eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Informatika Ingeniaritzako Gradua

Konputazioa

Gradu Amaierako Lana

Errealitate Birtualerako Jolas Hezitzaile baten Garapena

Egilea

Mikel Berganza Muguruza

2020

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Informatika Ingeniaritzako Gradua

Konputazioa

Gradu Amaierako Lana

Errealitate Birtualerako Jolas Hezitzaile baten Garapena

Egilea

Mikel Berganza Muguruza

Zuzendaria

Borja Calvo Molinos

Laburpena

Hezkuntza gero eta informatizatuago dago, eta horrek, hezkuntzarako teknikak aldatzea edota berritzea dakar. Horregatik, gradu amaierako lan honetan, ordenagailuetarako zein errealitate birtualeko gailuetarako erabilgarria izango den umeentzako joko hezitzailea egitea izango da helburu nagusia.

Bideo-jokoaren helburua, umeek biderketak praktikatzeko modu alternatiboa izatea da. Jolasen bidez zeharka ikasten badute, modu entretenitu batean egingo dute, eta interes gehiago jarriko diote.

Sortuko den bideo-jokoa, Unity motorearen bidez sortuko da eta bi bertsio egingo dira. Lehena, ordenagailuko mahaigainerako izango da, eta bigarrena, Oculus Go kaskoekin bateragarria izateko.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

1. KAPITULUA

Sarrera

Kapitulu honetan eta proiektuarekin hasi baino lehen, hainbat kontzeptu azaldu behar dira. Lehenik eta behin, gamifikazioa zer den azalduko da eta errealitate birtualeak gamifikazioan dauzkan abantailak ikusiko dira. Horretaz aparte, gaur egun gamifikaziorako erabiltzen diren teknologiei buruz hitz egingo da. Azkenik, Unity eta Unreal Engine motoreak ikusiko dira, baita VRrako (ingelesezko *Virtual Reality* terminotik) dauden plataforma eta kasko ezberdinak ere, hauen arteko konparaketak eginez.

1.1 Gamifikazioa

Gamifikazioa edo Ludifikazioa [Hamari et al., 2014], joko eta aisialdietan erabiltzen diren teknika, elementu eta dinamika propioetaz baliatzean datza, olgetarako ez diren jardueretan (enpresa munduan, osasunean, publizitatean, hezkuntzan...) erabili ahal izateko.

Gamifikazioak hainbat helburu dauzka. Adibidez, motibazioa indartu, arazoak konpontzeko portaera sendotu, produktibitatea hobetu, helburuak lortu, ikasketa sustatu edota pertsona jakin batzuk ebaluatu.

Gamifikazioa, heziketa-sisteman, ikasleak ikaskuntza esanguratsuagoa izateko modua lantzen du [Hamari et al., 2016]. Beste helburu bat, ikaslea jokoen bidez borondatez ikasi nahi izatea da.

Gamifikazioa betidanik erabili izan da heziketan, adibidez, alfabetoa, biderkatzeko tau-

lak edota herriak ikasteko kantak erabiliz, xakea erabiliz estrategia militarra irakasteko, antzara-jokoa edota senet mahai-jokoak erabiliz, etab.

Gaur egun, aldiz, jendea txikitatik gailu elektronikoekin edo bideo-jokoekin egotera ohitu da. Gainera, eskola eta institutu askotan ikus daitezke ikasleentzako tableta (gailu elektronikoa) edota ordenagailu eramangarriez ornitutako gelak. Horregatik, ez litzateke arraroa izango ikasketa prozesuan bideo-jokoen mekanikak inplementatzea, hala nola, puntuazio sistemak edo dominak lortu ahal izatea, sailkapen taulak eta errendimendu-grafikoak[1] erabiltzea; edo hauen dinamikak inplementatzea, adibidez, ikaskideen artean lehiaketa osasuntsua sustatzea.

Merkatuan, sistema asko daude umeen ikasketa bultzatzeko. Adibide bat, umeek matematiketan hobetzeko *Matific*[2] web-orria izan daiteke. Bertan ikasleek daramaten aurrerapenaren segimendua jarraitzen da txosten eta estatistiken bidez.

Errealitate birtualean, aldiz, ez daude hainbeste sistema edo baliabide umeek ikasi dezaten, nahiko teknologia berria delako. Errealitate birtuala, hurrengo urteetan asko bultzatuko den teknologia da eta gero eta gehiago erabiltzen hasiko da gamifikaziorako.

1.2 Errealitate birtuala gamifikazioan

Errealitate birtualak bilakaera handia izan du azken urteotan eta etorkizunean gero eta gehiago ustiatuko den merkatua da [Mikołajczyk, 2019]. Horri esker, geroz eta baliabide gehiago daude gamifikazioa hezkuntzan sustatzeko. Errealitate birtuala egungo hezkuntzan erabiltzeak onura edo abantaila hauek ekarriko lituzke:

- **Esperientzia bisual paregabea da.** Frogatuta dago irudiak ikustean irakurtzen baino gehiago ikasten dela, eta errealitate birtuala, gaur egun dagoen teknologiarik aurreratuenetarikoa da arlo horretan.

- **Interesa sortzen du.** Ikasleek beti nahiagoko dute zerbait ikustea irakurtzea baino. Errealitate birtualari esker klase normal batean lor ezin daitekeen esperientzia lor daiteke, errealitate desberdinak arakatzea baimentzen baitu.

[1] Sailkapen tauletan ez bezala, errendimendu-grafikoek ez dute jokalarien errendimendua beste jokalariekin alderatzen, baizik eta, norberaren errendimendua denboran zehar ebaluatzen dute.

[2] *Matific* web-orria: https://www.matific.com/es/es/home/

- **Ikasleen parte-hartzea handitzen du.** Irakasleek normalean duten zailtasunetako bat ikasleen parte-hartzea izaten da. Errealitate birtualari esker, arazo hori ez litzateke gertatuko edozein ikaslek parte hartu nahiko duelako jolastera animatzen zaituen errealitate birtualeko esperientzia bat bizitzera.
- **Ikaskuntza dibertigarriagoa egiten du.** Bideo, bisualizazio eta jokoen bidez informazio berria ikastean ez da lan bezala ikusiko, ikasketa asko arintzen duelako, eta beraz, gehiago ikasteko gogoa piz dezake.
- **Erabateko murgilketa izatea eragiten du.** Errealitate birtuala beste errealitate bat erakusten duenez, ikasgelan zaudela ahaztarazten dizu. Murgilketa horrek kontzentrazio guztia errealitate berrian jartzea eragiten du, produktibitatea handituz.

Beraz, laburbilduz, esan daiteke gamifikazioa heziketa-sisteman, ikasleak ikaskuntza prozesura bultzatu eta erakartzea duela helburu. Beste moduren batean lortu ezin daitekeen esperientzia interesgarri eta erakargarri baten bidez [Kiryakova et al., 2014].

1.3 Gamifikaziorako teknologiak

Azken urteotan ordenagailurako dauden hezkuntza-joko gehienak, nabigatzaile jokoak dira. Joko hauen teknologiaren atzean, *plugin*-mota[3] asko daude, gehienak doakoak, hala nola, Javako makina birtuala, *Adobe Shockwave* edota *Adobe Flash Player*. Teknologia hauek erabiltzen dituzten jokoak garatzeko erreminta espezializatuak daude, adibidez, *Adobe Shockwave*n bideo-jokoak garatzeko *Adobe Director* aplikazioa.

Arlo honetako joko asko, *Flash* jokoak bezala ezagutzen dira, *Adobe Flash* aplikazio bidez garatu izan direlako. Hala ere, joko hauen bizi-denbora amaitzen ari da.

Adobe Shockwave Player aplikazioaren garapena eten egin zen 2019ko apirilaren 9an. *Adobe Flash Player* aplikazioa, 2020ko abenduaren 31n eten egingo du bere garapena ere. Adobek planak iragarri zituen *Flash*ari "denbora-bonba" bat gehitzeko, bizi-amaieratik haratago dauden instalazioak saihesteko, erabiltzaileek *Flash* desinstalazioa eska dezaten, eta *Flash* instalatzaileentzako deskarga-lotura guztiak ezabatzeko.

Bideo-joko hauetaz aparte, pluginik behar ez dituzten jokoak daude. Bideo-joko hauek, HTML5 markaketa-lengoaia eta JavaScript lengoaiaren bidez egiten dira.

[3] ***Plugin*** bat, programa informatiko baten softwareari funtzionalitate gehigarri bat edo ezaugarri berri bat eransten dion aplikazioa da.

Hauek dira hezkuntzarako (ingelesez dauden webguneak) famatuak diren hainbat webguneen adibideak:

Adobe Flash erabiltzen duten webguneak:

- https://www.starfall.com/
- https://www.funbrain.com/
- http://www.sheppardsoftware.com/
- https://www.coolmath4kids.com/
- https://pbskids.org/

JavaScript erabiltzen duten webguneak:

- https://kahoot.com/
- https://www.funbrain.com/
- https://kids.nationalgeographic.com/
- https://pbskids.org/

Adobe Flash Player aplikazioaren amaierarekin, nabigatzailean dauden bideo-joko hezigarri gehienak ezingo dira gehiago erabili.

Horrek, beste aplikazioen bidez garatu diren bideo-jokoen areagotzea ekarriko du.

1.3.1 Unity vs Unreal Engine

Gaur egun bideo-jokoak garatzeko erabilienak eta ezagunenak diren bideo-joko motorrak[4] Unity eta Unreal Engine dira [Foxman, 2019][Šmíd, 2017].

Proiektua hasi baino lehen, mahaigaineko bideo-jokoa egiteko motorea aukeratzea komeni da. Horretarako, Unity eta Unreal Engine motorrak aztertuko dira, bien indarguneak eta ahuleziak alderatuz azken erabakia hartzeko.

Unity eta Unreal Engine motoreen abantailak eta desabantailak aukeratzeko orduan, kontuan hartuko da nolabaiteko eragina izan behar dutela proiektuan.

Hurrengo hauek dira Unity eta Unreal Engine motoreen abantaila eta desabantaila nagusiak:

[4] **Bideo-joko motorra**, ingelesezko *Game Engine* terminotik, bideo-joko baten diseinua, sorkuntza eta funtzionamendua ahalbidetzen duten programazio-errutina batzuei egiten die erreferentzia.

Unity abantailak:

- Doan $100K baino gutxiago irabazten duten hasiberrientzat.
- C# lengoaia, garapen-prozesua errazagoa eta azkarragoa.
- Erraza hasiberrientzat.
- Dokumentazio asko.
- Ona 2D eta 3D jokoetarako.
- Denda handia dohako *Asset*[5] askorekin.
- VR eta ARrako[6] SDK[8] eskuragarritasuna.
- VR garatzeko plataforma erabiliena.
- Multiplataforma jokoak egiteko egokia.

Unreal Engine abantailak:

- Gustiz dohakoa.
- Beste motoreak baino errendimendu handiagoa.
- Gama altuko grafikoak.
- Eskala handiko jokoetarako aproposa.
- 3D jokoetarako gehienbat.
- Denda handia dohako asset askorekin.
- Erreminta-multzo zabala eta editore sendoa.
- Unity baino errendimendu hobeagoa VR eta mugikor jokoetan.
- Multiplataforma jokoak egiteko egokia.

Unity desabantailak:

- Ez da egokia AAA[7]joko edo proiektu handietarako.
- Erabiltzaile-interfaze aldaketa asko.

Unreal Engine desabantailak:

- C++ lengoaia, C# lengoaia baino zailagoa.
- Hasiberrientzako zaila.
- Motore handiegia joku oso txikietarako.
- Ez da egokia banakako proiektuetarako.
- Behar espezifikoak asetzeko doikuntza gehiegi behar ditu.

[5]***Asset*** bat, jolas edo proiektu batean erabil daitekeen edozein elementuren adierazpena da, hala nola, irudi bat, audio-fitxategi bat edota 3D eredu bat.

[6]**Errealitate areagotua**, ingelesezko *Augmented Reality* terminotik, mundu fisikoaren zuzeneko edo zeharkako ikuspegi bat da zeinen elementuak areagotuak agertzen diren ordenagailuz sortutako estimuluen bitartez.

[7]AAA terminoa, tamaina ertain edo handiagoko enpresa batek ekoitzi eta banatutako bideo-jokoetarako erabiltzen den sailkapen informala da.

Beraz, bi motoreen alde positibo zein negatiboak ikusita, eta ikasleak Unity motorearekin aurretiko esperientzia pixka bat bazuela jakinda; proiektua garatzeko Unity motorea erabiltzea erabaki da.

1.4 VR plataformak

Errealitate birtuala itxura errealeko eszena edo objektuen ingurune bat da, teknologia informatikoaren bidez sortua, eta erabiltzaileari bertan sartuta egotearen sentsazioa sortzen diona. Erabiltzaileak errealitate birtualeko kasko edo betaurrekoak izeneko gailu baten bidez ikusten du ingurune hori.

Proiektua VRrako garatuko denez, gaur egun erabiltzen diren hainbat VR gailu begiratu eta konparatuko dira. VR gailu bakoitzak konfigurazio, kontrol eta botoi desberdinak baititu.

Errealitate birtualeko kasko edo betaurrekoak bi kategoria nagusitan banatzen dira:

- **Askeak edo independenteak.** Kaskoan integratutako errealitate birtualeko esperientziak emateko behar diren osagai guztiak dituzten gailuak dira. Eremu honetako VR plataformarik ezagunenak hurrengo hauek dira:
  - **Oculus Mobile SDK[8].** Oculus VRk garatua, bere kasko independenteetarako eta Samsung Gear VR kaskoetarako.
  - **Google Daydream.** Googleren Android sistema eragilean integratutako errealitate birtualeko plataforma.
- **Kabledunak edo menpekoak.** Errealitate birtualeko esperientzia bat emateko, beste gailu baten (PC bat edo bideo-jokoen kontsola bat) bistaratze-gailu gisa jarduten duten kaskoak dira. Hauek dira VR plataforma nagusiak:
  - **SteamVR.** Valveko Steam zerbitzuko kidea, HTC Vive, Oculus Rift edo beste hainbat kaskorekin bateragarria.

[8] **Softwarea garatzeko tresneria**, ingelesezko *Software Development Kit* terminotik, sistema jakin baterako softwarea garatzeko tresna-multzoa da.

  - **Oculus PC SDK.** Oculus Rift eta Oculus Rift Srentzako plataforma.
  - **Windows Mixed Reality.** Microsoft Corporationek garatua Windows 10 ordenagailuetarako.
  - **PlayStation VR.** Sony Computer Entertainmentek garatu du, PlayStation 4 bideo-jokoen kontsolarekin erabiltzeko.
  - **Open Source Virtual Reality.** Kode irekiko software proiektu bat da, saltzaile guztien kasko eta joko-kontrolagailuak Razer eta Sensicsek garatutako edozein jokorekin erabili ahal izatea lortu nahi duena.

1.4.1 VR kaskoen konparaketa

Gaur egun VR kaskorik ezagunenak edota erabilienak hurrengo hauek dira, hauen abantaila eta desabantailekin:

- **Oculus Quest.** Askoren ustez, VR kaskorik onena. 6DoF[9] dauzka eta bi kontrolagailurekin batera dator. Oculus Quest kalitate/prezio aukerarik onenetarikoa da.
  + Gustiz independentea
  + Kontrol jarraipen zehatza
  + UI azkarra
  + Aplikazio-liburutegi handia
  − Garesti samarra, 449 €
  − Oculus Storeren edukira mugatua
- **Oculus Rift S.** Mugimendu libreari dagokionez, Oculus Quest baino mugatuagoa da, baina esperientzia zehatzagoak eta dinamikoagoak lortzeko gai da. Oculus Quest kaskoen prezio bera daukate eta 6DoF dauzka baita ere.
  + Ez ditu kanpoko jarraipen estaziorik behar, barrutik kanporako jarraipena baitu
  + Joko liburutegi handia

[9] **DoF** edo **Askatasun-mailak**, ingelesezko *Degrees of Freedom* terminotik dator. 3DoF bada, hiru ardatzetako errotazioa onartzen da (zabu, kopadura/eskora, trabeskatze). 6DoF bada, aldiz, hiru ardatzetako errotazioaz aparte, hiru ardatzetako translazioa onartzen du. 1.1 irudian ikus daiteke VR kaskoek izan ditzazketen askatasun-mailak.

  + Bereizmen eta eguneratze-tasa bikainak
  − Oculus Quest kaskoen prezio bera
  − Kableduna
  − Ordenagailu indartsua behar du

- **Playstation VR.** Kontsoletarako VR entzungailurik onena da, eta VR jokoen liburutegi onenetakoa du. Kaskoak 6DoF dauzka.

  + Nahiko merkea, 299,99 €
  + Ordenagailuen kalitatetik gertu dagoen errendimendua
  + Joko lista ona
  + Independentea izan daiteke
  − Mugimendu-kontrolatzailearen segimendu irregularra
  − Beharrezko osagaiak falta zaizkio
  − Segur aski, PS5 atera eta gutxira ordezkatuko dute, beste kaskoekin konparatuz atzeratua geratzen ari baita.

- **Oculus Go.** Ez da Oculus Quest bezain indartsua, baina VRan sartzeko modu azkar, erraz eta eskuragarria da. VRan hasieratzeko aukera ona da. Kaskoak 3DoF dauzka.

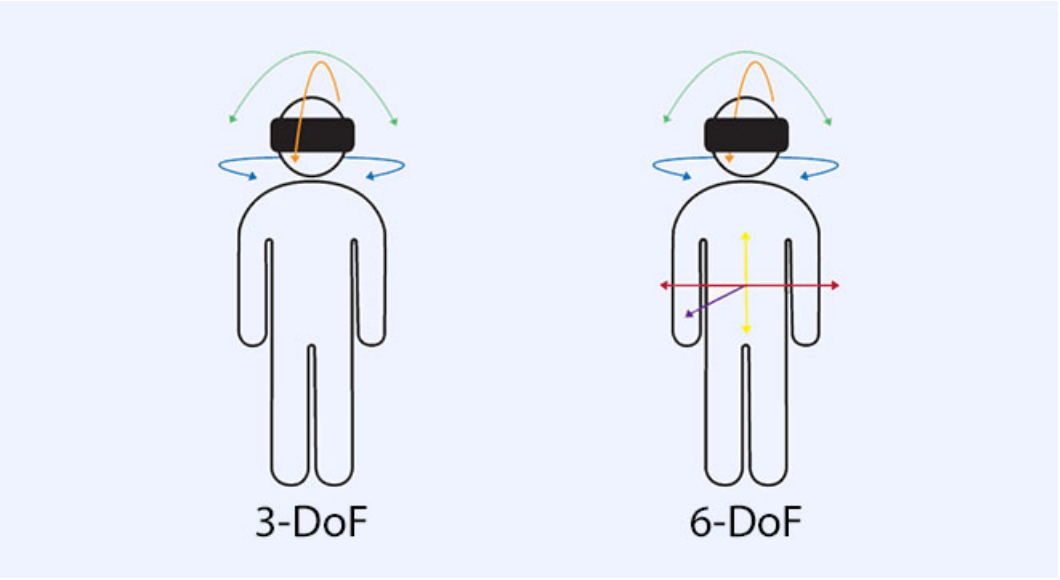
  + Gustiz independentea
  + Oso merkea beste kaskoekin konparatuta, 150 €
  + Aplikazio eta jokoen liburutegi handia
  − Kontrolatzaile bakarra
  − Ez dago jarraipen espazialik
  − 2020. urtearen amaieratik aurrera, ez ditu funtzionaltasun edo aplikazio berririk izango, akatsen zuzenketak eta segurtasun-adabakiak 2022rarte jarraituko dute

- **HTC Vive.** Ordenagailuarekin batera erabiltzeko VR kaskorik onenetarikoa. HTCk VRtik telefono mugikorra atzitzeko gaitasuna ematen du, baita mundu erreala ikusteko aukera ematen du, beharrezkoa bada. HTC kaskoak 6DoF dauzka.

  + Latentzia gutxiko grafiko leunduak
  + Joko liburutegi handia

  - \+ Ukipen-kontrolatzaileak erraz erabiltzen dira eta oso moldagarriak dira
  - – Garestia, 549 €
  - – Ez dauka txertatutako audioa
  - – Leku eta hargune elektriko behar ditu erabiltzeko

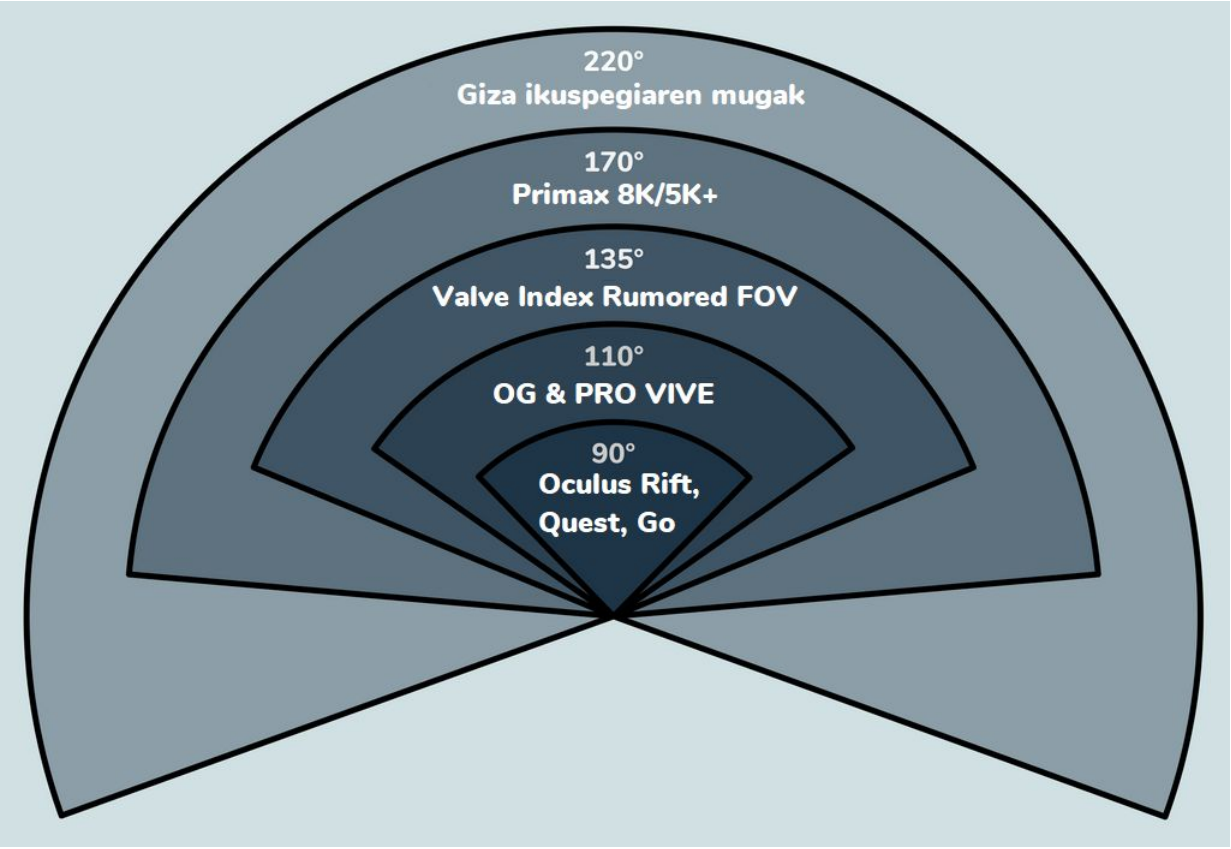
- **Valve Index.** Valve Index kaskoak, ordenagailurako dauden kaskorik onenak dira. HTC Viverekiko hobekuntza nabarmena da, eta etorkizunean VR kasko lehenetsiak izango dira. Kaskoak 6DoF dauzka.
  - \+ Beste kasko konparagarriak baino FOV[10] zabalagoa
  - \+ Audioaren kalitatea bikaina
  - \+ Etorkizunerako hedapen-aukerak
  - \+ Hatz-koskor motako kontrolagailuak, hatz bakoitzaren mugimendua jarraitu dezaketenak
  - – Oso astuna
  - – Zerrendako garestiena alde handiarekin, 1079 €
  - – Gama altuko GPUa behar du



1.1 Irudia: VR kaskoek duten DoF edo askatasun-mailen aukerak.

1.2 irudian ikus daiteke VR kasko ezberdinen ikus-eremua.

[10] **FOV** edo **ikus-eremua**, ingelesezko *Field Of View* terminotik, une jakin batean beha daitekeen mundu-hedadura da.



1.2 Irudia: VR kasko ezberdinen ikus-eremua.

1.4.2 VR kasko aukeraketa

Proiektuan mahaigaineko jokoaz aparte VRrako jokoa garatuko denez, VR kasko batzuk aukeratu behar izan dira; VR kasko ezberdinek kontrolagailu desberdinak dituztenez jokua programatzeko orduan eragina izango dutelako.

Kontuan izatekoa da proiektuaren helburua, hezkuntza-jokoa garatzea dela. Beraz, aukeratzerakoan, hezkuntzaren ikuspegitik begiratu behar da.

Eskolek VR kaskoak erostean, klase osoak betetzeko kopuru handiak erosi behar badituzte, VR kasko merkeak izatea komeniko litzateke. Puntu horretatik ikusita, Oculus Go gaur egun gehien saltzen diren kaskoak dira. Nahiko merkeak izateaz aparte, murgiltze handiko esperientziak eskaini ditzaketelako.

Beraz, proiektua Oculus Go kaskoekin bateragarria izateko garatuko da. Nahiz eta 2020. urtearen amaieratik aurrera, ez ditu funtzionaltasun edo aplikazio berririk izango, aukera ona jarraitzen izaten du. Gainera, kaskoen produkzioa amaituko balitz, aldaketa txiki batzuekin proiektuaren garapena, Oculus Quest kaskoekin ere bateragarria izango litzateke.

2. KAPITULUA

Proiektuaren Helburuen Dokumentua

Kapitulu honetan proiektua aurrera eramateko zehaztu diren helburuak azalduko dira. Horretaz gain, proiektua garatzeko sortu den plangintza ikusiko da, jarraituko den lan-metodologia, informazio- eta komunikazio-sistemak ikusiko dira, proiekturako sortu diren arriskuen kudeaketa aztertuko da, proiektuan sortu diren interesatuak ikusiko dira eta azkenik, proiektuaren jarraipena eta kontrola ikusiko da.

2.1 Proiektuaren deskribapena eta helburuak

Proiektu honen helburua, ordenagailuetarako zein errealitate birtualeko gailuetarako erabilgarria izango den umeentzako hezkuntza-jokoa egitea izango da. Horretarako, Unity motorra erabiliko da. Bideo-jokoaren sorkuntzaren helburu nagusia, umeek hezkuntzarako entretenigarria izan daitekeen eta disfrutatu dezaketen metodo alternatibo bat sortzea da, ikasketa bultzatzeko.

2.2 Proiektuaren plangintza

Atal honetan, proiektua garatzeko beharrezkoak izango diren atazak azalduko dira.

2.2.1 LDE diagrama

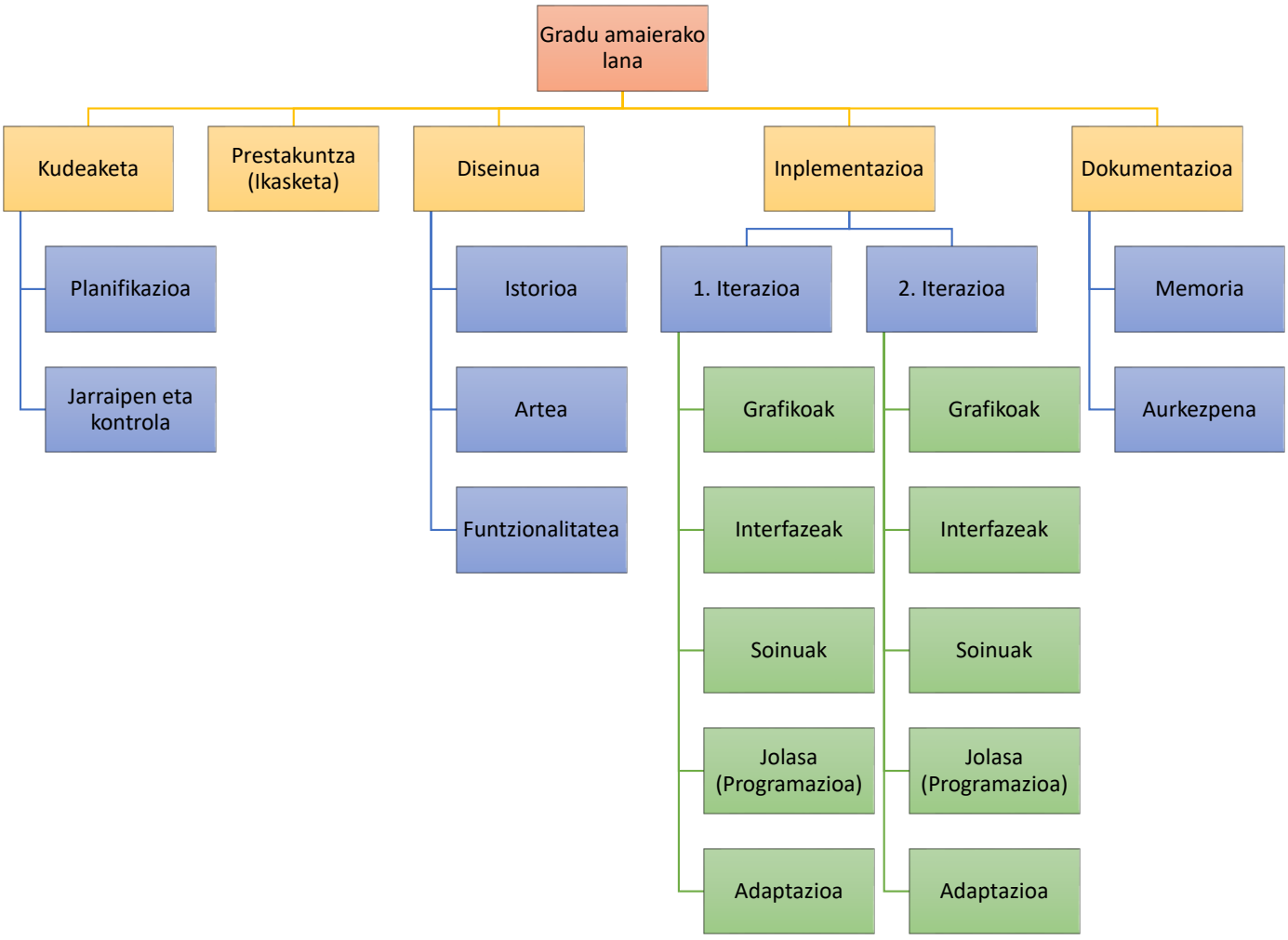
Proiektuan zehar burutu behar izango den lan guztia deskonposatzeko, Lanaren Deskonposaketa Egitura (LDE) diagrama erabili da. 2.1 irudian ikus daiteke nola deskonposatu den lana.

2.2.2 Lan-paketeak

2.2 irudian ikus daiteke ataza bakoitza betetzeko esleitu den denbora estimazioa. Nahiz eta erabili diren denborak estimazioak baino ez diren, errealitatetik asko ez urruntzeko ahaleginak egingo dira. Hala ere, proiektuan zehar aldaketak egon daitezke. Aldaketa horiek 2.8 atalean zehaztuko dira.

Proiektua bost lan-pakete nagusitan banatu da.

1. **LP1 Lan-paketea: Kudeaketa (9 ordu).** Kudeaketaren lan-paketea bi atazetan banatzen da:
    - **A1.1 ataza: Planifikazioa (2 ordu).** Ataza honetan proiektua garatzeko jarraitu beharreko planifikazioa sortuko da. Planifikazioan, proiektua definituko duten lan-pakete eta atazak sortuko dira, atazen denbora banaketa egingo da, emangarriak eta mugarriak definituko dira, eta jarraituko den lan-metodologia eta arriskuen kudeaketa adieraziko da.
    - **A1.2 ataza: Jarraipen eta kontrola (7 ordu).** Ataza honetan proiektua garatzen den ahala, helburuak betetzen direla egiaztatuko da, proiektuaren garapen zuzena bermatzeko. Horretarako, jarraipen bilerak eginez. Ataza hau proiektuaren bizitza-ziklo osoan zehar kontrolatuko da.
2. **LP2 Lan-paketea: Prestakuntza (22 ordu).** Lan-pakete honetan proiektua garatzeko beharrezkoak izango diren ezagutzen ikasketa egingo da.
3. **LP3 Lan-paketea: Diseinua (43 ordu).** Diseinua lan-paketearen helburua bideojokoaren diseinuaren dokumentu bat (*Game Design Document*) sortzea da. Dokumentu honetan estilo artistikoa, jokoaren dinamikak, plataformak, soinu-efektuak eta abar zehazten dira. Lan-pakete hau, hiru atazetan banatzen da:
    - **A3.1 ataza: Istorioa (4 ordu).** Ataza honetan jokoaren istorioa sortuko da. Istorioaren barnean jokoaren pertsonaiak, eszenarioa nolakoa izango den, eta horrelako hasierako kontzeptuak laburki zehaztuko dira.



2.1 Irudia: Proiektuaren LDE diagrama.

- **A3.2 ataza: Artea (17 ordu).** Artearen atazan bideo-jokoaren arte kontzeptuala sortuko da. Arte kontzeptuala jokoaren errepresentazio- edo ilustrazio-modu bat da, non helburu nagusia jokoaren diseinu grafikoaren zirriborro edo ilustrazio bisual bat sortzea den.
- **A3.3 ataza: Funtzionalitatea (22 ordu).** Ataza honetan jokoaren funtzionalitateak zehaztuko dira eta funtzionalitate hauek bi iterazioetan nola banatuko diren zehaztuko da.

4. **LP4 Lan-paketea: Inplementazioa (193 ordu).** Inplementazioaren lan-paketea bi iterazioetan banatzen da (2.3 atalean aurkitzen da bi atal hauen azalpena), eta iterazio bakoitza bost atazetan banatzen da:

   - **A4.1.1/A4.1.2 ataza: Grafikoak (33 ordu).** Ataza honetan bideo-jokoa erabiliko dituen grafikoak inplementatuko dira, hala nola, paisaia edo eszena, pertsonaia eta bestelako elementuak.
   - **A4.2.1/A4.2.2 ataza: Interfazeak (30 ordu).** Interfazeak atazan jokoa izango dituen interfaze guztien sorketa egingo da.
   - **A4.3.1/A4.3.2 ataza: Soinuak (20 ordu).** Ataza honetan bideo-jokoaren audio eta soinu-efektu guztiak inplementatuko dira.
   - **A4.4.1/A4.4.2 ataza: Jolasa (90 ordu).** Jolasa atazan joko osoaren funtzionalitateen inplementazioa egingo da.
   - **A4.5.1/A4.5.2 ataza: Adaptazioa (20 ordu).** Ataza honetan jokoaren VRrako egokitzapena egingo da.

5. **LP5 Lan-paketea: Dokumentazioa (60 ordu).** Dokumentazioaren lan-paketea bi azpiatazetan banatzen da:

   - **A5.1 ataza: Memoria (50 ordu).** Memoriaren atazan proiektuari buruzko informazio guztia jasotzen duen dokumentua garatuko da.
   - **A5.2 ataza: Aurkezpena (10 ordu).** Ataza honetan proiektuaren defentsarako erabiliko den aurkezpena garatuko da.

Atazak		Asteak (Ekainak 21 entrega, Ekainak 29-Uztailak 10 aurkezpenak, 16 aste guztira)																Guztira (ordu)
		2020/03/09	2020/03/16	2020/03/23	2020/03/30	2020/04/06	2020/04/13	2020/04/20	2020/04/27	2020/05/04	2020/05/11	2020/05/18	2020/05/25	2020/06/01	2020/06/08	2020/06/15 (E)	2020/06/22 (A)	
Kudeaketa	Planifikazioa	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
	Jarraipen eta kontrola	0,5	0,5	0,5	0,5	0,5	0	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0	7
Prestakuntza (Ikasketa)		3	3	3	3	3	2	1	2	1	1	0	0	0	0	0	0	22
Diseinua	Istorioa	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
	Artea	3	2	3	3	3	3	0	0	0	0	0	0	0	0	0	0	17
	Funtzionalitatea	4	5	3	3	2	5	0	0	0	0	0	0	0	0	0	0	22
Inplementazioa	1. Iterazioa Grafikoak	0	2	2	2	4	5	5	0	0	0	0	0	0	0	0	0	20
	1. Iterazioa Interfazeak	0	0	0	0	0	5	2	3	10	0	0	0	0	0	0	0	20
	1. Iterazioa Soinuak	0	0	0	0	0	0	5	5	0	0	0	0	0	0	0	0	10
	1. Iterazioa Jolasa	0	5	5	10	10	15	10	0	0	0	0	0	0	0	0	0	55
	1. Iterazioa Adaptazioa	0	0	0	0	0	0	2	6	4	0	0	0	0	0	0	0	12
	2. Iterazioa Grafikoak	0	0	0	0	0	0	0	2	5	3	3	0	0	0	0	0	13
	2. Iterazioa Interfazeak	0	0	0	0	0	0	0	0	0	0	3	2	5	0	0	0	10
	2. Iterazioa Soinuak	0	0	0	0	0	0	0	0	0	0	2	5	3	0	0	0	10
	2. Iterazioa Jolasa	0	0	0	0	0	0	0	5	5	10	10	5	0	0	0	0	35
	2. Iterazioa Adaptazioa	0	0	0	0	0	0	0	0	0	0	2	4	2	0	0	0	8
Dokumentazioa	Memoria	0	0	0	0	0	0	0	0	0	0	0	10	15	15	10	0	50
	Aurkezpena	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	10
Guztira (ordu)		12,5	19,5	16,5	21,5	22,5	35	25,5	23,5	25,5	14,5	20,5	26,5	25,5	15,5	10,5	10	325

2.2 Irudia: Gantt diagrama itxurako atazen denbora banaketa.

2.2.3 Emangarriak

Hauek dira proiektuan zehar sortuko diren emangarriak:

- *Game Design Documenta*
- Jokoaren exekutagarria eta jokoa exekutatzeko behar diren fitxategi guztiak
- Memoria
- Aurkezpena egiteko power pointa
- Posterra

2.2.4 Denbora planifikazioa

Azpiatal honetan, proiektua garatzeko sortu diren planifikazioaren atazen (Gantt diagramaren bidez) eta emangarrien mugarrien (mugarrien taularen bidez) denborak agertzen dira.

2.2.4.1 Gantt diagrama

2.2 irudian proiektuaren Gantt diagrama itxurako atazen denbora taula ikus daiteke. Bertan, ataza bakoitza burutzeko eskainiko zaion denbora estimazioa azaltzen da.

2.2.4.2 Mugarriak

Azpiatal honetan, proiektuan zehar sortu diren emangarrien entregatzeko datak ikusten dira 2.1 taulan.

Emangarriak	Entregatzeko data
Game Design Document	2020/04/26-2020/04/30
Jokoa	2020/06/21
Memoria	2020/06/21
Posterra	2020/06/21
Aurkezpena	2020/06/29-2020/07/10

2.1 Taula: Mugarrien taula.

2.3 Lan Metodologia

Proiektua garatzeko lan gehiena etxetik egingo den arren, jokoaren ordenagailuko bertsioaren egokitzapena errealitate birtualera fakultateko *Innovae* gelan egingo da. *Innovae* gela errealitate birtuala probatzeko hurrengo baliabideak ditu: Oculus Go kaskoak, HTC 2V Pro kaskoak, HoloLens kasko pare bat eta Noitom Hi5 VR Glove eskularruak.

Jokoaren garapena aurrera eramateko, gutxienez bi iterazio desberdinetan banatuko da prozesu guztia:

- **Lehen iterazioa.** Lehen iterazio hau garrantzitsuena eta denbora gehien beharko duena izango da. Lehen iterazioan jokoa behar dituen oinarrizko elementu guztiak egingo dira.

  Iterazio honen helburua jokoaren lehenengo bertsio funtzionala egitea izango da, jokoa helburu minimo batzuk betetzen dituela egiaztatzeko balioko duena. Beraz, jokoaren oinarrizko funtzionalitate guztia inplementatuta egon beharko da iterazio honen amaierarako.

- **Bigarren iterazioa.** Bigarren iterazioa, lehen iterazioa amaitzean hasiko da. Lehen iterazioaren gehigarria izango da, hau da, jokoa amaitzeko oinarrizkoak ez diren gauzak egiteko iterazioa izango da.

Iterazio bakoitzaren azalpen zehatzagoa *Game Design Document*eko 3.3 atalean aurkitzen da.

2.4 Informazio-sistema

Proiektua garatzeko erabiliko den material guztia digitalki gordeko da. Informazioa gordetzeko, hainbat iturri erabiliko dira:

- **Ordenagailu pertsonala.** Ordenagailu pertsonalean, bideo-jokoa sortzeko Unityk sortzen dituen fitxategi guztiak gordeko dira. Bideo-jokoaren bertsio desberdinak gordetzen joango dira, errore bat egongo balitz aurreko bertsio bat izateko. Horretaz aparte, dokumentazioaren bertsio guztiak gordetzen joango dira, badaezpada hodeiko informazioa galduko balitz. Gainera, dokumentuan erabiliko diren argazkien, eskemen, eta abarren kopiak gordeko dira.

- **Google Drive.** Google Drive zerbitzua ordenagailu pertsonalean gorde den material guztiaren azkenengo bertsioaren kopia bat izateko erabili izan da. Ordenagailuan arazoren bat izanez gero beste kopia bat izateko.

- **Overleaf.** Overleaf, dokumentu zientifikoak idazteko, editatzeko eta argitaratzeko erabiltzen den LaTeXonline editorea da. Bertan, dokumentazioaren informazio guztia gordeko da. Dokumentuaren azkenengo bertsioa izateaz aparte, aurreko bertsio desberdinak gorde izan dira ere, badaezpada erroreren bat balego.

2.5 Komunikazio-sistema

Proiektuaren zuzendariarekin komunikatzeko hurrengo kanalak erabili dira:

- **Bilerak.** Proiektuaren hasierako asteetan, bilerak astero **aurrez aurre** egin izan ziren Borja Calvo zuzendariaren bulegoan, horrela adostu izan zelako.

  Informatikako fakultatea itxi zutenean COVID-19 gaixotasunaren ondorioz, etxean eman izan ziren asteetan, Mikel Berganzaren eta Borja Calvoren arteko komunikazioa eta bilerak, **posta elektronikoz** eta **Whereby** aplikazioaren bidez egin izan ziren.

- **Bestelako komunikazioak.** Zalantza txikiak argitzeko **posta elektronikoa** erabili izan zen. Baita bileraren baten ordu edota egun aldaketa egin behar bazen ere.

2.6 Arriskuen kudeaketa

Atal honetan, proiektuan zehar aurkitu daitezkeen arriskuak zerrendatuko dira eta arrisku horiek saihesteko edota arrisku horiei aurre egiteko zer egingo den azalduko da.

2.6.1 Arriskuak

- **Informazio-galera.** Proiektua garatzeko erabili den materiala edo informazioa galtzeko edota online zerbitzuen akats teknikoren batengatik informazioa galdu ahal izateko arriskua.

- ***Innovae* gelaren ixketa.** Edozein arrazoi ezezagunengatik gelaren itxiera gertatu ahal izatea.
- **Bideo-jokoa amaitzeko denborarik ez egotea.** Bideo-jokoa amaitzeko denbora gutxiegi izatea, edo plangintzako denbora baino gehiago behar izatea atazak betetzeko.
- **Jokoaren sorketarako grafikoak ez aurkitzea.** Bideo-jokoaren sorketarako grafikoak ez aurkitzea.

2.6.2 Prebentzioa

- **Informazio-galera.** Erabiliko den informazio guztiaren hainbat segurtasun-kopiak egitea eta leku desberdinetan gordetzea, hala nola, ordenagailu pertsonalean eta hodeian. Gainera, segurtasun-kopia hauek eguneratzen joatea, bertsio desberdinak izateko. Bertsioren bat galdu ezkero, aurreko bat izateko.
- ***Innovae* gelaren ixketa.** Arrisku honetarako ez legoke prebentziorik. Bideo-jokoaren VRrako egokitzapena ezingo litzateke egin ixketa luzeegia bada, horregatik, mahaigaineko bertsioan jarriko litzateke atentzio guztia.
- **Bideo-jokoa amaitzeko denborarik ez egotea.** Bideo-jokoaren sorkuntza bi iteraziotan banatu. Bideo-jokoa amaitzeko denborarik ez balego, lehen iterazio funtzionala entregatuko da.
- **Jokoaren sorketarako grafikoak ez aurkitzea.** Proiektu honen helburua ez denez parte grafikoa oinarritzen, baizik eta bideo-jokoaren funtzionalitatean; grafiko sinpleak sortzea izango litzateke prebentzio-neurria.

2.7 Interesatuak

Proiektu honetan idenfitikatu diren interesatuak hurrengoak dira:

- Mikel Berganza.
- Borja Calvo.
- Defentsako irakasleak.

- Jokoaren xede publikoa.

2.8 Jarraipen eta kontrola

Atal honetan, proiektuan zehar bete egin diren arriskuak eta sortu izan diren problemak azalduko dira. Horretaz aparte, proiektua garatzeko behar izan diren atazen benetako denborak ikusiko dira, desbiderapen taula baten bidez.

Proiektua ekainean amaitzeko planifikatu egin zen 2.2 irudian ikus daitekeen moduan. Planifikazioan aurreikusitako denborak ezin bete ahal izatearen arrazoirik nagusiena, ikasturte honetan jasan behar izan den COVID-19 pandemia izan da.

Pandemiari esker, proiektuan lan egiteko denborak murriztu egin dira proiektuan definitutako puntu edo helburu batzuk ezin bukatzea eraginez. Gainera, probabilitate gutxien zeukan arriskua bete egin izan da: VRrako bertsioa garatzeko *Innovae* gelaren ixketa. Honi esker, ezin izan da proiektu hasieran planifikatu zen bezala lan egin eta horren ondorioz, proiektuaren entrega atzeratzea erabaki zen zorte pixka batekin VRrako egokitzapena irailaren hasieran egiteko.

2.2 taulan, atazen desbiderapen taula ikus daiteke. Bertan, planifikazioaren hasieran aurreikusitako denborak eta benetako denborak konparatzen dira. Inplementazioa lan-paketearen bi iterazioen atazen denborak batu egin dira, taula handiegia ez egitearren.

Taulan ikus daitekeen moduan, oro har, nahiko estimazio onak egin dira ataza bakoitzerako. Egon den desbiderapen nabarmen bakarra memoriaren atazarekin izan da. Hasieran aurreikusitako denbora baino askoz ordu gehiago behar izan dira, ikaslearen idazteko trebezia faltak (latexekin) eta dokumentazioa hasiera batean aurreikusitakoa baino nahiko luzeagoa geratzeak ordu gehiago behar izatea eragin du.

Atazak	Aurreikusitako denbora (ordutan)	Benetako denbora (ordutan)	Desbiderapena
Planifikazioa	2	15	+13
Jarraipen eta kontrola	7	5	-2
Prestakuntza (Ikasketa)	22	35	+13
Istorioa	4	2	-2
Artea	17	10	-7
Funtzionalitatea	22	15	-7
Grafikoak	35	20	-15
Interfazeak	30	40	+10
Soinuak	20	10	-10
Jolasa (Programazioa)	90	80	-10
Adaptazioa	20	25	+5
Memoria	50	90	+40
Aurkezpena	10	10	0
Guztira	325	357	+32

2.2 Taula: Atazen desbiderapen taula.

3. KAPITULUA

Game Design Document

Kapitulu honetan, bideo-jokoa diseinatzeko dokumentua (ingelesezko *Game Design Document* terminotik) ikusiko da. *Game Design Document*a, bideo-jokoen diseinuko dokumentua da, eta bideo-jokoen buruzko eduki oso deskribatzailea aurkitzen da.

Esan beharra dago bideo-jokoen industrian ez daudela estandar asko, baina estandar horietako bat *Game Design Document*a dela.

Kasu honetan, jokoaren deskribapena, hainbat zehaztapen (generoa, xede publikoa, jokoaren dinamika, etab.) eta iterazioen deskribapenak ikusiko dira.

3.1 Jokoaren deskribapen orokorra

Jokoa 3Dko hezkuntza-joko bat izango da, basoan giroturik egongo dena. Jokalariak defektuzko pertsonaia bat lehen pertsonan maneiatuko du eta jokoaren helburua, matematiketako galderak zuzen erantzutea izango da. Galderak zenbakien arteko biderketak izango dira.

Galderak ateratzen joango dira erabiltzailearen pantailan eta erantzuteko denbora bat izango du, zailtasun mailaren araberakoa. Galdera bat erantzuteko, zenbaki desberdinak dituzten hainbat itu edo objektibo egongo dira, eta erantzun bat aukeratu nahi bada, tiro egin beharko zaio.

Tiro egiteko hainbat arma desberdin egongo dira, eta bakoitza erabiltzeko zailtasunaren

araberako puntuak lortuko dira (arma bat beste bat baino zailagoa bada erabiltzeko puntu gehiago lortuko dira ondo erantzuterakoan).

Galdera ondo erantzuten bada, puntuak lortuko dira behar izan den denboraren arabera (azkarrago erantzuteak puntu gehiago emango ditu). Galdera bakoitzaren denbora agortzen denean edota gaizki erantzuten denean, bizitza bat galduko da. Bizitza kopuru zehatz bat galtzean jokoa amaitu egingo da.

3.2 Jokoaren zehaztapenak

3.2.1 Istorioa

Duela gutxi sortu duten robot baten adimen artifiziala entrenatzeko, sortzaileek robota hezkuntza tiro-eremu batera eraman dute basora, eta bertan botata utzi dute maila guztiak gainditu arte. Robotak matematika basikoa ikasteaz aparte, arma desberdinak erabiltzen trebatu beharko da, arma bakoitzaren jaurtigaiaren erorketara edota ezarri behar zaion indarrera moldatuz.

3.2.2 Plataforma

Jokoa Windows edota Linux sistema eragileak dituzten ordenagailuetarako garatuko da. Baita Oculus Go errealitate birtualeko gailuetarako.

3.2.3 Generoa

Jokoa hezkuntza-joko bat izango da. Umeek matematikak modu erosoago eta dibertigarriago batean ikas dezaten.

3.2.4 Xede publikoa

Printzipioz, jokoa lehen hezkuntzako hirugarren eta bostgarren maila artean (7-10 urte) dauden ume txikientzako egingo da. Hala ere, edozein adineko jendea jolastu ahalko du, zailtasuna gora egiten duen ahala edonorentzako erronka bihur daitekeelako.

3.2.5 Pertsonaia eta elementuak

Pertsonaia nagusia robot bat da. Robotak hainbat arma izango ditu aukeratzeko bere armategian eta ezkerrera eta eskuinera biratu ahalko du tiro egiteko.

Tiro-eremuan, hainbat itu egongo dira. Hasieran ez dira mugituko, baina gero eta maila altuagoa izanda, orduan eta norabide gehiagotan mugituko dira, jotzeko zailtasuna handituz.

3.2.6 Maila diseinua

Jokoak hainbat zailtasun maila izango ditu. Galdera bakoitza erantzuteko denbora, zailtasun mailaren araberakoa izango da. Horretaz gain, aterako diren biderketen zailtasuna ere aukeratu den mailaren araberakoa izango da. Geroz eta maila altuagoa aukeratuz, orduan eta denbora gutxiago edukiko du jokalariak galdera erantzuteko eta galderak gero eta zailagoak izango dira (biderkatuko diren zenbaki posibleak handiagoak izango dira).

Ituak mailaren araberako mugimenduak izango dituzte. Maila errazenean, ituak ez dira mugituko. Hurrengo mailan, alboetara mugituko dira. Hurrengoan, alboetara mugitzeaz aparte, gora eta behera ere mugituko da. Horrela jarraituko du azkenengo mailararte, non, ituek mugimendu aleatorioak izango dituzten.

Horretaz aparte, gero eta galdera gehiago erantzuten direla, orduan eta zailagoa egingo da maila. Galdera bakoitza erantzuteko denbora murrizten joango da eta ituak mugikorrak badira, orduan eta habiadura handiagoa izango dute.

3.2.7 Jokoaren dinamika

Jokoa hasten denean jokalariak berarentzat egokiena den zailtasun maila aukeratuko du. Jokalariak bizitza kopuru finko batekin hasiko du partida. Galdera bat oker erantzuten den bakoitzean edo galdera erantzuteko denbora agortzen denean, bizitza bat galduko du.

Bizitza guztiak amaitzean jokalariak galdu egiten du eta jokoa amaitu egingo da. Jokoa amaitzen den unean, aukera egongo da beste partida bat hasteko, beste zailtasun maila aukeratzeko edota jokutik ateratzeko.

Galdera bat ondo erantzutean zenbait puntu lortuko dira, denboraren edota armaren araberakoa izango direnak. Jokoaren helburua ahalik eta puntu gehien lortzea izango da.

Horretaz aparte, partida edozein momentutan gelditu ahalko da. Partida gelditzean, aukera egongo da partidarekin jarraitzeko, partidatik ateratzeko, edota jokutik ateratzeko.

VRrako bertsioan jokalariaren kamera estatikoa izango da.

3.1 taulan pertsonaiaren kontrolak ikusten dira. Bertan, pertsonaiaren mugimendu desberdin bakoitza nola egingo den adierazten da, plataforma desberdinetarako.

Kontrolak	Ordenagailurako	Errealitate birtualerako
Ezkerrera biratu	Sagua ezkerrera mugitu	Burua ezkerrera biratu
Eskuinera biratu	Sagua eskuinera mugitu	Burua eskuinera biratu
Tiro egin	Saguaren ezkerreko botoia sakatu	Kontroladore katua sakatu
Jokoa pausatu	P tekla sakatu	Kontroladore *touchpad*a sakatu

3.1 Taula: Kontrolen taula.

3.2.8 Diseinu grafikoa

Eszenan erabiliko den koloreen paletarako, baso batek dituen koloreetatik gehiegi ez hurruntzea saiatuko da, hau da, ez dira oso deigarriak diren koloreak erabiliko.

Efektu bisualak erabiliko dira. Jaurtigaia beste elementu baten kontra talka egitean leherketa txikia sortuko da.

Jokoaren eszena, baso bat izango da, mendiren batekin atzealdean. Horretaz aparte, pantailan partidaren aurrerapenari buruzko informazioa agertuko da (puntuazioa, zenbat bizitza gelditzen diren, galdera erantzuteko denbora...).

Jarraian dagoen 3.1 irudian jokoaren diseinu grafikoaren 2D zirriborroa dago.

3.2.9 Musika eta audioa

Jokoak baso ingurumenerako aproposa izango den atzealdeko musika izango du. Arma bakoitzerako soinu efektu desberdinak egongo dira, baita jaurtigaia zerbaiten kontra estanda egiterakoan ere.

3.1 Irudia: Diseinu Grafikoaren 2D bozetoa.

3.2.10 Garapen tresnak

Jokoaren programazio guztia Unity 3Dn egingo da. Jokoak izango dituen hainbat elementu grafikoak egiteko, Blender aplikazioa erabiliko da. Soinuak editatzeko, Audacity aplikazioa erabiliko da.

Hauek izango dira edukiak sortzeko erabiliko diren hainbat iturri:

- **Soinu efektuak.**
  https://freesound.org/
  http://99sounds.org/
  https://www.noiseforfun.com/
  http://soundbible.com/
- **Musika.**
  https://www.youtube.com/
  https://www.jamendo.com/start
  https://freemusicarchive.org/
  https://soundcloud.com/
  http://soundimage.org/
- **3D grafikoak.**
  https://free3d.com/
  https://www.turbosquid.com/
  https://3dwarehouse.sketchup.com/

- **3D animazioak.**
  https://www.mixamo.com/#/

3.3 Iterazioen deskribapena

Jokoa bi iterazio desberdinetan banatuko da.

- **Lehen iterazioa.** Lehen iteraziorako inplementatuko diren gauzak hurrengo hauek izango dira:
  - Jokoak 3 zailtasun maila izango ditu. Gero eta zailtasun maila handiagoa, orduan eta denbora gutxiago egongo da erantzuteko, eta, galderak zailagoak izango dira.
  - Itu edo objektiboak ez dira mugituko. Berdin dio zein den jokalariak aukeratzen duen zailtasun maila, ituak beti geldirik egongo dira.
  - Ituen artean erantzun zuzen bakarra egongo da eta hiru gaizki. Erantzun okerrak erantzun zuzenetatik gertu dauden balioak izango dira.
  - Arma bakarra egongo da. Laser pistola bat izango da. Laserrak, noski, ez du erorketarik izango.
  - Galdera bat ondo erantzutean puntuak gehituko dira. Txarto erantzuten bada, aldiz, bizitza kenduko da.
  - Maila aukeratzeko menu nagusia egongo da.
- **Bigarren iterazioa.** Bigarren iteraziorako inplementatuko diren gauzak hurrengo hauek izango dira:
  - Gero eta galdera gehiago erantzuten direla, orduan eta zailagoa egingo da maila. Galdera bakoitza erantzuteko denbora murrizten joango da.
  - Lehen mailako ituak geldirik egongo dira beti. Bigarren eta hirugarren mailetan, ituak mugitu egingo dira. Bigarren mailan, mugimenduak bertikalki edo horizontalki izango dira. Hirugarren mailan, aldiz, mugimenduaren norabidea ausazkoa izango da.
  - Bigarren eta hirugarren mailetan, gero eta galdera gehiago erantzuten direla, orduan eta azkarrago mugituko dira ituak, beti ere limite batekin.

- Zailtasun maila bakoitzerako arma desberdin bat egongo da. Beraz, hiru arma egongo dira, gero eta zailagoak erabiltzeko. Lehen mailan, jaurtigai erorketa gabe, bigarren mailan, jaurtigai erorketarekin eta azkenik, hirugarren mailan, jaurtigai erorketa izateaz aparte, indar barra bat egongo da (gero eta gehiago sakatu tiro egiteko botoia, orduan eta hurrunago joango da jaurtigaia).
- Menu nagusi osatua egongo da, zailtasun mailaren araberako puntuazio goreneko partidak gordetzeko, hizkuntza aldatzeko edota soinuaren bolumena kontrolatzeko.
- Bideo-jokoa VRrako egokituko da, bideo-jokoaren bertsio normalak dauzkan funtzionalitate guztiak VRrako bertsiorako egokituko dira.

4. KAPITULUA

Inplementazioa

Atal honetan proiektua garatzeko jarraitu diren pausoak azalduko dira. Hala nola, zein GameObject sortu diren bideo-jokoan, GameObject bakoitzaren funtzionalitate eta osagaien azalpena, *script* bakoitzaren azalpena edota Unity motorrean egin diren konfigurazioak. Unity motorraren informazio orokorra A eranskinean (A.1) kontsulta daiteke.

Kapituluaren hasieran jokoaren diseinu orokorra aurkituko da, bideo-jokoa nola dagoen antolatuta ikusteko. Hurrengo puntuetan jokoaren eszena nagusia eta menu nagusiaren eszena osatzen duten GameObjectak azalduko dira. Behin eszenako GameObject guztiak azaldu diren, bideo-jokoa egiteko sortu diren script guztiak azalduko dira. Ondoren, Unity motorrean egindako konfigurazio aldaketak ikusiko dira eta azkenik, bideo-jokoa VRra adaptatzeko egin diren aldaketak azalduko dira.

4.1 Jokoaren diseinu orokorra

Aplikazioaren diseinu orokorrean programazioarekin lotuta dauden GameObject eta scriptak nola dauden antolatuta azalduko da sarrera moduan. Bideo-jokoaren sistematizazioa ulertzeko hauen azalpen zehatzagoak eman baino lehen.

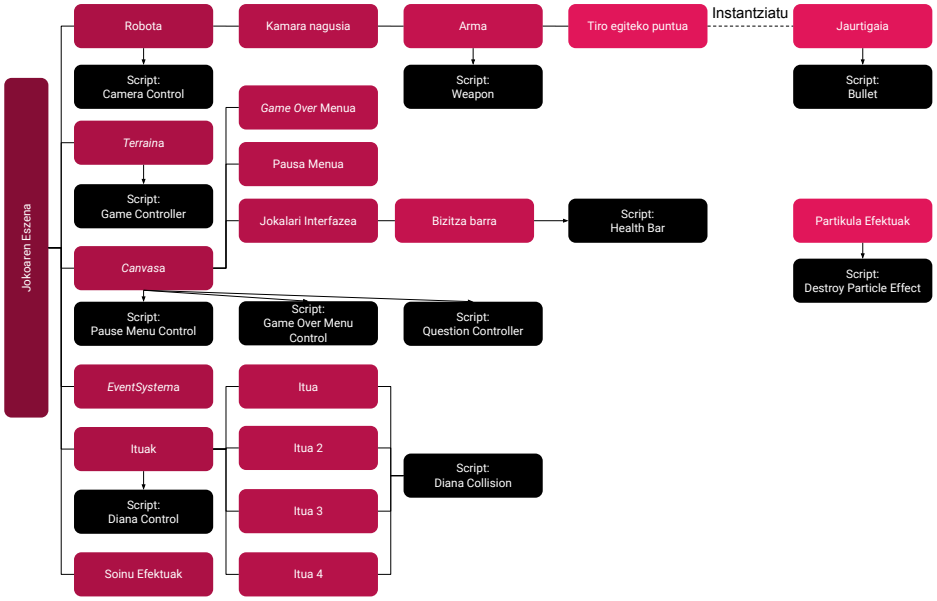
Kontuan izan behar da azalduko den diseinua, mahaigaineko bertsioaren diseinua dela eta VRrako egin izan diren egokitzapenak kapituloaren amaieran ikusiko direla.

Lehenik eta behin, eszenen antolamendua ikusiko da. Eszenek jokoaren objektuak barneratzen dituzte. Eszenak menu nagusia, banakako mailak, etab. sortzeko erabil daitezke.

Eszena bakoitza pantaila edo maila ezberdin bat izango balitz bezala ikus daiteke. Eszena bakoitzean, inguruneak, oztopoak, apangarriak, etab. jarriko dira, jolasa zatika diseinatzeko eta eraikitzeko.

Bideo-joko honek bi eszena nagusi dauzka: menu nagusiaren eszena eta jokatuko den mailaren eszena.

4.1 irudian jokoaren eszena nagusiak (jokatuko den mailaren eszena) dauzkan GameObject garrantzitsuenen antolamendua ikus daiteke, baita GameObjectek lotuta dauzkaten scriptak ere. 4.1 irudian, antolamendu bera ikusten da, baina kasu honetan menu nagusiaren eszenarako.

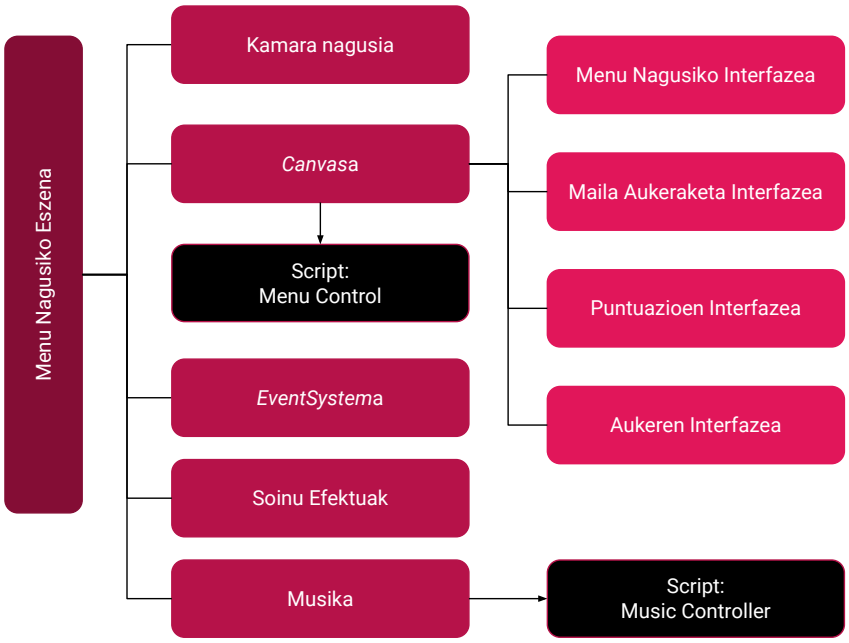


4.1 Irudia: Jokoaren eszena nagusiaren antolamendua.

Hurrengo ataletan bi eszenak dauzkaten GameObjectak azalduko dira. Lehenik eta behin, jokatuko den mailaren eszenarekin hasiko dira azalpenak.

4.2 Robota

Robota GameObjecta, jokalariak maneiatuko duen pertsonaia izango da. Jokalariak bideojokoaren eszena ikusi ahal izateko robota **kamera** batez hornitu da, lehen pertsonako pers-



4.2 Irudia: Menu nagusiaren eszenaren antolamendua.

pektiba izateko. Horretaz gain, ituei tiro egin ahal izateko, robota **arma** batekin hornitu da.

Robotak hurrengo osagaiak dauzka:

- ***Transform* osagaia.** Robota bideo-jokoan ez dela mugituko aukeratu denez; kasu honetan, bakarrik robotaren rotazio aldagaia (Y ardatzarekiko errotazioa hain zuzen ere, pertsonaiaren biraketa sortzeko) eraldatuko da Transform osagaitik. Honelako transformazio aldaketak script bidez egiten dira normalean. Script horren azalpena geroago ikusiko da, 4.7.1 azpiatalean.

  Transform osagaiari buruzko azalpenak A.1.3 puntuan aurki daitezke.

- ***Camera Control* scripta.** Jokalariak, pertsonaia lehen pertsonan jokatzen ari delaren sentsazioa emateko, kameraren mugimendua kontrolatu behar da. Funtzio hori, *CameraControl.cs* scriptaren bidez betetzen da. Script honen azalpen zehatzagoa 4.7.1 azpiatalean aurki daiteke.

Robota Unityra inportatzean daukan zati guztien egituraz aparte, kamera nagusia dauka seme bezala objektu hierarkian. Kamera nagusiaren seme bezala, arma dago eta honen atal guztien egituraz aparte, tiro egiteko transformazio-puntua dauka.

Egitura hau mugimendua errazteko erabili da. Arma kamera nagusiaren seme bezala ezarriz, kamera mugitzen edo biratzen den bakoitzean, arma kamerarekin batera mugitu edo biratuko da.

Robotaren hierarkia hurrengo listaren bidez eskematikoki ikus daiteke:

- Robota
  - Robota osatzen duten zati guztien egitura
  - Kamera nagusia
    - Arma
      - Arma osatzen duten zati guztien egitura
      - Tiro egiteko puntua

4.2.1 Kamera

Kamerak jokalariari mundua atzematen eta erakusten dioten gailuak dira. Esan bezala, eszenaren kamera nagusia lehen pertsona simulatzeko erabili izan da. Lehen pertsona simulatzeko kamera bideo-jokoetan erabiltzen den ikuspegia da, eta mundua pertsonaia protagonistaren ikuspegitik ikusten da. Lehen pertsonako ikuspegia, ikuspegi ohikoena da tiro egiteko bideo-jokoetan.

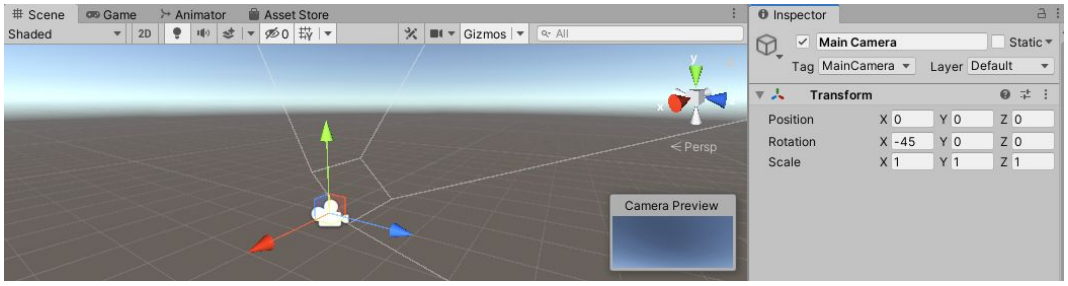
Lehen pertsonako ikuspegiak badu abantaila bat: errealismo handiagoa eta presentzia-sentsazio handiagoa ematen du bideo-jokoan; honek dakarren desabantaila, aurrerantz bakarrik ikus daitekeela da, eta beraz, jokalariak biratu egin behar du inguruan duena ikusteko. Bideo-jokoaren arabera, denbora gehiago edo gutxiago har dezake horrek.

Kasu honetan, bideo-jokoaren helburua ituak jotzea da galderak erantzuteko, beraz, bakarrik begiratu behar da ituei. Horregatik, eta errelismo handiagoa lortzen denez mahaigainerako zein VRrako bideo-jokorako, kamera lehen pertsona ikuspegian jartzea aukeratu da.

Unityk dakarren defektuzko kamera nagusiak, transformazio osagaiaz aparte bi osagai dauzka:

- **Transformazio osagaia.** Robotaren Y ardatzarekiko biraketa egiteaz aparte, lehen pertsonaren ikuspegia simulatzeko, gora eta beherako kamera mugimendua inplementatu behar da. Alboetara begiratzea, robota biratuz egiten da, hori baita edozein bideo-jokotan erabiltzen den teknika (tiro egiteko bideo-jokoetan ez du zentzurik bakarrik burua mugitzea alboetara begiratzean, errealistagoa da gorputz osoa mugitzea). Aldiz, gora eta behera begiratzeko, kameraren X ardatzarekiko biraketa aldatu egin behar da.

  4.3 irudian ikus daiteke nola kamera X ardatzaren gainean negatiboki biratzen denean, kamera gorantz begiratzen duela. Irudiaren *Camera preview* atalean ere ikus daiteke kamera gora begira dagoela bakarrik zerua ikus daitekeelako.



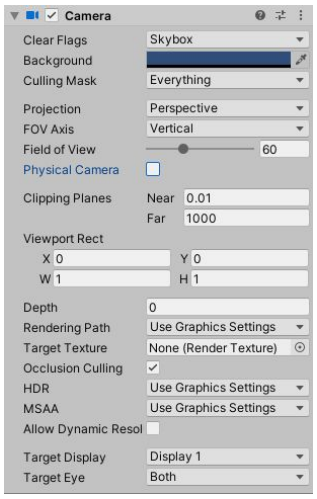
4.3 Irudia: Kameraren X ardatzarekiko biraketa.

- ***Camera* osagaia.** Camera osagaia, bideo-jokoan erabiliko den kamera nagusia pertsonalizatzeko erabiltzen den osagaia da. Kamera sortzean automatikoki ere osagaia sortzen da barnean, osagai hau beharrezkoa baita kameraren funtzionamendurako. Osagai honek dauzkan parametrorik garrantzitsuenak proiektu honetarako; proiekzio-mota, ikus-eremua (ingelesezko *Field of View* terminotik) eta *Clipping planes* parametroak izan dira.

  4.4 irudian ikus daiteke Camera osagaiak dauzkan parametroen zerrenda osoa.

  Kamerak bi proiekzio-mota dituzte: ortografikoa eta perspektibakoa. Perspektiba proiekzioak hiru dimentsioko objektu bat bi dimentsioko plano batean zuzen irudikatzeko balio du eta 3D jokoetan normalean erabiltzen den kamera mota da. Ortografikoa aldiz, 2D jokoetan erabiltzen da gehiago, proiekzio honen bidez ez delako sakonera sentsazioa sortzen. Horregatik, perspektiba proiekzioa aukeratu da, espazio-sakoneraren sentzazioa simulatzeko.

  Ikus-eremua[10], Unityko kamera zekarren defektuzko balioarekin utzi da (60º ardatz bertikalarekiko, hau da, kameraren beheko eta goiko puntuen arteko angelua).



4.4 Irudia: *Camera* osagaiak dauzkan parametroak.

Kameraren ikus-eremua gradutan neurtzen da eta Unityren kasuan, bakarrik ardatz baten (bertikal edo horizontal) araberako graduak ezarri daitezke.

Clipping planes aldagaiaren bidez, kameratik irudikapen edo errenderizazioa hasten eta gelditzen den planoen distantziak adierazten dira. Hau da, kameraren ikuspegia non hasi eta amaitzen den zehazten dute. Gertuko errenderizazio-mozketaren planoko distantziarako, Unityk ahalbidetzen duen minimoa erabili izan da (0.01 unitate). Unityko kamerak zekarren defektuzko balioa utzita (0.3 unitate), ez litzateke pertsonaiaren arma ikusiko. Urruneko plano distantzia, defektuzko balioarekin utzi egin da (1000 unitate).

Unityk unitate eskala lehenetsia metrotan dago, hau da, eszenaren 1 unitate = 1 metro. Bideo-jokoaren mahagaineko bertsiorako erabiltzen den unitate eskala ez du horrenbesteko garrantzirik, baina VRrako bertsiorako bai, distantziak eta tamainak errealagoak sentitzen direlako.

- ***Audio Listener* osagaia.** Audio Listener osagaia, mikrofonoaren antzeko gailu gisa funtzionatzen du. Eszenan dagoen edozein audio-iturri jasotzen du *input* bezala eta ordenagailuko bozgorailuen bidez soinuak erreproduzitzen ditu. Normalean, Audio Listener osagaia, kamera nagusian edo jokalariak maneiatuko duen pertsonaian jartzen da. Audio Listener osagaiak ez dauzka parametrorik.

4.2.2 Arma eta jaurtigaiak

Bideo-jokoaren helburu nagusia galderak erantzutea da. Horretarako, robota arma batekin hornitu da.

Armak bi osagaiez osatuta dago:

- ***Transform* osagaia.** Armaren kasuan, transformazio osagaia uneoro aldatzen dago. Transformazio osagaiko rotazio aldagaia da aldatzen dena hain zuzen, armaren funtzioa, uneoro pantailaren erdira begira egotea baita. Arma, aurrean dagoen objektuaren distantziaren araberako rotazioa izango du. Hau da, objektu bat oso gertu badago, armak ezkerrerako biraketa handiagoa izango du (Y ardatzarekiko errotazioa). Arma uneoro erdira apuntatzen egoteko *Raycast*a (A.1.3 atalean azalduta) erabili izan da.

  Honi buruzko azalpen zehatzagoa, 4.7.2 azpiatalean aurkitzen da.

- ***Weapon* scripta.** Helburu nagusia galderak erantzutea da, horretarako, tiro egiteko ahalmena sortu behar izan da. Hori guztia kontrolatzen duen scripta, *Weapon.cs* scripta da. *Weapon.cs script*ak bi funtzio garrantzitsu kontrolatzen ditu: lehenik eta behin, armak norabide egokian apuntatzeaz arduratzen da (uneoro pantaila erdira apuntatzea kontrolatzen du), eta bestetik, jokalariaren tiro egiteko baimena kontrolatzeaz arduratzen da (uneoro tiro ez egiteko).

  Honi buruzko azalpen osoa 4.7.2 azpiatalean aurkitzen da.

Jokalariak tiro egin ahal izateko, armaz aparte, jaurtigaiak behar dira.

Jaurtigaiak egiteko, Unityk daukan GameObject menuko esfera erabili izan da. Behin jaurtigaiaren GameObecta sortu dela, eta nahi adina aldiz tiro egiteko, GameObject honen klonak edo kopiak baino ez dira egin behar. Tiro egiten den bakoitzean jaurtigaiaren kopia sortuko da (*Weapon.cs script*ean azalduta), horrela objektu baten espazioa baino ez da okupatzen proiektuko karpetan.

Jaurtigaiek hurrengo osagaiak dauzkate:

- ***Transform* osagaia.** Betiko defektuzko osagaia. Kasu honetan, jaurtigaiek lerro zuzenean bidaiatu behar dute espazioan zehar (GameObjectaren transformazio osagaiko posizioa uneoro aldatzen joan behar da). Horretarako, rigidbodyek daukaten funtzio bat erabili izan da.

4.7.3 azpiatalean azalduta dago zehazki nola egiten den.

- ***Sphere (Mesh Filter)* osagaia.** Mesh Filter edo sare-iragazki osagaia, esfera GameObject bat sortzean defektuz sortzen den osagaia da. Mesh Filter osagaia, Mesh renderer osagaiarekin batera funtzionatzen du. Sare-iragazki osagaiak, sare bat hartzen du proiektuko direktorioan dauden asset[5]etatik (esfera sortzean, Unityk dakarren sare esferikoa pasatzen dio parametro bezala automatikoki), eta sare hori, sare-errenderizatzaileari pasatzen dio, honek pantailan errenderizatzeko.

  Beraz, osagai honen funtzionalitatea, laburki, errenderizatuko den sarea erreferentziatzea edo aipatzea da.

- ***Mesh renderer* osagaia.** Mesh renderer edo sare-errenderizatzailea, mesh filter edo sare-iragazkiaren geometria hartzen du eta GameObjectaren Transform osagaiak zehaztutako posizioan errenderizatzen du.

- ***Sphere collider* osagaia.** Sphere collider (colliderra A.1.3 azpiatalean azaltzen da) osagaia, mesh filter eta mesh renderer osagaiekin batera, esfera GameObjecta sortzean defektuz sortzen den osagaia da. Izenak dioen bezala esfera formako colliderra da, eta osagai honetan ez bada aldaketarik egiten, defektuz GameObjectaren tamaina berekoa izango da; kasu honetan egokiena delako.

- ***Rigidbody* osagaia.** Jaurtigaietan, rigidbody osagaia jaurtigaia arma apuntatzen duen norabidean bultzatzeko (rigidbody osagaia duten GameObjectak fisikaren menpean daudenez, hainbat funtzio aurki daitezke GameObject hauek maneiatzeko) erabiliko da. Baita talkak gertatu diren kontrolatzeko, collider osagaiarekin batera.

  Horretaz aparte, jaurtigaiek ez dutela grabitatearen indarra jasoko aukeratu da. Honen bidez, jaurtigaiek indar bat jasotzean infiniturantz zuzenduko dira beste GameObject batekin talka egin ezean.

- ***Bullet* scripta.** *Bullet.cs* scripta, jaurtigaien kontrol osoaz arduratuko da, hau da, jaurtigaien mugimendua eta talka detekzioa kontrolatuko da script honen bidez.

  4.7.3 azpiatalean aurki daiteke scriptaren azalpen osoa.

4.3 Jokoaren eszena nagusia

Eszena nagusia diseinatzerakoan, *Low Poly*[1] basolde bat irudikatzen saiatu da. Bideo-jokoa VRrako ere garatuko denez, diseinua low poly motakoa izatea aukeratu da, gehienbat konputazionalki astuna ez izateko.

Basoalde eszenarioa irudikatzeko, *Terrain*a erabili da oinarri moduan. Zerua, mendikateak eta bestelako eszenario objektuak irudikatzeko, low poly motako *prefab*[2] assetak[5] erabili izan dira.

4.3.1 *Terrain*a

*Terrain*a eszenako ikuspegia edo paisaia da. Unityk Terrain ezaugarrien multzo integratua du, eta, horri esker, paisaiak erantsi edota editatu daitezke bideo-jokoaren eszena leihotik. Hasieran, Terrain GameObject bat sortzean, plano handi bat (1000x1000 unitateko planoa) eransten zaio eszenari, hortik, Terrain-ikuskari (ingelesezko *Terrain's Inspector* terminotik) leihoa erabil daiteke paisaia xehatuagoa sortzeko.

Terraina sortzeko, GameObect menu zabalgarria erabili behar da, **Game Object > 3D Object > Terrain** bidea aukeratuz.

Terraina hurrengo osagaiak dauzka (transform osagaia ez da azalduko, ez baita ezertarako erabiltzen kasu honetan):

- ***Terrain* osagaia.** Terrain osagaian, Terraina editatzeko behar diren erreminta guztiak daude.

  4.5 irudian ikus daitekeenez, Terrain osagaiak bost erreminta (Terrain osagaiaren behekaldean dauden bost irudidun laukiak) dauzka barnean. Bost erreminta horiek hurrengoak dira:

  – **Aldameneko Terrain lauzak sortu.** Sortuta dagoen uneko Terraina handitzeko erabiltzen da erreminta hau. Uneko Terrain lauzaren goi, behe, ezker

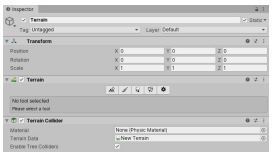
[1] **Low Poly** 3D ordenagailuko grafikoetako poligono-sare bat da, non, izenak dioen bezala, poligono kopuru nahiko txikia dituena. Sare bateko poligonoen kopurua faktore garrantzitsua da errendimenduarentzat, baina itxura txarra eman diezaieke lortzen diren grafikoei.

[2] ***Prefab*** edo **aurrefabrikatua**, ingelesezko *prefabricated* hitzetik Unityn erabiltzen den terminoa da, eszenan objektu baten instantzia berriak sortzeko txantiloi gisa jokatzen du. Prefab asset batean egindako edizio guztiak berak sortutako instantzia guztietan islatuko dira berehala, baina ere baliogabetu daitezke instantzia bakoitzaren osagaiak eta doikuntzak.

edo eskuinaldean Terrain lauzak gehitzeko erabiltzen da (ezin dira diagonalki gehitu).

- **Terraina zizelkatu eta margotu.** Terrainari forma emateko erabiltzen den erreminta da. Erreminta honek, sei funtzio dauzka: Terraina altxatu edo beheratu, zuloak egin, testura margotu, altuera finkatu, altuera leundu eta terraina estanpatu.
- **Zuhaitzak gehitu.** Zuhaitzak gehitzeko erabiltzen da erreminta hau, baita zuhaitzen kokapena eta ezaugarriak aldatzeko; adibidez, zuhaitzen altuera, zabalera, kokapen dentsitatea, rotazioa, etab.
- **Xehetasunak gehitu.** Terrainean xehetasunak gehitzeko erabiltzen da, hala nola belarra, loreak eta arrokak.
- **Hautatutako Terrainaren konfigurazio orokorra aldatu.** Terrainaren konfigurazioa aldatzeko erabiltzen da, hala nola, terrainaren argitasuna aldatzeko, terrainaren testuren edo sarearen bereizmena aldatzeko, zuhaitzen eta xehetasunenen marrazketa distantzia, etab.

  Kasu honetan, Terrainaren sarearen bereizmena gutxitu egin da errendimendua hobetzeko.



4.5 Irudia: *Terrain*aren osagaiak.

- ***Terrain collider* osagaia.** Terrain collider osagaiak, talka egiteko azalera bat ezartzen du atxikituta daukan (Terrain Data parametroan ezartzen den Terraina) Terrain objektuaren irudi berarekin. Honetaz aparte, Terrainari fisikako materiala (adibidez, lurra irristakorra, elastikoa... izan dadin) ezarri dakioke. Azkenik, zuhaitzen colliderrak aktibatu daitezke osagai honen bidez, Terrainean landatu diren (zuhaitzak gehitu erremintaren bidez sortu diren zuhaitzei soilik eragiten dio) zuhaitz guztiek collider bat (landatu den zuhaitz aurrefabrikatuak duen colliderra hain zuzen) izan dezaten.
- ***Game Controller* scripta.** *GameController.cs* scripta, bideo-jokoan garrantzitsuenak edo oinarrizkoak diren gauzetaz arduratzen da. Adibidez, bizitzak, denbora, puntuazioa, noiz amaitu den jokoa... kontrolatzen da script honen bidez.

Script honen azalpen osoa 4.7.4 azpiatalean aurkitzen da.

4.3.2 *Skybox*a

*Skybox*ak eszena osoaren inguruko bilgarri bat dira, mundua geometriatik haratago nola ikusiko litzatekeen erakusten duena.

Skyboxak eszena osoaren inguruan errenderizatzen dira, horizontean eszenario konplexu baten itxura emateko. Skyboxak objektu opaku guztien ondoren errenderizatu egiten dira. Errenderizatzeko erabiltzen den *mesh* edo sarea sei texturako kutxa bat edo teseladun[3] esfera bat da.

Unityk defektuz dakarren *Skybox*a aldatzeko argiztapen konfiguraziorako leihoa ireki behar da, *Window* > *Rendering* > *Lighting Settings* menu zabalgarria erabiliz, eta bertako *Skybox Material* parametroa aldatu.

4.8 azpiataleko 4.22 irudian ikus daiteke argiztapen leihoak dauzkan parametro gehienak eta lehen parametro bezala, *Skybox Material* parametroa.

4.3.3 Ituak

Behin arma batez hornitutako robota eta jokalariak inguratzen duen paisaia eginda daudela, galderak erantzuteko sistema egin behar da. Galderak erantzuteko, ituak erabiltzea aukeratu da.

Horretarako, GameObject guraso bat egin da, non barnean, lau ituak izango dituen. Hierarkia hau aukeratu da, errazagoa delako lau ituak kontrolatzea guraso batetik bakoitza bere aldetik baino.

4.6 irudietan ikus daiteke ituen 3D eredua eta ituek daukaten hierarkia, non Ituak GameObjecta gurasoa den eta barnean lau ituak dituen. Barneko itu bakoitzak, itua osatzen duten zatietaz aparte, testua dauka bere barnean. Testua galdera bakoitzaren lau erantzunak ezartzeko erabiliko da.

Ituak GameObject gurasoak, transform osagaiaz aparte, osagai bakarra dauka:

[3] **Teselaketa** terminoa, gainazal lau bat erabat estaltzen duten irudien patroi bati egiten dio erreferentzia. Bi baldintza bete behar dira: espazio librerik edo zulorik ez geratzea eta irudiak ez gainjartzea. Adibide bat kaleetako bide-zorua izan daiteke.

(a) Ituen 3D eredua. **(b)** Ituen hierarkia.

4.6 Irudia: Ituak eta hauen hierarkia.

- ***Diana Control* scripta.** *DianaControl.cs* scripta, ituetan galderen erantzunak jartzeko erabiltzen da. Erantzunak erdialeatorioak izango dira, eremu baten barnean daudela kontrolatzen baita (erantzun zuzenetik nahiko gertu dauden balioak ateratzeko). Horretaz aparte, scripta ere erabiltzen da jakiteko zein diren erantzun zuzen eta okerrak, geroago *DianaCollision.cs* scriptetik informazio hau erabili ahal izateko.

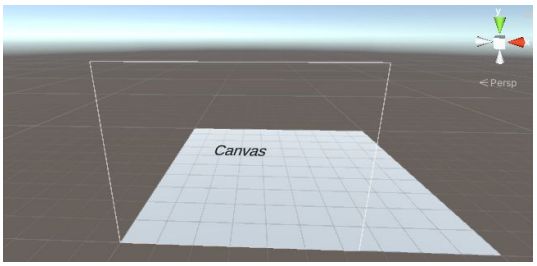
  4.7.5 azpiatalean aurki daitezke script honi buruzko azalpen guztiak.

Barneko itu bakoitzak (semeak) lau osagai dauzka:

- ***Transform* osagaia.** Osagai honen bidez, ituen mugimendua kontrolatuko da. Mugimendua sortzeko, Transform osagaiaren posizio aldagaia uneoro aldatzen egongo da. Mugimendua 2Dn izango da, beraz, X eta Y ardatzetan bakarrik eraldatuko da posizioa.

- ***Rigidbody* osagaia.** Rigidbody osagaia, ituen elkar arteko kolisioak detektatzeko erabili izan da. Ituen arteko talka gertatzen denean, mugimenduaren norabidea aldatu ahal izateko. Gainera, osagai hau izango ez balute, itu batek beste bat zeharkatuko luke talka detektatu gabe.

- ***Sphere collider* osagaia.** Sphere collider osagaia, jaurtigaiek ituekin (erantzuna okerra edo zuzena izan den jakiteko) eta ituek haien artean (mugimenduaren norabidea aldatzeko) talka egin dezaten erabiltzen da. Esfera colliderraren ordez 2D Zirkulu colliderra aukeratu zitekeen, baina 2D zirkulu colliderrak problemak ematen zituen. Jaurtigaia oso azkarra bada, fisikako motorraren eguneratze bakar batean (Unityko fisikako motorrak defektuz 50 aldiz egiten ditu kalkuluak segundoko), jaurtigaia colliderraren beste aldean egongo da; hau da, colliderra zeharkatuko luke honek talka detektatu gabe. Horretarako, esfera formako colliderra jartzea aukeratu da, esferak daukan sakonera jaurtigaia hobeto detektatzen laguntzen duelako. Beraz, errendimendu hobeagoa lortzen da sphere collider osagaia erabiliz.



(a) *Canvas* GameObjecta eszena barnean.

(b) *Canvas* hierarkia.

4.7 Irudia: *Canvas*a eta honen hierarkia.

- ***Diana Collision* scripta.** Script honen bidez, jaurtigai batek itua jo egin duela kontrolatzen da. Baita jo egin den itua erantzun zuzena edo okerra izan den kontrolatzen da. Horretaz aparte, ituen mugimendua kontrolatzeaz arduratzen da.

  4.7.6 azpiatalean *DianaCollision.cs* scriptaren azalpen osoa aurkitzen da.

4.4 Jokoaren eszenaren *Canvas*a

Eranskinetako A.1.3 puntuan azaldu den bezala, *Canvas*a UIak sortzeko eta hauen diseinua modu errazago batean egiteko erabiltzen den GameObjecta da. Bideo-jokoaren barneko eszena bakoitzetik Canvas GameObject mota bat izatearekin nahikoa da, Canvas baten barnean, hainbat interfaze desberdin diseina baitaitezke semeak bezala ezarriz. Hainbat interfaze badaude Canvasean, nahikoa da bat aktibatzearekin eta besteak desaktibatuta mantentzearekin une konkretu batean hauen artean nabigatzeko, edota interfazeek haien artean gainjarrriz. Adibidez, bideo-jokoa pausatzen bada, pausa interfazea uneko interfazearen gainean jar daiteke; pausa interfazea garden egitea edo ez proiektuaren araberakoa izango litzateke, biak erabilgarriak izan daitezkeelako.

Proiektu honen kasuan, jokoaren eszenaren Canvas GameObjectaren barnean hiru interfaze ezberdin diseinatu dira, hurrengo azpiataletan azalduko direnak: Jokalariaren interfazea, pausa interfazea eta joko amaiera interfazea.

4.7 irudietan Canvas GameObjecta nolakoa den eta proiekturako erabili den Canvas hierarkia erakusten da.

Barneko interfazeetaz aparte, Canvas GameObjecta hurrengo osagaiak dauzka:

- ***Rect Transform* osagaia.** Rect Transform osagaia, 2D Transform osagaiaren antzekoa da, baina, puntu bat adierazi ordez laukizuzen bat adierazten du, non UI

elementuak honen barruan jar daitezkeen. Osagai hau, funtzean, UIko elementuen kokapena errazteko erabiltzen da.

Canvas GameObjectak duen Rect Transform osagaiaren parametro guztiak blokeatuta daude eta ezin dira aldatu. Osagai hau Canvasaren seme guztiek heredatzen dute, eta parametroak, GameObject semeetan aldatu daitezke.

- ***Canvas* osagaia.** Canvas osagaiak UIa kokatzen eta errenderizatzen den espazio abstraktua adierazten du. Osagai honen bidez, Canvasa hainbat modutan errenderiza daiteke, baina proiektuaren kasuan, estalketa motako errenderizazioa aukeratu da. Modu honetan, Canvasa pantailara egokitzeko eskalatu egiten da, eta gero, zuzenean errenderizatzen da eszena edo kameraren erreferentziarik gabe. UIa beste edozein grafikoren gainean marraztuko da, hala nola kameraren ikuspegiaren gainean.

  4.8 irudian ikus daiteke nolakoa den erabili den estalketa motako errenderizazioa, non UIa objektuen gainean errenderizatzen den.

4.8 Irudia: *Canvas*aren estalketa errenderizazioa.

- ***Canvas Scaler* osagaia.** Canvas Scaler osagaia, Canvaseko UI elementuen eskala orokorra eta pixelen dentsitatea kontrolatzeko erabiltzen da. Eskala horrek dena ukitzen du mihisean, letra-tamaina eta irudi-ertzak barne. Eskalaketa hau, Canvaseko elementu guztiei eragiten die, letra-tamaina eta irudi-ertzak barne. Hainbat eskalaketa metodo daude, baina kasu honetan pantailaren tamainaren araberako eskalaketa metodoa aukeratu da. Metodo honen bidez, UIko elementuak pantailaren tamainaren edo bereizmenaren arabera eskalatzen dira.

- ***Graphic Raycaster* osagaia.** Graphic Raycaster osagaia, Canvas baten aurka izpiak jaurtitzeko (Raycast bidez) erabiltzen da. Raycasterrak grafiko guztiak begiratzen ditu Canvasean, eta horietako bat jo izan den zehazten du.

- ***Pause Menu Control* scripta.** Script honen bidez, jokoa noiz gelditu edo pausatuko den eta noiz jarraituko den kontrolatzen da. Jokoa gelditzean pausa interfaze bat aterako da eta horren kontrola baita ere script honen bidez egiten da.

4.7.7 azpiatalean aurkitzen da script honen azalpen osoa.

- ***Game Over Menu Control* scripta.** Script honen bidez, jokoaren partida amaitu egin denean ateratzen den joko amaierako interfazea kontrolatzen da.

  Script honen azalpena 4.7.8 azpiatalean aurkitzen da.

- ***Question Controller* scripta.** Script hau, galderak eta erantzunak sortzeko erabiltzen da. Scripta, galdera eta erantzunak sortu behar den bakoitzean aktibatzen da, eta bestela, desaktibatuta egongo da.

  QuestionController.cs scriptaren azalpen osoa 4.7.7 azpiatalean aurki daiteke.

4.4.1 Jokalariaren Interfazea

Canvaseko UI elementuak hierarkian agertzen diren ordena berean marrazten dira. Lehenengo semea marrazten da lehenik, bigarrena gero, eta horrela.

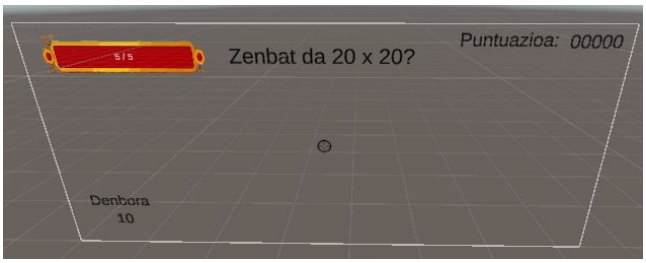
Jokalariaren interfazea interfazerik garrantzitsuena izango denez, canvaseko hiru interfazeetan lehena izatea aukeratu da. Gainera, bi UI elementu gainjartzen badira, azkena goiztiarrenaren gainean agertuko da. Canvaseko interfazeetatik hau izango da beti aktibatuta egongo den bakarra, beste biak aktibatu eta desaktibatu egingo direlako behar den une konkretuetan.

Jokalari interfazea hierarkian lehena denez, beste interfazeak aktibatzean ez litzateke jokalari interfazea desaktibatu beharko, honen gainean marraztuko direlako.

Jokalari interfazeak jokalariaren aurrerapena erakusten du partida osoan zehar, horregatik, uneoro aktibatuta egotea aukeratu izan da. Interfaze honen bidez, jokalariak unean dauzkan bizitza kopurua eta puntuazioa adierazten da, baita zein den momentuan erantzun behar den galdera eta galdera hori erantzuteko falta den denbora erakusten da. Horretaz aparte, tiro egiteko erretikulua interfaze honetan ere marrazten da, jokalariak jakiteko nora apuntatzen dagoen uneoro.

4.4.1.1 Bizitza puntuak

4.10 irudian jokalariak unean dauzkan bizitzak ikusteko sortu den bizitza-barra ikus daiteke. Bizitza-barra egiteko, bi irudi gainjarri egin dira, lehenik eta behin, atzealdeko irudia (hori koloreko bordeak barneko barra beltzarekin), irudi estatikoa da. Aldiz, barra gorria, atzealdeko irudiaren gainean jarri den beste irudi bat da; irudi hau *filled* (beteta) motako

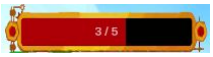


4.9 Irudia: Jokalari interfazea.

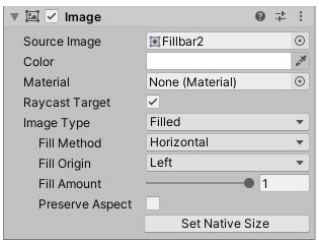
irudia da. Irudi mota hauekin irudiaren zati bat bakarrik erakustea ahalbidetzen da. Bi irudietaz aparte, testu bat jarri da bi irudien gainean uneko bizitza kopurua adierazteko.

4.11 irudian ikus daitezke *filled* motako irudi batek dauzkan parametroak. Parametro horietatik hiru dira garrantzitsuenak:

- ***Fill method* parametroa.** *Fill method* parametroak, barra nola hustu edo beteko den adierazten du (horizontalki, bertikalki, erloju baten orratza bezala...).
- ***Fill Origin* parametroa.** *Fill Origin* parametroak, betetze prozesuaren abiapuntua kontrolatzen du, *left* bada, barra hustean eskuinetik ezkerrera joango da txikitzen.
- ***Fill Amount* parametroa** *Fill Amount* parametroak, *filled* motako irudiek daukaten parametrorik garrantzitsuena da, parametro hau erakutsiko den irudiaren kantitatea adierazten du.



4.10 Irudia: Jokalariaren bizitza-barra.



4.11 Irudia: *Filled* motako irudien parametroak.

Bizitza-barra *Transform Rect* osagaia izateaz aparte, osagai bakarra dauka:

- ***Health Bar* scripta.** Script honen bidez, zein izango den jokalariak izango dituen bizitza kopurua erabakitzen da. Horretaz aparte, jokoan zehar bizitzak galtzean, barra eguneratzeaz arduratuko da (*Fill Amount* parametroa erabiliz).

  4.7.10 azpiatalean aurkitzen da script honen azalpena.

4.4.1.2 Puntuazioa, tenporizadorea eta galderak

Jokalari interfazearen puntuazioa, tenporizadorea eta galdera testuak adierazteko, *TextMeshPro* testu GameObjecta erabili izan da. GameObject mota hau, testuak sortzeko eta hauek editatzeko erabiltzen da eta testuekin lan egiteko mota indartsuena da. Testu mota hau sortzeko, **GameObject > UI > Text - TextMeshPro** menu zabalgarriko bidea erabili da.

Bi *TextMeshPro* GameObject daude, bata 3D objektu normala bezala sortzen da, eta bestea, UIetan bakarrik erabiltzeko sortzen da.

UIko *TextMeshPro* GameObjecta sortzean, Rect Transform osagaiaz aparte, bi osagai sortzen dira defektuz:

- ***Canvas Renderer* osagaia.** Izenak dioen bezala, osagai honek Canvas bateko UI objektu grafiko bat errenderizatzen edo marrazten du. Menuan eskuragarri dauden UIko objektu estandar guztiek (**GameObject > UI**) daukate Canvas Renderer osagaia erantsita.

- ***TextMeshPro - Text (UI)* osagaia.** Osagai honen bidez, dokumentu bat idazteko programa batek ematen dizkizun herreminten antzerako parametroak daude eskura, hala nola, idatzi nahi den testua idazteko, testuaren kolorea, tamaina, alineaziaoa edota letra mota aldatzeko, etab.

4.4.2 Pausa eta Joko Amaierako Interfazeak

Pausa eta joko amaierako interfazeak nahiko antzekoak direnez, elkarrekin azalduko dira. Bi interfaze hauetan jokalariak botoien bidez interfazearekin elkarreragin behar du.

4.12 irudietan, pausa eta joko amaierako interfazeak ikus daitezke. Bi interfazeak botoien bidezkoak dira eta jokalariak sagua mugituz eta botoi hauetan klikatuz interfazearekin elkarreragin ahalko du. Botoia, erabiltzaileak defektuz klik egin eta askatzen duenean

(a) Pausa Interfazea. **(b)** Joko Amaierako Interfazea.

4.12 Irudia: Pausa eta Joko Amaierako Interfazeak.

ekintza bat abiarazteko diseinatuta dago, beraz, saguaren klika askatu aurretik botoiaren kontroletik mugitzen bada, ekintza ez da gauzatuko.

Canvasean botoi bat sortzeko, **GameObject > UI > Button - TextMeshPro** menu zabalgarriko bidea aukeratu behar da. Behin sortu egin dela, bi osagai aurkitu daitezke barnean: *Image* osagaia eta *Button* osagaia.

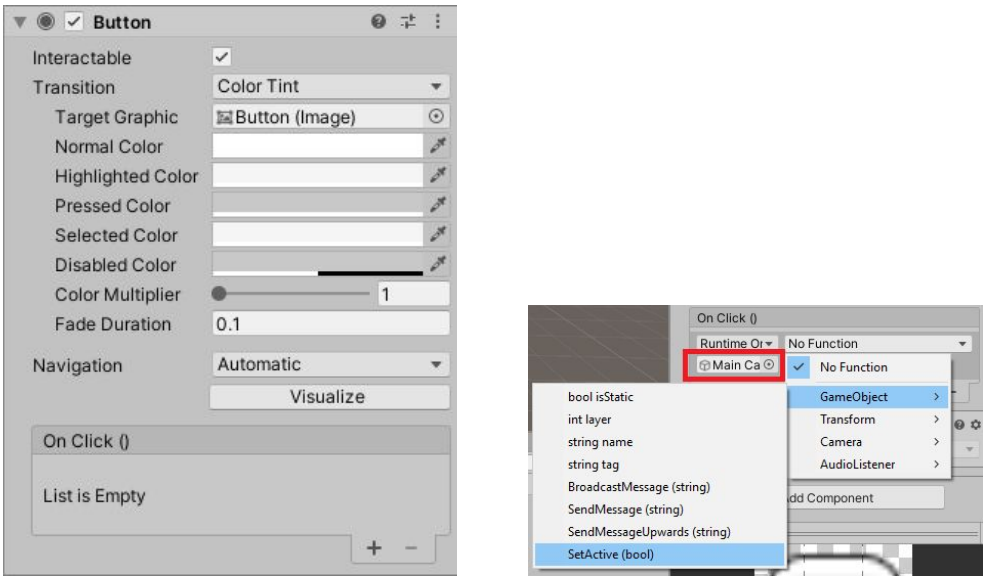
Image osagaia jada azaldu denez, bakarrik *Button* osagaia azalduko da orain. Button osagaia, Canvasean sortzen diren botoi guztiek daukaten osagaia da.

4.13 irudietako lehenak, botoi osagaiak dauzkan parametroak erakusten ditu. Parametro garrantzitsuenak hurrengo hauek dira:

- ***Interactable* parametroa.** Parametro honek botoiarekin elkarreragin daitekeen ala ez zehazten du, hau da, osagai honek *input*a onartuko duen edo ez.
- ***OnClick* funtzioa.** *OnClick* funtzioa, erabiltzaile batek botoian klik egin eta askatzen duenean deitzen den *UnityEvent* bat da. Modu errazago batean esanda, botoiari funtzionalitate bat ematen dio erabiltzaileak botoia sakatzean.

4.13 irudietako bigarren irudian *OnClick* funtzioa nola ezartzen den ikus daiteke. Irudian lauki gorri bat dago, bertako parametroa, eraldatu edo eragin nahi den GameObjecta adierazten du. Irudiko kasuan, kamera nagusia sartu da parametro bezala eta defektuz hainbat funtzio aurkitu daitezke (GameObject motaren araberakoak). *SetActive* funtzioa aurkeratu dela ikus daiteke irudian.

Behin funtzioa aukeratu dela, funtzio horren araberako parametroak atera daitezke. Kasu honetan, laukitxo bat ateratzen da hau markatzeko edo ez (*true* edo *false*); horren bidez,



(a) *Button* osagaiaren parametroak.

(b) Botoiari funtzionalitatea gehitzeko modua.

4.13 Irudia: Pausa eta Joko Amaierako Interfazeak.

jokoa hastean eta botoi hori sakatzean, parametro bezala sartu den GameObjecta desaktibatu edo aktibatuko litzateke aukeratu denaren arabera.

Botoian defektuzkoa ez den funtzionalitate bat jarri nahi bada (norberak sortutako script bidezko funtzioren bat), erabili nahi den funtzioa duen scripta pasatu beharko litzateke parametro giza. Script hori GameObject baten osagaia bada, GameObjecta pasa daiteke parametro bezala.

4.5 Menu nagusia

Menu nagusia normalean jokoa exekutatzean jokalariek ikusi eta esperimentatzen duten lehen erabiltzaile-interfazea da. Jokalarien hasiera-puntua da eta horren ondorioz, bideojokoetan menu sinple eta ulergarriak egiten dira.

Menu nagusiaren eszena, Canvas batez, kamera nagusi batez, soinu efektuak barnean dauzkan *Audio Source* motako GameObject (4.6.2 azpiatalean sakonago azalduta) batez, musika gordetzeko erabiliko den beste GameObject batez eta eta *EventSystem*a (A.1.3 puntuan azalduta) (Canvasa sortzean automatikoki sortzen den GameObjecta da, Canvasa interaktiboa izatea egiten duena) hornituta dago.

Menu nagusian erabiltzen den kamera nagusia, jokatuko den mailaren eszenaren kamera

(a) Menu nagusiko *Canvas*aren hierarkia

(b) Canvaseko menuetatik nabigatzeko adibidea.

4.14 Irudia: *Canvas*aren hierarkia eta nabigazioa.

nagusiaren berdina da. Aldatzen den gauza bakarra, kamera nagusi hau estatikoa dela da. Kamera honen kasuan, ez dira inolako aldaketarik egin parametrizazio aldetik.

Canvas GameObjectaren kasuan, defektuzko osagaietaz aparte, bakarrik script bat dauka:

- ***Menu Control script*a.** Osagai honek, menuaren kudeaketa guztiaz arduratzen da, hala nola, eszena aldatzeaz, hizkuntzaren aldaketaz edota puntuazio altuenen taulak erakuzteaz.

  4.7.9 azpiatalean aurkitzen dira MenuControl.cs scriptari buruzko azalpen guztiak.

4.14 irudietako ezkerreko irudian ikus daiteke nola menuko azpimenu guztiak Canvaseko "ume"bezala ezartzen diren. Canvaseko menuetatik nabigatzeko botoien *OnClick* funtzioa erabiltzen da, horrek inplementazioa asko errazten du, ez baita scripten beharrik botoi askoren funtzionamendurako. *OnClick* funtzioan, bakarrik uneko menuko GameObjecta desaktibatu eta hurrengo menukoa aktibatu behar da menuko nabigazioa lortzeko. Eskuineko irudian ikus daitekeen moduan, erabiltzailea menu nagusian dagoenean eta puntuazioen menura joateko botoia sakatzean, *OnClick* funtzioan menu nagusia eta puntuazioen menua pasatzen dira parametro bezala. Menu batetik bestera joateko, menu nagusia desaktibatu eta puntuazioen menua aktibatu baino ez da egin behar.

Menu nagusiko *Canvas*a, jokatuko den mailaren eszenak daukan *Canvas*aren antzekoa da. Funtzionalitate desberdinak dituzten botoiak 4.7.9 azpiatalean azalduko dira. Beraz, *Canvas* honetan azaldu ez den elementu edo GameObject bakarra, *Dropdown* GameObjecta (GameObject zabalgarria euskeraz) da.

UIko *Dropdown* GameObjecta, erabiltzaileak aukera-zerrendako aukera bakarra hautatzeko erabiltzen da. Kontrolak une horretan hautatutako aukera erakusten du. Klik egindakoan, aukeren zerrenda irekitzen du, aukera berri bat hauta dadin. Aukera berri bat hautatzean, zerrenda itxi egiten da, eta kontrolak hautatutako aukera berria erakusten du.

4.15 irudietan *Dropdown* GameObjecta, eta GameObject honek daukan *Dropdown - TextMeshPro* osagaiaren parametroak ikusi daitezke.

Dropdown GameObjecta, Canvaseko beste UI elementuek dauzkaten osagai berdinak dauzka (*Rect Transform, Canvas Renderer eta Image* osagaiak), osagai bat izan ezik:

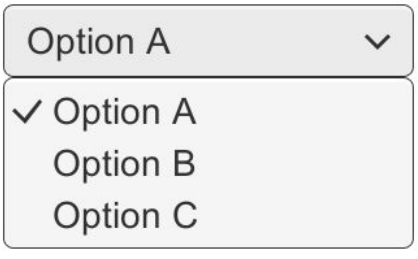
- ***Dropdown - TextMeshPro* osagaia.** Osagai hau, botoi batek daukan *Button* osagaiaren oso antzekoa da, hasierako parametro guztiak berdinak dira bi kasuetan. Ezberdinak diren parametroetatik, hiru dira garrantzitsuenak:
  - ***Value* parametroa.** Parametro honek, defektuz erakutsiko den aukeraren indizea adierazten du; adibidez, 0 bada, *Dropdown* GameObjectak lehen aukera izango du aukeratua defektuz, eta hori izango da erabiltzaileak ikusiko duena.
  - ***Options* parametroa.** Parametro honetan, *Dropdown* GameObjectak izango dituen aukera posibleen zerrenda adierazten da, non aukera bakoitzerako, testu bat eta irudi bat zehaz daitekeen.
  - ***On Value Changed(Int32)* funtzioa.** Funtzio hau, botoien *OnClick* funtzioaren antzekoa da. Erabiltzaileak *Dropdown* zerrendan aukeretako bat klikatzen duenean sortzen den *UnityEvent* bat da. Kasu honetan, zenbaki osoa bidaltzeko aukera ematen du, 0 lehenengo aukera da, 1 bigarrena, eta horrela. Balio hori, gero scripteko funtzioren batekin kontrolatu ahal izateko.

4.6 Miszelanea

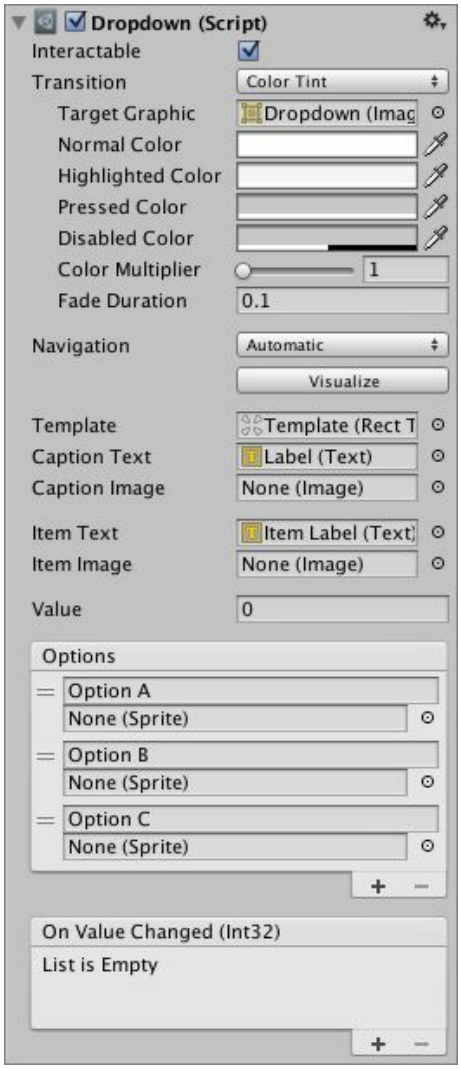
Atal honetan, bideo-jokoa egiteko erabili izan diren bestelako gauzen azalpenak egingo dira, hala nola, partikula-efektuak, animazioak edota soinu efektuak.

4.6.1 Partikula-efektuak

Partikula-sistema batek partikula izeneko irudi txiki edo *Mesh* (sare) asko simulatu eta errenderizatzen ditu ikus-efektu bat sortzeko. Sistema bateko partikula bakoitzak efek-



(a) UIko *Drowdown* GameObjecta.



(b) *Dropdown* - *TextMeshPro* osagaiaren parametrizazioa.

4.15 Irudia: *Dropdown* GameObjecta eta parametrizazioa.

tuaren elementu grafiko bat adierazten du. Sistemak partikula bakoitza kolektiboki simulatzen du efektu osoaren inpresioa sortzeko.

Partikula-sistemak erabilgarriak dira objektu dinamikoak sortu nahi direnean, hala nola sua, kea edota likidoak sortzeko; zaila baita horrelako objektuak sare (3D) batekin edo *Sprite* (2D) batekin irudikatzea.

Partikula-efektuak egiteko sistemak konplexuak dira, eta parametro-kopuru oso handia daukate, mota askotako partikula-efektu lortzeko. Horregatik, ez da asko sakonduko gai honetan.

Bideo-joko honetan erabili izan diren partikula-efektu guztiak, Unityk dakarren partikula sistemarekin egin izan dira.

Partikula-efektu bat egiteko, **GameObject > Effects > Particle System** bidea aukeratu behar da. Behin bide hori aukeratu dela, *Particle System* deituriko GameObject bat sortuko da.

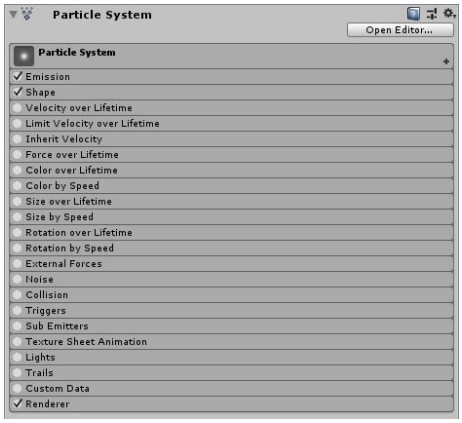
GameObject honek, bi osagai dauzka:

- ***Transform* osagaia.** Osagai honen bidez, sortuko den partikula-efektuen posizioa, rotazioa eta eskala adieraziko da. Kasu honetan, osagai garrantzitsua da, partikula-efektuak normalean erabilera bakarrekoak izaten direlako. Hau da, partikula-efektua sortzen da, eta horren iraupena (iraupen finkoa izaten dute, baina begiztan egon daitezke) agortzean, partikula-efektua ez da berriro erabiliko. Horregatik, partikula-efektu bakoitzeko hainbat instantziazio egiten dira, eta beraz, garrantzitsua da Transformazio egokia izatea.

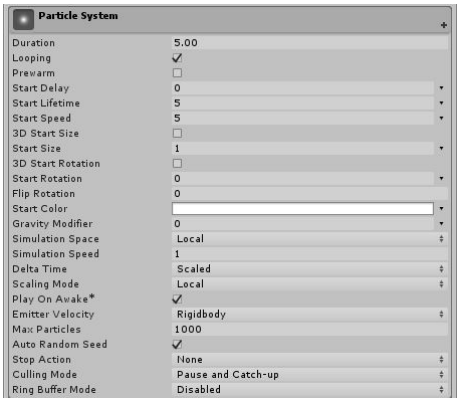
- ***Particle System* osagaia.** Osagai hau, partikula-efektuak egiteko erabiltzen da. Partikula-sistema osagaiak propietate asko dauzka, eta, erosoagoa izan dadin, ikuskatzaileak "modulu" izeneko atal tolesgarrietan antolatzen ditu.

  4.16 irudietako ezkerreko irudian, partikula-sistema osagaiak dauzkan modulu guztiak ikusi daitezke. Eskuineko irudian aldiz, modulu nagusiak dauzkan parametro guztiak ikusi daitezke. Modulu nagusiak, sistema osoari eragiten dioten propietate orokorrak dauzka. Propietate horietako gehienek partikula sortu berrien hasierako egoera kontrolatzen dute.

  Modulu nagusian parametro asko daude. Sistemarekin zer lor daitekeen ikusteko, modulu honetako parametro basikoenak azalduko dira, hala nola, *Duration, Looped, Start* parametroak edota *Gravity modifier*.



(a) *Particle System* osagaiaren moduluak.



(b) *Particle System* osagaiaren modulu nagusia.

4.16 Irudia: *Particle System* osagaiaren moduluak eta modulu nagusia.

Duration parametroari ezker, sistemak partikulak igortzen ditu iraupen konkretu batez, eta *Looped* (begiztatuta) propietatea erabiliz, partikulak etengabe emititzeko konfigura daiteke.

Start propietateek (bizi-denbora, abiadura, tamaina, errotazioa eta kolorea) igortzen ari den partikula baten egoera zehazten dute.

Gravity modifier parametroari ezker, partikulek jasango duten grabitate indarrak zehaz daiteke.

Balio finko batez edota kurba batez finkatu daitezke parametro gehienak. Adibidez, kurba baten bidez, partikulek denboran zehar izango duten tamaina zehaz daiteke.

4.6.2 Soinu efektuak

Unity bidez soinu efektuak egiteko *Audio Source* GameObjectak erabiltzen ditu. *Audio Source* GameObjectaren bidez, eszenan audio klip bat erreproduzi daiteke, eta soinua entzuteko, normalean kamera nagusiak daraman *Audio Listener* osagaia erabiltzen da. *Audio Source* GameObject bat sortzeko, **GameObject > Audio > Audio Source** menu zabalgarriko bidea aukeratu behar da. Behin bide hori aukeratu dela, eszenan *Audio Source* motako GameObjecta sortuko da.

Audio Source motako GameObejct batek, *Transform* osagaiaz aparte, osagai bakarra dauka:

- ***Audio Source* osagaia.** Osagai honen bidez, eszenan soinua erreproduzitzen da.

Osagai honek parametro asko dauzka; horregatik, oinarrizkoenak eta garrantzitsuenak baino ez dira azalduko:

- ***AudioClip* parametroa.** Parametrorik garrantzitsuena, erreproduzituko den soinu-kliparen fitxategiaren erreferentzia.
- ***Priority* parametroa.** Audio-iturriaren lehentasuna zehazten du eszenan dauden guztien artean.
- ***Volume* parametroa.** Zein zaratatsua izango den soinua *Audio Listener* osagaitik munduko unitate bateko distantziara.
- ***Spatial Blend* parametroa.** Audio-iturria 2D eta 3D artean non kokatzen den adierazten du. Hau da, parametroa 0 bada, soinua 2D bezala tratatuko da; parametroa 1 bada, soinua 3D bezala tratatuko da, eta 0 eta 1 artean badago, zenbakiaren araberako bien arteko nahasketa bezala tratatuko da. 2D iturriek espazializaziorik gabe erreproduzitzen dira, aldiz, 3D iturriei espazio-kokapenak eta hedaketak eragiten diete.

Audio-iturriak UIko elementuetan erabiltzeko (botoietan eta lista hedagarrietan), *OnClick* eta *OnValueChanged* funtzioak erabili dira, soinuen inplementazioa errazteko, horrela ez delako script bidez kontrolatu behar.

Gainera, UIko botoi eta lista hedagarriei, *Event Trigger* osagaia gehitu egin zaie kurtsorea botoiaren gainetik pasatzen denean soinu-efektuak erreproduzitzeko.

Event Trigger osagaia, espezifikoki soinu-efektuak sortzeko erabili denez, atal honetan azalduko da:

- ***Event Trigger* osagaia.** *Event Trigger* osagaia, Event Systemeko gertaera bakoitzerako deitu nahi diren funtzioak zehazteko erabiltzen da. Gertaera bakar bati funtzio bat baino gehiago eslei dakizkioke, eta *Event Trigger*ek gertaera hori jasotzen badu, funtzio horiek eman izan diren ordenan deituko ditu. Hurrengo hauek gertaera mota batzuk dira, baina gehiago aurkitu daitezke:
  - ***PointerEnter* gertaera.** Gertaera hau, saguaren erakuslea jokoaren objektuaren gainean sartzen denean deitzen da. Soinu-efektuak gehitzeko erabili izan den gertaera da, eta 4.17 irudian ikus daiteke nolakoa den *Event Trigger* osagaia *PointerEnter* gertaerarekin. Soinu-efektuak erreproduzitzeko, *Audio Source* GameObjecta pasatu behar zaio parametro bezala, eta deitu nahi den funtzioa ezarri; kasu honetan, *Audio Source*k daukaten *Play*() funtzioa erabili da.

– ***Drag* gertaera.** Gertaera hau, objektu bat arrastatzen den bitartean (arrastatze gertaera ez du zertan objektua mugitu behar; adibidez, saguaren ezkerreko botoiarekin aukeratu eta botoia sakatuta mantenduz, sagua mugitzeari deritzo) erakuslea mugitzen den bakoitzean deitzen da.

– ***Deselect* gertaera.** Gertaera hau, objektu berri bat aukeratzen denean deitzen da.

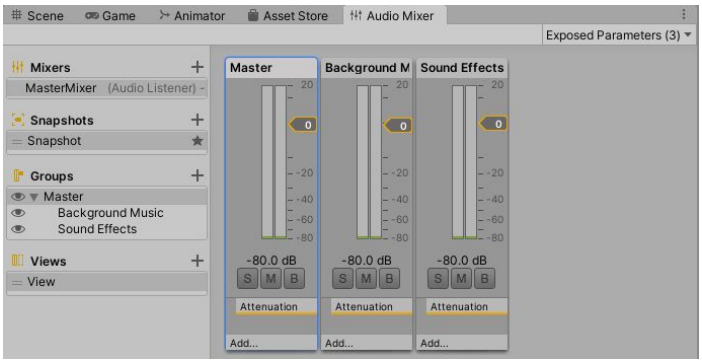
4.17 Irudia: *Event Trigger* osagaia.

Bideo-jokoaren soinua kontrolatzeko, *Audio Source* GameObjectak erabiltzeaz aparte, *Audio Mixers* edo audio-nahasgailuak erabili dira. Audio-nahasgailuak sortzeko, *Audio Mixer* leihoa ireki behar da, **Window > Audio > Audio Mixer** menu zabalgarriko bidea aukeratuz. Audio-nahasgailuaen bidez, modu erraz batean kontrola daiteke eszenan egongo diren soinu ezberdinen bolumenak. Horretaz aparte, soinuei efektuak (oihartzuna, erreberberazioa...) edota filtro desberdinak jarri dakizkioke.

4.18 irudian, audio-nahasgailuaren leihoa ikus daiteke. Audio-nahasgailu batean, talde bat osatzen duten hainbat audio-nahasgailu sortu daitezke. Taldeak, zuhaitz itxura izango du, eta gurasoaren bidez, beste guztiak kontrola daitezke. Adibidez, gurasoaren bolumena jaitsiz, seme guztien bolumena ere jaitsiko da, aldiz, semeen bolumena jaistean, ez du besteengan eraginik izango. Kasu honetan, hiruko talde bat sortu da jokoaren bolumena kontrolatzeko. *Master* nahasgailua, soinu orokorra kontrolatuko du, *background music* nahasgailua, musika kontrolatuko du, eta *sound effects* nahasgailua, soinu efektuak kontrolatzeko erabili da. Hiru audio-nahasgailu hauek, *MusicController.cs script*etik maneiatzen dira. Jokalariak, menu nagusiko aukeren interfazean dauden barra-irristagarrien bidez jokoaren bolumena kontrolatu ahalko du.

4.7.13 azpiatalean *MusicController.cs script*ari buruzko azalpenak aurkitu daitezke.

Eszenan dauden *Audio Source* GameObjectak audio-nahasgailuetatik eraldatzeko, hauen audioa audio-nahasgailuetara berbideratu behar da. Horretarako, *Audio Source* osagaiak daukan *output* edo irteera parametroa erabili behar da, audio kliparen soinua erreproduzituko duen soinu-nahasgailua aukeratzeko.



4.18 Irudia: *Audio Mixer* leihoa.

4.7 Scripten azalpenak

GameObjecten portaera, erantsita dauzkaten osagaien bidez kontrolatzen da. Nahiz eta Unityk erabilera askoko osagaiak dauzkan integratuta, gehienetan, sortu nahi den jokoaren ezaugarriak ezartzeko osagai hauek eman dezaketena baino gehiago behar da. Unityk, scriptak erabiliz nahi den osagai pertsonalizatua sortzeko aukera ematen du. Script hauen bidez edozein gauza lor daiteke, hala nola, gertaerak eragin, denboran zehar osagaien propietateak aldatu, erabiltzailearen inputari nahi den moduan erantzun...

Unityn script bat sortzeko, **Asset > Create > C# script** menu zabalgarriko bidea aukeratu behar da. Unityko script bat ez da programa baten ideia tradizionala bezalakoa; programa hauetan, kodea etengabe exekutatzen da begizta batean, harik eta lana osatu arte. Unityk, berriz, script bati pasatzen dio kontrola aldizka, haren barruan deklaratzen diren funtzio batzuei deituz. Funtzio bat exekutatu ondoren, kontrola Unityra bueltatzen da berriro. Funtzio horiei gertaera-funtzio deritze, Unityk aktibatzen baititu jokoan gertatzen diren gertaerei erantzuteko.

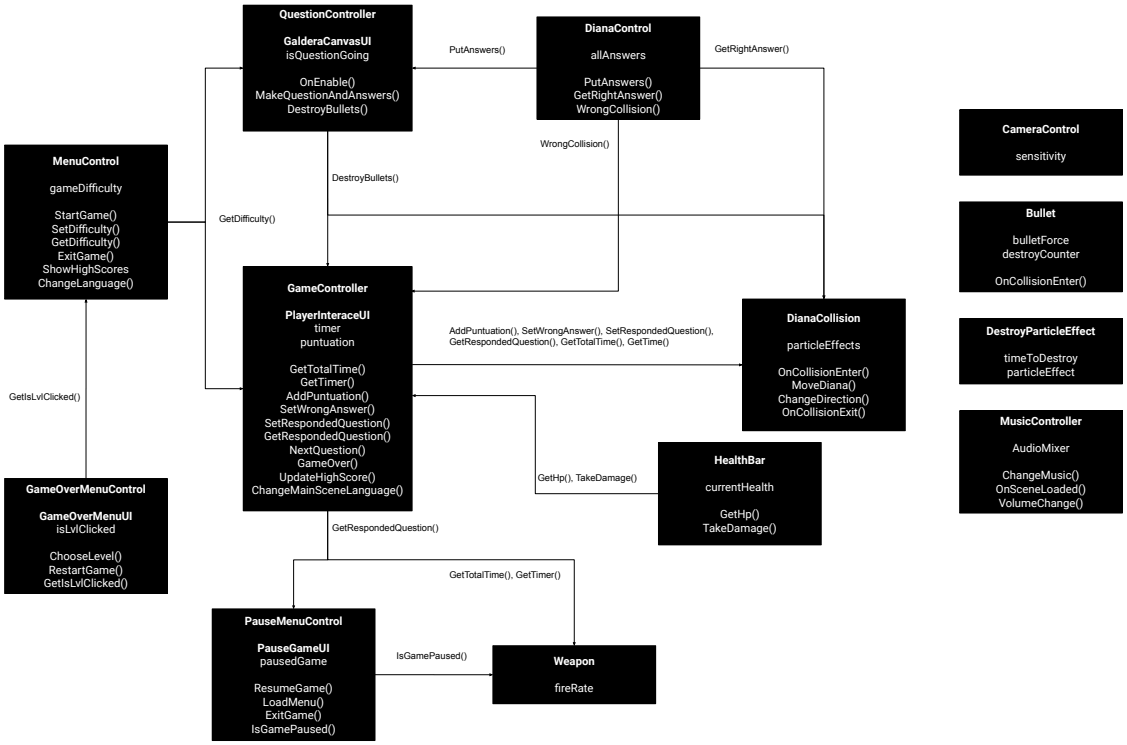
Unityk hainbat gertaera-funtzio dauzka, baina erabilienak edota ezagunenak hiru dira:

- ***Start* gertaera-funtzioa.** *Start* gertaera-funtzioa, script bat gaitzen den *frame* (fotograma) edo unean deitzen da, edozein eguneratze-metodo lehenengo aldian deitu baino lehentxeago. Start gertaera-funtzioa scriptaren bizitza osoan behin deitzen da, nahiz eta scripta daukan GameObjecta desgaitu eta berriro gaitu.

- ***Update* gertaera-funtzioa.** *Update* gertaera-funtzioa, *frame* bakoitzean deitzen da, scripta gaituta badago. *Update* gertaera-funtzioa, gehien erabiltzen den funtzioa da.

- ***Fixed Update* gertaera-funtzioa.** *Fixed Update* gertaera-funtzioa, fisikako kalkuluak egiteko eta fotograma-tasarekiko (*frame rate* edo fotogramak segunduko) independentea den funtzioa da. Fisikako sistemaren frekuentzia dauka; Unityk daukan balio lehenetsia 50 dei segundoko da. *Update* gertaera-funtzioa baino lehenago deitzen da.

4.19 irudian, bideo-jokoan erabili diren script guztien antolamendua ikus daiteke.



4.19 Irudia: Bideo-jokoaren script guztien antolamendua.

Hurrengo azpiataletan, bideo-jokoa egiteko sortu diren scriptak azalduko dira.

4.7.1 *Camera Control.cs script*a

*CameraControl.cs script*a, robotak duen kamera mugitzeaz arduratzen den scripta da. Lehenik eta behin, *Start* funtzioan kurtsorea pantaila erdian blokeatu egiten da. *Update* funtzioan, uneoro saguaren mugimendua hartuko da eta horren araberako errotazioa ezarriko da. Saguaren mugimendua detektatzeko, ***Input.GetAxis*** funtzioa erabiltzen da. Funtzio honen bidez, saguaren kurtsorea X (*"Mouse X"* parametroa pasatuz, **Edit > Project**

Settings > Input Manager menuan adierazita dago erabili nahi den edozein *input*aren defektuzko izena edo balioa) edo Y (*"Mouse Y"* parametroa pasatuz) ardatzetan mugitzen ari den jakin daiteke.

Sagua X ardatzean mugitzen bada, robotaren GameObjecta Y[4] ardatzarekiko biratuko da, hau da, alboetara biratuko da. Aldiz, sagua Y ardatzean mugitzen bada, kamera nagusia izango da biratuko dena. Kameraren biraketa, X ardatzarekikoa izango da, eta 180ºra mugatu egingo da, atzera begiratu ezin izateko gora edo behera begiratuz. Kameraren gora eta beherako mugimendutak mugatzeko, *Clamp* funtzioa erabili da. Funtzio honen bidez, parametro bezala pasatutako bi angeluen artean mugatuko da biraketa.

Script hau, robot GameObjectan kokatu da, robotaren *Transform* osagaia eraldatu behar zelako. Horrek, pixka bat errazten du inplementazioa, bestela gurasoari dei egin beharko litzaiokeelako. Kamera maneiua edozein puntutatik egin daiteke, *Camera.main* erabiliz kamera nagusia GameObjecta lortzeko.

4.7.2 *Weapon.cs script*a

*Weapon.cs script*a, robotak daukan arma GameObjectan dago kokatuta. Script honen bidez, armaren funtzionamendua kontrolatzen da, apuntatu eta tiro egin ahal izateko. Scriptaren lan osoa *Update* funtzioan egiten da.

Arma, kameraren erdiko puntura uneoro apuntatzen egoteko, Raycast funtzioa erabili da. 4.20 irudian *Raycast* funtzioa nola erabili den ikus daiteke. Lehenik eta behin, *Ray* motako izpi infinitua sortzen da *ScreenPointToRay* funtzioaren bidez, kameratik parametro bezala pasatzen den pantailako puntura dihoana. Izpia Raycast bidez botatzen da, eta zeozerren kontra talka egiten badu, talkaren informazioa gordetzen da *RaycastHit* egituran. Izpia talka egin badu, *RaycastHit* egitura erabiliko da jakiteko ze puntutan gertatu den talka *Ray.GetPoint()* funtzioarekin, eta armarekin horra begiratzeko *LookAt* funtzioarekin. *LookAt* funtzioaren ondoren dagoen 90º armaren biraketa, armaren errotazioa zuzentzeko da. Arma Blender aplikaziotik inportatzean -90º biraketa zeukan koordenatuen sistema aldaketagatik, eta ezkerrera apuntatzen zegoen.

Raycast funtzioa armarekin apuntatzeko erabili da, tiro-doitasun handiagoa lortzeko, bestela, distantzia handira balak zentrotik desbideratu egiten zirelako.

Nahiz eta fisikako funtzioak *FixedUpdate* gertaera-funtzioan erabiltzen diren, Raycast

[4] Y ardatzarekiko errotazioa, alboetara birazeko da; X ardatzarekikoa, gora eta behera biratzeko, eta Z ardatzarekiko errotazioa, alboetara okertzeko da.

```
//Pantailaren erdiko koordenatuak jaso bektore batean, eta
//kameratik puntu horretara doan izpia Ray objetuan gorde.
Ray ray = Camera.main.ScreenPointToRay(new Vector3(Screen.width / 2, Screen.height / 2, 0));
RaycastHit hitInfo; //Izpiak zeozer jotzean, bere informazia RaycastHit egituran gordeko da.

if (Physics.Raycast(ray, out hitInfo))//Izpia bota, eta zeozerren kontra talka egiten badu,
                                      //erantzuna hitInfo aldagaian jaso.
{
    //Arma pantaila erdira apuntatzeko, LookAt funtzioa erabili.
    transform.LookAt(ray.GetPoint(hitInfo.distance));
    transform.Rotate(0,90,0);//Armako prefab, 90º desbiderapena duenez, zuzendu
}
```

4.20 Irudia: *Weapon.cs script*aren *Raycast* inplementazioa.

kasua ezberdina izan daiteke. Eragiketa garestia da, eta *FixedUpdate* gertaera-funtzioa, *Update* baino gehiagotan deitzen denez, batzuetan komenigarriagoa izan daiteke *Update* funtzioan erabiltzea.

Horretaz aparte, tiro egiteko ahalmena kontrolatzen da *Update* funtzioan. Saguko ezkerreko botoia sakatzean tiro egingo da, jaurtigaia instantziatuz. Ezin da momentuoro tiro egin, tiro-kadentzia segunduko jaurtigai batean ezarrita dago. Jokoa pausatuta ez egotea edota galdera erantzuteko denbora dagoela ere kontrolatzen da.

4.7.3 *Bullet.cs script*a

*Bullet.cs script*a, jaurtigaien portaera kontrolatzeaz arduratzen da. Hiru funtzio ezberdin dauzka, *Start*, *Update* eta *OnCollisionEnter*.

Start funtzioan, jaurtigaiari indarra edo bultzada ematen zaio, armak apuntatzen duen norabide finkoan mugitzeko. Hori egiteko, *Rigidbody*ek daukaten *AddForce* funtzioa erabili da. Funtzio honi, parametro bezala norabide bektorea (normalean, bektore normalizatua balio finko batekin, kasu honetan bultzadaren indarrarekin, biderkatuz) ematen zaio.

Update funtzioa, jaurtigaiek daukaten kontagailua eguneratzeko erabiltzen da. Kontagailua 0ra heltzean, jaurtigaia desagertaraziko da (memoria hustutzeko), *Destroy* funtzioa erabilita.

OnCollisionEnter funtzioa, jaurtigaia beste *Collider* batekin talka egitean deitzen da eta jaurtigaia ere desagertaraziko da.

Script hau jaurtigai GameObjectan dago, horrela, jaurtigaiaren klon bat instantziatzean bultzada jasoko du; GameObject baten instantziazioa egitean haren osagai guztiak ere kopiatzen dira, scriptak barne, horrela, ez da kanpotik kontrol gehiagorik egin behar jaurtigaietan.

4.7.4 *GameControl.cs script*a

*GameControl.cs script*a, jokatuko den mailaren eszenaren script nagusia da. Script honen bidez, jokalariaren interfazea eguneratzen da (bizitzak, puntuazioa eta denbora), noiz amaitzen den jokoa kontrolatzen du, puntuazio altuenak eguneratzen ditu eta abar.

Script honek zazpi funtzio nagusi (*Setter* eta *Getter*rak ez dira azalduko, sinpleak direlako) dauzka: *Start*, *Update*, *AddPuntuation*, *NextQuestion*, *GameOver*, *UpdateHighScore*, eta *ChangeMainSceneLanguage* funtzioak.

Start funtzioan, behar diren aldagai guztien hasieraketak egiten dira, hala nola, kontagailua eta puntuazioa. Gainera, jokoaren maila eszenaren lengoaia aldaketa ere egiten da.

Update funtzioan, galdera erantzuteko kontagailua eguneratzen doa. Horretaz aparte, jokalariaren bizitzak kontrolatzen ditu, baita noiz atera behar den hurrengo galdera, edota joko amaierako pantaila. Erantzun oker bat jo bada edota denbora agortu bada, *DianaControl.cs script*ari erantzun zuzena erakusteko adierazten dio (*WrongCollision* funtzioaren bidez.).

AddPuntuation funtzioa, galdera bat ondo erantzun bada, jokalariak zenbateko puntuazioa jasoko duen kontrolatzen du. Jasotako puntuazioa, galdera erantzuteko geratu den denboraren araberakoa da.

NextQuestion funtzioa, uneko galdera erantzun denean deitzen da eta*QuestionController.cs script*a aktibatzen du hurrengo galdera sortu dadin. Horretaz aparte, galderak erantzuteko kontagailua berrasieratzen da.

GameOver funtzioa, jokalariak bizitza guztiak galdu dituenean deitzen da. Funtzio honen bidez, joko amaierako interfazea aktibatzen da, baita lortu den puntuazioa eguneratuko da.

UpdateHighScore funtzioaren bidez, zailtasunaren araberako hiru puntuazio altuenak gorde eta eguneratzen dira. Horretarako, *PlayerPrefs* klasea erabili izan da.

PlayerPrefs klasea, unityk daukan informazioa testu fitxategi batean gordetzeko modu erraz bat da. Testu fitxategian gorde den informazioa jokoaren saioen artean mantentzen da, eta normalean, jokalariak dituen lehentasunak (aldagai garrantzitsuak) gordetzeko erabiltzen da. Datu bakarra gorde daiteke testu fitxategi edo gako-hitz bakoitzeko.

Puntuazio altuenen balioak gordetzeko *string* motako aldagaia erabili da. *String* aldagaian, balio bat baino gehiago jarri daiteke, bestela bederatzi hitz-gako desberdin erabili beharko lirateke.

ChangeMainSceneLanguage funtzioa, jokoaren mailaren eszenaren hizkuntza aldatzeko erabiltzen da, menu nagusian aukeratu egin den hizkuntzaren arabera. Nahiz eta menu nagusian hizkuntza aldatzeko funtzio bat dagoen, eszenaz aldatzean ez dira balioak gordetzen. Horretarako, hizkuntza aldatzeko beste funtzio bat behar izan da.

4.7.5 *DianaControl.cs script*a.

*DianaControl.cs script*aren bidez, ituak kontrolatzen dira. *Awake* gertaera-funtzioa izateaz aparte, script honek hiru funtzio dauzka; *PutAnswers*, *GetRightAnswer* eta *WrongCollision* funtzioak.

Awake gertaera-funtzioa kasu honetan, *TextMeshPro* motako testuen hasieraketarako erabili izan da. *Awake* funtzioa, *Start* funtzioaren antzekoa da, baina exekutatzen den gertaera-funtzioetatik lehena da. *Start* funtzioa erabilita kasu honetan, hasieraketa erroreak ematen zituen scriptek daukaten exekuzioen ordenagatik. Beraz, *Awake* erabili da hasieraketa lehenago egiteko.

PutAnswers funtzioa, *QuestionController.cs script*etik deitzeko sortu den funtzioa da. Funtzio honen bidez, galdera bat eta bere lau erantzunak sortzen direnean (*QuestionController.cs script*ean), dianetako testuetan erantzunak zoriz jartzeaz arduratzen da. Gainera, erantzun zuzena eta honen indizea gordetzen ditu.

GetRightAnswer funtzioa, erantzun zuzenaren balioa (*integer*) itzultzen du. Funtzio hau, *DianaCollision.cs script*ean erabiltzeko sortu da, honen bidez, jo egin den itua zuzena edo okerra izan den jakingo da.

WrongCollision funtzioa, *DianaCollision.cs script*etik deitzen da. Erantzun okerra jo egin bada, funtzio honi deitzen zaio, erantzun zuzena zein zen adierazteko (partikula efektua sortuz).

4.7.6 *DianaCollision.cs script*a

*DianaCollision.cs script*a, ituen kolisioak eta mugimendua maneiatzeko erabiliko den scripta da. Script honek bost funtzio nagusi dauzka: *Update*, *OnCollisionEnter*, *MoveDiana*, *ChangeDirection* eta *OnCollisionExit*.

Update gertaera-funtzioaren bidez, ituen mugimendu osoa kudeatzen da. Ituen mugimendua, lauki baten bidez definitu da, hau da, definituriko laukiaren barnean mugituko dira.

Ituek definituriko laukitik ateratzen zaiatzean, kontrako norabidean bueltatuko dira. Horretaz gain, denbora bat pasata, ituek norabidez aldatuko direla kontrolatzen du. Galdera bat erantzun denean edota denbora agortu denean, ituek hasierako posiziora itzuliko dira.

OnCollisionEnter funtzioa, ituak beste *Collider* batekin talka egiten duenean deitu egiten da. Funtzio honetan, lehenik eta behin, jaurtigai baten kontra talka egin dela konprobatzen da, eta hala bada, *GameController.cs script*arekin komunikatzen da, abisatzeko galdera erantzun dela. Erantzuna zuzena izan bada, *GameController.cs script*ari puntuazioa gehitzeko abisatzen dio. Aldiz, okerra izan bada, erantzuna okerra izan dela adierazten dio *GameController.cs script*ari (*SetWrongAnswer* funtzioaren bidez).

Horretaz gain, itu batek beste baten kontra talka egin badu kontrolatzen du, eta hala bada, talka egin duten ituen mugimenduen norabideak aldatu egingo dira.

MoveDiana funtzioarekin, mugimendua ezartzen da. Itua, parametro bezala pasatzen den norabidean mugituko du.

ChangeDirection funtzioaren bidez, norabide berri bat aukeratuko da. Norabidea, zailtasunaren araberakoa izango da. Zailtasun normala aukeratzen bada, lau norabide (90ºko multiploak) egongo dira eskuragarri, eta bat aukeratuko da ausaz. Aldiz, aukeratu den zailtasuna zaila izan bada, zortzi norabideen (45ºko multiploak) artean egingo da aukeraketa.

OnCollisionExit funtzioa kolisioen balioen eguneraketarako erabili izan da. Kolisio bat gertatzen denean aldatzen diren balioak, kolisiotik ateratzean berriro eguneratzeko.

4.7.7 *PauseMenuControl.cs script*a

*PauseMenuControl.cs script*aren bidez, *Canvas*eko Pause Menu interfazea kontrolatzen da. Script hau, bost funtzio dauzka: *Update* gertaera-funtzioa, eta *ResumeGame*, *LoadMenu*, *ExitGame* eta *IsGamePaused* funtzio publikoak.

Update gertaera-funtzioan, jokoa pausatzeko botoia sakatu dela konprobatzen da uneoro. Jokoa pausatzeko botoia sakatu bada, eta galdera erantzuteko denbora badago; jokoa pausatuko da. Jokoa pausatzeko denbora gelditu egiten da (denbora eskala gelditzeko, Time.timeScale balioa aldatu behar da), sagua desblokeatu eta erakusgarri egiten da, eta azkenik, *Canvas*eko pausa interfazea aktibatzen da.

ResumeGame funtzioaren bidez, jokoa jarraitzeko botoiaren funtzionamendua inplemen-

tatzen da. Denbora berriro igarotzea eraginez, sagua blokeatuz berriro eta pausa interfazea kenduz.

LoadMenu eta *ExitGame* funtzioak nahiko antzekoak dira. Bi funtzio hauen bidez, menua kargatzeko botoiaren eta jokutik ateratzeko botoiaren funtzionamenduak inplementatzen dira, hurrenez hurren. *LoadMenu* funtzioaren bidez, uneko puntuazioa eguneratzen da, konprobatzeko ea puntuazio altuenetako taulan jarri behar den; eta menu nagusiko eszena kargatzen da. *ExitGame* funtzioaren bidez, puntuazio eguneraketa berdina gertatzen da, eta konprobaketa gauzatu eta gero, aplikazioa ixtea eragiten du (*Application.Quit* funtzioaren bidez).

IsGamePaused funtzioa, *Weapon.cs script*etik jokoa pausatua dagoen kontrolatzeko erabiltzen da. Funtzioaren bidez, jokoa pausatua edo ez dagoen adierazten da, *bool* balio baten bidez.

4.7.8 *GameOverMenuControl.cs script*a

*GameOverMenuControl.cs script*a *Canvas*eko Game Over interfazea kudeatzeko erabiltzen da.

Script honek lau funtzio nagusi dauzka, *Update* gertaera-funtzioa eta hiru funtzio publiko: *RestartGame*, *ChooseLevel* eta *GetIsLvlClicked* funtzioak.

Update funtzioan, Game Over interfazea aktibo dagoen kontrolatzen da, aktibo dagoen bitartean sagua pantaila erditik desblokeatu ahal izateko, eta kurtsorea erakusteko.

Restart funtzioan, jokoa berrabiarazteko botoiaren funzionamendua inplementatzen da. Horretarako, jokoaren eszena kargatzen da berriro. Eszena kargatzeko, eszena manageerreko eszena karga funtzioa erabiltzen da (*SceneManager.LoadScene*).

ChooseLevel funtzioan, maila aukeratzeko botoiaren funtzionamendua inplementatuta dago. Botoia sakatzean, menu nagusiaren eszenako maila aukeraketaren pantailara eramateko. Funtzio hau, *GetIsLvlClicked* funtzioarekin batera erabiltzen da. *GetIsLvlClicked* (*bool* motako balioa itzultzen du) funtzioaren bidez, *MenuControl.cs script*etik jakin daiteke maila aukeratzeko botoia sakatu den edo ez, menu nagusiko eszena kargatzean.

4.7.9 *MenuControl.cs scripta*

MenuControl.cs scripta, menu nagusiaren funtzionamenduaz arduratzen da. Script honek sei funtzio dauzka *Start* gertaera-funtzioaz aparte. *StartGame*, *SetDifficulty*, *GetDifficulty*, *ExitGame*, *ShowHighScores* eta *ChangeLanguage* funtzioak.

Start funtzioan, menu nagusiaren hizkuntza aldatu egiten da jokalariak azken aldiz aukeratu zuen (aurreko saioan) hizkuntzaren arabera. Bideo-jokoaren hizkuntza, *Dropdown* (lista zabalgarria) GameObjectaren bidez aukeratzen da. Horretaz gain, jokoaren hasierako bolumena ezartzen da aurreko saioan ezarri diren balioen arabera (lehen saioan, defektuz bolumen maximoa ezarriko da). Bolumena, menu nagusiaren aukera interfazearen bidez ezartzen da. Aukeratu den hizkuntza eta bolumenaren balioak, *PlayerPrefs* klasearen bidez gordeko dira.

Hasierako hizkunta jartzeaz aparte, jokalaria *Game Over* interfazeko maila aukeratzeko botoia sakatu duen kontrolatzen du. Botoi hori sakatu bada, menu nagusiko eszena kargatzean, jokatzeko botoia sakatuko da scriptetik bertatik, maila aukeratzeko interfazea karga dadin.

StartGame, *SetDifficulty* eta *ExitGame* funtzioek, botoi ezberdinen funtzionamenduak inplementatzen dituzte. Jokoa hasteko maila bat aukeratzean, *StartGame* eta *SetDifficulty* funtzioei deituko zaie. Lehenak, mailaren eszena kargatuko du, eta bigarrenak aukeratu den botoiaren araberako balioa gordeko du parametro batean (maila erraza aukeratzean, 0 balioa jasoko da; maila normala aukeratzean 1 balioa eta maila zailena aukeratzean, 2 balioa). *ExitGame* funtzioan, aplikazioa itxiko da.

ShowHighScores funtzioan, *PlayerPrefs* klasearen bidez gorde diren puntuazio altuenak (*string* bidez) erakutsiko dira menu nagusiko *High Score* interfazean. *High Score* interfazean dagoen lista zabalgarrian dauden aukeraren arabera agertuko dira balio maximoak. Lista zabalgarriko aukera bat aukeratzean, funtzio honi deitzen zaio balio ezberdin bat pasatuz (*integer* motako balioa) parametro bezala, adibidez, maila erraza aukeratzean 0 balioa pasatuz.

ChangeLanguage funtzioak, puntuazio altuenak erakusteko funtzioaren antzekoa da. Kasu honetan, menu nagusiaren testuak hizkuntza aldatzeko lista zabalgarriaren aukeratu den hizkuntzaren arabera eguneratuko dira. 4.21 irudian, *PlayerPrefs* klasea nola erabiltzen ikus daiteke. *ChangeLanguage* funtzioa deitzen den bakoitzean, fitxategiko balioa aldatuko da eta horren ondore, aldaketak gordeko dira.

```
//Lengoaia int moduan gorde testu fitxategi batean, eguneratua mantentzeko hurrengo sesio hasieraketetan.
//Hizkuntza aldatzeko aukera sakatzen den bakoitzean eguneratuko da txt fitxategiko hizkuntza balioa.
PlayerPrefs.SetInt("Language", language);
PlayerPrefs.Save();
```

4.21 Irudia: *ChangeLanguage* funtzioaren *PlayerPrefs* klasearen hizkuntza aldagaia.

4.7.10 *HealthBar.cs script*a

*HealthBar.cs script*a, *Canvas*eko jokalariaren interfazearen bizitza-barra GameObjectan dago kokatua. Script honen bidez, bizitza-barraren mugimendua eta jokalariaren bizitzak kontrolatzen direlako.

Hiru funtzio ezberdin dauzka, *Start* gertaera-funtzioa, *GetHp* funtzioa eta *TakeDamage* funtzioa.

Start funtzioan, hasierako bizitzak ezartzen dira, eta bizitza-barra bete egiten da *fillAmount* parametroari ezker.

GetHp funtzioa, funtzio publikoa da (*integer* motako balioa itzultzen du), eta *GameController.cs script*etik deitzeko sortu da, jokalariak unean daukan bizitza kopurua jakiteko.

TakeDamage funtzioa, *GameController.cs script*etik deitzeko funtzio publikoa da (*integer* motako balioa pasatzen da parametro bezala, galduko diren bizitza kopurua), funtzio honen bidez, bizitza bat galtzean, interfazeko bizitza kopuruaren testua eta barra eguneratuko dira.

4.7.11 *DestroyParticleEffect.cs script*a

*DestroyParticleEffect.cs script*a, partikula-efektuak desagertarazteko erabiltzen den script txikia da. Script hau, esan bezala, partikula-efektuko GameObject guztietan dago kokatuta.

Update gertaera-funtzioan dauka inplementazio guztia. Partikula-efektuak desagertarazteko kontagailua 0ra heltzen denean, partikula-efektua suntsitu egiten da.

4.7.12 *QuestionController.cs script*a

*QuestionController.cs script*a, galderak eta erantzunak sortzeaz arduratzen den scripta da. Script honek lau funtzio dauzka: *Start*, *OnEnable*, *MakeQuestionAndAnswers* eta *DestroyBullets* funtzioak.

Start funtzioaren bidez, scriptaren hasieraketak egiten dira, eta jokoaren zailtasuna lortzen da menu nagusitik, galderak jokalariak aukeratu duen zailtasunaren arabera sortzeko. Gainera, *Start* gertaera-funtziotik, galderak eta erantzunak sortzeko funtzioari deitzen zaio.

OnEnable funtzioa, scripta aktibatuko den bakoitzean deitzen da. Script hau, galdera bat egingo den bakoitzean deitzen denez, *OnEnable* funtzioa behar du kontrolatzeko noiz aktibatzen den. *OnEnable* funtzioa, *Start* funtzioa deitzeko erabiltzen da. *Start* funtzioa behin baino gehiagotan deitzeko modua da hau.

MakeQuestionAndAnswers funtzioa, zailtasunaren araberako galdera sortzeko erabiltzen da. Behin galdera sortu dela, erantzunak erdi-aleatorioki sortzen dira, bermatzeko erantzun zuzenetik gertu dauden beste hiru erantzun lortzen direla. Galderak eta erantzunak sortzean, *DianaControl.cs script*ari abisatzen dio sortu diren erantzunen testuak jartzeko.

DestroyBullets funtzioa, unean aktibo dauden jaurtigai guztiak suntsitzeko erabiltzen da. Funtzio hau, akats arraroak saihesteko erabili da. Adibidez, tiro egiten bada, eta itu bat jo baino lehen beste tiro bat eginda; lehen jaurtigaiak itu bat jotzen badu, bigarrenak hurrengo galderaren erantzuna jo lezake nahigabe.

4.7.13 *MusicController.cs script*a

*MusicController.cs script*aren bidez, bideo-jokoaren musika kontrolatzen da. Horretaz aparte, bolumena eraldatzeko audio-nahasgailuak maneiatzeko funtzioak script honetan daude.

Script honek bost funtzio nagusi dauzka: *Awake*, *Update*, *ChangeMusic*, *OnSceneLoaded* eta *VolumeChange* funtzioak.

Awake gertaera-funtzioaren bidez, parametroen hasieraketak egiten dira. Baita scripta daukan GameObjecta eszena-aldaketetan ez suntsitzeko ezartzen da. Horretarako *DontDestroyOnLoad* funtzioa erabili da.

Update funtzioaren bidez, ea eszena-aldaketa bat egon den detektatzen da, musika aldatzeko aktibo dagoen eszenaren arabera. Horretaz aparte, abesti bakoitzaren iraupena kontrolatzen da, amaitzean abestiz aldatzeko.

ChangeMusic funztioaren bidez, abestiak ez errepikatzeko moduan aldatuko dira. Abesti guztiak jo egin direnean bakarrik errepikatu egingo dira. Funtzio hau, abesti bat agortzen den bakoitzean deitzen da.

OnSceneLoaded funtzioa, jokalariak eszenaz aldatzen den bakoitzean deitzen da. Menuko eszena kargatu bada, menuko abestia joko da, bestela ausazko abesti bat joko da.

VolumeChange funtzioaren bidez, jokalariak aukeratu duen bolumenaren arabera erreproduzituko dira soinuak jokoan. Jokalariak menu nagusiko aukeren interfazean, bolumena aldatzeko dagoen barra irristagarriaren bidez aukeratuko du jokoaren bolumena. Bolumenaren balioa *PlayerPrefs* klasearen bidez gordeko da, hurrengo saioetan bolumena gordeta egoteko. Funtzio hau bezalako beste bi funtzio daude inplementatuta script honetan, bata atzealdeko musikaren bolumena kontrolatzeko, eta bestea soinu-efektuen bolumena kontrolatzeko. Bi funtzioek (*BackgroundMusicChange* eta *SoundEffectsChange*), aukeren interfazeko barra irristagarri desberdinen balioak hartzen dituzte baita ere.

4.8 Unity motorraren aldaketak

Atal honetan, proiektua egiteko Unityn egin izan diren aldaketak azalduko dira.

Nagusiki, bi aldaketa egin dira Unity motorean. Lehen aldaketa, argiztapenarekin zerikusia du, eta bigarrena, fisikako motorearekin.

Unityren argiztapena konfiguratzeko, argiztapen-leihoa ireki behar da. Argiztapen-leihoa, Unityren argiztapen-ezaugarriak kontrolatzeko puntu nagusia da. Argiztapen-leihoa irekitzeko, **Window > Rendering > Lighting Settings** menu zabalgarriko bidea aukeratu behar da. Argiztapena leihoa eszenako argia konfiguratzeko, argia optimizatzeko, baita kalitatea, abiadura eta biltegiratze-espazioa lortzeko.

4.22 irudian ikus daitekeen moduan, bi ezaugarri aldatu dira:

- ***Realtime Global Ilumination* ezaugarria.** Ezaugarri hau erabiltzea aukeratzen bada, Unityk denbora errealean argiztapen orokorra egiteko aukera ematen du, Enlighten softwarea erabiliz. Ezaugarri honek arazo bat dauka, argiztapen-mapak egiteko erabitlzen den Enlighten softwarea zaharkitua dagoela jada eta laster kenduko da. Horren ondorioz, ez erabiltzea aukeratu da, nahiko moteltzen duelako proiektua sorketa fasean.

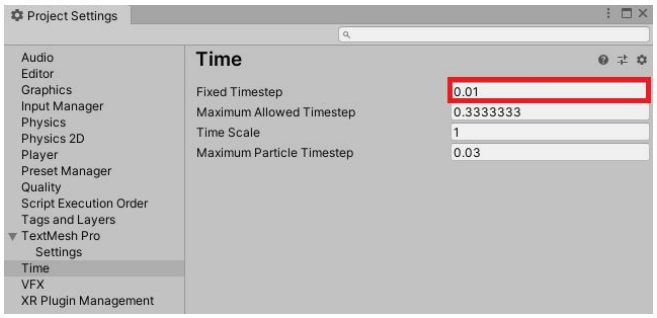
- ***Baked Global Ilumination* ezaugarria.** Ezaugarri hau erabiliz, eszenaren argiztapen-mapa bakarrik *baked* motako edo labekatutako argiak erabiliz sortuko da. Unityk *baked* motako argietarako aurretiko kalkuluak egiten ditu Unityren editorean, eta emaitzak diskoan gordetzen ditu argiztapen-datu gisa. Prozesu horri labekatzea edo

baking deitzen zaio. Exekuzio-denboran, Unityk labeko argiztapen-datuak kargatzen ditu, eta eszena argiztatzeko erabiltzen du. Nahiz eta prozesu hau Unityko editoretik egiten den aldez aurretik (jokoa hasi baino lehen), oso prozesu motela izaten da normalean, eta labekatzea prozesuan dagoen bitartean ezin denez jokoa probatu, ez erabiltzea aukeratu da. Bideo-jokoaren sorkuntza-fasea gehiegi ez moteltzeko.

4.22 Irudia: Unityko argiztapenaren konfigurazioa.

Unityko motorean egon den beste aldaketa, fisikako motorearekin zerikusia daukan ezaugarria da. Ezaugarria aldatzeko, **Edit > Project Settings > Time** menu zabalgarriko bidea aukeratu behar da.

4.23 irudian ikus daitekeen moduan, aldatu den ezaugarria *Fixed Timestep* ezaugarria izan da. Ezaugarri honek, fotograma-tasarekiko (fotogramak segunduko edo *framerate*) independentea den tarte bat adierazten du. Tarte hau, fisikako kalkuluak eta *FixedUpdate* gertaerak zenbat segunduro egingo diren adierazten du (balio lehenetsia 0.02 segundo da). Balio hau erdira txikitu egin da jaurtigaen talkak hobeto detektatzeko. Jaurtigaia azkarregia bada, *collider* bat zeharka lezake talka, fisikako kalkuluak berriz egiteko denbora gehiegi itxaron behar delako.



4.23 Irudia: Unityko proiektuaren konfigurazioa.

4.9 VR egokitzapena

Atal honetan jokoa VRra egokitzeko egin diren aldaketak azalduko dira. Denbora faltagatik ezingo dira VRrako behar diren aldaketa guztiak egin dokumentu hau entregatu baino lehen, hala ere, momenturarte egin izan direnak azalduko dira.

Hurrengo hauek dira egitea lortu izan diren aldaketak:

- **Robota eta kamera nagusia ordezkatu.** Jokoa VRra egokitzeko egin izan den lehen aldaketa, robotaren eta kamera nagusiaren ordezkapena izan da, bi GameObject hauek ezabatu behar izan direlako. VR kaskoen bidez automatikoki egiten da pertsonaiaren biraketa, jokalariak burua mugitzean. Horrez gain, Unityk defektuz daukan kamera nagusia ez du balio VRrako. Kamera nagusia, Oculus *asset*[5]ak dakarren *OVRCameraRig* GameObjectaren bidez ordezkatu behar da. GameObject honek, buruaren mugimendua inplementatuta dakar. VRrako kameran egin behar den aldaketa bakarra, kontrolatzailearen aldaketa da. Kasu honetan, arma errenderizatzeko kontrolatzailearen posizioan. Armari dagozkion aldaketak *Weapon.cs script*ean egin dira.

- ***Weapon.cs script*a aldatu.** Script honetan bi aldaketa nagusi egin dira. Lehena, tiro egiteko botoi sarrera edo *input*a da. VRan ez dagoenez teklatu eta sagurik, sortu diren input guztiak aldatu behar dira. Kasu honetan, Oculus Go kontrolatzaileak daukan katua aukeratu da tiro egiteko input bezala. Bigarren aldaketa, *Raycast*arekin zerikusia dauka. VRko bertsioan arma kontrolatzailearen bidez maneiatzen da, beraz, zeozer behar da nora apuntatzen den jakiteko. Horretarako, *Raycast*a erabili

[5]VR egokitzapena inplementatzeko erabili den *asset*a Oculus Integration 18.0 izan da (Oculus Gorako azkenengo bertsio funtzionala). Nahiz eta bertsio eguneratuagoak dauden, berrienak ezin dira Oculus Go kaskoetarako erabili zaharkituak geratu direnez ez dituzte hauentzako eguneraketa gehiago egin.

izan da. Armak apuntatzen duen puntura (armaren tiro puntutik aurreranzko norabidean) izpi bat botatzen da eta izpiak sortzen duen bide horretan lerro bat errenderizatuko da *LineRenderer* osagaiaren bidez. *LineRenderer* osagaiak bi puntu edo gehiago hartzen ditu 3D espazioan, eta lerro zuzen bat marrazten du bakoitzaren artean. Lerro-errenderizatzailea lerro zuzen sinple batetik espiral konplexu batera edozein gauza marrazteko erabil daiteke.

- ***Canvas*a aldatu.** *Canvas*ean egin den aldaketa nagusia, errenderizazio motan izan da. Jokoaren bertsio normalerako aukeratu izan zen *Canvas*erako errenderizazio mota, estalketa mota izan zen. VRrako ezin da errenderizazio mota hori erabili, kaskoek bi kamera erabiltzen dituztelako; bat begi bakoitzerako. Beraz, ezin da kamera bakarrerako errenderizazioa erabili. Horren ordez, munduko espazio (*World Space*) motako errenderizazioa erabiltzea aukeratu da. Errenderizazio mota honetan, *Canvas*ak kokapen fisikoa izango du, 3D posizioa emanda, eta eszenako beste edozein objektuk bezala jokatuko du. UIko elementuek aukeratu den 3D kokapenaren arabera eszenako beste objektu batzuen aurrean edo atzean errenderizatuko dira.

  Errenderizazio mota hau erabiltzean eta UIarekin elkarreragiteko sagua ez dagoela ikusita, UIko elementuetan hainbat aldaketa egin behar dira.

- **UIko botoien aldaketa.** Botoiekin elkarreragiteko, botoietan *Box Collider* osagaia sortu da. Horren bidez, armak botatzen duen izpiarekin kontrola daiteke noiz jotzen ari den UIko elementuren bat. UIarekin elkarreragiteko nahiko metodo sinplea sortu da, hala ere, denbora badago, hobetzen saiatuko da. Gerta daitekeelako *Canvas*eko elementu guztiekin ez funtzionatzea, adibidez *Dropdown* menu zabalgarriekin edota barra irristagarriekin (*Slider*).

5. KAPITULUA

Ondorioak eta etorkizuneko lana

Kapitulu honetan, proiektuari buruz atera diren ondorioak ikusiko dira. Horretaz gain, proiektu honek izan ditzazkeen hobekuntzak edo etorkizunean egin daitezkeen lanak ere azalduko dira.

5.1 Ondorioak

Proiektua garatzeko helburu nagusia, umeentzako joko hezitzailea garatzea izan da. Garapen hori aurrera eramateko, Unity motorra erabiltzea aukeratu da.

Behin jokoa garatzeko motorra aukeratu zela, jokoa nolakoa izango zen pentsatu behar zen. Jokoa definitzeko, *Game Design Document*a sortu zen. Proiektua amaitzerakoan *Game Design Document*ean ezarri ziren iterazioen puntu guztiak betetzea zen helburua.

Bideo-jokoaren garapenaren hasieran jokoaren atal grafikoa sortu zen. Behin jokoaren oinarria sortuta zegoela, jokoaren script bidezko funtzionamenduak jarraitu zion. Horren ostean, jokoaren interfazeak eta hauek osatzeko scriptak sortu ziren. Azkenik, osagai gehigarriak sortu ziren hauen scriptekin batera, hala nola, partikula-efektuak, soinu-efektuak eta azken momentuko ukituak.

Behin proiektuaren garapena amaitu dela eta atzera begiratuz, hasiera batean *Game Design Document*eko 3.3 atalean sortu ziren puntu edo helburuak bete diren egiaztatuko da.

Iterazioen deskribapena atalean definitutako lehen iterazioko funtzionalitate guztiak inplementatu izan dira jokoan. Lehen iterazioan definitutako helburuak garrantzitsuenak eta

denbora gehien behar izan dituztenak dira. Azken batean, jokoa behar zituen oinarrizko elementu eta funtzionalitate guztiak sortu dira.

Bigarren iterazioa, iterazio gehigarria zen. Iterazio honetan funtzionalitate gehigarriak inplementatuko ziren, adibidez, jokoaren bizi-kalitatea (ingelesezko *Quality of life* terminotik) hobetzeko. Aldaketa hauen bidez jokoaren erabilgarritasuna hobetu egiten da.

Bigarren iterazioko helburuak banaka aztertuko dira:

- **Geroz eta denbora gutxiago galderak erantzuteko.** Ez da inplementatu. Nahiz eta ez den oso helburu konplexua inplementatzeko, ez zaio lehentasuna eman puntu honi, ez baitu hainbesteko garrantziarik. Ez inplementatzearen arrazoia, denbora falta izan da.
- **Ituen mugimendua.** Inplementatu egin da, hala ere, beti egin daitezke hobekuntzak algoritmoan adibidez, mugimendua leuntzeko edota talkak hobeto kontrolatzeko.
- **Ituen mugimendua gero eta azkarragoa.** Ez da inplementatu. Helburu hau teorian itxura ona izan dezakeen ideia da, baina inplementatzeko orduan ez da hain komenigarria. Arazoak sortu ditzazke, adibidez, ituak azkarregiak izatea jaurtigaiekin jo egiteko.
- **Maila bakoitzerako arma desberdina.** Ez da inplementatu. Honen arrazoi nagusia, denbora falta izan da. Funtzio gehigarrien artean ez zuen beste batzuk bezainbesteko garrantzia.
- **Menu nagusi osoagoa.** Menu nahiko osotua sortu da, hala ere, beti daude hobetu daitezkeen gauzak. Jokalariari aukera gehiago eskaini ahal zaizkio, adibidez, jokoaren bereizmena aldatzeko, pantailaren distira edota koloreak aldatzeko (norbait daltonikoa bada), etab.
- **VR egokitzapena.** Egon izan den pandemia dela eta, VR egokitzapena egiteko gela itxita egon da. Horren ondorioz, irailaren hasieran hasi izan da VR egokitzapena. Nahiz eta funtzionalitate basikoenak egokitzea lortu izan diren, aste bat bakarrik izanda ezin izan dira funtzionalitate guztiak inplementatu.

Iterazioen helburuak ikusita, funtzionalitate gehienak inplementatu direla ikus daiteke. Hala ere, proiektu honen denbora mugatua denez, inplementatu gabe geratu diren funtzionalitateak etorkizuneko lanerako utzi beharko dira, beti ere, jokoarentzako hobekuntzak badira.

5.2 Etorkizunerako lana

Zaila da jakitea noiz amaitzen den bideo-joko baten garapena. Joko batek beti zeozer berria garatzeko edo jada inplementatuta daukan zeozer hobetzeko izan dezake. Horretaz gain, jokoak mantenimendu konstantea izan dezake jokalariek aurkitzen dituzten akatsak konpontzeko.

Etorkizunean hurrengo puntuen inplementazioa egin daiteke:

- **Inplementatu ez diren helburuak amaitzea.** Esan bezala, inplementatu gabe geratu diren helburuak amaitzea izango litzateke etorkizunerako geratuko litzatekeen lan bat.
- **Jokoaren funtzionamendua hedatu.** Joko hezitzailea denez, beti egon daiteke funtzionalitate berriak gehitzeko aukera. Adibide bat, jokoa biderketak egiteko bakarrik ez izatea izango litzateke gehiketak, kenketak edota zatiketak gehituz. Gainera, matematikak ez ezik, jokoa beste arloetara heda zitekeen hainbat gai desberdinei buruz ikasteko aukera emateko. Adibidez, ituen testuetan balio numerikoak agertu ordez, lurraldeetako irudiak edota lurraldeen izenak jarriz geografia ikasi lezakete edota hizkuntzak, ituetan hitz egokiak edota letra egokiak hautatzeko aukerak eskainiz.
- **Bateragarritasuna beste VR kaskoekin.** Etorkizunerako beste helburu bat, VR kaskoen bateragarritasuna izan daiteke. Oculus Go kaskoetaz aparte, beste batzuekin maneiatu ahal izateko, adibidez, Oculus Quest edota Samsung VR kaskoentzako bertsio bat garatuz.

Eranskinak

A. ERANSKINA

Unity eta Oinarri Teorikoak

A.1 Unity

Unity bideo-jokoen motor multiplataforma da, Unity Technologies konpainiagatik sortua. Unity garapen-plataforma gisa erabil daiteke Microsoft Windows, Mac OS eta Linuxen.

Proiektu honetan, Microsoft Windows erabiliko da bideo-jokoaren garapenerako.

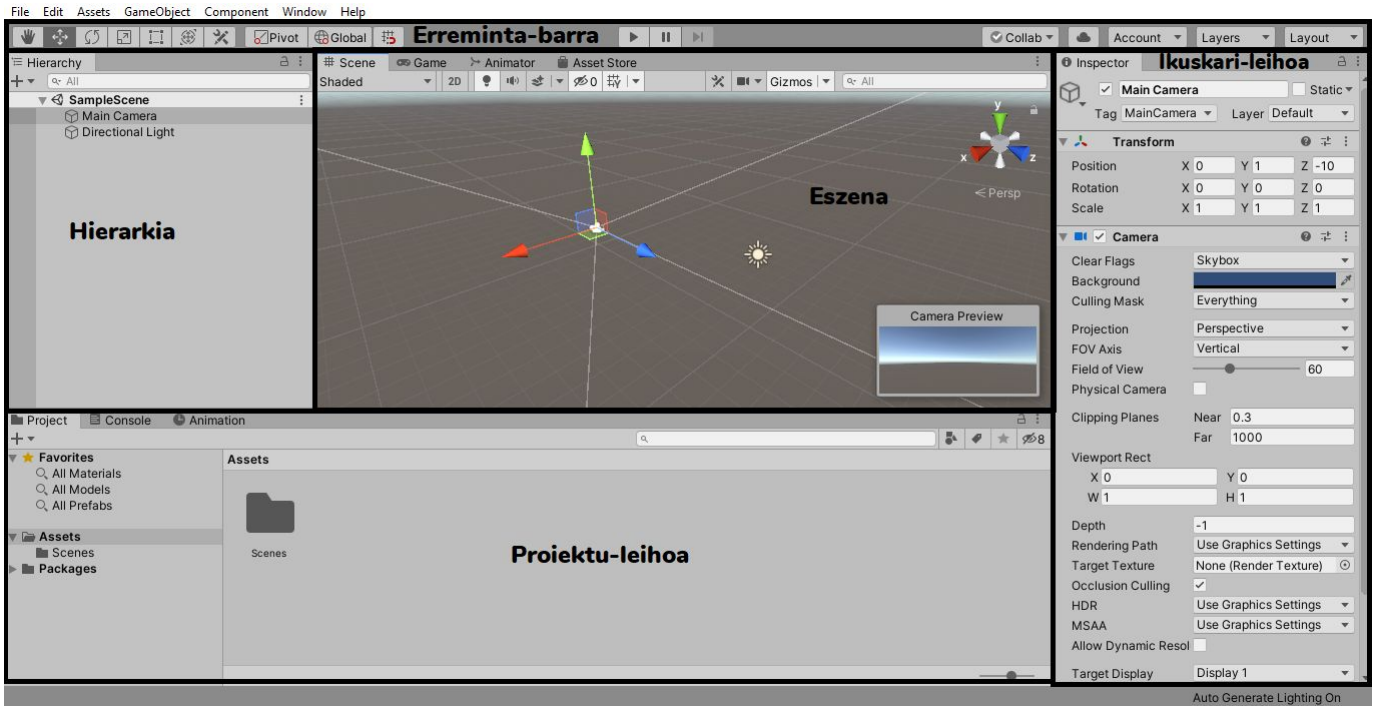
A.1.1 Oinarrizko funtzionalitateak

Unityk eta beste hainbat bideo-joko motorrek hurrengo oinarrizko funtzionalitateak dituzte: errenderizazio-motor bat (errenderizatzailea) 2D edo 3D grafikoetarako, fisika-motor bat edo talka-detekzio (eta talka erantzuna) sistema bat eta soinua, *scripting*a, animazioa, adimen artifiziala eta eszena-grafikoak sortzeko gaitasuna.

A.1.2 Unity interfazea

Unityn 3D proiektu bat hastean, A.1 irudian ikus daitekeen interfazea agertzen da.

Unityren interfazea bost atal nagusitan banatzen da (goiko barra kenduta, aukera ezberdinak hautatzeko barra), irudian ikus daitekeen moduan. Atal nagusiak hurrengoak dira:

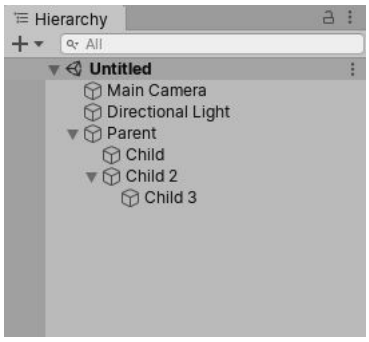


A.1 Irudia: Unityren 3D proiektuetarako Interfaze Grafikoa.

1. **Hierarkiako leihoa.** Hierarkiako leihoak uneko eszenaren GameObject guztien zerrenda du. Hierarkiak GameObjectak elkarrekin nola lotzen diren erakusten du. Unityk *parenting* edo gurasotasun kontzeptua erabiltzen du. GameObject talde bat sortzen denean GameObject altuenak edo eszenak, "guraso" du izena, eta azpian bildutako GameObjects guztiak "umeak" dira. GameObject nagusiaren (eszena) barnean beste guraso-ume loturak egon daitezke. A.2 irudian, Untitled izena duen GameObjecta, eszena da, beste GameObject guztien gurasoa. Barnean beste guraso-ume lotura dago, Parent GameObjecta beste Child GameObjectekin, baita Child 2 Child 3rekin.

2. **Erreminta-barra.** Erreminta-barrak oinarrizko lan-ezaugarrietarako sarbidea eskaintzen du. A.1 irudian ikus daitekeen moduan, erreminta-barraren ezkerrean, eszena ikuspegia eta haren barruko GameObjectak manipulatzeko oinarrizko tresnak (objektuak mugitu, biratu, eskalatu...) daude. Erdian, ezagunak izan daitezkeen *Play*, *Pause* eta *Step* botoiak daude *Game* ikuspegian erabiltzeko. Eskuineko botoiak kontura sartzeko, hodei-zerbitzura sartzeko, editorearen diseinua pertsonalizatzeko, etab. aukera ematen dute.

3. **Eszena-leihoa.** Eszena bisualki nabigatu eta editatzeko erabiltzen da. Sortzen ari



A.2 Irudia: GameObjecten hierarkia.

den munduarekiko ikuspegi interaktiboa da. Eszena-leihoa erabil daiteke GameObjectak kokatzeko, eraldatzeko, mugitzeko, etab. Eszenaren ikuspegia proiekzio ortografikoan edota perspektiban ikus daiteke, unean komeni denaren arabera. *Game* ikuspegia, eszenan dauden kameren bidez, jokoa nola ikusiko den simulatzen du. Play botoia sakatzean simulazioa hasten da.

4. **Ikuskari-leihoa.** Ikuskari-leihoak hautatutako GameObjecten propietate guztiak ikusi eta editazeko aukera ematen du. Ikuskari-leihoaren diseinua eta edukia aldatu egiten dira hautatuta dagoen GameObjectaren arabera, GameObject bakoitzak osagai desberdinak dituelako.

5. **Proiektuko leihoa.** Proiektuan erabiltzeko moduan dauden asset[5] guztien liburutegia erakusten du. Proiektuan assetak inportatzen direnean, leiho honetan agertzen dira.

A.1.3 Ordenagailu bidezko grafikoen hainbat kontzeptu

Ordenagailu bidezko grafikoak informatikaren adarretako bat da, non ordenagailuak erabiltzen diren bai irudi bisualak sintetikoki sortzeko, baita mundu errealeko informazio bisuala eta espaziala integratzeko edo aldatzeko.

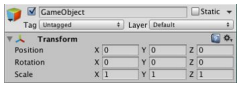
Proiektuaren garapeneko atalean ordenagailu bidezko grafikoen hainbat kontzeptu aterako dira. Horregatik, ordenagailu bidezko grafikoetan erabiltzen diren eta Unity bideo-jokoen motorrean ohikoak diren hainbat termino edo kontzeptu azalduko dira.

Unity bideo-joko motorrean erabiltzen diren ohiko kontzeptuak:

- ***GameObject.*** GameObjectak funtsezko objektuak dira Unityn, eta pertsonaiak, osa-

garriak eta eszenatokia irudikatzen dituzte. Objektu hauek ez dute ezer lortzen beren kabuz, baina edukiontzi gisa funtzionatzen dute *components*entzat[1]. Osagai hauek GameObjectaren benetako funtzionaltasuna ezartzen dute.

Adibide bezala, Transformazio osagaia ikus daiteke A.3 irudian.

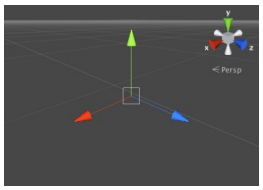


A.3 Irudia: GameObject baten Transformazio osagaia.

- ***Transform* osagaia.** GameObject baten osagaiak azaltzeko adibide bezala erabili da, transformazio osagaia, GameObject **guztiek** daukaten osagairik garrantzitsuena eta gehien aldatzen dena delako. Transformazio osagaia GameObjecten posizioa, orientazioa eta eskala definitzen ditu jokoaren munduan eta eszena ikuspegian. Osagai hau ezin da GameObject batetik ezabatu. Transformazio osagaia hierarkia edo guraso-sistema ere onartzen du, hau da, GameObject bat beste GameObject baten semea izan daiteke, gurasoari ezartzen zaizkion *Transform* osagaiko aldaketa berak ezarriko zaizkio semeari ere. Beste modu batean esanda, semea den GameObjecta, gurasoaren *Transform* osagaiko aldaketen menpean dago. Unityren GameObjectekin lan egiteko funtsezko atala da hau.

  Transformazioak 3D espazioan definitzen dira X, Y eta Z ardatzen bidez. Unityn, ardatz horiek kolore gorriz, berdez eta urdinez adierazten dira, hurrenez hurren.

  A.4 irudian ikus daiteke nola dauden kokatuta Unityk erabiltzen dituen ardatzak.



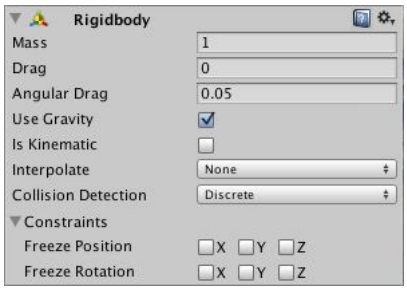
A.4 Irudia: Unityren ardatzak.

- ***Rigidbody*.** *Rigidbody*a, GameObject batek izan dezakeen osagaia da. Osagai honen bidez, fisikaren kontrolpean jarduteko aukera ematen zaio GameObjectari. Rigidbody osagaiari esker objektuak indarra eta torkea har ditzakete modu errealistan mugi daitezen. Rigidbody osagaia duen GameObject batean grabitateak eragina

[1] ***Component*** edo **osagai** bat, GameObject batean lotuta doan edozein osagaia da. Osagai hauek, GameObject horren portaera definitzen dute. Adibidez, errenderizatzeko metodoak, *script*ak, *Collider*rak, etab.

izan dezake, *scripting* bidezko indar gehigarrien menpean jardun dezake, edo NVIDIA Physx fisikako motorraren bidez beste objektu batzuekin elkarreraginean ari daiteke.

A.5 irudian ikus daiteke Rigidbody osagaiak dauzkan aukerak.



A.5 Irudia: GameObject baten Rigidbody osagaia.

- ***Collider* osagaia.** *Collider* osagaiek objektu baten forma definitzen dute fisika motorraren barruan. GameObject batek, collider osagai bat baino gehiago izan ditzake. Colliderrak ikusezina dira, eta forma geometriko ezberdinak izan dezakete (collider sinpleenak, primitiboak deitzen dira, eta kutxa, esfera edota kapsula forma izan dezakete). Collider batek ez du objektuaren forma zehatza izan behar, eta hain zuzen, hurbilketa bat askotan eraginkorragoa izan daiteke.

- ***Raycasting*.** *Raycast*[2] bat, funtsean, 3D edo 2D espazioko posizio batetik igortzen den izpia da, eta norabide jakin batean mugitzen da. *Raycasting*a bideo-jokoen garapenean erabili ohi da, besteak beste, jokalariaren edo IAren ikusmen-lerroa zehazteko, jaurtigai bat nora bota jakiteko, UIko elementuekin elkarreragiteko, etab.

- ***Sprite*.** *Sprite*ak 2D objektu grafikoak dira eta estatikoak edo animazioak izan daitezke. Unityko proiektua 2Dan bada, inportatzen den irudia automatikoki ezartzen da Sprite gisa. Aldiz, 3D proiektu batean, inportatzen diren irudiak testura gisa konfiguratzen dira. Irudi bat UIan erabili nahi bada, bere konfigurazioa aldatu behar da, Sprite motakoa izateko. A.6 irudian ikus daiteke irudi baten konfigurazio aukerak. Testura mota Sprite motara ezarrita, balio lehenetsiaren ordez.

- ***Canvas*a.** *Canvas*a, Canvas osagaia duen GameObject bat da. UI (erabiltzaile-interfaze) elementu guztiak Canvas objektuaren barnean egongo dira derrigorrez, hau da, UI elementu guztiek Canvasaren semeak izan behar dira. UI elementu berri bat sortuz,

[2]Raycast, ingelesezko bi hitz elkartuz lortzen den terminoa da, Ray edo izpia eta cast edo jaurtiketa.

A.6 Irudia: Irudi baten konfigurazioa aukerak.

adibidez irudi bat, **GameObject > UI > Image** menu zabalgarria erabiliz, automatikoki Canvas bat sortuko da eszenan dagoeneko bat ez badago. Irudia Canvasaren seme bezala sortuko da. Canvas eremua laukizuzen itxurarekin agertzen da eszena ikuspegian, errazago egiteko UI elementuen kokapena eszenatik. Canvasek *EventSystem* GameObjecta erabiltzen dute mezu-sistemarekin laguntzeko.

- ***EventSystem.*** *EventSystem*a *input*ean (teklatua, sagua, ukimena edo best input pertsonalizatua) oinarritutako aplikazioko objektuetara gertaerak bidaltzeko modu bat da. EventSystema osagai gutxi batzuek osatzen dute, eta elkarrekin funtzionatzen dute gertaerak bidaltzeko. Canvas batean UIko elementu interaktiboak funtziona dezaten beharrezko da EventSystem GameObjecta.

Ordenagailu bidezko grafikoetan erabiltzen diren hainbat termino:

- **Erabiltzaile-interfazea.** Erabiltzaile-interfazea (ingelesezko *User Interface* terminotik) gailu bateko giza eta ordenagailuen arteko elkarreragin- eta komunikazio-puntua da. Erabiltzaile batek aplikazio edo webgune batekin elkarreraginean jarduteko modua ere bada. Bideo-jokoetan erabiltzaile-interfazeak menu nagusiak, jokalariaren bizitza-barrak, inbentarioak, testuak, etab. osatzen dute.

- ***Rendering*** **edo Irudi-sintetisazioa.** *Rendering*a, 2D edo 3D eszena baten landu gabeko informazioa (poligonoak, materialak eta argiztapena) hartu eta azken emaitzaren irudia kalkulatzen duen prozesua da. Bi motako errenderizazioa dago, aldez aurretiko errenderizazioa eta denbora errealeko errenderizazioa.

  – Aldez aurretiko errenderizazioa, kalitate hobeagoko edo errealistagoak diren irudiak sortzeko erabiltzen da, normalean filmetarako erabiltzen da errenderi-

zazio mota hau. PUZ (Prozesatzeko Unitate Zentrala) bidezko errenderizazio bezala ezagutzen da.

- Denbora errealeko errenderizazioa, aldiz, 3D bideo-jokoetan erabiltzen da gehiago, jokalariarekin elkarreragin handia izatea eskatzen delako. GPU (Grafikoak Prozesatzeko Unitatea) bidezko errenderizazio bezala ezagutzen da.

- ***Ray tracing* edo Izpi-marraketa.** *Ray Tracing* bideo-jokoen argitasuna, itzal eta islapena hobetzeaz arduratzen den algoritmoa da. Argi-izpi errealak simulatuz funtzionatzen du, argi-sorta batek mundu fisikoan hartuko lukeen bidea trazatzeko algoritmo bat erabiliz. Ray Tracing algoritmoa, Raycastean (Raycastak jokoetatik kanpo hainbat erabilera izan ditzazke ere, adibidez, robotek talkak saihesteko erabil daiteke [Sauzé and Neal, 2010]) oinarritzen da.

- **Ordenagailu bidezko animazioa.** Ordenagailu bidezko animazioa digitalki animatutako irudiak sortzeko erabiltzen den prozesua da. 3D animazioetarako, objektuak (ereduak) ordenagailuko monitorean (modelatua) eraikitzen dira, eta normalean *rig* bezala ezagutzen den eskeleto birtual batekin manipulatzen dira.

- ***Shader* edo Itzal-sorketa.** Shader bat ordenagailu-programa mota bat da, 3D eszenetan itzala sortzeko erabiltzen dena (irudi errenderizatu batean argiaren, iluntasunaren eta kolorearen maila egokiak sortuz). Gaur egun beste erabilera batzuk izan dezakete, adibidez, irudietan efektu bereziak sortzeko (filtroak) edota beroak sortzen duen distortsioa simulatzeko [St-Laurent, 2004].

B. ERANSKINA

Erabilitako materialak

B.1 Materialak

Atal honetan, proiektua garatzeko erabili diren materialak zerrendatuko dira.

Erabilitako grafikoak:

- Robota.

  https://www.turbosquid.com/FullPreview/Index.cfm/ID/1154074

- Arma.

  https://free3d.com/3d-model/pistola-lser-lowpoly-136544.html

Erabilitako audioak:

- Kurtsorea botoien gainetik pasatzeko soinu-efektua.

  https://freesound.org/people/GameDevC/sounds/422836/

- Armaren tiro soinu-efektua.

  https://freesound.org/people/Smoker858/sounds/508854/

- Jaurtigaiaren talka soinua.

  https://freesound.org/people/Filmscore/sounds/515866/

- Galdera eta erantzunak zoriz aukeratzearen soinu-efektua.

  https://freesound.org/people/Breviceps/sounds/447918/

- Ibaiaren soinu-efektua.

  https://freesound.org/people/BurghRecords/sounds/415151/

- Erantzun zuzenaren partikula-efektuentzako soinu-efektua.

  https://freesound.org/people/newagesoup/sounds/344534/

- Alexander Nakaradaren *Nowhere Land* abestia.

  https://www.youtube.com/watch?v=FtlMGJ25x6I

- *Safe haven* abestia.

  https://www.youtube.com/watch?v=GNZkLtNz8WU

- *Glimpse of Eternity, Surreal Forest, Rain* eta *Contemplate the stars* abestiak.

  https://freemusicarchive.org/music/Meydan

- *Ambient Nature Music - The Forest Awakens* abestia.

  https://soundcloud.com/moonchant-music/ambient-nature-music-the-forest-awakens

  https://www.youtube.com/user/moonchantmusic

Erabilitako assetak:

- *Low Poly Pack*.

  https://assetstore.unity.com/packages/3d/environments/low-poly-pack-94605

- *Low-Poly Simple Nature Pack*.

  https://assetstore.unity.com/packages/3d/environments/landscapes/low-poly-simple-nature-pack-162153

- *Snowy Low-Poly Trees*.

  https://assetstore.unity.com/packages/3d/vegetation/trees/snowy-low-poly-trees-76796

- *UI Sound Effects Collection Pack 2: Buttons*.

  https://assetstore.unity.com/packages/audio/sound-fx/ui-sound-effects-collection-pack-2-buttons-27803

- *Standard Assets (for Unity 2018.4)*

  https://assetstore.unity.com/packages/essentials/asset-packs/standard-assets-for-unity-2018-4-32351

Bibliografia

[Foxman, 2019] Foxman, M. (2019). United we stand: Platforms, tools and innovation with the unity game engine. *Social Media + Society*, 5(4).

[Hamari et al., 2014] Hamari, J., Koivisto, J., and Sarsa, H. (2014). Does gamification work? — a literature review of empirical studies on gamification. In *2014 47th Hawaii international conference on system sciences*, pages 3025–3034. Ieee.

[Hamari et al., 2016] Hamari, J., Shernoff, D., Rowe, E., Coller, B., Asbell-Clarke, J., and Edwards, T. (2016). Challenging games help students learn: An empirical study on engagement, flow and immersion in game-based learning. *Computers in Human Behavior*, 54:170–179.

[Kiryakova et al., 2014] Kiryakova, G., Angelova, N., and Yordanova, L. (2014). Gamification in education. Proceedings of 9th International Balkan Education and Science Conference.

[Mikołajczyk, 2019] Mikołajczyk, K. (2019). Vr in education - a subjective overview of the possibilities. *e-mentor*, 79(2):33–40.

[Sauzé and Neal, 2010] Sauzé, C. and Neal, M. (2010). A raycast approach to collision avoidance in sailing robots. pages 25–32.

[Šmíd, 2017] Šmíd, A. (2017). Comparison of unity and unreal engine. *Czech Technical University in Prague*, pages 41–61.

[St-Laurent, 2004] St-Laurent, S. (2004). *Shaders for game programmers and artists*. Cengage Learning.