ANVENDT INFORMATIKK

eNytt

OPPTRYKK AV STUDENTARBEIDER I INFORMATIKK
VED
HØGSKOLEN I MOLDE
AVDELING FOR
ØKONOMI, INFORMATIKK OG SAMFUNNSFAG

DESEMBER 2007
NUMMER 7

REDAKSJON:
JUDITH MOLKA-DANIELSEN
HALVARD ARNTZEN

INNHOLD

Anmerkninger fra redaktørene:
Denne utgaven av Anvendt Informatikk - eNytt inneholder fire utvalgte prosjekter ved avdeling for økonomi, informatikk og samfunnsfag i høstsemesteret 2007. Rapportene som er presentert kan være noe forkortet i forhold til originalarbeidene, men de er ellers ikke redigert eller modifisert av redaksjonen eller andre.

Stikkord for de presenterte arbeidene er nettbutikker, business – lisenshåndterings -system, *network survivability, network routing, Second Life, user support*, strømmåler applikasjoner og strømsparing.

1. INF775: LISENSADMINISTRASJON FOR PROGRAMVARESELSKAP

AV – BØRGE GJENGSTØ OG OSKAR BÆVRE

MOTIVASJON

Innen programvareutvikling er det vanlig at programvareprodusenter ønsker å hindre/begrense piratkopiering av programvare. Programvaren som selges er gjerne bedriftens levebrød, og er derfor svært verdifull. Da en prøver å sikre sine verdier mest mulig, er det også naturlig å prøve å begrense bruken av programvaren til kun lisensierte brukere.

Oppgaven går derfor ut på å lage et lisenshåndterings-system og implementere dette inn i en applikasjon.

LISENSFILGENERATOR

Vårt lisensgenerator program genererer en xml-lisensfil. Lisensfilen er validert opp i mot et xml schema som sikrer at strukturen er korrekt. Lisensfilen inneholder informasjon om kunden, lisenstype og gyldighetsperiode. Lisensfilen er også kryptert med AES 128bit shared secret key, dette gjøres for å hindre brukere i å kunne endre lisensinformasjonen.

Lisensgeneratoren er implementert som en konsollapplikasjon som tar imot versjon, navn, nummer og utløpsdato som argument. Ut i fra disse argumentene genereres lisensfilen og en secret key, disse filene må brukeren av programvaren ha for å kunne bruke det lisensierte programmet. Generatoren har ingen mekanisme som flytter/sender filene til brukeren. Disse må flyttes manuelt ved epost, post, minnepinne eller annet flyttbart medie. Et alternativ kunne være å lage en installer som henter lisensfil fra webservice, slik at bruker får den tilsendt automatisk.

APPLIKASJON MED LISENSSJEKK

Vi har laget en liten notatblokk-applikasjon i java som krever en lisensfil og en secret key. Disse filene må ligge på samme sted som selve applikasjonen. Applikasjonen krever gyldig lisens for at brukeren skal få lov å bruke det. Det blir sjekket for gyldig lisens under oppstart, og bruker vil få en feilmelding om dette ikke er tilfelle. Mer spesifikt får bruker tilbakemelding om at lisensen har gått ut på dato eller om datoen har blitt tilbakestilt.

Applikasjonen implementerer et lisenssjekk-bibliotek vi har laget. Lisenssjekkbiblioteket leser inn den krypterte lisensfilen og secret key. Lisensfilen blir så dekryptert og informasjonen hentes ut. På grunnlag av disse dataene håndterer biblioteket verifisering av lisensen. For å hindre at systemklokken kan stilles tilbake ved utgått lisensperiode, lagres dato for siste aksessering i lisensfilen.

Løsningen demonstreres.

MICROSOFT SOFTWARE LICENSING AND PROTECTION SERVICES (SLPS)

SLPS tilbyr i hovedsak:

1. Beskyttelse av kildekode
2. Automatisering av lisenshåndtering.

Beskytte kildekode

Ubeskyttet kode er sårbar for "Reverse Engineering". Dette betyr at en kan reversere fra kompilert kode til lesbar kode, for eksempel via programmet: Lutz .NET Reflector [1]. En kan da stjele eller endre kildekoden, noe som gjør det lettere for eks. crackere å endre programmets oppførsel. Med dette mener vi eksempelvis å omgå lisenssjekken, eller å legge inn ondsinnet kode.

Programkildekode kan inneholde bedriftsintern informasjon, passord, ipadresser og lignende. Det kan derfor være ønskelig for mange å kunne sikre kildekoden mot "Reverse Engineering", da tilgang på slik kildekode vil kunne gi konkurrenter et konkurransefortrinn.

Se demo på hvordan dette kan løses i SLPS nedenfor.

Lisenshåndtering

Microsoft SLPS støtter flere aktiveringsmodeller for å aktivere lisensen til en applikasjon. Dette krever nettilgang da programmet må koble seg opp mot SLPS-server.
Ved å benytte Code protector SDK kan en gi forskjellig tilgangsrettighet til programmet ut fra lisenstype. Microsoft SLPS beskytter også mot piratkopiering ved å kreve gyldig lisens.

Vi hadde tenkt å ta en liten demo på dette også, men for å kunne benytte SLPS-lisenshåndteringen, må en registrere seg for å få en product-key. Dette er den eneste tilbakemeldingen vi har fått:

"*Thank you for signing up for an evaluation of Microsoft Software Licensing and Protection Services. A representative will be contacting you with more information."*

Det ble derfor ikke mulig for oss å prøve ut dette i praksis.

Krav

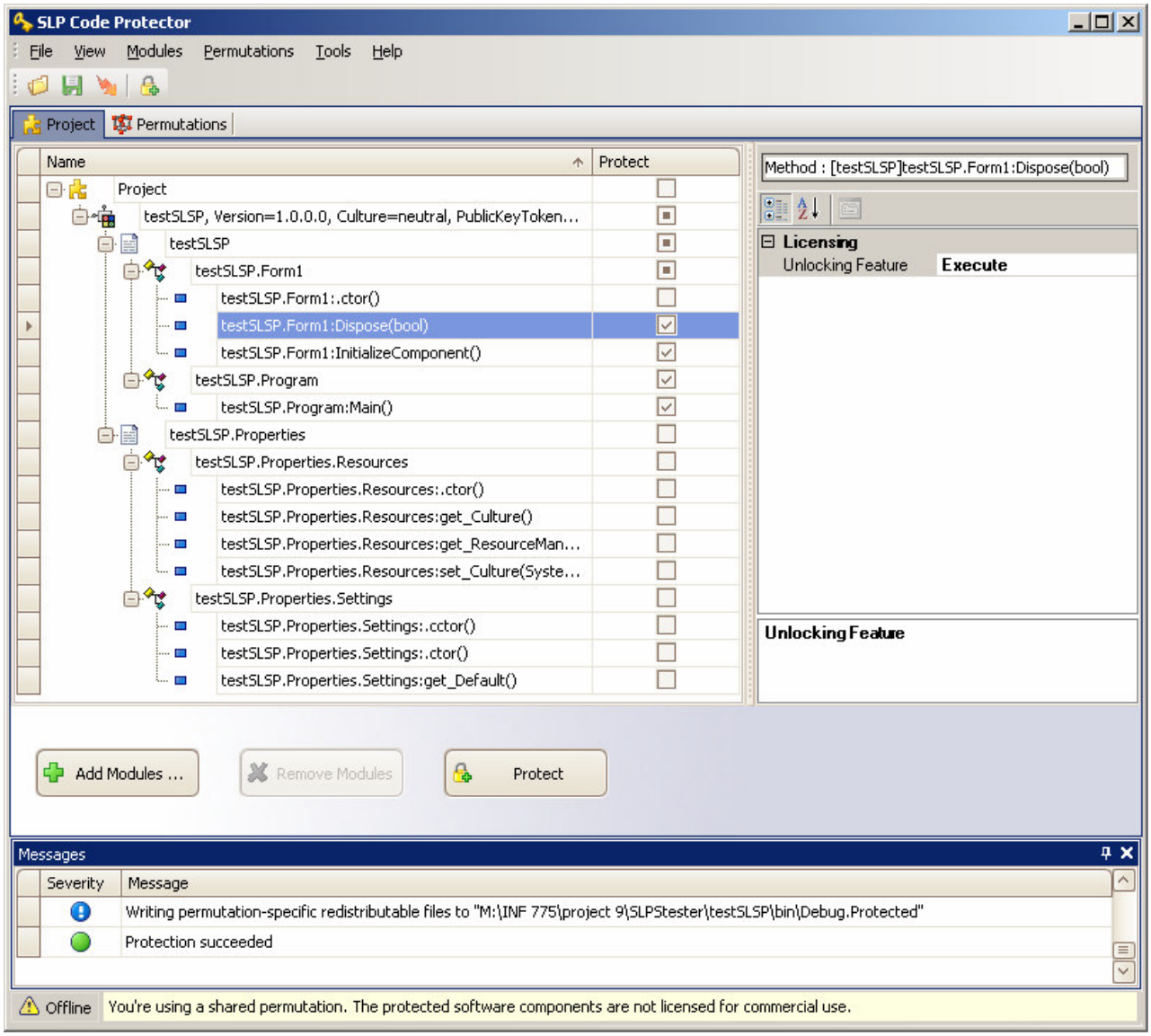
Som forventet av et system fra Microsoft kreves det at systemet kjører på en Windows plattform. Applikasjonene som skal beskyttes og/eller implementere lisenshåndtering, må være utviklet i .Net rammeverket. SLPS støtter følgende operativsystemer:

- Windows Server 2003
- Windows Vista
- Windows XP

I tillegg kreves: .Net 3.0 (SLPS 2008)

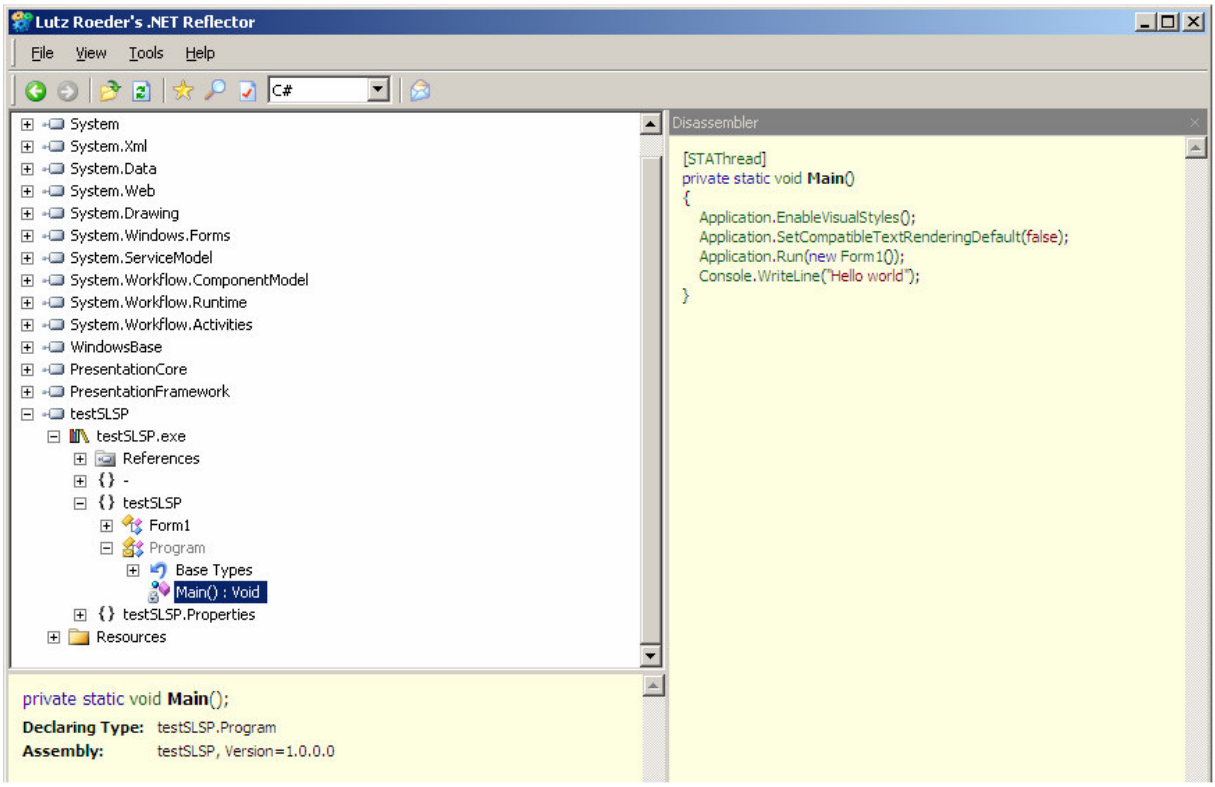
Demo: Kodebeskyttelse i SLPS

I offline modus er det ikke mulig å beskytte mer enn tre metoder, men det er tilstrekkelig for en liten demo. Vi lagde en liten .Net applikasjon som vi benyttet for å vise hva kodebeskyttelse går ut på.



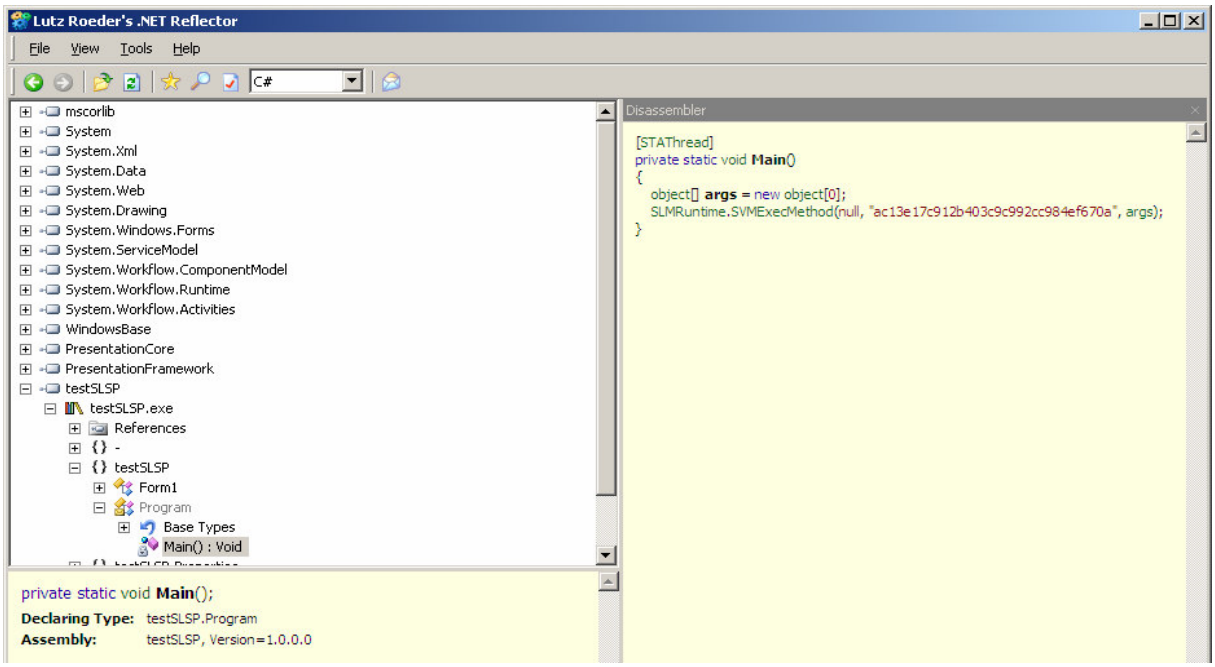
Ubeskyttet kode:

Her er ikke main() metoden beskyttet. Det var derfor ingen problemer å foreta en "reverse engineering" og få frem hele main() metoden, som vi kan se i bildet nedenfor.



Beskyttet kode:

Her er main() metoden beskyttet av SLPS, og som vi ser nedenfor er det ikke lett å få frem lesbar kildekode.



KONKLUSJON

Det er selvfølgelig store forskjeller mellom Microsoft's SLPS og vår lisenshåndterer, og kan i grunnen ikke sammenlignes. En av de største forskjellene er nok kompleksitet, pris og at SLPS også har kodebeskyttelse, det har ikke vår lisenshåndterer.

Det hadde nok vært mulig å tilnærme vår lisenshåndterer mer mot SLPS, men her må man ta totalkostnaden med i regnestykket. Har man et stort behov for lisenshåndtering, stor kundemasse og mange produkt, kan det fort bli stor kompleksitet i en lisenshåndterer. Det kan da bli ulønnsomt å utvikle lisenshåndtereren selv, og en kan med fordel benytte seg av en slik løsning. I motsatt fall, om man kan holde kompleksiteten nede, kan det være lønnsomt å utvikle lisenshåndtereren selv.

Hva kan vår lisenshåndterer:

- Beskytter mot piratkopiering (Delvis)*
- Kan håndtere lisensgyldighetsperiode

Hva kan Microsoft SLPS:

- Beskytte mot piratkopiering
- Håndtere lisensgyldighetsperioder
- Håndtere tilgangsrettigheter basert på lisenstype
- Beskytte kildekode

*Bedre beskytting om kildekoden beskyttes, kan også benytte andre og bedre krypteringsmetoder.

Bruk for oss

Siden vi utviklet vår applikasjon i Java og kjører Mac OS som plattform, egner SLPS seg helt klart ikke for oss, da SLPS krever Windows-plattform og at programmene utvikles i .Net.

Det kunne også i dette tilfellet vært mer lønnsomt å lage lisenshåndteringen selv, slik vi har gjort det, da programmet er lite og kundemassen liten.
Om vi hadde fått bruk for en mer kompleks lisenshåndtering finnes det også alternativer til Microsoft's SLPS, for eksempel JChain SA[2] og Java Code Protector[3]

Referanseliste

[1] Lutz .NET Reflector (url: http://www.aisto.com/roeder/DotNet/ dato: 21.11.2007).
[2] JChain SA (url: http://www.chainkey.com/en/jchains/index.htm dato: 21.11.2007)
[3] Java Code Protector (url: http://www.chainkey.com/en/jcp/ dato: 21.11.2007)

2. INF775: RELIABLE CONNECTIONS: LINK DISJOINT ROUTES

AV – OSKAR BÆVRE OG BØRGE GJENGSTØ

THEORY

Network survivability is increasingly important because the usage becomes higher every year. Nowadays a lot of critical information is exchanged over the internet, eg. hospitals, banks, stock exchanges. If these networks suddenly becomes unavailable there will be a lot of trouble for many many people, and also the economy could be heavily affected if the failure is over a long time.
We can divide network survivability into the three following categories:

PREVENTION

Prevention aims at preventing a failure in the network, this can in many cases be achieved by implementing various techniques, such as:

Technical techniques

- PKI
- Packet filtering
- Firewall
- IDS
- Etc.

Non-technical techniques

- Physical protection of hardware, cables etc.
- Proper personell training
- Scheduled maintenance
- Etc.

However, we can never be fully protected against failures. And if a failure occurs, prevention techniques will not be able to help us.

NETWORK DESIGN

Since prevention techniques only prevent us against failures, not recover from one, we need to design the net so that it can withstand a foreseen failure, and still be able to fulfill the given QoS requirements in the event of a failure. Optimization issues can also be solved here.

The design techniques typically used is graph theory and optimization-based formulas. Eg. Shortest path, joint-capacity allocation (JCA).

NETWORK MANAGEMENT

Network management can generally be divided into several functional areas: performance management (PM), configuration management (CM) and fault management (FM). Fault management handles and solves failures in the network. Fault management has the following major functions:

- Restore services, Eg. Rerouting, find new paths
- Root cause identification of failures
- Repair failed components
- Report the incidents.

THIS PROJECT

We built a webservice that provide two disjoint shortest paths for a given source and destination node in the topology. There is no given AP or BP, that is for the calling application to decide. The webservice could also be used in an APS 1+1 or 1:1 mechanism, due to the fact that we provide two reserved paths.
The purpose of this webservice is to give an illustration of how restoration and protection could be used in "the real world".

To provide two disjoint paths we used Suurballe and Tarjan's algorithm [1][2]. This is and extended version of the Dijkstra algorithm. We implemented it in a webservice, which has two methods. The first is for setting a network topology with a given link capasity, and the second is for finding, reserving and returning the two disjoint paths that fullfill the capacity requirement.
The webservice keeps track of the network topology and its current link state.

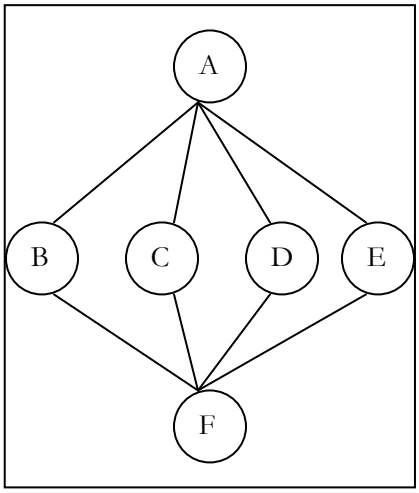
TESTS

We tried to utilize the entire network using random insertion of source –destination nodes and capacity needed. Because we have 100% redundancy we will never get more than 50% utilization of the network. However, as our tests clearly indicates, the utilization will be <50% in most cases.

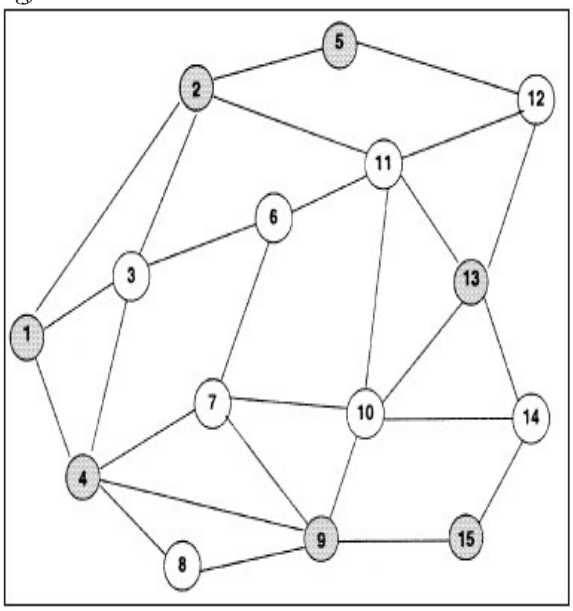
Setup

We tried 2 different topologies. The first one is constructed so that it utilizes exactly 50% of the network. The second topology is similar to multi-homed mesh networks.

Topology 1



Topology 2



Test 1

From node A – F on topology 1, n times until full capacity is reached.

Results:

n = 100 → Total usage: 50.0 %

See Attachment 1.1 for full test result

Test 2

From node 1 – 13 on topology 2, n times until full capacity is reached.

Uij = 40
n = 100
Cij = random[1-10]
Replications = 1

Results:

See Attachment 1.2 for full test result

We got a total network usage of nearly 13% at n=5, with a remaining capacity of 0% for the end to end path 1-13.

Test 3

Since a network usually has more than one user, we tried to make it more realistic by introducing variance in the start and end nodes. This way we try to simulate multiple network users and high network usage.

The test

- Selecting startnode and destination node randomly on topology 2
- running n times.
- k replications.

Since most of the randomly chosen start, destination nodes generates relatively short paths, and due to the fact that start and end node some time is equal, we use a fairly large n to achieve good results.

n = 1000 , k = 10.

Results:

Test#	Net usage in %
1	36.9
2	36.5
3	37.9
4	41.1
5	39.0
6	38.7
7	31.2
8	35.3
9	38.7
10	38.1
Avg usage:	**37.34**

On average we found the total network usage to be 37.34 %.

CONCLUSION

To improve the network usage in our example we could allow for shared backup paths, or one backup path for several working paths. Eg. 1:N / M:N APS.

As an example, if we have two working paths per backup path, the path redundancy would be 50% compared to the 100% redundancy 1:1 APS has.
The efficiency of the net is now greatly improved, but it gives a lower fault tolerance. This means that in some cases we can only tolerate one single link failure. For any critical infrastructure this can lead to a disaster.

As in any other topic, economy is considered when designing networks. Often we choose a solution that provides satisfactory Quality of Service (QoS), at the minimum cost.
Rings and p-Cycles can provide high restoration speed and fairly good reliability at a far lower cost than APS methods. However if we are dealing with critical infrastructure, cost may not be an issue. In such cases, network survivability and restoration speed are more important, and a fitting APS method could be considered.

REFERENCES

[1] Suurballe, J.W. and Tarjan, R.E., "A quick method for Finding Shortest Pairs of Disjoint Paths", *Networks*, Vol. 14, 1984, pp. 325-336

[2] Suurballe, J.W. "Disjoint Paths in a Network," *Networks*, 1974, pp. 125--45.

APPENDIX

SAMPLE OF TEST DATA

(FOR COMPLETE DATA CONTACT THE AUTHORS):

Test 1.1

Graf : a-b-2,a-c-3,a-d-4,a-e-1,b-f-4,c-f-3,d-f-2,e-f-1

Startnode : a Sluttnode : f

Runde : 1
Kostnad : 4
Path: [a,e, e,f : a,b, b,f]
--------Edges---------
link: a - b - 40 - 4 Ledig kapasitet: 36
link: a - c - 40 - 0 Ledig kapasitet: 40
link: a - d - 40 - 0 Ledig kapasitet: 40
link: a - e - 40 - 4 Ledig kapasitet: 36
link: b - f - 40 - 4 Ledig kapasitet: 36
link: c - f - 40 - 0 Ledig kapasitet: 40
link: d - f - 40 - 0 Ledig kapasitet: 40
link: e - f - 40 - 4 Ledig kapasitet: 36
--------/Edges---------
Total utnyttelse av grafen: 2.5

Runde : 2
Kostnad : 9

Path: [a,e, e,f : a,b, b,f]
--------Edges---------
link: a - b - 40 - 13 Ledig kapasitet: 27
link: a - c - 40 - 0 Ledig kapasitet: 40
link: a - d - 40 - 0 Ledig kapasitet: 40
link: a - e - 40 - 13 Ledig kapasitet: 27
link: b - f - 40 - 13 Ledig kapasitet: 27
link: c - f - 40 - 0 Ledig kapasitet: 40
link: d - f - 40 - 0 Ledig kapasitet: 40
link: e - f - 40 - 13 Ledig kapasitet: 27
--------/Edges---------
Total utnyttelse av grafen: 8.125

Runde : 3
Kostnad : 2
Path: [a,e, e,f : a,b, b,f]
--------Edges---------
link: a - b - 40 - 15 Ledig kapasitet: 25
link: a - c - 40 - 0 Ledig kapasitet: 40
link: a - d - 40 - 0 Ledig kapasitet: 40
link: a - e - 40 - 15 Ledig kapasitet: 25
link: b - f - 40 - 15 Ledig kapasitet: 25
link: c - f - 40 - 0 Ledig kapasitet: 40
link: d - f - 40 - 0 Ledig kapasitet: 40
link: e - f - 40 - 15 Ledig kapasitet: 25
--------/Edges---------
Total utnyttelse av grafen: 9.375

Runde : 4
Kostnad : 6
Path: [a,e, e,f : a,b, b,f]
--------Edges---------
link: a - b - 40 - 21 Ledig kapasitet: 19
link: a - c - 40 - 0 Ledig kapasitet: 40
link: a - d - 40 - 0 Ledig kapasitet: 40
link: a - e - 40 - 21 Ledig kapasitet: 19
link: b - f - 40 - 21 Ledig kapasitet: 19
link: c - f - 40 - 0 Ledig kapasitet: 40
link: d - f - 40 - 0 Ledig kapasitet: 40
link: e - f - 40 - 21 Ledig kapasitet: 19
--------/Edges---------
Total utnyttelse av grafen: 13.125

Runde : 5
Kostnad : 10
Path: [a,e, e,f : a,b, b,f]
--------Edges---------
link: a - b - 40 - 31 Ledig kapasitet: 9
link: a - c - 40 - 0 Ledig kapasitet: 40
link: a - d - 40 - 0 Ledig kapasitet: 40

link: a - e - 40 - 31 Ledig kapasitet: 9
link: b - f - 40 - 31 Ledig kapasitet: 9
link: c - f - 40 - 0 Ledig kapasitet: 40
link: d - f - 40 - 0 Ledig kapasitet: 40
link: e - f - 40 - 31 Ledig kapasitet: 9
--------/Edges---------
Total utnyttelse av grafen: 19.375

Runde : 6
Kostnad : 5
 Path: [a,e, e,f : a,b, b,f]
--------Edges---------
link: a - b - 40 - 36 Ledig kapasitet: 4
link: a - c - 40 - 0 Ledig kapasitet: 40
link: a - d - 40 - 0 Ledig kapasitet: 40
link: a - e - 40 - 36 Ledig kapasitet: 4
link: b - f - 40 - 36 Ledig kapasitet: 4
link: c - f - 40 - 0 Ledig kapasitet: 40
link: d - f - 40 - 0 Ledig kapasitet: 40
link: e - f - 40 - 36 Ledig kapasitet: 4
--------/Edges---------
Total utnyttelse av grafen: 22.5

Runde : 7
Kostnad : 10
 Path: [a,c, c,f : a,d, d,f]
--------Edges---------
link: a - b - 40 - 36 Ledig kapasitet: 4
link: a - c - 40 - 10 Ledig kapasitet: 30
link: a - d - 40 - 10 Ledig kapasitet: 30
link: a - e - 40 - 36 Ledig kapasitet: 4
link: b - f - 40 - 36 Ledig kapasitet: 4
link: c - f - 40 - 10 Ledig kapasitet: 30
link: d - f - 40 - 10 Ledig kapasitet: 30
link: e - f - 40 - 36 Ledig kapasitet: 4
--------/Edges---------
Total utnyttelse av grafen: 28.749999999999996

Runde : 8
Kostnad : 10
 Path: [a,c, c,f : a,d, d,f]
--------Edges---------
link: a - b - 40 - 36 Ledig kapasitet: 4
link: a - c - 40 - 20 Ledig kapasitet: 20
link: a - d - 40 - 20 Ledig kapasitet: 20
link: a - e - 40 - 36 Ledig kapasitet: 4
link: b - f - 40 - 36 Ledig kapasitet: 4
link: c - f - 40 - 20 Ledig kapasitet: 20
link: d - f - 40 - 20 Ledig kapasitet: 20
link: e - f - 40 - 36 Ledig kapasitet: 4

--------/Edges---------
Total utnyttelse av grafen: 35.0

Runde : 9
Kostnad : 7
 Path: [a,c, c,f : a,d, d,f]
--------Edges---------
link: a - b - 40 - 36 Ledig kapasitet: 4
link: a - c - 40 - 27 Ledig kapasitet: 13
link: a - d - 40 - 27 Ledig kapasitet: 13
link: a - e - 40 - 36 Ledig kapasitet: 4
link: b - f - 40 - 36 Ledig kapasitet: 4
link: c - f - 40 - 27 Ledig kapasitet: 13
link: d - f - 40 - 27 Ledig kapasitet: 13
link: e - f - 40 - 36 Ledig kapasitet: 4
--------/Edges---------
Total utnyttelse av grafen: 39.375

Runde : 10
Kostnad : 7
 Path: [a,c, c,f : a,d, d,f]
--------Edges---------
link: a - b - 40 - 36 Ledig kapasitet: 4
link: a - c - 40 - 34 Ledig kapasitet: 6
link: a - d - 40 - 34 Ledig kapasitet: 6
link: a - e - 40 - 36 Ledig kapasitet: 4
link: b - f - 40 - 36 Ledig kapasitet: 4
link: c - f - 40 - 34 Ledig kapasitet: 6
link: d - f - 40 - 34 Ledig kapasitet: 6
link: e - f - 40 - 36 Ledig kapasitet: 4
--------/Edges---------
Total utnyttelse av grafen: 43.75

Runde : 11
Kostnad : 2
 Path: [a,e, e,f : a,b, b,f]
--------Edges---------
link: a - b - 40 - 38 Ledig kapasitet: 2
link: a - c - 40 - 34 Ledig kapasitet: 6
link: a - d - 40 - 34 Ledig kapasitet: 6
link: a - e - 40 - 38 Ledig kapasitet: 2
link: b - f - 40 - 38 Ledig kapasitet: 2
link: c - f - 40 - 34 Ledig kapasitet: 6
link: d - f - 40 - 34 Ledig kapasitet: 6
link: e - f - 40 - 38 Ledig kapasitet: 2
--------/Edges---------
Total utnyttelse av grafen: 45.0

Runde : 12
Kostnad : 6

Path: [a,c, c,f : a,d, d,f]
--------Edges---------
link: a - b - 40 - 38 Ledig kapasitet: 2
link: a - c - 40 - 40 Ledig kapasitet: 0
link: a - d - 40 - 40 Ledig kapasitet: 0
link: a - e - 40 - 38 Ledig kapasitet: 2
link: b - f - 40 - 38 Ledig kapasitet: 2
link: c - f - 40 - 40 Ledig kapasitet: 0
link: d - f - 40 - 40 Ledig kapasitet: 0
link: e - f - 40 - 38 Ledig kapasitet: 2
--------/Edges---------
Total utnyttelse av grafen: 48.75

Runde : 15
Kostnad : 1
Path: [a,e, e,f : a,b, b,f]
--------Edges---------
link: a - b - 40 - 39 Ledig kapasitet: 1
link: a - c - 40 - 40 Ledig kapasitet: 0
link: a - d - 40 - 40 Ledig kapasitet: 0
link: a - e - 40 - 39 Ledig kapasitet: 1
link: b - f - 40 - 39 Ledig kapasitet: 1
link: c - f - 40 - 40 Ledig kapasitet: 0
link: d - f - 40 - 40 Ledig kapasitet: 0
link: e - f - 40 - 39 Ledig kapasitet: 1
--------/Edges---------
Total utnyttelse av grafen: 49.375

Runde : 32
Kostnad : 1
Path: [a,e, e,f : a,b, b,f]
--------Edges---------
link: a - b - 40 - 40 Ledig kapasitet: 0
link: a - c - 40 - 40 Ledig kapasitet: 0
link: a - d - 40 - 40 Ledig kapasitet: 0
link: a - e - 40 - 40 Ledig kapasitet: 0
link: b - f - 40 - 40 Ledig kapasitet: 0
link: c - f - 40 - 40 Ledig kapasitet: 0
link: d - f - 40 - 40 Ledig kapasitet: 0
link: e - f - 40 - 40 Ledig kapasitet: 0
--------/Edges---------
Total utnyttelse av grafen: 50.0
---

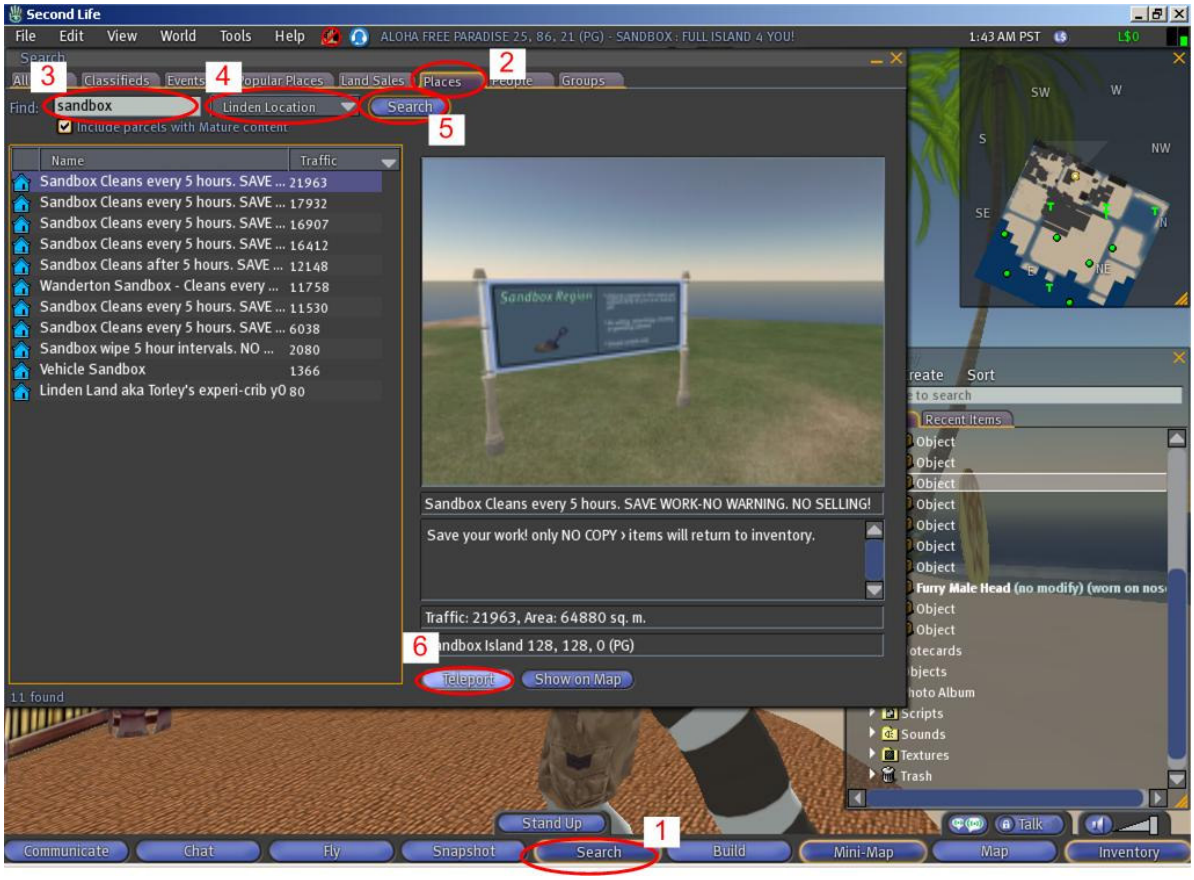
3. IBE209: HVORDAN LAGE EN LAMPE I SECOND LIFE - AV VEGARD SØBSTAD ALSLI, TOR OLAV REISTAD, SVEIN MAGNE PAULSEN, FRODE SANDBLÅST OG JOSTEIN HESTAD

Groups in the course IBE209 were asked to build a simple object in the virtual world of Second Life. After creating their object they were asked to write a web based tutorial. The object created by this group is a lamp. The virtual lamp is located in on Kamimo Island a virtual island for education owned by Molde University College. Kamimo Island can be found at the slurl: http://slurl.com/secondlife/Kamimo%20Island/117/143/25 .
After arriving on Kamimo Island at the Welcome area, the lamp is located in the Exhibition Center building. If you click on the lamp you will be sent to the web based tutorial. The web based tutorial is also located at the url: http://home.himolde.no/~molka/IBE209/index.php and it contains video instructions as well as the description below. All visitors to Kamimo Island are welcome to build in the Sandbox.

STEG1: GRUNNLEGGENDE

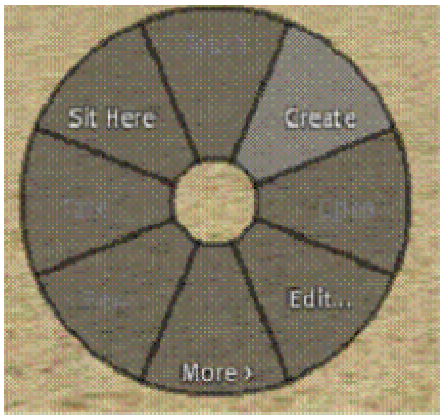
HVORDAN FINNE EN SANDKASSE (SANDBOX)

1. Åpne søkevindu
2. Klikk på *places* fanen
3. Skriv inn søkeord
4. Velg linden locations i rullegardinmenyen
5. Klikk *search*
6. Klikk *teleport*



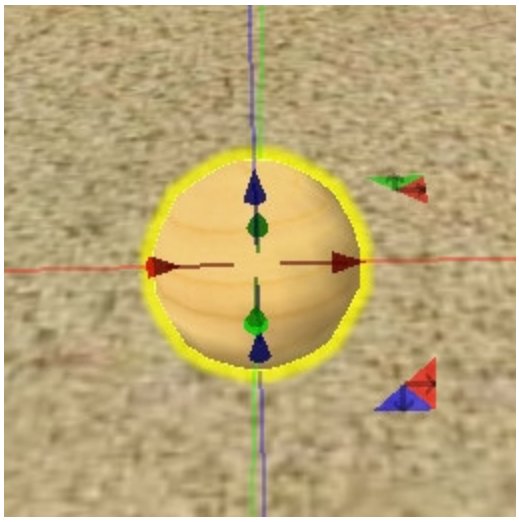
HVORDAN LAGE ETT OBJEKT

1. **Høyreklikk med musen på bakken** og **velg "Create"** fra sirkel-menyen.



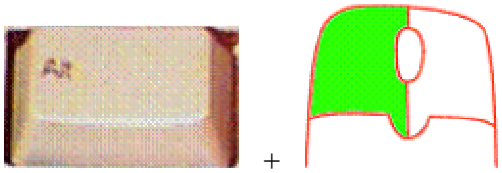
2. Her får du opp ”Create” vinduet. Her kan men velge hvilket objekt du skal lage. **Velg kule**.

3. **Trykk på bakken** der du vil plassere lampen. Da skal du få noe som ligner på dette:



HVORDAN ENDRE SYNSVINKEL

1. **Hold ned Alt + hold nede venstre museknapp på objektet**.



Pekeren blir forandret til dette ikonet: 

2. Du roterer rundt objektet ved å **flytte musen til høyre eller venstre**.
3. Ved å **dra musen opp og ned** kan du zoome ut og inn.

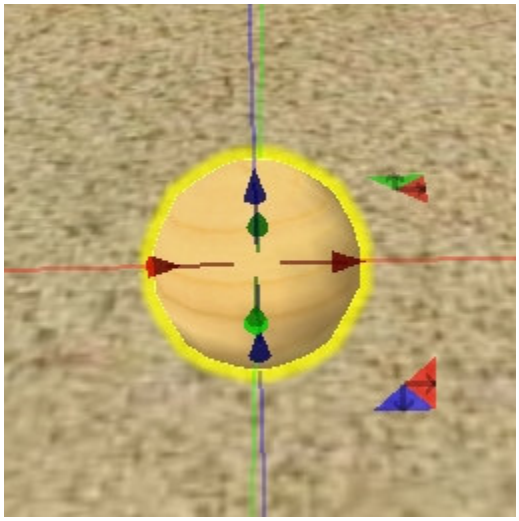
4. Ved å **holde nede Ctrl + Alt knappene** i tillegg til **venstre museknapp**, kan man vri kameraet opp og ned i forhold til objektet ved å **bevege musen opp og ned**.

Pekeren blir forandret til dette ikonet 

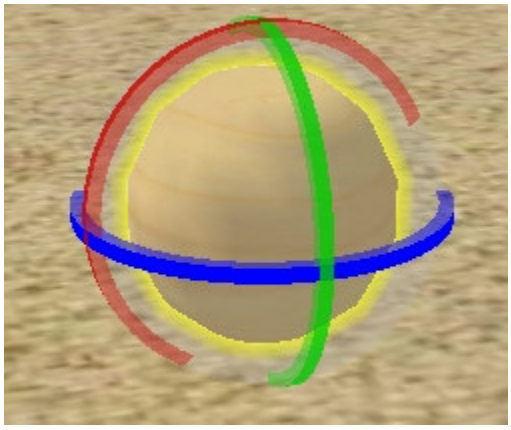
HVORDAN FLYTTE ETT OBJEKT

1. Når du står i **Edit modus**

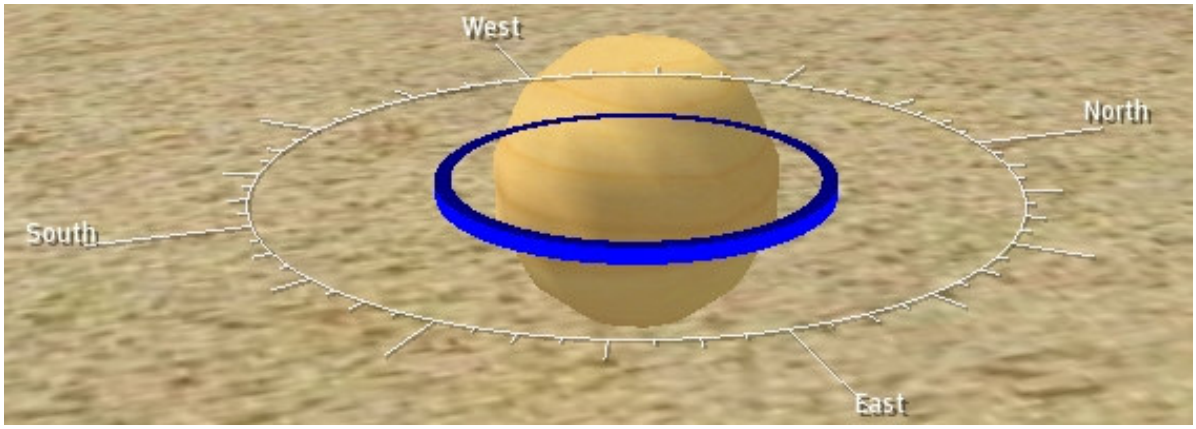
2. **Klikk på objektet** du ønsker å flytte på og bruk de fargede pilene på objektet til å flytte det rundt.
   - **Blå**: opp og ned
   - **Rød og Grønn**: til sidene



3. **Hold nede Ctrl knappen**, objektet du har valgt vil nå bli omkranset av en masse ringer.



Klikk og dra i disse for å rotere.



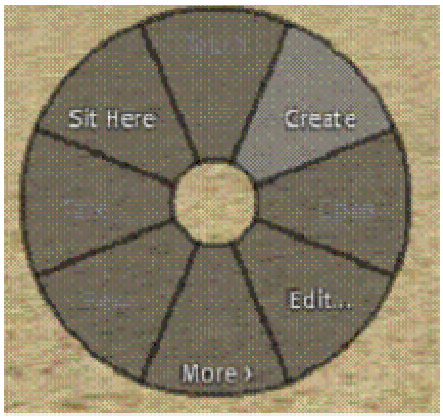
HVORDAN SLETTE ETT OBJEKT

1. For å slette ett objekt, **merk objektet** og **trykk på delete tasten**.

STEG 2: LAMPEFOT

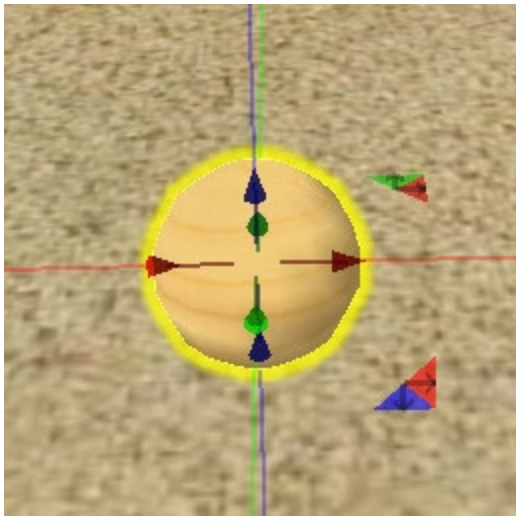
START MED LAMPEFOTEN

1. **Høyreklikk med musen** på bakken og **velg "Create"** fra sirkel-menyen.



2. Her får du opp ”Create” boksen. Her kan du velge hvilket objekt du ønsker å lage.
3. **Velg kule.**

4. **Trykk på bakken der du vil plassere lampen.**
5. Da skal du få noe som ligner på dette:

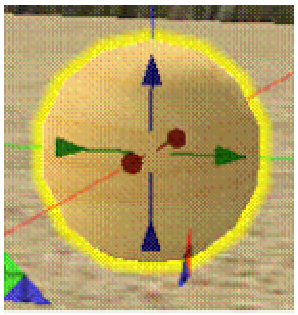


STEG 3: PÆRE

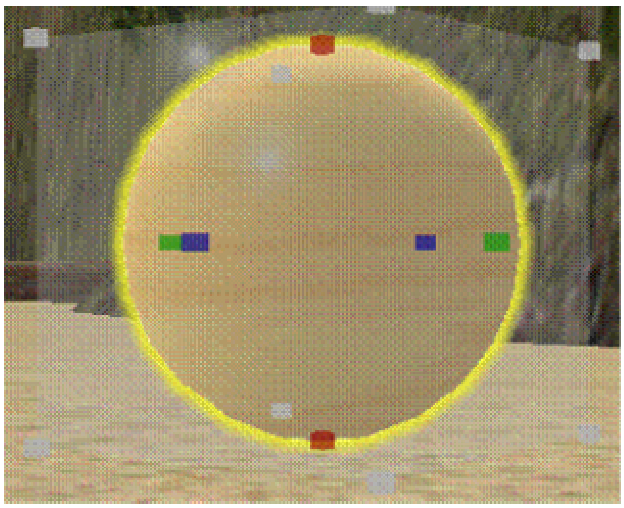
LAGE LYSPÆRE

1. Lag en ny kule. **Velg tryllestaven** eller **høyre klikk på bakken og velg create** fra sirkel menyen

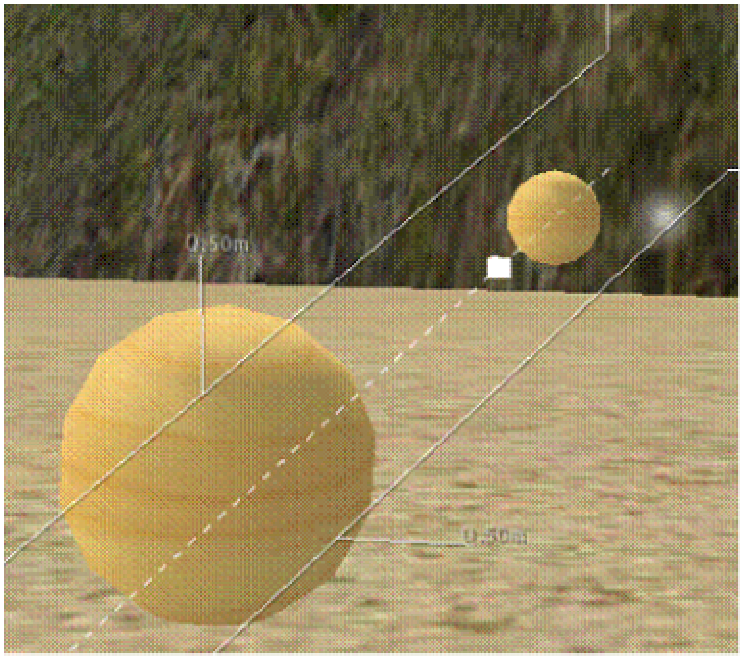
2. **Velg kule** og **trykk på bakken der du vil at objektet skal ligge**.



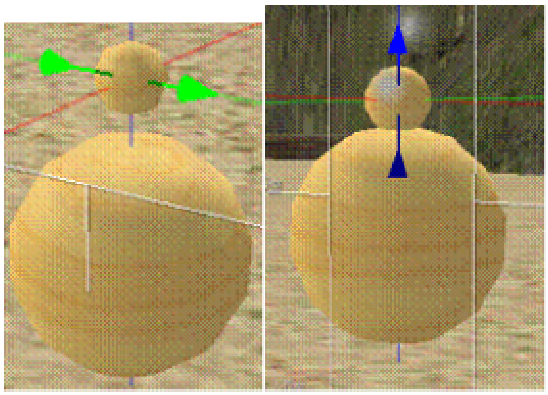
3. Nå skal vi forandre størrelsen på kula. **Hold nede ctrl + shift knappene** og kula vil se slik ut:



4. **Klikk med musen på en av de hvite firkantene i hjørene av kulen og dra til du har ønsket størrelse**.

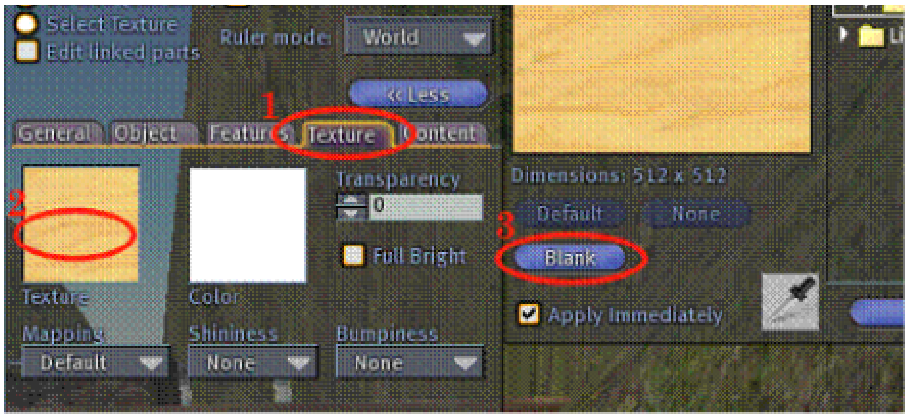


5. **Klikk og dra i de fargede pilene til kulen er i posisjon over lampefoten**.



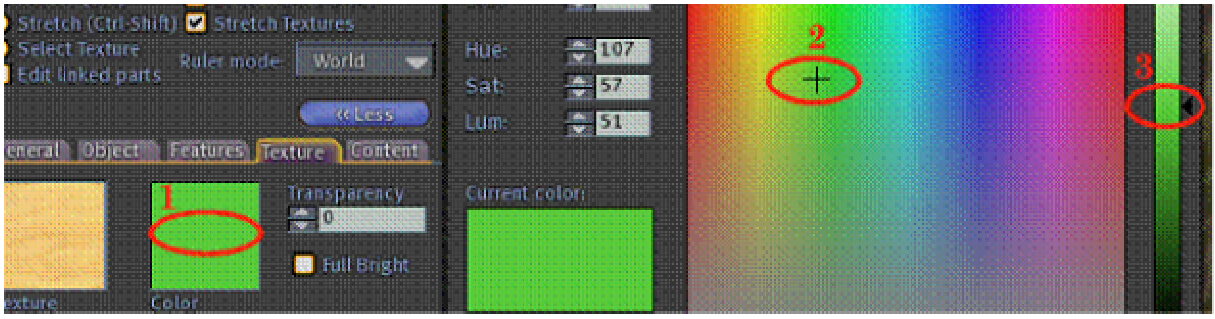
6. Nå skal vi forandre farge og fjerne texture på pæra. For å få opp utvidet informasjon **trykk på more knappen i edit boksen**(hvis det står less, er utvidet informasjon allerede valgt)

7. **Velg texture**[1], **trykk på den trefargede textureboksen**[2] og i det nye vinduet **trykk blank**[3]



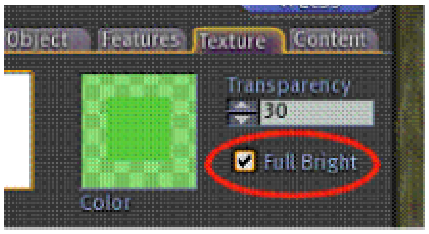
Dette vil fjerne texturen på pæra

8. Nå skal vi forandre farge på pæra. **Trykk fargeboksen**[1], **velg farge spekteret**[2] og **velg lysstyrke på fargen**[3].

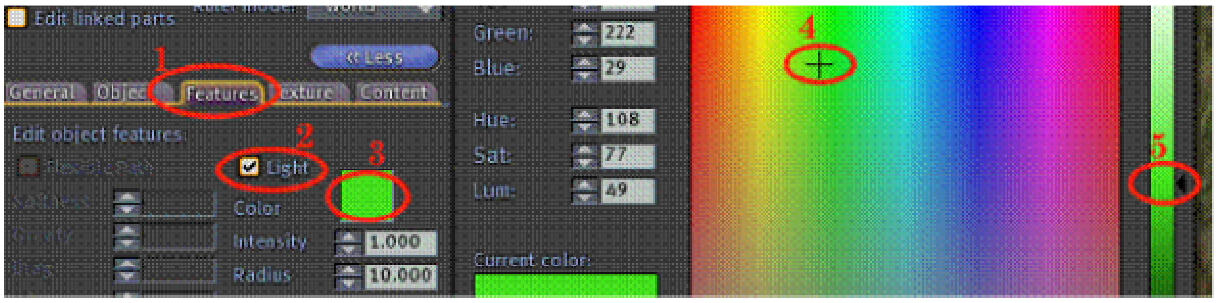


9. Nå skal vi gjøre pæren gjennomsiktig. **Juster Transparency til ønsket styrke**

10. Nå ska vi få pæren til å lyse. **Velg Full Bright i texture arkfanen**

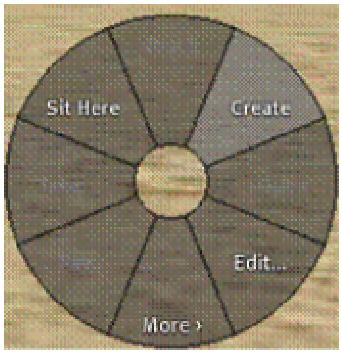


11. **Velg Features-arkfanen**[1], **velg light**[2], **trykk color boksen**[3], **velg farge fra spekteret**[4] og **velg fargens lysstyrke**[5]. Dette vil gjøre pæren til et lysende objekt i den fargen vi har valgt.



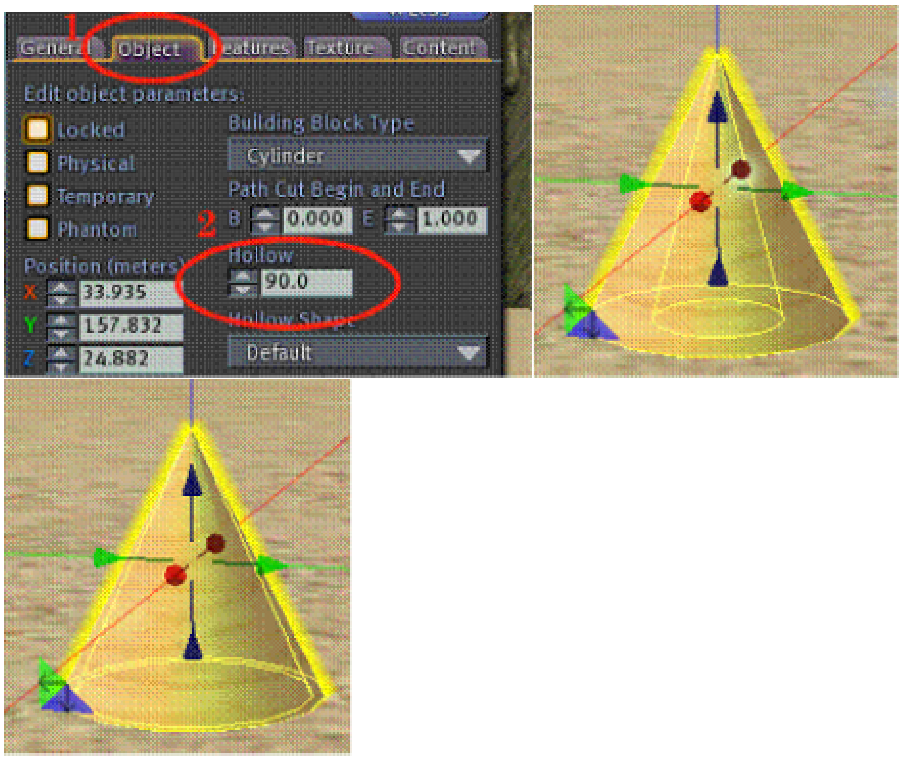
LAMPESKJERMEN

1. **Trykk tryllestaven** eller **høyreklikk på bakken og velg create fra menyhjulet** som kommer opp.

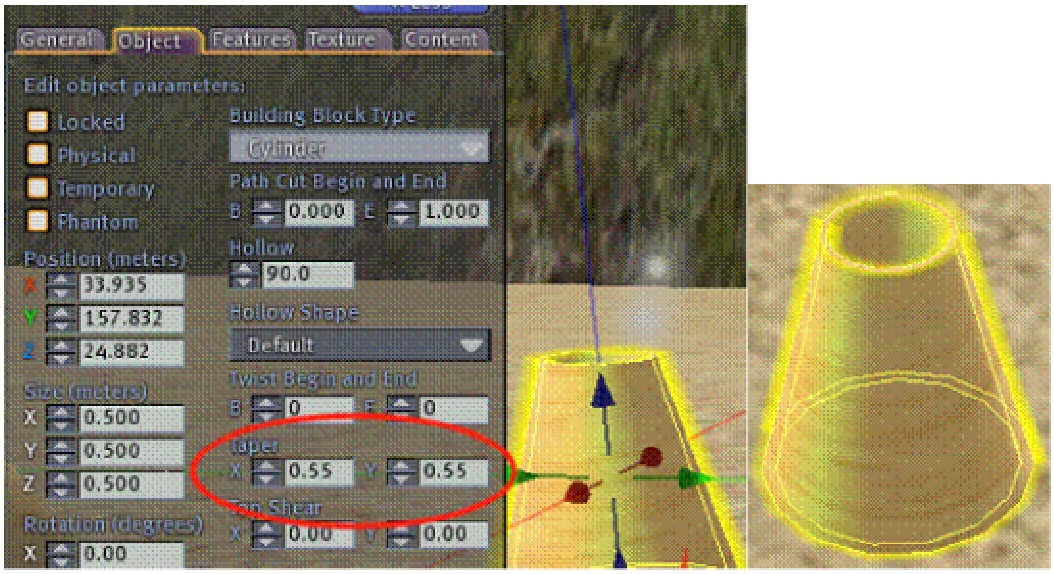


2. **Velg kjegle** og **trykk på bakken der du vil ha objektet**.

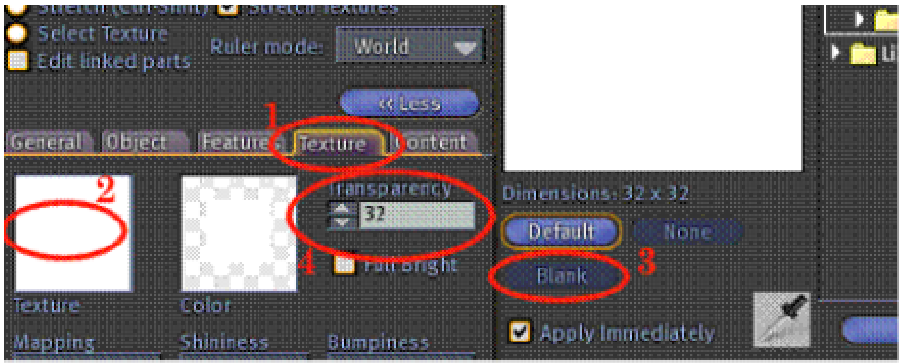
3. Vi skal nå gjøre den hul. **Velg objektarkfanen**[1], og **juster hollow parameteren**[2] til ønsket tykkelse.



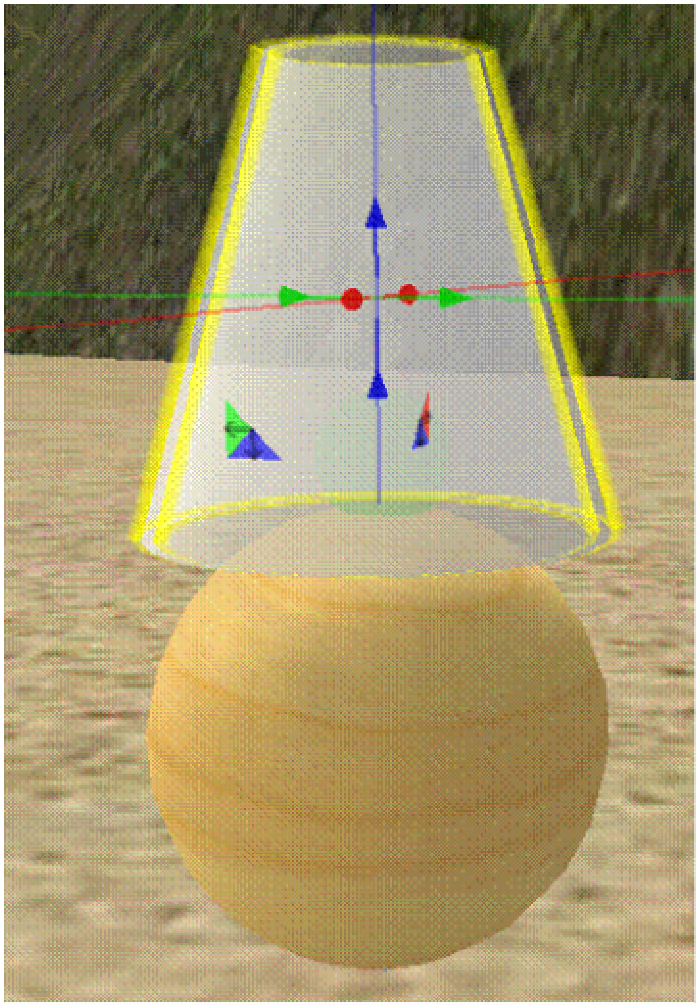
4. Nå skal vi tilpasse toppen av kjeglen. **Juster Taper like mye i begge retninger** til du får ønsket hullstørrelse.



5. Nå skal vi gjøre lampeskjermen gjennomsiktig. **Velg Texture**[1], **trykk texture boksen**[2]. **Velg blank fra det nye vinduet**[3]. **Juster så Transparency**[4] til ønsket gjennomsiktighet.



6. Flytt lampeskjermen på plass.

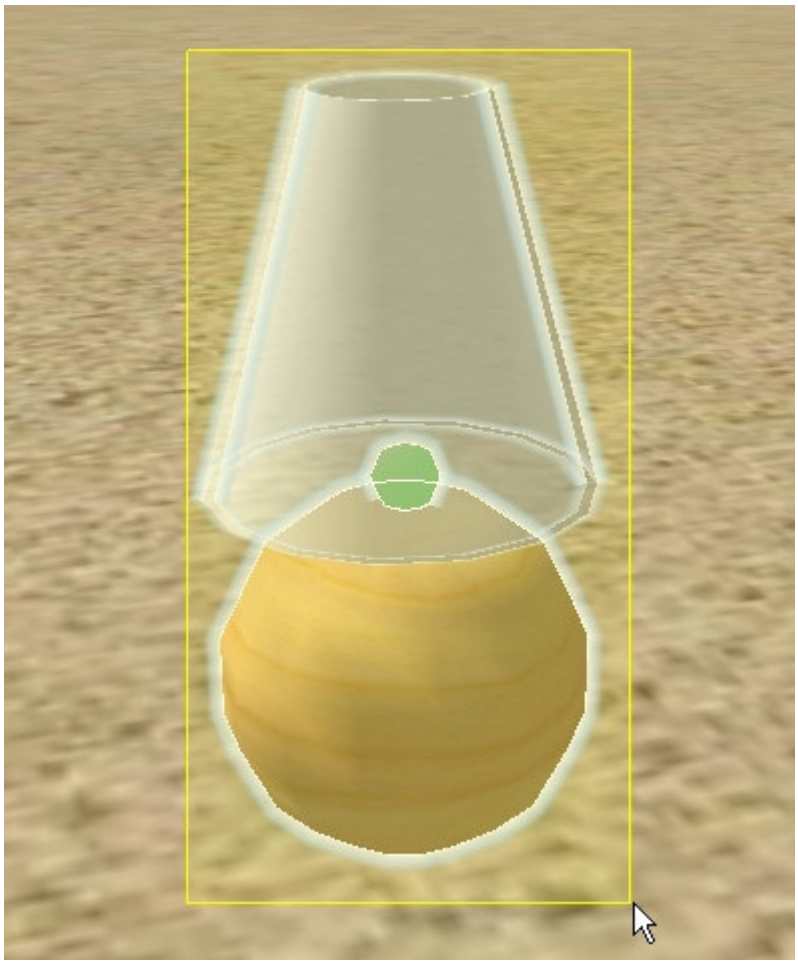


STEG5: MONTERING

MONTERING

For at lampen skal bli til en gjenstand må vi lenke (lime) objektene sammen.

1. Merk alle objektene dine ved å **holde inne venstre musetast** og dra en firkant rundt alle delene dine og **trykk Ctrl + L**



2. For å lagre lampen din til senere så **høyreklikk på lampen** og **velg Take**. Den vil da bli lagt til i inventaret ditt.

Gratulerer! Du er nå ferdig med å bygge lampen..

REFERENCES

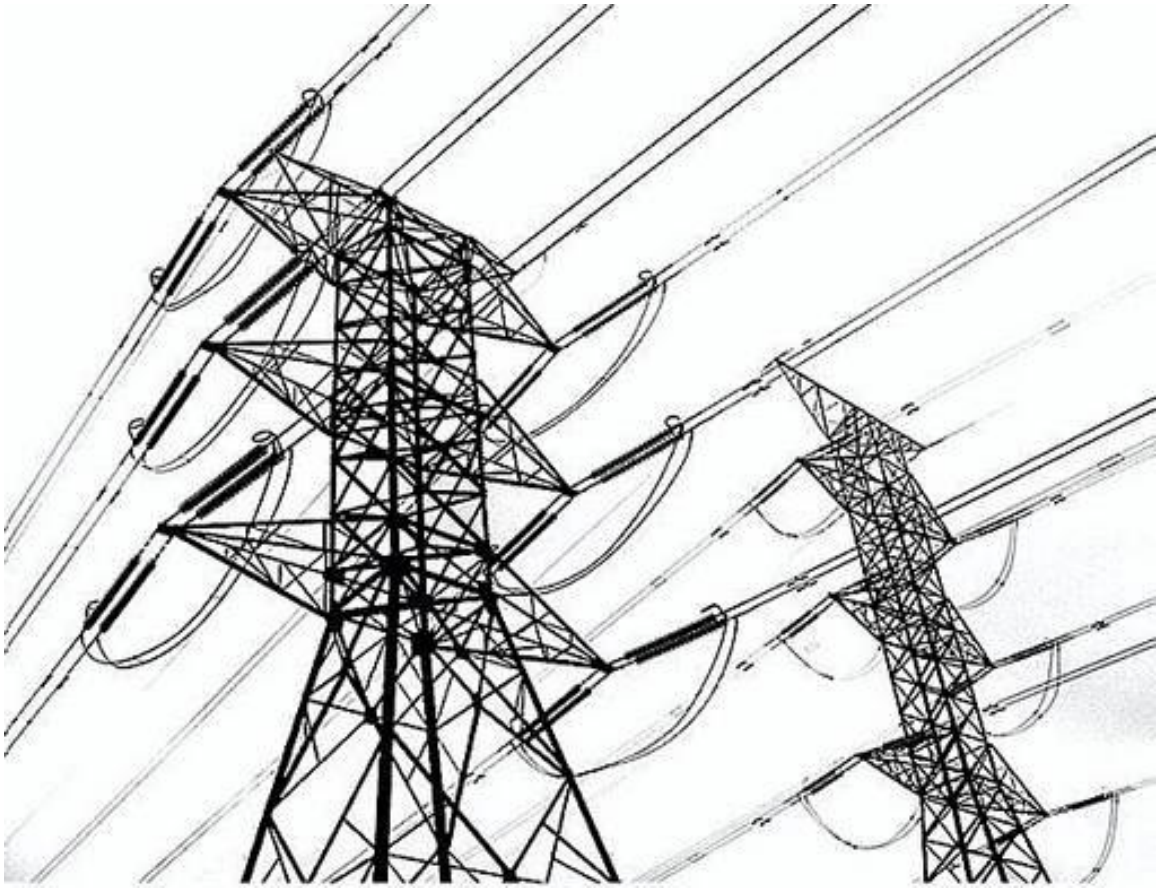
Second Life web site: http://secondlife.com

Du har lastet ned og installert second life. (http://secondlife.com/community/downloads.php)

Registrert en bruker i second life. (https://secure-web14.secondlife.com/join/index.php/Choose-Name?form%5Baction%5D=Skip+This+Step)

3. INF 245 - STRØMMÅLER

AV JAN ARNE JANSSEN



INTRODUKSJON

I dette prosjektet skal vi utvikle en strømmåler-applikasjon som er rettet mot mobile enheter. Målet med denne applikasjonen er at en bruker skal kunne registrere strømforbruket sitt, i tillegg til å kunne sende avlesningsresultatet til kraftleverandør. Applikasjonen skal gi brukeren mulighet til å få presentert strømforbruket sitt grafisk for en gitt periode. I tillegg skal applikasjonen kunne estimere det fremtidig strømforbruket til kunden, basert på foregående målinger. Applikasjonen skal også ha funksjonalitet for å hjelpe med å redusere strømforbruket.

Vi kommer først til å gå inn på planleggingen av applikasjonen og hva vi må ta hensyn til. Videre skal vi se på plattformen som vi skal utvikle applikasjonen i mot. Før vi i neste del tar for oss selve utviklingen av applikasjonen, der vi går inn på hvilke valg vi gjorde, og hvilke problemer som oppstod. Til slutt kjører vi en del tester mot applikasjonen for å finne feil og eventuelle forslag til forbedringer.

PLANLEGGING

2.1 SPESIFIKASJON

Prosjekt oppgaven har spesifisert en del senarier og funksjonalitet s0m bør være støttet av applikasjonen. Under kan du se disse punktene.

1. *The consumer wants to read his meter and send the reading to the power company. The consumer reads the meter and the meter identifier and enters them into a user friendly dialog on the device. and selects to send the reading. The device sends a correctly formatted sms-message to the power company.*

2. *The consumer wishes to see statistics for power consumption. The system shows statistics built upon all earlier readings for the meter as a graph.*

3. *The user wants to prepare for his power bill. The system shows what bill can be expected.*

4. *When trying to reduce consumption, the user wants to monitor his power consumption by taking frequent readings of the meter. The system supports this and shows total, hourly and daily power consumption since last reading. Statistics and graphics for the readings of the monitoring period should also be available.*

Når det kommer til det første punktet, så vil jeg slå det sammen med et av kravene fra punkt 4. Dette går da ut på at brukeren skal kunne ta målinger med hyppig frekvens for å kunne se sitt timelige, daglige og ukentlige forbruk. Som da skal hjelpe brukeren med å redusere forbruket sitt. Disse avlesningene er ikke ”offisielle” avlesninger, og skal derfor ikke sendes til kraftleverandør. Istedenfor å gi brukeren to forskjellige vindu for å utføre "offisielle” og ”uoffisielle” avlesninger, så har jeg tenkt å bygge disse to funksjonene inn et vindu, slik at brukeren slipper og forholde seg til forskjellige vindu for å gjøre avlesninger.

For det andre punktet må vi ta hensyn restriksjonene som en har med mobile enheter. I dette tilfellet så vil en stor restriksjon være størrelsen på skjermen. En graf kan bli ganske bred i forhold hvor stor periode vi vil se, og i høyden for hvor store svingninger det er i forbruket. Det er da viktig at brukeren skal slippe å ”scrolle” for mye for å se grafen. Er grafen veldig stor i både høyde og bredde, så må brukeren ”scrolle” både horisontalt og vertikalt, noe som gjør at brukeren ikke klarer å få et godt bilde av grafen fordi han mister oversikten.

Punkt 3 er et ganske greit punkt å gjennomføre. Det eneste en bør tenke på er hvor lang periode tilbake i tid en skal bruke som grunnlag for utregning av det fremtidige forbruket. Hvis en sier at en har en avlesning i månen, så bør en kanskje ikke bruke mer en de 2-3 siste månene som grunnlag. Grunnen til dette er at utregningen kan bli veldig unøyaktig, alt etter hvordan en regner ut det fremtidige forbruket.
Hvis en tar utgangspunktet i å bruke gjennomsnittlig forbruk for å regne ut det fremtidige forbruket, så kan det bli ganske unøyaktig. Dette er fordi strømforbruket som regel følger et fast mønster. Om høsten vil strømforbruket stige og holde seg høyt gjennom vinteren, og om våren vil det gå nedover og holde seg lavt gjennom sommeren. Hvis en da bruker en for stor periode som grunnlag, så vil en for eksempel når en kommer til høsten og vinteren når forbruket er på vei opp, få for lave estimat fordi forbruket for sommeren drar ned gjennomsnittet. Det samme vil da skje om våren, hvor en vil få for høye estimat.

For å gjøre estimatene mer nøyaktig, så har vi tenkt å ta hensyn til en del andre faktorer som har innvirkning å strømforbruket. De to tingene jeg vil ha med er forventet utendørs temperatur, og om det befinner seg en høytid i den fremtidige månen. I høytidene slik som for eksempel jul, så vet en at folk er mye innendørs og det blir generelt brukt mer strøm i denne perioden enn ellers på året. Dette skal da applikasjonen kunne ta hensyn til når den skal beregne det fremtidige forbruket.

Punkt 4 er en ganske grei oppgave å gjennomføre. Den delen som har med registrering av forbruk å gjøre vil bli innlemmet i punkt 1, slik at det som trengs å gjøres her, er å hente ut dataen og prosessere den. Den eneste utfordringen som er her, er å kunne presentere forbruket grafisk på en eller annen måte. Her tenkte vi da å vise noen enkle stolpediagram for å vise timelig, daglig og ukentlig forbruk.

2.2 PLATFORM

Når det kommer til valg av hvilken plattform vi skal utvikle imot, så har vi valget mellom Sun's Java Micro Edition og Microsoft's Compact .Net Framework. Begge disse utviklingsplattformene er "mini" versjoner av de større plattformene som blir brukt til desktop utvikling til vanlige datamaskiner og servere. Forskjellen fra de vanlige desktop plattformene og ned til de mobile plattformene, er at en bare har tatt med den mest nødvendige funksjonaliteten for å tilpasse bruken til mobile enheter som har en god del restriksjoner når det kommer til minne og prosessor kapasitet. Basisbibliotekene som følger med som standard, gjør at en kan starte og utvikle enkle applikasjoner uten store problemer. Men skal en lage applikasjoner der en må få tilgang direkte til noen av resurssene og funksjonaliteten til selve enheten som plattformen kjøres på, så må en installere så kalte SDKer (software development kits) som er tilpasset enheten og gir tilgang til resurser til systemet, slik som for eksempel kontaktlister og mulighet for å sende SMS direkte fra applikasjonene. Disse SDKene blir som regel gjort tilgjengelig av produsenten av enheten.

Jeg valgte å gå for .Net plattformen siden jeg har mest erfaring i å programmere i .Net. Jeg jobber til daglig med .Net uvikling mot desktop, men har jobbet veldig lite med utvikling mot mobile enheter. Dette vil da gi meg en verdifull erfaring som jeg ganske sikkert vil få bruk for på et senere tidspunkt. I tillegg ligger alle basis bibliotekene som trengs for mobil utvikling klar for bruk i Visual Studio. Noe som gjør det raskt og enkelt å starte med mobil utvikling. I Java utviklingsverktøy som for eksempel Netbeans, må en laste ned basisbibliotekene før en kan starte. Det er ikke spesielt vanskelig, men tar litt ekstra tid for å komme i gang med utviklingen.

2.3 APPLIKASJONSTYPE

En har tre muligheter når en skal velge hvilken applikasjonstype en skal utvikle. Den første muligheten er å lage en ren "stand-alone" applikasjon som kjører lokalt på enheten. Alt av input/output, prosessering og lagring av data skjer på selve enheten.
Skal en lage en avansert applikasjon som skal gjøre tunge beregninger eller som trenger mye minne og lagringsplass, så krever dette en kraftig og avansert enhet, som ofte er dyre og i mange tilfeller lite utbredt blant de store brukermasser. Noe som kan være negativt hvis applikasjonen er tenkt å være et produkt folk flest.

Den andre muligheten er å lage en webapplikasjon, der alt av prosessering skjer på webserveren. Den mobile enheten trenger bare å ha en webleser for å kunne bruke applikasjonen, noe som ikke krever veldig lite av enheten. Problemet en har her, er at serveren kan bli overbelastet hvis for mange brukere aksiserer websiden på samme tid, og i tillegg at prosesseringen av eventuelle data kan være tung. I tillegg er det restriksjoner på hvor avanserte brukergrensesnitt en kan lage på web i forhold til lokale applikasjoner.

Den tingen som er veldig positivt her, er at en bare trenger å oppdatere webapplikasjonen, så har alle brukere tilgjengelig nye versjoner uten å måte gjøre noe som helst. Har en lokale applikasjoner, så må brukeren drive å laste ned og installer nye versjoner etter hvert som de kommer ut.

Den siste muligheten er å dele applikasjonen i to, og bruke deler av de to metodene over.
En lager for eksempel en lokal applikasjon som tar seg av input og presentasjon av data. Så oppretter man en webtjeneste som den lokale applikasjonen kan koble seg til. En kan da for eksempel overføre data til webtjenesten, og la serveren ta seg av prosesseringen av dataen, før den blir sendt tilbake. Denne metoden gir mange muligheter for hvordan en kan sette opp applikasjoner, og fordele belastingen av prosessering. Det store problemet en slik løsning, er at en risikerer at den lokale applikasjonen blir ubrukelig når en oppdaterer webtjenesten, noe som gjør at brukere må laste ned ny versjoner av den lokale applikasjonen for å kunne bruke tjenesten.

Jeg valgte å gå for en lokal applikasjon siden det virker som den beste løsningen for meg. En kunne valgt å lage en webapplikasjon for å løse oppgaven. Men dette krever da at brukeren har mulighet for å komme seg på nett for å utføre oppgavene. I mange tilfeller kan dette bli dyrt hvis en ikke har mulighet for å koble seg opp via et trådløst LAN. Skal en bruke for eksempel 3G for å komme seg på nett, så må en som oftest betale for mengden data en mottar og sender. Det virker derfor unødvendig å gjøre dette, siden den eneste dataen en trenger å sende er forbruket, alt annet kan ligge lokalt på enheten. Forbruket sendes via SMS, og koster derfor veldig lite, og i tillegg så er det veldig få plasser en ikke har nok dekning til å kunne sende SMS. I motsetning til 3G som enda bare er bra tilgjengelig i og rundt større og mindre byer (4).

2.4 DATABASE

For å kunne lagre data på en strukturert måte på en mobil enhet, så trenger vi en database. Problemet er at vanlige databaser som blir brukt på vanlige datamaskiner og servere, er for ressurskrevende for å kunne bli kjørt på mobile enheter. Det må da løses ved å lage mobile databaser som bare har den mest nødvendige funksjonaliteten.
En annen løsning for å kunne lagre data med mobile enheter, er å lagre dataen på en ekstern server som kjører en vanlig database. Den mobile enheten kobler seg da opp mot serveren når den skal lagre eller hente data fra databasen. Problemet med dette er at enheten da må ha en stabil trådløs tilkobling, noe som ikke alltid er lett, hvis enheten skal brukes plasser der det er mange elementer som kan forstyrre tilkoblingen. Dette kan da føre til unødvendig venting for brukerens del, fordi han må vente eller lete etter sterkt nok signal for å få bli tilkoblet igjen, og kan hente og sende data igjen. I tillegg er muligheten tilstede for at data ikke blir overfør fullstendig, hvis tilkoblingen brytes midt under en overføring. Disse problemene blir løst ved å bruke en mobil database. Databasen ligger da lokalt på den mobile enheten, noe som gjør at brukeren alltid har den tilgjengelig.
I tillegg har mange av de mobile databasene støtte for synkronisering mot en ekstern database. Dette er fordi i mange tilfeller så vil en ha all data liggende på en sentral server, slik at en har all data tilgjengelig fra en plass. De to databasene vil da styre overføring av data i mellom seg når en tilkobling er tilgjengelig, uten at brukeren merker noe av det. I tillegg sørger de for at data blir overført korrekt selv om tilkoblingen skille mistes. (1)

Det finnes ikke et så stort utvalg av mobile databaser, men både Sybase, Oracle og Microsoft som er de største aktørene innen kommersielle databaser, tilbyr mobile løsninger av databasene sine (1). Henholdsvis Ultralite for Sybase, Oracle Lite for Oracle og Sql Server Compact for Microsoft. Compact server fra Microsoft er den enste av de tre som er gratis tilgjengelig for kommersiell bruk. Mens Sybase tilbyr gratis prøve versjoner av Sql Anywhere databasen sin, som inneholder Ultralite utgaven.

Jeg har valgt å bruke Sybase sin mobile database, først og fremst fordi jeg har brukt denne et par ganger før når jeg har drevet med mobile applikasjoner. Dette gjør da i utgangspunktet at jeg kan komme raskere i gang med utviklingen av applikasjonen.

GETTING STARTED

Jeg har utviklet mobile applikasjoner et par ganger før, men da bare med basisbibliotekene som følger Visual Studio. Applikasjonen av ikke spesielt avanserte og krevde heller ikke at en måtte akseresere ressursene til enheten. Disse gangene gikk det veldig raskt og enkelt å komme seg i gang med utviklingen, noe som jeg trodde det ville være denne gangen også. Det skulle uheldigvis vise seg å være god del være å komme i gang med utviklingen denne gangen av mange forskjellige årsaker.

3.1 UTVIKLINGS BIBLIOTEK

For å kunne få tilgang til enhetens ressurser for å for eksempel kunne sende SMS, så trenger en utviklingsbibliotek for plattformen som applikasjonen skal kjøre på (for eksempel Windows Mobile 5.0). Men dette var ikke noe jeg viste da jeg skulle starte å utvikle applikasjonen. Jeg tenkte at funksjonaliteten for å kunne sende SMS var noe som lå i basis bibliotekene. Grunnen til for dette er at jeg trodde at denne funksjonaliteten var lik i neste alle Windows Mobile versjoner og derfor ikke trengte egne utviklingsbibliotek. Da jeg ikke fant dette i basis bibliotekene, så startet jeg å lete på nettet etter andre folk som har utviklet mobile applikasjoner der de har brukt SMS funksjonalitet. Etter å ha brukt en del tid på å søke, så slo det meg hvor lite informasjon som fantes om mobil utvikling i forhold til .Net. Jeg vet at mobil utvikling er langt ifra så populært som vanlig desktop utvikling, men så få prosjekter som jeg fant offentlig tilgjengelig var nesten sjokkerende. Jeg fant utrolig nok et par prosjekter der en hadde tatt i bruk SMS funksjonalitet, men de brukte eksterne GMS modem med egne utviklingsbibliotek, noe som da ikke hjalp meg mye.
Etter å ha brukt en god del tid på å søke etter informasjon på nettet så fant jeg til slutt ut at jeg trengte noen utviklingsbibliotek for Windows Mobile 5.0, siden det var det operativ systemet jeg utviklet imot. Etter å ha funnet bibliotekene og installert dem, så måtte jeg finne informasjon om hvordan jeg skulle bruke dem, og spesifikt finne ut hvordan jeg kunne sende SMS. Det klarte jeg også etter en stund å finne, men det som jeg synes er horribelt er at alt dette fant jeg i forumet på MSDN siden til Microsoft (10), av vanlige folk som meg, som ikke har tilknytting til Microsoft i det hele tatt. Jeg fant så å si ingen ting om dette på Microsoft sine sider, noe jeg føler er ekstremt dårlig fra Microsoft sin side, som tilbyr å kunne lage mobile løsninger. Akkurat på dette punktet synes jeg Java har kommet mye lenger, siden det her finnes mye hjelp og tutorials i forhold til mobil utvikling.

3.2 DATABASE OG ”DLL HELL”

Jeg valgte som sagt å bruke Ultralite databasen i Sql Anywhere fra Sybase i dette prosjektet. For å kunne koble seg til databasen fra .Net, så må en importer inn to Dll filer som følger med installasjonen av Sql Anywhere. Problemet som oppstod her, var at det var feil mellom versjonene på de to Dll filene. Du merket ikke denne feilen før du prøvde å koble deg til databasen når du startet applikasjonen, da du bare fikk melding om at applikasjonen ikke kunne koble seg til databasen. På grunn av dette gikk lenge før jeg klarte å finne ut at det var problemer mellom de to Dll filene. Måten jeg fant ut av dette på, var ved å sammenligne versjonene av disse filene i et tidligere prosjekt der jeg hadde brukt filene og tilkoblingen til databasen fungerte som den skulle. Men i det prosjektet hadde jeg brukt en eldre versjon av Sql Anywhere enn den jeg brukte nå. I den nyere versjonen som jeg brukte nå, så var en av Dll filene oppdatert med en nyere versjon, mens den andre ikke var oppdatert. Jeg tenkte da at jeg kunne løse problemet ved å importere de gamle Dll filene, istedenfor de nye. Men dette ville ikke Visual Studio god ta, hver gang jeg prøvde å importere den gamle filene, ble de erstattet av de nye filene. Etter en god stund med prøving og feiling uten suksess, gav jeg opp på å

prøve å løse problemet. Isteden omgikk jeg problemet ved å kopiere hele det gamle prosjektet som jeg hadde laget. Jeg fjernet alle ting som jeg ikke trengte, og bygde strømmåler applikasjonen på denne grunnmuren i stedet. Dette fungerte helt fint, det eneste som du kan se er at prosjekt mappen og en del andre filer har navnet til den gamle applikasjonen. Ellers fungerte alt som det skal, uten problemer, men dette viser da hvordan "Dll Hell" (7)(8)(9) kan oppstå når du legger inn nye versjoner av en programvare.

3.3 EMULATOR OG DEBUGGING

En har to muligheter for å debugge applikasjonen. Den første og beste er å ha en mobil enhet tilkoblet datamaskinen via ActivSync, som er synkroniserings programmet som blir brukt sammen med Windows Mobile operativsystemer, for å synkronisere data. Når en starter debugging av applikasjonen fra Visual Studio, så blir applikasjonen overført direkte til den mobile enheten og startet der. En kan da bruke en applikasjonene på enheten, noe som da gir et optimalt bilde på hvordan applikasjonen fungerer i det virkelige liv.
Den andre muligheten er å bruke emulatorer for å debugge applikasjonen lokalt på datamaskinen. Fordelen med dette er at startingen av debuggingen går mye raskere siden en ikke trenger å overføre applikasjonen til en annen enhet. Problemet er at du ikke kan kjøre tester på en del funksjonalitet som du bygger inn i applikasjonen. I mitt tilfelle er SMS funksjonaliteten en slik ting som jeg ikke får testet, siden jeg ikke har noe mobil enhet som kjører Windows Mobile. Jeg mener at dette er for dårlig av Microsoft å ikke legge inn noen form for emulering for å sende SMS.

UTVIKLING

Når jeg et mye strev klarte å finne ut av oppstartsproblemene som jeg hadde, så gi selve utviklingen av applikasjonene ganske greit. Oppgaven førte med seg en fin utfordring i for hold til presentasjon av grafer for forbruket. Du har ingen ferdige komponenter i .Net som kan utføre denne funksjonaliteten, noe som betyr at en må lage til denne funksjonaliteten helt i fra grunnen av ved å bruke "Drawing" klassen til .Net. Denne utfordringen var ganske lærerik og spennende, siden denne funksjonaliteten sjelden blir brukt. Når en lager vanlige desktop applikasjoner, så bruker en som regel ferdige komponenter i .Net eller fra 3 parts leverandører for å kunne tegne og presentere data grafisk.

Jeg valgte å dele applikasjonen opp i 4 deler som hver inneholdt funksjonaliteten som ble etterspurt i spesifikasjonen. Oppdelingen og navngivingen kan du se på bildet under, som viser oppstartsvinduet for applikasjonene.

Videre skal vi gå detaljert inn på utviklingen av de forskjellige delen, hvor jeg skal gå inn på hvordan jeg løste oppgavene og hvilke problem jeg møtte på i forbindelse med utviklingen.

4.1 AVLESNING

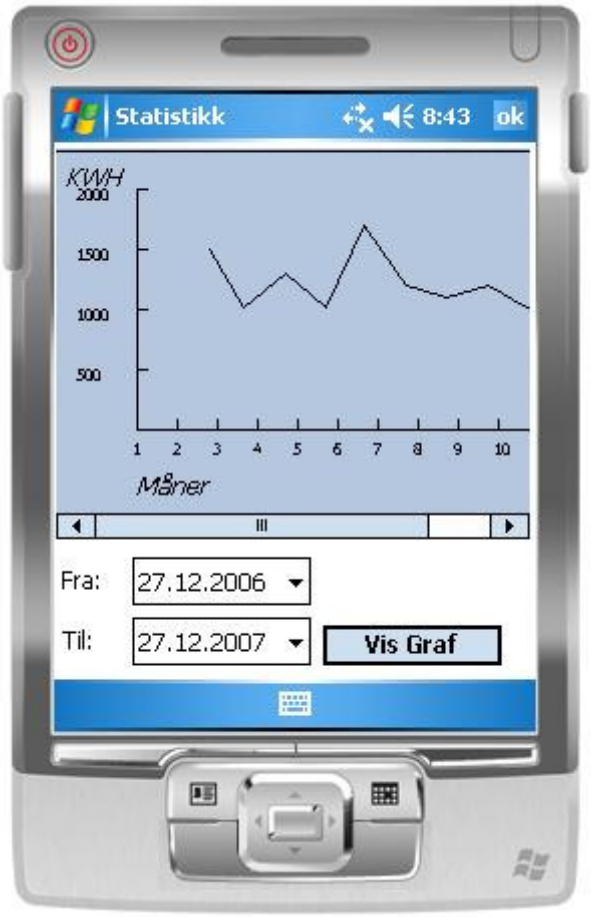
I denne delen av applikasjonen bestemte jeg meg for å legge inn funksjonalitet for å kunne registrere og sende ”offisielle” avlesninger til kraftleverandøren. I tillegg til å kunne ta hyppige avlesninger for å ha kontroll på strømforbruket, og ut ifra det ha mulighet for å redusere forbruket.
Brukeren velger om han bare vil lagre forbruket, eller om han både vil lagre og sende det, altså en ”offisiell” avlesning. Feltet for målernummeret blir aktivert hvis det er en offisiell måling, siden det skal sendes til kraftleverandøren sammen med forbruket.

Avlesningen blir lagret i en ganske enkel tabell i databasen, der det blir spesifisert en unik ID for avlesningen, dato og klokkeslett for avlesningen antall kWh som er brukt og bit verdi (0 eller 1) om det er en ”offisiell” avlesning. En del ville vell kanskje sagt at den unike Iden er overflødig, når en kan bruke dato og tidspunktet for avlesningen som en ”primary key” i tabellen, siden en aldri kan ta flere avlesninger på nøyaktig samme tidspunkt. Men jeg er en motstander av å bruke dato og tidspunkt som ”primary key” i tabeller. Grunnen til dette er at en kan få problemer når en henter dataene inn i .Net og bearbeide dataene i koden, for å så kjøre en UPDATE eller DELETE SQL settning mot tabellen på en spesifikk rad. For å finne den spesifikke raden, så må du spesifisere ”primary key” i SQL serningene. Problemet her oppstår når du henter dato og tidspunktet inn i .Net, og legger det i en dato variabel. Det kan da oppstå at .Net legger til eller trekker fra noen sekunder av tidspunktet, noe som da gjør at dato og tidspunket ikke reflekterer det opprinnelige tidspunktet som ble tatt inn. Dette fører da til at en ikke klarer å finne ”primary key” når en skal slette eller oppdatere

en spesifikk rad. Dette problemet oppstår ikke når hvis en for eksempel legger dato og tidspunktet inn i en "string" variabel, fordi verdien da blir lagt direkte inn i variabelen slik som den er. Skal en legge variabelen inn i en dato variabel, så kjører .Net en eller annen form for kalkulering, som gjør at det kan oppstå veldig små variasjoner. Dette har jeg opplevd en del ganger, og er derfor ganske forsiktig med å bruke dato og tidspunkt som "primary key".

Når det kommer til selve SMSen som skal sendes, så tok jeg å separerte den med semikolon, slik at for eksempel strømleverandøren kan ta imot SMSen og bearbeide den automatisk ved å dele opp i det forskjellige feltene som kommer i SMSen.

4.2 STATISTIKK



Statistikk delen av applikasjonen inneholder funksjonaliteten for å kunne vise forbruket grafisk gjennom en graf. Dette var helt klart den mest krevende delen av oppgaven, siden jeg måtte programmere dette helt i fra bunnen av. Den andre ting som gjorde det krevende er den begrensede størrelsen på skjermen på enheten. Dette betyr at en hele tiden må tenke på å ikke vise for mye, som gjør at brukeren mister oversikten. Dette er spesielt vanskelig med en graf, hvor både høyde og bredde kan endre seg.
For å løs denne utfordringen, valgte jeg å sette en fast høyde på grafen, der jeg tilpasset verdiene i forhold til det høyeste forbruket for en valgt periode. Når jeg fant det høyeste forbruket, rundet jeg

det opp til nærmeste 100 eller 1000 del, og brukte den verdien som topp verdi på Y aksen. Så delte jeg verdien likt ned over aksen, slik at jeg totalt hadde 4 verdier på aksen. På denne måten sikret jeg hele tiden at grafen aldri ble høyere, men alltid reflekterte det riktige forbruket.
Når det kom til bredden på grafen, så laget jeg den slik at den dynamisk tilpasset seg størrelsen på perioden som var valgt, der hver måne har en fast størrelse. Jeg valgt bredden på hver måne, slik at en nesten kunne se et helt år på skjermen uten å måtte "scrolle". Med en så bredd oversikt, så vil det være vanskelig for brukeren å miste oversikten, selv om han må "scrolle" for å se hele grafen for en veldig lang periode.
Jeg plasserte grafen inne i tillegg i et eget "panel", slik at grafen utvider seg inne i panelet og ikke i selve vinduet. Poenget med dette er da at en "scroller" panelet hvis grafen blir for bred, og ikke på selve vinduet. Dette betyr at brukeren alltid har tilgjengelig kontrollene nederst i vinduet. Hadde en ikke brukt panel, så måtte brukeren ha "scrollet" hele vinduet, noe som gjør at brukeren må "scrolle" tilbake igjen for å se kontrollene.

For å kunne tegne grafen bruker jeg et drawin objekt der jeg kan spesifiserer høyde og bredde.
Det første jeg gjør når jeg skal opprette grafen, er å spesifiserer en fast "marg" fra venstre side, og fra bunnen og opp. Disse størrelsene sier da hvor linjen på X og Y aksen skal ligge i forhold til side og bunn. Når dette er gjort, så tegner jeg på alle de små strekene og tallene, i tillegg til å tilpasse bredden på X aksen i forhold til perioden som er valgt. Dette gir da grunnmuren hvor jeg kan tegne inn grafen. Det å regne ut punktene hvor grafen skal tegnes imellom, er en litt vanskelig sak. Grunnen til dette er at en i drawing objektet ikke regner fra bunnen og opp, men fra toppen og ned, noe som gjør tingen mye mer innviklet.
Det første som skjer etter en har funnet forbruket, er at en må sammenligne det med topp verdien som er valgt for Y aksen, for å finne den prosentvise forskjellen. Denne prosentvise forskjellen brukes da til å finne hvor punktet skal ligge på Y aksen, i forhold til hvor høy aksen er i piksler. Denne verdien må beregnes i fra toppen og ned, og ikke fra bunnen og opp. Dette vil da gi oss punktet på Y aksen. Å finne hvor en skal plassere tidspunktet på X aksen, er litt lettere, siden en da regner fra siden og innover. En tar først og finner det totale antall dager for perioden. Så tar en og finner antall dager fra starten av perioden til tidspunktet for målingen. En kan da finne den prosentvise forskjellen på hvor tidspunktet ligger i forhold til perioden. Denne prosenten bruker en da på antall piksler for X aksen, noe som da gir oss et punkt på X aksen. En har nå et punkt for Y aksen, og et punkt for X aksen, som da gir oss det første punktet i grafen.
Når en har funnet alle punktene, kan en tegne opp grafen i sin helhet. Noe som en kan se et eksempel på i bildet øverst i dette avsnittet.

4.3 PLANLEGGER

"Planleggeren" gir brukeren mulighet til å finne ut noe lunde hva det fremtidige strømforbruket kommer til å bli. Jeg tar her utgangspunkt i strømforbruket for de tre siste månene, når jeg skal regne ut det fremtidige forbruket. Dette gjør da at jeg fanger opp faste mønster i forbruket, som da skjer om høsten og våren når forbruket går opp og ned. Dette gjør da at jeg kan gi en mer nøyaktig beregning. Måten jeg beregner forbruket, er å finne den gjennomsnittlige differansen mellom de registrerte forbrukene. Denne verdien legger jeg da til sist registrert forbruk.

Jeg har i tillegg lagt inn 2 ekstra funksjoner som skal gjøre det mulig å få mer nøyaktige beregninger. Den første er at brukeren kan oppgi en gjennomsnittlig temperatur som han tror det vil være for den fremtidige perioden. Programmet tar da hensyn til denne temperaturen, og legger til en vis prosent på forbruket, i forhold til hvor kalt det er. Jeg har spesifisert at det skal legges til 0.5 % for hver grade under 8 grader. Dette ligger ikke noe vitenskapelig utforskning bak disse verdiene, det skal bare vise mulighetene for det. Den andre tingen jeg har lagt til er muligheten for å velge om brukeren vil ta hensyn til høytider. En vet at i høytide som jul og påske, så brukes det mer strøm fordi folk er mer innendørs, i tillegg til at flere folk er hjemme. Og i tillegg må vi se på alle lysdekorasjonene som mange folk bruker i julen. Derfor la jeg inn mulighet for å ta hensyn til dette slik at beregningen kunne bli enda mer nøyaktig. Jeg har lagt inn jul og påske som de to høytidene som det blir tatt hensyn til. Programmet vil sjekke om den neste månen er desember eller april. Hvis det er desember,

så vil det bli lagt til 10 prosent på forbruket, mens det i april vil bli lagt til 5 prosent. Bak disse tallene er det heller ikke noe vitenskapelig, det er bare "proof of concept".

Brukeren får i tillegg lov til å skrive inn hva kWh prisen er, i tillegg til nettleie. Dette gjør da at brukeren får se et estimat av hva forbruket kan komme på, og hva prisen vil bli.

4.4 STRØMSPARING

Strømsparingsdelen skal gi brukeren mulighet til å kunne ta hyppige målingen, for så å se et detaljert bilde av forbruket sitt. Dette skal da hjelpe brukeren med å redusere strømforbruket sitt.
Brukeren skal få presentert timelig, daglig og ukentlig forbruk, i tillegg til en grafisk visning av dette forbruket.

For å finne dette forbruket, så tar jeg først og finner sist "offisielle" måling, slik at jeg har et startpunkt å forholde meg til. Så henter jeg ut alle uoffisielle målinger etter dette tidspunktet, og finner det totale forbruket. Dette forbruket splitter jeg da opp i timelig, daglig og ukentlig forbruk. Dette presenteres da først i tekstbokser. Planen var så å opprette en eller annen for søylediagram for å vise dette grafisk. Men på grunn av tidsnød kom jeg meg dessverre ikke så langt at jeg fikk implementert dette.

4.5 TESTING

Jeg hadde i utgangspunktet planlagt å få testet dette på en virkelig mobil enhet, slik at jeg fikk teste slike ting som å sende SMS. Men på grunn av at jeg ikke eier noen enhet som kjører Windows Mobile 5.0, eller har noen bekjente i omgangskretsen min som er villig til å låne bort enheten sin, så ble det dårlig meg virkelig testing. All testingen gikk derfor for seg i emulatorene til Visual Studio. Disse testene ble ganske enkelt gjennomført "on the run" mens jeg programmerte. Dette er fordi det ikke er spesielt mye kode som ligger bak applikasjonen, og det heller ikke mange plasser med input fra bruker som kan gi feil. Dette gjør da at jeg hadde ganske god kontroll over feil som kunne oppstå i selve koden. Noe som fører til at det nesten er overflødig og kjøre store tester på input og output verdier.
Den plassen det virkelig lønner seg å kjøre testing på, er selve brukergrensesnittet. Mobile enheter har som sagt en stor restriksjon i forhold til skjerm størrelse, og derfor bør skjermbildene være så enkle og lettfatelige som mulig. Dette hadde vært en interessant test og gjennomføre med virkelige brukere. Men på grunn av tidsnød og mangel på en test enhet, så ble det ikke foretatt noen større tester.

KONKLUSJON

Vi har i dette prosjektet sett på hvilke restriksjoner en har på mobile enheter, og hva en trenger for å kunne utvikle applikasjoner for disse enhetene. Det oppstod en del oppstartsproblemer, som vi etter hvert fikk løst på vanlig og litt mer ukonvensjonelle måter. Noe som viser at en ofte ikke kan tenke rett frem når en skal drive med utvikling, særlig når det gjelder utvikling mot en teknologi som langt i fra har samme utbredelse blant utviklere, som for eksempel desktop utvikling. Dette fører da til at en må tenke mye mer selv, og som oftest kan en ikke regne med hjelp fra andre steder. I forhold til .Net så er det stor mangel av ressurser på nettet. Også fra Microsoft sin side er det stor mangel på hjelpende ressurser. Noe som er ganske dårlig når det er dem som tilbyr muligheten for utviklingen av mobile applikasjoner. Under utviklingsdelen fikk vi se litt på hvordan en kunne gjøre små forbedringer i grensesnittet for å motvirke restriksjonen som mobile enheter blant annet har i forhold til skjermen.
Det ble dårlig med testing på grunn av mangel på en test enhet i tillegg til tidsnød. Men utviklingsdelen gav meg verdifull erfaring, og oppstartsproblemene lærte meg at jeg aldri bør gi opp på å løse problemet, uansett hvilken løsning det skulle være.

REFERANSER

1. **Wikipedia: Mobile Databases**
   http://en.wikipedia.org/wiki/Mobile_database
2. **Sql Server Compact**
   http://www.microsoft.com/sql/editions/compact/default.mspx
3. **Wikipedia: Utvikling av software mot mobile enheter**
   http://en.wikipedia.org/wiki/Windows_Mobile#Software_development
4. **Windows Mobile hjemside:**
   http://www.microsoft.com/windowsmobile/default.mspx
5. **Dekning 3G Norge**
   http://www.telenor.no/privat/mobil/dekning/dekningskart/
6. **Microsoft: Compact Framework**
   http://msdn2.microsoft.com/nb-no/netframework/aa497273(en-us).aspx
7. **Wikipedia: Dll Hell**
   http://en.wikipedia.org/wiki/DLL_hell
8. **SSW: Dll Hell**

http://www.ssw.com.au/ssw/Database/DLLHell.aspx
9. **Dr. Bobbs Portal: Dll Hell**
http://www.ddj.com/windows/184416837
10. **MSDN Forum**
http://forums.microsoft.com/msdn/default.aspx?siteid=1
11. Sybase hjemside
http://www.sybase.com/