# GP Tips and Tricks

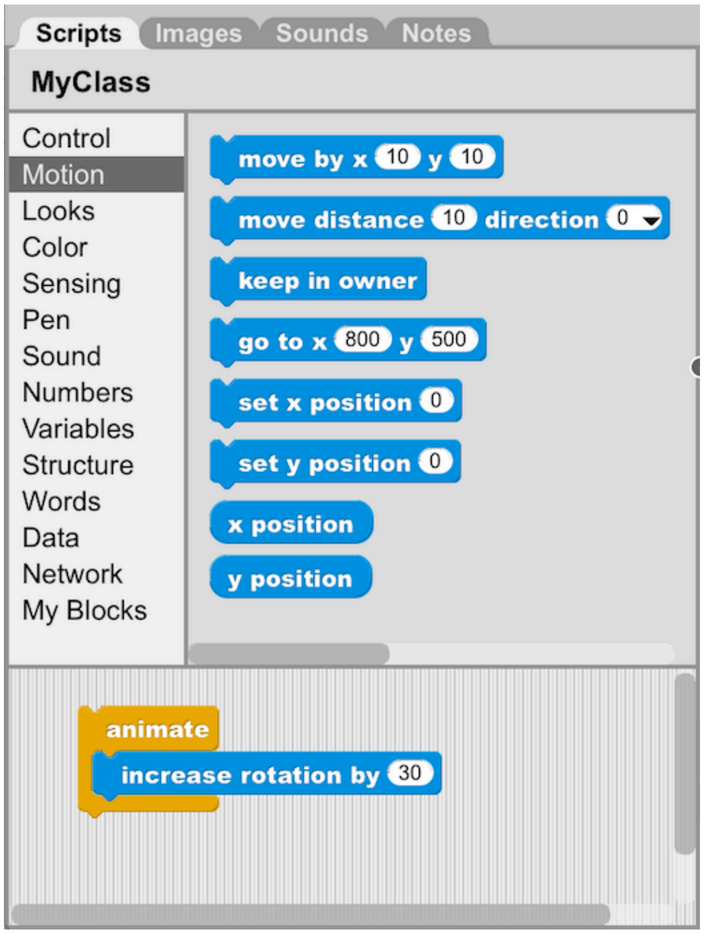
August, 2016

## Starting GP

1. (first time only) Extract the GP folder from the .zip file you downloaded
2. Open the GP folder (it will be called something like GP-alpha-055)
3. Double-click the file named “gp” (Mac/Linux) or gp.bat (Windows)

On a Mac, you may get a warning dialog the first time you run a newly downloaded version of GP. Depending on your security settings, you can either click the “Open” button to proceed or choose “open” from the context menu on the “gp” file. Linux users may need to run the “gp” shell script from a terminal window.

## The Scripter



The Scripter is where you create and view scripts. To create scripts, drag blocks from the blocks palette (top right) into the scripts area (bottom) and assemble them into a script. Select a category (Control, Motion, etc.) to see the blocks in that category.

## Arrows

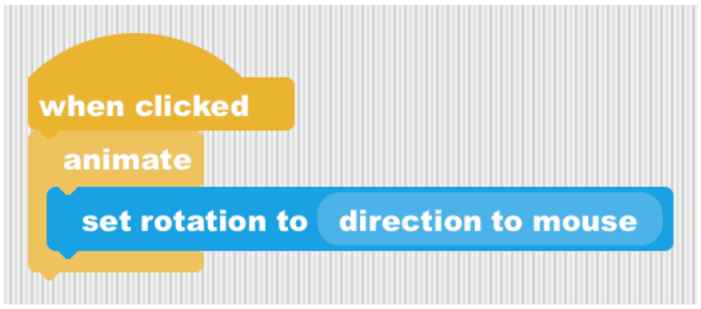
By default, an arrow points from the scripter to the object being viewed/scripted. When you run blocks in the palette or scripting area, they act on the object being viewed, so you can try out blocks and scripts to see what they do. If you view a different object, the arrow switches to that object. You can use the “show arrows” switch in the menu bar to hide or show arrows.

## Useful Gestures

alt-click (option-click on Mac) - view the option-clicked object in the scripter
right-click (control-click on Mac) - show the context menu, if any
shift-right-click (shift-control-click on Mac) - show object manipulation menu
escape key (esc) - stop all scripts and sounds and exit presentation mode

The “show all” command in the context menu for the stage (the content area to the right of the scripter) brings back objects that have gone off the screen. The “normal size” command sets the stage to its normal size.

## Blocks



GP scripts are built out of blocks that snap together. *Command blocks* have square ends and a notch in the top. They snap together to form a stack of blocks that are executed from top to bottom. Some command blocks have *input slots* – for example, the “set rotation to” block has an input slots that determines the object’s orientation. Some command blocks, such as “animate” allow you to nest a stack of blocks inside them.

*Reporter blocks* have rounded ends and can be inserted into the argument slots of Command blocks to provide values.

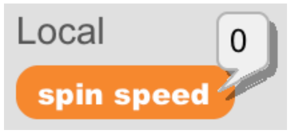
Finally, *hat blocks* have rounded tops, and are used to start scripts in response to events such as mouse clicks.

## Variables and Monitors

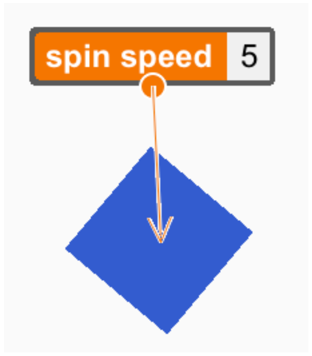
A variable allows an object to remember a value so that it can be used in scripts. For example, a “speed” variable might determine an object’s speed in a script that makes the object move. Other scripts might increase or decrease the number stored in the speed variable, thus changing the speed of the object.

The “Variables” category shows the variables for the scripted object. It's helpful to give variables descriptive names like “speed.” The Variables category has a reporter block for each variable. The “set to” and “increase by” blocks are used to set the variable to a new value or increase it by some amount. Variables can also store any kind of value used in GP: numbers, words, booleans, images, sounds, colors, lists of things, etc.

If you click on a variable's reporter, a small talk bubble appears showing its value at that moment:



To get a readout for a variable (or any reporter block that doesn't have inputs), click and hold on the block, then select “monitor” from the menu. This creates a readout that updates continuously. If “show arrows” is on, an arrow points from the monitor to the object to which it refers. This is helpful since there may be several monitors with the same variable name that refer to different objects.



Monitors can help you see how a variable or reporter changes over time. They can also be used as readouts in a finished project.

## Deleting

Blocks and monitors can be deleted by dropping them onto the blocks palette area of this scripter. (You can think of this as putting them back where you got them.) To delete other objects, select “delete” in the meta-menu of the object or the object's thumbnail in the “Instances” area below the stage.

## Instances and Classes

You can make more copies of a given type of object using the ⊕ button in the “Instances” area. Such copies are called *instances* in GP. Each instance can move, rotate, scale, change color independently of the other instances, but all instances of a class share the same scripts. Thus, if you add or change a script while viewing one instance, all the other instances automatically get that change. This makes it easy to fix a bug or add a new script even after you've already created many instances.

The scripts shared by a set of instances are stored in an object called a *class*. The class of the object you are currently editing is shown selected in the the “Classes” list. You can use use the Classes list to switch classes. The ⊕ button at the top of the list allows you to create new classes (i.e. new kinds of objects). The right-click (context) menu on a class name allows you to rename or delete that class. (There is no “undo” so be careful when deleting classes!)

Every instance of a given class has a set of *instance variables* with the same names. If you add or delete a variable while viewing any instance, all of the other instances will see that change. However, each instance has its own values for its variables. This is a powerful way to customize instances. For example, all instances of a “RaceHorse” class might have a “speed” variable, but different instances can have different values for that variable, allowing the horses to move at different speeds.

## Projects

GP projects are saved as files ending in “.gpp” in the “projects” folder within the GP folder. Use the buttons in the GP menu bar to open, save, or create a new project.

## Presentation Mode

You can present your projects in full-screen mode by clicking the “Present” button. Press the “esc” key to exit presentation mode (and also stop all scripts and sounds).

## Other

GP is short for “General Purpose Blocks Programming Language.”

This pre-release alpha version runs only on Mac, Windows, and Linux. Eventually, we hope GP will run on Android, iOS, and in web browsers as well.

GP is being developed by the Human Advancement Research Community (HARC), part of YC Research. GP has not yet been released and is not supported. You are welcome to keep this pre-release version for your own personal use, but please do not share or distribute it.