Cycle Préparatoire de Bordeaux

université de BORDEAUX

Année 2017-2018

Conception et évaluation d'un logiciel à interface graphique adaptative

Alice DRAILLARD et Hugo FOURNIER

Tuteur : Pierre-Alexandre FAVIER

Résumé

Au cours de ce mémoire, nous traiterons de la conception, du développement, puis de l'évaluation d'un logiciel dont l'interface graphique s'adapte à son utilisateur.
Nous étudierons le fonctionnement de la Conception Centrée Utilisateur (CCU) tout en précisant les étapes de la CCU dans lesquelles notre projet s'inscrit. Nous expliquerons et justifierons ensuite les choix de conception et de développement qui ont mené à notre logiciel, Interfia, qui est un jeu dont l'interface adaptative permet à son utilisateur d'accéder plus facilement à ses boutons de fonction.
Enfin, nous montrerons et analyserons les résultats du test d'Interfia que nous avons mené.

Mots clés : informatique, interface graphique, adaptation à l'utilisateur, CCU, conception, développement de logiciel, ingénierie cognitive, évaluation de logiciel

Abstract

In this essay we will talk about the design, development and evaluation of a software whose graphical interface adapts itself to its user.
We will explain UCD (User Centered Design) whilst indicating which steps of the UCD process we recreated. We will then show and support the design and development choices that led to the creation of our software, Interfia, which is a game whose adaptative interface provides to its user an easier access to its buttons.
Finally we will show and discuss Interfia's results from the test we conducted.

Keywords : computer science, graphical interface, adaptation to the user, UCD, design, software development, cognitive engineering, software evaluation

Introduction

Un logiciel vous fait pester ? Et si le problème venait de la conception de ce logiciel et non de vous ?

En 1990, l'utilisation des tableurs par la population intrigue : Visicalc, le premier tableur commercialisé, continue à être utilisé par une grande partie de la population malgré ses performances faibles, tandis que d'autres tableurs plus évolués et performants comme Wysiwyg sont peu et sous utilisés. En effet, le gain obtenu par un utilisateur comparé au temps d'apprentissage nécessaire pour utiliser le logiciel correctement est jugé trop faible.
On se rend alors compte que même si l'utilité de ce logiciel (sa capacité à atteindre les objectifs de l'utilisateur) est maximale, le problème réside dans son usabilité.
Mais qu'est ce que l'usabilité ?
Mot dérivé de l'anglais *usability*, c'est la qualité de l'interface d'un produit, qui détermine sa facilité d'apprentissage, sa facilité d'utilisation et la qualité des informations qu'il présente. Le logiciel doit être adéquat à la tâche à laquelle il est destiné, mais aussi à l'utilisateur, tout en restant cohérent dans son ensemble.

Et si l'Intelligence Artificielle (ou IA), dont on vante tant les mérites, pouvait faciliter cette Interaction Homme-Machine (IHM) ?
Tel était le questionnement, trop ambitieux en tant que problématique scientifique, qui nous a amenés à ce sujet.

Nous avons donc décidé de nous intéresser à cette notion d'usabilité concernant un logiciel s'adaptant à son utilisateur : Comment concevoir un logiciel à interface graphique adaptative et évaluer son usabilité ?
Nous répondrons à cette problématique par l'exemple, en nous inscrivant dans une démarche d'ingénierie cognitive pour la conception d'un tel logiciel et son évaluation.

Remerciements

Nous tenions à remercier plusieurs personnes pour l'aide qu'elles nous ont fourni au cours de ce projet.

Tout d'abord, notre tuteur Pierre-Alexandre FAVIER pour tout l'accompagnement qu'il nous a fourni avec bienveillance ainsi que Damien MARION pour son aide lors de notre traitement de questionnaires.

Ensuite, Paul DORBEC, notre professeur d'informatique, pour son aide en architecture logicielle et en gestion de versionnage en ligne.
Nous remercions aussi Isabelle ESCOLIN-CONTENSOU et Daniel BLAUDEZ pour leur accompagnement durant notre recherche de tuteur.
Egalement, Isabelle OTTOMANI et Denis COMBES pour nous avoir permis d'utiliser une salle informatique universitaire pour notre expérience.
Enfin, nos 13 amis qui sont gentiment venus tester notre logiciel et nous fournir ces données primordiales à notre projet.

Table des matières

Résumé 3
Abstract 3
Introduction 4
Remerciements 5
Table des matières 6
I. La Conception Centrée Utilisateur 7
I.1 Qu'est-ce que la CCU ? 7
I.1.1 Alternatives de conception 7
I.1.2 Evaluations itératives 8
I.1.3 Contrôle de qualité 8
I.2 Pourquoi la CCU ? 8
II. Conception et développement d'Interfia 9
II.1 Conception 9
II.1.1 Trouver un jeu pertinent 9
II.1.2 Conception d'Interfia 10
II.2 Développement 12
II.2.1 Cadre de développement 13
II.2.2 Architecture logicielle 13
II.2.3 Adaptation de l'interface 15
II.2.4 Version finale d'Interfia 16
III. Expérimentation 17
III.1 Matériel et méthode 17
III.1.1 Méthode 17
III.1.2 Matériel 18
III.2 Protocole 19
III.3 Résultats 21
III.4 Discussion 23
III.4.1 Baisses de performance 23
III.4.2 Echantillonnage et représentativité 24
III.4.3 Un premier cycle de conception 24
Conclusion 25
Bibliographie et sitographie 26

I. La Conception Centrée Utilisateur

I.1 Qu'est-ce que la CCU ?

Dans le cadre de l'obtention d'une usabilité exacerbée pour un logiciel, le principal problème réside dans le choix d'alternatives de conception telles que l'organisation des menus, le type de sélection ou la stabilité d'une fonction : est-il possible de l'utiliser plusieurs fois après avoir cliqué sur son bouton de fonction (commande bistable) ou bien une seule fois (commande monostable) ? Ces choix doivent être comparés aux caractéristiques de la population visée ainsi qu'aux tâches que le logiciel doit pouvoir accomplir.

Afin de mener ces choix à terme, une approche empirique de l'évaluation de l'interface d'un logiciel peut être envisagée : c'est le principe de la CCU, la Conception Centrée Utilisateur.

La CCU repose sur l'idée que l'utilisateur potentiel et ses besoins doivent être pris en compte à tout moment du processus de création (conception et codage) d'un logiciel.
Pour ce faire, dès l'initiation de la conception, ils sont pris en compte à travers des analyses informatiques et un travail sur l'humain, avec par exemple des modélisations de l'activité cognitive de l'utilisateur. Ainsi, dès les premières phases de conception, l'ergonomie est intégrée par l'évaluation de la qualité du logiciel par l'utilisateur, la proposition d'alternatives à une conception, et la justification des critères de choix parmi ces solutions de conception : la qualité ergonomique du produit est évaluée tout le long du processus.
En effet ce qu'on appelle les tests de conception peuvent être simplifiés dans leur déroulement de la manière suivante :

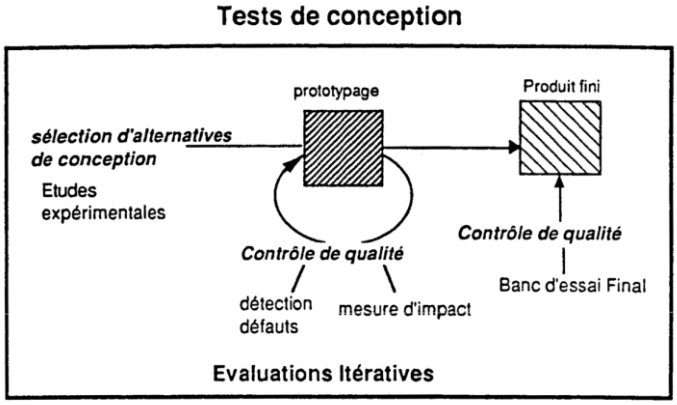


Figure 1 : Schéma des tests de conception dans le cadre de la CCU (B. Senach, *Evaluation ergonomique des interfaces homme-machine : une revue de littérature*, 1990).

I.1.1 Alternatives de conception

On sélectionne des alternatives de conception par des études expérimentales qui servent à départager des possibilités de conception apportant à première vue les mêmes avantages.
Dans le cas d'études de conception réalisées sans prototype, les principes généraux de la conception future apparaissent très rapidement.
On peut donc voir que la précision des objectifs du concepteur permet de guider la conception sans perdre de vue le but premier du logiciel et prime sur le niveau technologique du produit proposé.

L'accent est également mis sur les habitudes de l'utilisateur : on veut que l'interface qu'il utilise lui soit familière, point primordial dans le graphisme du logiciel, d'où l'utilisation d'icônes qui peuvent se retrouver dans d'autres logiciels pour la même fonction et une organisation des barres de tâches qui sont récurrentes dans énormément de logiciels.

I.1.2 Evaluations itératives

Une fois ces directives données, le logiciel est régulièrement soumis à des tests de conception durant lesquels le public, à l'aide de questionnaires normés, détermine les aspects du logiciel qui sont faciles à utiliser mais aussi ses défauts d'usabilité.

Durant chacune de ces évaluations itératives au cours desquelles des tests exploratoires sont menés, les performances et les opinions des populations sont enregistrées, analysées et interprétées, les aménagements de l'interface sont revus ; on étudie les solutions possibles et on choisit celle qui permet le plus de répondre aux attentes et aux besoins des testeurs, sans perdre de vue la performance idéale du logiciel, en créant ainsi un nouveau prototype.

Ces cycles de conception sont renouvelés jusqu'à ce que la dernière obtenue ait les meilleurs résultats possibles.

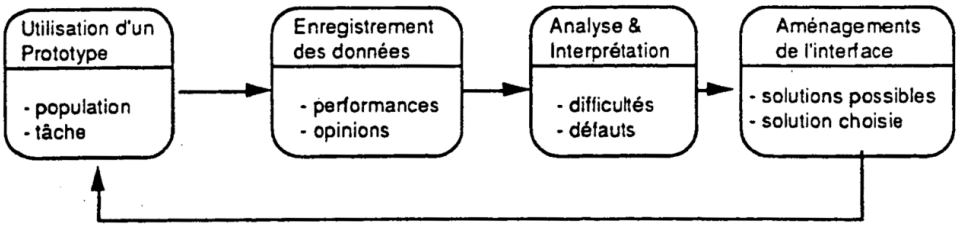


Figure 2 : Schéma d'un cycle d'évaluations itératives dans le cadre du prototypage issu de la CCU (B. Senach, *Evaluation ergonomique des interfaces homme-machine : une revue de littérature*, 1990).

I.1.3 Contrôle de qualité

Le contrôle de qualité, qui intervient à la fin du processus des évaluations itératives, ne se focalise pas sur l'usabilité de l'interface comme le font ces dernières, mais sur le produit dans son ensemble, son but étant d'en faire ressortir les qualités mais aussi et surtout les défauts.
Il permet de hiérarchiser les difficultés du logiciel en fonction de leur impact sur l'utilisateur et sur l'efficacité que le logiciel peut apporter, mettant ainsi en avant les aspects négatifs du logiciel devant impérativement être corrigés.

I.2 Pourquoi la CCU ?

Dans le cadre de notre projet, il était nécessaire de trouver un processus de conception adapté au type de logiciel que nous avions l'ambition de coder, c'est-à-dire un logiciel graphique s'adaptant à son utilisateur, or le processus de conception qu'est la CCU correspond exactement à nos ambitions : il nous permet de suivre une démarche scientifique aboutissant à un produit fait pour l'utilisateur (dans notre cas s'adaptant à son utilisateur), et non un produit auquel l'utilisateur doit s'adapter.

Soumis à des contraintes de temps et de niveau en informatique, nous avons décidé de nous inscrire dans la démarche de la CCU en en représentant un cycle, de la conception aux tests utilisateur suivis d'une interprétation des résultats obtenus et de propositions de changements à effectuer pour les versions à venir.

Nous nous inscrivons ainsi dans le cadre de l'évaluation de l'usabilité de notre interface par une approche empirique, plus précisément par des tests de conception comprenant une sélection d'alternatives, une boucle d'évaluation itérative et une proposition de solutions concernant les résultats de nos tests.

II. Conception et développement d'Interfia

II.1 Conception

II.1.1 Trouver un jeu pertinent

La genèse de la conception du logiciel s'est basée sur deux objectifs : en apprendre plus sur le génie logiciel (si possible en le codant en entier), et mener une démarche d'ingénierie cognitive, c'est-à-dire qu'on va chercher à maximiser l'usabilité de ce logiciel de manière empirique, par la conception d'une interface graphique adaptative.
Cependant, il a très tôt été nécessaire de revoir à la baisse certains aspects trop ambitieux ce projet, en particulier concernant l'intelligence artificielle du logiciel, dont la création était trop complexe en termes d'informatique et de temps nécessaire à sa conception. Elle aurait été également trop complexe à appréhender pour des testeurs en un laps de temps réduit.

Cependant, le dessein de coder un jeu qui s'adapterait à son utilisateur a très tôt émergé : l'idée était de proposer une création gratifiante, à la fois pour ses concepteurs et pour les sujets des tests, ce qui permettrait de mieux motiver les testeurs.
Le but a alors été de trouver un jeu simple, original, à parties rapides, et assez complexe pour nécessiter l'utilisation de différents enchaînements de fonctions (réparties en 4 à 6 boutons) pour arriver à un même résultat et pour cela, l'idée d'un jeu graphique s'est très rapidement imposée : plus facile à adopter pour un utilisateur, il permettait également d'utiliser un langage de programmation connu des concepteurs, Python, avec sa bibliothèque Tkinter destinée aux logiciels à interface graphique.
On s'intéresserait donc à un des cas les plus simples : l'adaptation porterait sur les boutons de cette interface graphique.
De là a émergé la problématique de trouver un jeu où l'adaptation graphique serait pertinente. En effet, il y a pléthore de jeux où l'interface ne nécessite qu'une seule fonction : les échecs, le puissance 4, le 2048 ...
Afin d'être sûrs de répondre à cette problématique, la décision a été prise de créer un jeu sur une interface ressemblant à celle de logiciels de création graphique tels que Paint et PhotoFiltre, à cela près que les fonctions sur le bandeau du logiciel ne seraient pas nécessairement les mêmes.

Alors ont émergé deux idées de jeu qui ont le potentiel pour servir de base à une adaptation graphique, soit des jeux très axés sur le côté visuel de la partie :

Tout d'abord, un jeu inspiré des livres "Où est Charlie?" : l'utilisateur ferait face à un espace rempli de formes géométriques colorées (carrés, rectangles, disques), de taille variable et les unes sur les autres, l'image étant ainsi composée de 4 à 5 plans (ou couches) de figures qui sont superposées.
Le but sera donc d'utiliser un ensemble de fonctions pour découvrir où se cache Charlie (un triangle jaune), sans pouvoir déplacer les figures. L'ensemble des fonctions auxquelles l'utilisateur a accès comprendrait :

- plan +1 : faire passer la figure dans le plan supérieur ;
- plan - 1: faire passer la figure dans le plan inférieur ;
- corbeille : espace de stockage pour un nombre défini de figures (2 ou 3); sa capacité étant donc limitée ;
- sortir de la corbeille ;
- rendre transparent : fait disparaître une figure, valable pour une seule figure à la fois ;
- marquer une figure ;
- colorier une figure.

Les boutons correspondant aux fonctions sont situés sur un bandeau, juste au-dessus de l'espace de jeu. La performance serait mesurée lors du test de ce jeu par la rapidité à trouver Charlie, les données relevées étant le temps et le nombre d'utilisations de chaque fonction.

L'autre jeu est un concours de dessin où il serait nécessaire de réaliser un dessin à partir d'un modèle, le tout sur une interface simplifiée de logiciel de création graphique, avec une intelligence artificielle basique.
Il a été décidé que ce jeu serait celui qu'on coderait puis testerait : il s'appellerait Interfia, pour Interface à Intelligence Artificielle.

II.1.2 Conception d'Interfia

Interfia est un jeu consistant à copier des dessins à partir de modèles dans une interface de création graphique pixélisée à l'aide d'outils basiques de logiciels de création graphique, dont l'interface s'adapte en proposant jusqu'à 4 raccourcis des boutons les plus cliqués.
Ce logiciel ludique dont les sessions doivent être délimitées dans le temps permet ainsi d'étudier la réaction de l'utilisateur par rapport au système d'interface adaptative proposé.
Interfia représente donc deux objectifs : tout d'abord un jeu pour l'utilisateur, mais aussi l'élaboration d'une interface adaptative pour le concepteur.

Nous avons d'abord commencé une phase de conception détaillée de ce jeu qui serait le support de notre expérience, pour ne pas perdre de temps plus tard dans le développement.

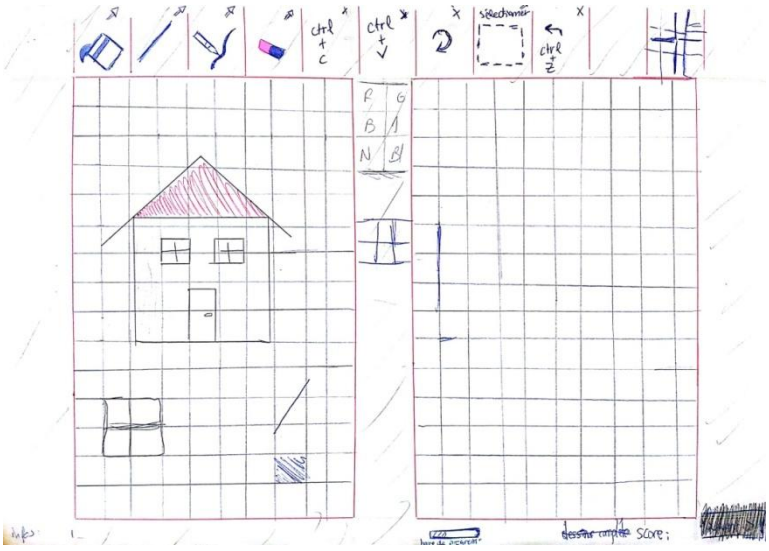


Figure 3 : Première maquette de conception d'Interfia.

Interfia se rapproche de logiciels de création graphique tels que Paint et PhotoFiltre, à cela près qu'il en diffère également par son canevas (c'est-à-dire la zone dans laquelle on dessine), qui est constitué de carreaux ne pouvant être coloriés que d'une couleur, donnant ainsi des rendus similaires à ce que l'on pourrait trouver en pixel art. Avec en cause les limites de mémoire sur Python, notre niveau en informatique, le temps nécessaire pour coder un tel programme, et la difficulté à recopier un dessin au pixel près pour les utilisateurs.
Les limites des carreaux seraient définies par un trait gris, soit une couleur non-présente dans les propositions du jeu, afin d'éviter toute confusion possible pour les utilisateurs, et les boutons des fonctions sont répartis sur un bandeau en haut de l'écran, le bas étant également divisé en son centre par un bandeau vertical en haut duquel se trouve une palette de couleur.
A gauche de ce bandeau se trouve l'espace dans lequel le modèle apparaît, et à sa droite se trouve le canevas, soit l'espace prévu pour que l'utilisateur puisse dessiner, ce modèle privilégiant donc le sens de lecture occidental, de gauche à droite.

La performance sur ce logiciel devait prendre en compte le nombre de clics (plus tard abandonné), et la progression, qui, elle, serait déterminée par un score où les entiers représenteraient le nombre de dessins achevés et les décimales le pourcentage de progression du dernier dessin commencé.
Le temps de dessin d'un jeu serait défini et les données collectées seraient le score, le nombre de clics de fonction sans utilisation (donc les hésitations du sujet) et le nombre de clics par fonctions (et donc l'efficacité).

Interfia posséderait les fonctions suivantes, comme le montre la maquette ci-dessous, qui a été réalisée afin de pouvoir visualiser l'interface :

- pot de peinture : algorithme récursif coloriant tout pixel proche du carreau sélectionné et étant de la même couleur ;
- trait : trace un segment d'un carreau à l'autre (prend en compte le carreau de départ et celui d'arrivée) ;
- crayon : colorie le(s) carreau(x) sur lequel (lesquels) la souris passe avec le bouton appuyé ;
- gomme : crayon blanc ;
- sélection rectangulaire : sélectionne tous les pixels traversés ou à l'intérieur du rectangle défini par le point où le bouton clic de la souris a été actionné, et celui où il a été relâché ;
- copier : enregistre les données pixel de la sélection, il n'est possible d'avoir qu'une seule image en mémoire à la fois et il n'est pas possible de copier le modèle ;
- coller : ajoute l'image copiée dans le canevas, le point du clic correspondant au coin en haut à gauche du rectangle sélectionné ;
- supprimer : remplace la sélection par un rectangle blanc de mêmes dimensions ;
- annuler : annule le dernier appel de fonction effectué, n'est utilisable qu'une fois après l'usage d'une autre fonction (chaque fonction sort deux listes : L qui correspond au nouvel affichage modifié, et L-1 qui était la matrice avant l'utilisation de la fonction) ;
- rotation : rotation de la partie sélectionnée de 90° dans le sens trigonométrique, utilisable 4 fois tant que la sélection est gardée, si ladite sélection n'est pas carrée, une fois la rotation effectuée, les pixels non-couverts par la figure sont blancs.

La palette de couleurs est composée des trois couleurs primaires de la peinture combinées à celles de la physique, soit rouge, bleu, jaune et vert, ajoutées au noir et au blanc, créant ainsi une palette simple mais permettant cependant de créer une large variété de dessins figuratifs.

Comme il est possible de le voir sur la figure 3, Interfia offre deux types de fonctions :

- celles où la fonction peut être utilisée à l'infini tant qu'on ne clique pas sur un autre bouton de fonction : pot de peinture, crayon, trait, gomme, sélectionner et annuler.
  Elles sont marquées par un curseur en haut à gauche de chaque bouton et fonctionnent avec une boucle *while(clic maintenu)* ou par clics uniques.
- celles qui nécessitent l'activation du bouton "sélectionner" avant d'être utilisées : rotation, copier et coller, marquées par une croix.

Nous avons choisi de ne pas rendre possible les raccourcis clavier, afin de privilégier la seule interaction homme-souris-écran, exclusive dans notre problématique scientifique.
Aussi, avoir des fonctions ne fonctionnant qu'une fois après un clic sur leur bouton aurait pu permettre de ne relever que le nombre de clics de fonctions ou le nombre d'utilisations de fonctions, mais nous avons décidé de ne pas mettre cette particularité en place afin de ne pas rendre Interfia frustrant pour l'utilisateur.

Ensuite s'est posé la question des dimensions de l'interface : Tkinter ne permettant pas d'adapter sa fenêtre aux dimensions de l'écran, il était nécessaire de créer des échelles. Pour ce faire, la taille d'un bouton de fonction (que nous appellerons icône) a été prise comme référence : les bandeaux feraient

une icône de largeur et les espaces du modèle et du canevas feraient 5 icônes de largeur et 9 de hauteur, avec un quadrillage de 10×18 carreaux.
La décision fut prise de créer deux échelles : l'échelle 1 (pour laquelle un carreau correspond à 35×35 pixels) et l'échelle 1,6 (56×56 pixels pour un carreau), ce qui a conduit à une maquette mise à l'échelle (bien que très légèrement différente à cause des dimensions du papier) :

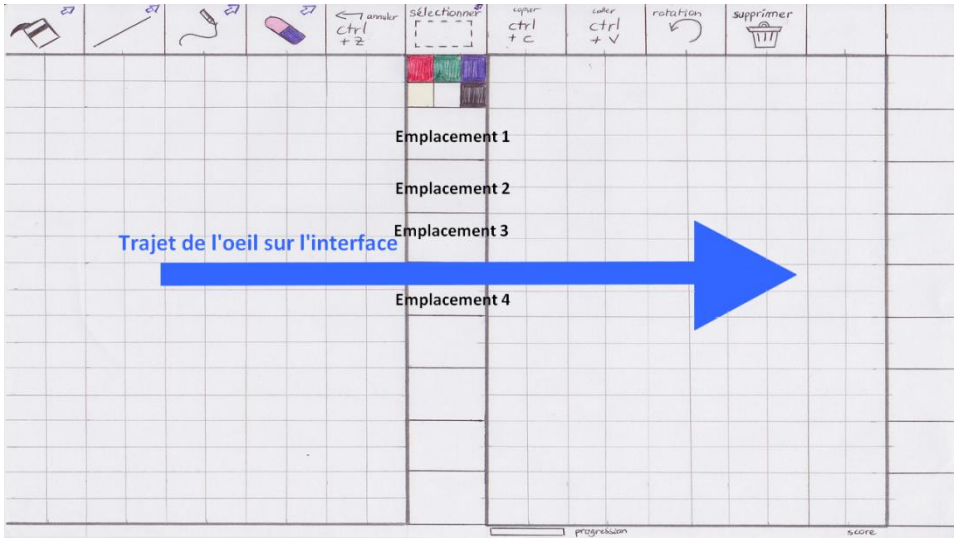


Figure 4 : Deuxième maquette de conception d'Interfia.

La nature de l'adaptation de l'interface fut facilement déterminée : les icônes se déplaceraient pour se rendre plus accessibles de l'utilisateur, et le jeu étant basé sur un sens de lecture occidental, l'endroit où les boutons ont le plus de chance de capter l'attention de l'utilisateur est le bandeau central car il se trouve sur le trajet de l'œil entre le modèle et le canevas ; les icônes apparaîtraient donc dessus.

La question qui s'est alors posée concernait le déroulement de cette adaptation : fallait-il attendre de discerner les habitudes de l'utilisateur avant de changer l'interface ? Ou bien la changer dès le départ au prix d'une évolution très rapide et peut être désarçonnante avant de se stabiliser ? Est-ce que déplacer les boutons ne serait pas bien trop perturbant pour l'utilisateur ?

La décision a été prise de les dupliquer afin de ne pas désorienter l'utilisateur, avec de l'espace pour quatre icônes au centre du bandeau vertical destinés aux boutons des fonctions les plus utilisées (en nombre de clics) et l'adaptation se mettrait en place dès le début du jeu : dans le bandeau central apparaissent dans l'ordre des raccourcis des quatre boutons les plus utilisés par le testeur(en haut le plus utilisé et en bas le quatrième plus utilisé).
Le mécanisme d'IA derrière l'adaptation est donc très simple puisqu'il s'agit d'un simple classement des nombres d'utilisations de chaque fonction.

II.2 Développement

Cette partie se concentre sur le côté technique et informatique de notre projet. Cependant, nous avons essayé de la rendre en majorité compréhensible même par un lecteur non informaticien. Un effort similaire a été réalisé pour notre code source disponible en annexe qui a été régulièrement commenté dans un effort de clarification, mais le code reste bien moins lisible que cette partie à cause de toutes les fonctions et logiques spécifiques au langage de programmation.

II.2.1 Cadre de développement

Suite à cette phase de conception détaillée, nous avons donc pu commencer sereinement le développement informatique du logiciel. Pour ce faire, nous avions déjà choisi le langage de Programmation Python car nous l'utilisons tous les deux. Après des recherches sur Github, nous étions convaincus que grâce à la bibliothèque Tkinter il nous permettrait de réaliser le jeu que nous avions conçu et dont nous avons besoin pour notre expérience, car étant la plupart du temps un bon langage pour faire de la programmation orientée objet.

Egalement, pour pouvoir nous échanger progressivement l'avancement du code en compilant les différentes versions des fichiers du code source, ainsi que le sauvegarder en ligne en cas de perte de disque dur, nous avons appris à utiliser un logiciel de gestion de version : Git. Avec cet outil très pratique, nous avons crée un dépôt git (accessible en public dès aujourd'hui à l'adresse suivante : https://services.emi.u-bordeaux.fr/projet/savane/projects/interfia) que nous avons pu héberger sur les serveurs du CREMI via leur site de gestion de projets « Savane », ce qui nous a permis de pouvoir régulièrement récupérer la dernière version du code source depuis plusieurs ordinateurs. Enfin, nous utilisions Idle 3.6 (dernière version) comme éditeur de code, choix simple mais efficace pour du Python. Voici ci-dessous une capture d'écran de ce à quoi a pu ressembler un écran lors d'une séance de développement, avec la toute première version de notre interface.

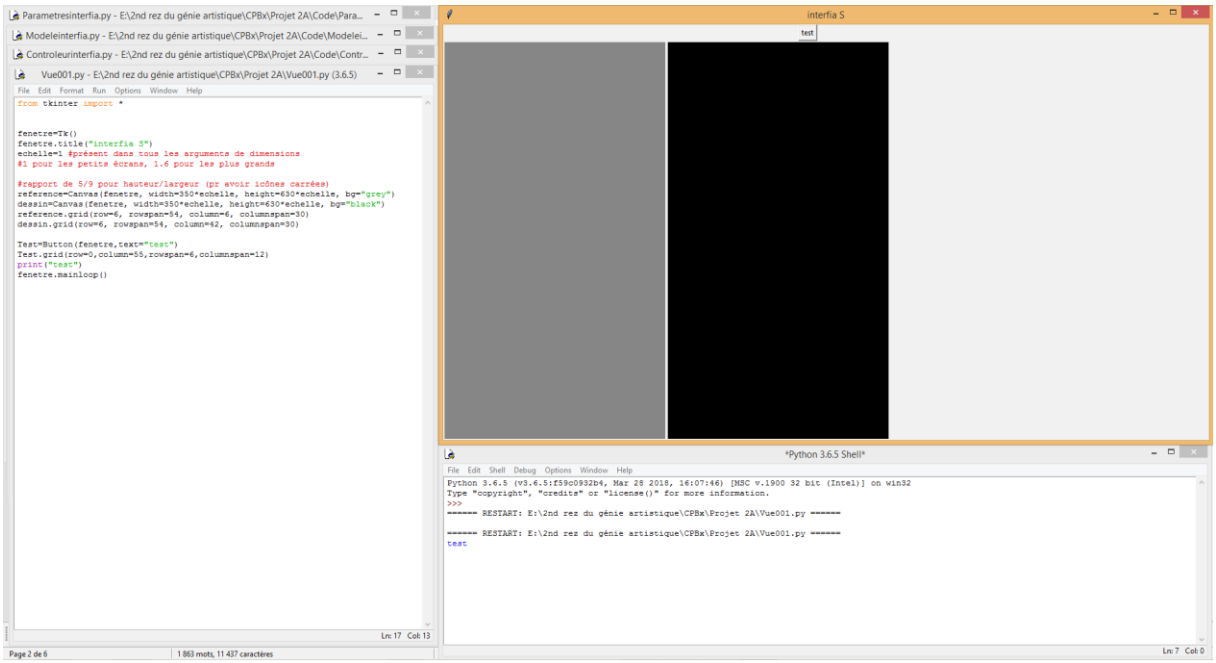


Figure 3 : Capture d'écran de notre environnement de développement et Interfia à ses débuts.

Les différents fichiers de code se trouvent à gauche, pour une navigation plus facile (les commentaires, qui ne sont pas lus quand on exécute le code, sont en rouge). En haut à droite, la fenêtre Tkinter est ouverte par l'exécution de Vue001.py. En bas, la console Python permet surtout, dans de tels programmes avec une autre interface, d'écrire des valeurs via la fonction « print » à différents endroits du code pour débuguer et de lire les messages d'erreurs quand il y a un souci.

II.2.2 Architecture logicielle

Pour organiser notre développement, nous avons choisi de suivre une architecture proche de celle dite « MVC » pour Model-View-Controller (Modèle-Vue-Contrôleur), classique pour les logiciels à interface graphique. Dans une telle architecture, le code source est donc divisé en 3 fichiers. La Vue gère tout ce qui est affiché à l'écran de l'utilisateur, tandis que le Modèle, invisible à l'utilisateur,

stocke et traite des données en arrière-plan. Le Contrôleur est le lien entre les deux qui, selon les actions de l'utilisateur, va mettre à jour le Modèle et la Vue de façon adéquate.

Dans notre cas, nous avons décidé d'y ajouter un fichier Paramètres pour pouvoir, dans la préparation du test, passer de l'interface version statique à l'interface adaptative, ou d'une séquence de dessins à l'autre, ou encore modifier le temps imparti avant que la fenêtre ne se ferme et les données qui s'écrivent dans un fichier texte. Voici la capture d'écran de la version Démo (5 minutes, commence par un dessin spécial, statique) de notre fichier Paramètres qui est commenté exhaustivement.

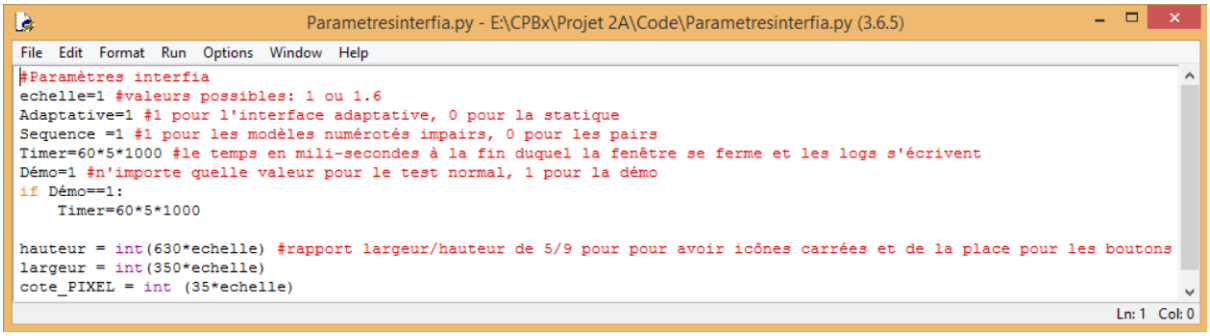
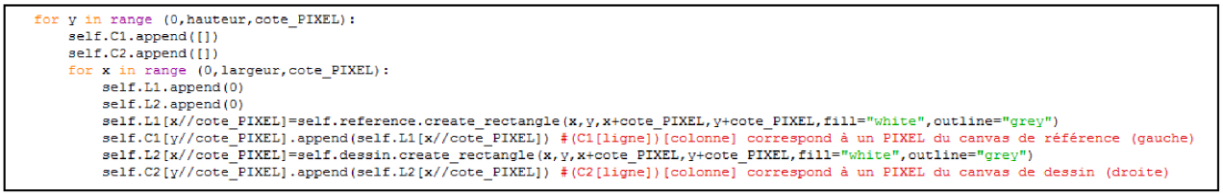


Figure 4 : Capture d'écran d'un fichier Paramètres d'Interfia dans son intégralité.

Les trois dernières valeurs, qui définissent les dimensions en nombre de pixels des deux zones de dessin (modèle à gauche, dessin de l'utilisateur à droite) sont régulièrement utilisées par les 3 autres fichiers. Ceci permet par exemple, en changeant un paramètre et deux lignes de code, de se rapprocher d'un logiciel plus professionnel (mal optimisé) avec des petits PIXELS (illustration en annexe).

Notre fichier Vue définit une seule classe d'objet, la classe Vue dont les méthodes (fonctions propres à une classe) permettent la génération et gestion de la fenêtre, ses boutons, ses canevas (« reference » à gauche, « dessin » à droite), son timer, la mise à jour des PIXELS (en échelle 1, un PIXEL est un des carrés de 35 pixels de côté visible sur le quadrillage) et des raccourcis.
A chacun de ses deux canevas sont associées une liste de listes (formant un tableau ou une matrice) de PIXELS. Voici la génération, par double boucle itérative, des deux quadrillages (précisons que le « self. » devant une variable signifie que cette variable est propre à son objet parent, ici la classe Vue) :



```
for y in range (0,hauteur,cote_PIXEL):
    self.C1.append([])
    self.C2.append([])
    for x in range (0,largeur,cote_PIXEL):
        self.L1.append(0)
        self.L2.append(0)
        self.L1[x//cote_PIXEL]=self.reference.create_rectangle(x,y,x+cote_PIXEL,y+cote_PIXEL,fill="white",outline="grey")
        self.C1[y//cote_PIXEL].append(self.L1[x//cote_PIXEL]) #(C1[ligne])[colonne] correspond à un PIXEL du canvas de référence (gauche)
        self.L2[x//cote_PIXEL]=self.dessin.create_rectangle(x,y,x+cote_PIXEL,y+cote_PIXEL,fill="white",outline="grey")
        self.C2[y//cote_PIXEL].append(self.L2[x//cote_PIXEL]) #(C2[ligne])[colonne] correspond à un PIXEL du canvas de dessin (droite)
```

Figure 5 : Classe Vue d'Interfia.

Le Modèle définit une autre classe d'objet, la classe Modèle, dont les méthodes permettent de mettre à jour le tableau associé à chaque zone de dessin. Notamment, un objet de classe Modèle possède un dictionnaire qui associe les couleurs utilisées à des valeurs numériques (notre rouge vaut 0 par exemple), ce qui permet de générer une matrice entièrement numérique, traduisible en quadrillage de couleurs.

Le Contrôleur est le fichier principal, celui qui permet de jouer à Interfia lorsqu'on l'exécute. Dedans sont importés la Vue et le Modèle pour ensuite générer une classe Vue, sa fenêtre, ses boutons, ainsi que deux classes Modèles, une pour chaque quadrillage. Cela permet notamment de comparer la référence au dessin pour afficher sa progression au joueur, ainsi que de récupérer facilement la couleur d'un PIXEL. C'est aussi dans ce fichier que sont définies les fonctions (en fait divisées en 2, 3 ou 4 fonctions entre la Vue et le Contrôleur) ensuite associées aux différents boutons, la séquence de modèles ainsi que l'adaptation de l'interface via la mise à jour des raccourcis.

L'architecture MVC+Paramètres+Icônes d'Interfia est ainsi résumée dans le schéma suivant :

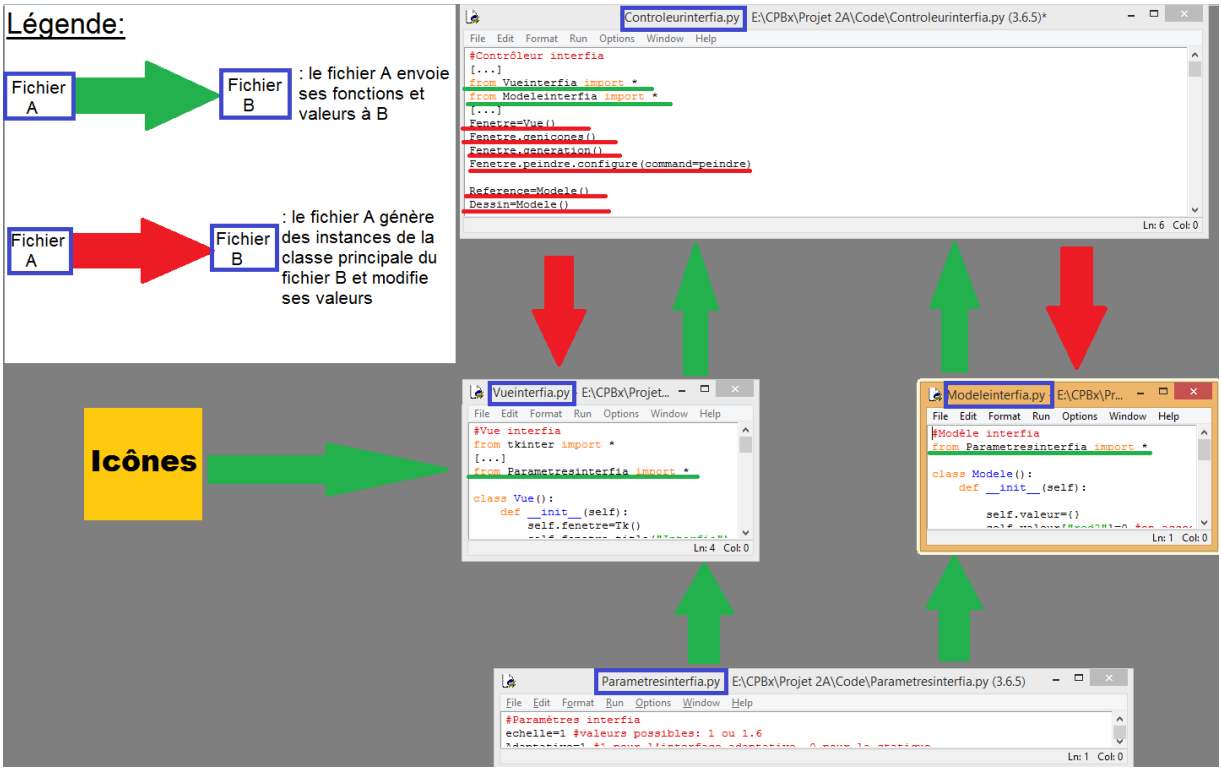


Figure 6 : Schéma du modèle MVC + paramètres + icônes d'Interfia (optionnel : zoomer pour lire le code).

II.2.3 Adaptation de l'interface

L'intérêt d'Interfia dans le cadre de notre projet réside dans son système d'interface adaptative, cet élément d'intelligence artificielle très simple, à savoir l'adaptation automatique des raccourcis.
Par rapport à la conception, nous avons décidé de finalement n'avoir que 4 raccourcis. Regardons donc le fonctionnement du système, sans capture d'écran cette fois car la fonction *adaptation(bouton)*, définie dans le contrôleur (code en annexe), est très longue et peu lisible. En effet, elle doit gérer séparément le cas de chaque bouton : comme prévu dans la conception, il s'agit d'un simple classement des fonctions les plus utilisées, mais pour en coder un fonctionnement parfait il faut être rigoureux.

Tout d'abord, à chaque lancement d'Interfia, des variables Clic_[nomdubouton] sont crées pour compter le nombre de clic de chaque bouton, elles commencent évidemment à 0 ; la variable associée au bouton augmente donc de 1 à chaque clic sur le bouton. Egalement, à chaque clic, la fonction *adaptation([bouton sur lequel on vient de cliquer])* est appelée. Celle-ci va classer le nombre d'utilisations des boutons par ordre croissant ; la difficulté étant qu'après le classement, le programme ne sait plus à quel bouton appartient quelle valeur : la fonction a simplement un classement croissant de ces chiffres ([0,0,3,4,5,7,20,22] par exemple). Donc, on teste si le Clic_bouton associé au bouton sur lequel on vient de cliquer (seule valeur que la fonction peut différencier des autres) arrive dans le top 4 de ce classement en le comparant à chaque valeur et en s'arrêtant quand on rencontre une valeur plus élevée.
Si c'est le cas, la fonction fait une sauvegarde des 4 raccourcis qui étaient là avant, puis place le raccourci du bouton fraîchement cliqué selon son classement (le plus haut si c'est devenu la fonction la plus cliquée). Ensuite, grâce à la sauvegarde, tous les raccourcis en dessous sont descendus d'un rang ; le 4ème raccourci qui descend est supprimé puisqu'il n'y a pas de 5ème emplacement pour les raccourcis.

Les raccourcis sont créés grâce à la méthode *creer_raccourci(bouton,emplacement)* de la Vue. Cette fonction crée en fait un nouvel objet Button avec la même commande et icône que le bouton de base pris en argument et le place selon l'emplacement donné en argument : juste en dessous des couleurs pour l'emplacement 0, le dernier emplacement est le numéro 3.
Egalement, la fonction va classer dans une liste propre à la Vue ce nouveau raccourci et en même temps donner une valeur à la variable *[nomdubouton]racc* qui vaut -1 si le bouton n'a pas de raccourci actuellement, ou la valeur de l'emplacement s'il y en a un. Là encore, cela rajoute beaucoup de lignes dans le code mais il est indispensable pour pouvoir mettre à jour le relief et bord du raccourci en même temps que son bouton (un bouton est enfoncé et encadré en bleu quand il est actif), ainsi que son état pour certaines fonctions (la fonction Copier est désactivée si rien n'est sélectionné).

II.2.4 Version finale d'Interfia

Afin que notre expérience et ces explications paraissent plus claires, voici un visuel légendé de la version finale (qui a donc été testée) de notre logiciel :

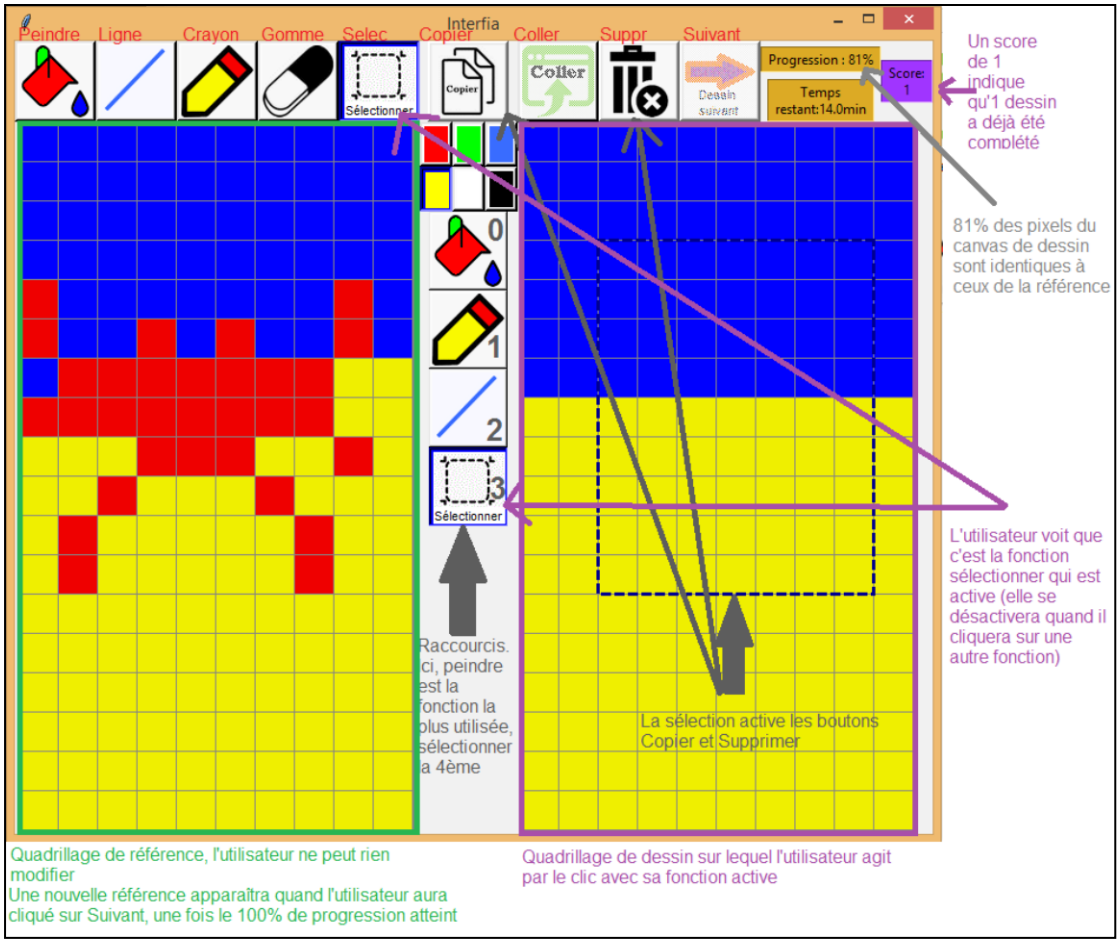


Figure 7 : Capture d'écran d'Interfia dans sa version finale.

Par rapport à la conception, nous avons dû modifier quelques choix et pris quelques libertés :

- par manque de temps (le temps de développement explosait régulièrement suite à des problèmes qu'on ne pouvait prévoir) les fonctions Rotation et Annuler n'ont pas été implémentées ;
- la progression au dessin suivant n'est finalement pas automatique, elle se fait par un bouton activable seulement quand la progression est à 100 % (dessin identique à la référence) ; ce

choix a été fait pour permettre aux joueurs n'utilisant surtout que 3 fonctions d'avoir ce bouton en tant que 4ème raccourci et pour éviter au joueur d'être surpris lors de la complétion de son dessin ;

- nous avons pu ajouter un minuteur, avec un affichage indiquant le temps restant à la minute près, qui ferme la fenêtre au bout d'un timer (voir Paramètres ci-dessus) et une fonction qui permet d'écrire automatiquement dans un fichier texte toutes les données qui nous intéressent lorsque la fenêtre se ferme.

III. Expérimentation

III.1 Matériel et méthode

III.1.1 Méthode

Pour pouvoir éclaircir notre problématique scientifique et terminer un premier cycle de conception, nous avions besoin de faire tester notre logiciel, nous avons donc invité via nos réseaux sociaux et contacts privés des amis à participer à une séance de test le Lundi 23 Avril à 14h, avec un prévisionnel de participants d'environ 15 personnes et 20 au maximum.
Ce sont 16 personnes qui nous ont répondu et à qui nous avons précisé par e-mail les informations pratiques et le déroulement de la séance dans ses grandes lignes ainsi que la nature du logiciel qui devait être testé (c'est un logiciel graphique se rapprochant de logiciels de traitement d'image tels que PhotoFiltre et Paint permettant de dessiner sur un canevas à carreaux, et ainsi de créer un dessin à style pixélisé), et joint une copie du formulaire de consentement éclairé (disponible en annexe) qu'ils ont eu à signer sur place. Sur ces 16 personnes, 13 ont finalement participé au test, dont une femme et douze hommes. Il est à noter que tous nos testeurs sont des étudiants de l'Université Bordeaux 1 (de bac +1 à bac +3) car ils sont les seuls à pouvoir ouvrir une session dans une salle informatique de l'université. Aucun d'entre eux n'avait testé l'interface que nous proposions avant l'expérience.

Ce choix d'une unique séance de test nous a permis un gain important de temps et l'assurance que les testeurs soient tous soumis aux mêmes conditions, apportant une validité supplémentaire aux résultats. Ce choix pratique était possible car toutes les données qui nous intéressaient étaient récupérables sur les différents ordinateurs des testeurs et sur les formulaires qu'ils devaient chacun remplir : nous n'avions pas besoin de suivre et d'accompagner précisément chaque utilisateur, comme ce peut être le cas dans d'autres expériences de test.

Pour aboutir à des résultats le moins biaisés et le plus pertinents possible, la séance de test a été organisée en contre-balancement : c'est-à-dire que les testeurs sont divisés en deux groupes A et B, le groupe A commençant essayer par la version statique du logiciel et continuant avec la version adaptative, et l'autre les découvrant dans l'ordre inverse. Ainsi, le calcul des moyennes des deux groupes permet aux résultats de ne pas subir de biais d'apprentissage lors du calcul des moyennes sur les données.
En effet, s'il n'y avait eu qu'un seul groupe de testeurs essayant d'abord la version statique de l'interface, puis sa version adaptative, une amélioration aurait pu être due à une habitude développée par l'utilisateur face au programme plutôt qu'à l'apparition des raccourcis : cette habituation à l'interface aurait donc pu confondue avec les bénéfices liés à l'apparition de l'adaptation dans l'interface.

La durée des sessions, fixée à 17 minutes, a été pensée sous 3 contraintes : permettre à chaque utilisateur de compléter au moins 8 dessins, ne pas trop fatiguer les testeurs et ne pas les garder trop longtemps. La durée totale de l'expérience est donc inférieure à une heure, ce qui permet une évaluation relativement rapide.

Sont prises en compte lors de la collecte des données :

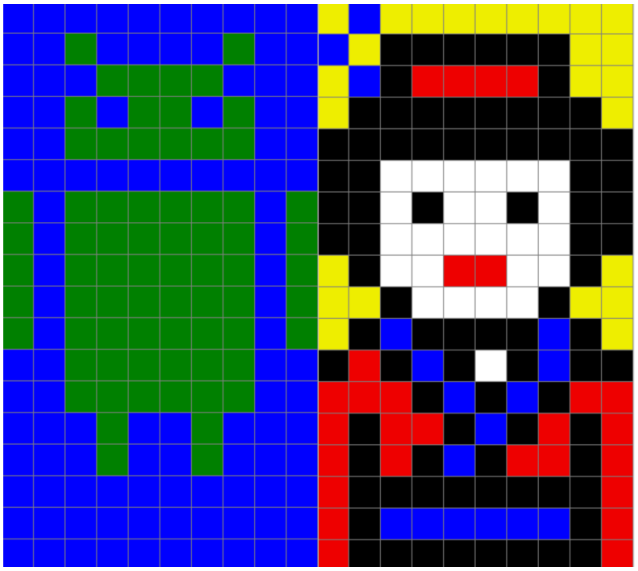
- le nombre de clics et d'utilisations de chaque fonction, permettant de mettre en lumière la stratégie du testeur face au jeu, ces données ne sont pas directement significatives mais pourraient montrer une corrélation inattendue ;
- le nombre d'hésitations (soit un clic sur une fonction qui n'est pas suivi de l'utilisation de ladite fonction sur le canevas), afin de voir si l'apparition de l'adaptation de l'interface désoriente ou perturbe les utilisateurs ;
- le score (sa partie entière correspondant au nombre de dessins achevés, et sa partie non-entière étant le pourcentage d'avancement du dernier dessin interrompu), soit l'efficacité de chacun ;
- la version d'interface et la séquence de dessin pour vérifier qu'il n'y a pas eu d'erreur lors de l'attribution des programmes à lancer.

Ces données ont été relevées aussi bien pour le test de la version adaptative du logiciel que pour sa version statique, tout en relevant le numéro de poste associé (afin que les données restent anonymes) sous forme de fichier texte séparés : un pour l'adaptative, un pour la statique.

III.1.2 Matériel

Nous avons obtenu pour 2h30 la disposition de la salle informatique 104 du bâtiment B5 du campus Bordeaux 1, équipée de 20 ordinateurs sous Windows dotés d'un clavier et d'une souris et d'un écran 17 pouces (l'échelle 1 d'Interfia remplit convenablement un tel écran). Nous avons utilisé plusieurs clés USB chargées au préalable avec les fichiers nécessaires au lancement du logiciel avec les bonnes séquences et version de l'interface.

Ont été créés en vue de l'expérience 36 modèles de dessin sur l'interface, séparés en deux groupes de 18 modèles chacun nommés "pair" et "impair" de difficulté à peu près équivalente et croissante dans leur réalisation, comme par exemple les modèles 4 et 29.



A gauche, un exemple de modèle simple issu du groupe pair, et à droite un modèle compliqué à copier issu du groupe impair.

Le reste des modèles est disponible en annexe.

Figure 8 : Deux des modèles utilisés dans le cadre de l'expérience sur Interfia.

Ont également été préparés 18 exemplaires (au cas où un testeur voudrait recommencer un questionnaire mal rempli) de chacun des documents suivants : un formulaire de consentement éclairé et les questionnaires normalisés AttrakDiff et SUS (disponibles en annexe), destinés aux testeurs afin

de respectivement informer les sujets des droits qu'ils ont au regard de l'expérience, et leur faire évaluer la performance et la pertinence du système d'adaptation.
Le SUS (System Usability Scale) détermine l'apport du système en termes d'usage, sa facilité d'utilisation.
L'AttrakDiff, pour sa part, jauge le plaisir d'utilisation du logiciel en questionnant le sujet sur son ressenti face à l'interface adaptative. Ce ressenti des utilisateurs est retranscrit en quatre catégories qui seront échelonnées lors de la collecte des résultats de l'AttrakDiff : la qualité pragmatique (QP), la stimulation (QHS), l'identité (QHI) et l'attractivité globale (ATT).

Il est à noter que la disposition de la salle ne permettait pas une visibilité optimale de la démonstration sans utilisation du vidéoprojecteur (que nous n'avons pas su faire fonctionner car nous n'avions pas pu avoir accès à la salle avant) comme le montre cette photo prise lors de la séance.

Figure 9 : Photographie de la démonstration d'Hugo sur Interfia devant les testeurs.

III.2 Protocole

Afin d'obtenir un protocole le plus efficace possible, il a été nécessaire d'organiser le déroulement de l'expérience pour ne pas perdre de temps, ne pas désorienter ou décourager les testeurs et minimiser les biais, il est donc issu de beaucoup d'échanges et réflexions au sein du binôme et avec notre tuteur. Pour être sûr de le suivre rigoureusement en tant que superviseurs de test, nous l'avons résumé sur une fiche que nous avions lors du test (qui est disponible en annexe) ; en voici le détail.

Nous commençons avec un accueil différé des premiers testeurs dès 13h45, à qui on ne communique aucune information supplémentaire mais les invitons tout d'abord à signer un formulaire de consentement éclairé (disponible en annexe) permettant l'utilisation de toutes les données (anonymes puisque organisées par numéro de poste) et photos collectées lors de l'expérience et expliquant que chacun peut quitter l'expérience à tout moment s'il le veut ; puis à ouvrir leur session et enfin nous leur donnons via clé USB les fichiers adéquats selon le poste sur lequel ils s'installent.

A 14h10, une fois tous les ordinateurs et testeurs prêts, commence une démonstration du logiciel, en version statique d'abord, afin que tout le monde puisse comprendre les commandes et leur utilité puis en version adaptative pour leur expliquer le système de raccourcis qu'ils devront évaluer, et également afin d'expliquer à tout le monde leur objectif, soit ici d'arriver à recopier un maximum de dessins dans chaque séquence de 17 minutes.

Ensuite commence la première séquence de tests : pendant 17 minutes, chacun doit essayer de copier un maximum de dessins à partir du modèle qui s'affichera sur la moitié gauche de l'écran.
Il est important de noter que tous les groupes ont commencé par travailler sur la série de dessins impaire.

Afin d'éviter au maximum un effet de difficulté de la part des testeurs, nous avons laissé un répit, une respiration de 8 minutes entre les deux séquences de test.

Pour la deuxième phase de test, les groupes ont travaillé sur la série paire de dessins.

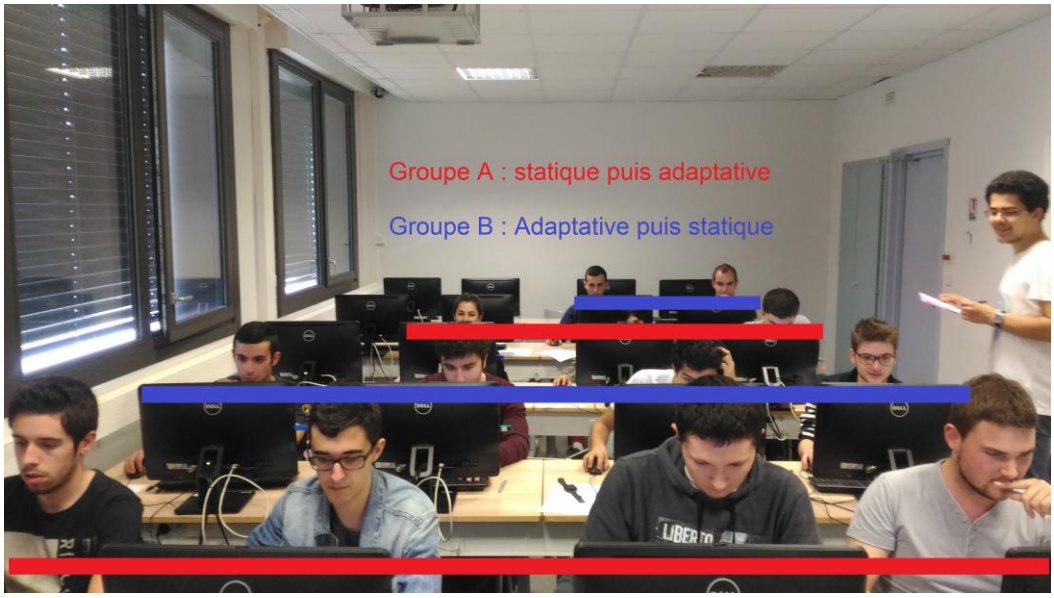


Figure 10 : Photographie légendée de nos deux groupes de testeurs en pleine session de jeu.

A l'issue de cette dernière phase, les testeurs répondent à deux questionnaires normés, après qu'on leur a précisé que leurs réponses doivent concerner uniquement la plus value de l'adaptation du logiciel et que seul nous intéresse leur ressenti honnête vis-à-vis de leur expérience utilisateur.
L'échantillon est confronté aux questionnaires dans l'ordre suivant : d'abord le SUS, puis l'AttrakDiff.

Après cela, les testeurs sont libres de partir mais invités à discuter de leurs ressentis avec les organisateurs du test, et à leur faire part de leur avis sur le logiciel, son système de raccourcis et à nous poser des questions.

A la fin de l'expérience, les données collectées par le programme durant la session d'expérience sont récupérées sur les ordinateurs par clé USB, une fois tous les testeurs partis, ce pour chaque poste de test.

III.3 Résultats

Les scores et les hésitations ont été collectés sous forme de fichiers texte et ont ensuite été compilés dans un tableur pour obtenir comme bilan une moyenne de ces données :

	Score moyen*	**Hésitation moyenne****
Version statique	12,5684615384615	3,92307692307692
Version adaptative	12,4521360683761	4
Différence (Statique - Adaptatif)	0.11632547008	-0.07692307692

Figure 11 : Tableau du score et de l'hésitation moyens selon la version d'Interfia.

* partie entière = nombre de dessins achevés ; partie non-entière = pourcentage de progression du dernier dessin en cours
** nombre de clics sur une fonction qui n'ont pas été suivis de l'utilisation de ladite fonction

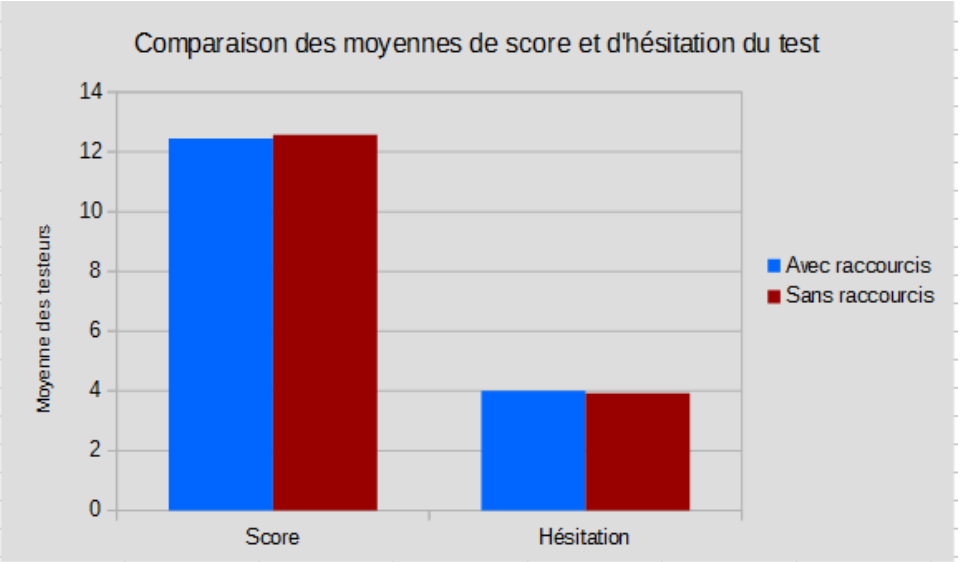


Figure 12 : Graphique du score et de l'hésitation moyens selon la version d'Interfia.

On peut donc observer une très légère diminution du score entre la version statique et la version adaptative du logiciel.
L'hésitation, elle, augmente lors du passage à la version adaptative.

Les questionnaires permettant une retranscription numérique, l'apport de l'adaptation de l'interface a abouti aux résultats suivants pour le questionnaire AttrakDiff :

Sous-échelle	Qualité pragmatique (QP)	Qualité hédonique - Stimulation (QHS)	Qualité hédonique - Identité (QHI)	Attractivité globale (ATT)
Score moyen /3	1,37369231	-0,04384615	-0,044	0,72538462

Figure 13 : Tableau des résultats des différentes échelles de l'AttrakDiff.

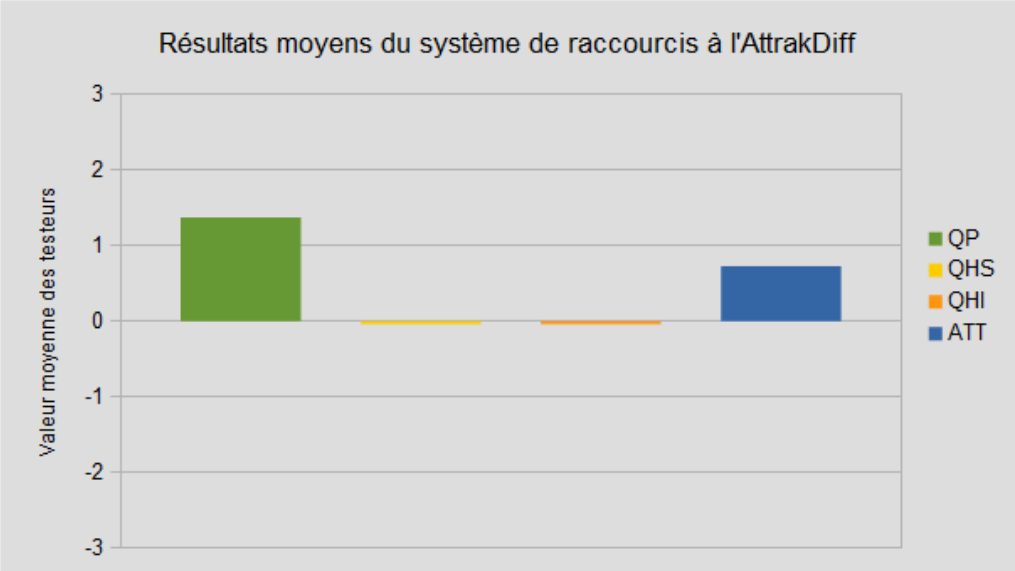


Figure 14 : Graphique des résultats des échelles de l'AttrakDiff.

On peut voir que la qualité pragmatique et l'attractivité du logiciel ont été évaluées positivement, tandis que la qualité hédonique est négative avec une moyenne de -0,04392308 pour cette catégorie.

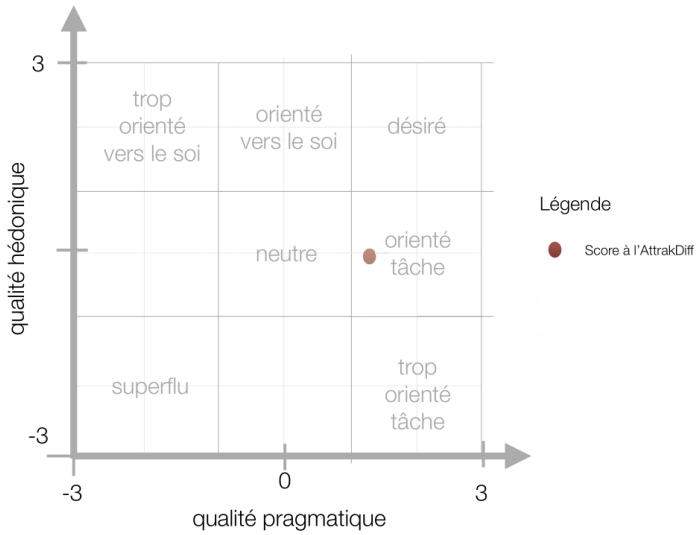


Figure 15 : Graphique de l'interprétation de notre résultat pour l'AttrakDiff (C.Lallemand, V.Koenig, G.Gronier, R.Martin, *Création et validation d'une version française du questionnaire AttrakDiff pour l'évaluation de l'expérience utilisateur des systèmes interactifs*, 2015).

En utilisant le graphique de présentation des résultats de l'AttrakDiff, on peut voir que le système est légèrement orienté tâche, avec une incertitude non calculée qui est très élevée étant donné la petite taille de notre échantillon.

Le SUS, en revanche, propose un score plus élevé, soit une usabilité bien présente de la part du système : avec un score moyen de 78,65384615, il peut être considéré comme bon (d'après l'échelle de Bangor, Kortum et Miller, 2008), c'est-à-dire qu'il est acceptable dans son utilisation.

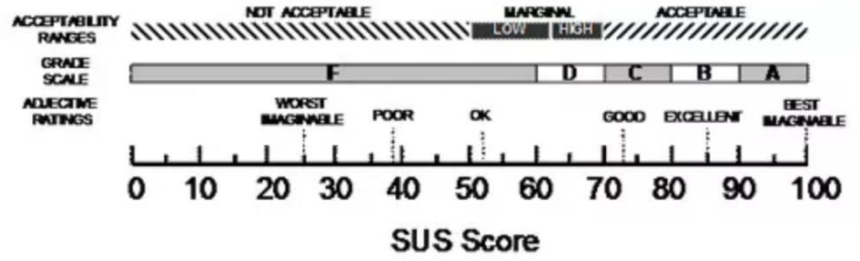


Figure 16 : Echelle d'acceptabilité (Bangor, Kortum et Miller, 2008) utilisée dans l'interprétation de notre résultat au SUS.

III.4 Discussion

III.4.1 Baisses de performance

Notons d'abord que les légères baisses de performance de l'interface adaptative par rapport à la version statique (-0,1 sur le score et +0,1 sur l'hésitation) ne sont probablement pas des différences statistiques significatives étant donné notre petit échantillon. Il s'agit probablement de fluctuations aléatoires non imputables à l'apport du système de raccourcis. Nous n'avons pas mené les tests de puissance statistique permettant de vérifier cela par manque de maîtrise du sujet.
Pour autant, nous pouvons expliquer ces tendances par une potentielle erreur dans notre protocole : dès le briefing nous avons expliqué que les testeurs allaient évaluer le système de raccourcis (en précisant tout de même que leur seul objectif était l'efficacité).
Ainsi, il est possible que certains de nos amis utilisateurs se soient forcés à utiliser les raccourcis pour nous fournir des résultats plus positifs, détournant ainsi leur attention du cœur de la tâche : recopier des modèles.

Ceci nous amène à un deuxième point : l'adaptation n'a pas été utile pour cette tâche.
En effet, ces raccourcis devaient surtout représenter un gain de temps sur les trajets œil et souris vers les boutons. Or, après nos tests de calibration et l'observation de notre expérience, nous nous sommes rendus compte que la principale difficulté d'Interfia consistait en cette tâche fatigante pour les yeux par la comparaison constante des PIXELS, et non pas dans la sélection des boutons.

La moindre importance de la sélection des boutons dans la tâche globale est en partie due à la faible diversité des fonctions utilisées : sur 26 des séquences de jeu analysées, 17 n'ont jamais utilisé la fonction Selec, et donc n'ont jamais pu utiliser les fonctions Copier, Coller et Suppr. Cela représente 4 fonctions sur 8 : pour la majorité des testeurs les 4 raccourcis étaient toujours les 4 premières fonctions, à savoir Peindre, Ligne, Gomme et Crayon.
Ce n'est pas exactement un problème d'utilité de ces fonctions puisque nous avions confirmé leur efficacité sur plusieurs modèles et dans plusieurs situations mais encore un problème de protocole, notamment dû à la contrainte de temps que nous avions imposé pour ne pas retenir nos camarades trop longtemps. En fait, les testeurs n'ont eu aucun temps d'exploration du logiciel et ont dû, juste après la démonstration, commencer à utiliser Interfia sous la pression du minuteur de 17 minutes, sans avoir le temps d'apprendre à utiliser ces 4 fonctions plus complexes et moins intuitives.
Nous aurions pu obtenir une plus grande diversité des fonctions utilisées en proposant un temps d'exploration, de prise en main, non évalué dans les données avant les séquences de test ; ou en modifiant complètement l'expérience et en demandant de compléter en un minimum de clics plutôt que dans un temps imparti.

III.4.2 Echantillonnage et représentativité

Il est aussi important d'observer que les résultats obtenus ne peuvent être représentatifs que d'une population d'étudiants en premières années en université de sciences et technologies en filières Informatique, Mathématiques, Physique et Chimie. Ce sont notamment des filières qui accueillent une majorité d'étudiants très habitués aux interfaces graphiques sur ordinateur et aux habitudes trop ancrées peuvent rendre tout à fait inutile un tel système d'adaptation d'interface. De plus, l'effectif de 6 et 7 personnes par groupe est faible (plusieurs travaux recommandent au minimum 8 personnes par groupe).
Ainsi, l'échantillon de testeurs n'était pas représentatif de la population globale à laquelle nous aurions souhaité pouvoir appliquer notre problématique, et nous pensons notamment que ce système pourrait obtenir de meilleurs résultats chez des utilisateurs moins à l'aise avec les logiciels sur ordinateur.

L'échantillon idéal aurait donc été composé de personnes de tous âges, plus ou moins habituées à l'utilisation d'un ordinateur, et donc avec des vitesses d'apprentissage pouvant varier d'une personne à l'autre. Il aurait également été beaucoup plus fourni et de multiples séances de tests auraient été menées.
Egalement, s'il avait été plus fourni, il aurait été possible de le répartir en un carré latin parfait, c'est-à-dire qu'il aurait été divisé en quatre groupes commençant respectivement par l'interface statique à dessins pairs, l'interface statique à dessins impairs, l'interface adaptative à dessins pairs et l'interface adaptative à dessins impairs, en enchaînant lors de la deuxième session sur l'interface opposée avec l'autre chaîne de dessins.

III.4.3 Un premier cycle de conception

Enfin, dans notre démarche de CCU, les traitements des données de cette expérience clôturent un premier cycle de conception duquel nous pouvons tirer plusieurs conclusions pour améliorer les prochains essais logiciels. Un point intéressant à discuter est la complexité de notre intelligence artificielle. Nous l'avions conçue très simple par contraintes techniques mais aussi par crainte que l'utilisateur soit perturbé par des adaptations qu'il ne sache comprendrait pas, effectuées par une IA plus complexe.
Lors du traitement des 28 items des 13 questionnaires AttrakDiff, l'un a reçu la même réponse par nos 13 testeurs :

Incontrôlable							X	Maîtrisable

Figure 17 : Interfia est maîtrisable d'après l'ensemble des testeurs.

Un autre a reçu un score moyen de 2,15, le plaçant approximativement ici (en haut sont les valeurs de score pour cet item correspondant aux réponses possibles) :

	3	2	1	0	-1	-2	-3	
Simple	X							Compliqué

Figure 18 : Un score moyen de 2,15 pour la simplicité d'Interfia.

Nous en concluons que l'IA que nous avons proposé n'a absolument pas perturbé les utilisateurs, et nous pouvons donc retenir une piste d'amélioration pour Interfia: complexifier l'adaptation pour la rendre plus pertinente et plus utile.

Conclusion

Essayons d'abord de conclure par rapport à notre problématique scientifique première: l'interface adaptative n'est peut-être pas une bonne piste pour adapter la machine à l'homme, du moins pour des utilisateurs habiles avec ces technologies. La réponse est évidemment à contraster puisque la validité de nos résultats est limitée par les problèmes que nous avons discutés et qu'on peut espérer trouver des situations et des logiciels où l'importance de la navigation entre les différentes fonctions serait bien plus élevée, et donc cette proposition d'adaptation serait plus utile et pertinente, apportant un vrai soutien à l'utilisateur.

Nous inscrire dans une démarche de CCU nous a permis de mieux préparer cette expérience scientifique et mieux concevoir le logiciel qui en a été le support. Cependant, un seul cycle de conception n'a pas été suffisant pour obtenir un système très satisfaisant et cela fait partie des limites de la CCU qui demande temps et réitérations.

Enfin, ce projet a été très formateur pour nous dans beaucoup de domaines : la conception informatique et le développement informatique, malgré le fait qu'on était très éloignés de conditions de travail en entreprise, le traitement de données, le travail en binôme et la présentation de projet orale et écrite.

Bibliographie et sitographie

SENACH Bernard, *Evaluation ergonomique des interfaces homme-machine : une revue de littérature*, INRIA, 1990

BASTIEN Christian, *Validation de critères ergonomiques pour l'évaluation d'interfaces utilisateurs*, INRIA, 1991

VALENTIN Anette, LANCRY Alain et LEMARCHAND Claude, *La construction des échantillons dans la conception ergonomique de produits logiciel pour le grand public. Quel quantitatif pour les études qualitatives ?* , 2010
https://www.cairn.info/revue-le-travail-humain-2010-3-page-261.htm?1=1&DocId=420334&hits=12217+11759+11736+11658+11389+11072+10665+10545+5340+5002+4447+3936+3898+3823+3786+3736+3476+3374+3296+3186+2732+2634+2545+2179+1950+1943+1526+1478+1127+1086+1028+964+928+692+522+449+406+367+339+218+79+11+

BROOKE John, *SUS – A quick and dirty usability scale*, 1996
http://www.usabilitynet.org/trump/documents/Suschapt.doc

BANGOR Aaron, MILLER James, KORTUM Philip, *Determining What Individual SUS Scores Mean : Adding an Adjective Rating Scale*, 2009
http://uxpajournal.org/determining-what-individual-sus-scores-mean-adding-an-adjective-rating-scale/

LALLEMAND Carine, KOENIG Vincent, GRONIER Gronier, MARTIN Romain, *Création et validation d'une version française du questionnaire AttrakDiff pour l'évaluation de l'expérience utilisateur des systèmes interactifs*, 2015, Revue Européenne de Psychologie appliquée

BELLOUTI Samir, *Le développement itératif*, 2007
https://bellouti.wordpress.com/2007/09/24/le-developpement-iteratif/

Documentation de tkinter
http://tkinter.fdex.eu/

Official Python 3.6.5 documentation
https://docs.python.org/3/

PyInstaller Manual
https://pyinstaller.readthedocs.io/en/v3.3.1/

Annexe

Table des matières de l'annexe

Table des matières de l'annexe 27
A. Formulaire de consentement éclairé 28
B. Modèles de dessin pour Interfia 29
B.1 Série impaire 29
B.2 Série paire 32
C. Questionnaire AttrakDiff 35
D. Questionnaire SUS 37
E. Fiche de déroulement test 38
F. Version plus professionnelle d'Interfia 39
G. Synthèse culturelle 40
H. Code d'Interfia 44
H.1 Vue d'Interfia 44
H.2 Contrôleur d'Interfia 52
H.3 Modèle d'Interfia 63
H.4 Paramètres d'Interfia 65

A. Formulaire de consentement éclairé

Formulaire de Consentement Libre, Eclairé et Exprès

Titre de l'expérience : Logiciel intelligent à interface adaptative

Equipe Interfia
CPBx – Cycle Préparatoire de Bordeaux – 351 cours de la Libération, bâtiment B6,
33400 Talence

Je certifie avoir donné mon accord pour participer à l'expérience **Logiciel Intelligent à interface adaptative**. J'accepte volontairement de participer à cette étude et je comprends que ma participation n'est pas obligatoire et que je peux stopper ma participation à tout moment sans avoir à me justifier ni encourir aucune responsabilité. Mon consentement ne décharge pas les organisateurs de la recherche de leurs responsabilités et je conserve tous mes droits garantis par la loi.

Au cours de cette expérience, j'accepte que soient recueillies des données sur mes réponses. Je comprends que les informations recueillies sont strictement confidentielles et à usage exclusif des investigateurs concernés.

J'ai été informé que mon identité n'apparaîtra dans aucun rapport ou publication et que toute information me concernant sera traitée de façon confidentielle. J'accepte que les données enregistrées à l'occasion de cette étude puissent être conservées dans une base de données et faire l'objet d'un traitement informatisé non nominatif par l'Unité de recherche. J'ai bien noté que le droit d'accès prévu par la loi « informatique et libertés » s'exerce à tout moment auprès de l'unité de recherche.

De plus, j'accepte que les organisateurs de la recherche utilisent et diffusent à titre gratuit et non exclusif mon image par des photographies et vidéos. Ces dernières pourront être exploitées et utilisées directement par les organisateurs, sous toutes formes et tous supports connus et inconnus à ce jour, dans le monde entier, sans limitation de durée, intégralement ou par extraits lors :

- de publications dans une revue, ouvrage ou journal scientifique portant sur l'objet de la recherche
- de diffusion interne et externe au sein du CPBx, Cycle Préparatoire de Bordeaux.

Les bénéficiaires de l'autorisation s'interdisent expressément de procéder à une exploitation des images susceptible de porter atteinte à la vie privée ou à la réputation, ni de les utiliser, dans tout support à caractère pornographique, raciste, xénophobe ou toute autre exploitation préjudiciable.

Je me reconnais être entièrement rempli de mes droits et je ne pourrai prétendre à aucune rémunération pour l'exploitation des droits visés aux présentes.

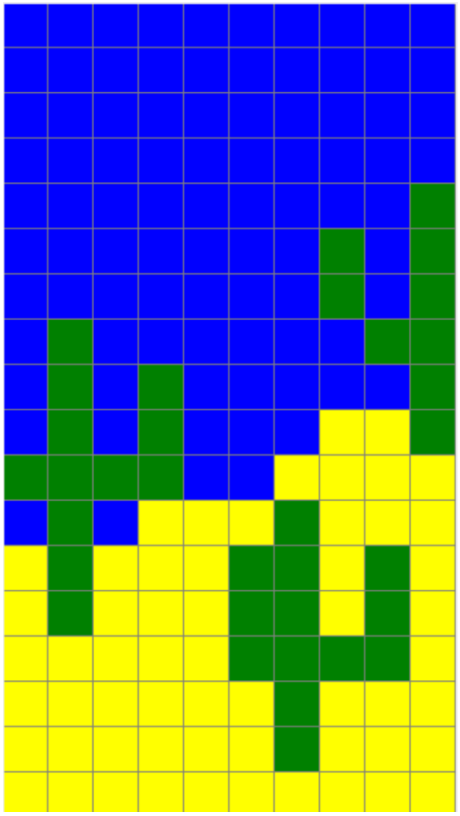
Date : .....
Nom du volontaire : …….
Signature du volontaire (précédée de la mention « lu et approuvé ») :

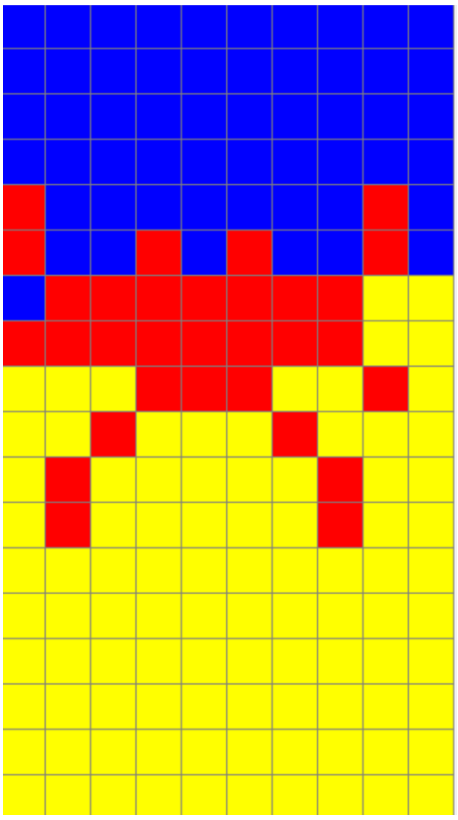
Nom de l'expérimentateur: .....

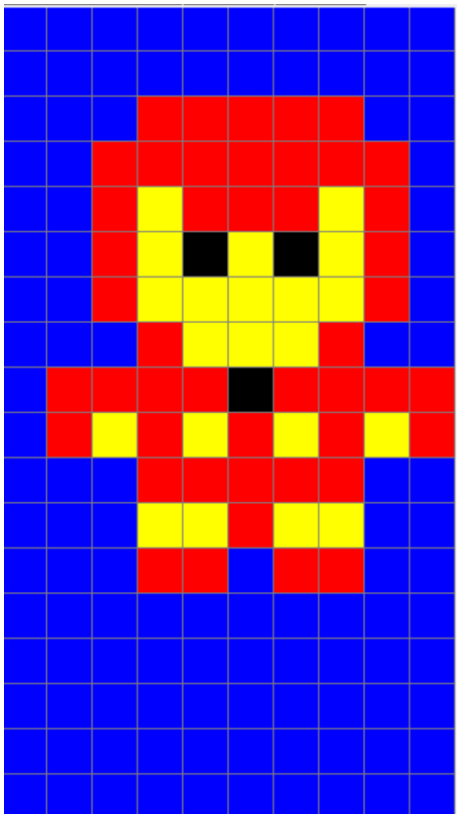
Signature de l'expérimentateur :

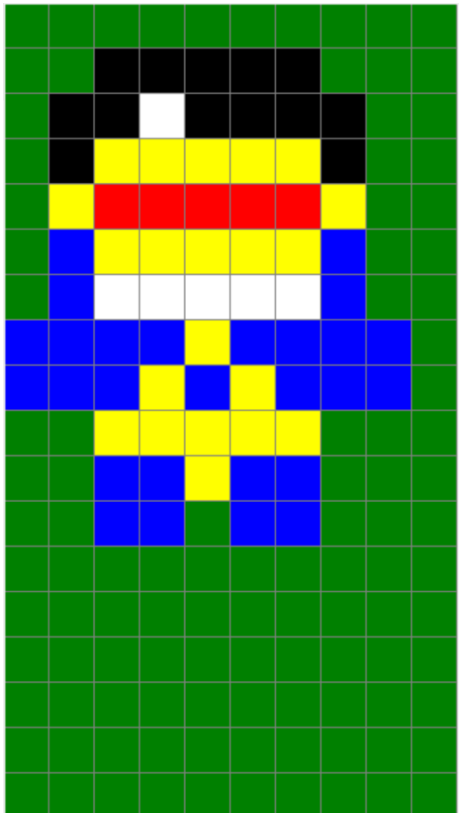
B. Modèles de dessin pour Interfia

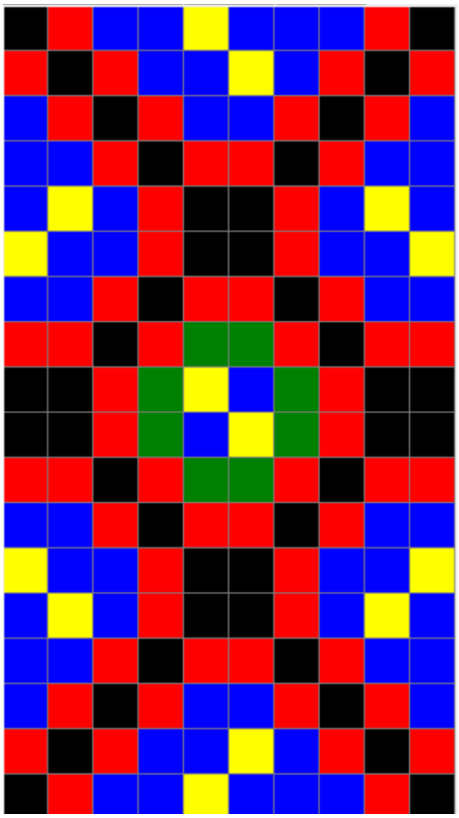
B.1 Série impaire

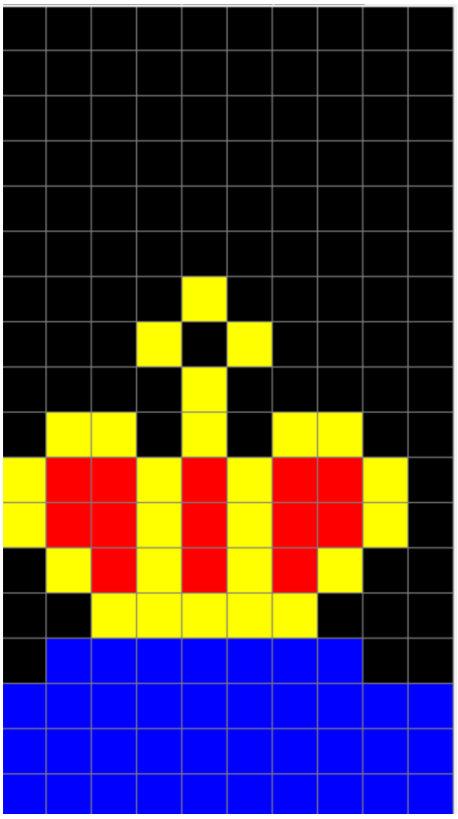


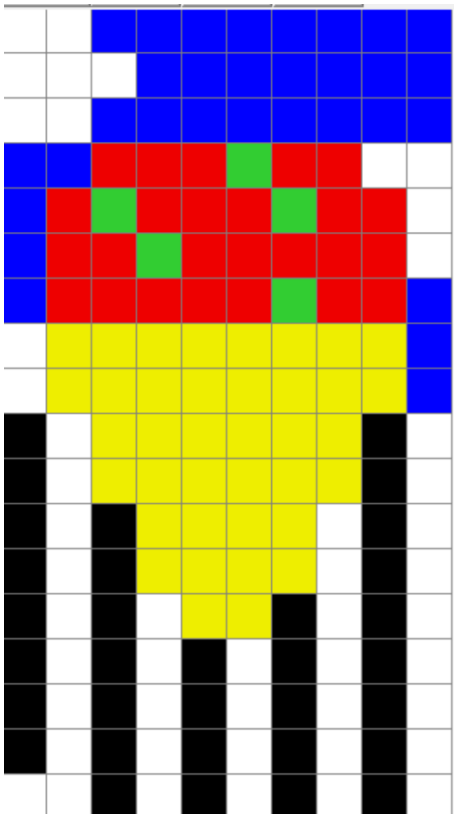
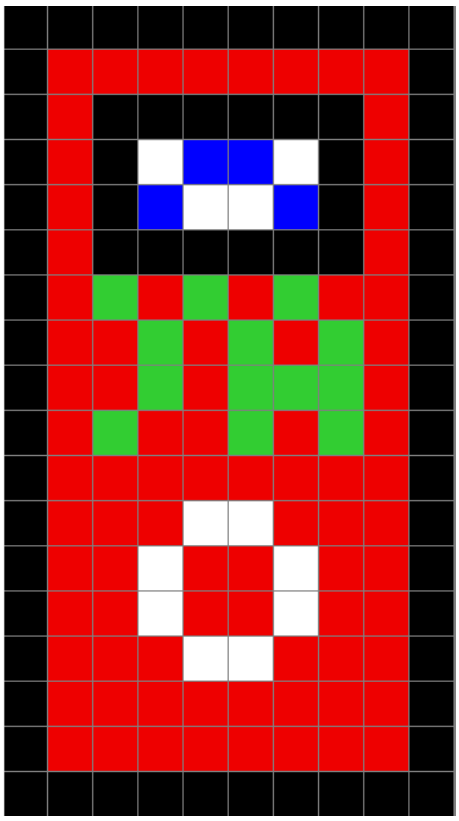
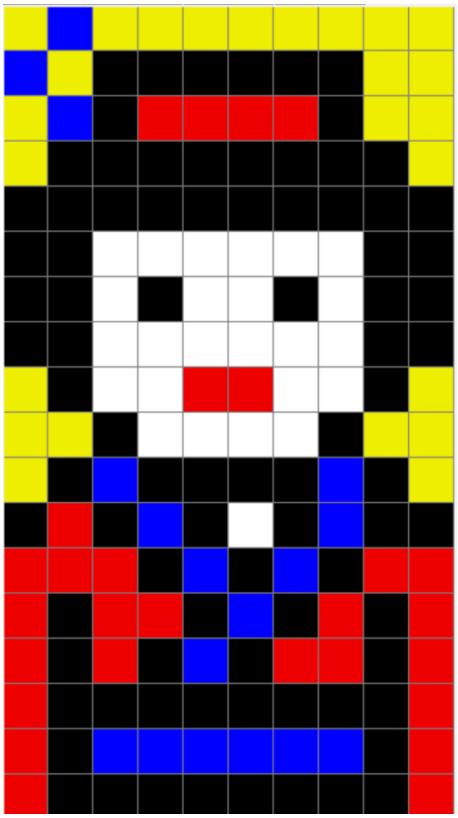
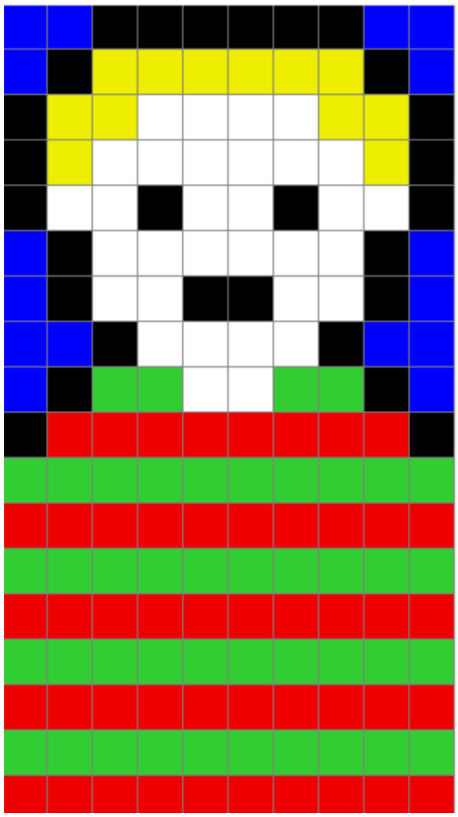
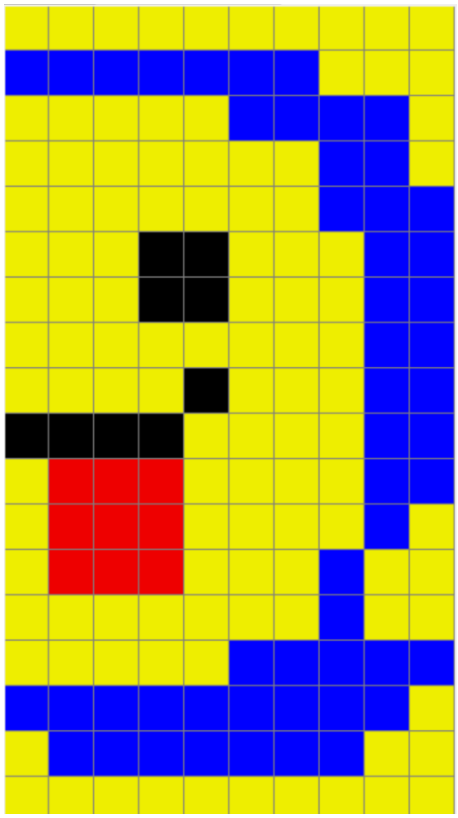
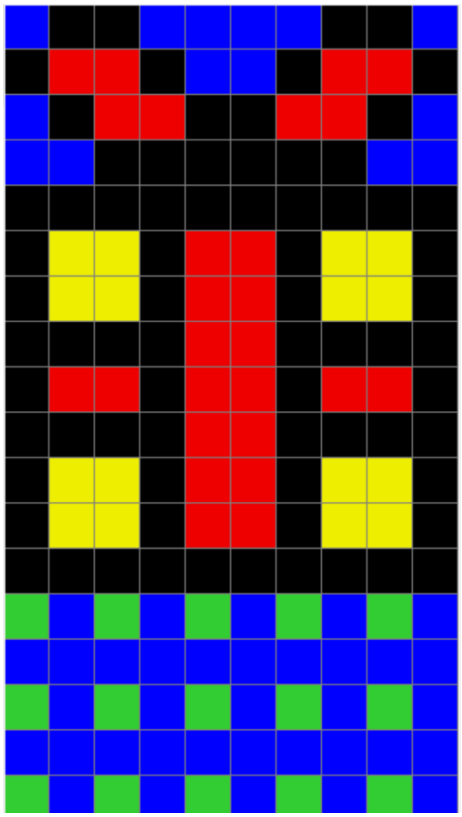




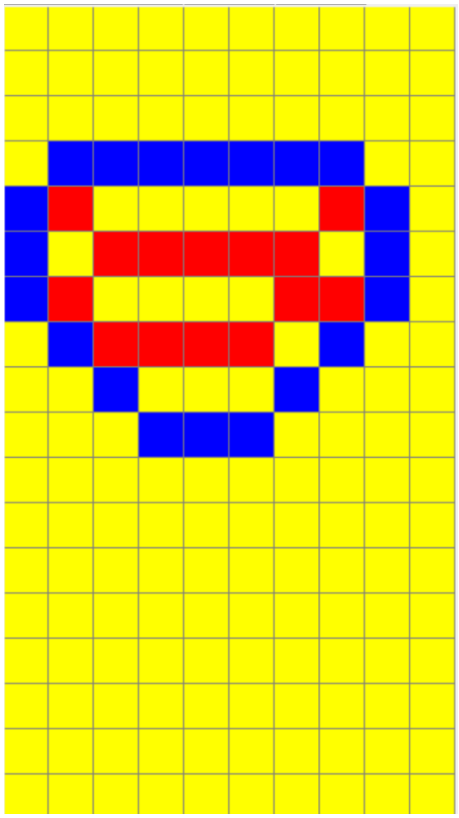


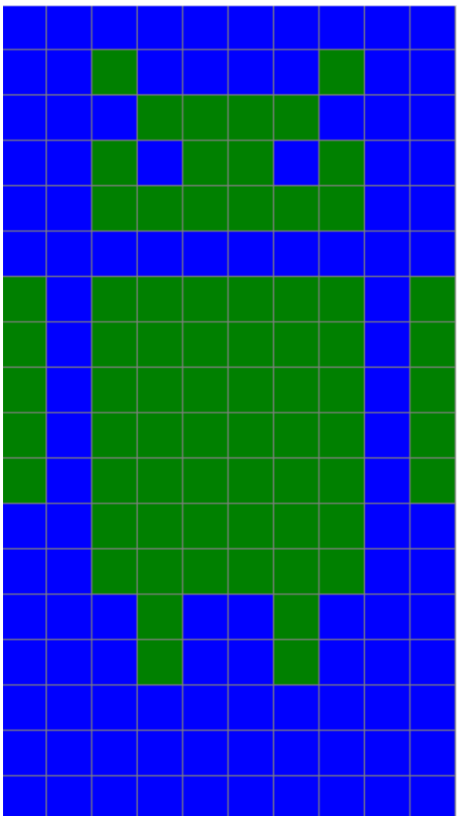


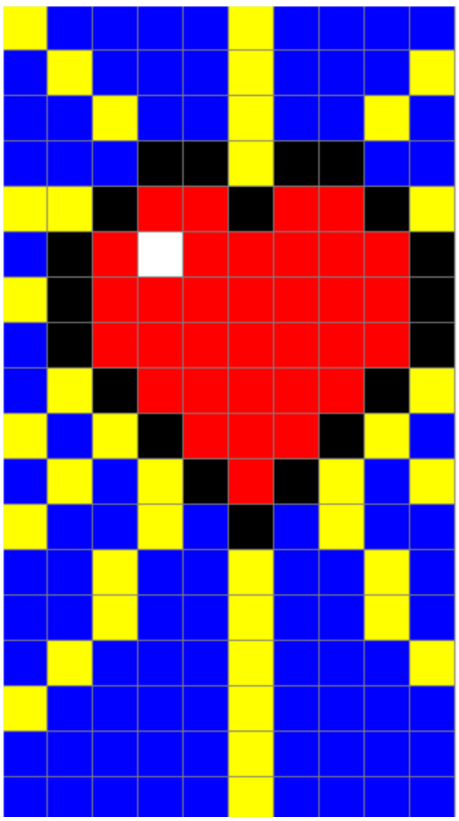


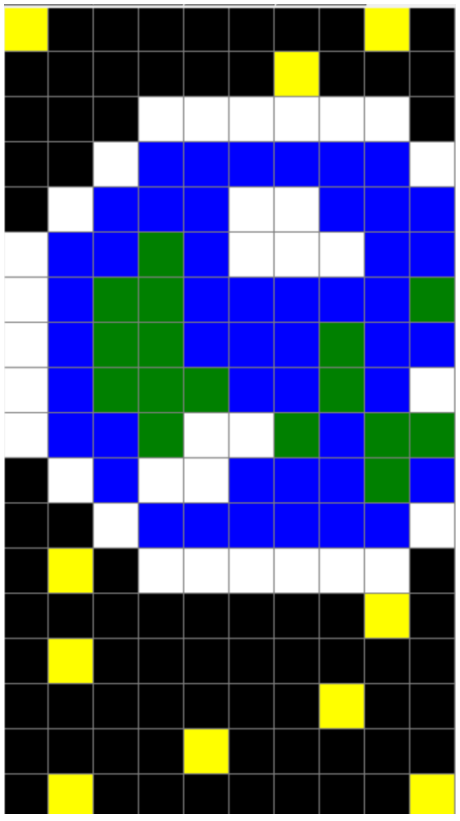







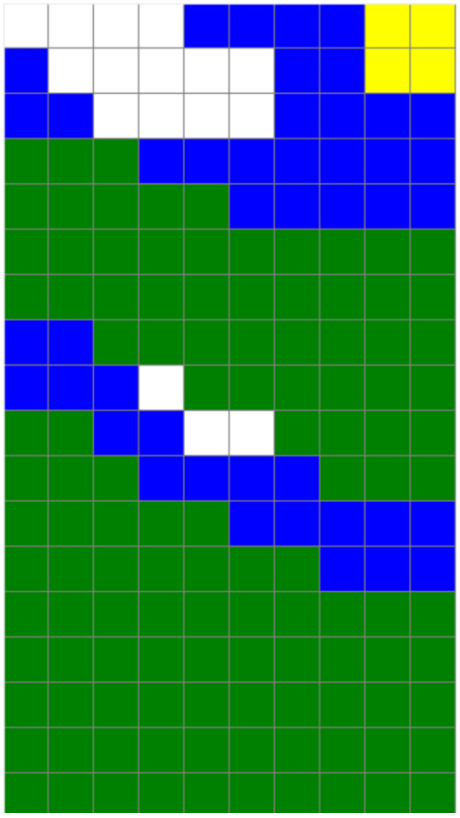
B.2 Série paire

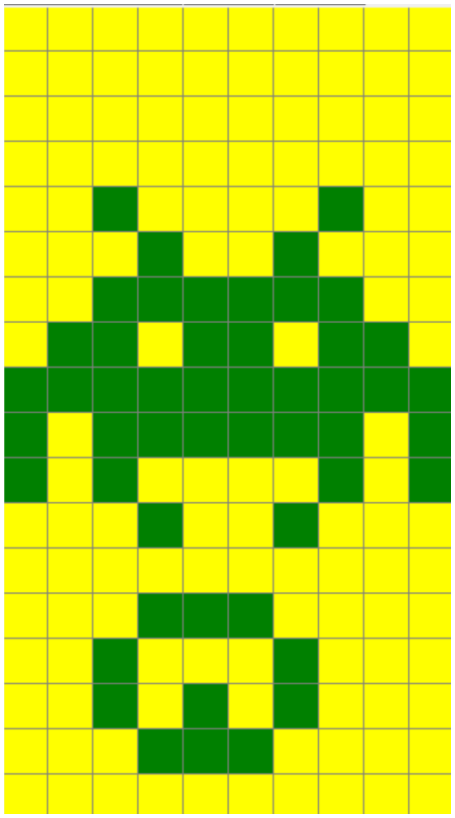
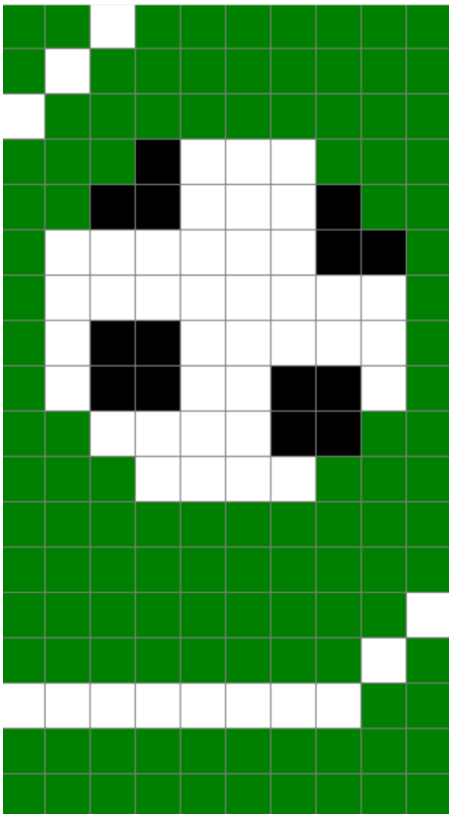
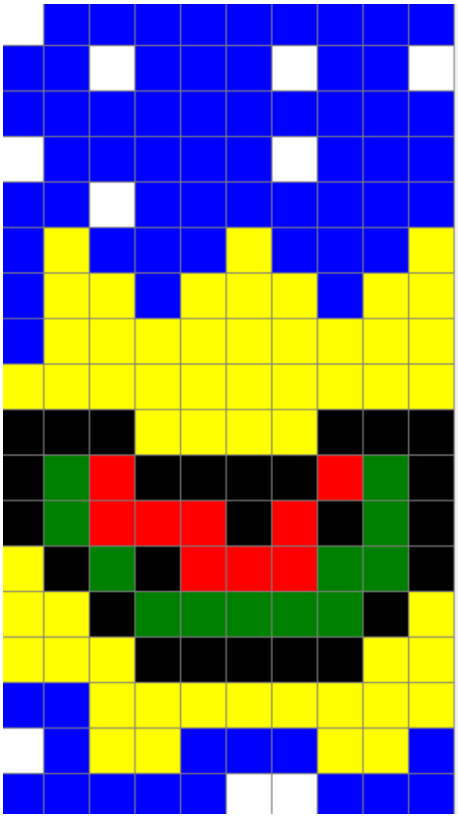
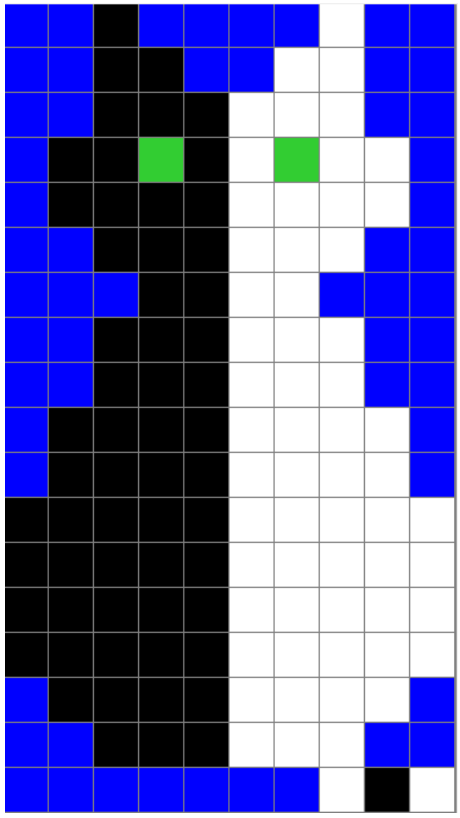
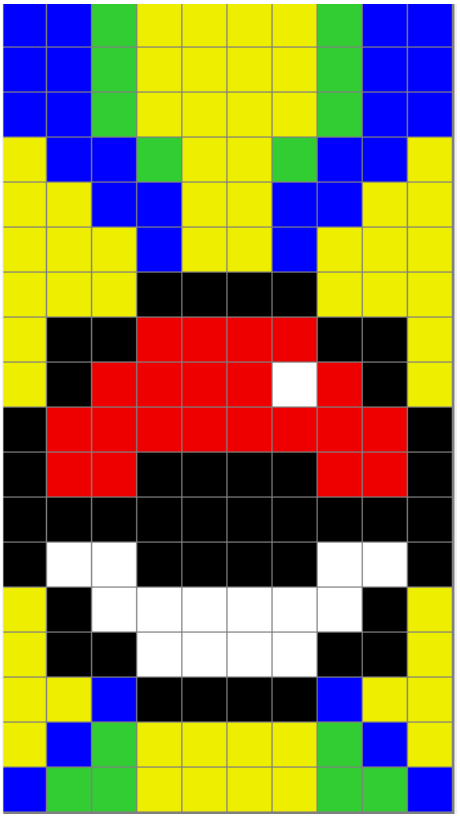
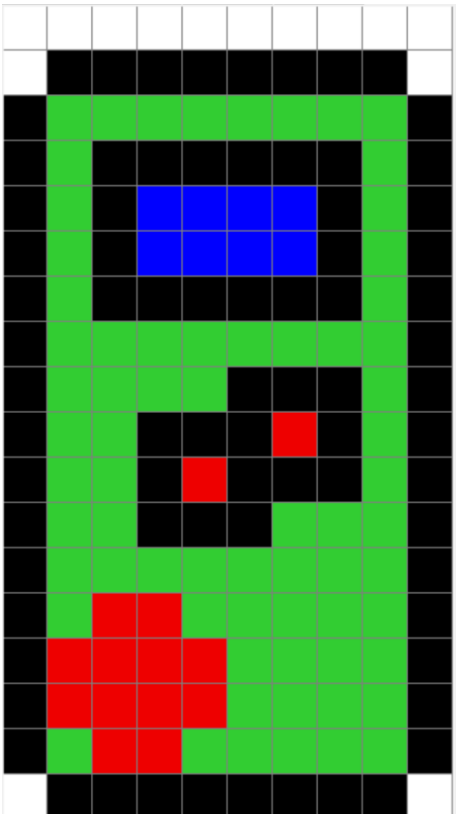


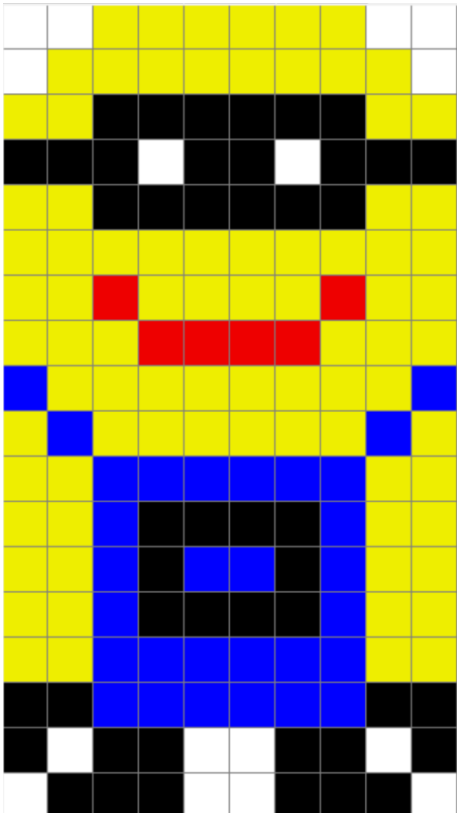
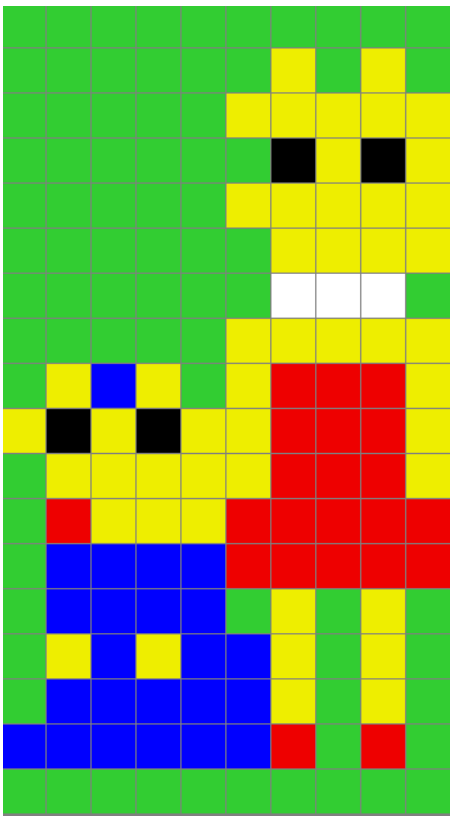
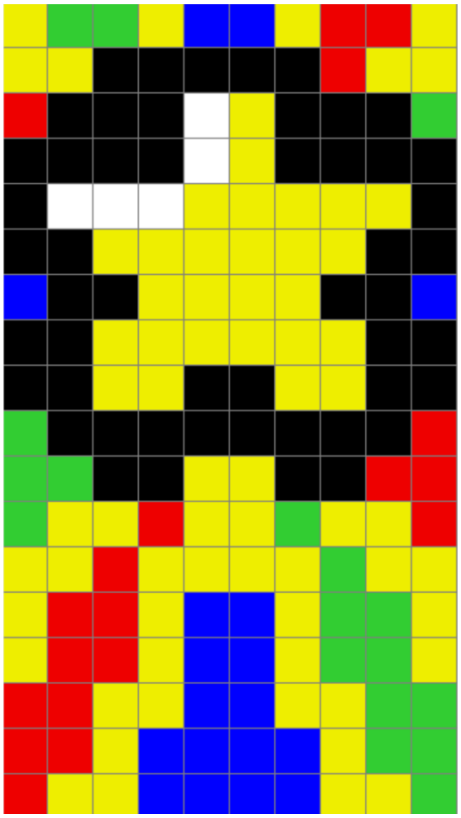
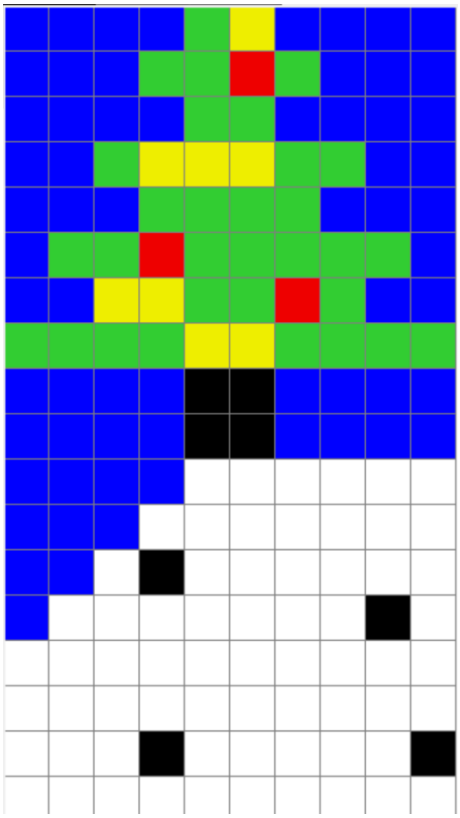
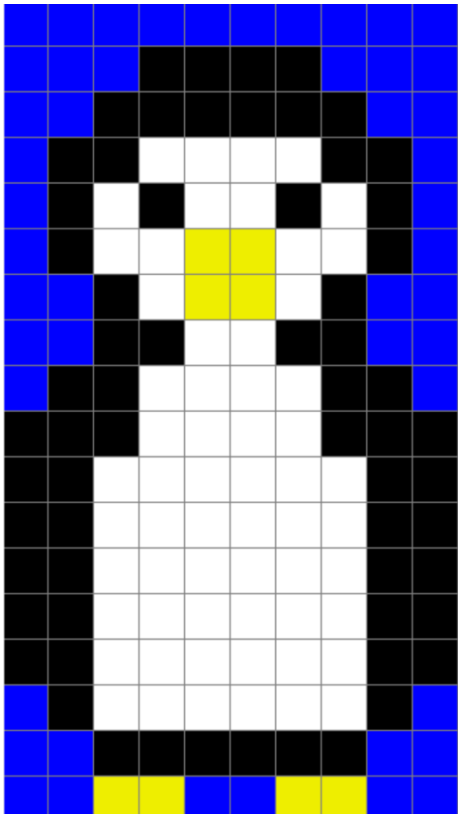
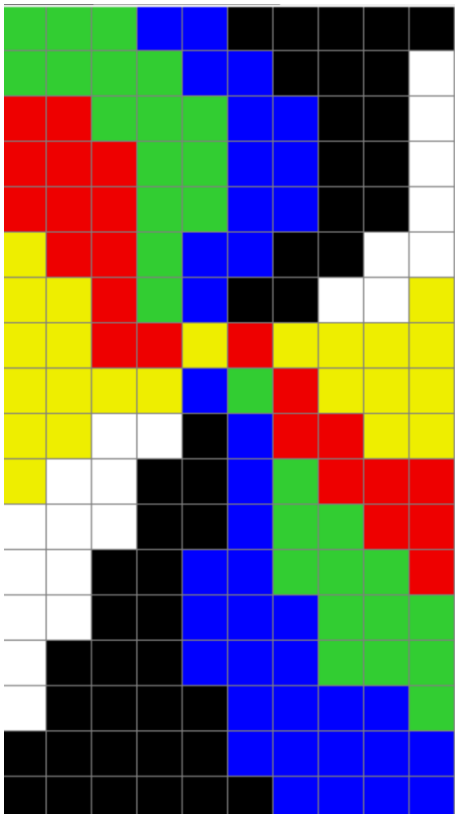








C. Questionnaire AttrakDiff

AttrakDiff – Passation – Utilisateur N° –Groupe N°

Consignes Questionnaire	**Dans le cadre de notre projet sur l'expérience utilisateur, nous souhaiterions évaluer vos impressions sur l'interface/service utilisé(e).** **Ce questionnaire se présente sous forme de paires de mots pour vous assister dans l'évaluation du système. Chaque paire représente des contrastes. Les échelons entre les deux extrémités vous permettent de décrire l'intensité de la qualité choisie.** **Répondez de manière spontanée même si vous avez l'impression que certains termes ne décrivent pas correctement l'interface/service utilisé(e). Dans ce cas, assurez-vous de donner tout de même une réponse. Gardez à l'esprit qu'il n'y a pas de bonne ou mauvaise réponse.** **Seul votre avis compte !**

AttrakDiff – Item 1/3 – Utilisateur N° – Groupe N°

Humain	○	○	○	○	○	○	○ Technique
M'isole	○	○	○	○	○	○	○ Me sociabilise
Plaisant	○	○	○	○	○	○	○ Déplaisant
Original	○	○	○	○	○	○	○ Conventionnel
Simple	○	○	○	○	○	○	○ Compliqué
Professionnel	○	○	○	○	○	○	○ Amateur
Laid	○	○	○	○	○	○	○ Beau
Pratique	○	○	○	○	○	○	○ Pas pratique
Agréable	○	○	○	○	○	○	○ Désagréable
Fastidieux	○	○	○	○	○	○	○ Efficace

AttrakDiff – Item 2/3 – Utilisateur N° – Groupe N°

De bon goût	○	○	○	○	○	○	○	De mauvais goût
Prévisible	○	○	○	○	○	○	○	Imprévisible
Bas de gamme	○	○	○	○	○	○	○	Haut de gamme
M'exclut	○	○	○	○	○	○	○	M'intègre
Me rapproche des autres	○	○	○	○	○	○	○	Me sépare des autres
Non présentable	○	○	○	○	○	○	○	Présentable
Rebutant	○	○	○	○	○	○	○	Attirant
Sans imagination	○	○	○	○	○	○	○	Créatif
Bon	○	○	○	○	○	○	○	Mauvais

AttrakDiff – Item 3/3 – Utilisateur N° – Groupe N°

Confus	○	○	○	○	○	○	○	Clair
Repoussant	○	○	○	○	○	○	○	Attrayant
Audacieux	○	○	○	○	○	○	○	Prudent
Novateur	○	○	○	○	○	○	○	Conservateur
Ennuyeux	○	○	○	○	○	○	○	Captivant
Peu exigeant	○	○	○	○	○	○	○	Challenging
Motivant	○	○	○	○	○	○	○	Décourageant
Nouveau	○	○	○	○	○	○	○	Commun
Incontrôlable	○	○	○	○	○	○	○	Maîtrisable

D. Questionnaire SUS

Poste :

SUS (System Usability Scale)
Evaluation de l'utilisabilité du sytème
de raccourcis de l'Interfia

	Pas du tout d'accord				Fortement d'accord
1. Je pense que j'aimerais utiliser ce système fréquemment					
	1	2	3	4	5
2. J'ai trouvé ce système inutilement complexe					
	1	2	3	4	5
3. J'ai pensé que ce système était facile à utiliser					
	1	2	3	4	5
4. Je pense que j'aurais besoin de l'aide d'un technicien pour pouvoir utiliser ce système					
	1	2	3	4	5
5. J'ai trouvé les différentes fonctions dans ce système étaient bien intégrées					
	1	2	3	4	5
6. J'ai pensé qu'il y avait trop d'incohérence dans ce système					
	1	2	3	4	5
7. J'imagine que la plupart des gens apprendraient à utiliser ce système très rapidement					
	1	2	3	4	5
8. J'ai trouvé ce système très lourd à utiliser					
	1	2	3	4	5
9. Je me suis senti en confiance en utilisant ce système					
	1	2	3	4	5
10. J'ai eu besoin d'apprendre beaucoup de choses avant de pouvoir démarrer avec ce système					
	1	2	3	4	5

E. Fiche de déroulement test

Déroulement de l'expérience

Accueil cordial (différé) afin d'éviter toute forme de pression envers les testeurs

Briefing :

- donner le formulaire de consentement éclairé
- but de l'expérience (préciser que chaque séquence dure 17 min)
- expliquer comment est le logiciel
- pendant ce temps, lancer les transferts USB sur les ordinateurs

Démonstration de toutes les fonctions et montrer comment le jeu progresse (5 min)

Session 1 (17 min)

Pause (8 min)

Session 2 (17 min)

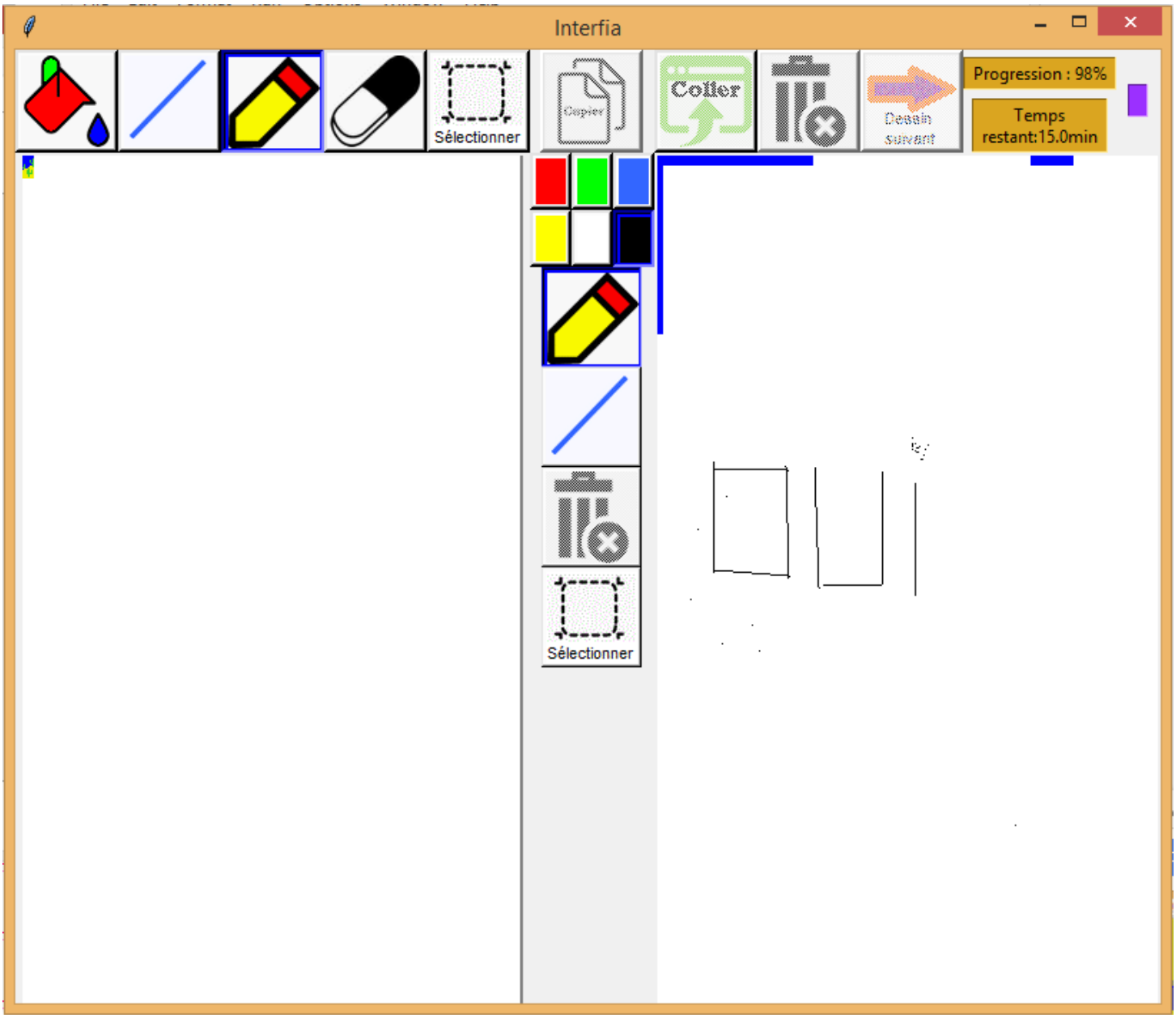
Questionnaires :

- dire qu'il n'y a pas de bonnes réponses, qu'on veut le ressenti des testeurs
- dire que le but est d'évaluer la plus value des raccourcis de la version adaptative de l'interface

Débriefing :

- remercier les testeurs
- dire qu'ils peuvent partir s'ils veulent, ou peuvent rester échanger avec nous

F. Version plus professionnelle d'Interfia



G. Synthèse culturelle

Synthèse culturelle

Repenser la pénible statique de l'interface logiciel

par Alice DRAILLARD et Hugo FOURNIER

Qui ne s'est jamais arraché les cheveux devant son écran d'ordinateur, frustré par un logiciel à l'interface mal conçue ?
Le rapide développement des Nouvelles Technologies de l'Informatique et de la Communication (NTIC) aussi bien dans la vie professionnelle que privée a, trop rapide, entraîné la prolifération de ces situations subies. Trop souvent, ces interfaces homme-machine (IHM) suscitent l'incompréhension de l'utilisateur.
Quelles sont les solutions à ces problèmes d'ergonomie numérique ?
Nous dresserons un rapide constat des problèmes sociétaux liés aux interfaces classiques, proposerons une piste de solutions et questionnerons sa pertinence.

La révolution numérique, comme on l'appelle, continue à s'étendre dans le monde du travail et pas seulement par les ordinateurs, si bien qu'en 2015, 55 % de la population active utilise un micro ordinateur au travail. Cette transformation sociétale s'accompagne de l'utilisation croissante de logiciels plus ou moins complexes qui nécessitent toujours un temps d'adaptation, d'apprentissage, et peuvent générer une « surcharge informationnelle et communicationnelle qui peut être contre-productive » [B. Mettling, 2015].
Egalement, l'adaptabilité à ces nouveaux outils variant beaucoup entre chaque individu, les moins rapides se retrouvent discriminés. Les sourds, les aveugles et les mal-voyants par exemple souffrent de ces logiciels aux interfaces non adaptées à leur condition, si bien qu'ils sont écartés de beaucoup d'emplois demandant une expertise logiciel [J.L. Metzger et C. Barril, 2004]. Benoît Thieffry, dans *L'accessibilité aux nouvelles technologies de l'information et de la communication et l'accessibilité en terme d'aménagement du territoire*, compare cette situation de non accessibilité numérique à la non accessibilité physique de beaucoup de lieux avant que l'État ne réglemente sur ce sujet (rampes d'accès, etc...).
Il est fréquent d'entendre son voisin de bureau se plaindre de ne pas trouver ce qu'il cherche dans son interface logiciel, et pourtant ces problèmes d'ergonomie, qui représentent une pénibilité quotidienne, semblent être peu pris en compte. En effet, peu de travaux universitaires, peu d'études d'opinions cherchent cette corrélation entre pénibilité au travail et frustration due à l'utilisation de logiciels mal interfacés; ce n'est pas non plus un critère pris en compte par la loi sur la pénibilité [Direction de l'information légale et administrative, 2018].

Les interfaces classiques, statiques et configurables, ternissent et austérisent donc le lien entre l'individu et son travail. Il y a une confrontation entre un utilisateur et une interface à l'immobilité mortuaire [A. Bruzan, R. Mocan et R. Vicovanu, 2015] là où on pourrait rechercher une association.

La conception centrée utilisateur (CCU) cherche la minimisation d'embarras tels qu'une mauvaise présentation et des procédures rigides en facilitant la prise en main dudit logiciel, aboutissant ainsi à une amélioration de l'expérience. La très complète étude *La construction des échantillons dans la conception ergonomique de produits logiciel pour le grand public. Quel quantitatif pour les études qualitatives ?*, par A.Valentin, A.Lancry et C.Lemarchand est un exemple d'une telle conception : des tests répétés sur des échantillons de population de taille moyenne mais diversifiés permettent des retours positifs et des rapports d'erreur essentiels, ainsi que des questionnements réguliers tout au long de la conception la dirigent vers une bonne ergonomie du système.
Cependant, d'autres facteurs jouent dans le ressenti d'une personne face à un écran. Le logiciel peut également toucher l'homme par son graphisme : la vue de son interface à l'aide du flat design, initiée par Apple, offre une épuration du logiciel agréable à l'oeil . Sont aussi à prendre en compte les sensations qui résultent de l'usage de cette interface ; l'interaction, le fait de créer, de changer l'interface est à la fois initiée et suivie par des émotions [Studio Catepeli, 2013] .
Le ressenti face au graphisme amène à un action qui peut générer d'autres sentiments : c'est l'interface naturelle (NUI), dans laquelle les intuitions humaines sont prises en compte [Julien Neuville, 2017] : les interactions deviennent toujours plus subtiles pour faire oublier la machine devant soi, s'humanisant toujours plus dans ses interactions ; on parle alors d'affordance car l'interface suggère sa propre utilisation en guidant intuitivement l'utilisateur vers l'action.
Ainsi, un parallèle est fait entre interface mouvante et mouvement humain : tous deux s'adaptent l'un à l'autre ; l'interface en se rapprochant de l'humain, et plus généralement de la nature, et ce dernier en utilisant l'interface : c'est l'*aiolomètis*, l'intelligence en mouvement [A. Bruzan, R. Mocan, R. Vicovanu, 2015]. Le mouvement et les modulations (telles qu'elles sont exprimées ici par l'interface) s'expriment à travers les perceptions de l'humain : le phénomène de compréhension n'est plus uniquement visuel et linguistique mais immédiat et purement physique, dans l'action [Stefan Kristensen, 2006].
L'interface dépasse sa condition statique pour devenir unique à chacun, s'adaptant à chaque fois d'une manière différente ; on peut dire qu'en créant une interface adaptative on "associe une personne à une machine" [Studio Catepeli, 2013], ou dans ce cas précis, on associe une personne à une interface. Ainsi, la technologie devient invisible pour l'utilisateur, qui ne la discerne que par ses ressentis et les émotions qu'il éprouve lorsqu'il en fait usage.

Cependant l'interface écran a ses limites (85 % des plus de 50 ans ont des problèmes de vue [B. Thieffry, 2003]) et ses contestataires, comme les designers du NoUI, interface sans écran (par exemple, le coffre d'une voiture qui s'ouvre lorsqu'on passe le pied sous celle-ci) [Julien Neuville, 2017]. Quid d'alternatives ?

Colin Schmidt imagine en 2007 une IHM perfectionnée, alliant une intelligence artificielle (IA) très avancée à l'IHM pour supprimer les pré-requis d'expertise logiciel, faciliter et rendre plus agréable l'expérience de l'utilisateur, ce qui réglerait les problèmes évoqués précédemment. Dans cette optique, plus besoin de reconfigurer l'interface puisqu'elle se reconfigure selon les données implicitement rentrées par l'utilisateur lors de l'échange homme-machine.
Mais si on perd le contrôle sur ce qui était l'élément de contrôle de l'Homme sur la Machine, à savoir l'interface configurable, la Machine est-elle victorieuse et l'Homme a t-il signé sa perte ?
Cette « personne numérique » humanisée jusqu'à l'extrême, combinant les avancées de la cognitique, IHM, IA, informatique, robotique, qui nous aiderait comme un employé chargé de relation clientèle n'est pas un Autre qui nous rend service. En fait, en fournissant des données à l'IA, nous nous aiderions nous-même : le client est à la fois client et prestataire de service, c'est ce que l'auteur appelle la serviduction.

Dans cette révolution numérique, on voit que l'interface, en tant qu'élément essentiel dans l'utilisation des technologies, est de plus en plus étudiée sous tous ses angles dans le but de faciliter et supprimer les frustrations et facteurs de discrimination liés à leur utilisation. Selon des démarches nouvelles de prise en compte du facteur humain, des solutions porteuses d'avenir et qui procurent un dynamisme nouveau au logiciel sont proposées. Il s'agit de rester vigilant par cette même démarche expérimentale qui semble être la plus porteuse de résultats : questionnons, essayons, avisons.

Corpus :

A.Valentin, A.Lancry et C.Lemarchand, *La construction des échantillons dans la conception ergonomique de produits logiciel pour le grand public. Quel quantitatif pour les études qualitatives ?* , 2010, pages 261 à 290
https://www.cairn.info/revue-le-travail-humain-2010-3-page-261.htm?1=1&DocId=420334&

Bernard Morand, *Le logiciel, sujet et objet de la norme*, 2007, pages 41 à 51
https://www.cairn.info/revue-droit-et-societe-2007-1-page-41.htm?1=1&DocId=247996&

Colin Schmidt, *Des automates intelligents ?* , 2007, pages 115 à 123
http://www.persee.fr/doc/colan_0336-1500_2007_num_151_1_4642?q=homme+face+%C3%A0+la+machine+interface

Benoît Thieffry, *L'accessibilité aux nouvelles technologies de l'information et de la communication et l'accessibilité en terme d'aménagement du territoire*, 2003, pages 54 à 60
http://www.persee.fr/doc/htn_0018-439x_2003_num_3_1_2848?q=logiciel+p%C3%A9nibilit%C3%A9+travail+utilisation

Jean-Luc Metzger et Claudia Barril, *L'insertion professionnelle des travailleurs aveugles et sourds : les paradoxes du changement technico-organisationnel*, 2004, pages 63 à 86
https://www.cairn.info/revue-francaise-des-affaires-sociales-2004-3-page-63.htm?1=1&DocId=364758&

Stefan Kristensen, *Maurice Merleau-Ponty, une esthétique du mouvement*, 2006, pages 123 à 146
https://www.cairn.info/revue-archives-de-philosophie-2006-1-page-123.htm

Adinel Bruzan, Raluca Mocan et Roxana Vicocanu, *Penser le mouvement. Littérature, arts et philosophie*, 2015
http://www.fabula.org/colloques/document2603.php

Un rapport par Bruno Mettling, *Transformation numérique et vie au travail*, 2015, pages 1 à 7
http://www.ladocumentationfrancaise.fr/var/storage/rapports-publics/154000646.pdf

Julien Neuville, *Interface invisible, NOUI, NUI … l'UI menacée ?* , 2017
http://www.usabilis.com/interfaces-invisibles-noui-nui/

Studio Catepeli, *UX & UI sont dans un bateau…* , 2013
http://catepeli.com/blog/ux-ui-sont-dans-un-bateau/

Direction de l'information légale et administrative, *Compte professionnel de prévention*, Critères de pénibilité, 2018
https://www.service-public.fr/particuliers/vosdroits/F15504

H. Code d'Interfia

Voici la majorité du code d'une des versions d'Interfia utilisée pour le test, dont nous n'avons coupé que les modèles et les fonctions utiles à leur implémentation. Les # marquent les débuts de commentaires extérieures au code mais utiles à la compréhension.

Vueinterfia.py

```
#Vue interfia
from tkinter import *
import time

from Parametresinterfia import *

class Vue():
    def __init__(self):
        self.fenetre=Tk()
        self.fenetre.title("Interfia")

        self.progress=StringVar()

        self.raccourcis=[0,0,0,0,0,0,0,0,0]
        self.preraccourcis=[0,0,0,0,0,0,0,0,0]

        self.peindreracc=-1
        self.ligneracc=-1
        self.crayonracc=-1
        self.gommeracc=-1
        self.selecracc=-1
        self.collerracc=-1
        self.copierracc=-1
        self.supprracc=-1
        self.suivracc=-1
#marque où sont les fonctions sont en raccourcis pour pouvour mettre à jour leur état désactivé ou activé

        self.horloge()

    def horloge(self):

        self.fenetre.after(1000,self.tictac)

    def loop(self):
        self.fenetre.mainloop()

    def tictac(self):
        self.time=self.time+1
        self.temps.set("Temps restant:" + str((Timer/(1000)-self.time)//60)+ "min")
```

```
        self.horloge()

    def genicones(self):

        self.temps=StringVar()
        self.time=0
#on stocke les icônes adéquates dans des variables utilisables en s'adaptant à l'une des deux échelles prévues définie dans le fichier de paramètres
        if echelle == 1:
            self.peindreIMG = PhotoImage(file="peindreIMG1.gif") #Icône 64x64 à l'échelle 1 et 100x100 echelle 1.6
            self.ligneIMG = PhotoImage(file="ligneIMG1.gif")
            self.crayonIMG = PhotoImage(file="crayonIMG1.gif")
            self.gommeIMG = PhotoImage(file="gommeIMG1.gif")
            self.ctrlzIMG = PhotoImage(file="ctrlzIMG1.gif")
            self.selecIMG = PhotoImage(file="selectionnerIMG1.gif")
            self.ctrlcIMG = PhotoImage(file="ctrlcIMG1.gif")
            self.ctrlvIMG = PhotoImage(file="ctrlvIMG1.gif")
            self.supprIMG = PhotoImage(file="supprIMG1.gif")
            self.rougeIMG = PhotoImage(file="rougeIMG1.gif")  #21x32 à l'échelle 1 et 33x50 echelle 1.6
            self.vertIMG = PhotoImage(file="vertIMG1.gif")
            self.bleuIMG = PhotoImage(file="bleuIMG1.gif")
            self.jauneIMG = PhotoImage(file="jauneIMG1.gif")
            self.blancIMG = PhotoImage(file="blancIMG1.gif")
            self.noirIMG = PhotoImage(file="noirIMG1.gif")

        elif echelle == 1.6:
            self.peindreIMG = PhotoImage(file="peindreIMG1,6.gif")
            self.ligneIMG = PhotoImage(file="ligneIMG1,6.gif")
            self.crayonIMG = PhotoImage(file="crayonIMG1,6.gif")
            self.gommeIMG = PhotoImage(file="gommeIMG1,6.gif")
            self.ctrlzIMG = PhotoImage(file="ctrlzIMG1,6.gif")
            self.selecIMG = PhotoImage(file="selectionnerIMG1,6.gif")
            self.ctrlcIMG = PhotoImage(file="ctrlcIMG1,6.gif")
            self.ctrlvIMG = PhotoImage(file="ctrlvIMG1,6.gif")
            self.supprIMG = PhotoImage(file="supprIMG1,6.gif")
            self.rougeIMG = PhotoImage(file="rougeIMG1,6.gif")
            self.vertIMG = PhotoImage(file="vertIMG1,6.gif")
            self.bleuIMG = PhotoImage(file="bleuIMG1,6.gif")
            self.jauneIMG = PhotoImage(file="jauneIMG1,6.gif")
            self.blancIMG = PhotoImage(file="blancIMG1,6.gif")
            self.noirIMG = PhotoImage(file="noirIMG1,6.gif")


    def generation(self):
        self.reference=Canvas(self.fenetre, width=largeur, height=hauteur, bg="grey")
```

```
        self.dessin=Canvas(self.fenetre, width=largeur, height=hauteur, bg="white")
 #ces dimensions remplissent convenablement un écran 1280x768 en echelle=1, alors qu'1.6 fonctionne bien sur du 20 pouces

        #6 lignes sur 6 colonnes = 1 grosse icone (1)=70x70 pixels
        self.reference.grid(row=6, rowspan=54, column=0, columnspan=30)
        self.dessin.grid(row=6, rowspan=54, column=36, columnspan=30)

        self.L1=[] #on génère le quadrillage de PIXELS (carré de 35*35 pixels en echelle=1)qu'on associe à une matrice ( liste de listes )
        self.C1=[]
        self.L2=[]
        self.C2=[]
        for y in range (0,hauteur,cote_PIXEL):
            self.C1.append([])
            self.C2.append([])
            for x in range (0,largeur,cote_PIXEL):
                self.L1.append(0)
                self.L2.append(0)

self.L1[x//cote_PIXEL]=self.reference.create_rectangle(x,y,x+cote_PIXEL,y+cote_PIXEL,fill="white",outline="grey")
                self.C1[y//cote_PIXEL].append(self.L1[x//cote_PIXEL]) #(C1[ligne])[colonne] correspond à un PIXEL du canvas de référence (gauche)

self.L2[x//cote_PIXEL]=self.dessin.create_rectangle(x,y,x+cote_PIXEL,y+cote_PIXEL,fill="white", outline="grey")
                self.C2[y//cote_PIXEL].append(self.L2[x//cote_PIXEL]) #(C2[ligne])[colonne] correspond à un PIXEL du canvas de dessin (droite)

        self.selection=0
        #Boutons principaux
        self.peindre=Button(self.fenetre, image=self.peindreIMG,borderwidth=3)
        self.peindre.grid(row=0, column=0, rowspan=6, columnspan=6)

        self.ligne=Button(self.fenetre, image=self.ligneIMG,borderwidth=3)
        self.ligne.grid(row=0, column=6, rowspan=6, columnspan=6)

        self.crayon=Button(self.fenetre, image=self.crayonIMG,borderwidth=3)
        self.crayon.grid(row=0, column=12, rowspan=6, columnspan=6)

        self.gomme=Button(self.fenetre, image=self.gommeIMG,borderwidth=3)
        self.gomme.grid(row=0, column=18, rowspan=6, columnspan=6)

        self.selec=Button(self.fenetre, image=self.selecIMG,borderwidth=3)
        self.selec.grid(row=0, column=24, rowspan=6, columnspan=6)
```

```
        self.copier=Button(self.fenetre, image=self.ctrlcIMG,state='disabled',borderwidth=3) #ces
boutons sont désactivés et grisés par défaut, ils auront besoin d'une sélection ou d'un copier
        self.copier.grid(row=0, column=30, rowspan=6, columnspan=6)

        self.coller=Button(self.fenetre, image=self.ctrlvIMG,state='disabled',borderwidth=3)
        self.coller.grid(row=0, column=36, rowspan=6, columnspan=6)

        self.suppr=Button(self.fenetre, image=self.supprIMG,state='disabled',borderwidth=3)
        self.suppr.grid(row=0, column=42, rowspan=6, columnspan=6)

        self.suivant=Button(self.fenetre, image=self.ctrlzIMG,state='disabled',borderwidth=3)
        self.suivant.grid(row=0, column=48, rowspan=6, columnspan=6)


        self.score=Message(self.fenetre,aspect=500,bg='goldenrod',justify='center',relief='sunken')
        self.score.grid(row=0, column=54,rowspan=3,columnspan=9)


self.timer=Message(self.fenetre,aspect=500,bg='goldenrod',justify='center',relief='sunken',textvariable
=((self.temps)))
        self.timer.grid(row=3,column=54,rowspan=3,columnspan=9)

        self.scoretot=Message(self.fenetre,aspect=150,bg='purple1',justify='center',relief='sunken')
        self.scoretot.grid(row=0,column=63,rowspan=6,columnspan=4)

        #Boutons palette
        self.couleur_active="red2" #la valeur par défaut est rouge
        self.rouge=Button(self.fenetre,
image=self.rougeIMG,command=self.rouge,relief="sunken",borderwidth=3,background='magenta4')
#donc par défaut, le bouton est de relief appuyé (=sunken)par défaut
        self.rouge.grid(row=6, column=30, rowspan=3, columnspan=2)

        self.vert=Button(self.fenetre, image=self.vertIMG,command=self.vert,borderwidth=3)
        self.vert.grid(row=6, column=32, rowspan=3, columnspan=2)

        self.bleu=Button(self.fenetre, image=self.bleuIMG,command=self.bleu,borderwidth=3)
        self.bleu.grid(row=6, column=34, rowspan=3, columnspan=2)

        self.jaune=Button(self.fenetre, image=self.jauneIMG,command=self.jaune,borderwidth=3)
        self.jaune.grid(row=9, column=30, rowspan=3, columnspan=2)

        self.blanc=Button(self.fenetre, image=self.blancIMG,command=self.blanc,borderwidth=3)
        self.blanc.grid(row=9, column=32, rowspan=3, columnspan=2)

        self.noir=Button(self.fenetre, image=self.noirIMG,command=self.noir,borderwidth=3)
        self.noir.grid(row=9, column=34, rowspan=3,columnspan=2)
```

```
    def rouge(self): #les fonctions très simples pour chaque bouton de la palette, qui modifient la valeur de self.couleur_active utilisée pour dessiner
        self.couleur_active="red2"
        self.desactiver_palette()
        self.bouton_actif(self.rouge)
    def vert(self):
        self.couleur_active="lime green"
        self.desactiver_palette()
        self.bouton_actif(self.vert)
    def bleu(self):
        self.couleur_active="blue"
        self.desactiver_palette()
        self.bouton_actif(self.bleu)
    def jaune(self):
        self.couleur_active="yellow2"
        self.desactiver_palette()
        self.bouton_actif(self.jaune)
    def blanc(self):
        self.couleur_active="white"
        self.desactiver_palette()
        self.bouton_actif(self.blanc)
    def noir(self):
        self.couleur_active="black"
        self.desactiver_palette()
        self.bouton_actif(self.noir)

    def creer_raccourci(self,bouton,emplacement): #duplique un bouton dans un des 4 emplacements du milieu : emplacement=0 pour celui en-dessous de la palette, emplacement = 3 pour le dernier
        #ils sont stockés dans une liste et indexés par leur emplacement
        if emplacement==self.peindreracc:
            self.peindreracc=-1
        elif emplacement==self.ligneracc:
            self.ligneracc=-1
        elif emplacement==self.crayonracc:
            self.crayonracc=-1
        elif emplacement==self.gommeracc:
            self.gommeracc=-1
        elif emplacement==self.selecracc:
            self.selecracc=-1
        elif emplacement==self.copierracc:
            self.copierracc=-1
        elif emplacement==self.collerracc:
            self.collerracc=-1
        elif emplacement==self.supprracc:
            self.supprracc=-1
        elif emplacement==self.suivracc:
            self.suivracc=-1 #si un des Vue.boutonracc vaut -1, c'est que ce bouton n'a pas de raccourci,
```

```
    #c'est important pour ne pas faire d'appel incohérent (ex : changer la couleur d'un bouton non existant)
        if bouton['command']==self.peindre['command']:
            self.peindreracc=emplacement
        elif bouton['command']==self.ligne['command']:
            self.ligneracc=emplacement
        elif bouton['command']==self.crayon['command']:
            self.crayonracc=emplacement
        elif bouton['command']==self.gomme['command']:
            self.gommeracc=emplacement
        elif bouton['command']==self.selec['command']:
            self.selecracc=emplacement
        elif bouton['command']==self.copier['command']:
            self.copierracc=emplacement
        elif bouton['command']==self.coller['command']:
            self.collerracc=emplacement
        elif bouton['command']==self.suppr['command']:
            self.supprracc=emplacement
        elif bouton['command']==self.suivant['command']: #toutes ces lignes permettent d'identifier quelle est la fonction de chaque raccourci
            self.suivracc=emplacement #c'est uniquement utile pour mettre à jour l'état visuel (relief+bordure) du bouton et de son raccourci en même temps

        self.raccourcis[emplacement]=Button(self.fenetre,image=bouton['image'])
        self.raccourcis[emplacement].configure(command=bouton['command'],state=bouton['state'])
        self.raccourcis[emplacement].grid(column=30, columnspan=6, rowspan=6, row=12+emplacement*6)

    def update_raccourcis(self): #voir fonction  adaptation du modèle
        for p in range (4):
            if self.preraccourcis[p]!=0:
                self.creer_raccourci(self.preraccourcis[p],p)
            elif self.preraccourcis[p]==0:
                q=p
                while q<8 and self.preraccourcis[q]==0:
                    q=q+1
                if self.preraccourcis[q]!=0:
                    self.creer_raccourci(self.preraccourcis[q],p)
                    self.preraccourcis[q]=0
                elif self.raccourcis[p]!=0:
                    self.raccourcis[p].destroy()

    def bouton_actif(self,bouton):
        bouton.configure(relief="sunken",background='blue')

    def bouton_inactif(self,bouton):
        bouton.configure(relief="raised",background='grey99')
```

```
    def desactiver_boutons(self):
        self.bouton_inactif(self.peindre)
        self.bouton_inactif(self.ligne)
        self.bouton_inactif(self.crayon)
        self.bouton_inactif(self.gomme)
        self.bouton_inactif(self.selec)
        self.bouton_inactif(self.coller)
        for i in range (4):
            if self.raccourcis[i]!=0:
                self.bouton_inactif(self.raccourcis[i])
#ces trois fonctions permettent de visualiser quel bouton est actif en le mettant en relief et en enlevant le relief des autres

    def desactiver_palette(self):
        self.bouton_inactif(self.rouge)
        self.bouton_inactif(self.vert)
        self.bouton_inactif(self.bleu)
        self.bouton_inactif(self.jaune)
        self.bouton_inactif(self.blanc)
        self.bouton_inactif(self.noir)

    def colorier_PIXEL(self,l,c,couleur): #fonction de base souvent réutilisée pour colorier un pixel de la zone de dessin
        self.dessin.itemconfigure(self.C2[l][c],fill=couleur)

    def colorier_PIXEL_modele(self,l,c,couleur):
        self.reference.itemconfigure(self.C1[l][c],fill=couleur)

    def effacer(self):
        for y in range (hauteur//cote_PIXEL):
            for x in range (largeur//cote_PIXEL):
                self.colorier_PIXEL(y,x,'white')

    def ligneclic1(self,event):
        if event.x<largeur and event.y<hauteur:
            self.l1=event.y//cote_PIXEL
            self.c1=event.x//cote_PIXEL

    def crayonclic(self,event):
        l=event.y//cote_PIXEL
        c=event.x//cote_PIXEL #retourne les cordonnées du PIXEL (l,c)=(ligne,colonne) sur lequel la souris est
        self.colorier_PIXEL(l,c,self.couleur_active)

    def gommeclic(self,event):
        l=event.y//cote_PIXEL
        c=event.x//cote_PIXEL
        self.colorier_PIXEL(l,c,"white")
```

```
    def selecclic1(self,event):
        if event.x<largeur and event.y<hauteur:
            self.l1=event.y//cote_PIXEL
            self.c1=event.x//cote_PIXEL
    def selecclic2(self,event):
        l=event.y//cote_PIXEL
        c=event.x//cote_PIXEL
        if event.x<largeur and event.y<hauteur:
            if self.c1<=c:
                deltax2=1
                deltax1=0
            else :
                deltax2=0
                deltax1=1
            if self.l1<=l:
                deltay2=1
                deltay1=0
            else:
                deltay1=1
                deltay2=0 #on compense pour que la sélection prenne une zone équivalente peu importe les
2 directions (haut vers bas, gauche vers droite,etc...)

self.selection=self.dessin.create_rectangle((self.c1+deltax1)*cote_PIXEL,(self.l1+deltay1)*cote_PIX
EL,(c+deltax2)*cote_PIXEL,(l+deltay2)*cote_PIXEL,width=3,outline='darkblue',dash=(20,20))
            self.copier.configure(state='normal')
            self.suppr.configure(state='normal')
            if self.copierracc>=0:
                self.raccourcis[self.copierracc].configure(state='normal')
            if self.supprracc>=0:
                self.raccourcis[self.supprracc].configure(state='normal')


    def resetselec(self):
            self.dessin.delete(self.selection)
            self.copier.configure(state='disabled')
            self.suppr.configure(state='disabled')
            if self.copierracc>=0:
                self.raccourcis[self.copierracc].configure(state='disabled')
            if self.supprracc>=0:
                self.raccourcis[self.supprracc].configure(state='disabled')
```

Controleurinterfia.py

```
#Contrôleur interfia
import copy

from Vueinterfia import *
from Modeleinterfia import *

#Données qu'on recupère sous forme de logs
Clic_peindre=0 #le nombre de clics pour chaque bouton
Clic_ligne=0
Clic_crayon=0
Clic_gomme=0
Clic_selec=0
Clic_copier=0
Clic_coller=0
Clic_suppr=0
Clic_suivant=0 #celui là est présent juste pour détecter si il y a eu peu avoir une faille

Use_peindre=0 #le nombre d'utilisations des fonctions qui modifient le clic
Use_ligne=0
Use_crayon=0
Use_gomme=0
Use_selec=0
Use_coller=0


Score=0 #1 point pour chaque dessin + le pourcentage de progression du dernier dessin
Hésitation=0
hésit=0

#les fonctions avec le nom du bouton (gomme,peindre,...) appellent les fonctions
# en -er (gommer,peindrer -désolé langue française-) qui appellent les fonctions de type boutonclic
# tout cela pour pouvoir assigner aux boutons des fonctions qui ne prennent pas de paramètre event,
# comme un clic de souris par exemple, en argument; et pour pouvoir actualiser le modèle
#enfin un peu près
#les fonctions associés aux boutons sont écrits dans l'ordre de l'interface en lecture de gauche à droite


def temps_ecoule():
    Fenetre.fenetre.after(Timer-2000,Data)   #on minute la session de test pour préciser les données
    Fenetre.fenetre.after(Timer,lafin)

def Data():
    if Adaptative==1:
        with open('logstesta.txt','w+') as donnees:
            if Sequence==1:
```

```
            seq='impaire'
        elif Sequence==0:
            seq='paire'
        donnees.write('données test : ' +'Interface : ' + ' adaptative' + '; Séquence : ' + seq
                + '; '+ 'Score total : ' + str(Score)+','+ str(progresser())
                +'; Hésitation : '+str(Hésitation)+'; Clic_peindre :' + str(Clic_peindre)+'; Use_peindre :'
+ str(Use_peindre)
                +'; Clic_ligne :' + str(Clic_ligne)+'; Use_ligne :' + str(Use_ligne)+'; Clic_crayon :' +
str(Clic_crayon)
                +'; Use_crayon :' + str(Clic_crayon)+'; Clic_gomme :' + str(Clic_gomme)+';
Use_gomme :' + str(Use_gomme)
                +'; Clic_selec :' + str(Clic_selec)+'; Use_selec :' + str(Use_selec)+'; Clic_copier :' +
str(Clic_copier)
                +'; Clic_coller :' + str(Clic_coller)+'; Clic_suppr :' + str(Clic_suppr)
                +'; Use_coller :' + str(Use_coller)   +'; Clic_suivant :' + str(Clic_suivant))
    elif Adaptative==0:
        with open('logstestb.txt','w+') as donnees:
            if Sequence==1:
                seq='impaire'
            elif Sequence==0:
                seq='paire'
            donnees.write('données test : ' +'Interface : ' + ' statique' + '; Séquence : ' + seq
                    + '; '+ 'Score total : ' + str(Score)+','+ str(progresser())
                    +'; Hésitation : '+str(Hésitation)+'; Clic_peindre :' + str(Clic_peindre)+'; Use_peindre :'
+ str(Use_peindre)
                    +'; Clic_ligne :' + str(Clic_ligne)+'; Use_ligne :' + str(Use_ligne)+'; Clic_crayon :' +
str(Clic_crayon)
                    +'; Use_crayon :' + str(Clic_crayon)+'; Clic_gomme :' + str(Clic_gomme)+';
Use_gomme :' + str(Use_gomme)
                    +'; Clic_selec :' + str(Clic_selec)+'; Use_selec :' + str(Use_selec)+'; Clic_copier :' +
str(Clic_copier)
                    +'; Clic_coller :' + str(Clic_coller)+'; Clic_suppr :' + str(Clic_suppr)
                    +'; Use_coller :' + str(Use_coller)   +'; Clic_suivant :' + str(Clic_suivant))


def lafin():
  Fenetre.fenetre.destroy()

def hesiter():
    if hésit==1:
        global Hésitation
        Hésitation=Hésitation+1

def marque(ligne,colonne,A): #utile pour que la fonction peindre_recursif ne revienne pas sur des
pixels déjà traités
    A[ligne][colonne]=1

def peindre_recursif(li,co,couleur,A):
```

```
    if (couleur==Dessin.couleur_PIXEL(li,co)) and (A[li][co]!=1):
        Fenetre.colorier_PIXEL(li,co,Fenetre.couleur_active)
        Dessin.actualiser(li,co,Fenetre.couleur_active)
        marque(li,co,A)

        l=li
        c=co
        bas=li+1
        haut=li-1
        droite=co+1
        gauche=co-1

        if l<17:
            #gestion de tous les cas bordures
            peindre_recursif(bas,c,couleur,A)
        if l>0:
            peindre_recursif(haut,c,couleur,A)

        if c<9:
            peindre_recursif(l,droite,couleur,A)

        if c>0:
            peindre_recursif(l,gauche,couleur,A)


def peindreclic(event):
    global Use_peindre
    Use_peindre=Use_peindre+1
    global hésit
    hésit=0
    l=event.y//cote_PIXEL
    c=event.x//cote_PIXEL
    coul=Dessin.couleur_PIXEL(l,c)
    A=[]
    B=[]
    for y in range (0,hauteur//cote_PIXEL):
        A.append([])
        for x in range (0,largeur//cote_PIXEL):
            B.append(0)
            A[y].append(B[x])
    peindre_recursif(l,c,coul,A)
    progresser() #à chaque modification de pixels de l'utilisateur, on met à jour le score de progression
affiché

def peindre():
    hesiter()
    global hésit
```

```
    hésit=1
    Fenetre.resetselec()
    Dessin.resetselec()
    Fenetre.dessin.bind("<Button-1>",peindreclic)
    Fenetre.dessin.unbind("<B1-Motion>")
    Fenetre.dessin.unbind("<B1-ButtonRelease>")

    global Clic_peindre
    Clic_peindre=Clic_peindre+1
    adaptation(Fenetre.peindre)

    Fenetre.desactiver_boutons()
    Fenetre.bouton_actif(Fenetre.peindre)
    if Fenetre.peindreracc!=-1:
        Fenetre.bouton_actif(Fenetre.raccourcis[Fenetre.peindreracc])

def ligner1(event):
    Fenetre.ligneclic1(event)

def ligner2(event):
    global Use_ligne
    Use_ligne=Use_ligne+1
    global hésit
    hésit=0
    if event.x<=largeur and event.y<=hauteur: #garde-fou de non sortie du canvas
        l2=event.y//cote_PIXEL
        c2=event.x//cote_PIXEL
        if Fenetre.l1<=l2: #ces sens sont utiles pour faire une double boucle for qui fonctionne dans tous
les sens de parcours de l'écran
            sensy=1
            deltay=1
        else :
            sensy=-1
            deltay=-1
        if Fenetre.c1<=c2:
            sensx=1
            deltax=1
        else:
            sensx=-1
            deltax=-1
        if Fenetre.l1==l2:
            for x in range (Fenetre.c1,c2+deltax,sensx):
                Fenetre.colorier_PIXEL(l2,x,Fenetre.couleur_active)
                Dessin.actualiser(l2,x,Fenetre.couleur_active)
        elif Fenetre.c1==c2:
            for y in range (Fenetre.l1,l2+deltay,sensy):
                Fenetre.colorier_PIXEL(y,c2,Fenetre.couleur_active)
                Dessin.actualiser(y,c2,Fenetre.couleur_active)
```

```
        else:
            diffx=abs(Fenetre.c1-c2)
            diffy=abs(Fenetre.l1-l2)
            if diffy>=diffx:
                for x in range (Fenetre.c1,c2+deltax,sensx):
                    for y in range (Fenetre.l1,l2+deltay,sensy):
                        if ((y-Fenetre.l1)*sensy//(diffy/diffx))==((x-Fenetre.c1)*sensx): #permet de tracer ce qui ressemble le plus à une ligne diagonale dans n'importe quelle situation différente d'une ligne droite
                            Fenetre.colorier_PIXEL(y,x,Fenetre.couleur_active)
                            Dessin.actualiser(y,x,Fenetre.couleur_active)
            elif diffx>diffy:
                for y in range (Fenetre.l1,l2+deltay,sensy):
                    for x in range (Fenetre.c1,c2+deltax,sensx):
                        if ((x-Fenetre.c1)*sensx//(diffx/diffy))==((y-Fenetre.l1)*sensy):
                            Fenetre.colorier_PIXEL(y,x,Fenetre.couleur_active)
                            Dessin.actualiser(y,x,Fenetre.couleur_active)
    progresser()


def ligne():
    hesiter()
    global hésit
    hésit=1
    Fenetre.resetselec()
    Dessin.resetselec()
    Fenetre.dessin.bind("<Button-1>",ligner1)
    Fenetre.dessin.bind("<B1-ButtonRelease>",ligner2)
    Fenetre.dessin.unbind("<B1-Motion>")

    global Clic_ligne
    Clic_ligne=Clic_ligne+1
    adaptation(Fenetre.ligne)

    Fenetre.desactiver_boutons()
    Fenetre.bouton_actif(Fenetre.ligne)
    if Fenetre.ligneracc!=-1:
        Fenetre.bouton_actif(Fenetre.raccourcis[Fenetre.ligneracc])


def crayonner(event):
    global hésit
    hésit=0
    global Use_crayon
    Use_crayon=Use_crayon+1
    if event.x<largeur and event.y<hauteur:
        Fenetre.crayonclic(event)
        Dessin.actualiser(event.y//cote_PIXEL,event.x//cote_PIXEL,Fenetre.couleur_active)
```

```
    progresser()

def crayon():
    hesiter()
    global hésit
    hésit=1
    Fenetre.resetselec()
    Dessin.resetselec()
    Fenetre.dessin.bind("<Button-1>",crayonner)
    Fenetre.dessin.bind("<B1-Motion>",crayonner)
    Fenetre.dessin.unbind("<B1-ButtonRelease>")

    global Clic_crayon
    Clic_crayon=Clic_crayon+1
    adaptation(Fenetre.crayon)

    Fenetre.desactiver_boutons()
    Fenetre.bouton_actif(Fenetre.crayon)
    if Fenetre.crayonracc!=-1:
        Fenetre.bouton_actif(Fenetre.raccourcis[Fenetre.crayonracc])


def gommer(event):
    global hésit
    hésit=0
    global Use_gomme
    Use_gomme=Use_gomme+1
    if event.x<largeur and event.y<hauteur:
        Fenetre.gommeclic(event)
        Dessin.actualiser(event.y//cote_PIXEL,event.x//cote_PIXEL,"white")
    progresser()

def gomme():
    hesiter()
    global hésit
    hésit=1
    Fenetre.resetselec()
    Dessin.resetselec()
    Fenetre.dessin.bind("<Button-1>",gommer)
    Fenetre.dessin.bind("<B1-Motion>",gommer)
    Fenetre.dessin.unbind("<B1-ButtonRelease>")

    global Clic_gomme
    Clic_gomme=Clic_gomme+1
    adaptation(Fenetre.gomme)

    Fenetre.desactiver_boutons()
    Fenetre.bouton_actif(Fenetre.gomme)
```

```
    if Fenetre.gommeracc!=-1:
        Fenetre.bouton_actif(Fenetre.raccourcis[Fenetre.gommeracc])


def selecer1(event):
    Fenetre.resetselec()
    Dessin.resetselec()
    Fenetre.selecclic1(event)

def selecer2(event):
    global hésit
    hésit=0
    global Use_selec
    Use_selec=Use_selec+1
    if event.x<=largeur and event.y<=hauteur:
        l2=event.y//cote_PIXEL
        c2=event.x//cote_PIXEL
        Dessin.x1selec=Fenetre.c1
        Dessin.y1selec=Fenetre.l1
        Dessin.x2selec=c2
        Dessin.y2selec=l2
        if Dessin.y1selec<=Dessin.y2selec:
            sensy=1 #les sens ont la même utilité que pour la fonction ligne
            deltay=1 #ces delta assurent la correspondance avec le rectangle de sélection visuelle (fichier
Vue)
        else :
            sensy=-1
            deltay=-1
        if Dessin.x1selec<=Dessin.x2selec:
            sensx=1
            deltax=1
        else:
            sensx=-1
            deltax=-1
        for y in range (Fenetre.l1,l2+deltay,sensy):
            for x in range (Fenetre.c1,c2+deltax,sensx):
                Dessin.selectionC[y][x]=Dessin.couleur_PIXEL(y,x)
    Fenetre.selecclic2(event)

def selec():
    hesiter()
    global hésit
    hésit=1
    Fenetre.dessin.bind("<Button-1>",selecer1)
    Fenetre.dessin.bind("<B1-ButtonRelease>",selecer2)
    Fenetre.dessin.unbind("<B1-Motion>")

    global Clic_selec
```

```
    Clic_selec=Clic_selec+1
    adaptation(Fenetre.selec)

    Fenetre.desactiver_boutons()
    Fenetre.bouton_actif(Fenetre.selec)
    if Fenetre.selecracc!=-1:
        Fenetre.bouton_actif(Fenetre.raccourcis[Fenetre.selecracc])


def copier():
    hesiter()
    Fenetre.coller.configure(state='normal')
    Dessin.copieC.clear()
    Dessin.copieL.clear()

    Dessin.copieC=copy.deepcopy(Dessin.selectionC) #cette méthode, importé en début de programme permet de cloner la matrice de sélection

    for k in range(len(Dessin.copieC)-1,-1,-1): #je peux ensuite enlever tous les -1 de la matrice de sélection pour la réduire à ce qui a vraiment été sélectionné
        long=len(Dessin.copieC[k])
        for i in range(len(Dessin.copieC[k])-1,-1,-1):
            if (Dessin.copieC[k][i])==(-1):
                del (Dessin.copieC[k][i])
        if (Dessin.copieC[k])==[]:
            del (Dessin.copieC[k])

    global Clic_copier
    Clic_copier=Clic_copier+1
    adaptation(Fenetre.copier)


def collerclic(event):
    global hésit
    hésit=0
    h=event.y//cote_PIXEL
    l=event.x//cote_PIXEL
    for y in range (h,h+len(Dessin.copieC)):
        for x in range (l,l+len(Dessin.copieC[0])):
            if ((x<largeur//cote_PIXEL) and (y<hauteur//cote_PIXEL) and (x>=0) and (y>=0)):
                Fenetre.colorier_PIXEL(y,x,Dessin.copieC[y-h][x-l])
                Dessin.actualiser(y,x,Dessin.copieC[y-h][x-l])
    progresser()
    global Use_coller
    Use_coller=Use_coller+1

def coller():
    hesiter()
```

```
    global hésit
    hésit=1
    Fenetre.resetselec()
    Dessin.resetselec()
    Fenetre.dessin.bind("<Button-1>",collerclic)
    Fenetre.dessin.unbind("<B1-ButtonRelease>")
    Fenetre.dessin.unbind("<B1-Motion>")

    global Clic_coller
    Clic_coller=Clic_coller+1
    adaptation(Fenetre.coller)

    Fenetre.desactiver_boutons()
    Fenetre.bouton_actif(Fenetre.coller)
    if Fenetre.collerracc!=-1:
        Fenetre.bouton_actif(Fenetre.raccourcis[Fenetre.collerracc])


def suppr():
    hesiter()
    if Dessin.y1selec<=Dessin.y2selec:
        sensy=1
        deltay=1
    else :
        sensy=-1
        deltay=-1
    if Dessin.x1selec<=Dessin.x2selec:
        sensx=1
        deltax=1
    else:
        sensx=-1
        deltax=-1
    for y in range (Dessin.y1selec,Dessin.y2selec+deltay,sensy):
        for x in range (Dessin.x1selec,Dessin.x2selec+deltax,sensx):
            Fenetre.colorier_PIXEL(y,x,'white')
            Dessin.actualiser(y,x,'white')
    Fenetre.resetselec()
    Dessin.resetselec()
    progresser()
    global Clic_suppr
    Clic_suppr=Clic_suppr+1
    adaptation(Fenetre.suppr)

def progresser():
    prog=0
    for y in range (hauteur//cote_PIXEL):
        for x in range ((largeur//cote_PIXEL)):
            if Dessin.couleur_PIXEL(y,x)==Reference.couleur_PIXEL(y,x):
```

```
            prog=prog+(5/9)
    Fenetre.score.configure(text="Progression : " + str(int(prog)) + "%")
    if int(prog)==100:
        Fenetre.suivant.configure(state='normal')
        if Fenetre.suivracc>=0:
            Fenetre.raccourcis[Fenetre.suivracc].configure(state='normal')
    else:
        Fenetre.suivant.configure(state='disabled')
        if Fenetre.suivracc>=0:
            Fenetre.raccourcis[Fenetre.suivracc].configure(state='disabled')
    return(prog)

def suivanter():
    Fenetre.effacer()
    Dessin.effacer()
    progresser()
    global Score
    Score=Score+1
    Fenetre.scoretot.configure(text="Score:"+ str(Score))
def suivant():
   #cette fonction définit la séquence de modèles de référence : les mod impairs ou les mod pairs
    global Clic_suivant
    Clic_suivant=Clic_suivant+1
    adaptation(Fenetre.suivant)
    if Sequence==1:
        if Score==0:
            mod4()
            suivanter()
[...]
        elif Score==17:
            fin()
            suivanter()

    elif Sequence==0:
        if Score==0:
            mod3()
            suivanter()
[...]
        elif Score==17:
            fin()
            suivanter()

Fenetre=Vue()
Fenetre.genicones()
Fenetre.generation() #on génère tout l'interface

boutons={} #on crée un dictionnaire pour pouvoir associer les boutons à leur variable Clic_bouton,
pour pouvoir les classer et proposer les raccourcis en adéquation
```

```
def adaptation(bouton):
   if Adaptative==1:

Classement=[Clic_peindre,Clic_ligne,Clic_crayon,Clic_gomme,Clic_selec,Clic_copier,Clic_coller,Clic_suppr,Clic_suivant]
      Classement.sort() #cette ligne classe les nb d'utilisations de Clic dans la liste Classement par ordre croissant
      boutons[Fenetre.peindre]=Clic_peindre
      boutons[Fenetre.ligne]=Clic_ligne
      boutons[Fenetre.crayon]=Clic_crayon
      boutons[Fenetre.gomme]=Clic_gomme
      boutons[Fenetre.selec]=Clic_selec
      boutons[Fenetre.copier]=Clic_copier
      boutons[Fenetre.coller]=Clic_coller
      boutons[Fenetre.suppr]=Clic_suppr
      boutons[Fenetre.suivant]=Clic_suivant

      k=8
      while (boutons[bouton]<Classement[k]) and (k>4):
         k=k-1

      Fenetre.preraccourcis=Fenetre.raccourcis
      save=[0,0,0,0,0,0,0,0,0]
      i=0
      while(i<9) and (Fenetre.raccourcis[i]!=0):
         save[i]=Fenetre.raccourcis[i]['command']
         i=i+1
      boutonsave=[0,0,0,0,0,0,0,0,0]
      for p in range (len(save)):
         if (save[p])==(Fenetre.peindre['command']):
            boutonsave[p]=Fenetre.peindre
         elif (save[p]==Fenetre.ligne['command']):
            boutonsave[p]=Fenetre.ligne
         elif (save[p]==Fenetre.crayon['command']):
            boutonsave[p]=Fenetre.crayon
         elif (save[p]==Fenetre.gomme['command']):
            boutonsave[p]=Fenetre.gomme
         elif (save[p]==Fenetre.selec['command']):
            boutonsave[p]=Fenetre.selec
         elif (save[p]==Fenetre.copier['command']):
            boutonsave[p]=Fenetre.copier
         elif (save[p]==Fenetre.coller['command']):
            boutonsave[p]=Fenetre.coller
         elif (save[p]==Fenetre.suppr['command']):
            boutonsave[p]=Fenetre.suppr
         elif (save[p]==Fenetre.suivant['command']):
            boutonsave[p]=Fenetre.suivant
```

```
        if (k>=5) and (boutonsave[8-k]!=bouton): #grâce à la liste triée je peux voir si le nb de clics de la fonction du bouton cliqué est dans le top 4 (et donc devient un raccourci)
            for r in range(len(Fenetre.preraccourcis)):
                if (Fenetre.preraccourcis[r]!=0)
and((Fenetre.preraccourcis[r]['command'])==(bouton['command'])):
                    Fenetre.preraccourcis[r]=0 #on élimine dans la liste des raccourcis les boutons semblables à celui qui vient d'être cliqué pour ne pas avoir de doublons

            for u in range (8,(8-k),-1):
                if (Fenetre.preraccourcis[u-1]!=bouton):
                    Fenetre.preraccourcis[u]=Fenetre.preraccourcis[u-1]

            Fenetre.creer_raccourci(bouton,8-k)
            Fenetre.update_raccourcis()

Fenetre.peindre.configure(command=peindre)
Fenetre.ligne.configure(command=ligne)
Fenetre.crayon.configure(command=crayon) #on assigne les commandes aux boutons ici pour pouvoir utiliser des fonctions issues du fichier contrôleur
Fenetre.gomme.configure(command=gomme)
Fenetre.selec.configure(command=selec)
Fenetre.copier.configure(command=copier)
Fenetre.coller.configure(command=coller)
Fenetre.suppr.configure(command=suppr)
Fenetre.suivant.configure(command=suivant)

Reference=Modele()
Dessin=Modele() #on génère les deux modèles : celui correspondant à la référence (parfois appelé modèle -de dessin- également) et le modèle du dessin, mis à jour par l'utilisateur et son crayon virtuel

if Sequence==1:
    mod2()
elif Sequence==0:
    mod1()
if Démo==1:
    exemple() #à activer pour la démo pré-test

temps_ecoule()
Fenetre.loop() #permet de faire tourner la fenêtre, doit être à la fin
```

Modeleinterfia.py

```
#Modèle interfia
from Parametresinterfia import *

class Modele():
    def __init__(self):
```

```
        self.valeur={}
        self.valeur["red2"]=0 #on associe une valeur numérique à chaque couleur de PIXEL à l'aide du dictionnaire "valeur"
        self.valeur["lime green"]=1
        self.valeur["blue"]=2
        self.valeur["yellow2"]=3
        self.valeur["white"]=4
        self.valeur["black"]=5
        self.valeur["purple3"]=6 #valeur impossible à dessiner pour l'écran de fin

        self.inv_valeur={}
        for cle,valeur in self.valeur.items(): #on crée le dictionnaire inverse
            self.inv_valeur[valeur]=cle

        self.M=[] #la liste des lignes
        self.R=[] #la liste des PIXELS, rangés de gauche à droite, qui sera rangée dans la liste des colonnes

        self.selectionC=[]
        self.selectionL=[] #la liste de liste utilisée par la fonction sélection

        self.copieC=[]
        self.copieL=[] #la matrice utilisée par la fonction copier

        self.x1selec=0 #utilisés pour délimiter la partie sélectionnée
        self.x2selec=0
        self.y1selec=0
        self.y2selec=0

        for y in range (0,hauteur//cote_PIXEL): #pour pouvoir stocker l'état des deux grilles
            self.M.append([])                   #dans une matrice numérique
            self.selectionC.append([])
            for x in range (0,largeur//cote_PIXEL):
                self.R.append(4)
                self.M[y].append(self.R[x])

                self.selectionL.append(-1)
                self.selectionC[y].append(self.selectionL[x])

    def actualiser(self,l,c,couleur):
            self.M[l][c]=self.valeur[couleur]

    def couleur_PIXEL(self,l,c):
        return self.inv_valeur[self.M[l][c]]

    def effacer(self):
        for y in range (hauteur//cote_PIXEL):
```

```
        for x in range (largeur//cote_PIXEL):
            self.actualiser(y,x,'white')

    def prntcouleur_PIXEL(self,l,c):
        print(self.inv_valeur[self.M[l][c]],"ligne=",l,"colonne=",c)

    def resetselec(self):
        for y in range (0,hauteur//cote_PIXEL):
            for x in range (0,largeur//cote_PIXEL):
                self.selectionC[y][x]=-1
```

Parametresinterfia.py

```
#Paramètres interfia
echelle=1 #valeurs possibles: 1 ou 1.6
Adaptative=1 #1 pour l'interface adaptative, 0 pour la statique
Sequence =1 #1 pour les mod numérotés impairs, 0 pour les pairs
Timer=60*17*1000 #le temps en mili-secondes à la fin duquel la fenêtre se ferme et les logs s'écrivent
Démo=0 #n'importe quelle autre valeur que 1 pour le test normal
if Démo==1:
    Timer=60*5*1000
hauteur = int(630*echelle) #rapport largeur/hauteur de 5/9 pour pour avoir icônes carrées et de la place pour les boutons
largeur = int(350*echelle)
cote_PIXEL = int (35*echelle) #Les PIXELS font 35 pixels de large
```