МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное   
учреждение высшего образования   
«Южный федеральный университет»

Институт математики, механики   
и компьютерных наук им. И. И. Воровича

Сидоренко Максим Павлович

РАЗРАБОТКА КОМПЛЕКСА УЧЕБНЫХ ПРОЕКТОВ ПО СОЗДАНИЮ ПРИЛОЖЕНИЙ С ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ НА ПЛАТФОРМЕ UNITY

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА  
по направлению подготовки  
02.04.02 – Фундаментальная информатика и информационные технологии  
направленность программы  
«Разработка игр и мобильных приложений»

**Научный руководитель** –   
доц., к. ф.-м. н. Абрамян Михаил Эдуардович

**Рецензент** –   
доц., к. т. н. Демяненко Яна Михайловна

Допущено к защите:  
руководитель   
образовательной программы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Демяненко Я.М.

Ростов-на-Дону – 2020

Оглавление

[Введение 3](#_Toc41780313)

[1. Постановка задачи 4](#_Toc41780314)

[2. Настройка среды разработки 5](#_Toc41780315)

[3. Обзор среды разработки 8](#_Toc41780316)

[3.1. Создание проекта 8](#_Toc41780317)

[3.2. Работа с редактором 10](#_Toc41780318)

[3.3. Создание скриптов 14](#_Toc41780319)

[3.4. Сборка проекта 17](#_Toc41780320)

[4. Проект 1: SystemInfo 21](#_Toc41780321)

[5. Проект 2: Events 24](#_Toc41780322)

[6. Проект 3: Calc 28](#_Toc41780323)

[7. Проект 4: Scenes 32](#_Toc41780324)

[8. Проект 5: Colors 38](#_Toc41780325)

[9. Проект 6: Clock 42](#_Toc41780326)

[10. Проект 7: Zoo 46](#_Toc41780327)

[Заключение 51](#_Toc41780328)

[Литература 52](#_Toc41780329)

[Приложение 53](#_Toc41780330)

# Введение

Платформа Unity, разработанная компанией Unity Technologies в настоящее время занимает лидирующие позиции в области разработки игр. Эту платформу можно использовать для создания двумерных, трехмерных приложений любого уровня сложности. По состоянию на 2018 год Unity использовалась для создания примерно половины мобильных игр.

Одним из основных преимуществ Unity является поддержка более 25 платформ, среди которых: Windows, Linux, Mac, Android, iOS. Это позволяет разрабатывать один проект под все основные мобильные и десктопные ОС.

Современные приложения часто нуждаются в поддержке широкого спектра различных разрешений экрана. Адаптивный дизайн является необходимым элементом кроссплатформенного приложения. Причиной этому служит большое количество различных устройств и платформ, которые отличаются разными разрешениями. Адаптивная версия должна работать на всех платформах с любым разрешением. Система создания интерфейсов в Unity снабжена рядом различных инструментов для адаптивного дизайна, которые также можно комбинировать между собой массой различных способов.

Создание Unity приложений имеет много особенностей. Разработка пользовательских интерфейсов на данной платформе сильно отличается от нативной разработки. По этой причине в данной работе поэтапно показывается процесс решения типичных задач, возникающих при создании графических интерфейсов. Некоторые примеры данной работы основаны на проектах, содержащихся в книгах [1-2].

В данной работе использованы материалы из источников [3–7]. Работа написана в стиле решения задач, использованном в [1-2].

# Постановка задачи

Задачей данной работы является разработка комплекса учебных проектов по созданию приложений с графическим интерфейсом на платформе Unity, а также сборка приложений под Android и Windows для демонстрации их работы. Набор должен содержать не менее 10 проектов, освещающих следующие темы:

* Настройка среды разработки
* Структура и основные элементы редактора
* Жизненный цикл приложения
* Обработка событий
* Простейшие визуальные элементы, графические элементы
* Приложения с несколькими экранами
* Перетаскивание объектов
* Работа с таймером
* Списки
* Хранение данных

Каждый проект должен быть снабжен подробным описанием процесса его разработки.

# Настройка среды разработки

Для начала необходимо скачать и установить Unity Hub. Ссылка для скачивания: <https://unity3d.com/get-unity/download>. Необходимо перейти по ссылке и нажать кнопку Download Unity Hub. Unity Hub позволяет управлять проектами, работать с различными версиями Unity и устанавливать дополнения для сборки под различные платформы.

Для работы с Unity Hub необходимо войти в аккаунт через почту, к которой привязан Unity ID и активировать лицензию:

1. В Unity Hub нажать на иконку пользователя в правом верхнем углу, далее Sign In (рис. 1).

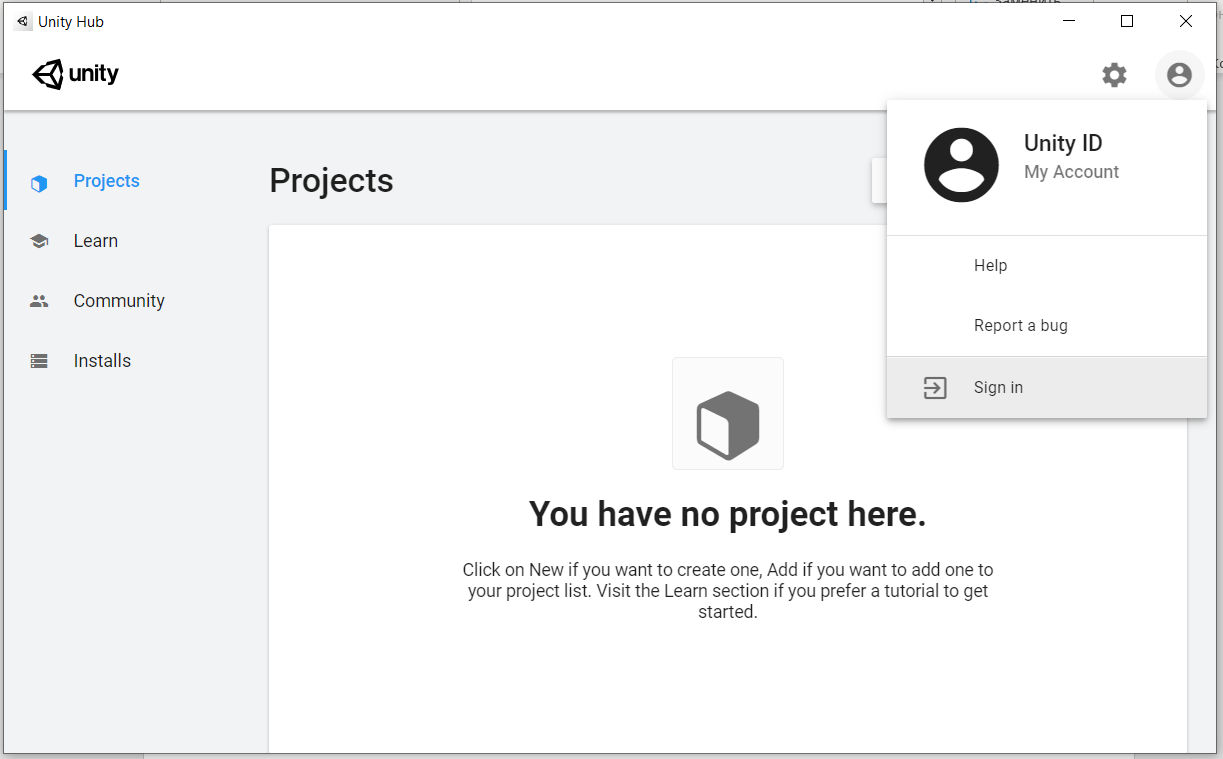


Рис. 1. Настройка Unity Hub, шаг 1

1. Если уже имеется почта с Unity ID, заполнить поля Email, Password, нажать Sign In (рис. 2). После этого вход будет выполнен, и можно сразу перейти к пункту 5. В противном случае, сначала нужно пройти регистрацию, нажав на ссылку create one (рис. 2), и перейти к пункту 3.

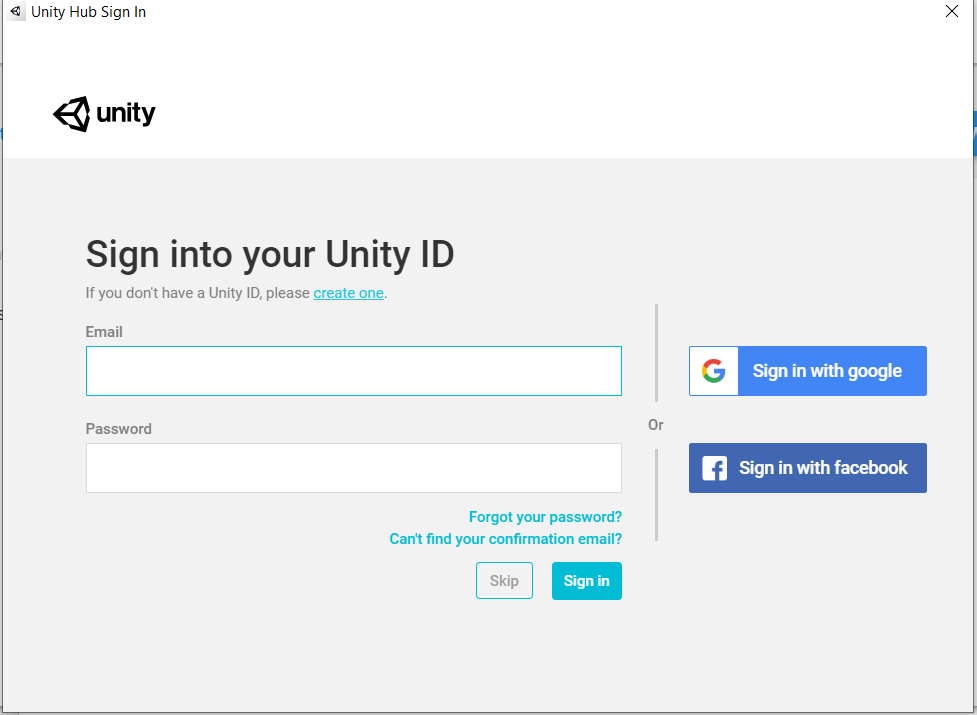


Рис.2. Настройка Unity Hub, шаг 2

1. Заполнить все необходимые поля, принять соглашение по использованию Unity, нажать кнопку Create a Unity ID (рис. 3).

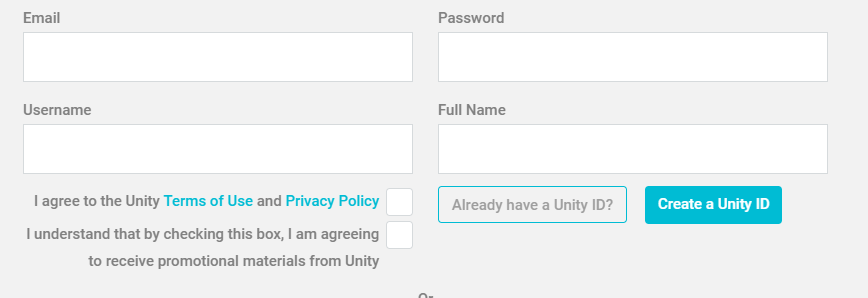


Рис.3. Настройка Unity Hub, шаг 3

1. Зайти на указанную на прошлом шаге почту и подтвердить её. После этого вернуться в Unity Hub и нажать Continue (рис. 4).

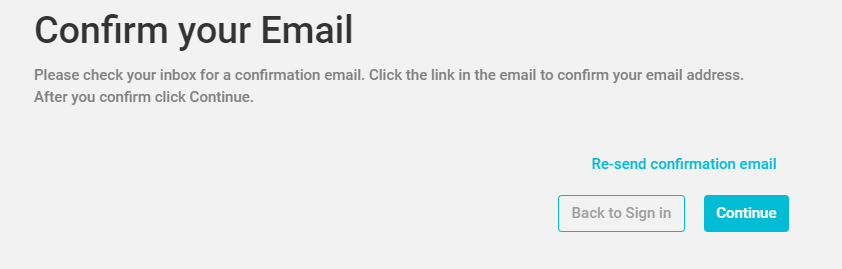


Рис.4. Настройка Unity Hub, шаг 4

1. Нажать на кнопку настройки в правом верхнем углу. Далее в левом боковом меню нажать на вкладку License Management, нажать на кнопку Activate New License (рис. 5).

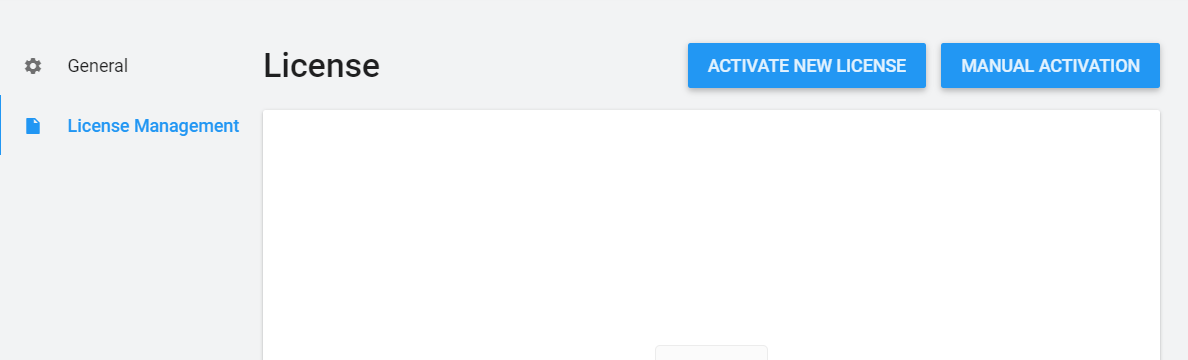


Рис.5. Настройка Unity Hub, шаг 5

1. Выбрать опции персональной лицензии (рис. 6), нажать кнопку Done. После этого в списке появится лицензия.

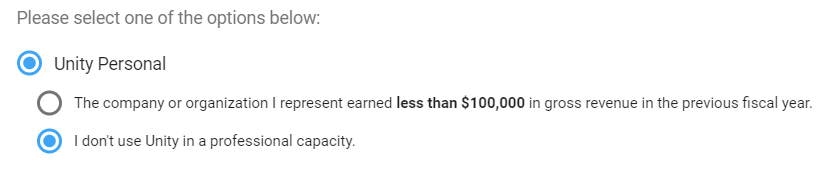


Рис.6. Настройка Unity Hub, шаг 6

1. Вернуться на главную страницу, в левом боковом меню нажать на вкладку Installs. Далее нажать на кнопку Add в правом верхнем углу (рис. 7).

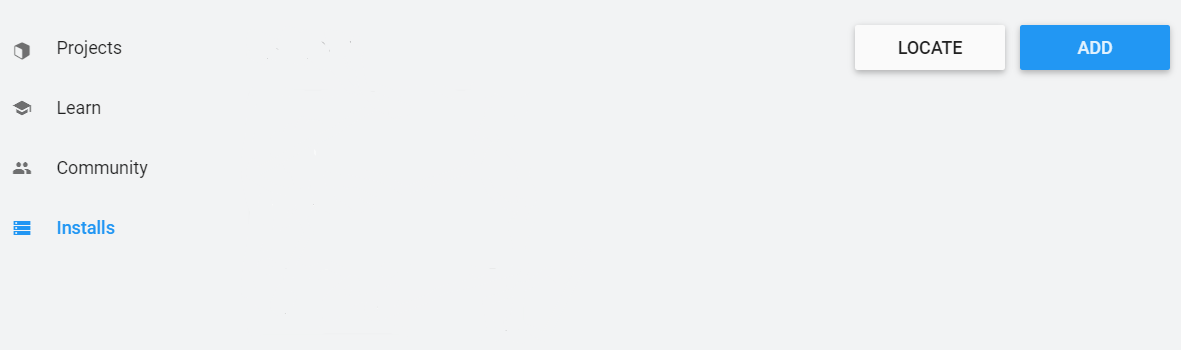


Рис.7. Настройка Unity Hub, шаг 7

1. На данном шаге происходит выбор версии Unity. Нужно оставить выбранной последнюю стабильную версию и нажать на кнопку Next (рис. 8).

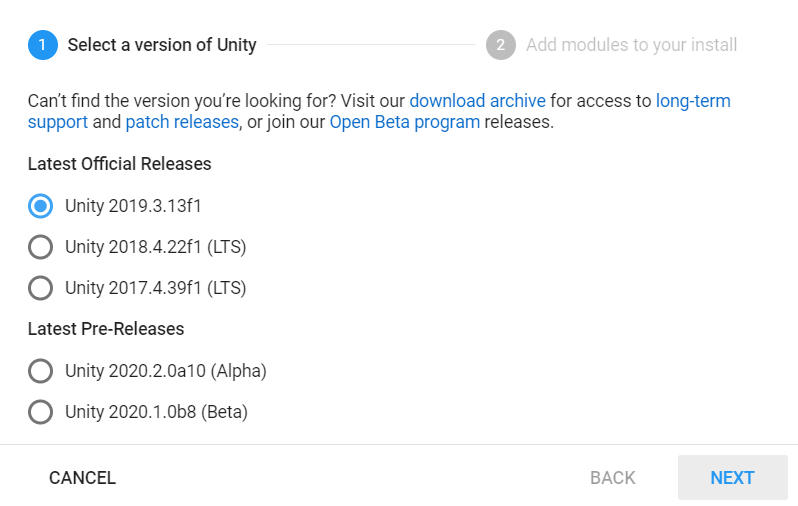


Рис.8. Настройка Unity Hub, шаг 8

1. Выбрать опцию Android Build Support для сборки под ОС Android (рис. 9) и нажать на кнопку Done. Дождаться окончания установки.

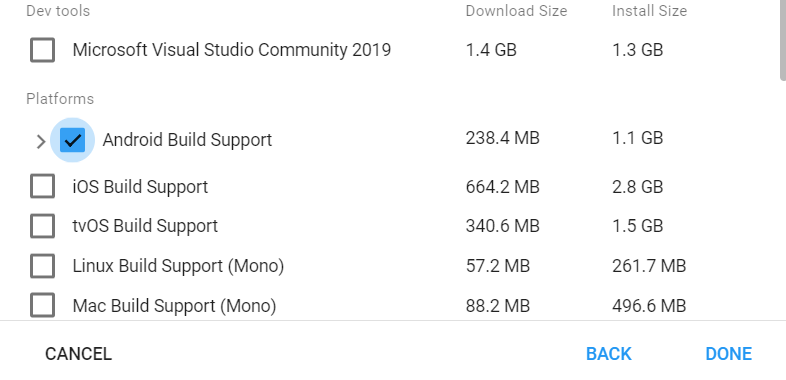


Рис.9. Настройка Unity Hub, шаг 9

# Обзор среды разработки

## Создание проекта

1. Зайти в Unity Hubs, во вкладке Projects нажать на кнопку New (рис. 10).

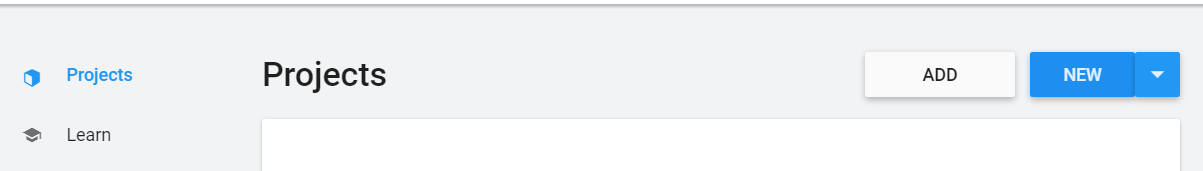


Рис.10. Создание проекта, шаг 1

1. Указать тип проекта 2D, написать название проекта, выбрать директорию расположения проекта и нажать на кнопку Create (рис. 11).

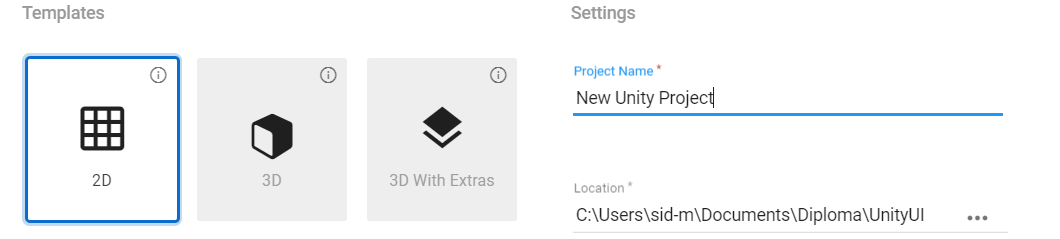


Рис.11. Создание проекта, шаг 2

1. В контекстном меню нажать File, далее Build Settings (рис. 12).

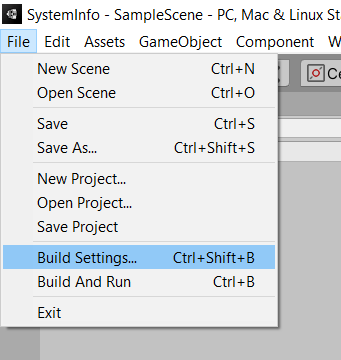


Рис.12. Создание проекта, шаг 3

1. Изменить текущую платформу на Android, нажать на кнопку Switch Platform (рис. 13). Этот шаг был выполнен с целью добавления большего количества разрешений экранов в предпросмотре редактора.

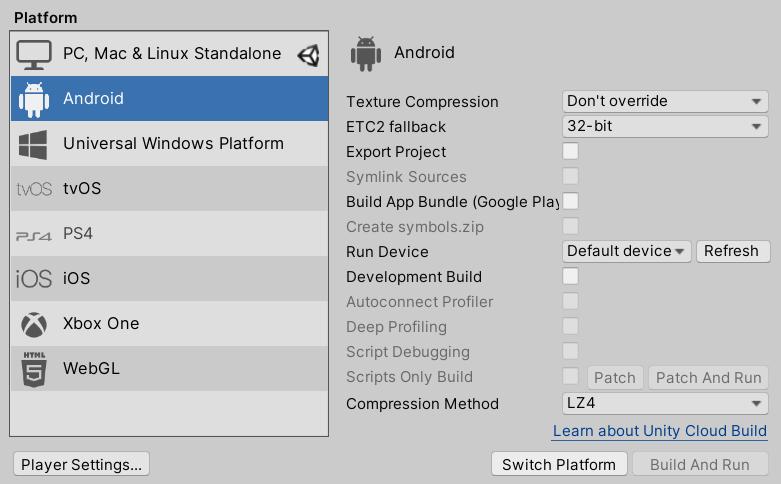


Рис.13. Создание проекта, шаг 4

1. Во вкладке Hierarchy выбрать объект Main Camera. Затем во вкладке Inspector нажать на поле с цветом напротив надписи Background - откроется модальное окно, указать значение цвета CCCCCC (рис. 14).

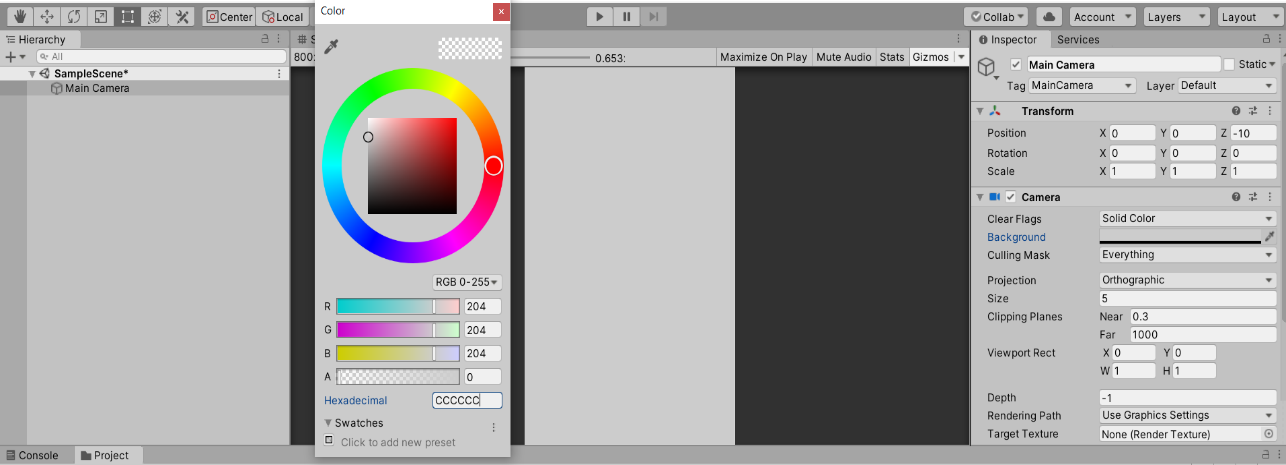


Рис.14. Создание проекта, шаг 5

1. Добавить Canvas на сцену: этот элемент является корневым, внутри его располагаются все остальные UI элементы. Нажать правой кнопкой по Main Camera в Hierarchy, далее в выпадающем меню выбрать UI и Canvas (рис. 15). Выделить Canvas в Hierarchy, потом в Inspector в разделе Canvas включить Pixel Perfect чекбокс (предотвращение размытия). Найти раздел Canvas Scaler, поставить для UI Scale Mode значение Scale With Screen Size (изменение размера в зависимости от размера экрана).

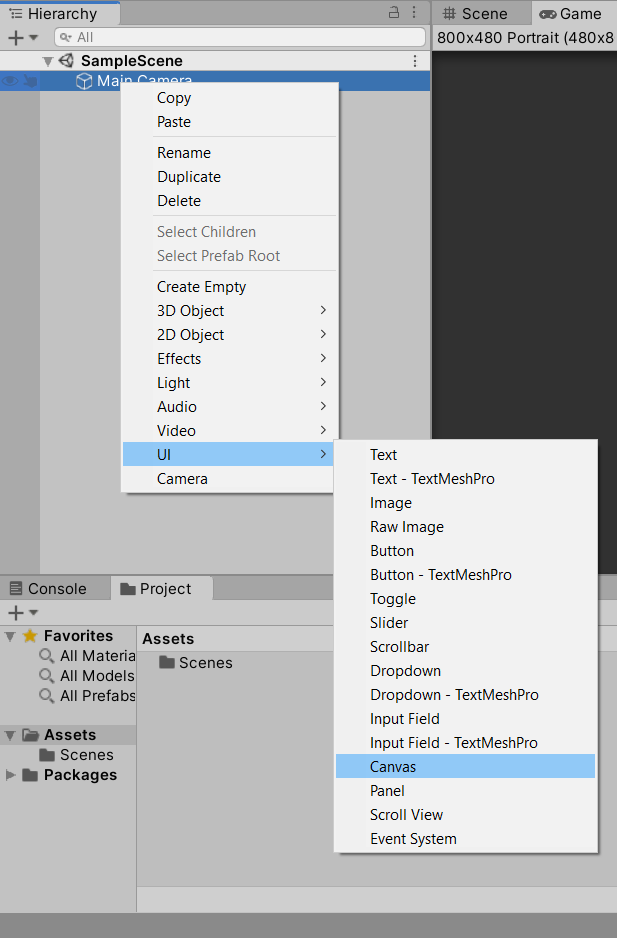


Рис.15. Создание проекта, шаг 6

1. В папке Assets создать директорию Scripts, для этого во вкладке Project нажать правой кнопкой по Assets, далее выбрать Create и Folder, изменить название папки на Scripts.
2. Добавить главный скрипт к элементу Canvas, для этого во вкладке Project нажать правой кнопкой по папке Scripts, далее выбрать Create и C# Script, переименовать скрипт в MainController. Во вкладке Hierarchy выбрать объект Canvas, потом во вкладке Inspector спуститься вниз, нажать кнопку Add Component, выбрать Scripts и MainController.

## Работа с редактором

Окно Game позволяет изменить текущее разрешение экрана в редакторе и запустить симуляцию работы при помощи кнопки старта (рис. *16*).

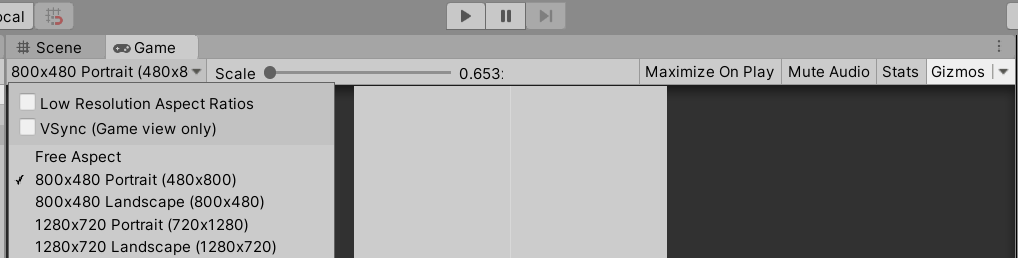


Рис.16. Окно Game

В окне Scene можно посмотреть текущее расположение UI элементов на сцене. Для отдаления и приближения используется колёсико мыши. Для перемещения по сцене необходимо выбирать инструмент Hand Tool в левом верхнем углу (рис. *17*).

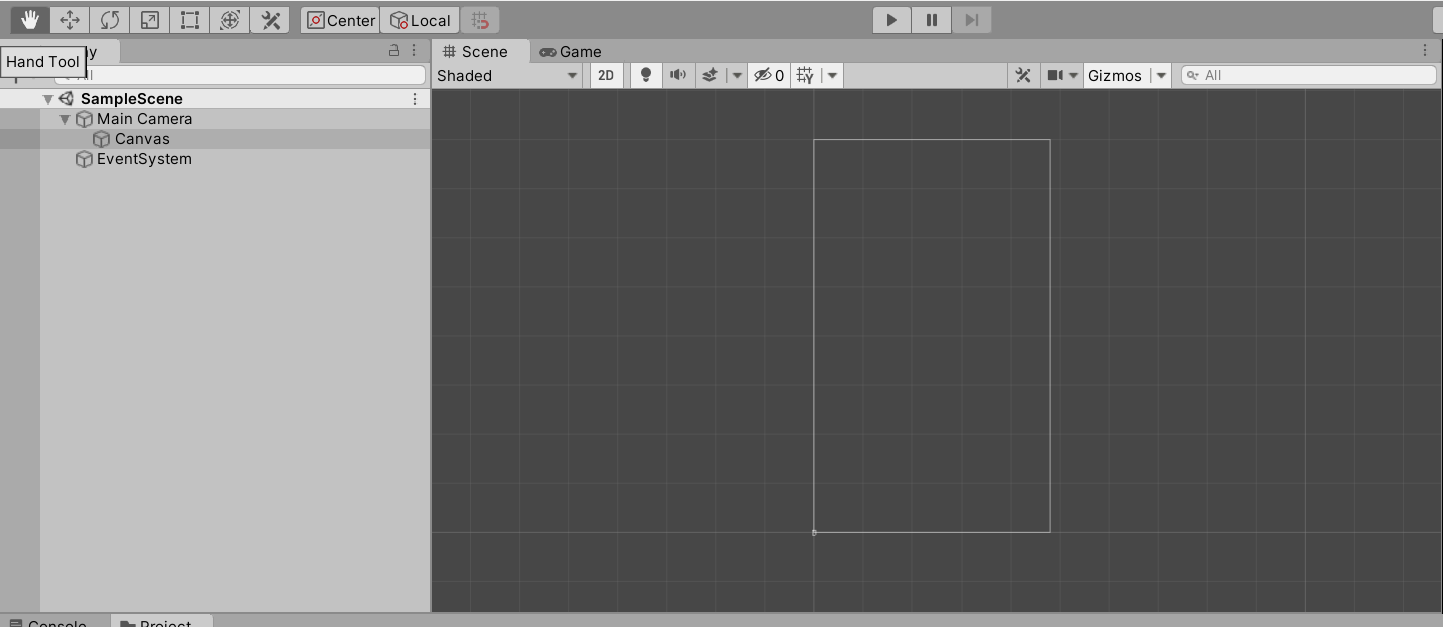


Рис.17. Окно Scene

Окно Hierarchy содержит список UI элементов внутри элемента Canvas. Можно менять порядок элементов, перетаскивая их вверх или вниз, менять вложенность, сделав их дочерними или родительскими. Элементы отображаются в том же порядке, в каком они расположены в иерархии. Для добавления нового элемента необходимо правой кнопкой нажать на родительский элемент, затем Create Empty, если необходимо добавить пустой элемент. Будет создан GameObject, у которого во вкладке Inspector отображаются настройки только для компонента Rect Transform. Rect Transform имеет значения положения, вращения и масштаба, а также ширину и высоту, используемые для указания размеров прямоугольной области элемента (рис. *18*). Для добавления UI элемента необходимо правой кнопкой нажать на родительский элемент, далее UI, затем выбрать необходимый элемент.

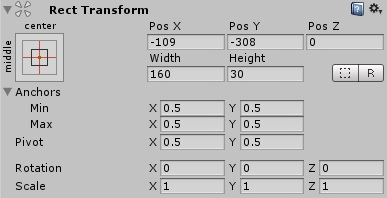


Рис.18. Компонент Rect Transform

Для дочерних элементов можно указывать, как он должен располагаться и изменяться относительно родительского элемента. Для этих целей существует компонент Anchor Presets, его настройку можно открыть, нажав на квадрат в левом углу компонента Rect Transform (рис. 19). Дочерний элемент можно привязать к сторонам или середине родительского элемента, а также растягивать его по ширине и высоте.



Рис.19. Компонент Anchor Presets

Для создания автоматической компоновки дочерних элементов используются Vertical и Horizontal Layout Group. Для их добавления необходимо выделить родительский элемент в Hierarchy, затем во вкладке Inspector спуститься вниз, нажать Add Component, выбрать Layout и необходимый компонент. Рассмотрим свойства компонента компоновки на примере Vertical Layout Group (рис. 20). Свойство Padding настраивает внутренние отступы родительского элемента. Свойство Spacing устанавливает расстояние между дочерними элементами. Свойство Child Alignment устанавливает выравнивание дочернего элемента, если он не заполняет все доступное пространство. Свойство Child Control Size отвечает за контроль высоты и ширины дочерних элементов. Свойство Child Force Expand расширяет дочерние элементы для заполнения свободного пространства.

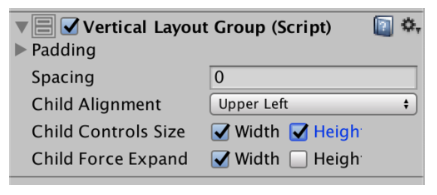


Рис.20. Компонент Layout Group

Для дополнительной настройки дочерних элементов используется компонент Layout Element. Добавляется в Inspector аналогично Layout Group для родительского элемента. Рассмотрим свойства компонента Layout Element (рис. 21). Свойства Min Width и Min Height – минимальная ширина и высота, которую должен иметь дочерний элемент. Свойства Preferred Width и Preferred Height – предпочтительная ширина и высота дочернего элемента. Свойства Flexible Width и Flexible Height – задание ширины и высоты по отношению к другим элементам в относительных единицах.

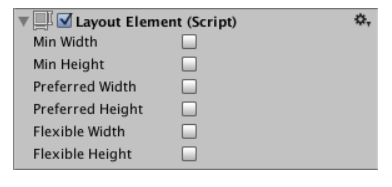


Рис.21. Компонент Layout Element

Окно Inspector позволяет просматривать и редактировать все свойства выбранного в данный момент элемента в Hierarchy. Через окно Project создаются новые скрипты, а в окне Console отображается вывод логов при запуске симуляции.

## Создание скриптов

В разделе 3.1 Создание проекта в пункте 8 был описан процесс создания главного скрипта MainController, добавляемого во все проекты. Скрипты хранятся в папке Assets/Scripts, созданной на 7 шаге описания по созданию проекта.

Добавление и редактирование скриптов осуществляется во вкладке Projects раздела Assets. Для добавления нового скрипта необходимо нажать правой кнопкой по папке Scripts, затем выбрать Create и C# Script, написать название скрипта.

Для того, чтобы скрипт начал работал, его необходимо добавить к объекту на сцене. Для того, чтобы добавить скрипт к объекту, нужно во вкладке Hierarchy выбрать необходимый элемент, потом во вкладке Inspector спуститься вниз и нажать кнопку Add Component, в выпадающем меню выбрать Scripts и скрипт по названию.

Рассмотрим жизненный цикл скрипта (рис. 22):



Рис.22. Жизненный цикл скрипта

Метод Awake вызывается только один раз за время существования скрипта и используется для инициализации переменных и состояний. Метод Awake будет вызван после того, как все объекты сцены будут созданы, поэтому в нём можно осуществлять безопасный поиск объектов на сцене. Метод Awake вызывается перед методом Start, сначала во всех скриптах отрабатывает Awake и только после этого произойдёт вызов Start.

Метод OnEnable вызывается, когда объект становится активным. Это происходит при создании сцены, если объект активен, а также при его активации.

Метод Start вызывается, перед первым вызовом Update, когда скрипт включён. Как и функция Awake, Start вызывается ровно один раз за время существования скрипта, однако Awake вызывается при инициализации объекта скрипта, независимо от того, включен ли скрипт. Функция Start также может быть использована для инициализации переменных. В случае необходимости взаимодействия с другими скриптами инициализация осуществляется в Awake, а в Start производится передача информации.

Метод Update вызывается при каждом обновлении кадра, когда скрипт включён. В нём производится обработка изменения состояний, например, проверяется нажатие клавиш на клавиатуре.

Метод OnDisable вызывается, когда объект отключается или становится неактивным.

Метод OnDestroy вызывается при разрушении объекта в последнем кадре, пока он ещё существует.

Рассмотрим два способа инициализации переменных для работы с объектами на сцене:

1. Первый способ - через редактор. Сначала нужно создать публичное поле необходимого типа в классе скрипта, затем после добавления скрипта к конкретному объекту, во вкладке Inspector появится раздел скрипта, в нём будет указано название переменной и возможность задать её значение (рис. 23). При нажатии на поле ввода откроется список с подходящими элементами, также значение можно установить перетаскиванием объекта из вкладки Hierarchy на поле ввода.

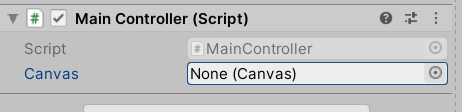


Рис.23. Задание значения переменной объекта

1. Второй способ - через код на примере инициализации переменной для Canvas. В классе скрипта создается приватное поле canvas типа Canvas, затем в методе Awake или Start осуществляется поиск по названию объекта. Название объекта задаётся во вкладке Inspector (рис. 24). Галочка перед названием отвечает за то, будет ли элемент активным при старте сцены.

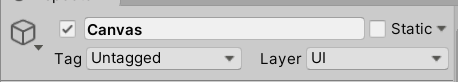


Рис.24. Указание названия игрового объекта

Для инициализации переменной canvas нужно написать следующий код (листинг 1):

Листинг 1. Инициализация переменной объекта через код

using UnityEngine;

public class MainController : MonoBehaviour {

Canvas canvas;

void Awake()

{

canvas = GameObject.Find("Canvas").GetComponent<Canvas>();

}

}

В методе Awake GameObject.Find находит GameObject по имени и возвращает его, а GameObject.GetComponet возвращает компонент заданного типа, если он прикреплен объекту и null, если нет.

Также рассмотрим создание обработчика нажатия на объект, к которому прикреплён скрипт на примере Canvas (листинг 2):

Листинг 2. Создание обработчика нажатия

using UnityEngine;

using UnityEngine.EventSystems;

public class MainController : MonoBehaviour, IPointerClickHandler

{

public void OnPointerClick(PointerEventData eventData)

{

Debug.Log("click work");

}

}

Подключается пространство имён UnityEngine.EventSystem для работы с событиями. Для обработки нажатия нужно реализовать интерфейс IPointerClickHandler. В методе OnPointerClick Debug.Log пишет сообщение в консоль во время игровой симуляции.

**Примечание.** Если написать такой код, то при нажатии на произвольное место на Canvas не будет срабатывать обработчик события. Он будет срабатывать только при нажатии на элементы вложенные в Canvas. Для того, чтобы обработчик работал при любом нажатии необходимо добавить на Canvas UI элемент Panel, который будет занимать все пространство на Canvas.

## Сборка проекта

Для сборки проекта необходимо выбрать целевую платформу и настроить параметры сборки приложения. Сначала нужно открыть окно Build Settings, в главного меню окна редактора выбрать File, далее Build Settings.

Для настройки параметров сборки в окне Build Settings кликнуть по кнопке Player Settings. В верхней части будут отображены основные настройки приложения общие для всех платформ (рис. 25). Здесь задается название организации и приложения, указывается версия и иконка приложения.

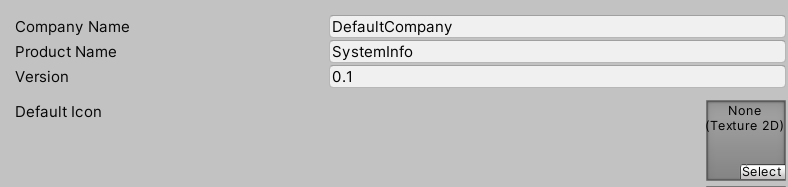


Рис.25. Основные настройки приложения

Рассмотрим некоторые наиболее важные настройки приложения для платформы Android. Для переключения между настройками платформ используется переключение между вкладками (рис. 26).

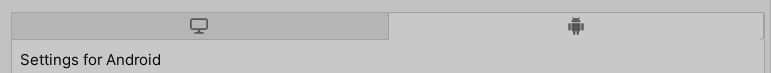


Рис.26. Переключение между настройками платформ

Рассмотрим настройки секции Resolution and Presentation для платформы Android (рис. 27). Включение свойства Start in fullscreen mode скрывает панель навигации. Свойство Render outside safe area разрешает рендеринг на всем доступном пространстве экрана, включая область выреза. Эту опцию необходимо выключать, чтобы на устройствах с вырезом нормально отображалась верхняя область приложения.

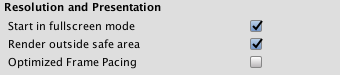


Рис.27. Resolution and Presentation платформа Android

Также в секции Resolution and Presentation для платформы Android присутствуют настройки ориентации экрана (рис. 28). Для свойства Default Orientation значением по умолчанию является Auto Rotation. При данной настройке ориентация экрана будет меняться автоматически в зависимости от положения устройства. Также существует возможность в качестве значения задать статическую ориентацию: Portrait, Portrait Upside Down, Landscape Left, Landscape Right. Если выбрано Auto Rotation, то с помощью настроек Allow Orientation for Auto Rotation можно запретить смену ориентации экрана для определенных значений.

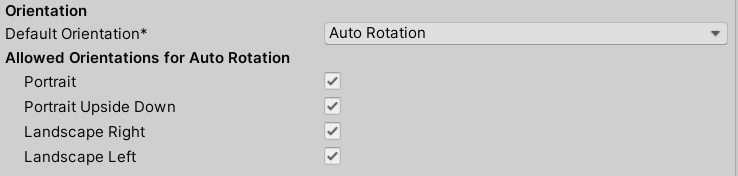


Рис.28. Настройки Orientation платформа Android

Теперь рассмотрим настройки секции Resolution and Presentation для платформы PC, Mac & Linux Standalone (рис. 29). Для этого в окне Player Settings необходимо переключиться на соответствующую вкладку (рис. 26). Свойство Fullscreen Mode определяет режим окна по умолчанию при запуске. Значение Fullscreen Window используется для полноэкранного режима приложения. Значение Windowedвключает оконный режим. Свойство Default Screen Width / Default Screen Height устанавливает начальную ширину / высоту окна, работает при свойстве Fullscreen Mode равном Windowed. При включении свойства Resizable Window пользователю разрешается изменять размер окна. Активация свойства Allow Fullscreen Switch разрешает пользовательское переключение между полноэкранным и оконным режимами.

Для сборок проектов под Windows устанавливаются значения свойств как на рис. 29. Изменяемые параметры: Fullscreen Mode – Windowed, Default Screen Width – 800, Default Screen Height – 600, Resizable Window – включено.

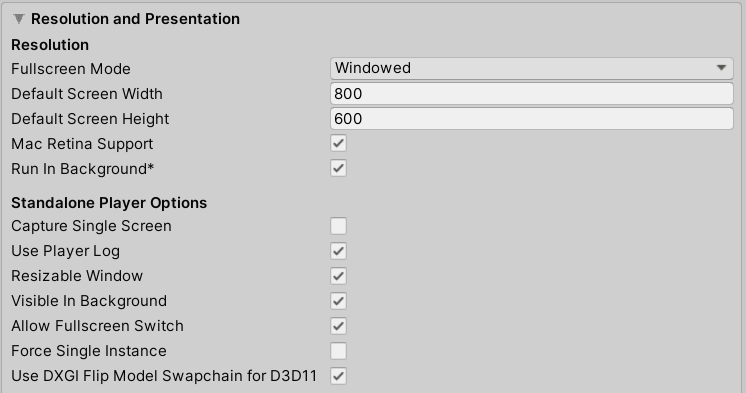


Рис.29. Resolution and Presentation платформа PC, Mac & Linux Standalone

В разделе 3.1 Создание проекта в пункте 4 осуществлялась смена текущей платформы на Android. Аналогично осуществляется смена платформы на PC, Mac & Linux Standalone. После выбора целевой платформы и настроек параметров сборки, для запуска процесса сборки в окне Build Settings есть две кнопки: Build и Build and Run. Build and Run помимо сборки осуществляет запуск приложения на подходящем устройстве. Для платформы PC, Mac & Linux Standalone после сборки автоматически откроется окно приложения. Для платформы Android необходимо либо подключить устройство по USB, либо запустить эмулятор.

Для подключения устройства по USB необходимо включить отладку по USB. Чтобы включить отладку по USB, необходимо включить режим разработчика. Для этого нужно найти номер сборки в меню настроек вашего устройства, нажать на номер сборки семь раз. После этого режим разработчика будет активирован. Перейти в Настройки > Параметры разработчика (Настройки > Система > Параметры разработчика) и поставить флажок Отладка по USB. Подключить устройство к компьютеру с помощью USB-кабеля. Для запуска эмулятора можно использовать AVD Manager Android Studio.

После нажатия на кнопку Build или Build and Run в папке с проектом создайте папку Build для осуществления сборки. Обратите внимание, что при сборке для платформы PC, Mac & Linux Standalone в корне проекта могут возникнуть ошибки, по этой причине была создана директория Build.

# Проект 1: SystemInfo

Темы, рассматриваемые в проекте:

* изменение свойств UI элемента;
* использование UI элемента Text;
* отображение информации об устройстве.

Выполнить все пункты раздела 3.1 Создание проекта. В пункте 2 дать проекту название SystemInfo.

Добавить элемент Text на сцену. Во вкладке Hierarchy правой кнопкой по Canvas, в выпадающем списке UI, далее Text. Во вкладке Inspector задать значения для компонента Rect Transform (рис. 30). После применения данных настроек элемент Text займет все пространство на сцене.

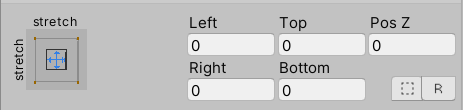


Рис.30. Значения Rect Transform элемента Text

Также необходимо задать настройки свойств элемента Text (рис. 31).

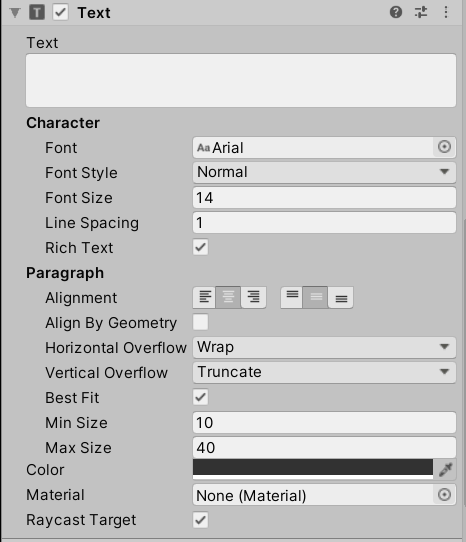


Рис.31. Значения свойств элемента Text

Рассмотрим описание свойств элемента Text (табл. 1):

|  |  |
| --- | --- |
| Text | Текст, отображаемый элементом. |
| Font | Шрифт, используемый для отображения текста. |
| Font Style | Стиль, примененный к тексту. Возможные варианты: Normal, Bold, Italic and Bold And Italic. |
| Font Size | Размер отображаемого текста. |
| Line Spacing | Вертикальное разделение между строками текста. |
| Rich Text | Использование элементов разметки в тексте |
| Alignment | Горизонтальное и вертикальное выравнивание. |
| Horizontal Overflow | Поведение при горизонтальном переполнении. Возможные варианты: Wrap и Overflow. |
| Vertical Overflow | Поведение при вертикальном переполнении. Возможные варианты: Wrap и Overflow. |
| Best Fit | Преобразовать размер текста для занятия всей отведенной области. |
| Color | Цвет, используемый для визуализации текста. |

Табл.1. Описание свойств элемента Text

Для отображения информации об устройстве нужно написать следующий код (листинг 3):

Листинг 3. SystemInfo, код MainController

using UnityEngine.UI;

public class MainController : MonoBehaviour

{

public Text label;

private void Awake()

{

label.text =

$"ОС: {SystemInfo.operatingSystemFamily} \n" +

$"Тип: {SystemInfo.deviceType} \n" +

$"Модель: {SystemInfo.deviceModel} \n" +

$"RAM: {SystemInfo.systemMemorySize}Mb\n" +

$"Батарея: {SystemInfo.batteryLevel \* 100}%\n" +

$"GPU: {SystemInfo.graphicsDeviceName} \n" +

$"CPU: {SystemInfo.processorType} \n";

}

}

В редакторе необходимо связать переменную label c элементом Text. Как это сделать, было описано в разделе 3.3 (первый способ - инициализация переменных для работы с объектами на сцене).

Окончательный вид приложения (рис. 32):



Рис.32. Вид приложения SystemInfo

# Проект 2: Events

Темы, рассматриваемые в проекте:

* создание обработчиков событий;
* использование UI элемента Button;
* добавление и удаление обработчиков событий.

Необходимо добавить на сцену две кнопки. Первая кнопка при нажатии либо меняет свою позицию, либо закрывает приложение в зависимости от состояния. Состояние переключается нажатием на вторую кнопку. При клике на вторую кнопку меняется её заголовок и обработчик нажатия на первую кнопку.

Выполнить все пункты раздела 3.1 Создание проекта. В пункте 2 дать проекту название Events. Выделить Canvas в Hierarchy, потом в Inspector в разделе Canvas Scaler установить Match в значение 0.5. Затем добавить две кнопки на Canvas: во вкладке Hierarchy правой кнопкой по Canvas, далее UI и Button. Отредактировать свойства первой кнопки, дать ей название Button1, в разделе Rect Transform поставить следующие значения: Pos X = 5, Pos Y = -5, Width = 120, Height = 40. Также изменить значение свойства Text для вложенного в Button1 элемента Text на «Приручить». Отредактировать свойства второй кнопки, дать ей название Button2, в разделе Rect Transform поставить следующие значения: Pos X = 140, Pos Y = -5, Width = 120, Height = 40. Также изменить значение свойства Text для вложенного в Button2 элемента Text на «Закрыть».

Создать следующие поля в классе MainController (листинг 4):

Листинг 4. Поля класса MainController

bool isWild = true;

Canvas canvas;

Button btn1;

Button btn2;

RectTransform btn2Tr;

RectTransform canvasTr;

Поле isWild используется для определения того, будет ли кнопка с заголовком «Закрыть» убегать от пользователя при нажатии или же завершать работу приложения. Поля canvas, btn1, btn2 связываются с элементами Canvas, Button1, Button2. А поля btn2Tr и canvasTr используются для определения размеров Button2 и Canvas.

В методе Awake инициализируются переменные для работы с Canvas, Button1, Button2, а также добавляются обработчики событий нажатия на кнопки (листинг 5).

Листинг 5. Events, метод Awake

void Awake()

{

canvas = GameObject.Find("Canvas").GetComponent<Canvas>();

btn1 = GameObject.Find("Button1").GetComponent<Button>();

btn2 = GameObject.Find("Button2").GetComponent<Button>();

btn2Tr = btn2.GetComponent<RectTransform>();

canvasTr = canvas.GetComponent<RectTransform>();

btn1.onClick.AddListener(ChangeState);

btn2.onClick.AddListener(WildClick);

}

При нажатии на первую кнопку срабатывает метод ChangeState (листинг 6). В зависимости от состояния определяемого переменной логического типа isWild происходит смена обработчика события второй кнопки, также меняется заголовок первой кнопки. Для получения доступа к вложенному в Button1 элементу Text используется метод GetComponentInChildren.

Листинг 6. Events, метод ChangeState

public void ChangeState ()

{

Debug.Log("ChangeState");

if (isWild)

{

btn2.onClick.RemoveListener(WildClick);

btn2.onClick.AddListener(QuitApplication);

}

else

{

btn2.onClick.RemoveListener(QuitApplication);

btn2.onClick.AddListener(WildClick);

}

Text labelBtn1 = btn1.GetComponentInChildren<Text>();

labelBtn1.text = isWild ? "Сброс" : "Приручить";

isWild = !isWild;

}

В листинге 7 представлены методы WildClick и QuitApplication, используемые в качестве обработчиков нажатия на вторую кнопку. В методе WildClick вычисляются случайные координаты для кнопки в пределах Canvas с учётом размера Button2, а в методе QuitApplication вызывает функция останова приложения.

Листинг 7. Events, методы WildClick и QuitApplication

void WildClick()

{

Debug.Log("WildClick");

float xCoeff = canvasTr.rect.width / 2 - btn2Tr.rect.width;

float x = Random.Range(-1f, 1f) \* xCoeff;

float yCoeff = canvasTr.rect.height / 2 - btn2Tr.rect.height;

float y = Random.Range(-1f, 1f) \* yCoeff;

btn2.transform.localPosition = new Vector3(x, y, 0);

}

void QuitApplication()

{

Debug.Log("QuitApplication");

Application.Quit();

}

**Ошибка.** При запуске приложения на Windows и изменении размера окна, а также смене ориентации экрана на Android, могла возникнуть ситуация, когда вторая кнопка покидает границы сцены.

**Исправление.** Для решения данной проблемы в скрипт MainController был добавлен обработчик на изменение размеров Canvas, в котором Button2 перемещается в центр сцены (листинг 7).

Листинг 7. Events метод OnRectTransformDimensionsChange

void OnRectTransformDimensionsChange()

{

if (btn2Tr == null) return;

float x = 0 - btn2Tr.rect.width / 2;

float y = 0 - btn2Tr.rect.height / 2;

btn2.transform.localPosition = new Vector3(x, y, 0);

}

Окончательный вид приложения (рис. 33):

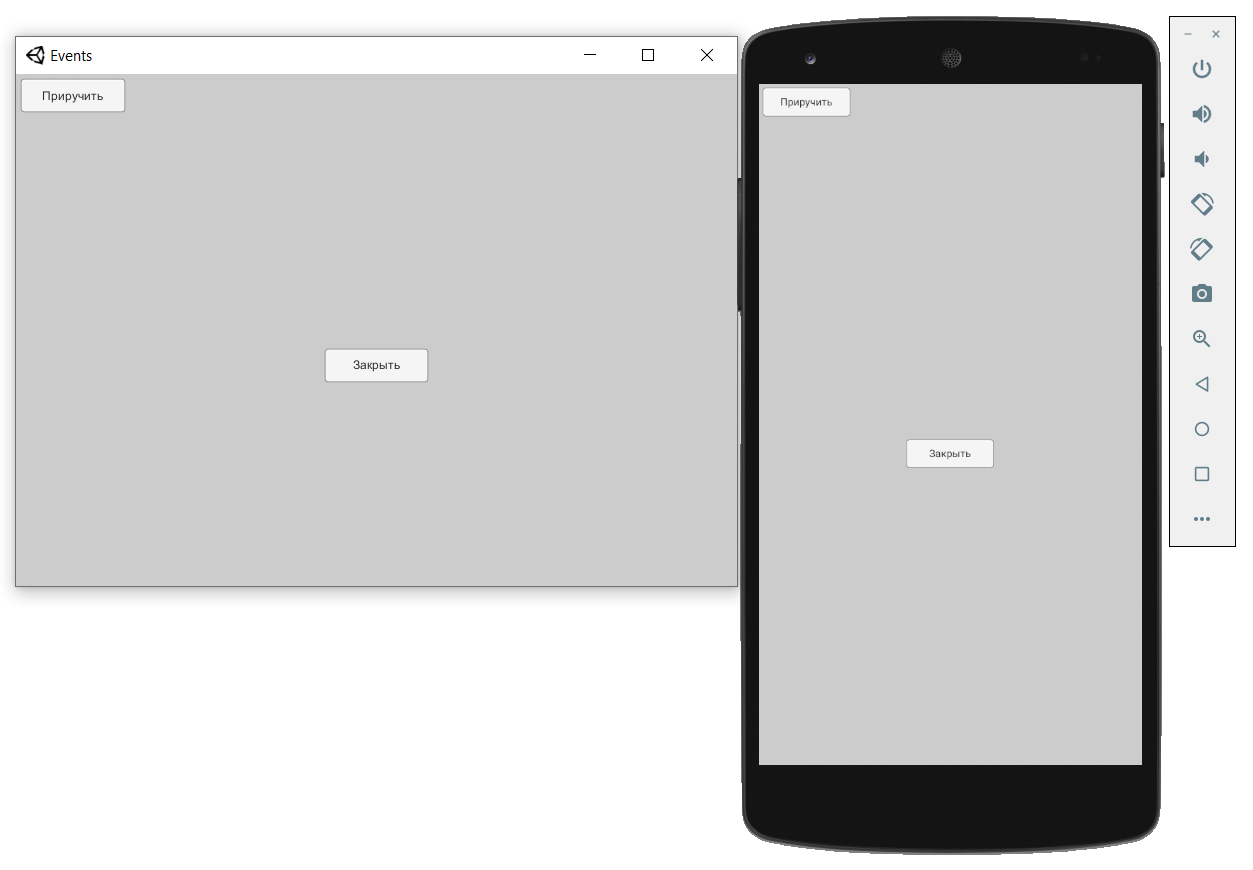


Рис.33. Вид приложения Events

# Проект 3: Calc

Темы, рассматриваемые в проекте:

* использование UI элемента InputField;
* использование Horizontal Layout Group;
* обработка ввода с клавиатуры.

Требуется создать приложение калькулятор с двумя полями ввода, между которыми отображается текст применяемой операции. Ввод осуществляется с клавиатуры, изменить операцию и получить ответ также можно при помощи кнопок на сцене.

Выполнить все пункты раздела 3.1 Создание проекта. В пункте 2 дать проекту название Calc.

На Canvas добавить два пустых GameObject, которые будут выполнять роль строк для полей ввода и кнопок операций. Во вкладке Hierarchy правой кнопкой по Canvas, затем Create Empty. Во вкладке Inspector для только что созданного GameObject изменить название на «Display». Поставить Anchor Presets по горизонтали stretch, по вертикали top, Pos Y = -10, Height = 60. Добавить Horizontal Layout Group: для этого во вкладке Inspector спуститься вниз, нажать на кнопку Add Component, далее Layout и Horizontal Layout Group. Дать следующие настройки для свойства Padding: Left = 5, Right = 5, Top = 10, Bottom = 5. Значение для Child Alignment оставить по умолчанию Upper Left. Для автоматического расчёта высоты вложенных элементов установить следующие чекбоксы: Height для Control Child Size, Height для Child Force Expand. Значения других чекбоксов необходимо поставить в выключенное состояние. Во вкладке Hierarchy скопировать объект Display, во вкладке Inspector изменить название на «Controls». Затем изменить значение Pos Y на -80. Отредактировать у Horizontal Layout Group свойство Padding: Left = 0, Right = 5, Top = 5, Bottom = 10; поставить Spacing = 3, Child Alignment = Upper Right.

Добавить два поля ввода для чисел, текст операции и ответа внутрь объекта Display. Для этого во вкладке Hierarchy правой кнопкой нажать по Display, далее UI и Input Field, аналогично добавить элемент Text. Один раз скопировать InputField, затем два раза Text. Дать созданным объектам следующие названия: InputField1, OperatorText, InputField2, EqualText, AnswerText. Установить следующие значения Width по порядку: 160, 40, 160, 40, 190. Изменить значения Text для OperatorText на «+», для EqualText на «=», для AnswerText на пустую строку. Включить опцию Best Fit у всех текстовых элементов.

Добавить кнопки операций внутрь объекта Controls. Для этого во вкладке Hierarchy правой кнопкой нажать по Controls, далее UI и Button. Установить значение Width = 60 и четыре раза скопировать созданную кнопку. Дать кнопкам следующие названия: Button1, Button2, Button3, Button4, Button5. Для вложенных в Button1 - Button5 элементов Text отредактировать значения Text: «+», «-», «x», «/», «=».

В папке Scripts создать новый скрипт ButtonController и добавить его к элементам Button1 - Button5. В классе скрипта создать следующие поля: operatorLabel и answerLabel типа Text, input1 и input2 типа InputField. Инициализировать поля класса в методе Awake вторым способом, описанным в разделе 3.3. Имена для поиска объектов: OperatorText, AnswerText, InputField1, InputField2. Реализовать интерфейс IPointerClickHandler для обработки нажатия по аналогии с примером раздела 3.3 (листинг 8):

Листинг 8. Calc, ButtonController, OnPointerClick

public void OnPointerClick(PointerEventData eventData)

{

Text label = GetComponentInChildren<Text>();

Debug.Log("operator click " + label.text);

if (label.text == "=")

{

Debug.Log("onAnswerClick");

try

{

double answer = 0;

double num1 = double.Parse(input1.text);

double num2 = double.Parse(input2.text);

string op = operatorLabel.text;

if (op == "+") answer = num1 + num2;

else if (op == "-") answer = num1 - num2;

else if (op == "x") answer = num1 \* num2;

else if (op == "/") answer = num1 / num2;

answerLabel.text = answer.ToString();

}

catch

{

Debug.Log("Parse Error");

answerLabel.text = "Error";

}

} else operatorLabel.text = label.text;

}

При нажатии на кнопки операций изменяется текст надписи, а при клике на кнопку получения ответа берутся значения полей ввода, производится необходимая математическая операция и выводится ответ.

Для обработки нажатия кнопок операции и получения ответа на десктопной версии в классе MainController создать следующие поля: operatorLabel типа Text, eqButton типа Button. Инициализировать поля класса в методе Awake вторым способом, описанным в разделе 3.3. Имена для поиска объектов: OperatorText, Button5. Написать следующий код в методе Update (листинг 9):

Листинг 9. Calc, MainController, Update

void Update()

{

if (Input.GetKeyDown(KeyCode.Return) ||

Input.GetKeyDown(KeyCode.KeypadEnter) ||

Input.GetKeyDown(KeyCode.Equals) ||

Input.GetKeyDown(KeyCode.KeypadEquals))

eqButton.OnPointerClick(null);

if (Input.GetKeyDown(KeyCode.Plus) ||

Input.GetKeyDown(KeyCode.KeypadPlus))

operatorLabel.text = "+";

if (Input.GetKeyDown(KeyCode.Minus) ||

Input.GetKeyDown(KeyCode.KeypadMinus))

operatorLabel.text = "-";

if (Input.GetKeyDown(KeyCode.KeypadMultiply))

operatorLabel.text = "x";

if (Input.GetKeyDown(KeyCode.KeypadDivide))

operatorLabel.text = "/";

}

**Недочёт.** Сейчас у пользователя существует возможность случайно ввести в поле ввода нечисловое значение. Также для более удобной работы с клавиатурой на десктопной версии можно добавить фокусировку полей ввода при нажатии на Tab.

**Исправление.** Для исправления данных недочётов был создан скрипт InputController и добавлен к полям ввода. В нём осуществляется валидация ввода и происходит смена фокусировки при нажатии на Tab (листинг 10).

Листинг 10. Calc, InputController

InputField input;

void Awake()

{

input = GetComponent<InputField>();

input.onValidateInput += CharValidate;

}

char CharValidate(string input, int charIndex, char c)

{

string newVal = input.Insert(charIndex, c.ToString());

double res;

bool isValid = double.TryParse(newVal, out res);

return isValid ? c : '\0';

}

void Update()

{

if (Input.GetKeyDown(KeyCode.Tab) && !input.isFocused)

{

input.ActivateInputField();

}

Окончательный вид приложения (рис. 34):

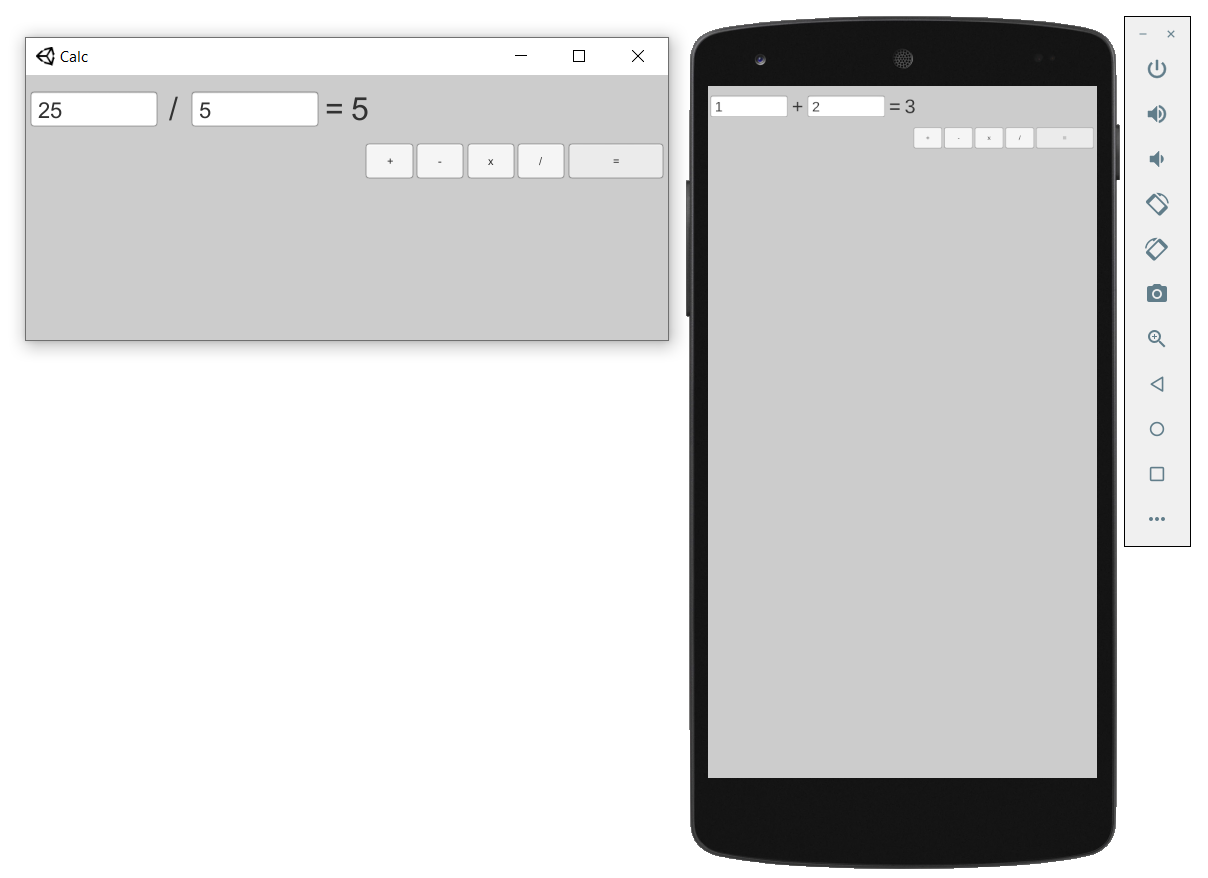


Рис.34. Вид приложения Calc

# Проект 4: Scenes

Темы, рассматриваемые в проекте:

* создание нескольких сцен;
* переход между сценами;
* обмен данными между сценами.

Требуется создать приложение с тремя сценами. На главной сцене отображается её заголовок и кнопки перехода на подчинённую сцену и сцену с настройками. Для подчинённой сцены производится подсчёт количества переходов, на ней располагается кнопка возврата на главную сцену и её заголовок. Сцена с настройками предназначена для изменения заголовков главной и подчинённой сцен.

Выполнить все пункты раздела 3.1 Создание проекта. В пункте 2 дать проекту название Scenes. Выделить Canvas в Hierarchy, потом в Inspector в разделе Canvas Scaler установить Match в значение 0.5.

Во вкладке Projects, Assets, Scenes переименовать файл сцены на «MainScene». Добавить ещё две сцены, нажать правой кнопкой по Scenes, затем Create и Scene, повторить два раза. Дать сценам следующие названия: «OtherScene», «SettingsScene». Для переключения между сценами нужно два раза кликнуть по сцене.

Выбрать MainScene, для отображения заголовка в Canvas добавить элемент Text, переименовать его в «MainTitle», установить Anchor Presets по горизонтали ­–­­ center, по вертикали – top, задать Pos X = 0, Pos Y = -5, Width = 300, Height = 60. Для компонента Text поставить Alignment по центру для горизонтали и вертикали, а также включить Best Fit. В Canvas добавить элемент Button, переименовать его в «ButtonNext», изменить заголовок кнопки на «Перейти к следующей сцене». Установить Anchor Presets по горизонтали ­–­­ center, по вертикали – middle, задать Pos X = 0, Pos Y = 25, Width = 300, Height = 60. Скопировать ButtonNext, переименовать скопированный элемент на «ButtonSettings», изменить Pos Y на 45, а заголовок кнопки на «Открыть окно настроек». Итоговый вид сцены представлен на рис. 35:

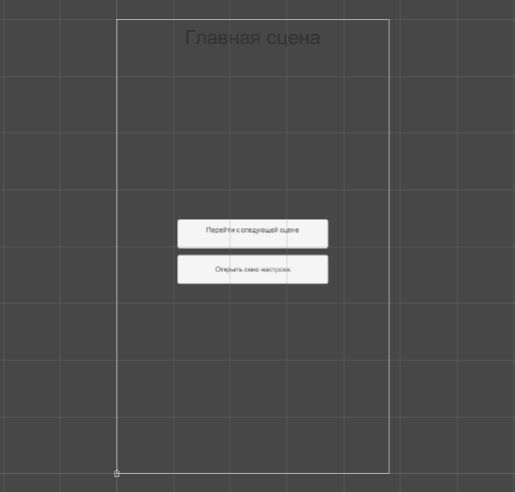


Рис.35. Scenes, MainScene

В папку Scripts добавить скрипт ScenesStore, написать в нём следующий код (листинг 11):

Листинг 11. Scenes, ScenesStore

public static class ScenesStore {

public static string mainSceneTitle = "Главная сцена";

public static string otherSceneTitle = "Следующая сцена";

}

Данный статический класс используется для хранения текущих значений заголовков главной и подчинённой сцен.

В методе Awake класса MainController нужно написать следующий код (листинг 12):

Листинг 12. Scenes, MainController, код метода Awake

Text mainTitle = GameObject.Find("MainTitle").GetComponent<Text>();

mainTitle.text = ScenesStore.mainSceneTitle;

Button nextBtn = GameObject.Find("ButtonNext").GetComponent<Button>();

nextBtn.onClick.AddListener(delegate {

SceneManager.LoadScene("OtherScene", LoadSceneMode.Single);

});

Button settingsBtn =

GameOject.Find("ButtonSettings").GetComponent<Button>();

settingsBtn.onClick.AddListener(delegate {

SceneManager.LoadScene("SettingsScene", LoadSceneMode.Single);

});

Происходит установка заголовка значением из ScenesStore, а также добавляются обработчики нажатия на кнопки. Для перехода на другую сцену по её названию используется метод SceneManager.LoadScene.

Выбрать сцену с названием OtherScene, создать элемент Canvas. Внутрь элемента Canvas добавить:

* Button с названием «ButtonBack», заголовком «Назад», Anchor Presets по горизонтали ­–­­ left, по вертикали – top; Pos X = 10, Pos Y = -10, Width = 150, Height = 40.
* Text с названием «TextCount», заголовком «Количество переходов на данную сцену равно», Anchor Presets по горизонтали ­–­­ center, по вертикали – middle; Pos X = 0, Pos Y = 0, Width = 400, Height = 60; Alignment по горизонтали – center, по вертикали – top; включенным BestFit.
* Text с названием «OtherTitle», заголовком «Следующая сцена», Anchor Presets по горизонтали ­–­­ right, по вертикали – top; Pos X = -10, Pos Y = -15, Width = 300, Height = 30; Alignment по горизонтали – right, по вертикали – center; включенным BestFit.

Итоговый вид сцены представлен на рис. 36:

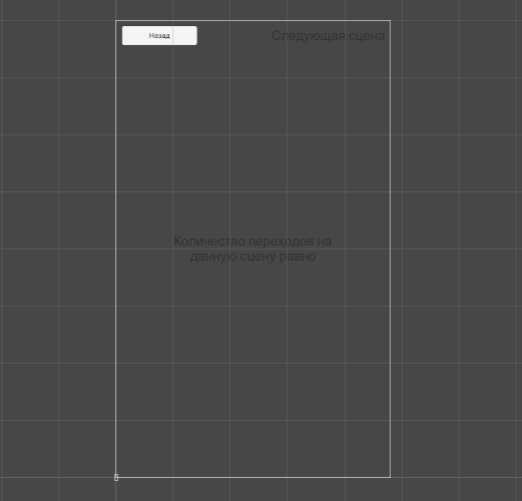


Рис.36. Scenes, OtherScene

В папку Scripts добавить скрипт OtherController, добавить его к Canvas OtherScene, написать в нём следующий код (листинг 13):

Листинг 13. Scenes, OtherController

private static int count = 0;

private void Awake()

{

Text textField = GameObject.Find("TextCount").GetComponent<Text>();

textField.text =

"Количество переходов на данную сцену равно " + ++count;

Text otherTitle =

GameObject.Find("OtherTitle").GetComponent<Text>();

otherTitle.text = ScenesStore.otherSceneTitle;

Button backBtn =

GameObject.Find("ButtonBack").GetComponent<Button>();

backBtn.onClick.AddListener(delegate {

SceneManager.LoadScene("MainScene", LoadSceneMode.Single);

});

}

Происходит подсчёт количества переходов на сцену через статическую переменную count, установка заголовка значением из ScenesStore, а также добавляется обработчик нажатия на кнопку.

Выбрать сцену с названием SettingsScene, создать элемент Canvas. Внутрь элемента Canvas добавить:

* Text с названием «Title», заголовком «Настройки», Anchor Presets по горизонтали ­–­­ center, по вертикали – top; Pos X = 0, Pos Y = -5, Width = 300, Height = 60; Alignment по горизонтали – center, по вертикали – center; включенным BestFit.
* Text с названием «Text1», заголовком «Заголовок1», Anchor Presets по горизонтали ­–­­ center, по вертикали – middle; Pos X = 0, Pos Y = 60, Width = 300, Height = 30; Alignment по горизонтали – left, по вертикали – top; выключенном BestFit.
* Input Field с названием «InputField1», Anchor Presets по горизонтали ­–­­ center, по вертикали – middle; Pos X = 0, Pos Y = 35, Width = 300, Height = 30.
* Text с названием «Text2», заголовком «Заголовок1», Anchor Presets по горизонтали ­–­­ center, по вертикали – middle; Pos X = 0, Pos Y = -15, Width = 300, Height = 30; Alignment по горизонтали – left, по вертикали – top; выключенном BestFit.
* Input Field с названием «InputField2», Anchor Presets по горизонтали ­–­­ center, по вертикали – middle; Pos X = 0, Pos Y = -40, Width = 300, Height = 30.
* Button с названием «SaveBtn», заголовком «Сохранить», Anchor Presets по горизонтали ­–­­ right, по вертикали – bottom; Pos X = -140, Pos Y = 5, Width = 120, Height = 30.
* Button с названием «CancelBtn», заголовком «Отмена», Anchor Presets по горизонтали ­–­­ right, по вертикали – bottom; Pos X = -10, Pos Y = 5, Width = 120, Height = 30.

Итоговый вид сцены представлен на рис. 37:

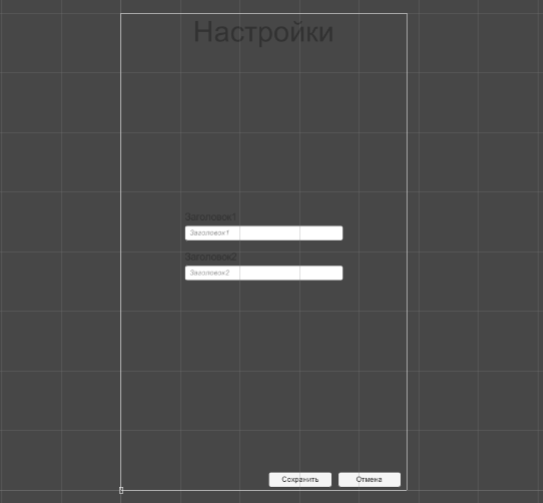


Рис.37. Scenes, SettingsScene

В папку Scripts добавить скрипт SettingsController, добавить его к Canvas SettingsScene, написать в нём следующий код (листинг 14):

Листинг 14. Scenes, SettingsController

void Awake()

{

var input1 =

GameObject.Find("InputField1").GetComponent<InputField>();

input1.text = ScenesStore.mainSceneTitle;

var input2 =

GameObject.Find("InputField2").GetComponent<InputField>();

input2.text = ScenesStore.otherSceneTitle;

Button saveBtn = GameObject.Find("SaveBtn").GetComponent<Button>();

saveBtn.onClick.AddListener(delegate {

ScenesStore.mainSceneTitle = input1.text;

ScenesStore.otherSceneTitle = input2.text;

SceneManager.LoadScene("MainScene", LoadSceneMode.Single);

});

Button cancelBtn =

GameObject.Find("CancelBtn").GetComponent<Button>();

cancelBtn.onClick.AddListener(delegate {

SceneManager.LoadScene("MainScene", LoadSceneMode.Single);

});

}

Значения полей ввода устанавливаются из ScenesStore. При нажатии на кнопку «Сохранить» в ScenesStore сохраняются новые значения для заголовков сцен. При нажатии на кнопку «Отмена» происходит переход на MainScene.

**Примечание.** При сборке приложения в окне BuildSettings необходимо добавить все сцены (рис. 38).

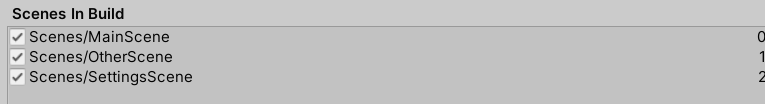
****

Рис.38. BuildSettings, добавление сцен

Окончательный вид приложения (рис. 39):

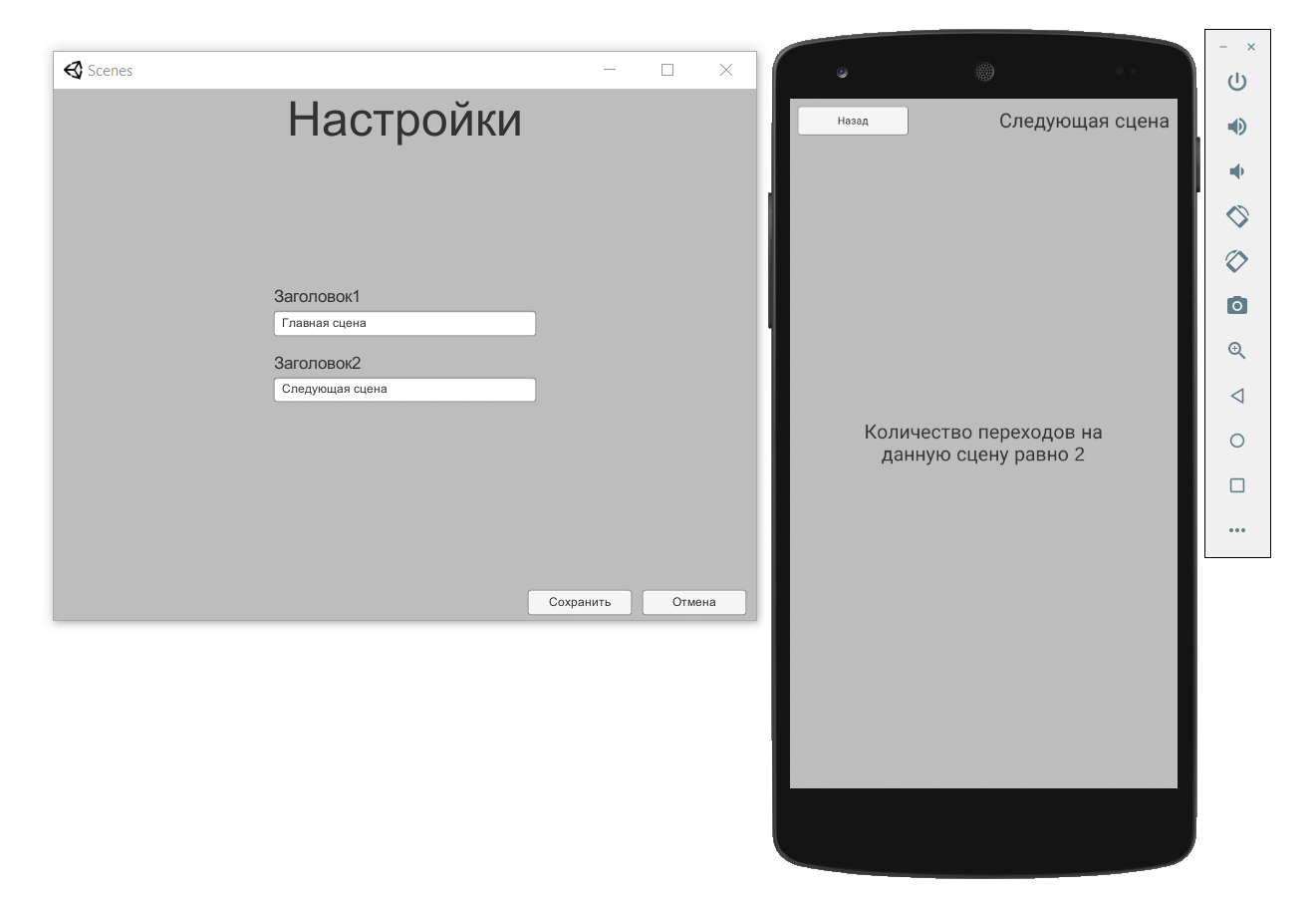


Рис.39. Вид приложения Scenes

# Проект 5: Colors

Темы, рассматриваемые в проекте:

* использование UI элемента Slider;
* работа с цветом на примере UI элемента Image;
* комбинирование Vertical и Horizontal Layout Group.

Требуется создать приложение для отображения цвета и его текстового значения. Значение цвета настраивается при помощи ползунков для красной, синей и зеленой состовляющих цвета. Также значение всех компонент можно одновременно изменять при помощи дополнительного ползунка.

Выполнить все пункты раздела 3.1 Создание проекта. В пункте 2 дать проекту название Colors.

На Canvas добавить пустой GameObject, изменить название на «Wrapper». Поставить Anchor Presets по горизонтали stretch, по вертикали stretch; свойствам Left, Top, Right, Bottom дать значение 0. Добавить Vertical Layout Group; для свойства Padding все значения поставить 10; Spacing = 0; Child Alignment = Upper Left; установить следующие чекбоксы: Width и Height для Control Child Size, Width для Child Force Expand; значения других чекбоксов поставить в выключенное состояние.

Внутрь Wrapper добавить пустой GameObject, изменить название на «SliderRow1». Добавить Horizontal Layout Group; для свойства Padding все значения поставить 0; Spacing = 0; Child Alignment = Middle Center; установить следующие чекбоксы: Width и Height для Child Force Expand; значения других чекбоксов поставить в выключенное состояние.

Внутрь SliderRow1 добавить:

* Text с названием «Text», заголовком «Alpha», Anchor Presets по горизонтали ­–­­ left, по вертикали – top; Width = 60, Height = 30; Font Size = 19; Alignment по горизонтали – center, по вертикали – center; выключенным BestFit.
* Slider с названием «SliderAlpha», Anchor Presets по горизонтали ­–­­ left, по вертикали – top; Width = 390, Height = 30.

Скопировать SliderRow1 четыре раза, изменить названия на SliderRow2 – SliderRow5; для элементов Text изменить заголовки на: «Red», «Green», «Blue», «Gray»; название элементов Slider изменить на: SliderRed, SliderGreen, SliderBlue, SliderGray. У элемента SliderAlpha поставить для Value значение 1.

Внутрь Wrapper добавить элемент Image, изменить название на «ColorRectangle», значение свойства Color поставить #000000. Добавить компонент Layout Element, включить Flexible Height, поставить значение 35.

Внутрь ColorRectangle добавить элемент Text с названием «ColorText», заголовком «000000», Anchor Presets по горизонтали ­–­­ center, по вертикали – middle; Width = 160, Height = 30; Alignment по горизонтали – center, по вертикали – center; включенным BestFit.

Создать следующие поля в классе MainController (листинг 15):

Листинг 15. Colors, поля класса MainController

Text colorText;

Image colorRectangle;

Slider sliderAlpha;

Slider sliderRed;

Slider sliderGreen;

Slider sliderBlue;

Slider sliderGray;

В методе Awake инициализируются поля класса, а также добавляются обработчики событий на изменения значений элементов Slider (листинг 16).

Листинг 16. Colors, метод Awake

void Awake()

{

colorText = GameObject.Find("ColorText").GetComponent<Text>();

colorRectangle =

GameObject.Find("ColorRectangle").GetComponent<Image>();

sliderAlpha =

GameObject.Find("SliderAlpha").GetComponent<Slider>();

sliderAlpha.onValueChanged.AddListener(onSliderValueChangeRGBA);

sliderRed = GameObject.Find("SliderRed").GetComponent<Slider>();

sliderRed.onValueChanged.AddListener(onSliderValueChangeRGBA);

sliderGreen =

GameObject.Find("SliderGreen").GetComponent<Slider>();

sliderGreen.onValueChanged.AddListener(onSliderValueChangeRGBA);

sliderBlue = GameObject.Find("SliderBlue").GetComponent<Slider>();

sliderBlue.onValueChanged.AddListener(onSliderValueChangeRGBA);

sliderGray = GameObject.Find("SliderGray").GetComponent<Slider>();

sliderGray.onValueChanged.AddListener(onSliderValueChangeGray);

}

В листинге 17 представлены методы onSliderValueChangeRGBA и onSliderValueChangeGray, используемые в качестве обработчиков изменения значений элементов Slider. При изменении значений ползунков меняется цвет элемента Image и текст подписи на нём, изменение SliderGray также изменяет значения других ползунков.

Листинг 17. Colors, обработчики изменения значений для Slider

void onSliderValueChangeRGBA(float value)

{

float r = sliderRed.value;

float g = sliderGreen.value;

float b = sliderBlue.value;

float a = sliderAlpha.value;

var newColor = new Color(r, g, b, a);

colorRectangle.color = newColor;

colorText.text = ColorUtility.ToHtmlStringRGBA(newColor);

}

void onSliderValueChangeGray(float value)

{

sliderRed.value = value;

sliderGreen.value = value;

sliderBlue.value = value;

var newColor = new Color(value, value, value, sliderAlpha.value);

colorRectangle.color = newColor;

colorText.text = ColorUtility.ToHtmlStringRGBA(newColor);

}

Окончательный вид приложения (рис. 40):

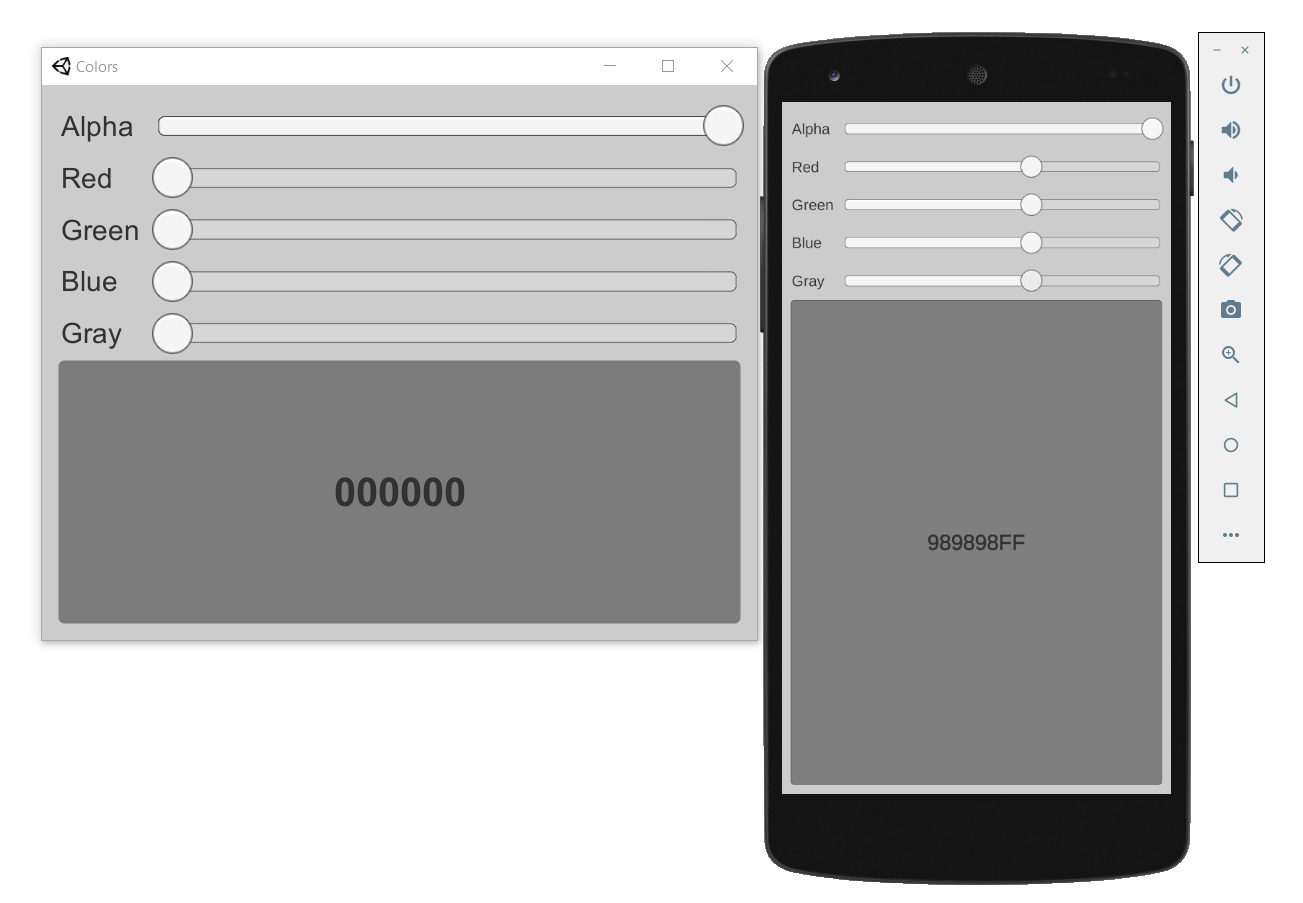


Рис.40. Вид приложения Colors

# Проект 6: Clock

Темы, рассматриваемые в проекте:

* отображение текущего времени;
* создание секундомера;
* использование UI элемента Toggle.

Необходимо отображать текущее время с возможностью переключения в режим секундомера. Также должны присутствовать следующие кнопки запуска/останова и сброса таймера. В режиме отображения времени данные кнопки должны быть неактивными.

Выполнить все пункты раздела 3.1 Создание проекта. В пункте 2 дать проекту название Clock. Выделить Canvas в Hierarchy, потом в Inspector в разделе Canvas Scaler установить Match в значение 0.5.

На Canvas добавить пустой GameObject, изменить название на «Display». Поставить Anchor Presets по горизонтали stretch, по вертикали middle; Left = 0, Right = 0, PosY = 50, Height = 400. Добавить Vertical Layout Group; для свойства Padding: Left = 5, Right = 5, Top = 0, Bottom = 0; Spacing = 0; Child Alignment = Upper Left; установить следующие чекбоксы: Width и Height для Control Child Size, Width и Height для Child Force Expand; значения других чекбоксов поставить в выключенное состояние. Внутрь Display добавить элемент Text с названием «TimeText», заголовком «00:00:00», Alignment по горизонтали – center, по вертикали – center; включенным BestFit; MaxSize = 150.

На Canvas добавить ещё один пустой GameObject, изменить название на «Controls ». Поставить Anchor Presets по горизонтали stretch, по вертикали middle; Left = 0, Right = 0, PosY = -100, Height = 100. Добавить Horizontal Layout Group; для свойства Padding: Left = 5, Right = 5, Top = 30, Bottom = 30; Spacing = 5; Child Alignment = Upper Center; все чекбоксы поставить в выключенное состояние.

Внутрь элемента Controls добавить:

* Toogle с названием «Toogle», заголовком «Timer»; Width = 100, Height = 40;
* Button с названием «TimerButton», заголовком «Start/Stop»; Width = 120, Height = 40; Pressed / Disabled Color = # C8C8C8; выключенным Interactable;
* Button с названием «ResetButton», заголовком «Reset»; Width = 120, Height = 40; Pressed / Disabled Color = # C8C8C8; выключенным Interactable.

Создать следующие поля в классе MainController (листинг 18):

Листинг 18. Clock, поля класса MainController

Text clockText;

Toggle checkbox;

Button toogleBtn;

Button resetBtn;

bool isRunTimer = false;

DateTime startTime;

DateTime? pauseTime = null;

TimeSpan pauseSpan;

Поле isRunTimer используется для определения состояния работы секундомера. Поля startTime, pauseTime хранят время начала работы и паузы секундомера. В поле pauseSpan накапливается время, в течении которого секундомер был на паузе.

В методе Awake, устанавливается русская локализация для работы со временем, инициализируются поля класса, а также добавляются обработчики событий нажатия на кнопки и изменения значения элемента Toggle (листинг 19).

Листинг 19. Clock, метод Awake

void Awake()

{

System.Threading.Thread.CurrentThread.CurrentCulture =

new System.Globalization.CultureInfo("ru-Ru");

clockText = GameObject.Find("TimeText").GetComponent<Text>();

checkbox = GameObject.Find("Toggle").GetComponent<Toggle>();

checkbox.onValueChanged.AddListener(onCheckBoxToogle);

toogleBtn = GameObject.Find("TimerButton").GetComponent<Button>();

toogleBtn.onClick.AddListener(ToogleTimer);

resetBtn = GameObject.Find("ResetButton").GetComponent<Button>();

resetBtn.onClick.AddListener(ResetTimer);

}

В листинге 20 представлены методы ToogleTimer и ResetTimer, используемые в качестве обработчиков нажатия на кнопки. В методе ToogleTimer происходит переключение работы секундомера и подсчитывается время проведённое на паузе. Метод ResetTimer обнуляет значение времени на паузе и обновляет время начала.

Листинг 20. Clock, обработчики нажатия на кнопки

void ToogleTimer()

{

isRunTimer = !isRunTimer;

if (isRunTimer && pauseTime.HasValue)

{

pauseSpan += DateTime.Now - pauseTime.Value;

pauseTime = null;

}

else if (!isRunTimer)

pauseTime = DateTime.Now;

}

void ResetTimer()

{

startTime = DateTime.Now;

pauseSpan = TimeSpan.Zero;

pauseTime = null;

clockText.text = "00:00";

}

В листинге 21 представлен метод onCheckBoxToogle, срабатывающий при изменении значения элемента Toogle. В нём, в зависимости от значения, меняется активность кнопок управления и запускается, либо останавливается секундомер.

Листинг 21. Clock, метод onCheckBoxToogle

void onCheckBoxToogle(bool val)

{

if (val)

{

toogleBtn.interactable = true;

resetBtn.interactable = true;

ResetTimer();

}

else

{

toogleBtn.interactable = false;

resetBtn.interactable = false;

isRunTimer = false;

pauseTime = null;

}

}

В методе Update, происходит изменение отображения времени в зависимости от выбранного режима. Время секундомера определяется как разница между текущим временем и временем начала работы секундомера минус время паузы (листинг 22).

Листинг 22. Clock, метод Update

void Update()

{

if (!checkbox.isOn)

clockText.text = System.DateTime.Now.ToLongTimeString();

else if (isRunTimer)

{

TimeSpan s = DateTime.Now - startTime - pauseSpan;

clockText.text = string.Format("{0}:{1}", s.Minutes \* 60 +

s.Seconds, "0" + s.Milliseconds.ToString()[0]);

}

}

Окончательный вид приложения (рис. 41):

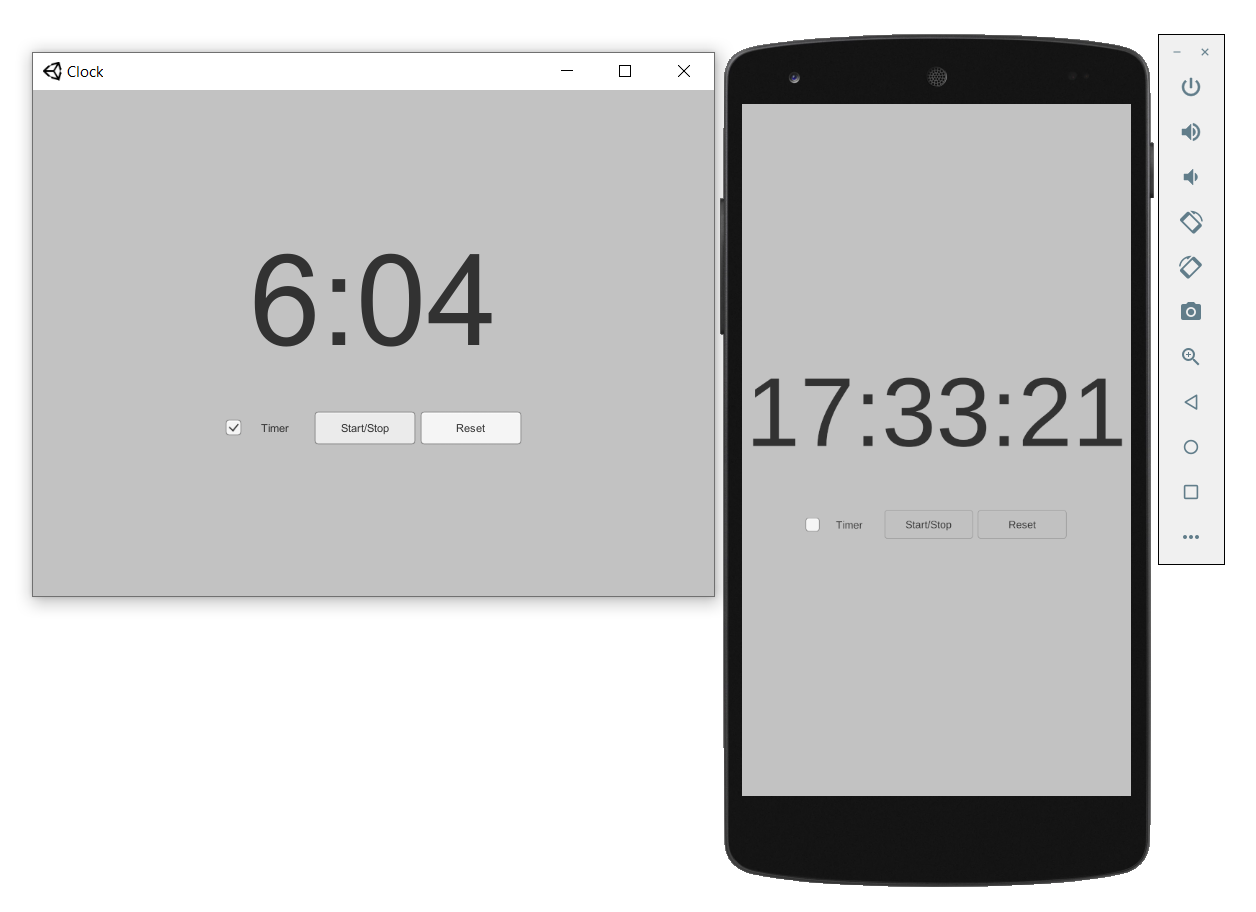


Рис.41. Вид приложения Clock

# Проект 7: Zoo

Темы, рассматриваемые в проекте:

* перетаскивание элементов;
* сброс состояния сцены;
* использование UI элемента Scroll View.

Требуется создать приложение с четырьмя метками различных животных и четырьмя областями, выступающих в роли приёмника. Метку можно свободно перемещать на пустую область сцены, при перемещении на другой элемент, не являющийся приёмником, происходит возврат в исходную позицию. При попадании метки на приёмник её дальнейшее перемещение запрещается. При перемещении другой метки на приёмник, с уже имеющейся меткой, одно из животных съедает другое.

Выполнить все пункты раздела 3.1 Создание проекта. В пункте 2 дать проекту название Zoo. Выделить Canvas в Hierarchy, потом в Inspector в разделе Canvas Scaler установить Match в значение 0.5.

На Canvas добавить ScrollView, для Scroll Rect отключить Horizontal, поставить Anchor Presets по горизонтали stretch, по вертикали stretch; Left = 0, Right = 0, Top = 0, Bottom = 0.

Для элемента Content, вложенного в ScrollView и Viewport:

* добавим элемент Text, c заголовком «Медведь», Anchor Presets по горизонтали ­–­­ center, по вертикали – top; PosX = 8.5, PosY= -50, Width = 160, Height = 30; Alignment по горизонтали – center, по вертикали – center; включенным BestFit.
* три раза скопируем элемент Text, изменим заголовки на: «Заяц», «Лисица», «Волк»; поставим PosY: -100, -150, -200.
* добавим элемент Image, Anchor Presets по горизонтали ­–­­ center, по вертикали – top; PosX = 8.5, PosY= -300, Width = 300, Height = 60.
* три раза скопируем элемент Image, изменим PosY: -400, -500, -600.
* Button с заголовком «Сброс» и включенным Best Fit для вложенного Text, Anchor Presets по горизонтали ­–­­ center, по вертикали – top; PosX = 8.5, PosY= -794, Width = 180, Height = 70.

Создать скрипт ResetBtnController и добавить его к элементу Button, реализовать интерфейс IPointerClickHandler для обработки нажатия (листинг 23). При нажатии на кнопку сброса будет заново загружаться текущая сцена.

Листинг 23. Zoo, ResetBtnController

using UnityEngine;

using UnityEngine.EventSystems;

using UnityEngine.SceneManagement;

public class ResetBtnController : MonoBehaviour, IPointerClickHandler

{

public void OnPointerClick(PointerEventData eventData)

{

Scene scene = SceneManager.GetActiveScene();

SceneManager.LoadScene(scene.name);

}

}

Создать скрипт DragController, написать следующий код (листинг 24):

Листинг 23. Zoo, DragController

using UnityEngine;

using UnityEngine.EventSystems;

public class DragController : MonoBehaviour,

IDragHandler, IEndDragHandler, IBeginDragHandler

{

public int powerPoints = 0;

CanvasGroup canvasGroup;

Transform rootParent;

void Awake()

{

rootParent = transform.parent;

canvasGroup = GetComponent<CanvasGroup>();

}

public void OnBeginDrag(PointerEventData eventData)

{

transform.SetParent(rootParent);

transform.SetAsLastSibling();

canvasGroup.alpha = 0.6f;

canvasGroup.blocksRaycasts = false;

}

public void OnDrag(PointerEventData eventData)

{

transform.position = eventData.position;

}

public void OnEndDrag(PointerEventData eventData)

{

canvasGroup.alpha = 1f;

canvasGroup.blocksRaycasts = true;

}

}

Создаётся публичное поле powerPoints (сила животного) для контроля поедания одного животного другим при попадании в одну клетку; rootParent для отображения метки поверх остальных элементов во время перетаскивания; canvasGroup для активации режима перетаскивания и смены прозрачности метки. Для работы перетаскивания были реализованы методы: OnBeginDrag, OnDrag и OnEndDrag.

Добавить скрипт DragController ко всем элементам Text, дополнительно добавить компонент Canvas Group. Проставить значения для свойства Power Points: «Заяц» = 0, «Лиса» = 1, «Волк» = 2, «Медведь» = 3.

**Недочёт.** Сейчас у пользователя существует возможность перемещать метки на другие элементы, не являющиеся приёмниками. Также необходимо предусмотреть возможность запрета перетаскивания.

**Исправление.** Для запрета такого поведения нужно изменить код DragController (листинг 24):

Листинг 24. Zoo, DragController, исправление

...

public class DragController : MonoBehaviour,

IDragHandler, IEndDragHandler, IBeginDragHandler

{

...

**public** **bool isAllowDrag = true;**

**Vector3 startPosition;**

**Transform startParent;**

...

public void OnBeginDrag(PointerEventData eventData)

{

**if (!isAllowDrag) return;**

...

**startParent = transform.parent;**

**startPosition = transform.position;**

}

public void OnDrag(PointerEventData eventData)

{

**if (!isAllowDrag) return;**

...

public void OnEndDrag(PointerEventData eventData)

{

...

**if (!isAllowDrag) return;**

**var dist = eventData.pointerEnter;**

**if (dist.name != "Viewport")**

**{**

**transform.SetParent(startParent);**

**transform.position = startPosition;**

**}**

}

}

Создать скрипт DropController, написать следующий код (листинг 25):

Листинг 25. Zoo, DropController

using UnityEngine;

using UnityEngine.EventSystems;

public class DropController : MonoBehaviour, IDropHandler

{

private void AddChild(GameObject obj, DragController targetScript)

{

var ancdPos = GetComponent<RectTransform>().anchoredPosition;

obj.GetComponent<RectTransform>().anchoredPosition = ancdPos;

obj.transform.SetParent(transform);

targetScript.isAllowDrag = false;

}

public void OnDrop(PointerEventData eventData)

{

GameObject target = eventData.pointerDrag;

var targetScript = target.GetComponent<DragController>();

if (!targetScript.isAllowDrag) return;

if (transform.childCount == 0)

AddChild(target, targetScript);

else

{

var child = transform.GetChild(0).gameObject;

var childController = child.GetComponent<DragController>();

if (childController.powerPoints >= targetScript.powerPoints)

{

Destroy(target);

} else

{

Destroy(child);

AddChild(target, targetScript);

}

}

}

}

При перемещении метки на приёмник, она становится его дочерним элементом. При добавлении последующих меток одна из них удаляется в зависимости от значений powerPoints. Запрет перетаскивания метки происходит через доступ к публичному полю isAllowDrag её скрипта DragController.

Добавить скрипт DropController ко всем элементам Image.

Окончательный вид приложения (рис. 42):

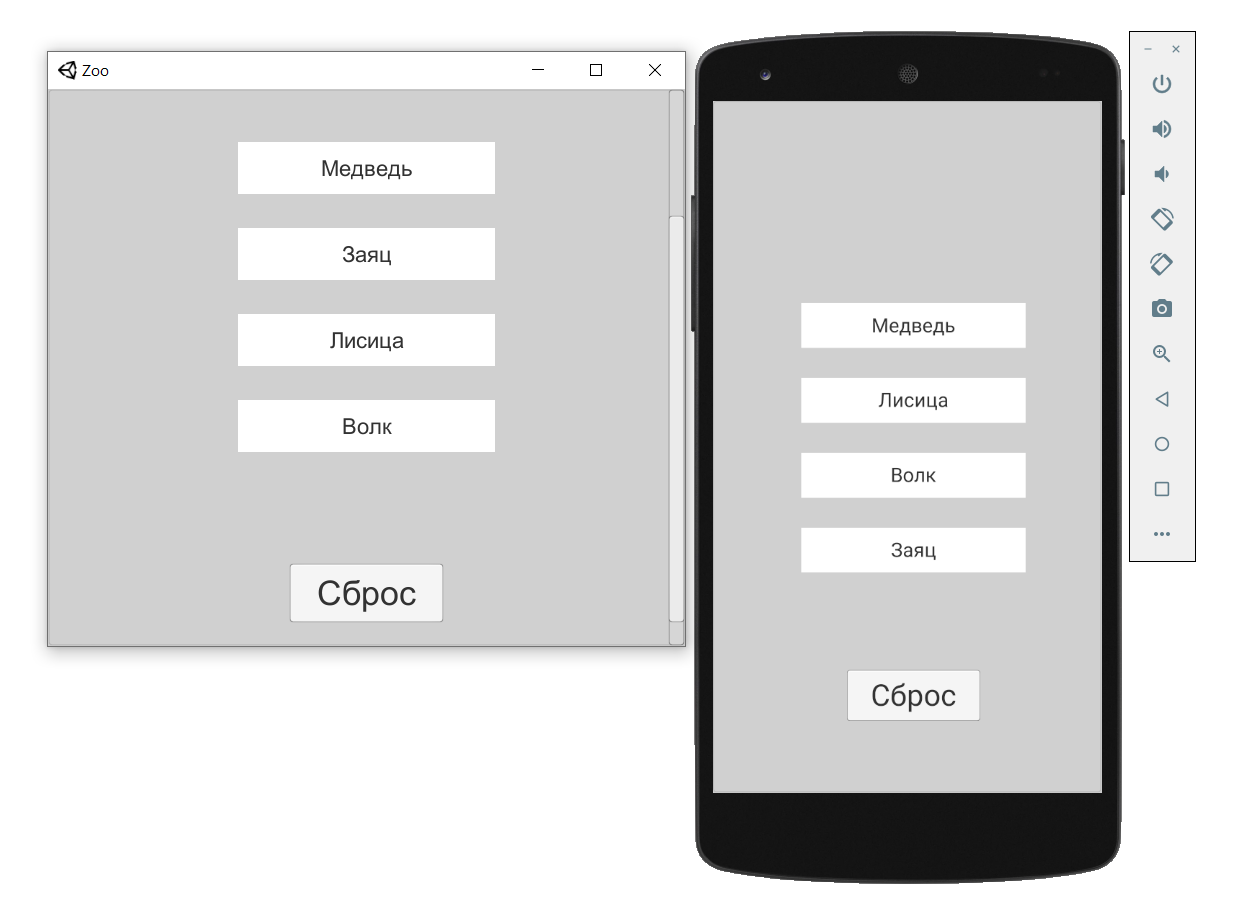


Рис.42. Вид приложения Zoo

# Проект 8: CreateViews

Темы, рассматриваемые в проекте:

* создание элементов через код;
* использование UI элемента Dropdown;
* группировка UI элементов Toggle.

Требуется создать приложение демонстрировавшее добавление UI элемента во время выполнения приложения. Выбор выравнивания элемента осуществляется через группу элементов Toggle, а цвета через выпадающий список Dropdown. Также должна присутствовать кнопка удаления всех созданных элементов.

Выполнить все пункты раздела 3.1 Создание проекта. В пункте 2 дать проекту название CreateViews. Выделить Canvas в Hierarchy, потом в Inspector в разделе Canvas Scaler установить Match в значение 0.5.

Добавить компонент Vertical Layout Group к Canvas; для свойства Padding: Left = 10, Right = 10, Top = 10, Bottom = 10; Spacing = 15; Child Alignment = Upper Left; установить следующие чекбоксы: Width и Height для Control Child Size, Width для Child Force Expand; значения других чекбоксов поставить в выключенное состояние.

На Canvas добавить пустой GameObject, изменить название на «Controls». Добавить Vertical Layout Group; для свойства Padding: Left = 0, Right = 0, Top = 0, Bottom = 0; Spacing = 10; Child Alignment = Upper Left; установить следующие чекбоксы: Width для Control Child Size, Width и Height для Child Force Expand; значения других чекбоксов поставить в выключенное состояние. Внутрь Controls добавить элемент Dropdown с названием «Dropdown», Height = 30, задать значения Options как на рис 43.

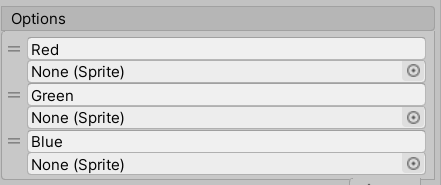


Рис.43. CreateViews, значения Options

Внутрь Controls добавить пустой GameObject, изменить название на «CheckboxGroup». Добавить Horizontal Layout Group; для свойства Padding: Left = 0, Right = 0, Top = 0, Bottom = 0; Spacing = 0; Child Alignment = Middle Center; значения всех чекбоксов поставить в выключенное состояние. Нажать кнопку AddComponent и UI, добавить компонент Toggle Group.

Внутрь CheckboxGroup добавить три элемента Toggle c названиями: LeftToggle, CenterToggle, RightToggle; и подписями: «Left», «Center», «Right».

На Canvas добавить пустой GameObject, изменить название на «Buttons». Для Buttons добавить Horizontal Layout Group; для свойства Padding: Left = 0, Right = 0, Top = 0, Bottom = 0; Spacing = 10; Child Alignment = Upper Right; значения всех чекбоксов поставить в выключенное состояние. Внутрь Buttons добавить два элемента Button c названиями: «CreateBtn», «ClearBtn»; и заголовками: «Create», «Clear».

На Canvas добавить пустой GameObject, изменить название на «Content». Для Content добавить компонент Layout Element, включить Flexible Height со значением 100.

Открыть файл MainController и написать следующий код (листинг 26):

Листинг 26. CreateViews, MainController, инициализация

using UnityEngine;

using UnityEngine.UI;

public class MainController : MonoBehaviour

{

Dropdown dropdown;

Toggle leftTg, rightTg, centerTg;

Button createBtn;

Button clearBtn;

GameObject content;

void Awake()

{

dropdown = GameObject.Find("Dropdown").GetComponent<Dropdown>();

leftTg = GameObject.Find("LeftToggle").GetComponent<Toggle>();

rightTg = GameObject.Find("RightToggle").GetComponent<Toggle>();

centerTg =

GameObject.Find("CenterToggle").GetComponent<Toggle>();

createBtn = GameObject.Find("CreateBtn").GetComponent<Button>();

createBtn.onClick.AddListener(CreateBtnClick);

clearBtn = GameObject.Find("ClearBtn").GetComponent<Button>();

clearBtn.onClick.AddListener(ClearBtnClick);

content = GameObject.Find("Content");

}

}

Объявляются поля для работы с UI элементами, в методе Awake осуществляется их инициализация и добавляются обработчики нажатия на кнопки.

Реализовать обработчики нажатия на кнопки (листинг 27):

Листинг 27. CreateViews, MainController, обработчики

...

public class MainController : MonoBehaviour

{

...

void ClearBtnClick ()

{

foreach (Transform child in content.transform)

Destroy(child.gameObject);

}

void CreateBtnClick ()

{

var optionText = dropdown.options[dropdown.value].text;

var gameObject = new GameObject();

Image img = gameObject.AddComponent<Image>();

if (optionText == "Red") img.color = Color.red;

else if (optionText == "Green") img.color = Color.green;

else if (optionText == "Blue") img.color = Color.blue;

gameObject.transform.SetParent(content.transform);

var rt = img.GetComponent<RectTransform>();

rt.sizeDelta = new Vector2(100, 100);

var lEdge = RectTransform.Edge.Left;

var rEdge = RectTransform.Edge.Right;

var tEdge = RectTransform.Edge.Top;

if (leftTg.isOn)

rt.SetInsetAndSizeFromParentEdge(lEdge, 0, rt.rect.width);

else if (rightTg.isOn)

rt.SetInsetAndSizeFromParentEdge(rEdge, 0, rt.rect.width);

rt.SetInsetAndSizeFromParentEdge(tEdge, 0, rt.rect.height);

Vector3 v = gameObject.transform.localPosition;

v.x = centerTg.isOn ? 0 : v.x;

var count = content.transform.childCount - 1;

v.y -= rt.rect.height \* 1.3f \* count;

gameObject.transform.localPosition = v;

}

}

Для добавления UI элемента Image при нажатии на кнопку «Create» в методе CreateBtnClick создаётся GameObject, к нему добавляется компонент Image. Для Image устанавливается цвет в соответствии с выбранным значением в списке; в зависимости от выбранного Toogle устанавливается выравнивание внутри родительского элемента. Нулевые координаты для localPosition соответствуют расположению в центре. По вертикали рассчитывается отступ в зависимости от количества дочерних элементов внутри Content.

Окончательный вид приложения (рис. 44):

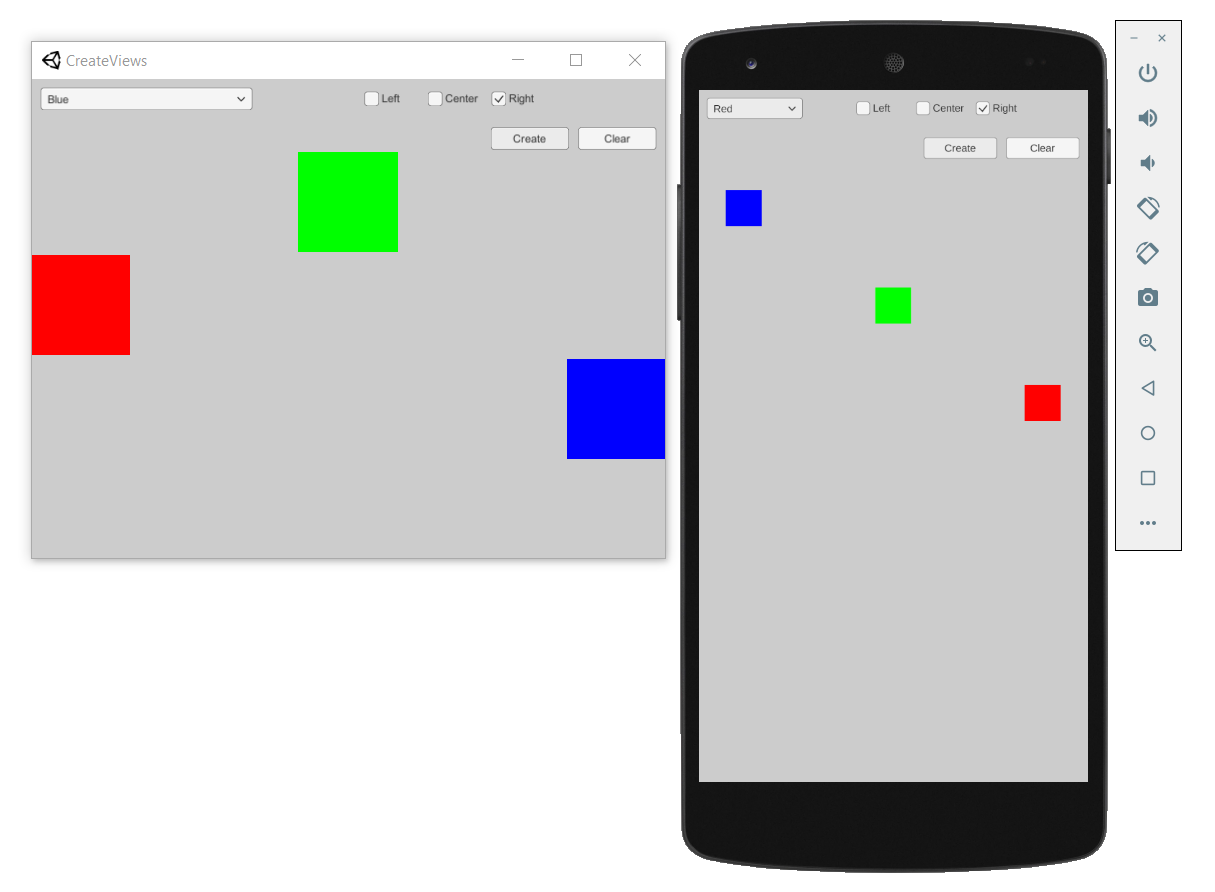


Рис.44. Вид приложения CreateViews

# Заключение

# Литература

1. Абрамян М.Э. Visual C# на примерах. СПб.: БХВ-Петербург, 2008. 482 с.
2. А. В. Абрамян, М. Э. Абрамян. Разработка пользовательского интерфейса на основе технологии Windows Presentation Foundation. Издательство Южного федерального университета, 2017. — 301 с.
3. Unity documentation: <https://docs.unity3d.com/Manual/index.html> (дата обращения 31.05.20)
4. Unity lessons: <https://unity3d.com/ru/learn/tutorials> (дата обращения 31.05.20)
5. Хокинг Джозеф. Unity в действии. Мультиплатформенная разработка на C#. 2-е межд. изд. — СПб.: Питер, 2019. — 352 с.: ил. — (Серия «Для профессионалов»).
6. Мэннинг Д., Батфилд-Эддисон П. Unity для разработчика. Мобильные мультиплатформенные игры. — СПб.: Питер, 2018. — 304 с.: ил. — (Серия «Бестселлеры O’Reilly»).
7. Бонд Джереми Гибсон. Unity и С#. Геймдев от идеи до реализации. 2-е изд. — СПб.: Питер, 2019. — 928 с.: ил. — (Серия «Для профессионалов»).

# Приложение