Bref guide d'utilisation du logiciel SPAF (Surfaces Par Agencement de Faces) Version JavaScript / p5.js

SPAF est un petit programme conçu pour créer et pour manipuler des figures 3D constituées de faces polygonales. L'entrée de ces figures se fait sous forme d'une suite d'instructions, où l'on donne une liste de sommets (via leurs coordonnées) et de faces (via une liste ordonnée des sommets qui les définissent). Par la suite, cependant, on accède à une plus grande liberté de visualisation et de manipulation de ces figures.

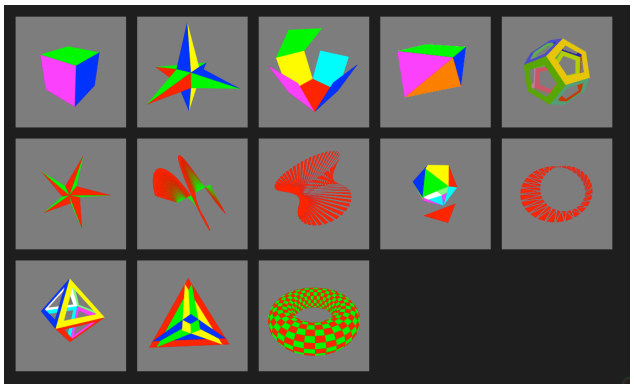
Par où commencer?

L'utilisation de SPAF se fait en deux temps :

1. On décrit tout d'abord une figure dans un éditeur de texte
2. On affiche et on manipule cette figure dans un navigateur web (comme ***Chrome***, ***Firefox***, ou ***Safari***),

Ce guide suit l'ordre indiqué ci-dessus, mais il faut avouer que l'étape 1 est plutôt aride, tandis que l'étape 2 est plus facile et plus gratifiante. Si vous le désirez, vous pouvez directement sauter à l'étape 2, en utilisant les exemples pré-définis, fournis avec SPAF. Quand vous aurez fini, vous pourrez revenir à l'étape 1, en continuant de lire le présent guide.

Pour passer directement à l'étape 2, vous vous rendez à la page web de SPAF
http://www.gruteam.uqam.ca/SPAF/p5/index.html
et vous cliquez sur l'image d'un des exemples proposés.



(Note : pour l'instant, je ne vous recommande pas de choisir la figure intégrée.)

Une nouvelle fenêtre, correspondant à l'exemple choisi, va apparaître. Pour manipuler l'image, il suffit, en général, de choisir une commande dans le menu local des *Actions*, puis de cliquer sur le bouton de la souris, ou de faire glisser celle-ci au-dessus de la figure, en maintenant le bouton enfoncé. Parfois, vous devrez aussi choisir la *Face Active* dans le menu local correspondant. Les glissières dites de « *vitesse* » serviront à changer la réactivité des commandes choisies dans le menu *Actions*. Si d'autres glissières sont disponibles, leur effet dépendra des intentions du concepteur de la figure.

Les quelques indications du paragraphe précédent devraient suffire pour commencer à explorer SPAF. Mais, si vous rencontrez des difficultés ou si vous voulez des précisions sur le fonctionnement de certaines commandes, vous pouvez toujours consulter la section « **L'exploration des figures dans SPAF** » du présent guide.

Installation

Précisons tout d'abord que tous les logiciels requis sont gratuits.

Vous devez tout d'abord télécharger le logiciel ***SPAF*** , en version p5.js, à l'adresse
http://www.gruteam.uqam.ca/SPAF/p5/index.html
Une fois téléchargé, vous décompressez le fichier SPAF.zip pour obtenir un dossier SPAF contenant lui-même deux dossiers : *SPAF_initial* et *Exemples*.

Le dossier *SPAF_initial* correspond à une figure vide, et comprend les fichiers *index.html* et *Figure.js*, ainsi que le dossier *librairies*.

Pour définir une nouvelle figure, nous vous suggérons de dupliquer et de renommer le dossier *SPAF_initial*. Comme nous le verrons par la suite, il faudra alors modifier le fichier *Figure.js*, et ce fichier seulement.

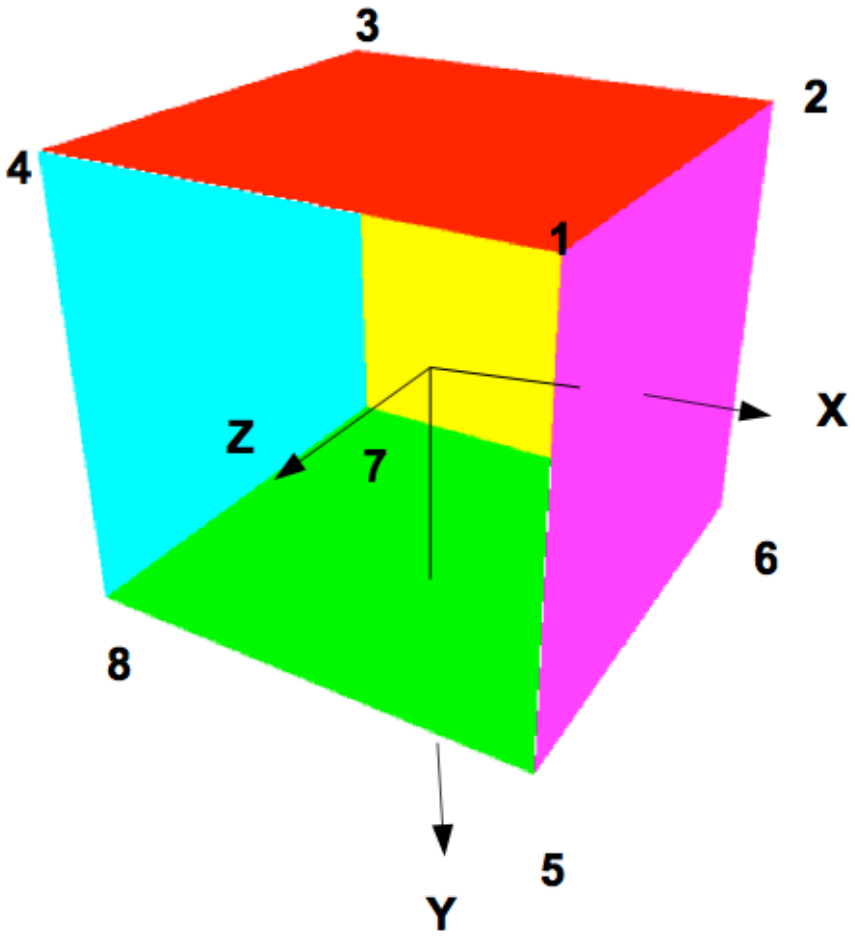
Pour modifier le fichier *Figure.js*, nous vous suggérons d'utiliser l'éditeur ***Brackets***, disponible à l'adresse brackets.io. Cet éditeur fonctionne conjointement avec le navigateur ***Chrome***, qu'on peut aussi télécharger à l'adresse
https://www.google.com/chrome/browser/desktop/index.html

Pour utiliser une figure déjà définie, il ne sera pas nécessaire d'avoir recours à ***Brackets*** ni à ***Chrome***. Dans votre navigateur web favori, il suffira d'afficher la page web décrite dans le fichier *html* du dossier de votre figure.

Veuillez noter que vous devrez activer ***JavaScript*** dans tous les navigateurs web où vous voudrez utiliser les figures créées avec ***SPAF***.

Notre première figure

Comme première figure, nous allons tracer un cube. Comme nous le savons tous, un cube a 8 sommets (numérotés de 1 à 8 sur le document 1 ci-dessous) et 6 faces (dont 5 sont colorées dans ladite figure). Nous avons aussi indiqué le système de coordonnées utilisé par SPAF, avec l'axe des x pointant vers la droite, l'axe des y dirigé vers le bas, et l'axe des z sortant de l'écran. Nous centrerons le cube à l'origine de notre système d'axes, et nous choisirons l'échelle des axes de telle sorte que les coordonnées du sommet 5 soient (1,1,1).



Document 1. La figure représentée dans le système de coordonnées de SPAF

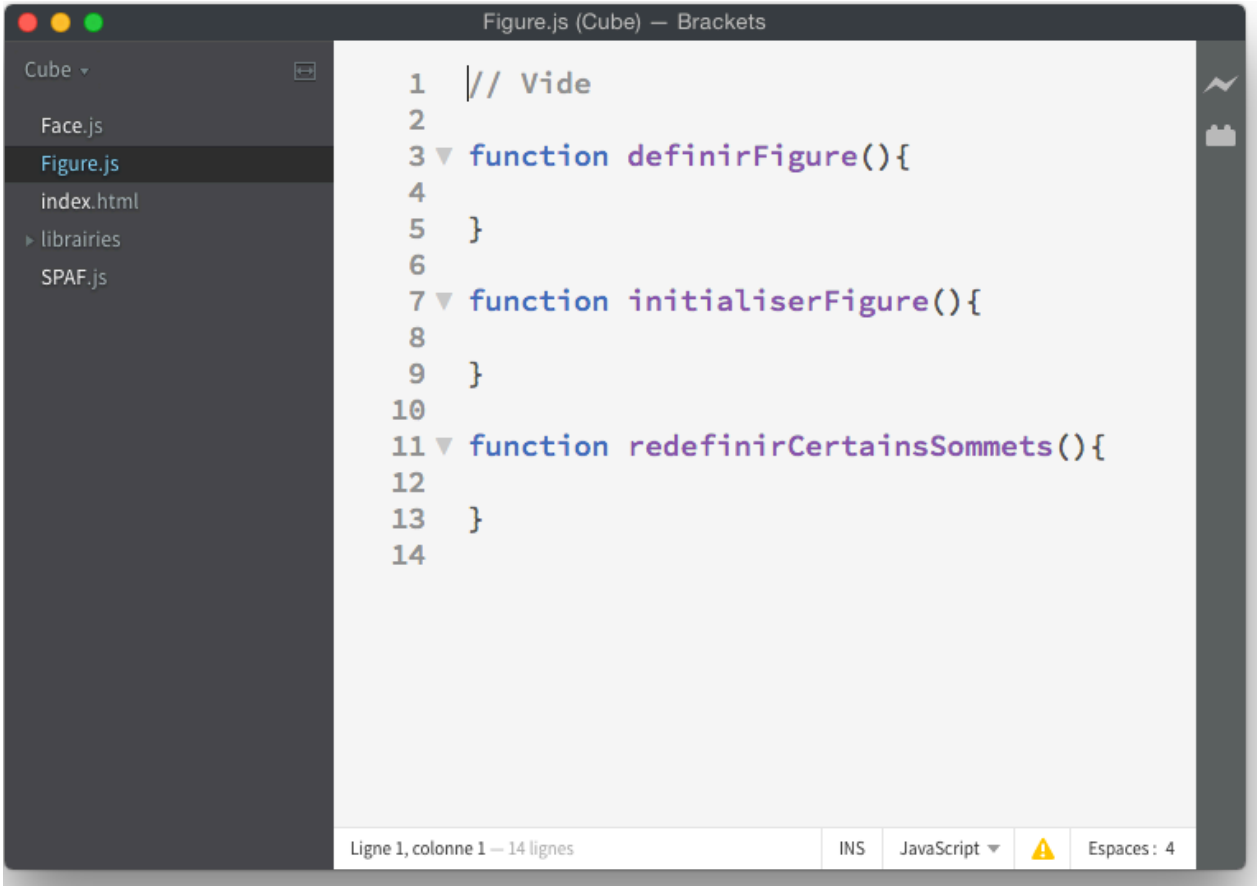
Voyons maintenant comment communiquer ces informations à SPAF. Nous commençons par dupliquer et renommer le dossier *SPAF_initial*. Nous supposerons que le nom donné à ce nouveau dossier est *Cube*. Notons que nous pouvons aussi renommer le fichier *index.html* à notre guise. Mais, si nous voulons renommer le fichier *Figure.js*, il faudra modifer *index.html* en conséquence, en changeant la ligne

<script language="javascript" type="text/javascript" src="Figure.js"></script>

pour

<script language="javascript" type="text/javascript" src="*NouveauNom*.js"></script>

Nous ouvrons ensuite ce dossier *Cube* dans ***Brackets*** via la commande « Ouvrir un dossier… » du menu « Fichier ». Après un clic sur « Figure.js », dans la colonne de gauche, nous obtenons la fenêtre illustrée ci-dessous.



Document 2. SPAF tel qu'il apparaît à son ouverture dans l'éditeur ***Brackets***, après avoir cliqué sur « Figure.js » (dans la colonne de gauche).

La définition de notre figure se fera au sein de la fonction « definirFigure », au moyen de déclarations

definirSommet(numéroSommet, x, y, z);

et

definirFace(numéroFace, NoSommet1, NoSommet2, NoSommet3, NoSommet4);

On pourra colorer chaque face au moyen de l'invocation

couleurFace(numéroFace, rouge, vert, bleu)

où chacune des composantes colorées (rouge, vert et bleu) peut prendre des valeurs entre 0 et 255.

Notez que, malheureusement, notre environnement est très sensible aux erreurs syntaxiques : l'oubli d'un seul caractère, que ce soit une parenthèse ou un point-virgule, pourrait avoir des conséquences étranges et catastrophiques.

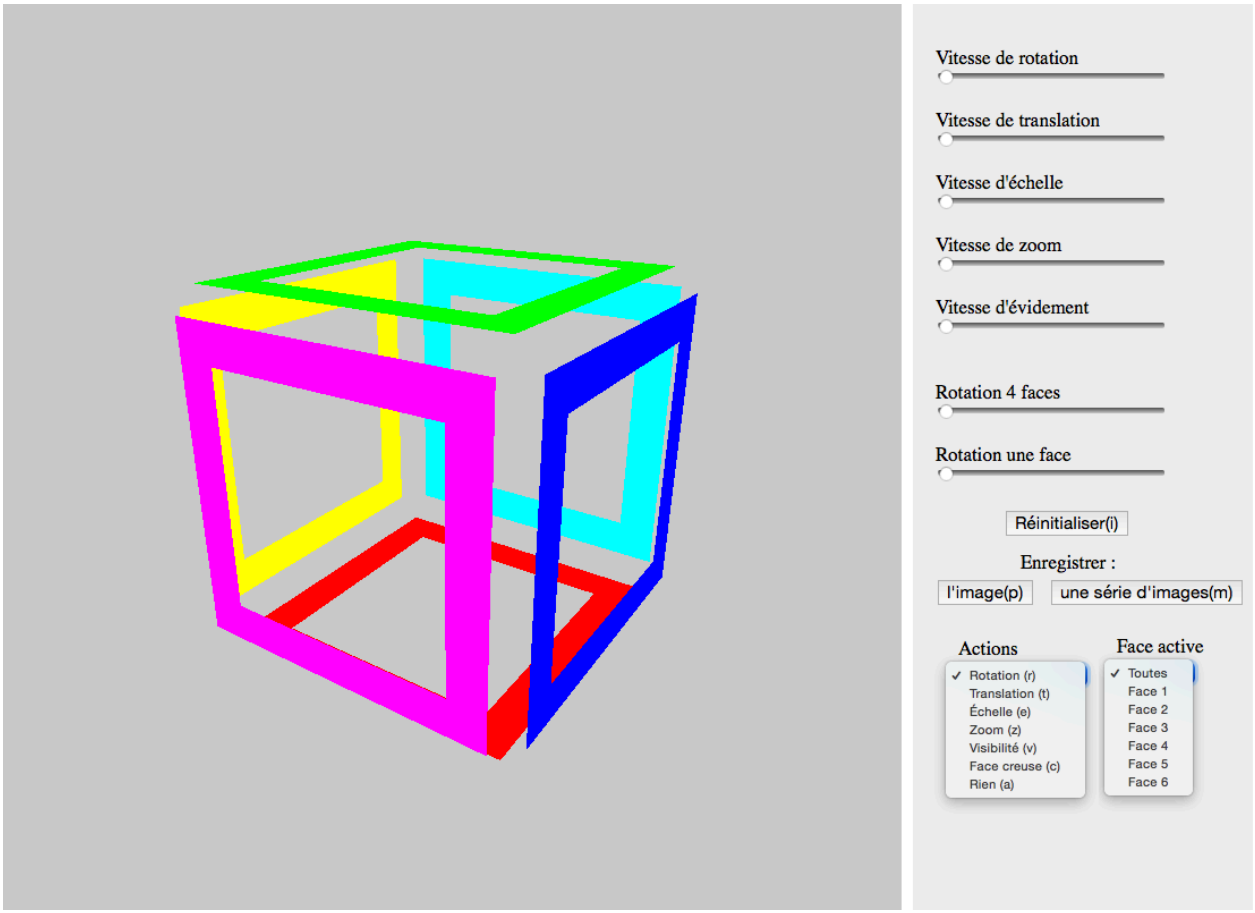
Le document 3 montre la définition finale de notre cube, et nous devons expliciter certaines choses. Tout d'abord la commande « *pixelsParUnite*(300) », qui sert à établir qu'une de nos unités (rappelons que chaque arète de notre cube mesure 2 unités) correspondra à 300 pixels à l'écran. Et enfin on remarque que la notation « // » indique que tout ce qui suit sur la ligne sera considéré comme un simple commentaire, dans ce cas pour nommer la couleur définie pas ses trois composantes.

```
function definirFigure(){
  pixelsParUnite(300);

  definirSommet(1, 1, -1, 1);
  definirSommet(2, 1, -1, -1);
  definirSommet(3, -1, -1, -1);
  definirSommet(4, -1, -1, 1);
  definirSommet(5, 1, 1, 1);
  definirSommet(6, 1, 1, -1);
  definirSommet(7, -1, 1, -1);
  definirSommet(8, -1, 1, 1);

  definirFace(1, 1, 2, 3, 4);
  definirFace(2, 5, 6, 7, 8);
  definirFace(3, 1, 4, 8, 5);
  definirFace(4, 3, 4, 8, 7);
  definirFace(5, 1, 5, 6, 2);
  definirFace(6, 2, 3, 7, 6);


  couleurFace(1, 255, 0, 0);       // rouge
  couleurFace (2, 0, 255, 0);      // vert
  couleurFace (3, 0, 0, 255);      // bleu
  couleurFace (4, 0, 255, 255);    // cyan
  couleurFace (5, 255, 0, 255);    // magenta
  couleurFace (6, 255, 255, 0);    // jaune
}
```

Document 3. Description de notre cube dans SPAF

Pour faire apparaître la figure définie au document 3, il suffit d'enregistrer le tout (commande « Enregistrer » du menu « Fichier »), puis de cliquer sur l'icône  , située en haut et à droite de la fenêtre. Le navigateur ***Chrome*** s'ouvre alors, en affichant une nouvelle fenêtre (voir document 4) : la figure est représentée à gauche, tandis que les commandes mises à notre disposition sont affichées à droite.

Au départ, notre figure ne paie pas de mine : elle ressemble à un simple carré bleu centré dans sa fenêtre. Nous savons qu'un cube se cache derrière cette face bleue, et nous allons le révéler à l'aide de la commande « Rotation (r) » du menu des actions (à gauche), appliqué à « Face active : Toutes » du menu de droite. Les déplacements de notre souris avec son bouton enfoncé dans la fenêtre de la figure produisent alors les rotations désirées. Soulignons que nous pouvons avoir recours à la glissière « Vitesse de rotation » pour modifier la réactivité de nos rotations.



Document 4. Une représentation de notre cube, obtenue via les commandes à droite

Notons en passant les caractères entre parenthèses dans les menus, qui agissent comme des équivalents clavier des items correspondants des menus. Soulignons aussi que certains items du menu des actions peuvent s'appliquer à chacune des composantes (y compris à toutes celles-ci), tandis que d'autres ne s'appliquent qu'à la totalité de celles-ci : à vous de découvrir lesquelles…

Pour obtenir le document 1, il faut faire disparaître la face bleue, qui a été la troisième face à être définie. Il suffit alors de choisir « Visibilité (v) » appliquée à « Face3 », puis à faire un clic dans la fenêtre de la figure. Un autre clic la fera réapparaître.

En choisissant « Echelle (e) » appliqué à toutes les composantes, on peut faire varier simultanément la taille de toutes les faces, toujours en déplaçant notre souris avec son bouton enfoncé dans la fenêtre de la figure. On pourra de même utiliser la commande « Face creuse (c) » pour obtenir la figure représentée à droite du document 4.

Avant de poursuivre la lecture de ce guide, nous vous suggérons de continuer à expérimenter les diverses commandes disponibles dans SPAF, que ce soit à partir du *Cube* qu'avec les autres exemples fournis avec SPAF. Amusez-vous bien !

La description des figures dans SPAF

La description des figures SPAF se fait dans le fichier « Figure.js », qui comporte au départ trois fonctions: **definirFigure**, **initialiserFigure** et **redefinirCertainsSommets**. Pour décrire lesdites figures, on pourra utiliser non seulement les primitives SPAF, mais aussi toute la puissance de **p5.js** (qui est lui-même une extension du langage de programmation **JavaScript**). Mais ceci ne devrait pas effrayer le débutant : on peut définir des figures très intéressantes sans grande connaissance de la programmation. Cependant, il faut minimalement se rappeler que nos instructions devront être séparées par des point-virgules. Par contre, des connaissances mathématiques seront, en général, nécessaires…

La description des figures SPAF se fait donc en **trois étapes** :

1. Dans un premier temps, on décrit les sommets, les faces (et leur couleur) et les barres de défilement de notre figure dans la fonction **definirFigure**. Notons que les faces ainsi décrites devront être planaires et convexes.
2. Par la suite, on décrit l'état initial de notre figure dans la fonction **initialiserFigure**. On peut voir cette étape comme une utilisation par programmation du menu des actions pour décrire l'état initial voulu de notre figure. Notons que cette fonction peut être vide (c.-à.-d. sans aucune commande) quand on ne désire pas modifier l'état initial de notre figure.
3. Enfin, dans la fonction **redefinirCertainsSommets**, on décrira l'effet de nos barres de défilement sur nos sommets. Cette fonction restera vide si nous n'avons défini aucune barre de défilement.

Primitives de SPAF pouvant être utilisées à l'étape 1, dans la fonction « definirFigure ».

pixelsParUnité(x)

Fixe le nombre de pixels pour représenter une unité (où x est un nombre réel).

definirSommet(numéro du sommet, x, y, z)

x, y, z sont les coordonnées du sommet.
Si on veut n sommets, ils doivent absolument être numérotés de 1 à n, même s'ils ne sont pas définis dans l'ordre

definirFace(numéro de la face, numéro du sommet 1, … , numéro du sommet k)

Les numéros des sommets définissant la face doivent être énumérés dans l'ordre d'un « parcours de la frontière » de la face. Rappelons que nos faces doivent être planes et convexes.

Notons que le nombre de sommets doit être d'au moins 3. Nous verrons plus loin comment définir une face à l'aide d'une liste de sommets.

couleurFace(numéro de la face, r, v, b)

Les composantes (rouge, vert, bleu) de la couleur sont des réels entre 0 et 255.

creerGlissiereG1(a, b, c, d, nom de la glissière) et **creerGlissiereG1(a, b, c, d)**

Crée une glissière dans le tableau de contrôle. **a** est la borne gauche, **b** la borne droite, **c** la valeur initiale à la création, et **d** le pas séparant deux valeurs consécutives.

La valeur courante est en permanence disponible dans la variable **valG1**.

Si le nom de la glissière est spécifié, il sera utilisé pour l'identifier dans le tableau de contrôle. Dans le cas contraire, le nom « G1 » sera utilisé.

On peut créer deux autres glissières **creerGlissiereG2** et **creerGlissiereG3** selon le même principe, et leurs valeurs seront **valG2** et **valG3**

Primitives de SPAF pouvant être utilisées aux étapes 2 et 3, dans les fonctions « initialiserFigure » et « redefinirCertainsSommets ».

Si on le désire, on peut redéfinir la position de certains sommets, ce qui aura pour effet de modifier les faces correspondantes, et donc la figure associée. Pour ce faire, on peut faire appel aux fonctions *xSommet*, *ySommet* et *zSommet* donnant les coordonnées des divers sommets.

redefinirSommet(i, x, y, z)

Change les coordonées du sommet numéro **i** en x, y, z. À l'étape 3, les valeurs utilisées pour **x**, **y** et/ou **z** seront des expressions faisant appel aux variables **valG1**, **valG2** et/ou **valG3** des glissières. À l'étape 2, ces variables auront encore leurs valeurs initiales.

xSommet(i), ySommet(i), zSommet(i)

Fonctions retournant respectivement les coordonnées (redéfinies ou non) en x, y, z du sommet **i**.

modifierCouleurFondSPAF(r, v, b)

Change la couleur de fond de la figure SPAF. Les composantes (rouge, vert, bleu) de la couleur sont des réels entre 0 et 255

Poursuivons en décrivant les commandes pouvant s'appliquer tant aux faces individuelles qu'à toute la figure. Quand on désire qu'une de ces commandes s'applique à toutes les faces, il suffit de l'employer avec un numéro de face égal à 0 (zéro).

fixeEchelle(i, valeur)

Fixe l'échelle de la face numéro **i** à la valeur spécifiée, un réel. Si la face **i** n'existe pas, il ne se passe rien.

fixeVisible(i, valeur)

Fixe la visibilité de la face numéro **i** à la valeur spécifiée, un booléen (*true* ou *false*).

fixeTailleCreux(i, valeur)

Lorsqu'on évide une face, l'évidemment est homothétique à la face elle-même. La valeur (entre 0 et 1) est en fait le rapport d'homothétie utilisé pour la face numéro **i**.

integrerDansFigure(i, valeur)

Indique si on veut que la face numéro **i** fasse partie (valeur = *true*) ou non (valeur = *false*) de la figure. Cette commande sert dans les cas où notre figure comporte un nombre variable de faces (dépendant, par exemple, de l'état de certaines glissières). Dans la définition initiale, on est contraint de définir le nombre maximal de faces possibles, car on ne peut créer par la suite de nouvelles faces.

changerCouleur(i, rouge, vert, bleu)

Pour déterminer, après la création initiale, la couleur de la face numéro **i**, en spécifiant ses composantes rouge, verte et bleue. Notez ici que **i** ne peut prendre la valeur 0, et donc qu'on ne peut appliquer ce changement de couleur à toutes les faces.

Les commandes précédentes servaient à donner des valeurs à certains paramètres définissant les faces de notre figure. Mais on peut aussi vouloir connaître la valeur de ces paramètres sans vouloir modifier celle-ci : c'est précisément le rôle des fonctions ci-dessous.

donneEchelle(i)

Cette *fonction* retourne l'échelle de la face numéro **i**.

donneVisible(i)

Cette *fonction* retourne *true* si la face numéro **i** est visible, et *false* sinon.

donneTailleCreux(i)

Cette *fonction* retourne le rapport d'homothétie entre le creux et de la face numéro **i**.

donneDansFigure(i)

Cette *fonction* retourne *true* si la face numéro **i** fait partie de la figure, et *false* sinon.

donneRouge(i) ***donneVert(i)*** ***donneBleu(i)***

Ces *fonctions* retournent les composantes de couleur de la face numéro **i**.

Passons maintenant à la description de commandes représentant des transformations globales s'appliquant à toute la figure.

fixeTranslationX(valeur) et **fixeTranslationY(valeur)**
ajouteTranslationX(valeur) et **ajouteTranslationY(valeur)**

Translate toute la figure (selon l'axe des X ou des Y) d'une distance spécéfiée par la valeur. Dans le cas de **fixeTranslation**, la translation s'effectue depuis la position initiale de la figure. Dans le cas de **ajouteTranslation**, la translation s'effectue depuis la position précédente de la figure.

fixeRotationX(valeur) et **fixeRotationY(valeur)**
ajouteRotationX(valeur) et **ajouteRotationY(valeur)**

Fait tourner toute la figure autour de l'axe des X ou des Y d'un angle dont la valeur est donnée en degrés. Dans le cas de **fixeRotation**, la rotation s'effectue depuis la position initiale de la figure. Dans le cas de **ajouteRotation**, la rotation spécifiée par la valeur s'ajoute aux rotations précédentes cumulatives de la figure.

Les commandes précédentes servaient à donner des valeurs à certains paramètres globaux. On peut aussi vouloir connaître la valeur de ces paramètres sans vouloir les modifier : c'est précisément le rôle des fonctions ci-dessous.

***donneTranslationX*()** et ***donneTranslationY*()**

Retourne la composante (en X ou en Y) de la translation globale.

***donneRotationX*()** et ***donneRotationY*()**

Retourne l'angle de rotation autour de l'axe des X ou de l'axe des Y.

La commande suivante permet de prendre une « photo » de la figure, et de la sauvegarder dans le format spécifié dans le nom.

prendrePhoto(nomFichier, extension)

L'extension est soit "jpg", soit "png". Le fichier résultant est enregistré dans le dossier des téléchargements. On peut se servir de cette commande pour enregistrer plusieurs vues de notre figures qui pourront par la suite être assemblées pour former une animation *QuickTime* ou autre. Exemples de **nomFichier** : "photoFigure" ou même "figure " + no (où « no » est un nom de variable). Avertissement : ne fonctionne pas bien dans *Safari*.

L'environnement **Brackets** manque d'outils de mise au point. Nous pouvons cependant utiliser la console *JavaScript* du navigateur *Chrome*, accessible via la commande « Options pour les développeurs » du menu « Afficher ». Les commandes suivantes, bien qu'assez primitives, peuvent nous aider à mettre au point les instructions décrivant nos figures. Les utilisateurs plus avancés pourront utiliser les autres composantes de ces « Options pour les développeurs », qui leur permettront notamment de placer des points d'arrêt et de comander des exécutions pas à pas.

affiche(valeur1, …, valeurN) et **afficheLigne(valeur1, …, valeurN)**

Affiche les valeurs fournies (séparées par des virgules) dans la console *JavaScript* de *Chrome*. Les valeurs affichées peuvent être des nombres ou des chaînes de caractères. **afficheLigne** sépare les valeurs affichées par des sauts de ligne. On peut même utiliser

afficheS(séparateur, valeur1, …, valeurN)

pour spécifier, au moyen de la chaîne de caractère « séparateur », ce qui sera écrit entre les valeurs affichées.

Éléments du langage JavaScript utiles pour décrire des figures plus complexes.

conditionnelle
if (condition) {instructions si condition vérifiée} else {instructions si condition non vérifiée}

boucle avec compteur
for (initialisation; condition; mise à jour) {instructions}

boucle avec condition
while (condition) {instructions}

À titre d'exemple, voyons maintenant comment on peut utiliser un vecteur de JavaScript pour définir une face, disons avec 100 sommets dont les numéros varient entre 101 et 200. On définit tout d'abord notre vecteur comme suit

```
mon_vecteur = [101, 102, … , 199, 200];
```

puis on utilise une forme spéciale de la fonction « definirFace »

```
definirFace(numero, mon_vecteur);
```

On peut même généraliser en supposant que les numéros des sommets sont compris entre **a** et **b**, où **a** et **b** sont des variables entières définies précédemment. On définit alors notre vecteur comme suit

```
mon_vecteur = [];
for (var k = a; k<=b; k++) {mon_vecteur[k-a]=k;};
```

et on termine de la même façon par

```
definirFace(numero, mon_vecteur);
```

Vous pourrez trouver dans les exemples inclus avec SPAF d'autres utilisations du langage JavaScript pour définir des figures…

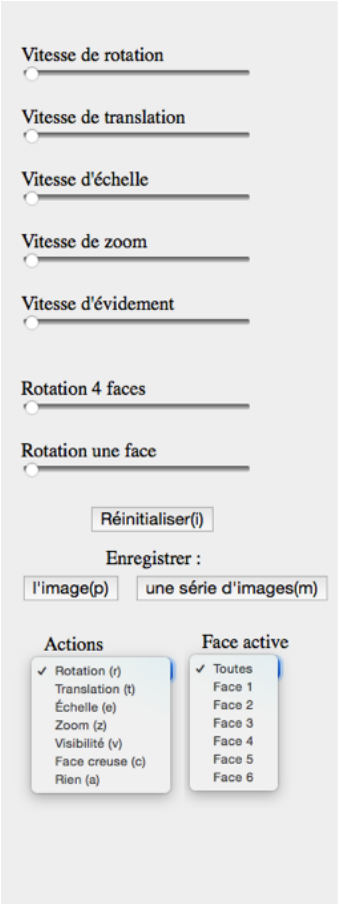
L'exploration des figures dans SPAF

Une fois la figure décrite, on peut en commander l'affichage, que ce soit par un clic sur l'icône  de la fenêtre d'édition, par le choix de l'item « Aperçu en direct » du menu « Fichier », ou par son équivalent clavier ⌥⌘P. C'est alors que le plaisir commence…

En plus de la zone graphique (où la figure est représentée) apparaît la zone des commandes (voir le document 5), qui va nous permettre d'interagir dynamiquement avec notre figure. Cette zone comporte quatre parties :

- Les glissières (en haut)
  Dans le cas illustré, cette zone ne comporte que des glissières affectant la vitesse d'exécution de certaines commandes. Quand une commande donnée est sélectionnée, la glissière correspondante peut être modifiée avec la souris ou avec le clavier : « - » pour en diminuer la valeur, et « + » ou « = » pour l'augmenter.
  De plus, comme on l'a vu précédemment, cette zone peut accueillir jusqu'à trois glissières supplémentaires définies par l'utilisateur.
- Les boutons pour « Réinitialiser » et « Enregistrer » (soit une image, ou une série d'images)
- Le menu déroulant des commandes (en bas, à gauche)
  Chacune de ces commandes peut être choisie avec la souris ou avec le clavier (en tapant la lettre entre parenthèses). L'effet de ce choix peut être immédiat (Initial, Rien), nécessiter un clic de souris (Visibilité), un appui prolongé sur un bouton de la souris(Zoom), ou un glisser de souris (Rotation, Translation, Echelle, Face creuse).

- Le menu déroulant des composantes (en bas, à droite)
  À l'exception de certaines commandes (Rotation, Initial), qui s'appliquent seulement à l'ensemble des composantes, ce menu permet de spécifier si les commandes s'appliqueront à toutes les composantes/faces (c'est-à-dire à toute la figure) ou bien seulement à une composante/face particulière. Les composantes/faces peut être choisies avec la souris, et les composantes numérotées de 0 à 9 peuvent même être choisies avec le clavier, en tapant le chiffre correspondant.



Document 5. La fenêtre des commandes

Décrivons maintenant brièvement l'effet de chacune de ces commandes.

Rotation

Fait tourner toute la figure autour de l'axe des X ou des Y, en fonction du glisser de la souris. Cette commande ne peut s'appliquer à une composante/face particulière : elle affecte globalement toute la figure.

Translation

Effectue une translation en fonction du glisser de la souris. Cette commande peut s'appliquer à une composante/face particulière ou à toute la figure. Quand elle s'applique à toute la figure, la translation se fait selon un vecteur du plan X-Y, en fonction du glisser de la souris. Quand elle s'applique à une face seulement, le déplacement s'effectue selon une direction perpendiculaire à la face.

Echelle

Effectue une homothétie (dilatation ou contraction) en fonction du glisser de la souris. Le centre d'homothétie est le centre de gravité de tous les sommets de la composante/face choisie. Si on a choisi "Toutes", elle s'applique à toutes les faces indépendamment les unes des autres.

Zoom

Effectue un *zoom in*, quand le bouton gauche est enfoncé, ou un *zoom out*, quand le bouton droit est enfoncé. Un mouvement de la souris change le centre du zoom.

Visibilité

Après avoir choisi cette commande et la composante visée, un clic rendra invisible la composante en question si elle est visible, et la rendra visible si elle est invisible.

Face creuse

Après avoir choisi cette commande et la composante visée, un glisser de souris augmentera ou réduira un trou dans la composante/face en question (ou simultanément dans toutes les faces).

Rien

Une fois cette commande choisie, aucune action de la souris dans la fenêtre de la figure n'est prise en compte. Cependant, on peut encore utiliser la souris pour sélectionner des commandes ou modifier des glissières.

Terminons en mentionnant les boutons, qui nous permettent

- *de réinitialiser la figure, c'est-à-dire de la retracer dans sa position originale, à la fin des étapes 1 (**definirFigure**) et 2 (**initialiserFigure**), comme si l'utilisateur n'avait jamais débuté ses interactions avec celle-ci*
- *d'enregistrer l'image de notre figure : une « photo » de son état actuel est placée dans le dossier de téléchargement*
- *d'enregistrer une série d'images de notre figure (de façon à fabriquer une animation par la suite).*

Un exemple de figure plus complexe

Pour illustrer d'autres facettes de la définition de figures, nous allons décrire une sorte de « toît refermable », avec des glissières pouvant spécifier le nombre de triangles utilisés, et le degré de fermeture du toît (voir le document 6).

Document 6. Diverses positions de notre toît

Le document 7 nous révèle la description de la figure que nous avons utilisée. Jetons-y un coup d'œil.

À l'étape 1, dans la fonction **definirFigure**, nous commençons par définir trois glissières : une pour spécifier le nombre de triangles formant notre toît (entre 3 et 20), une autre pour fixer la hauteur des sommets, et une dernière pour choisir le rayon du cercle sur lequel se trouveront les sommets du haut. Nous décrivons ensuite la position des sommets à la base du toît : régulièrement espacés sur un cercle dans le plan X-Z. De même, nous décrivons la position des sommets qui serviront à « refermer le toît » : régulièrement espacés sur un cercle dans un plan parallèle au plan plan X-Z, et décalés par rapport aux sommets de la base. Puis nous utilisons ces sommets pour définir les faces. Notons au passage que nous utilisons les numéros de ces faces pour « calculer » leur couleur.

Soulignons ici que nous devons créer à cette étape tous les sommets et toutes les faces (soit 20, le maximum). Plus tard, nous ne pourrons que redéfinir les coordonnées de certains sommets, et décider quelles faces seront inclues ou exclues de la figure.

À l'étape 2, dans la fonction **initialiserFigure**, nous nous contentons d'incliner la figure vers l'utilisateur.

À l'étape 3, dans la fonction **redefinirCertainsSommets** (qui est appelée périodiquement par SPAF), on lit la valeur des trois glissières et on modifie la figure en conséquence : changement de la position des sommets mobiles, inclusion des faces retenues, et exclusion des autres.

```
function definirFigure(){
  pixelsParUnite(150);


  // ajusterGlissiere si nécessaire
  creerGlissiereG1(3, 20, 20, 1, "nb de  triangles");  // nombre de côtés du polygone
  creerGlissiereG2(-4, 4, -4, 0.01, "hauteur");    // hauteur du sommet
  creerGlissiereG3(0, 2, 2, 0.01, "rayon");    // rayon supérieur


  // Définition des sommets
  for(var k=0;k<=20;k++) {
    definirSommet(k+1,2*cos(k*PI/10),0,2*sin(k*PI/10));
    definirSommet(k+22,2*cos(k*PI/10+PI/20),-2,2*sin(k*PI/10+PI/20));
  }


  // Définition des faces
  var coul;
  for(var  k=1;k<=20;k++) {
    definirFace(k, k,k+1,k+21);
    if (k % 2 == 0) {coul=round(k*255.0/20);} else {coul=255-round(k*255.0/20);}
    couleurFace(k,255, coul, 255-coul);
  }
}

function initialiserFigure(){
   fixeRotationX(20);
}

function redefinirCertainsSommets(){
   var nb = round(valG1);
   var h = valG2;
   var r = valG3;

   for(var k=0;k<=nb;k++) {
     redefinirSommet(k+1,2*cos(k*2*PI/nb),0,2*sin(k*2*PI/nb));
     redefinirSommet(k+22,r*cos(k*2*PI/nb+PI/nb),h,r*sin(k*2*PI/nb+PI/nb));
   }
   for(var k=1;k<=nb;k++) {
     integrerDansFigure(k,true);
   }
   for(var k=nb+1;k<=20;k++) {
     integrerDansFigure(k,false);
   }
}
```

Document 7. Description de notre figure

Incorporation d'une figure SPAF dans une page web

Par défaut, SPAF utilise une nouvelle page web pour chacune de ses figures, et place celle-ci en haut et à gauche de la page. Cependant, l'utilisateur avancé pourrait vouloir incorporer une figure SPAF dans une page web existante. Voilà comment procéder :

- dans le fichier *html* de la page web, on ajoute des instructions du type suivant dans la section *head* de la page
  `<script language="javascript" type="text/javascript" src=`*cheminVersFichier*`></script>`
  pour tous les fichiers JavaScript requis : *p5-SPAF.js, p5.dom.js, SPAF.js, Face.js et Figure.js*
- dans le fichier *html* de la page web, on ajoute une instruction du type suivant dans la section *body* de la page
  `<div id="contenantSPAF" style="height:500px; width:500px;"> </div>`
  (on suppose ici qu'on veut ramener notre figure SPAF à des dimensions 500x500)
- dans la fonction *initialiserFigure* du fichier JavaScript de notre figure, on ajoute les instructions suivantes
  ```
  cacherControles(); libererClavier(); canvasSPAF.resize(500,500);
  divSPAF = select('#contenantSPAF'); // où divSPAF est une variable globale
  ```
- dans la fonction *redefinirCertainsSommets* du fichier JavaScript de notre figure, on ajoute les instructions suivantes
  ```
  var posXY = trouverPosition(divSPAF.elt);
  canvasSPAF.position(posXY[0],posXY[1]);
  ```

Par souci de complétude, mentionnons que les fonctions *montrerControles* et *bloquerClavier* existent aussi, mais sont peu utilisées. Il existe aussi une fonction *debloquerTouches*, qui débloque les touches passées en argument. Par exemple, *debloquerTouches('r', 'p')* va débloquer la rotation et la possibilité d'ouvrir le menu pour prendre une photo. Mentionnons en terminant que nous pouvons déterminer l'état initial de SPAF par une appel, dans la fonction *redefinirCertainsSommets*, à la fonction *etatInitialSPAF(touche)*, où *touche* est l'une des touches du menu *Actions* (soit 'r', 't', 'e', 'z', 'v', 'c', ou 'a').

En terminant…

Nous sommes arrivés à la fin de cette brève documentation. Cependant, vous pouvez continuer à vous familiariser avec SPAF en consultant les exemples fournis avec le logiciel. Ceux-ci se retrouvent dans le dossier compressé SPAF.zip, disponible sur le site web de SPAF-p5, qui contient deux dossiers : le dossier *SPAF_imitial*, qui correspond en fait à une figure vierge, et le dossier *Exemples*. Ce dernier contient un ensemble de fichiers décrivant les divers exemples : on exécute l'exemple *NomFigure* en double-ciquant le fichier *NomFigure.html*, tandis qu'on peut modifier la figure en question en éditant le fichier *NomFigure.js*.