6. Mobile Art

Glen Bull, Jo Watts, and Alexis Kellam

Alexander Calder is a sculptor known for his innovative mobiles – kinetic sculptures so well balanced that the slightest air currents cause them to move and sway. One of the best-known Calder mobiles is in the atrium of the East Wing of the National Gallery of Art in Washington, D.C. This type of mobile art, such as the illustration shown below, can be emulated with a computer.

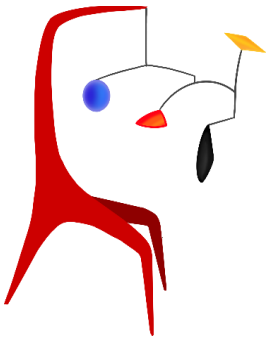
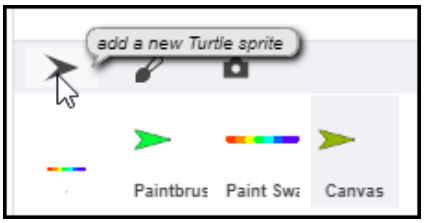


Illustration by Alexis Kellam

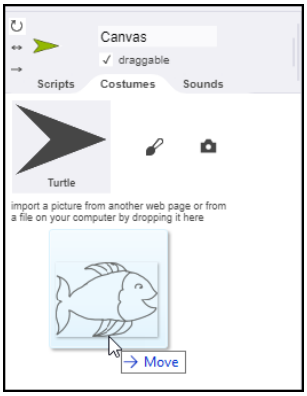
Digital prototypes developed on the computer can also be converted into art in the physical world. The animated art in this module builds upon and combines techniques from previous units.

Topic 6.1 Importing a Drawing

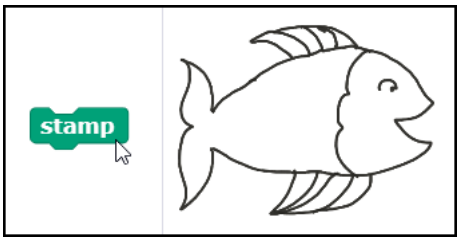
The paint application created in the previous unit provides one method of creating art and drawings using customized tools. Art drawn by hand on paper can also be photographed with a smart phone or camera and imported as the costume of a sprite. To import art or drawings, create a new sprite and name it *Canvas*.



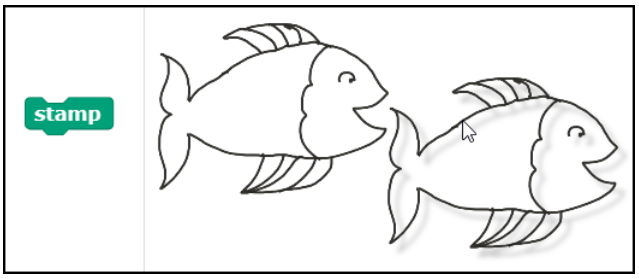
Then select the *Costumes* tab of the *Canvas* sprite and drag the image onto the *Costumes* tab.



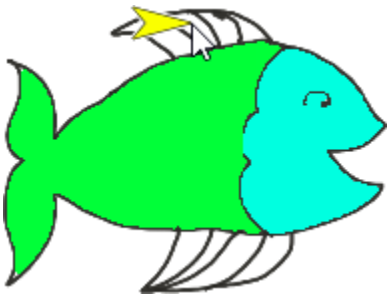
It is not possible to draw directly onto a sprite costume. However, the **Stamp** code block can be used to stamp a copy of the costume onto the stage. Since the stamped copy of the costume is a drawing, it can be filled with paint colors.



To access the stamped drawing, drag the *Canvas* sprite away from the stamped copy beneath the sprite. (If you lose track of which image is the drawing and which is the sprite, it is helpful to remember that a sprite can be moved while a drawing is a static image whose position cannot be moved.)



The drawing on the stage can now be filled with color using the paint program developed in the previous unit. (Note: the **Hide** block can be used to hide the sprite used to stamp the image.)

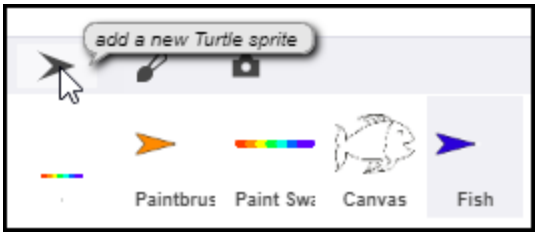


Exploration 6.1 Importing Hand-drawn Costumes

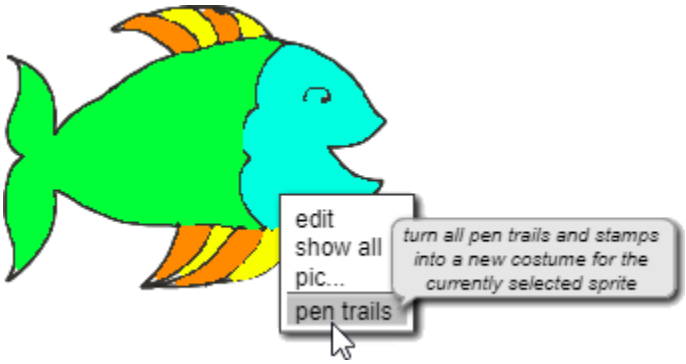
Create a hand-drawn image and import it into Snap*!* Then enhance the drawing using the paint application developed in the previous unit.

Topic 6.2 Converting a Drawing to a Costume

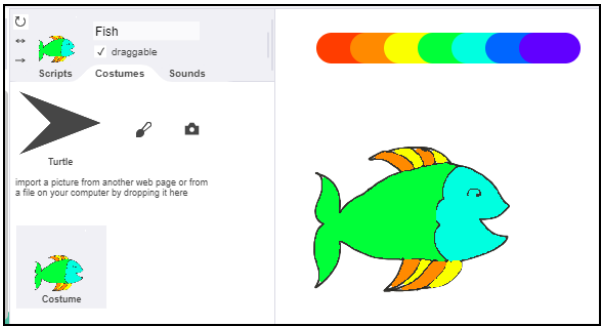
Any drawing on the Snap*!* stage can be converted into a sprite's costume. Once the drawing has been filled with color, create a new sprite. The new sprite in the illustration below has been named "Fish".



Then right-click the drawing and select the *Pen Trails* option to convert the drawing to a costume. (Note: the mouse pointer must be on a painted area in the drawing to access this option; if the mouse pointer is on an empty space that has not been filled, this option will not be available.)



The drawing on the stage, with color and enhancements added by the paint program, is now the costume of the *Fish* sprite.



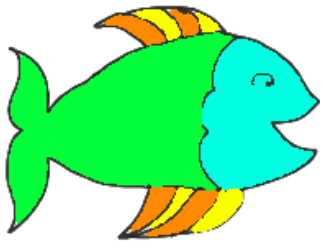
Once the drawing has been converted to a costume, the **Clear** block can be used to clear the screen if the old stamped drawing is no longer needed.

Exploration 6.2 Converting a Drawing to a Costume

Use the *Pen Trails* option to convert a drawing into a sprite costume.

Topic 6.3 Animating a Drawing

Motion blocks move a sprite, thereby animating the drawing. For example, placing the **Move 10 Steps** block in a **Forever** loop will cause the *Fish* sprite to continue moving until it moves off the right edge of the screen.



The **Go to X:** __ **Y:** __ block can be used to return the sprite to the center of the screen.

Essential Knowledge

CS Principle 3.3 Iteration

The **Forever** block is an *infinite loop*. The original address of the campus of the Apple computer company was at "1 Infinite Loop" in Cupertino, California – an example of computer humor.

Application of Essential Knowledge

In this example, the **Forever** block executes the **Move** instruction repeatedly until the *Stop* icon is clicked

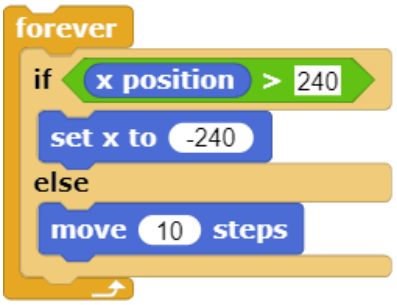
Exploration 6.3 Animating a Drawing

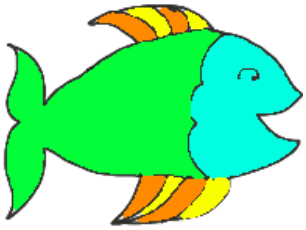
Combine a **Move** block with a **Forever** block to animate a sprite. What effect does changing the number of steps used as an input to the **Move** block have upon the motion of the sprite?

Topic 6.4 Wrapping the Animation around the Stage

To create an animated diorama with continual motion, animated sprites are wrapped around the stage so that when they exit the right side of the stage, they reappear on the left side of the stage. The X-coordinate of the right side of the stage is 240. By checking to see when the sprite's horizontal (X) position is greater than 240, the sprite can be reset to -240 (i.e., the left side of the stage) when this occurs.

This test can be combined with the **Forever** and **Move** blocks to create a continuously moving diorama that causes the fish to reappear on the left side of the stage after it exits the right side.





Essential Knowledge

CS Principle 3.3 Mathematical Expressions

An expression can consist of a value, a variable, an operator, or a procedure call that returns a value. Expressions are evaluated to produce a single value.

CS Principle 3.5 Boolean Expressions

A *Boolean value* is either true or false. In the example above, the **X Position** of the turtle is examined to see if it is greater than 240; the expression re and returns a value of *True* or *False*.

CS Principle 3.6 Conditionals

Selection determines which parts of an algorithm are executed based on a condition being *True* or *False*. Conditional statements affect the sequential flow of control by executing different statements based on the value of an expression.

In an **If … Else** code block the *first block of statements* is executed if the expression condition evaluates to true; otherwise the code in the *second block of statements* is executed.

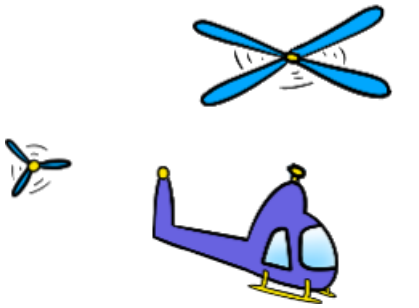
i

Exploration 6.3 Wrapping the Animation around the Stage

Refine the code for the previously animated drawing so that the animation wraps around the stage.

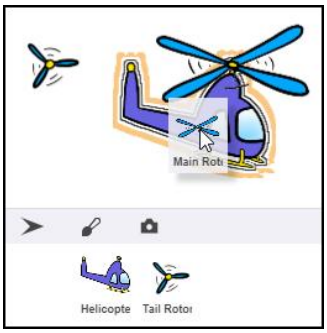
Topic 6.5 Animating Separate Elements of an Object

There are times when it may be desirable to group several sprites together. For example, in the illustration below, there are three separate sprites: the body of the helicopter, the main rotor, and the tail rotor. These elements can be grouped together so that all three components move together as one object, while still providing the flexibility for the rotors (which have their own individual script areas) to turn independently of the main body.



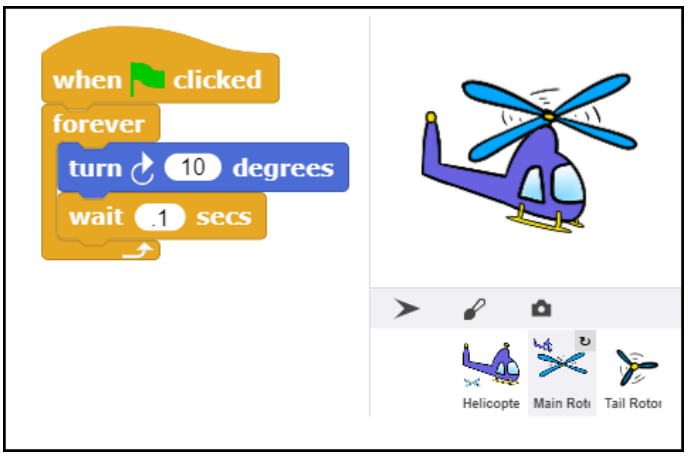
To accomplish this, first position the two objects in the desired relationship. In the example below, the main rotor has been positioned in the desired position at the top of the helicopter body.

Then drag the icon of the subcomponent – the rotor in this instance – from the sprite corral at the bottom of the stage onto the helicopter body. Once the icon is over the main component, it will be highlighted by a tan line. Release the mouse button when the tan line appears to group the objects.

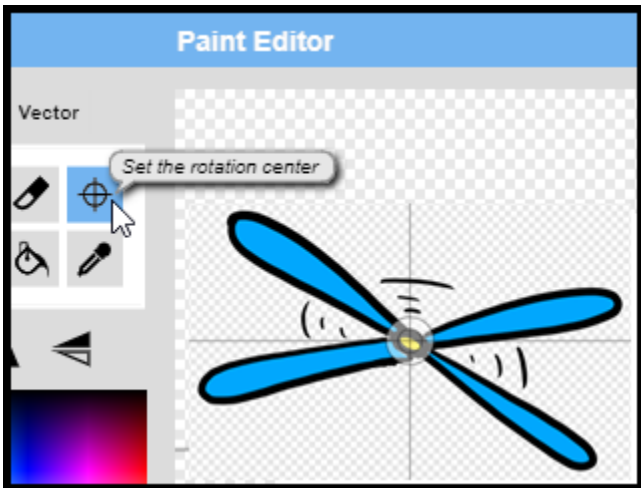


Once this is done, the icons in the sprite corral will indicate that the two parts have now been grouped together.

The following script can now be placed in the script area of the main rotor. This script will turn the rotor. The **Wait** is one-tenth of a second (0.1 seconds).

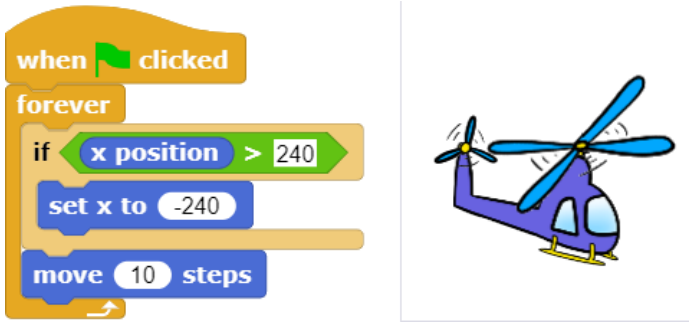


Note: if the blades of the rotor appear to be off-center as they turn, the *Paint Editor* can be used to set the rotation center of the image so that the blades rotate around the center of the object.



Repeat the grouping process for the tail rotor. Then copy the main rotor script into the script for the tail rotor. After this step is completed, clicking the green flag will cause both the main rotor and the tail rotor to rotate.

Once the rotors are turning properly, the following script placed in the script area for the body of the helicopter will cause the helicopter assembly to move across the screen while the rotors turn. When the helicopter reaches the right side of the stage, it will wrap around and reappear on the left side of the stage.



This method can be used to independently animate subcomponents within a larger grouped assembly.

Essential Knowledge

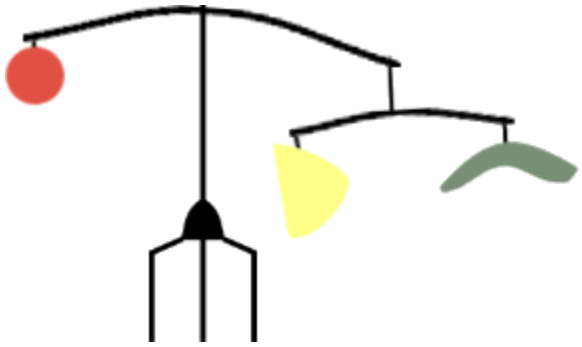
- **CS Principle 3.3** Mathematical Expressions
- **CS Principle 3.5** Boolean Expressions
- **CS Principle 3.6** Conditionals

Exploration 6.5 Animating Separate Elements of an Object

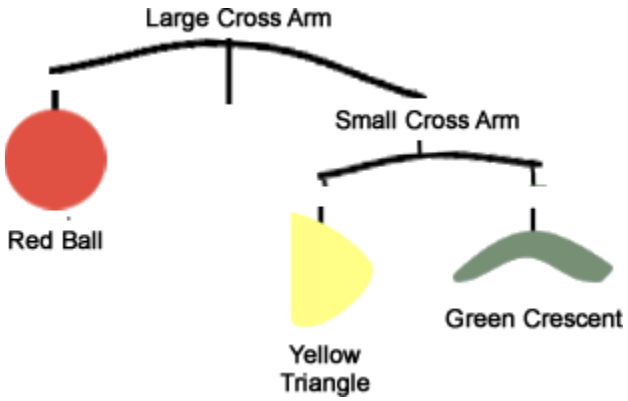
Create an animated diorama in which several sprites move independently.

Topic 6.6 Combining Elements to Simulate a Mobile Object

The method used to combine the elements of the helicopter in the preceding section can be extended to combine the elements of a simulated mobile.

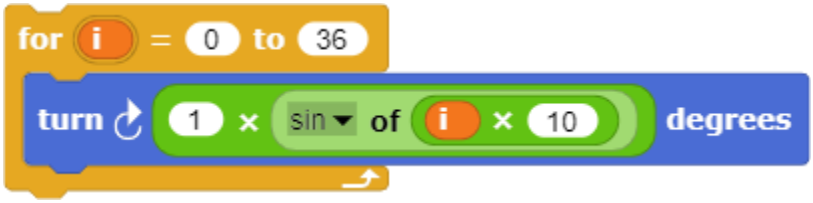


This illustrative mobile can be separated into five separate components: a red ball, a yellow triangle, a green crescent, and two cross arms (one large and one small). Each of these elements will become a separate sprite that will have its own individual script. The individual components will then be grouped into larger subassemblies in the same manner as the helicopter.

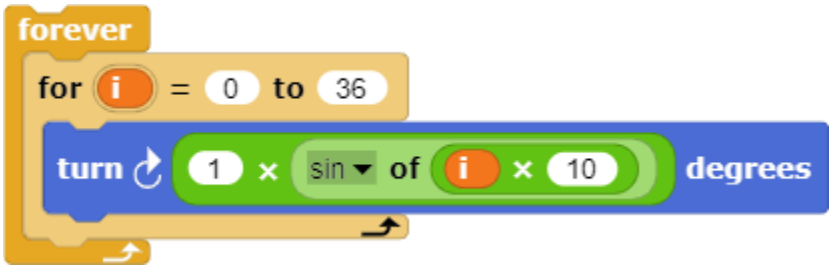


If a Turn code block is placed in the script area of the Small Crossarm sprite, it can be used to tilt the cross arm back and forth.

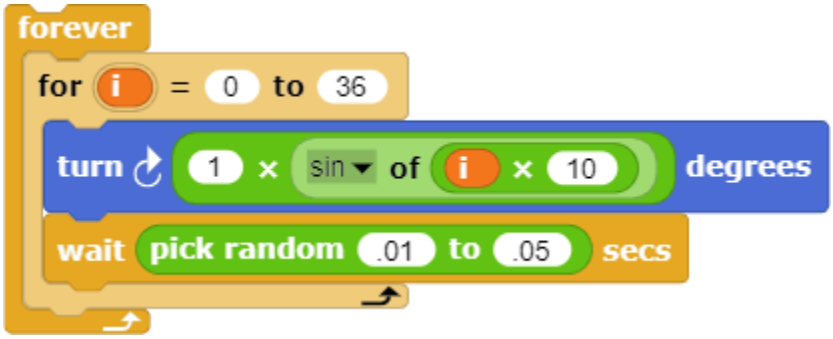
The following loop causes the crossarm to oscillate back and forth one time. The sine function describes the behavior of naturally oscillating objects. It will be explored in greater depth in future modules involving sound and music (for example, to describe the motion of a vibrating piano string).



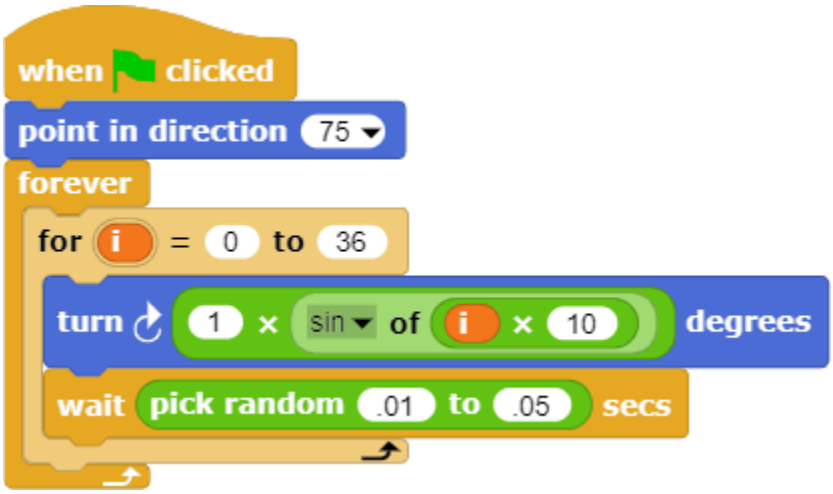
Addition of a **Forever** loop will cause the crossarm to oscillate back and forth indefinitely.



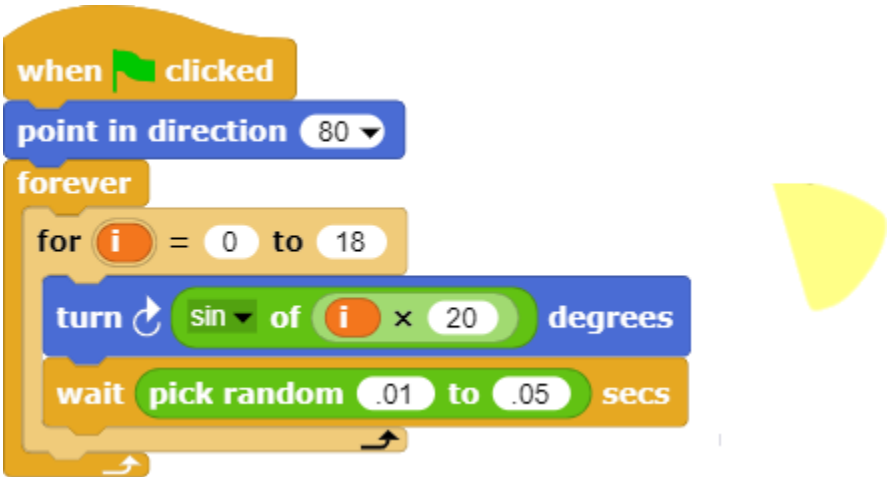
Introduction of a random wait between oscillations simulates the effect that might result from random air currents shifting as they flow past the crossarm.



Finally, addition of a green launch block will cause the crossarm script to execute when the green flag is clicked, along with scripts with green launch blocks in the script areas of the other sprites.

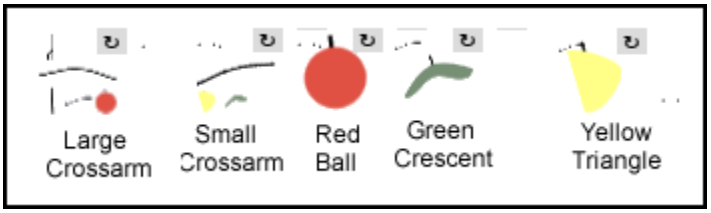


A similar rocking motion is applied to the yellow triangle. Since the yellow triangle is hanging beneath the small crossarm, its oscillation is constrained in a different way.



In this manner two scripts – one variant that controls the motion of the crossarms and a second variant that controls the motion of the red ball, yellow triangle, and the green crescent – are used to create oscillations in each of the individual elements.

Once scripts have been developed for the script areas of each of the five sprites that cause each one to move individually, the yellow triangle and the green crescent can be grouped with the small crossarm. The grouped small crossarm assembly and the red ball can then be grouped with the large crossarm.



Once the groupings have been completed, each of the elements will oscillate in a manner that is controlled by the individual script for that sprite, while moving as a whole within the larger assembly.

Exploration 6.6 Combining Elements to Simulate a Mobile Object

Create a simulation of a mobile with separate scripts that animate each element of the mobile. Then construction a physical model of the mobile. Do the elements of the mobile move in a way that resembles the simulation? In what ways does the movement of the physical model and the digital simulation differ?