20 septembre 2025

Dubel Nathan, 26K

Gymnase de Bienne et du Jura Bernois

Patrick Hochuli, informatique

Code du Bao game

Un jeu traditionnel africain

Une image contenant texte, cercle, capture d’écran, diagramme

Le contenu généré par l’IA peut être incorrect.

Table des matières

[1. Introduction au jeu 3](#_Toc210646968)

[1.1 Origine du jeu 3](#_Toc210646969)

[1.2 Explication et règles du jeu 3](#_Toc210646970)

[2. Commencement et déroulement du travail 3](#_Toc210646971)

[2.1 Démarche et logique 3](#_Toc210646972)

[2.2 Le code en quelques lignes 4](#_Toc210646973)

[3. Critique du travail effectué 5](#_Toc210646974)

[3.1 Déroulement du travail 5](#_Toc210646975)

[3.2 Travail complet 6](#_Toc210646976)

[3.3 Chemin parcouru 6](#_Toc210646977)

[4. Sources et bibliographie 7](#_Toc210646978)

[5. Remerciements 7](#_Toc210646979)

# 1. Introduction au jeu

## 1.1 Origine du jeu

Le Bao game est un jeu de stratégie originaire d’Afrique de l’Est, pratiqué aujourd’hui essentiellement dans les pays africains parlant le swahili. L’origine du nom, un nom pour la plupart inconnu, provient de la traduction directe de « graine » en swahili, soit « bao ». Evidemment, il existe plusieurs variantes de ce jeu, aussi appelé mancala, ici je me suis inspiré de la variante « kujifunza » qui veut dire « pour les touristes » en swahili, la variante étant la plus simple à comprendre.

## 1.2 Explication et règles du jeu

Le bao game se joue à deux, les deux joueurs étant des adversaires. Chacun des joueurs dispose de deux rangées de huit trous, chaque trou étant rempli de deux graines, soit trente-deux graines par joueur. Le but du jeu est simple : éliminer toutes les graines de son adversaire. Ainsi, chacun leur tour, les deux joueurs vont jouer un coup qui leur permet d’approcher ce but.

Un joueur, pour jouer, doit désigner un trou et déplacer les graines de ce trou de manière à les distribuer une par une dans les trous suivants, dans le sens inverse des aguilles d’une montre. Si la dernière graine atterrit dans un trou non vide (donc rempli d’une graine minimum), alors le joueur répète le processus avec les graines se trouvant dans le dernier trou. Si la dernière graine atterrit dans un trou vide, alors le tour du joueur s’arrête. Cependant, si la dernière graine atterrit dans un trou vide de la rangée du dessus et que le trou en face correspondant aussi à la rangée du dessus de l’adversaire est rempli d’un certain nombre de graines, alors le joueur ayant initialement joué récupère les graines du trou de son adversaire et les élimine de la partie.

Cas spécial : Le premier tour, le joueur n’a le droit de désigner un trou étant situé que sur la rangée du dessus.

# 2. Commencement et déroulement du travail

## 2.1 Démarche et logique

Tout d’abord, je me suis dit qu’il fallait que je code le plateau et le jeu. Une fois programmé, il fallait que je m’occupe des graines : d’abord les placer dans le jeu et ensuite réussir à les déplacer en cliquant dessus. J’y étais vraiment allé étape par étape où je ne déplaçais qu’une seule graine au début, puis les 32 graines d’un joueur, puis les 64 de tout le jeu. Cela m’a pris beaucoup de temps, étant donné que les graines se déplacent de manière rectangulaire (il y a deux rangées) mais en fin de compte j’ai réussi et je suis passé à la suite : il fallait que je réussisse à coder la capture de graine. Cela m’a pris moins de temps que le déplacement, et j’en ai profité pour afficher le nombre de graines capturées par les joueurs en dehors du plateau. Il me restait alors un problème à régler : les règles. Effectivement, mon jeu fonctionnait mais le tour des joueurs n’était pas respecté et la fin d’une partie n’était pas programmé non plus. Il fallait aussi que je code la règle qui disait qu’un joueur ne peut, qu’au premier tour, désigner un trou se situant uniquement sur la rangée du dessus. Cela a aussi été long, il a fallu que je code des messages et que je bloque le jeu en cas de non-respect des règles. Mais, je m’en suis sorti vivant, toujours avec l’aide de mon mentor et j’ai pu enfin terminer mon jeu. Cependant, une petite entorse encore faisait obstacle à mon rendu final : les graines se déplaçaient mais était affichée uniquement la position finale. Cela était problématique car un joueur qui désigne un trou a alors du mal à suivre les déplacements et ne peut se rendre compte d’une erreur de calcul. J’ai voulu donc me lancer dans le code des animations des graines. Au bout d’un certain temps passé dessus, j’ai réalisé que cela revenait à modifier mon code entier, et le recommencer. J’ai alors décidé de coder une pause, à chaque déplacement, afin que le joueur puisse suivre quand même.

Finalement, j’ai finalisé le projet en modifiant la taille des graines pour qu’on puisse mieux voir et rajouté des couleurs pour les deux adversaires (rouge et bleu).

## 2.2 Le code en quelques lignes

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.Pour faire fonctionner toute cette histoire de déplacements de graines, de trous et de captures, j’ai décidé d’utiliser le code des matrices, chaque trou correspondant à une case de la matrice et chaque graine correspondant au contenu de ces cases, donc à la valeur des cases de la matrice.

Ainsi, pour désigner une case de la matrice, il me suffit d’utiliser « jeu[x][y] » x étant la ligne et y la colonne souhaitée.

Ensuite, pour qu’une graine se déplace lorsque je clique dessus, j’ai créé un event que j’ai relié à la fin avec le clic de souris grâce au code « bind ».

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.Mais, la question qui se pose c’est comment relier les cordonnées d’un clic au numéro de case correspondant ? J’ai codé de manière que chaque trou soit distancé d’exactement 100. Ainsi, il me suffisait de diviser par cent les cordonnées d’un clic pour obtenir le numéro de ligne et le numéro de colonne correspondant.

Comme ça, jeu[li][co] correspond exactement au contenu du trou sur lequel on clique. C’est beaucoup plus simple comme ça, par exemple, pour déplacer les graines d’un trou, il suffit d’ajouter un à la valeur des trous suivants, les trous étant jeu[li][co-n] (n correspond au nombre de graines du trou initial). Et si le nombre de graines est plus grand que le nombre de cases qui reste, alors il faut distribuer les graines restantes dans les trous de la rangée du bas ou du dessus. Ainsi, à la fin d’un déplacement, il me suffisait de dire que le dernier trou devient alors le premier trou, comme si on avait cliqué dessus. Le code était alors lancé en boucle, jusqu’à ce que la dernière graine atterrisse dans un trou vide. J’ai alors ajouté ce code à la fin de mon event :

Une image contenant texte, capture d’écran

Le contenu généré par l’IA peut être incorrect.

Ce code dit, que si la graine atterrit dans un trou vide (donc que la valeur du dernier trou = 1), et que ce trou est situé sur la 3ème rangée, on capture automatiquement les graines qui sont en face, comme le disent les règles du jeu. Ainsi, les graines capturées sont enregistrées sur le compteur de chaque joueur et lorsqu’un joueur arrive à capturer les 32 graines de son adversaire, la partie se termine. Finalement, que le trou vide soit sur la 3ème ou la 4ème rangée, le code dit qu’on sort de la condition et qu’on recommence tout au début, mais pour le joueur suivant. (break)

Pour modifier la valeur d’une case de matrice, j’ai utilisé le code « itemconfig », qui change la valeur d’un objet désigné.



Ainsi, la valeur d’une case de matrice est modifiée, mis à jour et affiché en temps réel grâce au lien entre la matrice et le jeu que j’ai codé juste avant.

Une image contenant texte, capture d’écran, Police, logiciel

Le contenu généré par l’IA peut être incorrect.Finalement, il me restait plus qu’à coder les messages qui apparaissent lorsqu’un on joue un coup illégal et pour cela, j’ai utilisé des messagebox askokcancel. J’ai codé comme ça : Une certaine variable a une certaine valeur. Si cette variable a la valeur attribuée, alors on entre dans la condition. Dans la condition, si le clic est effectué sur une mauvaise rangée, alors un message apparait. Sinon, une nouvelle valeur est attribuée à la variable, correspondant à une autre condition. Dans cette autre condition, le principe est le même, mais elle est adaptée pour l’autre joueur. Au début de la partie, le principe est le même mais dans les rangées illégales, on y trouve aussi la rangée du dessous.

Ainsi, à chaque tour, si le coup est illégal, un message apparait et si le coup est légal, alors on change de joueur. J’ai aussi rajouté un code qui fait apparaitre un message lorsqu’on clique sur une case vide, pour éviter toute confusion. A la fin, un message de fin de partie termine la partie en affichant le vainqueur. Pour cela, j’ai juste fait en sorte d’apparaitre un message lorsque le nombre de graines capturées atteint les 32.

# 3. Critique du travail effectué

## 3.1 Déroulement du travail

Je n’ai eu aucun mal à commencer mon travail de maturité. Pour moi, c’était clair ce que j’avais à faire et je me suis tout de suite mis au travail. Cependant, quand j’ai eu mon deuxième rendez-vous avec mon mentor, M. Hochuli, j’ai vite compris que ce n’était pas aussi simple que je ne le pensais. Effectivement, j’avais vu très gros, je voulais, en plus du code, programmer une intelligence artificielle. Je pensais que j’aurai eu le temps de tout faire mais j’ai oublié un détail important : je n’avais jamais codé, j’étais sans expérience.

Mon mentor m’a affirmé qu’il fallait d’abord que je programme le jeu en tant que tel avant de m’attaquer à l’IA, et je l’ai donc écouté. Au début, j’étais motivé. Je travaillais énormément dès que je pouvais mais tout était encore flou. En effet, je prenais des chemins douteux, souvent des culs-de-sac. Alors je prenais rendez-vous avec mon mentor, et une fois j’ai dû recommencer à zéro. Il me dit qu’il faut que je code pas à pas, étape par étape afin d’avancer au lieu de recommencer. J’ai donc commencé mon travail de maturité avec ce raisonnement-là. J’avançais bien, jusqu’à mon premier obstacle : le déplacement de plusieurs graines. N’ayant aucune expérience, je n’avais aucune idée de comment faire et donc mon premier réflex c’était d’aller demander à mon mentor. Il m’a sorti quelque chose que je n’avais jamais vu et j’ai tout de suite perdu la motivation de continuer. Je ne comprenais rien alors pendant quelques semaines je n’ai rien touché. Et, en remettant la main à la pâte, j’ai eu un déclic, ce qui m’a permis de retrouver ma motivation !

Ça a été comme ça plusieurs fois, avec le déplacement « circulaire » des graines, la capture des graines, les tours des joueurs à respecter et finalement, les animations. Les animations ont été la seule fois où je n’ai pas trouvé de déclic, alors j’ai pensé à une alternative.

Malheureusement, j’ai dû travailler pendant la plupart des vacances, les vacances étant les seuls moments que j’avais à disposition pour bien avancer. Et la fin était un peu plus intense évidemment, il est toujours compliqué de superposer TM et tests écrits.

## 3.2 Travail complet

Ce qui a bien fonctionné dans ce travail, c’était surtout la motivation et le plaisir que je prenais à coder, mélangeant logique et projection. Je tiens aussi à remercier mon mentor pour sa présence active et son aide précieuse. Sans lui, je n’aurais jamais pu avancer comme je l’ai fait, aussi efficacement.

En revanche, ce qui a moins bien fonctionné, c’est la mise en place d’un travail régulier. En effet, j’avais du mal à travailler sur mon projet en même temps d’étudier pour l’école et d’un autre côté, trouver la motivation pour mon TM après une ou deux semaines de tests intensifs. Un autre point aussi, c’est que, en code, il est compliqué de trouver des sources qui permettent de traduire une logique dans notre tête avec un langage python. La plupart du temps, j’étais obligé de demander à mon mentor comment j’étais censé faire et ça prenait du temps avant que j’obtienne la réponse car on prenait souvent rendez-vous.

En conclusion, si je devais refaire ce travail de maturité, j’aurais dès le début mis en place un plan de travail régulier prenant en compte aussi les études du gymnase, et j’aurais réfléchi plus précisément à tout le code dont j’avais besoin, afin que je ne doive pas attendre la réponse de mon mentor à chaque fois. Evidemment, je n’aurais jamais pu prévoir exactement le code qu’il me fallait, mais un travail un peu plus structuré m’aurait bien aidé.

## 3.3 Chemin parcouru

En commençant ce projet, je n’avais aucune expérience en code. Certes, il me restait quelques bases des cours facultatifs de robotique de la deuxième année, mais, n’étant pas en OC informatique, je n’avais aucune compétence de programmer un code comme celui-ci. Cependant, au fur et à mesure que le projet avançait, j’apprenais à coder et ce, de manière de plus en plus complexe. Je suis parti de coder le visuel du jeu à coder les déplacements, les relations et la logistique des graines en complexifiant le code à chaque fois. Pour moi, ce projet était instructif et il m’a permis d’apprendre des choses tout au long du projet, un peu comme en OC informatique.

Pour conclure, je suis très content et fier de ce que ce projet m’a permis d’apprendre, je me sens nettement plus cultivé à la fin qu’au début et je pense que si je devais refaire le code à zéro, cela me prendrait beaucoup moins de temps.

# 4. Sources et bibliographie

Pour ce projet, je n’ai pas beaucoup de sources. En effet, la plupart du travail étant de la logique, je dirais que les sources venaient principalement de ma tête. Cependant, avant de commencer, j’ai lu et pris part du document que mon mentor avait mis sur teams, un document explicatif des différents codes de base en python. Au début, je m’appuyais sur ce document mais plus mon travail avançait, plus j’avais besoin de codes python complémentaires. Alors j’effectuais quelques recherches, sans jamais vraiment trouver ce qu’il me fallait. Je demandais donc presque toujours à mon mentor la suite, et c’est pour ça que je dirais qu’il fait partie de mes sources principales.

Parfois, j’utilisais ChatGPT. Non pas pour écrire du code à ma place, mais pour connaitre le code nécessaire qui n’était pas marqué dans le document ou alors comprendre pourquoi quand mon code crashait.

<https://chatgpt.com/c/6836bf77-f210-8000-91e5-da5d0daf76cc>

<https://chatgpt.com/c/68aac3e4-b6d0-832d-b381-7050a05a2c25>

<https://chatgpt.com/c/6858f2d9-c394-8000-ba9a-051f6916965f>

<https://chatgpt.com/c/684c599c-419c-8000-9cf2-05c80b096728>

Voici la seule vidéo tutoriel que j’ai regardé, au tout début, pour comprendre la mécanique du « bind » : <https://www.youtube.com/watch?v=Jb5Df2ul41M>

# 5. Remerciements

Un grand merci pour mon mentor qui m’a soutenu et aidé jusqu’à la fin dans mon travail, et qui m’a mis dans la bonne direction dès le début. Durant ce projet, j’avais énormément de questions et de doutes et mon mentor a su répondre présent à chaque fois.

Merci à mes parents, qui ont su m’aider lorsque je ne comprenais pas ce qui ne marchait pas, et qui m’ont permis de garder ma motivation lorsque je la perdais.

Et merci à mes amis qui m’ont soutenu et aidés pour me motiver à avancer.