

Utveckling av en visualiseringsapplikation för solinstrålningsdata

Niklas Krave

Civilingenjörsutbildningen i Lantmäteri
Lunds Tekniska Högskola

Institutionen för Naturgeografi och Ekosystemvetenskap
Lunds Universitet





LUNDS UNIVERSITET
Lunds Tekniska Högskola

Utveckling av en visualiseringsapplikation för solinstrålningsdata

EXTM05 Master uppsats, 30 hp

Civilingenjörsutbildningen i Lantmäteri

Författare:
Niklas Krave

Handledare:
Lars Harrie

Institutionen för Naturgeografi och Ekosystemvetenskaper

Februari 14, 2015

The development of an application for visualization of solar radiation data

Opponent: Magdalena Nyberg

Examinator: Petter Pilesjö och Stefan Olin

Copyright © Niklas Krave, LTH

Institutionen för Naturgeografi och Ekosystemvetenskaper
Lunds Universitet
Sölvegatan 12
223 62 Lund

Telefon: 046-222 30 30
Fax: 046-222 03 21
Hemsida: <http://www.nateko.lu.se>

Examensarbete i geografisk informationsteknik nr 14
Tryckt av E-tryck, E-huset, 2015

Förord

Detta examensarbete är ett avslutande moment i Civilingenjörsutbildningen Lantmäteri med inriktning mot geografisk informationsteknik. Arbetet är utfört för Lunds tekniska högskola i samarbete med Göteborgs universitet och Tyréns AB.

Ett stort tack riktas till min handledare Lars Harrie som har varit ett stöd och till stor hjälp under hela processen. Jag vill också tacka Fredrik Lindberg som bidragit med stöd under utvecklingen och har gett mig inblickar i hur solinstrålningsmodeller som SEBE fungerar. Tack ska också Tyréns AB ha för att ha bidragit med en välkomnande miljö och möjligheten att utnyttja deras lokaler. På Tyréns AB riktas ett särskilt tack till Dag Wästberg vars tekniska kunskap både har varit en inspiration och ett nödvändigt stöd för att genomföra arbetet.

Ett mer personligt tack riktas till min familj och framför allt min far som varit ett stöd under hela min utbildning och är en av anledningarna till att jag startade mina studier i Lantmäteri.

2014-12-01

Niklas Krave

Abstract

Increasing demand for ways to make use of renewable energy, has led to a greater need for methods to estimate the optimal position of energy system that make use of solar power. Systems for solar analysis make it possible to estimate incoming solar radiation on surfaces based on the structure of landscapes, buildings and vegetation. To make use of the data derived from solar analysis, there is a user need for applications that summarize and present the data in a meaningful way. It is possible to create an interactive environment through an application for visualization of 3D-models. This interactive environment can be used to create custom functions that are geared towards creating uses for incoming solar radiation.

An application for such visualization would function through the use of data on incoming solar radiation, a user interface and a technique for 3D-visualization. These components need to be integrated in working application architecture. Systems for solar analysis generate large amounts of data. This creates a need for solutions that allows the visualization application to function without performance related problems. Furthermore, a useful application has to be able to create a link between the data from solar analysis to existing structures such as buildings.

SEBE (Solar Energy on Building Envelopes) is a solar analysis system developed in cooperation between Gothenburg University and the consulting firm Tyréns AB. Data produced by this system were used in the case study of this project. The goal of the project was to investigate what possibilities exists for creating an application for visualization based on the data produced by SEBE. Through user analysis a set of user needs and technical requirements was developed for the application. Through research into what similar applications exist and how they functions a set of possible architectures were identified. The architecture thought to be most fitting for the application was identified, and used to create a prototype of the application. As part of the project the prototype was evaluated.

A plugin program, titled SEBEvisualizer was developed for QGIS with support for 3D-models created in OpenGL. This was the platform determined most fitting for development of an application for visualization. SEBEvisualizer makes use of vector layers, a database, the QGIS API and the Qt framework to handle and visualize incoming solar radiation. *Voxels*, or volume pixels, make it possible to create 3D-models of buildings and other complex objects without the need for any kind of sorting algorithms. *Voxel* based methods have drawbacks on performance but are easy to implement correctly.

The conclusions made from the project are that it is possible to create a useable application for visualization. There are problems with meeting the needs of all kinds of users in a single application. The development platform consisting of QGIS and OpenGL has benefits pertaining to the handling of geographic data and good performance. Drawbacks are that users have a harder time finding and using the application. With future progress in web based 3D-modeling, through supports such as WebGL, an application developed for the web might be more fitting to reach a broader user base.

Keywords: Geographical Information Systems, QGIS, plugin, solar irradiance, 3D, voxels.

Sammanfattning

Med en ökande efterfrågan på källor för förnyelsebar energi, däribland solenergi, ökar behovet av att utreda lämplig placering av solenergisystem. Genom solinstrålningsmodeller är det möjligt att beräkna solinstrålning på ytor utifrån ytmodeller av landskap, byggnader och vegetation. För att användare ska kunna ta del av informationen från en solinstrålningsmodell, behövs applikationer som sammanfattar och presenterar informationen. Genom en visualiseringsapplikation som skapar 3D-modeller av till exempel byggnader går det att skapa en interaktiv användarmiljö. Interaktiva miljöer tillåter att många typer av skräddarsydda funktioner skapas för att presentera solinstrålningsdata på ett så användbart sätt som möjligt.

För att kunna skapa en visualiseringsapplikation krävs det data om solinstrålningens intensitet, ett gränssnitt för att kommunicera med användaren och en visualiseringsteknik för att skapa en 3D-modell. Komponenterna måste sammanfogas genom en applikationsarkitektur som har stöd för alla delar. Solinstrålningsmodeller genererar stora mängder data. Det leder till frågeställningar om hur en visualiseringsapplikation kan tas fram för att hantera data utan att lösningen blir för långsam. För en användbar applikation krävs också att datan kan sammankopplas med objekt såsom byggnader.

SEBE (Solar Energy from Building Envelopes) är en solinstrålningsmodell framtagen i ett samarbete mellan Göteborgs universitet och konsultbolaget Tyréns AB. Data från modellen användes i fallstudien inom detta projekt. Målet med projektet var att undersöka vilka möjligheter som finns att ta fram en visualiseringsapplikation för solinstrålning baserad på data från SEBE. Genom att identifiera potentiella användare och utifrån en analys av deras behov formulera tekniska krav har en kravspecifikation tagits fram. Genom en litteraturstudie har liknande applikationer och möjliga applikationsarkitekturer undersökts. Av de identifierade applikationsarkitekturerna valdes den som verkade mest lämpad för utvecklingen av en prototyp. Som en del av projektet har applikationens prototyp därefter utvärderats.

Ett insticksprogram till QGIS med en 3D-modell framtagen i OpenGL beslöts vara den bästa plattformen för att skapa en visualiseringsapplikation. Visualiseringsapplikationen gavs arbetsnamnet SEBEvisualizer. SEBEvisualizer använder sig av vektorlager, en databas, QGIS kodbibliotek och applikationsramverket Qt för att utföra arbetet från hantering till visualisering av solinstrålningsdata. *Voxlar* eller volym-pixlar tillåter att man skapar 3D-modeller av byggnader och andra komplexa objekt utan att behöva lösa problem såsom sortering av indata. Metoden är prestandakrävande men gör det möjligt att enkelt skapa en automatiserad process avsedd att rendera miljöer på ett korrekt sätt.

Slutsatserna av projektet är att det är möjligt att skapa en användbar visualiseringsapplikation. Dock är det i nuläget svårt att uppfylla alla användares behov genom enbart en applikation. Den valda utvecklingsplattformen genom QGIS och OpenGL har fördelen att den underlättar arbetet med geografiska data och har god prestanda. Nackdelar är bland annat att plattformen försvårar spridning och användning av applikationen. Med framtida framsteg av stöd för utveckling av 3D-grafik på webben som WebGL, kan förutses att en webbapplikation blir bättre lämpad för att nå bredare användargrupper.

Nyckelord: Geografiska informationssystem, QGIS, insticksprogram, solinstrålning, 3D, voxlar.

Innehållsförteckning

1. Inledning	1
1.1 Bakgrund	1
1.2 Problemformulering	1
1.3 Grundidén med applikationen	2
1.4 Syfte	3
1.5 Metod	3
1.6 Rapportdisposition	4
2. Användarkrav	5
2.1 Behovsanalys	5
2.2 Kravspecifikation	9
3. Solinstrålning	10
3.1 Solinstrålningsteori	10
3.2 Solinstrålningsmodeller	11
4. Applikationsarkitektur	17
4.1 Webbapplikation	17
4.2 Insticksprogram	21
4.3 Fristående applikation	23
5. Visualiseringstekniker	25
5.1 OpenGL	25
5.2 WebGL	30
5.3 PyOpenGL	30
6. Val av teknisk miljö	31
6.1 Motivering	31
6.2 Beskrivning av valt ramverk	32
7. SEBEvisualizer	37
7.1 Applikationens arkitektur	37
7.2 Visualiseringsapplikationens indata	38
7.3 Grafiskt användargränssnitt	44
7.4 Geografisk miljö	44
7.5 Markeringsverktyg	45
7.6 Statistik	46
7.7 3D-visualisering	48
7.8 Licenser	53

8. Resultat.....	54
8.1 Avstämning mot kravspecifikation	54
9. Diskussion.....	56
9.1 Diskussion av prototyp	56
9.2 Framtida utveckling av SEBEvisualizer	58
9.3 Webblösning.....	62
9.4 Möjligheter för annan användning.....	62
10. Slutsatser	63
Källförteckning.....	64
Artiklar	64
Böcker.....	64
Rapporter	64
Webbsidor	65
APPENDIX A – Användarmanual.....	68
APPENDIX B & C – Källkod	72

Ordlista

API – *Application Programming Interface*. Ett API är en specifikation av hur olika applikationsprogram kan använda och kommunicera med en specifik programvara.

ASCII – *American Standard Code for Information Interchange*. En teckenkodning som representerar bokstäver och andra tecken i en dator.

ASCII-grid – En filtyp som bygger upp en variant av rasterbilder.

Azimut – Beskriver avvikandet från ett väderstreck. Den mäts i grader utifrån en given referenspunkt som till exempel söderläge.

GIS – Geografiskt informationssystem. Ett datorbaserat system som används för att hantera geografiska data.

Header – En typ av metadata beskrivning för olika filtyper. Metadata är en typ av data som beskriver data.

MATLAB – Ett programmeringsspråk för numerisk analys, visualisering och utveckling.

OpenGL – *Open Graphics Library*. Ett API för att skriva grafiska applikationer i två eller tre dimensioner.

PyOpenGL – En samling bindningar till OpenGL som tillåter att applikationer utvecklas i Python.

PyQt – En samling bindningar till Qt (se nedan) som tillåter att applikationer utvecklas i Python.

Python – Ett programmeringsspråk med fokus på enkelhet och integration mellan system.

Qt – Ett ramverk för utveckling av användargränssnitt i C++.

QGIS – Ett GIS framtaget genom Qt och distribuerat under öppen källkod.

SEBE - *Solar Energy from Building Envelopes*. En solinstrålningsmodell som uppskattar solpotential utifrån geografiska positioner och meteorologiska data.

Solceller – En cell som tar emot och omvandlar solinstrålning till ström. Celler seriekopplas för att bilda solpaneler.

Solenergisystem – De tekniska system som på något sätt drar nytta av solinstrålning.

Solelsystem – De tekniska system som på något sätt drar nytta av solinstrålning för att generera ström. Exempelvis solceller.

Solfångare – En teknik som tar emot värmen från solinstrålning och utnyttjar den till uppvärmning av exempelvis lokaler.

Solinstrålningsmodell – En automatisk process som approximerar mängden solinstrålning mot till exempel ytor i ett område.

Solvärmesystem – De tekniska system som på något sätt drar nytta av solinstrålning för att generera värme. Exempelvis solpaneler.

TIFF – *Tagged Image File Format*. En filtyp som bygger upp en variant av rasterbilder.

Zenit – Den tänkta punkt på himlavalvet som befinner sig rakt ovanför betraktaren.

1. Inledning

1.1 Bakgrund

Det finns en ökande efterfrågan på förnybar energi som solenergi (Energimyndigheten, 2012a). Med ökad efterfrågan och förbättringar inom soletstekniken ökar behovet av att utreda byggnaders solinstrålningsnivåer. Det kan röra sig om alltifrån att göra en ekonomisk vinst på energin från en byggnads solinstrålning till att minska miljöpåverkan från uppvärmning av bostäder och lokaler.

För att optimera användningen av solceller krävs undersökningar om lämpligaste placering. Undersökningarna kan underlättas genom ett verktyg som snabbt kan ta fram specifikt underlag för byggnader. Underlaget kan sedan användas för att utesluta olämpliga områden eller välja ut de ytor som man bör fokusera på. Användningen av information om inkommande solenergi kan också utvidgas till andra användningsområden såsom beräkning av inkommande värme och fönsterplaceringar. Det finns ett tydligt behov av en automatiserad process som kan hantera data för solinstrålning och presentera informationen för att möta olika tänkbara behov. För detta krävs en metod för att få fram data för solinstrålningen på ytor och sedan en applikation för att kategorisera, sammanfatta och visualisera datan.

En metod för att få fram värden för solinstrålning i ett område är att använda sig av en solinstrålningsmodell. SEBE (Solar Energy on Building Envelopes) är en solinstrålningsmodell som användes i fallstudien inom detta projekt. SEBE använder sig av ytmodeller för att uppskatta mätningar av solinstrålning på tak, mark och husfasader. Dessa data framställs i rasterformat där varje mätvärde representerar instrålningen inom en bestämd area kallad en cell.

Målet med projektet är att utveckla en applikation för hantering och visualisering av utdata som skapats av solinstrålningsmodellen. Solinstrålningsdata från modellen kan genom algoritmer kopplas till geografiska objekt som till exempel byggnader. Algoritmerna kan kopplas till komponenter som knappar och menyer i ett grafiskt användargränssnitt. Användaren kan sedan kommunicera med användargränssnittet och få solinstrålningsdata visualiserade för objekt som är av intresse.

1.2 Problemformulering

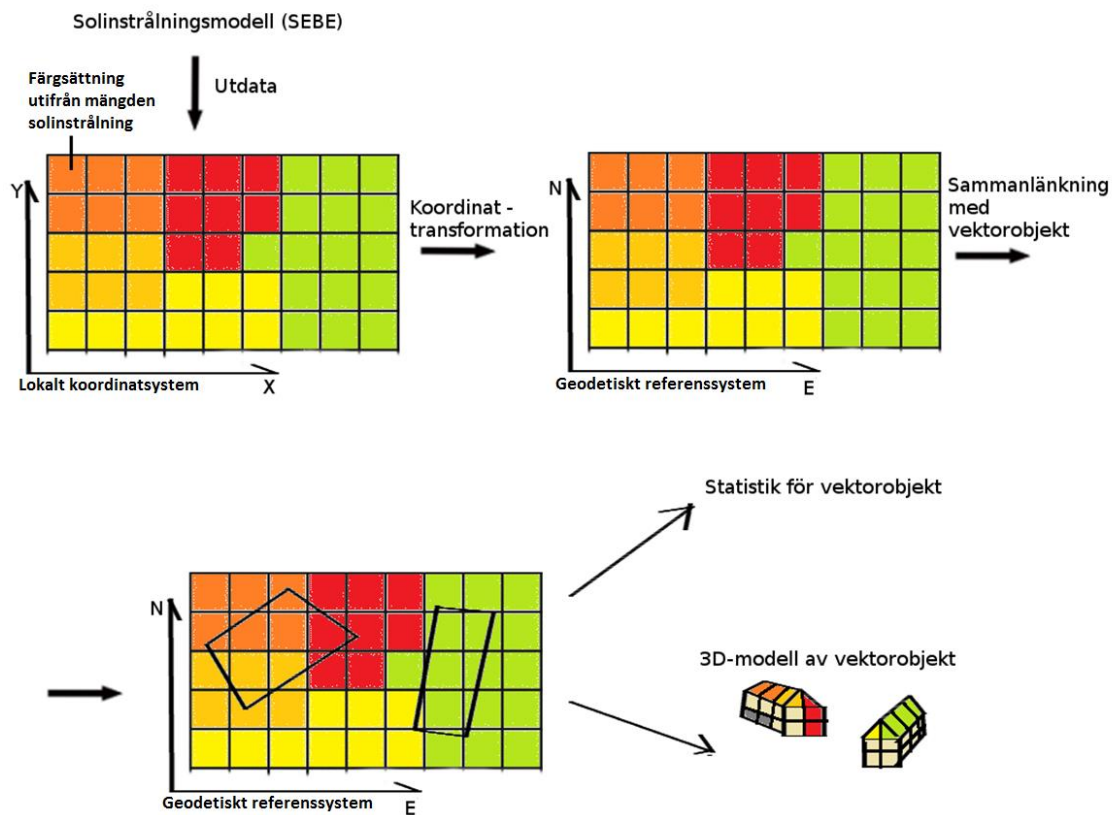
Solinstrålningsmodeller producerar en stor mängd information utan någon inbördes koppling mellan data utöver koordinater. Det är svårt att manuellt utläsa något användbart från solinstrålningsdatan. Det krävs en automatisk process som bearbetar data och presenterar dem så att de kan användas i praktiska tillämpningar, till exempel solpotentialen för ett tak. Detta görs bäst genom utveckling av en applikation. För att applikationen ska vara användbar krävs det att den uppfyller användarens behov. Användargrupper har olika behov och det kan uppstå konflikter där mer avancerade funktioner gör applikationen svårare att använda genom ett mer komplicerat användargränssnitt.

De data som genereras av SEBE och som användes i fallstudien inom detta projekt innehåller koordinater i tre dimensioner vilket gör det möjligt att skapa en tredimensionell (3D)-modell av de data som tagits fram av solinstrålningsmodellen. För att skapa en användbar 3D-modell krävs att man använder sig av metoder som skapar objekt som korrekt efterliknar verkligheten och tillåter lämpliga färgsättningar. Man måste också identifiera de potentiella problem och felkällor som kan uppstå vid visualiseringen.

Utvecklingen av en applikation leder också till nya typer av problematik som kräver lösningar. Problemen kan delas upp i två typer, metodologiska problem och tekniska problem. Metodologiska problem uppstår vid varje moment som applikationen måste utföra och kräver att en metod utvecklas för att hantera dem. Tekniska problem uppstår i applikationens arkitektur där de olika lösningarna ska kopplas samman och kunna hanteras av en användare.

1.3 Grundidén med applikationen

Målet med applikationen var att koppla punktvärden för solinstrålning definierade i ett lokalt koordinatsystem till intressanta objekt t.ex. byggnader i ett vektorlager i ett geodetiskt referenssystem. Solinstrålningspotential för plana ytor är beräknade utifrån ytmodeller. Genom ekvationer som identifierar vertikala språng i ytmodellen har punkter för väggar approximerats. Solinstrålningspotential har beräknats även på väggpunkter. Se Figur 1.1 för illustration av arbetsflödet. En bakgrundskarta lades också till för att förenkla navigation och förståelse hos användaren.



Figur 1.1, Grundidén för applikationen

När punktdata kopplats till ett byggnadsobjekt eller område blir det möjligt att skapa algoritmer för att ta fram objektsspecifik statistik. Vissa typer av information kan vara svår att utläsa enbart från statistik. En 3D-visualisering ger användaren möjlighet att själv navigera fram till intressanta ytor och skapar en mer välbekant kontext för solinstrålningspotentialen.

1.4 Syfte

Det generella syftet med projektet var att undersöka möjligheter och metoder för att skapa en applikation som kan hantera beräkningar på en stor mängd punktdata och koppla datamängden till geografiska objekt såsom byggnader i en användbar miljö. När punktdata blivit knuten till specifika geografiska objekt kan datan visualiseras och analyseras.

Projektet hade fyra specifika syften:

- Att utreda vilka verktyg och metoder som finns tillgängliga för utveckling av en kart- och visualiseringsapplikation.
- Att knyta och sammanfatta statistik och information för solinstrålning till geografiska objekt.
- Att skapa en metod för 3D-visualisering av utdatan från solinstrålningsmodeller.
- Sammanställa ovanstående punkter till en prototyp för en visualiseringsapplikation utvecklad för informationen som kommer från solinstrålningsmodellen.

1.5 Metod

Projektet inleds med att tillsammans med Fredrik Lindberg från institutionen för geovetenskaper på Göteborgs universitets och Dag Wästberg från Tyréns AB ta fram en behovsanalys som identifierar potentiella användare av applikationen och deras behov. Utifrån identifierade behov framställs tekniska krav på applikationen som sammanställs i en kravspecifikation.

Projektet fortskrider med att identifiera och studera rapporter och befintliga lösningar och verktyg för hantering av geografisk data och visualisering i 3D. Utifrån den information som tas fram beslutas om vilket/vilka verktyg och steg som anses lämpliga för att fortskrida med projektet och påbörja arbetet med att ta fram en prototyp. Fokus läggs på verktyg utvecklade med öppen källkod vilket tillåter att resultatet av arbetet kan återskapas. Kravspecifikationen används för att utvärdera den framställda prototypen. I de fall där ett krav inte har uppfyllts diskuteras anledningen i kapitel 9.

Den framtagna prototypen baseras på data från ett mindre område i Göteborgs kommun. Den geografiska information som finns tillgänglig är data från SEBE, vektordata för byggnader i området samt de geografiska data som existerar gratis på internet. Utvecklingen av applikationen behöver ta hänsyn till att fokusområdet i framtiden kan expanderas. Metoder kan alltså inte vara skraddarsydd just för området som används i fallstudien.

Projektet innefattar redovisning av verktyg och algoritmer som är användbara för en implementation av applikationen men också skapandet av en prototyp av applikationen. Det ligger också inom ramen för projektet att undersöka de möjligheter som inom en snar framtid kan finnas tillgängliga inom till

exempel webbutveckling. Den effekt de skulle kunna ha på val av implementationer redovisas. Målet är att göra en rekommendation för hur arbetet med utvecklad visualisering och hantering av resultatet från SEBE i framtiden bör fortskrida.

1.6 Rapportdisposition

Kapitel 2 beskriver en applikations tänkbara användargrupper och deras behov samt tekniska krav i en kravspecifikation. I kapitel 3 redovisas solinstrålningsteori och ett antal solinstrålningsmodeller presenteras. Kapitel 4 beskriver applikationsarkitekturer som kan användas för att utveckla en applikation, deras för- och nackdelar samt exempel på redan existerande liknande applikationer. I kapitel 5 presenteras visualiseringstekniker som kan användas för att skapa en 3D-modell och hur man går till väga för att bygga upp objekt i 3D. I Kapitel 6 motiveras valet av applikationsarkitektur baserat på fastställda behov och krav. Kapitlet innehåller också en detaljerad genomgång av applikationsarkitekturen.

Kapitel 7 beskriver utvecklingsarbetet, de alternativ som fanns för utvecklingen och de val som gjordes. I kapitel 8 presenteras resultatet av arbetet genom en avstämning mot kravspecifikationen. I Kapitel 9 förs en diskussion om säkerheten i resultatet, eventuella brister, hur dessa kan hanteras och hur framtida utveckling av den bakomliggande teknologin skulle kunna förändra implementationen av solinstrålningsapplikationen. I Kapitel 10 dras slutsatser om projektet och om hur arbetet med applikationen bör fortskrida. Appendix A innehåller en manual som beskriver användningen av applikationen. Appendix B och C innehåller källkod för delar av applikationen.

2. Användarkrav

2.1 Behovsanalys

En behovsanalys har utförts för att ta reda på vilka funktionaliteter som är önskvärd för en applikation. Behovsanalysen syftar till att identifiera vilka potentiella användare som skulle kunna utnyttja applikationen och hur deras behov ser ut. På grund av begränsad tidsram har analysen utförts genom diskussion med personer knutna till projektet samt genom litteraturstudier. Hade det funnits tid hade analysen involverat de potentiella användarna och innefattat deras åsikter.

En distinktion som är värd att känna till när man diskuterar nyttan av solinstrålning är att solfångare används för uppvärmning av byggnader eller varmvatten medan solceller och solpaneler används för att omvandla solinstrålningen till el (Energimyndigheten, 2012b). Statens solvärmestöd för installationen av solfångare upphörde vid årskiftet 2011/2012 och ersattes istället till viss del av möjligheten att få ROT-avdrag (Boverket, 2012). Enligt förordning (SFS 2009:689) om statligt stöd till solceller finns sedan 2009 möjligheten att få ekonomiskt stöd för installation av solceller genom Länsstyrelsen. Det är möjligt att ansöka om stödet fram till år 2016 men stödet gäller enbart för elnätsansluta solcellssystem. Stödet syftar till att användningen av solcellssystem och antalet aktörer som hanterar sådana system ska öka, att systemkostnaderna ska sänkas och att den årliga elproduktionen från solceller ska öka med minst 2,5 gigawattimmar under stödperioden.

Det är svårt att förutspå den exakta utvecklingen av solenergimarknaden men i rapporter identifieras en ökande trend för både solfångare och solcellssystem. En rapport från en analysgrupp tillsatt av Energimyndigheten utreder marknadsutvecklingen för solvärme mellan åren 2000 – 2010. Rapporten presenterar en kraftig ökning av försäljningen av glasade solfångare från 8 000 kvadratmeter per år 1998 till 25 000 kvadratmeter per år 2006-2008. Detta följdes av en minskning av försäljningen till ungefär 20 000 kvadratmeter per år 2009-2010. Den kraftiga ökningen tillskrivs delvis förbättringar inom tekniken men framförallt solvärmestödet som delades ut av Länsstyrelsen. Minskningen efter 2008 spekuleras vara kopplad till aktörer som lämnat marknaden och det hårda ekonomiska klimatet efter den finansiella krisen (Energimyndigheten, 2010). I ett pressmeddelande från Energimyndigheten fastställs att den totala effekten av installerade solceller i Sverige ökat kraftigt både 2012 och 2013. En del av ökningen tros bero på en prisreduktion i systempriser. Det konstateras att marknaden är bunden till politiska beslut trots ett stort allmänt intresse. För marknaden förutspås en fortsatt ökning för kommande år (Energimyndigheten, 2014).

Privata användare

En privatperson kan ha både en ekonomisk och politisk anledning till att vilja installera solceller på sin bostad. Vid installation av solenergisystem kan det förväntas att det är fastighetsägaren som tar beslutet. Intressenter är därför främst personer som äger sin egen bostad som till exempel villaägare. För år 2012 uppskattades det att 25 % av den totala genererade effekten från solceller i Sverige kommer från solsystemet för villor. Privatpersoner utgör alltså en betydande del av solcellsmarknaden även om den totala mängden producerad energi är liten i förhållande till den totala konsumtionen i hela Sverige (Lindahl, 2013). Ett problem med solenergisystem för en

bostadsbyggnad är att behovet av energi är som störst under kvällstid och produktionen är som högst under dagtid. Tekniska lösningar som förbättrar möjligheter att spara effekterna av solinstrålningen under dygnet kan tänkas öka efterfrågan på solesystem hos privatpersoner.

Privatpersoner har med stor sannolikhet inte någon tidigare erfarenhet av GIS-verktyg. Deras behov är främst att titta på ett eller ett fåtal hus. För de hus de är intresserade av räcker en ganska liten mängd information. Det handlar främst om en sammanfattning av solinstrålning för ett tak eller en del av ett tak för att få fram en lämplig placering av solpaneler.

Privata användare vill kunna nå informationen snabbt och enkelt. De vill arbeta i en bekant miljö. Om möjligt vill de slippa installera några nya programvaror.

Kommuner och företag

Kommuner och företag äger ofta en stor mängd byggnader. Fastighetsbeståndet kan bestå av byggnader utspridda i en stor geografisk yta. De har ett behov att kunna få fram information för både ett fåtal och en större mängd byggnader från fastighetsbeståndet. Ska större mängder byggnader analyseras krävs mer detaljerad statistik för att användare ska kunna skilja mellan olika områden och byggnader. Behovet att kunna rendera modeller med större geografisk utsträckning blir högre.

För kommuner och företag kan det handla om att utreda nyttan av solenergisystem för större projekt. De lokaler kommuner och företag äger används främst under dagen vilket innebär att solenergisystem kan dra direkt nytta av solinstrålningen. Det kan därför förväntas att de lägger ner mer arbete på underlaget. Detta innefattar att installera nya programvaror och lära sig nya användarmiljöer.

En studie utförd för Strängnäs kommun undersöker vilken potential det finns för solenergiteknik på kommunens fastigheter. Bakgrunden till undersökningen är att kommunen satt upp en strategisk plan där solenergianvändning ska uppmuntras. Studien omfattar cirka 100 byggnader och identifierar ett antal kriterier för lämplig placering. Exempelvis ett tak i sydlig riktning utan skugga. Stora delar av undersökningen förlitar sig på undersökningar som utförs på plats vid varje objekt (Energi Engagemang Sverige AB, 2014). Möjligheten att istället kunna göra delar av undersökningen för flera objekt genom en applikation som visualiserar instrålningsdata borde därför vara något med stor nyttopotential.

Energi- och klimatrådgivare

Utöver kommuners intresse för sina egna lokaler erbjuder de tjänster till allmänheten i form av energi- och klimatrådgivning. En energi- och klimatrådgivare arbetar bland annat med att ge invånare i kommunen råd om hur de kan minska sin energianvändning, klimatpåverkan och vilka tekniska lösningar de har att välja mellan. En energirådgivare har därför behov av att kunna ta fram underlag för specifik solinstrålning för alla byggnader i kommunen. Det är viktigt att informationen går att ta fram på ett sätt som gör det möjligt att sprida det underlag som en energirådgivare tagit fram till den eller de intressenter som rådgivningen riktar sig till.

Planarkitekter

Den svenska regeringen har ställt upp ett antal nationella miljömål som ska fungera som rekommendationer för hur bland annat kommuner driver sin verksamhet. En del av dessa miljömål handlar om att minska beroendet av fossila bränslen och öka användningen av energi från förnybara källor som solenergi (Naturvårdsverket, 2014). Dessa miljömål skulle kunna leda till ett större fokus på att skapa miljöer som maximerar möjligheterna att ta tillvara på solinstrålningen genom detalj- och översiktsplaner.

En planarkitekt hanterar detaljplaner och översiktsplaner. De kan vara intresserade av att utreda den generella solinstrålningssituationen i ett större område. I arbetet med en detaljplan är det också intressant att se hur ny byggnation påverkar solinstrålning för andra byggnader eller solinstrålning för en tänkt byggnad. För en planarkitekt är det alltså intressant att kunna påverka modellen genom att lägga in nya data och direkt få ut nya värden för solinstrålning.

Landskapsarkitekter/Stadsplanerare

En landskapsarkitekt arbetar bland annat med planering och miljökonsekvensanalyser då man exempelvis vill förändra användningen av landskapet. Det kan gälla till exempel planerad bebyggelse, vägar eller friluftsområden. Landskapsarkitekter jobbar också med att utforma en stads offentliga platser såsom torg och parker. För landskapsarkitekter är det därför viktigt att ta tillvara det inkommande solljuset för att skapa en positiv utomhusmiljö. Ett behov är då att kunna välja ett specifikt område, såsom en park, för att få fram statistik för solinstrålning i hela området. Denna statistik skulle då kunna användas för att bestämma lämplig placering av till exempel träd i en park så att skuggningen från vegetationen bidrar till utomhusmiljön.

Arkitekter

En arkitekt som arbetar med att designa ett specifikt hus skulle kunna vara intresserad av att studera en byggnads alla ytor mer i detalj. Byggnadssektorn svarar för ungefär en tredjedel av all energikonsumtion i västvärlden (Roecker och Probst, 2012) och det är därför viktigt att byggnader kan utformas för att dra maximal nytta av solinstrålning. För att maximera nyttan för uppvärmning måste ett hus utformas för att släppa in så mycket solinstrålning som möjligt samtidigt som energin bevaras. Det finns teknologi som gör det möjligt att omvandla inkommande solenergi till värme, el och till och med kylning. Det är därför en stor nytta ifall hus kan utformas på ett sätt där utplacering av till exempel solpaneler är så effektiv som möjligt. Samtidigt måste arkitekter skapa byggnader med en viss designkvalitet och estetik som uppfyller och tar hänsyn till en mängd andra krav utöver energifrågan. Ett behov arkitekter har är därför att på en byggnad kunna utvärdera flera olika ytor baserat på deras solinstrålningspotential och utifrån detta kunna välja det område där nyttan av solinstrålningen maximeras samtidigt som byggnadens estetiska kvaliteter kvarstår (Roecker och Probst, 2012).

Funktionalitet som tillåter att nya objekt skapas direkt i modellen och utvärderas genom SEBE:s beräkningar i förhållande till redan existerande förhållanden bedöms vara intressant för en arkitekt.

Arkitekten kan också använda sig av redan existerande objekt för att skapa sig en uppfattning om vilken effekt husets utformning har för absorption och bevarande av solinstrålning.

Generella behov

Solinstrålning står för det överlägset största förnybara energiflödet på jorden med ett flöde som överstiger den globala energianvändningen flera tusen gånger om (Naturvårdsverket, 2010). Den stora potentialen gör att solinstrålning blir en intressant energikälla även för ett land som Sverige som bara mottar runt hälften så mycket solinstrålning som ett av de länder där solinstrålningen är som störst. Den relativt låga solinstrålningen i Sverige ställer högre tekniska krav på solenergissystemen och kräver att de system som installeras är placerade i en optimal miljö (Naturvårdsverket, 2010). Generellt är därför behovet stort för verktyg som ökar möjligheter att utreda solpotential och solinstrålningssituationer. Det kan vara svårt att bemöta alla behov som finns för att ta fram information om solinstrålning enbart genom att framställa olika typer av statistik. Därför finns ett generellt behov av att kunna ta fram en 3D-modell som visualiserar solinstrålningen på till exempel byggnader. Genom en visualisering i 3D skapas en interaktiv miljö där användare kan undersöka intressanta ytor genom att rotera modellen i olika skalor.

Solceller är, utöver möjligtvis i någon typ av specialfall, aldrig lämpliga för placering av solpaneler. Med tekniska framsteg kan det tänkas att detta blir mer aktuellt i framtiden men i nästan alla fall tar takytor emot mer solinstrålning än fasader. Att kunna se mängden solinstrålning även för fasadytor kan dock vara behov i andra typer av sammanhang. Till exempel undersökningar av existerande byggnaders termiska egenskaper skulle kunna ha nytta av informationen. Att kunna se solinstrålningen längs fasadytor kan också tänkas vara ett behov vid installationen av nya fönster på en eller flera byggnader.

Det finns fler användargrupper som har behov av applikationen än de ovan nämnda. Alla områden som på något sätt hanterar byggnation eller solinstrålning är intressanta att nå och de användare som redan tagits upp, utgör bara ett par exempel. Generella behov som anses beröra flera användargrupper har därför tagits fram:

- Statistiken som visas bör kunna ses både utifrån ett helårsperspektiv och utifrån kortare perioder som kvartal.
- Användare vill kunna få fram solinstrålningssituationer för ett större område eller ett antal byggnader.
- Användare vill kunna få fram solinstrålningssituationer för enbart en byggnad.
- Användare vill kunna få fram solinstrålningssituationer som skiljer mellan data för en fasad eller ett hustak.
- Användare vill kunna få fram solinstrålningssituationer för ett område som bara täcker en mindre yta av en fasad eller ett hustak.

2.2 Kravspecifikation

Utifrån de behov som har identifierats har en kravspecifikation framställts. Några av kraven klassas som nödvändiga för applikationen och några som önskvärda, men en användbar applikation går att skapa utan dem.

Nödvändiga krav:

1. Läs in och visa vektorlager.
2. Modifiera vektorlager.
3. Läs in och visa rasterlager.
4. Utföra beräkningar på rasterlager.
5. Panorera över kartområdet.
6. Visualisera kartområdet i olika skalor.
7. En legend över visualiserade lager.
8. En bakgrundskarta över området för enkel navigering.
9. Verktyg för att markera objekt eller område på kartan.
10. Ta fram data från solinstrålningsmodell för markerat objekt eller område.
11. Beräkna statistik utifrån modelldata på objekt eller flera objekt i området.
12. Skilja på solinstrålning på olika höjder, till exempel tak och mark.
13. Skilja på solinstrålning för ytor med olika solvinklar, till exempel fasad och tak.
14. Visa statistik för olika tidsperioder.
15. Visualisera objekt och område, exempelvis i 3D.

Önskvärda krav:

16. Webbpublicering av huvudkraven.
17. Inkludera beräkningar för modelldata i applikationen för att tillåta ändringar av in- och utdata.

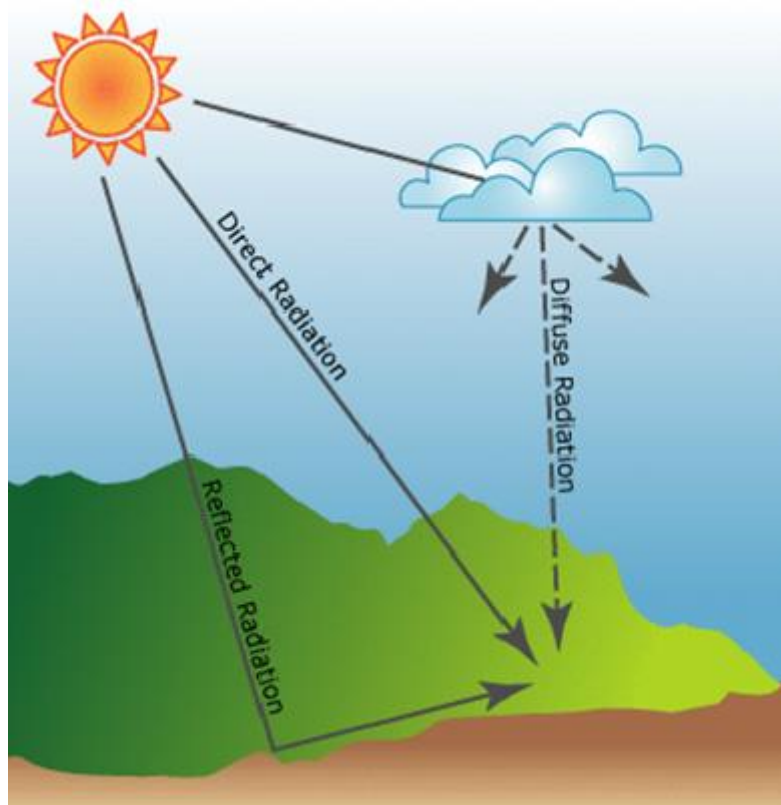
3. Solinstrålning

3.1 Solinstrålningsteori

Detta avsnitt presenterar faktorer som är viktiga för att beräkna solinstrålning. Beräkningar för inkommande solinstrålning ser ut som följer:

$$S = S_{dir} + S_{dif} + S_{ref} \quad (3.1)$$

där S är den totala solinstrålningen som når en punkt och mäts i enheten W/m^2 , S_{dir} är den direkta strålningen från solen som når en punkt, S_{dif} är den diffusa strålningen som når en yta från hela himlavalvet genom att brytas mot störningar som dimma, moln eller avgaser och S_{ref} är strålning som når en punkt men som tidigare har reflekterats från ett objekt, till exempel en byggnad. Formel 3.1 beskriver bara energiflödet och inte själva intensiteten av solinstrålningen. För att bestämma intensiteten på den inkommande solinstrålningen finns fyra faktorer som måste tas hänsyn till. Värdet på solarkonstanten, atmosfärens transmissivitet, platsens geografiska position i förhållande till ekvatorn och tiden på säsongen. Se Figur 3.1 för en illustration av de olika typerna av solinstrålning.



Figur 3.1, Direkt, diffus och reflekterad solinstrålning mot en punkt (Sinclair Knight Merz, 2010, p.27).

Värdet på solarkonstanten är den solenergi som når jordens yttre atmosfär. Konstanten har varit relativt oförändrad de senaste århundradena och uppskattas till $1368 W/m^2$.

Atmosfärens transmissivitet kan främst kopplas till mängden moln vilket kan minska intensiteten på

den totala solinstrålningen med upp till 80 %. Jordens form innebär att solinstrålningens intensitet är högre i områden nära ekvatorn och minskar i områden längre bort från ekvatorn. Jordens bana runt solen innebär en förändring i solinstrålningens intensitet över de olika säsongerna under ett år (Saha, 2008).

När beräkningar av solinstrålning sedan ska göras i mindre områden med högre detaljgrad måste ytterligare faktorer läggas in i beräkningarna. En ytas riktning och lutning har stort inflytande över den inkommande solinstrålningen under ett dygn och hänsyn måste tas till ifall kringliggande byggnation och vegetation skuggar ytan under olika perioder på dygnet.

3.2 Solinstrålningsmodeller

En solinstrålningsmodell, eller solsimulator, är ett verktyg som, givet vissa indata, genererar uppskattade värden för solinstrålning på till exempel ett område eller en byggnad. Det finns flera faktorer som en modell bör ta hänsyn till (se avsnitt 3.1) (SolEI-Programmet, 2014). I avsnitten som följer redovisas för olika solinstrålningsmodeller samt vilka indata som krävs, hur de hanteras och vilka beräkningar som utförs.

3.2.1 SEBE

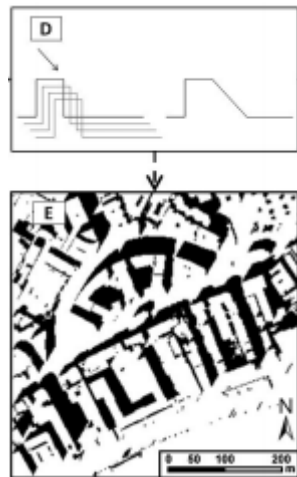
SEBE är den solinstrålningsmodell som används i fallstudien inom detta projekt. SEBE är ett verktyg för mätning av solpotential på byggnadsytor, markytor och väggytor som tar hänsyn till skuggning från kringliggande geografi som andra byggnader, topografi och vegetation samt andra aspekter som omkringliggande ytors reflexionsförmåga.

SEBE har ett rasterbaserat tillvägagångssätt vilket utnyttjar att raster tillåter effektivare beräkningar än till exempel ett vektorbaserat system. Detta gör det möjligt att köra modellen på stora områden som hela städer. SEBE kräver en ytmodell över byggnads- och markhöjder samt meteorologiska data över solinstrålning för att kunna utföra beräkningar. Dessa indata kan kompletteras med ytmodeller i samma storlek och upplösning som innehåller till exempelvis vegetation. Andra meteorologiska data som kan ha en effekt på solinstrålning kan adderas, exempelvis omkringliggande lufttemperatur och luftfuktighet. De meteorologiska data som används är global solinstrålning, direkt solinstrålning vinkelrätt mot solen och diffus solinstrålning (se avsnitt 3.1) under ett helt år per timme. Om datan saknar direkt eller diffus instrålning kan modellen istället beräkna dessa. Den diffusa instrålningen kan beräknas utifrån den globala solinstrålningen genom ett tillvägagångssätt utvecklat av Reindl et al. (1990). Reindls metod identifierar de fyra faktorer som har störst effekt på hur stor del av den globala solinstrålningen som blir diffus, det vill säga delvis dämpad, solinstrålning för ett område. Faktorerna är mängden klar himmel per timme, solens höjd, omkringliggande temperatur och luftfuktighet. Utifrån dessa krävs enbart mängden klar himmel per timme och solens höjd för att ett ungefärligt värde för den diffusa solinstrålningen ska kunna tas fram. Den direkta solinstrålningen på en yta vinkelrätt mot solen kan sedan uppskattas genom:

$$I = (D - G)/\text{Sin}(\eta) \quad (3.2)$$

där I är den direkta solinstrålningen och mäts i enheten W/m^2 , D den diffusa solinstrålningen, G den globala solinstrålningen och η är vinkeln mot solens höjd över horisonten. (Lindberg et al. 2014).

Modellen beräknar utifrån förskjutningar av ytmodellerna och solens azimut samt höjd under ett dygn skuggning för mark- och byggnadsytor enligt en metod utvecklad av Lindberg (2010). En del av metoden presenteras i Figur 3.2 som består av två rutor. Den övre rutan märkt D visar till vänster förskjutningen av ytmodellen och till höger den resulterande skuggvolymen som skapats utifrån förskjutningen. Den undre rutan märkt E visar en skuggbild som skapats utifrån en bestämd position för solen. I detta fall azimuth = 210° och höjd = 41° .



Figur 3.2, Del av beräkningen av skuggor utifrån ytmodeller och solens position (Lindberg, 2010).

Solinstrålningen för en takcell vid en viss tidpunkt ges av:

$$R = I\omega K + DK + G(1 - K)\alpha \quad (3.3)$$

där R är den totala solinstrålningen, I den direkta solinstrålningen vinkelrätt mot solen, ω solens infallsvinkel mot en yta, D den diffusa solinstrålningen, G den globala solinstrålningen, α generell ytalbedo (dvs. kringliggande ytors reflexionsförmåga) och K är skuggningen i cellen. Den sistnämnda beräknas genom:

$$K = K_b - (1 - K_v)(1 - \tau) \quad (3.4)$$

där K_b är skugga från byggnader, K_v är skugga från vegetation och τ är mängden solinstrålning som vegetationen släpper igenom. K_b och K_v representeras av 1 för närvaro av skugga och 0 för frånvaron av skugga. Formeln för instrålning mot en takcell beräknas för solens positioner under olika tider på dygnet och instrålningar för olika tider på året.

Väggar uppskattas genom ett filter som identifierar kanter i ytmodellen för byggnader och markytor. Uppskattade väggytor körs sedan genom samma formler som takytor men med vissa modifikationer för reflektion från marken (Lindberg et al. 2014).

SEBE är skrivet i programmeringsspråket MATLAB. För tillfället körs programmet genom MATLAB men ett användargränssnitt i form av ett insticksprogram till QGIS är under utveckling. För att göra en beräkning av solpotential genom SEBE följs ett arbetsflöde med sex steg:

1. Först laddas relevanta ytmodeller för landskap, byggnader och vegetation in i modellen.

2. I modellen kan man markera punkter som är speciellt intressanta. Modellen kommer sedan att generera textfiler med mer detaljerad information för dessa punkter som höjd och solinstrålning för just den punkten. Detta steg är frivilligt.
3. Steg tre är att förse modellen med relevanta skuggbilder. Dessa kan antingen laddas in av användaren eller skapas i modellen enligt metoden framtagen av Lindberg (2010) beskriven ovan (se Figur 3.2).
4. Vid detta steg bestämmer användaren den generella albedon som ska gälla i området, mängden solinstrålning vegetation släpper igenom och grupperingar av resultatet.
5. Vid detta steg definerar användaren en geografisk position för beräkningsområdet som används för att uppskatta förhållanden som solens position under dagar och säsonger på just den delen av jordklotet. Sedan lägger användaren in relevanta meteorologiska data som finns tillgänglig för området som global, direkt och diffus solinstrålning samt data för faktorer som luftfuktighet och lufttemperatur. Den enda informationen som är ett krav är data för global solinstrålning per timme över ett helt år men resultatet kan förbättras om ytterligare data tillförs.
6. Sista steget innebär att användaren kör modellen (Jonsson, 2011).

SEBE beräknar solpotential. Värdena bör tolkas som riktvärden och underlag för att gå vidare med undersökningar om lämpligheten för solenergysystemsinstallation.

3.2.2 PVGIS

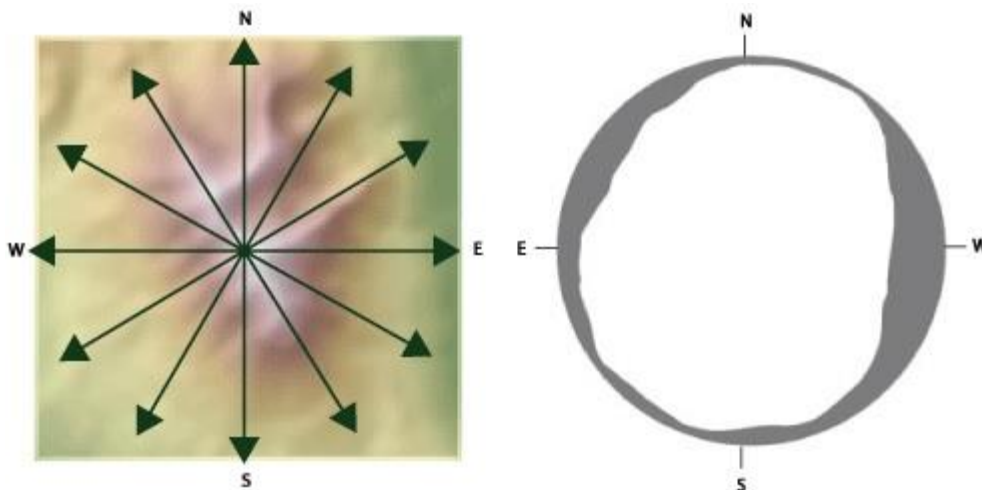
PVGIS är en modell framtagen av EU-kommissionens generaldirektorat med fokus främst på att undersöka lämpliga positioner för olika typer av solenergisystem. Modellen går igenom fyra beräkningssteg:

1. Beräkning av global solinstrålning vid klar himmel mot en horisontell yta.
2. Kombination av den globala solinstrålningen vid klar himmel med data för den genomsnittliga mängden störningar på himlen såsom moln.
3. Beräkning av olika delar av solinstrålningen såsom diffus solinstrålning samt global solinstrålning mot sluttande ytor.
4. Test av resultatets säkerhet.

PVGIS använder sig av rasterlager för beräkningar men tar inte hänsyn till aspekter som vegetation. Eftersom modellen fokuserar på att optimera effekterna vid installation av solenergisystem tar beräkningarna inte hänsyn till den nuvarande lutningen på ytor. Solenergisystem kan installeras i olika vinklar och resultatet redovisar därför mängden solinstrålning för horisontella ytor och ytor med en lutning på 15, 25 eller 40 grader. Solinstrålningen redovisas både per månad och per år. Resultatet blir ett antal rasterkartor där till exempel solinstrålning under januari mot ytor med en lutning på 40 grader blir en av rasterkartorna (European Commission, 2012).

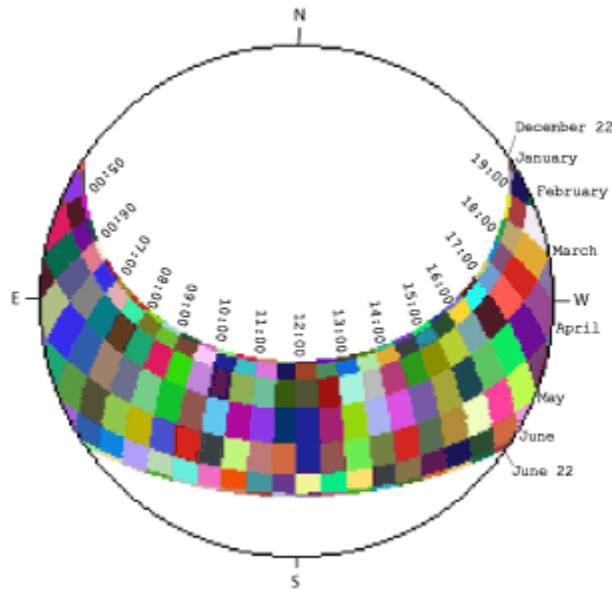
3.2.3 ArcMap Solar Analysis

ArcMap Solar Analysis är en tilläggsmodul i GIS-verktyget ArcMap och utgör en del av en samling metoder för rumslig analys. Då den reflekterade instrålningen i formeln från avsnitt 3.1 i de flesta fall bidrar minst till den totala instrålningen på en yta tar inte ArcMap-metoden hänsyn till denna. Solinstrålning till celler i en digital ytmodell beräknas genom att först uppskatta mängden synlig himmel från varje cell utifrån omkringliggande topografi. Figur 3.3 visar till vänster en ytmodell där en cell har markerats med riktningar utmålade och till höger den resulterande mängden synlig himmel. Vita celler representerar synlig himmel och grå celler icke synlig himmel.



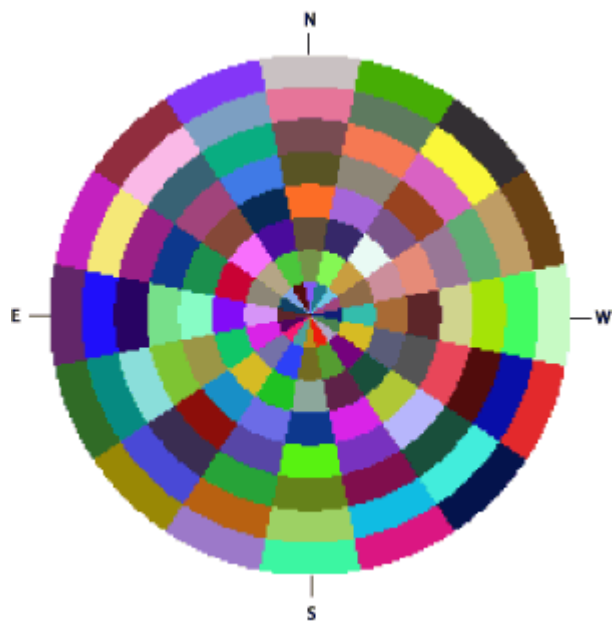
Figur 3.3, Figuren illustrerar mängden synlig himmel utifrån en cell i ett raster. Figuren till vänster visar en cell som representerar en punkt på jorden. Figuren till höger visar mängden synlig himmel från punkten (Esri Inc, 2007).

Mängden direkt solinstrålning från varje riktning på himlen beräknas med hjälp av en solkarta. En solkarta är ett raster som beskriver solens bana och beskrivs genom sektorer som baseras på solens position under dygnets timmar och under perioder på året. Solkartan skapas utifrån ytans geografiska position i förhållande till ekvatorn och de sektorer som definierats. För varje sektor beräknas den direkta solinstrålningen sedan separat (Esri Inc, 2007). Figur 3.4 visar ett exempel på en solkarta.



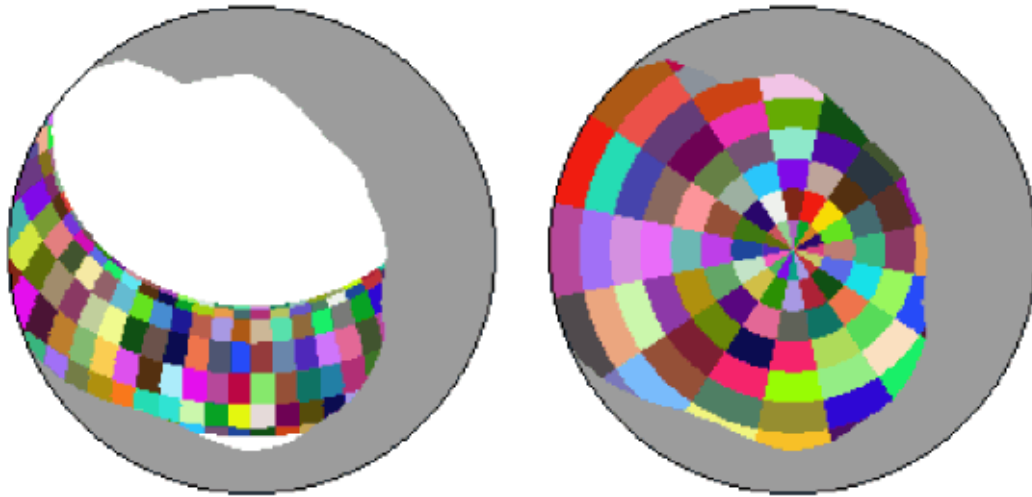
Figur 3.4, En solkarta som beskriver solens bana där varje sektor är direkt solinstrålning i området (Esri inc, 2007).

Diffus solinstrålning härstammar inte direkt från solen och kan därför stråla in mot en yta från alla riktningar på himlen. Den diffusa solinstrålningen beräknas därför genom att en karta skapas över hela himlavalvet. Kartan delas upp i sektorer där varje sektor tilldelas vinklar för zenit och azimut. Diffus instrålning beräknas sedan för varje sektor baserat på dess zenit- och azimutvinklar. Solkartan med direkt solinstrålning och kartan över himlavalvet med diffus solinstrålning överlagras mot rastret med mängden synlig himmel för en punkt och den totala instrålningen till en yta ges av summering av de båda (Esri Inc, 2007). Figur 3.5 visar ett exempel på en karta över diffus solinstrålning som täcker hela himla valvet.



Figur 3.5, Karta över hela himlavalvet där varje sektor beskriver mängden diffus solinstrålning beroende på zenit och azimut. 8 zenit uppdelningar och 16 azimut uppdelningar gäller för exemplet i figuren (Esri inc, 2007).

Figur 3.6 visar till vänster solkartan med direkt solinstrålning i sektorer överlagrad med rastret för mängden synlig himmel från en yta. Till höger kartan över himla valvet med diffus solinstrålning i sektorer överlagrad med samma raster.



Figur 3.6, Solkarta (direkt solinstrålning) med synlig himmel till vänster och karta över himlavalvet (diffus solinstrålning) med synlig himmel till höger (Esri inc, 2007).

4. Applikationsarkitektur

Detta avsnitt presenterar potentiella programarkitekturer för utveckling av en applikation. För varje teknik som presenteras ges en beskrivning av arkitekturen, för- och nackdelar med tekniken samt exempel på nuvarande tillämpningar av densamma.

4.1 Webbapplikation

Applikationen skulle kunna skapas som en webbapplikation (och användare skulle på så sätt kunna nå den genom valfri webbläsare).

4.1.1 Teoretisk uppbyggnad

En webbapplikation skulle kunna vara baserad på ett eller flera javascriptbibliotek för hantering av kartdata. Ett exempel på sådant bibliotek är OpenLayers. Genom OpenLayers kan man importera en bakgrundskarta som till exempel OpenStreetMap eller GoogleMaps och till denna genom ett Web Map Service (WMS)-anrop hämta ett vektorlager innehållande byggnader för området. Detta kräver att vektorlagret lagts upp på en serverstruktur såsom Geoserver (The GeoServer Project, 2014a).

Till vektorlagret med byggnader kan sedan information från solinstrålningsmodellen kopplas och hämtas ut genom ett anrop. Resultatet av anropet kan modifieras genom en mall i HTML-format (The GeoServer Project, 2014b).

För 3D-visualisering skulle till exempel WebGL kunna användas vilket stöds av de flesta moderna webbläsare som Google Chrome och Mozilla Firefox. WebGL har tagits fram av den organisationen KhronosGroup och är baserat på 3D-grafiks API:et OpenGL (Khronos group, 2014).

Applikationen skulle kunna tänkas implementeras så att när en användare valt ett objekt på 2D-kartan hämtas först information om solinstrålning för objektet och visas för användaren. Det ges också möjlighet att för detta objekt generera en 3D-representation.

Fördelar

- Enkelt att sprida och enkelt för en användare att nå.
- Kräver inte att en användare laddar ner något till sin egen hårddisk.
- De flesta användare bör vara bekanta med användningen.
- Enklare support

Nackdelar

- Begränsad tillgång till utvecklingsverktyg.
- Implementationen måste stödjas av webbläsare.
- Kräver tillgång till en webbdomän och serverstruktur.

Utifrån de identifierade fördelarna och nackdelarna är denna typ av arkitektur bäst lämpad för utveckling som riktar sig till privata användare. Webbaserade kartapplikationer är något som större delen av privata användare är bekanta med.

Nackdelarna med denna arkitektur bedöms ha större inverkan på kommuner, företag och andra aktörer som är intresserade av att inhämta data för större områden och mer komplicerade funktioner. Begränsningarna av utvecklingsmöjligheterna kan leda till att dessa användargrupperns behov inte går att bemötas på ett effektivt sätt.

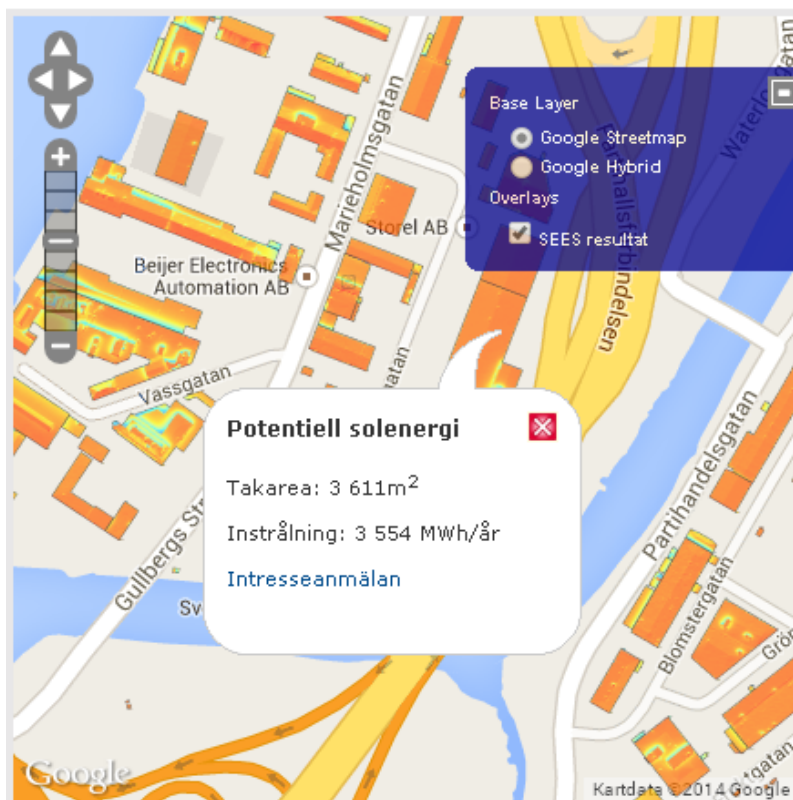
4.1.2 Exempel på applikationer

Solkartor

Solkartor för flera städer i Sverige som Göteborg, Stockholm och Lund har redan utvecklats. De flesta fungerar på ett liknande sätt även om implementationen kan ha skillnader.

Solkartan Göteborg

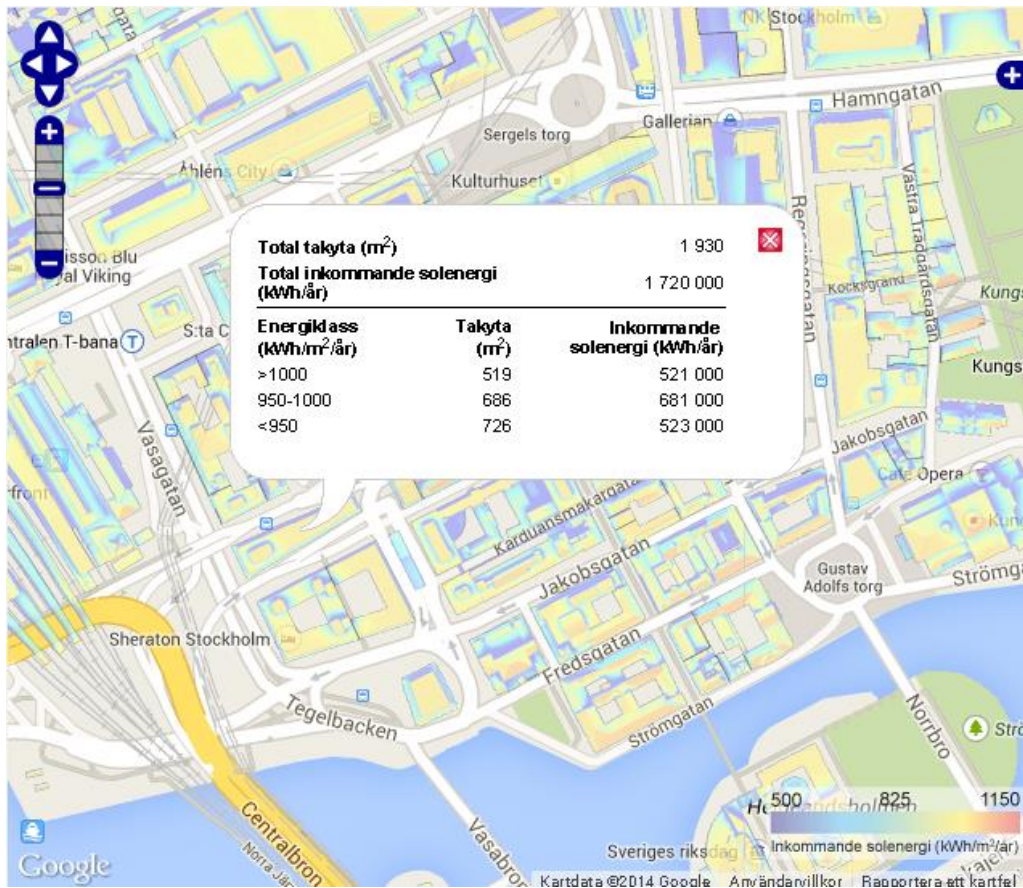
Solkartan i Göteborg använder SEBE för uträkning av solinstrålning (se avsnitt 3.2.1) med OpenLayers som en bas. Google maps används som bakgrundskarta. Ett vektorlager med information om solinstrålning överlagras bakgrundskartan och när en användare klickar på ett hus erhålls information om det specifika huset (Göteborg Energi AB, 2014). Se Figur 4.1 för bildexempel från solkartan.



Figur 4.1, Bild av solkartan Göteborg (Göteborg Energi AB, 2014).

Solkartan Stockholm

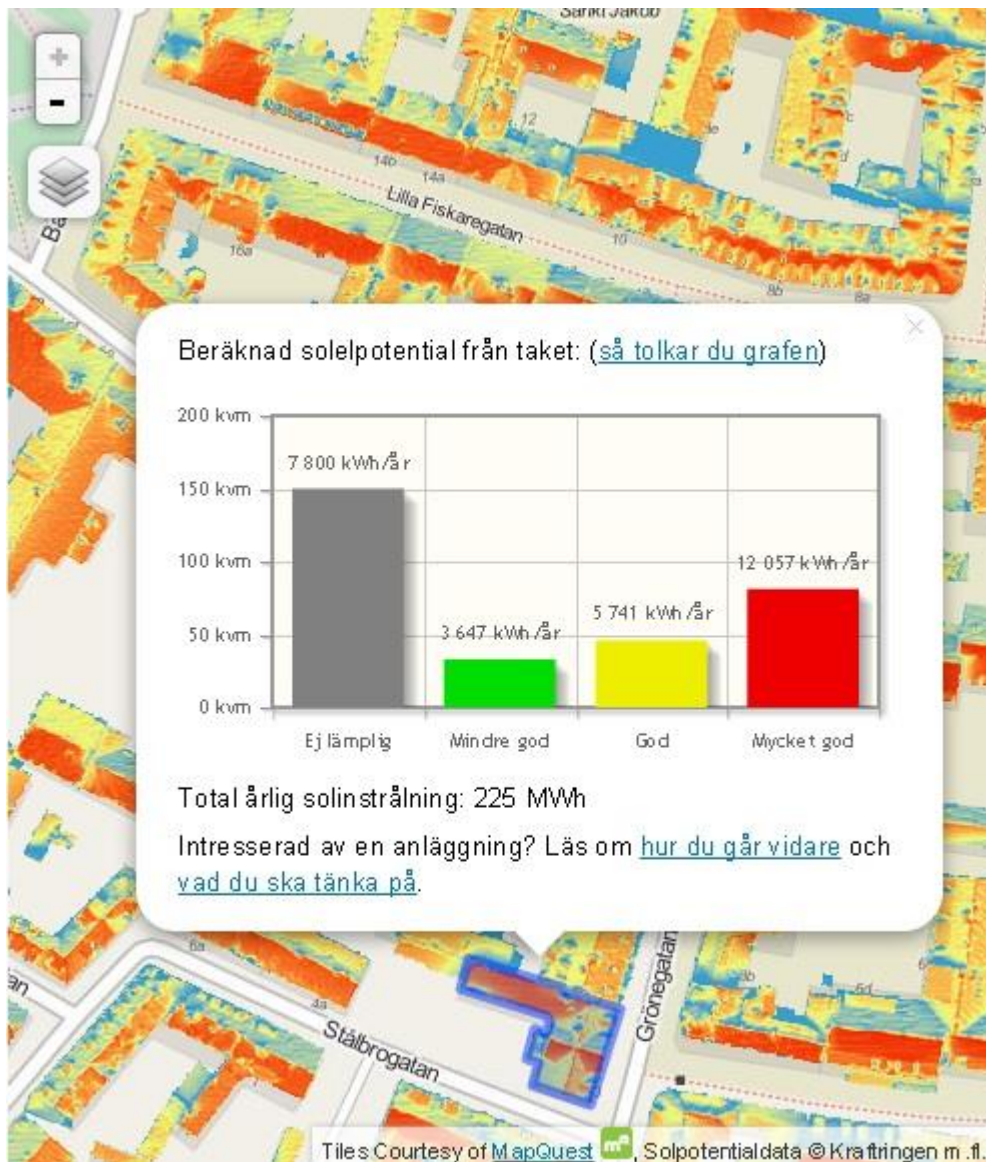
Datan för Solkartan Stockholm är också framtagen av SEBE. Detta bidrar till att implementation liknar den för Solkartan Göteborg. För varje byggnad har man valt att presentera mer detaljerad information där solinstrålning har grupperats efter instrålmängd. För varje grupp har sedan den totala takytan och mängden inkommande solenergi summerats. Färgschemat för byggnaderna i kartapplikationer är också annorlunda. Se Figur 4.2 för ett exempel från Solkartan Stockholm (Stockholms stad, 2014).



Figur 4.2, Bild av solkartan Stockholm (Stockholms Stad, 2014).

Solkartan Lund

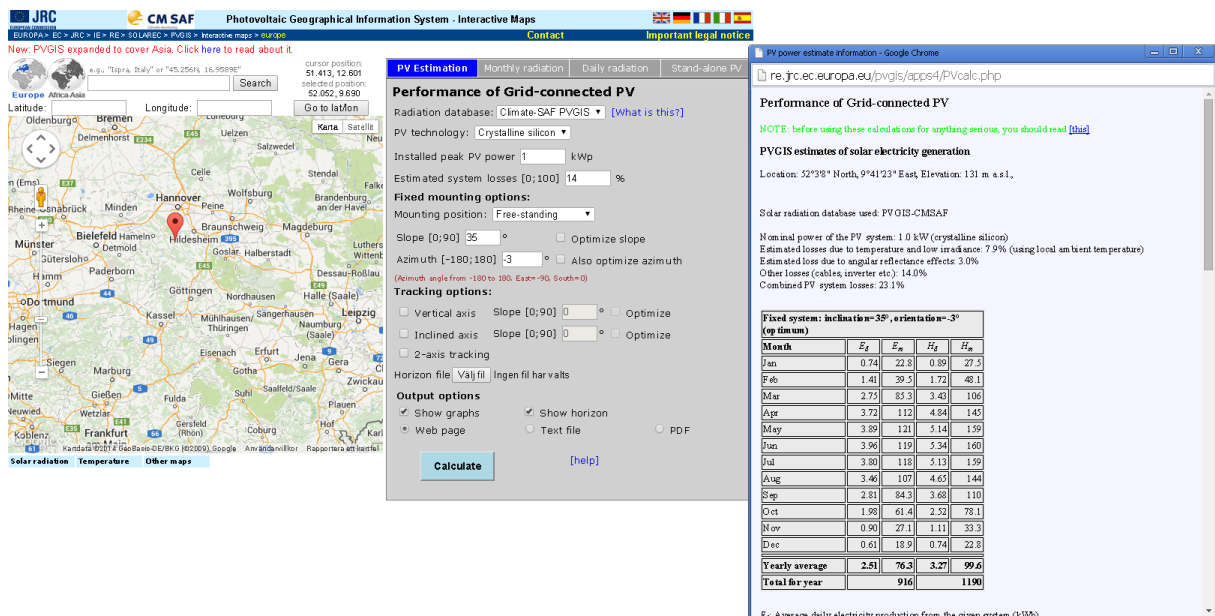
Solkartan för Lunds kommun är framtagen av Kraftringen i samarbete med Lunds Tekniska Högskola, Lunds kommun och Solar Region Skåne (Hedén, 2013). Den har skillnader i sin presentation i förhållande till både Solkartan Stockholm och Solkartan Göteborg. Information om solenergipotential presenteras i form av ett stapeldiagram med fyra staplar. Varje stapel representerar en klass som baseras på hur god en takytas solpotential är. Från en stapels höjd kan utläsas hur stor area av ett tak som tillhör en viss klass och den totala solinstrålningen över ett år för all takyta i en viss klass presenteras ovanför stapeln. Kartapplikationen är utvecklad genom utvecklingspråket Ajax och bakgrundskartan som används är hämtad från MapQuest. Underlaget för solinstrålningsdata är framtaget med hjälp av ArcMap Solar analyst (avsnitt 3.2.3) (Kraftringen, 2014). Se Figur 4.3 för ett exempel från Solkartan Lund.



Figur 4.3, Bild av solkartan Lund (Krafttringen, 2014).

PVGIS webbapplikation

Data från solinstrålningsmodellen PVGIS (se avsnitt 3.2.2) över Europa går att nå genom en webbapplikation. Webbapplikationen tillåter många olika typer av interaktion och användningen är ganska komplicerad. Applikationen tillåter bland annat att en användare skapar och placerar ut teoretiska solenergisystem som kan modifieras på många olika sätt som lutning och höjd. Applikationen kan sedan genom data från solinstrålningsmodellen PVGIS beräkna det teoretiska solenergisystemets effektivitet vid den platsen (JRC European Commission, 2014). Se Figur 4.4 för ett exempel från PVGIS webbapplikationen.



Figur 4.4, PVGIS webbapplikation med genererad statistik för ett teoretiskt solenergisystem (JRC European Commission, 2014).

4.2 Insticksprogram

Applikationen skulle kunna utvecklas som ett insticksprogram. Ett insticksprogram är ett program som inte körs självständigt utan installeras som en del av ett annat program. Insticksprogrammet fungerar som ett tillägg som tillför ytterligare funktionalitet och kan på så sätt använda kärnprogrammet som ett ramverk för dess utveckling. Ett exempel på ett program som använder sig av insticksprogram är webbläsare där det finns insticksprogram för Adobe Flash Player som tillför möjligheten att visa filer i formatet flash i en webbläsare.

Ett exempel på ett GIS som stödjer insticksprogram är QGIS. Om man utvecklar ett insticksprogram till QGIS får man direkt tillgång till ett applikationsbibliotek. Detta kan underlätta utvecklingen och programmet blir en del av ett redan fungerande GIS som innehåller funktionalitet som panorering och skalhantering av kartor.

4.2.1 Teoretisk uppbyggnad

Ett insticksprogram till QGIS utvecklas i Python. Geografiska objekt som vektorlager, polygoner, linjer och punkter finns redan implementerade och funktioner för att hantera dessa finns. För en mer djupgående inblick i ett insticksprogramms uppbyggnad se avsnitt 5.2.2.

Fördelar

- Spridningen görs enkelt genom att ladda upp insticksprogrammet till ett bibliotek där användare kan välja att ladda hem det.
- Ger automatisk tillgång till kraftfulla utvecklingsstöd.

- Stor frihet i utvecklingen av egen funktionalitet.

Nackdelar

- Kräver att användare installerar program och laddar hem insticksprogram.
- Kan vara svårt att göra användare uppmärksamma på att applikationen existerar
- Svårt att avgränsa och göra applikationens användning tydlig.
- Grafiska användargränssnittet för en stor del av applikationen bestäms av verktyget det kopplas till.

Den påverkan som fördelarna och nackdelarna har på utvecklingen bestäms i stor grad av det program som insticksprogrammet kopplas till. Att få tillgång till ett redan utvecklat applikationsbibliotek med stor frihet kan ses som en fördel för utveckling för alla typer av användare. Denna typ av utveckling gör det också möjligt för användare som är bekanta med programmering att bygga vidare på funktionaliteten genom att modifiera eller skriva ett eget insticksprogram som uppfyller deras behov.

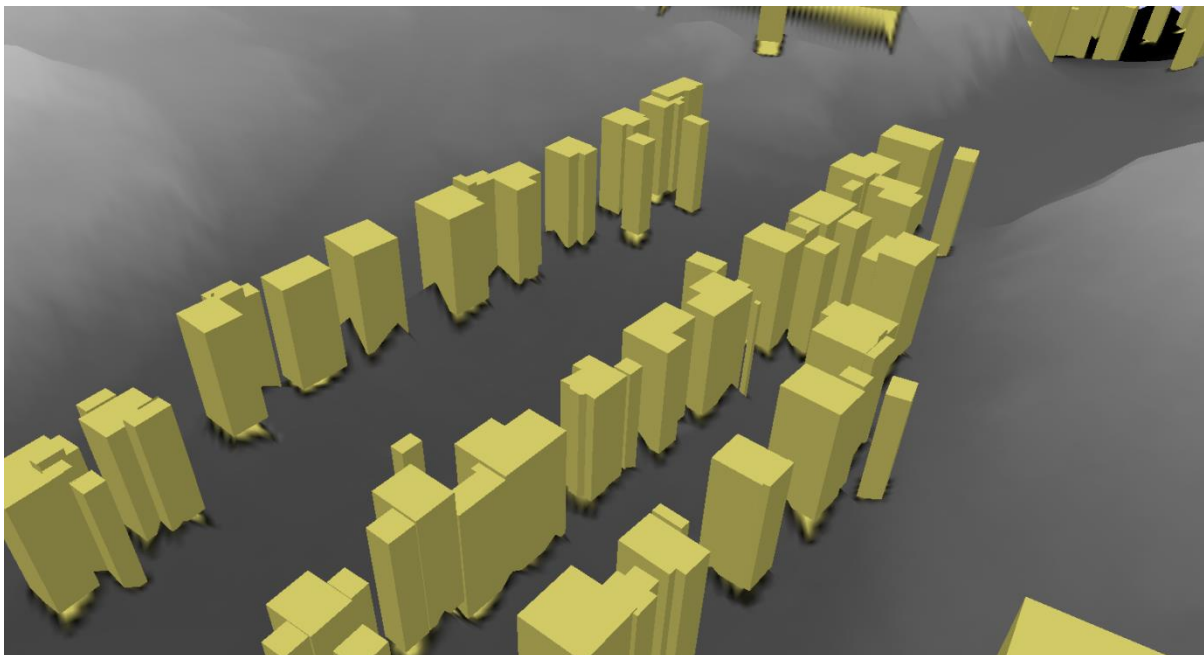
4.2.2 Exempel på applikationer

OpenLayers-insticksprogram

OpenLayers-insticksprogrammet till QGIS kan ses som ett typiskt exempel på ett insticks-program. Verktyget kan enkelt importeras av en användare för att skapa ny funktionalitet som tidigare inte existerade i QGIS. Insticks-programmet skapar när det har initierats en ny meny där en användare kan välja att lägga in bakgrundskartor som stöds av OpenLayers-biblioteket till sina projekt (Arko, 2014).

Qgis2Threejs

Detta insticks-program gör det möjligt för en användare att importera vektordata och rasterdata i form av ytmodell och tillsammans skapa en 3D-modell som exporteras till WebGL och visas för användaren i en webbläsare. Genom att ladda in en ytmodell skapas ett marklager i 3D utifrån de angivna höjderna i ytmodellen. Vektordatan som laddas in kommer att förskjutas i höjded baserat på ett användardefinierat statiskt värde eller utifrån en vald attributtabell ifall en sådan existerar i vektorlagret. Genom denna höjdförskjutning skapas 3D-polygoner som representerar vektordatan. Se Figur 4.5 för ett exempel på en 3D-modell skapat genom Qgis2Threejs.



Figur 4.5, 3D-modell skapad av Qgis2Threejs från ett vektorlager med byggnader och en ytmodell.

ArcMap Solar Analyst

ArcMap Solar Analyst är en typ av solinstrålningsmodell (se avsnitt 3.2.3). Verktöget är en del av GIS-verktyget ArcMap. Det är inte ett insticksprogram men verktöget utökar ArcMaps-funktionalitet på ett motsvarande sätt.

4.3 Fristående applikation

En fristående applikation kan utvecklas från grunden. I detta fall används ett valfritt programmeringsspråk och det grafiska användargränssnittet kan designas utifrån behov.

4.3.1 Teoretisk uppbyggnad

Ett grafiskt användargränssnitt skulle kunna implementeras utifrån programmeringsbibliotek eller utvecklingsverktyg. Det finns många olika befintliga kodbibliotek att välja bland för utveckling och en utvecklare kan välja bland de som lämpar sig bäst för projektet.

Det grafiska användargränssnittet behöver stöd för kartfönster och verktyg. Funktioner och algoritmer måste skapas eller importeras och sedan kopplas till användargränssnittets element. Den färdiga applikationen distribueras sedan till användaren som ett installationspaket som innehåller data, implementationer och eventuella programmeringsspråktolkar och relevanta bibliotek.

Fördelarna med en fristående applikation kan variera mycket beroende på valda utvecklingsmetoder men generellt gäller:

Fördelar

- Ger full frihet för utvecklingen.

Nackdelar

- Kräver att användare installerar program.
- Kräver djupgående kunskap inom flera olika områden av utvecklaren.
- Tidskrävande att utveckla.

Fördelen att kunna skräddarsy applikationen gör att denna arkitektur skulle kunna ses som optimal för alla typer av användare utöver privata användare som har enkla krav på funktionalitet och högre krav på enkel användning. Nackdelarna uppstår istället under utvecklingen då kraven på tid och kunskap är höga.

5. Visualiseringstekniker

Detta avsnitt ger en presentation av teori och metoder för att implementera visualisering och modellering i 3D. För att kunna beskriva hur applikationen fungerar och diskutera olika typer av problem som uppstår måste en viss bakgrundsinformation presenteras. Applikationsarkitekturen som applikationen ska utvecklas genom måste också ha någon typ av stöd för en visualiseringsteknik.

Vid utveckling av olika typer av applikationer som innehåller rendering av 3D-grafik är nästan alltid något av applikationsbiblioteken OpenGL eller Direct3D involverade. Utveckling med Direct3D kräver kommersiell licens som inte har kunnat göras tillgänglig för detta projekt. Därför kommer detta avsnitt att fokusera på OpenGL som är släppt under licens med öppen källkod (Wikipedia, 2014a).

5.1 OpenGL

OpenGL är ett applikationsbibliotek utvecklat av Silicon Graphics Incorporated. Det sköts nu av det icke-vinstdrivande konsortiet Khronos group. OpenGL är släppt under öppen källkod och fokuserar på rendering av 2D- och 3D-vektor grafik. Rendering uppnås genom kommunikation med datorns grafiska processor (GPU) för att maximera effektiviteten (Angel och Shreiner, 2012).

5.1.1 Primitiver

En primitiv enhet är ett grundläggande byggnadsblock i 3D-programmering där varje ny primitiv tillför ytterligare funktioner. På den mest grundläggande nivån existerar endast tre primitiver: punkter, linjesegment och trianglar där varje nivå ger tillgång till en ny dimension. Genom dessa approximeras sedan mer avancerade former som kurvor, sfärer eller byggnader (Angel och Shreiner, 2012). Se figur 5.1 för exempel på de tre nödvändiga primitiverna för att kunna nå tre dimensioner.



Figur 5.1, Nödvändiga primitiver för 3D-grafik.

Det har länge pågått en debatt ifall man bör sträva efter att hålla sig till en liten mängd primitiver eller implementera flera primitiver. Fördelen med att låta mer komplexa objekt bestå av approximerade trianglar är att relativt enkel matematik som triangulering kan användas i en datorns maskinvara (GPU:n) och öka prestandan. Allt fler system övergår till en form som utnyttjar triangulering (Angel & Shreiner, 2012).

Anledningen till att primitiver är intressanta för detta arbete är att SEBE:s utdata består av en mängd punkter vilka borde vara relativt enkelt att läsa in och omvandla med någon typ av 3D-rendering.

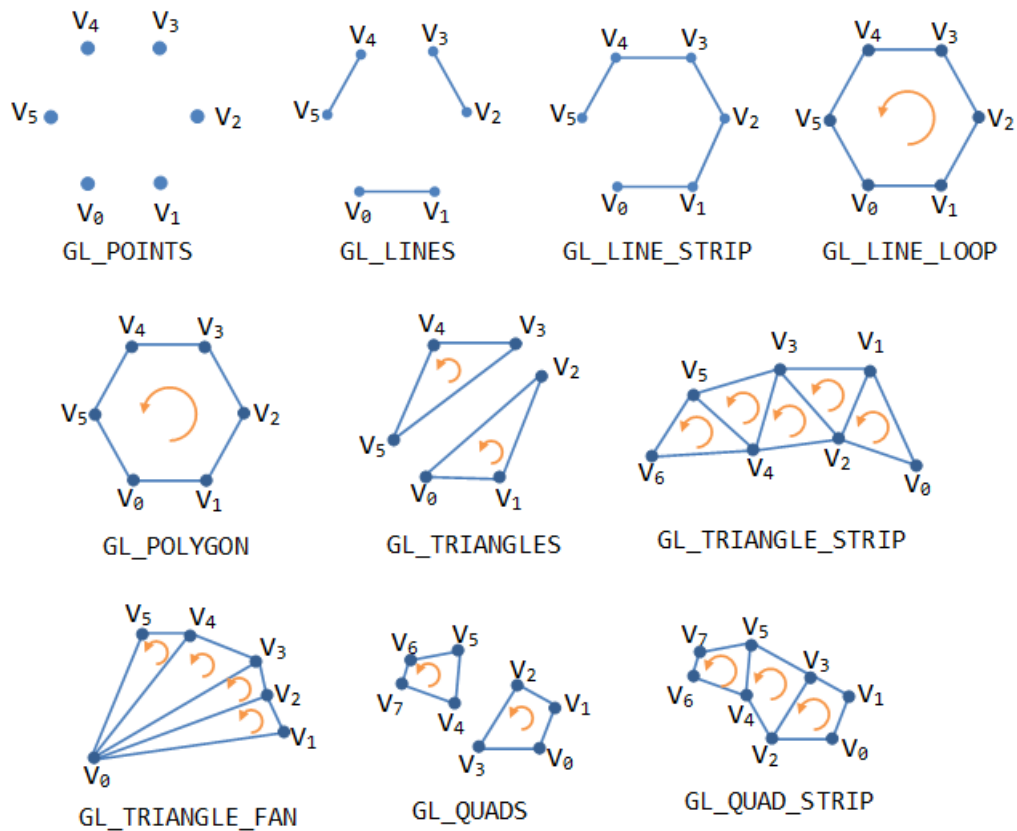
5.1.2 Vertices

En *vertex* är inom grafisk programmering ett objekt som kan ha flera attribut varav ett måste vara en position. En samling *vertices* kan användas för att definera plana ytor, typiskt trianglar, som sedan bygger upp ett mer komplext objekt (Wikipedia, 2014c).

I OpenGL skapas en tredimensionell kub genom att först definiera hur de plana ytorna utformas utifrån en samling *vertices*. *Vertices* i samlingen skulle definiera hörnen på kuben och genom målningsmetoder skulle hörnen bindas ihop och de plana ytorna målas upp. De olika metoderna som finns är (en *vertex* beskrivs som en punkt):

- *GL_POINTS* – Varje *vertex* blir en punkt utan koppling till någon annan *vertex*.
- *GL_LINES* – Varje punktpar bildar ett självständigt linjesegment.
- *GL_LINE_STRIP* – Det första punktparet skapar ett linjesegment, följande punkt i samlingen skapar ett nytt linjesegment från den tidigare punkten i samlingen.
- *GL_LINE_LOOP* – Som *GL_LINE_STRIP* men sista punkten binds ihop med första punkten.
- *GL_TRIANGLES* – Från samlingen hämtas tre punkter som binder ihop en självständig triangelyta.
- *GL_TRIANGLE_STRIP* – De tre första punkterna i samlingen skapar en triangelyta, följande två punkter i samlingen skapar en ny triangel med den senaste punkten i samlingen.
- *GL_TRIANGLE_FAN* – De tre första punkterna i samlingen skapar en triangelyta, följande punkt i samlingen skapar en ny triangel med de två tidigare punkterna i samlingen.
- *GL_QUADS* – Som *GL_TRIANGLES* men skapar en rektangel från fyra punkter.
- *GL_QUAD_STRIP* – Som *GL_TRIANGLE_STRIP* men rektanglar skapar utifrån två tidigare punkter och två nya punkter.
- *GL_POLYGON* – Utifrån punktsamlingen skapas en konvex polygon.

Polygonuppritningsmetoderna är OpenGL:s primitiver (Silicon Graphics, Inc. 2006). Det vill säga de grundläggande polygoner som kan användas för att bygga upp objekt (se avsnitt 5.1.1). Se Figur 5.2 för grafiskrepresentation av primitiverna.



OpenGL Primitives

Figur 5.2, OpenGL primitiver (Chua Hock-Chuan, 2012).

Ytorna kommer att färgläggas utifrån de färger som tilldelats en ytas *vertices* och interpoleras över hela ytan. Pilarna i Figur 5.2 visar ordningen som *vertices* ska läggas in för att korrekt representera ytans utsida. Detta kan sedan användas för att dölja ytor vars utsida är vända bort ifrån betraktaren och på så sätt minska antalet beräkningar för en modell. Figur 5.3 visar OpenGL-kod som definierar och ritar upp nödvändiga *vertices* för att skapa en tredimensionell kub.

Kodexempel för att skapa en kub i OpenGL:

```
glBegin(GL_QUADS); # Börjar måla upp rektangulära ytor.

glColor3f(0.0,1.0,0.0);
# Sätter färgen för inmatade vertices till grön.

glVertex3f( 1.0, 1.0,-1.0);
glVertex3f(-1.0, 1.0,-1.0);
glVertex3f(-1.0, 1.0, 1.0);
glVertex3f( 1.0, 1.0, 1.0);
# Definerar en rektangel.

glVertex3f( 1.0,-1.0, 1.0);
glVertex3f(-1.0,-1.0, 1.0);
glVertex3f(-1.0,-1.0,-1.0);
glVertex3f( 1.0,-1.0,-1.0);
# Definerar en rektangel.

glVertex3f( 1.0, 1.0, 1.0);
glVertex3f(-1.0, 1.0, 1.0);
glVertex3f(-1.0,-1.0, 1.0);
glVertex3f( 1.0,-1.0, 1.0);
# Definerar en rektangel.

glVertex3f( 1.0,-1.0,-1.0);
glVertex3f(-1.0,-1.0,-1.0);
glVertex3f(-1.0, 1.0,-1.0);
glVertex3f( 1.0, 1.0,-1.0);
# Definerar en rektangel.

glVertex3f(-1.0, 1.0, 1.0);
glVertex3f(-1.0, 1.0,-1.0);
glVertex3f(-1.0,-1.0,-1.0);
glVertex3f(-1.0,-1.0, 1.0);
# Definerar en rektangel.

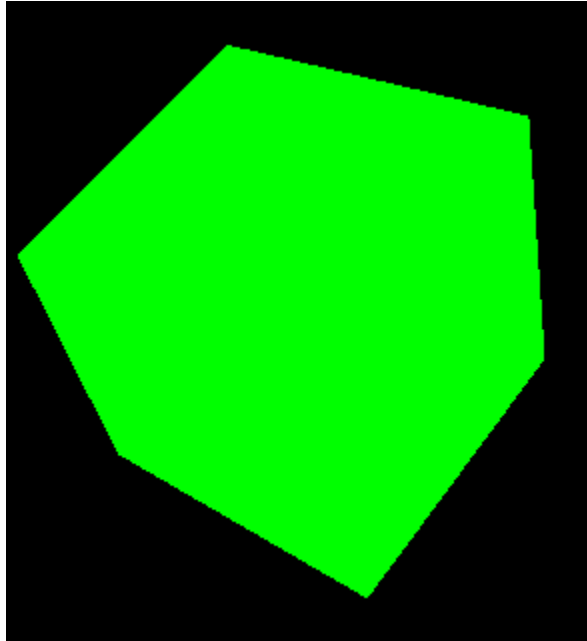
glVertex3f( 1.0, 1.0,-1.0);
glVertex3f( 1.0, 1.0, 1.0);
glVertex3f( 1.0,-1.0, 1.0);
glVertex3f( 1.0,-1.0,-1.0);
# Definerar en rektangel.

glEnd();

# Avslutar uppmålningen.
```

Figur 5.3, Kod för att skapa en tredimensionell kub.

Figur 5.4 visar resultatet från kodexemplet



Figur 5.4, Kub från kodexempel (roterad för tydlighet).

5.1.3 Shaders

Shaders kan ses som program vilka kan appliceras på olika objekt för att hantera dess attribut. De kan vara allt från enkla program som bara applicerar en placering och en färg på en pixel till extremt komplexa program.

Det finns olika typer av *shaders*, två av dem är *vertex shader* och *pixel shader*. En *vertex* används för att bygga upp objekt (se avsnitt 5.1.2). Objekt byggs upp av plana ytor som till exempel trianglar där varje hörn kallas för en *vertex*. Dessa hanteras av en *vertex shader* som på sin mest grundläggande nivå endast skickar vidare en koordinat för en *vertex*-position på skärmen. Andra attribut kan också appliceras och skickas vidare. Attributen interpoleras sedan över triangeln och skickas vidare till *pixel*-shadern som hanterar varje pixels visualisering där den mest grundläggande formen endast har en färg som utdata.

Shaders går att utveckla till mer komplicerade program som utför beräkningar och förändringar på renderingen av ett objekt. Detta gör det möjligt att skapa verklighetstroga ytor och effekter genom algoritmer och matematiska operationer. Till exempel kan en samling *vertices* som bildar en plan yta renderas som ett böljande hav genom *shader*-program. *Vertices* hade för detta exempel skickats till en *vertex shader* som över tiden förskjutit punkterna för att skapa vågor. Placeringen av en *vertices* hade sedan skickats vidare till en *pixel shader* som skulle ha hanterat attribut som färg, reflektion och genomskinlighet beroende på punktens position.

5.2 WebGL

WebGL är ett applikationsbibliotek baserat på OpenGL ES 2.0, en version av OpenGL som fokuserar på utveckling av mobila applikationer. WebGL gör det möjligt att genom *canvas*-element i HTML 5.0 skapa webbapplikationer med 3D-grafik komponenter.

Eftersom WebGL är baserat på ett språk framtaget för utveckling till mobila enheter leder detta till sämre stöd för stora miljöer som består av många *vertices* (se avsnitt 5.1.2). Det finns inom 3D-programmering många sätt att på olika sätt optimera en applikations prestanda men det kräver tidskrävande lösningar som ibland har en negativ effekt på applikationens visuella utseende. SEBE:s stora mängd punktdata gör att det kan tänkas uppstå problem vid rendering av stora miljöer som inkluderar flera byggnader om inte effektiva lösningar kan implementeras.

5.3 PyOpenGL

PyOpenGL är ett så kallat *wrapper library*, det vill säga en samling moduler som tolkar kod och tillåter att till exempel programmeringsspråk som använder olika datatyper kan samarbeta. PyOpenGL gör det möjligt att skriva OpenGL applikationer genom Pythonkod. Det finns flera olika versioner av PyOpenGL där versionerna 3.X är mer robusta och samtidigt mer krävande vad gäller datorprestanda än föregående versionerna (2.x) (Sourceforge, 2014b).

6. Val av teknisk miljö

Detta avsnitt motiverar den valda utvecklingsmetoden för prototyp av applikationen. Efter motiveringen följer en beskrivning av underliggande arkitektur för utvecklingsmetoden.

6.1 Motivering

Valet av den tekniska miljön motiverades utifrån de behov som identifierades för olika användargrupper i behovsanalysen (se avsnitt 2.1) och de tekniska krav som tagits fram för applikationen i kravspecifikationen (se avsnitt 2.2). Den applikationsarkitektur (se avsnitt 4) som valts måste fungera väl i kombination med visualiseringsteknikerna som presenterades i avsnitt 5. Utöver detta togs hänsyn till den tidsbegränsning som ligger på arbetet.

En webbapplikationsarkitektur hade varit lämpligast för privata användare med låga krav på funktionalitet men hade begränsat de användargrupper som har mer avancerade behov. Det finns flera redan existerande applikationer som till viss del uppfyller de privata användarnas behov (se avsnitt 4.1). En webbapplikation hade begränsat möjligheterna för visualiseringstekniker eftersom det huvudsakliga stödet ligger i WebGL vilket är en något begränsad visualiseringsteknik (se avsnitt 5.2). Två liknande lågnivå-applikationer framtagna i OpenGL respektive WebGL testades i ett tidigt skede av arbetet mot varandra genom att visualisera en stor mängd data. Testresultatet visade att det uppstod betydligt fler svårigheter för WebGL och applikationen blev problematisk att köra vid informationsmängder motsvarande tre medelstora byggnader i SEBE.

Arkitekturerna hos en fristående applikation och ett insticksprogram liknar varandra på många sätt men med den ökade komplexitet som krävs för utvecklingen av en fristående applikation ökar också tidsåtgången.

Utifrån ovanstående motiveringar valdes insticksprogram till befintligt GIS som arkitektur för utveckling av visualiseringsapplikationen. Ett insticksprogram minskar inte valmöjligheterna för olika visualiseringstekniker och är lämplig för alla typer av användare, utom möjligen för den privata användaren. Insticksprogram tillåter att avancerade användare utvecklar egna komplementär applikationer genom ytterligare insticksprogram. Ifall man i framtiden väljer att utveckla en fristående applikation, skulle de metoder som används av insticksprogrammet också fungera.

Privata användare är en stor grupp som till stor del utesluts av beslutet att utveckla ett insticksprogram. Rapporten innehåller därför en diskussion om hur en liknande visualiseringsapplikation hade kunnat utvecklas för webben.

När insticksprogram beslutats vara en lämplig applikationsarkitektur var valet tvunget att expanderas till det ramverk till vilket insticksprogrammet skulle vara kopplat. Av ramverket krävdes möjligheter att uppfylla de tekniska kraven i kravspecifikationen och stöd för OpenGL.

QGIS är ett GIS utvecklat med öppen källkod som ger stöd för utveckling och spridningen av insticksprogram. Genom redan implementerade funktioner i QGIS kan arbetsbördan minskas för hantering av geografiska data. QGIS ger genom sitt applikationsramverk stöd för både OpenGL och möjligheten att utveckla egna användargränssnitt. QGIS uppfyller också direkt kraven 1-7 och delvis krav 9 i kravspecifikationen. Detta gjorde QGIS till ett lämpligt ramverk för insticksprogrammet.

6.2 Beskrivning av valt ramverk

6.2.1 QGIS

QGIS, tidigare Quantum GIS, är ett geografiskt informationssystem. QGIS är i grunden skrivet i C++ med applikationsramverket Qt och består av ungefär 400 kärnklasser som bygger upp verktyget. Utöver detta finns en mängd användarutvecklad funktionalitet i form av insticks-program vilket gör att QGIS i många aspekter är lika kraftfullt som ett motsvarande kommersiellt geografiskt informationssystem.

Det är detta stöd för användarutvecklad funktionalitet som gör QGIS intressant för denna studie. Genom QGIS kan en applikation skapas där man enkelt får tillgång till kraftfullt stöd för hantering av geografiska data. Systemet fungerar som ett fundament för att skapa nya användningsområden (Sherman, 2014).

6.2.2 QGIS – insticksprogram

Tidigt under utvecklingen av QGIS skapades en arkitektur för insticksprogram. Detta tillåter användare att skräddarsy QGIS för egna behov genom att hämta insticksprogram som ökar funktionaliteten eller genom att utveckla egna insticksprogram.

Under år 2007 lades stöd till för utveckling av det dynamiska och lättlästa programmeringsspråket Python. Stödet sker genom PyQt som är ett tolkningsstöd från Qt till Python. Detta ger tillgång till QGIS grundklasser i Python och gör det möjligt att skriva Python-kommandon som sedan tolkas till det c++ ramverk som QGIS är skrivet i.

Insticksprogram kan i QGIS ingå i två olika grupper, kärnprogram och tillförda program. Kärnprogram ingår vid installation av QGIS och tillförda program måste hämtas från en datakatalog för att en användare ska få tillgång till funktionaliteten. Ifall ett tillfört insticks-program anses ge en grundläggande funktionalitet som är användbar för ett stort antal användare kan detta insticks-program bli ett kärnprogram (Sherman, 2014).

Vid utveckling av ett insticksprogram är det främst fyra olika moduler en utvecklare arbetar emot för att nå QGIS-funktionalitet:

- Qgis.core – Används för att nå kärnklasser. Kan användas för att till exempel skapa och hantera vektorlager.
- Qgis.gui – Används för att nå klasser som hanterar det grafiska användargränssnittet.
- Qgis.analysis – Används för att nå klasser relaterade till analys.
- Qgis.networkanalysis – Används för att nå klasser för nätverksanalys.

Ett insticksprogram består av en mapp som på sin mest grundläggande nivå innehåller ett relativt stort antal filer, några av de viktigare är:

- `__init__.py` – Det här scriptet innehåller endast en klass som initierar insticksprogrammet och gör det känt för QGIS.
- `Metadata.txt` – Innehåller metadata, till exempel krav på version av QGIS. Krävs för att QGIS ska känna igen insticksprogrammet.

- Plugin.py – Där plugin ändras till namnet på det specifika insticks-programmet. Detta är den fil som innehåller själva implementationen och utför det huvudsakliga arbetet med funktionerna.
- Resources.qrc – Innehåller referenser och beskrivningar av de resurser som insticks-programmet och dess användargränssnitt använder sig av.
- Ui_plugin.ui – Där plugin ändras till namnet på det specifika insticks-programmet. Filen för hantering av det geografiska användargränssnittet.

Det skapas också en mängd andra filer och vid distribution krävs att mappen med underliggande filer paketeras som en zip-fil (Sherman, 2014).

Tidigare krävdes att en utvecklare skapade filerna för insticksprogram arkitekturen själv men nu finns det flera hjälpverktyg som till exempel plugin builder (vilket i sig är ett insticksprogram till QGIS) där en utvecklare endast behöver fylla i formulär med den relevanta informationen som namn och versionsnummer för ett insticks-program för att generera alla nödvändiga filer. Efter att ha kompilerat några av de skapade filerna finns nu en fungerande arkitektur för att lägga till eftersträvd funktionalitet.

6.2.3 QGIS - Fristående applikation

QGIS API:et och Qt kan användas för att skapa en fristående applikation där till exempel bara verktyg som är relevanta kan inkluderas. Detta kan ses som mycket användbart eftersom användningen förenklas och om distribueringen är tänkt att nå ut till användare som är ovana vid ett geografiskt informationssystem kan denna förenkling vara nödvändig.

Om man vill skapa en fristående applikation krävs utöver implementationer av klasser för funktionalitet, också extra arbete för att ta fram ett grafiskt användargränssnitt. Detta kan göras genom att använda sig av verktyget Qt Designer eller genom extra PyQt-kod. Applikationen hade sedan kunnat samla alla nödvändiga pythonbibliotek, kod och data i ett installationspaket som sedan skulle användas för att distribuera applikationen och vara helt fristående.

6.2.4 QGIS – GRASS

GRASS (Geographic Resources Analysis Support System) är ett GIS till en början utvecklat av den amerikanska arméns ingenjörer som nu har expanderat till ett multinationellt projekt styrt av GRASS Project steering committee.

Genom ett insticksprogram till QGIS kan över 400 moduler i GRASS biblioteket nås även genom QGIS. Denna utökade funktionalitet inkluderar bland annat 3D-raster analys samt 3D-visualisering genom modulen NVIZ (GRASS Development team, 2012). GRASS 3D-kartor genereras genom så kallade *voxlar* (*volym pixlar*). En *voxel* är en tredimensionell kub som tilldelas en rumslig plats och ett värde (GRASS Development team, 2013a).

3D-kartorna kan genereras från vektordata, ytmodeller eller genom en ASCII-fil. ASCII metoden är den som tillåter att en användare manuellt skapar den data som ska visualiseras i 3D. GRASS modulen r3.in.ascii är den som används och de indata som krävs liknar ett ASCII-grid (jfr avsnitt 3.3.1). 3D-kartans attribut beskrivs i en header. De nödvändiga attributen är:

```

north: Slutkoordinat för y.
south: Startkoordinat för y.
east: Slutkoordinat för x.
west: Startkoordinat för x.
top: Slutkoordinat i höjdnivå.
bottom: Startkoordinat i höjdnivå.
rows: Antalet rader.
cols: Antalet kolumner.
levels: Antalet höjdnivåer.

```

Efter headern följer alla rastrets värden vilka avskiljs med blanksteg och radbrytningar. Varje rad som skapas kommer att ha konstant y-koordinat och konstant höjdnivå (GRASS Development team, 2013b).

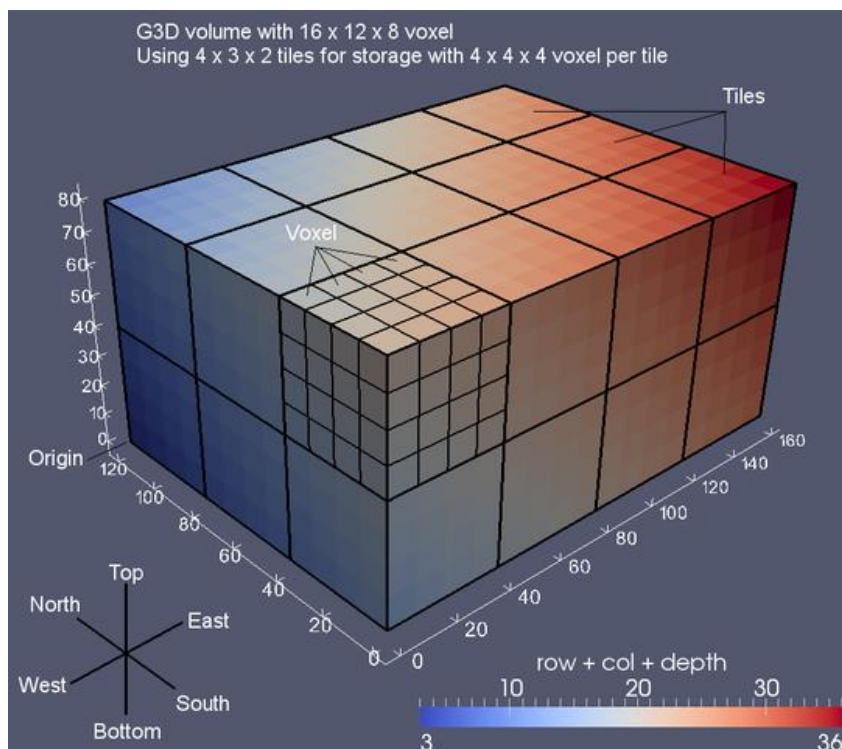
Exempel på värden i en fil för indata med tre rader, två kolumner och två höjdnivåer:

```

VärdeX1Y1Z1 VärdeX2Y1Z1 VärdeX3Y1Z1
VärdeX1Y2Z1 VärdeX2Y2Z1 VärdeX3Y2Z1
VärdeX1Y1Z2 VärdeX2Y1Z2 VärdeX3Y1Z2
VärdeX1Y2Z2 VärdeX2Y2Z2 VärdeX3Y2Z2

```

För en grafisk representation av ett 3D-raster i GRASS se figur 6.1.



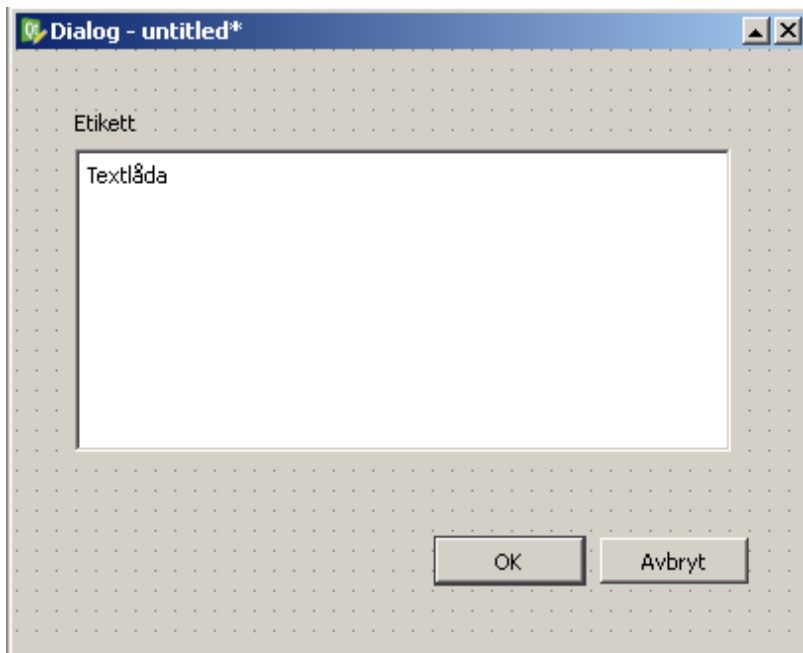
Figur 6.1, Grafisk representation av ett 3D-raster (GRASS Development team, 2013b).

6.2.5 Qt Designer

Qt Designer är ett verktyg för att skapa ett grafiskt användargränssnitt av Qt-kompatibla

komponenter. QGIS är skrivet i Qt och är därför kompatibelt med Qt Designer; detta innebär att det är möjligt att skapa ett grafiskt användargränssnitt för en självständig applikation som inkluderar QGIS eller en grafisk komponent till ett insticksprogram.

Qt Designer är ett grafiskt programmeringsverktyg där en användare kan skapa grafiska objekt som knappar, fönster och etiketter. Ett knapptryck kan sedan till exempel kopplas till en metod i applikationskoden som i sin tur modifierar en etikett i användargränssnittet (Qt Project, 2014). Figur 6.2 visar användargränssnittskomponenter framtagna i Qt.

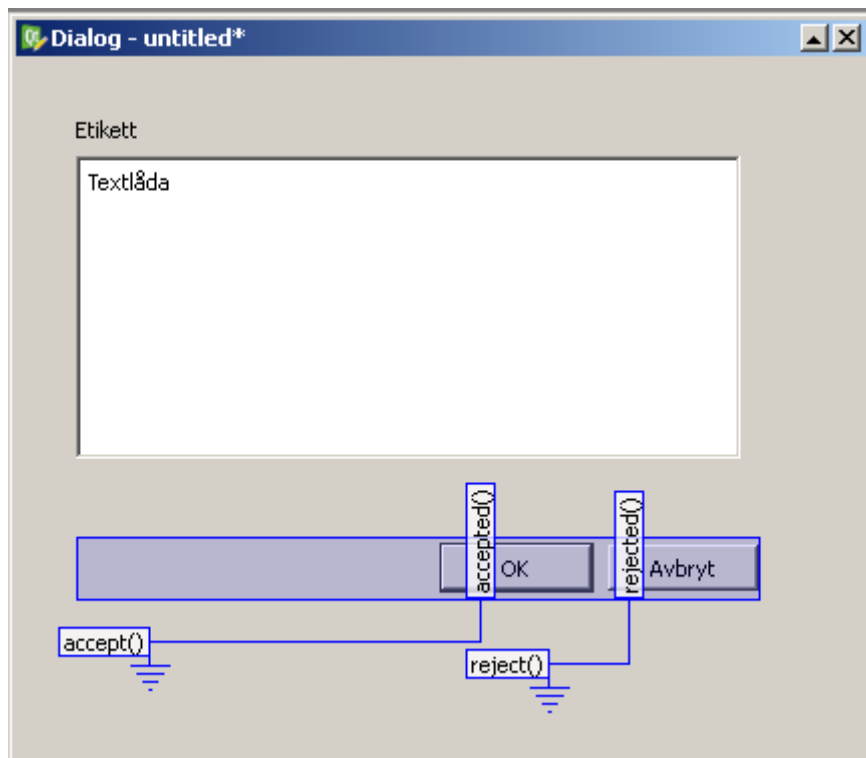


Figur 6.2, Enkla grafiska komponenter skapade i Qt Designer.

Kopplingar mellan olika grafiska komponenter och metoder i applikationskoden skapas genom signaler och öppningar. När en signal aktiveras genom till exempel ett knapptryck kan den skickas till en öppning på ett annat objekt vilket genererar ett resultat eller en reaktion.

I figur 6.3 visas grafiska komponenter med signaler och öppningar. När OK-knappen aktiveras skickas signalen `accepted()` till hela fönstret som tar emot det i öppningen `accept()`. När Avbryt-knappen aktiveras skickas signalen `rejected()` till öppningen `reject()` i fönstret.

Beroende på vilken signal som skickas och tas emot berättar det för applikationen vilket arbete som ska utföras. Tas signalen `accepted()` emot skulle det kunna säga åt applikationen att ta informationen i textlådan och skicka vidare det i applikationsarkitekturen medan signalen `rejected()` stänger fönstret och raderar all information i textlådan. Det går också att utveckla skräddarsydda signaler och öppningar.



Figur 6.3, Grafiska komponenter med signaler.

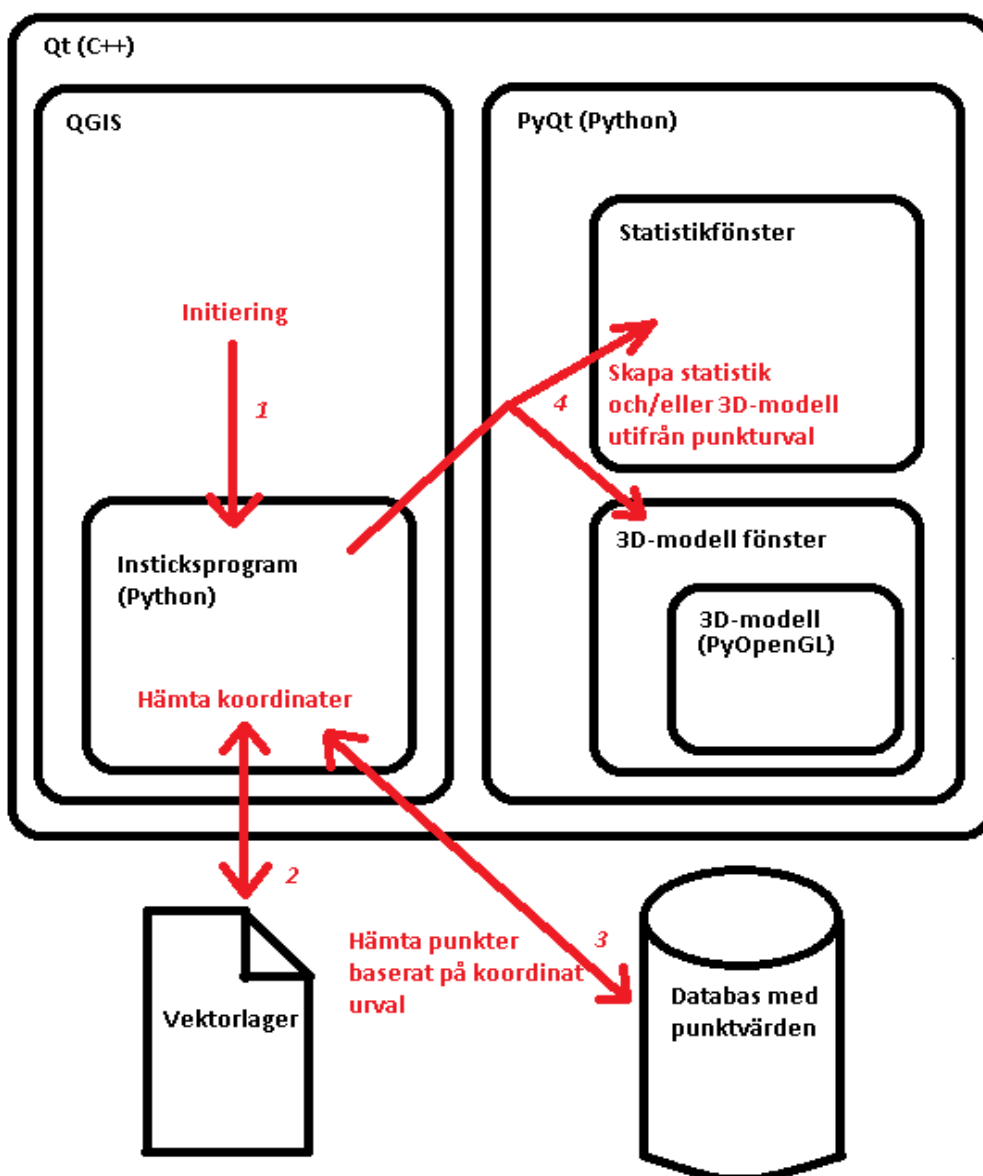
En grafisk komponent som har skapats i Qt Designer kan kompileras till Python-script.

7. SEBEvisualizer

SEBEvisualizer blev det arbetsnamn som har tilldelades applikationen. Detta avsnitt beskriver den data som används av SEBEvisualizer, applikationens arkitektur och de olika moment som genomförts för att skapa den. Vid varje moment beskrivs de valmöjligheter som funnits vid utvecklingen, likaså redovisas motiven för de val som har gjorts. Ett användarscenario för SEBEvisualizer presenteras i Appendix A.

7.1 Applikationens arkitektur

Figur 7.1 beskriver SEBEvisualizers arkitektur och hela arbetsflödet från initiering till resultatet som resultatpresentation.



Figur 7.1, Applikationens arkitektur och arbetsflöde.

Applikationens stomme är applikationsramverket Qt i vilket QGIS är skrivet. SEBEvisualizers största arbete utförs i insticksprogrammet utvecklat som ett tillägg till QGIS, insticksprogrammet består av en mapp med flera filer som innehåller kod främst skriven i Python (se avsnitt 6.2.2). Mappen innehåller också vektordata och för tillfället databasen som SEBEvisualizer använder sig av. Insticksprogrammet använder sig av redan utvecklade metoder i QGIS applikationsbibliotek för att ta fram koordinater för objekt i vektorlagret med byggnader. Utifrån dessa koordinater görs urval från databasen som innehåller punktvärden genererade av SEBE. Insticksprogrammet skickar vidare informationen från databasen till två Qt-dialoger i vilka informationen tas om hand och presenteras för användaren. Qt-dialogen för 3D-modellen innehåller ytterligare en Qt-komponent kallad *QGLWidget*. Denna typ av komponent kan hantera och rendera OpenGL-kod och det är i denna *QGLWidget* som 3D-modellen skapas och visas för användaren.

Förenklat kan arbetsflödet sammanfattas som följande (se Figur 7.1):

1. Initering av insticksprogram i QGIS.
2. Insticksprogrammet hämtar information från vektorobjekt.
3. Insticksprogrammet använder information från vektorobjekt för att hämta data från databasen.
4. Datan skickas från insticksprogrammet till Qt-dialoger för att bearbetas och presenteras för användaren.

7.2 Visualiseringsapplikationens indata

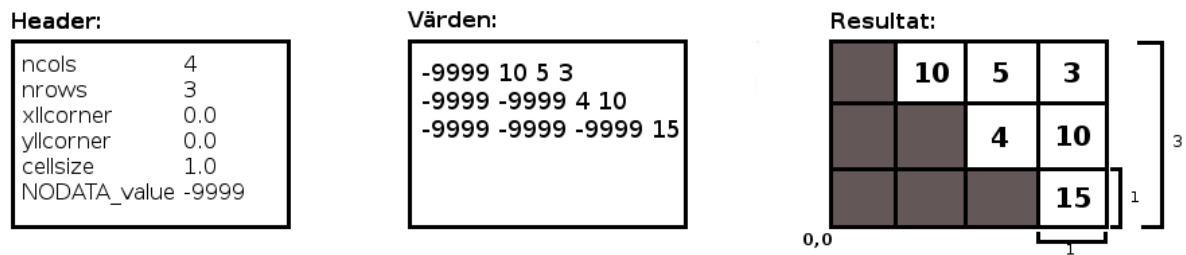
Detta avsnitt redogör för data som ligger till grund för arbetet med visualiseringsapplikationen. Det rör sig främst om de utdata som SEBE genererar men även vektorlager som används för att skapa ett topologiskt samband mellan punktdatan som SEBE skapar. Avsnittet redogör också för de avvikelser som kan minska på resultatets säkerhet och de beräkningar som måste utföras för att få korrekta resultat.

Då datan från SEBE är en del av grunden för applikationen är det viktigt att den presenteras mer ingående innan utvecklingen av applikationen diskuteras i detalj. Utdata från SEBE kan publiceras i två olika format, som rasterdata eller textfil. Rasterdatan genereras antingen som en TIFF-fil eller som ett ASCII-grid. Speciellt intressant för detta arbete är textformatet eftersom det enklast kan användas i arbetet med 3D-grafik (se avsnitt 5) då värden för solinstrålning presenteras tillsammans med koordinater i textfilerna. För rasterfilerna knyts istället värdena till koordinater genom metadata.

7.2.1 ASCII-grid

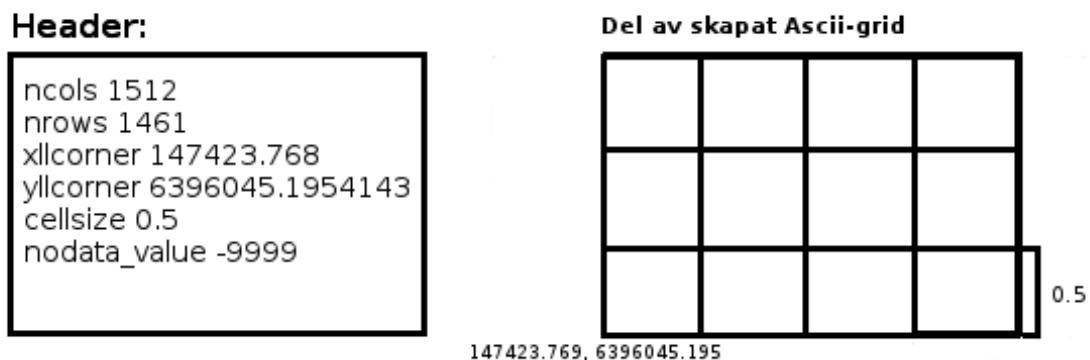
Ett ASCII-grid är en typ av rasterdata där rastrets utformning definieras i en *header*. Rastrets koordinater definieras av punkten i hörnet längst ner till vänster, dvs. i rastrets mest sydvästligt belägna cell. Rastrets utsträckning i östlig riktning definieras av antalet kolumner och i nordlig riktning av antalet rader samt storleken på varje cell. Det definieras också ett värde som används för att beskriva en tom cell som appliceras på de celler som saknar ett mätvärde.

Efter *headern* följer sedan varje cells värde där kolumner skiljs av med blanksteg och rader med en radbrytning (Esri inc, 2013). Se figur 7.2 för exempel.



Figur 7.2, ASCII-grid exempel.

Se Figur 7.3 för solinstrålningsmodellens specifikationer.



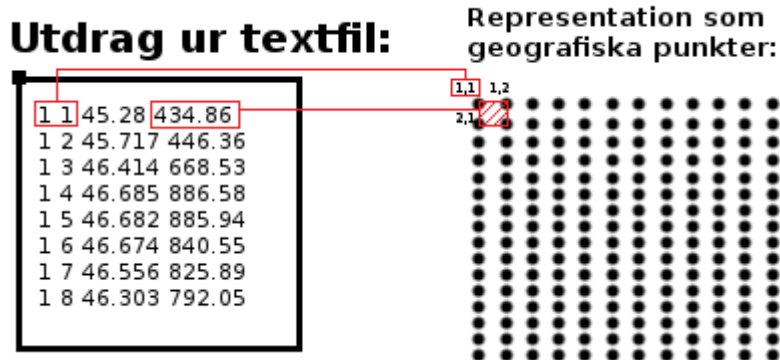
Figur 7.3, ASCII-grid skapat av solinstrålningsmodellen.

Från headern på ASCII-gridet kan man se att ytan som täcks av rastret innehåller 1512 * 1461 antal celler där varje cell har storleken 0.5 * 0.5 enheter. Ytan är definerad i koordinatsystemet SWEREF 99 12 00 vilket innebär att en enhet är 1 meter, alltså innehåller en kvadratmeter fyra stycken celler och värden. Koordinaterna är definerade som *N* för nordlig riktning samt *E* för östlig (Lantmäteriet, 2014b).

7.2.2 Textfiler

Solinstrålningsmodellen genererar tre olika text-filer i ett lokalt koordinatsystem. En textfil för punkter på plana ytor, en för fasader och en för vegetation. Området som innefattas har samma utsträckning som det genererade ASCII-gridet (se avsnitt 7.2.1) och punkterna stämmer överens med de värden som finns där med samma längd mellan varje punkt. Skillnaderna är att koordinaterna startar i punkten i det nordvästra hörnet och med värdena 1,1 (*y, x*). Att översätta det lokala koordinatsystemet till det geodetiska referenssystemet SWEREF 99 12 00 kräver att det görs en förskjutning på 0.5 meter i sydlig riktning. Punkterna i text-filen kan sedan kopplas till koordinaterna som definieras i ASCII-gridets *header* för att översättas till SWEREF 99 12 00.

I filerna representerar varje rad en punkt. Varje punkt har en Y-koordinat som representerar nordlig riktning, en X-koordinat som representerar östlig riktning samt en Z-koordinat som representerar höjd. Till varje punkt knyts ett värde för kilowattimmar (kWh). En rad har formatet: $Y X Z kWh$. Se figur 7.4 för ett exempel på utdrag ur textfilen och en geografisk representation av punkterna.



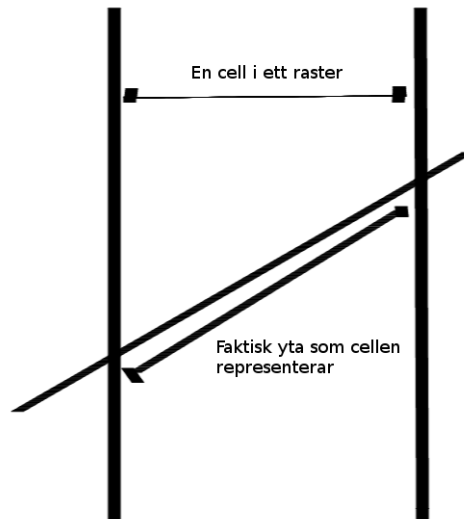
Figur 7.4, Exempel på utdrag ur textfil och representation som punkter. För djupare förklaring se text.

Höjdkoordinaterna är definierade i RH 2000, Sveriges nationella höjdsystem som är definierat av 50 000 punkter. RH 2000 täcker flera länder och nollpunkten är placerad i Amsterdam (Lantmäteriet 2014a).

Z-värdena är främst intressanta i fasad-filen. Flera punkter kan i denna fil ha samma y - och x -koordinat där Z -värdena skiljer dem åt vilket inte är möjligt med ASCII-gridet. Punkterna börjar räknas från den punkt där fasaden möter marknivån och ökar sedan i höjd med 1 meter per punkt. Fasaderna har för en byggnad uppskattats från den yttersta punkten för ett tak och sedan beräknats som att de därifrån har en 90 graders riktning tills de når marken. Detta innebär att de endast är approximationer och bör ha en större avvikelse från verkligheten än annan utdata. Man kan i den geografiska representationen av punkterna i Figur 7.4 se att uppbyggnaden i plan utsträckning är identisk den hos ett raster. Varje punkt som representeras av en rad har ett bestämt avstånd till omkringliggande punkter vilket gör att varje punkt också kan ses som att den representerar en utsträckning likt en cell i ett raster.

7.2.3 Korrekt cellarea

Varje cell har i rasterlagren en horisontell utsträckning av $0,25 \text{ m}^2$ (se avsnitt 7.2.1). Denna area stämmer bara ifall den yta en cell representerar är helt vinkelrät sett ovanifrån. I verkligheten är detta scenario väldigt ovanligt och höjder i en cell fluktuerar. En bättre approximation på ytan under en cell kan tas fram av SEBE genom trigonometri eftersom höjderna i en cells hörn genom ytmodeller är kända. Utifrån detta kan ett system med trianglar med känd höjd och bredd tas fram och användas för att beräkna hypotenusor vilket är en närmare approximation av ytan under en cell. Ska statistik kopplas till m^2 genereras en areakonstant av SEBE som kan multipliceras med cellens utsträckning för att få ut den korrekta arean. Se Figur 7.5 för illustration av skillnaden mellan en cells utsträckning och den faktiska ytan som ligger under den.



Figur 7.5, Illustration av skillnaden mellan längden på en linje i en cell och en faktisk linje i verkligheten.

7.2.4 Vektorlager

Vektorlagret som finns till grund för applikationen har tillhandahållits från Göteborgs kommun och innehåller vektordata för byggnader inom fallstudiens område. Vektorlagret är definierat i samma koordinatsystem som ASCII-gridet, SWEREF 99 12 00. Figur 7.6 visar vektordata för testområdets utsträckning.

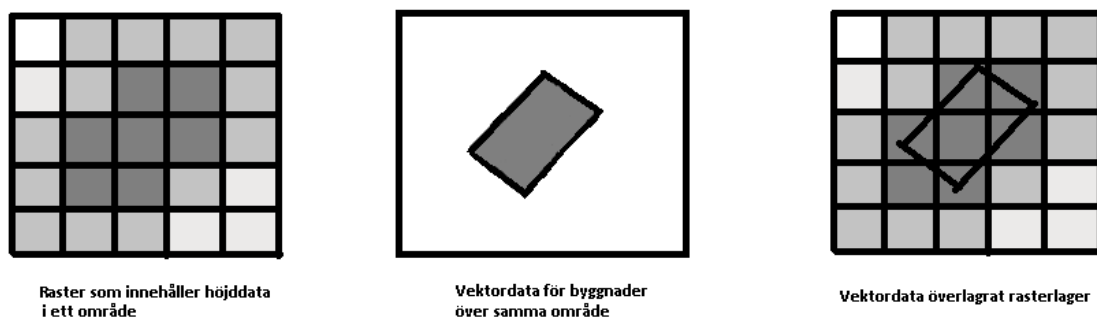


Figur 7.6, Figuren visar till vänster vektordata för byggnader inom testområdet som användes i fallstudien och till höger områdets belägenhet i Göteborg stad (Göteborgs kommun).

Vektordata beskrivs genom punkter, linjer och polygoner. Flera punkter kan länkas samman för att skapa linjer och om linjernas bana sluts skapas en polygon (Gisela, 2014).

7.2.5 Sammankoppling av vektor- och rasterdata

Ett raster bygger aldrig upp objekt på samma sätt som ett vektorlager gör men genom att se om cellerna i ett raster är innuti eller på kanten av en polygon kan man avgöra om en cell är en del av en byggnad eller inte. För att korrekt koppla punktdata i rasterformat till vektordata måste hänsyn tas till både vektorlagret och rasterdatans egenskaper. Figur 7.7 visar till vänster ett teoretiskt raster (med mycket få mätvärden) över en ytmodell som innehåller höjder i ett område vilket tydligt visar höjdskillnaden som en byggnad representerar. I mitten av figuren visas vektordata över samma byggnad och till höger visas vektordatan överlagrad rastret.

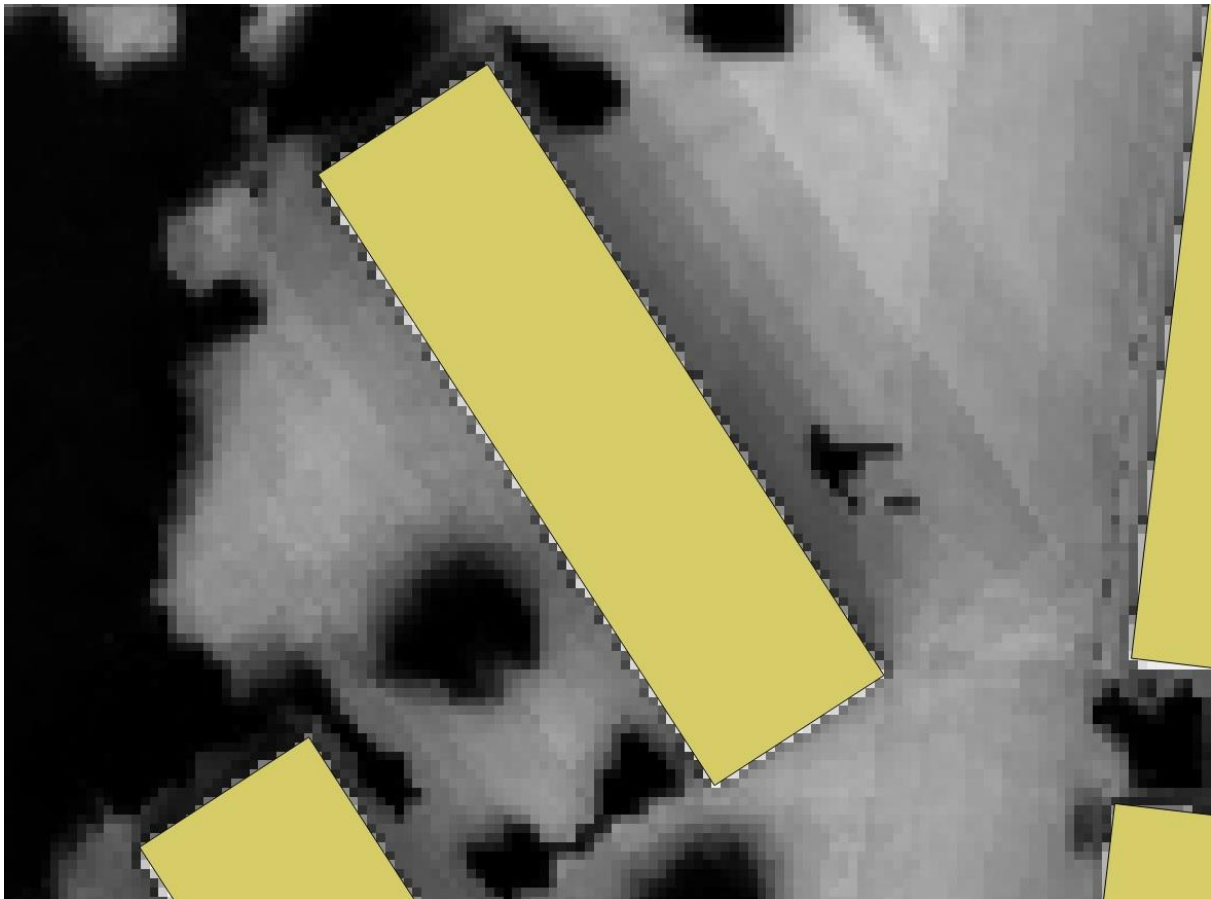


Figur 7.7, Illustration av skillnader i visualiseringen mellan raster- och vektordata.

Utsträckning

Exakt vad vektordatan som finns tillgänglig för projektet representerar är svårt att säga utan tillgång till information om vart mätningarna som bygger upp polygonerna är utförda. Det kan vara mätningar som gjorts under byggnationen och som representerar innerväggarna eller mätningar gjorda på den yttre fasaden. Eftersom ytterväggar på ett hus är tjocka ger dessa två olika mätningar väldigt olika stora polygoner. Väggarna i utdatan från SEBE är approximerade till att ligga vid takens yttersta spets (se avsnitt 3.2.1) vilket skapar väggar längre ut än i verkligheten eftersom tak ofta sticker ut från väggen och har stuprännor med mera. Denna kombination kommer att leda till att punktdata i de flesta fall kommer att ligga utanför vektordata vid jämförelser.

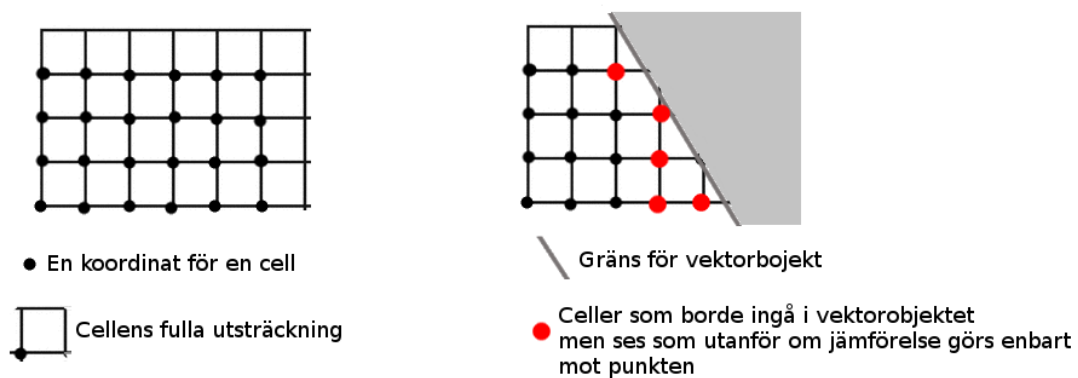
Figur 7.8 visar vektorlagret ovanpå ett rasterlager som visar solinstrålningen i området vilket också representerar takens utsträckning väl. Från figuren är det tydligt att takspetsarna ligger utanför vektorlagret i rastrets representation.



Figur 7.8, Vektorlager ovanför ett rasterlager som visar solintråning.

Uppbyggnad

Egenskaperna hos ett raster där ett värde representerar en hel cell bidrar till skillnaderna mellan rasterdata och vektordata. Detta innebär att det inte räcker med att göra en jämförelse utifrån en punkt och en polygon. Hela cellen måste jämföras mot polygonen annars finns risken att punkter exkluderas. Se Figur 7.9 för en grafisk representation av problemet.



Figur 7.9, Jämförelse av celler mot polygoner.

7.3 Grafiskt användargränssnitt

Det grafiska användargränssnittet är användarens metod för att kommunicera med visualiseringsapplikationen. Genom användargränssnittet kan användaren köra applikationens funktioner och få resultatet presenterat för sig. Det grafiska användargränssnittet skapas i QGIS genom applikationsbiblioteket Qt (se avsnitt 6.2). De användargränssnittskomponenter som tagits fram för visualiseringsapplikationen är framtagna genom Qt-designer (se avsnitt 6.2.5) eller skriva i python genom Qt:s python bindingar i PyQt.

När ett insticksprogram initieras i QGIS skapas menyobjekt som används för att nå insticksprogrammets funktioner. Eftersom prototypen innehåller flera olika funktioner och verktyg skapas en egen menyrad där programmets alla funktioner är samlade. Resultaten, dvs. statistik och 3D-modell, presenteras för användaren i form av dialogrutor. Figur 7.10 visar SEBEvisualizers menyrad.



Figur 7.10, SEBEvisualizers menyrad.

7.4 Geografisk miljö

För att visualiseringsapplikationen ska fungera krävs det att en geografisk miljö skapas. Denna miljö tillåter att data för solinstrålning kopplas till objekt såsom byggnader, det är också genom den geografiska miljön som en användare kan navigera för att hitta intressanta platser eller byggnader. Den geografiska miljön inkluderar ett vektorlager med byggnader för området och en bakgrundskarta för att göra navigation enklare.

7.4.1 Geografiskt lager

Det geografiska lagret kan paketeras med i insticksprogrammet i form av en shape-fil som sedan automatiskt läggs in i QGIS när en användare väljer att initiera applikationen. Detta fungerar för prototypen där områdets utsträckning är begränsad men hänsyn måste tas till att applikationen i framtiden ska kunna hantera större ytor.

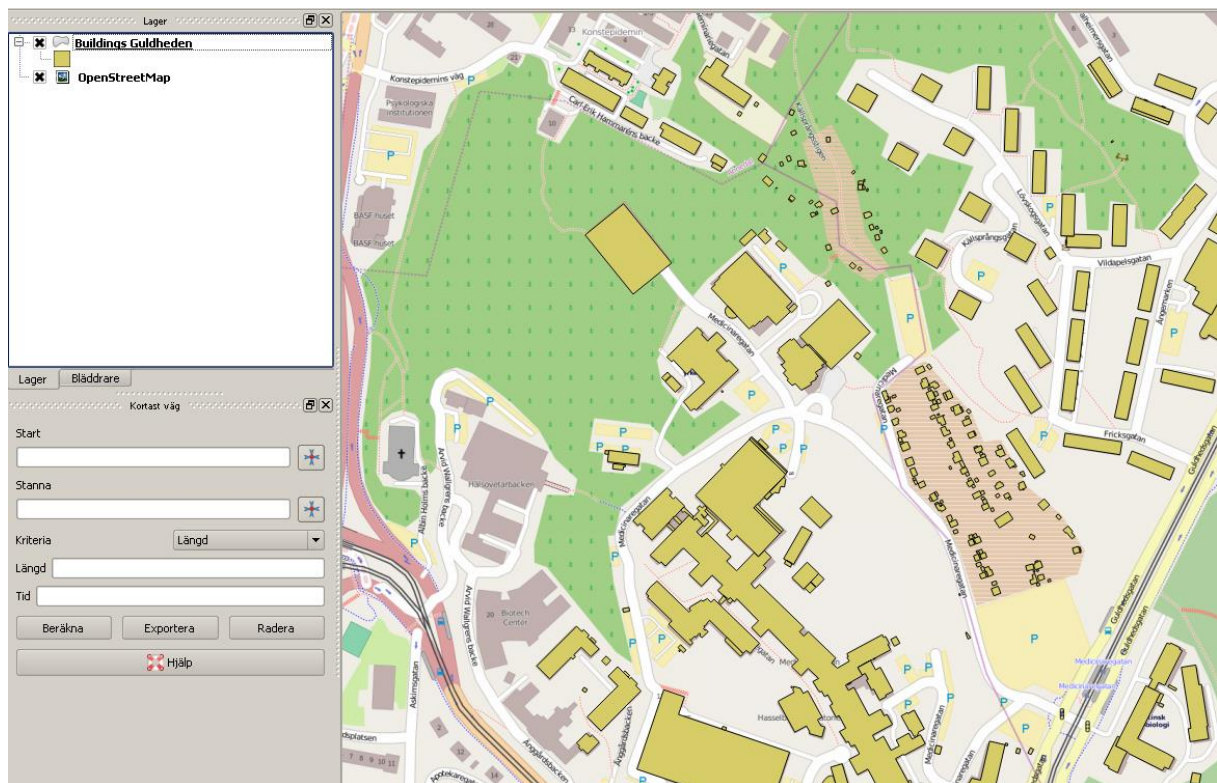
Två huvudsakliga möjligheter finns för att flytta det geografiska lagret från en lokal miljö till en server miljö. En möjlighet är att skapa en spatial databas som har möjlighet att lagra polygonlager. QGIS ger möjligheter till kopplingar för flera olika databaser som PostGIS, Spatialite och MSSQL-spatial.

QGIS ger också möjlighet att hämta geografiska objekt från OpenStreetMap där ett val kan göras att bara importera de polygoner som är markerade som byggnader (OpenStreetMap Wiki, 2014). Objekten i OpenStreetMap är skapade av användare vilket betydligt ökar risken för att objekt inte är markerade som byggnader trots att de är det eller att utsträckningen inte stämmer överens med verkligheten (OpenStreetMap, 2014a).

7.4.2 Bakgrundskarta

Enligt krav 8 (se avsnitt 2.2) skall en bakgrundskarta finnas för att göra navigationen enklare för användare. Det finns flera webbaserade-kartor som kan fungera väl som bakgrundskarta men QGIS har inga direkta möjligheter att nå dem. För att kunna hämta webbaserade bakgrundskartor från till exempel OpenStreetMap i QGIS finns ett användarutvecklat OpenLayers-insticksprogram.

När applikationen initieras görs ett försök att ladda in en bakgrundskarta genom OpenLayers-insticksprogrammet. Om OpenLayers-insticksprogrammet saknas uppmanas användaren att hämta detta program genom QGIS för att få tillgång till bakgrundskarta. Se Figur 7.11 för illustration av hur den geografiska miljön ser ut i QGIS.



Figur 7.11, Applikationens geografiska miljö i QGIS.

7.5 Markeringsverktyg

För att uppfylla kraven på applikationen krävs en mängd olika markeringsverktyg. Enligt krav 9 (se avsnitt 2.2) måste en användare ges möjligheten att ta fram statistik för ett eller flera objekt i ett område samt ett område i sig. Detta kräver att användare kan göra motsvarande markeringar.

QGIS har redan utvecklade verktyg för markering av objekt i ett vektorlager. För att göra det tydligt att de är kopplade till användningen av applikationen har menyobjekt för dessa även lagts till i applikationens menyrad (Figur 7.10).

För att användare ska kunna ta fram statistik över ett område måste nya verktyg tas fram eftersom det vektorlager som används av applikationen enbart innehåller byggnader. Verktygen som har tagits

fram skapar genom punkter valda av användaren temporära polygoner mot vilka statistiska jämförelser sedan kan göras.

7.6 Statistik

I ett tidigt stadium i utvecklingen bestämdes att prototypen främst skulle fokusera på att få fram fungerande statistik och 3D-modellering av byggnader. Intresset för solinstrålning på byggnader är betydligt större än intresset för omkringliggande områden. Särskilda markeringsverktyg för att tillåta statistik över ett område finns alltså utvecklade men är ej implementerade för framtagning av statistik och 3D-modell.

7.6.1 Punktdata

Underlag för solinstrålningsstatistik finns i tidigare presenterade utdata från SEBE (avsnitt 7.2).

Hantering

Enbart det begränsade område som används i fallstudien innehåller en stor mängd solinstrålningsdata. Bara textfilen för plana ytor innehåller fler än 2 miljoner punkter och filens storlek uppgår till runt 100 megabyte. Det är uppenbart att en applikation inte kommer att fungera om stöd inte finns för att kunna kommunicera med en databas som istället innehåller punktdatan.

Pythons utvecklingsbiblioteket innehåller en modul som stödjer SQLite-biblioteket vilket tillåter skapandet och hanteringen av en databas (Python Software Foundation, 2014). SQLite uppfyller de behov som finns på en databas för prototypens ändamål och används eftersom importen av biblioteket görs väldigt enkelt.

Den databas som skapats innehåller fyra tabeller utan inbördes relationer:

Roofs(X, Y, Z, energi, areakonstant) – Punkter på tak.

Skapades genom att köra textfilen för markytor genom ett script som översatte koordinaterna till referenssystem SWEREF99 12 00 samt gjorde jämförelser mot hela vektorlagret. Om en polygon i vektorlagret innehåller en punkt sparas punkten i tabellen.

Tabellen innehåller attributen X för koordinat i östlig riktning, Y för koordinat i Nordlig riktning, Z för koordinat i höjdlid, energi för solinstrålningen i ytan som punkten representerar och en areakonstant som står för den reella storleken på ytan.

Surface(X, Y, Z, energi, areakonstant) - Punkter på markytor. De punkter som inte träffar några objekt i vektorlagret. Innehåller samma attribut som Roofs-tabellen.

Walls(X, Y, Z, energi) - Punkter på väggar. Översatta koordinater till referenssystemet SWEREF99 12 00 från textfilen innehållande väggpunkter.

Tabellen innehåller attributen X för koordinat i östlig riktning, Y för koordinat i Nordlig riktning, Z för koordinat i höjdlid och energi för solinstrålningen i ytan.

Vegetation(X, Y, Z) - Punkter för vegetation. Översatta koordinater till referenssystemet SWEREF99 12 00 från textfilen innehållande punkter där det finns vegetation.

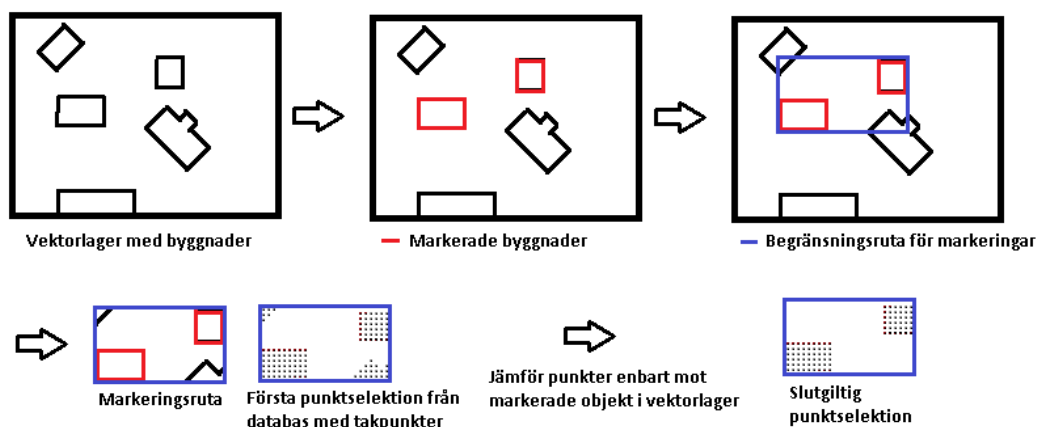
Innehåller bara attribut för koordinater då datan bara är till visuella ändamål.

För prototypens ändamål räcker det att se till att interaktion mot databasen fungerar, databasen körs därför för tillfället lokalt från en användares skrivbordsmiljö. Vid utveckling av en komplett applikation hade databasen flyttats ut till en servermiljö.

7.6.2 Sammankoppling av punktdata till vektorobjekt

Enligt krav 11-14 (se avsnitt 2.2) ska applikationen kunna framställa flera olika typer av statistik för olika typer av objekt. Som tidigare fastställts ligger fokus för prototypen på att lösa interaktionen mellan byggnadsobjekt i vektorlagret och punktdata i databasen.

Utifrån den eller de polygoner som markeras av en användare på vektorlagret skapas en begränsningsruta, det vill säga den minsta rektangel som kan innehålla alla de markerade objekten (Caldwell, 2005). Alla punkter kan sedan jämföras mot begränsningsrutans högsta och lägsta koordinat i nordlig riktning samt högsta och lägsta koordinat i östlig riktning vilket är en snabb selektion i en databas. De punkter som passerar selektionen jämförs sedan per punkt mot alla markerade vektorobjekt. Om en punkt överlappar med ett vektorobjekt börjar jämförelsen med nästa punkt i samlingen och om punkten inte överlappar med något vektorobjekt tas den bort från samlingen. Detta innebär att samlingen bara innehåller punkter som är en del av en byggnad efter iteration över alla punkter. Se figur 7.12 för illustration av arbetsflödet.



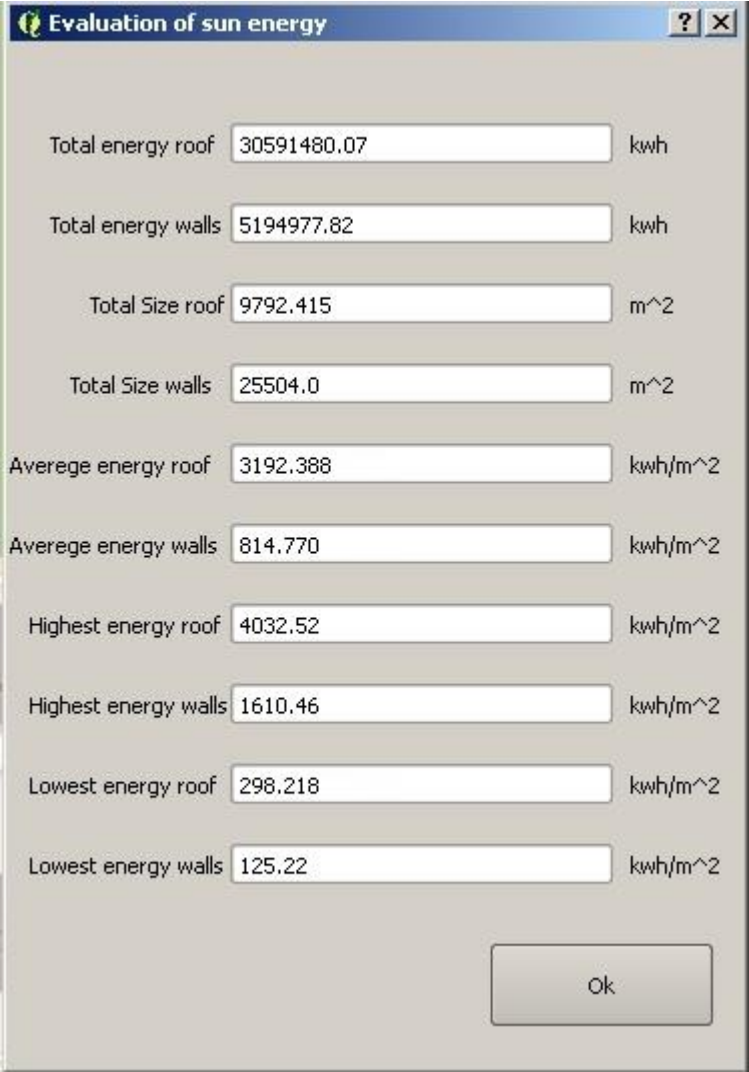
Figur 7.12, Arbetsflöde för punktselektion från databas.

För att få korrekta träffar när en punkt jämförs mot ett vektorobjekt krävs det att man tar hänsyn till att varje punkt representerar en större yta. (se avsnitt 7.2) Vid varje jämförelse skapas utifrån punkten en kvadrat för att representera upplösningen på det raster som ligger till grund för punktdata. Denna kvadrat används sedan för att se om överlapp finns gentemot vektorobjekt.

Jämförelser görs genom två metoder som körs varje gång en användare ber om statistik för markerade objekt. Jämförelserna resulterar i samlingar som sparas som temporära tabeller i databasen.

7.6.3 Sammanställning av statistik

Utifrån de punktsamlingar som tas fram kan databasoperationer utföras för att snabbt sammanfatta statistik för alla punkter i samlingarna. Den statistik som tas fram genom olika databasoperationer sammanställs och presenteras sedan till användaren i ett användargränssnittsobjekt utvecklat i Qt-designer (avsnitt 6.2.5). Se Figur 7.13 för den dialogruta som presenterar statistik för en användare.



The dialog box, titled "Evaluation of sun energy", displays the following statistics:

Category	Value	Unit
Total energy roof	30591480.07	kwh
Total energy walls	5194977.82	kwh
Total Size roof	9792.415	m ²
Total Size walls	25504.0	m ²
Average energy roof	3192.388	kwh/m ²
Average energy walls	814.770	kwh/m ²
Highest energy roof	4032.52	kwh/m ²
Highest energy walls	1610.46	kwh/m ²
Lowest energy roof	298.218	kwh/m ²
Lowest energy walls	125.22	kwh/m ²

An "Ok" button is located at the bottom right of the dialog.

Figur 7.13, Dialog som presenterar statistiken för en användare.

7.7 3D-visualisering

Krav 15 i kravspecifikationen specificerar att applikationen ska kunna visualisera data från SEBE i 3D. Krav 14 som gäller statistik kan också till viss del uppfyllas genom implementation av en 3D-modell.

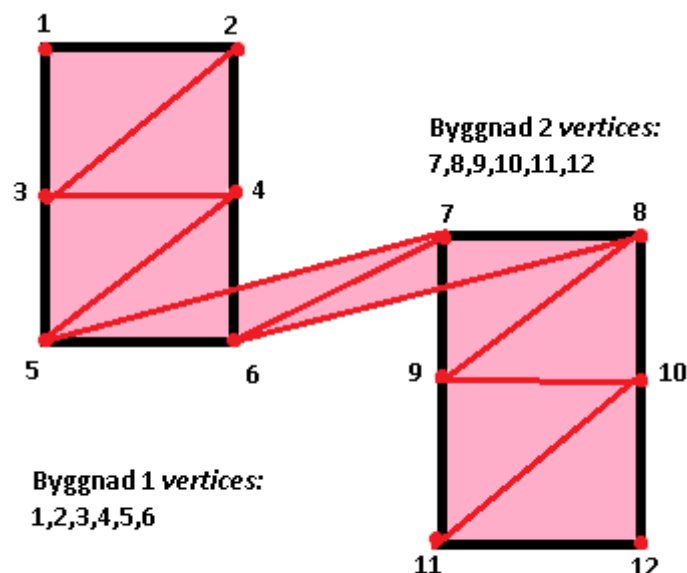
För att kunna skapa en 3D-modell krävs ett bibliotek som stödjer grafisk programmering. QGIS innehåller vid installation PyOpenGL-modulen. PyOpenGL modulen innehåller python-kopplingar och gör det möjligt att skapa OpenGL-modeller i Python (SourceForge, 2014b).

För att kunna rendera en 3D-modell krävs en kontext som talar om i vilken yta av skärmen som modellen ska visas. I de flesta fall används GLUT som den kontext där OpenGL-modeller visualiseras. Applikationsramverket Qt som QGIS är baserat på (se avsnitt 6.2.1) stöd dock öppnandet av nya fönster som inte är en del av ramverket. Detta leder till att en annan kontext för OpenGL krävs för 3D-modellen. Qt har istället för GLUT en egen kontext för OpenGL kallad en QGLWidget där modellen kan visualiseras och som används av applikationen (Qt Project, 2013).

7.7.1 Skapande av byggnader

Som diskuterats i avsnitt 5.1 krävs det för att skapa ett objekt i OpenGL att punkter kallade *vertices* binds ihop för att på så sätt skapa ytor som kan färgläggas. Två huvudsakliga metoder för att visualisera byggnaderna och datan för solinstrålning i 3D har identifierats. Bägge metoderna utgår ifrån punktsamlingar i temporära tabeller i databasen som tas fram på samma sätt som för statistiken (se avsnitt 7.6).

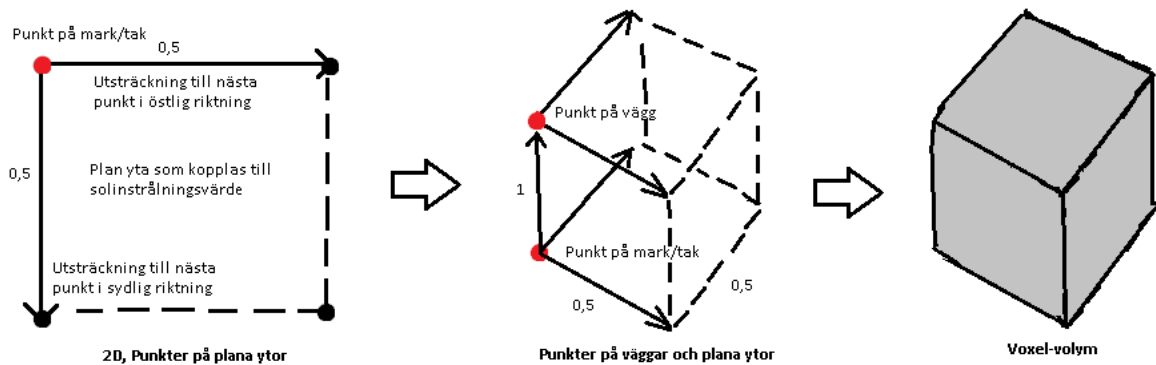
En metod är att först sortera punktsamlingarna på ett sätt som korrekt målar upp byggnaderna genom någon av uppritningssmetoderna i OpenGL (se avsnitt 5.1). Byggnader är komplexa objekt uppbyggda på många olika vis och att skapa en sorteringsalgoritm som fungerar effektivt och får fram korrekta objekt varje gång är svårt. Denna metod kräver också att det går att skilja på till exempel olika tak. Annars kommer ytor mellan två olika tak också hanteras som om de är sammanfogade och modellen kommer bli oanvändbar. Figur 7.14 illustrerar hur problemet skulle kunna visa sig mellan två tak på två byggnader och uppritningssättet *GL_Triangle_strip*.



Figur 7.14, Problematik vid uppritning utifrån *vertices*-samlingar.

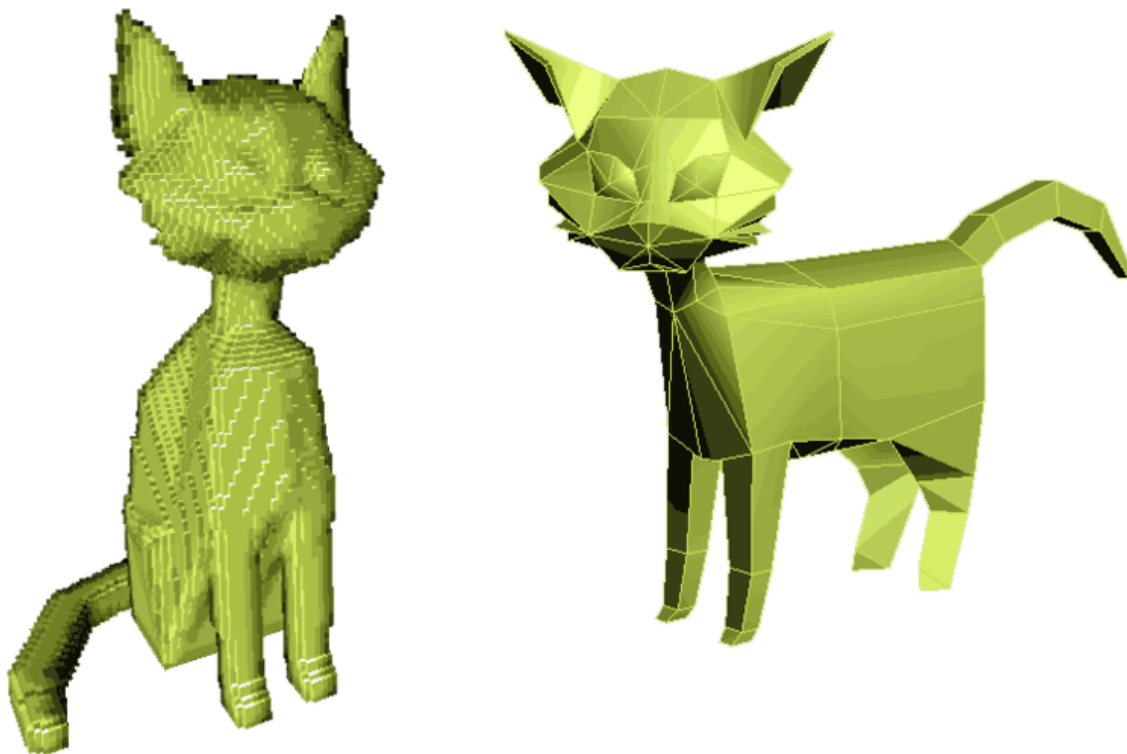
Den andra metoden är att likt GRASS (jfr avsnitt 6.2.4) använda sig av *voxlar* för att måla upp objekt. Varje punkt kommer då istället för att användas som en *vertex* till en byggnad användas till placeringen av en *voxel*. Denna metod har fördelarna att den är relativt enkel att implementera och

att varje byggnad är uppbyggd av självständiga objekt gör att visualiseringen blir korrekt så länge indata stämmer. Nackdelar med metoden är att den kräver mer prestanda och att datasetets upplösning syns i modellen genom att ytor som inte är vinkelräta kommer att framställas som kantiga. Minskningen i prestanda kommer sig av att varje punkt istället för en *vertex* kräver fyra för varje sida av *voxeln* som ska visas. Figur 7.15 visar hur en voxelvolym kan bestämmas utifrån punktmätningarnas täthet. I SEBEvisualizers fall gäller det att det är 0.5 enheter mellan punkter i plant led och 1 enhet mellan punkter i höjled (se avsnitt 7.2.2).



Figur 7.15, Bestämmandet av en voxels volym.

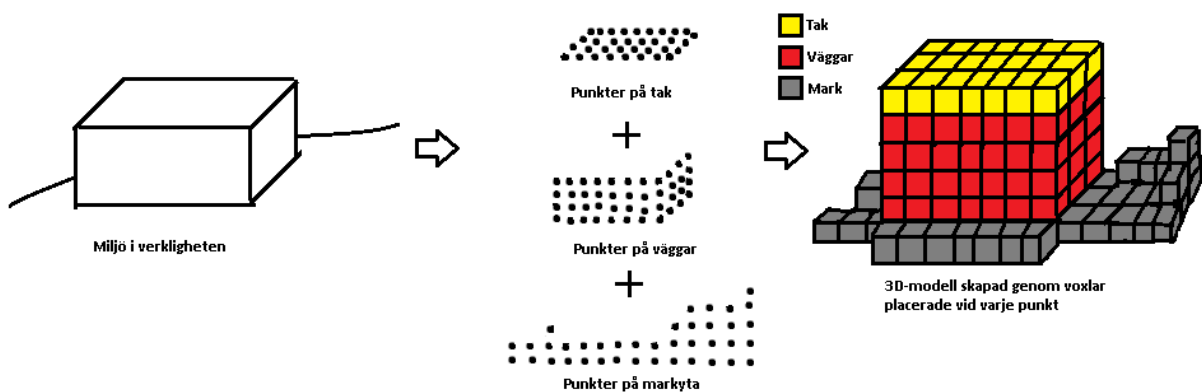
Figur 7.16 visar ett exempel på skillnaden på att skapa en modell utifrån voxlar och en modell utifrån polygonuppriktning. Från figuren är det tydligt att modellen till vänster använder sig av ett större antal objekt. Det är också tydligt att modellen till höger skulle kunna visualiseras helt inkorrekt om inläsningen görs i fel ordning.



Figur 7.16, Katt skapad genom voxlar till vänster och genom polygonuppriktning till höger (Wikimedia Commons, 2014).

Varje *vertex* i en modell kräver ytterligare beräkningar, det är därför tydligt att *voxel*-metoden är långt från optimal vad gäller prestanda. Trots detta är det denna metod som valts för applikationen främst på grund av att risken för felaktig visualisering är mycket lägre. Att direkt i modellen kunna se datasetets upplösning kan också vara en fördel då det ger en användare en bild av hur många mätningar som gjorts och därmed också säkerheten av resultatet.

3D-modellen byggs upp genom att *voxlar* placeras ut vid varje punkt från databasen som är en del av den av användaren valda ytan (se avsnitt 7.6.2). Varje *voxel* blir som ett byggnadsblock som tillsammans med andra block skapar en helhetsbild över hela ytan. Se Figur 7.17 för en illustration av hur en 3D-modell skapas utifrån punkter hämtade från de olika tabellerna i databasen.



Figur 7.17, Jämförelse mellan miljö i verkligheten, hämtade punkter från databasen och den 3D-modell som sedan skapas utifrån dessa punkter genom voxlar.

7.7.2 Färgläggning

Färgläggningen av modellen sker utifrån värdena av solinstrålning för varje punkt. Användare ges möjlighet att välja mellan två typer av färgsättning:

- En dynamisk färgsättning som utifrån lägsta och högsta solinstrålningsvärde i modellen interpoleras över en färgskala.
- En grupperad färgsättning där varje punkt placeras i en grupp beroende på dess solinstrålning och ges en färg baserat på gruppen. Användare ges möjlighet att ändra de värden som definierar en grupp.

Färgskalorna är baserade på tidigare färgsättningar publicerade i artiklar kopplade till SEBE och utgår ifrån färgsättning av så kallade *heat maps* (Wikipedia, 2014b). Färgskalan går från rött för höga värden till grönt för medelvärden och blått för låga värden. Figur 7.18 visar färgskalan som har använts vid tidigare arbeten med SEBE.



Figur 7.18, Färgskala använd för visualisering i artikel om SEBE (Lindberg, 2014).

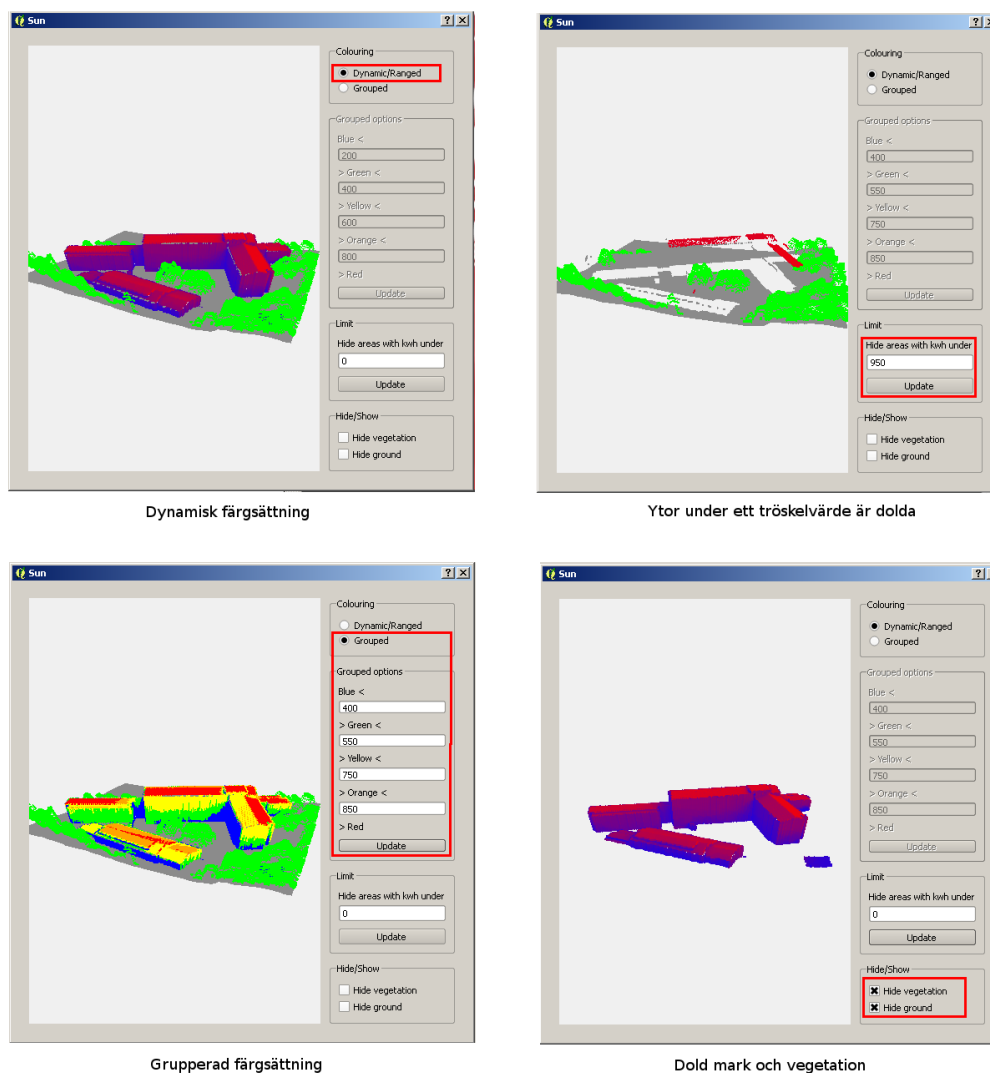
7.7.3 Användargränssnitt

Målet med 3D-modellen är inte visualiseringen i sig utan att utöka användbarheten av utdatan från SEBE. Den möjlighet visualiseringen ger en användare är bland annat att i detalj undersöka solinstrålningen på byggnader och att hitta ytor som lämpar sig för till exempel placering av solceller. Genom att tillåta att användaren genom ett användargränssnitt kan förändra modellen på olika sätt ger detta möjligheten att utreda situationer som är svåra att presentera genom statistik.

3D-modellens användargränssnitt tillåter att en användare modifierar modellen på olika sätt:

- Använda två olika typer av färgsättning (se avsnitt 6.5.2).
- Dölja alla ytor på väggar och tak vars solinstrålning är under ett bestämt värde.
- Visa och dölja omkringliggande vegetation och mark.

Se figur 7.19 för resultatet av visualisering i 3D för ett område med användargränssnitt.



Figur 7.19, 3D-modell med användargränssnittets olika möjligheter.

7.8 Licenser

Att använda sig av verktyg med öppen källkod innebär att man får tillgång till ett verktyg helt gratis men det innebär också att om man väljer att skapa en ny applikation baserat på ett av dessa verktyg kan det krävas att man följer vissa restriktioner som beror på verktygets licens. Vissa licenser är till exempel helt öppna och tillåter att en användare skapar en kommersiell applikation från ett verktyg med öppen källkod utan restriktioner men många kräver att allt som skapas fortsätter att vara öppen källkod.

Detta arbete fokuserar på att använda sig av verktyg med öppen källkod och det är därför viktigt att undersöka eventuella licenskrav som kan komma från de olika verktygen.

QGIS-licens

QGIS är distribuerat under licensen GNU General Public license. Denna licenstyp ger användare friheten att fritt använda, undersöka, anpassa och distribuera program baserade på denna licens. Licensen medföljer alla vidareutvecklade applikationer och all distribution måste vara gratis och innehålla applikationernas källkod (Free Software Foundation, 2007).

OpenGL-licens

Att utveckla mjukvara eller applikationer med OpenGL kräver ingen licens (Silicon Graphics Inc, 2014). PyOpenGL är distribuerad under en BSD-licens (SourceForge, 2014b) vilket tillåter att man använder grundmaterialet på de flesta tänkbara sätt. BSD-licenser tillåter att man tar material som tidigare varit i öppen källkod och gör den ofri, det vill säga exkluderar källkoden från distributionen (OpenSourceOrg, 2014).

OpenLayers-licens

OpenLayers är likt PyOpenGL distribuerat under en BSD-licens (OpenLayers, 2014).

OpenLayersPlugin-licens

OpenLayers-insticksprogrammet till QGIS är distribuerat under en GNU General Public License-licens (Kalberer & Walker, 2014).

OpenStreetMap-licens

OpenStreetMap är öppen data. Användare är fria att kopiera, distribuera, överföra och anpassa datan från OpenStreetMap. Krav finns på att OpenStreetMap och dess bidragsgivare anges som källa (OpenStreetMap, 2014b).

SEBEvisualizers licens

Eftersom GNU GPL licensen, som QGIS använder sig av, kvarstår för applikationer baserade på programvara under licensen innebär detta att SEBEvisualizer kommer att bli distribuerat under denna licens så länge QGIS används som utvecklingsplattform.

8. Resultat

Detta avsnitt presenterar det slutgiltiga resultatet av utvecklingen av applikationen. Detta görs främst genom en avstämning mot kravspecifikationen. För en mer detaljerad genomgång av hur visualiseringsapplikationen används se användningsmanualen (se Appendix A).

8.1 Avstämning mot kravspecifikation

Detta avsnitt redovisar om kraven i kravspecifikationen är helt, delvis eller inte ännu uppfyllda. För de krav som är delvis eller icke uppfyllda förs en diskussion i kapitel 9.

Nödvändiga krav:

1. Läs in och visa vektorlager. – **Uppfyllt**
2. Modifiera vektorlager. – **Uppfyllt**
3. Läs in och visa rasterlager. – **Uppfyllt**
4. Utföra beräkningar på rasterlager. – **Uppfyllt**
5. Panorera över kartområdet. – **Uppfyllt**
6. Visualisera kartområdet i olika skalor. – **Uppfyllt**
7. En legend över visualiserade lager. – **Uppfyllt**

Krav 1-7 uppfylls av QGIS. (se avsnitt 6.1) Flera av kraven har dock visat sig enbart vara användbara vid utvecklingen av applikationen eller har inte använts och därför inte en del av själva applikationen.

8. En bakgrundskarta över området för enkel navigering. – **Uppfyllt**

Krav 8 uppfylls genom OpenLayer-insticksprogrammet (se avsnitt 7.4.3).

9. Verktyg för att markera objekt eller område på kartan. – **Delvis uppfyllt**

Markering av objekt uppfylls av QGIS egna markeringsverktyg. Verktyg för markering av ett område har utvecklats men inte implementerats.

10. Ta fram modelldata från markerat objekt eller område. – **Uppfyllt**

Görs genom interaktion mellan databas och vektorlager (se avsnitt 7.6.2).

11. Beräkna statistik utifrån modelldata på objekt eller flera objekt i området. – **Uppfyllt**

Görs genom databasoperationer och kopplas till Qt-dialog (se avsnitt 7.6.3).

12. Skilja på solinstrålning på olika höjder, till exempel tak och mark – **Uppfyllt**

Görs genom interaktion mellan databas och vektorlager (se avsnitt 7.6.2).

13. Skilja på solinstrålning för ytor med olika solvinklar, till exempel fasad och tak. – **Uppfyllt**

Görs genom 3D-modellen (se avsnitt 7.7).

14. Visa statistik för olika tidsperioder. – **delvis uppfyllt**

Statistik för kvartal saknas.

15. Visualisera objekt och område, exempelvis i 3D. – **Uppfyllt**

Görs genom PyOpenGL och QGLWidget (se avsnitt 7.7).

Önskvärda krav:

16. Webbpublicering av huvudkraven. – **Ej uppfyllt**

Se kapitel 9.

17. Inkludera beräkningar för modelldata i applikationen för att tillåta ändringar av in- och utdata. – **Ej uppfyllt**

Se kapitel 9.

9. Diskussion

Detta avsnitt diskuterar applikationens nuläge och framtida utveckling.

9.1 Diskussion av prototyp

Som visualiseringsapplikationen står idag är alla som klassats som nödvändiga uppfyllda eller delvis uppfyllda. Inget av de önskvärda kraven har uppfyllts men tänkbara lösningar kommer att diskuteras i detta avsnitt. Det finns också vissa osäkerheter kring resultaten. I nuläget är de svåra att göra något åt utan indataförändringar men med annan indata hade de kunnat åtgärdas.

9.1.1 Återstående krav

Nödvändiga krav:

Krav 9: Möjlighet att markera objekt eller område på kartan.

Detta krav är inte helt uppfyllt på grund av att behovet att markera områden och objekt utöver byggnader bedömdes vara relativt litet. Det finns dock markeringsverktyg speciellt utvecklade för att kunna stödja möjligheten i framtiden. Detta skulle i så fall fungera genom att markeringsverktygen skapar temporära vektorobjekt som sedan skulle användas för att sammankopplas till data i databasen på samma sätt som för byggnader (se avsnitt 7.6.2). Det största arbetet som skulle behöva utföras om man i framtiden ska införa funktionen är att ge applikationen möjlighet att städa upp de temporära vektorobjekten efter det att de blivit använda. Statistiken som presenteras bortser för närvarande också från all punktdata som inte är en del av ett tak eller en vägg. Detta hade också behövt modifieras.

Krav 14: Visa statistik för olika tidsperioder.

Eftersom databasen lagras lokalt måste utrymmet som den upptar begränsas. Att lägga in ytterligare information om instrålning för varje kvartal hade medfört en större och mer svårhanterlig databas. Att uppdatera all data i databasen är också väldigt tidskrävande. Det är främst av dessa anledningar som statistiken för kvartal inte valdes att inkluderas i databasen. Kravet är därför inte helt uppfyllt.

Önskvärda krav:

På grund av tidsbrist hann inte utveckling påbörjas av något av de önskvärda kraven. Några tänkbara teoretiska lösningar kommer istället att presenteras här.

Krav 16: Webbpublicering av huvudkraven.

För webbpublicering av en applikation som uppfyller alla huvudkraven hade det varit lämpligare att skapa en webbapplikation istället för ett insticksprogram (se avsnitt 9.3). Insticksprogrammet hade dock kunnat gynnas av en viss del webbpublicering. WebGL hade kunnat vara ett sätt att tillåta att användare exporterar resultatet från en 3D-modell skapad i insticksprogrammet. Detta hade kunnat göras genom att skapa en stomme för en WebGL-applikation som utför ungefär samma arbete som 3D-modellen. Istället för att hämta data för 3D-modellens uppbyggnad från databasen skulle denna WebGL-applikation använda sig av en variabel i form av en lista som skulle innehålla all data för de

punkter som vore relevanta för en generad modell. Om användaren väljer att exportera en 3D-modell skapas variabeln utifrån det nuvarande databasurvalet och skickas till WebGL-applikationen. Modellen exporteras som en html-fil som sedan hade gått att öppna lokalt eller publiceras av användaren utan någon ytterligare koppling till insticksprogrammet.

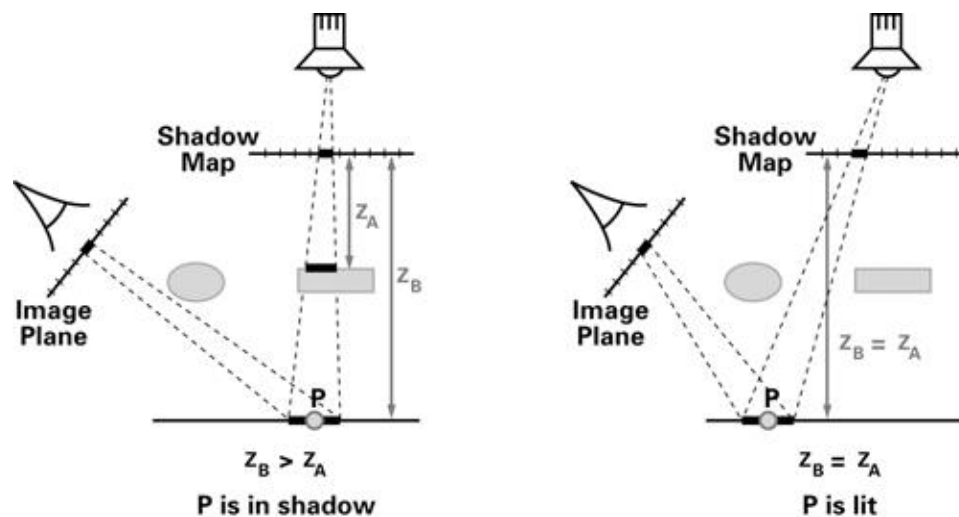
Krav 17: Inkludera beräkningar för modelldata i applikationen för att tillåta ändringar av in- och utdata.

Det är genom shaders möjligt att utföra många olika typer av beräkningar direkt i en 3D-modell (se avsnitt 5.1.3). Varje beräkning hade ökat tiden det tar att skapa modellen och eventuellt försämrat prestandan. Om det i nuläget är möjligt att flytta över delar av eller alla SEBE:s beräkningar till SEBEvisualizer och fortfarande ha en användbar 3D-modell är svårt att säga men det är värt att diskutera. Hade beräkningarna utförts direkt i SEBEvisualizer hade det till exempel varit möjligt att beräkna solpotential för en tänkt byggnad och effekterna av skuggningen från denna byggnad på omkringliggande objekt. Detta hade varit intressant för användare som sysslar med olika typer av byggnation. För att beräkningar ska vara möjliga i SEBEvisualizer måste applikationen ha tillgång till meteorologiska data. Tillgång till positionsdata från ytmodeller över landskap, byggnader och vegetation är också ett krav men finns redan genom databasen.

Det första steget mot en lösning är att tillåta att nya byggnader skapas av användare. Det största problemet med att skapa en sådan lösning är att byggnaden måste delas upp i punkter som följer samma indelning som resten av modellens indata för att sedan användas för att placera ut *voxlar*. En stor anledning till att SEBEvisualizer utvecklats är att det är svårt att tolka denna form av punktdata direkt ifrån text. Av samma anledningar är det svårt att förvänta sig att en användare klarar att skapa en tänkt byggnad i någon typ av textform. En metod för att göra det möjligt att skapa byggnader är att ta fram en funktion som tillåter att användare ritar upp enkla objekt såsom kuber i 3D för att sedan utifrån dessa objekt automatiskt ta fram de punktdata som krävs. En annan metod är att skapa någon typ av inläsningsstöd för någon av de filtyper som ofta används för att representera ett objekt inom 3D-grafik. Kan sådana filtyper läsas in kan en användare först ta fram en byggnad i ett verktyg för 3D-modellering för att sedan ladda in byggnaden i SEBEvisualizer som automatiskt kan generera lämpliga punkter för objektet. 3D-objekt byggs vanligtvis upp av polygoner. En metod för att omvandla polygonbaserade 3D-modeller till *voxel*-baserade modeller har tagits fram av Dijkstra et al. (1997) vilket skulle kunna fungera som en utgångspunkt för en lösning. Fullt fungerande och användbara metoder för dessa typer av stöd hade med all sannolikhet tagit lång tid att utveckla men hade varit till stor nytta för användare.

Nästa steg hade varit att beräkna de skuggvolymerna som uppstår utifrån de objekt som finns i modellen. Positioner för objekten kan hämtas från ytmodeller över landskap, byggnader, vegetation. Skuggvolymerna kan sedan användas för att beräkna mängden solinstrålning efter gällande solinstrålningsförhållanden utifrån olika meteorologiska data som solinstrålningsmängder, temperaturer och luftfuktighet. Inom 3D-grafik är korrekt ljussättning en stor del av modelleringen och det finns flera olika metoder för att uppskatta skuggor. Eftersom det i detta fall gäller realtids-rendering måste lösningen som används fokusera på prestanda över felsäkerhet. Ett relativt snabbt sätt att uppskatta skuggor är att använda sig av en skuggkarta. I vanliga fall renderas en 3D-scen enbart ur användarens perspektiv. Ska skuggkartor skapas renderas även scenen utifrån perspektivet för de ljuskällor som finns i scenen. När det som i detta fall rör sig om en rendering utifrån solens

position används en ortografisk projektion på grund av det långa avståndet mellan solen och jorden. Från renderingen ur detta perspektiv lagras värden för djupet till närmsta synliga objekt i varje pixel. Dessa samlade värden är vad som kallas för en skuggkarta och de kan sedan användas för att beräkna vilka delar av scenen som ligger i skugga (Williams, 1978). Se Figur 9.1 för en illustration av hur en skuggkarta fungerar. Notera att perspektivet som används för ljuskällan i figuren inte är en ortografisk projektion.



Figur 9.1, Illustration av hur skuggkarta fungerar (Kilgard, 2003).

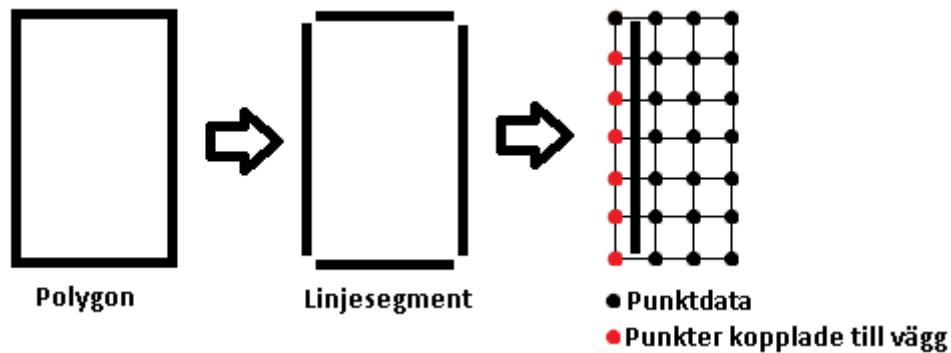
Solens position förändras ständigt under dygnet och säsongerna under året och nya skuggkartor måste tas fram vid varje position. Ska genomsnittlig solinstrålning beräknas över ett helt år kommer renderingen bli tidskrävande.

9.2 Framtida utveckling av SEBEvisualizer

Att ta SEBEvisualizer från prototyp till en fullt fungerande applikation, kräver utvecklingsinsatser som inte ryms inom ramarna för detta arbete. Det finns också andra aspekter att ta hänsyn till, bland annat versionsförändringar av QGIS och expansion av fallstudiens geografiska område.

Datasetet

Att det saknas koppling mellan vektordatan och solinstrålningsdatan från SEBE leder till sämre träffsäkerhet i resultaten och gör det svårare att till exempel koppla data korrekt till väggar (se avsnitt 7.2.5). Förändringar i hur SEBE utför sina beräkningar är vid det här stadiet svårt att tänka sig. Om istället vektordatan skapas med samma metoder som väggarna identifieras med i SEBE kommer utsträckningen stämma överens och algoritmer för att koppla data till separata väggar skulle kunna utvecklas. En potentiellt användbar metod hade kunnat vara att dela upp en byggnadspolygon i linjesegment. Varje linjesegment hade sedan kunnat jämföras mot solinstrålningsdata för att se ifall de överlappar. Figur 9.2 illustrerar metoden för linjesegment.



Figur 9.2, Metod för att koppla solinstrålningsdata till väggar.

Versionshantering

Att använda QGIS som ett ramverk ger många fördelar och förenklar många delar av utvecklingsarbetet. Att applikationen är ett insticksprogram innebär dock att förändringar i QGIS, i värsta fall leda kan till att applikationen slutar fungera utan uppdateringar. Detta gäller också för OpenLayer-insticksprogrammet som applikationen använder sig av för att lägga in bakgrundskartan. Sedan utvecklingsarbetet börjat har QGIS uppdaterats från version 2.2 till version 2.6. Uppdateringen innehåller ändringar i vissa metoanrop för både QGIS och OpenLayers-insticksprogrammet. Visualiseringsapplikationen är för närvarande optimerad till QGIS version 2.2 och OpenLayers-insticksprogram till motsvarande version. Äldre versioner av QGIS finns alltid att hämta från webben men valet att utveckla visualiseringsapplikationen som ett insticksprogram, betyder att det kommer att krävas kontinuerliga uppdateringar ifall man vill att applikationen ska fungera tillsammans med den senaste versionen av QGIS.

Databasen

Det finns mycket arbete som behöver göras med databasen innan SEBEvisualizer kan distribueras. Detta arbete har fokuserat på att se till att applikationen är utvecklad på ett sätt som är förenligt med en databas men den fungerar i nuläget bara i en lokal miljö. Om applikationen distribueras och databasen placeras i en serverstruktur krävs att databasen är säker och skyddar datan både från användarfel och intrångshot som SQL-injektion. De temporära tabeller som skapas för att bygga upp 3D-modellerna saknar också stöd för att två användare når dem samtidigt. Detta är nödvändigt att lösa om databasen ska förläggas till en servermiljö.

Arbetet som databasen utför är också det som för närvarande är mest tidskrävande när applikationen körs. Effektivisering i form av indexering av tabellerna och optimala SQL-kommandon för att hämta data hade både minskat söktiden för databasen och gjort applikationen effektivare. Valet av språket SQLite är motiverat främst av att det är en del av Python-biblioteket. En utredning av alternativa och mer effektiva databashanteringsbibliotek är värt tiden om applikationen ska tas från prototyp till fullständig applikation. Det finns också anledningar att överväga att utvidga databasen till en relationsdatabas. Om en relation skapas där ett tak är anslutet till flera väggar, hade det till exempel varit möjligt att dela upp statistik i fler kategorier. Bland annat separata uppgifter för varje vägg.

Databasen behöver underhåll i form av uppdaterad information med regelbundna intervall för att vara aktuell. I en stadsmiljö tillkommer och försvinner byggnader, vegetation och andra objekt som permanent skuggar områden vilket kraftigt förändrar solinstrålningspotentialen. För att underhålla databasen krävs att uppdaterade ytmodeller används av SEBE för att generera ny data. Denna nya punktinformation måste knytas till uppdaterade vektordata för att sedan läggas in i databasen. Automatiska script kan enkelt tas fram för att sköta stora delar av detta underhåll men arbetet är relativt tidskrävande, speciellt på stora ytor.

3D-modellens gränssnitt

Gränssnittet till 3D-modellen är viktigt för att låta användaren utforska solinstrålningen för en byggnad. Mer valmöjligheter och sätt att förändra modellen ger användaren större nytta av modellen. Ett nuvarande problem med den dynamiska färgsättningen är att de stora skillnaderna mellan solinstrålning på väggar och tak gör att få värden hamnar i mittspannet. Alla värden på väggar kommer att ligga relativt nära det lägsta värdet och alla värden på tak kommer att ligga nära det högsta värdet. En tredje användbar lösning, hade varit att ha en dynamisk färgsättning som gör beräkningarna separat för väggar och tak. Detta hade varit användbart för en användare som är intresserad av att hitta områdena med högsta solinstrålning både på väggar och tak. En annan användbar funktion hade varit att tillåta användaren att skapa egna färgsättningar och grupperingar.

Att kunna hämta ut statistik direkt ur 3D-modellen är något som hade ökat nyttan av modellen för flera användare. Det är möjligt att knyta attribut till *vertices* i modellen (se avsnitt 5.1.2). För tillfället används värden för solinstrålning enbart för att bestämma en färg på varje *voxel*. Det hade varit möjligt att i 3D-modellen även spara solinstrålningssinformation för varje *voxel*. Informationen hade kunnat användas av markeringsverktyg som hämtar ut värdena från markerade *voxlar* för att sedan till exempel skapa sammanfattningar. På detta sätt hade statistik för olika ytor kunnat redovisas direkt i 3D-modellen. En möjlig lösning när det gäller implementation av markeringsverktyg, är att använda så kallad *ray tracing* eller på svenska strålföljning. Denna metod används ofta inom 3D-grafik för att skapa fotorealistiska scener. Vid strålföljning spåras ljusstrålar i en scen baklänges tills de når ljuskällan. De spårade banorna kan sedan användas för att ta fram realistiska skuggor och reflektioner i scenen (Whitted, 1979).

En metod för att skapa markeringsverktyg genom strålföljning vore att skicka ut en stråle ifrån den pixel där användaren gör sin första markering. Den första *voxel* som strålen möter skulle sedan användas som ankarpunkt för markeringen. Ytterligare markeringar hade skapat en yta som statistik hade kunnat sammanfattas för. Eftersom modellen är i tre dimensioner måste metoden ha stöd för att hantera flera specialfall för till exempel perspektiv. Ett sätt att undvika problem med att markeringen inte följer en vägg, är att se till så att alla markerade *voxlar* är sammanhängande.

Expansion av statistik

Den statistik som presenteras för användaren är i nuläget enbart grundläggande. Det finns mycket information som kan vara användbar men som inte presenteras för användaren. Om databasen utvecklas till en relationsdatabas och om de skillnader som finns i utsträckningen mellan vektorlagret och rasterdatan för väggar rättas till (se avsnitt 7.2.5) skulle till exempel flera ytterligare algoritmer kunna utvecklas. De tak och väggar som hämtas in från ett användarvalt område hade då kunnat

hanteras separerat. Statistik hade kunnat presenteras för varje vägg och tak i området och inte bara sammanfattningar av solinstrålning på tak och väggar i området.

Ett vanligt behov är att undersöka den ekonomiska nyttan för installation av solpaneler på en byggnad. Det är då intressant att kunna jämföra solinstrålningspotential mot till exempel solpanelers kostnad och effekt per kvadratmeter samt den beräknade livslängden på en installation. Genom att skapa möjligheter för användaren att själv skriva in dessa uppgifter hade det kunnat utvecklas metoder för att automatiskt beräkna ungefärlig ekonomisk vinst eller förlust för en installation av en viss storlek. Det hade även varit möjligt att tillföra en funktion för att visa hur stor del av den valda ytan som är effektiv nog att genererar en viss vinst. Denna lösning hade haft fördelen att inte bli utdaterad med tiden. Tekniken för att tillvarata solenergi utvecklas och det är då säkrare att datan kommer från användarna själva. En ytterligare funktion som hämtar genomsnittliga kostnader och effektivitet för solenergisystem från databasen hade kunnat läggas in som komplement. Detta i syfte att underlätta för användare som inte har lätt tillgång till informationen. Denna lösning hade dock krävt att databasen underhålls och uppdateras kontinuerligt.

Sammankoppling av 3D-modell och statistik

Anledningen till att 3D-modellen och statistiken presenteras separat, är att korrekt statistik kräver ytterligare tidskrävande urval från databasen (se avsnitt 7.6.2). Separationen gör att 3D-modellen kan genereras snabbare. Det är möjligt att separat ta fram både statistik och 3D-modell från samma markering men det skulle kunna förenkla avläsningen om de bägge funktionerna slås ihop i ett sammanhängande fönster. Detta skulle antingen kunna vara ett kompletterande verktyg i menyraden eller vara det enda verktyg som används efter markering.

En sammanslagning tillsammans med en expansion av funktionerna för presentation av statistik och 3D-modell, hade gett ett verktyg där användare kan få ut både sammanfattad solinstrålningsinformation över ett större område och mer detaljerad information för mindre ytor samtidigt. Verktyget riskerar dock att bli svåröverskådligt och därför svårt att använda om inte integreringen kan lösas på ett bra sätt.

Optimal positionering för solcellssystem

Den solpotential som beräknas av SEBE och visualiseras av SEBEvisualizer är beräknad från varje ytas nuvarande lutning och riktning. Beräkningar tar inte hänsyn till att ett installerat solenergisystem kan placeras med en lutning som inte stämmer överens med takytan som det är installerat på. Ett optimalt svenskt solesystem är i de flesta fall orienterat rakt åt söder med en 30-50 graders lutning (Stridh, 2005). Tas ingen hänsyn till detta riskerar ytor med mindre än optimal lutning att representeras som en olämplig placering för ett solenergisystem i modellen när det i verkligheten går att installera systemet med en sådan lutning att ytan blir lämplig. Detta är relaterat till de beräkningar som görs i SEBE och inte det arbete som SEBEvisualizer i dagsläget utför. Påbörjas arbetet med att flytta beräkningsdelarna från SEBE till SEBEvisualizer kan förändringar övervägas.

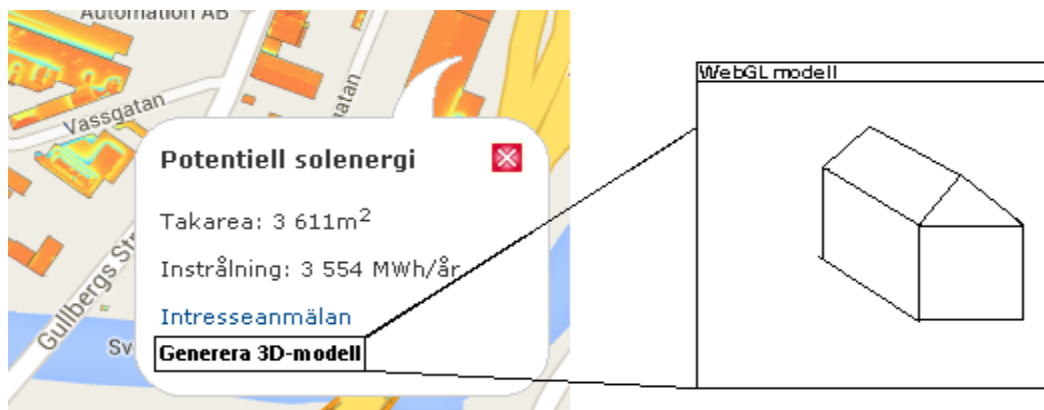
Utvidgning av studieområdet

Tanken med applikationen är att den ska kunna hantera mycket större områden än det som använts i fallstudien. Prototypen har utvecklats med detta i åtanke och inga värden som koordinater och dylikt är därför hårdkodade. Det enda som är förbestämt är *voxelvolymen*. Volymvärden måste ändras manuellt eller hämtas från indata om tillämpningen ska fungera korrekt. Att utvidga området borde inte vara svårare än att lägga in vektordata med tillhörande punktdata i insticksprogrammet respektive databasen. Att flytta vektordata från lokal lagring i insticksprogrammet till en spatial databas, borde dock övervägas.

9.3 Webblösning

Privata användare är en viktig målgrupp för SEBEvisualizer. Därför bör utvecklingen av en separat webbapplikation övervägas. Det är svårt att tänka sig att en oerfaren användare som enbart är intresserad av solinstrålningen för en byggnad är redo att sätta sig in i användningen av QGIS och insticksprogram.

En webbapplikation hade kunnat utvecklas likt den som tagits fram för Göteborgs Solkarta (se avsnitt 4.1). När en användare markerar en byggnad och får fram solinstrålningsinformation för taket, hade valmöjligheten samtidigt kunnat ges att skapa en 3D-modell i WebGL. WebGL har vissa prestanda relaterade problem men begränsas visualiseringen till ett hus i taget borde modellen kunna fungera felfritt. Den största utmaningen som måste lösas för implementeringen är kopplingen mellan husen och punktdata. Figur 9.3 redovisar hur 3D-modellen hade kunnat vara kopplad till webbapplikationen.



Figur 9.3, Potentiell webblösning för applikationen.

9.4 Möjligheter för annan användning

Så som applikationen är implementerad skulle den vara möjlig för en bred användning inom andra områden än solinstrålning. Det enda som applikationen kräver är vektorobjekt eller polygoner att knyta värden till och punktvärden med tillräckligt god upplösning för att de ska bli meningsfulla i förhållande till vektorobjektens storlek. En modell hade till exempel kunnat skapas och undersökas för ett objekt som har en area på mm²-nivå så länge det finns tillräckligt många mätvärden på samma yta.

10. Slutsatser

Arbetet har resulterat i en fungerande applikation som uppfyller alla krav som klassats som nödvändiga. Det är alltså möjligt att skapa en användbar visualiseringsapplikation för solinstrålning. Valet att jobba med QGIS har gjort bland annat arbetet med att koppla punktdata till byggnader enkelt. Utifrån omfattningen av avsnittet för diskussion är det tydligt att den prototyp som tagits fram genom SEBEvisualizer är väldigt grundläggande. Det finns flera tänkbara behov från olika av de tänkta användarna som inte helt har uppfyllts. Den nuvarande versionen av SEBEvisualizer bemöter bäst de behov som stora organisationer som kommuner och privata företag har. De enkla behov en privat användare har är också uppfyllda men programmets struktur är onödigt komplicerad för denna typ av användare. Det finns många möjligheter att expandera applikationens funktionalitet för att täcka fler behov men risken finns då att användningen blir mer komplicerad och på så sätt utesluter användare. Nyttan med visualisering av solinstrålningsdata är bred och behoven kan vara väldigt olika mellan användargrupper. Det bästa för framtida utveckling skulle vara att se de metoder som tagits fram inom detta projekt, som en stomme från vilken flera olika applikationer skulle kunna tas fram. Dessa kan riktas mot olika användargrupper och deras behov. Fortsätter utvecklingen att utgå ifrån insticksprogram till QGIS som systemarkitektur, är det enkelt att återanvända kod men att skraddarsy flera olika insticksprogram för olika typer av användning. Insticksprogrammen hade kunnat utgå ifrån samma databas och till och med bygga på eller kommunicera med varandra.

3D-visualiseringen genom *voxlar* skapar en miljö vars detaljrikedom enbart begränsas av indatans upplösning. Men en mer verklighetstrogen och effektiv modell hade kunnat tas fram genom polygonuppritning. Kan de problem som uppstår för en modell baserad polygonuppritning hanteras skulle detta vara en bättre lösning. Det finns också mycket att vinna på att arbeta mer med databasen i framtida utveckling. Området för fallstudien inom detta projekt är begränsat och innehåller redan en stor mängd punkter. Att skapa en 3D-modell över hela området är tids- och prestandakrävande. Eftersom målet är att expandera området till hela Göteborg krävs all optimering som är möjlig för att nå en så bra slutprodukt som möjligt. En fungerande 3D-modell över till exempel hela Göteborg är nog inte möjlig med dagens teknik genom metoderna som utvecklats för SEBEvisualizer. Men ju bättre optimerat alla delar är desto större områden kan visualiseras utan problem.

Den framtagna lösningens fördelar är främst snabb rendering och enkel utveckling. Spridningen genom insticksprogram till QGIS är mycket praktisk för användare som redan kan tänkas ha QGIS integrerat i sin verksamhet men för övriga användare krävs installation av ett stort program som har många funktioner som aldrig kommer att användas. Om det görs framsteg inom WebGL och andra webbaserade kartverktyg skulle en webbapplikation kunna uppfylla behoven hos flera användargrupper. Redovisning av statistik och Realtids rendering av enklare 3D-miljöer är inte en omöjlighet för en framtida webbapplikation. Samtidigt skulle spridningsmöjligheterna och användandet av applikationen underlättas. För användargrupper med mer komplexa behov som beräkningar direkt i applikationen och mer verklighetstrogna 3D-miljöer är det på grund av tidskrävande beräkningar och långa renderingstider lämpligt att en fristående applikation utvecklas.

Källförteckning

Artiklar

Energimyndigheten, 2014. *Fortsatt starkt intresse för solceller gav solcellseffekt på 43,1 MW under 2013*. Pressmeddelande från 2014-03-11

Lindberg, Fredrik. Jonsson, Per. Honjo, Tsuyoshi. Wästberg, Dag. *SOLAR ENERGY ON BUILDING ENVELOPES - 3D MODELLING IN A 2D ENVIRONMENT*. 2014

Lindberg, Fredrik. Grimmond, C.S.B. *Continuous sky view factor maps from high resolution urban digital elevation models*. Climate Res. Nr 42/2010

Reindl, D.T. Beckman, A. Duffie, J.A. *DIFFUSE FRACTION CORRELATIONS*. Solar Energy nr 45/1990

Böcker

Angel & Shreiner. 2012. *Interactive computer graphics – A top down approach with shader-based OpenGL, 6th edition*. Pearson education.

Kilgard, M.J & Fernando Randima. 2003. *The CG Tutorial: The Definitive Guide to Programmable Real-Time Graphics*. Addison-Wesley Professional

Saha, Kshudiram. 2008. *The Earth's Atmosphere - Its Physics and Dynamic*. Springer

Sherman 2014. *The PyQGIS progammer's guide*. Locate Press.

Rapporter

Caldwell, R. Douglas. 2005. *Unlocking the mysteries of the bounding box*.

Dijkstra, Tjeerd., Stijn, Oomes., Snoeren ,Peter. 1997. *3D Shape Representation: Transforming Polygons into Voxels*.

Energi Engagemang Sverige AB, 2014. *Förstudie om solenergi för Strängnäs Fastighets AB:s fastigheter*.

Energimyndigheten, 2010. *Solvärmestöd och marknadsutveckling*.

Hedén, Pontus. 2013. *Solelpotentialbedömning - framställning av solelpotentialkarta för Lund och utvärdering av laserdata*.

Jonsson, Per 2011. *Solar Energy from Existing Structures - Development of the SEES GIS tool that assesses solar irradiance on roofs*. WSP Analysis & Strategy.

Lindahl, Johan. 2013. *National Survey Report of PV Power Applications in Sweden 2012*.

Munari, Probst., Maria, Cristina., Roecker, Christian. 2012. *Solar energy systems in architecture – integration criteria and guidelines*.

Naturvårdsverket. 2010. *Förnybara energikällors inverkan på de svenska miljömålen*.

Sinclair Knight Merz. 2010. *The renewable energy potential of Victoria -Part 1 Energy Resources*.

Stridh, Bengt. 2005. *Nätanslutning av småskaliga solcellssystem för elförsäljning*.

Whitted, Turner. 1979. *An Improved Illumination Model for Shaded Display*.

Williams, Lance. 1978. *Casting curved shadows on curved surfaces*.

Webbsidor

Arko, Scott, 2014. *QGIS plugins – OpenLayers*.

<http://www.digital-geography.com/qgis-plugins-openlayers/> (Hämtad 2014-09-14)

Chua Hock-Chuan, 2012. *OpenGL Tutorial*.

http://www3.ntu.edu.sg/home/ehchua/programming/opengl/CG_Introduction.html (Hämtad 2014-05-25)

Energimyndigheten, 2012a. *Stark tillväxt för solenergin*.

<http://www.energimyndigheten.se/Press/Nyheter/Nyhetsarkiv-2012/Stark-tillvaxt-for-solenergin/> (Hämtad 2014-08-06)

Energimyndigheten, 2012b. *Vad är det för skillnad på solceller och solfångare?*

<http://www.energimyndigheten.se/Om-oss/FAQ/Vad-ar-det-for-skillnad-pa-solceller-och-solfangare/> (Hämtad 2014-11-29)

Esri inc, 2007. *Calculating solar radiation*.

http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm?TopicName=Calculating_solar_radiation (Hämtad 2014-08-20)

Esri inc, 2013. *ESRI ASCII raster format*.

http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/ESRI_ASCII_raster_format/009t0000000z000000/ (Hämtad 2014-04-11).

European Commission, 2012. *Solar radiation (Europe) in PVGIS*.

<http://re.jrc.ec.europa.eu/pvgis/solres/solrespvgis.htm> (Hämtad 2014-09-01)

Free Software Foundation, 2007. *GNU General Public License*.

<http://www.gnu.org/licenses/gpl.html> (Hämtad 2014-11-05)

Gisela, 2014. *Vektordata*

<http://gisela.humangeo.su.se/Vektordata.ashx> (Hämtad 2014-08-28)

GRASS Development team, 2013a. *3D raster data (voxel) processing in GRASS GIS*.

<http://grass.osgeo.org/grass70/manuals/raster3dintro.html> (Hämtad 2014-03-25).

GRASS Development team, 2013b. *R3.in.ascii*.

<http://grass.osgeo.org/grass70/manuals/r3.in.ascii.html> (Hämtad 2014-03-25).

GRASS Development team, 2012. *What's GRASS GIS?*

<http://grass.osgeo.org/documentation/general-overview/> (Hämtad 2014-03-10).

Göteborg Energi AB, 2014. *Solkartan*.

http://www.goteborgenergi.se/privat/projekt_och_etableringar/fornyelsebar_energi/solceller/solkartan (Hämtad 2014-04-20).

JRC European Commission, 2014. *Photovoltaic Geographical Information System - Interactive Maps*.

<http://re.jrc.ec.europa.eu/pvgis/apps4/pvest.php#> (Hämtad 2014-09-14)

Khronos Group, 2014. *OpenGL ES 2.0 for the Web*.

<http://www.khronos.org/webgl/> (Hämtad 2014-04-20).

Kraftringen, 2014. *Solkartan*.

<http://www.kraftringen.se/Privat/Solceller/Solkartan/> (Hämtad 2014-09-14)

Lantmäteriet 2014a, *RH 2000*.

<http://www.lantmateriet.se/sv/Kartor-och-geografisk-information/GPS-och-geodetisk-matning/Referenssystem/Hojdsystem/RH-2000/> (Hämtad 2014-09-10)

Lantmäteriet 2014b, *SWEREF99, Projektioner*.

<http://www.lantmateriet.se/Kartor-och-geografisk-information/GPS-och-geodetisk-matning/Referenssystem/Tvadimensionella-system/SWEREF-99-projektioner/> (Hämtad 2014-05-20)

Naturvårdsverket 2014, *Begränsad klimatpåverkan*.

<http://www.miljomal.se/sv/Miljomalen/1-Begransad-klimatpaverkan/> (Hämtad 2014-09-27)

OpenLayers, 2014. *OpenLayers 3*.

<http://openlayers.org/> (Hämtad 2014-11-05)

OpenSourceOrg, 2014. *The BSD 2-Clause License*

<http://opensource.org/licenses/BSD-2-Clause> (Hämtad 2014-11-05)

OpenStreetMap, 2014a. *About*.

<http://www.openstreetmap.org/about> (Hämtad 2014-05-20)

OpenStreetMap, 2014b. *Copyright and license*.

http://pyopengl.sourceforge.net/documentation/opengl_diffs.html (Hämtad 2014-11-05)

OpenStreetMap Wiki, 2014. *QGIS*.

<http://wiki.openstreetmap.org/wiki/QGIS> (Hämtad 2014-05-20)

Pirmin Kalberer & Mathias Walker, Sourcepole AG, 2014. *Openlayers plugin for QGIS*.

<https://github.com/sourcepole/qgis-openlayers-plugin> (Hämtad 2014-11-05)

Python Software Foundation, 2014. *Sqlite3 – DB – API 2.0 Interface for SQLite Databases*.

<https://docs.python.org/2/library/sqlite3.html> (Hämtad 2014-09-01)

Qt Project, 2014. *Qt project*.
<https://qt-project.org/> (Hämtad 2014-08-19)

Qt Project, 2013. *QGLWidget Class Reference*.
<http://qt-project.org/doc/qt-4.8/qglwidget.html> (Hämtad 2014-08-15)

Silicon Graphics, Inc. 2006. *glBegin*.
<https://www.opengl.org/sdk/docs/man2/xhtml/glBegin.xml> (Hämtad 2014-06-02)

Silicon Graphics, Inc. 2014. SGI OpenGL Licensing Programs.
<http://opengl3.org/about/licensing/> (Hämtad 2014-11-05)

SolEI-Programmet, 2014. *Energiberäkningar*.
<http://www.solelprogrammet.se/projekteringsverktyg/energiberakningar/> (Hämtad 2014-04-22).

SourceForge, 2014a. *About PyOpenGL*.
<http://pyopengl.sourceforge.net/index.html> (Hämtad 2014-09-15)

SourceForge, 2014b. *Speed Concerns*.
<http://pyopengl.sourceforge.net/index.html> (Hämtad 2014-11-22)

Stockholms stad, 2014. *Stockholms Solkarta*.
<http://www.stockholm.se/stockholmsolkarta> (Hämtad 2014-09-14)

The GeoServer Project, 2014a. *GeoServer User Manual*.
<http://docs.geoserver.org/stable/en/user/> (Hämtad 2014-09-08)

The GeoServer Project, 2014b. *GetFeatureInfo Templates*.
<http://docs.geoserver.org/stable/en/user/tutorials/GetFeatureInfo/index.html> (Hämtad 2014-09-08)

Wikimedia Commons, 2014. *Voxel-polygon-alleycats*.
<http://commons.wikimedia.org/wiki/File:Voxel-polygon-alleycats-v01.gif> (Hämtad 2014-09-22)

Wikipedia, 2014a. *Comparison of OpenGL and Direct3D*.
http://en.wikipedia.org/wiki/Comparison_of_OpenGL_and_Direct3D

Wikipedia, 2014b. *Heat map*.
http://en.wikipedia.org/wiki/Heat_map (Hämtad 2014-09-02)

Wikipedia, 2014c. *Vertex (computer graphics)*.
[http://en.wikipedia.org/wiki/Vertex_\(computer_graphics\)](http://en.wikipedia.org/wiki/Vertex_(computer_graphics)) (Hämtad 2014-05-25)

APPENDIX A – Användarmanual

A.1 Initiering

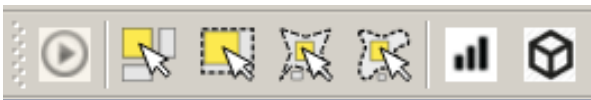
För att skapa ett tydligt arbetsflöde för applikation görs verktygen inte tillgängliga förrän då de är lämpliga att använda. Figur A.1 visar applikationens menyrad innan initiering och figur A.2 visar menyraden efter initiering.

1. Först måste applikationen initieras vilket skapar en bakgrundskarta som överlagras av ett vektorlager med byggnader i området.



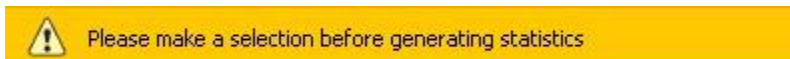
Figur A.1, Verktygsfält innan initiering.

2. Efter initiering får användaren tillgång till markeringsverktyg där en eller flera byggnader i vektorlagret kan markeras.



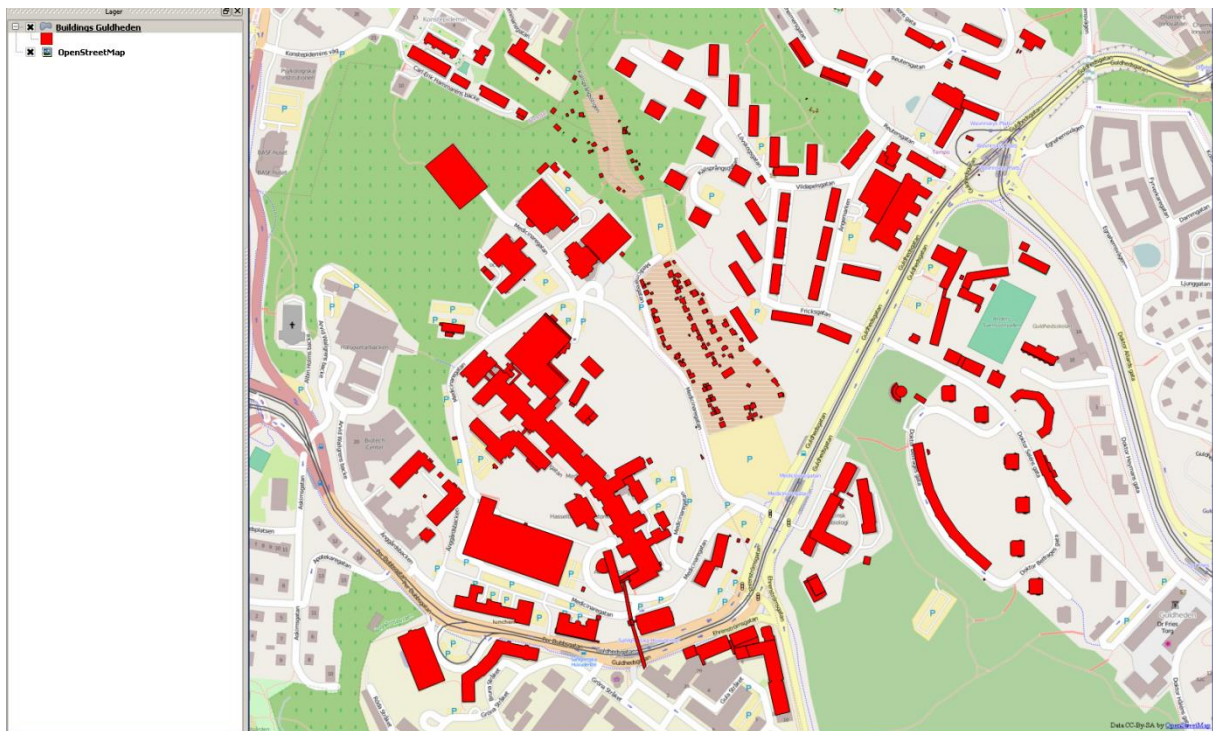
Figur A.2, Verktygsfält efter initiering.

3. Om användaren gör en markering ges möjligheten att ta fram statistik för solinstrålning för markeringen eller att utifrån markeringen skapa en 3D-modell. Försöker användaren skapa en 3D-modell eller statistik utan att ha gjort markering skapas istället ett meddelande som uppmanar användaren att göra markering. Figur A.3 visar varningsmeddelande.



Figur A.3, Varningsmeddelande som visas om användaren försöker generera statistik utan att har markerat något objekt.

Figur A.4 visar karttyv efter initiering.



Figur A.4, Kartvy efter initiering.

A.2 Markering

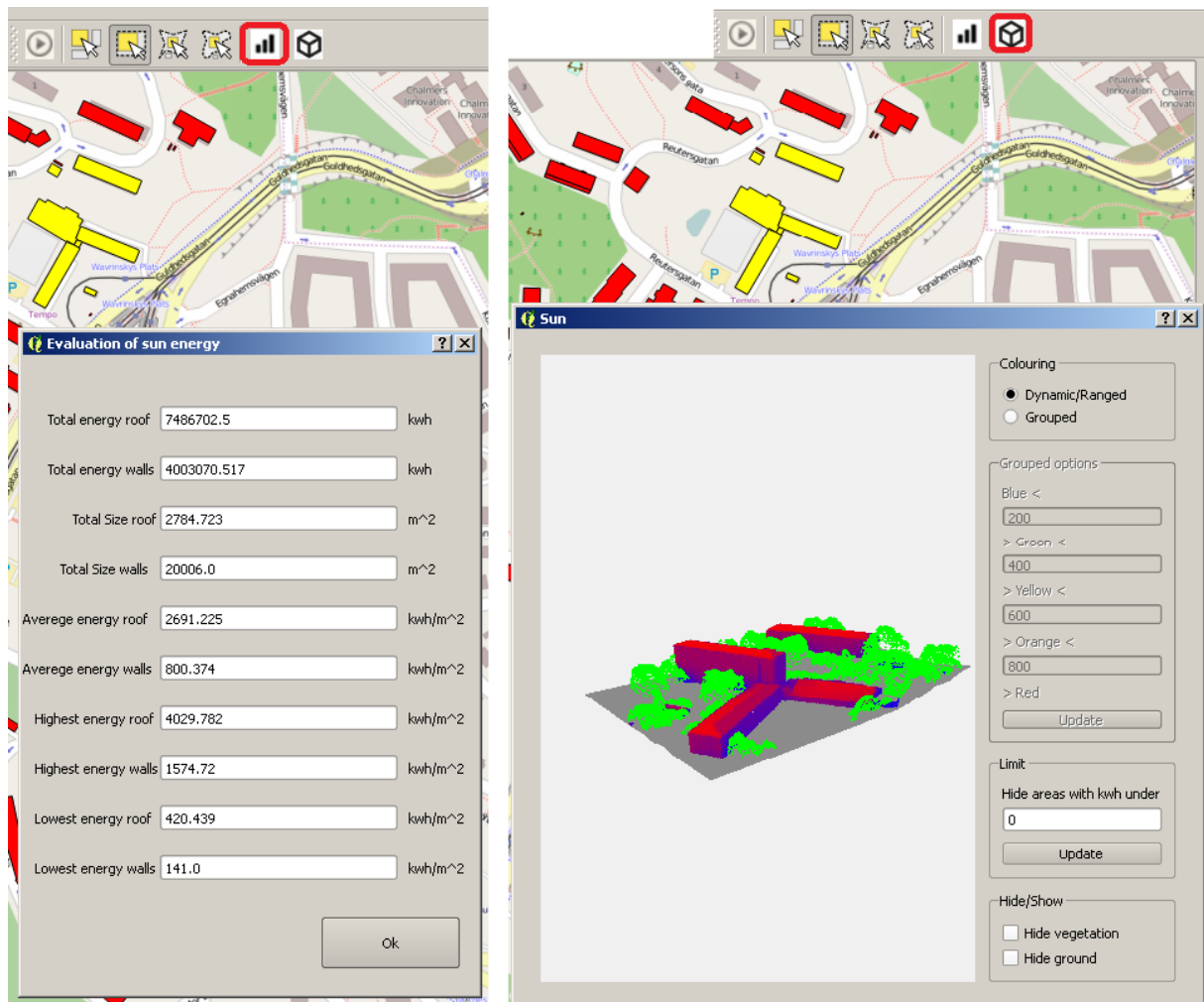
Nästa steg i användningen är att en användare markerar den eller de byggnader som som är intressanta. Genom QGIS redan utvecklade verktyg för markering av vektorobjekt kan en användare göra olika typer av markeringar på vektorlagret för byggnader. Se Figur A.5 för ett exempel på markering med markeringsverktyg.



Figur A.5, Markeringsverktyg samt markering på vektorlager.

A.3 Generera statistik/3D-modell

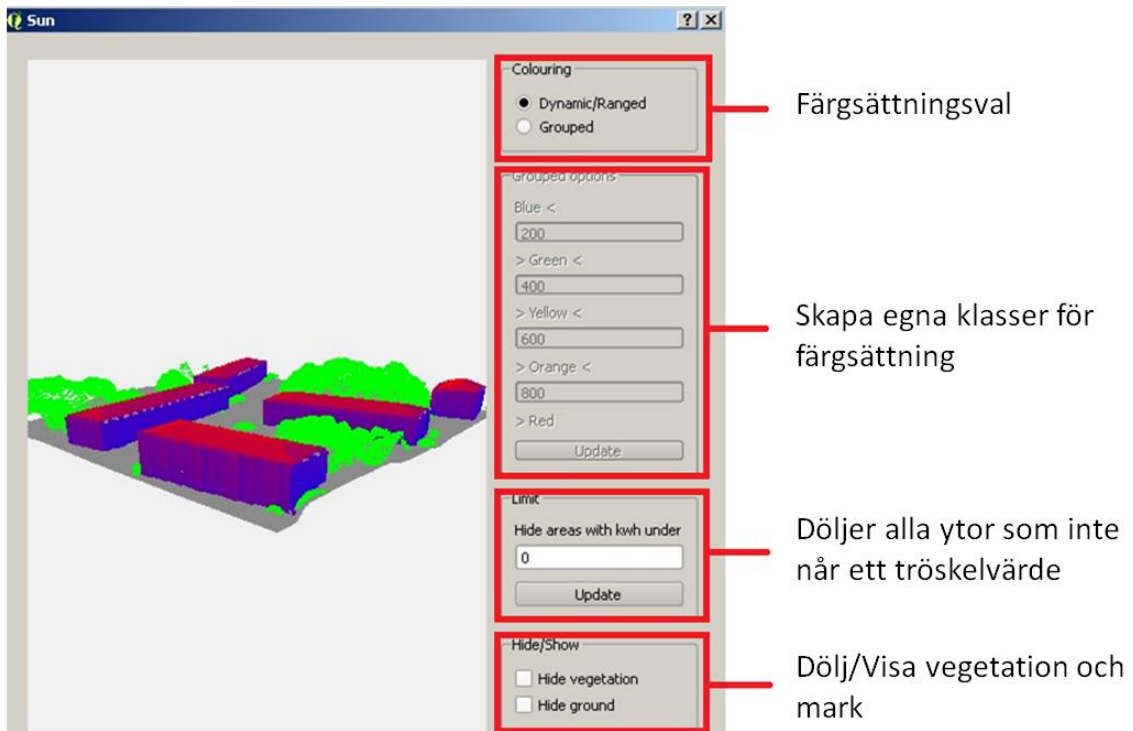
När en användare har markerat intressanta objekt kan statistik eller 3D-modell genereras för det markerade området genom två olika menyobjekt. För 3D-modellen används alla punkter som är en del av markeringens begränsningsruta och för statistik görs en mer detaljerad selektion bland punktsamlingen då den bara får innehålla punkter för de markerade byggnader och inte delar av omkringliggande byggnader. Figur A.6 visar fönster för 3D-modell och statistik.



Figur A.6, Statistik och 3D-modell för markerat område.

A.4 3D-modellens användargränssnitt

Som kan ses i figur A.6 har 3D-modellen ett tillhörande användargränssnitt. Detta användargränssnitt tillåter att den framtagna modellens utseende modifieras. Genom musrörelser kan en användare förflytta sig runt i modellen och genom användargränssnittskomponenter förändras modellen. Colouring tillåter att färgschemat för modellen ändras mellan två alternativ. För alternativet baserat på grupperade färger tillåts att användaren ställer upp egna gränsvärden. Användaren kan också dölja delar av modellen som vegetation, markytor och alla byggnadsytor som inte når ett visst solinstrålningsvärde. Figur A.7 redovisar möjligheterna för 3D-modellens användargränssnitt.



Figur A.7, 3D-modellens användargränssnitt.

APPENDIX B & C – Källkod

Appendix med källkod för insticksprogram och 3D-modell finns på <http://lup.lub.lu.se/>.

Institutionen av naturgeografi och ekosystemvetenskap, Lunds Universitet.

Student examensarbete (Seminarieuppsatser) i geografisk informationsteknik. Uppsatserna finns tillgängliga på institutionens geobibliotek, Sölvegatan 12, 223 62 LUND. Serien startade 2010. Hela listan och själva uppsatserna är även tillgängliga på LUP student papers och via Geobiblioteket (www.geobib.lu.se)

Serie examensarbete i geografisk informationsteknik

- 1 *Patrik Carlsson och Ulrik Nilsson* (2010) Tredimensionella GIS vid fastighetsförvaltning.
- 2 *Karin Ekman och Anna Felleson* (2010) Att välja grundläggande karttjänst – Utveckling av jämförelsemodell och testverktyg för utvärdering
- 3 *Jakob Mattsson* (2011) Synkronisering av vägdatabaser med KML och GeorSS - En fallstudie i Trafikverkets verksamhet
- 4 *Patrik Andersson and Anders Jürisoo* (2011) Effective use of open source GIS in rural planning in South Africa
- 5 *Nariman Emamian och Martin Fredriksson* (2012) Visualisering av bygglovsärenden med hjälp av Open Source-verktyg - En undersökning kring hur man kan effektivisera ärendehantering med hjälp av en webbapplikation
- 6 *Gustav Ekstedt and Torkel Endoff* (2012) Design and Development of a Mobile GIS Application for Municipal Field Work
- 7 *Karl Söderberg* (2012) Smartphones and 3D Augmented Reality for disaster management - A study of smartphones ability to visualise 3D objects in augmented reality to aid emergency workers in disaster management
- 8 *Viktoria Strömberg* (2012) Volymberäkning i samhällsbyggnadsprojekt
- 9 *Daniel Persson* (2013) Lagring och webbaserad visualisering av 3D-stadsmodeller - En pilotstudie i Kristianstad kommun
- 10 *Danebjerg Lisette och Nyberg Magdalena* (2013) Utbyte av geodata - studie av leveransstrukturer enligt Sveriges kommuner och landstings objekttypskatalog
- 11 *Alexander Quist* (2013) Undersökning och utveckling av ett mobilt GIS-system för kommunal verksamhet
- 12 *Nariman Emamian* (2014) Visning av geotekniska provborrningar i en webbmiljö
- 13 *Martin Fredriksson* (2014) Integrering av BIM och GIS med spatiala databaser – En prestandaanalys
- 14 *Niklas Krave* (2014) Utveckling av en visualiseringsapplikation för solinstrålningsdata