Pixels Live Wallpaper

Thank you!

Thank you for install Pixels Live Wallpaper, now you can create your own live wallpapers. The app was created keeping in mind simplicity, you don't need to learn all the tags just open one of the examples included and copy and paste the code in your project.

The app works reading an xml file stored in your SD card, this xml is easy to edit or create using an xml editor, notepad or any other text editor.

Remember that all your designs run in HTML5 as well. Enjoy adding all your projects on your website or even using the internet browser as a development platform.

Please read carefully the next instructions about how to create xml files. For more information remember to visit our website:

http://pixelslivewallpaper.github.io/jadsdsengine/

Table of contents

How can I open xml files? 4
How can I create xml files? 4
Basic 6
Simple animations 9
Pro animations 10
Images 13
Containers 31
Audio players 32
Text areas 34
Global variables 39
Timers 41
Gestures 43
Image alterations 44
Event tags 46
Conditionals 56
Examples 68
Find help quickly 73
Android options 74

How can I open xml files?

Exist thousands of options for this purpose. The easiest one is your Windows Notepad, otherwise any other text editor will be ok.

How can I create xml files?

You can use any xml file included in this application for your animations, just save it using “Save As” and pick a different name.

Is strongly recommended use “**Notepad++**”, is free and a very powerful tool.

All your animations need to be **saved inside the folder “PixelsLW”** on your SD card. Create a new folder, pick a name and include all your images and the xml inside the folder. The xml file must have the **same name** as the folder.

The xml file used in your animation is divided in sections, where "Basic" and "Images" are the most important.

Remember all the tags are case sensitive.

With “`<!--`” and “`-->`” you can comment xml tags and they will be ignored by the app. Example:

```
<!--<tag>false</tag>-->
```

The structure for all the xml files should looks like this:

```xml
<?xml version="1.0"?>
<doc>
  <basic>...</basic>
  <audioPlayers>...</audioPlayers>
  <textAreas>...</textAreas>
  <globalVariables>...</globalVariables>
  <timers>...</timers>
  <javaScriptActions>...</javaScriptActions>
  <gestures>...</gestures>
  <images>...</images>
</doc>
```

Note: Remember that the tag “textAreas” also can be included inside the tag “images”. Including the text areas inside the images area, allows you print images above the text. This can be useful if your project has messages that need to be printed above everything.

Basic

- **screenPhoneResolution**: For "*simple animation*" is recommended leave the values in "0". This instruction means use all the screen resolution of your phone. If you decide to change it for another custom resolution, keep in mind use a value smaller or equal to your screen phone resolution.

  For "*pro animations*" is recommended set a resolution because these animations require more memory and Android is quite limited about live wallpapers. 320x480 is a nice resolution, but you can try other ones if you wish.

  Using the screen resolution in "0" will tie your animation to your phone resolution, trying to run it in other resolutions (especially a smaller one) will cause some problems. The app always will try to fit the screen keeping the aspect ratio but if that is not possible the canvas size will be used instead. We strongly recommend use a resolution, it will be a guarantee that your animation will look the same everywhere doesn't matter the device.

  Note: You can increase the quality in the examples included with the app, almost all of them are using as "screenPhoneResolution" the values 320 x 480. The idea is used less memory and make possible enjoy the wallpapers even users with old smartphones.

  If you have a new smartphone maybe you can increase the resolution. The rule is simple, you must use as base the "screenPhoneResolution" included, example: 320x2, 480x2 = 640,960. If your smartphone has a screen resolution smaller or equal than the new values you can change it.

  Remember that Android is quite limited about live wallpapers, if after try a new resolution Android shut down the app is because your smartphone can't use it, in that case, change it again to 320x480.

- **antiAlias**: If you are stretching your images maybe this option needs to be set to true. It will help to reduce the distortion. By default is false.

- **canvasResolution**: It's the resolution for your project. The app will stretch your wallpaper until fit the phone screen always keeping the aspect ratio. Please check the next image.

```
<basic type="animations">
  <screenPhoneResolution>
    <width>320</width>
    <height>480</height>
  </screenPhoneResolution>
  <canvasResolution>
    <maxWidth>256</maxWidth>
    <maxHeight>128</maxHeight>
  </canvasResolution>
  <bgColor>#ff000000</bgColor>
  <grid>
    <addGrid>false</addGrid>
  </grid>
  <saveBattery>true</saveBattery>
</basic>
```

Like you can see the canvas was stretched and the width is bigger than the phone resolution. When this happens, the app will move the wallpaper automatically when the user moves his desktop screens. Usually, the average desktop has three screens.

For pixel art is recommended use small resolution, the main idea is stretch the images and create a pixelated effect.

- **bgColor**: Declare the default color for your background. Remember it uses the same logic used in HTML #ARGB, the values need to be in Hexadecimal.

  Example: `#ff63aeff (#Alpha-Red-Green-Blue)`

  Alpha is used for transparency levels where “ff” means no transparent color. We recommend learn more about this on internet.

- **grid:** It is one of the main features of Pixels Live Wallpaper and is available for all the animations, it creates that pixel effect that we all love.
  - **addGrid:** Enable or disable the effect. Possible values: True or False.
  - **gridColor:** Allow set a color for the grid.
  - **addPixelBetweenGrid:** Add a cool pixel effect. Possible values: True or False.

Icons:

After finish your project you must include an image named “icon.png”. The size is 64x64 pixels and save it inside the same folder.

The icon will be used for create a dynamic list with all your animations.

Simple animations

Simple animation is the best way to start. The xml is easy to understand and it doesn't require too many instructions.

Connect your phone to a computer and find a folder named "PixelsLW" in your SD card, inside you are going to find more folders with examples. Open a folder named "Pixels1" and then the xml file "Pixels1.xml".

Inside the tag "basic" we have "type", for this kind of animation the value need to be "simple".

```
<basic type="simple">
```

Basic section:

Simple animation only uses a special tag for the basic configuration.

changeImageEverySeconds: It's the time used between images. After the last one, the loop will start again. 1000 = 1 second.

Images section:

Only the tag <images> can be declared for simple animations. You can use these tags for change the background color and grid.

- **bgColor:** If your design requires it, it changes the background color.

- **grid:** The same tag "grid" used in basic section can be used here. A different grid can be used for each image.

Pro animations

After learn about simple animations and understand the logic about how the xml tags are used, you are ready for this new challenger.

Inside the tag “basic” we have “type”, for this kind of animation the value need to be “animations”.

```
<basic type="animations">
```

Basic section:

- **saveBattery**: This option allows reduce the frame rate for save battery. Possible values: True or False.

- **camera**: Allow control the camera, remember the size of the area for the camera in the same used in “**screenCanvasResolution**” tag. Please check the page 71 for more information.

    - **align**: Choose Top, Middle, Bottom or Custom. You’ll need these tags:

        - **customAlignLeft**: The coordinates for horizontal align.
        - **customAlignTop**: The coordinates for vertical align.

```
<align>Custom</align>
<customAlignLeft>-650</customAlignLeft>
```

    - **autoHorizontalScroll - autoVerticalScroll**: Allow move the camera automatically. You’ll need these tags:

        - **iniPosition**: Decide where the animation will begin. The values are between 0 and 1, where 0 is the beginning and 1 is the end. For example: 0.5 is the half. Use “;” for join the coordinates x and y.

```
<iniPosition>0;0</iniPosition>
```

- **iniPositionInPixels**: We strongly recommend use the previous tag “iniPosition” instead but, if you are confused, use this tag adding a value in pixels. Remember use negative values for move the camera to the left.

```
<iniPositionInPixels>-950;0</iniPositionInPixels>
```

- **pixelsMove**: The camera moves by default every 2 pixels. You can try another value.

- **cameraSpeed**: Change the animation speed, a higher number means a slower speed.

  Example:

```
<camera>
  <autoHorizontalScroll>true</autoHorizontalScroll>
  <iniPosition>0;0</iniPosition>
  <pixelsMove>-2</pixelsMove>
  <cameraSpeed>1</cameraSpeed>
</camera>
```

- **addBlackLines:** Create a nice cinematic effect. You’ll need these tags:

  - **blackLinesHeight:** The height for the black lines in pixels.

  - **blackLinesUpperTop & blackLinesLowerTop:** When black lines are used you have two, at the top of the screen or at the bottom. Here you decide in pixels the position.

```
<addBlackLines>
  <blackLinesHeight>50</blackLinesHeight>
  <blackLinesLowerTop>294</blackLinesLowerTop>
</addBlackLines>
```

- **customScroll:** Move the camera using the tags “moveAnimation” or “pathAnimation”. You can use the tags “iniPosition” and “iniPositionInPixels”. Remember that you can use the tags “iniPosition” and “iniPositionInPixels” and decide the start position for the camera.

```
<camera>
  <customScroll>
    <moveAnimation>
      <iniLeft>0</iniLeft>
      <maxLeft>-1920</maxLeft>
      <pixelsMoveLeft>2</pixelsMoveLeft>
      <pixelsMoveTop>0</pixelsMoveTop>
    </moveAnimation>
  </customScroll>
</camera>
```

- **events**: Trigger events with the camera. Please see the “Event tags” section for more information.

```
<events>
  <event>
    <eventType>TriggerActionByPosition</eventType>
    <eventActions>
      <eventAction>StopMove</eventAction>
    </eventActions>
    <whenGlobalVariableIs>
      <variable>
        <id>screen</id>
        <value>0</value>
      </variable>
    </whenGlobalVariableIs>
    <whenCurrentPositionIs>-960;0</whenCurrentPositionIs>
  </event>
</events>
```

- Remember that is possible control the camera using events, just use as id “Camera”. Check the page 47 for an example.
- The tags “iniPosition” and “iniPositionInPixels” are only used for auto or custom scrolls, do not confuse them with the align tags “customAlignLeft” and “customAlignTop”.

- **fps**: Change the default frame rate (20fps). You can add the attribute “showfps” to show on screen the current frame rate. Example:

```
<fps showfps=”true”>30</fps>
```

Images

- **fileName:** The file name of your image. Remember use the complete file name, including the extension. “MyImage.png”, for example.

  Optional: You can use the attribute “from” when the image is in another folder or if you just want borrow images from other animations.

  ```
  <fileName from="mapzLake">whirlpool1.png</fileName>
  ```

- **speed:** Change the animation speed, a higher number means a slower speed.

- **id**: Sometimes you are going to need add an id. Please check the event “`EditAnimationByID`” in the event tags section.

- **left & top:** These are used as coordinates for the images. Remember that these coordinates are based in the resolution used in the tag “canvasResolution” or “createBackgroundLayer” if you are using a background layer.

- **flipImage:** Flip the image, you have two options: “Horizontal” or “Vertical”.

- **coordinates:** Sometimes is necessary place the same image in difference places, using this tag instead the tags “left & top” is a way to do it easier. Use “;” for left and top and “,” for the next coordinate. Example: 6;41, 55;3, 108;30.

- **frames:** Load a group of images and create an animated object. Change the animation speed with the tag “framesAnimationSpeed” (see below). Leaving one of the frames empty, the background will be used as frame allowing you save some memory. Check the example named “jamr2.xml” for more information.

  - **frame**: This tag represent one frame and require other tags for configurations.

    - **fileName**: The file name of your image. Remember use the complete file name, including the extension. “MyImage.png”, for example. This tag is necessary and must be included.

Note: A nice trick is leave the tag empty and the background will be used as frame allowing you save some memory. The first frame never can be empty.

- **delay**: You can control the animation speed in real time using this tag. Like previous speed tags, a higher number means a slower speed. This tag is optional.

- **applyAlterationsToThisFrame**: Please check “Image alterations” section for more information.

- **currentFrame**: Declare the default frame index. Default value is 0.

- **loadFramesByImageIndex**: Allow you save memory re-using images from another “image tag”, just use its index number.

```
<?xml version="1.0"?>
<doc>
  <basic type="animations">...</basic>
  <images>
    0 <image>...</image>
    1 <image>...</image>
    2 <image>...</image>
    3 <image>...</image>
    4 <image>...</image>
    5 <image>...</image>
    6 <image>...</image>
    7 <image>...</image>
    8 <image>...</image>
    9 <image>...</image>
   10 <image>
        <frames>
          <loadFramesByImageIndex>2</loadFramesByImageIndex>
        </frames>
        <fillWithPattern>
          <repeatTimesHorizontal>2</repeatTimesHorizontal>
        </fillWithPattern>
        <attachToMainLayer>false</attachToMainLayer>
        <alpha>100</alpha>
        <speed>6</speed>
        <left>0</left>
        <top>164</top>
      </image>
   11 <image>...</image>
   12 <image>...</image>
  </images>
</doc>
```

Remember these numbers were added as a reference, they are **not** visible in your xml editor.

- **framesAnimationSpeed**: Change the animation speed, a higher number means a slower speed.

- **framesAnimationStop:** By default is false, changing its value to “true” the animation will be frozen. Remember you can toggle its value using events.

- **reUseTagFramesFromPreviousImage**: Allow you re-use images from other “image tags” and save some memory.

- **convertGifAniToFrames:** Convert an animated GIF to frames.

  Optional: Use the attribute “from” when the image is in another folder or if you just want borrow images from other animations.

```
<convertGifAniToFrames from="toads">toad.gif</convertGifAniToFrames>
```

- **convertImageSequenceInFrames:** Convert a sequence of images in frames. Please check the example:

  - **filePreName:** The file name before the numbers. Leave it empty if you do not need it.
  - **startNumber:** The lowest number in the sequence.
  - **endNumber:** The very last number in the sequence.
  - **formatNumber:** The format used for the numbers. Leave it empty if you do not need it.
  - **fileNameExtension:** The file extension of the image.

Image sequence: pic000120.jpg, pic000121.jpg, pic000122.jpg… pic000179.jpg

```
<images>
  <image>
    <frames>
      <convertImageSequenceInFrames>
        <filePreName>pic</filePreName>
        <startNumber>120</startNumber>
        <endNumber>179</endNumber>
        <formatNumber>######</formatNumber>
        <fileNameExtension>jpg</fileNameExtension>
      </convertImageSequenceInFrames>
    </frames>
    <left>0</left>
    <top>0</top>
    <speed>1</speed>
  </image>
</images>
```

Note: Remember Android is quite limited about the memory used for live wallpapers. Load a huge animated GIF or a big sequence of images, probably you are going to get an error for excess the memory limit.

Optional: Use the attribute "from" when the images are in another folder or if you just want borrow images from other animations.

```
<convertImageSequenceInFrames from="re0b2">
```

- **repeatThisLoopForTimes:** Repeat the animation a certain number of times and then the event "FrameAnimationRepeatLoopEnds" will be triggered**.**

- **frameCollection:** An animated object is able to have multiple frames. We can have frames for different kinds of animation. For Example: walking, look around, jump, etc. Remember you can change them with the action "ChangeFrameCollection". Please check "Events tags" for more information.

  - **repeatThisLoopForTimes**: This tag can be used here inside the tag "frameCollection" and has the same behavior, it repeats the animation a certain number of times and then the event "FrameAnimationRepeatLoopEnds" will be triggered.

  - **applyAlterationsToAllFrames**: Please check "Image alterations" section for more information.

  - **convertGifAniToFrames:** Convert an animated GIF to frames.

  - **convertImageSequenceInFrames**: Convert a series of images in frames.

- **defaultFrameCollection:** The frameCollection tags have an index number like the tag "loadFramesByImageIndex" explained previously. This tag allows you decide which collection will be loaded first. Default value is 0.

- **returnBackwardToFirstFrame**: An animation is an endless loop of frames that returns to the first frame after reach the last one. Changing the value to true, the animation after reach the last frame will return backward instead go to the first one. The default value is false.

- **colorCycling**: This app cannot be completed without this feature. Color Cycling was used in old video games as an alternative for save memory. The technique consists in use only one frame and the app literally "moves the colors" creating a feeling of animation.

With imagination you can create really beautiful designs with few lines of code. You will need these tags:

- **fileName**: The file name of your image.
- **backwardAnimation**: By default the color animation is forward, but changing this tag to true it moves backward.
- **basicPaletteColor**: These are the colors to be animated. The format is simple A, R, G, B (Alpha, Red, Green, Blue) and for the next color use "-".

Example:

```
<image>
  <frames>
    <colorCycling>
      <fileName>colorcycling.png</fileName>
      <basicPaletteColor>
                255,82,115,247-255,164,247,82-255,82,231,247-
                255,247,82,82-255,82,115,40
      </basicPaletteColor>
    </colorCycling>
  </frames>
  <left>317</left>
  <top>100</top>
</image>
```

Here we have an example of color cycling animation. The colors are moved forward.

- **reUseImageFromPreviousImage & loadImageByImageIndex**: Like the tags previously explained inside the tag "frames" with similar names, these ones allow you save memory. They re-use the images loaded is a previous image tag. In this case, use it only when the image is not animated.

- **fillWithPattern**: This is great for backgrounds, you can fill areas in your project using an image as pattern. The tags "repeatTimesHorizontal" or "repeatTimesVertical" allow you repeat the pattern multiple times.

Animated patterns are compactible just use the tag “frames”.

```
<image>
  <frames>
    <frame>
      <fileName>dcbackft1.png</fileName>
    </frame>
    <frame>
      <fileName>dcbackft2.png</fileName>
    </frame>
    <frame>
      <fileName>dcbackft3.png</fileName>
    </frame>
  </frames>
  <fillWithPattern>
    <repeatTimesHorizontal>4</repeatTimesHorizontal>
  </fillWithPattern>
  <speed>6</speed>
  <left>0</left>
  <top>164</top>
</image>
```

- **alpha**: Change the level of transparency in an image. The values are between 0 and 255, where 0 is completely transparent and 255 is the opposite.

- **moveAnimation**: Move an image from a point to another one. The trajectory is calculated by the app. Please check “**pathAnimation**” for another alternative.

  Use “pixelsMoveLeft” and “pixelsMoveTop” to indicate how many pixels the engine will move the image. If pixelsMoveLeft is 0 means vertical movements, both with values mean diagonal movements or use negative values for move it backward. By default the values are 2 pixels for both.

- **pixelsMoveLeft**: Horizontal movements.
- **pixelsMoveTop**: Vertical movements.
- **iniLeft & iniTop**: The start position for the image.
- **maxLeft & maxTop**: The final position for the image.
- **moveAnimationSpeed**: Like the tag “speed”, it changes the animation speed.
- **moveAnimationStop**: By default is false, but if your design required it, change it to “true” and the animation will be frozen. Remember you can change it to "false" again using “actions”.
- **returnToTheBeginning**: By default the image after reach the limits of the area return backward to the beginning. Changing this tag to true the image will be moved instantly to the beginning.

When we need move our image horizontally or vertically is easy to understand the configuration but in some cases our design requires diagonal movements and here is where people get confused. Please check the next images for understand the basics of moveAnimation.

Here “pixelsMoveLeft” and “pixelsMoveTop” are using the same values, causing the image keeps moving in a diagonal direction.

In the previous image an area is created with iniLeft, iniTop, maxLeft & maxTop. This means that the image can only move inside this area, after reach the limits automatically will return.

Playing with “pixelsMoveLeft” and “pixelsMoveTop” values change the direction of the image. Use negative values for **backward** movements.

Here we have another example but only for vertical movements, for reach that purpose we change the value of “pixelsMoveLeft” to 0.

Here “pixelsMoveLeft” was changed to 0 for vertical movements.

- **moveEndLessAnimation**: Creates horizontal or vertical loops following an image pattern. This is useful for animate roads or backgrounds.
    - **direction**: The animation can be done in 4 directions: Forward, Backward, Down and Up.
    - **pixelsMove**: The image pattern is moved following the quantity of pixels declared in this tag. Higher values create a feeling of speed, an image with a value 1 will move slower than another one with 4. Use negative values for backward movements.
    - **moveEndLessAnimationSpeed**: Like in previous speed tags, you can reduce or increase the speed using this tag.
    - **moveEndLessAnimationStop:** By default is false, but if your design required it, change it to “true” and the animation will be frozen. Remember you can change it to "false" again using “actions”.

```
<image>
  <fileName>trees.png</fileName>
  <moveEndLessAnimation>
    <direction>Backward</direction>
    <pixelsMove>3</pixelsMove>
  </moveEndLessAnimation>
  <left>0</left>
  <top>5</top>
  <speed>1</speed>
</image>
```

```
<image>
  <fileName>grass.png</fileName>
  <moveEndLessAnimation>
    <direction>Backward</direction>
    <pixelsMove>7</pixelsMove>
  </moveEndLessAnimation>
  <left>0</left>
  <top>107</top>
  <speed>1</speed>
</image>
```

moveEndLessAnimation is usually used for animate roads

- **pathEndLessAnimation**: Like the previous tag, it creates a loop but here we have more options. This is useful for create snow effect or clouds in animated skies.
    - **iniLeft, iniTop, maxLeft & maxTop**: You are not only limited to vertical or horizontal movements. These tags create a line between two points and will be used as path.
        - **iniLeft & iniTop**: The start point.
        - **maxLeft & maxTop**: The final point.
    - **numberOfLayers:** By default is 1, It means the quantity of times that the pattern will be printed horizontally.
    - **distanceBetweenLayers**: Change the distant (in pixels) between layers. Use this tag if you are working with multiples layers. See previous tag.
    - **numberOfImagesPerLayer**: You can repeat the image pattern vertically how many times is required.
    - **maxRandomNumberPerLayerInPixels**: If you are trying to create snow, rain, etc. this can be useful. The distance for the image pattern between the layers will be random. If you are creating an animated background, probably this value would be useful in 0.
    - **repeatLoop**: By the default is true but if you design require it, change it to false and after the loop is finished will stop.
    - **pixelsMove**: The image pattern is moved following the quantity of pixels declared in this tag. Higher values create a feeling of speed, an image with a value 1 will move slower than another one with 4. Use negative values for backward movements.
    - **pathEndLessAnimationSpeed**: Like in previous speed tags, you can reduce or increase the speed using this tag.

- **pathEndLessAnimationStop:** By default is false, but if your design required it, change it to “true” and the animation will be frozen. Remember you can change it to "false" again using “actions”.

- **createBackgroundLayer:** Usually our phones have desktops with multiples screens for make us easier find an application and keep everything in order. When we move from a screen to another, the background is moved slower creating a nice *parallax* effect.

  This tag creates an alternative “canvasResolution”. If you are planning create a background layer you must use a resolution smaller than the main “canvasResolution”, thereby the background layer will move slower when the scrolling happens.

  - **maxWidth & maxHeight:** It declares the size for the alternative canvas.

```
<?xml version="1.0"?>
<doc>
  <basic type="animations">
    <screenCanvasResolution>
      <width>196</width>
      <height>294</height>
    </screenCanvasResolution>
    <canvasResolution>
      <maxWidth>621</maxWidth>
      <maxHeight>224</maxHeight>
    </canvasResolution>
    <bgColor>#ff000000</bgColor>
    <camera>
      <autoHorizontalScroll>true</autoHorizontalScroll>
      <iniPosition>0;0</iniPosition>
      <pixelsMove>1</pixelsMove>
      <cameraSpeed>1</cameraSpeed>
    </camera>
  </basic>
  <images>
    <image>
      <fileName>dcback.png</fileName>
      <createBackgroundLayer>
        <maxWidth>458</maxWidth>
        <maxHeight>191</maxHeight>
      </createBackgroundLayer>
    </image>
    <image>...</image>
    <image>...</image>
    <image>...</image>
    <image>...</image>
    <image>...</image>
    <image>...</image>
    <image>...</image>
    <image>...</image>
    <image>...</image>
    <image>...</image>
    <image>...</image>
    <image>...</image>
    <image>...</image>
    <image>...</image>
  </images>
</doc>
```

1

2

3

The width used in createBackgroundLayer is smaller than the width used in canvasResolution.

- **reUseBackgroundLayerFromPreviousImage**: You don't need create a background layer multiple times, using this tag the previous background layer will be re-used again.

- **pathAnimation**: Similar to **moveAnimation**, it allows move an image from a specific point to another one creating a path with coordinates. The difference between both is freedom, moveAnimation is limited to use straight directions and pathAnimation not. The advance of **moveAnimation** is CPU – memory friendly and less complicated to configure.

    - **pathAnimationCoordinates**: pathAnimation includes different options for move your images. Please check the next tags:

        - **iniLeft, iniTop & maxLeft, maxTop:** The start point and final point. Use it for straight line movements.

        - **customCoordinates**: Create a path with coordinates. The syntax used is ";" for left and top and "," for the next coordinate. Example: 1;1, 2;2, 3;3. The image is moved following these coordinates.

        - **circleRadioWidth, circleRadioHeight, circleCenterPositionLeft & circleCenterPositionTop**: Create a circle path.

    - **pixelsMove**: The image pattern is moved following the quantity of pixels declared in this tag. Higher values create a feeling of speed, an image with a value 1 will move slower than another one with 4. Use negative values for backward movements.

- **pathAnimationSpeed**: Change the animation speed. A higher number means a slower speed.

- **startPosition**: pathAnimation creates a path of coordinates between “iniLeft & iniTop” and “maxLeft & maxTop”. This tag allows you decide where will be the start point.

- **pathAnimationStop**: By default is false, but if your design required it, change it to “true” and the animation will be frozen. Remember you can change it to "false" again using “actions”.

- **returnBackwardToTheBeginning**: By default the image after reach the limits of the area return backward to the beginning. Changing this tag to true the image will be moved instantly to the beginning.

- **imageTimer**: Is possible create a timer for trigger actions after a specific period of time. Please see the “Event tags” section for more information.

  - **timerDelay**: Decide the time between actions. 1000 means a second.
  - **timerStop**: By default is false and the timer starts just after the animation is loaded. Change it to true, will be in standby until the action “RunTimerOnlyForNumberTimes” is triggered.
  - **runTimerOnlyTimes**: Use it if you need run the timer only for a specific number of times. Remember the action “RunTimerOnlyForNumberTimes” allows change this value.

- **disableImage**: An image can be temporally disabled using this tag. Check the action “EnableByID” in the “Event tags” section for more information. The default value is false.

- **applyAlterations**: Please check “Image alterations” section for more information.

- **visible**: Do not confuse this tag with “disableImage”. The “visible” tag hides the image but all its animations are still running. Check the action “ChangeVisibleValue” ” in the “Event tags” section for more information. The default value is true.

- **width & height**: Allow you change the real size of the image by a new one in pixels.

- **rotate**: Rotate the image. This tag is incompatible with the tag “flipImage”. You must use “Image alterations” instead, check its section for more information. This tag is also incompatible with “imagePropertiesAnimation”, the tag value will be ignored.

- **perspective**: Change the image perspective. Remember include the attribute “orientation”, its values are “Vertical” or “Horizontal”. This tag is incompatible with “imagePropertiesAnimation”,

the tag values will be ignored. Example:

```
<perspective orientation="Horizontal">80</perspective>
```

- **imagePropertiesAnimation**: Allow you animate the properties Alpha, Scale and Rotation in an image.
    - **propetiesToAnimate:** Declare what properties will be animated.
        - **property:** Allow you configure all the settings for animate an image.
            - **propertyName**: Name of the property to animate.
                - **Alpha**: Like the previous tag alpha, this one allows you animate the transparency level of an image. The values are between 0 and 255, where 0 is completely transparent and 255 is the opposite.
                - **Scale**: Change the size of the image (width and height).
                - **Rotation**: Rotate the image.

Settings for Alpha:

- **minimumAlphaValue**: It is the minimum value that alpha property can reach, after that, the event “MinimumAlphaValueReached” will be triggered.

- **maximumAlphaValue**: It is the maximum value that alpha property can reach, after that, the event “MaximumAlphaValueReached” will be triggered.

- **animationAlphaAction**: It can be “Increase”, “Decrease” or “Stop” and allows animate the alpha value of the image.

- **updateAlphaValue**: Depending of what option was chosen in the previous tag, you can set a value for increase or decrease the alpha property of the image.

- **currentAlphaValue**: It is the default value that will be used for the alpha animation.

- **alphaSpeed**: Change the animation speed, a higher number means a slower speed.

Settings for Width & Height:

- **minimumWidthValue**: It is the minimum value that width property can reach, after that, the event “MinimumWidthValueReached” will be triggered.

- **maximumWidthValue**: It is the maximum value that width property can reach, after that, the event “MaximumWidthValueReached” will be triggered.

- **animationWidthAction**: It can be “Increase”, “Decrease” or “Stop” and allows animate the width value of the image.

- **updateWidthValue**: Depending of what option was chosen in the previous tag, you can set a value for increase or decrease the alpha property of the image.

- **currentWidthValue**: It is the default value that will be used for the width animation.

- **widthSpeed**: Change the animation speed, a higher number means a slower speed.

- **minimumHeightValue**: It is the minimum value that height property can reach, after that, the event “MinimumHeightValueReached” will be triggered.

- **maximumHeightValue**: It is the maximum value that height property can reach, after that, the event “MaximumHeightValueReached” will be triggered.

- **animationHeightAction**: It can be “Increase”, “Decrease” or “Stop” and allows animate the height value of the image.

- **updateHeightValue**: Depending of what option was chosen in the previous tag, you can set a value for increase or decrease the height property of the image.

- **currentHeightValue**: It is the default value that will be used for the height animation.

- **heightSpeed**: Change the animation speed, a higher number means a slower speed.

- **keepImageInItsPivotWhenScaling:** This property is false by default, change it to true if you want keep the image in its center when its size changes.

- **keepAspectRatioWhenScaling**: If you are working with the events MinimumWidthValueReached, MaximumWidthValueReached, MinimumHeightValueReached and MaximumHeightValueReached probably this tag will be handy. It will keep the aspect ratio of the image when its size is changed.

Settings for Rotation:

- **minimumDegreesValue**: It is the minimum value in degrees that the image can reach, after that, the event "MinimumRotationValueReached" will be triggered.

- **maximumDegreesValue**: It is the maximum value in degrees that the image can reach, after that, the event "MaximumRotationValueReached" will be triggered.

- **animationRotationAction**: It can be "Increase", "Decrease" or "Stop" and allows animate the image.

- **updateDegreesValue**: Depending of what option was chosen in the previous tag, you can set a value for increase or decrease the degrees of the image.

- **currentDegreesValue**: It is the default value that will be used for the animation.

- **rotationSpeed**: Change the animation speed, a higher number means a slower speed.

- **infiniteRotation**: The image after reach the maximumDegreesValue will return to the minimumDegreesValue automatically. The default value is “false”.

- **pivot**: These are the coordinates for the point of rotation. By default, will be the center of the image but you can add a custom coordinate. The configuration is simple use “;“ for split coordinates. Example: X;Y.

- **rotateImageAroundCoordinate**: These are the coordinates for keep the image orbiting a point. The configuration is simple use “;“ for split coordinates. Example: X;Y.

- **radius**: Depending on the previous tag, it declares the distance between the image and the point.

Settings for Perspective:

- **minimumPerspectiveDegreesValue**: It is the minimum value in degrees that the image can reach, after that, the event “MinimumPerspectiveValueReached” will be triggered.

- **maximumPerspectiveDegreesValue**: It is the maximum value in degrees that the image can reach, after that, the event “MaximumPerspectiveValueReached” will be triggered.

- **animationPerspectiveAction**: It can be “Increase”, “Decrease” or “Stop” and allows animate the image.

- **updatePerspectiveDegreesValue**: Depending of what option was chosen in the previous tag, you can set a value for increase or decrease the degrees of the image.

- **currentPerspectiveDegreesValue**: It is the default value that will be used for the animation.

- **perspectiveSpeed**: Change the animation speed, a higher number means a slower speed.

- **infiniteRotationPerspective**: The image after reach the maximumPerspectiveDegreesValue will return to the

minimumPerspectiveDegreesValue automatically. The default value is "true".

- **pivot**: These are coordinates for the point of rotation. By default, will be the center of the image but you can add a custom coordinate. The configuration is simple use ";" for split coordinates. Example: X;Y.

- **animationPerspectiveOrientation**: The perspective can be changed using the values "Vertical" or "Horizontal"

Note: Like happened with the tag "rotate", animate an image using the property "Rotation" will cause incompatibility with the actions flip image vertically or horizontally. For fix this problem use "Image alterations", check its section for more information and the example "smw2.xml" included.

Please check the next page, there we are playing with the properties alpha, width and height. Remember check the Event tags section for learn more about what events and actions are available for this animation.

```xml
<image>
  <fileName>ng3e.png</fileName>
  <left>49</left>
  <top>18</top>
  <alpha>0</alpha>
  <imagePropertiesAnimation>
    <propetiesToAnimate>
      <property>
        <propertyName>Alpha</propertyName>

        <minimumAlphaValue>0</minimumAlphaValue>
        <maximumAlphaValue>255</maximumAlphaValue>
        <updateAlphaValue>40</updateAlphaValue>
        <currentAlphaValue>0</currentAlphaValue>
        <animationAlphaAction>Increase</animationAlphaAction>
      </property>
      <property>
        <propertyName>Scale</propertyName>

        <minimumWidthValue>2</minimumWidthValue>
        <maximumWidthValue>68</maximumWidthValue>
        <updateWidthValue>2</updateWidthValue>
        <currentWidthValue>38</currentWidthValue>
        <animationWidthAction>Decrease</animationWidthAction>

        <minimumHeightValue>2</minimumHeightValue>
        <maximumHeightValue>62</maximumHeightValue>
        <updateHeightValue>2</updateHeightValue>
        <currentHeightValue>32</currentHeightValue>
        <animationHeightAction>Decrease</animationHeightAction>
      </property>
      <property>
        <propertyName>Rotation</propertyName>

        <minimumDegreesValue>36</minimumDegreesValue>
        <maximumDegreesValue>180</maximumDegreesValue>
        <updateDegreesValue>6</updateDegreesValue>
        <currentDegreesValue>0</currentDegreesValue>
        <rotationSpeed>1</rotationSpeed>
        <animationRotationAction>Decrease</animationRotationAction>
      </property>
    </propetiesToAnimate>
  </imagePropertiesAnimation>
</image>
```

Containers

A container allow you group images. If the container changes its position all the images attached will move as well.

The tag is included inside the tag “images” like a normal tag “image” and you can add an image with the tag “fileName” if your animation needs it. Usually, containers are used for group images that is why the tag “fileName” is optional.

- **container:** Create how many containers you need.
    - **Id:** The name for you container.

Example:

```
<images>
  <container>
    <id>containerA</id>
    <moveAnimation>
      <iniLeft>0</iniLeft>
      <maxLeft>288</maxLeft>
      <iniTop>0</iniTop>
      <maxTop>288</maxTop>
      <pixelsMoveLeft>2</pixelsMoveLeft>
      <pixelsMoveTop>2</pixelsMoveTop>
    </moveAnimation>
    <left>100</left>
    <top>100</top>
    <speed>1</speed>
  </container>
  <image>
    <fileName>area.png</fileName>
    <containerIDToAttach>containerA</containerIDToAttach>
    <left>100</left>
    <top>100</top>
  </image>
</images>
```

The example has an image attached to a container.

The container includes the tag “moveAnimation” when it moves the attached image will move as well. You can group as many images you want. The images can even have their own move animations and everything will be automatically calculated.

The way how an image is attached to a container is using the tag “**containerIDToAttach**” and adding the container id. You can include multiple containers. However, an image can be attached only to one container for obvious reasons.

Using the tag “createBackgroundLayer” in containers or in images attached to containers won’t work because it doesn’t make any sense.

Images using the tags “pathEndLessAnimation” and “moveEndLessAnimation” cannot be attached to containers.

Audio players

- **playerID:** All the players created in this section will be used later by the animations. The only way to do that is assigning an id to the player.

- **audioFile:** The audio file name.

  Optional: You can use the attribute "from" when the audio file is in another folder or if you just want borrow files from other animations.

  ```
  <audioFile from="musicFolder">sound.m4a</audioFile>
  ```

- **loop:** If this properly is set true, the player will keep playing the same sound infinite times.

- **volume:** The values for volume are between 0 and 1. Where 0 is muted and 1 is the loudest.

- **autoPlay:** Set this property to true and the sound will be played just after loaded.

- **events:** You can include events here, please check the section "Event tags" for more information.

Example: Here we added background music to our project.

```
<audioPlayers>
  <audioPlayer>
    <playerID>backgroundAudio</playerID>
    <audioFile>A-Bit of Daft Punk.m4a</audioFile>
    <loop>true</loop>
    <volume>0.5</volume>
    <autoPlay>true</autoPlay>
  </audioPlayer>
</audioPlayers>
```

Another example: Here we used an image as button for play a sound.

```
<image>
  <fileName>icon.png</fileName>
  <left>250</left>
  <top>20</top>
  <events>
    <event>
      <eventType>OnTouchDown</eventType>
      <eventActions>
        <eventAction>PlayAudio</eventAction>
      </eventActions>
      <audioPlayerID>sfxAudio</audioPlayerID>
      <audioFileToLoad>laser.mp3</audioFileToLoad>
      <audioVolumeToUpdate>0.5</audioVolumeToUpdate>
    </event>
  </events>
</image>
```

Text areas

- **left & top:** They are used as coordinates for the text areas.

- **disableText**: Hide the text area. Check the actions “EnableByID” and “DisableByID” in the “Event tags” section for more information.

- **textAreaID:** All the text areas need an id, is the only way to send instructions from any other animations.

- **stopText:** The text declared inside the text areas is printed immediately. You can stop it setting this tag to false.

- **color:** Change the color of your text. Remember, we are using the same logic used in HTML #ARGB, the values need to be in Hexadecimal.

- **bold:** Set it true for make your text bold.

- **font:** Change the font in your text, just write a font name.

  Note: If you want include custom fonts in your animation you're going to need this tag:

  - **customFontFileName**: Here you need include the file name of your font. Remember, you can load fonts from other animations with the attribute “from” like previously explained in the tag “image”.

    ```
    <customFontFileName>MyFont.ttf</customFontFileName>
    ```

    Loaded from another animation:

    ```
    <customFontFileName from="loading">Star.ttf</customFontFileName>
    ```

- **size:** Change the size of your text.

- **italic:** Set it true for italicize your text.

- **printOneByOne:** If your idea is show your text character per character, set this tag to true and adjust the speed with the next tag.

- **speed:** If the tag “printOneByOne” was set true, it changes the animation speed, a higher number means a slower speed.

- **events:** You can include events here, please check the section “Event tags” for more information.

- **spaceBetweenLines**: The space between lines is calculated automatically, but can be done manually with this tag.

- **textFromJavascript**: Adding a JavaScript function to the property “functionToUpdateText” allow you update text dynamically from JavaScript. Please check the section “Include the engine in your website”.

  Example:

```
<textArea>
  <left>10</left>
  <top>300</top>
  <textAreaID>testJava</textAreaID>
  <stopText>true</stopText>
  <textData>
    <color>#ffffffff</color>
    <bold>true</bold>
    <printOneByOne>true</printOneByOne>
    <speed>2</speed>
    <font>Latha</font>
    <size>12</size>
    <italic>true</italic>
    <textFromJavascript>textFromJavascript1</textFromJavascript>
  </textData>
</textArea>
```

- **text**: Write your text here, but remember the special codes. Please check next table.

#10;	New line	You must write your text in a single line inside the xml file, thereby the app will replace the code moving the next text to a new line.
#38;	&	“&” is a reserved character in xml, use this code instead and the app will replace it for “&” before print the text.
#62;	>	“>” is a reserved character in xml, use this code instead and the app will replace it for “&” before print the text.
#60;	<	“<” is a reserved character in xml, use this code instead and the app will replace it for “&” before print the text.
#37;	%	“%” is a reserved character in xml, use this code instead and the app will replace it for “&” before print the text.

Note: If you are planning to add text with different colors and styles you need to follow this pattern:

#font-properties : italic , bold , color ARGB;

Example:

#font-properties : false , true , #FFFF0000;

After including the previous parameter the text will be printed in that way from that point.

Example:

```
<text>The car is #font-properties:false,true,#FFFF0000;red #font-
properties:false,false,#FF000000;and #font-properties:true,false,#FF0000FF;blue</text>
```

The text printed is: The car is **red** and *blue*

Notice that after the word “red” we use the parameter again to return the text to the default style, in this case, black. Otherwise, the text printed would be always red.

Adding an event to a button:

```
<textAreas>
  <textArea>
    <left>10</left>
    <top>200</top>
    <textAreaID>TextArea1</textAreaID>
    <textData>
      <color>#ffffffff</color>
      <bold>true</bold>
      <printOneByOne>true</printOneByOne>
      <speed>2</speed>
      <font>Latha</font>
      <size>12</size>
      <italic>true</italic>
      <text>One#10;Two#10;Three#10;Four#38;Five</text>
      <spaceBetweenLines>20</spaceBetweenLines>
    </textData>
    <textData>
      <text>Six#10;Seven#10;Eight</text>
      <font>Roboto</font>
    </textData>
  </textArea>
</textAreas>

<image>
  <fileName>button.png</fileName>
  <left>150</left>
  <top>50</top>
  <id>button1</id>
  <events>
    <event>
      <eventType>OnTouchUp</eventType>
      <eventActions>
        <eventAction>NextText</eventAction>
      </eventActions>
      <ids>TextArea1</ids>
    </event>
  </events>
</image>
```

The text printed will look like this:

One
Two
Three
Four & Five

After press the button:

Six
Seven
Eight

Important

Remember that the tag “textAreas” also can be included inside the tag “images”. Including the text areas inside the images area, allows you print images above the text. This can be useful if your project has messages that need to be printed above everything.

Global variables

Global variables work in a similar way like “custom conditions”, the big difference is that they can be used by any other objects (images, text, etc.) in your project and not only the current object.

- **Variable:** Create how many variables you need.
    - **Id:** The name for you variable.
    - **Value:** The default value.

Example:

```
<globalVariables>
  <variable>
    <id>screen</id>
    <value>0</value>
  </variable>
</globalVariables>
```

Here we have a global variable named “screen” and its default value is “0”. In the next example we can see how it works.

```
<event>
  <eventType>TriggerActionByPosition</eventType>
  <eventActions>
    <eventAction>StopMove</eventAction>
  </eventActions>
  <whenGlobalVariableIs>
    <variable>
      <id>screen</id>
      <value>0</value>
    </variable>
  </whenGlobalVariableIs>
</event>
```

The previous example has an event “TriggerActionByPosition” and only will be triggered when the global variable named “screen” have the value “0”.

We can update the global variables, please check the next example:

```
<events>
  <event>
    <eventType>OnTouchUp</eventType>
    <eventActions>
      <eventAction>UpdateGlobalVariables</eventAction>
    </eventActions>
    <globalVariablesToUpdate>
      <variable>
        <id>screen</id>
        <value>0</value>
      </variable>
      <variable>
        <id>screen2</id>
        <value>4</value>
      </variable>
    </globalVariablesToUpdate>
  </event>
</events>
```

Timers

Timers work in the same way as the tag “imageTimer” works, the only ones differences are the tags “id” and “events”.

- **id:** You can use as many timers you need, the only way to change their properties later is giving them an id.

- **timerDelay**: Declare the time between actions. 1000 means a second.

- **timerStop**: By default is false and the timer starts just after the animation is loaded. Changing its value to true, the timer will be in standby until the action “RunTimer” is triggered.

- **runTimerOnlyTimes**: Use it if you need run the timer only for a specific number of times. Remember the action “RunTimerOnlyForNumberTimes” allows change this value.

- **events:** After the period time previously declared, the actions included inside this tag “events” will be triggered. Please check the “Event tags” section for more information.

Note: The next page has an example about how to create a timer. The most important difference is the “events” tag, they do not use the tag “eventType” because the engine adds automatically the event “Timer”.

The tag “ids” doesn’t work here because doesn’t make sense, the way how timers change the object properties is using the action “EditAnimationByID”, this action is added automatically by the engine.

The tag “avoidRunThisEventMultipleTimes” will have “true” as value and cannot be changed.

```xml
<timers>
  <timer>
    <timerDelay>2000</timerDelay>
    <timerStop>false</timerStop>
    <id>timer1</id>
    <runTimerOnlyTimes>2</runTimerOnlyTimes>
    <events>
      <event>
        <eventActions>
          <eventAction>ChangeCustomCondition</eventAction>
          <eventAction>ChangeTimerDelay</eventAction>
        </eventActions>
        <editAnimationByIDOptions>
          <option>
            <id>image27</id>
            <editActions>
              <editAction>ChangeSpeedAnimation</editAction>
            </editActions>
            <newSpeedAnimation>10</newSpeedAnimation>
          </option>
        </editAnimationByIDOptions>
        <whenCustomConditionIs>fast</whenCustomConditionIs>
        <newTimerDelay>5000</newTimerDelay>
        <newCustomCondition>slow</newCustomCondition>
      </event>
      <defaultCustomCondition>fast</defaultCustomCondition>
    </events>
  </timer>
</timers>
```

Another example:

```xml
<timers>
  <timer>
    <timerDelay>2000</timerDelay>
    <id>timer1</id>
    <events>
      <event>
        <eventActions>
          <eventAction>ChangeCustomCondition</eventAction>
          <eventAction>ChangeTimerDelay</eventAction>
        </eventActions>
        <editAnimationByIDOptions>
          <option>
            <id>ghost1,ghost2,ghost3,ghost4,ghost5</id>
            <editActions>
              <editAction>StopAnimationAndMove</editAction>
            </editActions>
          </option>
        </editAnimationByIDOptions>

        <whenCustomConditionIs>run</whenCustomConditionIs>
        <newTimerDelay>2000</newTimerDelay>
        <newCustomCondition>stop</newCustomCondition>
      </event>
      <defaultCustomCondition>stop</defaultCustomCondition>
    </events>
  </timer>
</timers>
```

Gestures

Capturing gestures is a common feature in these days, especially in smartphones. At the moment the engine captures the basic 8 movements:

The actions triggered by the gestures are: SlideRight, SlideLeft, SlideUp, SlideDown, SlideUpRight, SlideDownRight, SlideUpLeft, SlideDownLeft.

Example:

```
<gestures>
  <events>
    <event>
      <eventType>SlideRight</eventType>
      <eventActions>
        <eventAction>EditAnimationByID</eventAction>
      </eventActions>
      <editAnimationByIDOptions>
        <option>
          <id>arrow</id>
          <editActions>
            <editAction>ChangeFrameCollection</editAction>
          </editActions>
          <changeFrameCollectionIndexTo>0</changeFrameCollectionIndexTo>
        </option>
      </editAnimationByIDOptions>
    </event>
  </events>
</gestures>
```

The example captures the gesture “SlideRight”, it happens when you slide your finger from left to right.

Image alterations

The app has three tags that allow us make changes and save some memory re-using images. These tags are: “applyAlterations”, “applyAlterationsToAllFrames” and “applyAlterationsToThisFrame”.

- **applyAlterations**: The alterations will be done to all the frames currently used.
- **applyAlterationsToAllFrames**: Only the frames inside the tag “frameCollection” will be affected.
- **applyAlterationsToThisFrame**: Only the frame inside the tag “frame” will be affected.

You can use all the tags at same time for multiple alterations. The next page includes an example.

Available alterations:

The tag “alterationsToDo” allows you apply the desire effect. Use “,” for multiple effects.

```
<alterationsToDo>ApplyNewPaletteColor,FlipVertical</alterationsToDo>
```

These are the available effects:

- **FlipVertical, FlipHorizontal**: Like the tag “flipImage” flip the images.
- **ApplyNewPaletteColor**: Change the current colors for new ones.

  You need the tags “basicPaletteColor” and “newPaletteColor” for change the colors. The format is simple A, R, G, B (Alpha, Red, Green, Blue) and for the next color use “-“.

```
<applyAlterations>
  <alterationsToDo>ApplyNewPaletteColor</alterationsToDo>
  <basicPaletteColor> 255,99,8,255-255,74,0,255-255,49,16,148-
                      255,41,0,198-255,57,140,239
  </basicPaletteColor>
  <newPaletteColor> 255,66,255,8-255,0,255,36-255,148,143,16-
                    255,0,198,65-255,57,239,168
  </newPaletteColor>
</applyAlterations>
```

You don’t need include all the colors, just those ones that you are planning to change.

Example:

```
<image>
  <frames>
    <frameCollection>
      <frame>
        <fileName>localc1up.png</fileName>
        <applyAlterationsToThisFrame>
          <alterationsToDo>FlipHorizontal</alterationsToDo>
        </applyAlterationsToThisFrame>
      </frame>
      <frame>
        <fileName>localc2down.png</fileName>
      </frame>
      <frame>
        <fileName>localc3up.png</fileName>
      </frame>
      <applyAlterationsToAllFrames>
        <alterationsToDo>ApplyNewPaletteColor</alterationsToDo>
        <basicPaletteColor>255,49,16,148-255,99,8,255-
                            255,41,0,198-255,74,0,255
        </basicPaletteColor>
        <newPaletteColor>55,49,16,148-55,99,8,255-
                    55,41,0,198-55,74,0,255
        </newPaletteColor>
      </applyAlterationsToAllFrames>
    </frameCollection>
    <frameCollection>
      <frame>
        <fileName>localc1upback.png</fileName>
      </frame>
      <frame>
        <fileName>localc2downback.png</fileName>
      </frame>
      <frame>
        <fileName>localc3upback.png</fileName>
      </frame>
    </frameCollection>
  </frames>
  <applyAlterations>
    <alterationsToDo>ApplyNewPaletteColor,FlipVertical</alterationsToDo>
    <basicPaletteColor>255,49,16,148-255,99,8,255-
                    255,41,0,198-255,74,0,255
    </basicPaletteColor>
    <newPaletteColor>255,99,0,33-255,181,0,99-255,148,0,33-255,181,0,66</newPaletteColor>
  </applyAlterations>
  <left>317</left>
  <top>145</top>
  <speed>5</speed>
</image>
```

Event tags

The app can trigger events when a specific situation happens.

```
<image>
  <fileName>ff6citya.png</fileName>
  <moveAnimation>
    <maxLeft>0</maxLeft>
    <maxTop>224</maxTop>
    <iniLeft>0</iniLeft>
    <iniTop>134</iniTop>
    <pixelsMoveLeft>0</pixelsMoveLeft>
    <pixelsMoveTop>-1</pixelsMoveTop>
  </moveAnimation>
  <events>
    <event>
      <eventType>MoveAnimationMaxTopReached</eventType>
      <eventActions>
        <eventAction>StopMove</eventAction>
      </eventActions>
      <disableEventsAfterTriggered>true</disableEventsAfterTriggered>
    </event>
  </events>
  <left>0</left>
  <top>224</top>
  <speed>5</speed>
</image>
```

These events allow you trigger “actions”. The example above, show us an event triggered by moveAnimation when it reached the maximum vertical top value, thereby an action will be triggered to stop the moveAnimation.

Events tag:

Inside the tag “events” we can declare default values. They will be used later in our projects for trigger actions. Also, you can use global variables for the same purpose. Please see the “Global variables” section for more information.

- **defaultCustomCondition**: If you are planning to use custom conditions a default value is required. Please see “Conditionals” section for more information.

```
<events>
  <event>...</event>
  <event>...</event>
  <event>...</event>
  <defaultCustomCondition>running</defaultCustomCondition>
</events>
```

Event Tag:

- **eventType**: Checks the events table for more information. Remember that is possible include multiples “eventType” tags in one “event” tag.
- **eventAction**: Checks the actions table for more information.
- **disableEventsAfterTriggered**: The event after triggered will be disabled and will not be triggered again.
- **avoidRunThisEventMultipleTimes**: Do not confuse with the previous tag, this one will prevent trigger the same event multiple times but will not be disabled.

```
<events>
  <defaultCustomCondition>run</defaultCustomCondition>
  <event>
    <eventType>OnTouchUp</eventType>
    <eventActions>
      <eventAction>EditAnimationByID</eventAction>
      <eventAction>ChangeCustomCondition</eventAction>
    </eventActions>
    <avoidRunThisEventMultipleTimes>true</avoidRunThisEventMultipleTimes>
    <whenCustomConditionIs>run</whenCustomConditionIs>
    <newCustomCondition>stop</newCustomCondition>
    <editAnimationByIDOptions>
      <option>
        <id>Camera</id>
        <editActions>
          <editAction>StopMove</editAction>
        </editActions>
      </option>
    </editAnimationByIDOptions>
  </event>
  <event>
    <eventType>OnTouchUp</eventType>
    <eventActions>
      <eventAction>EditAnimationByID</eventAction>
      <eventAction>ChangeCustomCondition</eventAction>
    </eventActions>
    <avoidRunThisEventMultipleTimes>true</avoidRunThisEventMultipleTimes>
    <whenCustomConditionIs>stop</whenCustomConditionIs>
    <newCustomCondition>run</newCustomCondition>
    <editAnimationByIDOptions>
      <option>
        <id>Camera</id>
        <editActions>
          <editAction>ResumeMove</editAction>
        </editActions>
      </option>
    </editAnimationByIDOptions>
  </event>
 </events>
```

- **Ids**: Usually when an event is triggered the actions affect the current object itself. Using this tag the actions will affect only the ids included inside the tag. Use “,” for multiple ids.

Note: After use the “ids” tag all the actions will follow it ignoring the current object. For affect the current object, you have two options.

- Include the current object id inside tag with the other ids.
- Create a new “event tag” with the same “eventType”.

```
<events>
  <event>
    <eventType>PathAnimationEndPointReached</eventType>
    <eventActions>
      <eventAction>StopMove</eventAction>
    </eventActions>
    <ids>image1,image2</ids>
  </event>
  <event>
    <eventType>PathAnimationEndPointReached</eventType>
    <eventActions>
      <eventAction>ResumeMove</eventAction>
    </eventActions>
    <ids>image3,image4</ids>
  </event>
</events>
```

In this example we need to create the event “`PathAnimationEndPointReached`” twice because the actions only will follow one ids tag.

Note: The actions “EnableByID” and “DisableByID” use the tag “ids” exclusively, it means that any other actions in the same event tag will affect only the current image. You can use a new event tag with the same “eventType tag” like we did in the previous example for more actions in other images.

Remember that the tag “ids” is quick way to run actions in multiple objects, but don’t forget the tag “editAnimationByID”.

Please check the page 54 for more information.

- **assignEvents**: This is a nice feature that allows you add or update events in other animations in real time. It is quite intuitive, just declare the id and assign the event.

  Please check the example, where we have assigned a new event to the audio player “sfxAudio”, it will play a new sound when the current one is finished.

```
<image>
  <fileName from="dx">icon.png</fileName>
  <left>0</left>
  <top>200</top>
  <events>
    <event>
      <eventType>OnClickUp</eventType>
      <eventActions>
        <eventAction>AssignEvents</eventAction>
      </eventActions>
      <assignEvents>
        <assignEvent>
          <assignID>sfxAudio</assignID>
          <events>
            <event>
              <eventType>PlaybackIsFinished</eventType>
              <eventActions>
                <eventAction>PlayAudio</eventAction>
              </eventActions>
              <audioPlayer>
                <playerID>backgroundAudio</playerID>
                <audioFile>A-Bit of Daft Punk.m4a</audioFile>
                <volume>0.3</volume>
                <autoPlay>true</autoPlay>
              </audioPlayer>
            </event>
          </events>
        </assignEvent>
      </assignEvents>
    </event>
  </events>
</image>
```

- **Audio tags**: Please check the section “Audio players” to learn all about the tags available. Remember use the action “UpdateAudioPlayer” for update the audio player properties.

  Examples:

```
<image>
  <fileName>icon.png</fileName>
  <events>
    <event>
      <eventType>OnClickDown</eventType>
      <eventActions>
        <eventAction>PlayAudio</eventAction>
      </eventActions>
      <audioPlayer>
        <playerID>sfxAudio</playerID>
        <audioFile>laser.mp3</audioFile>
        <volume>0.5</volume>
        <autoPlay>true</autoPlay>
      </audioPlayer>
    </event>
  </events>
</image>

<image>
  <fileName>icon.png</fileName>
  <events>
    <event>
      <eventType>OnClickDown</eventType>
      <eventActions>
        <eventAction>UpdateAudioPlayer</eventAction>
      </eventActions>
      <audioPlayer>
        <playerID>backgroundAudio</playerID>
        <volume>1</volume>
      </audioPlayer>
    </event>
  </events>
</image>

<image>
  <fileName>icon.png</fileName>
  <events>
    <event>
      <eventType>OnClickDown</eventType>
      <eventActions>
        <eventAction>PauseAudio</eventAction>
      </eventActions>
      <audioPlayer>
        <playerID>backgroundAudio</playerID>
      </audioPlayer>
    </event>
  </events>
</image>
```

- **onlyVerticalMovements – onlyHorizontalMovements:** These tags depend on the eventAction: DragAndDropMove. It will keep the image moving only horizontally or vertically.

- **minVerticalValue, maxVerticalValue - minHorizontalValue, maxHorizontalValue:** These tags depend on the eventAction: DragAndDropMove. They allow you create an area where the image can be drag and drop. These tags can be used in combination with the previous tags.

  Example 1:

```
<event>
  <eventType>DragAndDrop</eventType>
  <eventActions>
    <eventAction>DragAndDropMove</eventAction>
  </eventActions>
  <onlyHorizontalMovements>true</onlyHorizontalMovements>
</event>
```

  Example 2:

```
<event>
  <eventType>DragAndDrop</eventType>
  <eventActions>
    <eventAction>DragAndDropMove</eventAction>
  </eventActions>
  <minVerticalValue>10</minVerticalValue>
  <maxVerticalValue>179</maxVerticalValue>
  <minHorizontalValue>100</minHorizontalValue>
  <maxHorizontalValue>269</maxHorizontalValue>
</event>
```

- **editAnimationByIDOptions**: These are the necessary tags for the action “EditAnimationByID”.

    - **id**: The id of the object to edit.
    - **editActions**: Here you can use all the available actions. Please check Actions table.
    - **Previous tags**: The Actions table not only say what actions are available, it also says what tags you need. The next example the action “ChangeVisibleValue” needs the tag “newVisibleValue” for update the property “Visible” of the image.

    Please check the next example:

```
<event>
  <eventType>TriggerActionByCurrentFrame</eventType>
  <eventActions>
    <eventAction>EditAnimationByID</eventAction>
  </eventActions>
  <whenCurrentFrameIndexIs>12</whenCurrentFrameIndexIs>
  <editAnimationByIDOptions>
    <option>
      <id>Ball7</id>
      <editActions>
        <editAction>ResumeMove</editAction>
        <editAction>ChangeVisibleValue</editAction>
      </editActions>
      <newVisibleValue>true</newVisibleValue>
    </option>
  </editAnimationByIDOptions>
</event>
```

Another example:

```
<events>
  <event>
    <eventType>OnClickDown</eventType>
    <eventActions>
      <eventAction>EditAnimationByID</eventAction>
    </eventActions>
    <editAnimationByIDOptions>
      <option>
        <id>test</id>
        <editActions>
          <editAction>UpdateImagePropertiesAnimation</editAction>
        </editActions>
        <newImagePropertiesAnimation>
          <propetiesToAnimate>
            <property>
              <propertyName>Alpha</propertyName>
              <minimumAlphaValue>0</minimumAlphaValue>
              <maximumAlphaValue>100</maximumAlphaValue>
              <updateAlphaValue>40</updateAlphaValue>
              <currentAlphaValue>0</currentAlphaValue>
              <animationAlphaAction>Increase</animationAlphaAction>
            </property>
          </propetiesToAnimate>
        </newImagePropertiesAnimation>
      </option>
    </editAnimationByIDOptions>
  </event>
 </events>
```

Use your imagination to guess combinations, in this new example we are using the tag “eventAction” with “EditAnimationByID” and “editAction” with “UpdateImagePropertiesAnimation” for update “ImagePropertiesAnimation” of an image

Remember that “UpdateImagePropertiesAnimation” is not the only one tag available, check page 62 for more information about how update animations using actions.

Please check the next page for more information about the tag “ids”.

ids tag vs EditAnimationByID action

Depending on your project probably you are confused about the tag “ids” and the action “EditAnimationByID”, the difference between both is that “ids” affect only the object ids inside the tag ignoring the object that is triggering the action. The best way understand is with examples:

```
<event>
  <eventType>OnClickUp</eventType>
  <eventActions>
    <eventAction>ChangeVisibleValue</eventAction>
  </eventActions>
  <ids>ghost1,ghost2</ids>
  <newVisibleValue>false</newVisibleValue>
</event>
```

```
<event>
  <eventType>OnClickUp</eventType>
  <eventActions>
    <eventAction>EditAnimationByID</eventAction>
  </eventActions>
  <editAnimationByIDOptions>
    <option>
      <id>ghost1,ghost2</id>
      <editActions>
        <editAction>ChangeVisibleValue</editAction>
      </editActions>
      <newVisibleValue>false</newVisibleValue>
    </option>
  </editAnimationByIDOptions>
</event>
```

Using the tag “ids” is faster and intuitive, the previous example we can use both without worry about conflicts.

```
<event>
  <eventType>OnClickUp</eventType>
  <eventActions>
    <eventAction>ResumeMove</eventAction>
    <eventAction>ChangeCustomCondition</eventAction>
  </eventActions>
  <ids>ghost1,ghost2</ids>
  <newCustomCondition>alpha</newCustomCondition>
</event>
```

```
<event>
  <eventType>OnClickUp</eventType>
  <eventActions>
    <eventAction>ChangeCustomCondition</eventAction>
    <eventAction>EditAnimationByID</eventAction>
  </eventActions>
  <editAnimationByIDOptions>
    <option>
      <id>ghost1,ghost2</id>
      <editActions>
        <editAction>ResumeMove</editAction>
      </editActions>
    </option>
  </editAnimationByIDOptions>
  <newCustomCondition>alpha</newCustomCondition>
</event>
```

In the previous example we must be careful, using the tag “ids” with the action “ChangeCustomCondition” will update the custom conditions of all the ids included inside the tag and probably that is not what we want. In these cases the solution is the action “EditAnimationByID” that allows update values to other objects without affecting the current one.

Note: Another way to solve the conflict using the tag “ids” is creating a new event with the same “eventType”, one event will trigger the action “ChangeCustomCondition” and second one will trigger “ResumeMove” as long as you are not using the tag “avoidRunThisEventMultipleTimes”

```xml
<event>
  <eventType>OnClickUp</eventType>
  <eventActions>
    <eventAction>ChangeCustomCondition</eventAction>
  </eventActions>
  <newCustomCondition>alpha</newCustomCondition>
</event>
<event>
  <eventType>OnClickUp</eventType>
  <eventActions>
    <eventAction>ResumeMove</eventAction>
  </eventActions>
  <ids>ghost1,ghost2</ids>
</event>
```

Note: Remember conditionals are always tied to the object triggering the events. If you are trying to use the tags “ids” or “EditAnimationByID” and you want trigger an event only when a conditional match a value remember use “whenGlobalVariableIs” because is the only one conditional that can be shared among all the objects

Conditionals:

You can decide when the action will be triggered.

whenFrameCollectionIndexIs
If the current frame collection index matches with the tag then the action is triggered.

```
<event>
  <eventType>FrameAnimationRepeatLoopEnds</eventType>
  <eventActions>
    <eventAction>FlipImage</eventAction>
  </eventActions>
  <whenFrameCollectionIndexIs>0</whenFrameCollectionIndexIs>
</event>
```

whenCurrentPositionIs
If the current position matches with the tag then the action is triggered. The configuration is simple use “;“ for split coordinates. Example: X;Y.

```
<event>
  <eventType>FrameAnimationRepeatLoopEnds</eventType>
  <eventActions>
    <eventAction>ChangeFrameCollection</eventAction>
  </eventActions>
  <whenCurrentPositionIs>2;368</whenCurrentPositionIs>
  <changeFrameCollectionIndexTo>5</changeFrameCollectionIndexTo>
</event>
```

whenCustomConditionIs
Custom conditions allows trigger actions without depend on animation values.

```
<event>
  <eventType>Timer</eventType>
  <eventActions>
    <eventAction>StopAnimationAndMove</eventAction>
    <eventAction>ChangeCustomCondition</eventAction>
  </eventActions>
  <whenCustomConditionIs>running</whenCustomConditionIs>
  <newCustomCondition>stoppped</newCustomCondition>
</event>
<event>
  <eventType>Timer</eventType>
  <eventActions>
    <eventAction>ResumeAnimationAndMove</eventAction>
    <eventAction>ChangeCustomCondition</eventAction>
  </eventActions>
  <whenCustomConditionIs>stoppped</whenCustomConditionIs>
  <newCustomCondition>running</newCustomCondition>
</event>
<defaultCustomCondition>running</defaultCustomCondition>
```

whenCurrentFrameIndexIs

An animation contains a group of frames (images) that are quickly showed one by one creating the feeling of animation. These frames are repeated in a loop infinite times. We can trigger actions only when a specific frame is on screen.

```
<events>
  <event>
    <eventType>TriggerActionByCurrentFrame</eventType>
    <eventActions>
      <eventAction>EnableByID</eventAction>
    </eventActions>
    <whenCurrentFrameIndexIs>3</whenCurrentFrameIndexIs>
    <ids>ball</ids>
  </event>
</events>
```

whenRotationDegreesIs

If you are rotating an image you can trigger an event only when the image reaches a specify degree.

```
<events>
  <event>
    <eventType>RotationDegreesReached</eventType>
    <eventActions>
      <eventAction>ChangeAlpha</eventAction>
    </eventActions>
    <whenRotationDegreesIs>54</whenRotationDegreesIs>
    <alpha>255</alpha>
  </event>
</events>
```

whenPerspectiveDegreesIs

If you are changing the image perspective you can trigger an event only when the image reaches a specify degree.

```
<events>
  <event>
    <eventType>PerspectiveDegreesReached</eventType>
    <eventActions>
      <eventAction>ChangeAlpha</eventAction>
    </eventActions>
    <whenPerspectiveDegreesIs>54</whenPerspectiveDegreesIs>
    <alpha>255</alpha>
  </event>
</events>
```

whenGlobalVariableIs

Only the event is triggered if the global variable match its value. Remember that is possible create conditions using multiple global variables.

```
<event>
  <eventType>TriggerActionByPosition</eventType>
  <eventActions>
```

```
          <eventAction>StopMove</eventAction>
        </eventActions>
        <whenGlobalVariableIs>
            <variable>
              <id>fire</id>
              <value>a</value>
            </variable>
            <variable>
              <id>water</id>
              <value>b</value>
            </variable>
        </whenGlobalVariableIs>
      </event>
```

whenIsInsideTheArea

If an object position is inside the area, the event will be triggered. The tag needs the coordinates of the area: X1;Y1;X2;Y2.

```
    <image>
      <fileName>demo.png</fileName>
      <left>112</left>
      <top>52</top>
      <events>
        <event>
          <eventType>DragAndDrop</eventType>
          <eventActions>
            <eventAction>DragAndDropMove</eventAction>
          </eventActions>
        </event>
        <event>
          <eventType>OnClickUp</eventType>
          <eventActions>
            <eventAction>EditPosition</eventAction>
          </eventActions>
          <newCoordinates>112;306</newCoordinates>
          <whenIsInsideTheArea>64;254;255;445</whenIsInsideTheArea>
        </event>
      </events>
    </image>
```

Events:

Events are triggered when something of interest occurs. Please check the next list:

MoveAnimationMaxTopReached
moveAnimation reaches the minimum vertical value.
MoveAnimationMaxBottomReached
moveAnimation reaches the maximum vertical value.
MoveAnimationMaxLeftReached
moveAnimation reaches the minimum horizontal value.
MoveAnimationMaxRightReached
moveAnimation reaches the maximum horizontal value.
FrameAnimationRepeatLoopEnds
frameCollection when the tag “repeatThisLoopForTimes” is completed.
FrameAnimationLoopEnds
An animation contains a group of frames (images) that are quickly showed one by one creating the feeling of animation. These frames are repeated in a loop infinite times. When the last frame is showed, this event is always triggered.
TriggerActionByCurrentFrame
Similar to the previous event, it is triggered when a specific frame is showed.
CirclePathMaxRightReached
pathAnimation when “pathAnimationCoordinates” is used for create a circle and the image reaches the max right point.
CirclePathMaxLeftReached
pathAnimation when “pathAnimationCoordinates” is used for create a circle and the image reaches the max left point.
CirclePathMaxTopReached
pathAnimation when “pathAnimationCoordinates” is used for create a circle and the image reaches the max top point.
CirclePathMaxBottomReached
pathAnimation when “pathAnimationCoordinates” is used for create a circle and the image reaches the max bottom point.
PathAnimationStartPointReached
pathAnimation when the first coordinate of the path is reached.
PathAnimationEndPointReached
pathAnimation when the last coordinate of the path is reached.
Timer
imageTimer reaches its time.
FrameAnimationReturnBackwardToFirstFrameEnds
frameCollection when the tag “returnBackwardToFirstFrame” is completed.
TriggerActionByPosition
It’s triggered when the image reaches a specific coordinate.
OnClickDown or OnTouchDown
It’s triggered when the user touch the image.
OnClickUp or OnTouchUp
It’s triggered when the user touch and release the image.

PlaybackIsFinished
It’s triggered when an audio player finishes its sound.
StartPrintingText
It’s triggered when a text area starts to print.
PrintingNewLine
It’s triggered when a text area starts to print a new line.
EndPrintingText
It’s triggered when a text area finishes printing its text.
MinimumAlphaValueReached
It’s triggered when the alpha property reach its minimum value.
MaximumAlphaValueReached
It’s triggered when the alpha property reach its maximum value.
MinimumWidthValueReached
It’s triggered when the width property reach its minimum value.
MaximumWidthValueReached
It’s triggered when the width property reach its maximum value.
MinimumHeightValueReached
It’s triggered when the height property reach its minimum value.
MaximumHeightValueReached
It’s triggered when the height property reach its maximum value.
MinimumRotationValueReached
It’s triggered when the image rotate and reach its minimum value.
MaximumRotationValueReached
It’s triggered when the image rotate and reach its maximum value.
RotationDegreesReached
It’s triggered when the image rotate and reach a custom value.
MinimumPerspectiveValueReached
It’s triggered when the image perspective reaches its minimum value.
MaximumPerspectiveValueReached
It’s triggered when the image perspective reaches its maximum value.
PerspectiveDegreesReached
It’s triggered when the image perspective reaches a custom value.
DragAndDrop
It’s triggered when the user drag and drop an image.
SlideRight, SlideLeft, SlideUp, SlideDown, SlideUpRight, SlideDownRight, SlideUpLeft, SlideDownLeft
The events are triggered after capturing a gesture.
MoveStopped & MoveResumed
It’s triggered when a move animation (moveAnimation, pathAnimation, etc.) was stopped or resumed.
AnimationStopped & AnimationResumed
It’s triggered when the animation was stopped or resumed.

Actions:

ChangeFrameCollection
If you animation has multiples frame Collections, you can change the current collection for another one adding a new index.
Depend on these Tags: • **changeFrameCollectionIndexTo**: The index number to update.
FlipImage
Flip the image.
Depend on these Tags: • **flipType**: Flip the image, you have two options: “Horizontal” or “Vertical”. Default value “Horizontal”.
ResumeMove
The image movement is resumed.
StopMove
The image movement is stopped.
StopAnimationAndMove
The image movement and animation is stopped.
ResumeAnimationAndMove
The image movement and animation is resumed.
ResumeAnimation
The image animation is resumed.
StopAnimation
The image animation is stopped.
EditAnimationByID
It allows change the animation properties in other images by id.
Depend on these Tags (and what you want to update): • **editAnimationByIDOptions**: Please check the page number 52 for more information.
ChangeSpeedAnimation
The image speed is changed, including all the animations (frames and movements).
Depend on these Tags: • **newSpeedAnimation**: The new value to update. A higher number means a slower speed.
ChangeTimerDelay
Update the delay for the Timer.
Depend on these Tags: • **newTimerDelay**: If a Timer has been included, you can update the delay.
RunTimerOnlyForNumberTimes
Run a Timer only for a specific number of times.
Depend on these Tags: • **runTimerOnlyTimes**: The number of times to run.

RunTimer

Run a Timer.

StopTimer

Stop a Timer.

EditPosition

Change the current image position.

Depend on these Tags:

- **newCoordinates**: The configuration is simple use ";" for split coordinates. Example: X;Y.

UpdateMoveAnimation, UpdatePathAnimation, UpdateMoveEndLessAnimation, UpdatePathEndLessAnimation, UpdateMode7, UpdateFillWithPattern, UpdateImagePropertiesAnimation

They allow you update the animation properties. Check page 72 for another example.

Depend on these Tags:

- **new + animation name**: Please check the next example.

```
<event>
  <eventType>FrameAnimationRepeatLoopEnds</eventType>
  <eventActions>
    <eventAction>UpdateMoveAnimation</eventAction>
  </eventActions>
  <newMoveAnimation>
    <iniLeft>2</iniLeft>
    <maxLeft>100</maxLeft>
    <pixelsMoveLeft>-2</pixelsMoveLeft>
    <pixelsMoveTop>0</pixelsMoveTop>
  </newMoveAnimation>
</event>
```

Remember you don't need add all the values, just those ones that you are planning to update.

DisableImage

Disable the current image.

EnableByID

Enable images previously disabled. This action cannot be used multiple times inside the "events" tag.

Depend on these Tags:

- **ids**: Use "," for multiple image ids. Example: image1, image2.

DisableByID

Disable images. This action cannot be used multiple times inside the "events" tag.

Depend on these Tags:

- **ids**: Use "," for multiple image ids. Example: image1, image2.

ChangeSpeedAnimationForFrames

Similar to "ChangeSpeedAnimation" but only affects the frames.

Depend on these Tags:

- **newSpeedAnimationForFrames**: The new value to update. A higher number means a slower speed.

ChangeSpeedAnimationForMovements
Similar to “ChangeSpeedAnimation” but only affects the movements.
Depend on these Tags: • **newSpeedAnimationForMovements**: The new value to update. A higher number means a slower speed.
ChangeVisibleValue
Hide – unhide an image. Remember, all the animations are still running. Use “Disable image” for hide an image including its animations.
Depend on these Tags: • **newVisibleValue**: New visible value.
CreateRandomCoordinates
Change the current image position to a new one generated by random numbers.
Depend on these Tags: • **parametersForRandomCoordinates**: The configuration is “X(ini),X(max),Y(ini),Y(max)”, where the values are joined by “,”. Example: `<parametersForRandomCoordinates>696,800,246,286</parametersForRandomCoordinates>`
ChangeAnimation
Changes the current animation for a new one.

Depend on these Tags:
• **changeAnimationTo**: Change the current animation. Example:
 	
`<events>`
`  <event>`
`  <eventType>OnTouchDown</eventType>`
`  <eventActions>`
`  <eventAction>ChangeAnimation</eventAction>`
`  </eventActions>`
`  <changeAnimationTo>animationname</changeAnimationTo>`
`  </event>`
`</events>`

PlayAudio, StopAudio, PauseAudio, ResumeAudio, UpdateAudioPlayer
Allow you control the music in your animation. Please check the Audio players section and page 50 for more information.
ChangeBgColor
Change the background color.
Depend on these Tags: • **newBgColor**: Assign a new background color.

ChangeCustomCondition

Change the current custom condition.

Depend on these Tags:

- **newCustomCondition**: Assign the new value.
- **newCustomConditionRandom**: You can assign a new value from a range of random numbers. Use ";" to assign the range of random numbers: Minimum ";" Maximum.

  Example:

  ```
  <newCustomConditionRandom>1;6</newCustomConditionRandom>
  ```

  A number between 1 and 6 will be randomly chosen.

Attributes:

- **waitForNextFrame**: Update the value only after the current frame is printed. It will allow custom conditions to behave like global variables. New values are not updated instantly, only after the current frame is finished. It can be used in both "*newCustomCondition*" and "*newCustomConditionRandom*".
- **preventConsecutiveValues**: Prevent return the same random value twice.

  Example:

  ```
  <newCustomConditionRandom waitForNextFrame="true"
  preventConsecutiveValues="true">1;6</newCustomConditionRandom>
  ```

NextText

Give instructions to a text area to show the next text.

PreviousText

Give instructions to a text area to show the previous text.

StopText

Give instructions to a text area to stop printing text.

PlayText

Give instructions to a text area to continue printing text.

LoadTextByIndex

Give instructions to a text area to print a text by its index.

Depend on these Tags:

- **changeTextAreaIndexTo**: Assign the new value for the index.

```
<events>
  <event>
    <eventType>OnTouchUp</eventType>
    <eventActions>
      <eventAction>EditAnimationByID</eventAction>
    </eventActions>
    <editAnimationByIDOptions>
      <option>
        <editActions>
          <editAction>LoadTextByIndex</editAction>
        </editActions>
        <id>Area1</id>
        <changeTextAreaIndexTo>0</changeTextAreaIndexTo>
      </option>
```

```
            </editAnimationByIDOptions>
          </event>
        </events>
```

ChangeAnimationAlphaAction
Change the current action used for alpha in imagePropertiesAnimation.
Depend on these Tags: • **animationAlphaAction**: You can choose between “Increase”, “Decrease” and “Stop”.
ChangeAnimationWidthAction
Change the current action used for width in imagePropertiesAnimation.
Depend on these Tags: • **animationWidthAction**: You can choose between “Increase”, “Decrease” and “Stop”.
ChangeAnimationHeightAction
Change the current action used for height in imagePropertiesAnimation.
Depend on these Tags: • **animationHeightAction**: You can choose between “Increase”, “Decrease” and “Stop”.
ChangeAnimationRotationAction
Change the current action used for rotation in imagePropertiesAnimation.
Depend on these Tags: • **animationRotationAction**: You can choose between “Increase”, “Decrease” and “Stop”.
ChangeAnimationPerspectiveAction
Change the current action used for perspective in imagePropertiesAnimation.
Depend on these Tags: • **animationPerspectiveAction**: You can choose between “Increase”, “Decrease” and “Stop”.
ChangeSize
Change the size of an image.
Depend on these Tags: • **height**: The new height value in pixels. • **width:** The new width value in pixels.
ChangeDegrees
Change the current degree value used for rotate an image.
Depend on these Tags: • **degrees**: Set a new value.
ChangePerspectiveDegrees
Change the current degree value used for image perspective.
Depend on these Tags: • **perspectiveDegrees**: Set a new value.
UpdateGlobalVariable
Update a global variable, please check the “Global variables” section for more information.
Please check the example:

```
<events>
  <event>
    <eventType>OnTouchUp</eventType>
    <eventActions>
      <eventAction>UpdateGlobalVariables</eventAction>
    </eventActions>
    <globalVariablesToUpdate>
      <variable>
       <id>screen</id>
       <value>0</value>
      </variable>
   </globalVariablesToUpdate>
  </event>
</events>
```

DragAndDropMove

It works together with the event “DragAndDrop”. When the user drag an image, it follows your finger.

Please check the example:

```
<image>
  <fileName>demo.png</fileName>
  <left>112</left>
  <top>52</top>
  <events>
    <event>
      <eventType>DragAndDrop</eventType>
      <eventActions>
        <eventAction>DragAndDropMove</eventAction>
      </eventActions>
    </event>
  </events>
</image>
```

ChangePivot

By default an image rotates based in its center, but you can change that adding a new value.

Depend on these Tags:

- **pivot**: The configuration is simple use “;“ for split coordinates. Example: X;Y.

Tips

The next image explains how moveAnimation events are triggered.

The next image explains how pathAnimation triggers events when “pathAnimationCoordinates” is used for create a circle path.

Remember that circle paths have an end and a beginning, it means that the events “PathAnimationStartPointReached” and “PathAnimationEndPointReached” will be triggered as well.

Examples

<canvasResolution>
<maxWidth>144</maxWidth>
<maxHeight>72</maxHeight>
</canvasResolution>
<image>
<fileName>pr2bb2.png</fileName>
<left>11</left>
<top>21</top>
</image>
(0,0)
144
(11,21)
72
(144,72)
Untitled-2 copy.PNG - Paint
Home
View
Clipboard
Image
Tools
Brushes
Shapes
Size
Color 1
Color 2
Colors
Edit colors
11, 21px
144 × 72px
500%

Now is time for advance training, all the examples until now have been as easy as possible for give you a clear idea about how the app works.

This time we are going to learn how create an advance animation using almost all the available features.

Images section

The big confusion here is handle events, they are simple and require just a little bit of logic.

```
<events>
  <event>
    <eventType>MoveAnimationMaxLeftReached</eventType>
    <eventActions>
      <eventAction>StopMove</eventAction>
      <eventAction>ChangeFrameCollection</eventAction>
    </eventActions>
    <whenFrameIndexIs>0</whenFrameIndexIs>
    <changeFrameCollectionIndexTo>1</changeFrameCollectionIndexTo>
  </event>
  <event>
    <eventType>FrameAnimationRepeatLoopEnds</eventType>
    <eventActions>
      <eventAction>ChangeFrameCollection</eventAction>
      <eventAction>FlipImage</eventAction>
      <eventAction>ResumeMove</eventAction>
    </eventActions>
    <whenFrameIndexIs>1</whenFrameIndexIs>
    <changeFrameCollectionIndexTo>0</changeFrameCollectionIndexTo>
  </event>
  <event>
    <eventType>MoveAnimationMaxRightReached</eventType>
    <eventActions>
      <eventAction>StopMove</eventAction>
      <eventAction>ChangeFrameCollection</eventAction>
    </eventActions>
    <whenFrameIndexIs>0</whenFrameIndexIs>
    <changeFrameCollectionIndexTo>1</changeFrameCollectionIndexTo>
  </event>
</events>
```

A good example is moveAnimation tag, when an image reaches the limits will trigger an event and you can add actions.

The most common action for moveAnimation is **FlipImage**. In the image above we have an animated image walking, after reach the limits the image returns but we need flip it before otherwise the image will look like walking backward.

Another common action for moveAnimation is **ChangeFrames**. Sometimes we need a different animation when a specific situation happen, something like look around and then returns.

In these cases remember use frameCollection tags for multiples frames.

```
<image>
  <frames>
    <frameCollection>
      <frame>
        <fileName>dcene1r3.png</fileName>
      </frame>
      <frame>
        <fileName>dcene2.png</fileName>
      </frame>
      <frame>
        <fileName>dcene1r3.png</fileName>
      </frame>
      <frame>
        <fileName>dcene4.png</fileName>
      </frame>
    </frameCollection>
    <frameCollection>
      <frame>
        <fileName>dceneb1.png</fileName>
      </frame>
      <frame>
        <fileName>dceneb2r4.png</fileName>
      </frame>
      <frame>
        <fileName>dceneb3.png</fileName>
      </frame>
      <frame>
        <fileName>dceneb2r4.png</fileName>
      </frame>
      <repeatThisLoopForTimes>1</repeatThisLoopForTimes>
    </frameCollection>
    <framesAnimationSpeed>7</framesAnimationSpeed>
  </frames>
```

Frames for "walking"

Frames for "look around"

Use "whenFrameCollectionIndexIs" for trigger the action in a specific moment. Following the previous example:

```
<event>
  <eventType>MoveAnimationMaxLeftReached</eventType>
  <eventActions>
    <eventAction>StopMove</eventAction>
    <eventAction>ChangeFrameCollection</eventAction>
  </eventActions>
  <whenFrameCollectionIndexIs>0</whenFrameCollectionIndexIs>
  <changeFrameCollectionIndexTo>1</changeFrameCollectionIndexTo>
</event>
```

Only when "`whenFrameCollectionIndexIs`" is 0 and "`MoveAnimationMaxLeftReached`" is reached, the actions "`StopMove`" and "`ChangeFrameCollection`" are triggered, stopping moveAnimation and changing the frame collection index to 1.

Camera

By default the camera is static and fit with your phone desktop. In this example we are going to use the tags “autoHorizontalScroll” and “customScroll”.

```
<camera>
  <autoHorizontalScroll>true</autoHorizontalScroll>
  <iniPosition>0;0</iniPosition>
  <pixelsMove>-2</pixelsMove>
  <cameraSpeed>1</cameraSpeed>
</camera>
```

Adding the previous camera tag to your project, automatically the camera will move left to right in an endless loop.

The tag “iniPosition” decide where the camera start point will be. These values are simple to understand, you can play with values between 0 and 1, where 0 is the beginning of your design, 0.5 is the half and 1 is the very end.

```
<iniPosition>X;Y</iniPosition>
```

They are like coordinates, where the first value is X and the second one is Y.

Example: `<iniPosition>0.01223241;0.02037037</iniPosition>`

You can be quite precise if you wish, adding an accurate position. For example is your canvas width is 1000 pixels, the half would be 500 what means 0.5.

(500 * 1) / 1000 = 0.5

This will be the value used in “iniPosition”.

Another example using “customScroll”

```
<camera>
  <iniPosition>0.01223241;0.02037037</iniPosition>
  <cameraSpeed>1</cameraSpeed>
  <customScroll>
    <moveAnimation>
      <iniTop>-11</iniTop>
      <maxTop>-381</maxTop>
      <pixelsMoveTop>-2</pixelsMoveTop>
    </moveAnimation>
  </customScroll>
  <events>
    <event>
      <eventType>MoveAnimationMaxTopReached</eventType>
      <eventActions>
        <eventAction>UpdateMoveAnimation</eventAction>
      </eventActions>
      <newMoveAnimation>
        <iniLeft>-8</iniLeft>
        <maxLeft>-620</maxLeft>
        <pixelsMoveLeft>-2</pixelsMoveLeft>
      </newMoveAnimation>
      <disableEventsAfterTriggered>true</disableEventsAfterTriggered>
    </event>
  </events>
</camera>
```

Here we are using the tag “moveAnimation” to move the camera to a custom position and using an event to move the camera again to another position.

Find help quickly

You can download for free the wonderful "**Notepad++**" on Internet. We recommend the portable version, it is free and you don't need install anything, just copy the folder in your phone SD card and use it when you need it.

Notepad++ includes several useful options, one of them is "Find in files". You can get a list about where a tag is being used. Please check the next image:

Just open the xml included inside the examples and play with the values. The best way to learn is changing values and sees what happen.

Remember that all your designs run in **HTML5** as well. Enjoy adding your designs on your website or even use the internet browser as development platform.

Android options

These are the available options:

1. **Animations**: All the available animations inside the folder “PixelsLW” will be listed here.

2. **Brightness**: Change the contrast between the background and the foreground icons.

3. **Change animations**: The animation will change after loading the app. This option is a good idea if you like a different wallpaper after turn on your phone.

4. **Scrolling wallpaper**: This option allows you enjoy scrolling wallpapers if your phone doesn’t support it. Every time when you move between screens the app will move the wallpaper creating a nice parallax effect.

5. **Copy animations**: The app includes plenty examples to help you understand how to create your own wallpaper. Even if a folder is deleted by accident the app will copy the animation again. If you don’t like some animations or you just don’t need them anymore, disabling this option will prevent copy them again.

6. **Save battery**: Reduce the frame rate to save battery.

7. **Force AntiAlias**: Improve image quality on some screens.

8. **Booster**: Use it if your animations have a low frame rate.

9. **Help**: The app opens the Github page with all the information and examples about the Jadsds engine. Here you will learn how to create your own live wallpapers.