Blekinge tekniska
högskola – BTH

Examensarbete

GPS-baserad virtuell geografisk inhägnad för mobil enhet

Kurs:

Deltagare: Jim Gunnarsson, di98jgu@gmail.com

Handledare: Anders Nelsson, anders.nelsson@bth.se

Källkod: https://github.com/di98jgu/

Datum:

Sammanfattning

Sammanfattning av detta exjobb...

Tekniks info som nyckel ord och dylikt samlas här på lämpligt sätt.

Innehåll

1 Inledning

1.1 Mobil utveckling

Mitt val av examensarbete speglar mitt intresse för den mobila utvecklingen som telefon, surfplatta etc. Den hastighet med vilken utveckling nu sker är mycket spännande. De är ett komplett datorsystem med nätverk, sensorer, och gränssnitt. Det som nu sker förändrar i grunden hur vi använder datorer och till vad.

1.2 Mobil säkerhet inom företag

Mobila enheter används allt mer systematiskt inom företag i den dagliga verksamheten. De är numera inte bara ett kommunikationshjälpmedel utan en förlängd del av företaget. En viktig del i trenden att kunna arbeta var som helst när som helst.

Mobila enheter utgör också risk då de kan innehålla information av intresse för utomstående. Det finns följande hinder för att bygga upp ett gott skydd[1]:

- Säkerhet är besvärligt från en användarens synvinkel. Säkerhet handlar till stor del om reglerad resurshantering, vem som har tillgång till vad och hur. En mobil enhet är en resurs som kräver ett visst handhavande.
- Datorer är kraftfulla och komplexa. En mobil enhet är en komplett dator och används till allt fler uppgifter. De innehåller och hantera känslig information som har ett ekonomiskt värde. Typiskt företagshemligheter, nätverksinformation, persondata etc.
- Datoranvändare är osofistikerade, de har liten eller ingen kännedom om säkerhet och vilka hot som finns. En mobil enhet är lätt att glömma eller av olika anledning lämna utan övervakning.
- Datorer skapades utan en tanke på säkerhet . Detta är inte sant vad gäller Android. I grunden så är Android ett Linux system. Användaren måste vid installation godkänna tillgång till resurser som kontakt uppgifter, Internet, GPS med flera.
- Trenden är att dela information, inte att skydda den. Typiskt exempel är sociala nätverk av olika slag.
- Information ska vara tillgängliga överallt. Den mobila enheten finns med överallt och är ständigt uppkopplad. Det är viktigt att vara uppdaterad.

1

- Säkerhet handlar inte om hårdvara och mjukvara. Säkerhet beskrivs som process som handlar först och främst handlar om de som använder hårdvara och mjukvara.
- Skurkarna är mycket sofistikerade. Immateriella tillgångar och kreditkortsuppgifter har ett ekonomiskt värde. Med ett minne på 512 Mbyte och uppåt kan en typisk mobil enhet innehålla en ansenlig mängd information.
- Ledningen ser säkerhet som en kostnad.

Mobila enheter behöver därför precis som all annan datorutrustning administreras. Ett viktigt verktyg för skydda den mobila enheten är att veta var den är. Positionering som GPS mottagare är de facto standard i de flesta enheter idag. Genom att upprätta en virtuell inhägnad runt enheten så kan man binda den till ett bestämt geografiskt område. Enheten larmar om den lämnar området eller vidtar andra åtgärder. Man kan tänka sig att man har flera områden med olika egenskaper.

2 Frågeställning

2.1 Problemformulering

En given enhet ska hägnas in med ett virtuellt staket med hjälp av GPS. Kan till exempel vara en telefon surfplatta etc. Inhägnaden formar ett område som innesluter enheten. Lämnar enheten området ska ett larm aktiveras. Området är en godtycklig månghörning, polynom. Man kan tänka sig flera områden med olika egenskaper.

Med larm avses att användaren av enheten informeras om att enheten har lämnat området. Bristande noggrannhet vid GPS positionering samt att positionering sker vid vissa tidpunkter gör att enhetens exakta läge vid en viss given tidpunkt kan vara svår att bestämma. Det finns en gränszoom mellan utanför och innanför.

Sverige är satt som en övre gräns för områdes omfattning. Noggrannhet och praktisk användbarhet sätter en nedre gräns. Enheten administreras centralt så ett område kan antingen skapas direkt på den mobila enheten eller hämtas från lämpligt kartmaterial. Etablering och upprätthållande av inhägnad sker på själva enheten.

Målsättningen är att bestämma den eller de algoritmer som behövs för att skapa en virtuellt inhägnad. Ett område ska kunna upprättas och upprätthållas på en mobil enhet med någon form av larm. Detta är vad som skall göras:

- Att från en källa hämta koordinater för den virtuella inhägnaden.
- Att omvandla koordinater till ett lämplig format.
- Bestämma lämplig algoritm eller algoritmer som behövs för att skapa och upprätthålla en virtuell geografisk inhägnad.
- Att hämta aktuell position med hjälp av GPS.
- Att informera användaren om enheten lämnar den virtuella inhägnaden.

2.2 Förutsättningar

Ambitionen för detta arbete ligger på att finna en för mobil enhet lämplig algoritm och testa den i ett faktiskt system. Vi har följande:

- Android är målsystemet för tester av GPS inhägnaden. Detta motiveras av att utvecklingsmiljön för Android är fri öppen och lättillgänglig.

- Algoritmen ska vara förutsättningslös vad gäller målenhet. Det finns ett flertal olika operativ system för mobila enheter samt olika versioner av dessa.
- Ambitionen är att algoritmen ska vara väl lämpad för rådande förutsättningar. En mobil enhet har till exempel begränsad batterikapacitet.
- Dokumentera algoritmen så att den lätt kan implementeras på önskat system.

Beroende på hur mycket tid som finns tillgänglig och vilka problem som dyker upp så kan arbetet komma att utökas och anpassas. Men detta sker i så fall med algoritmen för GPS inhägnad i fokus. Det kan vara att testa ett flertal enheter för ökad förståelse av hur väl systemet fungera i verkligheten. Kanske finns det andra algoritmer som kan vara av intresse. Hur påverkar brister i GPS positionering systemet och vad kan göras för att kompensera för detta.

2.3 Avgränsningar

Detta examensarbete är i viss mån flexibelt då det kan utökas om tid finns. Det kan vara att testa mer än en algoritm till exempel. Här räknas därför endast upp de avgränsningar upp där resurser helt saknas eller där tid inte finns alls.

- Att systematiskt testa olika mobila enheter. Skulle kräva omfattande tillgång till hårdvara. En eller ett fåtal enheter kommer att användas.
- Att utveckla en för vanliga användare fungerande applikation. Faller helt utanför målsättningen med detta examensarbete som är att hitta lämplig algoritm för GPS inhägnad.
- Att låt användaren rita upp eller på annat sätt skapa ett område för GPS inhägnad. Med detta avses att vi går från ett grafiskt gränssnitt där användaren väljer ett lämpligt område till en uppsättning koordinater. Detta är ett rent användarfall som inte tillför något.
- Att hantera flera områden. Det antas här att om vi kan hantera ett område så kan flera hanteras. Här avses implementation och praktiska tester. Val av algoritm sker med målsättningen att flera områden ska kunna hanteras.

3 Resultat

3.1 Definition av mobil enhet

3.1.1 Mobil enhet

Men en mobil enhet avses genomgående i detta arbete någon form av mobil enhet som användaren bär med sig. Typiskt en mobiltelefon och surfplatta. Fast det har börjat dyka upp andra typer av enheter som kameror med en mer öppen arkitektur. Att den är mobil och bärbar det är det viktigaste inte typ av enhet.

Praktiska omständigheter diktera att operativsystemet är Android, tillgång till andra utvecklingsmiljöer saknas under detta arbete. För att säkerställa att algoritmen är förutsättningslös vad gäller operativsystem så bör koppling mellan algoritm och Google API vara så liten som möjligt. Koden kommer att köras på själva enheten ej centralt.

3.1.2 Systemkrav

En mobil enhet karaktäriseras av att den har en processor med god kapacitet. Så pass god att java används som primärt programmeringsspråk i Android. Det var uppenbart en klokt val. Det räcker därför med att skatta stora O för att bedöma olika algoritmer.

Typiskt så har en mobil enhet 512 Mbyte i RAM minne och uppåt. Moderna operativsystem maximera användning av minne för att minimera svarstid. På Android så innebär det att applikationer försätts i dvala hellre än att helt stänga ner dem. För att dessa vilande applikationer inte ska ta upp allt minne så finns särskilda metoder för att återkräva minne. Administrativa metoder för att spara undan data kan därför behövas.

Batterikapacitet är den sista stora barriären vad gäller mobila enheter. Det är lätt att skapa applikationer som snabbt dränera all tillgänglig energi. GPS tillsammans med radio tar mycket energi och bör därför användas så sparsamt som möjligt. Ett sätt att göra detta är att dynamiskt reglera tiden mellan en positionering till en annan.

Det är lämpligt att reducera omfattande beräkningsoperationer vid regelbundet återkommande uppgifter som att kontrollera att användaren befinner sig inom inhägnaden[2]. Detta är typiskt en bakgrundsuppgift och samsas med andra bakgrundsuppgifter.

2 *Android Developers (u.å.) Performance Tips [Internet] Från: developer.android.com/training/articles/perf-tips.html [Hämtad: 3 Sep 2013]*

3.2 Positionering

3.2.1 Positionsleverantörer

I Android används flera olika tekniker för att tillhandahålla positionering[3]. Dessa benämns positionsleverantörer, eng. *location Providers*, vilka har sin unika uppsättning av styrkor och svagheter. De olika typerna av leverantörer komplettera varandra genom att vara lämpliga i olika situationer. Positionering hanteras av klassen *LocationManager*.

3.2.2 GPS

Med GPS så bestäms aktuell position hjälp av satelliter i omloppsbana runt jorden. Det behövs minst tre för att bestämma longitud och latitud. Precisionen är hög men kräver fri sikt till satelliterna. Därför fungera tekniken dåligt inomhus. Signalen kan studsa mot byggnader och liknande vilket förvränger signalen och minskar möjligheten att bestämma position. Detta gör att GPS fungera mindre bra i urbana miljöer.

3.2.3 Trådlösa nätverk WI-FI

Trådlösa nätverk kan användas för att bestämma enhetens position. Enheten samlar information om de basstationer som finns inom räckhåll. Informationen skickas till Google som tillhandahåller känd positionsdata om basstationerna. Varje basstation har en unik MAC adress och kan därför identifieras. Det är ett ömsesidigt utbyte av information. Google samlar på detta sätt systematiskt in data om basstationer. En basstation kan dock lätt flyttas och därmed så stämmer dess position inte längre.

3.2.4 Mobiltelefonnätet

Varje mast har ett unikt id vilket gör möjligt att bestämma position på motsvarande sätt som för trådlösa basstationer. Då en mobil enhet i regel har täckning så finns goda möjligheter att bestämma enhetens position. För öka precisionen så kan information om föregående master användas.

3.2.5 Passiv positionsbestämning

Passiv positionsbestämning innebär att applikation lyssnar efter uppdateringar av positionen men utan att själv begära en uppdatering. Detta används för program som körs i bakgrunden. Poängen är applikationen alltid har tillgång till senast kända position, användaren behöver aldrig vänta på att applikation ska hämta in aktuell position.

3

3.2.6 Positionering i praktiken

Det finns lite information om hur väl positionering fungera i praktiken. Svårigheten ligger i att den information som finns endast beskriver mjukvaran, alltså Android, men inte hårdvaran. Det är rimligt att finns stora skillnader mellan olika mobila enheter.

Generellt så gäller att GPS positionering är långsammast, upp till flera minuter för erhålla aktuell position, men ger störst precision. Enligt lantmätriet ca 10 meter. Nätverksleveratörer, alltså trådlösa nätverk och mobilnätverk är snabbare mer ger sämre precision. De har fördelen mot GPS att de drar mindre energi.

3.3 Virtuell geografisk inhägnad

3.3.1 Definition av område

Ett geografiskt område beskrivs av en konkav eller konvex polygon. Polygonen byggs upp av ett antal punkter, koordinater. Att polygon beskriver ett geografiskt område innebär att polygonen sluten och väldefinierad. Det sista innebär att överlappande linjer och eller koordinater inte får förekomma. Systemet ska kunna hantera ett större antal koordinater. Sveriges landgräns inkluderat kust är ca 5400 km. Så en övre gräns på 5000 koordinater bör vara fullt tillräckligt.

Enheten antas befinnas sig i området även om inget hindra att samma algoritm används för att exkludera enheten från ett givet område. Enheten är en till antalet, något som inte alls är självklart ty man kan lätt föreställa sig en situation med flera enheter där deras inbördes relation till varandra är av intresse. Men detta är en helt annan frågeställning.

Onoggrannhet vid positionering gör att det kan vara svårt att strikt bestämma om enheten är ute eller inne. En lösning är att upprätta två områden. Ett inre område för att varna och ett yttre för att larma. Detta förutsätter att onoggrannheten ej överstiger ett visst värde.

3.3.2 Koordinatsystem

Ett område kan antingen skapas av användaren på den aktuella enheten eller tillhandahållas från extern källa. Både är rimliga. I det här fallet så är enheten en klient i ett nätverk som administreras centralt.

Skapas området på själva enheten så kan man tänka sig att området ritas eller annat sätt markeras. Google maps som finns tillgängligt på Android och utgör, åtminstone på Android, enkel lösning. Apple tillhandahåller numera egna kartor för Iphone.

En annan lösning är att hämta gränser från befintligt kartmaterial. Kartmaterial från Lantmäteriet anger koordinater enligt RT90 eller SWEREF 99. Gränsen går vid 2007 då Lantmäteriet gick från RT90 till SWEREF 99. Detta är plana koordinater och måste översättas till GPS koordinater. Vi utgår här från att systemet kommer att användas i Sverige.

Det finns inget behov av att mäta faktiska avstånd eller vinklar. Detta underlätta då översättning från ett koordinat system till ett annat är en komplicerad procedur. Inhägnaden kommer internt att arbeta med enhetslösa x och y koordinater. All översättning mellan olika koordinatsystem ligger utanför hanteringen av inhägnaden. Detta innebär att översättning av olika koordinat-system är en problematik skild från hantering av inhägnaden.

3.3.3 Egenskaper hos enhet

Området antas vara stort i förhållande till rörelsehastigheten hos enheten. Det betyder att det tar en viss tid för användaren av enheten att flytta sig från en sidan av inhägnaden till den andra. Detta är inget krav men spar batteri ty positionering kan ske med större tidsintervall.

Användarens rörelsemönster är inte slumpartat utan beror av vad användaren befann sig vid föregående tillfälle. Vidare om enheten bärs av en användare till fots så kan vi anta att enheten rör sig lite eller inte alls under längre perioder.

Det är alltså lönt att skatta rörelsehastighet hos enheten. Det gör att mindre avbrott vid positionering kan kompenseras. Enheten kan avvakta med att rapportera saknad position och göra ytligare försök att fastsälla enhetens position. Rörelsehastighet är också användbart för att bestämma behov av positionering.

Enklast är att skriva in enheten i ett fyrkantigt delområde innanför inhägnaden. Detta mindre område har kända dimensioner och med skattad rörelsehastighet och senaste position vet vi när användaren tidigast kommer lämna delområdet. När enheten lämnar delområdet upprättas ett nytt delområde runt enheten.

3.3.4 Flera områden

Det finns ett intresse av att hantera flera områden. Behovet av resurser ökar i direkt proportion till antalet områden. Skrivs enheten in ett delområde blir hantering av flera områden enklare. Vi vet direkt i vilket eller vilka områden som enheten befinner sig. Man kan tänka sig att skapa relationer mellan de olika områdena.

Bortser man från relationer så är flera områden endast en lista med enskilda områden. Att hantera flera områden blir då en mer administrativ till sin karaktär. Relationer mellan områden underlättas om delområdena är geometriskt enkla som cirklar, trianglar och rektanglar.

3.4 Algoritm för virtuell inhägnad

3.4.1 Val av algoritm

Målet med de två föregående kapitlen var att bestämma algoritmens egenskaper algoritmen. Genom att skatta användarens rörelse mönster så kan systemet göras mer tillförlitligt. Det ska gå snabbt att bestämma om användaren befinner sig i visst område eller inte. Ett delområde upprättas runt användaren för att minimera behovet av att uppdatera GPS positionen. Vi kan spara undan data så att en initialt tidskrävande algoritm bara behöver köras en gång. Det krävs en ökad administration men att skyffla data är inget problem.

Det finns ett rikt utbud på olika algoritmer för polygoner. Den enklaste av dessa är låta en tänkt linje utgå från enheten till yttre gränsen för polygonen. Genom att räkna antalet gånger som linjen skär polygonen vet vi om vi inne eller ute. Udda antal skärningar så är enheten inne och jämt antal ute. De yttre gränserna kan enkelt bestäms genom att gå ett varv runt polygonen.

Den här typen algoritm är enkel och kräver inget minne. Den har två problem. Vi behöver kortaste avstånd till kant från given punkt i polygonen. Detta kräver att vi räknar på varje delsträcka. Det andra problemet är att algoritmen kräver ett varv runt polygon. Det ger komplexiteten stora O(n) där n är antalet koordinater. Det helt acceptabelt för ett mindre antal koordinater men vi har som målsättning att kunna hantera mer än 1000 koordinater.

Den mest lämpade och intressanta som jag funnit är att dela in polygonen i rutor. Grundidén här att rutorna delas in i tre kategorier. De är inne, ute eller innehåller en del av polygonen. Detta gör det går snabbt att bestämma om enheten är inne eller ute ty även då enheten befinner sig inom en ruta som korsas av polygonen så är beräkningarna reducerade till endast den rutan.

I bästa fall så är komplexiteten O(1) i värsta fall går den mot O(n). Det första fallet får vi om enheten antingen befinner sig ute eller inne. Det senare är specialfall, merparten av alla koordinater är samlade på ett mycket litet område. Detta är normalt inte trolig utan vi kan anta koordinaterna är jämt fördelade.

Låt oss anta att vi har en rektangel liknande polygon på 1000 koordinater och rutnätet är 30 gånger 30 rutor. Omkretsen är då 118 rutor eller 13 % av alla rutor upptas av polygonen resten är innanför och inga utanför. Rutnätet är optimerat för den aktuella polygon och är endast något större än polygonens min och max värde. Antalet koordinater per ruta är mindre än 10.

Nackdelen med ett rutnät är att det initialt krävs en hel del räknade. Varje delsträcka som ingår i polygonen behöver räkas ruta för ruta. Vi får O(n*m) där n är antalet koordinater och m är antalet rutor som en viss delsträcka passera. En diagonal från hörn till hörn skär maximalt antal rutor men vi kan sätta m till sidan av rutnätet då konstanten faller bort, alltså 30 i fallet ovan. Men även detta är ett specialfall, i verkligheten blir få eller inga sträckor så långa.

Det tillkommer ytligare initialt arbete. Vi måste säkerställa att inga koordinater eller sträckor överlappa. Det löser delvis kravet på en väl definierad polynom. Två sträckor kan fortfarande ligga mycket nära varandra.

3.4.2 Upprättande av inhägnad

De två viktigaste delarna för att upprätta ett rutnät är dela upp polynom ruta för ruta och att bestämma om rutornas hörn är inne eller ute. En given polynom är en serie av av segment där varje segment behandlas var för sig. Studera vi ett segment ab från en given polygonen så har vi tre grundläggande fall, se bild 1.

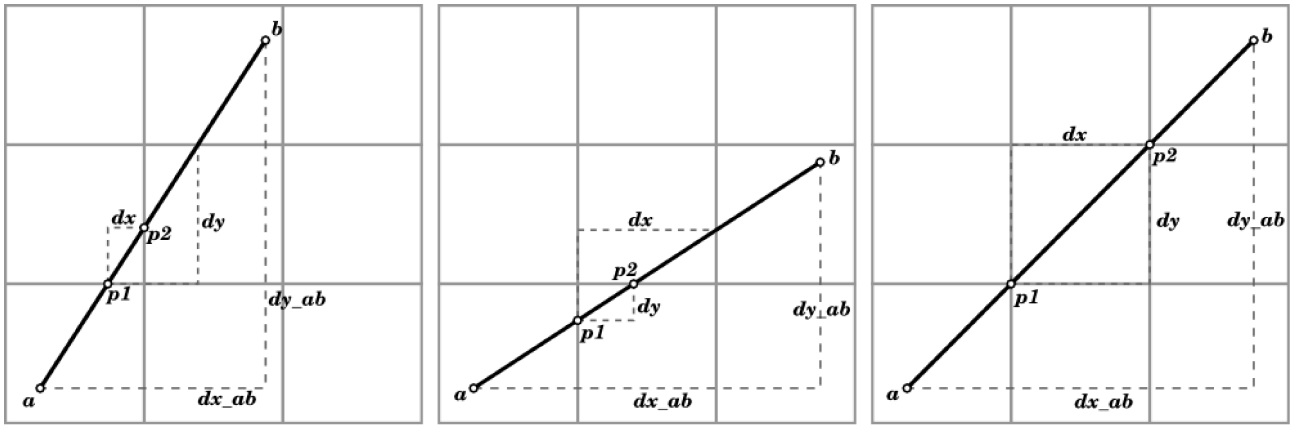

Bild 1: Ett segment från en given polygon

Betrakta segmentet som hypotenusan i rätvinklig triangel. Vi stycka upp segmentet i delsegment där segmentet skär rutnätet. Vi får då en delsträcka a till p1 och p1 till p2 och så vidare. Summan är av de olika sträckorna är den totala längden på segmentet.

Anta att vi står i punkt p1 och söker p2. Avstånd till rutnät i x och y led kan lätt bestämmas ty rutnätets dimensioner är kända. Detta ger dy och dx som är kateter i två trianglar av olika storlek. Den minsta av dessa, med kortaste hypotenusa, ger oss p2. För senare beräkningar är det också viktigt att spara information om var segmentet skär cellväggarna, sida tak eller golv.

Ett special fall kan inträffa när längden på hypotenusan är lika, bild 1 till höger. Detta är ett typiskt exempel på ett gränsvärde om inte utgör något realt problem men som lätt missas vid implementation. Lösningen är att konsekvent låta segmentet skära sidan eller taket på den aktuella cellen.

När samtliga segment har skrivits in i rutnätet så polygonen uppdelad cell för cell. Tre typer av celler finns, inre yttre och inhägnad. Djupare information behövs endast om de celler som innehåller en del av inhägnaden. Det lönar sig att använda dubbel bokföring, en lista med typ av cell och en med inhägnad.

Att plocka fram en cell för given position kan nu göras men för att bestämma om positionen är inne eller ute så behövs en referens. Hörnen i cellen är väl definierade, se bild 2. Hela den yttre ramen med alla dess hörn är ute. Vid a så skär polygon linje 1 i y led så hörnet 11 är inne. Hörnen växlar mellan att vara inne och ute vid b. Hela rutnätet avläses linje för linje.

En cell kan ha flera oberoende linjer som i c. Det betyder att polygonen här skär y linjen två gånger och hörn 43 är inne. En paritets bit eller motsvarande behövs alltså för varje cell för räkna antal skärningar.

Med hörnen som referens så är det möjligt att bestämma om en given position är inne eller ute. En rätt linje dras från den aktuella positionen mot önskat hörn. Antal gånger som linjen skär polygonen samt om hörnet är inne eller ute bestämmer om den givna positionen är innanför eller utanför inhägnaden. Studera bild 2 för att bekräfta detta.

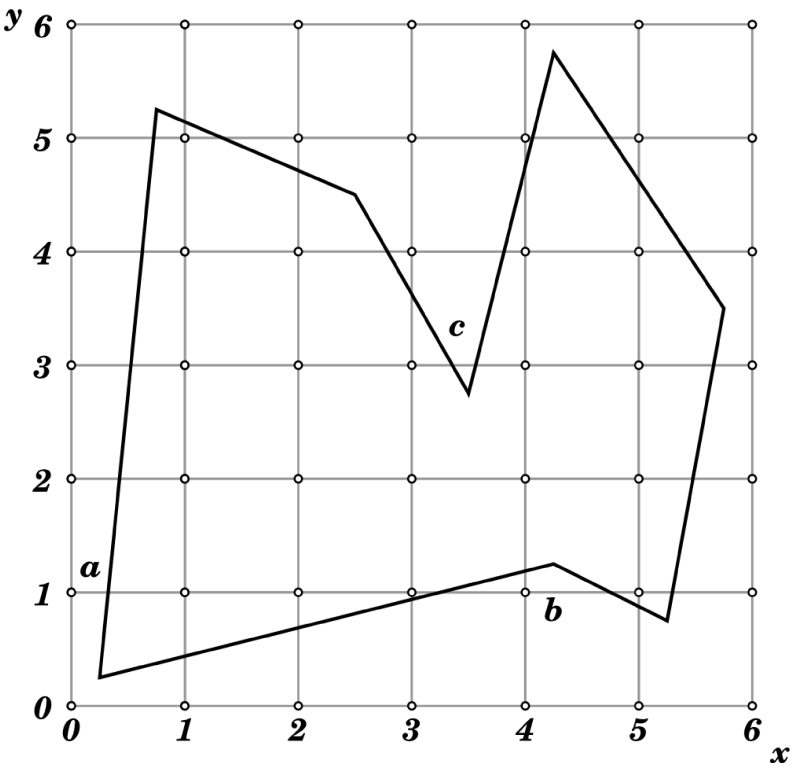

Bild 2: Polygonen inskriven i rutnätet.

Rutnätets dimensioner bestäms av inhägnaden och önskad upplösning. Min och max värde för inhägnaden bestäms först och från detta kan önskade dimensioner för rutnätet beräknas. Upplösning kan vara men behöver inte vara antal rutor i x och y led. Har vi till exempel flera områden så hängs de upp i ett gemensamt rutnät. En annan definition av upplösning kan då vara lämplig.

Att skriva in inhägnaden i rutnätet består alltså av följande delar:

- Beräkna rutnätets dimensioner.
- Att skriva in inhägnaden ruta för ruta.
- Att bestämma om rutnätets hörn är inne eller ut.

3.4.3 Odefinierat område

Inhägnaden kan inte upprättas om det finns odefinierade områden orsakade av korsande linjer. Att bestämma om två linjer korsar varandra kan göras genom att kontrollera varje segment mot alla andra segment. Detta är en uppenbart ineffektiv metod vars tidsåtgång är kvadratisk till antalet koordinater. Det finns mer effektiva lösningar som ger O(n log(n)) men dessa för den här uppgiften orimligt komplicerade. Sänks kravet och lokalt odefinierade områden inom en ruta accepteras så kan kontroll göras mycket snabbt, i storleksordningen O(n) där n är antal rutor med inhägnad.

Med ett lokalt odefinierat område menas att ett eller flera delpolynom överlappar varandra så att ett slutet område uppstår inom en ruta, se bild 3. Lokalt odefinierade områden kan accepteras då dessa inte påverka förutsättningarna för upprättandet av inhägnaden och användandet av denna i sin helhet. Korsar två delpolynom varandra så faller grunden för rutnätet då in och utsida ej kan fastställas.

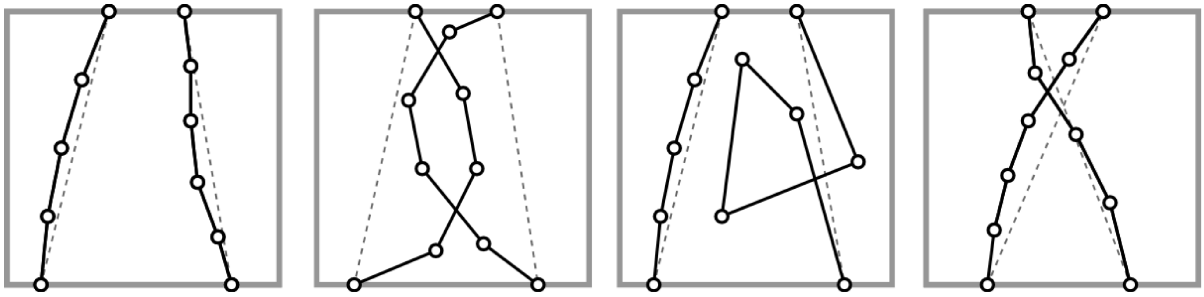

Bild 3: Lokalt odefinierade områden och korsande linjer

Kontroll görs genom att varje delpolynom ersätts med en rät linje från ingång till utgång, streckad linje i bild 3. Korsar dessa linjer varandra så har vi en ogiltigt inhägnad. Problemet med lokalt odefinierade områden reduceras med ett allt mer finmaskigt rutnät. En lämplig tumregel kan vara att att det totala antalet rutor ska vara av samma storleksordning som antalet koordinater.

3.5 Implementation på Android

3.5.1 Demo applikation

Avsikten med applikationen är att testa den virtuella inhägnaden på en faktisk mobil enhet. Applikationen är inte avsedd för vanliga användare. De funktioner som behövs:

- Att starta och stänga ned applikationen
- Att välja område.
- Att aktivera den virtuella inhägnaden.
- Att informera om enheten lämnar inhägnaden.
- Att visa information som kan vara av intresse för utveckling och felsökning.

Ett antal områden är bestämda på förhand och kan varken skapas eller på något sätt modifieras. De har skiljer sig åt genom områdets storlek, position och antal koordinater. Syftet med de olika områdena är dels att testa algoritmen och dels att felsöka koden.

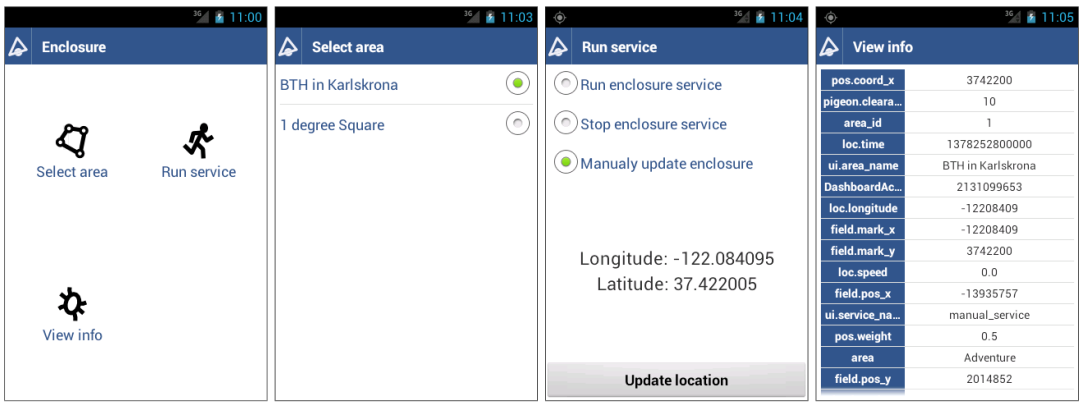

Bild 4: Demo app i Android

Användaren informeras med ett textmeddelande om enheten lämnar den virtuella inhägnaden. Ett nytt meddelande visas om enheten återvänder till inhägnaden. En enhet kan vara innanför, utanför och vid inhägnaden. Varje gång enheten växlar tillstånd så visas ett meddelande.

Information relevant för utveckling av koden är olika interna tillstånd, position och status för inhägnaden.

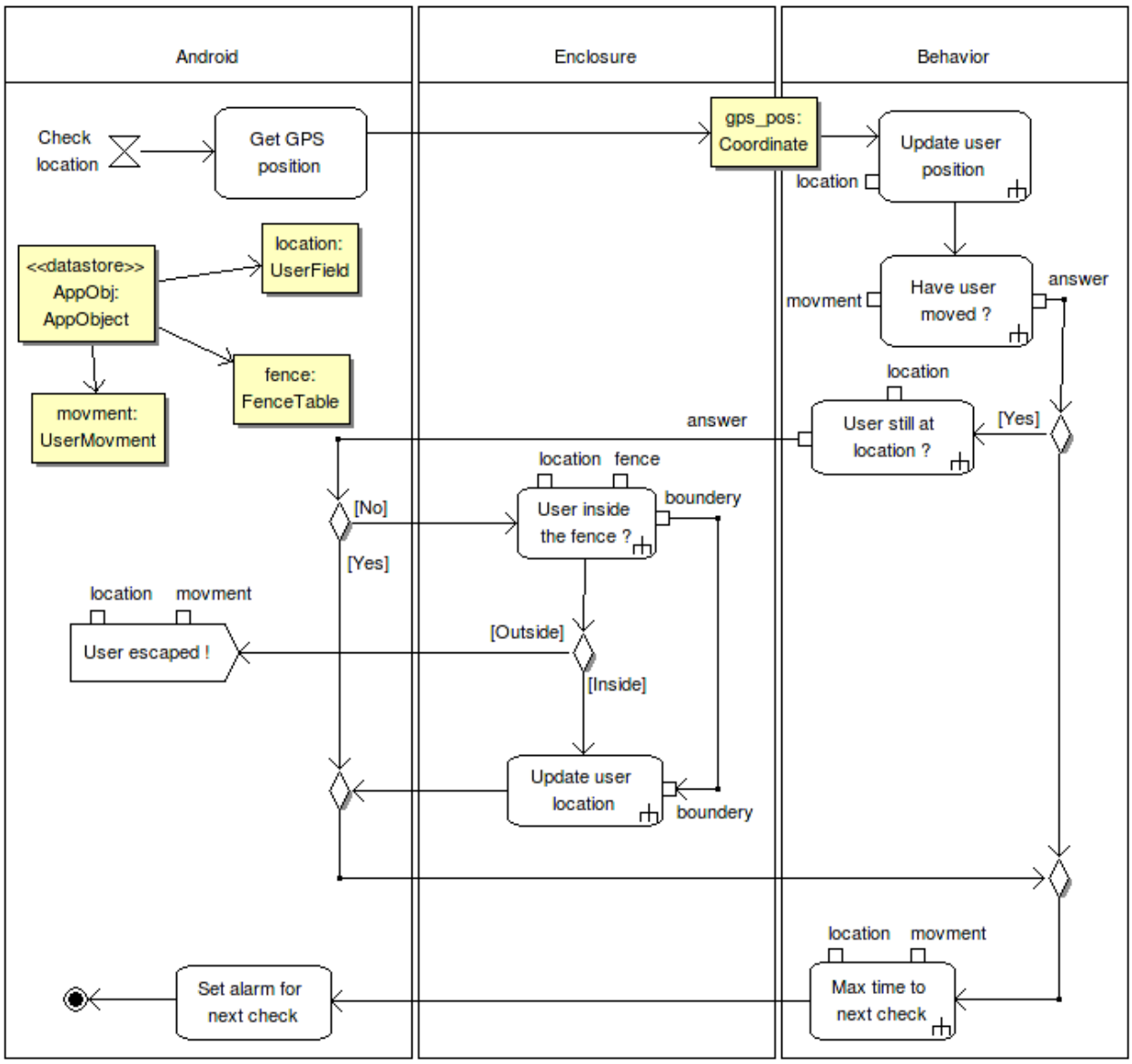

Bild 5: Aktivitetsdiagram, detta diagram är ej längre aktuellt och kommer att uppdateras.

3.5.2 Uppdatering av position

Detta är en service som aktiveras med bestämda tidsintervall. Bild 4 visar ett övergripande aktivitetsdiagram. En service i Android är ett byggnadsblock för hantera bakgrundsuppgifter[4]. Aktivitetsdiagramet visar de viktigaste delarna, `Android`, `Trek` och `Enclosure`. Inhägnaden hanteras av `Enclosure` medan `Trek` bestämmer nuvarande position och behovet av positionering. För att bestämma om enheten är innanför inhägnaden behöver följande uppgifter utföras:

- Hämta in senaste position, var användaren befinner sig just nu.
- Uppdatera användares position och kontrollera om användaren förflyttat sig.
- Kontrollera om användaren befinner sig innanför inhägnaden.
- Bestämma tidpunkt för nästa positionering.
- Informera användaren.

Att bestämma aktuell position görs med hjälp `LocationManager`. Här har jag utgått från Reto Meiers metod för positionsbestämning[5]. Det som särskiljer hans hans metod för att bestämma aktuell position är han först gör en passiv positionering. Det spara energi om det finns andra applikationer som också använder positionering. Uppfyller senast kända position inte vissa kriterier så begärs en ny positionering.

Med en ny position så kan enhetens position uppdateras vilket sker i `Trek`. Enheten representeras av klassen `Pigeon`, ett namn valt i bristen på ett bättre. Klassen `Pigeon` har en position, rör sig med viss hastig och befinner sig i ett bestämt område. Det är `Pigeon` som länkar samma Android och inhägnaden. Lämnar `Pigeon` sitt utstakade område så informera `Pigeon` om detta till eventuella lyssnare.

Befinner sig enheten, `Pigeon`, på eller i närheten av en viss position under en längre tid så anses användaren vara inaktiv. Någon kontroll om enheten befinner sig inom sitt område behöver bara göras om användaren är aktiv. Detta ger systemet en justerbar grad av tröghet vid positionering. Det område som `Pigeon` befinner sig inom är ett delområde av hela inhägnaden. Ett nytt delområde skapas vid behov av `Enclosure`.

För att bestämma tidpunkt för nästa positionering så behövs enhetens rörelsehastighet över en viss sträcka. Denna information tillhandahålls av `Pigeon`. Den tid det tar med nuvarande rörelsehastighet att nå gränsen för utstakat område är maximalt tidsintervall för nästa positionering. Att schema lägga tidsbestämda händelser görs med `AlarmManager` och beskrivs av Marko Gargenta i hans *Android Bootcamp Serie*[6].

4 *Android Developers (u.å.) Services [Internet] Från: developer.android.com/guide/components/services.html [Hämtad: 4 Sep 2013]*

5 *Meier Reto (23 juni 2011) Android Developers Blog, A Deep Dive Into Location [Blog] Från: android-developers.blogspot.se/2011/06/deep-dive-into-location.html [Hämtad: 4 Sep 2013]*

6 *Gargenta, Marko [NewCircle] (7 mars 2012) 25 - Alarms and System Services: Android Bootcamp Series 2012 [Föreläsning] Från: www.youtube.com/watch?v=QEMk4SwsJMg&list=SPE08A97D36D5A255F [Hämtad: 4 Sep 2013]*

3.5.3 Positionering och koordinater

Vid implementation används råa koordinater, det vill säga grader. Ingen omvandling sker från grader till ett plant koordinatsystem. Detta påverka inte inhägnaden och dess funktion. Koordinater för att faktiskt område som till exempel BTH har hämtats från hitta.se[7]. Samtliga områden är samlade i en statisk klass `Route`. Detta gör att nya områden lätt kan läggas till vid behov.

3.5.4 Inhägnad

Inhägnaden upprättas och upprätthålls av `Enclosure`. För att upprätta det rutnät som utgör grunden i algoritmen så behövs ett plan. Detta plan, basklassen `EuclidPlane`, är grunden i inhägnadens koordinatsystem. Andra klasser som använder koordinatsystem är `Boundary` och `Segment`. Klassen `Segment` är central för att bestämma om en given koordinat är innanför eller utanför inhägnaden.

Upprättande av själva inhägnaden sker i klassen `Enclosure` och det är denna klass som också administrera inhägnaden. Dess huvudfunktion är att skapa de delområde som används av `Pigeon`. Algoritmen för att upprätta inhägnaden som beskrivits i kapitel 3.4 är implementerad i `Enclosure`.

Två tabeller används vara spara information om inhägnaden. En tabell `TileTable` används för att spara data om de celler som innehåller en eller flera delar av inhägnaden. En cell som är tom sparas inte. Varje del av inhägnaden sparas i klassen `SectionTable`. Tre klasser behövs för att spara inhägnaden, `Grid`, `TileTable` och `SectionTable`. Det tillkommer också två gränssnitt `Tile` och `Section` som används för att administrera tabellerna.

3.6 Utvärdering

3.6.1 Komplexitet

Vid implementation så framkom det att den valda algoritmen var betydligt mer komplex än väntat. Totalt så handlar det om mer 1000 rader kod för `Enclosure`. Samtidigt så råder det osäkerhet om hur stort området måste vara för att dra nytta av den rutnätsbaserade algoritm som här har användes för upprätta och upprätthålla inhägnaden. Frågan som har uppstått är om den valda algoritmens komplexitet är rimlig i förhållande till den uppgift den löser. Detta ändra förutsättningarna för utvärdering av algoritmen. Avsikten var att testa på algoritmen på en eller flera enheter. Men om algoritmen är mer komplex än vad som behövs för att lösa uppgiften så är det bättre att utvärdera projektet i sitt nuvarande tillstånd.

7 *hitta.se (u.å.) Karta, vägkarta, satellitbilder och gatubilder [Karta] Från: www.hitta.se/karta [Hämtad: 4 Sep 2013]*

3.6.2 Vad är implementerat

All primär kod är implementerad. Detta innebär att applikationen kan starta och att all funktionalitet är implementerad. Inhägnaden upprättas och upprätthålls. Applikationen är funktionell men inte användbar. Den är allt för instabil för att installeras på en mobil enhet.

Fokus har legat på de två delarna `Enlosure` och `Android`. Att fokus kom att delas mellan `Enclosure` och `Android` beror på att undertecknad har begränsad erfarenhet av ekosystemet Android och hade inte tidigare arbetat med `LocationManager`. Den tredje delen `Trek` som hantera enhetens rörelse och position har implementerats så enkelt som möjligt.

Det som saknas är undantagshantering, *exceptions*, och systematisk testning med JUnit[8]. Löpande under utveckling har valda delar av koden testats i Python. Python är användbart för validering av algoritmer eller beräkningar. Det kan finnas andra mindre uppenbara problem som minnes läckage.

Totalt sätt så handlar det om mycket arbete och därför har fortsatt utveckling har avstannat till förmån för en analys av den valda algoritmen i dess nuvarande tillstånd.

3.6.3 Minsta storlek för den geografiska inhägnaden

Det svårt att bedöma rimlig storlek på geografiska inhägnaden då detta i så hög beror på vilken hårdvara som används. En människa till fots rör sig med en hastighet av 1 m/s, de flesta fortare än så men få rör sig fortare än 10 m/s. En mobilenhet kan positionera i storleksordning en gång vart 10:de sekund. Det gissning baserad på att det finns applikationer för bilnavigering.

Upplösningen blir då 10 meter men till skillnad från bilnavigering så saknas extern strömförsörjning. För en service så rekommenderas ett minsta tidsintervall på 5 minuter från en positionering till nästa[9]. Detta skulle ge en upplösning på 300 meter. Som jämförelse så är täcker BTH i Karlskrona en yta på 300 x 800 meter.

Det rimligt att gränsen ligger vid en upplösning på 100 meter. Detta är den minsta användbara storleken på en ruta i rutnätet. Gränsen sätts praktiken av energi kostnad, hur stor batterikapacitet som finns tillgängligt. Därefter så sätts gränsen av hur fort systemet kan uppdatera aktuell position och med vilken precision en position kan bestämmas.

3.6.4 Skalbarhet

Inhägnaden administreras av ett konstant antal klasser oberoende av antalet koordinater. Detsamma gäller också också vid upprättande av inhägnaden. Merparten av all data sparas i Javas `HashMap`. Att upprätta ett delområde är ett enkelt uppslag i en tabell. Detta gör att behovet av minne ökar linjärt med antalet koordinater. I sitt nuvarande tillstånd så använder demo applikationen 6 Mbyte minne totalt, merparten för de klasser som utgör applikationen.

8 *JUnit (u.å.) [Hemsida] Från: junit.org [Hämtad: 5 Sep 2013]*
9 *Android Developers (u.å.) LocationManager [Internet] Från: developer.android.com/reference/android/location/LocationManager.html#requestLocationUpdates(long, float, android.location.Criteria, android.app.PendingIntent) [Hämtad: 5 Sep 2013]*

Då inhägnaden upprättas så skapas nya koordinater i övergången från en cell till en annan. Det faktiska antalet koordinater blir därför större den ursprungliga mängden koordinater. Hur mycket större beror på rutnätets upplösning. Det blir två nya koordinater för varje cell som innehåller en del av inhägnaden. Detta problem kan minskas med balanserad avvägning mellan antalet koordinater och rutnätets upplösning.

Det är svårt att bestämma kortaste avstånd till inhägnad från en godtycklig punkt. Detta avstånd är av intresse vid skapandet av delområde då det är önskvärt att delområdet är stort som möjligt. Ett delområde är i den nuvarande implementation en cell i rutnätet. Svårigheten i att bestämma avstånd ligger i att flera celler korsas. Varje operation som involvera flera celler kräver betydligt mer arbete. Detta är en begränsning för mindre inhägnader då varje ruta blir liten. För större inhägnader så är detta ett mindre problem.

3.6.5 Förenkling av algoritmen

Fördelen med en rutnätsbaserad algoritm är just rutnätet. Det gör det enkelt att skapa flera områden. Det går snabbt att bestämma om enheten är inne eller utanför. För större områden, flera mil, så är den valda algoritmen lämplig. För mindre områden så är den orimligt komplex.

En variant på den valda algoritmen är att endast notera de celler som korsas av inhägnaden. De betraktas som innanför. Önskad upplösning styrs med rutnätets upplösning. Området BTH kan till exempel delas upp i 10 x 20 rutor. Varje ruta är då 30 x 40 meter.

En fördel med denna förenklade algoritm är att den kan användas sida vid sida med den mer komplexa. De komplettera varandra.

4 Diskussion

4.1 Diskussion

Skriv mig:

Vad fungerade inte och vilka andra problem finns det.

4.2 Framtida utveckling

Skriv mig:

Tankar och idéer kring framtida utveckling av systemet.

5 Slutsatser

5.1 Inhägnad av enhet

6 Referenser