Course Project Documentation
CS101 Project 2014-15

Zombie Apocalypse Simulator

Group : 170
Sattwik Deb Mishra, 140070036
Sanat Anand, 140040005
Shruti Hiray, 14D070016
Tejeshwar Thorawade, 140070015

Table Of Contents

1. INTRODUCTION :

The Ant Colony Optimization (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs.

ACO metaheuristic finds application in a variety of fields such as:

- Job scheduling problems
- Vehicle routing problem
- Device sizing problem in nanoelectronics physical design
- Image processing
- Protein folding

Zombie Apocalypse Simulator is an attempt to illustrate a lighter application of ACO and is mainly intended to be educational.

Zombies and the implications of a zombie apocalypse have been a popular topic amongst many people in the past several years and nowadays even have mainstream films, books, TV shows and games dedicated or involving them. The project aims to provide a simulation environment in which users can simulate the fate of humans (residing in various colonies) trying to survive in a zombie infested world.

Compilation and run Instructions (on Mac OS X only)

1. Open terminal at source folder from the downloaded .tar file
2. Run the command “g++ main.cpp all_constructors.cpp all.cpp zombie_nest_input_modified.cpp human_colony_input_modified.cpp help_window_2.cpp start_window.cpp food_source_input.cpp -framework SDL2” to compile.
3. Run “./a.out” to run the program.

2. PROBLEM STATEMENT:

The project's primary aim is to simulate the outcome of a 'world' infected by zombies and inhabited by humans.

Humans live in colonies and Zombies live in nests. Humans search, collect and return to colonies to store food. Zombies survive on humans and are thus on constant lookout for humans for their survival.

The input for the simulation is taken through a graphical user interface, which is the primary functional area . The output consists of the time evolved state of the system and should change continuously in real time.

3. REQUIREMENTS:

HARDWARE INTERFACE:

The hardware interface requirements for this program are:

- A pointing and clicking device.
- A keyboard
- A visual display unit

SOFTWARE INTERFACE:

The software interface requirements for this program are:

- The software must be used on a machine capable of running C++ 11 programs.
- The user should have SDL 2.0 library installed on his system.
- The software currently only works on Mac OS X for unknown reasons.

4. IMPLEMENTATION:

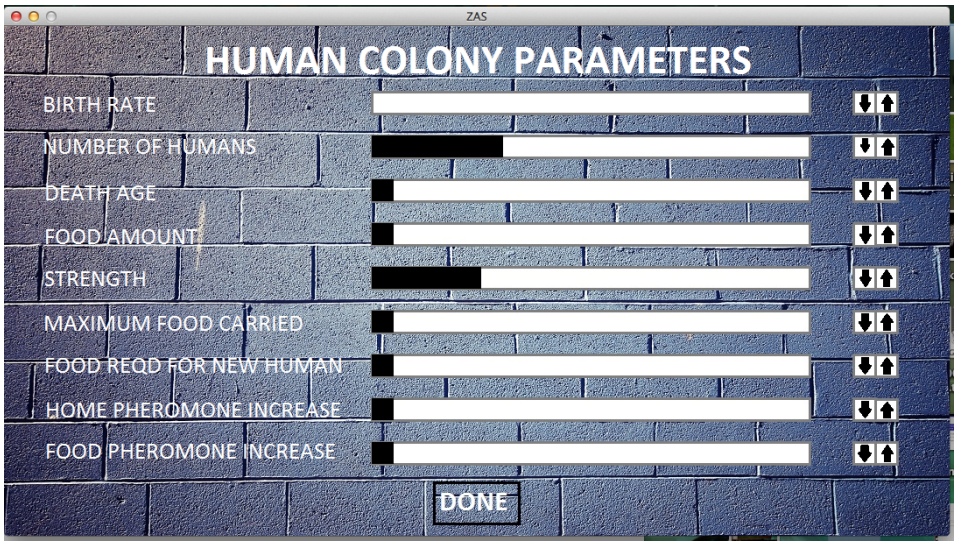
A)FUNCTIONALITY:

1. User Input:

User needs to provide the following inputs prior to start of simulation
a. Location of colony for humans (Coordinates on GUI – based on mouse click)

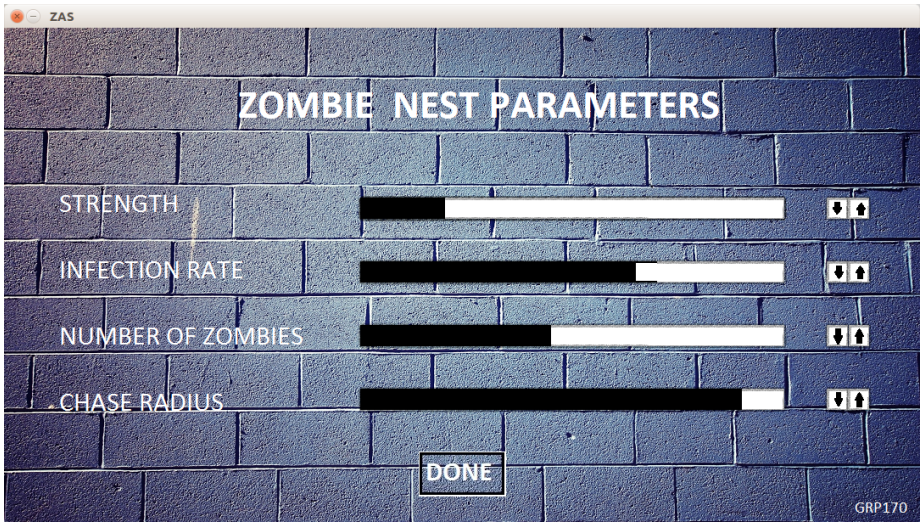
b. Location of nest for Zombie (Coordinates on GUI – based on mouse click)
c. Location of food source and their amount (Coordinates on GUI – based on mouse click)
d. For specifying multiple colonies for humans, nests for zombies and food sources, user must click on the GUI the required number of times
e. Parameters associated to different colonies to be specified are:

i. human birth rate
ii. initial number of humans
iii. death age of humans
iv.initial food amount in colony
v. human strength
vi. maximum food carried by human
vii. food required for new human
viii. pheromone increase.



f. Parameters associated to different nests to be specified are:

i. zombie strength
ii. zombie infection rate
iii. initial number of zombies
iv. zombie chase radius



g. Parameters associated to food sources to be specified are:

i.area of food source
ii. amount of food per node.

2. Output:

a. Time-evolved state of the system that changes continuously in real time

3. Processing Logic:

The program will display the results of the simulation according to the following logic:

a. Humans search for food following ACO algorithms (suitably modified)
b. Humans find and carry food back to the colony.
c. Humans can carry a limited amount of food (up to its maximum carrying limit)
d. Zombies walk about randomly and give chase to the closest human entering into its chase radius.
e. If a zombie catches up with a human, they fight and the outcome is decided in a probabilistic manner, taking into account the strength of zombie and human.
f. The outcome of a fight for a human and zombie are:

i. human lives and zombie dies
ii. human gets infected (becomes a zombie) and the zombie from the fight dies
iii. human dies and zombie lives
iv. human gets infected (becomes a zombie) and the zombie from the fight lives

g. The following outcomes from a fight are not allowed:
    i. Both the human and the zombie live
    ii. Both the human and the zombie die
h. The simulation ends when either all humans die or all zombies die or user decides to stop it through GUI.

Details about functionalities

HUMAN COLONY INVENTORY STATUS:

1. Food stored (community reserve) by each human colony.
2. Community reserve depletes by a fixed amount(if sufficient) when a new human is born at a colony after a fixed amount of simulation time passes.
3. Community reserve increases when a member of the colony successfully brings back food.

HUMAN ACTIVITY:

1. Food gathering by following trails left by them and other members of their respective colonies.
2. Each human can carry only limited amount of food up to a maximum amount.
3. Fighting in case zombie attacks.
4. Carrying food back to nest.

ZOMBIE ACTIVITY:

1. Random Searching
2. Chasing closest human when it enters chase radius
3. Fighting humans if they catch up

HUMAN-ZOMBIE INTERACTION:

1. Fight when at same position

2. Outcome:
1. Human lives, Zombie dies
2. Human infected, Zombie lives
3. Human infected, Zombie dies
4. Human dies, Zombie lives

FOOD SOURCE STATUS:

1. The amount of food in the source node decreases when a human carries food from it.

ALGORITHM

1. Humans start out from respective colonies in a random direction, with zero food and maximum food and home pheromones.
2. Human follow food pheromone when not carrying food and follow home pheromone while carrying food.
3. Humans can pick up food from grid points having food.
4. The humans follow a particular pheromone trail by deciding which direction(among 8) contains the highest amount of pheromone. If all the directions have the same amount, then the direction is (0,0) and hence, the movement either continues along previous direction or gets affected by random factors as described below.
5. The humans get bored from time to time based on a random number generated. If the random number (between 0 and 1) generated is greater than the chance of getting bored, they stop following a trail and simply continue moving in the direction in which they were moving before for a certain amount of time only.
6. The humans can also wander off a trail if a random number between 0 and 1 exceeds wander chance. Wandering means a random direction will be added to the human's proper direction for that particular step.
7. Zombies move about randomly.

8. Whenever a human and a zombie encounter each other inside a zombie's chase radius, they fight and the outcome is decided based on probability. The zombie wins if probability favours its strength and vice versa.

B) Graphical User Interface:

The images used in simulation are:

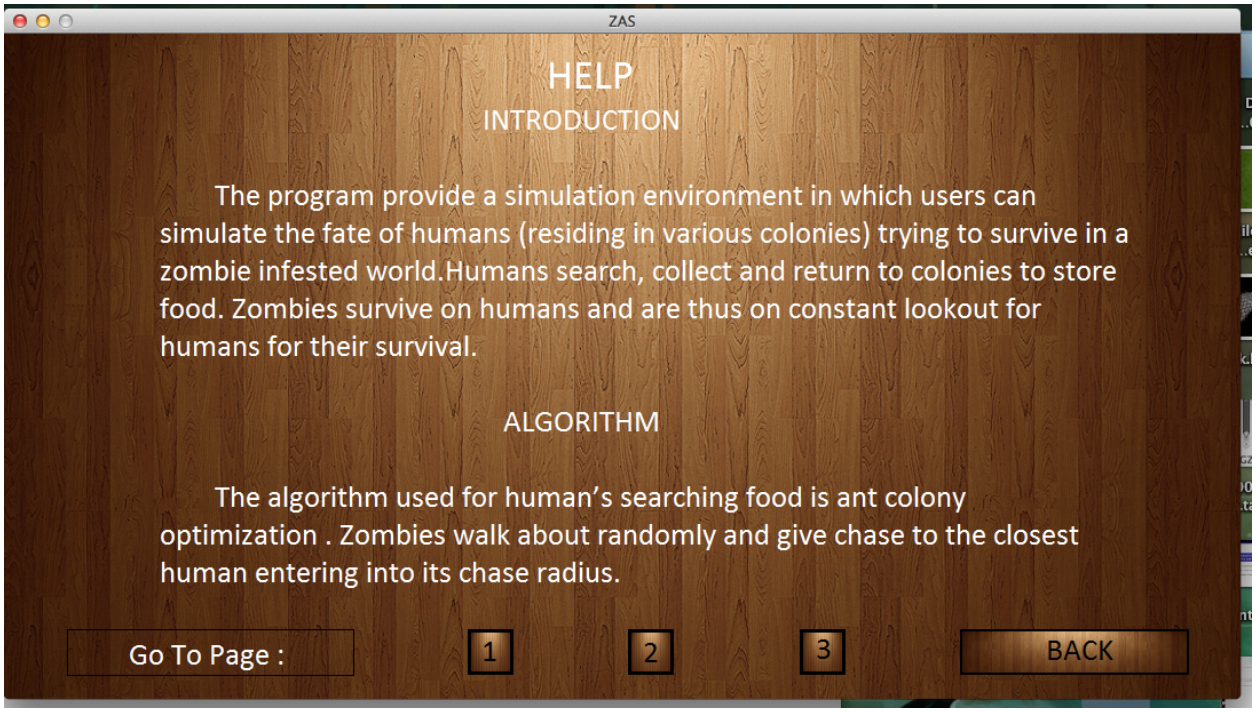
Human colony : 

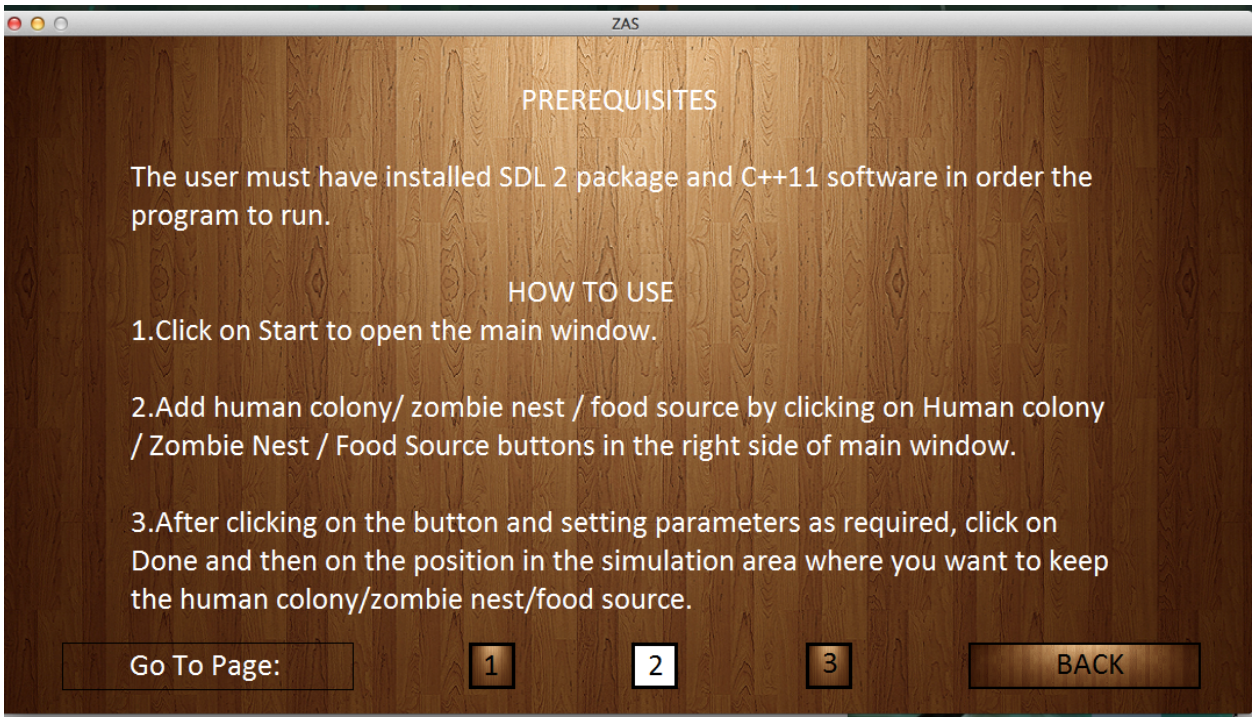
Zombie nest: 

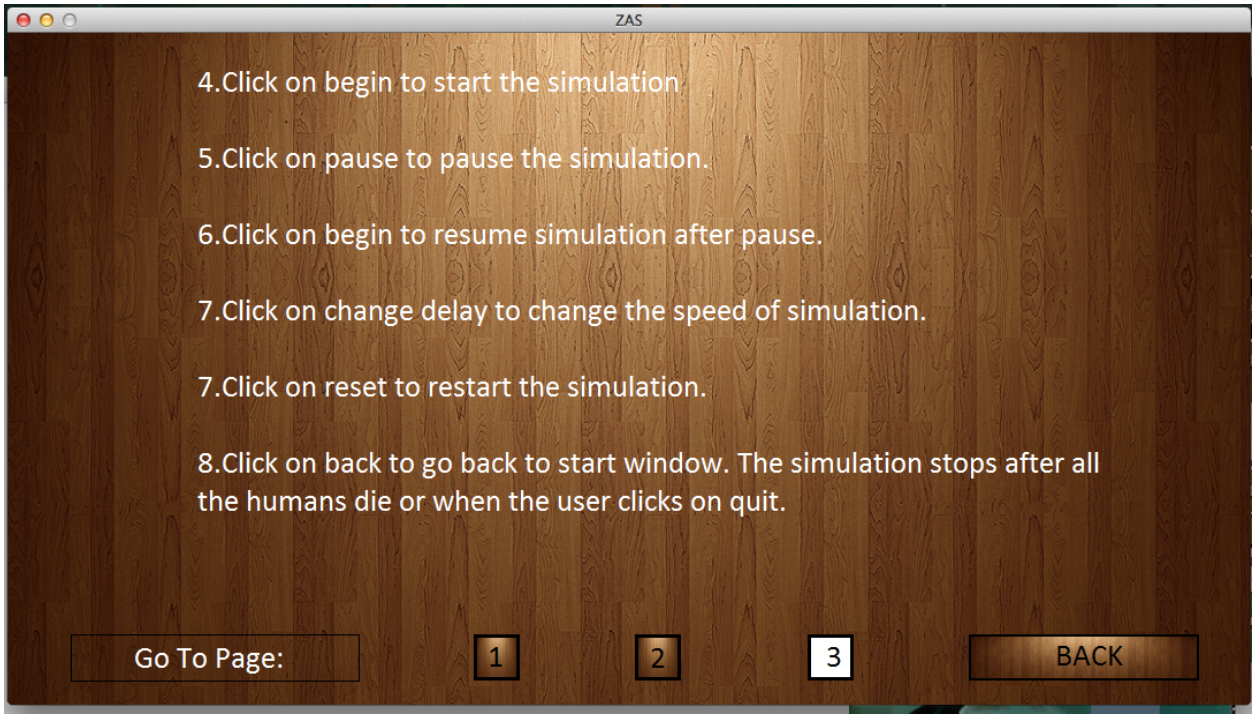
Food source: 

At the start of the program is the start window as shown below. It has two buttons namely Start and Help.

Help button opens up the window which contains instruction for the user on how to operate the program. It contains 3 pages and user can go on any page as he/she clicks on 1 , 2 , 3. To go back to start window back button is clicked.


ZAS
HELP
INTRODUCTION
The program provide a simulation environment in which users can simulate the fate of humans (residing in various colonies) trying to survive in a zombie infested world.Humans search, collect and return to colonies to store food. Zombies survive on humans and are thus on constant lookout for humans for their survival.
ALGORITHM
The algorithm used for human's searching food is ant colony optimization . Zombies walk about randomly and give chase to the closest human entering into its chase radius.
Go To Page :
1
2
3
BACK


ZAS
PREREQUISITES
The user must have installed SDL 2 package and C++11 software in order the program to run.
HOW TO USE
1.Click on Start to open the main window.
2.Add human colony/ zombie nest / food source by clicking on Human colony / Zombie Nest / Food Source buttons in the right side of main window.
3.After clicking on the button and setting parameters as required, click on Done and then on the position in the simulation area where you want to keep the human colony/zombie nest/food source.
Go To Page:
1
2
3
BACK



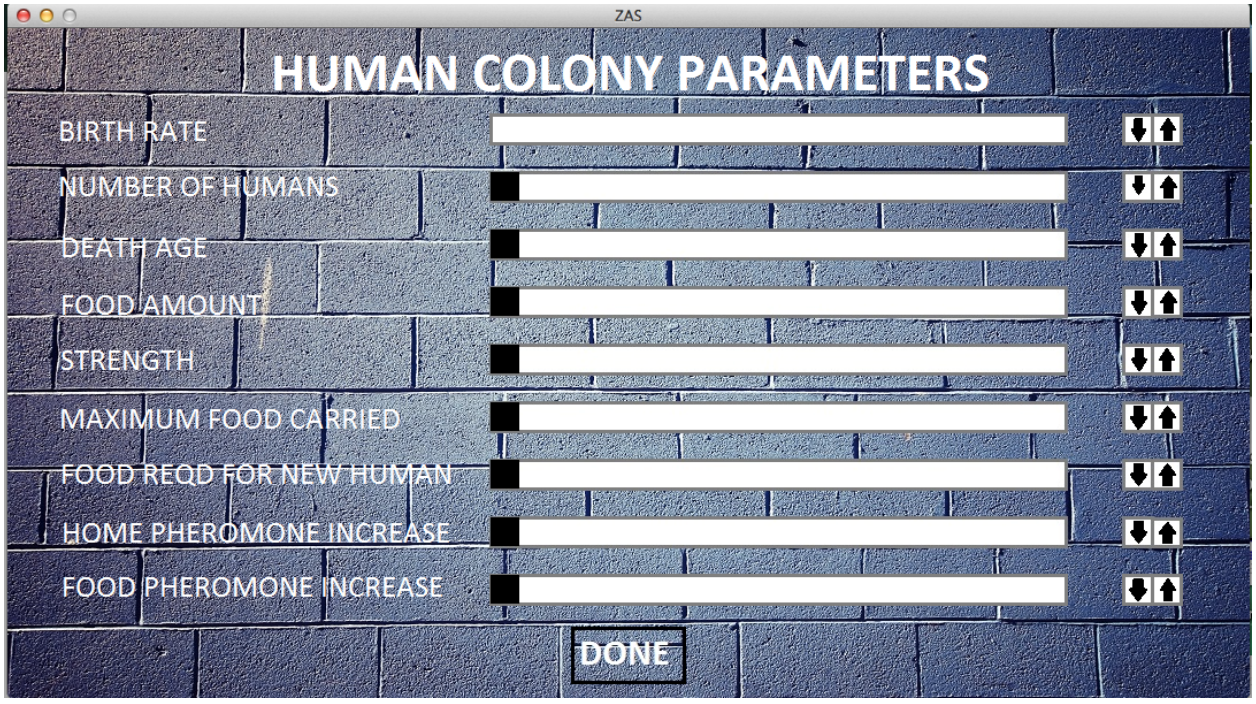
To start the simulation START button is pressed

Main window is shown below. The green area is the part where simulation will happen. User can add new human colony , zombie nest, food source by clicking on the right side buttons. Back button is to exit the program. Begin button begins the simulation

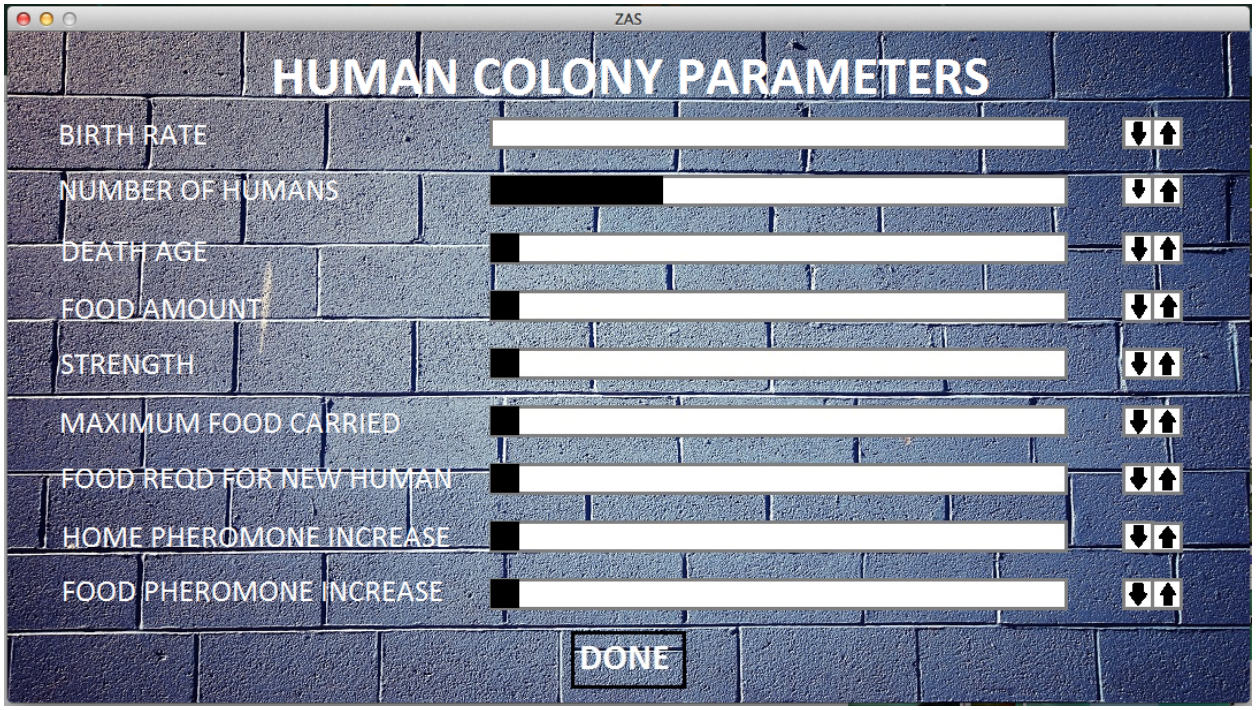

ZAS
ZOMBIE APOCALYSE SIMULATOR
BEGIN
BACK
HUMAN COLONY
ZOMBIE NEST
FOOD SOURCE

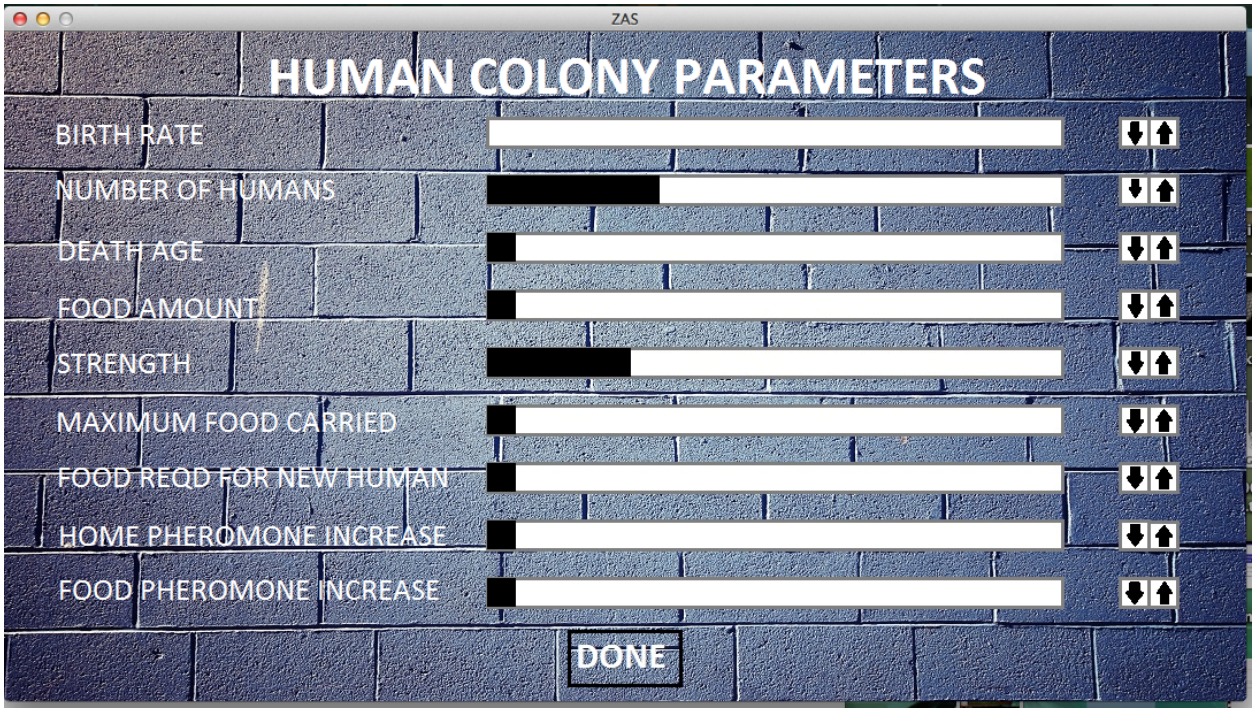

ZAS
ZOMBIE APOCALYSE SIMULATOR
BEGIN
BACK
HUMAN COLONY
ZOMBIE NEST
FOOD SOURCE

While setting the parameters for human colony this is the window that appears initially.

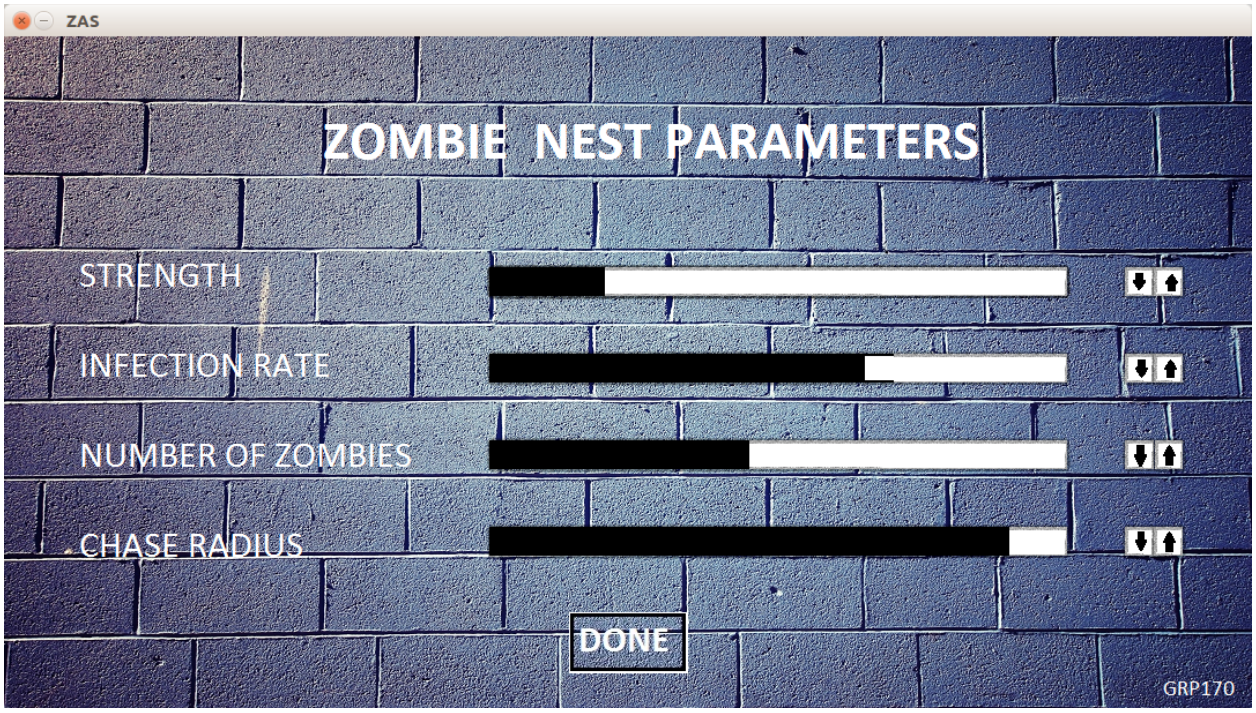


Parameter values can be increased or decreased using arrow buttons.

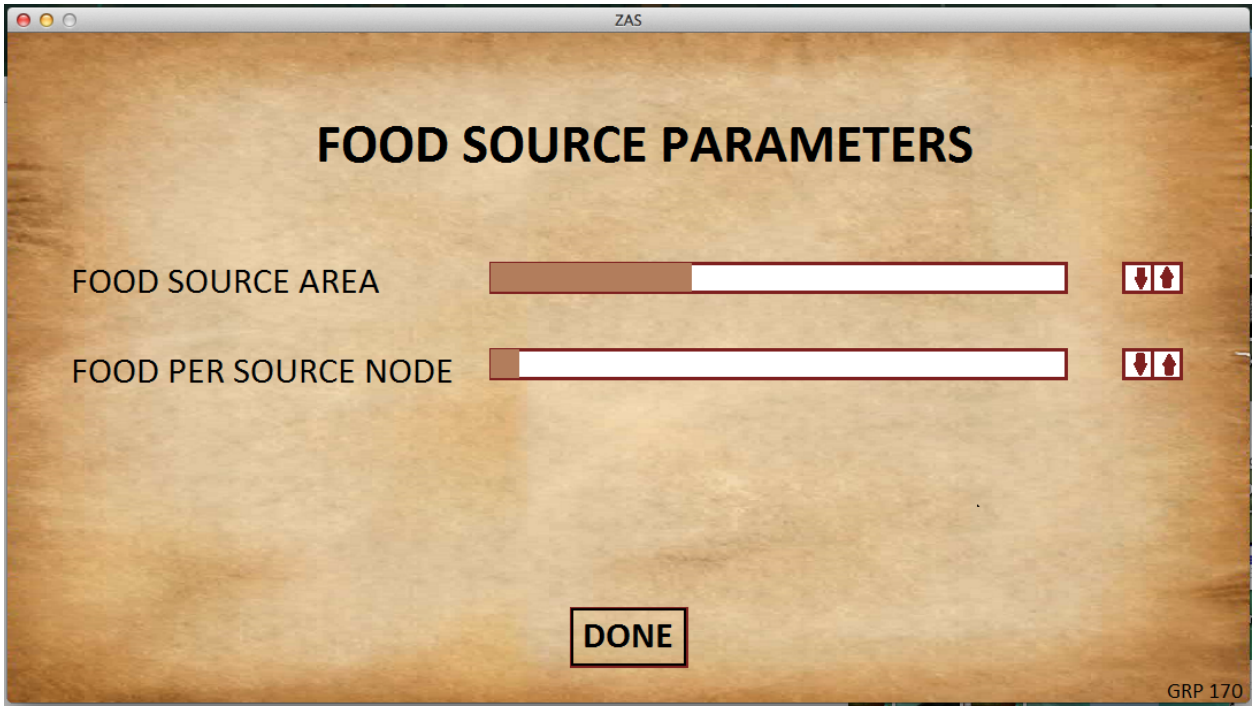




Similarly for zombie nest parameters



And for food sources parameters.



This window appears while setting the position of the newly added zombie nest/ human colony/food source.

Simulation of one human colony and one zombie nest happens as follow.

Simulation of one human colony and one zombie nest after some time. All zombies are dead and humans have formed a trail towards the food source.

Simulation can also be paused by the pause button.

Simulation can also be resumed by pressing the begin button.

Simulation can take place between multiple colonies and multiple zombie nests. Following is the example of one zombie nest, 2 human colonies and one food source.

After some time two human trails can be seen from the colonies to the food source. All zombies are dead.

5. TESTING STRATEGY AND DATA

SR NO	FUNCTIONALITY TO BE TESTED:	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT
1)	**Human colony inventory status**	A single human colony with a food source nearby.Human colony parameters are standard and not extreme.	The amount of food in the colony must change consistently according to Section 4: Human Colony Inventory Status	Humans being spawned is proof of food being deposited
2)a)	**Human activity**	A single human colony with a food source nearby.Human colony parameters are standard and not extreme.	The pattern of movement of humans should be similar to movement of ants (ACO).	Trails observed.
2)b)		A single human colony at the corner of simulation window,with trail decay rate = 100%, and a food source present nearby. No zombie nests.	Humans execute a random walk only and show no signs of following any optimal path.	Trails not observed at all.
2)c)		A single human colony at the corner of simulation window, a single food source at opposite corner and no zombie nests.	Humans should not be typically able to reach the food source before they reach natural death, and hence simulation should end with all humans dying	All working. Humans not able to reach food source if they are less in number, food is far away and their death age is small.

3)	**ZOMBIE ACTIVITY**	A single zombie nest anywhere in the simulation window with standard parameters, no human colony,no food sources.	Zombies execute a totally random pattern of movement.	Working as expected
4)	**HUMAN - ZOMBIE INTERACTION**	A single human colony with standard parameters at any position with a single zombie nest nearby, with no food sources.	All human-zombie interactions must result in the outcome being one among listed in Section 4.3 Processing Logic f.	All working except infect

SCREENSHOTS OF TEST CASES:

Test Case 1: Human Colony Inventory Status Initial

The blue dots are humans. Simulation is carried between one colony and one food source.

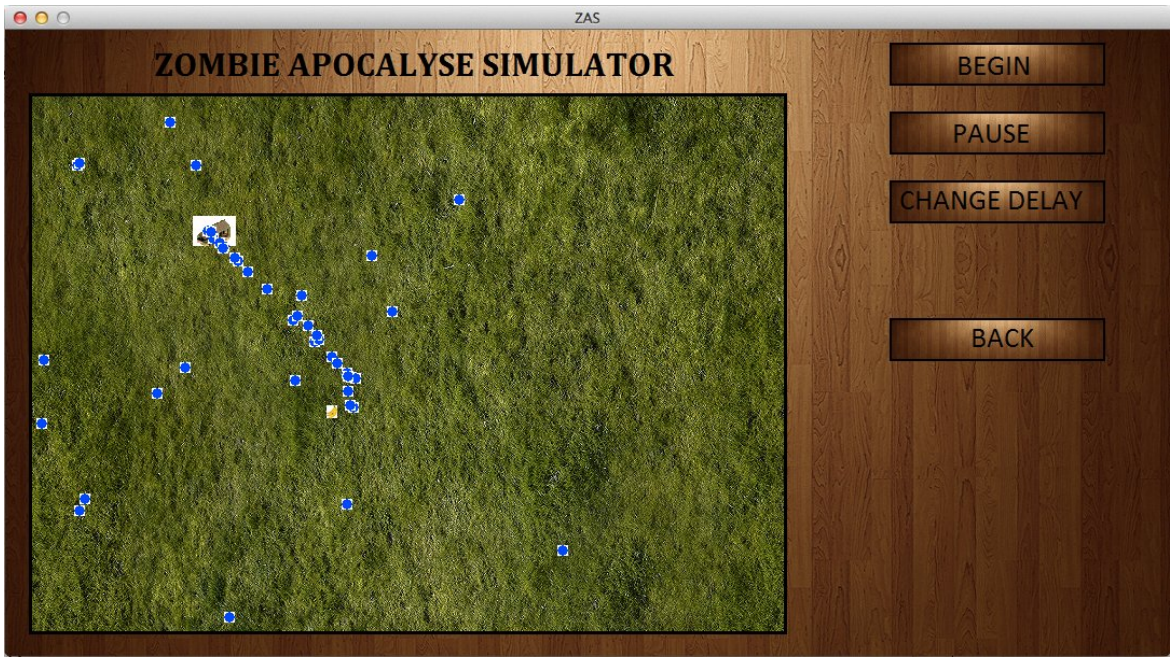
.

Test Case 1 : Human Colony Inventory Status Final:

Humans being spawned is the proof of food deposited at colony.

Test Case 2 a) Human Activity:

After some time trail of humans is observed from colony to food source



Test Case 2) b)

As trail decay rate is 100%, no human trails are being formed, near the 2 food sources.

Test Case 2) c) Human Activity Initial:

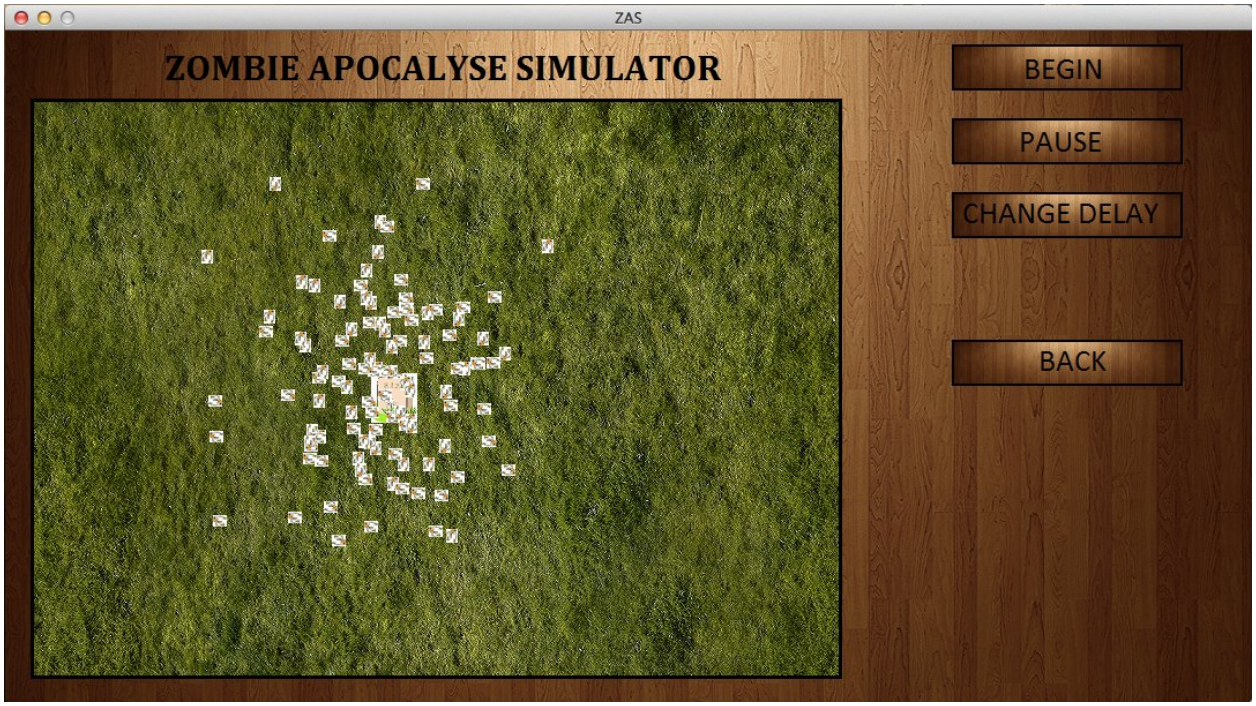
Colony at the corner of window and food source at large distance.

Test case 2 c) Final:

Humans having small death age die out before they can reach food at large distance. and come back to colony.

Test Case 3) Zombie Activity Initial:

Only zombie nest is present initially.



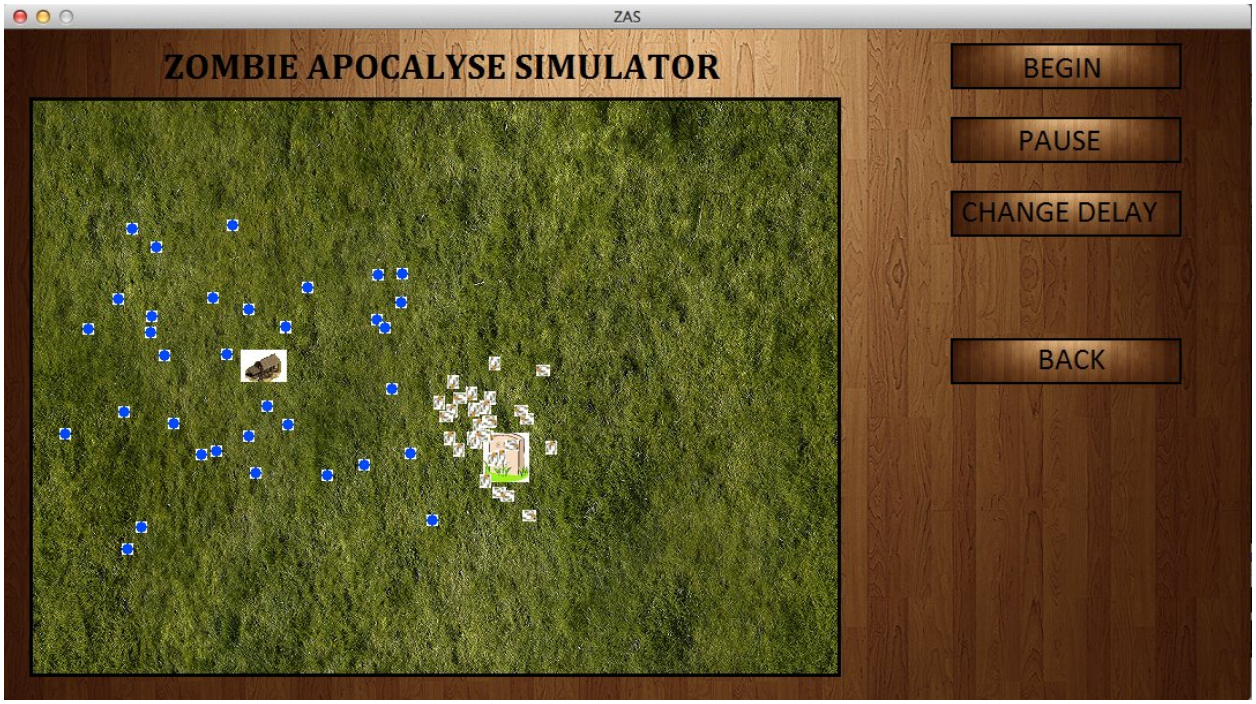
Test Case 3 b) Zombie Nest Final:

Zombies execute a totally random walk. No fixed pattern of movement is observed.

Test Case 4) Human Zombie Interaction:

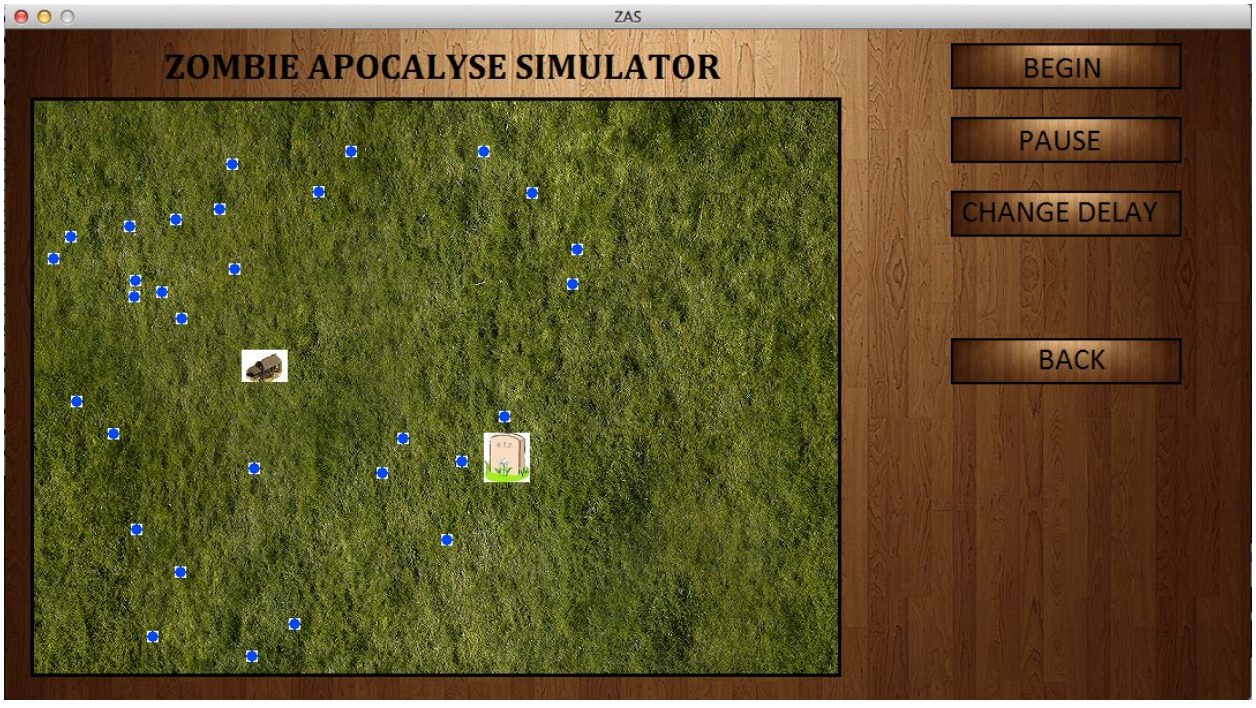
i)All zombies die (initial):

One human colony and one zombie nest is present.



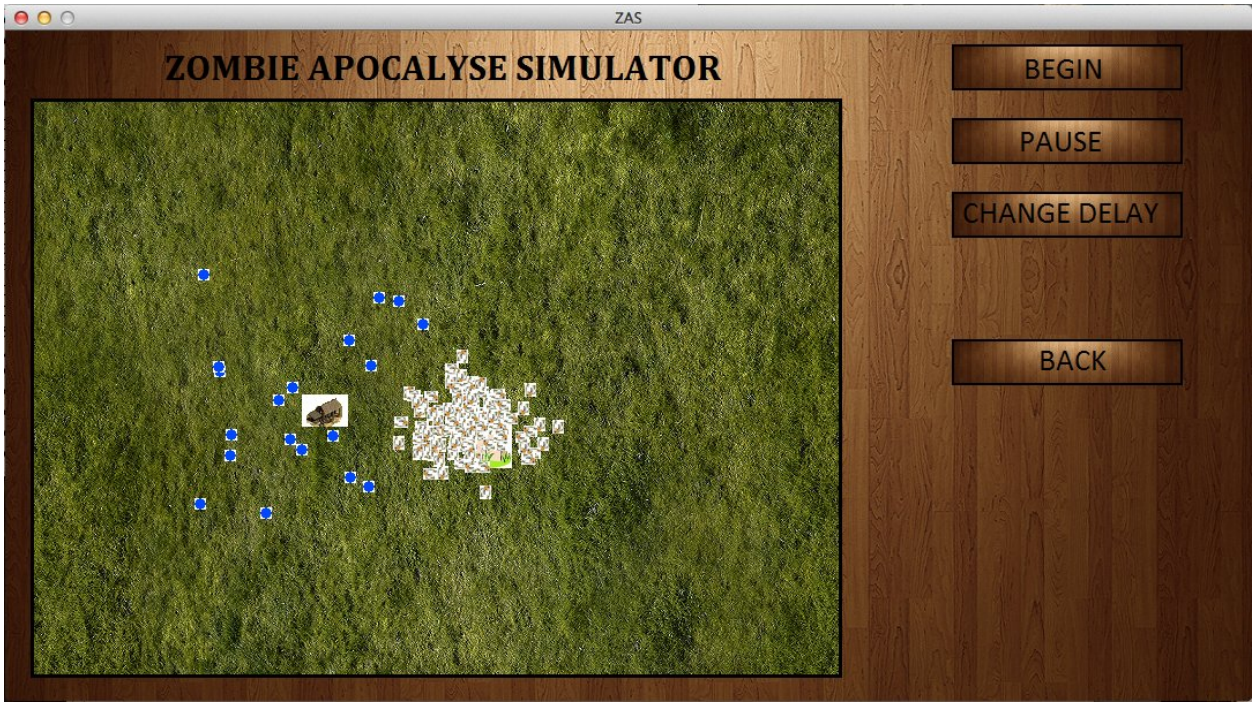
As the simulation happens Zombies and Humans start dying probabilistically depending on their strengths.

Finally only humans are left out:



ii)All Humans die(initial):

One human colony and one zombie nest is present. Number of zombies initially are more.



Finally all humans are killed and only zombies remain :


ZAS
ZOMBIE APOCALYSE SIMULATOR
BEGIN
PAUSE
CHANGE DELAY
BACK

6. DISCUSSION OF SYSTEMS:

A) What worked as per plan?

- The Final Ant colony optimization algorithm worked fine .
- SDL the graphics package we decided initially also worked out as per plan.
- The input windows from user such as human colony parameters, zombie colony parameter,food source also worked as per plan.
- The outcome of human - zombie interaction where one has to be killed also worked out perfectly.

B) What we added more than discussed in SRS?

- Added pause feature in the simulation to pause the simulation and also added the feature to resume it.
- Added change delay feature in the simulation so that user can increase/decrease speed of the simulation.
- Added graphical help window to help the user to operate the program.

C) Changes made in plan:

- Removed speed parameter and kept the speed of humans and zombies same.
- Removed outcome where after all zombies die , simulation ends so as to facilitate human colony- food source simulation.
- Had to change initial algorithm plan to make it better.

D) Challenges and solutions

- Implementing the ACO algorithm

SOLUTION:

Algorithm appears simple, but hard to implement. Used 2 types of pheromone(food pheromone and home pheromone)to solve the problem.

- Simulation involves handling a lot of objects in proper manner.

SOLUTION:

Learnt and used vector library for the same.

- Designing GUI and using images in SDL

SOLUTION:
Edited images in Paint for them to be used in SDL 2.0

- Choosing parameters for the algorithm and for the GUI from scratch.

SOLUTION:
Updated the list of parameters as and when required for the proper working of program.

7. FUTURE WORK:

- Implement the concept of day/night ,seasons etc.

- Humans/zombies calling for help during a fight, multiple creatures fighting and types of humans (like soldiers, elderly,children, workers)

- Different types of terrain

- Trade of food among human colonies and also provide the user dynamic control during simulation over single/multiple characters.
- Make the software availab

8. CONCLUSION

Our work can be generalized as simulation . It can be extended by adding more features to a full fledged strategy game where user can have dynamic control over the characters. Some features can be types of humans like soldiers , children , elderly, women, workers , etc, human/zombies calling for help in a fight.

9. REFERENCES:

- The ant simulator program : Myrmedrome

Ant Colony Optimization technique:

- http://www.aco-metaheuristic.org/about.html
- http://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms

C++ Style and Technique FAQ:

- http://www.stroustrup.com/bs_faq2.html

C++ coding standards guide:

- http://google-styleguide.googlecode.com/svn/trunk/cppguide.html

Tutorial for SDL 2.0 :

- http://lazyfoo.net/tutorials/SDL/

Using vector class:

- https://msdn.microsoft.com/en-us/library/9xd04bzs.aspx

Solved many of the debugging problems from:

- http://stackoverflow.com/

Github Repo :

- https://github.com/Sattwik/140070036_170.git

Youtube Video:

- https://www.youtube.com/watch?v=ARWIDJIBQ5I