Liceul de informatică “Tiberiu Popoviciu” , Cluj-Napoca

Proiect pentru obținerea atestării profesionale în informatică

SPARTALEX

Cluj-Napoca,

Mai 2021

Ciurtea Alexandru

cls a XII a C

Coordonator : prof. Andreea Racolta

Cuprins

1.Prezentarea generală a aplicației

Aplicația Spartalex este un joc în care utilizatorul controlează cu ajutorul tastelor W,A,S,D un spartan , luptându-se cu o varietate de inamici , atacându-i prin apăsarea tastei Space.Jocul înregistrează un scor , cele mai mari trei scoruri fiind afișate pe o tabelă.

Aplicația :

- Poate să genereze la infinit valuri de inamici.
- Afișează punctajul jucătorului la finalul fiecărei runde.
- Reține si afișează cele mai mari 3 scoruri obținute de diferiți jucători .
- Reține până la trei profiluri în funcție de numele introdus la începutul jocului .

2.Resurse hardware și software necesare

Resurse necesare:

- Procesor minim – 2GHz
- Memorie RAM minimă- 2 GB RAM
- Memorie minimă pe Hard Disk – 1 GB

3.Realizarea aplicației

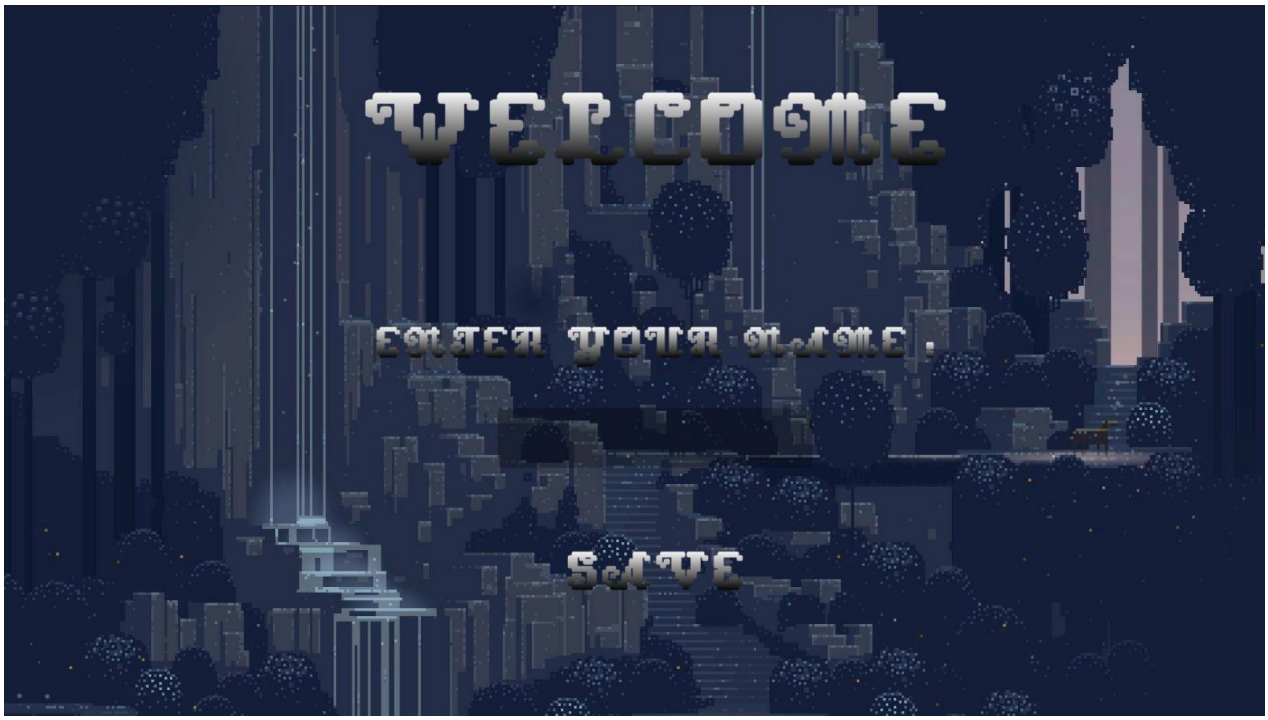
Spartalex este un joc ce pune la încercare dibăcia jucătorului acesta fiind nevoit să se lupte cu diferiți inamici .Jocul te pune în poziția unui caracter(un spartan) cu care te poți mișca in ce direcție dorești și să ataci inamicii(vrăjitori , samurai și alte creaturi) care sunt generați la nesfârșit , numărul de inamici crescând la fiecare nou val . Fiecare inamic are avantajele și dezavantajele sale acestea constând in daunele pe care le provoacă, în viața pe care o au , si in viteza cu care se deplasează. Numărul de inamici uciși formează un scor. Scorurile obținute fiind memorate. Grafica este una simplă tipică jocurilor mai vechi inducând nostalgie jucătorului. Caracterele sunt desenate in stilul PixelArt, iar animațiile fiind realizate printr-o succesiune rapidă a mai multor desene.

Pentru proiectarea jocului am folosit aplicația Unity. Jocul e structurat din mai multe scene care comunică între ele cu ajutorul unor Scripturi . Aceste scripturi ocupându-se si de logica din spatele jocului.

Scenele sunt:

1. Scena de Logare

În această scenă utilizatorul își poate alege numele pe care îl dorește jocul reținând acest nume și la viitoarele login-uri.

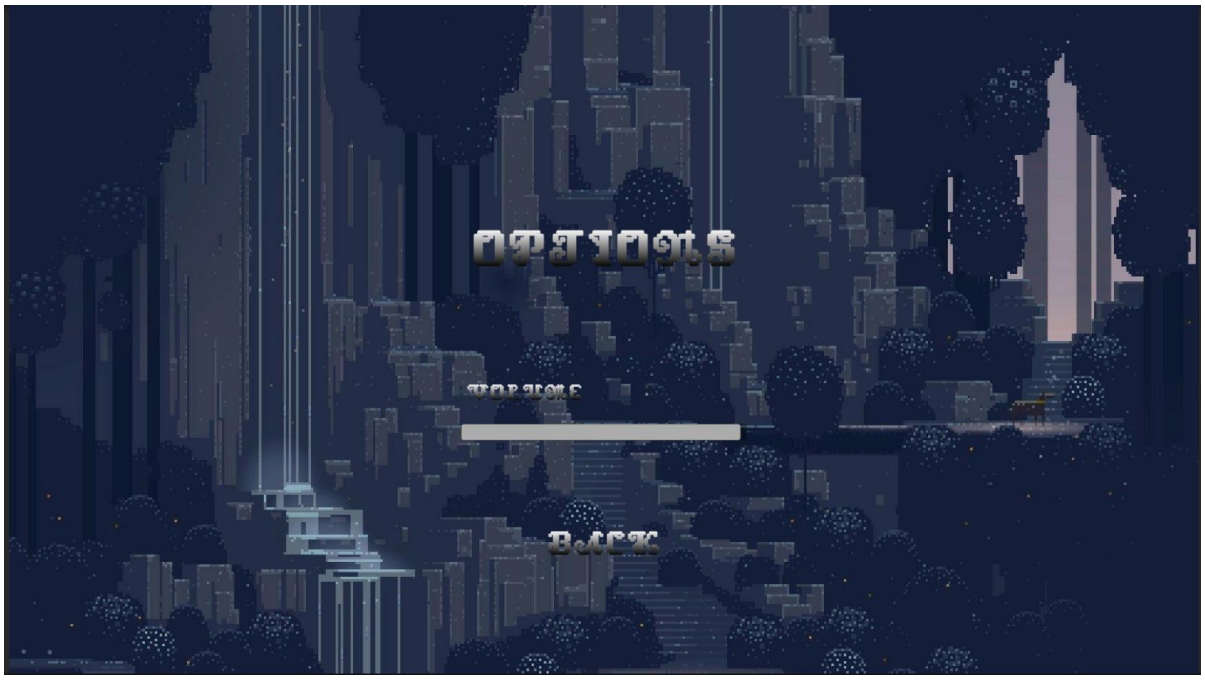


2. Scena Meniului Principal

Această scenă face legătura înspre restul scenelor în funcție de butonul apăsat. Jucătorul având posibilitatea să joace jocul , să intre in meniul cu setări, să vadă tabela cu scoruri sau să părăsească jocul .

3. Scena Setărilor

Aici utilizatorul poate să ajusteze sunetul jocului dupa bunul său plac cu ajutorul unu slider.



4. Scena Tabelei de Scor

În această scena utilizatorul poate observa numele jucătorilor cu cele mai mari 3 scoruri dar și scorul acestora. În cazul în care utilizatorul depășește unul dintre aceste scoruri el va observa apariția sa in tabelă.

5. Scena jocului propriu-zis

Aici jucătorul poate juca jocul cu ajutorul tastelor W,A,S,D pentru a se misca si a tastei Space pentru a ataca. Caracterului îi sunt atribuite mai multe scripturi care permit mișcarea acestuia dar si altele precum producerea unui atac și scăderea vieții în funcție de daunele primite în urma atacului de la un inamic .

```
public CharacterController2D controller;
public Animator animator;
public float runSpeed = 40f;
bool jump = false;
float horizontalMove = 0f;
// Update is called once per frame
void Update()
{
   horizontalMove = Input.GetAxisRaw("Horizontal") * runSpeed;
   animator.SetFloat("Speed", Mathf.Abs(horizontalMove));
   if (Input.GetButtonDown("Jump"))
   {
       jump = true;
       animator.SetBool("Jumping", jump);
   }


}
public void OnLanding()
{
    animator.SetBool("Jumping", false);
}
void FixedUpdate ()
{
    // Move character
    controller.Move(horizontalMove * Time.fixedDeltaTime, false , jump);

    jump = false;
```

```
public Transform attackPoint;
public float attackRange = 0.5f;
public LayerMask enemyLayer;
public int attackDamage = 40;
public float attackRate = 2f;
float nextAttackTime = 0f;

// Update is called once per frame
void Update()
{
    if (Time.time >= nextAttackTime)
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            Attack();
            nextAttackTime = Time.time + 1f / attackRate;
        }
    }
}
void Attack()
{
    FindObjectOfType<AudioManager>().Play("PlayerHit");
    animator.SetTrigger("Attack");
    Collider2D[] hitEnemies = Physics2D.OverlapCircleAll(attackPoint.position, attackRange, enemyLayer);
    foreach (Collider2D enemy in hitEnemies)
    {
        enemy.GetComponent<Enemy_Combat>().TakeDamage(attackDamage);
    }
```

Inamicii apar cu ajutorul unui ”spawner” acesta fiind capabil sa produca apariția inamicilor. Fiecare “spawner” creeând un anumit tip de inamic. Inamici urmăresc caracterul jucătorului cu ajutorul unui pathfinder care înregistrează terenul prin care nu pot trece caracterele si localizează poziția țintei sale .La fiecare inamic ucis scorul va crește cu câte un punct.

```
        searchCountdown -= Time.deltaTime;

        if (searchCountdown <= 0f)
        {
            searchCountdown = 1f;
            if (GameObject.FindGameObjectsWithTag("Enemy").Length <= 3)
         {
             return false;
         }

        }

         return true;
    }
    IEnumerator SpawnWave(Wave _wave)
    {
         state = SpawnState.SPAWNING;
         animator.SetBool("Spawning", true);
         for (int i = 0; i < _wave.count; i++)
         {
             SpawnEnemy(_wave.enemy);
             yield return new WaitForSeconds(1f / _wave.rate);
         }
         animator.SetBool("Spawning", false);

         state = SpawnState.WAITING;
         yield break;
    }
```

Atât caracterul controlat de către jucător cât și inamicii au sunete specifice in momentul în care atacă dar și când mor acest lucru oferind jucătorului o experiență mult mai bună in momentul jocului.

```
        if (currentHealth <= 0)
        {
            Die();
        }
        healthbar.SetHealth(currentHealth);
    }
    void Die()
    {

        animator.SetBool("Dead", true);
        GetComponent<CircleCollider2D>().enabled = false;
        GetComponent<Collider2D>().enabled = false;
        GetComponent<PolygonCollider2D>().enabled = true;
        FindObjectOfType<AudioManager>().Play("PlayerDeath");
        StartCoroutine(waiter());

        this.enabled = false;
    }
    IEnumerator waiter()
```

```
}
void Update()
{
    if (Time.time >= nextAttackTime)
    {

        Collider2D[] colliders = Physics2D.OverlapCircleAll(attackPoint.position, attackRange, enemyLayer);
        for (int i = 0; i < colliders.Length; i++)
        {
            if (colliders[i].gameObject != gameObject)
            {
                nextAttackTime = Time.time + 1f / attackRate;
                animator.SetTrigger("Attack");
                if (Eye == true) FindObjectOfType<AudioManager>().Play("EyeHit");
                if (Wizard == true) FindObjectOfType<AudioManager>().Play("WizardHit");
                if (Samurai == true) FindObjectOfType<AudioManager>().Play("PlayerHit");
                Check = Time.time + 0.3f;

            }
```

Jocul are si un meniu in momentul in care dorești sa pui pauză , acest lucru este posibil apăsând tasta ESC . Din acest meniu jucătorul poate sa revină la joc , să meargă la meniul principal ,acest lucru oprind complet jocul dar scorul fiind salvat, sau să părăsească jocul .

```
}
public void Resume()
{
    pauseMenuUI.SetActive(false);
    Time.timeScale = 1f;
    GameIsPaused = false;
}
void Pause()
{
    pauseMenuUI.SetActive(true);
    Time.timeScale = 0f;
    GameIsPaused = true;
}
public void LoadMenu()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex - 1);
    PlayerPrefs.SetInt("score", PlayerPrefs.GetInt("score1"));
}
public void QuitGame()
{
    Application.Quit();
}
```

În momentul în care caracterul tău este ucis pe ecran va apărea mesajul “You died” însoțit de scorul pe care l-ai obținut si posibilitatea de a merge la meniul principal sau de a ieși din joc .

4. Extinderi posibile ale aplicației

-Adăugarea a noi caractere cu diferite puteri si abilități cu care se pot juca utilizatorii

-Adăugarea unor noi inamici cu diferite abilități

-Extinderea hărții curente si creearea unor noi hărți

-Adăugarea unui sistem care sa răsplătească jucătorul pentru numărul de inamici doborâți