Contents

This lecture concerns :

Math teaching in France (mostly 20th century).

Most European countries tried to adapt their educational system to modern times, specially in sciences, and particularly in maths.
So French attempts are not isolated, but they are radical because of centralization. A lot of countries abandon to provinces, regions, counties, Länder, the authority to define programs and to apply them to councils. Some harmonization exists at the state level, but let us say that the system is more down-up than contrary.
In France the top-down model is the rule. Although this is a bit changing now some internal organization in the ministry known as 'Inspection Générale' had all power to define what should be presented to pupils in primary and secondary schools and how it should be done.
The seventies saw the victory of 'structuralism' and a kind of 'bourbakization' of math treaties and usual textbooks.
A lot of web sites tell the whole story with great details., but unfortunately and understandably all of them are written in French and accessible only to interested readers who know the language or have good online translators.
Here are some examples:
http://michel.delord.free.fr/rb/rb-placegeom.pdf by Michel Delord
http://www.univ-irem.fr/IMG/pdf/ensG.pdf
https://www.imo.universite-paris-saclay.fr/~perrin/Livregeometrie/DPPostface.pdf
but you can find a lot more about the same subject.
This is an extract from the first reference:

> *Avec le point de vue structural la géométrie élémentaire devient un simple chapitre de l'algèbre linéaire et c'est ce point de vue qui est mis en avant dans l'apprentissage de la géométrie [Choquet 1964] [Dieudonné 1964], quitte à chercher les moyens pédagogiques d'introduire à un tel point de vue.*

I translate the main idea to make simple :

With the victory of structuralists geometry becomes just a part of linear algebra, mainly the study of R^2 and R^3.

What does it mean in practice.?

Take for example this famous example of Pythagoras' theorem.

This wikipedia page shows classical demonstrations :

https://en.wikipedia.org/wiki/Pythagorean_theorem

The 'modern' demonstration of it can be found in the same page under the title 'Inner product spaces', and it's just a simple algebraic computation, masking the reality of the meaning and the ingenuity of the classical proofs (Greek and Chinese).

So according to the modernist classical (affine) geometry consists in a few algebraic formulas that can be understood by themselves without any reference to any drawing.

Classical geometry offers a mental frame to represent space. A 'plane' is not simply an affine variety of the second dimension, it is an abstraction from the sensible 'real' world.

And I would go farther, only with a good mental representation of space you can understand the results of linear algebra. The question relative to intersections and reunions of subspaces become

obvious for the one who was educated ‘the old way’. Can we say the same about students who were driven in geometry simply by affine spaces as a specialization of vector spaces ?

The story doesn’t finish here. A lot of teachers and parents were definitely against this ‘modern maths’ approach. And they finally won this battle. But maybe it was a pyrrhic victory ?

As early as 1977 the new trend in France is known as the ‘Contre Réforme’ . The full story is here :

https://halshs.archives-ouvertes.fr/halshs-00960313/document

Does it mean a return backward ?

I would say unfortunately no ! There was nothing like a come-back to tradition. Classical Euclidean Geometry is almost absent from the programs , or what is left is quite ridiculous. See for example the official program for the last year (terminale) of secondary (high) school, for the pupils who take maths specialty. In the following page under the title ‘Géométrie dans l’espace’.

http://media.education.gouv.fr/file/special_8_men/98/4/mathematiques_S_195984.pdf

So new adult students arrive in university or institutes to study maths or computer science without these mental structures, not really knowing what a geometrical transformation is, ignoring everything about conic sections and their properties.

I have been a moderator and redactor for a few years on the French site for programmers:

https://www.developpez.com/

Here I could see how some young men had to ‘reinvent the wheel’ when programming 3D vision programs, because of a lack of knowledge in basic geometry. Some of them were ready to compete for the Fields medal because they (re)discovered that it happens that the product of two rotations is again a rotation.

Clearly, something has to be done, I mean for first year students. On my point of view the linear algebra approach is not bad if correctly illustrated with figures which ‘bring back people down to earth’. Idea is to illustrate the theoretical concepts by geometrical figures.

As a result I proposed, among others, and in French language a course of affine geometry.

http://gilles.dubois10.free.fr/geometrie_affine/geometrie_affine.html

As a prerequisite I proposed, in the same site, a course of linear algebra:

http://gilles.dubois10.free.fr/algebre_lineaire/algebre_lineaire.html

All of this was written between 2008 and 2010. Today the peak of attendance is around 1400 visits a day and during summer it falls down to 300.

Dynamic geometry and related sofwares.

So by writing my web-pages (html documents) with some specific tool. I found bluefish to be the best, but it's just a matter of taste.

As to drawing figures I found GeoGebra already advanced at the time to be the most convenient, with a very small learning curve.

Of course as long as you concern is 'static figures' Geogebra was perfect offering the possibility to export any image in any format.

But the new softwares susch as GeoGebra and JSXGraph offer many other possibilities which we group by the name 'Dynamic Geometry'.

'Dynamic Geometry' introduces the notion of motion in different ways.

First can generate 'figures in motion' under the form of videos. This is specially useful to illustrate how a figure is transformed into another under a symmetry, a rotation, a translation. It's one thing to represent the rotated image of a solid and it's another one to see a solid rotating to reach a new position. This kind of videos can be used to illustrate, for example than a 2D symmetry can be interpreted as a 3D rotation, and that we can imagine some kind of isometry in 4D space which would bring a left glove exactly in coincidence with a right one.

But we can do more. In the previous examples, although the figures are not static, the student is passive (just watching) without interaction. 'Dynamic Geometry' offers the possibility to the reader to interact with the figures by changing places of some points, varying some parameters to examine a lot of possible situations and discover some law.

Here's a document by Mr Bantchev , dated from 2010, which explains the concept of 'Dynamic'gives a lot of 'Dynamic Geometry Softwares'. Unfortunately JSXGraph is not mentioned, more about this later.

http://www.math.bas.bg/bantchev/misc/dgs.pdf

About this point I wasn't able to find any history of JSXGraph and its relationship with GeoNext, apparently developed by the same team of the university of Bayreuth. Nevertheless I could find that version 0,6 was released at the end of 2008.

The following article by Adrian Oldknow explains the merits and benefits of DGS (dynamic Geometry Softwares)

http://wwwg.uni-klu.ac.at/stochastik.schule/ICTMT_5/ICTMT_5_CD/Special%20groups/CD_Special2.htm

The following YouTube video shows with a very simple example how to use DGS in the classroom:

https://www.youtube.com/watch?v=rtT77A9Fto0

Production of web documents using DGS (years 2000-2010)

When it comes to write web-sites dedicated to geometry during the 2000 decade what do we have on personal computers ?

Three operating systems :

- Microsoft windows on PC
- MacOs/iOS on Apple computers
- Linux in the PC world

Five browsers :

- Microsoft Explorer
- Mozilla Firefox
- Google Chrome (from 2008)
- Opera by Opera Software
- Safari on Apple computers

Mobile devices at the time under Android, iOS or Windows Mobile are not in significant number although the situation is going to change rapidly in the next decade (2010-2020).

So DGS should appear as applets in web pages. There were not so many ways at the times to write applets which could be executed on any shared server , I mean :

- Javascript applets
- Java applets

Now if we come back to the list proposed by Dr Bantchev with see that a lot of DGS softwares are connected to the Java world either because initially written in this language or because native files can be played directly through JVM (Java Virtual Machines) existing on most systems or through additional plugins (for example CabriJava).

Here is a short list :

- GeoGebra
- Cabri Géomètre
- Cinderella
- C.A.R.
- CARMetal
- Geometria
- GEONext
- JGEX
- etc…

Although already very popular, the French Cabri-Géomètre was eliminated by me because of its restriction to Microsoft systems, commercially distributed and not open source. Some users say good things about it, but my moral standards don't accept making money with the transmission of knowledge. Furthermore so many platform independent libraries exist for so many programming languages that I don't understand people continuing to develop only for one specific OS and I have even doubts about their competence. This is a strictly personal point of view.

It seemed to me at that time (2008-2010) that the best choice was GeoGebra, and by the way Dr Bantchev himself considered it as the most advanced project.

Let's come back to this technical point of playing java applets in a browser. JVM (Java Virtual Machine) should be installed on the machine and the browser should know where to find the dynamic libraries (.dll for Windows or .so files for Linux), usually through a plugin. This is a beginning of a real nightmare.

From the beginning of the 21st century, the trend for browser is to have access to more and more material resources of the host computer and do more and more tasks usually devoted to the operating system. As a result the systems using players with a lot of plugins began to be more and more unstable. This was specially the case with the JVM plugins. I could see by myself some of my applets freezing the robust Linux system on PC.

So developers began to update to new versions trying to correct the computer flaws. This led to new versions three or four times per year with endless updating.

Furthermore installation of plugins were variable according to OS and version of browsers. From 2008 to end of 2014 I spent my time in supporting my readers to install the correct configuration to play my applets not to answer their questions about maths subjects.

At the beginning of 2015 I decided that it couldn't continue like this. The situation was specially difficult with various Linux distros where .so files were located in different places. To offer assistance to readers was a real brain teaser.

I began to try to find any mean to convert GeoGebra applets (arount 450 of them) into javascript applets. Unfortunately, at that time GeoGebra didn't propose any simple, elegant and safe solution. So I got used to the idea that I should rewrite the whole thing.

At the time (beginning of 2015) JSXGraph began to be famous, documentation was not scarce and good support was provided by the community and the development team as well in a Google Group (kind of forum) currently counting about 300 users and 1200 conversations.

So I began to rewrite everything, the job took me two full months January and February 2015 .

First I was able to reproduce exactly the behavior of GeoGebra applets, without any restriction. With a lot of examples, a good reference and help of the forum, my job was easier day after day.

The result was quite satisfactory, and the main everything worked without any external libraries, plugins, virtual machines, etc. etc. Any html server knows how to play JavaScript programs. The loading of pages was very quick and no complaining about applet executions at all. It was a kind of new life for my site. Till now and for five years everything works fine.

Later on I had the opportunity to use JSXGraph again for an advanced calculus course . The application was not quite simple but worked very well, a bug was kindly corrected by a member of the community. Inclusion was made inside a ‘WordPress’ site of mine without any problem, pages of these sites are usual html pages after all ; fact that they are stored in a database is irrelevant to the use of JavaScript applets. The CMS (WordPress) just manages these pages conveniently.

More recently I had to use again JSXGraph to help my grandson (12 years old) during the quarantine period introducing Pythagoras’ theorem. Although I had not been using JSXGraph for a few years I could do everything very quickly. So using JSXGraph is like riding a bicycle, you never forget it !

3D Representations

Almost all DGS propose 3D extensions. But honestly everything is a little disappointing.

In fact these extensions are not useful at all if you notice that representing a figure in 3D on a plane supposes the use of a projection system, I mean a matrix with 2 lines and 3 columns for the calculation of the projection of a point of the space on the plane. But of course with some demerits due to perspective effects. So to correct these effects you must give the opportunity to the reader to turn the (solid) figure itself or to turn around it.

GeoGebra uses special light effects to represent classical volumes as spheres and it is sometimes enough to create static pictures by exportation. I used GeoGebra 3D for some static pictures see below for examples

Spheres

Tetrahedrons

I discovered the Phoria JavaScript Library by Kevin Roast which impressed me very much. Actually this library wasn't dedicated to DG but it was open source. So I could modify it to adapt it to my personal needs. Really Kevin's job is a great one but apparently he stopped to support and develop his work as early as 2014. Nevertheless it continues to do the job.

Here are some examples among others. I don't see any equivalent produced with 3D extensions of DG softwares, as to the rendering.

Ruled Surfaces

Sphere and cone

Dandelin sphere

Parabola

Ellipse

Hyperbola

GeoGebra v/s JSXGraph

GeoGebra and JSXGraph are the two DGS that I used intensively and for which I can compare merits and demerits.

The main advantage of GeoGebra is the learning curve. GeoGebra can be used by children, by people without any knowledge in programming, almost instantly, this is definitely not the case of JSXGraph.

In fact, although their goal is the same (production of clean dynamic geometry figures) the approach is completely different. GeoGebra is a all-in-one product quite closed. The ggb.files are definitely not text-files, but they can generate explicit construction files which have mathematical meaning.

On the other hand JSXGraph is simply a JavaScript library ;it means a tool to be used mostly by programmers. JSXGraph applets are text files, actually pure JS code.

The programmer has total control over the program and can interface it with usual window controls (buttons, check-boxes, lists, text area, etc…). A JSXGraph program can correspond to the graphical part of a wider educational program.

For examples see this page or this other one

If the purpose is to make graphical static figures I would recommend to use GeoGebra and export to any convenient graphical format.

For real interactive applet inside web-pages I would recommend JSXGraph.

Recently GeoGebra made some effort to offer possibility to run their applets without Java Runtime installed. But the operation is not so simple as they pretend it to be. They propose to use some instruction like :

<param name="filename" value="applet.ggb">

with a reference to an existing ggb file nearby.

I was never able to make it work ! And it seems that I’m not alone in this case.

Instead , I was successful using such instruction :

<param name="ggbBase64" value="UesDBBQACAA....">

But the parameter following the key-word value is a rather long code, for example if you watch this page. The code corresponding to the first figure is

```
<div id="ggb-element"></div>
<script>

  var ggbApp = new
GGBApplet({ "ggbBase64":"UEsDBBQACAgIAEaU4lAAAAAAAAAAAAAAAAAWAAAAZ2VvZ2VicmFfamF
2YXNjcmlwdC5qc0srzUsuyczPU0hPT/
LP88zLLNHQVKiu5QIAUEsHCEXM3l0aAAAAGAAAAFBLAwQUAAgICABGlOJQAAAAAAAAAAAAAAAADAAAAG
dlb2dlYnJhLnhtbO1abW/
bOBL+3P0VhD4s7rC1zXdJXbuLNMGiBbqbRdM7HO5wH2iJsbmRJa8kJ07RH79DUrIlu0njtN26h0vrUBK
HQ87zzAyHcsY/
rRcZutZlZYp8EpAhDpDOkyI1+WwSrOrLQRT89Py78UwXMz0tFbosyoWqJwG3kiadBCwWiVJCDmSo8IBP
+eVgKkM6oJomEaM6kmoaILSuzLO8+FUtdLVUib5I5nqhXheJqt3E87pePhuNbm5uhu1Uw6KcjWaz6XBd
pQGCZebVJGgunoG63qAb5sQpxmT0r19ee/UDk1e1yhMdIGvCyjz/
7sn4xuRpcYNuTFrPJ4GMWYDm2szmYFPI4wCNrNASAFnqpDbXuoKhnVtnc71YBk5M5bb/
```

```
ib9C2cacAKXm2qS6nAR4yAgXId/
+lgEqSqPzuhEmzaSjVt342ugbr9deuSl5gOqiyKbKqkTv3yOKKUZPbUN8Q6GR0ndh/
wwz31DfcN8IL8P9cO5FuZfhXoYDKtemMtNMT4JLlVUAockvS6Bvc1/
Vt5l262kebM0nT8GmyrwDYYbBTzzm8Bzjp/Yj4cNtx6hvJOnMWperAydtp5QRe/
iU9JMMZe2clMT7c1Jxh5nynkm93Q+xk4gOtDCV+
+8+ezOy+8zcndHff9qEkv8lJo5HbaiMm+hA1dzKNt5T60Vl44XFSMTW7QkSEBsyBC8XiMTQhBRBNCAiE
BdwSyIkbRsiFkIHRwxFyMoRhlxwiAh+8dApk0iAMvs0hJhEBCbiSDBEXExxBJGEXFxCjFIGEkIgAYPs9
IRaFUwiLuGORYjDGm1IhgQEGQyEe5ieIkYQs4NJiKhE0uoj3Ia6jOzSQSVFEiNJrEKIaohoH80gHyFmr
ZENXCZfruoeRMkibS/
rYrnhAqQhH23Tns9Pvaz4ZJypqc5go7iwTCJ0rTIbEW6iyyKvUUsi9c9mpVrOTVJd6LqGURX6XV2r16r
W659BumrndrJJkVe/
lUV9WmSrRV4hlBQZ3qy5yEjnmm5WDTes08G7HaLTITvX4QfnLaAHrSoN8xdl1YqrNH1lJbapAZA8z7Pb
F6VWV8vC9M0Yj9yeM9arJDOpUfk/
wVntLBYXtNmCos4WJCLRLqQo04vbCjwYrf+ty8LmGD6Muz9hgG59FyNiiLs/
QHiVKBt8PO4PiqDrtukTuD+qIU9fbyhSa721dlba0O7cvKpeFNn2kQPgVC3rVenKB7CrtGad5LNMOydx
oQ17c3I1LdYX3juY1/
X2dgl32K9gOnPAI0gOVAgQaNqpb52MXdpGCjsZ7CRw624m3fSTmDoJ105966TAf/
3SGlNJaybB7TSmcikNB03gtOnKer/d6le5qV+3N7VJrram2gG/
rhZTvfGhvk7yuXSORztONr7SZa6zxqeBzFWxqnyIdtw91YlZwK3vaCBRlq5/
wAL801TPSt0uPHOlmQfM9eKuu+49dqp+LovFq/
z6LfjCzgLGo3aV4yopzdL6HJrCPnClt16VmkrBNpJ2x9kgBNMTu10APLWFBsJzVc+L0lVfkFWgtbGX6Q
WUWqh27uU8dAPzuSviLJ6omP4OiW2z9/n+LWHQ/UFXc06psuVc2UKvMTpTt7rsweD0/
VKku+AA9s4CiPKlVWDZXWrtHcOvGC6WoNDFUy9PAd4VWttREbj07SQYuPHvfBHvq1hrrY2yXmr2T3eoA
v/xQH0Msu/
Vsqh+PC7gHgObGMa8hS2MvgBu+WqhS5NsMHnjMIORq2Y8HzaKPwVH3EGRPBDFRq7K7AEFLUzu1CyURQX
UTSvYc2s4oUHU59sTml9Ys2dBdWvPfzCCh/
YCcGTuRHhp1npTUEAwmneQfFTPlG3GraEauIIjT+XK8LrZANzFS5OmOt+s9UGhgj/
E+QEEdVy7QxNtcf2maWJkwxM9ep6SYrFQeYpyd1o4NWWS6WBbvipsUzdSxAaVh31Vtx2JV9ao2CMdNo4
O5clH8lgnczyE2QPT2GMAhjJR59ewUqhKEVrj5pXNLfbzo3ftkzVxqc32kebRO9LJcUBTadbopJU/
aaVOoJwegLdEOBS9UvOENXOccKfabjsnouPR+o/cm1P5WsWeFcylSR5DbxOLnuTmZpfq9BCq0/9T/
UGqGRnyiPZPDn2m/U7pmYad8hOovtAz+/zDoXx+B8vqfparRmfLo/
qqPHfLMhm5fCuGcP615YWliAvRP4o93jt2GDCLZWYSU2/wy2yCfZXXcEzQrk7er/6vtF7aY9d5/
rZUeWVf7XqZzqniDio3infITDyZao/
GE99xP5n9WvPkUVWmffM0883UN59OJhyuIypCTIXEEaFcirZ0lFISSuFDmAhjLD5LIfkwqNO7oH5xONQ
vjgZqNoxERMNYcsrCKKYbpAVmghJBBMaMi8900lkvSwg2W6+0hup1rUE59EyC7/9YFfWP501mmvhb9AP
625nx3yv85/wpanr/+3f/0qyPslUX7Oj+mvkJll3Wv1lYkAUbYOW0yVKWvi2kD0CJ9lB688ObPZjewK/
m6QPRoUeGDmO9V2fujZtLB2Hv+YHIsR5y798MGpDed7CD+h7wG2zweyCA7MgAJH2gmjM3H0rxIAAPyoT
JXiY89R1n0FD3gogdlhZPjyYtcsh/
EoeRwIJLyez3czYtUvvlImWU0xj+R7H84i+Azo4IklhICnBAMAIsEvutAowPBaMR7NMR5rAvf56XO31P
fA0V244TnnlfO91zwun9HmeLvw160yOpXtmQRYAulDqUMk5wk/
jsfiEwCSXBlEcAbshDX9sSCZ4oOCckhn9AC/
sype0jDxdne6zow44V+kiIGRBIqT33jpuMSlg3GBqfZ2HI4D4KIT8IQr4QKX/
BeeNedvdj7vIwdi+PhF06pLvp3L6xg6PkTu53ITfkPMSShpwIqJkj/O2eJu9gt/
sSaD+CZ4dxPDsSjoG3SOxuUD63xhELuWAxxTzmUHn6IMZDyKaMQa6lVERchv/
LNO+H8vwwmudHQjMZCtYLZeJZhqiFQwWEOSRkxqQIeRPNQDOz1TKLpeDgCeKbpfmuMl15iqd7FL88pCZ
/eTQFaMSGAn4iGdtodu8kbj2TQGLMJBRKkQhJ++1iOJQSy5AxHMc0jNu/
ivlS9ehHXquaA6pScyQx9TXfqXaxH3X//MD9mU/zJ6vP/
wRQSwcI212eGT0JAABPKwAAUEsBAhQAFAAICAgARpTiUEXM3l0aAAAAGAAAABYAAAAAAAAAAAAAAAAA
AAAAGdlb2dlYnJhX2phdmFzY3JpcHQuanNQSwECFAAUAAgICABGlOJQ212eGT0JAABPKwAADAAAAAAAAA
AAAAAAAABeAAAAZ2VvZ2VicmEueG1sUEsFBgAAAAACAAIAfgAANUJAAAAAA=="
   , "width": 800, "height": 600, "showToolBar": false, "showAlgebraInput":
false, "showMenuBar": false ,"enableRightClick":false}, true);


   window.addEventListener("load", function() {
       ggbApp.setHTML5Codebase('./GeoGebra/HTML5/5.0/web3d/');
        ggbApp.inject('ggb-element');
    });
</script>
```

Now, if some change or updating has to be done, the following steps have to be followed.

- Modify your ggb file.

- Export to html this construction.
- Copy the generated code.
- Paste the new code into the source page in place of the old one.

The problem of line cuts complicates the process.

Although the playing of the GeoGebra applet is not immediate, there's something like a compilation time 'on the fly' which slower the process. But it works !

JSXGraph produces pure Javascript code so inside an html page we can use the classical call to such routines :

So let's consider this page.

The inclusion of the last applet is made by :

```
<div id="box4" class="jxgbox" style="width:500px; height:500px;">
   </div>
   <script src="faisceaux4.js"></script>
```

and the code for it can been seen by the source explorer of the browser:

```
var brd4 = JXG.JSXGraph.initBoard('box4',{axis:false,boundingbox:[-4,4,4,-4]});
var lab4=brd4.create('text',[-3,2.5,"Espacement entre les droites"],
{color:'green'});
var s4 = brd4.create('slider',[[0,2.5],[2,2.5],[0,1,1]],{color:'green'});

var A4= brd4.create('point' ,[0,function(){return -2*s4.Value();}],{name:'A',
visible:false});
var B4= brd4.create('point' ,[0,function(){return -1*s4.Value();}],{name:'B',
visible:false});
var C4= brd4.create('point' ,[0,0],{name:'C', visible:false});
var D4= brd4.create('point' ,[0,function(){return s4.Value();}],{name:'D',
visible:false});
var E4= brd4.create('point' ,[0,function(){return 2*s4.Value();}],{name:'E',
visible:false});

var P4= brd4.create('point' ,[-3,-3],{name:'P'});
var Q4= brd4.create('point' ,[3.5,3.5],{name:'Q'});
var PQ4=brd4.create('line',[P4,Q4],{color:'red'});
var X4= brd4.create('point' ,[-2.5,-2.5],{name:'X'});
var Y4= brd4.create('point' ,[2.5,-2.5],{name:'Y'});

var XY4=brd4.create('line',[X4,Y4]);
var PA4=brd4.create('parallel',[XY4,A4]);
var PB4=brd4.create('parallel',[XY4,B4]);
var PC4=brd4.create('parallel',[XY4,C4]);
var PD4=brd4.create('parallel',[XY4,D4]);
var PE4=brd4.create('parallel',[XY4,E4]);

var I1=brd4.create('intersection',[PQ4,PA4], {name: 'I<sub>1</sub>'});
var I2=brd4.create('intersection',[PQ4,PB4], {name: 'I<sub>2</sub>'});
var I3=brd4.create('intersection',[PQ4,PC4], {name: 'I<sub>3</sub>'});
var I4=brd4.create('intersection',[PQ4,PD4], {name: 'I<sub>4</sub>'});
var I5=brd4.create('intersection',[PQ4,PE4], {name: 'I<sub>5</sub>'});


var HI1= brd4.create('normal', [XY4, I1],{color:'black', visible:false});
var HI2= brd4.create('normal', [XY4, I2],{color:'black', visible:false});
var HI3= brd4.create('normal', [XY4, I3],{color:'black', visible:false});
var HI4= brd4.create('normal', [XY4, I4],{color:'black', visible:false});
```

```
var H1=brd4.create('intersection',[HI1,PB4], {name:'H<sub>1</sub>'});
var H2=brd4.create('intersection',[HI2,PC4], {name:'H<sub>2</sub>'});
var H3=brd4.create('intersection',[HI3,PD4], {name:'H<sub>3</sub>'});
var H4=brd4.create('intersection',[HI4,PE4], {name:'H<sub>4</sub>'});

var I1H1=brd4.create('line',[I1,H1],{color:'black',straightFirst:false,
straightLast:false});
var I2H2=brd4.create('line',[I2,H2],{color:'black',straightFirst:false,
straightLast:false});
var I3H3=brd4.create('line',[I3,H3],{color:'black',straightFirst:false,
straightLast:false});
var I4H4=brd4.create('line',[I4,H4],{color:'black',straightFirst:false,
straightLast:false});

var T1 = brd4.create('polygon', [I1, H1, I2],{color:'yellow'});
var T2 = brd4.create('polygon', [I2, H2, I3],{color:'yellow'});
var T3 = brd4.create('polygon', [I3, H3, I4],{color:'yellow'});
var T4 = brd4.create('polygon', [I4, H4, I5],{color:'yellow'});
```

So for debugging or updating, direct intervention is possible on the source text-file faisceaux4.js.

This seems to me much simpler.

On the contrary of GeoGebra, JSXGraph applets execute immediately, without any delay, so reactivity of the whole page is better specially if it contains a lot of such applets.

Nether is nothing like a ‘black box’ in the JSXGraph developing process. For all of these reasons I confirm my preference to the use of JSXGraph for interactive complex applets, but I can and will continue to use GeoGebra for fixed images.

Now how to improve productivity in JSXGraph programming ?

First you can use tricks of your JavaScript editor. Personaly I use Bluefish for html and js as well. The possibility exist to create ‘snippets’ which are kinds of macros to save a lot of time.

You can also keep a text file with a lot of patterns use copy/paste and change what has to be changed ; it works very well too.

Of course the development team can invent a kind of ‘meta-language’ that would produce JSXGraph code using a graphical interface.

The best of the two worlds would be a GeoGebra interface producing GSXGraph code instead of obscure ggb files.