	REPRESENTATIONS			CONCLUSION
	Description des représentations	avantages	inconvénients	
PIÈCES	Rep1 : Structure unique par type de pièce Ex : avoir une struct Roi, une struct Pion etc..	+ Modulable si élément à rajouter ou modifier +Les pièces sont type-checkées automatiquement	- Les pièces n'auront pas le même type - Promotion de pions peu pratique	Pour les pièces, on a choisi une structure globale parce que cela permet une meilleure modifiabilité si des nouvelles fonctionnalités sont à ajouter, et permet d'avoir qu'une seule déclaration de structure pour l'entièreté des pièces. Cette représentationp permet de contenir le type, la couleur et éventuellement d'autres choses assez facilement.
	Rep2 : Structure globale piece avec un attribut typeDePiece	+ Modulable si élément à rajouter ou modifier + Les pièces ont le même type	- Devoir vérifier le type de la pièce manuellement	
	Rep3 : Par énumération/string constant	+ Facilement implémentable	- Ne permet pas de décrire les déplacements - Moins lisible -Très peu modulable - Moins pratique pour les couleurs - Devoir vérifier le type de la pièce manuellement	
PLATEAU	Tableau 2 dimensions 8*8 de pièce	+ Manipulable facilement, pas besoin de déclaration spéciale du tableau	- Ne contient contient que le plateau, et pas l'état complet du jeu.	Pour le plateau on a choisi un tableau 2 dimensions, parce que ne demande pas tant d'espace mémoire (64*taille de pièce), et permet un accès facile à partir des coordonnées.
	Structure avec un tableau 8*8 et potentiellement d'autres attributs comme les points des 2 joueurs. Qui Représenterait le jeu en général	+ Peut contenir plus de choses que juste le plateau	- Nécessite une architecture du code un peu plus élaborée.	
	Liste chainée	+ Ne contient que les cases occupées	- Il faut traverser la liste pour chaque modification, déplacement, prise, promotion etc.	
	Tableau 1 dimension		- Nécessite plus d'opération pour accéder un élément	
HISTORIQUE	Arbre	+ Permet d'avoir différentes branches sur les coups joués	- Pas intuitif a implémenter - Il existera toujours une limite au nombre de branches de jeu	Pour l'historique on a choisi une pile, parce que le principe de l'historique dans un jeu d'échec se rapproche de l'utilisation d'une pile, on va tout le temps ajouter les coups à la fin de l'historique.
	(double) Liste chainée	+ permet de facilement passer d'avant en arrière dans l'historique	- Ne permet de garder seulement 1 "chemin" - Peut rapidement mener à des erreurs, plus difficile à manipuler	
	pile	+ Structure de données très simple a manipuler.	- Ne permet de garder seulement 1 "chemin"	
PIÈCES PRISES	2 tableaux contenant respectivement les pièces prises noires et les pièces prises blanches	+ permet de distinguer les pièces blanches et noires	- Il faut de toute manière garder l'indice du dernier élément, c'est le comportement d'une pile pour l'ajout.	Pour les pièces prises, on a choisi une pile, pour la facilité des opérations, et comme on aura pas besoin de l'ordre des pièces capturés, l'utilisation d'une pile est une idée judicieuse.
	1 tableau avec toutes les pièces	+ permet de garder tout dans une seule grande liste.	- Il faut de toute manière garder l'indice du dernier élément, c'est le comportement d'une pile pour l'ajout.	
	pile	+ Comme l'ordre importe peu, on a juste besoin d'ajouter les pièces à la fin. + Accès au nombre de pièces capturés sans calcul.	- On ne peut pas accéder a n'importe quelle pièces prises.	
	Liste chainée	+ Permet d'avoir une taille variable + permet d'insérer facilement un nouvel élément, car on n'a pas besoin de l'ordre des pièces prises.	- Si on calcule les points des joueurs ou pour calculer le nombre de pièces capturées, il faut traverser toute la liste.	