MANUAL

UNLOCK SYSTEM

version 1.1.5

INTRODUCTION

This documentation was created in order to show how to work with the Unlock System Asset. What kind of Asset and its features were described in the Unlock System Documentation. US consists of 90% of the code (C #). Users who know the basics of this programming language or similar languages (C, C ++) will easily understand the logic of the Asset.

Most likely, the user will find errors both lexical and code. To simplify the task to the developer, I ask you to report any errors found to me at the email address (choco.16mail@mail.ru) or on the Asset forum.

In this documentation, I will briefly describe the parameters of scripts, objects (prefabs), asset hierarchy, etc.

Contents

SCENE OBJECTS

Description

Opening the demo-scene is the first thing you see is a level with rooms (levels). Each level has its own door with the inscription (difficulty level). Consider the objects of the scene. Figure 1 shows the hierarchy of objects in the scene.

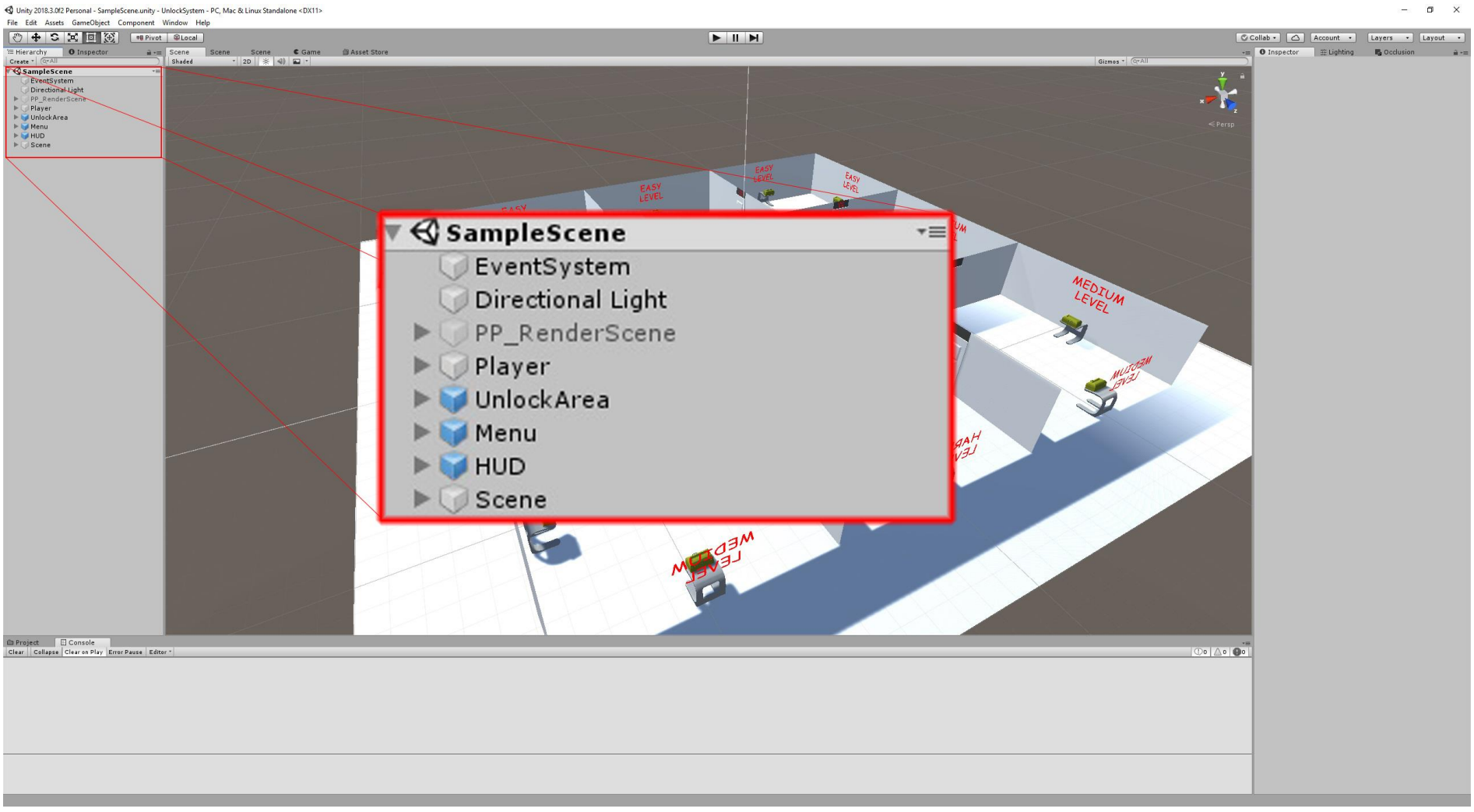


Figure 1

1. *PP_RenderScene* – this is a scene of rendering pictures into texture (render in the texture). The render of the scanning system takes place on it (Figure 2). It is activated when we press the scan button (the big red button);

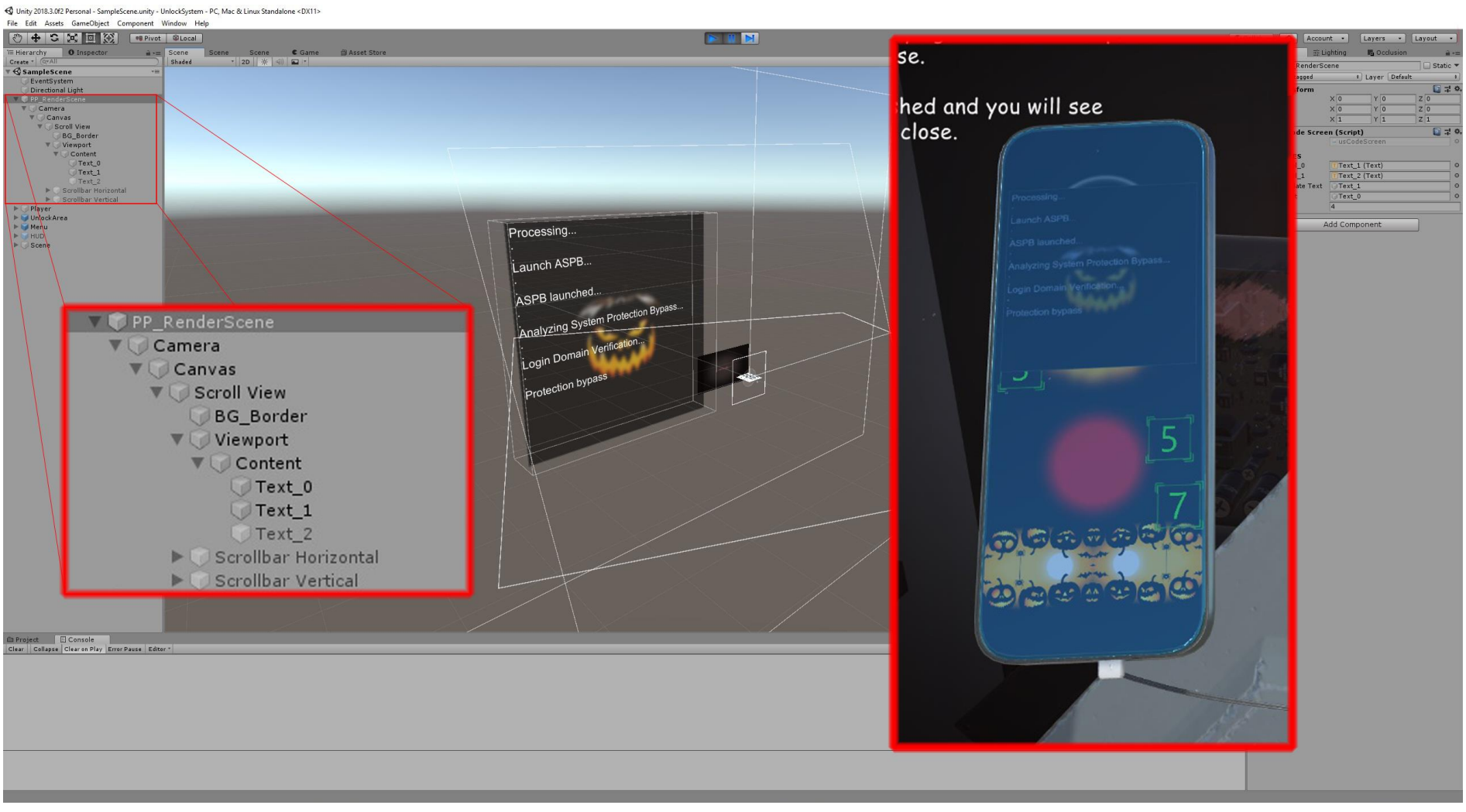


Figure 2

The *PP_RenderScene* object has a ***UsCodeScreen*** script, which has the parameters shown in Figure 3. The first 4 parameters are a link to the text objects that will scroll on the screen during the scan. *SpeedRoll* (or *Scroll*) is the speed of text scrolling, experiment with this parameter to see the result.

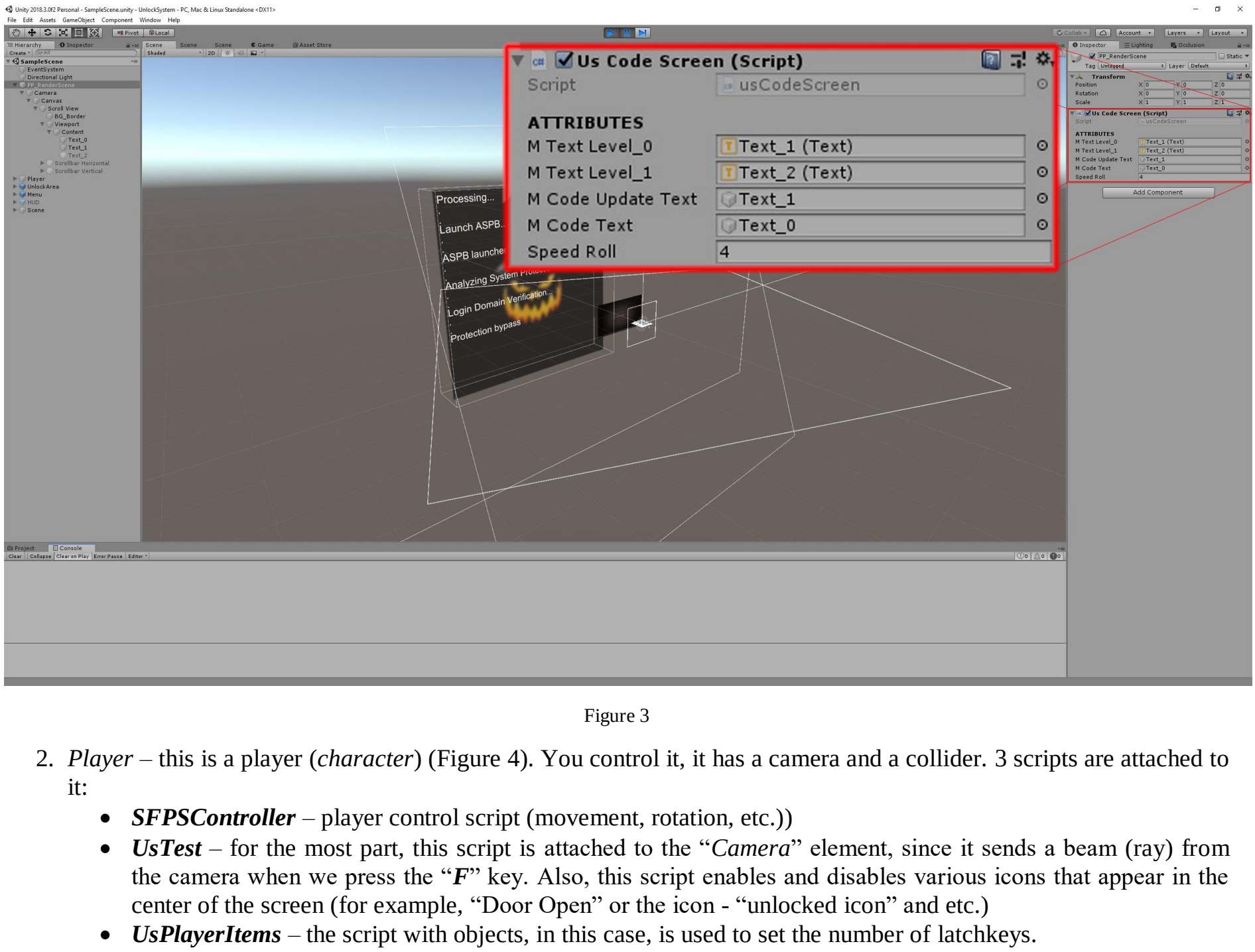


Figure 3

2. *Player* – this is a player (*character*) (Figure 4). You control it, it has a camera and a collider. 3 scripts are attached to it:
    - ***SFPSController*** – player control script (movement, rotation, etc.))
    - ***UsTest*** – for the most part, this script is attached to the "*Camera*" element, since it sends a beam (ray) from the camera when we press the "***F***" key. Also, this script enables and disables various icons that appear in the center of the screen (for example, "Door Open" or the icon - "unlocked icon" and etc.)
    - ***UsPlayerItems*** – the script with objects, in this case, is used to set the number of latchkeys.

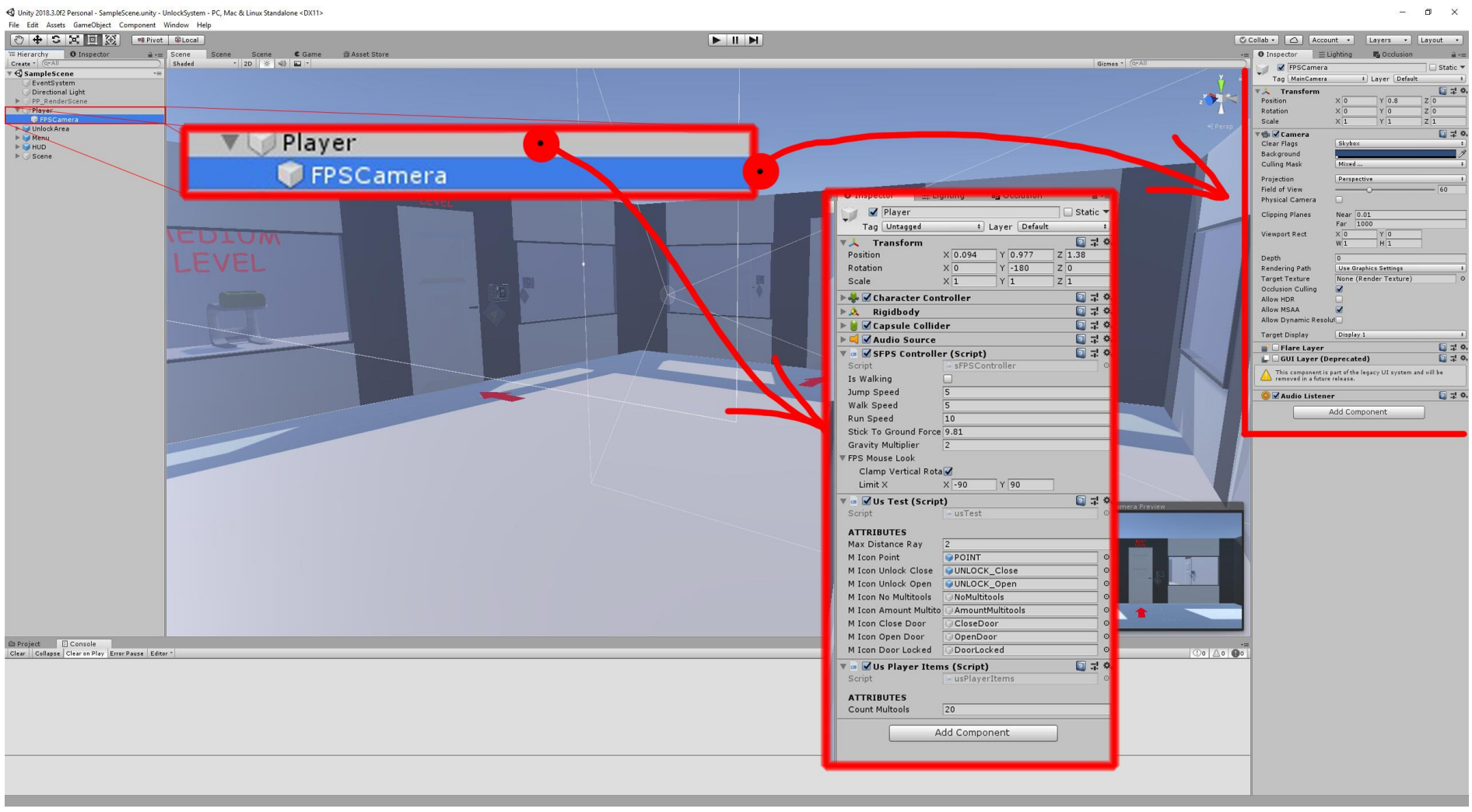


Figure 4

3. *UnlockArea* – system interface (mechanical and electronic). All events take place in this place. By default, it is inactive, when we approach to the door and starting the interface, the *UnlockArea* is activated and, depending on the selected interface, either a mechanical (*UnlockPoint_0*) or an electronic (*UnlockPoint_1*) interface appears on the screen.

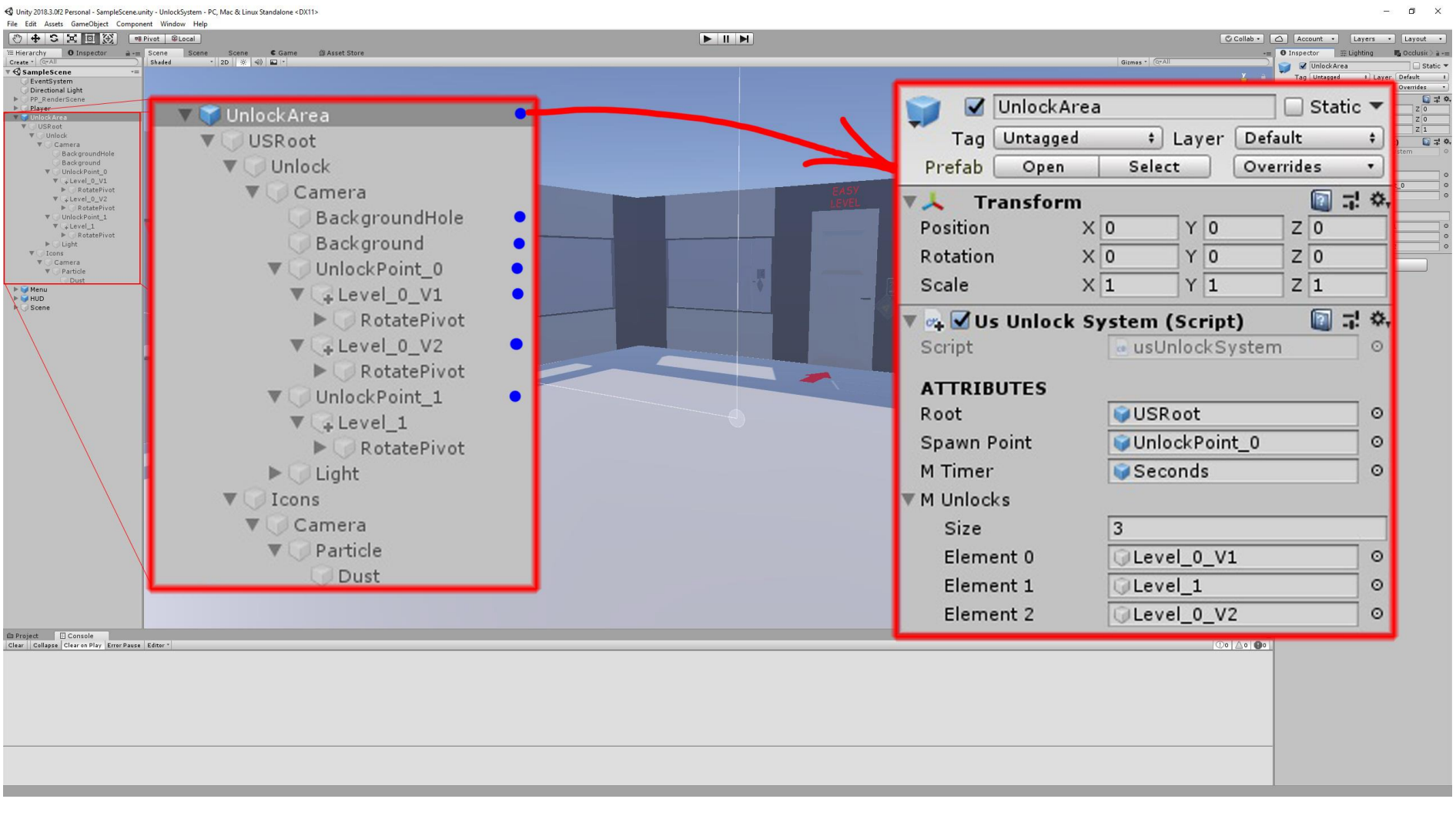
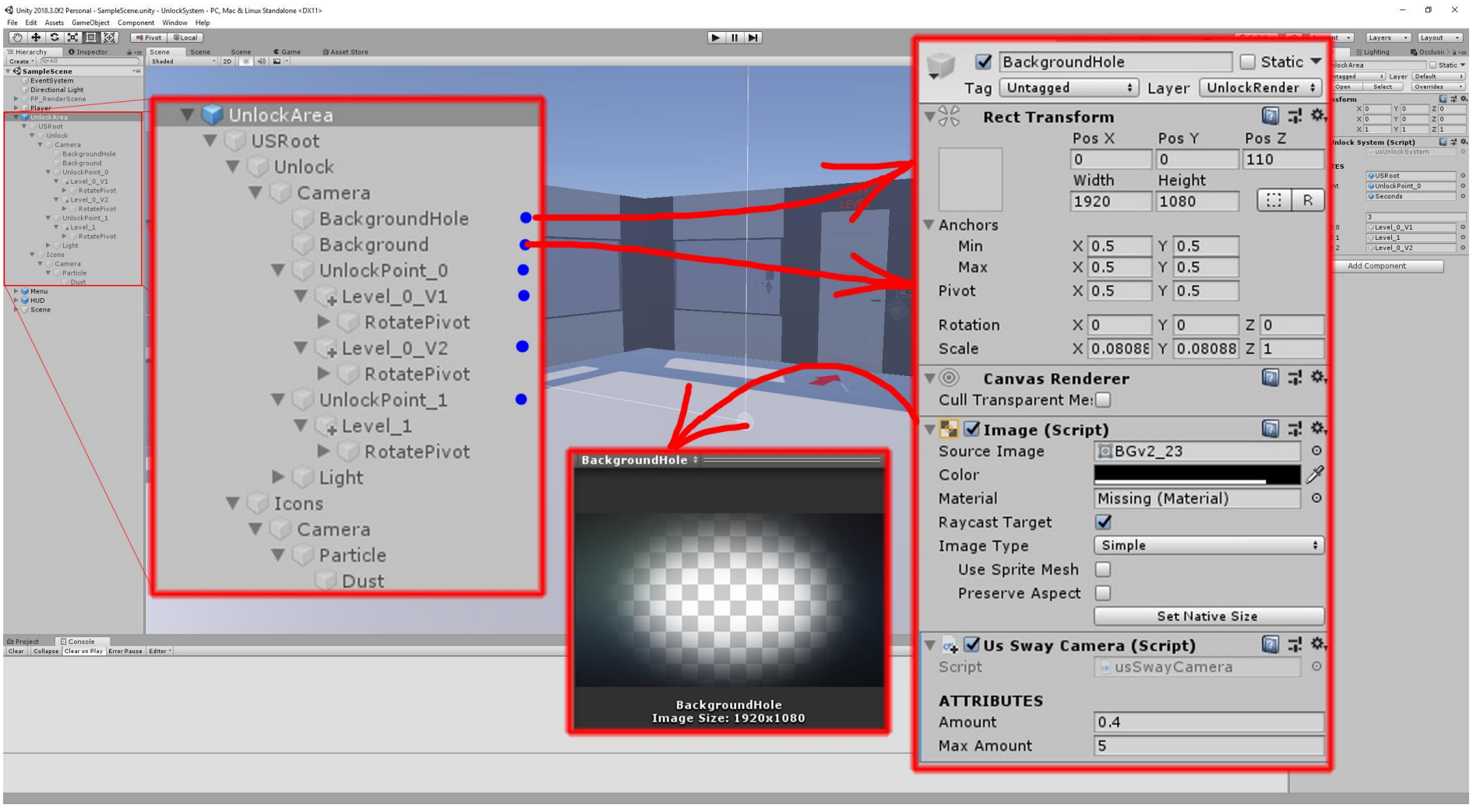


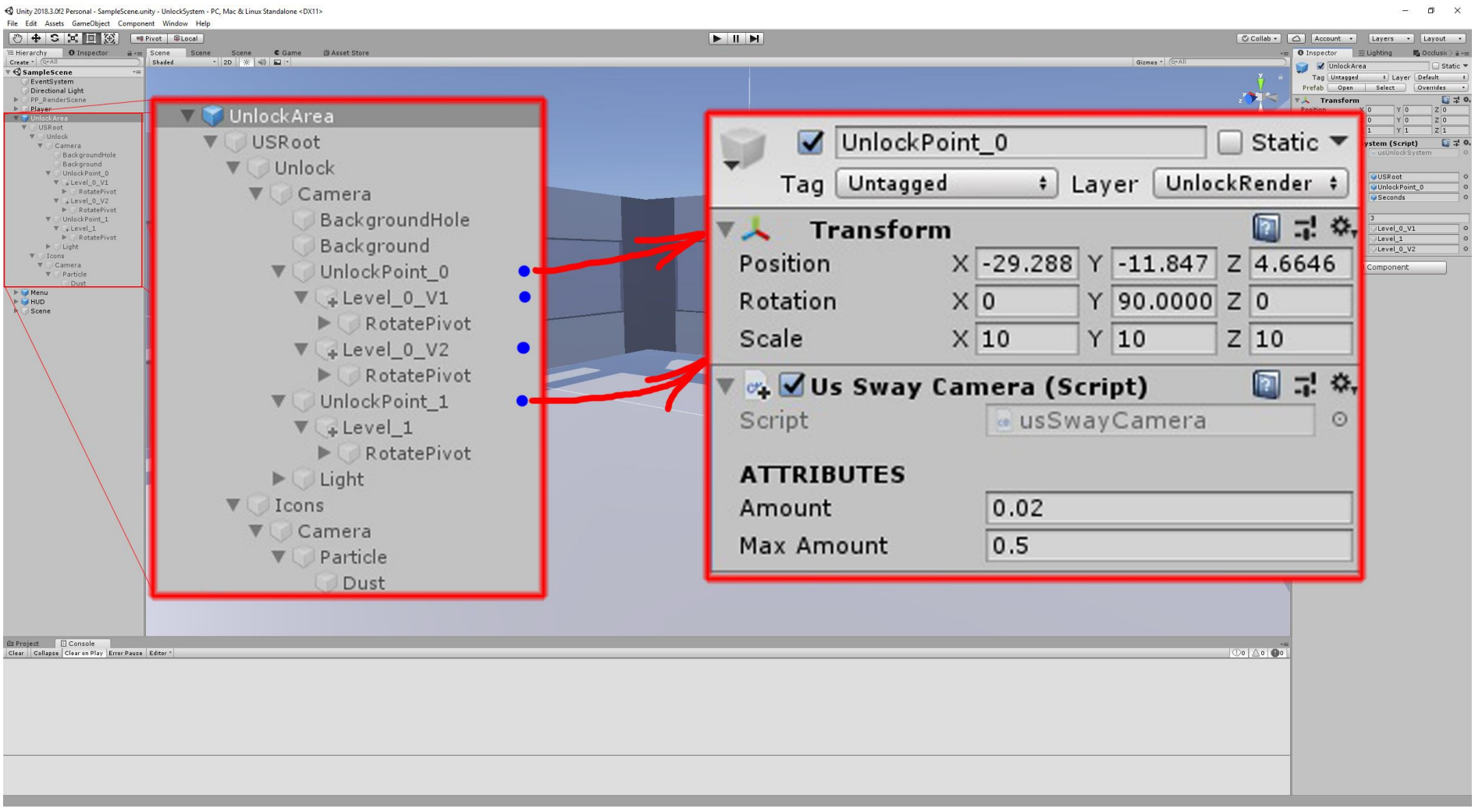
Figure 5

The *UnlockArea* object has a ***UsUnlockSystem*** script (Figure 5). The script that transfers to the interfaces all the parameters necessary for working with it. The *Root* parameter is a link to the root of the object, since it is not active, we will need to enable it when starting the interface. The *SpawnPoint* and *MTimer* parameters are not involved at this stage, they will be deleted later. The *MUnlocks* parameter is an array of "*levels*"; if you want to add your own version of the lock, then you should add it to this array.



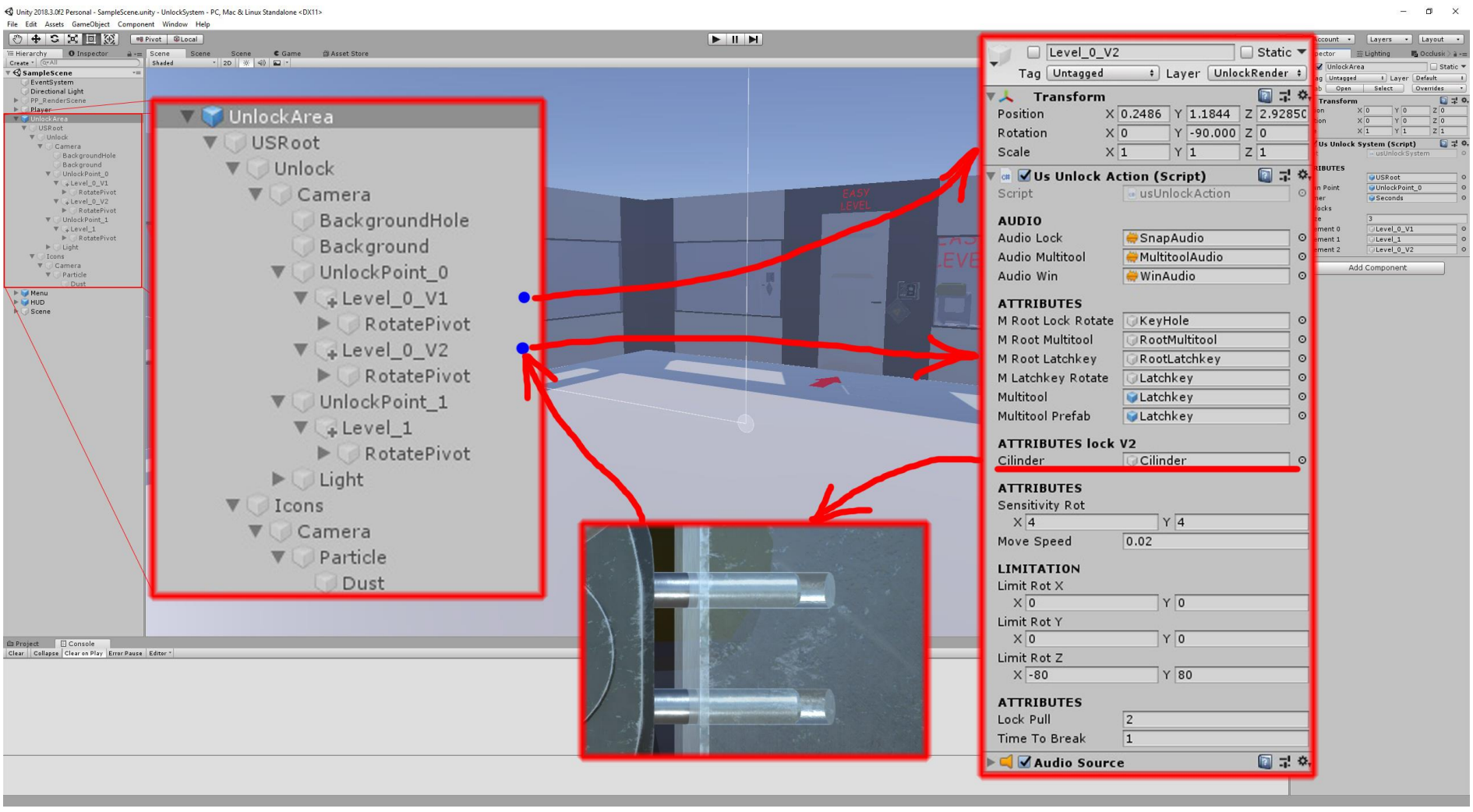
Fegure 6

BackgroundHole and *Background* objects play the role of the background. To give greater dynamics to the picture, they have a ***UsSwayCamera*** script, which gives them the effect of moving with the mouse. The only difference is that they have different textures.



Fegure 7

UnlockPoint_0 and *UnlockPoint_1* objects are, as already mentioned, system interfaces. Attached to them is the ***UsSwayCamera*** script in order to add dynamics.



Fegure 8

The *Level_0_V1* and *Level_0_V2* objects are two variants of the lock, add another interface variant here and add to the *MUnlocks* array (see above). Objects have a ***UsUlockAction*** script, this script only applies to the mechanical interface of the system. The first parameters are audio accompaniment. When we rotate the latchkey, the sound is played. The region "*ATTRIBUTES*" is a link to various objects, such as a latchkey, multitool, keyhole (its model) and so on. The next region "*ATTRIBUTES lock v2*" refers only to the second lock variant. These are certain elements (Figure 8) that will be unlocked when we win, imitating that the door is open. The next region "*ATTRIBUTES*" - the *SensitivityRot* parameter - rotation sensitivity by the latchkey (speed), the *MoveSpeed* parameter is the rotation speed of the multitool. The region "*LIMITATION*" is the region of rotation restrictions of the latchkey and the multitool. The region "*ATTRIBUTES*" - the *LockPull* parameter - is the maximum deviation of the vibration of the latchkey, the parameter *TimeToBreak* - is the time after which the latchkey will break.

The *Icons* object is a particle system. Which appears in the background of the interfaces.

Mechanical interface objects are not so interesting, because there are no independent elements in them. Let us consider in more detail the objects of the electronic interface.

The internal *EL_MeshAssembler* object has a ***UsLightSwitcher*** script that switches between bulbs once per frame (the *fps* = 1 parameter).

Figure 9

Opened this assembly and going down to the bottom, we will find 3 elements (Figure 10), we will talk about them.

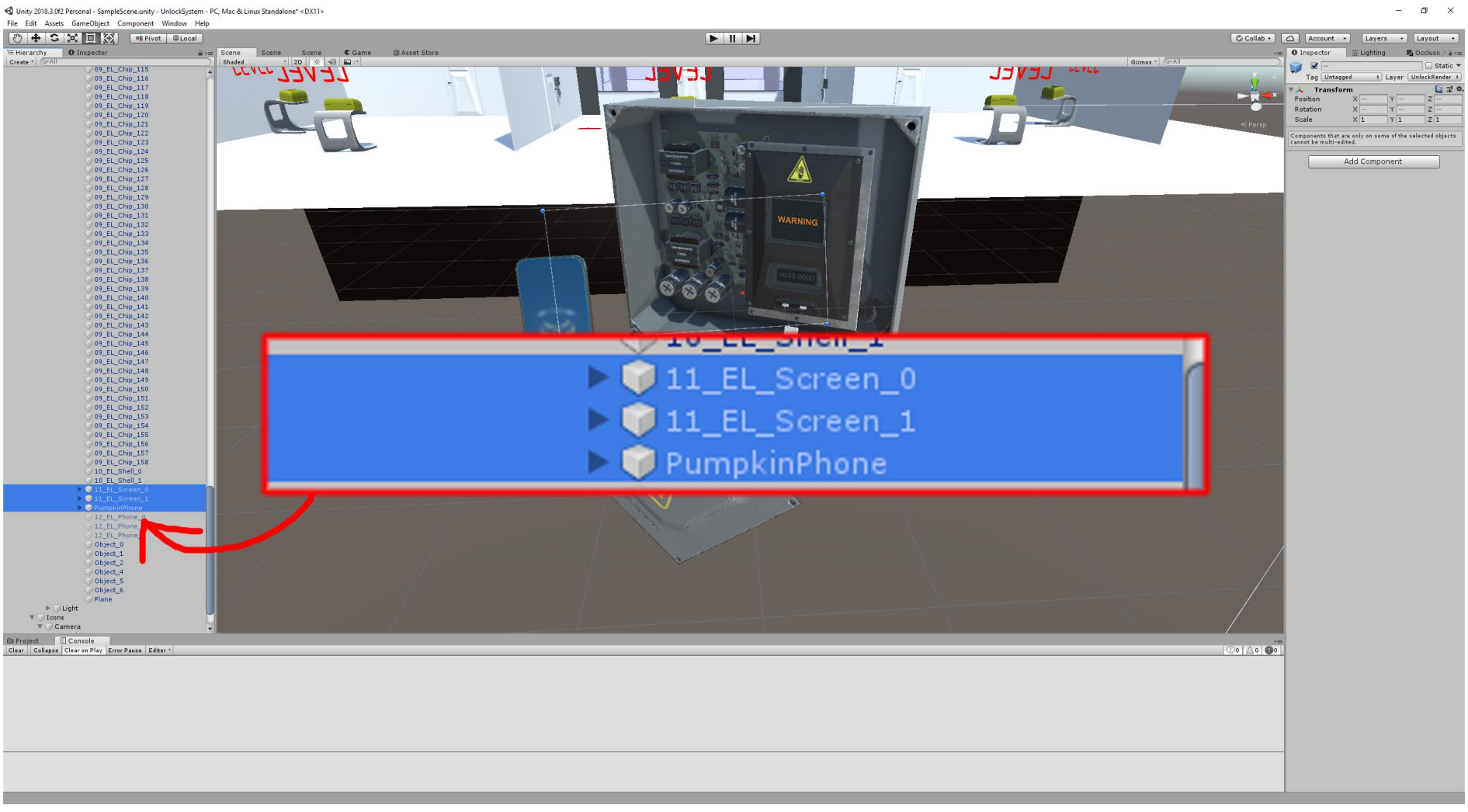


Figure 10

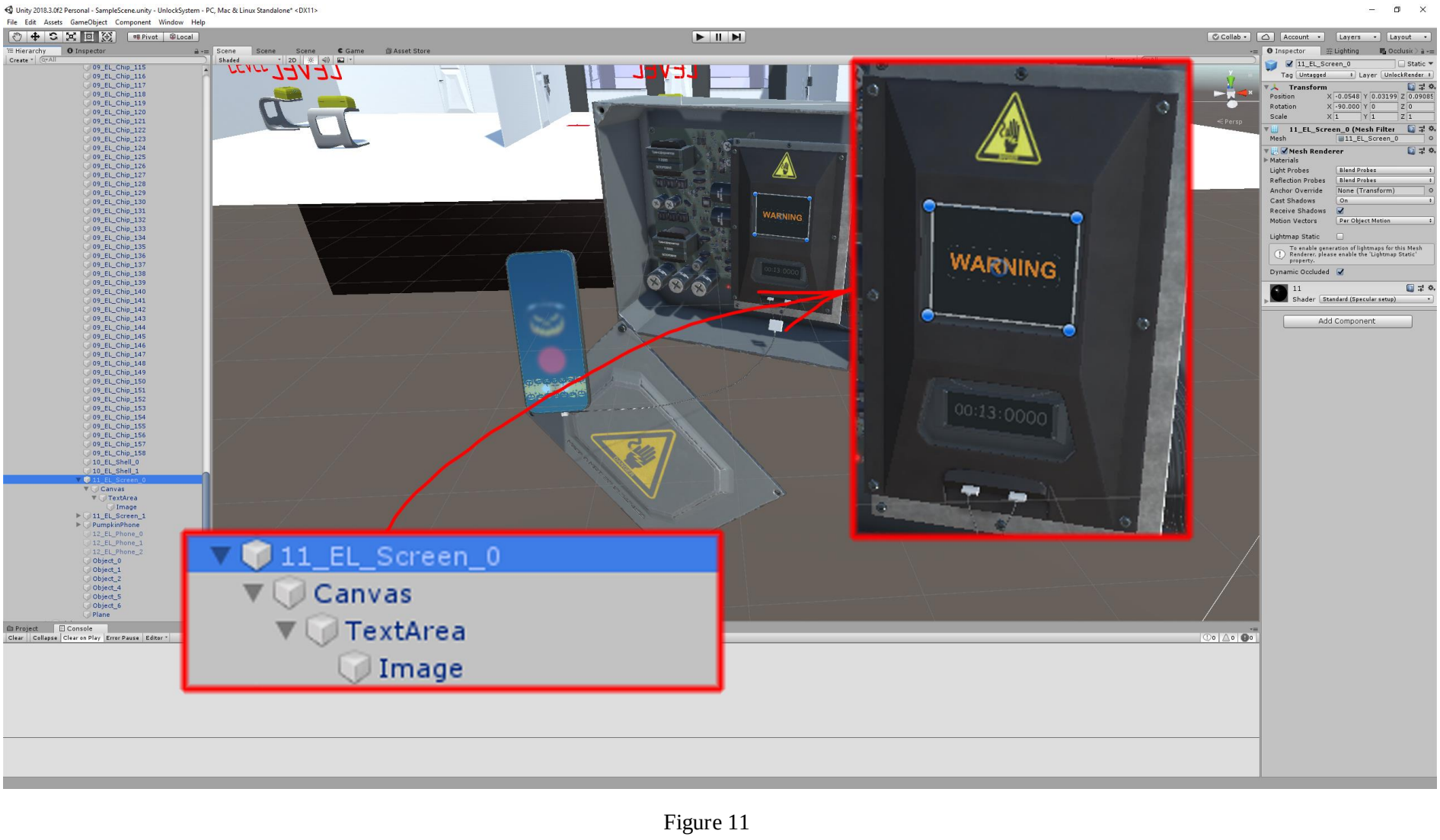


Figure 11

As I wrote earlier, one of the screens of the electronic box is free, the object *11_EL_Screen_0* is a free screen that has only one texture.

Figure 12 shows the second screen with a timer, it has two *Seconds* and *MilliSeconds* objects, this is our time, the ***UsSearchLootIcon*** script is attached to them, the script that animates the texture, it is not complicated, study it, and you will understand, how it works.

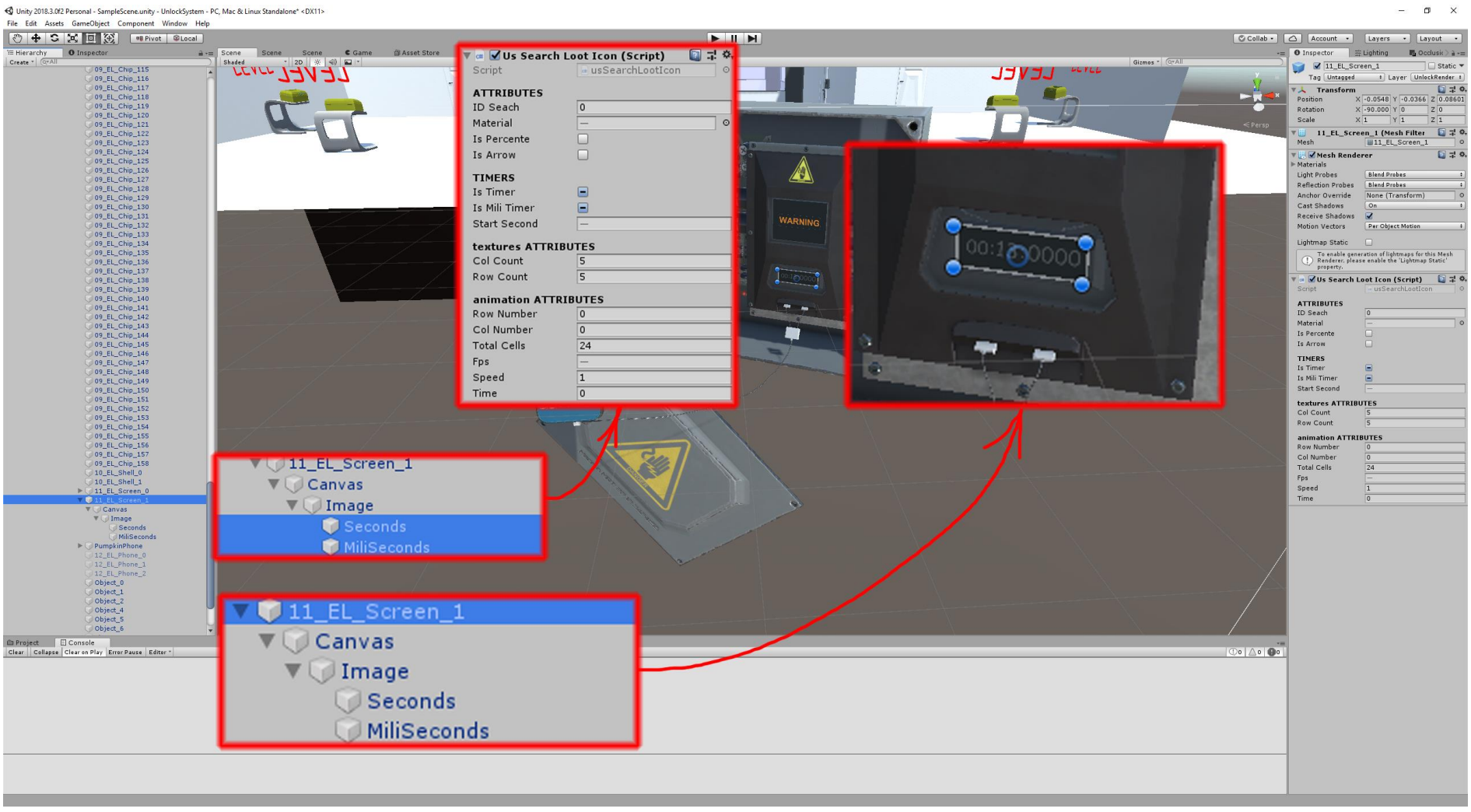


Figure 12

Well, the most interesting is the phone "*PumpkinPhone*". The *PumpkinPhone* object (Figure 13) is an interactive object, and the player interacts with it. The main script of the ***UsPhoneScript*** electronic interface is attached to this object. Which performs most of the functions of this interface.

The region "*ATTRIBUTES*" is skipped because it is clear, it is just a link to objects. The "*ICONS*" region is a link to a timer, since he will need to pass parameters (such as start time). Region "*CODE BUTTON*" - this region refers to our buttons on the screen (code buttons). The *CodeBox* parameter refers to the object where the buttons will appear. The *CodeBoxPrefab* parameter is the prefab of our button, which we will copy, depending on the level (the number of buttons depends on the level of complexity). The *MCodeBoxes* parameter is sprites (textures with numbers - figure 14). The *PositionCodeBoxInScene* parameter is an array of all possible button positions on the screen. You can make it random, but then you will need to write conditions for overlapping and limiting the spawn zone of the buttons on the screen.

Figure 13

Objects *12_EL_Phone_0* and *12_EL_Phone_003* are models of the case and phone glass. The *object 12_EL_Phone_004* is a canvas on which all actions take place.

Figure 14

Consider the *Button* object (Figure 15); this is the big red button. Attached to her script ***UsButtonScan*** responsible for the pulsation of the button. The button also has an action; when you click on it, the scanning system starts (see below).

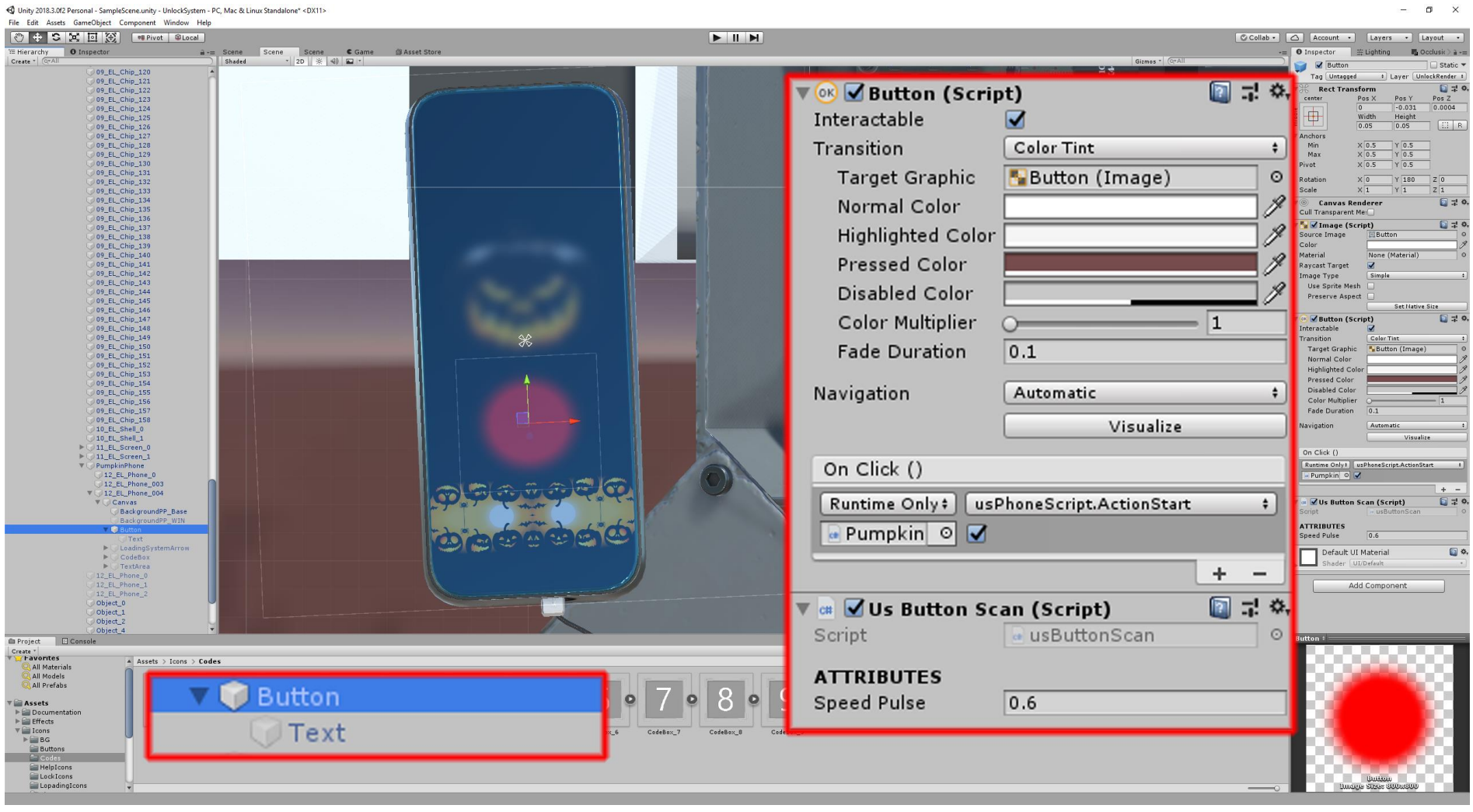


Figure 15

When the button is pressed, the scanning system is launched, a window appears (Figure 16), onto which a picture is projected from the scene of the *PP_RenderScene* render and the effect of scrolling through the code is created.

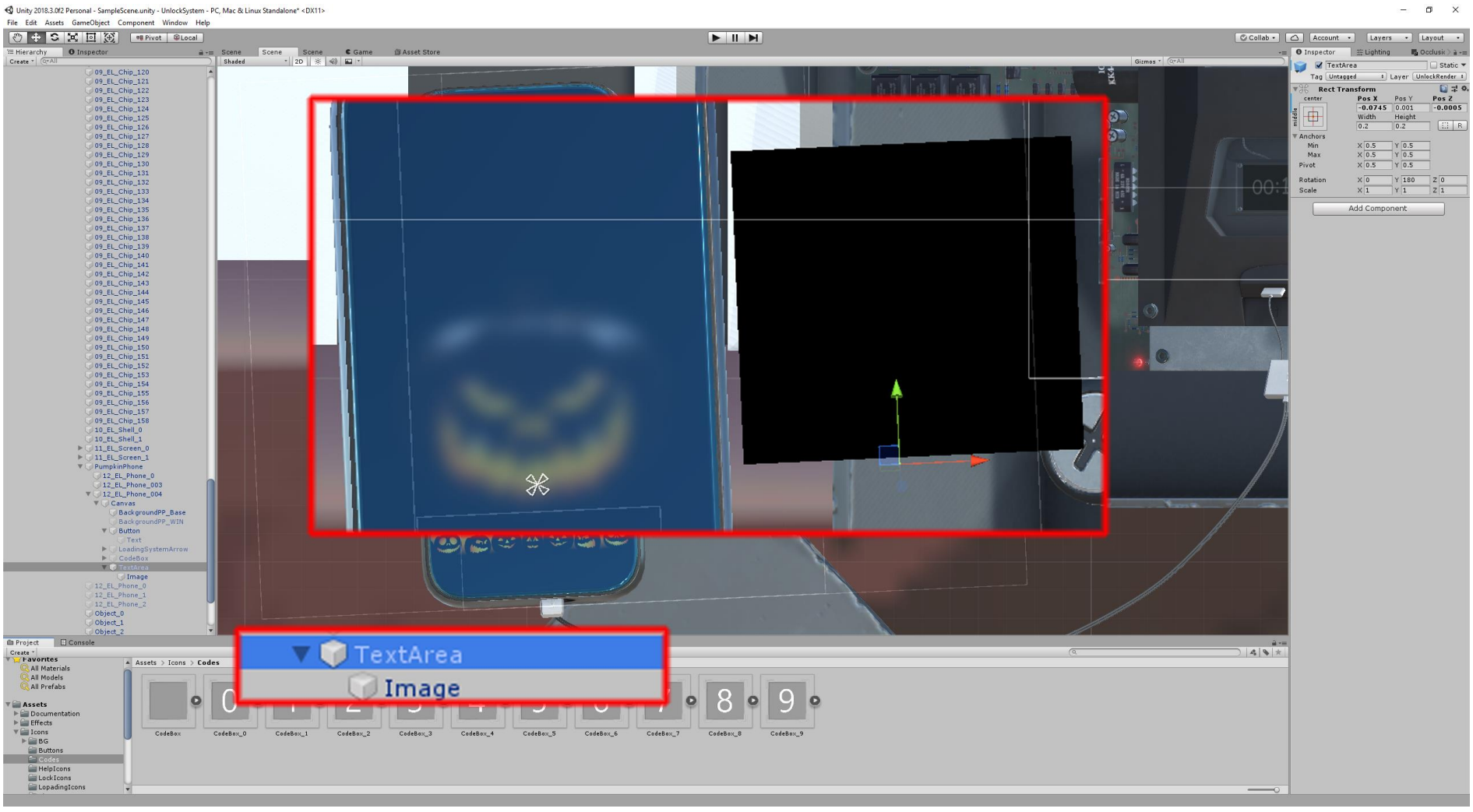


Figure 16

4. *Menu* – this is a menu object (Figure 17), containing items such as tooltips.

Figure 17

5. *HUD* – this is an interface (Figure 18) that periodically appears in the center of the screen (depending on the player's actions).

Figure 18

6. *Scene* – this object contains all the scene models (Figure 19). Here we will be interested in 2 objects - *LootBox* and *DoorPrefab*.

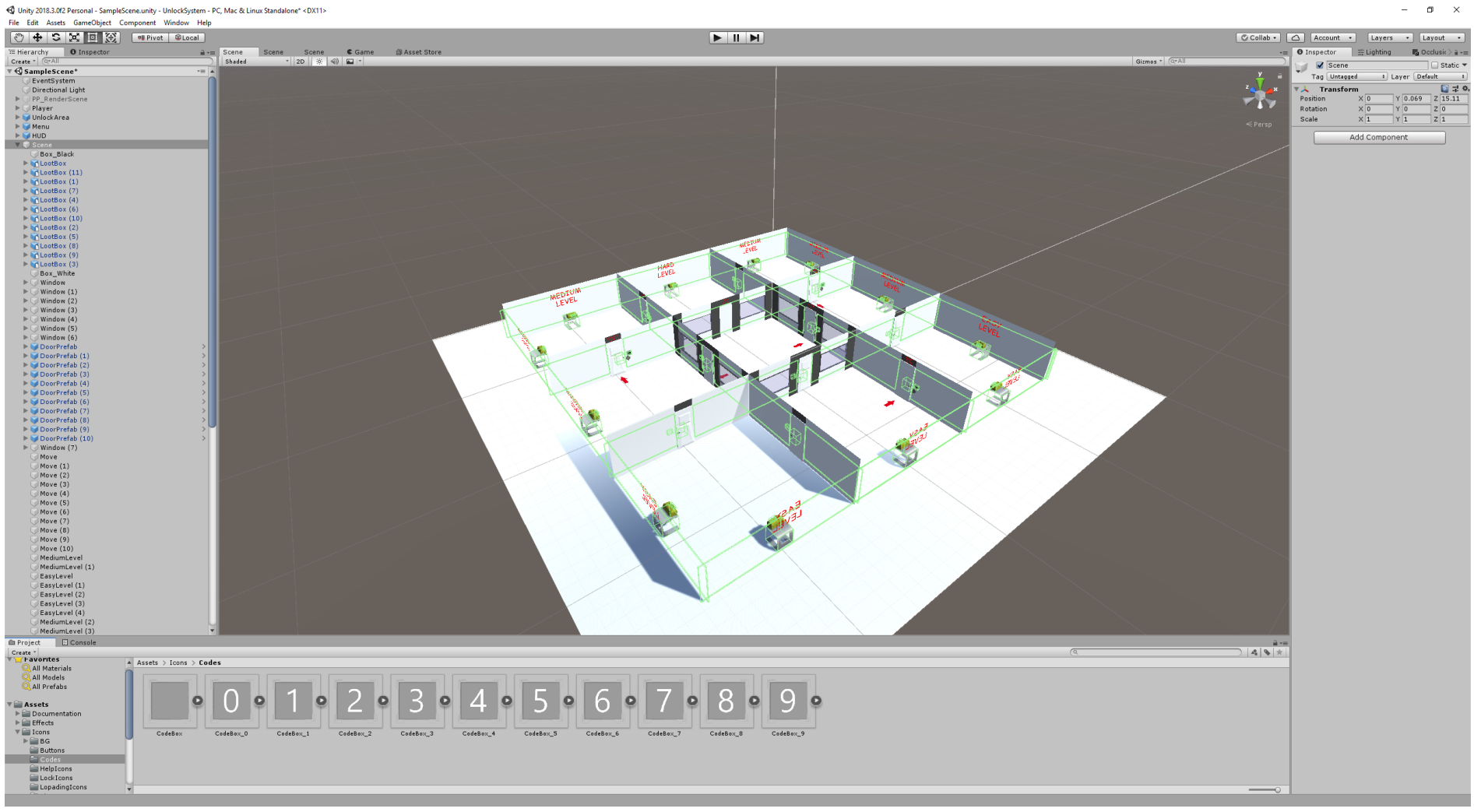


Figure 19

Consider these two objects in more detail.

The *LootBox* object (Figure 20) has one very important object in it - *LockBox*, it has a *Collider* component and the *LockBase* tag is assigned to it, which means that the object belongs to a mechanical interface. An ***UsLootBox*** script is attached to the object, which passes parameters to the interface. The *UsLockOfDifficulty* parameter is a difficult level, the *UsLockLevel* parameter is a lock version (interface, do not forget that by assigning the *LockBase* tag to an object, select the appropriate levels specific to the mechanical interface).

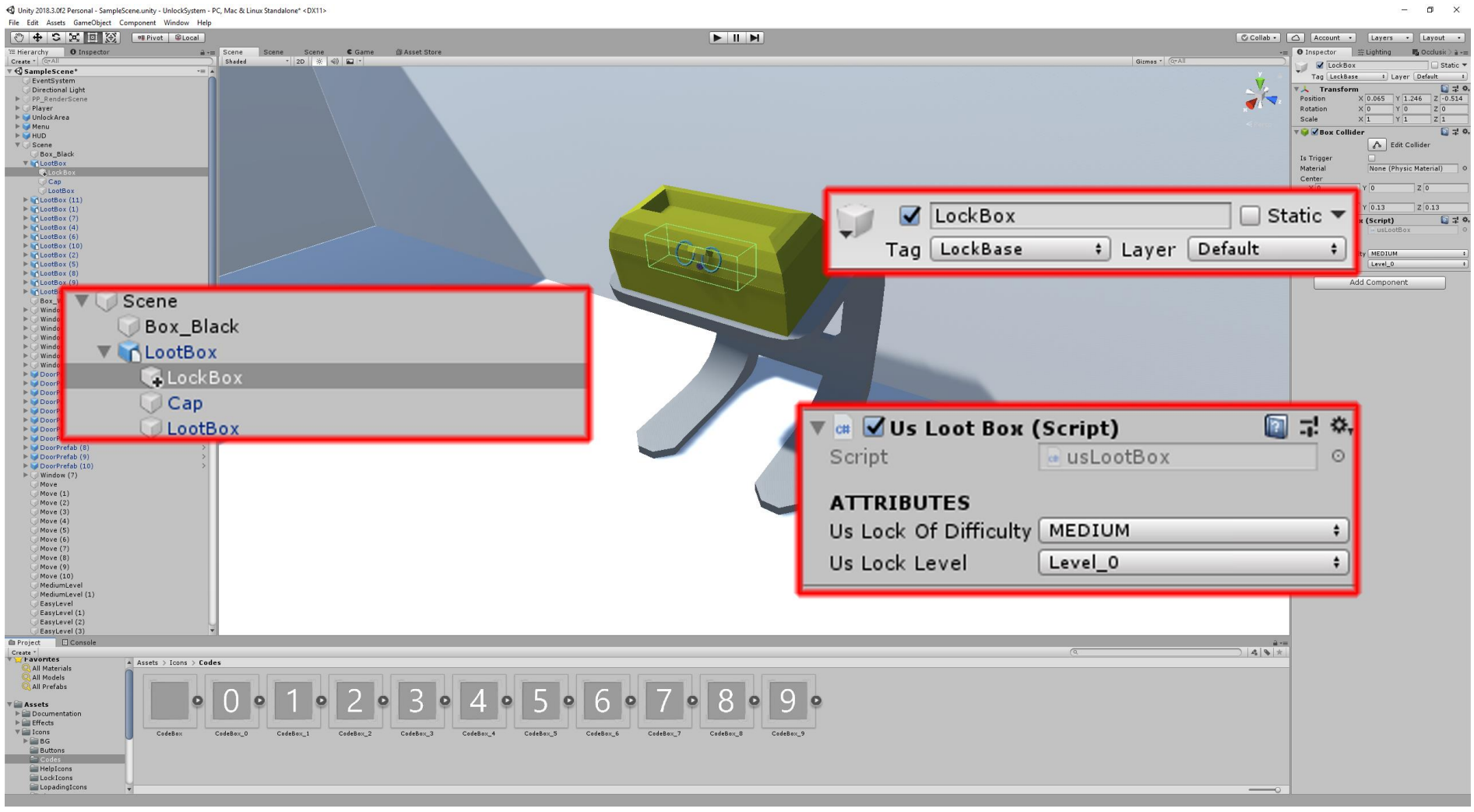


Figure 20

The *DoorPrefab* object (Figure 21) has one very important object in it - *EL_Box*, it has a *Collider* component and an *EL_Base* tag is assigned to it - this means that the object belongs to the electronic interface. An ***UsUnlockEL*** script is attached to the object, which passes parameters to the interface. The *UsLockOfDifficulty* parameter is a difficult level, the *UsLockLevel* parameter is a variant of the lock (in this case it is one).

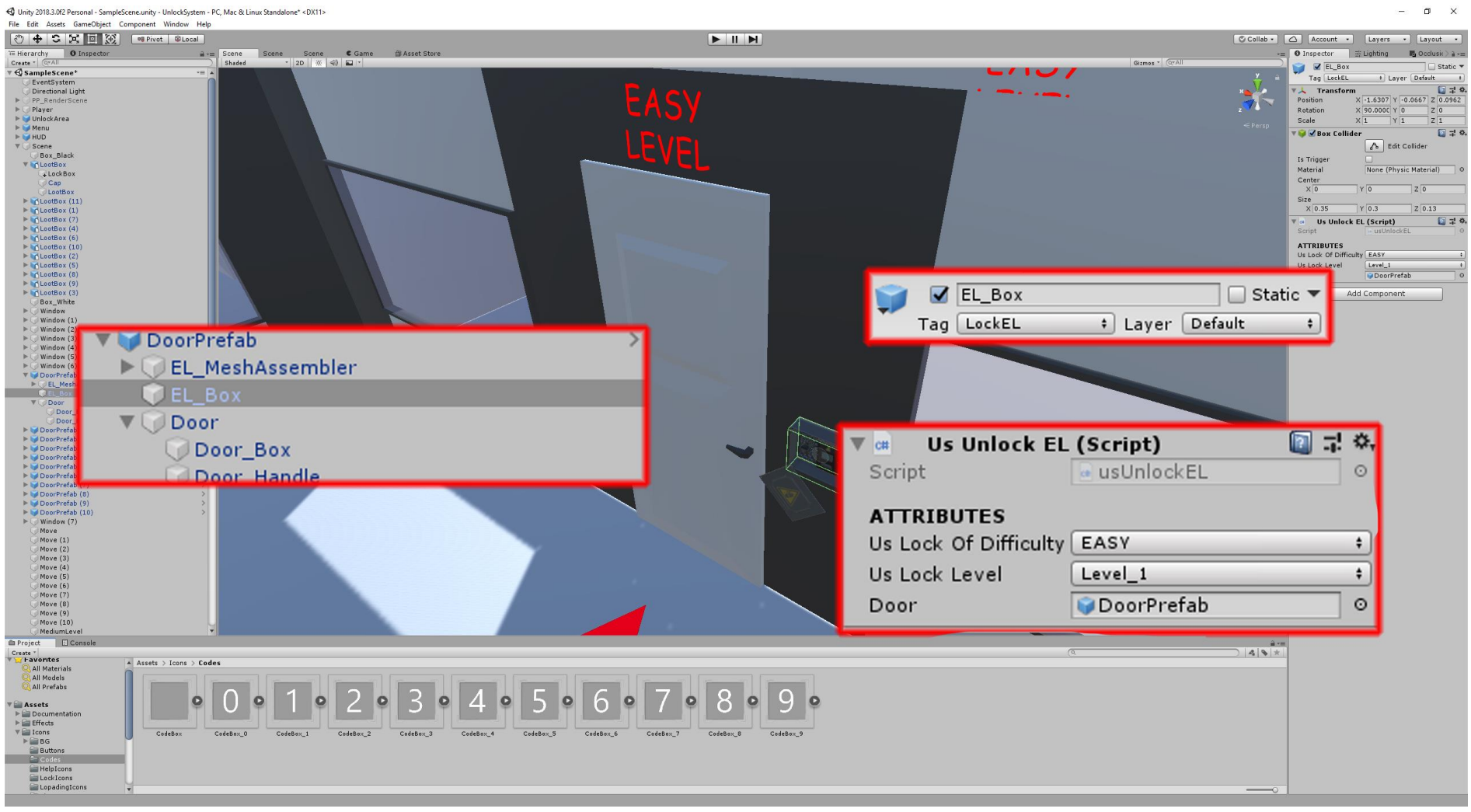


Figure 21

UPDATE 1.1.5

In update 1.1.5, a new interface was added, read more in the Documentation.

Door prefab has been added for the new interface. Also, a new “CodeLock” tag has been added for the new interface.

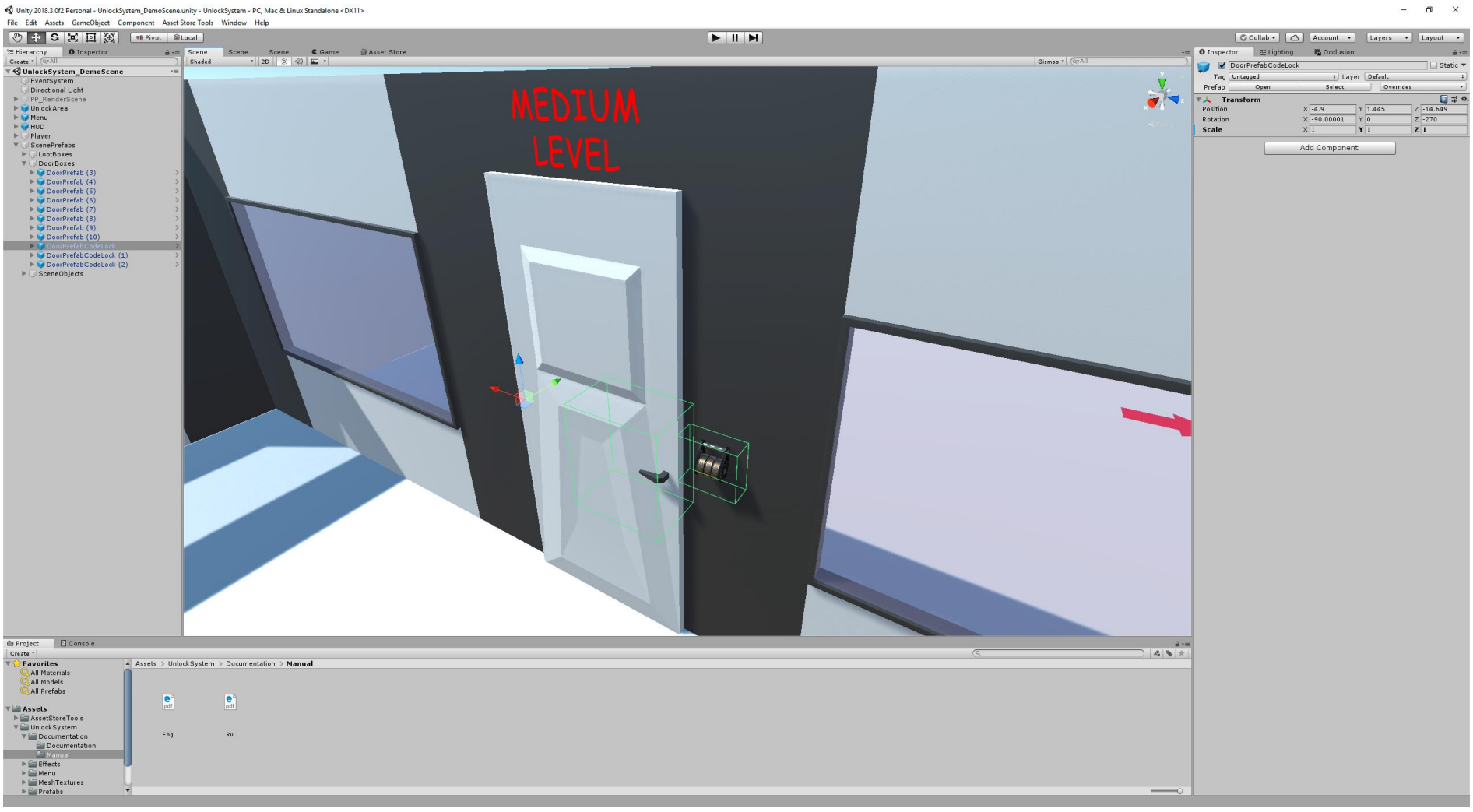


Figure 22

In the US_LootBox script, 2 new parameters were added to work with the new interface.

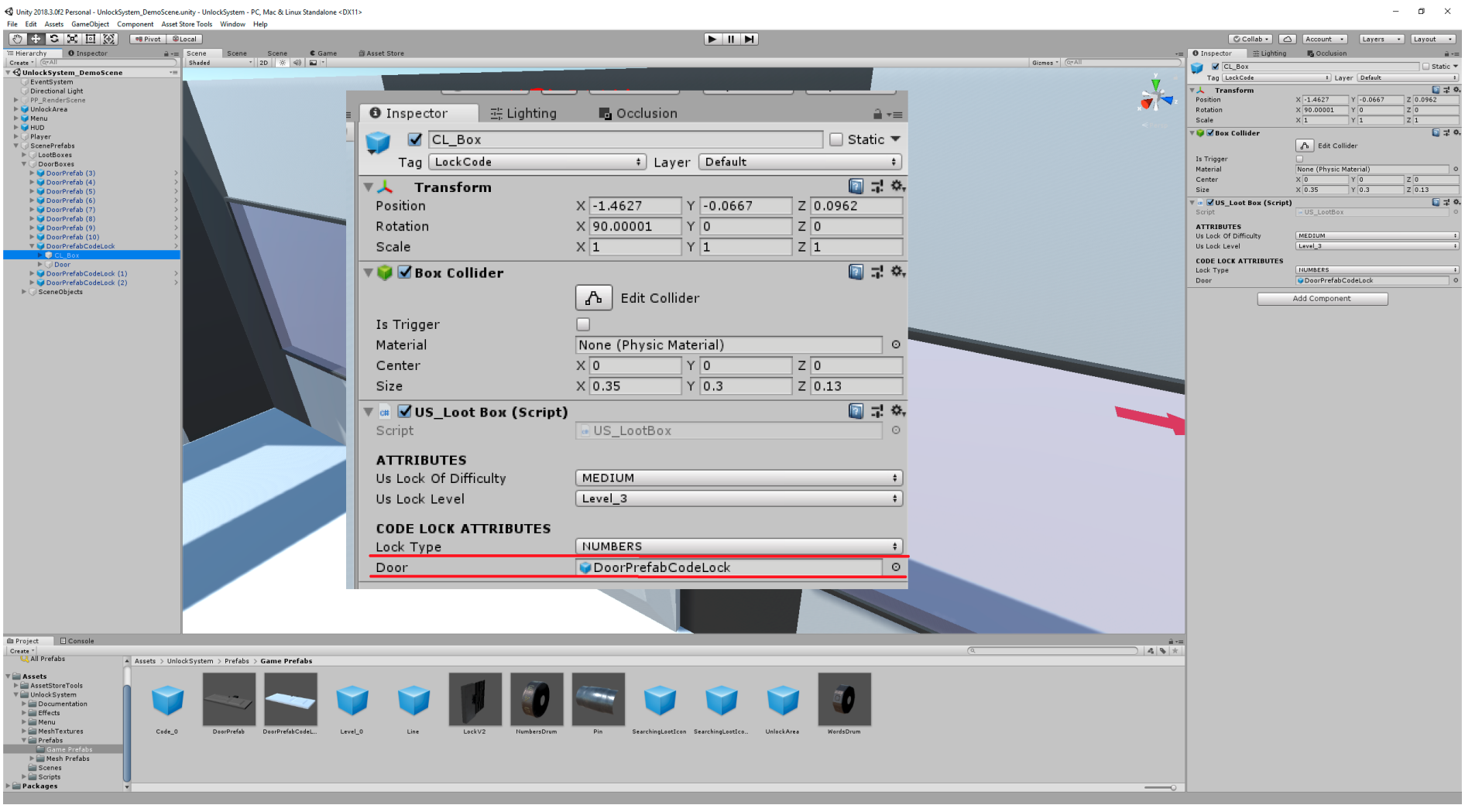


Figure 23

The interface consists only of housing parts and dummies. All other parts (drums, pins) will be created automatically and will depend on the difficulty level.

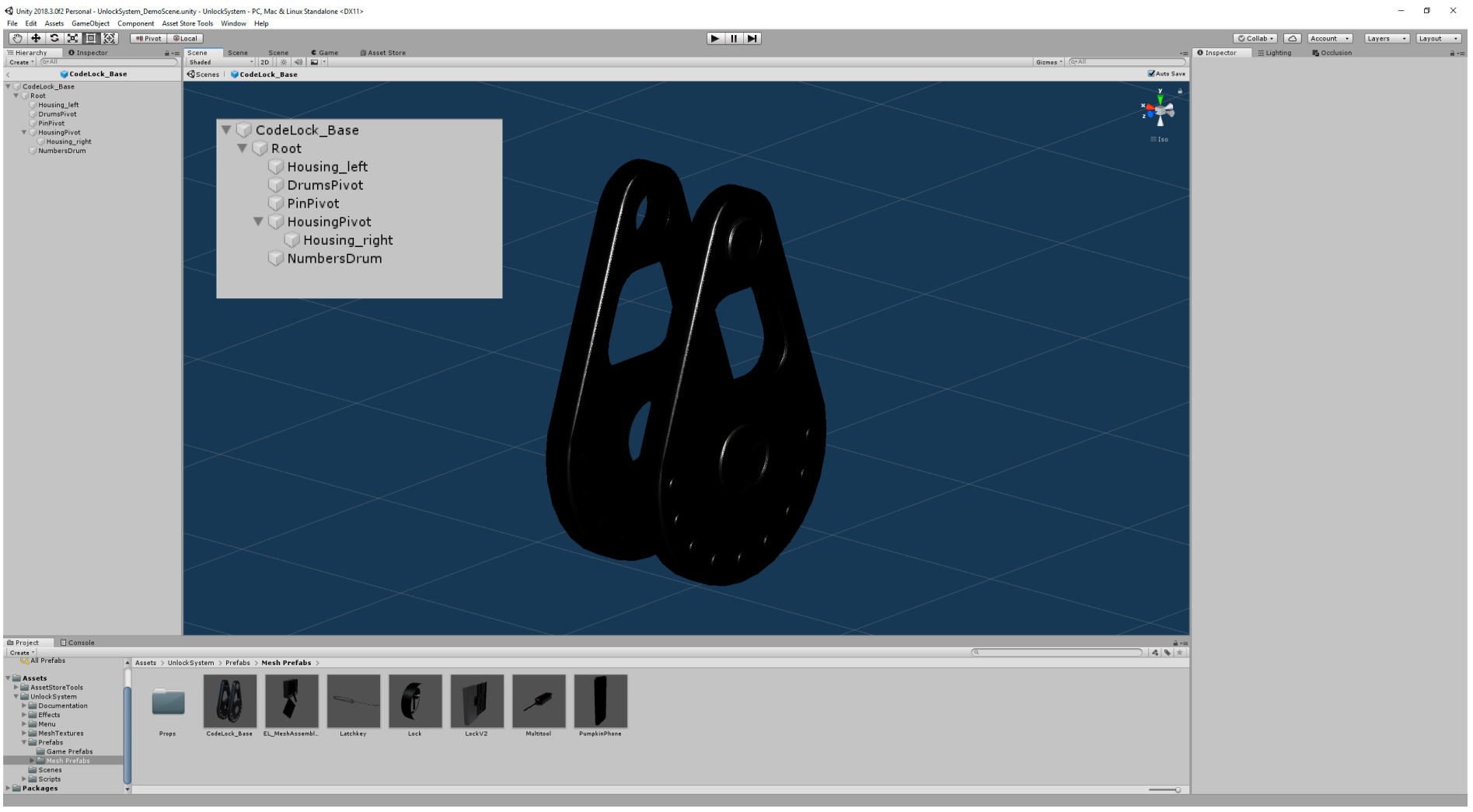


Figure 24

The drum includes two buttons. The buttons rotate the drum around axis.

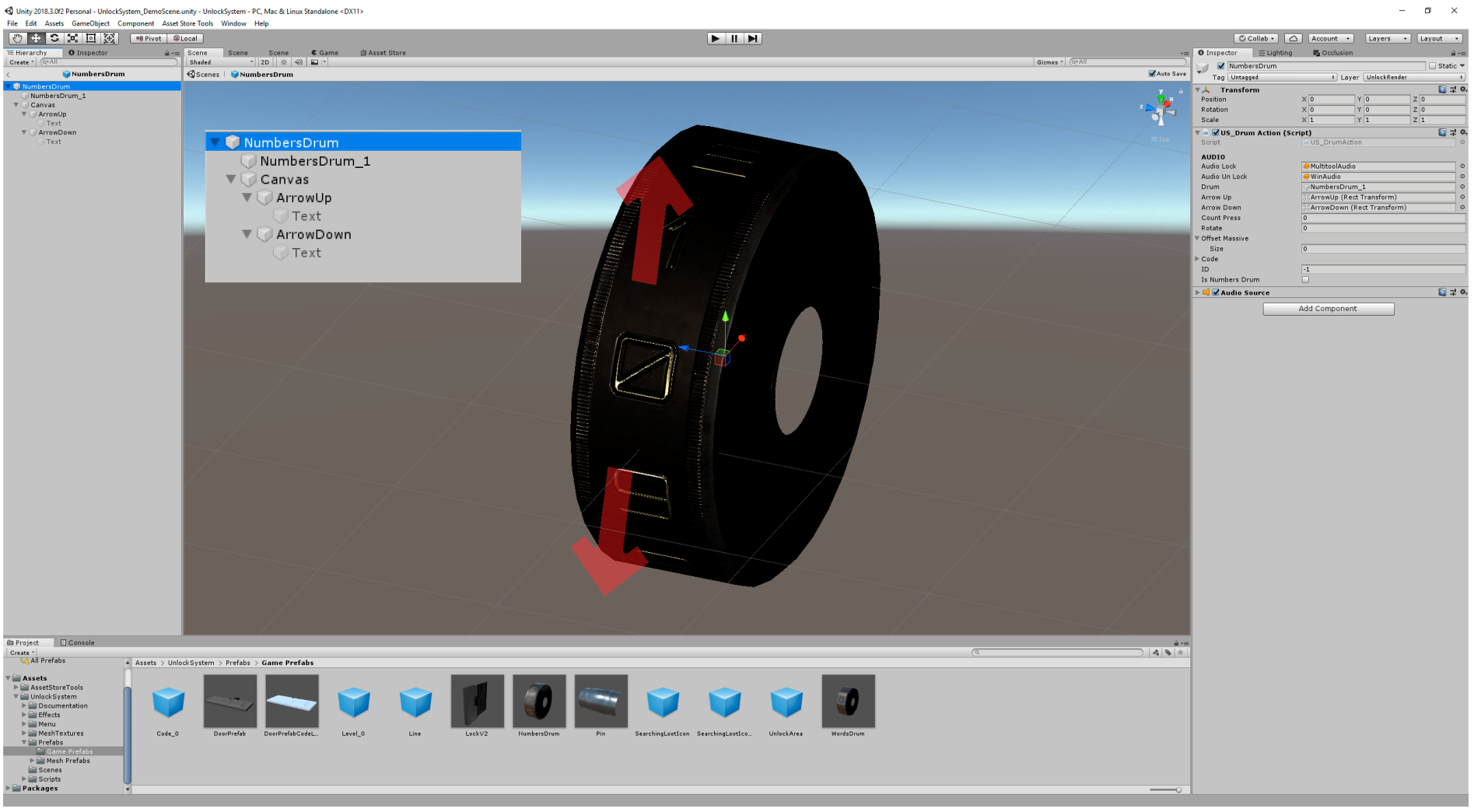


Figure 25