UPV/EHU-ko informatika fakultatearen bisita birtuala

Karrera-bukaerako proiektua

Iker Martin Urriza

Gainbegiraleak:

Joseba Makazaga Odria

Aitor Soroa Echave

ESKERRAK

Lehenik eta behin, eskerrak eman nahi dizkiet proiektu honen garapenean eta kudeaketan hainbeste lagundu nauten irakasle eta proiektu zuzendari diren Aitor Soroa eta Joseba Makazagari. Beren ideien, laguntza eta motibaziorik gabe proiektua eta garatutako aplikazio hau ez lirateke orain direna izango.

Bestalde, nire lagun eta klase-kideei eskerrak eman behar dizkiet beren ordenagailuetan probak egiteko emandako aukerarengatik, eta barkatu behin baino gehiagotan zuen ekipoak blokeatzeagatik.

Azkenik, milesker proiektuaren garapen guztian zehar nire ondoan egon, hainbeste animo eman eta hainbeste pazientzia izan duzuen guztiei.

AURKIBIDEA

Taula aurkibidea

Irudi aurkibidea

Kode aurkibidea

1-SARRERA

1.1- Aurkezpena

1.2- Gai nagusiak

1.1-Aurkezpena

Gaur egun aplikazio informatikoak arlo guztietan aurkitzea guztiz ohikoa da. Gizartean ohituta gaude jada mota guztietako aplikazioak ikustera, testu-editore sinple batetik fisika simulatzaile konplexu bateraino. Honekin batera, aplikazioek informazio bisuala ematera ohituta gaude, eta hiru dimentsiotako objektu eta irudien bidez bada hobeto. Jakina da irudi politek jendea liluratu eta erakarri egiten dutela.

Eguneroko bizitzan laguntza eskaintzen diguten aplikazioen kopurua handitzen doan heinean, erabiltzaile arruntentzat erabilgarri edo lagungarriak izan daitezkeen ideia eta aplikazio berriak sortzen dira. Erabiltzaileak aplikazioen bidez erosotasun sentsazioa lortzea gogoko duela esan daiteke, edota normalean egin ez ditzaken ekintzak nolabait birtualki burutzea.

Azken ideia honetatik dator proiektu honetan aurkezten den aplikazioaren motibazioa. Baina, hasi baino lehen, zein da proiektu honetan garatu nahi den aplikazioa? Donostiako UPV/EHU-ko informatika fakultatearen bisita birtual bat egitea ahalbidetzen duen web aplikazio bat hain zuzen.

Fakultatea bisitatzeko aukera edo denbora errealik ez daukaten pertsonei beren etxeetatik hau bisitatzeko aukera eskaintzea da helburua. Zerbitzu honen bidez, erabiltzaileak fakultateko pasillo eta gelak bisitatzeko gai izango da. Honek erabiltzaileari eskaintzen dion erosotasun sentsazioa benetan handia da, etxetik mugitzeko beharrik gabe dena ikusi baitezake.

1.2-Gai nagusiak

Proiektuaren azken helburua web aplikazio grafiko elkar-eragile bat garatzea da; hau da, erabiltzaileak fakultatea bisitatzen duen bitartean, bertako objektu batzuekin "jolastu" edota elkar-eragiteko aukera izango du. Adibidez, pasilloetan mugitzen den bitartean, bertan aurkitzen dituen ateen bidez gela ezberdinetara iritsi daiteke ate hauetan 'klik' eginez.

Proiektuan aztertu eta garatuko diren gai garrantzitsuenak hauek dira:

- **WebGL teknologia:** WebGL-ri eta HTML5-i esker, web orrietan 3D objektuak marraztu eta aurkezteko aukera daukagu.
- **3D modelatzea:** Aplikazioan erabiliko diren 3D objektuak sortzeko teknika ezberdinak aztertu eta modelatuko dira.
- **Errealismoa mantentzen:** Erabiltzaileak fakultatea bisitatzen duen bitartean, paretak ez zeharkatzea eta eskaileren bidez solairuz aldatzea nola detektatu eta kudeatzeko erabiltzen diren teknika eta algoritmoak aztertuko dira, beranduago garatu ahal izateko.

Hau zehaztuta, hurrengo pausoa jakina da: Lanean hasi.

2-PROIEKTUAREN HELBURU-DOKUMENTUA

2.1- Helburuak

2.2- Plangintza

2.3- Arriskuak

2.4- Garapenerako tresnak

2.5- Aurrekontua

2.1-Helburuak

Karrera amaierako proiektu honen helburua web bidez atzigarria izango den UPV/EHU-ko informatika fakultatearen hiru dimentsioko bisita birtual bat inplementatzea da. Bisita birtual honen bidez edozein erabiltzaileri fakultatea bisitatzeko aukera eskainiko zaio.

Fakultatearen bertsio birtuala sortzeko hiru dimentsioko modeloak erabiliko direnez, web bidez zerbitzu hau eskaini ahal izateko WebGL eta HTML5 teknologiak erabiliko dira. Honi esker hiru dimentsioko modeloak web orri batean kargatu eta kudeatu ahal izango dira, Javascript bidez programatuz. Teknologia hauen erabilerak eta web nabigatzaile baten beharra soilik izatea hainbat abantaila eskaintzen ditu:

- Ez dago aplikazioa instalatzeko beharrik, zerbitzu hau web bidez eskaintzen baita. Ondorioz, aplikaziorako behar diren fitxategi guztiak web zerbitzari batean egotearekin nahikoa da. HTML eta Javascript-en bidez erabiltzaileak nabigatzaile baten bidez atzituko aplikazioa.
- HTML5 eta WebGL onartzen duen edozein nabigatzaile erabili daiteke. Honi esker, aplikazioa honelako nabigatzaile bat onartzen duen edozein sistema eragiletara portablea izango da.
- Aplikazioak WebGL teknologia erabiltzeko nahikoa dut HTML5 eta Javascript. Ondorioz, erabiltzaileak ez du bere nabigatzailearendako plugin-ik instalatu behar.

WebGL-rekin lanean hasi aurretik, teknologia honen programazioa errazteko sortu diren motore edo liburutegien inguruan ikerketa bat burutuko da, proiektu honetarako aukera egokiena egin ahal izateko.

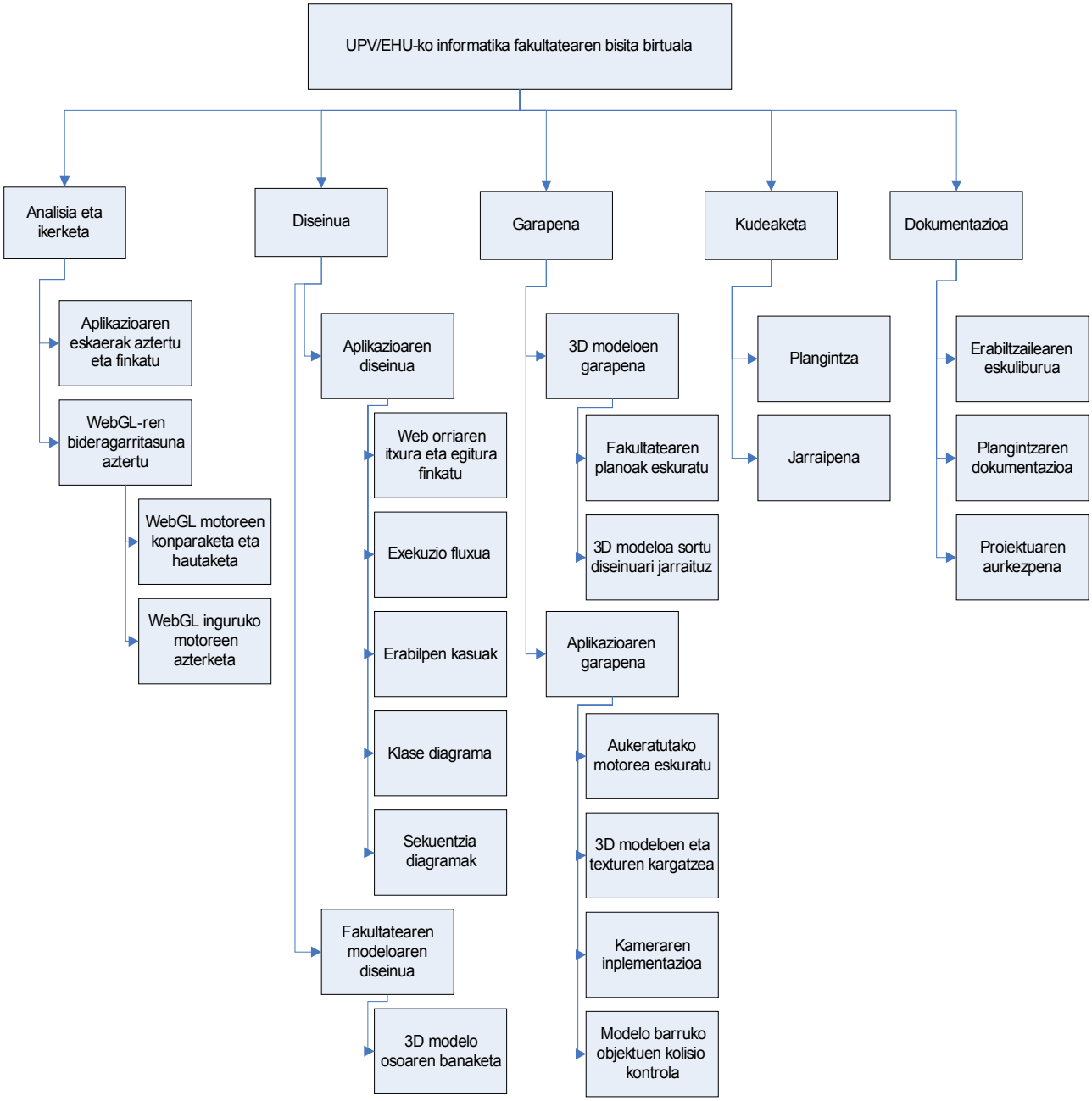
Hiru dimentsioko modeloak eraikitzeko Blender aplikazioa erabiliko da. Modelo hauek errealitateko egiturekiko ahalik eta fidelen mantenduko dira, eta zehaztasun maila ertain bat bete beharko da. testurak erabiliko dira gehienbat pareta eta lurrari itxura egoki bat ezartzeko. Hasiera batean ez da fakultate osoa modelatuko. Etorkizun

batean modeloak gehitzeko aukera izango da beti ere.

Lehen pertsonako ikuspegian ikusiko dira modeloak, beraz kamera bat erabiliko da hauen barruan mugitu ahal izateko. Objektuen arteko kolisioak kontuan izango dira kamera paretetan zehar ez pasatzeko.

2.2-Atazen plangintza

Diagrama honetan proiektuko atal ezberdinen laburpen bat ikusi daiteke:



Irudia 1: Proiektuaren LDE diagrama

Diagraman ikusten denez, 3D modeloen eta kodetze lanaren arteko desberdintasunak ongi zehaztu dira, hainbat ataza paraleloan burutzeko aukera baitago, eta, orokorrean, beraien artean independenteak dira bi multzo hauek. Bestalde, kontuan izan beharrekoa da diagraman ez dela atazen ordena definitzen.

Ondoko tauletan proiektuko ataza ezberdinen deskripzioak aurkeztuko dira. Proiektuan giza-baliabide bakar bat dagoenez (ikaslea), ataza guztietan giza-baliabideen kopurua ez da adieraziko. Arrazoi berdinagatik, ataza bakoitzeko arduradunak ez dira zehaztuko, ikaslea bera izango baita ataza guztien arduradun.

Ataza bakoitzeko honi dagokion estimatutako denbora (ordutan) eta deskribapen labur bat bilduko dira, beharrezko informazioa erraz eta azkar ulertu ahal izateko.

ANALISIA ETA IKERKETA:

Ataza multzo honetan proiektuan erabili beharko diren teknologia eta tekniken inguruko ikerketak burutuko dira. Hauekin batera, garapenerako beharko diren tresna ezberdinak zein izango diren pentsatu eta hautatu beharko da.

Aplikazioa WebGL-en oinarritzea denez helburua, teknologia honek eskaintzen dizkigun ezaugarriak aztertuko dira, proiekturako benetan bideragarria dela finkatzeko helburuarekin.

Teknologia honen gainean sorturiko liburutegi bat erabiltzea denez ideia nagusia, WebGL-ren erabilera errazten duten motore edo liburutegi ezberdinak aztertu beharko dira lehenik, gure proiektura ondoen egokitzen dena aurkitzeko helburuarekin. Motoreen ezaugarrien arteko konparaketa baten bidez hartuko da azken erabakia.

Ataza	Esfortzua	Deskripzioa
Aplikazioaren eskaerak aztertu eta finkatu	2 ordu	Aplikazioaren eskaerak aztertuko dira erabiliko diren teknologiak eta helburuak zehazteko.
WebGL-ren bideragarritasuna		
Motoreen azterketa	10 ordu	WebGL motoreen inguruko azterketa bat egin eta ezaugarriak bildu.
Motoreen konparaketa eta hautaketa	5 ordu	Motore bakoitzaren alde onak eta txarrak aztertu. Motorea aukeratu.
Totala:	17 ordu	

Taula 1: Analisi eta ikerketa atazen taula

DISEINUA:

Aplikazioaren diseinu egoki bat garrantzitsua da, garapenean inplementazio arazoak ekidin ahal izateko. Honetarako software ingenieritzan ohikoak diren atalak garatuko dira.

- Erabilpen kasuen diseinua.
- Exekuzio-fluxuaren diseinua.
- Klase-diagramaren diseinua.
- Erabilpen kasu nagusien sekuentzia diagramen diseinua.

Baina programaren inplementazioa zehaztuko duen diseinuaz aparte, fakultatearen 3D modeloaren diseinua eta aplikazioaren web orriaren diseinua ere zehaztu beharra dira.

- 3D modelo osoaren zatiketa zein izango den zehaztu beharra dago, aplikazioan modelo handi bat kargatzeak izan ditzaken ondorioak ekiditeko.
- Web orriaren itxura zehaztu behar da, baita bertan eskainiko diren aukerak zein izango diren, bisita birtualaz gain.

Ataza	Esfortzua	Deskripzioa
Aplikazioaren diseinua		
Web orriaren itxura eta diseinua	5 ordu	Bisita birtuala bilduko duen web orriaren itxura eta diseinua erabaki.
Exekuzio-fluxua	5 ordu	Aplikazioaren exekuzio-fluxua zehaztu, diagramak eta azalpenak gehituz.
Erabilpen kasuak	5 ordu	Erabilpen kasuak aztertu eta hauen diagrama egin.
Klase diagrama	5 ordu	Aplikazioan beharko diren klaseak zehaztu eta klase-diagrama eraiki.
Sekuentzia diagramak	5 ordu	Erabilpen kasu ezberdinen sekuentzia diagramak definitu.
Fakultatearen modeloaren diseinua		
3D modelo osoaren banaketa	2 ordu	Fakultatearen modeloa osoa nola eta zenbat zatitan banatuko den erabaki.
Totala:	27 ordu	

Taula 2: Diseinu atazen taula

GARAPENA:

Aplikazioaren garapen osoa bi atal nagusitan banatzen da. Bi atal hauen garapena paraleloan eraman daiteke ia proiektu osoan zehar.

Lehenengo atala web orriaren eta barneko kodeketaren garapenean datza. Aukeratutako liburutegi edo motoreaz baliatuz eta egindako diseinuari jarraituz, ondoko atalak inplementatuko dira:

- Web orria garatu: HTML5 erabiliko da web orriaren egitura sortzeko eta bisita birtuala aurkezteko balio izango duten Canvas etiketak gehitzeko.
- Sortutako 3D modeloak beren testurekin modu egokian kargatzeko kodea garatu beharko da aukeratutako motorearen baliabideak erabiliz. Sortutako modeloak motoreak onartzen duen formatu batera pasatu beharko dira, beraz, oinarri

aldaketak kontuan hartu beharko dira.

- Kargatutako modeloetan mugitzeko erabiliko den kamera inplementatu beharko da. Honetarako kontuan izan beharko da kamera definitzeko erabiltzen diren hasierako parametroak modeloen eskalara doitu behar direla.
- Kamerak modeloarekin izan ditzaken kolisioak detektatu eta kudeatzeko erabiliko den teknikaren algoritmoa inplementatu beharko da bukatzeko. Kolisio detekzioa ahalik eta eraginkorrena izan beharko da, baina ezin du aplikazioaren errendimenduan zama handia sortu.

Bigarren atala fakultatearen 3D modeloaren modelatzearen inguruan mugituko da. Kontzeptu aldetik sinplea da, baino lan gehiena atal honetan biltzen da.

- Modelatzen hasi aurretik, fakultatearen planoak eskuratu beharko dira, modeloa hauetan oinarrituta sortu ahal izateko.
- Behin planoak eskuratuta, diseinuan erabakitako banaketarekin sortuko dira planoen laguntzaz. Egitura orokorraz gain, ateak, leihoak eta beste hainbat gehigarri sartuko dira.

Ataza	Esfortzua	Deskripzioa
Aplikazioaren garapena (Kodeketa)		
Aukeratutako motorea eskuratu	1 ordu	Erabiltzea erabaki den WebGL motorea eskuratu eta instalatu.
Web orria garatu	5 ordu	Bisita birtuala bilduko duen web orria sortu.
3D modeloen eta testuren karga	5 ordu	Sortutako 3D modeloak eta dagozkien testurak kargatzeko kodea garatu.
Kameraren inplementazioa	4 ordu	Bisitariak modeloa ikusteko erabiliko duen kameraren mugimenduak inplementatu.
Kolisio detekzio eta kudeaketa	10 ordu	Kamerak modeloko objektuekin izan ditzaten kolisioak detektatu eta kudeatzeko kodea gehitu.
3D modeloaren garapena		
Fakultatearen planoak eskuratu	1 ordu	Fakultatearen solairu ezberdinen planoak lortu.
3D modeloa sortu plano eta diseinuaren arabera	50 ordu	Fakultatearen 3D modeloa sortu diseinu eta planoei jarraituz.
Totala:	76 ordu	

Taula 3: Garapen atazen taula

KUDEAKETA:

Kudeaketa atazetan proiektuaren hasieraren oinarria garatzen da: Plangintza.

Plangintzaren barruan proiektuaren helburuak, burutu beharreko atazak, arriskuak, erabilitako tresnak eta aurrekontuak bilduko dira. Hauek zehazten diren heinean, dokumentatu egin beharko dira atal guztiak. Kontuan izan beharko da plangintza egokitzeko beharra sortu daitekeela proiektua aurrera joan ahala, edozein gora-beheraren aurrean.

Ataza	Esfortzua	Deskripzioa
Plangintza	10 ordu	Proiektuaren plangintza sortu.

Taula 4: Kudeaketa atazen taula

DOKUMENTAZIOA:

Proiektuaren memoriaren idazketa landuko da ataza hauetan. Proiektuan zehar erabilitako teknologia eta tekniken inguruan egindako ikerketa guztia dokumentatu beharko da. Kontuan izan beharko da memoria edozeinek ulertzeko modukoa izan behar duela; hau da, irakurtzen duenak zeren oinarrizko ideiak ulertzeko gai izan behar da irakurtze hutsarekin, nahiz eta alde teknikoa ez ezagutu.

Kalitatea ere garrantzitsua izango da:

- Aurkibideak.
- Ulerterraza den idazketa.
- Informazioa modu argian antolatuta.

Ataza	Esfortzua	Deskripzioa
Proiektuaren memoria	50 ordu	Proiektuaren memoria idatzi
Proiektuaren aurkezpena	20 ordu	Proiektuaren defentsarako aurkezpena prestatu
Totala:	70 ordu	

Taula 5: Dokumentazio atazen taula

2.3-Arriskuak

Luzera handiko proiektuetan beharrezkoa da honengan eragina izan dezaketen arriskuak detektatu eta hauen konponbide bat planifikatzea. Hasiera batetik ez badira arrisku garrantzitsuenak kontuan hartzen, arazoak sortzen diren momentuan ondorio larriak izan ditzake erantzun egokia ematen ez bazaie arriskuei. Hau dela eta, garrantzitsua da proiektuaren plangintza burutzerako orduan eragin handiena izan dezaketen arriskuak bilatu eta hauen kudeaketarako plan bat sortzea, honetarako arduradun bat aukeratuz. Gure kasuan, proiektuaren garapen taldean kide bakar bat dagoenez, garatzailea arduratuko da arrisku hauen kudeaketaz.

Arriskuak bi eremutan banatu daitezke:

- Proiektu guztiari eragiten dioten arriskuak
- Proiektuko zeregin bat eragiten dioten arriskuak

Arrisku bakoitzak eragin handiagoa edo txikiagoa izan dezake, eragiten duen eremuaren arabera. Adibidez, ataza bati soilik eragiten dioen arrisku batek ataza horretan soilik ekarriko ditu arazoak. Atazaren iraupena luzatuz edo bere garapena denboran mugituz arriskuak sortutako ondorioak konpentsatu daitezke; hau da, ataza hori soilik birplanifikatu beharko da. Baina arriskuak proiektu osoan eragina badauka ondorioak askoz handiago eta larriagoak izan daitezke, proiektuan oraindik garatzeke dauden ataza guztiei eraginez. Honek proiektu osoaren birplangintza batera eraman dezake, denbora asko galduz.

<u>Arriskuen detekzio eta kudeaketa plana:</u>

- **Dedikazio partekatua:** Proiektua karrerako azken lauhilekoan burutuko da, beste hainbat irakasgairekin batera. Ondorioz, ezin izango dira lan ordu guztiak proiektura enfokatu, irakasgai hauetan egin beharreko lanek bere tartea hartuko baitute. Honetaz gain, ikasleak asteburuetan daukan lana dela eta, asteburuetako proiektuko dedikazioa oso baxua izango dela estimatzen da

hasiera batean. Honen eragin nagusiena proiektuaren garapenean atzerapenak sortu daitezkeela da, planifikatutako denbora tarteak aldatuz.

- Plangintza egiterako orduan kontuan hartuko dira proiektutik kanpo dauden egitekoak, atazen banaketa erlaxatuago bat eginez eta proiekturako denbora gehiago dagoen asteetan esfortzu gehiago eskatuz.

– **Partaide gutxiko proiektua:** Proiektuan soilik hiru pertsonek hartuko dute parte: Ikasleak, garatzailearen paperean, eta bi irakasleek, proiektu zuzendari modura. Garatzaile bakarra izanda, proiektuaren lan guztia pertsona baten eskuetan egongo da. Proiektu barnetik edo kanpotik datorren edozein ustekabek izandako eraginak izan dezake momentuan garatzen ari den atazan, denbora galera eta atzerapenak sortuz, ez baitago beste inor ataza horretan laguntza emateko. Bestalde, erabakiak hartzerako orduan ez direnez iritzi desberdin asko izango, errazagoa izango da erabaki okerrak hartzea.

- *Hartutako erabaki bakoitzaren jarraipen bat burutuko proiektuaren garapenean zehar. Erabaki batek zailtasunak sortzen baditu, taldea bildu eta honetarako konponbide edo bide desberdin bat bilatu beharko da. Erabaki okerrek eragin dezaketen atzerapenak konpentsatzeko proiektuari lan ordu gehiago eskaini beharko zaizkio astean zehar, garatzaileari esfortzu estra bat eskatuz.*

– **Partaide baten behin-behineko baja:** Proiektu taldeko partaideren batek taldea denbora batez uzteko beharra sortzen da. Arazo hau oso larria izan daiteke, batez ere kasu honetan, taldean partaide gutxi baitaude. Partaide baten galerak honek garatu beharreko atazen birplangintza bat eragiten du, atazak gainontzeko partaideen artean banatuz edo, kasurik txarrenean eta zoritxarrez gure kasuan, ataza atzeratuz beste garatzailerik ez izateagatik.

- *Kasu hauendako konponbide bakarra atzeratutako atazei astean eskaintzen zaien ordu kopurua handitzea izango da, galdutako denbora guztia konpentsatu arte.*

- **Teknologia berrien erabilera:** Proiektuan erabiliko diren teknologien azterketa eta oinarrizko ikasketak espero baino denbora gehiago har dezake, zaila baita hauen ikasketarako beharko den denbora estimatzea.
    - *Ikasketa prozesua luzatzen bada, ordu gehiago eman beharko zaizkie ikasketa atazei, eta gainontzeko atazen asteko plangintza berregin beharko da, astean ordu gehiago eskainiz galdutako denbora konpentsatzeko.*

- **Garapen ingurunearen matxura:** Proiektuaren garapenerako erabilitako ordenagailu eta tresnak garatzailearen eguneroko bizitzan ere erabiltzen dira. Ondorioz, tresnerian matxura edo arazoak sortzeko probabilitate handiagoa dago, atazen garapenean atzerapen handiak sortuz.
    - *Garapen prozesu osoan sortutako kode, irudi eta modelo guztien segurtasun kopia bat burutuko da aste bakoitzeko ostiralean, lanerako erabiltzen den ordenagailutik kanpo mantenduko den disko gogor batean. Ordenagailua matxuratzen bada, laneko behar diren tresna guztiak izango dituen bigarren ordenagailu bat prestatuko da, larrialdi kasuetarako.*

- **Garapen tresna desegokien hautaketa:** Modu eraginkorrean erabili ahal izateko ikasketa denbora handi bat eskatzen duten garapen tresnek denbora gehiegi kendu dezakete proiektuan, aplikazioaren garapenari ekin beharrean.
    - *Garapenerako balio duten hainbat aplikazio kontuan hartuko dira, eta hauetatik egokiena aukeratuko da. Hasierako aukerak zailtasun handiak sortzen baditu, hasierako multzotik beste aplikazio bat aukeratzeko aukera izango da, erosoago izango den bat bilatuz.*

Arrisku hauen kudeaketa egoki bat mantenduz, hauek proiektuan izan dezaketen eragina minimizatuko da, ahalik eta atzerapen gutxiago sortuz garapenean.

2.4-Garapenerako tresnak

Hurrengo lerroetan proiektuaren garapenean erabiltzeko hautatu diren tresnen zerrenda eta bakoitzak eskaintzen dituen ezaugarriak azaltzen dira. Software librea erabiltzea aukeratu da.

Tresna garrantzitsuenak 3D modeloen sorketa eta aplikazioaren kodeketarako erabiliko direnak izango dira. Modeloak sortzeko Blender erabiliko da, programa honen inguruan garatzaileak daukan ezagutza aprobetxatuz. Bestalde, programaziorako AptanaStudio-ren alde egin da, web aplikazioen programaziorako prestatutako ingurunea dela eta.

Memoria eta bertan txertatuko diren osagai guztiak sortzeko OpenOffice.org paketea eta DIA diagrama sortzailea erabiliko dira, hauen inguruko ezagutzak aprobetxatuz.

Blender

Irudia 2: Blender-en logoa

Blender modelatze, animazio eta 3D grafikoen sorketarako aplikazio informatikoa da, sistema eragile askotan eskuragarri dagoena.

Nahiz eta orain doakoa eta kode irekiko software bezala banatzen den, hasiera batean ez zen aplikazio honen kodea eskaintzen aplikazioarekin. Gaur egun, Windows, MacOSX,Linux,Solaris,FreeBSD eta IRIX sistema eragileen bertsio guztiekiko konpatiblea da Blender.

Bere interfaze grafikoa oso intuitiboa ez izateaz asko kritikatu da, leiho sistema klasikoan ez oinarritzeagatik. Dena dela, interfaze guztia bakoitzaren gustura konfiguratzeko aukera asko eskaintzen ditu, menu guztiak mugitzeko eta bista ezberdinak aldi berean izateko aukerak nabarmenduz.

Hauek dira bere ezaugarri nabarmenenak:

– Plataforma anitza, kode irekikoa, doakoa eta tamaina txikikoa beste 3D

aplikazioekin konparatuta.

- Oinarrizko figura geometriko asko: Sare poligonalak, objektu hutsak, NURBS kurbak, metabolak etab.
- Animazio aukera zabala, alderantzizko zinematikarekin batera.
- 3D objektuak hainbat formatu ezberdinetara esportatzeko aukera (COLLADA gure kasurako)
- Irudi formatu asko onartzen ditu: TGA, JPG, Iris, SGI ...
- ...

Blender fakultatearen hiru dimentsioko modeloak sortzeko erabiliko da proiektu honetan. Bertan sortutako modelo guztiak COLLADA formatuko fitxategietara esportatuko dira aplikazioan erabili ahal izateko, beranduago azalduko den bezala.

Aptana Studio

Irudia 3: Aptana Studio-ren logoa

Aptana Sturio web aplikazioen programaziorako kode irekiko garapen ingurune bat da (IDE). Javascript, HTML, DOM eta CSS lengoaietarako laguntza asko eskaintzen ditu, hala nola, kodea idazteko laguntzak, Javascript debugger bat, kodean dauden erroreak eta warning-ak. Aplikazioaren erabilpenean laguntzeko dokumentazio handi bat eskaintzen du. Gehigarri bezala hainbat plugin gehitzeko aukera eskaintzen du beste programazio lengoaiekin lan egin ahal izateko: Ruby on Rails, PHP etab.

Garapen ingurune honek eskaintzen dituen tresna interesgarrienak:

- Kode laguntzailea: Honi esker kodea idazten den bitartean laguntza eskaintzen da, klaseen metodo eta atributuekin edota idazten ari zaren agindua osatzeko laguntza.
- Nabigatzaileetarako euskarria. Javascript, HMLT eta CSS lengoaien kodean eskaintzen den laguntza estandarretan oinarritzen da.
- FTP/SFTP protokoloak erabiliz proiektua igo, deskargatu edota sinkronizatzeko

aukera.

– Debugger-a. Aplikazioaren exekuzioan bug-ak dauden aztertzeko balio duen tresna, kodean exekuzioa gelditzeko trazak gehituz eta momentuko objektu eta aldagaien egoera aztertzeko aukera eskainiz.

Aptanaren lizentzia aldatu egiten da erabiltzea erabakitzen den bertsioaren arabera. Eclipserako plugin modura erabiltzen bada, Eclipseren lizentziapean egongo da. Aldiz, 'StandAlone' bertsioa aukeratzen bada, aplikazioak bi lizentzia mota eskaintzen ditu: *Free Software/Open Soruce GNU GPL* edo *Aptana Public License (APL)*. Erabiltzaileak bi hauen artean erabakitzeko aukera izango du.

Aptana Studio Web orriaren eta honen atzean egongo den aplikazioaren kodeketa eskatzen duten atazetan erabiliko da, HTML, Javascript eta CSS lengoaien programaziorako hainbat laguntza eta tresna eskaintzen baititu.

OpenOffice.org

Irudia 4: OpenOffice.org-en logoa

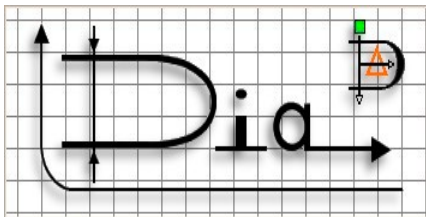
OpenOffice.org software libre eta kode irekiko ofimatika pakete bat da, Microsoft Office-n antzera, hainbat tresna eskaintzen dituena: Testu prozesatzailea, kalkulu orrriak, aurkezpen garatzailea, marrazketa bektorialerako tresnak eta datu-baseak. Plataforma askotarako dago eskuragarri, hala nola GNU/Linux, BSD, Solaris, MacOS, Windows... Nahiz eta Microsoft Office-n lehiakide zuzena den, honekiko ia guztiz konpatiblea da, Office-n bidez sortutako fitxategi gehienak irakurtzeko gai baita, datu-baseak izan ezik. Gainera, ISO/IEC OpenDocument (odf) formatua onartzen du, baita 110 hizkuntza desberdin ere.

OpenOffice.org-ek hasierako oinarri bezala StarOffice pakete ofimatikoa dauka, hasiera batean StarDivision-ek garatua eta beranduago Sun MicroSystem-ek eskuratuko zuena. Pakete honen garapena Oracle Corporation-en zuzendaritzapean dago, beste hainbat enpresen laguntzarekin batera (Novell, RedHat, Google, IBM ...).

Informalki proiektua eta aplikazioa bera OpenOffice bezala ezagutzen diren arren, izen ofiziala OpenOffice.org da, OpenOffice beste enpresa batek erregistratutako marka baita.

OpenOffice.org-en testu prozesatzailea proiektuaren memoria idazteko erabiliko da, eta eskaintzen duen kalkulu orrien tresna lan orduen jarraipena burutzeko erabiliko da. Azkenik, proiektuaren defentsan erabiliko diren gardenkiak Impress-en sortuko dira.

Dia



Irudia 5: Dia-ren logoa

Dia GNOME proiektuaren parte bezala garatutako diagrama sortze eta edizio programa da. Modu modular batean banatuta dago, forma eta figura ezberdinak eskaintzen dituzten paketeetan.
Microsoft Visio-ren lehiakide zuzena da.

Bere ezaugarri nabarmenak ondokoak dira:

- Mota askotako diagramak marrazteko tresna eta baliabideak eskaintzen ditu: UML diagramak, fluxu diagramak etab.
- Gordetako diagramek memoria gutxi okupatzen dute, fitxategi hauek irakurri eta gordetzeko gzip bidez konprimitutako XML formatuko fitxategiak erabiltzen baitira.
- Diagrama irudi bezala gorde daitezke EPS, SVG eta PNG formatuetan.

Dia aplikazioaren diseinuan sortuko diren diagrama ezberdinak eta dokumentaziorako beharko diren grafikoak sortzeko erabiliko da proiektu honetan.

Google Chrome


Irudia 6: Google Chrome-n logoa

Proiekturako Google Chrome aukeratzearen arrazoi handiena WebGL-ren soportea izan da. Ezinbestekoa da teknologia honen sostengua bermatzen duen nabilgatzaile bat erabiltzea, proiektua teknologia honetan oinarritzen baita.
Google Chrome Google-k garatutako kode irekiko doako web nabigatzailea da. Kode irekiko osagaiekin konpilatuta dago, daukan aplikazioen garapenen egitura (Framework) bezala. %1eko merkatu-kuota dauka eta doan deskargatzeko aukera dago zerbitzu-

baldintza zehatzen pean. Jarritako izena, erabilitako erabiltzailearen interfaze grafikoaren markotik dator.

Google Chromeren atzean dagoen software libre proiektuak *Chromium* izena du eta *BSD* eta *Creative Commons* aitorpena lizentzien menpe dago. Proiektu honen helburu nagusia abiadura, egonkortasun eta segurtasun handiagoa daukan nabigatzailea ateratzea da. Izatez, *Chromium* nabigatzailean oinarritzen da Google Chrome eta oro har ezaugarri berdinak ditu, logotipoa eta Googleren euskarri komertziala kenduta.

Google Chrome-n ezaugarri nagusi batzuk:

•**Segurtasuna eta egonkortasuna:** Periodikoki Googlek 2 zerrenda beltzen eguneratze berriak deskargatzen ditu eta erabiltzailea abisatzen du zerrenda beltzeko orrialde batean sartuko dela. Bestalde, prozesu honetan, Googlek orrialde horien jabeei abisatzen die, jakinarazteko bere orriaren segurtasun falta edo ikusgarritasuna.

•**Prozesuen isolamendua:** Garapenerako taldea, hasierako momentutik nabigatzaile hari-anitza egitea pentsatu zuen eta Chromek kontzeptu hau aplikatu du Internet Explorerreko multiprozesadoren arkitektura erabiliz. Eginkizun ezberdinentzat prozesu ezberdinak asignatzen zaizkie. Honekin lortzen dena da, eginkizun bat beste batekin ez interferitzea. Chromeko fitxa bakoitza isolatzen da, software gaiztoak (birusak etab.) saihesteko. Honekin, Software gaizto bat sartuz gero, ezin izango luke ordenagailutik edatu, prozesua isolatua baitago.

•**Abiadura:** Javascript V8 motorra diseinatu egin zen abiadurari garrantzia emanez. Beste ezaugarri berri batzuk ere inplantatu ziren motore honetan: klase ezkutuen trantsizioak, kodigoren dinamika generazioa, etab. Chromek ere badauka helbideen cache DNS bat, web orrien karga azeleratzeko.

•**WebGL:** 11.bertsiotik aurrera Google Chrome-k WebGL teknologia soportatzen du, hardware bidezko grafiko azelerazioa ahalbidetuz.

HTML5

HTML5 (***H**yper**T**ext **M**arkup **L**anguage*, 5 bertsioa) World Wide Web HTML oinarrizko lengoaiaren bostgarren berrikusketa garrantzitsua da. HTML5-ek sintaxiaren bi aldaera desberdin eskaintzen ditu **:** Betiko HTML ("text/html"), HTML5 bezala ezagututako

aldaera, eta XHTML5 sintaxia bezala ezagututako XHTML aldaera, XML bezala zerbitzatzen dena. Hau da XHTML eta HTML paraleloan garatzen diren lehenengo aldia.

Irudia 7: HTML5-en logoa

Proiektu honetan <canvas> etiketa berria erabiltzen da, web orri batean programazio bidez grafikoen generazio dinamikoa ahalbidetzen diguna. Hasiera batean Apple-ek bere Safari nabigatzailerako garatu zuen, baina geroago beste nabigatzaile batzuetara egokitu zen, FireFox 1.5 eta Opera-ren kasuan bezala. Azkenik WHATWG-k estandarizatu du.

WebGL

WebGL web nabigatzaileetan 3D grafikoak erakusteko aukera ematen digun espezifikazio estandar bat da. Honi esker, web orrietan hardware bidez azkartutako 3D grafikoak ikus daitezke, inolako plug-in-ik instalatu beharrik gabe, OpenGL ES 2.0 onartzen duen edozein plataforman.

Irudia 8: WebGL-ren logoa

Laburrago eta modu tekniko batean adieraziz, OpenGL ES 2.0-ren inplementazio natibo bat erabiltzea ahalbidetzen duen Javascript API bat da.

C3DL

"Canvas 3D Library". WebGL-ren erraztea duela helburu adierazten du C3DL-k. Eszena eta 3D objektuen kudeaketaz aparte, matematikarako hainbat baliabide ere eskaintzen ditu. Liburutegi honen ezaugarrien azterketa sakonago bat beranduago egingo da memoria honetan.

Irudia 9: C3DL-ren logoa

2.5-Aurrekontua

Proiektuaren aurrekontuan taldeko kide bakoitzak sartutako lan ordu kopurua eta erabilitako tresnen kostua hartuko da kontuan. Hauekin batera lanerako erabilitako ordenagailuaren kostua kalkulatu beharko da, kontuan izanik ez dela soilik proiektu honetarako erabiliko. Ondorioz proiektuan izango duen kostua soilik bere erabilpenarekin lotutako portzentai bat izango da.

Aukeratutako tresna guztiak software librea eta doakoak direnez, ez dute inolako kosturik suposatuko proiektuan.

Ondoko tauletan proiektuaren kostuak azalduko dira, tresna, garatzaile eta ataza multzo bakoitzerako.

Erabilitako tresnen aurrekontuan amortizaziorako hiru urte (4800 ordu) zehaztuko dira. Honek ordenagailuaren kostuari eragingo dio bakarrik, gainontzeko tresnek ez baitaukate oinarrizko kosturik.

- O.K.: Tresnaren oinarrizko kostua.
- A.D.: Amortizazio denbora.
- A.K.U.: Amortizazio kostu unitarioa.
- E.D.: Erabilpen denbora.
- P.K.: Proiekturako kostua

Tresna	O.K.	A.D.	A.K.U.	E.D.	P.K.
Ordenagailua	799,00 €	4800 ordu	0,1665 €/h	200 ordu	33,3 €
Blender	0,00 €	4800 ordu	0 €/h	-	0,00 €
AptanaStudio	0,00 €	4800 ordu	0 €/h	-	0,00 €
OpenOffice.org	0,00 €	4800 ordu	0 €/h	-	0,00 €
Dia	0,00 €	4800 ordu	0 €/h	-	0,00 €
Google Chrome	0,00 €	4800 ordu	0 €/h	-	0,00 €
C3DL	0,00 €	4800 ordu	0 €/h	-	0,00 €

Taula 6: Proiekturako tresnen aurrekontua

Taldekidea	Lan orduak	Kostua	P.K.
Diseinatzaile eta programatzailea	200	18 €/h	3600 €
Proiektu zuzendaria	10	20 €/h	200 €
		Totala:	**3800 €**

Taula 7: Proiektuko kideen lan orduen aurrekontua

Ataza multzoa	Esfortzu totala	Garatze kostua	P.K.
Analisi eta ikerketa	17 ordu	18 €/h	306 €
Diseinua	27 ordu	18 €/h	486 €
Garapena	76 ordu	18 €/h	1368 €
Kudeaketa	10 ordu	18 €/h	180 €
Dokumentazioa	70 ordu	18 €/h	1260 €
		Totala	**3600 €**

Taula 8: Proiektuko ataza multzoen aurrekontua

Kontzeptua	P.K.
Taldekideen kostua	3800 €
Tresneriaren kostua	33,3 €
Subtotala:	**3833,3 €**
BEZ (%18)	**689,994 €**
Totala:	**4523,294 €**

Taula 9: Proiektuko kostu totala

3-ANALISIA

3.1- OpenGL-tik WebGL-ra

3.2- WebGL motoreak

3.3- 3D modelatzea

3.4- Azken erabakia

3.1- OpenGL-tik WebGL-ra

Aplikazio hau WebGL teknologian oinarritzen dela azaldu da behin baino gehiagotan,eta ,aurretik adierazi den bezala, OpenGL ES 2.0-ren inplementazio natibo bat erabiltzea ahalbidetzen duen Javascript API bat da.

OpenGL ES 2.0 OpenGL-ek eskaintzen dituen funtzionalitateen azpimultzo batek osatzen du. WebGL ulertu ahal izateko, lehenik OpenGL-ren funtzionamendua ulertu beharko da, WebGL-ra iristeko egin diren aldaketak zehaztu ahal izateko.

3.1.1- OpenGL

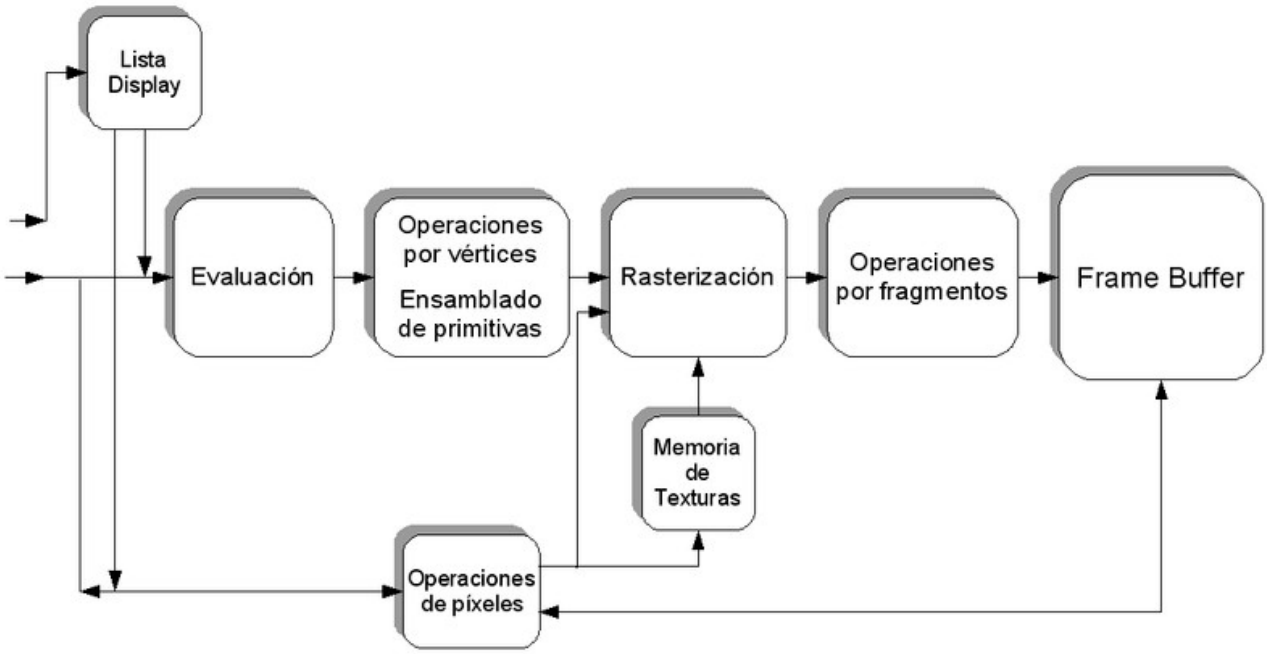
OpenGL liburutegi grafiko ireki bat da. 200 funtzio baino gehiago biltzen ditu, hiru dimentsiotako objektuak marrazteko erabiltzen direnak, erpin, ertz eta poligonoen erabileran oinarrituta. Nabarmendu beharrekoa da gehien marrazten diren poligonoak hirukiak direla, poligono sinpleena izateaz aparte, emaitza egokiak hirukien bidez bermatzen baitira.

Oinarrian, OpenGL funtzio multzo baten deskribapen eta betebeharrak biltzen dituen zehaztapen bat da. Zehaztapen honetatik abiatuta, hardware garatzaileek liburutegi espezifikoak inplementatzeko aukera dute, hardware bidezko azelerazioa txertatzeko aukerarekin.

Honi esker, OpenGL-ren helburu nagusia betetzen da: Txartel grafikoekin lan egiteko interfaze bat lortzen da, txartel grafiko bakoitzaren ezaugarriak kontuan izan beharrik gabe, eta honela programazioa errazten da.

OpenGL-ren oinarrizko funtzionamendua oinarrizkoak diren erpin, ertz eta poligonoak hartu eta pantailan pixel modura marraztean datza. Hau lortzeko OpenGL-k marrazte prozesu osoa hainbat zatitan banatzen du, OpenGL-ren *pipeline* grafikoa

bezala ezagutzen den prozesu segida lortuz.



Irudia 10: OpenGL-ren pipeline grafikoa

Irudi honetan biltzen da marrazte prozesu guztia. Erpin edo operazioak ezkerreko aldetik iristen dira, eta ondorengo prozesua jarraitzen dute:

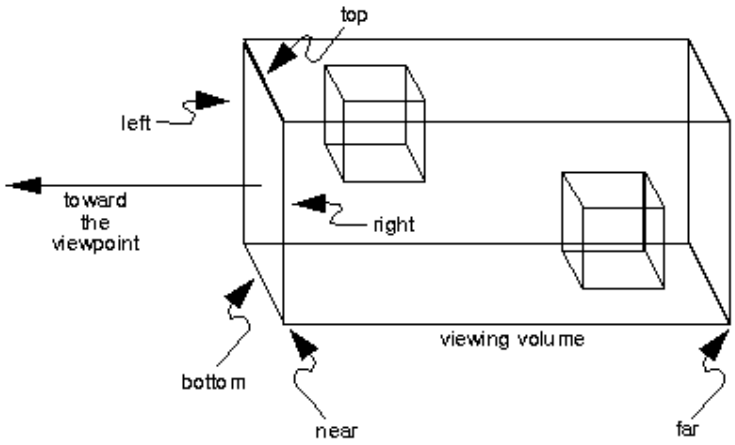
- Ebaluazioa: Funtzio polinomialen ebaluazioa burutzen da (NURBS, Bezier ...) hauen geometriaren hurbilpen bat lortzeko.
- Erpinen gaineko operazioak burutzen dira ondoren. Transformazioak, argiztapena etab. eta ikusten ez diren zatien ebaketa burutzen da. Puntu honetan erpinetan eragiten duten *shader*-ak sartzen dira jokoan, hauek erabiltzeko erabakia hartu bada.
- Rasterizatze prozesuan, erpinak prozesatuta lortzen den bi dimentsiotako eszena *raster* formatuko irudi batera bihurtzen da, pixelen balio egokiak finkatzeko.
- Hurrengo pausoan, pixel bakoitzari kolorea esleitzen zaio, rasterizatze prozesuan interpolatutako erpinen balioen arabera, edo memorian kargatutako testura baten arabera. Puntu honetan *fragment-shader*-ak sartzen dira, hauek erabiltzea erabaki bada.
- Azkenik, orain arteko prozesuari esker lortutako txatalak *FrameBuffer*-ean

kargatzen dira.

OpenGL-rekin programatzean, gure 3D objektuei transformazioak eragin eta pantailan irudikatu ahal izateko bi matrize erabiltzen dira:

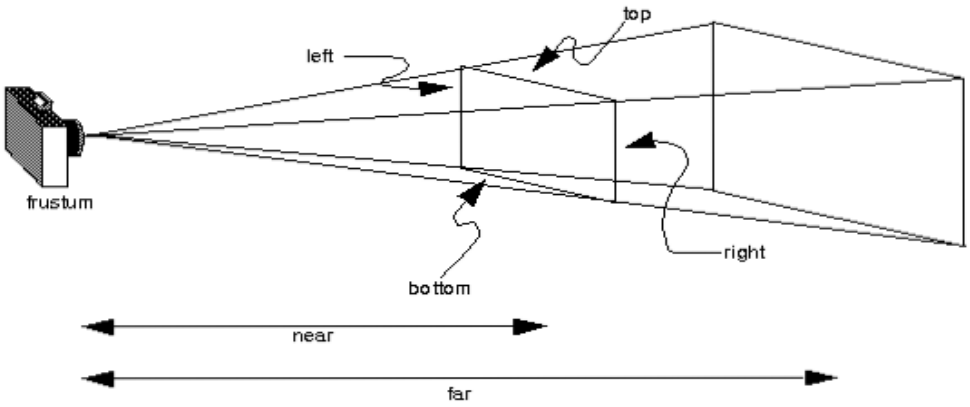
- ***Modelview* matrizea:** Objektu edo erpinei transformazioak eragiteaz arduratzen den matrizea. Honen bidez translazioak, errotazioak eta eskalatzeak burutu daitezke objektuetan.
- ***Projection* matrizea:** Erpinak proiekzio planoan nola proiektatuko den zehazten duen matrizea. Matrize honen bidez gure eszena bista ortogonalean edo perspektiban ikusteko aukera dago.

Proiekzio ortogonal batek proiekzio lerro paraleloak erabiltzen ditu. Bista modu honetan objektuaren tamaina ez dago ikuste puntutik daukan distantziarekin erlazionatuta, beraz tamaina berdineko bi objektu, nahiz eta distantzia ezberdinetara egon, berdin ikusiko dira proiekzioan. Ondorioz, bista honen bidez ez da sakontasun sentsaziorik lortzen, distantziak ez baitu tamainan eragiten.



Irudia 11: Proiekzio ortogonala

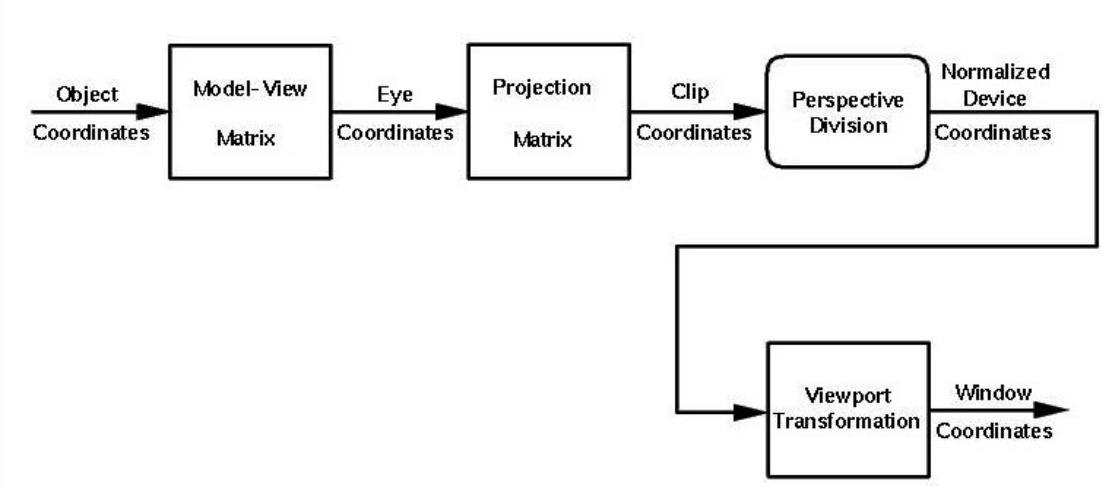
Sakontasun sentsazioa lortu ahal izateko proiekzio perspektiboa erabili behar da. Proiekzio mota hau erabiltzeko bi plano behar dira: Hurbileko eta urruneko planoak. Ikuste puntuaren eta bi plano hauen bitartez, bista perspektiboa definitzen duen prisma bat sortzen da.



Irudia 12: Perspektiba proiekzioa

Bi plano hauen artean mantentzen diren objektuak soilik marraztuko dira, eta hauen tamaina ikuste puntutik dauden distantziaren arabera aldatuko da. Honi esker, sakontasun sentsazioa lortzen da, errealitatean gauzak ikusten diren modura.

Objektuen marrazketarako bi matrize hauen erabilera ondoko grafikoan azaltzen da, objektu baten erpinak sarrera modura hartuta.



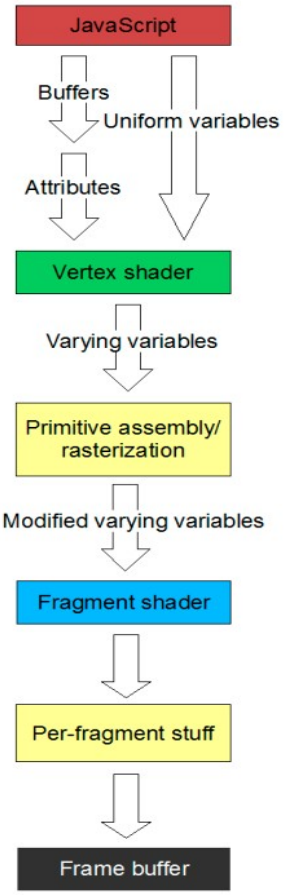
Irudia 13: Erpinen transformazio prozesua

3.1.2- WebGL

WebGL-k OpenGL-ko kontzeptu nagusiak mantentzen ditu, baina web ingurune batean. WebGL edozein ingurunetan exekutatu ahal izateko, Javascript bidez lotura bat sortzen da, nabigatzailearen bidez hardware-ari zer egin behar duen adierazi ahal izateko.

OpenGL 2.0 ES-n oinarritzen denez, bertsio honek oinarrizko OpenGL-rekiko dauzkan desberdintasunak ere mantendu egiten dira WebGL-n.

Aldaketa nabarmenetako bat *shader*-en nahitaezko erabilpena da. WebGL-n erpin eta txatal shader-ak erabiltzera behartzen da (*vertex-shader* eta *fragment-shader*). *Vertex-shader*-aren bidez erpinak prozesatu eta hauen kokapena kalkulatzen da. Bestalde, *Fragment-shader*-aren bidez txatalak prozesatzen dira, ondoren bakoitzari dagokion kolorearen informazioa lortu ahal izateko.



Irudia 14: WebGL-ren renderizatze pipeline-a

Shader-ek prozesua amaitu eta gero, lortutako emaitzak FrameBuffer batera bidaltzen dira, txartel grafikoarekin konektatu eta irudiak web orriko canvas etiketan marraztu ahal izateko.

Erpinak definitu eta marraztera bidaltzeko modua ere aldatu egin da WebGL-n. OpenGL-n erpinak glBegin() eta glEnd() aginduen artean definitzen ziren, erpin soilak, ertzak edo poligonoak marrazteko helburuarekin.

```
glBegin(GL_POLYGON);
   glVertex(0.0,0.0,0.0);
   glVertex(0.5,0.0,0.5);
   glVertex(1.0,0.0,0.0);
glEnd();
```

Kodea 1: Triangelu baten marrazketa OpenGL bidez

WebGL-n aldiz, *buffer* bat definitu eta sistemako *buffer* batekin lotzen da. Ondoren gure *buffer*-ari erpinen koordenatuak dauzkan array bat gehitzen diogu, eta zenbat balio eta ze tamainakoak diren esleitzen dira *buffer*-eko atributuetan.

```
squareVertexPositionBuffer=gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER,squareVertexPositionBuffer);

vertex=[
 1.0,1.0,0.0,
 -1.0,1.0,0.0,
 1.0,-1.0,0.0,
 -1.0,-1.0,0.0
];

gl.bufferData(gl.ARRAY_BUFFER,new Float32Array(vertex),gl.STATIC_DRAW);
squareVertexPositionBuffer.itemSize=3;
squareVertexPositionBuffer.numItems=4;
```

Kodea 2: WebGL-n karratu baten erpinak definitzeko kodea

Erpin hauek marraztera bidaltzean aurretik ikusitako prozesuaren bidez *shader*-ak aplikatuko zaizkio HTML orriko canvas-ean marraztu ahal izateko.

Gure aplikazioa garatzerako orduan, dena hasieratik programatu beharko zen edo jada lagungarri izango litzatekeen zerbait eskuragarri egongo zen eztabaidatu genuen. Nahiz eta beti merezi duen teknologia berri bat ezagutzeko honekin zuzenean lan egitea, kasu honetan WebGL-ren erabilera errazten duen motore edo liburutegi bat erabiltzea erabaki zen.

Motore edo liburutegi baten laguntzarekin kode gutxiago beharko da aplikazioaren oinarrizko eskaerak programatzeko, eta modu honetan denbora gehiago erabili daiteke fakultateko modeloak sortu eta hobetzeko, errealitatearekiko fidelagoak diren modeloak lortuz.

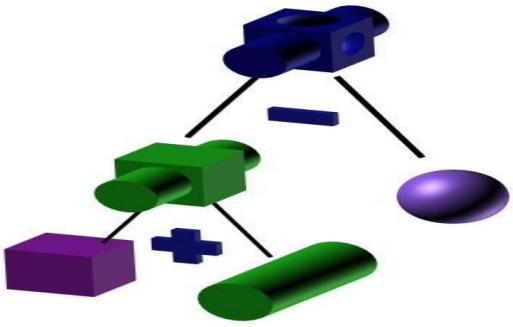
Aplikaziorako egokiena izango zen liburutegia aurkitu ahal izateko, honen inguruko azterketa bat burutu zen, liburutegi edo motore ezberdinek eskaintzen zituzten baliabide eta ezaugarriak ezagutzeko.

3.3- 3D modelatzea

OpenGL edo WebGL bidez bistaratu nahi diren hiru dimentsiotako objektuak sortzeko prozesuari modelatzea deritzaio. Nahiz eta OpenGL-n zuzenean erpinak, zuzenak eta aurpegiak marrazteko aukera eskaintzen den, eta hauen bidez edozein 3D objektu sortu daitekeen, benetan zaila eta gogorra da objektu konplexuak modu honetan sortzea. Kubo edo piramide bat sortzea erraza izan daiteke, baina Star Wars filmeko X-Wing baten modelo bat?

Puntu honetan sartzen da jokoan modelatzea. Software egokiaz baliatuz, posible da 3D objektu konplexuak modu bisual eta praktiko batean sortzea. Gainera, modelatzeko teknika, tresna edo bide bat baino gehiago daude. Modelatze teknika hauek hiru multzotan banatu daitezke:

- **Modelatze solidoa:** Solidoen geometria eraikitzailea bezala ere ezagutzen dira teknika hauek (*CSG Constructed Solid Geometry*). Modelo solidoek adierazten dituzten objektuen bolumena finkatzen dute, eta kasu askotan hauen masaren zentroa, barne materialen dentsitatea etab. Modelatze edo adierazpide mota honen ohiko adibide bat ondoko irudian azaltzen dena da: Objektu bat objektu sinpleagoen arteko eragiketa logikoen bidez sortzen da. Ordenagailu bidezko fabrikazioan asko erabiltzen dira.

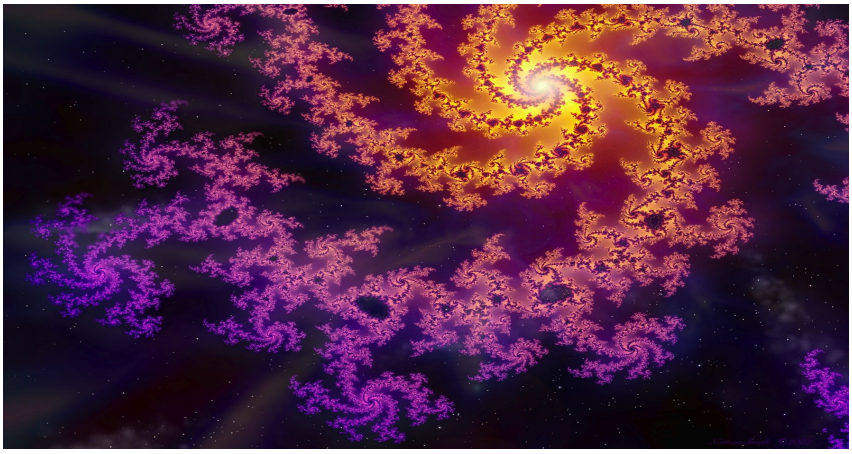


Irudia 15: CSG objektua eragiketa logikoen bidez

- **Azalera modelatzea:** Azalera adierazpidea bezala ere ezagutua (B-Rep, Boundary representation). Modelo hauek soilik objektuaren azala adierazten dute, barrukoa kontuan hartu gabe, hau baita garrantzi gehien daukana. Sortu eta eraldatzeko erosoagoak dira teknika hauen bidez sortutako modeloak. Gaur egungo 3D modelatze tresna gehienek teknika hauek sortzen dituzten

modeloak erabiltzen dituzte, eta hauek sortzeko tresnak eskaintzen dituzte. Multzo honen barruko teknika garrantzitsuenak modelatze poligonala eta kurba bidezko modelatzea dira.

- **Modelatze generatiboa:** Multzo honetako teknikak segida matematikoen jarraipenean oinarritzen dira modeloak sortzeko. Modu honetan denboran eraldatzen diren modeloak sortzen dira. Hauen adibide ezagunena fraktalak dira.



Irudia 16: Segida matematikoen bidez sortutako fraktala

Proiektu honetan azalera modelatze teknikak erabiltzea erabaki da, eta multzo honen barruan modelatze poligonala. Blender aukeratu da modelatze prozesurako, modelatze teknika honetarako prestatuta baitago batez ere, hainbat tresna erabilgarri eskainiz.

Modelatze poligonalari esker behar diren modeloak modu erraz eta intuitibo batean sortu ahalko dira Blender-en laguntzaz.

3.3.1- Modelatze poligonala

Modelatze poligonala erpinen erabilpenean oinarritzen da. Erpin ezberdinen arteko loturen bidez poligonoak sortzen dira, eta poligono ezberdinen bidez hiru dimentsioko objektuak sortzen dira. Objektua sortzeko erabiltzen diren hiru dimentsioko

poligonoek objektuaren sare poligonala eratzen dute.

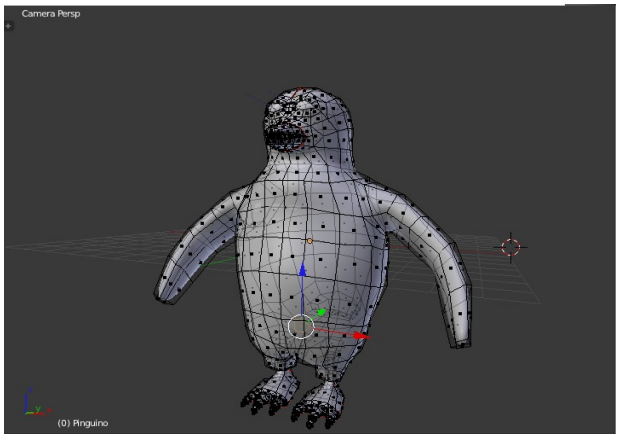
3D modelo baten osagai zehatzak:

- Erpina (Vertex): Hiru dimentsioko espazio batean kokatuta dagoen puntu bat. X,Y eta Z koordenatuen bidez definitzen da erpin baten posizioa. Hauen arteko loturak eginez, ertzak eta poligonoak sortzeko erabiltzen dira.
- Ertza (Edge): Bi erpinen loturaren bidez sortzen dira ertzak. Hiru ertz edo gehiagoren loturaren bidez poligonoak sortu daitezke. Gainera, bi poligonoren arteko muga den ertz batek bektore normal bat dauka bi poligono hauen arteko lotura leuna edo gogorra den adierazteko.
- Aurpegia (Face): Hiru ertz edo gehiago lotzean sortzen da aurpegi bat. Hauek definitzen dute modelo poligonalaren azalera. Aurpegi bakoitzak bektore normal bat dauka, honekiko perpendikularra dena.

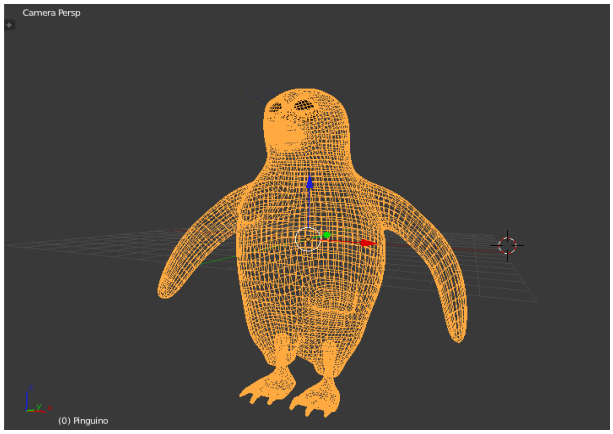
Adierazpide mota honek bi informazio mota eskaini dezake:

- Topologikoa: Erpin, ertz eta aurpegien arteko erlazioa, aurpegien norantza...
- Geometrikoa: Ertzen eta aurpegien informazio geometrikoa.

Objektu kurbatuak adierazi ahal izateko, hauen hurbilpen bat egin beharra dago, poligono txiki askoren erabilpenaren bidez. Zenbat eta poligono gehiago erabili, kurba leunagoa izango da, eta ez dira poligonoak antzemango. Hala ere, kontu handiz ibili beharra da poligonoen kopuruarekin. Geroz eta poligono gehiago, konplexuago izango da objektua, eta hau prozesatu eta marrazteko ere denbora gehiago beharko da. Gaur egun, bideojokuetan ikusten diren modeloak garatzerako orduan, adibidez, hasiera batean poligono asko dauzkan modeloa sortzen da, beranduago retopologizatu egiten dena poligono gutxiagoko edo "low-poly" modelo batekin. Hasieran sortutako modelo konplexuaren zehaztasunak testuretan gordetzen dira, normalen mapa baten bidez, ondoren "low-poly" modeloan erabiltzeko.



Irudia 17: Pinguinoaren "low-poly" modeloa



Irudia 18: Pinguinoaren "high-poly" modeloa

Beste adierazpideetako objektuak honetara itzultzen dira askotan, editatzeko erosoagoa baita askotan, eta irudia sortzeko sinpleagoa delako, emaitza onargarriak eskainiz. Gainera, poligono bakoitza bere aldetik tratatu daiteke, honek aurpegien ezkutaketarako Z-buffer teknikan laguntza asko eskainiz.

Blender-en modeloak sortzerako orduan, gure modeloak sortzeko hasiera puntu bezala kubo bat erabiltzea erabaki da. Kubo honen erpin eta aurpegiak bikoiztuz nahi dugun forma eman diezaiokegu, sortu nahi den objektua lortu arte, eta blender-ek eskaintzen dituen hainbat tresnen bitartez zehaztasun gehiago gehitu daitezke objektura, poligonoak zatitzen adibidez. Dena dela, kontuan izango da 3D objektuen poligono maila txikia izan beharra dela, beranduago aplikazioaren errendimenduan eragin handirik ez izateko.

Behin modeloak sortu direla, Blender-en laguntzarekin COLLADA formatuko fitxategietara esportatuko dira modeloak, gure aplikazioan erabili ahal izateko.

3.3.2-COLLADA

COLLADA (COLLAborative Design Activity) 3D eszena eta objektuen elkar-trukaketarako sortu zen fitxategi formatua da. XML eskema estandar batean oinarritzen da, eta 3D grafikoak sortzeko erabiltzen diren programa batzuen artean modelo

digitalen trukea ahalbidetzen du. Honela, software ezberdinen artean sortutako modeloak garatzeko ez dago arazorik, bi programa hauek COLLADA formatua onartzen duten heinean.

Hasiera batean COLLADA garapen prozesuan trukerako erabiltzeko formatu bezala planteatu zen, garatzaileek aplikazioaren azken bertsioan beste formatu egokiago bat erabili beharko zutela adieraziz, vertex eta fragment shader-ak arazorik gabe erabili ahal izateko. Dena dela, hainbat motore grafikok formatu hau arazorik gabe onartzen dute garatutako aplikazioan erabili ahal izateko.

Gure aplikazioak formatu honetako fitxategiak erabiliko ditu modeloak kargatu ahal izateko, aukeratutako motoreak hauen karga eta kudeaketarako hainbat erraztasun eskaintzen baititu. Bestalde, Blender-ek jada COLLADA esportatzaile bat daukanez, ez da inolako arazorik sortuko, zuzenean motoreak onartzen duen formatua erabili ahalko baita.

3.2- WebGL motoreak

WebGL ofizialki martxan jarri zenetik denbora "gutxi" pasa den arren, hainbat motore ezberdin garatu eta argitaratu dira. Motore hauen helburua WebGL bidezko programazioa hedatu eta erraztea da.

SceneJS:

Xeolabs-en eskutik datorren motorea. Gehienbat JSON-en oinarritutako API bat eskaintzen digu, hainbat eszena eta bertako objektuak aldi berean kudeatzeko aukera eskainiz. Definizio handiko renderizazioengatik nabarmentzen da batez ere.

Nahiz eta COLLADA fitxategiak erabiltzeko aukera motorean barneratuta ez dagoen, posible da fitxategi mota hau python script baten bidez JSON formatura pasatzea ondoren erabili ahal izateko.

Sagu eta teklatuaren gertaeren kudeaketarako Canvas-ek eskaintzen dituen baliabideak erabili behar dira motore honekin, ez baitu honetarako baliabiderik eskaintzen. Bestalde, ez du kamerak sortu eta kudeatzeko erraztasunik aurkezten, beraz, javascript bidez sortu eta kudeatzeko beharra sortzen da.

Sostengu aldetik, Wiki bat eskaintzen du GitHub bidez. Gainera, hainbat adibide eta bideo interesgarri aurki daitezke bere web orrian.

GLGE:

Bere web orrian azaltzen duen bezala: GLGE WebGL-ren erabilera errazteko javascript liburutegi bat da. Motore honen helburua WebGL-ren natura ezkutatzea da, bere erabilera erraztuz, web garatzaileak web orriko edukietan denbora gehiago pasatu eta eduki hobeak eskaini ahal izateko.

Blender-en sortutako fitxategiak zuzenean esportatzeko script bat eskaintzen du liburutegiaren garatzaileak, XML dokumentu bat sortzen duena. Modelo eta testuretaz

aparte, animazioak ere esportatzen ditu fitxategitik. Bestela, COLLADA formatuko objektu fitxategiak kargatzeko aukera ere garatuta dago. Eszenak hasieratik sortu edo eta fitxategi batetik kargatzeko aukera eskaintzen du, baita bertan kamerak, argiak etab. sortu eta kudeatzeko erraztasunak. Xagu eta teklatuaren gertaerak kontrolatzeko baliabideak eskaintzen ditu ere.

Nahiz eta GLGE-k ez duen fisika kudeatzeko ezaugarririk eskaintzen, JiglibJS liburutegia erabiltzen da motorearekin batera eskaintzen diren adibideetan, beraz posible da fisikak ere modu erraz batean inplementatzea.

Aipatu beharrekoa da sostengu egoki bat mantentzen duela, liburutegiaren dokumentazio eta foro aktibo baten bidez. Gainera liburutegiaren sortzailea edozein zalantza edo arazo argitu eta konpontzeko prest dago momentu oro.

Azkenik, motorea Github-en bidez kudeatzen da. Beraz, posible da (eta sortzaileak berak bultzatzen du) kodea hobetu edo hedatzeko ekarpenak egitea.

CubicVR:

COLLADA fitxategiak kargatu eta eszenak kudeatzeko aukera eskaintzen du, eszenari kamerak, argiak etab. gehitu eta kudeatzeko erraztasunak emanez, baina ez du fisika kudeatzeko zuzeneko baliabiderik eskaintzen. Honetarako Ammo.js liburutegiaren erabilera aipatzen da.

Sostengu aldetik,nahiz eta API-aren dokumentazioa eskuragarri dagoen, web orriko foroa nahiko hutsik eta "geldituta" ikusten da. GitHub bidez kudeatzen denez garapena, bertako wiki-an ere laguntza aurki daiteke. Dena den, web orrian demo interesgarri batzuk aurkezten dira, baita tutorial batzuk ere, motorearen oinarrizko funtzionamendua ezagutzeko.

Coppericht:

Ambiera enpresaren lizentzia komertzialeko motorea. Gertaerak, kamera eta

fisika kudeatzeko erraztasun asko eskaintzen ditu. Hainbat objektu fitxategi mota onartzen ditu, baina hauek kargatzeko CopperCube editoreaz baliatu beharra da lehenik Copperlicht-ek ulertzen duen fitxategi mota bat sortu ahal izateko.

Motore eta editorearen arteko dependentzia nahiko handia da, eta konplexua izango litzateke motorea soilik erabilita objektu fitxategi ezberdinak kargatzea. Honek arazo handi bat sortzen du gure kasuan, Blender bidez sortutako modeloak kargatzeko aukera bilatzen baita. Gainera, editorea erabili ahal izateko, 15 proba egun pasa eta gero, ordaindu egin behar da.

Sostenguari dagokionez, web orrian foro aktibo bat mantentzen da, eta dokumentazioa eskuragarri dago. Gainera hainbat tutorial eskaintzen dira motorearen hainbat baliabide nola erabili azalduz.

C3DL:

"Canvas 3D Library".GLGE-k bezala, WebGL-ren erraztea duela helburu adierazten du C3DL-k. Eszena eta 3D objektuen kudeaketaz aparte, matematikarako hainbat baliabide ere eskaintzen ditu.

COLLADA fitxategiak kargatzeko aukera barneratuta dauka, eta posible da ere partikula sistemak definitu eta bistaratzea. Eszenetan objektuak, kamerak, argiak etab. gehitu eta kudeatzeko erraztasun handiak aurkezten ditu. Kameraren mugimenduaren kontrola modu errazean egin daitezke, batez ere eskaintzen duen matematika baliabideak erabilita.

Ez dauka fisika kudeatzeko aukerarik inplementatuta, baina eskuragarri dauden hainbat liburutegi erabili daitezke, aurreko motoreekin egin daitekeen bezala.

Sostengu aldetik, ikasketarako hainbat tutorial eskaintzen ditu eszenen sorketa, kameraren kudeaketa, objektuen karga, materialen asignazioa etab. azaltzen. Bestalde,

liburutegiaren dokumentazio oso bat ere eskuragarri dauka web orrian eta honen kodea GitHub-en dago. Gainera wiki bat ere mantentzen dute orrialde honetan. Azkenik, hainbat demo interesgarri eskaintzen dira bere web orrian, benetan lagungarriak direnak.

Creative commons lizentziapean erregistratuta dago C3DL.

HAUEK BAKARRIK?:

Motore hauetaz aparte beste hainbat ere aurki daitezke WebGL-ren inguruan bilaketak burutzen baditugu:

- SpiderJS
- Kuda
- GammaJS
- EnergizeGL
- Three.js
- ...

Nahiz eta proiekturako ez den azterketa sakon bat egin azken liburutegi hauen gainean, aukera interesgarriak aurkezten dituztela adierazi beharra dago, eta goian azaldutako motoreek eskaintzen dituzten hainbat baliabide ere eskaintzen dituzte.

3.4-Azken erabakia

Motore ezberdinak aztertu eta gero, aplikazioaren garapenerako *C3DL* erabiltzea erabaki da. Liburutegi honek aplikazioan erabiliko diren modeloak COLLADA-ren laguntzarekin kargatzeko eskaintzen duen erraztasuna izan da arrazoi handienetako bat,baina ondoko ezaugarriek ere bultzatu dute *C3DL*-ren aukeraketa:

- Frustum culling teknika inplementatuta dauka: Honi esker, eszenan objektu asko kargatzen badira ere, kameraren bistaren barruan ez dauden objektuak ez dira marraztuko, eta ondorioz aplikazioaren errendimendu egoki bat mantendu ahalko da.
- Hiru dimentsiotako kolisio detekzioa inplementatuta dauka: Aplikazioan erabiliko den kamerak eszenako objektuekin izan ditzakeen talkak detektatu eta kudeatzeko balioko du. Honek lan asko aurreztuko du garapenean.
- Eszenako objektuekiko elkarrekintza posible da: Canvas-ean klik egitean, ea objektu baten gainean klik egin den edo ez kudeatzen du. Aplikazioan fakultateko solairuetako ateetan klik egitean gela ezberdinak kargatzeko erabiliko da ezaugarri hau batez ere.
- Liburutegiaren kodea gure beharretara egokitzeko aukera eta baimena ematen du: Baliteke garapenean zehar interesatzen ez zaizkigun baliabide batzuk aplikazioaren errendimenduan eragina izatea, edo inplementatuta dagoen zerbait hobetzea lortzea. Kasu hauetan kodea aldatu ahal izango da.

4-APLIKAZIOAREN DISEINUA

4.1-Modeloak garatzen

4.2-Modeloak esportatzen

4.3-Web aplikazioa

4.1- Modeloak garatzen

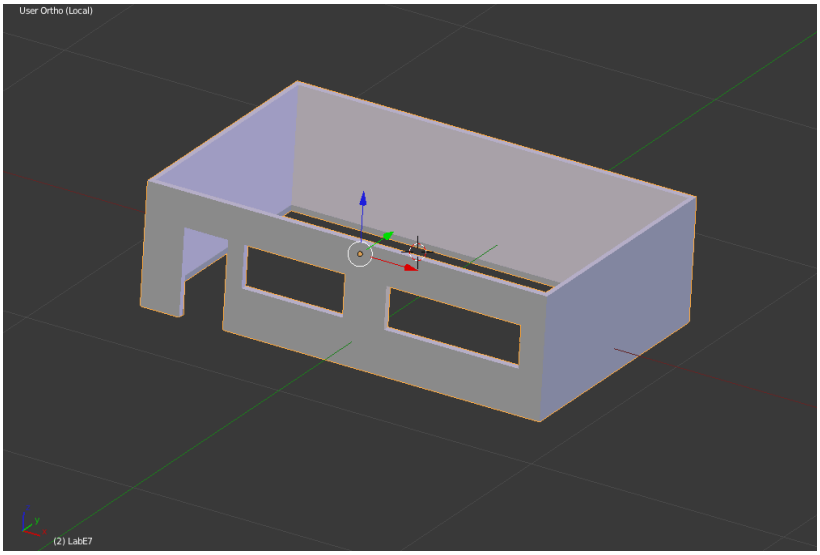
Aurreko kapituluan azaldu den bezala, modeloak sortzeko modelatze poligonalean oinarrituko gara, Blender-ek eskaintzen dituen tresnetaz baliatuz.

Modeloak garatzen hasi aurretik, fakultatearen modeloa nola zatituko behar den erabaki behar da. Kasu honetan hasieratik burura etortzen den ideia erabili da: solairutan banatzea.

Fakultateko solairu bakoitza bere aldetik modelatuko da, bertan osagaiak gehituz (leihoak, mahaiak, ateak, eskailerak ...). Modu honetan, aplikazioan fakultatea marrazterako orduan, soilik uneko solairua marraztuko da. Hala ere, kontuan izan beharra da beste solairuetarako sarrera bezala balio duten zatietan beste solairuetan ikusten dena modelatu beharko dela, nahiz eta errepikapena izan.

Solairuetaz gain, bulego, laborategi eta beste gela berezi batzuetarako aparteko modeloak sortuko dira, edozein momentutan kargatu ahal izateko, solairu bat kargatu behar izan gabe. Gainera, solairu eta gela hauetan gehigarri bezala erabiltzen diren objektuak aparteko objektu entitate bezala gordeko dira, beste edozein modelotan erabiltzeko erraztasuna eskaintzeko.

Modelo gehienak sortzeko, Kubo batetik hasita, *Extrude*-ren bidez, erpin zikloak gehituz eta sobran dauden erpinak ezabatuz, gela baten oinarrizko egitura sortzen da. Egitura honetan ate eta leihoendako hutsuneak erpinak ezabatuz lortzen dira.



Irudia 19: Gela bat oinarrizko egituratik habiatuta

Oinarrizko egitura hartuta, gela ezberdinak sortu behar dira hutsune ezberdinak eginez. Baina "kopia" hauek egin baino lehen, hasierako modeloari testurak ezarriko zaizkio, kopietan testurak jartzeko lana ekiditeko.

Gelak kopiatuz eta eszenan birkokatuz, solairu ezberdinak sortu ahalko dira. Honela hasieran sortutako gela oinarria aprobetxatzen da, eta modeloen garapenean denbora aurrezten da.

Solairuak ondoko ordenan sortuko dira:

- **Solairuartea:** Solairu guztien artean sinpleena denez, hasieran aplikazioan probak egin ahal izateko hau sortzea erabaki da, denbora gutxien hartuko duena izango baita.
- **3.solairua:** Solairu handien artean sinpleena da. Behin bukatuta, 1 eta 2 solairuen oinarri bezala erabiliko da. Hainbat bulegotarako ateak etab. gehituko dira.
- **2.solairua:** 3.solairuko modeloa oinarritzat hartuta, honen pareten posizio eta tamainak aldatuz, eta osagaiak kendu edo gehituz, 2.solairua eratuko da. Bulegoetarako ateak gehitu dira hemen ere.
- **1.solairua:** 3.solairuko modeloa oinarritzat hartuta sortuko da solairu hau. 2 eta 3 solairuetatik desberdina da, gela berezi ezberdin eta

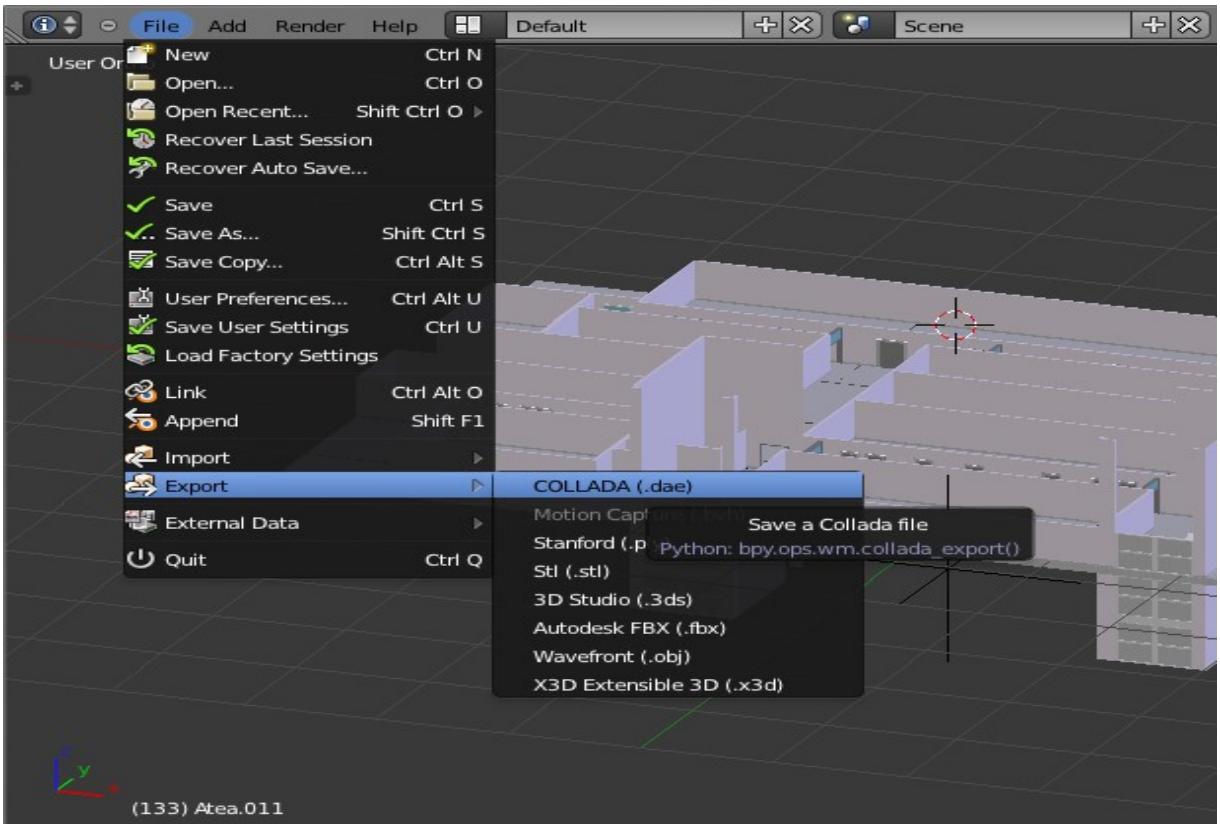
solairuarterako eskailerak baititu.

- **Behe solairua:** Solairu hau seguruenik ezin izango da sortu edo guztiz garatu. Dena dela, modeloarekin hastea lortzen bada, fakultateko sarreran jarriko da interes handiena, bertan estetika egoki bat eskaintzea gararntzitsua baita.

Solairuetan osagai bezala erabiliko diren objektuak (mahaiak, leihoak ...) garatzaileak modelatuko dituela erabaki da. Ikusiko diren 3D modelo guztiak edo gehienak proiektuan sortutakoak izango dira.

4.2- Modeloak esportatzen

Modelo bat bukatuta dagoenean, edo momenturaino egindakoa web aplikazioan nola ikusiko den probatu nahi bada, COLLADA formatuko fitxategi batera esportatuko 3D modeloa. Honetarako, Blender-ek esportatzaile bat dauka jada. Nahikoa izango da File -> Export -> COLLADA erabiltzea, eta fitxategiarendako izen bat aukeratzea.



Irudia 20: Modeloa COLLADA fitxategira esportatzen

Blender-ek zuzenean sortuko du COLLADA fitxategia. Fitxategi berri honetan, sortutako eszena guztia biltzen da, definitutako argi eta kamerak barne, nahiz eta C3DL-k COLLADA irakurtzean argiak eta kamerak ez dituen kargatuko. Objektuekin batera, hauei esleitutako testuren esleipena eta testuren kokalekua ere biltzen da.

COLLADA-ri begiratu bat ematen bazaio, XML hutsa den idazkera aurkitzen da. Honela definitzen da kubo baten geometria:

```
<geometry id="Cube-mesh" name="Cube">
      <mesh>
        <source id="Cube-mesh-positions">
          <float_array id="Cube-mesh-positions-array" count="24">1 1 -1 1 -1 -1 -1
-0.9999998 -1 -0.9999997 1 -1 1 0.9999995 1 0.9999994 -1.000001 1 -1 -0.9999997 1 -1 1
1</float_array>
          <technique_common>
            <accessor source="#Cube-mesh-positions-array" count="8" stride="3">
              <param name="X" type="float"/>
              <param name="Y" type="float"/>
              <param name="Z" type="float"/>
            </accessor>
          </technique_common>
        </source>
        <source id="Cube-mesh-normals">
          <float_array id="Cube-mesh-normals-array" count="18">0 0 -1 0 0 1 1 -2.83122e-
7 0 -2.83122e-7 -1 0 -1 2.23517e-7 -1.3411e-7 2.38419e-7 1 2.08616e-7</float_array>
          <technique_common>
            <accessor source="#Cube-mesh-normals-array" count="6" stride="3">
              <param name="X" type="float"/>
              <param name="Y" type="float"/>
              <param name="Z" type="float"/>
            </accessor>
          </technique_common>
        </source>
        <vertices id="Cube-mesh-vertices">
          <input semantic="POSITION" source="#Cube-mesh-positions"/>
        </vertices>
        <polylist material="Material" count="6">
          <input semantic="VERTEX" source="#Cube-mesh-vertices" offset="0"/>
          <input semantic="NORMAL" source="#Cube-mesh-normals" offset="1"/>
          <vcount>4 4 4 4 4 4 </vcount>
          <p>0 0 1 0 2 0 3 0 4 1 7 1 6 1 5 1 0 2 4 2 5 2 1 2 1 3 5 3 6 3 2 3 2 4 6 4 7 4
3 4 4 5 0 5 3 5 7 5</p>
        </polylist>
      </mesh>
      <extra><technique
profile="MAYA"><double_sided>1</double_sided></technique></extra>
    </geometry>
```

Kodea 3: Geometria informazioa COLLADA formatuan

Kodea aztertuz, kuboaren erpinak eta normalak zehazten direla ikusi daiteke. Bestalde, Blender-en egin diren transformazioak beste atal batean definitzen dira, kubo

objektua definitzen duen nodo batean, non geometria hau ezaugarri gisa txertatzen den.

```
<node id="Cube" type="NODE">
        <translate sid="location">0 0 0</translate>
        <rotate sid="rotationZ">0 0 1 0</rotate>
        <rotate sid="rotationY">0 1 0 0</rotate>
        <rotate sid="rotationX">1 0 0 0</rotate>
        <scale sid="scale">1 1 1</scale>
        <instance_geometry url="#Cube-mesh">
          <bind_material>
            <technique_common>
              <instance_material symbol="Material" target="#Material-material"/>
            </technique_common>
          </bind_material>
        </instance_geometry>
</node>
```

Kodea 4: Kubo objektuaren definizioa, eszenako nodo bezala

Aztertzen jarraitzen bada, kamerak eta argiak antzeko moduan definitzen direla ikusten da. Alde batetik kamera edo argiaren ezaugarri guztiak definitzen dira, eta ondoren nodo batean objektu bezala definitzen da, bertan transformazioak adieraziz.

```
 <camera id="Camera-camera" name="Camera">
      <optics>
        <technique_common>
          <perspective>
            <xfov sid="xfov">49.13434</xfov>
            <aspect_ratio>1.777778</aspect_ratio>
            <znear sid="znear">0.1</znear>
            <zfar sid="zfar">100</zfar>
          </perspective>
        </technique_common>
      </optics>
    </camera>

...

<node id="Camera" type="NODE">
        <translate sid="location">7.481132 -6.50764 5.343665</translate>
        <rotate sid="rotationZ">0 0 1 46.69195</rotate>
        <rotate sid="rotationY">0 1 0 0.619768</rotate>
        <rotate sid="rotationX">1 0 0 63.5593</rotate>
        <scale sid="scale">1 1 1</scale>
        <instance_camera url="#Camera-camera"/>
</node>
```

Kodea 5: Kamera baten definizioa COLLADA-n

Argiekin berdin jokatzen da, baina argien kasuan askoz ezaugarri gehiago definitu behar dira: Intentsitatea, itzalak, ahuldura, kolorea ...

Azkenik, eszena definitu ahal izateko, definitutako nodo guzti hauek eszena baten barruan kokatzen dira.

```
<library_visual_scenes>
 <visual_scene id="Scene" name="Scene">
 <node> ... </node>
...
    </visual_scene>
  </library_visual_scenes>
  <scene>
    <instance_visual_scene url="#Scene"/>
  </scene>
```

Kodea 6: Eszena baten definizioa COLLADA-n

4.3-Web aplikazioa

Atal honetan aplikazioa diseinatu eta garatzerako orduan erabili den motorraren erabilpenaren eta kolisio detekziorako egindako azterketa eta inplementazioaren inguruko azalpenak emango dira.

4.3.1- Motorraren erabilpena

C3DL erabilita asko errazten da programazio lana, hainbat klase eskaintzen baititu 3D eszena bat sortu eta kudeatu ahal izateko. Nabarmenena COLLADA bidez modeloak kargatzeko eta kamera mota ezberdinak sortu eta kudeatzeko eskaintzen dituen erraztasunak dira. Honi esker lan asko aurrezten da, ez baitago kameraren transformazio egokietarako lan asko egin beharrik, eta modeloak modu errazean kargatzea lortzen da, fitxategi mota zehatz batetik irakurtzeko funtzioak definitu behar izan gabe (wavefront motako fitxategiak adibidez).

Gainera, kolisio detekziora jada inplementatuta dauka, eta kode gutxi gehituta erabili daiteke baliabide hau. Honekin batera, Frustum Culling-a ere eszenako objektuen renderizatze prozesuan txertatuta dago, beraz, ez da koderik gehitu beharko alderdi honetatik.

Atal honetan aplikazioan beharko diren klaseen erabilpena azalduko da. Hala ere, funtzioak programatzen hasi aurretik, hasierako lerroetan aplikazioaren funtzio nagusia zein den adieraztea eskatzen du C3DL-k, web orriko Canvas etiketaren izenarekin batera. Honekin batera, aplikazioan erabiliko diren modelo guztiak erazagutzea eskatzen du. Erazagutu gabeko modelo bat erabiltzen saiatzen bada programatzailea, errorea emango du aplikazioak.

Hona hemen hasieran gehitu beharreko kodea:

```
//Funtzio nagusiaren izena eta pasako zaion canvas etiketaren izena
c3dl.addMainCallBack(canvasMain, "CanvasIzena");
//Aplikazioan erabiliko diren modeloen deklarazioa
c3dl.addModel("cube.dae");
```

Kodea 7: C3DL-ren hasierako deklarazio kodea

c3dl.Collada

c3dl.Collada klaseari esker COLLADA formatuan gordetako modeloak kargatzeko aukera eskaintzen da. Honetarako, nahikoa da klasearen instantzia bat sortzea, eta honen *init()* metodoari COLLADA fitxategiaren *path*-a pasatzea. Hau eginda, objektua prest egongo da jada eszenara gehitzeko.

```
var obj = new c3dl.Collada();
obj.init("cube.dae");
scene.addObjectToScene(obj);
```

Kodea 8: COLLADA objektuen karga

Beste hainbat metodo eskaintzen ditu klase honek bere erabilpenerako: Objektuari "izen" bat esleitu, transformazioak burutu, posizioa aldatu, objektuarekin elkarrekintza ahalbidetu ...

Objektuari izen bat esleitzea garrantzitsua da objektua kolisio detekzioan eta elkarrekintzetan kudeatu ahal izateko. Test hauetako bat burutzen den bakoitzean, emaitza objektu zerrenda bat izango da, non izenen bidez desberdintzen diren objektuak. Ondoko kodean izen esleipenaren adibide sinple bat azaltzen da, kolisio kudeaketarako erabilita.

```
obj.setName("Kuboa");
...
var ema = escene.getCollision();
for(var i = 0; i<ema.length; i++){
   if(ema[i].getName == "Kuboa") return true;
}
```

Kodea 9: Objektu baten izen esleipen eta erabilpena

Bestalde, OpenGL eta WebGL-n ohikoak diren transformazioak eragiteko metodoak ere erabil daitezke. Aplikazioan estatikoak diren objektuak erabiliko direnez,

soilik leku aldaketarako erabiltzen diren metodoak erabiliko dira, baina biraketa eta eskalatze funtzioak ere eskaintzen ditu klaseak.

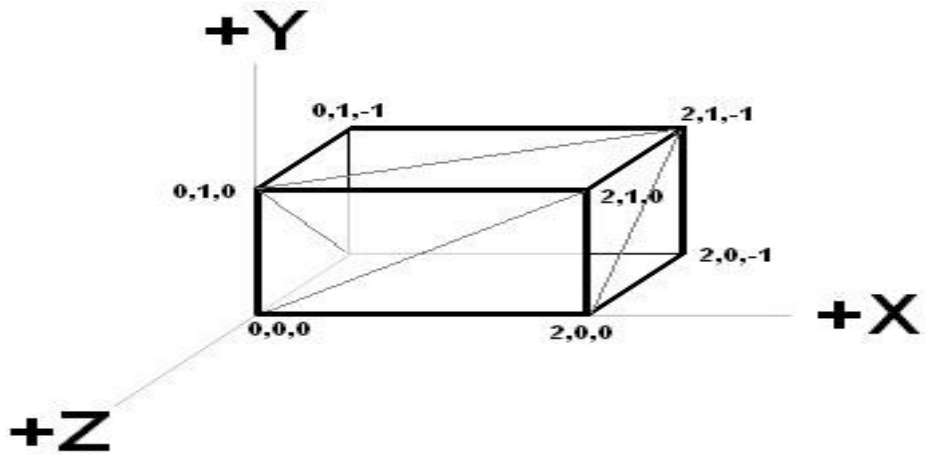
Leku aldaketarako bi metodo eskaintzen dira:

- *setPosition()*: Objektua adierazitako posizioan jartzen du, munduko erreferentzi sistemaren arabera.
- *translate()*: Objektuari translazio bat eragiten dio adierazitako balioen arabera.

```
obj.setPosition([1.0,0.0,0.0]); //Objektua  1.0,0.0,0.0 puntuan kokatuko da
obj.translate([1.0,0.0,0.0]); //Objektua X ardatzean 1.0 mugituko da
```

Kodea 10: Objektuaren leku aldaketa metodoak

Objektuen posizioa edo transformazio bat zehazterakoan, kontuan hartu beharra da C3DL-n OpenGL-ko erreferentzi sistema berdina mantentzen dela, beheko irudian agertzen dena izanik. X zabalera, Y altuera eta Z sakonera.



Irudia 21: C3DL-n OpenGL eta WebGL-ko erreferentzi sistema mantentzen da

Azkenik, objektua elkarrekintza testetan kontuan hartzeko, *setPickable()* metodoaren bidez objektua elkar-eragilea dela adierazi behar da, funtzioari *True* balioa pasatuz. Hau eginda, xaguarekin klik egiten den bakoitzean objektuaren gainean klikatu den aztertuko da.

```
obj.setPickable(true);
```

Kodea 11: Objektu elkar-eragilea

c3dl.FreeCamera

C3DL-k bi kamera mota eskaintzen ditu: c3dl.FreeCamera eta c3dl.OrbitCamera. Aplikaziorako c3dl.FreeCamera erabiltzea erabaki da, libreki mugitu daitekeen kamera bat definitzen baitu klase honek. c3dl.OrbitCamera, aldiz, espazioko puntu baten inguruan orbitatzera murriztuta dago, eta ondorioz ez dio aplikazioari inolako ekarpenik egiten oraingoz.

Klase honetan kameraren hasierako parametroak,mugimenduak eta biraketak kudeatzeko hainbat metodo eskaintzen dira. Edozein aldaketa eragin aurretik, kamera sortu eta hasieratu beharra dago, posizioa, begiratze puntua eta bertikalitatea adierazten duen bektorea definituz. Honekin batera, hurbileko eta urruneko planoak definitu behar dira perspektiba bistaren kalkuluetarako.

```
var cam = new c3dl.FreeCamera();
cam.setPosition([0.0,0.0,0.0]);
cam.setLookAtPoint([1.0,0.0,1.0]);
cam.setUpVector([0.0,0.1,0.0]);

cam.setNearClippingPlane(0.1);
cam.setFarClippingPlane(1000);

scene.setCamera(cam);
```

Kodea 12: Kamera berri baten hasieraketa

Parametro hauek definituta, kameraren erreferentzi sistema lortzeko kalkuluak klaseak berak egingo ditu. Hemendik aurrera kamera prest dago eszenan gehitu eta transformazioak jasateko.

Leku aldaketarako bi metodo erabili daitezke:

- *setPosition():* Kamera espazioko puntu zehatz batean kokatzen du.
- *setLinearVel():* Kamerari abiadura bektore bat esleitzen dio. Erloju Tick bakoitzean bektore hau bere posizioari gehitzen zaio, kamera mugiaraziz. Metodo honetan mugimenduaren animazioa egiten da.

Ondoko kodean kamera aurrera mugitzeko aginduak biltzen dira. Aurrera mugitu ahal izateko, kamerak nora begiratzen duen eskuratu beharra da *getDir()* metodoaren bitartez. Datu guztiekin, eta C3DL eskaintzen duen bektoreak biderkatzeko funtzioaren laguntzarekin, kameraren hurrengo mugimendua kalkulatzen da.

```
var mov = [0.0,0.0,0.0];
mov = c3dl.multiplyVector(cam.getDir(), 0.5, mov);

if(anim) cam.setLinearVel(mov);
else  cam.setPosition([pos[0]+mov[0],pos[1]+mov[1],pos[2]+mov[2]]);
```

Kodea 13: Kamera aurrera mugitzeko kodea

Errotazioak burutzeko metodo asko eskaintzen dira. Kameraren erreferentzi sistemako ardatz bakoitzean edota programatzaileak erabakitako ardatz baten inguruan errotazioak burutzeko aukera dago:

- *rotateOnAxis():* Kameraren zentrotik pasatzen den ardatz bat eta angelua pasata, kamera ardatz horren inguruan biratzen du.
- *pitch():* Kameraren erreferentzi sistemako X ardatzean biratzen da kamera.
- *roll():* Kameraren erreferentzi sistemako Z ardatzean biratzen da kamera.
- *yaw():* Kameraren bertikalitate bektorearen inguruan biratzen da kamera.
- *setAngularVel():* Kamerari abiadura bektore bat esleitzen zaio, tick bakoitzean errotazioari eragingo diona. Errotazio animazioa sortzen da honen bidez.

Ondoko kodean ikusi daiteke funtzio hauen erabilpena. Angelua radianetan adierazi behar denez, PI zenbakiaren hurbilpen bat erabiliko da angelua kalkulatzeko.

```
cam.rotateOnAxis([0.0,0.1,0.0], Math.PI*0.028); //Y ardatzaren inguruan biratu
cam.pitch(Math.PI*0.028); //Berdin roll() eta yaw() metodoekin
```

Kodea 14: Kameraren biraketa metodoak

Ikusten denez, kameraren kontrola oso sinplea eta erabilerraza da. Metodo gutxi batzuen laguntzarekin kamera modu egoki batean mugitu daiteke, honen erreferentzi

sistema kalkulatu eta egoki mantentzeaz arduratu beharrik gabe.

c*3dl.PositionalLight*

Hiru argi mota kudeatzeko klaseak definitu dira C3DL-n:

- c3dl.DirectionalLight: Norabidea soilik daukan argia, bere posizioa infinituan dagoela suposatuz. Eguzki moduko argiztapena eskaintzen du.
- ***c3dl.PositionalLight:*** Posizioa soilik definituta daukan argia. Bonbilla moduko argiztapena eskaintzen du, bere posiziotik norabide guztietan argiztatuz.
- **c3dl.SpotLight:** Posizioa eta norabidea definituta dituen argia. Foko moduko argiztapena eskaintzen du.

Aplikaziorako c3dl.PosiotionalLight klasea erabiltzea erabaki da, eszena osoa argiztatzeko argi bat edo bi soilik definitu behar izateko. Argi honen objektu bat sortu eta gero, honen intentsitatea, propietate difuso eta espekularra, posizioa etab. aldatzeko aukera eskaintzen da. Propietate guztiak definitu eta gero, argia "piztu" egin beharra da eszenan sartu aurretik.

```
var arg = new c3dl.PositionalLigth();
arg.setDiffuse([0.5,0.5,0.5,1.0]);
arg.setSpecular([0.0,0.0,0.0]);

arg.setPosition([0.0,2.0,0.0]);
arg.setOn(true);

scene.addLight(arg);
```

Kodea 15: Argia sortu eta eszenara gehitu

c3dl.Scene eta c3dl.WebGL

Bi klase hauen bidez sortzen da beranduago objektuak, kamera, argiak etab. sartu eta kudeatzeko erabiliko den eszena. c3dl.WebGL klasea renderizatzaile bat sortzeko erabiltzen da, geroago c3dl.Scene motako objektu batean erabiliko dena.

Lehenengo pausoa eszena objektua eta renderizatzailea sortzea da. Hau eginda, renderizatzailea eszenari esleitzen zaio, eta eszena objektuaren hasieraketa martxan jartzen da *init()* metodoarekin, web orriko Canvas etiketaren izena pasata.

```
var scn = new c3dl.Scene();
scn.setCanvasTag(canvasName);

// Renderizatzaile berria sortu
renderer = new c3dl.WebGL();
renderer.createRenderer(this);

// Renderizatzailea eszenari esleitu
scn.setRenderer(renderer);
scn.init(canvasName);
```

Kodea 16: Eszenaren hasieratzen prozesua

Behin eszena hasieratu dela, hasierako objektuak, kamera eta argiak gehitu behar dira eszena renderizatzen hasi aurretik. Honekin batera, posible da atzeko planoaren kolorea aldatzea, ingurune argiaren intentsitatea egokitzea etab. Ingurune argia kontuan izatea gomendatzen da, intentsitate handikoa bada eszenako objektu batzuk gehiegi argiztatzea eta ondo ikusteko aukera ez izatea gerta baitaiteke.

```
... //Collada objektua sortu
scn.addObjectToScene(obj);
... //Kamera sortu eta hasieratu
scn.setCamera(cam);
... //Argia sortu eta hasieratu
scn.addLight(arg);
scn.setBackgrounColor([0,0,0,0]); //Atzeko plano beltza
scn.setAmbientLight([0.4,0.4,0.4]); //Ingurune argia murriztu
```

Kodea 17: Eszenara osagaiak gehitzen eta parametroak aldatzen

Objektuak eta argiak eszenatik ateratzeko aukera ere dago noski, nahikoa da *removeObjectFromScene()* eta *removeLight()* metodoak erabiltzea, kendu nahi den osagaiaren aldagaia pasata; hau da, eszenara gehitzeko erabili diren parametro berdinak erabiltzen dira hauek eszenatik ateratzeko. Kameraren kasuan, uneko kamera beste kamera batekin ordeztu daiteke, baina ezin da eszenatik ezabatu, eszenak beti kamera bat izan behar baitu.

Hurrengo pausoa gertaerei erantzuteko erabiliko diren funtzioak eszenara esleitzea da. Funtzio hauek lehenetsitako parametro kopuru bat jasoko dute beti. Adibidez, teklatuan tekla bat sakatzen denean deituko den funtzioak parametro gisa teklaren kodea daukan gertaera bat jasotzen du, besterik ez. Hau kontuan izan beharra da funtzioak sortzerakoan. Hurrengo kode zatian teklatuaren gertaerak kudeatzeko metodoaren adibide bat azaltzen da.

```
function canvasMain(name){
...
//Tekla sakatzean eta uztean deituko diren funtzioak
scn.setKeyboardCallback(up,down);
//Eszena eguneratzean deituko den funtzioa
scn.setUpdateCallback(updateCb);
//Eszena gainean klik egitean deituko den funtzioa
scn.setPickingCallback(clickKudeatu);
...
}

function down(event){ sakatutakoTeklak[event.keyCode] = true;}
function up(event){ sakatutakoTeklak[event.keyCode] = false;}
function updateCb(time){ teklakKudeatu();}
function clickKudeatu(result){
//setPickable(true) daukaten objektuak jaso daitezke result aldagaiean
...
}
function teklakKudeatu(){
    //Teklak kudeatzeko kodea
}
```

Kodea 18: Gertaerak kudeatzeko funtzioen definizioa

Ikusten denez, teklaren bat sakatu edo uzten denean egiten den gauza bakarra tekla hori sakatuta dagoen edo ez array batean gordetzea da. Teklen kudeaketa beranduago egingo da, eszena eguneratzen denean, *teklakKudeatu()* funtzioaren bitartez.

Dena prest dagoenean, eszena "martxan" jarri beharko da *startScene()* metodoaren bitartez. Honen bidez objektuak renderizatzen, argiztapena aplikatzen etab. hasiko da aplikazioa.

```
//Eszena martxan jarri
scn.startScene();
```

Kodea 19: Eszena martxan jartzeko agindua

Hemendik aurrera gertaera guztiak kudeatuko dira, kolisioak barne. Gainontzeko gertaeren kudeaketarako funtzioak gehitu diren arren, kolisio kudeaketa beste modu batean egin beharra dago, jada eszenaren barruan inplementatuta baitago kolisioen detekzioa. Kolisioen detekzioa aktibatzeko nahikoa da eszenaren setCollision() eta setCollisionType() metodoak erabiltzea. Bigarren metodoan bi aukera posible daude: "Collada" eta "Geometry". "Collada" aukeratzen bada, kolisio detekzioan modeloen bounding-box-ak erabiliko ditu. Aldiz, "Geometry" aukeratzen bada, kolisio detekzioan modeloen geometria osoa hartuko da kontuan. Bi mota hauen artean edozein momentutan alda daiteke.

```
scn.setCollision(true);
scn.setCollisionType("Collada");
```

Kodea 20: Kolisio detekzioa aktibatzeko aginduak

Azkenik, kolisio detekzioak lortutako emaitzak eskuratu eta kudeatzeko, eszenaren *getCollision()* metodoa erabiltzen da. Honek kolisioak dituzten objektuen zerrenda bat itzultzen du. Lehen aipatu den bezala, objektu hauek identifikatzeko izena hartzen da kontutan. Ondoko kodean ikusi daiteke nola eskuratzen den kolisio detekzioaren emaitza eta hau kudeatzeko modu bat.

```
//Kolisioen emaitza eskuratu
var ema = scn.getCollision();
for(var i = 0; i<ema.length; i+=2){
   //Objektuak binaka tratatuko dira
   if(ema[i].getName() == "Kuboa" && ema[i+1].getName() == "Esfera")
         return true;
}
```

Kodea 21: Kolisio kudeaketa adibidea

4.3.2- Talkak

Kamera eta eszenako objektuen arteko talkak kudeatzeko objektuen arteko kolisio detekzioa burutuko da.

Bi objektu edo gehiagoren arteko ebakidura detektatzeko algoritmo edo teknikei deritzaie kolisio detekzioa. Kontzeptu hauek normalean bideojoko edota fisika simuladoreekin lotzen dira, baina robotika arloan ere aplikazioak dauzka. Kolisio detekziorako algoritmoak garatzerako orduan, algebra eta geometria inguruko kontzeptu asko ezagutu eta kontuan hartu behar dira emaitza onak lortu ahal izateko.

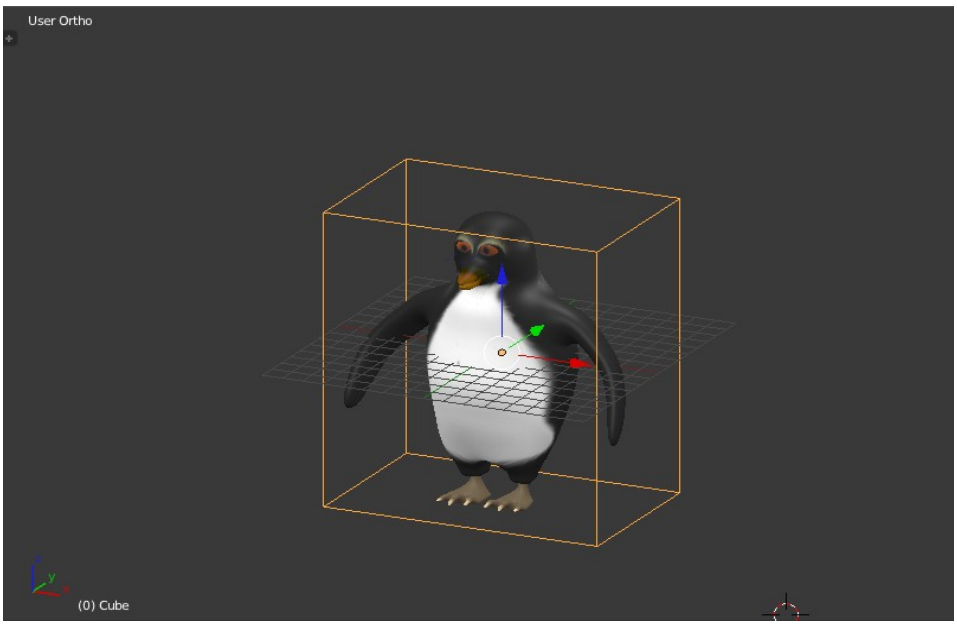
Behin objektuen arteko talkak kontrolatzea lortu dela, talka hauek sortzean zer erantzun eman beharra den zehaztu beharra da. Adibidez, joko batean jokalariak paretak zeharkatzea galarazi ahal izateko bi hauen artean kolisioa noiz gertatzen den kontrolatu behar da, kolisioa gertatzen denean jokalariari mugimendua galaraziz. Antzeko egoera sortu daiteke eszenako beste objektu batzuekin, edo ,adibidez, objektu zehatz bati ematen zaion erantzuna ate bat irekitzea edota objektua bultzatzea izan daiteke. Bestalde, fisika simulazio batean kolisio bat gertatzen denean, talka gertatu den momentuan objektu bakoitzak zeukan indarra, abiadura, norabidea, norantza etab. kontuan hartuta sortuko dira erantzunak.

Garatu den aplikazio honetan bilatu eta inplementatu dena kamera eta pareten arteko kolisio detekzioa izan da. Erabiltzaileak kamera baten kontrol osoa dauka, baina ez zaio sortutako modeloetako paretak zeharkatzea baimendu behar, errealismoa mantentzearren.

Nahiz eta ohikoena erabiltzailearen kameraren eta eszenako objektu guztien arteko kolisioak detektatzea izango litzatekeen, soilik paretekin izan ditzazken talkak kudeatzea erabaki da, kostu konputazionala txikitzearren. Gainera kamera gainontzeko objektu guztien gainetik pasatzen da, beraz ez dira objektua hauek kontuan hartu beharrik. Hau inplementatzeko modu ezberdinak aztertu eta erabili dira, hiru dimentsioko tekniketatik bi dimentsiokoetaraino.

BOUNDING-BOX bidezko kolisio detekzioa:

Hiru dimentsioetako aplikazioetan normalean inplementatzen den teknika aztertuko da lehenik: Objektuen bounding-box-en arteko kolisio detekzioa. Objektu bakoitzak inguratzen duen kutxa bat dauka (modeloaren puntuen koordenatu minimo eta maximoen bidez osatzen dena).



Irudia 22: 3D modeloa eta inguratzen duen bounding-box-a

Teknika honen bidez, kolisio detekziorako algoritmoek kutxen arteko intersekzioak detektatzea dute helburu. Honi esker, objektuen talkak detektatzea sinpleagoa da, baina zehaztasun asko galdu daiteke, objektuaren geometriaren arabera

Erabilitako C3DL motoreak teknika hau inplementatuta daukanez, hau erabiltzeko gehitu behar den kodea sinplea da. Ondoren agertzen den kodeak kamerarekin doan bounding-box-ak talkarik jasan duen detektatzen du.

```
scn.setCollision(true);
scn.setCollisionType("Collada");
...
var ema = scn.getCollision();
for(var i = 0; i<ema.length; i++){
   if(ema[i].getName()== "CamBound")
         return true;
}
```

Kodea 22: Kameraren kolisio kudeaketa Bounding-box bidez

Kamerari bounding-box egoki bat gehitzeko kubo bat erabiltzen da, momentu oro kameraren posizio berdina mantentzen duena. Paretekin kolisioak ongi detektatu eta kudeatzen dira, eta modu nahiko eraginkorrean, baina ateekin arazoak sortu dira. Bounding-box-ak modelo guztia inguratzen duenez, ezinezkoa da ateak zeharkatzea.

Arazo hau konpontzeko bi irtenbide pentsatu dira:

1. Arazoak sor ditzaketen modeloak berregituratu, gutxienez hiru objektu ezberdinetan banatuz ateak objektu horien arteko hutsunea izan dadin. Modu honetan ateak ez dira objektuen parte izango, eta bounding-box-ek ez dute oztoporik jarriko.
2. Kolisio detekzio sakonagoa burutzea; hau da, kolisioak ez dira bounding-box-en arabera egingo, objektuen geometria edo sare poligonalaren arabera baizik. Honela ez dago modeloak berregituratu beharrik, maila honetako kolisio detekzioak atearen hutsunea ondo kontrolatuko baitu.

Modelo asko aldatzeko beharra sortzen da lehenengo aukerarekin, beraz bigarren aukerarekin aurrera jarraitzea erabaki da.

Objektuaren GEOMETRIAN oinarritutako kolisio detekzioa:

Kasu honetan, kolisio detekziorako objektuaren geometria zehatz osoa aztertzen da. Puntu, lerro eta plano guztiak aztertzen dira bi objekturen artean intersekziorik dagoen ikusteko, eta ondorioz kolisio zehatzagoak detektatzea posible da. Honi esker, aurreko kasuan ateekin sortzen zen arazoa ekiditzea lortuko da, atearen hutsunea ere aztertzen baita.

Zorionez C3DL-k kolisio mota hau ere inplementatuta dauka, beraz kolisio mota hau detektatu eta kudeatzeko kodea gehitu eta probak egitea erraza da. Aurreko kasuko kode berdina mantentzen da, kolisio motaren zehaztapena izan ezik.

```
scn.setCollision(true);
scn.setCollisionType("Geometry");
...
var ema = scn.getCollision();
for(var i = 0; i<ema.length; i++){
   if(ema[i].getName()== "CamBound")
         return true;
}
```

Espero bezala, ateak zeharkatzea lortu da, eta paretetan kolisioak ondo detektatzen dira. Hala ere, konputazio kostu aldetik igoera nabarmena izan da, aplikazioaren errendimendua asko jaitsi da eta ondorioz kameraren mugimenduak asko moteldu dira.

Arazo honen arrazoia nabarmena da: Kolisio detekzioa burutzen den bakoitzean, objektuaren geometria osoa begiratzen da. Nahiz eta modeloak oso konplexuak ez diren, hauen geometriak karga handi bat sortzen du teknika hau erabiltzerako orduan.

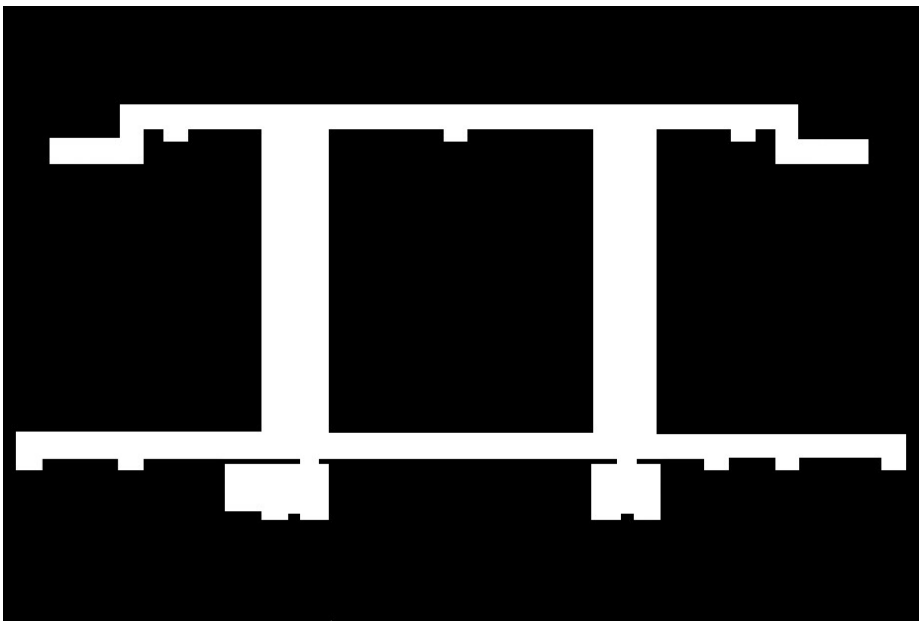
Arazo hau konpontzeko, kolisioak bi dimentsioetan kudeatzea erabaki da. Honetarako bi aukera berri pentsatu dira:

1. Kamera non mugitu daitekeen definitzen duen plano bat sortu, eta kamera mugitzean honen ertzekin ebaketarik dagoen aztertu. Ebaketarik sortzen bada, kolisio modura kudeatuko da eta kamera ebaketa aurretik zeukan posiziora itzuliko da.
2. Modeloa oinarritzat hartuta, txuri eta beltzezko mapa bat sortu. Kolore zuriak kamera non mugitu daitekeen adierazten du. Aplikazioan, kamera mugitzen den bakoitzean, mapan dagokion pixelaren kolorea aztertuko da. Kolore zuria bada, kamerak aurrera jarraitu dezake. Kolore beltza bada, kamerak ezin du jarraitu eta aurreko posizioan mantenduko da.

Bi aukera hauetatik bigarrena aukeratu da. Nahiz eta mapak sortzerako orduan zehaztasun handia beharko den, kostu konputazional oso baxua eskaintzen du. Honi esker aplikazioaren errendimendua ez da okertuko, eta erabiltzaileak arazorik gabe mugitu ahalko du kamera.

MAPAN oinarritutako kolisio detekzioa:

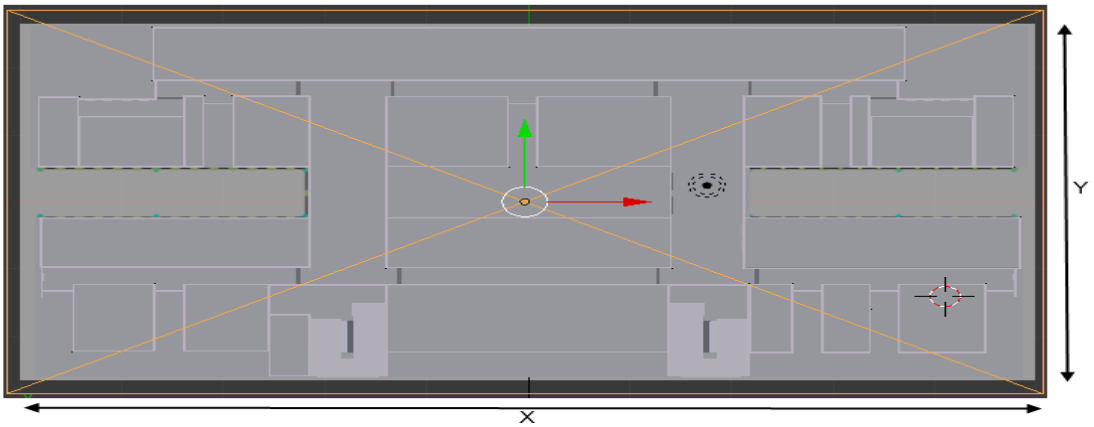
Bi dimentsiotako kolisio detekzioa mapen bidez egin ahal izateko, txuri eta beltzeko mapak erabiltzen dira kasu honetan.



Irudia 23: Fakultatearen 2.solairuko mapa

Blender-en laguntzarekin objektuen materialen kolorea eta ezaugarriak aldatu eta solairu bakoitzeko modeloaren txuri-beltzeko mapak sortu dira , goiko bistaren (ortogonalean) render baten bidez. Lortutako irudiei azken ukitu batzuk eman eta gero, kolisio detekzioan erabiliko diren mapak lortu dira. Hurrengo pausoa modelo bakoitza dagokion mapan mapeatzea izan da; hau da, kamera modeloan mugitzen den heinean, honen posiziotik abiatuta irudi-mapan zein pixeletan dagoen kalkulatu ahal izatea.

Honetarako modeloaren dimentsioak eta dagokion irudi-maparen altuera eta zabalera ezagutzea nahikoa da.



Irudia 24: Modelo osoaren dimentsioak kalkulatzen

Datu hauek bilduta, ondoko algoritmoaren bidez, kameraren posizioari irudian dagokion pixela kalkulatu daiteke, eta honen kolorearen arabera mugimendua baimendu edo debekatu.

```
a = kameraPos.x;
b = kameraPos.y;
pixX = (solairu.DimX/2 + a)*(mapaZab/solairu.DimX);
pixY = (solairu.DimY/2 - b)*(mapaAlt/solairu.DimY);

pixel = canvas.getPixel(pixX,pixY);
if(pixel == txuria){
      //Mugimendua baimendu
      return true;
}
else{
      //Mugimendua debekatu
      return false;
}
```

Kodea 23: Kolisio detekzio algoritmoa

Algoritmo honi esker, kolisioak modu egoki eta eraginkorrean kontrolatzea lortzen da. Gainera, tick bakoitzean egin beharreko gauza bakarra kameraren posizioari dagokion pixela eskuratzea da. kostu konputazional txikia gehituz prozesuari, eta ondorioz aplikazioaren errendimendu on bat mantenduz.

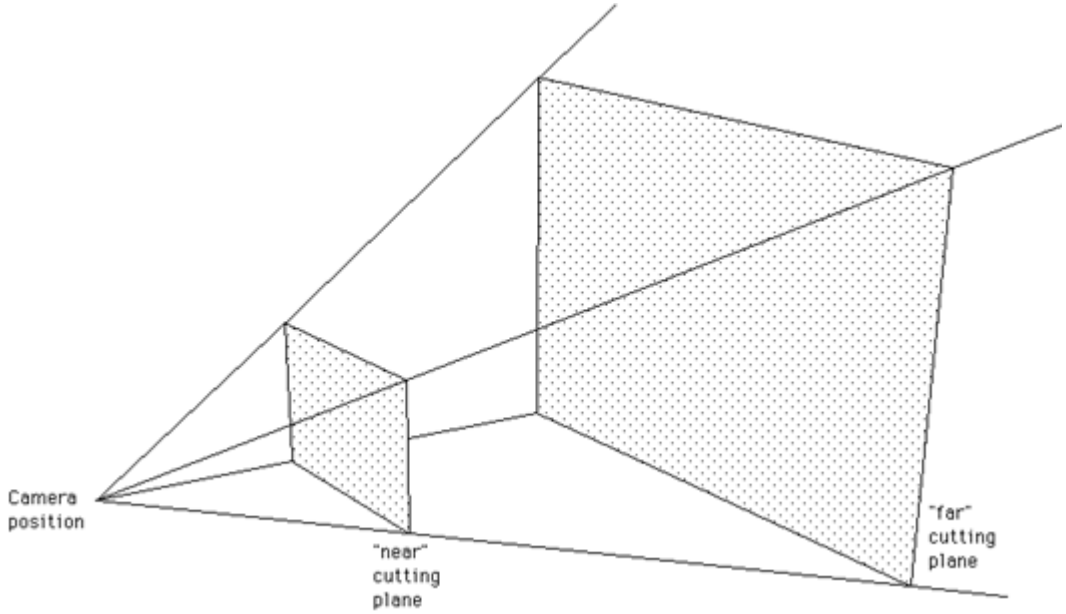
Irudiaren pixelak arazorik gabe eskuratu ahal izateko, mapak HTML5-eko canvas etiketa batean kargatzen dira. Egoera hau aprobetxatuz, aplikazioari osagai berri bat gehitzea erabaki da: Minimapa. Mapa kargatuta dagoenez, eta gainera kamerari mapan

dagokion pixela ezaguna denez, mapan puntu gorri baten bidez markatzen da kameraren posizioa. Modu honetan erabiltzaileak mapan bere kokapena begiratu dezake.

4.3.3- Frustum Culling

Kameraren perspektiba bistan definitutako planoen artean sartzen ez diren objektuak renderizaziotik kanporatzeko teknikari deritzaio Frustum Culling-a. Sinpleago esanda, ikusten ez dena ez da marrazten.

Teknika honi esker ikusten ez diren objektuak ez dira renderizatzera bidaltzen, eta ondorioz aplikazioaren eraginkortasuna hobetzen da, denbora gutxiago erabiltzen baita tick bakoitzean marrazterako orduan.



Irudia 25: Kameraren frustum bista

Frustum bistaren piramidea osatzeko atzeko edo "near" planoaren eskuineko koordenatua, ezkerreko koordenatua, goiko koordenatua, beheko koordenatua eta ikuslearekiko distantzia eta aurreko edo "far" planoaren ikuslearekiko distantzia behar dira. Datu hauekin piramidea definitu daiteke.

C3DL-n jada inplementatuta dago teknika hau, eta eszenako objektuak marrazterako orduan aplikatzen da beti.

5-ONDORIOAK ETA ETORKIZUNEKO LANAK

5.1- Ondorioak

5.2- Etorkizuneko lanak

5.1- Ondorioak

Proiektuaren garapenari amaiera emanda, lortutako emaitzak eta hasiera batean finkatutako helburuak konparatu daitezke. Zorionez, proposatutako puntu garrantzitsuenak burutzea lortu da:

- WebGL, HTML5 eta javascript soilik erabilita aplikazioa garatzea lortu da. Honi esker, erabiltzaileak web nabigatzaile bat bakarrik beharko du aplikazioa erabiltzeko, plugin-ik instalatu behar gabe.
- Fakultateko 3D modeloak web aplikazio batean kargatzea lortu da, testura guztiekin.
- Modeloak ikusteko kamera baten funtzionamendu egokia lortu da, teklatuaren kontrolpean.
- Kamera eta modeloetako pareten artean kolisioak detektatu eta kudeatzea lortu da.
- Erabiltzaileak modeloetako ateen bidez gela ezberdinen modeloetan barneratzea lortu da.
- Solairuz aldatzeko bi bide sortu dira: Web orriko menuaren bitartez edota modeloetako eskaileretara hurbilduz.

Baina proiektuaren emaitzak ez dira soilik garatu den aplikaziora murrizten. Garatzailearen ezagutza aldetik ere aurrerapenak lortu dira.

- Proiektu "handi" baten konplexutasuna hasiera batean egindako plangintza eta analisien arabera aldatu daiteke. Plangintza on batek atazen banaketa hobe bat ahalbidetzen du, eta edozein arazoren aurrean erantzun hobeak emateko aukera. Hala ere, beti da zaila egin beharko den lanaren estimazioak egiten, honetarako proiektuaren ataza bakoitzean egin beharrekoa ondo ezagutu behar baita.
- OpenGL eta WebGL-ren inguruan ezagutzak bildu dira, batez ere hauen barne funtzionamenduaren ingurukoak. Honekin batera, WebGL-rentzat sortu diren hainbat liburutegi eta motore aztertu izanak etorkizun baterako balio dezake,

beste proiektu handi batean honelako zerbait erabili behar den kasuan.

- Poligono bidezko 3D modelatzean esperientzia gehiago eskuratu da, eta Blender erabiltzeko erraztasuna hobetu egin da modeloen garapen prozesuan zehar.
- Nahiz eta proiektu hau pertsona bakar batek garatu duen, talde lana da beti egokiena, honela atazak pertsona ezberdinen artean banatu baitaitezke, garapen prozesua azkartzeko. Gainera, espezialitate desberdinetako taldekideak izanda, bakoitzari ongien egokitzen zaion ataza esleitzean emaitza egoki eta azkarrak eskuratzeko probabilitate handiak daude.

5.2- Etorkizuneko lanak

Nahiz eta karrera amaierako proiektu hau puntu honetan bukatu den, garatutako aplikazioak oraindik hobekuntzak eta baliabide gehiago eskaini ditzake berarekin lanean jarraitzen bada.

0.Fakultatearen modelatzea bukatu

Aplikazioak arrera ona badauka, garrantzitsua izango litzateke modelatzeko falta diren solairuak sortzea, fakultate OSOA bisitatu ahal izateko. Bestalde, modeloen itxura hobetzeko testura hobeak sortu edo eskuratzea ondo egongo litzateke.

1.Aplikazioaren web orria atzigarri

Bere etorkizun hurbilena Internet bidez edozein nabigatzailek aplikazioa erabiltzeko aukera eskaintzea da. Nahiz eta oraindik ez den fakultate guztia modelatu, modeloak gehitzen diren bitartean erabiltzaileek egindakoa aprobetxatu dezakete fakultatearen zati bat bisitatu ahal izateko.

2.Informazio gehiago eskaini erabiltzaileari

Informazio puntuak txertatu daitezke modeloetan, erabiltzaileari momentuan dagoen solairuaren informazio gehiago eskainiz, edota hurbilen dauzkan geletan zein irakasgai lantzen diren eta hauen ordutegia zein den adierazteko balio dutenak. Ezaugarri hau garatzea benetan interesgarria izango litzateke, bisitaz gain informazio idatzi garatuago bat eskaintzen baizaio erabiltzaileari.

3.Modeloei bizitza gehiago eman

Modeloetan mugitzen diren pertsonak gehitu daitezke, bisitari bizitasuna eta errealismo gehiago gehituz. Honek soilik alde estetikoari eragiten dio, baina kontuan izan beharrekoa da hasieran dena begietatik sartzen dela.

4.Kolisio detekzioa hiru dimentsioetara eraman

Memoria honetan azaldu den bezala, kamera eta moeloaren arteko kolisioak bi

dimentsiotako mapen bidez burutzen dira. Etorkizunean hobetu beharreko puntu garrantzitsu bat da hau. Kolisio detekzioa hiru dimentsiotan modu eraginkor batean egiteko teknika edo algoritmoren bat garatzea egokia izango litzateke, batez ere mapak sortzeko lana aurrezteko.

5.Beste fakultateetara zabaldu

Sortutako aplikazioa berrerabili daiteke UPV/EHU-ko beste fakultateen bisita birtualak garatu ahal izateko. Egin beharko zen lan handiena 3D modeloak sortzea izango litzateke.

6-BIBLIOGRAFIA

Joseba Makazaga, Asier Lasa (). *Ordenadore bidezko irudigintza*.

C3DL-ren inguruko informazioa eta dokumentazioa

http://www.c3dl.org

Blender

Roland Hess (2011). *Blender*.

http://www.blender.org

http://www.blenderguru.com/

Wikipedia

http://es.wikipedia.org

OpenGL informazioa

http://www.opengl.org/

WebGL informazioa

https://www.khronos.org/registry/webgl/specs/1.0/

http://learningwebgl.com/blog/

http://www.jlabstudio.com/webgl/

7-ERANSKINAK

7.1- Erabiltzailearen eskuliburua

7.2- Diseinuaren UML diagramak

7.1- Erabiltzailearen eskuliburua

SARRERA

Ongi etorri UPV/EHU-ko Informatika fakultateko bisita birtualeko erabiltzailearen eskuliburura. Dokumentu honetan esku artean daukazun aplikazioa erabili ahal izateko beharko dituzun argibide guztiak azalduko dira, hainbat irudiren laguntzarekin. Behin dokumentu hau irakurrita, aplikazioak eskaintzen duen guztia ikusi eta probatu ahalko duzu.

Kontuan izan aplikazioan ikuskatzen diren hiru dimentsiotako modelo bat edo beste oraindik guztiz bukatu gabe daudela, beraz ez kezkatu momenturen batean grafikoetan gauza arraroren bat topatzen baduzu, laster konponduko baita zure aldetik aldaketarik egin behar gabe.

APLIKAZIOAREN AURREBALDINTZAK

Aplikazioa WebGL teknologian oinarrituta sortu denez, abantaila handi bat eskaintzen du: Ez dago nabigatzailearendako inolako plugin edo gehigarririk deskargatu eta instalatzeko beharrik.

Hala ere, dena ez baita posible, aurrebaldintza batzuk bete behar dira. Ordenagailuko txartel grafikoa WebGL eta hardware bidezko 3D azelerazioa soportatzeko prest egon beharra da aplikazioa erabili ahal izateko, eta, honekin batera, WebGL onartzen duen web nabigatzaile bat erabili beharra da.

Dokumentu hau idatzi den momentuan WebGL onartzea baieztatzen duten nabigatzaileak hauek dira:

- Google Chrome v19.0.1084.56
- Opera v12.00
- Mozilla Firefox-en “Nightly” bertsio bat

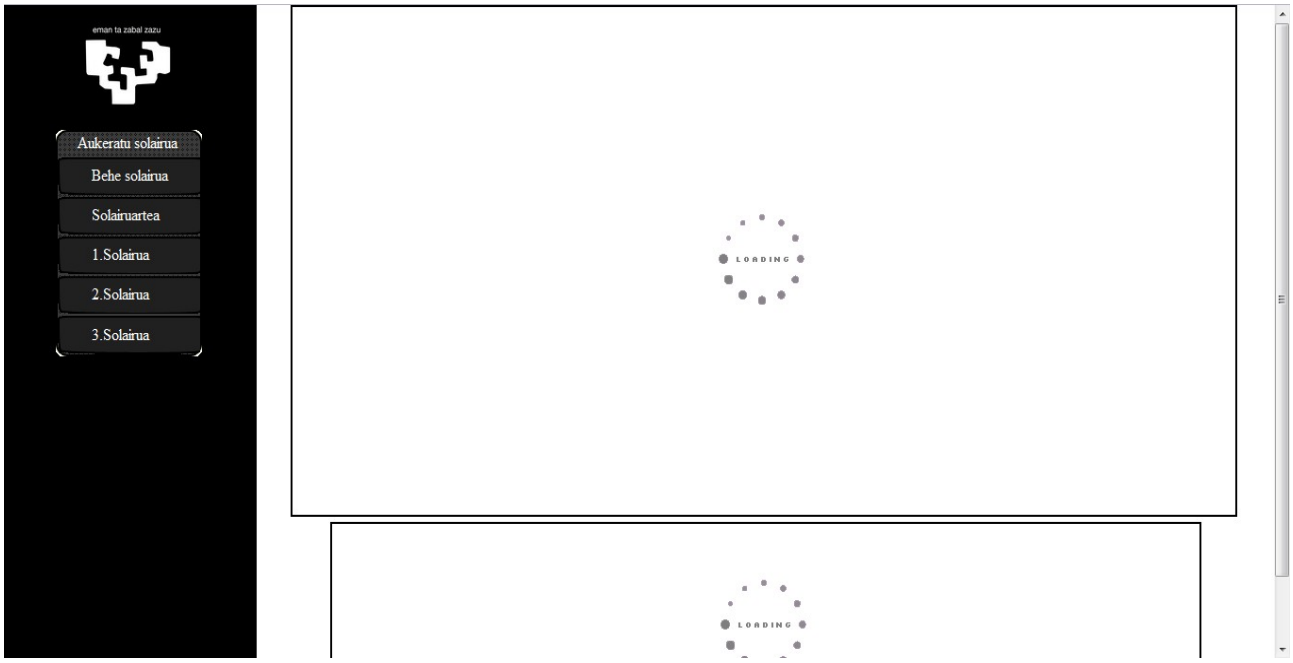
Aplikazioa Google Chrome nabigatzailean erabili da une oro, emaitza egokiekin eta inolako arazorik gabe. Opera eta Firefox-en egindako probetan, ordea, erroreak sortu dira. Beraz, Google Chrome erabiltzea gomendatzen da aplikazioaren funtzionamendu egokia bermatzeko.

APLIKAZIOAREN ERABILPENA

Aplikazioa erabili ahal izateko, lehenik eta behin honen web orrian sartu beharko da, ondoko estekari jarraituta:

http://127.0.0.1/KAP/index.html

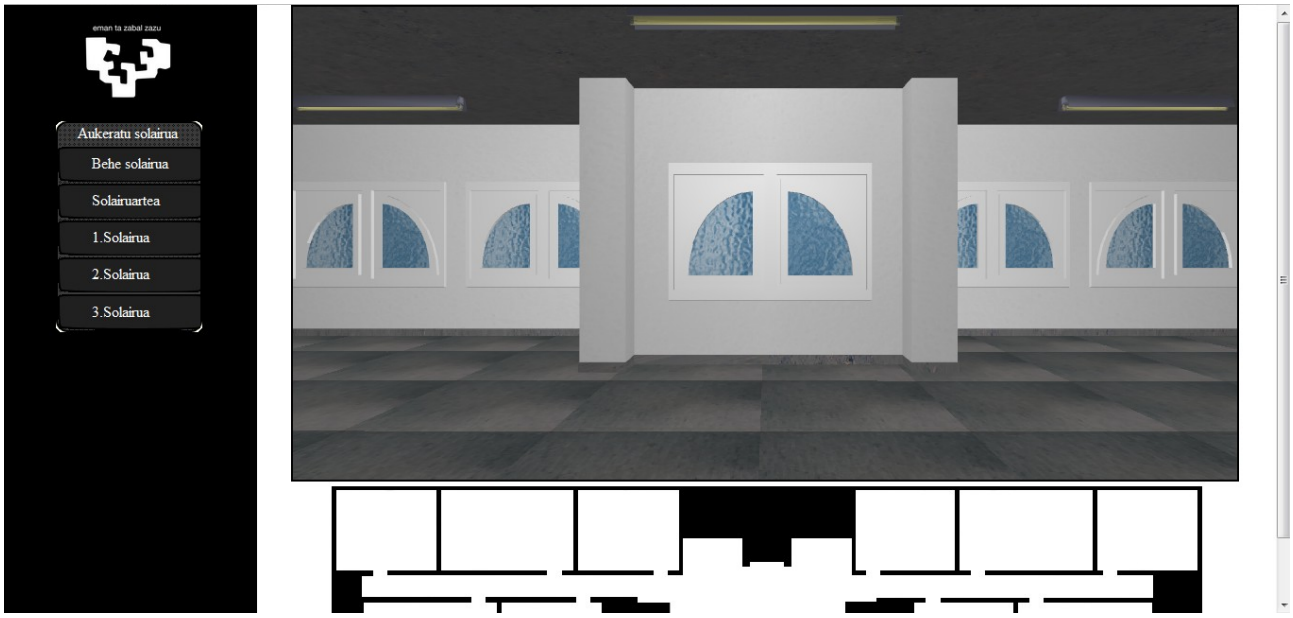
Web orrian sartu bezain laster, ezker aldean bisitatu nahi den solairua aukeratzeko menua agertuko da, eta eskuinalde osoan bi kutxa handi LOADING adierazten dutenak. Goiko kutxan fakultatearen 3D modeloak bisitatzeko aukera eskainiko da, eta beheko kutxan, berriz, momentuan kargatuta dagoen solairuaren mapa bat.



Web orriaren hasierako egoera: Modeloen karga prozesua

Pazientzia apur bat izan beharko da aplikazioaren osagai guztiak kargatu arte. Hainbat modelo sarearen bidez kargatu behar direnez, denbora bat kostatzen zaio hasieran. Hala ere, behin dena kargatu dela, ez da berriro itxarote momenturik izango aplikazioa irekita mantentzen den bitartean.

Karga prozesua bukatuta, fakultatea bisitatzeko prest egongo da dena eta zuzenean kargatuko da fakultateko solairuartea eta kontrolatuko den kamera hasiera puntuan kokatuko da. Puntu honetatik aurrera erabiltzaileak aplikazioaren kontrol guztia izango du.



Bisita birtualaren lehenengo bista: Solairuartea

Orain argi ikusten da aplikazioaren itxura osoa. Ezkerrean solairuen menua, eta eskuin aldean bisitarako "leihoak". Bisitako leihoan kamera modeloaren barruan mugitzeko aukera dago orain azalduko diren kontrolen bidez.

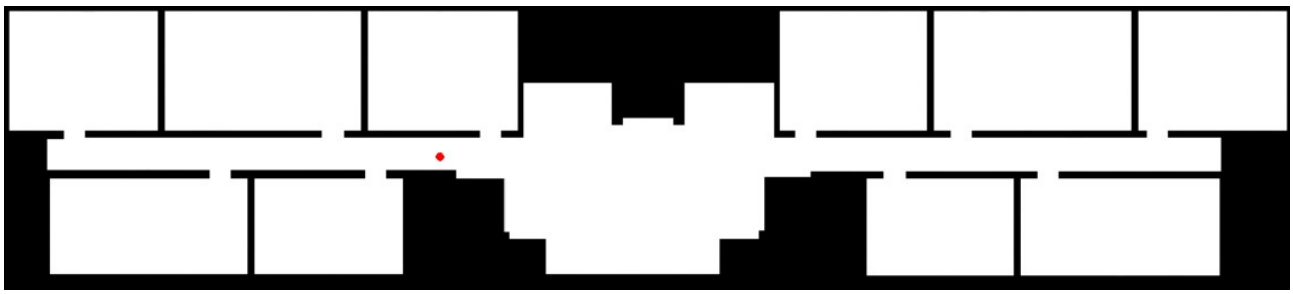
Kameraren kontrolak:

Kamera kontrolatzeko ohikoak diren teklak erabiliko dira:

- W tekla: Kamera aurrerantz mugituko da.
- S tekla: Kamera atzerantz mugituko da.
- A edo ← tekla: Kamera ezkerrerantz biratuko da.
- D edo → tekla: Kamera eskuinerantz biratuko da.
- Q tekla: Kamera ezkerrerantz mugituko da.
- E tekla: Kamera eskuinerantz mugituko da.
- Z edo ↑ tekla: Kamerak gora begiratuko du.
- X edo ↓ tekla: Kamerak behera begiratuko du.

Kamera kontrolatzen den bitartean, modeloen ikuskatze kutxaren behean dagoen mapan kameraren uneko posizioa azaltzen da momentu oro, puntu gorri baten bidez. Momenturen batean galduta sentitzen bada erabiltzailea, maparen bidez gidatu daiteke

modeloan zehar.



Solairuarteko mapa

Solairua aldaketa:

Uneko solairuaz aspertu edo dena bisitatu eta gero, solairuz aldatzeko aukera dago. Honetarako bi bide daude:

1- Solairuko eskaileren bidez.

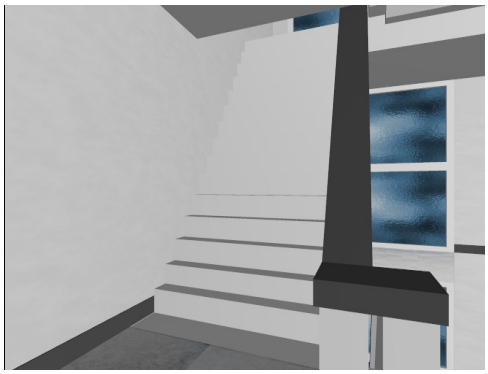
2- Web orriaren ezker aldean dagoen menuaren bidez.

Eskailera bidezko solairu aldaketa:

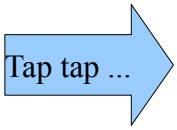
Eskailera bidez solairuz aldatzeko modua ezaguna eta ulertzeko sinplea da. Uneko solairuko eskailerak bilatu behar dira lehenik, eta behin hauetara nahikoa hurbilduta, solairu aldaketa gertatuko da zuzenean.

Solairuetako bat bukatu gabe, edo aldaketa prozesuan badago, nahiz eta eskailera egokira hurbildu, solairua ez da kargatuko, dagozkion aldaketak bukatu arte.

Modelo guztiak hasieran kargatu direnez, solairu aldaketa itxaron beharrik gabe gertatzen da. Solairu aldaketarekin batera, honen behean agertzen den mapa ere eguneratzen da, uneko solairuarena kargatuz.



2.Solairuko eskailerak



3.Solairuko eskailera aurreko sarrera

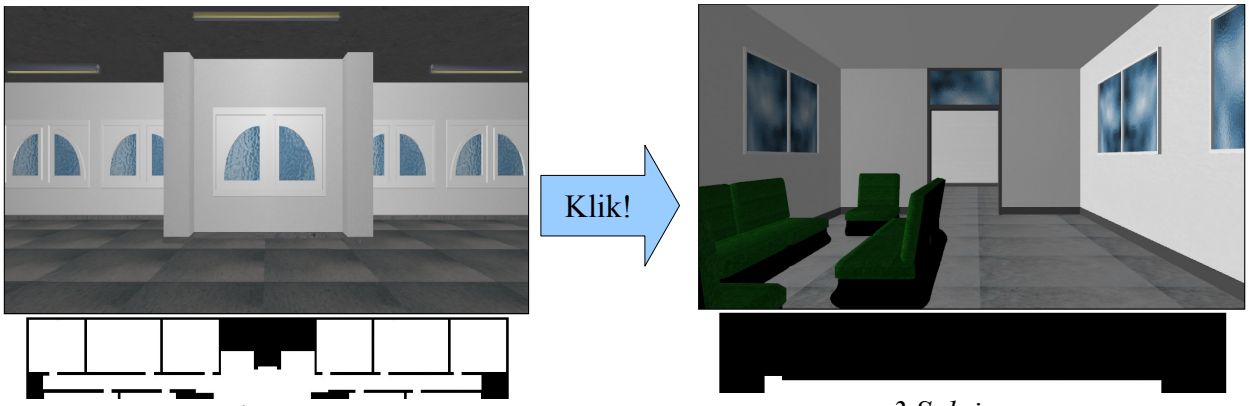
Menu bidezko solairu aldaketa:

Solairua aukeratzeko menua

Menuaren bidez modu erraz eta azkarrean aldatu daiteke bisitatu nahi den solairua. Nahikoa da menuan agertzen den aukeretako baten gainean klik egitea zuzenean solairua aldatzeko.

Solairuetako bat bukatu gabe, edo aldaketa prozesuan badago, nahiz eta dagokion botoian klik egin ez da solairua kargatuko, dagozkion aldaketak bukatu arte.

Modelo guztiak hasieran kargatu direnez, solairu aldaketa itxaron beharrik gabe gertatzen da. Solairu aldaketarekin batera, honen behean agertzen den mapa ere eguneratzen da, uneko solairuarena kargatuz.



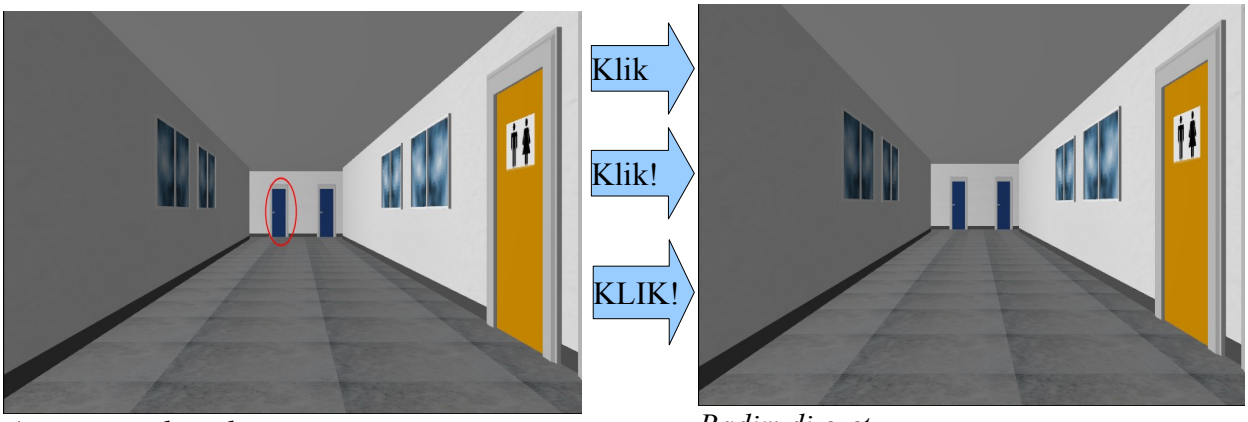
Solairuartea *3.Solairua*

Fakultateko ateen misterioa

Solairu batzuetan ate asko ikusi ahalko dira. Batzuk dekoraziorako erabiltzen dira, baina badira gela ezberdinetara sartzeko balio duten ateak.

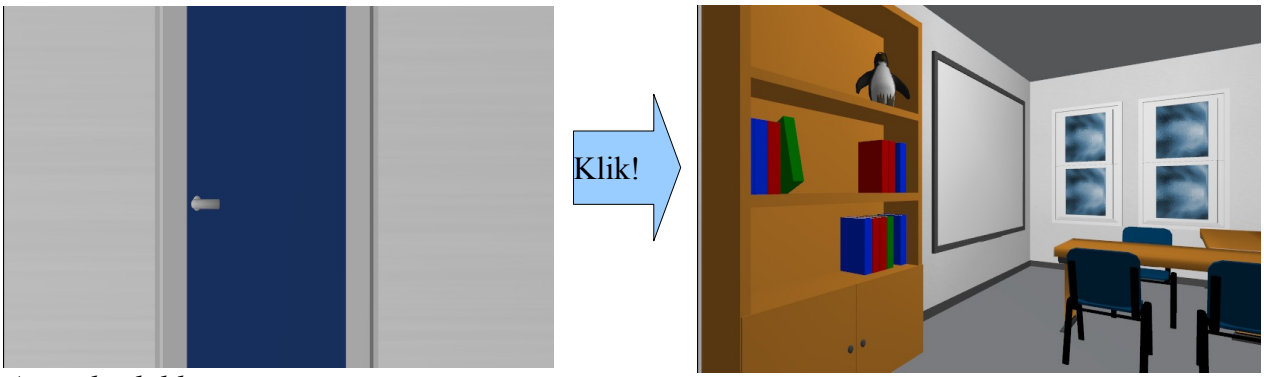
Nola erabili ate hauek geletan sartzeko? Oso erreza da ate hauek erabiltzea. Nahikoa da ate gainean klik egitea eta zuzenean gelara sartuko du bisitaria. Baina honek atetik gertu dagoenean bakarrik izango du eragina noski, beraz atera hurbildu beharko da hau egin ahal izateko. Bisitariak ezingo du pasilloaren beste puntan dagoen ate bat gelaz aldatzeko erabili.

Hona hemen adibide grafiko sinple bat:



Atea urruti dago baina agian ...

Badirudi ezetz ...



Agian hurbilduta ...

Oraingoan bai! Bulegoa ikusi daiteke!

Ate askoren atzea bulegoak daude, baina badira laborategietara eta beste gela batzuetara bideratzen dituztenak ere.

Gela berria bisitatu eta gero, bertatik atera ahal izateko atean berriro klik egitearekin nahikoa izango da. Hau eginda, gelara sartu baino lehen zegoen leku berdinera itzuliko da bisitaria.

7.2- Diseinuaren UML diagramak

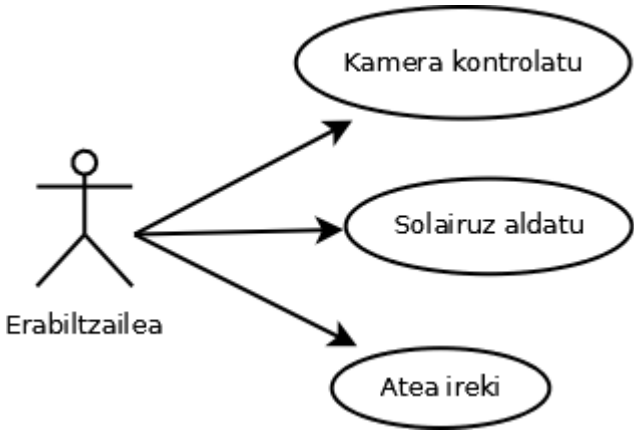
Atal honetan aplikazioa diseinatzerako orduan sortu diren diagrama ezberdinak azalduko dira.

7.2.1- Erabilpen kasuak

Erabilpen kasuak definitu aurretik, aplikazioan parte hartzen duten "aktoreak" identifikatu behar dira. Aktore bat aplikazioarekin elkar-eragiten duen kanpoko entitate bat da.

Aplikazio honen kasuan, erabiltzailea da parte hartzen duen aktore bakarra. Kasu batzuetan sistema administratzaileak etab. hartzen dira kontuan, baina kasu honetan, aplikazioak ez du inolako mantentze-lanik behar, ez baita erabiltzaileen kudeaketarik etab. burutu behar, eta 3D modelo berriak sartzerako orduan, lan guztia zerbitzarian egin beharko baita.

Erabiltzailea aktore bezala hartuta, hurrengo pausoa honi dagozkion erabilpen kasuak definitzea. Erabilpen kasuek aktorearen ikuspuntutik aplikazioak eskaintzen duen portaera definitzen dute; hau da, aplikazioak eskaintzen dituen funtzionalitateen deskribapenak dira.



Irudia 26: Erabilpen kasuen diagrama

Goiko irudian erabiltzaileari dagozkion erabilpen kasuen diagrama ikusi daiteke. Bertan hiru funtzionalitate desberdintzen dira:

- **Kameraren kontrola:** Teklatu bidez kamera kontrolatzeko aukera, kameraren ezaugarriak aldatuz. Kolisio detekzio eta kudeaketa burutzen da kamera mugitzen den bitartean. Honekin batera, kamera area zehatz batean barneratzean solairuz aldatuko da, eszena garbitu eta solairu berriaren eta honi dagozkion ateen modeloak kargatuz.
- **Solairu aldaketa:** Erabiltzaileak solairuz aldatu dezake web orriko menu baten bitartez. Honek eszena garbitu eta solairu berriaren eta honi dagozkion ateen modeloak kargatzea eragiten du.
- **Ateen irekiera:** Erabiltzaileak modeloetako ateetan klik eginda gela edo solairu desberdinen modeloak kargatu ditzake.

Ondoko tauletan erabilpen kasu hauen ezaugarri zehatzak biltzen dira.

Erabilpen kasua: KAMERA KONTROLATU	
Deskripzioa	Teklatuaren bidez kamera mugitu eta biratzen da, kolisio detekzioa burutuz eta gertaera areetan sartzen den aztertuz.
Aktorea	Erabiltzailea
Aurrebaldintza	Aplikazioaren web orria irekita, kamera eta modeloak kargatuta
Ohiko fluxua	- Teklatuaren sarrerak kontrolatu - Kamerari biraketak eragin eta hurrengo posizioa kalkulatu - Posizio berriarekin kolisiorik gertatzen den egiaztatu - Kolisiorik badago: Kamerari aldaketarik ez eragin - Kolisiorik ez badago: Kamerari posizio berria esleitu - Gertaera area batean sartu den aztertu - Sartu da: Solairua eta kameraren ezaugarriak aldatu
Bukaera egoera	Kolisiorik gertatu ez bada: Kamera lekuz aldatuta edo/eta biratuta Gertaera area batean sartu bada: Kamera solairu berri batean kokatuta

Taula 10: "Kamera kontrolatu" erabilpen kasuaren zehaztapenak

Erabilpen kasua: SOLAIRUZ ALDATU	
Deskripzioa	Web orriko menuko aukeretan klik eginda solairu berri bat kargatzen da.
Aktorea	Erabiltzailea
Aurrebaldintza	Aplikazioaren web orria irekita, kamera eta modeloak kargatuta
Ohiko fluxua	- Erabiltzaileak web orriko menuko aukera batean klik egiten du - Eszenan kargatuta dauden modeloak bertatik ezabatu - Aukeratutako solairuaren eta honi dagozkion ateen modeloak kargatu. - Kameraren posizioa eta begiratzen puntu berriak zehaztu
Bukaera egoera	Solairu berriaren modeloa kargatuta eta kamera birkokatuta.

Taula 11: "Solairuz aldatu" erabilpen kasuaren zehaztapenak

Erabilpen kasua: ATEA IREKI	
Deskripzioa	Bisita birtualeko ate batean klik eginda, dagokion gela edo solairua kargatzen da.
Aktorea	Erabiltzailea
Aurrebaldintza	Aplikazioaren web orria irekita, kamera eta modeloak kargatuta, kamera atetik gertu

Ohiko fluxua	- Erabiltzaileak ate gainean klik egiten du.
	- Kamera atetik hurbil badago:
	  - Solairu batean bagaude, solairuaren informazioa gorde.
	  - Uneko eszena garbitu eta dagokion gela edo solairua kargatu.
	  - Kameraren posizioa eta begiratze puntu berria zehaztu.

Bukaera egoera	**Solairu batean bageunden:** Gela baten modeloa kargatuta eta kamera birkokatuta. **Gela batean bageunden:** Gelan sartu aurretik geunden solairua kargatuta eta kamera birkokatuta.

Taula 12: "Atea ireki" erabilpen kasuaren zehaztapenak

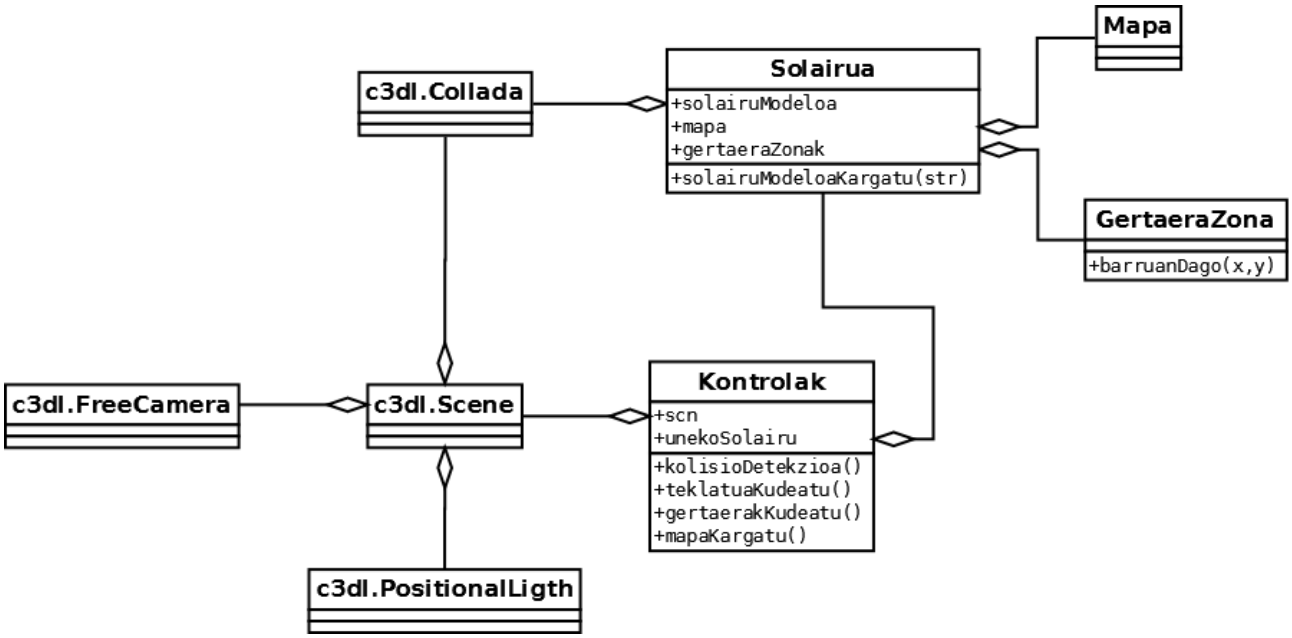
7.2.2- Klase diagrama

Aplikazioaren diseinuko hurrengo pausoa klaseen diseinua burutzea da, klase diagrama batean laburbiltzen dena. Klase hauen bidez, aplikazioan erabiliko den informazioa kudeatzeko erraztasuna eskaintzen duen interfaze moduko bat sortuko da, objektu abstraktuak erabiliz informazio sakabanatuaren ordez.

Ondoko objektuen beharra sortzen da:

- **Kamera:** Erabiltzaileak bisitan erabiliko duen kamera kudeatzeko klase bat beharko da. Honi esker kameraren posizioa, begiratze puntua etab. kudeatzeko erraztasuna lortuko da. C3DL-ren c3dl.FreeCamera klasea erabiliko da honetarako.
- **Solairua:** Solairu bati dagokion informazio guztia bilduko duen klasea (solairuaren 3D modeloa, gehigarriak, modeloaren dimentsioak etab.).
- **Gertaera zona:** Erabiltzaileak kamera mugitzen duen bitartean, zerbait gertatzea nahi den zonak definitu eta kudeatzeko klasea. Zona hauek lau koordenaturen bidez definituko dira beti.
- **3D objektua:** Solairuaz aparte aplikazioan kargatuko diren 3D objektuak kudeatzeko C3DL-k eskaintzen duen c3dl.Collada klasea erabiliko da.
- **Mapa:** Solairu baten maparen informazioa bildu eta kudeatzeko klasea. Maparen altuera eta zabalera eta mapa irudiaren helbidea kudeatu beharko da.
- **Eszena:** Aplikazioan kargatzen diren 3D modelo guztiak kudeatzeko klasea. C3DL-ren c3dl.Scene klasea erabiliko da.
- **Argiak:** Bisita birtualeko eszena argiztatu eta ikusi ahal izateko klasea. C3DL-k eskaintzen duen c3dl.PositionalLight klasea erabiliko da kasu honetarako.
- **Kontrolak:** Aplikazioaren kontrola eramango duen klasea. Klase hau kameraren kontrolaz, kolisio detekzioaz etab. arduratu beharko da.

Klase hauek beheko irudian agertzen den moduan erlazionatzen dira:



Irudia 27: Klase diagrama

Ikusten denez, lau klase bakarrik sortu beharko dira proiektuan: Solairua, Mapa, GertaeraZona eta Kontrolak. Gainontzekoak C3DL-k eskaintzen dituen klaseak izango dira.

Inplementatu beharreko klaseen ezaugarri eta funtzionalitate nagusiak aztertuko dira, hauen diseinua zehazteko. Dena dela, inplementatzerako orduan lagungarriak izango diren atributu eta metodoak ere inplementatuko dira.

Mapa

Klase honen bidez solairu bati dagokion maparen oinarrizko informazioa gordeko da, beranduago kolisio detekziorako erabilia izango dena. Honetarako ondoko atributuak erabiliko dira:

- **Altuera:** Mapa irudiaren altuera adierazten du (pixeletan).
- **Zabalera:** Mapa irudiaren zabalera adierazten du (pixeletan).
- **path:** Mapa irudiaren helbidea adierazten du.

Solairua klasearen barruan erabiltzeko pentsatuta dago, solairu baten mapa kargatzerako orduan datuak behin eta berriz sartzen ibili behar ez izateko. Hala ere, posible da klase honen instantziak sortzea, eman nahi zaion erabilpenaren arabera.

GertaeraZona

Klase hau eszenan area edo zonak definitzeko sortuko da. Erabiltzailea kamerarekin zona hauetan sartzen den aztertzea izango da honen helburu nagusia, gertaera honi dagokion erantzuna emanez. Kasu honetan gertaera zona hauek eskailera bidezko solairu aldaketa kontrolatuko dute; hau da, kamera eskailera batera hurbiltzen denean, zona batean sartzean, solairuz aldatuko da.

Zona hauek lau koordenaturen bidez definitzen dira:

- X minimo eta maximoa. Zabalera adierazteko.
- Y minimo eta maximoa. Sakonera adierazteko.

Gainera, helburua solairu aldaketa denez, bi atributu gehiago gehitzen zaizkio klaseari, zein solairutara eta nondik joaten den adierazteko.

Behin zona bat definitu dela, koordenatu pare bat zonaren barruan dagoen edo ez jakiteko metodo bat eskaintzen du: ***barruanDago(x,y)***. X parametroak zabalera adierazten du, eta Y parametroak sakonera.

Solairua

Klase honen bidez solairu baten informazio eta modelo guztiak gorde eta kudeatuko dira. Bere atributu garrantzitsuenak hauek izango dira:

- **Solairuaren modeloa:** Solairuaren modeloa gordeko du, *c3dl.Collada* motako objektu baten bitartez.
- **Gehigarrien modeloen zerrenda:** Solairuan gehigarriak izango diren objektuen zerrenda bat gordeko du. Objektu hauek *c3dl.Collada* objektuak

izango dira.

- **Mapa:** Gure *Mapa* klasearen instantzia bat gordeko da, dagokion maparen informazioarekin.
- **Gertaera zonen zerrenda:** Gure *GertaeraZona* klasearen instantzien zerrenda bat gordeko du, solairuari dagozkion gertaera zonak kudeatu ahal izateko.
- **X(zabalera) eta Y(sakonera) dimentsioak:** Bi atributu desberdin hauetan solairu modeloaren X eta Y dimentsioak gordeko dira. Bi balio hauek kolisio detekzioan erabiliko dira beranduago.
- **Solairuko ate handi eta txikien posizio zerrendak:** Solairuan kargatu beharko diren ate handien eta txikien posizioak gordetzeko bi zerrenda gordeko dira bi atributu desberdinetan.

Atributu bakoitzarendako ohikoak diren *set()* eta *get()* metodoetaz aparte, klasearen funtzionalitate nagusiak ondokoak dira:

solairuModeloaKargatu(fitxPath): COLLADA fitxategi baten helbidea pasata, bertan definitutako modeloa kargatzen du solairu modeloaren atributuan, honetan c3dl.Collada objektu baten instantzia sortuz.

gehigarriModeloaGehitu(Obj): *c3dl.Collada* motako objektu bat jasota, gehigarri modeloen zerrendan gordetzen du.

gertaeraZonaGehitu(gert): *GertaeraZona* motako objektu bat jasota, gertaera zonen zerrendan gordetzen du.

setModeloDimentsioak(xmin, xmax, ymin, ymax): Zabalera balio minimo eta maximo bat eta sakonera balio minimo eta maximo bat jasota, hauek definitzen duten zabalera eta sakonera dimentsioak kalkulatu eta X eta Y dimentsio atributuetan gordetzen ditu.

setZ(x): Metodoak Z atributuari x balioa edo objektua esleituko dio. Z klaseko edozein atributu izan daiteke.

getZ(): Metodoak Z atributua itzuliko du. Z klaseko edozein atributu izan daiteke.

Kontrolak

Aplikazioko gertaerak kudeatu eta erantzuna emateko erabiliko da klase hau. Momentu oro eszenaren kontrola mantenduko du, eta honekin batera kameraren eta kargatutako solairuaren kontrola.

Bere atributu nagusiak:

- **Eszena:** c3dl.Scene klasearen instantzia bat gordeko da, aplikazio hasieran definitu dena.
- **Uneko solairua:** Kargatutako solairua gordetzen duen Solairua klaseko instantzia.
- **Maparen canvas-a:** Web orrian mapa bistaratzeko erabiltzen den canvas etiketaren “id” atributuaren izena gordeko du.
- **gertaera funtzioa:** Kamera gertaera zona batean sartzen denean deitzea nahi den funtzioa gordetzen duen "atributua". Funtzio hau eraikitzailean pasa beharra zaio klaseari.

Atributu bakoitzarendako ohikoak diren *set()* eta *get()* metodoetaz aparte, klasearen funtzionalitate nagusiak ondokoak dira:

teklatuaKudeatu(): Eszenako kamera eskuratu eta teklatu bidez jasotako sarreren arabera biraketak eta translazioak eragingo zaizkio posizio berria kalkulatzeko. Behin posizio berria kalkulatu dela, kamerari esleitu aurretik posizio horretan kolisiorik gertatzen den aztertuko da. Kolisiorik ez badago kamerari posizioa esleituko zaio.

kolisioDetekzioa(pos): Kameraren posizioa jasota, uneko solairuaren dimentsioak eta mapa eskuratu eta kamerari mapan dagokion posizioa kalkulatuko da. Kamerari mapan dagokion pixela zuria bada, kolisioa gertatu ez dela adierazi beharko da. Zuria ez bada, kolisioa gertatu dela adieraziko da.

solairuaKargatu(S): Solairua motako objektu bat jasota, eszena garbitu eta solairuaren modeloa eszenan sartuko du, honekin batera kamera solairuaren hasiera puntura eramanez eta solairuaren mapa kargatuz.

mapaKargatu(zab,alt,path): Mapa baten zabalera, altuera eta helbidea jasota,

mapa web orriko canvas-ean kargatzen du, canvas-ari zabalera eta altuera berriak esleituz.

gertaerakKudeatu(): Eszenako kamera uneko solairuari dagokion gertaera zona batean sartu den aztertzen da. Hauetako baten barruan dagoela detektatzen bada, klaseko *gertaeraFun* atributuan adierazita dagoen funtzioari dei egiten zaio gertaerari erantzuna emateko.

7.2.3- Exekuzio fluxuaren diagrama

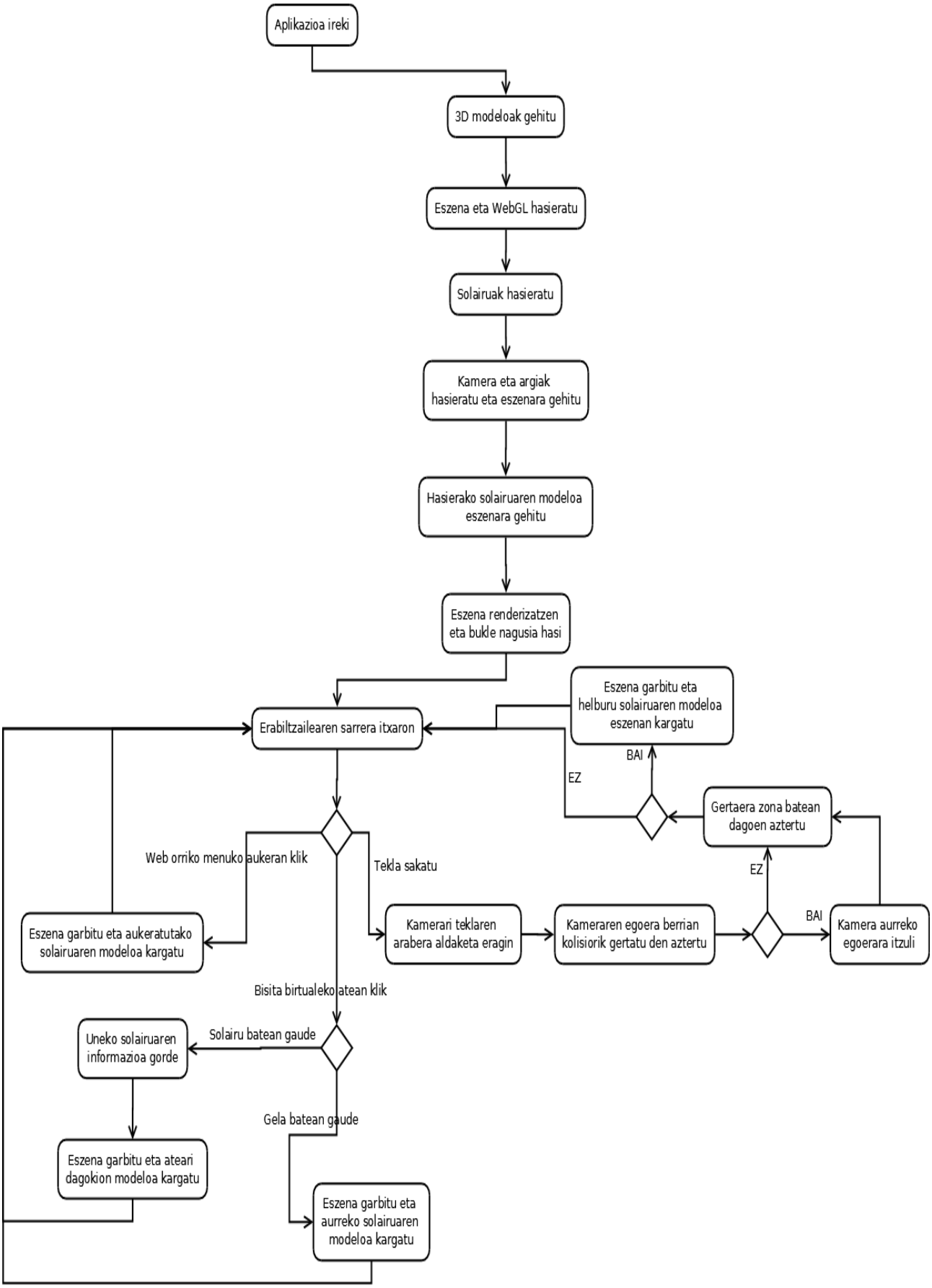
Atal honetan aplikazioaren oinarrizko exekuzio fluxua azalduko da. Beheko diagraman ikusi daiteke grafikoki zein den aplikazioaren exekuzio fluxua.

Aplikazioa erabiltzaileak web orria irekitzen duenean zuzenean martxan jartzen da. Behin erabiltzaileak web orria ireki duela, kontrola hasieran aplikazioari pasatzen zaio, modeloen karga eta aldagaien hasieraketak burutu ahal izateko.

Behin hasieraketak burutu direla, kontrola erabiltzaileari itzultzen zaio, eta aplikazioa gertaera baten zain gelditzen da. Gertaera hauek ondokoak izan daitezke:

- **Erabiltzaileak tekla bat (edo gehiago) sakatzen du:** Kamerari aldaketak eragiten zaizkio. Kolisio eta gertaera zonen kudeaketa burutzen da.
- **Erabiltzaileak web orriko menuan solairu bat aukeratzen du:** Eszena garbitu eta aukeratutako solairua aplikazioan kargatzen da.
- **Erabiltzaileak bisitako ate batean klik egiten du:** Eszena garbitu eta dagokion gela edo solairua kargatzen da.

Garrantzitsua da aplikazioak amaiera punturik ez daukala azpimarratzea. Erabiltzaileak web orria isten duenean amaituko da aplikazioa. Hau kontuan izatea garrantzitsua da, erabiltzaileak aplikazioa irekita uzten badu, erabili gabe, ordenagailuko baliabideak kontsumitzen jarraituko baitu.



Irudia 28: Aplikazioaren exekuzio fluxua