KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Žaidimų ir interaktyviųjų programėlių kūrimas (T120B166)

3 laboratorinio darbo ataskaita

Atliko:

IFF-5/1 gr. studentas

Deividas Šeimys

2018 m. birželio 6 d.

Priėmė:

Prof. Rytis Maskeliūnas

KAUNAS 2018

TURINYS

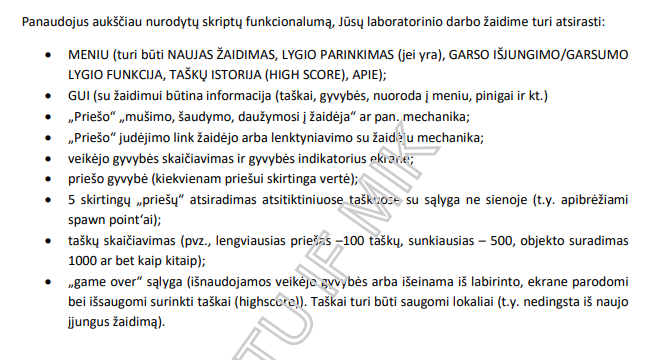
[1. Užduotis 3](#_Toc512869536)

[2. Implementacija 3](#_Toc512869537)

[3. Ekranvaizdžiai 4](#_Toc512869538)

[4. Šaltiniai 12](#_Toc512869539)

# Užduotis



# Implementacija

Buvo programuota naudojant unity ir C#.

# Ekranvaizdžiai

## MENIU kitų galimų option įgyvendinimas

using UnityEngine;

using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour {

//Naudojamas užkrauti pagrindiniai scenai

public void PlayGame ()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);

Time.timeScale = 1f;

WaveSpawner.EnemiesAlive = 0;

}

//naudojams sugrįžti į pagrindinį meniu

public void ExitToMainMenu()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex - 1);

}

//naudojamas išeiti iš game

public void QuitGame()

{

Application.Quit();

}

//Pause ir Unpause

public void Pause()

{

Time.timeScale = 0f;

}

public void Unpause()

{

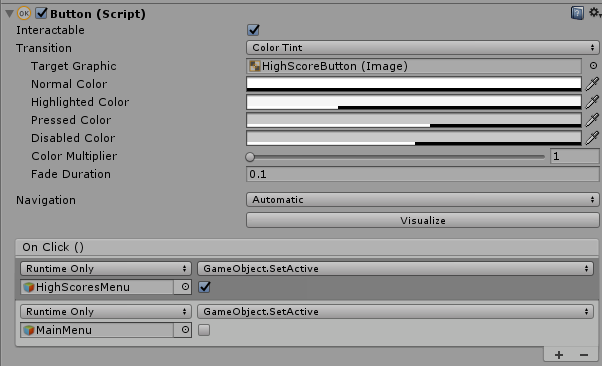
Time.timeScale = 1f;

}

}}

Panašių skriptų turėjau kelis, tačiau daug ką pavyko padaryt su Unity pagalba. Pvz:

Lengviau naršyti menu ir nereik scriptų naudojant on click objektus nustatant active arba inactive.



Paveiksliukas 1. Button funkcionalumas

## GUI įgyvendinimas

using TMPro;

using UnityEngine;

public class LivesUI : MonoBehaviour {

public TextMeshProUGUI livesText;//naudojau projektui unity asset textmeshpro

void Update()

{

livesText.text = PlayerStats.Lives.ToString();//čia priskiriam naują tekstą, tai yra gyvybęs saugomas PlayerStats

}

}

Didžioji dalis susijusi su tekstu GUI buvo įgyvendinta panašiai(kai kur buvo atnaujinama vidui kito skripto).

Options MENU buvo panaudotas anksčiau pateiktas skriptas(3.1 dalyje) ir taip įgyvendinta grįžimo į pagrindinį MENU galimybė.

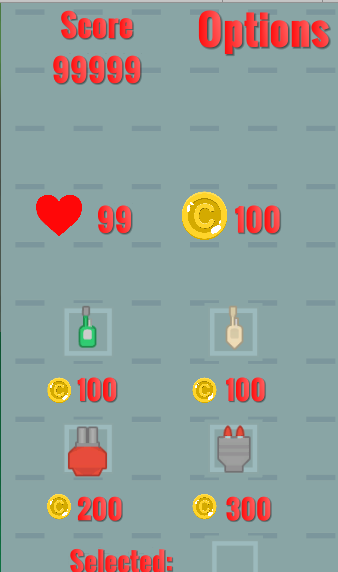
//naudojams sugrįžti į pagrindinį meniu

public void ExitToMainMenu()

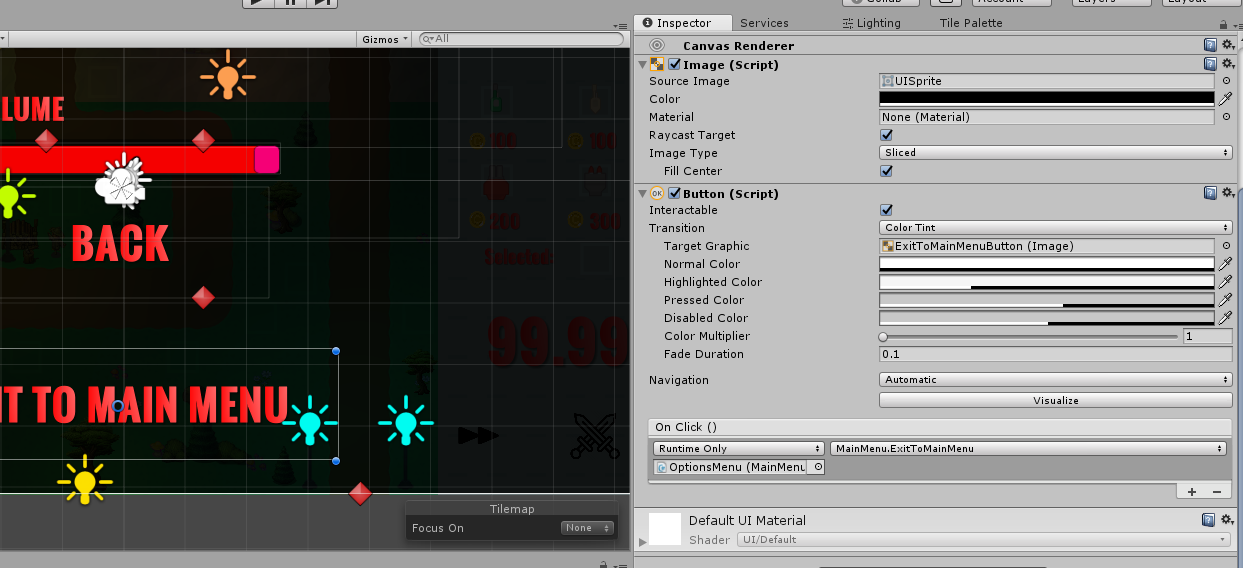
{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex - 1);

}



Paveiksliukas 2. GUI iškarpa



Paveiksliukas 3. Exit To Main Menu funkcionalumas

## „Priešo“ „Šaudymas į žaidėją“

/// <summary>

/// Susiranda target kuris yra range, kad galetu ji pult

/// </summary>

void UpdateTarget()

{

GameObject[] turrets = GameObject.FindGameObjectsWithTag(turretTag);

float shortestDistance = Mathf.Infinity;

GameObject nearestTurret = null;

foreach (GameObject turret in turrets)

{

float distanceToEnemy = Vector3.Distance(transform.position, turret.transform.position);

if (distanceToEnemy < shortestDistance)

{

shortestDistance = distanceToEnemy;

nearestTurret = turret;

}

}

if (nearestTurret != null && shortestDistance <= range)

{

targetToShoot = nearestTurret.transform;

}

else

{

targetToShoot = null;

}

}

Tada update funkcijoj tikrinamas „cooldown“ iki kito šūvio ir jei jis 0 šaunama.

if (targetToShoot != null)

if (fireCountdown <= 0f)

{

Shoot();

fireCountdown = 1f / fireRate;

}

fireCountdown -= Time.deltaTime;

Sukuriama nauja kulka, kuri keliauja iki target‘o

void Shoot()

{

GameObject bulletGO = (GameObject)Instantiate(bulletPrefab, transform.position, transform.rotation);

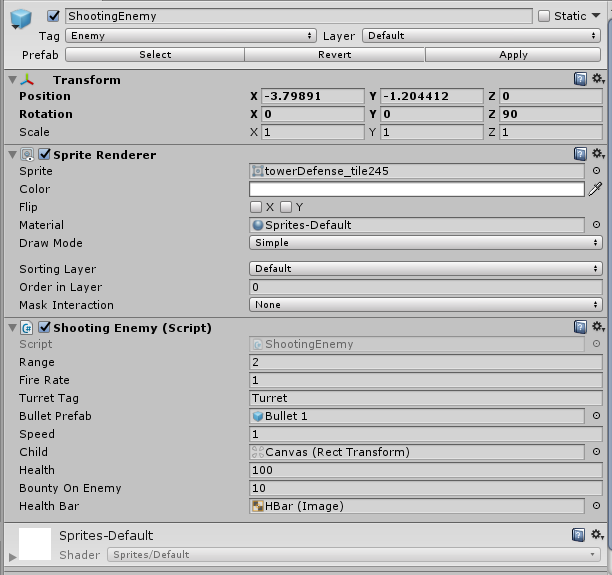
Bullet bullet = bulletGO.GetComponent<Bullet>();

if (bullet != null)

bullet.Seek(targetToShoot);

}

Kulkos skripte yra mechanika, kuri apdoroja, kada jai išnykti ir kiek žalos, kam daryti. Bullet skriptas yra ant bullet prefab.



Paveiksliukas 4. Priešo mechanikai naudotas skriptas

## „Priešo“ judėjimas link žaidėjo

Šiuo atveju buvo judėjimas iki mūsų žemėlapio pabaigos, nustatytais „waypoints“.

void Update()

{

Vector3 dir = target.position - transform.position;//Gaunam kryptį

transform.Translate(dir.normalized \* speed \* Time.deltaTime, Space.World);//Judėjimas

float angle = Mathf.Atan2(dir.y, dir.x) \* Mathf.Rad2Deg;

int angleInt = Mathf.RoundToInt(angle);

if (angleInt == 1)//laikinas bugfixas kai angleInt būna 1

angleInt = 0;

Quaternion q = Quaternion.AngleAxis(angleInt, Vector3.forward);

transform.rotation = Quaternion.Slerp(transform.rotation, q, Time.deltaTime \* 100);

RotateHealthBar(angleInt, child);

if (Vector3.Distance(transform.position, target.position) <= 0.02f)//Jei pasiekėm tašką paimtų kitą

GetNextWaypoint();

}

/// <summary>

/// waypoint gavimas

/// </summary>

void GetNextWaypoint()

{

if (wavepointIndex >= Waypoints.points.Length - 1)

{

EndPath();

return;

}

wavepointIndex++;

target = Waypoints.points[wavepointIndex];

}

/// <summary>

/// jei kelio pabaiga

/// </summary>

void EndPath()

{

PlayerStats.Lives--;//praranda gyvybę

WaveSpawner.EnemiesAlive--;//gyvų priešų skaitiklis sumažėja vienu

Destroy(gameObject);//sunaikinamas priešas

}

## Veikėjo gyvybės indikatorius ekrane

Įgyvendintas buvo sekant PlayerStats kiek gyvybių turi žaidėjas ir jas perduodant į HealthText objektą mūsų žaidime, kuris ir yra text objektas.

using TMPro;

using UnityEngine;

public class LivesUI : MonoBehaviour {

public TextMeshProUGUI livesText;//naudojau projektui unity asset textmeshpro

void Update()

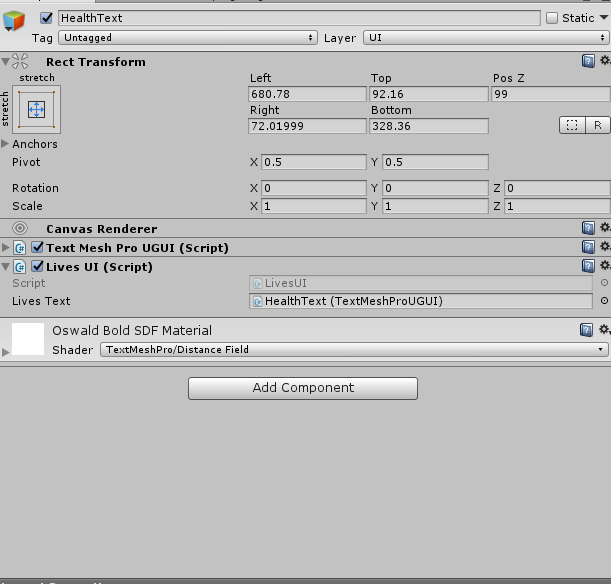
{

livesText.text = PlayerStats.Lives.ToString();//čia priskiriam naują tekstą, tai yra gyvybęs saugomas PlayerStats

}

}

PlayerStats.lives yra static ir galima pasiekt iš betkurios klasės.



Paveiksliukas 5. HealthText scripts Lives UI

## Priešo gyvybės

Kai priešas gauna žalos(arba bokštelis) tai perteikiama ir žaidėjui.

public void TakeDamage(int dmgAmount)

{

health -= dmgAmount;//dmg gavimas

HealthBar.fillAmount = health / maxHealth;//atnaujinamas health bar

if (health <= 0)//mirti jei nėra gyvybių

Die();

}



Paveiksliukas 6. Priešo gyvybės visi skirtingai turi

## 5 skirtingų „priešų“ atsiradimas su sąlgyga ne sienoje

Įgyvendinau tik 4 skirtingus priešus ir jie atsiranda viename spawn point

Atsiradimo logikai buvo sukurtas „WaveSpawner“ skriptas.

using UnityEngine;

using System.Collections;

using TMPro;

public class WaveSpawner : MonoBehaviour {

public static int EnemiesAlive = 0;

public Wave[] waves;

public Wave boss;

public Transform spawnPoint;

public AudioSource waveSpawnSound;

public float timeBetweenWaves = 5f;

public float countdown = 0f;

private float timeBetweenEnemies = 0.5f;

private bool isSpeeded = false;

private bool isFirstWave = true;

private bool isCoroutineStarted = false;

private int waveNumber = 0;

public TextMeshProUGUI timerText;

void Update()

{

if (EnemiesAlive > 0)//jei gyvų priešų yra nieko nedaryti

return;

if (countdown <= 0f && !isFirstWave)// jei pirmas wave ir countdown 0 nieko nedaryti

{ // kitu atveju spawn wave pagal time po pirmo wave(yra galimybė ir pačiam paleist wave nepasibaigus laikui)

StartCoroutine(SpawnWave()); //pradeda spawn coroutine

countdown = timeBetweenWaves; //timeris

return;

}

countdown -= Time.deltaTime; //timeris mažėjai pagal mūsų Time, o ne frmaes

countdown = Mathf.Clamp(countdown, 0f, Mathf.Infinity);

timerText.text = string.Format("{0:00.00}", countdown);

}

public void StartCoroutineViaButton() //Galimybė pradėti coroutine mygtuko paspaudimu(pirmam wave būtina paspaust)

{

if (isCoroutineStarted)

return;

StartCoroutine(SpawnWave());

isFirstWave = false;

countdown = timeBetweenWaves;

timerText.text = string.Format("{0:00.00}", 0f);

}

public void SpeedUp()//Galėjimas pagreitinti eiga

{

if (!isSpeeded)

{

Time.timeScale = 2f;

isSpeeded = true;

}

else

{

Time.timeScale = 1f;

isSpeeded = false;

}

}

IEnumerator SpawnWave()//spawn coroutine

{

isCoroutineStarted = true;

waveSpawnSound.Play();//garsas paspawninus wave

Wave wave = waves[waveNumber];//iš visų wave array paimamas atitinkamas wave kurį ir spawninam

for (int i = 0; i < wave.amountOfEnemiesToSpawn; i++)//ciklas spawnint visus wave enemy

{

SpawnEnemy(wave.enemyPrefab);

yield return new WaitForSeconds(1f / wave.rate);

}

if ((waveNumber + 1) % 5 == 0)//kas penkta wave boss tipas

SpawnEnemy(boss.enemyPrefab);

waveNumber++;//padidinam wave numerį

if (waveNumber == waves.Length)

{

Debug.Log("Level won!");//TODO galima būtų padaryt ekrana kad laimėjai, bet dabar sąlyga tik kad pralaimėsim

this.enabled = false;

}

isCoroutineStarted = false;

}

void SpawnEnemy(GameObject enemyPrefab)//spawn enemy metodas

{

Instantiate(enemyPrefab,spawnPoint.position, spawnPoint.rotation);

EnemiesAlive++;

}



Paveiksliukas 7. Keli skirtingi priešai

## Taškų skaičiavimas

Taškai turi „Score UI“ skriptą, kuris atnaujiną taškus. Patys taškai saugomi „PlayerStats“. Taškai padidinami kiekvieną kartą kai priešas mirštą.

using TMPro;

using UnityEngine;

public class ScoreUI : MonoBehaviour {

public TextMeshProUGUI scoreText;

void Update()

{

scoreText.text = "SCORE\n"+PlayerStats.Score.ToString();

}

}

using UnityEngine;

public class PlayerStats : MonoBehaviour {

public static int Gold;//auksas žaidimo metu

public int startGold = 400;//auksas pradžioj

public static int Lives;//gyvybės žaidimo metu

public int startLives = 20;//gyvybės pradžioj

public static int Score;//score

void Start()

{

Gold = startGold;

Lives = startLives;

Score = 0;

}

}

Iš enemy script paimta ištrauka:

void Die()

{

PlayerStats.Gold += bountyOnEnemy;

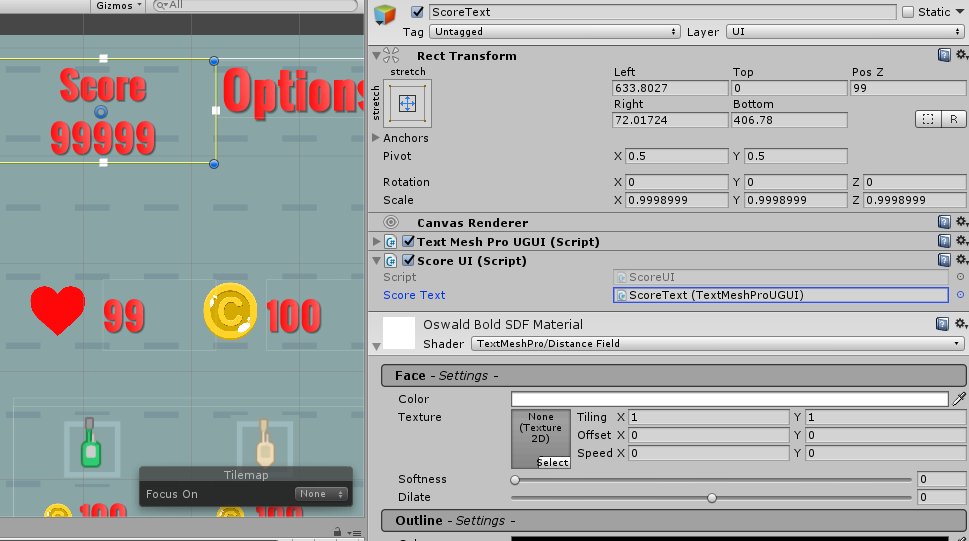
PlayerStats.Score += bountyOnEnemy \* 10;//taškai skaičiuojami "priešo vertė"\*10

//skirtingi priešai turi skirtinga vertę tai ir skirtingai skaičiuojama

WaveSpawner.EnemiesAlive--;

Destroy(gameObject);

}



Paveiksliukas 8. Taškų tekstas

## Game Over sąlyga ir taškų saugojimas lokaliai

Pasibaigus gyvybėms, įjungiamas game over ekranas ir žaidimas neveikia background‘e.

Game over ekranas įjungiamas per game manager, ten pat ir sutvarkomi high scores naudojant „PlayerPrefs“.

Galimybė Retry arba grįžt į Menu.

Išvedamas score ir vietą kurią užėmei ( jei užėmei).

using TMPro;

using UnityEngine.SceneManagement;

using UnityEngine;

public class GameOver : MonoBehaviour {

public TextMeshProUGUI scoreText;

public string highScoreKey = "HighScore";

void OnEnable()

{

scoreText.text = "GAME OVER\n" + "Score: " + PlayerStats.Score.ToString();

Time.timeScale = 0f;

int place = 0;

for (int i = 1; i <= 5; i++)

{

if (PlayerStats.Score > PlayerPrefs.GetInt(highScoreKey + i.ToString()) && place == 0)

place = i;

}

if (place != 0)

scoreText.text += "\n PLACED - " + place;

}

public void ExitToMainMenu()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex - 1);

}

public void Retry()

{

Time.timeScale = 1f;

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);

WaveSpawner.EnemiesAlive = 0;

}

}

using UnityEngine;

public class GameManager : MonoBehaviour {

private bool gameOver = false;

public string highScoreKey = "HighScore";

public GameObject gameOverUI;

void Update()

{

if (gameOver)

return;

if (PlayerStats.Lives <= 0)

EndGame();

}

void EndGame()

{

gameOver = true;

gameOverUI.SetActive(true);

SetHighScores();

}

void SetHighScores()

{

int replaceIndex = 0;

for (int i = 1; i <= 5; i++)

{

if (PlayerStats.Score > PlayerPrefs.GetInt(highScoreKey + i.ToString()) && replaceIndex == 0)

replaceIndex = i;

}

if (replaceIndex == 0)

return;

else

for (int i = 5; i > replaceIndex; i--)

{

PlayerPrefs.SetInt(highScoreKey + i.ToString(), PlayerPrefs.GetInt(highScoreKey + (i - 1).ToString()));

}

PlayerPrefs.SetInt(highScoreKey + replaceIndex.ToString(), PlayerStats.Score);

}

}



Paveiksliukas 9. Game over ekranas

# Šaltiniai

**Remtasi tutorial:** <https://www.youtube.com/watch?v=beuoNuK2tbk&list=PLPV2KyIb3jR4u5jX8za5iU1cqnQPmbzG0>

**Sprites:**

Environment SprideSheet‘as iš kurio ėmiau kaukoles, kaktusus ir dar kelis kitus sprite: <http://i.imgur.com/qYFvsmq.png>

Tree1: <https://i.pinimg.com/originals/b5/5e/06/b55e068de8ba7038f7455d1606ac7e6e.png>

Trees SprideSheet‘as iš kurio imta dauguma medžių: <https://orig00.deviantart.net/21dd/f/2011/107/d/7/new_tree_set_by_pinkfirefly-d3e8jc6.png>

TreeType4: <https://i.pinimg.com/originals/ed/65/20/ed65200f9100cdd9b09aa39ac573d8b5.png>

Namukas: <https://img.itch.zone/aW1hZ2UvMTMyNzA3LzYwOTIyNS5wbmc=/347x500/rengbO.png>

Žibintai: <https://www.engine001.com/community/resources/lit_lamppost-IKQBNX.gif>

Ugnies SprideSheet‘as: <https://mir-s3-cdn-cf.behance.net/project_modules/disp/38280d30108625.5613dd2706e3b.png>

Tiles/Sprites map‘ui: <https://opengameart.org/content/tower-defense-300-tilessprites>

**Garsai ir muzika:**

Muzika: <http://soundimage.org/fantasywonder/>

<http://soundimage.org/positive-upbeat/>

**iš šitų dviejų visos penkios muzikėlės paimtos**

Sounds: <http://soundimage.org/sfx-combat/>

<http://www.orangefreesounds.com>

<http://soundimage.org/sfx-household/>

<http://soundimage.org/sfx-scifi/>

**iš šitų keturių visi 10 garsų paimta**