Python TP1

Fichiers

Objet_fichier = open(nom_fichier, mode) : ouverture d'un fichier.

Exemple : fic = open('fichierTP1.txt','wt')

```
Modes :
r, pour une ouverture en lecture (READ).
w, pour une ouverture en écriture (WRITE), à chaque ouverture le contenu du
fichier est écrasé. Si le fichier n'existe pas python le crée.
a, pour une ouverture en mode ajout à la fin du fichier (APPEND). Si le
fichier n'existe pas python le crée.
b, pour une ouverture en mode binaire.
t, pour une ouverture en mode texte.
x, crée un nouveau fichier et l'ouvre pour écriture
```

Pensez à fermer les fichiers : fic.close()

Lecture : fic.read()

Ecriture : fic.write(« mon texte \n»)

L'utilisation de with est assez pratique (fermeture auto) :

```
with open("data.txt", "r") as fic:
        print(fic.read())
```

Travail du TP

Vous pouvez travailler dans une VM ou sur votre portable personnel en installant les outils. Commencez avec un éditeur de texte en lançant le programme dans un shell puis passez à l'IDE PyCharm (par exemple).

1. Afficher « Bonjour le monde ! »
2. En mode console, proposez un petit menu : 1. Choisir un nom de fichier, 2. Ajouter un texte (que vous demanderez à l'utilisateur), 3. Afficher le fichier complet, 4. Vider le fichier, 9. Quitter le programme. Vous écrirez les fonctions associées aux choix.
3. a. Concevoir une classe Date (avec redéfinition (surcharge) de == (__eq__) et de < (__lt__)) ;
   b. Concevoir une clase Etudiant (avec une méthode adresselec qui fabrique l'adresse électronique prenom.nom@etu.univ-tours.fr et une méthode âge) ;
   c. Lire le fichier fichetu.csv et constituer une liste d'objets Etudiant.

Python TP2

Interface graphique tkinter

Tkinter (Tk interface) est un module intégré à la bibliothèque standard de Python, il offre un moyen de créer des interfaces graphiques via Python. Tkinter est disponible sur Windows et la plupart des systèmes Unix. Les interfaces que vous pourrez développer auront donc toutes les chances d'être portables d'un système à l'autre. Notez qu'il existe d'autres bibliothèques pour créer des interfaces graphiques. Tkinter a l'avantage d'être disponible par défaut, sans nécessiter une installation supplémentaire.

Pour installer des nouvelles bibliothèques, aller dans File/Settings/Project/Project Interpreter puis cliquer sur le + vert.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from tkinter import *

fenetre = Tk()

label = Label(fenetre, text="Hello World")
label.pack()

fenetre.mainloop()
```

Ensuite il suffit d'ajouter dans une fenêtre des éléments graphiques que l'on nomme widget (bouton, zone de saisie…). Il est possible aussi de créer une barre de menu.

Travail du TP

1. Faire une calculatrice (10 chiffres, 4 opérations, =, C, AC) qui récupère dans une chaîne les touches cliquées par l'utilisateur. C efface la dernière touche cliquée et AC efface toutes les touches bufferisées.
2. Résoudre le calcul et afficher le résultat (vous pouvez utiliser la fonction eval())
3. Ajouter un menu avec des commandes de votre choix (aide, mode basique/scientifique…)
4. Ajouter les parenthèses
5. Continuer d'améliorer votre calculatrice (décimale, puissance, trigo…)

Python TP3 : Exceptions et chiffrement

Exceptions

Les exceptions permettent de gérer les erreurs qui peuvent survenir sans « planter » le programme.

https://openclassrooms.com/courses/apprenez-a-programmer-en-python/les-exceptions-4

Hashage

Le hashage permet de stocker et vérifier un mot de passe sans le garder en clair dans un fichier.

```
import hashlib

password = 'mot de passe'
mdp = hashlib.sha512( password.encode() ).hexdigest()
```

Travail du TP

1. Reprenez vos programmes du TP1 (menu) et TP2 (calculatrice) et ajouter la gestion des exceptions (try, except, else, finally).

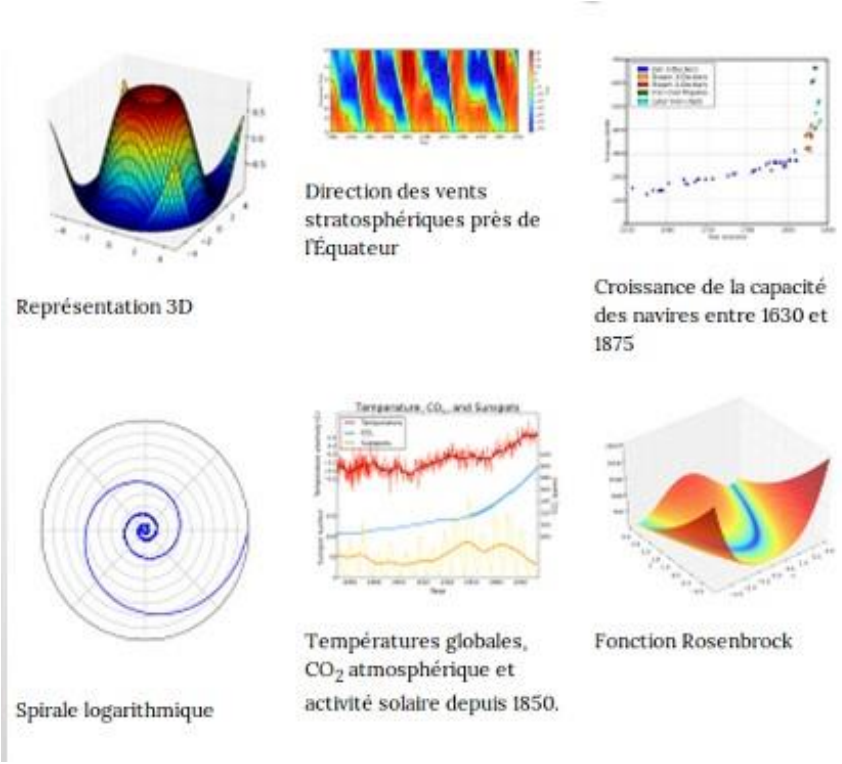
2. Lever une exception avec assert et raise.

3. Demander pour enregistrement un login (input) et un mot de passe à l'utilisateur avec getpass() en mode console ou avec une fenêtre tkinter affichant des * à la place des caractères saisis.
   Stocker le couple login et hash mot de passe dans un fichier texte.
   Demander pour vérification un login et un mot de passe, vérifier la présence dans le fichier.
   Utiliser le « salage » pour renforcer le système avec une chaîne comprenant le login et une partie fixe.

4. Demander à l'utilisateur le nom d'un fichier existant (par ex. un fichier texte que vous avez créé) que vous allez chiffrer dans un nouveau fichier en utilisant le mot de passe précédent avec l'algorithme AES 256. Vous pouvez utiliser la bibliothèque PyCryptodome http://pycryptodome.readthedocs.io/en/latest/ ou une autre si elle pose problème.
   Proposer aussi le déchiffrement du fichier.

Python TP4 : Matplotlib

Graphiques avec Matplotlib



http://matplotlib.org/

http://apprendre-python.com/page-creer-graphiques-scientifiques-python-apprendre

Travail du TP

1. Générer des nombres aléatoires
2. Afficher la courbe de ces données dans une fenêtre matplotlib
3. Afficher plusieurs courbes avec styles et couleurs variés
4. Modifier les noms des axes, la légende, ajouter des flèches pour montrer des zones…
5. Afficher un histogramme et un camembert
6. Afficher une surface 2D dans un espace 3D (mesh)

Python TP5 : Base de données

SQLite3

SQLite3 est une bibliothèque de gestion de bases de données relationnelle. Elle est déjà en général installée (windows, linux…).
Commandes indispensables : create, select…from…, insert.

Vous pourrez trouver des informations utiles sur les liens suivants :
https://docs.python.org/3.6/library/sqlite3.html
http://pythoncentral.io/introduction-to-sqlite-in-python/

Au cas où vous voudriez gérer les erreurs, les erreurs possibles sont : Error, DatabaseError, DataError, IntegrityError, InternalError, NotSupportedError, OperationalError, ProgrammingError, InterfaceError, Warning.

Vous pouvez aussi utiliser SQLAlchemy ORM : http://docs.sqlalchemy.org/en/latest/orm/

XML

https://docs.python.org/3/library/xml.etree.elementtree.html

Travail du TP

On dispose de plusieurs fichiers dans data_insee.zip :

- Un fichier communes.csv contenant différentes informations sur chaque ligne dont Code département, Code commune, Nom de la commune, Population totale. Attention au format du csv et aux espaces pour millier.
- Un fichier departements.csv contenant différentes informations sur chaque ligne dont Code département, Nom du département, Code région.
- Un fichier regions.csv contenant différentes informations sur chaque ligne dont Code région, Nom de la région.
- Un fichier ensemble.xls fournit à titre indicatif (ne sert pas dans le TP) à partir duquel ont été extraits les fichiers précédents, il contient l'ensemble des informations provenant du dernier recensement (2013). Ce fichier a été récupéré sur le site de l'INSEE (https://www.insee.fr/).
- Un fichier table-appartenance-geo-communes-16.xls qui contient les informations communales 2016 avec à la fin les nouvelles régions. 2 fichiers (communes-2016.csv et zones-2016.csv) ont été extraits de ce fichier.

Lire des fichiers csv est un exercice que vous avez fait dans le TP1.

1. Concevoir une base de données contenant les tables Communes, Departements, Regions à partir des 3 premiers fichiers ci-dessus et des seuls champs cités ci-dessus.
2. Calculer et afficher les populations totales de chaque département et région. Les comparer à la main avec les populations indiquées dans les fichiers.
3. Déterminer les communes ayant le même nom et un département différent. Afficher le nom de la commune suivi de la liste des n° de départements.
4. Ecrire une fonction pour sauvegarder la base dans un fichier XML et une autre pour charger la base à partir de ce fichier.
5. A partir des fichiers communes-2016.csv et zones-2016.csv, ajouter une table NouvellesRegions et un champ à la table Departements. Calculer les populations de ces nouvelles régions.

Python TP6 : Numpy et Scipy

Numpy

http://www.numpy.org/

Installation : télécharger sur http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy le fichier « numpy-1.11.2+mkl-cp35-cp35m-win32.whl » (pour python 3.5). Puis l'installer dans un shell (win-R / cmd) avec la commande (tab pour auto-complétion) : pip install numpy … .whl

Permet la création et l'utilisation de tableaux multidimensionnels d'éléments de même type (généralement des nombres). Les axes correspondent à la dimension du tableau. Le rang (rank) est le nombre d'axes du tableau, chaque axe a une longueur (nombre d'éléments).

Exemple : [[2., 3., 0.], [5., -3., 1.]] a un rang de 2 avec longueur de 2 pour le 1er axe et 3 pour le 2e. Son shape est (2, 3) et le size est 6.

Utilisation :

```
import numpy as np
a = np.array([2,3,4])
```

Scipy

Installation : télécharger sur http://www.lfd.uci.edu/~gohlke/pythonlibs/#scipy le fichier « scipy-0.18.1-cp35-cp35m-win32.whl » (pour python 3.5). Puis l'installer dans un shell (win-R / cmd) avec la commande (tab pour auto-complétion) : pip install scipy … .whl

Il contient des modules pour l'optimisation, l'algèbre linéaire, les statistiques, le traitement du signal ou encore le traitement d'images.

Travail du TP

Numpy

1. Créer un tableau de dimension 3 avec un shape de (4, 3, 2) remplit avec des nombres aléatoires.
   Vous afficherez les attributs du tableau : ndim, shape, size, dtype, itemsize, data.
2. Créer 2 matrices 3x3 initialisées avec les entiers de 0 à 8 pour la 1e et de 2 à 10 pour la 2e puis calculer le produit des 2 (différence entre * et dot). Transposer une matrice.
3. Calculer le déterminant et l'inverse d'une matrice. Résoudre un système d'équations linéaires. Calculer les valeurs et vecteurs propres d'une matrice.

Scipy

4. Approcher un ensemble de points par une courbe (optimize.curve_fit ou interpolate.interp1d).
5. Lire une image jpeg (matplotlib.imread) et afficher l'image originale et réduite en taille (matplotlib.imresize).
6. Explorer d'autres fonctionnalités de scipy (https://docs.scipy.org/doc/scipy/reference/).

Python TP7 : Serveur et page web

Serveur web

```
import http.server

PORT = 8888
server_address = ("", PORT)

server = http.server.HTTPServer
handler = http.server.CGIHTTPRequestHandler
handler.cgi_directories = ["/"]
print("Serveur actif sur le port :", PORT)

httpd = server(server_address, handler)
httpd.serve_forever()
```

Page web

```
#!/usr/bin/python3
import cgi

form = cgi.FieldStorage()
print("Content-type: text/html; charset=utf-8\n")

print(form.getvalue("name"))

html = """<!DOCTYPE html>
<head>
    <title>Mon programme</title>
</head>
<body>
    <form action="/index.py" method="post">
        <input type="text" name="name" value="Votre nom" />
        <input type="submit" name="send" value="Envoyer information au serveur">
    </form>
</body>
</html>
"""

print(html)
```

Travail du TP

1. Créer un serveur web
2. Créer une page web dans un fichier index.py. L'afficher : localhost:8888/index.py
3. Utiliser cette page pour stocker ou récupérer des informations envoyées par le web **sur le serveur**. http://localhost:8888/index.py?name= Toto%20Titi
4. Permettre l'accès aux données qu'aux utilisateurs avec un login+mdp enregistré
5. Regarder le framework Django pour aller plus loin
   http://apprendre-python.com/page-django-introduction-python
   https://www.djangoproject.com/

Python TP8 – Prog. asynchrone et fourmis

Programmation asynchrone

Multi-threading

https://openclassrooms.com/courses/apprenez-a-programmer-en-python/la-programmation-parallele-avec-threading

http://apprendre-python.com/page-programmation-asynchrone-python-thread-threading

Multi-processing : plus lourd à mettre en place que le multi-threading surtout au niveau partage de données en mémoire mais permet plus facilement de faire du multi-core (CPU).

https://docs.python.org/3/library/multiprocessing.html

https://docs.python.org/3/library/subprocess.html

Et un article un peu long : https://zestedesavoir.com/articles/1568/decouvrons-la-programmation-asynchrone-en-python/

Peinture avec fourmis

Lire (parcourir) le fichier ic2_Fourmis_Artificielles_vol2_chap_7-extrait.pdf qui explique le principe que nous allons mettre en œuvre.

Travail du TP

1. Tester un calcul (style calcul_long ci-dessous) avec du multi-threading et du multi- processing en observant l'utilisation des cœurs de votre CPU.

```
def calcul_long():
    n = 1E7
    while n>0:
        n -= 1
```

2. Faire un programme qui construit et affiche une peinture avec des fourmis. Les paramètres (page 205 du pdf) seront à mettre dans un fichier texte ou mieux XML.

L'interaction a pris une importance considérable dans certaines manifestations artistiques, au point d'acquérir le statut de performance. Ces performances sont présentées sous la forme d'interventions de l'artiste produisant son message en temps réel et en public. Certaines performances ont même introduit le public dans l'œuvre comme matériau passif (par exemple par la vidéo) ou comme concepteur. Le statut d'artiste est alors réduit (à l'essentiel ?) à celui qui a permis à d'autres d'accéder à la création artistique.

La suite de ce chapitre détaille deux approches à base de fourmis artificielles, dans le cadre de la production de peintures et de musique.

7.2. Des fourmis peintres

7.2.1. *La génération de peintures par les fourmis*

Les systèmes de peinture automatique qui s'inspirent, ou utilisent, des fourmis sont encore rares. Voici quelques travaux se rapportant à la production de peinture, ou d'image, par des agents-fourmis. Tout d'abord, nous pouvons citer le travail de E. Tzafestas [TZA 00], qui a utilisé des fourmis capables de ramasser et déposer de la peinture sur une image afin d'étudier l'émergence d'une forme de complexité, ainsi que le mécanisme de régulation qui se met en place dans ce cas. Avec des préoccupations plus artistiques, L. Moura et V. Ramos ont produit des peintures d'essaims[3] (*swarm paintings*) selon une approche qu'ils qualifient d'« art computationnel ». Ensuite, L. Moura, avec H. Garcia-Pereira, a poursuivi cette idée de « fabriquer les artistes qui fabriquent de l'art » (*Making the Artists that make the Art*) en utilisant une colonie de robots qui réagissent à leur environnement et aux modifications qu'ils y produisent en dessinant sur le sol [MOU 04]. P. Urbano a aussi étudié le rôle artistique que peuvent prendre les phéromones dans un système multi-agent [URB 05]. Enfin, G. Greenfield est reparti des travaux présentés dans ce chapitre. En particulier, il a étudié la question de l'évolution génétique des paramètres des fourmis [GRE 05]. Il a proposé une mesure d'évaluation (*fitness*) pour diriger automatiquement l'évolution des paramètres des peintures de fourmis, indépendamment de l'humain qui, comme nous le verrons dans la suite, a un rôle actif dans le cas de nos travaux.

7.2.2. *Description du modèle de fourmis pour la peinture*

Dans ce travail sur la création de peintures par des fourmis, les fourmis artificielles sont des agents capables de se déplacer sur une image virtuelle. On peut aussi parler d'une toile virtuelle dont les pixels correspondent au grain d'une toile à peindre. Tout

3. http://www.lxxl.pt/aswarm/aswarm.html.

comme les fourmis réelles, qui déposent des phéromones sur le chemin qu'elles empruntent, les fourmis artificielles déposent des phéromones virtuelles. Contrairement aux phéromones naturelles, qui ne se voient pas, mais se sentent, les phéromones virtuelles utilisées correspondent à de la couleur, ou, si l'on préfère, de la peinture, déposée par les fourmis. Pour être précis, les fourmis se déplacent sur la toile de pixel en pixel et font des choix de mouvement de façon stochastique, tout en privilégiant une sorte de conservation de leur mouvement (pour donner une impression de mouvement « naturel »). Deux types de mouvements sont envisageables, suivant le voisinage que l'on autorise pour la fourmi : dans une configuration 4-voisins, la fourmi peut continuer son mouvement tout droit ou bien tourner à angle droit (à gauche ou à droite), et dans une configuration 8-voisins, la fourmi peut continuer son mouvement tout droit (cela ne change pas par rapport à l'autre mode de déplacement) et tourner légèrement (à gauche ou à droite). La figure 7.1 illustre ces deux modes de déplacement (que, par la suite, nous appellerons respectivement « mouvement droit » et « mouvement oblique »).

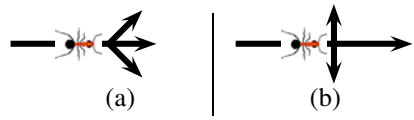


Figure 7.1. *Exemples de mouvements possibles pour une fourmi : (a) mouvement oblique (noté D_o), (b) mouvement (à angle) droit (noté D_d)*

Afin de caractériser le mouvement d'une fourmi, trois paramètres sont utilisés pour définir ses choix : sa probabilité de tourner à gauche (notée P_g), à droite (notée P_d) et, de façon complémentaire, sa probabilité d'aller tout droit (notée P_t)[4].

L'odeur déposée caractérise également une fourmi : comme il s'agit de peinture numérique, dans notre cas, la piste de phéromone déposée par une fourmi a une couleur définie par les composantes informatiques de la couleur[5], à savoir la proportion de rouge (C_R), de vert (C_G) et de bleu (C_B).

Si la modélisation des fourmis s'arrêtait ici, il n'y aurait pas lieu de parler de fourmis artificielles. Les fourmis que nous proposons sont donc capables de percevoir localement leur environnement, c'est-à-dire les odeurs-couleurs présentes dans les pixels voisins. En plus de la couleur qu'elle dépose, une fourmi est aussi caractérisée par une odeur qui l'attire (qui peut être totalement différente). Cette couleur d'attirance est également définie par les trois composantes RVB : (S_R, S_V, S_B).

4. Notons que le caractère stochastique du mouvement d'une fourmi implique : $P_g + P_d + P_t = 1$.

5. On parle des composantes RVB de la couleur pour « rouge », « vert » et « bleu », qui sont, pour les trois composantes, des valeurs entières de $\{0, \ldots, 255\}$.

A chaque pas, la fourmi inspecte son voisinage à la recherche de l'odeur qui l'attire. Pour déterminer si la couleur perçue par une fourmi correspond à la couleur qu'elle tente de suivre, nous avons introduit une distance simple entre deux couleurs, en nous basant sur un aspect de la perception que l'on peut avoir des couleurs : la luminance. Ainsi, une fourmi, pour reconnaître la couleur qu'elle cherche à suivre, compare les luminances des couleurs qu'elle rencontre. En dessous d'un seuil, paramétrable, la différence de luminance est considérée comme négligeable et la couleur-odeur est acceptée par la fourmi.

Nous avons utilisé la formule suivante pour le calcul de luminance :

$$\mathrm{Lum}(R, G, B) = 0{,}242\,6 \cdot R + 0{,}715\,2 \cdot V + 0{,}072\,2 \cdot B, \tag{7.1}$$

où R, V et B sont les composantes du pixel inspecté par la fourmi. La différence de luminance est alors :

$$\Delta(S_R, S_V, S_B, R, V, B) = |\mathrm{Lum}(S_R, S_V, S_B) - \mathrm{Lum}(R, V, B)|. \tag{7.2}$$

Par la suite, le seuil pour la valeur Δ, calculée ci-dessus, a été fixé à 40.

Quand une fourmi a reconnu la couleur qu'elle cherche à suivre, elle a une probabilité P_s de suivre cette couleur (et donc de se déplacer sur le pixel qui la contient).

L'objectif de ce mécanisme de recherche d'une couleur particulière est d'introduire une compétition entre les fourmis, qui se traduit par une lutte, inconsciente, pour la meilleure représentation de sa couleur (C_R, C_V, C_B), tout en recouvrant la couleur (S_R, S_V, S_B).

Pour permettre un rendu plus varié, nous avons ajouté un paramètre de taille à chaque fourmi. Cette taille est utilisée pour la diffusion de l'odeur de la fourmi sur la toile. Une taille de 0 indique que la fourmi modifie la couleur du pixel où elle se trouve en y mettant la couleur (C_R, C_V, C_B). Dans ce cas, cela signifie qu'il n'y a pas de diffusion sur les pixels voisins. Une taille de 1 provoque une diffusion (donc un mélange avec les odeurs déjà présentes) de l'odeur déposée par la fourmi. Pour traduire cette action, nous utilisons un produit de convolution discret, calculé sur les neuf pixels voisins (c'est-à-dire un pixel voisin, tout autour de la fourmi). Nous utilisons la matrice de convolution suivante :

$$M_c = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Cette notion de taille de diffusion a été étendue jusqu'à trois : le produit de convolution est appliqué sur un voisinage de 7×7 cases (c'est-à-dire trois pixels voisins, tout autour de la fourmi).

Enfin, l'environnement de déplacement des fourmis est toroïdal : la toile n'a pas de bord, quand les fourmis « sortent » d'un côté, elles réapparaissent sur le côté opposé. Cette technique nous permet d'afficher la peinture obtenue sous la forme d'une mosaïque sans séparations apparentes.

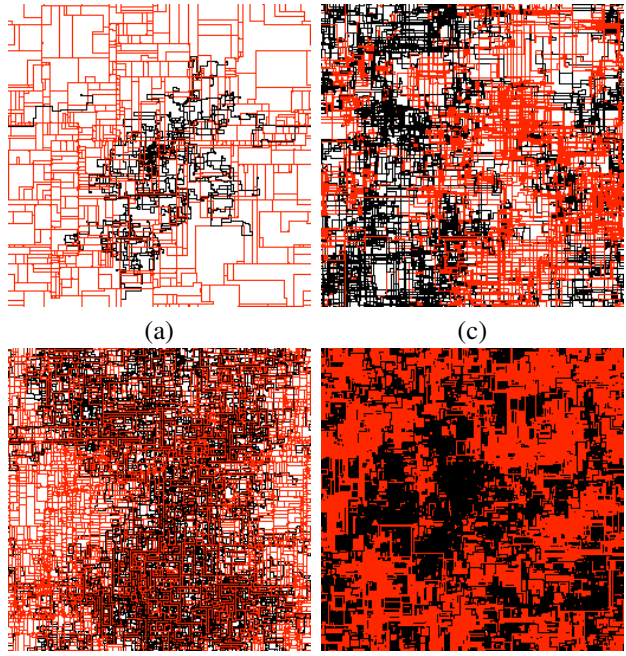


Figure 7.2. *Peintures obtenues avec deux fourmis et différents jeux de paramètres. Chaque fourmi est attirée par sa propre couleur : (a) après* 10^5 *itérations, et (b) après* 10^6 *itérations. Chaque fourmi est attirée par la couleur déposée par l'autre fourmi : (c) après* 10^5 *itérations, et (d) après* 10^6 *iterations.*

En résumé, pour une peinture de fourmis, il faut choisir les paramètres suivants :

– la taille de la toile (en pixels) ;

– le nombre de fourmis ;

– pour chaque fourmi :
  - la couleur déposée : (C_R, C_V, C_B),
  - la couleur suivie : (S_R, S_V, S_B),
  - les probabilités qui régissent le mouvement de la fourmi : (P_g, P_d, P_t),
  - le type de mouvement : D_o or D_d,
  - la probabilité de suivre la coureur suivie (si elle est trouvée) : P_s ;

– le nombre d'itérations (c'est-à-dire le nombre de pas de chaque fourmi), mais c'est optionnel.

7.2.3. *Résultats*

Une fois les paramètres fixés, la simulation des déplacements de fourmis est lancée et le spectateur regarde la peinture se construire sur un écran (la vitesse des fourmis est paramétrable, sans que cela ne modifie les motifs construits). La simulation peut être interrompue, suspendue et relancée depuis le début (les positions initiales de différentes fourmis sont conservées).

Regardons quelques exemples simples (c'est-à-dire construits avec peu de fourmis). Prenons le cas de deux fourmis : une rouge et une noire (ou bien une claire et une foncée). Hormis la couleur déposée, les deux fourmis ont les mêmes paramètres de déplacement et cherchent à suivre leur propre couleur (figure 7.2a et figure 7.2b) ou bien la couleur de l'autre fourmi (figure 7.2c et figure 7.2d). On remarque bien l'effet de la compétition dans cette deuxième configuration : la surface est remplie beaucoup plus rapidement.

D'autres exemples sont donnés dans la figure 7.3 et les paramètres utilisés pour les produire sont dans le tableau 7.1.

fourmi	couleur déposée (C_R, C_V, C_B)	couleur suivie (S_R, S_V, S_B)	probabilités de mouvement (P_g, P_g, P_t)	type de mouvement $D_{d\|o}$	probabilité de suivi P_s
Exemple 7.3(a)					
1	(192, 0, 255)	(255, 155, 3)	(0.04, 0.95, 0.01)	D_d	0.7852
2	(255, 155, 3)	(76, 68, 181)	(0.01, 0.98, 0.01)	D_d	0.8409
Exemple 7.3(b)					
1	(145, 0, 94)	(145, 0, 94)	(0.24, 0.73, 0.03)	D_d	0.99
2	(132, 0, 114)	(132, 0, 114)	(0.27, 0.68, 0.05)	D_d	0.99
Exemple 7.3(c)					
1	(5, 211, 149)	(255, 12, 0)	(0.80, 0.10, 0.10)	D_d	0.80
2	(145, 11, 149)	(255, 12, 0)	(0.80, 0.10, 0.10)	D_o	0.80
3	(255, 204, 0)	(255, 255, 0)	(0.01, 0.98, 0.01)	D_d	0.75
4	(255, 255, 0)	(255, 204, 0)	(0.01, 0.98, 0.01)	D_d	0.75
5	(255, 153, 0)	(255, 204, 0)	(0.01, 0.98, 0.01)	D_o	0.75
6	(255, 51, 0)	(255, 255, 0)	(0.01, 0.98, 0.01)	D_d	1.00
Exemple 7.3(d)					
1	(14, 108, 15)	(210, 193, 253)	(0.18, 0.40, 0.42)	D_d	0.67
2	(210, 193, 253)	(14, 108, 15)	(0.08, 0.01, 0.91)	D_d	0.81

Tableau 7.1. *Paramètres des fourmis pour les peintures de la figure 7.3*

En écartant momentanément l'aspect esthétique des résultats obtenus, on remarque que la construction collective et dynamique des peintures montre bien la compétition qui s'opère entre les couleurs déposées par les fourmis. Chaque fourmi tente de remplacer la couleur, déposée par une de ses rivales, par sa propre couleur. Nous pouvons

observer l'influence des valeurs des paramètres sur l'efficacité des fourmis dans ce combat. Prenons, par exemple, le cas de la peinture 7.3d, élaborée avec deux fourmis. On constate que la fourmi de couleur foncée a couvert une surface beaucoup plus faible (pour le même nombre d'itérations). En effet, dans les paramètres respectifs de ces deux fourmis (tableau 7.1), on constate que la fourmi de couleur claire a une stratégie plus agressive, avec une probabilité très forte de suivre l'autre couleur, si elle se présente à elle.

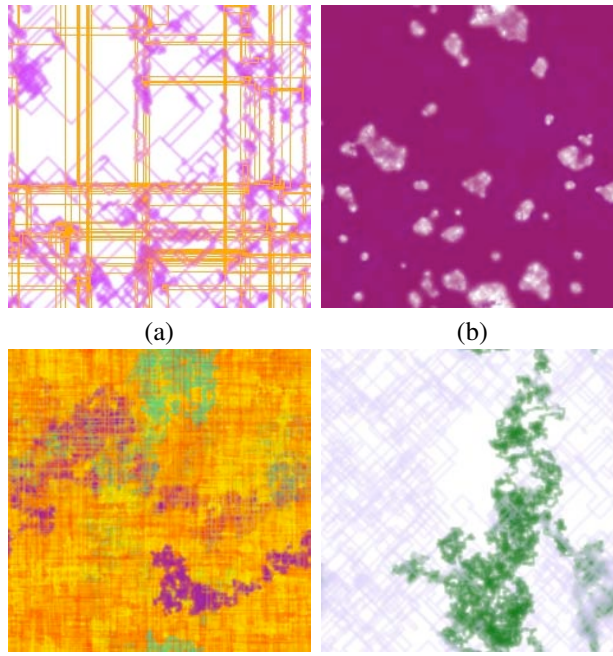


Figure 7.3. *Peintures obtenues avec différents jeux de paramètres*

Enfin, l'aspect dynamique de la construction est intéressant d'un point de vue visuel, mais ne peut pas être retranscrit dans ce chapitre !

7.3. Evolution interactive des peintures de fourmis

Comme nous l'avons présenté, une peinture de fourmis nécessite de nombreux paramètres. Nous présentons, dans cette section, un moyen d'impliquer l'utilisateur, sans lui demander de maîtriser, dans le détail, des réglages de la méthode.

7.6. Bibliographie

[BEN 01] Bentley P., Corne D., *Creative Evolutionary Systems*, Morgan Kaufmann, 2001.

[BIL 94] Biles J. A., « GenJam : A genetic algorithm for generating jazz solos », *International Computer Music Conference (ICMC'94)*, The Computer Music Association, 1994.

[BLA 04] Blackwell T., Young M., « Swarm granulator », *2nd European Workshop on Evolutionary Music and Art (EVOMUSART2004)*, Coimbra, Portugal, Springer, 2004.

[BON 99] Bonabeau E., Dorigo M., Theraulaz G., *Swarm Intelligence : From Natural to Artificial Systems*, Oxford University Press, 1999.

[CLA 08] Clair R., Monmarché N., Slimane M., « Interactions between an artificial colony of musical ants and an impaired human composer : towards accessible generative arts », *XI Generative Art Conference*, Milan, Italie, 16-18 décembre 2008.

[COL 92] Collins R., Jefferson D., « AntFarm : Towards Simulated Evolution », C. Langton, C. Taylor, J. Farmer, S. Rasmussen (dir.), *Proceedings of the workshop on Artificial Life II*, vol. 10, p. 579–601, Addison-Wesley, 1992.

[DAW 86] Dawkins R., *The Blind Watchmaker*, Longman, Harlow, 1986.

[DOR 96] Dorigo M., Maniezzo V., Colorni A., « The Ant System : Optimization by a colony of cooperating agents », *IEEE Transactions on Systems, Man, and Cybernetics Part B : Cybernetics*, vol. 26, n° 1, p. 29–41, 1996.

[GRE 05] Greenfield G., « Evolutionary Methods for Ant Colony Paintings », F. Rothlauf, J. Branke, S. Cagnoni, D. Corne, R. Drechsler, Y. Jin, P. Machado, E. Marchiori, J. Romero, G. Smith, G. Squillero (dir.), *Applications on Evolutionary Computing : EvoWorkkshops 2005 : EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, and EvoSTOC*, vol. 3449 de *Lecture Notes in Computer Science*, p. 478–487, Lausanne, Suisse, 30 mars-1 avril 2005.

[HEM 99] van Hemert J., Eiben A., « Mondriaan Art by Evolution », E. Postma, M. Gyssens (dir.), *Proceedings of the Eleventh Belgium/Netherlands Conference on Artificial Intelligence* (bnaic'99), p. 291–292, Kasteel Vaeshartelt, Maastricht, Pays-Bas, 3-4 novembre 1999.

[HOL 75] Holland J., *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor, 1975.

[MCC 96] McCormack J., « Grammar Based Music Composition in Complex Systems 96 : From Local Interactions to Global Phenomena », *Complex Systems*, p. 320–336, ISO Press, Amsterdam, 1996.

[MIC 96] Michalewicz Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, 1996.

[MIR 07] Miranda E. R., Biles J. A., *Evolutionary Computer Music*, Springer-Verlag, 2007.

[MON 07] Monmarché N., Mahnich I., Slimane M., « Artificial Art made by Artificial Ants », J. Romero, P. Machado (dir.), *The Art of Artificial Evolution : A Handbook on Evolutionary Art and Music*, Natural Computing Series, p. 227–247, Springer-Verlag, Berlin, Heidelberg, 2007.

[MOU 04] Moura L., Garcia Pereira H., *Man + Robots : Symbiotic Art*, Institut d'art contemporain, Villeurbanne, 2004.

[ROM 07] Romero J., Machado P., *The Art of Artificial Evolution : A Handbook on Evolutionary Art and Music*, Natural Computing Series, Springer-Verlag, Berlin, Heidelberg, 2007.

[RON 99] Ronald E., Sipper M., Capcarrère M., « Testing for Emergence in Artificial Life », D. Floreano, J. Nicoud, F. Mondana (dir.), *Proceedings of the Fifth European Conference on Artificial Life (ECAL)*, p. 13–20, Springer-Verlag, Heildelberg, 1999.

[ROO 01] Root-Bernstein R., « Music, Creativity and Scientific Thinking », *Leonardo*, vol. 34, n° 1, p. 63–68, 2001.

[TAK 01] Takagi H., « Interactive evolutionary computation : fusion of the capabilities of EC optimization and human evaluation », *Proceedings of the IEEE*, vol. 89, p. 1275–1296, septembre 2001.

[TZA 00] Tzafestas E. S., « Integrating drawing tools with behavioral modeling in digital painting », *Proceedings of the 1st International Workshop « Bridging the gap : Bringing together new media artists and multimedia technologists »*, Los Angeles, CA, novembre 2000.

[URB 05] Urbano P., « Playing in the Pheromone Playground : Experiences in Swarm Painting », F. Rothlauf, J. Branke, S. Cagnoni, D. Corne, R. Drechsler, Y. Jin, P. Machado, E. Marchiori, J. Romero, G. Smith, G. Squillero (dir.), *Applications on Evolutionary Computing : EvoWorkkshops 2005 : EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, and EvoSTOC*, vol. 3449 de *Lecture Notes in Computer Science*, p. 527–532, Lausanne, Suisse, Springer-Verlag, Berlin, Heidelberg, 30 mars-1 avril 2005.

[XEN 81] Xenakis I., *Musiques formelles : nouveaux principes formels de composition musicale*, Stock, Paris, 1981.