

Sprite

Use this class if you have an image with no frames (so no animation). Especially good for backgrounds, terrain, walls, etc..

Properties

x	X position where image is to be painted on the canvas
y	Y position where image is to be painted on the canvas
width	Actual width of the image in pixels
height	Actual height of the image in pixels
useWidth	Width to use when painting the image. Image will be stretched/shrunk if different from the actual width.
useHeight	Height to use when painting the image. Image will be stretched/shrunk if different from the actual height.
dx	Change in direction in the X axis for each screen repaint
dy	Change in direction in the Y axis for each screen repaint
image	The image variable for the image. Note that all images must first be loaded into image variables before using in these classes.
alpha	Transparency of the Sprite from 0.0 (invisible) to 1.0 (fully visible)

collision	Indicates if the Sprite is in collision with another object or not. This is only checked when one of the collision methods are called. Note that every time a collision method is called collision is first set to false and then checked for collision again. This is important since the criteria for the different collision methods can be different.
visible	Indicates if the image is to be painted or not when draw is called. Can also be used to decide if image should be used for collision detection.

Methods

draw()	Using all the current properties, the image is drawn on the canvas
checkCollisions(obj)	<p>Returns a Boolean indicating if this Sprite and the passed object (can be another Sprite, SpriteSheet, or SpriteArray) are in collision or not. True = they are; false = they are not.</p> <p>It also sets the Sprites collision property to the appropriate value as well.</p>
checkBottomCollision(obj)	<p>Returns a Boolean indicating if this Sprite and the passed object (can be another Sprite, SpriteSheet, or SpriteArray) WILL BE IN COLLISION if this Sprite moves DOWN one more pixel. True = they are; false = they are not.</p> <p>It also sets the Sprites collision property to the appropriate value as well.</p>
checkTopCollision(obj)	<p>Returns a Boolean indicating if this Sprite and the passed object (can be another Sprite, SpriteSheet, or SpriteArray) WILL BE IN COLLISION if this Sprite moves UP one more pixel. True = they are; false = they are not.</p> <p>It also sets the Sprites collision property to the appropriate value as well.</p>

<code>checkRightCollision(obj)</code>	<p>Returns a Boolean indicating if this Sprite and the passed object (can be another Sprite, SpriteSheet, or SpriteArray) WILL BE IN COLLISION if this Sprite moves RIGHT one more pixel. True = they are; false = they are not.</p> <p>It also sets the Sprites collision property to the appropriate value as well.</p>
<code>checkLeftCollision(obj)</code>	<p>Returns a Boolean indicating if this Sprite and the passed object (can be another Sprite, SpriteSheet, or SpriteArray) WILL BE IN COLLISION if this Sprite moves LEFT one more pixel. True = they are; false = they are not.</p> <p>It also sets the Sprites collision property to the appropriate value as well.</p>

Usage

```

var background = new Image();
background.src = "maze2.png";
var bg = new Sprite(0,0,1200,600,1200,600, background);

```

Line 1: Create a new image variable.

Line 2: Then set the image file (src) to your image's file name. Note that if the image is not in the same folder as your HTML you'll need to provide the path to the image as well in the name. game.imageDir will point to the image sub-directory by default.

Line 3: Create the Sprite variable passing the following values in order:

1. X position to paint
2. Y position to paint
3. Image width in pixels
4. Image height in pixels
5. Width to use in pixels
6. Height to use in pixels
7. Image from step #1

SpriteSheet

Use this for an image that has animation/motion. Use if all the “frames” are collected within a single image file.

Properties

x	X position where image is to be painted on the canvas
y	Y position where image is to be painted on the canvas
width	Actual width of the image in pixels
height	Actual height of the image in pixels
useWidth	Width to use when painting the image. Image will be stretched/shrunk if different from the actual width.
useHeight	Height to use when painting the image. Image will be stretched/shrunk if different from the actual height.
dX	Change in direction in the X axis for each screen repaint
dY	Change in direction in the Y axis for each screen repaint
image	The image variable for the image. Note that all images must first be loaded into image variables before using in these classes.
alpha	Transparency of the Spite from 0.0 (invisible) to 1.0 (fully visible)

<code>collision</code>	Indicates if the Sprite is in collision with another object or not. This is only checked when one of the collision methods are called. Note that every time a collision method is called <code>collision</code> is first set to false and then checked for collision again. This is important since the criteria for the different collision methods can be different.
<code>visible</code>	Indicates if the image is to be painted or not when <code>draw</code> is called. Can also be used to decide if image should be used for collision detection.

Methods

<code>draw()</code>	Using all the current properties, the image is drawn on the canvas
<code>checkCollisions(obj)</code>	<p>Returns a Boolean indicating if this Sprite and the passed object (can be another Sprite, SpriteSheet, or SpriteArray) are in collision or not. True = they are; false = they are not.</p> <p>It also sets the Sprites collision property to the appropriate value as well.</p>
<code>checkBottomCollision(obj)</code>	<p>Returns a Boolean indicating if this Sprite and the passed object (can be another Sprite, SpriteSheet, or SpriteArray) WILL BE IN COLLISION if this Sprite moves DOWN one more pixel. True = they are; false = they are not.</p> <p>It also sets the Sprites collision property to the appropriate value as well.</p>
<code>checkTopCollision(obj)</code>	<p>Returns a Boolean indicating if this Sprite and the passed object (can be another Sprite, SpriteSheet, or SpriteArray) WILL BE IN COLLISION if this Sprite moves UP one more pixel. True = they are; false = they are not.</p> <p>It also sets the Sprites collision property to the appropriate value as well.</p>

checkRightCollision(obj)	<p>Returns a Boolean indicating if this Sprite and the passed object (can be another Sprite, SpriteSheet, or SpriteArray) WILL BE IN COLLISION if this Sprite moves RIGHT one more pixel. True = they are; false = they are not.</p> <p>It also sets the Sprites collision property to the appropriate value as well.</p>
checkLeftCollision(obj)	<p>Returns a Boolean indicating if this Sprite and the passed object (can be another Sprite, SpriteSheet, or SpriteArray) WILL BE IN COLLISION if this Sprite moves LEFT one more pixel. True = they are; false = they are not.</p> <p>It also sets the Sprites collision property to the appropriate value as well.</p>

Usage

```

var img = new Image();
img.src = "pacman.png";
pacman = new SpriteSheet (100,100,128,128,32,32, 3,3,img);

```

Line 1: Create a new image variable.

Line 2: Then set the image file (src) to your image's file name. Note that if the image is not in the same folder as your HTML you'll need to provide the path to the image as well in the name.

Line 3: Create the Sprite variable passing the following values in order:

1. X position to paint
2. Y position to paint
3. Image width in pixels
4. Image height in pixels
5. Width to use in pixels
6. Height to use in pixels
7. Direction – # of rows for the 2 dimensional array of images
8. Frame – # of columns of the two dimensional array of images
9. Background image from step #1

SpriteArray

Use this for an image that has animation/motion. Use if all the “frames” are spread across multiple image files. You will first need to compile all of the images into an array of images.

Properties

x	X position where image is to be painted on the canvas
y	Y position where image is to be painted on the canvas
width	Actual width of the image in pixels
height	Actual height of the image in pixels
useWidth	Width to use when painting the image. Image will be stretched/shrunk if different from the actual width.
useHeight	Height to use when painting the image. Image will be stretched/shrunk if different from the actual height.
dX	Change in direction in the X axis for each screen repaint (change is NOT automatically implemented, you must use this value yourself in code)
dY	Change in direction in the Y axis for each screen repaint (change is NOT automatically implemented, you must use this value yourself in code)

<code>image</code>	The image variable for the image. Note that all images must first be loaded into image variables before using in these classes.
<code>alpha</code>	Transparency of the Sprite from 0.0 (invisible) to 1.0 (fully visible)
<code>collision</code>	Indicates if the Sprite is in collision with another object or not. This is only checked when one of the collision methods are called. Note that every time a collision method is called collision is first set to false and then checked for collision again. This is important since the criteria for the different collision methods can be different.
<code>visible</code>	Indicates if the image is to be painted or not when draw is called. Can also be used to decide if image should be used for collision detection.

Methods

<code>draw()</code>	Using all the current properties, the image is drawn on the canvas
<code>checkCollisions(obj)</code>	<p>Returns a Boolean indicating if this Sprite and the passed object (can be another Sprite, SpriteSheet, or SpriteArray) are in collision or not. True = they are; false = they are not.</p> <p>It also sets the Sprites collision property to the appropriate value as well.</p>
<code>checkBottomCollision()</code>	<p>Returns a Boolean indicating if this Sprite and the passed object (can be another Sprite, SpriteSheet, or SpriteArray) WILL BE IN COLLISION if this Sprite moves DOWN one more pixel. True = they are; false = they are not.</p> <p>It also sets the Sprites collision property to the appropriate value as well.</p>

checkTopCollision()	<p>Returns a Boolean indicating if this Sprite and the passed object (can be another Sprite, SpriteSheet, or SpriteArray) WILL BE IN COLLISION if this Sprite moves UP one more pixel. True = they are; false = they are not.</p> <p>It also sets the Sprites collision property to the appropriate value as well.</p>
checkRightCollision(obj)	<p>Returns a Boolean indicating if this Sprite and the passed object (can be another Sprite, SpriteSheet, or SpriteArray) WILL BE IN COLLISION if this Sprite moves RIGHT one more pixel. True = they are; false = they are not.</p> <p>It also sets the Sprites collision property to the appropriate value as well.</p>
checkLeftCollision(obj)	<p>Returns a Boolean indicating if this Sprite and the passed object (can be another Sprite, SpriteSheet, or SpriteArray) WILL BE IN COLLISION if this Sprite moves LEFT one more pixel. True = they are; false = they are not.</p> <p>It also sets the Sprites collision property to the appropriate value as well.</p>

Usage

```

var starArray = [];
starArray[0] = "star coin rotate 1.png";
starArray[1] = "star coin rotate 2.png";
starArray[2] = "star coin rotate 3.png";
starArray[3] = "star coin rotate 4.png";
starArray[4] = "star coin rotate 5.png";
starArray[5] = "star coin rotate 6.png";
aStar = new SpriteArray(45,45,1957,2242, 45, 45,0, 0, starArray);

```

Line 1: Create the array that will store all of the images. Line 2: Load the images into individual elements within the array Line 3: Create the Sprite variable passing the following values in order:

1. X position to paint
2. Y position to paint
3. Image width in pixels
4. Image height in pixels

5. Width to use in pixels
6. Height to use in pixels
7. Direction – the row for the 2 dimensional array of images
8. Frame – the column of the two dimensional array of images
9. Image array #1

ScrollingBackground

This Object is very much like a regular Sprite, except special features have been added to enable scrolling of the image

Properties

x	X position where image is to be painted on the canvas
y	Y position where image is to be painted on the canvas
width	Actual width of the image in pixels
height	Actual height of the image in pixels
useWidth	Width to use when painting the image. Image will be stretched/shrunk if different from the actual width.
useHeight	Height to use when painting the image. Image will be stretched/shrunk if different from the actual height.
dX	Change in direction in the X axis for each screen repaint
dY	Change in direction in the Y axis for each screen repaint
image	The image variable for the image. Note that all images must first be loaded into image variables before using in these classes.

alpha	Transparency of the Sprite from 0.0 (invisible) to 1.0 (fully visible)
collision	Indicates if the Sprite is in collision with another object or not. This is only checked when one of the collision methods are called. Note that every time a collision method is called collision is first set to false and then checked for collision again. This is important since the criteria for the different collision methods can be different.
visible	Indicates if the image is to be painted or not when draw is called. Can also be used to decide if image should be used for collision detection.
advanceBackground	Boolean that indicates IF the background is to scroll. Utilizes the bgDX and bgDY values from the GameMaster object to perform this action

Methods

draw()	Using all the current properties, the image is drawn on the canvas
--------	--

Usage

```

var background = new Image();
background.src = "maze2.png";
var bg = new ScrollingBackGround(0,0,1200,600,1200,600,background);

```

Line 1: Load the images into individual image variable
Line 2: Create the Sprite variable passing the following values in order

1. X position to paint
2. Y position to paint
3. Image width in pixels
4. Image height in pixels
5. Width to use in pixels

6. Height to use in pixels
7. Image array #1

GameText

Lets you print text to the canvas – like a score


Properties

x	X position where image is to be painted on the canvas
y	Y position where image is to be painted on the canvas
font	Font to use – note must be on the computer
fillstyle	Color
alpha	Transparency of the Spite from 0.0 (invisible) to 1.0 (fully visible)

Methods

draw()	Using all the current properties, the image is drawn on the canvas
--------	--

Usage


```
 var score = new GameText();  
    // score.font = "arial";  
    // score.fillStyle = "red";  
    // score.alpha = 0.5;  
    // score.x = 100;  
    // score.y = 360;  
    score.draw("Score: 100");
```

Line 1: Create the text object Line 2-5: (Optional) change text settings Line 6: Create the Text

passing the text to display in the parenthesis

Keyboard Mapping

Create the object myKeys as below...

```
 var myKeys = new KeysPresses();
```

This will give you instant access to all the properties below which will indicate if the key is currently pressed down or not.

Key reference

Code	Name	Code	Name	Code	Name	Code	Name
8	backSpace	54	key_6	86	key_v	114	f3
9	tab	55	key_7	87	key_w	115	f4
13	enter	56	key_8	88	key_x	116	f5
16	shift	57	key_9	89	key_y	117	f6
17	ctrl	65	key_a	90	key_z	118	f7
18	alt	66	key_b	91	leftWindowKey	119	f8
19	pauseBreak	67	key_c	92	rightWindowKey	120	f9
20	capsLock	68	key_d	93	selectKey	121	f10

27	escape	69	key_e	96	numpad0	122	f11
33	pageUp	70	key_f	97	numpad1	123	f12
34	pageDown	71	key_g	98	numpad2	144	numLock
35	end	72	key_h	99	numpad3	145	scrollLock
36	home	73	key_i	100	numpad4	186	semiColon
37	leftArrow	74	key_j	101	numpad5	187	equalSign
38	upArrow	75	key_k	102	numpad6	188	comma
39	rightArrow	76	key_l	103	numpad7	189	dash
40	downArrow	77	key_m	104	numpad8	190	period
45	insert	78	key_n	105	numpad9	191	forwardSlash
46	delete	79	key_o	106	multiplyKey	192	graveAccent
48	key_0	80	key_p	107	addKey	219	openBracket
49	key_1	81	key_q	109	subtractKey	220	backSlash
50	key_2	82	key_r	110	decimalPoint	221	closeBracket

51	key_3	83	key_s	111	divideKey	222	singleQuote
52	key_4	84	key_t	112	f1		
53	key_5	85	key_u	113	f2		

Mouse Tracking

keyPresses will also keep track of certain mouse conditions:

```
mouseX // = current position of the mouse in the X direction on the page ( not the canvas)
mouseY // = current position of the mouse in the Y direction ( not the canvas)
mouseLeftButtonClick // = true if left Mouse Button Clicked
mouseRightButtonClick // = true if Right Mouse Button Clicked
mouseCenterButtonClick // = true if Center Mouse Button Clicked
mouseLeftButtonUp // = true if left Mouse Button Up
mouseRightButtonUp // = true if Right Mouse Button Up
mouseCenterButtonUp // = true if Center Mouse Button Up
mouseLeftButtonDown // = true if left Mouse Button Down
mouseRightButtonDown // = true if Right Mouse Button Down
mouseCenterButtonDown // = true if Center Mouse Button Down
```