JADSDS Engine

Thank you!

JADSDS engine is a friendly tag based JavaScript animator created to make easier web projects using the power of HTML5.

At the moment the engine includes features like color cycling, scale and rotation, animated GIF support, multiple layers for parallax effects, capturing events and trigger actions and much more.

All your web projects run on Android as well, an app is included with the engine allowing you run your project on an Android device.

This is just the first step, more functions will be included in the future. All will depend if the people fall in love with the project.

For more information remember to visit the **Git** repository:

https://pixelslivewallpaper.github.io/jadsdsengine/

Also you more examples here:

https://pixelslivewallpaper.github.io/jadsdsengine/indexb.htm

https://pixelslivewallpaper.github.io/jadsdsengine/Zelda.htm

Table of contents

How can I open xml files?

The easiest way is using your Windows Notepad but any other text editor will be ok.

A nice option is the text editor "**Notepad++**", it is free and it has more features than the classic Notepad.
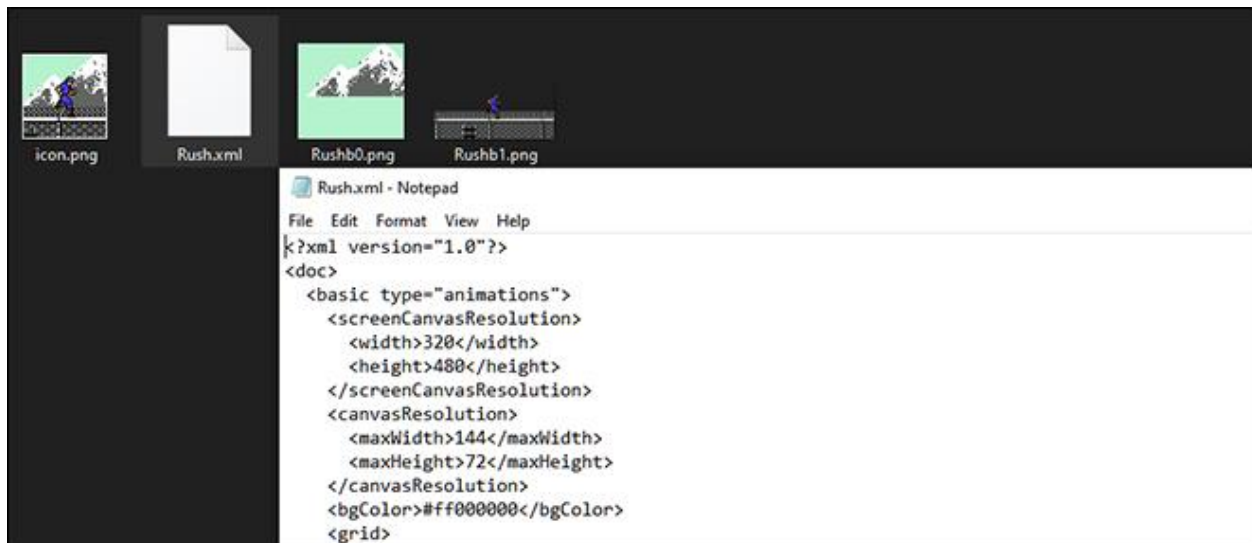
How can I create my animations?

In HTML5, all your animations need to be **saved inside a folder named "animations"** ([you can change it](#)) in your website. 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 new folder.

In Android, all your animations need to be **saved inside a folder named "JadsdsEngine"**. After installing the app and running it for the first time, the app will create the folder and inside of it will be an example animation. Keep in mind that the app will always run first the animation inside the folder "**main**" (also created after running the app), you can't delete this folder and it will be the start animation for all your projects. Also, inside the folder "JadsdsEngine" you have another folder "loading". It has the animation used during the loading times.

The xml file used in your animation is divided into 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 engine. Example:
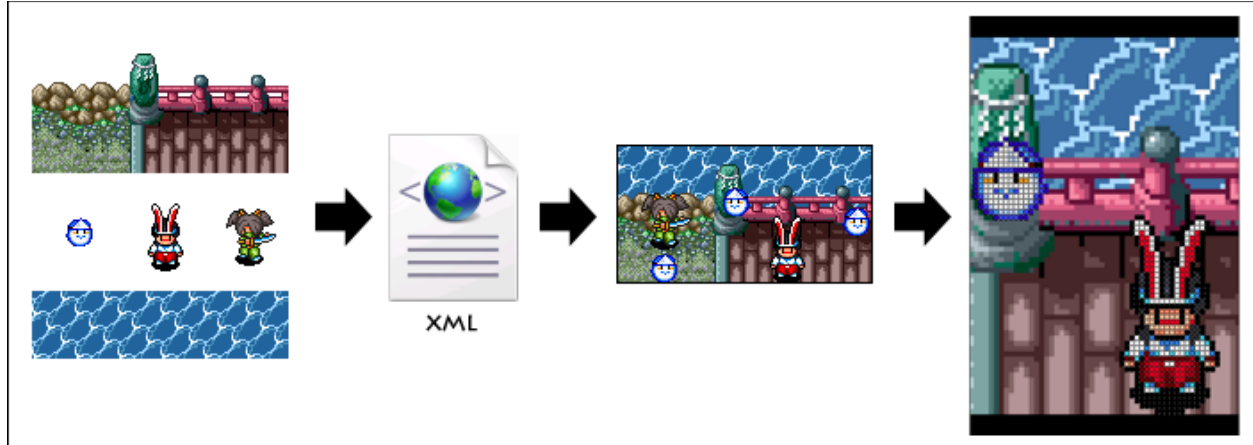
```
<!--<tag>false</tag>-->
```

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

Let's start checking all the tags available inside the tag "basic". It contains all the basic configuration for your animation.

- **screenCanvasResolution**: It's the size of your canvas html object located on your website. Use "0" for default size or add values for resizing it. If you are planning use your animation on your Android phone, it needs to be the screen resolution.

  Note: *screenPhoneResolution* and *screenCanvasResolution* are the same. The app will read them in the same way, is just for avoid confusions between Android and HTML 5.

- **canvasResolution**: It's the resolution for your project. The app will stretch your animation until fit the canvas html object always keeping the aspect ratio.

- **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.

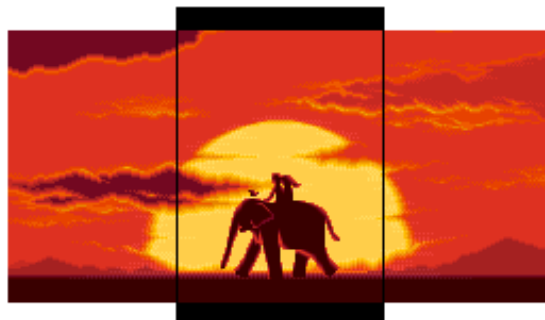 Example: `#ff63aeff` `(#Alpha-Red-Green-Blue)`
  Alpha is used for transparency levels where "ff" means no transparent color. We recommend learn more about this on internet.

- **antiAlias**: If you are stretching your images maybe this option needs to be set to true. It will help to improve the quality. Please check page for more information.

```
<basic type="animations">
  <screenCanvasResolution>
    <width>320</width>
    <height>480</height>
  </screenCanvasResolution>
  <canvasResolution>
    <maxWidth>256</maxWidth>
    <maxHeight>128</maxHeight>
  </canvasResolution>
  <bgColor>#ff000000</bgColor>
</basic>
```

- **grid:** If you are creating a pixel art animation this can be handy, 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:** This is a cool effect. 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 page for more information.

  - o **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>
      ```

  - o **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>
      ```

o **forceLandscape**: This tag is only for the Android app, it forces the animation to be always in landscape orientation.

```
<forceLandscape>true</forceLandscape>
```

o **forcePortrait**: This tag is only for the Android app, it forces the animation to be always in portrait orientation.

```
<forcePortrait>true</forcePortrait>
```

o **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>
```

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

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

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

```xml
<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 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".

- **postEffects:** Please go to page. These effects can be disabled with the tag "disable" or they can be enable or disabled with the actions "EnablePostEffects" & "DisablePostEffects".

Example:

```xml
<basic type="animations">
    <screenCanvasResolution>
        <width>256</width>
        <height>224</height>
    </screenCanvasResolution>
    <canvasResolution>
        <maxWidth>256</maxWidth>
        <maxHeight>224</maxHeight>
    </canvasResolution>
    <bgColor>#00000000</bgColor>
    <postEffects>
        <disable>true</disable>
        <effect>
            <effectName>ice</effectName>
            <iceColors>
                    239-239-255,90-89-255,140-142-255,66-65-255,132-134-255,197-
                    198-255,173-174-255,214-215-255,255-255-255
            </iceColors>
            <colorsToFreeze>
                    74-148-0,181-33-181,148-0-148,148-0-0,222-74-222,148-222-
                    74,148-148-181,181-148-0,181-74-0,74-74-107,255-0-0,181-181-
                    222,107-107-148,255-222-0,222-148-0,222-222-255,255-255-
                    255,222-181-0,107-222-255,33-33-74,33-107-0,107-181-33,0-0-0
            </colorsToFreeze>
        </effect>
    </postEffects>
</basic>
```

- **events**: You can use the tag events here and trigger actions. For example:

```
<events>
  <event>
    <eventType>OnLoad</eventType>
    <eventActions>
      <eventAction>BlockUserInterface</eventAction>
    </eventActions>
  </event>
</events>
```

After the first frame is printed, the event "OnLoad" will be triggered.

- **fps**: Change the default frame rate (20fps). In JavaScript you have the property "fpsToUse", this tag will override the property value. In Android you can add the attribute "showfps" to show on screen the current frame rate. Example:
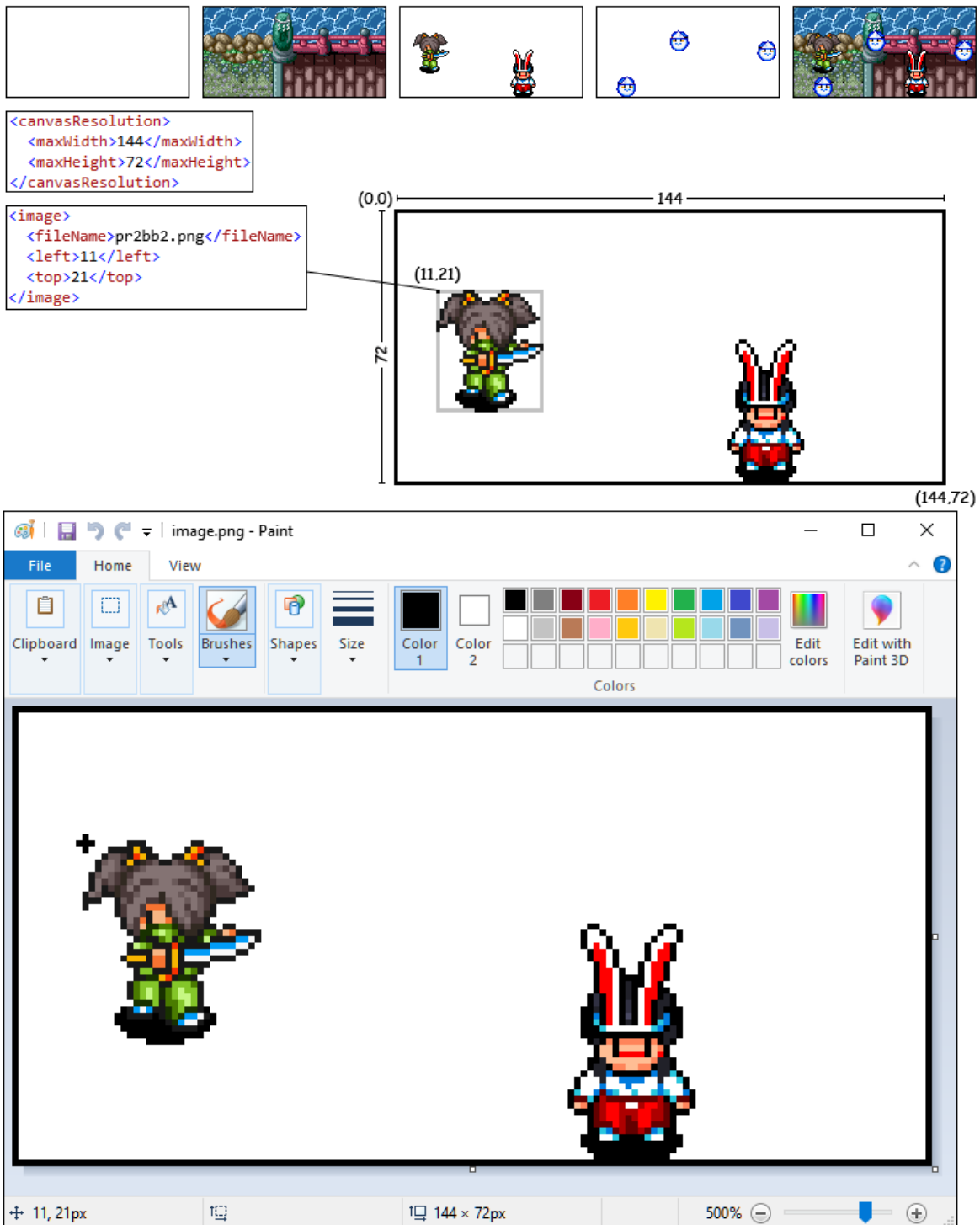
```
<fps showfps="true">30</fps>
```

In JavaScript check page to learn how to show the current frame in your animation.

- **keyMapper:** It is only for Android because in JavaScript we have the function "getRealKeyName" (see page). This tag solves the problem when you are capturing key presses and the key name is changed.

  - o **key:** A new key value.
    - **id:** The integer returned by Android after pressing a key (check on Internet to know all the id values, each key as a different id).
    - **code:** This needs to be the same always in JavaScript or Android and is used in the tag "whenKeyNameIs" after the events "OnKeyDown" or "OnKeyUp" are triggered.

```
<keyMapper>
  <key id="19" code="ArrowUp" />
  <key id="20" code="ArrowDown" />
  <key id="21" code="ArrowLeft" />
  <key id="22" code="ArrowRight" />
  <key id="66" code="Enter" />
  <key id="67" code="Backspace" />
</keyMapper>
```

Images

Adding images is easy and can be done using a text editor (an image editor comes handy too).


```xml
<canvasResolution>
  <maxWidth>144</maxWidth>
  <maxHeight>72</maxHeight>
</canvasResolution>
```

```xml
<image>
  <fileName>pr2bb2.png</fileName>
  <left>11</left>
  <top>21</top>
</image>
```

- **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>
  ```

- **canvasName**: It uses an animation instead of an image. You can include other animations inside the current one just adding the IDs of the HTML Canvas elements.

  ```
  <canvasName>IDCanvas</canvasName>
  ```

  Remember that you can hide the others canvas using CSS if you want only show the main animation.

- **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
<?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.

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

- o **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

  ```xml
  <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>
  ```

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

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

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

- o **createFramesBasedOnPostEffect:** If you are planning to use post effects as image alterations, create frame per frame can take a lot of lines. This tag can create all the frames for you. Keep in mind that not all the post effects are compatible with this tag. Check this [page](#) to get more information about post effects and know which ones support this tag.

```
<images>
    <image>
        <frames>
            <createFramesBasedOnPostEffect>
                <fileName>noiseb.png</fileName>
                <postEffects>
                    <effect>
                        <effectName>CreateNoise</effectName>
                        <replaceColors>21-95-217</replaceColors>
                        <randomColors>255-255-255,0-0-255,255-255-0,0-0-0</randomColors>
                        <numberOfFrames>10</numberOfFrames>
                    </effect>
                </postEffects>
            </createFramesBasedOnPostEffect>
        </frames>
        <left>0</left>
        <top>0</top>
    </image>
</images>
```

- **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. Example:

```xml
<image>
  <frames>
    <frameCollection>
      <frame>
        <fileName>ene1f.png</fileName>
        <delay>5</delay>
      </frame>
    </frameCollection>
    <frameCollection>
      <frame>
        <fileName>ene1l.png</fileName>
        <delay>5</delay>
      </frame>
    </frameCollection>
    <frameCollection>
      <frame>
        <fileName>ene1s.png</fileName>
        <delay>5</delay>
      </frame>
    </frameCollection>
    <frameCollection>
      <frame>
        <fileName>ene1n.png</fileName>
      </frame>
    </frameCollection>
    <frameCollection>
      <frame>
        <fileName>ene1p.png</fileName>
      </frame>
      <frame>
        <fileName>ene1q.png</fileName>
      </frame>
    </frameCollection>
    <framesAnimationSpeed>3</framesAnimationSpeed>
  </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>
```
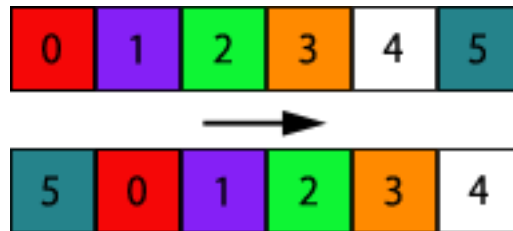

Here we have an example of color cycling animation. The colors are moved forward.

- **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".

```xml
<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>
```
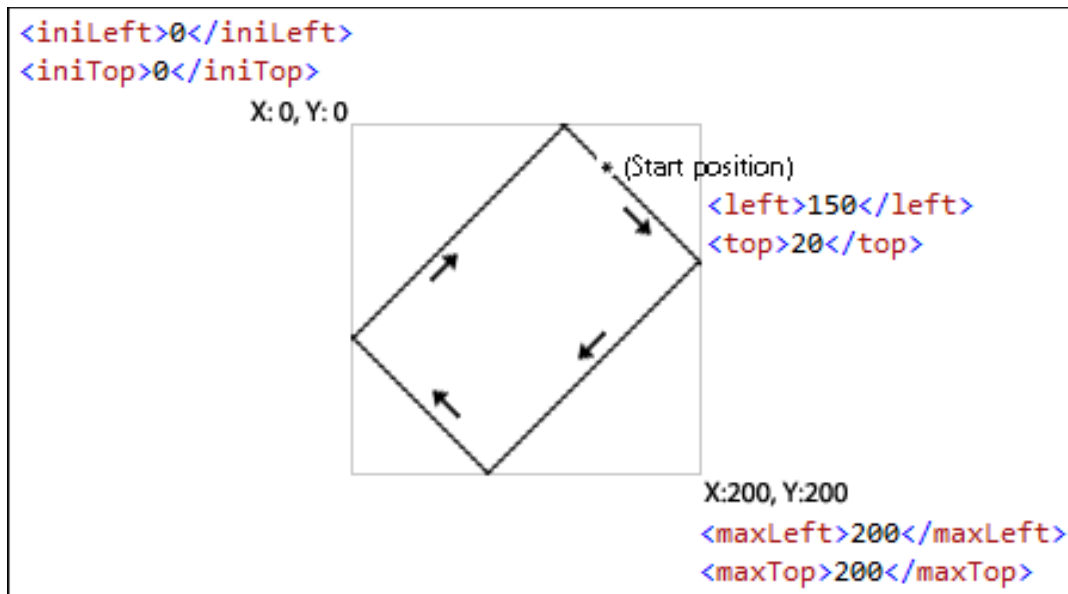

- **alpha**: Change the level of transparency in an image. The values are between 0 and 255, where 0 is completely transparent and 255 is the opposite.

- **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 to 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 understanding the basics of moveAnimation.
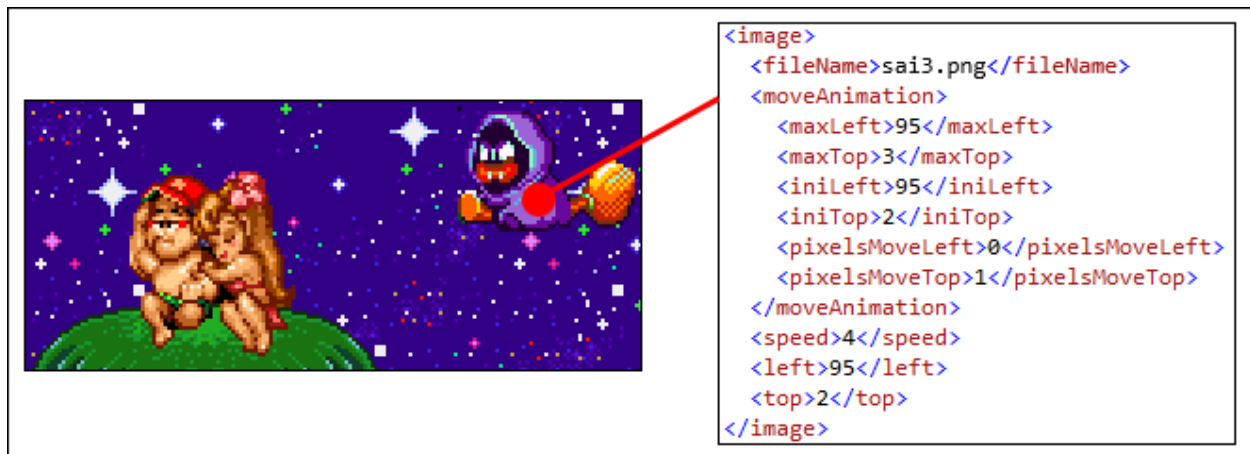
```
<iniLeft>0</iniLeft>
<iniTop>0</iniTop>
```

X: 0, Y: 0

*. (Start position)
```
<left>150</left>
<top>20</top>
```

X:200, Y:200
```
<maxLeft>200</maxLeft>
<maxTop>200</maxTop>
```

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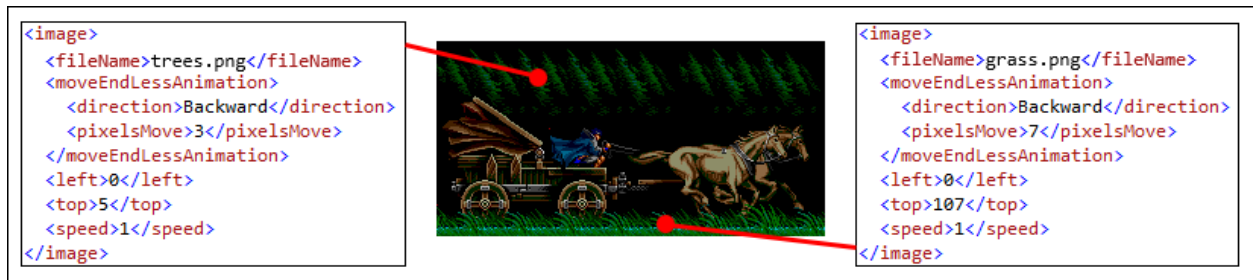
Playing with "pixelsMoveLeft" and "pixelsMoveTop" values change the direction of the image. Use negative values for **backward** movements.

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

```
<image>
  <fileName>sai3.png</fileName>
  <moveAnimation>
    <maxLeft>95</maxLeft>
    <maxTop>3</maxTop>
    <iniLeft>95</iniLeft>
    <iniTop>2</iniTop>
    <pixelsMoveLeft>0</pixelsMoveLeft>
    <pixelsMoveTop>1</pixelsMoveTop>
  </moveAnimation>
  <speed>4</speed>
  <left>95</left>
  <top>2</top>
</image>
```

Here "pixelsMoveLeft" was changed to 0 for vertical movements.

- **moveEndLessAnimation**: Create 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.

24

- o **numberOfImagesPerLayer**: You can repeat the image pattern vertically how many times is required.

- o **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.

- o **repeatLoop**: By the default is true but if you design require it, change it to false and after the loop is finished will stop.

- o **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.

- o **pathEndLessAnimationSpeed**: Like in previous speed tags, you can reduce or increase the speed using this tag.

- o **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:** If you are working with a camera and scrolling you can create multiple layers with this tag for 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:** Declare the size for the alternative canvas.

```xml
<?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>
  </images>
</doc>
```

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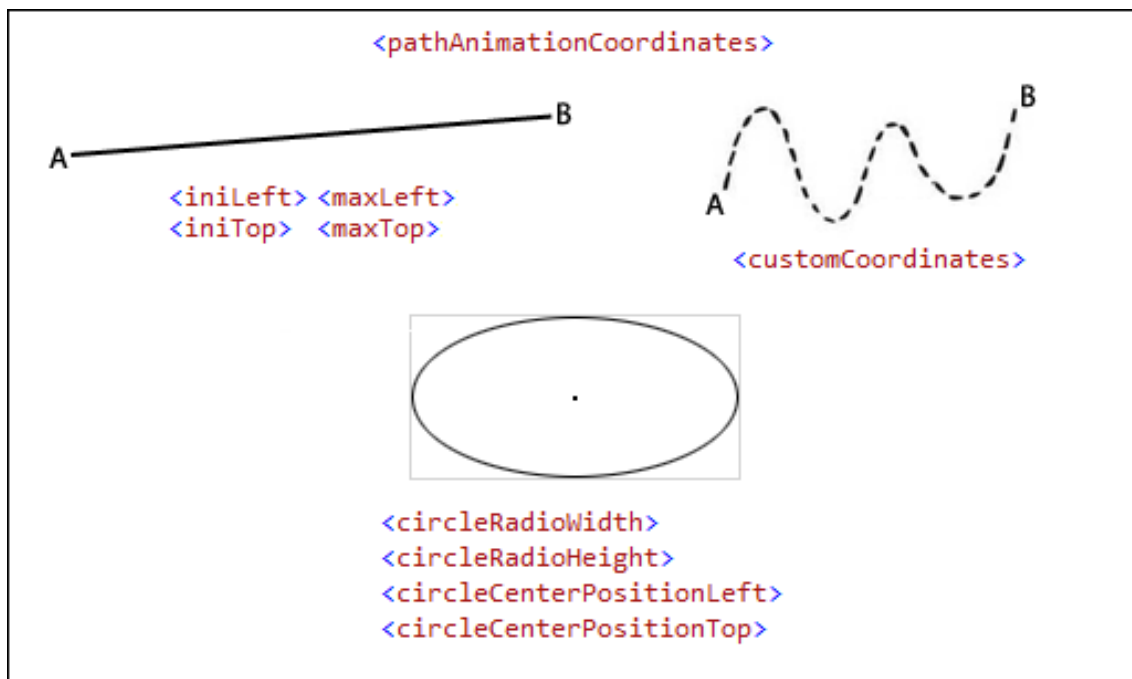
  - **pixelsMove**: The image pattern is moved following the quantity of pixels declared in this tag. Higher values create a feeling of speed, an image with a value 1 will move slower than another one with 4. Use negative values for backward movements.

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

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

- o **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".

- o **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.

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

- **containerIDToAttach**: Attack you image to a container. See page.

- **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.
        - **Perspective**: Change the image perspective.

      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 "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.

- **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 the image vertically or horizontally. For fix this problem use "Image alterations", check its section for more information.

Please check the next page, there we are playing with the properties alpha, width and height. Remember to check the Event tags section for learning 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>
```

- **onFocus**: If you are planning to track focus in your animation, use this tag to assign the default focus. When the animation starts the focus will be moved here automatically.

Video

The tag is included inside the tag "images" like a normal tag "image".

Image files are not the only one available source for animation frames, using this tag you can include videos. Keep in mind that image manipulations are limited on videos.

- **playerID**: It is quite important include a player id, it's the only way to manipulate the video from other animations. The video tag belongs to the image tag, it means that the player id and the image id are the same. If you have included a player id, you do not need to include an image id.

- **videoFile**: The file name of your video. Remember use the complete file name, including the extension, "MyVideo.mp4". The attributes available are:

  o **from (optional)**: You can use the attribute "from" when the video is in another folder or if you just want borrow videos from other animations.

  o **streamed**: By default is true for HTML and false for Android. Streamed videos is a nice way to speed up loading times especially if the video file is quite heavy. Please check here for more information about it. Changing this option to false, the video will be fully downloaded before play it.

- **forceType (HTML only)**: The application automatically get the file type reading the values submitted in the tag "videoFile", but sometimes if you are using videos from Internet (http, https) the url can be confusing making impossible know the file type. In these cases, you should use this tag and include manually the file type. Please check on Internet for more information. For .mp4 files use: "video/mp4".

- **autoPlay**: By default is true, after load the video it will play immediately.

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

- **currentTime**: Allow play a video from a specific position in time. The values are based in seconds. Check note about this tag on Android.

  - **milliseconds**: By default is false. If you need more precision, change it to true.

    ```
    <currentTime milliseconds="true">20987</currentTime>
    ```

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

- **useVideoFromPlayerID**: You can re-use a video in multiples "image tags" adding the player id in this tag. It comes handy if you are planning print the same video in different places in your canvas with different effects. Keep in mind that all the videos will be playing the same video and currentTime.

- **bringPlayerOnTop (Android only)**: It will send the player above everything (Z-Order). Default is true.

Please check these examples using the tag video:

```
<image>
  <video>
    <videoFile streamed="false" from="videoexample">Video.mp4</videoFile>
    <autoPlay>true</autoPlay>
    <volume>0.4</volume>
    <currentTime>90</currentTime>
    <playerID>videoPlayerBeta</playerID>
    <loop>false</loop>
  </video>
  <left>0</left>
  <top>0</top>
  <width>520</width>
  <height>320</height>
</image>

<image>
  <video>
    <useVideoFromPlayerID>videoPlayerBeta</useVideoFromPlayerID>
  </video>
  <left>0</left>
  <top>322</top>
  <width>420</width>
  <height>220</height>
  <id>Im a copy</id>
</image>

<image>
  <fileName>play.png</fileName>
  <left>300</left>
  <top>322</top>
  <id>PlayButton</id>
  <events>
    <event>
      <eventType>OnClickDown</eventType>
      <eventActions>
        <eventAction>PlayVideo</eventAction>
      </eventActions>
      <video>
        <playerID>videoPlayerBeta</playerID>
      </video>
    </event>
  </events>
</image>

<image>
  <video>
    <videoFile>Video.mp4</videoFile>
    <bringPlayerOnTop>true</bringPlayerOnTop>
    <autoPlay>false</autoPlay>
    <volume>0.2</volume>
    <currentTime milliseconds="true">20987</currentTime>
    <playerID>videoPlayerBeta</playerID>
  </video>
  <containerIDToAttach>containerA</containerIDToAttach>
  <left>4</left>
  <top>4</top>
  <width>320</width>
  <height>197</height>
  <moveAnimation>
    <iniTop>0</iniTop>
    <maxTop>44</maxTop>
    <pixelsMoveTop>2</pixelsMoveTop>
  </moveAnimation>
</image>
```

```xml
<image>
  <fileName>bringPlayer.png</fileName>
  <left>50</left>
  <top>200</top>
  <events>
    <event>
      <eventType>OnClickDown</eventType>
      <eventActions>
        <eventAction>UpdateVideoPlayer</eventAction>
      </eventActions>
      <video>
        <playerID>videoPlayerBeta</playerID>
        <bringPlayerOnTop>true</bringPlayerOnTop>
      </video>
    </event>
  </events>
</image>
```

Note: Video is available in Android and HTML. However, they behave differently. In HTML is treated like an image where you can apply image manipulations like "rotation" but in Android, the video is treated as a new layer above your project and the changes are limited to only "position" and "size".

Another challenger in Android is the order of how the images and videos are printed. It doesn't matter if in your XML file the video is first and the images later. The video will be printed above them all the time.

A way to fix this is using the tag "**bringPlayerOnTop**" and assigning it to false (default is true). It will send the player behind everything and using on your project as background a transparent color, will allow you to see the player. You can build projects like a video player, where the video is behind everything and the buttons Play, Stop, etc. are images above the video.

The tag "bringPlayerOnTop" can be changed any time through the action "UpdateVideoPlayer". You can bring the player above your animation and send it back based on your needs.

In Android, the tag "**currentTime**" can be less precise than HTML. The player in Android always tries to find the nearby "intraframe" causing a small difference compared to HTML. The values in the tag "currentTime" are seconds but changing the attribute "**milliseconds**" to true will allow you to use milliseconds instead of seconds.

```xml
<currentTime milliseconds="true">20987</currentTime>
```

Another important difference between Android and HTML is the attribute "**streamed**". In HTML the default value is true and in Android is false. When you are in HTML the app is hosted and online, playing streamed videos is the best option. Otherwise, the app will need to download the whole video before starting playing it.

However, on Android, the app is running on your phone. It is not online, the video can be played directly and that is the reason why the "streamed" attribute is false by default.

If your videos on Android are big and the size of your project is limited, you can upload your videos online and change the attribute "streamed" to true. The phone needs to be connected to the internet.

Playing videos from another host when you are in HTML is possible by adding the complete URL. Keep in mind that some browsers can give you CORS errors when you are trying to access to content that belongs to another host.

Streamed videos need to buffer enough data to start playing. Maybe during this time you want to show a loading icon, a message or even block your app to prevent the user press (or touch) something.

When videos start buffering, the event "**MediaBufferingStarts**" will be triggered and when it buffers enough data to play, it will trigger "**MediaBufferingEnoughToPlay**".

Check the actions "**BlockUserInterface**" and "**UnblockUserInterface**" if you want to block the app when it is buffering.

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:

```xml
<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 have used an image as button to play a sound.

```xml
<image>
  <fileName>icon.png</fileName>
  <left>250</left>
  <top>20</top>
  <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>
```

In this example the player "backgroundAudio" already have a sound loaded, we just need use the action "PlayAudio" and the "playerID" to play the sound.

```xml
<image>
  <fileName>icon2.png</fileName>
  <events>
    <event>
      <eventType>OnClickDown</eventType>
      <eventActions>
        <eventAction>PlayAudio</eventAction>
      </eventActions>
      <audioPlayer>
        <playerID>backgroundAudio</playerID>
      </audioPlayer>
    </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: Using custom fonts in HTML is easy, just add it on your web page and pick a name.

  Example:

  ```
  <style>
       @font-face {
           font-family: 'MyFont';
           src: url('styles/MyFont.ttf');
       }
   </style>
  ```

  Later using the tag "font" you can load your custom font.

  ```
  <font>MyFont</font>
  ```

  However, using custom fonts in Android is more complicated. 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".

    o **parameters (optional)**: You can send extra parameters to your JavaScript function.

    o **updateEveryFrame (optional)**: Assigning this parameter to true, the application will be checking if the text was changed every frame and will update it inside your animation.

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 parameters="0"
updateEveryFrame="true">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:

```xml
<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.
    - o **Id:** The name for you variable.
    - o **Value:** The default value.

Example:

```xml
<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.

```xml
<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:

```xml
<events>
  <event>
    <eventType>OnTouchUp</eventType>
    <eventActions>
      <eventAction>UpdateGlobalVariables</eventAction>
    </eventActions>
    <globalVariablesToUpdate>
      <variable>
        <id>screen</id>
        <value>0</value>
      </variable>
      <variable>
        <id>screen2</id>
        <value>7</value>
      </variable>
    </globalVariablesToUpdate>
  </event>
</events>
```

Timers

Timers work in the same way as the tag "imageTimer" works, the 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 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 is 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>
```

JavaScript actions

Trigger actions using events is a way to control objects in your project, but is not the only one. The tag "javaScriptActions" allows you trigger actions from JavaScript.

These are the necessary tags:

- **javaScriptAction**: Use how many actions you need, they are unlimited.

  - **Id**: The id will be used later from your JavaScript code.
  - **events**: You can include events here, please check the section "Event tags" for more information.

Example:

```xml
<javaScriptActions>
  <javaScriptAction>
    <id>Stop</id>
    <events>
      <event>
        <eventActions>
          <eventAction>StopAnimationAndMove</eventAction>
        </eventActions>
        <ids>image1,image2,image3</ids>
      </event>
    </events>
  </javaScriptAction>
  <javaScriptAction>
    <id>Resume</id>
    <events>
      <event>
        <eventActions>
          <eventAction>ResumeAnimationAndMove</eventAction>
        </eventActions>
        <ids>image4,image5,image6</ids>
      </event>
    </events>
  </javaScriptAction>
</javaScriptActions>
```

Calling the actions from JavaScript:

```javascript
function testRunJS() {
    jadsdsEngine.runAction("Stop");
    jadsdsEngine.runAction("Resume");
}
```

The method "runAction" runs the actions included inside the tag "javaScriptActions". Please check the section "Include the engine in your website" for more information.

Note: Please remember the action "RunJavascriptFunction", it allows you call JavaScript functions. Check page for more information.

Gestures

Capturing gestures is a common feature in these days, especially in smartphones. At the moment the engine captures the basic 8 movements:
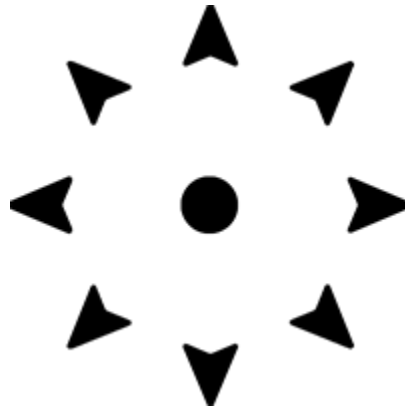
The events triggered by the gestures are: SlideRight, SlideLeft, SlideUp, SlideDown, SlideUpRight, SlideDownRight, SlideUpLeft, SlideDownLeft.

Example:
```xml
<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 mouse (or 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>
```
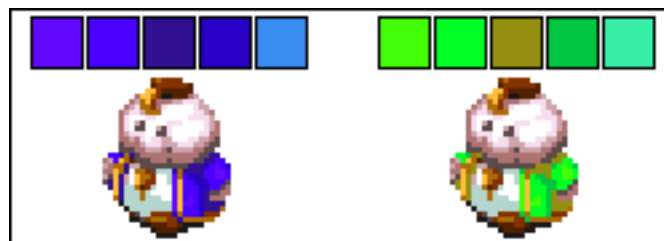

You don't need include all the colors, just those ones that you are planning to change.

Example:

```xml
<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>
```

Note: Remember you can use **post effects** inside the tag "applyAlterations", check [page](#).

Using post effects with image alterations

After reading the information available here, there are two important things to keep in mind. The post effects can be calculated when the animation is loading or frame by frame.

The attribute "frameByFrame" when is set to "true", the post effects are calculated frame by frame.

Example:

```
<applyAlterations>
    <postEffects>
        <effect>
            <effectName frameByFrame="true">ModifyColor</effectName>
            <modifyColorType>InvertColors</modifyColorType>
        </effect>
    </postEffects>
</applyAlterations>
```

By default, the post effects inside "applyAlterations" are calculated only when the app is loading. However, assigning the attribute "frameByFrame" to "true", will request to calculate them frame by frame like the ones inside the "basic" tag.

Note: Post effects frame by frame are very CPU intensive. Always try to make your animations with the attribute set to "false".

More examples:

```xml
<image>
   <frames>
      <frameCollection>
         <frame>
            <fileName>dcmap1.png</fileName>
            <applyAlterationsToThisFrame>
               <postEffects>
                  <effect>
                     <effectName frameByFrame="true">Melting</effectName>
                  </effect>
               </postEffects>
            </applyAlterationsToThisFrame>
         </frame>
      </frameCollection>
      <frameCollection>
         <frame>
            <fileName>dcmap1.png</fileName>
            <applyAlterationsToThisFrame>
               <postEffects>
                  <effect>
                     <effectName frameByFrame="false">ModifyColor</effectName>
                     <modifyColorType>GameBoy</modifyColorType>
                  </effect>
               </postEffects>
            </applyAlterationsToThisFrame>
         </frame>
         <frame>
            <fileName>dcmap1.png</fileName>
         </frame>
      </frameCollection>
      <defaultFrameCollection>0</defaultFrameCollection>
   </frames>
   <events>
      <event>
         <eventType>PostEffectMeltingFinished</eventType>
         <eventActions>
            <eventAction>ChangeFrameCollection</eventAction>
         </eventActions>
         <changeFrameCollectionIndexTo>1</changeFrameCollectionIndexTo>
      </event>
   </events>
   <speed>7</speed>
   <left>550</left>
   <top>20</top>
   <id>stopB</id>
</image>
```

Events, Actions and Conditionals tags

The app can trigger events when a specific situation happens.

```xml
<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.

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

Event Tag:

- **eventType**: Check the events table for more information. Is is possible include **multiples** "eventType" tags in one "event" tag. Example:

```
<event>
  <eventType>FrameAnimationReturnBackwardToFirstFrameEnds</eventType>
  <eventType>MoveAnimationMaxLeftReached</eventType>
  <eventType>MoveAnimationMaxRightReached</eventType>
```

- **eventAction**: Check the actions table for more information. It is possible to include **multiples** actions. Example:

```
<eventActions>
  <eventAction>StopMove</eventAction>
  <eventAction>ChangeFrameCollection</eventAction>
</eventActions>
```

- **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.

Example:

```xml
<events>
  <defaultCustomCondition>run</defaultCustomCondition>
  <event>
    <eventType>OnClickUp</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>OnClickUp</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 a quick way to run actions in multiple objects, but don't forget the tag "editAnimationByID".

  Please check the page 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.

```xml
<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:

```xml
<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>
```

- **Video tags:** Please check the [page](#) to learn all about the tags available. Remember use the action "UpdateVideoPlayer" for update the video properties.

  Examples:

```xml
<image>
  <fileName>stop.png</fileName>
  <left>0</left>
  <top>322</top>
  <events>
    <event>
      <eventType>OnClickDown</eventType>
      <eventActions>
        <eventAction>UpdateVideoPlayer</eventAction>
      </eventActions>
      <video>
        <playerID>videoPlayerBeta</playerID>
        <currentTime>90</currentTime>
        <volume>0.7</volume>
      </video>
    </event>
  </events>
</image>

<image>
  <fileName>stop.png</fileName>
  <left>40</left>
  <top>322</top>
  <events>
    <event>
      <eventType>OnClickDown</eventType>
      <eventActions>
        <eventAction>UpdateVideoPlayer</eventAction>
      </eventActions>
      <video>
        <videoFile streamed="false">Trailer.mp4</videoFile>
        <playerID>videoPlayerBeta</playerID>
        <currentTime>0</currentTime>
        <volume>0</volume>
      </video>
    </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".

    o **id**: The id of the object to edit. Remember use "," for multiples ids.
    o **editActions**: Here you can use all the available actions. Please check Actions table.
    o **Previous tags**: The Actions table not only says 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 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"

```
<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 to trigger an event only when a conditional match a value remember to 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 allow 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.

```xml
<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.

```xml
<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.

```xml
<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>
```

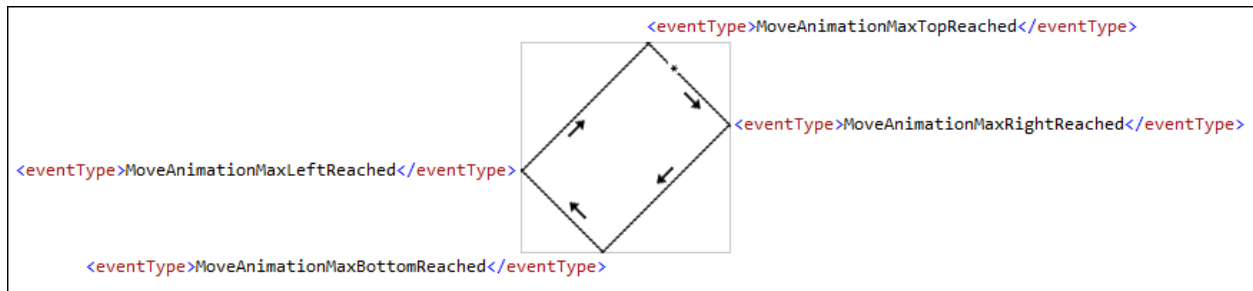| whenKeyNameIs |
|---|
| Use it if you are capturing the events OnKeyUp or OnKeyDown only when specific keys are pressed.

```xml
<event>
  <eventType>OnKeyUp</eventType>
  <eventActions>
    <eventAction>MoveFocus</eventAction>
  </eventActions>
  <whenFocused>true</whenFocused>
  <whenKeyNameIs>ArrowRight</whenKeyNameIs>
  <moveFocusTo>mainoption2</moveFocusTo>
</event>
```

The previous example is using the value "ArrowRight". These values are returned by the device when a key is pressed and should match with the values inside the tag. As you might guess, in Android and JavaScript they are different. Please check page (getRealKeyName property) for JavaScript and page (keyMapper tag) for Android. There you will find information about how to make your code work in both systems. |
| **whenFocused** |
| Use it when you want to trigger actions only when the focus is on the image.

```xml
<event>
  <eventType>OnKeyUp</eventType>
  <eventActions>
    <eventAction>MoveFocus</eventAction>
  </eventActions>
  <whenFocused>true</whenFocused>
  <whenKeyNameIs>ArrowRight</whenKeyNameIs>
  <moveFocusTo>mainoption2</moveFocusTo>
</event>
``` |

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.
FrameAnimationReturnBackwardToFirstFrameEnds
frameCollection when the tag "returnBackwardToFirstFrame" is completed.
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.
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.

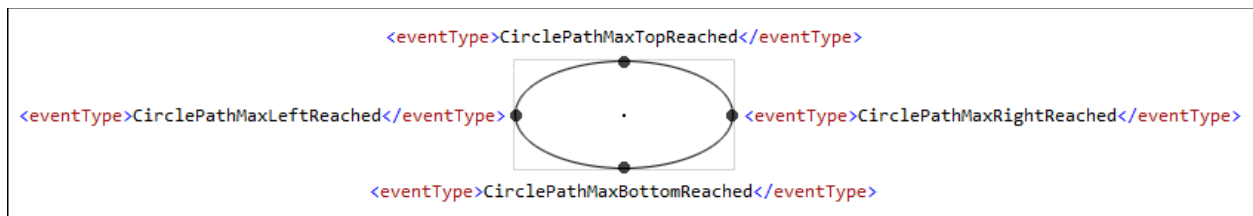| PlaybackIsFinished |
|---|
| It's triggered when an audio player finishes its sound. |
| **StartPrintingText** |
| It's triggered when a text area starts to print. |
| **PrintingNewLine** |
| It's triggered when a text area starts to print a new line. |
| **EndPrintingText** |
| It's triggered when 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 rotates and reach its minimum value. |
| **MaximumRotationValueReached** |
| It's triggered when the image rotates and reach its maximum value. |
| **RotationDegreesReached** |
| It's triggered when the image rotates 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. |
| **OnMouseOver** |
| Like in HTML the event is triggered when the mouse is over the image. This event is only for HTML. |
| **OnMouseOut** |
| Like in HTML the event is triggered when the mouse is out of the image area. This event is only for HTML. |
| **OnKeyDown** |
| It's triggered after a key in the keyboard is pressed. |
| **OnKeyUp** |
| It's triggered after a key in the keyboard is released. |

OnFocus
It's triggered when the focus is moved to the current image.
OnFocusOut
It's triggered when the focus is lost.
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.
MediaBufferingStarts
It's triggered when a streaming video starts Buffering.
MediaBufferingEnoughToPlay
It's triggered when a streaming video has enough data to start playing.
OnLoad
It's triggered from the "basic" tag when the first frame was printed.
NetworkFailure
It is triggered when a request is made and the Internet is not responding. Keep in mind that this event is triggered only when you are trying to play a video. If the internet suddenly fails, this event won't be triggered.
PostEffectMeltingFinished
It's triggered when the post effect "Melting" is finished. Only when the effect is used inside image alterations. Example here.

Example

The next image explains how moveAnimation events are triggered.

The next image explains how pathAnimation triggers events when "pathAnimationCoordinates" was used to create a circle path.

Remember circle paths have start and end points, it means that the events "PathAnimationStartPointReached" and "PathAnimationEndPointReached" will be also triggered.

Actions

After an event is triggered you can trigger actions. Please check the next list:

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
Allow change the animation properties in other images by id.
Depend on these Tags (and what you want to update): • **editAnimationByIDOptions**: Please check the page 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.

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.

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 for another example.

Depend on these Tags:
 	
 • **new + animation name**: Please check the next example.
 	
<pre>   <event>
      <eventType>FrameAnimationRepeatLoopEnds</eventType>
      <eventActions>
        <eventAction>UpdateMoveAnimation</eventAction>
      </eventActions>
      <newMoveAnimation>
        <iniLeft>2</iniLeft>
        <maxLeft>100</maxLeft>
        <pixelsMoveLeft>-2</pixelsMoveLeft>
        <pixelsMoveTop>0</pixelsMoveTop>
      </newMoveAnimation>
    </event></pre>
   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.

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
Change 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 player section and page for more information.

PlayVideo, StopVideo, PauseVideo, ResumeVideo, UpdateVideoPlayer
Allow you control video players in your animation. Please check the page and page 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.

```xml
<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" and "Decrease".

ChangeAnimationWidthAction
Change the current action used for width in imagePropertiesAnimation.

Depend on these Tags:

- **animationWidthAction**: You can choose between "Increase" and "Decrease".

ChangeAnimationHeightAction
Change the current action used for height in imagePropertiesAnimation.

Depend on these Tags:

- **animationHeightAction**: You can choose between "Increase" and "Decrease".

| **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. |

| **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. |

| **RunJavascriptFunction** |
| Allow you run JavaScript code (not available in Android). |
| Depend on these Tags:
• **runJavascriptFunction**: Type you code here. It is highly recommended create a function in JavaScript with all the code and just call it from here. |

```
<event>
 <eventType>MinimumRotationValueReached</eventType>
 <eventActions>
  <eventAction>RunJavascriptFunction</eventAction>
 </eventActions>
 <runJavascriptFunction>helloworld("Hello World");</runJavascriptFunction>
</event>
```

UpdateGlobalVariable

Update a global variable, please check the "Global variables" section for more information.

Please check the example:

- **globalVariablesToUpdate**: Add all the variables to update here.

```xml
<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 the mouse (or your finger).

Please check the example:

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

UpdateFPS

Allow you update the current frame rate.

Depend on these Tags:

- **fps**: The new frame rate value.

MoveFocusTo
Move the focus. You need to use the id as value.
Please check the example:

- **moveFocusTo**: Add the id where the will be moved.

```
<event>
  <eventType>OnKeyUp</eventType>
  <eventActions>
    <eventAction>MoveFocus</eventAction>
  </eventActions>
  <whenFocused>true</whenFocused>
  <whenKeyNameIs>ArrowRight</whenKeyNameIs>
  <moveFocusTo>mainoption2</moveFocusTo>
</event>
```

BlockUserInterface
The user interface is blocked, stopping mouse (touch) and keyboard interactions.
UnblockUserInterface
After the action "BlockUserInterface" is called, you can unblock the user interface.
EnablePostEffects
Enable post effects inside the tag basic.
DisablePostEffects
Disable post effects inside the tag basic.
ResumePostEffectsFrameByFrame
Enable post effects inside image alterations.
StopPostEffectsFrameByFrame
Disable post effects inside image alterations.
UpdateStringOfCoordinates
Edit coordinates done by the tag "coordinates" inside the tag "image".
Please check the example:

- **newStringOfCoordinates**: Type de new coordenates.

```
<images>
  <image>
    <fileName>g7.png</fileName>
    <left>60</left>
    <top>0</top>
    <events>
      <event>
        <eventType>OnClickUp</eventType>
        <eventActions>
          <eventAction>UpdateStringOfCoordinates</eventAction>
        </eventActions>
        <newStringOfCoordinates>60;60,80;80,100;100,120;120,140;140,160;160</newStringOfCoordinates>
        <ids>moveme</ids>
      </event>
    </events>
  </image>
  <image>
    <fileName>g6.png</fileName>
    <id>moveme</id>
    <coordinates>10;10,30;30,50;50,70;70,90;90,110;110</coordinates>
  </image>
</images>
```

Training

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

```xml
<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>
```
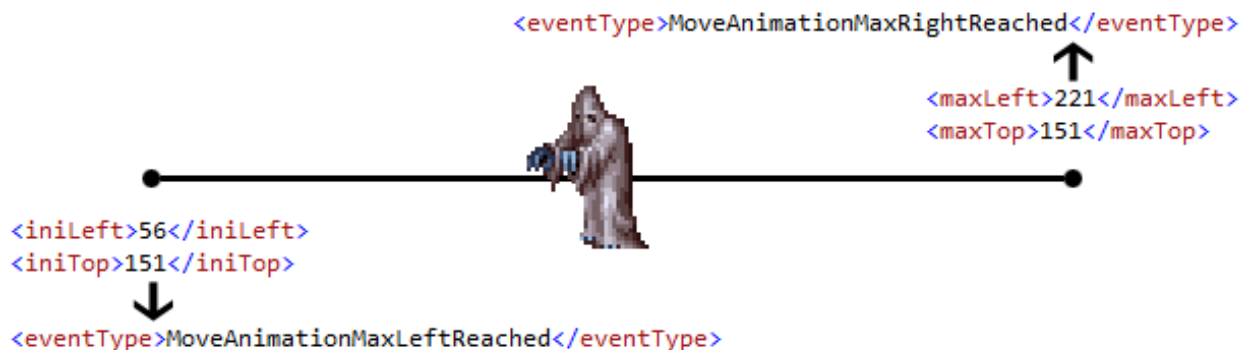

A good example is moveAnimation tag, when an image reaches the limits will trigger an event and you can add actions.
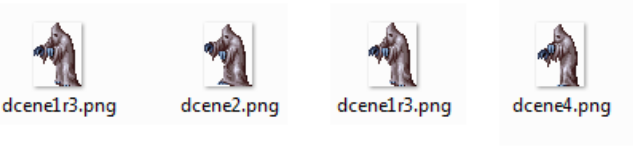
The most common action for moveAnimation is **FlipImage**. In the image above we have an animated image walking, after reach the limits the image returns but we need flip it before otherwise the image will look like walking backward.

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.
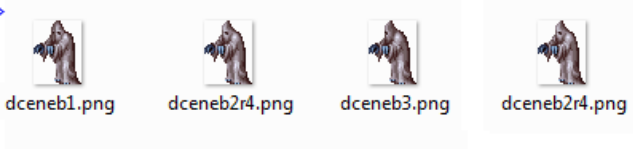
In these cases remember use frameCollection tags for multiples frames.

```xml
<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>
</image>
```

Frames for "walking"

dcene1r3.png   dcene2.png   dcene1r3.png   dcene4.png

Frames for "look around"

dceneb1.png   dceneb2r4.png   dceneb3.png   dceneb2r4.png

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

```xml
<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. 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"

```xml
<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.

Include the engine on your website

Before begin is recommended have at least a basic knowledge of JavaScript to make easier understand the next instructions.

```html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script src="JadsdsEngine_min.js"></script>
    <script type="text/javascript">
        window.onload = function () {
            var jadsdsEngine = new JadsdsEngine('mycanvas');
            //jadsdsEngine.fixCanvasSize = false;
            //jadsdsEngine.renderMode = RenderMode.Advanced; //RenderMode.Simple;
            //jadsdsEngine.antiAlias = true;
            //jadsdsEngine.systemFolder = "animations";
            //jadsdsEngine.fpsToUse = 20;
            //jadsdsEngine.loadingMessage = "Loading...";
            jadsdsEngine.loadAnimation("dc");
        };
    </script>
</head>
<body>
    <canvas id="mycanvas" width="320" height="480" style="border-style: solid; border-width: 1px"></canvas>
</body>
</html>
```

Above the method "loadAnimation()" loads the animation "dc" which is located inside the folder "animations" on your website.

Like you can see some lines of the code are commented, these values are optional. Play with them if you project requires it.

List of available properties:

- **fixCanvasSize**: The engine reads the tag "screenCanvasResolution" and changes the size of the canvas html object following the values submitted by the tag. If you don't want this change, set "fixCanvasSize" property to false.
- **renderMode**: By default the renderMode is "Advanced", If you are working with huge resolutions in an old PC consider change renderMode to "Simple", the quality is not the same but it can be useful to keep a good frame rate.
- **antiAlias**: If you are planning work with the property "fixCanvasSize" in false and "renderMode" in "Advanced", probably you are going to need to set this property to true to improve the image quality.
- **systemFolder**: By default, the folder where the animations are located is "animations", but if your website already has a folder with that name, you can change it by another one.

- **fpsToUse**: The frame rate by default used is 20, this is a nice way to keep a low use of CPU. Change it if your animations need a higher value. Check [page](#) to know about the tag "fps".
- **showFrameRate** & **frameCounterID**: If you want keep tracking the frame rate in your animation for debug purposes, you will need to set the property "showFrameRate" to "true" and assign the id of your "DIV" tag were the frame rate counter will be displayed.

```
jadsdsEngine.showFrameRate = true;
jadsdsEngine.frameCounterID = "d_debug";
```

- **loadingMessage**: If your website runs in a different language, with this property you can change the loading message by another one.
- **runFunctionAfterLoad**: Allow you run code after loading the animation. Check the next example:

```
jadsdsEngine.runFunctionAfterLoad = function () {
    alert("ok");
}
```

- **runAfterLoadAnimation**: All the animations after be loaded run immediately, you can set this property to false and the engine will be paused. Use the method "resume" for resume the animation.
- **stopAnimationWhenIsNotVisible**: If your project is bigger than the user visible area and is using multiples instances of the engine, probably you are going to require an intensive use of the CPU. You can set this property to true and the animation will pause itself when is not visible.
- **loadAnimationScreen**: If you animation is heavy and takes a long time to load, using an animation screen while the web browser is downloading the data is a nice way to keep the user waiting until the animation is fully loaded. Just include the name of your loading animation in this property.

    Note: The animation need to be simple with almost no images, the idea is something quick and easy to download, otherwise the user will see a black screen.
    Example:
    jadsdsEngine.loadAnimationScreen = "myLoadingScreen";

- **manipulateCanvas:** Just before printing a frame this property is called allowing you manipulate the canvas. It's useful if you are planning to add some effects in your animations. It only works when the property "renderMode" is "Advanced".

  Example:

  ```
  jadsdsEngine.manipulateCanvas = function grayscale(canvasContextToUse, canvasWidth, canvasHeight)
  {
      var r, g, b, brightness;
      var frameData = canvasContextToUse.getImageData(0, 0, canvasWidth, canvasHeight);
      var data = frameData.data;
      for (var i = 0; i < data.length; i += 4) {
          r = data[i];
          g = data[i + 1];
          b = data[i + 2];
          brightness = (r + g + b) / 3;
          data[i] = brightness;
          data[i + 1] = brightness;
          data[i + 2] = brightness;
      }
      frameData.data = data;
      canvasContextToUse.putImageData(frameData, 0, 0);
  }
  ```

  Note: The previous code adds a grayscale effect to the canvas. The engine sends you "canvasContextToUse", "canvasWidth" and "canvasHeight". These variables are based on the current canvas used by the engine.

- **functionToUpdateText:** If you animation includes text updated dynamically, a function needs to be assigned to this property (read more about the tag "textFromJavascript" on page).

  You can include 2 parameters in you JavaScript function. The next example has "textID" and "parameters". The first one is mandatory and the second one is optional.

  Also you can create dynamic variables (in this case is "myVariable") and if your animation has the parameter "updateEveryFrame" assigned to true, the text inside your animation will be updated every time when your dynamic variables change their values.

  Example:

  ```
  jadsdsEngine.functionToUpdateText = function (textID , parameters) {
      var valueToReturn = "";
      switch (textID) {
          case " textFromJavascript1":
              valueToReturn = "All#10;is#10;ok#10;!";
              break;
          case " textFromJavascript2":
              valueToReturn = "Good! " + parameters + myVariable;
              break
      }

      return valueToReturn;
  }
  ```

- **getRealKeyName:** If you are capturing key press events (OnKeyDown and OnKeyUp) probably you will need this. By default, you get the default key names. However, in Android and HTML these key names are different. If you have problems with the default values, adding your own method can solve the situation.

Example:

```
jadsdsEngine.getRealKeyName = function (event) {
        return event.key;
}
```

- **getImageData**: If you have worked with HTML canvas before, you know about "getImageData". It returns an array with all the pixels in rgba format.

This method will do the same based on your current animation. Combining animations and using the method "manipulateCanvas", you can create interesting effects.

The next example is using 3 animations. One of them is used as a pattern to print another animation on it.

The pattern animation only has 2 colors, black and white. Adding the necessary code inside the method "manipulateCanvas", you can only fill the black areas creating a nice effect.

```
function loadSubCanvas() {
    var subBJadsdsEngine = new JadsdsEngine('mysubcanvasB');
    var subJadsdsEngine = new JadsdsEngine('mysubcanvas');
    subJadsdsEngine.loadAnimation("zZora"); //Animation 2 (subJadsdsEngine)
    subJadsdsEngine.runFunctionAfterLoad = function () {
        mainAni(); //Animation 1 (your main animation - jadsdsEngine)
        jadsdsEngine.runFunctionAfterLoad = function () {
            subBJadsdsEngine.loadAnimation("pattern"); //Animation 3 (pattern - subBJadsdsEngine)
            subBJadsdsEngine.runFunctionAfterLoad = function () {

                jadsdsEngine.manipulateCanvas = function takeFromAnotherCanvas(canvasContextToUse, canvasWidth,
canvasHeight) {
                    var frameData = canvasContextToUse.getImageData(0, 0, canvasWidth, canvasHeight);
                    var data = frameData.data;
                    var dataSub = subJadsdsEngine.getImageData(0, 0, canvasWidth, canvasHeight);
                    var dataSubB = subBJadsdsEngine.getImageData(0, 0, canvasWidth, canvasHeight);

                    for (var i = 0; i < data.length; i += 4) {
                        if (dataSubB[i] == 0 && dataSubB[i + 1] == 0 && dataSubB[i + 2] == 0) { //000 black
                            data[i] = dataSub[i];
                            data[i + 1] = dataSub[i + 1];
                            data[i + 2] = dataSub[i + 2];
                        }
                    }

                    canvasContextToUse.putImageData(frameData, 0, 0);
                }
            }
        }
    }
}
```
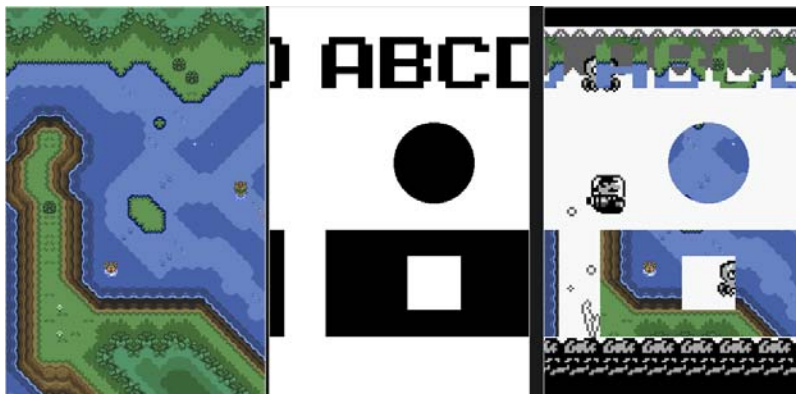

Animation 2 (to be printed) – Animation 3 (pattern) – Animation 1 (main)

List of available methods:

- **stop**: Stop the animation. Example: jadsdsEngine.stop();
- **resume**: Resume the animation. Example: jadsdsEngine.resume();
- **runAction**: Trigger the actions included inside the tag "javaScriptActions". The parameter is the javaScriptAction id. Check the "JavaScript actions" section for more information.

```
function testRunJS() {
    jadsdsEngine.runAction("Stop");
}
```

- **unloadAnimation**: If you are planning change the animation, remember always use this method before call "loadAnimation".
- **loadAnimation**: Allow you load an animation. The parameter is the animation name.

  Example:

```
function changeAnimation(animationName) {
    jadsdsEngine.unloadAnimation();
    jadsdsEngine.loadAnimation(animationName);
}
```

Post effects

This tag adds effects to the canvas after printing an animation frame or directly on the image. The examples below will help you to understand how powerful this tag could be. The effects can be used in Android and HTML.

At the moment these are the effects available:

1) **Mode7snes**: A nice mode 7 effect well remembered by Super NES fans.
2) **ModifyColor:** Allow you to change the colors. These are the available options:
   o **GameBoy:** This one will make your animations look like a classic Game Boy screen.
   o **Brightness:** Adjust the brightness.
   o **Contrast:** Adjust the contrast.
   o **InvertColors**: Invert the colors.
   o **GrayScale:** Apply grayscale.
   o **ReplaceColors:** It works in the same way as "ApplyNewPaletteColor" found in image alterations.
3) **ApplyNight:** It creates a nice transition night effect.
4) **CreateNoise**: The same effect that happens in old TVs when they didn't have a signal.
5) **Ice**: It changes the colors making it look frozen.
6) **Melting**: Create a melting effect.

It has to be included inside the tag "basic" or can be used on image alterations.

- **postEffects**: The main tag.
  o **disable**: The animation starts with the post effects enabled by default. You can disable them using this tag. Remember, the actions "EnablePostEffects" and "DisablePostEffects" allow you to disable or enable the effects. Only for post effects inside the "basic" tag. Example here.

  o **effect**: The effect to use (you can include as many you want).
    ▪ **effectName**: The name of the effect. You can use the attribute "frameByFrame" if you are planning using post effects on image alterations. Click for more information.
    ▪ **setting tags**: Some effects will need additional settings.

  o **stopPostEffectsFrameByFrame**: Similar to the previous tag "disable", it enables or disables the effects when the attribute "frameByFrame" is set to "true" inside image alterations. You can also use the actions "StopPostEffectsFrameByFrame" & "ResumePostEffectsFrameByFrame". Example here.

Remember that you can use the tag "canvasName", this tag allows you to print another animated canvas inside your canvas. For example, if you are planning to create an F-Zero animation, the track can be done in a different canvas then apply the post effect "mode7snes" and print it inside your canvas adding the cars, backgrounds, etc.

Note: At the moment, only "mode7snes" is not available on Android due to it works using two canvas.

Note: Post effects are very CPU intensive. They are calculated frame by frame all the time. If you are on Android and you want to save battery, remember you can use post effects inside the tag "applyAlterations", check page. Post effects inside "applyAlterations" are calculated only when the app is loading.

Before starting with the examples let's see how the tag can be included in your animations.

Inside the "basic" tags:

```
<basic type="animations">
  <screenCanvasResolution>
    <width>256</width>
    <height>1280</height>
  </screenCanvasResolution>
  <canvasResolution>
    <maxWidth>256</maxWidth>
    <maxHeight>1280</maxHeight>
  </canvasResolution>
  <bgColor>#00ffffff</bgColor>
  <postEffects>
    <effect>
      <effectName>mode7snes</effectName>
      <profile>ffvi</profile>
    </effect>
    <effect>
      <effectName>gameBoy</effectName>
    </effect>
  </postEffects>
</basic>
```

Inside the "image" ([image alterations](#)) tag:

```
<image>
  <fileName>house.png</fileName>
  <applyAlterations>
    <postEffects>
      <effect>
        <effectName>modifycolor</effectName>
        <modifyColorType>gameboy</modifyColorType>
      </effect>
    </postEffects>
  </applyAlterations>
</image>
```

Note: If you are planning to use post effects as image alterations, create frame per frame can take you a lot of time. The tag "createFramesBasedOnPostEffect" can create all the frames for you. Keep in mind that not all the post effects are compatible with this tag, check page for more information. These are the post effects where "**createFramesBasedOnPostEffect**" can be used:

- Ice
- CreateNoise
- Melting

You can add multiple post effects but only the first one will be used for "createFramesBasedOnPostEffect"

Example:

```
<frameCollection>
  <createFramesBasedOnPostEffect>
    <fileName>ryu1.png</fileName>
    <postEffects>
      <effect>
        <effectName>Melting</effectName>
      </effect>
      <effect>
        <effectName>ModifyColor</effectName>
        <modifyColorType>ReplaceColors</modifyColorType>
        <colorList>176-96-80,176-160-128,192-96-0,208-96-0</colorList>
        <colorListToReplace>16-0-0,49-0-0,82-24-0,107-49-0</colorListToReplace>
      </effect>
    </postEffects>
  </createFramesBasedOnPostEffect>
  <repeatThisLoopForTimes>1</repeatThisLoopForTimes>
</frameCollection>
```

Disable the effects inside the tag [basic](#):

```
<basic type="animations">

    <screenCanvasResolution>
        <width>256</width>
        <height>224</height>
    </screenCanvasResolution>
    <canvasResolution>
        <maxWidth>256</maxWidth>
        <maxHeight>224</maxHeight>
    </canvasResolution>
    <bgColor>#00000000</bgColor>
    <postEffects>
        <disable>true</disable>
        <effect>
            <effectName>ice</effectName>
            <iceColors>
                    239-239-255,90-89-255,140-142-255,66-65-255,132-134-255,197-
                    198-255,173-174-255,214-215-255,255-255-255
            </iceColors>
            <colorsToFreeze>
                    74-148-0,181-33-181,148-0-148,148-0-0,222-74-222,148-222-
                    74,148-148-181,181-148-0,181-74-0,74-74-107,255-0-0,181-181-
                    222,107-107-148,255-222-0,222-148-0,222-222-255,255-255-
                    255,222-181-0,107-222-255,33-33-74,33-107-0,107-181-33,0-0-0
            </colorsToFreeze>
        </effect>
    </postEffects>
</basic>
```

It can be enabled using [actions](#) "EnablePostEffects" & "DisablePostEffects".

Disable the effects inside [image alterations](#)

```
<image>
    <fileName>ice.png</fileName>
    <applyAlterations>
        <postEffects>
            <stopPostEffectsFrameByFrame>true</stopPostEffectsFrameByFrame>
            <effect>
                <effectName frameByFrame="true">ice</effectName>
                <iceColors>
                    173-174-255,214-215-255,255-255-255
                </iceColors>
                <colorsToFreeze>
                    222-222-255,107-222-255,33-33-74
                </colorsToFreeze>
            </effect>
        </postEffects>
    </applyAlterations>
    <id>ice</id>
</image>
```

It can be enabled using [actions](#) "StopPostEffectsFrameByFrame" & "ResumePostEffectsFrameByFrame".

Mode7snes

It is a nice mode 7 effect well remembered by Super NES fans.

Configuration tags:

Some effects need settings before being used.

- o **profile**: You have two profiles.
  - fzero: It makes the canvas look like the tracks in F-Zero.
  - ffvi: This one looks like the Final Fantasy VI intro that everybody loves.

Note: This post effect is only for HTML.

Example:

Here we are going to make a simple animation that looks like the racing game F Zero for SNES. Previously we explained that it is possible to use the the tag "canvasName" and that is exactly what will be done here, one canvas for the track and another one for everything else.
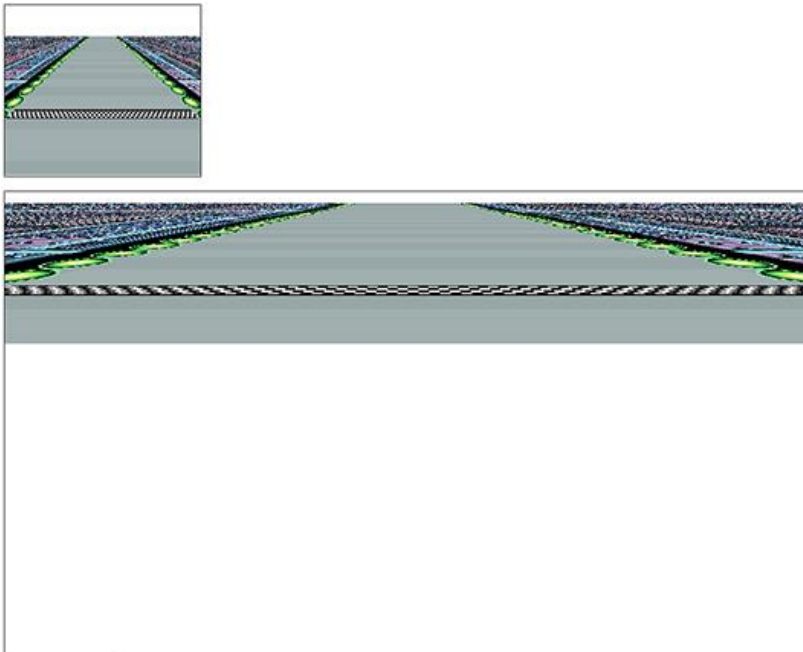
We have two canvas, the idea is to print one of them inside the other one.

Main canvas:

```xml
<?xml version="1.0"?>
<doc>
  <basic type="animations">
    <screenCanvasResolution>
      <width>256</width>
      <height>224</height>
    </screenCanvasResolution>
    <canvasResolution>
      <maxWidth>256</maxWidth>
      <maxHeight>224</maxHeight>
    </canvasResolution>
    <bgColor>#ffffffff</bgColor>
  </basic>
  <images>
    <image>
      <canvasName>mysubcanvas</canvasName>
      <left>0</left>
      <top>26</top>
      <width>256</width>
      <height>600</height>
    </image>
  </images>
</doc>
```

The main canvas is your animation and everything will be printed.

Sub-canvas (mysubcanvas):

```xml
<?xml version="1.0"?>
<doc>
  <basic type="animations">
    <screenCanvasResolution>
      <width>1054</width>
      <height>608</height>
    </screenCanvasResolution>
    <canvasResolution>
      <maxWidth>1054</maxWidth>
      <maxHeight>608</maxHeight>
    </canvasResolution>
    <bgColor>#00ffffff</bgColor>
    <postEffects>
      <effect>
        <effectName>mode7snes</effectName>
        <profile>fzero</profile>
      </effect>
    </postEffects>
  </basic>
  <images>
    <image>
      <fileName>mode7.png</fileName>
      <moveEndLessAnimation>
        <direction>Down</direction>
        <pixelsMove>10</pixelsMove>
      </moveEndLessAnimation>
      <left>0</left>
      <top>0</top>
      <speed>1</speed>
    </image>
  </images>
</doc>
```

The sub canvas has the track and it will use the post effect "mode7snes - fzero" to make it looks like the SNES mode 7.

The main canvas is using the tag "canvasName", which allows you to print the sub-canvas inside, just use the same id used in your HTML code.

Note: Remember that you can use CSS to hide the sub-canvas.

Modifycolor

Allow you to change the colors. These tags are shared in all the available modifications.

- **onlyTheseColors**: The effect will affect only the selected colors. The pattern is "R-G-B" and uses "," for multiple colors. Example: 247-255-66,255-189-66.
- **avoidTheseColors**: The effect will avoid the selected colors. The pattern is "R-G-B" and uses "," for multiple colors. Example: 247-255-66,255-189-66.

These are the available changes:

- **Gameboy**: It will make your animations look like a classic Game Boy screen.

    Configuration tags: You don't need settings.

    Example:

    It changes the image making it look like the old Game Boy.

```
<postEffects>
  <effect>
    <effectName>modifycolor</effectName>
    <modifyColorType>gameboy</modifyColorType>
  </effect>
</postEffects>
```

- **Brightness**: Adjust the brightness.

  Configuration tags:
  - **brightnessLevel**: Choose between 0 to 100 (or -100 to 0 to reduce brightness).

  Example:

  ```xml
  <postEffects>
      <effect>
          <effectName>ModifyColor</effectName>
          <modifyColorType>Brightness</modifyColorType>
          <brightnessLevel>5</brightnessLevel>
          <avoidTheseColors>23-0-0</avoidTheseColors>
      </effect>
  </postEffects>
  ```
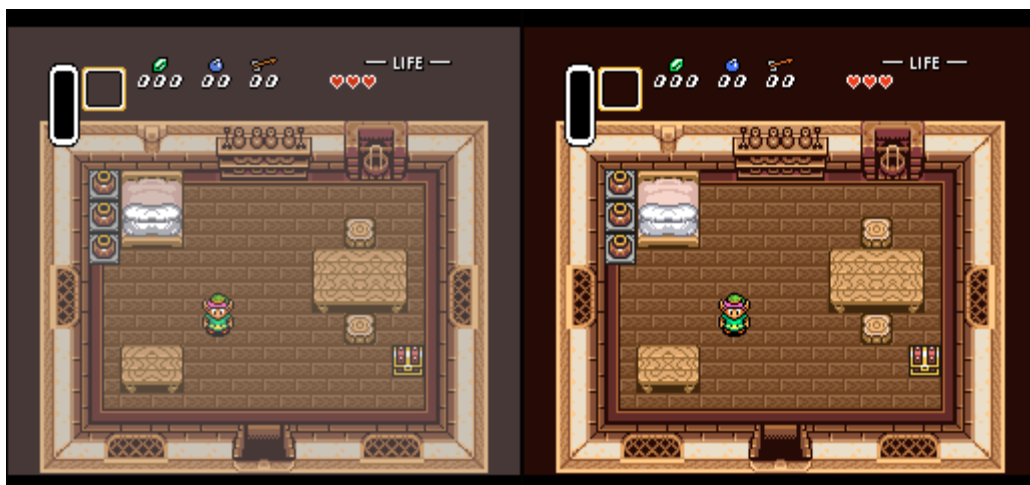
- **Contrast**: Adjust the contrast.

  Configuration tags:
  - **contrastLevel**: Choose between 0 to 100 (or -100 to 0 to reduce contrast).

  Example:

  ```xml
  <postEffects>
      <effect>
          <effectName>ModifyColor</effectName>
          <modifyColorType>Contrast</modifyColorType>
          <contrastLevel>50</contrastLevel>
          <onlyTheseColors>57-41-41,123-90-66</onlyTheseColors>
      </effect>
  </postEffects>
  ```

- **InvertColors**: It inverts the image colors.

  Configuration tags: You don't need settings.
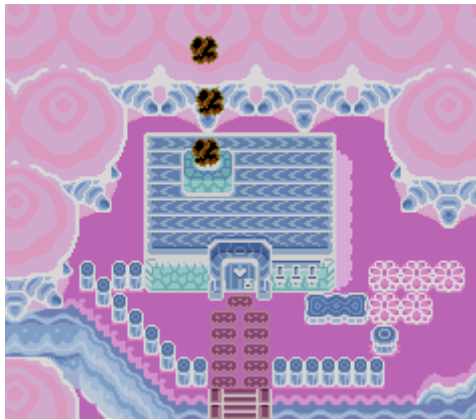
  Example:

  ```
  <postEffects>
    <effect>
      <effectName>ModifyColor</effectName>
      <modifyColorType>InvertColors</modifyColorType>
    </effect>
  </postEffects>
  ```

- **GrayScale**: It removes the color applying grayscale.

  Configuration tags: You don't need settings.

  Example:

  ```xml
  <postEffects>
    <effect>
      <effectName>ModifyColor</effectName>
      <modifyColorType>GrayScale</modifyColorType>
    </effect>
  </postEffects>
  ```

- **ReplaceColors**: It works in the same way as "ApplyNewPaletteColor" found in image alterations. Replacing colors was a common practice in old games. You can create all kinds of nice effects just by changing some colors. Like "ApplyNewPaletteColor" you need a list with the colors to be replaced and another list with the new colors. The pattern is "R-G-B" and uses "," for multiple colors. Example: 247-255-66,255-189-66.

   Configuration tags:
   - **colorList**: A list with the colors to be replaced.
   - **colorListToReplace**: The list with the new colors.

   Example:

   ```
   <postEffects>
     <effect>
       <effectName>ModifyColor</effectName>
       <modifyColorType>ReplaceColors</modifyColorType>
       <colorList>176-96-80,176-160-128, …</colorList>
       <colorListToReplace>16-0-0,49-0-0, …</colorListToReplace>
     </effect>
   </postEffects>
   ```
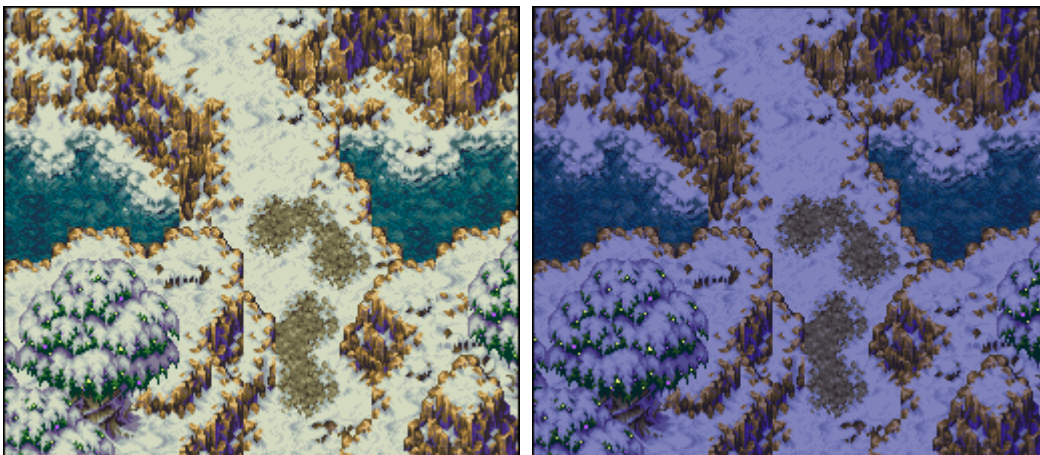

This image was turned into gold just by changing a few colors

ApplyNight

It creates a nice transition night effect.

Configuration tags:

- o **rulePalette**: The default rule palette is "46%-46%-2%".It is based on RGB, which means that the rule will reduce 46% of red, 46% of green and only 2% of blue. The idea is to create a dark blue atmosphere.
- o **avoidTheseColors**: It allows you to create an even more realistic night effect. Here some colors will ignore the palette rule (excellent for light effects). The pattern is "R-G-B" and uses "," for multiple colors. Example: 247-255-66,255-189-66.
- o **progressive**: You choose between false or true. False will apply the palette rule completely turning the image into night at the moment. Choosing true, the effect will be progressive (animated). This is only available inside the "basic" tag.
  - ▪ **speedTransition**: Time in milliseconds for the progressive transition.
  - ▪ **returnTo**: Choose between true or false. True will return the image to daylight.
    - • **returnToAfter**: Time in milliseconds to return the image to daylight.
- o **daytime**: You can adjust the level of night choosing a value between 0 and 10. 0 means no effect applied and 10 is the maximum level.

Example:

```
<postEffects>
  <effect>
    <name>applyNight</name>
    <rulePalette>46%-46%-2%</rulePalette>
    <avoidTheseColors>255-66-165,107-156-255,57-255-99,165-66-247</avoidTheseColors>
    <progressive>true</progressive>
    <speedTransition>200</speedTransition>
    <returnTo>true</returnTo>
    <returnToAfter>5000</returnToAfter>
  </effect>
</postEffects>
```

CreateNoise

The same effect that happens in old TVs when didn't have a signal.

Configuration tags:

- o **replaceColors**: The colors added here will be replaced by the noise effect. The pattern is "R-G-B" and uses "," for multiple colors. Example: 247-255-66,255-189-66.
- o **randomColors**: These are the colors used to create the effect. The pattern is "R-G-B" and uses "," for multiple colors. Example: 247-255-66,255-189-66. Optional, the app will pick random colors when the tag is not used.
- o **numberOfFrames**: This is optional and only used with the tag "createFramesBasedOnPostEffect". You can assign the number frames used to create the noise effect. The more frames you use, the more realistic it looks.

Example:

```
<postEffects>
  <effect>
    <name>CreateNoise</name>
    <replaceColors>21-95-217</replaceColors>
    <randomColors>255-255-255,0-0-255,255-255-0,0-0-0</randomColors>
  </effect>
</postEffects>
```

Ice
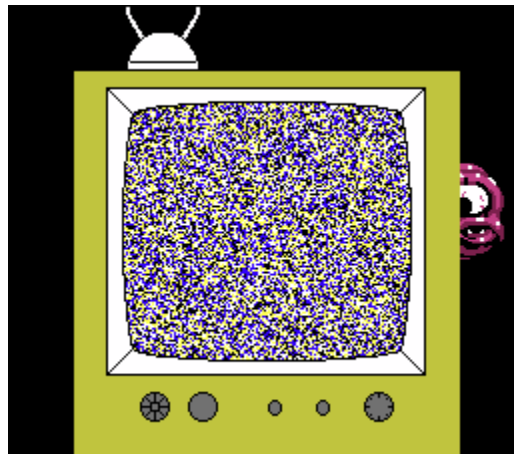
It changes the colors making it look frozen. The way it works is simple, the colors are replaced and color cycling is used for the animation.

Configuration tags:

- o **timePause**: The ice effect works using color cycling. However, it pauses to make it look more realistic. Choose here how many milliseconds will be stopped.
- o **timeSpeedAnimated**: Choose here how many milliseconds will be animated.
- o **iceColors**: This will be the palette of colors used to replace the current ones and create the ice effect. The pattern is "R-G-B" and uses "," for multiple colors. Example: 247-255-66,255-189-66.
- o **colorsToFreeze**: You can choose what colors will be frozen. This tag is optional, without it all the colors will be frozen. The pattern is "R-G-B" and uses "," for multiple colors. Example: 247-255-66,255-189-66.
- o **startFromFrame**: The animation is done using color cycling based on the colors included inside the setting tag "**iceColors**". You can push the colors forward by adding a number. For example, if you add a value of 5, the color cycling animation will be pushed 5 positions forward.

Example:

```
<postEffects>
  <effect>
    <name>ice</name>
    <timePause>1000</timePause>
    <timeSpeedAnimated>30</timeSpeedAnimated>
    <iceColors>239-239-255,90-89-255,140-142-255</iceColors>
    <colorsToFreeze>74-148-0,181-33-181,148-0-148</colorsToFreeze>
  </effect>
</postEffects>
```

Melting

This post effect only affects the first frame and has its own animation. Create a melting effect and when it is finished, the animation returns to normal and the effect is disabled.

Configuration tags:

- o **changeFrameRate**: This effect runs better at 60 fps. You can change the frame rate in your animation momentarily here. The frame rate will return to normal after the effect is finished. Keep in mind that doesn't work inside "createFramesBasedOnPostEffect". However, you can use the event "TriggerActionByCurrentFrame" and the action "UpdateFPS" to do the same.
- o **maxFramesEffect**: This is optional and only used with the tag "createFramesBasedOnPostEffect". The default value is the height in pixels of the image.

Note: Use Melting inside "applyAlterationsToThisFrame" is not possible, use "createFramesBasedOnPostEffect" instead.

Example:

Inside "basic" tag.

```xml
<basic type="animations">
  <screenCanvasResolution>
    <width>210</width>
    <height>315</height>
  </screenCanvasResolution>
  <canvasResolution>
    <maxWidth>287</maxWidth>
    <maxHeight>221</maxHeight>
  </canvasResolution>
  <bgColor>#ff000000</bgColor>
  <postEffects>
    <effect>
      <effectName>Melting</effectName>
      <changeFrameRate>60</changeFrameRate>
    </effect>
  </postEffects>
</basic>
<images>
  <image>
    <frames>
      <frame>
        <fileName>ryu1.png</fileName>
      </frame>
```

Inside "createFramesBasedOnPostEffect" tag.

```xml
<images>
  <image>
    <frames>
      <frameCollection>
        <createFramesBasedOnPostEffect>
          <fileName>ryu1.png</fileName>
          <postEffects>
            <effect>
              <effectName>Melting</effectName>
            </effect>
          </postEffects>
        </createFramesBasedOnPostEffect>
        <repeatThisLoopForTimes>1</repeatThisLoopForTimes>
      </frameCollection>
      <frameCollection>
        <frame>
          <fileName>ryu1.png</fileName>
        </frame>
        <frame>
          <fileName>ryu2.png</fileName>
        </frame>
```

The tag "changeFrameRate" doesn't work here but you can use the event "TriggerActionByCurrentFrame". Example:

```xml
<events>
  <event>
    <eventType>TriggerActionByCurrentFrame</eventType>
    <eventActions>
      <eventAction>UpdateFPS</eventAction>
    </eventActions>
    <whenCurrentFrameIndexIs>0</whenCurrentFrameIndexIs>
    <whenFrameCollectionIndexIs>0</whenFrameCollectionIndexIs>
    <fps>60</fps>
  </event>
  <event>
    <eventType>TriggerActionByCurrentFrame</eventType>
    <eventActions>
      <eventAction>UpdateFPS</eventAction>
    </eventActions>
    <whenCurrentFrameIndexIs>0</whenCurrentFrameIndexIs>
    <whenFrameCollectionIndexIs>1</whenFrameCollectionIndexIs>
    <fps>20</fps>
  </event>
</events>
```