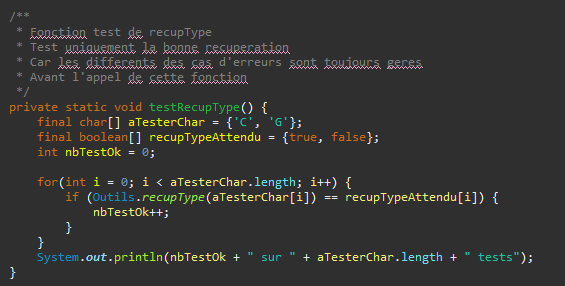
# Tests de outils :

## Test de recupType :

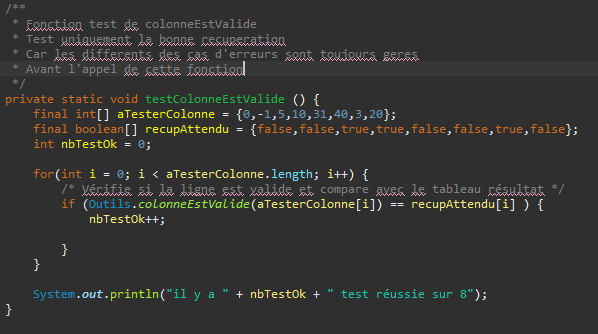


Ce test doit juste servir à vérifier que notre fonction est bien capable de récupérer le type du pion est que celle-ci retourne bien le bon résultat en fonction du type.

Ce test contient bien l’ensemble des possibilités requises à tester (C, G) et vérifie bien que la fonction retourne la bonne valeur attendue (Les différents cas d’erreurs étant teste avant)

Résultat :

## Test de colonneValide :

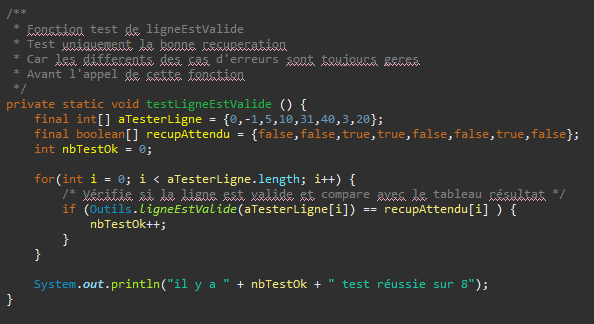


Ce test vérifie que notre fonction est capable d’identifier si la colonne rentrée par l’utilisateur est valide.

Pour cela on test un l’ensemble des cas possible dans cette fonction : chiffre négatif, Trop grand, trop petit et dans l’intervalle valide. Ce qui nous permet de tester tous les cas possibles et vérifier que le fonctionnement de notre fonction de vérification est Valide.

Résultat :

## Test de ligneValide :

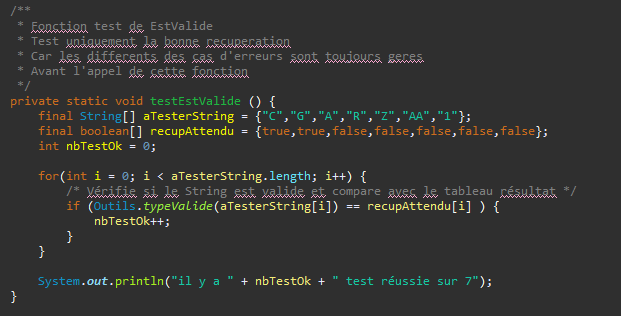


Ce test vérifie que notre fonction est capable d’identifier si la ligne rentrée par l’utilisateur est valide.

Pour cela on test un l’ensemble des cas possible dans cette fonction : chiffre négatif, Trop grand, trop petit et dans l’intervalle valide. Ce qui nous permet de tester tous les cas possibles et vérifier que le fonctionnement de notre fonction de vérification est Valide.

Résultat :

## Test de typeValide :

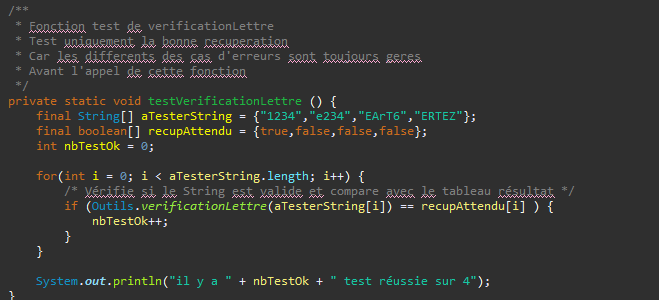


Ce test vérifie que notre fonction est capable d’identifier si l’utilisateur a rentre un type non valide de pion.

Pour cela on test tous les cas possibles : chiffre, lettre non valide (ne correspondant pas à un type), plus d’un caractère, et le cas ou le caractère est valide. On vérifie que le résultat attendu correspond à celui obtenu et si tous les tests correspondent cela signifie que notre fonction est capable de gérer tous les cas d’erreurs.

Résultat :

## Test verificationLettre :



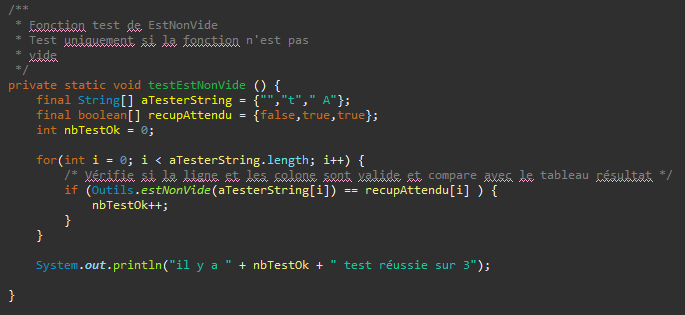
Ce test vérifie que notre fonction est bien capable d’identifier une chaine de caractère avec et sans lettre. Pour cela on test un ensemble de cas possible :

Pas de lettre, que des lettres, combinaison de lettre et de chiffre

Puis on vérifie que les résultats attendus lors des tests de ces chaines de caractères correspondent a celui attendu pour tester l’ensemble des cas possibles

Résultat :

## Test estNonVide :



Fonction qui vérifie si une chaine de caractère est vide ou non. Pour cela on test un ensemble de possibilité : Non vide, vide, test avec un premier caractère vide

Puis on test que tous nos tests renvoient le résultat attendu.

Résultat :

# Test visuel Outils :

## RecupereLignePion, RecupereColonnePion :

(Pas vraiment de tests ou de cas d’erreur a testé cette fonction celle-ci ne peut pas être appelé si l’un des cas d’erreurs pouvant la faire potentiellement crasher est présent)

## afficherConfigDispo :

Récupère l’ensemble des configurations enregistrées et les retourne sous forme de String :

(Utilisation dans l’application :)



## AttribuerNomJ1, attribuerNomJ2 :

Pas de réel test nécessaire car aucun cas d’erreur possible

Exemple attribution nom J1 : (Fonctionnement similaire pour J2 donc exemple de test pour les deux fonctions) :

### Personnalisé :





### Défaut :

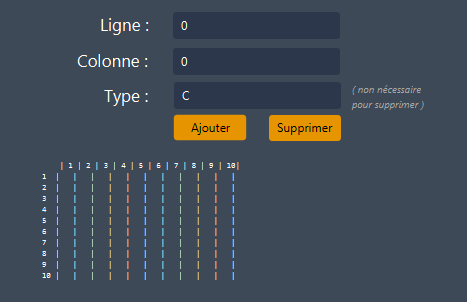




## cordOk :

Vérifie que le pion est bien dans le tableau :

### Essai trop petit :





### Essai trop grand :





## Test configExiste :

Vérifie que le numéro de la config entrée est valide

(Les cas d’erreurs comme les lettres etc. sont testés précédemment la fonction de peut pas rencontrer ces cas d’erreurs)



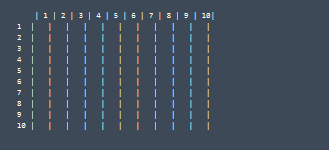


## Test de placementPion :

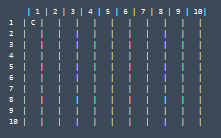
Place le pion entré par l’utilisateur dans le plateau en cours de création

(Les vérifications de validité sont effectuées précédemment)

Avant :



Après :



## Test sauvegarder :

Avant :



Après :



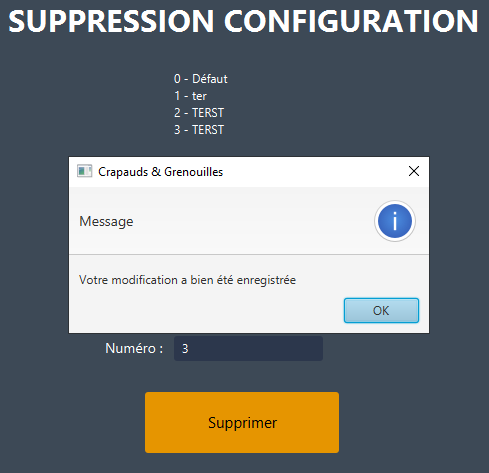
## Test supprimerConfig :

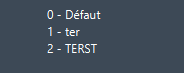
Vérifie que l’entrée de l’utilisateur est correcte puis supprime une config

Avant :



Après :





## Test suppressionPion :

Supprime un pion lors de la création du plateau si l’entrée de l’utilisateur est valide

Avant :



Après :



## VerifVictoire :

Aucun test réel fonction qui retourne un Boolean si la variable victoire indique qu’une équipe a gagné

## configIni :

Appel estNonVide, verificationLettre, coordValide

Pas de test réel appel une série de tests et affiche les messages d’erreurs / exécute la création du plateau si tout est correct (cf. voir les tests des fonctions appelées)

## placementPion :

Appel estNonVide, verificationLettre, cordOk,type Valide

Pas de test réel appel une série de tests et affiche les messages d’erreurs / exécute le code de placement du pion (cf. voir les tests des fonctions appelées)

## configValide :

Appel estNonVide, verificationLettre, configExiste

Pas de test réel appel une série de tests et affiche les messages d’erreurs / Retourne true si tout est valide (cf. voir les tests des fonctions appelées)

## actualisationPlateau :

Vérifie que les éléments entrés par l’utilisateur sont corrects et actualise le plateau de jeu

Appel estNonVide, verificationLettre, ligneEstValide, colonneEstValide

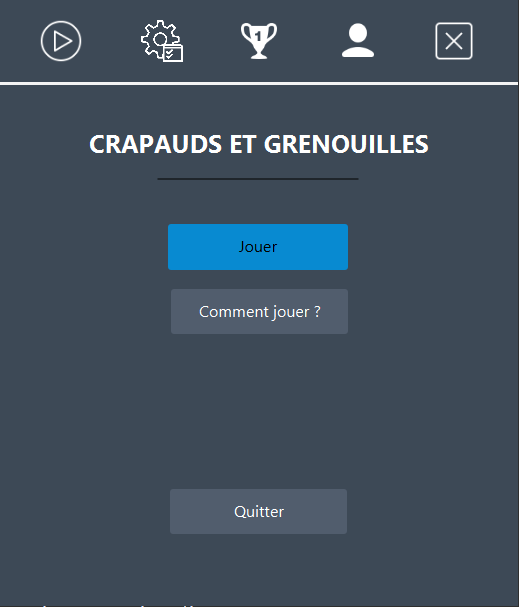
Pas de test réel appel une série de tests et affiche les messages d’erreurs / actualise le plateau de jeu après mouvement du pion (cf. voir les tests des fonctions appelées)

# Test InterfaceController :

## Test showMVP (Pas de réel test) :

Affiche la page MVP

Avant :



Après :



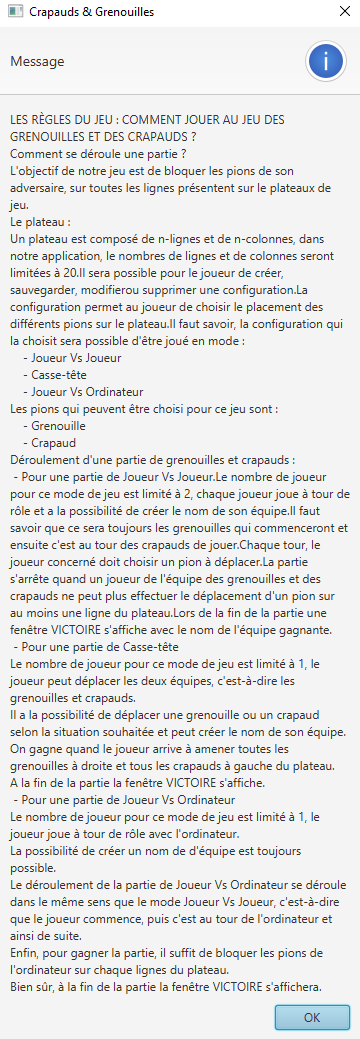
## Test showRegle (Pas de réel test) :

Affiche le texte d’aide

Avant :

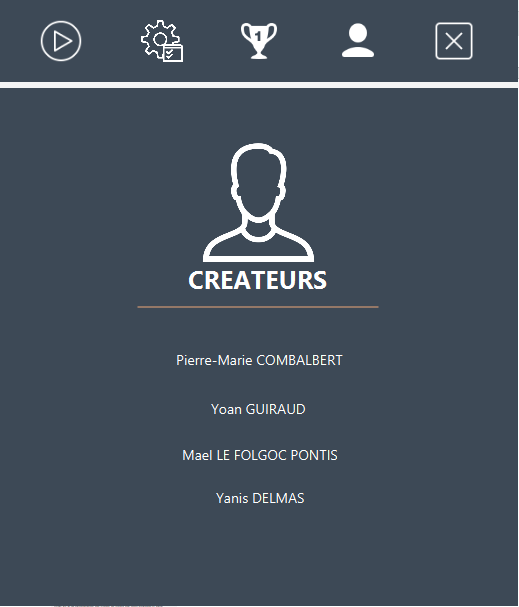


Après :



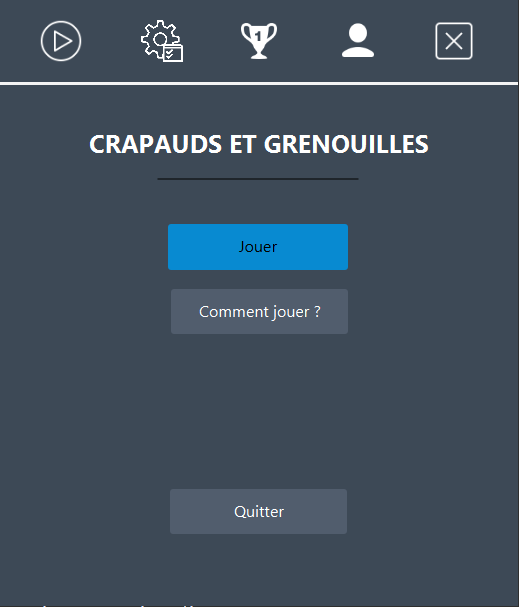
## Test showCreateur(Pas de réel test) :

Affiche page CREATEUR



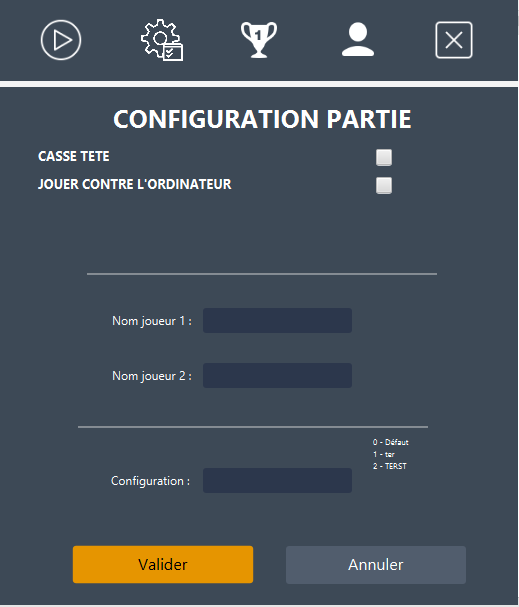
## Test showMenu(Pas de réel test)  :

Affiche menu principale :



## Test showConfigurationPartie (Pas de réel test) :

Affiche configuration de la partie :



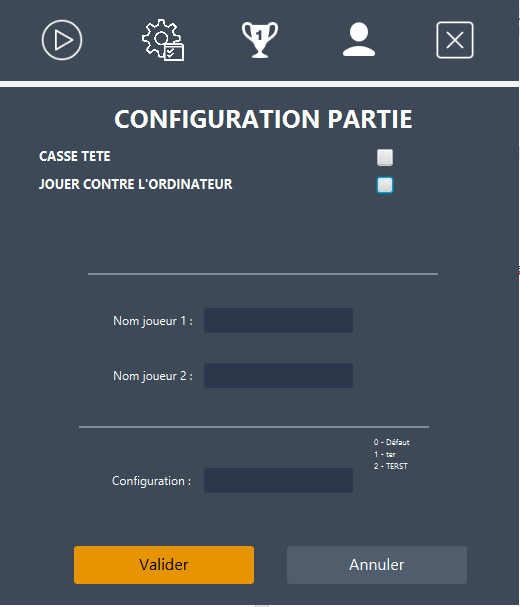
## Test showConfiguration (Pas de réel test) :

Affiche menue des configurations :

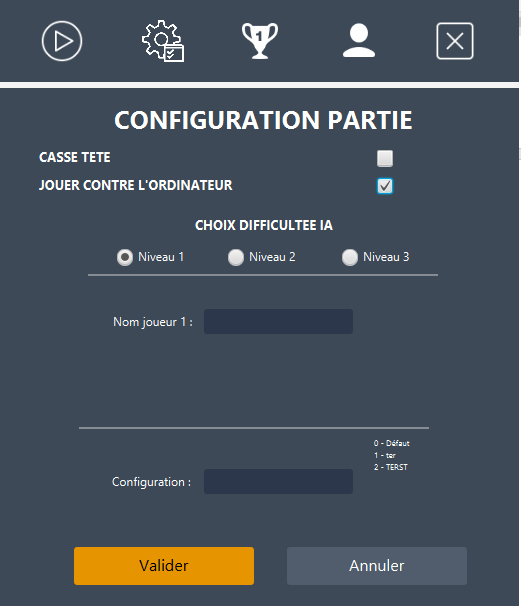


## Test verificationCheck, caseIaCheck, caseIaPasCheck, cacherJ2, cacherIA:

Case IA pas check :

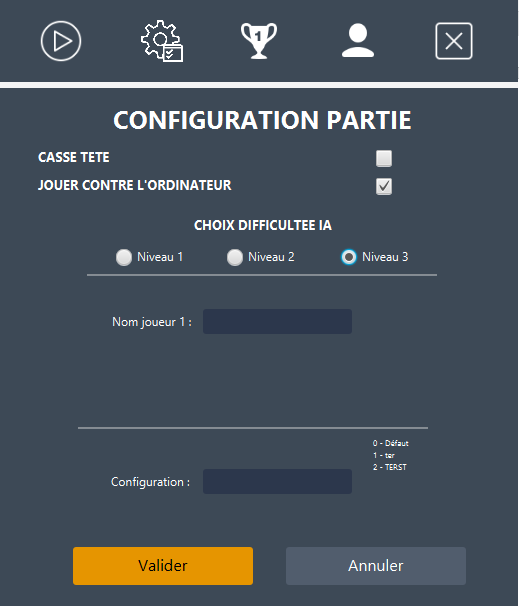


Case IA check :

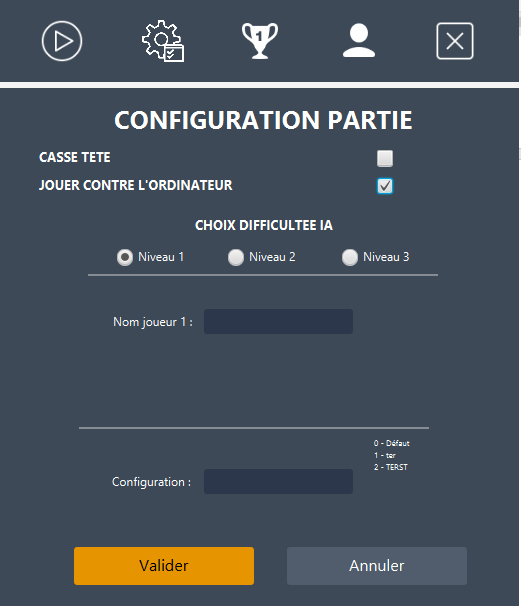


## Test razIA:

Avant :



On cache l’IA puis on la remet et celle-ci c’est bien re mise par défaut



## Test razPartie :

Avant :



On revient au menu puis retourne sur configuration Partie

Après :

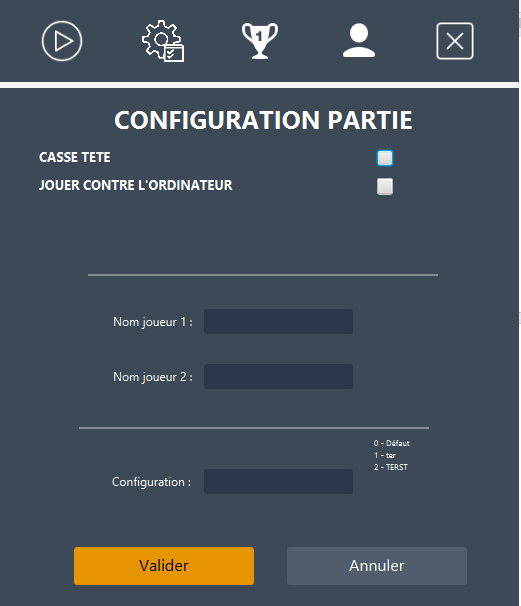


## Test verificationModeJeu, casseTeteConf, versusConf :

Mode casse-tête :

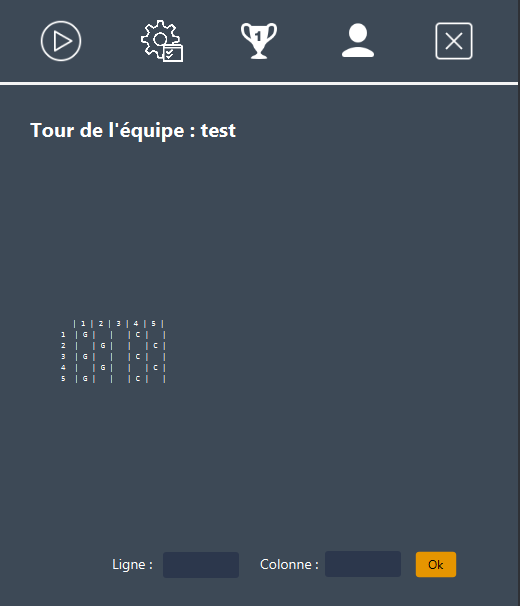


Mode versus :

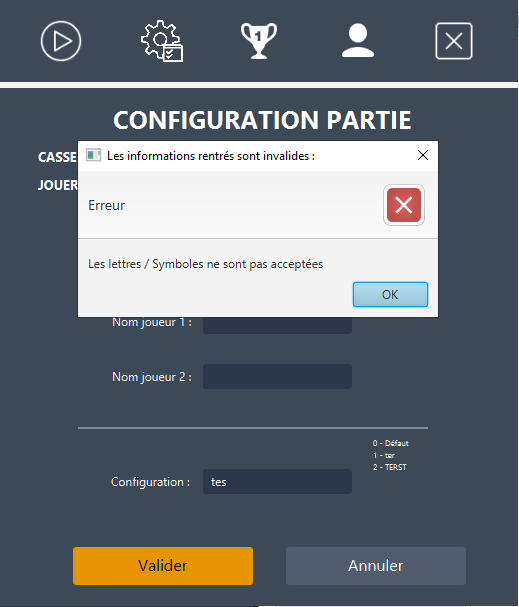
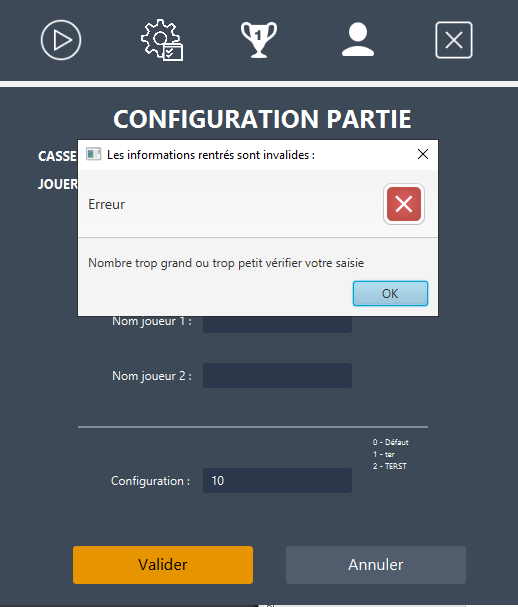


## Test showGameBoard, recupNomEquipe :

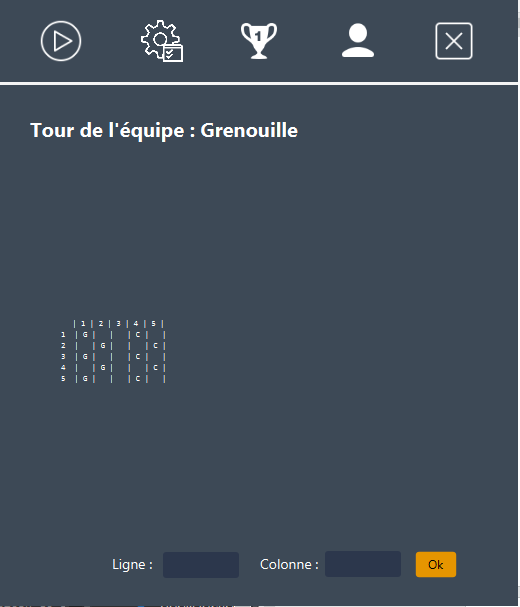
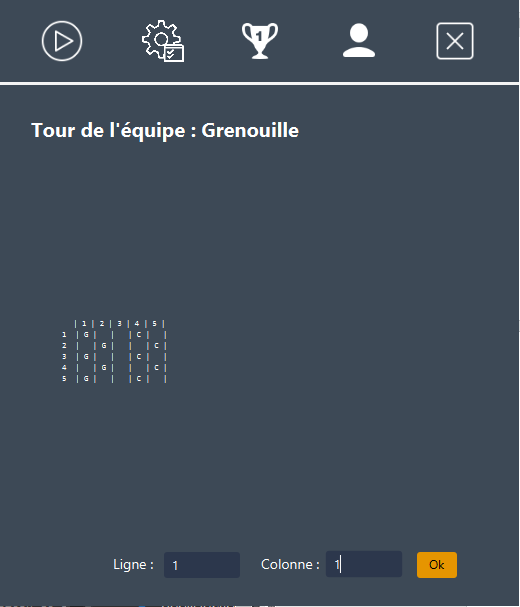
Cas entrée valide :

Cas non valide :

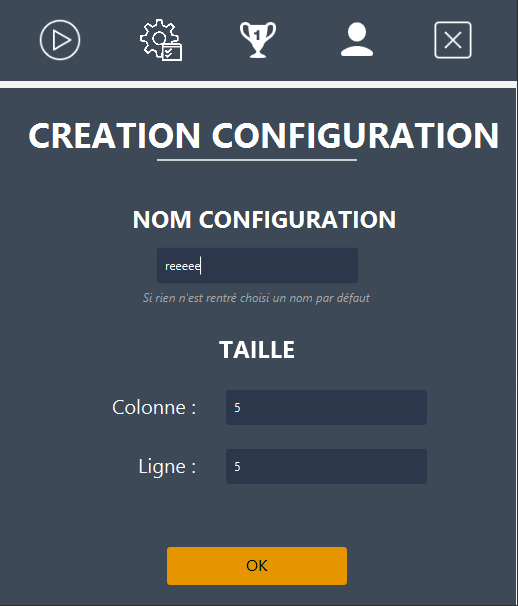
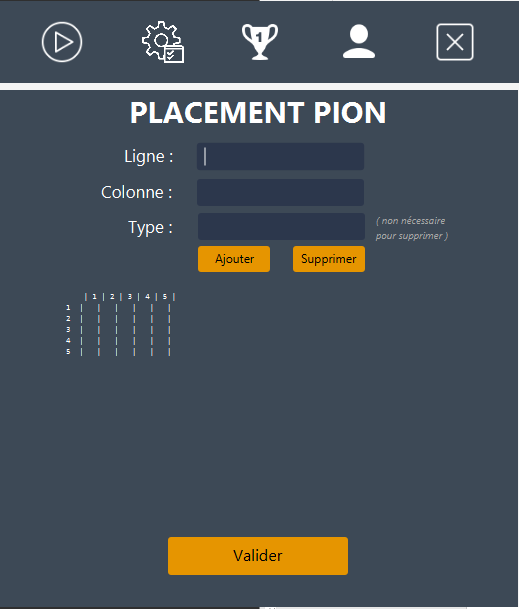
## Test actualiserJeu, rafraichirJeu :

Avant :  
 

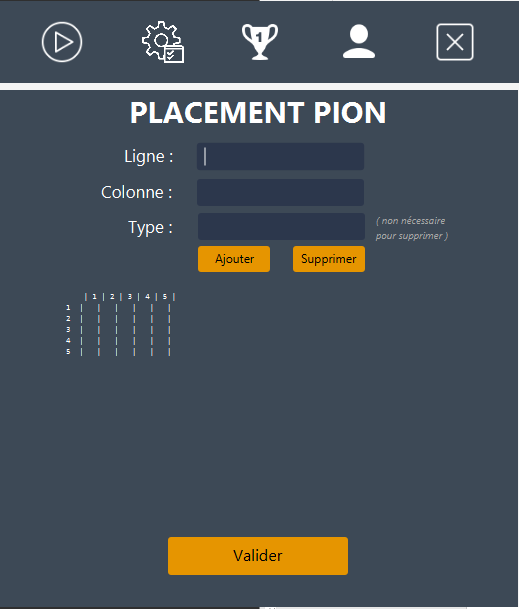
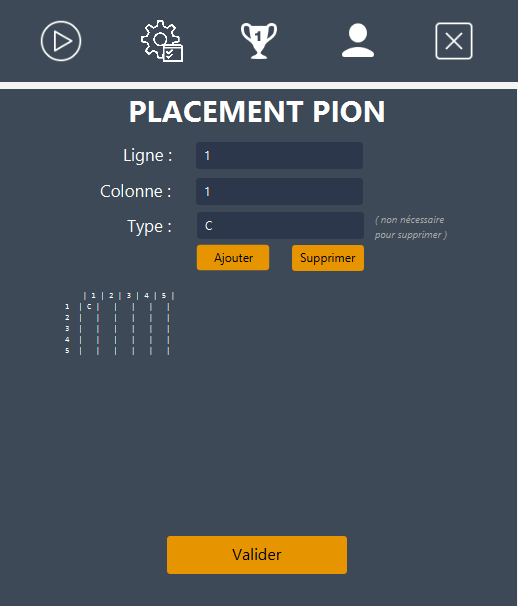
Après :



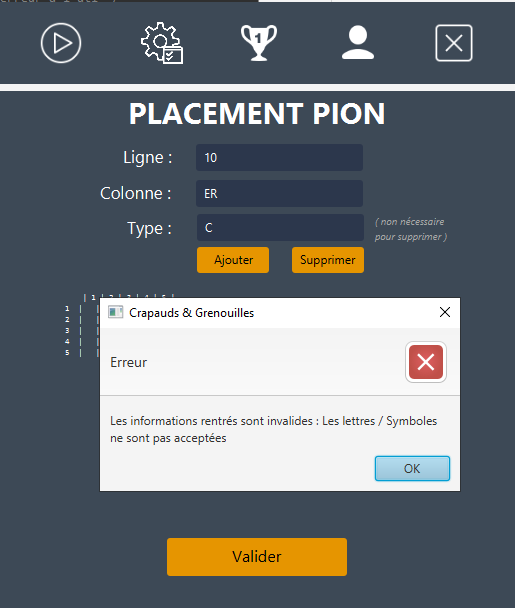
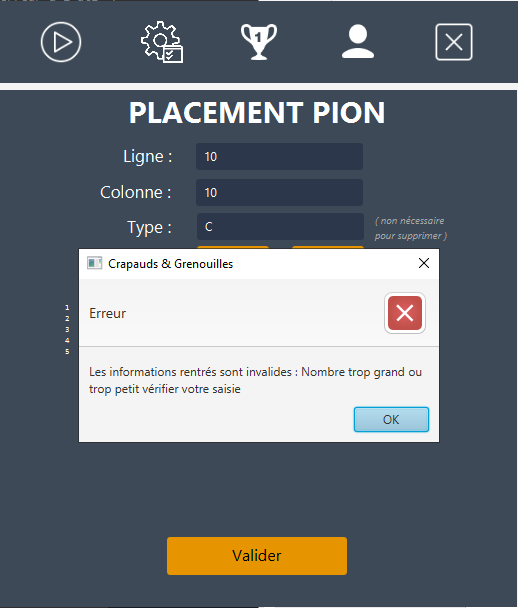
## Test configInitialisation :

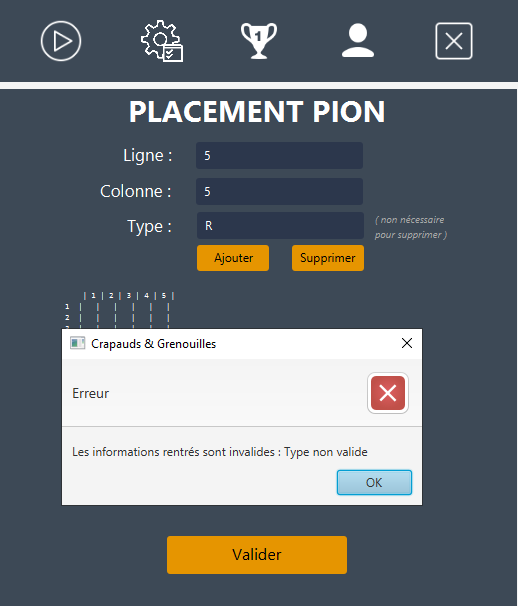
 

## Test ajoutPionConfig :

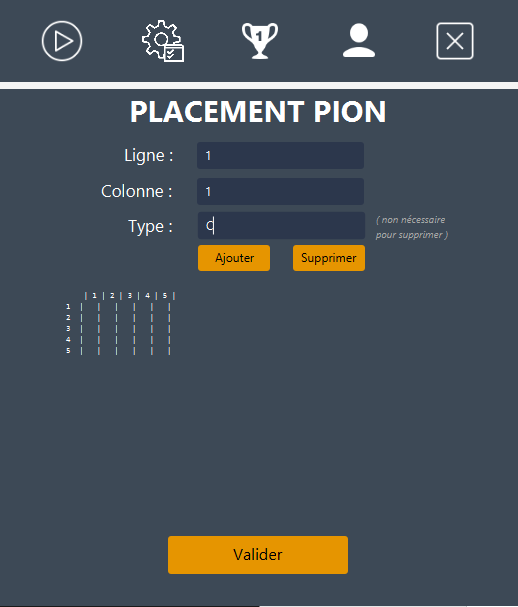
Cas d’erreur vérifier dans la création de configuration :

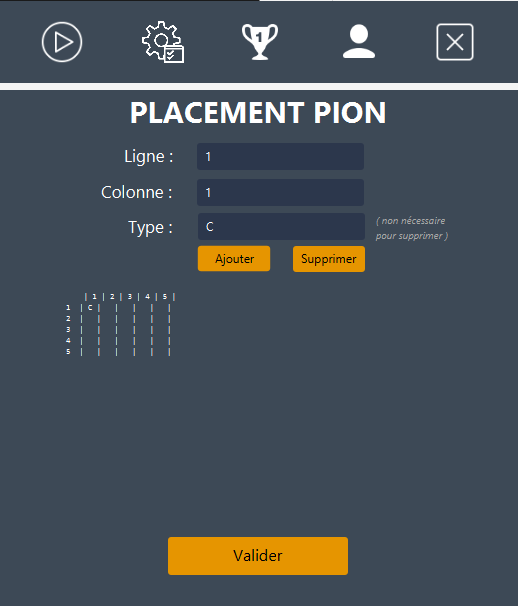
 



## Test rafraichirConf:

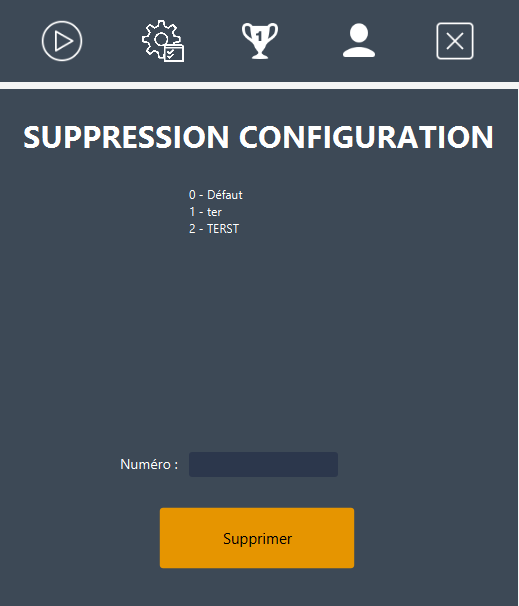
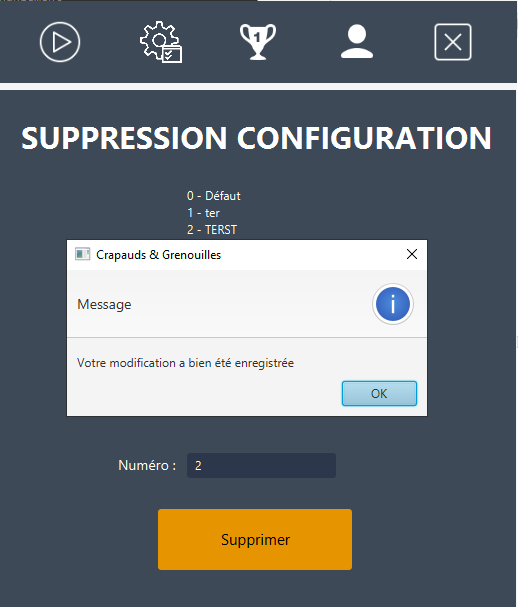
Avant:



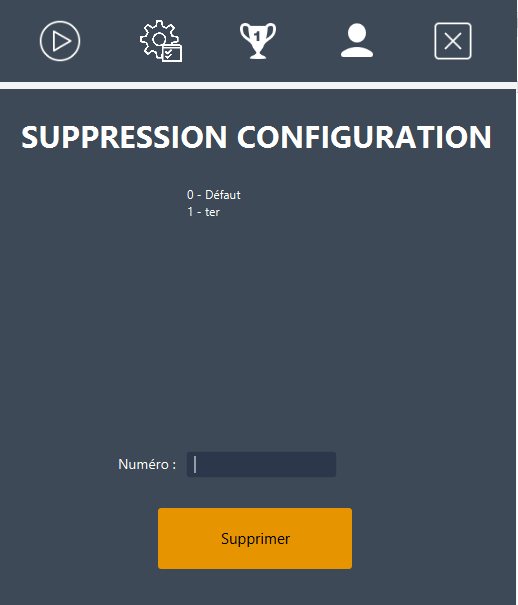


## Test deleteConfig:

Avant :

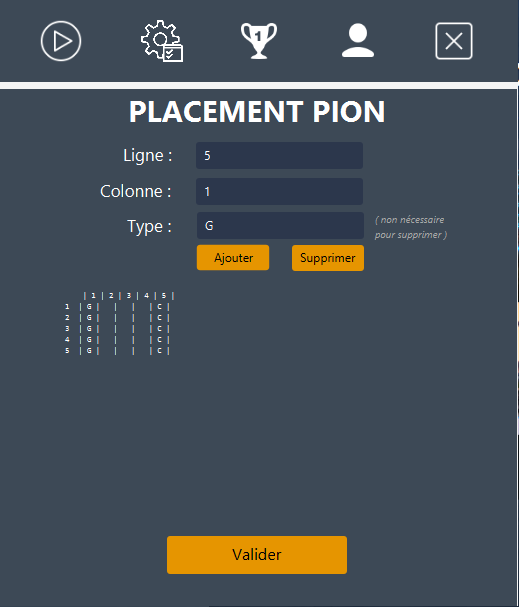
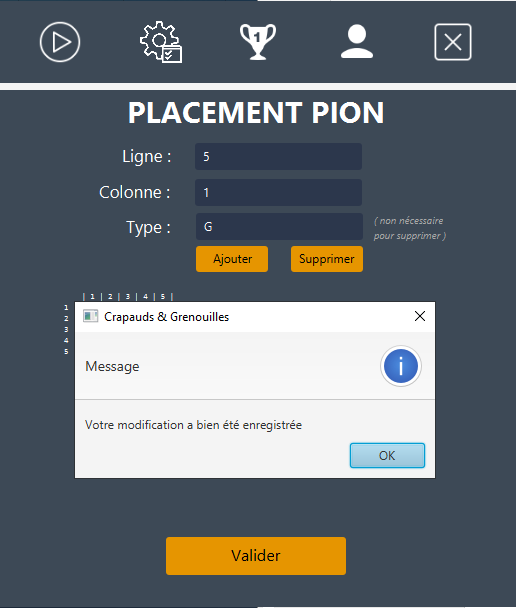
 

Après :

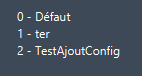


## Test enregistrerConfig :

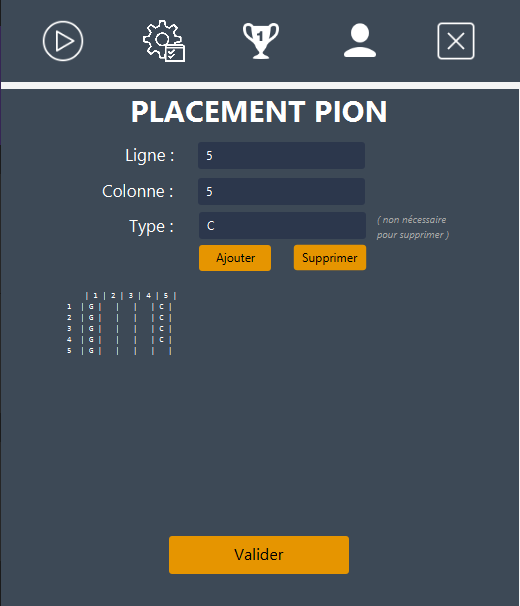
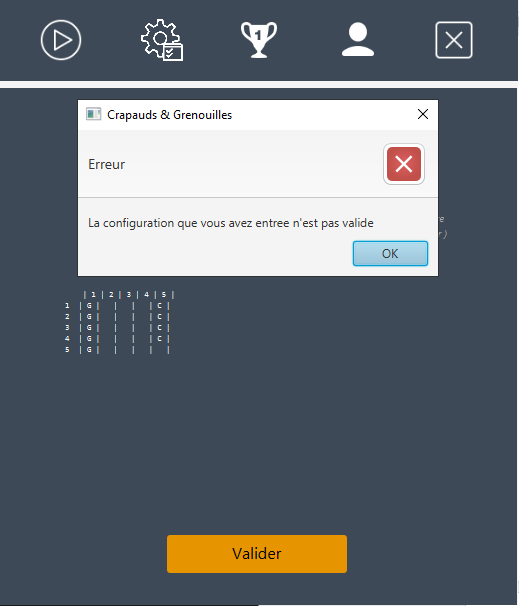
Config valide :

Après ajout :



Config non valide :

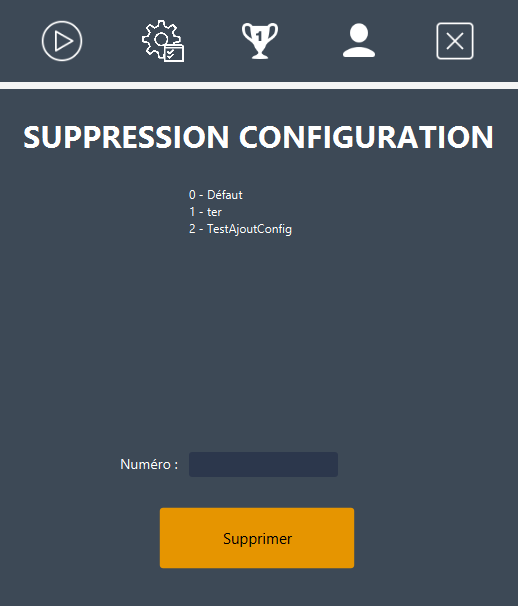
## Test showCreationConfig :

Affiche la page création configuration :



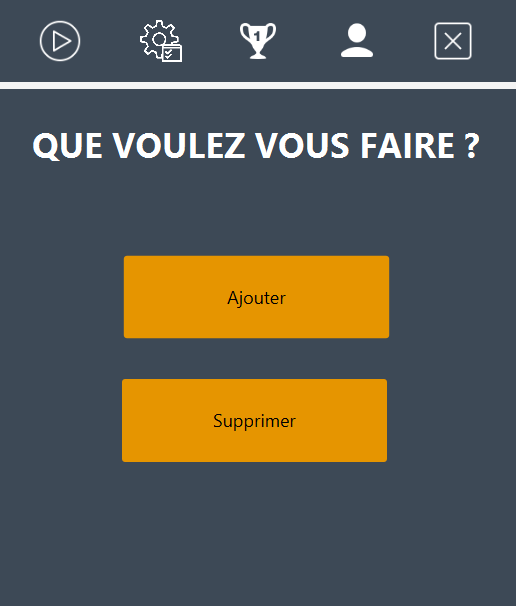
## Test showSupprimerConfig :

Affiche la page supprimer Config



## Test showInitialisationConfig :

Affiche la page de choix :



showInitialisationConfig

Fonctions sans réel test possible :

* Leave
* Reinitialiser
* recupModeJeu
* recupAdversaire
* recupNomConf