УТВЕРЖДЕН

46.07622667.00001-01 12 04-3 90 01-ЛУ

ПОДСИСТЕМА СОЗДАНИЯ 3D СЦЕН.

РЕДАКТОР 3D СЦЕН

IIOT.ISTOK 3D EDITOR

Инструкция по работе

46.07622667.00001-01 12 04-3 90 01

Листов 37

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

2020

Литера

АННОТАЦИЯ

Настоящий документ является инструкцией по работе с 3D IIoT.ISTOK 3D Editor (далее по тексту – Редактор 3D сцен или IIoT.ISTOK 3D Editor) и описывает подходы, функции и методы программирования поведения объектов на трехмерных сценах в IIoT.ISTOK.

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

1.1. Цифровой двойник

Трехмерные сцены в IIoT.ISTOK используются для создания цифровых двойников (см. рис. 1), которые повторяют поведение объекта или процесса.

Цифровой двойник – это компьютерное 3D представление конкретного физического изделия, группы изделий, механического или технологического процесса, который включает не только трехмерную геометрию, технические характеристики и текущие параметры работы, но и другую важную информацию – окружающую среду и условия эксплуатации, техническое состояние и наработку, взаимодействие с другими объектами, данные предиктивной аналитики, в том числе, по прогнозированию отказов и сбоев. Цифровой двойник может быть как упрощенным, так и очень детальным и отражать широкий спектр самых разных характеристик как самого изделия, так и технологических и производственных процессов.

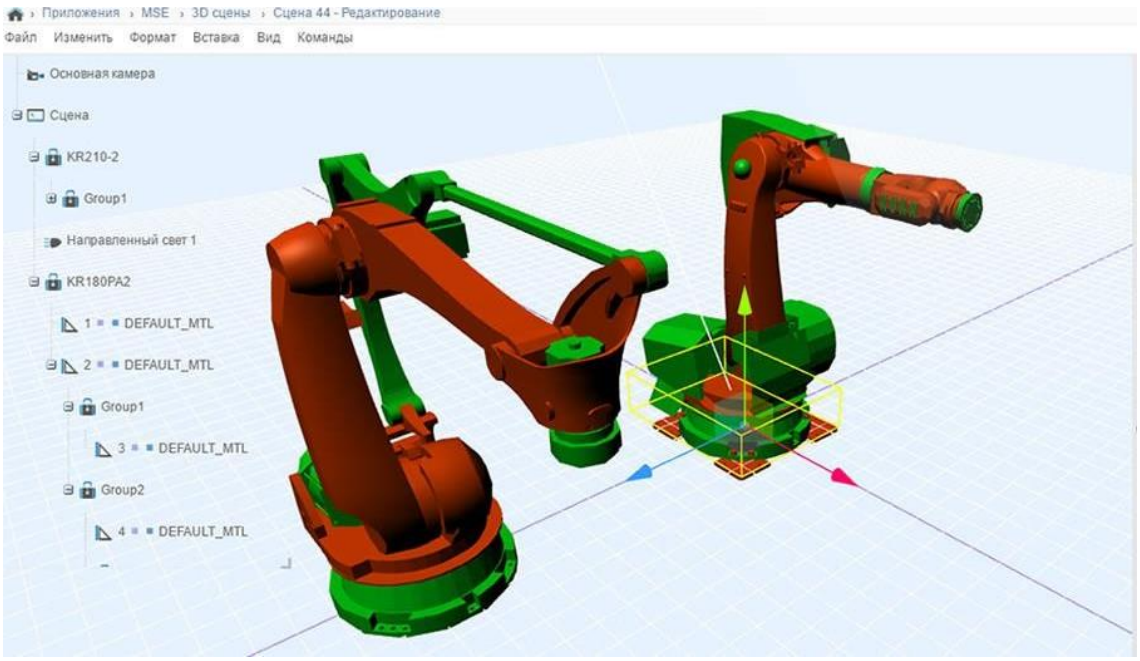

Рисунок 1 – Трехмерная сцена

Наличие трехмерного Цифрового двойника помогает организовать связь изделия с подключенными к нему объектами, программным обеспечением, отвечающим за управление изделием, контроль рабочего состояния и процесса эксплуатации и т.д. Цифровой двойник представляет особую

ценность, когда он наиболее точно отображает реальное состояние и рабочие характеристики своего физического прообраза. Какими бы точными, детальными и проработанными не были действия на этапах проектирования, моделирования и подготовки производства, в реальной жизни, как правило, процессы протекают немного иначе и именно Цифровой двойник способен выступить тем самым мостиком к необходимой информации о реальной эксплуатации изделий. Данную информацию можно использовать по-разному, например, для оценки узких мест, возможностей для улучшений и изменений, подтверждения целесообразности изменений и т.д. Поскольку цифровой двойник – это трехмерный объект, его работа с ним для человека гораздо понятнее, чем работа с любыми таблицами или графиками, он позволяет заглянуть внутрь реального физического объекта непосредственно во время работы без необходимости остановки оборудования и открытия панелей, которые закрывают доступ к узлам, требующим проверки.

1.2. Основные элементы трехмерной сцены

Создание Цифрового двойника происходит путем моделирования поведения объектов на трехмерной сцене. Элементы интерфейса трехмерной сцены содержат:

1) основное меню (позиция 1, рис. 2);
2) дерево объектов сцены (позиция 2, рис. 2);
3) основное подменю (позиция 3, рис. 2).

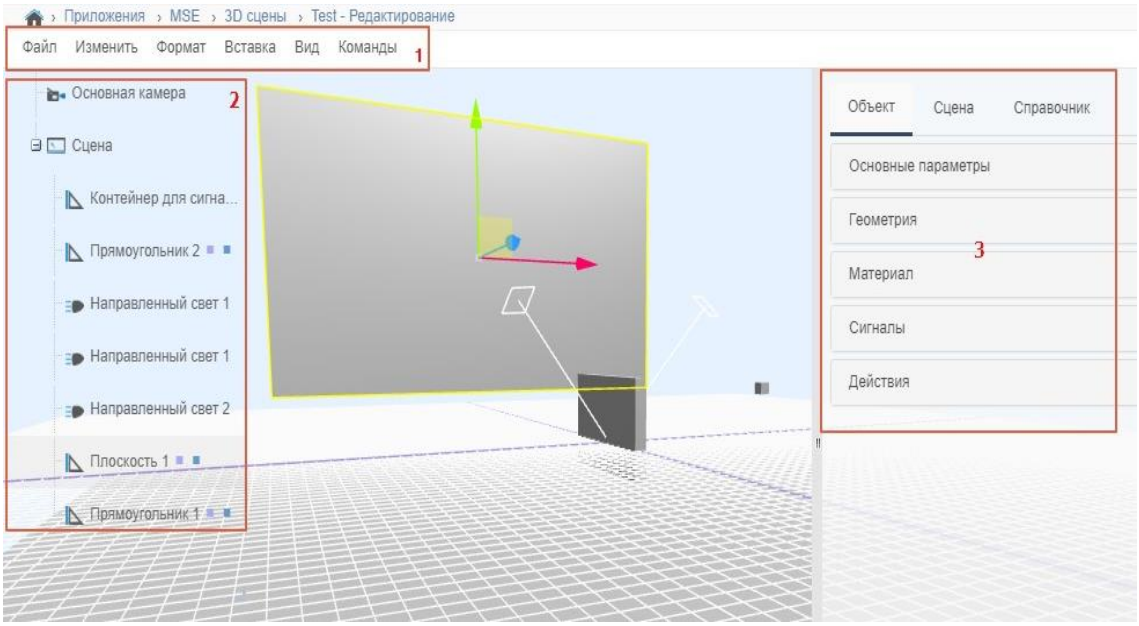

Рисунок 2 – Элементы интерфейса для создания трехмерной сцены

Дерево объектов сцены (позиция 2, рис. 2) является одним из наиболее важных элементов, сцены. В дереве выполняется группировка объектов (включение одного объекта в другой), что в дальнейшем требуется для упрощения процесса моделирования поведения объектов.

1.2.1. Работа с объектами

1.2.1.1. Выполнение группировки объекта

Для выполнения группировки объектов необходимо перетащить мышкой один объект в другой. Для изменения порядка размещения объектов в дереве, нужно выбрать объект и переместить его мышкой на нужный уровень в дереве.

1.2.1.2. Добавление и создание объекта

Добавление или создание объектов на трехмерной сцене может быть выполнено следующими способами:

– создание простых геометрических примитивов (основное меню – вставка);

– создание геометрических объектов путем программирования (подменю – объект – действия);

– загрузка геометрии в нейтральном формате из CAD и другого специализированного ПО (основное меню – файл – импорт). Поддерживаемые форматы: STL, VRML, OBJ, Blender, Collada и json;

– вставка геометрии из справочника активов, (см. рис. 3) (подменю «Справочник»). Готовые справочники различных трехмерных объектов (роботы, оборудование, датчики, мебель и т.д. доступны для скачивания на портале).

При использовании трехмерной геометрии рекомендуется избегать высокой точности моделей и большого количества поверхностей так как это может сказываться на производительности отрисовки сцены на слабых компьютерах и мобильных устройствах.

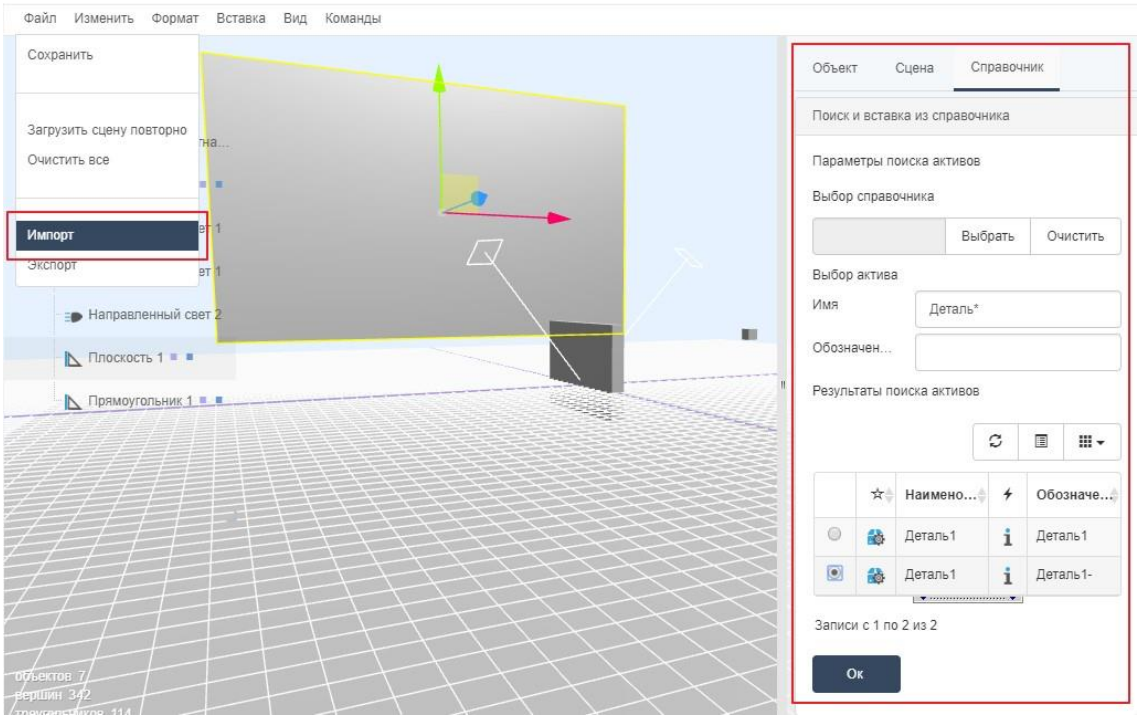

Рисунок 3 – Вставка геометрии из справочника активов

1.3. Работа с сигналами

Любые сигнал всех изделий, которые связаны с приложением, можно использовать в трехмерных сценах. Для назначения сигналов на объект сцены нужно выбрать объект, раскрыть подменю «Сигналы» и выбрать операцию «Добавить новый сигнал» (см. рис. 4).

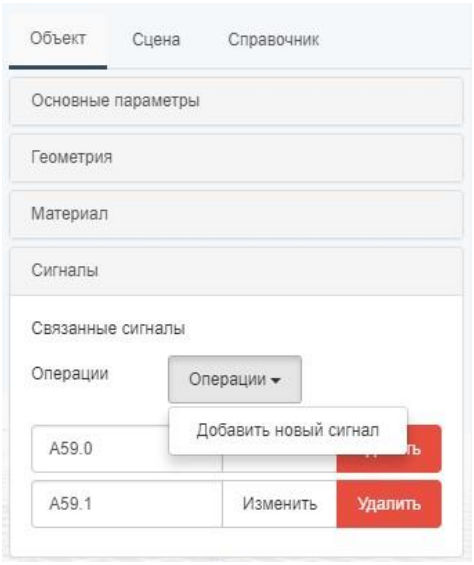

Рисунок 4 – Подменю «Сигналы». Добавление нового сигнала

Примечание. Рекомендуется указывать уникальные названия сигналов после добавления т.к. это облегчит работу с ними при моделировании поведения. Уникальность присвоенных названий не проверяется, что является ответственностью пользователя. Количество сигналов неограниченно.

1.4. Моделирование поведения объектов

Создание, расстановка и изменение объектов на трехмерной сцене объектов выполняется посредством разработки программ на языке Javascript. Данные программы в Цифровом двойнике называются Действиями.

1.4.1. Создание нового действия

Действия можно создавать для любого из объектов трехмерной сцены и их количество неограниченно. Для создания нового действия нужно выбрать объект, раскрыть подменю действий и выбрать операцию «Добавить новое действие» (см. рис. 5).

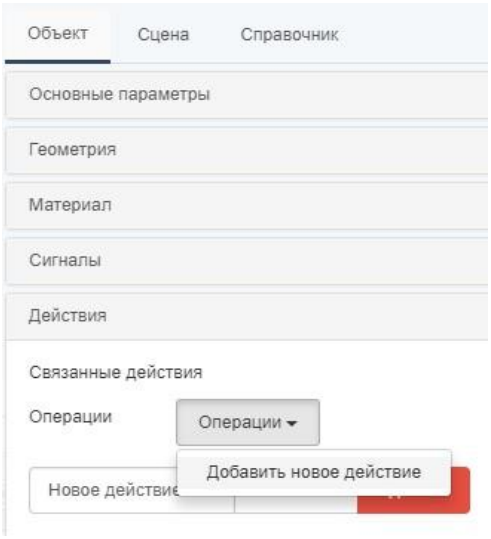

Рисунок 5 – Подменю «Действия». Добавление нового действия

В связи с тем, что действий может быть много, для удобства работы каждому действию можно дать имя, а также отфильтровать дерево сцены и показать только те объекты, у которых есть действия.

Примечание. Количество действий неограниченно. После создания нового действия можно перейти непосредственно к программированию поведения.

1.4.2. Программирование поведения

Для программирования поведения следует нажать «Изменить» напротив нового действия, после чего появится окно программирования (см. рис. 6).

Для вызова справки следует нажать (?) (позиция 1, рис. 6). Для вызова справки по текущему объекту следует нажать точку (.) (позиция 2, рис. 6).

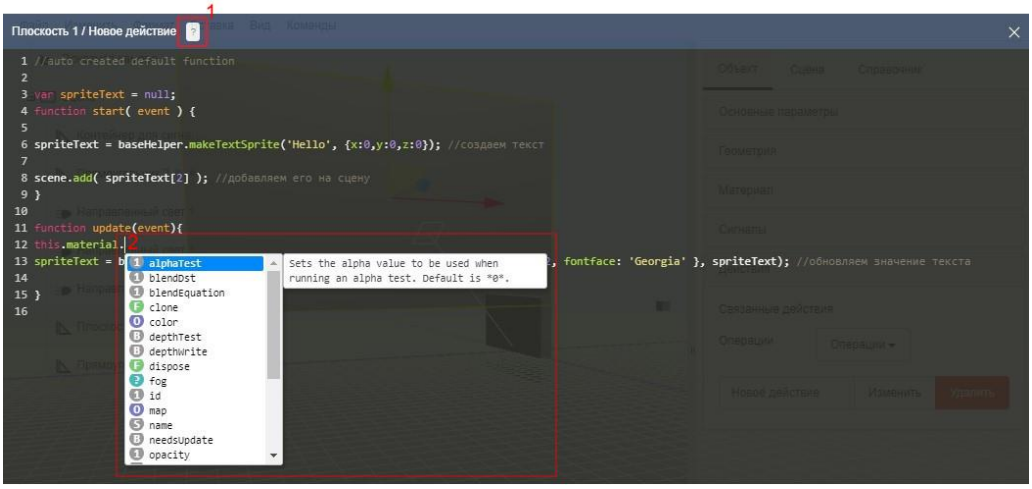

Рисунок 6 – Окно программирования. Вызов справки

2. ПРОГРАММИРОВАНИЕ В 3D СЦЕНАХ

2.1. Глобальные переменные

В любом месте кода действий доступны следующие глобальные переменные:

- **appid** – идентификатор текущего приложения;
- **oid** – идентификатор текущей 3D сцены;
- **renderer** - основной рендерер (webgl по умолчанию) 3D сцены;
- **scene** – основная сцена;
- **camera** – основная камера (камера по умолчанию);
- **this** – текущий выбранный на 3D сцене объект.

2.2. Глобальные функции

Для действий доступны следующие глобальные функции: init, start, stop, update, keydown, keyup, mousedown, mouseup, mousemove, touchstart, touchend, touchmove.

```
function init(){
 //выполняется один раз в момент инициализации выбранного объекта
}

function start(){
 //выполняется один раз после инициализации выбранного объекта
}
function stop(){
 //выполняется один раз при завершении проигрывания сцены
}
function update( event ){
 //выполняется в цикле, каждые 60 fps т.е. каждые 16 миллисекунд

 console.log(event.delta); //миллисекунд [формат double] после пред. выполнения.  //При первом выполнении event.delta < 0

 console.log(event.time); //миллисекунд [[формат double] от первого выполнения

 console.log(event.A59); //получение объекта signal json по signal ascii  // (только если имя сигнала не определено на 3D). A59 представлено как пример.
console.log(event['A59.1']); //получение объекта signal json по имени сигнала в 3D. //A59.1 представлено как пример.

 //Доступные поля объекта signal json:
 //event_ms : 1503344599551
```

```
 //event_time : "21.08.2017 22:43:19" //в текущей локали пользователя
 //name : "A59.1"
 //product_name : "Siemens S7 300 840.2"
 //product_serial : "123456789"
 //product_uuid : "E0B794D3-238E-4F7D-A5B2-1206E63B8513"
 //signal_ascii : "A59"
 //signal_name : "M06 Смена инструмента"
 //value : "55"

 console.log(event['A59.1'].signal_name);
}  function keydown( KeyboardEvent ){
//выполняется при нажатии клавиши
console.log(KeyboardEvent.key);
console.log(KeyboardEvent.keyCode);
}  function keyup( KeyboardEvent ){
//выполняется если клавиша отпущена
console.log(KeyboardEvent.key);
console.log(KeyboardEvent.keyCode);
}  function mousedown( MouseEvent ){
//выполняется при нажатии кнопку мыши
console.log(MouseEvent.layerX);
console.log(MouseEvent.layerY);
}  function mouseup( MouseEvent ){
//выполняется кнопка мыши отпущена
console.log(MouseEvent.layerX);
console.log(MouseEvent.layerY);
}  function mousemove( MouseEvent ){
//выполняется при перемещении мыши
console.log(MouseEvent.layerX);
console.log(MouseEvent.layerY);
}

function touchstart( TouchEvent ){
 console.log(TouchEvent);
                                  }

 unction touchend( TouchEvent ){ f
 console.log(TouchEvent);
                                 }

function touchmove( TouchEvent ){
 console.log(TouchEvent);
                                  }
```

2.3. Глобальные помощники и их функции

Для уменьшения размера кода у каждой трехмерной сцены есть помощники. Помощники – это объекты содержащие заранее подготовленные функции, вызов которых может быть выполнен как в асинхронном, так и в блокирующем режиме. Рекомендуется использовать асинхронные вызове там, где это возможно. На данный момент в 3D сцене доступно три помощника:

– baseUtils;
– base3DUtils;
– baseHelper.

Функции помощников описаны в таблицах 1 – 3.

Таблица 1 – Помощник «baseUtils»

Функция	Описание
`baseUtils.nls( 'text in english from baseUtils locale file [1])' );`	Возвращает локализованный текст
`baseUtils.isIE();`	Проверяет принадлежность браузера к Internet Explorer
`baseUtils.isFirefox();`	Проверяет принадлежность браузера к Firefox
`baseUtils.showError(message, details);`	Выводит сообщение об ошибке
`baseUtils.showErrorUserDefined(title, message);`	Выводит сообщение об ошибке с заголовком пользователя
`baseUtils.showWarning(title, message);`	Выводит предупреждение
`baseUtils.showSuccess(title, message);`	Выводит сообщение об успешном выполнении операции
`baseUtils.printPage();`	Открывает диалог печати страницы
`baseUtils.getCurrentDateTime( clockType, date_separator, time_separator );`	Возвращает текущую дату и время в нужном формате. clockType - DateTime | Time
`baseUtils.sendSignal( appid, productUUID, signalAscii, signalValue, serialNumber, callBack callBackError );`	Отправка управляющего сигнала на устройство
`baseUtils.saveSignal(appid, productUUID, signalAscii, signalValue, serialNumber, callBack, callBackError );`	Сохранение значения сигнала в облаке

[1] \WinnumPlatform\lib\winnum\static\resources\main\resources\themes\current\locale\[curent_lang]\winnum-uihelper.json

Таблица 2 – Помощник «base3DUtils»

Функция	Описание
`base3DUtils.getValue('lang');`	Возвращает текущий язык браузера, например, en_us
`base3DUtils.getValue('baseUrl');`	Возвращает начальный адрес, например, dev-1/Winnum/
`base3DUtils.getValue('baseImgUrl');`	Возвращает адрес корневой директории с картинками, например, /Winnum/resources/themes/current/images/app/scene/
`base3DUtils.getValue('baseCssUrl');`	Возвращает адрес размещения таблиц стилей, например, /Winnum/resources/themes/current/base/theme/
`base3DUtils.getValue('baseJsUrl');`	Возвращает адрес корневой директории скриптов, например, /Winnum/web/views/pages/app/scene/editor/
`base3DUtils.getValue('baseFontsUrl');`	Возвращает адрес корневой директории шрифтов, например, /Winnum/web/views/pages/app/scene/editor/fonts/
`base3DUtils.nls( 'text in english from baseUtils locale file` [2)] `' );`	Возвращает локализованный текст
`base3DUtils.timems();`	Возвращает текущее время в формате строки
`base3DUtils.hexToRgb( hex );`	Переводит цвет в формате hex в формат rgb. Например, из 0x000000 в #000000
`base3DUtils.rgbToHex( rgb );`	Переводит цвет в формате rgb в формат hex. Например, из #000000 в 0x000000
`base3DUtils.roundFloat( value, size );`	Округляет число до указанного знака после запятой. Например, из 0.00310000 в 0.003
`bae3DUtils.uuid();`	Возвращает уникальный единовременно созданный идентификатор
`base3DUtils.latestSignal( signalAscii, productUUID, callBackFunc );`	Возвращает значение последнего сигнала
`base3DUtils.latestSignalPool( signalAscii, poolSize, productUUID, callBackFunc );`	Возвращает указанное количество последних значений сигнала
`base3DUtils.latestTagShiftDuration( appid, tagID, productUUID, timeout, callBackFunc );`	Возвращает длительность успешного выполнения Тега за текущие сутки с разбивкой по сменам

[2)] \WinnumPlatform\lib\winnum\static\resources\main\resources\themes\current\locale\[curent_lang]\winnum-3dhelper.json

Окончание таблицы 2

Функция	Описание
`base3DUtils.latestTagDuration( appid, tagID, productUUID, timeout, callBackFunc );`	Возвращает длительность успешного выполнения Тега за текущие сутки
`base3DUtils.latestTagEntry( appid, tagID, productUUID, timeout, callBackFunc );`	Возвращает результат расчета Тега по самым последним сигналам
`base3DUtils.loadScene( oid, appid, callBackFunc );`	Загружает сцену с сервера в объект json
`base3DUtils.loadAssetFile( oid, appid, callBackFunc );`	Загружает файл актива с сервера
`base3DUtils.saveScene( oid, appid, scene, callBackFunc );`	Сохраняет сцену на сервере
`base3DUtils.sendData( theUrl, theData, callBackFunc );`	Отправляет запрос на сервер

Таблица 3 – Помощник «baseHelper»

Функция	Описание
`var line = baseHelper.makeLine( point1, point2, colorHex );`	Создание линии
`var spriteText = baseHelper.makeTextSprite( message, points, parameters, spriteText );`	Создание спрайт текста
`var middleVector3 = baseHelper.getCenterPoint( mesh );`	Получение координат центра построения объекта

2.4. Загрузка HTML страниц в 3D сцену

В связи с особенностями используемого рендера по умолчанию в текущей версии загруженная HTML страница будет всегда впереди всех остальных объектов на сцене.

```
//auto created default function
var cssScene, rendererCSS, cssObject, flag = false;

function loadHTML( planeMesh ) {

      var planeWidth = planeMesh.geometry.parameters.width;
    var planeHeight = planeMesh.geometry.parameters.height;

      cssScene = new THREE.Scene();
      cssScene.add( planeMesh );
```

```
      var element = document.createElement('iframe');
      element.src = "/Winnum/views/navigation/home/list.jsp";

     var elementWidth = 1024;    var aspectRatio =
planeHeight / planeWidth;   var elementHeight =
elementWidth * aspectRatio;

     element.style.width  = elementWidth + "px";    element.style.height
= elementHeight + "px";
    element.style.border = 'none';

      cssObject = new THREE.CSS3DObject( element );

     cssObject.position.x = planeMesh.position.x;   cssObject.position.y
= planeMesh.position.y;     cssObject.position.z = planeMesh.position.z;
      cssObject.rotation = planeMesh.rotation;

      var percentBorder = 0.00;
 cssObject.scale.x  /=  (1  +  percentBorder)  *  (elementWidth  /  planeWidth);
cssObject.scale.y /= (1 + percentBorder) * (elementWidth / planeWidth);
      cssScene.add(cssObject);

      // create a renderer for CSS
      rendererCSS = new THREE.CSS3DRenderer();
      rendererCSS.setSize( window.innerWidth, window.innerHeight );
      rendererCSS.domElement.style.position = 'absolute';
      rendererCSS.domElement.style.top = 0;
      rendererCSS.domElement.style.zIndex = 1;

      renderer.domElement.parentNode.appendChild( rendererCSS.domElement );

      flag = true;
}

  function
update(){
```

```
      if ( !flag ){
            loadHTML(this);
      }else{
            rendererCSS.render( cssScene, camera );
      }
}

function stop(){
      if (flag){
            cssScene.remove(cssObject);
            rendererCSS.domElement.parentNode.removeChild(rendererCSS.domElement);
      }
}
```

2.5. Использование видео в качестве текстур

По умолчанию поддерживается воспроизведение видео файлов в формате ogv и mp4.

```
 //auto created default function
var video;

function init( ){

      // create the video element
      video = document.createElement( 'video' );
      //video.id = 'video';
      //video.type = ' video/ogg; codecs="theora, vorbis" '; //for ogv
     //video.type = ' video/mp4; codecs="avc1.42E01E, mp4a.40.2" '; //for mp4
video.src =
"/Winnum/servlets/WinnumFileDownload?oid=winnum.org.file.WNFileDefinition%3A59&mode=y
es";
     video.load(); // must call after setting/changing source
video.play();

     var texture = new THREE.VideoTexture(video);
texture.minFilter = THREE.LinearFilter;   texture.magFilter =
THREE.LinearFilter;
      texture.format = THREE.RGBFormat;

      var parameters = { color: 0xffffff, map: texture, side: THREE.DoubleSide };
//THREE.FrontSide //THREE.BackSide
      var material = new THREE.MeshLambertMaterial( parameters );

     this.material = material;   this.material.needsUpdate
= true;
      this.material.map.needsUpdate = true;
```

```
}
function start( ){


}  function
stop(){
      video.pause();
}

function update( event ) {
                           }
//console.log(event)
;
```

2.6. Использование аудио на 3D сцене

По умолчанию поддерживается воспроизведение видео файлов в формате mp3 и ogg.

```
//auto created default function
var sound;
function init(event){

      var audioLoader = new THREE.AudioLoader();

      var listener = new THREE.AudioListener();
      sound = new THREE.PositionalAudio( new THREE.AudioListener() );
            audioLoader.load(

      '/Winnum/servlets/WinnumFileDownload?oid=winnum.org.file.WNFileDefinition%3A61&
mode=yes',
                  function( buffer ) {
                        sound.setBuffer( buffer );
                        sound.setRefDistance( 1 );
                        sound.setLoop(true);
                        sound.setVolume(0.5);
                        sound.play();
            });

}

function stop(){
      sound.stop();
}
```

2.7. Использование изображений в качестве текстур

По умолчанию поддерживаются следующие форматы файлов изображений: png, gif, bmp, jpeg и jpg.

```
//auto created default function
function init( event ) {

	var material = new THREE.MeshPhongMaterial( { map:
THREE.ImageUtils.loadTexture('/Winnum/servlets/WinnumFileDownload?oid=winnum.org.file
.WNFileDefinition%3A60&mode=yes') } );

	this.material = material;
	this.needsUpdate = true;
}
```

2.8. Динамический текст на основе Sprite

Для создания динамического текста рекомендуется использование Sprite объекта. Использование данного объекта снижает нагрузку на процессор в момент обновления текста, что позволяет изменять текст с частотой 60 fps (если это требуется).

```
var spriteText = null;
function start( event ) {

	spriteText = baseHelper.makeTextSprite('Hello', {x:0,y:0,z:0}); //creating text

	scene.add( spriteText[2] ); //добавляем его на сцену
}

function update(event){

	spriteText = baseHelper.makeTextSprite(event.time, {x:0,y:0,z:0}, { fontsize:
72, fontface: 'Georgia' }, spriteText); //update text value

}
```

Дополнительные параметры Sprite текста и параметры по умолчанию:

```
//color format - { r:0, g:0, b:0, a:1.0 }

parameters[ 'fontface' ]; // default - 'Arial'
parameters[ 'fontsize' ]; // default - 18 parameters[
'borderThickness' ]; // default - 4 parameters[
```

```
'borderColor' ]; // default – undefined parameters[
'fillColor' ]; // default - undefined
parameters[ 'textColor' ]; // default - { r:0, g:0, b:255, a:1.0 }
parameters[ 'radius' ]; // default - 6 parameters[ 'vAlign' ]; //
default - 'center' parameters[ 'hAlign' ]; // default - 'center'
```

2.9. Получение значения последнего сигнала

Пример использования данной функции представлен в двух вариантах (асинхронный – сцена не ждет получение данных и продолжает работать, блокирующий – сцена и все остальные объекты на ней ждут пока выполниться функция). Рекомендуется использование асинхронного вызова.

2.9.1. Пример асинхронного вызова

```
//base3DUtils.latestSignal
//get last signal, return valus are:
//id - signal ascii
//event - signal event datetime ( yyyy-MM-dd hh:mm:ss.SSS )
//value - signal value
//type - signal data type

//auto created default function
var delay = 0;
function update(event){
      delay += parseInt(event.delta);
      if ( delay < 5000 ) { return; } else { delay = 0; } //delay for 5 seconds

      //call async
      base3DUtils.latestSignal(
            event['A59.0'].signal_ascii, //signalAscii
            event['A59.0'].product_uuid, //productUUID
            function( data ){
                  if ( data.status !== 'success' ) { return; } //exit if not success

                  if ( data.id === undefined || data.id === '' ) { console.log('no
data'); return; } //exit if no data

                  //id - signal ascii
                  //event - signal event datetime ( yyyy-MM-dd hh:mm:ss.SSS )
                  //value - signal value
                  //type - signal data type

                  console.log( data.id + ' - ' + data.value + ' at ' + data.event );

            }
      );

}
```

2.10. Пример блокирующего вызова

```
 //base3DUtils.latestSignal
//get last signal, return valus are:
//id - signal ascii
//event - signal event datetime ( yyyy-MM-dd hh:mm:ss.SSS ) //value
- signal value
//type - signal data type
```

```
//auto created default function var
delay = 0; function update(event){
delay += parseInt(event.delta);
      if ( delay < 5000 ) { return; } else { delay = 0; } //delay for 5 seconds

      //call blocked
     data = base3DUtils.latestSignal(
event['A59.0'].signal_ascii, //signalAscii
event['A59.0'].product_uuid //productUUID
      );

      if ( data.status !== 'success' ) { return; } //exit if not success

     if ( data.id === undefined || data.id === '' ) { console.log('no data'); return; }
//exit if no data

      //id - signal ascii
      //event - signal event datetime ( yyyy-MM-dd hh:mm:ss.SSS )
      //value - signal value
      //type - signal data type

      console.log( data.id + ' - ' + data.value + ' at ' + data.event );
}
```

2.11. Получение указанного количества последних значений сигнала

Пример использования данной функции представлен в двух вариантах:

1) асинхронный – сцена не ждет получение данных и продолжает работать;

2) блокирующий – сцена и все остальные объекты на ней ждут пока выполнится функция.

Рекомендуется использование асинхронного вызова.

2.11.1. Пример асинхронного вызова

```
//base3DUtils.latestSignalPool
//get pool (array) of last signals, return valus are:
//values [value1,value2,value3,...]
//rows [
// { id, event, value, type },
// { id, event, value, type },
// ...
//]
//where:
//id - signal ascii
//event - signal event datetime ( yyyy-MM-dd hh:mm:ss.SSS )
//value - signal value
//type - signal data type

//auto created default function
var delay = 0;
function update(event){

      delay += parseInt(event.delta);
      if ( delay < 5000 ) { return; } else { delay = 0; } //delay for 5 seconds

      //call async
      base3DUtils.latestSignalPool(
            event['A59.0'].signal_ascii, //signalAscii
            5, //poolSize
            event['A59.0'].product_uuid,//productUUID
            function( data ){
                  if ( data.status !== 'success' ) { return; } //exit if not success

                  if ( data.rows === undefined || data.rows.length === undefined ||
data.rows.length < 1 ) { return; } //exit if no data

                  //values [value1,value2,value3..]
                  //rows [
                  // { id, event, value, type },
                  // { id, event, value, type },
                  // ...
                  //]
                  //where:
                  //id - signal ascii
                  //event - signal event datetime ( yyyy-MM-dd hh:mm:ss.SSS )
                  //value - signal value
```

```
                  //type - signal data type

                  for( var i = 0 ; i < data.rows.length; i++){
                        console.log( data.rows[i].id + ' - ' + data.rows[i].value + '
at ' + data.rows[i].event );
                  }

                  //OR just values
                  for( var i = 0 ; i < data.values.length; i++){
                        console.log( data.values[i] );
                  }

            }
      );

}
```

2.12. Пример блокирующего вызова

```
//base3DUtils.latestSignalPool
//get pool (array) of last signals, return valus are:
//values [value1,value2,value3,...]
//rows [
// { id, event, value, type },
// { id, event, value, type },
// ...
//]
//where:
//id - signal ascii
//event - signal event datetime ( yyyy-MM-dd hh:mm:ss.SSS )
//value - signal value
//type - signal data type

//auto created default function
var delay = 0;
function update(event){

      delay += parseInt(event.delta);
      if ( delay < 5000 ) { return; } else { delay = 0; } //delay for 5 seconds

      //call blocked
      data = base3DUtils.latestSignalPool(
            event['A59.0'].signal_ascii, //signalAscii
            5, //poolSize
            event['A59.0'].product_uuid //productUUID
      );

      if ( data.status !== 'success' ) { return; } //exit if not success
```

```
	if ( data.rows === undefined || data.rows.length === undefined ||
data.rows.length < 1 ) { return; } //exit if no data

	//values [value1,value2,value3,..]
	//rows [
	// { id, event, value, type },
	// { id, event, value, type },
	// ...
	//]
	//where:
	//id - signal ascii
	//event - signal event datetime ( yyyy-MM-dd hh:mm:ss.SSS )
	//value - signal value
	//type - signal data type

	for( var i = 0 ; i < data.rows.length; i++){
		console.log( data.rows[i].id + ' - ' + data.rows[i].value + ' at ' +
data.rows[i].event );
	}

	//OR just values
	for( var i = 0 ; i < data.values.length; i++){
		console.log( data.values[i] );
	}

}
```

2.13. Получение длительности выполнения Тега за текущие сутки с разбивкой по сменам

Пример использования данной функции представлен в двух вариантах:

1) асинхронный – сцена не ждет получение данных и продолжает работать
2) блокирующий – сцена и все остальные объекты на ней ждут пока выполнится функция.

Рекомендуется использование асинхронного вызова.

2.13.1. Пример асинхронного вызова

```
//base3DUtils.latestTagShiftDuration
//get current day tag success and total duration per workshift, return values:
//success_ms
//success_per //(in percent)
//total_ms
//rows [
// { name, success_ms, success_per, total_ms },
// { name, success_ms, success_per, total_ms },
// ...
//]

//auto created default function
var delay = 0;
function update(event){

      delay += parseInt(event.delta);
      if ( delay < 5000 ) { return; } else { delay = 0; } //delay for 5 seconds

      //call async
      base3DUtils.latestTagShiftDuration(
            appid, //appid
            'MSE_TEMPERATURE_VALUE', //tag name
            event['A59.0'].product_uuid,//productUUID
            60000, //timeout ms
            function( data ){
                  if ( data.status !== 'success' ) { return; } //exit if not success

                  if ( data.rows === undefined || data.rows.length === undefined ||
data.rows.length < 1 ) { return; } //exit if no data

                  //success_ms
                  //success_per
                  //total_ms
                  //rows [
                  // { name, success_ms, success_per, total_ms },
                  // { name, success_ms, success_per, total_ms },
                  // ...
                  //]

                  for( var i = 0 ; i < data.rows.length; i++){

                        console.log( data.rows[i].name );
```

```
                        console.log( data.rows[i].success_per );

                        //keep only 4 digits after zero ( from 0.00091 to 0.0009 )
                        console.log( base3DUtils.roundFloat(
data.rows[i].success_per, 4 ) );
                  }

            }
      );

}
```

2.14. Пример блокирующего вызова

```
//base3DUtils.latestTagShiftDuration
//get current day tag success and total duration per workshift, return values:
//success_ms
//success_per //(in percent)
//total_ms
//rows [
// { name, success_ms, success_per, total_ms },
// { name, success_ms, success_per, total_ms },
// ...
//]

//auto created default function
var delay = 0;
function update(event){

      delay += parseInt(event.delta);
      if ( delay < 5000 ) { return; } else { delay = 0; } //delay for 5 seconds

      //call blocked
      data = base3DUtils.latestTagShiftDuration(
            appid, //appid
            'MSE_TEMPERATURE_VALUE', //tag name
            event['A59.0'].product_uuid,//productUUID
            60000 //timeout ms
      );

      if ( data.status !== 'success' ) { return; } //exit if not success

      if ( data.rows === undefined || data.rows.length === undefined ||
data.rows.length < 1 ) { return; } //exit if no data

      //success_ms
      //success_per
      //total_ms
      //rows [
```

```
	// { name, success_ms, success_per, total_ms },
	// { name, success_ms, success_per, total_ms },
	// ...
	//]

	for( var i = 0 ; i < data.rows.length; i++){

		console.log( data.rows[i].name );
		console.log( data.rows[i].success_per );

		//keep only 4 digits after zero ( from 0.00091 to 0.0009 )
		console.log( base3DUtils.roundFloat( data.rows[i].success_per, 4 ) );
	}

}
```

2.15. Получение длительности выполнения Тега за текущие сутки

Пример использования данной функции представлен в двух вариантах (асинхронный – сцена не ждет получение данных и продолжает работать, блокирующий – сцена и все остальные объекты на ней ждут пока выполниться функция). Рекомендуется использование асинхронного вызова.

2.15.1. Пример асинхронного вызова

```
//base3DUtils.latestTagDuration
//get current day tag success and total duration, return values:
//success_ms
//success_per //(in percent)
//total_ms

//auto created default function
var delay = 0;
function update(event){

	delay += parseInt(event.delta);
	if ( delay < 5000 ) { return; } else { delay = 0; } //delay for 5 seconds

	//call async
	base3DUtils.latestTagDuration(
		appid, //appid
		'MSE_TEMPERATURE_VALUE', //tag name
		event['A59.0'].product_uuid,//productUUID
		60000, //timeout ms
		function( data ){
			if ( data.status !== 'success' ) { return; } //exit if not success

			//success_ms
			//success_per
			//total_ms
			console.log( data.success_per );

			//keep only 4 digits after zero ( from 0.00091 to 0.0009 )
			console.log( base3DUtils.roundFloat( data.success_per, 4 ) );
		}
	);

}
```

2.15.2. Пример блокирующего вызова

```
//base3DUtils.latestTagDuration
//get current day tag success and total duration, return values:
//success_ms
//success_per //(in percent)
//total_ms

//auto created default function
var delay = 0;
function update(event){

	delay += parseInt(event.delta);
	if ( delay < 5000 ) { return; } else { delay = 0; } //delay for 5 seconds

	//call blocked
	data = base3DUtils.latestTagDuration(
		appid, //appid
		'MSE_TEMPERATURE_VALUE', //tag name
		event['A59.0'].product_uuid,//productUUID
		60000 //timeout ms
	);

	if ( data.status !== 'success' ) { return; } //exit if not success

	//success_ms
	//success_per
	//total_ms
	console.log( data.success_per );

	//keep only 4 digits after zero ( from 0.00091 to 0.0009 )
	console.log( base3DUtils.roundFloat( data.success_per, 4 ) );

}
```

2.16. Получение результата расчета Тега по самым последним сигналам

Пример использования данной функции представлен в двух вариантах (асинхронный – сцена не ждет получение данных и продолжает работать, блокирующий – сцена и все остальные объекты на ней ждут пока выполниться функция). Рекомендуется использование асинхронного вызова.

2.16.1. Пример асинхронного вызова

```
//base3DUtils.latestTagEntry
//get latest (actual) tag value, where value can be:
//-1 - tag value out of date
//0   - tag value failed
//1   - tag value success

//auto created default function
var delay = 0;
function update(event){

      delay += parseInt(event.delta);
      if ( delay < 5000 ) { return; } else { delay = 0; } //delay for 5 seconds

      //call async
      base3DUtils.latestTagEntry(
            appid, //appid
            'MSE_TEMPERATURE_VALUE', //tag name
            event['A59.0'].product_uuid,//productUUID
            60000, //timeout ms
            function( data ){
                  if ( data.status !== 'success' ) { return; } //exit if not success

                  //value:
                  //-1 = tag value out of date
                  //0  = tag value failed
                  //1  = tag value success
                  console.log( data.value );
            }
      );

}
```

2.16.2. Пример блокирующего вызова

```
//base3DUtils.latestTagEntry
//get latest (actual) tag value, where value can be:
//-1 - tag value out of date
//0   - tag value failed
//1   - tag value success

//auto created default function
var delay = 0;
```

```
function update(event){

      delay += parseInt(event.delta);
      if ( delay < 5000 ) { return; } else { delay = 0; } //delay for 5 seconds

      //call blocked
      data = base3DUtils.latestTagEntry(
            appid, //appid
            'MSE_TEMPERATURE_VALUE', //tag name
            event['A59.0'].product_uuid,//productUUID
            60000 //timeout ms
      );

      if ( data.status !== 'success' ) { return; } //exit if not success

      //value:
      //-1 = tag value out of date
      //0  = tag value failed
      //1  = tag value success
      console.log( data.value );
}
```

2.17. Работа с объектами сцены

Ниже представлен перечень функций, которые могут быть полезны при работе с объектами сцены.

Вывод выбранного на сцене объекта со всеми его атрибутами и свойствами в консоль браузера

```
console.log( this );
```

Удаление камеры на 5 единиц

```
camera.position.x += 5;
camera.updateMatrix();
```

Если меняются координаты камеры (camera.position x,y,z) то всегда после этого нужно вызывать

```
camera.updateMatrix();
```

во всех остальных случаях этого можно не делать.

Вращение камеры по оси Z на 0.001 угол, где Vector3 это оси X, Y, Z

```
camera.rotateOnAxis(
      new THREE.Vector3(0, 0, camera.position.z),
      0.001);
```

Вращение камеры по двум осям Y + Z

```
camera.rotateOnAxis(
      new THREE.Vector3(0, camera.position.y, camera.position.z),
      0.001);
```

Перемещение камеры к текущему объекту

```
camera.lookAt(
      new THREE.Vector3(this.position.x, this.position.y, this.position.z)
      );
```

2.18. Сохранение сигнала в облаке

Ниже представлен пример асинхронного вызова функции для сохранения нового значения сигнала в облаке.

```
function saveSignal(){
baseUtils.saveSignal(
            'winnum.org.app.WNApplicationInstance:3', //appid
            '9D58FD4C-FDF2-4782-A012-A2A4F697D942', //productUUID
            'A474', //signalAscii
         'test', //signalValue       '',
//serialNumber          saveSignal_callBack,
//callBack
            saveSignal_callBackError //callBackError
      );
}
function saveSignal_callBack(data){
console.log(data);     if ( data.status
!= 'success' ){
            baseUtils.showErrorUserDefined('', data.message);
      }
}
function saveSignal_callBackError(data){
      console.log(data);
      baseUtils.showErrorUserDefined('', data.message);
}
```

2.19. Отправка управляющего сигнала на устройство

Ниже представлен пример асинхронного вызова функции для отправки управляющего сигнала на устройство и сохранение его в облаке.

```
function sendSignal(){
baseUtils.sendSignal(
            'winnum.org.app.WNApplicationInstance:3', //appid
            '9D58FD4C-FDF2-4782-A012-A2A4F697D942', //productUUID
            'A474', //signalAscii
          'test', //signalValue       '',
//serialNumber          sendSignal_callBack,
//callBack
            sendSignal_callBackError //callBackError
      );
}
function sendSignal_callBack(data){
console.log(data);      if ( data.status
!= 'success' ){
            baseUtils.showErrorUserDefined('', data.message);
      }
}
function sendSignal_callBackError(data){
      console.log(data);
      baseUtils.showErrorUserDefined('', data.message);
}
```

ИСТОРИЯ ИЗМЕНЕНИЙ

Дата изменения	Версия	Описание изменения
29.01.2020	1	Обновление до версии 4.0

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

ПО	–	программное обеспечение
CAD	–	Computer-aided Design (компьютерная поддержка проектирования)
CSS	–	Cascading Style Sheets (каскадные таблицы стилей)
HTML	–	HyperText Markup Language («язык гипертекстовой разметки»)
OBJ	–	формат файлов описания геометрии, разработанный в Wavefront Technologies для их анимационного пакета Advanced Visualizer
STL	–	формат файла, широко используемый для хранения трёхмерных моделей объектов для использования в аддитивных технологиях
VRML	–	Virtual Realit Modeling Language (стандартизированный формат файлов для демонстрации трёхмерной интерактивной векторной графики)

Лист регистрации изменений									
Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	Номер документа	Входящий номер сопроводи-тельного документа и дата	Подпись	Дата
	изме-ненных	заме-ненных	новых	аннули-рованных					