Les exercices de géométrie dans Mathenpoche

Sur un exemple simple, nous allons détailler comment créer un exercice de géométrie utilisant les instruments virtuels.

Exercice :
réaliser un exercice comportant cinq questions demandant à l'élève de mesurer la longueur d'un segment dessiné.

I. Programmation de l'exercice :

1. Création d'un modèle du point

La programmation de l'exercice se fait dans l'image 3 du calque des actions mais avant de commencer nous avons besoin de créer un clip « point » qui nous servira de modèle pour créer les points que nous afficherons à l'écran. Ce sera le seul objet créé dans l'interface graphique de Flash ... tout le reste de la programmation, dont les figures présentées à l'élève sera créé par ActionScript !

La zone de texte servant à le nommer est dynamique (utilisant la variable locale « nom »), ce qui nous permettra de tirer aléatoirement le nom des points avec un code similaire à celui-ci :

```
nbre_points=2;
code = [];
for (i=1; i<=nbre_points; i++) {
        code[i] = random(26)+65;
        for (j=1; j<i; j++) {
                if (code[i] == code[j]) {
                        i = i-1;
                }
        }
}
```

Remarque : ne pas oublier de vérifier si le code ASCII d'une lettre est tiré pusieurs fois.

2. Création de la figure

Maintenant que notre modèle de point est créé nous pouvons envisager la création du reste de notre figure, basée sur la duplication du clip « point ».
Par exemple, pour une figure utilisant deux points, on utilisera le code suivant :

```
for (i=1; i<=nbre_points; i++) {
        duplicateMovieClip(point, "point"+i, 10+i);
        set("_root.point"+i+".nom", String.fromCharCode(code[i]))
}
```

Remarque :
on pourrait aussi créer le modèle de point et le laisser dans la bibliothèque de l'animation (en le déclarant « exportable pour ActionScript ») plutôt que de le placer sur l'espace de travail. Dans ce cas la création de la figure fera appel à la fonction « attachMovie » et non « duplicateMovieClip ».
Ce qui donnerait :

```
var nom= new Array();
for (i=1; i<=nbre_points; i++) {
        this.attachMovie("nom_du modele", "point"+i, 10+i);
        nom[i] = String.fromCharCode(code[i]);
        this["point"+i].nom = nom[i];
}
```

Les deux clips créés sont respectivement nommés « point1 » et « point2 » et s'affichent sur l'écran avec les lettres correspondant aux codes « code[1] » et « code[2] » définis plus haut. Leur position est définie aléatoirement dans la zone de travail.

Ne nous reste plus qu'à tracer le segment à mesurer :

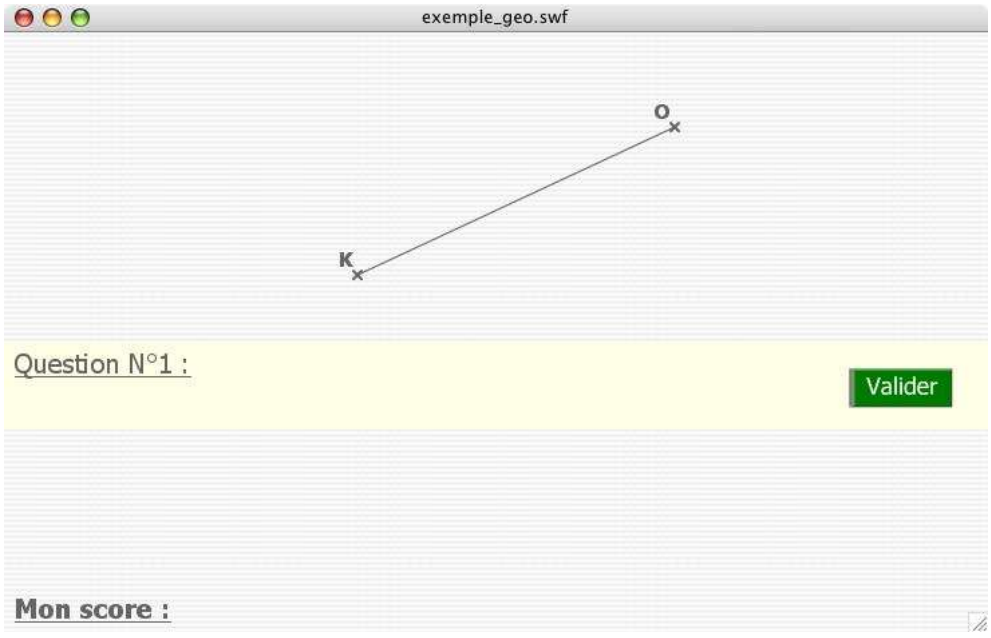
```
_root.createEmptyMovieClip("segment", 10+nbre_points+1);
with (segment) {
        lineStyle(1, 0x666666, 100);
        moveTo(point1._x, point1._y);
        lineTo(point2._x, point2._y);
}
```

... et à calculer sa longueur :

```
longueur = Math.sqrt(Math.pow(point1._x-point2._x, 2)+Math.pow(point1._y-point2._y, 2));
longueur = Math.round(longueur/3)/10;
longueur = longueur.toString();
if (longueur.indexOf(".")>0) {
        temp = longueur.split(".");
        longueur = temp[0]+","+temp[1];
}
```

Remarques :
- un cm de la règle Mathenpoche correspond à 30 pixels de Flash ;
- nous devons transformer cette longueur en une chaîne de caractères puisque la réponse de l'élève sera donnée sous la forme d'une chaîne de caractères. Par exemple, l'entier 4 n'est pas égal au caractère « 4 ».

La figure est prête :



3. Création des zones des textes et de la réponse de l'élève :

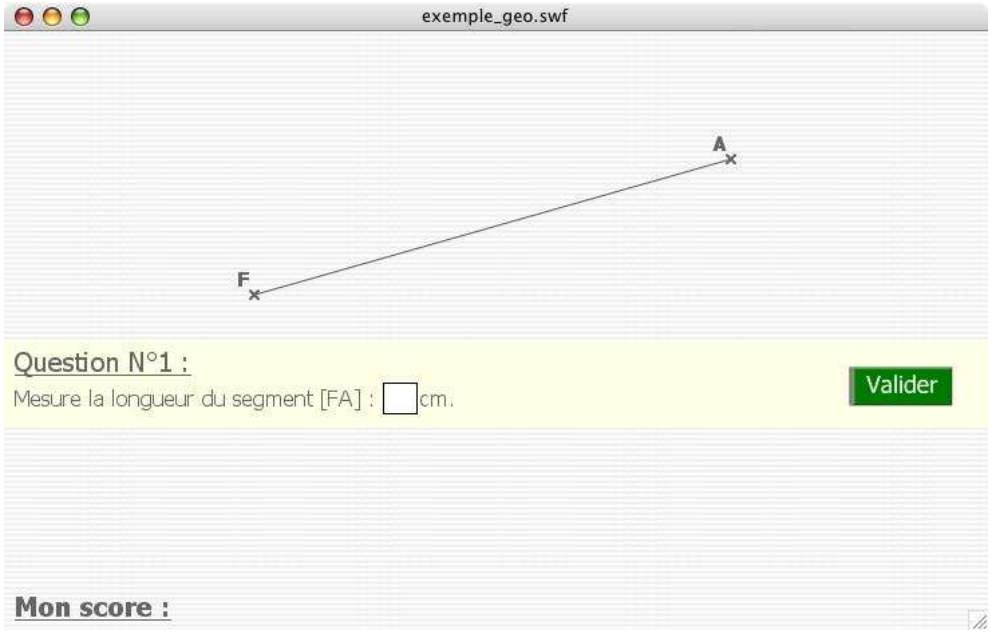
Avec la méthode vue dans les exercices pécédents, nous allons maintenant créer les zones de consignes accueillant respectivement la consigne et la réponse de l'élève :

```
	createTextField("consigne1", 1, affichage_question._x, affichage_question._y+affichage_question._height+3, 0, 30);
	 with (consigne1) {
		setNewTextFormat(format_consigne);
		background = false;
		border = false;
		align = "left";
		text = "Mesure la longueur du segment ["+point1.nom+point2.nom+"] :";
		textColor = "0x666666";
		type = "static";
		autoSize = true;
		selectable = false;
	 }
	createTextField("zone_saisie", 2, consigne1._x+consigne1._width+3, consigne1._y, 0, 30);
	 with (zone_saisie) {
		setNewTextFormat(format_saisie);
		background = true;
		border = true;
		maxChars = 4;
		restrict = "0-9.,";
		align = "center";
		text = "";
		textColor = "0x7484A6";
		type = "input";
		autoSize = true;
		tabIndex = 1;
	 }
	createTextField("consigne2", 3, zone_saisie._x+zone_saisie._width+3, zone_saisie._y, 0, 30);
	 with (consigne2) {
		setNewTextFormat(format_consigne);
		background = false;
		border = false;
		align = "left";
		text = "cm.";
		textColor = "0x666666";
		type = "static";
		autoSize = true;
		selectable = false;
	 }
btn_valider.tabIndex = 2;
btn_suite.tabIndex = 3;
btn_valider._focusrect = false;
btn_suite._focusrect = false;
if (numquestions>1) {
	Selection.setFocus("zone_saisie");
}
```

Pour permettre indifféremment l'utilisation des caractères « , » ou « . » lors de la saisie de de la virgule par l'élève et replacer automatiquement les zones de texte les unes par rapport aux autres, nous ajouterons ceci :

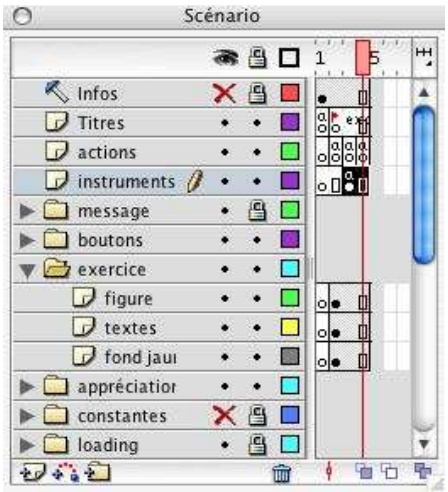
```
	zone_saisie.onChanged = function() {
		zone_saisie.textColor = "0x7484A6";
		if (zone_saisie.text.indexOf(".")>0) {
			temp = zone_saisie.text.split(".");
			zone_saisie.text = temp[0]+","+temp[1];
		}
		consigne2._x=zone_saisie._x+zone_saisie._width+3
	 }
```

Notre exercice se précise :



4. Intégration des instruments :

Pour intégrer les instruments à notre exercice, nous allons copier l'image contenue dans le fichier « instruments_geo.fla » dans notre animation et ne garder que ce qui nous intéresse.
Il est préférable, pour que les instruments « se rangent » automatiquement d'une question à une autre de placer le clip à partir de l'image 3 de notre animation.



Puisque nous n'avons besoin que de la règle graduée dans cet exercice, nous pouvons effacer tous les clips correspondant aux autres instruments.
Pour les mêmes raisons nous allons également « nettoyer » le code de l'image qui contient les instruments afin de ne garder que ce qui nous est nécessaire.

Ici, le clip « menu » (placé au niveau 99 pour être au dessus des autres éléments de la figure) se contentera d'attendre les évènements clavier (enfoncement des flèches du clavier) ou souris liés à la règle.

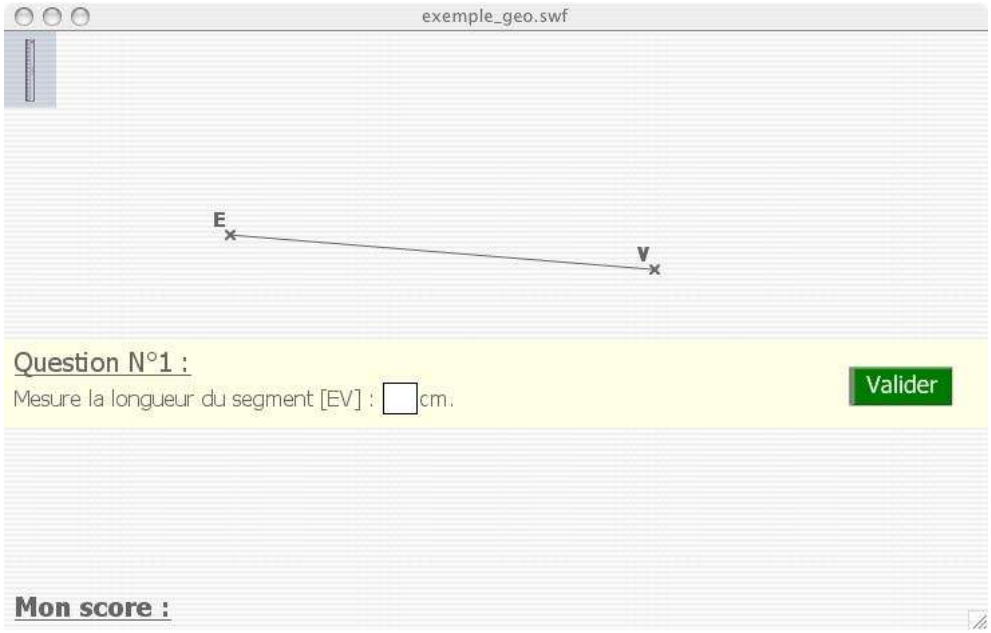
Les actions de l'image se limiteront donc à :

```
_root.n = 1;
//placer les instruments au desssus :
 _root.createEmptyMovieClip("instruments", 99);
 menu.swapDepths(instruments);
 menu.onEnterFrame = function() {
	if (_root.regle_select == 1) {
		_root.menu.requerre_actif._visible = false;
	} else {
		_root.menu.regle_actif._visible = false;
	}
	if (_root.n == 1) {
		if (_root.regle_select == 1) {
			if (Key.isDown(16)) {
				if (Key.isDown(37)) {
					rot2 = _root.menu.regle._rotation;
					rot2 = rot2-10;
					_root.menu.regle._rotation = rot2;
				} else if (Key.isDown(39)) {
					rot2 = _root.menu.regle._rotation;
					rot2 = rot2+10;
					_root.menu.regle._rotation = rot2;
				} else if (Key.isDown(38)) {
					y = _root.menu.regle._y;
					y = y-10;
					_root.menu.regle._y = y;
				} else if (Key.isDown(40)) {
					y = _root.menu.regle._y;
					y = y+10;
					_root.menu.regle._y = y;
				}
			} else if (Key.isDown(39)) {
				rot2 = _root.menu.regle._rotation;
				rot2 = rot2+1;
				_root.menu.regle._rotation = rot2;
			} else if (Key.isDown(37)) {
				rot2 = _root.menu.regle._rotation;
				rot2 = rot2-1;
				_root.menu.regle._rotation = rot2;
			} else if (Key.isDown(38)) {
				y = _root.menu.regle._y;
				y = y-1;
				_root.menu.regle._y = y;
			} else if (Key.isDown(40)) {
				y = _root.menu.regle._y;
				y = y+1;
				_root.menu.regle._y = y;
			}
		}
	}
 }
```

Pour plus de confort au niveau de l'utilisation, on peut aimanter les points afin de positionner la règle précisément lorsqu'elle est placée à proximité de l'un d'eux. Pour ce faire, sur le clip « regle », on ajoutera :

```
on (release) {
	if (_root.n == 1) {
		_root.regle_select = 1;
		startDrag(_root.regle);
		_root.n = 0;
	} else if (_root.n == 0 && _root.regle_select == 1) {
		stopDrag();
		_root.n = 1;
		for (i=1; i<=_root.nbre_points; i++) {
		test=Math.abs(_root.menu.regle._x-getProperty("_root.point"+i, _x))<5;
		test=test && Math.abs(_root.menu.regle._y-getProperty("_root.point"+i, _y))<5
			if (test) {
				_root.menu.regle._x = getProperty("_root.point"+i, _x);
				_root.menu.regle._y = getProperty("_root.point"+i, _y);
			}
		}
	}
}
```

L'exercice est prêt, il ne nous reste plus qu'à envisager le test de correction de la réponse de l'élève.



II.Validation et correction

La mise en place du modèle des exercices de Mathenpoche a été pensé de façon à limiter les modifications d'un exercice à un autre, permettant une meilleure intégration à l'interface réseau (par l'intermédiaire des mêmes noms de variables par exemple).
Aussi, pour valider la réponse de l'élève, il nous suffira de changer le « test » de la fonction « Valider() ». Ici, nous nous contentons de tester si sa réponse est égale à la longueur du segment :

```
test = (zone_saisie.text == longueur);
```

Pour ce qui est de la fonction « correction() », elle consiste à rayer la réponse fausse et afficher la réponse attendue :

```
function correction() {
	//on barre la réponse de l'élève en rouge:
	barre.lineStyle(2, 0x7F0A0D, 60);
	barre.moveTo(zone_saisie._x, zone_saisie._y+zone_saisie._height);
	barre.lineTo(zone_saisie._x+zone_saisie._width, zone_saisie._y);
	consigne2._x = zone_saisie._x+zone_saisie._width+3;
	//on affiche la correction en dessous de la réponse fausse de l'élève :
	_root.createTextField("corrige", 31, zone_saisie._x, zone_saisie._y+zone_saisie._height+3, 30, 30);
	with (corrige) {
		setNewTextFormat(format_corrige);
		background = false;
		border = false;
		text = longueur+" cm";
		textColor = "0x336600";
		selectable = false;
		autoSize = true;
		align = "center";
	}
	if (numquestions<nbre_questions) {
		message_reponse = "Regarde bien la correction ...\ret clique sur \"Suite\" !";
	} else {
		message_reponse = "Faux ! Regarde bien la correction !";
	}
	commentaire.setTextFormat(format_correction);
}
```

III.Remarques complémentaires

Plusieurs remarques au sujet de cet exemple dans le soucis de rigueur et afin d'éviter quelques pièges fréquents dans l'utilisation de Flash :

- Attention à la profondeurs (« levels ») des clips créés par ActionScript. Ils représentent la couche sur laquelle le clip est créé. Une couche ne peut correspondre qu'à un unique clip ; c'est pourquoi nous utilisons un entier différent par clip.
  Dans l'exemple décrit, nous avons réservé :
  level ≥ 1 : consignes et zone de saisie ;
  level ≥ 10 : figure ;
  level ≥ 30 : correction ;
  level 99 : instruments de géométrie (au dessus du reste).
- La règle Mathenpoche permet de mesurer jusqu'à 15 cm : il faudrait donc tester si la longueur du segment obtenu respecte bien cette condition (par l'intermédiaire d'une boucle « while » par exemple).
- Dans le même ordre d'idée, il pourrait être intéressant de vérifier si les points ne sont pas trop proches l'un de l'autre et éviter des figures peu lisibles.
- Enfin, penser à effacer certains clips qui généraient l'affichage de l'appréciation finale.

IV.Exercices

- **Exercice 0 :**
  Réaliser l'exercice décrit dans l'exemple.
- **Exercice 1 :**
  Même exercice mais avec des figures de plus en plus complexes au fil des questions.
  - Variante : proposer de calculer le périmètre d'un polygone.
- **Exercice 2 :**
  Réaliser un exercice demandant à l'élève de tracer un segment d'extrémités données.
  - Variante : tracer un segment de longueur donnée (charge à l'élève de placer ses extrémités où il le souhaite)
- **Exercice 3 :**
  Réaliser un exercice demandant de tracer la perpendiculaire à une droite donnée passant par un point donné.
- **Exercice 4 :**
  Réaliser un exercice demandant à l'élève de tracer un cercle dont on connaît le centre et le rayon.
- **Exercice 5 :**
  Réaliser un exercice demandant à l'élève de tracer un triangle connaissant la longueur de ses côtés.