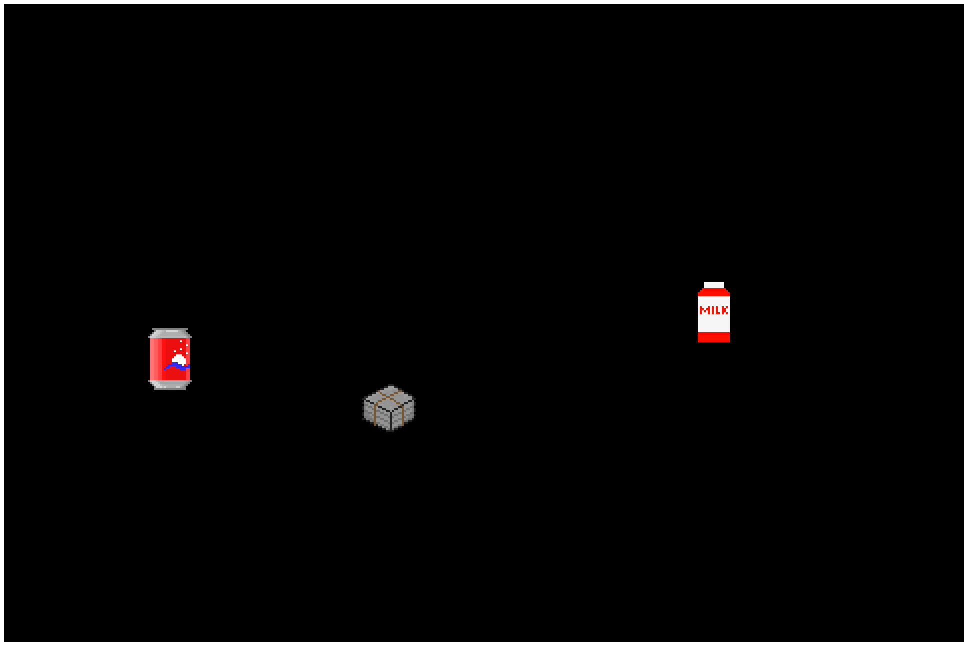
How to Make “Dropcycle” in GameSalad

by J. Matthew Griffis

Note: this is a Beginner-level tutorial. It is recommended, though not required, to read the separate PDF ***GameSalad Basics*** *before continuing.*

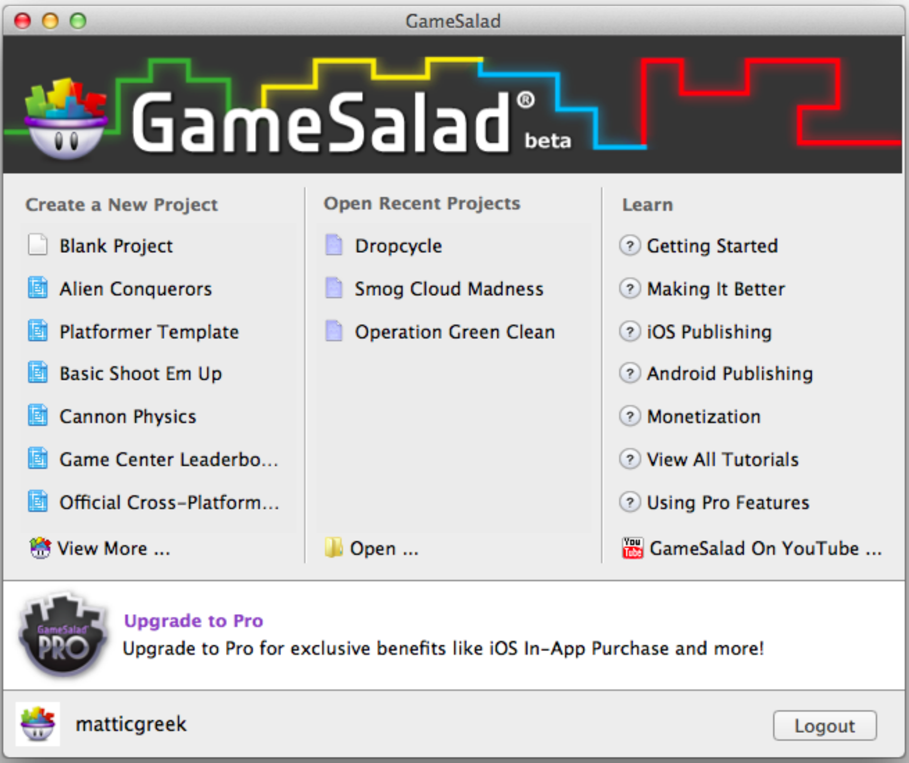


In Dropcycle, the goal couldn’t be simpler: click the falling recyclables to make them reappear at the top of the screen. If they fall to the bottom, they disappear forever! This simplicity makes Dropcycle a great introduction to creating games in GameSalad.

In this tutorial, you’ll learn how to recreate Dropcycle. Make sure to download the folder of Resource Files for the game, and play the game on the website to see it in action.

OK! Let’s get started.

Open GameSalad and double-click on “Blank Project” in the upper-left:

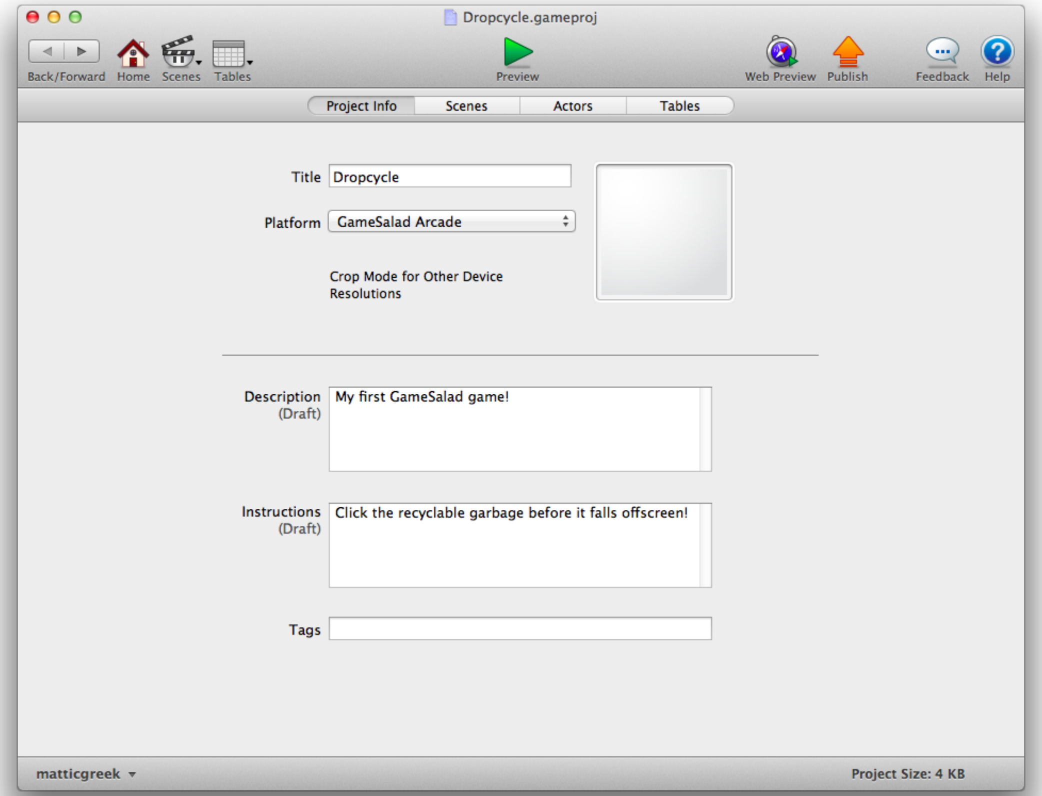


The project opens on the *Project Info* screen. Give your game a title, description and instructions.

Make sure to select a Platform. GameSalad defaults to iPhone but if you want to publish your game online, you'll need to select GameSalad Arcade. When you change the Platform, you'll get a pop-up message warning you that the size of your game window will change. Click "OK."

Finally, click "File" and then "Save" to save your work! **Be sure to do this frequently.**

Your screen should now look something like this:



Title: Dropcycle

Platform: GameSalad Arcade

Crop Mode for Other Device Resolutions

Description (Draft): My first GameSalad game!

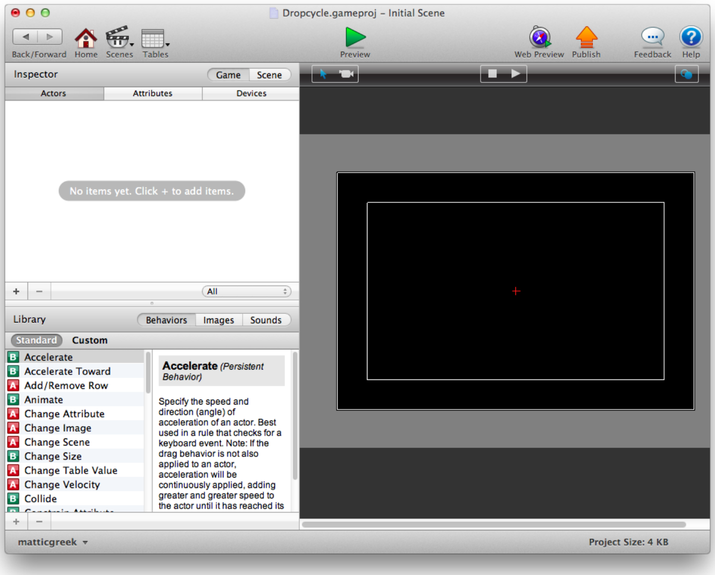
Instructions (Draft): Click the recyclable garbage before it falls offscreen!

Tags:

matticgreek

Project Size: 4 KB

Our game is pretty boring on account of having nothing inside it. Let's add content. Click the *Scenes* tab to the right of *Project Info*, then double-click on "Initial Scene." You should see this:

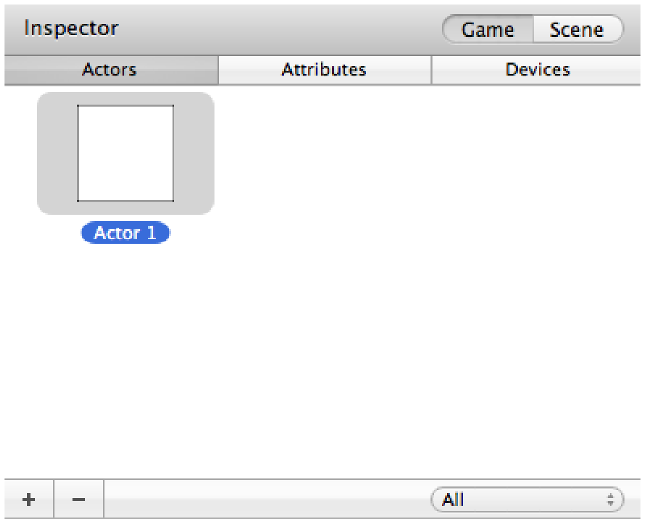


Accelerate *(Persistent Behavior)*

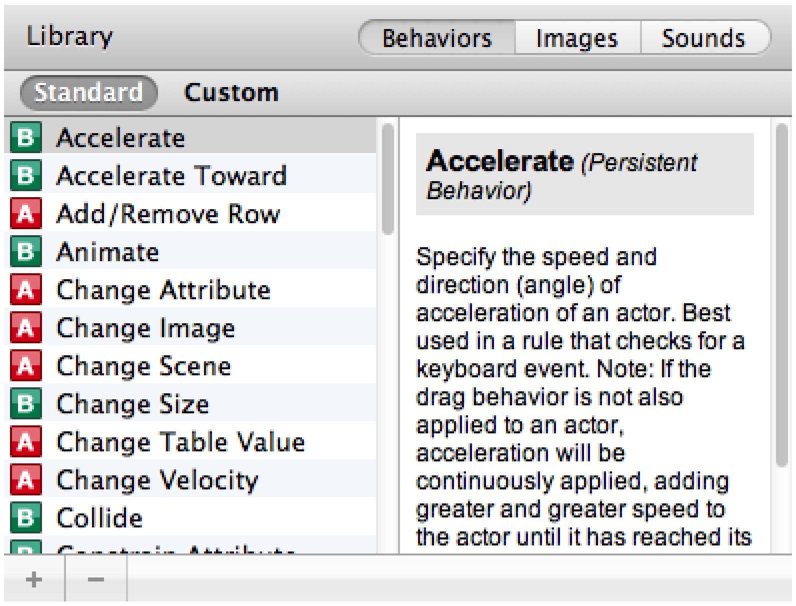
Specify the speed and direction (angle) of acceleration of an actor. Best used in a rule that checks for a keyboard event. Note: If the drag behavior is not also applied to an actor, acceleration will be continuously applied, adding greater and greater speed to the actor until it has reached its

On the right is the “Scene” itself, which is what GameSalad calls the game screen. The black rectangle is the area the player will see. We will drag and drop the components of our game directly onto the Scene. On the left are the tools for creating those components. GameSalad calls the components “Actors.” There is a serious theater metaphor going on here.

Let’s make an Actor! This will be one of the pieces of falling garbage. The white box on the top-left is called the “Inspector,” and in the lower-left of the Inspector is a plus button. Click it to create an Actor:



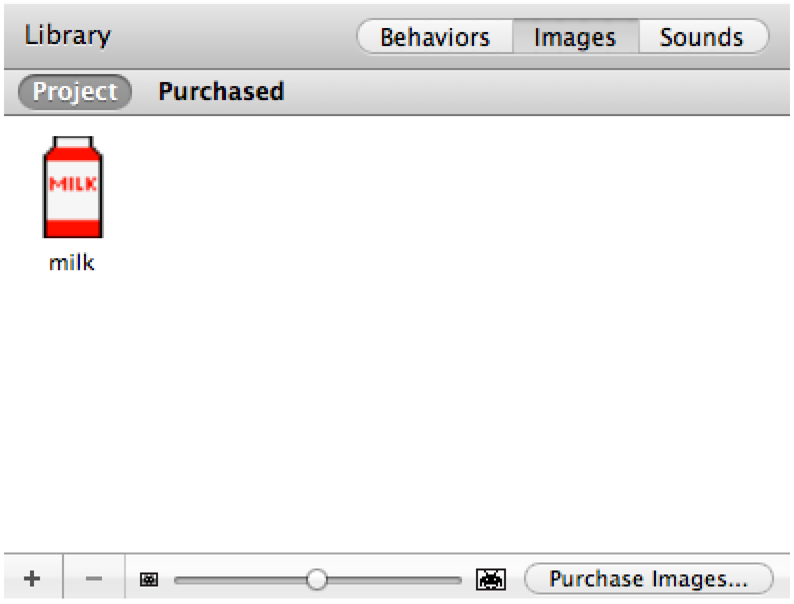
Wait, that looks nothing like any garbage I’ve ever seen! Of course that’s because we haven’t assigned an image yet. To do so, look underneath the Inspector at the “Library”:



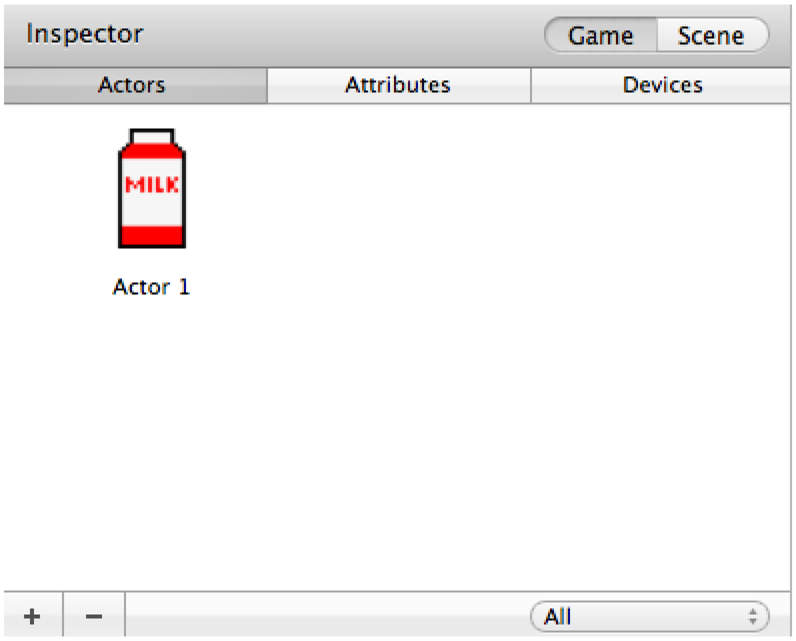
Accelerate *(Persistent Behavior)*

Specify the speed and direction (angle) of acceleration of an actor. Best used in a rule that checks for a keyboard event. Note: If the drag behavior is not also applied to an actor, acceleration will be continuously applied, adding greater and greater speed to the actor until it has reached its

In the upper-right, click on the *Images* tab. Now, find the Resource Files folder you downloaded. Locate the "milk.png" file in the "Sprites" subfolder (a "sprite" is a kind of image), and drag and drop it onto GameSalad in the Images box. GameSalad will import the file and make it available for use:

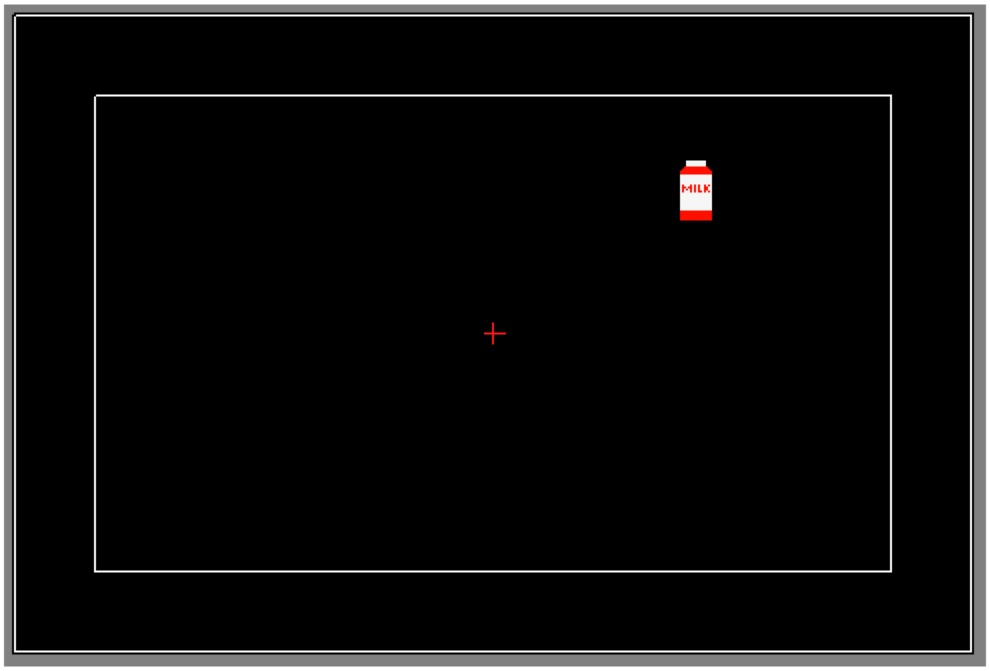


Now, click on "milk" within the Images box and drag and drop it onto "Actor 1" in the Inspector:



That's more like it!

If you look at the Scene, you'll see there's still nothing there. Having created the Actor, you must add it to the Scene. To do so (you guessed it!), drag and drop Actor 1 somewhere within the black rectangle of the Scene:



Alright! You've created an Actor and placed it in the Scene. Let's see it act! Click the large green arrow labeled "Preview" at the top-center of GameSalad and watch as...

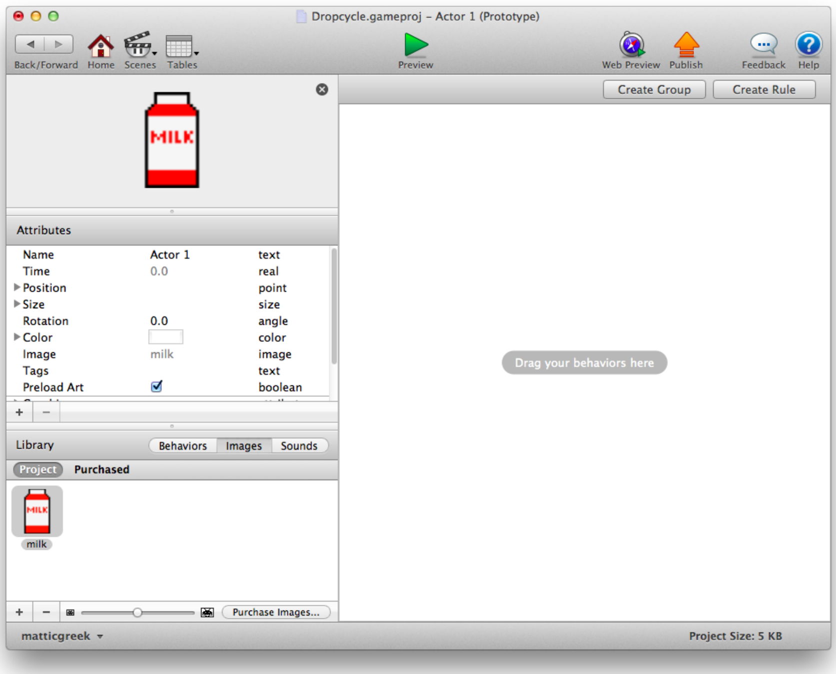
...nothing happens.

Well, you haven't told it to do anything yet. Like any good Actor it requires direction, which GameSalad calls "Behavior."

Let's give the milk carton some behavior. Click the Back button in the upper-left of GameSalad to return to the Scene interface, then double-click on Actor 1 in the Inspector.

Note: it is important that you double-click on the Actor within the Inspector, rather than within the Scene. That is because the milk carton in the Scene is an instance, *whereas the milk carton in the Inspector is a* prototype. *For more, see the separate PDF* ***GameSalad Basics****.*

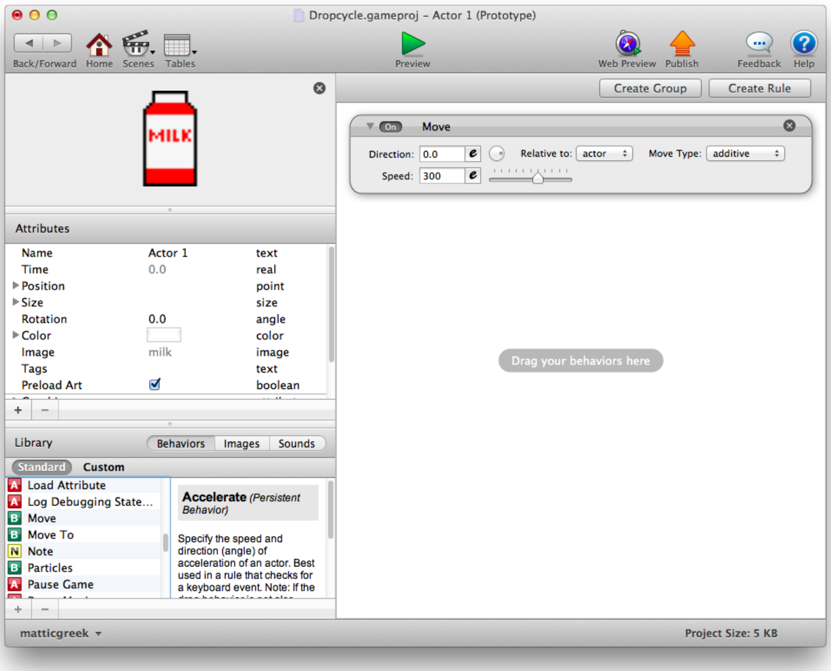
Look at that big empty space on the right, waiting patiently to be filled with instructions:



Attributes

Name	Actor 1	text
Time	0.0	real
Position		point
Size		size
Rotation	0.0	angle
Color		color
Image	milk	image
Tags		text
Preload Art	✓	boolean

We no longer need to do anything with the image, so in the Library click on the *Behaviors* tab instead. The first thing we want to do is make the milk carton "fall," i.e. move downward. Scroll through the Behaviors until you find "Move," then drag and drop it onto the window on the right:



Move
Direction: 0.0 Relative to: actor Move Type: additive
Speed: 300

Attributes

Name	Actor 1	text
Time	0.0	real
Position		point
Size		size
Rotation	0.0	angle
Color		color
Image	milk	image
Tags		text
Preload Art	✓	boolean

Load Attribute
Log Debugging State...
Move
Move To
Note
Particles
Pause Game

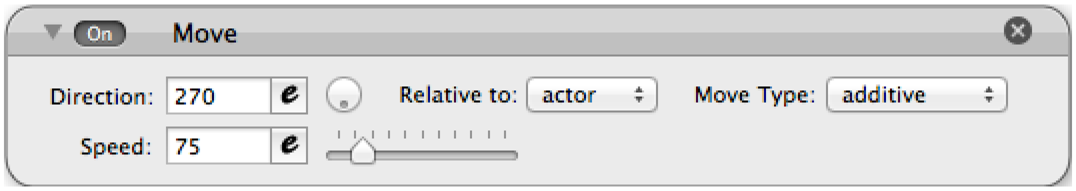
Accelerate *(Persistent Behavior)*

Specify the speed and direction (angle) of acceleration of an actor. Best used in a rule that checks for a keyboard event. Note: If the

Click Preview again. Hey, it moves!...but to the right. We need to change the direction of movement. Click the Back button, then look within the Move behavior you gave the milk carton. You'll see Direction defaults to 0.0, meaning an angle of zero degrees. Set it to 270, which is straight down. (-90 will also work; for more, see a Geometry teacher.)

Click Preview again. Yes! The milk carton moves downward.

It's awfully fast, though; remember that the player will need time to click it. Go back and change Speed to 75, or something else that feels good to you. Don't be afraid to Preview the results, then tweak Speed, then Preview, then tweak, etc. This is an important part of the design process, which will be more important once we've set the rest of the behavior. For now, your Move behavior should look like this:



Move
Direction: 270 Relative to: actor Move Type: additive
Speed: 75

Now it's time to tell the game to destroy the milk carton if it falls offscreen.

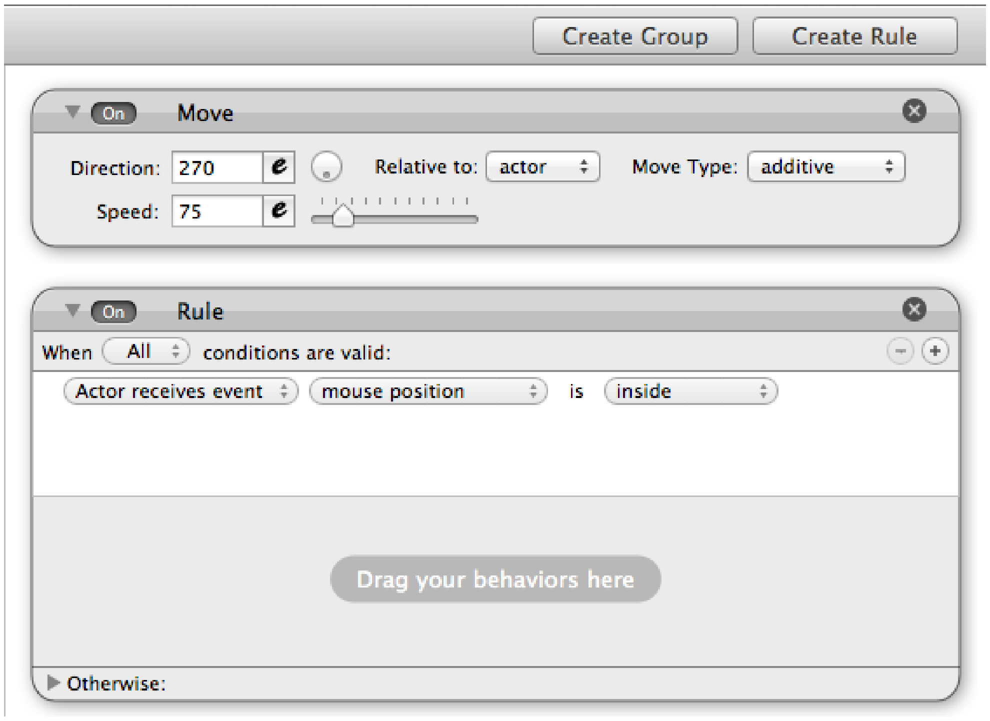
Note: Why do this? Remember that the screen is just what the player sees--it is not the limit of what GameSalad sees. When the Actor vanishes below the screen, the player may not be *able to interact with it anymore, but GameSalad still knows it's there, and keeps moving it downward in virtual space as per the behavior assigned to it. Potentially, GameSalad will move the milk carton downward forever (as long as the game runs).*

This may not seem like a big deal--and for the single milk carton, it is not--but the more things your game has to keep track of and keep updating (even if they have no relevance to gameplay anymore), the worse the game will run, just like how it's hard for us to do well if we have to focus on many things at once. As such, it's smart to destroy any out-of-bounds Actors. GameSalad may actually do this automatically at a certain distance offscreen, but it's good practice to take care of it yourself, especially if you ever move on to other programs that don't do it for you.

So, how to do this? It's very easy. We check to see if the position of the milk carton is below the bottom of the screen. GameSalad tracks the position of Actors using a standard Cartesian coordinate system, with the origin point (0,0) at the lower-left of the game screen, the x-axis running positively to the right, and the y-axis running positively upward.

In this scenario, the position of the milk on the x-axis doesn't matter; what we care about is whether its position on the y-axis falls below the bottom of the screen. At the bottom of the screen, y = 0, so anything below that will have a y-position less than zero (i.e. negative).

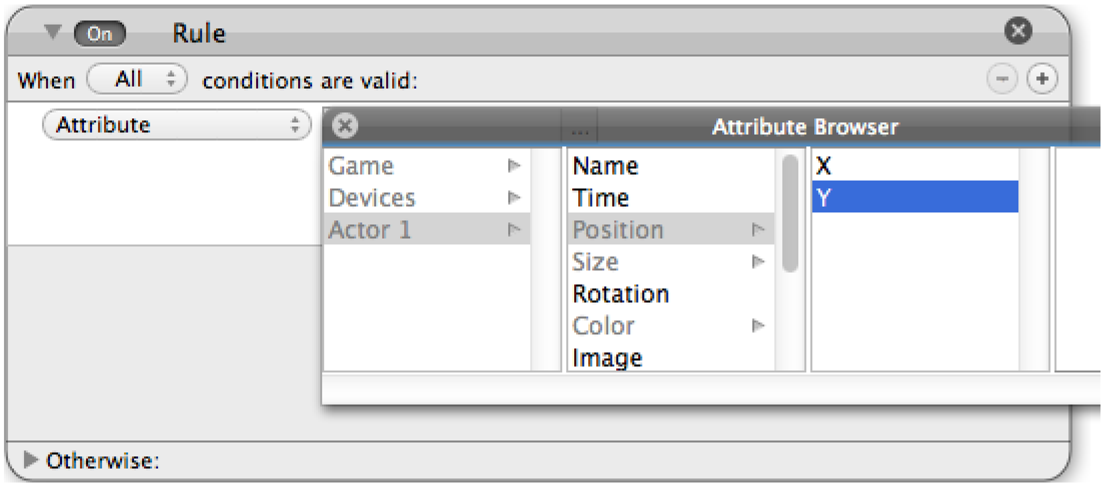
Let's try it. Previously, we gave the milk carton perpetual behavior by dropping Move directly onto the Behavior window. This time we need to use a "Rule." A Rule checks constantly to see if something is true. In the upper-right of the Behavior window, click "Create Rule":



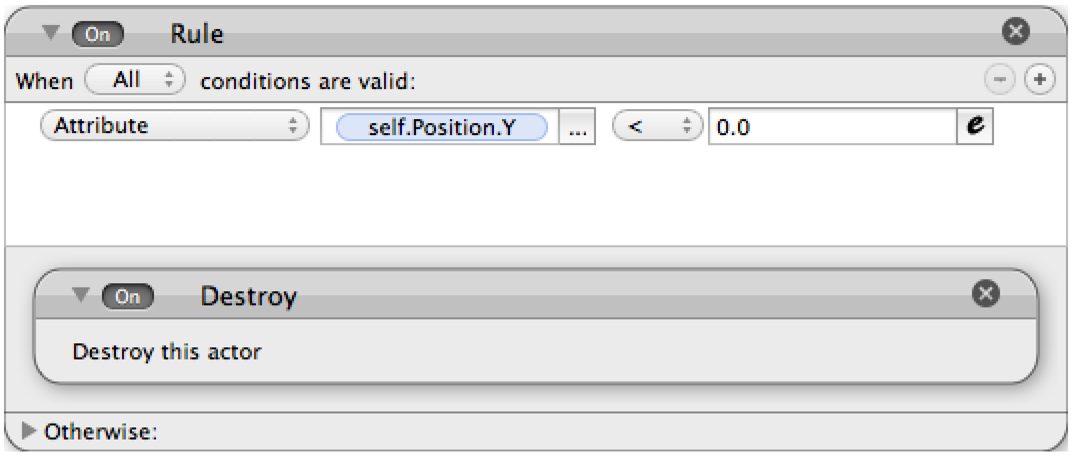
Direction: 270 Relative to: actor Move Type: additive
Speed: 75

When All conditions are valid:
Actor receives event mouse position is inside

The empty Rule appears below the Move behavior. By default the Rule checks for an "event" (usually input from the player), but we want to check one of the Actor's "Attributes," in this case y-position. Click "Actor receives event" and then choose "Attribute" from the drop-down menu. Then click in the blank box that appears to the right. A pop-up menu will appear, which includes all the available Attributes. Click to select Actor 1, then "Position," then "Y":



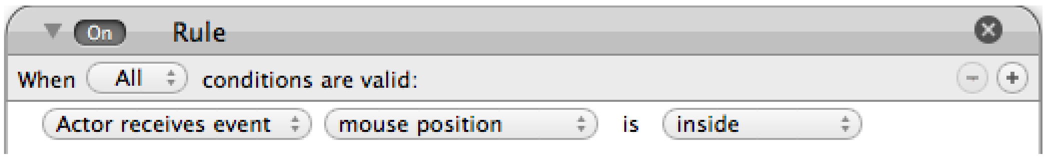
Double-click on "Y" to select it. The pop-up menu will disappear and "self.Position.Y" will be in the box. Click on the equal sign that appeared to the right and change it to the "less-than" symbol. Scroll through the Behaviors in the Library and drag and drop "Destroy" onto the Rule:



The milk carton falls, and presumably gets destroyed when it falls too far; we're doing well, but if there's no interactivity then the whole thing is just a boring animation. Let's make the milk carton jump to the top of the screen when we click on it.

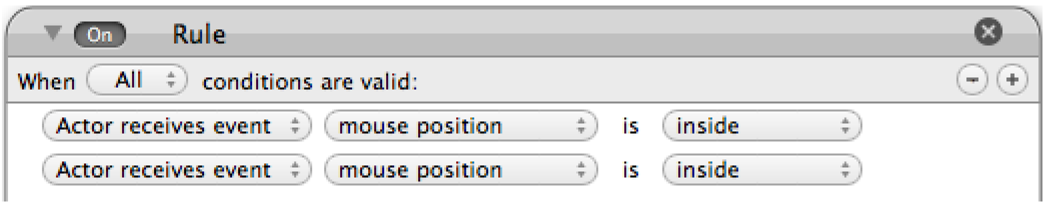
You may remember that when we clicked "Create Rule," the Rule came with default settings.

Make sure you're looking at the milk's Behavior, then click Create Rule again and examine them:



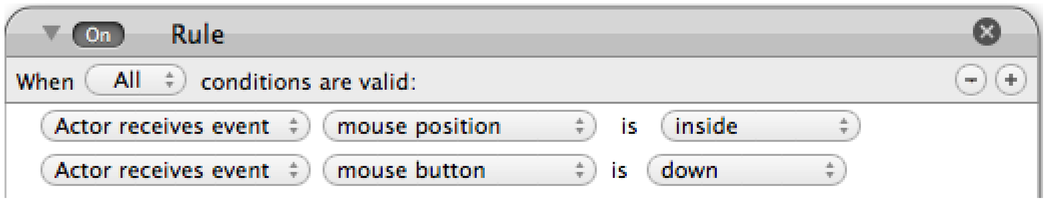
Actor receives event | mouse position | is | inside

This says that something should happen when the position of the mouse is "inside." "Inside" refers to the bounds of whatever Actor this behavior is assigned to--in this case, the milk. In other words, the player has to move the mouse onto the milk. That's good. It's not enough. We also want the player to click. Look in the upper-right of the Rule and click the plus button:



Actor receives event | mouse position | is | inside
Actor receives event | mouse position | is | inside

Oh good, now we have two things that say the same thing. Let's change the second one. Click on "mouse position," then choose "mouse button." Now we have this:



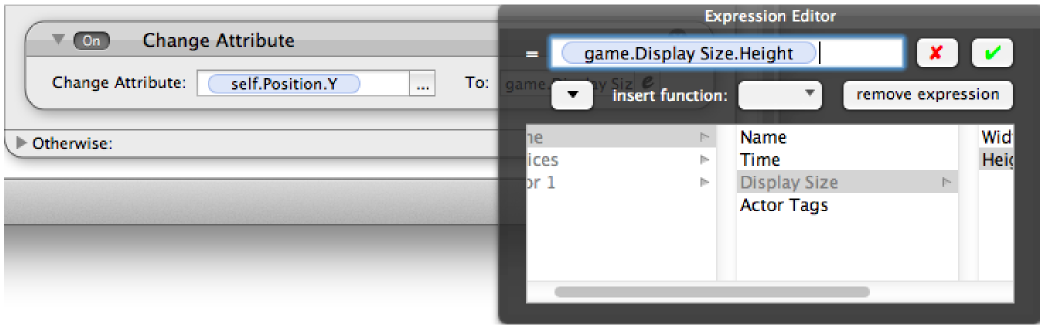
Actor receives event | mouse position | is | inside
Actor receives event | mouse button | is | down

Notice that above the first criterion, it says "when All conditions are valid." In other words, both of the following statements must be true for the Rule to take effect. The player must move the mouse onto the milk carton and then click. If the player moves the mouse onto the milk but doesn't click, or clicks anywhere off the milk, the Rule does not apply.

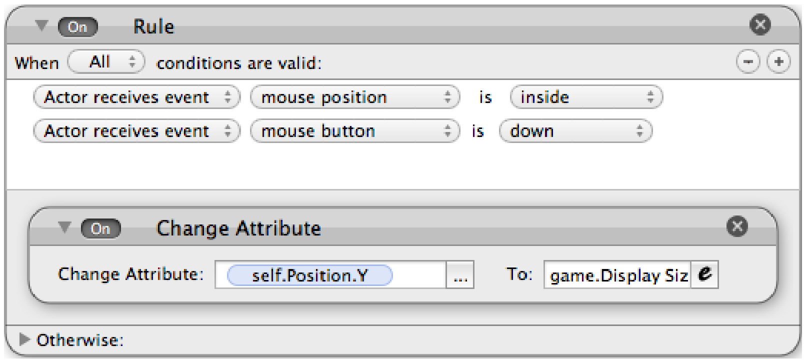
We have a rule now but no behavior in the rule. We want the milk to jump to the top of the screen. You might think to use the Move behavior again, but remember that it has a Speed. Move changes the position *incrementally* (a bit at a time). We want to change the position instantly from wherever the player clicked it to the top of the screen.

If you look through the Behaviors in the Library, you'll see one called "Change Attribute." As we know, position is an Attribute. Drag Change Attribute into the new Rule. Click the ellipsis (three dots) to the right of the blank box, then navigate to the y-position as before (Actor 1 → Position → Y) and double-click on it to select it.

Remember that y = 0 at the bottom of the screen and increases going upward. So, what does y equal at the top of the screen? We don't know. But GameSalad does. The distance between the bottom and the top of the screen is, of course, the screen's *height*--and yes, there's an Attribute for that. Click on the e symbol to the right of the "to" field's blank box to open the Expression Editor, then click the down-arrow on the left and choose "Game → Display Size → Height":



Double-click on "Height" to fill the box with it, then click the green check mark to close the Expression Editor. The complete rule now looks like this:

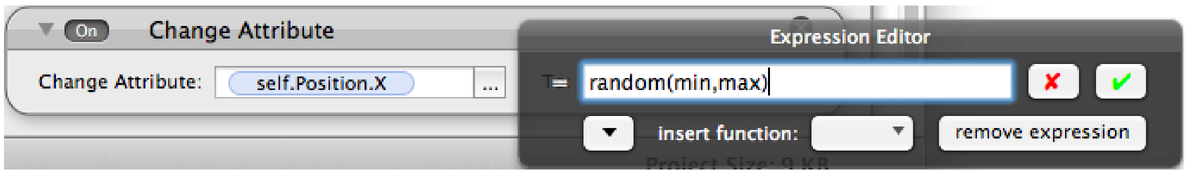


Click Preview to run the game, then try clicking on the falling milk carton. It reappears at the top of the screen! Finally, we have an actual game: click on the milk carton to keep it in play, lest it fall below the screen and disappear. Nice work!

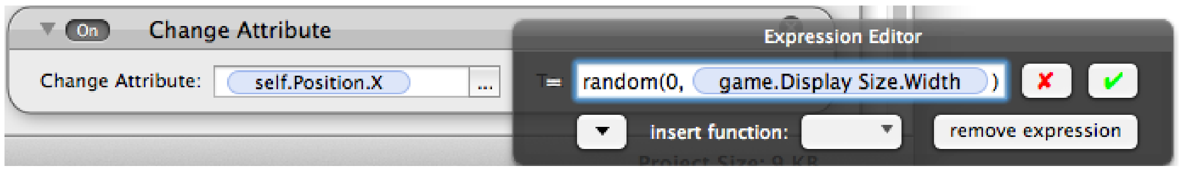
We are very close to finished; however, the game isn't very interesting right now. It's too easy, because there's only one thing to click on, and it always reappears in the same place. What if we could change where the milk reappears every time we click on it? Well...we can!

Pull up the Rule we just created for clicking on the milk. We'll use a very common technique in games: randomized positioning. It makes sense to make the milk reappear at the top of the screen each time, using the same y-position, so let's randomize its x-position. Drag Change Attribute from the Library into the Rule once again, but this time choose the "X" Position

Attribute instead of Y. Next, open up the expression editor like before, but this time, click the down-arrow to the *right* of “insert function” and click on “random.” You’ll see the following:



The “random” function requires you to choose a lower and an upper limit, and will then pick a value between them. We want the milk to appear anywhere along the whole width of the screen; can you guess what the min and max will be?



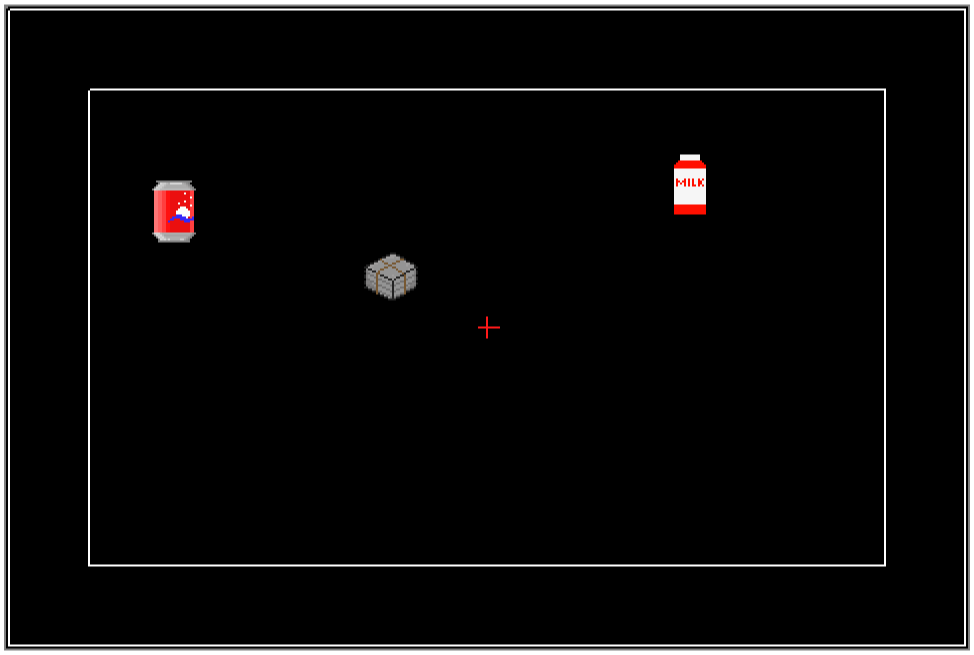
(To put in the Attribute, double-click on the word “max” in the expression to highlight it, then click our usual friend the left-most down-arrow and navigate to “Game → Display Size → Width.”)

Preview the game. Yes! Now when you click the milk it reappears at a different spot on the top of the screen each time.

As a last touch, let’s add more falling items. We could add more milk cartons, but that’s boring. Instead, create new Actors, starting with new images. To do so, navigate back to the Scene view (you can always click on the “Scenes” button at the top of the interface, then click a specific Scene). Look at the Library and click on the *Images* tab again. Return to the “sprites” folder in the Resource Files, and this time drag and drop “can01.png” and “newspaper.png” into *Images* in GameSalad. GameSalad will import them, just like it did the milk carton.
Now here’s a handy trick. Drag the can and the newspaper from *Images* and drop them into *Actors* in the Inspector. Poof! GameSalad will create new Actors using those images.

Here’s another handy trick. Double-click on the milk carton in *Actors* to open up its behavior again. Click on any of its behaviors (the selected one will become outlined in blue), then press Command+a (Ctrl+a on PC) to select all the behavior. Now, press Command+c (Ctrl+c on PC). We’ve just copied all of the milk’s behavior. Next, click the Back button, then double-click on the can in *Actors*. Finally, click in its Behavior window (which is currently blank), then press Command+v (Ctrl+v on PC). It’s magic! We’ve just pasted the complete set of milk behaviors into the can, so it behaves the same as the milk. Do the same thing for the newspaper.

Remember, we haven't put either of the new Actors into the game yet. Return to the Scene view, then drag a newspaper Actor and a can Actor into the Scene. The result should look something like this:



Click Preview. The game is now much harder! Even though all three objects share the same behavior, the fact that there are three instead of one, and that they all start off at different heights (which means they reach the bottom at different times), greatly increases the challenge.

Great job! You've made Dropcycle!

Think about how you might improve the game further. Different behavior for different objects? Falling speed that changes whenever you click an item? Different ways to interact with the recyclables?

With this tutorial you've learned the basics of game creation in GameSalad. Feel free to build this game further, work through the other tutorials in the curriculum, or even start your own project. The possibilities are limitless!