



Projet Hexxagon



UE LIF7 – Semestre printemps 2012

Mickaël Rivollet – Thibault Lazert



Sommaire

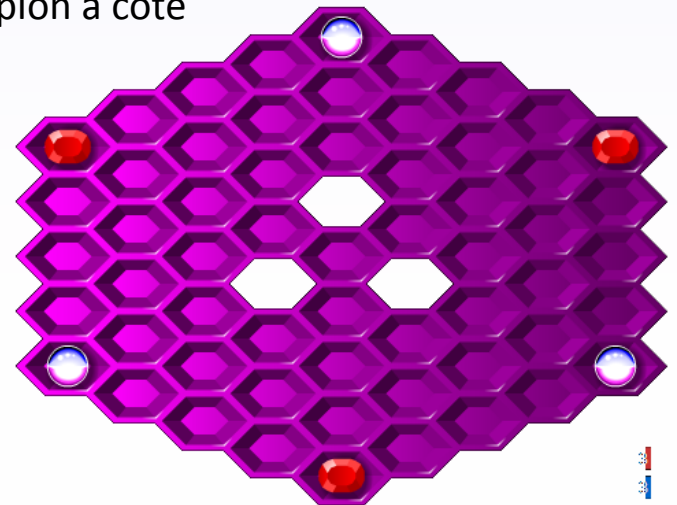
- Objectifs du projet et principe du jeu
- Fonctionnalités développées et tâches
- Diagramme des modules
- Détails concernant les modules
- Conclusion



Objectifs et principe du jeu

- Nos objectifs étaient :
 - Programmer en C un jeu connu dont nous connaissions le principe
 - Utiliser des bibliothèques pour un affichage 2D à l'écran et non dans la console
 - Travailler à deux sur un projet en utilisant SVN
- Principe du jeu:
 - Dupliquer (case adjacente) ou déplacer (case séparée par une autre) un de ses pions
 - Obtenir les pions adverses lorsque l'on place son pion à côté
 - Avoir plus de pions que l'adversaire pour gagner

ci-contre : illustration du jeu dont nous nous sommes inspirés.





Fonctionnalités et tâches

- Les possibilités de l'utilisateur :
 - Choix du plateau parmi quatre
 - Choix du mode de jeu
 - Contre un autre joueur
 - Contre l'ordinateur (différents niveaux)
 - Fonction de sauvegarde de la partie en cours
 - Fonction d'aide pendant le jeu
- Répartition des tâches au sein du groupe :
 - Répartition équitable au fur et à mesure
 - Travail en parallèle grâce à SVN (utilisation de GoogleCode)
 - Travail sur le même fichier
 - Historique des mises à jour
 - Fichier « TODO » avec la liste des prochaines tâches

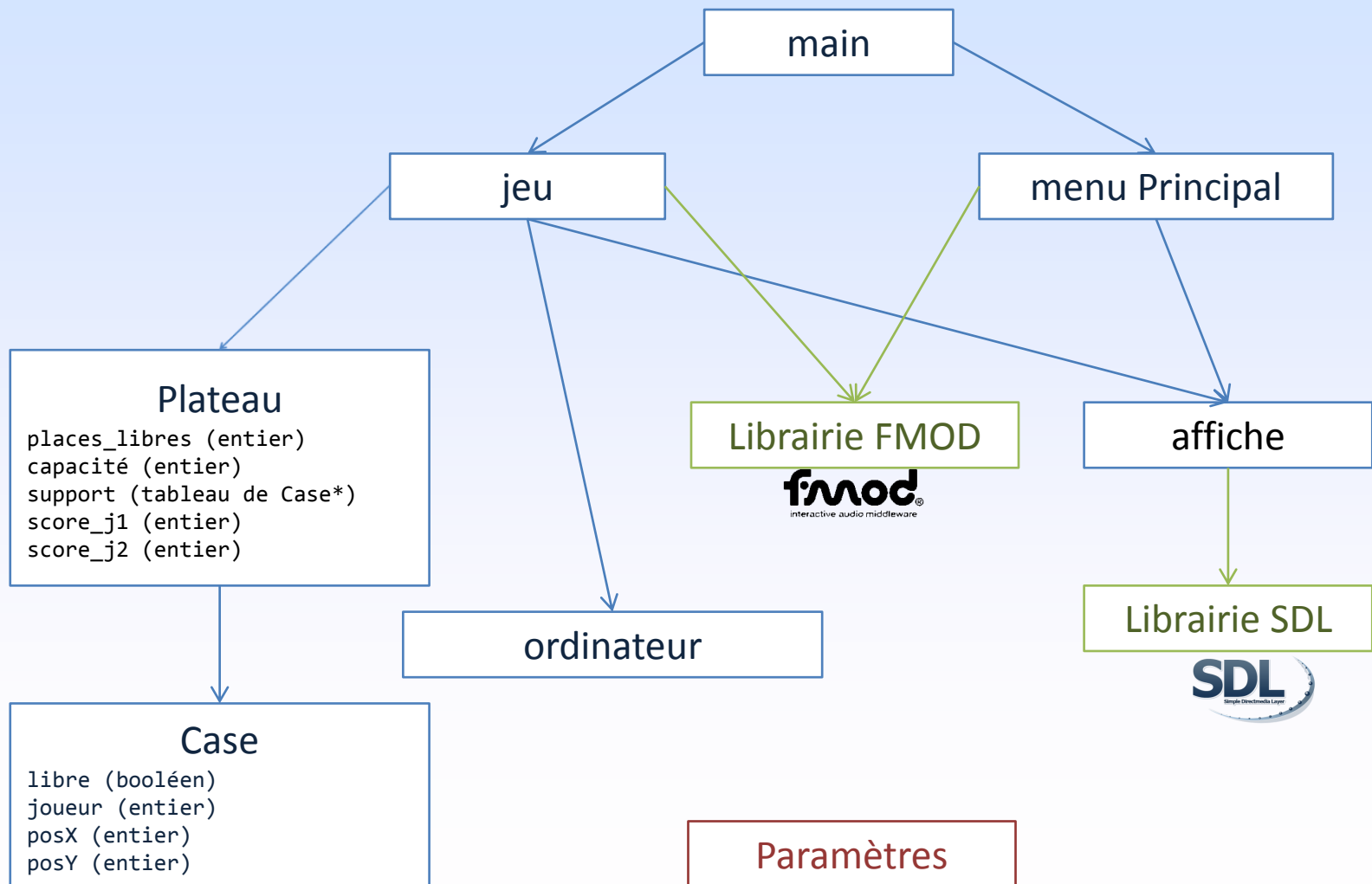


Diagramme des modules du jeu Hexxagon



Main et menu-principal

Main :

```
int      main (int argc, char *argv[])  
int      regression (void)
```

- Appel du jeu avec le paramètre « -r » pour les tests de régression.
- Le main gère les échanges entre le menu principal et le jeu.

Menu principal :

```
int      menuPrincipal (int *contreordinateur, int *niveauordinateur, int *plateau)
```

- Cette fenêtre contient un descriptif du jeu et de ses fonctionnalités.
- C'est dans cette fenêtre que l'utilisateur peut choisir le plateau et le mode de jeu.
- Il est possible de reprendre une partie enregistrée depuis ce menu.



Aperçu du menu principal





Jeu

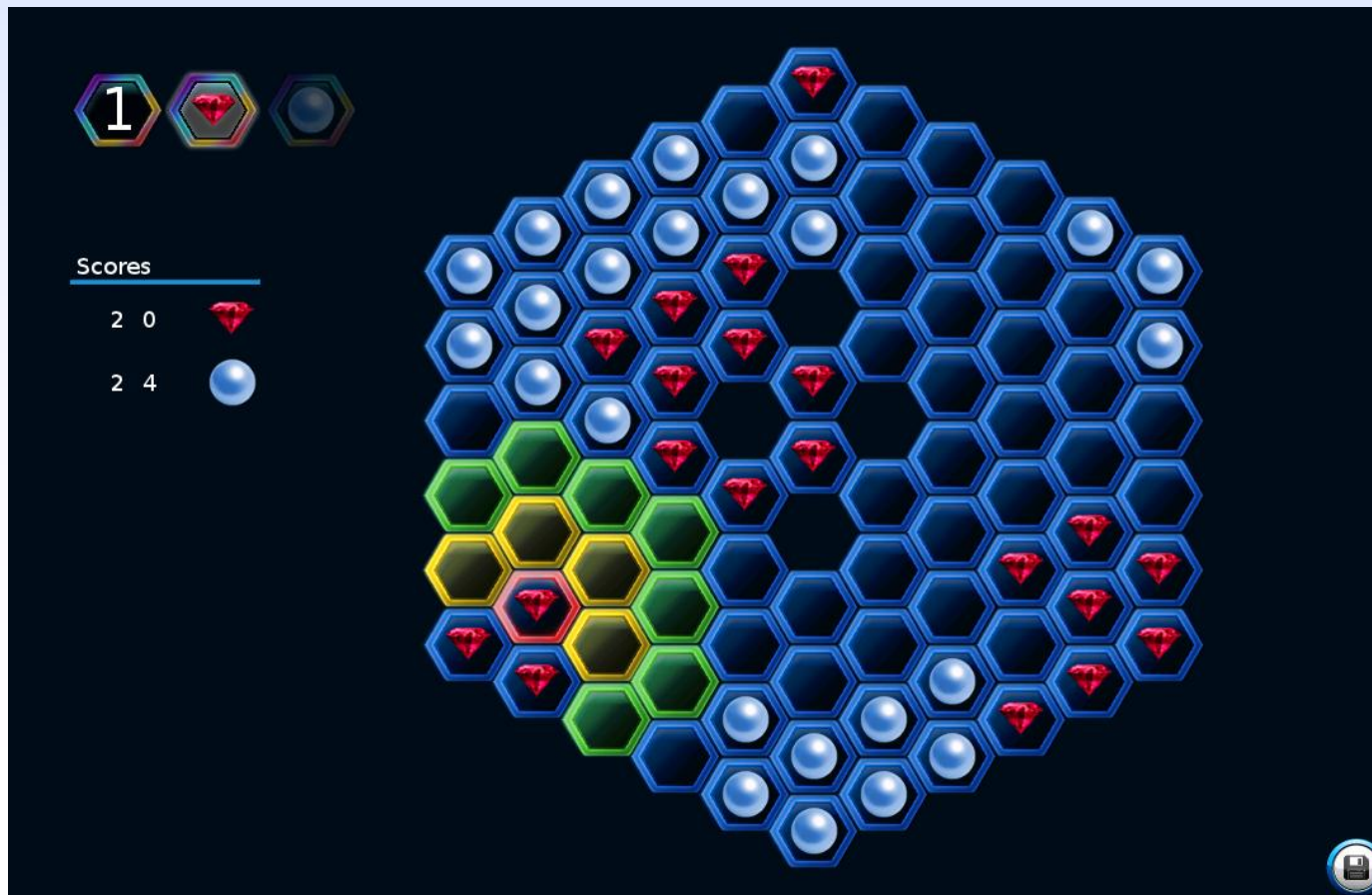
```
int Jouer (const int contreordinateur, const int niveauordinateur, const int plateau)
```

Jeu :

- Fenêtre de jeu dans laquelle se déroule la partie.
- Le bouton pour enregistrer la partie est en bas de cette fenêtre.
- Dans cette fenêtre, on voit le plateau complet ainsi que les scores des joueurs et le pion de celui qui a la main.



Aperçu du jeu





Plateau

```
void    plateauInit (Plateau *p, int capacite)
int     plateauSourisDansCase (int sx, int sy, const Case *c, int dim)
Case *  plateauCaseSurvollee (int sx, int sy, const Plateau *p, int dim)
void    plateauLireFichier (Plateau *p, const char filename[], int *joueur)
void    plateauEcrireFichier (Plateau *p, int qui_joue, int contreordi, int niveauordi)
int     plateauNbPossibilites (const Plateau *p, Case *c)
int     plateauTestCaseProche (int x, int y)
void    plateauVolerPions (Plateau *p, Case *c, int joueur)
int     plateauNbPionsPerdu (Plateau *p, Case *c, int joueur)
int     plateauPeutJouer (const Plateau *p, int j)
void    plateauTestament (Plateau *p)
void    plateauTestRegression (void)
```

Module Plateau :

- Module qui contient le plateau de jeu et notamment : les scores, la liste des cases ainsi que la capacité et le nombre de places restantes.
- C'est dans ce module que l'on lit et enregistre les fichiers des plateaux.



Case

```
Case *    caseInit (void)
int       caseTestCoordonnees (Case *c, int sx, int sy)
void      caseTestament (Case *c)
void      caseTestRegression (void)
```

Module Case :

- Contient les informations sur une case :
 - Coordonnées
 - Joueur présent
- Le système de coordonnées est indépendant de l’affichage : fonctions de conversion nécessaires.



Ordinateur

```
void      ordinateurJouer (Plateau *p, int joueur, int niv)
```

Principe :

- Cette fonction calcule le prochain mouvement de l'ordinateur.

Fonctionnalités :

- Détection des cases à défendre ou à attaquer en priorité
- Calcul du nombre de pions adverses à voler
- Différents tests effectués :
 - Calcul de la meilleure place pour l'ordinateur
 - Calcul des risques au pro rata du nombre de pièces volées



Affiche (1/2)



Fonctions d'affichage :

- Ces fonctions utilisent la SDL.

Modularité du code :

- Il serait possible de programmer ce module en utilisant une autre librairie sans changer les autres modules.
- Redéfinition des types et des constantes SDL.

```
Rectangle xy2rect (int x, int y)
int afficheInit ()
void afficheScores (int s1, int s2, Image *chiffres[], Image *haut, Image *pion_j1, Image
                  *pion_j2, Image *ecran)
void afficheNiveauOrdinateur (int x, int y, int niv, Image *chiffre[], Image *ecran)
void afficheQuiJoue (int j, Image *ecran)
void afficheFinJeu (int s1, int s2, int ordi, Image *ecran)
void afficheImage (int x, int y, Image *image, Image *ecran)
void afficheImageRect (Rectangle rect, Image *image, Image *ecran)
void afficheJeu (const Plateau *p, Image *image_case, Image *pion_j1, Image *pion_j2,
                Image *ecran)
void afficheVerifChargement (Image *img)
void afficheCaseJeu (const Case *c, Image *image, Image *ecran)
```



Affiche (2/2)



```
void    afficheCasesAutour (const Plateau *p, const Case *c, Image *img_dupliquer, Image
        *img_deplacer, Image *ecran)
void    afficheFree (Image *img)
void    afficheQuit ()
void    afficheVideEcran (Image *ecran)
Image *  afficheChargeImage (char fichier[])
int      afficheEvenements (Evenements *ev)
int      afficheCoordonneeSouris (Evenements *ev, char coord)
void     afficheMiseAJour (Image *ecran)
Image *  afficheSetEcran ()
void     afficheErreur ()
void     afficheSetTitre (char titre1[], char titre2[])
int      sourisDansRectangle (int x, int y, Rectangle rectangle)
```



Paramètres

- Le fichier contient les variables de préprocesseur avec les valeurs des paramètres.
- Les paramètres correspondent aux chemins vers les fichiers, ou aux options pour la compilation (sons, commentaires).
- Le fichier contient les variables fixés utilisés pour l’affichage.



Difficultés rencontrées

- Placement et alignement des hexagones sur l'écran
- Localisation de cases proches
 - Adjacentes
 - Distantes d'une case
- Animations des pions
- Programmation et améliorations de la fonction ordinateur (IA)
 - Chercher la meilleure position sur la meilleurs position
 - Remplir en priorité les cases en danger ou qui peuvent rapporter beaucoup
 - Calculer avec un coup d'avance



Conclusion

- Ce projet nous a permis de ...
 - ... découvrir et d'utiliser des librairies
 - SDL pour l'affichage à l'écran
 - FMOD pour les sons
 - ... travailler à plusieurs sur un projet en utilisant un logiciel de gestion de versions (SVN avec GoogleCode)
 - ... mettre concrètement en application nous connaissances de LIF5 sur les modules
 - ... développer une IA pour imiter la stratégie d'un joueur
- Nous avons apprécié travailler sur ce projet
- Notre jeu a été testé par différentes personnes



Merci de votre attention