Livret de projets

Préambule:
Pour certains projets, il est nécessaire de savoir manipuler les chaînes de caractères (fonctions utiles: CaractereEn(ch,i) et Longueur(ch)). Vous devrez aussi parfois manipuler des tableaux (la fonction à connaître est Taille(tab)).

Un certain nombre de projets demande d'utiliser les fonctions graphiques de l'environnement de programmation suivantes:
•*Point(x,y,c)* : qui trace un point aux coordonnées x, y de couleur c (c est une chaîne de caractères qui décrit la couleur (soit 'red', 'green',...; soit 'rgb(150,100,120)' (nb: la fonction rgb(r:entier, g:entier, b:entier) : chaîne permet de construire la chaîne de caractère décrivant une couleur en fonction de ses composantes de rouge, vert et bleu..
•*Rectangle(x,y,l,h,c)* et *RectanglePlein(x,y,l,h,c)* qui trace un rectangle (rempli ou non) de taille l*h à partir de la position x, y.
•*Ligne(x1,y1,x2,y2,c)* : qui trace une ligne de couleur c entre les points (x1,y1) et (x2,y2).

Enfin, quelques projets nécessitent l'utilisation des fonctions de manipulation de sons suivantes:
•*var son=CreerSon(data,frequence)* qui crée un son à partir du tableau *data* (un tableau d'entiers compris entre 0 et 255) qui décrit la courbe d'un son et *frequence* est la fréquence d'échantillonnage du son (ex: pour un son d'une seconde à une fréquence de 22050 Hz, il faut un tableau contenant 22050 valeurs...). Le résultat est un «objet» sonore.
•*son.play()* permet de jouer un objet sonore (sortie sonore).
•*var son=ChargerSon(url)* permet de charger un son situé à l'adresse *url* donnée. Le résultat est un objet sonore. Rq: si l'on souhaite lire un fichier sonore qui se trouve sur le gestionnaire de fichier interne, il faut combiner cette fonction avec *readFile(nom)* de cette manière: *var son=ChargerSon(readFile('Data/exempleson.wav')).*
•*var data=ImportSon(url) permet d'obtenir le tableau d'entier stocké dans un fichier wav en vue de le manipuler. Il est souvent utile d'obtenir également sa fréquence d'échantillonnage ce qui est fait grâce à la fonction SampleRate(url). Rq: même remarque que précédemment en ce qui concerne les fichiers stockés sur le gestionnaire de fichiers interne.*

Dans un certain nombre de projets, il faudra jouer une animation à une cadence fixée et ou définir des interactions avec la souris ou le clavier, vous pouvez consulter sur Madoc la page « Gestion du clavier, de la souris et animations en algoscript ».

Vous vous référerez un maximum à l'aide intégrée à l'interface qui contient des exemples très utiles pour ces projets.

Une indication du niveau de difficulté est fournie. L'ambition sera privilégiée, à condition évidemment d'arriver au bout du sujet.

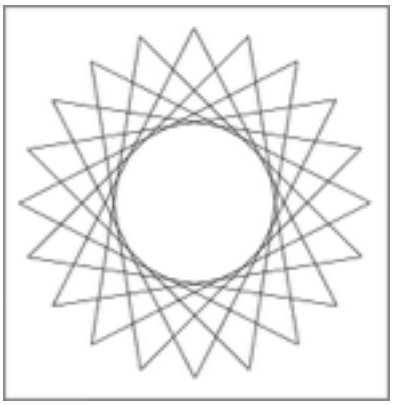
Sujet no 1 : Graphismes 1

Parcourez tous les points de l'écran (de taille 600 pixel sur 400 pixel) et attribuez à chaque point une couleur qui dépend (éventuellement) de sa position (x,y):

1. Tous les points doivent être en rouge
2. Tous les points tels que x<300 doivent être en rouge, les autres doivent être en bleu
3. Tous les points prennent comme couleur rgb(A,B,C) où A=couleur(x), B=couleur(y) et C=couleur(x+y) et couleur est une fonction à définir telle que

   couleur(i)=Math.floor(125 * Math.cos(i * 3.14 / 180) + 127);
4. Tous les points du cercle de centre (300,200) et de rayon 180 prennent comme couleur rgb(A,B,C) où A=couleur(x), B=couleur(y) et C=couleur(x+y) et ceux à l'extérieur prennent comme couleur rgb(100,100,D) où D=couleur(2*y)
5. Refaire toutes les questions en remplaçant le tracé de points par des rectangles de taille 2x2, puis 4x4, point 8x8. Le but sera de remarquer qu'au détriment d'une qualité amoindrie, le tracé peut-être beaucoup accéléré.
6. Mettre au point une animation qui permet de faire passer doucement l'écran d'un dégradé de rouge à un dégradé de bleu.
7. Proposez d'autres rendus graphiques jolis.

Sujet no 2 : Graphismes 2



On souhaite obtenir des figures de ce type composées de suites de lignes tracées entre deux points d'un cercle espacés d'un certain angle.
Dans cette figure, on remarque que l'on arrête de tracer des traits dès qu'on tombe sur un point par lequel on est déjà passé. On va donc définir un tableau de 360 booléens chargés de mémoriser les points par lesquels on est déjà passé. Créer un tableau de 360 cases et initialisez ces cases à faux (boucle qui parcours les 360 cases et leur assigne la valeur faux).
Ecrire une fonction secante(*x,y,rayon,angle1,angle2*) qui trace un trait entre les points d'angles *angle1* et *angle2* sur le cercle de centre (*x,y*) et de rayon *rayon* (en maths, ce trait s'appelle une sécante d'où le nom de la fonction).
Ecrire l'algorithme qui demande à l'utilisateur de saisir un angle et qui trace cette figure (il y a une boucle qui doit s'arrêter dès que l'on tombe sur un point déjà visité. Tester avec 90, 120, 126, 121 et 179.

Sujet no 3 : Graphismes 3

Le but de cet exercice est d'afficher une horloge qui donne l'heure courante. Nous procèderons par étape:

1. Ecrire une fonction cadran() sans paramètre qui affiche un cadran (la partie en bleu de la figure) à savoir des petits traits pour chacune des 60 minutes (il faut faire une boucle) et des traits plus longs pour tous les multiples de 5 minutes (il faut faire un test).
2. Ecrire une fonction Aiguille(longueur, couleur, n) qui affiche une aiguille (un trait partant du centre du cercle) de longueur et couleur donnée correspondant à la n-ème minute (ou seconde). Cette fonction servira pour tracer les trois aiguilles d'heure, minutes et secondes en prenant des paramètres différents.
3. Ecrire une fonction AfficheHeure(h,m,s) qui prend en paramètre trois entiers et affiche le cadran, puis affiche les trois aiguilles aux bons endroits (vous ferez en sorte que l'aiguille de l'heure s'affiche exactement où elle doit être affichée, cela implique de prendre aussi en compte les minutes lors de l'affichage de l'heure).
4. Ecrire un algorithme qui utilise l'objet Date pour récupérer l'heure courante et l'afficher de manière graphique. Ex d'utilisation:

```
var aujourdhui=new Date();
var heure=aujourdhui.getHours();
var minutes=aujourdhui.getMinutes();
var secondes=aujourdhui.getSeconds();
```

5. [Facultatif] vous pourrez essayer d'animer vous horloge de manière à ce que toutes les secondes, l'écran soit effacé et l'horloge soit réaffichée avec l'heure courante.

Sujet no 4 :

Le but de ce projet est de réaliser une saisie contrôlée de date (sous le format JJ/MM/AAAA) puis de réaliser quelques calculs autour de cette date. Voici les différentes étapes à réaliser:

1. Ecrire une fonction BonFormat(ch) qui retourne le booléen vrai ou faux selon que la chaîne ch est formatée sous la forme d'une date JJ/MM/AAAA où les J, M et A sous des chiffres ou non. Par exemple, BonFormat('13/02/2012') doit retourner vrai (true).
2. Ecrire une fonction SaisieDate() qui retourne une chaîne de caractère correctement formatée pour stocker une date. Globalement, il s'agit de demander la Saisie d'une chaîne de caractères jusqu'à obtenir une chaîne dans le bon format.
3. Ecrire trois fonctions Jour(ch), Mois(ch), Annee(ch), où ch est une chaîne décrivant une date (bien formatée) qui retournent toutes un entier égal respectivement au jour, au mois ou à l'année de cette date.
4. Ecrire une fonction bissextile(annee) qui retourne vrai si annee est bissextile et faux sinon.
5. Ecrire une fonction JourDeLaSemaine(ch) qui retourne une chaîne de caractère, le jour de la semaine de la date donnée (par exemple JourDeLaSemaine('13/02/2012') doit retourner 'lundi'. Il faudra faire des boucles pour avoir le bon décalage en tenant compte des année bissextiles. Indice: le 13/02/2011 est un dimanche, le 13/02/2012 un lundi et le 13/02/2013 un mercredi (décalage de 2 jours car 2012 est bissextile).

Sujet no 5 : Chaînes de caractères 2

Le but de ce projet est de mettre au point un codage de Vigénère dont voici le principe: on associe à chaque lettre un entier (A -> 0, B->1, ..., Z->25). Pour coder un mot (tout écrit en majuscule et sans ponctuation), par exemple 'BONJOUR', on utilise une clé (tout écrit en majuscule et sans ponctuation), par exemple 'CLE' de la manière suivante:
le 'B' est codé à l'aide du 'C' de la clé par 'B'+'C'=1+2=3='D', le 'O' à l'aide de la seconde lettre de la clé, le 'N' à l'aide de la 3ème lettre, puis on recommence avec la première lettre de la clé, ce qui donne:

Message	B	O	N	J	O	U	R
	1	14	13	9	14	20	17
Clé	C	L	E	C	L	E	C
	2	11	4	2	11	4	2
Message codé	D	Z	R	L	Z	Y	T
	3	25	17	11	25	24	19

1. Ecrire une fonction encode(message,cle) qui retourne le message codé selon la méthode de Vigénère (vous expliquerez comment vous comptez traiter le cas de accents et des ponctuations).
2. Ecrire une fonction decode(message, cle) qui effectue le contraire de la fonction précédente. Le but de cette question n'est pas de réécrire la fonction précédente mais il faudra s'arranger pour l'utiliser intelligemment.
3. Combien existe-t-il de décodages possibles pour une clé de longueur 5 (donner la ligne de code javascript qui permet de faire le calcul). Est-ce réaliste d'afficher tous les décodages possibles et de demander à un humain de trouver celui qui est le bon ?
4. NB: la longueur de la clé est un indicateur déterminant dans le codage de Vigénère. Il existe des méthodes pour essayer de la déterminer lorsque celle-ci est inconnue. Nous programmerons celle appelée «Test de Friedman» qui repose sur le calcul d'un indice de coincidence (formule ci-contre, où A est l'alphabet {A,B,...,Z} n_a est le nombre de fois où la lettre a apparaît et n est le nombre total de lettre prises en compte).
$$I = \frac{\sum_{a \in A} n_a(n_a - 1)}{n(n-1)}$$
Quand le message ne veut rien dire, I vaut environ 0.038 et quand le message est du français ou de l'anglais (codé) alors I vaut environ 0.075. Le test de Friedman consiste donc tester une longueur de clé de 1 (i.e., prendre toutes les lettres du message), calculer les fréquences des lettres sélectionnées et calculer I, puis tester une longueur de clé de 2 (i.e., prendre une lettre sur 2 du message), calculer les fréquences des lettres sélectionnées et calculer I, etc,... jusqu'à ce que le I calculé soit plus proche de 0.75 que de 0.038. Il faudra retourner la longueur pressentie.
5. Faire plein de tests qui prouvent que ça fonctionne !

Sujet no 6 : Graphisme 4: sprites

Le but de ce projet est de gérer des images stockées dans un tableau.

On va maintenant mettre au point un procédé de manipulation d'objets graphiques (souvent appelés sprites en anglais qui sont à la base de la plupart des animations). En gros, une image est définie par deux choses: une palette de couleurs utilisables (un tableau de 16 cases contenant des définitions de couleurs (ex: Palette[5]='red',...); un sprite (de taille 10x10) sera défini par un tableau en 2 dimensions (ie., une tableau de taille 10 dont chaque case est un tableau de taille 10), chaque cellule/cas du tableau contenant un numéro (entre 0 et 15) correspondant à une couleur définie.

1.Définir une palette de 16 couleurs;

2.Définir un sprite (vous êtes libre du dessin que vous voulez faire apparaître);

3.Ecrire une fonction AfficheSprite(x,y,l,sprite) qui affiche le sprite passé en paramètre en position (x,y) (pour son premier point) à l'aide de carrés de taille l.

4.Ecrire les fonctions RetournementVertical(sprite) et RetournementHorizontal(sprite) qui retournent toutes deux un sprite égale respectivement au retournement vertical du sprite (le sprite sera affiché la tête en bas) puis à son retournement horizontal.

5.Vous testerez toutes ces fonctions dans un algorithme ambitieux.

Sujet no 7 : Automates cellulaires

Les automates cellulaires sont des objets mathématiques/informatiques qui permettent de représenter des phénomènes biologiques compliqués par des règles «locales» très simple qui modifient un environnement donné. Par exemple, les motifs de certains coquillages, comme les cônes et les cymbiolae, sont générés par des mécanismes s'apparentant au modèle des automates cellulaires. Les cellules responsables de la pigmentation sont situées sur un bande étroite le long de la bouche du coquillage. Chaque cellule sécrète des pigments selon la sécrétion (ou l'absence de sécrétion) de ses voisines et l'ensemble des cellules produit le motif de la coquille au fur et à mesure de sa croissance.

Dans ce projet, nous étudierons une version simplifiée des automates cellulaires. Une génération de l'automate cellulaire sera représentée par une chaîne de caractère (de longueur 50) composée de 1 et de 0. Pour passer d'une génération à l'autre il faut appliquer une règle (décrite sous forme d'un tableau de 0 cases contenant 0 ou 1. Cette règle sera appliquée de la manière suivante:

indice	0=000	1=001	2=010	3=011	4=100	5=101	6=110	7=111
règle	0	1	1	1	1	0	0	0

Génération n : 0 1 0 0 1 0 1 0 0 1 0 1 1

Génération n+1 : 0 1 1 1 1 0 1 1 1 1 0 1 0

Dans cet exemple, les caractères de la génération n+1 sont calculés en fonction des 3 caractères (en position i-1, i, i+1) de la génération précédente (il faut tenir compte des débordement pour les caractères des bords de la génération) et en appliquant la règle donnée par le tableau (l'indice est calculé selon la formule 4*CaractereEn(generation,i-1)+2* CaractereEn(generation,i)+1* CaractereEn(generation,i+1).

Ecrire la fonction AppliquerRegle(regle,generation) qui calcule une génération en fonction de la précédente.
Ecrire une fonction JoliAffichageLettres(generation) qui retourne une chaîne de caractères où tous les 0 de génération sont remplacés par des espaces et tous les 1 par des 'X'.
Ecrire un algorithme qui demande la saisie d'une règle (remplissage de toutes les cases du tableau, réflechissez à la manière la plus conviviale de le réaliser), qui demande la saisie d'une génération, qui demande le nombre de générations à afficher et qui affiche successivement toutes les générations obtenues en appliquant la règle.
Améliorez deux choses à ce programme: ajoutez une création de génération aléatoire (de longueur saisie par l'utilisateur) et un affichage dans l'interface graphique des différentes générations.
Vous effectuerez de nombreux tests,

Sujet no 8 : Le puissance 4

Le but de ce projet est de programmer un jeu de puissance 4.

Concrètement, chaque colonne du puissance 4 sera codé par une chaîne de caractère (chaîne vide s'il n'y a pas de pion, les pions rouges sont codés par 'X' et le jaunes par 'O'). Le jeu complet sera codé par un tableau de ces chaînes de caractères. Tour à tour, les joueurs devront saisir le numéro de la colonne dans laquelle ils veulent mettre leur pion. Il donc être en mesure de gérer une partie à savoir, saisir les choix avec vérification de la validité de la colonne saisie (colonne non pleine); afficher le jeu courant, vérifier si la partie est terminée,...

1. Ecrire une fonction AfficheGrille(jeu) qui affiche le tableau de colonnes de manière conviviale (on pourra envisager un affichage graphique).
2. Ecrire une fonction SaisieColonne(jeu) qui demande la saisie d'une colonne et en vérifie la validité. Cette fonction retourne donc un entier.
3. Ecrire une fonction Termine(jeu) qui vérifie si un joueur à réussi à aligner qui pions de même couleur.
4. Programmer une partie du jeu.
5. Vous ferez jouer l'ordinateur en envisageant des stratégies plus ou moins complexes.

Sujet no 9 : Jouer une mélodie 1

Le but de ce projet est de proposer un outil qui joue une mélodie avec un son de piano (son qui devra être synthétisé par le programme lui-même).

Concrètement, il s'agira de

1. Proposer une méthode de stockage de la mélodie. Vous choisirez une ou deux mélodies (il est très facile de récupérer des partitions sur internet).
2. Générer (par synthèse sonore) un ensemble d'objets sonores pour chaque note à jouer. Se référer à l'exercice 1 des fiches de TP sur les applications sonores.
3. Jouer la mélodie.
4. Accompagner la mélodie d'une sortie visuelle jolie.

Sujet no 10 : Jouer une mélodie 2

Le but de ce projet est de proposer un outil qui joue une mélodie à la guitare (son qui devra être synthétisé par le programme lui-même).

Concrètement, il s'agira de

1. Proposer une méthode de stockage de la mélodie. Vous choisirez une ou deux mélodies (il est très facile de récupérer des partitions sur internet).
2. Générer (par l'algorithme de Karplus-Strong) un ensemble d'objets sonores pour chaque note à jouer. Se référer à l'exercice 2 des fiches de TP sur les applications sonores.
3. Jouer la mélodie.
4. Accompagner la mélodie d'une sortie visuelle jolie.

Sujet no 11 : Une boite à rythme

Le but de ce projet est de proposer un outil qui permet de jouer (en boucle) un rythme composé de plusieurs sons échantillonnés.

Concrètement, il s'agira de

1. Proposer une méthode de stockage du rythme.
2. Proposer une méthode de stockage des différents instruments. Vous récupèrerez sur internet un ensemble de sons au format wav et libres de droits.
3. Jouer le rythme donné (il faudra se donner un nombre maximal de boucles pour arrêter le son et une vitesse de lecture du rythme).
4. Proposez un éditeur visuel pour la boite à rythme.

Sujet no 12 : Le jeu du pendu

Le but de ce projet est de développer un jeu du pendu.

Plusieurs versions seront développées.

1. Dans une première version, le mot caché est demandé à l'utilisateur. le joueur propose alors des lettres jusqu'à avoir retrouvé le mot complètement. A chaque erreur, un compte (commençant à 10) est décrémenté. Le joueur gagne s'il trouve toutes les lettres en moins de 10 essais.
2. Proposer un affichage graphique de ce jeu.
3. Proposez une méthode permettant de stocker un ensemble de mots qui seront chargés grâce à la fonction readFile.
4. Terminez le jeu

Sujet no 13 : Le jeu de la vie

Le principe du jeu de la vie a été introduit par John H. Conway en 1970. Ce «jeu» a donné naissance plus tard à toute une théorie mathématique dite des automates cellulaires (cf. sujet no 7). Ce jeu a trouvé des applications également dans l'étude de quelques modes de développement de populations. Malgré des règles simples de reproduction et de décès, le mimétisme avec des processus «réels» est frappante.

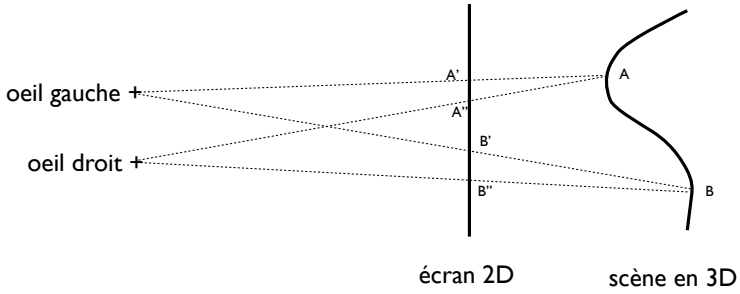
Grossièrement, le jeu de la vie se compose d'une grille sur laquelle évolues des cellules. Chaque point de la grille peut-être occupée par au plus une cellule. A chaque étape du jeu, des cellules peuvent apparaître ou disparaître selon la règle suivante :

- si une case vide est entourée d'exactement trois case pleines, alors une cellule nait dans cette case à la génération suivante;
- si une cellule vivante est entourée de deux ou trois cellules vivantes alors elle le reste sinon elle meurt.

1) Comment représenter une génération du jeu.
2) Ecrire la fonction AppliquerRegle(generation) qui calcule une génération en fonction de la précédente.
3) Ecrire une fonction JoliAffichageLettres(generation) qui retourne une chaîne de caractères où tous les 0 de génération sont remplacés par des espaces et tous les 1 par des 'X'.
4) Améliorez deux choses à ce programme: ajoutez une création de génération aléatoire (de longueur saisie par l'utilisateur) et un affichage dans l'interface graphique des différentes générations.

Sujet no 14 : Graphismes : les autostéreogrammes

Produire l'illusion de la 3D sur un écran en 2D est un problème que l'on se pose depuis que la photo a été crée. De nombreuses solutions ont été proposées, la plupart basées sur l'utilisation d'un matériel adapté (lunettes stéréoscopiques il y a longtemps, écrans 3D plus récemment). Cependant, il existe des solutions pour créer l'illusion de la 3D sans utiliser de matériel particulier, en maîtrisant une formule mathématique simple et pourvu que l'on soit capable de «loucher» un petit peu. Les techniques utilisées s'appellent alors stéréogrammes ou auto-stéréogrammes et repose sur le schéma suivant:



Le problème va être d'attribuer aux deux points A' et A" de l'écran la même couleur puisqu'ils correspondent au même point de la scène en 3D. Le calcul des coordonnées de A' et A" dépendent normalement de beaucoup de paramètres (la distance des yeux par rapport à l'écran, de la scène par rapport à l'écran, de l'écartement des yeux,....). Cependant, on obtient une approximation réaliste en fixant l'abscisse de A' comme étant la même que celle de A et en calculant la distance lA'A"l par la formule suivante:
lA'A"l=[360*(1-lAA'l/3)*(2-lAA'l/3)].

1) Créez une image de taille 800x600 dont tous les points ont une couleur aléatoire
2) Créez une image de taille 800x600 dont tous les points ont une couleur aléatoire sauf les points d'un cercle de centre (400,300) et de rayon 200 qui seront rouges.
3) Créez une image de taille 800x600 dont tous les points ont une couleur aléatoire mais aussi dont tous les points d'un cercle de centre (400,300) et de rayon 200 sont «éloignés» de l'écran d'une distance lAA'l égale à 0.5. Afin d'éviter les problèmes de recouvrement, vous utiliserez un tableau «memecouleur[A"]» qui contient l'abscisse de A'.Faites varier la distance en fonction du rayon du cercle pour obtenir d'autres effets 3D.
4) Imaginer d'autres images encore plus frappantes (par exemple des cônes en autostéréogramme).

Sujet no 15 : Algorithmique numérique

Le but de ce projet est de résoudre numériquement l'équation f(x)=0. Deux méthodes seront abordées: une résolution dichotomique (qui ne s'appliquera que dans des cas particuliers) et la méthode de Newton, plus générale.
La première méthode, dite de la résolution **dichotomique** repose sur l'hypothèse que dans un certain intervalle [a,b], la fonction f s'annule une seule fois et les signes de f(a) et f(b) sont différents. Le méthode dichotomique va donc consister à chaque étapes à découper l'intervalle [a,b] en deux en ne gardant que le sous-intervalle qui contient la solution. L'algorithme s'arrête que l'intervalle de recherche est très petit (de taille inférieure à une certaine précision). Ecrire la fonction RacineDichotomique(a,b,precision) qui retourne la solution qui se trouve dans l'intervalle [a,b]. On suppose que la fonction f a également été définie.
La deuxième méthode est plus générale. Cette méthode, appelée méthode de **Newton** (ou de Newton-Raphson) nécessite uniquement que la fonction f soit dérivable (dans la suite, on notera f' cette dérivée). Elle repose sur un développement de Taylor à l'ordre 1 de f. En effet, $f(x) \approx f(x_0)+f'(x_0)\ (x-x_0)$ et ainsi, si x est la solution recherchée (i.e., f(x)=0), alors $x \approx x_0 - f(x_0)/f'(x_0)$. L'algorithme de Newton repose donc sur la calcul de la suite $x_{k+1} = x_k - f(x_k)/f'(x_k)$ jusqu'à ce que x_k et x_{k+1} soient à une distance inférieure à une précision données. Ecrire la fonction RacineNewton(x_0,precision) qui calcule la solution de l'équation f(x)=0 en lui fournissant le premier point x_0 de la suite. A noter: lorsqu'il est difficile d'obtenir une formule pour la dérivée de f, il est possible de remplacer la dérivée f' par la sécante. Dans ce cas, la méthode va consister à calculer la suite $x_{k+1} = x_k - (x_k-x_{k-1})\ f(x_k)/(f(x_k)-f(x_{k-1}))$. Ecrire la fonction RacineSecante(x_0, x_1, precision) qui calcule la solution de l'équation f(x)=0 en lui fournissant les deux premiers points x_0 et x_1 de la suite.

Vous appliquerez les deux méthodes sur des exemples bien choisis et en étendierez leur efficacité (en terme du nombre d'itérations nécessaires pour calculer les résultat attendu).

Sujet no 16 : Le plus court chemin

Calculer un plus court chemin sur une carte routière ou plus généralement dans un graphe est une tâche très classique. L'algorithme de Dijkstra, publié par l'informaticien néerlandais Edsger Dijkstra en 1959, permet de calculer le plus court chemin entre deux villes données dans un graphe orienté pour des coûts ayant des valeurs strictement positives (c'est typiquement le cas des cartes routières). Le plus de ce projet est d'illustrer cet algorithme.

1) Le premier problème consiste à encoder la carte routière. On supposera dans la suite que la carte contient n villes numérotées de 0 à n-1. Un tableau à deux dimensions, dont la (i,j)-ème case contient la valeur infinie s'il n'y a pas de route entre les villes i et j et qui contient la distance (par la route) entre les deux villes sinon, permettra de définir s'il existe une route entre deux villes. Créez le tableau correspondant aux villes françaises de plus de 200000 habitants (cf. http://www.linternaute.com/ville/classement/villes/population). Les distances seront celles obtenues par autoroute uniquement. Créez un autre tableau à n lignes et deux colonnes permettant de stocker les coordonnées géographiques (en x dans la première colonne et y dans la deuxième colonne) de toutes les villes. Utilisez ces deux tableaux pour afficher graphiquement la carte autoroutière de ces villes (évidemment, chaque autoroute sera représentée par une droite reliant deux villes de la carte.
2) Programmez l'algorithme de Dijkstra permettant de calculer le plus court chemin entre deux villes. Le code peut-être trouvé dans une version Algorithmique ici : http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm.
3) Ecrivez un programme qui demande à l'utilisateur de saisir deux villes, trace la carte autoroutière et affiche en rouge le plus court chemin trouvé par l'algorithme de Dijkstra.
4) Faites un test sur une carte de France plus grande.

Sujet no 17 (6 sujets en 1 seul): Traitement de son en temps réel

Il est très facile de récupérer et traiter le signal reçu par le microphone de son ordinateur. Ceci se fait de la manière suivante, en utilisant l'instruction « StartMicrophone(Traitement); » où Traitement est une fonction qui prend en paramètre un tableau T (un tableau de 2048 réels compris entre -1 et 1 qui décrivent environ 46 millièmes du signal sonore reçu par le micro). La fonction Traitement est alors appelée tous les 46 millièmes de secondes. On obtient donc très facilement un oscilloscope en définissant la fonction Traitement par:

```
function Traitement(T) {
  AfficherCourbe(T,false,false,'red','',1);
}

StartMicrophone(Traitement);
```

On peut alors imaginer énormément de traitements possible de ce signal et autant de sujets de projet:

1) Sujet de niveau 1: Faire un affichage qui change en fonction du niveau sonore du signal reçu.
1) Sujet de niveau 2: Rejouer le signal reçu en lui ajoutant un écho (paramétrable) ou tout autre effet.
2) Sujet de niveau 2: On obtient une estimation de la fréquence de base d'un signal en comptant le nombre de fois où le signal change de signe. Ceci fonctionne relativement bien lorsque ce signal est quasi sinusoïdal, comme dans le cas d'une note jouée au piano. Le but de ce sujet est de mettre au point un accordeur de piano.
3) Sujet de niveau 4: On obtient une estimation de la fréquence de base d'un signal en comptant le nombre de fois où le signal change de signe. Ceci fonctionne relativement bien lorsque ce signal est quasi sinusoïdal, comme dans le cas d'une note jouée au piano. Le but de ce sujet est de mettre au point un jeu de type *Guitar Hero*.
4) Sujet de niveau 5: Pour obtenir une valeur précise de la fréquence de base d'un signal, on utilise une transformation de Fourier qui permet d'en obtenir le spectre (et donc, la valeur maximale du spectre est obtenue pour la fréquence de base du signal). Ecrire un algorithme qui calcule la transformée de Fourier Discrète du signal, si possible en utilisant un algorithme de transformation de Fourier rapide. Puis utiliser ce spectre pour, au choix:
    1) faire un accordeur
    2) faire un jeu de Guitar Hero
    3) Tranformer le signal (faire un filtre sur les fréquences à jouer, décaler les fréquences pour transformer la voix, etc…)

Sujet no 18 : Traitement de vidéo en temps réel via une webcam

Il est très facile de récupérer et traiter le signal reçu par la caméra de son ordinateur. Ceci se fait de la manière suivante, en utilisant l'instruction « StartWebcam(Traitement); » où Traitement est une fonction qui prend en paramètre un tableau T (un tableau de 640*480 (plus précisément une « imageData » javascript) valeurs correspondant à chaque pixels de l'image en cours de traitement). Ce tableau se manipule facilement grâces aux deux fonctions getImagePoint et setImagePoint qui permettent respectivement de lire et modifier la couleur d'un point (en précisant en tableau de 4 entiers entre 0 et 255 correspondants aux composantes de rouge, vert et bleu de la couleur ainsi que sa transparence). La fonction Traitement est alors appelée environ 30 fois par seconde (en fonction des performances de la machine). On obtient donc très facilement des effets sur les vidéos. Par exemple, le programme suivant transforme l'image en noir ou blanc en fonction d'un seuil.

```
function Traitement(T) {
  var i, j, col;
  for (i = 0; i < 640; i = i + 1) {
    for (j = 0; j < 480; j = j + 1) {
      col = getImagePoint(T, i, j); // col est un tableau de 4 entiers
      if (col[0] + col[1] + col[2] < 50 * 3) {
        setImagePoint(T, i, j, [0, 0, 0, 255]);
      } else {
        setImagePoint(T, i, j, [255, 255, 255, 255]);
      }
    }
  }
  ctx.putImageData(T, 0, 0); // affichage de l'image obtenue sur la sortie graphique
}

StartWebcam(Traitement);
```

Proposer des outils de traitement de l'image en temps réel (retournement, changements de couleurs, ajout d'un fond à l'image, détection de formes, etc…).

Autres idées de sujet sans description

Le jeu du 2048 avec simulation de stratégies « automatiques » de jeu.

Certains exercices de TP « algorithmes pour les mathématiques » peuvent servir de base à un projet.