Un parcours d'étude et de recherche sur la géométrie et l'algorithmique en seconde.

Auteurs : Byache Paul, Beaubiat Denis, Spaier Clément, enseignants au lycée Diderot, Marseille, 13013

Intentions initiales :

Le point de départ du travail présenté ici est une conférence de Jill-Jênn Vie sur *La géométrie dans les jeux vidéos*[1]. Ce doctorant, ancien président du concours Prologin[2], apporte un point de vue original et quelques idées simples mais qui ne semblent bizarrement pas toujours évidentes dans le milieu des professeurs de mathématiques. Comme par exemple, qu'il est bien plus motivant de coder des éléments graphiques et des mouvements, plutôt que des calculs ou des programmes abstraits... Et surtout : qu'on peut lier à maintes occasions l'étude de la géométrie avec la programmation des jeux vidéos.

Nous avons essayé de mettre en œuvre certaines de ces idées.

Notre parti-pris essentiel consiste à ne pas considérer l'algorithmique comme une simple illustration de quelques points du cours de mathématiques, mais comme une science qui met en application plusieurs notions du programme de mathématique de seconde. L'algorithmique est un élément de motivation pour les élèves, d'abord par sa forme : elle permet de faire bouger des objets comme dans un jeu vidéo ; mais elle est également motivante sur le fond, parce qu'elle participe à donner une raison d'être aux notions mathématiques enseignées en classe de seconde.

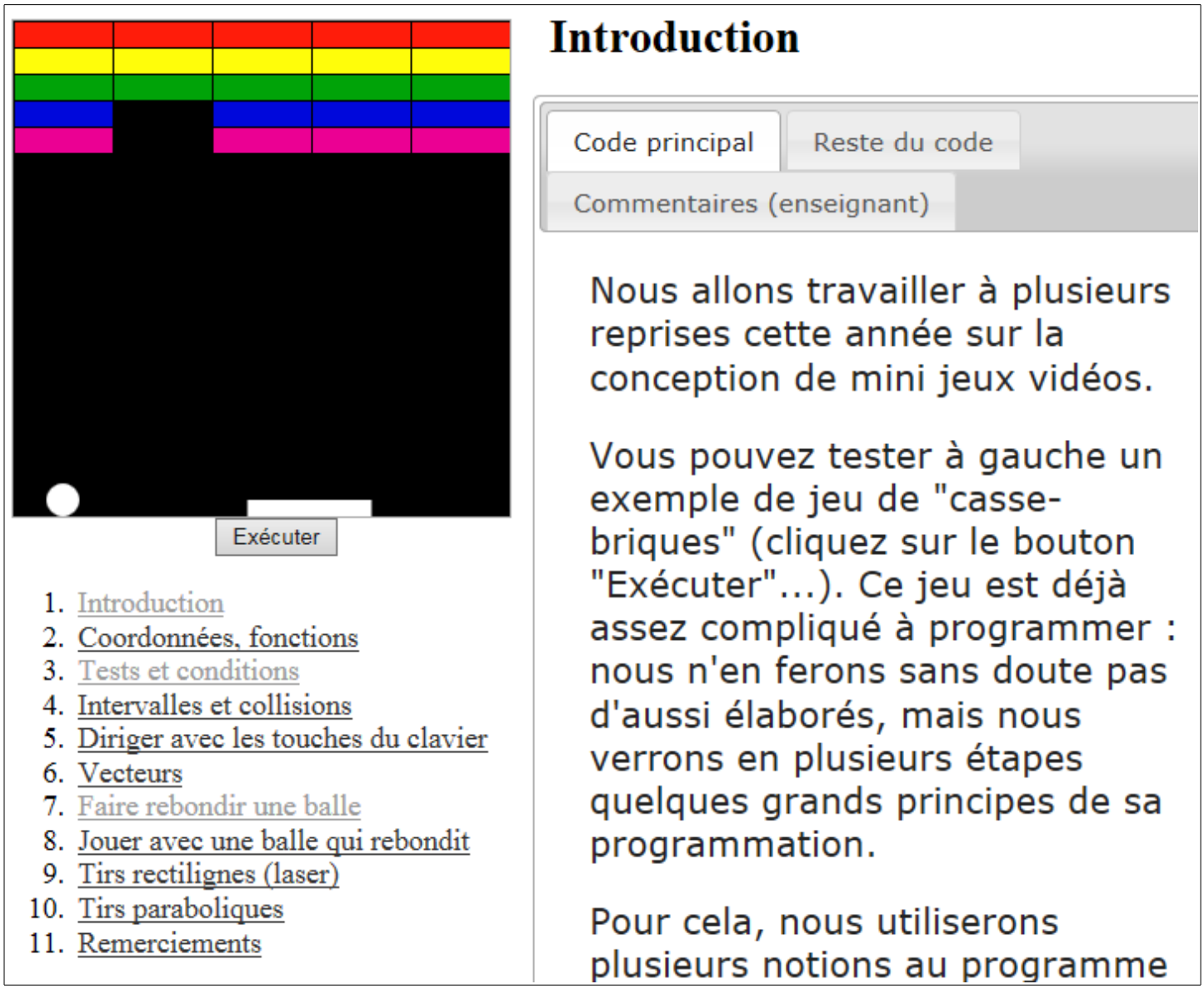
Enfin, ce fil rouge « jeux vidéos », que nous proposons sur l'ensemble de l'année scolaire, redonne une cohérence à toute une partie du programme de seconde. « Comment un ordinateur (qui, à priori, ne sait faire que des calculs) peut-il faire pour proposer un jeu vidéo à l'écran ? » Cette question est génératrice de toute une série de sous-questions auxquelles une grande partie du programme de seconde en mathématiques apportera des réponses.

Nous avons fait travailler nos élèves à plusieurs reprise au cours de l'année sur cette question. Il leur a été demandé d'afficher à l'écran des objets simples comme un disque rouge ou un rectangle vert, de les faire se déplacer, de faire déclencher certains événements comme des explosions à des moments bien précis, de faire rebondir les objets sur les parois de la zone graphique ou encore de leur faire suivre des trajectoires paraboliques. A chaque fois, le travail permettait d'apporter une amélioration dans la programmation des mouvements et utilisait les notions mathématiques qui étaient en cours de travail. Le parcours d'étude et de recherche s'est donc étalé sur une longue période, suivant l'avancement de la progression choisie.

Nous avons également travaillé l'algorithmique suivant d'autres modalités avec nos classes, par exemple en faisant programmer à nos élèves quelques algorithmes sur calculatrice.

1 http://jill-jenn.net/conferences/#geometrie-dans-les-jeux-video

2 https://prologin.org/

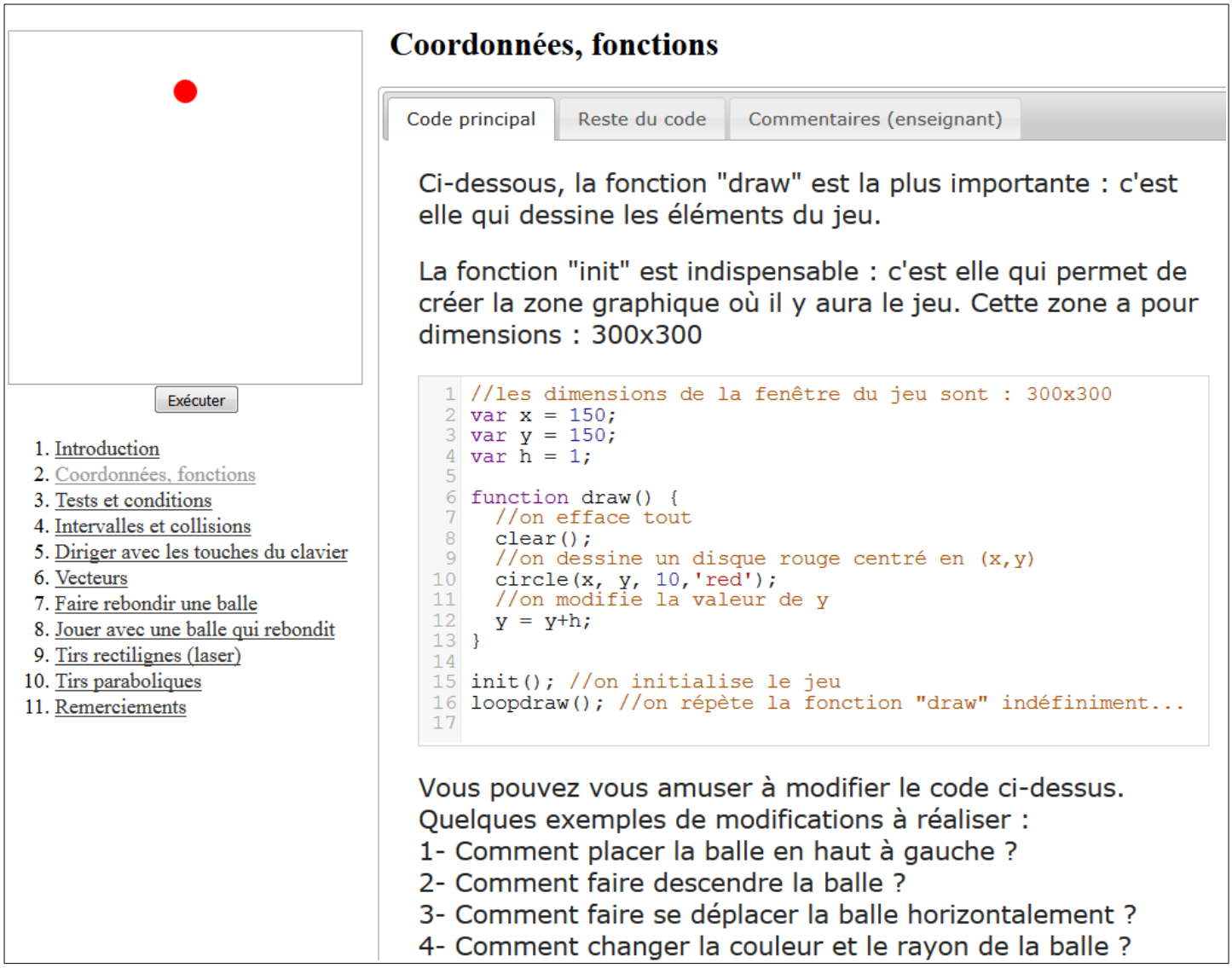


Copie d'écran d'une page web utilisée avec des élèves pour l'introduction au parcours « jeux vidéos »

Nous avons travaillé à trois enseignants de mathématiques sur ce projet, avec trois classes de seconde[3], mais de façons diverses : chacun de nous a fait ses propres expérimentations en fonction de ses habitudes de travail et des réactions de ses élèves.
Tout ce qui est présenté dans cet article est visible en ligne. Pour les collègues intéressés, nous contacter pour obtenir les fichiers sources.

Aspects techniques du travail mené cette année :
Nous sommes dans un premier temps partis du code source utilisé par Jill-Jênn Vie[4]. Cette base de travail fournit une série de pages web contenant du code html / javascript, qui peut être modifié et immédiatement exécuté dans la page elle-même[5]. En revanche, le code produit ne peut pas être enregistré, sauf par l'intermédiaire d'un copié-collé.

3 classées « ZEP » jusqu'à présent, tant que cette classification n'a pas encore disparu pour les lycées !...
4 lui-même l'ayant emprunté à un premier auteur américain : Bill Mill
5 voir la copie d'écran suivante, ou alors ici : http://byachepaul.web4me.fr/GeometrieJeuxVideo/2nde/build/index.html



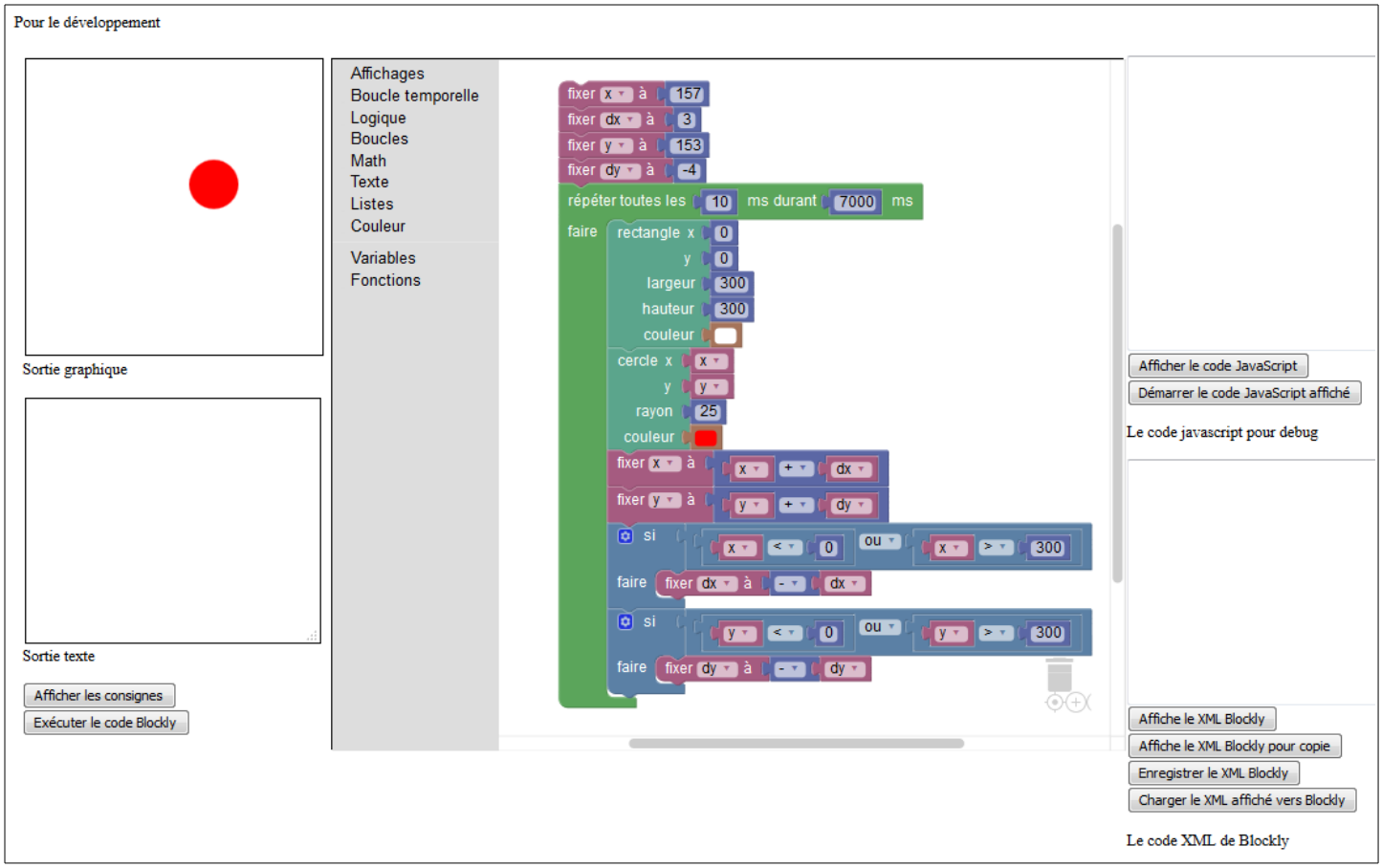
Un travail sur l'utilisation des coordonnées en algorithmique : le disque rouge se déplace à l'écran ; les élèves pouvaient modifier le code javascript et cliquer sur le bouton « Exécuter » pour voir le résultat produit.

Les instructions en javascript sont peut-être de prime abord assez effrayantes pour les élèves, mais ils apprennent rapidement comment intervenir sur le code pour modifier le jeu ou la petite animation produite. Ainsi, ils peuvent prendre conscience de leur capacité à comprendre les principes de la programmation d'une page web.

Des échanges avec un collègue de sciences de l'ingénieur, partageant certaines classes avec nous à l'occasion d'un enseignement d'exploration, nous ont convaincus d'utiliser un langage par bloc, de type « Scratch ». Cela permet une grande cohérence pour les élèves car l'interface utilisée en S.I. pour programmer les modules Arduino et presque la même que celle utilisée en cours de mathématiques. Cependant, le logiciel Scratch est peu adapté pour notre projet car il offre certaines possibilités qui court-circuitent les notions que l'on doit enseigner en mathématiques[6].
La découverte de Blockly nous a permis de disposer d'un environnement de travail où les élèves peuvent coder par bloc mais où le professeur peut contrôler les blocs qui sont disponibles[7].

6 Comme par exemple : « Si la couleur rouge touche la couleur verte, faire... » ou : « Faire rebondir l'objet si le bord est atteint », ou encore : « déplacer l'objet d'un point A à un point B en 2 secondes »… toutes ces instructions sont trop simples à utiliser ! Si on programme sans disposer de ces instructions, en revanche, on aura besoin de calculer des distances en utilisant les coordonnées, de calculer des équations de droites, d'utiliser les coordonnées des vecteurs, etc.

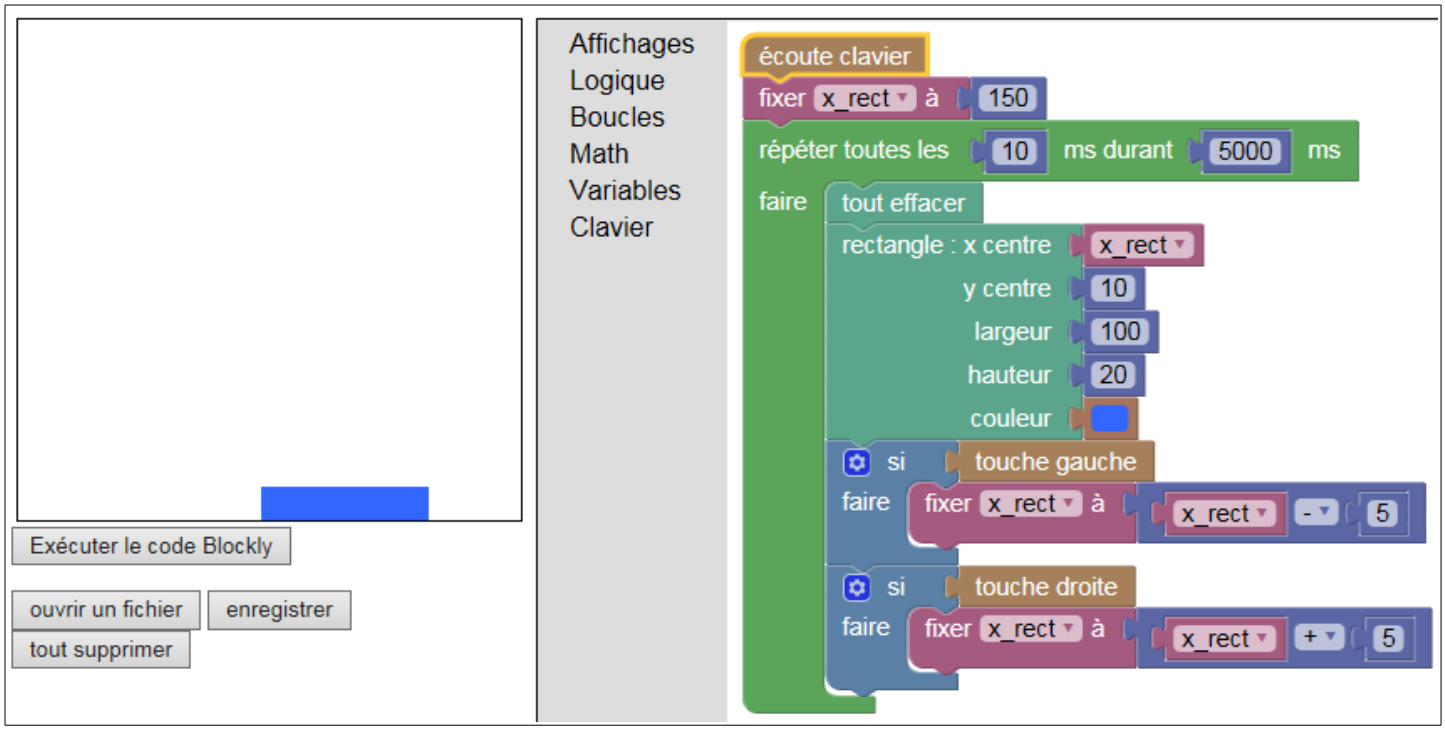
7 Une première version de cet environnement est visible ici :http://beaubiat.fr/Algo-avec-blockly/blockly/blockly_et_graphique/Conteneur_de_developpement.html



L'interface utilisée pour le développement des blocs mis ensuite à disposition des élèves.

Nous avons développé une version utilisable avec les élèves[8], où le nombre de blocs disponibles est volontairement fortement restreint, mais où par ailleurs certains nouveaux blocs ont été ajoutés. Cela permet, d'un côté, de rendre nécessaire l'utilisation des notions du programme de mathématiques de seconde ; et, d'un autre côté, de court-circuiter certaines difficultés propres à l'algorithmique : c'est un peu l'inverse de ce qui se passe dans le logiciel Scratch où les difficultés mathématiques sont levées mais où tout est fait pour permettre de travailler la programmation.
Cet environnement permet également d'enregistrer un algorithme ou d'ouvrir un fichier préalablement enregistré[9].

8 La version 0.3 est par exemple visible ici : http://byachepaul.web4me.fr/blockly/blockly/0.3/
9 Au format « xml ».



Un exemple qui utilise les blocs spéciaux proposés aux élèves :
codage (relativement simple) d'un curseur bleu qu'on pilote avec les flèches du clavier.

Activités proposées aux élèves (aspect pédagogique) :
Rappelons que notre objectif n'est pas uniquement d'enseigner certains rudiments d'algorithmique, mais avant tout d'utiliser l'algorithmique pour faire travailler certaines notions mathématiques d'une façon différente, pour les inscrire dans un cohérence globale et pour leur donner une raison d'être supplémentaire. Plusieurs chapitres du programme de seconde ont toute leur place dans ce parcours : ceux où on étudie les notions de fonction, d'intervalle, les équations de droites, les vecteurs, les expressions du second degré, notamment.

La question initiale évoquée ci-dessus en introduction : « Comment un ordinateur peut-il faire pour proposer un jeu vidéo à l'écran ? », se décline en plusieurs sous-questions plus précises. Chacune d'entre elles justifie l'introduction d'une notion mathématique du programme de seconde et le travail de certains types de tâches mathématiques[10]. En voici quelques-unes que nous avons abordées sous une forme ou une autre cette année :

- « Comment créer l'illusion du mouvement ? »

Réponse :
pour donner l'impression qu'un objet – par exemple un disque rouge – se déplace, on affiche l'objet à un certain endroit de l'écran, puis on l'efface et on le ré-affiche très vite, à un endroit légèrement différent. On répète cela très rapidement et un grand nombre de fois.

La programmation de cet algorithme très simple oblige à utiliser des axes de coordonnées pour définir la position du centre du disque rouge, des variables pour stocker ces coordonnées et les modifier à chaque itération ; et même quelques lignes de code qu'il faudra répéter un grand nombre de fois et qu'on peut placer dans une structure appelée classiquement une « fonction » en algorithmique
Si on veut contrôler le mouvement du disque rouge – vitesse, direction, *etc.* – , il faut travailler sur les coordonnées d'un point.
Ce travail s'inscrit également naturellement dans le chapitre sur les fonctions et leurs représentations graphiques. La similitude entre une courbe, qui est un ensemble de points dont les coordonnées sont liées par une relation, et un mouvement à l'écran, qui se décompose en une succession d'images où

10 On reprend ici une stratégie d'enseignement qui a notamment été théorisée par la théorie anthropologique du didactique, avec les notions d'AER et de PER. Voir par exemple : http://yves.chevallard.free.fr/spip/spip/IMG/pdf/TAD_-_Pistes_Jalons_-_Didirem.pdf

la position de l'objet est également repérée, est un point d'appui. Par contre, les différences de signification des mots « variable » et « fonction » en mathématiques et en algorithmique sont bien sûr à prendre en compte.

- « Comment détecter qu'un objet atteint les bords de la zone graphique ? »

Réponse :
avec des tests du type « si $x>300$, alors... ». (La zone graphique dans notre environnement correspond à $x_{min}=y_{min}=0$ et $x_{max}=y_{max}=300$.)

L'étude de ce type de tests très simples remplit un objectif algorithmique mais donne aussi l'occasion de travailler sur les droites d'équation $y=0$; $y=300$; $x=0$ et $x=300$, qui seront généralisées par la suite.

- « Comment détecter la collision entre deux objets rectangulaires ? »

Réponse :
laissée au lecteur ! (voir l'activité 1 jointe en annexe)

C'est une activité intéressante du point de vue logique, mais surtout : il donne l'occasion d'un travail sur les intervalles et les inégalités, qui est différent des habituelles résolutions d'inéquations.

- « Comment faire en sorte qu'un rayon laser atteigne une cible donnée ? »

Réponse :
en calculant la pente appropriée de la droite qui va représenter ce rayon laser.

La fameuse méthode qui consiste à lire graphiquement la pente d'une droite prend ici tout son sens : on trace un rayon laser point par point en augmentant à chaque itération l'abscisse, de « 1 », et l'ordonnée, d'une valeur égale à la pente…

- « Comment représenter la vitesse d'un objet, de façon à pouvoir programmer facilement, par exemple, son rebond sur les bords de la zone graphique ? »

Réponse :
grâce à un vecteur vitesse et à ses coordonnées.

Le rebond sur une des parois de la zone graphique rectangulaire se programme très simplement, en inversant le signe d'une des coordonnées du vecteur vitesse. C'est donc l'occasion de travailler sur les vecteurs mais également, à nouveau, sur les coordonnées.

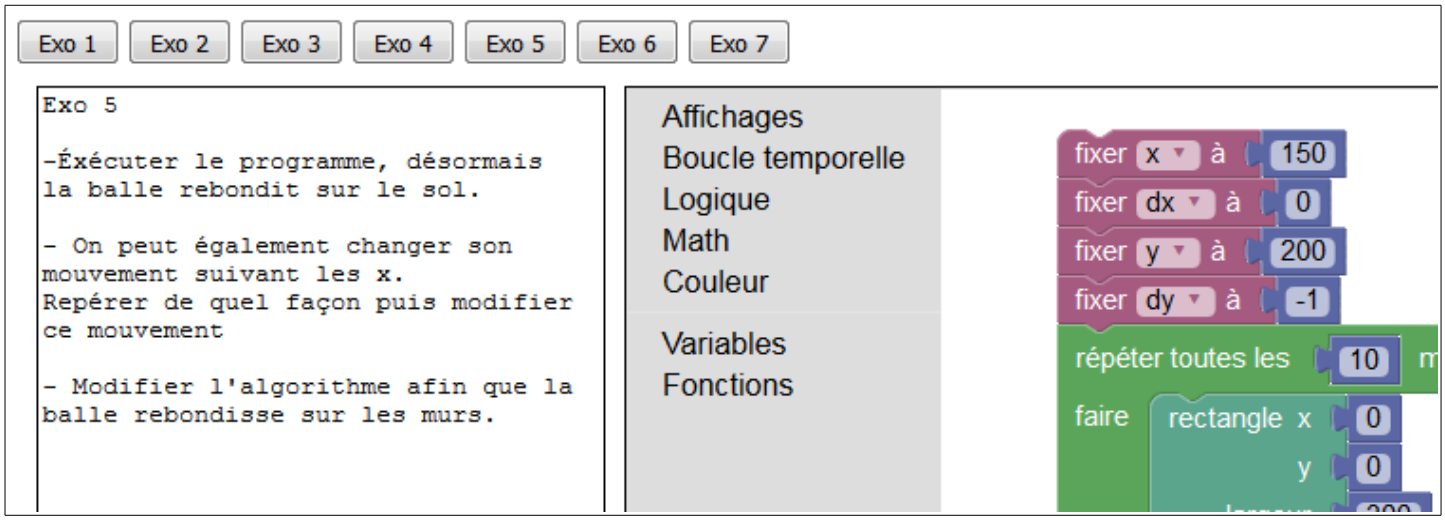
- « Comment donner l'illusion qu'un objet est lancé, puis retombe au sol sous l'action de son poids ? »

Réponse :
si on admet que la trajectoire de l'objet en chute libre est une parabole, il suffit de programmer point par point le dessin d'une parabole à l'écran...

C'est l'occasion de retravailler sur les courbes vues comme ensembles de points et les méthodes pour les dessiner à partir d'un tableau de valeurs… mais si on veut maîtriser la forme de la trajectoire, tout un travail sur les expressions du second degré est possible. Voici quelques pistes simples :
- Pour que la trajectoire ne sorte pas de l'écran, on peut jouer sur l'influence du coefficient de « x^2 ».
- Pour maîtriser l'endroit où l'objet retombe : un travail sur les expressions factorisées « $a(x-x_1)(x-x_2)$ » est tout à fait approprié.

- Pour être certain que l'objet passe par un point A donné, on peut utiliser un travail sur les équations du type « $y_A = ax_A^2 + bx_A$ » où l'inconnue est b.

Les travaux évoqués ci-dessus peuvent être organisés avec les élèves de différentes façons :
- travail en groupe-classe, comme dans l'activité 1 proposée en annexe.
- travail en salle informatique dans un environnement ouvert, avec éventuellement des consignes données sur papier et des algorithmes envoyés aux élèves sous forme de fichier « xml » (voir l'activité 2 proposée en annexe).
- travail en salle informatique dans une suite d'environnements plus restreints incluant des consignes précises (voir l'image ci-dessous[11])



Exemple de série d'exercices où les consignes et les algorithmes sur lesquels doivent travailler les élèves sont entièrement donnés sur écran.

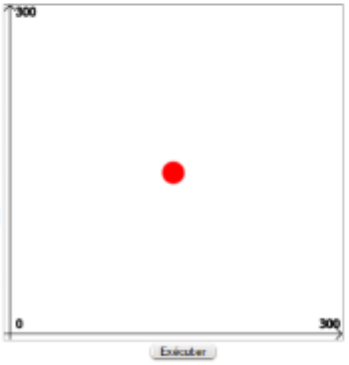
Les rendus demandés aux élèves peuvent être : un fichier contenant un algorithme, une feuille sur laquelle ils ont recopié les modifications effectuées dans le code initial, des explications et un calcul mathématique, *etc*.

Ces travaux sur le thème des jeux vidéos prennent bien sûr toute leur place lors de la mise au point de nouvelles techniques en mathématiques ou en algorithmique, mais ils peuvent aussi être inclus dans les évaluations par exemple : voici un extrait d'un exercice utilisé lors d'un contrôle.

Exercice 6 :

Voici ci-dessous une partie d'un programme informatique affichant à l'écran un disque rouge qui se déplace de la façon suivante :

- il est placé au départ au centre de l'image
- ensuite, il se déplace vers le haut de l'image
- arrivé en haut de l'image, il s'arrête




```
//les dimensions de la fenêtre du jeu sont : 300x300
var x = 150;
var y = 150;
var h = 1;

function draw() {
  clear(); //on efface tout
  circle(x, y, 10,'red');//on dessine un disque rouge centré en (x,y)
  axes(); //on affiche les axes

  //on modifie la valeur de y
  if(y < 300)
      y = y+h;
  else
     y = y;
}

init(); //on initialise le jeu
loopdraw(); //on répète la fonction "draw" indéfiniment...
```

1) Modifiez le programme de telle sorte que le disque rouge soit placé au départ en bas de l'image

11 Plusieurs exercices de ce type sont visibles ici : http://beaubiat.fr/Algo-avec-blockly/blockly/blockly_et_graphique/Exercices_B.html

Constats / évolutions :
Comme nous avons en général manqué de temps pour mettre au point nos activités – tant sur le plan technique que sur le plan pédagogique – , rien n'est parfait et les progrès à réaliser sont nombreux. Cependant, l'intérêt des élèves pour ce type de travail ne s'est pas démenti et la cohérence du parcours sur toute l'année de seconde[12] est d'un grand intérêt.

Sans surprise, nous avons pu constater que les élèves manipulent beaucoup plus facilement le code de type « Blockly » que le « javascript ». Cependant, présenter un peu de javascript au cours de l'année présente aussi des intérêts et ne doit pas à notre avis être écarté trop vite.
Si on fournit un algorithme aux élèves, leur premier réflexe est de le manipuler, de voir comment on peut « trafiquer » le code initial pour obtenir tel ou tel comportement. Ils parviennent alors rapidement à obtenir de petits résultats, mais souvent par des voies surprenantes, voire totalement inattendues[13] !
Si on veut qu'ils lisent réellement le code, de façon linéaire et en étant capable de comprendre comment il fonctionne, il faut batailler un peu et éventuellement avoir recours au dispositif "frontal" avec vidéoprojecteur. Un autre moyen consiste bien sûr à leur demander d'écrire un algorithme à partir de zéro[14] et dans ce cas Blockly est bien plus pertinent que le javascript.

Il est très important, au delà de l'environnement de travail qu'on a préparé, de réfléchir aux consignes que l'on va donner aux élèves et à la façon qu'on aura de gérer la séance. C'est un travail difficile car on a peu de recul : on améliore les choses petit à petit en observant les réactions des élèves.
Il faut aussi signaler que bien sûr, pour certains élèves, la motivation s'érode rapidement devant la difficulté : pour garder l'aspect ludique et éviter au maximum les problèmes trop techniques, il est important de prévoir quelques blocs supplémentaires qui permettent d'obtenir à peu de frais de petites animations intéressantes :
« explosion(x;y) » (dessine une image d'explosion au point de coordonnées (x;y) ;
« collision(x1,y1,w1,h1,x2,y2,w2,h2) » (détecte dans un booléen que les rectangles dont les coordonnées sont indiquées en arguments s'intersectent) ;
« curseur(x_initial;y) » (crée un curseur initialement placé au point dont les coordonnées sont données) ;
« personnage(x;y) » (crée un personnage au point dont les coordonnées sont données)
...[15]

Enfin, une amélioration sans doute très intéressante consisterait à créer un « micro-monde » dans lequel les élèves pourraient programmer leur propre petit jeu-vidéo, y ajoutant des éléments au fur et à mesure de l'avancement de l'année et en parallèle du travail fait en classe. Les efforts entrepris cette année nous rapprochent de cet objectif, mais il reste des limitations, comme la difficulté d'écrire un code relativement long avec des blocs Blockly.[16]

Les choses évoluent très vite actuellement en ce qui concerne les rapports entre mathématiques et algorithmique dans l'enseignement, mais aussi bien sûr au niveau du développement des outils logiciels disponibles. Nous espérons que des lecteurs intéressés signalerons certains outils que nous ne connaissons pas ou reprendrons certains des outils que nous avons développés, pour les

12 On peut travailler dans la même veine avec d'autres classes, par exemple en Tale Arts Appliqués : http://byachepaul.web4me.fr/GeometrieJeuxVideo/Tale/build/ . La question initiale « comment un ordinateur fait-il pour tracer des courbes à l'écran » est très riche pour le programme de ce niveau, où figurent les notions de courbes de fonctions du 2e ou 3e degré, de tangentes, d'équations cartésienne et paramétriques de cercles et d'ellipses, *etc*.

13 Par exemple, si on code l'affichage d'un disque rouge de rayon très grand, on obtient un rectangle… puisque toute la zone graphique est colorée en rouge… Autre exemple : si une boucle itérative code l'affichage d'un disque rouge pendant 10 secondes et qu'on clique plusieurs fois sur le bouton « Exécuter », on obtient plusieurs disques rouges, puisque l'ordinateur lance plusieurs processus dont les temps d'exécution se chevauchent !

14 « from scratch »...

15 récemment, un article très intéressant est paru dans Mathematice, qui détaille notamment la façon de programmer de nouveaux blocs dans Blockly : http://revue.sesamath.net/spip.php?article815

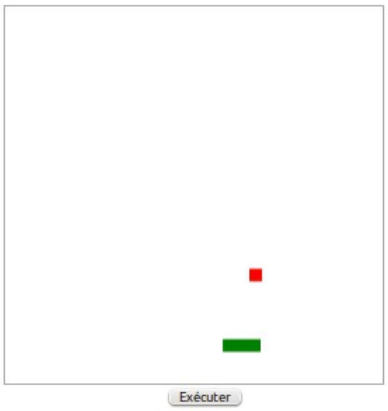
16 Bien sûr, passer dans un des formidables environnements que constituent « Scratch » ou « Snap » peut être tentant, mais ce type d'environnement est « trop riche » car comme nous l'avons expliqué auparavant, il ne permet pas le travail de certains types de tâches mathématiques que nous avons identifiés.

améliorer à leur tour. N'hésitez pas à réagir à cet article...

Annexes :

Ci-dessous, deux exemples d'activités sont proposés. Les fiches ont été utilisées telles qu'elles avec des élèves cette année.
Dans le premier exemple (voir ci-après la fiche-élève suivie de la fiche-professeur), les élèves avaient déjà travaillé à une ou deux reprises sur la modification de lignes de code permettant l'affichage d'objets qui se déplacent dans une zone graphique. Les élèves comprenaient donc très bien de quoi il s'agit et ont été un peu déçus en s'apercevant que l'activité ne porte pas réellement sur de la programmation mais plutôt sur un travail mathématique préalable au codage proprement dit.
Comme l'activité est difficile, elle a été proposée en demi-groupe (15 élèves) et de façon « frontale », avec vidéo-projecteur. Un groupe a mis beaucoup plus de temps que l'autre à s'approprier le problème. L'énoncé a été revu entre les deux séances. Pour un des deux groupes, il a fallu faire une activité préalable de type «figure téléphonée » : un élève dessinait deux rectangles sur son cahier et un autre, au tableau, ne les voyaient pas. Le premier ne donnait que les coordonnées des sommets au second, qui devait décider si les deux rectangles s'intersectaient ou non. Après quelques essais qui consistaient à refaire la figure à partir des données des coordonnées des sommets, le groupe d'élèves a pu commencer à réfléchir à la question posée dans l'activité.
Les exercices 1 et 2 proposés dans la fiche-professeur n'ont pas été travaillés.

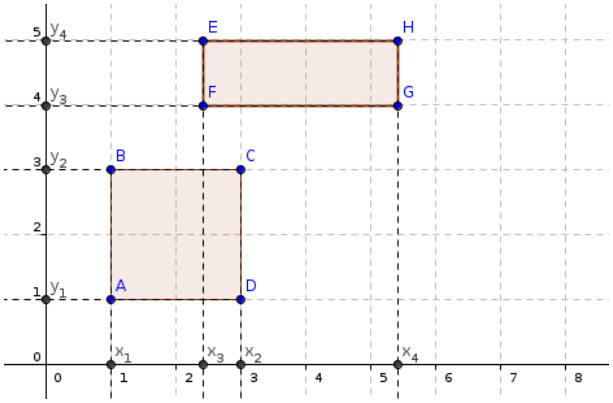
Voici ci-dessous une partie d'un programme informatique affichant à l'écran :
- un carré rouge qui se déplace du haut vers le bas
- et un rectangle vert qui se déplace de la droite vers la gauche.



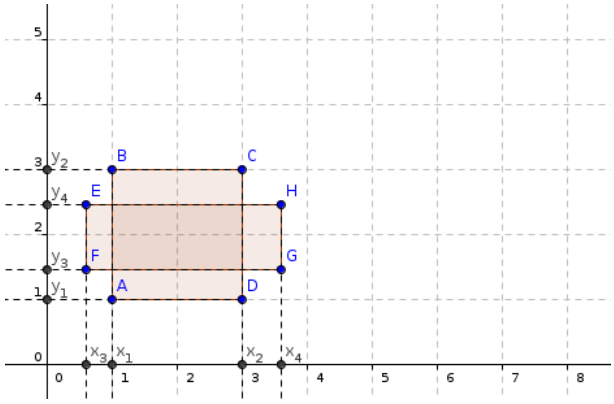
1. Introduction
2. Coordonnées, fonctions
3. Tests et conditions
4. Intervalles et collisions

```
var h = 2;
//centre du carré rouge
var x1 = 150;
var y1 = 300;
//centre du rectangle vert
var x2=100;
var y2=30;

function draw() {
  clear();
  rect(x1, y1, 10, 10,'red');
  if(y1 > 0)
    y1 = y1 - h;
  else{//si le carré rouge atteint le bord,
    y1 = 300; //on le remet en haut
    x1 = alea(0,300); //avec un "x" aléatoire
  }

  rect(x2,y2,30,10,'green');
  if(x2 > 0)
    x2 = x2 - h;
  else //si le rectangle vert atteint le bord,
    x2 = x2 + 300; //on le fait réapparaître sur le bord opposé
}

init(); //on initialise le jeu
loopdraw(); //on répète la fonction "draw" indéfiniment...
```

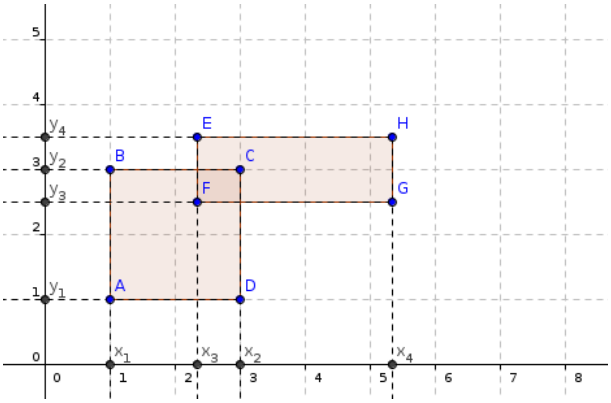
Parfois, le carré et le rectangle entrent en collision. On cherche à détecter ces collisions pour déclencher un événement, comme par exemple une explosion.



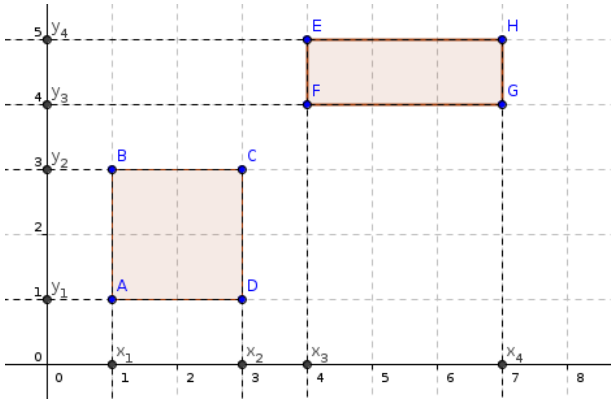
Exemple 1



Exemple 2



Exemple 3



Exemple 4

On suppose qu'on connaît seulement les coordonnées des points A, B, C, D et E, F, G, H :

$$x_1, x_2, x_3, x_4 \quad et \quad y_1, y_2, y_3, y_4$$

Comment peut-on savoir s'il y a une collision ?

Fiche prof de l'activité 1 :

Cette activité est faite pour :
1) voir un cas (autre que dans le contexte des inéquations) où le formalisme des intervalles est utile
2) manipuler un peu les intersections d'intervalles
3) faire un peu de logique
4) faire un peu d'algorithmique

- certains élèves veulent tout de suite écrire du code : il faut expliquer qu'on doit déjà résoudre le problème d'un point de vue théorique : quelle est la méthode qu'on va pouvoir utiliser sachant que la machine ne peut pas « voir » le dessin. Elle ne connaît que les 8 nombres qui sont les coordonnées.

- peut-être que certains élèves vont dire « il y a collision quand un point du rectangle vert est dans le carré rouge » ou quelque chose du même genre. L'exemple 2 permet de corriger cette intuition fausse.

- lorsque les élèves rentrent dans le problème, certains proposent des formulations du genre :
« il y a collision quand l'un des x du carré est entre les x du rectangle » : c'est une bonne piste… mais le professeur doit demander de l'exprimer en utilisant des inégalités ou alors en terme d'intersection d'intervalles : pour qu'il y ait collision, il faut que $[x_1;x_2]\cap[x_3;x_4]\neq\emptyset$

- attention, la condition précédente ne suffit pas :
il y a collision si $[x_1;x_2]\cap[x_3;x_4]\neq\emptyset$ ET si $[y_1;y_2]\cap[y_3;y_4]\neq\emptyset$
(Demander à un élève de venir dessiner un contre-exemple au tableau...)

- Si la classe est épuisée, on peut très bien en rester là : cela permet de signaler l'existence de la fonction « collision(x1,y1,w1,h1,x2,y2,w2,h2) » qui est rendue disponible dans la suite du parcours d'étude et de recherche et qui permet de rajouter des explosions dans les petits jeux vidéo des élèves..

- Si la classe a bien réagi et qu'on peut aller plus loin :
***Exercice* 1 :** formuler avec des intervalles « il n'y a pas collision » … (négation de la condition précédente)

- ensuite,
Exercice 2 : formuler cette négation en utilisant des inégalités à la place des intervalles. En effet, dans un langage de programmation on dispose en général des inégalités mais pas des signes « crochet » et «∩» .

Solution :
il n'y a pas collision si :

$$x_2<x_3 \quad ou \quad x_4<x_1 \quad ou \quad y_2<y_3 \quad ou \quad y_4<y_1$$

- Si on va jusqu'à une formulation en langage de programmation, voici un exemple en javascript (mais c'est beaucoup trop difficile pour rester exigible dans le cadre du programme de seconde).

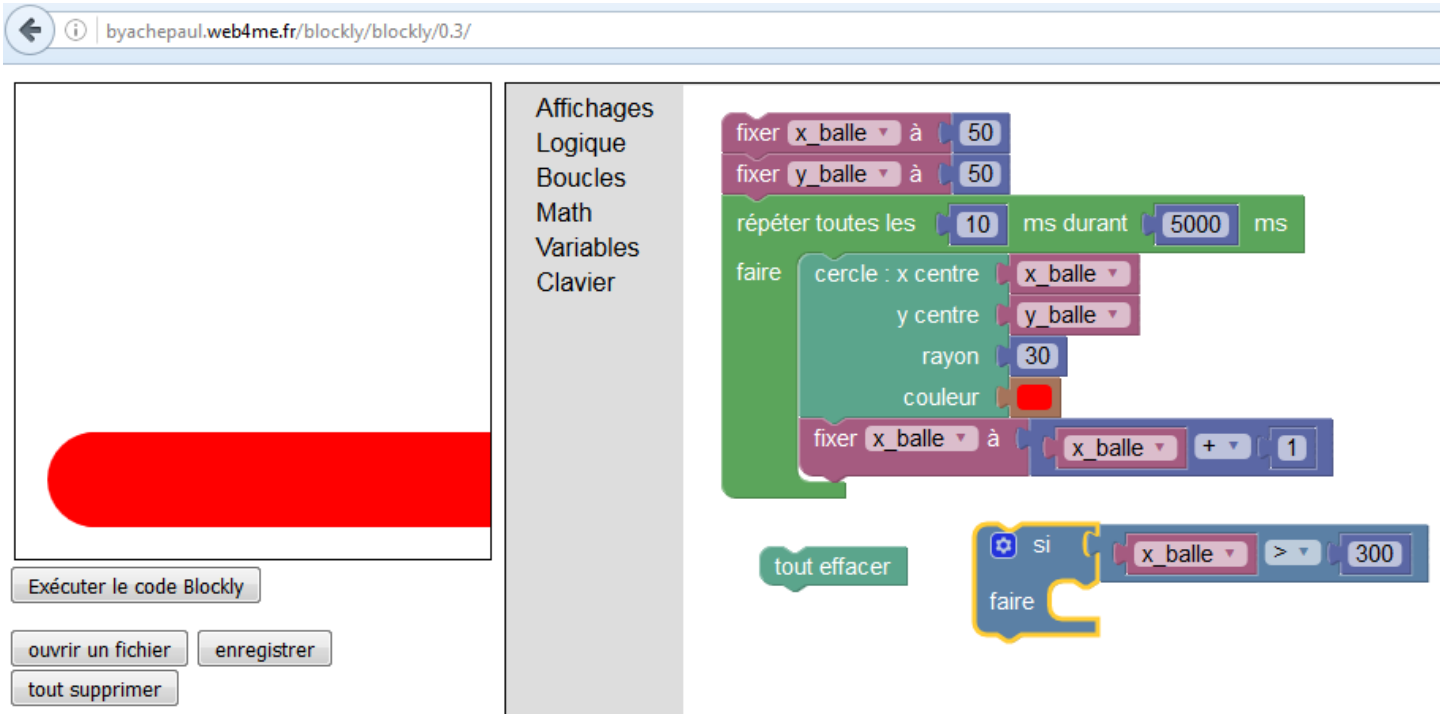
```
function intersect(a1,b1,a2,b2){//intersection de deux intervalles
  //renvoie "false" si l'intersection de [a1 a2] et [b1 b2] est vide
  if(b1<a2 || b2<a1) return false;
  else return true;
}

function collision(x1,y1,w1,h1,x2,y2,w2,h2){
  //renvoie "true" si les deux rectangles se rencontrent
  if( intersect(x1-w1/2,x1+w1/2,x2-w2/2,x2+w2/2) && intersect(y1-h1/2,y1+h1/2,y2-h2/2,y2+h2/2) ) return
true;
  else return false;
}
```

La seconde activité proposée ci-dessous a été utilisée lors d'une reprise du parcours d'étude et de recherche, au cours d'un chapitre sur les vecteurs.
Les élèves étaient en salle informatique, seuls ou à deux devant un ordinateur, par groupe de 15 élèves.
La problématique du parcours était bien connue des élèves et en introduction, l'enseignant montrait une animation représentant une balle rouge qui se déplace dans une zone graphique en rebondissant sur les parois : le but de l'activité était de coder cette animation en « blockly ». C'était la première fois que le langage utilisé était blockly (le début du parcours avait été fait en javascript).
Un des groupes connaissait déjà blockly pour l'avoir utilisé en sciences de l'ingénieur. Dans les deux groupes, la prise en main de l'interface n'a pas posé de problème. Par contre, certains élèves étaient beaucoup plus lents que d'autres, c'est pourquoi le fichier « balle1.xml » a été proposé à la question 5 lors de la séance avec le deuxième groupe : cela donne aux élèves les plus lents la possibilité de disposer d'un fichier fonctionnel pour réfléchir aux dernières questions.

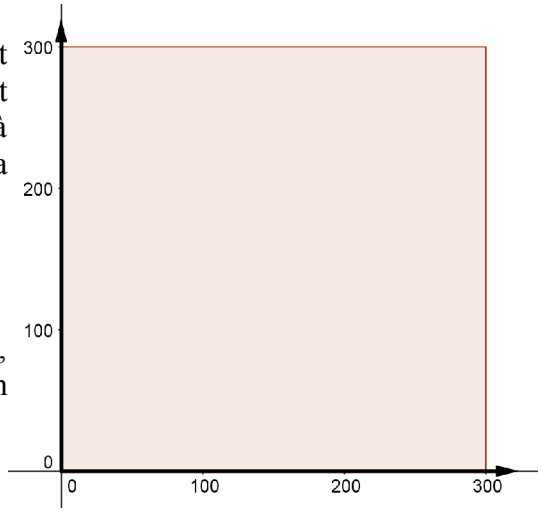
Aller sur la page : http://byachepaul.web4me.fr/blockly/blockly/0.3

ouvrir le fichier : balle0.xml

En « exécutant le code Blockly » de ce fichier, vous verrez une balle rouge se déplacer vers la droite et dessiner ainsi une large bande rouge.



La fenêtre graphique dans laquelle se dessine cette balle rouge est rapportée à un repère dont l'origine est dans le coin en bas à gauche et dont les axes sont gradués de 0 à 300. Ce repère ne s'affiche pas à l'écran, mais vous pouvez l'utiliser pour coder les déplacements de la balle.



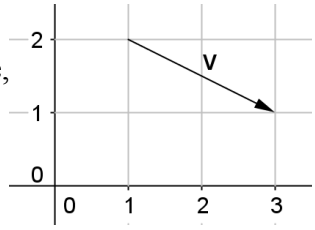
Travail à faire :

1) Faire en sorte que la balle s'efface à chaque fois qu'elle a été dessinée, avant de se redessiner à un autre endroit : on aura ainsi l'impression d'un mouvement au lieu de voir une bande rouge.

2) Faire en sorte que la balle démarre du centre de la zone graphique.

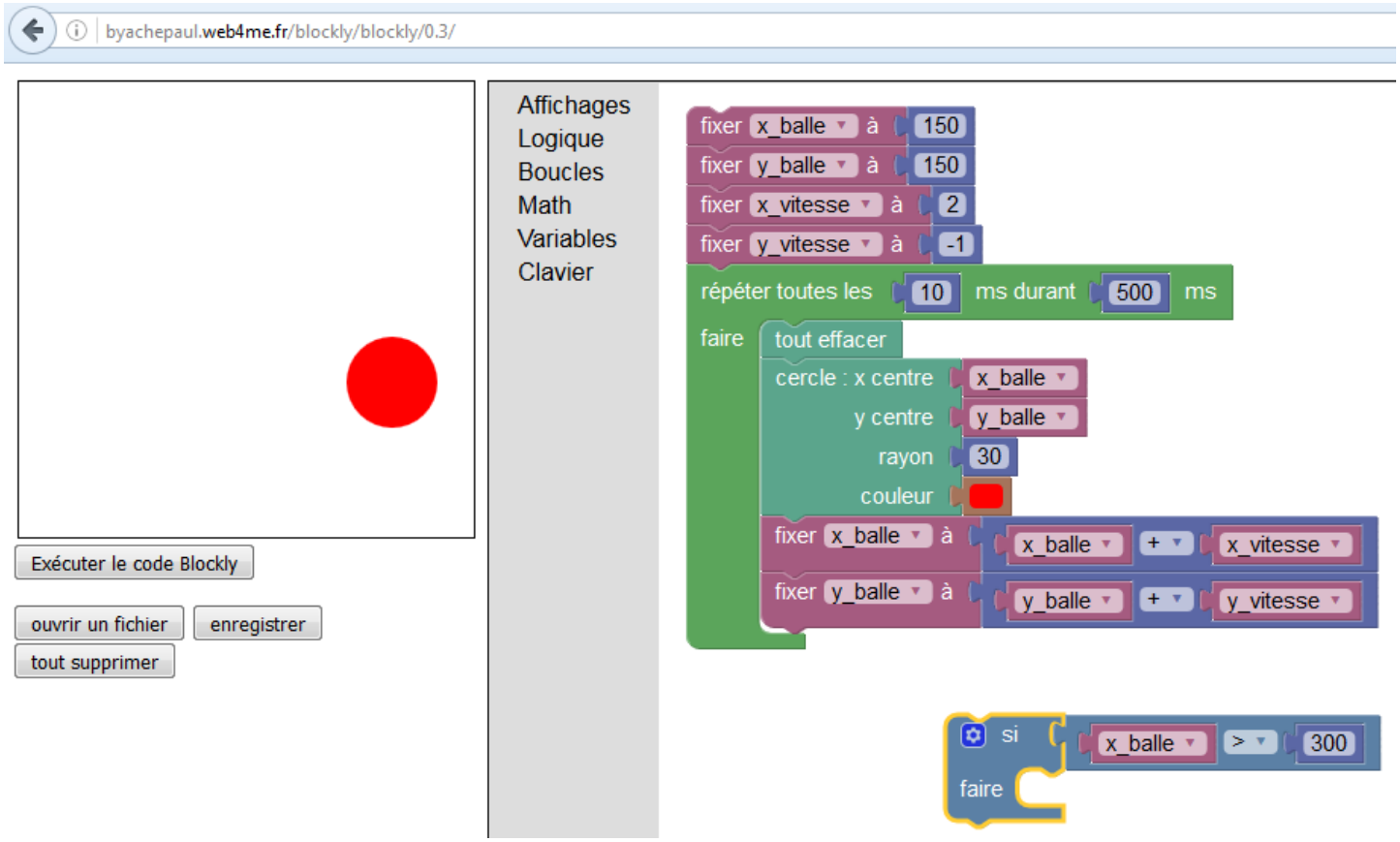
3) Faire en sorte que l'ordonnée du centre de la balle se modifie à chaque fois que la balle est redessinée (ainsi, la balle ne va pas se déplacer uniquement dans la direction horizontale)

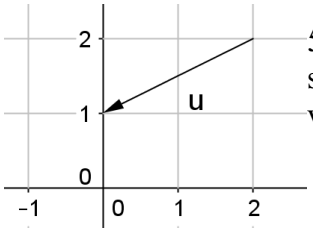
4) Faire en sorte que la balle se déplace suivant le vecteur vitesse $\vec{v}$ dessiné ci-contre, à droite.



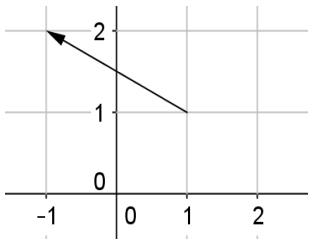
TSVP (Tournez s'il vous plaît.)

Ouvrir le fichier « balle1.xml », qui contient un corrigé des questions précédentes et deux nouvelles variables : x_vitesse et y_vitesse.
Le but de la suite du travail est de programmer une balle qui rebondisse sur les parois de la zone graphique.





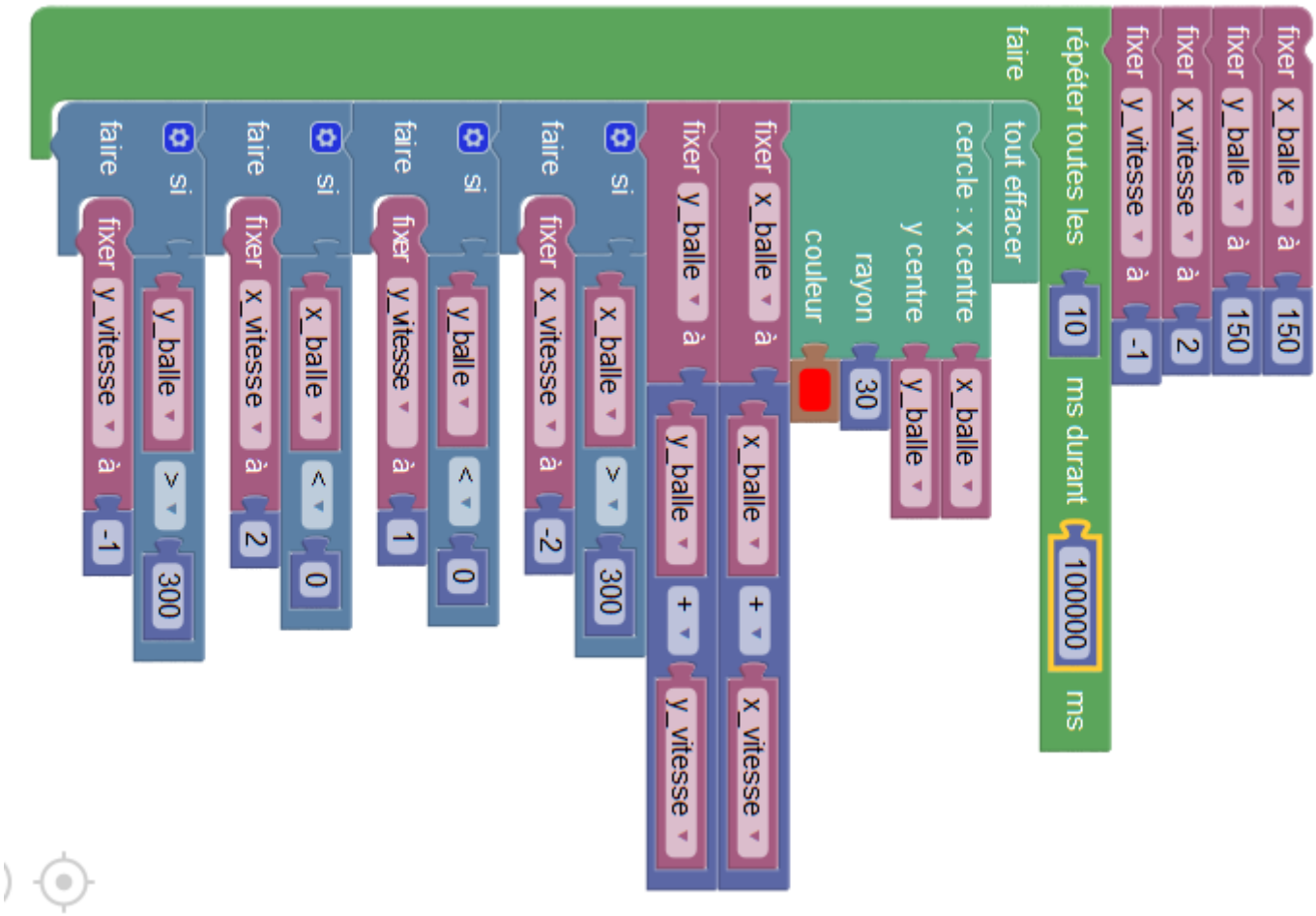
5) Faire en sorte que si « x_balle » devient supérieur à 300, le déplacement de la balle suive le vecteur vitesse $\vec{u}$ dessiné ci-contre, à gauche : cela donnera l'impression visuelle que la balle rebondit sur la paroi de droite de la zone graphique.



6) Faire en sorte que si le centre de la balle atteint le bas de la zone graphique, elle rebondisse en suivant le vecteur dessiné ci-contre à droite :

7) Ajouter des instructions pour que la balle rebondisse sur les 4 parois de la zone graphique.

Correction :



Une autre solution plus élégante :