



© 1997–2004, Millennium Mathematics Project, University of Cambridge.

Permission is granted to print and copy this page on paper for non-commercial use. For other uses, including electronic redistribution, please contact us.

January 1999

Staff room

Geometer's corner

by Mike Pearson



If you'd like to contribute to "Geometer's corner" please write to PASS Maths using [Any comments?](#).

Paul Blythin introduced us to the Geometer's Sketchpad in our [last Geometer's corner](#). If you haven't already done so, then do read his article and follow his instructions for downloading and running the free trial version of the GSP package. Alternatively, just go to the [download page](#) and fetch a copy. The GSP requires a Windows or Macintosh machine.

As you are probably aware, dynamic geometry programs have been around for quite sometime, although they have seen more use in the States than in the UK. As an introduction to the vast amount of material and discussion that is available on the subject, you may like to visit the excellent page at <http://forum.swarthmore.edu>.

I hope by now that you have all tried out at least one of the many dynamic geometry programs available on the Internet. So perhaps now is the time to start asking the all important question of how children using these programs can learn some maths! But first...

What exactly *is* a dynamic geometry program?

At one level, it is a construction tool, in exactly the same way as pencil, paper, compass and ruler are construction tools. The difference is that the constructions created by a dynamic geometry program yield not *one* drawing, but a whole family of drawings.

A few *given* objects (be they points or lines) are placed anywhere in the program's window, and then the remainder of the diagram is constructed from these *givens*. The program remembers the *givens* and the *construction rules*, and updates the diagram whenever the user drags any of the *givens* around the screen.

In the Geometer's SketchPad, it is possible to view the *givens* and *construction rules* by recording a sketch in a script. It's worth trying this so you can see the way the program stores the sketch. The idea of a *given* is a difficult one for a child to grasp, and so this concrete representation of a *given* is useful reinforcement. Remember all those exam questions that take the form '*Given x, prove y?*'

Now for the lesson plan!

Viewing the program as just another construction tool is a really good way of reusing all those old lesson plans that you have on geometric constructions! Just replace *pencil, ruler, and compass* with your program of choice and ask all the old questions again! Get the children to record their sketches, and you have a whole new set of follow-up questions to ask. Time for an example!

Draw a parallelogram

Try this out on a class, and you are sure to get quite a few different constructions. Here are three that I drew and exported to the JavaSketchPad. If you have a Java capable browser, you can view them here:–

- [Parallelogram Construction 1](#)
- [Parallelogram Construction 2](#)
- [Parallelogram Construction 3](#)

Now let's see if we can squeeze out some maths!

- How and why do the sketches differ?
- Are all the sketches valid parallelograms?
- How many *substantially* different constructions are there?
- *Roughly* how many parallelograms can be drawn in each sketch? Infinitely many? Countably many? A finite number?
- Can some sketches draw parallelograms that other sketches cannot draw?
- Each sketch suggests a set of parallelograms. Can you define these sets? Can you prove that one set contains the other set?
- Which sketches are good representations of a general parallelogram?

Scripts

Here are the scripts that generate the parallelograms in the examples above.

Parallelogram 1

Given:

Point A

Point B

Point C

Steps:

1. Let [j] = Segment between Point A and Point B.

2. Let [B'] = Image of Point B translated by vector A->C.

3. Let [j'] = Image of Segment [j] translated by vector A->C.

4. Let [k] = Segment between Point A and Point C.

5. Let [l] = Segment between Point B and Point [B'].

Parallelogram 2

Given:

Point A

Point B

Point C

Geometer's corner

Steps:

1. Let [j] = Segment between Point A and Point B.
2. Let [A'] = Image of Point A rotated 180 degrees about center Point C.
3. Let [B'] = Image of Point B rotated 180 degrees about center Point C.
4. Let [j'] = Image of Segment [j] rotated 180 degrees about center Point C.
5. Let [k] = Segment between Point A and Point [B'].
6. Let [l] = Segment between Point B and Point [A'].

Parallelogram 3

Given:

Point A

Point B

Steps:

1. Let [j] = Segment between Point A and Point B.
2. Let [c1] = Circle with center at Point A passing through Point B.
3. Let [C] = Random point on Circle [c1].
4. Let [k] = Segment between Point A and Point [C].
5. Let [C'] = Image of Point [C] translated by vector A→B.
6. Let [l] = Segment between Point B and Point [C'].
7. Let [m] = Segment between Point [C] and Point [C'].

Questions about scripts

Clearly each script generates a sketch, but there's more to it than that...

- Can *different* scripts generate the same diagram?
- If so, can you find a set of equivalence classes that eliminate some of the redundancy in the script representation. [Hint: Think about naming!]
- Do you now have a set of equivalence classes where each class generates a distinct sketch? If not, what other equivalence relations would help? [This is not easy!]
- Can you find a less verbose textual (algebraic?) representation of your sketch which contains all the necessary information.
- Can this representation be generalised to represent the geometric idea of a parallelogram? If so, is it a useful representation? (In the same sense as $\frac{3}{4}$ and 0.75 are useful representations of a particular number.)
- Set theory was useful in comparing sketches. Does your notation capture this? Can you use your notation to prove anything about sketch 1 and sketch 2?
- How many parallelograms do the *scripts* describe? Interestingly, this is *not* the same number as can be drawn!
- When would the sketch be the more useful representation, and when would the script be more useful?

As you can see, there is a lot of mathematics here, leading on to algebraic geometry and computation theory. Interestingly, the *deep* questions show up when analysing the *simplest* sketches!

About the author



Mike Pearson is a member of the PASS Maths editorial team.



Plus is part of the family of activities in the Millennium Mathematics Project, which also includes the NRICH and MOTIVATE sites.