#ifndef BOARD\_HPP

#define BOARD\_HPP

#include "critter.hpp"

#include <random>

#include <iostream>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* RANDOM STARTING POINT FUNCTION I'm pretty partial to using this

\* kind of setup lately.

\* This will rely on numberOfColumns and numberOfRows ints in Board class.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//setting up the Mersenne Twister

static std::random\_device randyEngine;

static std::mt19937 theTwister(randyEngine());

class Board {

private:

bool customBoard = false;

Critter\*\*\* gameBoardPtr;

int numberOfRows;

int numberOfColumns;

public:

bool isCustom();

void initBoard();

void printBoard();

void breedCritters();

int getXLimit();

int getYLimit();

void makeMove();

//I made two because I wasn't confident that returning an array would work so well.

//One function handles the X dimension of the board, another handles the Y

//The functions know if the board is custom and will adapt to that situation.

//If the board is "standard," (20x20), it will behave appropriately.

void randomStartingSpawns(int, int);

int randomSpawnPointX();

int randomSpawnPointY();

~Board();

};

#endif