OpenSourcePS2

This open-source package contains several components and demos relating to the viewer for Technology Preview synths on the Photosynth site: http://photosynth.net/preview.

**Note**: the viewer used on Photosynth.net is not using this version of the source code and not using this built system (this version supersedes the one used on Photosynth but is also missing some feature: css sprite generation …)

# Build folder

This folder is originally empty but will contains minified css and js files once you’ve run the build process with grunt.

**Note:** CSS sprite generation has not been integrated in the build process yet.

# CSS folder

This folder contains CSS files and images for the PacketPlayer and the AnnotationEditor.

# Demos folder

This folder contains all the demos.

# Docs folder

You are reading the main document from this folder ☺

# Src folder

* Core
* API
* PacketPlayer
* AnnotationViewer
* AnnotationEditor
* SDK
* Progress
* Exif
* ThirdParty

## Core

* Core.js: PS namespace definition + PS.extend() and PS.isWebGLEnabled()
* WebGLMemoryUsage.js: Singleton used to keep track of the memory used (textures+buffers).

### Touch

* SingleTouchInputHandler.js: class used to handle single touch event across all browsers
* MultiTouchGestureHandler.js: class used to handle multi-touch gesture across all browsers (handle translation and scale).

### Utils

* Utils.js: ImageDownloader, tryParse, generateRangeArray(),getUrlParams()…

#### Async

* Async.js: PS.Utils.Async namespace + fireOnce
* Parallel.js: legacy parallel calls function (use Queue instead!)
* Queue.js: lightweight Queueing system inspired by async.js
* PriorityQueue.js: same as Queue but each item as a priority

#### Request

* Request.js: mandatory lightweight xhr class.

#### Tween

* Legacy.js: adhoc tween object with hard-coded Exponential.easeOut transition (do NOT use it)
* Tween.js: lightweight tween class (transitions are copied from MooTools)

## API

* GeoWikipedia.js: find closest Wikipedia article to a lat,lng using api.wikilocation.org
* Photosynth.js: basic (without error checking) implementation of a client
* PhotosynthRead.js: helper functions to make **read** calls to the Photosynth REST apis.
* PhotosynthWrite.js: helper functions to make **write** calls to the Photosynth REST apis.
* SimpleAnnotationProxy.js: helper class to fetch annotations from Photosynth.net in read only.
* SimpleAnnotationStorage.js: helper class to connect to the experimental AnnotationStorage
* SimpleStaticStorage.js : helper class to connect to a read-only json based annotation storage
* SimpleSynthLinker.js : helper class to connect to the experimental SynthLinker

## PacketPlayer

Please read README\_ PacketPlayer.docx if you want to get more information about the PacketPlayer.

## SDK

The SDK is what you should be using for your development. It’s basically a thin wrapper on top of the internal classes (PacketPlayer, AnnotationViewer, AnnotationEditor). The internal classes are using callbacks and have lots of options (more than you should need). The SDK is white-listing some of these options and using events instead of callbacks. So you can use addEventListener() and removeEventListener(). Thus making component working together is easier and require less code and is less error-prone.

* SDK.js: Photosynth namespace (SDK is using Photosynth instead of PS as namespace)
* EventDispatcher.js: base class to handle conversion from callbacks to events
* PacketPlayer.js: SDK version of the PacketPlayer
* PacketPlayerEventDispatcher.js: EventDispatcher used by SDK version of the PacketPlayer
* AnnotationViewer.js: SDK version of the AnnotationViewer
* AnnotationViewerEventDispatcher.js: EventDispatcher used by SDK version of the AnnotationViewer
* AnnotationEditor.js: SDK version of the AnnotationEditor
* AnnotationEditorEventDispatcher.js: EventDispatcher used by SDK version of the AnnotationEditor

## Progress

This tiny component is used by the PacketPlayer to display the loading state of the HD frames of the synth. It’s disabled by default and the user need to enable UI.progressBar in the ‘42’ hidden menu to enabled it.

* Progress.js: tiny progress bar displayed on top or bottom of the screen

## Exif

This component is used by the upload page of Photosynth: it is a javascript jpeg Exif parsing library. It can extracts a thumbnail and other useful information: capture date, gps, focal length…

* Exif.js: javascript Exif parsing library

# Test folder

This folder contains jasmine spec for unit test and some javascript files with manual test (it’s rather empty at this time and is mostly meant to be a placeholder).

# Utils folder

This folder contains several tools provided as basic (=one day hack) implementation but should not be used for production:

* AnnotationStorage: This tool is used to create a persistent storage for annotations. It’s using node.js and mongodb. It’s a webservice that you can call to create/update/delete annotations. There is also a tool to import existing sets of annotations from a json dump.

You can check: README\_ AnnotationStorage.docx.

* SynthLinker: This tool is used to compute the transform between 2 synths used for transition between 2 connected synths.

You can check: README\_ SynthLinker.docx.