

Ai→Canvas

A project by [Mike Swanson](#)

Introduction

The Ai->Canvas plug-in enables Adobe® Illustrator® to export vector and bitmap artwork directly to an HTML5 canvas element using JavaScript drawing commands. Animation can be added to control rotation, scaling, opacity, and motion along a path. Then, events can be used to trigger other animations. Finally, the exported HTML and JavaScript can be extended and used in your own applications running on the latest versions of Internet Explorer, Firefox, Chrome, Safari, and Opera.

The *Tutorial Assets* folder includes the graphic and audio assets used in the examples.

You can also watch a [video version of these tutorials on YouTube](#).

Drawing

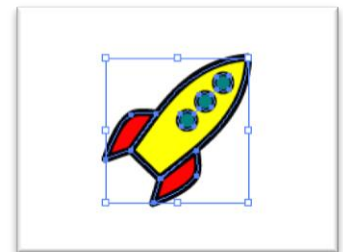
Almost any Illustrator artwork can be exported to a HTML5 canvas element. Where canvas lacks support for a specific feature, Ai->Canvas automatically rasterizes the artwork to a bitmap and properly positions the image on the canvas.

Artwork

([related video on YouTube](#))

Because Illustrator supports so many file formats (AI, PDF, AutoCAD, CorelDRAW, EPS, FreeHand, SVG, and more), it's easy to get started by opening existing drawings or by downloading artwork from the Internet.

Or, you can create new drawings. Ai->Canvas is able to export complex shapes, fill styles, line styles, gradients, transparency, native text, bitmaps, pattern fills, symbols, drop shadows, and more.



All artwork is converted to JavaScript drawing commands, and the resulting code can be copied to your own application for further customization.

Example

Note that these instructions are based on Adobe Illustrator CS6. The examples will also work with Adobe Illustrator CC, but the commands may be slightly different.

1. Open Illustrator and create a new document named "Tutorial".
2. Use Illustrator's drawing tools to create a simple rocket ship, or download a free vector rocket ship from: <http://www.clker.com/clipart-rocket.html> (the SVG format works well).

3. Change the name of the layer to **“rocket();”**.
4. Choose File/Export, select “<canvas>” as the type, then click Save to view the results in the browser.

Images

([related video on YouTube](#))

Properly positioning bitmap images on a canvas can be a challenge, especially if the images have been rotated, scaled, or skewed. Illustrator makes it easy to import and place multiple image formats (GIF, JPG, PNG, and more), and Ai->Canvas exports equivalent canvas transformation commands. This makes it much easier to build complex layouts.



Ai->Canvas supports both placed and embedded images. Placed images are exported as a reference to the file on-disk, and embedded images are exported as PNG files that include an alpha channel.

Example

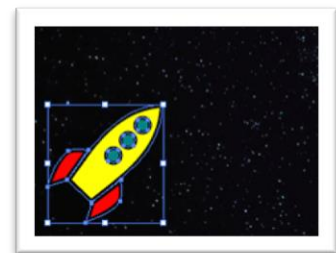
1. Create a new layer called **“stars();”** below the **“rocket();”** layer.
2. Use File/Place to position the “stars.png” file, and click Embed to make the star field an embedded image.
3. Similarly, create three new layers called **“sun();”**, **“earth();”**, and **“moon();”**, and use File/Place to position the “sun.png”, “earth.png”, and “moon.png” files on their respective layers. To leave these as linked images, do not click the Embed button.
4. Choose File/Export/<canvas>, and view the results in the browser.

Rasterizing

([related video on YouTube](#))

Rasterization is the process of converting a vector drawing made of shapes into a raster image made of pixels that is stored as a bitmap file.

Some Illustrator drawings use features that have no direct equivalent in the HTML5 canvas specification (e.g. gradient meshes, Gaussian blurs, and opacity masks). For these features, Ai->Canvas automatically rasterizes the artwork to a PNG file and places it on the canvas.



For performance reasons, it is often useful to convert complex vector artwork into a simple bitmap. This saves the browser from having to draw a series of complex shapes, lines, and fills for artwork that may not change.

Example

1. Create a new layer called **“sunGlow();”** below the **“sun();”** layer.
2. Draw an orange circle behind the sun. Choose Effect/Blur/Gaussian Blur and set the radius to 10.
3. Choose File/Export/<canvas>, and notice that the glow has been automatically rasterized to a bitmap image.

4. Change the “rocket();” layer name to “rocket(**rasterize:rocket**);”
5. Choose File/Export/<canvas>, and notice that the rocket ship drawing has been exported as a file named “rocket.png”.

Animation

Bring your artwork to life by adding animations to your Illustrator drawings. Ai->Canvas supports rotation, scaling, fading, and movement along a path. For more advanced users, add triggers that start one animation when another animation completes.

Rotating, Scaling, and Fading

([related video on YouTube](#))

Each drawing layer can be rotated, scaled, and faded by configuring animation clocks. Animation clocks control how properties change over time, and each clock has eight possible settings (see the *Extended Documentation* for details). Settings include the direction of the animation, its total duration, whether or not it reverses direction, an easing function, and an initial delay before the animation begins.



When animating a shape, it is often useful to reposition the origin point (the location where the x and y axes intersect at 0, 0). Ai->Canvas makes it easy to reposition the origin for each layer so that exported animations are transformed correctly.

Example

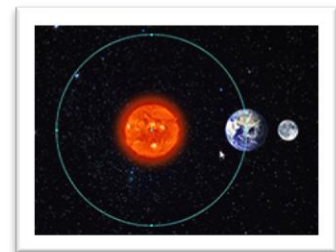
1. Rename the “moon();” layer to “moon(**origin: -1.5, 0.5**);”. This positions the origin point at one-and-a-half times the full width of the moon to the left (outside the bounds of the moon) and at half of its height.
2. To add a second property, insert a semicolon before the new property and its value. Change the layer name to “moon(origin: -1.5, 0.5; **rotate-direction: cw**);”. This sets the rotate animation to move in a clockwise (“cw”) direction.
3. Choose File/Export/<canvas>, and watch the moon rotate around its new origin point.

Moving

([related video on YouTube](#))

Artwork can be moved along a path, and paths can be as simple or as complex as necessary. The path itself is not drawn. Instead, the path is used as a guide to move associated artwork based on the location of its origin point. Any Illustrator tool that creates a basic path (or can be converted to a basic path, like text) can be used to animate motion.

Drawings are also able to follow the orientation of the path. This is useful for elements that need to point in a specific direction as they move.



Example

1. Create a new layer called **“earthOrbit(type: animation; direction: forward; duration: 10);”**. This sets the layer as an animation path, sets a forward direction, and sets the duration of the animation to 10 seconds.
2. In the new layer, use the ellipse tool to draw a circle with the sun at its center. Set its fill to “none” and its stroke to gray. This represents the motion path of Earth’s orbit.
3. To associate the earth layer with the animation path (and reposition its origin point), change the layer name to **“earth(origin: center; animation: earthOrbit);”**.
4. To associate the moon layer, change the layer name to **“moon(origin: -1.5, 0.5; rotate-direction: cw; animation: earthOrbit);”**.
5. Choose File/Export/<canvas>, and watch the Earth and moon move along the path.
6. Create a new layer called **“rocketPath(type: animation; direction: forward; duration: 10);”**.
7. In the new rocketPath layer, use the pen tool to draw a path for the rocket to follow.
8. To associate the rocket with the path (and reposition its origin point), rename the rocket layer to **“rocket(rasterize:rocket; origin: center; animation: rocketPath; follow-orientation: 90);”**. The follow-orientation property sets the “front” of the rocket to 90 degrees (straight up) and configures the drawing to follow the orientation of the path. If the rocket artwork isn’t facing straight up, be sure to rotate it.
9. To allow the rocketPath to begin and end “outside” of the stars, change the stars layer name to **“stars(crop: yes);”**. The crop property clips the artwork to the bounds of the layer.
10. Choose File/Export/<canvas>, and watch the rocket follow the new path.

Easing

([related video on YouTube](#))

For animations to progress in a more natural way, each animation clock can be configured with its own timing function. A timing function affects how fast or how slow an animation progresses at various points along its timeline. For example, these functions can be used to simulate an object that slows (or “eases”) to a stop, like a rolling ball that loses its momentum.



Ai->Canvas includes 24 of [Robert Penner’s easing equations](#) along with additional timing functions for both stepped and random changes. It is also easy to add your own custom timing functions.

Example

1. Create a new layer named **“asteroid();”**, and use File/Place to position the “idaMoon.png” file.
2. Create another layer named **“asteroidPath(type: animation; duration: 8; iterations: 1);”**, and use the line segment tool to draw a straight line for the asteroid to follow. The iterations property controls the number of times an animation repeats.
3. Associate the asteroid with the new path (and reposition its origin point) by renaming the asteroid layer to **“asteroid(origin: center; animation: asteroidPath);”**.

4. Add a timing function that slows down the asteroid as it progresses by renaming the asteroidPath layer to “asteroidPath(type: animation; duration: 8; iterations: 1; **timing-function: quintEaseOut**);”.
5. Choose File/Export/<canvas>, and watch the asteroid slow down as it moves along the path.
6. To make the asteroid look like it’s moving into the distance, add a scale animation to the asteroid layer by renaming it to “asteroid(origin: center; animation: asteroidPath; **scale-direction: shrink; scale-duration: 8; scale-iterations: 1; scale-timing-function: quintEaseIn**);”.
7. Choose File/Export/<canvas>, and watch the asteroid shrink into the distance.

Triggers

([related video on YouTube](#))

Sometimes, one animation needs to start when another animation finishes. To accommodate situations like this, Ai->Canvas enables *triggers*. A trigger is a combination of an *event* (like when an animation clock finishes) with an *action* (like starting another animation clock).



Each animation clock has four events that can be connected to eight actions to orchestrate more sophisticated movement (see the *Extended Documentation* for details). These events can also be used to trigger your own custom JavaScript functions. See the *Events* section under *Code* for an example that shows how to play a sound when an event occurs.

Example

1. To set the asteroid path to start its animation when the moon makes an iteration around the Earth, change the layer name to: “asteroidPath(type: animation; duration: 8; iterations: 1; timing-function: quintEaseOut; **start: moon-rotate-iterated**);”.
2. To set the asteroid scale animation to start at the same time, change its name to: “asteroid(origin: center; animation: asteroidPath; scale-direction: shrink; scale-duration: 8; scale-iterations: 1; scale-timing-function: quintEaseIn; **scale-start: moon-rotate-iterated**);”.
3. Choose File/Export/<canvas>, and notice that the path and scale animations start after the moon completes its first iteration around the Earth.

Code

The purpose of Ai->Canvas is to accelerate your HTML5 canvas development. The HTML and JavaScript that it exports is meant to be copied into your own application and extended. Learn how to subscribe to built-in events, determine if a shape has been clicked, and debug the exported code.

Events

([related video on YouTube](#))

Like the built-in trigger feature (see *Triggers* under *Animation*), your JavaScript code can subscribe to animation clock events and perform a custom action when that event occurs. For example, you may want to play a sound.

```
}  
    I  
function playSound()  
    var whoosh = document.getElementById("whoosh");  
    whoosh.play();  
}
```

Example

1. Open the exported HTML file in your favorite text editor (Visual Studio, Notepad, TextMate, etc.).
2. Add a new audio element just beneath `<canvas>...</canvas>` near the end of the file:

```
<audio id="whoosh" preload>  
    <source src="whoosh.mp3" />  
    <source src="whoosh.wav" />  
</audio>
```

Be sure to copy both audio files to the same folder.

3. Add a new JavaScript function called “playSound();” after the `init()` function:

```
function playSound() {  
    var whoosh = document.getElementById("whoosh");  
    whoosh.play();  
}
```

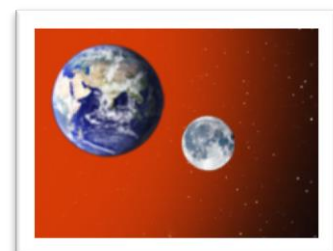
4. Last, find the array index for `asteroidPath` by looking at the “`var animations =`” line near the top of the file (index numbers start at 0). Then add the following to the `init();` function just before “// Start animation clocks”: `animations[0].pathClock.started.subscribe(playSound);`. This causes the `playSound();` function to be called when the `asteroidPath` animation starts.
5. Save the file, and reload it in the browser. The sound should play when the asteroid animation starts.

Interaction

([related video on YouTube](#))

To create interactive canvas applications, it is useful to determine when a shape (like a button) is clicked. Because a canvas is a simple bitmap, testing a mouse click requires configuring the custom shape, then checking to see if the mouse position is within that shape.

Ai->Canvas makes it easy to draw complex shapes that can easily be tested with the canvas `isPointInPath` command.



Example

1. Create a new layer named “**sunSpot();**”, and use the ellipse tool to draw a circle that overlaps the clickable area of the sun.
2. Because the sunSpot layer is only used to configure a path to check for mouse clicks, there is no need to actually see it on the canvas. So, set both fill and stroke styles to “none.”
3. To create a “supernova” animation that will start when the sunSpot area is clicked, rename the sunGlow layer to: “sunGlow(**origin: center; scale-direction: grow; scale-duration: 6; scale-iterations: 1; scale-multiplier: 12; scale-offset: 1; scale-timing-function: expoEaseOut; alpha-direction: fade-out; alpha-duration: 6; alpha-iterations: 1;**)”. The scale-multiplier property multiplies the scale value, and the scale-offset property adds a percentage of the original range.
4. Choose File/Export/<canvas>, and open the exported file in a text editor.
5. Add a new JavaScript function called “sunClick();” after the init() function:

```
function sunClick(e) {  
    sunSpot(ctx);  
    var x = e.clientX + document.body.scrollLeft +  
            document.documentElement.scrollLeft - canvas.offsetLeft;  
    var y = e.clientY + document.body.scrollTop +  
            document.documentElement.scrollTop - canvas.offsetTop;  
    if (ctx.isPointInPath(x, y)) {  
        sunGlow.scaleClock.restart();  
        sunGlow.alphaClock.restart();  
    }  
}
```

6. To prevent the sunGlow animations from starting when the page loads, delete “sunGlow.scaleClock.start();” and “sunGlow.alphaClock.start();” from the init() function.
7. At the end of the init(); function, add “**canvas.addEventListener("click", sunClick, false);**” to capture mouse clicks.
8. Save the file, and reload it in the browser. Click the sunSpot area to start the animations.

Debugging

([related video on YouTube](#))

As drawings and animations increase in complexity, the ability to quickly debug their behavior becomes even more important. By holding down the left shift key during export, helpful information is added to the exported HTML and JavaScript.

In debug mode, the actual frame rate is displayed, and the mouse can be used to scrub forward and backward through time, making it easy to analyze intricate animations.



Example

1. Choose File/Export/<canvas>, and hold down the left shift key while clicking Save (or when clicking Yes to confirm a file overwrite).

2. Note the image, animation, and draw information located below the canvas. For example, the index of an animation function makes it easy to refer to the animation path in code (e.g. `animations[1]`).
3. To scrub the animations, click the mouse anywhere on the canvas, then move the mouse left and right. The Y location of the click determines the total length of the time window, from 0 at the top, to 120 seconds at the bottom. The current and total times are displayed in the upper-left corner of the canvas.