Universitat de Girona
Escola Politècnica Superior

Treball final de Màster

Estudi: Màster en Enginyeria Informàtica

Títol: Localització de nous hotels segons les regions d'influència dels punts d'interès d'una regió

Document: Memòria

Alumne: Jordi Bagot Soler

Director/Tutor: Dra. Marta Fort i Masdevall
Departament: IMAE
Àrea: Llenguatges i sistemes informàtics

Convocatòria (mes/any): 09/2016

Índex

Capítol 1

Introducció

1 Introducció i objectius del projecte

1.1 Motivació

S'ha fet aquest projecte degut a la constant necessitat d'expansió d'algunes empreses de serveis, com les cadenes hoteleres, de restauració o tèxtils. Aquestes empreses solen comprar o construir noves seus amb regularitat. Però per fer-ho necessiten fer molts procediments manuals que costen molt temps i diners. Aquests procediments consten d'estudis de viabilitat i d'estudis de la competència. Necessiten varies persones per poder estudiar totes les seus que podrien ser competència. A més a més, per saber si una seu serà viable o no han de saber si la zona és turística i si tindrà suficients clients per poder tenir beneficis. Així doncs, han d'estudiar varies zones d'una ciutat o regió per saber quina és la millor i quina els hi donaria més beneficis. Les empreses més importants tenen departaments sencers d'expansió que realitzen aquestes tasques. Les distàncies són per la xarxa de carreteres, això complica encara més aquests procediments. En el projecte s'utilitza el mapa de carreteres per calcular les distàncies.

Per aquest motiu s'ha volgut desenvolupar una eina que els hi proporcioni ajuda en aquest procés tant tediós. Aquest projecte de final de màster pot ser el principi d'un projecte molt més gran i elaborat que sigui de gran utilitat per aquest tipus d'empreses.

Aquesta eina pot tenir un gran potencial si té acceptació en el sector, caldria millorar-la i adaptar-la a les necessitats de cadascú parametritzant valors i així assolir l'objectiu final que seria poder-lo vendre.

Hi ha una empresa hotelera que el farà servir i així podrà ser provat en el món real.

1.2 Objectiu del projecte

L'objectiu d'aquest projecte és desenvolupar una eina on donat un mapa d'una ciutat i les seus existents amb un valor d'influència associat, indicant el seu poder d'atracció o la seva importància, mostri la viabilitat de construir una nova seu en cada punt del mapa. Les seus existents que hi haurà en el mapa seran de dos tipus: unes que aporten benefici a la nova seu i que per tant són seus amb influència positiva, com podria ser una catedral, una zona de compres, una zona de negocis. I també hi pot haver seus que treuen benefici, és a dir, amb influència negativa que bàsicament serien seus semblants a la que es vol construir, en el cas de voler construir un hotel, la seva competència seran hotels de característiques semblants.

Es vol obtenir un mapa d'influències de les possibles noves seus tenint en compte les ja existents i els pesos tant de la nova com de les existents. La influència d'una seu es propaga en la xarxa de carreteres de la ciutat i disminuirà de forma proporcional amb la distància a la seu, a més distància de la seu, menys influència, en valor absolut. Es vol un model que calculi distàncies de forma realista, és a dir que calculi la distància real entre dues seus, aquesta vindrà donada pel camí més curt d'una seu a una altra en una xarxa de carreteres. Tot i que aquesta és la idea, ens cal una bona formalització del problema, per tal de resoldre'l de forma ràpida, eficient i robusta, integrada en una eina que ha de permetre amb un simple cop d'ull saber quines són les millors zones de la ciutat per construir una nova seu o les zones on no s'ha de construir.

A més més, aquest projecte té un altre objectiu menys tècnic que és ajudar al departament d'expansió de cada empresa fent que la seva feina no sigui tant manual i repetitiva. Això pot proporcionar un avantatge sobre la competència perquè un algoritme pot trobar zones en les quals els humans no han pensat abans i això en aquests sectors pot ser molt important degut a la competència ferotge que tenen aquestes empreses. Qualsevol petita innovació pot proporcionar molts beneficis i aquest projecte podria ser una d'aquestes.

I l'objectiu final és que esdevingui un producte de necessitat que proporcioni una gran ajuda a la presa de decisions de l'empresa. Un cop estigui estabilitzat i funcioni correctament l'objectiu més ambiciós és vendre'l a qualsevol empresa que hagi de construir una seu i depengui de la influència de la competència o dels visitants.

1.3 Equip de treball

Aquest projecte s'ha fet de manera individual amb l'ajuda de la tutora Dra. Marta Fort. Inicialment es va parlar amb el departament d'expansió de l'empresa hotelera esmentada, per tal de poder definir un model teòric de càlcul d'influències que s'adaptés a les seves experiències i manera de fer actual.

1.4 Planificació

Per assolir els objectius esmentats anteriorment s'ha definit una planificació que s'ha dividit en quatre fases diferents:

- **Fase 1: Estudi previ:** Aquesta primera fase serà d'investigació, on es decidiran quines tecnologies s'utilitzaran per poder obtenir els millors resultats possibles ja sigui amb la implementació d'algoritmes, com en la visualització o en la implantació.
- **Fase 2: Anàlisi i disseny:** S'estructuraran les diferents parts del projecte i es decidirà com es poden enfocar cada una d'aquestes parts. Aquest disseny haurà de donar solució als objectius plantejats.
- **Fase 3: Implementació i proves:** Aquesta fase serà on es desenvoluparan les solucions dissenyades a la fase 2 amb la tecnologia decidida a la fase 1.
- **Fase 4: Documentació:** Finalment s'escriurà una documentació on es mostri el procés i l'esforç que hi ha hagut en el desenvolupament d'aquest projecte.

Cal dir que aquest projecte serà una primera versió, un començament. Els requisits en part s'han obtingut del departament d'expansió però uns altres són d'elaboració pròpia. En aquest projecte s'ha volgut fer una mostra del que realment podrà ser i per fer veure del que la geometria computacional és capaç de fer per l'empresa i la importància de donar-hi suport. S'han utilitzat mapes de prova i s'han classificat les seus d'una manera coherent però sense supervisió d'un personal professional en aquest tema. S'està parlant d'un projecte que es posarà en pràctica a l'entorn laboral i que a partir d'allà es millorarà perquè sigui més autònom i perquè mostri uns resultats més acurats o més adaptats a cada empresa.

Capítol 2

Estudi de viabilitat

2 Estudi de viabilitat

Per poder determinar la viabilitat del projecte hem d'analitzar el coneixement previ necessari alhora de desenvolupar, la infraestructura i el pressupost. A continuació els detallem.

2.1 Coneixement

Per poder començar a dissenyar i preparar aquest projecte cal tenir uns coneixements previs sobre les empreses de serveis, la manera com s'expandeixen i els procediments que utilitzen per fer-ho.

Aquest coneixement està adquirit gràcies a l'entorn laboral tot i que s'ha de profunditzar en els procediments concrets d'expansió per entendre com es fan i fins on pot arribar aquest projecte i on ha d'entrar el factor humà. Aquests coneixements s'adquiriran preguntant al departament d'expansió de l'empresa quins procediments utilitzen amb quins paràmetres i com els posen en pràctica. L'eina ha de proporcionar una ajuda per a aquest departament i s'ha de adequar a les seves necessitats. Aquesta recopilació de coneixements definiran els requisits funcionals d'aquesta aplicació.

A part, també calen uns coneixements tècnics per poder assolir els objectius, aquests coneixements s'han d'adquirir però no són requisits funcionals tot hi que sense ells no es podrien assolir els requisits funcionals:

1. **Base matemàtica:** Aquesta base matemàtica s'ha obtingut d'assignatures cursades al grau o al màster, de la tutora i d'Internet. A l'apartat 5.2.Execució de l'aplicació i algoritmes s'explicarà en detall quins són aquests coneixements.
2. **Manipulació de grafs:** El projecte es basa en la manipulació de grafs ja que un mapa de carreteres està representat per un graf. Aquest coneixement s'obtindrà de l'assignatura Matemàtica Discreta del grau en enginyeria informàtica i d'Internet.
3. **Geometria computacional:** És un projecte de geometria computacional i vol solucionar un problema d'aquest àmbit així que calen uns bons coneixements. Concretament en *facility location*. Aquests coneixements s'han après a l'assignatura Processament de dades espaials del màster i també de la tutora i d'Internet.
4. **Llenguatges de programació i visualització:** A la primera fase del projecte es decidiran quines tecnologies s'utilitzaran per desenvolupar aquest projecte i com es visualitzaran les dades. Hi ha moltes possibilitats ja que no hi ha limitacions pels requisits funcionals i no funcionals, es podran explorar noves tecnologies que prèviament no s'han explorat.

Així doncs, cal dir que els reptes que es proposen en aquest apartat són perfectament assequibles.

2.2 Infraestructura

Per desenvolupar el projecte s'utilitzarà les eines pròpies ja que al ser un projecte

individual no cal compartir el codi ni crear zones comunes. S'utilitzarà un ordinador i el programari necessari per poder desenvolupar aquest projecte. El programari utilitzat no és lliure però gràcies a la universitat es poden utilitzar llicències acadèmiques precisament per poder desenvolupar projectes com aquest. També s'utilitzarà un repositori que estarà al núvol. Tot i que el projecte sigui individual, aquest repositori permetrà tenir un control de versions del codi i treballar en varies branques. Això serà interessant per tal de no perdre codi i de tenir guardats tots els canvis que s'hagin fet durant tot el procés de desenvolupament. A més a més hi ha un *log* que serà molt útil per saber quan s'han fet els canvis. Com és lògic també ens servirà de *backup* així no hi haurà perill de perdre la feina feta. Finalment, també ens permetrà treballar en diferents entorns si cal fer-ho.

Per poder implantar i provar l'aplicació no ens cal cap servidor ja que amb el maquinari propi és suficient.

Per tant, en aquest aspecte no hi haurà cap problema de viabilitat ja que ja es disposa de tot el que es necessita.

2.3 Pressupost

En aquest projecte no ens cal pressupost ja que tot el que es necessita per desenvolupar-lo és propi i no cal invertir diners amb cap programari privatiu, ni cap maquinari. Totes les màquines que s'utilitzaran seran Linux així que no hi ha cap cost addicional en aquest aspecte.

Per tant, en aquest últim aspecte tampoc hi ha cap problema de viabilitat, així doncs es pot donar per viable aquest projecte.

Capítol 3

Metodologia

3 Metodologia

Per desenvolupar una aplicació informàtica es poden utilitzar varies metodologies, les quals proporcionen una serie de pautes que al seguir-les fan que el producte resultant sigui millor. Les metodologies en el desenvolupament d'aplicacions no és nou, fa molts anys que existeixen però últimament s'estan posant molt de moda les metodologies àgils. Prèviament hi havia les metodologies de cascada, V o en espiral, tot i que encara hi ha empreses que les utilitzen, la majoria utilitzen metodologies àgils.

3.1 Què és una metodologia àgil?

Les principals característiques de les metodologies àgils són:

- Comunicació i importància entre les persones.
- Programari funcional.
- Comunicació continua amb el client.
- Resposta ràpida al canvi de requisits.

Aquestes quatre característiques i els dotze principis estan escrits en el manifest àgil que es va escriure al 2001 [1].

Així doncs, el terme àgil diu que el més important és la comunicació, les persones i la predisposició al canvi. Això és un gran canvi respecte les metodologies anteriors perquè abans si es canviava un requisit tenia un cost molt gran en temps de desenvolupament perquè hi havia un protocol que no permetia fer-ho amb rapidesa.

El que s'intenta amb aquest tipus de metodologies és que el client pugui veure el que s'està desenvolupant contínuament, això fa que el client pugui veure si realment el que s'està fent és el que ell realment vol. S'estableixen uns períodes anomenats *sprints* que solen ser de dos setmanes i al finalitzar cada *sprint* s'ensenya al client el que s'ha fet i llavors és quan el client decideix fer canvis de requisits o no. Això és molt positiu perquè al final del desenvolupament el client tindrà el producte que ell ha volgut. En canvi, amb altres metodologies el que passa és que el client dona els requisits i al cap d'un any veu el producte acabat i no s'assembla en res amb el que ell havia pensat i el que és pitjor, no soluciona els seus problemes.

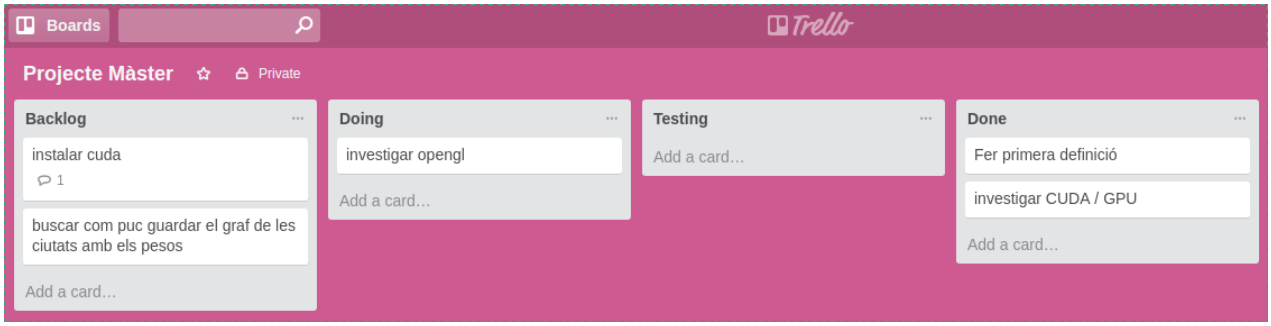
3.2 Perquè s'ha escollit una metodologia àgil?

En aquest projecte s'ha escollit una metodologia àgil per les entregues periòdiques. En el cas d'aquest projecte el client són les empreses a les quals se'ls hi entregarà o vendrà el producte en un futur, especialment a l'empresa hotelera que està disposada a provar-la. Però en aquest àmbit acadèmic es simularà que el client és la tutora ja que és qui controlarà la feina periòdicament i qui decidirà si les funcionalitats es corresponen a les del disseny o no.

Una de les parts més importants de les metodologies àgils és la comunicació entre l'equip, però aquest projecte és individual, per tant, no hi pot haver-hi comunicació. Tot hi que s'ha creat una pissarra de tasques o històries d'usuari (és el nom que s'utilitza a

SCRUM) les quals han sigut introduïdes per un mateix (Il·lustració 1). Tot i que això no ha de ser així, això és un entorn acadèmic i s'ha simulat d'aquesta manera. La utilització de pissarres en les metodologies àgils és una pràctica molt estesa. S'utilitzen pissarres virtuals (com és el cas d'aquest projecte) o pissarres físiques (en cas que l'equip treballi en un lloc físic concret). En aquestes pissarres hi ha varies columnes que es poden modificar però les més comunes són les següents:

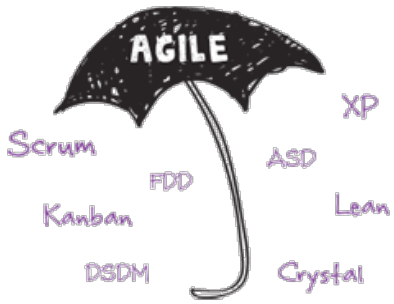
- **Product Backlog:** Conté totes les tasques que s'han de fer i si n'hi ha de noves s'afegiran aquí. Aquesta columna ha d'estar ordenada per prioritat.
- **Backlog:** Aquesta columna hi haurà les tasques que es faran en el període actual (*sprint*). Les tasques s'agafen de la columna *Product Backlog* i es col·loquen aquí. Això es fa al principi de cada *spint*. Sempre s'agafen les tasques més prioritàries.
- **Doing:** Aquí hi ha les tasques que s'estan fent per l'equip actualment.
- **Testing:** Aquí hi ha les tasques que s'han acabat de desenvolupar però que encara s'han de testejar.
- **Done:** Aquí hi ha les tasques finalitzades.



Il·lustració 1: Pissarra de tasques o històries d'usuari

A més a més, l'elecció d'una metodologia àgil ha estat encertada ja que els requisits han variat mentre es feia el desenvolupament. Al capítol 6.Requisits del sistema es parlarà dels requisits del projecte.

3.3 Quins tipus de metodologies àgils existeixen?



Il·lustració 2: Metodologies àgils

Hi ha vàries metodologies que es consideren àgils perquè compleixen el manifest, però

tenen algunes diferències. Ara cal escollir quina és la més adequada per aquest projecte [2].

Ha de quedar clar que no n'hi ha una de millor ni una de pitjor, simplement s'adapten millor o pitjor a un tipus de projectes o d'equips/empreses.

Aquí es definiran tres metodologies, les que s'utilitzen més. Tot hi que n'hi ha moltes més.

3.3.1 Kanban

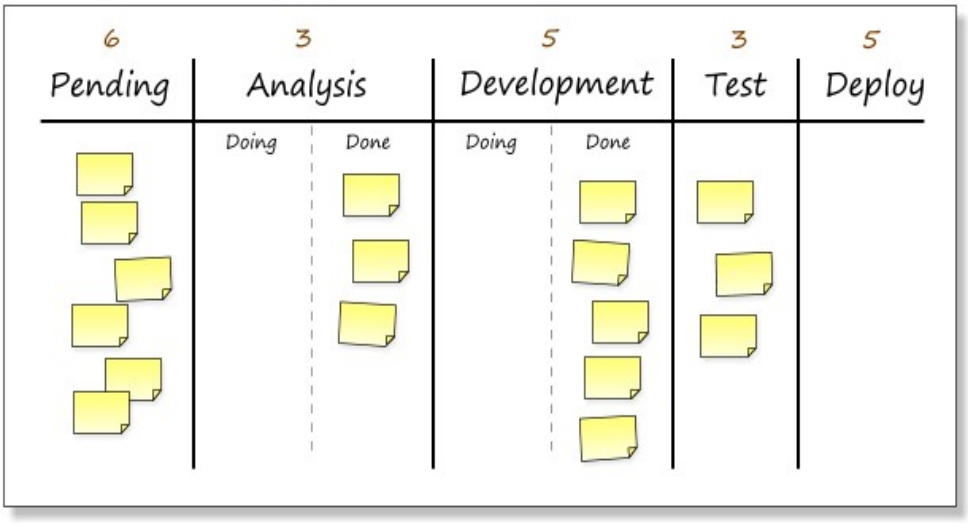
[3] Aquesta va ser una de les primeres que hi va haver. I està inspirada en la metodologia que utilitzava Toyota anomenada *just-in-time*.

Aquesta metodologia és la més senzilla. Simplement ens entren noves tasques que es posen a la primera columna de la pissarra i els membres de l'equip les agafen en ordre, es podria fer una similitud amb una cua *FIFO* (Il·lustració 3), quan un membre de l'equip l'agafa, s'analitza, es desenvolupa, es prova i finalment es puja a producció.

Il·lustració 3: Flux de Kanban

Una pissarra *Kanban* sol tenir les columnes:

- Pending
- Analysis
- Doing
- Testing
- Done



Il·lustració 4: Pissarra de Kanban

Només té una limitació o requisit que s'ha de complir, el *WIP*. Són les sigles de *work-in-time* i és un número que es posa a sobre de cada columna de la pissarra i indica quantes tasques hi pot haver-hi com a màxim a cada columna. Això es fa per no crear colls d'ampolla. Els *WIPs* els decideix l'equip.

3.3.2 SCRUM

[4] És la més utilitzada amb gran diferència. És una metodologia més complerta que *Kanban*. Està molt enfocada a la gestió d'equips.

Hi ha una sèrie de rols que ha d'ocupar cada membre de l'equip:

- **Product Owner (PO):** És la persona que s'encarrega de crear les tasques a l'equip i les defineix amb gran precisió. També és qui ordena les tasques per prioritat. I és el que parla amb el client per saber el requisits i plasmar-los en tasques. Té una gran responsabilitat, ja que si les tasques (anomenades històries d'usuari (HU)) no estan ben especificades l'equip pot perdre velocitat. No està a dins de l'equip de desenvolupadors.
- **Scrum Master (SM):** És un desenvolupador de l'equip però a més a més s'encarrega de que l'equip compleixi amb les pautes de *SCRUM*. També té la tasca d'evitar les distraccions de l'equip, els ha de protegir. És com un capità.
- **Desenvolupadors:** és l'equip de desenvolupadors i ells han de desenvolupar les tasques que els hi ha assignat el *Product Owner*.

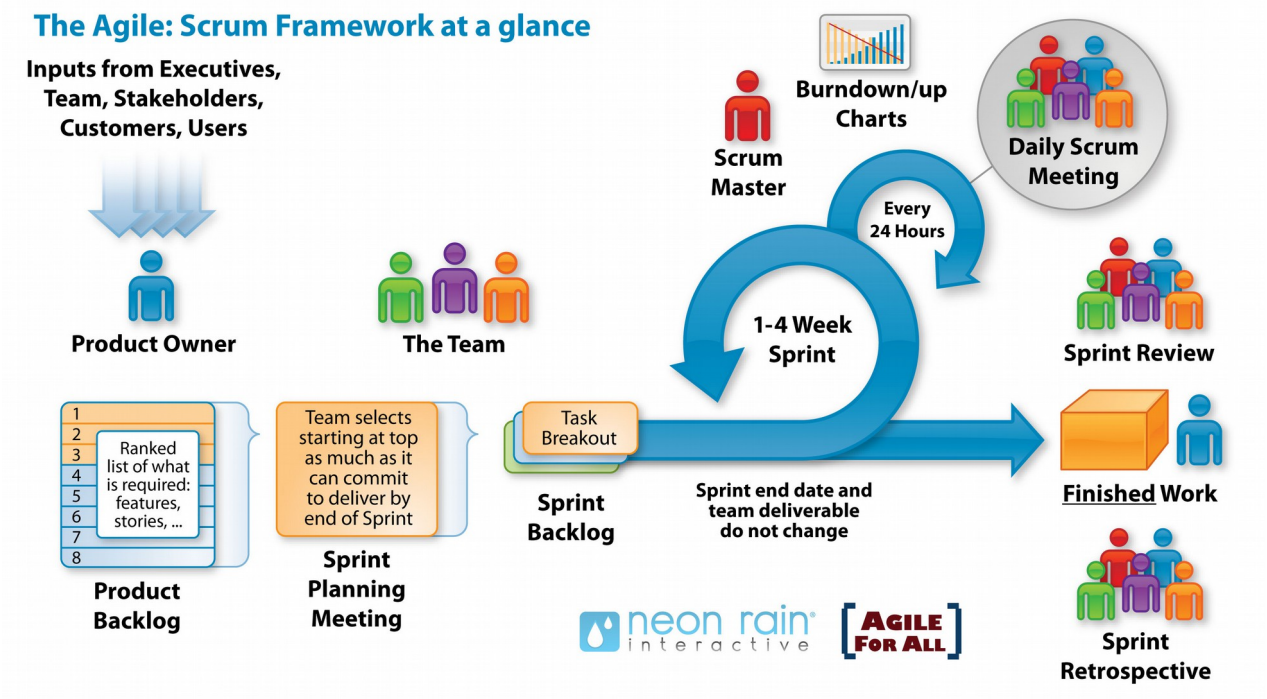
Hi ha varies pautes molt importants en *SCRUM*:

- Cada desenvolupador només pot desenvolupar una sola tasca a la vegada, l'ha d'agafar i no la pot deixar fins que l'acabi (a no ser que estigui mal definida i l'hagi de tornar al *Backlog*).
- Hi ha uns períodes anomenats *sprints*. A continuació s'explicaran.
- Hi ha una reunió diària que s'anomena *daily*. En aquesta reunió cada membre de

l'equip explica què va fer ahir i què farà avui. Si hi ha alguna cosa important a comunicar es pot comunicar, però ha de ser ràpid. Les *dailys* han de durar entre 5 i 15 minuts. El *SM* és qui s'encarrega de que això es compleixi.

Característiques dels *sprints*:

- Són períodes que solen ser de dos a quatre setmanes.
- A l'inici de l'*sprint* es fa una reunió anomenada *Sprint Planning* on es vota de manera democràtica entre l'equip de desenvolupadors l'esforç que suposa cada història d'usuari HU. Els desenvolupadors han de respectar aquest esforç i cenyir-se a ell. M'entre es van votant els esforços es van movent les tasques del *Product Backlog* (columna on el PO hi afegeix les HU prioritzades) al *Backlog* de *l'sprint* actual fins arribar al límit de l'esforç que l'equip pot assumir en un *sprint*. Una vegada s'ha fet, comença *l'sprint*.
- Al final de cada *sprint* s'ha de presentar una versió del producte funcional que es presenta en una reunió anomenada *Sprint Review* on cada membre de l'equip mostra el que ha fet en una demostració on hi haurà l'equip, el PO i el client (si vol).
- Al final de cada *sprint* es fa un reunió anomenada *Sprint Retrospective* on els membres de l'equip voten quines coses han funcionat bé a *l'sprint* que s'acaba i quines coses han anat malament. Les que volen que segueixin així, les que volen que disminueixin o augmentin. És una reunió per opinar i per millorar *sprint* a *sprint* l'equip.



Il·lustració 5: Flux de SCRUM

3.4 Quina ha estat l'escollida?

Una barreja. S'ha escollit una barreja entre *Kanban* i *SCRUM*. Hi ha llocs on l'anomenen *SCRUMBAN*.

En aquest projecte no es pot aplicar *SCRUM* ja que no hi ha un equip, però *Kanban* per si sol és poc detallat i per aquest motiu s'ha escollit una barreja.

La barreja consta de que cada setmana o cada quinze dies s'ha de presentar una versió funcional al client (tutora). Hi ha *sprints* i s'ordenen les tasques per prioritat. Però no hi ha els rols de *SCRUM* assignats perquè no hi ha equip.

S'ha utilitzat una pissarra com la de *Kanban* i s'han desenvolupat les tasques tal i com es fa a *Kanban* amb *WIP*.

SCRUMBAN no està definit, qualsevol pot escollir el que li sembli millor d'aquestes dos metodologies o el que s'adapti millor al projecte o equip. Per exemple, la gran majoria d'empreses comencen per fer la *daily* però no fan res més d'*agile*. Això simplement ja suposa una gran millora en un equip ja que millora molt la comunicació.

Capítol 4

Planificació

4 Planificació

Per poder fer aquest projecte s'ha fet una planificació que ens servirà per poder tenir objectius marcats en el temps i saber si s'està anant pel bon camí o pel contrari s'està anant massa lent per poder complir els terminis especificats.

En aquest cas s'ha fet una planificació amb quatre fases les quals s'han esmentat a l'apartat 1.4.Planificació i ara es detallaran.

4.1 Fase 1: Estudi previ

En aquesta primera fase es farà investigació. La tipologia d'aquest projecte dona llibertat per utilitzar varies tecnologies tot i que n'hi ha que són més adequades que d'altres. És molt important aquesta fase ja que sedimentarà el projecte i tot girarà al voltant de les decisions que es prenguin aquí. S'ha d'escollir bé i la millor elecció és fer un projecte atractiu, que suposi un repte però que a la vegada amb esforç es pugui assolir amb el temps estipulat.

A part de la tecnologia s'estructurarà el projecte i es faran reunions amb la tutora per poder compartir els avenços i per rebre respostes i recomanacions sobre les tecnologies o l'estructura del projecte.

Cal dir que aquest projecte suposarà un repte d'entrada, perquè part dels objectius marcats mai s'han estudiat en el grau o el màster, per tant seran reptes interessants i que farà que aquest projecte esdevingui acadèmic i d'on es pugui aprendre conceptes nous que es podran aplicar en el futur.

A part, també s'haurà de recopilar informació sobre el tema del projecte que és l'expansió de les empreses en la compra de noves seus. Per obtenir aquesta informació es faran reunions amb l'empresa que provarà el projecte quan aquest estigui acabat. En aquestes reunions es rebrà una formació que servirà de base per poder entendre el funcionament de les empreses en aquest aspecte i això donarà les idees i requisits funcionals per poder dissenyar el projecte.

4.2 Fase 2: Anàlisi i disseny

En aquesta fase es dissenyarà el projecte partint de la base que s'ha obtingut en les reunions prèvies i en la investigació de les tecnologies.

El disseny que es plantejarà ha de donar solucions a tots els requisits funcionals que s'han obtingut a les reunions amb l'empresa i a tots els requisits no funcionals que s'han parlat amb la tutora i amb els reptes proposats personalment per proporcionar un repte que aporti una motivació per poder fer aquest projecte i acabar-lo.

En el cas d'aquest projecte donarà lloc a una aplicació d'escriptori. Tindrà diverses parts però formaran un únic producte i hi haurà un disseny adequat a aquesta aplicació d'escriptori. Ara bé, la part important és la de l'anàlisi del problema que s'ha plantejat, aconseguir fer-ne una bona formalització i dissenyar una bona solució amb les eines esmentades.

4.3 Fase 3: Implementació i proves

En aquest punt del projecte es durà a terme el que s'ha dissenyat en l'anterior fase i es farà realitat l'aplicació i la implementació dels algoritmes. Aquesta serà la fase que més temps durarà i serà necessària una organització per tal de fer un codi net, estructurat i que compleixi amb els requisits. Per aconseguir aquests propòsits s'utilitzarà la metodologia *SCRUMBAN* que s'ha esmentat en el capítol anterior. Així doncs s'estructurarà per *sprints* i es faran reunions setmanals amb la tutora per posar en comú els avanços i possibles millores o solucions. Al ser una metodologia àgil es podran fer modificacions dels requisits en cas que hi hagi necessitat de fer-ho.

A més a més també s'utilitzarà un control de versions *git* per poder tenir el codi versionat i a més a més desat per prevenir les tant inoportunes pèrdues de codi. El *git* estarà en un repositori al núvol privat de *Bitbucket.* Això també farà més senzill accedir-hi des de qualsevol lloc i poder treballar des de diferents llocs.

4.4 Fase 4: Documentació

Aquesta fase és l'última i és on ha de quedar plasmat el resultat i l'esforç que hi ha hagut en el desenvolupament d'aquest projecte. La documentació servirà per explicar i ensenyar a terceres persones aquest projecte. També servirà pels nous desenvolupadors que puguin desenvolupar aquest projecte en el futur. Si la documentació és correcte, la comprensió del projecte serà senzilla i això farà que en el futur es pugui millorar, modificar o fins i tot vendre d'una manera més senzilla ja que qui hagi de veure el projecte es podrà recolzar en una documentació on li explicarà de manera senzilla el projecte.

4.5 Temporalització

El projecte consta de quatre fases les quals ja s'han explicat i ara es calendaritzaran.

Les primeres tres fases estan totalment lligades i no es pot fer una sense abans haver fet l'anterior. Així doncs, es començarà per la primera, després la segona i finalment la tercera.

En canvi, la fase 4, la documentació és una fase que s'haurà de fer durant tot el projecte, ja que hi ha apartats que s'han d'emplenar abans de fer el projecte, d'altres mentre s'està desenvolupant i finalment tancar la documentació quan el desenvolupament hagi finalitzat.

A la següent taula es veu la planificació que s'ha fet respecte les fases esmentades i els mesos que ha durat el desenvolupament d'aquest projecte.

	Març	Abril	Maig	Juny	Juliol	Agost
Estudi previ	■	■				
Anàlisi i disseny		■	■			
Implementació i proves		■	■	■	■	
Documentació	■			■	■	■

Capítol 5

Marc de treball

5 Marc de treball

Primerament, cal dir que el resultat final d'aquest projecte serà una aplicació d'escriptori. I estarà dividida en varies parts:

1. Obtenció i manipulació del mapa.
2. Execució de l'aplicació i algoritmes.
3. Visualització del resultat.

A continuació s'explicaran més detalladament cada part però sense entrar en aspectes massa tècnics. Això es durà a terme al capítol 6.Requisits del sistema esmentant tots els requisits i en el 8.Anàlisi, disseny i model matemàtic on es tractarà el tema més tècnic.

5.1 Obtenció i manipulació del mapa

El projecte està basat en un mapa d'entrada que ens introdueix l'usuari. En aquest apartat s'explicarà el procés d'obtenció i manipulació del mapa.

Obtenció del mapa

El paràmetre d'entrada del projecte és un mapa. Aquest mapa ha de ser un mapa *d'Open Street Maps* en format *osm*, aquest format simplement és un fitxer *xml* amb unes etiquetes que segueixen un protocol que ha definit *Open Street Maps*. S'ha escollit aquest format perquè és molt senzill d'obtenir, és gratuït i està gestionat per la comunitat. Això fa que el contingut estigui actualitzat constantment i sigui gratuït. Per obtenir el mapa hi ha varis llocs webs, però es pot fer des de la mateixa web oficial [5] [6]. Des d'aquí es pot buscar una zona en el mapa i fer un rectangle per obtenir la part del mapa que es desitgi estudiar. Després s'ha de clicar a exportar i es descarregarà un fitxer *osm.*

Fitxers osm

Els fitxers *osm* contenen nodes i camins. Els nodes són la base del mapa i són els que contenen tota la informació. Els nodes contenen les coordenades, el tipus, el nom, les característiques, etc.

Un node pot estar a un punt d'una carretera on es faci un canvi de direcció i per tant es necessitin marcar les coordenades de nou. O també pot ser cada element d'un mapa, com per exemple, un pas de zebra, un semàfor, un arbre, etc. En els nodes es poden afegir tantes característiques com siguin necessàries. Així serà com es posarà la influència a les seus.

L'altre element són els camins, que simplement són un conjunt de nodes. Els nodes definits prèviament es col·loquen dins dels camins en ordre per formar-los.

Col·locació de les seus

Aquest procés hauria de ser automàtic i en el treball futur aquest és un dels punts més importants que s'explicaran. En aquesta versió del projecte, aquest procés serà manual i

no formarà part del desenvolupament. És un pre-procés que s'ha de fer per poder utilitzar l'aplicació. Aquest procés és tediós i llarg. Consta de buscar cada seu en el mapa descarregat i assignar-li una influència. Aquest valor es posarà com a una característica del node que fa referència a la seu en el fitxer *osm.* El valor pot anar de 0 a 10 amb decimals (float).

Per fer aquest procés s'ha trobat un programari lliure desenvolupat amb *Java* que pot ser executat directament (*standalone*) ja que és un *jar*. Aquest programari s'anomena *JOSM* [7]. Amb aquest programari es pot carregar un fitxer *osm* i manipular-lo, ell s'encarrega de modificar el fitxer *osm* de tal manera que quedi tal hi com se li ha indicat en el mapa. Permet crear nodes i camins amb molta facilitat i modificar-los.

Neteja del mapa

Els mapes tenen molt de detall i en aquest projecte només es necessiten una sèrie d'elements, per tant hi ha un pre-procés de neteja del mapa. Aquest procés el que fa és eliminar tots els elements del mapa que no aporten informació rellevant per aquest projecte. En definitiva, s'eliminen tots els elements menys les carreteres, carrers, camins i les seus que ja s'han definit prèviament.

Aquesta neteja si que la farà la pròpia aplicació.

5.2 Execució de l'aplicació i algoritmes

Una vegada s'ha tractat el mapa per obtenir l'entrada desitjada ja es pot executar l'aplicació. I aquesta execució tindrà varies parts:

1. Lectura del mapa tractat i creació d'estructures.
2. Càlcul de les influències.

Lectura del mapa tractat i creació d'estructures

Quan s'executi el programa s'haurà de llegir el mapa en format *osm* i convertir-lo en una estructura on es puguin aplicar els càlculs i algoritmes que es defineixin en el disseny de l'aplicació.

Les estructures que s'utilitzaran han de permetre desar un mapa de carreteres i aplicar-li càlculs de trajectòries, etc.

Càlcul de les influències (model matemàtic)

Finalment, s'hauran de fer els càlculs pertinents de geometria computacional de càlcul d'influències. Aquest procés donarà uns resultats on es veuran quines zones del mapa són les millors i les pitjors per construir un hotel respecte uns paràmetres establerts que s'explicaran al capítol 8.Anàlisi, disseny i model matemàtic.

Existeixen molts algoritmes que donen solució a problemes semblants al que es proposa en aquest projecte, però cap s'hi ajusta perfectament, així doncs s'ha creat un nou algoritme que té algunes diferències respecte la resta.

A l'assignatura del màster Processament de dades espaials [8] es va parlar dels models existents per trobar les influències de les seus en un mapa:

- Voronoi: no s'ajusta al problema d'aquest projecte ja que no es vol saber la influència de cada seu, ni saber la influència de la nova seu, sinó trobar el punt on

el valor de la influència sigui màxim sumant les seus positives i restant les negatives.

- K-veïns: és necessari crear grups de seus d'un número concret. Tampoc és el que s'està buscant en aquest projecte. El que es vol és saber el punt màxim de tot el mapa sense importar que hi hagi seus properes o no.

A més a més cal recordar que aquest projecte treballa sobre una xarxa de carreteres per les quals la influència de les seus ha de propagar-se seguint la distància de la carretera.

Hi ha forces estudis en els quals es calcula la influència que exerceixen les seus existents, cal remarcar articles que treballen amb:

- Regions d'influència
- Influència comuna
- Influència mútua
- Influència reversa

L'algoritme que es proposa en aquest difereix de tots els estudis fets, es tracta d'un problema bichromatic, hi ha un grup de seus d'un tipus que compateixen contra un altre grup de seus d'un altre tipus. Però en tots els articles existents es treballa amb els k-veïns més propers i en aquest cas no hi intervenen en cap cas. Per altra banda els estudis existents no treballen amb distàncies en xarxes de carreteres sinó amb la distància euclídia.

Tal i com veurem gràcies a la xarxa de carreteres i les influències de les seus, s'ha trobat un algoritme senzill i eficient que utilitza l'algoritme de *Dijkstra* modificat per propagar la influència de les seus en els nodes. A continuació s'ha creat un algoritme molt eficient perquè s'ha pogut aprofitar la localitat del problema gràcies a haver propagat la influència a tots els nodes amb el Dijkstra, això fa que el problema sigui local. A més a més, la influència està basada en la distància, per tant, és continua i això fa que no s'hagi de calcular cada punt del mapa com s'explica en les referències de l'assignatura, ja que calculant uns punts concrets i aprofitant la interpolació lineal es pot aconseguir el resultat i a més a més és un resultat exacte. Més endavant, en el capítol 8.Anàlisi, disseny i model matemàtic s'explicarà més detalladament. [9] [10]

5.3 Visualització del resultat

Per acabar l'aplicació caldrà veure els resultats que ha calculat. S'ha de tenir en compte que s'ha de veure el mateix mapa que el client ha entrat però amb una visualització senzilla que permeti al client veure de manera ràpida i intuïtiva quines són les zones bones i quines no. Per tant, hi ha un requisit no funcional molt clar que s'haurà de complir i que fa que les formes de crear aquesta visualització es redueixin. Al capítol 7.Estudis i decisions s'explicarà la decisió i el perquè.

Capítol 6

Requisits del sistema

6 Requisits del sistema

Per poder definir els requisits de l'aplicació s'han fet varies reunions. Per una part, s'han fet reunions amb l'empresa hotelera que s'encarregarà de provar l'aplicació un cop feta i que també s'ha encarregat d'explicar les necessitats que tenien i així s'han pogut definir els requisits funcionals. D'altre banda, s'han fet varies reunions amb la tutora del projecte per definir els requisits no funcionals de l'aplicació.

A continuació es detallarà cadascun dels requisits de l'aplicació, tant els funcionals com els no funcionals.

6.1 Glossari

Abans de començar a definir els requisits del sistema, es definiran les paraules necessàries per poder comprendre millor aquest projecte.

- **Influència:** és un valor que està associat a una seu. El valor sempre serà un número natural i serà entre 0 i 10 ambdós inclosos. El valor sempre serà positiu però la influència que exerceix la seu sobre una altre seu pot ser negativa o positiva depenen del tipus de seus considerades.
  La influència és un valor que indica la potència d'atracció d'una seu, com més gran sigui la influència més metres tindrà el radi d'influència d'aquella seu. La influència està lligada amb la distància completament. Les distàncies són sobre la xarxa de carreteres, per tant, la influència es propagarà per les carreteres.
- **Mapa:** és un arxiu *osm (Open Street Maps)* on hi ha el tros de mapa que es vol analitzar. En aquest mapa hi ha d'haver les seus amb la seva influència introduïda.
- **Seu:** Hi ha dos tipus de seus:
  - Seu negativa: És un establiment que aporta influència negativa sobre el negoci que es vulgui construir, és la competència. Per exemple, en el cas de que es vulgui construir un hotel, la competència seran altres hotels o altres establiments que fan que la zona ja estigui ocupada i no sigui interessant crear un hotel nou.
  - Seu positiva: Són construccions o zones que tenen una influència positiva sobre el negoci que es vol construir. Per exemple, en el cas dels hotels interessa que hi hagi turistes i una seu amb influència positiva podria ser una catedral o un centre comercial, etc. Aquestes seus fan que una zona sigui atractiva per poder construir o comprar un establiment.
- **Usuari:** és l'únic actor de l'aplicació. És la persona que s'encarrega d'executar l'aplicació amb un mapa amb el format correcte i amb les seus ben introduïdes. I és el que rebrà el resultat per pantalla i podrà llegir el mapa resultant.

6.2 Requisits funcionals

En el cas d'aquest projecte només hi ha un actor que és el que interactua amb l'aplicació i l'executa. Per tant, només hi ha requisits funcionals per aquest actor. Són aquests:

- **Carregar mapa:** L'usuari ha de poder executar l'aplicació amb un mapa *osm* i amb les influències de les seus introduïdes.
- **Veure mapa resultant:** L'usuari ha de poder veure el mateix mapa que ha introduït però amb els carrers pintats de diferents colors indicant la influència de cada punt del mapa.

6.3 Requisits no funcionals

Els requisits funcionals indiquen què ha de fer l'aplicació. Els requisits no funcionals indiquen com s'han d'assolir els requisits funcionals.

En el cas d'aquest projecte es va començar pensant uns requisits no funcionals diferents dels que al final han acabat siguen. A continuació es definiran primer els requisits inicials i després els requisits que finalment s'han acabat fent. Aquest canvi de requisits ha estat per varis motius, hi ha hagut una part de falta de temps de desenvolupament i d'altre banda els requisits de l'inici van perdre sentit mentre es va avançar el projecte i es van trobar altres maneres de solucionar els requisits funcionals.

Els requisits no funcionals d'aquest projecte estan enfocats en l'eficiència dels càlculs, estructures i temps d'execució, en tenir un model matemàtic ben dissenyat.

6.3.1 Requisits no funcionals inicials

A l'inici del projecte es van decidir uns requisits no funcionals ambiciosos dels quals, alguns no s'han pogut assolir, altres s'han modificat perquè el projecte ha anat canviant o finalment d'altres que sí que s'han assolit.

- **Eficiència:** s'ha de crear una aplicació que pugui desar, manipular i pintar un mapa de manera eficient, per tant s'han de construir unes estructures prou dinàmiques com per poder tractar qualsevol mapa. Per aconseguir aquesta eficiència és necessari que aquestes estructures tinguin les dades indispensables per complir l'objectiu.
- **Rapidesa:** L'aplicació ha de ser ràpida. Quan l'usuari executi l'aplicació espera un resultat ràpidament i se li ha d'entregar el resultat amb la màxima brevetat possible. Per aconseguir-ho s'estudiaran varies tecnologies i tècniques que al capítol 7.Estudis i decisions s'explicaran.
- **Paral·lelització:** Fer càlculs sobre un mapa és repetitiu. Es poden paral·lelitzar aquests càlculs i així fer que l'aplicació sigui molt més ràpida. En el cas d'aquest projecte es pot veure com si tingués un mapa amb molts píxels i s'ha de fer un càlcul per cada píxel per saber quina puntuació té. Per tant, té molt sentit fer paral·lelització.
- **GPU:** Enllaçat amb el requisit anterior, s'estan manipulant mapes i píxels, els quals se'ls hi ha d'aplicar un càlcul a cadascun. Treballar amb GPU té moltes limitacions però si el problema es pot solucionar amb GPU probablement serà

molt més ràpid. A més a més hi ha una part didàctica, on l'alumne volia aprendre la programació amb GPU, aprofitant també, que és una de les especialitats de la tutora del projecte.

- **Visualització clara i dinàmica:** Finalment, el mapa s'ha de visualitzar. I aquesta visualització ha de ser prou clara perquè el client ràpidament pugui veure els resultats i entendre'ls.

6.3.2 Requisits no funcionals finals

Degut a decisions mentre es desenvolupava el projecte s'han modificat alguns dels requisits inicials.

- **Sense paral·lelització:** No s'ha utilitzat programació amb fils o processos per paral·lelitzar l'execució.
  D'entrada es volia utilitzar aquest mecanisme per calcular la influència associada a cada punt, per tal d'obtenir un mapa discret d'influències, al final s'ha obtingut un algoritme que dóna el resultat de forma exacte i no hi ha hagut la necessitat de fer aquest càlcul de forma discreta per cada punt.
  Aquesta decisió s'explicarà detalladament a l'apartat 7.6.Paral·lelització.

  Per tal de visualitzar els resultats sí que s'ha utilitzat la paral·lelització integrada al llenguatge OpenGL d'interpolació lineal del color associat a un objecte, en aquest cas un segment de recta.

- **Sense GPU:** A conseqüència del punt anterior no s'ha utilitzat programació amb GPU ja que si no s'ha de paral·lelitzar perd tot el sentit. Com en el cas anterior s'explicarà detalladament a l'apartat 7.6.Paral·lelització. Tal i com es veurà s'hauran de calcular diferents *Dijkstras* a sobre de la xarxa de carreteres, es podria paral·lelitzar aquest càlcul, però no és senzill i seria aplicar algoritmes ja existents que no donarien massa valor afegit al projecte.

Capítol 7

Estudis i decisions

7 Estudis i decisions

A continuació s'explicaran les decisions que s'han pres durant el projecte. Ara que els requisits han estat definits es poden escollir les tecnologies que s'utilitzaran per poder satisfer aquests requisits.

Aquestes decisions són de diferents tipus així que es dividiran en varis apartats:

- Maquinari.
- Sistema operatiu.
- Llenguatges de programació.
- Programari.
- Control de versions.
- Estructures.
- Algoritmes.
- Paral·lelització.
- Visualització.

7.1 Maquinari

Les característiques d'aquest projecte fan que el maquinari sigui poc important ja que amb les decisions de programari que s'han decidit i que es comentaran a continuació l'aplicació és ràpida i lleugera.

Això ha estat així després de canviar el requisit de no utilitzar la GPU per desenvolupar, ja que al principi del projecte es va pensar de comprar una targeta gràfica per poder utilitzar *CUDA*.

En aquesta fase de desenvolupament no ha estat necessari cap servidor, simplement un ordinador per poder desenvolupar, aquest ordinador té aquestes característiques:

- Processador Intel® Core™ i7-6600U (Dual Core, 2.6GHz, 4M cache, 15W)
  - 4 nuclis amb 2 fils per nucli
- Un DIMM de memòria RAM:
  - 8GB (1x8GB) 1600MHz DDR3 Memory
- Un disc dur sólid
  - M.2 256GB SATA Class 20 Solid State Drive
- Tarja gràfica integrada
  - Intel

7.2 Sistema operatiu

El sistema operatiu és molt important ja que una mala elecció pot fer que el projecte no tingui els resultats desitjats ja que no es pot utilitzar algun tipus de programari o per recursos.

Per tant, es necessita un sistema operatiu que tingui les següents característiques:

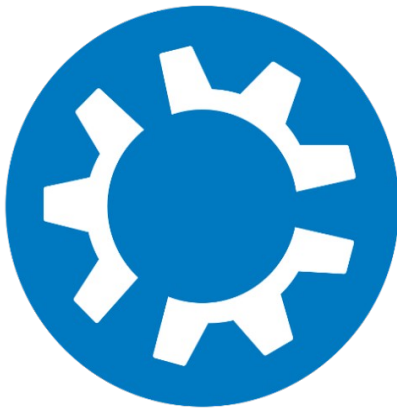
- Segur.
- Estable.
- Vigent, amb actualitzacions periòdiques de programari i de seguretat.
- Amb una comunitat gran i activa.
- Familiar pel desenvolupador del projecte.
- Bona gestió de recursos.

S'han estudiat varis sistemes operatius:

- Windows 10.
- Debian.
- Ubuntu.

Finalment, el sistema operatiu serà Linux, concretament Ubuntu que és una distribució basada en Debian. Perquè és el sistema operatiu més familiar pel desenvolupador, gestiona bé els recursos, és ràpid i té tot el programari necessari per poder desenvolupar el projecte. A més a més, no s'utilitza programari privatiu que pugui comportar costos addicionals més endavant.

Concretament, cal dir que el sistema operatiu és Ubuntu, però una versió anomenada Kubuntu que és un Ubuntu amb l'entorn gràfic *KDE* [11].



Il·lustració 6: Kubuntu

7.3 Llenguatges de programació

Per fer aquest projecte s'han estudiat principalment dos llenguatges de programació:

- Python
- C++

Els dos llenguatges són molt potents, cadascun té punts positius i negatius. L'escollit no té perquè ser millor, simplement és perquè s'adapta millor al projecte.

7.3.1 Python

[12]

Il·lustració 7: Python

És un llenguatge interpretat molt potent i molt utilitzat en l'actualitat en molts àmbits diferents: acadèmic, visualització, *scripting*, etc.

Python és un llenguatge pensat per ser senzill d'obtenir resultats i proporciona eines molt potents que amb poc esforç poden generar un gran resultat.

Avantatges

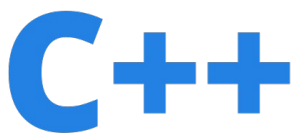
- Familiar pel desenvolupador.
- Moltes llibreries.
- Modern.
- Senzill.
- Comunitat molt gran i activa.
- Bona documentació

Desavantatges

- Poca compatibilitat amb *CUDA*.
- Gestió de recursos.
- Eficiència.

7.3.2 C++

[13]



Il·lustració 8: C++

C++ és un llenguatge compilat que prové de C però és orientat a objectes. És un llenguatge molt potent que s'utilitza molt a nivell acadèmic i matemàtic.

Avantatges

- Molt potent.
- Punters.
- Molt eficient.
- Compatibilitat amb *CUDA*.
- Bona documentació.
- Comunitat gran i activa.
- Pots accedir a seccions de memòria (programació abaix nivell).

Desavantatges

- Complicat.
- Massa estricte.
- Rígid.

L'escollit ha estat C++. Principalment perquè suposa un repte i aquest projecte al ser acadèmic permet fer eleccions que al món laboral no es podrien fer, ja que escollint C++ el desenvolupament serà més lent per la poca familiaritat que hi té el desenvolupador.

També cal dir que C++ té unes llibreries molt potents per gestionar *CUDA* i *OpenGL*.

Finalment, C++ és més eficient, un clar exemple és que les llibreries més ràpides de *Python* estan desenvolupades amb C++.

Hi ha una opció que barreja els dos llenguatges anomenada *Cython*, que és una barreja on es pot programar amb C++, es compila i es pot importar com a un mòdul de *Python* preparat per utilitzar. Es va estudiar aquesta possibilitat, però finalment es va decidir optar per C++ pur.

Finalment, cal dir que al començament del desenvolupament es va fer una part amb *Python* i finalment es va traspassar tot a C++ formant un sol projecte. A l'apartat 9.1.Canvi de llenguatge s'explicarà detalladament.

7.4 Programari

S'ha escollit una sèrie de programari per poder desenvolupar el projecte. S'ha de tenir en compte que s'han pres dos decisions molt importants arribats a aquest punt. El sistema operatiu Linux i el llenguatge de programació C++.

7.4.1 IDEs

- **Clion:** és l'IDE de C++ de l'empresa *jetbrains* [14]. Aquesta empresa té varis IDEs. Són privatius i no són gratuïts però es pot aconseguir una llicencia anual si es posseeix un correu electrònic acadèmic i si la finalitat del projecte no és comercial com és aquest cas.
- **Pycharm:** és l'IDE de *Python* de l'empresa *jetbrains*. És el mateix que el punt anterior però per *Python*.

Il·lustració 9: Jetbrains

7.4.2 Editors de mapes *Open Street Maps*

Es necessita un programari per poder editar el mapa descarregat *d'Open Street Maps* ja que aquesta part no forma part del projecte. El projecte necessita un fitxer *osm* on la influència de les seus ha d'estar reflectida. Per fer això es necessita un programari que ens ajudi a fer-ho. A la *wiki d'Open Street Maps* hi ha una llista amb tots els programes que fan aquesta tasca [15].

- **Qgis:** És un projecte molt gros, difícil d'entendre i d'utilitzar. Té moltes funcions que no són necessàries pel nostre projecte [16].
- **Merkaartor:** És un programari instal·lable senzill però té poca documentació [17].
- **JOSM:** És un programa fet amb Java el qual es pot executar fàcilment ja que és un *jar*.
  Llegeix els fitxers *osm* i mostra el mapa del fitxer per pantalla. Després amb les eines facilitades es pot editar qualsevol element del mapa molt fàcilment [7].

L'elecció ha estat *JOSM* per la facilitat d'ús i perquè no cal instal·lar-lo ja que és un *jar.* Per tant, és totalment portable.

Il·lustració 10: JOSM

7.4.3 Netejadors de mapes *osm*

Els mapes *osm* són molt complerts, tenen molts elements que per aquest projecte no tenen cap rellevància. I era necessari fer neteja d'aquests nodes inútils. Això farà que l'aplicació sigui molt més ràpida ja que alhora de llegir el fitxer *osm* hi haurà molta menys informació. Per fer aquesta neteja també ens hem servit d'un programari que elimina de forma automàtica tipus concrets de nodes.

En el cas d'aquest projecte, el que es necessita és eliminar tots els nodes menys els de les carreteres i els nodes de punts d'interès o hotels.

Hi ha dos opcions que són pràcticament iguals.

- **Osmosis:** Aquest programari és un petit script de Linux instal·lable que llegeix el fitxer *osm* i elimina els nodes del tipus que se li especifica. La comunitat era petita i les funcions eren limitades [18].
- **Osmfilter:** Aquest, com l'altre, és un programa instal·lable en el Linux i s'executa des de consola. Té una comunitat més gran i és més potent ja que a molts fòrums els problemes que tenien amb *l'Osmosis* els solucionaven amb aquest. A més a més, la documentació és molt complerta [19].

Per tant, l'elecció ha estat *Osmfilter*. Com s'ha dit, és un programa instal·lable que s'executa des de consola. Amb això s'aconsegueix que el fitxer entrant sigui molt més petit i això beneficia el rendiment de l'aplicació. Aquesta part està dins de l'aplicació i quan s'executa el projecte, a dins, s'executa la comanda d'*Osmfilter.* És transparent per l'usuari.

7.4.4 Lectura de mapes *osm*

Un mapa *osm* és un *xml*, per tant es necessita una llibreria que pugui *parsejar xml.* Qualsevol llibreria que s'utilitzi no evitarà els mal de caps de *parsejar xmls* en C++. Comparat amb altres llenguatges és molt més complicat, ja sigui per la necessitat d'instal·lar una llibreria, compilar-la i utilitzar-la.

S'han estudiat varies llibreries per manipular *xml* en C++:

- **Xerces-C++:** és una llibreria d'*Apache*, per tant té tota una comunitat important desenvolupant-la i és de confiança. És una llibreria molt grossa que té una gran quantitat de funcions [20].
- **RapidXML:** és una llibreria petita i ràpida. És fàcil d'utilitzar i la documentació és bona [21].
- **PugiXML:** Té totes les avantatges del *RapidXML* però a més a més és molt més senzilla d'utilitzar. És un projecte de *GitHub* que està en continu desenvolupament per la comunitat. En el moment d'escriure aquesta documentació, només feia 8 dies que es va fer l'últim *commit* [22].
- **TinyXML:** és simple i petit com els dos anteriors, però llegint varis fòrums es va arribar a la conclusió de no utilitzar-lo ja que molts usuaris comentaven que era poc eficient a nivell de memòria i la documentació no és massa complerta [23].

L'elecció final ha estat **pugiXML** perquè és fàcil d'instal·lar i proporciona les funcions necessàries per desenvolupar el projecte, ni molt poques, ni masses. Aquesta web va servir de molta ajuda per poder prendre aquesta decisió [24].

7.5 Control de versions

És una part molt important en qualsevol projecte de software. Tenir un control de versions aporta molts beneficis:

- Tenir el codi versionat.
- Tenir un *log* de tots els canvis.
- Tenir un *backup* del codi de cada moment.
- Tenir el codi al núvol.
- Treball en equip.
- Treballar en varies branques.

Aquest projecte és petit i individual, això fa que tenir un control de versions sigui més prescindible, però inclús en un projecte així aporta molts beneficis i cap inconvenient.

Així doncs, s'utilitzarà un control de versions, principalment per tenir *backups*, per poder treballar en varies branques així es poden desenvolupar funcionalitats diferents sense necessitat de trepitjar-les i per tenir-ho protegit en el núvol, així s'hi pot accedir des de qualsevol lloc.

Hi ha varis programes de control de versions, els més famosos són *Subversion, Mercury* i G*it.* Però en aquest projecte només s'ha estudiat *Subversion* i *Git.*

- **Subversion:** és antic, funciona de tal manera que només guarda les línies que s'han modificat, no tot l'arxiu. Té molts problemes alhora de fer els *merges* [25].
- **Git:** és el més popular i el que funciona millor. Guarda tot el codi a cada ordinador que es descarrega el projecte i si hi ha canvis guarda tot l'arxiu. Fa els *merges* correctament i això és molt important ja que no s'ha de repassar el codi i es pot confiar en ell [26].

Clarament, l'elecció ha estat *Git*. És molt utilitzat i funciona molt bé.

A més a més, està integrat a l'IDE que s'utilitzarà. Cal dir que *Subversion* també està integrat.

Per tant, el control de versions es gestionarà des del propi IDE.

El repositori es pujarà al núvol, concretament a *Bitbucket* [27] ja que ja ha estat utilitzat en projectes anteriors, funciona bé i es poden crear repositoris privats a diferencia de *GitHub* que ha de ser públic.

Il·lustració 11: Git

Il·lustració 12: Bitbucket

7.6 Paral·lelització

En un inici, el projecte es va plantejar fer-lo paral·lelitzant els càlculs i per fer això es volia fer amb programació de GPU. Una part perquè era factible i era un idea bona per millorar el rendiment, però principalment era una decisió acadèmica, ja que és una programació nova i un repte que aportava una motivació extra cap al projecte.

Per programar en GPU es van estudiar dos llenguatges:

- **CUDA:** És un llenguatge de programació de GPUs de l'empresa *NVIDIA* i per tant és necessari tenir una targeta gràfica *NVIDIA* per poder-ho utilitzar. Aquest és el principal problema ja que no hi havia cap targeta *NVIDIA* disponible. Es va estar apunt de comprar una targeta però finalment es van optar per altres opcions. Tot i que *CUDA* és el llenguatge que fa servir la tutora i és el més complert per GPUs ja que al tenir les targetes dedicades és més senzill fer-ho [28].
- **OpenCL:** Es podria dir que és el *CUDA opensource*, funciona amb totes les targetes no només amb les *NVIDIA*. Aquesta era la opció que s'hauria utilitzat ja que la targeta gràfica de la que es disponia era *Intel* [29].

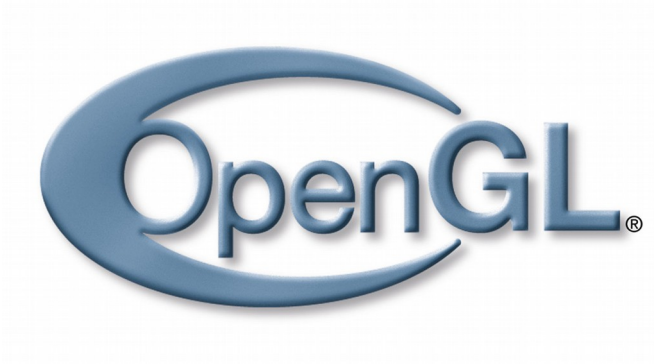
Finalment, com ja s'ha comentat, no se'n va escollir cap perquè es va trobar una solució millor sense necessitat d'utilitzar GPU.

7.7 Visualització

Per la part de visualització del mapa resultant també s'ha utilitzat una llibreria. Es van

estudiar dos llibreries:

- **Vulkan:** d'aquí poc temps serà la llibreria per excel·lència si continua així. Té moltes compatibilitats i té el suport d'*ATI* que li aporta una estabilitat necessària per lluitar contra *DirectX* de *Microsoft. Vulkan* és molt nou, són unes llibreries molt potents però també es necessita una targeta gràfica potent. Es va intentar instal·lar, però després de varies proves no es va poder aconseguir ja que els *drivers* per les targetes gràfiques *Intel* encara no funcionaven correctament [30].
- **OpenGL:** són un conjunt de llibreries que porten molts anys al mercat i és la competència actual de *DirectX* tot i que no té el potencial suficient. És *open source* i té una comunitat important. Però la documentació és molt dolenta i és complex aprendre'n perquè hi ha poca informació. Té bona compatibilitat amb C++. *Vulkan* serà el seu successor [31] [32] [33] [34].



Il·lustració 13: OpenGL

La decisió ha estat *OpenGL* perquè *Vulkan* no es va poder instal·lar correctament. S'ha utilitzat la versió 2.0 perquè és la més comuna i no hi havia necessitat d'utilitzar versions més noves per les visualitzacions que s'havien de fer en aquest projecte.

Capítol 8

Anàlisis, disseny i model matemàtic

8 Anàlisi, disseny i model matemàtic

L'anàlisi ha de servir per trobar totes les necessitats del projecte i es detallaran amb els casos d'ús de l'actor del projecte per desenvolupar els requisits comentats en el capítol 6.Requisits del sistema. S'utilitzaran diagrames de cas d'ús per il·lustrar els requisits.

A més a més, es presentarà la formalització matemàtica del problema que es vol resoldre i també es presentaran els algoritmes bàsics utilitzats per desenvolupar el projecte.

8.1 Formalització del problema

8.1.1 Glossari

Abans d'entrar en la pròpia formalització es definiran les inicials utilitzades, per ajudar a la comprensió del lector (ordenats alfabèticament).

A: aresta

D: distància

DM: distància mínima entre punts P

E: error

G: graf no dirigit

GS: graf resultant

I: influència

N: tot node de G, ja sigui seu S o no

NF: node final d'una aresta A

NI: node inicial d'una aresta A

P: tot punt del graf GS

PC: punt de canvi, on una seu S comença o acaba d'influir

PM: punt amb el VI més alt

S: seu

SC: seu competidora (seu)

SN: seu amb influència negativa

SO: seu objectiva

SP: seu amb influència positiva

VI: valor d'influència

VII: valor d'influència del node N inicial

8.1.2 Definició del problema

Es vol col·locar una nova seu, la seu objectiva SO on, donat un mapa de carreteres representat amb un graf G no dirigit, on hi ha seus S que poden influir positivament SP o seus que poden influir negativament SN, s'ha de trobar el punt on hi hagi la influència positiva màxima PM, aquest punt és el millor lloc on col·locar la SO.

8.1.3 Definició de la solució

Com a idea general, a cada punt P del mapa s'ha de sumar la influència I de les SP que influeixin en aquest punt i restar la influència I de les SN que influeixen en el punt P. Es mostraran tots els VI mostrant el mapa amb un degradat de color d'acord amb el VI dels diferents punts del mapa. Després, comparant tots els VI de tots els punts P s'aconseguirà el PM on s'hi podrà col·locar la SO.

$$VI = \sum_{SP} I - \sum_{SN} I$$

Estenent aquesta idea s'han fet varis algoritmes per desenvolupar-la.

Primer s'ha fet una simplificació del graf G. S'ha utilitzat l'algoritme de *Reumann-Witkam* per passar d'un graf G amb n nodes N a un graf simplificat GS amb n-m nodes N, on m<n però sense perdre informació rellevant.

A continuació, s'ha utilitzat l'algoritme de *Dijkstra* però aplicant-hi unes modificacions per propagar la influència I de totes les seus S a cada node N. On N pot ser una seu S o no.

Per cada seu S del graf simplificat GS es propaga la seva influència I a través de les arestes. Per cada node N que S influeixi a N se li afegeix la informació de S i la influència que té en el node N. L'algoritme passa a la següent seu S quan la distància D recorreguda és major a la influència inicial I de S. $D > I$

La influència I és distància D, per tant, mentre es recorren les arestes la distància D recorreguda augmenta i la I disminueix de la mateixa manera.

Així doncs, quan l'algoritme de *Dijkstra* acabi tots els nodes N tindran la informació de totes les seus S que l'influeixen i sabrà quina influència I de cada S té en aquell node N. Això fa que el problema passi de ser global a local. Ja que ara es pot trobar el valor d'influència VI de cada punt P només calculant les influències I que té guardat el node N inicial i final de l'aresta del punt P.

Ara es podria recórrer tot el GS i calcular cada VI de cada punt P, però s'hauria de definir una distància mínima DM entre els punts P, ja que aquests són infinits. Així doncs, sempre hi hauria un error E de com a màxim distància mínima DM. Per tant, el PM no seria exacte. Tot i que es podria anar refinant i disminuint E segons les necessitats de l'usuari en zones concretes d'estudi.

Sabent que la influència I és distància D, es pot afirmar que si es recorre una aresta la D augmentarà i la I disminuirà igual, fins que una seu S que influeixi deixi d'influir o una de nova comenci a influir-li, en els trams en què les seus que hi influeixen són les mateixes el valor de la influència canvi de forma contínua.

Sabent això, només hi ha certs punts P en el graf GS que siguin candidats a ser PM. Aquests punts P, s'anomenaran punts de canvi PC.

Els PC són aquells punts P on una seu S comenci o acabi d'influir. Aquests són els únics

punts on hi pot haver el PM, ja que la influència I de les seus S mentre es recorri l'aresta augmentarà o disminuirà, però ho farà fins a cert punt, on acabarà d'influir, si augmentava, el VI més elevat serà quan acabi d'influir i si disminuïa el VI més elevat serà quan ha començat a influir.

Cal tenir en compte, que només hi ha un valor de PM en el GS. En tot moment es guarda el PC amb un VI més elevat i quan s'acabi d'executar l'algoritme el PC amb VI més alt serà el PM. Ara bé, aquest PM es pot arribar a assolir en més d'un punt del GS.

L'algoritme de calcular els VI dels PC funciona així:

Per cada node N no visitat, on N és el node d'una aresta A, cada A té dos N, l'inicial NI i el final NF, sempre es començarà per l'inicial, es cerquen tots els PC de l'aresta A. Això es pot fer ja que NI i NF tenen la informació de totes les seus que els influeixen i la influència I o distància D que els hi queda per influir. Aquestes seus S s'ordenen per la D que els hi queda, de menor a major. Els PC són tots els punts on una seu S comença o acaba dins l'aresta A inclosos els NI i NF.

Primer es calcula el VI de manera «tradicional» $VI=\sum_{SP} I-\sum_{SN} I$ on, SP i SN són les seus que té el propi node NI guardats gràcies al Dijkstra. Aquest VI s'anomenarà valor d'influència inicial VII. A més a més es guardarà una estructura U amb la informació si augmenta la influència I o disminueix de cada seu S, s'indica amb un 1 en cas de que augmenti i amb un -1 si disminueix.

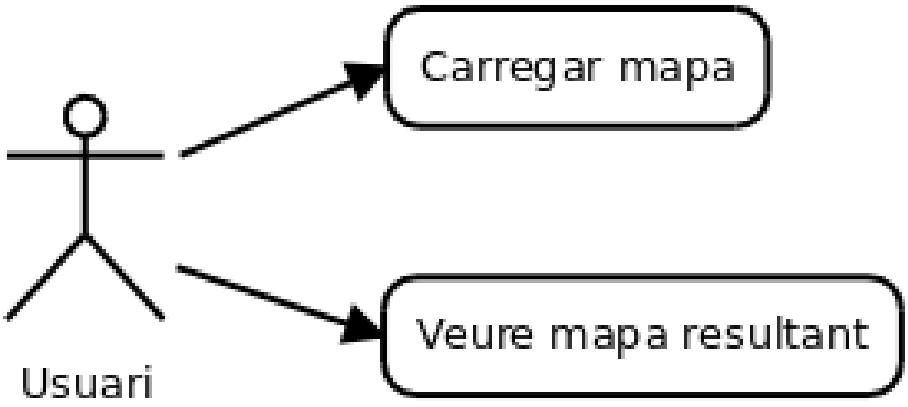
A continuació es recorre l'aresta A fins al primer PC (recordar que estan ordenats) on es calcularà el VI del PC mitjançant el VII i l'estructura esmentada.

$$VI=VII+\sum_{S}\left(I\times U\right)$$

Finalment, es marca NI i NF com a visitats, així no es torna a processar la mateixa aresta en sentit contrari. I es torna al principi per trobar el següent node N no visitat.

Per acabar, tinguen tots els VI de tots els PC es poden pintar en un mapa resultant i el propi OpenGL utilitza la interpolació lineal per emplenar tots els P que no són PC. Així doncs, queda un mapa resultant amb tota la xarxa de carreteres pintada, al menys fins on arriba alguna influència.

8.2 Usuari



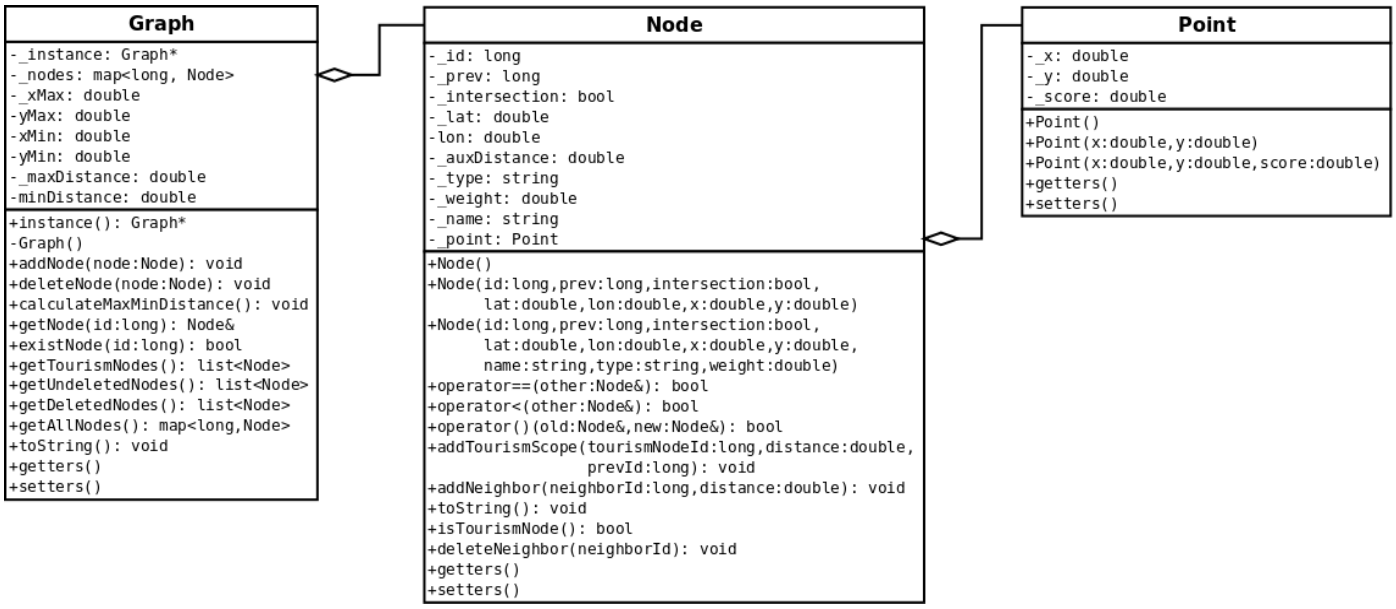
Il·lustració 14: Cas d'us de l'usuari

Com ja s'ha comentat en el capítol anterior, només existeix un actor en aquest projecte, que és l'usuari que executa l'aplicació. A continuació les seves fitxes de casos d'ús:

Identificador	U-1
Nom	Carregar mapa
Descripció	Un usuari pot carregar un mapa perquè l'aplicació la pugui analitzar
Actors	Usuari
Precondicions	L'usuari ha introduït les influències de les seus en el mapa
Seqüència normal	1. Executa l'aplicació amb el mapa com a paràmetre 2. S'executa l'aplicació correctament
Postcondicions	L'usuari executa l'aplicació correctament
Notes	

Identificador	U-2
Nom	Veure mapa resultant
Descripció	Un usuari rep el mapa resultant amb el resultat
Actors	Usuari
Precondicions	L'usuari ha introduït un mapa correctament i l'execució ha funcionat correctament
Seqüència normal	1. Executa l'aplicació 2. Veu el mapa resultant amb les influències indicades
Postcondicions	L'usuari veu el mapa correctament i l'aplicació finalitza
Notes	

8.3 Diagrama de classes

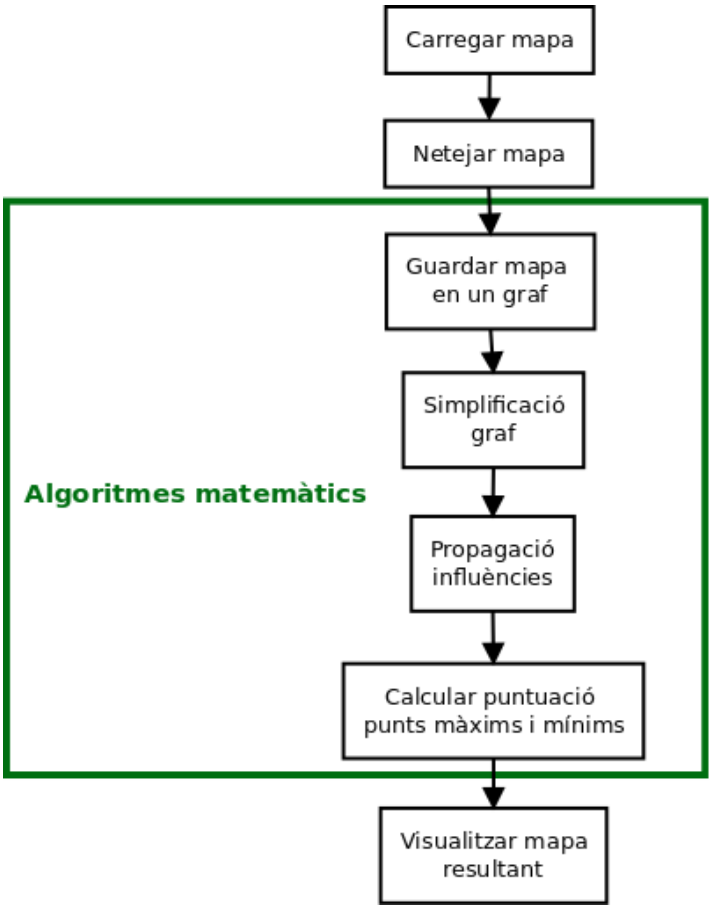


Il·lustració 15: Diagrama de classes

8.4 Flux de treball

L'aplicació té un flux de treball des de la seva execució fins a la seva finalització. I consta d'aquests punts en l'ordre següent:

1. Netejar el mapa entrat per l'usuari.
2. Lectura i guardat del mapa netejat en una estructura de dades en memòria.
3. Simplificar el mapa desat.
4. Propagar la influència de les seus per l'estructura de dades.
5. Calcular les puntuacions dels punts màxims i mínims.
6. Visualitzar el mapa resultant.



Il·lustració 16: Flux de treball

A continuació es procedirà a detallar cada punt del flux de treball explicant quin anàlisi es va fer:

8.4.1 Netejar el mapa entrat per l'usuari

El primer que calia fer era netejar el mapa que es rebia per paràmetre quan s'executa l'aplicació. El mapa arriba amb molts elements innecessaris i calia netejar-lo. Es va utilitzar una eina que ja s'ha comentat a l'apartat 7.4.3.Netejadors de mapes osm. Això retorna el mapa net que s'utilitzarà al proper punt.

8.4.2 Lectura i guardat del mapa netejat en una estructura de dades en memòria

Per llegir el mapa cal un *parsejador* de *xml* que com s'ha comentat a l'apartat 7.4.4.Lectura de mapes osm, es va escollir *pugiXML.*

Mentre es *parsejava* el mapa es guardava en una estructura de dades en memòria que seria amb la qual es treballaria a partir d'aquest punt.

L'estructura de dades havia de permetre mostrar i recórrer l'estructura de carreteres i aplicar-hi el model matemàtic que proporcionarà els resultats desitjats.

Aquesta estructura de carreteres s'havia de representar amb un **graf**, ja que és l'estructura més semblant a una xarxa de carreteres i s'hi podran aplicar algoritmes de graf molt útils com *Dijkstra*.

8.4.3 Simplificar el mapa desat

El mapa que l'usuari carregui a l'aplicació en aquest punt ja estarà netejat, per tant tindrà molts menys elements. Però de totes maneres tindrà molta informació irrellevant, això s'ha de solucionar ja que pot afectar molt al rendiment de l'aplicació. Les carreteres del mapa tenen moltes corbes i per poder mantenir la corba té molts nodes.

Per optimitzar l'aplicació s'ha aplicat un algoritme de simplificació de trajectòries anomenat *Reumann-Witkam*.

8.4.3.1 Algoritme Reumann-Witkam

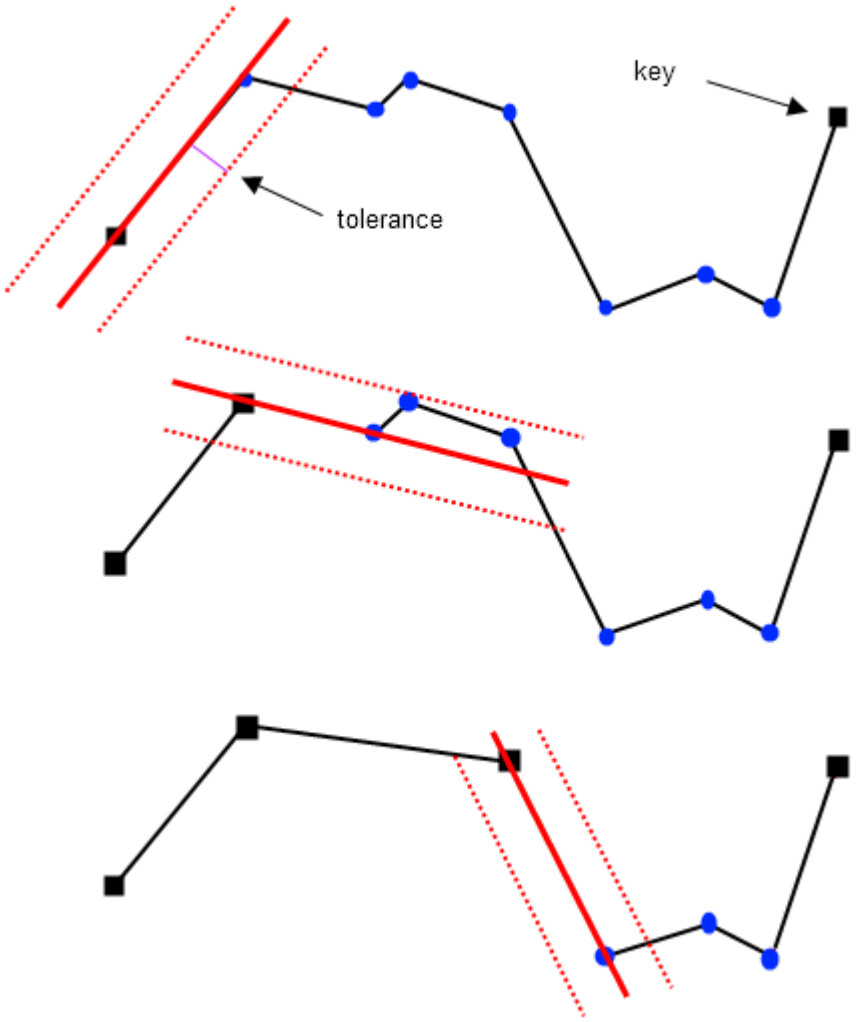
Aquest algoritme s'aplica per reduir el número de nodes d'una trajectòria mantinguen la seva originalitat. Aquest algoritme va ser estudiat en el projecte final de l'assignatura de Processament de dades espaials perquè es va fer un projecte sobre trajectòries.

L'algoritme més utilitzat de simplificació de trajectòries és el *Douglas-Peucker* [35] però és menys eficient que el *Reumann-Witkam* [36] [37].

Aquest algoritme també ha estat implementat pel desenvolupador del projecte.

Com es pot comprovar a la il·lustració següent (Il·lustració 17) hi ha varis passos:

1. Seleccionar el node inicial, marcar-lo com a node clau de la trajectòria i traçar una recta R fins al següent node.
2. Definir una tolerància que serà una constant durant l'execució de tot l'algoritme. Aquesta tolerància com més gran sigui més nodes suprimirà. La tolerància és la distància perpendicular de la recta R on si hi ha algun node serà suprimit. Això el que fa és crear una àrea A al voltant de la recta R.
3. Ara es comprova si hi ha algun node (a part dels dos primers) a l'interior d'aquesta àrea A.
    1. Si no n'hi ha: es marca el segon node com a node clau ja que no s'ha pogut eliminar. I es torna al punt 1 però començant per aquest node clau i el següent.



Il·lustració 17: Algoritme de Rumann-Witkam

2. Si n'hi ha: es busca de manera seqüencial cada node dels següents que estigui a dins de l'àrea A, quan s'arribi a l'últim que està a dins, es marca aquest últim com a node clau i s'eliminen tots els nodes que estaven dins de l'àrea A i no són claus i això també inclou el segon node que s'ha utilitzat per crear la recta. I es torna al punt 1, començant des d'aquest últim node clau.

8.4.4 Propagar la influència de les seus per l'estructura de dades

Per propagar la influència de les seus per la xarxa de carreteres s'ha utilitzat un algoritme anomenat *Dijkstra*. Utilitzar un *Dijkstra* és molt comú, però propagar la influència cap als nodes no tant. La primera idea era fer totalment el contrari, era agafar cada punt del mapa i comprovar quines seus arriben a aquest punt concret. Però això feia que el problema fos molt més complex a nivell de rendiment.

Amb la solució proposada, al principi, quan es crea el graf ja es poden posar a cada

node totes les seus que hi arriben i això s'ha fet amb el *Dijkstra*.

El mapa *osm* conté les coordenades geogràfiques de cada node, així doncs es pot calcular la distància amb coordenades. En un principi es va fer, però comportava varis problemes que s'explicaran a l'apartat 9.4.Canvi de coordenades i es va decidir modificar les coordenades geogràfiques per coordenades x, y en el pla. Aquesta decisió no afecta als resultats perquè una ciutat no és prou gran com perquè li afecti la corba de la Terra, per tant podem posar la ciutat en el pla i treballar més fàcilment amb les distàncies.

8.4.4.1 Algoritme de Dijkstra

És un algoritme de grafs per trobar el camí mínim entre un punt i la resta [38].

Té varis passos:

1. Es corren tots els nodes del graf i se'ls assigna un valor de distància infinit perquè així sigui substituït més endavant. Ara bé, en el node inicial se li posarà el valor de 0, ja que la distància entre el node inicial i ell mateix és 0.
2. Anem a tots els veïns del node en el que estem i comprovem per cada un la distància entre el node en el que estem fins al node veí més el pes del node actual és més petita de la que tenim guardada, si és així la modifiquem, i indiquem el node previ al veí, que és el node actual.
3. Es marca el node actual com a completat.
4. S'extreu el node del graf no completat que tingui un pes menor.
5. Es torna al pas 2 mentre hi hagi nodes no marcat com a completats.

8.4.4.2 Algoritme de Dijkstra en el projecte

Ara que s'ha explicat l'algoritme de *Dijkstra* cal dir que s'ha utilitzat una cua de prioritats per millorar el rendiment del *Dijkstra* així sempre es té el node no completat de pes menor a la primera posició i l'execució és molt més ràpida.

S'executa un algoritme de *Dijkstra* per cada seu que té el mapa. La propagació del *Dijkstra* no es para quan tots els nodes han estat visitats sinó quan la seu deixa de tenir influència, és a dir quan ha arribat a distància igual al seu valor d'influència. S'aprofita per guardar a tots els nodes del graf que el node inicial (la seu) els hi arribi: l'id de la seu, la distància entre el node actual i la seu i també el node previ al node actual per saber per on ha arribat la influència d'aquesta seu.

D'aquesta manera, quan s'hagin executat tots els *Dijkstra* per totes les seus, cada node del graf (sigui seu o no), tindrà una llista de seus amb unes distàncies i nodes previs i això és la influència que els hi arriba de cada seu. Així doncs, s'ha propagat la influència per la xarxa de carreteres. A partir d'ara, el problema s'ha convertit en un problema més local, perquè quan s'hagi de calcular una aresta concreta del mapa no serà necessari buscar cada seu, només caldrà revisar la llista de seus que tenen els nodes que formen l'aresta.

8.4.5 Calcular les puntuacions dels punts màxims i mínims

Aquest és l'últim punt on s'utilitzen algoritmes matemàtics. És el punt més important i on s'ha utilitzat una fórmula nova que ha millorat substancialment el rendiment de

l'aplicació.

Gràcies al punt anterior, i a la informació guardada durant la propagació de la influència, s'ha pogut fer aquest punt de la manera que s'ha fet.

En un principi, s'havia pensat de discretitzar les arestes i calcular en paral·lel la influència de cada punt de la xarxa de carreteres, això era una gran càrrega i faria que l'aplicació consumís molta memòria i el rendiment seria molt pitjor.

Com s'ha comentat anteriorment, amb la propagació de la influència el càlcul de la influència exercida per les seus existents als nodes del graf es pugui fer amb temps constant. Amb un exemple s'entendrà millor:

Sigui A l'aresta entre els nodes P i Q. Tant en el punt P com en el punt Q hi ha una llista de seus que tenen la influència sobre aquests nodes respectivament, amb la distància entre P i Q i aquests nodes. Això fa que es pugui trobar fàcilment la influència de les seus existents sobre els nodes P i a Q. Per exemple, si en el node P hi ha guardat l'id d'una catedral que hi exerceix una influència 10 i els de dos hotels que hi exerceixen influència 2 i 3 respectivament, la influència resultant serà 5 ja que la catedral té influència positiva i la dels hotels és negativa. Cal tenir en compte que la influència exercida per una seu amb influència màxima W a sbore d'un punt a distància d de la seu és 0 si d>W i d-W altrament, i que la nova seu serà sempre un hotel, per tant els hotels seran competidors i la seva influència restarà.

Això es pot fer de la mateixa manera pel node Q. Tot i que a continuació es veurà que no és necessari.

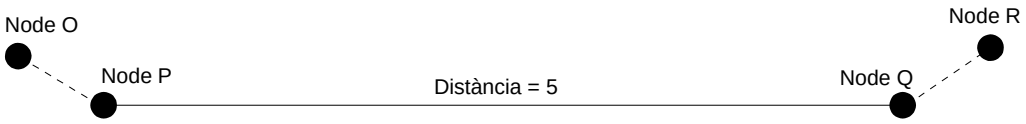
El que es vol trobar a l'aresta A són els punts on la influència d'una seu que influeixi al node P o a Q comenci o acabi. Aquests punts rebran a partir d'ara el nom de punts de canvi PC, ja que és en els nodes o en aquests punts on hi pot haver el màxim que s'està buscant. Per tant es troben els punts del graf on hi ha canvis en les seus que hi exerceixen influència. En els trams en què les seus no canvien el valor d'influència es pot calcular fàcilment per interpolació lineal.

8.4.5.1 Com funciona l'algoritme

El primer que s'ha de fer és calcular la influència del node P, però per poder fer-ho s'ha de calcular la influència inicial de cada seu i restar-la de la distància entre el node i la seu. Així s'obté la influència que falta per exhaurir-se.

Per tant, tot i que *Dijkstra* calculi distàncies el que es guarda als nodes P i Q serà una llista de seus amb la influència restant, enlloc de la distància de la seu al node concret, i el node previ del camí, després es veurà perquè cal guardar-lo. Veure el dibuix (Dibuix 1) i la taula (Taula 1) següents.

A continuació es continua l'explicació intercalant-hi un exemple en el que la distància entre el node P i Q és de 5. S'aprofita aquest exemple per explicar com s'obtenen tots els punts de canvi així com el seu valor d'influència a partir de la influència del darrer punt de canvi analitzat.



Dibuix 1: Aresta de P a Q

Node P		
Seu	**Influència restant**	**Node Previ**
Catedral (A)	2	Q
Hotel Melià (B)	7	O
Hotel Eurostars (C)	3	O
Cul de la Lleona (D)	2	O

Node Q		
Seu	**Influència restant**	**Node Previ**
Catedral (A)	7	R
Hotel Melià (B)	2	P
Església Sant Feliu (E)	4	R
Cul de la lleona (F)	3	R

Taula 1: Llista de seus en el node P i Q

Com es pot veure, hi ha 4 seus que la seva influència arriba al node P i n'hi ha 3 que arriben al node Q.

Algunes d'aquestes seus són atraccions que tenen una influència positiva (Catedral, Cul de la Lleona i Església de Sant Feliu) i la resta són hotels que tenen una influència negativa.

A partir d'ara, com ja s'ha dit abans, es suposarà que els càlculs es fan amb la influència restant de cada seu i no amb la distància entre la seu i el node P.

El node inicial d'aquesta aresta és P i el primer que s'ha de fer és calcular la puntuació total d'aquest node. Es fa d'aquesta manera:

$PN = \sum SI - \sum SC$ on:

- SI són les influències restants de totes les seus d'interès (esglésies, monuments, etc).
- SC són les influències restants de totes les seus competidores (hotels).

Ara s'ha obtingut la puntuació del node (PN) que s'utilitzarà de base.

El resultat és: (A+D)-(B+C)=>(2+2)-(7+3)=**-6**

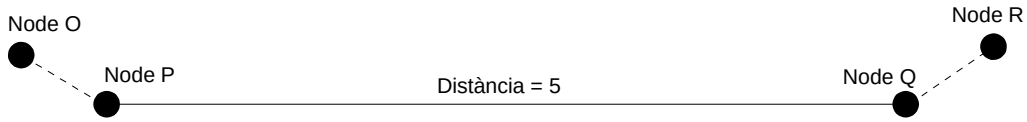
El segon pas és col·locar totes les seus de Q a P perquè l'algoritme recórrer l'aresta a partir de P i es necessiten totes les seus de l'aresta, ja que un punt de canvi pot ser provocat per l'inici de la influència d'una seu que arribava a Q però no a P. Hi ha varis casos:

1. Quan la seu existeix a P i a Q, a l'exemple hi ha la Catedral, l'hotel Melià i el Cul de la Lleona. Quan això passa:

1. Si el node previ és P o Q, no es passaran les seus de P a Q ja que les seus segueixen el mateix recorregut i s'estarien duplicant les influències. Per tant, com es pot veure a la imatge anterior la Catedral i l'hotel Melià segueixen el mateix camí perquè els nodes previs són P o Q.
2. Si el node previ no és P o Q, significa que la seu arriba a P i a Q per camins diferents, per tant, la influència pot canviar dins l'aresta, per tant sí s'ha de posar el node de Q a P. Això passa amb el Cul de la Lleona. Aquest és un cas complex que es tractarà per separat en el següent apartat 8.4.6.Cas especial del Cul de la Lleona més endavant per no causar confusió en l'algoritme.

2. Quan una seu de Q no existeix a P, aquesta seu s'haurà de posar a P, però totes les seus de Q que es posen a P se'ls hi ha de restar la distància de l'aresta. Això significa que seran seus amb distància negativa sempre, si no fos negativa significaria que la seu també arriba a P, però no és el cas. Que sigui negativa simplement significa que és un node que no està a P però si que afecta a punts de dins de l'aresta.

Quedaria d'aquesta manera:



Dibuix 2: Aresta de P a Q

Node P		
Seu	**Influència restant**	**Node Previ**
Catedral (A)	2	Q
Hotel Melià (B)	7	O
Hotel Eurostars (C)	3	O
Cul de la Lleona (D)	2	O
Església Sant Feliu (E)	4-5 = -1	R
Cul de la lleona (F)	3-5 = -2	R

Taula 2: Seus del node P amb les del node Q i amb la influència restant

El tercer pas és crear una nova estructura auxiliar que servirà per poder fer el nou algoritme de càlcul d'influència al llarg de l'aresta. Aquesta estructura ha de contenir cada seu amb un 1 positiu o un 1 negatiu. Després aquests 1 es sumaran per obtenir el multiplicador per calcular cada punt. Més endavant s'explicarà millor.

- L'1 positiu significa que la influència de la seu augmenta mentre es recórrer l'aresta.

- L'1 negatius significa que la influència disminueix.

Per saber si augmenta o disminueix hi ha varis paràmetres que cal tenir en compte:

- **Tipologia:** Si la seu és un competidor o un punt d'interès.
- **Node previ:** Si el node previ és el node Q o no.
- **Signe:** Si la seu és de P o de Q, si és negatiu significa que és una seu de Q.

Amb aquests tres paràmetres es poden fer els càlculs necessaris:

1. Si la influència que queda és positiva:

   $C \wedge (prev = Q) = -1$
   $C \wedge (prev != Q) = 1$
   $I \wedge (prev = Q) = 1$
   $I \wedge (prev != Q) = -1$

2. Si la influència que queda és negativa (és una seu de Q):

   $C = -1$
   $I = 1$

Aquest càlcul es fa per cada una de les seus de la taula i es desa en una estructura auxiliar. A part, també es fa la suma dels 1 dels nodes amb influència positiva.

Seu	Uns
A	1
B	1
C	1
D	-1
E	1
F	1

Taula 3: Taula d'uns

La suma és: A+B+C+D=**2**

El quart pas és ordenar la taula amb la influència més petita amb valor absolut, per saber amb quin ordre les seus provocaran els diferents punts de canvi.

Node P		
Seu	**Influència restant**	**Node Previ**
Església Sant Feliu (E)	-1	R
Catedral (A)	2	Q
Cul de la Lleona (D)	2	O
Cul de la lleona (F)	-2	R
Hotel Eurostars (C)	3	O
Hotel Melià (B)	7	O

Taula 4: Seus del node P ordenades per la influència restant

El cinquè pas és situar el pròxim punt de canvi, és a dir moure el punt que volem calcular la puntuació a la distància donada per la influència restant més petita amb valor absolut. Per tant, arribarà a la distància P+1 per l'aresta, perquè es pararà en el punt on comença la influència de l'Església Sant Feliu, el canvi del valor d'influència entre P i P+1 és continu i s'obté per interpolació lineal dels valors d'influència de P i P+1. Així doncs només s'ha mogut 1 de distància.

Ara s'ha d'aplicar la fórmula clau de l'algoritme:

$PN + ƛ \sum \mu f$ on:

- PN: és la puntuació del node P, en el darrer punt calculat, a l'exemple és **-6**.
- ƛ: és la distància entre P i el punt actual. En el cas de l'exemple, la distància és **1**.
- μf: són els 1, per tant la suma de l'exemple és **2**.

Per tant, -6+1*2=-**4**. Amb aquesta fórmula el resultat és -4, ara es comprovarà fent el càlcul normal. Per fer-ho s'ha de sumar la influència de cada seu i restar o sumar 1 perquè la distància recorreguda ha estat de 1. I la suma i resta ja s'ha explicat anteriorment i a més a més només els positius, ja que els negatius indiquen que encara no s'ha arribat a la seva zona d'influència. Així doncs:

A-B-C+D => (2+1)+(-7+1)+(-3+1)+(2-1) => 3-6-2+1=**-4**

Com es pot comprovar el resultat és el mateix però de la primera manera és més senzill gràcies a l'estructura auxiliar dels 1s.

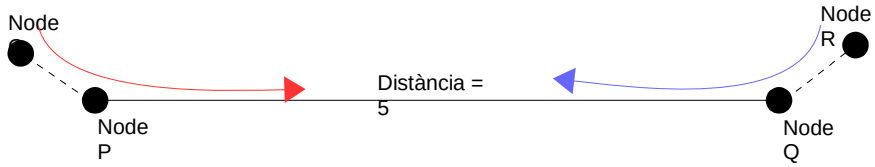
El sisè pas és el d'afegir o treure de la suma de 1s el (μf) del punt on s'ha arribat. En el cas de l'exemple s'ha d'afegir el valor de la seu E. En aquest cas s'ha d'afegir perquè era una seu amb influència negativa que indicava que era una seu de Q. Per tant, a partir d'aquest punt, aquesta seu ja té influència sobre l'aresta.
Si es busca a l'estructura auxiliar, es pot comprovar que és un 1 positiu que s'ha d'afegir a la suma, per tant, ara serà de **3**.

Finalment, el sisè pas és tornar al quart pas, amb el següent node amb menys

influència. I això s'ha de repetir fins que no quedin seus a la llista.

8.4.6 Cas especial del Cul de la Lleona

Aquest cas com s'ha comentat en l'apartat anterior és quan hi ha una seu que influeix en el node P i en el node Q però els nodes previs són diferents, per tant, són camins diferents.

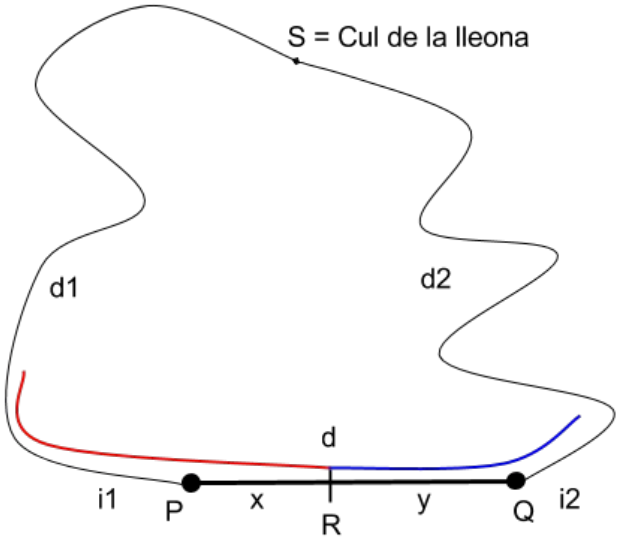


Dibuix 3: Aresta de P a Q amb la influència del Cul de la Lleona que arriba per dos camins diferents i talla a l'aresta PQ

Quan això passa s'ha de trobar quin dels dos camins dóna la màxima influència a la seu en cada punt de l'aresta PQ. Poden passar varis casos que comporten fer accions diferents:

1. Els dos camins no s'intersequen. Això significa que la influència que prové de P i la que prové de Q no es toquen dins l'aresta PQ i per tant no hi ha cap conflicte. Simplement, es tractarà com la resta de seus. Al principi tindrà influència el camí que passa per P fins que s'acabi, més endavant arribarà un punt on començarà a tenir influència pel camí que passa per Q. Cal tenir-ho en compte alhora de afegir i treure els 1s a la taula de 1s. Però no suposa un comportament especial.
2. Els dos camins s'intersequen. Significa que les influències dels dos camins es solapen i evidentment no es pot comptar la influència dels dos camins a la vegada en els punts on hi siguin. Per tant, s'ha de determinar on acabarà una influència i on començarà l'altre. Aquest punt serà el punt on la influència que passa per P deixi de ser més gran que la que passa per Q. Cal dir que sempre la influència que passi per P serà major que la que passa per Q a l'inici de l'aresta, perquè si això no fos així, en el procés de *Dijkstra* no s'hauria marcat el camí de P com a camí amb més influència per la seu en qüestió. És possible que la influència que passa per Q pugui arribar a P, però amb una influència menor que la de l'altre camí. Això cal recordar-ho, perquè es dóna per suposat que això és així i és una evidència.

   Per fer-ho més entenedor s'ha fet un dibuix que ho esquematitza (Dibuix 4).



Dibuix 4: Mapa amb influències d'una mateixa seu que s'intersequen

El que s'està buscant és el punt on els dos camins tenen la mateixa influència i segur que hi és si arriben a creuar-se. Per obtenir aquest punt s'ha de resoldre un sistema d'equacions:

– $i1-x=i2\times y$ Els dos camins tenen la mateixa influència.

– $x+y=d$ Estan dins els segment.

i1: la influència que arriba a P de la seu S.

i2: la influència que arriba a Q de la seu S.

R: el punt on les influències dels dos camins és igual. És el punt que s'està buscant.

x: la distància entre P i R.

y: la distància entre Q i R.

d: distància entre P i Q

Si es soluciona el sistema dóna s'aconsegueix una equació per x, així que ja se sap la distància i per tant, es pot saber R.

$$x=\frac{d+i1-i2}{2}$$

Una vegada s'ha trobat el punt, aquest punt passa a ser un altre punt de canvi a tenir en compte i quan sigui el seu moment en comptes d'eliminar l'1 de la taula de 1s se li haurà de canviar el signe i agafar els valors del camí que passa per Q i no el de P.

8.4.7 Visualitzar el mapa de valor d'influència resultant

Per la visualització del resultat s'ha utilitzat *OpenGL* per calcular els valors d'influència entre els punts de canvi de forma automàtica, aprofitant la interpolació lineal del color que fa aquest llenguatge. La visualització desenvolupada és senzilla però s'ha fet pensant en que és una primera versió del producte i fent que sigui fàcil de veure i

entendre el resultat que en el punt actual del projecte és el que interessa.

La visualització que s'ha desenvolupat és el mateix mapa d'entrada que l'usuari ha introduït. De fet, no és el mapa de l'usuari exactament, perquè en el mapa inicial se li ha aplicat l'algoritme de simplificació que modifica la xarxa de carreteres del mapa. Per tant, no és el mateix mapa, però sí que és semblant això fa que l'usuari estigui familiaritzat amb el que veu i li sigui senzill entendre el resultat. També es mostren les seus. Les seus positives són rectangles de color groc i les seus negatives de color blau, són colors diferents als que mostraran el resultat, s'han utilitzat aquests colors per crear un gran contrast i veure ràpidament les seus i el resultat, així a simple vista es poden extreure conclusions.

Fins ara només és una representació del mapa introduït per l'usuari, ara falta pintar amb colors les carreteres per mostrar el resultat dels càlculs que s'han explicat en l'apartat anterior. Els colors escollits han estat el verd i el vermell per mostrar les zones del mapa on és bo construir una seu o dolent respectivament. S'ha fet amb aquests dos colors perquè són els colors que representen el positiu i el negatiu i això pels éssers humans és fàcil d'entendre. Tot hi que a nivell de visualització es recomana utilitzar colors semblants amb diferents tons, per exemple una escala de verd clar fins a verd fosc, al final no s'ha fet perquè fent una petita enquesta es va concloure que s'entenia millor el sistema del verd i el vermell.

El problema d'utilitzar dos colors és que en el punt on es passa d'un color a un altre es veu com un canvi molt brusc quan realment a nivell de valor poden ser molt semblants. El que s'ha fet és pintar els valors positius en verd i els negatius en vermell. Però evidentment, tenen una escala que indica el valor positiu o negatiu respectivament, això fa que si el verd és molt intens el valor és més elevat, per tant, millor. Si el verd és més fosc, el valor és més baix però positiu. El mateix passa amb el vermell, quan el vermell és clar el valor és poc negatiu fins a arribar a un vermell molt fosc que significa que és una zona molt dolenta per construir una nova seu.

Per determinar l'escala del verd i del vermell s'han utilitzat uns càlculs per limitar els valors perquè no puguin arribar a ser blancs o negres, a més a més s'ha normalitzat el valor perquè *l'OpenGL* ho necessita.

Per fer-ho s'ha utilitzat aquesta fórmula que el que fa és modificar el valor perquè estigui dins el rang de valors possible.

$$c = \frac{score - minScore}{maxScore - minScore} \times (maxColor - minColor) + minColor$$

on:

- score: és el valor que volem transformar.
- minScore: és el valor mínim que pot tenir la puntuació. Aquest valor quan el *score* és positiu (verd), és 0.
- maxScore: és el valor màxim que pot tenir la puntuació. Aquest valor quan el score és negatiu (vermell), és 0.
- maxColor: és 255. El rang de color va de 0 a 255, és el rang que té *OpenGL*.
- minColor: és 50. S'ha decidit posar 50 per evitar els negres, si el color baixa de 50

es veu molt negre i no es pot distingir bé quin color és, així que el rang que s'utilitza en aquest projecte va de 50 a 255.

Gràcies a la utilització d'OpenGL, la visualització d'aquest projecte té un gran factor de millora ja que s'ha fet una visualització simple per poder mostrar els resultats. Però OpenGL proporciona una gran quantitat d'utilitats que poden fer molt més atractiva l'aplicació. Per exemple, pintar text, o seleccionar punts directament a sobre de la imatge pintada o tenir un menú amb botons, etc. En el capítol 12.Treball futur es desenvoluparà més aquesta idea.

Capítol 9

Implementació i proves

9 Implementació i proves

El projecte s'ha implementat tot el que s'ha esmentat utilitzant C++ i *OpenGL,* en el capítol 10.Implantació i resultats es mostraran els resultats. S'han utilitzat jocs de proves petits d'estudi per validar la implementació i la solució proposada i altres amb models més complets com es podrà observar.

A continuació es detallaran alguns dels aspectes remarcables trobats al llarg de la implementació que s'ha dut a terme i de les proves realitzades.

9.1 Canvi de llenguatge

El projecte es va començar amb *Python* i C++, finalment es va suprimir el codi de *Python*.

Problema

Al principi es va utilitzar *Python* per fer la part de recollida lectura del mapa (*osm*) i desar-lo en un graf. Aquest graf era un *JSON* que després s'enviava al codi C++ i des d'allà es feien els càlculs pertinents i es mostrava el resultat.

Aquesta decisió es va prendre perquè *Python* és un llenguatge molt més senzill i pràctic. Era la manera més senzilla i ràpida de desenvolupar el projecte. Però això va esdevenir un problema ja que per executar el projecte sencer, primer s'havia d'executar la part de *Python* perquè generés el *JSON* i després s'havia d'enviar per paràmetre a la part de C++. Això implicava que entre les dues parts hi havia d'haver una comunicació que no es va arribar a fer mai, es feia manual.

Solució

Es van pensar dues solucions:

1. Utilitzar *Cython [39]*: aquesta va ser la primera solució que es va pensar. *Cython* és un llenguatge que barreja *Python* i C (C++). Es poden fer mòduls de *Python* amb C++ que són més eficients, però el programa principal seria *Python*. La majoria de mòduls que té *Python* estan desenvolupats amb C++ per ser més eficients, *Cython* et permet crear els teus i compilar-los. Es van demorar varis dies investigant i provant *Cython*, però finalment es va decidir no utilitzar-lo, tot i les avantatges que s'han esmentat i les ganes d'aprendre més de *Python*. La raó s'explica en el següent punt.
2. Eliminar *Python* i *Cython*: Aquesta solució es va donar mentre es provava *Cython*. Es va veure la manera de no haver d'utilitzar ni *Python* ni *Cython*, simplement C++. Això feia que el projecte fos més eficient i que no s'havien d'utilitzar varis llenguatges o llenguatges amb la barreja dels dos llenguatges. A més a més, aquest projecte es va plantejar per refrescar i aprendre C++. Així doncs, això és el que s'ha fet. Tot el projecte està desenvolupat en C++. La part inicial del projecte, la lectura i el desament del mapa en una estructura es va eliminar de *Python* i es va passar a C++, això va suposar una gran millora ja que no hi havia cap mena de comunicació entre dos llenguatges (o dos parts diferents) i d'altre banda no

s'havia de duplicar la feina. És clar que quan hi havia *Python*, es feia una lectura i després es desava en una estructura (un diccionari de *Python*) i s'enviava a través d'un *JSON* que després havia de llegir C++ i tornar a desar el que havia llegit en una estructura de C++ (map).

9.2 Canvi de JSON a XML

Aquest punt està relacionat amb l'anterior. La manera de llegir el mapa es va canviar de *JSON* a *XML*.

Problema

Com s'ha explicat en el punt anterior, es va començar el projecte amb *Python* i es va fer la part de lectura del mapa i es guardava en un *JSON* que després s'enviava a l'entrada del C++. Això va fer que es necessités llegir un *JSON* amb C++ i guardar el contingut en un map. *JSON* és un tipus de fitxer que s'utilitza per enviar dades entre webs o entre pars de webs i és molt utilitzat, de fet, està superant amb claredat el *XML*. Totes les noves tecnologies web utilitzen *JSON* sobretot perquè la majoria són *Javascript*.

C++ té llibreries per llegir *JSON* però són complicades d'instal·lar i es van perdre hores instal·lant i configurant aquestes llibreries. Com a conseqüència dels problemes que es van tenir es va pensar en modificar el projecte i aquesta va ser una de les raons per la que es va eliminar la part de *Python*.

Solució

La solució va ser eliminar *Python*, això va fer que el propi C++ llegís el fitxer *osm* (*XML*). Es va instal·lar el *PugiXML*, com s'ha explicat al capítol 7. I això va suposar que directament es llegís un *XML* i es guardés la informació a l'estructura del C++. Evitant doncs, haver de llegir *XML* amb *Python*, guardar-ho en un diccionari de *Python*, convertir-ho en *JSON*, enviar-lo a C++, que el C++ llegís el *JSON* i finalment guardés la informació del *JSON* en una estructura de C++.

9.3 Utilització de classes i herència

El projecte va començar amb un simple *main.* Fins que va arribar un dels canvis en el codi més importants del projecte. On es va dividir en varies classes on heretaven unes de les altres.

Problema

El projecte creixia i no podia estar tot en el mateix arxiu. S'havia d'estructurar per poder entendre'l millor i que fos més senzill de programar.

Solució

La solució passava per crear classes. Es van crear varies classes:

- Graph: Contenia els mètodes per manipular el graf. Era un conjunt de nodes.
- BaseNode: Era la super classe, de la que heretaven Node i TourismNode. Era una classe abstracte que tenia mètodes comuns pels dos tipus de nodes que hi havia.
- Node: heretava de BaseNode i eren els nodes normals. Els que no eren seus.

- TourismNode: heretava de BaseNode i eren els nodes que eren seus.

Cal dir que aquesta estructura nova es va programar en una branca nova del git. Això va permetre canviar a l'estructura anterior del problema i tornar a aquesta molt fàcilment. Va ser una gran decisió perquè es va haver de retrocedir varies vegades.

Això va fer que el programa fos molt millor degut a que tenia una estructura adequada. Però mai va arribar a funcionar aquesta estructura perquè hi havia herències creuades que no es podien programar. Així doncs, l'estructura va evolucionar i va quedar així:

- Graph: Contenia els mètodes per manipular el graf. Era un conjunt de nodes.
- Node: Era l'estructura atòmica. La més petita, contenia els mètodes i atributs necessaris per poder gestionar els nodes normals i les seus.

9.4 Classe Point

Es va afegir una nova classe a la part final del desenvolupament. Aquesta classe es va anomenar Point.

Problema

En l'estat tant avançat del codi es va decidir fer una nova classe perquè es veia clarament que era necessari fer un *refactor* per simplificar algunes estructures.

Moltes de les estructures del projecte tenien tuples amb les coordenades i la puntuació, això feia que fos complicat el tractament amb aquestes estructures.

Solució

Es va crear una classe amb tres atributs: x, y i *score*. Això va fer possible la simplificació de varies estructures i extreure uns atributs i mètodes de Node.

Node hereta de Point ja que l'utilitza perquè Point és una part de Node.

9.5 Canvi de coordenades

El fitxer *osm* conté les coordenades amb el format latitud i longitud. Però fer operacions sobre aquest tipus de coordenades és complicat.

Problema

Les coordenades en el format latitud longitud tenen en compte la curvatura de la terra i és complicat fer operacions per obtenir les distàncies [40].

Solució

Transformar les coordenades en el format *x y*. Aquest tipus de coordenades són en el pla, és més senzill fer operacions i calcular les distàncies. A més a més, els mapes d'aquest projecte són de ciutats així que la transformació de les coordenades de 3D a 2D no fa perdre massa informació i realment és un canvi positiu en la simplificació de càlculs [41] [42].

Per modificar les coordenades de *lat long* a *x y* s'ha utilitzat la projecció de *Mercator* que és la projecció amb la que es fan els mapes que estem acostumats a veure.

9.6 Id dels nodes

Problema

Quan es van provar mapes de ciutats reals. El projecte va donar errors. L'error principal va ser per culpa dels ids dels nodes perquè els ids que hi havia en el fitxer *osm* eren molt grans i no hi cabien en un *int.*

Solució

Canviar el tipus de l'atribut que desava l'id del node a *long.*

9.7 Pes normalitzat

El pes de les seus s'havia de normalitzar per tenir un control dels valors.

Problema

No hi havia cap control sobre aquest valor. Això feia que l'usuari podia posar el valor que volgués.

Solució

Posar un rang de valors possibles. El rang és de 1 a 10, a més a més, aquest valor es va normalitzar per facilitar els càlculs amb les distàncies ja que les distàncies també ho estan. D'aquesta manera no cal revisar si estan en el mateix format, sempre està tot normalitzat i així es pot operar sobre aquests valors.

9.8 Mostrar tot el mapa

Problema

Alhora de mostrar el mapa no es mostrava tota la xarxa de carreteres sencera. El motiu era perquè només es mostraven les carreteres que tenien influència d'alguna seu, les que no els hi arribava cap influència no es mostraven.

Solució

Primer es pinta una capa amb tota la xarxa de carreteres. Després es pinta la capa de les influències.

9.9 Moviment per les arestes

Com s'ha explicat a l'apartat 8.3.5.1.Com funciona l'algoritme poden haver una sèrie de punts al mig d'una aresta que són candidats a ser punts màxims. S'ha de recórrer l'aresta per arribar a aquests punts.

Problema

Una aresta la podem tractar com un vector i per tant, es pot recórrer, però es necessita una fórmula perquè pot tenir la direcció que sigui. En el cas del projecte es sabia el punt inicial i el final del vector. Però el problema era que donat un punt inicial del vector i una distància per recórrer s'havia de saber la coordenada resultant per poder mostrar-ho en el mapa resultant.

Solució

Per solucionar-ho s'ha aplicat la següent fórmula:

$x = inicialX + (\frac{dist}{longAresta}) \times (finalX - inicialX)$, on:

- x: és la coordenada x del nou punt que s'està buscant.
- InicialX: és la coordenada X del node inicial de l'aresta.
- finalX: és la coordenada X del node final de l'aresta.
- dist: és la distància que es vol recórrer de l'aresta (des del node inicial).
- longAresta: és la longitud de l'aresta.

Aquesta fórmula és per la coordenada X, s'ha de fer el mateix per la coordenada Y:

$y = inicialY + (\frac{dist}{longAresta}) \times (finalY - inicialY)$

9.10 Canvi del càlcul de la propagació de la influència 1

Com s'ha comentat en l'apartat anterior la propagació de la influència és un dels càlculs més complexos d'aquest projecte i a patit varies modificacions.

Problema

En un principi el càlcul de la influència es va fer recorrent totes les arestes i revisant el valor d'influència de punts concrets marcats per una distància. O sigui que es buscava el valor cada certa distància. Aquesta distància era relativament petita com per poder buscar el valor d'influència varies vegades a cada aresta. Això provocava una càrrega important ja que té un cost molt elevat. La distància havia de ser petita per no perdre cap possible punt màxim en el mapa.

Solució

Per poder proposar la solució, primer cal dir que es va trobar que les influències eren continues. Mentre s'avança per l'aresta la influència augmenta, disminueix o es manté de manera continua. Això passa perquè hi ha les mateixes seus influint en l'aresta i si s'avança per l'aresta, s'avança la mateixa distància per totes les seus iguals. Per tant, si se sap quin valor té en el punt inicial de l'aresta es pot saber el valor que tindrà en una part de l'aresta, concretament, fins al punt on una seu deixi de tenir influència o una seu nova comenci a influir. Aquests punts, s'anomenen punts de canvi.

La solució va canviar tot el projecte ja que ara el projecte es basaria en punts de canvi. Com s'ha explicat en el capítol anterior, els punts de canvi són aquells punts candidats a ser el punt màxim o mínim. Així doncs, només cal revisar el valor de la influència en aquests punts. Això va fer que el problema es reduís moltíssim. Ara el cost era molt menor ja que hi ha molts menys punts de canvi que no pas punts separats per una distància petita per cobrir tot el mapa.

Aquesta millora és la més important del projecte perquè el va transformar en un projecte molt més eficient i molt més simple del que es va pensar en un principi. Ja que aquesta solució va suposar que ja no es necessités paral·lelitzar.

9.11 Canvi del càlcul de la propagació de la influència 2

Tot i la modificació del càlcul que s'ha explicat a l'apartat anterior es podia millorar encara més l'eficiència.

Problema

A cada punt de canvi es calculava el valor d'influència que tenia. Això suposava recórrer totes les seus que influïen en el punt de canvi. Era un procés repetitiu que no té massa cost perquè un punt de canvi té poques seus però si hi ha molts punts de canvi pot ser bastant costós i sobretot sabent que es pot evitar.

Solució

Com en l'anterior apartat, aquesta solució es pot aplicar perquè se sap que les influències són lineals.

Es calcula el valor de la influència del node inicial de l'aresta recorrent totes les seus. Però no només això, sinó que cal apuntar-se en una estructura auxiliar quines seus participen en el valor i saber si augmenten o disminueixen la influència mentre s'avança per l'aresta. D'aquesta manera, el càlcul del proper punt de canvi és automàtic ja que té el valor del punt de canvi anterior, la distància recorreguda i el comportament de cada seu.

Això fa que s'evitin molts bucles innecessaris i per tant, es millori el cost del càlcul. Hi ha un procediment més complex en el càlcul del primer valor d'influència però després és automàtic per la resta.

Està explicat amb detall en el capítol anterior.

9.12 Seus zonals

Problema

Durant el desenvolupament del projecte, hi ha hagut casos de seus que no són un simple punt en un mapa i afecten a un punt concret de la xarxa de carreteres. Hi ha seus grans que ocupen una zona en el mapa i tenen varies entrades a diferents punts de la xarxa de carreteres. Per exemple, estadis, centres comercials, etc.

Solució

Habilitar l'opció de crear seus zonals. On es pugui marcar quin tram de la xarxa de carreteres hi ha la seu. En comptes de marcar un simple punt, que es puguin marcar amb segments. Així doncs, en qualsevol punt del segment la distància seria 0 i per tant, la influència podria arribar més lluny sense importar en quina direcció estigués el node influït. El problema actual és que si el punt de la seu està en un carrer, els nodes que estiguin en carrers paral·lels la influència que els hi arriba és menor perquè la influència ha de recórrer un camí fins a poder accedir al carrer paral·lel. Si aquesta seu se la marqués com a zona i tingués marcada els dos carrers paral·lels perquè resulta que té dos sortides, una a cada carrer, els dos carrers tindrien la mateixa influència i seria més real.

Capítol 10

Implantació i resultats

10 Implantació i resultats

En aquest capítol s'explicarà com s'ha fet la implantació del projecte, la màquina i procediments que s'han seguit per aconseguir-ho. També s'exposaran els resultats obtinguts.

10.1 Implantació

A l'apartat 7.1.Maquinari ja s'ha comentat el maquinari i el sistema operatiu que s'ha escollit per desenvolupar i també per implementar aquest projecte. Al ser un projecte petit que no consumeix massa recursos no ha estat necessari utilitzar un servidor.

10.1.1 Compilació

El projecte està desenvolupat en C++, per tant, s'ha de compilar. Aquest procés s'ha fet a través de *Cmake.* És un programari que permet compilar els programes en C++ per qualsevol plataforma i inclou les dependències, etc. És una capa que fa més senzilla la configuració de la compilació. Utilitza el compilador de C++ (g++) i el *make.*

El *CMake* s'ha utilitzat perquè l'IDE escollit (*CLion*) l'utilitza. Degut al desconeixement inicial, es va complicar la compilació, sobretot quan es va dividir el projecte en diferents directoris i es van afegir noves llibreries con la del *pugiXML* o *OpenGL*

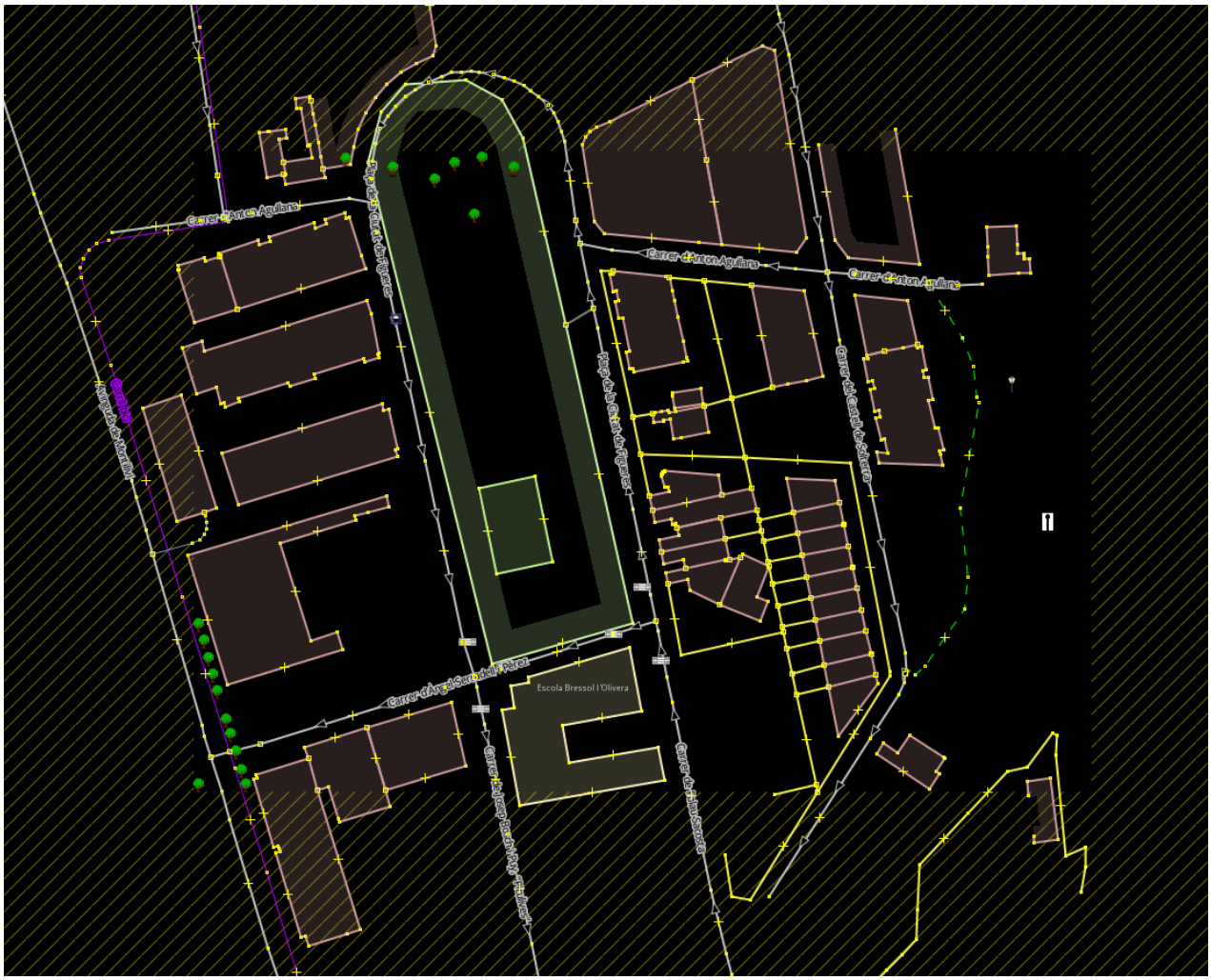
Per configurar el *Cmake* s'utilitza un fitxer anomenat *CMakeLists.txt* en aquest fitxer s'especifiquen els paràmetres amb els quals es vol compilar el programa. Amb quins *flags*, els directoris que tenen codi o el que té *includes* o *llibreries*.

Un dels canvis que es van haver de fer va ser quan es va introduir *OpenGL* al projecte, ja que calia indicar al compilador que s'estava utilitzant *OpenGL* i altres llibreries perifèriques necessàries per mostrar el mapa resultant [43] [44] [45].

10.2 Resultats

10.2.1 Neteja del mapa

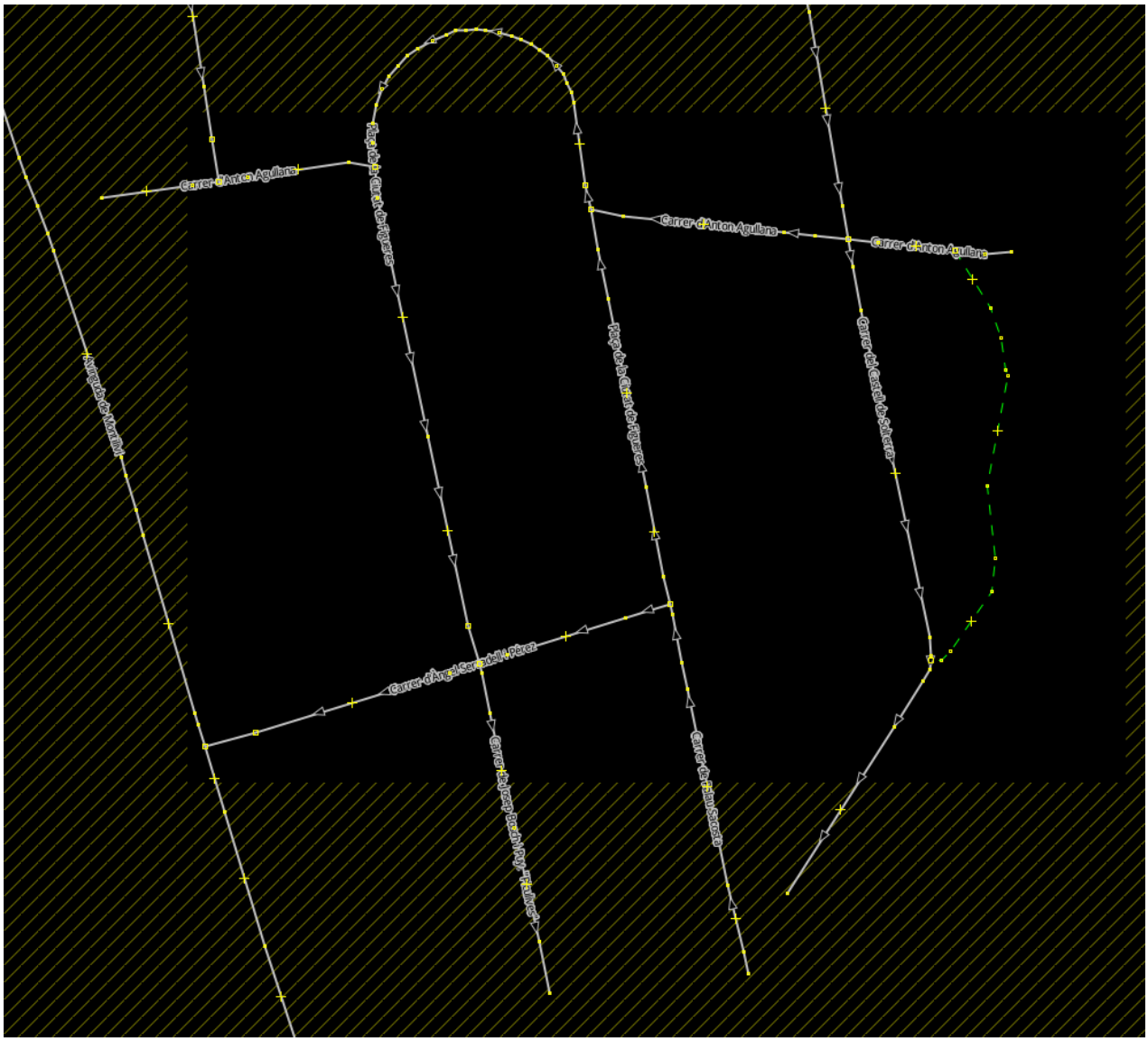
A la següent il·lustració (Il·lustració 18) es pot observar el mapa d'una part del barri de Montilivi, concretament la plaça Ciutat de Figueres, aquesta il·lustració és del software *JOSM* que ja s'ha comentat a l'apartat 7.4.2.Editors de mapes Open Street Maps. Es poden veure molts detalls com edificis, arbres, semàfors, etc.



Il·lustració 18: Mapa de la plaça Ciutat de Figueres sense netejar. Visualitzat des del JOSM

Com es pot observar, hi ha molta informació que no és útil en aquest projecte. Cal fer la neteja amb el software *osmfilter* com s'ha esmentat a l'apartat 7.4.3.Netejadors de mapes osm.

A la següent il·lustració (Il·lustració 19) es pot comprovar com queda el mapa després d'haver-lo netejat.



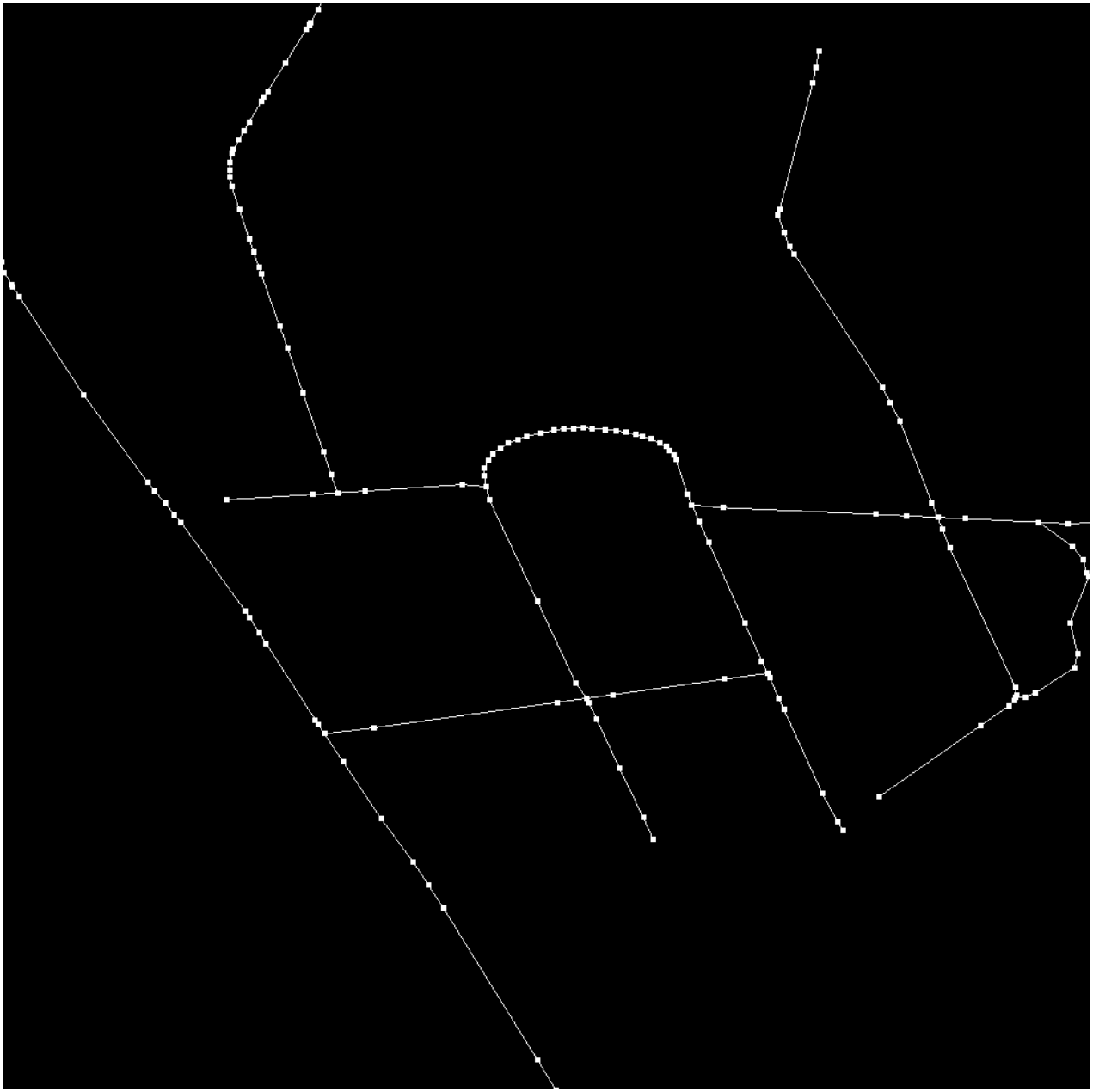
Il·lustració 19: Mapa de la plaça Ciutat de Figueres netejat amb l'osmfilter. Visualitzat des del JOSM

Comparant les dos il·lustracions es pot veure que la neteja del mapa ha estat important. Després de la neteja només hi ha la xarxa de carreteres que és el que interessa en aquest projecte. S'ha eliminat qualsevol element que no siguin carreteres o camins. En aquest mapa d'exemple no hi ha seus, però en cas que hi haguessin seus tampoc s'haurien eliminat.

Ara que ja s'ha obtingut el mapa netejat seria hora d'introduir les seus.

10.2.2 Simplificació

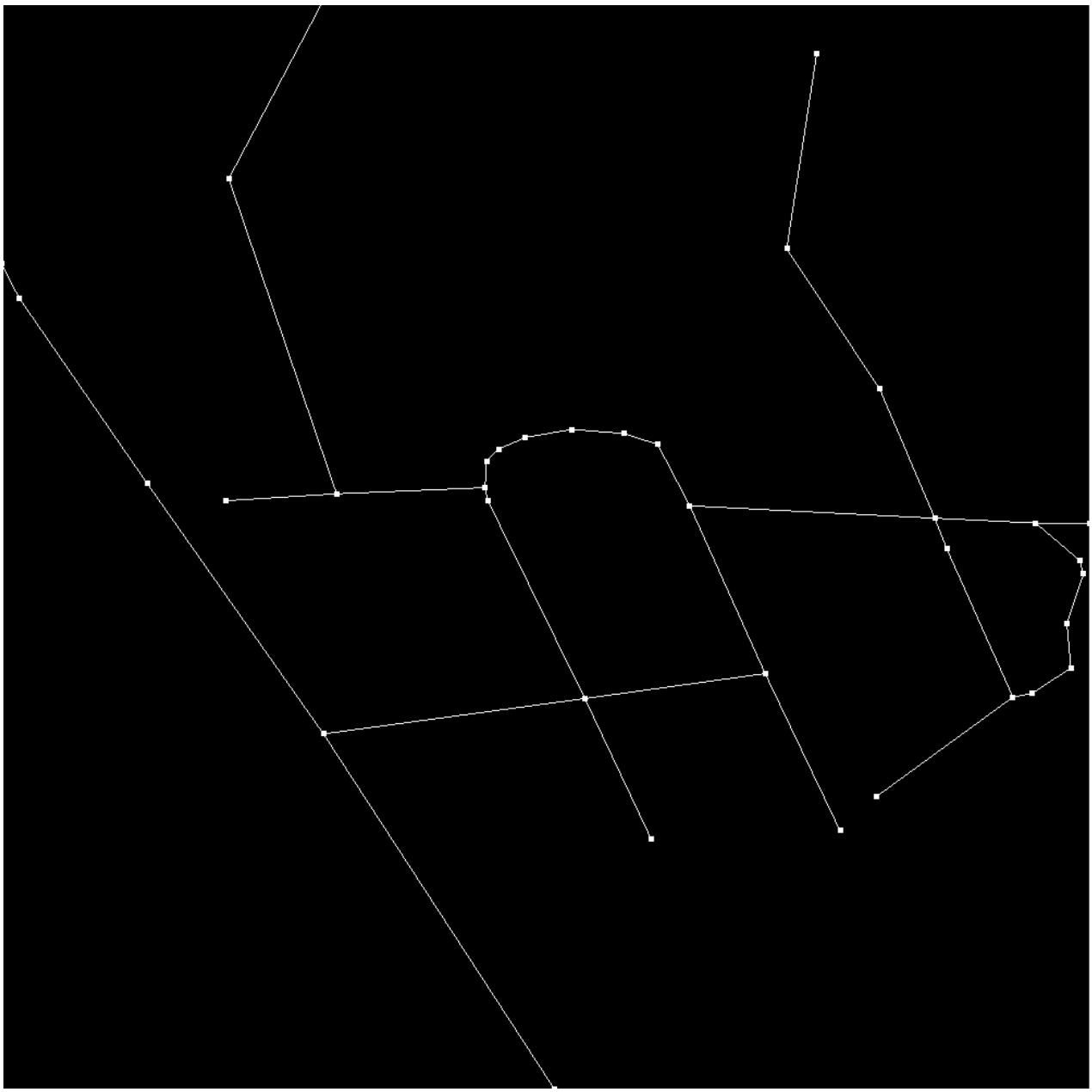
A continuació es mostraran els resultats de l'algoritme de simplificació. Primer es pot veure a la següent il·lustració (Il·lustració 20) el mateix mapa que a l'apartat anterior però ara sí com a resultat del projecte, en aquesta il·lustració no s'ha utilitzat l'algoritme de simplificació.



Il·lustració 20: Mapa de la plaça Ciutat de Figueres no simplificat. Visualitzat des de l'aplicació

Com es pot comprovar té molts nodes que no aporten cap informació. I hi ha una zona on hi ha una corba on hi ha molts nodes, realment calen tants nodes? Cal tenir tantes arestes i haver de propagar per totes aquestes? Mentre hi hagi una distància aproximada entre les seus i no es perdi la forma de la xarxa de carreteres és suficient per fer correctament l'estudi que es vol aconseguir en aquest projecte.

A la següent il·lustració (Il·lustració 21) es pot veure el mateix mapa després d'aplicar-hi l'algoritme de *Reumann-Witkam* que s'ha explicat a l'apartat 8.3.3.1.Algoritme Reumann-Witkam.



Il·lustració 21: Mapa de la plaça Ciutat de Figueres simplificat. Visualitzat des de l'aplicació

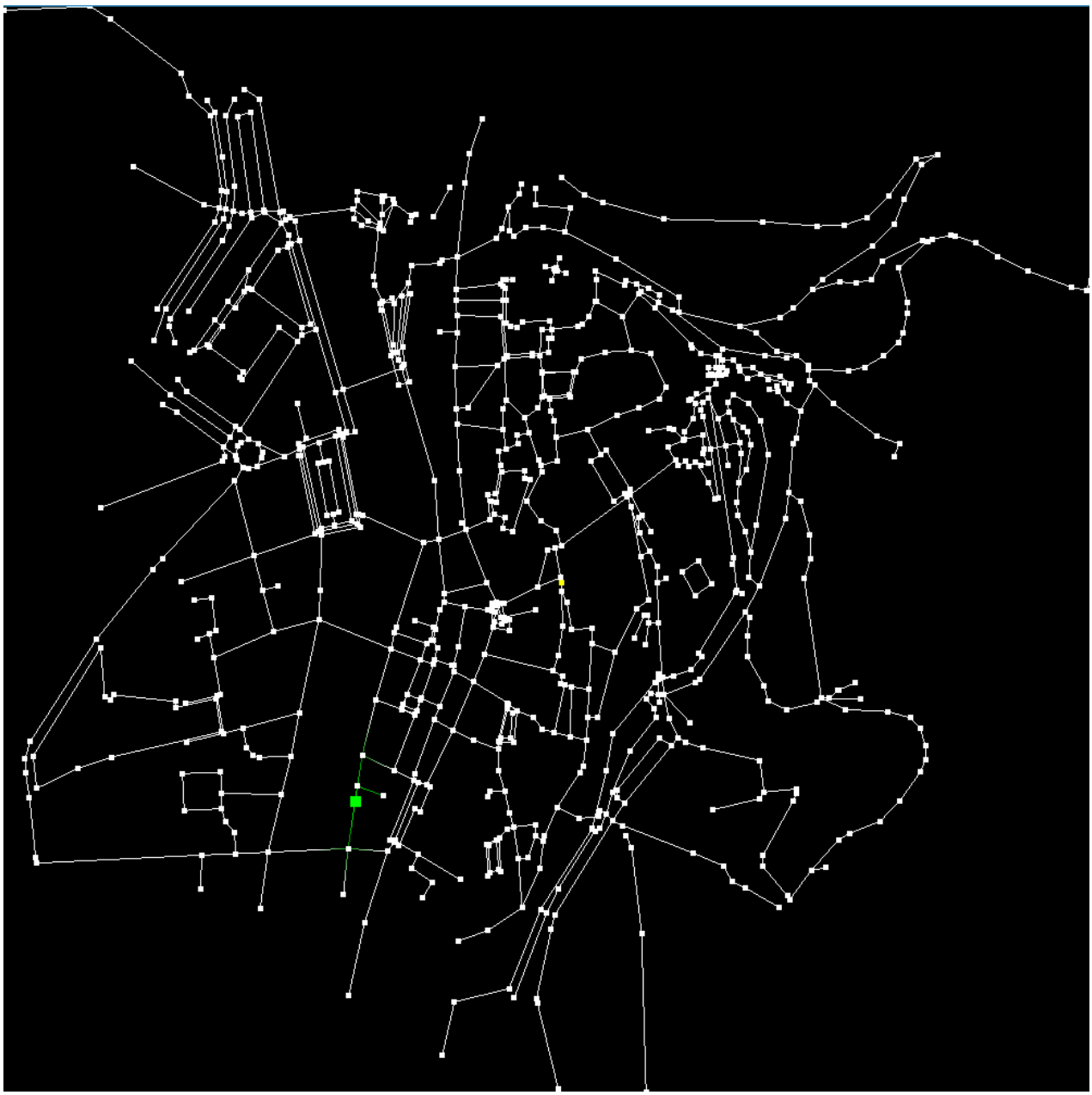
Com es pot veure el mapa és el mateix però hi ha molts menys nodes i les carreteres ja no tenen tanta corba perquè s'ha perdut informació, però com es pot veure no és important ja que es pot utilitzar de la mateixa manera i ara l'algoritme de propagació d'influència serà molt més ràpid perquè hi ha molts menys nodes.

En aquest cas, perdre informació és positiu, perquè és informació que no és rellevant.

10.2.3 Propagació de la influència

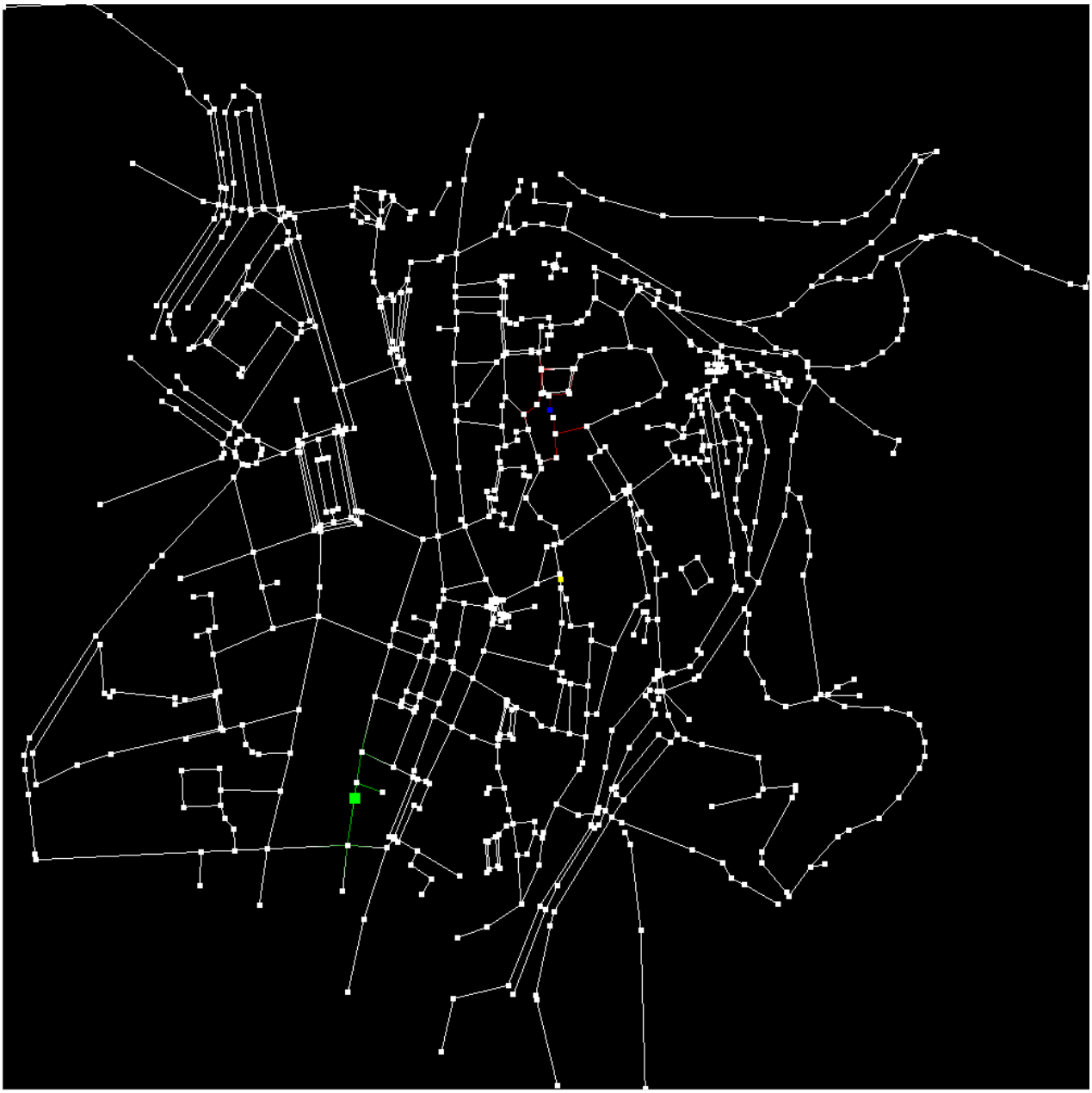
A la següent il·lustració (Il·lustració 22) es pot veure el centre de Girona, un mapa més gran però que funciona de la mateixa manera. Es pot veure com s'ha introduït una seu

positiva i ha propagat la seva influència per la xarxa de carreteres.



Il·lustració 22: Mapa del centre de Girona amb una seu positiva. Visualitzat des de l'aplicació

A la següent il·lustració es pot veure el mateix però s'ha afegit un hotel. Ara la propagació de la influència és negativa, per aquest motiu és vermella.



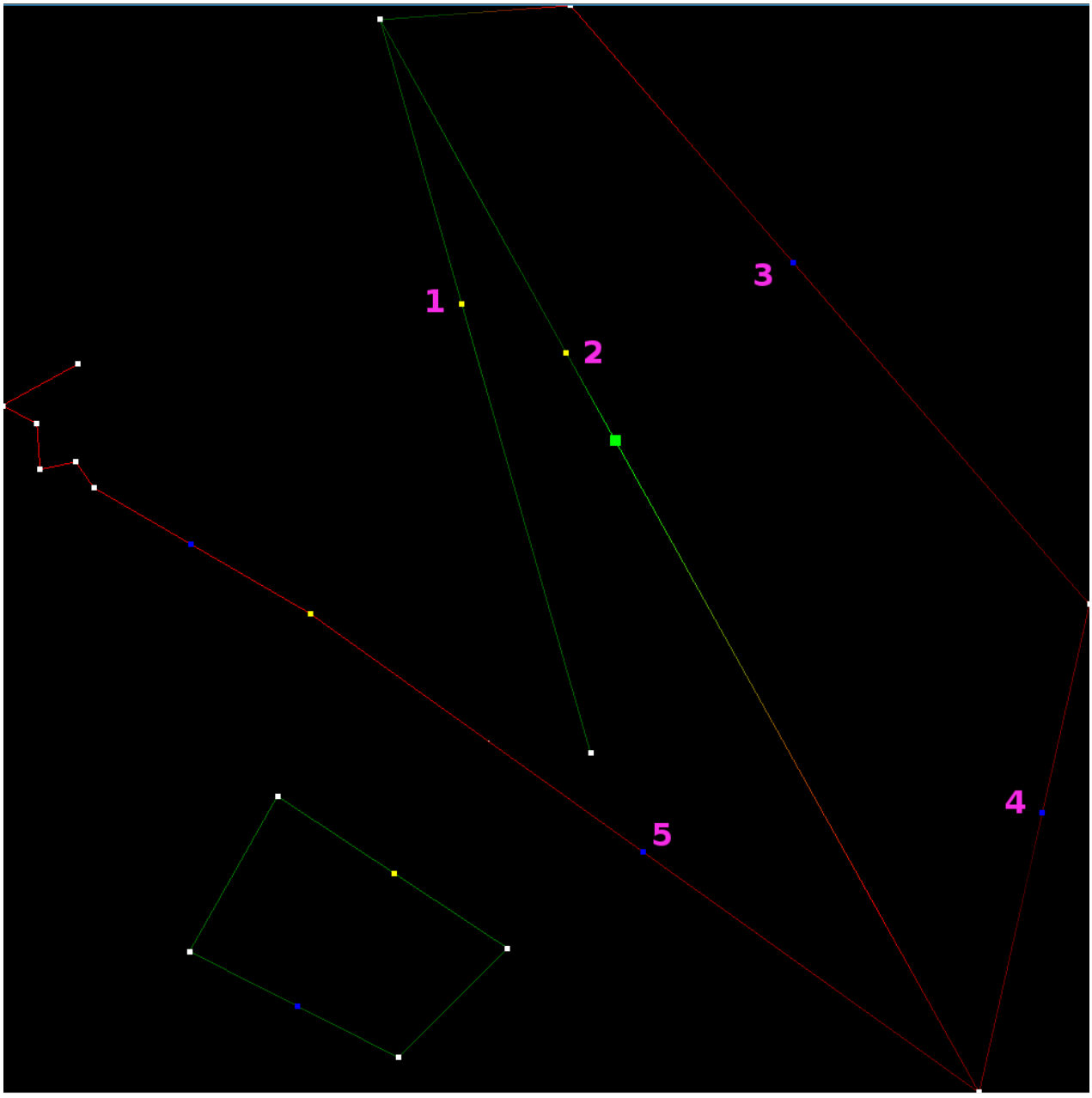
Il·lustració 23: Mapa del centre de Girona amb una seu positiva i una negativa. Visualitzat des de l'aplicació

A les dos imatges es veu un node verd més gran que la resta. Aquest node és el punt amb la puntuació màxima del mapa. En el cas d'aquestes il·lustracions coincideix amb la seu positiva.

A continuació es mostrarà un mapa més petit que és més senzill de visualitzar així es podrà entendre millor.

A la il·lustració (Il·lustració 23) es pot veure el resultat final de l'aplicació. S'hi poden veure varies seus, tant positives (grogues) com negatives (blaves). També es poden observar nodes de color blanc perquè no són seus.

Analitzant les arestes o carreteres, es poden veure dos colors, vermell i verd, amb diferents tons. Com més clar és el color més puntuació té. Jugant amb la interpolació d'*OpenGL* s'ha aconseguit aquest efecte continu entre dos punts de canvi.



Il·lustració 24: Mapa inventat. Resultat final. Visualitzat des de l'aplicació

A simple vista és complex entendre perquè el punt màxim està on marca el punt verd. Cal revisar els números i la influència de cada node per entendre-ho millor. S'han afegit números per poder explicar millor la imatge.

Els nodes 1 i 2 són seus positives i els nodes 3, 4 i 5 negatives.

El punt màxim està al mig d'una carretera molt llarga, si es mira la part de baix de la

carretera, es pot observar com aquesta és vermella i mentre puja es va convertint en verda. Això passa perquè les seus 4 i 5 tenen una influència negativa que es propaga fins a cert punt de la carretera on entra la influència de la seu 2 que és positiva, a més a més, mentre es puja per la carretera, les seus 4 i 5 perden influència perquè cada vegada estan més lluny, en canvi amb el 2 passa al revés. Per tant, arriba un punt on la carretera passa a ser verda perquè la influència de 2 és més forta que la suma de 4 i 5. També pot ser que el 4 o 5 acabés molt abans i només hi hagués un sol node negatiu lluitant contra el 2.

Per la part superior es pot observar com la carretera és verda, però es verd fosc, això significa que és proper a 0. Això passa perquè la seu 3 és molt potent, però no és suficientment potent com per guanyar la suma de 1 i 2. Tot hi això es pot observar com la potència de 3 sobrepassa la seu 2 i acaba en el punt màxim marcat en el mapa. Per tant, el punt màxim és aquí perquè la influència de la seu 3 s'acaba i només queda la de la 2, ja que per baix el 4 i el 5 no hi arriben tampoc.

Capítol 11

Conclusions

11 Conclusions

Aquest capítol serà escrit en primera persona perquè les conclusions són personals i així seran més pròximes al lector.

11.1 Objectius del projecte

L'objectiu del projecte era desenvolupar una eina que els hi fos d'utilitat a les empreses que es volen expandir a decidir on col·locar la seva nova seu, ja sigui un hotel, un restaurant, una botiga, etc. Aquest objectiu està assolit, ja que l'eina desenvolupada mostra un mapa on indica el millor lloc per col·locar aquesta nova seu i a més a més dóna informació de tota la xarxa de carreteres que se l'hi hagi introduït.

D'altre banda, cal recalcar que aquest projecte pot ser molt més complert i se'l pot vestir amb varies eines al voltant que facin que l'experiència de l'usuari mentre l'utilitzi sigui molt més satisfactòria. En el capítol 12 les explicaré, tot hi que ja s'han anomenat en varis capítols anteriors.

En trets generals estic satisfet del projecte desenvolupat, però m'agrada fer-ho millor i s'hauria pogut fer. El projecte el vaig començar tard, perquè no tenia clar què volia fer, després, quan el vaig començar, em vaig entretindre molt en la part de GPU i després en la de visualització amb *Vulkan* i *OpenGL*. Després de tenir clar què utilitzaria i què no, vaig començar el projecte amb *Python*, perquè era molt senzill per mi fer el *parseig* del fitxer *osm*, així que vaig continuar per aquest camí, desenvolupant un part amb *Python* i una altre amb C++. Veient que era inviable mantenir els dos projectes i no era lògic vaig decidir eliminar la part *Python* per continuar amb l'objectiu d'aprendre C++, tot i que sabia programar molt millor i més ràpid amb C++. Aquesta decisió en el món laboral hauria estat la contraria, però ja que estem en l'àmbit educatiu i els projectes serveixen per aprendre vaig decidir continuar amb C++. El camí ha estat més complicat, però ha estat satisfactori.

Així doncs, el camí correcte, el vaig començar a seguir a finals de Març. Això ha fet que tot el projecte vagi condicionat a aquesta falta de temps de desenvolupament. Unes decisions errònies a l'inici del projecte i la falta de temps del dia a dia ha fet que no hagi pogut dedicar les hores que jo hauria volgut dedicar-hi. Arrel d'això es va decidir allargar a l'última convocatòria, cosa que mai havia fet, però per temes laborals no hi havia cap altre opció.

11.2 Objectius personals

El meu objectiu quan vaig decidir fer aquest projecte va ser aprendre una branca nova per a mi de la informàtica, la geometria computacional. Una vessant més matemàtica, tot i que mai ha sigut un dels meus punts forts. Per aquest motiu també em va motivar, ja que suposava un repte. Em vaig decidir a fer-ho perquè l'assignatura que havia fet el segon quadrimestre de segon de màster, Processament de dades espaials, em va agradar molt i vam fer un projecte amb en Mariano Trebino molt interessant. Aquelles reunions de varies hores dibuixant a la pissarra van ser molt divertides i va ser una de les raons principals per escollir el departament de Geometria computacional i la Dra. Marta Fort com a tutora, també volia desenvolupar el projecte amb GPU, aquest era

l'atractiu inicial del projecte. Finalment, a nivell de programació volia tornar a desenvolupar amb C++ perquè és un dels llenguatges més potents. És més complex però es pot programar a més baix nivell fent que l'eficiència millori molt. Això i que hi ha moltes ofertes de feina interessants em motivava molt també.

Però d'aquests objectius inicials que tenia no s'han pogut complir tots. La part matemàtica per la meva part ha estat molt satisfactòria ja que hem desenvolupat uns algoritmes interessants i que en un principi del projecte no ens podíem ni imaginar, van anar sorgint de les reunions periòdiques que vam anar fent amb la Dra. Marta Fort. He après molt en aquest aspecte i d'això n'estic molt satisfet.

La part de C++ ha estat bé per refrescar ja que feia anys que no l'utilitzava i he recordat la complicació que tenia en comparació amb altres llenguatges. Estic content d'haver-lo escollit, perquè ha suposat un repte, però l'esforç que hi he hagut de dedicar per fer aquest projecte, probablement hauria estat molt menor amb un llenguatge com *Python* que en part, també domino més.

I finalment, el que em sap més greu és no haver pogut desenvolupar amb GPU. L'inici del projecte el vaig dedicar a instal·lar i provar *CUDA* i *OpenCL*, però va ser impossible per temes de maquinari, així que es va deixar de banda. Però més endavant vam descobrir el nou algoritme de propagació d'influència que va fer que ja no ens tornéssim a plantejar la GPU perquè no ens calia la paral·lelització. Per tant, ha estat una pena a nivell personal no poder-ho provar i aprendre'n, sobretot tinguen l'oportunitat de treballar amb la Dra. Marta Fort que és qui domina aquest tema. Però cal dir, que estic molt orgullós d'haver trobat el nou algoritme que ha fet que no calgués paral·lelitzar. Així doncs, trobar l'algoritme ha compensat la no utilització de la GPU.

11.3 Conclusió final

Per tots aquests motius, puc afirmar, que el camí ha estat dur però satisfactori. Tornaria a repetir l'elecció del departament i la tutora, però sempre em quedarà el regust de saber què hauria pogut ser d'aquest projecte si hi hagués pogut dedicar el temps que realment es mereix un projecte de final de màster.

Cal dir, que aquest projecte no acaba aquí i això és el que em fa més il·lusió. El projecte ha estat seguit per l'empresa hotelera i en les properes setmanes s'utilitzarà i si tot segueix el seu curs s'hi dedicaran recursos perquè pugui continuar endavant ja que han sabut veure el potencial que pot tenir. En el capítol 12 de treball futur es parlarà de les millores que cal fer i que es faran per aconseguir un producte atractiu. Ara en aquesta empresa, però qui sap si mai arribarà més lluny.

Finalment, vull donar les gràcies a la tutora d'aquest projecte la Dra. Marta Fort que ha sigut un pilar fonamental del projecte ajudant-me en tots els apartats del projecte: anàlisi, estructura, desenvolupament, documentació, etc. Sense ella no hauria estat possible.

Capítol 12

Treball futur

12 Treball futur

Aquest capítol és un capítol que barreja il·lusió i tristesa, el primer perquè si les propostes que venen a continuació es compleixen significarà que el projecte ha agradat a l'empresa i hi posaran recursos per poder continuar millorant el projecte i desenvolupant noves funcionalitats. I la tristesa és perquè algunes funcionalitats es podrien haver fet amb més temps.

A continuació s'enumeraran les millores que es poden fer en el futur per millorar el projecte.

12.1 Automatitzar la creació del mapa

Un dels punts més importants que falta per fer és la de crear un mapa vàlid per introduir en el programa desenvolupat. Això és molt important, ja que forma part del flux del programa.

Ara mateix s'ha de descarregar el mapa de la web d'*Open Street Maps* o d'una altre web que t'ho permeti. Una vegada s'ha aconseguit el fitxer *osm,* s'ha d'editar mitjançant un software addicional, el que s'ha proposat és el *JOSM* i finalment quan s'han col·locat les seus amb els atributs correctes s'ha de passar per paràmetre al programa.

Aquest procés és manual i evidentment això s'ha d'evitar. El que es proposa en aquesta millora és que des del mateix programa es pugui accedir a una web on es selecciona la zona que es vol descarregar i automàticament es vegi el mapa descarregat en el programa, això amb *OpenGL* és totalment factible. El mapa mostrat ja podria estar netejat tal i com es fa en la actualitat. A continuació, s'haurien de proporcionar eines per poder crear seus a la xarxa de carreteres. S'haurien de poder crear seus positives i negatives i a cada seu se l'hi hauria de definir un valor de 1 a 10.

Per aconseguir fer aquesta millora cal fer un gran desenvolupament en la part d'*OpenGL* que és qui proporcionarà totes les eines visuals a l'usuari. Com per exemple un menú on poder seleccionar el tipus de seu i clicar en el punt on es vulgui crear la seu.

Al ser una part tant visual i complexa es va decidir no incloure-ho en el projecte actual en una primera versió perquè la prioritat era fer els algoritmes i no la part visual.

12.2 Radi d'afectació d'una seu

Com en la millora anterior, aquesta també és una millora visual per millorar l'experiència de l'usuari mentre utilitza l'aplicació.

Aquesta millora consta de que l'usuari pugui clicar sobre una seu i es mostri el radi d'afectació que té aquesta seu, així es pot fer una idea de la importància de cada seu, ja que en la actualitat no es té informació d'aquestes i no se sap quina és més important o menys.

12.3 Mostrar informació de les seus

Lligat amb el punt anterior, una de les mancances que té la visualització actual és que no dóna cap informació de cap seu ni del mapa. S'hauria de poder clicar cada seu i

rebre informació, com per exemple, el nom, la influència, el tipus, etc.

12.4 Automatitzar la influència

Una de les preocupacions que s'han tingut des de l'inici del projecte ha estat la manera de puntuar cada seu. Evidentment, si uns professionals en hotels coneixen els hotels d'una ciutat podran decidir més o menys quin és més o menys important, però és difícil comparar-los i definir uns valors coherents.

Es va pensar que una manera objectiva de fer-ho és agafant les puntuacions de les seus. Per exemple, si les seus negatives són hotels es pot recollir la puntuació de *Booking* dels usuaris de cada hotel. O de *TripAdvisor* en el cas de les seus positives o de restaurants, etc.

És una manera senzilla d'automatitzar i millorar les influències de les seus. De fet, aquesta millora es va pensar incloure-la a la primera fase ja que aquests procediments s'han fet amb anterioritat en altres projectes i no hauria suposat un esforç gaire elevat.

Cal dir, que aquestes influències podrien ser una recomanació i l'usuari les podria editar en cas de no estar-hi d'acord o inclús en cas de volgué donar més protagonisme a una seu concreta.

12.5 Automatitzar les seus

Ampliant l'últim punt, es podria arribar a l'extrem de recuperar les seus automàticament i així estalviar molta feina manual o tota a l'usuari.

Per poder aconseguir recuperar les seus caldria determinar unes coordenades del rectangle descarregat a la web d'*OpenStreetMaps.* Això és senzill perquè a la mateixa web apareixen les coordenades.

Un cop es tenen les coordenades del rectangle es poden fer peticions a *Booking* o altres webs per recuperar les seus de la regió marcada, en el cas de *Booking* serien hotels, apartaments o hostals. Per aconseguir les seus positives com punts d'interès es pot aconseguir del mateix mapa d'OpenStreetMaps. De fet, a la versió actual ja s'estan guardant les seus que són hotels o punts d'interès com museus, centres religiosos, miradors, etc. Però s'hauria de millorar ja que ara no estan enllaçats en un punt de la xarxa de carreteres i això és un problema ja que a la versió actual del programa totes les seus han d'estar en una carretera. Amb aquesta millora i l'anterior es podria aconseguir una aplicació totalment automàtica on només caldria configurar petits filtres per obtenir el resultat esperat per l'usuari.

12.6 Filtrar seus

Per millorar la cerca de l'usuari i fer que el resultat sigui el més fiable possible s'haurien d'afegir filtres a l'aplicació. Aquests filtres per exemple constarien de la tipologia de les seus. Per exemple, si es vol construir un hotel de negocis a Girona possiblement interessarà més saber la competència d'hotels categoritzats com de negocis i potser no tant d'hotels turístics. Així doncs, si es permetés categoritzar les seus, es podria filtrar per categoria i millorar les cerques. S'ha posat com a exemple la tipologia d'un hotel, però les categories s'haurien de poder crear per l'usuari i crear-ne tantes com es vulguin. I el programa fer els càlculs sobre els filtres demanats.

Així doncs, es podria canviar de filtre durant l'execució del programa i s'aniria recalculant el resultat segons el filtre demanat.

12.7 Aplicar zoom en el mapa

Una de les millores més senzilles d'implementar és el zoom. A la versió actual es pot moure el mapa amb el ratolí però no es pot fer zoom. Cal un petit desenvolupament per aconseguir-ho. A més a més és una utilitat necessària en mapes grans perquè la xarxa de carreteres pot quedar molt atapeïda.

12.8 Descarregar el resultat

Aquesta és una millora que va demanar l'empresa hotelera ja que la seva idea és posar les imatges dels mapes resultants en informes per poder-los presentar als seus caps.

S'hauria d'habilitar l'opció de crear una imatge a partir del resultat que retorna l'aplicació. És una altre millora de visualització.

Capítol 13

Bibliografia

13 Bibliografia

1. 2001, http://www.agilemanifesto.org/iso/ca/manifesto.html
2. 2016, Mike McLaughlin, https://www.versionone.com/agile-101/agile-methodologies/
3. 2015, David Peterson, http://kanbanblog.com/explained/
4. 2012, Softway, https://www.softwaysolutions.com/blog/might-scrum-thing/
5. 2016, http://www.openstreetmap.org/export#map=16/41.9832/2.8264
6. 2016, http://learnosm.org/en/osm-data/getting-data/
7. 2016, https://josm.openstreetmap.de/
8. 2015, M. Fort, Influència de facilitats, apunts de l'assignatura de Processat de Dades Espaials, Màster en Enginyeria Informàtica
9. 2009, Y. Díez, M. Fort, J.A. Serllarès, Solving Reverse k-Nearest Queries on Road Networks with the GPU, http://metodosestadisticos.unizar.es/~egc09/index_archivos/Trabajos/sellares.pdf
10. 2013, M. Fort, J.A. Sellarès, Solving common influence region queries with the GPU, http://congreso.us.es/ecgeometry/proceedingsECG2013.pdf
11. 2016, http://www.kubuntu.org
12. 2016, https://www.python.org
13. 2016, http://www.cplusplus.com
14. 2016, http://www.jetbrains.com
15. 2016, http://wiki.openstreetmap.org/wiki/Editors
16. 2016, http://qgis.org/en/site
17. 2016, http://www.merkaartor.be
18. 2016, http://wiki.openstreetmap.org/wiki/Osmosis
19. 2016, http://wiki.openstreetmap.org/wiki/Osmfilter
20. 2015, https://xerces.apache.org/xerces-c
21. 2009, http://rapidxml.sourceforge.net
22. 2016, https://github.com/zeux/pugixml
23. 2015, https://sourceforge.net/projects/tinyxml
24. 2008, http://stackoverflow.com/questions/170686/best-open-xml-parser-for-c
25. 2016, https://subversion.apache.org
26. 2016, https://git-scm.com
27. 2016, https://bitbucket.org

28. 2016, http://www.nvidia.com/object/cuda_home_new.html
29. 2016, https://www.khronos.org/opencl
30. 2016, https://www.khronos.org/vulkan
31. 2016, Donald Urquhart, http://www.swiftless.com/opengltuts.html
32. 2016, https://www.opengl.org/documentation/books
33. 2015, https://en.wikibooks.org/wiki/OpenGL_Programming
34. 2016, https://www.opengl.org
35. 2010-2011, Elmar de Koning, psimpl, Douglas-Peucker, http://psimpl.sourceforge.net/douglas-peucker.html
36. 2010-2011, Elmar de Koning, psimpl, Reumann-Witkam, http://psimpl.sourceforge.net/reumann-witkam.html
37. Setembre 2011, Sadan Ekdemir, Efficient Implementation of Polyline Simplification for Large Datasets and Usability Evaluation, http://uu.diva-portal.org/smash/get/diva2:444686/FULLTEXT01.pdf
38. 1956, Edsger W. Dijkstra, https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
39. 2016, http://cython.org
40. 2014, Robert Schrader, http://www.ehowenespanol.com/calcular-distancia-puntos-latitud-longitud-como_452715
41. 2016, http://mathworld.wolfram.com/MercatorProjection.html
42. 2016, https://en.wikipedia.org/wiki/Mercator_projection#Mathematics_of_the_projection
43. 2000-2016, https://cmake.org
44. 2016, https://en.wikipedia.org/wiki/CMake
45. 2011, https://ubuntuforums.org/showthread.php?t=1703770