3D geolokalizazioa androidetan.

Karrera Bukaerako Proiektua

2012.eko uztailaren 11

Maria Teresa Ruiz Monzón

Gainbegiralea:
Joseba Makazaga

eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Eskertza

Nire nebari,
aplikazioa mendian probatzeagatik
ezer gutxi marrazten bazuen ere.

Laburpena

Karrera amaierako proiektu honetan telefono mugikor aurreratuentzako aplikazio grafiko bat garatu da.

Aplikazioak mendizaleei lainoa dagoenean orientatzen lagundu nahi die, ikuspen ezagatik galdu ez daitezen. Hori lortzeko, erabiltzailearen inguruan dagoen paisaia hiru dimentsiotan pantailaratuko da. Non kokatuta dagoen jakiteko GPS sistema atzitzen da. Erabiltzaileak aurrera egiten duen heinean marraztutako ingurunea eguneratzen da.

Orientatzen laguntzeko, begiratzen ari den norantzan dagoen lurrazala bakarrik marraztuko zaio mendizaleari.

Gaien Aurkibidea

Irudien Zerrenda

Taulen Zerrenda

1 Kapitulua

Helburuak

Gaien Aurkibidea

1.1 Helburuak

Proiektu honen helburua mendizaleei lainoa dagoenean bidea aurkitzen laguntzea da. Lainoa dagoenean mendian orientatzea asko zailtzen da urrun dauden erreferentzia-puntuak galtzen baitira. Horrez gain, laino oso itxia dagoenean erreferentziak guztiz galdu daitezke.

Egoera hauetan gertatzen den orientazio eza nolabait gutxitu nahi du aplikazio honek. Horretarako erabiltzailearen inguruan dagoena pantailaratuko zaio.

Mendian mapa sinple batekin kokatzea zaila gerta daitekeenez, ingurunea hiru dimentsiotan marraztuko da. Gainera, 3Dan izanik, gertu amildegiren bat dagoen ikus daiteke, eta beraz, istripuak ekidin.

Helburu horrez gain, karreran zehar ikasitakoak ere praktikan jarri nahi dira. Banakako lana sustatu eta proiektu bat aurrera eramateko nondik norakoak ulertzeko.

Aplikazioak honako baldintza minimoak bete beharko ditu:

- Erabiltzailearen posizioa ahalik eta zehatzen lortu.
- Erabiltzaielaren inguruan dagoen paisaia marraztu.

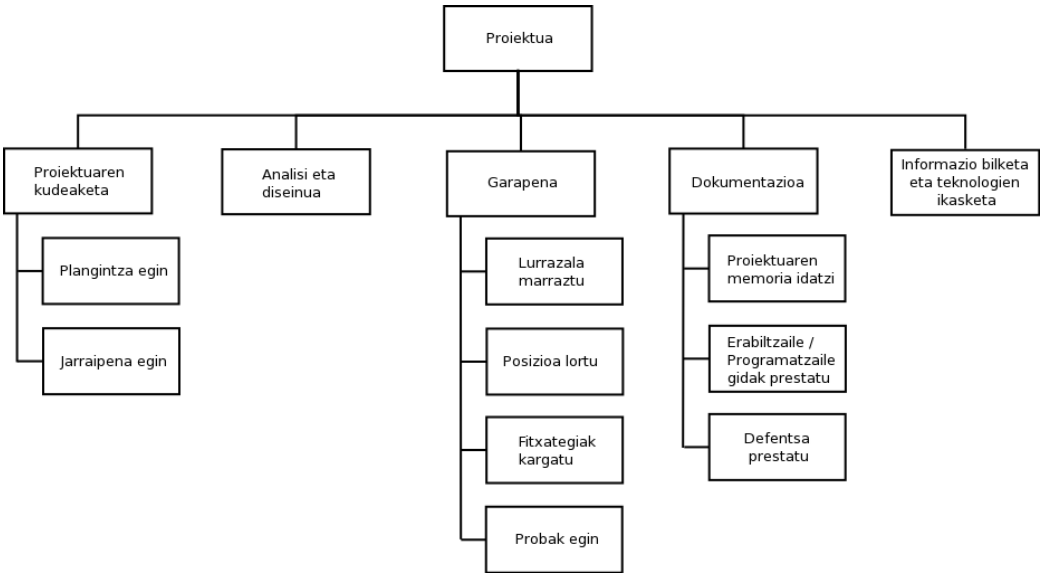
2 Kapitulua

Plangintza

Gaien Aurkibidea

2.1 LDE Diagrama

Proiektuaren planifikazioa hasi baino lehen, hainbat bloke ezberdinetan zatitu da. Alde batetik proiektuaren planifikazioa eta kudeaketa banatu daiteke. Meta atazak izango dira hauek, atazak kudeatuko baitituzte. Beste alde batetik, proiektuaren analisi eta diseinua daude. Aplikazioaren beharrak eta behar horiek nola beteko diren erabaki beharko da zati honetan. Aplikazioaren garapena da beste bloke bat. Bloke honetan aplikazioa garatuko da. Azkenik, dokumentazioarekin zerikusia duen guztia dago. Hala nola memoria, defentsa eta eskuliburuak.



Irudia 2.1: LDE diagrama

2.2 Atazak

Bloke bakoitzaren barruan ataza txikiagoak identifikatu dira. Ataza hauek erabiliko dira gero esfortzua estimatzeko eta jarraipena egiteko. Ataza txikiago hauek identifikatzeko zer nolako lana egin beharko den estimatu da.

- Kudeaketa egin
  - Planifikazioa
  - Jarraipena

- Analisia egin
  - Erabilpen kasuak
- Diseinua egin
  - Klase diagrama
  - Sekuentzi diagrama
- Garapena egin
  - Lurrazala marraztu
    - Edozein lurrazal zati marraztu
    - Lurrazala zatitan banatu
    - Lurrazal zatiak kudeatu
    - Lurrazala sinplifikatu
    - Argiztapena eta texturak jarri
  - Posizioa
    - GPS posizioa lortu
    - Koordenatuak UTM formatura bihurtu
    - Koordenatuei dagokien koadrantea erabaki
    - Telefonoaren orientazioa lortu
  - Fitxategiak
    - Fitxategien deskarga egin
  - Guztia bateratu
  - Probak egin eta erroreak zuzendu
- Dokumentazioa egin
  - Proiektuaren memoria idatzi
  - Erabiltzaile/Programatzaile gidak prestatu
  - Defentsa prestatu
- Informazio bilketa eta teknologien ikasketa
  - Koordenatuak

- Androiderako garapena
- OpenGL ES 2.0ko garapena (Shader, VBO)
- JNI
- Bestelakoak

2.3 Esfortzuaren estimazioa

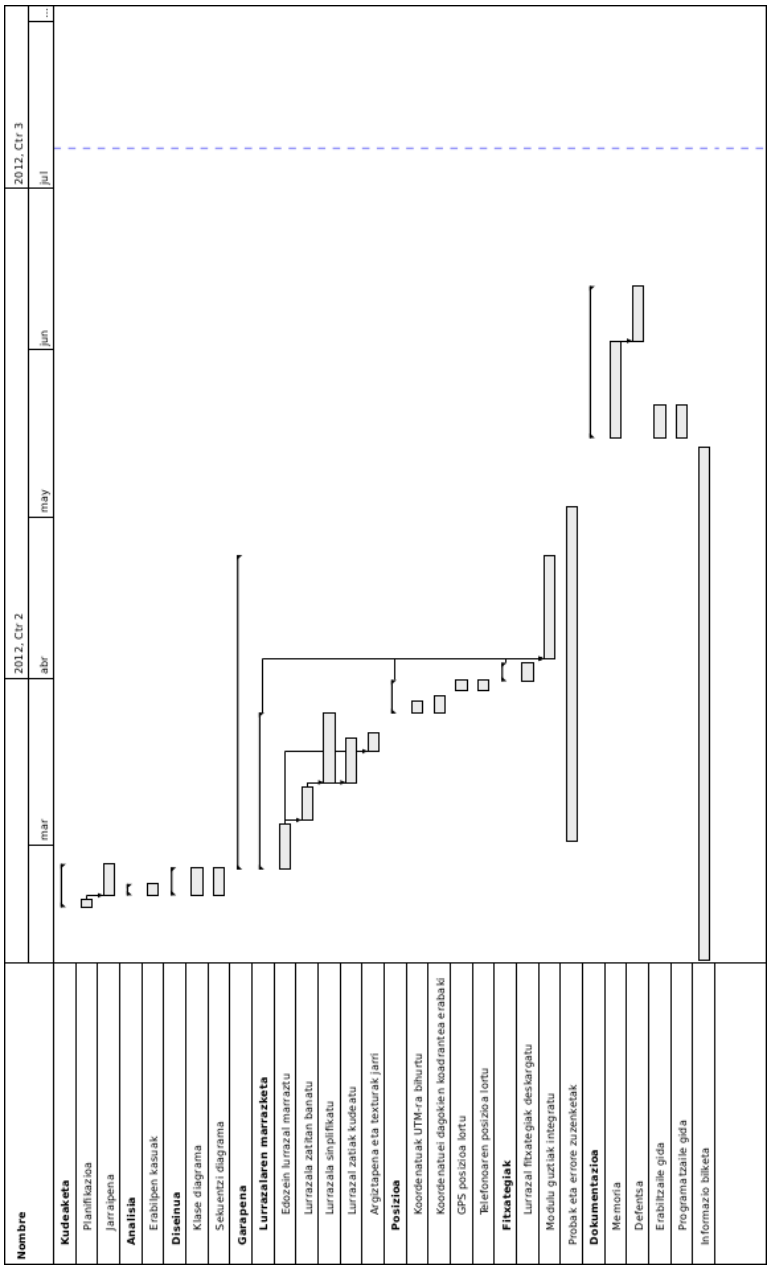
Ataza bakoitza amaitzeko zenbat denbora beharko den estimatu egin da. Ataza bakoitzari esleitutako esfortzua 2.1 taulan ikus daiteke. Aipatzekoa da teknologia batzuen ikasketarako esleitutako ordu kopuru txikia. Klasetik kanpo nire kabuz ikasitako teknologiak dira hauek. Hala ere, proiekturako beharrezkoak dira eta karreran irakasten ez direnez zerrendan jarri dira.

Ataza	***Esfortzua***
1. **Kudeaketa egin**	**14h**
1.1. Planifikazioa	8h
1.2. Jarraipena	6h
2. **Analisia egin**	**4h**
2.1. Erabilpen kasuak	4h
3. **Diseinua egin**	**10h**
3.1. Klase diagrama	5h
3.2. Sekuentzi diagrama	5h
4. **Garapena egin**	**96h**
4.1. *Lurrazala marraztu*	*64h*
4.1.1. Edozein lurrazal zati marraztu	8h
4.1.2. Lurrazala zatitan banatu	8h
4.1.3. Lurrazal zatiak kudeatu	12h
4.1.4. Lurrazala sinplifikatu	24h
4.1.5. Argiztapena eta texturak jarri	12h
4.2. *Posizioa*	*16h*
4.2.1. GPS posizioa lortu	2h
4.2.2. Koordenatuak UTM formatura bihurtu	4h
4.2.3. Koordenatuei dagokien koadrantea erabaki	8h
4.2.4. Telefonoaren orientazioa lortu	2h
4.3. *Fitxategiak*	*4h*
4.3.1. Fitxategien deskarga egin	4h
4.4. *Guztia bateratu*	*12h*
5. **Probak egin eta erroreak zuzendu**	**24h**
6. **Dokumentazioa egin**	**52h**
6.1. Memoria	24h
6.2. Defentsa prestatu	16h
6.3. Erabiltzaile gida	6h
6.4. Programatzaile gida	6h
7. **Informazioa bilketa eta teknologien ikasketa**	**72h**
7.1. Koordenatuak	10h
7.2. Androiderako garapena	1h
7.3. OpenGL ES 2.0ko garapena	1h
7.4. JNI	10h
7.5. Bestelakoak	50h
Guztira	***272h***

Taula 2.1: Esfortzuen estimazioa

2.4 Gantt diagrama

Gantt diagraman ikus daiteke, proiektuaren lan gehiena hasierako hilabeteetan egitea planifikatu dela. Estimatutako baino denbora gehiago behar balitz aste libre nahikoak utzi dira.



Irudia 2.2: Gantt diagrama

2.5 Baliabideak

Proiektu honetarako erabili behar diren baliabideak bakar batzuk baino ez dira. Giza baliabideen aldetik, pertsona bakarrak egingo du lan.

Beharrezko baliabide materialak bi izango dira.

- Ordenagailu bat
- Android mugikorra

Ordenagailua aplikazioa programatzeko beharko da. Mugikorra aldiz, aplikazioaren probak egin ahal izateko.

2.6 Kostuak

Proiektuaren kostuak giza baliabidearen orduak osatuko dute gehienbat.

Giza baliabideak proiektua aurrera eramateko behar izango dituen orduak 272 ordutan estimatu dira. Esperientzia gabeko ikasle bat denez, orduko 15€ko soldatarekin konformatuko dela suposatu da, eta beraz 4080€ ordaindu beharko zaio.

Baliabide materialen artean, ordenagailua hilabete horietan erabiltzeak izan duen kostua bakarrik kontuan artuko da. 5 hilabete behar izango dira proiektua garatzeko. 600€ko ordenagailua izanik eta 5 urteko bizitza duela suposatuz, hilabete bakoitzeko 10€ko kostua duela estimatu daiteke. Mugikorra bigarren eskuz erosi da eta 150€-ko kostua izan du.

Horrez gain, argiaren salneurria igo denez, ordenagailua martxan mantentzeko eta mugikorra kargatzeko behar den elektrizitatearen kostua ere gehitzea erabaki da.

Kalkulatutako kostuari %20a gehituko zaio irabazteren bat izateko.

Baliabidea	***Unitateak***	***Kostua (unitateko)***	***Kostua***
Ordenagailua	5 hilabete	10€	50€
Mugikorra	1	150€	150€
Lan orduak	272 ordu	15€	4080€
Bestelakoak (argia, kafea)	-	-	140€
Irabazte margina		%20	*884€*
Guztira (BEZ gabe)			***5304€***

Taula 2.2: Kostuen estimazioa

2.7 Arriskuak

- *Egindako lanaren galera*
  Egindako lanaren fitxategiak gal daitezke. Adibidez, disko gogorra hondatzen bada. Lana galtzeak berriz hasi behar izatera derrigortzen du.
  Probabilitatea: Ertaina
  Eragina: Altua
  Trataera: Segurtasun kopiak mantendu kopiaren bat galtzearen eragina arintzeko.

- *Gaixotasuna*
  Ikaslea gaixotasunen bat izan dezake proiektuan lan egitetik aldenduko duena.
  Probabilitatea: Ertaina
  Eragina: Altua
  Trataera: Koltxoi denbora utzi, lana atzeratu behar bada, denbora extra horretan lana berreskuratu ahal izateko.

- *Beste ikasgaiek denbora gehiegi behar izatea*
  Beste ikasgai batzuetarako egin beharreko lanak, edota azterketek proiektuan lan egiteko denbora ez uztea.
  Probabilitatea: Altua
  Eragina: Altua
  Trataera: Klaseko lan gehiago egongo diren egunak identifikatu, eta proiektuko lana beste egun batzuetan planifikatu.

2.8 Komunikazioa

Proiektuan partaide bakarrak lan egingo duenez, egongo den komunikazio bakarra proiektuko zuzendariarekin izango da.

Astero egindakoaren laburpen bat bidali beharko zaio. Horrez gain, noiz-behinka bilera bat egongo da aurrerapenak komentatzeko eta iradokizunak jasotzeko.

3 Kapitulua

Jarraipena

Gaien Aurkibidea

3.1 Lan gaztigua

Ataza bakoitzerako emandako orduak 3.1 taulan aurkitzen dira. Horietako batzuk nahiko zehatzak dira denbora neurtu egin delako. Beste batzuetarako aldiz, estimazio bat egin behar izan da. Adibidez, informazioa bilatzen zenbat denbora igaro den jakitea zaila da, 100 ordu jarri dira, baina gehiago ere izan zitezkeen.

Atazak ez dira planifikazioan jarritakoen guztiz berdinak, batzuk konbinatu dira aldi berean egin direlako. Horrez gain, egin behar ez diren ataza batzuk daude. Adibidez, hasiera batean lurrazala sinplifikatu beharko zela uste zen, mugikorrak guztia marrazteko edota memoria arazoak izango zituelakoan, baina azkenean ez da beharrezkoa izan.

Ataza	***Denbora***
1. **Kudeaketa egin**	**6h**
1.1. Planifikazioa	4h
1.2. Jarraipena	2h
2. **Analisia egin**	**1h**
2.1. Erabilpen kasuak	1h
3. **Diseinua egin**	**4h**
3.1. Klase diagrama	4h
4. **Garapena egin**	**81h**
4.1. *Lurrazala marraztu*	*64h*
4.1.1. Lurrazalaren marrazketa	24h
4.1.2. Lurrazala zatitan banatu	40h
4.2. *Posizioa*	*9h*
4.2.1. GPS posizioa lortu eta UTMra itzuli	4h
4.2.2. Telefonoaren orientazioa lortu eta kamera mugitu	5h
4.3. *Fitxategiak*	*8h*
4.3.1. Fitxategien deskarga egin	8h
5. **Probak egin**	**30h**
6. **Dokumentazioa egin**	**45h**
6.1. Memoria	45h
6.2. Defentsa prestatu	-
7. **Informazioa bilketa eta teknologien ikasketa**	**100h**
Guztira	***267h***

Taula 3.1: Lan gaztigua

3.2 Desbiderapena

3.2 taulan planifikatutako orduen eta benetan lan egindako orduen arteko aldea azaltzen da. Proiektuaren diseinu, analisi eta kudeaketan izan dira alde gehien egon diren puntuak.

Harrigarria iruditu zait garapenerako planifikatutako baino ordu gutxiago behar izatea, izan ere, alderantziz gertatuko zela uste nuen.

Desbiderapen gehien izan duen ataza informazio bilaketa izan da. Are alde gehiago egotea ere posiblea da, 100 ordu estimatutako balioa delako.

Ataza	***Estimazioa***	***Benetakoa***	***Aldea***
1. **Kudeaketa egin**	**14h**	**6h**	**-8h**
2. **Analisia egin**	**4h**	**1h**	**-3h**
3. **Diseinua egin**	**10h**	**4h**	**-6h**
4. **Garapena egin**	**96h**	**81h**	**-15h**
4.1. *Lurrazala marraztu*	*64h*	*64h*	*0h*
4.2. *Posizioa*	*16h*	*9h*	*-7h*
4.3. *Fitxategiak*	*4h*	*8h*	*+4h*
4.4. *Guztia bateratu*	*12h*	-	*-12h*
5. **Probak egin**	**24h**	**30h**	**+6h**
6. **Dokumentazioa egin**	**52h**	**45h**	**-7h**
7. **Informazioa bilketa**	**72h**	**100h**	**+28h**
Guztira	***272h***	***267h***	***-5h***

Taula 3.2: Konparaketa

4 Kapitulua

Analisia

Gaien Aurkibidea

4.1 Betebeharrekoak

Aplikazioaren betekizun minimoak gutxi batzuk baino ez dira.

Garrantzitsuena, lurrazal bat hiru dimentsiotan marraztu behar du. Horretarako zerbitzari batetik eskuratutako altuera informazioa erabiliko da. 3D-ko lurrazal zuri batean kokatzea lainotan kokatzea baino are zailagoa denez, lurrazal zatiari dagokion argazkia jarriko zaio.

Erabiltzailearen inguruan dagoen lurrazala marraztu behar duenez, telefonoaren kokapena lortu beharra dago. Kokapen horrekin zein lurrazal zati marraztu behar den erabaki beharko da, dagokion fitxategitik informazioa kargatu eta pantailaratu.

Horrez gain, erabiltzailea nora begira dagoen detektatzea ere desiragarria da alde horretara dagoen lurrazala marrazteko. Hala ez balitz, behintzat lurrazalaren iparra zein den adierazi beharko da, erabiltzaileari orientatzeko aukera ematearren.

Laburtuz, kudeatu beharko den informazioa honako hau da:

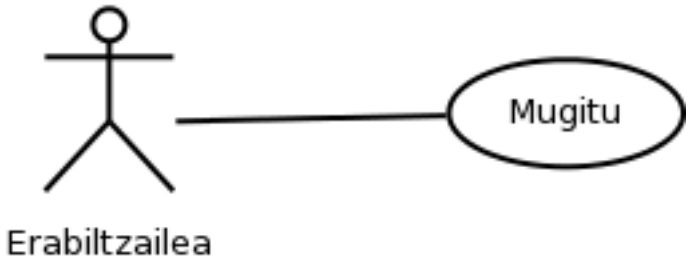
1. Kokapena
2. Begiratze direkzioa
3. Ingurunearen informazioa
   (a) Sateliteko argazkia
   (b) Lurreko altueren informazioa

Baldintza minimo horiek betetzen badira, beste batzuk ere egin daitezke: fitxategien deskarga, bide arriskutsuen detekzioa etab.

4.2 Erabilpen kasuak

Aplikazioa erabiltzailearen kokapena eta norabidea atzitzen ditu eta informazio horrekin dagokion lurrazal zatia marrazten dio. Beraz, erabiltzailearekiko interakzioa oso txikia da. Izan ere, aplikazioak ez du interfaze grafikorik behar.

Hori dela eta, erabilpen kasua oso sinplea da: Erabiltzailea mugitu egiten da. Aplikazioak hori detektatuko du eta pantailan marrazten dena eguneratu.



Irudia 4.1: Erabilpen kasua

4.3 Teknologien azterketa

Aplikazio hau hainbat arlo ezberdinetan oinarrituta dago, eta beraz, erabilitako teknologiak oso heterogeneoak dira.

Android sistema eragilea erabiltzea aukeratu da eta horrek hainbat gauza behartzen ditu. Adibidez, hiru dimentsioko grafikoetarako liburutegi grafiko bakarra eskaintzen du, OpenGL ES®. Programazio lengoaien artean ere, aukera gutxi ematen ditu. Lengoaia hobetsia Java da, baina C/C++ erabiltzeko aukera ere badago. Proiektu honetan biak erabiliko dira.

Horrez gain, erabiltzailearen posizioa jaso behar da. Androidek posizioarentzako bizpahiru aukera ematen ditu, baina mendian ondo lan egiten dutenik bakarra: GPS. Beraz, aproposena GPS sistematik koordenatuak atzitzea da. Koordenatu horiek erabilgarri suertatzen den formatu batera bihurtu behar dira. Itzulketa hau egiteko liburutegi laguntzaile bat erabiltzen da.

Alde teknikoekin jarraituz, badaude beste bi eginbeharreko: erabiltzailea nora begira dagoen erabakitzea eta fitxategiak zerbitzaritik deskargatzeko aukera ematea.

Noranzkoa ipar orratz elektronikoa erabiliz lortuko da, smartphone guztiek halakoa baitaukate eta Androidek kalkulu horiek egiteko erraztasunak ematen baititu. Fitxategien deskarga egiteko aldiz, Apache-ren liburutegi lagungarria erabiliko da.

Azkenik, erabilitako irudi motak bi dira: TIFF eta JPEG. Texturak sortzerakoan, memoria gutxiago erabiltzearren, informazioa konprimituko da.

Dokumentazioarentzat ere, hainbat teknologia eta programa ezberdin erabili dira. Proiektuaren memoria sortzeko LaTeX lengoaia erabili da eta diagramak

Dia programarekin egin dira. Dokumentuan zehar agertzen diren irudiak sortzeko Blender eta Gimp aplikazioak erabili dira.

Proiektuaren planifikazio eta kudeaketa lanak egiteko, Planner eta Taskcoach programak erabili dira. Lehenengoa planifikazioa eta Gantt diagramak egiteko erabili da gehienbat. Bigarrena aldiz, proiektua egiten igarotako ordu kopurua jarraitzeko.

4.3.1 Android

Irudia 4.2: Android logoa

Android Google-ek telefono mugikor aurreratuentzako garatutako sistema eragilea da. Azkeneko urteetan izugarrizko bultzada izan du mugikorren munduan eta dagoeneko smartphonen artean liderra bilakatu da. Linux-en oinarrituta dagoen arren, Dalvik makina birtuala erabiltzen du aplikazioak exekutatzeko.

Androiden bertsio ugari daude, bakoitza bere API ezberdinarekin. Bertsio berri batek bere aurrekoen API-ak exekutatu ditzazke, beraz, ahalik eta bertsio zaharrenean garatzea gomendagarria da. Horrela, ahalik eta mugikor gehiagori zerbitzua eman ahal zaio.

Proiektu honetarako garatutako aplikazioa, Froyo (2.2) bertsiorako egin da. Hainbat arrazoi egon dira erabaki hori hartzeko, baina guztien artean nagusiena, OpenGL bertsioa izan da. Izan ere, Froyo da OpenGL ES 2.0 liburutegia duen lehenengo bertsioa. Horrez gain, zenbat mugikorretan ezin izango den exekutatu kalkulatu da, eta %5-a bakarrik kanpoan uzten dela ikusi da. Beste muga batzuk egongo direla jakinik (RAM adibidez), Android 2.2 aukera egokia dela erabaki da.

Android SDK

Java lengoaian garatzeko, Google-k Android SDK eskaitzen du. SDK honek asko errazten du aplikazioen garapena. Osatzen duten tresnen artean, adibidez, emuladore bat dago, aplikazioak gailu fisiko bat gabe probatu ahal izateko. Tresna hau arduratzen da aplikazioak konpilatu, paketatu eta mugikor zein emuladorean instalatzeaz.

Gainera, DDMS tresnak arazketa egiteko baliabideak eskaintzen ditu. Memoria baliabideak, hariak eta nahi izanez gero, pantailaren edukiak ikusteko aukera ematen du.

Android NDK

Kode natiboarekin lan egiteko tresna lagungarria da hau. SDK-k bezala, kodea konpilatzeko tresnak dauzka. C/C++ konpiladorea dakar eta konpilatzea errazten duten hainbat tresna lagungarrik osatzen dute. Adibidez, Android.mk (Makefileen antzekoak) fitxategien bitartez aplikazioaren zati natiboak konpilatu ahal izateko.

4.3.2 Lengoaiak

Aplikazioa bi lengoaiatan idatzi da. Alde batetik Java lengoaia erabili da, Androideko aplikazioak garatzeko lengoaia estandarra dena. Beste alde batetik C++ erabili da memoria behar handiak dituen zatietarako. Bien arteko lotura Java Native Interface (JNI) izeneko Javaren funtzionalitate bat erabiliz egin da.

Java

Java Sun Microsystems (orain Oracle) enpresak sortutako programazio lengoaia da. Bere helburua programa portableak sortzea ahalbidetzea da: "write once, run everywhere". Hori lortzeko makina birtual batez baliatzen da: Java Virtual Machine (JVM). Java programak makina birtualean exekutatu daitekeen bytecodera itzultzen da, eta objektu kode hori JVM instalatu daitekeen edozein gailutan exekutatu ahalko da.

Androidek Java lengoaia erabiltzen du aplikazioak sortzeko, nahiz eta erabiltzen duen makina birtuala JVM ez den.

C/C++

C 1970 urtean sortutako maila baxuko programazio lengoaia da. UNIX sistema eragilea garatzeko sortu zen. Memoria erabiltzeko malgutasun handia eskaintzen du eta eraginkortasun handia lortzen du.

C++ C lengoaiaren luzapena da, non objektuekin lan egiteko gaitasunak gehitu zaizkion.

JNI

Java Native Interface (JNI) Java eta C/C++-ren arteko elkarrekintza ahalbidetzen duen interfazea da. Javatik zati natiboari dei egin ahal zaio, eta zati natibotik Javako funtzionalitateak atzi daitezke.

Eclipse

Irudia 4.3: Eclipse logoa

Eclipse kode irekiko IDE-a da. Java-n idatzitako programak kodetzeko erabili ohi da, baina lengoaia askotarako pluginak dituenez, ia edozein lengoaiarako balio du.

Android aplikazioen garapena Java-n egin ohi da, eta normalean, Eclipse edo Netbeans tresnak erabiltzen dira. Webgune ofizialean, Eclipserako plugin-a nola instalatu adierazten da, eta beraz, errazago da martxan jartzea. Gainera, C/C++ lengoaian kodetzeko plugina ere eskaintzeaz gain, proiektu hibridoak (Java eta C/C++ erabiltzen dituzten proiektuak) kudeatzeko tresnak dauzka. Horregatik eta hobespen pertsonalagatik, Eclipse erabili da.

4.3.3 Git

Git bertsio kontrol sistema banatua da kodea kudeatzeko oso baliogarria dena. Kodearen progresioa eta aldaketak gordetzen joateko aukera ematen du eta aldaketaren bat desegin behar bada, erraz egin daiteke.

Irudia 4.4: Git logoa

Gitek bi funtzio betetzen ditu proiektu honetan. Alde batetik, kodearen bilakaera modu txukun batean egin ahal izateaz arduratuko da. Bestetik, sistema banatua denez, segurtasun kopiak egiteko eta mantentzeko erabili da. Hainbat konputagailutan kopia bat izanez gero, kopia horiek aldaketa berriekin eguneratzea oso erraza baita.

Eta kopiaren bat galduz gero, besteak guztiz erabilgarriak dira. Segurtasun gehiago izateko kopia bat interneteko zerbitzari batean kokatuta dago.

4.3.4 Irudiak

Bi irudi fitxategi mota erabiltzen dira aplikazioan: GeoTIFF eta JPEG. Lehenengoa altuera informazioa gordetzeko erabiltzen da, bigarrena aldiz, argazkientzako. Horrez gain, texturetan memoria gutxiago behar izateko, formatu trinkotua erabiltzen da, ETC1 alegia.

GeoTIFF

TIFF formatua (Tagged Image File Format), etiketetan oinarritutako irudi formatua da. Etiketa horietan adierazten da, adibidez, zer motako trinkoketa erabiltzen den datuentzat edota datuak zenbat bitetan eta zein ordenean (endianness) adierazita dauden.

GeoTIFF formatu horren hedapena da, non geoposizioarekin zerikusia duten metadatu batzuk gehitu diren. Adibidez, irudiaren koordenatuak edota pixelen arteko distantzia. Altuera informaziorako erabiltzen denez, aplikazioan erabiltzen diren irudiak gris eskalakoak dira. Behar beste altuera gorde ahal izateko 16 bit erabiltzen dira pixel bakoitzeko. Horrez gain, irudiek konpresiorik ez dute erabiltzen. Hala ez balitz, informazioa galduko litzateke.

Fitxategi hauen irakurketa errazteko libtiff liburutegia erabili da.

JPEG

JPEG oso ohikoa den formatua da. Galeradun trinkotze metodoa erabiltzen du irudiaren tamaina murrizteko. Trinkotze maila aldagarria da, irudiaren kalitatearen eta tamainaren arteko erlazioa erabaki ahal izateko.

Irudi mota hauek erabiltzen dira argazkiak gordetzeko. Fitxategi hauen karga eta irudiaren erauztea zaila suertatu daiteke. Pixka bat errazteko libjpeg liburutegia erabili da.

ETC1

Memoria gutxiago behar izateko, texturak trinkotu egin daitezke. Trinkotzeko hainbat metodo daude. Androiden bertsioarekin gertatzen den bezala, trinkoketa mota aukeratzeko orduan, ahalik eta gailu gehienetan erabili ahal izatea kontuan hartu da. Nahiz eta beste batzuk trinkoketa maila altuagoak lortu, ETC1 (Ericsson Texture Compression) da gailuen artean hedatuena. Izan ere, OGLES2.0 onartzen duten guztiek trinkotze mota hau daukate.

4.3.5 OpenGL ES

Irudia 4.5: OpenGL® ES logoa

OpenGL hiru dimentsioko grafikoak sortzeko erabiltzen den estandarra da. OpenGL ES sistema txertatuentzako erabiltzen den OpenGL-ren bertsio murriztua da. Oro har, beharrezkoak ez diren funtzioak kendu zaio.

Androidek OGLES-ren bi bertsio aukeran jartzen ditu, 1.1 eta 2.0. Gailu gehienek bi bertsioak exekutatzeko aukera ematen dute. Baina badaude gutxi batzuk 1.1 bakarrik dutenak.

Bien artean ezberdintasun ugari daude, baina agerikoena, 2.0-an "Fixed pipeline"delakoaren gabezia da. Bertsio horretan, derrigorrez, shaderrak erabili egin behar dira.

OGLES2.0 erabiltzea erabaki da bi arrazoi nagusiengatik:

- Fixed pipeline-a kentzeak, bertsio berria dezente azkarragoa bihurtzen du. Beharrezkoak ez diren hainbat funtzio kentzen direlako.
- Hainbat kalkulu marrazterako orduan egin nahi dira memoria aurreztu ahal izateko eta horretarako shaderrak erabili behar dira. Beraz, shaderrak erabiltzearen derrigortzea ez da arazoa.

GLSL

GLSL OpenGL-k erabiltzen duen shader lengoaia da. Bere sintaxia C-ren antzekoa da. Shaderrek OpenGL-ren "fixed pipeline-a ordezkatzen dute. Bere

betekizuna puntuen azkeneko posizioa eta pixel bakoitzaren kolorea kalkulatzea da.

OpenGL ES 2.0 bertsioan derrigorrezkoa da shaderrak erabiltzea funtzionalitate finkorik ez dagoelako.

Aplikazioa shaderrez baliatzen da beharrezko memoria kopurua zertxobait murrizteko. Adibidez, puntuen normalak aurretik kalkulatuta izan ordez, shaderrean kalkulatzen dira.

GLM

Funtzionalitate finkoaren faltak beste ondorioa batzuk ere baditu. OpenGL-k matrizeak kudeatzeari uzten dio, eta matrizeekin lan egiteko ardura programatzaileari uzten zaio.

Matrizeen eragiketak errazteko GLM liburutegia erabili da. Liburutegi hau, leku aldaketa, biraketa, matrizeen arteko biderkaketak eta beste eragiketa asko egiteaz arduratzen da.

4.3.6 Posizioa

Aplikazioak erabiltzaileari kokatzen lagundu nahi dio. Horretarako, erabiltzailearen posizioa zein den jakin behar da. Androidek aukera bat baino gehiago ematen ditu kokapena lortzeko, baina horien artean bakar bat, GPS sistema, da egokia mendian kokatzeko.

GPS

GPS-ak (Global Positioning System) sateliteak erabiltzen ditu gailu baten kokapena zehazteko. Gailu bat hainbat satelitetara (gutxienez lau) konektatzen da eta triangulazioaren bitartez, bere kokapen zehatza kalkulatzen du.

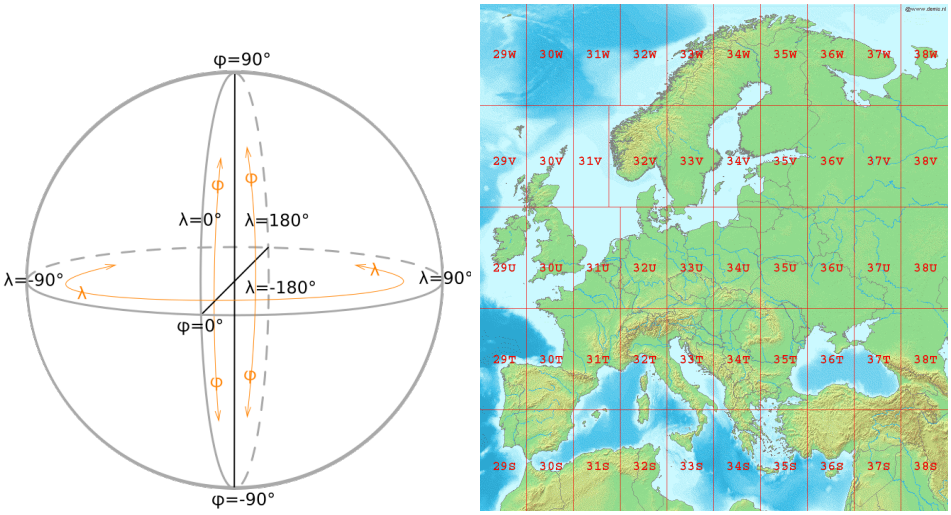
Koordenatu geografikoak

Koordenatu geografikoak normalean erabiltzen diren koordenatuak dira. Koordenatu mota honetan, latitudea eta longitudea zehazten dira kokapen bat adierazteko. Latitudea ekuatorearen eta puntu baten arteko angelua da.

Longitudea aldiz, Greenwicheko meridianoaren eta puntuaren arteko angelua da.

UTM Koordenatuak

Universal Transverse Mercator (UTM) koordenatuak proiekzio geografikoan oinarritutako koordenatu sistema da. Koordenatu geografikoekin alderatuta, angelutan neurtu ordez, UTM koordenatuak metrotan neurtu egiten dira. Puntu baten zehazpena egiteko x eta y koordenatuak erabiltzen dira.



(a) Geografikoak (b) UTM

Irudia 4.6: Koordenatu sistemak (Wikipedia)

Koordenatuen arteko konbertsioa

GPS-ak ematen dituen koordenatuak geografikoak dira. Mapek erabiltzen dituztenak aldiz, UTM. Hori dela eta, bi koordenatu moten arteko konbertsioa egin behar da behin eta berriz. Koordenatuen arteko konbertsioa ez da batere erraza eta geodesiaren nondik norako asko ulertu behar da.

IBM-ren liburutegi[1]bat erabili da UTM-rekin zerikusia duten eragiketa guzti horiek egiteko.

[1]https://www.ibm.com/developerworks/java/library/j-coordconvert/

Mapa zatiak

Euskal Herriko mapa hainbat zatitan banatuta dago. Marrazketa egiterakoan, koordenatu jakin bat zein zatiri dagokion jakin behar da. Esleipen hori egin ahal izateko, mapa zati bakoitzaren informazioa behar da.

Zati bakoitzari buruz ezagutu behar diren datuak hauek dira: zatiaren izkinetan dauden puntuak eta zati horren irudiak gordetzen dituzten fitxategien izena. Zatien informazio hori Shapefile motako fitxategi berezietan gordetzen da.

4.3.7 Dokumentazioa

LaTeX

Dokumentuak sortzeko tresna da LaTeX. Makro multzo bat da TeX errazteko asmoz sortuta. Dokumentu teknikoak sortzeko erabiltzen da kalitate handia eskaintzen duelako.

Gummi

LaTeX editore sinple bat da. Idatzitako fitxategiaren emaitza pantailaren eskuin ondoan erakusten du.

Dia

Diagramak egiteko software gehiegirik ez dago, eta Linux-en lan egiten duenik are gutxiago. Dia da horietako bat, kode askeko softwarea da eta edozein motako diagramak egiteko erraztasunak ematen ditu.

Blender

Modelatze, animazio eta hiru dimentsioko grafikoak egiteko balio duen programa da. Software libre eta plataforma anitzekoa da, hala ere programa komertzialen ezaugarri asko ditu.

Memorian erabili diren hainbat irudi sortzeko erabili da.

Irudia 4.7: Blender logoa

GIMP

GIMP (GNU Image Manipulation Program) irudiak editatzeko software librea da.

Blender-rekin sortutako irudiak editatzeko erabili da gehienbat.

Irudia 4.8: GIMP logoa

4.3.8 Planifikazio eta kudeaketa

Planner

GNOME ingurumenarentzako proiektu kudeaketa softwarea da Planner. Gantt diagrama eta, oro har, proiektuaren hasierako plangintza egiteko erabili da.

Taskcoach

Task Coach ataza pertsonalen eta eginbeharrekoak kontrolatzeko aplikazioa da. Atazen jarraipena egiteko erabili egin da gehienbat. Jarduera bakoitzean zenbat ordu eman diren modu errazean kalkulatu ahal izateko.

5 Kapitulua

Diseinua

Gaien Aurkibidea

5.1 Sarrera

Diseinua egiteko orduan hainbat gauza izan behar dira kontuan.

Aplikazioak honako funtzionalitateak izango ditu:

- Zerbitzari batetik irudi batzuk deskargatu behar dira.
- Irudi horiek lurrazal zati bat marrazteko erabiliko dira.
- Erabiltzailetik posizioa eta begiratze norabideak atzituko dira eta lurrazala mugitzeko erabili.

Informazio asko eta pisutsua da, horregatik ahalik eta modu eraginkorrenean antolatu eta kudeatu beharko da.

Ezin da ahaztu telefono mugikorretan exekutatuko den aplikazioa dela, mugikorrek kontuan hartu beharreko muga batzuk ezartzen baitituzte.

5.1.1 Mugak

Telefono mugikorrak geroz eta ahaltsuagoak izan arren, muga anitz ezartzen dituzte aplikazioak garatzerakoan eta horiek saihestea izan da lanik makalena.

Muga guztien artetik, aplikazio honetan eragin handiena izan duena memoria muga izan da. Bi memoria arazo dira nagusi:

- Android-ek ezarritako muga.

  Android sistema eragileak, aplikazio bakoitzak erabili dezakeen memoria maximo bat ezartzen du. Limite hori Android bertsioen eta gailuen artean ezberdina da. Oro har, bermatu daiteke gutxienez 16MBkoa izango dela. Limite hori altuagoa da bertsio berriagoetan, adibidez, 2.3 bertsioan ohikoena 24MBko limitea da eta 4.0 bertsioan 48MB limitea omen dago.

  Aplikazioan erabiltzen den informazio kantitatea ikaragarria da, batez ere irudiak oso handiak direlako, eta muga hori betetzea ia ezinezkoa egiten da. Zorionez, badago limite hori saihesterik.

  Android-rentzako aplikazioak, Javan egiten dira normalean, baina badago C/C++ lengoaian ere idaztea. Lengoaia horretan idatzitako zatiak kode natibora konpilatzen dira. Kode natiboa denez, zati horiek zuzenean prozesagailuan exekutatzen dira, eta ez makina birtualean.

Memoria muga Javaren makina birtualean alokatutako memoriari dagokio bakarrik. Hori dela eta, C/C++-en idatzitako programaren zatiek ez dute limite artifizial hori bete behar. Honek behartzen du lurrazalarekin zerikusia duen gehiena C/C++ lengoaian idaztera.

- Gailuen memoria kopurua.

  Oztopo artifizial hori alde batera utzita, ezin da ahaztu Android mundua oso heterogeneoa dela. Gama baxuan 128MB-eko memoriadun mugikorrak daude, eta gama altuan 2GB-era ailegatzen dira. Beraz, nahiz eta nahi beste memoria alokatu ahal izan, limite fisikoa ezin da saihestu. Probetarako erabilitakoek 512MBeko memoria dute, eta tamaina hori izanik ondo ibiliko dela berma daiteke. Hori baino gutxiagokoetan posible da funtzionatzea, baina bermerik ez da ematen.

5.2 Java eta zati natiboaren arteko interfazea

Bi lengoaien artean deiak egin ahal izateko JNI erabiltzen da. JNI nahiko nahasgarria da, horregatik bi lengoaien arteko interfazea ahalik eta txikien egin nahi da. Beraz, zati natiboak Javatik behar duena baino ez du jasoko: kokapena eta kameraren posizioa.

Zati natibotik Javara deiak egitea are zailagoa denez, hasiera batean norantz horretan deirik ez egitea erabaki zen. Zoritxarrez, deskargak Javan egin behar dira nahitaez (ikus 5.5 atala) eta zein fitxategi falta diren zati natiboak baino ez daki. Beraz, Javari adierazi beharko zaio zein fitxategi deskargatu behar diren.

5.3 Erabiltzailearekiko interfazea

Erabiltzailearekiko interakzioa, oraingoz, bi modu ezberdinetan egiten da:

- Alde batetik erabiltzailearen posizioa jakin behar da. Posizio hori GPStik irakurriko da eta marraztu behar den lurrazal zatia kalkulatzeko eta kamera kokatzeko balioko du. GPSak eskaintzen dituen koordenatuak sistema geografikoan adierazita daude (Latitudea/Longitudea). Aplikazioan erabiltzen den argazki eta informazio guztia aldiz, UTM mo-

tako koordenatuak erabiltzen dituzte. Horregatik konbertsioa egiteaz arduratuko den liburutegi bat ere egongo da.

- Erabiltzaile posizioaz gain, mugikorraren orientazioa ere atzitzen da. Informazio hau kamera mugitzeko erabiliko da. Adibidez, erabiltzailea iparralderantz begira badago, iparraldera dagoen lurrazal zatia marraztuko zaio, edota malda batean behera begiratzen badu, kamara beherantz begira jarriko da.

Nahiz eta proiektuko helburua ez izan, lurrazalean zehar mugitzeko beste moduak ere eman nahi dira (ikus hobekuntzen 7.1.2 atala). Aukera horiek eman ahal izateko erabiltzailearekiko interfazea gehitu beharko litzateke. Alde batetik, lurrazalean zehar mugitzeko metodoa aldatzeko aukera eman beharko da: kokapenarekin edo modu librea. Bestetik, behatzen keinuak detektatu beharko dira. Adibidez, zooma kontrolatzeko bi behatz hurbildu edo aldendu.

5.4 Lurrazala

Lurrazala marraztu ahal izateko, informazio asko mantendu behar da: uneko posizioa, mapa zatien informazioa, altuerak, sateliteko argazkiak, etab.

Ahalik eta lurrazal zati handiena marraztu nahi da, betiere memoria agortu gabe. Horrez gain, kontuan izan behar da, erabiltzailea ez dela beti puntu berdinean egongo, eta mugitzen den heinean lurrazala eguneratu beharko dela.

Informazioaren antolaketa oso garrantzitsua izango da puntu honetan.

5.5 Fitxategien kudeaketa

Askotan aipatu den bezala, erabiltzen diren fitxategiak oso handiak dira. Horregatik, kudeaketa ahalik eta modurik eraginkorrenean egin nahi da.

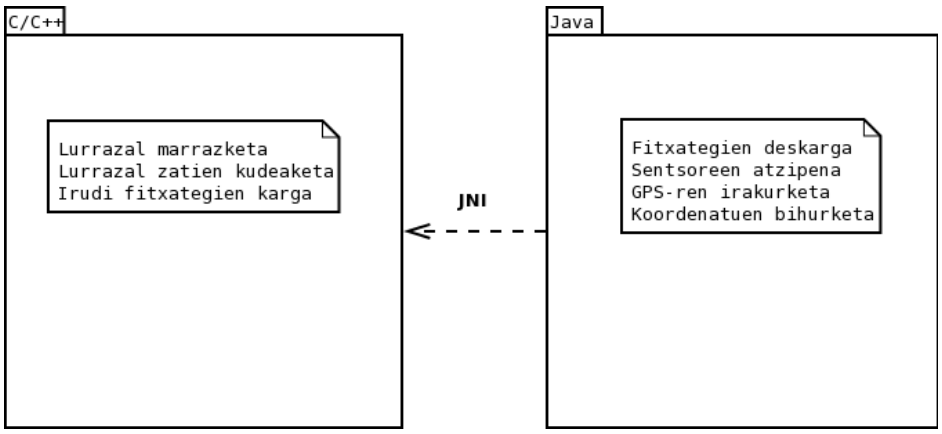
Deskarga egiten den bitartean aplikazioa lur jota gelditu ez dadin, deskargak hari berri batean egingo dira. Androiderako garatzean, hari berriak beti makina birtualaren barruan sortzea gomendatzen da. Horregatik, deskargak Javan kodetu beharko dira.

Konkurrentzia erabiltzen denez, prozesuen arteko sinkronizazioa gehitu beharra dago. Kasu honetan nahiko zaila izango da komunikazioa gehitzea. Sinkronizatu beharrekoak, deskarga egiten duen prozesua eta fitxategiak kargatzen dituena dira, eta bi horiek ez dira lengoaia berdinean idatzita egongo. Amaitu gabeko fitxategiak kargatzea ekiditeko, oinarrizko sinkronizazioa egin daiteke. Fitxategien izen aldaketa ia berehalakoa denez, deskarga beste izen batekin egin daiteke eta amaitzean izen egokiarekin izendatu.

Behin eta berriro fitxategi berdina deskargatu behar ez izateko, dagoeneko lortutakoak SD txartelean mantenduko dira. Gainera, fitxategiak dagoeneko SD txartelean eskuragarri badaude Internet gabe funtzionatzeko aukera eman beharko du.

5.6 UML diagramak

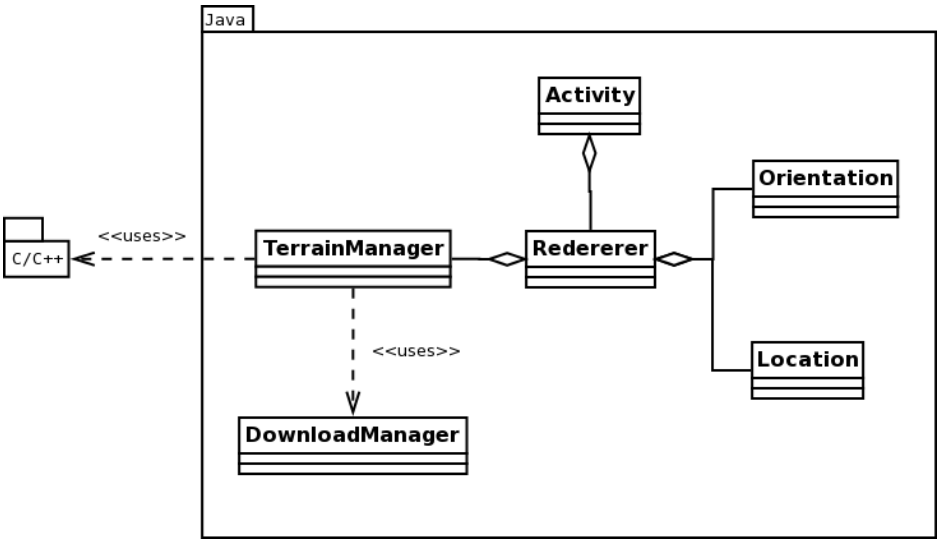
Javan idatzitako kodeak erabiltzailearekiko interfaze lanak beteko ditu gehienbat. Lurrazalaren karga, marrazketa eta kudeaketa ardurak kode natiboari lagako zaizkio.



Irudia 5.1: Diseinu orokorra

5.6.1 Java zatia

Activity Android aplikazioen oinarria da. Aplikazioa puntu honetatik hasten da, eta aplikazioak beharko dituen aldagai guztien hasieraketa hemen egingo da.



Irudia 5.2: Java zatiaren klase diagrama

Rendererek marrazketarekin zerikusia duen guztia kudeatuko du. Adibidez, OpenGL kontextua ezartzeaz arduratuko da. Horrez gain, frame bakoitzeko, kokapen eta orientazio berriak atzitu eta kamera ezarri ondoren, kode natiboari deiak egingo dizkio lurrazala marrazteko.

TerrainManager kode natiboarekin interfazea izango da eta kode natiboari deiak egiteko erabiliko da.

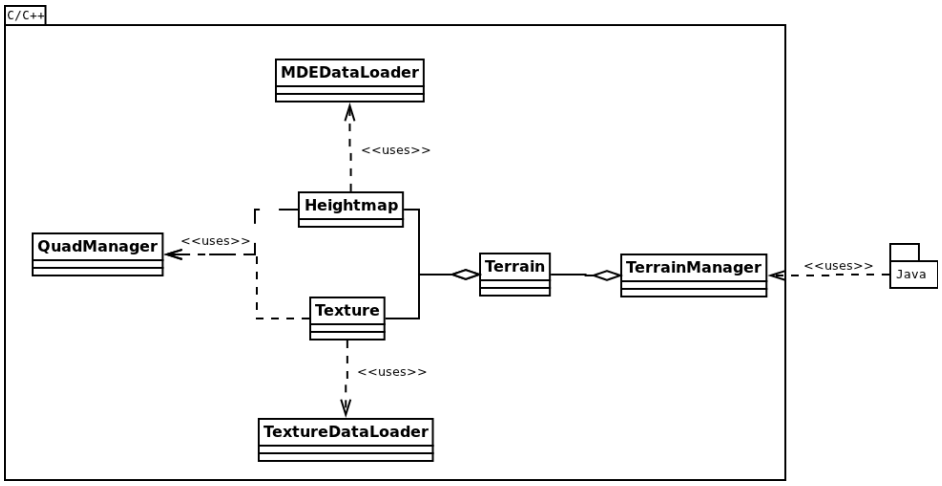
DownloadManagerek fitxategien deskargak kudeatuko ditu. Fitxategi berri bat eskuratzeko eskatzen zaion bakoitzean, deskarga egiteaz arduratuko den hari berri bat sortuko du.

Location eta Orientation erabiltzailearen kokapen eta direkzioa atzitzeaz arduratuko dira.

5.6.2 Zati natiboa

Zati natiboan lurrazalarekin zerikusia duen guztia batu da. Terrainek lurrazal zatiak kudeatuko ditu. Zati berriak sortuko ditu beharrezkoa denean, eta urrun daudenak ezabatu. TerrainManager Javarekiko interfazea da eta Javatik deiak jasotzeko balio du. Deskargak behar direnean Javari abisatzeaz ere arduratuko da.

Texture eta Heightmapek lurrazalaren informazioa gordetzen dute. Lehengoak argazkiak mantentzen ditu, eta besteak altuera informazioa. MDEDa-



Irudia 5.3: Zati natiboaren klase diagrama

taLoader eta TextureDataLoader oso antzekoak dira, lehenengoak altuera irudiak kudeatuko ditu, besteak argazkiak. Irudiak diskotik irakurriko dituzte eta haietatik eskatzen zaizkien zatiak itzuli.

6 Kapitulua

Garapena

Gaien Aurkibidea

6.1 Ingurunearen marrazketa

Marrazketa aplikazioaren atalik garrantzitsuenetarikoa da eta garapena aurrera egin ahala, hainbat erabaki hartu behar izan dira.

Marrazketa prozesua hainbat urratsetan bana daiteke:

1. Uneko posizioa lortu
2. Kokapen horren fitxategiak zerbitzaritik eskuratu
3. Fitxategiak diskotik irakurri
4. Marraztu nahi den zatiak behar dituen datuak prestatu (puntuen koordenatuak eta bektore normalak)
5. Guztia pantailaratu

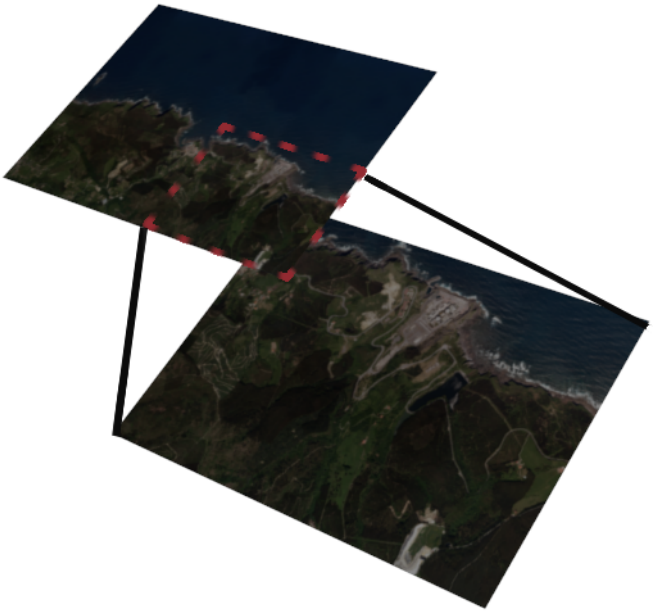
6.1.1 Informazioaren antolaketa

Zerbitzaritik eskuratzen diren fitxategiak bereizmenez sailkatuta daude. Bereizmen bakoitzeko fitxategiek azalera jakin bat hartzen dute, bereizmen txikiagokoek azalera handiagoa daukate eta bereizmen altuagoa dutenak aldiz, azalera txikiagoa. Maila hierarkia bat antolatuta dago, maila bateko mapa zati bat bereizmen handiagoko beste 4 zatitan banatzen da. Bereizmen handieneko fitxategiek 8 Km^2-ko azalera daukate.

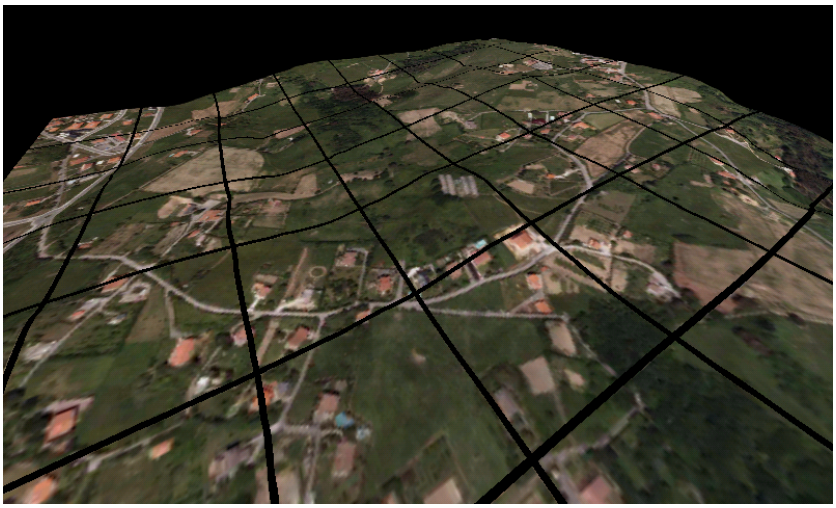
Mapa zati osoak marraztea gehiegi dela erabaki da, gehienbat erabiltzailea zati baten mugan badago, bere inguruan dauden laurak marraztu beharko liratekeelako. Horregatik marraztuko dena aukeratzeko beste metodo bat erabili da.

GPStik lortzen den posizioa zentro bezala jarriko da. Puntu horren inguruan karratu bat antolatu da, puntu kardinal bakoitzerantz metro kopuru jakin bat hartuko dituena. Erabiltzailea pixka bat mugitzen denean zati osoa berriz kargatu behar ez izateko, karratu hori beste zati txikiagotan banatu da. Erabiltzailea mugitzen den heinean, bere noranzkoan dauden mapa zatitxo berriak kargatzen joango dira, eta urrunen gelditzen direnak ezabatu. 6.2 irudian ikus daiteke zatiak nola antolatu diren.

Lurrazal zatitxo bakoitzari bere hasierako koordenatuak eta koordenatu horiei dagokien argazki zatia esleitzen zaizkio. Informazio hori beharrezkoa izango da marrazketa egiterakoan.



Irudia 6.1: Bereizmen mailak



Irudia 6.2: Zatikako marrazketa

Altuera informazioa gris eskalako irudi batean gordetzen da, baina beste modu batean antolatu da. Zatitxo bakoitzeko irudi bat gorde ordez, marraztu nahi den karratu osorako irudi bakarra gordetzen da. Erabiltzailea mugitzean berriz kargatu behar izatearen arazoa konpontzeko, irudi hori marraztuko den karratua baino handiagoa da. Erabiltzaileak aurrera egiten badu ere, metro

batzuetan ez da berriz kargatu beharko (6.3 irudia). Ertzera ailegatzean bakarrik kargatuko da irudi berria.

(a) Hasieran (b) Erabiltzailea mugitzean

Irudia 6.3: Altuera informazioa

6.1.2 Marrazketa

Hiru dimentsioko irudigintzan marrazketak poligonoen bidez egiten dira, normalean hirukiak. Hiruki horiek sortzeko, hiru puntu behar dira eta puntu bakoitzeko, bere espazioko koordenatuak (x, y eta z).

Lurrazala 2 dimentsioko puntu sare batez adierazita dago. Puntu bakoitzari altuera bat esleitzen zaio 3 dimentsiotako koordenatuak lortzeko. Gero, puntu horiek hirukietan antolatzen dira lurrazala marraztu ahal izateko.

Karratu osoa marraztearren, prozesu hori zatitxo bakoitzeko puntuekin egin beharko da. Koordenatu mordo hori memorian gorde behar ez izateko, puntuen koordenatuen kalkulua txartel grafikoan egiten da shaderrak erabiliz. Shaderrak txartel grafikoan exekutatzen diren programa txikiak dira, puntu eta pixelekin eragiketak egiteko prestatuta daudenak.

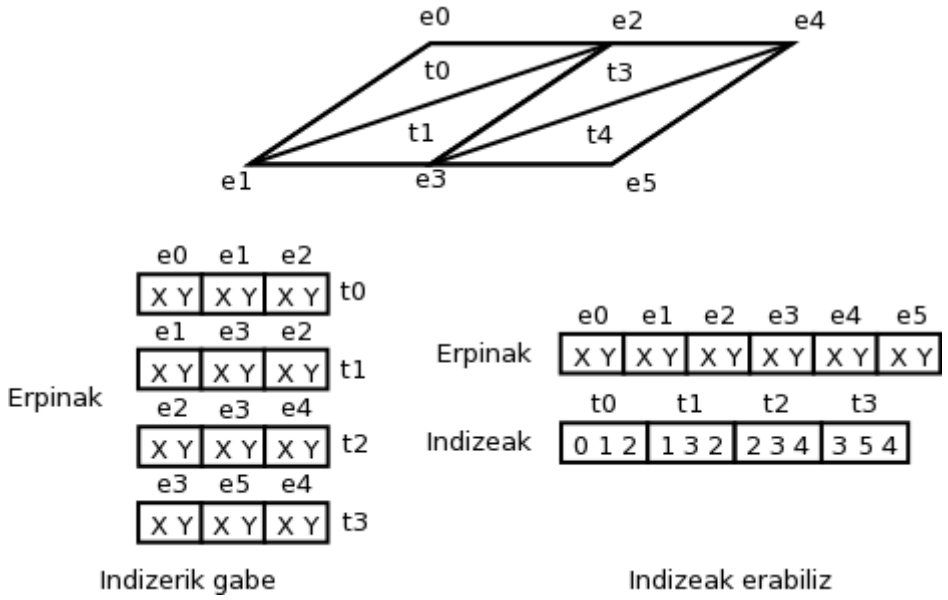
Shaderrek egingo dute kalkulu gehiena, baina bere lana egin ahal izateko, hainbat datu pasa behar zaizkie.

Alde batetik, nahiz eta azkeneko puntuen koordenatuak txartel grafikoan sortu, txantiloi moduko bat behar da. Txantiloi hori puntu sare sinple bat izango da. (0,0) koordenatuetatik hasita, zati batek beharko dituen puntu

guztien 2D koordenatuak gordeko dira bertan. Marrazteko orduan, beneta-ko koordenatuak lortzeko, zati bakoitzari dagozkion munduko koordenatuak gehitu baino ez da egin beharko. Altuera irudi batetik atzituko da.

Puntu sare bakar bat erabili beharko da beraz lurrazal osoaren marrazketa egiteko. Hala ere, bakarra behar izatearekin memoria kopurua asko txikitzen bada ere, sarea hirukiz betetzean, puntu asko errepikatu egiten dira. Hori ekiditeko badago indizeak erabiltzerik.

Buffer batean puntuak gordeko dira, bakoitza behin baino ez. Beste buffer batean, hirukiak sortzeko puntu horiek zein ordenetan jarri behar diren adieraziko da.

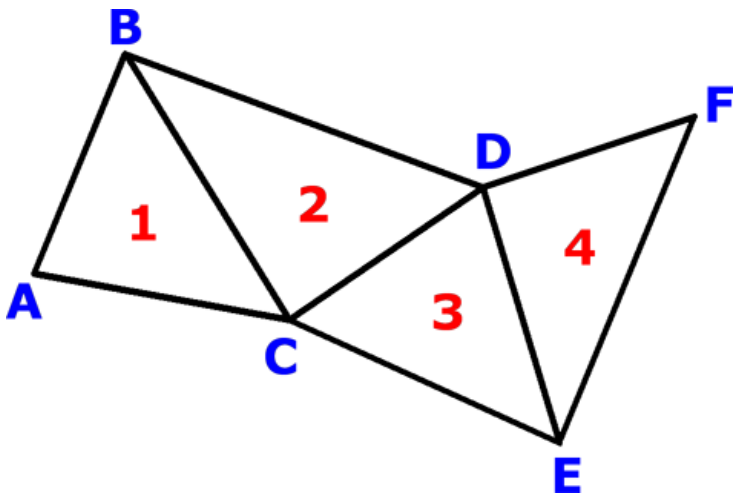


Irudia 6.4: Puntu bufferak

Are memoria gutxiago behar izateko, triangeluak gorde ordez "triangle strip" deritzona erabili da. Triangelu zerrenda da hau: puntu bat bere aurretik dauden biekin elkartzen da hirukiak osatzeko.

Triangelu zerrenda bakarra erabili nahi da zati osoa marrazteko, baina tentuz ibili behar da ilara batetik hurrengoa pasatzean.

Alde batetik, ilara bateko azkeneko puntua jarri orduko hurrengo ilarakoa jartzen bada, nahi ez dugun aurpegi bat agertzen da. Hiruki hori guztiz bertikala izango da, eta kasu batzuetan ez da ikusiko lurrazalak berak estaltzen duelako. Baina baliteke kontrakoa gertatzea. 6.6a irudian ikus daiteke ezker aldean aurpegi urdin bat sortu dela.



Irudia 6.5: Triangle strip (Wikipedia)

Aurpegi hori ekiditeko modu erraz bat dago: ilara bakoitzaren azkeneko puntua errepikatzea. Puntu bat errepikatzean ingeleraz "Degenerated triangle" deritzona sortzen da. Triangelu horiei horrela deitzen zaizkie bi puntu errepikatzen badira ez delako hirukia sortzen, zuzena baizik.

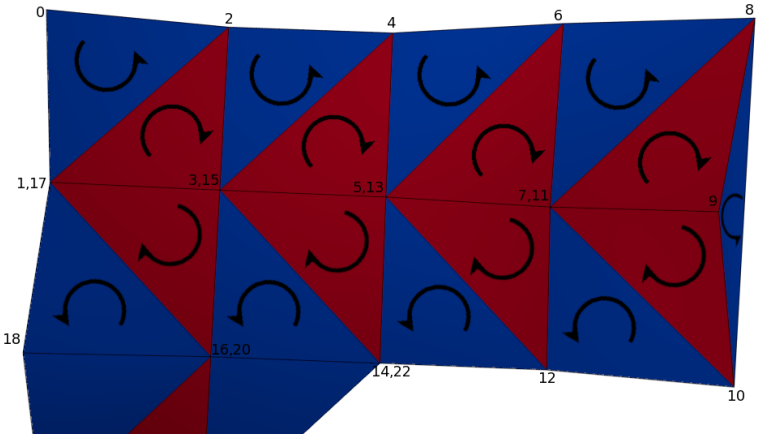
Beraz, puntua errepikatuz aurpegi ezkutuaren arazoa konpontzen da, baina begi-bistakoa ez den beste arazo bat sortzen da. Bigarren arazo hau ulertzeko, OpenGL-k aurpegien noranzkoa nola detektatzen duen jakin behar da.

Hirukiek bi aurpegi dituzte, aurrekoa eta atzekoa. Marrazketa optimizatzeko, atzeko aldeak ez marrazteko aukera dago, baina funtzionalitate hau ondo funtzionatzeko, aurpegi guztien noranzkoa ondo egon behar da. OpenGL-k aurpegi baten noranzkoa zein den erabakitzeko, hirukia osatzen duten puntuak zein ordenatan marraztu diren begiratzen du. Erlojuaren kontrako ordenean dituena aurreko aurpegia izango da.

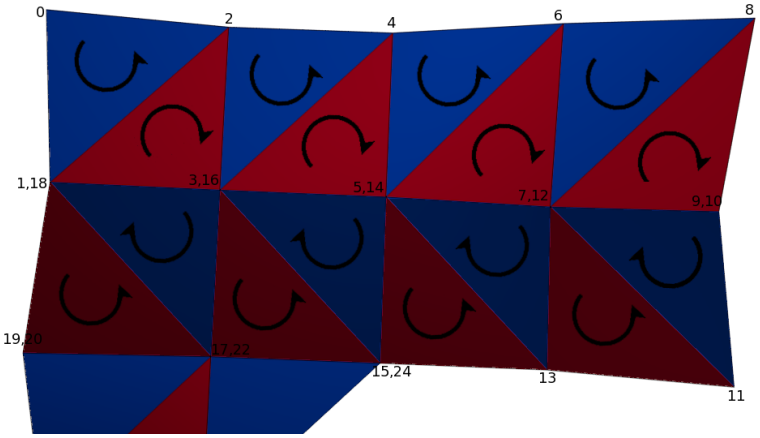
6.6 irudietan ikus daitekeenez, triangelu zerrendan, bigarrengo hirukia atzerantz begira gelditu beharko litzateke. OpenGL-k, zerrenda hauek ondo marraztearren, automatikoki hiruki bikoitiei buelta ematen die.

Puntu errepikatua sortzean, aurpegien ordena aldatzen da eta ilara bikoiti guztiak beste aldera begira gelditzen dira (6.6b irudia).

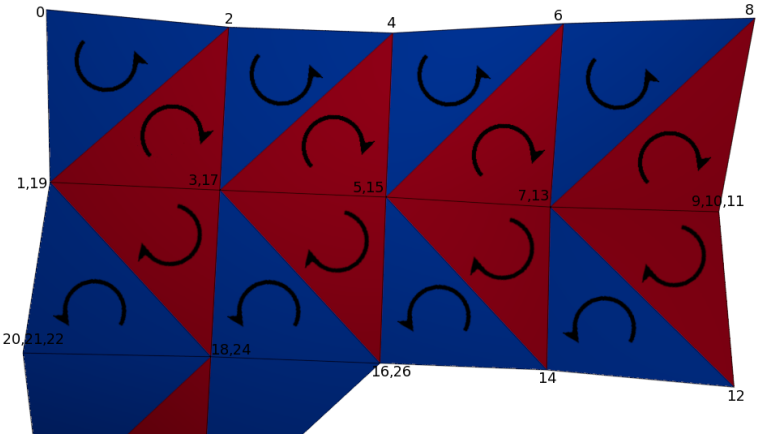
Guztiz ondo marrazteko bi aldiz errepikatu behar da puntua. Horrela bai, egon behar ez diren aurpegiak ekiditen dira eta aurpegien noranzkoa ondo mantentzen da (6.6c).



(a) Aurpegi ezkutua



(b) Aurpegi noranzko okerra



(c) Lurrazal egokia

Irudia 6.6: Triangelu zerrenden arazoak

6.2 Fitxategien karga

Irudien karga hainbat ataletan bana daiteke. Alde batetik fitxategietatik karga egongo da, irudia irakurtzeaz eta memorian cacheatzeaz arduratuko da. Beste alde batetik zati bati dagokion textura osatzea egongo da, aplikazioak erabili dezan.

Bi irudi mota kargatzen dira aplikazioan, batak lurrazalaren altuera informazioa gordetzen du, besteak lurrazal zatiari dagokion argazkia. Bien arteko ezberdintasun nagusiena fitxategi mota izango da. Fitxategiak behin eta berriz diskotik irakurri behar ez izateko, cache moduko bat antolatu da. Cachean orain arte irakurritako mapa zatien irudiak gordetzen dira. Argazkien kasuan, handiegiak direnez, bereizmen txikiagoko argazkiak bakarrik mantentzen dira cachean. Bereizmen handikoak ezin dira osorik irakurri, eta beraz, diskotik zatika irakurri beharko dira beharrezkoa denean.

6.2.1 Altuera informazioa

Altuera informazioa gris eskalako GeoTIFF motako irudietan gordetzen da. Pixel bakoitzaren intentsitateak puntu bateko altuera kodetzen du. Behar adina altuera ezberdin kodetu ahal izateko 16 bit erabiltzen dira pixel bakoitzeko. Horrez gain, bereizmena handitzearren, altuerak metrotan neurtu ordez, dezimetrotan adierazita daude. 6553,5 m neurtu daitezke beraz, Euskal Herriko altuera guztiak kodetzeko nahikoa.

libTiff liburutegia erabiltzen da TIFF irudien karga errazteko.

6.2.2 Argazkiak

Argazkiak JPEG formatuan gordeta daude. JPEG formatu honek konpresioa erabiltzen du normalean. Horrek irudien karga pixka bat zaildu egin du.

Adibidez, kameratik gertuen dauden zatiei bereizmen altuagoko texturak esleitu nahi zaie zehaztasun handiagoa izateko. Zoritzarrez, bereizmen handiko irudiak handiegiak dira osorik memoriak cacheatzeko, eta zatiak zuzenean diskotik hartzen dira. Konpresioa dela eta, ezin da irudiaren n. ilara irakurri aurretik dauden guztiak irakurri gabe. Horrek zatitxoen karga izugarri moteltzen du eta ezin da erabili (karga beste prozesu batean jartzen ez bada ez behintzat).

libjpeg liburutegia JPEG fitxategien irakurketaz arduratzen da, deskonpresio eta pixelen deskodetzea modu garden batean eginez.

6.3 Texturak

Textura OpenGL-k irudiei ematen dien izena da. Normalean 3Dko modelo bati kolorea emateko erabiltzen dira. Aplikazio honetan bi textura mota erabiltzen dira. Alde batetik argazkiak daude, lurrazala koloreztatzeko balioko dutenak. Bestetik altuera informazioa gordeko duen textura dago. Textura honek ez ditu koloreak gordeko, altuerak baizik. Shaderrean atzituko da puntu bati dagokion altuera zein den jakiteko.

Texturak sortzeko eman behar diren pausuak berdintsuak dira bi motentzako. Hala ere, aurrerago azalduko diren ezberdintasun batzuk daude.

6.3.1 Texturen karga

Lehenik eta behin, zein mapa zatiko pixelak irakurri behar diren erabaki behar da. Zati bakarretik izan daiteke, baina posible da ere bi edo lau ezberdinetatik kargatu behar izatea. 6.7 irudian ikus daiteke 4 mapetatik kargatu beharko den textura baten adibidea.

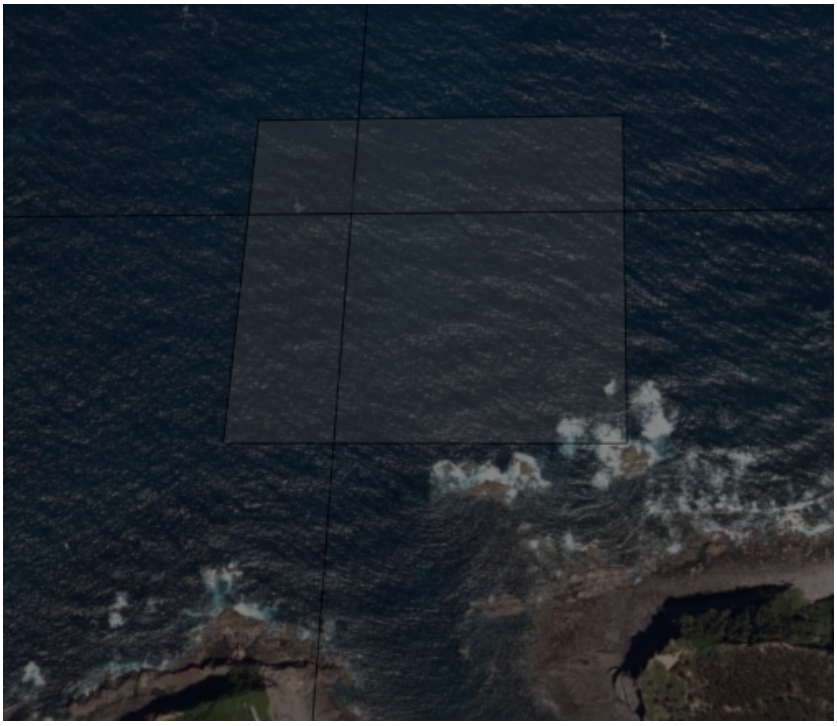
Karga egiterakoan beraz, mapa zati bakoitzetik behar diren pixelak hartuko dira eta irudi bakar bat osatzeko batu. Irudi horiek pasako zaizkio OpenGL-ri texturak sortzeko.

OpenGL-ko texturen tamainak biren berreturakoa izan behar du, baina erabiltzen ari garen irudiak ez dira normalean tamaina horretakoak izango. Horregatik, sortutako irudiak biren berreturako tamaina duen texturen barnean sartzen dira. Sobran dauden pixelak ez dira erabiliko.

6.3.2 Argazki texturak

Argazkiekin osatzen diren texturak trinkotu egiten dira memoria gutxiago erabiltzeko. ETC1 formatu trinkotua erabiltzen da. Trinkoketa hori erabiltzeak arazo bat sortzen du.

Lurrazala ondo marrazteko eta hainbat zatiz marraztu dela ez ikusteko, texturen arteko loturak perfektua izan behar du. Trinkoketa egitean, ertzetan



Irudia 6.7: Zati anitzetako textura

dauden pixelek, ondoan dutenen eragina izaten dute eta bere kolorea pixka bat aldatu egiten da. Ondoz ondoan dauden bi texturako ertz bakoitzak, auzokide ezberdinak dituzte, eta beraz, kolore aldaketa ez da berdina izango. Hori dela eta, trinkoketa erabiltzean, texturak elkartzen diren puntuan marra bat ikus daiteke (6.8).

Marratxo horiek kentzeko, pixel lerro gehiago kargatu behar dira, trinkoketa egiterakoan auzokideen informazioa edukitzeko. Texturak biren berreturako tamainakoak egitean erabili gabeko lekua gelditu denez, posible da pixel osagarri horiek gehitzea.

Textura koordenatuak kalkulatzean, kontuan hartu beharko da texturak pixel gehiago dituela.

6.3.3 Altuera texturak

Altuera texturarekin ere arazotxo bat dago. Altuera informazioa gris eskalako 16 bitetan ematen da Euskal Herriko altuera guztiak gorde ahal izateko. OpenGL ES sistemek, grisezko textura bakarra daukate eta 8 bit baino ez

Irudia 6.8: Texturen arteko lotura

ditu onartzen. OpenGL-k badu 16 biteko grisak erabiltzen dituen textura mota, baina hori ez dago erabilgarri OpenGL ES sistemetan.

Arazo hau konpontzeko, bi aukera aurkitu ziren. Koma higikorreko texturak erabiltzea, edo gris gehi alfa kanala duen textura erabiltzea. Koma higikorreko texturak ezin direnez mugikor guztietan erabili, alde batera utzi dira.

Alfa kanala duen gris eskalako irudia erabiltzea konponbide itsusia da, baina behintzat mugikor guztietan erabil daiteke. Textura mota honetan 8 bit erabiltzen dira gris intentsitatea adierazteko eta beste hainbeste alfarentzako. Alfa kanalak normalean gardentasuna adierazten du. Kasu honetan, 16bit gorde behar direnez, erdia griserako esleitutako bitetan kodetuko da, eta beste zortziak alfa kanalean. Shader programa bit horiek irakurri eta modu egokian kodetzeaz arduratu beharko da.

Tamalez, ez da hasieran uste bezain erraza (pisu gehieneko 8 bitak 256-rekin biderkatu eta besteei gehitu). Informazioa shaderrei pasatzerakoan, texturen pixelen informazioa koma higikorreko [0.0-1.0] tartera mapeatzen dira. Beraz, altueraren informazioa eskuratzea pixka bat zaildu egiten da. Hala ere, kalkulu gehiago egin behar badira ere, altuera eskuratzea posiblea da. 16 bitak konbinatu baino lehen, pixel informazioa berriz [0-255] tartera pasa baino ez da egin behar.

Kontuan izan behar da baita, koma higikorreko kalkuluek errorea sortzen dutela. Azkenean honako formula erabili da shaderrean balioa eskuratzeko:

$$altuera = \lfloor alpha * 255.0 + 0.5 \rfloor * 256.0 + \lfloor red * 255.0 + 0.5 \rfloor$$

6.4 Fitxategien deskarga

Fitxategien deskarga atzeko planoan egiten da, deskarga bakoitza hari berri batean. Horrela, deskarga egiten den bitartean, aplikazioak martxan jarraitzen du. Aldi berean gehienez bi deskarga egon daitezke. Hari bakoitzak hiru pausu betetzen ditu:

- FTP gunetik fitxategia eskuratu
- .zip fitxategitik irudi fitxategia destrinkotu
- Beharrezkoak ez diren fitxategiak diskotik ezabatu

Behin eta berriro fitxategi berdina ez deskargatzeko, fitxategia dagoeneko diskoan dagoen begiratu egiten da. Honek ahalbidetzen du aplikazioa Internetik gabe lan egiteak, betiere aplikazioak behar dituen fitxategiak SD txartelean badaude.

Oraingoz, deskargak Eusko jaurlaritzako webgunetik egiten dira. Fitxategiak oso handiak dira eta, gainera, eskuratu ondoren destrinkotu beharra dago. Oro har, gutxien optimizatutako zatia da hau. Hobekuntzen atalean jarri den bezala, zati hau hobetzeko era anitz daude.

6.5 Mapa zatien informazioa

Kokapen bati dagokion fitxategiak deskargatu eta irakurri ahal izateko, koordenatu horiek zein mapa zatiren barruan dauden jakin beharra dago. Esleipen hori egin ahal izateko zati bakoitzaren informazioa izan behar da.

Informazio hori eskuragarri dago shapefile motako fitxategietan.[1]Informazio hori erabili behar da zatien artean uneko posizioari dagokion zatia zein den jakiteko.

Esleipen hau frame bakoitzeko hainbat alditan egin behar da. Denbora gehiegi bilaketak egiten ez pasatzeko, pixka bat optimizatu nahi izan da.

[1]Euskadiko webgunean dauden fitxategiak ez dute zehatz-mehatz koadranteen informazioa adierazten. Horregatik, aplikazio lagungarri bat egin da, zati guztien benetako informazioa .shp fitxategi batean gordetzen duena.

Hasiera batean, bilaketa algoritmo hobeagoa erabiltzea pentsatu da, adibidez, quadtree moduko hierarkia bat osatuz, konparaketa kopurua murriztearren. Baina azkenean, beste metodo sinpleago bat erabili da.

Optimizaziorik sinpleena honako hipotesi honetan oinarritzen da: *"Aurreko koordenatuak zati batean kokatuta bazeuden, unekoak zati berdinean egongo dira."* Beraz, koordenatu berriak koadrante berdinean dauden baino ez da begiratu behar. Ez baleude, bilaketa normal osoa egin beharko litzateke.

Azkenean, hipotesi hori hedatu da are bilaketa gutxiago egin behar izateko: *"Aurreko puntua K zatian bazegoen, unekoa K-n edo K-ren auzokide batean egongo da."* Kasu honetan, koadrante guztien artean bilaketa bakarra egiten da, hasieran, aurretik non zegoen ez dakigun une bakarrean. Zati bat esleitu ondoren, horretatik ateratzen bada, norantz joan den begiratuko da, eta harantz dagoen hurrengo mapa zatia esleituko zaio.

6.6 Shaderrak

Aplikazioan oreka lortu nahi izan da memoria beharren eta exekuzio denboraren artean. Horregatik, eragiketa asko shaderretan bildu dira. Oro har, shaderrak txantiloitik puntuak hartuko ditu, puntu bakoitzari dagokion altuera eta posizio finala kalkulatuko ditu eta, normala lortu ondoren, argiztapena eta textura aplikatuko dizkio.

Shaderren kodea III eranskinean ikus daiteke.

6.6.1 Vertex Shader

Vertex shaderretan puntuei dagozkien azkeneko koordenatuak kalkulatu behar dira. Horrez gain, Fragment shaderrak beharko dituen parametro batzuk prestatzen ditu.

Aplikazio honetan, shader programa zati honek betekizun hauek izango ditu:

- Puntuaren azkeneko posizioa osatu.

  Texturatik puntuaren altuera atzituko da eta txantiloitik jasotako koordenatuei desplazamendua gehitu beharko zaio posizioa kalkulatzeko. Azkenik Modelview eta Projection matrizeak biderkatuko zaizkio pantailako koordenatuak lortzeko.

- Puntu horri dagokion normala kalkulatu.

  Fragment Shaderra argiztapena egiteko bektore normalak behar ditu, normalek adierazten baitute aurpegi bat nora begira dagoen, eta beraz, aurpegi bat argiztatu behar den edo itzaletan geldituko den. Normalak aurretik kalkulatu ez direnez, shader honetan egin behar da. Kalkulu gehiegirik egin behar ez izateko, normalak kalkulatzeko eragiketak sinplifikatu egin dira. Ikus II eranskina erabilitako metodoaren azalpena ikusteko.

- Argi direkzioa sortu.

  Fragment Shaderrak argiztapena egiteko beharko duen informazioa sortuko da. Vertex shaderrean sortzea erabaki da argia inoiz aldatzen ez delako. Argia aldatuko balitz, programatik jaso beharko lirateke balioak.

- Textura koordenatuak kalkulatu.

  Fragment Shaderrean textura koordenatuak aldatzea ez da batere eraginkorra, pixel bakoitzeko egin behar delako. Horregatik Vertex shaderrean egiten da.

Eragiketa horiek egin ahal izateko jaso behar duen informazioa asko da. Alde batetik, txantiloiko koordenatuak daude, baina beste batzuk ere behar dira.

- Modelview eta Projection matrizeak.
- Txantiloitik jasotako puntua.
- Altuera mapa duen textura.
- Altuera maparen hasierako koordenatuak.
- Altuera maparen tamaina.
- Altuera maparen pixelen artean dagoen distantzia.
- Argazki texturaren koordenatuak kalkulatzeko konstante bat.

Altuera texturarekin zerikusia duten parametro gehienak sinplifikatu zitezkeen argazkiekin egin denaren antzeko zerbait eginez. Hau da, aurretik konstante bat kalkulatuz eta posizioari konstante hori biderkatuz.

6.6.2 Fragment shader

Fragment shaderren betebeharra pixel bakoitzaren azkeneko kolorea kalkulatzea da. Kasu honetan argiztapen eta texturizazio shader estandar bat erabili da. Vertex shaderretik jasotako argi direkzioarekin gertatzen den bezala, argiaren kolorea ere finkoa da. Argiztapen eredu sinplifikatua egiten da. Lurrazal zati handia denez, erabaki da puntu espekularra sobran dagoela, eta beraz, ez da kalkulatzen.

Jasotzen dituen parametroak ezinbestekoak diren gutxi batzuk dira:

- Argazkia duen textura.
- Textura atzitzeko koordenatuak.
- Bektore normala.
- Argiaren direkzioa.

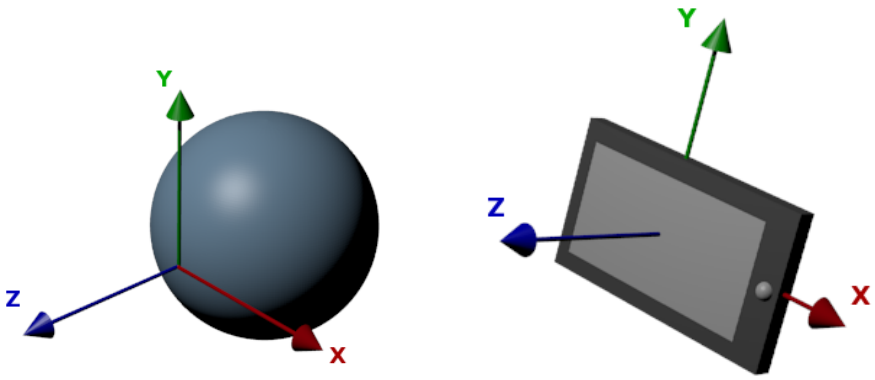
6.7 Telefonoaren orientazioa

Kameraren mugimendua egiteko, telefonoaren orientazioa atzitzen da. Alde batetik ipar-orratz elektronikoak emandako balioa dago eta bestetik, giroskopioaren balioak. Ipar-orratzak esaten digu iparra norantz dagoen, giroskopioak aldiz, ardatz bakoitzean zenbateko indarra jasotzen ari den. Bi informazio horiek konbinatuz mugikorraren errotazioa zein den kalkulatu daiteke.

Lurrazal birtuala telefonoa begira ari den norabide horretan dagoen benetako lurrazalarekin bat egitea nahi da. Horregatik, aplikazioan erabilitako datuak erreferentzi sistema horren pean marraztu behar dira. Oro har, errotazio matrizea sortzea baino ez da egin behar. Erraz lor daiteke hori Androidek matrize hori sortzeaz arduratzen den funtzioa baitauka. Matrizea lortu orduko, marrazten diren puntuak matrize horrekin biderkatu baino ez da egin behar.

Zoritxarrez, ez da hain erraza. Telefonoaren orientazioa adieraz ten duen matrizeak, OpenGL-k erabiltzen duen ardatz sistemaren (6.9b irudia) ezberdina den ardatz sistema bat erabiltzen du. Orientazioa munduaren koordenatu sistemaren (6.9a irudia) pean ematen da.

Irudietan ikusten den koordenatu sistemak berdinak badirudite ere, ezberdintasun garrantzitsu bat dago: OpenGL-n gorantz doan ardatza Y ardatza



(a) Munduaren koordenatu sistema (b) OpenGLren koordenatu sistema

Irudia 6.9: Koordenatu sistemak

da. Mundu koordenatuetan aldiz, Y ardatza iparrerantz doa, eta gorantz doana Z ardatza da.

Hori dela eta, lortzen den biraketa matrizea ezin da edonola erabili eta erreferentzi sistema aldaketa egin beharko da. Berriz ere, Androidek funtzio laguntzaile bat eskaintzen du funtzioa aldaketa hori egiteaz arduratuko dena.

7 Kapitulua

Hobekuntzak

Gaien Aurkibidea

7.1 Oinarrizko hobekuntzak

Erabilgarritasuna hobetuko duten hobekuntza anitz egin daitezke. Atal honetan aplikazio bukatuta eta argitaratzeko moduan egoteko beharrezkoak estimatu diren hobekuntzak azaltzen dira.

7.1.1 Fitxategien deskarga

Bi arazo nagusi daude inplementatu den deskarga sisteman.

Deskargatu behar diren fitxategiak oso handiak dira (argazkienak 25-30MB inguru). Fitxategi horiek deskargatzeko denbora asko behar da, eta ez da ezohikoa izango erabiltzailea itxaroteaz aspertzea eta aplikazioa ixtea. Deskarga guztiz amaitu ez bada, aplikazioa berriz irekitzerakoan, deskarga hasieratik hasten da. Deskargak berriz hasi ordez, gelditu egin den puntutik jarraitzea hobekuntza handia izango litzateke.

Konexioa galtzen denean automatikoki deskargatzen jarraitzea ere beharrezkoa da.

Horrez gain, tamaina horretako fitxategiak deskargatu behar izatea ez da onargarria. Gainera, deskargatutako fitxategiak destrinkotu behar dira eta horrek ere denbora asko behar du. Horregatik beste zerbitzari batetatik fitxategiak lortu beharko dira. Ahal bada bereizmen maila ezberdinak eskaintzen dituen zerbitzaria, nahi den bereizmena erabaki ahal izateko.

Mendian ez da beti 3G konexiorik egongo eta, baldin badago ere, mapa handiak direnez, aurretik deskargatuta izatea desiragarria litzateke. Erabiltzaile gidan azaldu den bezala, badago mapak aurretik deskargatzeko aukera, baina aplikazioak berak deskarga horiek egiteko erraztasunak ematea ere beharrezkoa litzateke.

7.1.2 Kameraren mugimendua

Kamera mugikorraren orientazioarekin aldatzea ondo dago, baina puntu baten inguruan dagoena ikusteko aukera baino ez du ematen.

Erabiltzaileari lurrazal birtualean zehar mugitzeko beste metodo batzuk eskaini nahi zaio. Adibidez, kameraren ikuspuntua gerturatzeko edo urruntzeko aukera emanez (zoom). Horrez gain, pantailan ikusi nahi duena agertzen ez bada eta hainbat metro urrunago zer dagoen ikusi nahi badu; bera haraino

joan behar gabe zer dagoen ikusteko aukera eman nahi zaio. Horretarako lurrazalean zehar behatzekin mugitzeko interfazea gehitu beharko da.

7.1.3 Koordenatuak eskuz sartu

Koordenatuak GPStik atzitu ordez, eskuz sartzeko aukera ere eman nahi da. Kameraren mugimendu librearekin lotzen bada, Euskal Herriko edozein eskualde bisitatzeko gai izango da erabiltzailea, bertara joan behar izan gabe.

Gainera, GPSrik ez duen mugikorretan exekutatzeko aukera emango luke honek.

7.1.4 Argazkiak desaktibatu

Nahiz eta argazkiak marraztuz informazio gehiago eskaintzen den, askotan altuera aldaketak ez dira ondo ikusten. Argazkirik gabe marrazten denean, altuera aldaketak nabariagoak dira. Argazkiak desaktibatzeko aukera eman daiteke, erabiltzaileak nahi badu kentzeko aukera izan dezan.

Horrez gain, fitxategi handienak argazkiak direnez, altuera bakarrik marrazteko aukera ematean, fitxategi horiek deskargatu behar ez izatea ahalbidetzen du. Adibidez, fitxategiak aurretik deskargatu ez badira, agian erabiltzaileak ez du 3G konexioaren kuota argazkiekin xahutu nahiko.

7.2 Hobekuntza aurreratuak

Badaude guztiz beharrezkoak ez diren hobekuntza batzuk, baina erabilgarritasun aurreratua emango dutenak.

7.2.1 Bereizmen ezberdineko argazkiak

Kameratik gertuago dauden zatien argazkia nahiko pixelatuta eta lausoa ikusten da. Efektu hori gutxitzeko interesgarria litzateke gertuen dauden zatiei bereizmen altuagoko argazkiak esleitzea. Halaber, urrunago dauden zatiei bereizmen txikiagokoak jar ahal zaizkie.

7.2.2 Irudien karga optimizatu

Lurrazala zati txikitan banatuta dago erabiltzailea mugitzen denean informazio gutxiago kargatu ahal izateko. Izan ere, lerro bat osatzen duten zatitxoak bakarrik kargatu behar dira. Hala eta guztiz ere, 16 textura kargatzea eta sortzea nahiko motela da. Horregatik irudien karga nolabait optimizatu zitekeen.

Hari ezberdin batean

Texturen kargak marrazketa ez moteltzeko, karga guztiak hari ezberdin batean egin daitezke. Metodo hau erabiltzen bada, harien arteko sinkronizazioa gehitu beharko da, textura kargatzen noiz amaitu den jakiteko.

Frame bakoitzeko karga kopurua mugatu

Beste aukera bat kargatu behar diren texturen zerrenda bat sortzea da. Frame bakoitzean, zerrenda horretatik textura bat edo bi bakarrik kargatuko dira. Horrela, segundoko frame kopurua ez da gehiegi murriztuko, frame batean kargatuko den textura kopurua mugatuta egongo delako.

7.2.3 Bereizmen handiagoko altuera fitxategiak

Aplikazioak erabiltzen dituen altuera irudiak nahiko ondo daude. Gordetzen dituen altueren artean 5 metroko aldea baino ez dago. Azken urteetako altuera fitxategiek aldiz, 1mko aldea eta zentimetrorako bereizmena eskaintzen dute. Aplikazioan erabili ez badira, bi arrazoi nagusiengatik izan da:

- Testuzko fitxategiak dira.

  Altuerak irudi baten barruan kodetu ordez, testuzko fitxategietan gordetzen dira. Testuzko fitxategiak irakurtzea dezente motelagoa da, karakterez karaktere irakurri behar direlako. Proba egin zenean, fitxategi bakar bat irakurtzeko hainbat minutu behar izan ziren eta, gainera, memoria arazoak ematen zituen.

- Izugarri handiak dira.

  Fitxategi horietan, 5x5mko bereizmena duten fitxategietan baino 25 aldiz informazio gehiago gorde behar da. Horrez gain, testuzko fitxa-

tegiak direnez, irudiek baino askoz byte gehiago behar dute. Adibidez, 800 KB-ko altuera irudia duen zati batek, 65MB inguru behar ditu fitxategi berriekin.

Hala eta guztiz ere, beste fitxategi mota batean gordeko balira, bereizmen handiagoko altuerak erabiltzea gomendagarria da. Alde batetik informazio eguneratua erabiltzeko aukera ematen duelako. Bestetik, marraztutako lurrazala zehatzagoa izango delako, eta beraz, seguruagoa.

7.2.4 Argiztapena

Lurrazala argiztatzeko argi finko bat erabiltzen da. Ikusgarria izango litzateke argiaren norabidea orduarekin aldatzen joatea eguzkia balitz bezala. Goizean ekialdetik argiztatuko litzateke eta arratsaldean aldiz, mendebaldetik.

7.2.5 Altuera aldaketak nabariagoak egin

Altuera aldaketak ez dira beti ondo ikusten. Altuera aldaketa horiek nabariagoak egitea erabilgarria izan daiteke. Horretarako irudietatik atzitutako altuerak eskalatu daitezke, bi puntuen arteko diferentzia handiagoa izateko.

Agian erabiltzaileari zein eskala erabili nahi duen adierazteko aukera eman ahal zaio.

7.2.6 Arriskuen detekzioa

Segurtasun hobekuntza da hau. Bi puntuen artean altuera ezberdintasun handia dagoen detektatzean, abisu bat eman ahal zaio erabiltzaileari. Istripuak ekiditeko balio dezake, gertu amildegiren edo desnibel handi bat egotekotan, erabiltzailea jakinaren gainean egongo delako.

Adimen artifiziala

Arriskuen detekzioarekin lotuta, mendia jaisteko segurua den bide bat bila zitekeen erabiltzailea bertatik bideratzeko.

Agian, bidea aplikazioak sortu ordez, mendi bakoitzak dituen bideetako bat kargatu zitekeen eta lurrazalaren gainean marraztu. Erabiltzailea bidetik urruntzen denean, gertueneko bidera ailegatzeko nondik joan beharko den adieraz zitekeen.

7.2.7 Bidea marraztu

Track-ak gordetzeko aplikazioak oso popularrak dira mendizaleen artean eta hori gehitzea hobekuntza bat izan daiteke. Koordenatuak gordetzeaz gain, egin den bidea lurrazalean marraztea lagungarria izan daiteke.

Aurretik sortutako *track*-ak kargatzeko aukera ere eman daiteke, erabiltzaileak bidea nondik doan jakin dezan.

8 Kapitulua

Ondorioak

Gaien Aurkibidea

8.1 Ondorioak

Karrera amaierako proiektu honek lainoagatik mendian galdutako mendizale bati bidea aurkitzeko aukera eman nahi zion. Horretarako 3Dko aplikazio bat egitea erabaki zen.

Hasierako helburu guztiak bete egin dira: aplikazioa erabiltzailearen kokapena lortzeko gai da, behar dituen argazkiak kargatu egiten ditu eta puntu horri dagokion ingurunea pantailan marrazten du. Gainera, marrazten den modeloan lurrazalaren altuera aldaketak agerikoak dira.

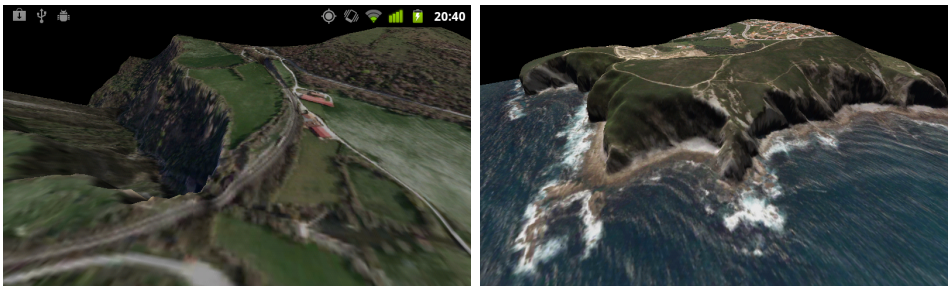
Horrez gain, beste helburu batzuk ere bete dira. Hala nola, fitxategiak zerbitzari batetik deskargatzea edo kamera erabiltzailearen orientazioaren arabera aldatzea.

Hobekuntza ugari egin daitezke baina, oro har, helburu teknikoak bete direla uste da.

Beste helburuak ere bete egin direla uste da. Arlo askotako ezagutzak eskatzen dituen aplikazioa da. Karreran zehar ikasitakoak praktikan jarri behar izan dira eta klasetik kanpo nire kabuz ikasitako zerbait ere. Horrez gain, beste gauza berri asko ikasi behar izan dira proiektua aurrera eraman ahal izateko (adibidez JNI edo UTM koordenatuak).

Planifikatutako eta ordu kopuru errealaren artean egon den aldea harrigarria suertatu zait. Espero nuen planifikatutako baino ordu gehiago ematea proiektua egiten.

Oro har egindako lanarekin pozik nago, eta aplikazioak sortzen dituen paisaia batzuk harrigarriak iruditzen zaizkit. Agian Euskal Herriaren meritoa izango da hori.



(a) Gujuli urjauzia (b) Bizkaiko kosta

I Apendizea

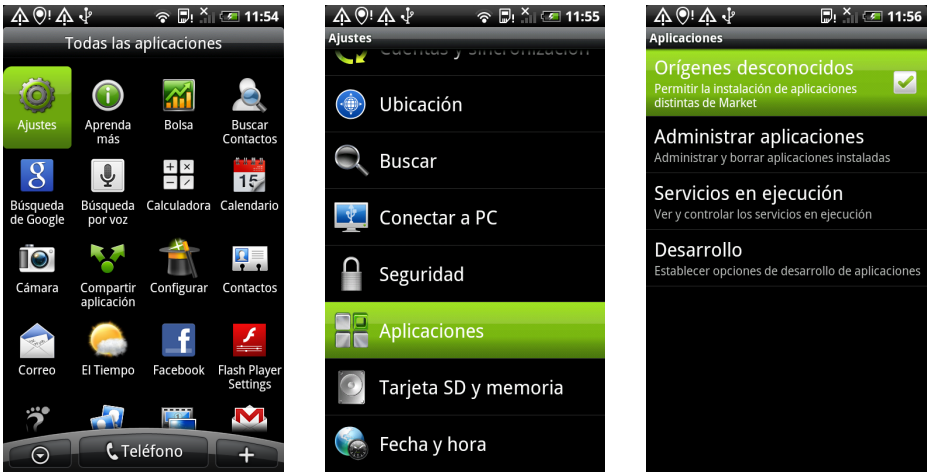
Erabiltzailearen eskuliburua

Gaien Aurkibidea

I.a Aplikazioaren instalazioa

Aplikazio hau instalatu ahal izateko, Market-aren kanpoko aplikazioak instalatzeko baimena eman behar da.

Horretarako "Ajustes → Aplicaciones → Orígenes desconocidos" aukera aktibatu beharra dago.



Irudia I.1: Iturburu ezezagunak ahalbidetu

Aplikazioa instalatzeko hainbat pausu eman behar dira.

1. .apk fitxategia SD txartelera pasa.

   SD txartelera pasa ahal izateko, telefonoa USB bidez ordenagailura konektatu behar da, fitxategiak konpartitzeko moduan jarri eta EMendiK.apk fitxategia mugikorrean kopiatu.

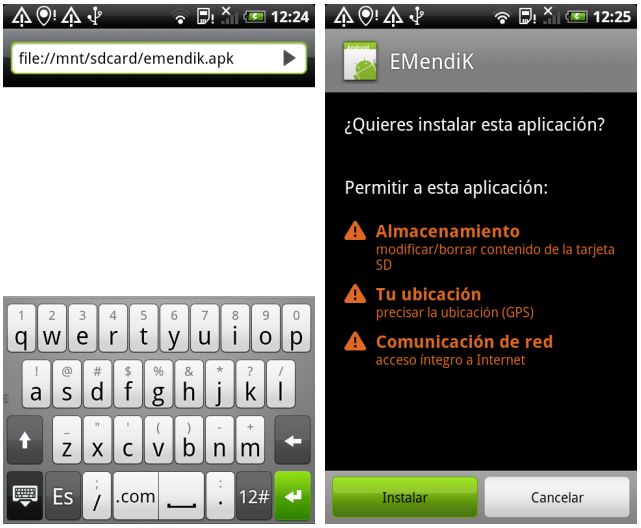
2. Web arakatzailearekin fitxategi hori ireki.

   Android defektuz ez dakar fitxategiak kudeatzeko aplikaziorik, horregatik kopiatutako fitxategia atzitzeko web arakatzailea erabiliko da. "file://mnt/sdcard/EMendiK.apk"[1] helbidera joanez gero, aplikazioa instalatzeko pantaila agertuko da.

3. Aplikazioa instalatzeko baimena eman.

   Aplikazioak behar dituen baimenak agertzen dira pantaila honetan. Ados egonez gero, aplikazioa mugikorrean instalatuko da.

[1] Helbide hori SD txarteleko goiko direktorioan kopiatu bada balio du bakarrik. Beste karpeta batean gorde bada, helbidea ezberdina izango.



(a) Fitxategia ireki (b) Baimenak

Irudia I.2: Aplikazioaren instalazioa

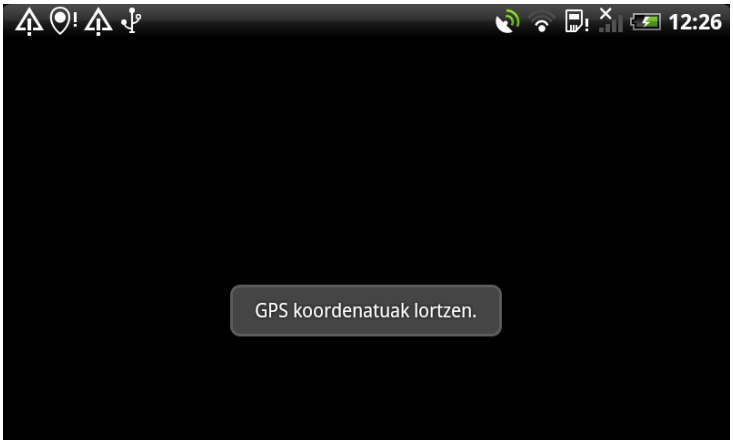
I.b Aplikazioa exekutatu

Aplikazioa abiarazi baino lehen, GPS sistema aktibatu behar da. Menuaren "Ajustes → Ubicación → Utilizar satélites GPS" aukerarekin martxan jar daiteke.

Irudia I.3: GPSaren aktibazioa

GPSa aktibatu orduko aplikazioa exekutatu daiteke.

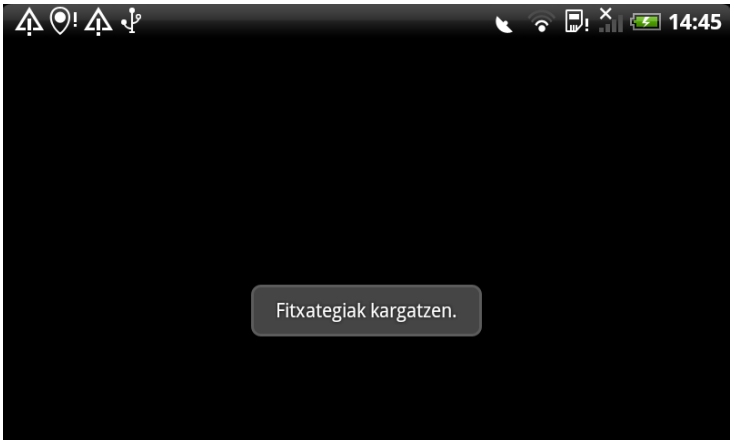
Hasi bezain pronto GPS kokapena lortzen saiatzen da. Une batez abisu bat pantailaratzen du zertan ari den adierazteko. Gero, posizioa lortu arte pantaila beltza gelditukо da.



Irudia I.4: GPSari itxaroten

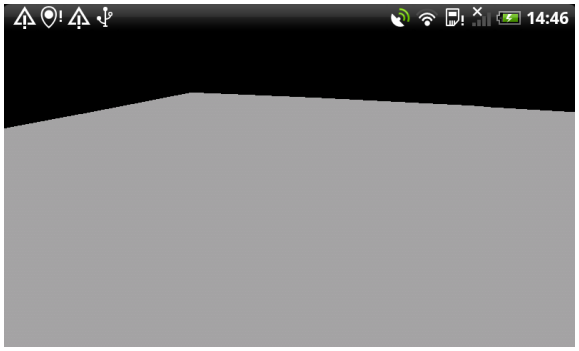
GPSaren ikonoa dir-dirka mantenduko da sateliteak bilatzen dituen bitartean. GPS posizioa lortzen denean aldiz, ikonoa piztuta gelditukо da.

Kokapena lortu orduko, fitxategiak kargatzen hasiko da. Beste abisu bat pantailaratuko da karga hasten denean.



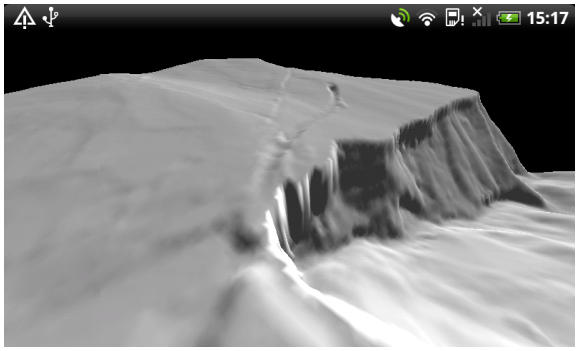
Irudia I.5: Fitxategiak kargatzen

Fitxategiak prest ez badaude, deskarga hasiko da eta karratu zuri bat baino ez da marraztuko.



Irudia I.6: Karratu zuria

Fitxategiak eskuratzen diren heinean pantailan agertzen joango dira. Hasiera batean altuera fitxategiak jasoko dira eta lurrazal zuria marraztuko da.



Irudia I.7: Altuera bakarrik

Fitxategi guztiak prest daudenean lurrazal osoa marraztuko da.

Irudia I.8: Lurrazala

I.c Fitxategien aurretiko deskarga

Fitxategiak aurretik deskargatzea gomendagarria da. Argazkiak behintzat aurretik eskuratzea ezinbestekoa da.

I.c.1 Fitxategien aukerapena

Lehenik eta behin, zein fitxategi deskargatu behar diren jakin behar da. Horretarako GeoEuskadi bisorea[2] erabil daiteke.

Bisorean deskargatu nahi den tokian zoom egin. Ezkerreko menuan "Tresnak"menua zabaldu, "Kartografiaren deskarga-n "Hojas 5000"aukeratu behar da eta "Aktibatu"botoian klik egin.

Irudia I.9: Mapa zatien marrazketa aktibatu

Deskargak aktibatzeak mapak deskargatzeko aukera ematen du. Baina 2010 eta 2011 urteetako fitxategiak deskargatzeko aukera baino ez du ematen. Urte batetik bestera mapa zatien kokapena pixka bat aldatu egiten da eta aplikazioan erabiltzen den altuera informazioa 2009 urtekoa da. Altuera eta argazkiak bat egiteko urte berdinekoak izan behar dira. Horregatik 2009 deskargatzea aholkatzen da.

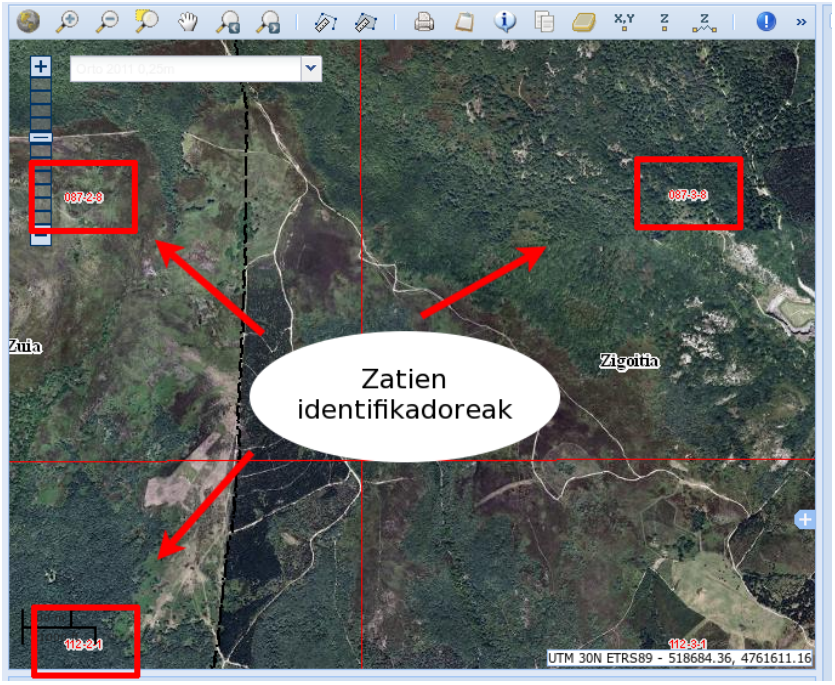
Deskargatu nahi diren zatien identifikadorea apuntatu (I.10 irudia).

I.c.2 FTP webguneak

FTP deskarga gunea ireki behar da orain. Webguneak honako hauek dira:

- Argazkiak:
  ftp://ftp.geo.euskadi.net/cartografia/Cartografia_Basica/Ortofotos/ORTOS_2009/5000/

[2]Bisorearen webgunea:
http://www.geo.euskadi.net/s69-bisorea/eu/x72aGeoeuskadiWAR/index.jsp?lang=eu



Irudia I.10: Zatien identifikadoreak

- Altuera:
  ftp://ftp.geo.euskadi.net/cartografia/Cartografia_Basica/MDE/MDT_2009/5000/

Eskuragarri dauden fitxategietatik lehen identifikatutakoak deskargatu behar dira.

Irudia I.11: FTP webgunea

I.c.3 Irudiak mugikorrera kopiatu

Zip fitxategiak destrinkotu eta .jpg eta .tif irudiak android mugikorraren SD txartelera pasa behar dira. Txarteleko honako direktorioetan gorde behar dira:

- Argazkiak (.jpg):
  Android/data/es.mt4.EMendiK/cache/foto/

- Altuera (.tif):
  Android/data/es.mt4.EMendiK/cache/mde/

Fitxategiak kopiatu orduko aplikazioa abiarazteko prest egongo da.

II Apendizea

Normalen kalkulua

Gaien Aurkibidea

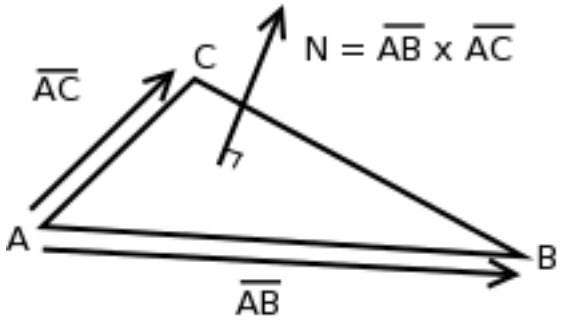
II.a Normalen kalkulua

Bektore normala aurpegi baten elkarzuta dagoen bektorea da. 3D irudigintzan asko erabiltzen da, gehienbat modeloen argiztapena egin ahal izateko. Izan ere, aurpegi bat argiztatu behar den edo itzaletan gelditukо den erabakitzeko, aurpegia argiari begira dagoen ala ez jakin behar da.

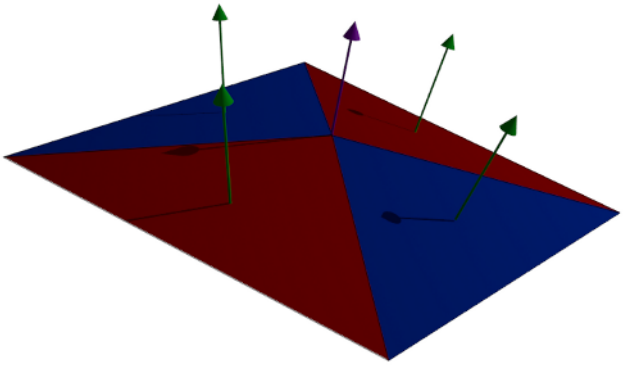
Aurretik sortutako modeloetan normalak ematen dizkigute. Kasu honetan, lurrazalaren modeloa prozeduralki sortzen da, eta beraz, bektore normalak ere kalkulatu beharra dago.

Hiruki baten bektore normala kalkulatzea nahiko erraza da: hiru erpinekin bi bektore sortu eta haien biderketa bektoriala egin. II.1 irudian triangelu baten normalaren kalkulua ikus daiteke.

Puntu baten normala kalkulatzeko, puntu hori parte hartzen duen hiruki guztien normalen batezbestekoa egiten da.



Irudia II.1: Triangelu baten bektore normala



Irudia II.2: Puntu baten bektore normala

Gure lurrazalan puntuen normalak kalkulatzeko beraz, honako algoritmoa erabil daiteke:

1. Puntu auzokideekin bektoreak sortu
2. Ondoz-ondoko bektoreen biderkaketa bektorialak egin
3. Normal guztiak batu
4. Emaitza normalizatu

Laburtuz, erabiltzen den formula honako hau izango da:

$$N = normalize[(\vec{O} \wedge \vec{L}) + (\vec{R} \wedge \vec{O}) + (\vec{L} \wedge \vec{I}) + (\vec{I} \wedge \vec{R})]$$

non $\vec{O}$, $\vec{I}$, $\vec{R}$ eta $\vec{L}$ zentroko puntuaren eta ondoko puntuen artean sortzen diren bektoreak diren.

II.b Normalen kalkulu azkarra

Normalen kalkulua shaderrean egiten denez, frame guztietan marrazten den puntu bakoitzeko egin beharko da. Aurreko atalean azaldu den algoritmoan, 4 biderkaketa bektorial eta normalizazio eragiketa bat egin behar dira. Frame guztietan hainbeste eragiketa egitea astunegia izan daiteke.

Normalak kalkulatzeko formula garatu eta sinplifikatzen bada, askoz azkarragoa izango den formula bat lor daiteke.

Formula berri hori honakoa da:

$$N = normalize(C - A, 2.0 \times s, D - B)$$

Kasu honetan, A, B, D eta C ondoko puntuen altuera da, eta s ondoz-ondoko bi puntuen artean dagoen distantzia.

Begibistakoa da metodo honek askoz eragiketa gutxiago behar dituela, eta beraz, shader batean exekutatzeko egokiagoa dela.

[0]http://www.gamedev.net/topic/163625-fast-way-to-calculate-heightmap-normals/

III Apendizea

Shaderren kodea

III.a Vertex Shader

```
precision mediump float;
uniform mat4 mvMatrix;
uniform mat4 projMatrix;
varying vec3 fnormal;
varying vec3 sundir;
varying vec2 ftexcoord;
uniform vec4 offset; // (x,y) are offset of the quadrant (z,w) are steping
attribute vec2 position;
uniform vec2 hmappos;
uniform sampler2D heightmap;
uniform vec4 texsize;// (x,y) is size of heightmap, (z,w) constant to multipy to position to calcul

float getHeight( vec2 pcoor )
{
    vec2 heightcoor = vec2( (offset.x+pcoor.x-hmappos.x)/texsize.x, (offset.y+pcoor.y-hmappos.y)/te
    vec4 p = texture2D( heightmap, heightcoor.st );
    return (floor( p.a*255.0+.5)*256.0 + floor( p.r*255.0+.5))*.1;
}
vec3 computeNormal( vec2 p )
{
    // Fast normal calculation
    float A = getHeight(vec2(p.x+1.0,p.y));
    float B = getHeight(vec2(p.x,p.y+1.0));
    float C = getHeight(vec2(p.x-1.0,p.y));
    float D = getHeight(vec2(p.x,p.y-1.0));
    vec3 N = vec3( C - A, 2.0 * offset.z, D - B );\
    return normalize( N );
}

void main()
{
    sundir = normalize(vec3( mvMatrix * vec4(vec3(1.0, 1.0, .0), 0.0) ));
    float alt = getHeight(position);
    vec3 n = computeNormal( position );
    fnormal = vec3( mvMatrix * vec4(n, 0.0) );
    vec4 pos = vec4( (position.x+offset.x), (position.y+offset.y), alt, 1.0 );
    gl_Position = projMatrix*mvMatrix * pos;
    ftexcoord = (position+vec2(4.0,4.0)) * texsize.zw;
}
```

Kodea 1: Vertex Shaderra

III.b Fragment Shader

```
precision mediump float;
varying vec3 fnormal;
varying vec3 sundir;
uniform sampler2D ortofoto;
varying vec2 ftexcoord;
void main() {
    vec4 ambient = vec4( .2, .2, .2, 1.0 );
    vec4 diffuse = vec4(1.0, 1.0, 1.0, 1.0);
    vec3 n = normalize(fnormal);
    float NdotL = max (0.0, dot(n, sundir));
    vec4 color = texture2D( ortofoto, ftexcoord );
    gl_FragColor = color * (ambient + diffuse * NdotL);
}
```

Kodea 2: Fragment Shaderra