Le but de notre projet est de faire un jeu d'action en C. X-JAC est un jeu du type shoot'em up. Pour pouvoir mener à bien notre projet nous avons mis en place un environnement de travail et utilisé des outils de programmation et de rédaction pour faire nos rapports. Le projet n'est pas seulement un listing de code, mais il est aussi composé d'un ensemble d'outils et de logiciels qu'il a fallu utiliser et donc étudier.

Nous developperons dans ce cahier technique les aspects techniques et les étapes de la constitution, en passant par le paramétrage, jusqu'au l'exploitation du systéme pour lancer le jeu XjAC. Le principe de fonctionnement et les mécanismes du jeu ne seront donc pas abordés ici; il est question de l'explication de la plateforme de developpement linux Mandrake 8.00, d'Allegro, et de XjAC.

A) Présentation de l'environnement et des outils de travail

1)Les différents logiciels:

En premier lieu il fallait que notre programme fonctionne sous Linux. Pour répondre à cette contrainte nous avons tous installer Linux sur notre machine personnelle. Nous avons tous installé la même distribution Linux Mandrake 8.0, car elle est facile à installer et propose en standard le serveur graphique X appellé Xfree86 4.00.

Notre choix dans la librairie de développement c'est porté sur Allegro-4.0.0, cette version d'Allegro disponible depuis le 9 décembre 2001 sur le site (www.allegro.cc) est la dernière distribution disponible. Cette dernière version d'Allegro est une version stable qui a été tester par diffèrent développeurs avant d'être disponible sur internet, elle s'appuie fortement sur la plateforme Xfree86 4.00.

En outre, cette version d'Allegro est fournit avec de nombreux exemple et de la documentation. Pour faire la rédaction de notre rapport et de nos notices nous avons utiliser le logiciel Staroffice 5.1. Pour communiquer entre membre du projet nous avons utiliser Pine disponible depuis gevrey ou givry à l'iut et nous avons aussi mis en place des répertoires de travail sous bordeaux.

2)Comment installer Allegro 4.0.0:

Avant d'installer Allegro il faut vérifier que Xfree86 4.0.0 ou plus soit bien installer. Si la version de Xfree86 que vous possédez est antérieur a la version 4.0.0, vous ne pourrait pas gérer le buffer principal très utile fournit par Allegro. De plus, vérifiez que les composants ci-dessous sont bien présents quand vous avez installé Xfree86 :

- -Xfree 86 403.7
- -Xfree 86-100dpi-Fonts
- -Xfree 86-75dpi-Fonts

-Xfree 86-devel
-Xfree 86-libs
-Xfree 86-server
-Xfree 86-XFS

Bien, nous pouvons effectivement passer à l'installation d'Allegro

B) Installation d'Allegro 4.00

Avant toutes choses, le cd-rom contient des fichiers .log qui sont les traces d'execution, les resultats de plusieurs étapes décritent par la suite. En tout les cas, je vous conseille de rediriger vos resultats de commandes dans des fichiers .log

Au debut vous êtes en mode Super-utilisateur root comme l'indique l'invite à gauche avec le chemin local. (certaines commandes d'Allegro necessite le super utilisateur -section critique-)
Voici les étapes :

On decompacte le rpm depuis le cdrom (rpm install)
[root@localhost]rpm -i /mnt/cdrom/allegro-4.0.0-1.src.rpm
Le fichier allegro-4.0.0.tar.gz a ete genere dans /usr/src/rpm/sources
(chez moi en tout cas, sinon faire un find)
[root@localhost]tar -xvfz allegro-4.0.0.tar.gz
Voila Allegro a ete installe ! :)
[root@localhost /usr/src/rpm/sources/allegro-4.0.0]
Autorisons ces fichiers en execution, pour lancer la configuration d' Allegro 4.0.0
[root@localhost /usr/src/rpm/sources/allegro-4.0.0]chmod u+x configure fix.sh
On appel un adapteur (script shell) avec le mode unix en parametre
[root@localhost /usr/src/rpm/sources/allegro-4.0.0]./fix.sh unix
On lance confige...
[root@localhost /usr/src/rpm/sources/allegro-4.0.0]./configure
[root@localhost /usr/src/rpm/sources/allegro-4.0.0]chmod u+x allegro-config xmake.sh

On lance un make (script sh) pour gestion X11, ca peut etre assez long ...
[root@localhost /usr/src/rpm/sources/allegro-4.0.0]./xmake.sh

Enfin, on lance la configuration avec le fichier configure
dont voici les options:
--enable-static - builds a statically linked library
--disable-shared - disables the default shared libraries
--enable-dbglib - builds a debug version of the library
--enable-dbgprog - links test programs with the debug library

(Pensez a sauvegarder le resultat de cette commande dans un fichier configure.log
afin de pourvoir determiner d'ou eventuellement un probleme pourrait survenir)

[root@localhost usr/src/rpm/sources/allegro-4.0.0]# ./configure >>configure.log
On lance les makes ....
[root@localhost usr/src/rpm/sources/allegro-4.0.0]# make
[root@localhost usr/src/rpm/sources/allegro-4.0.0]# make install

Pour une aide Allegro dans le man...
[root@localhost usr/src/rpm/sources/allegro-4.0.0]# make install-man
[root@localhost usr/src/rpm/sources/allegro-4.0.0]# make depend
Finissons les makes, par le make des Documentation des methodes
(notez le changement de repertoire)
[root@localhost usr/src/rpm/sources/allegro-4.0.0/docs]# ./makedoc

Voici donc la structure de cette distribution
rappel : (mon_path_allegro=$ALLEGRO=/usr/src/RPM/SOURCES/allegro-4.0.0/)

/usr/src/RPM/SOURCES/allegro-4.0.0/include : Les HEADERS dont allegro.h et Xalleg.h
/usr/src/RPM/SOURCES/allegro-4.0.0/lib : Les LIBRARIES : les statiques .a & les dynamiques.so
/usr/src/RPM/SOURCES/allegro-4.0.0/docs : Les DOCS au format .txt & .html
/usr/src/RPM/SOURCES/allegro-4.0.0/tests : Les SOURCES DE CODE ALLEGRO dont test.c
/usr/src/RPM/SOURCES/allegro-4.0.0/examples : Les EXEMPLES : pour tester les fonctions
/usr/src/RPM/SOURCES/allegro-4.0.0/

Le resultat de la commande ci dessous indique les chemins a ajouter
dans le /root/.bashrc, alors ajoutez les exports dans le .bashrc
[root@localhost allegro-4.0.0]# ./allegro-config --env
export PATH=$PATH:/usr/local/bin
export ALLEGRO=/usr/src/RPM/SOURCES/allegro-4.0.0
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib:$ALLEGRO/lib
export LIBRARY_PATH=$LIBRARY_PATH:/usr/local/lib:$ALLEGRO/include

export C_INCLUDE_PATH=$C_INCLUDE_PATH:/usr/local/include
export CPLUS_INCLUDE_PATH=$CPLUS_INCLUDE_PATH:/usr/local/include
export OBJC_INCLUDE_PATH=$OBJC_INCLUDE_PATH:/usr/local/include
[root@localhost allegro-4.0.0]#

Testez avec un exemple (attention aux anti-quotes)
[root@localhost usr/src/rpm/sources/allegro-4.0.0/tests]gcc test.c `allegro-config --libs --cflags`

C) Paramétrage des fonctionnalitées d'Allegro 4.0.0

Lorsque l'on installe allegro sur une machine il faut le configurer pour qu'il fonctionne sous le système que l'on utilise, pour régler les différents mode il faut modifier le fichier « allegro.cfg ». Ce fichier est chargé à chaque compilation d'un programme, il contient tous les réglages dont un programme peut avoir besoin. Des réglages d'affichage et de configuration des pilotes d'affichage, des pilotes sonores, souris, ,clavier, langage, etc...

Nous ne sommes pas obliger de tous les régler, puisque des routines d'Allegro se chargent de « l'autodétection » du meilleur environnement possible. Mais en forçant certains drivers au matériels dont on dispose nous nous assurions une plateforme stable de developpement. Voici une partie su fichier « allegro.cfg » configuré pour fonctionner pour fonctionner avec notre « parc » de developpement.

Ensuite dans votre repertoire d'installation Allegro, pour pouvez ouvrir le fichier allegro.cfg. Il s'agit de la configuration et du comportement d'allegro paramétrable pour differents OS, modes graphiques, pilotes des divers phériphérique (clavier,souris,ecran..). Si jamais vous

avez des problèmes avec l'affichage sous X, vérifiez vos réglages dans ce fichier. En principe, ne pas préciser de valeurs laissée à Allegro le soin de détecter le materiel, ce qu'il fait trés bien notons-le.

Voici le fichier .log en question :

```
[system]

# Unix system drivers:
#
#  XWIN     - Xwindows
#  LNXC     - Linux console

system = XWIN

[graphics] //Description des différents mode graphique selon les OS (Operating System)

# Linux console graphics drivers:
#
#  FB           - fbcon device
#  VBAF         - VBE/AF
#  SVGA         - SVGAlib
#  VGA          - Standard VGA
#  MODX         - Mode-X
#
# X graphics drivers:
#
#  XWIN         - standard X Windows
#  XWFS         - Fullscreen X Windows
#  XDGA         - XFree86 Direct Graphics Access 1.0 (DGA)
#  XDFS         - Fullscreen DGA 1.0 mode
#  DGA2         - DGA 2.0 mode
#  DGAS         - DGA 2.0 software only mode
#

# You can also specify different drivers for a particular mode or color
# depth, eg. "gfx_card_640x480x16 = VBE3", "gfx_card_24bpp = VBE1", or
# you can provide a number of alternative drivers, for example
# "gfx_card1 = VGA", "gfx_card2 = MODX", etc.

gfx_card =
```

D) Structure d'une « distribution » XjAC

fichiers bitmap 256 couleurs requis :

chargerModif_m.bmp gameOver.bmp sauver_b.bmp

charger_b.bmp
charger_image_1.bmp
charger_image_2.bmp
charger_image_3.bmp
charger_image_4.bmp
charger_image_5.bmp
charger_image_6.bmp
decor.bmp
ennemy.bmp
fond.bmp
fond3.bmp
frame_score.bmp
gagne.bmp
logo_score.bmp
musique.bmp
musique_non.bmp
musique_oui.bmp
new_partieModif_m.bmp
new_partie_b.bmp
nom.bmp
optionsModif_m.bmp
options_b.bmp
options_menu.bmp
quitModif_m.bmp
quit_b.bmp
sauverModif_m.bmp
scoreModif_m.bmp
score_b.bmp
score_value.bmp
son.bmp
son_non.bmp
son_oui.bmp
start_xjac.bmp
start_xjac2.bmp
vaiss.bmp
vaisseau.bmp
xjac.bmp

fichier texte requis

fichier des scores	:	essai.dat

Script de compilation/lancement

fichier build	:	script de lancement des compilations
fichier compilc	:	compile du code c (gui.c)
fichier compilcpp	:	compile du code c++ (xjac.cpp)
fichier sprite.h	:	header associé (contient des signatures et du code!!)

fichiers C et C++

file.c	:	Chargement de fichier texte
gui.c	:	GUI v1.2
gui_anim.c	:	GUI beta v1
testscroll.c	:	Programme de comparaison de type de scrolling
testScroll2Ec.c	:	Test Scrolling v.1
xjac.cpp	:	Scrolling + Gestionnaire de Sprite v1.8.

"y'a plus qu'à..."

./build	Lance la compilation
./gui	Lancement module Gui (Point d'acces dans le jeu par la gui)
./xjac	Lancement module Xjac