breakthrough 8x8 en mode bitboard

Lire `bkbb64.h`

nb_playout_per_sec pour savoir combien de playout il peut jouer par sec

Lire `nb_playout_per_sec.cpp`

Compilé avec `g++ -std=c++11 -Wall -O3`

- Apple M1 Max : 3.119.200.000 /sec
- AMD EPYC 7643 48-Core Base Clock 2.3GHz : 1.632.510.000 /sec

joueur aléatoire en C/C++

Lire `rand_player.cpp`

On donne au programme une board et un joueur et il répond un coup. Si `debug` est activé, il affiche la board.

```
$>./rand_player @@@@@@@@@@@@@@@@..............................00000000000000000 0
board 1
  A B C D E F G H
8 @ @ @ @ @ @ @ @ 8
7 @ @ @ @ @ @ @ @ 7
6 . . . . . . . . 6
5 . . . . . . . . 5
4 . . . . . . . . 4
3 . . . . . . . . 3
2 0 0 0 0 0 0 0 0 2
1 0 0 0 0 0 0 0 0 1
  A B C D E F G H
F2-E3
```

joueur Ludii (qui appelle le joueur C/C++)

Dans le répertoire `Ludii`

On a un `rand_player` C/C++ dans `bin`. Avec le `java`, on fait un `.class` placé dans `bk` pour faire un `.jar`. On exécute `Ludii` et on sélectionne le `.jar` créé pour jouer avec le `RandPlayerLocal`

Lire `RandPlayerLocal.java` (le joueur java qui appelle le joueur C/C++)

Lire `makeJar.sh` (prg à exécuter pour faire un nouveau `.jar`)

Lire `runLudii.sh` (prg à exécuter pour lancer `Ludii` après avoir déplacé le `jar` et l'exécutable `rand_player` au bon endroit)

Penser à télécharger la dernière version (1.3.14) de Ludii ICI (https://ludii.games/download.php)

Pour faciliter l'exécution, renommer votre exécutable `rand_player` avec les initiales des membres du groupe. On lance les exécutions à partir d'un répertoire `LudiiPlayers` qui contient les fichiers `jar` et un répertoire `bin` avec les exécutables.

Pour un groupe avec Astrid B. et Christian D., on a les initiales AbCd. Le programme `rand_player.cpp` devient `ab-cd.cpp` et son binaire devient `ab-cd`. `RandPlayerLocal.java` devient `AbCd.java` et `RandPlayerLocal.jar` devient `AbCd.jar`. Revoir les fichiers en concordance avec les initiales des personnes de votre groupe.