## ÐÐ°ÑÑÐ¸Ð½ÐºÐ¸ Ð¿Ð¾ Ð·Ð°Ð¿ÑÐ¾ÑÑ ktu logo

**KAUNAS UNIVERSITY OF TECHNOLOGY**

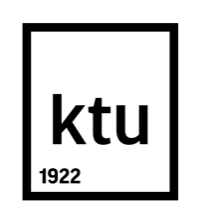
**FACULTY OF INFORMATICS**

**T120B166 Development of Computer Games and Interactive Applications**

*Roguevania*

|  |
| --- |
| *IFF-8/12 Gytautas Kazlauskas*  *IFF-8/12 Justas Kučinskas*  *IFF-8/13 Mykolas Paulauskas*  *IFF-8/13 Jokūbas Keturakis* |
|  |
| Date: *2021.02.28* |

**Tables of Contents**

[ 1](#_Toc72074030)

[**Tables of Images** 4](#_Toc72074031)

[**Table of Tables/functions** 5](#_Toc72074032)

[**Work Distribution Table:** 6](#_Toc72074033)

[**Description of Your Game** 7](#_Toc72074034)

[**Solution** 8](#_Toc72074035)

[**Task #1. *Basic character controller*** 8](#_Toc72074036)

[**Task #2. *Basic HUD (HP), with a way to test if it works*** 11](#_Toc72074037)

[**Task #3. *Main character’s attack functionality (animations, basic logic)*** 11](#_Toc72074038)

[**Task #4. *Basic enemy interaction (taking damage, dying, animations)*** 12](#_Toc72074039)

[**Task #5. *Changing scenes*** 14](#_Toc72074040)

[Task #6. *Combat action - fireball* 16](#_Toc72074041)

[Task #7. *Defence task - dash scatter fireball glow* 16](#_Toc72074042)

[**Solution** 17](#_Toc72074043)

[**Task #1. *Expanding world environment, animating enemies with particle effects*** 18](#_Toc72074044)

[**Task #2. *Experiment with different physics*** 20](#_Toc72074045)

[**Task #3. *Background, invisible walls, segway to new level, light effect on mobs*** 21](#_Toc72074046)

[**Task #4. *Expand the game by adding a new level*** 24](#_Toc72074047)

[**Task #5. *Add a trapdoor.*** 26](#_Toc72074048)

[**Task #6. *Defence task*** 26](#_Toc72074049)

[**Solution** 29](#_Toc72074050)

[**Task #1. *Game menu*** 29](#_Toc72074051)

[**Task #2. *Audio*** 31](#_Toc72074052)

[**Task #3. *Bonfires*** 32](#_Toc72074053)

[**Task #4. *Game over*** 37](#_Toc72074054)

[Movement.cs 42](#_Toc72074055)

[Player.cs 43](#_Toc72074056)

[HPController.cs 43](#_Toc72074057)

[HUDController.cs 44](#_Toc72074058)

[PulsatingLight.cs 44](#_Toc72074059)

[NPC.cs 45](#_Toc72074060)

[ObjectTracking.cs 45](#_Toc72074061)

[Parallaxing.cs 45](#_Toc72074062)

[PlayerCombat.cs (Justas) 46](#_Toc72074063)

[Enemy.cs (Justas) 47](#_Toc72074064)

[SceneLoader.cs (Justas) 48](#_Toc72074065)

# **Tables of Images**

**Figure 1.** Player moving to the right

**Figure 2.** Player dashing to the right

**Figure 3.** HP UI showcase, the values are dynamic

**Figure 4.** Animation of attacking

**Figure 5.** Mob spawning animation

**Figure 7.** Animation of enemy getting killed and disappearing

**Figure 8.** Door sprite upon colliding with which, the user is taken to the dungeon’s entrance scene

# **Table of Tables/functions**

**Table 1.** Movement controller Update code

**Table 2.** Dash velocity and cooldown code

**Table 2.** The code used to update UI elements to the respected values

**Table 3.** Code responsible for attacking enemies

**Table 4.** Code responsible for enemy dying

# **Work Distribution Table:**

|  |  |
| --- | --- |
| ***Name/Surname*** | ***Description of game development part*** |
| *Gytautas Kazlauskas* | *Lab1: The starting hub world with placeholder interactions, basic character controller with animations as well as the basics of the main HUD (HP).*  *Lab2: Expand the game by creating and designing a new level, experiment with tilemaps, tilemap colliders. Add a trapdoor which stepped upon breaks, releases particle effects and emits a sound*  *Lab3: Sound effects, game menu, game over screen, bonfires, inventory use.* |
| *Justas Kučinskas* | *Lab1: Level 1 (Entrance) layout of the dungeon, animations and functionality of character’s attacks, animations and basic interactions with enemies, loading of a new scene.*  *Lab2: Add a new enemy, which would scope out the surroundings and patrol an area, animate him with three different types of particles, add an area upon entering which the player gets slowed down, add skybox(background) and invisible walls which would prevent player from leaving map boundaries, add entry segway to Escape level.*  *Lab3: Sound effects, boss level, boss mob, audio, boss HUD, transition from Escape area to Boss area.* |
| *Mykolas Paulauskas* | *Lab1: Advanced player movement ability - dashing, which utilizes after image pool prefab. Basic enemy follow script and player hit detection.* |
| *Jokūbas Keturakis* | *Lab1: Advanced player combat ability - fireball, which allow players to cast fireballs at enemies. Also, a design for shop’s interface was made along with the addition of gold counter for player HUD.* |

# **Description of Your Game**

Description of Your Game.

1. 3D or 2D? *2D*
2. What type is your game? *The game as the name implies is a mix of metroidvania, where the player traverses the world acquiring upgrades to reach previously unreachable places, with the twist that the only way to acquire those upgrades is from the hub that can be accessed after dying in previous runs.*
3. What genre is your game? *2D side scroller, action, adventure.*
4. Platforms (mobile, PC or both?) *PC*
5. Scenario Description. *The player awakens in a desolate graveyard without any recollection of why they are there. They are then greeted by a mysterious statue that explains that they are undead that were awakened after a mysterious castle has appeared. The player is quickly introduced to the hub world and ventures into the castle. The castle is filled with enemies which upon killing them the main character receives soul fragments that are needed to upgrade his physical attributes and unlock new abilities. Upon entering the player realizes that there is no way to get out of the castle and the only way to return to the hub is to die upon which he is then awakened in the same graveyard as when the game started. And the cycle continues until the player finds the reason why the castle appeared and why they were awakened.*

Laboratory work #1

**List of tasks** (main functionality of your project)

1. Basic character controller - Gytautas
2. Basic HUD (HP), with a way to test if it works - Gytautas
3. Main character’s attack functionality (animations, basic logic) - Justas
4. Basic enemy interaction (taking damage, dying, animations) - Justas
5. Changing scenes (dungeon entrance) - Justas
6. Combat action - fireball - Jokūbas

# **Solution**

## **Task #1. *Basic character controller***

Description of the implementation (3-5 sentences). *For the first laboratory work we have decided to implement our own character controller and then later use a pre built one if we can find a suitable one. The way it was implemented is by using clamped force as our main way of moving, with an instant stop once there is no input. This is accomplished by taking the horizontal force of input and multiplying it by speed and then clamping if velocity exceeds the predefined ranges. Same is done for jumping except that velocity is not clamped instead the input is ignored until collision with ground is detected. We also added a player dash, which used an after image pool of sprites to produce a smooth animation. The dash itself is realized by adding force to the rigidbody2D component and is regulated by a cooldown.*



**Figure 1.** Player moving to the right

*void Update()*

*{*

*float hForce = Input.GetAxisRaw("Horizontal") \* Speed \* Time.deltaTime;*

*if (hForce == 0.0f)*

*{*

*rBody.velocity = new Vector2(0, rBody.velocity.y);*

*animator.SetFloat("Speed", 0.0f);*

*}*

*else*

*{*

*rBody.AddForce(Vector2.right \* hForce);*

*animator.SetFloat("Speed", Mathf.Abs(hForce));*

*if (hForce > 0.0f)*

*{*

*transform.eulerAngles = new Vector3(0.0f, 0.0f, 0.0f);*

*}*

*else if (hForce < 0.0f)*

*{*

*transform.eulerAngles = new Vector3(0.0f, 180.0f, 0.0f);*

*}*

*}*

*rBody.velocity = new Vector2(Mathf.Clamp(rBody.velocity.x, ForceClamp \* -1, ForceClamp), rBody.velocity.y);*

*if (Input.GetKey(KeyCode.Space) && grounded)*

*{*

*rBody.AddForce(Vector2.up \* JumpForce, ForceMode2D.Impulse);*

*grounded = false;*

*animator.SetBool("Landed", false);*

*}*

*if (Input.GetKey(KeyCode.LeftShift))*

*{*

*//print("Atempting to dash");*

*if (Time.time >= (lastDash + dashCooldown))*

*{*

*AttemptToDash();*

*}*

*}*

*CheckDash();*

*animator.SetFloat("YVelocity", rBody.velocity.y);*

*}*

**Table 1.** Movement controller Update code



**Figure 2.** Player dashing to the right

*private void AttemptToDash()*

*{*

*isDashing = true;*

*dashTimeLeft = dashTime;*

*lastDash = Time.time;*

*print("Atempting to dash");*

*PlayerAfterImagePool.Instance.GetFromPool();*

*lastImageXpos = transform.position.x;*

*}*

*private void CheckDash()*

*{*

*if (isDashing)*

*{*

*if(dashTimeLeft > 0)*

*{*

*rBody.velocity = new Vector2(dashSpeed \* facingDirection, rBody.velocity.y);*

*dashTimeLeft -= Time.deltaTime;*

*if (Mathf.Abs(transform.position.x - lastImageXpos) > distanceBetweenImages)*

*{*

*PlayerAfterImagePool.Instance.GetFromPool();*

*lastImageXpos = transform.position.x;*

*}*

*}*

*if(dashTimeLeft <= 0)*

*{*

*isDashing = false;*

*}*

*}*

*}*

**Table 2.**  Dash velocity and cooldown code

## **Task #2. *Basic HUD (HP), with a way to test if it works***

Description of the implementation (3-5 sentences). *The point of this task was to create a starting point for the heads up display (HUD) of the game. For now the player HP was connected to a HP bar. And to test if it works a NPC was used that modifies HP level of the player when a collision happens. The implementation is simple, the UI elements receive the player gameObject that is assigned by hand in the editor. On update the UI element then updates its values depending on the players state.*





**Figure 2.** HP UI showcase, the values are dynamic

*public void Start()*

*{*

*slider = GetComponent<Slider>();*

*HUDController = this.transform.parent.gameObject.GetComponent<HUDController>();*

*player = HUDController.Player.GetComponent<Player>();*

*slider.maxValue = player.MaxHealth;*

*}*

*public void Update()*

*{*

*Debug.Log(player.CurrentHealth);*

*slider.value = player.CurrentHealth;*

*HPTextValue.text = $"{player.CurrentHealth} / {player.MaxHealth}";*

*}*

**Table 3.** The code used to update UI elements to the respected values

## **Task #3. *Main character’s attack functionality (animations, basic logic)***

Description of the implementation (3-5 sentences). *I wanted to make our character be able to have at least the simplest attack functionality, therefore i used the prefabs that were at our disposal to create an animation. I needed to also create an attackingPoint which would scan for enemies that were in range for our attack.. For it to make sense i further needed to create something our character would attack - enemies - and make them interact with our character.*



**Figure 4.** Animation of attacking

*void Update()*

*{*

*if (Input.GetKeyDown(KeyCode.Mouse0))*

*{*

*Attack();*

*}*

*}*

*void Attack()*

*{*

*// Play attack animation*

*animator.SetTrigger("Attack");*

*// Detect enemies that are in range of attack*

*Collider2D[] hitEnemies = Physics2D.OverlapCircleAll(attackPoint.position, attackRange, enemyLayers);*

*// Damage them*

*foreach(Collider2D enemy in hitEnemies)*

*{*

*Enemy comp = enemy.GetComponent<Enemy>();*

*if (comp != null)*

*{*

*enemy.GetComponent<Enemy>().TakeDamage(attackDamage);*

*}*

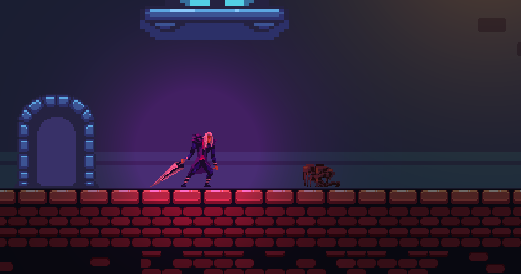
*}*

*}*

**Table 4.** Code responsible for attacking enemies

## **Task #4. *Basic enemy interaction (taking damage, dying, animations)***

Description of the implementation (3-5 sentences). *In order to test our attacking functionality, I created a new enemy - skeleton for which I also made a spawn animation. Our attacks would inflict damage to him and upon taking more damage than his maximum health he would die. Also we created an script follow script for the enemy and adjusted so that whenever the enemy collides with the player it prints out a log detecting the hit.*

**

**Figure 5.** Mob spawning animation



**Figure 6.** Animation of enemy getting killed and disappearing

*void Start()*

*{*

*animator = GetComponent<Animator>();*

*currentHealth = maxHealth;*

*m\_Rigidbody2D = GetComponent<Rigidbody2D>();*

*player = GameObject.FindGameObjectWithTag("Player").GetComponent<Rigidbody2D>().transform;*

*}*

*private void FixedUpdate()*

*{*

*FollowPLayer();*

*}*

*private void LateUpdate()*

*{*

*SpriteFlip();*

*}*

*private void FollowPLayer()*

*{*

*if (player.position.x > transform.position.x + stoppingDistance)*

*{*

*if (m\_Rigidbody2D.velocity.x < max\_walking\_velocity)*

*m\_Rigidbody2D.AddForce(new Vector2(force \* Time.deltaTime, 0f));*

*}*

*else if (player.position.x < transform.position.x - stoppingDistance)*

*{*

*//Debug.Log(m\_Rigidbody2D.velocity);*

*if (m\_Rigidbody2D.velocity.x > -max\_walking\_velocity)*

*m\_Rigidbody2D.AddForce(new Vector2(-force \* Time.deltaTime, 0f));*

*}*

*}*

*public void TakeDamage(int damage)*

*{*

*currentHealth -= damage;*

*animator.SetTrigger("Hurt");*

*// Play hurt animation*

*if (currentHealth <= 0)*

*{*

*Die();*

*}*

*}*

*private void OnCollisionEnter2D(Collision2D collision)*

*{*

*if(collision.collider.tag == "Player")*

*{*

*Debug.Log("Player attacked");*

*}*

*}*

*void Die()*

*{*

*Debug.Log("Enemy died!");*

*// Die animation*

*animator.SetBool("IsDead", true);*

*// Disable the enemy*

*GetComponent<Collider2D>().enabled = false;*

*this.enabled = false;*

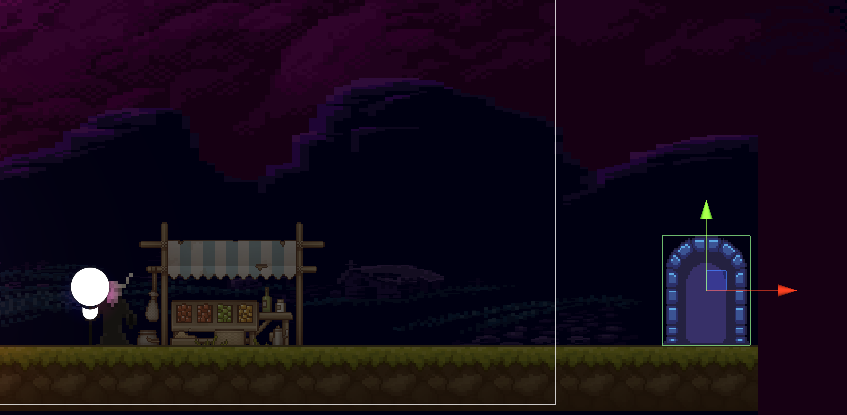
*Destroy(gameObject, animator.GetCurrentAnimatorStateInfo(0).length - 0.25f);*

*}*

**Table 5.** Code responsible for enemy taking damage, following the player and dying

## **Task #5. *Changing scenes***

Description of the implementation (3-5 sentences). *During the game our character will enter many different levels, for which scene loading is needed. Using a door sprite as a portal, I added a box collider with a trigger upon touching which, the character would be transported to my dungeon scene.*



**Figure 7.** Door sprite upon colliding with which, the user is taken to the dungeon’s entrance scene

*private void OnTriggerEnter2D(Collider2D collision)*

*{*

*GameObject collisionGameObject = collision.gameObject;*

*if (collisionGameObject.name == "Player")*

*{*

*LoadScene();*

*}*

*}*

*void LoadScene()*

*{*

*if (useIntegerToLoadLevel)*

*{*

*SceneManager.LoadScene(iLevelToLoad);*

*}*

*else*

*{*

*SceneManager.LoadScene(sLevelToLoad);*

*}*

*}*

**Table 6.** Code responsible for attacking enemies

## Task #6. *Combat action - fireball*

Description of the implementation (3-5 sentences). *During the game players may cast fireball at their enemies which inflicts damage.*

**

**Figure 8.** Door sprite upon colliding with which, the user is taken to the dungeon’s entrance scene

*void OnTriggerEnter2D(Collider2D hitInfo)*

*{*

*Enemy enemy = hitInfo.GetComponent<Enemy>();*

*if(hitInfo.tag == "Enemy")*

*{*

*enemy.TakeDamage(Damage);*

*}*

*if(enemy != null)*

*{*

*enemy.TakeDamage(Damage);*

*}*

*Destroy(gameObject);*

*}*

*}*

**Table 7.** Code responsible for ranged attack

## Task #7. *Defence task - dash scatter fireball glow*

*For the defence we were tasked with developing a fireball scatter attack after the dash, and to apply a glow effect to the fireball.*

**

**Figure 8.** Door sprite upon colliding with which, the user is taken to the dungeon’s entrance scene

*public void Scatter()*

*{*

*Instantiate(fireBallPrefab, firePointUp.position, firePointUp.rotation);*

*Instantiate(fireBallPrefab, firePointDown.position, firePointDown.rotation);*

*Instantiate(fireBallPrefab, firePointLeft.position, firePointLeft.rotation);*

*Instantiate(fireBallPrefab, firePointRight.position, firePointRight.rotation);*

*Instantiate(fireBallPrefab, firePointUL.position, firePointUL.rotation);*

*Instantiate(fireBallPrefab, firePointUR.position, firePointUR.rotation);*

*Instantiate(fireBallPrefab, firePointDL.position, firePointDL.rotation);*

*Instantiate(fireBallPrefab, firePointDR.position, firePointDR.rotation);*

*}*

**Table 8.** Code responsible for scattering fireballs

Laboratory work #2

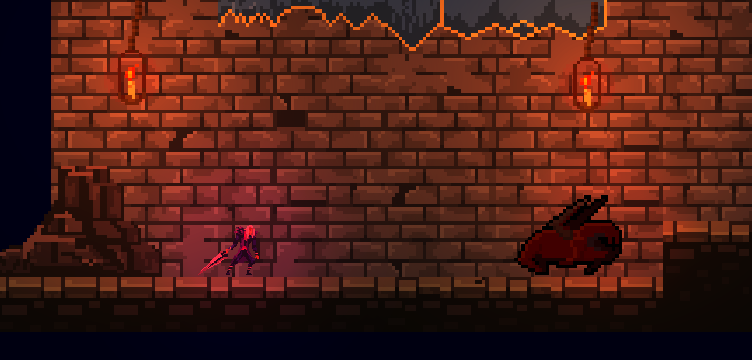
**List of tasks** (main functionality of your project)

1. Expanding world environment, animating enemies with particle effects (Justas)
2. Experiment with different physics (Justas)
3. Background, invisible walls, segway to new level, light effect on mobs (Justas)
4. Expand the game by adding a new level (Gytautas)
5. Add a trapdoor (Gytautas)

# **Solution**

## **Task #1. *Expanding world environment, animating enemies with particle effects***

Description of implementation (3-5 sentences).  *Add a new enemy, which would scope out the surroundings and patrol an area, animate his movement, add animations for getting hit (knocked back), add particles on hit and different particles upon death.*



**Figure 1.** A new type of patrol mob scouting the area



**Figure 2.** Blood particles splatter on the walls

****

**Figure 3. Mob dies and blood particles splatter on the floor**

PlayerCombat.cs:

*void Attack()*

*{*

*// Play attack animation*

*animator.SetTrigger("Attack");*

*// Detect enemies that are in range of attack*

*Collider2D[] hitEnemies = Physics2D.OverlapCircleAll(attackPoint.position, attackRange, enemyLayers);*

*attackDetails[0] = 40;*

*attackDetails[1] = transform.position.x;*

*// Damage them*

*foreach(Collider2D enemy in hitEnemies)*

*{*

*Enemy comp = enemy.GetComponent<Enemy>();*

*if (comp != null)*

*{*

*enemy.GetComponent<Enemy>().TakeDamage(attackDamage);*

*}*

*enemy.transform.parent.SendMessage("Damage", attackDetails);*

*}*

*}*

BasicEnemyController.cs:

*private void Damage(float[] attackDetails)*

*{*

*currentHealth -= attackDetails[0];*

*Instantiate(hitParticle, alive.transform.position, Quaternion.Euler(0.0f, 0.0f, Random.Range(0.0f, 360.0f)));*

*if(attackDetails[1] > alive.transform.position.x)*

*{*

*damageDirection = -1;*

*}*

*else*

*{*

*damageDirection = 1;*

*}*

*//Hit particle*

*if(currentHealth > 0.0f)*

*{*

*SwitchState(State.Knockback);*

*}*

*else if(currentHealth <= 0.0f)*

*{*

*SwitchState(State.Dead);*

*}*

*}*

**Table 4. Code fragment of player attacking and mob getting hurt and displaying animation**

## **Task #2. *Experiment with different physics***

Description of implementation (3-5 sentences). *Added an area upon entering which the player gets slowed down. Idea behind it was to use tilemap to create a new type of platform. Then we added a TileCollider onto it, which would let our player walk on it and write a script which would slow down the player’s movement speed upon colliding with it.*



**Figure 4.** A player walking on ground which slows him down



**Figure 5.** Tilemap of map elements upon colliding with which, the user is slowed down

*public PlayerData pd;*

*public GameObject player;*

*public bool useIntegerToLoadLevel = false;*

*void Start()*

*{*

*player = GameObject.FindWithTag("Player");*

*}*

*private void OnCollisionEnter2D(Collision2D collision)*

*{*

*player = collision.gameObject;*

*if (player.name == "Player")*

*{*

*pd.movementVelocity = 5.0f;*

*}*

*}*

*private void OnCollisionExit2D(Collision2D collision)*

*{*

*player = collision.gameObject;*

*if (player.name == "Player")*

*{*

*pd.movementVelocity = 10.0f;*

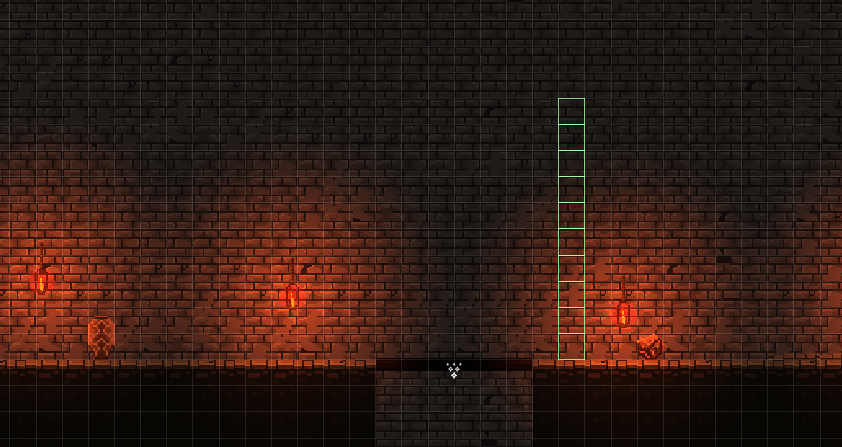
*}*

*}*

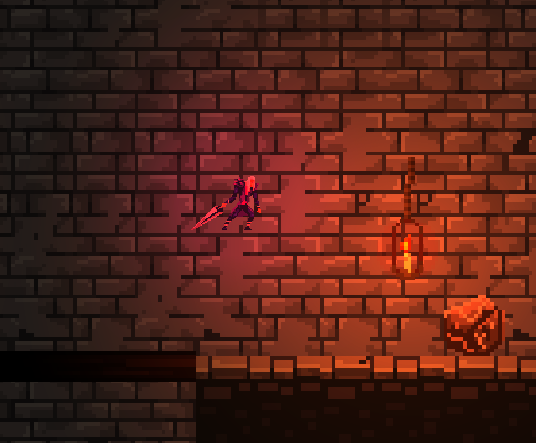
**Table 5. Fragment of code responsible for slowing down player’s movement speed upon enter and restoring back to normal upon leaving**

## **Task #3. *Background, invisible walls, segway to new level, light effect on mobs***

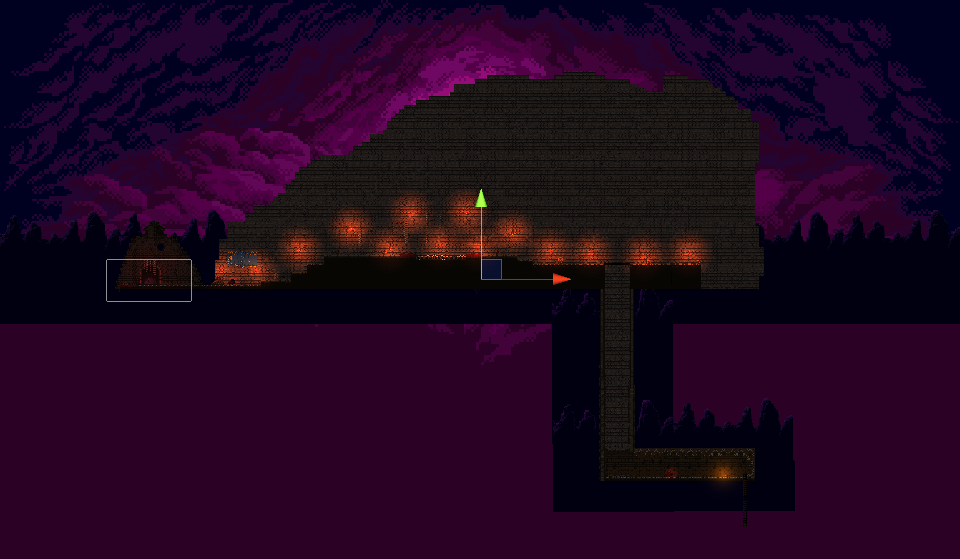
Description of implementation (3-5 sentences). *Add skybox(background) to the world so the background would look pretty. Also add invisible walls which would prevent player from leaving map boundaries (Escape level, right before trapdoor down), add entry segway to Escape level (from Entrance level).*



**Figure 6.** Tilemap of invisible wall



**Figure 7.** Player colliding with invisible wall, which he cannot pass



**Figure 8.** Added a custom background

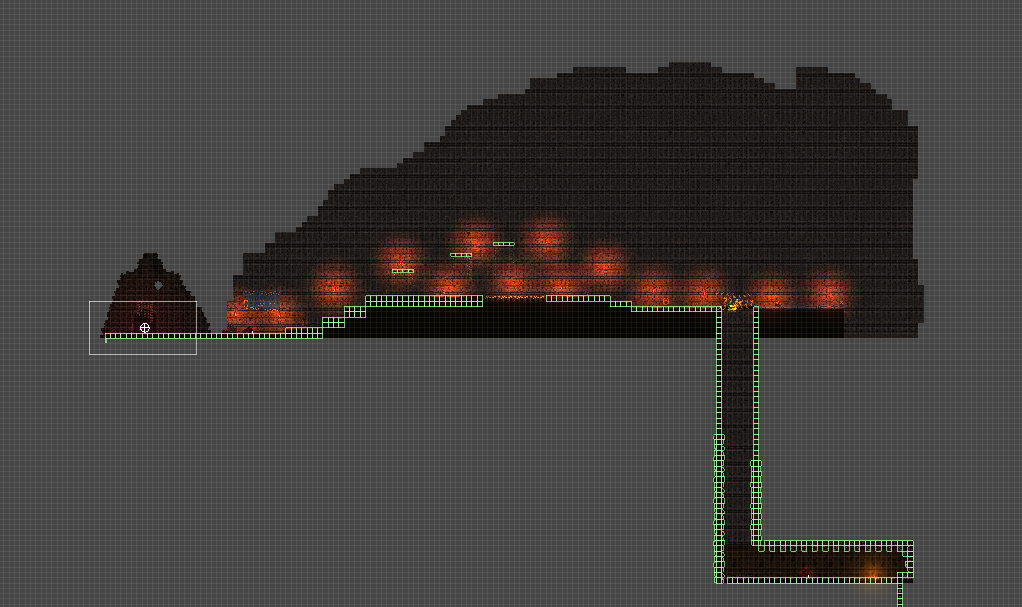
*Expand the game by creating and designing a new level, experiment with tilemaps, tilemap colliders, add a trapdoor which stepped upon breaks, releases particle effects and emits a sound*

## **Task #4. *Expand the game by adding a new level***

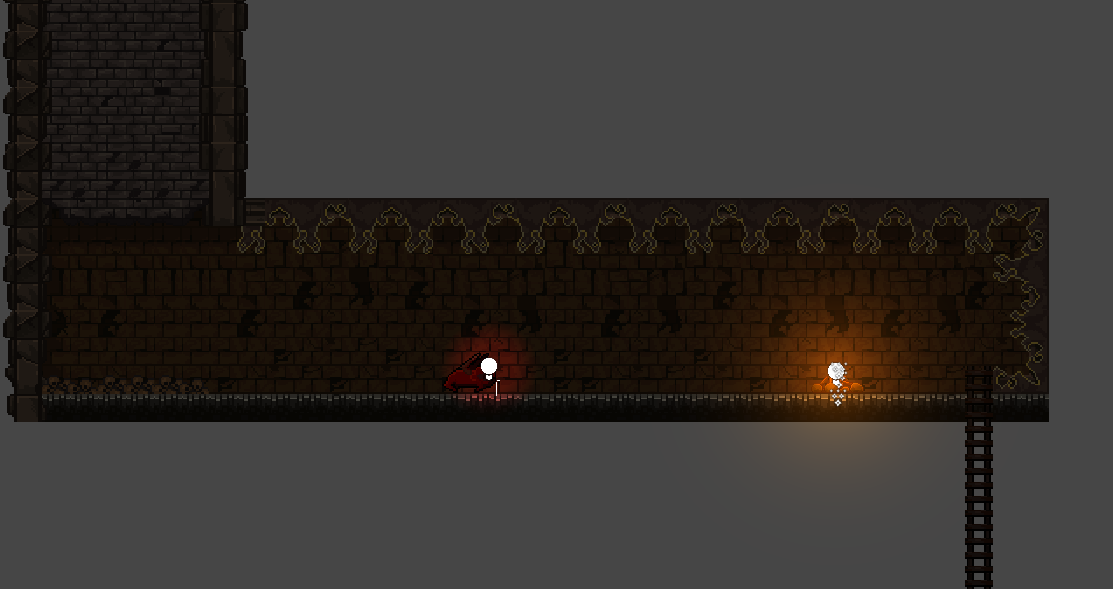
Description of implementation (3-5 sentences). *Expand the game by creating and designing a new level, called Escape. For this task I used Tilemap combined with Tilemap renderer and Tilemap Collider 2D where needed. For lighting, I used torches with Light 2D components, also added the pulsating lights script.*

****

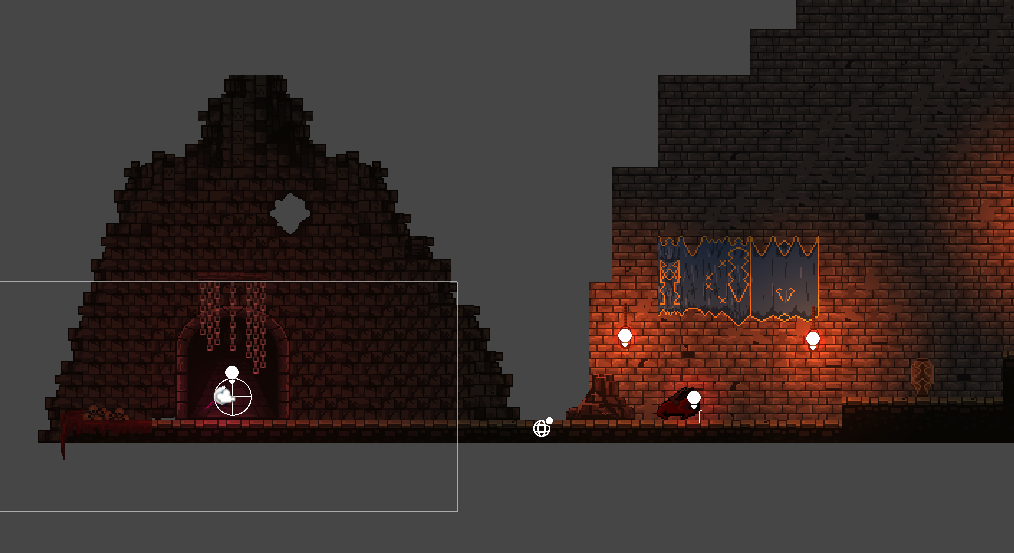
**Figure 9.** Primitive level layout



**Figure 10.** A specific layer of tilemap which is responsible for defining boundaries which prevent user from accessing unneeded areas



**Figure 11.** Area to which a user falls into upon stepping a trapdoor, which holds a ladder that will lead to another level



**Figure 12.** Starting area where the player appears once he enters the “Escape” level

## 

## **Task #5. *Add a trapdoor.***

Description of implementation (3-5 sentences). *Add a trapdoor which stepped upon breaks, releases particle effects and emits a sound. Trapdoor required a sprite, a boxcollider component, to catch the event of Player stepping on it plus a script for destroying it and playing a sound.*

**Figure 11.** A trapdoor with particle effects that get triggered upon Player’s collision with it. If the player steps on the trapdoor, it breaks, emits an explosion sound and creates particle effects.

*ParticleSystem ps;*

*// Start is called before the first frame update*

*void Start()*

*{*

*ps = this.GetComponent<ParticleSystem>();*

*var em = ps.emission;*

*em.enabled = false;*

*}*

*private void OnTriggerEnter2D(Collider2D collision)*

*{*

*var em = ps.emission;*

*em.enabled = true;*

*this.GetComponent<SpriteRenderer>().enabled = false;*

*this.GetComponent<AudioSource>().Play();*

*}*

**Table 6. Fragment of code responsible for trapdoor functions - breaking, emitting particles and a sound.**

## **Task #6. *Defence task***

Description of implementation (3-5 sentences). *Upon colliding with enemy, remove all light sources and elements from view in the world and just leave the player and the enemy highlighted.*



**Figure 13.** Before colliding with enemy



**Figure 13.** After colliding with enemy

*using System.Collections;*

*using System.Collections.Generic;*

*using UnityEngine;*

*using UnityEngine.Tilemaps;*

*using Light2DE = UnityEngine.Experimental.Rendering.Universal.Light2D;*

*public class Dimmer : MonoBehaviour*

*{*

*public Light2DE OnHit;*

*private void OnCollisionEnter2D(Collision2D collision)*

*{*

*if (collision.gameObject.tag == "Enemy")*

*{*

*foreach (Light2DE light in Resources.FindObjectsOfTypeAll(typeof(Light2DE))) {*

*if (light == OnHit)*

*{*

*continue;*

*}*

*light.enabled = false;*

*}*

*foreach (SpriteRenderer sr in Resources.FindObjectsOfTypeAll(typeof(SpriteRenderer)))*

*{*

*if (sr.gameObject == collision.gameObject || this.gameObject == sr.gameObject)*

*{*

*continue;*

*}*

*sr.enabled = false;*

*}*

*foreach (TilemapRenderer tr in Resources.FindObjectsOfTypeAll(typeof(TilemapRenderer)))*

*{*

*tr.enabled = false;*

*}*

*OnHit.enabled = true;*

*}*

*}*

*}*

Laboratory work #3

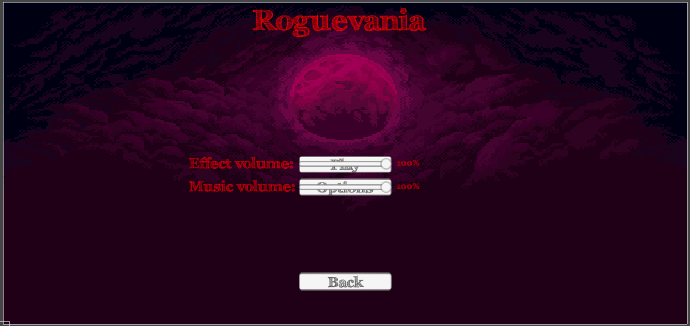
**List of tasks** (main functionality of your project)

1. Game menu – Gytautas
2. Audio – Gytautas
3. Bonfires – Gytautas
4. Game over – Gytautas
5. Functional shop - Jokūbas

# **Solution**

## **Task #1. *Game menu***

Description of implementation (3-5 sentences). *Separate scene that contains a canvas object that has all the UI elements. A script object that controls all UI buttons. Audio mixer reference for controlling volume.*



**Figure 7.** Editor mode game menu screenshot

*public class MenuHandler : MonoBehaviour*

*{*

*public GameObject MainPanel;*

*public GameObject OptionPanel;*

*public AudioClip OnClickSound;*

*public AudioMixer Mixer;*

*private Text EffectLabel;*

*private Text MusicLabel;*

*private AudioSource Audio;*

*public void Start()*

*{*

*EffectLabel = GameObject.Find("Options/EffectText").GetComponent<Text>();*

*MusicLabel = GameObject.Find("Options/MusicText").GetComponent<Text>();*

*OptionPanel.SetActive(false);*

*Audio = this.GetComponent<AudioSource>();*

*}*

*public void OnOptionsClick()*

*{*

*MainPanel.SetActive(false);*

*OptionPanel.SetActive(true);*

*Audio.PlayOneShot(OnClickSound);*

*}*

*public void OnOptionsBackClick()*

*{*

*MainPanel.SetActive(true);*

*OptionPanel.SetActive(false);*

*Audio.PlayOneShot(OnClickSound);*

*}*

*public void EffectVolumeChange(System.Single value)*

*{*

*EffectLabel.text = value.ToString() + "%";*

*if (value == 0)*

*{*

*Mixer.SetFloat("EffectVolume", -80);*

*}*

*else*

*{*

*Mixer.SetFloat("EffectVolume", Mathf.Log10(value / 100) \* 20);*

*}*

*}*

*public void MusicVolumeChange(System.Single value)*

*{*

*MusicLabel.text = value.ToString() + "%";*

*if (value == 0)*

*{*

*Mixer.SetFloat("MusicVolume", -80);*

*}*

*else*

*{*

*Mixer.SetFloat("MusicVolume", Mathf.Log10(value / 100) \* 20);*

*}*

*}*

*public void OnPlayClick()*

*{*

*Audio.PlayOneShot(OnClickSound);*

*SceneManager.LoadScene("Hub");*

*}*

*public void OnExitClick()*

*{*

*Audio.PlayOneShot(OnClickSound);*

*Application.Quit();*

*}*

*}*

## **Task #2. *Audio***

Description of implementation (3-5 sentences). *Create an audio mixer with two groups, one for music, one for sound effects. Added multiple audio sources (player, enemy, scene), and assigned them to a audio mixer group. Added ambient music and audio clips from the internet, most of them are played as one shots.*



**Figure 8.** Audio mixer configuration

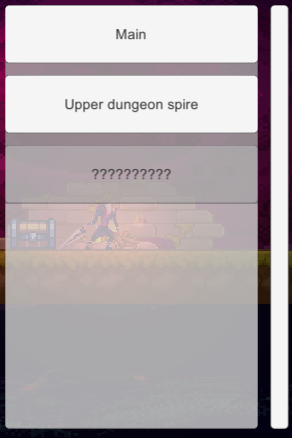


**Figure 9.** Example audio source, that plays ambient music on the Music audio mixer group

*player.AudioSource.PlayOneShot(player.playerData.AudioOption("Attack").Audio);*

## **Task #3. *Bonfires***

Description of implementation (3-5 sentences). *The main way to interact with different game areas. Bonfires provide a list of locations that the player has discovered and when clicking on it the player is teleported to the selected bonfire.*



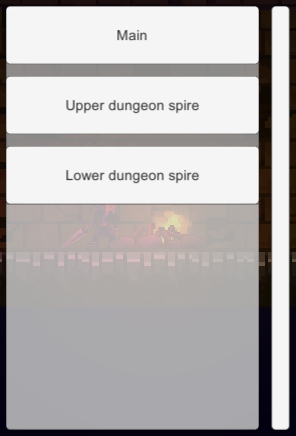
**Figure 9.** The initial bonfire list with one bonfire undiscovered



**Figure 10.** Unlit bonfire



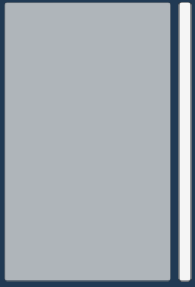
**Figure 11.** After interacting with the bonfire it is now lit



**Figure 12.** The location is now no longer greyed out and can be teleported from and to

The system consists of 3 main elements, the actual bonfire, the bonfire UI and the teleportation.

UI:



**Figure 13.** The empty bonfire UI window

*public class BonfireDropdown : MonoBehaviour*

*{*

*public GameObject OptionButton;*

*public AudioClip OnInteract;*

*void Start()*

*{*

*GenerateOptions();*

*}*

*public void GenerateOptions()*

*{*

*foreach (Transform child in this.transform)*

*{*

*Destroy(child.gameObject);*

*}*

*foreach (BonfireLocation l in BonfireGameState.Locations)*

*{*

*GameObject go = Instantiate(OptionButton, this.transform);*

*var button = go.GetComponent<UnityEngine.UI.Button>();*

*if (l.IsLit)*

*{*

*button.interactable = true;*

*button.GetComponentInChildren<Text>().text = l.LocationName;*

*button.onClick.AddListener(() => BonfireSelected(l));*

*}*

*else*

*{*

*button.interactable = false;*

*button.GetComponentInChildren<Text>().text = "??????????";*

*}*

*}*

*}*

*public void BonfireSelected(BonfireLocation bonfire)*

*{*

*Debug.Log("Warping to " + bonfire.LocationName);*

*BonfireGameState.BonfireLocation = bonfire.LocationName;*

*this.GetComponent<AudioSource>().outputAudioMixerGroup = AudioState.EffectGroup;*

*this.GetComponent<AudioSource>().PlayOneShot(OnInteract);*

*StartCoroutine(LoadScene(bonfire));*

*}*

*IEnumerator LoadScene(BonfireLocation bonfire)*

*{*

*yield return new WaitForSeconds(OnInteract.length - 1.0f);*

*AsyncOperation asyncOperation = SceneManager.LoadSceneAsync(bonfire.Scene);*

*asyncOperation.allowSceneActivation = false;*

*while (!asyncOperation.isDone)*

*{*

*if (asyncOperation.progress >= 0.9f)*

*{*

*asyncOperation.allowSceneActivation = true;*

*}*

*yield return null;*

*}*

*}*

*}*

Bonfire object:

*public class Bonfire : MonoBehaviour*

*{*

*int LocationIndex = -1;*

*bool IsLit = false;*

*GameObject LightEntity = null;*

*GameObject UI;*

*// Start is called before the first frame update*

*void Start()*

*{*

*LightEntity = this.transform.Find("Lights").gameObject;*

*for (int i = 0; i < BonfireGameState.Locations.Count; i++)*

*{*

*if (BonfireGameState.Locations[i].Scene == SceneManager.GetActiveScene().name)*

*{*

*if (BonfireGameState.Locations[i].NameInScene == this.gameObject.name)*

*{*

*LocationIndex = i;*

*break;*

*}*

*}*

*}*

*UI = GameObject.Find("BonfireUI");*

*UI.GetComponent<Canvas>().enabled = false;*

*}*

*// Update is called once per frame*

*void Update()*

*{*

*LightEntity.SetActive(BonfireGameState.Locations[LocationIndex].IsLit);*

*}*

*public void OnInteraction()*

*{*

*Debug.Log("Interacted with bonfire");*

*// On first interaction light the bonfire*

*if (!BonfireGameState.Locations[LocationIndex].IsLit)*

*{*

*BonfireGameState.Locations[LocationIndex].IsLit = true;*

*// Regenerate UI*

*GameObject go = UI.transform.Find("OptionsPanel/Grid").gameObject;*

*go.GetComponent<BonfireDropdown>().GenerateOptions();*

*}*

*// Show/Hide UI*

*if (UI.GetComponent<Canvas>().enabled)*

*{*

*UI.GetComponent<Canvas>().enabled = false;*

*}*

*else*

*{*

*UI.GetComponent<Canvas>().enabled = true;*

*}*

*}*

*public void OnLeave()*

*{*

*UI.GetComponent<Canvas>().enabled = false;*

*}*

*}*

Teleportation:

*public class BonfireTransition : MonoBehaviour*

*{*

*void Start()*

*{*

*if (BonfireGameState.BonfireLocation != "")*

*{*

*// Find the bonfire game object*

*for(int i = 0; i < BonfireGameState.Locations.Count; i++)*

*{*

*if (BonfireGameState.Locations[i].LocationName == BonfireGameState.BonfireLocation)*

*{*

*Debug.Log("Positioning at " + BonfireGameState.BonfireLocation);*

*GameObject go = GameObject.Find(BonfireGameState.Locations[i].NameInScene);*

*this.transform.position = go.transform.position + new Vector3(0, 2, 0);*

*return;*

*}*

*}*

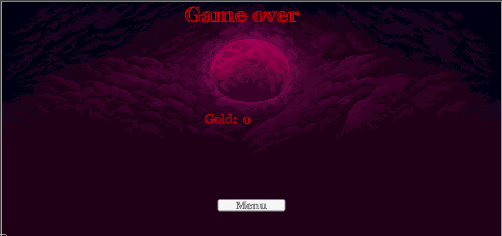
*}*

*}*

*}*

## **Task #4. *Game over***

Description of implementation (3-5 sentences). *Separate game over scene. Its loaded when ever the player hp falls bellow 0. The scene has a navigation button to go back to the menu.*



**Figure 14.** Game over scene view

UI controller:

*public class GameOverHandler : MonoBehaviour*

*{*

*public AudioClip OnClickSound;*

*public GameObject GoldText;*

*public PlayerData Data;*

*private AudioSource Audio;*

*void Start()*

*{*

*Audio = this.GetComponent<AudioSource>();*

*GoldText.GetComponent<Text>().text = Data.Gold.ToString();*

*}*

*public void OnMenuClick()*

*{*

*Audio.PlayOneShot(OnClickSound);*

*SceneManager.LoadScene("Menu");*

*}*

*}*

State checker:

*public class HPController : MonoBehaviour*

*{*

*private HUDController HUDController;*

*private Slider slider;*

*private PlayerData player;*

*public Text HPTextValue;*

*public void Start()*

*{*

*slider = GetComponent<Slider>();*

*HUDController = this.transform.parent.gameObject.GetComponent<HUDController>();*

*player = HUDController.Player.GetComponent<Player>().playerData;*

*slider.maxValue = player.MaxHealth;*

*}*

*public void Update()*

*{*

*//Debug.Log(player.CurrentHealth);*

*slider.value = player.CurrentHealth;*

*HPTextValue.text = $"{player.CurrentHealth} / {player.MaxHealth}";*

*if (player.CurrentHealth <= 0)*

*{*

*// Game over*

*SceneManager.LoadScene("GameOver");*

*}*

*}*

*}*

## **Task #5. *Boss scene***

Description of implementation (3-5 sentences). *Created a new area(scene) that is designed to have a game-finishing boss fight wit the Magnificent Slatt.*

*A picture containing light, lit, dark

Description automatically generated*

**Figure 15.** Boss area layout

*A picture containing light, brick, lit, dark

Description automatically generated*

**Figure 16.** Boss area layout

*A picture containing dark, light, lit, night

Description automatically generated*

**Figure 17.** Boss area layout

## **Task #6. *Boss mob***

Description of implementation (3-5 sentences). *Created a new type of mob (Boss) - the Magnificent Slatt. Intended to be the last boss of the game. Animated, added attacks, a HUD (simulating Dark Souls boss fights plus intense background music)*

*A screenshot of a video game

Description automatically generated with medium confidence*

**Figure 18.** Boss mob

*A screenshot of a video game

Description automatically generated*

**Figure 19.** Boss mob being hit by player

A screenshot of a video game

Description automatically generated

**Figure 20.** Player being hit by Boss

Graphical user interface

Description automatically generated **Figure 21.** Boss mob Animator states

Graphical user interface

Description automatically generated

**Figure 22.** Boss mob audio for “Epic” music

Mob.cs:

using System.Collections;

using UnityEngine;

[RequireComponent(typeof(CanvasGroup))]

public class Mobs : MonoBehaviour

{

private enum State

{

Moving,

Knockback,

Dead

}

private State currentState;

static int AnimatorWalk = Animator.StringToHash("Walk");

static int AnimatorAttack = Animator.StringToHash("Attack");

Animator \_animator;

[SerializeField]

private float

groundCheckDistance,

wallCheckDistance,

movementSpeed,

knockbackDuration;

public float

maxHealth;

public float

currentHealth;

private float

knockbackStartTime;

private int

facingDirection,

damageDirection;

[SerializeField]

private Transform

groundCheck,

wallCheck;

private Vector2 movement;

private bool

groundDetected,

wallDetected;

[SerializeField]

private LayerMask whatIsGround;

[SerializeField]

private Vector2 knockbackSpeed;

[SerializeField]

private GameObject

hitParticle,

deathChunkParticle,

deathBloodParticle;

private Rigidbody2D aliveRb;

public int damage = 5;

public int GoldWorth = 10;

public AudioClip AttackSound;

private void Update()

{

switch (currentState)

{

case State.Moving:

UpdateMovingState();

break;

case State.Knockback:

UpdateKnockbackState();

break;

case State.Dead:

UpdateDeadState();

break;

}

}

void Start()

{

aliveRb = GetComponent<Rigidbody2D>();

\_animator = GetComponent<Animator>();

currentHealth = maxHealth;

facingDirection = -1;

}

private void EnterMovingState()

{

}

private void UpdateMovingState()

{

groundDetected = Physics2D.Raycast(groundCheck.position, Vector2.down, groundCheckDistance, whatIsGround);

wallDetected = Physics2D.Raycast(wallCheck.position, transform.right, wallCheckDistance, whatIsGround);

if (wallDetected)

{

Flip();

}

else

{

// Target the player

GameObject player = GameObject.FindGameObjectWithTag("Player");

if (player.transform.position.x < this.transform.position.x)

{

if (facingDirection != -1)

{

Flip();

}

}

else

{

if (facingDirection != 1)

{

Flip();

}

}

movement.Set(movementSpeed \* facingDirection, aliveRb.velocity.y);

aliveRb.velocity = movement;

}

}

private void ExitMovingState()

{

}

private void EnterKnockbackState()

{

knockbackStartTime = Time.time;

movement.Set(knockbackSpeed.x \* damageDirection \* 2, knockbackSpeed.y);

aliveRb.velocity = movement;

//aliveAnim.SetBool("Knockback", true);

}

private void UpdateKnockbackState()

{

if (Time.time >= knockbackStartTime + knockbackDuration)

{

SwitchState(State.Moving);

}

}

private void ExitKnockbackState()

{

//aliveAnim.SetBool("Knockback", false);

}

private void EnterDeadState()

{

Instantiate(deathChunkParticle, transform.position, deathChunkParticle.transform.rotation);

Instantiate(deathBloodParticle, transform.position, deathBloodParticle.transform.rotation);

Destroy(gameObject);

}

private void UpdateDeadState()

{

}

private void ExitDeadState()

{

}

private void OnPlayerHit(Player player)

{

currentHealth -= player.playerData.AttackDamage;

Vector3 position = transform.position;

position.y += 0.75f;

position.x += 1.0f \* facingDirection;

Instantiate(hitParticle, position, Quaternion.Euler(0.0f, 0.0f, Random.Range(0.0f, 360.0f)));

if (player.transform.position.x > transform.position.x)

{

damageDirection = -1;

}

else

{

damageDirection = 1;

}

//Hit particle

if (currentHealth > 0.0f)

{

SwitchState(State.Knockback);

}

else if (currentHealth <= 0.0f)

{

player.playerData.Gold += GoldWorth;

SwitchState(State.Dead);

}

Debug.Log("attacked for:" + player.playerData.AttackDamage.ToString());

Debug.Log("current mob health:" + currentHealth.ToString());

player.AudioSource.PlayOneShot(player.playerData.AudioOption("Attack").Audio);

}

private void Flip()

{

Debug.Log("Called flip");

facingDirection \*= -1;

transform.Rotate(0.0f, 180.0f, 0.0f);

}

IEnumerator Animate()

{

yield return new WaitForSeconds(5f);

while (true)

{

\_animator.SetBool(AnimatorWalk, true);

yield return new WaitForSeconds(1f);

\_animator.transform.localScale = new Vector3(30, 30, 1);

yield return new WaitForSeconds(1f);

\_animator.SetBool(AnimatorWalk, false);

yield return new WaitForSeconds(1f);

\_animator.SetTrigger(AnimatorAttack);

yield return new WaitForSeconds(1f);

\_animator.SetTrigger(AnimatorAttack);

yield return new WaitForSeconds(1f);

\_animator.SetTrigger(AnimatorAttack);

yield return new WaitForSeconds(5f);

}

}

private void SwitchState(State state)

{

switch (currentState)

{

case State.Moving:

ExitMovingState();

break;

case State.Knockback:

ExitKnockbackState();

break;

case State.Dead:

ExitDeadState();

break;

}

switch (state)

{

case State.Moving:

EnterMovingState();

break;

case State.Knockback:

EnterKnockbackState();

break;

case State.Dead:

EnterDeadState();

break;

}

currentState = state;

}

public void OnCollisionEnter2D(Collision2D collision)

{

if (collision.gameObject.tag == "Player")

{

Player p = collision.gameObject.GetComponent<Player>();

p.playerData.CurrentHealth -= damage;

// Knock back

p.RB.velocity = new Vector2(knockbackSpeed.x \* facingDirection, knockbackSpeed.y);

p.AudioSource.PlayOneShot(AttackSound);

\_animator.SetTrigger(AnimatorAttack);

}

}

}

## **Task #6. *Transition to boss area***

Description of implementation (3-5 sentences). *Added a transition from “Escape” scene to be taken to the final area – boss area.*

A picture containing text

Description automatically generated

**Figure 23.** Transition through ladder to boss area

A screenshot of a video game

Description automatically generated

**Figure 24.** On collision teleported to Boss area

SceneLoader.cs:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class SceneLoader : MonoBehaviour

{

// Start is called before the first frame update

public int iLevelToLoad;

public string sLevelToLoad;

public bool useIntegerToLoadLevel = false;

private void OnTriggerEnter2D(Collider2D collision)

{

GameObject collisionGameObject = collision.gameObject;

if (collisionGameObject.name == "Player")

{

LoadScene();

}

}

void LoadScene()

{

if (useIntegerToLoadLevel)

{

//DontDestroyOnLoad(inventory);

SceneManager.LoadScene(iLevelToLoad);

}

else

{

SceneManager.LoadScene(sLevelToLoad);

}

}

}

## **Task #7. *Boss HUD***

Description of implementation (3-5 sentences). *Added a HUD for boss’s health for it to simulate Dark Souls-like boss fight.*

*A screenshot of a video game

Description automatically generated*

**Figure 25.** HUD showing the health of The Magnificent Slatt

BossHPController.cs:

using UnityEngine.SceneManagement;

using UnityEngine.UI;

public class BossHPController : MonoBehaviour

{

public Text HPTextValue;

private Slider slider;

private GameObject Boss;

private float currBossHealth;

private float maxBossHealth;

void Start()

{

slider = GetComponent<Slider>();

Boss = GameObject.FindGameObjectWithTag("Boss");

currBossHealth = maxBossHealth = Boss.GetComponent<Mobs>().maxHealth;

slider.maxValue = currBossHealth;

}

// Update is called once per frame

void Update()

{

currBossHealth = slider.value = Boss.GetComponent<Mobs>().currentHealth;

HPTextValue.text = $"{currBossHealth} / {maxBossHealth}";

if (currBossHealth <= 0)

{

// Game over

HPTextValue.text = "0";

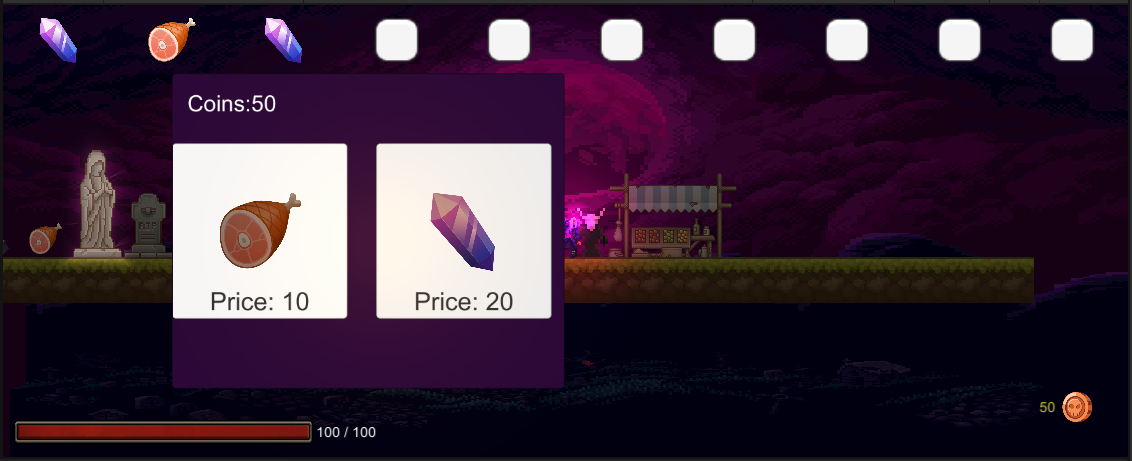
}

}

}

## **Task #8. *Functional shop.***

Description of implementation (3-5 sentences). *Added an interactive UI for shop, now items can be bought for gold and items transition into other scenes.*



**Figure 26.** Buying items in Hub scene.



**Figure 27.** Using items bought from the Hub scene.

ShopManagerScript.cs:

public int[,] shopItems = new int[5, 5];

public Text coinsTxt;

public Player player;

private PlayerData playerData;

public Inventory inventory;

// Start is called before the first frame update

void Start()

{

playerData = player.GetComponent<Player>().playerData;

//inventory = player.GetComponent<Inventory>();

//coinsTxt.text = "Coins:" + coins.ToString();

coinsTxt.text = "Coins:" + playerData.Gold;

//item IDs

shopItems[1, 1] = 1;

shopItems[1, 2] = 2;

shopItems[1, 3] = 3;

shopItems[1, 4] = 4;

//Price

shopItems[2, 1] = 10;

shopItems[2, 2] = 20;

shopItems[2, 3] = 30;

shopItems[2, 4] = 40;

//Quantity

shopItems[3, 1] = 0;

shopItems[3, 2] = 0;

shopItems[3, 3] = 0;

shopItems[3, 4] = 0;

}

// Update is called once per frame

public void Buy()

{

GameObject ButtonRef = GameObject.FindGameObjectWithTag("Event").GetComponent<EventSystem>().currentSelectedGameObject;

int itemID = ButtonRef.GetComponent<ButtonInfo>().ItemID;

if (playerData.Gold >= shopItems[2, itemID] && !inventory.IsFull())

{

playerData.Gold -= shopItems[2, itemID];

shopItems[3, itemID]++;

coinsTxt.text = "Coins:" + playerData.Gold;

ButtonRef.GetComponent<ButtonInfo>().QuantityTxt.text = "";//shopItems[3, ButtonRef.GetComponent<ButtonInfo>().ItemID].ToString();

GameObject shopItem = ButtonRef.GetComponent<ButtonInfo>().item;

GameObject newItem = Instantiate(shopItem);

inventory.AddItem(newItem);

}

else

{

if (inventory.IsFull())

{

coinsTxt.text = "Inventory Full!";

}

else

{

coinsTxt.text = "Insufficient funds!";

}

}

}

Inventory.cs:

public bool IsFull()

{

for(int i = 0; i < InventoryState.Inventory.Length; i++)

{

if (InventoryState.Inventory[i] == null)

{

return false;

}

}

return true;

}

InteractionObject.cs:

public void OpenShop()

{

// Interact with shop NPC

if (gameObject.name == "NPC")

{

GameObject shopUi = gameObject.transform.Find("Shop\_UI2").gameObject;

if (shopUi.activeSelf)

{

shopUi.SetActive(false);

}

else

{

Debug.Log(message);

shopUi.SetActive(true);

}

}

}

ButtonInfo.cs:

public int ItemID;

public Text PriceTxt;

public Text QuantityTxt;

public GameObject item;

public GameObject ShopManager;

void Update()

{

PriceTxt.text = "Price: " + ShopManager.GetComponent<ShopManagerScript>().shopItems[2, ItemID].ToString();

QuantityTxt.text = "";//"Quan.: " + ShopManager.GetComponent<ShopManagerScript>().shopItems[3, ItemID].ToString();

}

User's manual (for the Individual work defence)

**How to play?** *Aenean eu quam gravida, laoreet nisl eu, sagittis quam. Donec sit amet nunc nisi. Sed vel ipsum metus. Nullam accumsan vestibulum ex. Aenean eu quam gravida, laoreet nisl eu, sagittis quam. Donec sit amet nunc nisi. Sed vel ipsum metus. Nullam accumsan vestibulum ex.*

Screenshot

**Figure 10.** Screenshot #5

*Nunc vel enim vel magna interdum dapibus id nec nisl. Suspendisse elit augue, accumsan tempor erat sed, gravida suscipit urna. Duis blandit lacus et finibus finibus. Mauris pretium pharetra orci dictum luctus. Nullam commodo magna a tincidunt malesuada.*

Screenshot

**Figure 11.** Screenshot #5

*Sed sollicitudin justo erat, viverra luctus mi consequat non. Sed ut condimentum libero. Duis rutrum lacus ante, vitae feugiat ex faucibus at. Maecenas pulvinar et augue sed commodo.*

**Descriptions of the rules of the game**. Nunc quis condimentum lacus. Quisque felis neque, ullamcorper vel posuere eget, blandit non neque. Nam in varius erat. Duis molestie sit amet eros vel rhoncus. Nunc quis condimentum lacus. Quisque felis neque, ullamcorper vel posuere eget, blandit non neque. Nam in varius erat. Duis molestie sit amet eros vel rhoncus.

**Descriptions of the controls / keys.** Donec et lorem vitae ligula bibendum faucibus. Suspendisse interdum quis augue sed luctus. Curabitur ac diam augue. In hac habitasse platea dictumst. Curabitur maximus maximus tortor. Nunc quis condimentum lacus. Quisque felis neque, ullamcorper vel posuere eget, blandit non neque. Nam in varius erat. Duis molestie sit amet eros vel rhoncus.

Literature list

1. [*https://assetstore.unity.com/packages/2d/gui/icons/gui-parts-159068*](https://assetstore.unity.com/packages/2d/gui/icons/gui-parts-159068)
2. [*https://assetstore.unity.com/packages/2d/characters/static-sprites-20-118836*](https://assetstore.unity.com/packages/2d/characters/static-sprites-20-118836)
3. [*https://assetstore.unity.com/packages/2d/environments/pixel-art-platformer-village-props-166114*](https://assetstore.unity.com/packages/2d/environments/pixel-art-platformer-village-props-166114)
4. [*https://assetstore.unity.com/packages/2d/characters/gothicvania-cemetery-120509*](https://assetstore.unity.com/packages/2d/characters/gothicvania-cemetery-120509)
5. [*https://assetstore.unity.com/packages/2d/environments/medieval-pixel-art-asset-free-130131*](https://assetstore.unity.com/packages/2d/environments/medieval-pixel-art-asset-free-130131)
6. [*https://www.youtube.com/watch?v=5E5\_Fquw7BM*](https://www.youtube.com/watch?v=5E5_Fquw7BM)

ANNEX

## Movement.cs

*using System.Collections;*

*using System.Collections.Generic;*

*using UnityEngine;*

*public class Movement : MonoBehaviour*

*{*

*public float JumpForce = 1000.0f;*

*public float Speed = 1.0f;*

*[Range(0.0f, 1000.0f)]*

*public float ForceClamp = 0.0f;*

*public GameObject Ground;*

*private Rigidbody2D rBody;*

*private Animator animator;*

*private bool grounded = true;*

*void Start()*

*{*

*rBody = GetComponent<Rigidbody2D>();*

*animator = GetComponent<Animator>();*

*}*

*void OnCollisionEnter2D(Collision2D collision)*

*{*

*print("Collided");*

*if (collision.gameObject == Ground)*

*{*

*grounded = true;*

*animator.SetBool("Landed", true);*

*}*

*}*

*void Update()*

*{*

*float hForce = Input.GetAxisRaw("Horizontal") \* Speed \* Time.deltaTime;*

*if (hForce == 0.0f)*

*{*

*rBody.velocity = new Vector2(0, rBody.velocity.y);*

*animator.SetFloat("Speed", 0.0f);*

*}*

*else*

*{*

*rBody.AddForce(Vector2.right \* hForce);*

*animator.SetFloat("Speed", Mathf.Abs(hForce));*

*if (hForce > 0.0f)*

*{*

*transform.eulerAngles = new Vector3(0.0f, 0.0f, 0.0f);*

*}*

*else if (hForce < 0.0f)*

*{*

*transform.eulerAngles = new Vector3(0.0f, 180.0f, 0.0f);*

*}*

*}*

*rBody.velocity = new Vector2(Mathf.Clamp(rBody.velocity.x, ForceClamp \* -1, ForceClamp), rBody.velocity.y);*

*if (Input.GetKey(KeyCode.Space) && grounded)*

*{*

*rBody.AddForce(Vector2.up \* JumpForce, ForceMode2D.Impulse);*

*grounded = false;*

*animator.SetBool("Landed", false);*

*}*

*animator.SetFloat("YVelocity", rBody.velocity.y);*

*}*

*}*

## Player.cs

*using System.Collections;*

*using System.Collections.Generic;*

*using UnityEngine;*

*public class Player : MonoBehaviour*

*{*

*public int CurrentHealth = 100;*

*public int MaxHealth = 100;*

*}*

## HPController.cs

*using System.Collections;*

*using System.Collections.Generic;*

*using UnityEngine;*

*using UnityEngine.UI;*

*public class HPController : MonoBehaviour*

*{*

*private HUDController HUDController;*

*private Slider slider;*

*private Player player;*

*public Text HPTextValue;*

*public void Start()*

*{*

*slider = GetComponent<Slider>();*

*HUDController = this.transform.parent.gameObject.GetComponent<HUDController>();*

*player = HUDController.Player.GetComponent<Player>();*

*slider.maxValue = player.MaxHealth;*

*}*

*public void Update()*

*{*

*Debug.Log(player.CurrentHealth);*

*slider.value = player.CurrentHealth;*

*HPTextValue.text = $"{player.CurrentHealth} / {player.MaxHealth}";*

*}*

*}*

## HUDController.cs

*using System.Collections;*

*using System.Collections.Generic;*

*using UnityEngine;*

*public class HUDController : MonoBehaviour*

*{*

*public GameObject Player;*

*}*

## PulsatingLight.cs

*using System.Collections;*

*using System.Collections.Generic;*

*using UnityEngine;*

*using UnityEngine.Experimental.Rendering.Universal;*

*public class PulsatingLight : MonoBehaviour*

*{*

*Light2D lightToManipulate;*

*public float Interval = 1.0f;*

*[Range(0.0f, 10.0f)]*

*public float From = 0.0f;*

*[Range(0.0f, 10.0f)]*

*public float To = 1.0f;*

*bool up = true;*

*// Start is called before the first frame update*

*void Start()*

*{*

*lightToManipulate = GetComponent<Light2D>();*

*}*

*// Update is called once per frame*

*void Update()*

*{*

*if (up)*

*{*

*lightToManipulate.intensity += Interval \* Time.deltaTime;*

*if (lightToManipulate.intensity >= To)*

*{*

*up = false;*

*}*

*}*

*else*

*{*

*lightToManipulate.intensity -= Interval \* Time.deltaTime;*

*if (lightToManipulate.intensity <= From)*

*{*

*up = true;*

*}*

*}*

*}*

*}*

## NPC.cs

*using System.Collections;*

*using System.Collections.Generic;*

*using UnityEngine;*

*public class NPC : MonoBehaviour*

*{*

*private void OnTriggerEnter2D(Collider2D collision)*

*{*

*if (collision.gameObject.tag == "Player")*

*{*

*onPlayerTouch(collision.gameObject);*

*}*

*}*

*void onPlayerTouch(GameObject player)*

*{*

*player.GetComponent<Player>().CurrentHealth -= 10;*

*}*

*}*

## ObjectTracking.cs

*using System.Collections;*

*using System.Collections.Generic;*

*using UnityEngine;*

*public class ObjectTracking : MonoBehaviour*

*{*

*public GameObject Target;*

*private Vector3 offset;*

*// Start is called before the first frame update*

*void Start()*

*{*

*offset = transform.position - Target.transform.position;*

*}*

*// Update is called once per frame*

*void Update()*

*{*

*transform.position = Target.transform.position + offset;*

*}*

*}*

## Parallaxing.cs

*using System.Collections;*

*using System.Collections.Generic;*

*using UnityEngine;*

*public class Parallaxing : MonoBehaviour*

*{*

*// Entities that are affected*

*public Transform[] Backgrounds;*

*// Entity Z scale values*

*private float[] BGScales;*

*// Parallax movement speed*

*public float Smooth = 1.0f;*

*// MainCamera transform and position*

*private Transform camTransform;*

*private Vector3 prevCamPosition;*

*// Awake is called before Start()*

*void Awake()*

*{*

*camTransform = Camera.main.transform;*

*}*

*// Start is called before the first frame update*

*void Start()*

*{*

*prevCamPosition = camTransform.position;*

*BGScales = new float[Backgrounds.Length];*

*for (int i = 0; i < Backgrounds.Length; i++)*

*{*

*BGScales[i] = Backgrounds[i].position.z;*

*}*

*}*

*// Update is called once per frame*

*void Update()*

*{*

*for (int i = 0; i < Backgrounds.Length; i++)*

*{*

*float parallax = (prevCamPosition.x - camTransform.position.x) \* BGScales[i];*

*float bgTargetPosX = Backgrounds[i].position.x + parallax;*

*Vector3 bgTargetPos = new Vector3(bgTargetPosX, Backgrounds[i].position.y, Backgrounds[i].position.z);*

*Backgrounds[i].position = Vector3.Lerp(Backgrounds[i].position, bgTargetPos, Smooth \* Time.deltaTime);*

*}*

*prevCamPosition = camTransform.position;*

*}*

*}*

## PlayerCombat.cs (Justas)

*using System.Collections;*

*using System.Collections.Generic;*

*using UnityEngine;*

*public class PlayerCombat : MonoBehaviour*

*{*

*public Animator animator;*

*public Transform attackPoint;*

*public LayerMask enemyLayers;*

*public float attackRange = 0.5f;*

*public int attackDamage = 40;*

*// Update is called once per frame*

*void Update()*

*{*

*if (Input.GetKeyDown(KeyCode.Mouse0))*

*{*

*Attack();*

*}*

*}*

*void Attack()*

*{*

*// Play attack animation*

*animator.SetTrigger("Attack");*

*// Detect enemies that are in range of attack*

*Collider2D[] hitEnemies = Physics2D.OverlapCircleAll(attackPoint.position, attackRange, enemyLayers);*

*// Damage them*

*foreach(Collider2D enemy in hitEnemies)*

*{*

*Enemy comp = enemy.GetComponent<Enemy>();*

*if (comp != null)*

*{*

*enemy.GetComponent<Enemy>().TakeDamage(attackDamage);*

*}*

*}*

*}*

*private void OnDrawGizmosSelected()*

*{*

*if (attackPoint == null)*

*return;*

*Gizmos.DrawWireSphere(attackPoint.position, attackRange);*

*}*

*}*

## Enemy.cs (Justas)

*using System.Collections;*

*using System.Collections.Generic;*

*using UnityEngine;*

*public class Enemy : MonoBehaviour*

*{*

*private Animator animator;*

*public int maxHealth = 100;*

*int currentHealth;*

*// Start is called before the first frame update*

*void Start()*

*{*

*animator = GetComponent<Animator>();*

*currentHealth = maxHealth;*

*}*

*public void TakeDamage(int damage)*

*{*

*currentHealth -= damage;*

*animator.SetTrigger("Hurt");*

*// Play hurt animation*

*if (currentHealth <= 0)*

*{*

*Die();*

*}*

*}*

*void Die()*

*{*

*Debug.Log("Enemy died!");*

*// Die animation*

*animator.SetBool("IsDead", true);*

*// Disable the enemy*

*GetComponent<Collider2D>().enabled = false;*

*this.enabled = false;*

*Destroy(gameObject, animator.GetCurrentAnimatorStateInfo(0).length - 0.25f);*

*}*

*}*

## SceneLoader.cs (Justas)

*using System.Collections;*

*using System.Collections.Generic;*

*using UnityEngine;*

*using UnityEngine.SceneManagement;*

*public class SceneLoader : MonoBehaviour*

*{*

*// Start is called before the first frame update*

*public int iLevelToLoad;*

*public string sLevelToLoad;*

*public bool useIntegerToLoadLevel = false;*

*void Start()*

*{*

*}*

*// Update is called once per frame*

*void Update()*

*{*

*}*

*private void OnTriggerEnter2D(Collider2D collision)*

*{*

*GameObject collisionGameObject = collision.gameObject;*

*if (collisionGameObject.name == "Player")*

*{*

*LoadScene();*

*}*

*}*

*void LoadScene()*

*{*

*if (useIntegerToLoadLevel)*

*{*

*SceneManager.LoadScene(iLevelToLoad);*

*}*

*else*

*{*

*SceneManager.LoadScene(sLevelToLoad);*

*}*

*}*

*}*