

SUPINFO Academic Dept.

**Algorithmique et programmation en Python**

Mini - Projet

Pentago

Version 1.0

Last update: 15/12/2015

Use: Students/Staff

Author: Laurent GODEFROY

SOMMAIRE

1 Préambule 3

2 Généralités sur ce projet 3

3 Des algorithmes pour jouer à ce jeu 4

3.1 Initialisation 4

3.2 Rotation d’un quadrant 5

3.3 Test d’un alignement 6

4 Une interface graphique 8

4.1 Affichage du plateau de Jeu 8

4.2 Pose d’un pion 9

4.3 Rotation d’un quadrant 10

4.4 La procédure de jeu 11

5 des bonuS 12

5.1 Sauvegarder une partie 12

5.2 Une animation pour la fin de jeu 12

5.3 Des grilles de différentes tailles 13

5.4 Jeu contre l’ordinateur 13

6 barème indicatif 13

# Préambule

Cet examen est à réaliser par groupes de deux étudiants. Dans l’unique cas où le nombre d’étudiants de la promotion est impair, un et un seul groupe de trois est autorisé.

Toute forme de plagiat ou utilisation de codes disponibles sur internet ou tout autre support, même de manière partielle, est strictement interdite et se verra sanctionnée d’un 0, d’une mention « cheater », et le cas échéant d’un conseil de discipline.

Vous devrez envoyer les codes sources de votre projet par mail à **votre formateur** avant le dimanche 24 janvier 2016 à 23h59, heure locale. Au delà de cette date et heure votre note sera de 0. Vous comprimerez ces codes sources dans une archive au format « .zip » que vous nommerez en respectant la convention **ID-1ADS-votreCampus-MP** (exemple : 66280-1ADS-Tours-MP). L’objet du mail sera ce même nom. Vous mettrez également **en copie** la boîte [1ADS@supinfo.com](mailto:1ADS@supinfo.com). Si vous ne mettez pas cette dernière adresse en copie aucune réclamation ultérieure ne sera recevable.

Ce mini-projet donnera lieu à des soutenances qui se dérouleront la semaine du 25 janvier 2016. Vos horaires de passages vous seront communiqués par votre campus.

Les soutenances sont également par groupes de deux. Elles dureront **20 minutes** pendant lesquelles vous montrerez à votre examinateur le bon fonctionnement de votre programme en en faisant la démonstration. Si vous n’avez pas implémenté le projet dans sa totalité, vous exposerez les parties fonctionnelles.

Pour appuyer votre présentation, vous devrez préparer un fichier de type Powerpoint, dans lesquels vous expliquerez les points du code que vous jugez les plus importants et significatifs. Il n’est pas nécessaire d’envoyer ce fichier à votre examinateur, ce dernier le découvrira le jour de la soutenance. Une communication précisant tout cela vous sera envoyée courant janvier.

Un barème **indicatif** vous est donné dans la dernière partie de ce sujet.

# Généralités sur ce projet

**Remarque importante**: aucun code n’est demandé dans cette partie qui n’est qu’explicative.

Le but de ce mini-projet est d’écrire en langage Python un programme permettant de jouer au célèbre jeu de stratégie « Pentago ».

Si vous n’êtes pas familier avec ses règles, je vous invite à les consulter, par exemple sur Wikipédia : <https://fr.wikipedia.org/wiki/Pentago>

Les droits de ce jeu appartiennent à la compagnie Mindtwister.

Vous aurez peut-être besoin de plusieurs lectures du sujet pour avoir une bonne vue d’ensemble du projet. Prenez donc le temps nécessaire à une bonne compréhension avant de commencer les codes demandés dans les parties suivantes.

Dans la partie 3 on implémentera les algorithmes nécessaires au déroulement du jeu (rotation d’un quadrant du plateau, tests des alignements, etc.).

Dans la partie 4 on rajoutera une interface graphique.

Enfin, dans la partie 5, on proposera quelques bonus et extensions possibles.

# Des algorithmes pour jouer à ce jeu

**Il vous est fortement recommandé de lire l’intégralité de cette partie avant de commencer à coder. Les travaux demandés sont mis en évidence avec une couleur bleue.**

**Remarque importante**: dans cette partie on ne travaillera qu’en mode « console ». Le plateau sera une liste à deux dimensions d’entiers, un 0 représentant une case vide, un 1 un pion blanc du premier joueur, et un 2 un pion noir du second joueur.

## Initialisation

**Dans un fichier que l’on nommera « bases.py », implémenter les sous-programmes suivants :**

* Une fonction prenant en paramètre un entier **n** et retournant une liste à deux dimensions de **n** lignes et **n** colonnes dont toutes les valeurs valent **0**.
* Une procédure réalisant l’affichage des valeurs d’une liste à deux dimensions passée en paramètres. Cette procédure ne servira qu’à vérifier le bon fonctionnement des sous-programmes des parties 3.2 et 3.3.

## Rotation d’un quadrant

**Dans un fichier que l’on nommera « rotations.py », implémenter les sous-programmes suivants :**

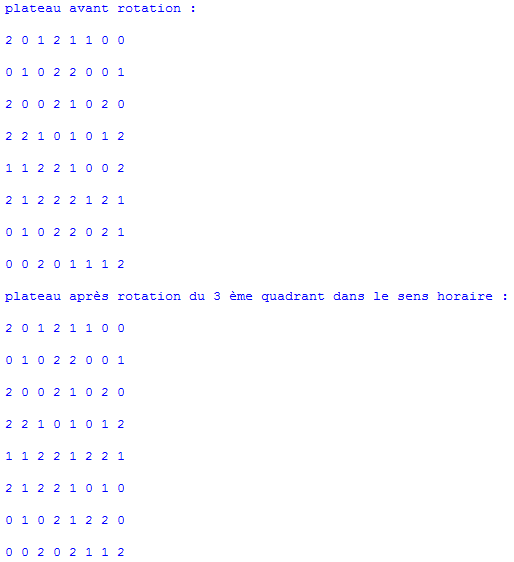
* Une procédure prenant en paramètre un entier **m** et une liste à deux dimensions de **m** lignes et **m** colonnes. Cette procédure réalisera une rotation de cette liste de 90 degrés dans le sens des aiguilles d’une montre.
* Même chose pour réaliser cette fois une rotation de 90 degrés dans le sens contraire des aiguilles d’une montre.
* Une procédure prenant en paramètre un entier **n** supposé pair, une liste à deux dimensions de **n** lignes et **n** colonnes, un entier représentant le numéro du quadrant et un booléen indiquant le sens de rotation. Cette procédure réalisera une rotation de 90 degrés selon le sens passé en paramètre du quadrant passé lui aussi en paramètre. On pourra par exemple adopter la numérotation suivante pour désigner un quadrant :

|  |  |
| --- | --- |
| 1 | 4 |
| 2 | 3 |

**Exemple de fonctionnement**: le code suivant

|  |
| --- |
| from random import randint  plateau = [[randint(0,2) for j in range(8)] for i in range(8)]  print("plateau avant rotation : ")  affichePlateau(plateau)  rotationPlateau(plateau,8,3,True)  print("plateau après rotation du 3 ème quadrant dans le sens horaire : ")  affichePlateau(plateau) |

doit conduire à un résultat similaire à celui-ci :



**Remarque** **importante** : dans le cas où vous n’arrivez pas à implémenter de telles procédures pour des listes à **n** lignes et **n** colonnes, **n** étant quelconque et pair, vous pourrez implémenter une version simplifiée ne fonctionnant que pour une valeur de **n** égale à **6**. La notation de ces procédures s’en trouvera fortement impactée mais vous pourrez quand même alors réaliser la suite du projet.

## Test d’un alignement

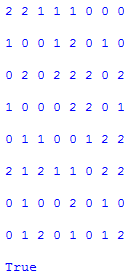
**Dans un fichier que l’on nommera « alignements.py », implémenter les sous-programmes suivants.**

* Une fonction prenant en paramètre un entier **n**, une liste à deux dimensions de **n** lignes et **n** colonnes, un entier **p** et un entier **j**. Cette fonction retournera **True** si dans au moins une des lignes est présent un alignement de **p** valeurs consécutives égales à **j**.
* Même chose pour tester cette fois-ci un alignement vertical.
* Même chose pour tester cette fois-ci un alignement dans la diagonale allant du bas à gauche vers le haut à droite.
* Même chose pour tester cette fois-ci un alignement dans la diagonale allant du haut à gauche vers le bas à droite.
* Une fonction prenant en paramètre un entier **n**, une liste à deux dimensions de **n** lignes et **n** colonnes, un entier **p** et un entier **j**. Cette fonction retournera **True** si dans au moins une des quatre directions est présent un alignement de **p** valeurs consécutives égales à **j**.

**Exemple de fonctionnement**: le code suivant (pour un plateau déjà créé)

|  |
| --- |
| affichePlateau(plateau)  print(alignement(plateau,8,4,1)) |

doit conduire à un résultat similaire à celui-ci :



**Remarque** **importante** : dans le cas où vous n’arrivez pas à implémenter de telles fonctions pour des listes à **n** lignes et **n** colonnes, **n** et **p** étant quelconques, vous pourrez implémenter une version simplifiée ne fonctionnant que pour une valeur de **n** égale à **6**, et de **p** égale à **4**. La notation de ces fonctions s’en trouvera fortement impactée mais vous pourrez quand même alors réaliser la suite du projet.

# Une interface graphique

**Il vous est fortement recommandé de lire l’intégralité de cette partie avant de commencer à coder. Les travaux demandés sont mis en évidence avec une couleur bleue. On devra nécessairement utiliser la librairie graphique Pygame. L’usage d’une autre librairie ne sera pas pris en compte.**

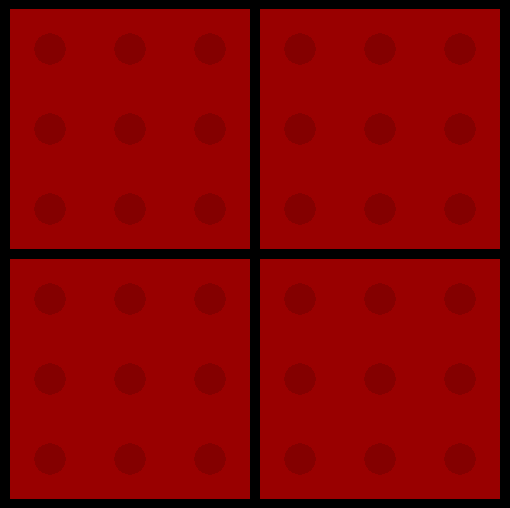
**Remarque 1**: dans cette partie on continuera à modéliser le plateau de jeu par une liste à deux dimensions d’entiers, un 0 représentant une case vide, un 1 un pion blanc du premier joueur, et un 2 un pion noir du second joueur.

**Remarque 2** : on se restreindra dans cette partie à des plateaux de 6 lignes et 6 colonnes, et à des alignements de 5 pions. C’est-à-dire aux règles originelles du jeu.

**Remarque 3** : vous êtes libre de désigner votre jeu comme bon vous semble tant que les fonctionnalités sont présentes. Les captures d’écran de cette partie ne sont là que pour vous donner une idée.

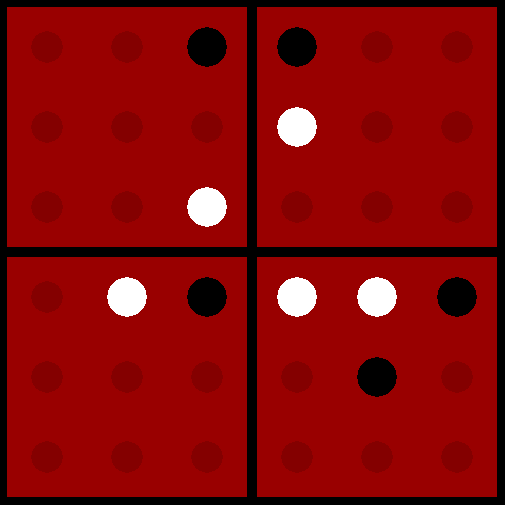
## Affichage du plateau de Jeu

**Dans un fichier que l’on nommera « pentagoGUI.py », implémenter une procédure permettant de dessiner un plateau de jeu qui devra ressembler initialement à cela :**



* Cette procédure appellera plusieurs fois une autre procédure chargée elle de ne dessiner qu’un seul quadrant.
* Elles prennent toutes deux en paramètres une liste à deux dimensions représentant le plateau de jeu, ainsi que la surface sur laquelle il sera dessiné.

Le même plateau après quelques tours de jeu aura une allure similaire à celle-ci :



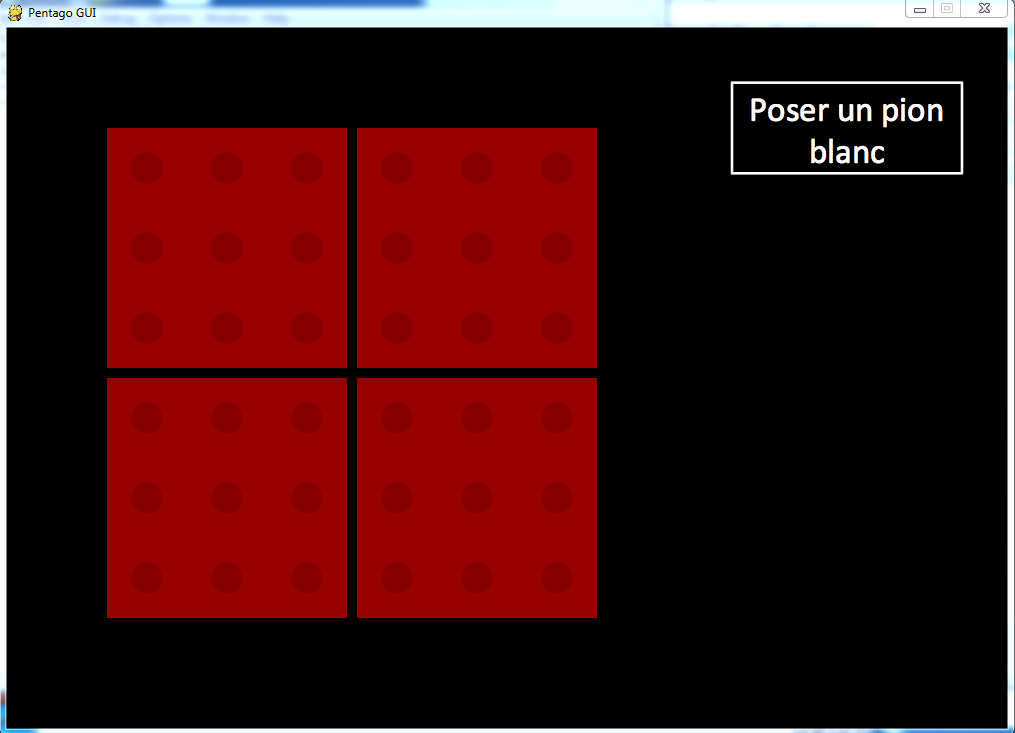
**Remarque**: dans la suite, lorsque le plateau sera modifié, on ne le réaffichera pas complètement, on se contentera de dessiner à nouveau le quadrant impacté.

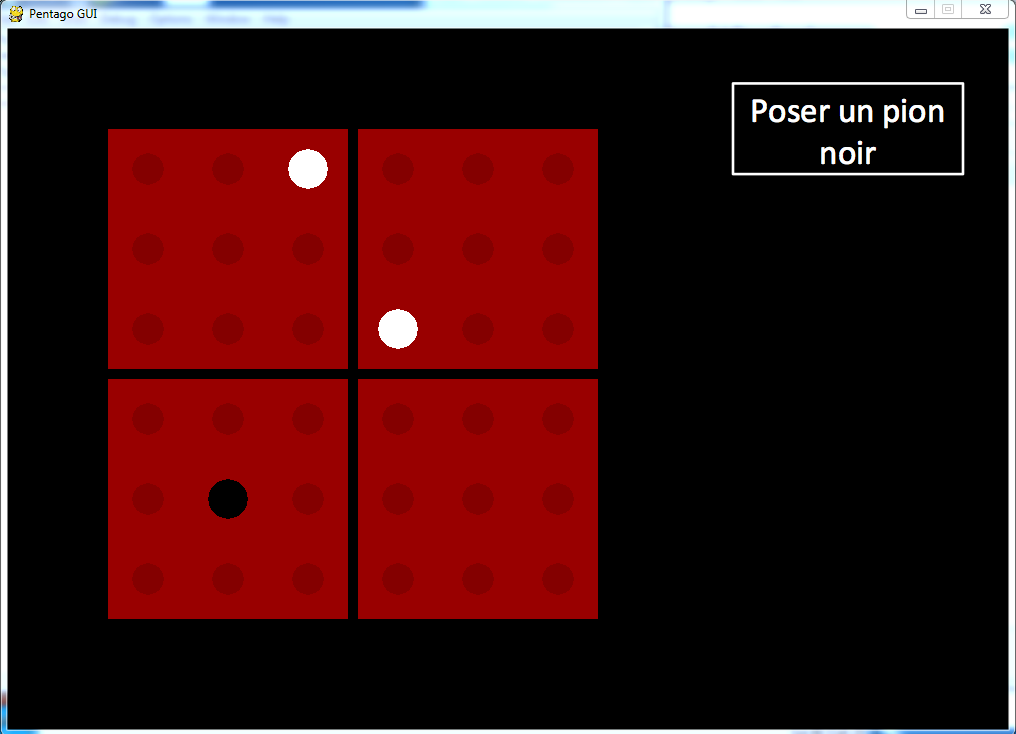
## Pose d’un pion

**Implémenter dans le fichier précédent une procédure permettant à un joueur de poser un pion sur le plateau :**

* Cette procédure prend en paramètre une liste à deux dimensions représentant le plateau de jeu, la surface sur laquelle il sera dessiné et un entier indiquant quel joueur est en train de jouer (cet entier vaut donc 1 ou 2).
* Le choix d’une case se fera à la souris, en prenant bien sûr garde à vérifier que la case en question est vide.
* Avant la pose on indique le joueur dont c’est le tour.

**Exemples de rendu**: (les exemples sont indépendants)



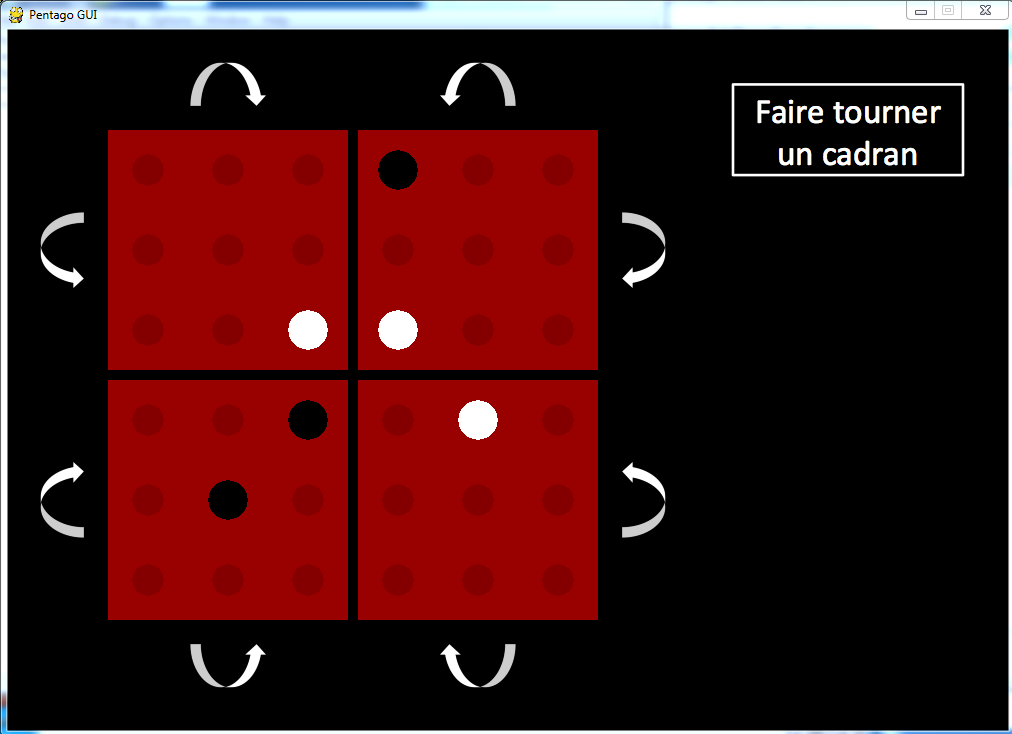


## Rotation d’un quadrant

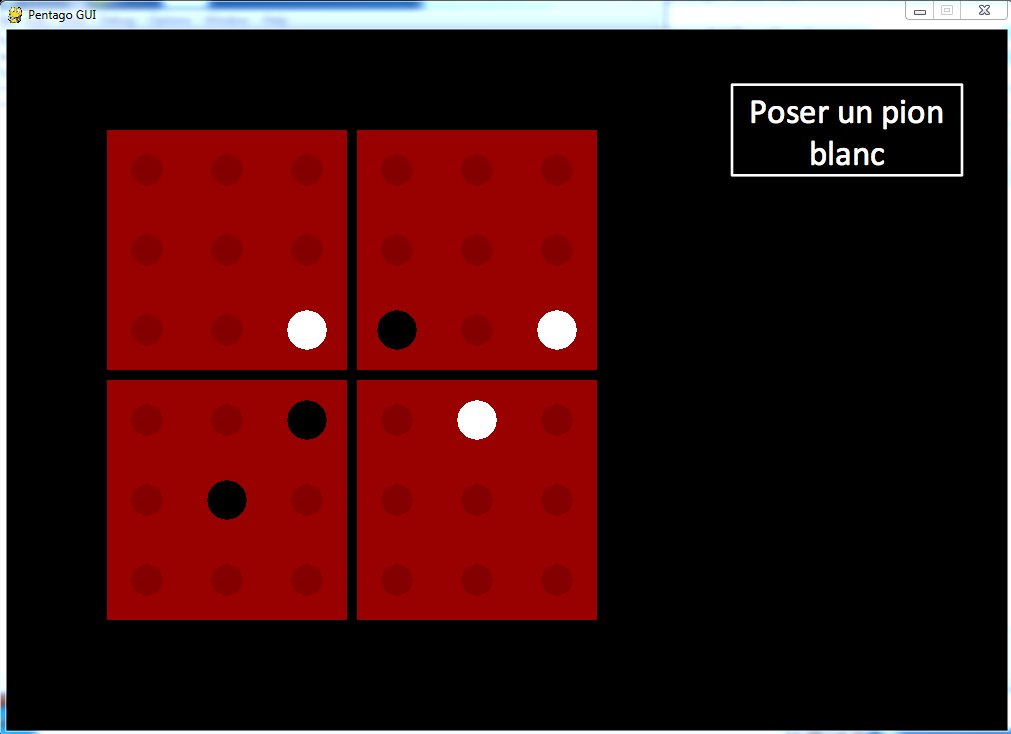
**Implémenter dans le fichier précédent une procédure permettant à un joueur faire tourner un des quadrants :**

* Cette procédure prend en paramètre une liste à deux dimensions représentant le plateau de jeu, ainsi que la surface sur laquelle il sera dessiné.
* Huit flèches seront affichées, et le choix de l’une d’entre elles se fera à la souris.
* Avant ce choix on indique qu’il faut choisir un quadrant.
* Une fois ce choix fait, on efface les flèches.

**Exemple de rendu** : le joueur ayant les pions noirs vient de poser un pion, il doit maintenant choisir un quadrant et un sens de rotation



Et après le choix de faire tourner le quadrant 4 dans le sens contraire des aiguilles d’une montre, le plateau devient :

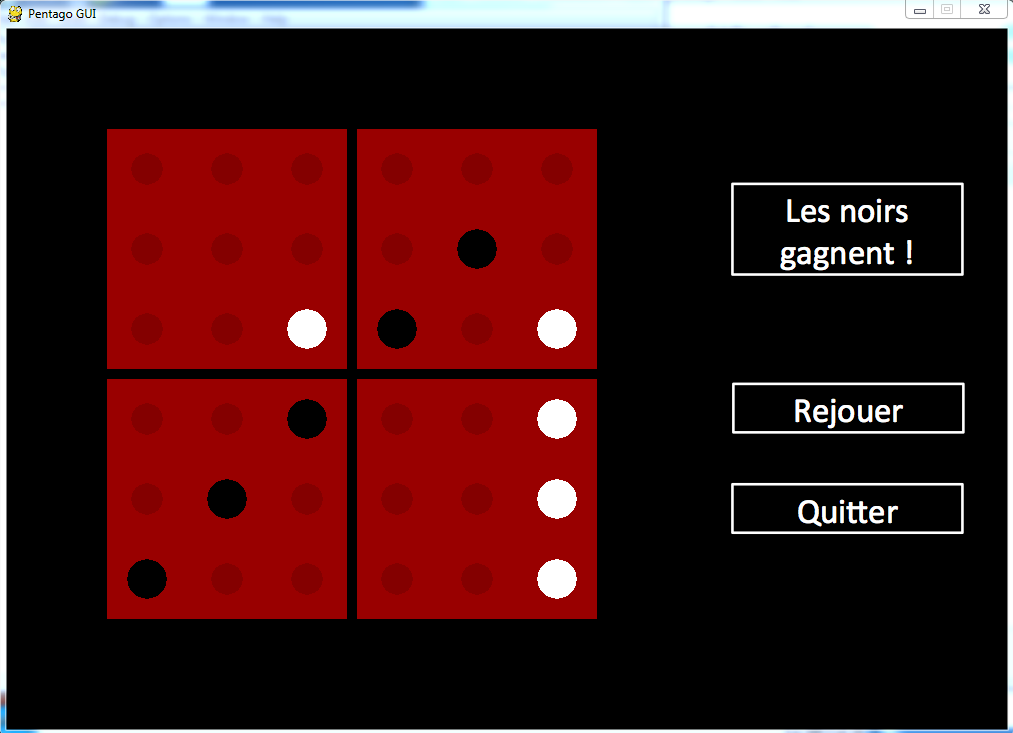


## La procédure de jeu

**Implémenter dans le fichier précédent une procédure permettant de jouer au Pentago avec une interface graphique :**

* Cette procédure devra créer une surface graphique, puis faire jouer alternativement les deux joueurs jusqu’à que l’un d’eux réalise un alignement d’au moins 5 pions. Ou que le plateau soit rempli, ce qui signifie alors qu’il y a match nul.
* Le joueur qui a les pions blancs commence.
* Le résultat de la partie sera affiché, et on proposera ensuite de jouer de nouveau ou de quitter le jeu.

Voici un **exemple** de fin de partie :



# des bonuS

Vous êtes libre d’implémenter zéro, un ou plusieurs des bonus suivants.

## Sauvegarder une partie

Rajouter une option permettant de sauvegarder l’état d’une partie dans un fichier texte.

Implémenter également une possibilité de reprendre le cours d’une partie mémorisée dans un fichier texte.

## Une animation pour la fin de jeu

Rajouter une animation visuelle et sonore (se documenter sur ce dernier point) afin d’indiquer la victoire ou la défaite.

## Des grilles de différentes tailles

Proposer le choix entre des grilles de différentes tailles.

## Jeu contre l’ordinateur

Implémenter un mode de jeu contre l’ordinateur, celui-ci jouant de façon aléatoire.

# barème indicatif

Ce barème peut être amené à évoluer, il n’est donc qu’i**ndicatif**.

* Partie 3: 15 points
* Partie 4 : 25 points
* Bonus : 10 points

Ce qui fait un total de 40 points, votre soutenance étant évaluée sur 20 points. La note totale sera alors ramenée sur 20 points par proportionnalité.