NGPS Core Documentation

1. Structure

1. Folder structure

..

scripts [ contains the core of the engine ]

plugins [ contains built-in and purchased apps ]

style [ contains global styling mechanisms, Bootstrap ]

statistics [ contains statistic information collected from the user in order to improve the experience ]

index.html [ application entry point ]

1.1 Core code structure

1.1.0 Bare core

TweenMax.min.js [ awesome 2D transforms library ]

require.js [ used for dynamically loading js files]

drivers.js [generic utility and platform specific functions]

container.js [ container code, building block upon which the system is built]

interaction.js [ extension module that adds interaction possibility to a container ]

camera.js [ extension module that turns a container into a camera]

app.js [ extension module that allows a container to host an app ]

1.1.1 Factory Wrapper

factory.js [ does initial setup, links theme with container creation, is an adapter for container & camera creation ]

init [ initial setup function, creates the factory.root object which is the root of the presetation ]

setup [ function defined by external sript for custom initial setup ]

AMS [ Anti Monotony Script dictates how every container will be styled ]

. constructors [ folder containing all the custom setup scripts ]

. themes [ folder containing all the AMS’ ]

. descriptors

containers.js [ file containing all the container styles owned by the user ]

links.js [ file containing all the link styles owned by the user ]

1.1.2 Events

gem.js [ is a generic event manager that handles event registering, dispatching, unregistering and monitoring ]

1.1.3 Saving & Loading a presentation

save.js [ handles saving the presentation in a HTML format ]

load.js [ loads the presentation from a HTML format ]

1.1.4 Applications

Every applications is identified by a unique name, for example “text\_editor” and is contained within a folder with the same name. Applications are found in the “plugins” folder:

plugins

text\_editor

main.js [ this is the application’s entry point, it is responsible for including dependencies and setting up the app]

1.1.5 CLI

cli.js [ is a command line interface that allows NGPS developers to easily test and manipulate the system ]

1. Meet the APIs

\* Method parameters enclosed in < > are optional

2.1.0 container : building block of NGPS system ( everything lives inside a container )

2.1.1 Type

A container can be raw, a camera, a link or an app. In order to define what it is, a container has the following flags:

* isCamera
* isLink
* isApp

//TODO: check if the flag is set before the addChild event is fired

Members

|  |  |
| --- | --- |
| UID | unique numeric identifier for the container |
| DOMreference | the pointer to the actual HTML element. This one contains all the properties required to manipulate HTML properties like: innerHTML, style, etc. Also if you want to use jQuery with a NGPS container you simply need to use the DOMreference property:  $(a.DOMreference).on(“click”,function(){}); |
| parent | the NGPS parent of the current container |
| properties | the descriptor that was passed to the load function when the container was created |
| child |  |
| children | a list of containers belonging to this one. The children containers are always masked by the parent container. |
| outgoing | outgoing links of the current container |
| incoming | incoming links of the current container |
| events | a list of events that this container is listening for ( used to optimize event firing: in stead of firing a GEM event every time it happens, the event is only fired if there is at least one active listener for that particular event ) |
| allowMove | a flag enabling / disabling the movement of the container: this only applies to the interact.js functions not to the container.js functions for moving |
| allowTrigger |  |

2.1.2 Permissions

Permissions control how containers are handled by the system facilities. The permissions of the current container are inherited by all its children.

NGPS permissions for containers:

* save – permission to save the container in the final presentation ( some applications are only meant to be visible when editing the presentation therefore the containers they create must not be saved )
* connect – permission to connect this container with another one ( .link() )

Callbacks

|  |  |  |
| --- | --- | --- |
| Name | Parameters received | Override effect to system |
| onMouseDown | container, mouse\_event\_object | none |
| onMouseUp | container, mouse\_event\_object | none |
| onMoved | delta x, delta y, container | The system no longer moves the container, the callback function must deal with that. For example if you want to make the container unmovable by mouse/touch interaction override this callback with an empty function  c.onMoved = function(){}; |
| onTrigger | container, mouse\_event\_object | none |
| onRotated | delta angle | Same as for onMoved |
| onZoomed | delta scale | Same as for onMoved |

* delta = difference between current and previous value
* Callbacks are much faster than event listeners but have the disadvantage of not being able to call multiple functions when the event occurs, therefore it is preferable to use GEM events when building apps

Methods

|  |  |
| --- | --- |
| load(<parent>) | creates the HTML object representing the container, without calling load the container is not functional and not visible |
| extend( extensions ) | extends a container’s functionality – is used to turn a container into Camera / App and to make container interactive. It adds the functions and members of the extensions object to the current container. |
| strip ( extensions ) |  |
| addChild( properties ) | adds a child container to the current one |
| removeChild( UID ) | removes a child container by UID |
| discard( ) | deletes the container ( cleans everything up – links, events, children ). The call is bubbled through to all children. |
| changeParent( parent ) | changes the parent of the container |
| addPrimitive( descriptor ) | adds a primitive HTML object inside the current container. Used to load images, videos, text, websites inside the current container.  Example of primitive objects ( any HTML tag ):  <img></img> or <iframe></iframe> |
| removePrimitive( ) | removes a primitive HTML object |
| show( ) |  |
| hide( ) |  |
| getPos( cx, cy, <refX>, <refY> ) | Gets the position of the container. refX and refY are where the point represented by the position will be on the container.  example:  .getPos(0,0) – upper left corner position  .getPos(0.5,0.5) – center of the container  .getPos(1,1) – lower right corner |
| getWidth() |  |
| getHeight() |  |
| getPureWidth( ) | - does not work |
| getPureHeight( ) | - does not work |
| local2global(<refX>,<refY>,<stopAt>) | gets the position of the current container on the root object or at the container specified by stopAt. |
| global2local(x,y); | converts root object coordinates to local container coordinates |
| setWidth( width ) |  |
| setHeight( height ) |  |
| setAngle( angle, <ox>, <oy> ) | Set the angle of the container. The center of rotation is specified by refX and refY in percentages  0,0 – upper left  0.5,0.5 – center  1,1 – bottom right |
| putAt( x, y, <refX>, <refY> ) |  |
| move( dx, dy, <noevent> ) |  |
| scale( amount, <ox>, <oy>) |  |
| enlarge( amount ) |  |
| rotate( delta\_angle, <ox>, <oy> ) |  |
| getAncestors( node ) |  |
| greatestCommonParent( target ) | Finds the greatest (deepest) parent that the current container and the target one have in common |
| link( target, descriptor ) | Creates a outgoing link between the current container and the target. |
| unlink( target ) | Deletes an outgoing link between the current container and the target |
| unlinkAll() | Deletes all links that are connected to the current container |
| changeLinkTarget( oldTarget, newTarget) | change the target of an outgoing link |
| maintainLink( target ) | this function updates the link position, size and angle in case any of the two connected containers changes in any way ( width, height, position ) |
| maintainLinks() | maintains all the links of a container |
| addEventListener(event, handler, <context>) |  |
| removeEventListener( event, handler, <context>) |  |
| loadApp(appName) | loads an application onto the current container |

Events

|  |  |
| --- | --- |
| loadContainer |  |
| addChild |  |
| removeChild |  |
| discardContainer |  |
| hideContainer |  |
| showContainer |  |
| changeParent |  |
| changeWidth |  |
| changeHeight |  |
| changeAngle |  |
| changePosition |  |
| link |  |
| unlink |  |
| linkChange |  |
| appLoaded |  |

* + 1. Interact – extension module that makes a container interactive

container.interactive(true); enables interaction on the container

container.interactive(false); disables interaction on the container

Members

|  |  |
| --- | --- |
| propagation | determines what happens if an event fires on the container  propagation = 0 – default, event is not propagated  = 1 – event is ignored  = 2 – event is propagated to the parent container |
| triggerCount | the number of times a the container was triggered (clicked ) |
| origin | the object on which the move operation has started |

Methods

|  |  |
| --- | --- |
| onMouseDown( e, <context> ) | event e fired on container - context |
| onMouseMove( e , <context> ) |  |
| onMouseUp( e, <context> ) |  |
| onMouseOut( e, <context> ) |  |
| touchstart( e, <context> ) |  |
| touchmoved( e, <context> ) |  |
| touchend( e, <context> ) |  |
| interactive( isInteractive ) | activates/deactivates interaction on container |

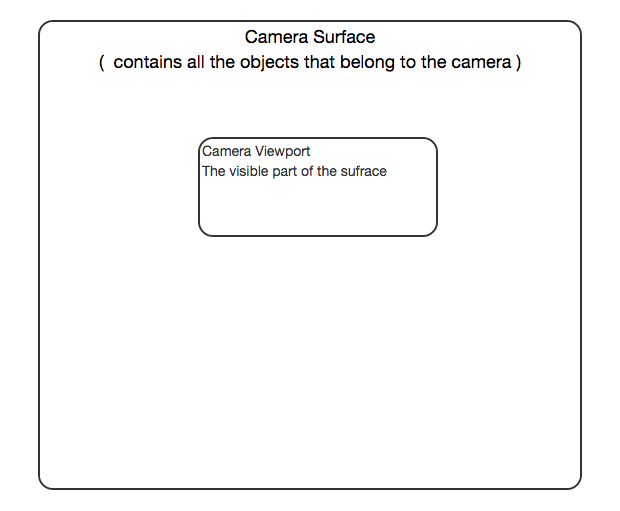
Events

|  |  |
| --- | --- |
| mouseDown |  |
| mouseMove |  |
| mouseUp |  |
| mouseOut |  |

* + 1. Camera – extension module that enables a container to be a camera

When the container is converted into a camera (becomes a viewport) it automatically instantiates a child called display (also called surface). The display container will host all the elements belonging to the camera’s perspective.

addChild() will, by default, add the child containers directly to the display container



Getters and setters must be used carefully as a camera is comprised of 2 containers: the parent and the display.

For example calling

container.move(10,0) will move the viewport but will not change the perspective of the camera while

container.display.move(10,0) will move the surface, therefore changing the perspective of the camera

There are two types of functions that operate on the camera perspective:

* **Fluid** – always start with ‘c’ – these functions operate on a time scale and create an animation effect (they do not complete immediately)
* **Granular** – always start with ‘c\_’ – these functions complete the operation immediately and have no intermediate steps
* Getters or other functions that simply retrieve/set information but do not change the camera perspective may start with ‘c’ even though they are Granular. ( because of legacy reasons … ) This should be changed on next major refactoring

//TODO: talk about boundaries and relations

Members

|  |  |
| --- | --- |
| display | the camera perspective container |
| czoomLevel | floating point number representing how much zoom has been applied to the camera perspective ( example: 2.5 ) |
| cangle | the angle of the camera perspective ( in radians, or degrees [ TODO: check angle measuring units ]) |
| callow | flag that tells the camera if it’s allowed to change it’s perspective.  if set to false, functions that change the perspective of the camera do not run. |
| cops | list of independently running operations |
| cinterval | the period of the camera operation ticks |
| allowInertia | tells the camera if inertia is allowed |
| crelations | holds the camera relationships ( a camera can have relationships on certain perspective properties with other cameras, example: when a camera moves to the right another camera moves to the right as well ) |
| boundaries | holds the camera’s boundary rules. Example: you want limit the camera scrolling so that user won’t be able to infinitely scroll to the right or the left. |

Methods

|  |  |
| --- | --- |
| addChild( descriptor, <addToFrame> ) | override of the container’s addChild function  if addToFrame = true then the child is added to the parent in stead of the display child of the camera. |
| cstart | starts and initializes the camera |
| ccancel(<what>) | cancels all camera’s operations |
| cstop | stops the camera |
| getContentPositioning( ) | get the position and size of the camera.display |
| cmove (dx,dy) | move the camera by dx and dy |
| czoom ( amount, <ox>, <oy> ) | zoom the camera by amount ( for example if the camera’s perspective is 2 times bigger than the original, by calling czoom(0.5) the perspective will return to normal )  Optional paremeters ox and oy specify center point of the zoom ( default ox and oy = 0.5) |
| crotate (amount, <ox>, <oy> ) | rotates the camera by amount.  ox and oy specify the center point of the rotation ( by default ox and oy = 0.5 ) |
| cpan ( panx, pany) | not implemented yet |
| cfocusOn ( target, <options> ) | focuses camera on the target object ( moves, zooms and rotates the camera so that the camera object is as big as possible on the screen )  options: define how the camera will focus on the target ( for example one may not want the camera to zoom the object in, only bring that container to the center ) |
| addRelated( cam, descriptor) | adds a relationship between this camera and the one specified in the first parameter.  descriptor contains the exact relationships between the two camera’s properties |
| removeRelated (camera) | removes the relationship between this camera and the provided camera with the first argument |
| setBoundaries (boundaries) | adds boundary rules to the camera |
| unsetBoundaries (boundaries) | removes boundary rules from the camera |
| tween(data) | not implemented yet |

* + 1. App - Extension module to enable a container to host an app

2.1.3.1 AppMgr – global object containing general app manager

Members

|  |  |
| --- | --- |
| status | Status of the current app that is currently running in the foreground |
| running\_app\_parent | The parent container of the current foreground app |
| loadedApps | Contains the app sources loaded ( contains the classes of the apps loaded ) |
| workers | hosts all the worker functions and properties |
| maxAppWorkers | number of maximum workers that an app can have |

2.1.3.2 App related extensions applied to the container class

loadAppCode( name, class ); is a global function that is used to load the apps

every app’s main.js must declare a class ( object ) and load it with this function.

Members

|  |  |
| --- | --- |
| isApp | flag signaling that the container is hosting an app |
| app | an instance of the app’s class |
| aworkers | integer used to index the app’s workers |
| cover | a child of the host container used to capture interaction events.  When the trigger event fires on this container it hides itself and shows the exit container ( this way all the interaction events go to the app ) |
| exit | a child of the host container which shows the cover container and hides itself when clicked. |

Methods

|  |  |
| --- | --- |
| ainit | initializes the app inside the parent container |
| adestroy | completely removes the app from the host container ( also calls the app’s shutdown method ) |
| ashow | makes this app the current foreground app and  calls the app’s show function ( this translates to telling the app that it is now running in the foreground, some apps may have interfaces that need to be show now ) |
| ahide | the current app is no longer running in the foreground  calls the app’s hide function ( this translates to telling the app that it has lost focus and is no longer running in the foreground, some apps that have interfaces may need to hide their interfaces now ) |
| arun | calls the app’s run function |
| asuspend | calls the app’s suspend function |
| requestWorker( worker, interval ) | periodically runs the worker function at the given interval   * returns:   an integer ( Worker ID ) that the app can use to stop the worker  -1 if the a worker could not be allocated |
| stopWorker( id ) | stops the worker that has the provided id |

* 1. Factory - presentation layout manager ( container, camera creation & customization according to themes )

Members

|  |  |
| --- | --- |
| initialized | 0 if the factory is not initialized  1 if the factory is initialised |
| defaultContainer | a default container descriptor |
| settings | an object containing factory settings |
| root | the root container of the presentation |
| AMS | the Anti Monotony Script. Loaded from /scripts/themes/ |

Methods

|  |  |
| --- | --- |
| init( ) | initializes the factory, configures the factory settings and runs the custom setup function is it is defined. |
| setup( ) | custom function defined by any script in /scripts/constructors/  used to create custom setups ( presentation themes ) |
| newContainer( possize, tag, <parent>) | creates a new container.  possize: specifies the position and size of the container ( format {x:0,y:0,width:100,height:100} )  tag: the name of a descriptor to used in order to style the container  parent: the parent container where the new container will be attached |
| createContainer( descriptor, <parent>) | creates a new container entirely described by the descriptor passed  parent: specifies the container parent where the new container will be attached |
| newCamera( possize, tag, <parent>, <interval>) | creates a new camera:  possize: same as for container  tag: same as for container  parent: same as for container  interval: the period between camera operation ticks |
| newGlobalApp( appName ) | loads a new global application, it adds the app containers as children of the factory.root |

2.2.1 Descriptors

2.2.2 Anti Monotony Scripts

* 1. General Events Manager (GEM)

Members

|  |  |
| --- | --- |
| events | Stores all event handlers and their required information ( object, run\_context ) |
| debug | flag that determines if GEM prints events on the CLI.  Default: false |

Methods

|  |  |
| --- | --- |
| fireEvent( data ) | fires a GEM event |
| addEventListener( event, object, handler, <run\_context> ) | adds an GEM event listener.  Parameters:  event: name of the event [string]  object: the object on which the event occurs [object]  handler: function that handles the event  run\_context: object that needs to be accessible from inside the handler[object]  \* if run context is present, handler needs to be a string naming a member function of run\_context |
| removeEventListener( event, object, handler) | removes event listener |
| cancelAll( ) | cancels all event listeners |
| list( verbose ) | lists all the GEM event listeners |

2.4 Apps

An apps main.js file contains all the necessary code to include dependencies, setup the app and comply with the NGPS environment. This file has a set structure so that apps can be easily loaded and managed by NGPS, therefore every app is a class ( object ) with the following structure:

var myApp = function( data )

{

this.config = { <add configuration properties here> };

this.parent = data['parent'];

this.init = function(){ } //called only one when bound to container

this.run = function(){ } //called whenever the container is triggered

this.suspend = function(){ } //called whenever the container looses focus ( or gets out of view )

this.shutdown = function(){ }//called only when app is unloaded from container

this.show = function(){ } //shows app

this.hide = function(){ } //hides app

}

//this function must be called in order to load the app into NGPS

loadAppCode("myAppName",myApp);

NGPS apps have complete control over events, therefore if an app is loaded in a container then it becomes unmovable. To solve this NGPS has a built in system that gives or takes the app control over events. By clicking once on a container with an app it will cause the app to run ( the run function is called ) and therefore all events will be diverted to the app. A round tick will be displayed in the upper left corner which if clicked will suspend the app ( the suspend function is called ) and take back event control from the app making the div movable again.

* 1. Command Line Interface (CLI)

\* Used as a developer tool

Members

|  |  |
| --- | --- |
| node | the NGPS container that the console is focused on ( in order to operate on containers the CLI needs to focus on a container and then it can call the container’s member functions or reference it in other functions ) |
| UI | CLI UI div reference |
| UI.UIout | CLI output div reference |
| UI.UIin | CLI input field reference |

Methods

|  |  |
| --- | --- |
| showPrompt( <message> ) | shows the NGPS CLI prompt. Optional parameter message is a string and is appended to the prompt string. |
| show( ) | shows the CLI |
| hide( ) | hides the CLI |
| keysHandler( e ) | handles keyboard events and executs command when enter key is pressed |
| onExec( e ) | executes command in UI.UIin and updates the output |
| fetchParameters( parameters ) | evaluates parameters ( the raw string command must be split into separate strings separated by white spaces and passed to this function) |
| execute( command\_string ) | executes command passed as a string |
| shobjects( object ) | shows the members of an object |
| help( ) | displays information about the CLI on the NGPS CLI prompt |
| shtree( ) | shows the NGPS container tree |
| sh( ) | shows information about the current container the CLI is focusd on |
| cn( id ) | changes the CLI node to the container with the passed id. ( Focuses the CLI on a new node ) |
| rst( ) | resets the CLI node to the factory.root |
| ldtest( test\_name ) | loads a test script from /scripts/tests/ |
| require( script ) | includes a js |
| debugCondig( ) | configures the NGPS engine for debugging and testing  (it initialized the factory, loads a FPS meter and loads the benchmark test suite ) |