Type d'activité

Integer Chaser

Pré-requis :

- Notion de nombres relatifs
- Opérations avec les nombres relatifs
- Notion d'algorithme

Objectifs :

- Travailler les calculs avec des nombres relatifs de manière ludique (4ème)
- Revoir la notion d'algorithme et l'appliquer pour programmer des déplacements (4ème)
- Travail sur la représentation de l'information (2nde SNT)
- Programmation Javascript (1ère NSI)

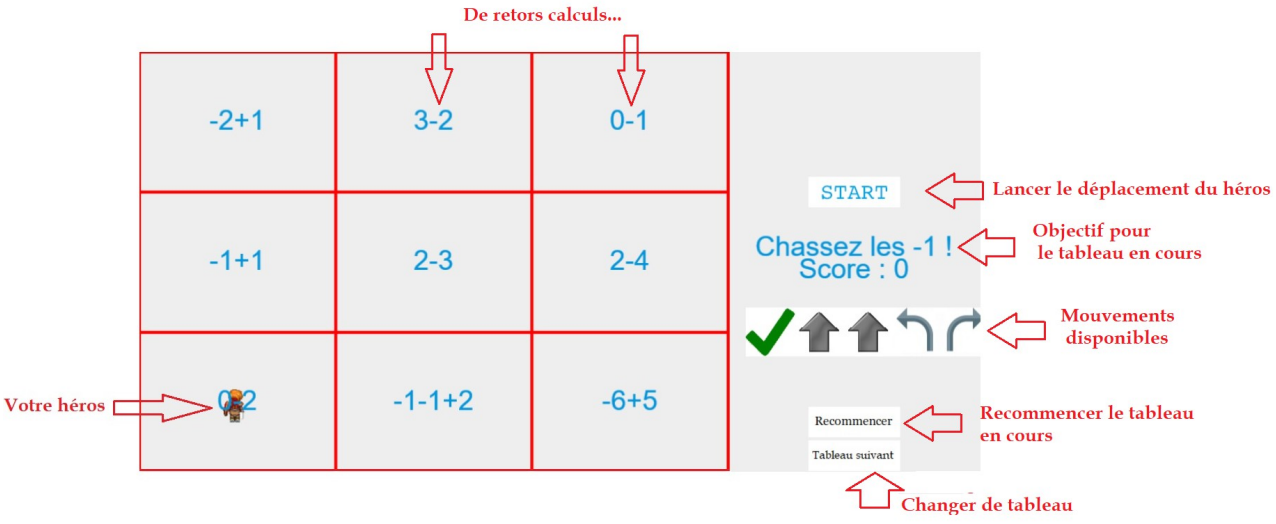
Situation :

Votre héros se retrouve piégé dans un labyrinthe infesté de hideux nombres relatifs. Seuls des déplacements minutieusement commandés et d'inégalables compétences en calcul vous permettront de vous en extirper sain et sauf !

Principe :

Chaque tableau est une grille constituée de cases et propose un objectif. Dans chaque case se présente un calcul avec des nombres relatifs. L'élève doit déplacer le héros dans la grille à l'aide de commandes, dans le but de sélectionner tous les calculs dont le résultat est égal à l'objectif (le cas échéant, on dira que le calcul est *correct*). Lorsque l'élève valide un calcul correct, de nouveaux mouvements lui sont offerts qui lui permettent de poursuivre sa progression dans la grille. Dans le cas contraire, il se peut que des mouvements lui soient également offerts, mais qui ne lui permettront pas de finir le tableau. La victoire est assurée lorsque l'élève a trouvé tous les calculs corrects.

Présentation de l'écran de jeu :



Dans ce tableau, l'objectif est -1. L'élève doit donc trouver tous les calculs dont le résultat est -1, puis doit déplacer son héros sur les cases de ces calculs pour les valider, ce en créant des algorithmes de déplacement avec les mouvements qu'il a à sa disposition.

Déplacer le héros et valider un calcul :

Le héros peut être orienté de quatre manières différentes.

	Sud		Est
	Ouest		Nord

Les mouvements élémentaires à disposition de l'élève sont les suivants :

	Avancer le héros d'une case dans la direction où il est orienté (si le héros est orienté vers le Nord, il avance d'une case vers le haut ; s'il est orienté vers l'Est, il avance d'une case vers la droite ; etc.)
	Orienter le héros sur *sa* gauche (si le héros est orienté vers le Nord, il le sera alors vers l'Ouest ; s'il est orienté vers l'Est, il le sera alors vers le Nord ; etc.)
	Orienter le héros sur *sa* droite (si le héros est orienté vers le Nord, il le sera alors vers l'Est ; s'il est orienté vers l'Est, il le sera alors vers le Sud ; etc.)
	Valider un calcul

Pour programmer un déplacement, l'élève doit créer un algorithme en sélectionnant une suite de déplacements élémentaires. Pour ce faire, il doit déplacer **au-dessus du bouton « Start »**, à l'aide de la souris, les mouvements disponibles (initialement sous le bouton « Start ») qu'il souhaite utiliser.

Par exemple ici, l'élève souhaite orienter son héros sur sa gauche, puis avancer d'une case, puis valider le calcul. Il arrivera ainsi sur la case « 3-2 » (et validera donc au passage un calcul incorrect…) Il n'est pas indispensable d'utiliser tous les mouvements disponibles.

Remarque ergonomique : les mouvements choisis par l'élève n'ont pas besoin d'être parfaitement alignés verticalement, ainsi qu'on le constate sur la copie d'écran ci-dessus. En revanche, ils doivent être correctement ordonnés de gauche à droite...

Si le calcul validé est correct, l'élève augmente son score et de nouveaux mouvements lui sont offerts qui lui permettent de continuer ses déplacements jusqu'à avoir trouvé tous les calculs corrects et ainsi achever le tableau.

Si le calcul validé est incorrect, le score n'augmente pas. Il se peut que l'élève ne reçoive aucun mouvement, ou bien qu'il reçoive traîtreusement des mouvements qui ne lui permettront pas d'achever le tableau… Il lui faudra alors faire preuve de persévérance et recommencer le tableau (en appuyant sur le bouton adéquat).

Exploitation en classe avec une classe de 4ème :

1ère séance :

- Le professeur procède à une démonstration au tableau afin d'expliquer le but du jeu et de bien faire saisir aux élèves le principe de déplacement du héros et de validation des calculs. En principe les élèves sont familiers, notamment par le biais de Scratch, de la notion de « Tourner sur sa gauche » et « Tourner sur sa droite ».
- Les élèves sont laissés en autonomie, individuellement ou en binôme, afin d'essayer de compléter le premier tableau. Les plus rapides pourront essayer le deuxième tableau, de plus large dimension et présentant des calculs plus élaborés.

2ème séance :

Lors d'une deuxième séance, il pourrait être proposé aux élèves de créer leur propre tableau.

L'élève doit définir un objectif pour son tableau, puis un calcul pour chaque case. L'élève doit également réfléchir aux mouvements qui sont offerts pour chaque case. Cette dernière étape est particulièrement subtile : il faut que les mouvements offerts permettent de continuer à se déplacer efficacement, mais également que les mouvements offerts n'indiquent pas non plus de manière évidente où sont les prochaines cases correctes…

Une version papier est créée par chacun, puis les élèves défient leurs camarades avec leur création et se critiquent mutuellement de manière constructive.

Exploitation en classe avec une classe de 2nde en SNT, ou de 1ère en NSI :
(ou éventuellement à la suite, avec une classe de 4ème curieuse et motivée)

On reprend rapidement les idées précédentes exploitées en classe de 4ème. On s'intéresse ensuite à la représentation de l'information dans le programme. Comment est défini un tableau de ce jeu ? Quels en sont les caractéristiques ? Comment représenter informatiquement ces caractéristiques dans une structure complexe ?

```
var tableau = { taille: {x:3, y:3},
                perso : {x:0, y:2, orientation: 'N'}, //position initiale
                question : [['-2+1','-1+1','0-2'],
                            ['3-2','2-3','-1-1+2'],
                            ['0-1','2-4','-6+5']],
                cadeau : [[['E','F','F','L','R','R'],['E','E','E'],''],
                            ['',['E','L','L','F','F','R','R'],''],
                            [['E','L','L','R','F','F'],'',['E','F','F','L']]],
                objectif : -1,
                score_a_atteindre : 4
}
```

Dans le code Javascript (« *main.js* »), un tableau est défini par un 6-uplet tel l'exemple ci-dessus.
Les calculs proposés figurent dans le tableau bi-dimensionnel nommé *question*.
Le tableau *cadeau*, de même dimension que le tableau *question*, correspond aux mouvements offerts lorsque l'élève valide un calcul, où F (*forward*) figure un déplacement vers l'avant, L (*left*) un déplacement vers la gauche, R (*right*) un déplacement vers la droite et E (*eat*) une validation.

On peut proposer aux élèves de faire le lien entre les informations contenues dans cette structure et les informations à l'écran dans le jeu. Après analyse, les élèves modifient ces informations pour implémenter leur propre tableau.

Améliorations possibles :

Ce modeste jeu peut ouvrir la voie à de multiples développements, éventuellement envisageables dans le cadre de projets d'apprentissage de Javascript avec des élèves de NSI :

- l'aspect graphique et esthétique peut clairement être amélioré
- de nouveaux tableaux pourraient être proposés, notamment par les élèves
- des tableaux pourraient être proposés où les questions ne sont pas des calculs, mais par exemple des affirmations qui peuvent être vraies ou fausses, abordant divers points des programmes de collège ou de seconde
- une possibilité de « Annuler le dernier mouvement » pourrait éviter de recommencer un tableau depuis le début en cas d'erreur de l'élève
- de nouveaux déplacements pourraient être proposés, permettant ainsi une complexification algorithmique : création de boucles, voire de structures conditionnelles, utilisables sur des tableaux plus vastes et plus élaborés
- ...