



KAUNAS UNIVERSITY OF TECHNOLOGY

FACULTY OF INFORMATICS

T120B166 Development of Computer Games and Interactive Applications

Banisher

Group, Name and Surname:
IFF 7-11, Mantas Klimašauskas
Date: 2020-03-02

Kaunas, 2020

Table of Contents

<i>Table of Images</i>	4
<i>Table of functions</i>	5
<i>Work Distribution Table:</i>	6
<i>Description of Your Game</i>	7
<i>Laboratory work #1</i>	8
List of tasks.....	8
<i>Solution</i>	8
1. Second level design.....	8
2. Basic movement player physics.....	8
3. Second level design.....	9
4. Active ragdolls, which try to stay upright (standing).....	10
<i>Laboratory 1 defense task</i>	11
3. Added four area lights (colors: red, green, blue, yellow).....	11
4. In Window->Rendering->Lighting settings enabled “Auto generate”, to enable automatically baking lightmaps	11
<i>Laboratory work #2</i>	12
List of tasks.....	12
<i>Solution</i>	12
Task #1. Experiment with ProBuilder tools, start expanding, building your World environment.....	12
Task #3. Create and/or animate 5 objects of your choice in your World	13
Task #4. Create at least 5 particle effects for your environment	14
Task #5. Add a custom skybox	15
Task #6. Create at least 3 different Physics Materials for various parts of the map	16
Task #7. Create 4 types of objects that use OnCollisionEnter / OnCollisionEnter2D / OnTriggerEnter / OnTriggerEnter2D.....	17
Task #8. Assign optimal colliders for the environment objects that are not moving and set their flags to static (test performance before and after)	18
Task #9. Bake a lightmap and measure performance	18
Task #10. Optimize all textures depending on their parameters and measure graphical memory load	20
Task #11. Try hard vs soft shadows, different quality settings and measure the performance	20
Defense task	22
<i>Laboratory work #3</i>	24
List of tasks.....	24
<i>Solution</i>	24
Task #1. Title of Task	24
Task #2. Title of Task	24
Task #3. Title of Task	25

<i>User's manual</i>	27
<i>Literature list</i>	28
<i>ANNEX</i>	29

Table of Images

Figure 1. Second level layout	8
Figure 2. Player ground position before moving above the white box and after	9
Figure 3. Inside the warehouse	9
Figure 4. Looking into the warehouse through the alleyway	10
Figure 5. Active ragdoll, trying to stand upright	11
Figure 6. Lab 1 defense result	11
Figure 7. Ac unit with custom fans made with ProBuilder	12
Figure 8. AC Unit propeller animation.....	13
Figure 9. Flickering light animation	13
Figure 10. Mist particle effect using a custom texture	14
Figure 11. Custom mist texture	14
Figure 12. Sparks particle effect.....	14
Figure 13. Spark custom texture.....	15
Figure 14. Ambiguous particle effect for AC Unit propeller	15
Figure 15. Custom skybox.....	15
Figure 16. Soccer ball with a bouncy physic material.....	16
Figure 17. Slippery ramp physic material	17
Figure 18. Active ragdoll, following an animation	17
Figure 19. Before marking objects as static	18
Figure 20. After marking objects as static	18
Figure 21. Light performance before lightmapping	19
Figure 22. Light performance after lightmapping	20
Figure 23. Scene without shadows	21
Figure 24. Scene with soft shadows	22
Figure 25. Defense task: infinitely bouncing ball	23
Figure 26. Screenshot #1	24
Figure 27. Screenshot #2	25
Figure 28. Screenshot #3	25
Figure 29. Screenshot #5	27
Figure 30. Screenshot #5	27

Table of functions

Table 1. Title of fragment #1	Error! Bookmark not defined.
Table 2. Title of fragment #2.....	Error! Bookmark not defined.
Table 3. Title of fragment #3.....	Error! Bookmark not defined.
Table 4. Title of fragment #1	24
Table 5. Title of fragment #2.....	25
Table 6. Title of fragment #3.....	26

Work Distribution Table:

<i>Name/Surname</i>	<i>Description of game development part</i>
<i>Mantas Klimašauskas</i>	<i>Programming</i>
<i>Deividas Verbickas</i>	<i>Programming, level design</i>

Description of Your Game

Description of Your Game.

1. 3D or 2D? *3D*
2. What type is your game? *Action*.
3. What genre is your game? *Fighting*.
4. Platform? *PC*.
5. Scenario Description. *A fighting game where the player has the ability to enter a separate mirror dimension.*

The dimension is called the Exsilium and the one who can enter it are called outcasts (exsul). In the Exsilium you can see and touch (and even affect) the real world. Time moves slower in the Exsilium relative to the real world, which gives an advantage to anyone who is in it. In order to stay in the Exsilium the outcast must use their Vis (energy). The more matter the outcast takes to the mirror dimension the more Vis is being used. Also, the outcast uses more energy the further away they are from their physical body (in the real world). Vis is also directly related to stamina of the outcast in the real world.

Inside the Exsilium outcasts can use their energy to acquire buffs, such as super strength. Most of the enemies encountered by the player do not possess the power to enter use the Exsilium to their advantage. But there will be bosses, who will be just as capable (or even stronger) than the player. Anyone who is damaged while in the mirror dimension cannot be killed and simply loses their energy. In order to regain the energy, the outcast must use special items.

Laboratory work #1

List of tasks

1. Second level basic layout
2. Basic movement player physics
3. Add map details and lights
4. Active ragdolls, which try to stay upright (standing)

Solution

1. Second level design

Created a small alleyway, that leads to a warehouse. The alleyway is surrounded by a few buildings. The warehouse has three holes in the walls as entrances.

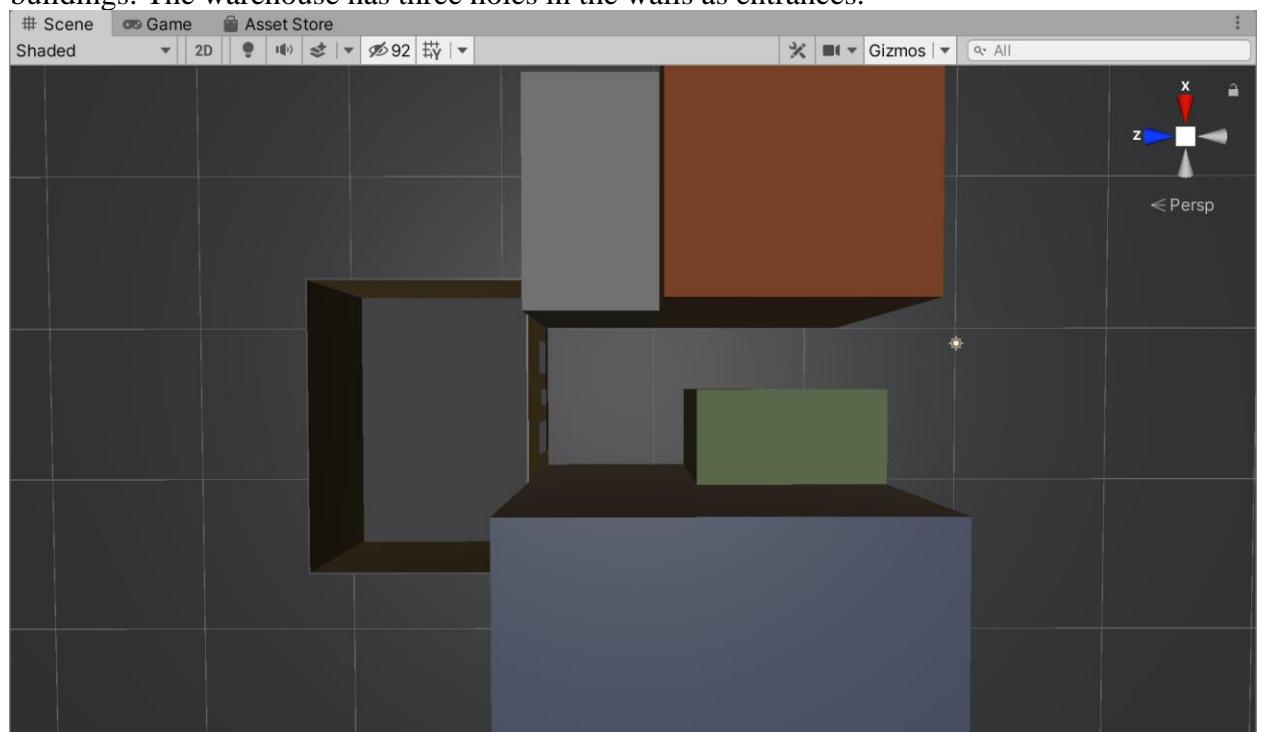


Figure 1. Second level layout

2. Basic movement player physics

Added a script for the SteamVR Player prefab, to enable basic physical ground detection for the player.

When the player moves over a collider, the script will push (teleport) the player up, to adjust the real-world ground to the object the player character in the game is standing on.

When the player steps off an object, a rigidbody attached to the Player prefab is enabled (is kinematic is turned off) and the player falls, until the ground is close enough.

To detect the ground below the player, a raycast with a limited distance is casted down. If the raycast hits something, that something is considered the ground, and the player's position is adjusted accordingly. If the raycast doesn't hit anything, the attached rigidbody is activated for the player to fall.

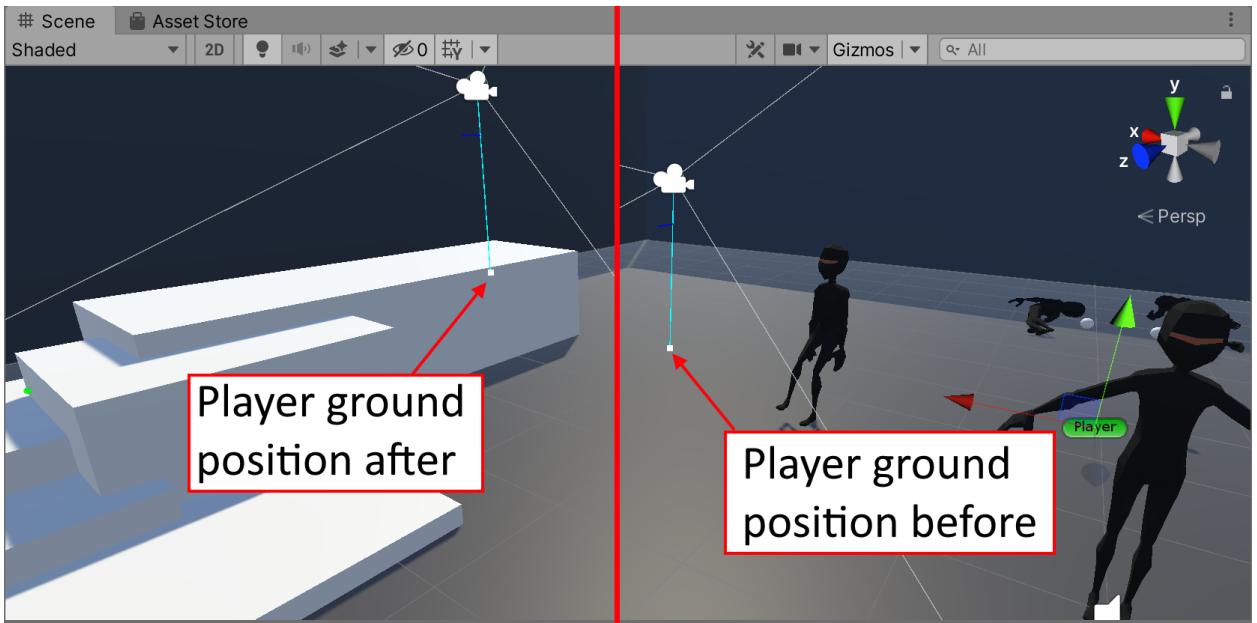


Figure 2. Player ground position before moving above the white box and after

3. Second level design

Added doors and ramps to some of the buildings. Also, added an indoor second-floor office, with windows. The windows are made with a transparent material. Also added a few pillars inside the warehouse.

For lighting, added five spot lights inside the warehouse, change the direction of the directional light (the sun) to make it an evening setting.

Also made a few prefabs, such as doors, ramps, lights.

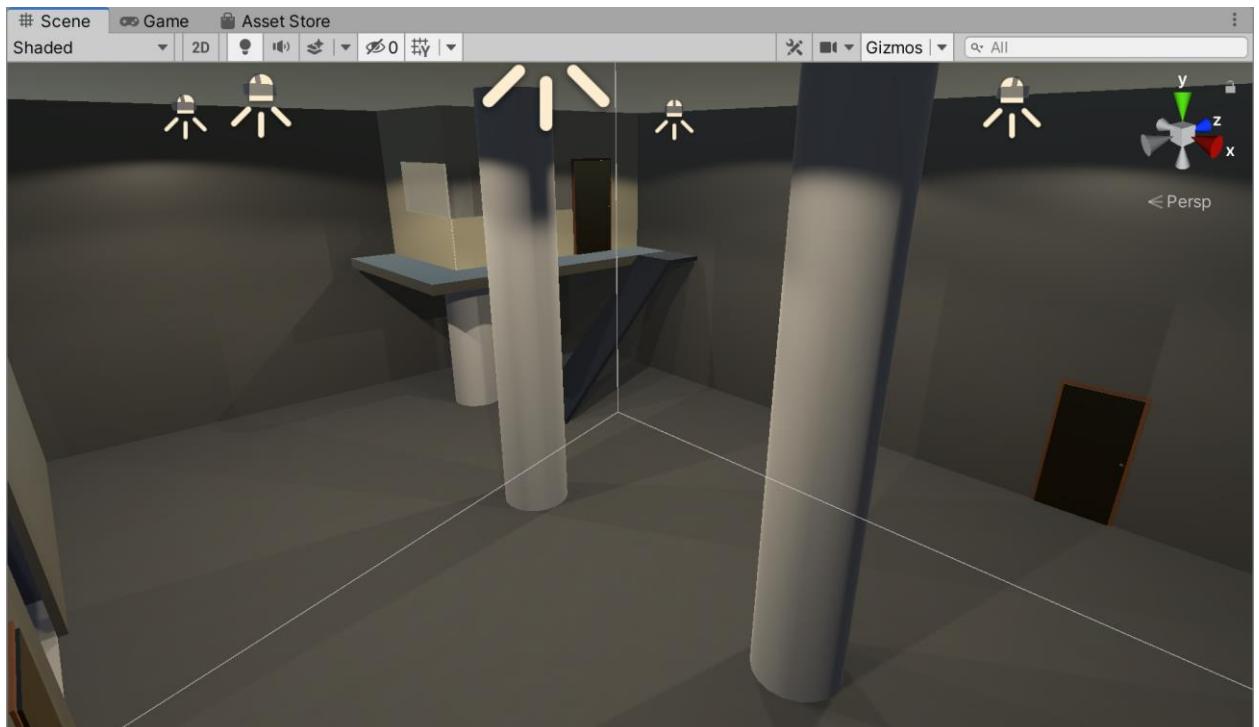


Figure 3. Inside the warehouse



Figure 4. Looking into the warehouse through the alleyway

4. Active ragdolls, which try to stay upright (standing)

Started the work on active ragdolls (ragdolls, which try to physically move to certain positions), for using them on enemies and the player body.

To create the ragdoll itself, a free asset* from the Unity Store for a rigged humanoid character model. Then, using the Unity's built-in ragdoll creation tools (GameObject->3D Object->Ragdoll...) setup the character joints and rigidbodies of the character.

The actual active part of the ragdoll is composed of multiple custom scripts.

The first script is used to push and rotate rigidbodies to a target position and rotation.

The second script is used to calculate the wanted position and rotation of a few parts of the ragdoll. Then using the first script, the ragdolls certain parts are pushed and rotated towards the calculated target spots. To keep the ragdoll standing, a raycast is casted, to calculate where the ground is, relative to the ragdoll.

* Lopoly Ninja by Tetra Arts

(<https://assetstore.unity.com/packages/3d/characters/humanoids/lowpoly-ninja-157942>)

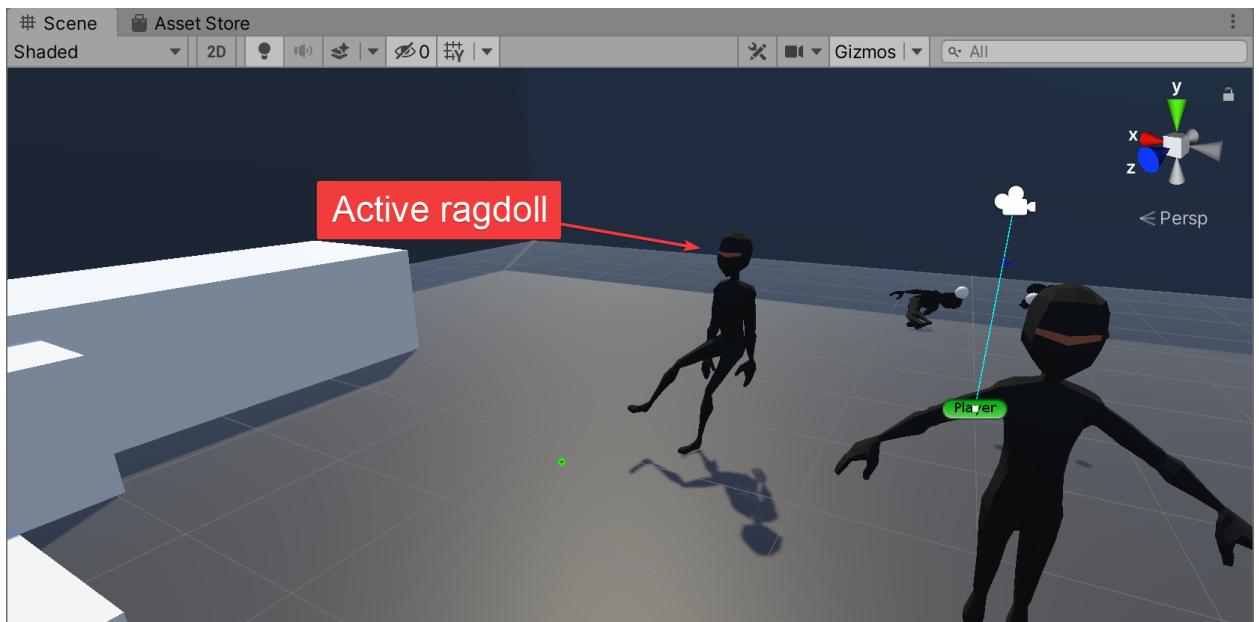


Figure 5. Active ragdoll, trying to stand upright

Laboratory 1 defense task

Create a Microsoft logo using a plane for logo surface and area lights for colors.

Steps:

1. Created a 3D plane
2. Marked the plane as static, to enable baked lighting for it
3. Added four area lights (colors: red, green, blue, yellow)
4. In Window->Rendering->Lighting settings enabled “Auto generate”, to enable automatically baking lightmaps

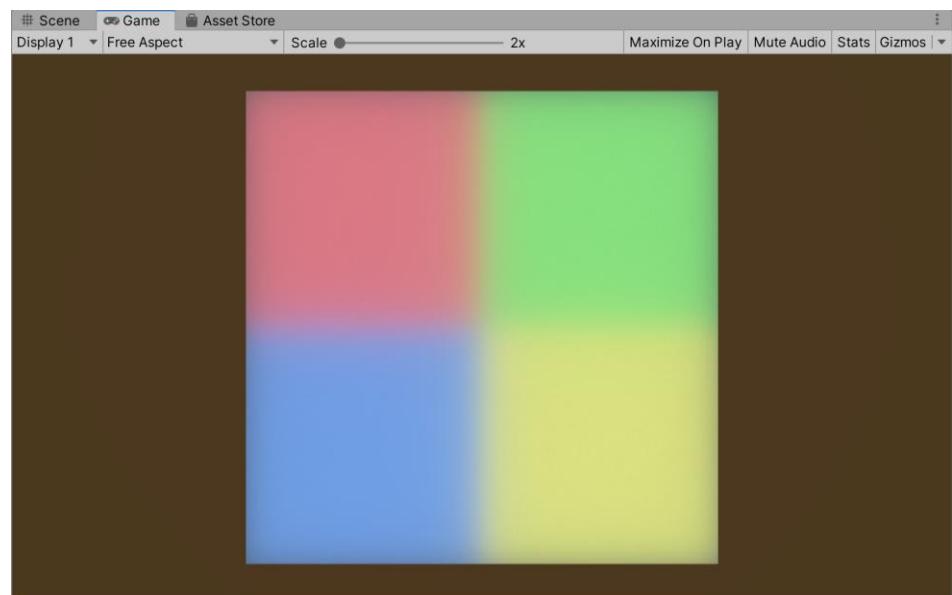


Figure 6. Lab 1 defense result

Laboratory work #2

List of tasks

1. Experiment with ProBuilder tools, start expanding, building your World environment;
2. Add animations to your game character (Assuming you have a working PlayerController. If you don't, create your own or get them from.
3. Create and/or animate 5 objects of your choice in your World (you can use Animator Component and/or Timeline);
4. Create at least 5 particle effects for your environment (dust, explosion, smoke gas, light sparks, etc.);
5. Add a custom skybox;
6. Create at least 3 different Physics Materials for various parts of the map (either it's a slippery platform/ice, bouncy wall, non-slippery ground, etc.);
7. Create 4 types of objects that use OnCollisionEnter / OnCollisionEnter2D / OnTriggerEnter / OnTriggerEnter2D.
8. Assign optimal colliders for the environment objects that are not moving and set their flags to static (test performance before and after);
9. Bake a lightmap and measure performance;
10. Optimize all textures depending on their parameters and measure graphical memory load;
11. Try hard vs soft shadows, different quality settings and measure the performance;

Solution

Task #1. Experiment with ProBuilder tools, start expanding, building your World environment

Created some custom fans with ProBuilder for an AC unit simple model.

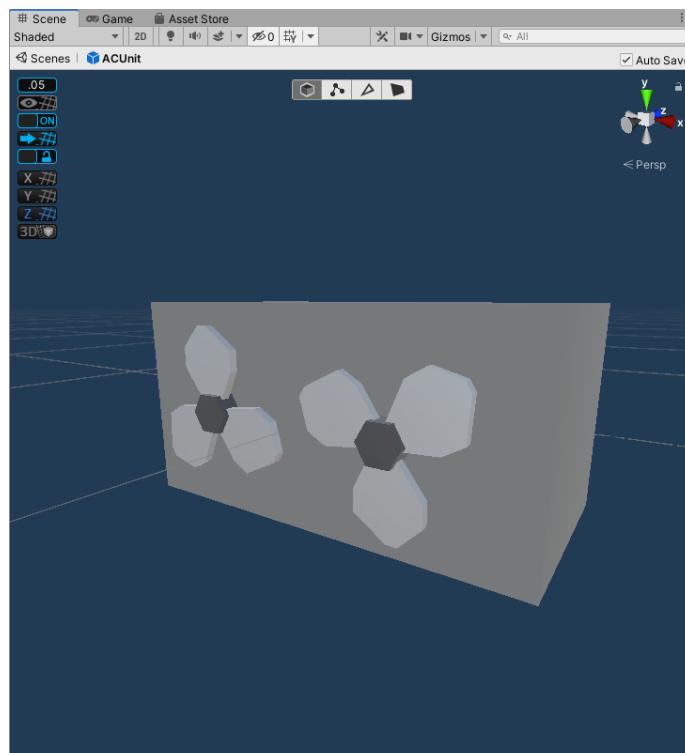


Figure 7. Ac unit with custom fans made with ProBuilder

Task #3. Create and/or animate 5 objects of your choice in your World

Animated AC Unit propellers to spin: one spins faster than the other.

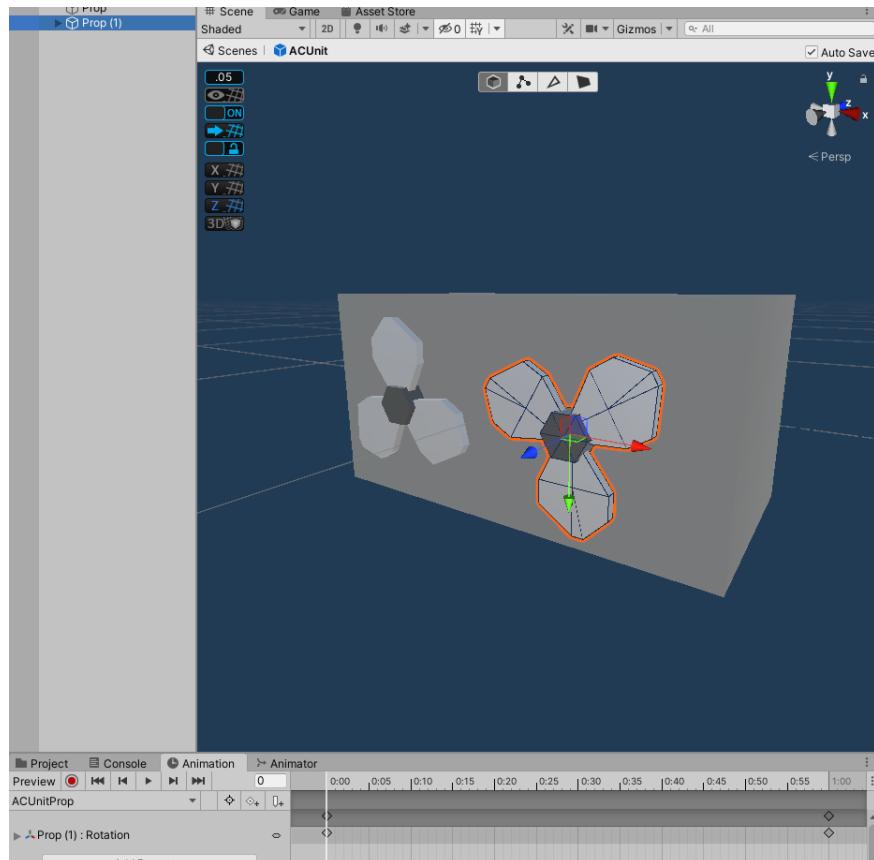


Figure 8. AC Unit propeller animation

Also animated a light inside the warehouse to flicker.

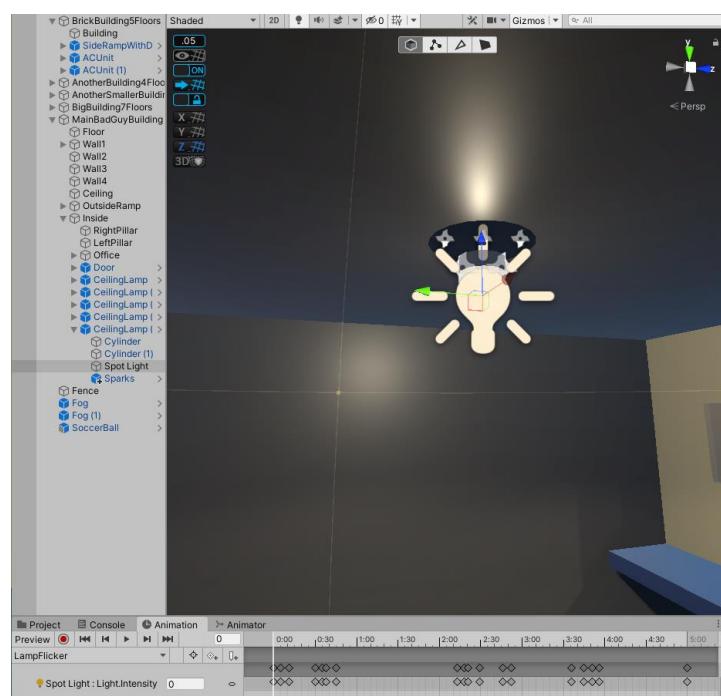


Figure 9. Flickering light animation

Task #4. Create at least 5 particle effects for your environment

Added a mist particle effect using a custom-made texture.

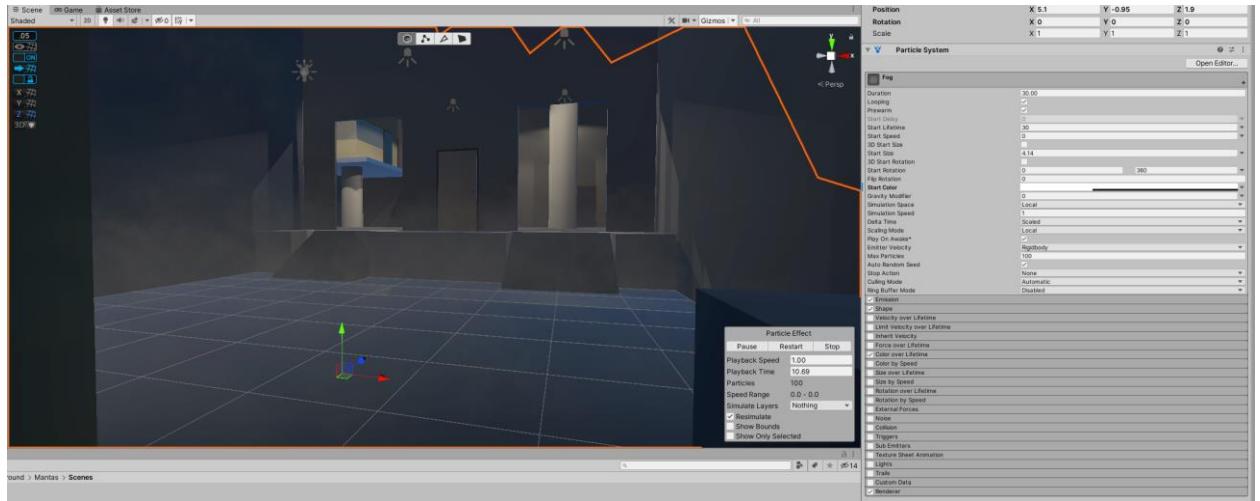


Figure 10. Mist particle effect using a custom texture

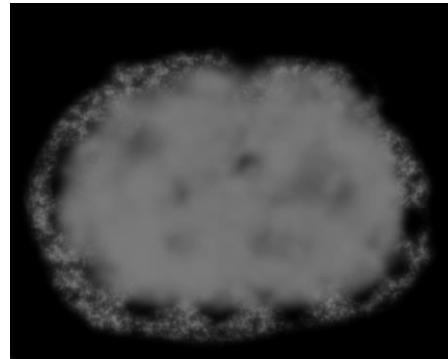


Figure 11. Custom mist texture

Also added a sparks particle effect to the flickering light in the warehouse with a custom spark texture.

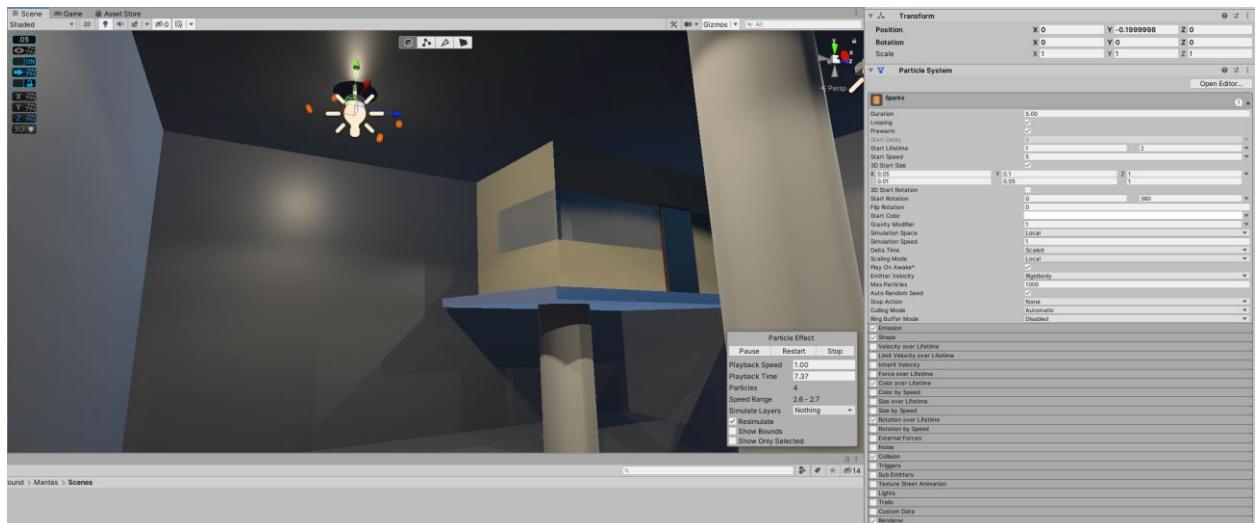


Figure 12. Sparks particle effect



Figure 13. Spark custom texture

Added some ambiguous particle effect for the AC Unit, falling from one of the propellers.

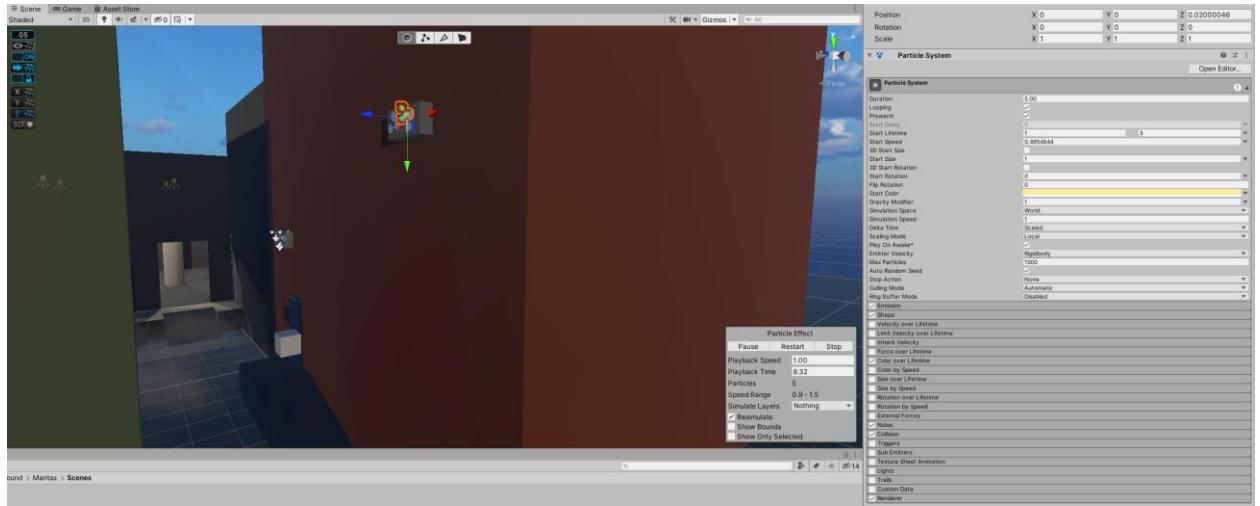


Figure 14. Ambiguous particle effect for AC Unit propeller

Task #5. Add a custom skybox

Imported a free asset (AllSky Free) from the asset store for a custom skybox. Then created a custom material, applied it to the skybox settings in the lighting window and rotated the skybox according to the directional light of the scene.

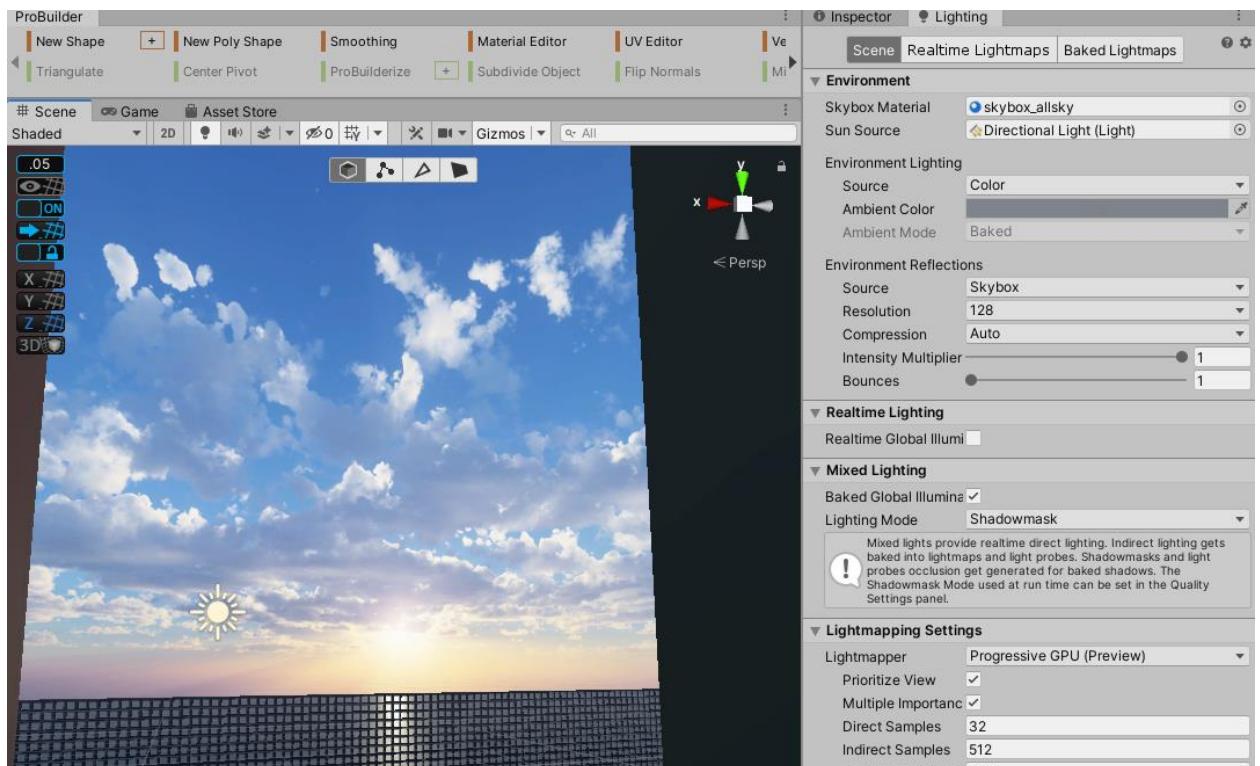


Figure 15. Custom skybox

Task #6. Create at least 3 different Physics Materials for various parts of the map

Added a soccer ball (free model downloaded from Asset Store, Soccer Ball) and created a bouncy physic material.

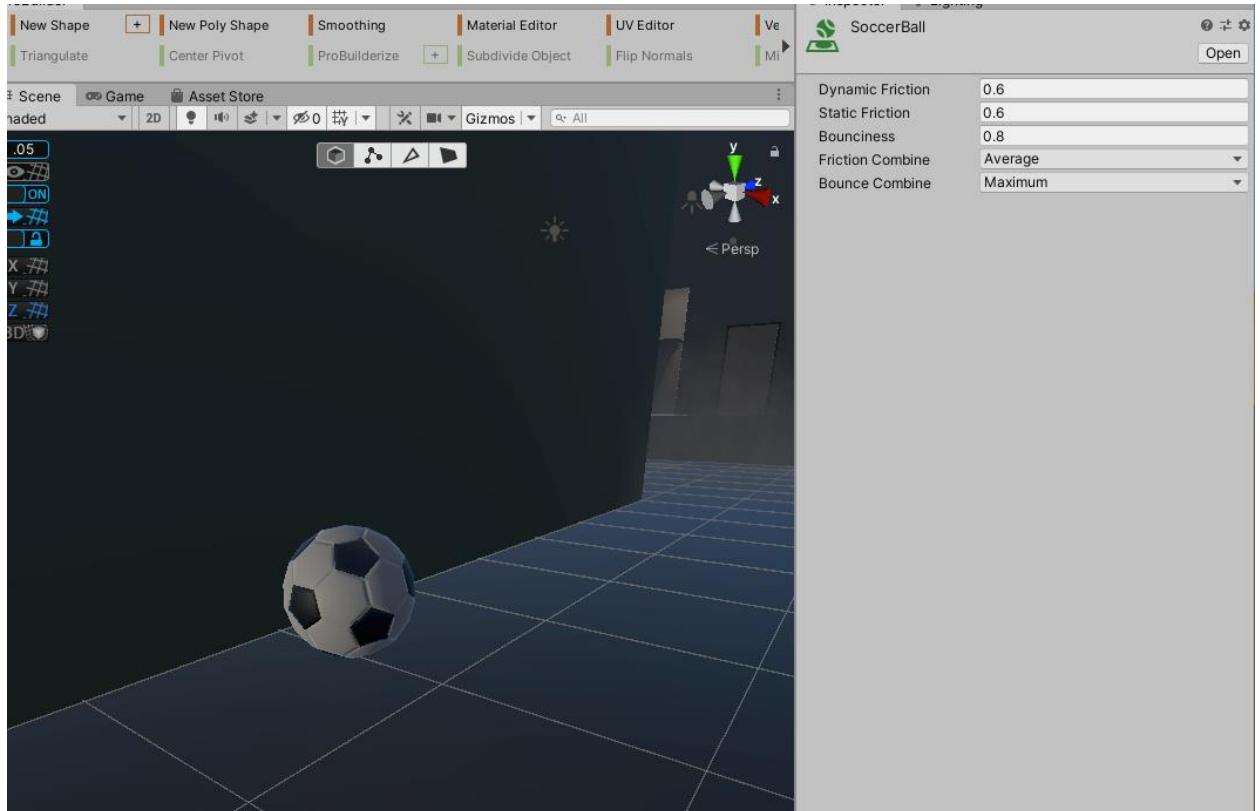


Figure 16. Soccer ball with a bouncy physic material

Also made a slippery ramp physic material.

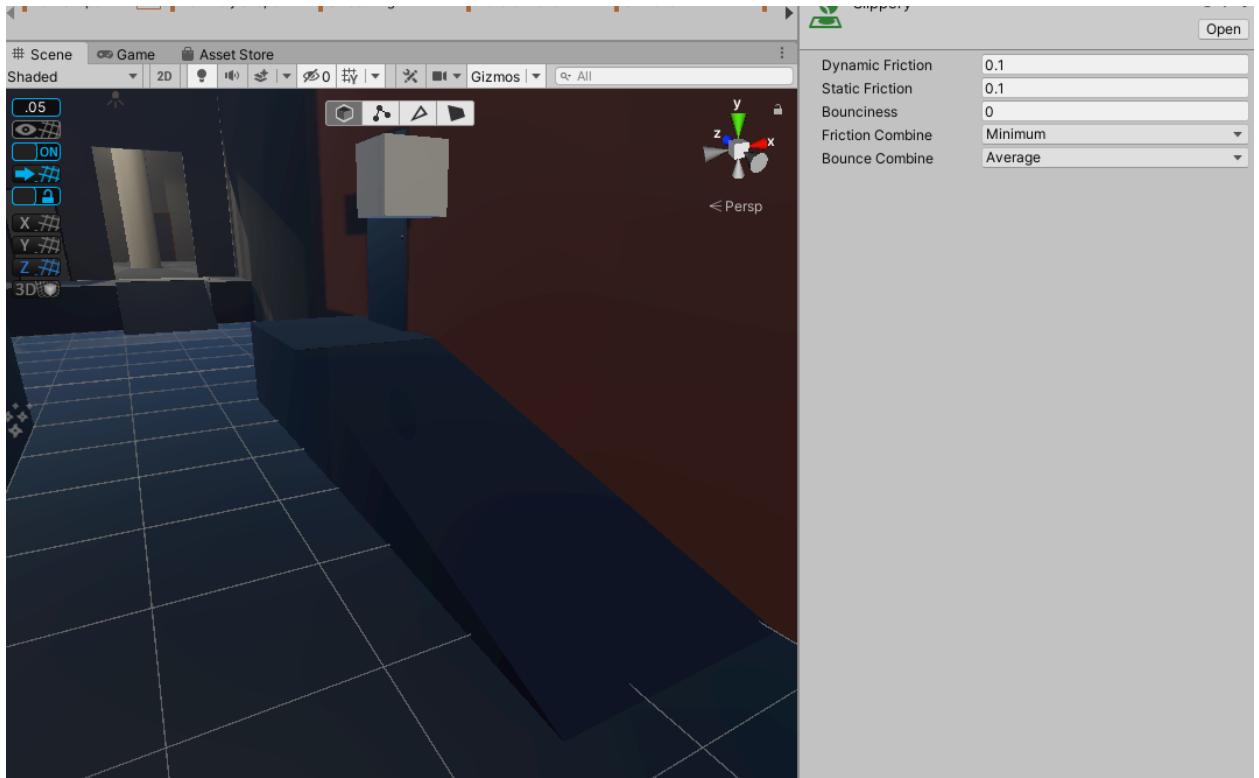


Figure 17. Slippery ramp physic material

Task #7. Create 4 types of objects that use *OnCollisionEnter* / *OnCollisionEnter2D* / *OnTriggerEnter* / *OnTriggerEnter2D*

Active ragdolls are now a bit improved. Right now, it is possible to apply an animation to the ragdoll, and the ragdoll will try to mimic the animation, as closely as possible. The character was also redone with configurable joints instead of character joints to be able to be able to use the joint drives, for specifying the target rotations of the joints, which provides more physically accurate character movements.

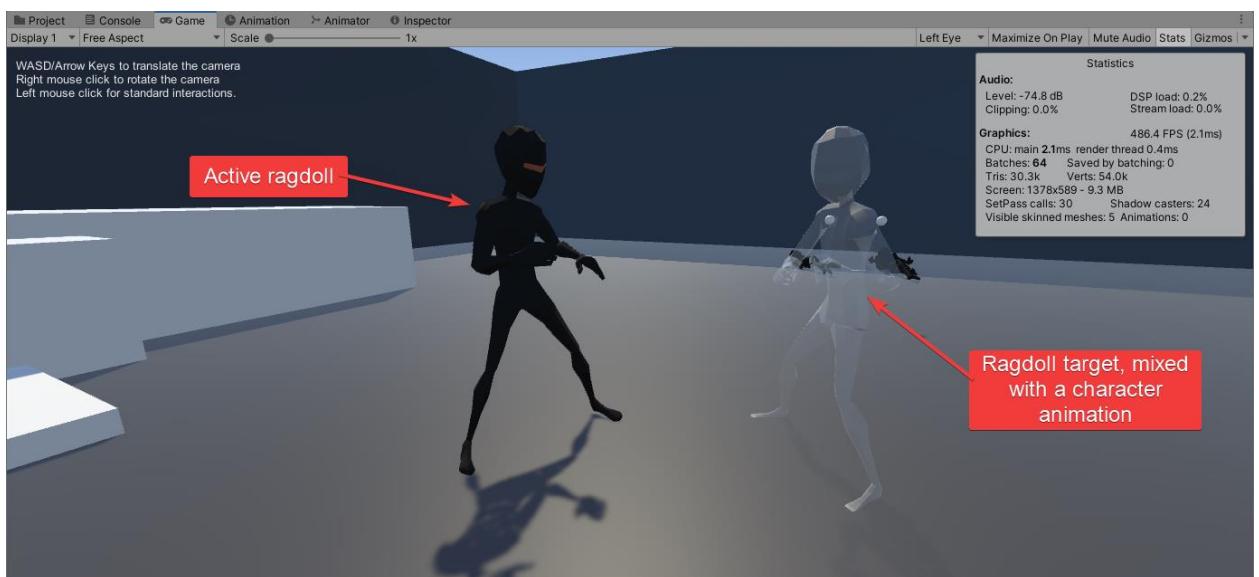


Figure 18. Active ragdoll, following an animation

Task #8. Assign optimal colliders for the environment objects that are not moving and set their flags to static (test performance before and after)

To test physics performance using static objects, I duplicated a bunch of falling balls all around the scene and inspected the profile on how much CPU time is used on physics. Since there is not a lot of physics going on in the game world, there was no noticeable performance difference.

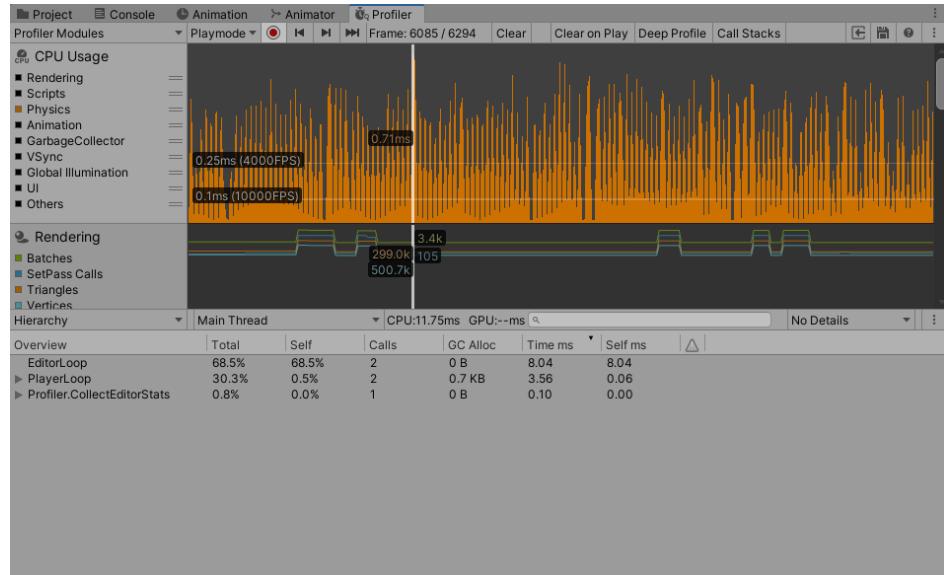


Figure 19. Before marking objects as static

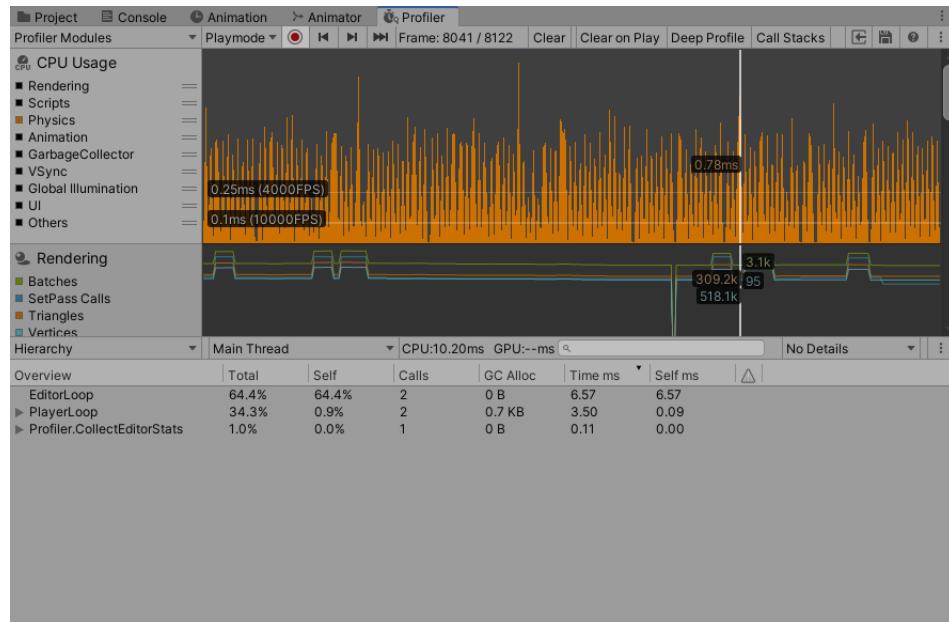


Figure 20. After marking objects as static

Task #9. Bake a lightmap and measure performance

To test light performance, I set all non-moving objects to static, including lights, enabled mixed mode in all static lights and baked the lightmaps using the default lightmapper settings. The game uses a differed rendering path.

There was no noticeable performance improvement, although render thread CPU time decreased from 1.1ms to 0.9ms, batches decreased from 294 to 65 and saved by batching increased from 4 to 260. With a more complex map, this would definitely have a more noticeable difference.

The quality of lighting around the map also increased, because the lightmapper baked in ambient occlusion and global illumination.

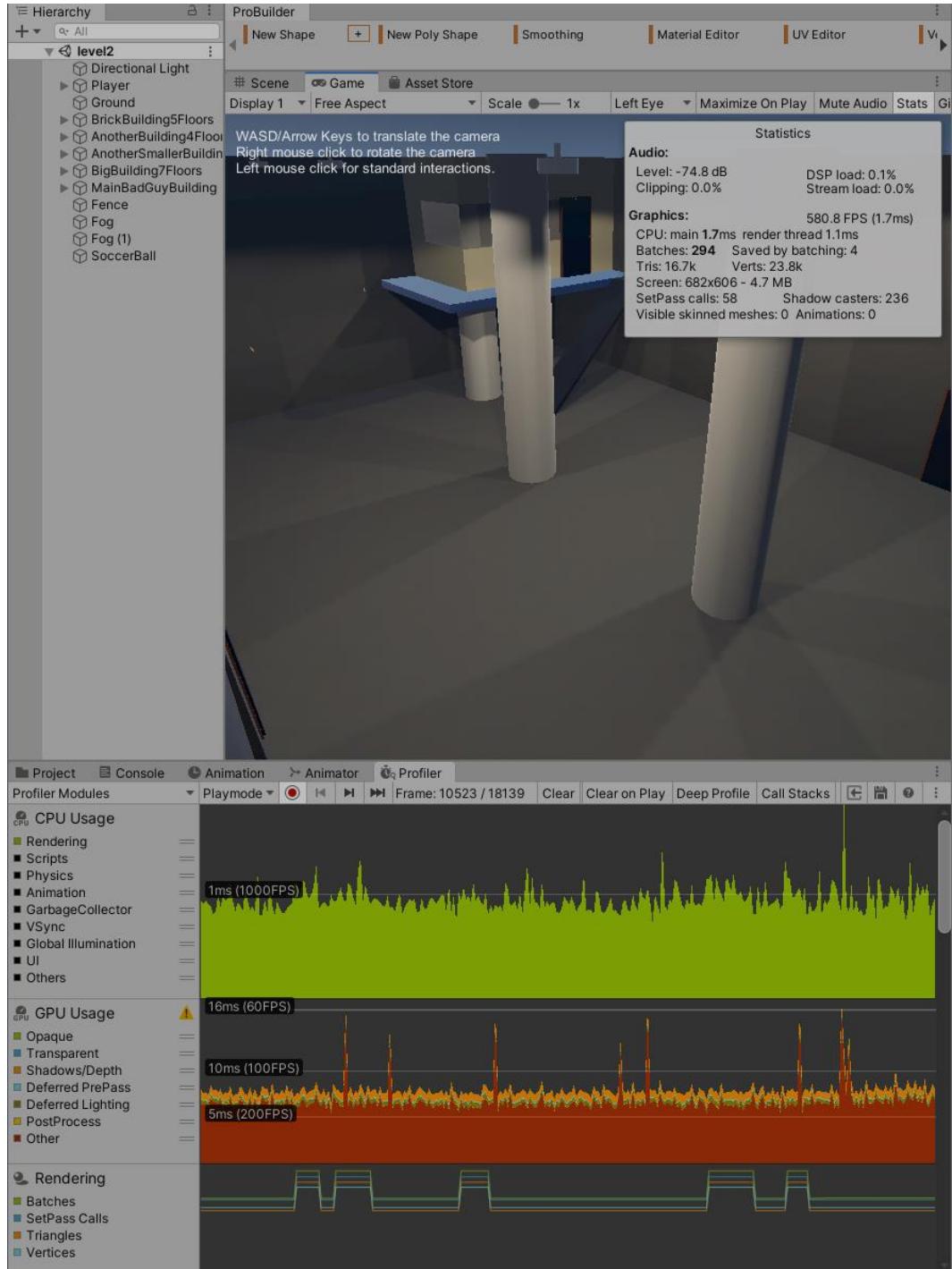


Figure 21. Light performance before lightmapping

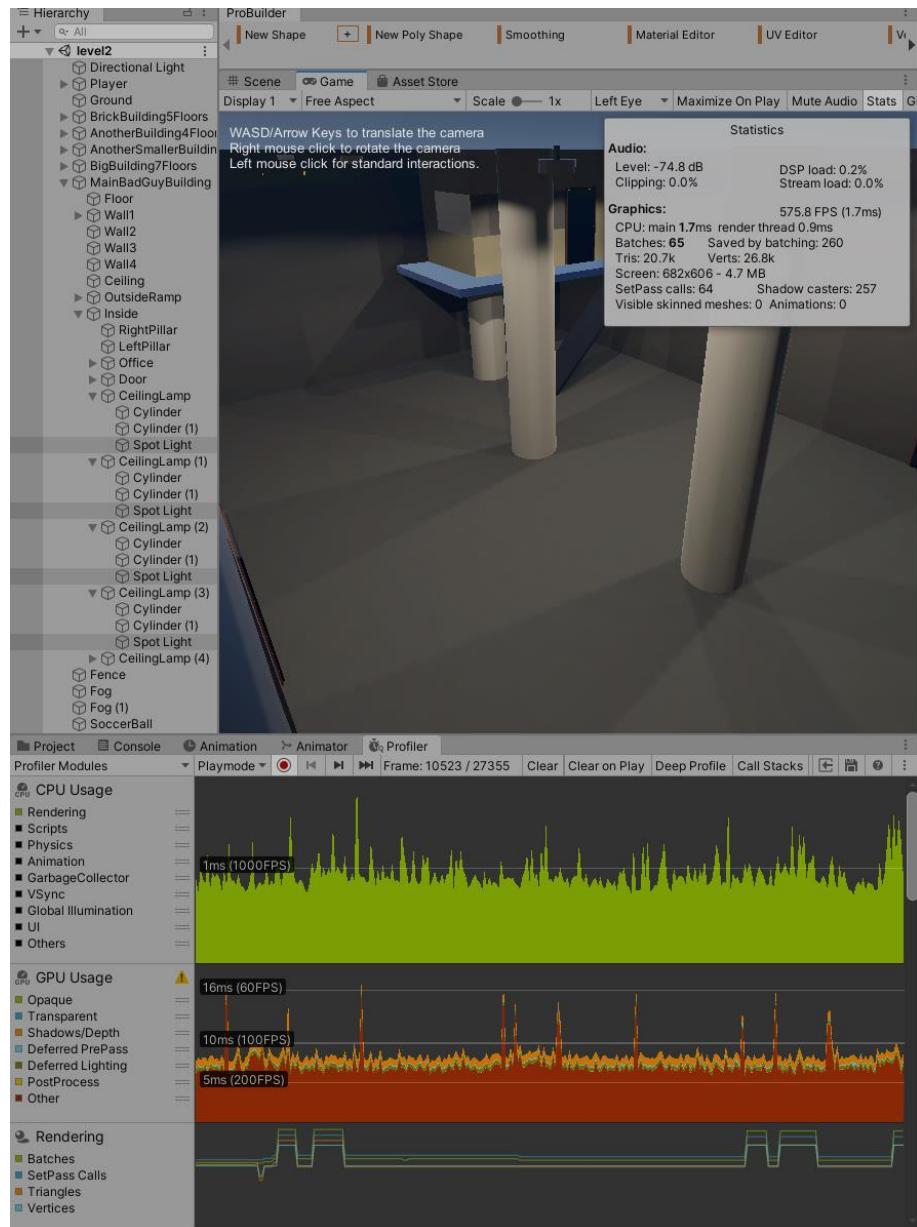


Figure 22. Light performance after lightmapping

Task #10. Optimize all textures depending on their parameters and measure graphical memory load

There are almost no textures used in the game world, since the game uses a simple low-poly, no-textures (flat) design.

Task #11. Try hard vs soft shadows, different quality settings and measure the performance

To test the performance difference of shadows, I tested with them on and off. Turning off the shadows reduced the render thread from 0.9ms to 0.8ms, which in turn increased FPS by a bit.

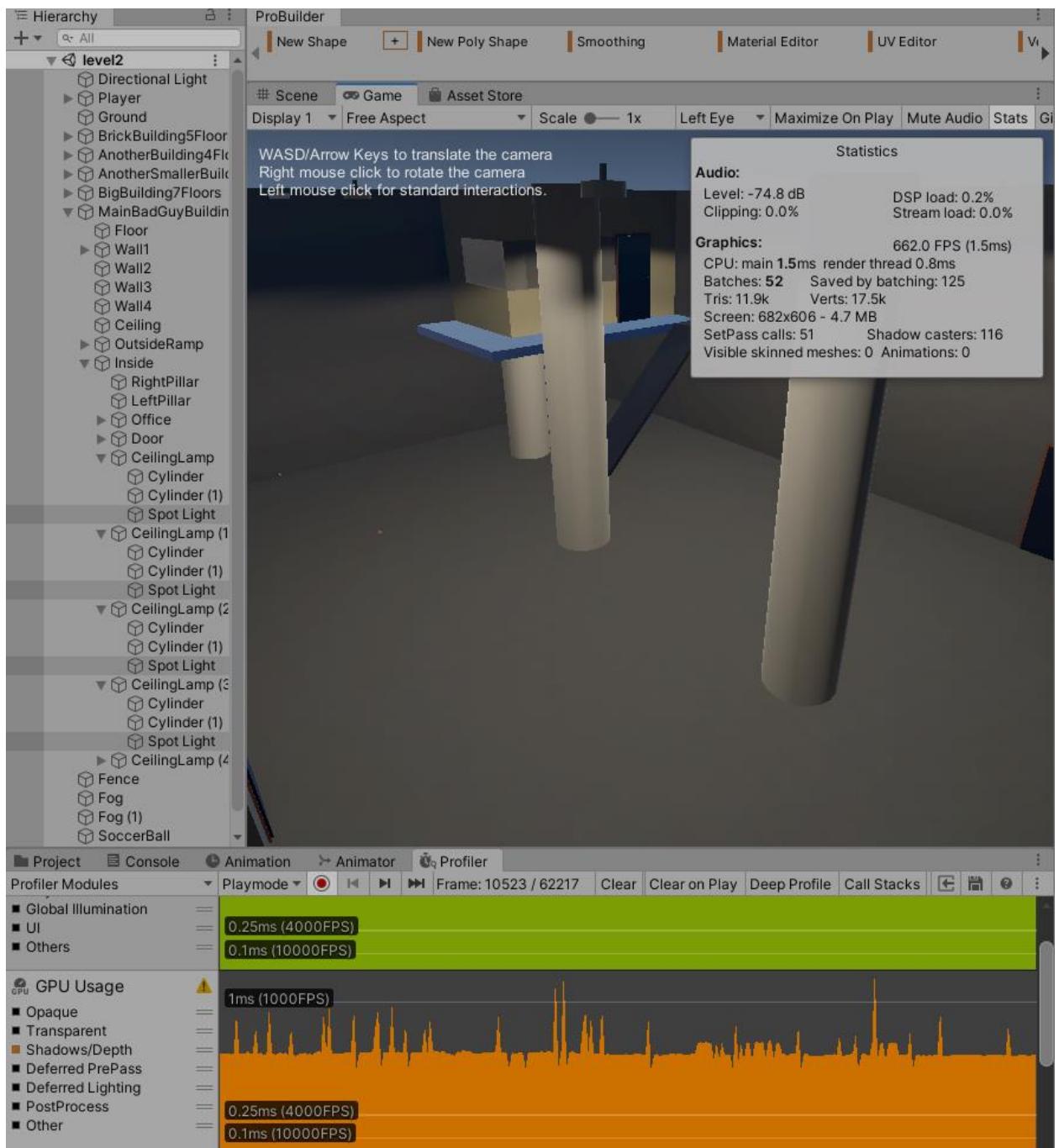


Figure 23. Scene without shadows

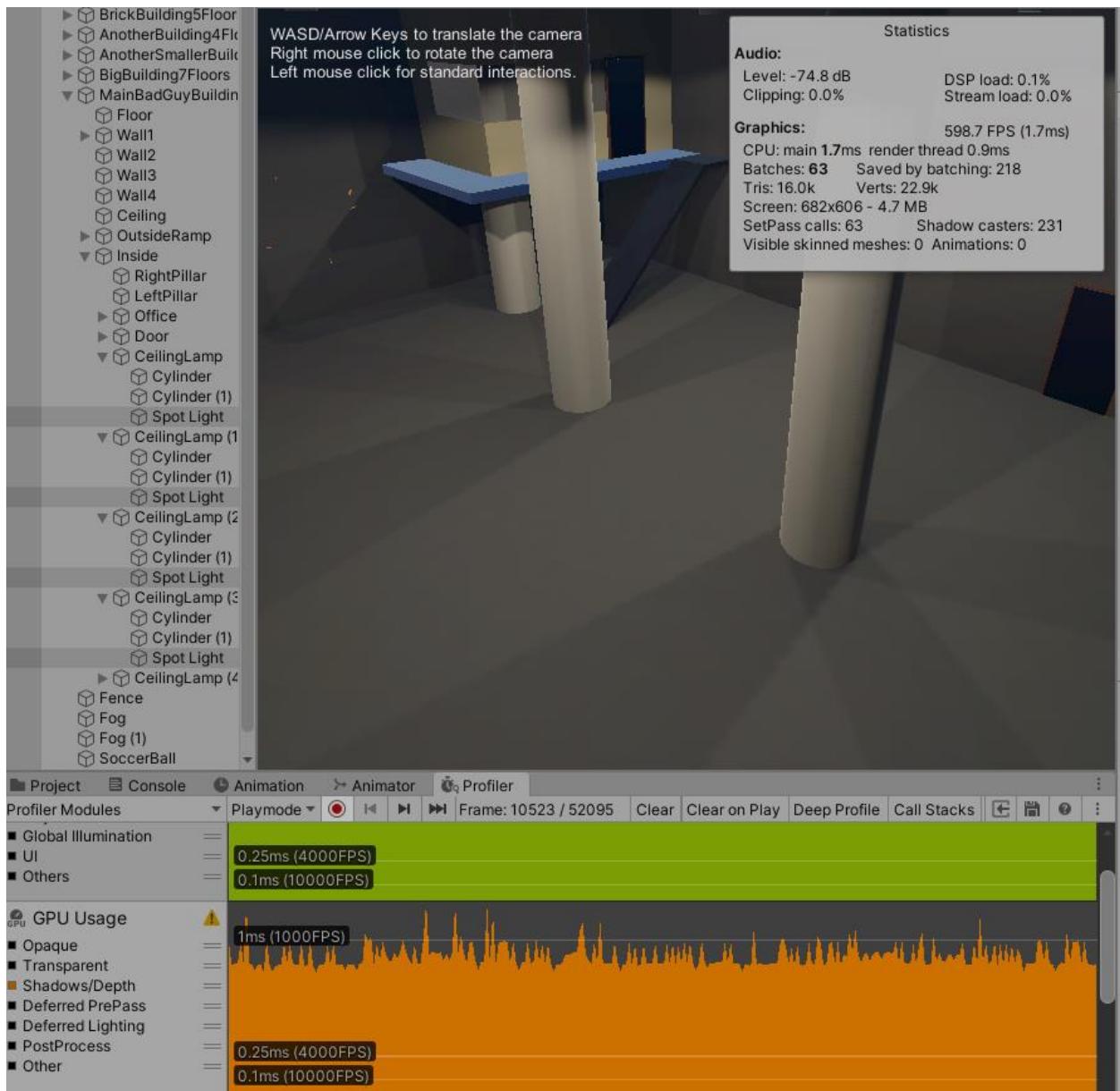


Figure 24. Scene with soft shadows

Defense task

Add a plane and a ball (with a rigidbody) to the scene and make it bouncy infinitely on the plane using a script.

1. Added a plane and a ball
2. Attached a rigidbody on the ball
3. Created a script called “BouncyBall” on the ball
4. Added this code to the script:

```

public class BouncyBall : MonoBehaviour {
    private new Rigidbody rigidbody;

    private void Awake() {
        rigidbody = GetComponent<Rigidbody>();
    }

    private void OnCollisionEnter(Collision other) {
        rigidbody.AddForce(0, 5, 0, ForceMode.Impulse);
    }
}

```

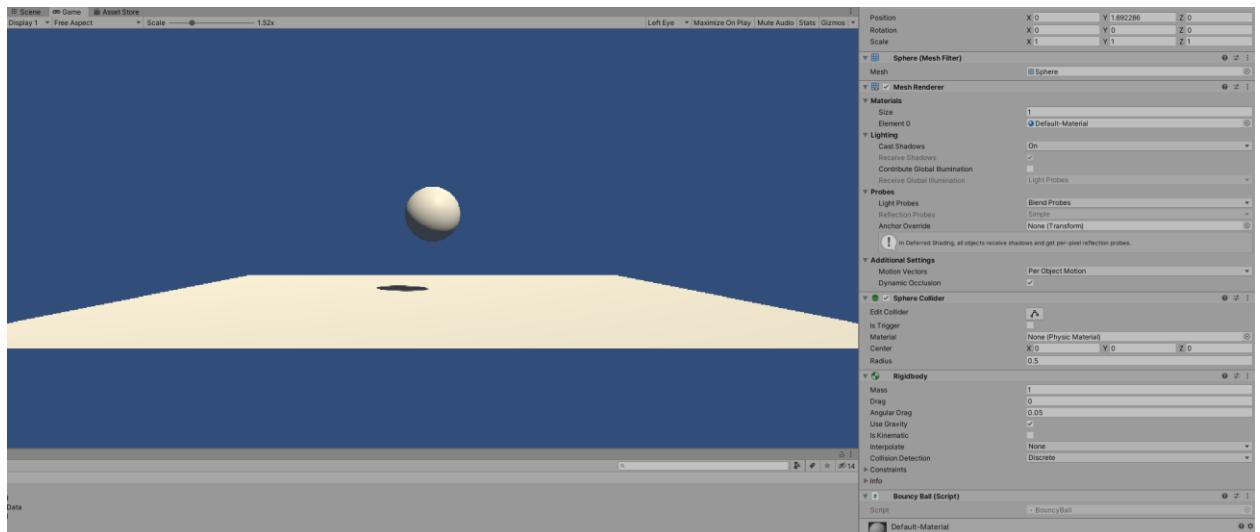


Figure 25. Defense task: infinitely bouncing ball

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.*

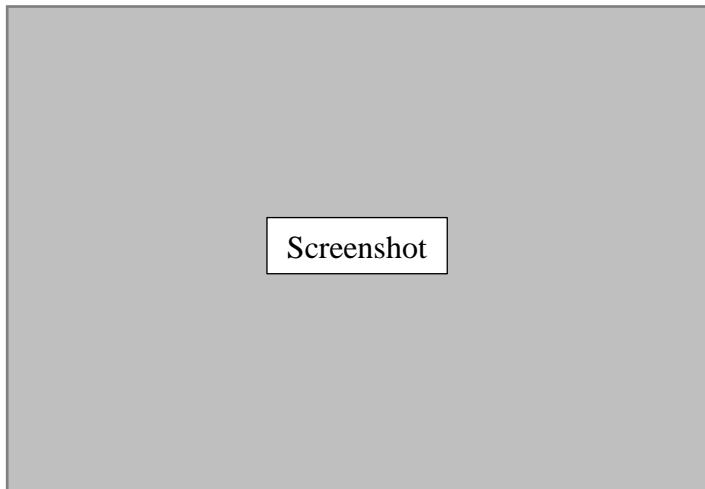


Figure 26. Screenshot #1

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 Code

Table 1. 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.*

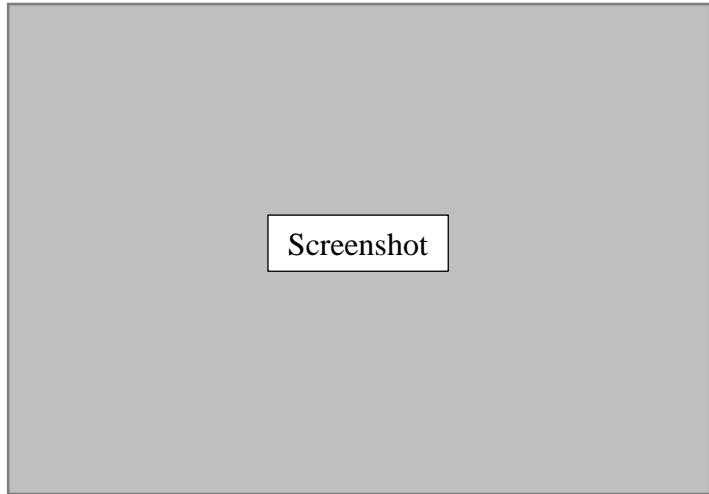


Figure 27. Screenshot #2

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 Code

Table 2. 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.*

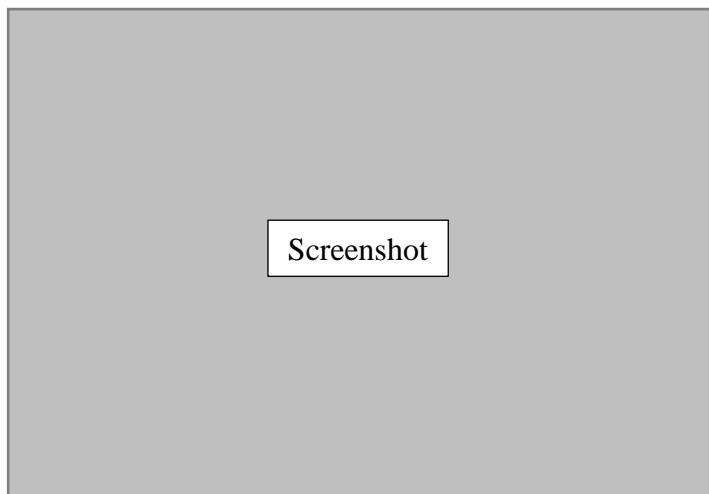


Figure 28. Screenshot #3

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 Code

Table 3. 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.*

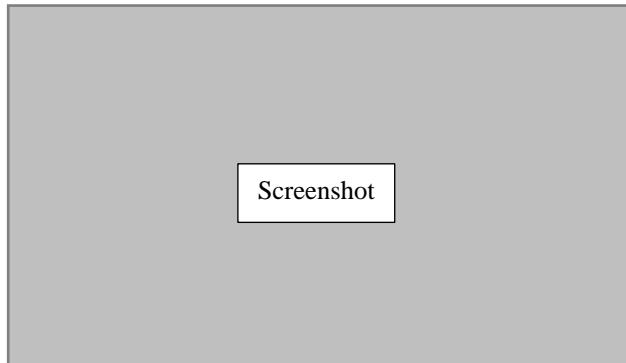


Figure 29. Screenshot #5

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

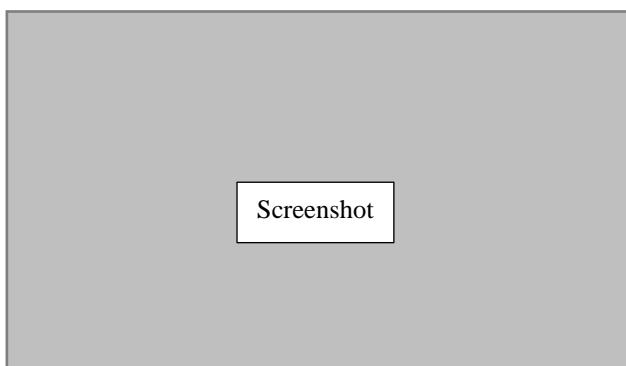


Figure 30. Screenshot #5

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

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

Literature list

1. Source #1. *Url*
2. Source #2. *Url*
3. ...
4. Source #N. *Url*

ANNEX

All source code is contained in this part.