1. NATURE

Descriptif: X-JAC est un shoot-em up avec un scrolling.

Un vaisseau jouable évolue dans une carte grâce a un scrolling. La carte est un élément statique tandis que un ennemi, par exemple, et un élément mouvant. Un tableau (ou niveau, stage, défi) est une phase dans le jeu, il se compose de ces 2 types éléments et se termine lorsque le vaisseau arrive à la ligne d'arrivée de ce tableau. Ensuite, le tableau suivant s'offre au joueur.

Le vaisseau entrant en collision avec un élément statique ou mouvant est détruit, perd une vie et le joueur doit recommencer le tableau en cours au début. Si le vaisseau est détruit et qu'il ne reste plus de vie, la partie est terminée. Si le joueur recommence une nouvelle partie c'est a partir du 1er tableau.

Buts

X-JAC fonctionnera sous Linux et sous windows (si possible).
X-JAC répondra dans chaque étape du développement à la philosophie OpenSource
Le jeu doit être fluide et rapide.

Limites

Les graphismes du jeu sont en 2D (tableaux, éléments,menus)
La résolution minimale est 320x200 en 256 couleurs en mode fenêtré
ou en plein écran

2. DÉFINITIONS

X-JAC:	X-window Jeu d'Arcade en C.
OpenGL:	Regroupement de librairies graphiques dont le code source est disponible selon le critère Open Source.
OpenSource:	*Critère* rendant disponible légalement le code du programme
Notice:	Fichier-manuel expliquant un accus ou un procédé d'utilisation d'une ressource informatique.
Partie:	*U*ne session de jeu *X-JAC*
Tableau:	*U*ne phase dans une session de jeu. Cette phase a un départ et une arrivée
X-window:	*U*ne interface graphique en standard sous Linux, sert de plate-forme pour les

gestionnaires de fenêtres.

KDE, Debian:	Différents gestionnaires de fenêtres utilisant X-window
GUI:	Graphic User Interface. Interface graphique pour l'utilisateur final
Utilisateur final:	Le joueur jouant à X-JAC.
Bitmap :	Carte de bit, tableau a double entrée, structure de donnée de base qu'offre Allegro
Écran physique :	Est représenté par le bitmap vidéo enfichable directement sur l'écran (BITMAP * screen)
Écran logique :	Il représente le bitmap stocké en mémoire dans le but d'être affiché (memory_bitmap ou video bitmap).C'est un bitmap intermédiaire ou buffer, finalement il sera « collé » sur screen
Couche :	Implémentation en tableau d'un ensemble contenant tout les éléments de même nature.
Tuile ou tile :	*Élément* graphique élémentaire de plus haut niveau et plus spécifique que les BITMAP d'Allegro.

3. CONTRAINTES GÉNÉRALES

Recherche de libraire graphique OpenGL codée en C/C++ avec les ressources informatiques de l'iut.
Programmation en C/C++. Avec une nette préférence préalable pour le C car il est plus bas niveau et que nous en général avons plus travailler dessus.
Développement : Linux Mandrake 8.00 KDE (O.S personnels). Nous sommes tous seulement débutant ou presque dans la gestion du système d'exploitation Linux.
Bus d'information : Linux Debian (Bordeaux). Informations sur le projet consultable seulement à l'iut.

4. RESSOURCES A NOTRE DISPOSITION

Internet (à l'IUT)
- Salle de discussion, site hébergeant des projets de librairies en cours (www.sourceforge.net)
- Site conseil sur des librairies déjà testées (chat, faq, newsgroup)
- Tutoriels et manuels au format HTML (pages internet)
- Sites hébergeant des bibliothèques d'images

Utilitaires: StarOffice 5.1 pour les textes. KDE Pixmap2Bitmap. KDE Paint, The Gimp, ScreenCapture
Livres de programmation : GTK, Applications portables en C/C++, Linux
Stockage : Lecteur Zip mis à notre disposition par M Dabancourt, Disquette 1.44 Mo

5. TACHES

1ere: Recherche de l'existant du 22/11/2001 au 05/01/2002

La recherche d'une bibliothèque graphique 2D fonctionnant au moins sous Linux est importante, elle conditionne directement le choix d'une parmi les autres. La librairie choisie sera utilisée pendant toute la période de développement.

BUTS

Chercher une librairie graphique et obtenir une présentation générale, les contraintes de fonctionnement, les avantages/inconvénients généraux, etc...
Coder grâce a cette librairie une fenêtre graphique avec quelques éléments pour se rendre compte de la facilité d'écriture des méthodes, du niveau de compatibilité avec la plate-forme proposée, et de divers inconvénients techniques,etc....
Préparer une synthèse de ces travaux claire et facile à utiliser, puis la présenter au groupe de projet.
Après cette présentation et d'éventuelles corrections, rendre disponible cette nouvelle ressources sur bordeaux.
Obtenir des avis professionnels sur différentes bibliothèques graphiques pour des jeux dans le style X-JAC.

CONTRAINTES

La libraire étudiée doit être assez répandue pour trouver facilement des sources assez riches et variées pour la prendre en main rapidement.
Bien sur, les contraintes générales et limites du projet sont respectées. Une recherche sur une libraire qui n'y répondrait pas doit être abandonnée après concertation.

RÉALISATIONS

Synthèse de l'existant
Notices d'installation, de réglages et paramétrages des librairies
Documentation pour l'utilisation des librairies.
Études des programmes codés avec les différentes librairies.

RÉPARTITION

Commun	:	Installer Linux sur pc personnel, Installer la libraire sur Bordeaux et chez soi Rechercher l'information brut (les sources, la documentation, et des programmes d'exemples de la librairie) Conception du cahier des charges

Individuel

Jean-Claude	:	SDL
Arnaud	:	GTK
Thomas	:	Allegro
Gwenael	:	EZWGL
Mustafa	:	Recherche de source de jeu
Mathieu	:	OpenGUI

Conseil

Gwenael et Mustafa	:	Recherche d'avis professionnels sur les librairies choisies et d'autres
Mathieu	:	Recherche librairies OpenGL et des projets de librairies OpenGL en cours

Gestion du projet

Mathieu	:	Réalisation du cahier des charges

MÉTHODOLOGIE

CERNER LA LIBRAIRE:

Déterminer la nature et la source de la librairie.
Déterminer son champ d'activité.
Vérifier que la libraire respecte les contraintes.
Lister les composantes requises pour la librairie. (.h .so .a)
Lister les composantes mise à disposition grâce à cette librairie. (.h .so .a)

SAVOIR UTILISER LA LIBRAIRE:

Étudier les définitions et le code de ces composantes.
Rechercher de la documentation sur l'utilisation de la librairie.
Noter les réglages et les paramétrages nécessaire à son installation et sa manipulation
Noter les avantages/inconvénients, les problèmes/solutions.
Revenir sur ‘Cerner la librairie’ pour «s'auto-valider»

TESTER LA LIBRAIRIE:

Faire plusieurs tests de base. (Une fenêtre. Des objets statiques dessinés à l'écran)
Rechercher des sources de jeux basé sur cette librairie.
Tester le jeu en analysant les codes sources.
Identifier les composantes logicielles utiles à réutiliser.
Tester ces composantes à part.
Noter les remarques appropriées.

Chaque librairie sera ainsi décortiquée séparément. Un compte-rendu regroupant toutes ses tâches sera fait lors d'une réunion.

2ere: Conception principale et codage du 18/01/2001 au 17/02/2002

La conception principale et son codage a pour but de définir les bases du système d'informations spécifique à la construction du jeu vidéo. Le module de Scrolling issu de cette phase de develeppemment devra être presque terminé par la phase suivante.

BUTS

Concevoir des structures de données optimisées pour la fluidité dans le but de contenir et manipuler les données graphiques pour le scrolling.
Coder les méthodes de scrolling et les intégrer a un premier prototype autonome, ces méthodes doivent être composées des composantes les plus réutilisable d'Allegro (BITMAP). De plus, les méthodes qui agissent dessus doivent être peu variées afin de nous permettre toujours de déterminer si le code fonctionnerait sur d'autre plateforme .
Maintenir la simplicité & l'adaptabilité du code afin de pouvoir y intégrer rapidement les codes issus de la conception secondaire
En déduire les spécifications de la conception secondaire

CONTRAINTES

Les tests d'Allegro 4.0.0. n'ont pas été suffisemment abouti.
Spécification de l'environnement de travail ne sont encore pas concrètement disponible.
Difficultés à obtenir les tâches dans les délais impartis.
Les tâches qui sont définis en groupe, une fois affectés à des sous-groupes de travail font rarement l'objet d'une analyse/conception mais plus souvent d'une écriture de code systématique.
Les tâches sont trop souvent effectuées en dehors de leur attribution ce qui produit des codes inexploitables.

RÉALISATIONS / MÉTHODOLOGIES

Prototype du scrolling
Spécification détaillées de la conception
Spécification de l'environnement de travail
Réaliser des diagrammes.
Réaliser des tests de vitesse d'exécution, prévoir des ajustements
Réaliser des maquettes d'écrans.

RÉPARTITION

Commun
Individuel

Jean-Claude & Gwenael	:	Datafile, Sprites & Diagramme UML
Arnaud	:	Scrolling, Structure de données
Thomas	:	Sprites,
Mustafa	:	GUI
Mathieu	:	Structure de données, tuiles, Scrolling

Conseil
Jean-Claude & Gwenael : Méthodologie de travail
Arnaud, Mathieu : Gestion ressources mémoire

3eme: Conception secondaire et codage du 27/02/2001 au 18/03/2002

La conception secondaire et son codage doit permettre de greffer les fonctionnalités de Gestion des Sprites et de Collision au module principal de Scrolling.

BUTS

Concevoir les méthodes de gestion des Sprites et de calcule de leur coordonnée
Optimiser le temps pris par la « boucle d'écoute des Sprites »
Maintenir la simplicité & l'adaptabilité du code
Maintenir la fluidité du scrolling

CONTRAINTES

Difficultés pour travailler en groupe a des horaires fixe.
Difficultés à obtenir les tâches dans les délais impartis.
Les tâches qui sont définis en groupe, une fois affectés à des sous-groupes de travail font rarement l'objet d'une analyse/conception mais plus souvent d'une écriture de code systématique.
Les tâches sont trop souvent effectuées en dehors de leur attribution ce qui produit des codes inexploitables.

RÉALISATIONS / MÉTHODOLOGIES

Réalisation module des Sprites
Réalisations méthodes de calcul et de raffraichissement des coordonnées des sprites

RÉPARTITION

Individuel

Jean-Claude & Gwenael : Datafile,diagramme UML.
Arnaud : Scrolling, Gestion des sprites
Thomas : Événementiel : séquence des actions (Collision,...)

Mustafa : GUI
Mathieu : Structure de données, Gestion des sprites

6. SYNTHESE DES CHOIX

Cette synthése présente succintement et par ordre chronologique les princiales directions choisies par le groupe de projet et validées par le chef de projet. Elle explique et justifie rapidement les décisions mais n'en n'explique pas tout ces aspects qui sont explicitées dans le rapport de projet.

Le premier choix fut celui d'attendre la fin de la recherche de l'existant pour définir une libraire graphique de travail : Allegro 4.0.0 fut retenue.

Ensuite, nous avons opté pour d'un stockage mémoire séparé :

Les états logiques subissent plus de manipulation de lecture/écriture que les états graphiques. Par ailleurs, ces premiers peuvent avoir un « lien de parenté » (héritage) étroit les liant, ainsi pouvoir appliquer sans distinction des méthodes au structures filles comme à la structure mère est attrayant. Il semble donc qu'une implémentation avec une structure de données en C++ soit préférable.

Les états graphiques s'il subissent au plus autant de modification que les états logiques, il n'en reste pas moins qu'ils sont l'objet d'un traitement de copier/coller (blit() qui pour être optimal doit être sélectif) bien plus lourd en temps d'exécution. De plus, la charge mémoire des images graphiques auquels ces états sont liés doit être continuellement surveillée. Une implementation en C pur et en Allegro pour ces images (structure en couche) est préférable.

Le troisième choix concerne le scrolling qui défile grâce a deux écrans tampon en mémoire vive. Ce système a été adopté (après nos tests) de pour son aspect modulable et sa rapidité. En effet, le premier tampon écran i et le deuxiéme écran i+1 est compatible avec le module de chargement/sauvegarde de niveau. Celui-ci viendrait s'exécuter juste aprés la fin de l'écran i pour charger l'écran i+2 (qui viendrait remplacer alors l'écran i), et juste aprés l'écran i+1 pour charger l'écran i+3 (qui viendrait remplacer alors l'écran i+1).

Nous avons décidé dans un quatriéme temps, d'implémenter une commande de pilotage qui gére l'inertie du vaisseau du joueur. Ainsi nous orientons le jeu vers un jeu action où le joueur doit piloter son vaisseau vers la fin du niveau. Cette décision a été prise car nous savions que le temps imparti pour le projet ne nous laisserai pas assez de temps pour implémenter des ennemis qui tireraient sur le joueur (Gestion des trajectoire missiles et IA des ennemis). Le jeu semble s'orienter vers un jeu de course où les autres sprites dits jusqu'à maintenant « ennemi » deviendraient plutôt des « conccurents ».