1) We picked XNA because it is a well documented engine that has an ample amount of tutorials. It lacks an integrated physics engine, but we could easily hook Farseer into it and start working on the project quickly. One of the members of the team also had some familiarity with the engine going in and this gave us a kick start. We had considered other engines, but felt that the structure that XNA provided would enable us to spend more time learning about AI and less time fighting with the framework.

The Project:

Playing the Game:

You are controlling a purple circle. WASD moves the player around the screen. left clicking with the mouse creates/destroys walls blocks and right clicking will create non-moving agents. Some moving agents will be spawned when the map is created. pressing P pauses and R resets your position to a random free spot.

The meat of the code:

This Code deals with the following areas:

* Gamplay.cs(What is in charge of "The Game")
* Map
* Sensors
* Mobs
* Player

For the sake of discussion we will talk about each piece individually.

Gamplay.cs:

The Gameplay.cs file is manly in charge of everything. There are a variety of different files that are in charge of getting you there, but everything that we would consider "the game" is deal with in that file. The first thing it does is setup different pieces of data; globals, the map, the player and the mobs. After that point the file is essentially a manager. It's update function will go call update functions on the game's different parts(map and mobs) as well as handling the game's input(either handling the input itself like when pausing or calling a function for the player to handle the input). The same thing is done when drawing. Gamplay.cs goes through each of the parts of the game and draws them, the order of the calls will dictate what shows up on top though, so special care has to be taken to ensure that the correct parts show up on top.

Map:

The map is created when gamplasy.cs starts. It is a procedurally generated map based on a grid. To start the map creates rooms(rectangular areas) on the map and random open spots. If this was it though, you couldn't move anywhere so we then continually clear out space until all open floor is attached. Finally we cover the border of the map with walls, unfortunately this may destroy some connections. The rooms are attached by selecting a point on the map that is in a room and a random direction and then moving in that direction while clearing out walls until it hits another clear spot. Before each step of the wall remover it will have a 10% chance of turning. Originally there was no heading and the wall remover always took a random direction, but this lead to less pathways and more cleared out spaces. The wall remover tended to eat out an area as opposed to moving to a new one. We also had the color of the wall pieces change based on it coordinates as to create different regions in the world. The map generation is good for now, but will be a point of interest in the future.