My first idea was to make this more realistic, by implementing a map from google.maps, and load it in

to my program. I had to use matplotlib.image as module. This gave me the possibility to get the map as background and each pixel value. Since I was able to check the pixel value for all the pieces (zombies and humans), I could make different rules for where they could move.

One of the areas that I looked at was the university. Here I decided that zombies couldn’t move inside. This gave the humans a free area. This is also reasonably if we compare with how the humans often try to keep the zombies outside (or inside if that is more preferably).

Mode

One important discussion for this program has been the movement of the pieces. The basic program from KA consisted of humans and zombies. Here the zombies were looking for humans inside of a given radius, and moved towards the mass of population. The humans behaved similar, while these were moving away.

One problem I experienced with this movement pattern was that a zombie could end up between two humans, not deciding where to go. So I changed this so the zombie would follow the closest human.

To make this program as close to the model I had to add classes for infected and dead also. The infected humans would move random around until it came in a new state (dead or zombie). While the dead was acting dead(no magic to wake them up).

Since it was interesting to compare the results with the model, we wanted to make a random walk simulation. A random walk is more likely when we look at how deceases will be spread in a population. To make the movement smoother, I’m drawing a random number for number of steps in addition to direction.

The last mode that I made was moving smart. This is a combination of random walk and hunting. Here they will move random around until they detects any danger.

Adjust parameters

An important function for the program is to be able to run simulations with different parameters. Here I choose to implement a method from HPL compendium, which uses argparse. This is a method, which is readily for the users. This gives an explanation of the parameters. The users are also able to write short script for use of the program.

Super- and subclasses

After several hours of programming, I saw some residual code in the different classes. A lot of the basic things were similar. I made a subclass called creature. This took care of basic things as position, colour, direction and id. One advantage is of course shorter code, but this also makes the program more accessible to changes.

One step

For each round I run a function called one\_step. This function goes in to the update() in everyone. This is different between humans and zombies. I choose to take the battles in the zombie class.

Each round I update a matrix for the position for zombies and a matrix for the humans. The matrix is build up as a grid, where I do a integer division on the position to find the right element in the matrix. So when we call the update for a zombie, it will automatically look in the human matrix for victims to fight. It will check the same position as it self, and all the elements around. In total nine elements. It also checks the zombie matrix for support. And the numbers of creature will be able to affect the battle. A human can only fight one zombie each round.

After this each creature checks the surroundings before next step, and update direction and matrix. This choice depends on which mode they are sat to.

One challenge that I had with the movement structure was the trap that the corners became. The humans often ended up trapped. So I removed the walls to get rid of this problem. The zombies will now compare which way that is fastest, through the wall or not.

The two possibilities after a fight are infected and dead. So after an update, the program will check for these conditions by looking for the creatures colour. I have added a waiting step from human to infected and zombie to dead.

Pygame

A tip from a fellow student to improve the graphics was to add Pygame. Pygame is a module to create games, and it was new to me. So instead of only coloured dots, we could get up small images. This had of course no influence on the math behind, but it gives a better impression of the simulation.

The normal way to run Pygame is with a while loop, which stop when quitting the game. To build up each picture, I used screen.blit. This function adds the map as background and gave me the possibility to add figures up on that with screen.blit. To show the image, I used display.flip(). Since I also wanted to make movie out of it, I had to save each image. I found out that the .png format was ok.

I also made a break inside this while loop, which intended to quit if the number of zombies/humans was sat to zero.

Movie making

Here I had a lot of problems. There are several ways to make movies, and I ended up with avconv. This gave me the opportunity to choose fps and vcodec. This area is still open for several opportunities and I will probably be more comfortable in the future.