# TP sous JavaScript avec CaRMetal

Alain Busser(*)

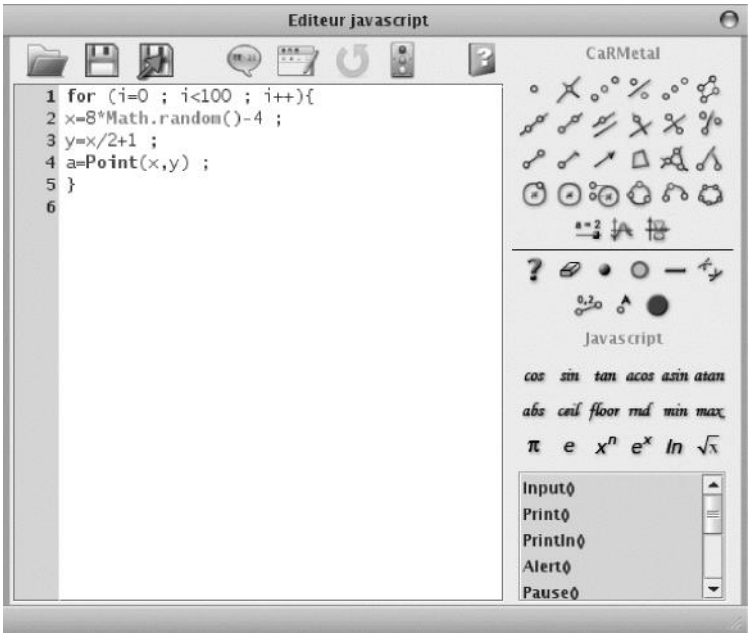
## Choix de l'outil

Le choix du langage de programmation n'a pas précédé le choix de l'outil ; c'est au contraire le choix de l'outil qui a entraîné celui du langage de programmation. En effet CaRMetal étant un logiciel de géométrie dynamique, peut servir à faire à la fois de la géométrie et de la programmation, ou plutôt de la géométrie sous couvert de programmation, ce qui est finalement un gain de temps (en fait deux choses en même temps) et permet de présenter les notions de base de l'algorithmique, comme la création et la modification de variables, sous une forme graphique donc moins abstraite.
Le langage *ECMAscript*, plus connu sous le nom de *JavaScript*, est très utilisé pour établir des sites Internet, au point que c'est un des trois langages inclus dans la suite bureautique *OpenOffice*. Inspiré du célébrissime langage C++, il ressemble à plusieurs langages de programmation répandus comme Java, le langage de programmation de Xcas, et même AlgoBox. Ces arguments, de peu de poids pour un prof de maths, sont très prisés pas les élèves, qui apprécient la « valeur ajoutée sur le marché du travail ».

L'outil de programmation est CaRMetal, téléchargeable ici :
http://db-maths.nuxit.net/CaRMetal/

C'est un logiciel de géométrie dynamique, ce qui permet de l'utiliser aussi pour autre chose que la programmation (de la géométrie par exemple, y compris dans l'espace). CaRMetal est léger, libre et multiplateforme, ce qui permet à la plupart des élèves de finir le TP chez eux. Une fois le logiciel démarré, on clique sur l'entrée « JavaScript » du menu et là, on peut cliquer sur l'entrée « Ouvrir l'éditeur de scripts » pour voir s'ouvrir la fenêtre de la page suivante.
Son aspect en noir et blanc ne permet pas de montrer l'effet de la coloration syntaxique, à laquelle les élèves sont très vite devenus accros (programmer avec un outil sans coloration syntaxique, ils refuseraient). Pour écrire un programme en JavaScript, il n'est d'ailleurs même pas nécessaire de compter sur cette coloration syntaxique, puisque le clic sur les icônes de droite construit le programme étape par étape. La moitié du bas est consacrée aux instructions JavaScript de base, alors que celle du haut est dédiée aux instructions graphiques, ce qui ouvre plein de possibilités (voir ci-dessous).

(*) abusser@ac-reunion.fr




### Affectation et modification de variables

Les sujets de TP, ainsi que des commentaires sur la manière dont ils se sont déroulés, sont publiés dans une rubrique qui leur est consacrée sur le site de l'IREM de la Réunion, à l'adresse suivante :

http://www.reunion.iufm.fr/recherche/irem/spip.php?rubrique58

On peut notamment y télécharger l'énoncé du TP numéro 1[(1)], portant sur des créations et affectations successives de variables numériques. Par exemple, le premier exercice était « que contiendra la variable x à la fin de ce script? » :

```
Affecter 7 à la variable x      var x=7;
Remplacer x par son carré       x=x*x;
Afficher x et aller à la ligne  Println(x);
```

La première ligne fait deux choses en une : création d'une variable nommée x, et affectation à cette variable de la valeur 7. L'instruction, comme toutes les instructions JavaScript, se termine par un point-virgule. Le langage est dynamiquement typé, ce qui veut dire que lorsqu'on « range 7 dans le tiroir x », JavaScript comprend alors que x est un nombre. Avant ça x était considéré comme indéterminé (en anglais « undefined »): Ç'eût pu tout aussi bien être une chaîne de caractères ou un tableau, pour ce qu'on en savait… Il y a lieu de débattre là-dessus : un langage plus rigoureux aurait réclamé que les variables soient créées toutes dans le préambule, et que leur type soit choisi définitivement à ce moment : Exemples, *Pascal*, *Java* ou *Algobox*… La question du point de vue du prof est « vaut-il mieux passer du temps à développer ces

(1) À l'adresse suivante : http://www.reunion.iufm.fr/recherche/irem/spip.php?article171

habitudes de rigueur, ou choisir une solution souple qui évite de multiplier les questions que se poseront les élèves ? ». Du point de vue de l'élève, ce serait plutôt la seconde alternative…
La seconde ligne *remplace* x par son carré. C'est une affectation, plus complexe que la première (la valeur 7 ne dépendait pas de x) mais sans création préalable. Là encore, JavaScript offre une souplesse rare : Écrire « var x=x*x » (qui créera une deuxième fois x) ne provoque pas de message d'erreur. De même l'oubli de « var » dans la première ligne ne rend pas le script faux. La dernière ligne affiche la valeur de x pour qu'on puisse vérifier qu'il est bien égal à 49, ce qui d'ailleurs a été prévu par la plupart des élèves.
Quelques élèves ont développé lors de cette activité un comportement un peu surprenant : si on enlève la troisième ligne de ce script, ils ne comprennent plus la question, comme si une variable n'existait que lorsqu'on affiche sa valeur. En ce sens, ils semblent illustrer la théorie de Kant sur l'espace étudié en géométrie : « Si nous sortons de la condition subjective […] la représentation de l'espace ne signifie plus rien » (*in* « Critique de la raison pure »).

Le TP a été finalement très court (environ une demi-heure) et si c'était à refaire il aurait mieux valu illustrer la différence entre créer une variable et la modifier, par le fait que CaRMetal permet de créer des points (par « Point(,) ») et les modifier, c'est-à-dire les déplacer (par « Move(,) »). La difficulté supplémentaire du fait qu'il faut deux coordonnées pour un point est compensée par le fait que les élèves perçoivent mieux ce qui est visuel que les nombres, plus abstraits que les figures géométriques.

Par exemple, avec le script suivant :

```
Créer un point de coordonnées 2 et 3 et    var a=Point(2,3);
affecter son nom à la variable a
Déplacer le point dont le nom est dans a,  Move(a,-1,5);
vers la position (-1;5)
```

Initialement (c'est-à-dire à l'issue de la première ligne), un point est créé et placé en (2;3). Mais lors de l'exécution de la deuxième ligne, a est *déplacé*, c'est-à-dire que ses coordonnées sont modifiées, la nouvelle abscisse étant −1 et la nouvelle ordonnée 5[(2)].

### Deuxième TP: Les fonctions

Le deuxième TP[(3)] ressemblait plus à un vrai TP puisque les élèves avaient à créer entièrement un programme, demandant l'entrée d'une variable et affichant l'image de

(2) La première ligne crée une variable a, qui est de type « chaîne de caractères ». Sa valeur est le nom du point (en général, « P1 »). Pour moi comme pour mes élèves, concrètement a n'est pas un nom mais un point. Déplacer le point de nom « P1 » vers la position (−1;5) devient dans nos cerveaux avides de raccourcis, mettre a en (−1;5). Ce raccourci qui permet d'être vite opérant en JavaScript, ne gêne pas au début de l'année. Prendre conscience que a est un nom et pas un objet est facile (il suffit d'ajouter un « Println(a) » et pas indispensable.
(3) Le compte-rendu est ici :
http://www.reunion.iufm.fr/recherche/irem/spip.php?article185

cette variable par une fonction. Malgré la relative complexité de la syntaxe de JavaScript (puissances, π) les élèves ont facilement pu créer les programmes voulus, ce qui donne à ce « nouvel » outil un intérêt formidable pour illustrer les classiques questions sur les écritures algébriques. Par exemple, une question comme « que contient la variable a suivante ? »:

```
Affecter à la variable a l'expression          var a=2+3*4;
2+3*4
```

Exercice classique avec la calculatrice graphique, et toujours aussi difficile avec le nouvel outil !
Dans la même catégorie il y a aussi les questions sur l'écriture en ligne de fractions, les parenthèses, et l'écriture scientifique.
Les notions d'entrée de données et de sortie de données ont également été abordées lors de ce TP. L'impression qui s'en dégage est qu'il est parfaitement raisonnable d'exiger de tous les élèves d'une classe de Seconde qu'ils sachent faire ça : affecter des variables, deviner le résultat, et entrer et sortir des données du programme. Ceci même si les variables en question ne sont pas numériques, le sens de la concaténation de chaînes dont certaines sont obtenues numériquement comme dans

```
Concaténer la chaîne « L'image de          Println(« L'image
 » avec la valeur de x, puis ajouter       de »+x+ «  par f est
à celle-ci, par concaténation, la          »+y );
chaîne « par f » et ajouter la
valeur de y; afficher le résultat
avec retour à la ligne
```

étant apparu *spontanément* à certains élèves. Il s'agit tout de même d'un texte bricolé à partir de morceaux dont certains (x et y) sont convertis à partir de nombres ! Cette notion, *a priori* difficile, au point par exemple qu'en section scientifique on hésite à faire faire la manip sous GeoGebra à des élèves, est apparemment très naturelle pour ceux-ci, ou au moins pour les meilleurs d'entre eux.

### Troisième TP : Les boucles

Le script qui apparaît dans la copie d'écran de la fenêtre d'édition (au début de cet article) est extrait du TP consacré aux boucles[(4)], notion apparemment bien plus complexe que prévu. En l'occurrence il s'agissait de créer un nuage de points, dont l'abscisse est un nombre aléatoire compris entre −4 et 4, et l'ordonnée est 1 de plus que la moitié de l'abscisse. La syntaxe des boucles sous JavaScript (on fait tout 100 fois) a été jugée simple par les élèves. Par contre peu d'entre eux ont trouvé l'intervalle parcouru par x (malgré un intérêt énorme pour l'instruction « random() » et les nombres pseudo-aléatoires) et peu d'entre eux ont su expliquer l'alignement des points : le cours sur les fonctions affines n'avait pas encore été fait, et la plupart des élèves se souviennent d'avoir vu un cours sur les fonctions affines au collège, sans se souvenir du contenu de ce cours…
Dans un script tel que

(4) Le compte-rendu est ici :
http://www.reunion.iufm.fr/recherche/irem/spip.php?article213

```
Démarrer une boucle dont l'indice est un
entier allant de 0 à 99 (par pas de 1)
Dans la boucle, créer un point dont les
coordonnées sont pseudo-aléatoires entre
0 et 1 et affecter à la variable a le nom
de ce point
Fin de la boucle
```

```
for(indice=0;indice<100;
indice=indice+1){
    a=Point(Math.random(
),Math.random());

}
```

il est évident qu'on crée 100 points (en fait on crée 100 fois un point), et c'est évident parce que dans la boucle, *il n'est fait aucune référence à l'indice*. Par contre dans le script de calcul de la somme des premiers entiers :

```
Créer une variable « somme »,
initialement nulle
Démarrer une boucle, la variable
« indice » allant de 0 à 15 (compris)
Dans la boucle, ajouter à la valeur
courante de « somme », la valeur
courante de l'indice
Fin de la boucle
```

```
var somme=0;

for(indice=0;indice<=15
;indice=indice+1){
    somme=somme+indice;

}
```

la somme des valeurs successives de l'indice est calculée au fur et à mesure, *ce qui est nettement moins bien perçu par les élèves*. La différence entre les deux scripts est que, dans le deuxième, l'indice apparaît dans la boucle. Donc les élèves ont du mal à concevoir que « indice » est nul lors du premier passage dans la boucle, puis égal à 1 lors du deuxième passage, puis égal à 2, etc. En fait il semble qu'une perception dynamique des boucles (on entre dans la boucle, *puis* on retourne au début, etc.) soit déjà défaillante pour certains élèves, qui voient plutôt la boucle comme un objet statique (le tout se passe 16 fois, sans saisir que ce n'est pas le même tout à chaque passage)[(5)]. Cette difficulté rappelle beaucoup celles vécues par les élèves de Première lors du cours sur les suites[(6)].

Sur les boucles aussi, CaRMetal offre un bon moyen de visualiser la notion d'indice changeant, un peu comme le mode pas-à-pas de certains débogueurs :

(5) Pour tenter d'expliquer l'origine de cette difficulté, je n'ai trouvé qu'un article majestueux d'Yves Martin sur le site de l'IREM de la Réunion. La première partie de cet article est ici :

http://www.reunion.iufm.fr/recherche/irem/spip.php?article232

et la deuxième partie est ici :

http://www.reunion.iufm.fr/recherche/irem/spip.php?article238

La lecture de ce long article présente un intérêt général qui dépasse largement le cadre de *JavaScript*. Guillaume Connan, de l'IREM de Nantes prévoit une difficulté similaire avec la récursivité dans cet article de MathemaTice :

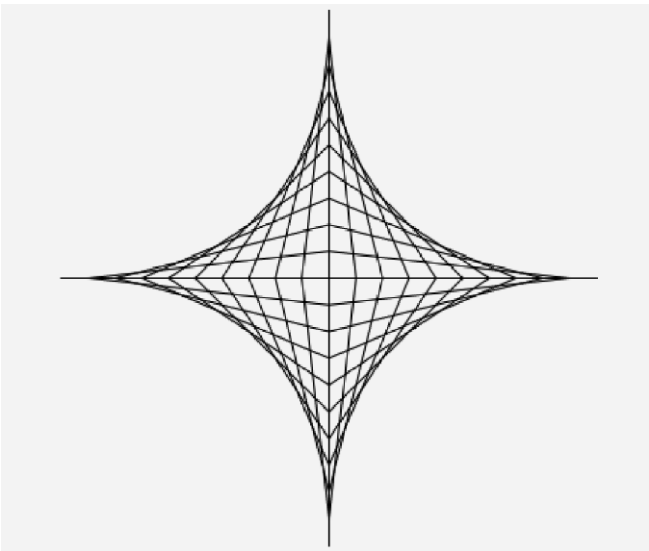
http://revue.sesamath.net/spip.php?article216

(6) En coupant le cours en deux : les suites arithmétiques au premier trimestre, les suites géométriques au troisième trimestre, la pilule semble plus facile à avaler. Cette méthode a été testée avec un certain succès en Première STG. Mais les exemples ont été donnés sur tableur, qui donne une vision globale, et statique, des suites.

| | |
|---|---|
| Créer le point de coordonnées -6 et 3,6 et affecter son nom à la variable a | a=Point(-6,3.6); |
| Faire en sorte que les coordonnées de ce point soient affichées | SetShowValue(a,true); |
| Démarrer une boucle dont l'indice x est un réel allant de -6 à 6 par pas de 0,1 | for(x=-6;x<=6;x=x+0.1){ |
| Dans la boucle, affecter à y la valeur de $\frac{x^2}{10}$ | y=x*x/10; |
| Déplacer le point a vers la position de coordonnées x et y | Move(a,x,y); |
| Attendre 0,4 secondes | Pause(400); |
| Puis passer à la valeur suivante de l'indice | } |

Ce script est en fait un dessin animé, représentant un point dont les coordonnées initiales sont (−6;3,6) et qui se déplace le long de la représentation graphique de la fonction $\frac{x^2}{10}$. Les coordonnées du point sont affichées pendant tout ce trajet, sous la forme du cours (parenthèses), ce qui aide les élèves à réaliser qu'à chaque passage dans la boucle, l'indice varie et comment il varie (en l'occurrence, parce que l'abscisse du point est justement la valeur de l'indice).

La question difficile du TP était celle-ci : Reproduire le tableau de fils ci-dessous à l'aide d'un script :



Le corrigé est ici :

| | |
|---|---|
| Démarrer une boucle pour l'indice i allant de 0 à 10 par pas de 1 | `for(i=0;i<=10;i++){` |
| Créer le point de coordonnées i et 0, mettre son nom dans la variable a et le cacher | `a=Point(i,0);SetHide(a,true);` |
| Créer le point de coordonnées 0 et 10-i et le cacher (son nom est stocké dans b) | `b=Point(0,10-i);SetHide(b,true);` |
| Créer le segment joignant les deux points précédents et affecter son nom à la variable s | `s=Segment(a,b);` |
| Créer le point de coordonnées 0 et i-10, le cacher (son nom est stocké dans b dont l'ancienne valeur n'est plus nécessaire) | `b=Point(0,i-10);SetHide(b,true);` |
| Créer le segment joignant le point a au nouveau point b et affecter à s son nom | `s=Segment(a,b);` |
| Créer le point de coordonnées -i et 0 et le cacher (son nom est stocké dans a dont l'ancienne valeur n'est plus nécessaire) | `a=Point(-i,0);SetHide(a,true);` |
| Créer à nouveau le point de coordonnées 0 et 10-i (on avait perdu sa trace), affecter son nom à b et le cacher | `b=Point(0,10-i);SetHide(b,true);` |
| Créer le segment joignant le nouveau point a et le nouveau point b (son nom est stocké dans s dont l'ancienne valeur n'est plus nécessaire) | `s=Segment(a,b);` |
| Créer le point de coordonnées 0 et i-10 et le cacher après avoir stocké son nom dans la variable b | `b=Point(0,i-10);SetHide(b,true);` |
| Créer le segment joignant a et b et stocker son nom dans s | `s=Segment(a,b);` |
| Fin de la boucle | `}` |

Aucun élève n'a réussi à faire ça en moins d'une heure (trop pris par les deux exercices précédents), mais plusieurs ont été capables de créer ce script chez eux, sur la version de CaRMetal qu'ils avaient téléchargée. Ce qui est plutôt encourageant. Une difficulté non prévue pour ce TP a été la découverte de la fonction affine $10 - x$, que peu d'élèves ont faite, peut-être parce qu'elle est décroissante et que les élèves ne sont pas habitués aux fonctions décroissantes.

### Qu'est-ce que CaRMetal apporte?

La principale raison du choix de l'outil est apparue lors du cinquième TP[7] : c'est qu'il est possible de raisonner sur des objets géométriques comme par exemple des points aléatoires[8], ce qui donne aux activités un impact visuel important. Par exemple, on peut très bien créer deux points, l'un dont les coordonnées sont les moyennes de celles de A et B, l'autre directement défini comme milieu de [AB], et

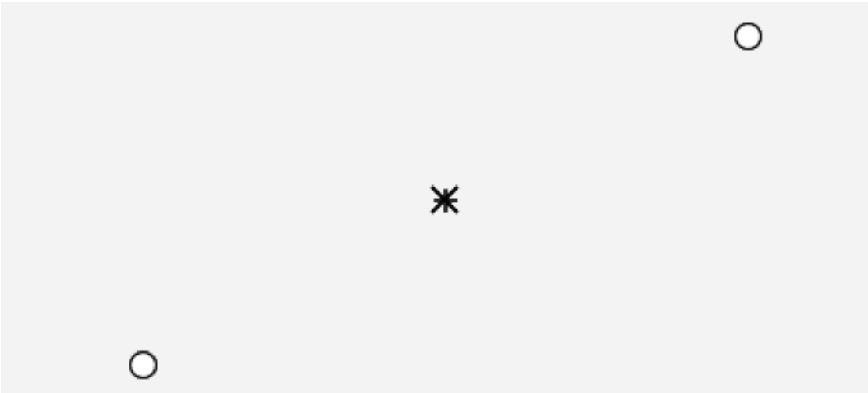
(7) Le compte-rendu est ici :
http://www.reunion.iufm.fr/recherche/irem/spip.php?article302

(8) Le sixième TP quant à lui portera sur le calcul de $\pi$ par la méthode de Monte-Carlo.

*comparer* ces deux points. Ceci dit, cela impose presque qu'on travaille dans le langage de CaRMetal (et plus seulement en JavaScript) pour écrire ceci :

```
Créer un point de nom « A » et de coordonnées aléatoires entre 0 et 1
a=Point("A",Math.random(),Math.random());
Créer un point de nom « B » et de coordonnées aléatoires entre 0 et 1
b=Point("B",Math.random(),Math.random());
Créer le point dont les coordonnées sont les moyennes arithmétiques de celles de A et B
c=Point("(x(A)+x(B))/2","(y(A)+y(B))/2");
Donner à ce point une forme de « x »
SetPointType(c,"dcross");
Créer le milieu de [AB] (son nom est stocké dans la variable m)
m=MidPoint(a,b);
Lui donner une forme de « + »
SetPointType(m,"cross");
```

Sur la figure obtenue, les deux points (en « + » et en « x ») coïncident, *même après avoir déplacé les extrémités à la souris* :



Le TP 6 a permis de parler de statistiques sans quitter le contexte de la géométrie repérée, puisqu'il portait sur le calcul d'une distance moyenne[(9)]

**Bilan provisoire**

L'enthousiasme des élèves (tous les élèves) est nettement supérieur pour ces séances de programmation que pour le cours de « vraies » maths. Le gain est donc plus grand pour les élèves les plus faibles, que pour les autres.

S'il y a une chose pour laquelle la programmation apporte quelque chose de positif, ce sont les moyennes trimestrielles, les élèves les meilleurs en programmation n'étant pas forcément les plus matheux.

Sur l'échantillon de 30 élèves étudié, les plus à l'aise en programmation sont les filles. Est-ce que ça se généralise ?

(9) Difficile de savoir qui, du professeur ou des élèves, est le plus stimulé par cette possibilité. Des détails peuvent se trouver ici :
http://www.reunion.iufm.fr/recherche/irem/spip.php?rubrique83

**Pour en savoir plus**

Un tutoriel, au format pdf, est téléchargeable au bas de la rubrique suivante :
http://www.reunion.iufm.fr/recherche/irem/spip.php7rubrique58

Des éléments de formation se trouvent ici :
http://www.reunion.iufm.fr/recherche/irem/spip.php?article 186

Le cas particulier de la géométrie dans l'espace est traité ici:
http://www.reunion.iufm.fr/recherche/irem/spip.php?article210

La rubrique suivante comprend les corrigés sous forme de **CarScripts**(10) de la quasi-totalité des sujets de l'épreuve expérimentale du bac S 2009 :
http://www.reunion.iufm.fr/recherche/irem/spip.php?rubrique79
(même ceux qui ont été prévus sur tableur ! 41 sujets au total...)

Enfin, une très bonne source d'information sur le sujet, quoique en cours d'élaboration, est le forun qui lui est consacré sur le site de CaRMetal :
http://db-maths.nuxit.net/CaRMetal/forums/viewforum.php?f~6 ;
en particulier, P. M. Mazat y a placé d'intéressants exercices de probabilité-statistiques dont certains sont réalisables en Seconde, dans cette rubrique :
http://db-maths.nuxit.net/CaRMetal/forums/viewtopic.php?t=300

Et un article du portail des IREM résume assez synthétiquement l'état des lieux sur CaRMetal et l'algorithmique :
http://www.univ-irem.fr/spip.php?article27

(10) Néologisme pour abréger l'expression « scripts sous CaRMetal ».