

ИНСТРУКЦИЯ

СИСТЕМА ВЗЛОМА (UNLOCK SYSTEM)
версия 1.1.5

ВВЕДЕНИЕ

Данная документация создана в целях того, чтобы показать, как работать с активом Unlock System. Что это за актив и его особенности были описаны в Документации по Unlock System. US состоит на 90% из кода (C#). Пользователям, которые знают азы этого языка программирования или подобных языков (C, C++), будет легко разобраться с логикой актива.

Вероятней всего, что пользователь будет находить ошибки как лексические, так и кодовые. Чтобы упростить задачу разработчику (мне) я прошу Вас сообщить о найденных ошибках мне по электронному адресу (choco.16mail@mail.ru) или на форуме актива.

В этой документации я кратко опишу параметры скриптов, объекты (префабы), иерархию актива и т.д.

Оглавление

ВВЕДЕНИЕ.....	2
ОБЪЕКТЫ СЦЕНЫ.....	4
Описание.....	4

ОБЪЕКТЫ СЦЕНЫ

Описание

Открывая демо-уровень первое, что вы увидите это сцену с комнатами (уровнями). Каждый уровень имеет свою дверь с надписью (уровень сложности). Рассмотрим объекты сцены. На рисунке 1 показана иерархия объектов в сцене.

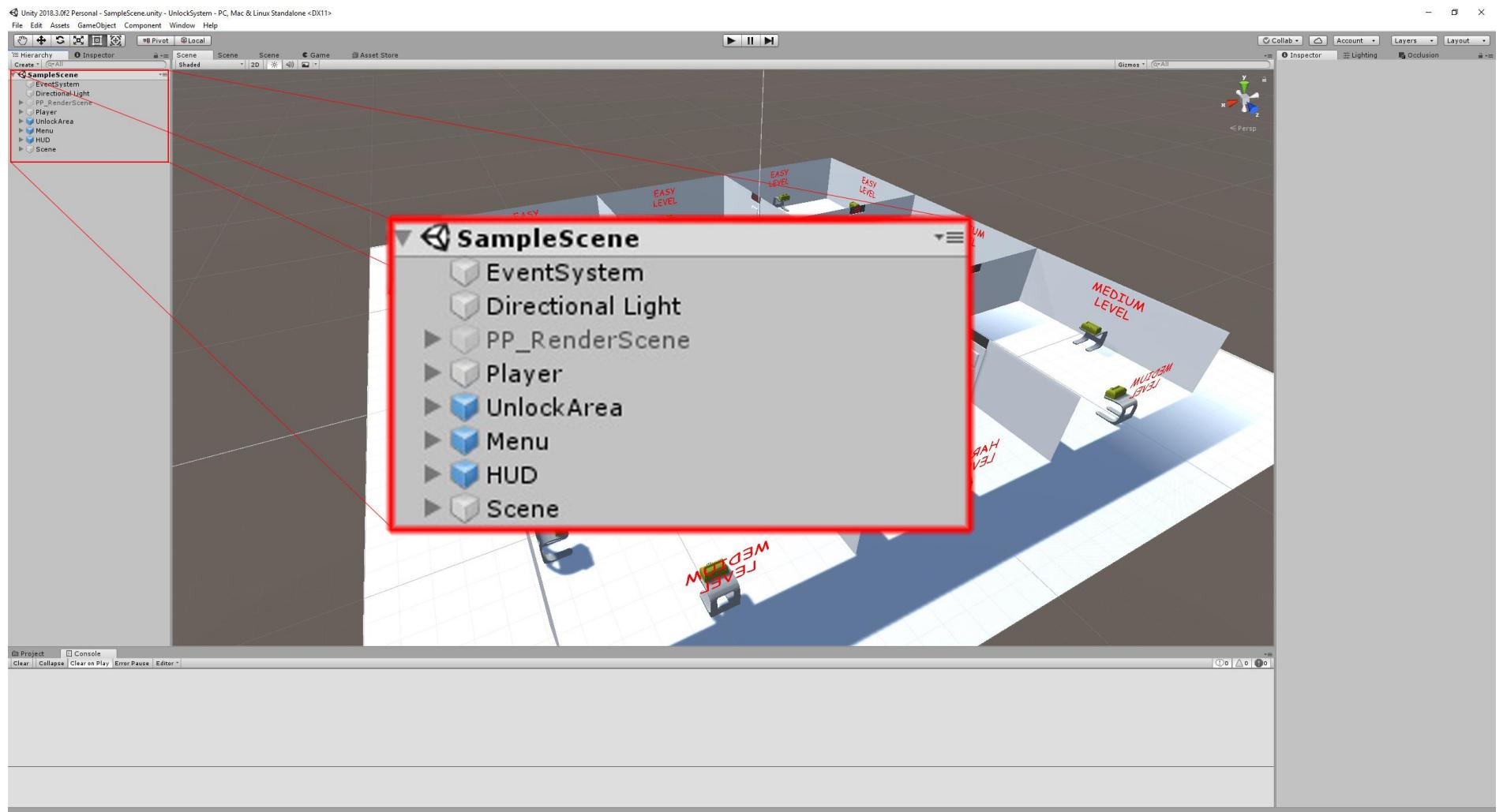


Рисунок 1

1. *PP_RenderScene* – это сцена рендера картинки в текстуру. На ней происходит рендер системы сканирования (рисунок 2). Она активируется, когда мы нажимаем кнопку сканирования (большую красную кнопку);

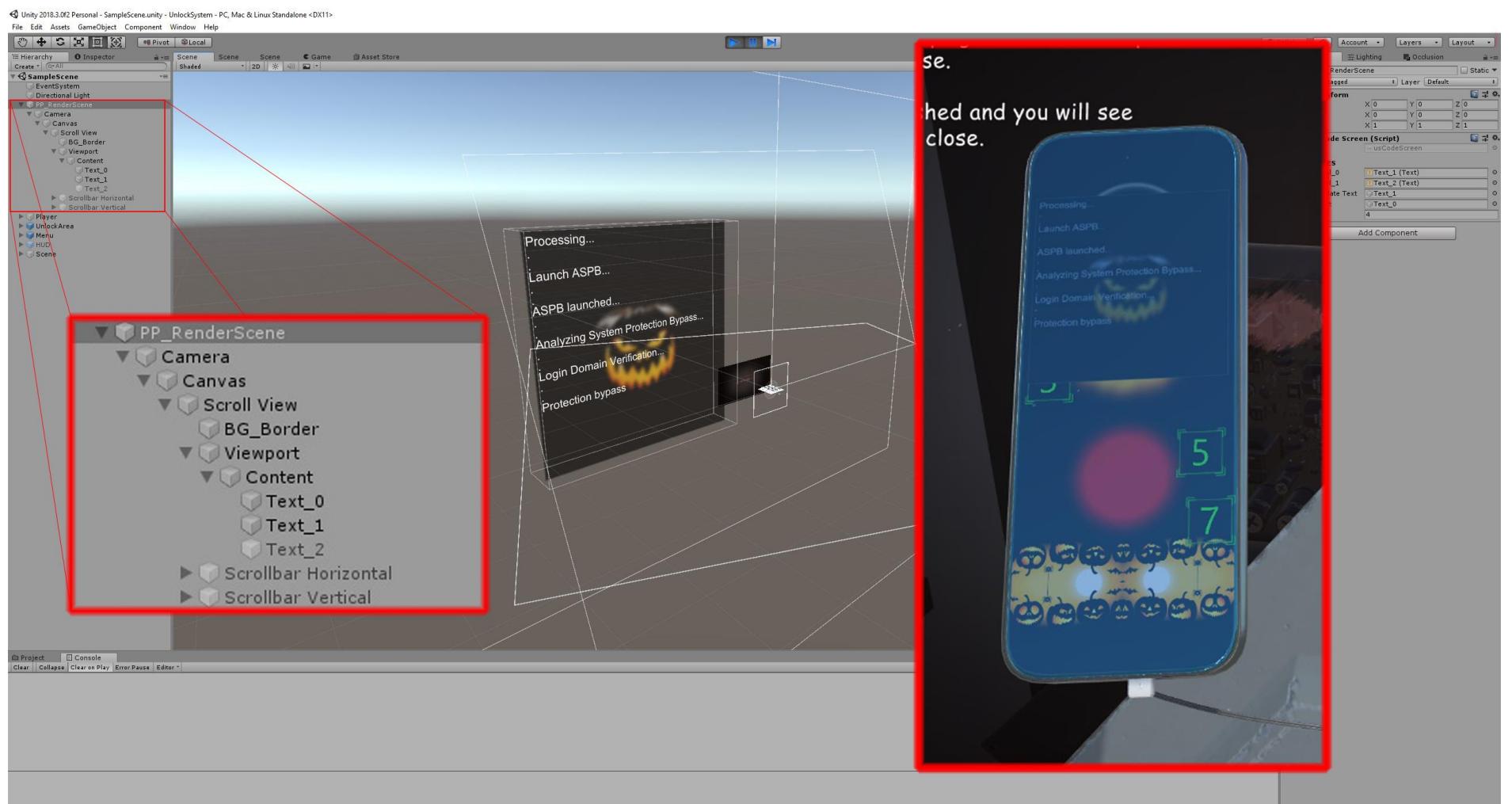


Рисунок 2

Объект *PP_RenderScene* имеет скрипт *UsCodeScreen*, который имеет параметры, показанные на рисунке 3. Первые 4 параметра это ссылка на текст, который будет прокручиваться на экране в процессе сканирования. Параметр *SpeedRoll* (или *Scroll*) это скорость прокрутки текста, поэкспериментируйте с этим параметром, чтобы увидеть результат.

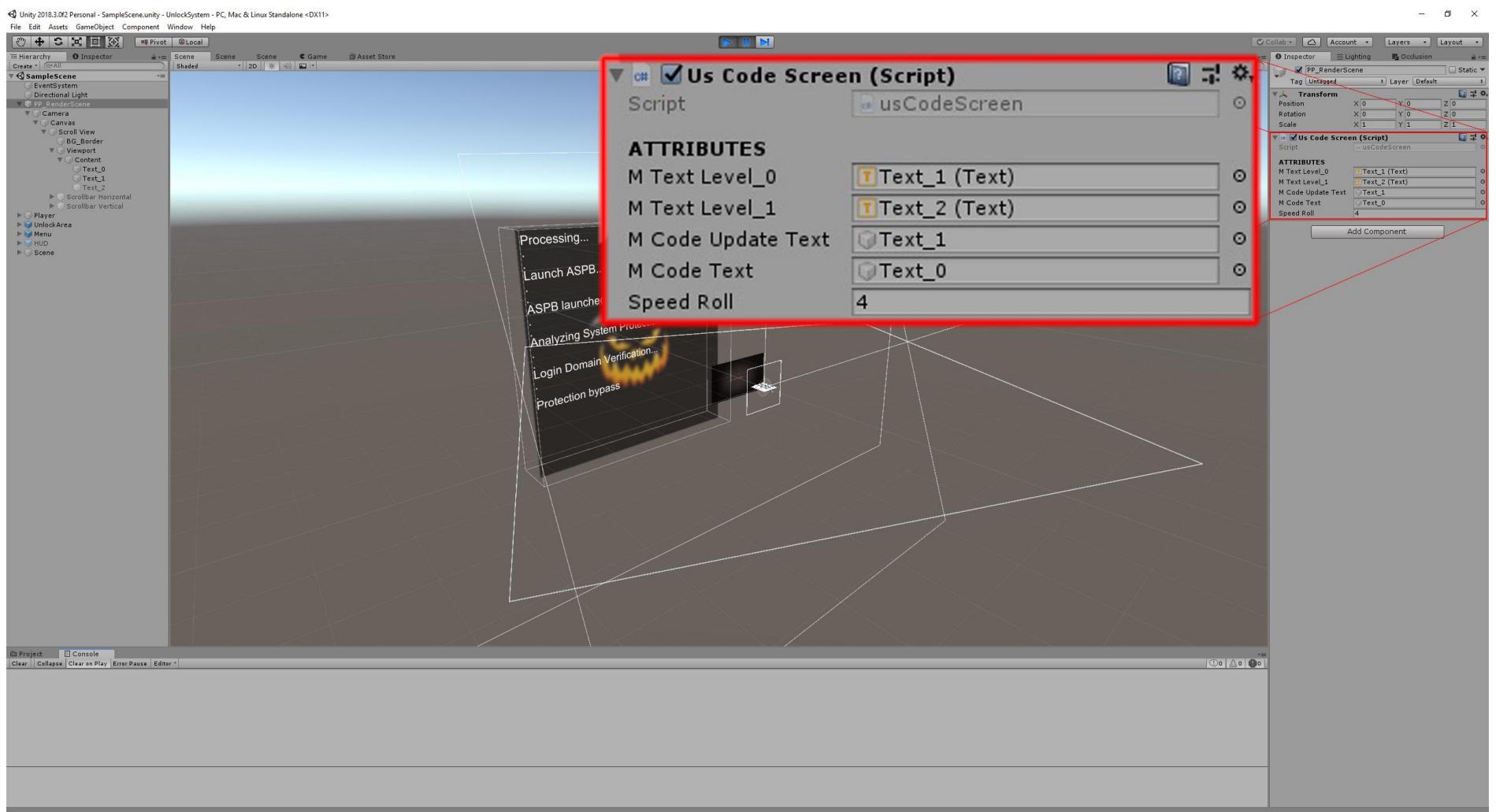


Рисунок 3

2. *Player* – это игрок (*character*) (рисунок 4). Им вы управляете, он имеет камеру и колайдер. К нему прикреплены 3 скрипта:

- *SFPSController* – скрипт управления игроком (движение, вращение и тп)
- *UsTest* – по большей части этот скрипт прикреплен к элементу “*Camera*”, т.к. он посылает луч от камеры, когда мы нажимаем на клавишу “F”. Также этот скрипт включает и отключает разные иконки, которые появляются в центре экрана (например, “Door Open” или иконку – “замок разблокирован” и тп)
- *UsPlayerItems* – скрипт с предметами, в данном случае используется для задания количества отмычек.

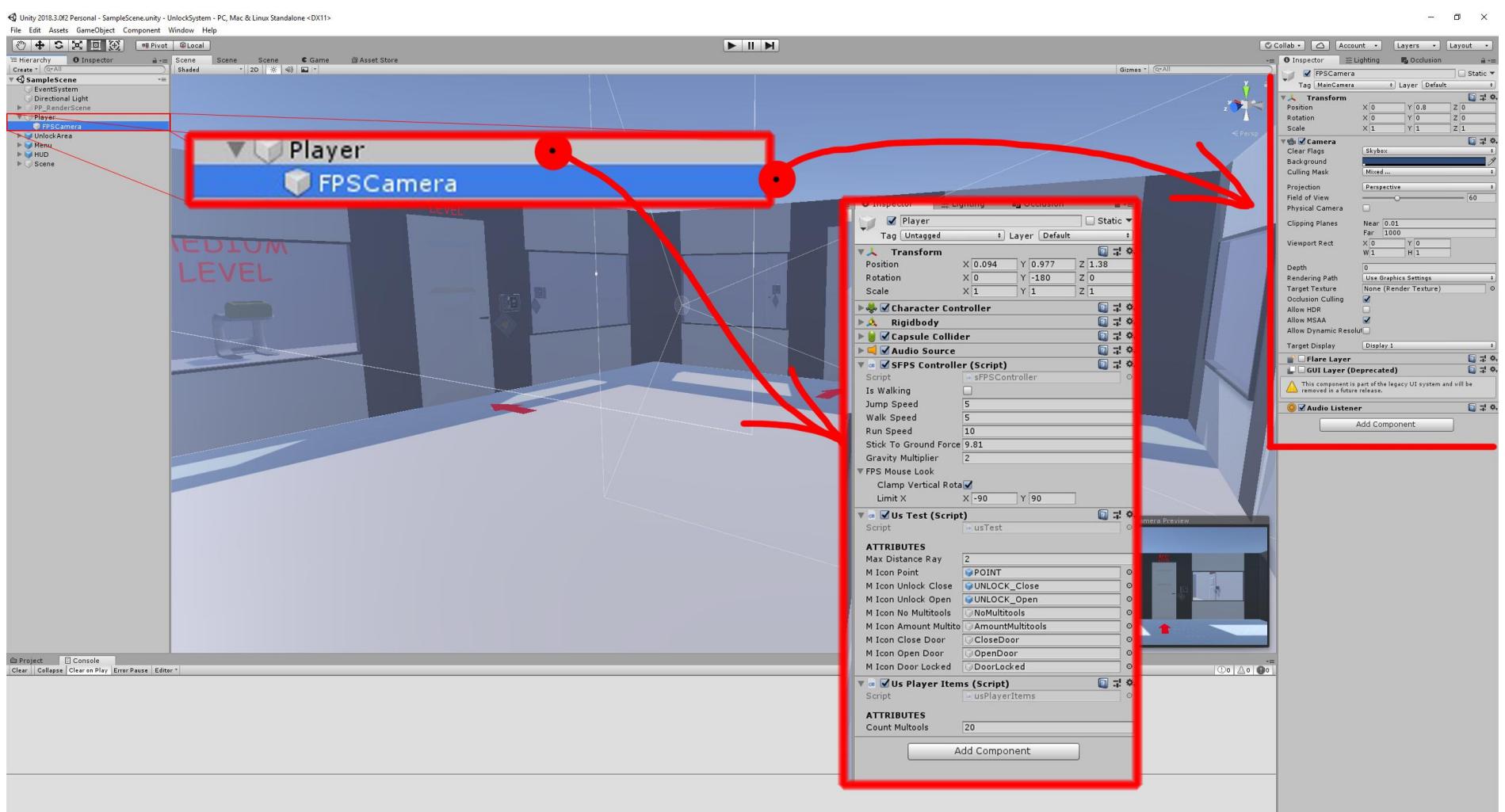


Рисунок 4

3. *UnlockArea* – интерфейс системы (механический и электронный). В этом месте происходят все события. По дефолту он неактивен, когда мы подходим к двери и запускаем интерфейс, то *UnlockArea* активируется и в зависимости от выбранного интерфейса появится либо механический (*UnlockPoint_0*), либо электронный (*UnlockPoint_1*) интерфейс.

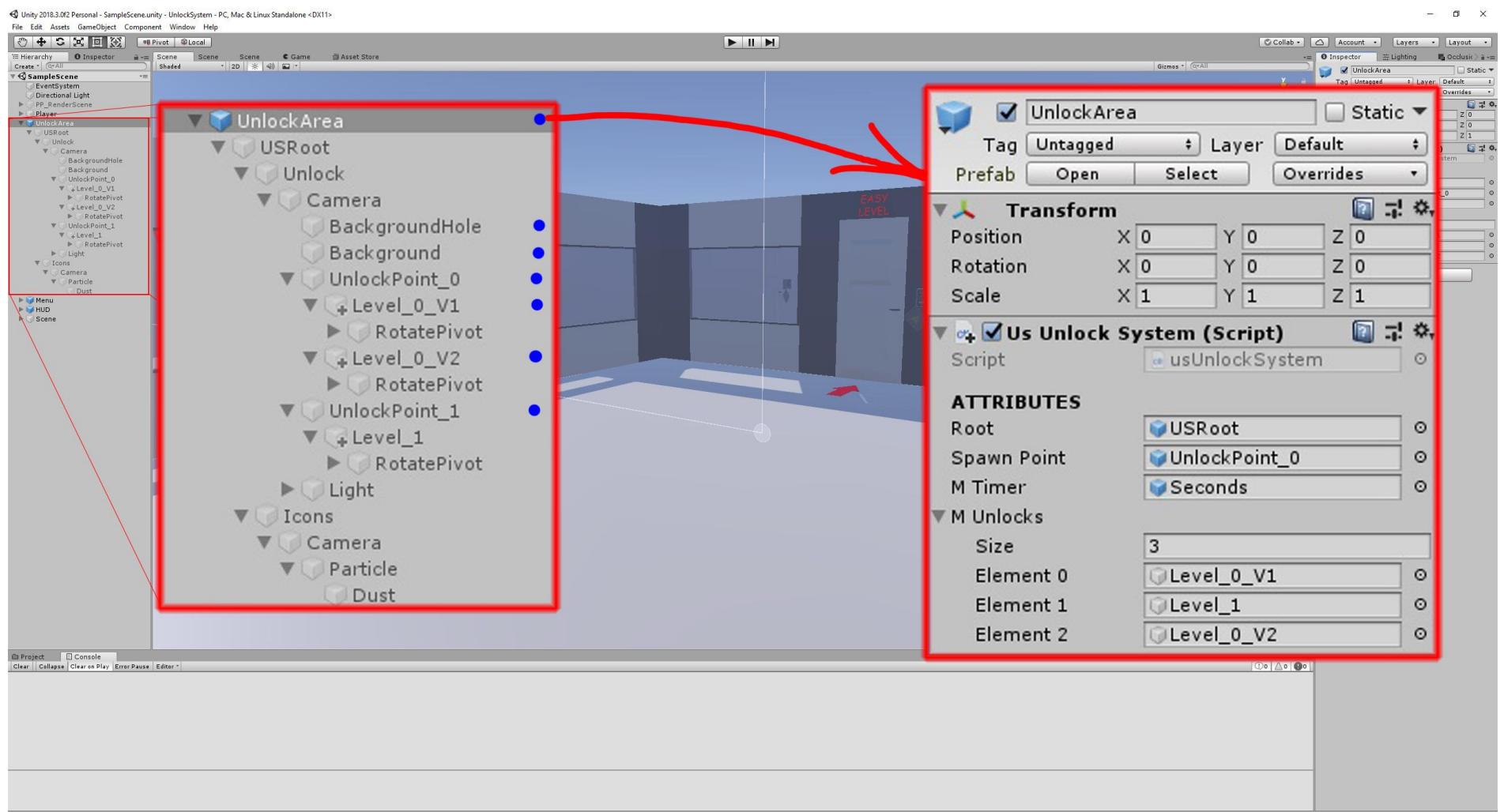


Рисунок 5

Объект *UnlockArea* имеет скрипт *UsUnlockSystem* (рисунок 5). Это скрипт, который передает интерфейсам все параметры, необходимые для работы с ним. Параметр *Root* – это ссылка на корень объекта, т.к. он не активен нам нужно будет его включить при запуске интерфейса. Параметры *SpawnPoint* и *MTimer* на данном этапе не задействованы, в дальнейшем они будут удалены. Параметр *MUnlocks* это массив “уровней”, если вы хотите добавить свой вариант замка, то вам следует добавить его в этот массив.

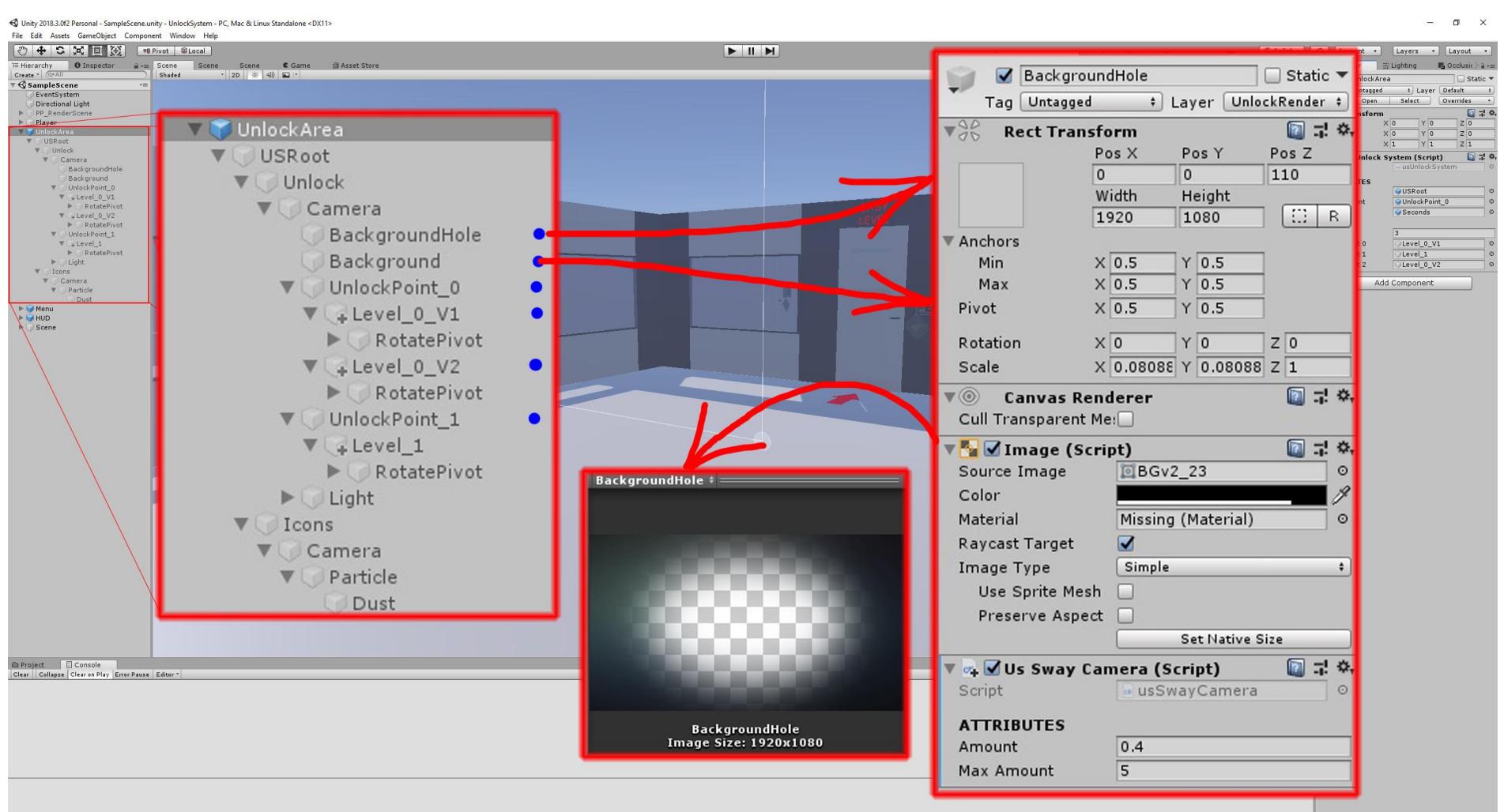


Рисунок 6

Объекты *BackgroundHole* и *Background* выполняют роль заднего фона. Для придания большей динамики картине они имеют скрипт *UsSwayCamera*, который придает им эффект перемещения при движении мышью. Отличия между ними только в том, что они имеют разные текстуры.

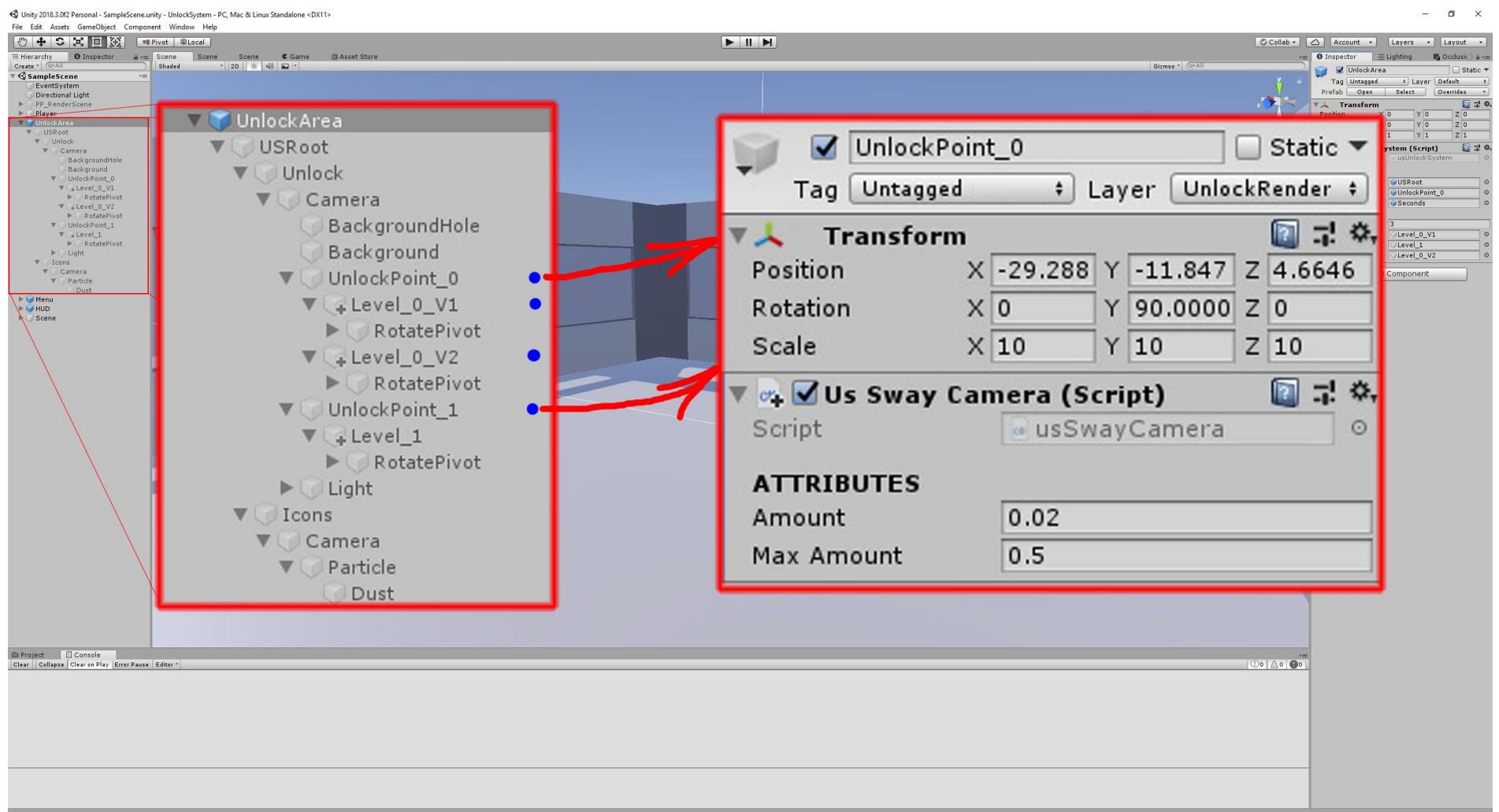


Рисунок 7

Объекты *UnlockPoint_0* и *UnlockPoint_1* – это, как уже было написано ранее, интерфейсы систем. К ним прикреплен скрипт *UsSwayCamera* для придания динамики.

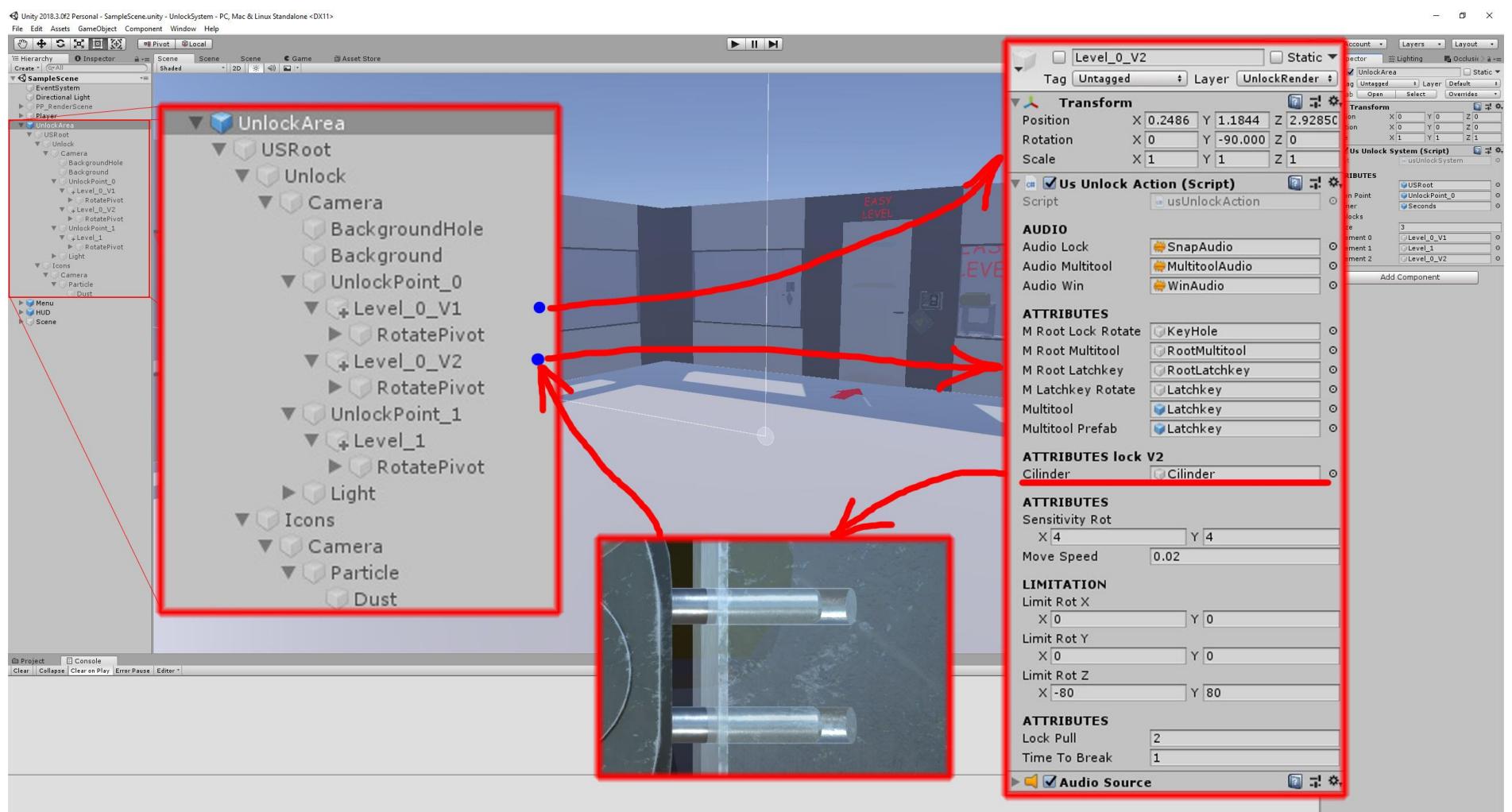


Рисунок 8

Объекты *Level_0_V1* и *Level_0_V2* это два варианта исполнения замка, добавьте сюда еще один вариант исполнения интерфейса и пополните массив *MUnlocks* (см. выше). Объекты имеют скрипт *UsUnlockAction*, этот скрипт относится только к механическому интерфейсу системы. Первые параметры это аудио сопровождение. Когда мы вращаем отмычку то воспроизводится звук. Регион “*ATTRIBUTES*” это ссылка на различные объекты, такие как отмычка, мультитул, замочная скважина (его модель) и тп. Следующий регион “*ATTRIBUTES lock v2*” относится только ко второму варианту замка. Это некие элементы (рисунок 8), которые будут отпираться, когда мы выигрываем, имитируя, что дверь открыта. Следующий регион “*ATTRIBUTES*” – параметр *SensitivityRot* – чувствительность вращения отмычкой (скорость), параметр *MoveSpeed* – скорость вращения мультитула. Регион “*LIMITATION*” – регион ограничений вращения отмычки и мультитула. Регион “*ATTRIBUTES*” – параметр *LockPull* – предельное отклонение вибрации отмычки, параметр *TimeToBreak* – время, после которого отмычка сломается.

Объект *Icons* – это система частиц. Которая появляется на заднем плане интерфейсов.

Объекты механического интерфейса не так интересны, т.к. в них нет самостоятельных элементов. Рассмотрим подробней объекты электронного интерфейса.

Внутренний объект *EL_MeshAssembler* имеет скрипт *UsLightSwitcher*, который раз в один кадр (параметр *fps* = 1) переключается между лампочками.

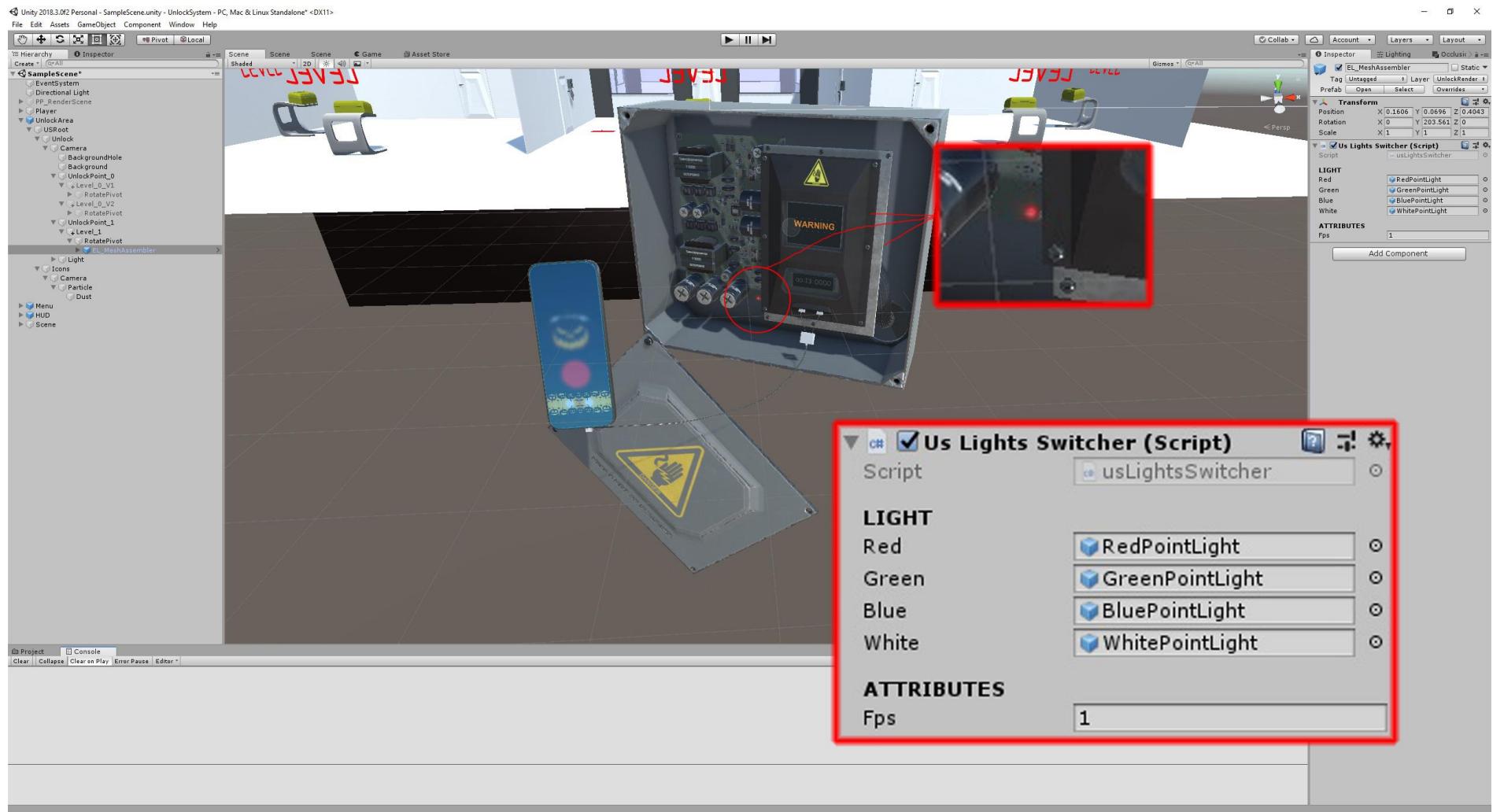


Рисунок 9

Открыв эту сборку и опустившись в самый низ, мы найдем 3 элемента (рисунок 10), о них и поговорим.

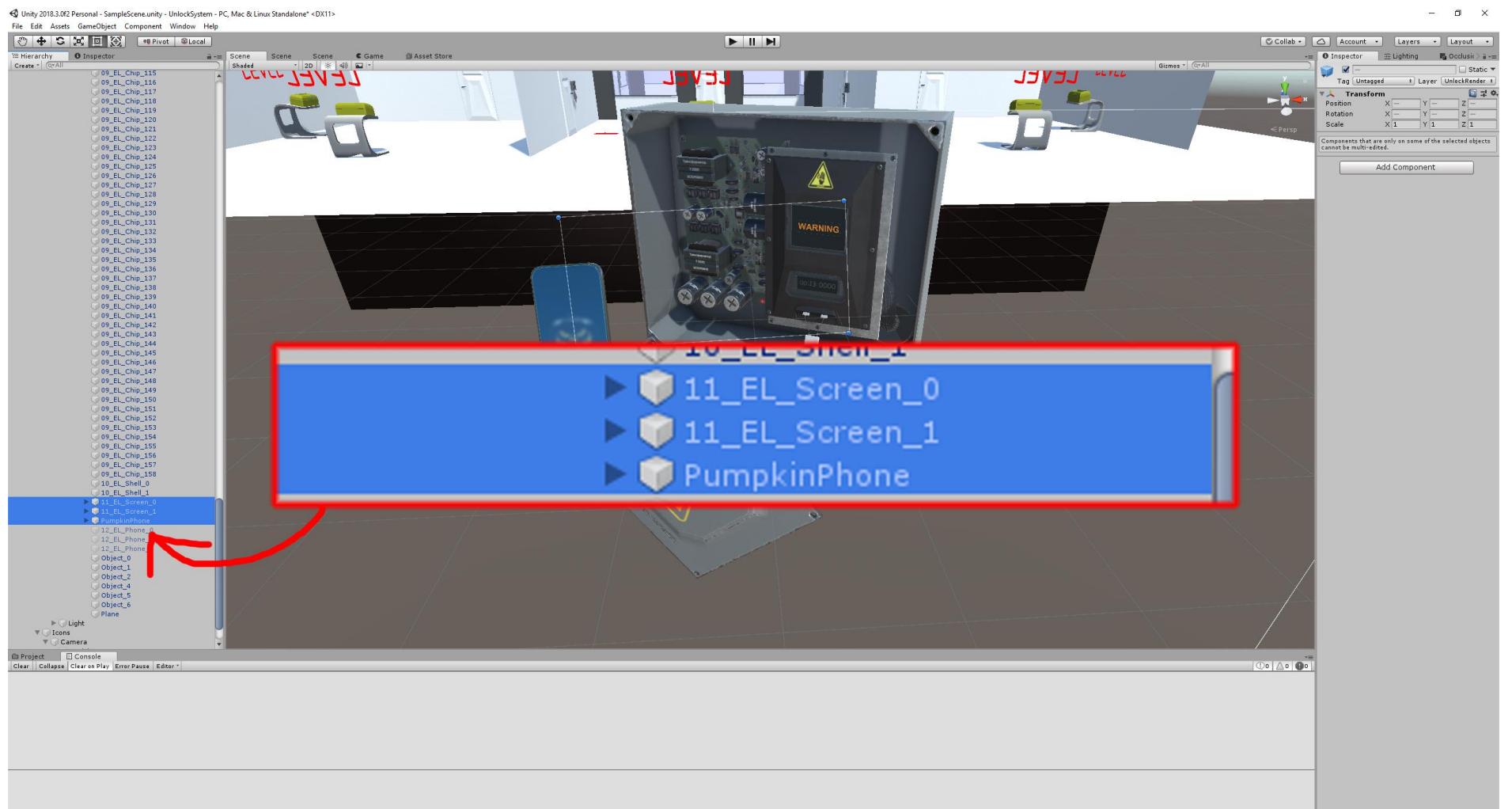


Рисунок 10



Рисунок 11

Как я уже писал ранее, один из экранов электронной коробки свободен, объект *11_EL_Screen_0* это свободный экран, который имеет только одно изображение.

На рисунке 12 показан второй экран с таймером, в нем есть два объекта *Seconds* и *MilliSeconds* это и есть наше время, к ним прикреплен скрипт *UsSearchLootIcon*, скрипт, который анимирует текстуру, он не сложный, изучите его, и вы все поймете.

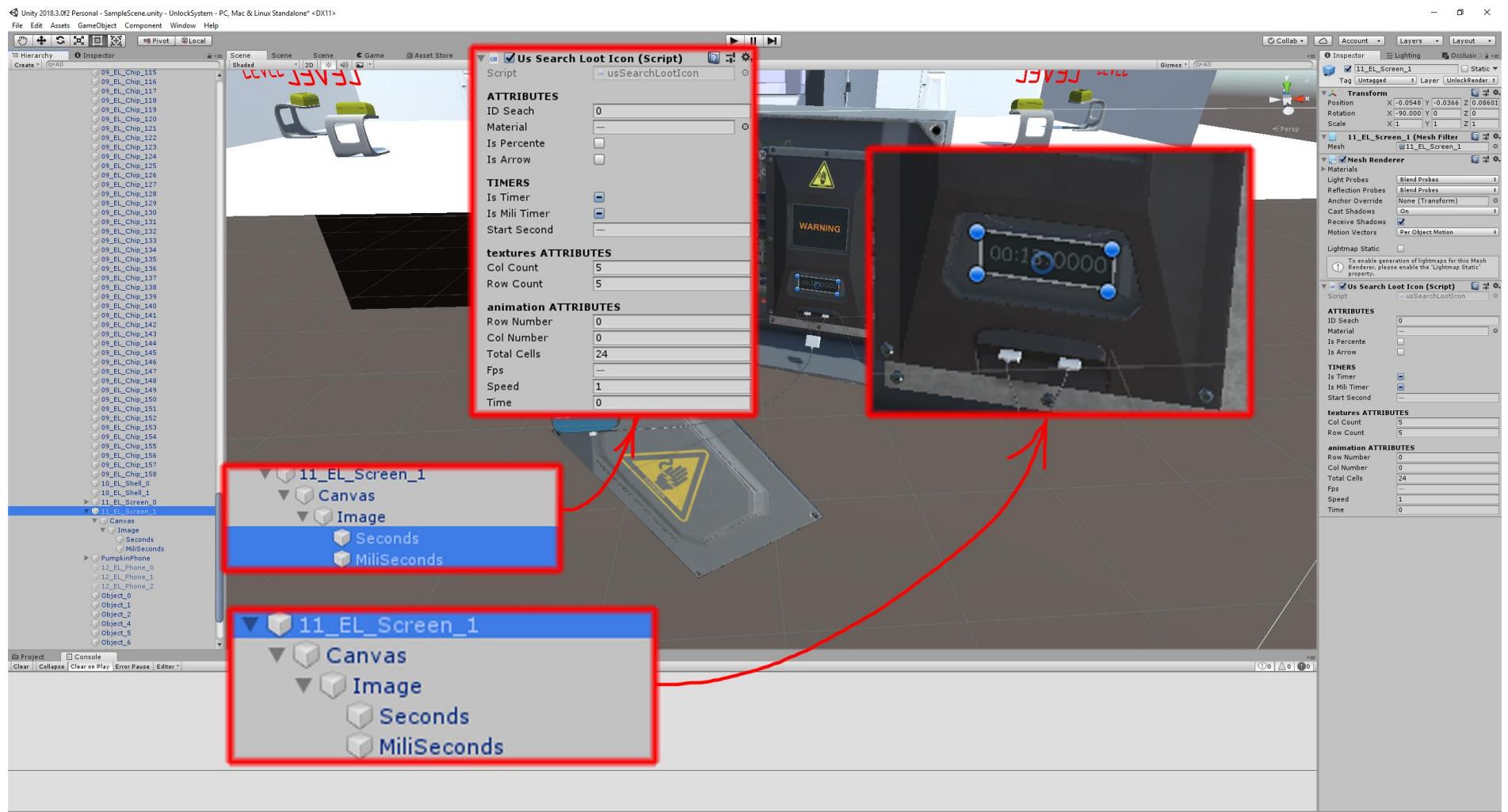


Рисунок 12

Ну и самое интересное это телефон “*PumpkinPhone*”. Объект *PumpkinPhone* (рисунок 13) это интерактивный объект, с ним и взаимодействует игрок. К этому объекту прикреплен главный скрипт электронного интерфейса *UsPhoneScript*. Который выполняет большинство функций этого интерфейса.

Регион “*ATTRIBUTES*” пропустим, т.к. он понятен, это всего лишь ссылка на объекты. Регион “*ICONS*” это ссылка на таймер, т.к. ему нужно будет передавать параметры (такие как время старта). Регион “*CODE BUTTON*” этот регион относится к нашим кнопкам на экране (кодовые кнопки). Параметр *CodeBox* ссылка на объект, где будут появляться кнопки. Параметр *CodeBoxPrefab* это префаб нашей кнопки, которую мы будем копировать, в зависимости от уровня (количество кнопок зависит от уровня сложности). Параметр *MCodeBoxes* это спрайты (текстуры с цифрами – рисунок 14). Параметр *PositionCodeBoxInScene* это массив всех возможных позиций кнопок на экране. Вы можете сделать его рандомным, но тогда вам нужно будет писать условия на перекрытие и ограничение зоны спавна кнопок на экране.

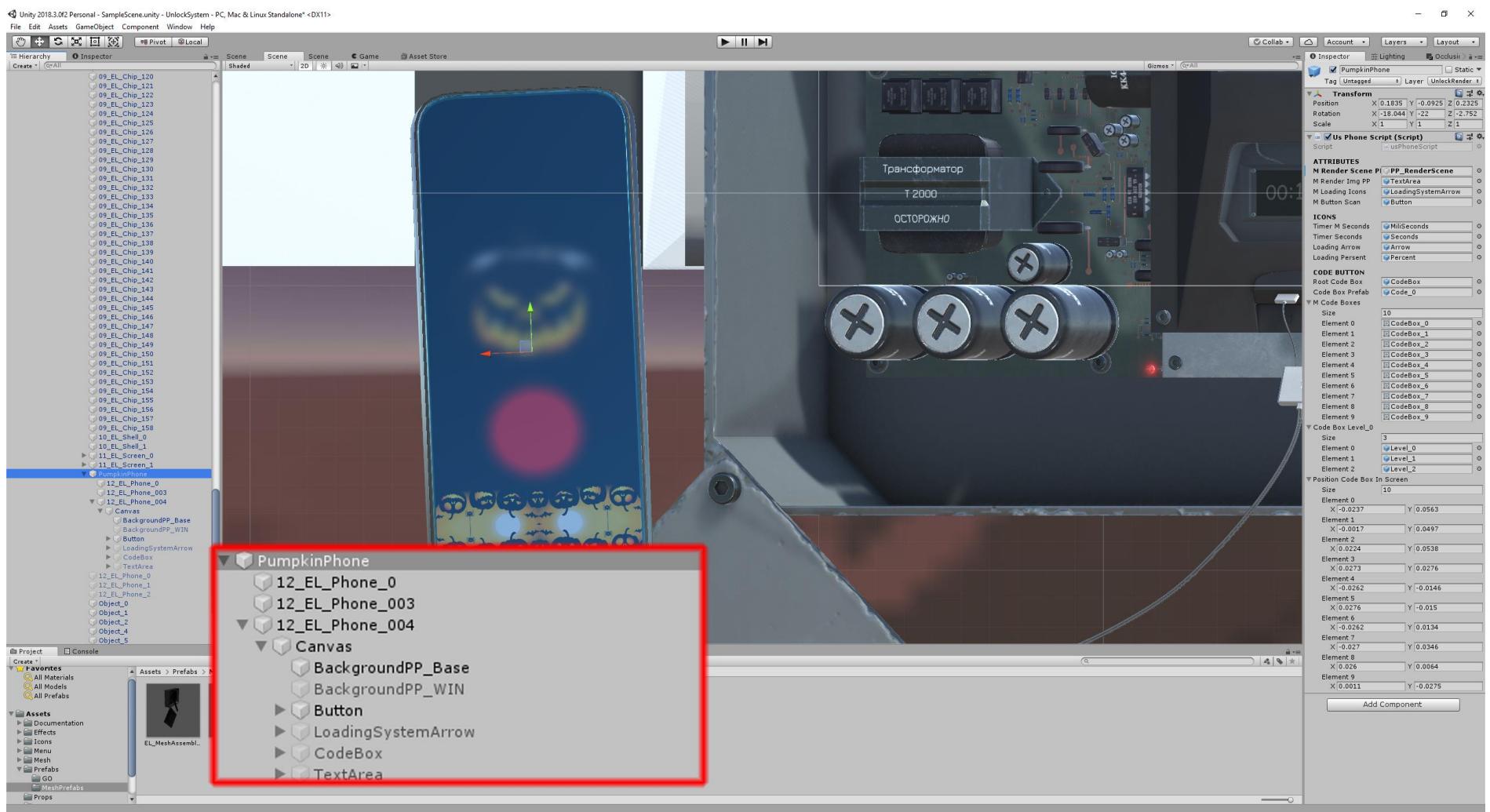


Рисунок 13

Объекты *12_EL_Phone_0* и *12_EL_Phone_003* это модели корпуса и стекла телефона. Объект *12_EL_Phone_004* это полотно (*canvas*), на котором происходят все действия.



Рисунок 14

Рассмотри объект *Button* (рисунок 15), это большая красная кнопка. К ней прикреплен скрипт *UsButtonScan* отвечающий за пульсацию кнопки. Также кнопка имеет действие, при нажатии на нее запускается система сканирования (см. ниже).

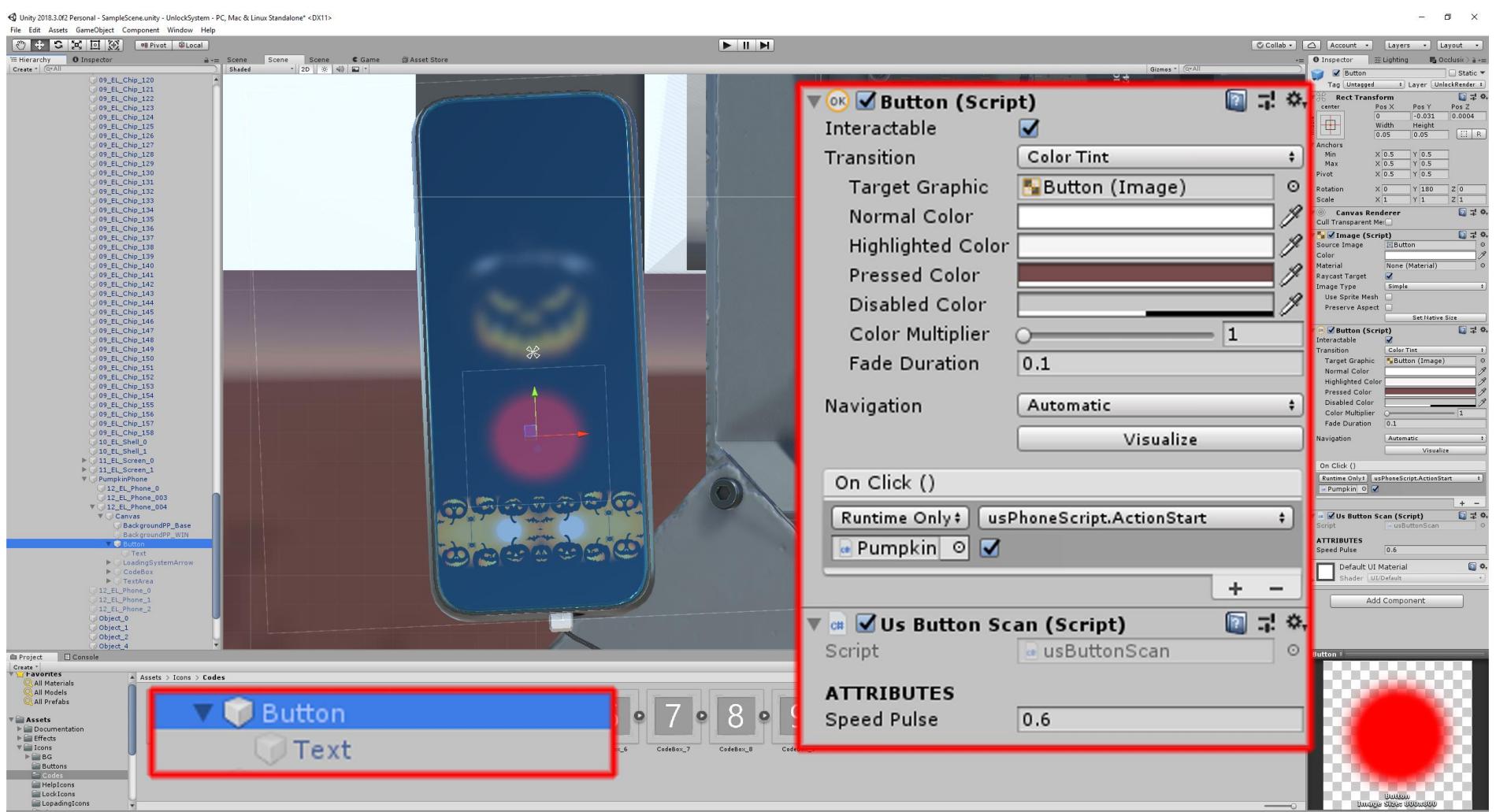


Рисунок 15

По нажатии кнопки запускается система сканирования, появляется окно (рисунок 16), на которую проецируется из сцены рендера *PP_RenderScene* картинка и создается эффект пролистывания кода.

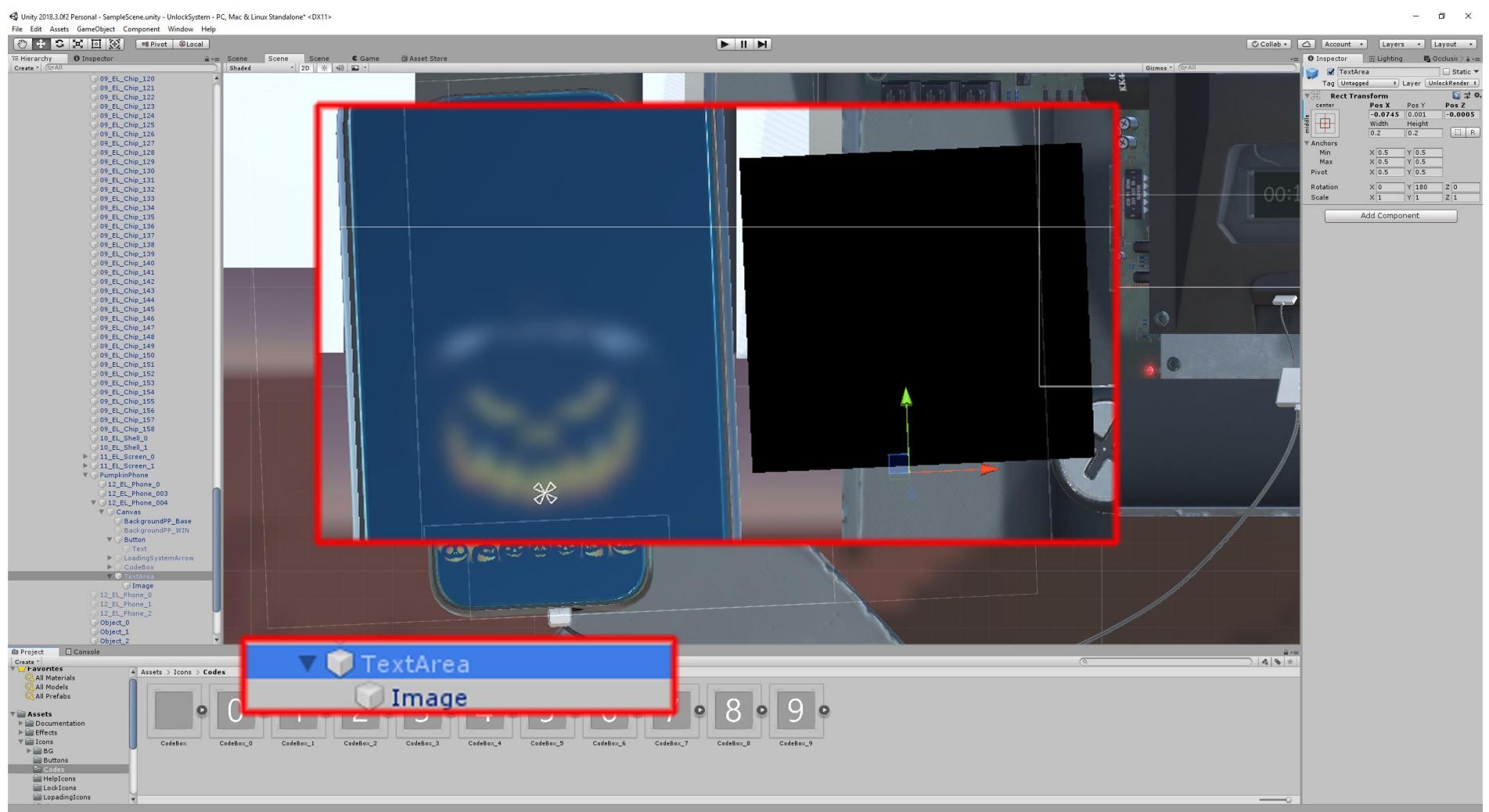


Рисунок 16

4. *Menu* – это объект меню (рисунок 17), в нем находятся такие элементы как подсказки.

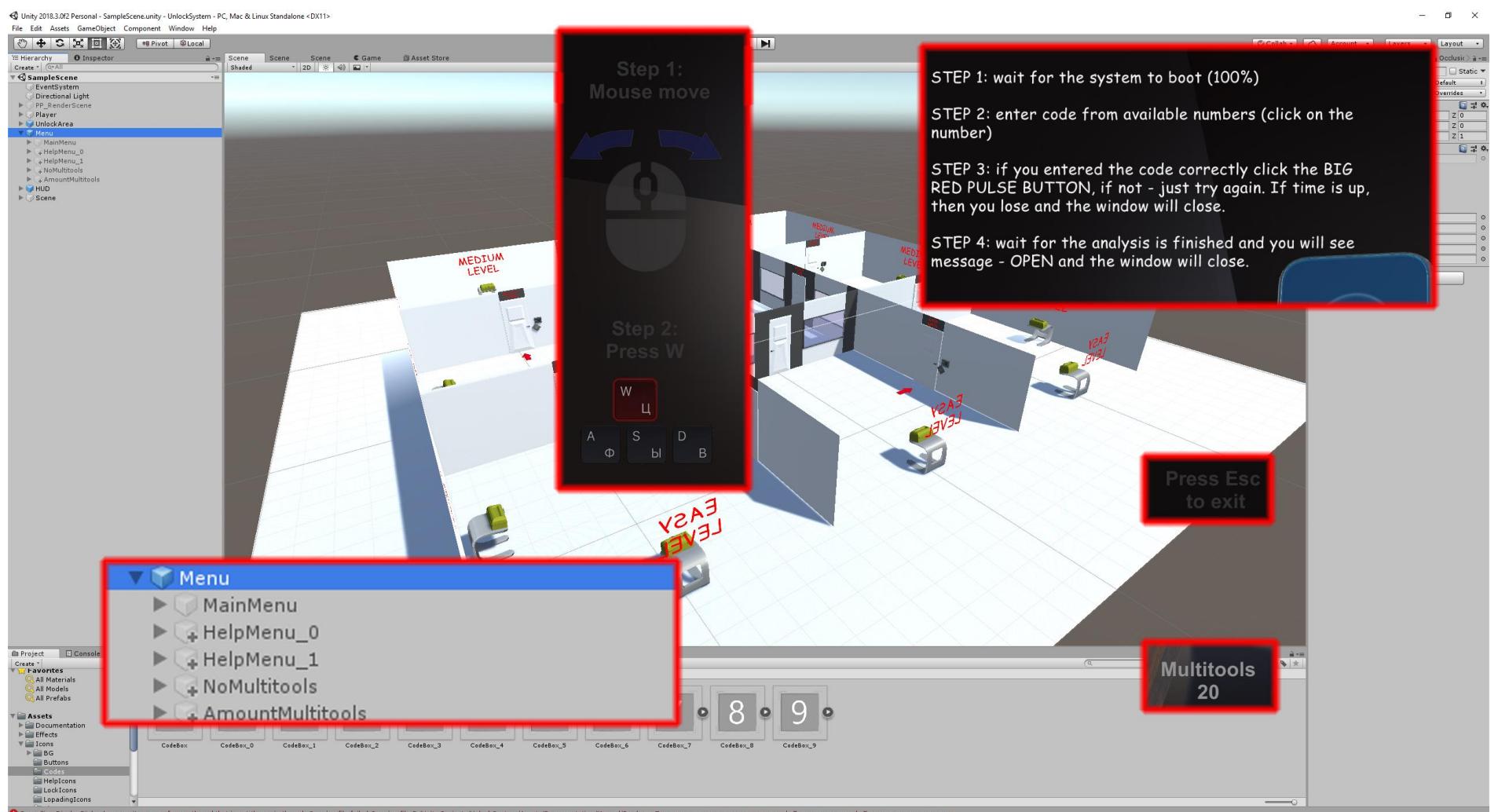


Рисунок 17

5. *HUD* – это интерфейс (рисунок 18), который периодически появляется в центре экрана (зависит от действий игрока).

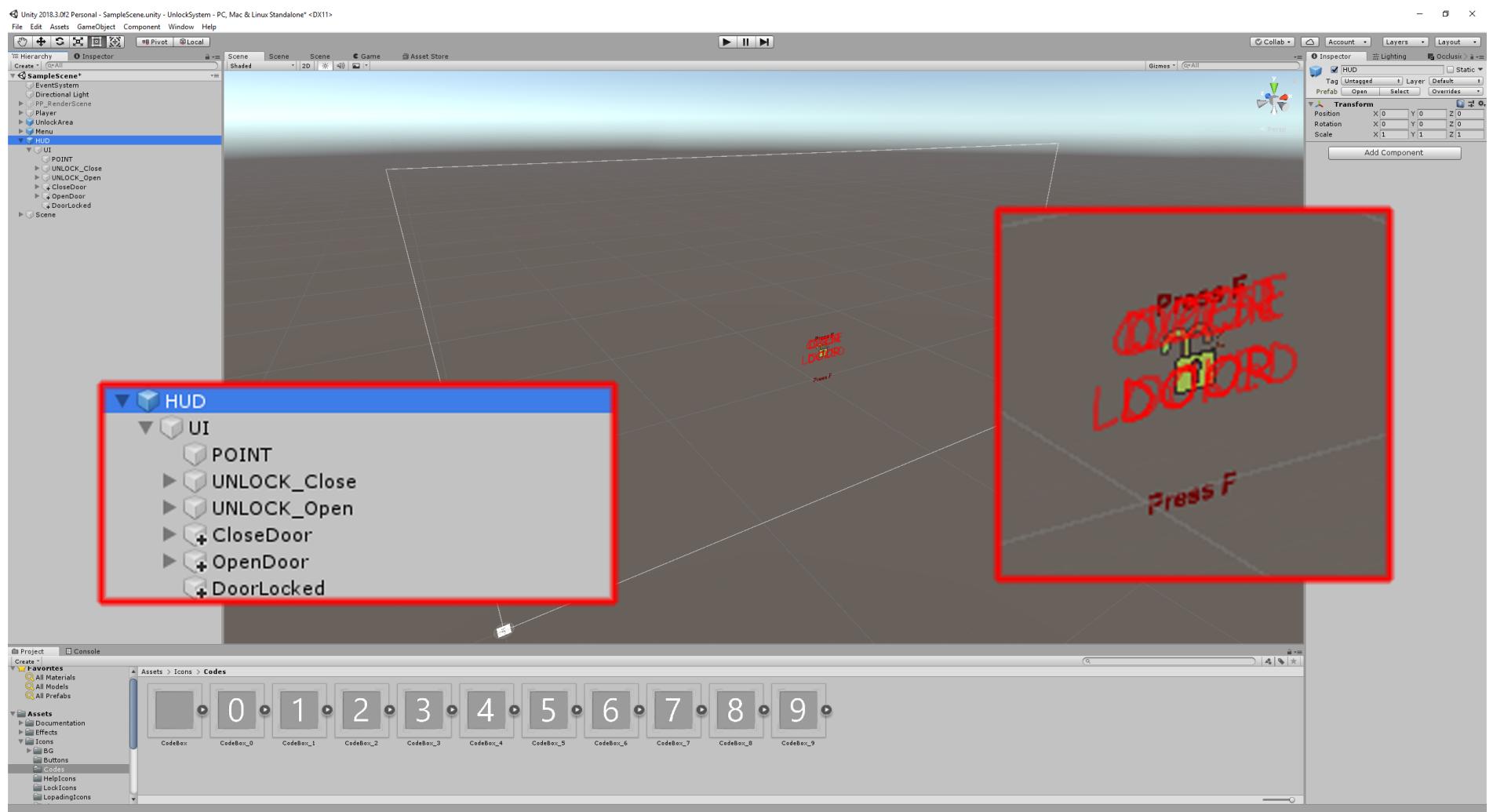


Рисунок 18

6. *Scene* – в этом объекте находятся все модели сцены (рисунок 19). Тут нам будут интересны 2 объекта – *LootBox* и *DoorPrefab*.

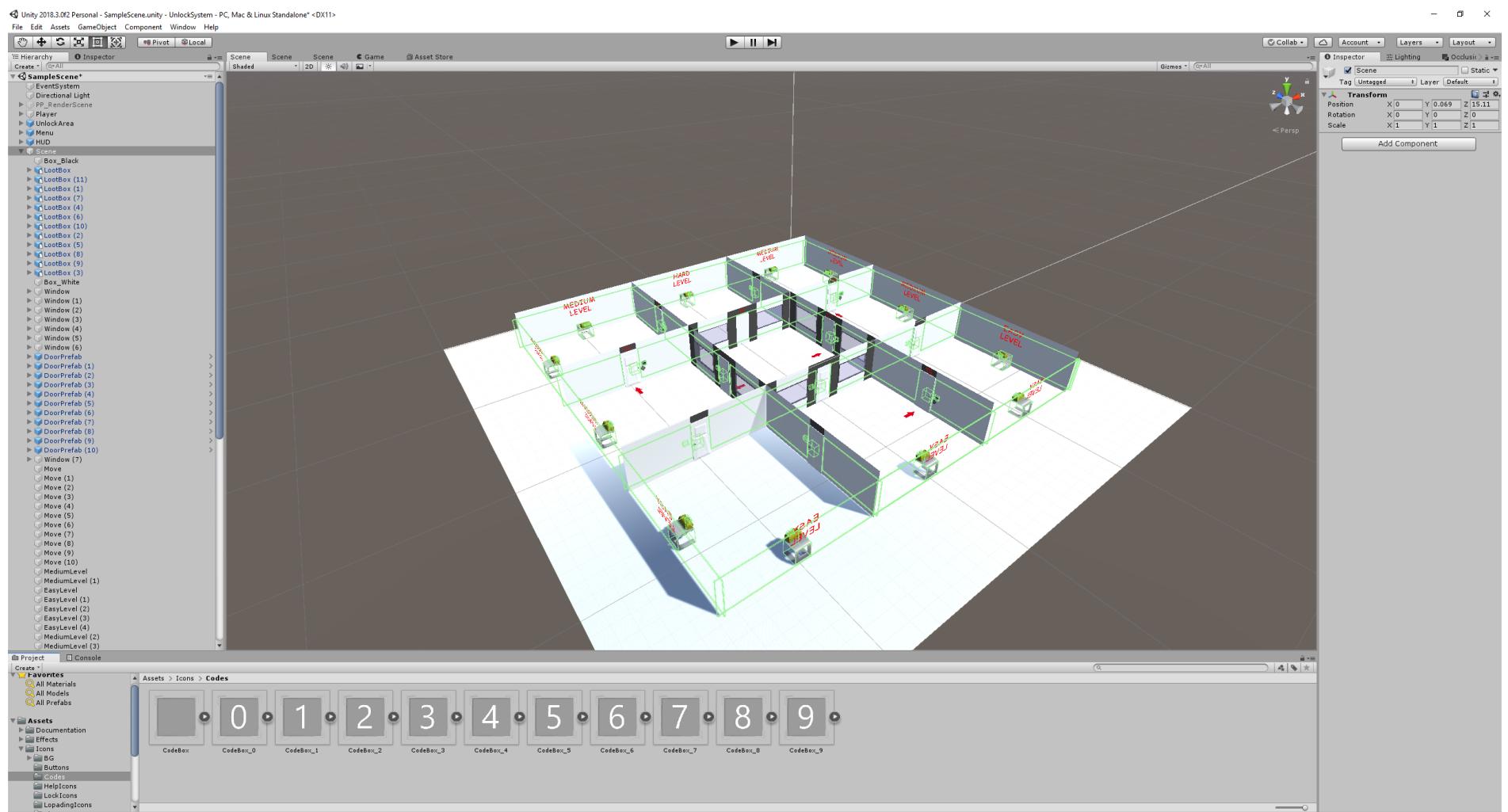


Рисунок 19

Рассмотри эти два объекта подробней.

Объект *LootBox* (рисунок 20) имеет в себе один очень важный объект – *LockBox*, он имеет компонент *Collider* и ему назначен тэг *LockBase* – это значит, что объект относится к механическому интерфейсу. К объекту прикреплен скрипт *UsLootBox*, который передает параметры в интерфейс. Параметр *UsLockOfDifficulty* – уровень сложности, параметр *UsLockLevel* – вариант исполнения замка (интерфейс, не забывайте, что, присвоив тэг *LockBase* объекту выбирайте соответствующие уровни, характерные для механического интерфейса).

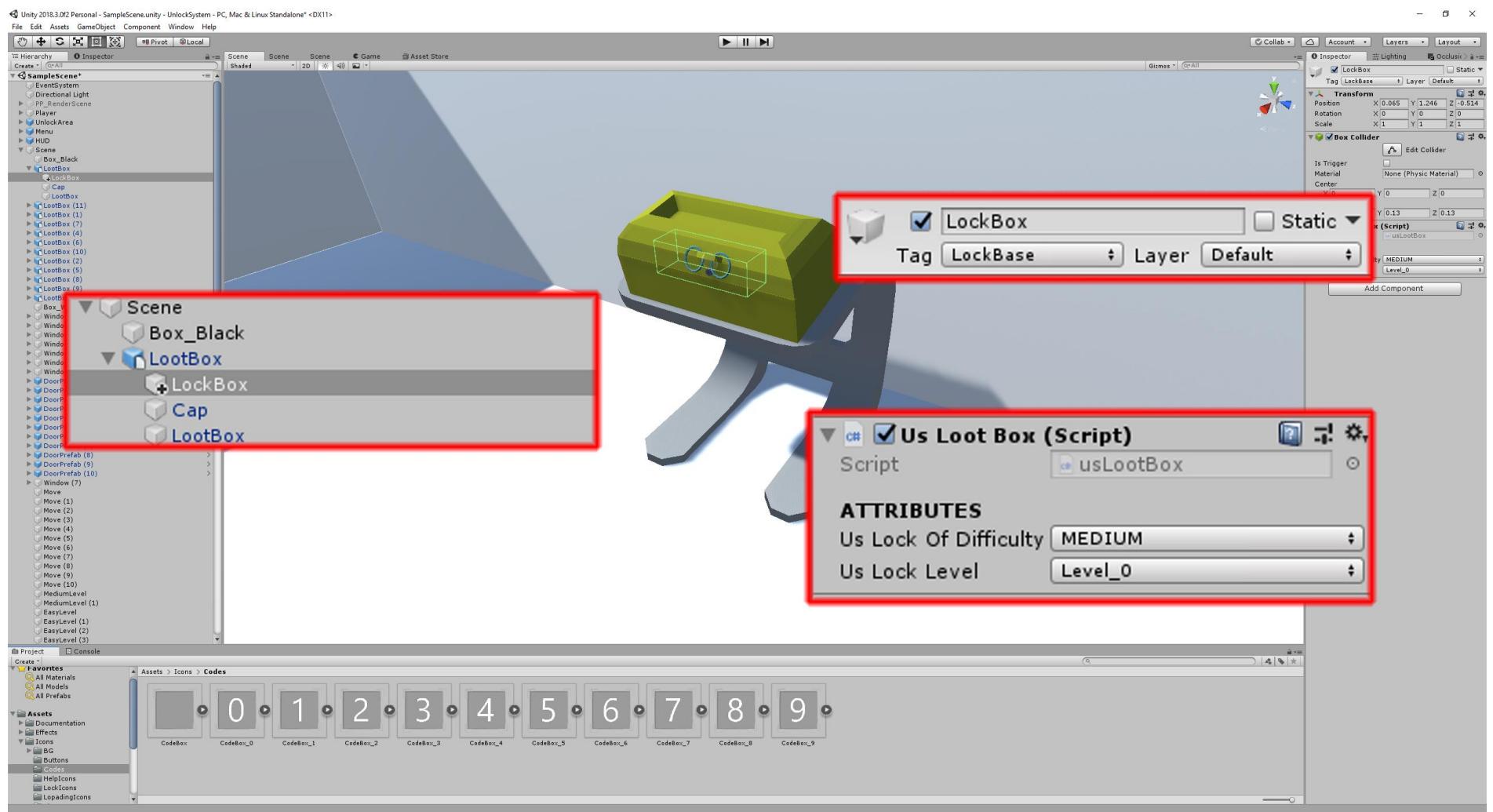


Рисунок 20

Объект *DoorPrefab* (рисунок 21) имеет в себе один очень важный объект – *EL_Box*, он имеет компонент *Collider* и ему назначен тэг *EL_Base* - это значит, что объект относится к электронному интерфейсу. К объекту прикреплен скрипт *UsUnlockEL*, который передает параметры в интерфейс. Параметр *UsLockOfDifficulty* – уровень сложности, параметр *UsLockLevel* – вариант исполнения замка (в данном случае он один).

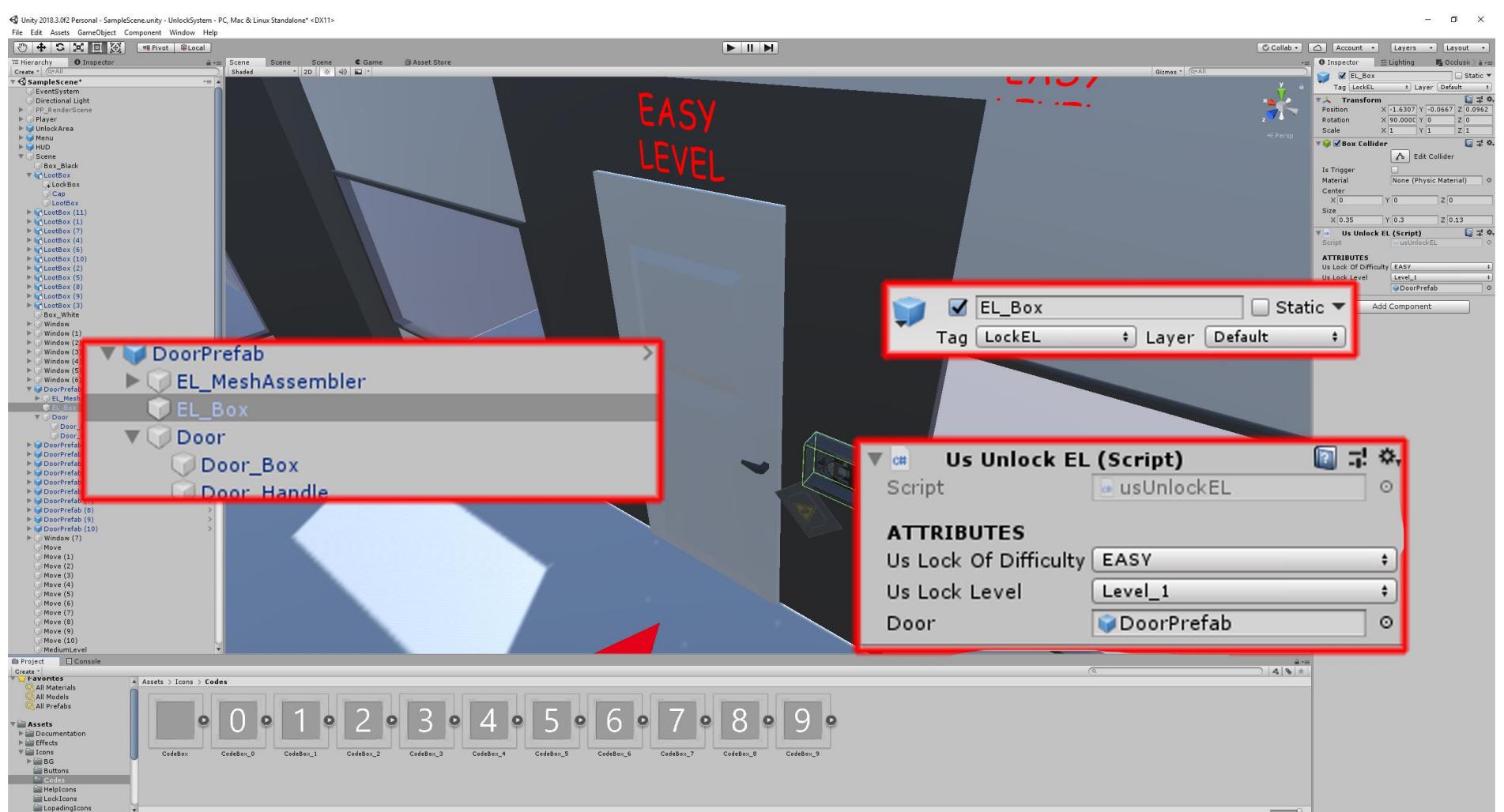


Рисунок 21

ВЕРСИЯ 1.1.5

В версии 1.1.5 был добавлен новый интерфейс, подробнее читайте в документации.

Был добавлен префаб двери для нового интерфейса. Также для нового интерфейса добавлен свой тег «CodeLock».

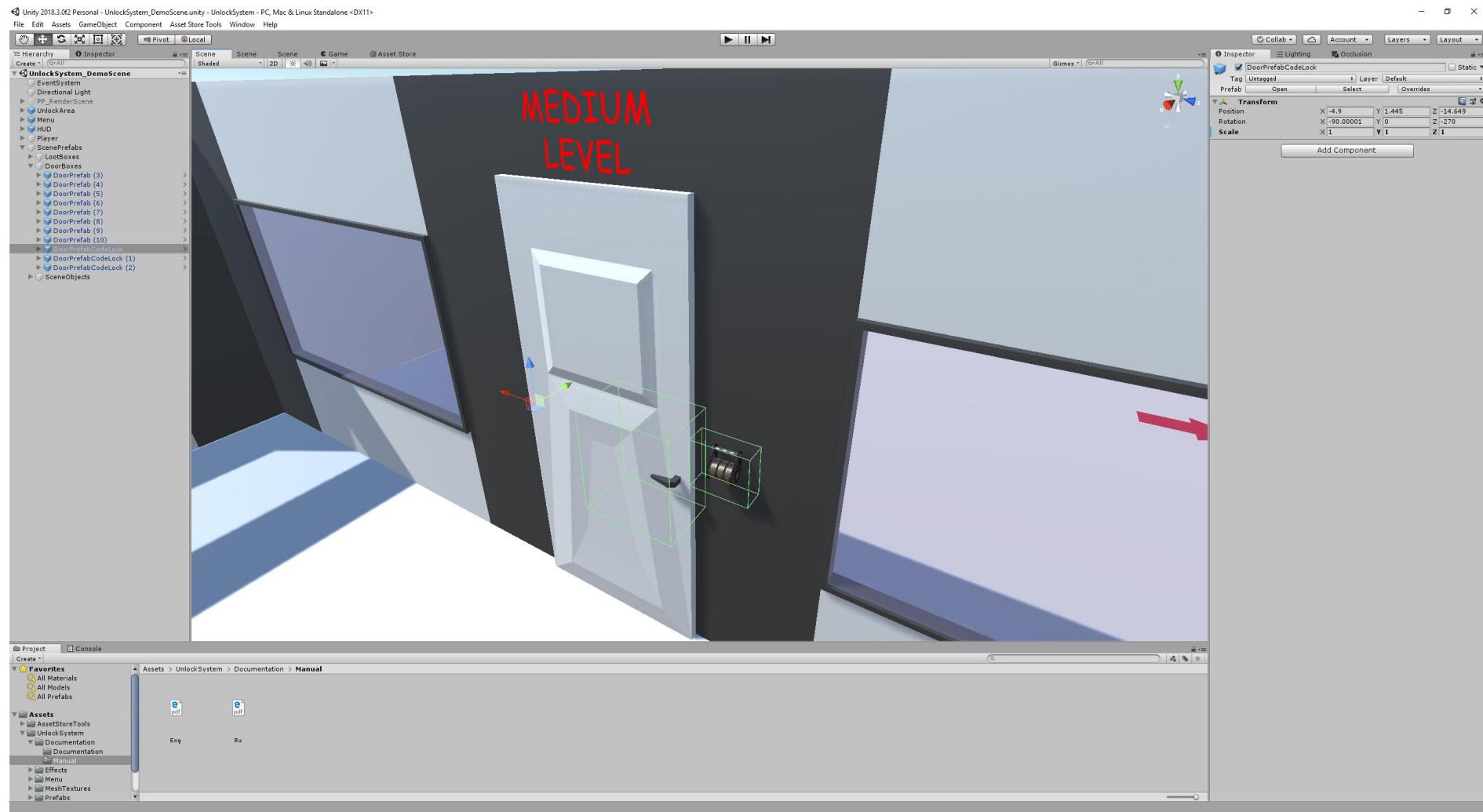


Рисунок 22

В скрипт US_LootBox были добавлены 2 новых параметра для работы с новым интерфейсом.

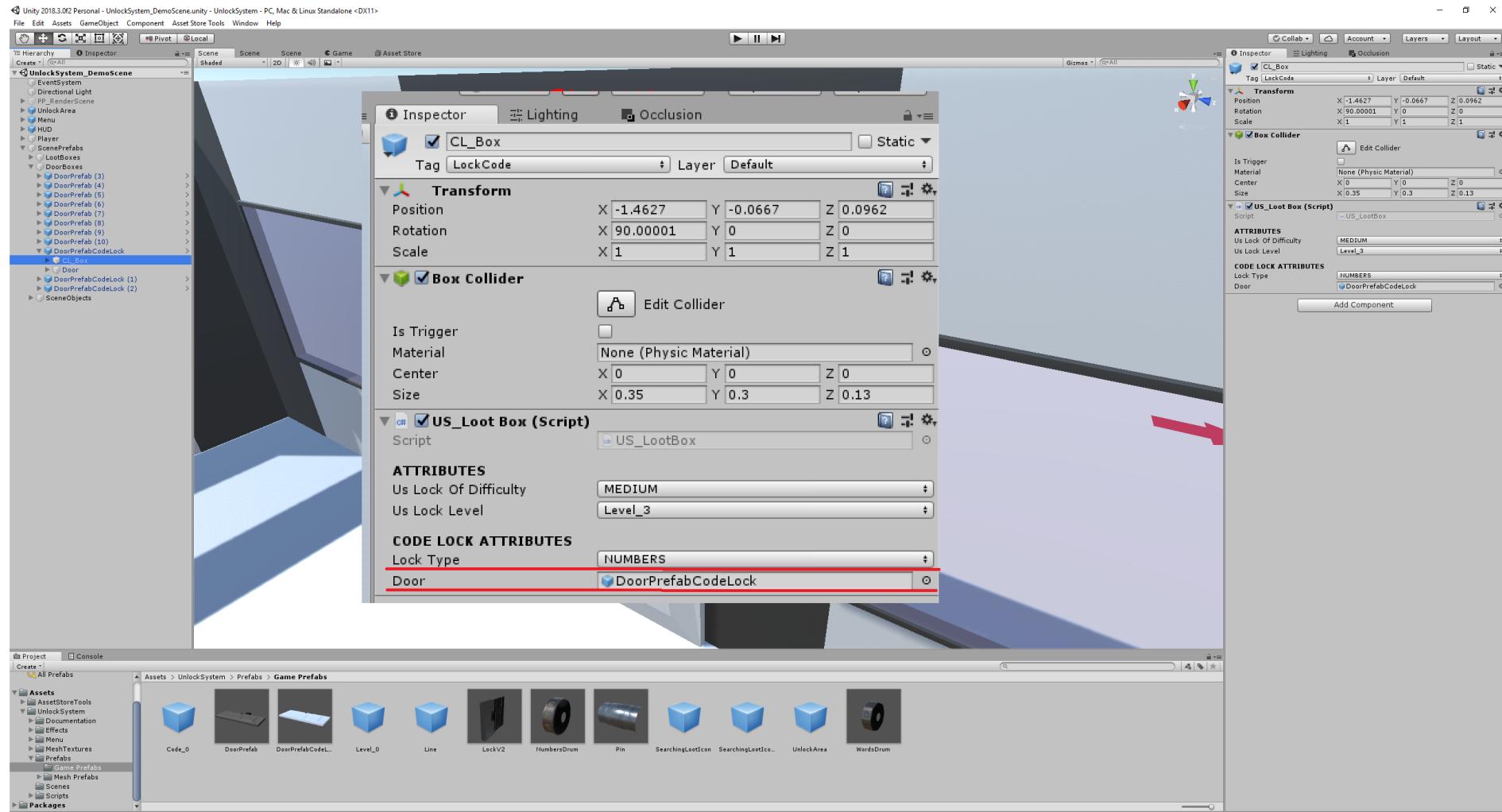


Рисунок 23

Сам интерфейс состоит только из корпусов замка. Все остальные детали (барабаны, оси) будут создаваться автоматически и будут зависеть от уровня сложности.

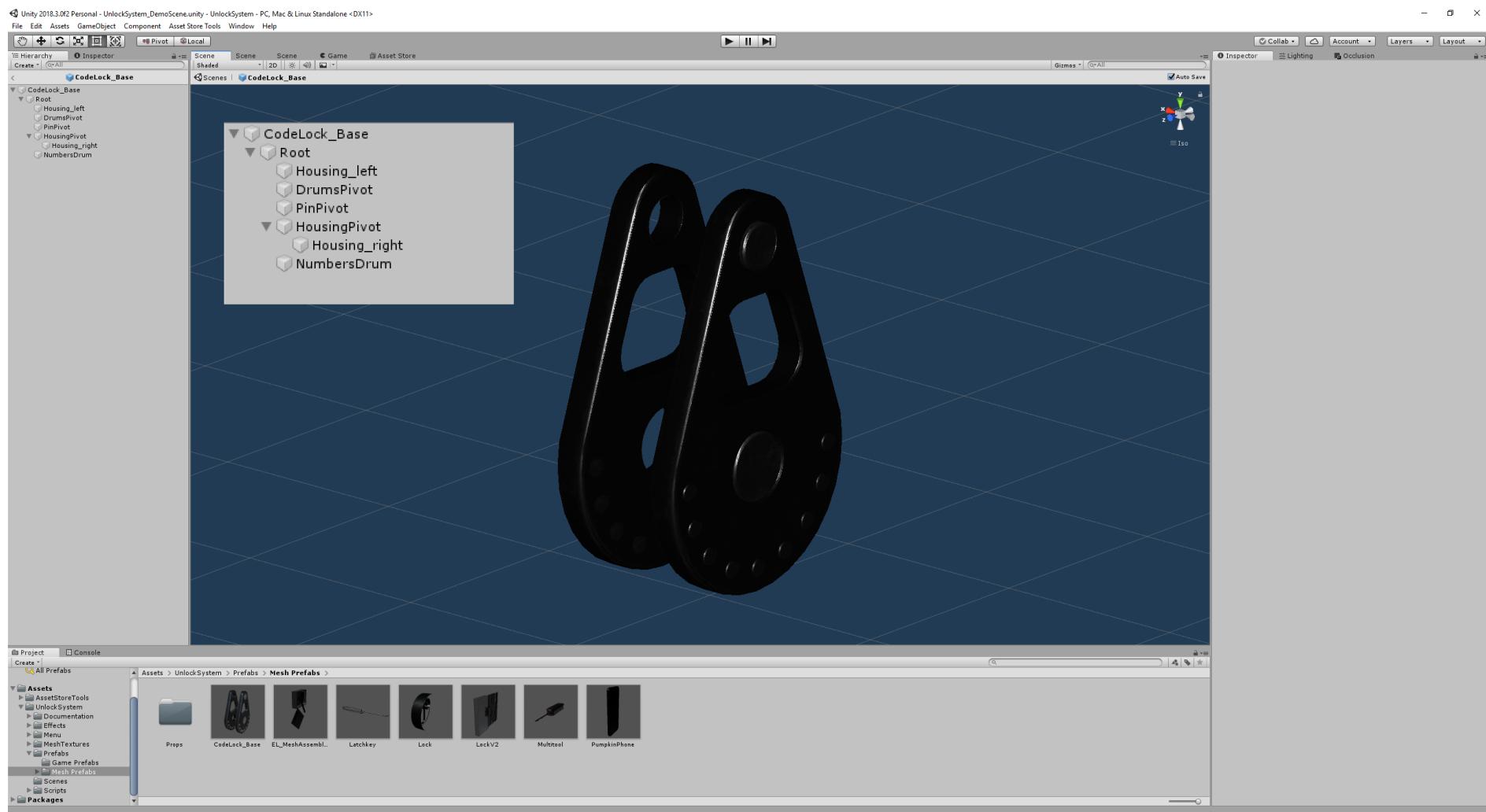


Рисунок 24

Баран включает в себя две кнопки, к которым привязаны действия поворота барабана по оси на заданный угол.

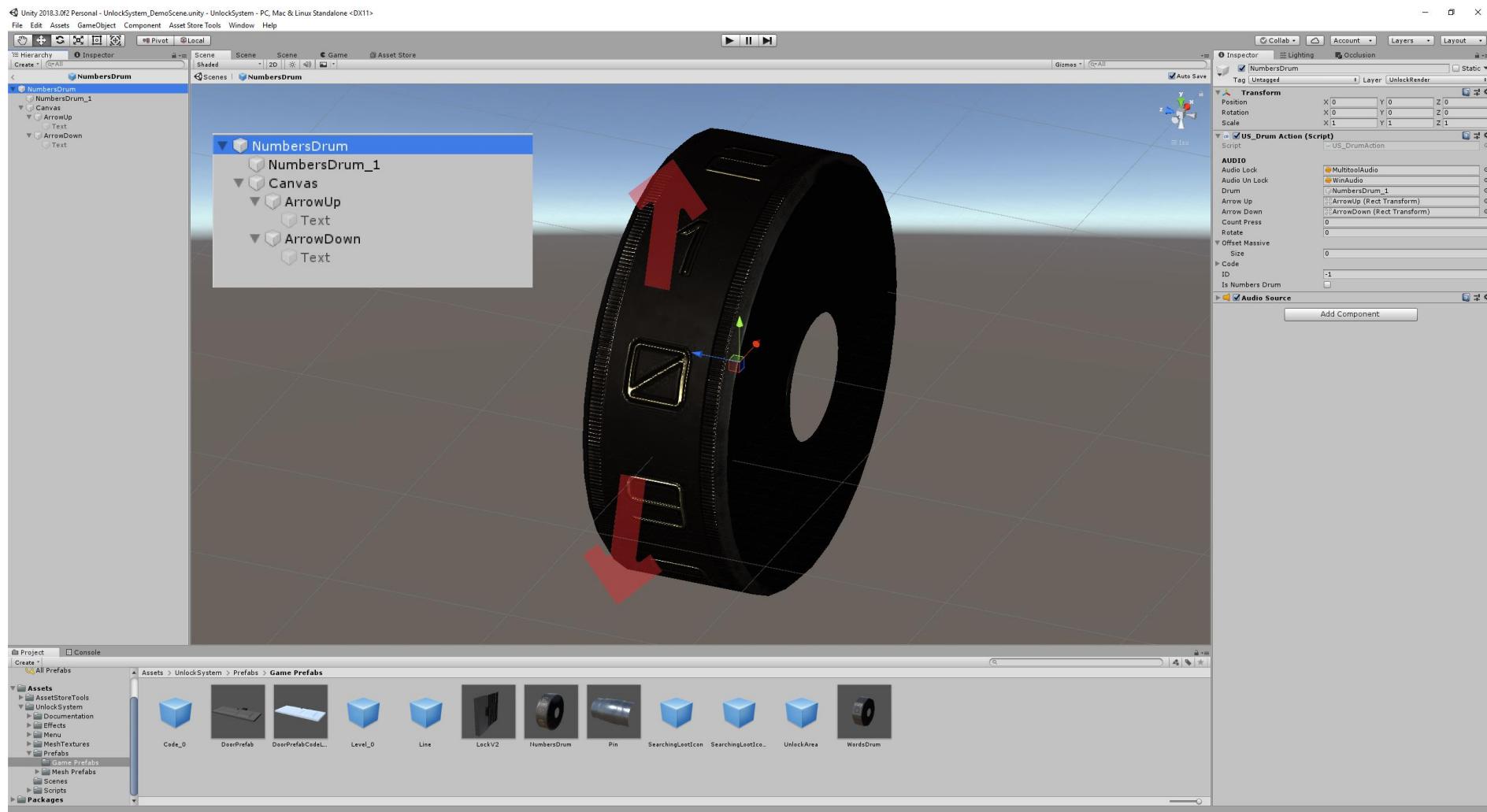


Рисунок 25