Introduction

This is a documentation of a program that simulates a habitat consisted of simple bug-like organisms that consist of preys and predators. In this simulation; time is calculated within steps. At each step every existing critter moves according to their specified instructions. They all gather their areal information from the main class 'World'. World class is responsible for keeping the track of the bug vector that contain the list of alive bugs. The function is also responsible for bug movement activation, bug generation and bug removal. Short terms, the World class contains bunch of helpful subfunctions and data classes that ease the step automation process and visualization.

Class: Organism

Organism is the framework class for bugs living in the world. Inheritance from this class gives most of the basic functionalities such as location, breeding step size and movement methods.

gmove (ghost move): Moves according to given char input

move (normal move): Moves randomly according to given areal check integers(up down left right), can be overloaded

type: Integer values of animal types. 0 is the default value that no animal uses. 1 is for ants, 2 is for poisonous ants and 3 is for critters. Best initialization practice to set this value is in construction function for every inherited class. But it can also be changed with *'setType()'* function.

Class: Doodlebug

Doodlebug class is for predator organisms that feed and survive on other ant creatures.

move (normal move): This overloaded function also make moves randomly according to given areal check integers(up down left right). But in addition, it picks spaces where ants exist. Priority is up, down, left, right. After movement decision been made, it decreases or increases the life integer. If life already hit 0, does nothing.

Class: World

World class operates creatures movement, detection, breeding and death functions on a defined 2d grid. Generating desired amount of population in random places and making graphics is being simplified with build up functions.

Creatures are held in vector pointers. Creatures dynamically generated and then assigned to this pointer vector list. At the end of the task, they should be deleted with the '*pesticide()'* function.

Random generation functions: *'addAnt(bool)'* and *'addDoodles()'* are two random generation functions. They pushback a new organism pointer and then assign it to the new creature memory at the end of the vector. They are mostly useful for generating new simulations. addAnt function creates poisonous ant if given true.

Determined generation functions: *'addAnt(bool,x,y)'* and *'addDoodles(x,y)'* are two generation functions that put organisms in set positions. If given space is available, they pushback a new organism pointer and then assign it to the new creature memory at the end of the vector. They are mostly useful for simulation loop. addAnt function creates poisonous ant if given true.

Died crit: *'diedCrit(int)'* Frees specified creature memory from vector pointer list and removes the pointer. This changes the list length.

Check square: *'checkSq(x,y)'* Gives information integer about given coordinate. If it is: out of bounds:-1, empty space:0 else it is creatures type number.

Get bug row: *'getBugRow(x,y)'* Searches for bug vector position at given coordinate. If it is empty or invalid returns -1.

Move all: *'moveAll()'* Is the master function that commits one step for every creature. Move priority is given to predators with *'moveDoodles()'* to allow them to catch their preys if they can. After that, other creatures are moved. Breeding status taken cared of the last with the *'breedCrits()'* function.

Move doodles: *'moveDoodles()'* Is the function that moves every predator for one step. If they can eat, function removes target ant from the vector list. If their life integer is zero(can be checked with starve()), they are moved from the list. Predator areal checkings are put in temporary integers since it can change if any bug removed from the vector.

Breed crits: *'breedCrits()'* Is the function that first manipulates its breed count, then checks if the bug became eligible for breeding (when getBreedC() returns 0). If it is eligible, it picks a random neighbor space then checks it is available. If it is, it adds new creature to vector.

Print graph: *'printGraph()'* Is the for printing graphics of worlds current state to the terminal. Ants are represented with ○, poisonous are represented with ©, doodles are represented with ╳.

Pesticide: *'pesticide()'* Is the purge function that deletes every dynamically pointed memory and vector pointers. It is also called for world destructor.