# Иерархический список классов

## Иерархия классов

Иерархия классов.

MonoBehaviour

Billboard

CameraLayerControl

CubeCreator

DataSource

Distributor

ElementController

Flymode

LaserPointerController

MonitorController

Reciever

ScrollBarButton

SteamVRLaserWrapper

UniversalConverter

UniversalTranslator

Visualiser

BarVisualiser

Callout

ColorChanger

NodeVisualisator

StreamVisualiser

VRUIItem

# Алфавитный указатель классов

## Классы

Классы с их кратким описанием.

**BarVisualiser (Класс для контроля столбчатых визуализаторов)** pagenum

**Billboard (Класс, направляющий объект, которому привязан в сторону основной камеры)** pagenum

**Callout (Класс для контроля визуализаторов сигналов тревоги)** pagenum

**CameraLayerControl (Класс, регулирующий отображение слоев на картинке с главной камеры)** pagenum

**ColorChanger (Класс для контроля цветовых визуализаторов)** pagenum

**CubeCreator (Класс-конструктор элементов объемного визуализатора)** pagenum

**DataSource (Класс источника данных для визуализаторов)** pagenum

**Distributor (Синглтон-дистрибутор данных)** pagenum

**ElementController (Класс для контроля элементов интерфейса, отображающих данные)** pagenum

**Flymode (Класс для перемещения аватара игрока полётом )** pagenum

**LaserPointerController (Класс для контроля лазерного указателя, сканера и меню )** pagenum

**MonitorController (Класс для контроля интерфейса монитора оборудования )**

**NodeVisualisator (Класс экземпляров объемных визуализаторов )**

**Reciever (Класс для получения данных с сервера через MQTT )** pagenum

**ScrollBarButton (Класс для прокручивания списка элементов интерфейса )** pagenum

**SteamVRLaserWrapper (Класс, исправляющий работу SteamVR\_LaserPointer )** pagenum

**StreamVisualiser (Класс для контроля потоковых визуализаторов )** pagenum

**UniversalConverter (Синглтон-конвертер входных данных в настройки визуализаторов )** pagenum

**UniversalTranslator (Класс для преобразования типов, получения единиц измерения и переводов )** pagenum

**Visualiser (Родительский класс всех объектов для визуализации )** pagenum

**VRUIItem (Класс для автоматизированного создания коллайдеров на элементах интерфейса )** pagenum

# Классы

## Класс BarVisualiser

Класс для контроля столбчатых визуализаторов

##### Авторы

Пивко Артём

##### Версия

1.0

##### Дата

13.05.20

##### Предупреждения

Если source не присвоить в сцене, объект не будет функционировать!

Этот класс занимается контролем высоты и цвета столбца-визуализатора, которым должен являться объект, к которому прикреплён скрипт. Контроль происходит в соответствии с данными из источника данных **DataSource** source и внутренних настроек объекта.

### Методы

#### void BarVisualiser.Awake ()[private]

Установочная функция, вызываемая в момент активации объекта на сцене.

В первую очередь в ней устанавливается ссылка на материал рендерера родительского объекта. Если это делать более традиционным путем - префабной связью - скриптом изменяется глобальный материал всех визуализаторов данного типа.

После этого стандартный цвет приравнивается цвету, которым обладает объект на начало исполнения программы.

Так же задается целевой объект изменения высоты, который в данной версии является объектом, к которому прикреплен скрипт, а также максимальная высота объекта, равная высоте объекта до начала выполнения программы. После всех описанных действия начинается исполнение сопрограммы DelayedUpdate с частотой обновлений в updateTime.

#### IEnumerator BarVisualiser.DelayedUpdate (float *\_time*)[private]

Сопрограмма-таймер основного цикла.

##### Аргументы

|  |  |
| --- | --- |
| *\_time* | Время обновления |

Отвечает за регулярное обновление высоты объекта в соотвествии с данными источника. В бесконечном цикле внутри сопрограммы из источника данных source берется значение данных, после чего вызываетася метод SetHeight. После этого производится ожидание на время обновления до повторения цикла.

#### override string BarVisualiser.Scan (out string *visType*, out string *dataType*, out string *topic*)[virtual]

Метод снятия показаний с визуализатора сканером

##### Аргументы

|  |  |
| --- | --- |
| *visType* | Тип визуализатора |
| *dataType* | Тип данных |
| *topic* | MQTT-топик данных |

##### Возвращает

Визуализируемые данные в формате string

#### void BarVisualiser.SetHardHeight (float *h*)[private]

Метод задания абсолютной высоты объекта в локальных координатах.

##### Аргументы

|  |  |
| --- | --- |
| *h* | Абсолютная высота. Используется только после предварительной обработки данных. |

#### void BarVisualiser.SetHeight (float *\_h*, float *\_time*)

Метод плавного задания высоты объекта по данным в локальных координатах.

##### Аргументы

|  |  |
| --- | --- |
| *\_h* | Входные данные |
| *\_time* | Время смены в секундах. Запускает сопрограмму плавной смены высоты SmoothChange. |

#### void BarVisualiser.SetHeight (float *data*)[private]

Метод задания высоты объекта по данным в локальных координатах.

##### Аргументы

|  |  |
| --- | --- |
| *h* | Входные данные. |

В случае если входные данные меньше или равны порогу minData, задается минимальная высота объекта и ему задается синий цвет.

В случае если входные данные находятся между minData и maxData, данные нормализуются относительно порогов и высот, и по получивнемуся значению задается высота объекта. Цвет объекта устанавливается равным стандартному.

В случае если входные данные больше или равны порогу maxData, задается максимальная высота объекта и ему задается красный цвет.

По окончанию метода задается текущее показание визуализатора, равное входным данным.

#### IEnumerator BarVisualiser.SmoothChange (float *\_h*, float *\_time*)[private]

Сопрограмма, для плавного изменения высоты объекта.

##### Аргументы

|  |  |
| --- | --- |
| *\_h* | Входные данные |
| *\_time* | Время смены в секундах |

Изменяет высоту диаграммы с начального до \_h за \_time секунд

### Данные класса

#### float BarVisualiser.curData[private]

Текущее значение отображаемых данных.

#### Color BarVisualiser.defColor[private]

Цвет столбца в обычном состоянии. Берется из изначального цвета столбца в редакторе.

#### Material BarVisualiser.material = null[private]

Ссылка на материал изменяемого объекта.

#### float BarVisualiser.maxData = 100f[private]

Значение данных, при которых высота столбца будет максимальной. Регулируется из редактора.

#### float BarVisualiser.maxHeight = 1f[private]

Максимальная высота столбца в локальных единицах измерения. Регулируется из редактора.

#### float BarVisualiser.minData = 10f[private]

Значение данных, при которых высота столбца будет минимальной. Регулируется из редактора.

#### float BarVisualiser.minHeight = 0.1f[private]

Минимальная высота столбца в локальных единицах измерения. Регулируется из редактора.

#### DataSource BarVisualiser.source

Источник данных, контролирующий цвет. Присваевается в сцене до начала работы скрипта.

#### GameObject BarVisualiser.target = null[private]

Изменяемый объект

#### float BarVisualiser.updateTime = 1f[private]

Время обновления объекта

#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**BarVisualiser.cs**

## Класс Billboard

Класс, направляющий объект, которому привязан в сторону основной камеры

##### Авторы

Пивко Артём

##### Версия

1.0

##### Дата

13.05.20

Используется для направления иконок тревоги в сторону пользователя.

### Методы

#### void Billboard.Update ()[private]

Функция основного цикла.

В зависимости от параметра isRotating3D каждый кадр вращает объект либо по оси Y, либо по всем осям.

### Данные класса

#### bool Billboard.isRotating3D = false[private]

Если этот параметр отмечен, вращение будет происходить по всем трём осям. Иначе только по оси Y.

#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**Billboard.cs**

## Класс Callout

Класс для контроля визуализаторов сигналов тревоги.

##### Авторы

Пивко Артём

##### Версия

1.0

##### Дата

13.05.20

##### Предупреждения

Если source не присвоить в сцене, объект не будет функционировать!

Этот визуализатор занимается активацией и деактивацией объекта сигнала тревоги в соотвествии с данными из источника данных **DataSource** source. Так же осуществляет динамичное расположение цилиндра-тела выноски между начальной и конечной её точками, что избавляет от необходимости выставлять его позицию вручную.

Важно: работает не только во время работы приложения, но и в редакторе.

### Методы

#### override string Callout.Scan (out string *visType*, out string *dataType*, out string *topic*)[virtual]

Метод снятия показаний с визуализатора сканером

##### Аргументы

|  |  |
| --- | --- |
| *visType* | Тип визуализатора |
| *dataType* | Тип данных |
| *topic* | MQTT-топик данных |

##### Возвращает

"Active" в случае если объект активен, "Off" в случае если нет. Внимание, передаваемые в метод параметры при его вызове должны стоять после ключегого слова out.

Замещает **Visualiser** (*стр.pagenum*).

#### void Callout.Update ()[private]

Функция основного цикла. Отвечает за покадровое обновление позиции цилиндра, а так же за включение и выключение объекта в зависимости от данных с источника.

#### void Callout.UpdateCylinderPosition (GameObject *cylinder*, Vector3 *beginPoint*, Vector3 *endPoint*)[private]

Метод установки позиции цилиндра между начальной и конечной точкой.

##### Аргументы

|  |  |
| --- | --- |
| *cylinder* | Устанавливаемый цилиндр. |
| *beginPoint* | Координаты начала |
| *endPoint* | Координаты конца |

Устанавливает объект cylinder на середину между точкой начала и конца, затем поворачивает его, чтобы его ось Y совпала с линией между точками, затем растягивает его так, чтобы основания объекта касались начальной и конечной точек.

### Данные класса

#### GameObject Callout.cylinder = null[private]

Цилиндр, соединяющий начало и конец выноски.

#### Transform Callout.finish = null[private]

Точка конца выноски.

#### GameObject Callout.model = null[private]

Компонент, который выключается и включается в зависимости от данных источника. Рекоммендуется задавать здесь сам корневой объект выноски.

#### DataSource Callout.source = null[private]

Источник данных, в соответствии с которым работает визуализатор.

#### Transform Callout.start = null[private]

Точка начала выноски (обычно утапливается в объект, к которому применена выноска).

#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**Callout.cs**

## Класс CameraLayerControl

Класс, регулирующий отображение слоев на картинке с главной камеры.

##### Авторы

Пивко Артём

##### Версия

1.0

##### Дата

13.05.20

Этот визуализатор занимается динамическим изменением цвета родительского объекта в соотвествии с данными из источника данных **DataSource** source. Для конверсии используется синглтон UniversalController

### Методы

#### void CameraLayerControl.Awake ()[private]

Метод, выполняющийся на старте программы. Устанавливает основную камеру сцены как текущую камеру.

#### void CameraLayerControl.NotRender (int *n*)

Метод для включения/выключения отображения слоев на основной камере.

##### Аргументы

|  |  |
| --- | --- |
| n | Номер переключаемого слоя. n должен принадлежать промежутку [0,31]. Если n - корректный номер слоя, включает его отображение если тот был выключен, и выключает если тот был включен. |

### Данные класса

#### Camera CameraLayerControl.current = null[private]

Камера, с которой работает объект.

#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**CameraLayerControl.cs**

## Класс ColorChanger

Класс для контроля цветовых визуализаторов

##### Авторы

Пивко Артём

##### Версия

1.2

##### Дата

13.05.20

##### Предупреждения

Если source не присвоить в сцене, объект не будет функционировать!

Этот визуализатор занимается динамическим изменением цвета родительского объекта в соотвествии с данными из источника данных **DataSource** source. Для конверсии используется синглтон UniversalController

### Методы

#### void ColorChanger.Awake ()[private]

Установочная функция, вызываемая в момент активации объекта на сцене.

В первую очередь в ней устанавливается ссылка на материал ренерера родительского объекта. Если это делать более традиционным путем - префабной связью - скриптом изменяется глобальный материал всех визуализаторов данного типа.

После установки материала начинается исполнение сопрограммы DelayedUpdate с частотой обновлений в updateTime.

#### IEnumerator ColorChanger.DelayedUpdate (float *\_time*)[private]

Сопрограмма-таймер основного цикла.

##### Аргументы

|  |  |
| --- | --- |
| *\_time* | Время обновления |

Отвечает за регулярное обновление температуры и цвета объекта в соотвествии с данными источника.

В бесконечном цикле внутри сопрограммы из источника данных source берется значение температуры, после чего вызываетася метод SetTemp на время равное половине времени обновления. После этого производится ожидание на время обновления до повторения цикла.

#### override string ColorChanger.Scan (out string *visType*, out string *dataType*, out string *topic*)[virtual]

Метод снятия показаний с визуализатора сканером

##### Аргументы

|  |  |
| --- | --- |
| *visType* | Тип визуализатора |
| *dataType* | Тип данных |
| *topic* | MQTT-топик данных |

##### Возвращает

Визуализируемые данные в формате string Внимание, передаваемые в метод параметры при его вызове должны стоять после ключегого слова out.

#### void ColorChanger.SetTemp (float *\_temp*)

Метод, устанавливающий температуру и, соотвественно, цвет на объекте

##### Аргументы

|  |  |
| --- | --- |
| *\_temp* | Устанавливаемая температура |

#### void ColorChanger.SetTemp (float *\_temp*, float *\_time*)

Метод, устанавливающий температуру и, соотвественно, цвет на объекте за указанное время. Интерфейс для корутины SmoothChange.

##### Аргументы

|  |  |
| --- | --- |
| *\_temp* | Устанавливаемая температура |
| *\_time* | Время изменения |

#### void ColorChanger.SetTempCold ()

Метод, быстро устанавливающий недопустимо холодную температуру. Для тестов.

#### void ColorChanger.SetTempOK ()

Метод, со средней скоростью устанавливающий оптимальную температуру. Для тестов.

#### void ColorChanger.SetTempWarm ()

Метод, медленно устанавливающий недопустимо холодную температуру. Для тестов.

#### IEnumerator ColorChanger.SmoothChange (float *\_temp*, float *\_time*)[private]

Сопрограмма, для плавного изменения цвета объекта.

##### Аргументы

|  |  |
| --- | --- |
| *\_temp* | Устанавливаемая температура |
| *\_time* | Время изменения |

Что у нас есть:

• Целевой RGB цвет

• Время смены

• Изначальный RGB цвет

• DeltaTime – время смены кадра в Unity

RGB цвет состоит из трёх компонент, собственно R G и B (ещё есть альфа-канал, но в данном случае он не важен).

Для плавной смены цвета нужно чтобы каждый кадр значение цвета менялось в сторону целевого цвета на величину пропорциональную DeltaTime. Должна соблюдаться пропорция:

DeltaColor/(ЦелевойЦвет-ИзначальныйЦвет) = DeltaTime / ВремяСмены

Где DeltaColor – изменение цвета за кадр

Таким образом

DeltaColor = DeltaTime \* (ЦелевойЦвет-ИзначальныйЦвет) / ВремяСмены

При линейной плавной смене цвета таких дельты будет 3: для каждого из каналов RGB. Если же смена нелинейная, а идет как, например, в нашем случае через зеленый, то дельтировать нужно контролирующий параметр, то есть температуру.

### Данные класса

#### float ColorChanger.absCold = 11f[static], [private]

Температура, ниже которой не отслеживается

#### float ColorChanger.absHeat = 38f[static], [private]

Температура, выше которой не отслеживается

#### float ColorChanger.allowedFinish = 28f[static], [private]

Граница верхней недопустимой температуры

#### float ColorChanger.allowedStart = 21f[static], [private]

Граница нижней недопустимой температуры

#### Material ColorChanger.material = null[private]

Ссылка на устанавливается не префабной связью а в Awake ввиду того что иначе изменяется глобальный материал

#### float ColorChanger.maxTransp = 0.4f[static], [private]

Максимальное значение прозрачности объекта

#### float ColorChanger.optimalFinish = 25f[static], [private]

Верхняя граница оптимальной температуры

#### float ColorChanger.optimalStart = 23f[static], [private]

Нижняя граница оптимальной температуры

#### DataSource ColorChanger.source

Источник данных, контролирующий цвет. Присваевается в сцене до начала работы скрипта.

#### float ColorChanger.temp = 23f

Текущее значение температуры объекта

#### float ColorChanger.updateTime = 5f[private]

Время обновления объекта

#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**ColorChanger.cs**

## Класс CubeCreator

Класс-конструктор элементов объемного визуализатора.

##### Авторы

Пивко Артём

##### Версия

1.0

##### Дата

13.05.20

##### Предупреждения

Методы этого класса очень медленные и не предназначены для исполнения во время работы программы. Только в режиме редактирования.

Скрипт занимается созданием и размещением объектов node на ребрах объемной сетки размера zoneSize и плотности density.

Рассчётное применение класса - создание элементов объемного визуализатора и задание им общего типа данных.

### Методы

#### void CubeCreator.DestroyCubes ()

Метод уничтожения объемной сетки объектов Метод уничтожает уже созданную в текущем выполнении программы сетку.

#### void CubeCreator.GenerateCubes ()

Метод создания объемной сетки объектов Метод рассчитывает количество генерируемых объектов по каждой оси на основании размеров зоны и плотности. После этого вычисляется смещение offset, необходимое чтобы объекты находились внутри зоны, не пересекая её границ. Далее объекты инициируются на узлах 3D сетки и становятся по иерархии детьми данного объекта. Их названия меняются на Node[x,y,z], где x,y,z - его номер в объемной матрице. Если объект - элемент объемного визуализатора с компонентом **NodeVisualisator**, то его тип слушаемых данных устанавливается равным dataType.

#### void CubeCreator.OnDrawGizmosSelected ()[private]

Служебная функция, отображающая размеры границы в редакторе.

### Данные класса

#### string CubeCreator.dataType = null[private]

Тип данных, которые будут слушать размещаемые элементы объемного визуализатора.

#### Vector3 CubeCreator.density[private]

Плотность размещения объектов по трём измерениям. Задается из редактора.

#### GameObject CubeCreator.node = null[private]

Размножаемый объект. Предпочтительно префаб. Задается из редактора.

#### GameObject [,,] CubeCreator.nodes = null[private]

Трёхмерная матрица, хранящая ссылки на созданные объекты.

#### int CubeCreator.xn[private]

#### int CubeCreator.yn[private]

#### int CubeCreator.zn = 0[private]

Текущие размеры матрицы объектов.

#### Vector3 CubeCreator.zoneSize[private]

Размер зоны по трём измерениям. Задается из редактора.

#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**CubeCreator.cs**

## Класс DataSource

### Подробное описание

Класс источника данных для визуализаторов.

##### Авторы

Пивко Артём

##### Версия

1.1

##### Дата

19.05.20

##### Предупреждения

Для работы объекта на сцене требуется объект **Distributor**

Класс, хранящий текущие данные с сервера. По сути является интерфейсом ко всей системе получения данных для визуализаторов.

Данные источника задаются присутствующим на сцене объектом **Distributor** в соответствии с MQTT-топиком источника, который является именем этого объекта. Может функционировать в двух вариантах: в качестве источника данных непосредственно по прямой ссылке с визуализатора, либо как объемный источник данных. Для работы во втором режиме задается доверительный радиус источника, который определяет в сфере какого радиуса данные с этого источника актуальны.

Тип данных - дополнительный фильтр, позволяющим разным объемным визуализаторам отображать разные типы данных.

### Методы

#### void DataSource.Awake ()[private]

Метод инициализации объекта В случае если источник является дочерним объектом, то объект переименовывается в полный путь до объекта.

#### IEnumerator DataSource.DelayedInit (float *\_time*)[private]

Метод отложенной инициализации объекта Требуется в случае если дистрибутор не успевает инициализироваться до источника данных. В данной версии не применяется.

#### float DataSource.GetData ()

Метод получения текущих данных источника

##### Возвращает

Данные источника

#### float DataSource.GetRadius ()

Метод получения доверительного радиуса источника

##### Возвращает

Доверительный радиус источника

#### string DataSource.GetType ()

Метод получения типа данных источника

##### Возвращает

Тип данных источника

#### void DataSource.OnDrawGizmosSelected ()[private]

Служебная функция, рисующая каркас сферы действия источника.

#### void DataSource.Set (float *\_data*, string *\_type*)

Метод установки значения и типа источника.

##### Аргументы

|  |  |
| --- | --- |
| *[float]* | \_data Данные |
| *[string]* | \_type Тип данных |

#### void DataSource.SetData (float *\_data*)

Метод установки значения источника.

##### Аргументы

|  |  |
| --- | --- |
| *\_data* | Устанавливаемые данные |

### Данные класса

#### float DataSource.data = 0[private]

Текущие данные источника

#### bool DataSource.isAutonomous = false[private]

Если истинна, то объект не пытается переименоваться в Awake.

#### float DataSource.radius = 0[private]

Доверительный радиус источника

#### string DataSource.type = null[private]

Тип данных источника

#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**DataSource.cs**

## Класс Distributor

Синглтон-дистрибутор данных

##### Авторы

Пивко Артём

##### Версия

1.1

##### Дата

19.05.20

##### Предупреждения

Этот скрипт должен быть размещен на сцене в ЕДИНСТВЕННОМ экземпляре. Если его не будет, не будут работать никакие визуализаторы. Если их будет много, останется только один.

Этот класс составляет список всех источников данных на сцене и хранит её для передачи визуализаторам. Преимущественно объемным.

### Конструктор(ы)

#### Distributor.Distributor ()[private]

### Методы

#### void Distributor.Awake ()[private]

Функция, инициализурующая синглтон дистрибутор. Если ссылка на синглтон ещё не была инициализированна, то инициализирует её этим объектом. Если же она уже не пуста, то это значит что синглтор уже иницализирован. Тогда функция уничтожает данный объект и выводит предупреждение.

После этого производится поиск всех источников данных на сцене и ссылки на них заносятся в список sources. По списку источников формируется список MQTT-топиков этих источников. MQTT-топик источника это его имя. После этого подключается к MQTT брокеру по заданному в объекте IP-адресу с заданным в объекте идентификатором. Привязывает событие получения данных к обработчику. Затем подписывается на топики.

.

#### IEnumerator Distributor.DelayedRefresh (float *\_time*)[private]

#### static Distributor Distributor.GetInstance ()[static]

Функция-интерфейс для получения синглтона Возвращает ссылку на синглтон.

#### List<DataSource> Distributor.GetSources ()

Возвращает список источников в сцене.

#### int Distributor.RefreshSources ()

Функция, обновляющая список источников на сцене. Возвращает разницу в длинне старого и нового списка.

#### bool Distributor.SourceSet (string *topic*, float *data*, string *type*)

Метод установки данных и типа источника

##### Аргументы

|  |  |
| --- | --- |
| *topic* | MQTT-топик источника, он же его имя. |
| *data* | Данные |
| *type* | Тип данных Производит линейный поиск по списку имен источников и при нахождении устанавливает значение и тип источника равными аргументам. Если источник был найден, возвращает истину, иначе ложь. |

### Данные класса

#### Distributor Distributor.instance[static], [private]

Статичная ссылка на единственный экземпляр класса

#### List<DataSource> Distributor.sources[private]

Список всех источников данных на сцене

#### List<string> Distributor.sourcesnames = new List<string>()[private]

Список имен источников на сцене

#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**Distributor.cs**

## Класс ElementController

Класс для контроля элементов интерфейса, отображающих данные

##### Авторы

Пивко Артём

##### Версия

1.0

##### Дата

13.05.20

Интерфейс для установки параметров элементов интерфейса.

В случае, если требуется добавление типа даных, это делается в switch'e функции set и добавлением новой иконки в блоке Icon.

### Методы

#### void ElementController.Set ()

#### void ElementController.Set (string *type*, string *name*, string *data*)

Метод установки элемента

##### Аргументы

|  |  |
| --- | --- |
| *type* | Тип данных, определяющий иконку. |
| *name* | Отображаемое название данных |
| *data* | Отображаемые данные |

### Данные класса

#### Sprite ElementController.alarmIcon = null[private]

Ссылка на иконку тревоги. Должна быть задана в редакторе.

#### TextMeshProUGUI ElementController.dataText = null[private]

Ссылка на текстовый объект Text Mesh Pro, отображающий значение. Должна быть задана в редакторе.

#### Sprite ElementController.defaultIcon = null[private]

Ссылка на иконку по-умолчанию. Должна быть задана в редакторе.

#### Sprite ElementController.humIcon = null[private]

Ссылка на иконку влажности. Должна быть задана в редакторе.

#### Image ElementController.icon = null[private]

Ссылка на контролируемую иконку. Должна быть задана в редакторе.

#### TextMeshProUGUI ElementController.nameText = null[private]

Ссылка на текстовый объект Text Mesh Pro, отображающий название. Должна быть задана в редакторе.

#### Sprite ElementController.tempIcon = null[private]

Ссылка на иконку температуры. Должна быть задана в редакторе.

#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**ElementController.cs**

## Класс Flymode

Класс для перемещения аватара игрока полётом

##### Авторы

Пивко Артём

##### Версия

1.0

##### Дата

13.05.20

Перемещение осуществляется нажатием на заданную кнопку. При её нажатии аватар игрока начинает перемещаться по направлению от шлема к руке, на которой нажата кнопка. Чем дальше рука от шлема, тем быстрее летит аватар.

### Методы

#### void Flymode.Update ()[private]

Функция основного цикла. Если полёт разрешен, то по активации действия лететь в сторону от шлема к руке.

### Данные класса

#### bool Flymode.allowFlight = true

Разрешить/запретить передвижение полётом

#### SteamVR\_Action\_Boolean Flymode.flyAction

Действие SteamVR, отвечающее за полет.

#### Hand Flymode.leftHand[private]

Ссылка на левую руку аватара

#### Hand Flymode.rightHand[private]

Ссылка на правую руку аватара

#### float Flymode.speed = 10f

Скорость полёта

#### Camera Flymode.VRCamera[private]

Ссылка камеру виртуальной реальности

#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**Flymode.cs**

## Класс LaserPointerController

Класс для контроля лазерного указателя, сканера и меню.

##### Авторы

Пивко Артём

##### Версия

1.0

##### Дата

13.05.20

Этот класс занимается контролем инструментов взаимодействия пользователя с моделью: лазерного указателя для управления меню, сканера показаний визуализаторов и наручного меню. Для корректной работы скрипт требуется разместить на том же объекте, где находится скрипт руки виртуального аватара.

### Методы

#### void LaserPointerController.Awake ()[private]

Метод инициализации объекта В методе получется ссылка на SteamVR\_LaserPointer, на скрипт VR-руки Hand и на текстовые блоки интерфейса сканера.

#### void LaserPointerController.Scan ()[private]

Метод сканирования визуализаторов Метод производит проекцию луча вдоль отображаемого SteamVR\_LaserPointer луча, вычленяет из объекта, пересеченного лучом компонент, наследованный от **Visualiser**. После чего производится вызов метода Scan и вывод полученных данных на интерфейс сканера.

#### void LaserPointerController.Update ()[private]

Метод основного цикла. Производит включение лазера и меню при нажатии на кнопки и выключение при отпускании.

### Данные класса

#### string LaserPointerController.data[private]

Переменыне для временного хранения данных сканера.

#### string LaserPointerController.dataType[private]

#### SteamVR\_Input\_Sources LaserPointerController.handType[private]

Ссылка на текущую руку

#### SteamVR\_Action\_Boolean LaserPointerController.laserInput

Действие, включающее лазер

#### Canvas LaserPointerController.menuCanvas = null[private]

Ссылка на канвас меню на данной руке. Должна быть задана в редакторе.

#### SteamVR\_Action\_Boolean LaserPointerController.menuInput

Действие, включающее меню

#### int LaserPointerController.n = 4[private]

Размер массива ссылок

#### Canvas LaserPointerController.scanerCanvas = null[private]

Ссылка на канвас сканера на данной руке. Должна быть задана в редакторе.

#### SteamVR\_LaserPointer LaserPointerController.target[private]

Ссылка на скрипт SteamVR\_LaserPointer на данной руке

#### TextMeshProUGUI [] LaserPointerController.text = null[private]

Массив ссылок на блоки интерфейса сканера.

#### string LaserPointerController.topic[private]

#### string LaserPointerController.visType[private]

#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**LaserPointerController.cs**

## Класс MonitorController

Класс для контроля интерфейса монитора оборудования

##### Авторы

Пивко Артём

##### Версия

1.0

##### Дата

14.05.20

Этот класс занимается контролем текстового интерфейса, отображающего все данные, имеющиеся о единице оборудования.

Для включения/выключения интерфейса используйте метод TurnTV через кнопки на оборудовании.

### Методы

#### void MonitorController.Awake ()[private]

Метод инициализации объекта. Собирает списки источников и тревог, ищет в поддереве подконтрольный ScrollRect и запоминает ссылку на него, затем запускает метод обновления интерфеса. В случае если объект дочерний, запускает сопрограмму отложенного сбора списка тревог.

#### void MonitorController.ClearElements ()[private]

Метод очистки всех подъобъектов списка элементов интерфейса.

#### void MonitorController.CollectAlarms ()[private]

Метод получения списка тревог в поддереве

#### void MonitorController.CollectSources ()[private]

Метод получения списка источников в поддереве

#### void MonitorController.CreateElement ()

Метод создания и инициализации элемента интерфейса заданными данными

#### void MonitorController.CreateElement (string *type*, string *name*, string *data*)[private]

Метод создания и начальной инициализации элемента интерфейса на экране

##### Аргументы

|  |  |
| --- | --- |
| *type* | Начальный отображаемый тип данных |
| *name* | Начальное отображаемое название измерения |
| *data* | Начальные отображаемые данные. Создает элемент интерфейса на экране, установка его полей переданными в функцию параметрами и регулировка высоты зоны, в которой элемент размещается. |

#### IEnumerator MonitorController.DelayedAlarmCollect ()[private]

Сопрограмма отложенного сбора списка тревог Используется дочерними мониторами для получения полного списка тревог, перекрывающих монитор. Задержка нужна для того чтобы вышестоящий объект успел собрать свой список тревог.

#### List<Callout> MonitorController.GetAlarms ()

Метод получения списка визуализаторов тревог данного объекта. Используется дочерними мониторами.

#### void MonitorController.RefreshUI ()[private]

Метод обновления элементов интерфейса. В отличии **UpdateList()** не уничтожает элементы перед реинициализацией, а только обновляет их значения. Осторожно, не добавляет новых элементов, но может удалять старые. Можно использовать в процессе работы программы.

#### void MonitorController.TurnAlarms (bool *on*)

Метод включения/выключения тревог в поддереве

##### Аргументы

|  |  |
| --- | --- |
| *on* | Включить (True) или выключить (False) |

#### void MonitorController.TurnTV (bool *on*)

Метод включения/выключения экрана монитора оборудования

##### Аргументы

|  |  |
| --- | --- |
| *on* | Включить (True) или выключить (False) |

#### void MonitorController.Update ()[private]

Метод основного цикла. Каждый кадр обновляет значения элементов интерфейса.

#### void MonitorController.UpdateList ()[private]

Метод полного обновления элементов интерфейса. Полностью уничтожает и реинициализирует все элементы интерфейса монитора. Применяется для инициализации монитора. Не рекомендуется использовать во время работы программы из-за медлительности, используйте **RefreshUI()**

### Данные класса

#### List<Callout> MonitorController.alarms[private]

Список сигналов тревоги, которые могут перекрывать монитор

#### GameObject MonitorController.element = null[private]

Ссылка на префаб элемента интерфейса. Должна быть задана в редакторе.

#### List<ElementController> MonitorController.elements[private]

Список элементов интерфейса, подконтрольных объекту.

#### bool MonitorController.isChild = false[private]

Отметка является ли данный монитор подчиненным другому монитору

#### List<DataSource> MonitorController.sources[private]

Список источников в поддереве этого объекта

#### ScrollRect MonitorController.target = null[private]

Ссылка ScrollRect, в котором размещены элементы интерфейса. Должна быть задана в редакторе.

#### Canvas MonitorController.tv = null[private]

Ссылка на канвас монитора. Должна быть задана в редакторе.

#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**MonitorController.cs**

## Класс NodeVisualisator

Класс экземпляров объемных визуализаторов

##### Авторы

Пивко Артём

##### Версия

1.1

##### Дата

14.05.20

##### Предупреждения

В случае если на сцене будет отсутствовать дистрибутор данных, объект не будет функционировать.

Этот класс занимается динамическим изменением цвета родительского объекта в соотвествии с данными из источника данных **DataSource** source и настроек конвертации температуры из данного файла.

### Методы

#### void NodeVisualisator.Awake ()[private]

Установочная функция, вызываемая в момент активации объекта на сцене. Вызывает сопрограмму DelayedInit ввиду необходимости инициализации с задержкой. Подробнее в описании DelayedInit.

#### IEnumerator NodeVisualisator.DelayedInit ()[private]

Установочная сопрограмма. Выполняет получение ссылки на материал данного объекта, получает ссылку на экземпляр глобального дистрибутора данных и запускает бесконечную сопрограмму DelayedRefresh. Перед выполнением всех вышеописанных действий ожидает 3 секунды для того чтобы дистрибутор успел собрать сведения об источниках на сцене.

#### IEnumerator NodeVisualisator.DelayedRefresh ()[private]

Сопрограмма-таймер основного цикла. Отвечает за регулярное обновление отображаемых данных в соотвествии с данными источника.

В бесконечном цикле внутри сопрограммы вызывается функция Init, после чего производится ожидание на время обновления до повторения цикла.

#### override string NodeVisualisator.Scan (out string *visType*, out string *dataType1*, out string *topic*)[virtual]

Метод снятия показаний с визуализатора сканером

##### Аргументы

|  |  |
| --- | --- |
| *visType* | Тип визуализатора |
| *dataType* | Тип данных |
| *topic* | MQTT-топик данных |

##### Возвращает

Визуализируемые данные в формате float В случае если визуализатор находится вне зоны дейстия источников, в качестве типа данных будет передано сообщение об этом, а топик передан не будет. В случае если визуализатор слушает только один источник, будет передан его топик. В случае если несколько, будет передано сообщение о смешанных топиках.

Внимание, передаваемые в метод параметры при его вызове должны стоять после ключегого слова out.

Замещает **Visualiser** (*стр.pagenum*).

#### void NodeVisualisator.UpdateObject ()[private]

Функция обновления отображаемого значения объекта Функция использует данные из валидных источников для обновления значения отображаемых данных.

• Если подходящих источников источников нет, то объект окрашивается в серый цвет и Log выводится сообщение "Node [X,Y,Z]: No valid sources detected", где X, Y, Z - номера объекта в матрице, созданной скриптом **CubeCreator.cs**.

• Если подходдящий источник один, то визуализатор отображает данный с него.

• Если подходящих источников несколько, то значение, отображаемое объектом вычисляется на основе удаленности от источников следующим образом:

L - расстояние между объектом и источником

R - Доверительный радиус источника

X - Данные на источнике

b = 1 - (L / R) - Ненормализованный вес источника для данного объекта

a = b \* Х - взвешенные ненормализованные данные с источника

sumk - сумма b всех источников

sum - сумма всех а

k = 1 / sumk - коэфициент нормализации

y = sum \* k - Итоговое значение данных на источнике

Пример: есть 2 источника с радиусами R1 = R2 = 7. Источник 1 показывает X1 = 10, источник 2 показывает X2 = 20. Куб находится на расстоянии a1=4 от источника 1 и на расстоянии a2=6 от источника 2.

1) Считаем долю от радиуса каждого куба.

A1 = a1/R1 = 4/7 = 0.57

A2 = a2/R2 = 6/7 = 0.86

2) Вычитаем их из единицы

B1 = 1-A1 = 1-0.57 = 0.43

B2 = 1-A2 = 1-0.86 = 0.14

3) Ищем коэффициент нормализации:

K = 1/(B1+B2) = 1/(0.14+0.43) = 1/0.57 = 1.75

4) Считаем финальное значение:

Y = (X1\*B1 + X2\*B2)\*K = (10\*0.43 + 20\*0.14)\*1.75 = 12.4.

#### void NodeVisualisator.UpdateValidSources ()[private]

Функция обновления списка актуальных источников Обнуляет текущий список актуальных источников, получает из дистрибутора список из ВСЕХ источников. Затем, в случае если этот список не пуст, сравнивает доверительный радиус источников с расстоянием до этого источника. Если источник покрывает этот объект, то он считается для него актуальным и заносится в список validSources.

### Данные класса

#### float NodeVisualisator.curData = 0[private]

Текущее значение визуализируемой информации на этом визуализаторе

#### string NodeVisualisator.dataType = null

Тип данных, визуализируемых данным объектов (какой параметр запрашивать у источника)

#### Material NodeVisualisator.material = null[private]

Ссылка на текущий материал данного визуализатора. Устанавливается не префабной связью а в Awake ввиду того что иначе изменяется глобальный материал.

#### float NodeVisualisator.refreshTime = 10f[private]

Время обновления визуализатора

#### List<DataSource> NodeVisualisator.validSources[private]

Актуальный список источников, влияющих на данный визуализатор.

#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**NodeVisualisator.cs**

## Класс Reciever

Класс для получения данных с сервера через MQTT.

##### Авторы

Пивко Артём, Стрельцов Григорий

##### Версия

1.0

##### Дата

14.05.20

Этот визуализатор занимается динамическим изменением цвета родительского объекта в соотвествии с данными из источника данных **DataSource** source. Для конверсии используется синглтон UniversalController

### Методы

#### void Reciever.Awake ()[private]

Метод инициализаии объекта Вызывает сопрограмму отложенной инициализации.

#### void Reciever.Connect ()[private]

Функция создания клиента MQTT и подключения его к брокеру Задает такие параметры подключения, как IP адрес MQTT брокера, порт, ID клиента, его догин и пароль, а также топик, на которых необходимо подписаться. Далее по заданным параметрам создает MQTT клиента, привязывает Callback функцию получения сообщений к функции OnMessage, затем производит подключение клиента к брокеру и подписывается на необходимый топик.

#### IEnumerator Reciever.DelayedInit ()[private]

Сопрограмма отложенной инициализаии объекта Получает ссылку на глобальный дистрибутор данных и запускает подключение к серверу по MQTT. Задержка в инициализации требуется чтобы а) Дистрибутор успел инициализироваться и б) чтобы тот успел собрать все источники на сцене.

#### void Reciever.OnMessage (object *sender*, MqttMsgPublishEventArgs *e*)[private]

Функция обработки полученных сообщений По приходу нового JSON сообщения парсит его, а затем производит поиск по датчикам, имена которых содержат специальное ключевое слово, которое позволяет идентифицировать те датчики, которые необходимо визуализировать. Если имя датчика содержит данное слово, то, значение с этого датчика, имя устроства, на котором стоит датчик, а также имя самого датчика передаются в функцию SourceSet(). Затем происходит проверка наличия триггера у датчика и, при наличии триггера, передает в функцию SourceSet() значение триггера, а также его имя и имя устройства, на котором возникла ошибка.

Если имя датчика содержит ключевое слово

Если тип сообщения цифровой

Если к датчику привязан триггер

### Данные класса

#### string Reciever.brocker\_ip = "134.209.224.248"

IP-адрес MQTT брокера. Может задаваться через редактор.

#### MqttClient Reciever.client1[private]

Объект клиента MQTT.

#### string Reciever.clientId = "qwerty"

ID MQTT клиента. Может задаваться через редактор.

#### Distributor Reciever.distributor[private]

Ссылка на глобальный дистрибутор данных

#### string Reciever.keyword = "Python"

Ключевое слово, которое идентифицирует датчики, которые необходимо визуализировать. Может задаваться через редактор.

#### string Reciever.password = "qwerty123456"

Пароль MQTT клиента. Может задаваться через редактор.

#### string Reciever.topic = "/analyzer\_data"[private]

Топик данных, поступающих в реальном времени

#### string Reciever.username = "collector"

Логин MQTT клиента. Может задаваться через редактор.

#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**Reciever.c**

## Класс ScrollBarButton

Класс, исправляющий работу SteamVR\_LaserPointer.

##### Авторы

Пивко Артём

##### Версия

1.0

##### Дата

14.05.20

Этот класс используется для исправления не полностью функционирующего лазерного указателя SteamVR. Должен быть размещен на том же объекте что и SteamVR\_LaserPointer.

### Методы

#### void SteamVRLaserWrapper.Awake ()[private]

Установочная функция, вызываемая в момент активации объекта на сцене.

Получает ссылку на SteamVR\_LaserPointer и привязывает кастомные обработчики событий.

#### void SteamVRLaserWrapper.OnPointerClick (object *sender*, PointerEventArgs *e*)[private]

Обработчик события клика лазером

#### void SteamVRLaserWrapper.OnPointerIn (object *sender*, PointerEventArgs *e*)[private]

Обработчик события начала касания интерактивного элемента и лазера

#### void SteamVRLaserWrapper.OnPointerOut (object *sender*, PointerEventArgs *e*)[private]

Обработчик события окончания касания интерактивного элемента и лазера

### Данные класса

#### SteamVR\_LaserPointer SteamVRLaserWrapper.steamVrLaserPointer[private]

Ссылка на SteamVR\_LaserPointer.

#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**SteamVRLaserWrapper.cs**

## Класс StreamVisualiser

Класс для контроля потоковых визуализаторов

##### Авторы

Пивко Артём

##### Версия

1.0

##### Дата

14.05.20

##### Предупреждения

Функционирует постоянно. Не только во время работы программы, но и в редакторе.

Этот класс визуализатор использует систему частиц для визуализации "потоковых" данных, таких как параметры электропроводки, водяного или воздушного охлаждения. Система частиц должна иметь включенными модули Emission, Shape, Trails и Renderer. Рекомеднуется использовать только как часть префаба SteamVisualiser

### Методы

#### void StreamVisualiser.Awake ()[private]

Метод инициализации объекта Находит в поддереве объекта систему частиц и коллайдер, затем запускает сопрограмму основного цикла.

#### Color StreamVisualiser.DataToColor (float *data*)[private]

Метод преобразования данных в цвет потока

##### Аргументы

|  |  |
| --- | --- |
| *data* | Входные данные |

##### Возвращает

Выходные данные Используется преобразование с помощью **UniversalConverter**

#### float StreamVisualiser.DataToDencity (float *data*)[private]

Метод преобразования данных в плотность потока

##### Аргументы

|  |  |
| --- | --- |
| *data* | Входные данные |

##### Возвращает

Выходные данные

#### float StreamVisualiser.DataToSpeed (float *data*)[private]

Метод преобразования данных в скорость потока

##### Аргументы

|  |  |
| --- | --- |
| *data* | Входные данные |

##### Возвращает

Выходные данные

#### IEnumerator StreamVisualiser.DelayedUpdate (float *\_time*)[private]

Сопрограмма-таймер основного цикла.

##### Аргументы

|  |  |
| --- | --- |
| *\_time* | Время обновления Отвечает за регулярное обновление данных потока с источников |

#### void StreamVisualiser.OnDrawGizmos ()[private]

Служебная функция, необходимая для отображения границ системы частиц и координат начала и конца в редакторе

#### void StreamVisualiser.RefreshFromSources ()[private]

Метод обновления данных на основе источников

#### override string StreamVisualiser.Scan (out string *visType*, out string *dataType*, out string *topic*)[virtual]

Метод снятия показаний с визуализатора сканером

##### Аргументы

|  |  |
| --- | --- |
| *visType* | Возвращается конструкция вида Тип\_источника\_скорости:Значение |
| *dataType* | Тип данных источника цвета |
| *topic* | Возвращается конструкция вида Тип\_источника\_плотности:Значение |

##### Возвращает

Визуализируемые данные цвета в формате float Так как потоковый визуализатор это единстенный визуализатор, использующий несколько разных источников с разными независимыми данными, его сканирование отличается от сканирования остальных визуализаторов. Из-за ограниченности сканера его текстовые строки приходится использовать для отображения нескольких данных.

Внимание, передаваемые в метод параметры при его вызове должны стоять после ключегого слова out.

#### void StreamVisualiser.Update ()[private]

Функция основного цикла. 1) Размещает систему частиц на точке начала и направляет её в точку конца

2) Рассчитывает время жизни частиц так, чтобы они долетали от начала до конца потока

3) Устанавливает частоту испускания частиц так, чтобы соблюдалась необходимая плотность потока

4) Устанавливает цвет выпущенных частиц так, чтобы соблюдался необходимый цвет потока

5) Устанавливает радиус основания излучателя частиц так, чтобы соблюдался необходимый радиус потока

6) Устанавливает цвет выпущенных частиц так, чтобы соблюдался необходимый цвет потока

7) Устанавливает хвосты у выпущенных частиц, если это опция активирована в настройках потока

8) Устанавливает положение коллайдера вдоль потока

### Данные класса

#### GameObject StreamVisualiser.collObj = null[private]

Ссылка на объект, служащий коллайдером визуализатора для сканера. Задается в редакторе.

#### float StreamVisualiser.dencData[private]

Текущая плотность потока

#### DataSource StreamVisualiser.dencitySource = null[private]

Ссылка на источник данных о плотности потока

#### Transform StreamVisualiser.finish = null[private]

Ссылка на точку окончания потока. Задается в редакторе.

#### Color StreamVisualiser.finishColor

Цвет конца потока по-умолчанию. Задается в редакторе.

#### float StreamVisualiser.finishColorData[private]

Текущие данные, определяющие цвет конца потока

#### DataSource StreamVisualiser.finishColorSource = null[private]

Ссылка на источник данных о цвете потока

#### bool StreamVisualiser.isControlledBySources = true

Контролируется ли поток источниками данных?

#### float StreamVisualiser.particleDencityMultiplier = 2f

Множитель плотности потока по-умолчанию. Задается в редакторе.

#### ParticleSystem StreamVisualiser.ps[private]

Ссылка на контролируемую систему частиц.

#### float StreamVisualiser.radius = 2f

Радиус потока

#### float StreamVisualiser.refreshTime = 1f

Время обновления данных с источников

#### float StreamVisualiser.speed = 5f

Скорость потока по-умолчанию. Задается в редакторе.

#### float StreamVisualiser.speedData[private]

Текущая скорость потока

#### DataSource StreamVisualiser.speedSource = null[private]

Ссылка на источник данных о скорости потока

#### Transform StreamVisualiser.start = null[private]

Ссылка на точку начала потока. Задается в редакторе.

#### Color StreamVisualiser.startColor

Цвет начала потока по-умолчанию. Задается в редакторе.

#### float StreamVisualiser.startColorData[private]

Текущие данные, определяющие цвет начала потока

#### DataSource StreamVisualiser.startColorSource = null[private]

Ссылка на источник данных о цвете потока

#### bool StreamVisualiser.trails = false

Оставляют ли частицы хвосты?

#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**StreamVisualiser.cs**

## Класс UniversalConverter

Синглтон-конвертер входных данных в настройки визуализаторов

##### Авторы

Пивко Артём

##### Версия

1.0

##### Дата

8.04.20

##### Предупреждения

Этот скрипт должен быть размещен на сцене в ЕДИНСТВЕННОМ экземпляре. Если его не будет, не будут работать никакие визуализаторы. Если их будет много, останется только один.

Для работы скрипта требуется заполнение в инспекторе настроек преобразовнаия цветов.

Этот класс занимается превращением цифровых данных с источников (температуры, влажности и т. д.) в параметры для визуализатров в Unity (Материалы различных цветов, ... ).

### Конструктор(ы)

#### UniversalConverter.UniversalConverter ()[private]

### Методы

#### void UniversalConverter.Awake ()[private]

Функция, инициализурующая статическую ссылку на данный объект. Если ссылка ещё не была инициализированна, то инициализирует её этим объектом. Если же она уже не пуста, то это значит что синглтор уже иницализирован. Тогда функция уничтожает данный объект и выводит предупреждение.

#### static UniversalConverter UniversalConverter.GetInstance ()[static]

Функция-интерфейс для получения синглтона Возвращает ссылку на синглтон.

#### string UniversalConverter.GetLocation ()

Функция, возвращающая строку с локацией синглтона. Требуется для обнаружения несанкционированных экземпляров синглтона в случае если самоуничтожился неправильный синглтон.

#### Color UniversalConverter.TempToColor (float *\_temp*)

Интерфейс к конвертеру значения температуры в цвет, использующий настройки по-умолчанию.

##### Аргументы

|  |  |
| --- | --- |
| *\_temp* | Конвертируемая температура |

##### Возвращает

Конвертированный в соответствии с температурой цвет

#### Color UniversalConverter.TempToColor (float *\_temp*, string *generalType*)

Конвертер значения float в цвет в соответствии с выбранными настройками.

##### Аргументы

|  |  |
| --- | --- |
| *\_temp* | Конвертируемые данные |
| *generalType* | название используемых при конвертации настроек (обычно - имя типа данных) |

##### Возвращает

Конвертированный цвет Метод использует значения настроек цветов из массива tSettings. Если настройки с именем не generalType найдены, используются первые настойки из массиваtSettings и в консоль выводится сообщение об ошибке. Далее входные данные \_temp нормализуются и нормализованное число аппроксимируется по градиенту настроек, получая нужный цвет.

#### Color UniversalConverter.TempToColorOld (float *\_temp*)

Устаревиший конвертер значения температуры в цвет.

##### Аргументы

|  |  |
| --- | --- |
| *\_temp* | Конвертируемая температура |

##### Возвращает

Конвертированный в соответствии с температурой цвет Рекомендуется использовать вместо этого метода TempToColor

Метод использует значения настроек цветов из этого файла как границы перехода цветов:

• Если температура <= absCold, то цвет будет чистым синим с обычной прозрачностью

• Если absCold <= температура < allowedStart, то цвет будет голубым и приблежаться к бирюзовому с ростом температуры. Прозрачность нормальная

• Если allowedStart <= температура < optimalStart, то цвет будет бирюзовым и приблежаться к зеленому с ростом температуры. Прозрачность тоже будет расти до полного исчезновения

• Если optimalStart <= температура < optimalFinish, Объект полностью прозрачен

• Если optimalFinish <= температура < allowedFinish, то цвет будет желтым и приблежаться к оранжевому с ростом температуры. Прозрачность тоже будет падать до нормального значения

• Если allowedFinish <= температура < absHeat, то цвет будет оранжевым и приблежаться к красному с ростом температуры. Прозрачность нормальная

• Если температура > absCold, то цвет будет чистым красным с обычной прозрачностью

#### void UniversalConverter.ValidateSettings ()

Метод, инициализирующий все текущие настройки преобразования цветов Задача этого метода - превратить настройки преобразования цветов в цветовой градиент, по которому потом будет происходить преобразование.

Метод проходит по массиву настроек температуры и для каждого набора настроек нормализует значения границ относительно интервала [0, 1]. После этого каждый цвет из настроек устанавливается в соответствующее место в градиенте.

### Данные класса

#### float UniversalConverter.absCold = 20f[private]

Температура, ниже которой не отслеживается. Устарело.

#### float UniversalConverter.absHeat = 100f[private]

Температура, выше которой не отслеживается. Устарело.

#### float UniversalConverter.allowedFinish = 85f[private]

Граница верхней недопустимой температуры. Устарело.

#### float UniversalConverter.allowedStart = 35f[private]

Граница нижней недопустимой температуры. Устарело.

#### UniversalConverter UniversalConverter.instance[static], [private]

Статичная ссылка на единственный экземпляр класса

#### float UniversalConverter.maxTransp = 0.4f[private]

Максимальное значение прозрачности объекта. Устарело.

#### float UniversalConverter.optimalFinish = 70f[private]

Верхняя граница оптимальной температуры. Устарело.

#### float UniversalConverter.optimalStart = 50f[private]

Нижняя граница оптимальной температуры. Устарело.

#### TempSettings [] UniversalConverter.tSettings

Массив различных настроек преобразования цветов. Внимание, настройки цветов должны располагаться в порядке возрастания границ.

#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**UniversalConverter.cs**

## Класс UniversalTranslator

Класс для преобразования типов, получения единиц измерения и переводов.

##### Авторы

Пивко Артём

##### Версия

1.1

##### Дата

27.05.20

##### Предупреждения

Этот скрипт должен быть размещен на сцене в ЕДИНСТВЕННОМ экземпляре. Если его не будет, не будут работать никакие визуализаторы. Если их будет много, останется только один.

Для корректной работы компонента требутеся заполнить словари в редакторе. 4

##### Необходимо сделать:

Сделать TranstypeToRussian

Этот класс-синглтон используется как глобальный сервис, выполняющий 3 функции:

1) Преобразование имени датчика в имя типа

2) Преобразование имени типа в символ, отражающий единицу измерения (например, символ процента "%")

3) Перевод имени датчика в понятное название

### Конструктор(ы)

#### UniversalTranslator.UniversalTranslator ()[private]

### Методы

#### void UniversalTranslator.Awake ()[private]

Функция, инициализурующая статическую ссылку на данный объект. Если ссылка ещё не была инициализированна, то инициализирует её этим объектом. Если же она уже не пуста, то это значит что синглтор уже иницализирован. Тогда функция уничтожает данный объект и выводит предупреждение.

#### static UniversalTranslator UniversalTranslator.GetInstance ()[static]

Функция-интерфейс для получения синглтона Возвращает ссылку на синглтон.

#### string UniversalTranslator.TransGeneralTypeToUnit (string *type*)

Метод преобразования типа в единицу измерения

##### Аргументы

|  |  |
| --- | --- |
| *type* | Тип данных |

##### Возвращает

символ(ы) единицы измерения в формате string Ищет в словаре generalTypeToUnitDictionary ключ, полученный в аргументе type и возвращает его значение.

В случае, если ключ не был найден, выводит ошибку в консоль и возвращает пустую строку, что позволяет оторбражать величину просто как числовое значение.

#### string UniversalTranslator.TransTypeToGeneralType (string *type*)

Метод преобразования имени датчика в тип

##### Аргументы

|  |  |
| --- | --- |
| *type* | Имя датчика |

##### Возвращает

Общий тип данных датчика string Ищет в словаре typeToGeneraTypeDictionary ключ, полученный в аргументе type и возвращает его значение.

При заполнении словаря можно использовать специальный синтаксис: если ключ словаря начинается с символа вопросительного знака "?", всё что находится после него испольщуется не как ключ, а как строка для поиска. Если она присутствует в полученном в метод типе, возвращается значение этой пары.

Например, если как ключ поставить "?PythonRoomT", а как значение roomtemp, то при принятии аргумента "PythonRoomT5" будет возвращено roomtemp.

#### string UniversalTranslator.TranstypeToRussian (string *type*)

Метод преобразования типа в единицу измерения

##### Аргументы

|  |  |
| --- | --- |
| *type* | Имя датчика |

##### Возвращает

Переведенная строка в формате string Ищет в словаре typeToRussianDictionary ключ, полученный в аргументе type и возвращает его значение.

В случае, если ключ не был найден, выводит ошибку в консоль и возвращает входную строку, что позволяет оторбражать имя датчика без перевода.

### Данные класса

#### MyDictionary2 UniversalTranslator.generalTypeToUnitDictionary

Словарь перевода типа в символ единицы измерения. Заполняется в редакторе

#### UniversalTranslator UniversalTranslator.instance[static], [private]

Статичная ссылка на единственный экземпляр класса

#### MyDictionary2 UniversalTranslator.typeToGeneraTypeDictionary

Словарь перевода названия датчика в его тип. Заполняется в редакторе

#### MyDictionary2 UniversalTranslator.typeToRussianDictionary

Словарь перевода названия датчика в описание. Заполняется в редакторе

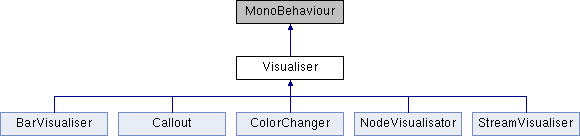
#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**UniversalTranslator.cs**

## Класс Visualiser

Родительский класс всех объектов для визуализации

Граф наследования:Visualiser:



##### Авторы

Пивко Артём

##### Версия

1.0

##### Дата

13.05.20

От этого абстрактного класса наследуются все объекты визуализации.

### Методы

#### abstract string Visualiser.Scan (out string *visType*, out string *dataType*, out string *topic*)[pure virtual]

Универсальный метод снятия показаний с визуализатора

##### Аргументы

|  |  |
| --- | --- |
| *visType* | Тип визуализатора |
| *dataType* | Тип данных |
| *topic* | MQTT-топик данных |

##### Возвращает

Визуализируемые данные в формате float

Унаследованные реализации функции обязаны будут возвращать 4 параметра: опорные данные визуализатора, тип визуализатора, тип данных, MQTT-топик данных.

Функция необходима для работы сканера визуализаторов. Внимание, передаваемые в метод параметры при его вызове должны стоять после ключегого слова out.

Замещается в **NodeVisualisator**, **StreamVisualiser**, **ColorChanger** (*стр*), **BarVisualiser** и **Callout**

#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**Visualiser.cs**

## Класс VRUIItem

Класс для автоматизированного создания коллайдеров на элементах интерфейса

##### Авторы

Пивко Артём

##### Версия

1.0

##### Дата

14.05.20

##### Предупреждения

Требует компонента RectTransform на объекте для работы

Коллайдеры требуются для управления интерфейсом VR контроллерами.

### Методы

#### void VRUIItem.OnEnable ()[private]

Установочная функция

#### void VRUIItem.OnValidate ()[private]

Установочная функция при обновлении объекта

#### void VRUIItem.ValidateCollider ()[private]

Установочная функция создания или проверки правильности коллайдера.

### Данные класса

#### BoxCollider VRUIItem.boxCollider[private]

Ссылка на создаваемый коллайдер

#### RectTransform VRUIItem.rectTransform[private]

Ссылка на RectTransform объекта, к которому создается коллайдер

#### Объявления и описания членов класса находятся в файле:

D:/GitProjects/VR\_Data\_Center/Assets/Scripts/**VRUIItem.cs**