Site3D Guide

01.05.2020

All interaction with the library will be through the site3d. All of our 3D space we'll call a scene, any 3D object in it we'll call a model.

Getting started

First let's look at the library structure (content of the site3d directory):

- base – required scripts
- components – ready-made components: fillings and models
- modules – additional modules
  - fly.js – flight of the camera and models along the specified trajectory
- pictures – images
- translations – interface phrases in different languages
- include.js – connection script
- style.css – style file

By default, the library folder should be located in the site's root directory. If this is not the case, then you must specify the path in the site3d_path parameter of the connection script. It is also a place to connect ready-made fillings and additional modules.

To start working with the library, you need to include the js script "site3d/site3d.js".

Fill scripts contain the method site3dFill[Script name](s3d, options), where s3d – the scene object, and options – advanced setting.

Model scripts contain the method site3dAdd[Script name](s3d, name, options), where s3d – the scene object, name – the model name, and options – advanced setting.

Quick start

To demonstrate a 3D model, just call the site3d.widget(containerId, name, path, options) function, passing the HTML tag ID, the name and the path to the model, and additional parameters if necessary:

- autoLoad – true if the model loads automatically, otherwise it loads by clicking (by default true)
- isPreload – true if you want to show the download window (by default true)
- autoPreload – true if you want to hide the loading window after loading the model (by default true)
- preloadCompleted – preload scene completion function (you need to call the preload scene method to indicate the end of the preload scene)
- nav – list of navigation buttons (by default ['full', 'rotate', 'save'], you can pass false to disable the panel)

- pos – model position (by default [0, 0, 0])
- scale – model scale (by default 1)
- rotate – rotation angles as an array [x, y], where x is the rotation angle relative to the global x axis, and y is relative to the local y axis (by default [0, 0])
- scaleControl – model zoom control: is_enabled (enable or disable control, by default enabled), min (minimum value, by default 0.5), max (maximum value, by default 2)
- rotateControl – model rotation control settings (by default enabled): is_enabled (enable or disable control, by default enabled), min (minimum rotation angle, by default -90), max (maximum rotation angle, by default 90)
- autoRotate – true if auto-rotation is enabled (by default true)
- rotateSpeed – auto-rotation speed relative to the y axis (by default 10)
- environment – environment settings (parameters are passed from the enableEnvironment scene method, enabled by default, pass false to disable it)
- shadows – shadow settings (enabled by default, pass false to disable it): angle – angle of view (by default 90), near and far – space in front of the camera (by default 0.1 and 100)
- ambientLight – ambient lighting parameters: color (colour, by default white) and power (power, by default 1)
- directionalLight – directional lighting parameters: color (colour, by default white), power (power, by default 1), pos (position, by default [1, 1, 1]) and target (direction position, by default [0, 0, 0])
- alpha – true if transparency is enabled (by default true)
- additional parameters of the importModel method

In the HTML container's styles you must specify the width and height of the 3D scene.

Example:

```
<div id="containerDemo" style="width: 100%; height: 500px;"></div>

<script>site3d.widget(' containerDemo ', 'demo', 'models/demo', {auto_rotate: 60});</script>
```

The method returns the scene object.

Rules for setting parameters

- pos, target, rot – values that transmit values along three coordinate axes are set and returned in an array format (e.g. pos = [1, 2, 3] – position with coordinates: x = 1, y = 2, z = 3)
- rotation angles are given in degrees, time in seconds, and speed in units of movement per second
- axes – axes in string format (e.g. "xy" is for x and y axes)
- color – colour in HTML string format (for example, "#ff0000" is red)

- fill – filling the model with colours, textures, video, shaders (it is desirable to choose texture sizes as multiples of the power of two, for example: 128, 256, 512 pixels, etc.), the following options are possible:
    - name – name of previously loaded fill using the scene method add_fill(name, fill)
    - color – colour line
    - texture – path string to texture in JPG or PNG format
    - video – Identifier of the HTML tag <video>
    - shader – object {vars, pixel, vertex} for transmitting the result of shader execution (GLSL program), where vars is an array of paths to textures or an arbitrary parameter object for shaders, and pixel and vertex are javascript identifiers or code strings for pixel and vertex shaders
    - {value, count, side, color, emissive, shininess, metalness, roughness, blending, transparent, opacity}: value – color, texture, video or shader, then there are additional properties: count – by how many parts to multiply (for textures), side – which edges are filled ("inside" – only inside, "out" – only outside (by default), "both" – inside and outside), color - colour, emissive – luminosity colour (by default black), shininess – the level of glare (from 0 till 150), metalness – metal properties (from 0 till 1), roughness – harshness properties (from 0 till 1), blending – blending mode («no», «normal» (by default), «additive», «subtractive», «multiply»), transparent – true if transparency is supported (by default false), opacity – degree of transparency (from 0 to 1)
    - array from {name, fill} – only imported models: name – model section name, fill – any of the previous filling values
- vars – variables for a shader in the form of an array, whose elements are arrays [name, value] of the variable name and value

Constructor

site3d(canvas, options) – scene creation, where canvas – the identifier of the <canvas> tag, which will show our scene, and options:

- load(info) – function of scene loading process processing, which returns an object with the following properties as an argument:
    - countModel – number of models added for loading
    - countModelLoaded – number of loaded models
    - isPreloadCompleted – true if the scene preload is completed (this scene is marked with the preload method)
- alpha – true if transparency is enabled (by default true)

Scene loading

- preload() – mark the moment when the scene preload ends

Scene rendering

- render() – drawing the current state of the scene
- using one of the child classes of site3dRender to automatically draw the scene
    - site3dRenderFull – class for rendering the scene constantly
    - site3dRenderActions – class for rendering the scene only when the user interacts

site3dRender methods:

- constructor(s3d) – the constructor accepts a scene object
- render() – drawing the current state of the scene
- start() – start the automatic rendering of the scene
- stop() – stop the automatic rendering of the scene

Scene settings

- background(color) – setting the background color
- skybox(path, size) – creating a three-dimensional background, where path – path to the panorama texture file, size – the size of the volume
- effects(options) – scene effects, options:
    - brightness – brightness from 0 till 1 (by default 0.15)
    - contrast – contrast from 0 till 1 (by default 0.3)
- enableEnvironment(options) – turn on the environment, where options: path – path to a panoramic texture file in HDR format or one of the preset values: "grey" (by default), "sunset", tone – degree of texture overlay on objects from 0 to 1 (by default 0.5), is_background – true, if you want to display the texture as a scene background (by default false)
- disableEnvironment() – turn off the environment
- fog(color, near, far) – setting of fog: color – colour, near and far – parameters of fog density depending on the distance

Adding and accessing models

All models have a unique name, parameter set, and fill. Some models have a load(model) function to inform about the end of the boot process (the model itself is passed as a parameter). For such models, the beginning of interaction must be performed in this function or tracking the result of the model method isLoaded().

Let's consider methods of adding models with a description of specific parameters:

- plane(name, width, height, fill) – plane: width, height – length and height
- cube(name, width, height, depth, fill) – cube: width, height, depth – length, height and depth

- sphere(name, radius, detail, fill) – sphere: radius, detail – radius and level of detail
- hemisphere(name, radius, detail, fill) – hemisphere: radius, detail – radius and level of detail
- text(name, text, options, fill, load) – 3D text: text, options – text and additional parameters (font_path – path to the font file in json format, font_size – font size, text_align – alignment with the model position (values "left" and "center" are available))
- importModel(name, path, options) – import the model in GLB format, where path –path to the model file, and options – additional parameters:
    - load(model) – function that is called after the model has finished loading
    - progress(model, info) – function that is called during the model loading, where info contains:
        - percent – percentage of the model loading
- importGroup(name, items, options) – import a group of models in GLB format, where items – array with information about models (possible values: {name, model}, {name, path, options}, where name is the model name, model is the model object, path and options are parameters as in importModel), and options – additional parameters:
    - isAutoConnect – if true, all models in the group are connected to the first model (by default true)
    - load – function that is called after all models have finished loading

Let's consider the directory structure for imported models:

- name.glb – model file

Any model can be accessed via the model(name) method.

Working with model fill

- addFill(name, fill) – add named fill

Adding and accessing lighting sources

All lighting sources have a unique name, colour, and power.

Let's consider methods of adding light with a description of specific parameters:

- ambientLight(name, {color, power}) – general: color – colour (by default white), power – light intensity (by default 1)
- directionalLight(name, {color, power, pos , target}) – directed: pos – source position (by default [1, 1, 1]), target – direction position (by default [0, 0, 0])
- spotLight(name, {color, power, pos , target, angle, blur}) – cone: angle – cone angle (by default 45), blur – smoothness of the light spot (by default 0.5)

The source is accessed via the light(name) method.

Basic Camera Operation

- camera(options) – adjustment of basic parameters: angle – angle of view (by default 50), near and far – space in front of the camera (from and to, by default 0.1 and 100), pos – position (by default [0, 0, 1]), target – observation point (by default [0, 0, 0])
- getCameraPos() – get camera position
- cameraPos(pos) or cameraPos(x, y, z) – camera position setting
- isCameraPos(pos) – true if the camera is in pos position
- moveCamera(step) – camera movement in the direction of the point of the view along the xz plane
- moveCamera(step, pos) – camera movement in the direction of the pos point
- moveCamera(stepX, stepY, stepZ) – camera movement on three axes relative to the observation point
- getCameraRot() – get camera rotation status in [x, y, z] format
- cameraRot(rot) or cameraRot(x, y, z) – set camera rotation status
- rotateCamera(pos) – rotate the camera in the direction of the pos
- rotateCamera(angle, options) – rotate the camera around the position by angle (options: pos – position, axe – rotation axis string (by default "y"), is_look – truth, if you want to watch the look (by default true), all options are saved and you can call the method without them)
- rotateCamera(stepX, stepY, stepZ) – camera rotation on three axes

Camera Flight

fly(object, options, end) – camera's flight to a certain position or to their set (the end function signals that the flight is over).

Possible values of an object:

- Model name string
- Position

You can specify an array of any combination of these values so that the camera can make a sequential flight over several positions.

Possible options values:

- direction – the direction of camera movement ("forward" – forward (by default), "back" – backward, "none" – without turning)
- distance – at what distance from the object to stop (if "none", there is no movement, only rotation)

- duration – flight time (if speed is specified, it is ignored)
- speed – flight speed
- loop – true if you need constant repetition of the entire flight chain

Also, each object can be given unique properties: {value: object, options: {direction, distance, duration, speed, end}}, where end – the completing phase of flight camera.

Any the camera isFly() and stopFly() methods can be used to stop the flight.

Control the viewing of the entire scene

- enableControls(options) – enable
- disableControls() – switch off

Let's take a look at the viewing options:

- rotate: {events, axes, speed} – rotation
  - events – array of control rows (possible values: "mouse_left" – left mouse button, "mouse_right" – right mouse button (by default ["mouse_left", "mouse_right"]).
  - axes – axes of rotation (x, y or x and y simultaneously (by default "xy"))
  - speed – speed (by default 1)

Methods for models

Downloading

- isLoaded() – true if the model is loaded
- hide() – hide the model
- show() – show the model

Basic methods

- size() – get information about the size of the model, returns an object with the following properties:
  - x – length along the x-axis
  - y – length along the y-axis
  - z – length along the z-axis
  - pos – the center of the model
- getPos() – get position
- pos(pos) or pos(x, y, z) – position setting
- move(pos) or move(stepX, stepY, stepZ) – movement on three axes
- moveLocal(pos) or moveLocal(stepX, stepY, stepZ) – movement along local axes of the model

- getScale() – get scale
- setScale(scale) or setScale(x, y, z) or scale(step) – scale setting (you can set one value to set the same value in three axes)
- scale(scale) or scale(stepX, stepY, stepZ) – scaling on three axes (you can set one value for uniform scaling on three axes)
- getRot() – getting the rotation status
- rot(rot) or rot(x, y, z) – setting of rotation status
- rotate(stepX, stepY, stepZ, options) – rotate along three axes that are local to the model or global (options – advanced setting (optional parameter): is_local – true if the axes are local (by default true), duration – animation time).
- isRotate() – returns true if a rotation is performed
- pauseRotate() – pause the rotation
- playRotate() – continue the rotation
- stopRotate() – stop the rotation
- fill(fill) – edit model filling
- getColor(name) – get the model color (if the name is specified, the color of the model's part)
- The model is flown using the fly, isFly, and stopFly methods, as well as the corresponding camera flight methods

Viewing management

- enableControls(options) – enable
- disableControls() – switch off

Let's take a look at the viewing options:

- scale: {min, max, speed} – zooming with the scroll wheel
  - min – minimum scale (by default 0.5)
  - max – maximum scale (by default 2)
  - speed – скорость (by default 1)
- rotate: {events, axes, speed} – rotation
  - events – array of control rows (possible values: "mouse_left" – left mouse button, "mouse_right" – right mouse button (by default ["mouse_left", "mouse_right"]).
  - axes – axes of rotation (x, y or x and y simultaneously (by default "xy"))
  - speed – speed (by default 1)

Camera or model connection management

- connect(options) – enable
- disconnect() – switch off

Let's take a look at the options:

- modelName – model name (if empty, the connection to the camera)
- moveLocal - offset within the local axes of the model
- rotateLocal – rotation relative to local axes

Link management

- link(click, options) – enable, where click – click processing function (returns as arguments the model and the standard object of the click result (event)) or URL address
- unlink() – switch off

Let's look at the reference parameters (options):

- boundHover – true if the hover extends to the area in the form of a parallelepiped (by default false)
- newWindow – open link in a new window (actual, if URL address is passed as click parameter)
- hover – mouse hover event
- out – mouse out event
- move – mouse move event
- hint – a string of the identifier of the HTML element with a hint, which appears when you hover the mouse and disappears when you hover it.

Working with shaders

- setVars(vars) – set variables for the shader

Label management

- setLabels(labels) – add labels, where labels – an array of objects with the following properties: name (label's name), content (HTML-element identifier), pos (position relative to the model (by default [0, 0, 0])) and is_show (true if the placemark is visible (by default true))
- setLabel(name, options) – set new properties for an already added label or add a new label, where name – label's name, and options – properties similar for method set_labels objects
- deleteLabel(name) – delete a label by name
- showLabels() – show all labels
- hideLabels() – hide all labels

Methods for lighting sources

Basic methods

- setColor(color) – colour setting
- setPower(power) – set lighting intensity (by default 1)
- getPos() – get position
- pos(x, y, z), pos(pos) – position setting
- move(stepX, stepY, stepZ) – movement on three axes
- getTarget() – getting position of lighting direction
- target(x, y, z), target(pos) – setting of lighting direction position
- moveTarget(stepX, stepY, stepZ) – movement of lighting direction along three axes

Working with shadows

To display shadows you must configure the light sources that will generate them and specify the objects that will cast and receive shadows.

Configuring light sources:

- enableShadows(options) – turn on, where options:
  - size – shadow texture size (from 1 till 3, by default 2)
  - parameters for directional light: angle – angle of view (by default 90), near and far – space in front of the camera (by default 0.1 and 100)
- disableShadows() – turn off

Configuring models:

- enableShadows(actions) – turn on, where actions – mode strings (by default «cast_receive»):
  - «cast» – cast shadows
  - «receive» – receive shadows
  - «cast_receive» – cast and receive shadows
- disableShadows() – turn off

Working with shaders

To include shaders in your project, just apply the appropriate fill {vars, pixel, vertex} to the model (this is mentioned in the rules for setting parameters at the beginning of this guide). If you do not specify uniform vars variables, the following values will be passed automatically:

- vec3 iResolution – the length and width of the scene (the third parameter vec3 will be set to 1)

- float iTime – time in milliseconds since the scene have started loading
- vec4 iMouse – the mouse coordinates (x and y – click, z and w – movement)
- sampler2D iChannel0 and iChannel1 – textures

Working with HTML content

In Site3D, you can assign your content to models for display in a pop–up window.

For this purpose, the identifier of the container of the dialog window and the content itself should be passed through the contentParams(container, content) scene method (by default, container names are taken from the name canvas from the scene constructor with the ends "_window" and "_window_content" respectively).

Further, it is necessary to assign models the identifier of their content through the setContent(content) method, and to output, the window uses the showContent() method.