**Тема 6. Разработка UI-интерфейса для работы с симулятором установки**

1. Создать холст **CANVAS**, на котором расположить элемент **Text** по центру на размер холста с небольшими полями, «подложить» под него элемент **Image** для создания фона, расположив его на холсте **CANVAS** выше текстового окна, и в свойствах текстового окна набрать информацию о назначении симулятора установки (*в дальнейшем после перехода к этапу работы с симулятором будет необходимо добавить в него информацию о правилах работы с симулятором*).
2. В левом верхнем углу холста разместить кнопку - элемент **Button** с подложенной в элементе **Image** текстурой со знаком вопроса (*можно использовать текстуры для кнопок и в другие используемые в дальнейшем текстуры имеющиеся в шаблоне* ***Ustanovka\_test***), которая должна вызывать по событию **OnClick** во «всплывающем» текстовом окне справочную информацию по правилам работы с симулятором установки, а по событию «ухода» курсора мыши с кнопки **PointerExit** (*добавить для этого к кнопке компоненту* ***Event/Event Trigger***) – закрытие окна справки.
3. Разместить на холсте в нижней его части элемент **Text** *(«подложить» под него аналогично предыдущему элемент* ***Image*** *для создания соответствующего фона*) на почти всю ширину холста и высотой примерно в 3 строки текста для вывода в него текущей информации по ходу работы с симулятором установки.
4. Разместить на холсте в правом верхнем углу панель управления – элемент **Panel**, на котором разместить по вертикали список кнопок **Button** с заголовком «УСТАНОВКА» и названиями элементов установки, используемых при ее функционировании, а также кнопку «ПРАКТИКА» для запуска практической работы с установкой и подчинить все кнопки панели управления.
5. Последовательно для всех кнопок с названиями элементов установки запрограммировать по событию надвижения курсора мыши на кнопку **PointerEnter –** вызов во всплывающее текстовое окно в нижней части холста информации о назначении выбранного элемента установки.
6. Добавить для всех кнопок с названиями элементов установки по событию надвижения курсора мыши на кнопку **PointerEnter** в код вызываемой функцииеще иподсвечивание ярким цветом выбранного элемента установки, а по событию ухода курсора мыши с соответствующей кнопки **PointerExit** возврат к первоначальному цвету выбранного элемента установки.
7. Добавить для всех кнопок с названиями элементов установки по событию **OnClick** переход камеры в удобный ракурс для просмотра выбранного элемента установки.

***РЕКОМЕНДАЦИИ.*** *Для получения положения камеры в нужном ракурсе для просмотра соответсвующего элемента установки рекомендуется поступать следующим образом:*

1. *Запомнить значения позиции* ***Position*** *и поворота* ***Rotation*** *камеры в исходном положении.*
2. *Рассчитать (возможно вручную на отдельном чертеже-схеме) желаемую позицию и поворот камеры для удобного ракурса просмотра соответсвующего элемента установки – сдвиг рассчитывается как разность значений позиций XYZ между установкой и камерой, а поворот камеры на нужный угол определяется в конструкции кватерниона. Например, для просмотра элемента на обратной стороне установки требуемый угол поворота будет равен:*

***needRotaton = Quaternion.AngleAxis(-180, new Vector3(0,1,0));***

*а для просмотра сверху:*

***needRotaton = Quaternion.AngleAxis(0, new Vector3(0,1,0));***

1. *В скрипте для перемещения камеры в нужные ракурсы задать для каждой кнопки отдельные функции. Например, для просмотра обратной стороны установки:*

***public void Move() {  
     move = true;  
     startPosition = transform.position;  
     startRotation = transform.rotation;  
     needPosition = new Vector3(277.0f, 251, 6);  
     needRotaton = Quaternion.AngleAxis(-180, new Vector3(0,1,0));  }***

*здесь булевая переменная* ***move****используется для запуска движения, а стартовый позиция и угол поворота задаются каждый раз для исходного положения камеры перед ее перемещением.*