Fibrum SDK v1.0.7

Документация

Оглавление

Быстрый старт

1) Установите **Fibrum SDK** (ссылка на ассет стор)
2) **Edit-ProjectSettings-Player** Вкладка **Resolution and Presentation** Выберите в листе **Default Orientation** строку **Landscape Left**.
3) Найдите префаб **VR_Camera** в папке **FibrumSDK/Prefabs** и перетащите его на сцену.
4) Запустите сцену в редакторе. Пока сцена запущена в редакторе – вы можете вращать камеру мышкой (имитировать вращение головой).
5) Скомпилируйте вашу сцену для Andoid, IOS или другой платформы, использующей гироскоп девайса.

Демо-сцена (1 -DemoTracking)

1) Установите **Fibrum SDK** (ссылка на ассет стор)
2) **Edit-ProjectSettings-Player** Вкладка **Resolution and Presentation** Выберите в листе **Default Orientation** строку **Landscape Left**.
3) Откройте сцену **FibrumSDK/Scenes/Demo_1.scene**
4) Запустите сцену в редакторе Unity.
5) Скомпилируйте сцену для смартфона.

Демо-сцена использования Canvas-GUI Unity3d 4.6-5.x (2 - NewGUI_demo)

1) Установите **Fibrum SDK** (ссылка на ассет стор)
2) **Edit-ProjectSettings-Player** Вкладка **Resolution and Presentation** Выберите в листе **Default Orientation** строку **Landscape Left**.
3) Откройте сцену **FibrumSDK/Scenes/2 - NewGUI_demo.scene**
4) Запустите сцену в редакторе Unity.
5) Скомпилируйте сцену для Android или IOS.

Пояснения к Demo-сцене.

В префабе виртуальной камеры VR_Camera находится скрипт VR_canvasUIcontroller, управляющий работой с Canvas-GUI. Если вы НЕ планируете работать с canvas-GUI – вы можете отключить этот скрипт.

Если скрипт VR_canvasUIcontroller включён на виртуальной камере, то игрок сможет поворотом головы взаимодействовать с кнопками, toggle, скроллами и т.д., созданными с помощью стандартных средств Unity3d Canvas-UI.

Сами элементы UI вы можете создавать на сцене в обычном режиме, для работы виртуальной камеры с кнопками не нужно ничего дополнительно настраивать.

Если вы хотите, чтобы при наведении взгляда на элемент управления на нем появлялся курсор (см демо-сцену), вам всего-лишь нужно положить в нужный Canvas элемент Image с изображением курсора, деактивировать его и назвать VRpointer. При взгляде на элемент управления автоматически выполнится поиск такого курсора в данном канвасе.

Вы можете создавать сколько угодно канвасов в своей сцене.

Вы можете подгружать/активировать/деактивировать/создавать скриптами любые элементы UI – виртуальная камера будет с ними взаимодействовать.

Использование Canvas-UI в качестве игрового меню

Если в игре используется canvas-меню, его можно адаптировать для виртуальной реальности с помощью скрипта VR_canvas (FibrumSDK/Fibrum/VR_GUI/VR_canvas.cs)

Порядок действий:

1) Создайте игровое меню. Оно должно представлять из себя стандартный Unity-canvas, с любым количеством вложенных элементов интерфейса.
2) Добавьте к объекту, который содержит компонент Canvas, скрипт VR_canvas.cs (FibrumSDK/Fibrum/VR_GUI/VR_canvas.cs)
3) Выберите тип отображения Bind Type:
   - Fixed Orientation – меню будет следовать за игроком, но не будет менять поворота в пространстве (наиболее редкое использование)
   - Fixed In Center – меню всегда показывается ровно в центре экрана (хорошо подходит для диалоговых сообщений)
   - Always In View Field – по меню можно будет водить взглядом, взаимодействовать с элементами интерфейса. При попытке увести взгляд за пределы меню, меню будет довёрнуто, таким образом оно всегда будет попадать в поле зрения как минимум наполовину (оптимально для отображения игровых меню)
4) Выберите масштаб scale. При scale=1 меню будет по горизонтали полностью заполнять обзор игрока.
5) Для отображения данного объекта canvas-меню экране достаточно добавить его на сцену, или активировать. При необходимости скрыть меню объект можно удалить, либо деактивировать его.

Демо-сцена использования и настройки джойстика (3-joystickSetup_demo)

1) Установите **Fibrum SDK** (ссылка на ассет стор)
2) **Edit-ProjectSettings-Player** Вкладка **Resolution and Presentation** Выберите в листе **Default Orientation** строку **Landscape Left**.
3) Откройте сцену **FibrumSDK/Scenes/3 - joystickSetup_demo.scene**
4) Запустите сцену в редакторе Unity.

Пояснения к демо-сцене

Объект Joystick_Simple_FPS_character имеет CharacterController имеет 2 дочерних объекта – стандартную стереокамеру и GUI, при нажатии на который запускается функция SetupJoystick в Joystick_Simple_FPS_character.

По сцене можно ходить с помощью джойстика #1 и стрелять кнопокой A.

Подробное описание смотрите в разделе FibrumInput.

VR_Camera

Префаб VR_Camera представляет собой GameObject со скриптом-контроллером. У него есть дочерний объект VRCamera, и именно он повторяет движения девайса в пространстве, относительно неподвижного родителя VR_Camera. Поэтому ваш герой может двигаться по сцене другими программными способами (бегать с помощью джойстика или ехать по заданной кривой по американским горкам), и при этом осматриваться вокруг.

Свойства контроллера виртуальной камеры VRCamera

Для удобного доступа к скрипту виртуальной камеры на сцене, объявлена статическая переменная VRCamera FibrumController.vrCamera. При появлении виртуальной камеры на сцене FibrumController.vrCamera ссылается на скрипт VRCamera.

Например: FibrumController.vrCamera.vrCameraHeading – переменная типа Transform, возвращающая положение смартфона в пространстве. (подробнее см. ниже описание свойств VRCamera)

Cameras – массив камер. Используется для настройки эффектов стерео-камер, автоматического подгона изображения физический размер экрана, замена камер местами при запуске в редакторе Unity (возможность оценивать 3d-эффект простым скашиванием глаз ☺), использование шейдера коррекции линз при достаточной производительности девайса.

Mouse Sensitivity – чувствительность мыши при поворотах виртуальной камеры в редакторе.

FPS – текущий fps. Сканирование FPS происходит постоянно, используется контроллером для первоначальной оценки производительности сцены на данном девайсе, чтобы включать или выключать дополнительные шейдеры на камерах.

IfGoodFpsSoDoLensCorrection – если выставлено, при запуске сцены превые 1.5 секунду виртуальная камера будет закрыта чёрным экраном. За это время будет оценен FPS сцены и если FPS>20, то к камерам будет применён пост-эффект коррекции линз. Если FPS ниже 20, пост-эффект будет выключен. Если IfGoodFpsSoDoLensCorrection не выставлен – не будет чёрного экрана при запуске сцены, на камерах будет изначально включён пост-эффект коррекции линз.

UseCompassForAntiDrift – (для подавляющего большинства приложений – не обязательный параметр) если выставлено, камера будет ориентироваться на компас телефона. Полезно, если нужно исключить дрифт камеры, связанный с ошибкой гироскопа при быстрых поворотах головы. Т.е. при резком повороте головы и резком возвращении её на место, виртуальная камера обычно не возвращается в изначальное положение. Чтобы после серии поворотов головы человек в реальном пространстве был повёрнут в ту же сторону, что и до серии поворотов – используется компас сматрфона.

BlackTexFade – чёрная текстура, используемая при затемнении виртуально камеры при проверке производительности сцены.

Дополнительные публичные свойства VRCamera

Transform VRCamera.vrCameraHeading – матрица Transform, непосредственно отслеживающая положение телефона в пространстве.

Свойства контроллера Canvas-GUI VR_CanvasUIcontroller

LookToPressTime – время, на которое нужно задержать взгляд на элементе, чтобы произвести над ним действие.

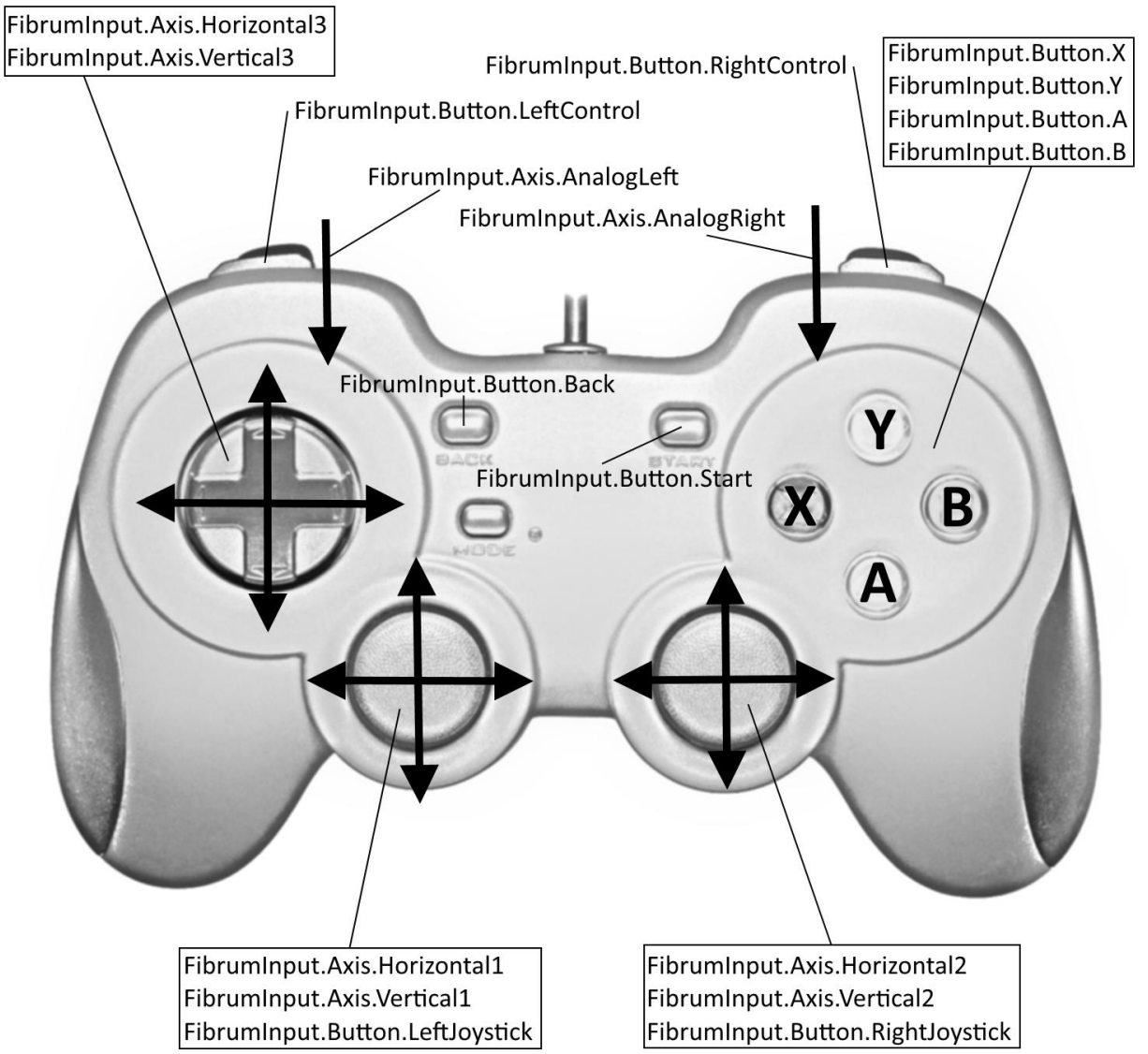
ProgressBarTex – текстура, которая будет заполняться по кругу при выборе элемента Canvas-GUI.

FibrumInput

FibrumInput – это статический класс, призванный упростить общение приложения с осями и кнопками джойстиков и геймпадов. В нём реализованы функции получения состояния осей и кнопок геймпадов, переназначения функций осей и кнопок геймпадов «ингейм», сохранение и загрузка параметров геймпада в PlayerPrefs. (после первой настройки джойстика далее его настройки будут автоматически подгружаться при новых запусках приложения).

Т.к. класс статический – в приложении не нужно создавать его экземпляр, или проводить инициализаций.

В FibrumInput перечислены следующие оси джойстиков:



float FibrumInput.GetJoystickAxis(FibrumInput.Axis axis) - возвращает значение оси джойстика, например: FibrumInput.GetJoystickAxis(FibrumInput.Axis.Vertical1)

bool GetJoystickButton(Button buttonNum) – возвращает состояние кнопки, например FibrumInput.GetJoystickButtonDown(FibrumInput.Button.A)

bool GetJoystickButtonDown(Button buttonNum) – возвращает true, если кнопка была нажата в текущем фрейме.

bool GetJoystickButtonUp(Button buttonNum) – возвращает true, если кнопка была отпущена в текущем фрейме.

bool FibrumInput.InitializeJoystickSetup() – Инициализация настройки джойстика. Создаётся невидимый Canvas-UI объект FibrumInput.joystickSetupGO,и привязывается к камере. Далее необходимо указать, какие оси и кнопки нужно настроить (см. функции ниже). Возвращает true, если объект успешно создан. Если объект уже существует – возвращает false.

FibrumInput.joystickSetupGO.SetControl(FibrumInput.Axis axis,string description,Texture tex) – указывает, какую ось необходимо будет настроить. Axis – ось для настройки, description – текст надписи для пользователя, пояснение, для чего настраивается данная ось,tex – картинка, поясняющая настройку. (см. пример ниже)

FibrumInput.joystickSetupGO.SetControl(FibrumInput.Axis axis,bool checkAxisDirection,string description,Texture tex) – функция работает аналогично предыдущей, но при этом теперь играет роль направление наклона джойстика. Если checkAxisDirection=true, значит требуется положительный сигнал от оси джойстика, иначе отрицательный. Полезно при использовании нестандартных джойстиков, например с перевернутыми осями, tex – картинка, поясняющая настройку. (см. пример ниже)

FibrumInput.joystickSetupGO.SetControl(FibrumInput.Button button,string description,Texture tex) – указывает, какую кнопку необходимо настроить. button – кнопка для настройки, description – текст надписи для пользователя, пояснение, для чего настраивается данная кнопка, tex – картинка, поясняющая настройку. (см. пример ниже)

FibrumInput.joystickSetupGO.StartSetup() – делает объект FibrumInput.joystickSetupGO видимым и запускается процесс назначения кнопок и осей. После окончания настройки FibrumInput.joystickSetupGO удаляет себя, а все настройки сохраняются в PlayerPrefs, привязываясь к id текущего джойстика.

ClearControls() – сбрасывает список осей и кнопок джойстика для настройки. Небходимо использовать, если кнопки и оси будут настраиваться скриптом. (см. ниже)

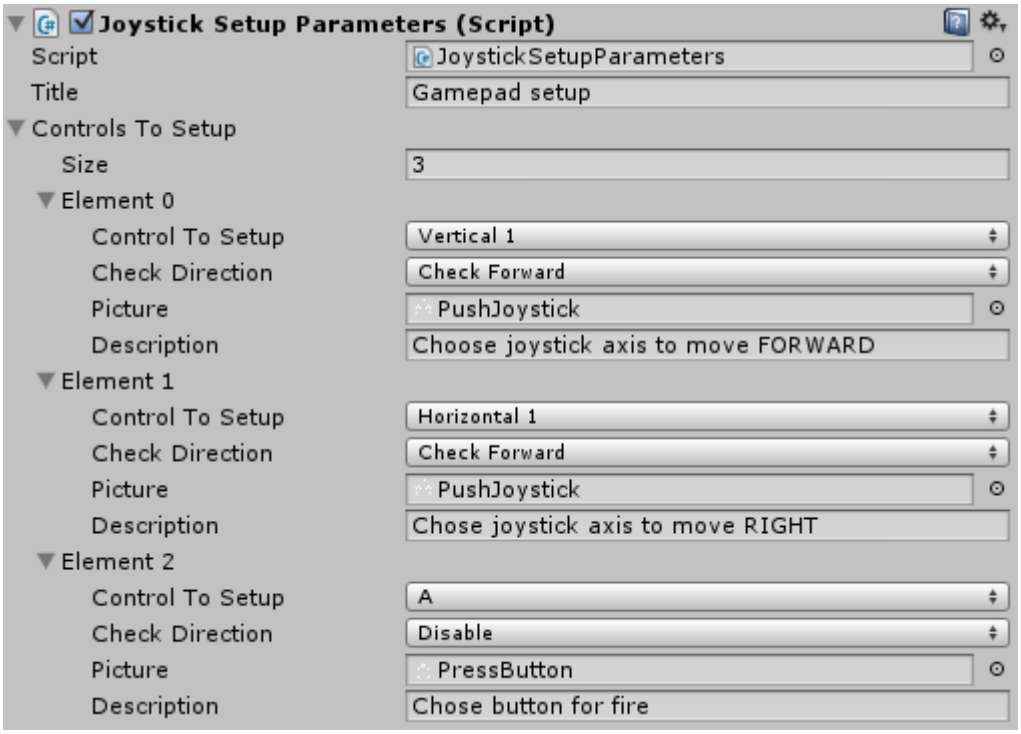
Пример запуска настройки джойстика из скрипта

```
public void SetupJoystickScripted()
{
    if( FibrumInput.InitializeJoystickSetup() )
   {
      FibrumInput.joystickSetupGO.ClearControls();
      FibrumInput.joystickSetupGO.SetTitle("Setup joystick input");
      FibrumInput.joystickSetupGO.SetControl(FibrumInput.Axis.Vertical1,true,"FORWARD\nChoose joystick axis to move forward",null);
      FibrumInput.joystickSetupGO.SetControl(FibrumInput.Axis.Horizontal1,false,"RIGHT\nChose joystick axis to move right",null);
      FibrumInput.joystickSetupGO.SetControl(FibrumInput.Button.A,"Chose button for fire",null);
      FibrumInput.joystickSetupGO.StartSetup();
   }
}
```

Запуск процедуры настройки джойстика из скрипта легко локализовать. Достаточно подставлять в функции SetControl и SetTitle тексты на других языках.

Настройка управления джойстиком в редакторе

В префабе VR_Camera имеется компонент JoystickSetupParameters, с помощью которого можно задать настраиваемые оси и кнопки джойстика, аналогично со скриптовым методом.



При вызове функции инициализации окна настройки джойстика **FibrumInput.InitializeJoystickSetup()** автоматически идёт подгрузка настроек джойстика из данного компонента JoystickSetupParameters. При этом вызов окна активации будет выглядеть так:

```
public void SetupJoystick()
{
   if( FibrumInput.InitializeJoystickSetup() )
   {
      FibrumInput.joystickSetupGO.StartSetup();
   }
}
```

История версий

1.0.7 – Поправлены настройки виртуальной камеры, увеличена стабильность UI системы, возможность отключать систему антидрифта.

1.0.6 – По умолчанию камеры разворачивают изображение на весь экран, можно изменить тип отображения камер. При отсутствии гироскопа на девайсе, трекинг будет эмулироваться аксерерометром и компасом. Увеличена стабильность антидрифт-системы. Исправлены ошибки с отображением настройки джойстика.

1.0.5 – Добавлена поддержка хедтрекинга для ОС Windows Phone 8/8.1; Добавлена возможность ограниченного хедтрекинга в случае, если смартфон не имеет гироскопа (в этом случае по преднему будет высвечиваться предупреждение об отсутствии гироскопа).

1.0.4 – Добавлена поддержка осей и кнопок стандартных геймпадов, добавлена возможность переназначения осей и кнопок геймпада в процессе работы приложения, с возможностью сохранения и автоматической загрузки настроек для каждого джойстика. Добавлена подстройка размеров и положений изображения под физический размер смартфона (для комфортного просмотра в шлеме Fibrum). Добавлена информационная надпись для смартфонов без встроенного гироскопа.

1.0.3 – Добавлена поддержка Unity Canvas-элементов из коробки, взаимодействие с интерфейсом путём хедтрекинга.

1.0.2 – Добавлена проверка производительности смартфона, в случае низкой производительности отключаются эффекты искажения линз.

1.0.1 – Добавлена система антидрифта, как на основе накапливающихся данных, так и на основе магнитного компаса смартфона.

1.0.0 – SDK для смартфонов под управлением ОС Android и IOS, поддерживающий хедтрекинг.