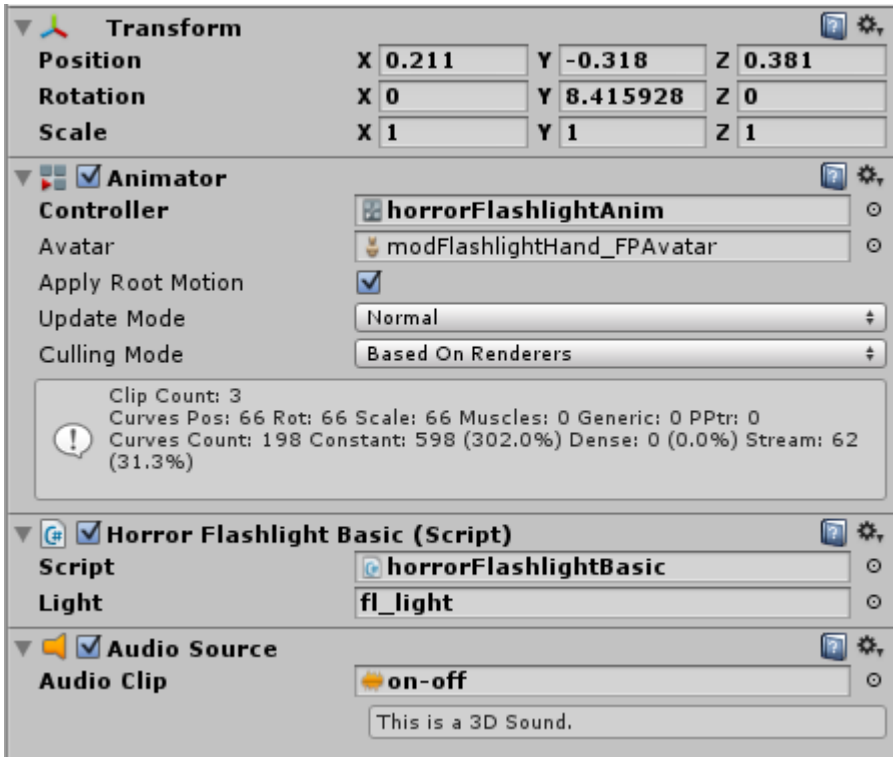
FPP Horror Flashlight – Basic Pack

MANUAL

1. **Deployment**
   After purchase download the contents of the package into Your Unity assets folder, open the package folder and check if it contains all elements listed in point 2. This guide is mostly a troubleshooting guide, the package should be good to go right after dropping it into the scene.

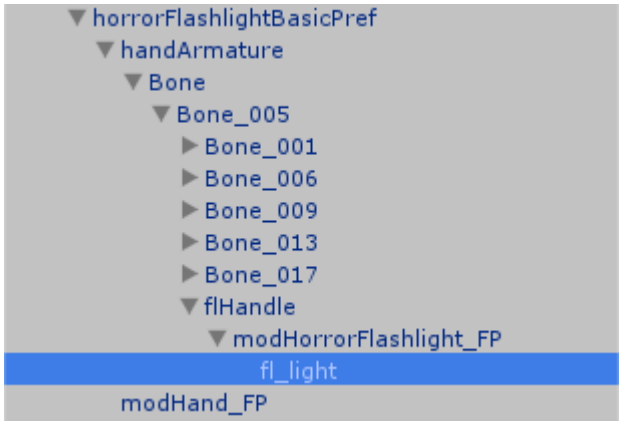
2. **Package elements**
   - "Materials" folder containing materials for both the flashlight and the hand
   - "sfx" folder containing the on/off switch audio
   - "Pickups" folder containing pickup model and textures for battery.
   - flashlightCookie.tga image
   - TGA normal and diffuse textures for both models
   - horrorFlashlightAnim Animator Controller
   - rigged FP hand model
   - Flashlight model
   - horrorFlashlightBasic C# script
   - **horrorFlashlightBasicPref prefab**

3. **Usage**
   Step 1. Drop the *horrorFlashlightBasicPref* prefab to Your scene or to the hierarchy. When done select the whole Game Object just dropped. In the Unity inspector check if all components and assignment are in place.



The prefab should contain: an **Animator** with the **horrorFlashlightAnim controller** assigned, **Horror Flashlight Basic Script** with a spotlight named **fl_light** attached, and an **Audio source** component with the **on-off.wav** sound attached.

Another thing to check is the light for **the flashlight**



In the hierarchy there should be a child object for the Flashlight model named ***fl_light*** with the flashlight cookie added in the inspector. You may customize the properties of this light like intensity, angle and range to what You wish it to appear and also use Your own light cookie if You wish. For default the light is set to **Realtime only** and **No Shadows**. If You

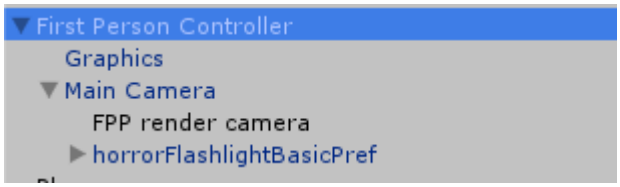
want the flashlight to cast shadows, You must turned them on in the light settings.

Make sure if the **fl_light** object is added in the **horror Flashlight Script** component in the **Light** field. If it isn't – simply drag'n' drop it there from the hierarchy.

If any of the elements are missing or unassigned add them to the Game Object as on the screenshots provided above.

Step 2. Check if all materials and textures are assigned to the models correctly. If there are any issues simply drag the appropriate materials to the objects.

Step 3. If all is in place You will now need a First Person controller. You can either use the built-in Unity First Person Controller available in the standard assets to import in the Character Controllers folder, or use Your own custom controller. Now – depending on what You want to achieve, You may attach the flashlight hand to the main camera or to the character controller in the hierarchy. This is an example how the hierarchy for the camera attachment should look like:



Drag and drop the Horror Flashlight prefab to Your designed hierarchy object and position it in a desired angle and distance according to the camera using the Scene and Game views. The **FPP render camera** object is an optional camera, which purpose is explained in the **FAQ section of this document**.

Step 4. If the prefab is assigned to the controller and You have checked all the instructions above You should be good to go! Run the test scene and test it. The hand should animate the idle state while standing and

the walk state while walking. Hit “F” to toggle the flashlight on and off.

4. **Control customization**

   As default the prefab uses F for the flashlight toggle, and WSAD for detecting movement. This is for beginners for fast deployment without additional setting end coding. You may customize this settings.

   Step 1. The on-off toggle

   Let’s open up the horror Flashlight Script for editing.

```
20      // Update is called once per frame
21      void Update () {
22
23          //On-off controlls
24          if (Input.GetKeyDown (KeyCode.F)) {
25              if (turnedOn == true) {
26                  spotlight.enabled = false;
27                  anim.SetTrigger("on-off");
28                  turnedOn = false;
29                  audio.Play ();
30              } else {
31
32                  spotlight.enabled = true;
33                  anim.SetTrigger("on-off");
34                  turnedOn = true;
35                  audio.Play ();
36              }
37
38
39          }
40          //On-off controlls end
```

   On the Update() void You have a commented On-off controls section. If You want to change the key assigned for the on-off toggle simply change the KeyCode in the **(KeyCode.F)** phrase in line **24**.

   For example, if You want the toggle to be under space, the code will look like this:

```
23          //On-off controlls
24          if (Input.GetKeyDown (KeyCode.Space)) {
```

   Also, If You have a Input list defined in the player settings You may use

the player input command here. For more information about the Input settings see the Unity Engine web manuals.

Step 2. The walk animation

```
42          // WALK ANIMATION CONTROLS
43          if (Input.GetKey (KeyCode.W) || Input.GetKey (KeyCode.S) || Input.GetKey (KeyCode.A) || Input.GetKey (KeyCode.D)) {
44              isWalking = true;
45                  } else {
46              isWalking = false;
47          }
48      if (isWalking) {
49              anim.SetBool("walking", true);
50                  } else {
51              anim.SetBool("walking", false) ;
52                  }
53      }
54      // WALK ANIMATION CONTROLS END
```

On the screenshot above we see the very basic Input check in line 43. It checks if the **W, S, A** or **D** keys are down every frame in the **Update()** void. Below this if statement is another one checking if the **isWalking** bool variable is set to **true**. That means, that if You want the walk animation to play in other conditions, You may modify or delete the input check *(lines 43 to 47)* and simply set the **isWalking bool** to true from any other custom script written into this one ore elsewhere in the project.

Still if You are a beginner You may leave the code as is or change the KeyCode assignments for all left/right/top/bottom and it will work just fine.

5. **FAQ**

How do I make my flashlight not glitch the walls?
This is why we used the **FPP render camera** object in the demo. There are a few ways to achieve this, but the easiest way to achieve this is to duplicate the main camera and make it a **child** of the base camera. When done, You must set the child camera **depth** do 1 (so it displays on top of the parent camera), then use the **Culling mask** setting to make it render only the FPP hand layer (You assign the layer to the hand and flashlight in the **inspector** top right corner). The last step is to change the **Clear Flags** option to **Depth only**. This makes the background transparent so it renders ONLY the FPP layer on top of the Main Camera. In the Parent

Camera do a similar thing by turning the FPP layer off in the culling mask. If You don't have a layer for the FPP objects You must create one.

For more information on layers and camera culling masks see the Unity online documentation and unity answers forum.

Can I create more animations for the flashlight hand?
Yes. More animations will be available in the **advanced pack**, or You can make them on Your own using the Unity **Animation** panel to animate the rig.

How do I change the light properties and cookie?
In the hierarchy of the prefab there is an object named **fl_light** which is a normal Unity spotlight. While selected You may change all its properties including the Cookie to achieve the desired effect. The **fl_light** object is a parent of the flashlight object under the **flHandle** bone in the rig.

I've positioned the hand as desired, but a part of the hand seems to be transparent close to the screen. What is happening?
Every camera in Unity has a property named **Clipping Planes** with two values **Near** and **Far**. The Clipping planes set how close and how far from the screen will the objects be drawn. Setting the Near value to a lower number (like 0.15) should solve the problem and is even advised for an FPS game, to remove any other wall and object glitches in the future.

How do I change the control buttons for the hand?
See the Control customization (pt. 4) of this manual.

Can I change the on-off audio sample?
Yes. You need to reassign Your own sample in **the horrorFlashlightBasicPref** object **Audio Source** component. After doing this the sound sample will change automatically.

What are the other sounds used in the online demo? Are They available in the package?

No. The ambient windy sound is downloaded from the **freesound.org** archive, and the monster sounds are available for separate purchase on the asset store in the **Ogre growl monster audio** and **Zombie rawr voice package** asset packs with more additional sounds.

Have fun!