KTH Computer Science
and Communication

Simulation of Pedestrian Behavior

FRIDA JANSSON
fridajan@kth.se

ÅSA SPROGE
sproge@kth.se

Bachelor's Thesis at NADA
Supervisor: Michael Minock
Examiner: Mårten Björkman

Abstract

The idea for the project, which is the focus of this report, was based on the game "The Sims". The goal of the project was to find out whether it was possible to make a simulated person behave as a human while walking in a city environment. Our final simulation of a human can handle walking and avoiding obstacles, but is very primitive and lacks a lot of human qualities. To make a more accurate simulation would require a lot of work and knowledge.

Sammanfattning

Idén bakom projektet, som denna rapport bygger på, har hämtat inspiration från spelet “The Sims”. Målet med projektet var att ta reda på om det är möjligt att skapa en simulerad person som beter sig som en människa när den promenerar i en stadsmiljö. Vår slutgiltiga simulering av personen klarar av att gå och att undvika hinder, men är väldigt primitiv och saknar många mänskliga egenskaper. För att kunna skapa en mer realistisk simulering krävs en hel del arbete och kunskap.

Preface

This is a bachelor thesis for the course DD143X given at KTH in Stockholm, Sweden during the spring term 2012. The report writing and the programming has both been carried out by Frida Jansson and Åsa Sproge. The implementation has been a collaborative work and the report has been written together.

Contents

Chapter 1

Introduction

"The Sims" is a strategic life-simulation PC-game that was first released in February in the year 2000.[1] The game was developed by Maxis and published by Electronic Arts.[1] It has become a popular game with a lot of sequels such as "The Sims 2" and "The Sims 3" along with numerous expansion packs and spin-offs. The original edition alone has to date been sold in more than 16 million copies.[1]

In the game the player creates human-like simulations, called "Sims", and then follows them through their life, controlling and making decisions for them. The player builds and furnishes houses for the Sims and creates whole life stories for them. The stories include work achievements, relationships and even the making of new generations.

Inspired by The Sims we wanted to create a simulation of a human and the goal was to make it act as human-like as possible. The human simulations in The Sims do not often make their own decisions but when they do, the result is usually poor. Players of the game discover that for example when a Sim does not have anything to do, it rather performs unrealistic actions than something useful that a human would do. We wanted our simulations to make as good decisions as possible and also to think through their decisions as thoroughly as possible.

1.1 Problem statement

This study was performed with the intention of finding out how detailed human behavior could be simulated. We wanted to answer the following questions:

> Is it possible to place a Sim, that has human feelings such as hunger, in a simulated town and get it to reach a geographic point behaving like a real human? Would it even be possible for the Sim to walk down a street following the pavement, without being hit by cars and without walking into other pedestrians?

The topic of simulating human behavior is interesting because it can be used in a lot of different contexts:

- When studying humans exposed to extreme situations such as tsunamis and earthquakes, experiments with Sims could be performed, instead of studying real humans potentially exposing them to danger.
- It could be used to increase credibility and detail in computer generated entertainment, such as computer games and special effects in movies.
- When developing robots with the ambition to make them behave as humans.

1.2 Limitations

To create a large simulated surrounding where all the different aspects of human life are covered is a great deal of work, especially if the mission is to equip all simulated people with human-like decision making. Therefore we limited our study to a Sim that was walking from a point A to a point B. Since we were only interested in their walking behavior our study did not include the life events known from the game "The Sims", such as work, school, family and relationships. Our simulated town was implemented in 2D and consisted of a map in a summer setting. The simulated people that we developed were represented as moving dots and therefore we did not include the ability of speech.

1.3 Terminology

The Sims

The popular strategic life-simulation game that was our project inspiration.

Sim

A simulated person.

Artificial Intelligence, AI

Russel and Norvig defines AI in their book as "the study of agents that receive percepts from the environment and perform actions". [2, p. VII]

Agent

A simulated person, an other word than "Sims" used within Artificial Intelligence.

Heuristics

Rules that finds an effective and easy solution to a problem, but not necessarily the correct one.

Chapter 2

Background

2.1 Artificial Intelligence

The Sims is a good example of how human behavior can be imitated and an example of "Artificial Intelligence". Russell and Norvig describes Artificial Intelligence as systems that think like humans, act like humans, think rationally and act rationally.[2, p. 2]

Artificial Intelligence is often divided into two subcategories: "weak AI" and "strong AI". Weak AI is about the simulation of intelligence: even though the computers can perform intelligent actions they lack real understanding of what they are doing.[3] They are intelligent in some ways but have no mind nor consciousness. Examples of weak AI are aeroplanes that can take off and land themselves and programs that can diagnose heart diseases.[4]

Strong AI is described by Ray Kurzweil as "machine intelligence with the full range of human intelligence".[4] Philosophically this means that the computers actually have an understanding of their actions and that they have minds of their own.[3] To this date, computers like the ones described above have not been created.[4]

2.2 Game Artifical Intellegence

The Sims uses "Game Artificial Intelligence" which is a kind of weak AI. The game has boundaries that the player has to keep within, so it has a fairly simple use of Artificial Intelligence. For example the games with use of Game Artificial Intelligence often include events which are not always realistic. In The Sims we have seen two Sims bumping into each other and for some time standing literary "in half" of each others bodies. A slower reactivity is also often implemented into battle games because of natural reasons: if the simulated AI-players are not constrained they would too easily win over the human players!

2.3 Crowd simulation

While our simulation is based on The Sims it also resembles so called "crowd simulation". A notable example of crowd simulation is the software "Massive" that was developed during the filming of the "Lord of the Rings"-trilogy[5]. Massive was used for simulating lots of warriors on the battlefield for the numerous battle scenes of the film. Instead of animating each warrior Massive offered a simple way of creating large numbers of warriors which looked natural on screen.

Crowd simulation is not only used as special effects for movies. The research group Gamma, based in North Carolina, USA, is working with what they call "geometric algorithms for modeling and motion".[6] Their work can be used for evacuation plans, architectural planning and creating virtual worlds.

Chapter 3

Analysis

3.1 Study

The goal for this project was to create a simulated person who would behave as close to a human as possible. The Sim should be able to walk from a point A to a point B in a town while making human-like decisions. In the beginning of the project the initial question was the following: how do humans really behave?

3.1.1 Approach

To find out how real humans would perform the task of walking from a point A to a point B we decided to make a study. The goal of the study was to find out more about the decisions that humans make while walking and why they essentially make them. Our simulation would need to specify rules about all decisions that the Sim would make, decisions that real humans never really think about. From the study we wanted to get a detailed overview of the situations that people face in the streets. The study was made on six different people by emailing them instructions and questions. Six people seemed sufficient for such a small study like this. The study consisted of four questions that had to be answered for each new event the test subject faced during an optional walk from a point A to a point B. The questions that the subjects were supposed to answer for each event were:

1. What did you do?
2. Why did you do it?
3. How did you do it?
4. What consequences did your action have?

3.1.2 Problems

Three of our test subjects had some trouble understanding the instructions. They said that they did not really understand the meaning of the study and the necessity

of detail. But after some further instructions they managed to complete the study. We found their problems interesting because it proved to us how easy the task of walking down a street is to humans and how little people think about the decisions that are made in that situation.

When we got the results we were surprised by how many of the decisions that involved avoiding slippery areas of ice. We had not considered how the season would affect the decision making of our subjects and we had not planned to do a simulation in a winter setting. We thought about simulating a winter setting with its accompanying problems but we rather wanted to make a general simulation. Therefore, unfortunately, a lot of the answers that we got from the study were not of good use in our implementation.

3.1.3 Results

Despite the problems, we got some usable results. The subjects mentioned the following decisions:

- Changing side of the street to avoid blocking objects.
- Walking on the street to avoid obstacles (if there are no cars in sight).
- Stopping or changing direction while walking or running due to blocking object.
- Walking fast while in a hurry.
- Making short cuts.
- Walking on the pavement if there is one.
- Walking as far to the side of the street if there is no pavement.
- Avoiding walking too close to strangers by either slowing down or walking past.
- Stopping while looking left and right searching for cars when crossing the street.
- Using the pedestrian crossing, if there is one, while crossing the street.

During the implementation phase we tried to implement as many of these decisions as possible.

3.2 Implementations

3.2.1 Approach

The implementation of the project was done over a couple of stages, starting small and then growing in detail. We followed this approach because, as this is a very complex problem, we thought it would be easier if we divided the problem into smaller ones. The benefit of starting small was that we got familiar with the problem while ignoring the most difficult parts. We could advance in a short amount of time and we easily saw the flaws in the simulation which could be improved in the next stage.

At first we built a simple environment with only one street, pavement, a couple of objects and two Sims. At this stage we wanted to evolve a Sim that walked on pavements, walked straight when not restricted and that never bumped into other Sims. Next we made the surroundings more complex, with more streets and with more objects such as a moving car and more Sims. At our final stage we wanted our Sim to have needs that influenced its behavior and its route. This unfortunately proved to be too difficult and time consuming within the range of this course so we did not attempt to implement this stage.

We implemented our simulation in Java. For the graphical parts we used the package "awt", which enabled us to draw geometric figures. We chose Java because we both had experience in that language. Our graphic map was implemented as a 2D map seen from above with our Sims represented as a moving circles.

The code for the classes `Sim`, `MovingObject` and `CoordinateSurrounding`, can be found in the appendices. The main class is not included in the appendices due to its size but the important parts are featured in the following sections. We used other classes as well, but they are not necessary for understanding the implementation. Therefore they are not included in the appendices.

3.2.2 Stage 1

A single moving Sim

During this stage we started small, as planned. We created a moving Sim and placed it in an empty environment. The first requirement for the Sim (which we named "Estelle") was to get it to walk from a starting point towards a specified goal. We made it able to walk straight or diagonally, depending on the goal's position.

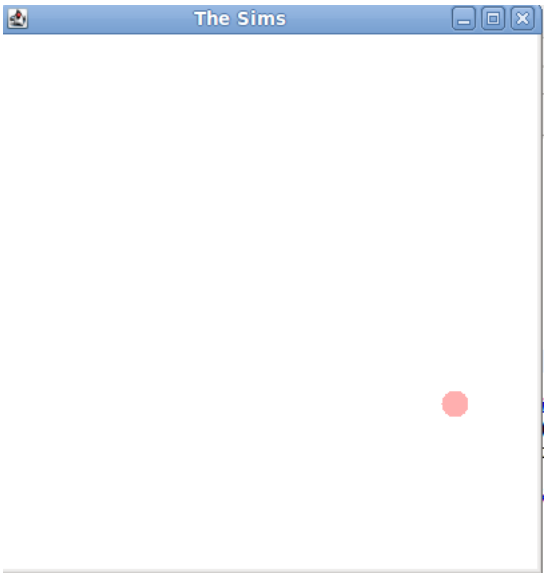

Figure 3.1. A moving Sim represented as a circle in an empty environment.

The move-algorithm that was implemented used coordinates to identify the position of the Sim and the goal. Every time the Sim made a move the algorithm compared its coordinates to the coordinates of the goal in order to check if it should walk up, down, left or right. This was repeated in a loop until it reached the goal. It is relevant to know that the coordinate system in Java places, unlike the standard coordinate system, origin in the upper left corner and increases the y-axis downwards.

```
public static void move(Sim sim){
      if(sim.getX() > goalX) {
          sim.moveWest();
      } else if(sim.getX() < goalX) {
          sim.moveEast();
      }
      if(sim.getY() > goalY){
          sim.moveNorth();
      } else if(sim.getY() < goalY) {
          sim.moveSouth();
      }
  }
```

Listing 3.1. The method for making the Sim move.

Adding obstacles

When the Sim could walk successfully, we extended the environment by adding a street, pavement, a crossing and houses. The requirement was now to get it to walk on the pavements and over the crossing, as a human would do. We added rules which specified the areas that was not allowed to walk on for the Sim. This was

implemented by adding coordinates for all forbidden areas and objects in an array called `coordList` . Every time the Sim attempted to move, the algorithm compared the coordinates of the desired next step with the coordinates in the array of the forbidden areas. This comparison was done to make sure that the Sim would avoid obstacles such as streets and houses. We made our Sim use the crossing by simply not including the crossing's area in the `coordList` array, see figure 3.2.

```
/**
 * Checks if the Sim is allowed to go west, returns true if allowed.
 */
public static boolean checkWest(Sim sim){
    if(!coordinates.notAllowed((sim.getX()-sim.getSpeed()), sim.getY(),
        sim.getSize())){
        return true;
    }
    return false;
}
```

Listing 3.2. The method for checking if the Sim is allowed to go west. The methods for the other directions are very similar. Calls the `notAllowed` method in the `CoordinateSurrounding` class, see listing 3.3.

```
/**
 * Checks if the input coordinates are allowed to visit.
 * Returns true if they are not allowed, else false.
 */
public boolean notAllowed(double x, double y, double size){
    for(int i=0; i<coordList.size(); i++){
        if(coordList.get(i).isWithin(x, y)){
            return true;
        } else if(coordList.get(i).isWithin((x+size), (y+size))){
            return true;
        } else if(coordList.get(i).isWithin((x+size), y)){
            return true;
        } else if(coordList.get(i).isWithin(x, (y+size))){
            return true;
        } else if(coordList.get(i).isWithin((x+(size/2)), y)){
            return true;
        } else if(coordList.get(i).isWithin((x+(size/2)), (y+size))){
            return true;
        } else if(coordList.get(i).isWithin(x, (y+(size/2)))){
            return true;
        } else if(coordList.get(i).isWithin((x+size), (y+(size/2)))){
            return true;
        }
    }
    return false;
}
```

Listing 3.3. The method for checking if the input coordinates are allowed for the Sim to visit. The method compares all the edges of the Sim, as if it was a square, to make sure that no part of the Sim will enter a forbidden area.

Figure 3.2. The Sim crosses the street correctly by using the crossing. The grey areas represents the street and the white represents pavements and a crossing. The yellow represents houses.

The main issue when adding obstacles to the map was to make the Sim walk around them. The problem aroused when the Sim's x-coordinate was the same as the goal but not the y-coordinate. If an obstacle was in its way along the y-axis the Sim got stuck because it did not know that it could move sideways to get around the forbidden area. To solve this problem we added a method that kept the Sim going sideways until it could move up or down again. This was implemented to work with whatever direction the Sim was stuck in.

```
public static void findNewPath(Sim sim){
   if(checkSouth(sim)) {
       while(!checkWest(sim) || !checkEast(sim)){
           if(checkSouth(sim)) {
               delay();
               sim.moveSouth();
           } else {
               moveSideways(sim);
               break;
           }
       }
   } else if(checkWest(sim)) {
       while(!checkNorth(sim) || !checkSouth(sim)){
           if(checkWest(sim)) {
               delay();
               sim.moveWest();
           } else {
               moveVertically(sim);
               break;
           }
       }
   }
/.../
```

Listing 3.4. The method that was used when the Sim got stuck.

Adding another Sim

The final act in this stage was to add a stationary Sim (named "Christer") who Estelle was supposed to avoid. This was achieved by using the method for avoiding forbidden areas described in listing 3.3. The primary reason for not making him move yet was that the code in stage 1 was not suited for parallel movement of objects.

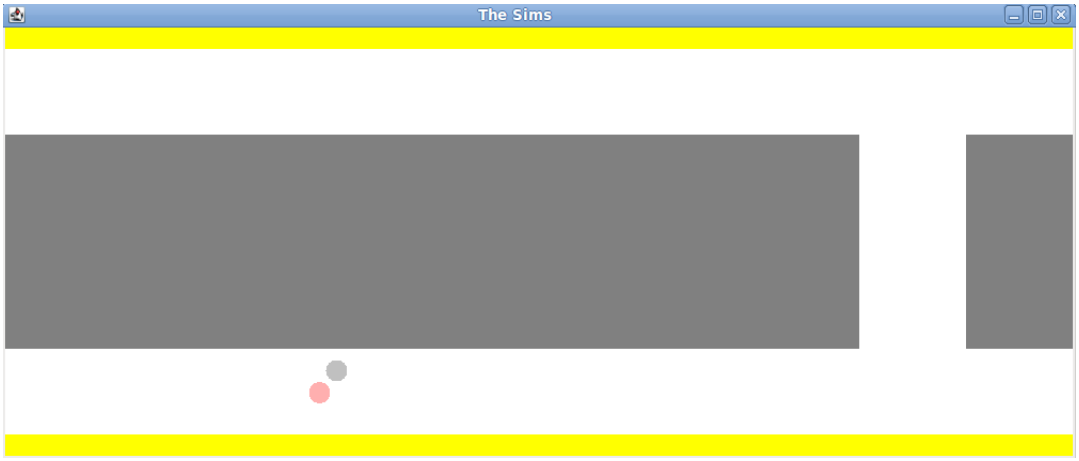

Figure 3.3. Estelle avoids Christer on her way to her goal.

The Sim's behavior in the first stage was based on very primitive rules and therefore it did not act very human-like because its reactions and movements were based on algorithms. As an example the Sim walked too close to the street compared to what a human would do. A human would walk in the middle of the pavement to avoid walking too close to the street and too close to the houses. This behavior was something that we wanted to improve in the next stage. We also wanted to implement parallel movement to create a more complicated and realistic surrounding.

3.2.3 Stage 2

Multiple moving Sims

The initial approach in the second stage was to make Sims and other objects move simultaneously. In stage 1 the move method was implemented as a loop so that if it had been used for more than one Sim, the current Sim would have to reach its goal before the next Sim could start moving. This meant that it was impossible to have more than one Sim moving at the same time.

Initially we tried to implement multi-threading but the result was not graphically satisfying. Instead we created four different Sims and put them in an array called `listofsims`. Then we made a loop that iterated through `listofsims`, making each Sim move one step at a time. Because this was carried out very fast by the computer it created the illusion that they were moving at the same time, see listing 3.5.

```
while(!finished){
        for(int i=0; i<listofsims.length; i++){
              delay();
           Walk(listofsims[i]);
        }
        /../
}
```

Listing 3.5. The loop that made the illusion of parallel movement.

Now that the Sims were moving independently at the same time we had to make them avoid bumping into each other. We used the `notAllowed` method again (see listing 3.3) and added the Sims' coordinates into `coordList`. The biggest issue with this was that each Sim not only interpreted other Sims as forbidden areas, but also themselves. In order for them to be able to move this was not allowed to happen. This was avoided by keeping track on the indices of the Sims in the array by storing them in the Sim objects in the Sim class.

The next step was to evaluate the Sims' ability to avoid solid objects and to find a way around them. The method used for this in stage 1 was very poor and needed to be improved. Problems occurred for example when a Sim was exposed to more than one obstacle at the same time. It only managed to handle one of the obstacles at a time and became caught in a track that it could not get out of.

Our solution was to create a heuristic method that made the Sim follow a pattern that would help it find a way out of its track. The method made the Sim try different directions, if one did not work then it would try a new one and so on, according to listing 3.6.

```
public static void findNewPath(Sim sim){
    if(sim.getX() == sim.getGoalX() && sim.getY() > sim.getGoalY()){
        while(!checkNorth(sim) && moveSpecificDirection("east", sim)){
            delay();
        }
        if(moveSpecificDirection("north", sim)){
            return;
        } else {
           while(!checkNorth(sim) && moveSpecificDirection("west", sim)){
                delay();
            }

            if(moveSpecificDirection("north", sim)){
                return;
            } else {
                while(!checkWest(sim) && moveSpecificDirection("south",
                    sim)){
                    delay();
                }

                if(checkWest(sim)){
                    while(!checkNorth(sim) && moveSpecificDirection("west"
                        , sim)){
                        delay();
                    }
                    if(moveSpecificDirection("north", sim)){
                        return;
                    }
```

```
                } else {
                    while(moveSpecificDirection("east", sim)){
                        delay();
                    }
                    if(moveSpecificDirection("north", sim)){
                        return;
                    }
                }
            }
        }
    }
```

Listing 3.6. The heuristic method that was used for a trapped Sim that could not move north. The method was supposed to lead it into freedom.

Now we had four Sims that could move around freely while avoiding each other and while avoiding an obstacle, see figure 3.4.

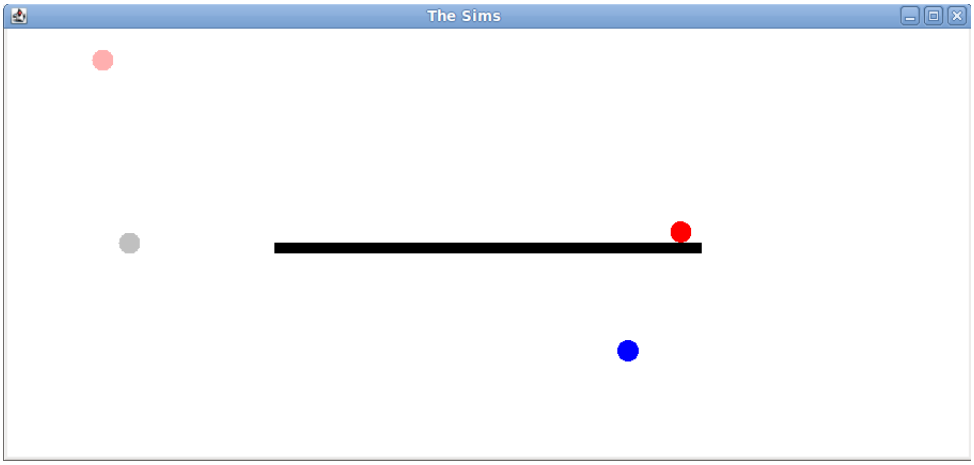

Figure 3.4. Simulation of four separate Sims who moves randomly while avoiding each other and an obstacle.

A bigger surrounding

When we had succeeded with the above we changed the empty surrounding to one that consisted of a street with two blocks, crossings, pavements and houses, see figure 3.5. A requirement was now to create a moving car that the Sims should avoid. We created a car that drove along the street and then the Sims were made to avoid it by using the `notAllowed` method once more. The method was changed so that it could identify which type of object the Sim confronted. When the Sim was crossing the street and was blocked by the car, the Sims were made to stop and wait for the car to move, instead of finding an alternate path, see listing 3.7. We had now succeeded with creating an environment with moving Sims, that in some way resembled human behavior.

```
 /**
  * Checks if the Sim is allowed to go west. Returns true if allowed.
  */
public static boolean checkWest(Sim sim){
    while(coordinates.notAllowed((sim.getX()-sim.getSpeed()), sim.getY(),
        sim.getSize(), sim.getID()) == 3){
        delay();
        moveCar();
    }
    if(coordinates.notAllowed((sim.getX()-sim.getSpeed()), sim.getY(), sim
        .getSize(), sim.getID()) != 0){
        return false;
    }
    return true;
}
```

Listing 3.7. The altered code for checkWest, where the Sim waits for the car to move.

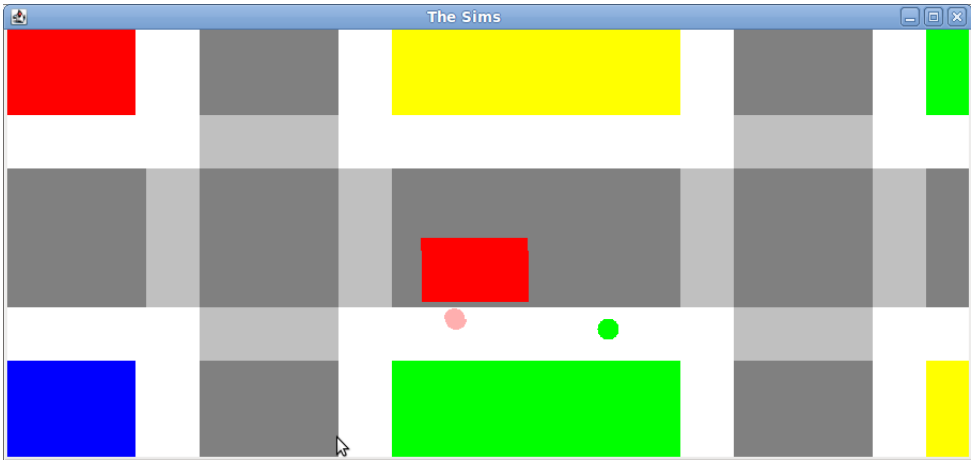

Figure 3.5. The environment for stage 2 with a moving car and two moving Sims. The grey areas represents the streets, the light grey are crossings, the white are pavements and the red rectangle is the car. The rest represents houses.

At first we planned to extend the project to include a bigger environment and more realistic Sim behavior, but during stage 2 we realized that it was not feasible to achieve this within our time plan.

Chapter 4

Result

Our problem statement for this thesis was the following:

> Is it possible to place a Sim, that has human feelings such as hunger, in a simulated town and then get it to reach a geographic point behaving like a real human? Would it even be possible for the Sim to walk down a street following the pavement, without being hit by cars and without walking into other pedestrians?

As a result of this project we have successfully done a simulation of human behavior in a city surrounding. We have graphically simulated people walking on pavements and using crossings when crossing the streets. Our Sims avoid bumping into other pedestrians and they also avoid being hit by cars. In other words we succeeded with implementing the last sentence of the problem statement.

We made a study to find out what real human pedestrian behavior would be and we have compared our final simulation to the results of the study. We have implemented the behaviors with the highest relevance:

- Stopping or changing direction while walking or running due to blocking object.
- Making short cuts.
- Walking on the pavement if there is one.
- Use the pedestrian crossing if there is one while crossing the street.

Unfortunately our simulation lack some detail of real human behavior: the Sims use illogical problem solving to find their way out of traps, they do not plan their route ahead, nor do they keep normal distance to the street, the houses and to other Sims. This means that we did not succeed with implementing the first sentence of the problem statement.

Chapter 5

Discussion

5.1 Use of Artificial Intellegence

Our simulation uses "Weak AI", since the Sims in our implementation lack any real understanding of what they are doing and since all their actions are results of heuristics. We specifically found it difficult to come up with a realistic and effective algorithm for making the Sims walk around obstacles. If we had studied more human problem solving and statistics about human behavior, we would have been able to write better algorithms and heuristics. Then we could probably have created more realistic Sims.

5.2 Difficulties

What we do realize after the completion of the project is that thorough simulations of human behavior are long processes and that we never would have been able to get close to humanlike results within our time range. One of the hardest parts with this simulation was attempting to make the Sim think ahead and in some way interpret the surrounding. Unfortunately our Sims only looked ahead at the pixels of the next attempted move and they did not have eyes to scan the surrounding with. This disadvantage made it difficult to implement some of the results from our study. For example the behavior to walk on the street instead of the pavement to avoid obstacles when there is no car around would require the Sim to be able to determine when a car is approaching. This is possible to implement (for example with a path finding algorithm), but we found it difficult to combine methods to make the Sim both move accurately as well as to anticipate upcoming incidents.

5.3 Possible usage

In order to use our simulations for a usable purpose it would need to go through some improvements. If this was achieved the simulation could for example be used for studying evacuation plans or primitive games. Even though the idea for the

project was taken from the game "The Sims", the final result did not have many resemblances with it. This is because we early in the process decided not to include the life events from the game in our simulation and also that the graphics are very unlike each other.

5.4 Expected result

Our final result is pretty much what we expected to achieve, although we had hoped to get a more accurate result. Since we know that similar kinds of simulations exist it was not so much a question of "is it possible?" but rather a question of "how do we do it?". Although we only succeeded with the last part of the problem statement it does not necessarily mean that it is impossible to accomplish the first part, merely that it was not possible for us with the knowledge, tools and time that we had during the project.

Chapter 6

Conclusions

Making a realistic simulation of human behavior, even if it is only for pedestrian behavior, is difficult. There are many different behaviors and situations that need to be implemented, and to take them all into consideration would require a lot of work and knowledge. However, it was possible for us to make a simulation that was limited to certain requirements, such as moving in a relatively uncomplicated environment and avoiding obstacles. The shortcomings in our simulation are due to difficulties with making realistic and effective algorithms as well as implementing some kind of perception of the surrounding. Simulations of human behavior is an interesting topic and although this thesis has given us some insight on the subject there is much more to learn in order to be able to make more realistic simulations.

Bibliography

[1] Wikipedia. The Sims. 2012-03-06 *http://en.wikipedia.org/wiki/The_Sims*. (Viewed 2012-03-07).

[2] Russell S, Norvig P. *Artificial Intelligence - a Modern Approach*. Pearson, New Jersey, 2003, Second edition.

[3] Hauser, Larry. Chinese Room Argument. 2005-07-27 *http://www.iep.utm.edu/chineser*. (Viewed 2012-03-07).

[4] Kurzweil, Ray, Long Live AI. 2005-08-15 *http://www.forbes.com/home/free_forbes/2005/0815/030.html*. (Viewed 2012-03-07).

[5] Massive Software. 2011 *http://www.massivesoftware.com/film.html*. (Viewed 2012-03-07).

[6] The Gamma research group. *http://gamma.cs.unc.edu/research/crowds/*. (Viewed 2012-03-07).

Appendix A

Source code for “Sim”

```
import java.awt.*;
import java.awt.geom.*;

/*
* A Sim (simulated person). The Sim is represented by a circle.
*/
public class Sim extends MovingObject {
    private Ellipse2D.Double circle;
    private Color color;
    private double diameter;
    private double speed = 0.5;

    /**
    * Creates a Sim with given diameter, position in x and y, color, goal
        coordinates and index.
    */
    public Sim(double diameter, double xpos, double ypos, Color color, Canvas
         canvas, double goalX, double goalY, int id){
       super(xpos, ypos, color, canvas, goalX, goalY, id);
       circle = new Ellipse2D.Double(xpos, ypos, diameter, diameter);
       this.color = color;
       this.diameter = diameter;
    }

    /**
    * Draws the Sim on the canvas.
    */
    public void draw() {
       canvas.setForegroundColor(color);
       canvas.fill(circle);
    }

    /**
    * Erases the Sim from the canvas.
    */
    public void erase() {
       canvas.setbackgroundColor(Color.white);
       canvas.erase(circle);
    }

    public void moveEast() {
       erase();
```

```
        xPosition += speed;
        circle.x = xPosition;
        draw();
    }

    public void moveWest() {
        erase();
        xPosition -= speed;
        circle.x = xPosition;
        draw();
    }


    public void moveNorth() {
        erase();
        yPosition -= speed;
        circle.y = yPosition;
        draw();
    }

    public void moveSouth() {
        erase();
        yPosition += speed;
        circle.y = yPosition;
        draw();
    }

    public double getSpeed(){
        return speed;
    }

    public double getSize(){
        return diameter;
    }
}
```

Listing A.1. Code for the `Sim` class

Appendix B

Source code for “MovingObject”

```
import java.awt.*;

/*
* Used for a moving object, for example a Sim or a car.
*/
public class MovingObject {
    public Color color;
    public double xPosition;
    public double yPosition;
    public Canvas canvas;
    private double goalX;
    private double goalY;
    private boolean finished;
    private int id;

    /**
    * Creates an moving object with given positions in x and y, color, goal
        coordinates and index.
    */
    public MovingObject(double xpos, double ypos, Color color, Canvas canvas,
         double goalX, double goalY, int id){
       this.color = color;
       this.canvas = canvas;
       xPosition = xpos;
       yPosition = ypos;
       this.goalX = goalX;
       this.goalY = goalY;
       finished = false;
       this.id = id;
    }

    public double getX(){
       return xPosition;
    }

    public double getY(){
       return yPosition;
    }

    public double getGoalX(){
       return goalX;
```

```
    }

    public double getGoalY(){
       return goalY;
    }

    public boolean finished(){
       return finished;
    }

    public void reachedGoal(){
       finished = true;
    }

    public void setGoal(double x, double y) {
       goalX = x;
       goalY = y;
    }

    public int getID(){
       return id;
    }
}
```

Listing B.1. Code for the `MovingObject` class

Appendix C

Source code for "CoordinateSurrounding"

```
import java.util.ArrayList;

/*
* Sets up the coordinates for objects.
*/
public class CoordinateSurrounding {
   private ArrayList<Coordinates> list;

   /**
    * Creates a list for the coordinates of the objects.
    */
   public CoordinateSurrounding() {
       list = new ArrayList<Coordinates>();
   }

   /**
    * Adds the coordinates of a new object to the list.
    */
   public void addCoordinates(double startX, double startY, double endX,
       double endY) {
       list.add(new Coordinates(startX, startY, endX, endY));
   }

   /**
    * Adds the coordinates of a new object to the list with a given index.
    */
   public void addCoordinates(double startX, double startY, double endX,
       double endY, int i) {
       list.add(new Coordinates(startX, startY, endX, endY, i));
   }

   /**
    * Changes the coordinates of a an object with the given index.
    */
   public void changeCoordinates(int i, double startX, double startY, double
       endX, double endY) {
       list.get(i).changeCoordinates(startX, startY, endX, endY);
   }
```

```
    /**
     * Checks if a Sim is allowed to move the specified coordinates.
     * Returns 0 if the given coordinates are ok, 1 if there is a road,
     * 2 if there is a house, 3 if there is a car and 4 if there is another
         Sim.
     */
    public int notAllowed(double x, double y, double size, int index){
        for(int i=0; i<list.size(); i++){
            if(index != list.get(i).getIndex()){
                if(list.get(i).isWithin(x, y)){
                    return determineObject(list.get(i).getIndex());

                } else if(list.get(i).isWithin((x+size), (y+size))){
                    return determineObject(list.get(i).getIndex());

                } else if(list.get(i).isWithin((x+size), y)){
                    return determineObject(list.get(i).getIndex());

                } else if(list.get(i).isWithin(x, (y+size))){
                    return determineObject(list.get(i).getIndex());

                } else if(list.get(i).isWithin((x+(size/2)), y)){
                    return determineObject(list.get(i).getIndex());

                } else if(list.get(i).isWithin((x+(size/2)), (y+size))){
                    return determineObject(list.get(i).getIndex());

                } else if(list.get(i).isWithin(x, (y+(size/2)))){
                    return determineObject(list.get(i).getIndex());

                } else if(list.get(i).isWithin((x+size), (y+(size/2)))){
                    return determineObject(list.get(i).getIndex());
                }
            }
        }
        return 0;
    }

    /**
     * Determines the type of an object with the given index.
     */
    public int determineObject(int index){
        if(index == -1){
            return 1;
        } else if(index == -2){
            return 2;
        } else if(index == 2){
            return 3;
        } else {
            return 4;
        }
    }
}
```

Listing C.1. Code for the `CoordinateSurrounding` class