THE HOUSE

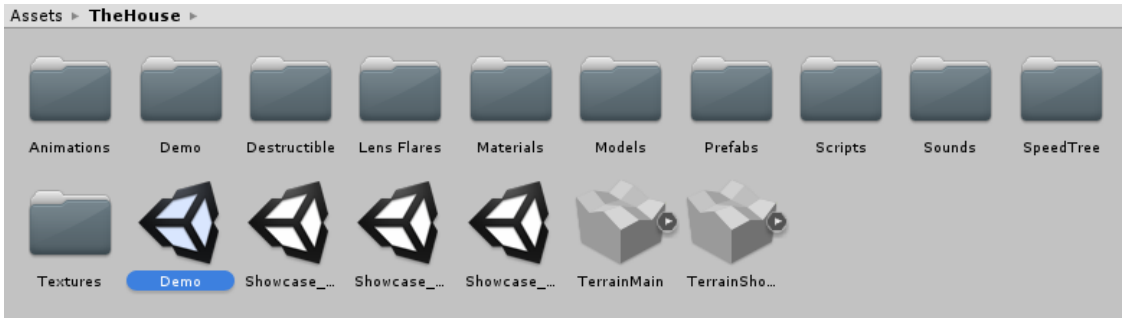
DOCUMENTATION

Thank you for purchasing this package. And you made the right choice.

Let's get going. In this package, you only need a couple of steps to configure the project for PC gameplay.

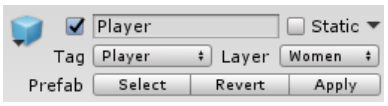
DEMO SCENE

1. First, open the demo of the project scene placed in the “The House” folder.



2. In the hierarchy let’s find a Player game object:

3. You will see 2 player objects. It is okay, you need the second **Player**, is a woman, where the first inactive **Player** is a man. In order to determine that the player is the woman tells you the Women layer in the inspector this layer is not handled only by a flashlight on the weapon.



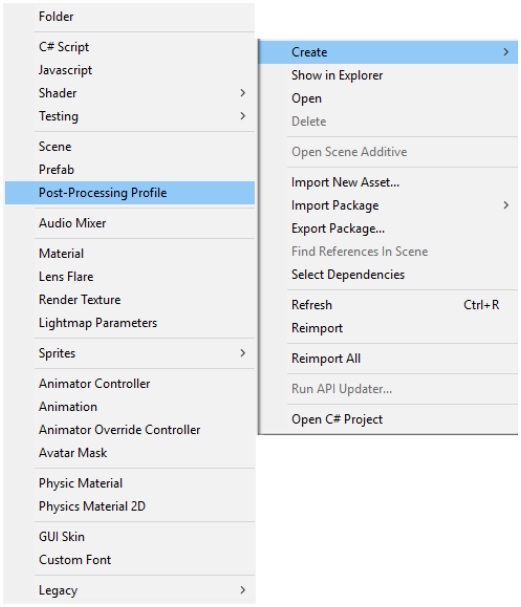
GRAPHICS

Attention: Speed Trees are not included because Per SpeedTree's EULA that oversees all of their proprietary content and it is a violation to redistribute or resell any works created from

SpeedTree applications or obtained through the SpeedTree marketplace.

1. For beautiful graphics you need to download the package called Post Processing Stack by Unity Technologies from the Asset Store it's free.

2. When you imported this asset into project you need to create Post Processing Profile like on the next image:

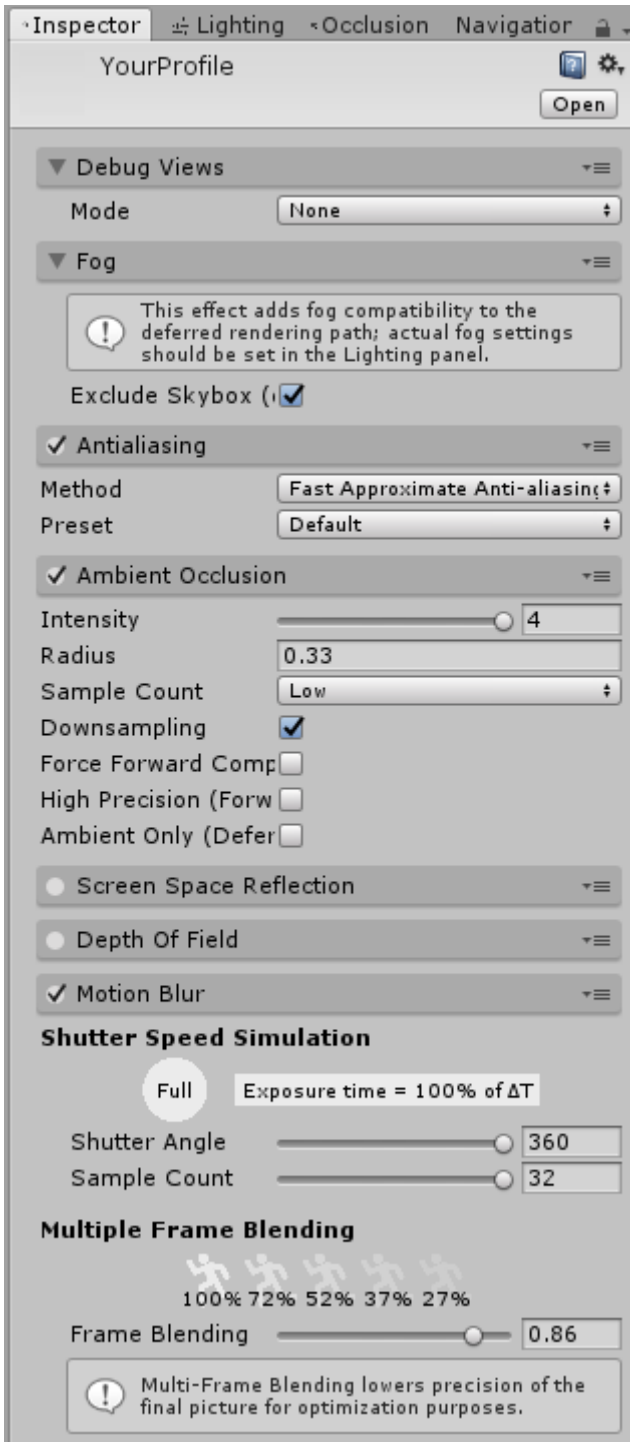


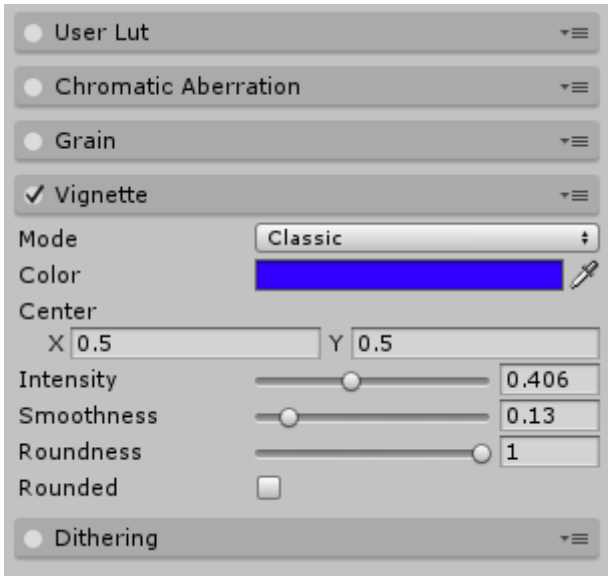
3. After that you will have a profile 

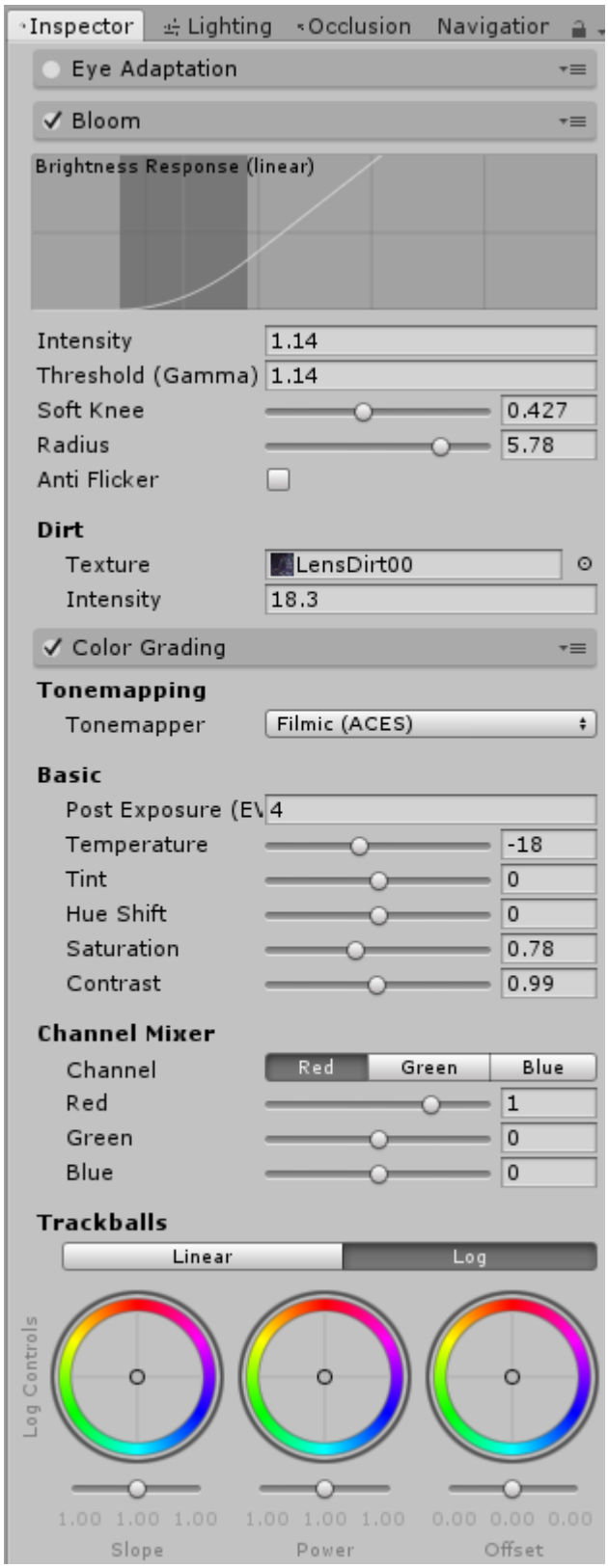
in the project, rename it as you wish.

4. Now back to our woman **Player** and find in her structure the main camera and attach the component called Post Processing Behaviour to the **Main Camera** in the inspector like on the next screen:

5. Ok, click on the profile you created in the 3rd paragraph and adjust the settings like on the next screens:



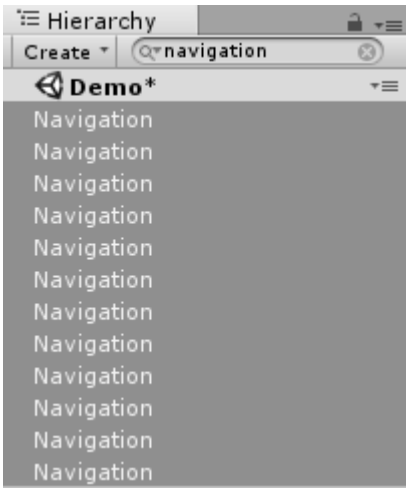


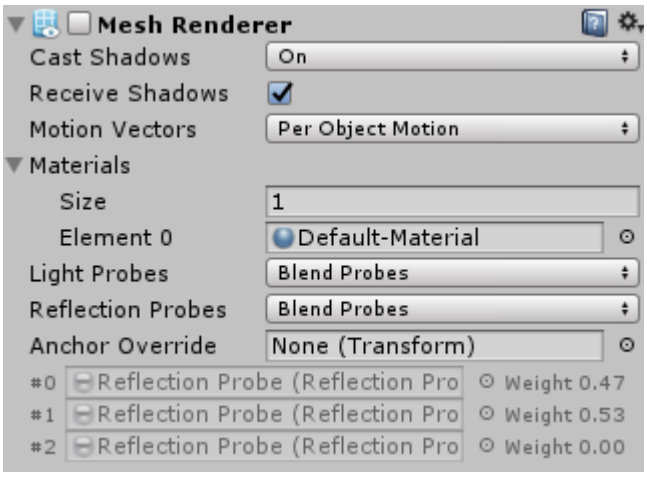


Note! Similarly set the same profile on the main camera of the man **Player**.

NAVIGATION

As soon as you set up the graphics. Let's move on to navigation. The level is built on navigation. All enemies move with a pre-calculated navigation. Before miscalculation of navigation, you need to arrange physical limitations, in this you will help already prepared at the level of game objects called **Navigation** in the hierarchy.





By default the Mesh Renderer disabled because we don’t need render it. Enable checkbox before miscalculation of navigation paths.

LEVEL TRIGGERS

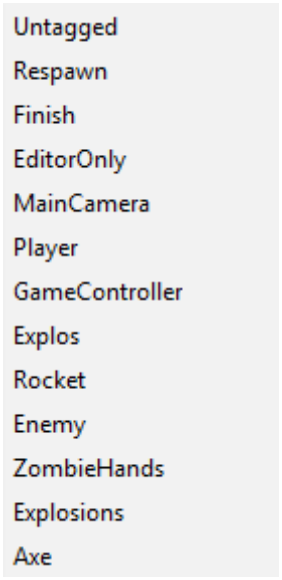
At the level there are several areas that are treated as triggers. These are the usual patterns in the form of game objects (boxes). Below are the names of the triggers in the hierarchy:

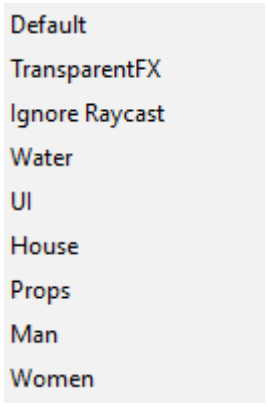
- **RainCameraTriggers** is a game object that controls the appearance of the rain effect outdoor in different conditions of the player's location. (More info about working with rain triggers look into Scripts chapter)
- **RainCameraTriggersIndoorView** this game object is located in the areas from which rain will be seen.
- **FootstepZone_(name)** these are objects for playing a specific sound while walking.
- **WallExplosionTrigger** it is used to determine the boundaries of the area where the collision with the object occurs, for example when a bottle hits this wall, the explosion action will work.
- **AxeExplosionField** this is an object that checks for a collision with game objects, such as a rocket, or a huge sledgehammer. When the sledgehammer touches this field, a shock wave appears in the form of a previously prepared particle system called **EarthExplosionHolder** placed in the destructible Prefabs folder in the project.

TAGS AND LAYERS

It is very important to follow all tags and layers, check them:

TAGS





LAYERS

This is important, as some objects are processed by tags and layers.

PLAYER SETTINGS

The structure of the game moving model of the player or the enemy contains a skeleton consisting of bones. These player settings are already compatible with VR mode.
Let's take another look at the player's woman. On her structure as a player:

HitCollider it is used to determine a collision that looks like a blow when a zombie strikes it.

Rotator itself looks at the object that is embedded in the main camera, and the main camera rotates this embedded object. Thus, the rotator performs a smooth rotation of the character and its child bones, for example as the back bone and of the main camera, this is necessary so that there are no problems with rotating the character in the VR mode with head, as well as in the normal mode with mouse.

RainMaterialParameterControl controls the man or woman model shader during the rain (dry or wet).

OnScreenBlood_Canvas used when the player hits. The screen shows the impact effect.

SpineView this object is needed in order to correctly handle the axis of rotation of the Spine1 bone.

Rain using when the player crossed the border of the rain trigger.

TargetViewCamera_(number) they are needed for correct rotation of the Spine1 bone and looking forward correctly while switching animations. In case the bones of your model are not set correctly. You can set them up manually, without changing the graphics editors.

M4_Walk_Fwd contains 6 animations for playing.

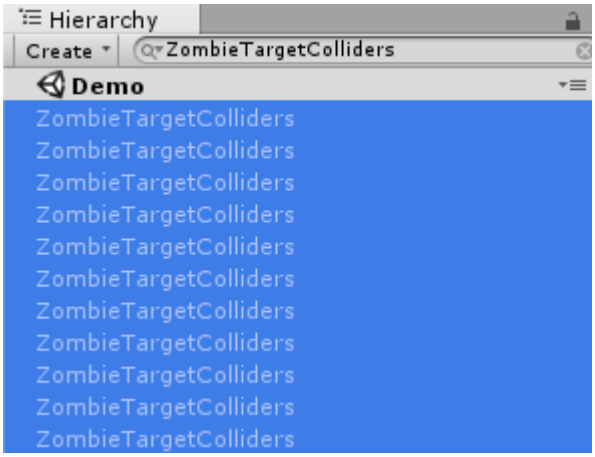
M4 contain machine gun mesh, flashlights, particles. By default shotgun are not included. You can put a shotgun in the Right hand but do not forget to keep the hierarchy like for M4 machinegun.

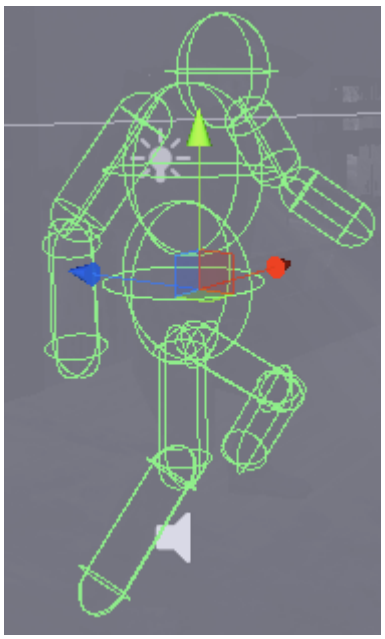
Note! A similar hierarchy contains the man Player.

Copter game object. AI based on animations and using the script called CopterLogic. The script is not associated with any other scripts in the project. The helicopter has two bullet spawners for shooting.

ENEMY SETTINGS

Let's look on **ZombieHolderMain** game object placed in hierarchy. This is the first enemy at the level on which the continuation of the chain of actions depends. Killing him, you will attack the next zombie but of a different type, and we will consider them. First let's talk about the first zombie. This zombie is idle, and the corresponding animation is reproduced, all animations of enemies, you can find in the Models-Characters-RapZombie folder of the project. After you shot at him, he will react at the expense of the hit animation then he will go to you. For these 3 actions, ZombieLogic and ZombieLife scripts respond (more detail you can learn directly in this scripts). These scripts handle the commands of the beginning of a certain action, the completion of the action, the behavior of the zombies. Scripts are also attached by analogy to all other zombies. This zombie contains two channel sound effects **ChannelSFX_(number)** this is necessary for the sounds of steps, voices and attacks to be processed separately and audible.
Let's find **ZombieTargetColliders:**





These are triggers that are used to take damage, and handle the event of hitting and playing the corresponding animation. They also need to use a **ZombieLife** script. Triggers send certain variables to the same script, which only is directly on the zombie. Variables received during the damage from the Player (Look in the script that you will learn this process). **HandColliderRight** and **HandColliderLeft** (Inactive by default) these are triggers that convey the state of the blow to the player. When these triggers are active and collided with the **Player**, the effect of blood appears on the screen, the impact was received.
After this zombie is killed, the next action with the effect of the knocked out door is triggered, and a **ZombieHolderEpileptic** appears.

ZombieHolderEpileptic the epileptic effect is achieved by mixing the animations of the upper and lower bones, where the upper bone is responsible for the body, and the lower bone is responsible for the legs. In the rest is similar to the main zombie.

ZombieAxe he has turned off the animation of the hit reaction, since this is not necessary, for reasons of logic.

ZombieHolderBottleStanding it's a zombies who are inactive but at certain times throw objects, such as a bottle. Based on switching animations. They contain a game object that causes the zombie to look at the target.

ZombieRunBottle a once-appeared and disappeared zombie on the background, is applicable to depicting the presence of someone outside the home. It’s contain **ThrowingHolder** game object as target to running and **ZombieThrowingSpineLook** for looking and rotating only upper body.

ZombieHolderBottle uses animations applicable to actions with objects like run with bottle.

BigZombie this zombie appears at the end of the game, first follows to the target which is processed by navigation, then the zombie stops, turns and looks toward the player. Uses similar scripts like all other zombies (Look in the script that you will learn this process).

SCRIPTS

AnimationSpeeds

Zombie animations speed controller. Used in the first frame, the optimal values are set.
The script must be attached to the any animated object, to control the playback speed of the specified animation.

BottleExplosion

This script are using on Bottles or can be use on any destructible objects as you wish.

BottleSpawner

This script generates a new object after a specified time, and gives it an ejection force of bottle (BottleSpawner)
First 4 gameobjects in the hierarchy called BottleSpawner using this script when the player moves to the second action scenario with Bottles. The bottles are flying into the house and exploding.

CameraRotation

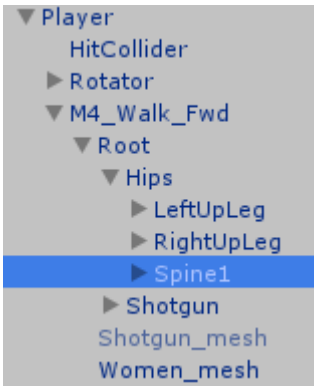
Turn off this script from main camera if you're turned VR mode
Rotates **Main Camera** based on mouse movements. Script attached to **Main Camera** game object in the inspector.

CameraShake

Shake effect. This script placed on Zombie gameObject in the inspector. Used when an action has occurred (Attack, Kick, Axe, Bottle)

ComboTest

This script is used on the bone of the main characters (Spine1). Ensures correct rotation along the X axis.



Allows the skeleton to rotate in the desired axes by motions (idle, walk, run)

CopterBulletsExplosions

This script are using on **Rocket** game object analog **BottleExplosion** script or can be use on any destructible objects as you wish.

CopterLogic

Contains the logic of the behavior of the helicopter. The **Copter** shoots with rockets at random time.

Destroy

General destoying objects. Must be placed on any object in the inspector for destroying.

DestroyObject

Destroying an object for a specific case

DestructibleProps

The script must be on physical objects that will be destroyed. For example **Angel** statue game object on the cemetery.

FootStepControls

Simple footsteps controller. Playing footstep sounds in the specific areas. This script attached to "**FootstepZone**" gameObjects in the hierarchy.

LeakFireAfterExplosion

This script can be using with Leak fire effects but in order to optimization, we skip this function, and make it an alternative method of ordinary explosion. Use this script only to obtain an acceptable visual effect.

PlayerControl

It controls the behavior of the player. Changes animations while moving. Exchanges variables between Shooting script. Where **PlayAction** coroutine is switches actions according to the specified scenario. As it was said in the enemy settings chapter, on the example of the main zombie, when you shoot it, the first action will start according to the scenario. As the main zombie must once perform certain actions and subtleties when the optimal picture is achieved according to a given scenario, then the variable **FirstAction** of **ZombieLogic** and **ZombieLife** scripts is initially set. This variable instantly switches the state of the main zombie. For other zombies, their appearance is controlled by the **ActionNum** variable.
In order to add a new Action, you need to adhere to the following code structure:

```
IEnumerator PlayAction(int ActionNum)
    {
        yield return new WaitForSeconds(5); //wait before launch next action
        {
            if (ActionNum == 1)
            {
                yield return new WaitForSeconds(2); //waiting as you want
                //Do something
            }

            if (ActionNum == 2)
```

```
            {
              //Do something
            }

            if (ActionNum == N)
            {
              //Do something
            }

        }
```

The value of **ActionNum** and **nextAction** gets from the script of ZombieLogic when the zombie was killed.
There is a Reloaded coroutine. The values of variables **BulletsCount** and **capacity** are sent to the **Shooting** script:

```
IEnumerator Reloaded()
    {

yield return new WaitForSeconds(1.8f);

Player.GetComponent<Shooting>().BulletsCount += 100; //sending to Shooting script 100
bullets after animation played

Player.GetComponent<Shooting>().capacity -= Player.GetComponent<Shooting>().BulletsCount;
//reducing capacity
        DontMove = false;
    }
```

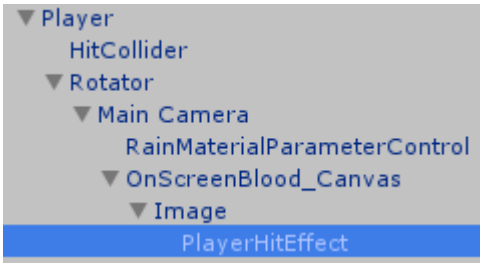
Waiting until "reloading" animation of Player will be played.

PlayerHit

Checks if **Zombie** hits the **Player**. Sends the value of the variable **PlayerKick** to the **Player** script when **HandColliderRight** or **HandColliderLeft** of Zombie game object collided with **HitCollider** game object placed on **Player** as child game object in the hierarchy. **HitCollider** must be triggered.

PlayerHitEffect

On screen blood controlled by **Canvas**. Attached to **PlayerHitEffect** gameobject in the inspector.



RainCameraTrigger

Used to control the rain in front of the camera. Attached to the triggers called "**RainCameraTriggers**" and "**RainCameraTriggersIndoorView**" game objects in the hierarchy.
Shows rain particle in front of **Main Camera** and turns the properties of physical material into wet or dry on **Player** character.

RainMaterialParameter

Turns the properties of physical material into wet or dry on Player character material (for example **Girl _AlbedoM** material on **Women_mesh** in the hierarchy)

RotateObject

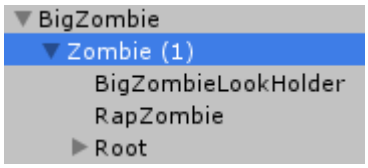
Rotating any objects for example blades of copter or bottles

Shooting

This script is designed to control the remaining bullets, shots and hits at enemies. It contains variables for controlling particles, flickering light, flashing.
This script is always must accessed by a **PlayerControl** script. For more information look into this script.

SmoothLookAT

Looking script with a given type. Where if game object is not **Copter** game object then all other objects look at the objects specified by the name but if **Copter** you should assign the game object you are viewing directly to the inspector. **BigZombie** will always smoothing look at **Player** when this **BigZombie** stops. The script attached to this game object in the **BigZombie** hierarchy:



SmoothLookAT1

Additional looking script.

TriggererExplosion

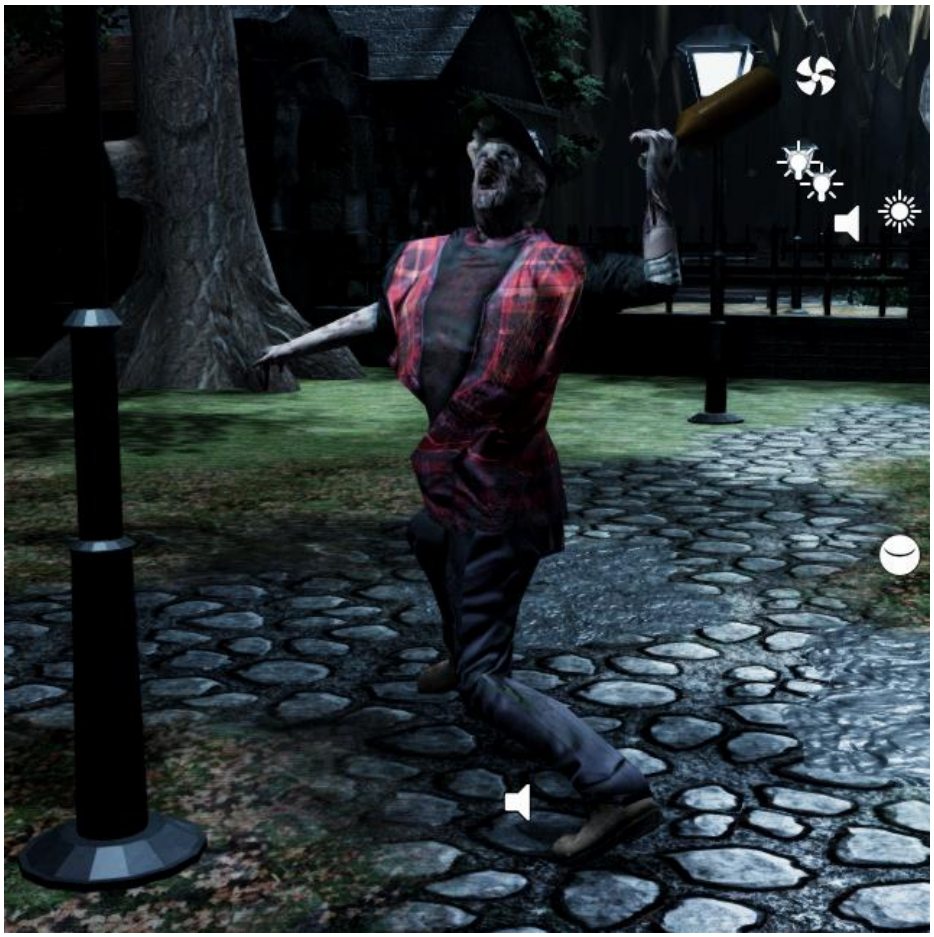
The script is used for a shock wave from a huge sledgehammer of BigZombie.
The reaction of the shock wave occurs due to the collision of the sphere of Sledgehammer and the floor, which is under the terrain ("**AxeExplosionField**" game object in the hierarchy).

UVW_Animated

Animate UVW's of any material for example using for Waterfall mesh.

ZombieBottleLookBone

Using on Spine bone when zombie running at start of second action (ZombieRunBottle gameobject in the hierarchy - Spine1)



ZombieLife

Provide zombie hit reaction
Interacts with the **ZombieLogic** and **Shooting** scripts
Using on **Zombie** hit collisions and **Zombie** game object interacts with each other.

ZombieLogic

This script is used for the logic of zombie behavior for a particular type base on navigation agents.

VR GRAPHICS OPTIMIZATIONS

This project can be used for VR mode. It is enough to understand the basic principles of project customization. All models present in the project are initially designed to support mobile devices and VR mode. Before building the project for the Oculus rift or the HTC vive you need to import

those SDK from official page. This SDKs not included in this project because it is prohibited, based on the terms of license distribution. To properly prepare a project for playback in VR mode, recommended to follow this steps:

- Remove all dynamic lights and shadows. Or bake static lighting with static shadows.
- Use only mobile shaders playing on mobile.
- Use the same materials and textures where possible
- Optimize the resolution of textures, and the number of generated particle effects
- Set the state of objects in static
- Reduce the number of transparent objects, such as trees, grass and other vegetation.
- Set the optimal project quality in the project settings.
- Carefully use the computer's resources, turn off post-effects, where possible.

Demo scene uses Occlusion Culling for best cut out the invisible parts of the level at the **Main Camera**.

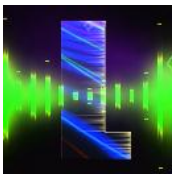
CONCLUSION

Now you know all the principles of project configuration, more detailed information on scripts, you can find inside them in the form of comments.

Maybe you will be interested in other projects from the Sergey Shushunov publisher?

MOBA FANTASY ENVIRONMENT

This pack contains over 50 prefabs to create stylized fantasy in moba genre.



LIAVIS MUSIC PLAYER STARTER PACK

This package determining Basic concept of media player. Includes Beat Detection in realtime.

GLOW VR

This package is the base to create a set of virtual reality games in the arcade genre. Create your own abstract worlds for more exciting adventures, all you need is in this package.

EQUILIBRIUM VR

This is a completed project with an episode and a simulation of the eagle's flight. With colorful game levels, with full VR mode support

CASTLE BATTLE VR

The battle of castles is a game performed in a stylized style. The virtual world is as if drawn on paper.
3 kingdoms are fighting for themselves. VR ready project.