Sprite Properties

When you add a sprite to a map the Million Tile Engine appends several properties used for positioning and control. These properties are easily retrieved to ascertain the sprite's position, scale, and size in relation to the map. Most should be treated as read-only properties. They're provided as a convenience but not intended for direct modification unless otherwise stated.

locX
The X coordinate of the sprite's location on the map.

locY
The Y coordinate of the sprite's location on the map.

levelPosX
The X coordinate of the sprite's position on the map. On orthographic maps a sprite's level position is almost always equal to a sprite's .x and .y properties. If sprite.x is 100, it is usually safe to assume that sprite.levelPosX is also 100. The exception to this is when worldWrap is enabled and the camera approaches the edge of a map. In order for the camera to "see" a sprite across the world wrap, the sprite is moved by the engine into the camera view, which means the sprite's .x and .y properties are changed. In this case the levelPosX of a sprite is the true position on the map while the .x of a sprite is the position of the object in the group as reckoned by Corona.

Isometric maps are another situation where a sprite's .x is not equal to a sprite's levelPosX (and usually never is). The isometric perspective necessitates stricter control of sprite positions by the engine.

levelPosY
The Y coordinate of the sprite's position on the map. (see levelPosX)

offsetX
The X rendering offset of the sprite. Modifies the sprite's apparent position without affecting the sprite's actual level position or location within the world or map.

offsetY
The Y rendering offset of the sprite. Modifies the sprite's apparent position without affecting the sprite's actual level position or location within the world or map.

layer
The layer containing the sprite.

level
The level containing the sprite.

levelWidth
The width of the sprite in relation to the world. For example, if the map tiles are 32 x 32, a levelWidth of 32 will make the sprite appear one tile wide, regardless of the scale of the tiles or the scale of the level.

levelHeight
The height of the sprite in relation to the world. (see levelWidth)

xScale
Same as Corona SDK's xScale property. The xScale is calculated by the engine to make a sprite conform to it's levelWidth and levelHeight.

yScale
Same as Corona SDK's yScale property. (see xScale)

deltaX
Table containing all of a sprite's queued moves along the X axis, used internally by moveSprite() and moveSpriteTo().

deltaY
Table containing all of a sprite's queued moves along the Y axis, used internally by moveSprite() and moveSpriteTo().

isMoving
Boolean, "true" if the sprite is moving, "false" or nil otherwise. Useful for detecting when a sprite has finished a move from tile to tile in RPG movement systems.

light
(Lighting System) Table containing the data which defines the sprite's dynamic light source (see addLight below).

pointLight
Table describing a sprite's point light source, used with normal mapping. See Corona Lab's documentation for composite.normalMapWith1PointLight. This property and it's parameters are all optional. A sprite set to be a point light source in MTE will emit a default white light.

Format for pointLight:

pointLightPos: (table) Sets the offset of the source from the sprite in the x, y, and z axis. For example, {0, 100, 0.1} places the light 100 pixels below the sprite on the screen. The z axis parameter primarily effects how the light is cast from the source. A z value closer to 0 will produce more prominent shadows along a normal mapped surface.

pointLightColor: (table) Sets the color and alpha of the light. Also has the effect of setting the color of the entire normal mapped layer when ambientLightIntensity is not 0.

For example, a value of {1, 0, 0, 1} will turn the light red and also cast a red hue across the layer if ambientLightIntensity is greater than 0; the “shadows” will appear reddish.

ambientLightIntensity: (number) The intensity of the light far from the source. The light from the point source will decrease in intensity with distance down to the ambientLightIntensity.

attenuationFactors: (table) A table containing the constant, linear, and quadratic attenuation factors.

color
Table setting the absolute color of the sprite as a table of red, green, and blue values. For example a fully lit sprite will have the following color table: {1, 1, 1}. This property is only necessary when lighting is applied to a sprite. If a sprite’s lighting property = false, it is not needed.

Setting a sprite’s color property allows the developer to tint a sprite and also set the sprite to react to lighting. Otherwise MTE’s lighting system will overwrite any color applied to the sprite.

Sprite Methods

The function addSprite() also adds the following methods for converting from abstract internally managed coordinates to the world’s level position.

addLight(light)
(Lighting System) Adds a dynamic light to a sprite. ‘light’ is a table defining the light source. Sprites may only have one light source. The parameters below may be changed directly by the user anytime after addLight has been called. They are accessible at sprite.light, so source for example would be set at sprite.light.source. Calling addLight multiple times for the same sprite is not recommended.

Format for light:

source: Specifies the red, green, and blue brightness of a light source as an array of three values. For example, a source of {255, 255, 255} will produce white light of maximum brightness while {0, 50, 0} will produce a very dim green light.

arc: Light sources emit light in 360 degrees by default. This property is an array of two values defining the start and stop angle of a limited arc. A value of {0, 90} will restrict the light output to the arc between 0 and 90 degrees.

rays: Light sources emit light in 360 degrees by default. This property is an array of rays. A value of {0, 90} will cause a light source to emit exactly two rays of light, one aimed at 0 degrees (directly to the right), and another aimed at 90 degrees (directly down towards the bottom of the screen). The array may contain as many rays as desired.

range: Specifies the range of the red, green, and blue light emitted from a source. The unit of measure is 1 tile and defines a radius around a source. A light with a range of {10, 5, 1} will cast red light up to 10 tiles, green light up to 5 tiles, and blue light up to 1 tile from the source.

falloff: The decrease in red, green, and blue brightness when light from a source moves 1 tile in any direction. The distance light travels is measured in a straight line along the light ray emitted by the source. This property is useful for creating lights which dim towards darkness along their edges. For example, a source of {255, 255, 255} with a range of {10, 10, 10} and a falloff of {25, 25, 25} will create a circle of light which is very bright at the center but gradually diminishes towards darkness at its maximum range.

layer: Each light source can only react to opaque tiles on one layer of a map at a time. This property specifies the layer as an absolute value. A light with a layer of 1 will be impeded by opaque tiles on layer 1, but continue uninterrupted through opaque tiles on layer 2. The layer can be different from the layer of the source object.

layerRelative: Similar to above, but defined in relation to the source of the light. If a light-emitting tile is on layer 2 and its lightLayerRelative property is -1, the light will react to tiles on layer 1.

layerFalloff: The decrease in red, green, and blue brightness when light moves from one layer to an adjacent layer above or below it. Set as an array of three values, one each for red, green, and blue. For example, say we have a light source on layer 1 giving off max brightness white light {255, 255, 255} and our lightLayerFalloff is {10, 10, 10}; tiles on layer 2 will only receive a maximum of {245, 245, 245} from the light source, no matter how close to the source they are.

levelFalloff: This works the same as lightLayerFalloff above, however the decrease is applied when light from a source moves from one level to an adjacent level above or below it. This is useful for when you want a dynamic light source to only light tiles on the level it is moving through, leaving levels above and below dark. Both layerFalloff and levelFalloff can be used together.

id: The unique identifier of the light source. The user should not alter this property.

removeLight()
(Lighting System) Removes a sprite's dynamic light.

addLightingListener(lightID, listener)
(Lighting System) Monitors the sprite for light of a specific source. If the sprite is illuminated by a light with the specified lightID, dispatches an event to the specified listener function. Returns an event table.

lightID: (string) The ID of the light source.

listener: (function) The listener to be triggered when light of the correct light ID is detected.

Format for event table:

name: (string) The ID of the light source.

target: The sprite executing the listener.

source: The table defining the light source. Dynamic lights store a reference to their display object owners in the source table such that event.source.object is the display object emitting the light.

light: An array containing the red, green, and blue color values of the light. For example; {155, 200, 17}

phase: The phase of the event. Valid values are “began”, “maintained”, and “ended”.

Example:

```
local onLighting = function(event)
	if event.phase == “began” then
		event.target:setSequence(“caught”)
		event.target:play()
	end
end

mySprite:addLightingListener(“enemyFlashlight”, onLighting)
```