}

0 references
public void QuitGame()
{
    Application.Quit();
}

```

Codul responsabil cu mișcarea caracterului.

```

private void Update() {

    mx = Input.GetAxisRaw("Horizontal");

    if (Input.GetButtonDown("Jump") && IsGrounded())
    {
        Jump();
    }
    if(faceright==true && mx<0)
    {
        transform.localScale = new Vector3(-1, 1, 1);

        faceright = false;
    }
    else if(faceright==false && mx>0)
    {
        transform.localScale = new Vector3(1, 1, 1);
        faceright = true;
    }
    if (Input.GetKeyDown(KeyCode.LeftShift))
    {
        isruning = true;
    }
    if (Input.GetKeyUp(KeyCode.LeftShift))
    {
        isruning = false;
    }
    animator.SetFloat("xVelocity", Mathf.Abs(rb.velocity.x));
    animator.SetFloat("yVelocity", rb.velocity.y);
    animator.SetBool("Jump", !IsGrounded());
}

```



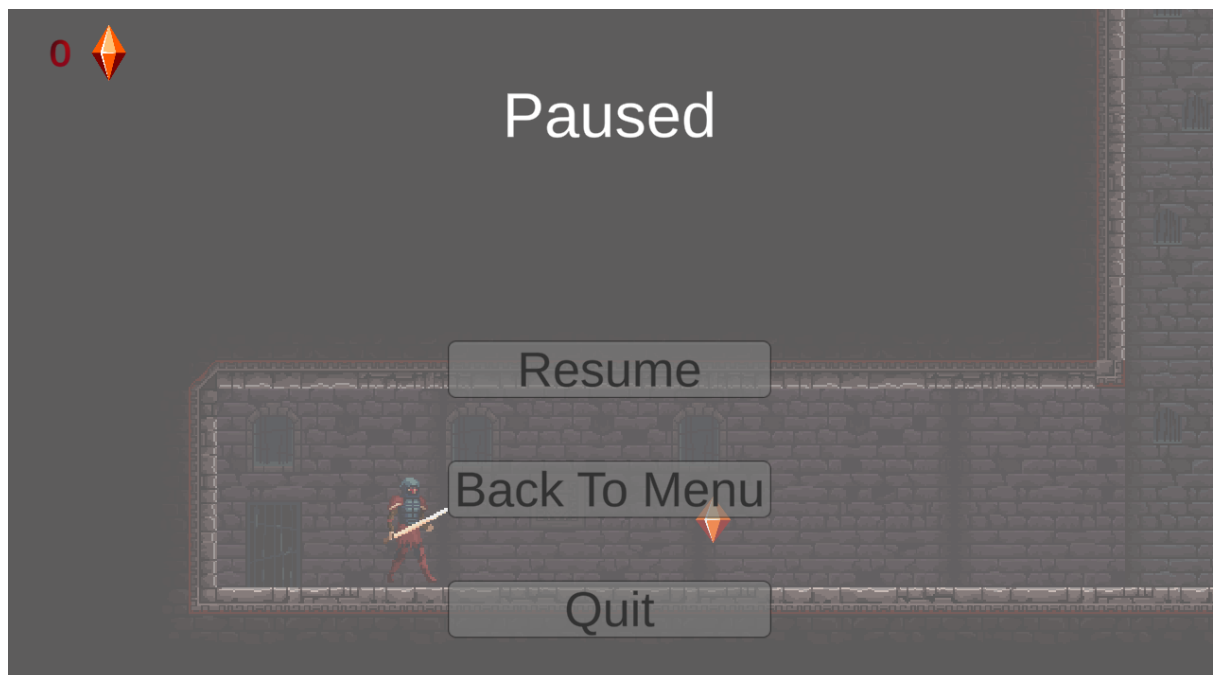
```

private void FixedUpdate()
{
    mx = mx * movementSpeed;
    if (isrunning)
        mx = mx * movementSpeedmodifier;
    Vector2 movement = new Vector2(mx,rb.velocity.y);
    rb.velocity = movement;
}
1 reference
void Jump(){
    Vector2 movement = new Vector2(rb.velocity.x,jumpForce);
    rb.velocity = movement;
}
0 references
public bool IsGrounded1()
{
    Collider2D groundCheck = Physics2D.OverlapCircle(feet.position,0.5f,groundLayers);
    if(groundCheck != null){
        return true;
    }
    return false;
}
2 references
public bool IsGrounded()
{
    RaycastHit2D hit = Physics2D.BoxCast(colider.bounds.center, colider.bounds.size, 0, Vector2.down, 0.1f, groundLayers);
    return hit.collider != null;
}

```

## 12.Meniul de pauză

Acest meniu se suprapune peste scena nivelului curent.



Codul din spatele meniului de pauză:

```
public GameObject pauseMenu;

private static bool isPaused=false;
@ Unity Message | 0 references
private void Awake()
{
    isPaused = false;
}

@ Unity Message | 0 references
private void Update()
{
    if (Input.GetKeyDown(KeyCode.Escape))
    {
        if (isPaused)
            ResumeGame();
        else
            PauseGame();
    }
}

1 reference
void PauseGame()
{
    pauseMenu.SetActive(true);
    Time.timeScale =0f;
    isPaused = true;
}

1 reference
void ResumeGame()
{
    pauseMenu.SetActive(false);
    Time.timeScale = 1f;
    isPaused = false;
}

0 references
public void GoToMainMenu()
{
    Time.timeScale = 1f;
    SceneManager.LoadScene(4);
}

0 references
public void QuitGame()
{
    Application.Quit();
}
```

## 4. Extinderi posibile ale aplicației

Se mai pot adăuga nivele cât și texturi noi pentru caracter , acestea fiind modificate după plac.

Se mai poate reține timpul de parcurgere al fiecărui nivel pentru fiecare jucător.