

Rapport : projet annuel

AIDE A LA RESOLUTION DE GRILLE DE KAKURO
YOUNES BENSITEL, YASSINE MEKHELEF

Table des matières

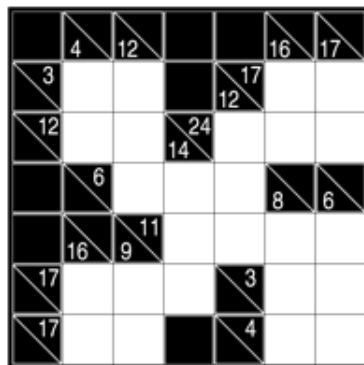
0.1	Introduction	2
0.1.1	Le Kakuro	2
0.1.2	Le sujet	2
0.2	Kakuro : mode jeu	3
0.2.1	Représentation non graphique	3
0.2.2	Les sauvegardes et chargements	4
0.2.3	Représentation graphique	4
0.3	Kakuro : Mode Editeur	7
0.3.1	La classe ApplicationEditeur	7
0.4	Aide à la résolution	8
0.4.1	Le Filtrage	8
0.4.2	L'aide graphique pour mauvaise réponse instantanée	9
0.4.3	L'aide graphique pour les cases faciles à remplir	10
0.4.4	Le système de validation	10
0.5	Déroulement du projet	11
0.5.1	Planning/grandes étapes	11
0.5.2	Le backlog	12
0.5.3	Nos choix et nos difficultés	13
0.5.4	Concernant nos choix	13
0.6	Conclusion	14
0.7	Notice d'utilisation	15
0.7.1	Mode jeu	15
0.7.2	Le mode éditeur	17

0.1 Introduction

0.1.1 Le Kakuro

Le Kakuro est un jeu d'origine Japonaise. Son nom provient du mot Kassan Cross qui signifie "addition en croix". Le jeu ressemble très fortement à un jeu de mots croisés mais adapté numériquement avec des chiffres. Au Japon, ce jeu est très populaire, il est le deuxième jeu après le sudoku. Un problème de Kakuro consiste à remplir une grille avec des chiffres de 1 à 9. Un bloc est un groupe de cases voisines sur une même ligne (ou une même colonne). Le total de cases de chaque bloc doit être égal au nombre indiqué en début de bloc. Enfin un même chiffre ne peut apparaître plus d'une fois dans un même bloc.

Voici à quoi ressemble le Kakuro :



0.1.2 Le sujet

L'objectif du projet est la réalisation d'un outil d'aide à la résolution de Kakuro. Ce que nous avons décidé est de mettre en place un jeu de Kakuro ainsi qu'un outil d'aide à la résolution. Notre programme se scinde en deux fichiers .py, l'un est un mode éditeur permettant, comme son nom l'indique, d'éditer une grille de Kakuro, d'en créer une, de la résoudre, de l'enregistrer. L'autre mode est le mode de jeu, il permet entre autres de jouer au Kakuro, de demander de l'aide si on est bloqué, de demander la solution, d'importer sa propre grille de Kakuro, de sauvegarder une partie en cours. Nous avons également implémenté un système de thème pour rendre le jeu plus divertissant, vous pourrez donc choisir parmi une large gamme de thèmes celui qui vous correspond le mieux. Il est utile de préciser que les thèmes dits « spéciaux » sont en fait des thèmes animés.

Nous avons décidé de créer cet outil d'aide à la résolution en python avec la librairie Tkinter. Ce choix a été fait car python est l'un des meilleurs langages pour créer de petits projets où l'utilisation de la mémoire est négligeable. De plus, Tkinter est présent sur tous les ordinateurs équipés de Python. A noter que pour jouer à notre Kakuro, il est impératif de posséder Python 3. Nous tenons à remercier particulièrement M. Patrice Boizumault pour nous avoir expliqué de manière claire en quoi consistait le projet.

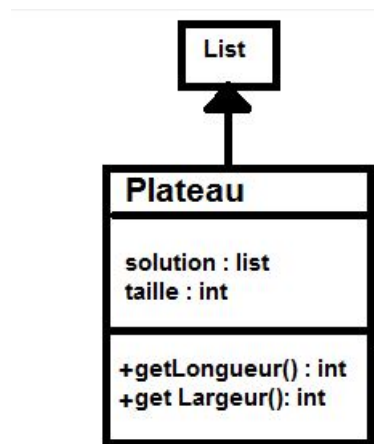
Voici le sujet du projet proposé tel quel par M. Boizumault : Concevoir et mettre en œuvre un outil graphique d'aide à la résolution de Kakuro. Il ne s'agit pas ici de développer une méthode de résolution « presse-bouton » comme demandé dans l'étape 1 du premier sujet, mais un ensemble d'outils d'aide à la résolution mis à la disposition de l'utilisateur.

Par exemple, colorer les cases (vides) dont la valeur semble « la plus facile » à déterminer compte tenu de l'état courant de la grille, visualiser les valeurs possibles d'une case, ... Pour cela, on pourra étudier les fonctionnalités proposées (ou pas) par les joueurs en ligne (<http://fr.wikipedia.org/wiki/Kakuro>).

0.2 Kakuro : mode jeu

0.2.1 Représentation non graphique

Nous avons décidé de représenter le plateau Kakuro par une classe nommée Plateau qui hérite de List. Il prend en entrée une liste qui est le plateau, et une deuxième liste qui est censée être la solution (mais ce deuxième argument est facultatif).



Lorsqu'on voudra afficher un plateau, on créera un objet de type Plateau qu'on passera à notre classe Application qui est en fait notre vue/contrôleur (nous reviendrons plus en détails sur la classe Application dans la suite du rapport). Ainsi, le Kakuro est représenté par une liste où chaque élément est une ligne du Kakuro. Cette liste qui représente la ligne contient elle-même des listes qui contiennent les informations de la colonne du Kakuro. Exemple : [[1, 2, 3], [1, 1, 1]]

Cette liste contient deux lignes et trois colonnes.

1	2	3
1	1	1

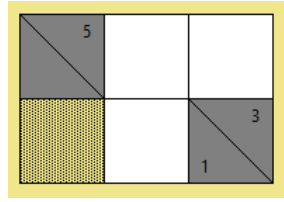
En mode graphique, on observerait ce résultat :

Il restait à savoir comment représenter :

- Les cases vides
- Les cases où il est impossible de jouer
- Les cases d'informations ?

Pour les cases vides, on les représente simplement par des 0. Pour les cases nulles on placera -1 dans la liste. Pour les cases d'informations c'est un peu plus complexe : on placera un tuple dans la case.

Exemple : [[(0,5), 0, 0], [-1, 0, (1, 3)]]



En mode graphique, on observerait ce résultat :

0.2.2 Les sauvegardes et chargements

Les sauvegardes

Pour éviter que le fait de quitter une partie soit définitif, on instaure dans le jeu un système de sauvegarde permettant de se souvenir de la progression du joueur. Dans ce but, on crée un fichier qui contiendra les informations relatives aux parties sauvegardées par le joueur. Celles-ci contiennent, entre autres l'avancement de la grille, la correction si il y en a et la taille en pixels du plateau. Lorsque l'utilisateur sauvegarde son plateau pour pouvoir y rejouer plus tard, le plateau est stocké en dur par rastérisation. On utilise en fait la librairie pickle pour transformer l'objet Plateau, qui contient les informations de la partie, en fichier impossible à modifier (donc impossible à corrompre).

Pour charger sa partie, on dé-rastérise le fichier : on prend en entrée un fichier rastérisé pour le retransformer en objet Plateau, puis on envoie ce nouveau modèle à notre classe Application qui se chargera de modifier la Vue. Ainsi, quand le joueur souhaite sauvegarder une partie pendant le jeu, on fait d'abord apparaître une nouvelle interface qui dépendra du système d'exploitation (asksaveasfilename). La fonction qui permet de transformer un objet en fichier se nomme sauvegardeobj. Elle prend en paramètres un objet et la destination du futur fichier.

Les chargements

De la même façon, lorsque le joueur souhaite charger une partie, on commence par faire apparaître une nouvelle interface (askopenfilename). De cette manière, on est capable de conserver les données concernant les parties du joueur qu'il souhaite sauvegarder, et de relancer les parties sauvegardées. La fonction qui permet de transformer ce fichier en objet s'appelle chargeobj et prend en paramètre un chemin. Elle peut être utilisée pour charger n'importe quel fichier auparavant rastérisé.

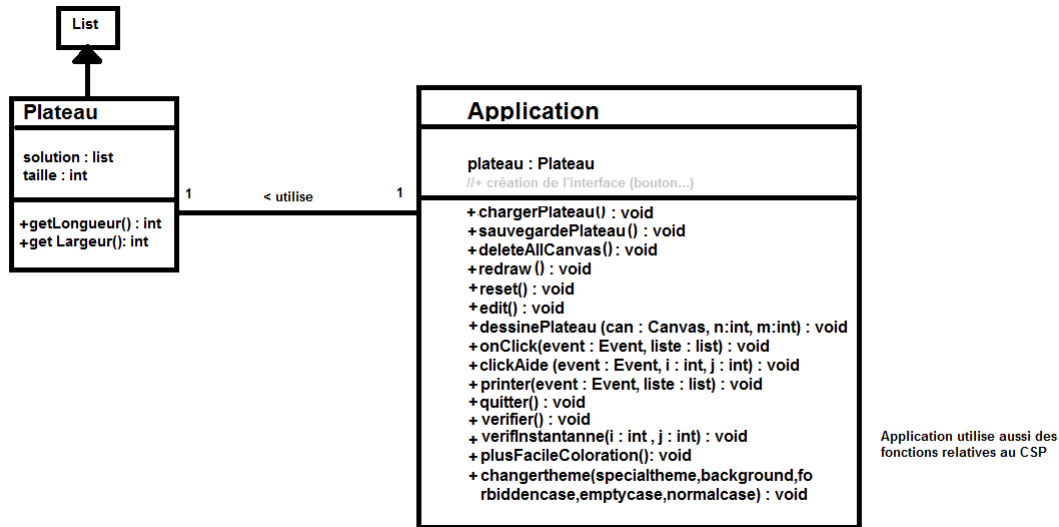
0.2.3 Représentation graphique

Pour la partie graphique, nous avons utilisé la librairie Tkinter : une interface sobre mais efficace et rapide. On utilise tkinter.messagebox pour afficher des informations quand une erreur intervient par exemple. On utilise tkinter.filedialog pour permettre d'enregistrer et de sauver des fichiers en ouvrant une fenêtre de dialogue. Toutes les informations nécessaires pour pouvoir jouer se trouvent dans la partie VI du rapport.

La classe Application

L'appelle de la classe Application permet de lancer le Kakuro. Lors de l'initialisation de cette classe, toute l'interface se crée. Cette classe Application est une Vue/Contrôleur qui

prend en entrée un objet Plateau qu'elle affichera et éditera en fonction des actions de l'utilisateur.



Pour les méthodes de la classe Application, elles sont pour la plupart explicites (exemple : reset supprime toutes les cases remplies dans la Vue et dans le modèle Plateau).

Notre plateau graphique est dessiné en respectant simplement la taille de notre plateau non graphique. Le reste est géré grâce à des constantes sur la taille de la case. Il nous suffit de parcourir deux boucles grâce à la taille de notre plateau (longueur et largeur) et de tracer les lignes pour former la grille. Nous dessinons aussi des rectangles (avec `create_rectangle` de Tkinter) afin de reproduire les cases car cela permettra de les rendre dynamique en cliquant dessus ou suivant les actions. Les rectangles tracés sont alors les cases du jeu. En parcourant notre plateau nous vérifions si cette case est :

- vide
- remplis d'un nombre
- un début de bloc
- cases noir

Une grosse partie arithmétique est importante pour toute la partie graphique du plateau (suivant la position dans la grille, la taille d'une case, la valeur écrite).

Si elle est vide : on dessine un carré vide. Si elle est remplie on dessine un carré vide et on ajoute le texte qui la contient (avec `create_text`). Le plus difficile est d'écrire le nombre au milieu de la case et de le centrer suivant sa longueur.

Des listeners sont créés suivant le numéro de la case cliquée grâce à des tags qu'on place dans les objets du canvas. Par exemple, `plateau[2,3]` aura pour tag "2,3". A chaque fois qu'on modifie l'intérieur d'une case, notre application modifie notre modèle et appelle la méthode `redraw` pour redessiner le canevas.

Les thèmes et la classe Sakura

On peut charger un thème pour rendre l'interface plus dynamique. Certains thèmes commençant par « Special » jouent des animations sur le plateau et changent les cases nulles en fonction du thème. Par exemple le thème.

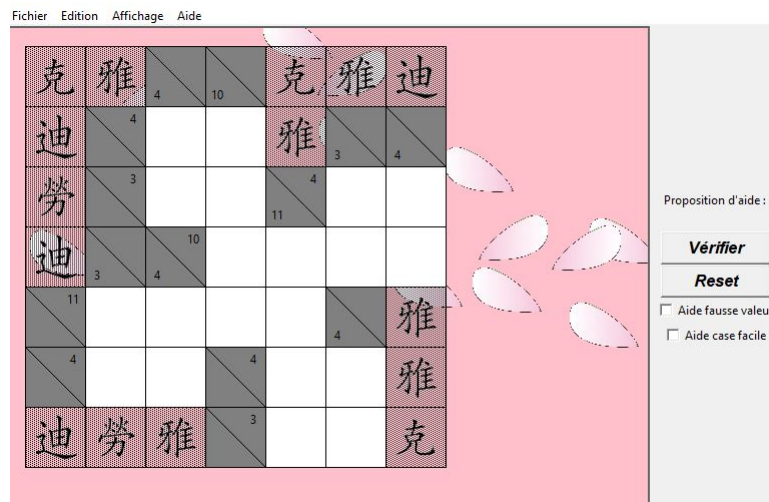
La classes Sakura permet de créer un thème dynamique. Son nom vient de la fleur de ceri-

sier car notre premier thème était le thème des fleurs de cerisier qui tombait au vent pour faire allusion au Kakuro, jeu japonais.

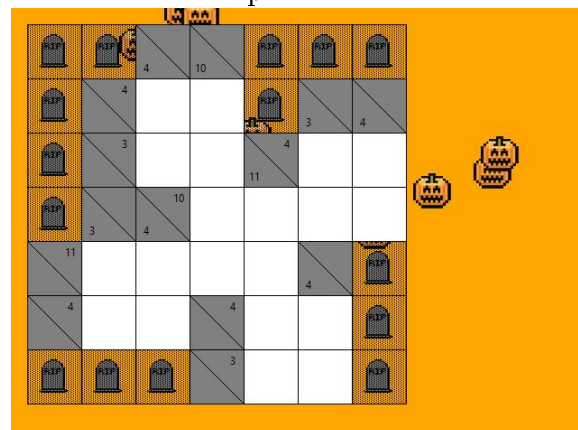
Pour créer un thème dynamique, il suffit d'instancier une classe Sakura et de donner les objets qui tomberont à cette classes, la fenêtre et le canevas où tomberont ces objets. Il ne faut pas oublier que un objet de la classe Sakura est un pétale seulement (en effet, sakura signifie pétale de cerisier, si nous voulions faire tomber plusieurs pétales nous l'auront appelée sakurada qui est le pluriel de sakura). Pour créer 100 pétales, on fera donc une boucle qui créera 100 objets Sakura. De base lorsqu'on ne passe pas d'objets à la classe Sakura, elle fera tomber des pétales de cerisier (exemple : Sakura(self.can,self)). Sakura(self.can,self,"images/winter.gif","images/winter2.gif") fera tomber des flocons de neige winter.gif et winter2.gif.

Capture d'écran des thèmes spéciaux

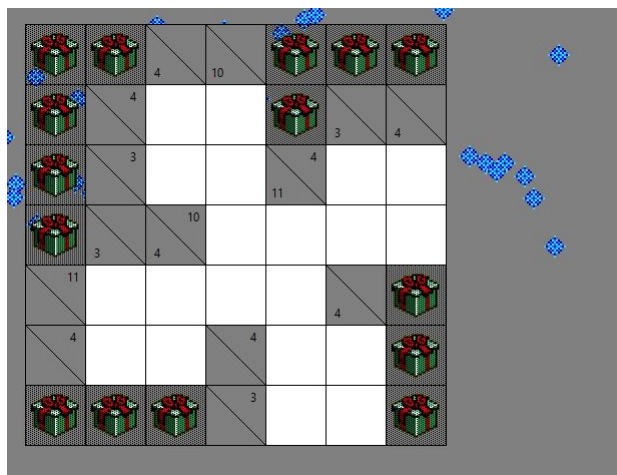
Thème Special Sakuro



Thème Special Halloween



Thème Special Hiver



Pour un thème non dynamique nous modifions simplement les couleurs dans la méthode "changetheme".

0.3 Kakuro : Mode Editeur

Le mode éditeur n'est pas si différent du mode de jeu. On a juste ajouté quelques méthodes au Plateau qui permettent de créer des cases spécifiques lors de clics. On a changé Application par ApplicationEditeur en enlevant les méthodes inutiles et en ajoutant les méthodes propres à l'édition d'un plateau. Il est possible d'ajouter sa propre solution à la grille pour une meilleure reconnaissance des erreurs lorsqu'on voudra jouer dans la partie Jeu.

On a également changé Plateau en PlateauEditeur qui implémente de nouvelles méthodes qui permettent l'édition du plateau.

0.3.1 La classe ApplicationEditeur

Les nouvelles méthodes de l'éditeur permettent de modifier le plateau en largeur et longueur avec les méthodes changeWidth et changeHeight. On modifie seulement les listes de la classe Plateau.

D'autres aussi permettent de modifier l'état des cases. Dans la classe ApplicationEditeur : la méthode "infoCase" s'active quand on double clique. Elle permet l'ajout des sommes ou d'une somme d'un bloc. La méthode "forbideCase" s'active au clique droit et permet de rendre une case interdite à écrire dedans. "EditCase" est la méthode qui s'active en cliquant une fois avec le bouton gauche et permet de créer une case éditale (là où le joueur rentrera ces chiffres). Ces trois méthodes utilisent les méthodes de plateauEditeur, respectivement "autreCase(self,x,y,x1,y2)" qui permet d'ajouter un couple de deux sommes, "InterdireCase(self,x,y)" qui passe une case à 1 et "autoriserCase(self,x,y)" qui met une case à 0 et donc éditale par le joueur.

Pour plus de détails sur les fonctionnalités du mode éditeur, veuillez s'il-vous-plaît vous référer à la partie VI du rapport.

0.4 Aide à la résolution

0.4.1 Le Filtrage

Qu'est-ce qu'un CSP ?

Un Constraint Satisfaction Problem (problème par satisfaction de contrainte en français) est un problème mathématique. Le but est de trouver une ou des solutions satisfaisant le problème suivant une ou plusieurs contraintes. En informatique il est possible de modéliser certaines de ces problèmes avec des algorithmes. Mathématiquement :

Un CSP peut se modéliser par un triplet (V, D, C) où :

- $V = (v_1, v_2, v_n)$ est un ensemble de n variables
- $D = (d_1, d_2, d_n)$ est l'ensemble des domaines associés aux variables ; chaque d_i est de cardinalité finie ; soit $d = \max |d_i|$
- $C = c_1, c_2, \dots$ est l'ensemble des e contraintes.

A partir de là il existe deux façons de procéder pour les contraintes :

- Soit par "support" avec filtrage par consistance d'arc
- Soit par comparaison sur toutes les variables.

Nous n'avons pas appliqué la méthode par support étant de niveau Master et cela aurait pu être un sujet de projet à part entière.

Notre façon de faire :

Les variables : les nombres du bloc plus petit que la somme.

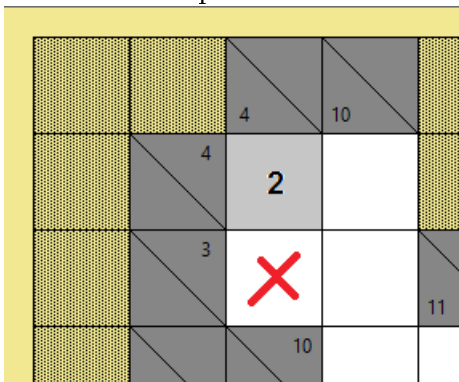
Domaine : de 1 à $S - 1$ (où S est la somme du bloc)

Contrainte : il existe deux contraintes :

AllDifferent : $v_i \neq v_j$ pour $1 \leq i < j \leq n$

Contrainte sur les sommes du bloc : la somme des v_n doit valoir S .

Voici un exemple d'illustration du CSP dans notre cas. Pour la case ayant une croix rouge :



Pour l'ensemble V : $V = \text{case}(2,1), \text{case}(2,2), \text{case}(3,2)$: les cases sur le bloc horizontal et vertical de la case sélectionnée. *En prenant le coin supérieur gauche en $(0,0)$.

Pour l'ensemble D :

$D = 1,2,3,4,5,6,7,8,9$ car les règles du Kakuro ne permettent d'avoir uniquement des cases de 1 à 9 (dans notre conception on peut cependant dépasser ces limites).

Pour l'ensemble C :

c_1 = Contrainte sur les sommes : la somme horizontale doit valoir 3 et verticalement 4 $c_2 =$

Contrainte Alldifferent : la case doit être différente de 2.

Explication d'un point de vue informatique :

D'un point de vue informatique il suffit simplement d'implémenter les contraintes sur les données de la grille.

Nous avons donc mis au point les fonctions :
AllDifferent et la contrainteSomme.

AllDifferent compare donc un nombre par tous les nombres de la ligne et de la colonne. S'il y est déjà c'est False, donc ce nombre ne pourra pas être une solution.

ContrainteSomme fait la somme des nombres déjà présents sur la ligne et compare avec la valeur de la somme du bloc vertical et horizontal. ContrainteSomme2 et contrainteSomme3 permettent d'affiner cette contrainte s'il manque seulement une case à remplir sur la ligne ou la colonne et donne la solution de cette dernière case.

D'autres fonctions pour compléter ces fonctions existent tel que getLigne, getColonne qui permettent d'obtenir la liste des variables de la ligne et de la colonne de la case choisie ou getSomme qui permet d'obtenir le couple de la Somme de la ligne et de la colonne d'une case. Algorithmiquement parlant, voici un schéma explicatif sans les affinements avec les autres fonctions sur la contrainte somme :

Proposition d'aide :
[2, 3]

Aide

Vérifier

Reset

- 1) On récupère la ligne (**getLigne**)
=> [1]
- 2) On récupère la colonne (**getColonne**)
=> []
- 3) On récupère les une ou deux sommes de la case (**getSomme**)
=> (4,4)
- 4) On teste les chiffres de 1 à 9 pour la case en vérifiant qu'ils ne sont pas déjà sur la ligne ou la colonne (**allDifferent**) tout en vérifiant si la somme de la ligne et de la colonne avec ce chiffre n'est pas supérieur à leur somme (**contrainteSomme**)
=> 2 (2 n'est pas sur la ligne ni sur la colonne, pour la ligne $2 + 1 < 4$, pour la colonne $2 + 0 < 4$)
3 (même logique)

0.4.2 L'aide graphique pour mauvaise réponse instantanée

Cette aide utilise le filtrage énoncé précédemment. Nous vérifions simplement si le nombre entré par le joueur appartient à l'ensemble des solutions du filtrage. Si ce n'est pas le cas, la case passe alors en rouge. Si une solution est donnée pour une grille, l'aide regarde simplement si le numéro de la case correspond à la solution.

0.4.3 L'aide graphique pour les cases faciles à remplir

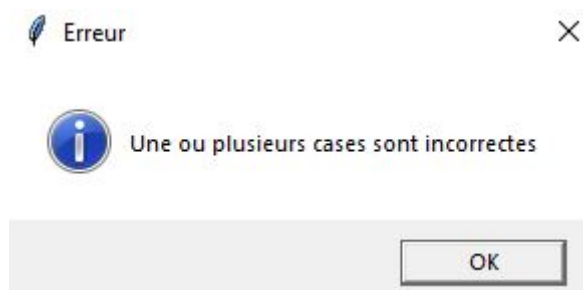
Cette aide utilise exclusivement le filtrage. Elle colorie les cases (vides) dont la valeur semble « la plus facile » à déterminer compte tenu de l'état courant de la grille.

0.4.4 Le système de validation

Cette aide permet de déterminer si un Plateau rempli par l'utilisateur est valide. Celle-ci s'active pour l'ensemble de la grille lors d'un clic sur le bouton (valider) du mode jeu.

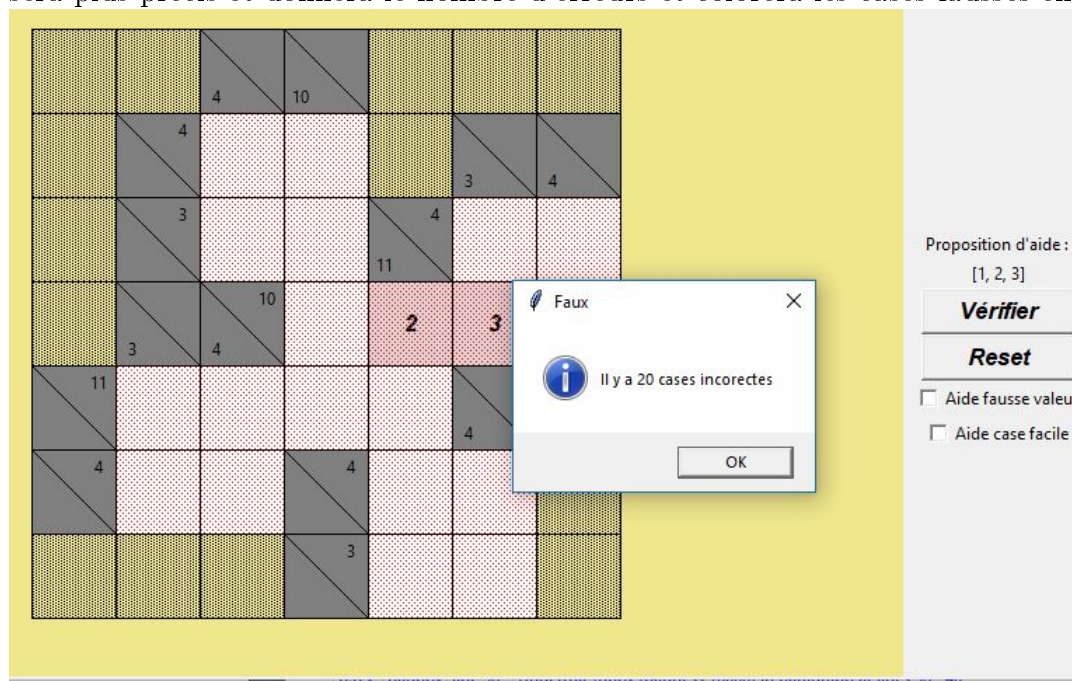
Cas d'une grille sans correction

Dans le cas d'un plateau qui ne possède pas de correction en argument, l'utilisateur doit finir de remplir toutes les cases pour savoir si son remplissage est faux. La fonction `est_solvable` de notre projet analyse le plateau et détecte les incohérences. Une fenêtre apparaîtra donc pour indiquer si le plateau est bien rempli ou non, mais ne donnera pas le nombre d'erreurs et ne les localisera pas.



Cas d'une grille avec une correction

Dans le mode joueur, l'utilisateur a en fait la possibilité de sauvegarder un plateau en tant que correction une fois qu'il l'a terminé et que la grille est correcte. Les fichiers de correction sont de type ".corr". Depuis le mode éditeur, il peut choisir un de ces fichiers pour attribuer une correction à son plateau. Ainsi, lorsqu'il rejouera à la grille de Kakuro, le système de validation sera plus précis et donnera le nombre d'erreurs et colorera les cases fausses en rouge.



La capture d'écran ne permettant pas de visualiser correctement les cases coloriées en rouge, nous vous invitons à tester de vous même grâce au fichier Sauvegarde.dat qui contient une correction.

0.5 Déroulement du projet

0.5.1 Planning/grandes étapes

Concernant les grandes étapes du projet, nous avons en fait respecté les trois grands jalons obligatoires. Nous avons pu nous entretenir deux fois avec notre tuteur : une fois en octobre une autre fois en février. Cela notamment à cause de son arrêt maladie (voir la partie sur nos difficultés).

Jalon 1 : Novembre

Lors de notre premier rencontre avec notre tuteur en octobre, nous avons fait le point sur le sujet et ce qu'il attendait de nous. Notre première idée était de représenter la grille et de pouvoir l'afficher. Nous avons donc développé le jeu du Kakuro. Il nous était alors possible de voir une grille et d'y rentrer des nombres et de valider le jeu. Nous avons ajouté des thèmes pour pouvoir personnaliser notre grille. Comme il s'agissait du prototype, nous nous sommes contentés de ça.

Jalon 2 : Janvier

Nous n'avons pas eu temps de revoir notre tuteur. Comme nous n'avions pas fait toutes les fonctionnalités expliquées lors de la première réunion, nous nous y sommes attaqués. Nous avons donc ajouté le filtrage qui a été une grosse partie dans le projet ainsi qu'un éditeur de grille. Avec cela, la sauvegarde et chargement de grille a été développé. Après ce Jalon, le plus gros du travail avait été fait.

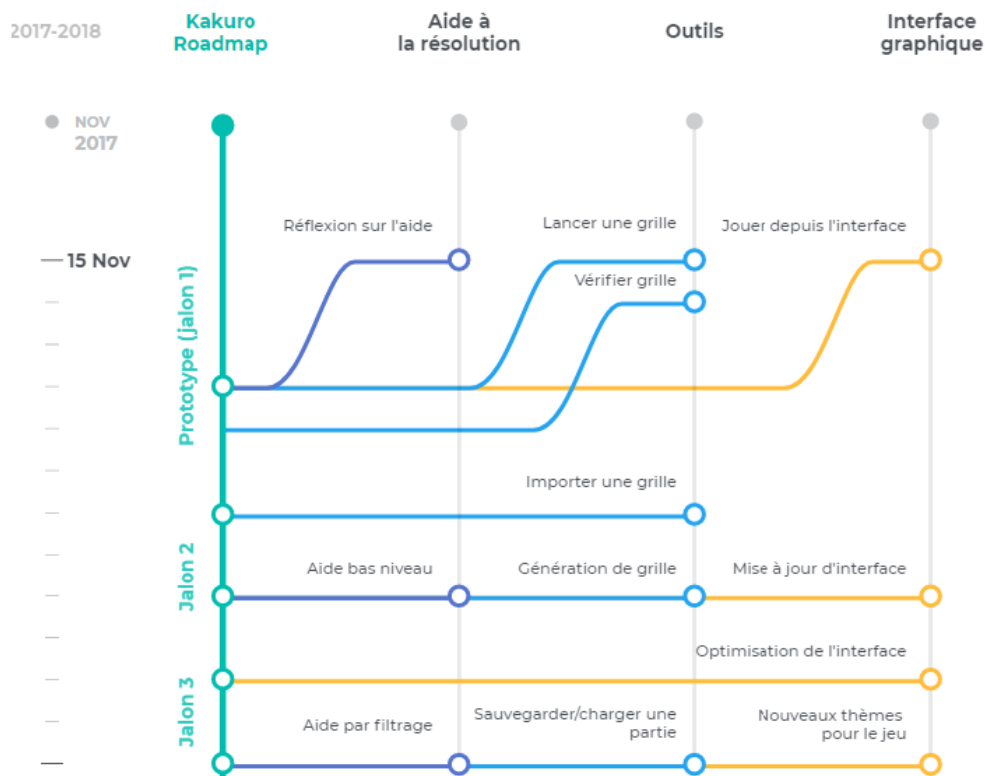
Jalon 3 : Mars

Comme M.Boizumault était absent, nous avons ajouté les aides "cases faciles à remplir" et "cases fausses instantanées". Suite à cela, nous avons donc eu un deuxième et dernier rendez-vous avec M.Boizumault en février. Cette dernière réunion nous a permis de faire le point sur notre projet et avoir son avis après sa longue absence. Tout était bon pour lui. Nous avons donc seulement fait quelques ajustements dans notre code ainsi que rédiger le rapport. Durant cette période, notre emploi du temps était très serré, ce qui a été très difficile pour la rédaction du rapport ou pouvoir ajouter des choses dans le code.

0.5.2 Le backlog

Le backlog a bien été respecté :

Kakuro 2017-2018				
Tâche	Priorité	État	Date de début	Date d'échéance
En tant qu'utilisateur, je peux visualiser une grille de kakuro	Élevée	Terminée	20/10/2017	15/11/2017
En tant qu'utilisateur, je peux obtenir le résultat d'une grille de kakuro déjà enregistrée	Élevée	Terminée	20/10/2017	10/11/2017
En tant qu'utilisateur, je peux demander de l'aide plus ou moins avancée (aide pour une case jusqu'à dire la solution)	Élevée	Terminée	20/10/2017	31/12/2017
En tant qu'utilisateur, je peux choisir un thème pour la grille de kakuro parmi les thèmes proposées par les développeurs	Faible	Terminée	20/10/2017	14/03/2018
En tant qu'utilisateur, je peux jouer au kakuro à la souris et au clavier	Normale	Terminée	22/10/2017	31/12/2017
En tant qu'utilisateur je peux importer une grille de kakuro et y jouer	Normale	Terminée	16/11/2017	01/01/2018
En tant qu'utilisateur je peux sauvegarder et charger une partie de kakuro déjà commencée	Faible	Terminée	01/01/2018	07/01/2018
En tant qu'utilisateur je peux cliquer sur un bouton pour générer aléatoirement une grille de kakuro	Faible	Annulée	07/01/2018	16/01/2018



0.5.3 Nos choix et nos difficultés

Au cours du projet, nous avons été confrontés à plusieurs difficultés. La difficulté majeure a été la longue absence de notre tuteur, Monsieur Patrice Boizumault pour cause de problème de santé. Après avoir rendu notre deuxième jalon le 17 janvier, nous attendions que notre tuteur puisse nous faire un retour afin de voir ce qu'on pourrait faire de plus ou améliorer. Cependant, à ce moment-là, Monsieur Boizumault a été absent de mi-janvier à fin février. Nous ne l'avons revu qu'à son retour fin février juste avant le début des vacances. Après ces vacances, il ne restait que 10 jours pour rendre notre projet.

La grosse difficulté suite à ce problème a été de savoir si nous étions sur la bonne voie. Nous n'étions pas sûr si les fonctionnalités que nous avons faites étaient ce qu'il attendait de nous. Concernant l'avancement nous étions plutôt bien avancé à ce moment-là. En attendant son retour nous avons développé d'autres aides comme le système de mauvaise réponse ou le système de cases les plus faciles à remplir.

A son retour, nous avons eu un retour positif de nos travaux.

Une autre difficulté a été le système de filtrage par CSP. En effet, cette partie a été difficile car plusieurs façons de procéder existent dont une qui demande un bon niveau en aide à la décision.

La dernière difficulté était de trouver des aides à la résolution. À part celles que nous avons proposées, il est difficile d'en trouver d'autres qui soient utiles.

0.5.4 Concernant nos choix

Au-delà du fait de proposer une aide à la résolution, la partie interface graphique était quelque chose de très important dans notre sujet. Nous avons donc décidé d'améliorer l'expérience utilisateur pour jouer au Kakuro. Le choix de thème a été une évidence pour permettre au joueur de jouer dans les conditions qu'il aime.

Nous avons aussi proposé un éditeur de grille. Encore une fois l'interface graphique étant important pour notre projet, nous avons pensé que cela pourrait être utile et pourrait améliorer l'expérience utilisateur. Le joueur peut ainsi créer une infinité de grille et y jouer.

Nous avons passé beaucoup de temps à affiner les détails, comme par exemple l'ergonomie lorsque l'utilisateur appuie sur une case du Kakuro, ou encore lorsque dans l'éditeur il souhaite remplacer une case ou l'interdire rapidement. Comme énoncé précédemment, nous avons peu de temps pour finir le Kakuro ainsi que le rapport pendant les dix jours restant : examens, DM, tout arrivait au même moment alors que nous venions tout juste de discuter avec M. Boizumault à propos de notre projet. Nous avons donc, comme vous avez pu le voir dans le backlog, abandonné certains points que nous allons énumérer :

- La génération aléatoire de grille de Kakuro
- La correction automatique du plateau
- Un moyen plus facile pour créer une case "informative" dans le mode éditeur.

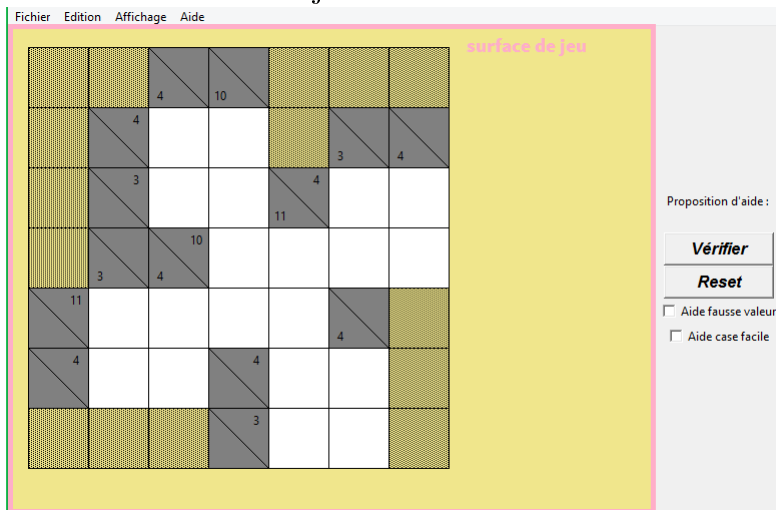
0.6 Conclusion

Pour conclure, nous pouvons dire que l'objectif de notre sujet "Outil graphique d'aide à la résolution de Kakuro" a bien été rempli. En effet, nous avons développé un jeu de Kakuro avec lequel nous pouvons jouer tout en disposant d'aide à la résolution. La partie graphique du jeu a bien été faite : tant pour le mode joueur que pour le mode éditeur. Nous avons codé des outils d'aides à la résolution, comme la validation qui permet de voir nos erreurs : la validation avec une correction qui permet de voir les erreurs de l'utilisateur, et la validation sans correction préalable qui permet la même chose mais avec une précision moins accrue. Nous avons également ajouté une aide qui permet de remplir les cases les plus faciles en bleu ainsi que celles fausses en rouge si on remplit une case. L'aide la plus importante a été le filtrage qui permet d'aider le joueur à remplir une case lorsqu'il fait un clic droit sur la case qu'il décide. Bien évidemment, si le temps nous l'avait permis, nous aurions pu encore améliorer notre projet en implémentant d'autres aides, plus complexes et plus difficiles à programmer, avec un filtrage par support pour améliorer l'efficacité de celui-ci. Cependant coder de tels aides aurait pu faire l'objet d'un sujet à part entière. Nous aurions pu également ajouter une méthode de résolution "presse-bouton" : une résolution de grille en un clic, même si cela n'était pas demandé dans le sujet. Développer une telle aide aurait permis à l'utilisateur de connaître directement la solution. En codant cette résolution en un clic, le bouton d'importation de correction n'aurait plus été d'une grande utilité, ainsi que tout le système filtrage que nous avons mis au point. Le projet annuel nous a permis d'en apprendre plus sur le CSP et de développer en python. Nous avons pu mêler nos efforts en synergie avec nos compétences pour finalement obtenir le résultat que nous voulions. On a beaucoup appris en travaillant sur ce projet.

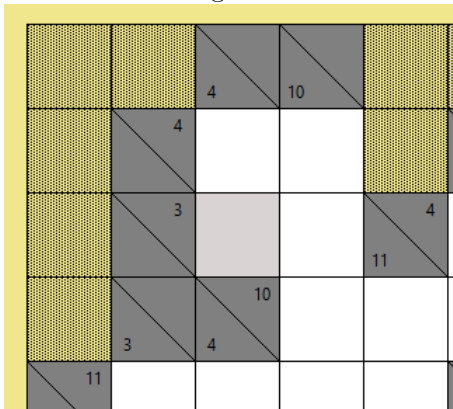
0.7 Notice d'utilisation

0.7.1 Mode jeu

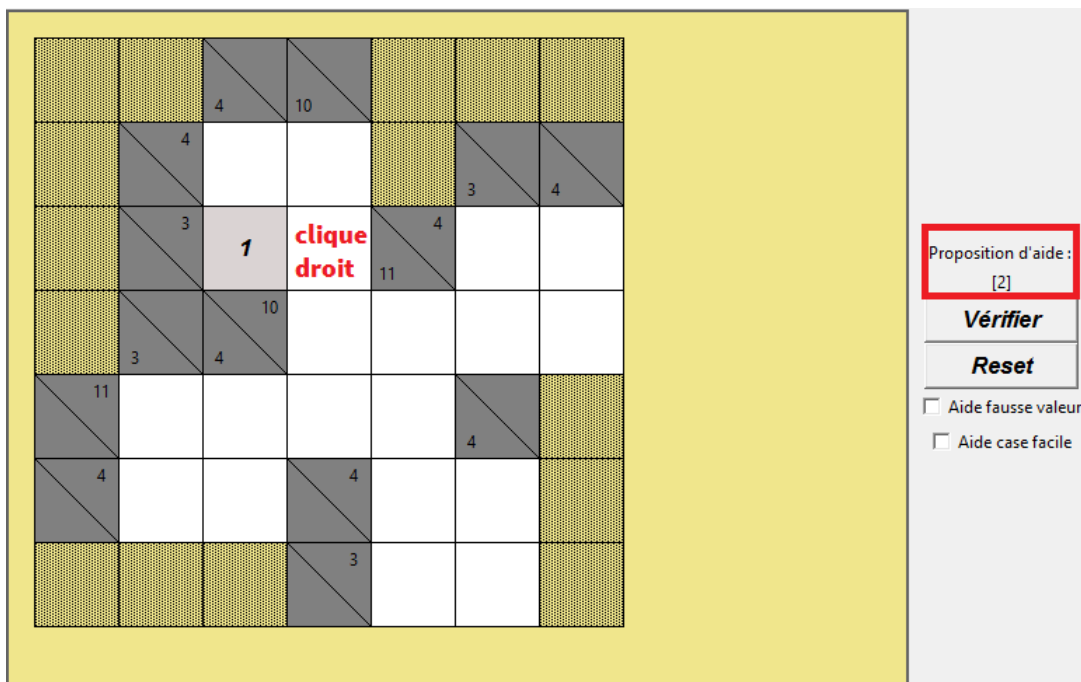
Pour le lancer, compiler kakuro.py.
Voici l'écran du mode jeu :



Pour jouer il suffit de remplir les cases. Pour cela, cliquer sur une case avec le clic gauche. Elle deviendra grise. Il vous suffit alors de taper le nombre voulu à l'intérieur de la case.

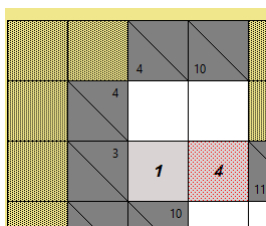


Pour avoir des propositions d'aide il suffit de cliquer sur la case voulue avec le clic droit. Les proposition grâce au filtrage apparaissent alors sur la droite.

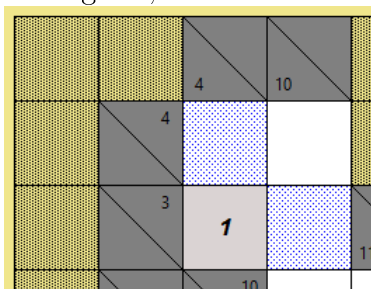


Pour avoir une aide si la case remplie immédiatement est fausse il suffit de cocher "Aide fausse valeur".

Ensuite en remplissant une case elle deviendra alors rouge :

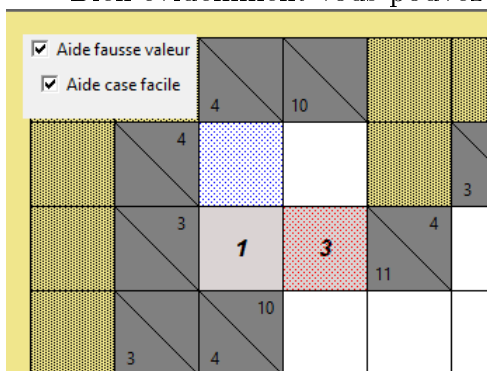


Pour avoir de l'aide afin de savoir si des cases sont faciles à remplir suivant la configuration de la grille, il suffit de cocher "aide case facile" :

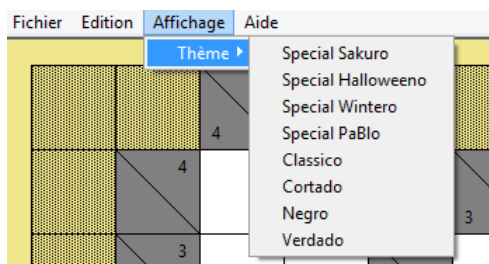


Après les cases apparaissent alors en bleu.

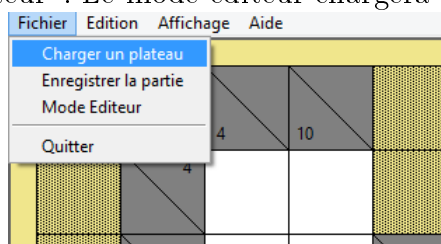
Bien évidemment vous pouvez activer les deux aides en même temps.



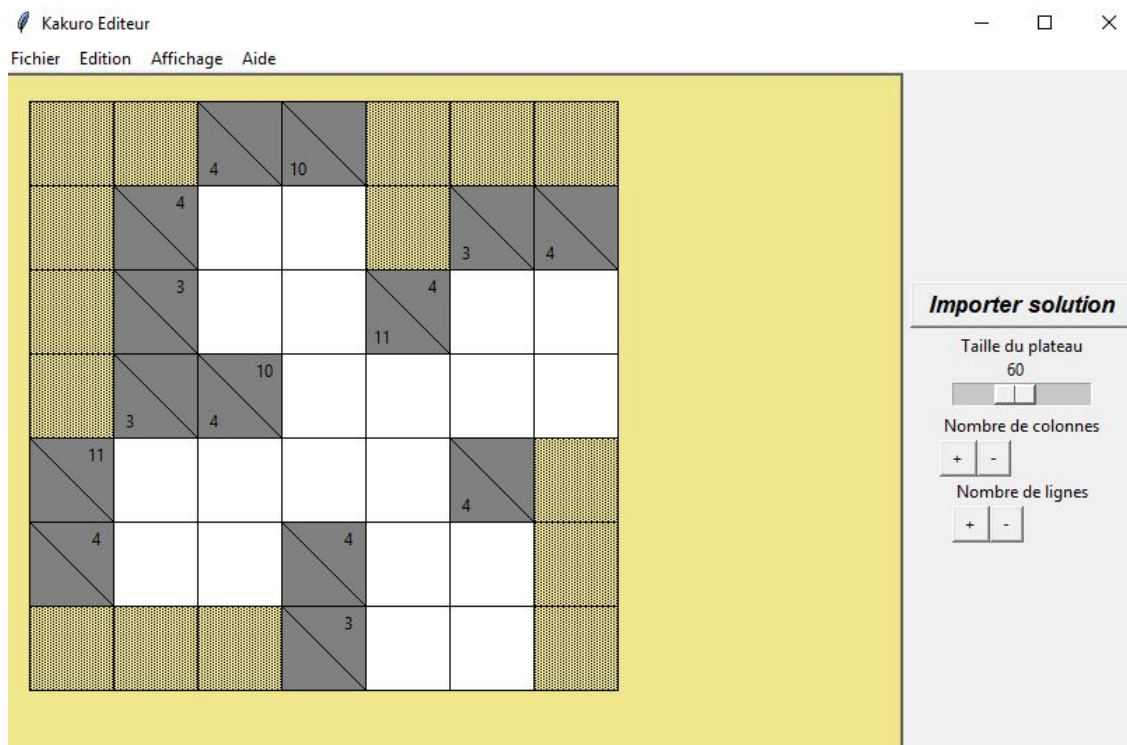
Vous pouvez changer de thème dans Affichage > Thème :



Pour charger une partie allez dans Fichier > Charger une un plateau. Vous pouvez sauvegarder votre partie dans "Enregistrer la partie" et enfin passer au mode éditeur avec "Mode Editeur". Le mode éditeur chargera alors la grille du mode jeu.

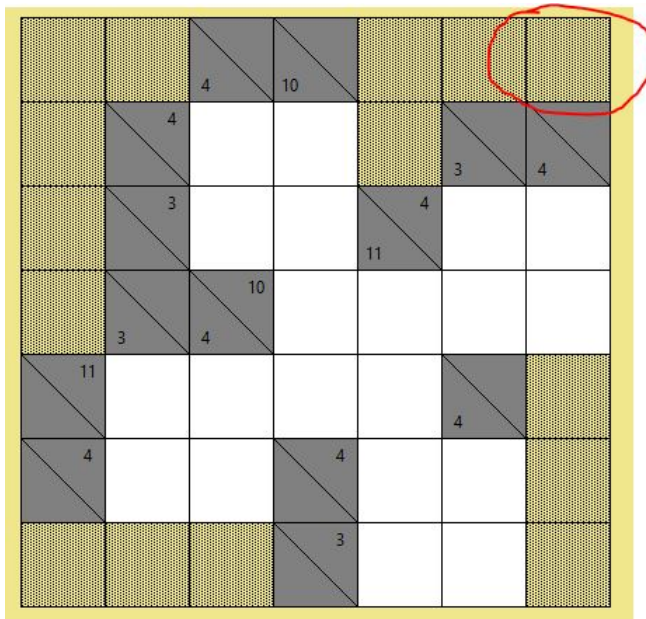


0.7.2 Le mode éditeur



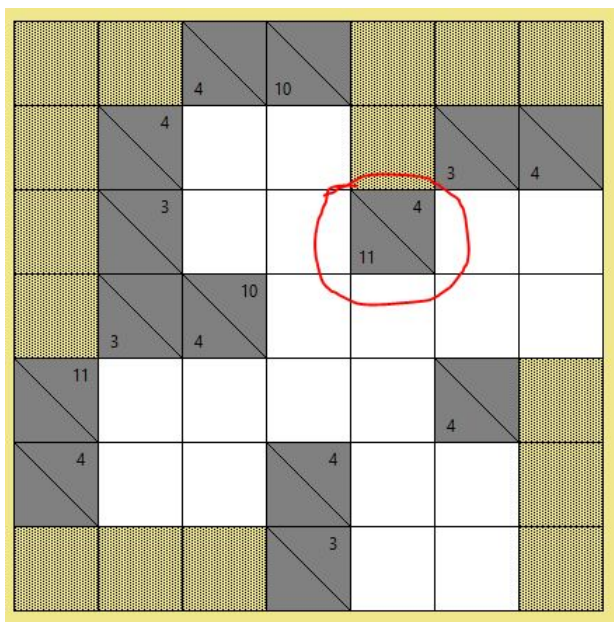
Le mode éditeur permet d'éditer une grille, d'en créer, de la résoudre, de l'enregistrer pour y jouer dans le mode joueur. Vous pouvez y accéder à partir du fichier editeur.py.

Créer une case interdite

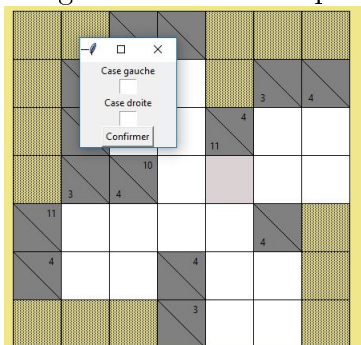


Pour créer une case interdite, faites un clic droit sur la case que vous souhaitez interdire.


Créer une case informative



Pour créer une case informative comme celle d'au dessus, vous devez effectuer un double clic gauche sur la case que vous souhaitez, puis entrer les chiffres qui seront à gauche et à droite.



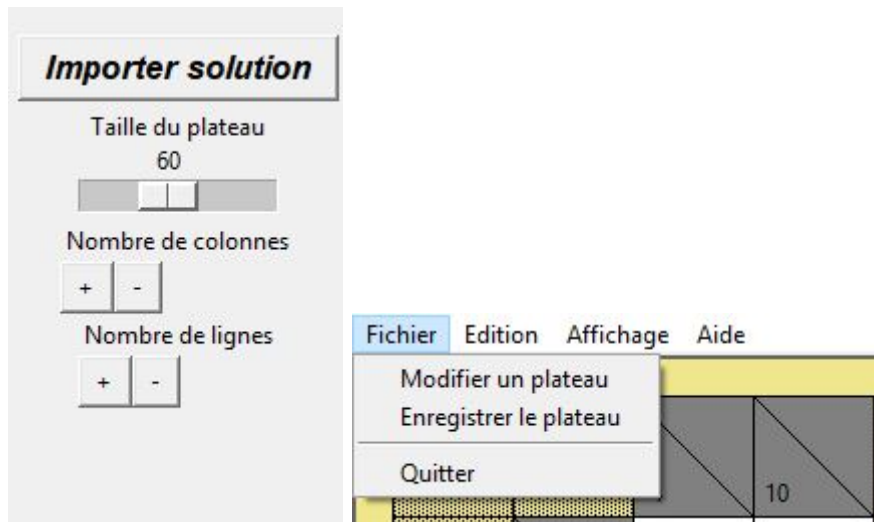


Si vous souhaitez obtenir la case (0,4) , il ne faudra pas oublier de préciser que la case gauche vaut 0.

Efface une case

Pour effacer une case ou pour autoriser une case à être remplie, il faut effectuer un simple clic gauche.

Autres options



Vous pouvez effectuer beaucoup d'autres opérations comme l'ajout de colonnes, de lignes à la grille, mais ces options sont facilement trouvables quand vous lancez l'interface graphique (voir la capture d'écran au-dessus). Il est également possible de redimensionner la taille de la surface de jeu, ce qui est particulièrement utile lorsque l'utilisateur décide par exemple de créer une grille petite ou une grande grille. Cette dimension de la grille sera sauvegardée lorsqu'on sauvegardera le plateau. Dans le mode jeu, le plateau apparaîtra tel que celui sauvegardé dans le mode éditeur.

Vous pouvez appuyer sur le bouton "importer une solution" pour donner une correction à cette grille : cela servira à optimiser l'aide lorsque l'utilisateur jouera en mode jeu.