game.cpp

```
//-----------------------------------------------------------
//
// Fichier        : game.cpp
// Auteur(s)      : (C) Mariaux Ewan & Nicolas Sonnard
// Date           : 2020-12-2022
// But            :
//
// Modifications  :
// Remarque(s)    : Le soft a pas été terminé il se compile pas
//                  et il manque du code pour qu'il fonctionne
//
//-----------------------------------------------------------

/**
 *
 *  nous garderons la classe robot. La classe terrain sera dégagée au profit de game qui regroupera l'ensemble
 *  des instructions pour faire avancer le robot.
 *
 */
#include "game.h"
#include "robot.h"
#include "annex.h"
#include "iostream"
#include <algorithm>
#include <numeric>
#include <random>

using namespace std;



void positionnement(Game game, int& posX, int& posY){

    //positionnement aléatoire du robot dans le terrain
    do{
        posX = nbrAleatoire(1, game.largeur - 1);
        posY = nbrAleatoire(1, game.hauteur - 1);
        //vérifie que l'emplacement est libre
    }while();

    find_if(game.vRobots.begin(), game.vRobots.end(), Robot(posX, posY));
    count_if(game.vRobots.begin(),game.vRobots.end(),Robot(posX, posY));

}

void creerRobot(Game game, int nbreRobot){

    int posX,
        posY;

    for(size_t i = 0; i < (size_t)nbreRobot; ++i){

        positionnement(game, posX, posY);

        game.vRobots.push_back(Robot((int)i,posX, posY));
        cout << game.vRobots[i];

    }

}


Game::Game(int nbreRobots, int largeur, int hauteur){

    this->vRobots.reserve(nbreRobots);

    this->largeur = largeur;
    this->hauteur = hauteur;

    creerRobot(*this, nbreRobots);

}

/**
```

```
 *
 *  ici on va caller le différentes instructions de jeu.
 *  il sera appelé dans le main tant qu'il il y a plus d'un robot il retourn un vrai
 */
bool Game::tourJeu(){

}

void Game::afficheTerrain() const{
    // Affiche haut du terrain
    cout << string((size_t) this->largeur + 2, '-') << endl;

    // Affichage les lignes du terrain
    for (size_t y = 0; y < (size_t) hauteur; ++y) {
        cout << "|";
        for (size_t x = 0; x < (size_t) largeur; ++x) {
            /**
             *  TO DO
             *  foncteur pour afficher si il y a quelque chose
             */


        }
        cout << "|" << endl;
    }
    // Affiche bas du terrain
    cout << string((size_t) this->largeur + 2, '-') << endl;
}




int Game::getLargeur() const {return this->largeur;}

/**
 *  retourne la hauteur du terrain
 * @return hauteur du terrain
 */
int Game::getHauteur() const {return this->hauteur;}

/**
 * gere la partie
 * @param game
 */

void jouer(Game game){
        // id du robot à supprimer
        int id;
        do{
            //tri aléatoire des robots dans le vecteur
            shuffle(game.vRobots.begin(), game.vRobots.end(),default_random_engine());

            for(size_t i = 0; i < game.vRobots.size(); ++i){
                //déplace le robot
                game.vRobots[i].deplacer();

                //si l'emplacement n'est pas libre on détruit l'ancien robot
                if(!find_if(vRobots.begin(), vRobots.end(),vRobots.id()))
                {
                    // TO DO
                    //le robot qui est déplacé n'est pas encore affiché
                    //c'est donc le robot qui était avant à cet emplacement qui est retourné

                    //détruire le robot à la nouvelle position de vRobots[i]

                }

                //affichage du robot à son nouvel emplacement

            }
            Game::afficheTerrain();
            //on s'arrête dès qu'il reste 1 robot
        } while (game.vRobots.size() != 1);
}
```