# Самостоятельное задание на основе Музея в Unity

Начальный уровень, №1. Переключение панелей в пользовательском интерфейсе

**Оглавление**

[Самостоятельное задание на основе Музея в Unity 1](#_Toc124182699)

[Проблема до/после 2](#_Toc124182700)

[Более подробное описание задачи и исходных данных 3](#_Toc124182701)

[Подсказка №1 4](#_Toc124182702)

[Подсказка №2 4](#_Toc124182703)

[Подсказка №3 4](#_Toc124182704)

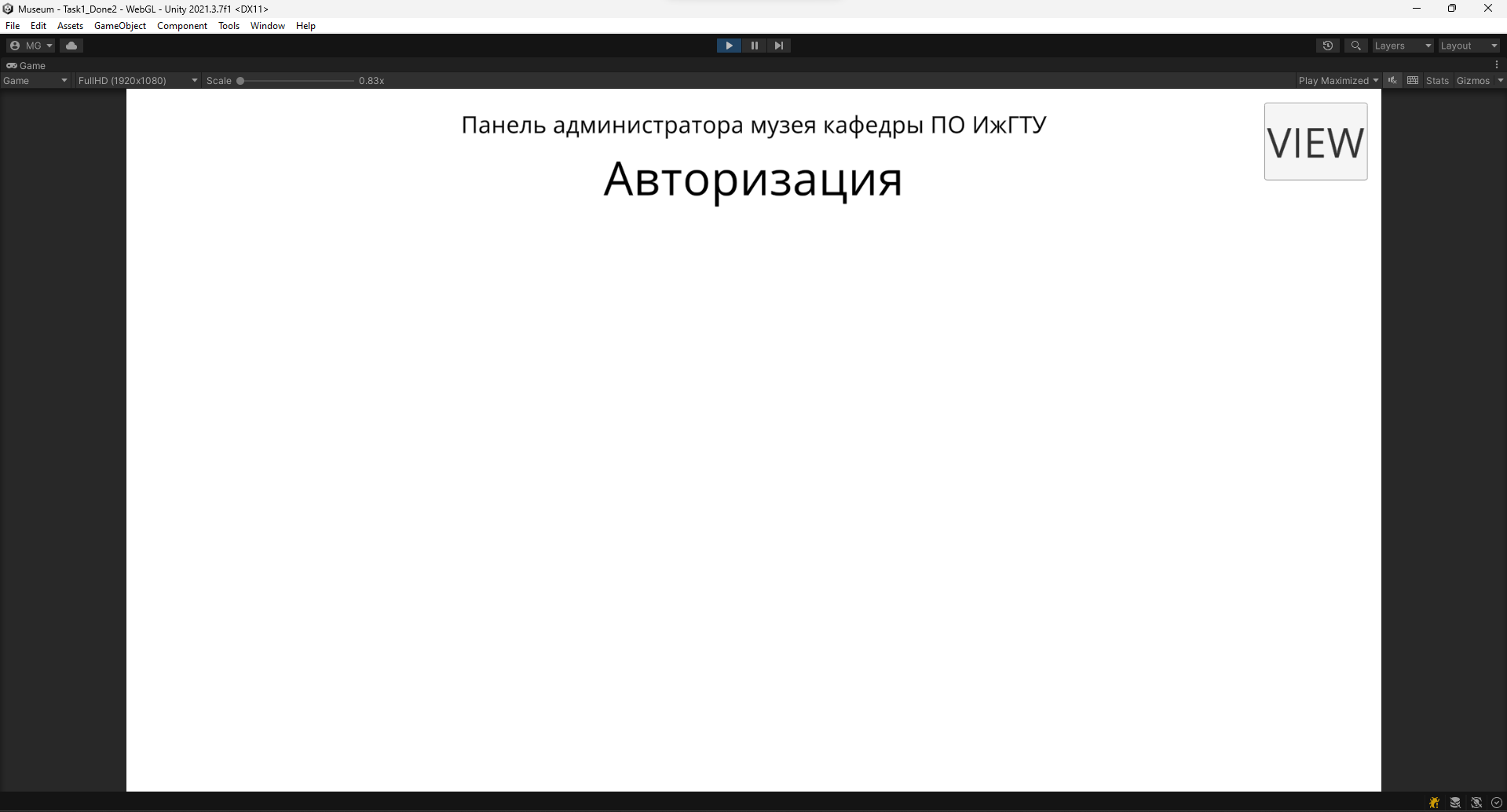
[Подсказка №4 4](#_Toc124182705)

[Подсказка №5 5](#_Toc124182706)

[Отправка результата 5](#_Toc124182707)

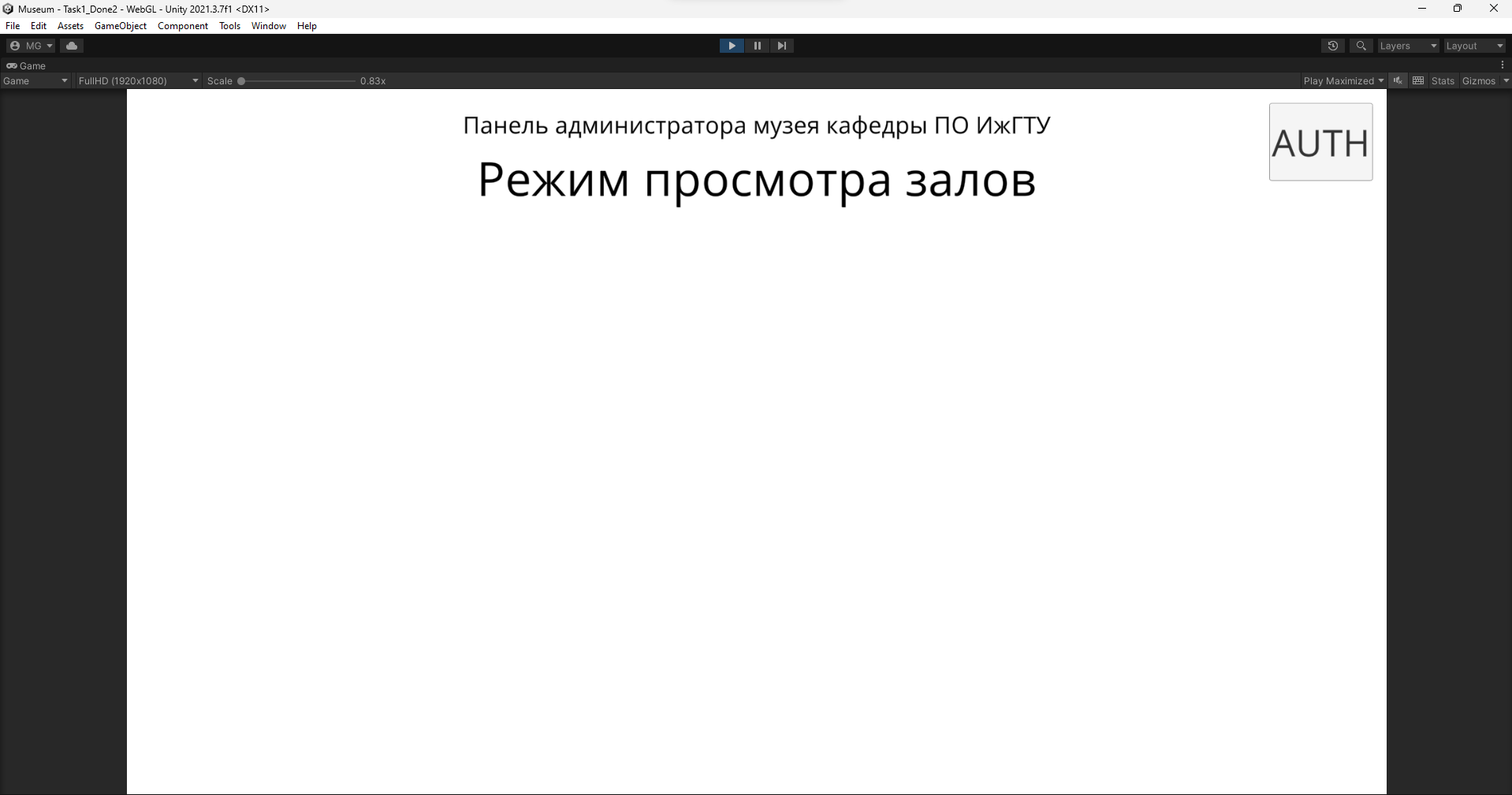
## Проблема до/после

До решения:



Интерфейс есть на экране пользователя, но он не может переключить панель на другую – например, на просмотр залов музея.

После решения:



При нажатии по кнопке VIEW, панель авторизации стала неактивной, а панель просмотра залов – активной. Повторное нажатие, уже по кнопке AUTH вернет приложение в начальное состояние, переключив активность панелей снова.

## Более подробное описание задачи и исходных данных

Если папка с заданием не открыта, откройте папку Admin/Tasks/Beginner/Task1. Здесь подготовлена рабочая сцена Task1. В ней, дочерними объектами к объекту Canvas, есть объекты Panel – Auth (UI панель авторизации) и Panel – View Mode (UI панель просмотра).

Нужно сделать, чтобы при нажатии на кнопку одной панели активировалась вторая, первая деактивировалась. Задание можно решить и без кода.

К заданию можно приступить и сейчас, дальше будут приведены подсказки и советы. Возможно, задание слишком сложно без первой подсказки.

## Подсказка №1

На панелях есть компонент – Canvas Group. Именно с помощью него рекомендуется менять активность панелей. Внутри панелей есть текст и кнопки, благодаря которым должен осуществляться переход между панелями.

## Подсказка №2

У кнопок, в компоненте Button, есть список событий OnClick, который вызывается при нажатии на кнопку. Сюда можно привязать выполнение событий, связанных с Canvas Group.

## Подсказка №3

В Canvas Group нужно использовать 3 свойства для изменения – alpha (прозрачность панели), blocksRaycasts (принимает ли панель клики на себя, или пропускает их глубже в интерфейс), interactable (можно взаимодействовать с элементами панели).

## Подсказка №4

Решение «в лоб», средней эффективности и расширяемости, представлено в папке Solution, файл Task1\_Done1.

В нем к кнопке панели интерфейса привязана другая панель интерфейса, которая включается, а старая панель – отключается.

Это решает проблему, но приходится делать очень много однотипных действий (создавать 6 событий в каждой кнопке), при надобности добавить третью, новую, панель, придется долго переделывать вторую панель, чтобы она ссылалась не на первую, а на третью.

## Подсказка №5

Более эффективное и расширяемое решения представлено в папке Solution, файл Task1\_Done2.

В нем написан класс CanvasGroupExtension, который расширяет класс CanvasGroup, который используется панелями. Для удобства, мы сделали метод SetActive, который отвечает за понятную функциональность, сокращая код или действия программиста.

Также создается класс и игровой объект на сцене – PanelsController. В него программист должен привязывать и привязывать в дальнейшем все панели из сцены. Класс отвечает за включение передаваемой панели и отключение остальных.

В кнопках мы создаем всего одно событие – в него привязываем PanelsController, выбираем метод ChangePanelTo, и в качестве аргумента привязываем нужную нам панель.

## Отправка результата

Результат для проверки нужно собрать в Unity пакет – выберите Export Package, в нем выберите только папку Task1, оставьте отмеченной галку Include Dependencies.

Такой подготовленный Unity пакет можно отправлять для проверки. Возможно, вы сделали очень эффективное и нестандартное решение.