KAUNAS UNIVERSITY OF TECHNOLOGY

FACULTY OF INFORMATICS

T120B166 Development of Computer Games and Interactive Applications

Lost In Cold (survival game)

IFF-6/6 Ignas Jasonas:

Date: *2019.03.12*

Kaunas, 2019

Tables of Contents

Tables of Images

Table of Tables/functions

No table of figures entries found.

Work Distribution Table:

Name/Surname	***Description of game development part***
Ignas Jasonas	*Full game development*

Description of Your Game

Description of Your Game.

1. 3D or 2D? *3D*
2. What type is your game? *First person survival game*
3. What genre is your game? *Survival*
4. Platforms (mobile, PC or both?) *PC*
5. Scenario Description. *Main character goes on hiking in the mountains, when suddenly a blizzard comes and he gets lost. Wondering around he sets up a camp to stay overnight and get through the storm. When he wakes up, the storm has passed, but he is lost. Main goal is to find home and survival the winter cold.*

Laboratory work #1

List of tasks (main functionality of your project)

1. Create a map
2. Add environment elements
3. Add lighting sources
4. Add sounds
5. Use at least 20 different materials

Solution

Task #1. *Create a map*

Description of the implementation (3-5 sentences). *I created 24 500x500 terrain plains and raised or lowered the layout of them. Since the full map is not created yet, half the terrains remain plain. By raising and lowering the ground I created mountains and lake.*

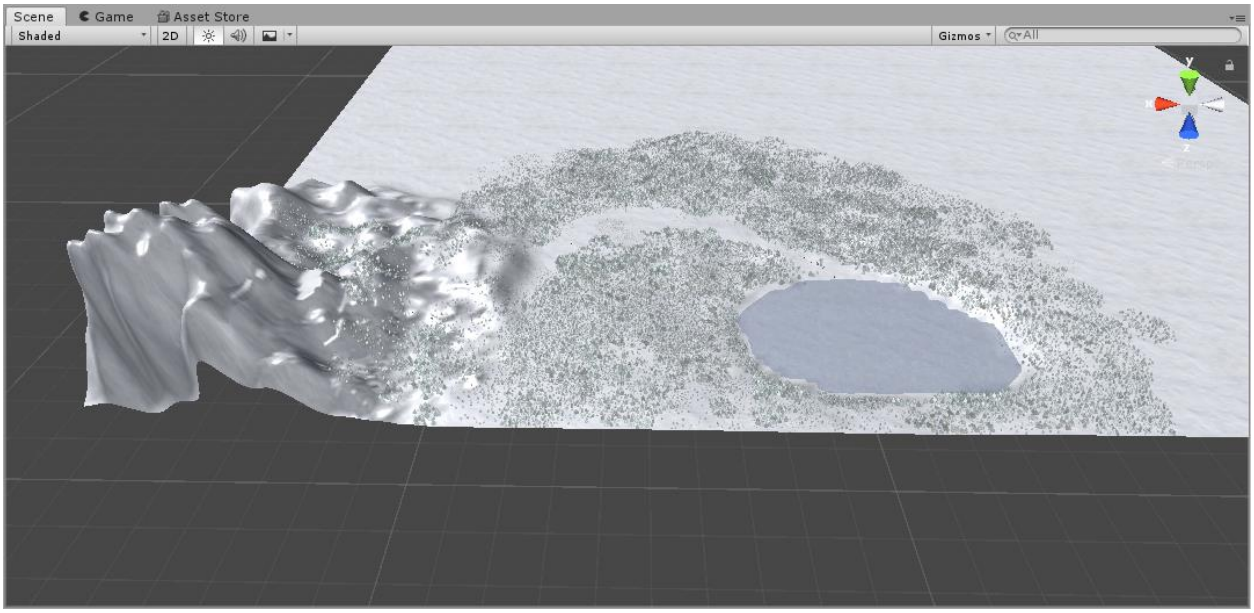

Figure 1. Created terrain

Task #2. *Add environment elements*

Description of the implementation (3-5 sentences). *I downloaded winter themed nature pack Winterland by Shapes (trees, bushes, rocks) and placed them all around with a terrain pain brush. Also by using various free 3D models sites I downloaded and used various objects that could get player attention and guide him towards the goal. I created hunter watchtower and an abandoned house*

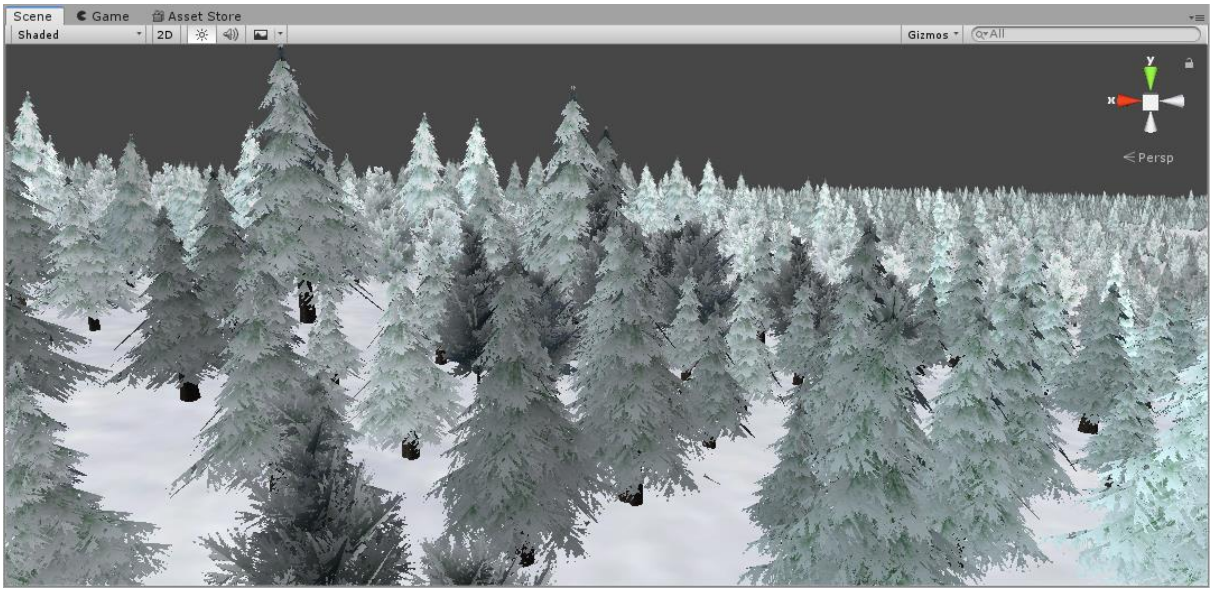

Figure 2. Winter forest

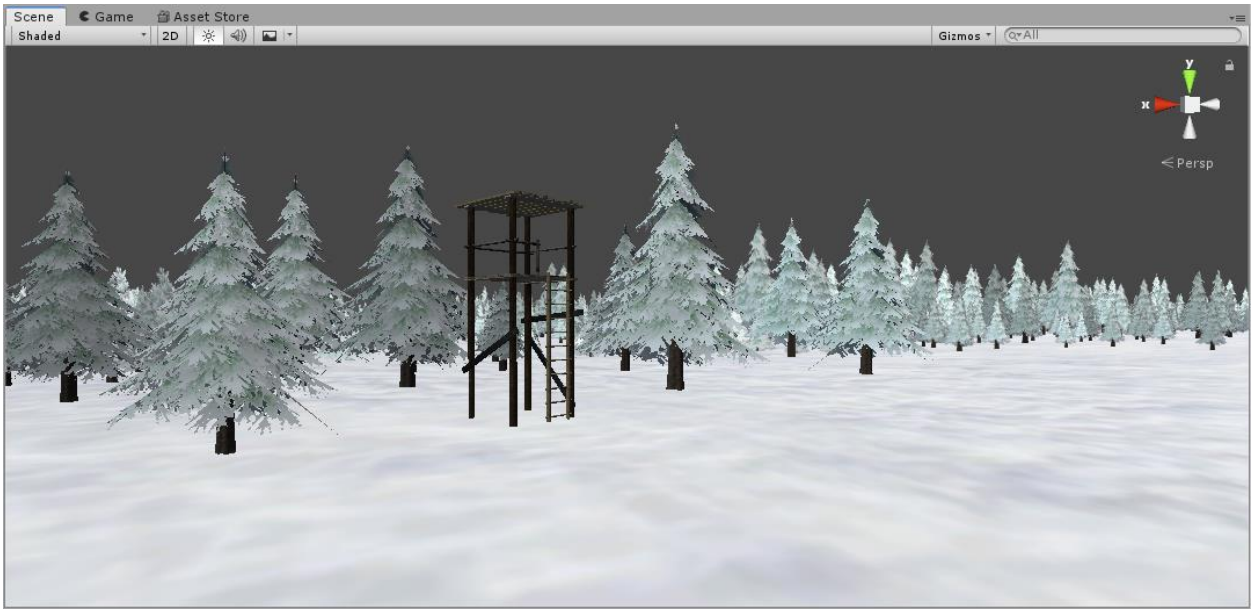

Figure 3. Hunter watchtower

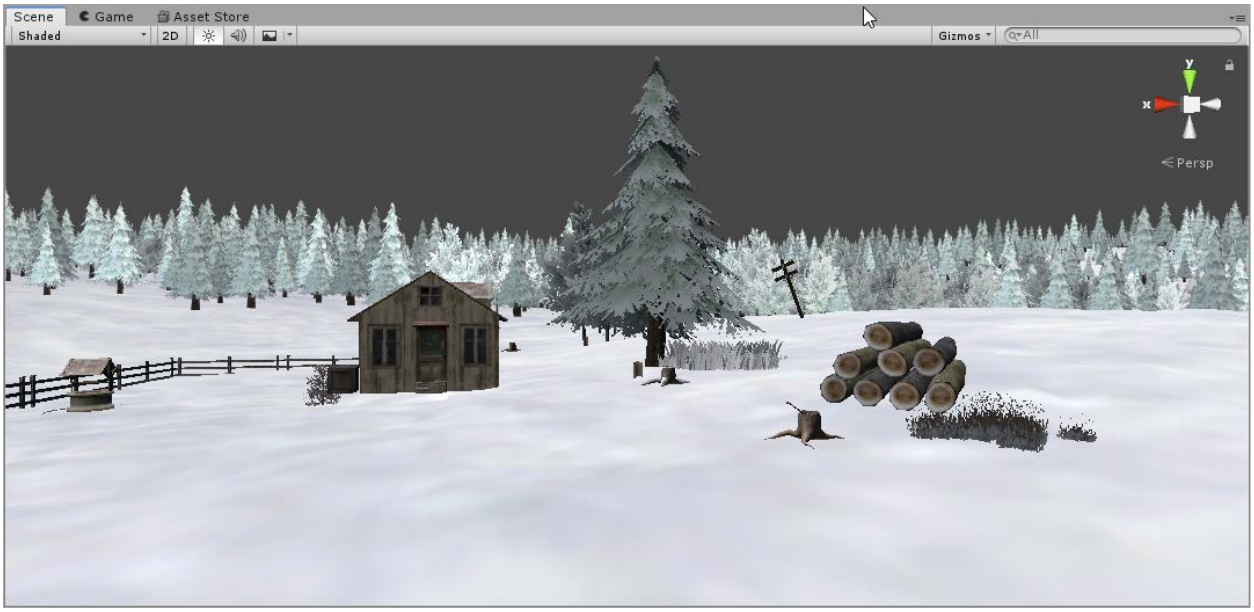

Figure 4. Abandoned house

Task #3. Add lighting sources

Description of the implementation (3-5 sentences). I created sun like source which will have a day/night cycle. There are two more sources, one of them is a flashlight (directional, white light), second one is a lantern (point with yellow color light)

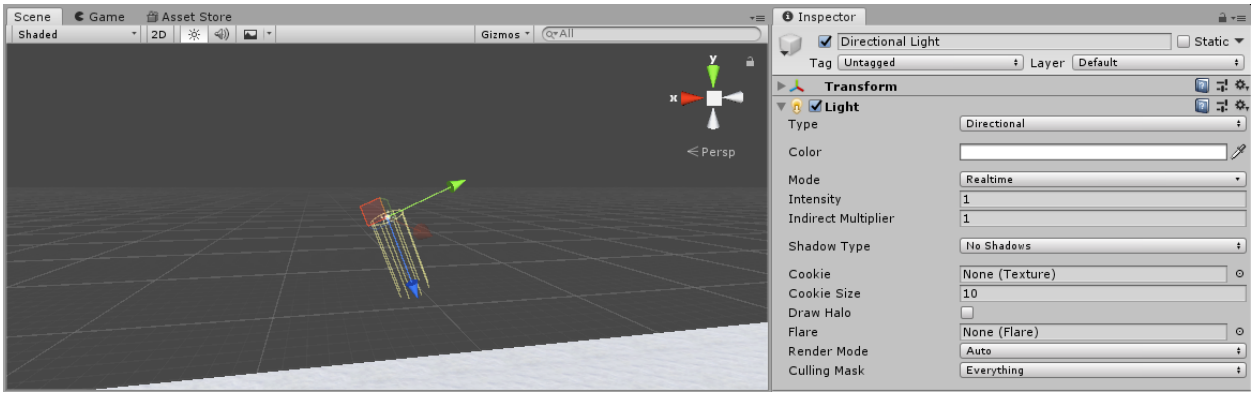

Figure 5. Sun light object

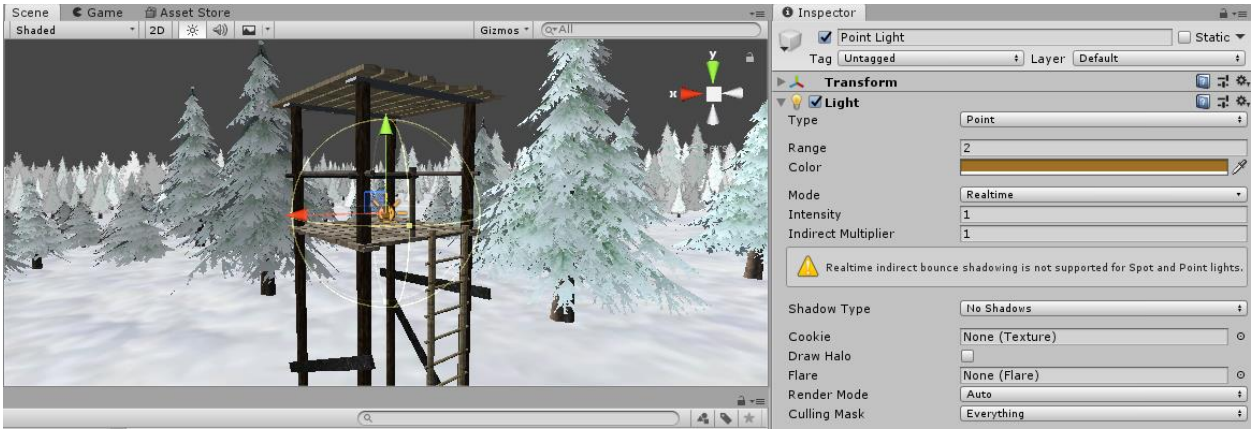

Figure 6. Lantern

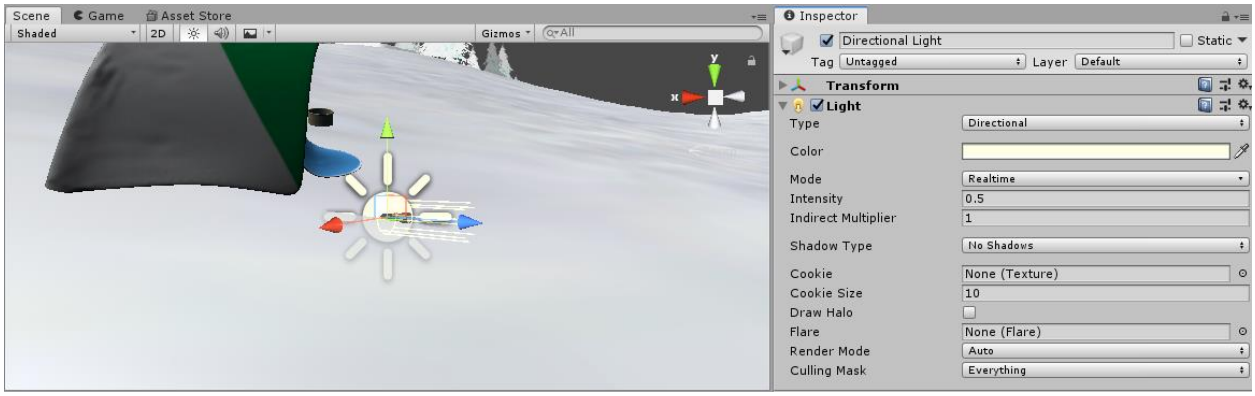

Figure 7. Flashlight

Task #4. Add sounds

Description of the implementation (3-5 sentences). I downloaded 5 background sounds, which is played in certain areas. They are mostly wind/forest ambience sounds

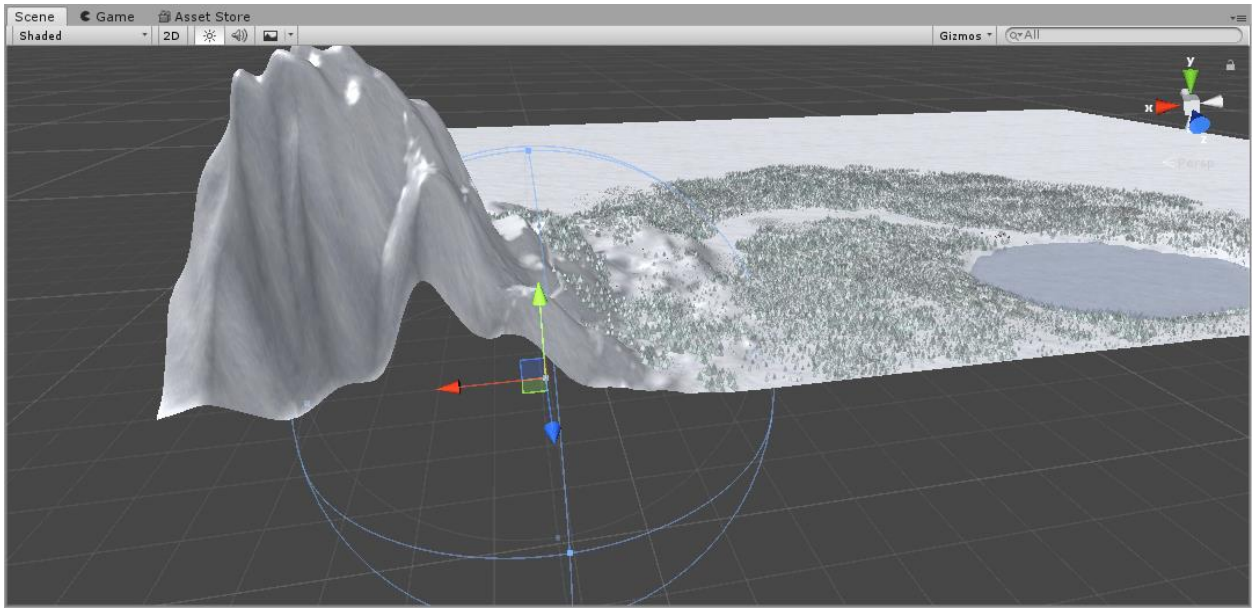

Figure 8. Blizzard background sound and its radius

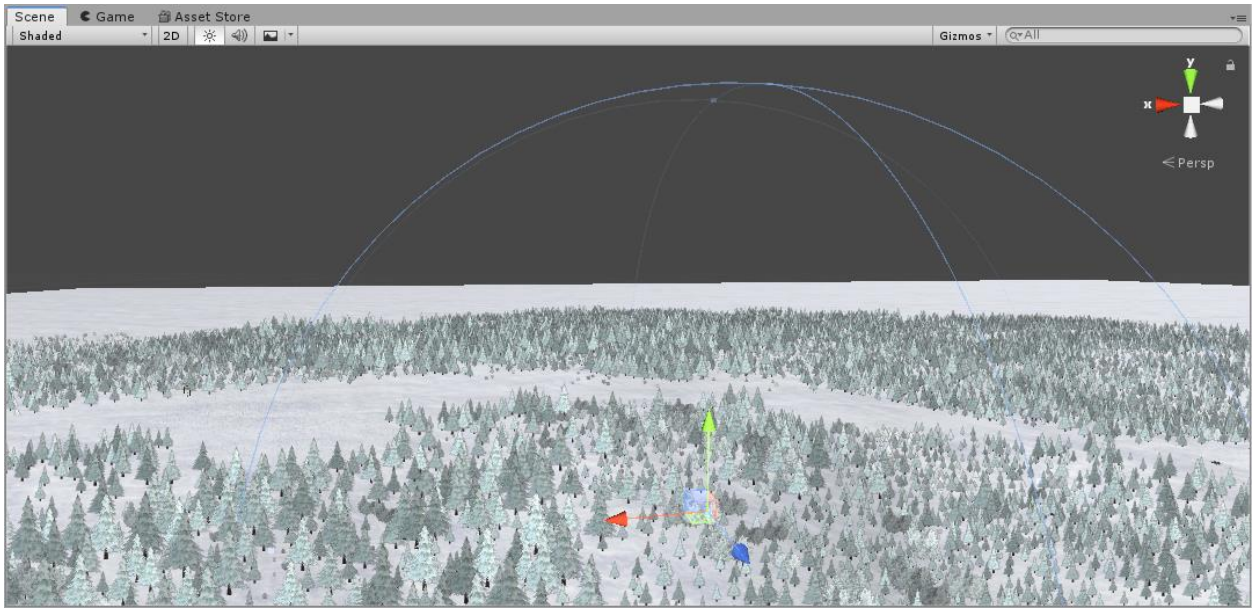

Figure 9. Forest + wind background sound and its radius

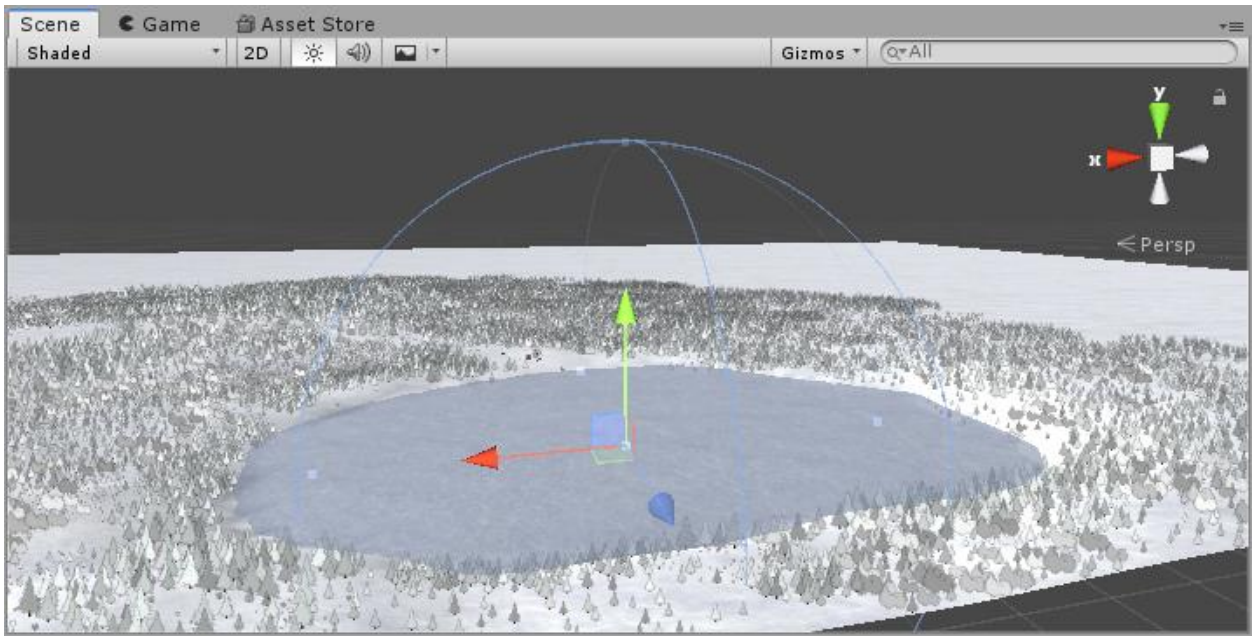

Figure 10. Breeze sound and its radius

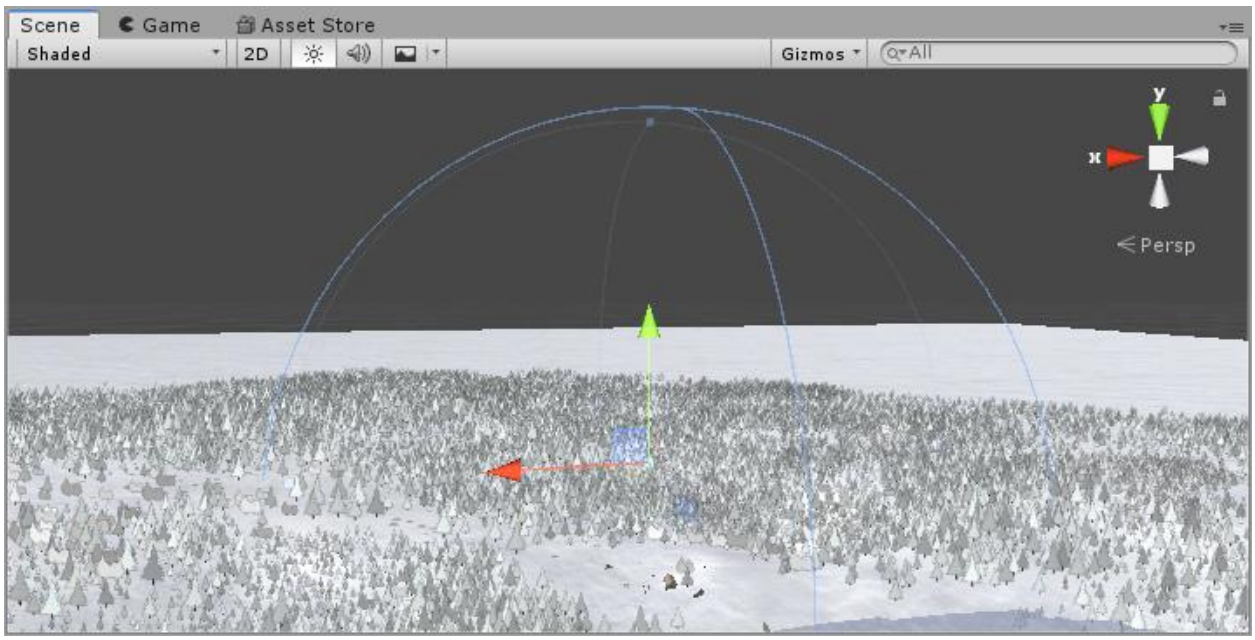

Figure 11. Winter forest sound and its radius

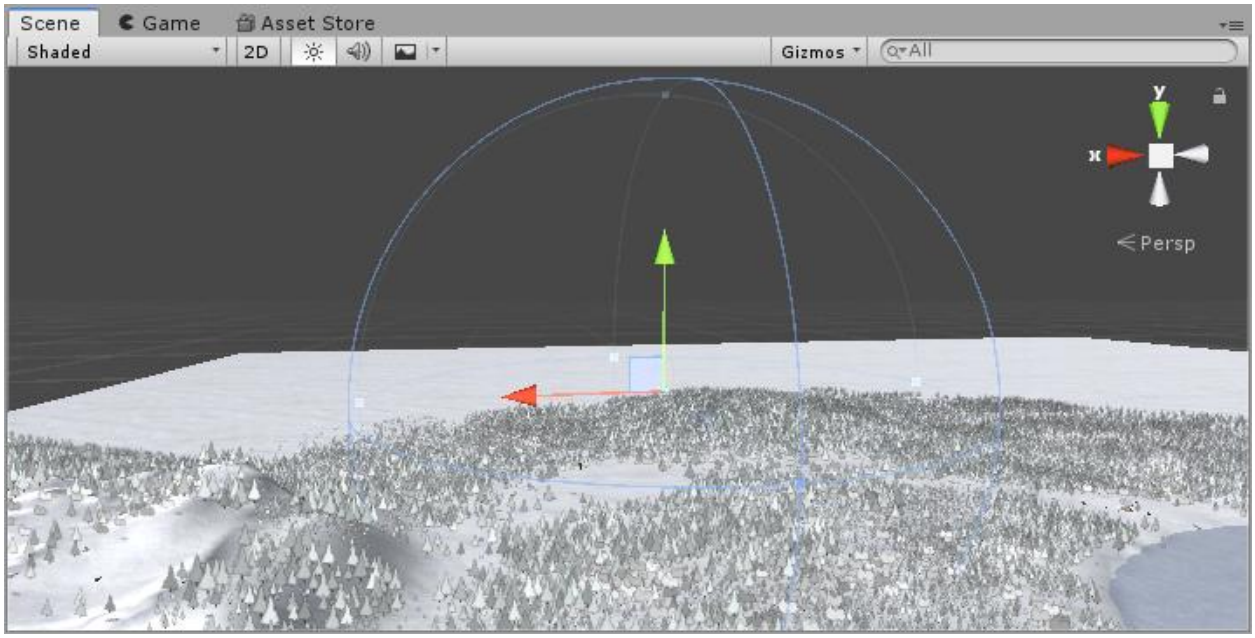

Figure 12. Light wind sound and its radius

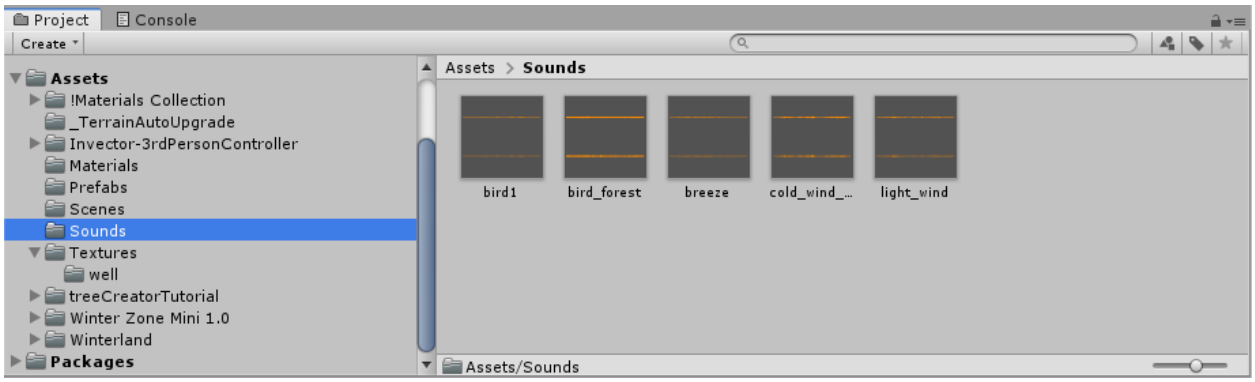

Figure 13. Used sounds

Task #5. Use at least 20 materials

Description of the implementation (3-5 sentences). *I downloaded textures and created materials for elements and objects I used in this map. Currently using 19 materials*

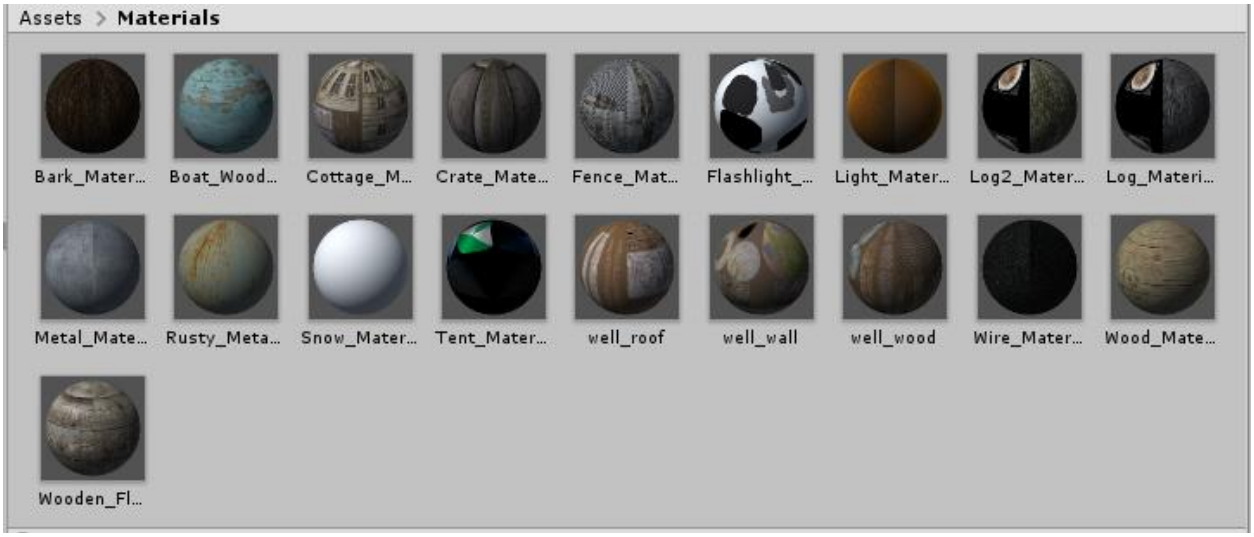

Figure 14. Used materials

Laboratory #1 defence task

Make half of the map blue and other half red by using light

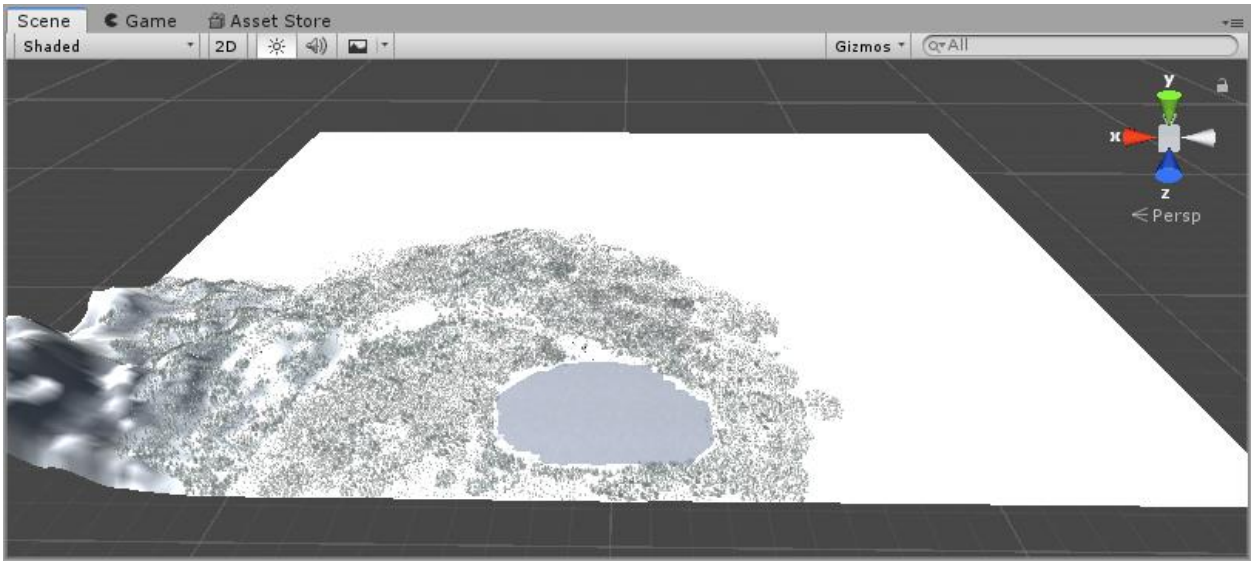

Figure 15. Initial map state

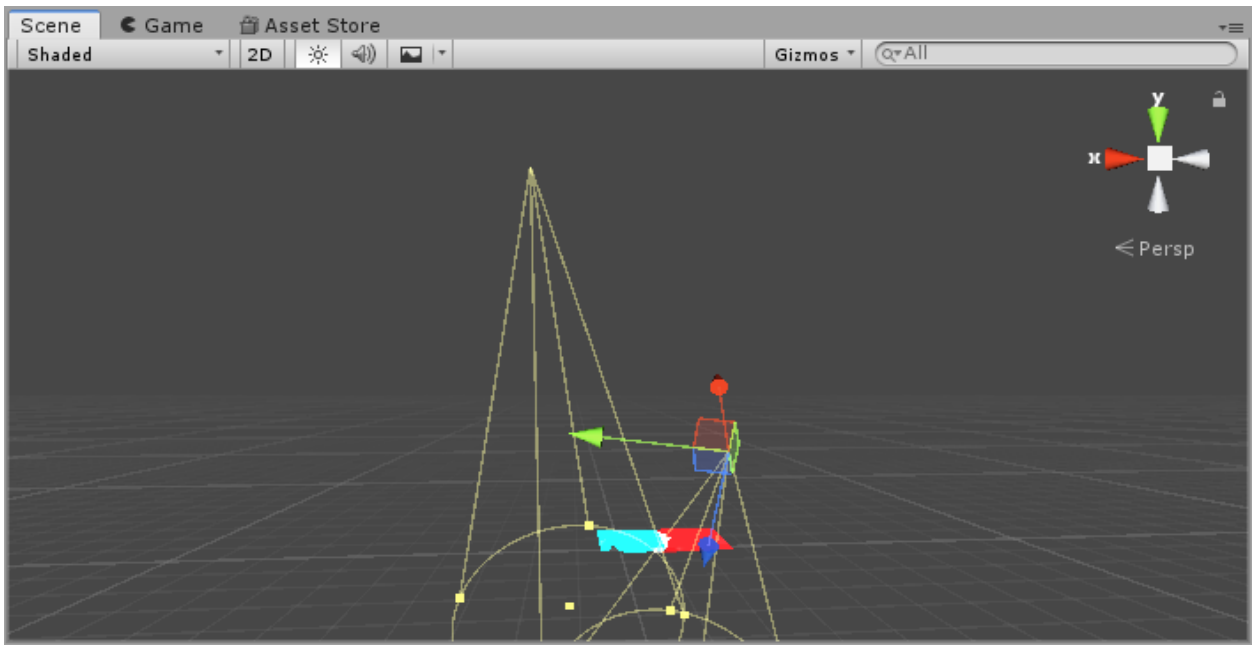

Figure 16. Two spotlights

I created to spotlight objects, changed their spot range and intensity to very high and positioned them so that each of them covers half of the map.

Results

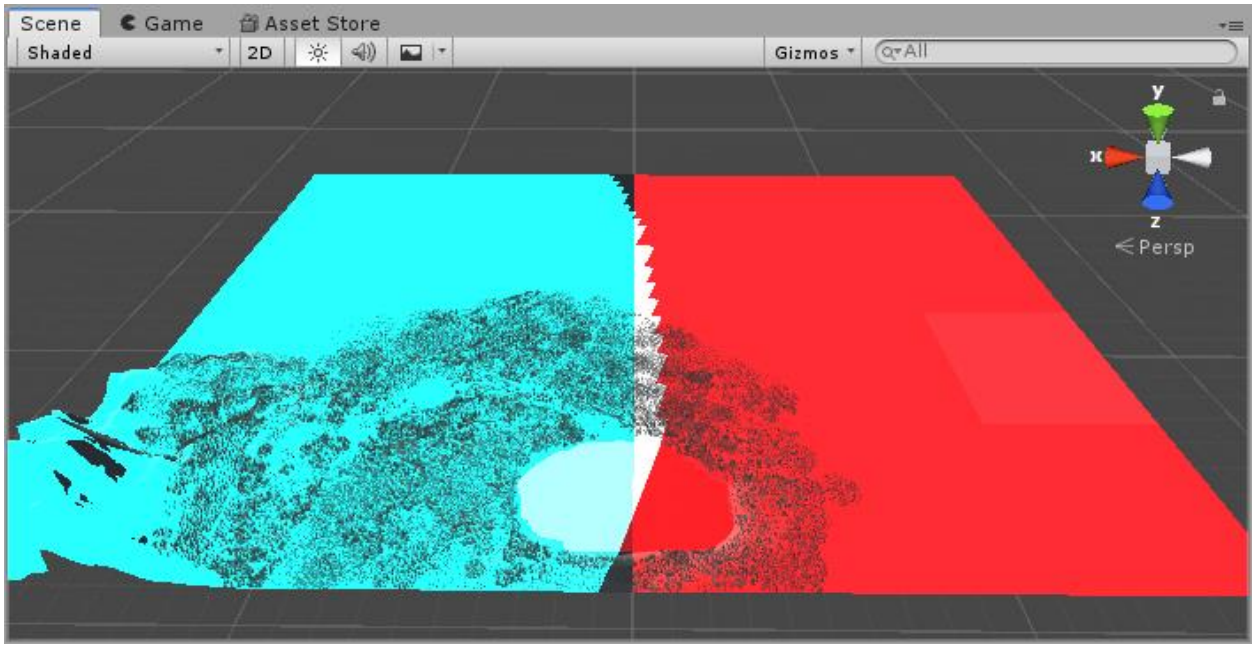

Figure 17. Results

Laboratory work #2

List of tasks

1. Terrain
   1.1. Create terrain and color it using 5 or more materials
   1.2. Create 5 trees
2. Animation
   2.1. Import unique model and assign to third person character controller
   2.2. Add and animate 5 objects
3. Special effects
   3.1. Create 5 unique particle effects
   3.2. Add custom skybox
4. Physics
   4.1. Create 5 different physics materials and apply them in to the project
   4.2. Assign optimal collider based of the shape of the object for all objects in the map
   4.3. Add and animate 5 objects using physics, force and triggers to your game map
5. Optimization

1. Terrain

1.1. Create terrain and color it using 5 or more materials

I created 5 materials and using them, I colored some parts of the map

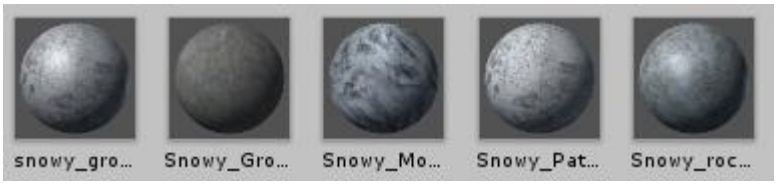

Figure 18. 5 color materials

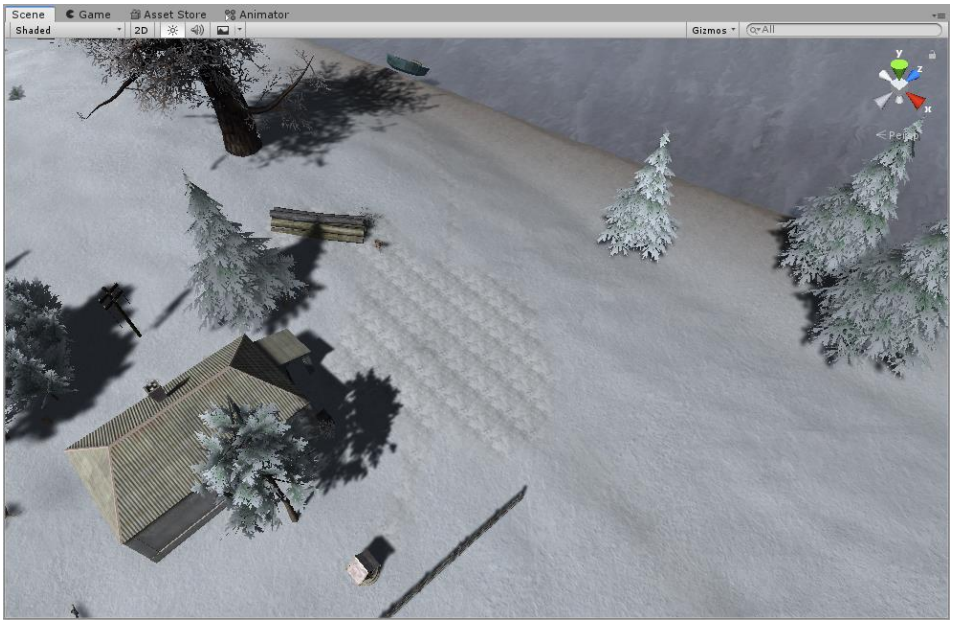

Figure 19. Colored ground near house

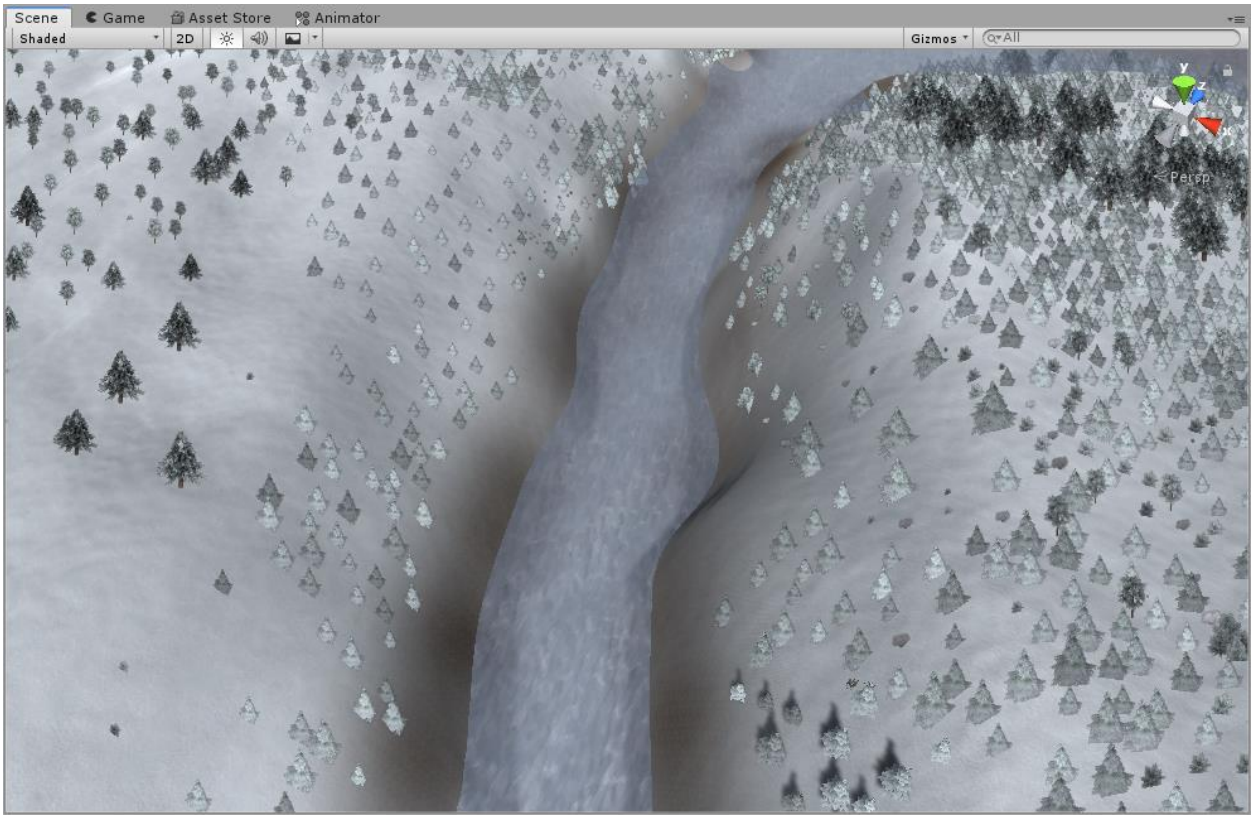

Figure 20. Colored river banks

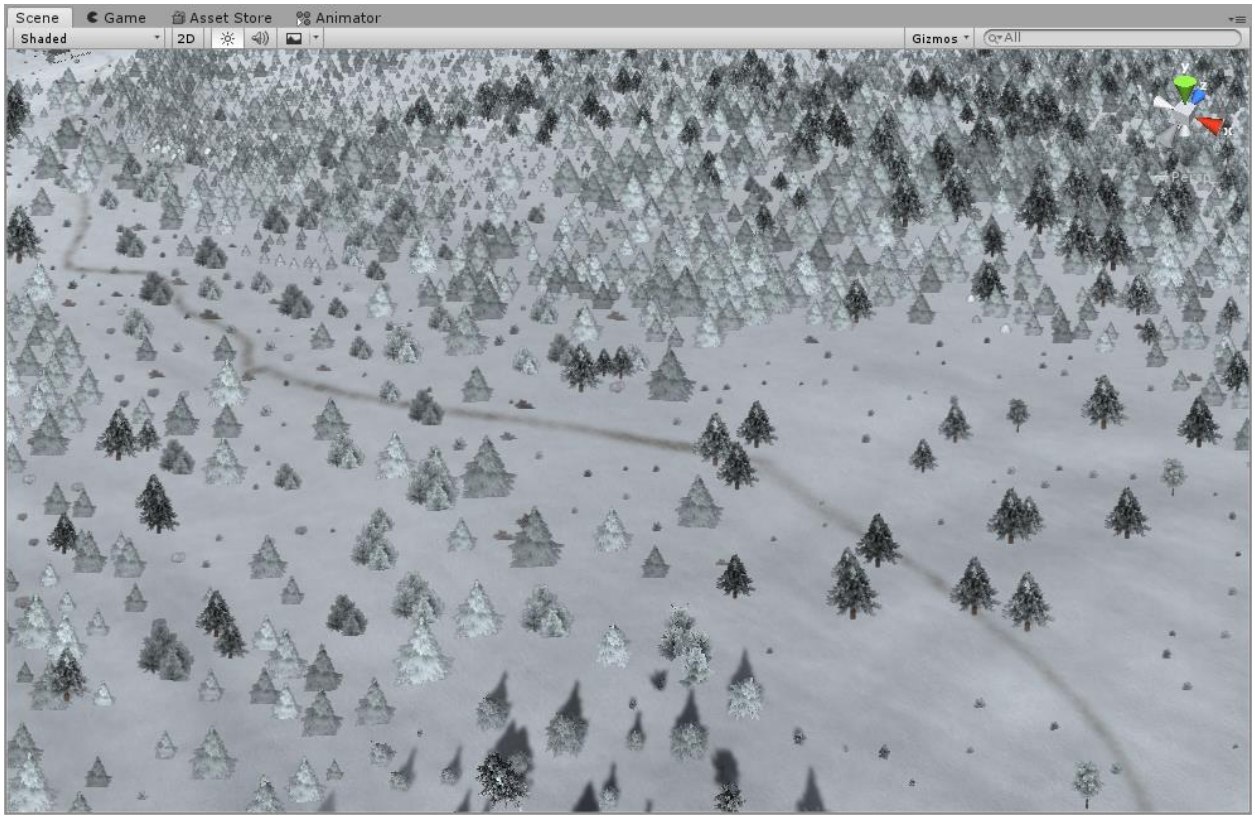

Figure 21. Colored path

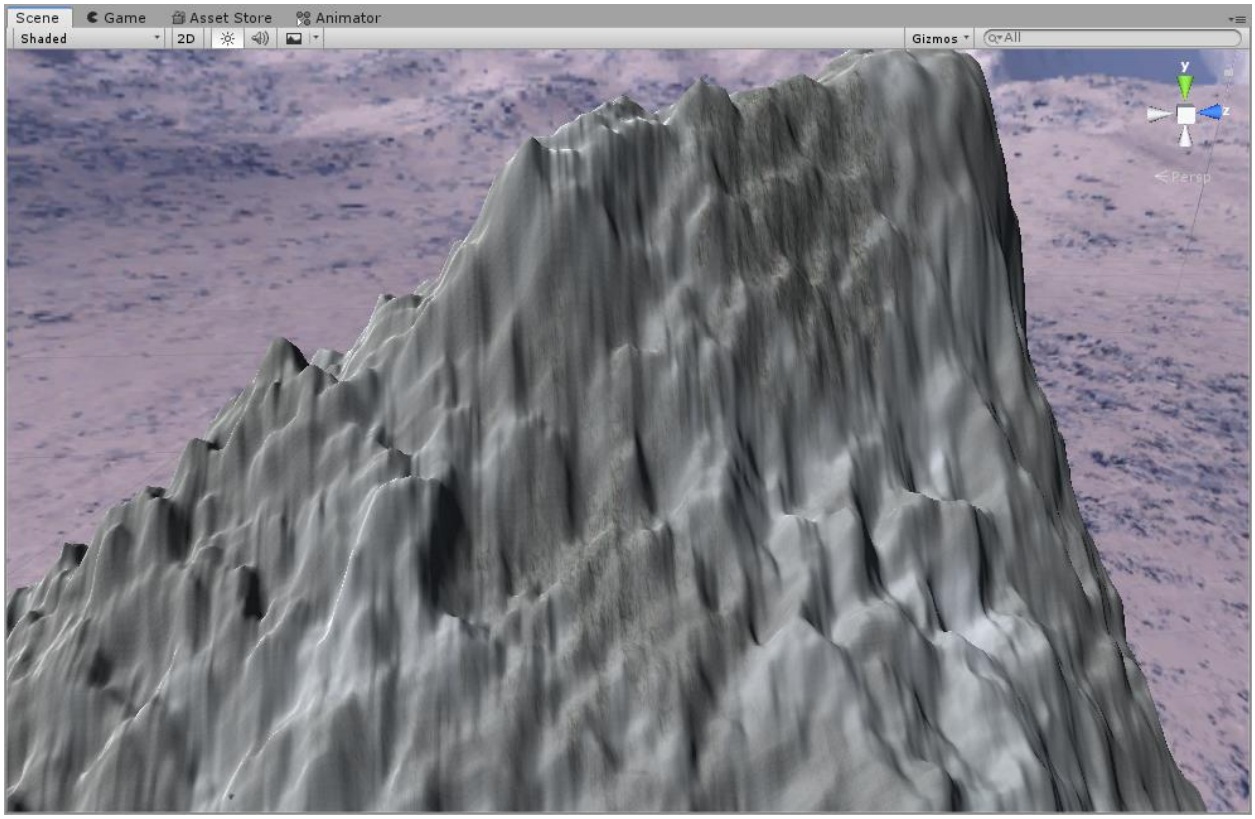

Figure 22. Colored mountain (2 materials)

1.2. Create 5 trees

Using terrain tree creator, I created 5 different trees

Figure 23. 5 created trees

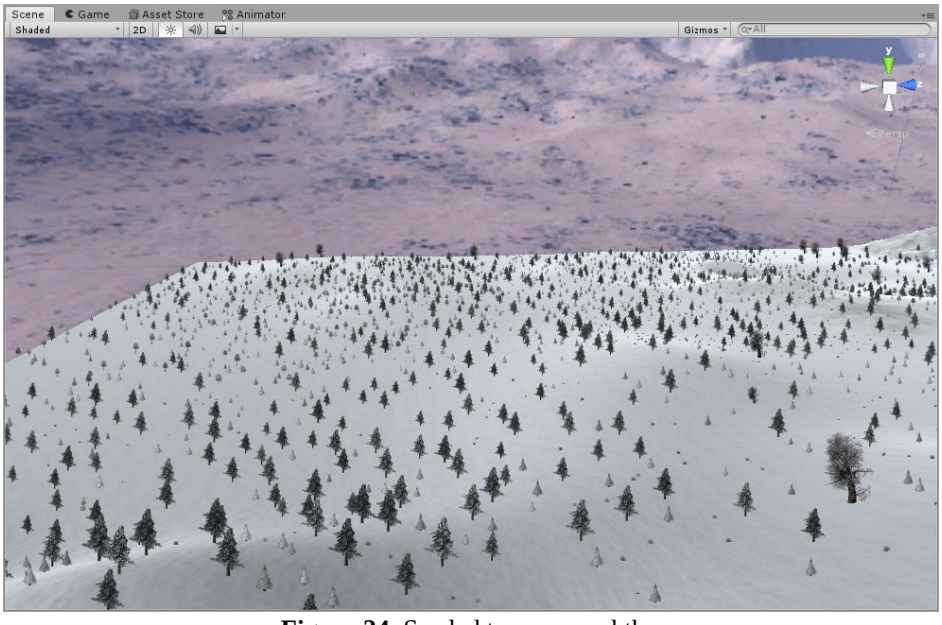

Figure 24. Seeded trees around the map

2. Animation

2.1. Import unique model and assign to third person character controller

Using free 3d models site, I downloaded character model, and assigned it to the character controller

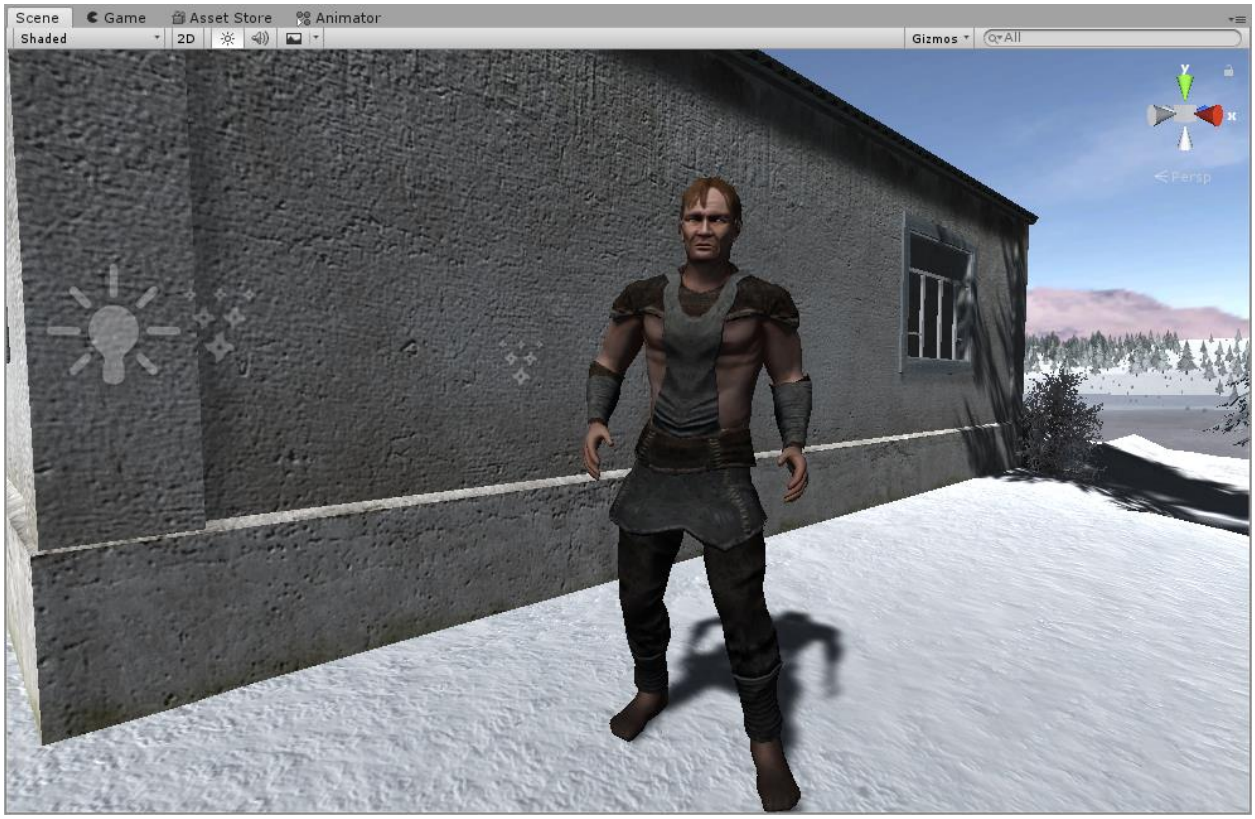

Figure 25. Character controller model

2.2. Add and animate 5 objects

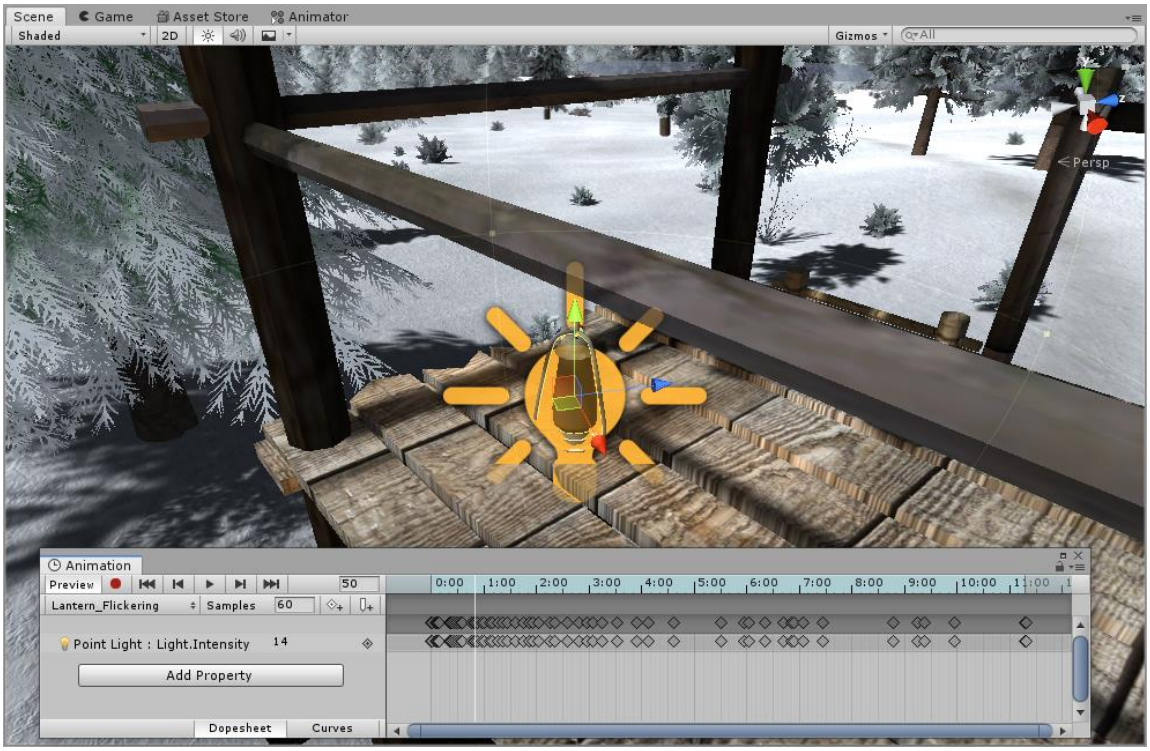

Figure 26. Flickering lantern light animation

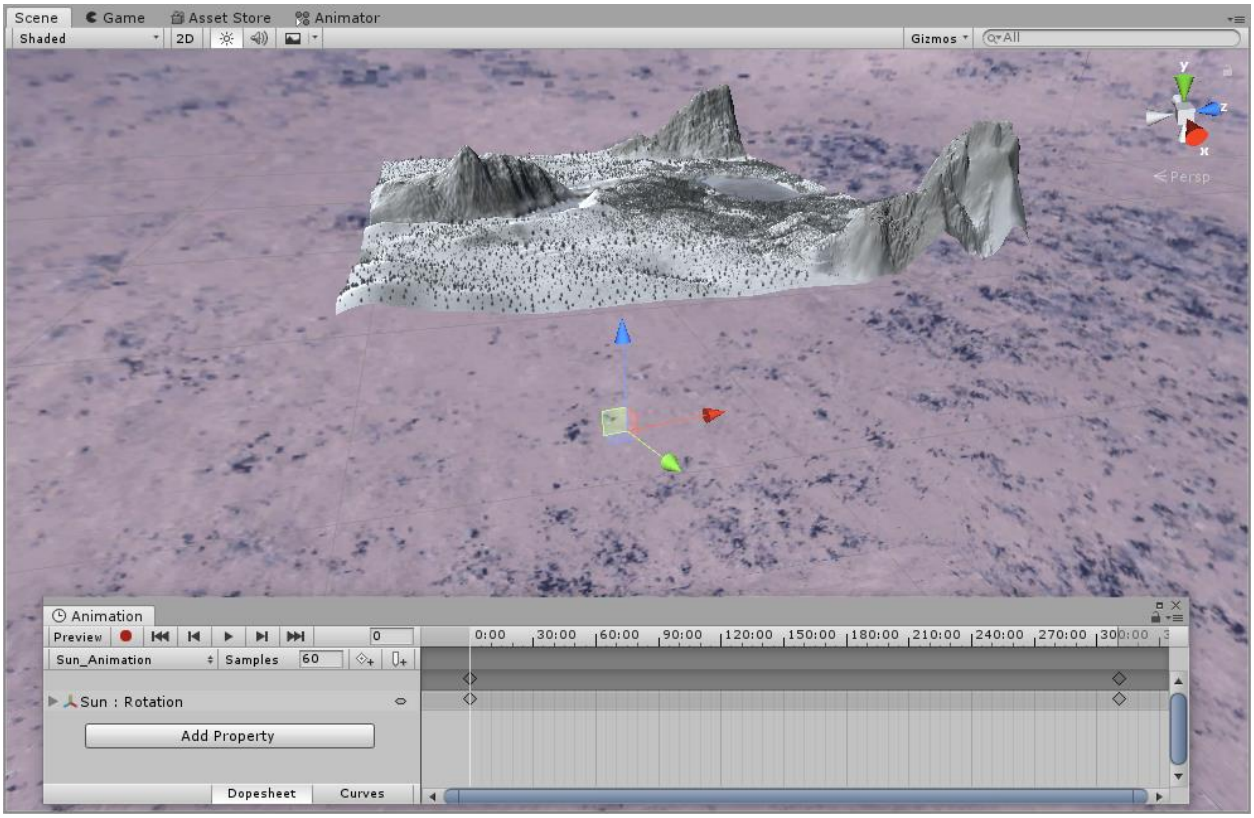

Figure 27. Rotating main light animation (sun, day-night cycle)

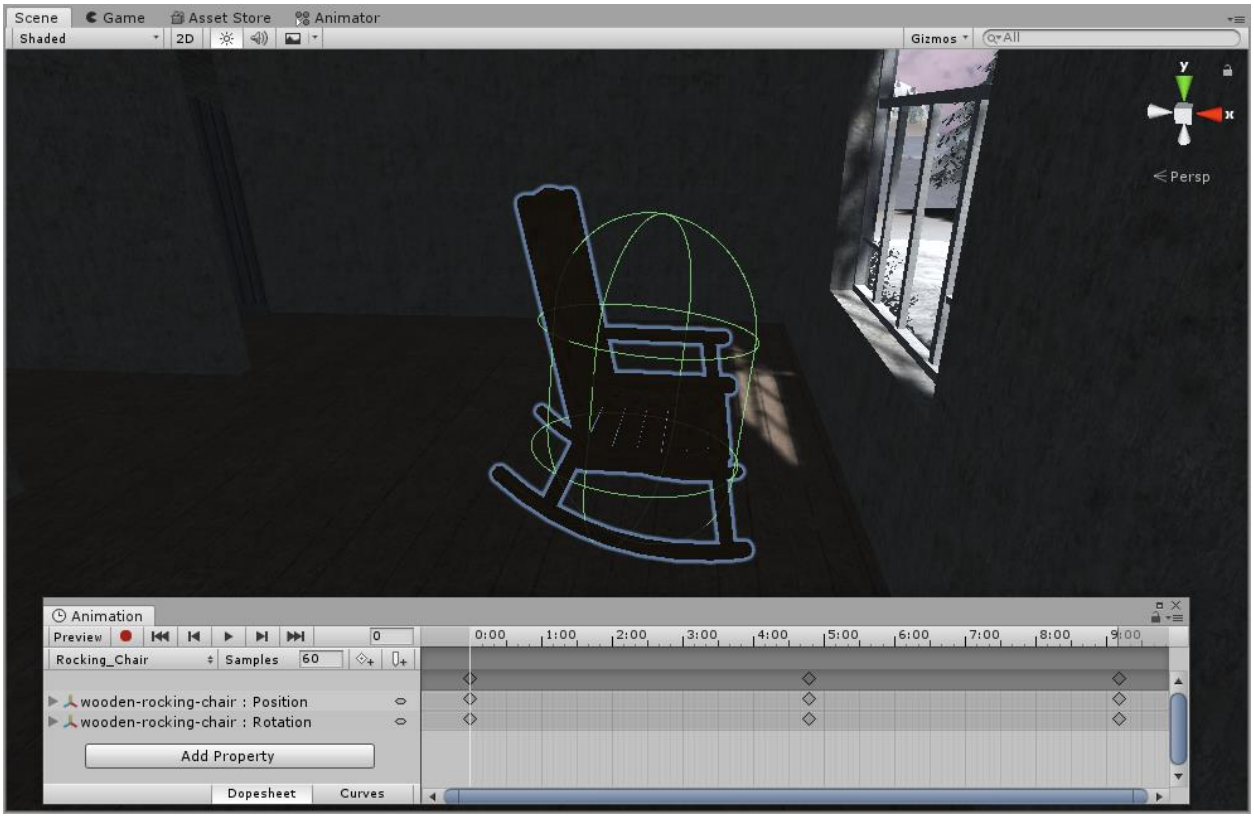

Figure 28. Rocking chair animation

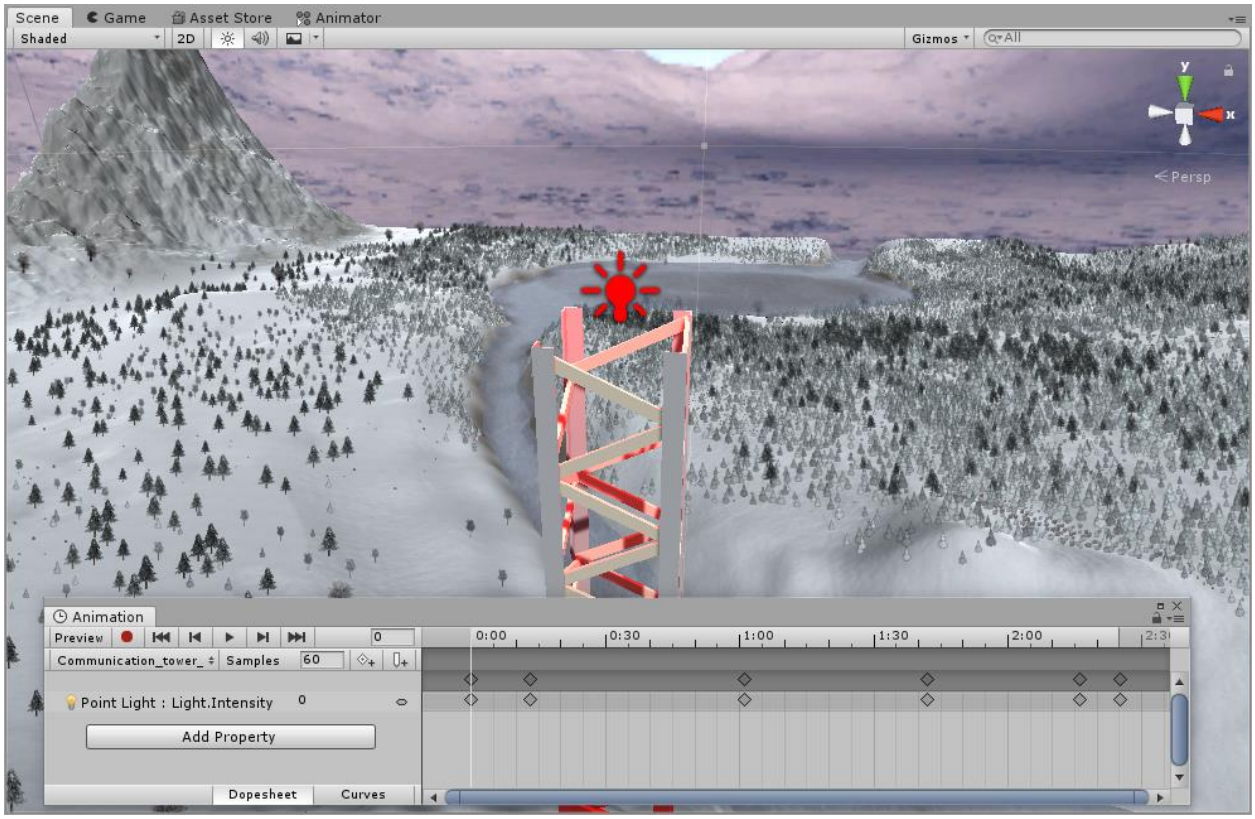

Figure 29. Communication tower light animation

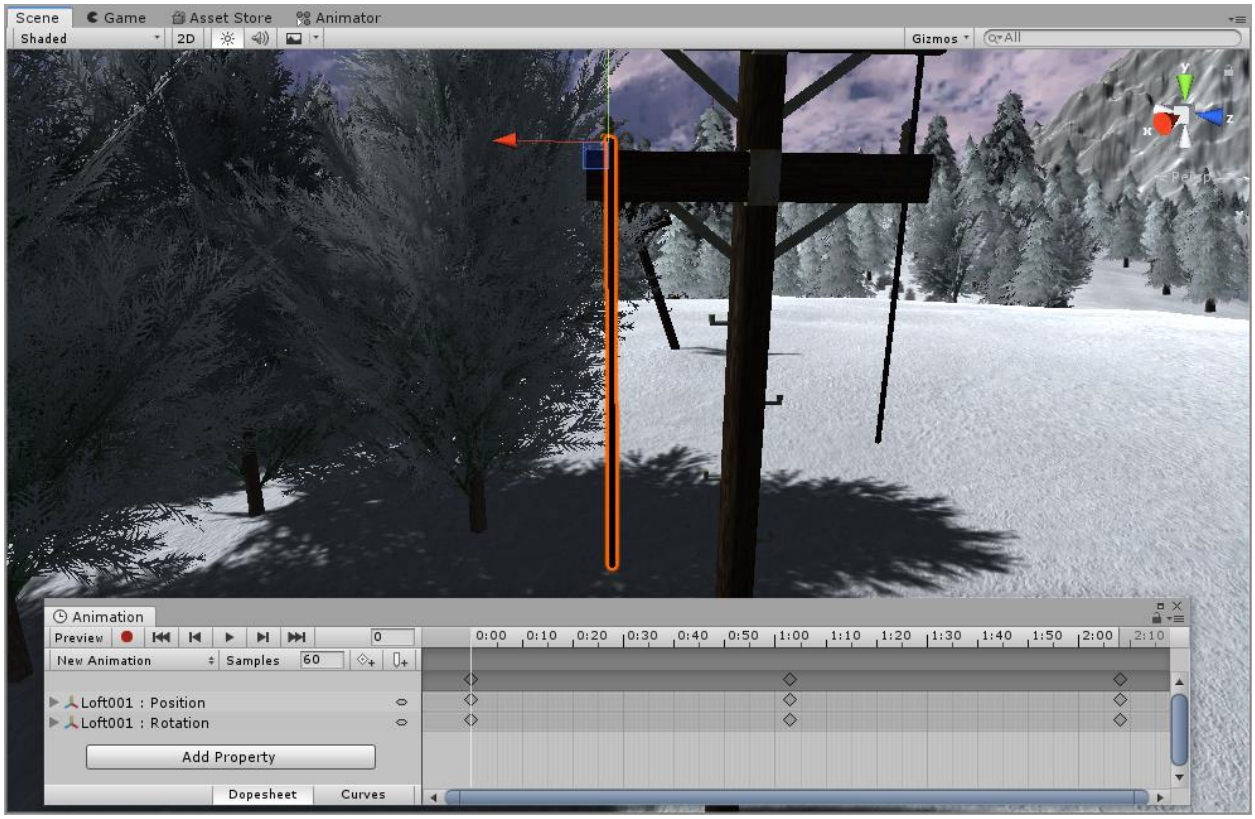

Figure 30. Power pole cable animation

3. Special effects

3.1. Create 5 unique particle effects

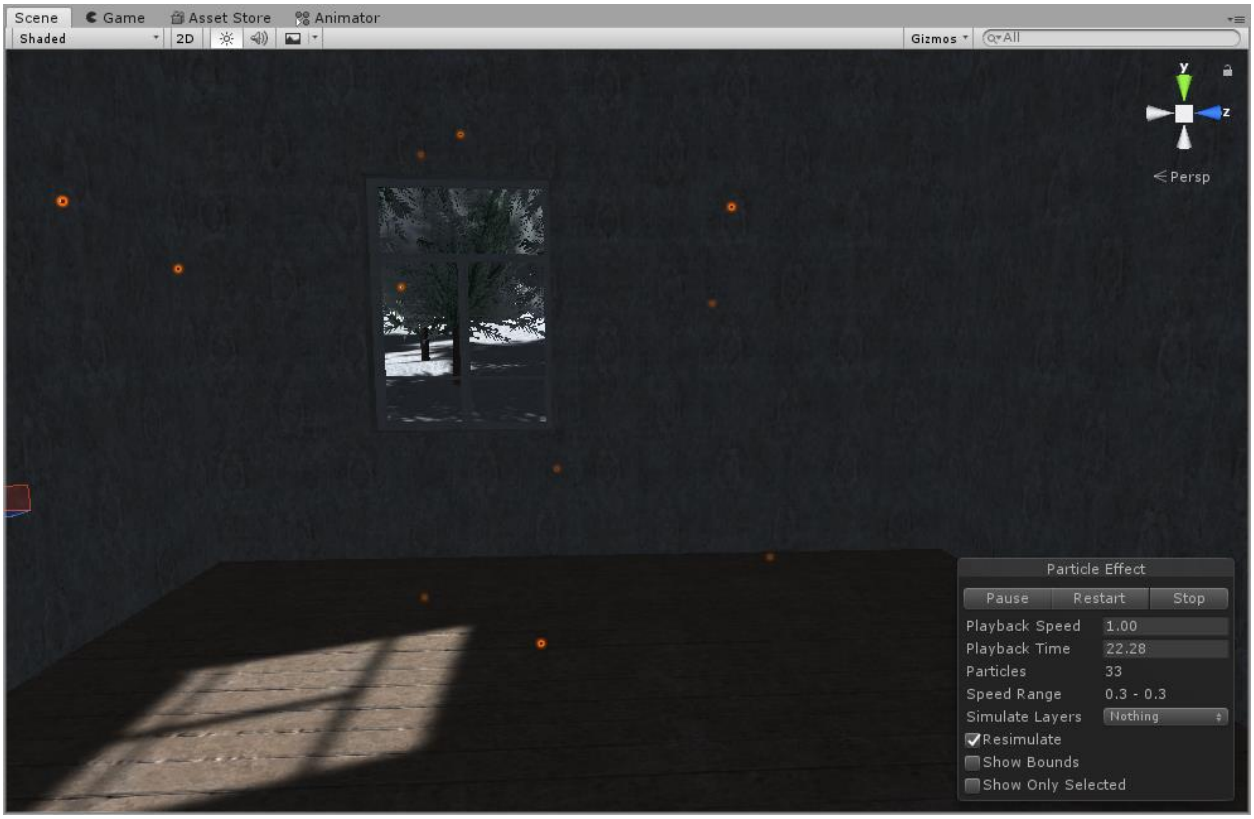

Figure 31. Dust particle effect

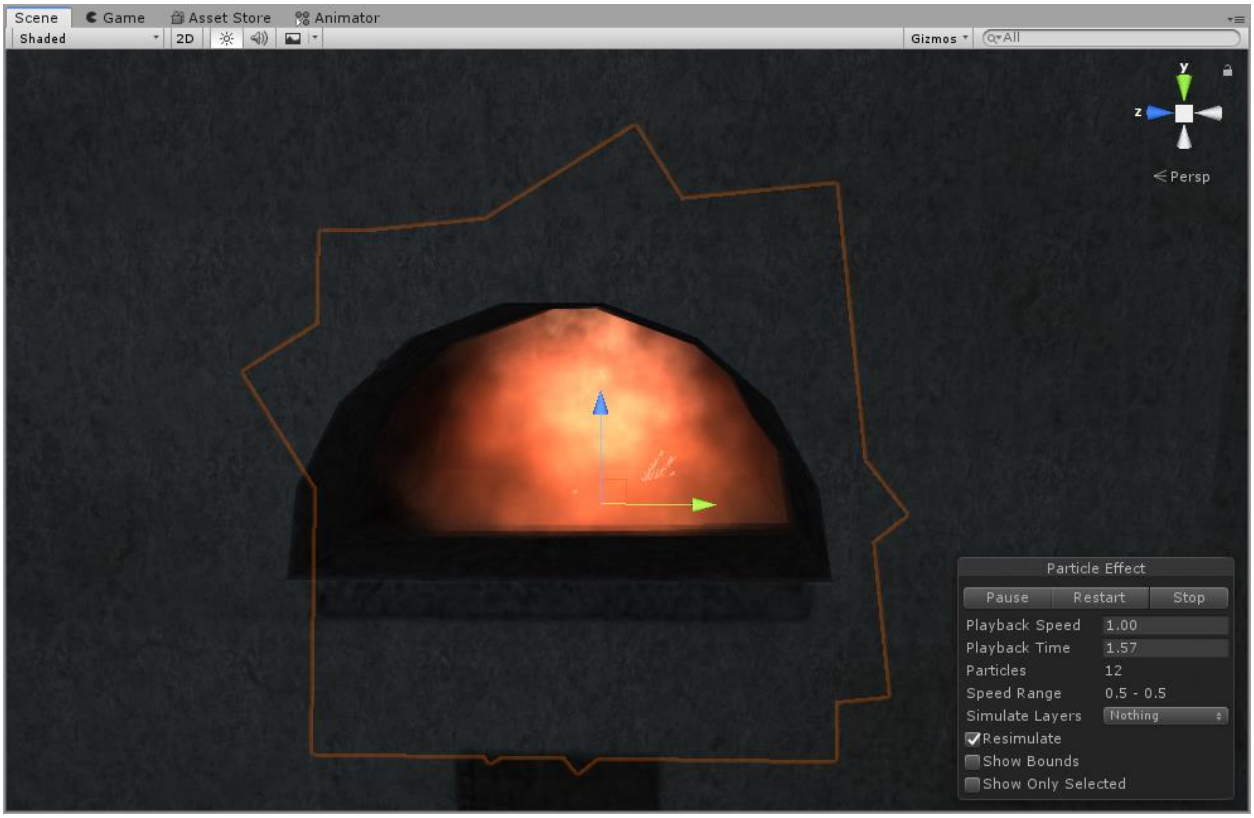

Figure 32. Fire particle effect

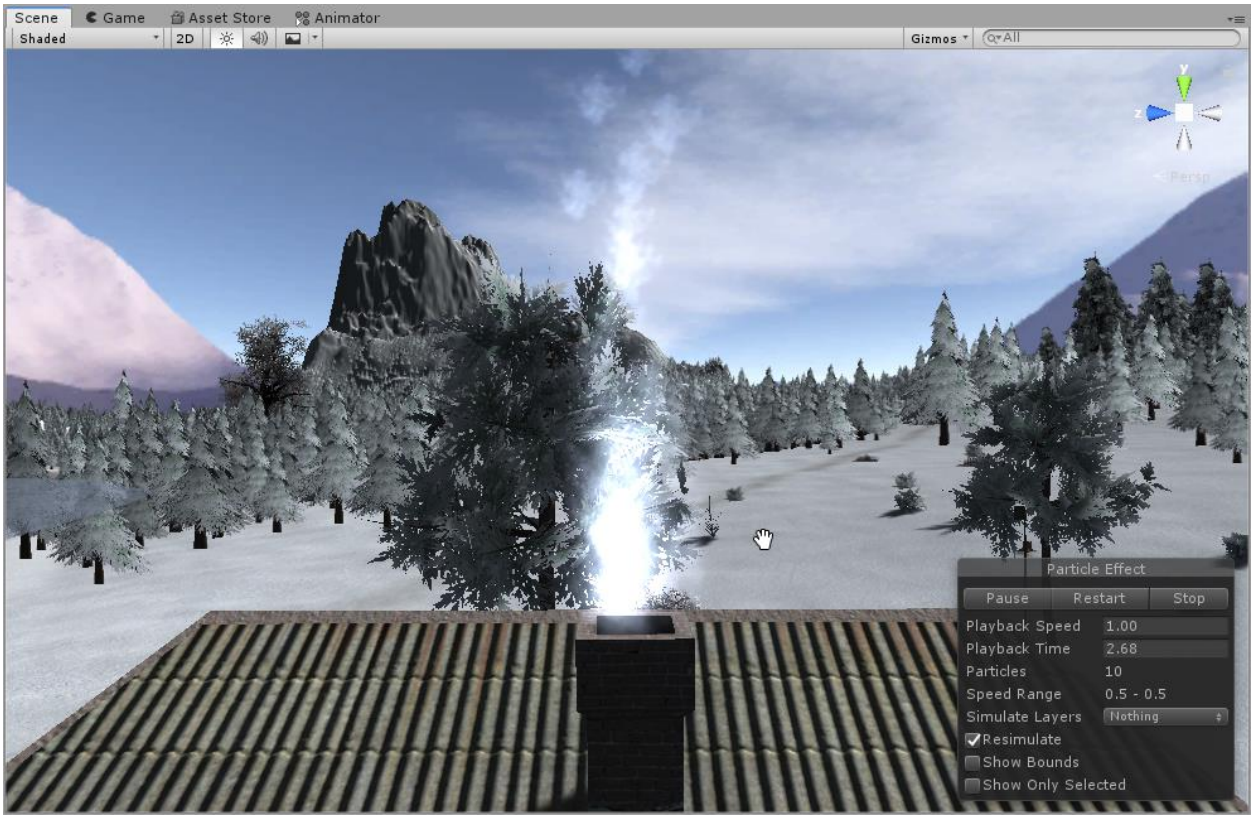

Figure 33. Smoke particle effect

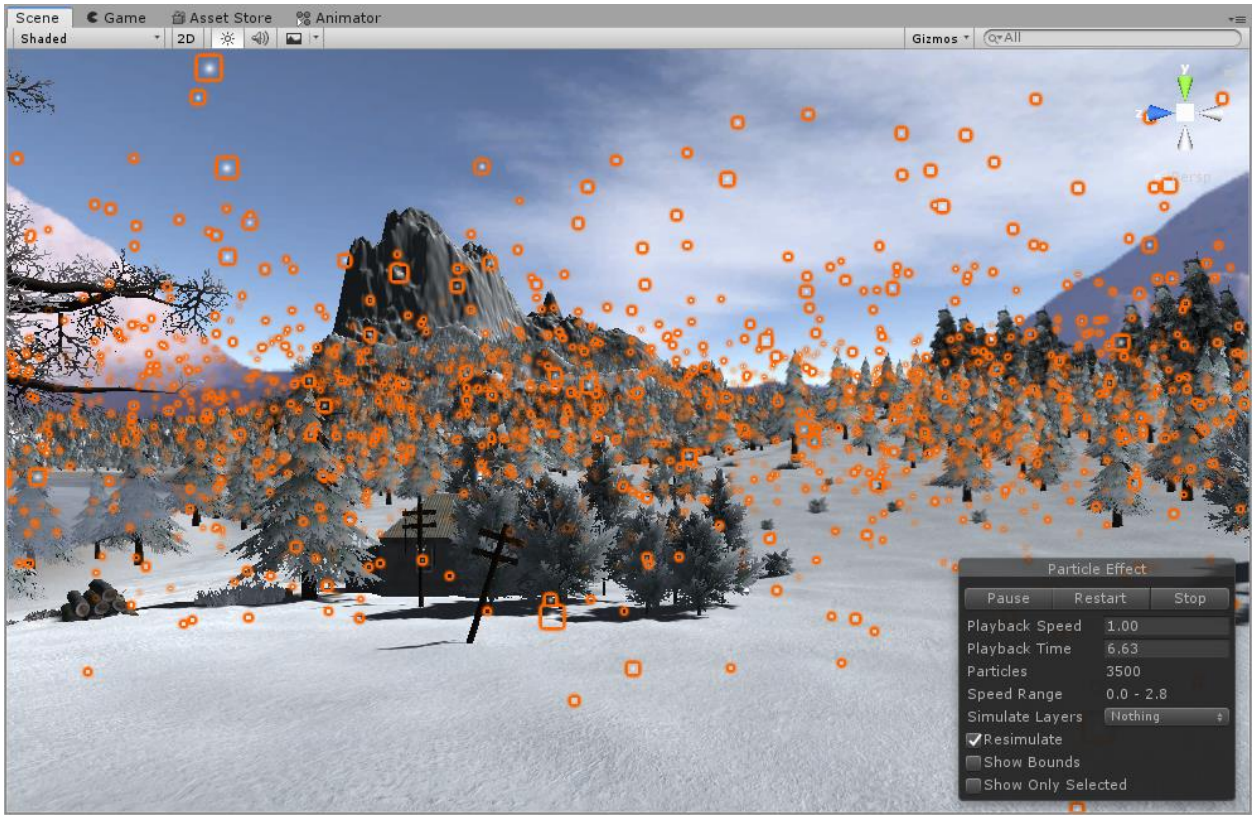

Figure 34. Snow particle effect

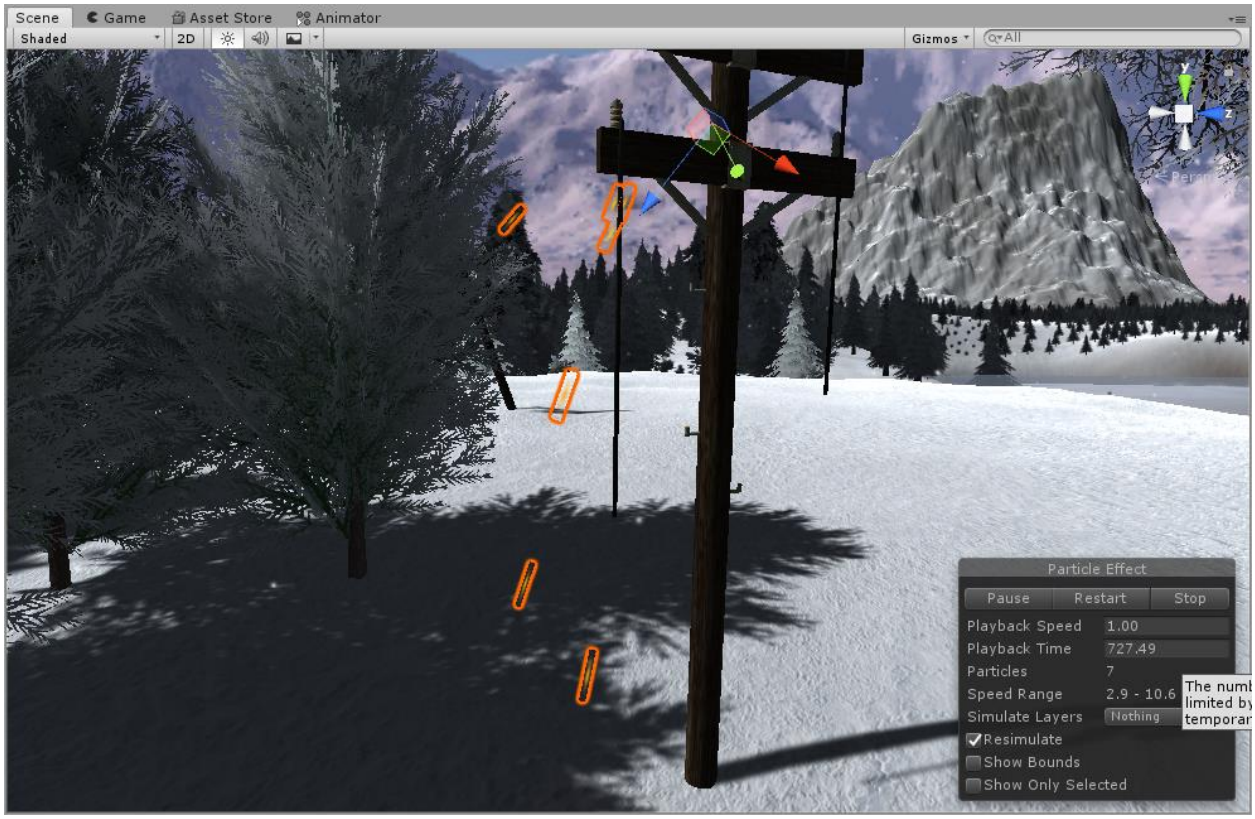

Figure 35. Electric sparkles particle effect

3.2. Add custom skybox

I downloaded mountains panorama picture and divided it into equals parts. Then I assigned them to the skybox

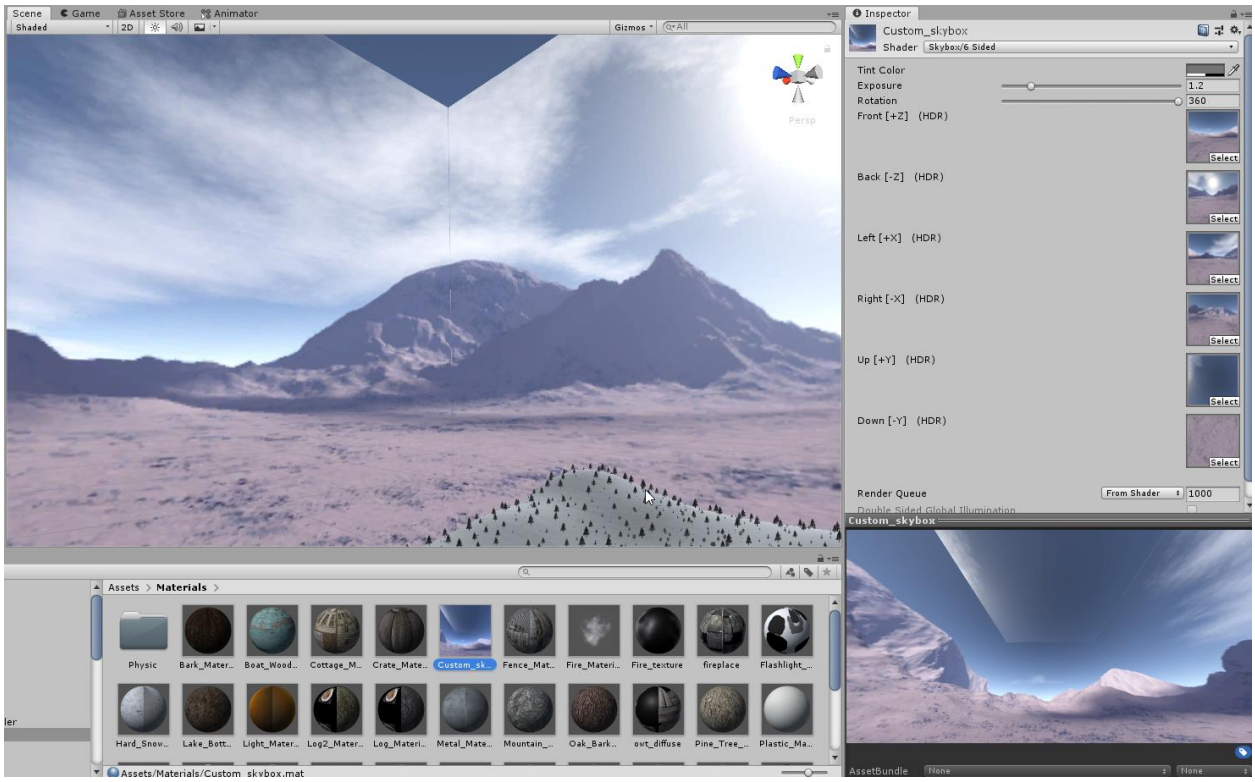

Figure 36. Custom skybox

4. Physics

4.1. Create 5 different physics materials and apply them in to the project

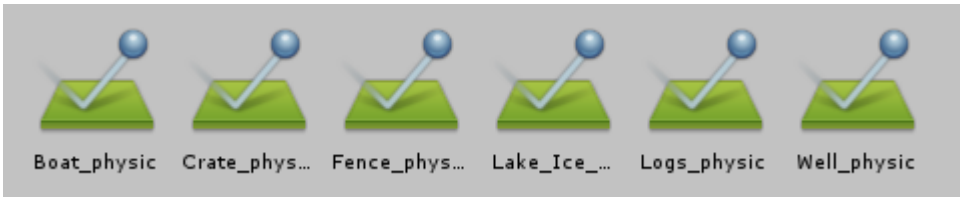

Figure 37. 6 physic materials

4.2. Assign optimal collider based of the shape of the object for all objects in the map

For every object in the map, I assigned a collider based on objects form

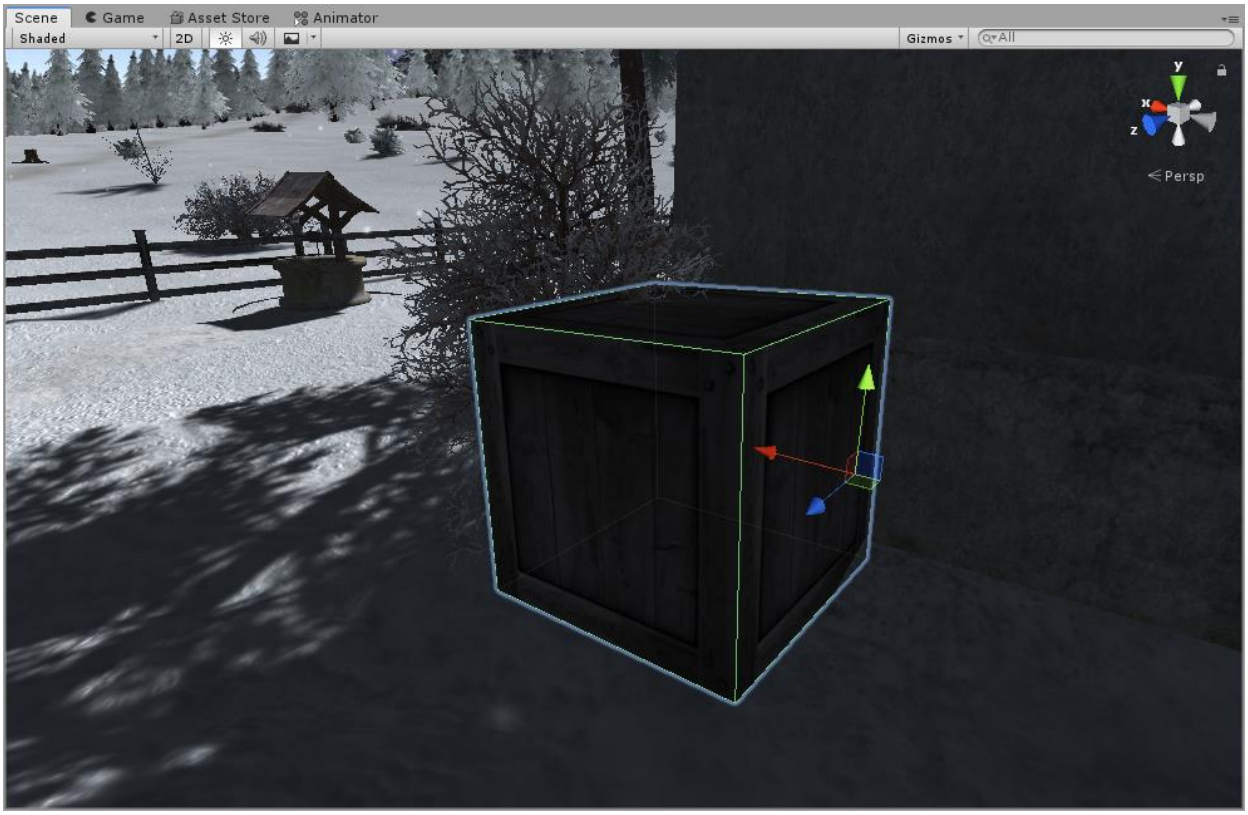

Figure 38. Box collider on crate

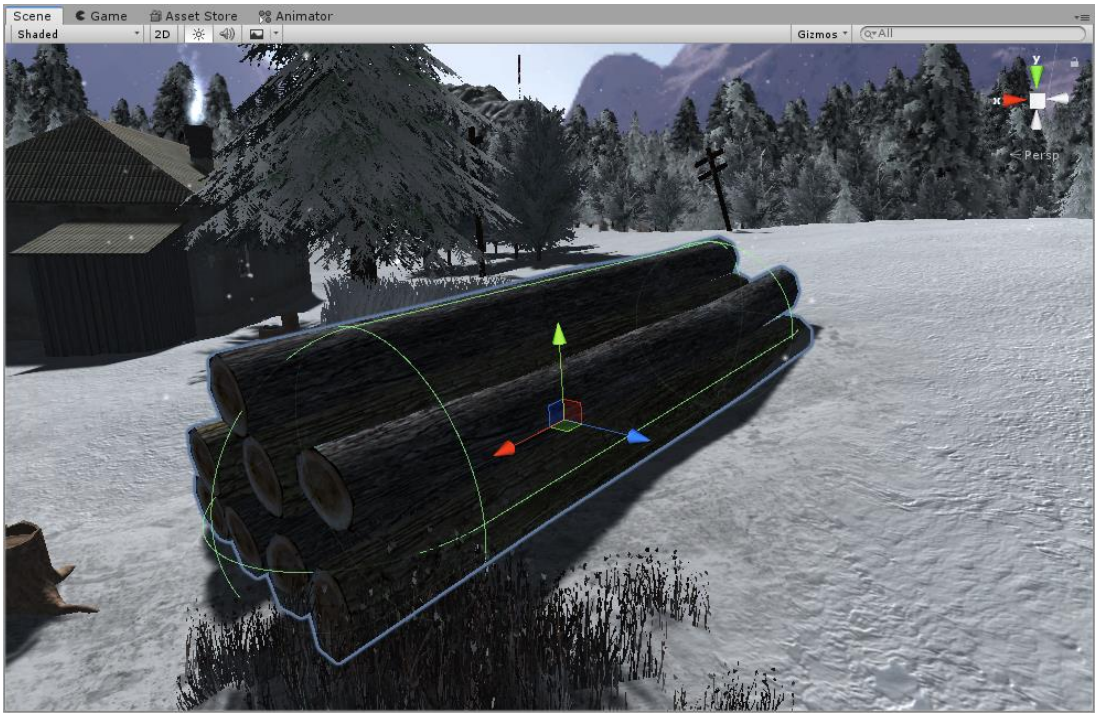

Figure 39. Capsule collider on logs

4.3. Add and animate 5 objects using physics, force and triggers to your game map

Using scripts, trigger collision and player interaction I created some animations

After triggering control object by player, rocks starts falling from the hill (using force)

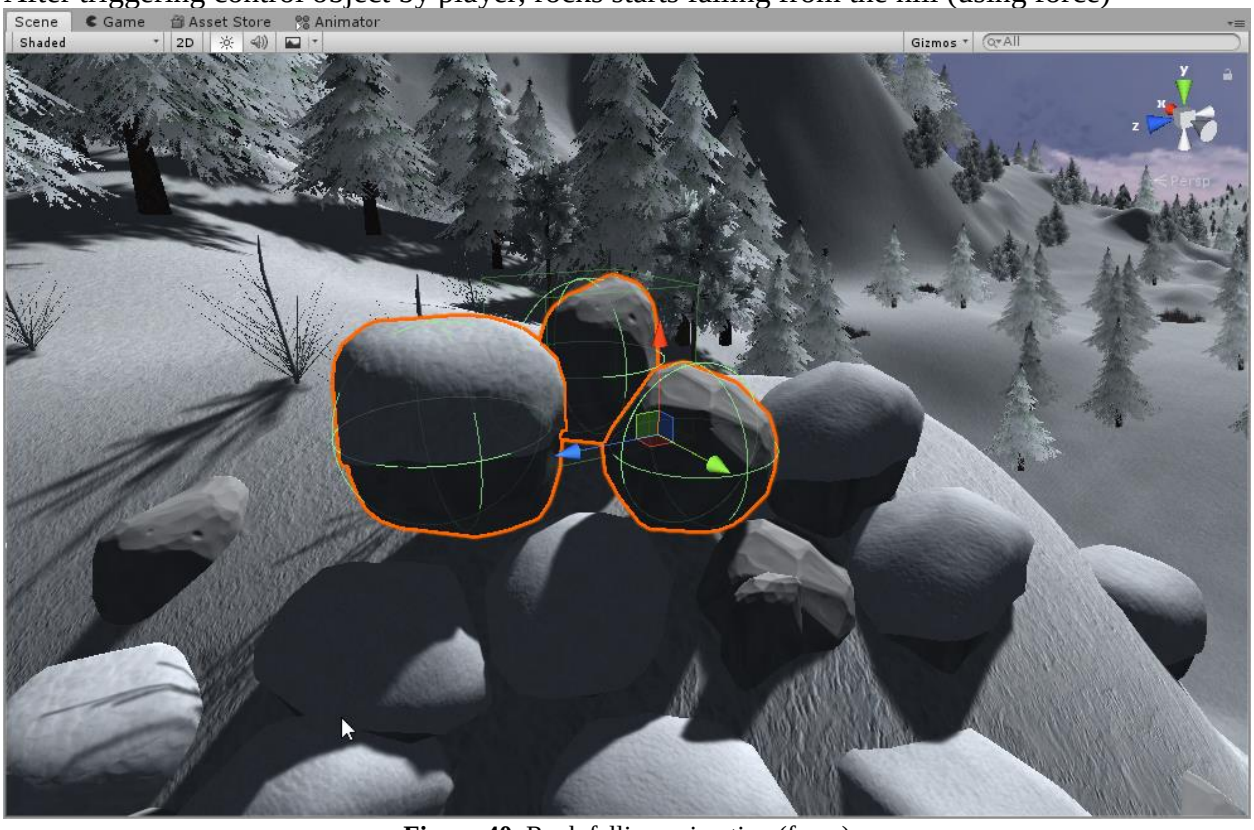

Figure 40. Rock falling animation (force)

After player collision with trigger collision object, ceiling fan starts spinning

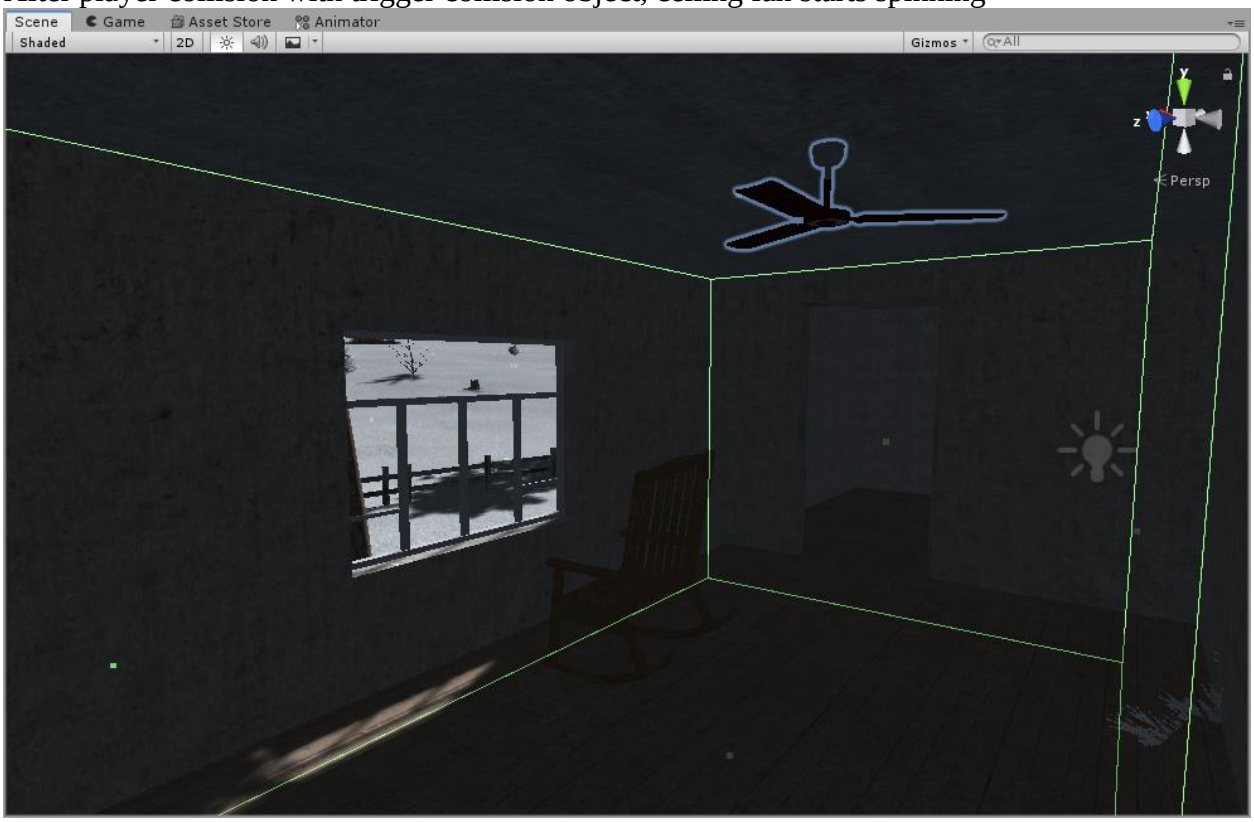

Figure 41. Ceiling fan animation (trigger)

After player interaction with window, window is closed

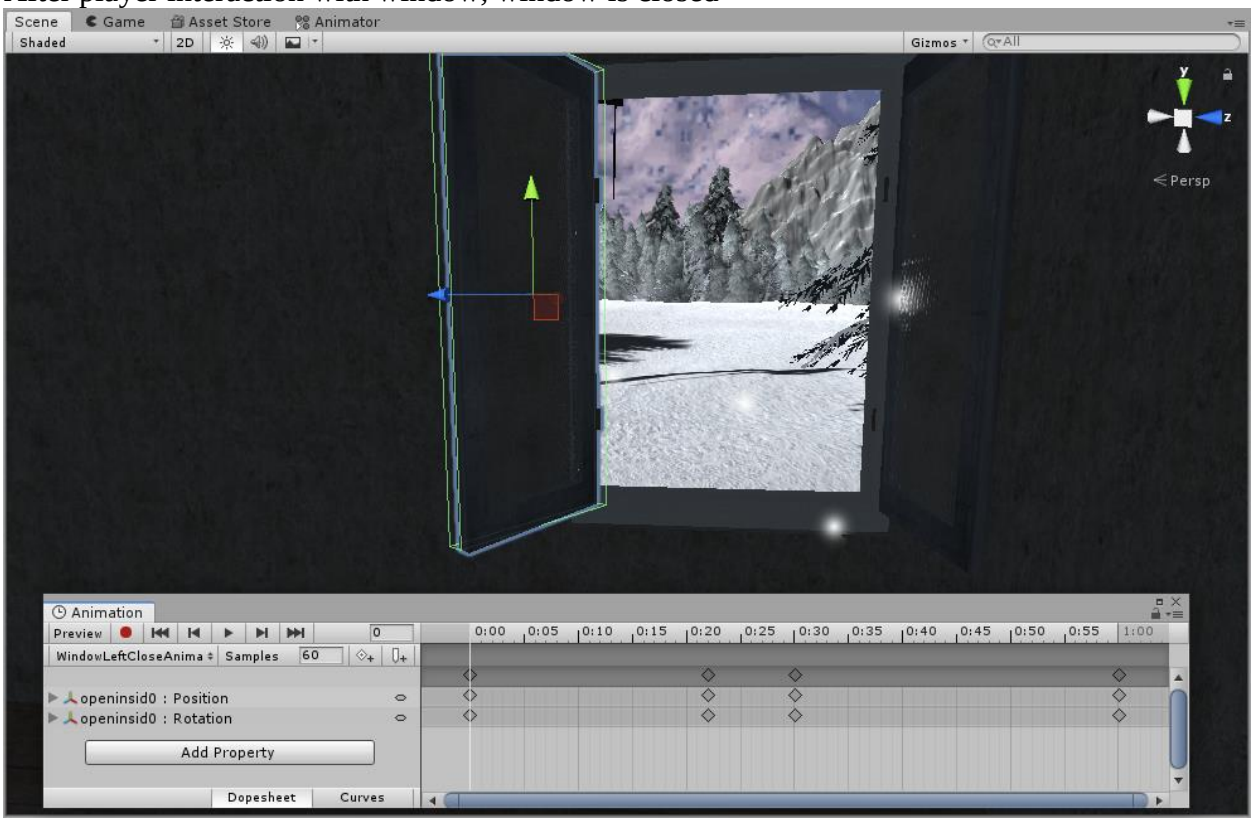

Figure 42. Windows close animation (trigger)

After player interaction with light switch, animation triggers

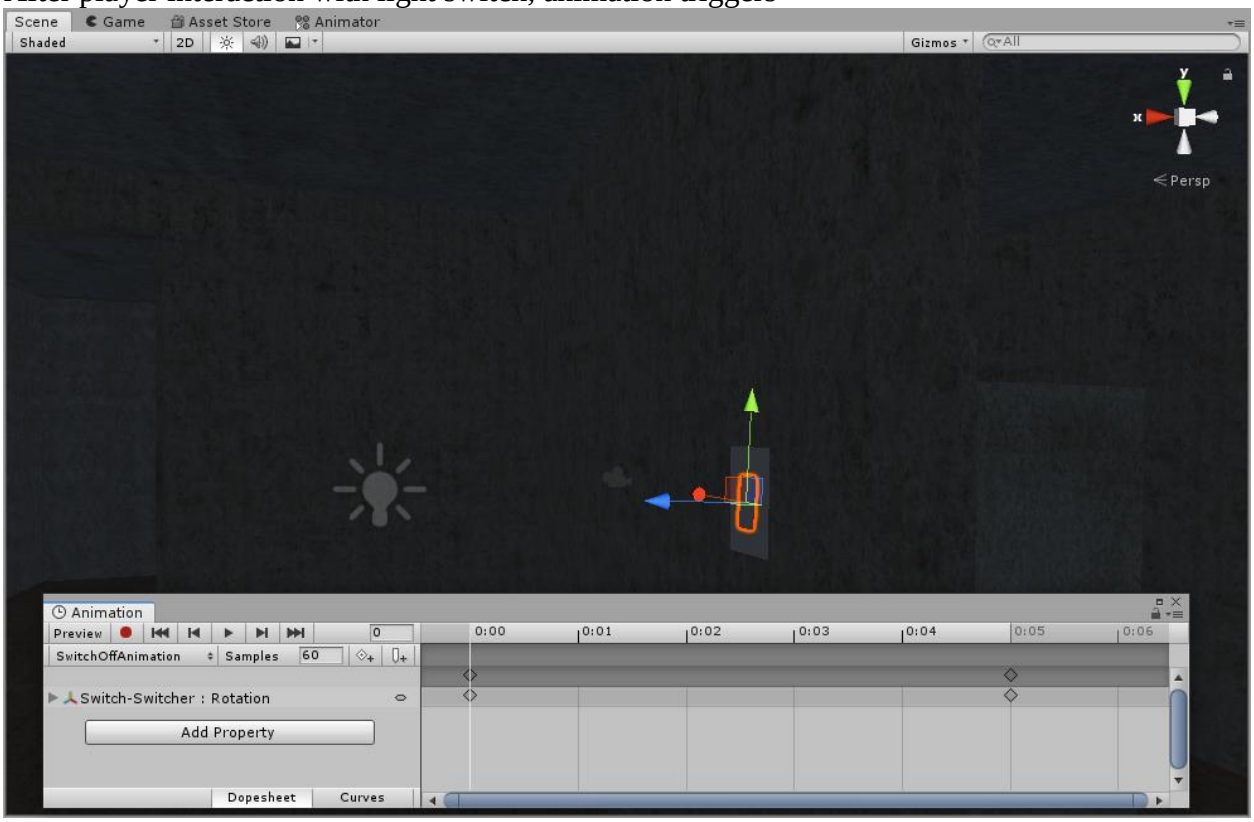

Figure 43. Light switch animation (trigger)

After player interaction with candle, candle gets lit

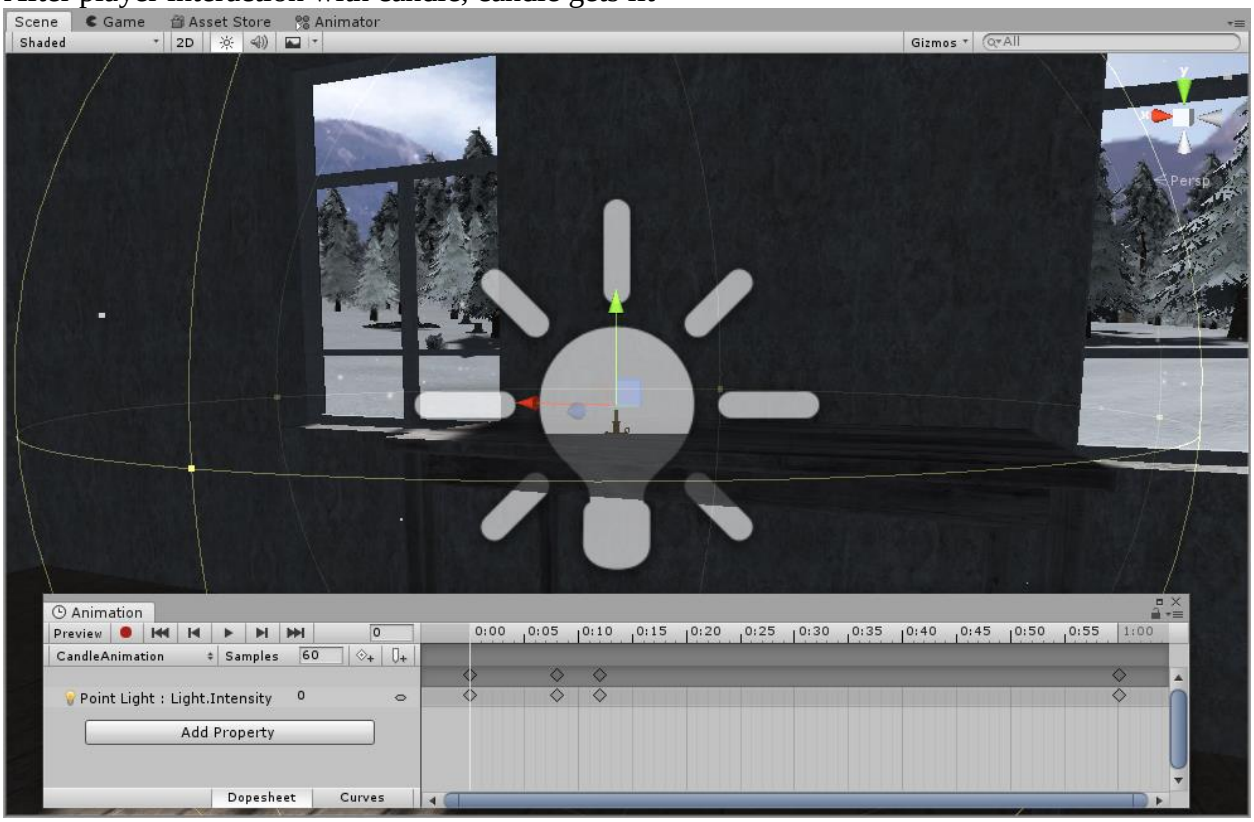

Figure 44. Candle litting animation (trigger)

5. Optimization

-

Laboratory work #3

List of tasks

1. Menu
2. Options
3. GUI
4. Attack mechanics
5. AI implementation
6. Health
7. Scoring system
8. Game over condition
9. High scores / rankings

1. Menu

Main menu has 5 buttons: new game, highscores, options, about and exit. All menus have back/exit buttons.

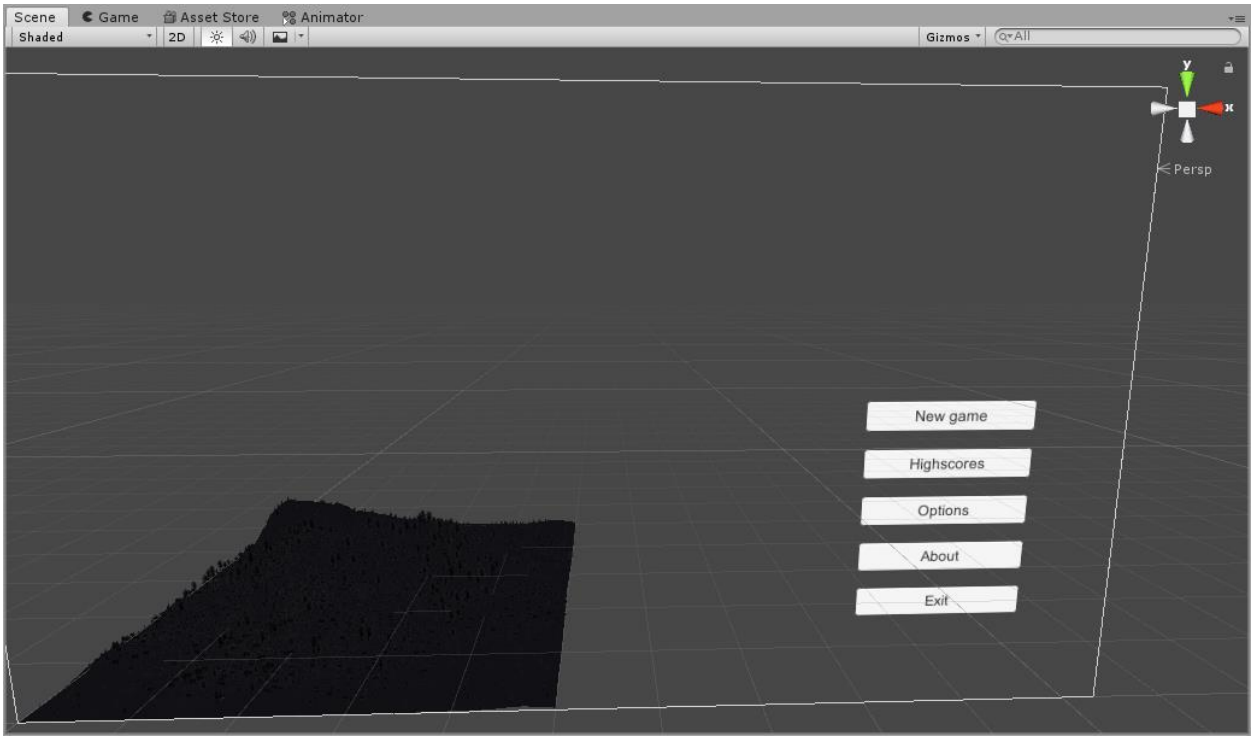

Figure 45. Main menu development window

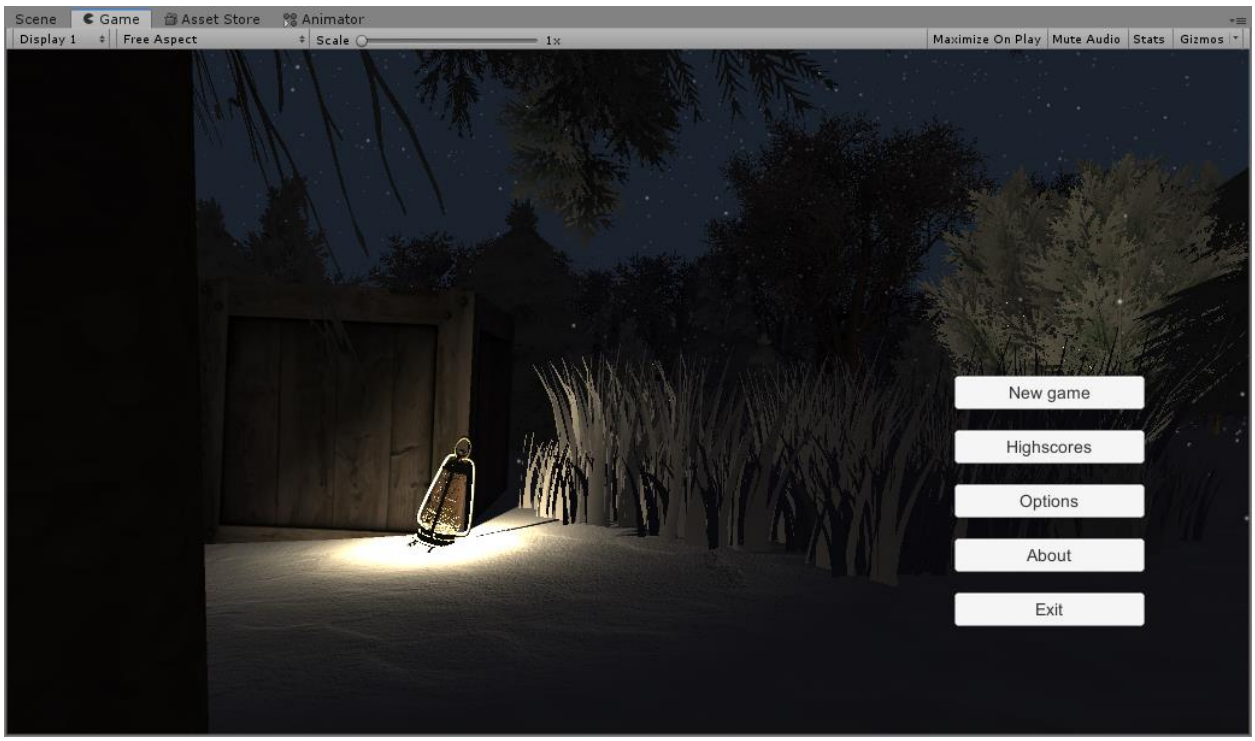

Figure 46. Main menu in-game window

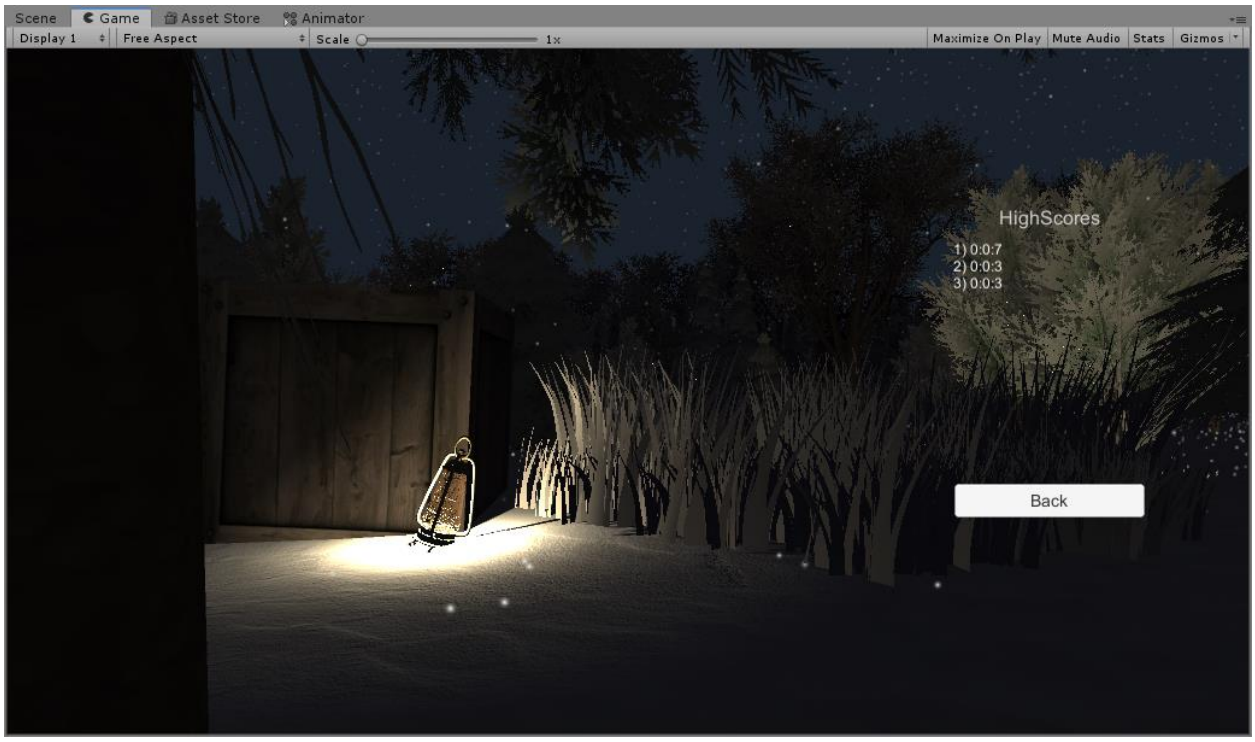

Figure 47. Highscores window

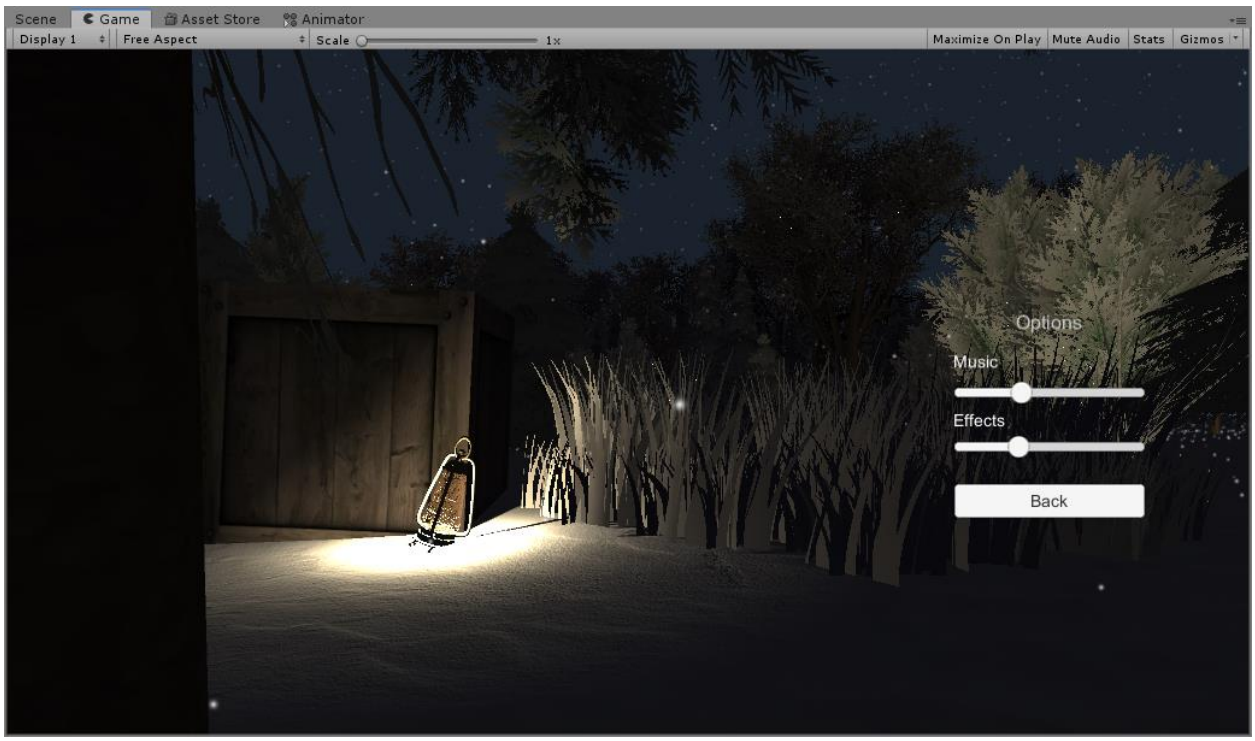

Figure 48. Options window

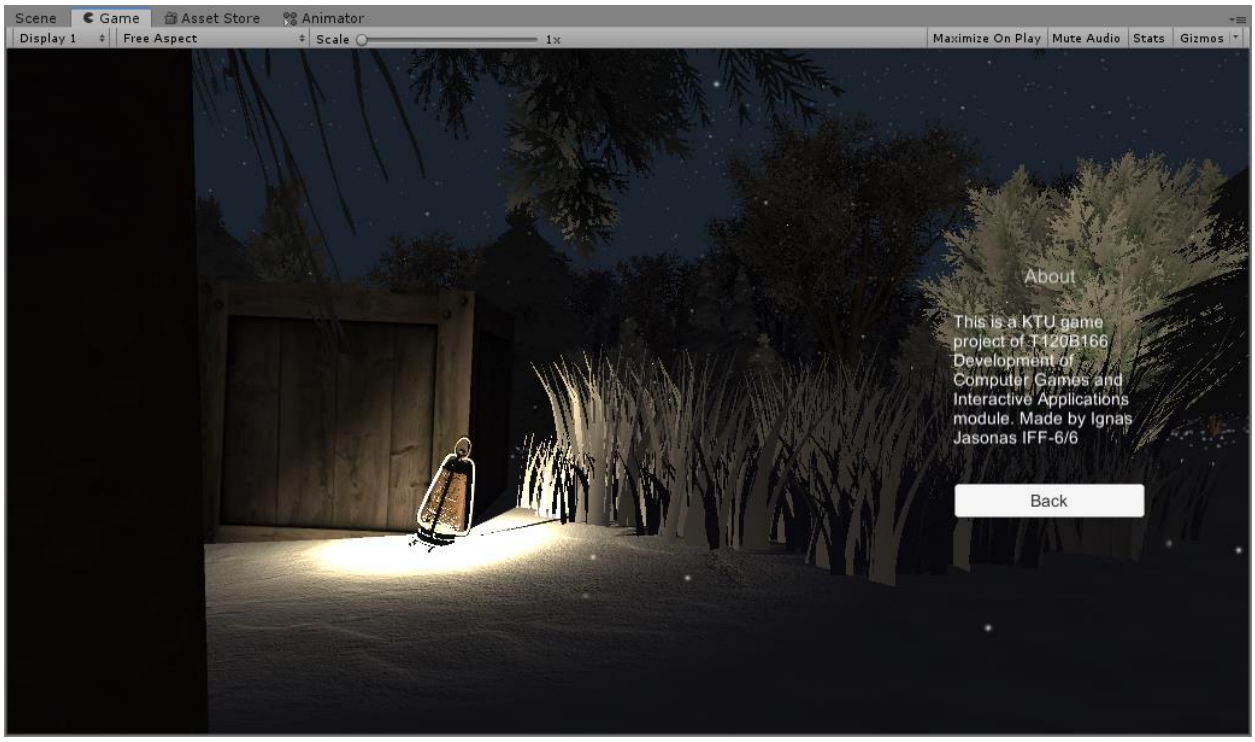

Figure 49. About window

2. Options

Options menu has music and effects audio sliders, which affects the volume of bass, effects and music sound groups.

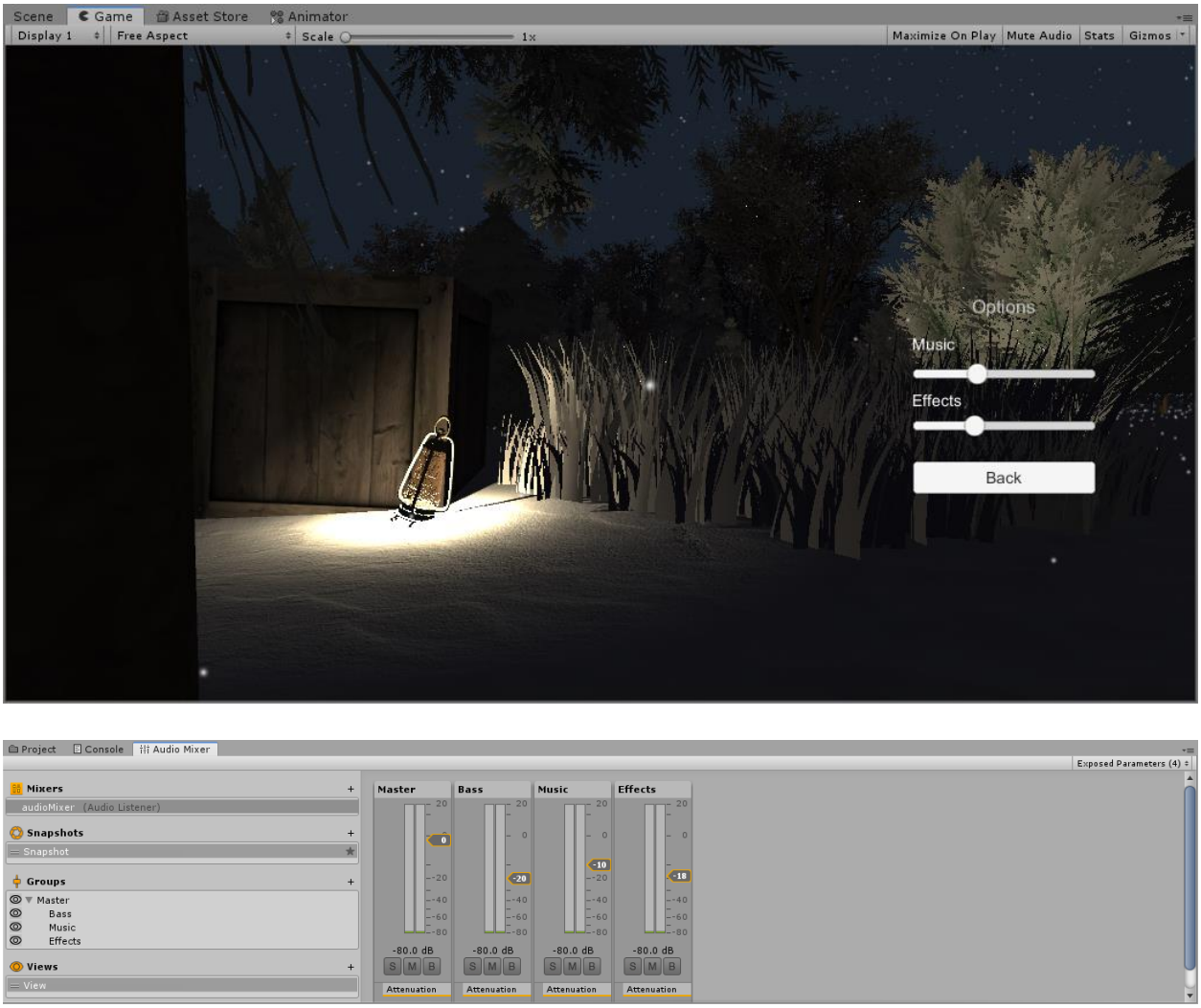

Figure 50. Audio groups

```using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Audio;

public class OptionsMenu : MonoBehaviour
{
    public AudioMixer audioMixer;

    public void SetEffectsVolume(float volume)
    {
        audioMixer.SetFloat("Effects", volume);
    }

    public void SetMusicVolume(float volume)
    {
        audioMixer.SetFloat("Music", volume);
        audioMixer.SetFloat("Bass", volume);
    }
}
```

Figure 51. Audio groups sliders code

3. GUI

Player has health bar and coldness bar on left-top corner of the screen, branches and matches counts on right-top corner of the screen and if the player is near to the interactable object ‘e’ button is displayed in the center of the screen

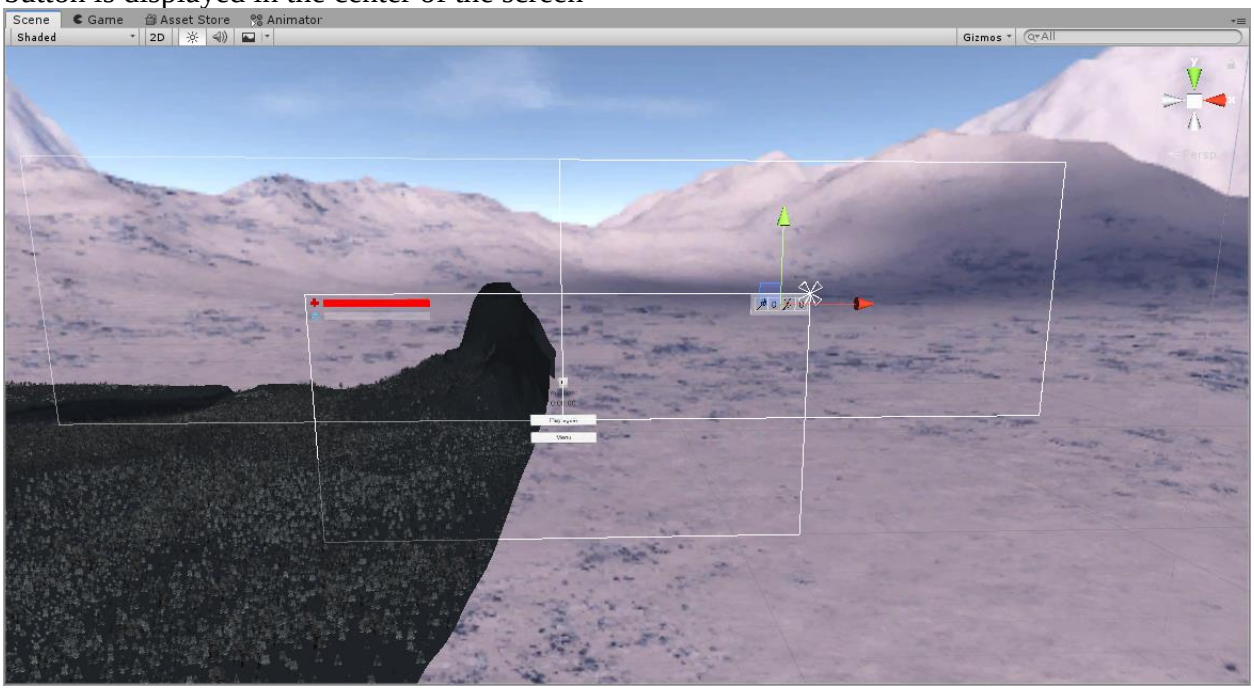

Figure 52. Game GUI development window

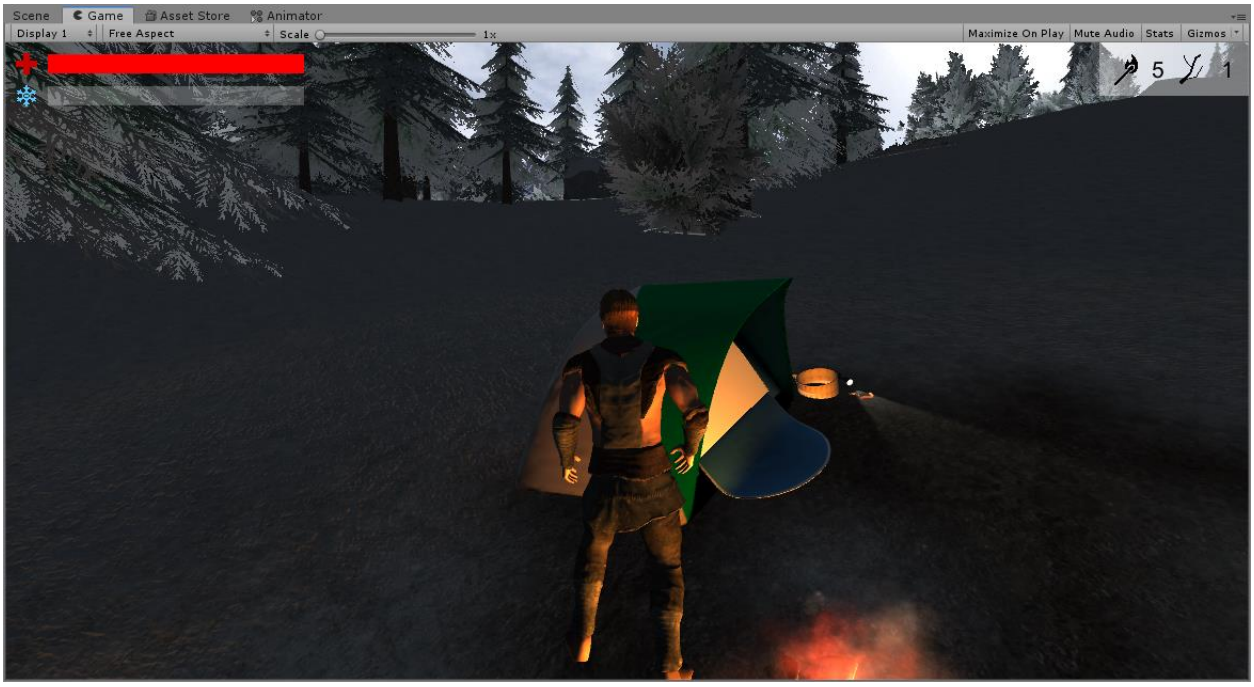

Figure 53. In-game GUI window

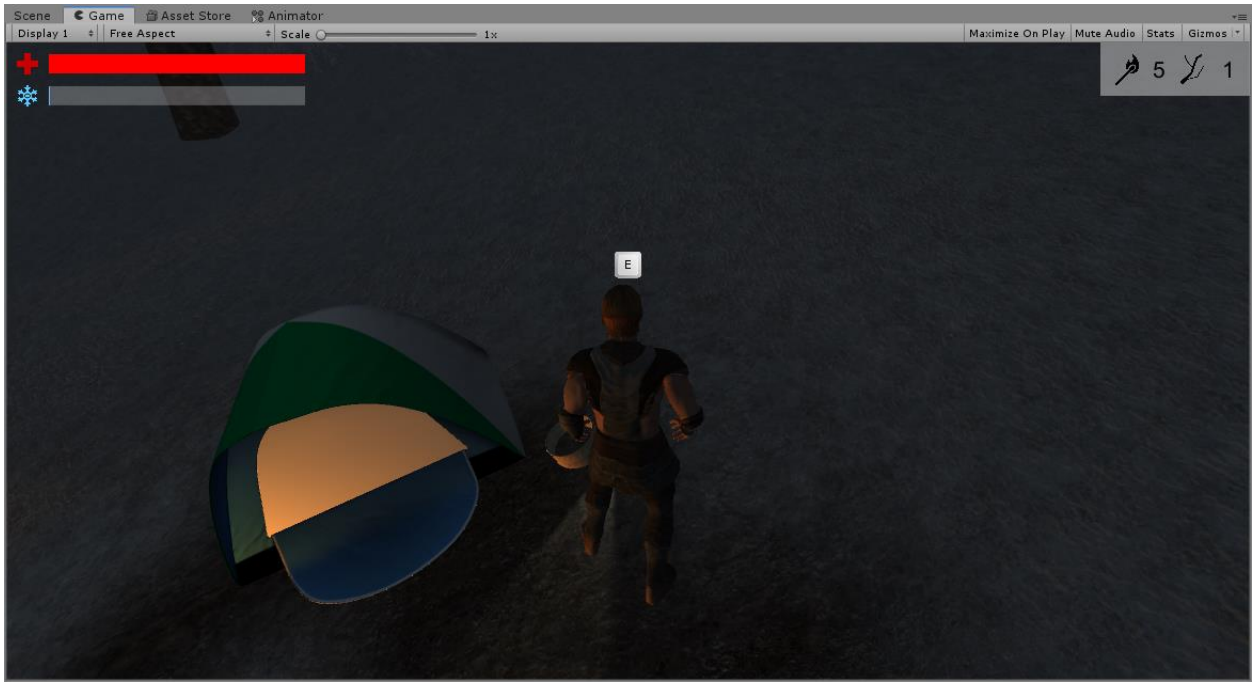

Figure 54. „E“ button display

4. Attack mechanics

This is a game where player can’t attack, he can only run from danger. Demon is NPC which can attack when player is near enough.

Figure 55. Attacking demon

5. AI implementation

Enemy (demon) has AI, which follows player. If player is close enough, he will begin chasing him. If player gets far away, demon will lose his trace and will try to follow player. The more time demon spends searching for player, the closer he will get to him. With searching time

demon aggressiveness level is increasing. If player get near fire, demon is scared away, loses all aggressiveness and will wander around, till player gets away from fire.

```
ing System.Collections;
ing System.Collections.Generic;
ing UnityEngine;

blic enum demonStates: int {
  idle = 0,
  walking = 1,
  attacking = 2,


blic class DemonController : MonoBehaviour

  float gravity = 8;
  const int maxDistNearPoint = 30;
  GameObject player;
  Vector3 moveDir = Vector3.zero;
  CharacterController controller;
  Transform target;
  DemonState demonState;
  GameObject firstAppearance1Point;
  GameObject firstAppearance2Point;
  GameObject startPoint;
  GameObject targetObject;
  public AudioClip[] audioClips;
  PlayerControllerScript playerScript;

  bool firstAppearance = false;
  bool firstAppearanceTriggered = false;

  float aggressionLevel = 0;
  bool chasingPlayer = false;
 void Start()
{
    controller = GetComponent<CharacterController>();
    player = GameObject.FindGameObjectWithTag("Player");
    playerScript = player.GetComponent<PlayerControllerScript>();
    demonState = new DemonState(GetComponent<Animator>(), GetComponents<AudioSource>()[0],
        GetComponents<AudioSource>()[1], audioClips, player, this.gameObject);

    demonState.changeState(demonStates.walking);
    demonState.playerCanTakeDamage = true;

    firstAppearance1Point = GameObject.Find("DemonPoints/FirstAppearance1");
    startPoint = GameObject.Find("DemonPoints/StartPoint");
    firstAppearance2Point = GameObject.Find("DemonPoints/FirstAppearance2");
    targetObject = GameObject.Find("DemonPoints/Target");
    if (firstAppearance) targetObject.transform.position =
            new Vector3(player.transform.position.x + (maxDistNearPoint - aggressionLevel) * (Random.Range(0, 1) * 2 - 1), player.transform.position.y,
            player.transform.position.z + (maxDistNearPoint - aggressionLevel) * (Random.Range(0, 1) * 2 - 1));
    target = targetObject.transform;

    if (!firstAppearance) this.transform.position = firstAppearance1Point.transform.position;
    Invoke("randomSound", 5);
}

 void Update()
{
    if (!firstAppearance) firstAppearanceMode();
    else
    {
        if (!player.GetComponent<PlayerControllerScript>().dead) aggresiveWanderingMode();
        else demonState.changeState(demonStates.idle);
    }
    transform.LookAt(target.position);
    controller.Move(transform.forward * Time.deltaTime * demonState.speed);
    var gravityForce = new Vector3(0, -1 * gravity, 0);
    controller.Move(gravityForce * Time.deltaTime);
}
 public void firstAppearanceMode()
{
    if (!firstAppearanceTriggered)
    {
        if (Vector3.Distance(this.transform.position, player.transform.position) <= 70)
        {
            demonState.changeState(demonStates.walking);
            target = firstAppearance2Point.transform;
            firstAppearanceTriggered = true;
        }
    }
    else
    {
        if (Vector3.Distance(this.transform.position, firstAppearance2Point.transform.position) <= 2)
        {
            firstAppearance = true;
            this.transform.position = startPoint.transform.position;
            targetObject.transform.position =
                new Vector3(player.transform.position.x + (maxDistNearPoint - aggressionLevel) * (Random.Range(0, 1) * 2 - 1),
                player.transform.position.y, player.transform.position.z + (maxDistNearPoint - aggressionLevel) * (Random.Range(0, 1) * 2 - 1));
        }
    }
}
```

```
public void aggresiveWanderingMode()
{
        if (Vector3.Distance(this.transform.position, player.transform.position) <= 1.5)
        {
            demonState.changeState(demonStates.attacking);
        }
        else
        {
            if ((!chasingPlayer && (Mathf.Abs(this.transform.position.x - target.transform.position.x) +
            Mathf.Abs(this.transform.position.z - target.transform.position.z)) <= 12)
                || (chasingPlayer && Vector3.Distance(this.transform.position, player.transform.position) > 10) ||
                (chasingPlayer && playerScript.nearWarm > 0))
            {
            if (!chasingPlayer) increaseAggression();
            targetObject.transform.position = new Vector3(player.transform.position.x +
                (maxDistNearPoint - aggressionLevel) * (Random.Range(0, 1) * 2 - 1), player.transform.position.y, player.transform.position.z +
                (maxDistNearPoint - aggressionLevel) * (Random.Range(0, 1) * 2 - 1));
            target = targetObject.transform;
            chasingPlayer = false;
        }
        else if (!chasingPlayer && Vector3.Distance(this.transform.position, player.transform.position) <= 10 && playerScript.nearWarm <= 0)
        {
            target = player.transform;
            chasingPlayer = true;
        }


    }
    demonState.demonAttackUpdate();
}
```

```
public void randomSound()
{
    demonState.randomSound();
    Invoke("randomSound", Random.Range(4f, 10f));
}

public void increaseAggression()
{
    if (aggressionLevel < maxDistNearPoint - 5)
    {
        aggressionLevel += 5;
    }
    if (playerScript.nearWarm > 0) aggressionLevel = 5;
}
```

```
public class DemonState: MonoBehaviour
{
    public bool playerCanTakeDamage = true;
    public float speed = 0;
    public demonStates currentState = demonStates.idle;
    public demonStates previousState = 0;
    private Animator anim;
    private AudioSource audio;
    private AudioSource walkingAudio;
    public AudioClip[] audioClips;
    private AudioClip[] currentClips;
    private GameObject player;
    private GameObject demon;


    public DemonState(Animator Anim, AudioSource Audio, AudioSource WalkingAudio, AudioClip[] AudioClips, GameObject Player, GameObject Demon)
    {
        anim = Anim;
        audio = Audio;
        walkingAudio = WalkingAudio;
        audioClips = AudioClips;
        player = Player;
        demon = Demon;
        playerCanTakeDamage = true;
        changeState(demonStates.idle);
    }
```

```
public void changeState(demonStates state)
{
    if (currentState == demonStates.attacking)
    {
        if (anim.GetCurrentAnimatorStateInfo(0).normalizedTime > 1 && !anim.IsInTransition(0))
        {
            currentState = demonStates.walking;
            switch (state)
            {
                case demonStates.idle:
                    speed = 0;
                    currentClips = new AudioClip[2] { audioClips[0], audioClips[1] };
                    break;
                case demonStates.walking:
                    walkingAudio.Play();
                    currentClips = new AudioClip[2] { audioClips[0], audioClips[1] };
                    speed = 2.4f;
                    break;
                case demonStates.attacking:
                    walkingAudio.Stop();
                    speed = 0f;
                    break;
                default:
                    speed = 0;
                    break;
            }
            anim.SetInteger("state", (int)currentState);
            playerCanTakeDamage = true;
        }
    }
    else
    {
        if (!(currentState == demonStates.attacking && anim.GetCurrentAnimatorStateInfo(0).normalizedTime > 1 && !anim.IsInTransition(0)))
        {
            currentState = state;
            switch (state)
            {
                case demonStates.idle:
                    speed = 0f;
                    currentClips = new AudioClip[2] { audioClips[0], audioClips[1] };
                    break;
                case demonStates.walking:
                    walkingAudio.Play();
                    currentClips = new AudioClip[2] { audioClips[0], audioClips[1] };
                    speed = 2.6f;
                    break;
                case demonStates.attacking:
                    walkingAudio.Stop();
                    audio.clip = currentClips[1];
                    audio.Play();
                    speed = 0f;
                    break;
                default:
                    speed = 0f;
                    break;
            }
            anim.SetInteger("state", (int)currentState);
        }
    }
}
```

```
public void demonAttackUpdate() {
    if (currentState == demonStates.attacking)
    {
        if (playerCanTakeDamage == true && Vector3.Distance(demon.transform.position, player.transform.position) <= 1.5 &&
            anim.GetCurrentAnimatorStateInfo(0).normalizedTime <= 0.3f && anim.GetCurrentAnimatorStateInfo(0).normalizedTime >= 0.05f)
        {
            player.GetComponent<PlayerControllerScript>().takeDamage();
            playerCanTakeDamage = false;
        }
        if (anim.GetCurrentAnimatorStateInfo(0).normalizedTime > 1 && !anim.IsInTransition(0))
        {
            if (Vector3.Distance(demon.transform.position, player.transform.position) > 1.5) changeState(demonStates.walking);
            playerCanTakeDamage = true;

        }
    }
}

public void randomSound()
{
    audio.clip = currentClips[0];
    audio.Play();
}
```

Figure 56. Demon AI implementation code

6. Health

Player has 100 health which can be taken down by demon attacks and cold weather. If player reaches 0, he dies. Player can regenerate health by standing near fire.

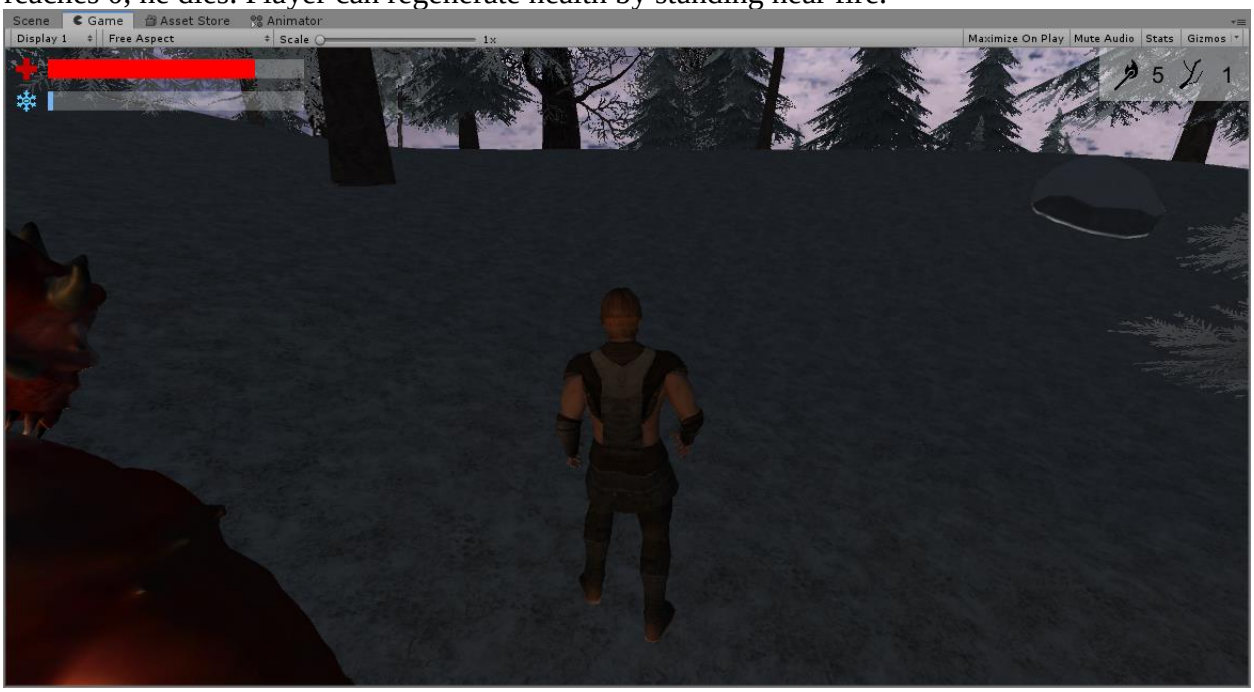

Figure 57. Player lost some health

7. Scoring system

Player main objective is to survive as long as possible. So his score is total survived time.

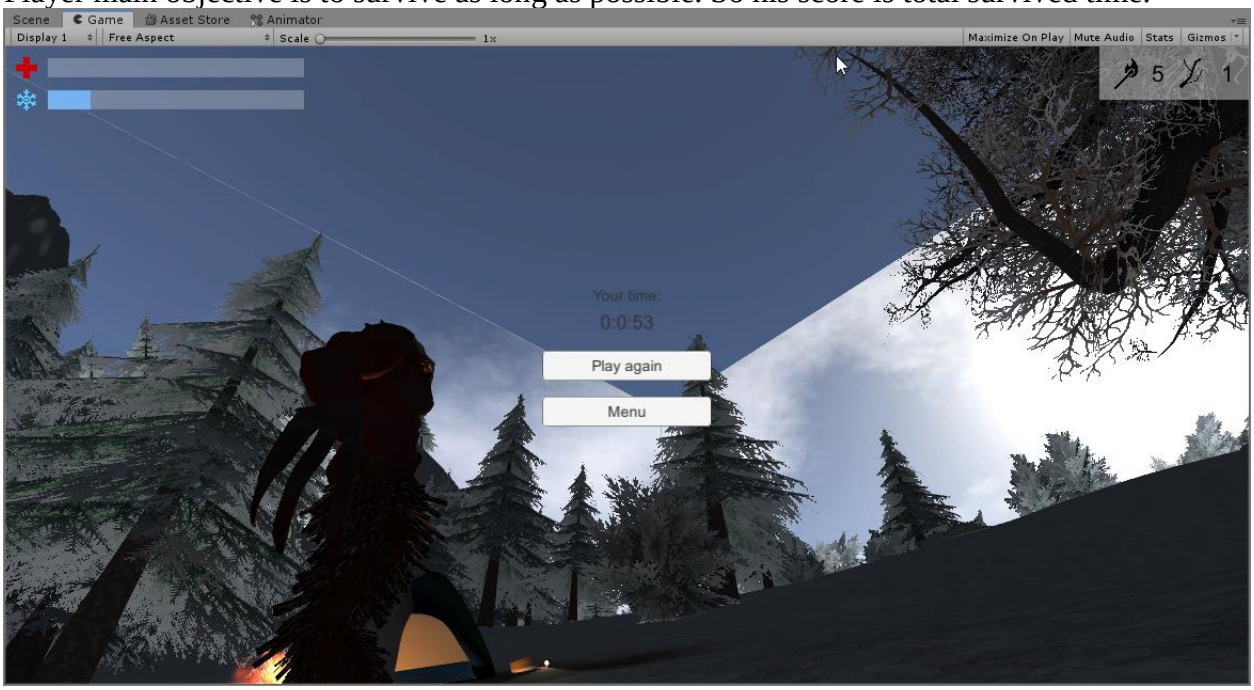

Figure 58. Game over score window

8. Game over condition

Game over condition is for player to reach 0 health.

```
public void takeDamage()
{
    painSource.Play();
    if (health - 10 > 0) health -= 10;
    else
    {
        Die();
        health = 0;
    }
    healthBar.UpdateBar(health, 100);
}
public void coldEffectDamage()
{
    int damage = 0;
    if (nearWarm <= 0)
    {
        if (cold >= 6000 && cold <= 7000)
        {
            damage = 1;
        }
        else if (cold > 7000 && cold <= 8500)
        {
            damage = 2;
        }
        else if (cold > 8500)
        {
            damage = 3;
        }
        if (health - damage > 0) health -= damage;
        else
        {
            Die();
            health = 0;
        }
    healthBar.UpdateBar(health, 100);
    }
    Invoke("coldEffectDamage", 5f);
}
```

Figure 59. Game over condition code

9. High scores and rankings

After dying player total survived time (score) is saved in text file outside code. So if game is re-opened, score still is available. Game save total of 10 highscores.

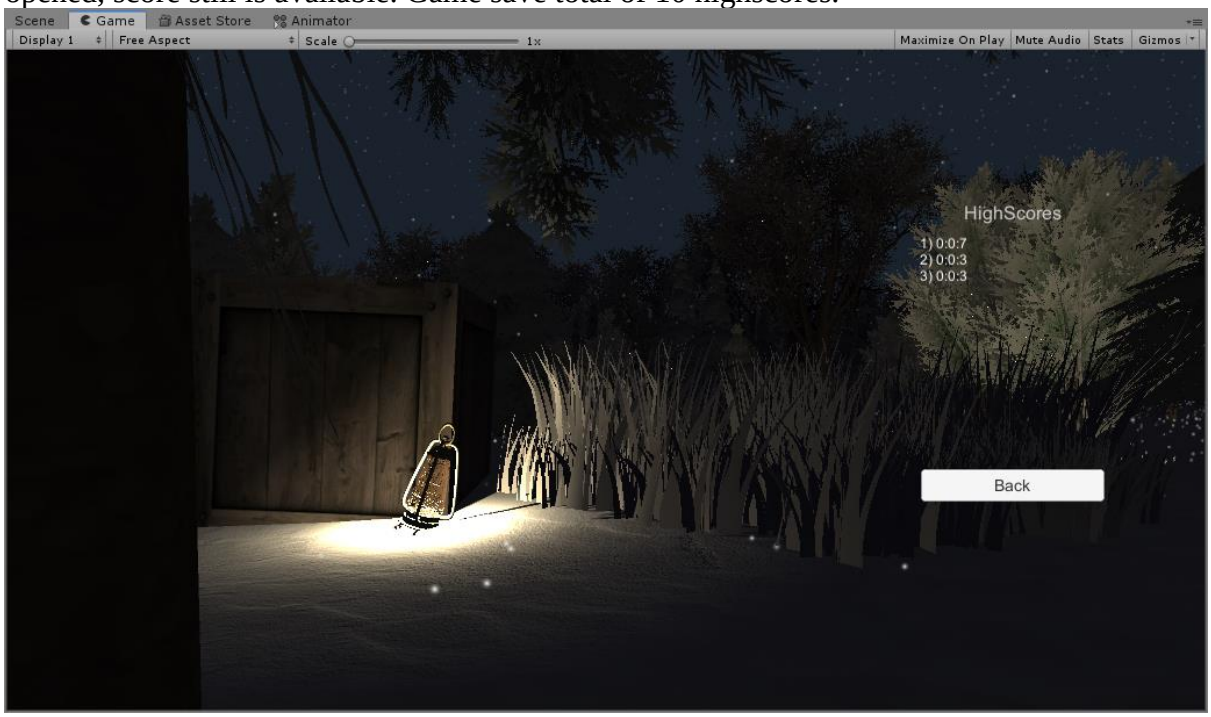

Figure 60. Highscores window

Literature list

1. *Tent by* **printable_models** https://free3d.com/3d-model/green-mesh-tent-v1--54743.html
2. *Axe by* **ffmstudios** https://free3d.com/3d-model/axe-11562.html
3. *Logs by* **ancel** https://free3d.com/3d-model/logs--2256.html
4. *Flashlight by* **pedro_h_br** https://free3d.com/3d-model/simple-flashlight-70024.html
5. *Power pole by* **conicstudios** https://free3d.com/3d-model/wooden-power-pole-65361.html
6. *Fence by* **mapelisin111** https://free3d.com/3d-model/fence-snow-220713.html
7. *Crate by* **tyrosmith** https://free3d.com/3d-model/crate-86737.html
8. *Watchtower by* **crazyclown** https://free3d.com/3d-model/2-wood-towers-65990.html
9. *Boat by* **ahmetsalih** https://free3d.com/3d-model/boat-37697.html
10. *Well by* **animatedheaven** https://free3d.com/3d-model/old-well-low-polygon-440593.html
11. *Cooking pot by* **noah28** *https://3dwarehouse.sketchup.com/model/874e884e24d060ea7d3fac2e7dde0c57/cook-pot*
12. *Winterland package by* **Shapes** *https://assetstore.unity.com/packages/3d/environments/winterland-61174 (Unity assets)*
13. Ceiling fan by **storque12** https://free3d.com/3d-model/ceiling-fan-88692.html
14. Antique candlestick by **tyrosmith** https://free3d.com/3d-model/antique-candlestick-73781.html
15. Communication tower by **gunnarcorrea** https://free3d.com/3d-model/communication-tower-12911.html
16. Table by **sniefy** https://free3d.com/3d-model/cinema4d-table-66762.html
17. Window by piratemorgan https://www.turbosquid.com/3d-models/3d-model-italian-window/491122
18. Rocking chair by **ideastudio** https://free3d.com/3d-model/annie-wooden-rocking-chair-79547.html
19. Campfire pack by Dramdev Studios https://assetstore.unity.com/packages/3d/environments/fantasy/campfire-pack-11256
20. Survival game tools package by cookiepopworks.com https://assetstore.unity.com/packages/3d/props/tools/survival-game-tools-139872

ANNEX

All source code is contained in this part.

AfterGame.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class AfterGame : MonoBehaviour
{

    public void Restart()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
    }

    public void BackToMainMenu()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex - 1);
    }
}
```

bonfireControllerScript.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class bonfireControllerScript : MonoBehaviour
{
    GameObject player;
    bool playerNotified = false;
    bool playerNearWarmNotified = false;
    PlayerControllerScript playerScript;
    public AudioClip fire;
    public AudioClip fireGoesOut;
    public AudioClip matchSound;
    AudioSource audio;
    bool isBurning = false;
    bool isTryingToStart = false;
    GameObject fireEffect;
    Light fireLight;
    float addition;

    void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player");
        fireEffect = transform.Find("fire").gameObject;
        fireEffect.SetActive(false);
        audio = GetComponent<AudioSource>();
        playerScript = player.GetComponent<PlayerControllerScript>();
        fireLight = GetComponent<Light>();
    }

    // Update is called once per frame
    void Update()
    {
        if (isBurning)
        {
            addition = Random.Range((float)-0.55, (float)0.55);
            if (fireLight.intensity + addition > 4 && fireLight.intensity + addition <
12) fireLight.intensity += addition;
```

```
            if (Vector3.Distance(this.transform.position, player.transform.position) <=
3.5f)
            {
                if (!playerNearWarmNotified)
                {
                    playerScript.nearWarm++;
                    playerNearWarmNotified = true;
                }

            }
            else
            {
                if (playerNearWarmNotified)
                {
                    playerScript.nearWarm--;
                    playerNearWarmNotified = false;
                }
            }
        }
        if (Vector3.Distance(this.transform.position, player.transform.position) <= 1.2f)
        {
            if (!isTryingToStart && !isBurning)
            {
                if (!playerNotified)
                {
                    playerScript.interactablesNear++;
                    playerNotified = true;
                }
            }
            if (isBurning)
            {
                if (playerNotified)
                {
                    playerScript.interactablesNear--;
                    playerNotified = false;
                }
            }
        }
        else
        {
            if (playerNotified)
            {
                playerScript.interactablesNear--;
                playerNotified = false;
            }

        }
        if (Input.GetKeyDown("e"))
        {
            if (Vector3.Distance(this.transform.position, player.transform.position) <=
1.2f && !isBurning && !isTryingToStart)
            {
                if (playerScript.matches > 0)
                {
                    playerScript.updateMatches(-1);
                    tryToStartFire();
                }
            }

        }
    }

    public void tryToStartFire()
    {
        isTryingToStart = true;
```

```
        audio.clip = matchSound;
        audio.Play();

        Invoke("tryToStartFireResult", Random.Range(1.5f, 3f));
    }

    public void tryToStartFireResult()
    {
        int success = Random.Range(0, 17);
        if (success <= 4)
        {
            audio.clip = fireGoesOut;
            audio.Play();
        }
        else
        {
            fireEffect.SetActive(true);
            fireLight.intensity = 7;
            isBurning = true;
            audio.clip = fire;
            audio.loop = true;
            audio.Play();
            Invoke("goOut", Random.Range(7f, 30f));
        }
        isTryingToStart = false;
    }

    public void goOut()
    {
        isBurning = false;
        audio.loop = false;
        audio.Stop();
        audio.clip = fireGoesOut;
        audio.Play();
        Invoke("destroy", 1f);
    }

    public void destroy()
    {
        if (playerNotified)
        {
            playerScript.interactablesNear--;
            playerNotified = false;
        }
        if (playerNearWarmNotified)
        {
            playerScript.nearWarm--;
            playerNearWarmNotified = false;
        }
        Destroy(gameObject);
    }
}
```

branchController.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class branchController : MonoBehaviour
{
    GameObject player;
    bool playerNotified = false;
    PlayerControllerScript playerScript;
    // Start is called before the first frame update
    void Start()
```

```
    {
        player = GameObject.FindGameObjectWithTag("Player");
        playerScript = player.GetComponent<PlayerControllerScript>();
    }

    // Update is called once per frame
    void Update()
    {
        if (Vector3.Distance(this.transform.position, player.transform.position) <= 2.5f)
        {
            if (!playerNotified)
            {
                playerScript.interactablesNear++;
                playerNotified = true;
            }
        }
        else
        {
            if (playerNotified)
            {
                playerScript.interactablesNear--;
                playerNotified = false;
            }
        }
        if (Input.GetKeyDown("e"))
        {
            if (Vector3.Distance(this.transform.position, player.transform.position) <=
2.5f)
            {

                if (playerScript.branches < 99)
                {
                    playerScript.updateBranches(1);
                    if (playerNotified)
                    {
                        playerScript.interactablesNear--;
                        playerNotified = false;
                    }
                    Destroy(gameObject);
                }
            }

        }
    }
}
```

CeilingFanTrigger.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CeilingFanTrigger : MonoBehaviour
{
    [SerializeField] private Animator animationController;
    [SerializeField] private Animator animationController1;
    [SerializeField] private Animator animationController2;

    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            animationController.SetBool("play", true);
            animationController1.SetBool("play", true);
            animationController2.SetBool("play", true);
        }
```

```
    }

    private void OnTriggerExit(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            animationController.SetBool("play", false);
            animationController1.SetBool("play", false);
            animationController2.SetBool("play", false);
        }
    }
}
```

DemonController.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public enum demonStates: int {
    idle = 0,
    walking = 1,
    attacking = 2,
}

public class DemonController : MonoBehaviour
{
    float gravity = 8;
    const int maxDistNearPoint = 30;
    GameObject player;
    Vector3 moveDir = Vector3.zero;
    CharacterController controller;
    Transform target;
    DemonState demonState;
    GameObject firstAppearance1Point;
    GameObject firstAppearance2Point;
    GameObject startPoint;
    GameObject targetObject;
    public AudioClip[] audioClips;
    PlayerControllerScript playerScript;

    bool firstAppearance = true;
    bool firstAppearanceTriggered = false;

    float aggressionLevel = 0;
    bool chasingPlayer = false;

    void Start()
    {
        controller = GetComponent<CharacterController>();
        player = GameObject.FindGameObjectWithTag("Player");
        playerScript = player.GetComponent<PlayerControllerScript>();
        demonState = new DemonState(GetComponent<Animator>(),
GetComponents<AudioSource>()[0],
            GetComponents<AudioSource>()[1], audioClips, player, this.gameObject);

        demonState.changeState(demonStates.walking);
        demonState.playerCanTakeDamage = true;

        firstAppearance1Point = GameObject.Find("DemonPoints/FirstAppearance1");
        startPoint = GameObject.Find("DemonPoints/StartPoint");
        firstAppearance2Point = GameObject.Find("DemonPoints/FirstAppearance2");
        targetObject = GameObject.Find("DemonPoints/Target");
        if (firstAppearance) targetObject.transform.position =
```

```
                new Vector3(player.transform.position.x + (maxDistNearPoint -
aggressionLevel) * (Random.Range(0, 1) * 2 - 1), player.transform.position.y,
                player.transform.position.z + (maxDistNearPoint - aggressionLevel) *
(Random.Range(0, 1) * 2 - 1));
        target = targetObject.transform;

        if (!firstAppearance) this.transform.position =
firstAppearance1Point.transform.position;
        Invoke("randomSound", 5);
    }

    void Update()
    {
        if (!firstAppearance) firstAppearanceMode();
        else
        {
            if (!player.GetComponent<PlayerControllerScript>().dead)
aggresiveWanderingMode();
            else demonState.changeState(demonStates.idle);
        }
        transform.LookAt(target.position);
        controller.Move(transform.forward * Time.deltaTime * demonState.speed);
        var gravityForce = new Vector3(0, -1 * gravity, 0);
        controller.Move(gravityForce * Time.deltaTime);
    }



    public void firstAppearanceMode()
    {
        if (!firstAppearanceTriggered)
        {
            if (Vector3.Distance(this.transform.position, player.transform.position) <=
70)
            {
                demonState.changeState(demonStates.walking);
                target = firstAppearance2Point.transform;
                firstAppearanceTriggered = true;
            }
        }
        else
        {
            if (Vector3.Distance(this.transform.position,
firstAppearance2Point.transform.position) <= 2)
            {
                firstAppearance = true;
                this.transform.position = startPoint.transform.position;
                targetObject.transform.position =
                    new Vector3(player.transform.position.x + (maxDistNearPoint -
aggressionLevel) * (Random.Range(0, 1) * 2 - 1),
                    player.transform.position.y, player.transform.position.z +
(maxDistNearPoint - aggressionLevel) * (Random.Range(0, 1) * 2 - 1));
            }
        }
    }

    public void aggresiveWanderingMode()
    {
            if (Vector3.Distance(this.transform.position, player.transform.position) <=
1.5)
            {
                demonState.changeState(demonStates.attacking);
            }
            else
            {
```

```
                if ((!chasingPlayer && (Mathf.Abs(this.transform.position.x -
target.transform.position.x) +
                Mathf.Abs(this.transform.position.z - target.transform.position.z)) <=
12)
                    || (chasingPlayer && Vector3.Distance(this.transform.position,
player.transform.position) > 10) ||
                    (chasingPlayer && playerScript.nearWarm > 0))
            {
                if (!chasingPlayer) increaseAggression();
                targetObject.transform.position = new Vector3(player.transform.position.x
+
                    (maxDistNearPoint - aggressionLevel) * (Random.Range(0, 1) * 2 - 1),
player.transform.position.y, player.transform.position.z +
                    (maxDistNearPoint - aggressionLevel) * (Random.Range(0, 1) * 2 - 1));
                target = targetObject.transform;
                chasingPlayer = false;
            }
            else if (!chasingPlayer && Vector3.Distance(this.transform.position,
player.transform.position) <= 10 && playerScript.nearWarm <= 0)
            {
                target = player.transform;
                chasingPlayer = true;
            }


        }
        demonState.demonAttackUpdate();
    }

    public void randomSound()
    {
        demonState.randomSound();
        Invoke("randomSound", Random.Range(4f, 10f));
    }

    public void increaseAggression()
    {
        if (aggressionLevel < maxDistNearPoint - 5)
        {
            aggressionLevel += 5;
        }
        if (playerScript.nearWarm > 0) aggressionLevel = 5;
    }
}

public class DemonState: MonoBehaviour
{
    public bool playerCanTakeDamage = true;
    public float speed = 0;
    public demonStates currentState = demonStates.idle;
    public demonStates previousState = 0;
    private Animator anim;
    private AudioSource audio;
    private AudioSource walkingAudio;
    public AudioClip[] audioClips;
    private AudioClip[] currentClips;
    private GameObject player;
    private GameObject demon;


    public DemonState(Animator Anim, AudioSource Audio, AudioSource WalkingAudio,
AudioClip[] AudioClips, GameObject Player, GameObject Demon)
    {
        anim = Anim;
        audio = Audio;
```

```
        walkingAudio = WalkingAudio;
        audioClips = AudioClips;
        player = Player;
        demon = Demon;
        playerCanTakeDamage = true;
        changeState(demonStates.idle);
    }
    public void changeState(demonStates state)
    {
        if (currentState == demonStates.attacking)
        {
            if (anim.GetCurrentAnimatorStateInfo(0).normalizedTime > 1 &&
!anim.IsInTransition(0))
            {
                currentState = demonStates.walking;
                switch (state)
                {
                    case demonStates.idle:
                        speed = 0;
                        currentClips = new AudioClip[2] { audioClips[0], audioClips[1] };
                        break;
                    case demonStates.walking:
                        walkingAudio.Play();
                        currentClips = new AudioClip[2] { audioClips[0], audioClips[1] };
                        speed = 2.4f;
                        break;
                    case demonStates.attacking:
                        walkingAudio.Stop();
                        speed = 0f;
                        break;
                    default:
                        speed = 0;
                        break;
                }
                anim.SetInteger("state", (int)currentState);
                playerCanTakeDamage = true;
            }
        }
        else
        {
            if (!(currentState == demonStates.attacking &&
anim.GetCurrentAnimatorStateInfo(0).normalizedTime > 1 && !anim.IsInTransition(0)))
            {
                currentState = state;
                switch (state)
                {
                    case demonStates.idle:
                        speed = 0f;
                        currentClips = new AudioClip[2] { audioClips[0], audioClips[1] };
                        break;
                    case demonStates.walking:
                        walkingAudio.Play();
                        currentClips = new AudioClip[2] { audioClips[0], audioClips[1] };
                        speed = 2.6f;
                        break;
                    case demonStates.attacking:
                        walkingAudio.Stop();
                        audio.clip = currentClips[1];
                        audio.Play();
                        speed = 0f;
                        break;
                    default:
                        speed = 0f;
                        break;
                }
```

```
                anim.SetInteger("state", (int)currentState);
            }
        }
    }

    public void demonAttackUpdate() {
        if (currentState == demonStates.attacking)
        {
            if (playerCanTakeDamage == true && Vector3.Distance(demon.transform.position,
player.transform.position) <= 1.5 &&
                anim.GetCurrentAnimatorStateInfo(0).normalizedTime <= 0.3f &&
anim.GetCurrentAnimatorStateInfo(0).normalizedTime >= 0.05f)
            {
                player.GetComponent<PlayerControllerScript>().takeDamage();
                playerCanTakeDamage = false;
            }
            if (anim.GetCurrentAnimatorStateInfo(0).normalizedTime > 1 &&
!anim.IsInTransition(0))
            {
                if (Vector3.Distance(demon.transform.position, player.transform.position)
> 1.5) changeState(demonStates.walking);
                playerCanTakeDamage = true;

            }
        }
    }

    public void randomSound()
    {
        audio.clip = currentClips[0];
        audio.Play();
    }
}
```

firePlaceController.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class firePlaceController : MonoBehaviour
{
    GameObject player;
    bool playerNearWarmNotified = false;
    PlayerControllerScript playerScript;
    public AudioClip fire;
    AudioSource audio;
    Light fireLight;
    float addition;

    void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player");
        audio = GetComponent<AudioSource>();
        audio.clip = fire;
        audio.loop = true;
        audio.Play();
        playerScript = player.GetComponent<PlayerControllerScript>();
        fireLight = GetComponent<Light>();
    }

    // Update is called once per frame
    void Update()
    {
        addition = Random.Range((float)-0.55, (float)0.55);
```

```
        if (fireLight.intensity + addition > 4 && fireLight.intensity + addition < 8)
fireLight.intensity += addition;
        if (Vector3.Distance(this.transform.position, player.transform.position) <= 3f)
        {
            if (!playerNearWarmNotified)
            {
                playerScript.nearWarm++;
                playerNearWarmNotified = true;
            }
        }
        else
        {
            if (playerNearWarmNotified)
            {
                playerScript.nearWarm--;
                playerNearWarmNotified = false;
            }
        }
    }
}
```

Flickering.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class flickering : MonoBehaviour
{
    public Light light;
    Random random;
    float addition;
    // Start is called before the first frame update
    void Start()
    {
        light = GetComponent<Light>();
        random = new Random();
    }

    // Update is called once per frame
    void Update()
    {
        addition = Random.Range((float)-0.55, (float)0.55);
        if (light.intensity + addition > 4 && light.intensity + addition < 12)
light.intensity += addition;
    }
}
```

highscoresLoad.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Text;
using UnityEngine;

public class highscoresLoad : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
        float[] scores = new float[10];
        string text = "";
        const Int32 BufferSize = 128;
        using (var fileStream = File.OpenRead("highscores.txt"))
        using (var streamReader = new StreamReader(fileStream, Encoding.UTF8, true,
BufferSize))
        {
            string line;
            int i = 1;
            while ((line = streamReader.ReadLine()) != null)
            {
                float score = float.Parse(line);
                text += i.ToString() + ") " + ((int)(score / 60 / 60)).ToString() + ":" +
((int)(score / 60 % 60)).ToString() + ":" + ((int)(score % 60)).ToString() + "\n";
                i++;
            }
        }
        GetComponent<UnityEngine.UI.Text>().text = text;
    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

LitScript.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LitScript : MonoBehaviour
{
    public Animator anim;
    public Light light;
    public bool lit;
    GameObject player;
    PlayerControllerScript playerScript;
    bool playerNotified = false;
    // Start is called before the first frame update
    void Start()
    {
        anim = GetComponent<Animator>();
        light = GetComponent<Light>();
        lit = false;
        player = GameObject.FindGameObjectWithTag("Player");
        playerScript = player.GetComponent<PlayerControllerScript>();
    }

    // Update is called once per frame
    void Update()
```

```
    {
        if (Vector3.Distance(this.transform.position, player.transform.position) <= 1.5f)
        {
            if (!playerNotified)
            {
                playerScript.interactablesNear++;
                playerNotified = true;
            }
        }
        else
        {
            if (playerNotified)
            {
                playerScript.interactablesNear--;
                playerNotified = false;
            }
        }
        if (Input.GetKeyDown("e"))
        {
            if (Vector3.Distance(light.transform.position, player.transform.position) <=
1.5f)
            {
                if (!lit)
                {
                    anim.Play("CandleAnimation");
                    light.intensity = 2;
                    lit = true;
                }
                else
                {
                    anim.Play("CandleUnlitAnimation");
                    light.intensity = 0;
                    lit = false;
                }

            }
        }
    }
}
```

MainMenu.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour
{
    public void StartGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }

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

}
```

matchBoxController.cs

```
using System.Collections;
using System.Collections.Generic;
```

```
using UnityEngine;

public class matchBoxController : MonoBehaviour
{
    GameObject player;
    bool playerNotified = false;
    PlayerControllerScript playerScript;
    // Start is called before the first frame update
    void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player");
        playerScript = player.GetComponent<PlayerControllerScript>();
    }

    // Update is called once per frame
    void Update()
    {
        if (Vector3.Distance(this.transform.position, player.transform.position) <= 1.2f)
        {
            if (!playerNotified)
            {
                playerScript.interactablesNear++;
                playerNotified = true;
            }
        }
        else
        {
            if (playerNotified)
            {
                playerScript.interactablesNear--;
                playerNotified = false;
            }
        }
        if (Input.GetKeyDown("e"))
        {
            if (Vector3.Distance(this.transform.position, player.transform.position) <=
1.2f)
            {

                if (playerScript.matches < 99)
                {
                    int matchesCount = Random.Range(2, 7);
                    playerScript.updateMatches(matchesCount);
                    if (playerNotified)
                    {
                        playerScript.interactablesNear--;
                        playerNotified = false;
                    }
                    Destroy(gameObject);
                }
            }

        }
    }
}
```

OptionsMenu.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Audio;

public class OptionsMenu : MonoBehaviour
{
```

```
    public AudioMixer audioMixer;

    public void SetEffectsVolume(float volume)
    {
        audioMixer.SetFloat("Effects", volume);
    }

    public void SetMusicVolume(float volume)
    {
        audioMixer.SetFloat("Music", volume);
        audioMixer.SetFloat("Bass", volume);
    }
}
```

PlayerControllerScript.cs

```
using UnityEngine.UI;
using UnityEngine;
using System.IO;
using System;
using System.Text;

public class PlayerControllerScript : MonoBehaviour
{
    public SimpleHealthBar healthBar;
    public SimpleHealthBar coldBar;
    public Text matchesText;
    public Text branchesText;
    public Image EImage;
    public AudioClip walking;
    public AudioClip running;
    public AudioClip pain;
    private AudioClip current;
    private AudioSource walkingSource;
    private AudioSource painSource;
    AudioSource[] array;
    float footSeconds = 0f;
    bool notMoving = true;
    bool resetInvoke = false;
    bool invokePlaying = false;
    int health = 100;
    int cold = 0;
    int maxCold = 10000;
    public int nearWarm = 0;
    public bool dead = false;
    Vector3 preDeathVector;
    GameObject afterGameScreen;
    float time = 0.0f;
    public int matches = 5;
    public int branches = 1;
    int maxBranchDistance = 25;
    int minBranchDistance = 10;
    public int interactablesNear = 0;

    public GameObject[] branchInstances;
    public GameObject bonfireInstance;

    // Start is called before the first frame update
    void Start()
    {
        afterGameScreen = GameObject.Find("AfterGameUI");
        afterGameScreen.SetActive(false);
        array = this.GetComponents<AudioSource>();
        walkingSource = array[0];
        painSource = array[1];
```

```
        coldEffect();
        coldEffectDamage();
        matchesText.text = matches.ToString();
        branchesText.text = branches.ToString();
        healthBar.UpdateBar(health, 100);
        coldBar.UpdateBar(cold, maxCold);
        createBranch();
        regenerationEffect();
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown("q"))
        {
            if (!dead && branches >= 5)
            {
                updateBranches(-5);
                Vector3 bonfirePosition = new Vector3(transform.position.x + 1,
this.transform.position.y, this.transform.position.z + 1);
                GameObject bonfire = Instantiate(bonfireInstance, bonfirePosition, new
Quaternion());
            }
        }
            if (interactablesNear > 0) EImage.enabled = true;
        else EImage.enabled = false;
        if (dead)
        {
            this.transform.position = preDeathVector;
            GameObject.Find("TimeText").GetComponent<UnityEngine.UI.Text>().text =
((int)(time / 60 / 60)).ToString() + ":" + ((int)(time / 60 % 60)).ToString() + ":" +
((int)(time % 60)).ToString();
        }
        else time += Time.deltaTime;
        if (Input.GetKeyDown(KeyCode.W) || Input.GetKeyDown(KeyCode.W) ||
Input.GetKeyDown(KeyCode.W) || Input.GetKeyDown(KeyCode.W))
        {
            if (invokePlaying) resetInvoke = true;
            notMoving = false;
            current = walking;
            walkingSource.clip = current;
            walkingSource.Play();
        }
        else if ((Input.GetKey(KeyCode.W) || Input.GetKey(KeyCode.W) ||
Input.GetKey(KeyCode.W) || Input.GetKey(KeyCode.W)) &&
Input.GetKeyDown(KeyCode.LeftShift))
        {
            if (invokePlaying) resetInvoke = true;
            notMoving = false;
            current = running;
            walkingSource.clip = current;
            walkingSource.loop = true;
            walkingSource.Play();
        }
        else if ((Input.GetKey(KeyCode.W) || Input.GetKey(KeyCode.W) ||
Input.GetKey(KeyCode.W) || Input.GetKey(KeyCode.W)) && Input.GetKeyUp(KeyCode.LeftShift))
        {
            if (invokePlaying) resetInvoke = true;
            notMoving = false;
            current = walking;
            walkingSource.clip = current;
            walkingSource.loop = true;
            walkingSource.Play();
        }
```

```
        else if (!Input.GetKey(KeyCode.W) && !Input.GetKey(KeyCode.A) &&
!Input.GetKey(KeyCode.S) && !Input.GetKey(KeyCode.D) && !Input.GetKey(KeyCode.LeftShift))
        {
            walkingSource.Stop();
            notMoving = true;
        }
    }

    public void createBranch()
    {
        Vector3 branchPosition = new Vector3(transform.position.x +
UnityEngine.Random.Range(minBranchDistance, maxBranchDistance) *
(UnityEngine.Random.Range(0, 1) * 2 - 1),
            this.transform.position.y + 2,
            this.transform.position.z + UnityEngine.Random.Range(minBranchDistance,
maxBranchDistance) * (UnityEngine.Random.Range(0, 1) * 2 - 1));

        GameObject branchInstance =
Instantiate(branchInstances[UnityEngine.Random.Range(0, 4)], branchPosition, new
Quaternion());
        Invoke("createBranch", UnityEngine.Random.Range(3f, 10f));

    }

    public void updateMatches(int difference)
    {
        if (matches + difference > 99) matches = 99;
        else matches += difference;
        matchesText.text = matches.ToString();
    }

    public void updateBranches(int difference)
    {
        branches += difference;
        branchesText.text = branches.ToString();
    }

    public void coldEffect()
    {
        if (nearWarm <= 0)
        {
            if (cold < maxCold) cold += 3;
        }
        else
        {
            if (cold > 0) cold -= 5;
        }
        coldBar.UpdateBar(cold, maxCold);
        Invoke("coldEffect", 0.1f);
    }

    public void coldEffectDamage()
    {
        int damage = 0;
        if (nearWarm <= 0)
        {
            if (cold >= 6000 && cold <= 7000)
            {
                damage = 1;
            }
            else if (cold > 7000 && cold <= 8500)
            {
                damage = 2;
            }
            else if (cold > 8500)
```

```
            {
                damage = 3;
            }
            if (health - damage > 0) health -= damage;
            else
            {
                Die();
                health = 0;
            }
        healthBar.UpdateBar(health, 100);
        }
        Invoke("coldEffectDamage", 5f);
    }
    public void regenerationEffect()
    {
        if (nearWarm > 0 && cold < 1000)
        {
            health += 1;
            healthBar.UpdateBar(health, 100);
        }
        Invoke("regenerationEffect", 2f);
    }

    public void takeDamage()
    {
        painSource.Play();
        if (health - 10 > 0) health -= 10;
        else
        {
            Die();
            health = 0;
        }
        healthBar.UpdateBar(health, 100);
    }

    public void prepareForDeath()
    {
        Rigidbody[] bodies = GetComponentsInChildren<Rigidbody>();
        foreach (Rigidbody rb in bodies)
        {
            if (rb.GetComponent<CapsuleCollider>() != null)
rb.GetComponent<CapsuleCollider>().enabled = true;
            else if (rb.GetComponent<BoxCollider>() != null)
rb.GetComponent<BoxCollider>().enabled = true;

        }
    }

    public void Die()
    {
        prepareForDeath();
        dead = true;
        GetComponent<Animator>().enabled = false;
        preDeathVector = new Vector3(this.transform.position.x,
this.transform.position.y, this.transform.position.z);
        afterGameScreen.SetActive(true);
        Cursor.visible = true;
        Cursor.lockState = CursorLockMode.None;
        if (!File.Exists("highscores.txt"))
        {
            StreamWriter sr = File.CreateText("highscores.txt");
            sr.Write(time.ToString());
            sr.Close();

        }
```

```
        else
        {
            float[] scores = new float[10];
            int maxI = 0;
            const Int32 BufferSize = 128;
            using (var fileStream = File.OpenRead("highscores.txt"))
            using (var streamReader = new StreamReader(fileStream, Encoding.UTF8, true,
BufferSize))
            {
                string line;
                int i = 0;
                bool added = false;
                while ((line = streamReader.ReadLine()) != null)
                {
                    float score = float.Parse(line);
                    if (score < time && !added) {
                        scores[i] = time;
                        if (i < 10)
                        {
                            i++;
                            maxI++;
                            scores[i] = score;
                            if (i < 10)
                            {
                                maxI++;
                                i++;
                            }
                        }

                        added = true;
                    }
                    else
                    {
                        if (i < 10)
                        {
                            scores[i] = score;
                            maxI++;
                            i++;
                        }
                    }

                }
            }
            using (var fileStream = File.OpenWrite("highscores.txt"))
            using (var streamWriter = new StreamWriter(fileStream))
            {
                for (int i = 0; i < maxI; i++)
                {
                    streamWriter.WriteLine(scores[i].ToString());
                }
            }

        }

    }
}
```

RockFallTrigger.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class RockFallTrigger : MonoBehaviour
{
```

```
    Rigidbody rock1Rigid;
    Rigidbody rock2Rigid;
    Rigidbody rock3Rigid;
    bool triggered;
    int applyForceCount = 0;
    GameObject player;
    public AudioClip rockFall;
    bool final = false;
    AudioSource audio;
    // Start is called before the first frame update
    void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player");
        rock1Rigid = GameObject.Find("rockForFalling1").GetComponent<Rigidbody>();
        rock2Rigid = GameObject.Find("rockForFalling2").GetComponent<Rigidbody>();
        rock3Rigid = GameObject.Find("rockForFalling3").GetComponent<Rigidbody>();
        audio = GetComponent<AudioSource>();
        audio.clip = rockFall;
        triggered = false;
    }

    // Update is called once per frame
    void Update()
    {
        if (!triggered)
        {
            if (Vector3.Distance(transform.position, player.transform.position) <= 10)
            {
                triggered = true;
                rock1Rigid.isKinematic = false;
                rock2Rigid.isKinematic = false;
                rock3Rigid.isKinematic = false;
            }
        }
        if (applyForceCount < 300 && triggered)
        {
            if (!final && applyForceCount > 200)
            {
                audio.Play();
                final = true;
            }
            Vector3 moveVector = new Vector3(-1, 0, -1);
            rock1Rigid.AddForce(moveVector, ForceMode.Acceleration);
            rock2Rigid.AddForce(moveVector, ForceMode.Acceleration);
            rock3Rigid.AddForce(moveVector, ForceMode.Acceleration);
            applyForceCount++;
        }

    }
}
```

Swtich.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Swtich : MonoBehaviour
{
    public Animator anim;
    public bool on;
    GameObject player;
    // Start is called before the first frame update
    void Start()
    {
```

```
        anim = GetComponent<Animator>();
        on = false;
        player = GameObject.FindGameObjectWithTag("Player");
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown("e"))
        {
            if (Vector3.Distance(transform.position, player.transform.position) <= 1.5)
            {
                if (!on)
                {
                    anim.Play("SwitchOnAnimation");
                    on = true;
                }
                else
                {
                    anim.Play("SwitchOffAnimation");
                    on = false;
                }

            }
        }
    }
}
```

tentFireplaceScript.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class tentFireplaceScript : MonoBehaviour
{
    GameObject player;
    bool playerNearWarmNotified = false;
    PlayerControllerScript playerScript;
    public AudioClip fire;
    AudioSource audio;
    Light fireLight;
    float addition;

    void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player");
        audio = GetComponent<AudioSource>();
        audio.clip = fire;
        audio.loop = true;
        audio.Play();
        playerScript = player.GetComponent<PlayerControllerScript>();
        fireLight = GetComponent<Light>();
    }

    // Update is called once per frame
    void Update()
    {
        addition = Random.Range((float)-0.55, (float)0.55);
        if (fireLight.intensity + addition > 4 && fireLight.intensity + addition < 8)
fireLight.intensity += addition;
        if (Vector3.Distance(this.transform.position, player.transform.position) <= 3f)
        {
            if (!playerNearWarmNotified)
            {
```

```
                playerScript.nearWarm++;
                playerNearWarmNotified = true;
            }
        }
        else
        {

            if (playerNearWarmNotified)
            {
                playerScript.nearWarm--;
                playerNearWarmNotified = false;
            }
        }
    }
}
```

WindowAnimationController.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class WindowAnimationController : MonoBehaviour
{
    public Animator animLeft;
    public Animator animRight;
    public bool opened, animationEnded;
    bool playerNotified = false;
    PlayerControllerScript playerScript;
    GameObject player;
    // Start is called before the first frame update
    void Start()
    {
        animLeft = transform.Find("openinsid0").GetComponent<Animator>();
        animRight = transform.Find("openinside").GetComponent<Animator>();
        opened = true;
        animationEnded = true;
        player = GameObject.FindGameObjectWithTag("Player");
        playerScript = player.GetComponent<PlayerControllerScript>();
    }

    // Update is called once per frame
    void Update()
    {
        if (Vector3.Distance(this.transform.position, player.transform.position) <= 1.2f)
        {
            if (!playerNotified)
            {
                playerScript.interactablesNear++;
                playerNotified = true;
            }
        }
        else
        {
            if (playerNotified)
            {
                playerScript.interactablesNear--;
                playerNotified = false;
            }
        }
        if (Input.GetKeyDown("e") && animationEnded)
        {
            if (Vector3.Distance(transform.position, player.transform.position) <= 1.2f)
            {
                animationEnded = false;
                if (!opened)
```

```
                {
                    animLeft.Play("WindowLeftOpenAnimation");
                    animRight.Play("WindowRightOpenAnimation");
                    StartCoroutine(Wait());
                }
                else
                {
                    animLeft.Play("WindowLeftCloseAnimation");
                    animRight.Play("WindowRightCloseAnimation");
                    StartCoroutine(Wait());
                }
            }
        }
    }

    IEnumerator Wait()
    {
        yield return new WaitForSeconds(1);
        opened = !opened;
        animationEnded = true;
    }
}
```