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*

Kaunas, 2021

Tables of Contents

Tables of Images

Table of Tables/functions

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).*
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.*

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

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.*

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);
        }

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

Table 1. Movement controller Update 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 2. 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 3. 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 3. 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.*

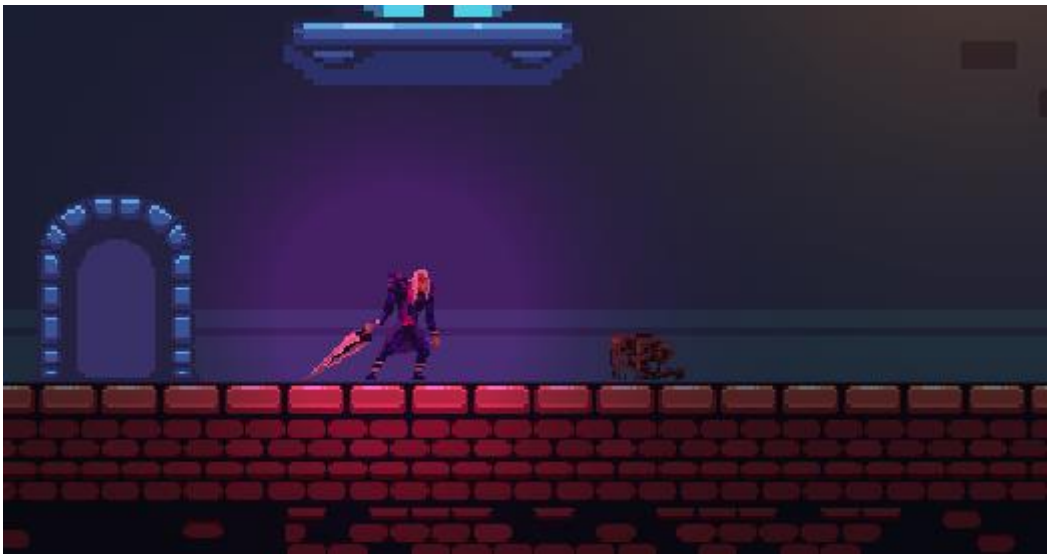

Figure 4. Mob spawning animation

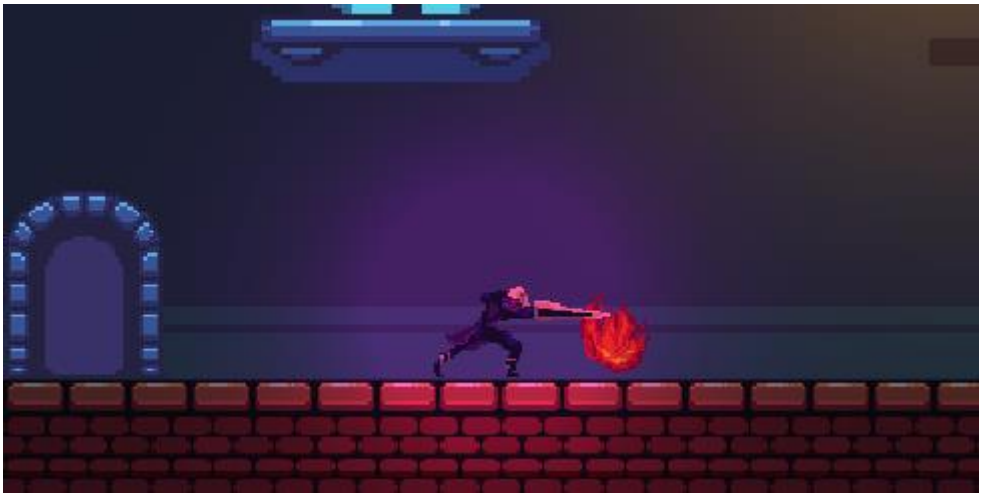

Figure 5. Animation of enemy getting killed and disappearing

```
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);
    }
```

Table 4. Code responsible for enemy taking damage 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.*

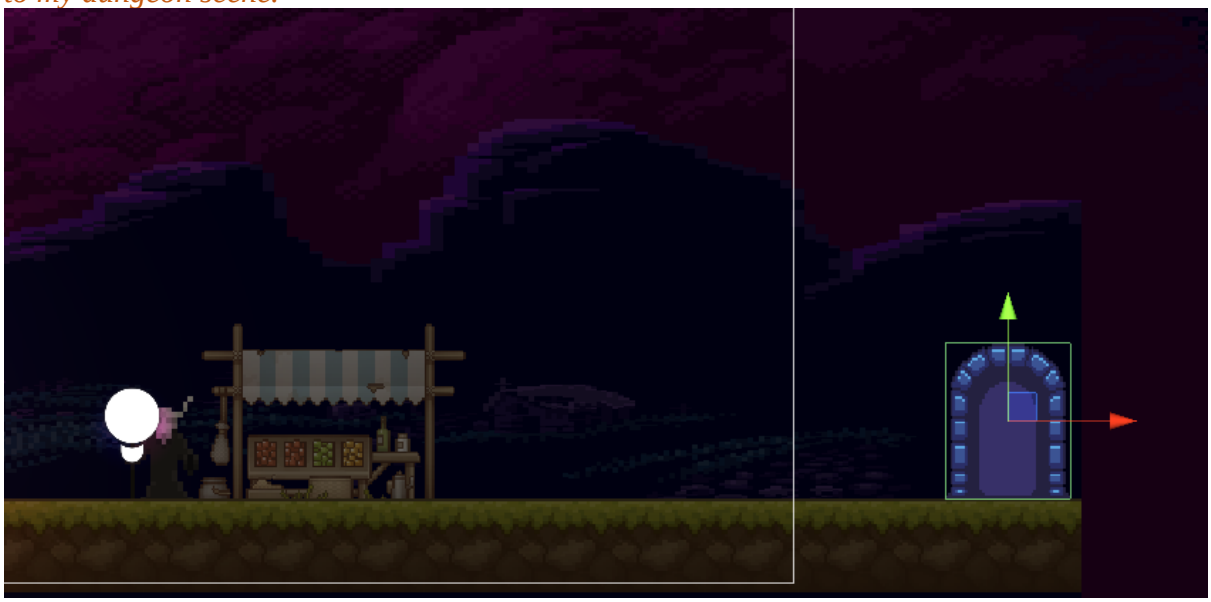

Figure 6. Door sprite upon colliding with which, the user is taken to the dungeon's entrance scene

```
    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 5. Code responsible for attacking enemies

Laboratory work #2

List of tasks (main functionality of your project)

1. Title of Task #1
2. Title of Task #2
3. Title of Task #3
4. …

Solution

Task #1. *Title of Task*

Description of implementation (3-5 sentences). *Vestibulum hendrerit felis at turpis ultrices imperdiet. Nulla facilisi curabitur vitae semper nulla. Etiam rhoncus orci dolor, ac dictum erat iaculis sed. Aliquam pulvinar viverra consequat. Nam eu mi in mauris semper pellentesque eget ut erat.*

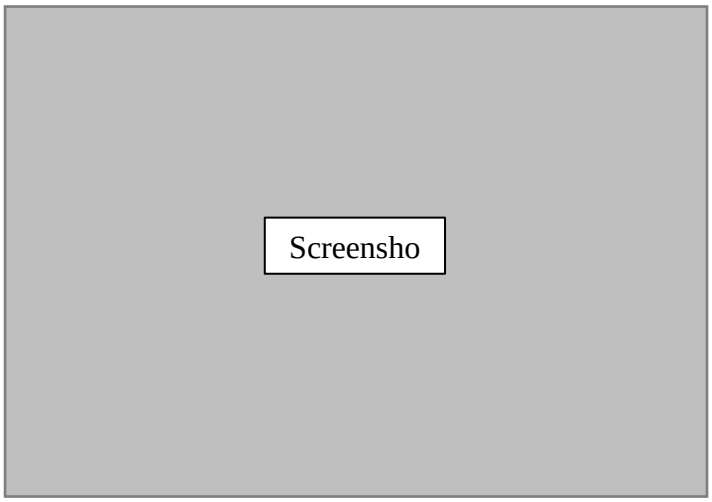

Figure 4. Screenshot #1

In the case of using functions, the description of each main function should be completed with the source code FRAGMENTS (the functions should be indexed in a separate table of contents);

Fragment of Source

Table 4. Title of fragment #1

Task #2. *Title of Task*

Description of implementation (3-5 sentences). *Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed ultricies nunc sit amet sem blandit, at ultricies nibh commodo. Duis ut*

mollis risus. Proin hendrerit libero eu felis dapibus imperdiet. Fusce posuere felis ornare luctus molestie. Duis ut odio pretium, bibendum elit et, molestie quam.

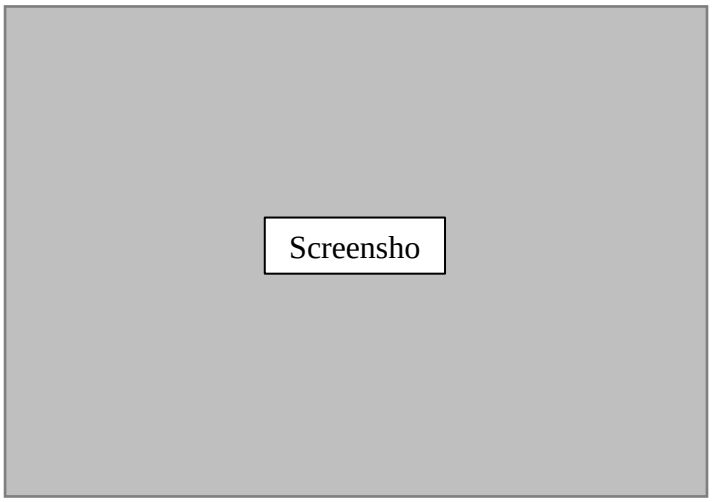

Figure 5. Screenshot #2

In the case of using functions, the description of each main function should be completed with the source code FRAGMENTS (the functions should be indexed in a separate table of contents);

Fragment of Source

Table 5. Title of fragment #2

Task #3. *Title of Task*

Description of implementation (3-5 sentences). *Vestibulum hendrerit felis at turpis ultrices imperdiet. Nulla facilisi curabitur vitae semper nulla. Etiam rhoncus orci dolor, ac dictum erat iaculis sed. Aliquam pulvinar viverra consequat. Nam eu mi in mauris semper pellentesque eget ut erat.*

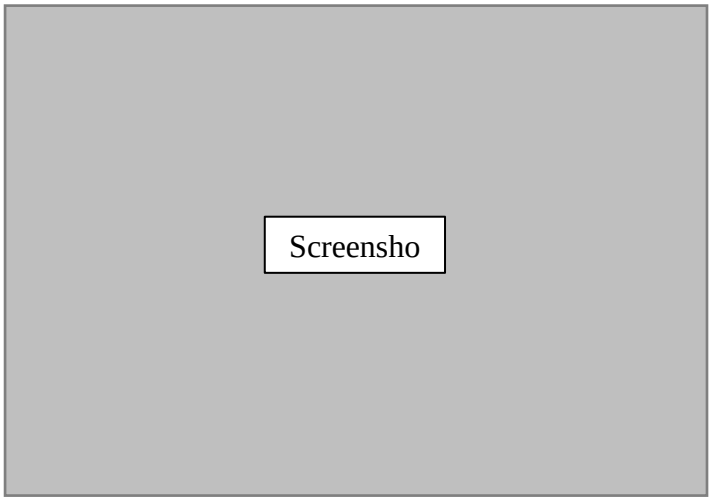

Figure 6. Screenshot #3

In the case of using functions, the description of each main function should be completed with the source code FRAGMENTS (the functions should be indexed in a separate table of contents);

Fragment of Source

Table 6. Title of fragment #3

Laboratory work #3

List of tasks (main functionality of your project)

1. Title of Task #1
2. Title of Task #2
3. Title of Task #3
4. …

Solution

Task #1. *Title of Task*

Description of implementation (3-5 sentences). *Vestibulum hendrerit felis at turpis ultrices imperdiet. Nulla facilisi curabitur vitae semper nulla. Etiam rhoncus orci dolor, ac dictum erat iaculis sed. Aliquam pulvinar viverra consequat. Nam eu mi in mauris semper pellentesque eget ut erat.*

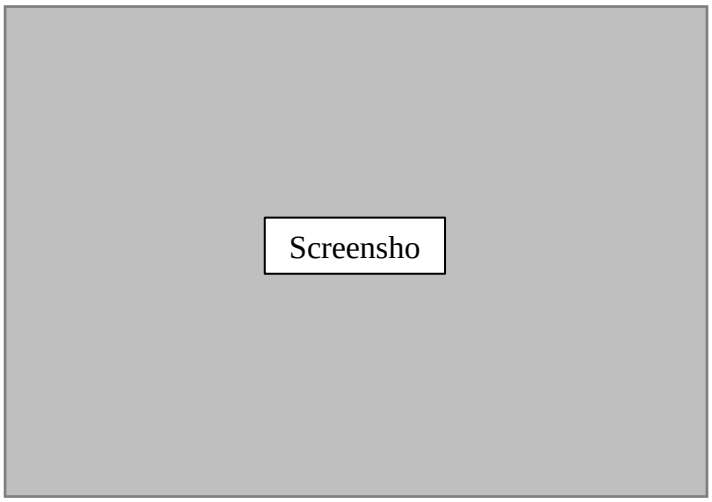

Figure 7. Screenshot #1

In the case of using functions, the description of each main function should be completed with the source code FRAGMENTS (the functions should be indexed in a separate table of contents);

Fragment of Source

Table 7. Title of fragment #1

Task #2. *Title of Task*

Description of implementation (3-5 sentences). *Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed ultricies nunc sit amet sem blandit, at ultricies nibh commodo. Duis ut*

mollis risus. Proin hendrerit libero eu felis dapibus imperdiet. Fusce posuere felis ornare luctus molestie. Duis ut odio pretium, bibendum elit et, molestie quam.

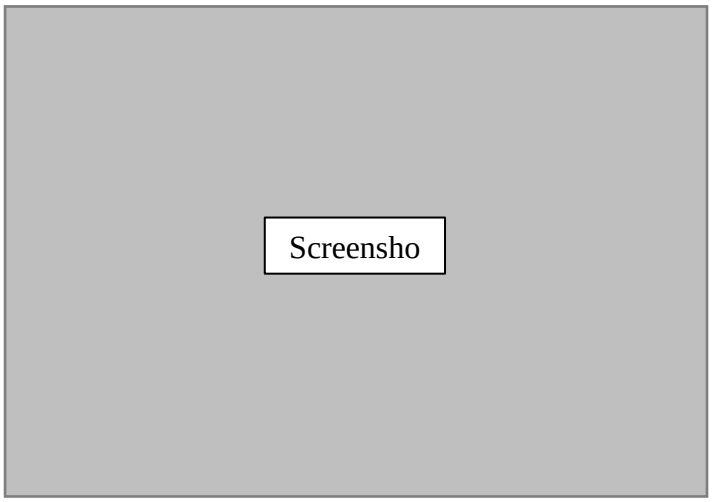

Figure 8. Screenshot #2

In the case of using functions, the description of each main function should be completed with the source code FRAGMENTS (the functions should be indexed in a separate table of contents);

Fragment of Source

Table 8. Title of fragment #2

Task #3. *Title of Task*

Description of implementation (3-5 sentences). *Vestibulum hendrerit felis at turpis ultrices imperdiet. Nulla facilisi curabitur vitae semper nulla. Etiam rhoncus orci dolor, ac dictum erat iaculis sed. Aliquam pulvinar viverra consequat. Nam eu mi in mauris semper pellentesque eget ut erat.*

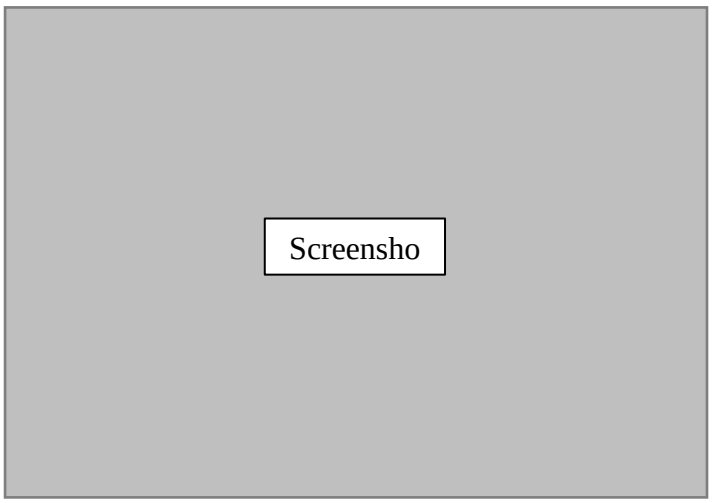

Figure 9. Screenshot #3

In the case of using functions, the description of each main function should be completed with the source code FRAGMENTS (the functions should be indexed in a separate table of contents);

Fragment of Source

Table 9. Title of fragment #3

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.*

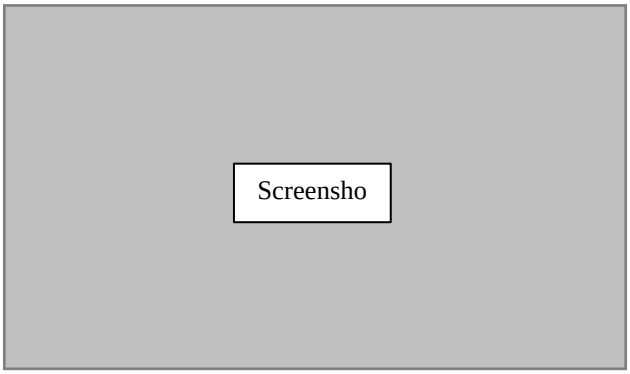

Figure 10. Screenshot #5

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.

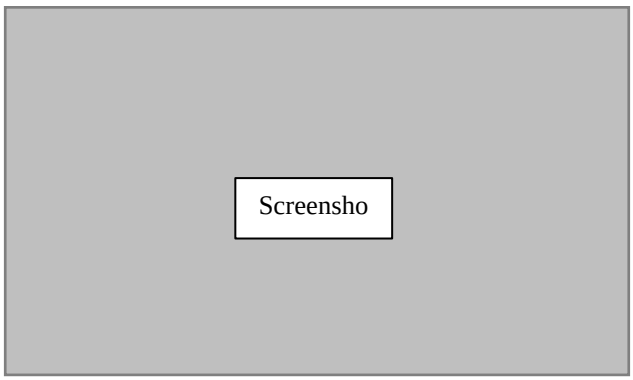

Figure 11. Screenshot #5

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*
2. *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*
4. *https://assetstore.unity.com/packages/2d/characters/gothicvania-cemetery-120509*
5. *https://assetstore.unity.com/packages/2d/environments/medieval-pixel-art-asset-free-130131*
6. *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);
        }
    }
}
```