Projet logiciel en langage C
L2 informatique

1 Sujets : programmation

Vous devez choisir un sujet parmi trois possibles :

- Le jeu HEX,
- Un des sujets FabLab : communication ou guidage de non-voyants.

Chaque sujet présente un travail de programmation à effectuer. En plus de ce travail, le projet intègre une étude préliminaire de type génie logiciel et les documents correspondants, et un cadre de gestion et suivi de projet. Ce cadre du projet est le même pour tous les sujets. **Les sujets sont détaillés en annexes**.

2 Gestion de projet (pour tous les sujets)

2.1 Etude préliminaire, documents et suivi de projet

Vous devez fournir au client (votre tuteur) certains documents classiques dans le développement de projet :

- document de spécification : ce qui va être programmé et qui doit être validé par le client,
- dossier de conception : comment vous allez le programmer,
- fiche préliminaire et fiches d'avancement de projet : pour l'équipe et le chef de projet (votre tuteur va aussi partiellement jouer ce rôle),
- fiches PEC : bilan du projet pour vous,
- validation (recette) : présentation du programme au client, réponses à ses questions,
- présentation orale : bilan du projet pour vous et le public constitué de votre tuteur et de vos pairs.

2.2 Gestionnaire de version (git)

Git est un gestionnaire de version qui simplifie grandement le travail collaboratif. Il est **obligatoire** de l'utiliser pour ce projet (**moins un point sur la note finale si vous faîtes l'impasse**). Bien que Git soit intéressant à utiliser localement pour le développement, il prend tout son sens si on l'utilise avec un gestionnaire web comme Github ou Bitbucklet. Ce travail en plus devrait se révéler rentable et vous permettre de mieux maitriser le projet.

Git et Github ou Bitbucklet
Il existe des cours en ligne comme :
https://openclassrooms.com/courses/gerer-son-code-avec-git-et-github (basé sur Github).

https://training.github.com/kit/downloads/fr/github-git-cheat-sheet.pdf (aide mémoire)
http://git-scm.com/book/en/v2 (un guide complet).
https://bitbucket.org/cumulomimbus/publiccodeprojectl2 (repository du code simple.c)

Je vous proposerai en plus, 1 à 2h de cours GIT en amphi.

3 Evaluation

Le programme que vous allez rendre sera classé dans l'une des catégories détaillées ci-dessus et sera noté par rapport à la note maximale associée. Dans les 3 versions vous devez utiliser divers modules (.h et .c) que vous compilerez séparément avec un fichier *makefile* adapté (utilitaire *make*).

3.1 Programmation (10 points)

Selon la version choisie, vous êtes notés sur :

- version 1 : 6 points
- version 2 : 8 points
- version 3 : 10 points
- jusqu'à 1 point pour l'UFC

Les critères de notation sont :

- efficacité du code (efficacité des algorithmes),
- forme du code (modularité, lisibilité, commentaires bien choisis),
- présence d'en-têtes de fonctions et de modules informatifs, conventions de présentation homogènes...),
- présence de fichiers de tests unitaires et de fichiers de tests fonctionnels.

Cette notation sera faite lors de la recette et de l'étude ultérieure du code.

3.2 Documents (10 points)

Les 10 points restant sont relatifs à la partie gestion de projet :

- document de spécification (2 points)
- dossier de conception (2 points)
- fiche préliminaire et fiches d'avancement de projet (3 points pour l'ensemble des fiches)
- fiches PEC (1 point)
- validation et présentation orale (2 points)

Les critères de notation sont :

- pertinence des écrits,
- forme de l'écrit (lisible, compréhensible, informatif),

- d'une manière générale : utilité de l'écrit. S'il ne contient rien d'intéressant pourquoi lui mettre des points ?

Vous serez donc potentiellement notés sur 20 points. Vous devez toutefois être conscients que les points de 16 à 20 seront plus durs à gagner.

4 Règles de codage minimales

Au niveau des modules : Chaque module de l'application débutera par une en-tête. Vous définirez un modèle d'en-tête dans lequel devront figurer les informations que vous jugerez nécessaires pour une bonne compréhension du rôle de ce module.

Exemple : le ou les noms des auteurs, les dépendances,...

Au niveau des sous-programmes : Chacun sera commenté par les spécifications d'entrée et de sortie.

Exemple : liste des paramètres, rôle et type (entrée, sortie) de chaque paramètre. Les commentaires figurant dans le corps de la fonction devront être pertinents et placés en fonction des différentes étapes de l'algorithme associé.

Choisissez des règles de nommage des variables en fonction du statut de celles-ci (locales à une fonction, à un module, externes).

Remarque : utiliser *git* facilite grandement ce type de gestion et peut alléger les commentaires en omettant des informations comme : la date de création du fichier, la date et le pourquoi de la dernière modification, qui sont, a priori, gérées par Git et l'OS.

5 Les fichiers de tests

1. Tests unitaires : Un fichier de test pour chaque fichier *test_u_module.c* contenant un main appelant les diverses fonctions du module à tester.
2. Tests d'intégration au niveau d'un module *test_nom_du_module.c* permettant de tester les interactions entre plusieurs sous programmes ou fichiers, du module.
3. Tests fonctionnels : Un fichier test *test_fonctionnel.c* qui permet de tester le logiciel intégré avec une jeu de tests ou des grilles prédéfinies.
4. On pourra placer chacun de ces tests dans un sous-répertoire du répertoire courant et créer un Makefile associé à chaque test.

Remarque : l'intérêt de garder les tests est de pouvoir faire des tests régressifs en cas de changement majeur.

6 Organisation

- Le travail sera réalisé en groupes de 2 ou 3 personnes. Le faire seul est risqué (beaucoup de travail pour une personne), de même que le faire à quatre (coordination plus compliquée).
- Un encadrant sera affecté à chaque groupe.
- Le projet est constitué d'un programme informatique, de documents techniques et de fiches de suivi de projet à rendre à intervalles réguliers.
- Le calendrier est sur Moodle

7 Annexes

7.1 Sujet 1 : le jeu HEX (Vincent.Dugat@irit.fr)

7.1.1 Présentation du jeu (https://fr.wikipedia.org/wiki/Hex)

Le jeu de **Hex** est un jeu de société combinatoire abstrait pour deux joueurs. Il se joue sur un tablier en forme de losange dont les cases sont hexagonales. Toutes les dimensions du côté du losange sont possibles, la plus traditionnelle est celle composée par 11 hexagones, mais on trouve aussi les valeurs 13 ou 19. L'un des inventeurs du jeu, John Nash, préconise un tablier de taille 14 × 14. Ce jeu possède des similarités avec le go.

Ce jeu, inventé par des mathématiciens fait uniquement appel à la logique, à l'image du go ou des échecs. Son étude est source d'inspiration, non seulement en théorie des jeux, mais aussi pour d'autres branches des mathématiques comme la topologie ou la géométrie algébrique.

Si l'on sait qu'il existe une stratégie gagnante pour le premier joueur, cette stratégie est inconnue si le tablier n'est pas de petite taille (de côté strictement plus petit que 9). La recherche de stratégies efficaces, à défaut d'une stratégie gagnante, est l'objet d'études en intelligence artificielle.

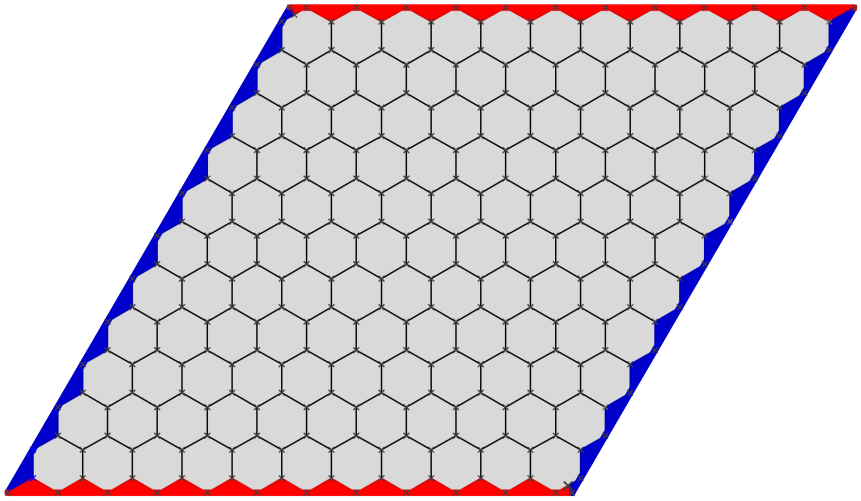


Fig. 1: plateau de jeu

7.1.2 But du jeu

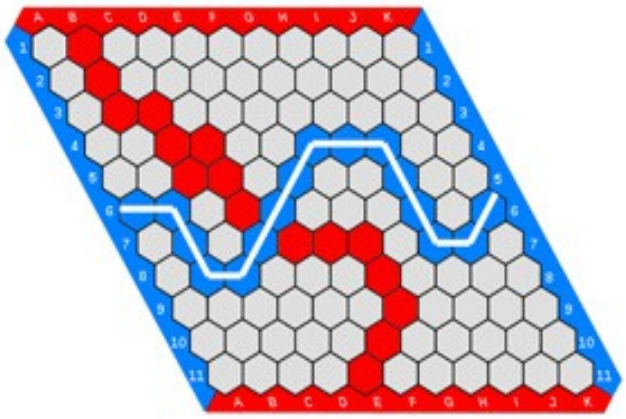


Fig. 2: Les bleus gagnent

Le but du jeu est, pour chaque joueur, de relier les deux bords opposés du plateau de jeu (rouge ou bleu) avec un chemin continu de pions.

7.1.3 Les coups

Chaque joueur joue à tour de rôle en posant un pion sur une case libre n'importe où sur le plateau de jeu. Le premier qui crée un chemin contigu de pions gagne la partie.

7.1.4 Ressources pour le jeu

Youtube : Micmaths, la chaîne de Michaël Launay :
https://www.youtube.com/playlist?list=PLNefH6S6myiMPc5-9XAeUPfxe21Hxso-s

Articles et sites :
Jeu en ligne :
http://www.novelgames.com/fr/hex/
http://fr.boardgamearena.com/#!join (nécessite une inscription)

Sites :
http://images.math.cnrs.fr/Le-jeu-de-Hex.html
http://www.swarthmore.edu/NatSci/math_stat/webspot/Campbell,Garikai/Hex/

7.2 Travail à faire (programmation)

Le travail à faire comprend :

- une interface graphique basée sur la librairie SDL (Simple Directmedia Layer). Plutôt que la dernière version de la SDL (2.0.3), nous utiliserons la 1.2 car il y a plus de documentation et d'exemples en C. Vous pouvez malgré tout utiliser la 2.0.3 si vous le souhaitez.
- le programme de jeu proprement dit.

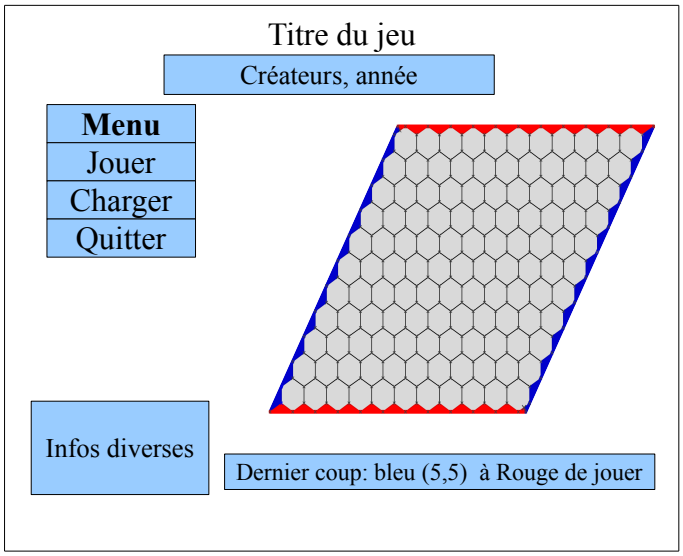
Attention toutefois, la partie « interface graphique » ne doit pas représenter plus d'un tiers de votre temps de programmation, le reste étant consacré à l'algorithmique et aux tests.
(en clair : pas de super interface sans rien derrière. L'interface n'est qu'un moyen d'interaction avec la programme.)

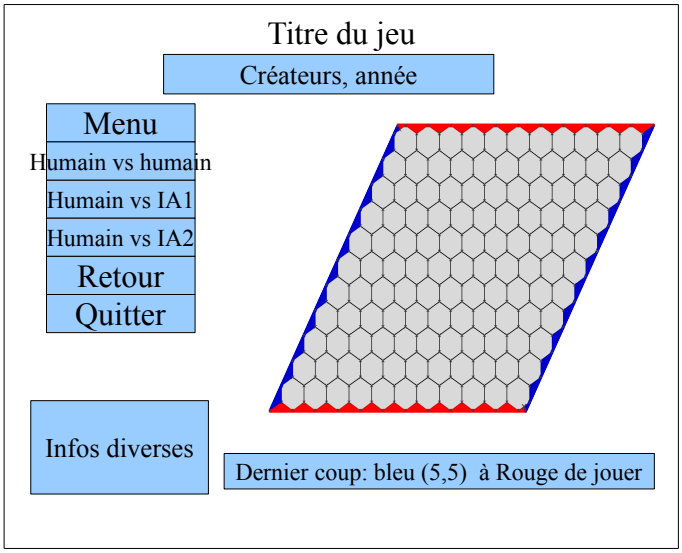
Le projet est divisé en trois niveaux.

7.2.1 Version 1 : take it easy! Humain contre humain (6 points en jeu)

1. *Développer une interface utilisateur graphique avec SDL pour piloter le reste du programme (cette version de l'interface sera dénommée SDLv1 dans la suite).*
   1. *Afficher la plateau de jeu (pour simplifier on affiche une image[1])*
   2. *Afficher l'ensemble des fonctionnalités sous forme de « menu » (du texte dans des rectangles)*
   3. *Afficher des informations : tour de jeu, dernier coup joué, nombre de coups, etc.*
   4. *Gérer les évènements clics sur le menu et le plateau*
   5. *Lancer les fonctionnalités associées*
   6. *Mettre à jour l'interface : menus, plateau de jeu avec les nouvelles cases jouées en couleur (on va plutôt utiliser une image de pion coloré)*

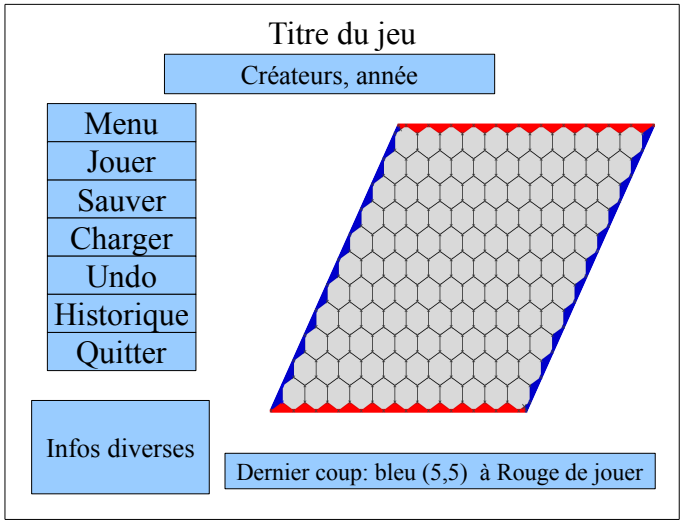
2. *Fonctionnalités du programme :*
   1. *Initialiser le plateau pour le jeu (une seule taille : 11x11).*
   2. *Gérer le jeu humain contre humain :*
      1. *choix du joueur qui commence.*
      2. *gérer les tours de jeu de chaque joueur.*
      3. *vérifier la légalité des coups des joueurs.*
      4. *détection d'un vainqueur éventuel*
   3. *Gérer une historique, la sauvegarde, la restauration, l'annulation du dernier coup, l'abandon de la partie en cours, lancer une nouvelle partie*
   4. *Quitter le jeu avec ou sans sauvegarde*





1 La taille du plateau est donc fixe (11x11). On pourrait dessiner les hexagones mais ça nous entraînerait un peu loin.

Vue 1 *Vue 2*



Vue 3

7.2.2 Version 2 : get it tough ! Humain contre ordinateur, première manche (2 points de plus en jeu)

1. *Intégrer dans l'interface SDLv1 le jeu humain contre ordinateur (SDLv2).*
2. *Programmer la fonctionnalité de jeu automatique (IA1) : IA1 part d'un bord et cherche à faire un chemin continu en privilégiant le plus court chemin (distance du bord en nombre d'hexagones).*

7.2.3 Version 3 : dirty badass fighting! Humain contre ordinateur, le combat final (2 points de plus en jeu)

1. *SDLv3 = SDLv2 + Humain vs IA2 + IAx vs IAx (x = 1 ou 2).*
2. *Stratégie IA2 : IA2 cherche en priorité à bloquer, couper les chemins adverses. Dans un deuxième temps il cherche à faire lui-même un chemin.*
3. *Stratégie perso si ça vous dit.*

7.3 Ressources pour la programmation du jeu

7.3.1 Cours SDL

Youtube :
https://www.youtube.com/watchv=7xFaWRzWqr0&list=PL894088275284BDEA

Web :
Documentation SDL 1.2 : https://www.libsdl.org/release/SDL-1.2.15/docs/html/
Faire un jeu : http://fearyourself.developpez.com/tutoriel/sdl/morpion/

Openclassroom :
openclassrom+sdl dans un moteur de recherche. Vous trouverez notamment :
https://openclassrooms.com/courses/apprenez-a-programmer-en-c/creation-d-une-fenetre-et-de-

surfaces

7.3.2 Grilles hexagonales

Web :
Dans quel hexagone je clique ?
http://www.gamedev.net/page/resources/_/technical/game-programming/coordinates-in-hexagon-based-tile-maps-r1800 (le plus utile. Attention à la numérotation.)

http://www-cs-students.stanford.edu/~amitp/Articles/Hexagon1.html (une autre façon plus compliquée)

Distances, coordonnées : http://www.redblobgames.com/grids/hexagons/

Mathématiques des hexagones :
http://www.redblobgames.com/grids/hexagons/implementation.html

7.3.3 Moodle

Vous y trouverez :

- des images de plateau de jeu et de pions que vous pouvez « retailler » avec Gimp pour les adapter à votre interface.
- un squelette de programme C et SDL montrant une boucle de gestion d'évènements. C'est un programme de démonstration, il n'est volontairement pas structuré pour vous obliger à le comprendre et le modifier. Si vous l'utiliser vous devez impérativement créer des sous programmes et le réorganiser pour le rendre lisible et efficace.

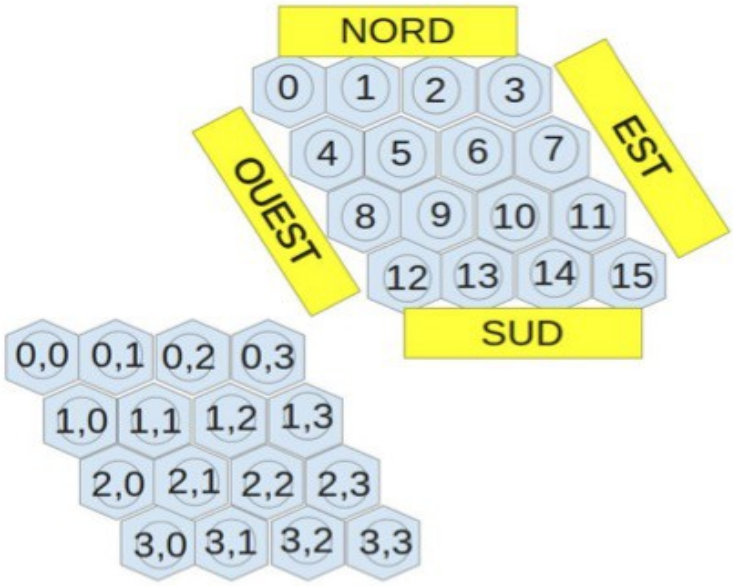
Dans ces ressources, vous allez trouver du code et des algorithmes pour vous aider. Ce code est là pour être copié mais vous devez citer en commentaire l'origine du code et ce que vous avez modifié. De plus vous devez comprendre le code que vous copiez.

7.3.4 Votre tuteur

N'oubliez pas que vous pouvez solliciter votre tuteur. Il est conseillé de préparer des questions précises avant chaque réunion.

7.3.5 Données techniques du sujet HEX

Numérotation des hexagones : Il existe différentes façon de s'y prendre. En voici deux qui ont le mérite de respecter la logique des tableaux du C :



Doc. 1: Exemples de numérotation des hexagones du jeu.

Vocabulaire utile … ou pas :

- *chemin* (connecté) : suite d'hexagones connexes (chacun a un bord commun avec son suivant et un bord commun avec son prédécesseur),
- *cycle* : chemin circulaire formant une boucle,
- *chemin sans cycle* : pas de cycle (boucle) dans le chemin,
- *plus court chemin* : c'est le chemin qui a le moins d'hexagones parmi tous les chemins possibles satisfaisant à une condition,
- *chemin gagnant* : chemin reliant les deux bords opposés de son camp,
- *ligne* : chemin non connexe (il y a des trous),
- *voisinage* : ensemble des six hexagones entourant un hexagone,
- *distance* : c'est le nombre d'hexagones qui séparent deux hexagones par le plus court chemin.

Il est conseillé d'aborder le projet avec une approche globale interface graphique + fonctionnalités, les deux aspects étant intimement liés.

En clair :

1. mettez au point les structures de données (SD),
2. développez l'interface et la gestion des évènements (en lien avec les SD),

3. appelez les fonctionnalités (qui modifient la SD et l'affichage)

Il y a clairement du travail a fournir pour l'interface graphique et les stratégies, mais ne négligez pas le volume de travail induit par les calculs annexes (conversion entre clic souris et hexagone, chemins, distance, etc.), ni par l'algorithmique des stratégies.

7.3.6 Fichiers de configuration et sauvegarde

Les fichiers de configuration sont des fichiers texte (.txt) qui contiennent les informations suivantes :

Le fichier commence par le mot clé `\hex`.

La description du plateau de jeu annoncé par le mot clé « `\board` », puis viennent 11 lignes de 11 caractères représentant une configuration du plateau de jeu :

- Si la case (i,j) est occupée par un pion rouge, le caractère sera « R »
- Si la case (i,j) est occupée par un pion bleu, le caractère sera « B »
- Si la case (i,j) n'est pas occupée le caractère sera « . »

Le mot clé `\endboard` termine cette description.

Le fichier est clos par le mot clé `\endhex`

Le programme doit vérifier la cohérence de toutes les informations du fichiers par rapport aux règles.

Exemple :

La figure 3 sera représentée par les lignes :

```
\hex
\board

\endboard
\endhex
```

Pour la mémorisation de l'historique du jeu on ajoute avant le `\endhex` une nouvelle section :

```
\game
```

suivie d'une ligne par coup :

```
\play joueur i j
```

Exemple : `\play B 5 5`

On termine par un `\endgame` et `\endhex`.

A la fin du fichier on doit pouvoir déduire le joueur dont c'est le tour.

Le programme devra vérifier que les coups sont cohérents et construisent bien le plateau.

7.4 CampusFab 1 : guidage de non-voyants

On souhaite développer un système de guidage pour non-voyants, d'un coût peu élevé, pouvant être déployé facilement pour des événements temporaires, comme une exposition, un salon, etc..

Le cadre est le suivant : conception et fabrication d'un prototype de système de guidage, grâce aux équipements disponibles au sein du CampusFab de l'université.
Le système de guidage repose sur des « bornes interactives » associant un tag NFC (lisible par un lecteur de tag connecté à un Raspberry-PI), écriture braille, et chemin matérialisé par une « bande rugueuse » au sol.

Vous aurez à faire :

- Proposition d'un dispositif de guidage simple (par exemple : par vibrations indiquant la direction à prendre, vers la gauche ou la droite).
- Interface adaptée pour permettre à la personne de sélectionner l'endroit où elle se trouve et où elle souhaite se rendre.
- Algorithme de création de graphe et calcul du plus court chemin permettant de déterminer les différents points de passage permettant de se rendre d'une borne de départ à une borne d'arrivée. L'algorithme doit permettre de simuler le fonctionnement d'un GPS, mais en intérieur.
- Algorithme de guidage permettant d'indiquer à la personne le chemin qu'elle doit prendre pour se rendre à son point d'arrivée, recalculer le plus court chemin en cas d'erreur.
- Développement du système complet et fabrication d'un prototype, qui sera testé dans le local du FabLab (reproduction d'un hall d'exposition miniature). On souhaite que le projet soit déployé en grandeur réelle et pas seulement sur une maquette.

  Ce projet utilisera les machines et matériels électroniques disponibles au CampusFab de l'UT3 (campusfab.univ-tlse3.fr). Il sera codé en C et devra fonctionner sur Raspberry-PI afin de permettre la mobilité d'une personne non-voyante dans un environnement équipé des bornes interactives et des bandes rugueuses.

  Il s'adresse à des étudiants motivés, pouvant proposer des solutions innovantes et originales.

  Contact : Véronique Gaildrat veronique.gaildrat@irit.fr 05 61 55 74 31

7.5 CampusFab 2 : communication

L'Institut des Jeunes Aveugles a proposé un Hackathon « objets interactifs » dont un des sujets proposé est à la base du projet proposé dans le cadre du L2 Informatique.
(http://cherchonspourvoir.org/faislepourvoir/index.php/fr/aucun-titre-2)

Le sujet du Hackathon est le suivant :
Atelier « **Communication** » : trouver un moyen pour permettre aux enfants avec des troubles cognitifs d'exprimer une demande, répondre à une sollicitation extérieure, exprimer un sentiment, une envie. Pour cela, l'idée serait d'utiliser la communication visuelle et en particulier l'orientation du regard et la désignation par le regard pour exprimer le désir d'objet « montrer avec les yeux » ce que veut l'enfant, mais aussi un souhait, un sentiment, une envie. Ainsi, il s'agirait d'adapter et faire varier les supports (images, photos, objets du quotidien, images exprimant des sentiments, etc.). Il serait imaginable que l'outil interactif, à l'aide de caméras qui seraient situées sur les objets ou les images proposées qui capteraient le regard, permettrait de répondre à la désignation visuelle en indiquant le nom de l'objet, de la personne ou de l'action. On pourrait alors proposer un imagier interactif qui permettrait de répondre aux besoins quotidiens des jeunes déficients visuels et non verbaux et d'assurer ainsi une communication non verbale.

Le cadre précis des limites du projet (choix des interactions souhaitables) se fera en relation avec des chercheurs de l'équipe Elipse de l'IRIT qui travaillent régulièrement avec l'IJA sur des projets liés à l'interaction dans le cadre du handicap.

1. Dans le cadre du projet de L2, on souhaiterait faire développer un système interactif, permettant un fonctionnement correspondant au sujet ci-dessus, se basant sur des interactions pour simuler le choix d'une personne handicapée. Ce système sera à concevoir et à développer sans mettre en œuvre le dispositif de suivi de regard mais en se basant sur des interactions usuelles effectuées à l'aide de la souris et /ou du clavier.
2. Une fois que ce système sera pleinement fonctionnel, vous proposerez un prototype interactif permettant à l'utilisateur de communiquer avec ce système à partir d'interactions simples. Il sera par exemple possible d'utiliser une webcam ou un touch-board pour identifier les choix de l'utilisateur.

Vous aurez à faire :
- Analyse des données et conception des structures de données.
- Conception et développement du système interactif fonctionnel tel que présenté au point 1.
- Conception et fabrication d'un prototype, simulant la communication visuelle.

Ce projet pourra utiliser les machines et matériels électroniques disponibles au CampusFab de l'UT3 (campusfab.univ-tlse3.fr) ou achetés spécifiquement pour permettre la fabrication de votre prototype.
Il s'adresse à des étudiants motivés, pouvant proposer des solutions innovantes et originales.

Contact :
Véronique Gaildrat veronique.gaildrat@irit.fr 05 61 55 74 31