GAVisTool
User Manual
Version 0.1

Spencer T. Parkin

July 7, 2012

This is the manual for the GAVisTool program. It is the quickest way to learn how to use this software, provided the reader is already familiar with geometric algebra and the conformal model. For more information on these topics, see "Geometric Algebra for Computer Science" by Dr. Leo Dorst, et. al. Another good source of information on these topics can be found in Dr. Christian Perwass's "Geometric Algebra with Applications in Engineering."

1 The Console Window

When the GAVisTool program comes up, you are presented with two windows: the console window and the canvas window. Ignoring the canvas window for the moment, the console window is where you will interact with the GA environment by inputting expressions in the text box at the bottom, then seeing the evaluation of these expressions as output in the main part of the window.

1.1 The GAVisTool Language

A design goal of the GAVisTool software was that of not requiring the user to learn a language unique to the software. On the contrary, the tool better serves its user as a means to learning the language of geometric algebra and mathematics in general. With that in mind, the language recognized by GAVisTool deviates from the language of mathematics only where unique

features of the GAVisTool software are concerned. In fact, if you're familiar with basic mathematics, then you already know the entire syntax of the GAVisTool language. Specificly, its syntax consists of left and right unary operators, infix binary operator notation, and function call syntax. That's it. It's not any more or less complicated than that. All that remains then for the user to learn is what unary and binary operators are supported, what functions are available, and what types of numbers are the subject of all these things.

1.1.1 Numbers

The concept of number is abstracted in the software to mean whatever is applicable to current environment. As of this writing, the only environment presented to the user is the GA environment, and so the numbers in this context are simply the elements of any geometric algebra defined over the field of real numbers. The specific GA being used depends on what basis vectors the user chooses to mix into his or her expressions. The following table lists the currently recognized constants in the GA environment.

Console	Math Symbol	Description
no	o	Null vector at origin
ni	∞	Null vector at infinity
e0	e_0	Unit X-axis Euclidean vector
e1	e_1	Unit Y-axis Euclidean vector
e2	e_2	Unit Z-axis Euclidean vector
I	$e_0 \wedge e_1 \wedge e_2 \wedge o \wedge \infty$	Unit psuedo-scalar of 3D CGA
i	$e_0 \wedge e_1 \wedge e_2$	Unit psuedo-scalar of 3D EGA
E0	$e_1 \wedge e_2$	Unit YZ-plane Euclidean 2-blade
E1	$e_2 \wedge e_0$	Unit XZ-plane Euclidean 2-blade
E2	$e_0 \wedge e_1$	Unit XY-plane Euclidean 2-blade

The vectors e_0, e_1 and e_2 form a right-handed orthonormal basis for 3-dimensional Euclidean space.

You may enter any one of the above variable names into the console to see it recognized by the GA environment. The first five symbols given in the table form the basis for the geometric algebra used by the conformal model of 3-dimensional space. The remaining five variables are provided for convenience.

Also recognized by the GA environment are scalar literal values. These are bits of text such as "3.14159" and "0.12345" and so on.

1.1.2 Unary and Binary Operators

Many unary and binary operators are recognized by the GA enviornment, but the first of note are simply those that will allow us to generate linear combinations of the basis vectors given in the previous section. This is done just as you would expect. For example, try inputting the following expressions. ...

1.1.3 Functions

1.1.4 Variables

When the GA environment encounters a named number that it doesn't recognize as a built-in constant, it treats it as a variable, and remembers variable values. To store a value (a number) in a variable, use the assignment operator.

2 The Canvas Window

The console window by itself is useful for making calculations in geometric algebra, but the main purpose of GAVisTool is to provide a way to visualize the results of your mathematical expressions. This is where the canvas window comes in.

2.1 Bind Targets

Bind targets are things that appear in the canvas window and that are bound to a variable in the GA environment. The type of bind target and how its bound to the variable determine how that bind target will interpret the value of the variable to which it is bound. For example, if we have a variable named sph that we would like to visualize as a conformal sphere, then we can input the expression bind_sphere(sph) to create and bind to it a bind target in the canvas window that is designed to interpret sph as a conformal sphere. Once bound, any changes we make to the value of the variable sph will be reflected as changes to the way the sphere is manifested in the canvas window by the

bind target. Conversly, any changes we make to the bind target (a sphere in this case) will be reflected as changes to the value of the variable sph in the console window. (We'll go over canvas interaction in the next section.)

Another type of bind target is a user interface. As a single element of GA may be characterized by a number of properties, these properties may be manifested through a simple user interface. These interfaces appear as floatling control panels that can be docked about the canvas window. As of this writing, the only currently available bind target interface is a scalar interface which can be bound to a scalar (real number) variable in the console window.

The following table lists the currently available bind target functions.

Function	Description
bind_point	Bind a variable to a conformal point
bind_sphere	Bind a variable to a conformal sphere
bind_circle	Bind a variable to a conformal circle
bind_pointpair	Bind a variable to a conformal point-pair
bind_line	Bind a variable to a conformal line
bind_plane	Bind a variable to a conformal plane
bind_flatpoint	Bind a variable to a conformal flat-point
bind_scalar_iface	Bind a variable to a scalar interface

Left out of the table for ease of reading were functions such as bind_dual_point and bind_dual_sphere. These interpret elements of GA as dual points and dual spheres, respectively. A dual version of all bind target functions for conformal geometries is provided.

Also left out of the table was a function named bind_inferred_geo. A bind target for a specific conformal geometry assumes a certain form for elements of GA it is given to interpret. The inferred geometry bind target, however, analyzes a given element first to determine what type of conformal geometry it may represent, then interprets it as that geometry. As of this writing, this feature has not yet been implemented.

2.2 Canvas Interaction

Just as with the console window, the canvas window too provides both output and input. Specifically, through the canvas window the user can interact with the set of currently created bind targets. Continuing with the conformal sphere example in the previous section, right-clicking on this sphere in

the canvas window will select it. You can then move the sphere in a plane orthogonal to the viewing direction by holding the CTRL key and left mouse button down while dragging the mouse in the canvas window. Holding the CTRL and SHIFT keys down while moving the mouse wheel will rotate the selected bind target on an axis horizontal and perpendicular to the viewing direction, and going through the center of the bind target's geometry. (Notice that conformal spheres don't rotate, because they are invariant under rotations about their center.) Without the SHIFT key down, that same action will scale the bind target's geometry up and down.

Right-clicking on no bind target's geometry unselects the currently selected bind target, if any.

2.3 Canvas Camera Control

Since geometry movement control is based on the camera orientation, we must have a way of moving the camera around. We would also like to be able to look at the scene from different perpsectives.

Holding the left mouse button down while dragging the mouse in the canvas window allows you to rotate the viewing location about the focal point. Doing so while the ALT key is down allows you to pan the camera in a plane orthogonal to the viewing direction, which also changes the focal point. The focal point is an invisible point in space in front of the camera. If a bind target is selected, then hitting the SPACE bar will set the focal point of the camera to the center of the bind target's geometry. Subsequent rotations of the camera about the focal point allow you to then view the bind target's geometry from different angles.

3 Constraints

The GAVisTool software maintains a set of constraints between the current set of bind targets. Like bind targets, until you actually create a constraint, there are none maintained by the system. A constraint names one or more bind targets as its set of inputs and one or more bind targets as its set of outputs. When a constraint executes, it pulls values from the variables bound to its input bind targets, performs some sort of calculation and/or algorithm on this data, and then pushes result values to the variables bound to its output bind targets. A constraint is schedualed to execute when it is

detected that one or more of its inputs has changed.

One can imagine that with multiple constraints maintained by the system, these form a dependency graph as one constraint's set of inputs may overlap with another contraint's set of outputs. Indeed, when a contraint executes, it may cause other constraints to execute as well. There is no need to worry about creating circular dependencies. When a single bind target is changed, an entire chain of constraints is schedualed for execution before actually being executed. If a constraint can't be schedualed for execution, (due to a circular dependency), it is simply not executed. This schedualling failure is not necessarily a design flaw on the part of the author of the constraint system. When rigging together a bunch of bind targets with constraints, it is not uncommon to constrain two bind targets to one another using two mutually dependent constraints.

As of this writing, there is currently only one type of constraint that can be created, but it is a powerful one. It is called a formulated constraint.

4 Scripts