http://3dthis.com/developer/papiv30.pdf
2017-12-19

PHOTOANIM JAVASCRIPT API V3.0

1 INTRODUCTION

The PhotoAnim API is a high level API, requiring very little knowledge from HTML, JavaScript and 3D techniques. Its main purpose is to allow self-hosting and fine control of animations developed with PhotoAnim for Windows or with 3dthis.com online apps.
A typical roadmap is as follows:
- Create a model template using PhotoAnim for Windows or a 3dthis.com online app. Note that PhotoAnim for Windows can import models in *.OBJ and *.3DS formats.
- Publish the model on 3dthis.com (public, private or unlisted).
- Follow this document to download and animate your model, add fine controls and host the resulting animation on your website.

NOTE: If you are familiar with 3D vertices, textures, normals, trajectories, etc… then you may also create animations from scratch, see §7

2 CREATION

The PhotoAnim API JavaScript code can be found under https://3dthis.com/developer/photoanim.js

Your animation code can be found under https://3dthis.com/download.htm or you may use this test animation https://3dthis.com/getanim.php?h=NTc0OTAx. The default file name is paJSON.js

Between <head> and </head> include:

```
<script src="photoanim.js"></script>
<script src="paJSON.js"></script>
```

In the <body> define the canvas, where the animation should be played:

```
<canvas id="mycanvas" width="640" height="480"></canvas>
```

Finally, include the following Javascript code:

```
<script>
var pagl = new PhotoAnim("mycanvas", paJSON);
renderFrame();
function renderFrame(){
      pagl.renderFrame();
      window.requestAnimationFrame(renderFrame);
}
</script>
```

NOTES
1. You can have only one animation per page. If you want several animations, then you should play them inside an <iframe>
2. Having several animations running at the same time may slow down the rendering device.
3. You may also define animations from scratch, see §7.

3 ANIMATOR

An animator defines the timely behavior of a given parameter. Most animation parameters like camera position, light, objects use animators. All animators have default values, so only a few property changes are necessary to create new effects.

```
animator = {startvalue:float, value:float, min:float, max:float, delay:float,
speed:float, random:float, alternate:bool, loop:bool};
```

Most animator values range from -1 to +1, except angle animators which use degrees.

Animator Property	Type	Default Value	Description
startvalue	float	0	Start value of the animator when animation is started
value	float	-	Current animator value. Changing this value has immediate effect on next frame redraw.
min	float	-bigvalue	Minimum animator value
max	float	+bigvalue	Maximum animator value
delay	float	0	Animator start delay in seconds
speed	float	0	Animator speed in units/second
random	float	0	Random amount added to speed to create random effects
alternate	boolean	true	if true, run animator in opposite time direction when max is reached.
loop	boolean	true	if true, run animator from min to max (to min if alternate) in a loop
run	boolean	true	if loop is false, run will be set to false when min/max value is reached. Setting it to true will restart animator.

Example
Define an angle animator which starts at 10° and runs a continuous circle at 45°/sec

```
myangle = {startvalue:10, min:-180, max:180, speed:45, alternate:false,
loop:true};
```

NOTE: Animators have additional properties, like ability to follow trajectories, sync to variables… see §6.3.

4 PROPERTIES

Only most used properties are listed here. For a full properties list refer to §7.5

4.1 MAIN PROPERTIES

Property	Type	Description
pagl.error	read	false if no error, or error message (WebGL not supported)
pagl.pause	Read/write	true to pause animation, false to run it
pagl.oldtime	write	set to -1 to restart animation
pagl.refresh	write	true to request frame rendering
pagl.canvas	read	a reference to the canvas object

Examples

Use start/stop buttons to run animation:

```
<input type = ”button” value = ”Start” onclick = ”pagl.pause = false”>
<input type = “button” value = “Stop” onclick = “pagl.pause = true”>
```

Change camera angle:

```
pagl.global.camera.yrot.value = 45;
pagl.refresh = true;
```

4.2 TRIGGERING CONSTANTS

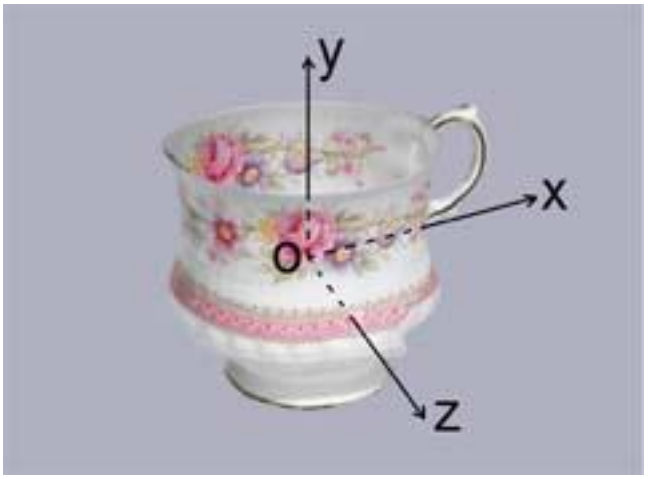
Following global constants are used for triggering:

trigger = **NONE** | **ONLOAD** | **ONCLICK** | **ONOVER**

NONE	No triggering occurs
ONLOAD	Triggering occurs on page load
ONCLICK	Triggering occurs when the user clicks or taps the canvas
ONOVER	Triggering occurs when the user hovers the mouse on the canvas

4.3 AXIS COORDINATES

The axis coordinates follow the WebGL standard: Origin in the middle of canvas, X axis towards right, Y axis towards top, Z axis towards viewer.



The scale is from -.5 to +.5 in the X direction. If image format is 4/3 then Y scale would be from -.375 to +.375.

4.4 ALLOWED USER ACTIONS

Allowed user actions (mouse or touch) properties are accessed by `pagl.global.user`

Property	Type	Default Value	Description
rot	boolean	true	user is allowed to rotate camera
mov	boolean	true	user is allowed to translate camera
zoom	boolean	true	user is allowed to zoom camera lens
rotlimit	Boolean	false	if true, camera rotation is limited to the min/max values from the xrot, yrot, zrot camera animators.

4.5 CAMERA

Camera properties are accessed by `pagl.global.camera`

Property	Type	Default Value	Description
start	trigger	ONLOAD	Camera animation trigger mode
xmov	animator	min=-1, max=1	move camera along X axis
ymov	animator	min=-1, max=1	move camera along Y axis

zmov	animator	min=-1, max=1	move camera along Z axis (dolly)
zoom	animator	startvalue=.8 min=.1, max=10	Camera lens zoom
xrot	animator	min=-180, max=180	Rotate around X axis (vertical rotation)
yrot	animator	min=-180, max=180	Rotate around Y axis (horizontal rotation)
zrot	animator	min=-180, max=180	Rotate around Z axis

Example: Rotate camera 360° around subject in 10s when user clicks the canvas

```
var cam = pagl.global.camera;
cam.start = ONCLICK;
cam.yrot.max = 360; cam.yrot.speed = 36; cam.yrot.loop = false;
cam.yrot.alternate = false;
```

Run this example : 

4.6 LIGHT

Light properties are accessed by `pagl.global.light`

Property	Type	Default Value	Description
start	trigger	ONLOAD	Light animation trigger mode
vector	[x, y, z]	[0, 0, 1]	Initial light direction (default front). Typical values: Top left = [.408, -.408, .816] Top right = [-.408, -.408, .816] Bottom left = [.408, .408, .816] Bottom right = [-.408, .408, .816]
followcam	boolean	true	true if light is attached to camera, set it to false if xrot/yrot are used.
xrot	animator	min=-180, max=180	Rotate light around X axis (vertical rotation)
yrot	animator	min=-180, max=180	Rotate light around Y axis (horizontal rotation)

shading	animator	min=0, max=1.5	0 = no shading, 1.5 = very strong shading
ambient	animator	min=-1, max=1	Amount of ambient light. -1 = no ambient light, 1 = strong ambient light, defaults to 0 (average ambient light)
back	animator	startvalue=.5 min=0, max=1	Amount of back light
specular	animator	min=0, max=1	Amount of specular light (reflection), defaults to 0.

Example: Rotate light around model, control shading and specular with sliders.

Run this example 

4.7 ANIMATED OBJECTS

Up to 64 objects(*) can be defined and each object can be animated separately.
An individual object is accessed by `pagl.objects[i]` where i is the object number from 0.
Each animation object has following properties:

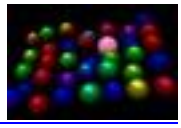
Property	Type	Default Value	Description
center	[x, y, z]	-	Object center point, all moves/rotates are relative to center
xmov	animator	animator default	Move object along X axis
ymov	animator	animator default	Move object along Y axis
zmov	animator	animator default	Move object along Z axis
xrot	animator	animator default	Rotate object around X (center relative)
yrot	animator	animator default	Rotate object around Y (center relative)
zrot	animator	animator default	Rotate object around Z (center relative)
scale	animator	startvalue = 1	Scale object

inflate	animator	startvalue = 1	Z factor from object. If inflate = 0 object is flat on X Y plane
aspect	animator	startvalue=1	Y/X factor from object. If aspect = 0 object is flat on X Z plane.
brightness	animator	startvalue=0 min = -1, max = 1	brightness correction. If -1, object is black
contrast	animator	startvalue=0 min = -1, max = 1	contrast correction, if -1, object has no contrast
hue	animator	startvalue=0 min = -1, max = 1	hue correction from -180° to 180°
saturation	animator	startvalue=0 min = -1, max = 1	saturation correction. If -1, object is displayed in black & white
red(**)	animator	startvalue=0 min = -1, max = 1	red correction. If -1, red is removed
green(**)	animator	startvalue=0 min = -1, max = 1	green correction. If -1, green is removed
blue(**)	animator	startvalue=0 min = -1, max = 1	blue correction. If -1, blue is removed
opacity	animator	startvalue=1 min = 0, max = 1	0: Object is fully transparent, 1: Object is fully opaque

objects animators share a common trigger defined by `pagl.main.start = trigger` (NONE | ONLOAD |ONOVER | ONCLICK). The default value is `main.start = ONLOAD`.

Notes:
(*) Older phones and tablets do not support 64 objects.
(**)red/green/blue are applied after hue and saturation.

Example: 35 spheres with random animation in the X,Y,Z directions, with sequential brightness control

Run this example 

5 METHODS

Only most used methods are listed here. For a full methods list refer to §7.6

Method	Description
renderFrame()	To be called in the window.requestAnimationFrame callback. If pagl.pause = true renderFrame has no effect, except if pagl.refresh has been set to true.

resizeCanvas()	To be called in case of canvas resize:
	pagl.canvas.width = 800;
	pagl.resizeCanvas();

updateBgrdColor(r, g, b, a)	Change the background color, values are from 0 to 1, "a" is the full animation opacity. pagl.updateBgrdColor(0 , 0, 1, 1); //blue background

6 ADVANCED FEATURES

6.1 MOUSE/TOUCH CAPTURE

Mouse/touch capture is useful to override the default API behavior. A typical application is to individually control objects according to user gesture.

Mouse/touch capture is implemented using a callback function defined at creation time:
var pagl = new PhotoAnim(canvas$str, paJSON, **onMouse**);
function onMouse(event){…}

Following global constants are used for mouse/touch capture actions:
action = NONE, ROTATE, TRANSLATE, ZOOM, INFO

Capture action property can be set with `pagl.mouseAction`

Property	Default Action	Valid Actions	Description
hover	INFO	NONE, INFO	Define action when mouse is hovered on canvas (no button, no key)
drag	ROTATE	NONE, ROTATE, TRANSLATE, ZOOM	Define action when mouse is dragged without right button and without SHIFT/CTRL key
dragshift	TRANSLATE	NONE, ROTATE, TRANSLATE, ZOOM	Define action when mouse is dragged with SHIFT key

dragctrl	ZOOM	NONE, ROTATE, TRANSLATE, ZOOM	Define action when mouse is dragged with CTRL key or with right button.
touch1	ROTATE	NONE, ROTATE, TRANSLATE, ZOOM	Define action for single finger touch
touch2pinch	ZOOM	NONE, ZOOM	Define action for 2 fingers pinch
touch2pan	TRANSLATE	NONE, ROTATE, TRANSLATE, ZOOM	Define action for 2 fingers pan
wheel	ZOOM	NONE, ROTATE, TRANSLATE, ZOOM	Define action when mouse wheel is used
click	INFO	NONE, INFO	Define action when canvas is clicked or touched

onMouse(event) is called upon user action. It should return true if it processes the action.

onMouse event format:

event.action	**event properties**	**Description**
NONE	-	Do not process this action
ROTATE	dx, dy	delta rotate by dx, dy in degrees (xrot.value += dx; yrot.value += dy;)
TRANSLATE	dx, dy	delta translate by dx, dy (xmov.value += dx; ymov.value += dy;)
ZOOM	kzoom	multiply camera zoom or object scale by kzoom (zoom.value *= kzoom)
INFO	x, y	x, y are in webGL coordinates (relative to canvas center). Useful to detect an object under cursor or to link to an url on click.

Examples:

1- Shift object0 when user drags canvas or uses single finger touch:

```
pagl.mouseAction.drag = pagl.mouseAction.touch1 = TRANSLATE;
```

```
function onMouse(event){
      if (event == TRANSLATE){
            pagl.objects[0].xmov.value += event.dx;
            pagl.objects[0].ymov.value += event.dy;
            return true; //tell processed
      }
}
```

2- A very simple object detect and move: A sphere and a cylinder, highlights the object when mouse hovers, moves the object on drag.

Run this example 

6.2 BACKGROUND IMAGE

Animations produced with PhotoAnim for Windows or 3Dthis may include a background image. The API allows animating this image, by defining property **bgrd** in pagl.global

bgrd property	Type	Default Value	Description
show	boolean	true	Show or hide the background image
start	trigger	NONE	Define background image animation trigger
xmov	animator	animator default	Move background image along X axis (no effect if followcam = true)
ymov	animator	animator default	Move background image along Y axis (no effect if followcam = true)
zoom	float	1	Background image scale
followcam	boolean	false	Background image position follows camera rotation
followamount	float	1	If followcam = true, defines camera rotation follow amount (can be negative).

Example:
Use mouse capture to control background image position.

Run this example 

6.3 TRAJECTORIES

Trajectories allow moving an object in a predefined way. A trajectory can be defined on any animator by specifying property **trajx**. Once trajx is defined, the other animator properties are ignored.
trajx is the index into a trajectory pool array accessed by `pagl.trajectory`.
Each trajectory entry has following format: sz, value, value… value
where sz is the trajectory size.
Trajectory durations are defined by `pagl.global.duration` in seconds (default 10 seconds). All trajectories have same duration.
A single special animator `pagl.main.main` has only useful properties **run**, **value**, **loop** and **alternate**. This allows looping and reverse play (default loop=true, alternate=true).

Example: We want object0 to move on a square x,y pattern in 5 seconds

```
var traj = pagl.trajectory;
var obj = pagl.objects[0];
obj.xmov.trajx = traj.length; // xmov traj start index
traj.push(5); //size
traj.push(-.25); traj.push(-.25); traj.push(.25); traj.push(.25);
traj.push(-.25);
obj.ymov.trajx = traj.length; // ymov traj start index
traj.push(5);
traj.push(-.25); traj.push(.25); traj.push(.25); traj.push(-.25);
traj.push(-.25);
pagl.main.main.alternate = false; // keep only loop
pagl.global.duration = 5;
```

Run this example 

Note: A new “replay” global animator has been introduced with V3.0, see §9 for a detailed description of objects and vertices trajectories.

7 API FULL REFERENCE CHART

This section assumes familiarity with common 3D principles and WebGL. It may be useful when creating animations from scratch.
Optional parameters are shown in *italic*

7.1 CREATION

```
var pagl = new PhotoAnim(canvas$str, animdef, onmouse, ontexloaded);
```

canvas$str: The id string of the rendering canvas
animdef: JSONstring | myAnim
JSONstring: A JSON encoded string, compressed, as downloaded from http://3dthis.com/download.htm
myAnim: [header, mesh, global, main]
onmouse: Optional callback on user mouse/touch actions, see §6.1
ontexloaded: Optional callback when texture is loaded (no parameters)

if canvas$str is false, animdef will just be parsed and no further action will occur. The parsed animation object can then be accessed by pagl.padef.

7.2 TRIGGERING CONSTANTS (trigger)

NONE, **ONLOAD**, **ONCLICK**, **ONOVER** (globals)

7.3 ANIMATOR

An animator defines the timely behavior of a given parameter.

```
var animator = {startvalue:float, value:float, min:float, max:float,
delay:float, speed:float, random:float, alternate:bool, loop:bool,
run:bool, trajx:int, trajpos:bool, trajzoom:bool, sync:"property"}
```

Defaults: startvalue:0, value:startvalue, min:-bigvalue, max:+bigvalue, delay/speed/random:0, alternate/loop/run:true, trajx, trajpos ,trajzoom, sync: undefined.
trajx is the start trajectory index into **vxobjtraj** array (see §7.4.2.4), trajx=0 or undefined means no trajectory. if a trajectory is defined, setting trajpos true adds startvalue to the current trajectory value, setting trajzoom true multiplies the current trajectory value by startvalue. The main purpose is to allow user actions even when a trajectory is defined.

sync allows synchronizing the animator on a property. For example, if we want to synchronize an object rotation to the camera rotation, we can define `sync:"cyrot"` because cyrot is the current camera y rotation.

7.4 ANIMATION DEFINITION

```
var myAnim = [header, mesh, global, main];
```

After PhotoAnim creation, myAnim can be accessed by pagl.padef

7.4.1 pagl.header (myAnim[0])

```
header = {bgrdcolor:array, aspect:float, projection:obj, vrphoto:obj}
```

bgrdcolor: [r, g, b, a] rgba range from 0 to 1, 'a' is full animation opacity.
aspect: defaults to canvas.width/canvas.height. May be used when the texture aspect is different from the original image aspect.
projection: {ortho:[l, r, b, t]} or {perspective:[fovy, aspect, znear, zfar]}
default projection: {perspective:[45, 1, .1, 100]}
vrphoto: Allows defining a virtual reality sequence of images, see new features V2.1 §8.

7.4.2 pagl.mesh (myAnim[1])

```
mesh = [vertices, texcoord, triangles, vxobjtraj, texture];
```

7.4.2.1 mesh[0] - vertices array

```
vertices = [x, y, z, trajx, nx, ny, nz, … ];
```
7 values for each vertex.
x, y, z: vertex coordinates
trajx: vertex trajectory, start index into vxobjtraj, allows vertex animation, see §9.
special values: trajx=0 – no trajectory, trajx=-1 – vertex belongs to background image.
nx, ny, nz: vertex normal vector – can be computed using method calcNormals(vertices)

Note: before V3.0, vertices count was limited to 65536. There is no limitation now, as most WebGL implementations accept extension OES_element_index_uint, which specifies vertices indexes on 32 bits.

7.4.2.2 mesh[1] - texcoord array

```
texcoord = [u, v, …]
```
2 values for each entry, parallel to vertices.
Count of entries should be same as vertices count (vertices.length/7 == texcoord.length/2)
u, v define a point from the texture image, u, v coordinates are relative to texture top left, u towards right and v towards bottom, in the range 0, 1.

7.4.2.3 mesh[2] - triangles array

```
triangles = [vx1, vx2, vx3, … ]
```
3 values for each entry, vertex indexes. Triangles should be oriented CCW for external faces.

7.4.2.4 mesh[3] - vxobjtraj array

Trajectory pool, holding vertices and animators trajectories.
vxobjtraj[0] must be 0 See §9 for a full description of trajectories.

7.4.2.5 mesh[4] - texture

texture can be an **url** string, a **dataurl** string, a **canvas** element or a **video** element.
dataurl is the recommended way as it allows a single file for animation definition. Also it prevents browser from throwing cross-origin exceptions when run locally on a computer.
texture url/dataurl supported image formats are jpeg and png.

texture width and height must be a power of two. Practical maximum texture size is 4096x4096 to be playable on most recent devices.
Only one texture is supported. If an animation needs multiple textures, then they should be merged into a single texture image (PhotoAnim for Windows does that automatically).

7.4.3 pagl.global (myAnim[2])

```
global = {duration:float, bgrd:obj, camera:obj, light:obj, user:obj};
```

duration: needed only if active trajectories, default value = 10 seconds

7.4.3.1 global.bgrd

Defines the animation of an optional background image. A background image should be included in the texture and is identified by vertices trajx = -1

```
bgrd = {show:bool, start:trigger, xmov:animator, ymov:animator, zoom:float,
followcam:bool, followamount:float};
```

defaults: show:true, start:NONE, zoom:1, followcam:false, followamount:1

7.4.3.2 global.camera

Defines camera animation

```
camera = {movecamera:bool, start:trigger, xmov:animator, ymov:animator,
zmov:animator, xrot:animator, yrot: animator, zrot:animator, zoom:animator}
```

defaults: movecamera:true, start:ONLOAD, zoom.startvalue:.8

if movecamera is false, the scenery will be moved.

7.4.3.3 global.light

Defines light animation

```
light = {start:trigger, vector:[x, y, z], followcam:bool, xrot:animator,
yrot:animator, shading:animator, ambient:animator, back:animator,
specular:animator}
```

defaults: start:ONLOAD, vector:[0, 0, 1], followcam:true, back.startvalue:.5

7.4.3.4 global.user

Defines allowed user mouse/touch actions

```
user = {rot:bool, mov:bool, zoom:bool, rotlimit:bool}
```

Default: rot = mov = zoom = true, rotlimit=false

If **rotlimit** is true, the user cannot rotate the camera more than the max/min values specified in xrot, yrot.

7.4.4 pagl.main (myAnim[3])

```
main = {main:animator, replay:animator, start:trigger, objs:[object1, …
objectn]};
```

main.main and **main.replay** are used to control objects and vertices trajectories. See §9.

start defaults to ONLOAD and controls the main animator.

Up to 64 objects can be defined by main.objs[], each animated separately

7.4.5 pagl.objects (main.objs) -> object = pagl.objects[i]

```
object = {center:[x, y, z], vxstart:int, xmov:animator, ymov:animator,
zmov:animator, scale:animator, inflate:animator, aspect:animator,
xrot:animator, yrot:animator, zrot:animator, brightness:animator,
contrast:animator, hue:animator, saturation:animator, red:animator,
green:animator, blue:animator, opacity:animator};
```

center: object center point, the object will move/scale around this point
vxstart: vertex start index for this object, successive objects vxstart should be in increasing order.
inflate: object Z amount
aspect: object Y/X amount

defaults:
scale, inflate, aspect, opacity – startvalue:1
brightness, contrast, hue, saturation, red, green, blue – min:-1, max:1

7.5 PROPERTIES

Property	Type	Description
pagl.error	read	false if no error, or error message (WebGL not supported)
pagl.pause	Read/write	true to pause animation, false to run it
pagl.oldtime	write	set to -1 to restart animation
pagl.refresh	write	true to request frame rendering

pagl.canvas	read	a reference to the canvas object
pagl.hasanim	read	true if at least one animator is active
pagl.hastrigger	read	true if some animation is triggered by a mouse/touch action
pagl.mouseAction	read/write	see mouse/touch capture
pagl.padef	read/write	reference to myAnim
pagl.header	read/write	reference to myAnim[0]
pagl.mesh	read/write	reference to myAnim[1]
pagl.global	read/write	reference to myAnim[2]
pagl.main	read/write	reference to myAnim[3]
pagl.objects	read/write	reference to main.objs object array
pagl.vertices	read/write	reference to mesh[0]
pagl.texcoord	read/write	reference to mesh[1]
pagl.triangles	read/write	reference to mesh[2]
pagl.trajectory	read/write	reference to mesh[3]
pagl.forcedtime	write	in ms, force play at a specific time, see §8.
cxrot, cyrot, czrot, cxmov, cymov, ctransl	read	current camera position

7.6 METHODS

Method	Description
renderFrame()	To be called in the window.requestAnimationFrame callback. If pagl.pause = true renderFrame has no effect, except if pagl.refresh has been set to

	true.

resizeCanvas()	To be called in case of canvas resize:
	pagl.canvas.width = 800;
	pagl.resizeCanvas();

updateBgrdColor(r, g, b, a)	Change the background color, values are from 0 to 1, "a" is the full animation opacity. pagl.updateBgrdColor(0 , 0, 1, 1); //blue background
updateLight()	To be called after a change in global.light.vector
updateProjection()	To be called after a change in header.projection
updateTexture(tex)	tex can be a canvas/video or an image string (url or dataurl)
calcNormals(vertices)	compute vertices normal. 'vertices' should have same format as in mesh.vertices. Note: this is computer intensive, do not call it from the window.requestAnimationFrame callback…
refreshVertices(vertices)	To be called after a change in vertices (mesh[0]), including after calcNormals
updateTexcoord(texcoord)	To be called after a change in texcoord (mesh[1])
updateBgrdShow	To be called after a change in global.bgrd.show
padefCheck()	For debugging, checks the validity from myAnim and logs errors on browser console. Is also called at PhotoAnim creation time if PADEBUG = true
drawFrame(frame)	Used in Virtual Reality mode do render a specific frame, see §8.

7.7 OPTIONAL CALLBACKS

7.7.1 onmouse(event) – defined at PhotoAnim creation time

See Mouse/Touch capture

7.7.2 ontexloaded() – defined at PhotoAnim creation time

Used to display a 'load in progress' message/icon when the texture image is an external resource

7.7.3 ontrigger

```
pagl.ontrigger = myFunction;

function myFunction(){…}
```

myFunction is called when an animation is started or stopped by an user action (ONOVER or ONCLICK). Useful to cancel “click to start” messages or control sound.

8 NEW FEATURES V2.1

8.1 FORCED TIME

The forcedtime property in milliseconds allows rendering a frame at a given time outside of real time. It is useful to convert an animation to a different format, like animated GIF.
Example:

```
pagl.oldtime = -1;
pagl.forcedtime = 0;
pagl.renderFrame(); // render frame at time 0
//… do something with the animation canvas
pagl.forcedtime = 100; // render frame at time 100ms
pagl.renderFrame();
// … do something with the animation canvas
```

8.2 VIRTUAL REALITY HANDLING

A texture can hold a series of individual pictures, which can be played like a video, with added user interactivity: The user can control with mouse/fingers or device orientation which picture is being played. This is used on 3Dthis.com for apps Live Selfie and iVideo.
Virtual Reality is defined by the header.vrphoto object (myAnim[0].vrphoto)

```
vrphoto = {shoots:int, dangle:float, cols:int, texkw:float, texkh:float};
```

shoots: total number of pictures in the image
dangle: angle in degrees between each picture
cols: number of picture columns in the texture
texkw: texture width coefficient (picture_width/texture_width)
texkh: texture height coefficient (picture_height/texture_height)

Vertices and triangles must be organized as follows:

```
var vertices = [-.5,.5,0, 0, 0, 0, 1, .5,.5, 0, 0, 0, 0, 1,  .5,-.5, 0, 0,
                0, 0, 1, -.5,-.5, 0, 0, 0, 0, 1];

var texcoord = [0,0, 0,0, 0,0, 0,0];

var triangles = [0,2,1, 0,3,2];
```

All picture sizes should be identical.

Example:
The following texture consists of 8 pictures, organized in 2 rows of 4 columns. Each picture is 256x256 for a total texture size of 2048 x 1024.

```
var vrphoto = {shoots:8, dangle:45, cols:4, texkw:.25, texkh:.5};
```

Run this example

Animation is controlled by the camera yrot animator (pagl.global.camera.yrot).
Setting yrot.min / yrot.max determines the first and last frames limits. To play all frames, set yrot.min to 0 and yrot.max to vrphoto.shoots * vrphoto.dangle.
For a 360° view, like in the example above, set yrot.min = -bigvalue, yrot.max = bigvalue.

Method drawFrame(frame) may also be invoked to render a given frame.

9 TRAJECTORIES

Trajectories are the key to 3D animation. The API implements object trajectories (i.e. camera trajectories), individual vertex trajectory and shared vertices trajectories. Trajectories are controlled by 2 special animators: pagl.main.main and pagl.main.replay. Shared vertices trajectories are very useful to animate complex 3D models like facial expressions with a minimum data amount. This is known as mesh distortion.
All trajectories are defined in a single array pagl.trajectory, the first array entry must be zero.

9.1 MAIN ANIMATOR

pagl.main.main is preset with values min=0, max=1, startvalue=0, speed=1/global.duration. These properties should not be changed. Properties which can be modified are main.value (from 0 to 1), main.run, main.alternate, main.loop

9.2 REPLAY ANIMATOR

pagl.main.replay has the following special format:

```
pagl.main.replay = {duration:float, time:float, run:bool, audio:url,
offset:float, onended:function}
```

It must be created using **pagl.addReplay(duration)**, which sets run to false and time to 0.
duration: total animation duration in seconds.
time: current time from 0 to 1.
audio: optional, not handled by the API
offset; in seconds, optional, offset between audio playback and animation, not handled by the API.
onended: optional function called when time reaches 1
The replay animator is forward only and loop. To stop it when it finishes playback, the onended function must set run = false.

9.3 OBJECTS TRAJECTORIES

An object trajectory can be defined on any animator except main.main and main.replay by setting property **trajx**, which is the trajectory start index into **pagl.trajectory**.
The trajectory format is

```
sz, value, value, …
```

If sz is >0 then sz values follow and are related to the main animator.
If sz is <0 then –sz values follow and are related to the replay animator.

Example: rotate camera from -90 to 90 degrees using replay animator

```
pagl.trajectory = [0, -2, -90, 90];
pagl.global.camera.yrot.trajx = 1;
```

9.4 VERTICES TRAJECTORIES

A vertex trajectory can be defined on any vertex by setting pagl.vertices[7*vx + 3] to the trajectory index start in pagl.trajectory.

9.4.1 SINGLE VERTEX TRAJECTORY

```
sz, tx,ty,tz,  tx,ty,tz, …
```

with sz>0 being the number of triplets (tx,ty,tz)
In this case, the main.main animator is used and the tx, ty, tz values are added to the initial vertices x, y, z.

9.4.2 SHARED VERTEX TRAJECTORY

```
n, y,kx,ky,kz, y,kx,ky,kz…
```

with n < 0, -n being the number of y,kx,ky,kz entries
y is a shared trajectory start index into pagl.trajectory. The shared trajectory format is
sz, value,value,….
If sz > 0, sz values follow and main.main animator is used.
If sz < 0, -sz values follow and main.replay animator is used.

kx, ky, kz are weights on the shared trajectory values:
The original x, y, z vertex value is replaced by x + kx * value, y + ky * value, z + kz * value.

Note that this allows a vertex to share multiple trajectories. Typically, on face animation, a vertex position may depend on several key dots like mouth opening or mouth width.

10 RESERVED SYMBOLS, DEBUGGING AND LIMITS

10.1 RESERVED SYMBOLS

Following global symbols are **reserved** by the API and should not be used/modified by other JavaScript code:
CanvasMatrix4, vertexshader, fragmentshader, shaderproblem, gphotoanim, PA_InitHandlers, PA_Grid_Mesh, NONE, ONLOAD, ONOVER, ONCLICK, ROTATE, TRANSLATE, ZOOM, INFO

10.2 DEBUGGING

All examples from this documentation can be run locally on computer (no need to upload to a server). However, when using external resources like image textures or video, cross-origin security errors may be thrown by some browsers (i.e. Chrome). In this case they must be run from a server.

Setting PADEBUG to true, allows checking the animation definition when creating PhotoAnim. Potential errors are reported on the browser console.

10.3 LIMITS

Maximum number of objects: 64, may be less on older mobile devices.
Maximum texture size: 4096x4096 px (not checked but advisable to be compatible with most platforms)

11 CONTACT AND FEEDBACK

Your feedback is highly welcomed!!!
Please send any comment or bug report to webmaster@3dthis.com