Création d'exercices
pour le logiciel Mathenpoche

http://www.mathenpoche.net

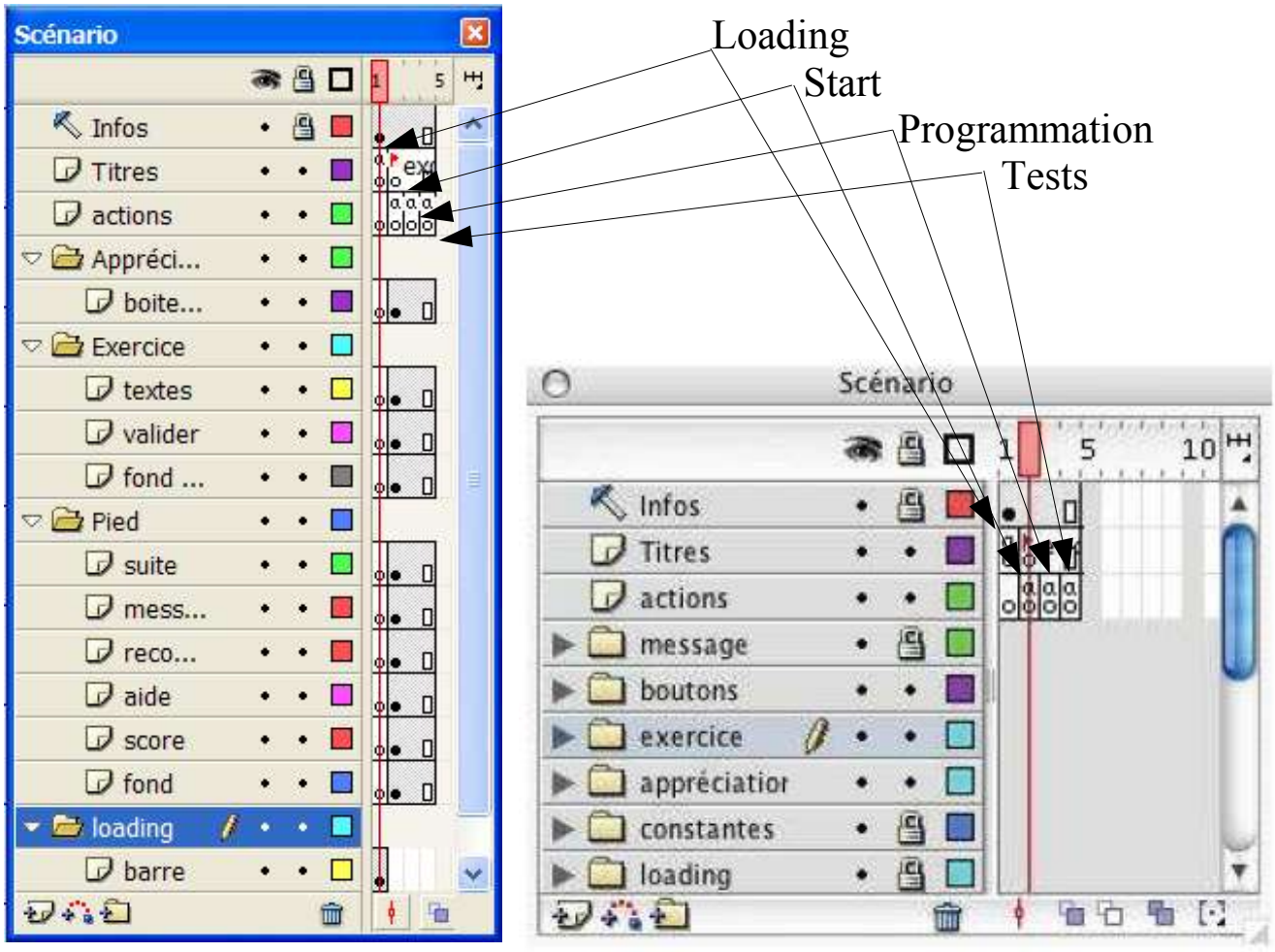
Mathenpoche est un logiciel libre. Ses sources sont en téléchargement sur le site du projet et sont modifiables librement et gratuitement conformément aux termes de la licence GPL.
Il s'agit de fichiers ".fla" zippés, éditables à l'aide du logiciel Flash MX 2004 de Macromédia dont une version d'évaluation de 30 jours est disponible sur le site de l'éditeur.

Tous les exercices de Mathenpoche sont divisés en quatre images, ayant chacune un rôle bien défini :

- l'image 1 concerne le chargement de l'exercice (« loading ») ;
- l'image 2 concerne l'initialisation des variables communes et le pré-chargement de l'aide ;
- l'image 3 concerne la programmation de l'exercice (variables, consignes, figures, ...)
- l'image 4 concerne la validation et la correction de la réponse de l'élève.

Nous allons ici détailler les actions de chacune de ces images, en particuliers les éléments communs à tous les exercices.

Remarque :
pour naviguer dans les actions des différentes images : depuis le scénario, sélectionnez le calque « Actions » puis l'image correspondante.
Affichez les actions en enfonçant la touche « F9 » (ou sélectionnez « Actions » dans le menu « fenêtres »).



I . L'image 1 : le « loading »

Cette image contient les éléments relatifs au chargement de l'exercice. Ils permettent de suspendre la lecture de l'animation jusqu'à ce qu'elle soit complètement téléchargée par le navigateur.

Cette image est vérouillée et ne nécessite aucune modification.

II. L'image 2 : l'initialisation

Cette image, nommée « `start` », contient toutes les variables et fonctions communes aux exercices Mathenpoche.

1 . Les variables :

Vous trouverez ci-dessous la liste des variables utilisées : elles sont communes à tous les exercices de Mathenpoche afin d'interagir correctement avec l'interface réseau. Seuls le code de l'exercice, le non du fichier d'aide et le nombre de questions peuvent être amenés à être modifiés dans cette image.

- `exo_nom` :
  le code de l'exercice utilisé dans l'interface réseau (par exemple "`6G1s1ex1`")
- `juste` :
  booléen indiquant à l'interface réseau si la réponse est juste ou non
- `pause` :
  entier indiquant un délai de réaction (1 seconde par défaut)
- `nbre_questions` :
  entier désignant le nombre de questions de l'exercice (5 ou 10) ;
- `erreur1` :
  entier désignant le nombre de premières erreurs (initialisé à 0) ;
- `neleve` :
  chaîne de caractères désignant le login de l'élève (initialisé par l'interface réseau)
- `score` :
  entier désignant le score actuel (initialisé à 0) ;
- `numquestions` :
  entier désignant le numéro de la question en cours (initialisé à 0) ;
- `nb_chances` :
  entier désignant le nombre de chances pour répondre (désactivée ou initialisée à 1) ;
- **`a`**`ffiche_score` :
  texte d'affichage du score ;
- `texte` :
  texte au format html affichant le score ("`<u><b>Mon score :</b></u>`").

2 . Les formats :

Mathenpoche utilise différents formats de texte pour communiquer avec l'élève. Ils sont définis en début d'exercice afin de faciliter leurs appels :

- Formats utilisés par les commentaires renvoyés à l'élève (après une validation par exemple) :
  - `format_faux` : utilisé pour mentionner une erreur ou une précision à l'élève ;
  - `format_bravo` : utilisé en cas de première réponse juste ;
  - `format_bien` : utilisé en cas de deuxième réponse juste ;
  - `format_correction` : utiliser pour inviter l'élève à consulter la correction.
- Formats utilisés par les zones de texte :
  - `format_saisie` : utilisé pour toutes les zones de saisie (là où écrit l'élève) ;
  - `format_consigne` : utilisé pour écrire l'énoncé de l'exercice ;
  - `format_corrige` : utilisé pour écrire la correction.

3 . Les fonctions :

Ces fonctions sont communes à tous les exercies de Mathenpoche.

- `charge_aide()` : charge l'aide animée en arrière plan ;
- `chargement` **`des fonctions réseaux`** (situées dans le clip externe « fonctions_reseau.swf ») ;
- `avance()` : détermine le passage à la question suivante ou l'affichage de l'appréciation finale ;
- `message(tTitre, tTexte)` : détermine les paramètres du message d'appréciation final ;
- `appreciation(res)` : affiche l'appréciation finale en fonction du score de l'élève et du nombre de questions de l'exercice.

III. L'image 3 : la programmation de l'exercice

Cette image nommée « `programmation` » concerne le contenu de l'exercice. La validation de la réponse se fait dans l'image suivante et renvoie sur celle-ci pour l'affichage de la nouvelle question. On y trouvera l'initialisation des objets ou variables, le tracé des figures relatives à la question en cours, l'affichage des textes de la question ... et, avant tout, les paramètres de cette question.

1 . Initialisation des variables et objets :

- Initialisation des variables :
  `suite = 0` (la variable suite permet de changer de question lorsqu'elle vaut 1)
  `occupe = false` (permet de vérifier si le logiciel est en train de travailler)
  `numquestions = numquestions+1` (incrémentation du numéro de la question en cours)
- Initialisation des textes :
  `message_reponse = ""` (iniitalisation de l'indication renvoyée à l'élève après la validation)
  `affiche_numquestions = "<u>Question N°"+numquestions+" :</u>"`
  • Affichage (ou masquage) des boutons :
  `btn_recommence._visible = false`
  `btn_valider._visible = true`
  `btn_aide._visible = false`
  `btn_suite._visible = false`
- Création/effacement du clip qui barre la réponse fausse de l'élève :
  `createEmptyMovieClip("barre", 10)`
- Création/effacement du clip qui accueille la correction :
  `c``reateEmptyMovieClip("corrige", 11)`

2 . La programmation de l'exercice (exemple) :

Une fois les variables, objets et textes initialisés, il ne nous reste « plus qu'à » programmer l'exercice à proprement dit. Par exemple, le code suivant demande à l'élève de recopier un chiffre choisi aléatoirement :

```
chiffre_choisi = random(10);
createTextField("consigne", 1, affichage_question._x ,
                affichage_question._y+affichage_question._height+3, 0, 30);
with (consigne) {
        setNewTextFormat(format_consigne);
        background = false;
        border = false;
        align = "left";
        text = "Recopie le chiffre "+chiffre_choisi+" dans la zone de saisie :";
        textColor = "0x666666";
        type = "static";
        autoSize = true;
        selectable = false;
}
createTextField("zone_saisie", 2, consigne._x+consigne._width+3, consigne._y, 0, 30);
with (zone_saisie) {
        setNewTextFormat(format_saisie);
        background = true;
        border = true;
        maxChars = 1;
        restrict = "0-9";
        align = "center";
        text = "";
        textColor = "0x7484A6";
        type = "input";
        autoSize = true;
        tabIndex = 1;
}
zone_saisie.onChanged = function() {
        zone_saisie.textColor = "0x7484A6";
};
btn_valider.tabIndex=2
btn_suite.tabIndex=3
btn_valider._focusrect=false
btn_suite._focusrect=false
```

Le résultat obtenu :

La question est programmée, nous pouvons maintenant attendre la réponse de l'élève !

IV. L'image 4 : la validation de la réponse

Cette dernière image contient tout ce qui est relatif à la correction de la réponse de l'élève : la validation, la correction, l'affichage de l'aide et les différents commentaires qui lui seront renvoyés.

1 . La validation :

Tout est regroupé dans la fonction « `Valider()` », dans laquelle il faudra envisager plusieurs cas de figures :

Une réponse incomplète :

on ne comptabilise pas cette non réponse comme fausse mais on l'indique à l'élève.

Une première réponse juste :

`(test = true et erreur =0)`

on affiche le message « Bravo ! » en utilisant un des formats de texte définis, le score et on passe à la question suivante (en revenant à l'image « programmation »).

Une première erreur :

`(test = false et erreur =0)`

on indique que la réponse est fausse (en rouge) et invite l'élève à consulter l'aide par le message et l'affichage de son bouton d'ouverture.

Une seconde réponse juste :

`(test = true et erreur=1)`

on valide la réponse, affiche le score et passe à la question suivante.

Une seconde réponse fausse :

`(test = false et erreur =1)`

on ouvre l'aide. Pendant ce temps, la réponse de l'élève est barrée, la correction s'affiche à côté ainsi que le bouton « suite » pour passer à la question suivante.

Concernant notre exemple, voici le code obtenu :

```
function Valider() {
        message_reponse = "";
        if (zone_saisie.text == "") {
                message_reponse = "Il faut répondre à la question !";
                commentaire.setTextFormat(format_faux);
                Selection.setFocus("zone_saisie");
        } else if (numquestions<=nbre_questions) {
                _root.occupe = true;
                zone_saisie.type = "dynamic";
                test = (zone_saisie.text == chiffre_choisi);
                if (test == true) {
                        juste = true;
                        if (erreur == 0) {
                                message_reponse = "Bravo !";
                                commentaire.setTextFormat(format_bravo);
```

```
                        } else if (erreur == 1) {
                                message_reponse = "Bien ! Tu as corrigé ton erreur !";
                                commentaire.setTextFormat(format_bien);
                        }
                        btn_valider._visible = false;
                        score = score+1;
                        affiche_score = score+" sur "+numquestions;
                        _root.suite = 1;
                } else {
                        juste = false;
                        if (erreur == 0) {
                                erreur = erreur+1;
                                erreur1 = erreur1+1;
                                message_reponse = "Faux ! Encore un essai !\rUtilise l'aide !";
                                commentaire.setTextFormat(format_faux);
                                btn_aide._visible = true;
                                zone_saisie.type = "input";
                                zone_saisie.textColor = "0x7F0A0D";
                                Selection.setFocus("zone_saisie");
                                Selection.setSelection(_root.zone_saisie.text.length,
                                                        _root.zone_saisie.text.length);
                                _root.occupe = false;
                        } else {
                                erreur = erreur + 1;
                                if (numquestions<nbre_questions) {
                                        message_reponse = "Non ! Regarde bien l'aide !";
                                        commentaire.setTextFormat(format_faux);
                                        _root.chrono2.gotoAndPlay(2);
                                } else {
                                        pause = 2;
                                        correction();
                                        suite = 1;
                                }
                                btn_valider._visible = false;
                                affiche_score = score+" sur "+nbre_questions;
                        }
                }
                commphp();
        }
}
```

D'un exercice à un autre, la différence se fera au niveau du test et de l'affichage des erreurs de l'élève, en ajoutant éventuellement des messages spécifiques aux erreurs « classiques » liées à l'exercice.

Attention !
L'appel de la fonction « commphp() » permet de communiquer les résultats de l'élève à l'interface réseau. Présente dans le clip externe « fonctions_reseau.swf » chargé dans l'image 1, elle permet d'enregistrer les scores dans la base de données afin qu'ils apparaissent dans le bilan de l'élève et la visualisation de la séance par le professeur ... à n'oublier sous aucun prétexte !

2 . La correction :

La fonction « `correction()` », comme son nom l'indique, est chargée d'afficher la correction suite à une deuxième erreur de l'élève. Elle se chargera donc de rayer la réponse fausse et d'afficher la réponse attendue.

Concernant notre exemple, voici le code obtenu :

```
function correction() {
        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);
        _root.createTextField("corrige", 11, zone_saisie._x+zone_saisie._width+3, zone_saisie._y,
                                30, 30);
        with (corrige) {
                setNewTextFormat(format_corrige);
                background = false;
                border = false;
                text = chiffre_choisi;
                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);
}
```

V . Exercices

- **Exercice 0 :**

  Réaliser l'exercice décrit dans l'exemple.

- **Exercice 1 :**

  Réaliser un exercice demandant la somme de deux nombres choisis aléatoirement.
  - Variante : proposer une différence, un produit, un quotient ou un autre calcul plus compliqué !

- **Exercice 2 :**

  Réaliser un exercice, composé de cinq questions, interrogeant chacune sur une opération différente (avec un choix aléatoire de l'opération de la dernière question).
  - Variante 1 : proposer une progression dans la difficulté des nombres choisis.
  - Variante 2 : proposer le même exercice avec un ordre aléatoire des opérations.

- **Exercice 3 :**

  Réaliser un exercice demandant la simplification d'une fraction donnée.
  - Variante 1 : adapter l'exercice avec une progression dans la difficulté.
  - Variante 2 : addition de deux fractions de même dénominateur.

V . Exercices

- **Exercice 0 :**

  Réaliser l'exercice décrit dans l'exemple.

- **Exercice 1 :**

  Réaliser un exercice demandant la somme de deux nombres choisis aléatoirement.
  - Variante : proposer une différence, un produit, un quotient ou un autre calcul plus compliqué !

- **Exercice 2 :**

  Réaliser un exercice, composé de cinq questions, interrogeant chacune sur une opération différente (avec un choix aléatoire de l'opération de la dernière question).
  - Variante 1 : proposer une progression dans la difficulté des nombres choisis.
  - Variante 2 : proposer le même exercice avec un ordre aléatoire des opérations.

- **Exercice 3 :**

  Réaliser un exercice demandant la simplification d'une fraction donnée.
  - Variante 1 : adapter l'exercice avec une progression dans la difficulté.
  - Variante 2 : addition de deux fractions de même dénominateur.

V . Exercices

- **Exercice 0 :**

  Réaliser l'exercice décrit dans l'exemple.

- **Exercice 1 :**

  Réaliser un exercice demandant la somme de deux nombres choisis aléatoirement.
  - Variante : proposer une différence, un produit, un quotient ou un autre calcul plus compliqué !

- **Exercice 2 :**

  Réaliser un exercice, composé de cinq questions, interrogeant chacune sur une opération différente (avec un choix aléatoire de l'opération de la dernière question).
  - Variante 1 : proposer une progression dans la difficulté des nombres choisis.
  - Variante 2 : proposer le même exercice avec un ordre aléatoire des opérations.

- **Exercice 3 :**

  Réaliser un exercice demandant la simplification d'une fraction donnée.
  - Variante 1 : adapter l'exercice avec une progression dans la difficulté.
  - Variante 2 : addition de deux fractions de même dénominateur.