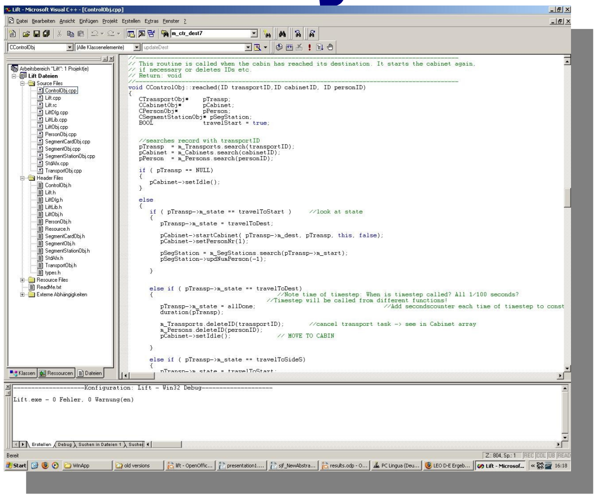
„Wettbewerbsarbeit Schweizer Jugend forscht
41. Nationaler Wettbewerb jung und innovativ Fribourg“

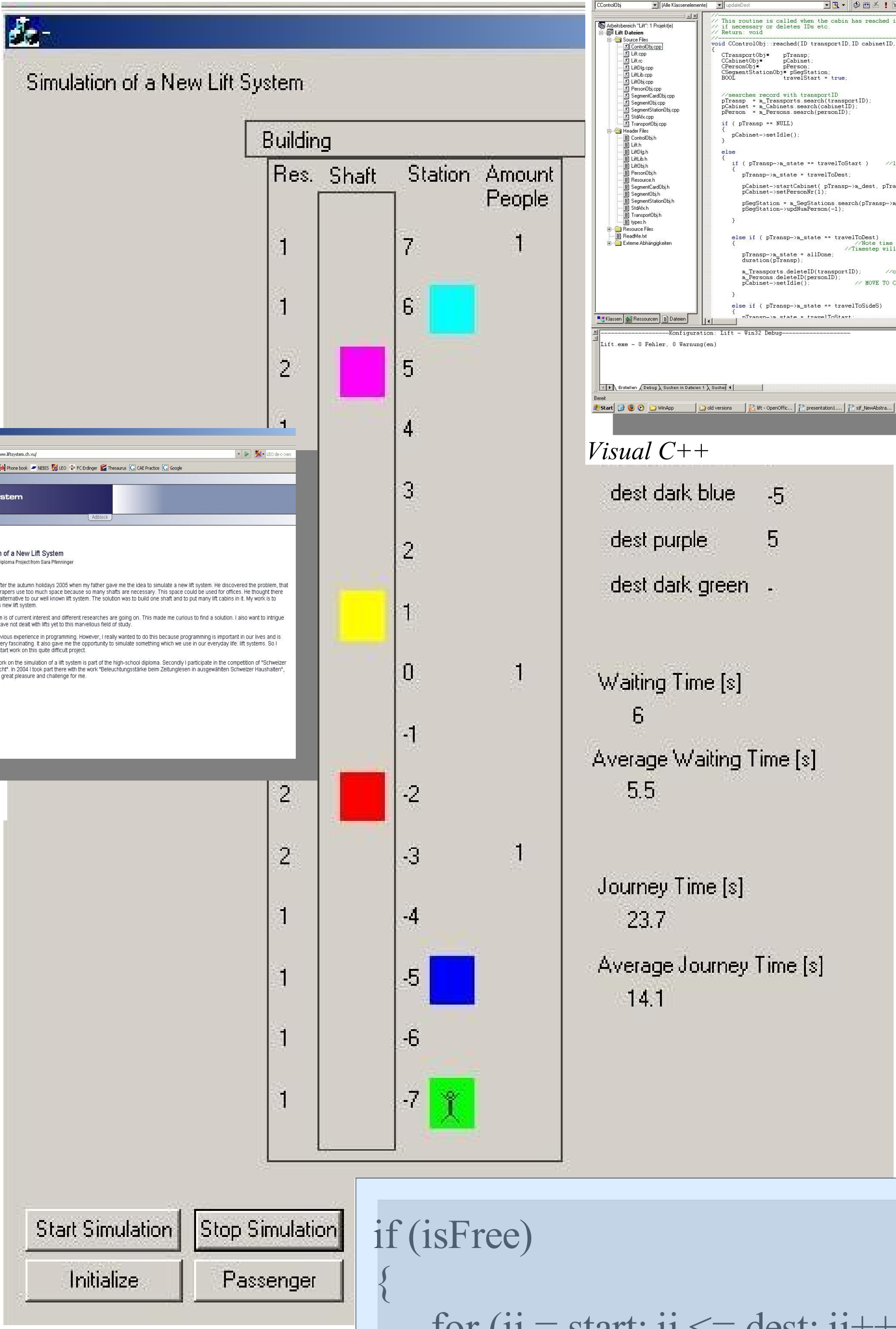
Simulation of a New Lift System



```
//------------------------------------------------------------------------------------------
// This routine is called when the cabin has reached its destination. It starts the cabinet again,
// if necessary or deletes IDs etc.
// Return: void
//------------------------------------------------------------------------------------------
void CControlObj::reached(ID transportID,ID cabinetID, ID personID)
{
   CTransportObj*    pTransp;
   CCabinetObj*      pCabinet;
   CPersonObj*       pPerson;
   CSegmentStationObj* pSegStation;
   BOOL              travelStart = true;


   //searches record with transportID
   pTransp  = m_Transports.search(transportID);
   pCabinet = m_Cabinets.search(cabinetID);
   pPerson  = m_Persons.search(personID);

   if ( pTransp == NULL)
   {
      pCabinet->setIdle();
   }

   else
   {
      if ( pTransp->m_state == travelToStart )     //look at state
      {
         pTransp->m_state = travelToDest;

         pCabinet->startCabinet( pTransp->m_dest, pTransp, this, false);
         pCabinet->setPersonNr(1);

         pSegStation = m_SegStations.search(pTransp->m_start);
         pSegStation->updNumPerson(-1);

      }


      else if ( pTransp->m_state == travelToDest)
      {                                  //Note time of timestep: When is timestep called? All 1/100 seconds?
                                      //Timestep will be called from different functions!
         pTransp->m_state = allDone;                                //Add secondscounter each time of timestep to const
         duration(pTransp);

         m_Transports.deleteID(transportID);       //cancel transport task -> see in Cabinet array
         m_Persons.deleteID(personID);
         pCabinet->setIdle();              // MOVE TO CABIN

      }

      else if ( pTransp->m_state == travelToSideS)
      {
```

```
--------------------Konfiguration: Lift - Win32 Debug--------------------

Lift.exe - 0 Fehler, 0 Warnung(en)
```

Visual C++



Simulation of a New Lift System

Building

Res.	Shaft	Station	Amount People
1		7	1
1		6	
2		5	
1		4	
		3	
		2	
		1	
		0	1
		-1	
2		-2	
2		-3	1
1		-4	
1		-5	
1		-6	
1		-7	

dest dark blue	-5
dest purple	5
dest dark green	-

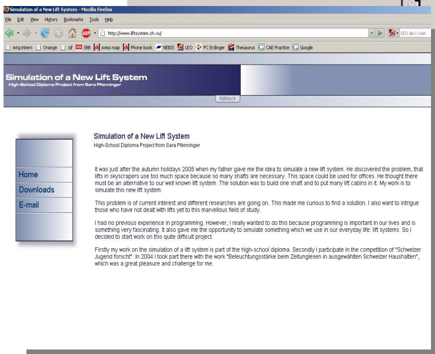
Waiting Time [s]
6

Average Waiting Time [s]
5.5

Journey Time [s]
23.7

Average Journey Time [s]
14.1

Graphical User Interface



Simulation of a New Lift System
High-School Diploma Project from Sara Pfenninger

Simulation of a New Lift System
High-School Diploma Project from Sara Pfenninger

It was just after the autumn holidays 2005 when my father gave me the idea to simulate a new lift system. He discovered the problem, that lifts in skyscrapers use too much space because so many shafts are necessary. This space could be used for offices. He thought there must be an alternative to our well known lift system. The solution was to build one shaft and to put many lift cabins in it. My work is to simulate this new lift system.

This problem is of current interest and different researches are going on. This made me curious to find a solution. I also want to intrigue those who have not dealt with lifts yet to this marvellous field of study.

I had no previous experience in programming. However, I really wanted to do this because programming is important in our lives and is something very fascinating. It also gave me the opportunity to simulate something which we use in our everyday life: lift systems. So I decided to start work on this quite difficult project.

Firstly my work on the simulation of a lift system is part of the high-school diploma. Secondly I participate in the competition of "Schweizer Jugend forscht". In 2004 I took part there with the work "Beleuchtungsstärke beim Zeitunglesen in ausgewählten Schweizer Haushalten", which was a great pleasure and challenge for me.

Webpage

```
if (isFree)
{
      for (ii = start; ii <= dest; ii++)
          {
                  pSegment = search(ii);
                  pSegment->m_state = occupied;
      }
}
```

Programming Code

Sara Pfenninger
Tutor: Mr F. Nlabu
March 2007
MNG Rämibühl, Zürich

Contents

1 Preface

1.1 Reasons for Choosing This Topic

It was just after the autumn holidays 2005 when my father gave me the idea to simulate a new lift system. He discovered the problem that lifts in skyscrapers use too much space because so many shafts are necessary. This space could be used for offices. He thought there must be an alternative to our well known lift system. The solution was to build one shaft and to put many lift cabins in it. My work consists of simulating this new lift system.

This problem is of current interest and different researches are going on. This made me curious to find a solution. I also want to intrigue those who have not dealt with lifts yet to this marvellous field of study.

I had no previous experience in programming. However, I really wanted to do this because programming is important in our lives and is something very fascinating. It also gave me the opportunity to simulate something which we use in our everyday life: lift systems. So I decided to start work on this quite difficult project.

Illustration 1: Old lift in Rome

This old lift in Rome shows that there was a lot of development since the early days of lift construction. By the way: this lift in Rome is still in use. Lift constructors are looking constantly for improvements. The many new high-rise buildings and the search for more comfort lead to an increasing production of lift systems.

Firstly, my work on the simulation of a lift system is part of the high-school diploma. Secondly, I participate in the competition of „Schweizer Jugend forscht“. In 2004 I took part there with the work “Beleuchtungsstärke beim Zeitunglesen in ausgewählten Schweizer Haushalten“, which was a great pleasure and challenge for me.

1.2 Acknowledgements

I would like to thank Mr F. Nlabu who is the supervisor of this project of “Schweizer Jugend forscht”. He works for Schindler and provided me with different information about how lifts work. He mentioned what is good and what could be improved in this project. This motivated me a lot. I wish to thank Mr G. Schäppi who was the supervisor of the high-school diploma project and was always ready to answer questions. I would also like to thank my father, Ernst Pfenninger, who gave me the idea to this work and explained important aspects when programming in Visual C++ as well as Fabian Würtz for putting up with me in stressful times and supporting me with excellent ideas. I also want to say thank you to Jan-Filip Zagalak who studies information technology, read my work and pointed out to what I have to focus on. Thank you, Mr Simon Pfenninger for the introduction of the language Visual C++. Last but not least my acknowledgements are directed to all proofreaders and supporters who did a great job in helping me.

2 Abstract

2.1 Aim

The aim of this work is to test a new lift system by a simulation. It possesses the possibility to vary different parameters. The system has to be optimised, so that the waiting period is as short as possible. However, the use of the software and the optimisation are beyond this work.

2.2 Methods

The simulation is programmed in Visual C++. This is an object-oriented language, which is an important aspect.

A distinguishing feature of this system is that there is more than one cabin in a shaft. The cabins are driven by small electric motors and do not use any traction ropes. The cabins can travel vertically up and down in the same shaft. When the cabins are at rest in order to let people in or out and to wait for the next transport task, they are in the station outside of the shaft. The shaft is then free for the other cabins to travel. There is a mechanical device which makes it possible for the cabins to move horizontally from the shaft to the station. The elaboration of this mechanical device is not within the scope of my present work.

2.3 Results

The simulation is divided into different parts, so-called classes. They interact with each other. By using these different classes it is easy to change the code of the program. In the simulation you can define the lift system with its cabins and the passengers. At the end, the program can calculate the time each passenger needed to reach its target floor. The simulation is visualized so that the user can see the people waiting, entering and leaving the cabin.

2.4 Discussion

There are many advantages of the new lift system. Firstly, the people do not have to change the shaft in high-rise buildings to reach their destination. Secondly, the cabins do not stop while moving with a passenger in it which is a high comfort. Thirdly, there are many cabins travelling at the same time. However, there are also disadvantages. Firstly, lift cabins sometimes pass people waiting in a floor. They have to wait then for a next cabin. Secondly, the amount of passengers in a cabin is at a maximum six and therefore such cabin is quite small. Thirdly, each cabin has to carry its motor.

2.5 Conclusion

The aim of the expected work has been achieved, the lift simulation has been implemented and I could weigh the advantages and disadvantages between the conventional and the new lift system. I won lots of experience in programming an object-oriented language such as Visual C++. Finally the new lift system represents an alternative which should be taken into account for further development of the elevator system.

3 Introduction

Lifts are a very important invention especially for transporting passengers in buildings. Many people could not live without them. They are here to save physical energy and time. Who does not know the situation in front of the lift? You press the button and wait for the sound of the arriving lift. Nothing happens. You are waiting and you get angry. Time is a scarce resource and you do not want to waste it. This is a well known problem; see the following cartoon, illustration 2.

Illustration 2: llustration 2: Cartoon: “I say we take the stairs” (Source: http://selections.rockefeller.edu/cms/images/stories/cartoons/elevator_cartoon_small.png Dec. 2006)

„Elevator systems are the most important transportation method for high-rise buildings. However, in the conventional elevator, only one elevator car, suspended by ropes, occupies the whole elevator shaft. Because of this, the taller the building is, the lower the performance of the elevator becomes. With the background of progress in linear-motor technology and increasing needs for high-performance transportation systems for large-scale buildings, rope less multicar elevators (MCE[1]) that have several cars driven by linear motors in a single elevator shaft, are attracting attention as a novel transportation system. However, considering the control of MCE, we can only apply the knowledge of existing elevator systems to MCE to a limited extent, and there is a need for new controller design methods“[2].

1 MCE: Multicar Elevator
2 Markon, Sandor et al. 2006. Control of Traffic Systems in Buildings. Page 18.

Illustration 3 shows this multi cabin lift system.

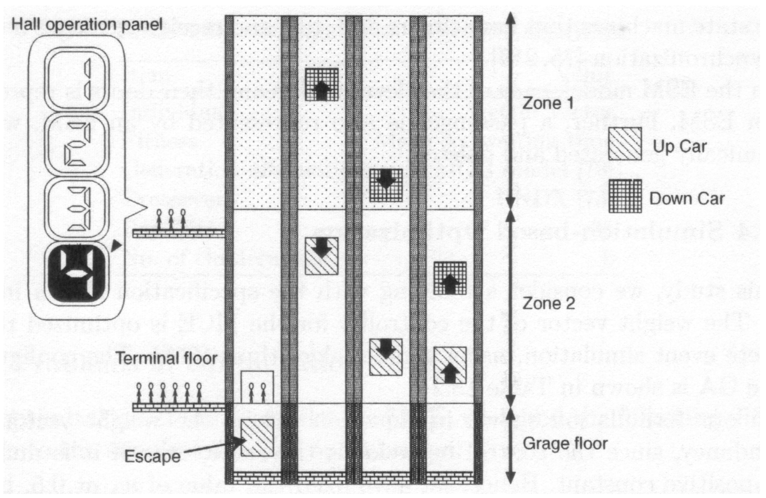


Illustration 3: Multi cabin lift system (MCE). (Source: Markon et al. 2006: 223)

The lift system that I am going to study is a multi cabin system which is similar to the MCE mentioned above. There are small cabins for limited number of passengers who travel directly from the start to the destination. There are no stops during the journey for letting other people in and out.

„The level of service provided by an elevator system may be specified in two ways: the capacity of the system or the average passenger waiting time"[3] This work focuses on both of these issues, because one is the result of the other. If the capacity of a system is low, one result will be long waiting times for passengers.

3.1 Aim

The aim of this work is to test the new lift system with direct travelling and multiple cabins in a shaft. The test is not done in reality, of course, but in the way of a simulation. The simulation gives the possibility to vary the different parameters. The system has to be optimised, so that the waiting period is as short as possible. There are a lot of questions: Is direct travelling enough powerful or are there too many cabins needed? Do the cabins block each other when circulating in the same shaft? Are there many parallel running shafts necessary? The simulation can answer these questions, which are why I write this simulation software. The use of the software and the optimisation is beyond this work.

3.2 Lifts versus Elevators

„An elevator is a transport device used to move goods or people vertically. Outside North America, elevators are known more commonly as lifts"[4].

I will use British English. The following table goes into greater detail and shows the equivalent expressions in British and American English.

3 Gamse, Beryl. A Simulation of an Elevator System for a Moderate Height Building. Page 1.
4 Wikipedia. Elevator. http://en.wikipedia.org/wiki/Elevators (June 2006)

British English	*American English*
lift	elevator
cabin	car
control	controller

Table 1: British versus American English

3.3 The Programming Language: Visual C++

In the past, high level languages, such as BASIC or FORTRAN were used to program a simulation. Nowadays the object oriented languages, such as C++, are the likely choice according to Barney.

The programming language I use is Visual C++. It is based on C. The C language has steadily increased in popularity since it was created in the early 1970s. Since 1990, C++ is one of the most popular commercial languages because it has a good performance. The software development system Visual C++ is a widely used standard package.

I did not know anything about this programming language when I started this project. Therefore my first aim was actually to get used to programming in C. To learn this language I used the book “C for yourself”[5] which is easy to understand.

3.4 Simulation and Computer Aided Design

„Where the traffic design requirements are more complicated, computer simulations may need to be carried out“[6]. It is very difficult to test the new lift system with technical calculations. For this reason, I decided to simulate the lift system. In addition, simulations are used to prove specific cases and not general designs. A simulation is just an approximation of the real conditions. It is the ideal and does not correspond to real cases in every way.

Definition: “Simulation is the development and use of models to aid in the evaluation of ideas and the study of dynamic systems or situations“[6].

Definition: “Computer aided design is where a designer interacts directly with a complex computer process and in so doing closes the design loop.“[7].

The following table 2 on page 9 helps to visualize the computer aided design.

5 Microsoft Corporation. C for yourself.
6 Barney, Gina. Elevator traffic handbook. Page 355
7 Ditto. Page 356

Start	
INPUT	Building data Lift data Passenger data
SIMULATION	Discrete digital simulation Graphical progress display
OUTPUT	Graphical outputs Listings
Finish	

Table 2: Phases in the computer aided design of lift traffic systems[1]

4 Methods

4.1 Lift Systems in General

The building data set comprises the following according to Barney[8]: The performance of a lift system depends on three data sets such as the building, the lift system and the passengers. There are points added which are due to the new lift system.

1. number of floors
2. inter floor distance
3. express jump
4. conversion time from 1 shaft to other[9]
5. conversion time from vertical to horizontal moving and vice-versa

Item 3 above, the express jump, may be nothing more than a few extra metres between the main terminal floor and the first served floor or could be some distance if the first served floor above the lobby is an express zone terminal floor high in the building.

The lift system data set comprises the following:

1. number of cars
2. rated car capacity (load)
3. rated speed
4. flight times between floors
5. door opening times
6. door closing times
7. miscellaneous times (such as car and landing door dwell times, etc.)
8. traffic control system
9. due time because of safety

Item 9 above, corresponds to the avoidance of a collision between two cabins. Two cabins can not cross.

The passenger data set comprises the following:

1. number of passengers boarding from specific floors
2. number of passengers exiting at specific floors
3. traffic mode, i.e.: unidirectional or multi directional, etc.
4. transfer times for passengers entering and leaving cars
5. passenger actions

8 Barney, Gina. Elevator traffic handbook. Pages 103-104
9 This only occurs if there are two shafts, one for moving upwards and one for moving downwards.

Item 5 above, is included to cover passenger's misbehaviour (door holding, excessive operation of push-buttons, etc.)

This work follows this data set. In other words the simulation inputs contain these data. Of course, as it is just a simulation, a few inputs from the data set are neglected, for example *passenger actions*.

The following table helps us to ascertain whether the output, that is to say the time interval, is too long or not.

„Table 3 indicates the percentage and time values for several grades of service over one hour of peak activity in an office building. An hour of peak activity is taken in order to obtain sensible and realisable results. It should be possible to obtain the grades of service indicated in the table during the worst hour of activity [...] [10].

Grade of service	***Percentage of calls answered in***		***Time to answer calls (s)***	
	30 s	***60 s***	***50%***	***90%***
Excellent	>75	>98	<20	<45
Good	>70	>95	<22.5	<50
Fair	>65	>92	<25	<55
Poor/unacceptable	<65	<92	>25	>55

Table 3: Office Building Average System Response Time Performance[10]

4.2 Description of the New Lift System

A distinguishing feature of this system is that there is more than one cabin in one shaft. The cabin is driven by small electric motors and there is no use of traction ropes in the shaft. The cabin is driven by the motor which is attached on it. The cabins can travel vertically up and down in the same shaft. When the cabins are at rest in order to let people in and out and to wait for the next transport task, they are outside the shaft in the station. The shaft is then free for the other cabins to travel. There is a mechanical device which makes it possible as this mechanical device allows the cabins to move horizontally from the shaft to the station. There, the doors of the cabin open and the passengers can exit. Now the cabin is idle and waits till it has been asked to fetch other people. In illustration 4 on the following page you can see the lift cabins as well as the stations. There are small cabins for a limited number of passengers who travel directly, without stopping from the start to the destination as mentioned above.

10 Barney, Gina. Elevator traffic handbook. Page 136.

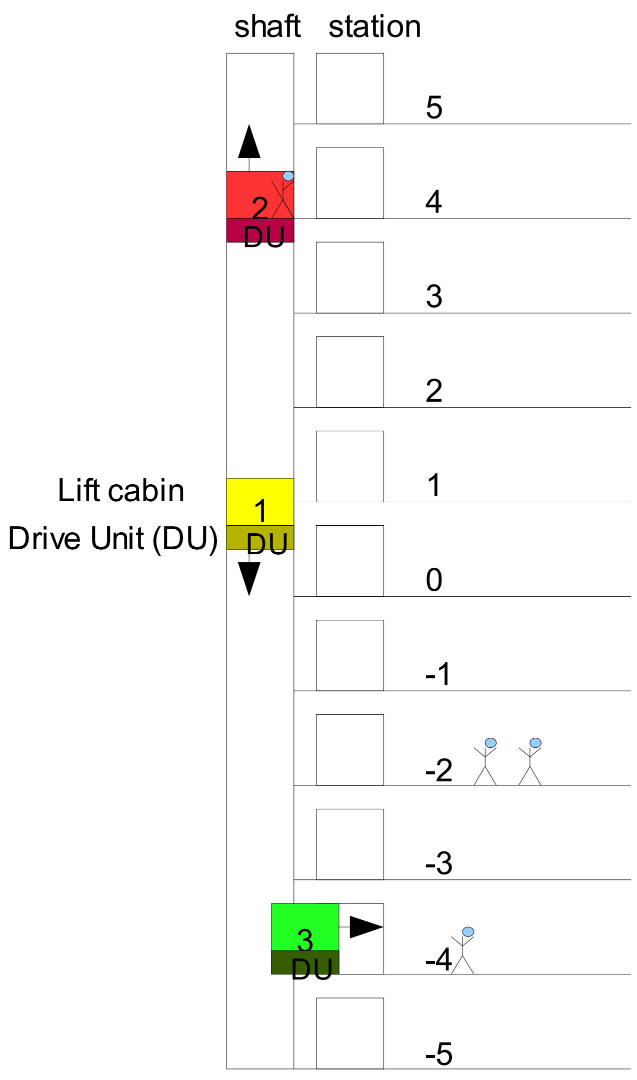


Illustration 4: Overview of the new lift system

This work does not focus on the mechanical design. However, with the basic mechanical features mentioned above, that there are no traction ropes for example, the lift system is possible to realise.

The following concept from Vernon[11] shows a mechanical design, which has to be tested. As the motor runs, the lift cabin needs no ropes to move itself up or down a shaft. All the cabin needs is the electric power, which can be supplied via „trolley“ connections to power cables along the shaft wall. Each cabin motor should actually be a motor-generator, so that when any cabin descends in a shaft, power can be generated to supply some other lift cabins.

The simulated lift system is of the following type: „The simplest form of automatic lift control is the single call automatic control. Single call push-buttons are provided on the landings. This form of control is also termed “non collective” or automatic push-button (APB) control. The passengers directly operate the lift by pressing landing [button on floor] and car buttons [in the cabins], so no attendant is necessary. Car calls are given absolute preference over landing calls, which are only answered if the lift is available“[12].

11 Vernon. Multiple-Cab Elevator Loop. http://www.halfbakery.com/idea/Multiple-Cab_20Elevator_20Loop (November 2006)

12 Barney, Gina. Elevator traffic handbook. Page 136.

4.3 Other Lift Systems

The idea of several cabins in one common shaft was realised in the paternoster system, also known as the cyclic elevator. At one side, several cars were moving upwards without stopping, at the other side, they were travelling downwards. Passengers had to enter and leave it while the cabin was moving which was quite dangerous. This was the reason why they were phased out.

Illustration 5: Paternoster lift system (Source: www.joostdevree.nl Dec.2006)

The double deck cabin, which still exists, has similar aspects. There is a double cabin in the shaft; the cabin is divided in two cabins, one over the other. The double deck cabin is suspended on ropes and has no drive unit. The two parts always drive together and can not be separated. This lift system is of course more efficient than the conventional lift system as the cabin reaches different floors by each stop at a time. In illustrations 6 and 7 above you can see such a lift system.

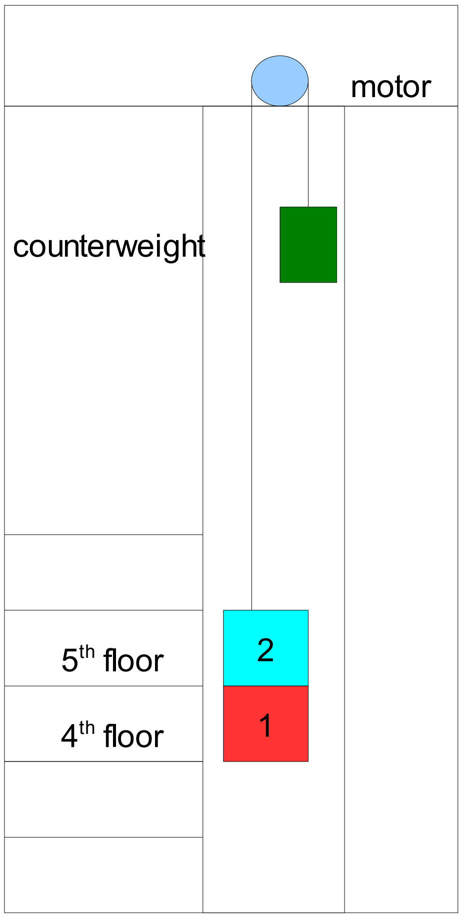


Illustration 6: Double deck cabin

Illustration 7: Double deck cabins

Another lift system is the “circulating multi-car elevator”. On the 1st of March 2006 the Mechanical

Engineering Research Laboratory presented their idea to improve the transport capacity of lifts to reduce waiting time. Many cabins circulate in two shafts, one for travelling upwards and one for travelling downwards. They say that it would become possible to more than double the transport capacity of conventional lifts in high-rise buildings with no lack in comfort. There is this basic principle in illustration 8 below.

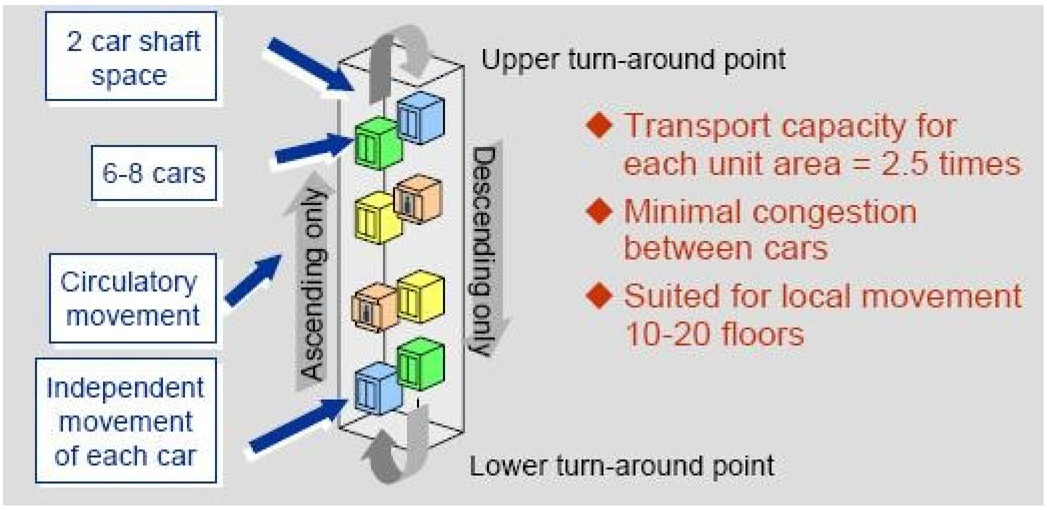

2 car shaft space
6-8 cars
Circulatory movement
Independent movement of each car
Ascending only
Descending only
Upper turn-around point
Lower turn-around point
Transport capacity for each unit area = 2.5 times
Minimal congestion between cars
Suited for local movement 10-20 floors

Illustration 8: Basic principle of the circulating multi-car lift (Source: www.hqrd.hitachi.co.jp/global/news_pdf_e/merl060301nrde_elevator.pdf Nov.06)

The lift system that is simulated within my work is much different: The cabins can overtake each other, because they rest in the station which is found on each floor. Therefore the cabins move freely similar to cars on the motorway.

4.4 Mechanical Design

Sensors monitor the location of the cabins in the improved system (“circulating multi-car elevator”). There are also sensors in this new lift system and send messages to the supervision control regarding where each cabin is exactly located in the shaft.

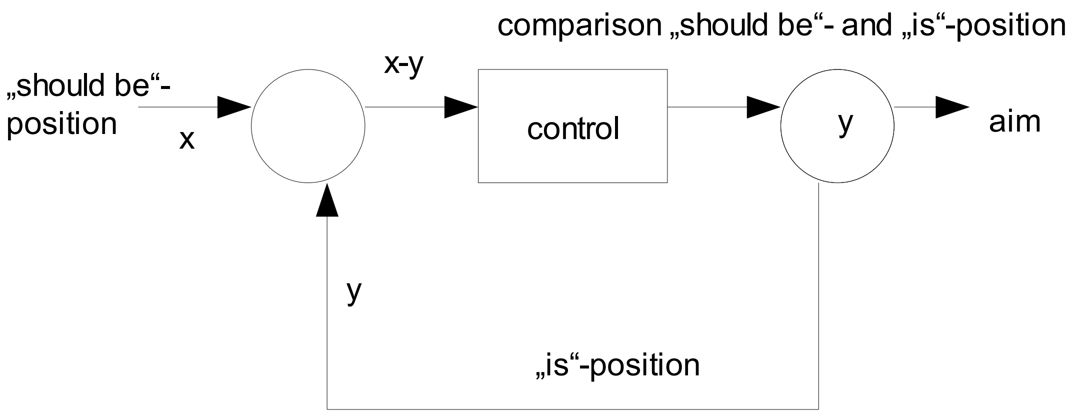


Illustration 9: Connection mechanism and simulation

Illustration 9 above illustrates the connection between the mechanism and the simulation. The sensors measure the „is“-position, labelled by *y*, that is to say the position of the cabin at a certain time. The „should be“-position, labelled by x, is the position where the cabin should be according to the calculations of the computer, the simulation. The difference between these two values should be zero. If this is not the case either the position of the cabin, the *y* variable, or the calculated position, the *x* variable in the computer, has to be adjusted to zero by the lift controller, which is the simulating computer in my case. The cabin must therefore move a little bit or the computer has to change position *x* depending on the situation. As soon as the difference of these two values is zero, the condition is fulfilled and the target position of the cabin is reached. Innumerable other steps follow in the same way. This method is very important for the accuracy of the lift positioning system. When a cabin has reached its destination the difference between the ground of the cabin and the floor (ground of the house) should not exceed 3 mm otherwise it is very dangerous for old people or people with a trolley or a baby carriage. The reason for this is that nobody will pay attention to a difference between the grounds as everybody is used to the fact, that there is normally no danger.

Illustration 10 on page 15 shows the sensors of the cabins. Of course they are not drawn in the right ratio to the rest.

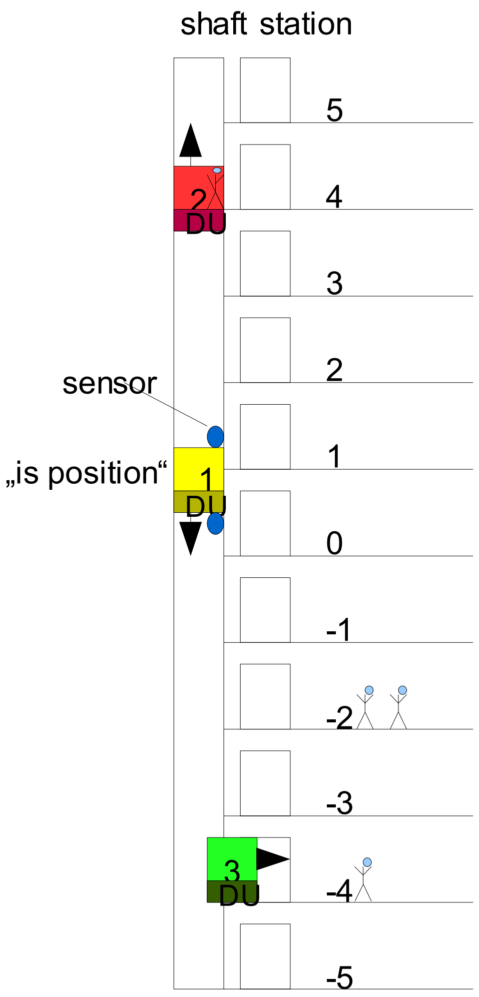


Illustration 10: Mechanical design: Sensors

The transition time t_{trans} is the time which a passenger needs to reach its destination. Firstly, when a passenger has pressed the button, there is the waiting time t_w for cabin to arrive. Secondly, the cabin needs the time t_v to change from vertical to horizontal movement. Thirdly, there is the travelling time $t_{travelling}$. This is illustrated in illustration 11 below.

Formula:

$t_{trans} = t_w + t_v + t_{travelling}$

Other time periods such as the passenger transfer time or door operating time are negligible.

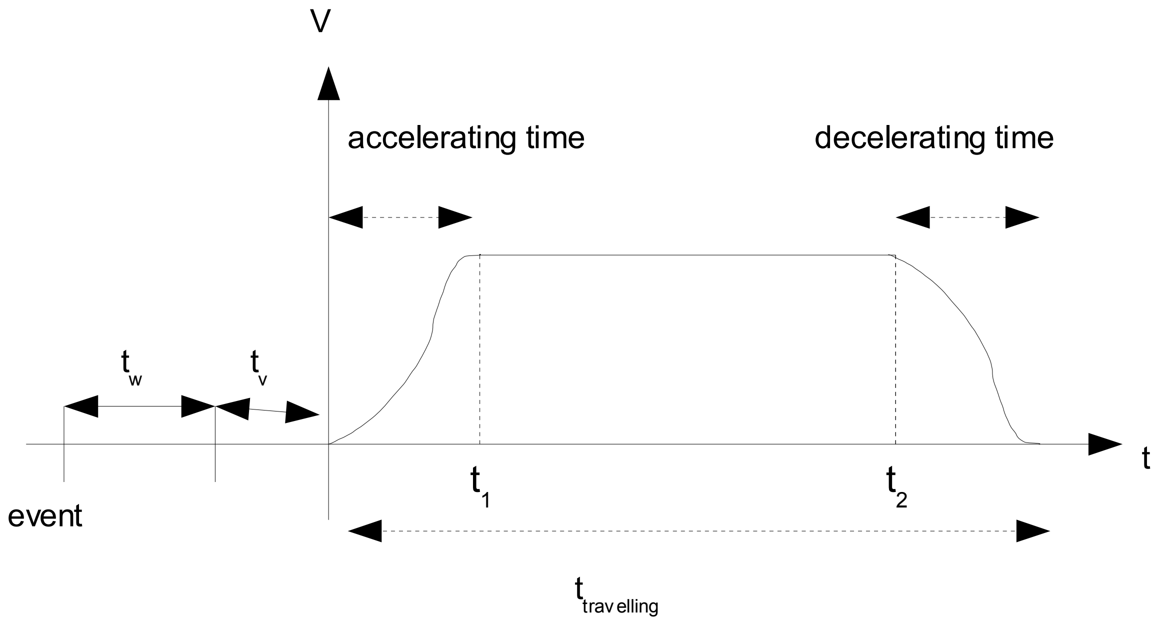


Illustration 11: Overview of the time periods

4.5 The Program

4.5.1 Overview

The idea of the system is comparable to a motorway. The normal shaft would be the motorway. Cabins which want to move in the normal shaft are cars. The control is the car driver who wants to drive on the motorway and has to check that the way leading to its destination is free.

The new lift simulation program is also comparable with a taxi service. Each person must be treated as an individual. Only one person can enter a lift cabin in the simulation. Therefore the cabins represent different taxis. The computer control would then be the central taxi service station.

4.5.2 The Simulation

The simulation software is written with the intent to vary the number of cabins, passengers and floors. The programming language Visual C++ allows a structured programming where it is possible to vary the number of classes, e.g. the number of shafts. In the simulation there is only one shaft with cabins in it. The cabins can travel in both directions. No special events such as fire alarms or defective cabin will happen.

The simulation is divided into different parts, called classes, which you can see in table 4 below. These classes will be explained in more detail in following chapter 5.

Name/Description	***Class Name in my Visual C++ Program***
Control	ControlObj
Cabin	CabinetObj
Cabins	CabinetArr
Passenger	PersonObj
Passengers	PersonArr
Segment card	SegmentCardObj
Segment cards	SegmentCardArr
Segment	SegmentObj
Segments	SegmentArr
Station segment	SegmentStationObj
Station segments	SegmentStationArr
Transport card	TransportObj
Transport cards	TransportArr
Library	LiftLib
Definitions of types and variables	types

Table 4: Overview of the objects

It is very important that any simulator's user can change the programming code easily in the basic simulation to what I focused on. By using different classes, the whole simulation gets much easier. Then, it is determined that you have an overview as there is a structure in it and every user is able to see the different classes very clearly. Otherwise I would have a very complicated and long program at the end, whereas my aim is to have one program, which is as simple as possible.

For the simulation the following assumptions are made:

- the traffic profile is ideal
- all floors are equally populated
- various lost times, such as passenger disturbance, etc. are negligible

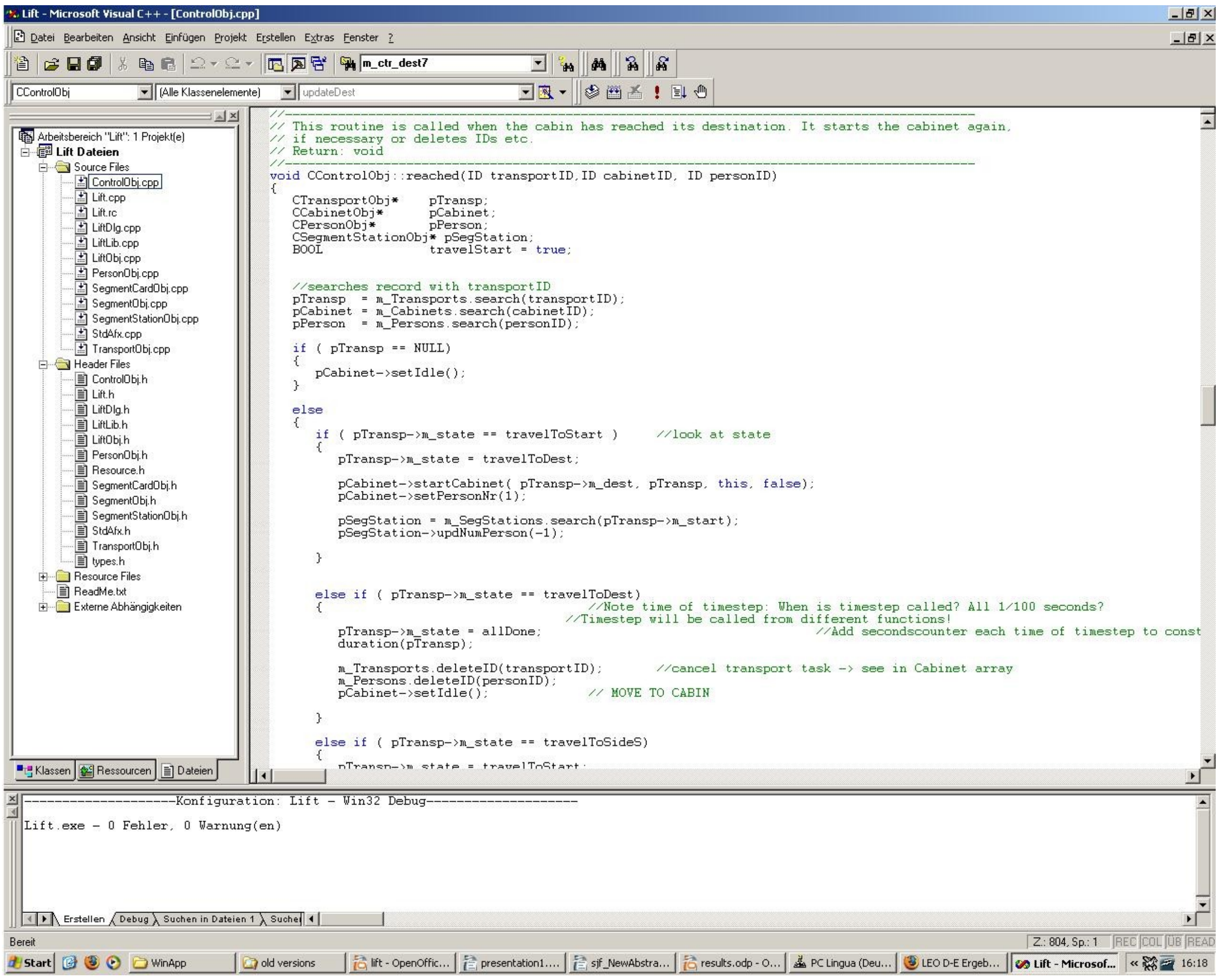
5 Results

5.1 The Program

Remark: All examples of programming codes written in italic in this report are simplified because of the readability. They should show the code of the lift simulation in a way which is easy to understand.

5.1.1 Technical Data

To program the simulation, the programming language C++ is used, as mentioned above. The programming environment is Visual C++. Have a look at the screenshot of the programming environment in illustration 12 . The operating systems are Windows as well as Linux.



```
//------------------------------------------------------------------------------------------
// This routine is called when the cabin has reached its destination. It starts the cabinet again,
// if necessary or deletes IDs etc.
// Return: void
//------------------------------------------------------------------------------------------
void CControlObj::reached(ID transportID,ID cabinetID, ID personID)
{
   CTransportObj*     pTransp;
   CCabinetObj*       pCabinet;
   CPersonObj*        pPerson;
   CSegmentStationObj* pSegStation;
   BOOL               travelStart = true;


   //searches record with transportID
   pTransp  = m_Transports.search(transportID);
   pCabinet = m_Cabinets.search(cabinetID);
   pPerson  = m_Persons.search(personID);

   if ( pTransp == NULL)
   {
      pCabinet->setIdle();
   }

   else
   {
      if ( pTransp->m_state == travelToStart )     //look at state
      {
         pTransp->m_state = travelToDest;

         pCabinet->startCabinet( pTransp->m_dest, pTransp, this, false);
         pCabinet->setPersonNr(1);

         pSegStation = m_SegStations.search(pTransp->m_start);
         pSegStation->updNumPerson(-1);

      }


      else if ( pTransp->m_state == travelToDest)
      {                                     //Note time of timestep: When is timestep called? All 1/100 seconds?
                                         //Timestep will be called from different functions!
         pTransp->m_state = allDone;                                    //Add secondscounter each time of timestep to const
         duration(pTransp);

         m_Transports.deleteID(transportID);       //cancel transport task -> see in Cabinet array
         m_Persons.deleteID(personID);
         pCabinet->setIdle();            // MOVE TO CABIN

      }

      else if ( pTransp->m_state == travelToSideS)
      {
         pTransp->m_state = travelToStart;
```

```
--------------------Konfiguration: Lift - Win32 Debug--------------------
Lift.exe - 0 Fehler, 0 Warnung(en)
```

Illustration 12: Screenshot of Visual C++

5.1.2 Overview of Objects

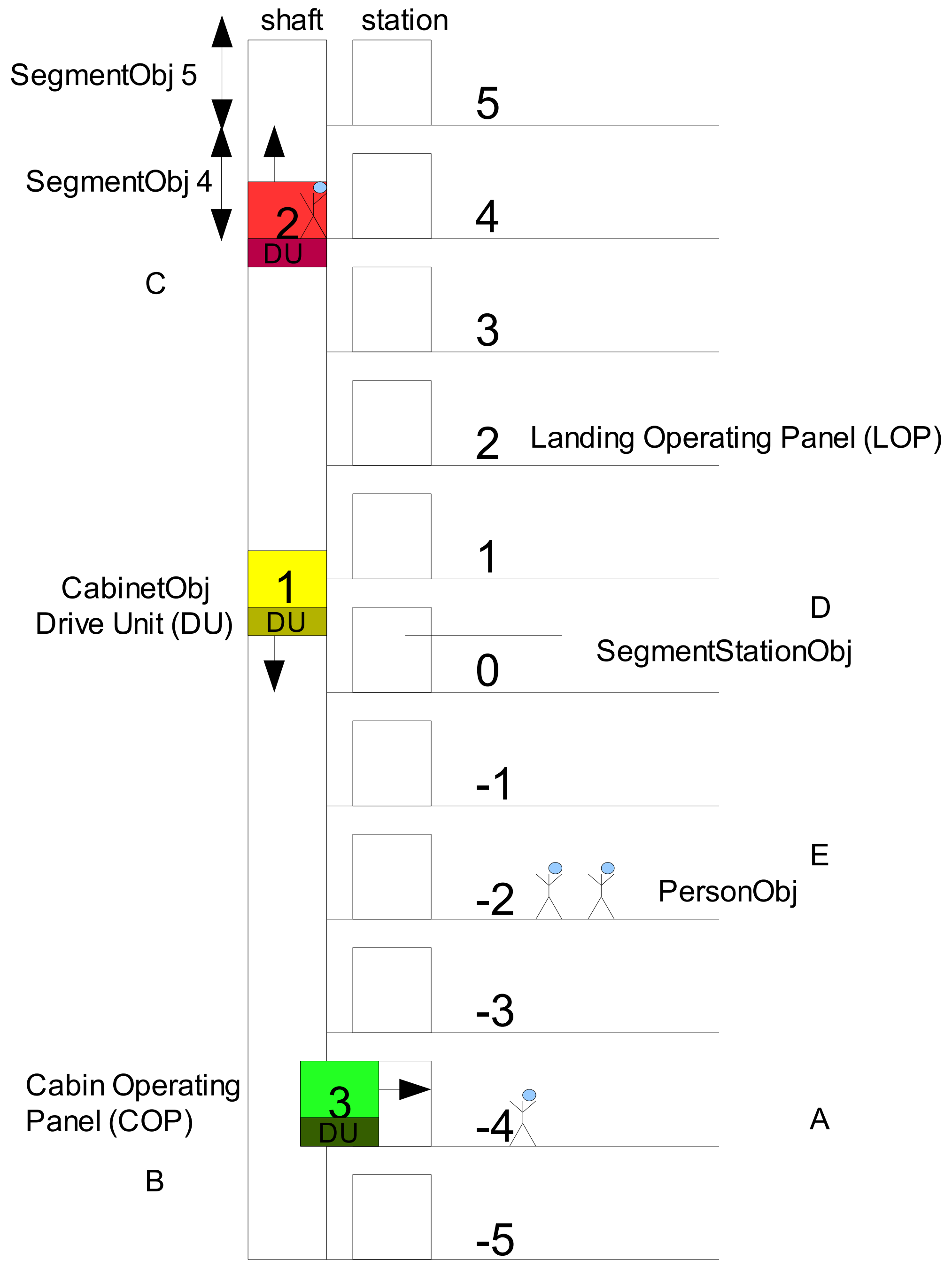


Illustration 13: Overview of objects

In illustration 13 there is the new lift system with part of the objects shown. The numbers from -5 to 5 are the floor numbers.
The following summary referring to illustration 13 illustrates the sequence of the different actions. It is very simplified as it should provide just an overview of the program.
By the event A, a passenger presses the floor button with the arrow pointing upwards as this passenger wants to go to the fifth floor. There is a Landing Operating Panel, called LOP, which interacts with the elevator Group Control. This sends a cabin to floor - 4 as fast as possible so that the passenger can enter the cabin. The Group Control has focuses on the avoidance of collisions

between the cabins and therefore different segments have to be reserved as they are shown in the illustration. This happens in correspondence with the Cabin Control. In event B, you can see the Cabin Operating Panel, called COP. The passenger can press in the cabin the button corresponding to the floor it wants to go. In this case the passenger presses button 5. Then the Cabin Control is responsible for the reservation of the segments as well as the station segment and then the cabin can move. In event C, you can see cabin 2 travelling with a passenger in it to the fifth floor. In event D, a station segment is drawn. The passenger has to wait until the cabin is in the station segment and then he can leave the cabin.
In event E, you can see more passengers waiting for a cabin. As soon as possible there will be one in floor -2. However only one passenger can enter the cabin and the other passengers have to wait for the next one.

5.1.3 Overview of the Program

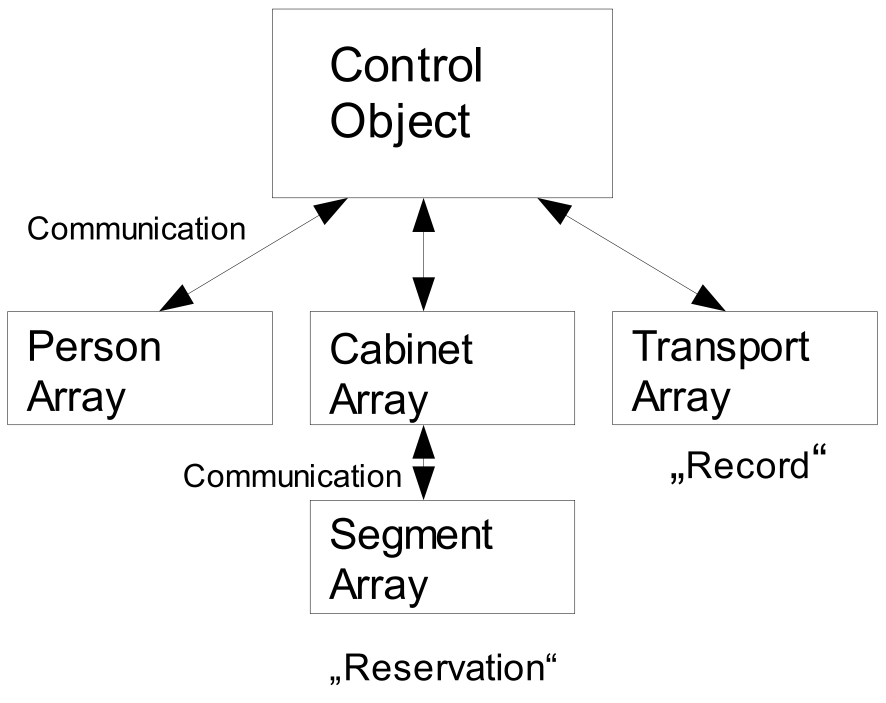


Illustration 14: Overview of most the important objects

This overview shows the most important objects and how they interact with each other. The main object is the control which stays in contact with the other objects. All arrows stay for communication. Firstly, the control communicates with the array of the passengers. A person will be created. Secondly, the control creates a record, called transport card in the array of the transport cards. There it writes the desire of the person, that is to say its actual position and target position. Thirdly the control takes the transport card with the oldest ID and looks for a cabin in the array of the cabins to send it to the person. Afterwards the control does not directly communicate with the array of the segments. However, the cabin is clever because it has an own cabin control and calls the array of the segments to reserve a segment so that there will not be a collision between two cabins. As soon as the cabin reaches its destination, it sends a message to the control which will delete the transport card as well as the data that it no longer needs. In the next paragraphs all objects will be explained in more detail in alphabetical order.

5.1.4 Graphical User Interface

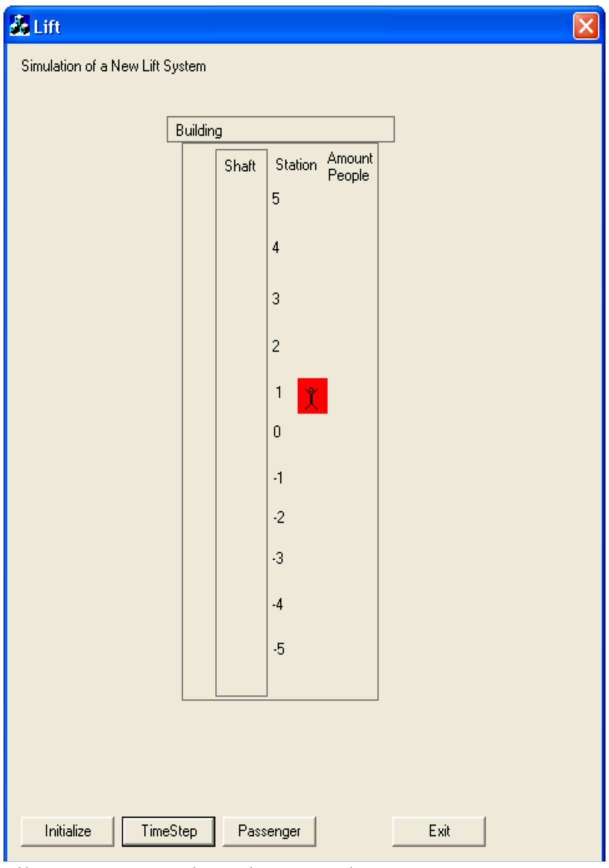


Illustration 15: The cabin is in the station segment

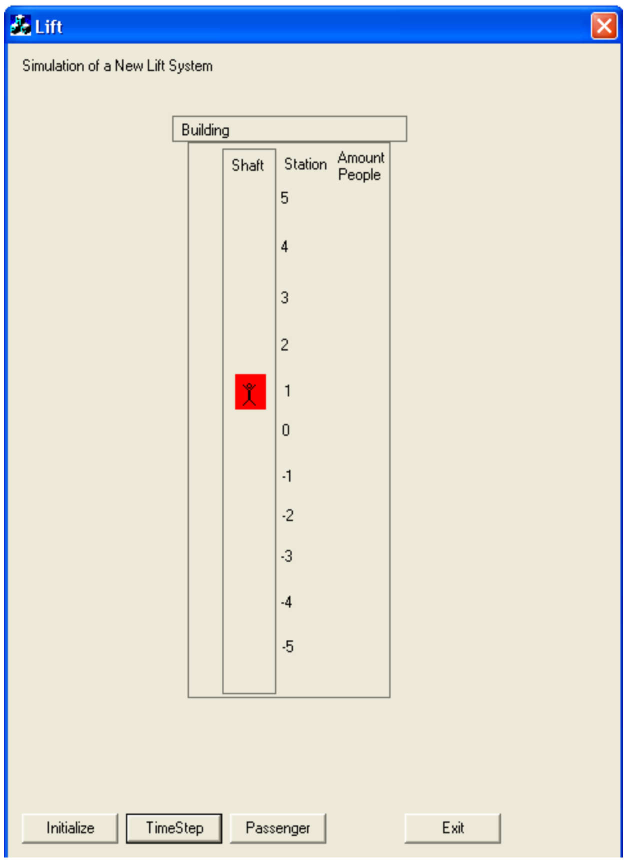


Illustration 16: The cabin is in the shaft

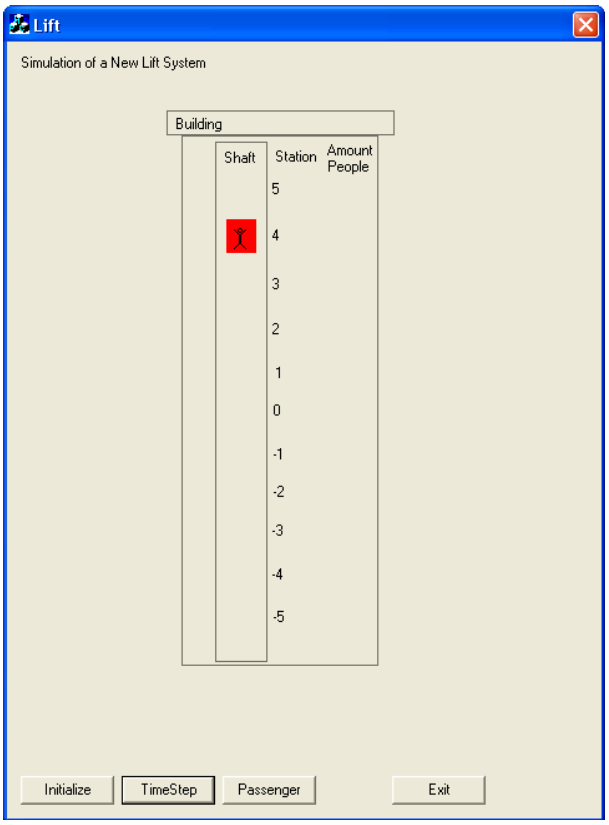


Illustration 17: The cabin is moving in the shaft.

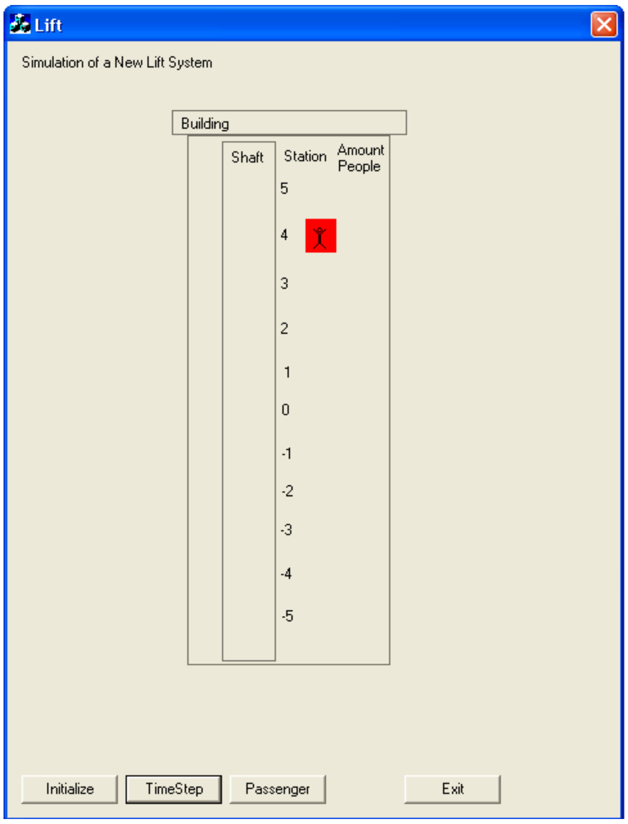


Illustration 18: The cabin has reached its destination and is in the station segment

Illustrations 15, 16, 17 and 18 illustrate the program. Of course there would be much more illustrations on the screen till the cabin arrives at the destination. These four illustrations above show just the way of one cabin, the rest is omitted so that one can clearly follow the way of the red cabin.

During the simulation the graphical user interface can look like illustration 19 on page 23.

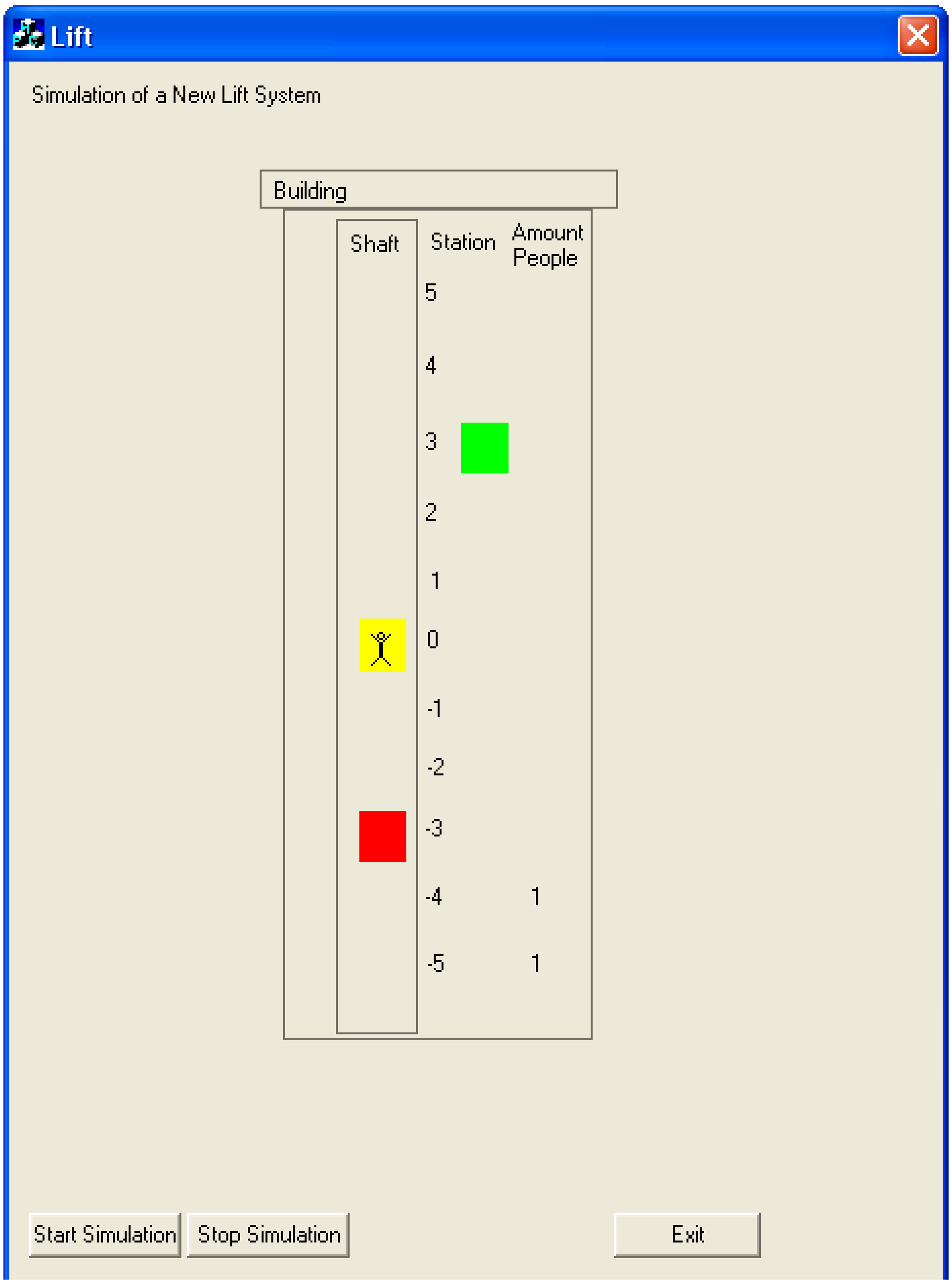


Illustration 19: Graphical user interface

5.1.5 The Control

The control is actually the object which calls almost every function, controls and supervises everything. It is the „main office“. The aim of the control is to look for the passengers and satisfy their needs with regard to the safety issue. Such a design is the best one in the view of the information technology because it is easier to make corrections just in one class namely the control than to search the mistake in all different objects.

The following example is simplified. It illustrates the main idea of the control. Of course there are many other different cases to which this project had focused on.

Somebody wants to go from the fourth to the seventeenth floor. The control sends an idle cabin to the fourth floor. When the cabin arrives, it opens the doors and the passenger can enter it. Then the control sends the cabin from the fourth to the seventeenth floor. The task is fulfilled.

In the following paragraph will be explain this example in more detail.

The control is responsible for the starting process where everything is initialized and the cabins are created as well as the different segments and the station segments. The segment as well as the station segment will be explained in more detail below. The high-rise building with its lift system is created. When the control gets a task from a person, this will be written on a record called the transport card. An ID is given to each record. The control takes this record card and asks for an idle cabin. So it sends a message to the array of the cabins. The array will look for a cabin. If no cabin is idle the record will be put in a queue. If it has found one, a segment card will be created from the cabin. The control will take the next segment card and ask the array of the segments for the corresponding segments as well as the array of the station segments for the according station. If at least one segment is reserved, the control must wait for the next timeStep[13]. If the segments are free, but the station is not, the control asks the array of the cabins which cabin is there. If the cabin is idle it will be sent away. Otherwise we have to wait to the next timeStep. If the segments as well as the station are free, the cabin is set to move.

Then the timeStep function of the array of the cabin class is called and time passes. At the end the control gets a message that everything has been fulfilled that is to say the cabin has reached its destination. Therefore the transport card, all the reserved segments and the person will be cancelled and the cabin becomes idle again. The control notes the time when the person pushed the button and when it left the cabin. Therefore the control can calculate the time, the person needed to reach its destination.

5.1.6 The Cabin and the Array of the Cabins

The cabin is very smart and prevents the control from having to do everything. It executes the different tasks from the control. After the task has been fulfilled, it sends a message to the control about the fulfilled process.

13 The timeStep will be explained in chapter 5.5.3 on page 31.

The following variables define the cabin. Each cabin has an ID. The position of the cabin is exactly defined. The distance from one floor to the next has the unit 1. Therefore a cabin in illustration 20 stays on the position 5.3. It is between the fifth and the sixth floor, but nearer to the fifth one.

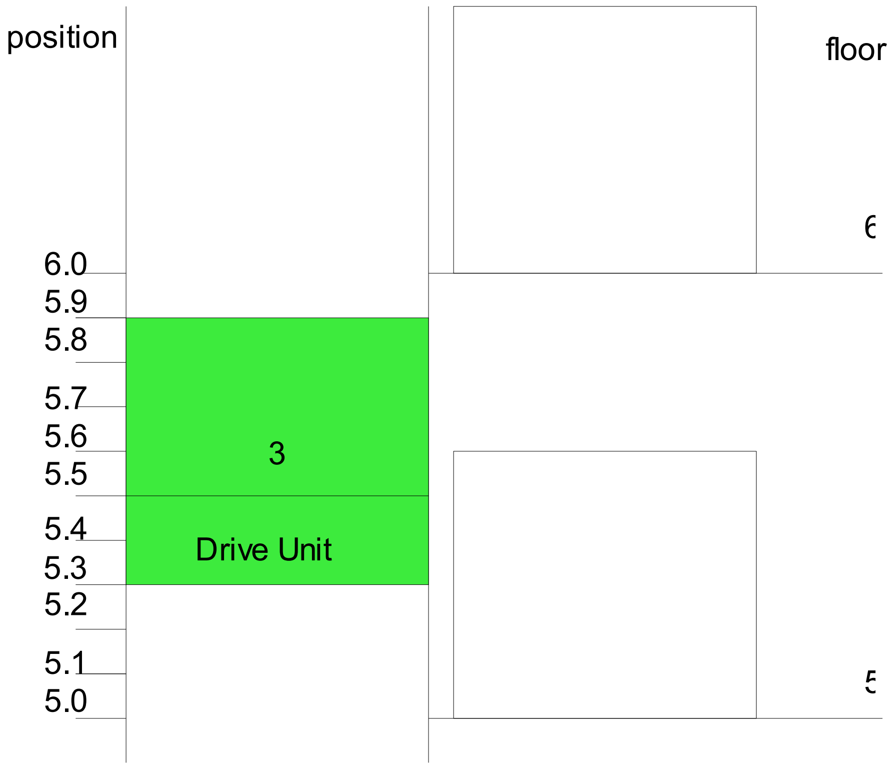


Illustration 20: Position of a cabin

Each cabin has a state which can be *idle*, *occupied*, *moving*, *already arrived* or *arrived*. If it is *idle* it can be *occupied* sooner or later by a passenger. It is *already arrived* if it is already at the destination floor without moving. If a cabin reaches its destination, it is set as *arrived*. These states must be well distinguished.

When the control asks for an idle cabin, the array of the cabins looks for one by asking each cabin if it is *idle* or not. As soon as it has found one, namely the nearest oldest cabin, it sends a message to the control. Afterwards the control calls the starting process of the cabin. It wants to move now but it must reserve the segments before. Therefore the cabin sends a message with all desired parameters to the array of the segment cards which creates a segment card.

Later, the control wants to move the cabin. Therefore the cabin state is set as *moving* or *already arrived* and a corresponding speed is allocated to it.

As the timeStep-function of the array of the cabin class has been called, time passes and each cabin moves. When one has reached its destination, it sends a message to the control. The state will be set as *arrived* or *idle* depending on the situation. If it has just reached the position, where the person waits to enter, the cabin is only set as *arrived*. If it has reached the destination the person wanted to get to, the cabin is set to *idle*.

5.1.7 The Person and the Array of the People

By using a random principle[14] passengers are created. One can define how many passengers should be created in one hour on average. However, it is difficult to estimate the amount of passengers because the distribution and size of the population of a high-rise building changes regularly. So the amount of the passengers created per hour should just be realistic. The passenger's position and destination is determined by random.

It is assumed that every floor is equally populated and that the upwards traffic is more or less equal to the downwards traffic. In reality it is probable that more people want to go upwards using the lift and downwards by using the stairs. This will be pointed out in more detail in the discussion part.

Each person has an ID (identification). As each one contributes a part to the statistic, it is important, that the time is noted when a passenger presses the button and when it leaves the cabin. Therefore one can see how long each passenger has needed to reach its destination. After each task, the person objects are cancelled.

To test a special case it is possible to determine the position and the destination of each person one wants to create. However this function is just for testing the lift system and looking at different cases.

5.1.8 The Segment Card and the Array of the Segment Cards

A segment card is a card created for the cabin. The starting position and also the destination are noted on this segment card. A cabin can only move if there is a segment card. There is an ID to each segment card so that for example one can see which is the oldest. Each segment also has a state which can either be *queue*, *executing* or *finished*. *Queue* means that it is waiting to be read and fulfilled. The segment card is *executing* when the cabin is executing the task and it is *finished* when the task on it is fulfilled and it can be cancelled.

The control asks the array of the segment cards to give the oldest segment card. Thus it follows more or less the rule: „First come, first served“. However, there is a special type of the segment card, called the *priority* segment. They are given to the control even though it is not the oldest. They are more important because an *idle* cabin must give a station segment *free,* so that another cabin, which is *occupied,* can reserve this station for itself.

A segment card is created when it gets a message from the cabin to create one. The state is set as *queue*. As soon as the control has asked for the next segment card, and the corresponding segments are free as well as the station segment of the destination, the state of the segment card is *executing*. When the cabin has reached the destination the state of the segment card is set as *finished*.

You can imagine the segment cards as cards in a record like in illustration 21.

Illustration 21: Record (Source: www.valoony.ch Nov. 2006)

5.1.9 The Segment and the Array of the Segments

The shaft contains different segments attached to each other. This can be seen in illustration 13 on page 19. If a cabin wants to drive to its destination, the according segments must be reserved to avoid a collision. For each floor there is one segment. The height of the building (in floors) minus the lowest floor (for example -3) gives the amount of floors and therefore the amount of the segments.

14 This random principle is explained in chapter 5.5.4 on page 32.

The array of the segments gives an ID to every segment which is created. Each one has a state, which can either be *SegFree* or *occupied*. The *SegFree* means no cabin has yet reserved this segment and it is free whereas the *occupied* means the segment is reserved.

A cabin can not move until every segment, which will be used, is reserved. When the reservation is fulfilled, the cabin can enter the shaft and travel to its destination. As soon as it has travelled one floor, the segment of the floor it has just travelled through will be set as *SegFree* and another cabin can reserve it. As soon as the whole task from the cabin is fulfilled the card will be erased and all segments must be *SegFree* again.

The following code in table 5 is part of the program. It shows how the segments are reserved. *isFree* is a BOOL variable which can either be true or false. It is true when all segments, the cabin needs are free. Then it sets the state of them as *occupied*, see line 6. Illustration 22 shows a possible situation of a collision, if this problem would not have been solved.

```
if ( isFree )
{
    for (ii = start; ii <= dest; ii++)
    {
        pSegment = search(ii);
        pSegment->m_state = occupied;
    }
}
```

Table 5: Code in Visual C++

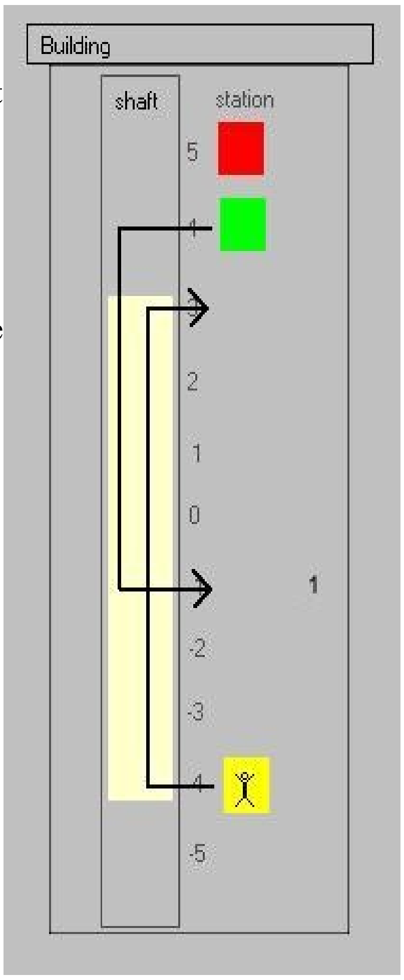


Illustration 22: Avoiding a crash

5.1.10 The Station Segment and the Array of the Station Segments

The station segment is very similar to the segment. There is a station segment on each floor see illustration 13 on page 19. A person can enter or leave the cabin only at the station, called station segment.

If a cabin wants to travel to its destination, the station segment at the destination must be reserved so that the avoidance of collisions is ensured. The array of the station segments gives an ID to every station segment which is created. The state of every station segment determines whether it is has the state *free*, *occupied* or a third state, the so called *ststNot*. If it is only *occupied,* a cabin can remove the cabin which is in the desired station segment. If the state is *ststNot,* a cabin can not remove the cabin which is inside the desired station segment because it is occupied by a reserved cabin which has a passenger in it.

5.1.11 The Transport Card and the Array of the Transport Cards

A transport card is a record. It is very similar to the segment card. When a person is created, the control constructs a transport card and attaches an ID to it. They will be put into a queue. The control asks the array of the transport cards for a new transport card. The time will be written on it as this corresponds to a person pressing the button. The motto „first come, first served“ or „first in, first out“, as the array of the segment cards uses, is applied here, so the array of the transport cards takes the oldest one. If no cabin is idle right now the transport card has to wait for the next timeStep. If it has found an idle cabin, the starting process of it will be called and a segment card is created. When the cabin has reached the destination the person wanted to go to, the arrival time is noted and the transport card will be deleted. The array of the transport cards is equal to a filing box where all records are in a queue.

5.1.12 Library

This class is a set of general functions which are often used by several client-classes, and instead of duplicating them, they just use this library class.

5.1.13 Types

In this file there are many definitions. The names of the basic data types are changed that they get a uniform labelling as table 6 shows.

Type Name	***New Name***	***Description***	***Range***
char	sInt8	signed integer 8 bit	-128 to 127
unsigned char	uInt8	unsigned integer 8 bit	0 to 255
signed int	sInt16	signed integer 16 bit	-32768 to 32767
unsigned int	uInt16	unsigned integer 16 bit	0 to 65535
signed long	sInt32	signed integer 32 bit	-2147483648 to 2147483647
unsigned long	uInt32	unsigned integer 32 bit	0 to 4294967295
float	float7	7-digit precision	$\sim 1.2 \cdot 10^{-38}$ to $3.4 \cdot 10^{38}$
double	float15	15-digit precision	$\sim 2.5 \cdot 10^{-308}$ to $1.8 \cdot 10^{308}$
double	float19	19-digit precision	$\sim 3.4 \cdot 10^{-4932}$ to $1.2 \cdot 10^{4932}$

Table 6: New labelling of basic data types

There are different variables which are defined so that it is much easier to vary these parameters in one file, for example the amount of the cabins or the height of the building. An ID has in all objects the same type. It becomes much easier to change this type just by changing it in one file namely in the types file. So all IDs are from the type *tpID,* which is defined here.

5.1.14 Connection

Order	***Function***	***Class***
A	allTimeSteps(deltaTime)	CControlObj
B	timeStep(deltaTime)	CControlObj
C	controlPersons()	CControlObj
D	generatePerson()	CControlObj
E	AddPerson()	CPersonArr
E	newTransport(x)	CControlObj
F	AddTransport(x)	CTransportArr
G	newID()	CTransportArr
C	takeTransport()	CControlObj
D	searchTransport()	CTransportArr
E	search(x)	CTransportArr
D	reserveCabin(x)	CControlObj
E	lookForCabinet()	CCabinetArr
E	startCabinet(x)	CCabinetObj
F	AddSegmentCard(x)	CSegmentCardArr
G	newID()	CSegmentCardArr
F	search(x)	CSegmentCardArr
C	*depending on the situation:* repeat takeTransport	CControlObj
C	takeSegCard()	CControlObj
D	searchSegmentCard()	CSegmentCardArr
D	lookForSegments(x)	CSegmenArr
D	moveCabinet(x)	CCabinetObj
C	*depending on the situation:* repeat takeSegCard()	CControlObj
B	startTimeStep(x)	CCabinetArr
C	timeStep(x)	CCabinetObj
...	...	...

Table 7: Overview of functions

This overview shows the beginning of the call hierarchy of the function timeStep called by pressing timeStep button. It gives one an idea how the program connects the different objects. In table 7 one can see on the right hand side the different classes. This table shows just one possible flow. „x“ stands for different variables. The order shows where the function is called. For example function E is called from function D and function D is called from the function C or B or A. So A is the original function, and B-... G are part of this function.

5.2 Graphical User Interface

5.3 Testing

There are four buttons in the graphical user interface for testing the simulation.

By pressing the *Initialize* button one can initialize the lift system which must be done before starting the simulation. The time is set to zero and the program will create a defined amount of cabins. One can determine this amount before running the program.

By pressing the *Passenger* button one can „create“ a person. This button is just for testing the system as the person will be created in the timeStep randomly. By pressing this button different events happen such as creating a new transport card and other parts in connection with a person.

By pressing the *TimeStep* button the time passes and different events are called. By pressing directly the *TimeStep* button without the *Passenger* one, the passengers will be created by a random principle.

The output, that is to say the duration of the travel, can be seen in the program. There is no special graphical user interface yet.

By pressing the last button called *Exit* one can leave the simulation.

5.4 Simulating

The graphical user interface for simulating has three buttons. One can start the simulation by pressing the button *Start Simulation* and stop it by pressing the button *Stop Simulation*. For leaving the program one has to click the *Exit* button.

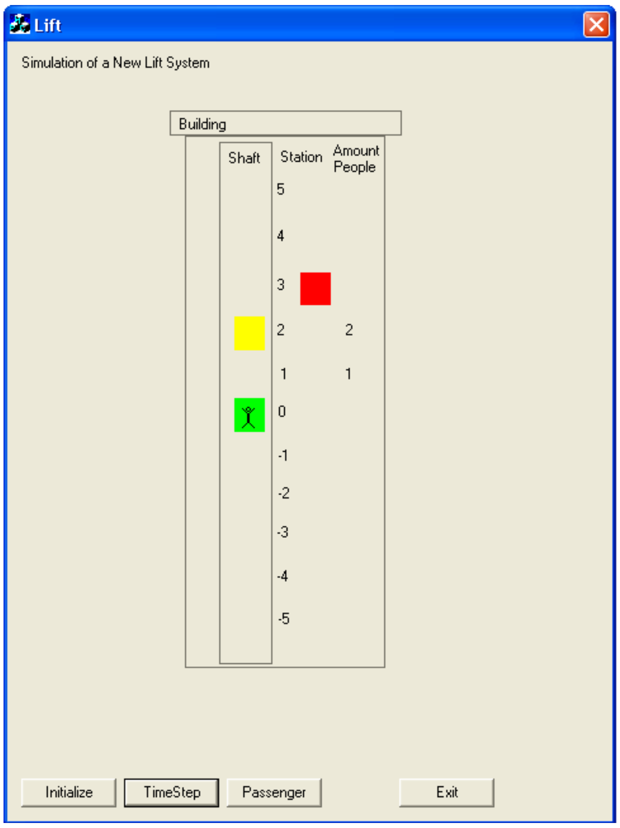


Illustration 23: Graphical user interface: Testing

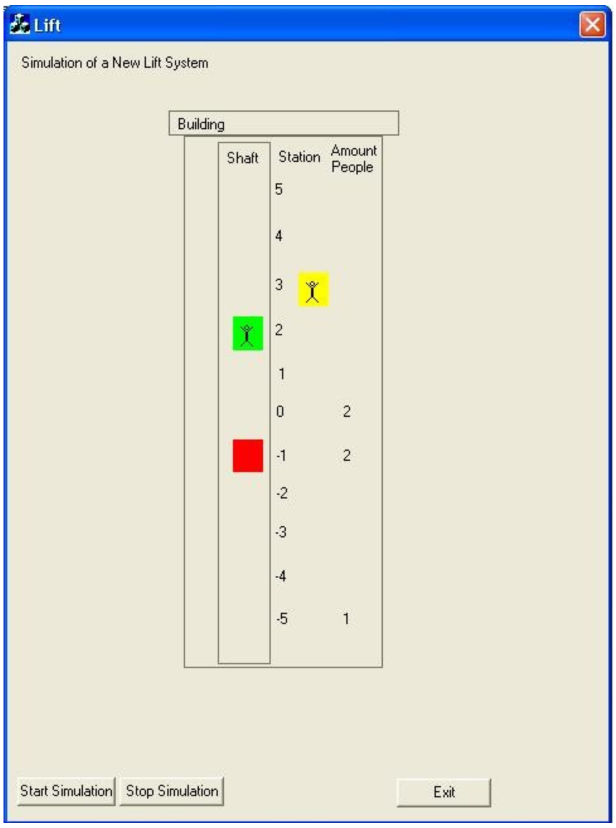


Illustration 24: Graphical user insterface: Simulating

5.5 Important Parts of the Program

5.5.1 Simulation

It is easy to add different functions to the program because there is not just one button which starts every process. By dividing the whole processing into classes, changes can be made locally and often do not affect other classes. They interact with each other as it is described in chapter 4 on page 10. The control does not have to do everything by itself but it gives different messages to the different classes and the classes have to execute these tasks. They are in a way smart but controlled by the control. Using this design structure makes it easier to change the code of the program as it is mentioned above.

5.5.2 The Deadlock Problem

A very interesting but difficult problem, which is partly solved now, was the deadlock one. A possible situation is shown in illustration . The green cabin wants to travel from the third to the second floor and the red cabin from the second to the third floor. They block each other which results to a deadlock of these two cabins. The solution is that the program knows whether there is a deadlock of two or more cabins. As soon as there is one it must solve it. The red cabin can not move the green cabin away because the latter is not idle as it is occupied by a person. Therefore one cabin must travel away with a person in it which is not that efficient, but there is no other solution to solve a deadlock.

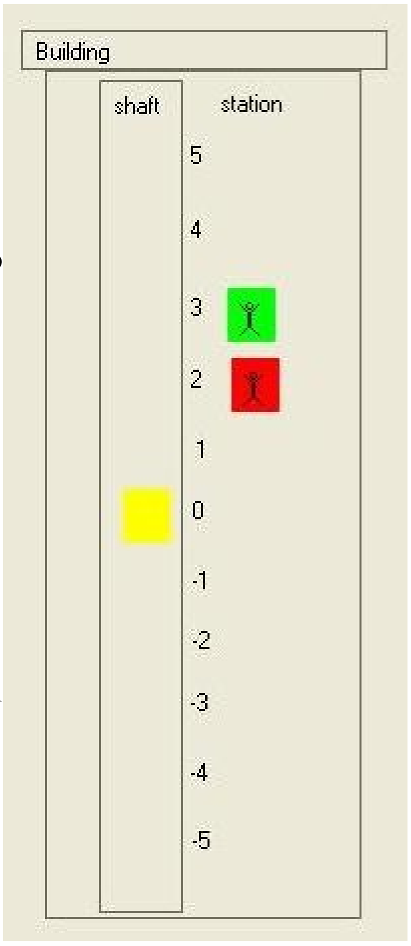


Illustration 25: Situation of deadlock

5.5.3 The Idea of the TimeStep[15]

The timeStep is a very important part of the program. A timeStep is a division of the time into small parts called steps. There are different processes going on at the same moment. A timeStep allows us to add a little time to the first one, then to the second, then to the third one and so on. Like this a process, which actually happens parallel to another one, will not be completed before the other one. Of course it can not be a one hundred per cent parallel but they are followed by each other just by a difference of one timeStep which is very small. It is possible to vary the duration of one timeStep. It can for example have the value 0.001 second, but it would also be possible to give it the value 1 hour which would not be very smart in this simulation.

The simulation program therefore has a time counter. By adding a time step, it adds the same amount of time to the time counter of the program. Like this it is ensured that time passes like in reality. Processes done by the computer, for example calculating, are infinitely fast. They are not counted. In reality this happens so fast that nobody can notice it.

The time based method, and not the event based, one ensures that the error of the time is very small, namely equal to the amount of one timeStep.

15 The word timeStep is a newly invented word.

5.5.4 The Passenger Generator

People are created by a random function. This is the code of this function in the Visual C++ program.

Number	Code
1.	personsInSecond = (personsPerHour/3600.0);
2.	for(time = m_timeCounter + m_time; time >= 1; time = time - 1)
	{
	number = random(0, 100);
3.	if (number <= personsInSecond*100)
4.	{
	generatePerson();
	}
	m_timeCounter = m_timeCounter - 1;
5.	}

Table 8: Code of the passenger generator

Explanation:

1. One can determine how many people shall take the lift in one hour on the average. The variable *personsPerHour* is used for this amount. The variable p*ersonsInSecond* calculates the average amount of people who press the button each second.
2. This line checks whether one second has passed or not.
3. This function returns a random number between 0 and 100.
4. On this line it is checked, on a random principle whether a person will be created or not. If the number returned from the third point is smaller than the variable *personsInSecond*, a person will be created.
5. This last function subtracts 1 second to ensure that the function in line three is called all seconds.

5.5.5 The Search-Function

The array of the cabins, array of the people, array of the segment cards as well as the array of the transport cards have a very important function which is used quite often. It is called the search-function. These arrays receive an ID from any object. Then the search-function searches the array for the object with the corresponding ID and returns a pointer to this object. The array of the segments and the array of the station segments have a function which is very similar to them. They receive the desired floor number instead of the ID. Then they search for the according pointer and return it.

5.6 The Visualizing Part of the Program

5.6.1 In General

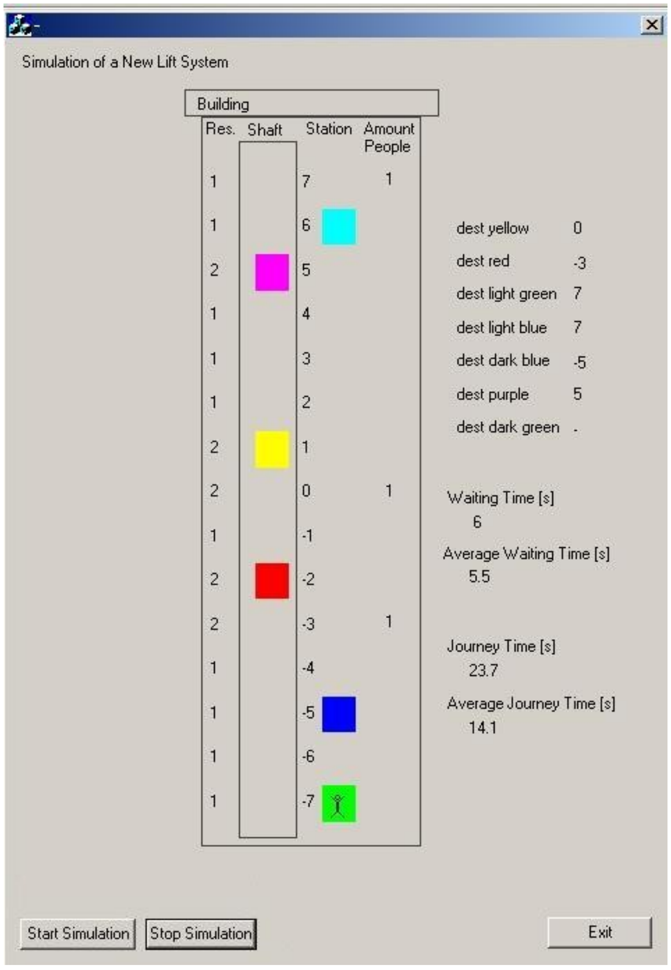
To control and illustrate the program the whole shaft with its cabins in it is visualized in the graphical user interface, see illustration 19 on page 23. In this way it is easy to follow the way of a cabin. After each timeStep the whole graphical user interface is updated. That is to say the cabins are redrawn on the new or on the same position .

The lift system in this visualizing part is defined by five floors above the ground floor and five below it, so there are 11 floors. There are up to three cabins but it is possible to add a new one with minimal changes of the code of the software.

The amount of the people who are waiting for a cabin to arrive is shown in the column *amount people.*

5.6.2 Improvements

The following illustration shows that it is easy to change the parameters of the building as well as to add new functions and visualize them. It is an expansion of the original program.


Simulation of a New Lift System

Building

Res.	Shaft	Station	Amount People
1		7	1
1		6 🟦	
2	🟪	5	
1		4	
1		3	
1		2	
2	🟨	1	
2		0	1
1		-1	
2	🟥	-2	
2		-3	1
1		-4	
1		-5 🟦	
1		-6	
1		-7 🟩	

dest yellow	0
dest red	-3
dest light green	7
dest light blue	7
dest dark blue	-5
dest purple	5
dest dark green	-

Waiting Time [s]
6

Average Waiting Time [s]
5.5

Journey Time [s]
23.7

Average Journey Time [s]
14.1

Illustration 26: Screenshot of improved program

6 Discussion

6.1 Advantages of the new lift system

The new lift system has the following advantages: People can reach their destination in large buildings without changing the lift shaft as this would be the case in the conventional lift system. Currently, people move with the lift cabin for example to the 30th floor and take another cabin in another shaft to get to the 60th floor. The reason why there is not a direct lift to the 60th floor is that the ropes would be too long and therefore are elastic and heavy. As longer the ropes are as greater is their expansion. Therefore the cabins can not stop very accurately at the floor. The weight of such a long rope is also a problem. So the length of a shaft is limited to about 30 floors. Illustration 27 illustrates this. The new lift system eliminates this problem because there are no ropes.

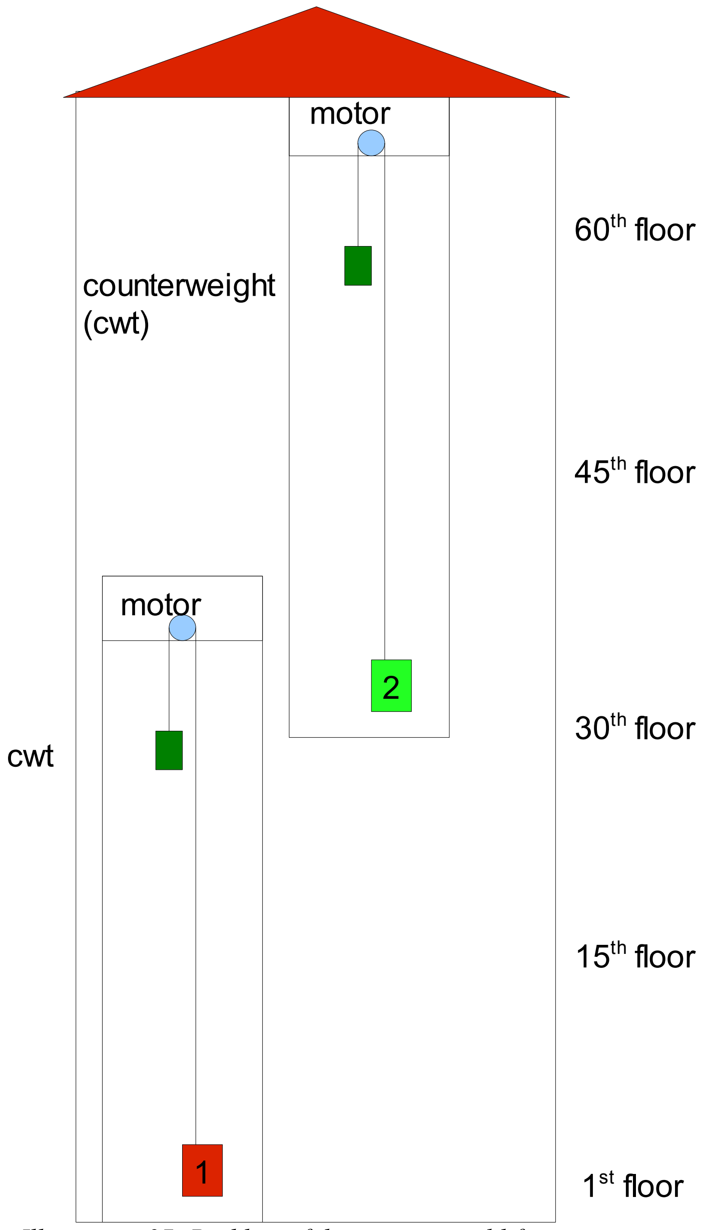


Illustration 27: Problem of the conventional lift system

A high comfort is guaranteed if all lifts are placed together in contrast to the situation, if all lifts are distributed around a building. In this lift system, all cabins are in one or several shafts and are therefore placed together.

A cabin moving with a person in it, does not stop at any other floor to let other passengers in. It moves directly to the desired destination of the person. This ensures that the person in it reaches the fastest way its floor as soon as the cabin is moving. If more than one person of the same floor wants to go to the same destination, the amount of people in a cabin can be up to six passengers. This could be the case if it is midday and many people who stay on the same floor want to go to the cafeteria. There are several cabins moving at the same time. This is a very efficient way to handle the passengers.

6.2 Disadvantages of the new lift system

A disadvantage is that a cabin can take at a maximum six people with it which are not many compared to the conventional lift system. If there is a large group of people all going to the same floor, several cabins are necessary to transport them.

The fact that the new lift system does not stop at any other floor to let other people in and out is an advantage for those who want to travel to a floor of great distance, but a disadvantage for those who would like to enter and travel a short distance with the cabin. Those people have to wait for a some time at the station for the cabin to arrive which is reserved for them. It could be that there are a lot of almost empty cabins travelling by until the reserved empty cabin arrives.

This lift design suffers in efficiency versus cabled elevators. It must carry the heavy motor in the lift instead of it being kept stationary. There is no counterbalance; thus the elevator must be capable of lifting the car's weight (including motor) and passengers' weight. A conventional design only needs to be able to lift the passengers' and the cable's weight. This results in using more power, and requiring a larger motor. This disadvantage has to be tested and studied.

6.3 Analysis of the Control System

„There are two levels of traffic control for lift systems. The lower level simply commands individual lift cars to move up or down, to stop or start and to open and close the doors. [...] The higher level of traffic control has the function of coordinating the activity of a group of lift cars, by means of a set of logical rules (the lift control algorithm) defined by the lift designer. The four primary tasks of a group traffic control system in serving both car and landing calls should be:

- to provide even service to every floor in a building
- to minimise the time spent by passengers waiting for service at a landing
- to minimise the time taken by passengers to move from one floor to another
- to serve as many passengers as possible in a given time.“ [16]

The first step, the so-called *lower level*, is fulfilled in a simplified version. The next aim is to find the optimum, the best lift system by changing different parameters.

The program follows these five important rules. „A lift system has to follow a set of rules, five in number, which the passengers accept and understand. [...]

- Rule 1: Car calls always take precedence over landing calls.
- Rule 2: A lift must not reverse its direction of travel with passengers in the car.

16 Barney, Gina. Elevator traffic handbook. Pages 296 - 270

- Rule 3: A lift must stop at a passenger destination floor (it must not pass it).
- Rule 4: Passengers wishing to travel in one direction must not to enter a lift committed to travel in the opposite direction [There are signals.]
- Rule 5: A lift must not stop at a floor where no passengers wish to enter or leave the car.“ [16]

In the new lift system, it makes perfect sense to follow the first rule, because there are many cabins in one shaft. Why would it be a good idea to stop a cabin with a person in it, when a second one can come quite fast? The second rule is applied, because there is only one person in a cabin and rule 1 is applied. Consequently the cabin travels firstly to the destination of the person in it and afterwards becomes *idle* and can be reserved again. By using rule 3, it is ensured that as soon as a person is in a cabin, it will travel as fast as possible to the destination. Rule 4 is fulfilled because there would be signals in case there are two shaft, one only for moving upwards and one only for moving downwards. The signals show in which direction the cabins travel. Of course there are always people who press the wrong button, it is impossible to avoid that. However the amount of people concerned is negligible. The cabin with a person does not stop to let other people in which is rule 6. This is ensured with the transport card.

6.4 Improvement Possibilities

There are different aspects which could be improved. The simulation is simplified but it would be possible to add improvements to the program.

6.4.1 Cabinet

Fact is, that a cabin can take at a maximum sic people with it. In the generally used lift system are many people in it. If for example two people wait on the same floor and want to go to the same destination, it would be much easier to take the same cabin. The waiting time decreases. However there must always be a limit of the amount of people who can enter the same cabin.

The cabin in the simulation does not accelerate, it travels at a constant speed all the time it is moving. That is to say, one has to be aware of the fact that a cabin must accelerate at the beginning and decelerate shortly in front of the destination. „Note that there is no limit to the velocity at which a passenger may travel in an enclosed lift car, as speed is not noticeable to the passenger. But the values of acceleration/deceleration (rate of change of velocity) should be limited to about one eighth of g [= $\frac{1}{8} * g = \frac{1 * 9.81\text{m}}{8\text{s}^2}$] or 1.5 m/s^2 and the values of jerk (rate of change of acceleration) to 2.0 m/s^3.“ [17]

Therefore one could take the values for the rated speed as well as for the acceleration from the following table:

„Table 9 provides guidance on the selection of the speed of a lift based on the premise that the total time to travel the distance between terminal floors at rated speed should take between 20 s and 30 s. In the table the single floor flight times assume a 3.3 m inter floor distance and are slightly larger than theoretically derived values to allow for the doors to be locked and proved, the brake to lift and other start-up delays.[...]” [18]

17 Barney, Gina. Elevator traffic handbook. Page 84.
18 Ditto. Page 114.

Lift travel (m)	Rated speed (m/s)	Acceleration (m/s^2)	Single floor flight time (s)
<20	<1.00	0.4	10.0
20	1.00	0.4-0.7	7.0
32	1.60	0.7-0.8	6.0
50	2.50	0.8-0.9	5.5
63	3.15	1.0	5.0
100	5.00	1.2-1.5	4.5
120	6.00	1.5	4.3
>120	>6.00	1.5	4.3

Table 9: Typical lift dynamics [18]

The travelling time would probably be shorter if the cabins were stationed on the ground floor in the middle or at the top depending on the time. In the morning they could be on the ground floor. In the evening they could be stationed at the top. The rest of the time they could stay on different floors. The place which is best, has to be tested.

6.4.2 Person

In reality we have to be aware of the fact, that there are people who press the wrong button or other unwanted actions of people. So the chance to press the right button could be calculated and included. There can be special events in the skyscraper like fire alarm or a cabin defect.

The majority of the people want to go upwards in the morning and downwards in the evening. To determine the calculation the book „Elevator traffic handbook“ from Gina Barney (2003) could be used. This is not included in the simulation up to now.

This simulation does not focus on the fact that lifts are used mostly for long distance travel as the following table shows. However, this changes the results of the simulation just a little.

„Table 10 offers some guidance to the division of passengers between lifts and escalators in offices. The use of escalators is mainly inhibited by the length of time travelling.

Floors travelled	Lift	Escalator
1	10%	90%
2	25%	75%
3	50%	50%
4	75%	25%
5	90%	10%

Table 10: Lifts and escalators: Division of traffic

The provision of well signed and positioned stairs and escalators can considerably lessen the demands made on the lifts.“ [19]

The people have different characters. However, for the simulation it is negligible that for example one hates intimate zones so there is place for less people in a cabin. It is not necessary to focus on

19 Barney, Gina. Elevator traffic handbook. Page 21.

the difference in physical appearances what influences negligibly the maximal number of people who can enter a cabin.

There are people who take the lift upwards but walk downstairs. The former is negligibly small because then you would also have to include the amount of people who walk upwards as it is good for their health. These calculations are very small and therefore not included in the simulation.

6.4.3 Segment, Segment Card and Transport Card

In the simulation there is only one shaft, but for other results there could be two shafts. In one shaft the cabins would only drive upwards in the other they would only drive downwards as it is shown in illustration . It is simplified to make it easier to understand. Therefore the stations are not drawn.

A cabin would switch between the lift shafts. Maybe it would be easier if there were two stations on a floor which connect the two shafts. There could also be fast lifts which do not stop at each floor and travel for instance about 20 floors. There would have to be a special shaft for this. This must be tested and then compared to this lift system.

The array of the transport cards always takes the oldest transport card. The simulation could be improved if the best transport card is taken. The same is true for the array of the segment cards.

6.4.4 Visualizing part

There are different possibilities to improve the appearance of the simulation. Instead of picture, photography could be used. The people who are waiting are not drawn which would be nice. One idea is that the people who are waiting have the colour grey. If they are waiting for a long time they change the colour and get red. If they have to wait still longer they get nervous. Thus they are blinking read which means to the producer that the people are not satisfied.

The aim of this project does not highlight the visualizing part. Thus the simulation is visualized rather in an abstract way. It would lead to far to focus on the visualizing part. This would be a project for itself.

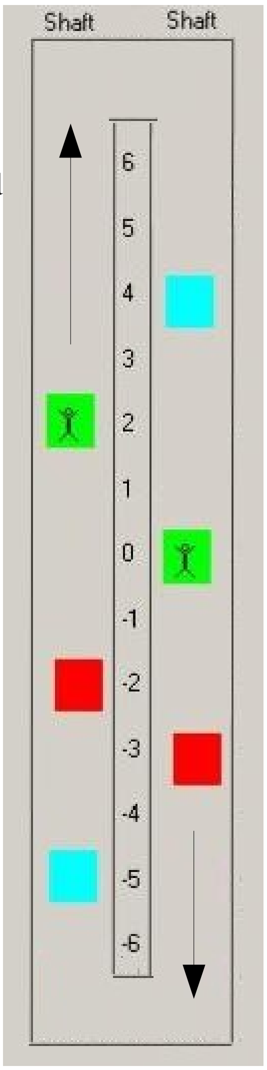


Illustration 28: Two shafts

6.4.5 In General

Gamse wrote in his report that people seem to view waiting as wasting time, but travelling as doing something positive, which means that it is more important to reduce the waiting period than the travelling time. The new lift system pays more attention to the former than the latter.

The simulation must be improved so that no deadlock appears at all and no other mistake can be found. However on the whole, the program is near to reality and could be used to test the new lift system. As soon as everything is improved it will be quite exact. One can analyse the lift system by the simulation using tables and diagrams from which you could get good new interesting statistics.

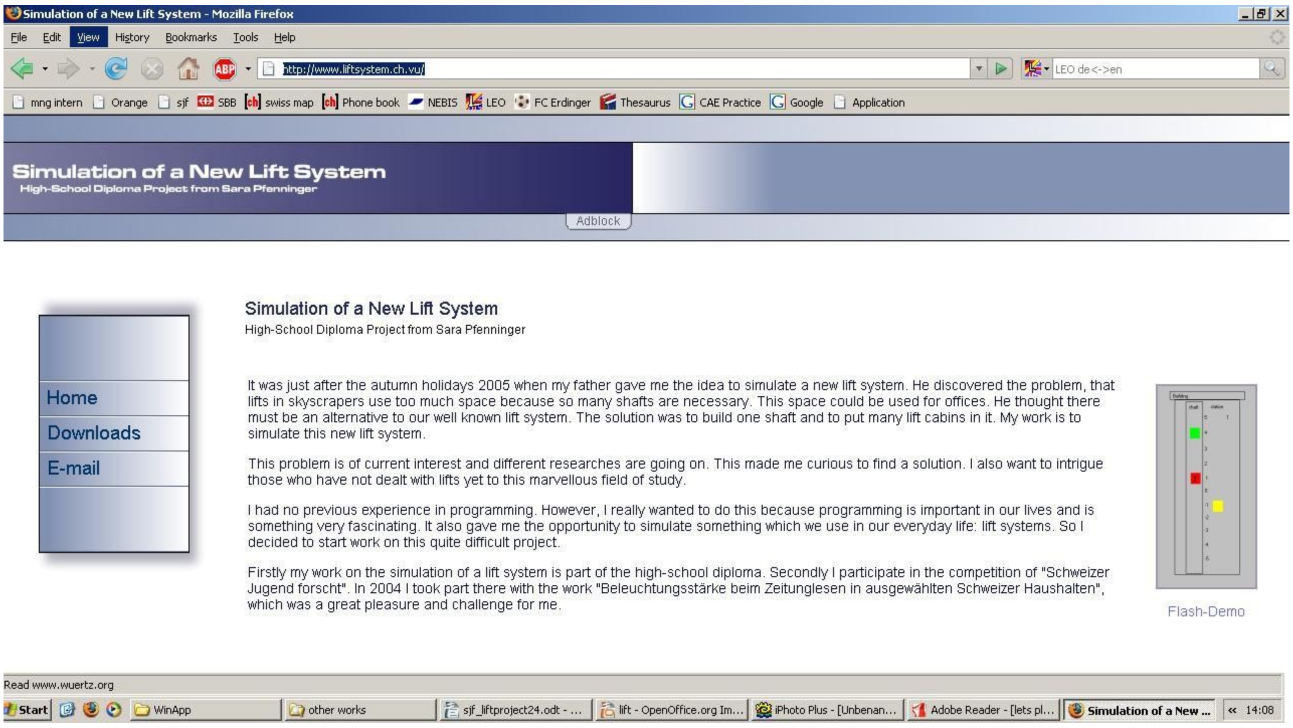
6.5 Conclusion

The simulation can be used to weigh the advantages and disadvantages. However the lift system must be optimised regarding what is possible because it is easy to change the code of the program.

7 Closing Words

I managed to simulate a basic lift system. I made a lot of interesting experiences in programming in Visual C++. The beginning was quite hard, but now I developed an understanding how programming in C++ works. It was not just the program itself which took me hours to work on the same function, but also the logical thinking. There are very often many ways which lead to the goal, but there is usually only one good way. I can never say that the program is finished; I always get new ideas of how the program could be improved. I still really enjoy this project and I think I will never get bored by it.

There is also a webpage of this project. Have a look at the screenshot in illustration 29. So, if you are interested in the program, you can download it at www.liftsystem.ch.vu.


Simulation of a New Lift System - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.liftsystem.ch.vu/
LEO de<->en
mng intern Orange sjf SBB swiss map Phone book NEBIS LEO FC Erdinger Thesaurus CAE Practice Google Application
Simulation of a New Lift System
High-School Diploma Project from Sara Pfenninger
Adblock
Home
Downloads
E-mail
Flash-Demo
Read www.wuertz.org
Start WinApp other works sjf_liftproject24.odt - ... lift - OpenOffice.org Im... iPhoto Plus - [Unbenan... Adobe Reader - [lets pl... Simulation of a New ... 14:08

Simulation of a New Lift System
High-School Diploma Project from Sara Pfenninger

It was just after the autumn holidays 2005 when my father gave me the idea to simulate a new lift system. He discovered the problem, that lifts in skyscrapers use too much space because so many shafts are necessary. This space could be used for offices. He thought there must be an alternative to our well known lift system. The solution was to build one shaft and to put many lift cabins in it. My work is to simulate this new lift system.

This problem is of current interest and different researches are going on. This made me curious to find a solution. I also want to intrigue those who have not dealt with lifts yet to this marvellous field of study.

I had no previous experience in programming. However, I really wanted to do this because programming is important in our lives and is something very fascinating. It also gave me the opportunity to simulate something which we use in our everyday life: lift systems. So I decided to start work on this quite difficult project.

Firstly my work on the simulation of a lift system is part of the high-school diploma. Secondly I participate in the competition of "Schweizer Jugend forscht". In 2004 I took part there with the work "Beleuchtungsstärke beim Zeitunglesen in ausgewählten Schweizer Haushalten", which was a great pleasure and challenge for me.

Illustration 29: Webpage: www.liftsystem.ch.vu

8 Bibliography

8.1 Books

1. Barney, Gina. 2003. Elevator traffic handbook. Spon Press. London.
2. Barney, Gina. 1995. Elevator technology 6. The international association of elevator engineers. England
3. Barney, Gina. 1986. Elevator technology. Ellis Horwood limited. England.
4. Barney, Gina. 1977. Elevator Traffic Analysis Design and Control. Books Short Run Press Ltd., England.
5. Gamse, Beryl. 1976. A Simulation of An Elevator System For a Moderate Height Building. Institute of Transportation Studies University of California. California.
6. Krämer, Walter. 1999. Wie schreibe ich eine Seminar- oder Examensarbeit? Campus Verlag. Frankfurt / Main.
7. Markon, Sandor und Kita, Hajime et al. 2006. Control of Traffic Systems in Buildings. Springer-Verlag. London.
8. Microsoft Corporation. 1988. C for yourself. Microsoft Corporation. Ireland.
9. Niederhauser, Jürg. 2000. Die schriftliche Arbeit. Dudenverlag. Zürich.
10. Schellong Helmut. 2005. Moderne C-Programmierung. Springer-Verlag. Berlin Heidelberg.
11. Lübke, Diethard. 2002. Aufsatz Deutsch. Dudenverlag. Zürich.
12. Willms, André. 1998. C++ Programmierung. Addison Wesley Verlag. Deutschland.

8.2 Internet

1. Wikipedia. C++. http://en.wikipedia.org/wiki/C%2B%2B (June 2006)
2. Wikipedia. C (Programming language). http://en.wikipedia.org/wiki/C_programming (June 2006)
3. Wikipedia. Elevator. http://en.wikipedia.org/wiki/Elevators (June 2006)
4. Vernon (2004). Multiple-Cab Elevator Loop. http://www.halfbakery.com/idea/Multiple-Cab_20Elevator_20Loop (November 2006)
5. Fukumoto Hideshi (2006). Circulating multi-car elevator. http://www.hqrd.hitachi.co.jp/global/news_pdf_e/merl060301nrde_elevator.pdf (November 2006)

9 List of Illustrations and Tables

List of Illustrations

List of Tables

10 Appendix

10.1 Enclosed CD

On the enclosed CD on the cover page one can find the simulation of the new lift system which was programmed in Visual C++. There are two files of the simulation. One is for testing the lift system and one is for simulating it. These user graphical interfaces were described in chapter 5.2 above on page 30. One can also find the report on the CD of this project in the *Portable Document Format* as well as all articles from the internet I used.

10.2 Confirmation

I confirm with signature that this work was independently written and brought into written form, that the cooperation is limited to advising and proofreading and that all used documents and people who helped are mentioned.

Date: 19th March 07 Signature: S. Pfenninger