Hartley Hyde
cactus.pages@internode.on.net

The centres of a triangle

This is the first of two articles which describe how to use *JavaSketchPad* to explore the centres of a triangle. This introductory exercise is suggested in the *GSP Workshop Guide*.

Your students can use *JavaSketchPad Interactive Geometry* (JSP) at home at no cost. They are likely to impress their parents with their enthusiasm for geometry and all they need on their computer is a Java enabled web browser and a suitable editor. They do not need to be on-line to use *JavaSketchPad*

The **start.html** code shown, here in red, is ordinary html code which generates the web page shown on the next page. It describes a simple 1×3 table. The top cell displays a heading and the bottom cell tells the user how to drive the contents of the second cell.

The code for the second cell is shown in black text and consists of a Java applet that directs the constructing of the geometric figure that appears in the middle cell. You can explore this and other models at
http://users.on.net/~hhyde/cactus/jskpd.

If you are using Windows you may need to enable Java by visiting www.java.com.

You can download the JSP Applet from
http://www.dynamicgeometry.com/
JavaSketchpad/Download_Center.html
as a zip file. The unpacked JSP folder can be distributed to students who do not have the internet. Unless you are a keen Java programmer you may decide to regard the contents of the JSP folder as a "black box"—it interprets any instructions from the line
`<applet code="GSP.class" codebase="jsp"` down to `</applet>` and builds the geometric model that the numbered instructions describe.

Users must place the JSP folder inside the project folder from which they will launch their html files. The code is typed into a suitable editor and saved into the project folder as **start.html**. To avoid typing errors, it may be easier to copy the **start.html** code from my

start.html

```
<head>
 <title>Start</title>
</head>
<body>
 <table width=640 border="0" bgcolor="#D0D0D0">
  <tr>
   <td>
    <p>&nbsp;</p>
    <h1 align="center">Start</h1>
    <p>&nbsp;</p>
  </td>
 </tr>
 <tr>
  <td align="center">
<applet code="GSP.class"  codebase="jsp"
archive="JSP4.jar"
width="600" height="480"
align=center>
<param name=BackRed value=240>
<param name=BackGreen value=240>
<param name=BackBlue value=240>
<param name=LabelSize value=24>
<param name="Construction" value="
{1} Point(470,150) [black,label ('A (drag)')];
{2} Point(350,450) [black,label ('B')];
{3} Point(100,60) [black,label ('C')];
{4} Polygon(1,2,3) [white];
{5} Segment(1,2) [thick,black];
{6} Segment(2,3) [thick,black];
{7} Segment(3,1) [thick,black];
 ">
Sorry, this page requires a Java-compatible web
browser.
</applet>
   </td></tr>
  <tr>
   <td><p>&nbsp;</p><blockquote>
     <p>The three points A, B and C define a
           triangle.</p>
     <p>The size and shape can be changed
           by dragging any of the points A, B or
C.</p>
    <p>&nbsp;</p>
    </blockquote>
  </tr>
 </table>
</body>
</html>
```

user.on.net website and distribute it to students. The only items of mathematical significance are the numbered instructions.

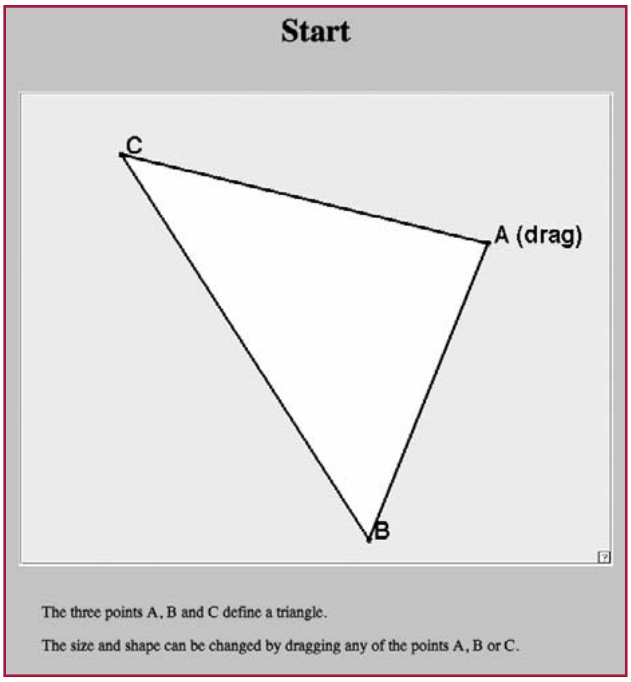
Most modern wordprocessors are unsuitable to use as editors because they attempt to open an html file as a web page instead of code. Sometimes it is possible to copy the source file from a browser window and paste it into a word-processor. After editing, the code can then be saved as a text file .txt and then manually changed from .txt to .htm. This is very messy and time consuming. A dedicated text editor is far easier to use.

Ideally we need a screen large enough to place the browser and the editor windows next to each other. A degree of overlap still works well because we have to click in the window we need to activate anyway.

If your students use *Internet Explorer* from a *Windows* platform they can view source code by choosing **View–>Source**. This opens the html file using *Notepad* and the code can be modified and saved as an html file. Each time they edit the source file (in this case **start.html**) they can save the file, click to activate the browser window and then click the refresh icon at the top of the browser screen. The browser will then read and interpret the updated file.

On old Macintosh computers *SimpleText* could be used in much the same way as *Notepad*. After exploring a new Macintosh with the usual selection of *iWorks* and *iLife* applications I couldn't find any software that can be used as a simple html editor. Even the new version of *TextEdit* opens html files as a webpage instead of source code. It is possible to drop html fragments into *iWeb* but it crashes if the code includes Java. *Safari* users can download *BBEdit*, which is quite expensive. Alternatively Macintosh users can install the *FireFox* browser and use *Ñandú* as the editor. Both of these downloads are free.

When your students open the **start.html** file using a Java enabled browser such as *Internet Explorer*, *Safari* or *Firefox* they should find that the page looks like this:



Start

The three points A, B and C define a triangle.

The size and shape can be changed by dragging any of the points A, B or C.

It is possible to manipulate this figure in the same way that we use GSP or Cabri. However, to add to the figure, instead of reaching for a convenient tool icon, we need to write more numbered instruction lines. That is why a simple editor is so important.

I gave my class a JSP folder, a **start.html** file and a printout of the *JavaSketchPad Construction Grammar* which I have saved at http://users.on.net/~hhyde/cactus/jskpd as a .pdf. After a few nights of exploration students were able to build quite sophisticated dynamic geometry models.

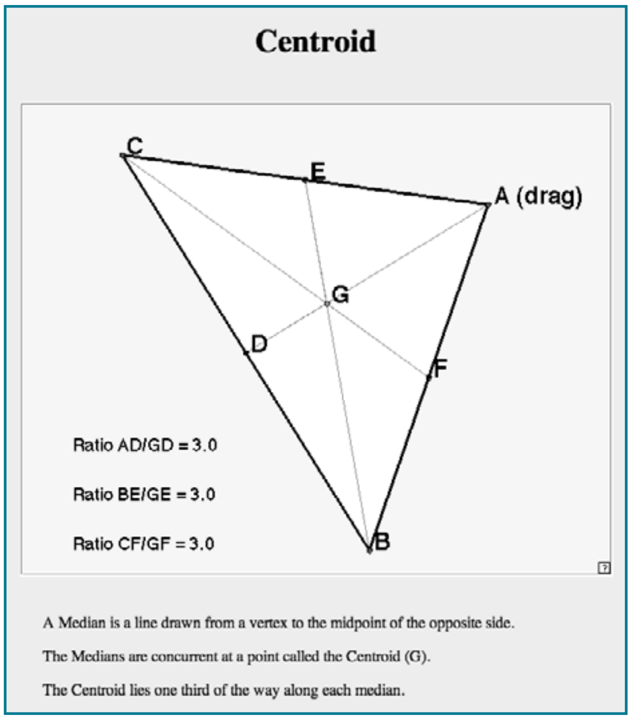
Our class studied the instructions in the **start.html** code. Items {1}, {2} and {3} define three points based on a coordinate system that counts from the top left-hand corner of the JSP screen. We changed the coordinates of these points and observed what happened.

Instruction {4} defines a white triangle that identifies the original triangle more clearly after many construction lines have been added. Instruction {5} Segment(1,2) [thick,black]; describes a thick, black line segment (AB) that joins the points defined by instructions {1} and {2}. Instructions {6} and {7} are similar.

I sent my students home to add instructions to find the centroid of the triangle.

The students had no trouble defining the midpoints D,E and F and joining these to the opposite vertices. Exploring the different types of intercepts can be tricky when curves are involved, but it isn't hard to find the intercept of two medians to find the centroid (G).

Instructions {15}, {16} and {17} identify the segments DG, EG and FG so that the ratio of these lengths to the corresponding medians can be calculated. This allows the instruction
{18} Ratio/Segments (11,15,50,350,'Ratio AD/GD = ')
to calculate the ratio of the length AD, defined by {11}, to the length GD, defined by {15} and give the answer at coordinates (50, 350).


Centroid

Ratio AD/GD = 3.0

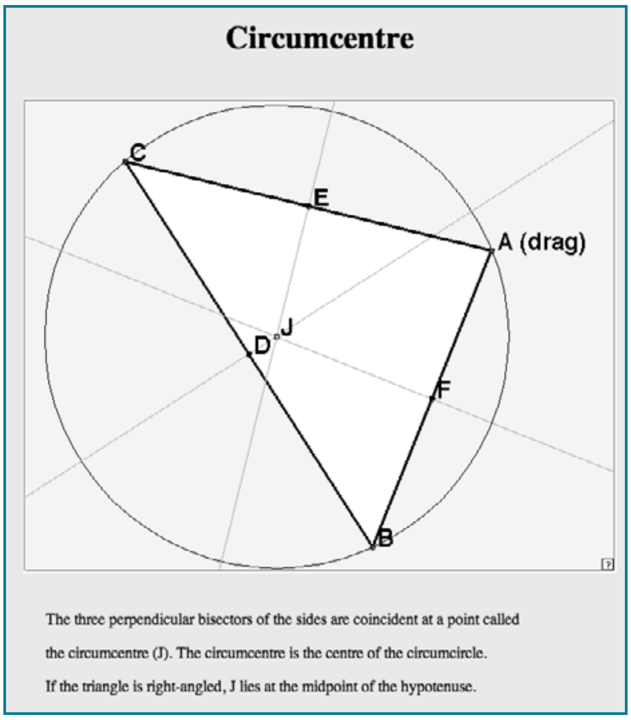
Ratio BE/GE = 3.0

Ratio CF/GF = 3.0

A Median is a line drawn from a vertex to the midpoint of the opposite side.

The Medians are concurrent at a point called the Centroid (G).

The Centroid lies one third of the way along each median.


Circumcentre

The three perpendicular bisectors of the sides are coincident at a point called the circumcentre (J). The circumcentre is the centre of the circumcircle.

If the triangle is right-angled, J lies at the midpoint of the hypotenuse.

From centroid.html

```
{1} Point(470,100) [red,label ('A (drag)')];
{2} Point(350,450) [red,label ('B')];
{3} Point(100,50) [red,label ('C')];
{4} Polygon(1,2,3) [white];
{5} Segment(1,2) [thick,black];
{6} Segment(2,3) [thick,black];
{7} Segment(3,1) [thick,black];
{8} Midpoint(6) [black,label ('D')];
{9} Midpoint(7) [black,label ('E')];
{10} Midpoint(5) [black,label ('F')];
{11} Segment(1,8) [green];
{12} Segment(2,9) [green];
{13} Segment(3,10) [green];
{14} Intersect(11,12) [green,label ('G')];
{15} Segment(8,14) [green];
{16} Segment(9,14) [green];
{17} Segment(10,14) [green];
{18} Ratio/Segments (11,15,50,350,'Ratio AD/GD = ');
{19} Ratio/Segments (12,16,50,400,'Ratio BE/GE = ');
{20} Ratio/Segments (13,17,50,450,'Ratio CF/GF = ');
```

From circumcentre.html

```
{1} Point(470,100) [red,label ('A (drag)')];
{2} Point(350,450) [red,label ('B')];
{3} Point(100,60) [red,label ('C')];
{4} Polygon(1,2,3) [white];
{5} Segment(1,2) [thick,black];
{6} Segment(2,3) [thick,black];
{7} Segment(3,1) [thick,black];
{8} Midpoint(6) [black,label ('D')];
{9} Midpoint(7) [black,label ('E')];
{10} Midpoint(5) [black,label ('F')];
{11} Perpendicular(5,10) [cyan];
{12} Perpendicular(6,8) [cyan];
{13} Perpendicular(7,9) [cyan];
{14} Intersect(11,12) [cyan,label ('J')];
{15} Circle(14,1) [black];
```

If students get confused, it is a good idea to give each construct a different colour to debug the code. When the **centroid.html** code is understood, it is easy to modify the instructions to construct a circumcentre.

The code is almost identical except that perpendiculars are constructed at the midpoints instead of medians. Constructing the circumcircle {15} provides a good check on the accuracy of the model.

Students adapt to this system quickly and they soon discover features of the instruction grammar that enhance their models. The html files that they create can be added to their private web pages. Nothing drives student achievement quite so effectively as an opportunity to show their work to the world.

JSP provides a simple macro level introduction to programming that encourages further exploration. Students find the task enjoyable and happily take files home to keep working—possibly using a different platform.

Some of you are worried about the cost and availability of software for any computers which may be supplied following the election promises. This free dynamic geometry software will run on any Java enabled browser, running under any operating system.