

SEGWAY

| | |
|---------------------|----------|
| MATS WEDELL | MATWE812 |
| JENS OLOVSSON | JENOL753 |
| ALEXANDER PETROVSKI | ALEPE261 |
| JOHAN WALLENTIN | JOHWA576 |
| JACOB SHEIK | JACSH289 |

HANDLEDARE ANNA LOMBARDI

2004 01 18

A B S T R A C T

The purpose of this project is not really to make the best simulation possible, but to realise the problems involved in a physics simulation and the estimations that are needed to get a good result. The decision of which object to graphically simulate stranded on a segway scooter. It is easy to realise that it is hard to simulate the physics, in particular when you do not really know how the segway scooter works, but the result looks good on the screen. There may be some problems if we should continue implementing more things in the simulation, and some changes would then be needed. Nevertheless, the result is acceptable, and we feel that we have learned a lot.

P R E F A C E

This report is put together, by the Segway group, to give the reader an understanding of what it means to model a physical system and to show that everything does not have to be mathematically correct in order to look good. In fact, the main problem in modelling, one could say, is to know how and when to make use of so called “shortcuts”. For further reading on how the model works in reality please see the segway homepage in the list of references at the end of the report.

(<http://www.segway.com/>)

We would like to thank Anna Lombardi for assisting us during the project and keeping us on track.

TABLE OF CONTENTS

| | |
|--|-----------|
| 1 BACKGROUND | 2 |
| 2 AIM | 2 |
| 3 METHOD | 3 |
| 3.1 PHYSICS | 3 |
| 3.2 PROGRAMMING | 3 |
| 3.3 3D-MODELLING | 3 |
| 3.4 APPLICATIONS | 3 |
| 3.4.1 <i>Microsoft Visual C++ 6.0</i> | 3 |
| 3.4.2 <i>3D Studio MAX 5.1 & 6.0</i> | 3 |
| 3.4.3 <i>CrossRoads 3D 1.0</i> | 4 |
| 3.4.4 <i>3DWIN 4.9</i> | 4 |
| 3.4.5 <i>Microsoft Word 2002</i> | 4 |
| 3.4.6 <i>Bulletproof FTP Server 2.21</i> | 4 |
| 3.4.7 <i>Microsoft Remote Desktop</i> | 4 |
| 3.4.8 <i>Adobe PhotoShop 7.0</i> | 4 |
| 4 PHYSICS | 5 |
| 5 PROGRAMMING | 7 |
| 6 MODELLING AND GRAPHICS | 7 |
| 6.1 MODELLING SEGWAY | 7 |
| 6.2 MODELLING WILSON | 8 |
| 6.3 TEXTURING | 9 |
| 7 EXECUTION AND NAVIGATION | 10 |
| 7.1 STARTING THE PROGRAM | 10 |
| 7.2 NAVIGATION | 10 |
| 8.1 CONCLUSIONS AND ANALYSIS | 11 |
| 8.2 RESTRICTIONS | 11 |
| 8.3 THINGS THAT COULD HAVE BEEN MADE DIFFERENT | 11 |
| 8.4 REFERENCES | 12 |
| 8.4.1 <i>Book references</i> | 12 |
| 8.4.2 <i>Internet references</i> | 12 |
| 8.4.3 <i>Tutorial references</i> | 12 |

1 BACKGROUND

The idea of the modelling project is for students to use their knowledge from previous courses, mainly the course Modelling and Simulation, to describe a physical system and to model it. Since the system to model can be almost anything, it is up to the students to find a suitable one, neither too complex nor too simple. As soon as the group had been formed, the first topic on the list was to select a system that everyone felt they would like to work with. The first suggestions were to model a pool table or an air balloon. The main reason for doing the pool table was that a lot of time then could be spent on improving the visual appearance of the simulation. These ideas soon fell short though, since all the members of the group opted for more interactivity while running the simulation. At a pool table the only input is the force and angle of the cue. So instead, the group's choice fell on a, hopefully, more interactive system, the Segway. One of the group members had recently found an article describing the "transportation of the future" which everyone found very interesting. After some research on the Internet and consultation with Anna Lombardi it was all set. The system to be modelled was a Segway. (<http://www.tlb.org/scooter.html>)

2 AIM

The primary goal with this simulation project is, except learning that it is hard to do a realistic simulation, to model and simulate a segway scooter. A segway scooter is a self-balancing, personal transportation device. It only has two wheels, but yet it manages to stay upright by itself. To move the segway scooter you just lean forward or backward, but to maintain balance, the motor activates and turns the wheels at just the right speed to create a torque that will pull you back and keep you from falling down. To turn left or right, the driver simply turns the steering grip left or right, and the motors spin one wheel faster than the other, or spin the wheels in opposite directions, so that the vehicle rotates.

The plan is to implement height differences (3-dimensional movement) and some factors that we can change during the simulation, like gravity, friction, mass, materials etc.

It is also necessary to use some kind of graphical interface, in our case an interactive program made in C++ with OpenGL as the graphics engine.

3 METHOD

3.1 PHYSICS

The physics of the Segway is first calculated on paper before it is implemented in the program. By using the laws of physics and some common sense, the fundamental physics and behaviour of the Segway is estimated. It is also discussed in detail which physical quantities, such as maximum speed and steering, that should be implemented later on in the project.

3.2 PROGRAMMING

Due to previous knowledge and the graphical advantage, C++ was the natural choice as the programming language. The group also wanted to learn more about programming in C++ and it is a language that will be useful in future courses.

3.3 3D-MODELLING

The making of all the 3D-models is done with a real 3D-modelling program, rather than making them directly in OpenGL. It seems both faster and easier to model in a 3D-modelling program and export it to our program, than creating models in OpenGL where every vertex has to be defined by hand. The choice of program is 3D Studio Max. In 3D Studio Max the models are exported as 3DS-format. These files are then converted to obj-format so that the object reader can read them.

3.4 APPLICATIONS

The applications that are used in this simulation project are the ones listed below. They are not presented in order of usage.

- *Microsoft Visual C++ 6.0*
- *3D Studio MAX 5.1 & 6.0*
- *CrossRoads 3D 1.0*
- *3DWIN V 4.9*
- *Microsoft Word 2002*
- *Bulletproof FTP server 2.21*
- *Microsoft Remote Desktop*
- *Adobe PhotoShop 7.0*

3.4.1 MICROSOFT VISUAL C++ 6.0

This application is the editor that is used to write all the code, but also to compile and debug the program. The main reason for choosing it is because of previous experiences with Microsoft Visual C++ and already written code that opens an OpenGL window. An alternative to Microsoft Visual is called Dev C++, but then parts of the code would have to be rewritten to be suitable for the Dev compiler package.

3.4.2 3D STUDIO MAX 5.1 & 6.0

Discreet's 3D Studio Max has all the tools that are required for modelling the Segway. The main advantage of using 3D Studio is that all of the group members have previous experience with it, mainly from the course "3D Computer graphics & VR". The reason why two versions are used is that the modelling is made at different locations – at home and in school.

3.4.3 CROSSROADS 3D 1.0

This freeware-application is converting files from 3DS-format to the Alias wavefront OBJ format. The OBJ-file that CrossRoads generates contains vertices and uv-coordinates.

3.4.4 3DWIN 4.9

This shareware-application is also a file-format conversion tool. Since CrossRoads 3D does not print the face normals of the polygons, this is an alternative.

3.4.5 MICROSOFT WORD 2002

To write the report, this standard application is used. Main reason is the overall-availability and knowledge.

3.4.6 BULLETPROOF FTP SERVER 2.21

This FTP-client is making all the transfers to and from the server that stores all the segway-simulator files. There are no other known alternatives and it works good most of the time.

3.4.7 MICROSOFT REMOTE DESKTOP

This application is used to remote-control a personal computer over the Internet. To have a remote-controlled computer is adequate, because it removes the limitations of applications that are not installed. 3DWIN and Crossroads 3D are programs that could not be installed at Campus. There are alternative programs as NetOP, but once again, previous knowledge in some programs compensates for their unavailability.

3.4.8 ADOBE PHOTOSHOP 7.0

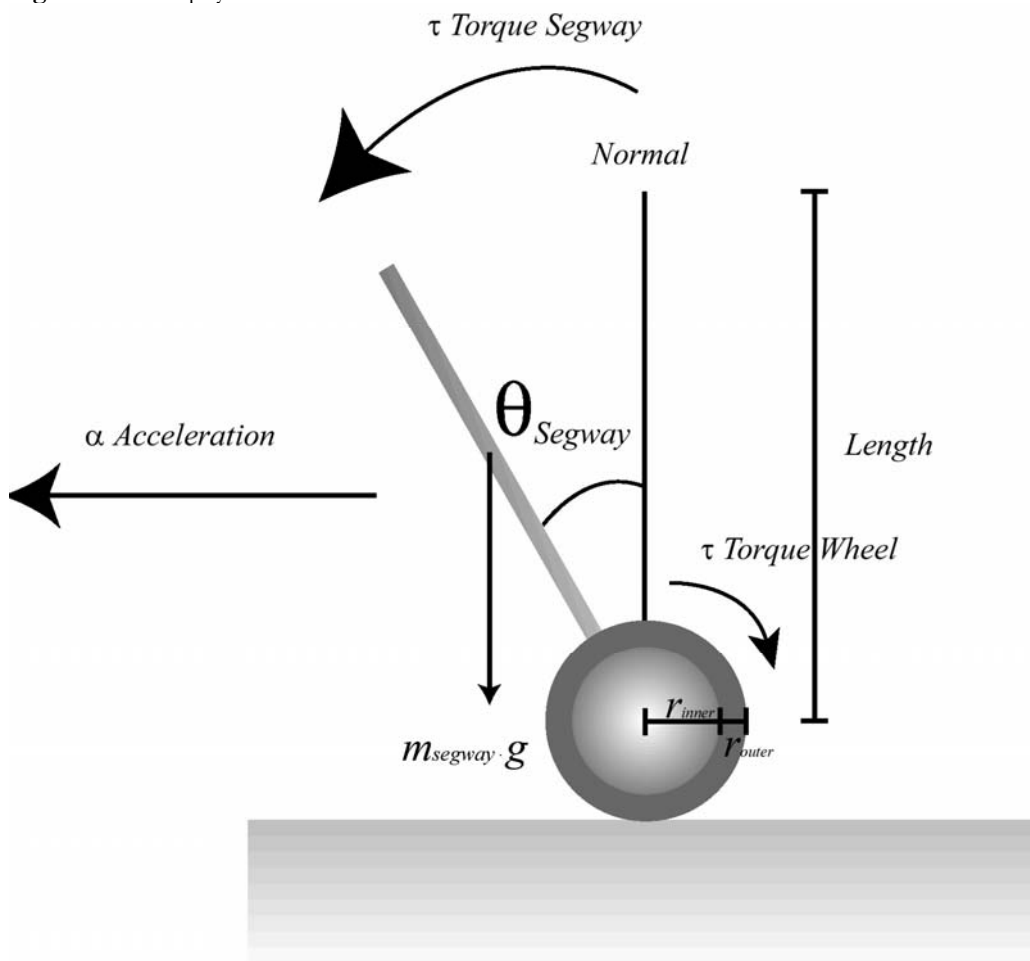
PhotoShop is used to design the textures. At the moment, there are no competitors to this application. The reason to use PhotoShop is still previous knowledge, but also its competence and capabilities.

4 P H Y S I C S

The principle of the Segways physics is quite easy. Just make sure that when you lean forward, the Segway accelerates forward and when you lean back it accelerates back. This will prevent you from falling down.

Simulating, rather than building, a real Segway has the advantages of knowing exactly how the model behaves and that the engine can have unlimited power. Therefore a pretty simple physics model can be implemented for a start.

Figure 1 – Basic physics



The picture shows the basic forces and torques acting on the Segway.

The first physics model just compensates for the torque that the Segway causes when you lean by accelerating the Segway so that the wheels cause an equal torque in the opposite direction. The formula for the wheel angular acceleration depends on the angle of the Segway:

$$\alpha_{Wheel} = \frac{m_{Segway} \cdot g \cdot l_{Segway}}{(m_{Wheel} \cdot (r_{Outer}^2 + r_{Inner}^2))} \cdot \sin(\theta_{Segway}), \text{ where } \theta_{Segway} \text{ is the only variable.}$$

Notice that only the mass of the Segway is used in the equation above. When accelerating forward, the person who drives just leans the Segway forward and not himself. However, when accelerating backwards the person will lean back and therefore the mass of the person must be used in the equation.

By knowing the angular acceleration of the wheels, the wheel angular velocity is first determined with the numerical method Euler. The inputs are the old wheel angular velocity, the current wheel angular acceleration and a static step, which in this case is 0.01 seconds. It returns the new angular velocity of the wheels. The c++ code for this numerical method is rather simple:

```
staticStepEuler(float old, float delta, float step)
{
return (old + delta*step);
}
```

Knowing the wheel angular velocity and acceleration, the acceleration, speed and displacement of the Segway can easily be determined by the following equations:

$$\alpha_{Segway} = r_{Outer} \cdot \alpha_{Wheel}$$

$$v_{Segway} = r_{Outer} \cdot v_{Wheel}$$

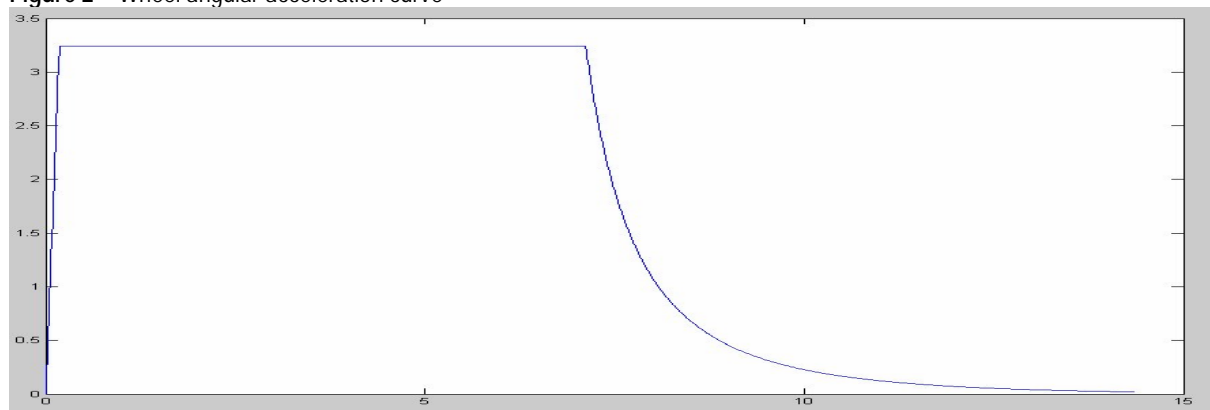
These are the only variables that are needed to make the first simulation of the Segway. Now a more complex physics model can be implemented.

A real engine has a power limit and a maximum speed that must be implemented. The Segway should not be able to move backwards, but programming a speed regulator solves these problems and does not affect the physical model.

Steering is implemented by making one wheel go faster than the other one. This is achieved by adding to the velocity of one wheel and subtracting from the other one. When the Segway stands still it can spin around pretty fast on the spot, but if you try to make a quick turn at high speeds you would most likely tip sideways. Therefore the maximum turning speed is reduced when the Segway reaches a higher speed.

The mathematic formulas used in the physical model are based on formulas retrieved from the books: University Physics and Physics handbook. For more information please see the list of references.

Figure 2 – Wheel angular acceleration curve



The curve shows the wheel angular acceleration with aspect of time when the model goes from standing still to maximum velocity

5 PROGRAMMING

C++ is the programming language chosen for the implementation of the project. The main reason for this is the group's previous experience with projects written in the language. In order to display a graphical window, the techniques used are OpenGL and WIN32 API. This choice is made because the group has previous experience with these techniques and to maintain complete control over the display process.

The decision to use OpenGL and WIN32 API allowed the group to reuse an old window class. Setting up the window quickly saves time in other aspects of the programming, e.g. creating the possibility to test the coding of the physics engine described earlier.

The first problem encountered during early development stages was the implementation of a realistic physics engine. The initial step is the implementation of differential equations which is done using Euler's algorithm for fixed step sizes. This is enough to enable basic physics calculations needed for the simulation of the Segway.

A derived problem from the physical simulation is the graphical representation of the simulation at hand, specifically, the timing of the representation on the screen. To cope with different calculation speeds, when using different computers, a timer function is developed to control the calculation process. This function limits the calculations done to once every 10 ms and therefore making the simulation work on a computer with a calculation capacity of at least 100 frames per second. If this is not the case a very "jerky" but physically correct simulation is visualised.

The structure of the project is separated into different classes depending on the use of the code. The 3D-model data is handled by an object model class. It has but one purpose: to read in and store the 3D-model data. A texture reader class reads the picture files used to texturize the 3d-model. The Segway specific code, physics calculations, is placed in a Segway class. The character, Wilson, is placed in a separate class to simplify the coding of his movements. In order to coordinate the classes, a player class is used to communicate between the different objects and the system specific code.

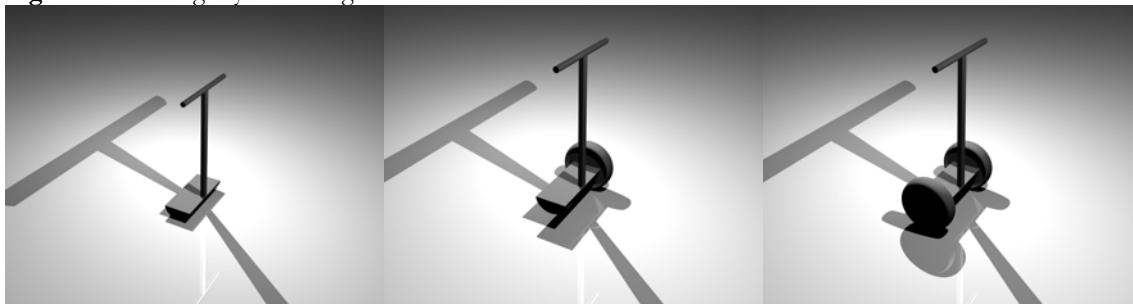
6 MODELLING AND GRAPHICS

6.1 MODELLING SEGWAY

3D Studio Max is the application that is used to graphically model the Segway. The graphical model is made up of simple primitives such as cylinders, toruses and boxes. To make the handlebars, one long horizontal cylinder is modelled. Under the handlebars, a vertical box is created as a pole and connected to the underside of the handlebars. Further, the underside of the pole is connected to a rectangular and vertical box that is round on the underside. The round part on the bottom of the box is actually a cylinder. Together they are attached and play the Segways role as the standing-board. On the sides of the standing-board the wheels are placed. The wheels are made up of a torus, which looks like a donut. Inside the torus there is a cylinder that is supposed to look like a rim.

The standing-board, handlebars and the pole are grouped into one mesh and the two wheels are separate objects. Now the Segway consists of three parts.

Figure 2 - The Segway modelling



The left picture shows the handlebars, pole and standing-board. The middle and the right picture show the complete Segway including wheels.

6.2 MODELLING WILSON

Wilson is a character that is included in one of 3D Studio Max Character studio 4 tutorials. This gives the advantage of not having to graphically model the mesh for the character that drives the Segway. In this project the physical simulation has higher priority than the graphical, therefore using Wilson is the obvious choice.

Figure 3 – The Wilson model



A picture of the Wilson model included in one of 3D Studio Max Character studio 4 tutorials

Still Wilson must be implemented in the program and some work on him will be done. First Wilson's mesh must fit the Segway and for this a biped is linked to the mesh. The biped works

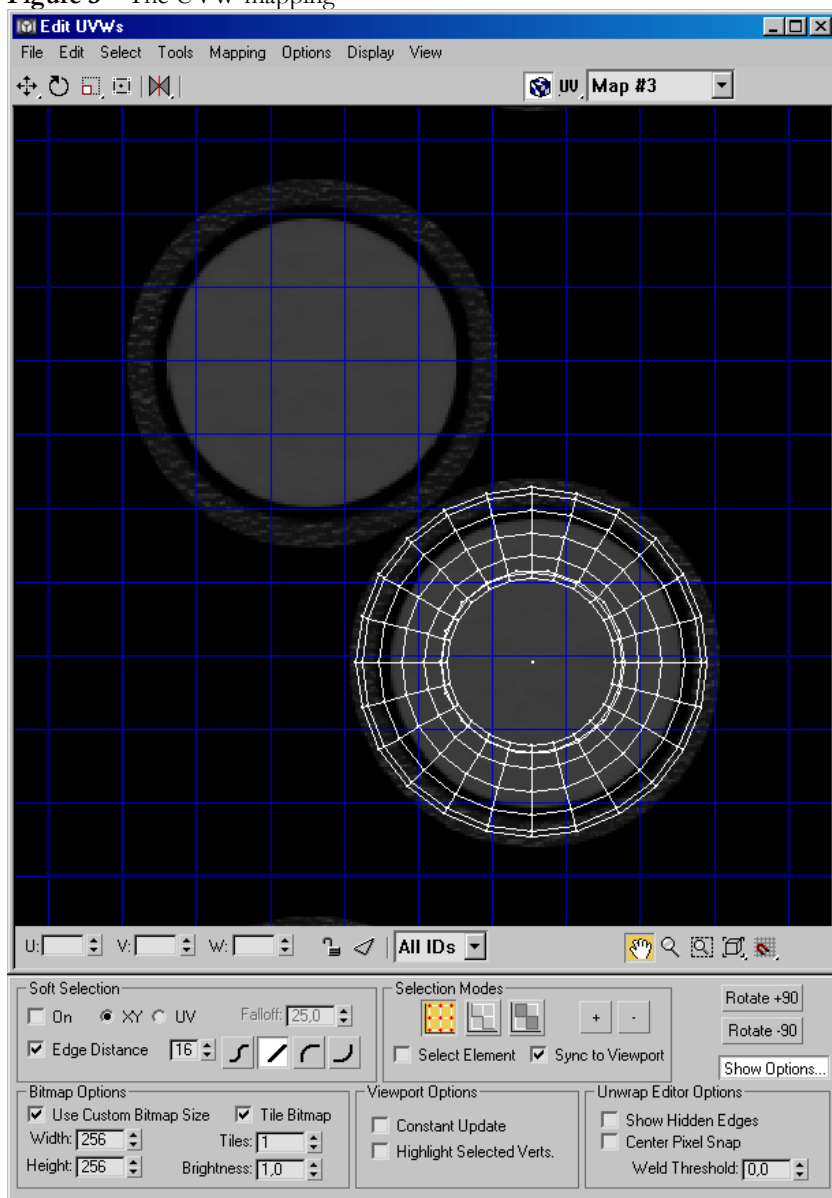
like a skeleton for the mesh and allows you to transform the mesh. When Wilson fits the Segway, the mesh is divided. First his arms are detached into four new meshes, one over- and underarm for each arm, thus it would be possible to move his arms in the program. The head, hair and eyes are attached to one single mesh instead of three. The head remains separated from the head because it will make the texturing easier later on. Now Wilson consists of six parts, the head, the body and both over- and underarms.

(3D Studio Max Character studio 4 Tutorials - Getting Started with Physique)

6.3 TEXTURING

To map the texture on to the object there is a function in 3D Studio MAX called unwrap uvw-map. This function allows the unwrapping of desired textures, and then parts of the map are placed on desired places on the mesh. There is no need of having a uniform texture. The Segway consist of two textures, one for the wheels and one for the standing-board, handlebars and pole. For Wilson, six textures are required, one for each part.

Figure 3 – The UVW-mapping



The picture above shows how the graphical 3D-model for one of the wheels is placed on a texture map. When assigning a texture to a 3D-model it requires the user to "tell" the program how to map it.

7 EXECUTION AND NAVIGATION

7.1 STARTING THE PROGRAM

First thing to do is to check that the computer satisfies the requirements.

Requirements:

- *1 GHz CPU with 256mb of RAM*
- *Geforce2 with 32mb graphics RAM*

Find the *segway.exe*-file and double-click on it. If everything is correct, a Segway should be displayed.

7.2 NAVIGATION

Before starting to navigate the Segway the camera-view should be chosen [optional]. There are three predefined camera-views. The views can be chosen by pressing '1', '2' or '3' on the left side of the keyboard.

To make the Segway move forward, just press the '+' on the numerical pad a couple of times to increase the tiltangle. This action makes the Segway move forward. To decrease speed or stop press '-'. Steering is made by pressing the 'o' or 'p'.

Move the camera position up and down by pressing 'q' respectively 'e'. To move it forward and back press 'w' respectively 's' and to move it left and right press 'a' respectively 'd'. Move the camera direction by using the arrows.

When pressing the escape-key the program is ended.

8 RESULTS

8.1 CONCLUSIONS AND ANALYSIS

At first, when the decision was made that it was a segway that should be simulated, it was quite hard to believe that the physics and mathematics should be a very big problem. However, as time went by, we realised that we did not really know how to simulate the physics of the segway. Some rough estimates were needed to get an appropriate result. It is a well-known fact that it is almost impossible to simulate real physics in a computer program, but it is a good lesson to experience the problems of a real physics simulation. Nevertheless, the simulation looks good, and for the layman it is hard to know that the physics is not really working for all cases.

8.2 RESTRICTIONS

The group first agreed to start with the simplest model possible, where the Segway only moves back and forth, and then adding to that model. This resulted in the fact that there was never enough time to implement some of the things that were initially decided on. The time shortage became apparent when physics problems arose that had not been thought of before. Because a lot of time had to be allocated to those problems the wish to make the Segway able to move uphill and downhill then became virtually impossible to implement. Friction on the ground was another idea that had to be removed from the plan due to insufficient time and the difficulties involved in selecting an appropriate friction constant. When creating the graphical Wilson model, the idea was to separate his body into six parts in order to make him bend his arms instead of his whole body when changing the tilt angle. Unfortunately there was no time to implement this which makes Wilson appear less like a real person. Since the real Segway has a person driving it that works as a regulator, the predictor used in the simulation has some flaws that would never appear in reality. If for example the modelled Segway is travelling close to 0 Km/h (minimum) and the tilt angle is set to a negative number, then the Segway receives a negative velocity. This would never in real life because a person, when reaching 0 Km/h, would take a step back and put down his/her foot. Since the predictor works the same way for the maximum and minimum velocity the problem is also visible when travelling close to 30 Km/h and increasing the tilt angle.

8.3 THINGS THAT COULD HAVE BEEN MADE DIFFERENT

The main problem with this simulation project was that all behaviours and properties of the real Segway scooter were not known. It is hard to simulate something you have not really seen with your own eyes. Maybe it would have been smarter to choose a different object to simulate and/or spend more time on the pre calculations.

It may also be easier to continue implementing things in this simulation project if the physic equations had been split into real forces, instead of integrated forces in the equations.

8.4 REFERENCES

8.4.1 BOOK REFERENCES

Benson, H. (1996). *University Physics*. John Wiley and Sons LTD

Nordling, Carl. (2000). *Physics Handbook*. SOS Free Stock.

8.4.2 INTERNET REFERENCES

Blackwell, Trevor (2003). *Building a Balancing Scooter* [www]
<<http://www.tlb.org/scooter.html>> Viewed 21/10 2003

Segway LLC (2003). *Segway* [www]
<<http://www.segway.com/>> Viewed 20/10 2003

8.4.3 TUTORIAL REFERENCES

3D Studio Max Character studio 4 Tutorials - Getting Started with Physique.