UNIVERSITAT DE BARCELONA

Treball final de grau

DOBLE GRAU EN MATEMÀTIQUES I ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Videojoc 3D per al reconeixement i construcció de figures geomètriques

Autor: Oriol Sors Vidal

Directora: Dra. Anna Puig
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 24 de gener de 2022

Abstract

The ultimate goal of this final project is to create a 3D video game that serves as a complementary tool in learning the geometry of space in secondary school.

In this sense, the aim is for students to obtain a series of pieces necessary to be able to assemble, in the end, a three-dimensional construction that has been previously designed by a teacher. During this process, they must be able to solve challenges and geometric issues also designated by teachers, which involve the recognition of solids and their fundamental characteristics.

Therefore, a close communication between student-teacher is pursued that allows the adaptability of the game according to the needs of the students. In any case, this dialogue should not be restricted exclusively to the classroom, but at all times the evolution of the student can be followed.

The development of the video game has also required an exploration of certain mathematical concepts, which have been necessary to overcome non-trivial implementation problems that have arisen throughout working on some aspects of functionality.

Resum

La finalitat màxima d'aquest Treball Final de Grau és crear un videojoc 3D que serveixi com a eina complementària en l'aprenentatge de la geometria de l'espai a Secundària.

En aquest sentit, es busca que l'alumnat obtingui un seguit de peces necessàries per a poder muntar, al final, una construcció tridimensional que ha dissenyat prèviament un/a professor/a. Durant aquest procés, ha de ser capaç de resoldre reptes i qüestions geomètriques designades també pel professorat, les quals involucren el reconeixement de sòlids i les seves característiques fonamentals.

Per tant, es persegueix una comunicació estreta entre estudiant-professor que permeti l'adaptabilitat del joc segons les necessitats de l'alumnat. En qualsevol cas, aquest diàleg no ha de restringir-se exclusivament a l'aula, sinó que en tot moment es pot seguir l'evolució de l'estudiant.

El desenvolupament del videojoc ha requerit també una exploració de certs conceptes matemàtics, els quals han estat necessaris per superar problemes no trivials d'implementació que han aparegut al llarg de treballar alguns aspectes de la funcionalitat.

2020 Mathematics Subject Classification. 11R52, 05C85, 51M20

Agraïments

Vull agrair a Anna Puig per tot el compromís mostrat durant el curs, per la motivació que sempre ha mostrat i per la voluntat de millorar el projecte sempre que ha sigut possible.

Vull agrair també a la meva família, amics i a la meva parella per haver-me acompanyat durant el temps que li he dedicat al treball.

Taula de continguts

Capítol 1

Introducció

L'ansietat matemàtica és un terme que cada vegada es troba més present a la societat [24]. No són pocs els estudiants de Secundària que afirmen que l'assignatura de matemàtiques és la més difícil i/o la que menys els hi agrada, fet que provoca més suspensos i, en conseqüència, causa encara més pànic i angoixa que en un inici. És un circuit del qual és difícil sortir, ja que l'alumne acaba per no entendre els conceptes i es limita únicament a aprendre unes instruccions que li garanteixin l'aprovat.

Amb aquest projecte es vol potenciar el coneixement de geometria en tres dimensions del primer Cicle de Secundària i 4t de la ESO, exercitant de manera dinàmica la percepció de l'espai de l'alumne i consolidant els coneixements exposats a classe. Tot això es vol assolir a partir d'un enfoc molt jugable, on l'estudiant estigui realment entretingut i recuperi la voluntat d'aprendre matemàtiques.

El/la professor/a és una peça fonamental en l'aprenentatge i, per tant, ha de ser capaç d'adaptar-se a les necessitats de cada alumne. Això mitjançant mètodes tradicionals és logísticament impensable, però amb les noves eines tecnològiques com la que es planteja en aquest projecte és força viable. D'aquesta manera, s'aconsegueix fer un seguiment de cada estudiant i entendre quines necessitats té cadascú.

Per tant, es vol evidenciar que l'ús dels videojocs va molt més enllà del que es pot concebre en un inici, i és imprescindible aprofitar aquesta tecnologia per millorar tots aquells àmbits que requereixen un canvi urgent, com és el cas de l'educació matemàtica.

1.1 Context

En aquest treball es busca desenvolupar un joc seriós amb una forta component de gamificació per tal que els alumnes puguin abordar la geometria de l'espai de manera més dinàmica i entretinguda. Consisteix, doncs, en un videojoc 3D que permeti consolidar els conceptes teòrics i a la vegada que reforci l'orientació espacial de l'estudiant.

El disseny del codi es fonamenta en els continguts adquirits al llarg de les assignatures de Programació II i Disseny de Software del grau d'Enginyeria Informàtica. En la mateixa línia, per tal d'assolir alguns objectius cal implementar algorismes treballats a Algorísmica Avançada i Estructures de Dades i, finalment, es fa ús de les metodologies *Agile* vistes a Enginyeria del Software per a planificar i desenvolupar el producte.

L'assignatura de Gràfics i Visualització de Dades desperta la majoria de fonaments matemàtics que hi ha darrere de la implementació d'un videojoc i que es detallen al llarg del segon capítol d'aquest treball. En aquest sentit, les assignatures del grau de Matemàtiques que han permès aprofundir-hi són Geometria Lineal, Topologia i Geometria Global de Superfícies, i Estructures Algebraiques.

1.2 Motivació

La motivació d'aquest projecte és aportar al món de l'educació matemàtica una eina de gamificació que permeti a l'alumnat de Secundària aprofundir en els coneixements de la geometria tridimensional. En aquest sentit, es busca que el professorat pugui participar activament en el procés d'aprenentatge dels estudiants i sigui capaç de veure la progressió que tenen.

A l'actualitat cada vegada és més fàcil que els conceptes adquirits a classe es consolidin a casa. Seguint aquesta direcció, neix també la necessitat de que el videojoc sigui jugable als ordinadors o dispositius portàtils personals dels alumnes i no només a les aules de l'escola.

1.3 Objectius

L'objectiu fonamental és crear un videojoc 3D que permeti als estudiants treballar el reconeixement de figures geomètriques i a la vegada entrenar la seva percepció espacial al construir-ne. El professor ha de poder editar completament la partida de l'estudiant i assignar-li missions que involucrin reptes personalitzats, així com poder fer una supervisió de la seva evolució. Per tant, l'objectiu del projecte queda descomposat com:

1. Disseny i desenvolupament d'un constructor de figures 3D:
   (a) Desenvolupar i gestionar el moviment de la càmera.
   (b) Desenvolupar un sistema que comprovi si una figura és vàlida segons el seu posicionament.
   (c) Desenvolupar una interfície que permeti assignar missions als alumnes.
   (d) Desenvolupar un sistema que comprovi si la figura construïda és la correcta.
2. Disseny i desenvolupament dels escenaris jugables:
   (a) Dissenyar les interfícies de les escenes del joc.

(b) Dissenyar les mecàniques de cada component jugable.

(c) Dissenyar una interfície que connecti les diverses escenes del joc.

3. Disseny i desenvolupament d'un sistema de persistència de dades:

   (a) Desenvolupar un sistema d'autenticació d'usuaris.

   (b) Dissenyar l'estructura de dades serialitzables del videojoc.

   (c) Desenvolupar un sistema de comunicació a temps real entre professor i alumne.

1.4 Planificació

S'ha planificat el projecte a 24 setmanes, tenint en compte que la primera reunió es va produir al juliol. Es requeria una formació prèvia en el motor gràfic Unity i es van realitzar 2 cursos durant el mes d'agost per adquirir els coneixements bàsics. A partir d'aquí, el disseny i el desenvolupament s'organitzen mitjançant *sprints* d'una o dues setmanes segons s'ha convingut a les reunions. Es veu de manera general a la Figura 1.1.

Mesos	Agost				Setembre				Octubre				Novembre				Desembre				Gener			
Tasca/Setmana	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Formació prèvia																								
Disseny del joc																								
Desenvolupament																								
Testing																								
Documentació																								

Figure 1.1: Planificació temporal del projecte

.

1.5 Organització de la memòria

L'estructura d'aquesta memòria es desglossa en els següents capítols:

1. **Introducció**: on es presenta el projecte i s'exposen els objectius.
2. **Formalització del problema**: on es treballen els fonaments matemàtics necessaris per al desenvolupament.
3. **Disseny de l'aplicació**: on s'exploren els requeriments, els treballs relacionats amb el projecte i es proposa el disseny del videojoc.
4. **Disseny i desenvolupament del software**: on s'analitza l'arquitectura usada, la persistència i els algorismes principals.
5. **Simulacions i resultats**: on es mostren els resultats visuals i d'aprenentatge.
6. **Conclusions i treball futur**: on es detallen els objectius satisfets i la perspectiva de futur del projecte.

Capítol 2

Formalització del problema

En aquest capítol es presenten els fonaments matemàtics en els que es basen els diferents aspectes del joc de construcció de sòlids 3D a partir de peces bàsiques.

En primer lloc, s'introdueixen definicions elementals de sòlids a partir de les quals es definirà l'objecte 3D a construir. Seguidament, s'explicarà com s'hi associen els grafs posicionals i com es poden visualitzar aquestes construccions mitjançant diverses rotacions. Finalment, es farà una exposició dels atributs de la càmera en una escena 3D i es mostraran tècniques per a poder visualitzar correctament la construcció.

2.1 Sòlids geomètrics

S'exposen a continuació un seguit de definicions i propietats geomètriques que, adaptades a un nivell de secundària, apareixen dins el joc i són una eina que pot usar el professor per a potenciar individualment cada alumne segons el que cregui més convenient.

Considerem $\mathbb{R}^3$ amb la mètrica euclidiana.

Definició 2.1. Diem que $S \subset \mathbb{R}^3$ és un **sòlid geomètric** si $\forall x \in S$ i $\forall V_{x,r} \subset S$ veïnat respecte S tal que $V_{x,r} = \{y \in S \mid \|y - x\| < r\}$, $\exists p, d \in \mathbb{R}$ tals que la bola oberta $B(p,d) = \{q \in \mathbb{R}^3 \mid \|q - p\| < d\} \subset V_{x,r}$. En altres paraules, $S = \overline{\mathring{S}}$, és a dir, que S és la clausura de l'interior de S. A topologia, diem que S és un conjunt regular tancat [17].

Dins d'aquests sòlids es distingeixen fonamentalment dues classes: els políedres i els sòlids de revolució. Tot i així, el segon tipus no queda contemplat en aquest projecte i, per tant, es detallen aquelles característiques que es consideren necessàries dels primers.

Es pot entendre un políedre desde una perspectiva més geomètrica:

Definició 2.2. Sigui $P \subset \mathbb{R}^3$ un sòlid geomètric i $A_k x + B_k y + C_k z + D_k = 0$, $k \in \{1, ..., F\}$ plans de l'espai amb $F \in \mathbb{N}$ finit. Diem que P és un **políedre** si

$$P = \{(x,y,z) \in \mathbb{R}^3 \mid A_k x + B_k y + C_k z + D_k \leq 0, \forall k \in \{1, ..., F\}\} \text{ és acotat.}$$

La generalització d'aquesta definició a un sistema d'hiperplans de dimensió d s'anomena **polítop** d-dimensional [5].

Les parts del políedre incloses a cada pla s'anomenen **cares** o polígons (F), els segments resultat de les interseccions dels plans es diuen **arestes** (A) i als punts inicial i final de les arestes se'ls designa com **vèrtexs** del políedre (V).

Definició 2.3. Sigui P un políedre, $F \subset P$ un polígon i $\{X_1, Y_1\}, ..., \{X_n, Y_n\} \subset F$ arestes. Diem que F és **regular** si $\|\overrightarrow{X_iY_i}\| = \|\overrightarrow{X_jY_j}\|$ per $i, j = 1, ..., n$, i si $|\overrightarrow{X_kY_k} \cdot \overrightarrow{X_lY_l}| = |\overrightarrow{X_lY_l} \cdot \overrightarrow{X_mY_m}|$ per $Y_k = X_l$ i $Y_l = X_m$ amb $k, l, m = 1, ..., n$ tals que $k \neq l \neq m$.

Es pot donar un enfoc més topològic al concepte de políedre si es relaciona amb la definició de triangulació propia de la topologia de superfícies.

Definició 2.4. Un políedre és la unió de tots els n-símplexs d'un complex simplicial, és a dir, d'un espai triangulable.

Per a un tractament concís d'aquesta definició topològica veure [6].

2.1.1 Algunes nocions dels políedres

S'exposen ara les propietats geomètriques dels políedres que el videojoc permet treballar: convexitat i la regularitat, la característica d'Euler, les simetries i la dualitat de políedres.

Convexitat i regularitat

Un políedre és convex si compleix la condició de conjunt convex, és a dir, que per a dos punts qualssevol del políedre, el segment que els uneix està contingut al políedre. Això és equivalent a dir:

Definició 2.5. Sigui P un políedre. Diem que P és **convex** si $\forall X, Y \in P$

$$Xt + (1-t)Y \in P, \text{ per a } t \in [0, 1].$$

Definició 2.6. Sigui P un políedre. Diem que P és **regular** si compleix:

- $\forall F$ cara de P és regular,
- $\forall F_1, F_2$ cares de P amb $\{X_1, Y_1\}, ..., \{X_n, Y_n\} \subset F_1$ i $\{Z_1, T_1\}, ..., \{Z_n, T_n\} \subset F_2$, tenen el mateix nombre d'arestes per cara i $\|\overrightarrow{X_iY_i}\| = \|\overrightarrow{Z_jT_j}\|$ per $i, j = 1, ..., n$,
- i $\forall F_1, F_2, F_3$ cares de P, $|(A_1, B_1, C_1) \cdot (A_2, B_2, C_2)| = |(A_2, B_2, C_2) \cdot (A_3, B_3, C_3)|$, on (A_i, B_i, C_i) és el vector normal a la cara F_i, i $\exists \{X_i, Y_i\}, \{Z_j, T_j\}$ tals que $\{X_i, Y_i\} \subset F_1, F_2$ i $\{Z_j, T_j\} \subset F_2, F_3$ per $i, j = 1, ..., n$.

És a dir, si els polígons que el formen són tots regulars, amb el mateix nombre d'arestes per cara i de igual longitud d'aresta, i els angles entre ells també són iguals.

Característica d'Euler

Definició 2.7. Sigui P un políedre convex amb F cares, V vèrtexs i A arestes. Aleshores, es compleix que

$$V - A + F = 2.$$

Aquesta equació s'anomena **relació d'Euler**.

En cas que el políedre no fos convex la relació original no se satisfà. Aleshores, es generalitza la relació d'Euler a la **característica d'Euler**.

Definició 2.8. Sigui P un políedre i $\mathcal{T}$ una triangulació de P. Denotem per t el nombre de triangles de $\mathcal{T}$, per a el nombre d'arestes i per v el nombre de vèrtexs. Definim, doncs, la característica d'Euler de P com

$$\chi(P) = v - a + t,$$

la qual no depén de la triangulació escollida.

Simetries

Sigui P un políedre de $(\mathbb{R}^3, d)$ com a espai mètric. Sigui $f : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$ un isomorfisme.

Definició 2.9. Diem que $f : P \longrightarrow P'$ és una **simetria** de P si $\forall p, q$ punts de P, $d(x, y) = d(f(x), f(y))$, amb $f(x), f(y) \in P'$. És a dir, és un isomorfisme que preserva distàncies, transformant el políedre en si mateix.

Diem que P i P' són congruents si $\exists f : P \longrightarrow P'$ simetria de P.

Definició 2.10. Diem que P és **uniforme** si els polígons que formen les seves cares són regulars i si, per a $p, q \in P$ vèrtexs qualssevol, $\exists f : P \longrightarrow P$ simetria tal que $f(p) = q$.

Definició 2.11. Diem que P té **simetria axial** (rotació respecte un eix) si $\exists p, q \in P$ tals que la direcció $p + \overrightarrow{pq}$ és un eix de rotació, i el políedre P' obtingut de la rotació es congruent amb P.

Definició 2.12. Diem que P té **simetria especular** (reflexió respecte un pla) si $\exists \pi$ un pla de $\mathbb{R}^3$ i $\exists f$ una simetria que separa el políedre en dues parts congruents.

Dualitat

Definició 2.13. Sigui P un políedre regular amb p vèrtexs per cara, i amb q cares envoltant a cada vèrtex. Denotem el **símbol de Schläfli** de P per $\{p, q\}$.

Definició 2.14. Sigui P un políedre regular amb símbol de Schläfli $\{p, q\}$. Diem que Q és el seu políedre **dual** si el símbol de Schläfli de Q és $\{q, p\}$.

2.1.2 Sòlids platònics

Els sòlids platònics són els únics políedres regulars i convexes i es demostra a partir de la fòrmula d'Euler vista abans [5]. Consisteixen en els sòlids base que formaran la figura final en el videojoc desenvolupat.

S'entaulen a continuació els trets principals associats, els quals es basen en les propietats descrites a l'apartat anterior.

	F	A	V	Euler	V per F	F per V
Tetraedre	4	6	4	2	3	3
Hexaedre	6	12	8	2	4	3
Octaedre	8	12	6	2	3	4
Dodecaedre	12	30	20	2	5	3
Icosaedre	20	30	12	2	3	5

	Simetria	Uniforme	Dual
Tetraedre	A / E	SÍ	Tetraedre (autoconjugat)
Hexaedre	A / E	SÍ	Octaedre
Octaedre	A / E	SÍ	Hexaedre
Dodecaedre	A / E	SÍ	Icosaedre
Icosaedre	A / E	SÍ	Dodecaedre

Table 2.1: Taula de sòlids platònics.

En el joc proposat es pretén reforçar aquests conceptes i les relacions entre sòlids.

2.2 Sòlid 3D com a composició de sòlids platònics

Tal i com s'ha comentat a l'inici del capítol, un dels objectius fonamentals del projecte és la construcció de figures en tres dimensions. Més concretament, on la geometria resultant no sigui de tipus *non-manifold*.

Definició 2.15. Diem que un sòlid és ***non-manifold*** si tres cares o més comparteixen una mateixa aresta o si dos o més cares comparteixen un vèrtex però no una aresta.

Per a discernir aquests sòlids no vàlids dels que sí ho són, s'utilitzaran els grafs. Amb aquesta eina, s'aconsegueix dotar d'un esquelet posicional al sòlid 3D. En geometria 3D és habitual associar un graf a les arestes i vèrtexs d'un políedre, però en aquest projecte es gestiona la posició de cada sòlid platònic base a l'espai tridimensional i s'estudien les relacions entre aquestes posicions.

Definició 2.16. Un **graf** G és un parell ordenat $G := (V, E)$ on V és un conjunt no buit d'elements anomenats vèrtexs (o nodes) i E és el conjunt d'arestes.

Els elements de E són parells no necessàriament ordenats de dos elements de V, és a dir, per a qualsevol $e \in E$, tenim que $e = \{v_1, v_2\}$ per a $v_1, v_2 \in V$.

Sigui $G := (V, E)$ un graf. Tenim les definicions següents:

Definició 2.17. Un **subgraf** G_s de G és un parell ordenat $G_s := (V_s, E_s)$ on $V_s \subset V$, $V_s \neq \emptyset$ i $E_s \subset E$, on $\forall e \in E_s$, $e \subset V_s$.

Definició 2.18. G és **dirigit** si els elements del conjunt E són parells ordenats de vèrtexs de V, és a dir, per a qualsevol $e \in E$, tenim que $e = (v_1, v_2)$ per a $v_1, v_2 \in V$.

Definició 2.19. Sigui G no necessàriament dirigit. Siguin $e_1, e_2 \in E$ tals que $e_1 = e_2 = \{v_1, v_2\}$ per a $v_1, v_2 \in V$. Aleshores diem que e_1 i e_2 són **arestes múltiples**.

Definició 2.20. Sigui G no necessàriament dirigit. Sigui $e \in E$ tal que $e = \{v, v\}$ per a $v \in V$. Aleshores diem que e és un **bucle**.

Definició 2.21. Sigui G no dirigit. Si G no té cap aresta múltiple ni cap bucle, diem que G és **simple**.

Definició 2.22. Definim l'**ordre** de G, $|V|$, al nombre de vèrtexs del graf. Definim la **mida** de G, $|E|$, al nombre d'arestes del graf.

Definició 2.23. Sigui G un graf simple. Podem definir la **matriu d'adjacència** de G com la matriu quadrada i simètrica A d'ordre $|V|$, on els elements $A_{ij} = A_{ji} = \mathbb{1}_E(e_{ij})$, on $e_{ij} = \{v_i, v_j\}$, amb $v_i, v_j \in V$.

$\mathbb{1}_E : V \times V \longrightarrow \{0, 1\}$ fa referència a la funció indicatriu del conjunt de les arestes E, on

$$\mathbb{1}_E(\{v, w\}) = \begin{cases} 1 & \text{si } \{v, w\} \in E \\ 0 & \text{si } \{v, w\} \notin E. \end{cases}$$

Definició 2.24. Sigui v un vèrtex de G. Definim el **grau** de v com la suma de cardinals

$$Gr(v) = \#Adj(v) + \#Buc(v),$$

on $Adj(v) = \{e \in E \mid v \in e\}$ són les arestes on apareix v i $Buc(v) = \{e \in E \mid e = \{v, v\}\}$ els bucles de v, que quedaran, doncs, comptabilitzats dues vegades.

Definició 2.25. Diem que G és **regular** si $\forall v_1, v_2 \in V$, $Gr(v_1) = Gr(v_2)$.

Definició 2.26. Diem que G és **complet** si $\forall v_1, v_2 \in V$, $\exists e \in E$ tal que $e = \{v_1, v_2\}$.

Definició 2.27. Siguin $v, w \in V$ dos vèrtexs diferents. Diem que v, w són **connexos** si $\exists v = v_1, ..., v_n = w \in V$ tals que $\{v_i, v_{i+1}\} \in E$, per a $i = 1, ..., n-1$.

Definició 2.28. Diem que G és **connex** si $\forall v, w \in V$ són **connexos**. Altrament, diem que és **inconnex**.

Definició 2.29. Sigui $G_s \subset G$ subgraf. Diem que G_s és una **component connexa** de G si G_s és connex.

2.2.1 Representació posicional en un graf d'un sòlid tridimensional

Sigui $S \subset \mathbb{R}^3$ un sòlid geomètric de l'espai euclidià format pels sòlids platònics $\{P_1, ..., P_n\}$, amb m cares cada políedre i de igual longitud d'aresta tots ells. Denotem per $\pi_{1i}, ..., \pi_{mi}$ els plans que formen el políedre P_i.

Podem definir els **grafs posicionals** de S com els grafs $G_{\mathbf{d}} = (V, E_{\mathbf{d}})$ amb

$$V = \{p \in P_i \mid d(p, \pi_{1i}) = ... = d(p, \pi_{mi})\},$$

on $i = 1, ..., n$, i els conjunts d'arestes associades

$$E_{\mathbf{d}} = \{\{v_1, v_2\} \in V \times V \mid d(v_1, v_2) = \mathbf{d}\},$$

amb $\mathbf{d} \in \mathbb{R}$ un escalar.

És a dir, considerem el graf format pels centres de cada políedre, amb les arestes que uneixen els vèrtexs separats una distància $\mathbf{d}$.

Per determinar si el sòlid S és vàlid, primer de tot cal comprovar que el graf és connex (equivalentment, només té una component connexa) per assegurar que no hi ha figures flotants a l'espai. Per a això s'aplica l'algorisme DFS (*depth-first search*) [23] sobre $G_{\mathbf{d}}$.

Ara, per verificar que S no satisfà les condicions de *non-manifold* exposades a l'inici de l'apartat, cal superar un dels següents escenaris.

1. Siguin $v_1, v_2 \in V$ tals que $d(v_1, v_2) = \sqrt{2}\mathbf{d}$. Aleshores $\exists w \in V$ tal que $\{v_1, w\}$, $\{w, v_2\} \in E_{\mathbf{d}}$. És a dir, que v_1, v_2 són connexos amb un intermediari.
2. Siguin $v_1, v_2 \in V$ tals que $d(v_1, v_2) = \sqrt{3}\mathbf{d}$. Aleshores $\exists w_1, w_2 \in V$ tal que $\{v_1, w_1\}$, $\{w_1, w_2\}$, $\{w_2, v_2\} \in E_{\mathbf{d}}$. És a dir, que v_1, v_2 són connexos amb dos intermediaris.

El cas 1 permet assegurar que no succeeix l'escenari de la Figura 2.1 (a) i el cas 2 ho fa amb l'escenari de la Figura 2.1 (b).

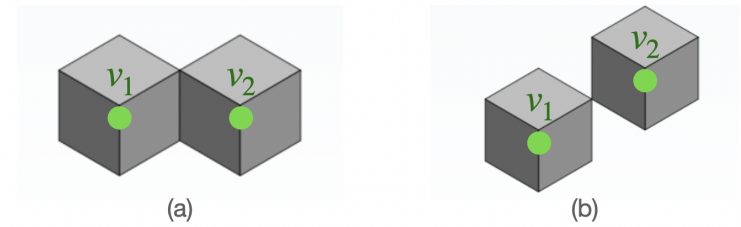

Figure 2.1: Sòlids *non-manifold*.

2.3 Rotacions en escenes 3D

Quan es tracta de videojocs en tres dimensions o senzillament de gràfics per computador, una de les transformacions lineals més utilitzades a l'hora de donar vida a les escenes són les rotacions. Es troben presents en multitud de situacions, ja sigui en girar la càmera principal o bé en visualitzar els moviments dels objectes de l'escena.

Tots aquests moviments necessiten ésser calculats al motor gràfic i les tècniques són moltes, com per exemple les matrius de rotació en 2D i 3D, les quals fan ús del cos dels complexos i els angles d'Euler respectivament. Les rotacions amb aquests darrers angles es poden modelar de la següent manera:

Definició 2.30. Sigui $(x, y, z) \in \mathbb{R}^3$ un vector. Podem expressar la rotació amb angles d'Euler amb el producte matricial següent:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{2.1}$$

on $(x', y', z') \in \mathbb{R}^3$ és el vector resultant de la rotació en ψ angles en l'eix Z, θ angles en l'eix Y, i ϕ angles en l'eix X.

No obstant això, hi ha una altra estratègia que, tot i no ser tant evident a priori, pot suposar una millora de càlcul computacional respecte les esmentades matrius de rotació i, a més, soluciona un problema provinent dels angles d'Euler: el *gimbal lock*. Aquest succeeix quan els angles del producte definit a l'equació (2.1) perden un grau de llibertat. Es veurà amb més detall a l'apartat 2.5.3.

És aquí on entra en joc el conjunt dels quaternions, un $\mathbb{R}$-espai vectorial que extén els nombres complexos, té estructura de grup abelià amb la suma i forma un anell de divisió amb el producte Hamiltonià, és a dir, és un anell unitari amb tot element no nul invertible però el seu producte no és commutatiu.

Tot seguit s'introdueixen els quaternions, mostrant la seva estructura d'anell amb el producte Hamiltonià, es representen les rotacions tridimensionals mitjançant aquest conjunt de nombres i s'analitzen les interpolacions lineals i esfèriques associades.

2.4 L'anell $\mathbb{H}$ dels quaternions

Definició 2.31. Un **quaternió** q és una expressió

$$q = s + x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \tag{2.2}$$

amb $s, x, y, z \in \mathbb{R}$ escalars, on s és la **part escalar** del quaternió i $x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ és la **part vectorial**.

Habitualment, es pot escriure el quaternió q de la forma $q = s + \mathbf{v}$, o bé pel parell ordenat $q = (s, \mathbf{v})$ on $s \in \mathbb{R}$, $\mathbf{v} = (x, y, z) \in \mathbb{R}^3$, alternatives que es mostren a [3].

Com s'ha esmentat abans, el conjunt dels quaternions

$$\mathbb{H} = \{s + x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \mid (s, x, y, z) \in \mathbb{R}^4\}$$

forma un $\mathbb{R}$-espai vectorial, satisfent l'addició entre dos elements i el producte per un escalar habituals. En concret, té dimensió 4 i observem que $\mathcal{B} = \{1, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$ n'és una base. El fet que $\mathbb{H}$ formi un espai vectorial dóna directament el resultat d'ésser un grup abelià amb l'addició.

S'exposa ara la multiplicació entre els elements d'aquest conjunt, la qual permetrà dotar-lo d'una estructura d'anell de divisió.

El producte Hamiltonià de dos quaternions es defineix mitjançant la propietat distributiva però tenint en compte unes regles de multiplicació per als símbols $\mathbf{i}, \mathbf{j}, \mathbf{k}$:

$$\begin{aligned} \mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 &= -1 \\ \mathbf{ij} &= -\mathbf{ji} = \mathbf{k} \\ \mathbf{jk} &= -\mathbf{kj} = \mathbf{i} \\ \mathbf{ki} &= -\mathbf{ik} = \mathbf{j} \end{aligned} \tag{2.3}$$

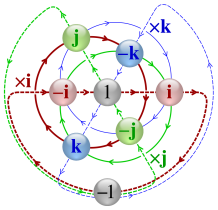

Figure 2.2: Normes multiplicatives. Reproduït de "Wikipedia", per Cmglee, 2021 (https://en.wikipedia.org/wiki/Quaternion). CC-BY-NC-ND.

Si s'analitzen aquestes normes, ràpidament es pot veure que provoquen la no commutativitat del producte, de tal manera que $\mathbb{H}$ ja no podrà assolir l'estructura de cos.

Definició 2.32. Siguin $q_1 = s_1 + x_1\mathbf{i} + y_1\mathbf{j} + z_1\mathbf{k} \in \mathbb{H}$, $q_2 = s_2 + x_2\mathbf{i} + y_2\mathbf{j} + z_2\mathbf{k} \in \mathbb{H}$ dos quaternions, definim el **producte Hamiltonià** entre q_1 i q_2 usant la propietat distributiva i les regles (2.3) de la següent manera:

$$\begin{aligned} q_1 q_2 = (s_1 + x_1\mathbf{i} + y_1\mathbf{j} + z_1\mathbf{k})(s_2 + x_2\mathbf{i} + y_2\mathbf{j} + z_2\mathbf{k}) = \\ (s_1 s_2 - x_1 x_2 - y_1 y_2 - z_1 z_2) \\ +(s_1 x_2 + x_1 s_2 + y_1 z_2 - z_1 y_2)\mathbf{i} \\ +(s_1 y_2 - x_1 z_2 + y_1 s_2 + z_1 x_2)\mathbf{j} \\ +(s_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 s_2)\mathbf{k} \end{aligned} \tag{2.4}$$

És interessant observar que, com $s_1, s_2, x_1, x_2, y_1, y_2, z_1, z_2 \in \mathbb{R}$, del resultat es pot concloure que $q_1 q_2 \in \mathbb{H}$ i, per tant, el producte Hamiltonià és tancat. Com s'ha mencionat abans, d'aquest fet i del caràcter distributiu amb el que es defineix aquest producte, se'n pot deduir que el conjunt de $\mathbb{H}$ forma un anell amb l'addició habitual i el producte Hamiltonià.

Proposició 2.33. En notació de la forma $q_1 = s_1 + \mathbf{v_1}$, $q_2 = s_2 + \mathbf{v_2}$, el producte Hamiltonià es pot escriure

$$q_1 q_2 = s_1 s_2 - \mathbf{v_1} \cdot \mathbf{v_2} + s_1 \mathbf{v_2} + s_2 \mathbf{v_1} + \mathbf{v_1} \times \mathbf{v_2} \tag{2.5}$$

En un principi, usant aquesta notació, es podria pensar el producte com l'aplicació directe de la propietat distributiva i el producte escalar de dos vectors. Però aleshores no es respectarien les normes exposades a l'equació (2.3). Es veu a continuació com s'obté aquest contraintuitiu resultat:

Demostració. Si prenem el producte obtingut a (2.4) amb la notació proposada i reordenem termes a la part vectorial, obtenim

$$q_1 q_2 = s_1 s_2 - x_1 x_2 - y_1 y_2 - z_1 z_2 + \begin{pmatrix} s_1 x_2 + s_2 x_1 + y_1 z_2 - z_1 y_2 \\ s_1 y_2 + s_2 y_1 + z_1 x_2 - x_1 z_2 \\ s_1 z_2 + s_2 z_1 + x_1 y_2 - y_1 x_2 \end{pmatrix}$$

$$= s_1 s_2 - \mathbf{v_1} \cdot \mathbf{v_2} + s_1 \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} + s_2 \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} + \begin{pmatrix} y_1 z_2 - z_1 y_2 \\ z_1 x_2 - x_1 z_2 \\ x_1 y_2 - y_1 x_2 \end{pmatrix}$$

$$= s_1 s_2 - \mathbf{v_1} \cdot \mathbf{v_2} + s_1 \mathbf{v_2} + s_2 \mathbf{v_1} + \mathbf{v_1} \times \mathbf{v_2}.$$

□

Observació 2.34. En notació de la forma $q_1 = (s_1, \mathbf{v_1})$, $q_2 = (s_2, \mathbf{v_2})$, l'escrivim

$$q_1 q_2 = (s_1 s_2 - \mathbf{v_1} \cdot \mathbf{v_2}, s_1 \mathbf{v_2} + s_2 \mathbf{v_1} + \mathbf{v_1} \times \mathbf{v_2}) \tag{2.6}$$

Es dedueix a continuació que aquest anell de $\mathbb{H}$ és també unitari.

Observació 2.35. L'element neutre del producte Hamiltonià és el quaternió $q = 1$.

Demostració. Fem servir, per exemple, la notació de l'observació anterior.

Siguin $q_1, q_2 \in \mathbb{H}$ dos quaternions tals que $q_1 = (s, \mathbf{v})$ i $q_2 = (1, \mathbf{0})$. Aleshores, tenim el següent:

$$q_1 q_2 = (s - \mathbf{v} \cdot \mathbf{0}, s\mathbf{0} + \mathbf{v} + \mathbf{v} \times \mathbf{0}) = (s, \mathbf{v}).$$

□

Per acabar de veure que els quaternions formen una estructura d'anell de divisió, cal definir els inversos dels elements no nuls de $\mathbb{H}$. Es mostra primer la noció de conjugació, anàloga a la de $\mathbb{C}$:

Definició 2.36. El **conjugat** d'un quaternió $q = s + \mathbf{v}$, que designem per $\bar{q}$, és un quaternió que ve donat per $\bar{q} = s - \mathbf{v}$.

A les referències [1] i [3], s'esmenta que se satisfà la commutativitat del producte Hamiltonià per a conjugats. A continuació es fa una petita extensió i es mostra cert per a quaternions amb part vectorial paral·lela.

Proposició 2.37. Siguin $q_1, q_2 \in \mathbb{H}$ quaternions. El producte Hamiltonià commuta entre ells $\iff$ les seves parts vectorials són paral·leles.

Demostració. Siguin $q_1 = s_1 + \mathbf{v_1}$, $q_2 = s_2 + \mathbf{v_2}$. Apliquem el producte vist a (2.5):

$$q_1 q_2 = s_1 s_2 - \mathbf{v_1} \cdot \mathbf{v_2} + s_1 \mathbf{v_2} + s_2 \mathbf{v_1} + \mathbf{v_1} \times \mathbf{v_2}$$
$$q_2 q_1 = s_1 s_2 - \mathbf{v_1} \cdot \mathbf{v_2} + s_1 \mathbf{v_2} + s_2 \mathbf{v_1} + \mathbf{v_2} \times \mathbf{v_1}$$

Aleshores, si fem $q_1 q_2 - q_2 q_1 = \mathbf{v_1} \times \mathbf{v_2} - (\mathbf{v_2} \times \mathbf{v_1}) = 2(\mathbf{v_1} \times \mathbf{v_2})$, tenim finalment que

$$q_1 q_2 - q_2 q_1 = 0 \iff \mathbf{v_1} \times \mathbf{v_2} = 0 \iff \mathbf{v_1} || \mathbf{v_2}.$$

□

Això demostra també directament la commutativitat entre conjugats, però es vol mostrar també una versió concreta ja que el resultat servirà per a definir els inversos multiplicatius.

Corol·lari 2.38. Sigui $q = (s, \mathbf{v}) \in \mathbb{H}$ un quaternió. Aleshores, el producte Hamiltonià commuta entre conjugats.

Demostració. Sigui $\bar{q} = (s, \text{-}\mathbf{v})$ el conjugat de q. Cal veure que $q\bar{q} = \bar{q}q$.

Usant (2.6) obtenim el següent:

$$q\bar{q} = (s^2 + \|\mathbf{v}\|^2, -s\mathbf{v} + s\mathbf{v} + \mathbf{v} \times \text{-}\mathbf{v}) = (s^2 + \|\mathbf{v}\|^2, 0)$$
$$\bar{q}q = (s^2 + \|\mathbf{v}\|^2, s\mathbf{v} - s\mathbf{v} + \text{-}\mathbf{v} \times \mathbf{v}) = (s^2 + \|\mathbf{v}\|^2, 0)$$

□

Observació 2.39. En notació $q = s + \mathbf{v}$, podem escriure $q\bar{q} = \bar{q}q = s^2 + \|\mathbf{v}\|^2$.

Definició 2.40. Sigui $q \in \mathbb{H}$. Definim la **norma** de q com

$$\|q\| = \sqrt{q\bar{q}} = \sqrt{s^2 + \|\mathbf{v}\|^2} \in \mathbb{R} \tag{2.7}$$

.

Teorema 2.41. Sigui q un quaternió no nul i $\bar{q}$ el seu conjugat. Sigui $\|q\| \in \mathbb{R}$ la norma de q. Aleshores,

$$q^{-1} = \frac{\bar{q}}{\|q\|^2} \tag{2.8}$$

és l'**invers** de q.

Demostració. S'ha vist que el producte Hamiltonià commuta per conjugats. Per tant, tot usant (2.7)

$$q\frac{\bar{q}}{\|q\|^2} = \frac{q\bar{q}}{q\bar{q}} = 1$$
$$\frac{\bar{q}}{\|q\|^2}q = \frac{\bar{q}q}{q\bar{q}} = \frac{q\bar{q}}{q\bar{q}} = 1$$

□

S'ha provat, doncs, que existeixen inversos per a tots aquells elements de $\mathbb{H}$ no nuls i que té estructura d'anell unitari. Per tant, es pot assegurar que els quaternions formen un cos no commutatiu.

En el següent apartat s'exposarà la representació de les rotacions tridimensionals, però fent ús d'aquesta extensió dels reals i els complexos que s'acaba de veure.

2.5 Rotacions amb quaternions

Si es parla a nivell d'operacions aritmètiques, l'anell dels quaternions es pot fer servir per a modelar rotacions de manera més eficient que amb les matrius de rotació. A més, pot ser un complement o una alternativa a la transformació mitjançant angles d'Euler, sobretot quan es vol rotar al llarg d'un angle sobre un eix qualsevol.

Tot i així, en cert moment es pot necessitar tenir una representació matricial del quaternió per aplicar a posteriori una altra transformació lineal que no sigui una rotació, com per exemple una translació que proporcioni moviment a un objecte a la vegada que va rotant. En aquest sentit, s'estableix la conversió de quaternió a la matriu de rotació equivalent i es provarà el canvi d'angles d'Euler a quaternions per tal d'evitar el problema del *gimbal lock*.

Finalment, es farà un tractament de l'interpolació lineal i esfèrica sobre aquest anell, sovint implementada per a realitzar animacions al llarg del projecte.

2.5.1 Rotació d'un vector mitjançant els quaternions

Definició 2.42. Sigui $v = V_x\mathbf{i} + V_y\mathbf{j} + V_z\mathbf{k} \in \mathbb{H}$ un quaternió amb part escalar nul·la. Diem que v és un quaternió **pur**.

Juntament amb la notació vista a l'apartat anterior, a [2] s'observa que aquesta definició ens permet identificar vectors reals tridimensionals amb els quaternions purs.

Definició 2.43. Sigui $q = s + x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \in \mathbb{H}$ un quaternió, q^{-1} el seu invers i $v \in \mathbb{H}$ un quaternió pur. Representem la **rotació** de v amb els productes Hamiltonians

$$v' = qvq^{-1} \tag{2.9}$$

on v' és el quaternió resultant de la rotació.

A [2] es comenta que el resultat d'aquest producte és també un quaternió pur. Però vegem-ho en detall.

Proposició 2.44. El quaternió v' resultant de la rotació és un quaternió pur.

S'exposa primer una definició que ajudarà a demostrar la proposició.

Definició 2.45. Sigui $q \in \mathbb{H}$. Diem que q és un quaternió **unitari** si $\|q\| = 1$.

Podem veure clarament que en el cas que q sigui unitari, l'equació (2.9) es pot expressar com

$$v' = qv\bar{q} \tag{2.10}$$

on $\bar{q}$ és el conjugat de q.

En el cas que q i q^{-1} no siguin unitaris, sempre es poden trobar dos quaternions que sí ho siguin ja que

$$v' = qv\frac{\bar{q}}{\|q\|^2} = \frac{q}{\|q\|}v\frac{\bar{q}}{\|q\|} = \frac{q}{\|q\|}v\frac{\bar{q}}{\|\bar{q}\|}$$

Demostració de la Proposició 2.44. Sigui $q = s + \mathbf{w} \in \mathbb{H}$ unitari i $v = 0 + \mathbf{v}$ un quaternió pur. Pel producte Hamiltonià i aplicant (2.10)

$$\begin{aligned} qv\bar{q} &= (s + \mathbf{w})\mathbf{v}(s - \mathbf{w}) \\ &= (-\mathbf{w}\cdot\mathbf{v} + s\mathbf{v} + \mathbf{w}\times\mathbf{v})(s - \mathbf{w}) \\ &= -s\mathbf{w}\cdot\mathbf{v} + s^2\mathbf{v} + s\mathbf{w}\times\mathbf{v} + (\mathbf{w}\cdot\mathbf{v})\mathbf{w} - s\mathbf{v}\mathbf{w} - (\mathbf{w}\times\mathbf{v})\mathbf{w} \\ &= s^2\mathbf{v} + 2s\mathbf{w}\times\mathbf{v} + (\mathbf{w}\cdot\mathbf{v})\mathbf{w} - \mathbf{w}\times\mathbf{v}\times\mathbf{w}, \end{aligned} \tag{2.11}$$

acabem observant que el resultat té forma de quaternió pur ja que tots els termes corresponen a la part vectorial. □

Per tant, recuperant la identificació de vectors reals tridimensionals amb quaternions purs, podem entendre aquesta rotació com una transformació d'un vector en un altre (Lengyel, 2016, p.87)[2].

Ara s'estableix la relació entre la transformació mitjançant quaternions i la rotació d'un cert angle sobre un eix qualsevol. Per a poder enllaçar-ho bé, es fa ús de la propietat del doble producte vectorial on, per $A, B, C \in \mathbb{R}^3$ vectors qualssevol, es compleix que

$$A \times (B \times C) = B(A \cdot C) - C(A \cdot B)$$

i se'n busca l'analogia, tal i com es suggereix a [1], amb la *fórmula de Rodrigues*:

$$v' = v\cos\alpha + (k \times v)\sin\alpha + k(k \cdot v)(1 - \cos\alpha)$$

on k és l'eix de rotació (vector unitari) i α és l'angle.

En aquest sentit i guiant-nos de la referència esmentada, si recuperem l'equació (2.11), aplicant el doble producte vectorial al darrer terme, podem reescriure el quaternió v' com:

$$\begin{aligned} v' = qv\bar{q} &= s^2\mathbf{v} + 2s\mathbf{w} \times \mathbf{v} + (\mathbf{w} \cdot \mathbf{v})\mathbf{w} - (\|\mathbf{w}\|^2\mathbf{v} - \mathbf{w}(\mathbf{w} \cdot \mathbf{v})) \\ &= (s^2 - \|\mathbf{w}\|^2)\mathbf{v} + 2s\mathbf{w} \times \mathbf{v} + 2(\mathbf{w} \cdot \mathbf{v})\mathbf{w} \\ &= \mathbf{v}(s^2 - \|\mathbf{w}\|^2) + (\mathbf{w} \times \mathbf{v})2s + 2(\mathbf{w} \cdot \mathbf{v})\mathbf{w} \end{aligned}$$

i expressant $\mathbf{w} = \|\mathbf{w}\|\mathbf{w_u}$, finalment tenim

$$v' = \mathbf{v}(s^2 - \|\mathbf{w}\|^2) + (\mathbf{w_u} \times \mathbf{v})2s\|\mathbf{w}\| + \mathbf{w_u}(\mathbf{w_u} \cdot \mathbf{v})2\|\mathbf{w}\|^2. \tag{2.12}$$

Si igualem aquesta fórmula a la de Rodrigues, observem que s'imposa el següent sistema d'equacions:

$$\begin{cases} s^2 - \|\mathbf{w}\|^2 = \cos\alpha \\ 2s\|\mathbf{w}\| = \sin\alpha \\ 2\|\mathbf{w}\|^2 = 1 - \cos\alpha. \end{cases}$$

Sumant la primera i la tercera equació, ens resulta $s^2 + \|\mathbf{w}\|^2 = 1$.

De la tercera equació, veiem que $\|\mathbf{w}\|^2 = \dfrac{1 - \cos\alpha}{2}$. D'aquí, per les raons trigonomètriques de l'angle meitat, obtenim que

$$\|\mathbf{w}\| = \sqrt{\frac{1 - \cos\alpha}{2}} = \sin\left(\frac{\alpha}{2}\right). \tag{2.13}$$

Substituint aquest resultat a la igualtat de sobre, veiem

$$s^2 + \|\mathbf{w}\|^2 = 1 \implies s^2 + \sin^2\left(\frac{\alpha}{2}\right) = 1 \implies s^2 = \cos^2\left(\frac{\alpha}{2}\right) \implies s = \cos\left(\frac{\alpha}{2}\right) \tag{2.14}$$

i efectivament satisfà també la segona equació degut a la fórmula del sinus de l'angle doble $\sin\alpha = 2\sin\frac{\alpha}{2}\cos\frac{\alpha}{2}$.

Ara, amb aquests valors obtinguts, podem escriure el quaternió unitari de (2.10) com

$$q = s + \mathbf{w} = s + \|\mathbf{w}\|\mathbf{w_u} = \cos\left(\frac{\alpha}{2}\right) + \sin\left(\frac{\alpha}{2}\right)\mathbf{w_u} \tag{2.15}$$

el qual representa una rotació d'angle α al voltant de l'eix de direcció $\mathbf{w_u}$, aplicable a qualsevol vector real v tridimensional (o en el seu equivalent quaternió pur) mitjançant el producte definit a l'equació (2.9), i donant com a resultat de la transformació un nou vector v', també quaternió pur (i per tant interpretable com un vector real) com hem vist abans. Es pot observar la rotació a la Figura 2.3.

2.5.2 Matriu de rotació equivalent

Sovint interessa aplicar varies transformacions lineals a un objecte. L'expressió poc intuïtiva de rotació mitjançant quaternions obliga a expressar-lo com una matriu 3x3 molt similar a la que deriva de la *fórmula de Rodrigues*, tal i com s'indica a [1].

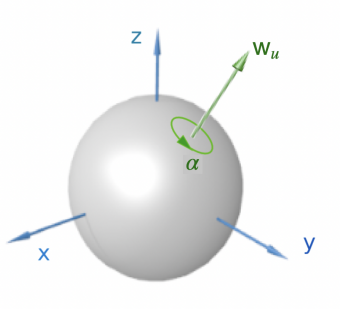

Figure 2.3: Representació de la rotació d'angle α sobre l'eix de direcció $\mathbf{w_u}$.

Per a això, s'exposa el següent resultat on es comprova l'equivalència entre la representació de la rotació mitjançant el producte Hamiltonià i l'aplicació d'una matriu d'ordre 3 sobre un vector.

Proposició 2.46. Sigui $q = s + x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \in \mathbb{H}$ un quaternió unitari i $v = 0 + \mathbf{v} \in \mathbb{H}$ un quaternió pur. Aleshores, se satisfà la següent igualtat:

$$v' = qv\bar{q} = \begin{pmatrix} s^2+x^2-y^2-z^2 & -2(sz-xy) & 2(sy+xz) \\ 2(sz+xy) & s^2-x^2+y^2-z^2 & -2(sx-yz) \\ -2(sy-xz) & 2(sx+yz) & s^2-x^2-y^2+z^2 \end{pmatrix} v \tag{2.16}$$

Demostració. Per a veure l'obtenció d'aquesta matriu, podem separar l'equació (2.12) i dotar-la d'estructura matricial. Per tant, expressant $q = s + \mathbf{w} = s + \|\mathbf{w}\|\mathbf{w_u}$, on $\mathbf{w_u} = (W_x, W_y, W_z)$ i $\mathbf{v} = (V_x, V_y, V_z)$ observem que:

$$v' = qv\bar{q} = \mathbf{v}(s^2 - \|\mathbf{w}\|^2) + (\mathbf{w_u} \times \mathbf{v})2s\|\mathbf{w}\| + \mathbf{w_u}(\mathbf{w_u} \cdot \mathbf{v})2\|\mathbf{w}\|^2$$

$$= (s^2 - \|\mathbf{w}\|^2)\begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} + 2s\|\mathbf{w}\|\begin{pmatrix} W_yV_z - W_zV_y \\ W_zV_x - W_xV_z \\ W_xV_y - W_yV_x \end{pmatrix} + 2(W_xV_x + W_yV_y + W_zV_z)\|\mathbf{w}\|^2\begin{pmatrix} W_x \\ W_y \\ W_z \end{pmatrix}$$

$$= \begin{pmatrix} s^2-\|\mathbf{w}\|^2 & 0 & 0 \\ 0 & s^2-\|\mathbf{w}\|^2 & 0 \\ 0 & 0 & s^2-\|\mathbf{w}\|^2 \end{pmatrix}\begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} + \begin{pmatrix} 0 & -2s\|\mathbf{w}\|W_z & 2s\|\mathbf{w}\|W_y \\ 2s\|\mathbf{w}\|W_z & 0 & -2s\|\mathbf{w}\|W_x \\ -2s\|\mathbf{w}\|W_y & 2s\|\mathbf{w}\|W_x & 0 \end{pmatrix}\begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix}$$
$$+2\|\mathbf{w}\|^2\begin{pmatrix} W_x^2 & W_xW_y & W_xW_z \\ W_xW_y & W_y^2 & W_yW_z \\ W_xW_z & W_yW_z & W_z^2 \end{pmatrix}\begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix}$$

$$= \begin{pmatrix} s^2-\|\mathbf{w}\|^2+2\|\mathbf{w}\|W_x\|\mathbf{w}\|W_x & -2s\|\mathbf{w}\|W_z+2\|\mathbf{w}\|W_x\|\mathbf{w}\|W_y & 2s\|\mathbf{w}\|W_y+2\|\mathbf{w}\|W_x\|\mathbf{w}\|W_z \\ 2s\|\mathbf{w}\|W_z+2\|\mathbf{w}\|W_x\|\mathbf{w}\|W_y & s^2-\|\mathbf{w}\|^2+2\|\mathbf{w}\|W_y\|\mathbf{w}\|W_y & -2s\|\mathbf{w}\|W_x+2\|\mathbf{w}\|W_y\|\mathbf{w}\|W_z \\ -2s\|\mathbf{w}\|W_y+2\|\mathbf{w}\|W_x\|\mathbf{w}\|W_z & 2s\|\mathbf{w}\|W_x+2\|\mathbf{w}\|W_y\|\mathbf{w}\|W_z & s^2-\|\mathbf{w}\|^2+2\|\mathbf{w}\|W_z\|\mathbf{w}\|W_z \end{pmatrix}\begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix}$$

$$= \begin{pmatrix} s^2+x^2-y^2-z^2 & -2sz+2xy & 2sy+2xz \\ 2sz+2xy & s^2-x^2+y^2-z^2 & -2sx+2yz \\ -2sy+2xz & 2sx+2yz & s^2-x^2-y^2+z^2 \end{pmatrix} \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix}$$

$$= \begin{pmatrix} s^2+x^2-y^2-z^2 & -2(sz-xy) & 2(sy+xz) \\ 2(sz+xy) & s^2-x^2+y^2-z^2 & -2(sx-yz) \\ -2(sy-xz) & 2(sx+yz) & s^2-x^2-y^2+z^2 \end{pmatrix} v$$

□

Corol·lari 2.47. Els quaternions $q, -q$ representen la mateixa rotació.

Demostració. Si observem la matriu definida a l'equació (2.16), veiem clarament que tots els termes de cada element tenen grau 2 i, per tant, es cancel·len tots els factors negats, donant com a resultat que la matriu $qv\bar{q} = -qv(\overline{-q})$. □

2.5.3 Correspondència amb els angles d'Euler

Una qüestió interessant de la representació de rotacions tridimensionals fent ús de quaternions és el fet d'evitar el problema principal causat pels angles d'Euler, el *gimbal lock*. A més d'aconseguir una millora a nivell de càlcul aritmètic, se soluciona amb el cos de $\mathbb{H}$ aquest contratemps, el qual ve provocat pel fet de no poder recobrir tot l'espai de rotacions usant els valors d'Euler.

Es veurà l'equivalència dels angles d'Euler definits a (2.1) amb les rotacions expresades mitjançant quaternions, tot aprofitant la representació matricial de l'apartat anterior, i es desembocarà a l'anàlisi de les singularitats que provoquen aquest bloqueig.

Si fem el producte de les matrius definit a l'equació (2.1), aleshores obtenim

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi+\sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi+\cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi+\sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi+\cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{2.17}$$

Ara, buscarem l'obtenció dels angles d'Euler a partir de tenir-ne l'expressió matricial. Sigui $\mathbf{R}$ la matriu que acabem de veure.

És clar que $\psi = \arctan \dfrac{\mathbf{R_{21}}}{\mathbf{R_{11}}}$, i també observem que $\phi = \arctan \dfrac{\mathbf{R_{32}}}{\mathbf{R_{33}}}$. Finalment, tenim que $\theta = \arcsin(-\mathbf{R_{31}})$.

Per tant, si apliquem aquests càlculs, tal i com es suggereix a [4], a la matriu expressada a l'equació (2.16), obtenim el següent:

$$\begin{cases} \phi = \arctan\left(\dfrac{2(sx+yz)}{s^2-x^2-y^2+z^2}\right) \\ \theta = \arcsin\left(2(sy-xz)\right) \\ \psi = \arctan\left(\dfrac{2(sz+xy)}{s^2+x^2-y^2-z^2}\right). \end{cases} \tag{2.18}$$

D'aquí, però, ens ve ràpidament que el domini de la funció arcsin està restringit a $[-1,1]$ i, per tant, no pot passar que $|sy - xz| > \frac{1}{2}$. Per tant, veiem que tindrem un problema de definició en el cas que l'angle θ associat a l'eix Y s'apropi a $\pm\frac{\pi}{2}$.

I és clar, doncs si provem a substituir aquest angle $\theta = \frac{\pi}{2}$ a l'equació de (2.17), obtenim el següent resultat

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 0 & -\cos\phi\sin\psi + \sin\phi\cos\psi & \sin\phi\sin\psi + \cos\phi\cos\psi \\ 0 & \cos\phi\cos\psi + \sin\phi\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\psi \\ -1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$= \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 0 & \sin\phi - \psi & \cos\phi - \psi \\ 0 & \cos\phi - \psi & \sin\psi - \phi \\ -1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

gràcies a les identitats trigonomètriques del sinus i cosinus de la suma d'angles.

Això desencadena una singularitat en el càlcul dels altres dos angles, ϕ i ψ, ja que aleshores no es pot obtenir el quocient $\frac{\mathbf{R_{21}}}{\mathbf{R_{11}}}$ i $\frac{\mathbf{R_{32}}}{\mathbf{R_{33}}}$.

Com a addició, es pot percebre clarament d'aquesta matriu de rotació que, quan θ val $\frac{\pi}{2}$, la rotació sobre el vector perd un grau de llibertat. Això és degut a que s'assigna un valor a l'angle θ ($\frac{\pi}{2}$ en aquest cas) corresponent a l'eix Y i el valor que es pugui donar als angles ψ i ϕ és redundant ja que, si s'observa la matriu, en tot moment es manté la rotació sobre l'eix X.

Anàlogament succeïx amb $\theta = -\frac{\pi}{2}$, entre d'altres combinacions dels tres angles que provoquen la mateixa singularitat. Aquestes singularitats es coneixen com *Gimbal Lock*.

2.5.4 Interpolació de quaternions

A l'actualitat, quan es parla de gràfics per computador en 3D, no es poden concebre les transformacions d'objectes i personatges sense tenir en compte el nombre d'imatges que es projecten per segon (FPS).

En aquest aspecte, s'entén que si es vol animar un cert moviment associat a un element de l'escena, es necessita obtenir la posició o angle d'aquest a cada imatge projectada. En una primera instància, hom únicament té coneixement del punt inicial i del punt objectiu al finalitzar la transformació que vol aplicar. Per tant, cal fer ús d'alguna eina que ens permeti calcular al llarg de cada imatge o *frame* la situació de l'objecte fins a obtenir el resultat final al cap de cert nombre de *frames*.

Es treballa, doncs, el concepte d'interpolació exposat a [1] que, aplicat al cos dels quaternions (apartat 2.4), aconsegueix suavitzar l'animació de gir i "marcar el camí" que segueix la rotació d'un objecte. Per a fer-ho, es computen els quaternions adequats a cada imatge projectada per tal de simular el moviment que es realitza.

Definició 2.48. Siguin $q_1, q_2 \in \mathbb{H}$ dos quaternions unitaris tals que $q_1 \neq -q_2$. Definim l'**interpolació lineal** (*lerp*) entre q_1, q_2 com una aplicació continua $\mathbf{q}_L$ tal que

$$\begin{aligned} \mathbf{q}_L \colon [0,1] &\longrightarrow \mathbb{H}_1 = \{q \in \mathbb{H} \mid \|q\| = 1\} \\ t &\longmapsto \frac{(1-t)q_1 + tq_2}{\|(1-t)q_1 + tq_2\|}. \end{aligned} \tag{2.19}$$

on $\mathbf{q}_L(0) = q_1$ i $\mathbf{q}_L(1) = q_2$.

L'aplicació està ben definida ja que totes les imatges pertanyen al subconjunt dels quaternions unitaris i, a més, el denominador només s'anul·la quan $t = \frac{1}{2}$ i $q_1 = -q_2$ que, per hipòtesi, no pot passar. Té sentit partir d'aquesta premisa, ja que el negat d'un quaternió en representa la mateixa rotació (tot i que a [1] no es parteix d'aquesta condició, hem trobat necessari fer aquest esment a la definició).

El conjunt $\mathbb{H}_1$ es pot interpretar com una hiperesfera unitaria de quatre dimensions. Aleshores, relacionant amb la premisa de l'enunciat 2.48, es pot dir que per definir $\mathbf{q}_L$, els elements q_1 i q_2 no poden ser punts antipodals.

Tot i així, si es consideren els quaternions q_1 i el quaternió interpolat en un cert moment t, $\mathbf{q}_L(t)$, com a vectors 4D, a [1] es pot veure que la velocitat amb que l'angle entre aquests dos vectors canvia no és constant. Això pot provocar petites pertorbacions no desitjades a l'escena a temps de *frame*. Es necessita, llavors, una funció d'interpolació que solucioni aquest problema, permetent que l'angle es modifiqui en correspondència amb el vector interpolat. És a dir, es vol interpolar explícitament els angles.

Proposició 2.49. Siguin $q_1, q_2 \in \mathbb{H}$ dos quaternions unitaris. L'aplicació continua

$$\begin{aligned} \mathbf{q}_S \colon [0,1] &\longrightarrow \mathbb{H}_1 \\ t &\longmapsto \frac{\sin(\theta(1-t))}{\sin\theta} q_1 + \frac{\sin(\theta t)}{\sin\theta} q_2 \end{aligned}$$

soluciona el problema proposat.

Una breu exposició geomètrica d'aquest resultat es pot trobar a [1], però s'estima necessari presentar certes indicacions més detallades i afinades.

Demostració. Si prenem la circumferència unitat i els quaternions q_1, q_2 com vectors dins aquesta, podem entendre el quaternió interpolat $q_t := \mathbf{q}_S(t)$ com el resultat d'aplicar la llei del paral·lelogram sobre q_1, q_2, de tal manera que $q_t = a(t)q_1 + b(t)q_2$, on $a(t)$ i $b(t)$ escurcen els quaternions inicial i final.

- Si observem la imatge de l'esquerra de la Figura 2.4, veiem ràpidament que el costat blau petit correspon a $\sin(\theta(1-t))$, i el costat blau gran correspon a $\sin\theta$.
- Si observem la imatge de la dreta de la Figura 2.4, veiem ràpidament que el costat blau petit correspon a $\sin\theta t$, i el costat blau gran correspon a $\sin\theta$.

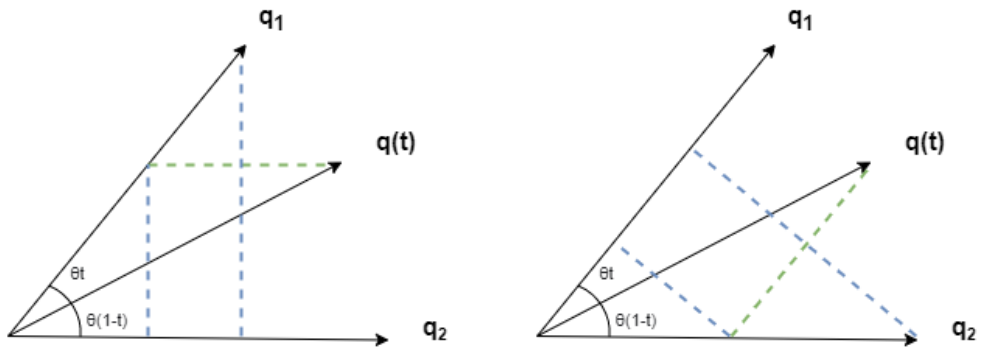

Figure 2.4: Triangles semblants amb angle θ.

Si prenem $a(t)$ i $b(t)$ com els segments desde l'origen fins al punt de tall de la perpendicular blava amb la perpendicular verda de la figura de l'esquerra i de la dreta respectivament, aleshores fent servir les raons dels triangles semblants generats per les perpendiculars blaves, obtenim directament que

$$a(t) = \frac{\sin(\theta(1-t))}{\sin\theta}, \qquad b(t) = \frac{\sin(\theta t)}{\sin\theta}.$$

És clar que amb aquesta representació estem interpolant linealment l'angle de separació θ i, en funció d'aquest mateix, estem obtenint la direcció del vector associat al quaternió interpolat. □

D'aquest resultat se'n diu **interpolació lineal esfèrica** o *slerp*.

2.6 *View frustum*

Un aspecte clau a tenir en compte en la visualització de gràfics per computador és la regió visible de l'espai en un cert moment. Així mateix, quan un professor estigui dissenyant una construcció tridimensional o un alumne estigui jugant per a recol·lectar les peces, cal assegurar en tot moment que les figures necessàries queden projectades correctament a la pantalla de l'usuari.

En aquesta secció, doncs, es defineixen els atributs associats al camp de visió o *view frustum* que gestionen tots els elements visibles en una escena 3D.

Definició 2.50. El ***view frustum*** és la regió de l'espai tridimensional que conté totes les figures projectades a la pantalla.

Aquest volum té forma de piràmide de base rectangular i el vèrtex representa la posició de l'ull o de la càmera. Amb aquesta estructura s'intenta simular la visió de con de l'ull humà, però amb la base adaptada a les pantalles d'ordinador.

Una piràmide està constituïda per 5 plans però en el *view frustum* se n'afegeix un 6è, situat entre el vèrtex i el pla que representa la base. D'aquesta manera, queden els plans associats a les cares del políedre, *right frustum plane*, *left frustum plane*, *bottom frustum plane*, *top frustum plane* i els de les bases, *near clipping plane* i *far clipping plane*, perpendiculars a la direcció de visió. S'observa a la Figura 2.5.

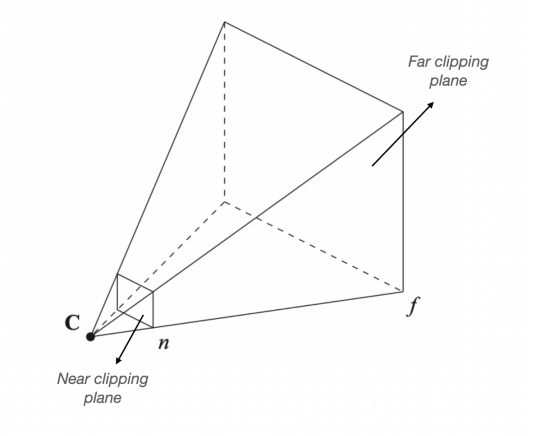

Figure 2.5: View frustum. Reproduït de [1].

A part d'aquest espai visible dels objectes, cal ressaltar un pla de visió o de projecció (***view plane***) on es projectaran les figures contingudes al *view frustum*. Per tant, la visualització serà diferent en funció del moment i s'ha de treballar amb un sistema de referència associat: el **VRC** (*viewing reference coordinate*).

L'origen de coordenades s'anomena **VRP** (*viewing reference point*) i coincideix amb un punt del *view plane*. Les coordenades de l'observador es poden designar per **OBS** i, aleshores, la direcció de visió,

$$\mathbf{VPN} = \frac{\mathbf{OBS} - \mathbf{VRP}}{\|\mathbf{OBS} - \mathbf{VRP}\|},$$

s'anomena *view plane normal*, la qual correspon amb el vector normal al *view plane* i amb la direcció Z del **VRC**. Amb aquests dos atributs es té el pla de projecció determinat.

Per establir l'eix Y del **VRC**, es pren el vector **VUP** que correspon al vector en direcció amunt a la càmera o, equivalentment, el que està inclòs al *view plane* i és perpendicular al **VPN**. Finalment, per a l'eix X es calcula el producte vectorial $\mathbf{VUP} \times \mathbf{VPN}$.

El pla de projecció es troba situat entre el *near clipping plane* i el *far clipping plane* en aquest sistema de coordenades dins el *view frustum*, el qual va variant segons la posició i orientació de la càmera.

Això pot proporcionar també un efecte de *zoom* a la representació de l'escena degut a la distància, g, corresponent a $\|\mathbf{OBS} - \mathbf{VRP}\|$ anomenada *focal length*. Es pot observar a

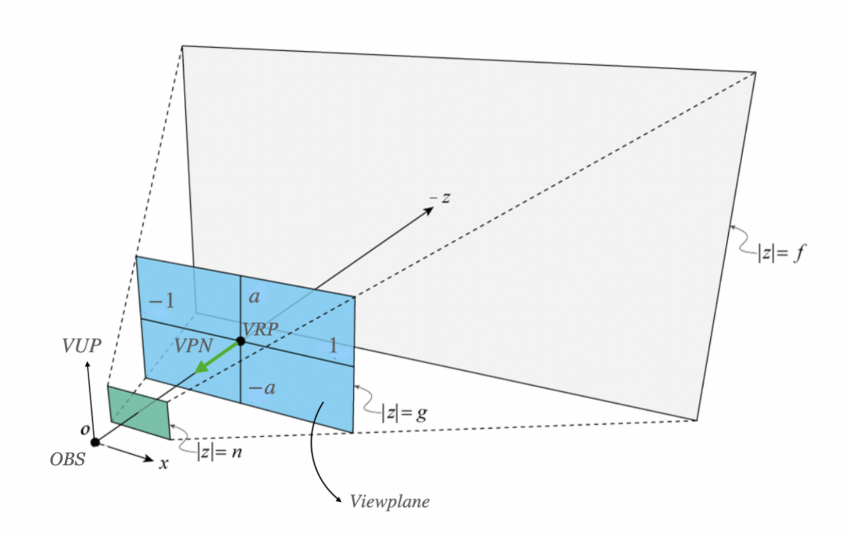

Figure 2.6: View i clipping planes. Adaptat de [2].

la Figura 2.6 que el pla de projecció interseca el *right frustum plane* i *left frustum plane* en $x = 1$ i $x = -1$ respectivament.

Tot i així, mantenint aquests valors de x, es pot augmentar o disminuir el valor de g si es modifica l'angle α que formen el *right frustum plane* i *left frustum plane*. De tal manera que

$$g = \frac{1}{\tan(\alpha/2)}.$$

En la mateixa direcció, es pot veure que el *view plane* interseca el *top frustum plane* i *bottom frustum plane* en $y = a$ i $y = -a$ respectivament, on $a = height/width$ seria la relació d'aspecte de la pantalla. S'obté, doncs, l'angle β que formen aquests dos plans

$$\beta = 2\arctan(a/g).$$

L'angle α s'anomena *horizontal field of view* i, l'angle β, *vertical field of view*.

2.7 Capsa mínima contenidora

Un cop explicats els atributs de la càmera i el *view frustum*, és important saber obtenir la direcció a on apunta la càmera. En altres paraules, establir la direcció $-$**VPN**.

En el cas del projecte, si s'ha d'observar una figura construida a partir d'altres figures més simples, el primer que cal és obtenir el punt mig de l'estructura, és a dir, el centre

d'una capsa mínima contenidora que englobi tota la construcció. Existeixen diferents mètodes per a calcular-la i s'introdueixen els dos més utilitzats en les seccions següents.

2.7.1 Capsa alineada amb els eixos

Una primera estratègia consisteix en agafar tots els vèrtexs de les figures i generar la capsa a partir dels eixos de coordenades habituals, prenent com a referència els punts de l'espai $(x_{min}, y_{min}, z_{min})$ i $(x_{max}, y_{max}, z_{max})$ corresponents al mínim i màxim de cada eix de tots els objectes a representar. Però això dóna lloc a una capsa gran amb regions buides, apareixent amb freqüència al *view frustum* i provocant un renderitzat innecessari.

2.7.2 Anàlisi de components principals

A [1] es proposa un mètode per a trobar un nou sistema de referència basat en la matriu de covariància anomenat **anàlisi de components principals**, que es detalla a continuació.

Siguin $\{P_1, ..., P_N\}$ el subconjunt de $\mathbb{R}^3$ de tots els punts de l'espai a englobar i $P = (X_1, X_2, X_3)$ un vector aleatori on cada entrada és una variable aleatòria associada a les components dels P_i. Es pot construir la matriu simètrica de covariància, C, on

$$C_{ij} = \text{Cov}\,(X_i, X_j), \text{ on } i, j \in \{1, 2, 3\}.$$

Si aquesta matriu és diagonal, és a dir $C_{ij} = 0$ per a $i \neq j$, aleshores vol dir que tots els vèrtexs estan distribuïts uniformement al voltant de cada eix i per tant no podem reduir la capsa.

En qualsevol cas, C és simètrica amb coeficients reals.

Teorema 2.51. *(Teorema espectral).* Sigui $C \in \mathcal{M}_n(\mathbb{R})$ simètrica. Llavors, $\exists P \in \mathcal{M}_n(\mathbb{R})$ tal que $P^{-1}CP$ és una matriu diagonal, per a $P^{-1} = P^t$.

Amb el resultat obtingut a 2.51 es veu que C és diagonalitzable per una matriu ortogonal. Aleshores, els vectors propis associats, U_x, U_y, U_z, formaran una base ortonormal del nou sistema de referència on les components dels vèrtexs en aquesta nova base no estan correlacionades.

Ara, per a obtenir el rang que ocupa la figura, seguint la idea exposada al mètode 2.7.1 pero usant els vectors propis, es calcula per a cada P_i, el pla amb normals U_x, U_y, U_z. De cada cas, s'obtenen aquells dos plans generats pels dos punts que tinguin la component $D_k = -U_k \cdot P_i$ màxima i mínima respectivament, per a $k = x, y, z$.

És a dir,

$$D_k = \max_{1 \leq i \leq N} \{-U_k \cdot P_i\} \quad , \quad d_k = \min_{1 \leq i \leq N} \{-U_k \cdot P_i\}$$

Finalment, en el cas de la component mínima en cada eix, es pren la normal al pla i la propia component amb signe contrari. Sigui la capsa mínima la regió delimitada pels 6 plans:

$$\begin{cases} U_x \cdot P + D_x \leq 0 \\ -U_x \cdot P - d_x \geq 0 \\ U_y \cdot P + D_y \leq 0 \\ -U_y \cdot P - d_y \geq 0 \\ U_z \cdot P + D_z \leq 0 \\ -U_z \cdot P - d_z \geq 0 \end{cases}$$

Els punts mitjos del segment que abasta cada eix són $\frac{D_k - d_k}{2}$. Com els vectors U_k són unitaris, queda clar que el punt mig de la capsa mínima serà

$$Q = \sum_{k=1}^{3} \frac{D_k - d_k}{2} U_k. \tag{2.20}$$

Es pot concloure doncs, que aquest punt Q serà el punt **VRP** necessari per a obtenir la direcció de la càmera en aquelles situacions on es requereixi observar una representació complexa en tres dimensions. No obstant, però, figures amb més detall com ara còniques i sòlids de revolució quedarien mal embolcallades per la capsa quadrada i seria convenient triar un altre tipus de recobriment més eficient.

2.8 Cas d'exemple en el desenvolupament: quaternions per moure la càmera

Com ja s'ha fet èmfasi, sovint en el projecte cal fer ús d'aquesta representació de les rotacions per tal d'animar els girs de certs elements i proporcionar una transició suau del moviment.

Si se situa inicialment la càmera a la posició $(0,0,0)$ de l'escena amb rotació nul·la sobre cada eix, això permet assignar-hi, per exemple, el quaternió $q_1 = \cos\frac{0}{2} + \sin\frac{0}{2}(0,1,0)$ com a rotació inicial.

Si s'afegeixen certes figures a l'escena i es vol rotar la càmera per a que segueixi observant el centre de la construcció (calculat amb (2.20)), cal definir un quaternió final. Suposem, per exemple, que l'objectiu és girar la càmera $90^o = \frac{\pi}{2}$ a la dreta. Això equival a mantenir el mateix eix de rotació però modificant l'angle α del quaternió. S'obté doncs $q_2 = \cos\frac{\pi}{4} + \sin\frac{\pi}{4}(0,1,0)$.

Entenent aquests quaternions com vectors 4D unitaris, es pot evidenciar l'angle de separació entre ells:

$$\theta = \arccos\left((1,0,0,0) \cdot \left(\frac{\sqrt{2}}{2}, 0, \frac{\sqrt{2}}{2}, 0\right)\right) = \arccos\frac{\sqrt{2}}{2} = \frac{\pi}{4}.$$

No obstant, si l'eix de rotació fos diferent entre els dos quaternions, el resultat d'aquest angle θ no seria trivial.

La pregunta és quan es triga a realitzar aquesta rotació i com es produeix la transició a temps de *frame*.

Suposant que els fotogrames es renderitzen amb una actualització fixa de 50 FPS, això dóna un temps de 0.02 segons per imatge projectada. Fent servir *slerp*, s'obté la següent equació:

$$\mathbf{q}(t) = \frac{\sin\left(\frac{\pi}{4}(1-t)\right)}{\sin\frac{\pi}{4}} q_1 + \frac{\sin\left(\frac{\pi}{4}t\right)}{\sin\frac{\pi}{4}} q_2$$

que, substituint pels quaternions inicial i final definits,

$$\mathbf{q}(t) = \sqrt{2}\sin\left(\frac{\pi}{4}(1-t)\right) + \sin\left(\frac{\pi}{4}t\right) + \sin\left(\frac{\pi}{4}t\right)(0,1,0)$$

amb $t = 0.02n$, per $n = 0, ..., 50$.

Per tant, al *frame* n, la rotació representada ve donada per la funció

$$\mathbf{q}(n) = \sqrt{2}\sin\left(\frac{\pi}{4} - \frac{\pi n}{200}\right) + \sin\left(\frac{\pi n}{200}\right) + \sin\left(\frac{\pi n}{200}\right)(0,1,0) \tag{2.21}$$

on efectivament $\mathbf{q}(0) = q_1$ i $\mathbf{q}(50) = q_2$.

Es mostren a la Figura 2.7 les captures de 4 *frames* de l'editor de Unity on es pot veure el quaternió interpolat (en format (x, y, z, s)) a cadascun d'ells després d'aplicar el resultat vist en aquest apartat. També s'observa de manera gràfica com afecta la rotació a la càmera de l'escena al llarg dels fotogrames.

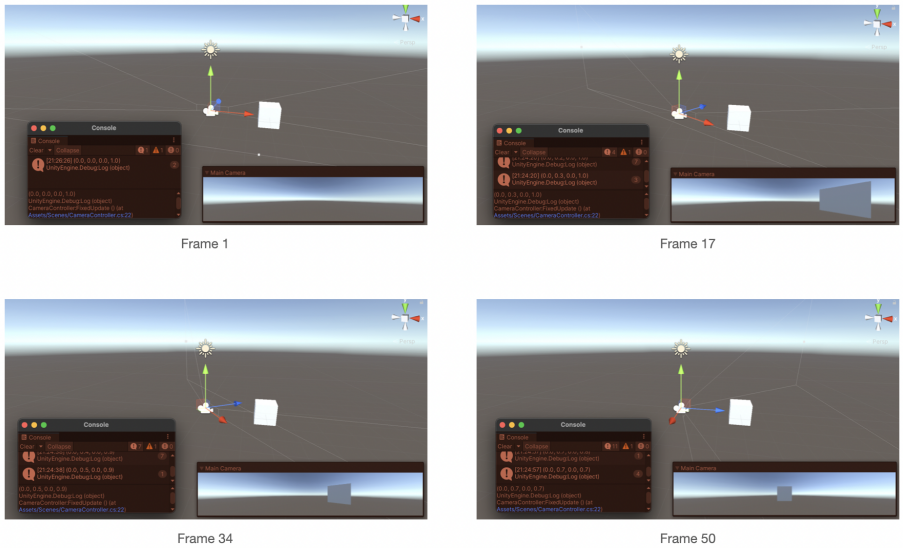

Figure 2.7: Rotacions de la càmera en 4 *frames* al girar $\frac{\pi}{2}$.

Capítol 3

Disseny de l'aplicació

3.1 Anàlisi del videojoc

El concepte base d'aquest projecte es fonamenta, en la seva vessant didàctica, en objectius semblants als dels treballs previs de *Geopieces* [7][8] i la seva ampliació [9]; a saber, complementar l'aprenentatge de la geometria dels alumnes de secundària i facilitar al professorat l'adaptació del coneixement a cada estudiant. En qualsevol cas, en la vessant tecnològica i de desenvolupament es plantegen uns requeriments diferents que donaran lloc a la proposta del treball.

S'exposen a continuació els objectius didàctics, així com el conjunt de requisits funcionals i no funcionals associats al videojoc que, un cop estudiats els treballs previs relacionats en la següent secció, permeten conduir al disseny de l'aplicació.

3.1.1 Objectius d'aprenentatge

La geometria tridimensional es comença a tocar al cicle superior de primària de manera molt intuïtiva, però no és fins al final del primer Cicle de Secundària o 4t de la ESO on es treballen més propietats geomètriques com els volums. Per tant, el projecte es focalitza en aquest públic objectiu i més concretament en treballar característiques exposades a la secció 2.1.1, ja que alguns conceptes basats en el càlcul poden ser complicats de gamificar. En qualsevol cas, es mostren a continuació els objectius que es persegueixen amb aquest complement de geometria 3D a l'aula:

- ***Reconeixement de sòlids i propietats geomètriques.*** L'usuari ha de ser capaç d'identificar visualment políedres regulars i convexos, i diferenciar-los dels que no ho són. Entenent les propietats geomètriques descrites a la secció 2.1.1, ha d'aconseguir tant classificar aquests sòlids com contestar a certes preguntes de caire matemàtic.

- ***Rotacions i orientació a l'espai.*** Es desitja que els alumnes aprofundeixin en l'orientació a l'espai usant la càmera de l'escenari i adaptant-la al seu favor. A-questa, però, depèn únicament de les rotacions, essent el jugador l'encarregat de saber posicionar-la en un espai 3D per a mostrar per pantalla el contingut que li convé.
- ***Projeccions.*** Les projeccions sobre el pla són un concepte fonamental que es pot explorar amb sombres o graelles que representen una figura tridimensional. Es desitja que l'estudiant entengui els plans de projecció de l'espai i tingui la capacitat de resoldre una construcció a partir d'aquestes vistes.
- ***Construcció de figures tridimensionals.*** Lligat amb els dos punts anteriors, l'usuari ha d'aprendre a encaixar figures tot situant-se a l'espai i orientant-se amb els eixos.

3.1.2 Especificació del *software*

A continuació s'analitzen quins han estat els requisits per a iniciar el desenvolupament del *software*.

Es desitja que el joc sigui usable per dos tipus d'usuari: els alumnes (*player*) i els propis professors (*designer*) d'aquests. Es considera que ha de ser una eina de comunicació entre els dos tipus d'usuaris per tal de complementar el coneixement que s'assoleix a l'aula, ja sigui durant la classe o a casa de l'estudiant. Així doncs:

- Professor/a: serà l'encarregat/da de dissenyar noves missions personalitzades per a certs estudiants, relacionades amb les característiques explicades a 2.1.1 i amb la possibilitat d'obtenir *feedback* sobre el que ha treballat l'alumnat.
- Alumne: destinatari principal del videojoc i superarà qüestions relacionades amb la geometria tridimensional segons el problema elaborat pel professor, guiant-se a partir de certes vistes i projeccions.

3.1.3 Requeriments no funcionals

Per tal de jugar el joc a l'aula, es considera que el joc usarà un cursor de ratolí o de *touchpad*. En aquest sentit es planteja que funcioni sobre un ordinador amb Windows 10, d'equipament estàndard usat en els equips informàtics a les escoles o els portàtils personals dels alumnes, o bé una tauleta Android amb una pantalla suficientment gran i amb possibilitat de connectar-hi un teclat i un *mouse*. En addició, es considera que existeixi connexió a internet per tal de poder establir la comunicació professor-alumne mitjançant el joc, el qual fa servir la persistència de dades en remot per a mantenir múltiples sessions en diferents temps i classes. Finalment, la resolució designada per a l'aplicació és 1920x1080, considerada el format estàndard per a ordinadors en el moment de realització del projecte.

3.2 Antecedents

En aquesta secció es mostren els aplicatius o treballs que s'han explorat i que tenen unes arrels didàctiques força relacionades amb el projecte actual. S'analitzen els punts forts i febles, com incideixen en l'especificació que hem fet i s'obtenen les conclusions necessàries que permeten encaminar la proposta de disseny.

3.2.1 Geometria bàsica en 2D

Els treballs originals sobre *Geopieces* fan un tractament de la geometria bidimensional, posant èmfasi en el reconeixement de figures i els trets distintius associats com ara les simetries, el perímetre o l'àrea.

- ***Geopieces: Part 2D*** [7]. El videojoc se centra en el concepte de "campanya", a partir del qual s'incita a l'usuari a recollir peces pròpies de la geometria plana per tal d'acabar desenvolupant un sòlid tridimensional. En aquest procés, el jugador ha de resoldre preguntes relacionades amb la geometria, amb la possibilitat de veure la progressió que porta de la missió en qualsevol moment. Funciona amb Windows 10 i el professor pot editar les missions en arxiu local però no hi ha persistència remota ni assignació.

3.2.2 Millora en la instrucció de la geometria 3D

Novament es ressalten els projectes sobre *Geopieces* i la seva ampliació, els quals busquen complementar l'aprenentatge de la geometria a l'espai mitjançant exercicis de desplegament i de construcció de sòlids. Es destaquen també diversos *applets* interactius que potencien la percepció de l'espai i la construcció 3D exposades a la secció 3.1.1.

- ***Geopieces: Part 2D - 3D*** [8]. Un cop obtingudes les figures planes a la part 2D, el jugador ha d'aconseguir desplegar una figura 3D objectiu fent servir les peces que ha recollit i guardat en el seu inventari. L'estudiant, per tant, explora el desplegament en el pla de figures tridimensionals. Funciona amb Windows 10 i el professor pot editar les missions en arxiu local però no hi ha persistència remota ni assignació.
- **Ampliació de *Geopieces*** [9]. En aquesta extensió del joc original es busca que l'usuari, mitjançant un inventari de sòlids tridimensionals bàsics, aconsegueixi construir una estructura determinada que es mostra en un minimapa. Funciona amb Windows 10, MacOS i Linux, i el professor disposa d'un servidor per dissenyar les missions, però no es troba del tot vinculat amb el joc.
- ***WisWeb applets*** [10]. Conjunt d'aplicatius destinats a millorar certs aspectes de la geometria 2D i 3D, separats per categories o edat de l'usuari. Entre ells destaca *Building with blocks*. Funcionen a la web i no hi ha cap tipus de personalització per alumnes.

- ***Building with blocks*** [11]. Aplicació en línia que permet construir figures a partir de blocs (cubs) donant com a pistes diverses formes de visualització de la figura objectiu.
- ***Isometric Drawing Tool*** [12]. Eina digital interactiva que permet construir una figura amb cubs i quadrats, rotar-los i veure'n les projeccions en tres plans diferents. Es troba a la web i no permet el guiatge del professor ni assignació d'objectius als alumnes.

Es mostra a la Figura 3.1 les imatges de les aplicacions mencionades.

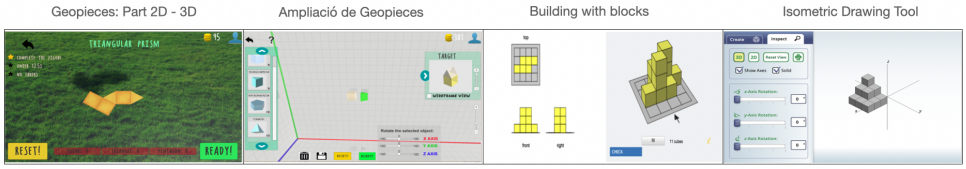

Figure 3.1: Comparativa dels jocs i *applets* 3D.

3.2.3 Conclusions

Tot i que aquest projecte es concentra en identificar la geometria tridimensional, el fil conductor i la base de farmeig que proporciona el joc en 2D [7] i 2D-3D [8] de *Geopieces* es troba força present en els videojocs de l'actualitat i, per tant, és una bona estratègia de gamificació del projecte per tal de motivar els alumnes. Tot i així, es considera més dinàmic i participatiu un estil de joc a l'aula on el professor sigui una peça fonamental. Aquest aspecte seria l'equivalent al "Mode classe" que no es va arribar a implementar en els treballs originals, o el segon objectiu de l'ampliació realitzada l'any següent d'aquest mateix projecte.

En quant a aquesta ampliació [9], ja es treballa en la representació tridimensional i s'observa una gran varietat de possibilitats de millorar la distribució espacial per part de l'alumne, ja que tenir una estructura de referència i provar de reconstruir-la és un exercici fonamental que permet treballar aquesta percepció. Per a realitzar les comprovacions necessàries i guiar al jugador, aquest projecte fa ús dels grafs i els isomorfismes entre ells, aprofundint doncs en un problema considerat NP complet. Tot i així, es pot aprofitar aquesta direcció i realitzar el guiatge a través d'altres formes de visualització que no siguin partint de la rèplica exacta.

Finalment, cal ressaltar les motivacions didàctiques dels aplicatius i eines com *WisWeb* [10] o *Isometric Drawing Tool* [12], les quals realitzen un molt bon treball didàctic per a potenciar la coordinació 3D de l'usuari, però no tenen gaire component d'adaptabilitat segons l'alumne.

Tenint en compte tot això i establint una relació amb la idea mencionada a la secció anterior, s'exposa a continuació la proposta de disseny del videojoc 3D desenvolupat en aquest treball.

3.3 *Game design*

Després de treballar sobre els requisits inicials del joc, haver tingut en compte els objectius d'aprenentatge i fer una exploració dels projectes anteriors, en aquest apartat es presenten el *storyboard* i el model de domini proposats de l'aplicació.

Tot i que el videojoc es recolza en els treballs previs sobre *Geopieces*, aquest projecte està realitzat desde zero amb especificacions força diferents, dotat d'un enfoc més *arcade* i sense seguir un fil de campanya en concret. Es fa ús, però, de l'estratègia de farmeig habitual en molts videojocs, amb la qual es busca l'obtenció d'uns determinats ítems necessaris per a realitzar una tasca concreta.

La missió màxima és que l'alumne aconsegueixi superar els reptes de construcció 3D que ha dissenyat el professor de manera personalitzada per a ell o per a un grup determinat. Per tal d'assolir-ho, l'estudiant ha de recollir primer el nombre de peces que formen la construcció, tot resolent durant el procés qüestions relacionades amb geometria de l'espai basades en les propietats que ha decidit el professor al moment d'editar la tasca.

3.3.1 *Storyboard*

Començant per les pantalles de *Register* i *Log In* (Figura 3.2), l'usuari es registra i s'autentica amb un correu electrònic, un nom d'usuari i una contrassenya. Finalment, pot escollir si el compte és o *player* o bé *designer*.

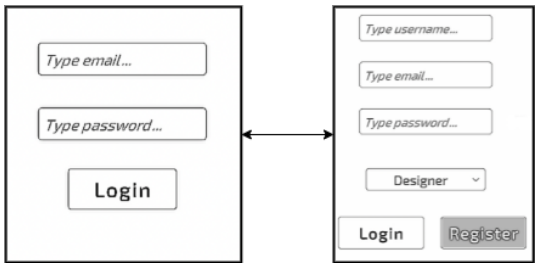

Figure 3.2: Pantalles de registre i login.

En el cas de ser un professor, el que es mostra és la llista de missions dissenyades per ell i un botó amb possibilitat de crear-ne una de nova. Ho podem veure a la figura 3.3. En aquest mateix menú, es pot observar les estadístiques de cada alumne a cada missió dissenyada, és a dir, el percentatge de figures obtingudes per cada jugador.

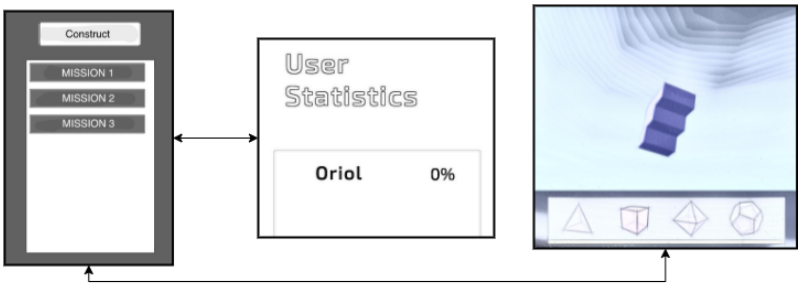

Figure 3.3: Camí del professor.

L'editor 3D consisteix en una eina per construir, mitjançant políedres units cara a cara, una figura més complexa. El professor disposa d'un minimapa per tenir un punt de vista global de la figura que està construint, així com un botó per re-ubicar la càmera. El nombre d'objectes disponibles per afegir queda en tot moment indicat.

La pantalla de Crear missió (Figura 3.4) permet assignar la construcció realitzada prèviament a un alumne o a un grup d'ells, tot seleccionant, per a cadascun individualment, les característiques geomètriques que el *designer* cregui convenient. Aquestes venen separades en blocs i en tot moment es mostra el *player* que s'està assignant per a facilitar l'experiència d'usuari.

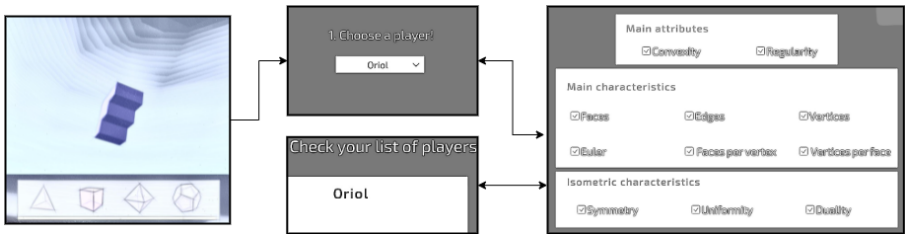

Figure 3.4: Creació de la missió.

Passem al cas que l'usuari sigui un alumne (Figura 3.5). La primera pantalla és la llista de missions que té assignades i han estat creades pels *designers*/professors. Es pot observar el percentatge que indica el progrés de cada missió i l'inventari de figures que porta a cadascuna. Si es selecciona una missió, es va a la tria de minijoc.

Per a fer-ho més dinàmic i entretingut, es distingeixen almenys tres minijocs on l'alumne pot aconseguir les peces necessàries, exercitant a la vegada les capacitats exposades a 3.1.1. En un primer, l'usuari treballarà la seva capacitat d'identificació de diferents sòlids gràcies a discriminar-los segons les característiques associades que ha designat el professor. En el segon i tercer, l'estudiant haurà de fer ús dels seus coneixements per a obtenir una recompensa, tot contestant preguntes relacionades amb les nocions geomètriques que se li han designat de manera personal. Per tal d'adaptar-se als controls d'aquests

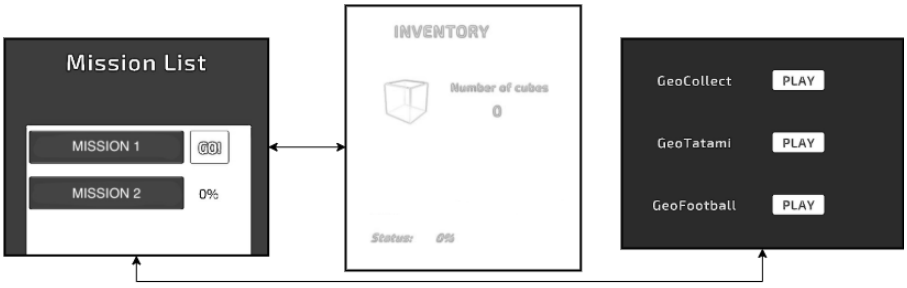

Figure 3.5: Camí de l'alumne.

dos minijocs, el jugador haurà de familiaritzar-se amb les rotacions de la càmera, entrenant així la seva orientació en l'espai.

A la figura 3.6 es mostren els minijocs a jugar.

En el primer, *GeoCollect* (Figura 3.6 (a)), el *player* ha d'intentar recollir cinc figures seguides que tinguin la propietat geomètrica de l'esquerra. Aquests objectes van caient des de dalt de la pantalla marcant una ombra a la paret. Si obté un sòlid amb la propietat del mig pot seguir amb la ratxa, però en cas que s'impacti amb una figura categoritzada amb la propietat de la dreta, es perd la partida. Un cop aconseguida la ratxa, es pot obtenir la figura objectiu per a la construcció final.

En el cas de *GeoTatami* (Figura 3.6 (b)), el jugador pren possessió d'una esfera i ha d'intentar eliminar les enemigues sense caure ell del tatami, fet que provocaria la derrota automàticament. L'alumne ha d'orientar el seu moviment a través de rotar la càmera. El minijoc està separat per nivells on, eventualment, apareix una figura associada a la construcció. Un cop recollida no tornarà a apareixer en aquest nivell i serà necessari avançar més en el minijoc. Per tal d'aturar el moviment dels enemics es disposa a cada nivell d'un *bonus* que activa una pregunta tipus test relacionada amb una temàtica de geometria 3D que ha designat el professor al crear la missió. Només en el cas que s'encerti la pregunta, s'activa la bonificació.

El darrer minijoc, *GeoFootball* (Figura 3.6 (c)), té el mateix fonament que l'anterior pero de caire futbolístic. L'objectiu és fer entrar les pilotes enemigues a la seva propia porteria i evitar que entrin a la del jugador. En el cas que es produeixi Gols en contra − Gols a favor = 5, es perd la partida. Els *bonus*, les preguntes geomètriques associades i la obtenció de figures segueixen la mateixa línia que a *GeoTatami*.

Aquests dos minijocs estan adaptats dels que s'han desenvolupat al curs de Unity 3D, *Junior Programmer Pathway*, de la plataforma Unity Learn [13], durant el juliol, agost i setembre previs a l'inici curricular del TFG.

Finalment, quan el *player* completa l'inventari associat a una missió, es pot dirigir a *GeoSudoku* (Figura 3.6 (d)) on l'objectiu de cada estudiant que hi arribi és reconstruir la figura exacta que ha dissenyat el professor en un inici. Es proporcionen com a indicadors tres graelles corresponents a les projeccions de la figura en els plans XY, ZY i XZ, on a

cada posició hi apareix el nombre de cubs amb les mateixes coordenades. Per a facilitar el guiatge de l'alumnat, s'adjunten també a la GUI unes altres 3 graelles associades a la figura que està realitzant el propi estudiant. D'aquesta manera, es pot observar en temps real a on està posant les figures, el nombre que porta d'elles a cada direcció i si va o no en bon camí. Quan l'usuari cregui convenient, pot verificar si és correcte el muntatge que ha fet.

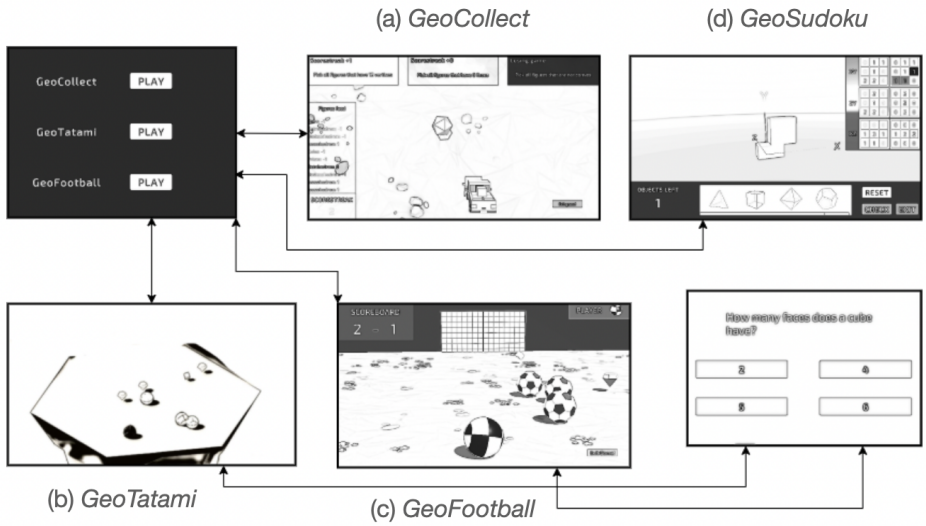

Figure 3.6: Esquema dels minijocs.

3.3.2 Model de domini

Un cop plantejat el *storyboard* anterior, es pot desembocar en el disseny del model de domini associat al projecte. S'observa a la Figura 3.7.

Es mostren els dos tipus d'usuari, alumne i professor, on aquest darrer dissenya varies construccions que generen una missió per a un o diversos alumnes. En quant a aquestes, disposen d'un estat de completació i d'un inventari associat a cada alumne, i es destaquen els tres minijocs que s'han exposat en l'apartat anterior. S'observa també els desafiaments que disposen d'una propietat geomètrica, la qual caracteritza un conjunt de figures. Aquesta relació suposa un problema a tractar ja que les característiques escollides pel professor poden contradir-se en el joc de *GeoCollect*, i es planteja una solució als Algorismes 4 i 5 de la secció 4.4.

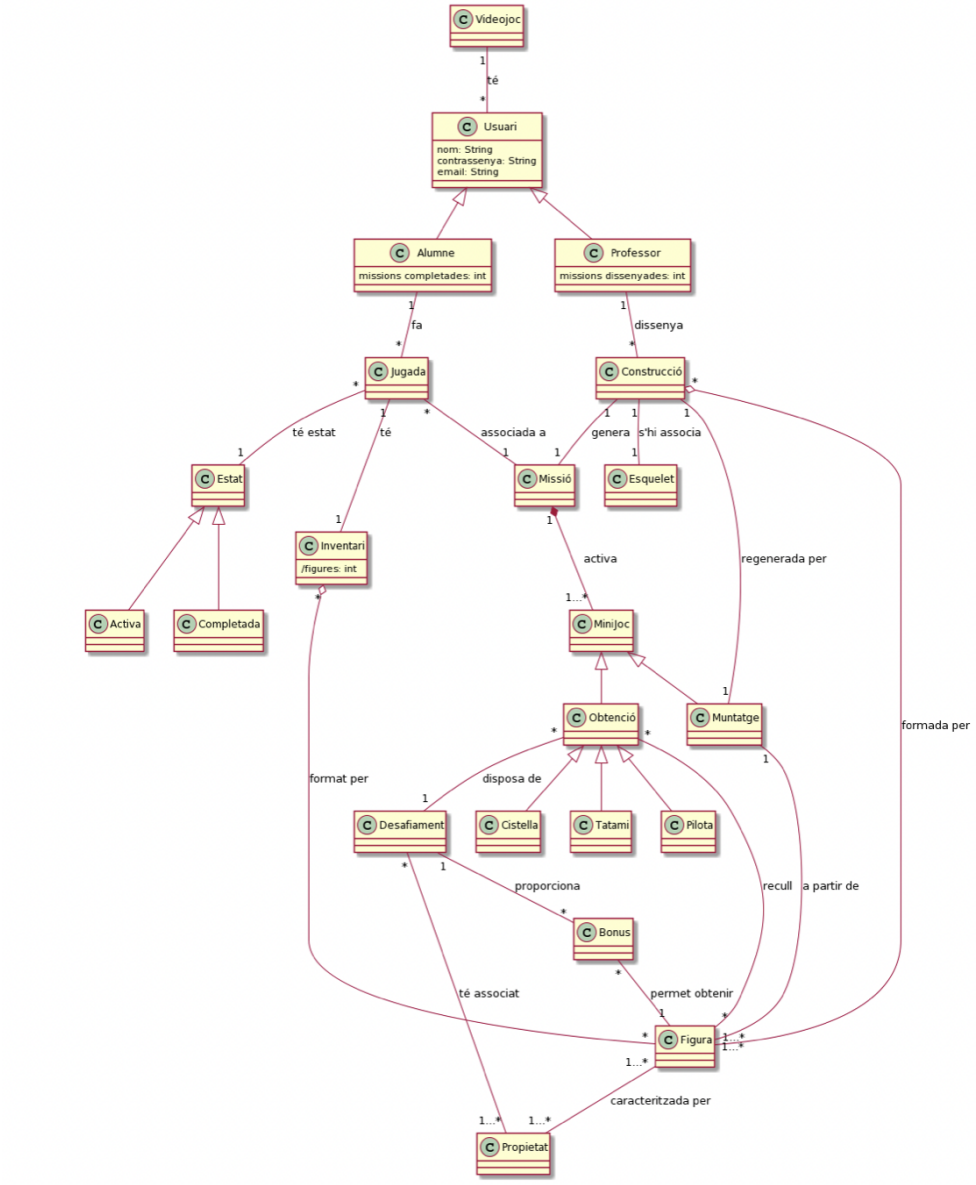

Figure 3.7: Model de domini.

Capítol 4

Disseny i desenvolupament del software

En aquest capítol es detallen les explicacions associades a la part de desenvolupament del projecte, argumentant en primer lloc el perquè s'ha triat una eina i no una altra, exposant també l'arquitectura de codi utilitzada i com s'ha adaptat a la tecnologia emprada i, finalment, mostrant els algorismes més rellevants del videojoc i com han implicat la necessitat d'explorar els conceptes formalitzats al capítol 2.

4.1 Tecnologia i arquitectura del sistema

El més primordial quan es vol començar a desenvolupar un videojoc és escollir el motor gràfic. En un inici, es poden contemplar diverses opcions viables per a realitzar-lo, entre les moltes existents:

- ***Unity 3D*** [18]: motor de videojocs desenvolupat per Unity Technologies, multiplataforma (videoconsoles, PC, Mac, Linux, Android, iOS) i es programa en C#.
- ***Unreal Engine 4*** [19]: motor de videojocs desenvolupat per Epic Games, multiplataforma i es programa en C++.

Fora dels motors gràfics habituals, també seria possible produir un videojoc mitjançant un IDE com ara:

- ***Android Studio*** [20]: entorn de desenvolupament integrat (IDE) oficial d'Android, desenvolupat per Google. Es programa en Java o Kotlin.
- ***XCode*** [21]: entorn de desenvolupament integrat (IDE) oficial de MacOS, desenvolupat per Apple, i destinada a desenvolupar *software* per a MacOS o iOS. Es programa en Swift o Objective-C.

En qualsevol cas, aquestes darreres opcions queden ràpidament descartades ja que l'objectiu fonamental és desenvolupar un videojoc 3D, cosa per la qual no estan preparades, i poder executar-lo en un ordinador o portàtil a l'aula de classe (com a cas excepcional, en una tauleta amb Android). Per tant, l'elecció queda fitada dins els motors gràfics anteriors, i s'ha escollit **Unity 3D** degut a la seva suau corba d'aprenentatge, la flexibilitat per desenvolupar a qualsevol plataforma, l'abundant documentació oficial i no oficial en fòrums de la comunitat.

Veient les dificultats per vincular el servidor de Fracsland [14] als treballs originals de *Geopieces* així com a la seva ampliació, era un objectiu fonamental del projecte tenir un *backend* i un servidor senzills, adaptables i funcionals, donant així la possibilitat de satisfer la comunicació entre professor i alumnat de manera ràpida. D'aquesta manera s'ha permès fer més èmfasi en el propi desenvolupament del videojoc i mantenir totes les dades persistents en remot. S'ha escollit ***Firebase***, una plataforma per al desenvolupament d'aplicacions web i aplicacions mòbils desenvolupada per Google, amb possibilitat d'integrar-se amb el motor gràfic de Unity.

4.1.1 *Firebase Realtime Database*

Firebase [22] és una base de dades de tipus NoSQL allotjada al núvol, sincronitzada amb tots els clients en temps real i amb persistència de dades sense connexió guardades en format JSON, és a dir, parelles *<key>:<value>* on la clau és una cadena de caràcters i els valors poden ser dels tipus descrits a més endavant. A part, s'hi pot associar la *Firebase Authentication* de manera que només els usuaris autenticats tinguin la possibilitat de llegir i/o escriure.

En el projecte s'ha fet servir aquesta eina degut a la seva fàcil i ràpida integració per a realitzar la lectura i escriptura de dades entre professor i alumnes, i durant les següents seccions es faran explícites aquestes operacions de comunicació sempre que sigui necessari. Cal mencionar, però, que els únics tipus de dades possibles per guardar en els valors dels arbres JSON a la Firebase són els següents [16]: ***string***, ***long***, ***double***, ***bool***, ***Dictionary <string, Object>***, ***List <Object>***.

Per tant, en cert moment s'han d'adaptar les dades de l'aplicació a algun d'aquests formats per poder serialitzar-les correctament.

4.1.2 Arquitectura per components de Unity

Qualsevol escena en un projecte en Unity [18] està formada per uns elements anomenats *GameObjects*. Aquests objectes, que poden ser de tipus GUI, 2D o 3D, són les peces fonamentals dels projectes i estan formats per diverses característiques que els diferència uns dels altres (*Components*). Es pot observar un exemple a la Figura 4.1. Aquestes components són variades: desde la posició i l'escala (*Transform*), fins a la capsa que els recobreix (*Collider*), passant per la component que determina la física de l'objecte (*Rigidbody*) o el

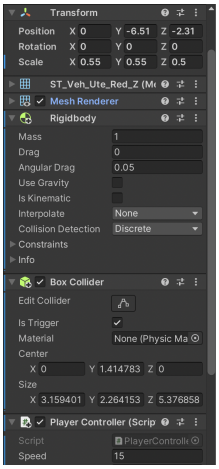

Figure 4.1: Components d'un *GameObject*.

text associat en cas que sigui un objecte de la GUI (*Text Mesh*). De totes maneres, l'element més important que els permet "funcionar" és el conjunt de *scripts* que tinguin adherits.

Els *scripts* de Unity són els arxius de codi en C# encarregats de modelar, mitjançant els mètodes heretats de la classe base *MonoBehaviour*, el comportament d'un GameObject. No obstant això, el llenguatge de programació C# és orientat a objectes i, per tant, és possible aplicar patrons de disseny i arquitectures de *software*.

4.2 Proposta de disseny de *software*

En aquest projecte es proposa l'arquitectura de disseny MVC (*Model - View - Controller*), treballada sovint al llarg de moltes assignatures de desenvolupament de programari. Es tracta d'un patró que permet categoritzar els objectes segons la seva funcionalitat a l'aplicació: per una banda, els que s'encarreguen de treballar sobre la **vista** del programa (V), per una altra, el **model**, designat a representar dades (M) i, finalment, els que estableixen la comunicació entre els altres dos amb el **controlador** (C). Així doncs, per adaptar-se al sistema de *scripts* de Unity, s'ha implementat el MVC de la següent manera:

- ***View.*** Carpeta del projecte que conté totes les escenes del videojoc (GUI + 3D). Aquestes, al seu torn, contenen tots els *scripts* derivats de *MonoBehaviour* que permeten el correcte funcionament dels *GameObjects* (càmera, personatges, enemics...). Són els fragments del codi que allotgen els algorismes destinats a moviments i rotacions, així com els encarregats de demanar als *controllers* les dades necessàries per tal d'actualitzar la GUI.

- ***Model.*** Representa els objectes i classes serialitzables del projecte. Modelen la informació associada als usuaris, així com les característiques de les missions i de les

construccions dissenyades. Es comuniquen amb la Firebase mitjançant els *controllers*, modificant així les seves dades. No hereten de la classe *MonoBehaviour*.

- ***Controller.*** Són els encarregats de llegir/escriure les dades de la Firebase, traduir-les en els objectes convenients del *Model* i traslladar la informació a les classes de la *View*. A més, gestionen la creació i autenticació d'usuaris, així com l'assignació i actualització de missions a cadascun i la persistència del joc. Al igual que el *Model*, aquests *scripts* no hereten de *MonoBehaviour*.

A la Figura 4.2 s'observa el diagrama d'aquesta arquitectura, indicant en color verd el paquet de la *view*, en color lila el *controller*, en color gris el *model* i en color blau la Firebase.

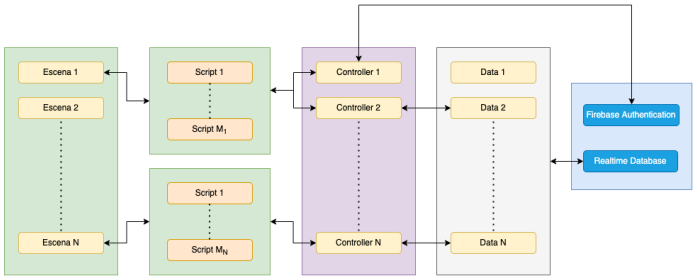

Figure 4.2: Diagrama de l'arquitectura MVC aplicada al projecte.

4.2.1 Estructura del model de dades

Tenint en compte la proposta de disseny del capítol anterior, s'ha decidit per la següent estructura de classes relacionades amb les dades de l'aplicació.

- ***User.*** Classe abstracta que representa els usuaris de l'aplicació. Disposa d'un nom d'usuari i una contrassenya (*strings*) com a atributs i deriva en les següents classes:
  - ***Designer / Player***. Classes que representen els professors i alumnes respectivament, i afegeixen la llista de missions dissenyades o la llista de missions assignades com a atribut (*MissionDesigner* i *MissionPlayer*). Implementen els mètodes d'escriptura a la base de dades declarats a la superclasse i tenen una sobrecàrrega al constructor segons si s'instancien de nou o com a resultat de la lectura de dades de la Firebase. Això es veurà més endavant.

- ***MissionDesigner.*** Classe que representa les dades de la missió que ha generat un *Designer*. Disposa dels següents atributs: nom de la missió i professor que la ha dissenyat (*strings*), nombre de figures que té la construcció (*integer*) i la llista de posicions d'aquesta (llista de *Vector3*), i un *Dictionary* dels *players* assignats amb clau un *string* del nom i que té com a valor la *MissionPlayer* associada a cada jugador.

- ***MissionPlayer.*** Caracteritza les dades de la missió assignada a un *player*, tenint com a atributs els mateixos que la *MissionDesigner* a excepció del diccionari, però afegint a més la llista de característiques associades a l'alumne (*strings*), l'inventari de figures (*integer*) i un objecte de tipus *Tatami* i *Football*.

Els dos tipus de missions també implementen mètodes de lectura/escriptura a la base de dades i tenen una sobrecàrrega al constructor per a instanciar objectes a partir de la lectura de la Firebase.

- ***Minigame.*** Classe que representa les dades associades als minijocs de *Tatami* i de *Football*, és a dir, un *Dictionary <int, bool>* que representa a quins nivells o onades d'enemics s'ha obtingut una figura necessària per a la construcció final.
  - ***Tatami / Football***. Classes que hereten de *Minigame* i omplen l'atribut que s'ha mencionat de la superclasse. Hi ha també una sobrecàrrega al constructor per a la instanciació a partir de lectura de la base de dades i s'implementen mètodes d'escriptura en aquesta.

4.3 Comunicació amb *Realtime Database*

A continuació es detalla la serialització d'aquestes dades i els processos de lectura i escriptura a la *Realtime Database*, tot involucrant els *controllers* necessaris i els *scripts* de la vista que ho requereixin.

4.3.1 Serialització de les dades

Les dades del videojoc estan representades amb objectes, on molts dels atributs no respecten els tipus designats a la secció 4.1.1. Per tant, és necessari tenir una estructura de serialització de les classes esmentades a l'apartat 4.2.1, de tal manera que sigui possible la seva persistència com a arbre JSON a la base de dades en remot.

La idea és que es serialitzi únicament els objectes de tipus *User*, tant els dissenyadors com els jugadors, ja que aquests tenen la resta de classes del model referenciades en els atributs. Si s'observa la classe *MissionPlayer*, es veu que els objectes de tipus *Tatami* i *Football* no compleixen amb els requisits de serialització definits a la secció 4.1.1, ja que la seva classe base, *Minigame*, disposa d'un *Dictionary <int, bool>* no vàlid per l'arbre JSON.

S'han creat noves classes, *SaveData*, associades a cadascuna de les dades mencionades anteriorment, permetent així la persistència dels atributs no vàlids pel format JSON.

1. Es reformulen els parells *<int, bool>* en una nova classe *SaveDataFigureInWave* que té com a atributs el nombre associat al nivell i el booleà de si ha obtingut la figura o no.

2. S'assigna una llista de *SaveDataFigureInWave* en les classes dels minijocs per substituir el *Dictionary <int, bool>* no vàlid. Per tant, en el moment de serialitzar un *Minigame* es crea un objecte de tipus *SaveDataMinigame* que omple la llista a partir dels parells del diccionari.

3. La *MissionPlayer* té associada una *SaveDataMissionPlayer* on els atributs *Tatami* i *Football* queden substituïts per dos de tipus *SaveDataMinigame*.

4. De manera similar, una *MissionDesigner* es vincula amb la nova classe *SaveDataMissionDesigner* que reemplaça el *Dictionary <string, MissionPlayer>* per *List <SaveDataMissionPlayer>*.

5. Finalment, es generen els objectes *SaveDataDesigner* i *SaveDataPlayer*, els quals tenen una llista de *SaveDataMissionDesigner* i de *SaveDataMissionPlayer* respectivament.

4.3.2 Esquema de *SaveData* i arbre JSON

Les classes de tipus *SaveData* i atributs d'aquesta queden guardats en parelles de tipus *<key>:<value>*, amb el nom de la variable i el seu valor respectivament. Tenint en compte les estructures exposades a les seccions 4.2.1 i a 4.3.1, es mostra a les Figures 4.3, 4.4 i 4.5 un exemple de *Player* serialitzat en l'arbre JSON.

Figure 4.3: *Player JSON.*

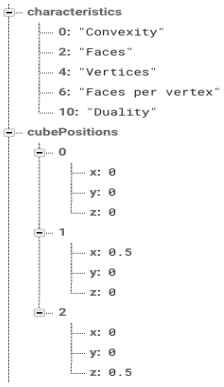

Figure 4.4: Dades de missió.

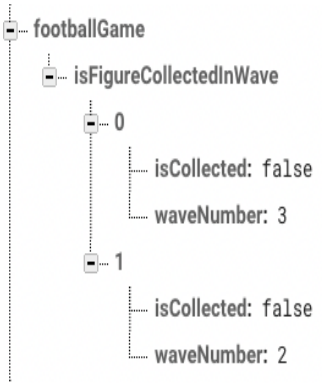

Figure 4.5: *Football*

4.3.3 Processos de comunicació

Quan es vol llegir/escriure dades de la Firebase s'ha de solucionar un problema de sincronia: els mètodes associats a la recuperació de dades o a l'autenticació són tasques asíncrones i sovint és necessari esperar a que finalitzin per continuar amb el flux general d'execució. Per tant, cal afegir a la capçalera dels mètodes asíncrons l'etiqueta *async* i indicar que l'objecte de retorn és de tipus *Task<>*. En la crida d'aquests mètodes s'ha d'afegir l'operador *await*, el qual permet esperar a que la tasca finalitzi i poder usar així el seu resultat.

Seguidament es mostren els procediments que involucren la persistència de dades: autenticació, assignació de missions i persistència de les dades del joc.

Autenticació d'usuari

Per a gestionar l'autenticació dels usuaris s'ha aplicat el patró *Singleton* al controlador *AuthController*, amb crides provinents tant del *RegisterScript* com del *LoginScript* del paquet de la vista, que estan associats a les pantalles de registre i de Log In.

Per a registrar i logejar un nou usuari cal passar per dos mètodes de la *Firebase Authentication*: *CreateUserWithEmailAndPasswordAsync(email, password)* i *SignInWithEmailAndPasswordAsync(email, password)* respectivament. Aquests mètodes, però, només s'encarreguen de registrar o autenticar un correu electrònic amb contrassenya per evitar duplicats d'usuaris, tot proporcionant un identificador aleatori format per nombres i lletres. Cal dir que en cap moment s'estan escrivint a l'arbre JSON les dades de l'usuari.

Per a registrar un nou usuari a la *Realtime Database* cal fer una crida d'escriptura i serialitzar en format JSON. Per tant, després de fer el registre i l'inici de sessió a la *Firebase Authentication*, el *AuthController* instància un *User* i fa cridar el mètode abstracte *WriteNewUserToDB()* implementat a les classes filles, el qual crea un objecte *SaveDataDesigner* o *SaveDataPlayer* segons el tipus. S'hi aplica la funció *ToJson(Object)* convertint els noms dels atributs i els seus valors en parelles *<key>:<value>* per després fer l'escriptura amb el mètode *SetRawJsonValueAsync(string json)*, que queda guardat com a valor dins una fulla de l'arbre JSON que té com a clau l'identificador de l'usuari autenticat.

Per a iniciar sessió amb un usuari escrit a la *Realtime Database*, cal deserialitzar les dades de tipus JSON a *SaveDataDesigner* o *SaveDataPlayer* i assignar-les a una instància que hereti del tipus *User*. Després del *Sign In* explicat abans, s'obté l'identificador de l'usuari actual de la *Firebase Authentication* i es demana a l'arbre el valor associat a aquesta clau amb la funció asíncrona *GetValueAsync()* i les dades en format JSON amb *GetRawJsonValue()*. Aleshores, es crida al constructor alternatiu de *Player* o *Designer* que té com a paràmetre un objecte de tipus *SaveDataPlayer* o *SaveDataDesigner*, i s'assignen aquestes noves dades a l'atribut *currentUser* de tipus *User* del *AuthController*, tenint a disposició en totes les escenes les dades de l'usuari actual. Tot aquest procediment es pot veure a la figura 4.6.

Assignació de missions

En el cas que un *Designer*/professor vulgui assignar la missió que ha construït, l'*script NewMissionManager* de l'escena de l'Editor 3D es posa en contacte amb el *MissionController*. Li passa el nom de la missió, el *currentUser*, la llista de posicions de la construcció i un diccionari amb el nom d'usuari de cada *player* i la llista de característiques que se li han adjudicat. El controlador fa una instància de *MissionDesigner* i un diccionari de *MissionPlayer* amb cada alumne com a clau, i ho delega en el *UserController*.

Com l'usuari actual és el dissenyador, aquest controlador li pot afegir la nova missió directament a la llista. El *currentUser* fa novament el procés d'escriptura de l'usuari a

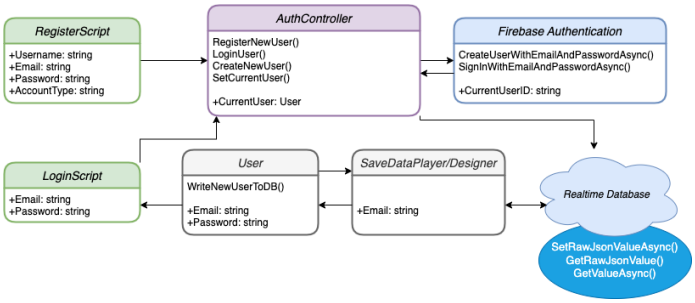

Figure 4.6: Diagrama de classes relacionades amb el procés d'autenticació. A cada classe només es mostren aquells mètodes essencials en el procés.

la *Realtime Database* a la fulla amb clau l'identificador actual, que és sempre accessible després de l'autenticació.

En canvi, per assignar als estudiants els objectes de tipus *MissionPlayer*, el *UserController* ha de fer la lectura asíncrona a la base de dades de tots els usuaris que siguin de tipus *Player* amb els seus identificadors i, per cadascun d'aquests, afegir la missió a la llista i fer el procés d'escriptura d'usuari usant aquests *id's*. Es pot veure a la Figura 4.7.

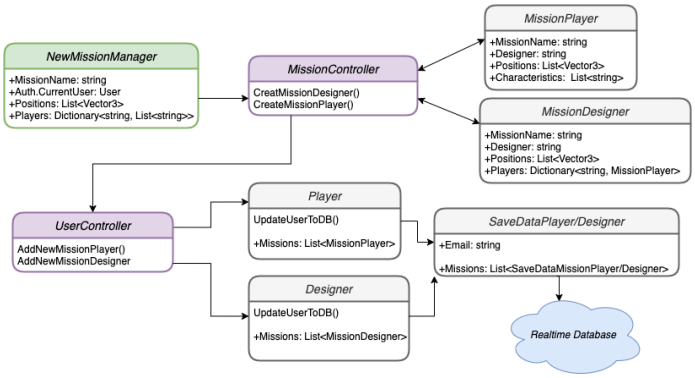

Figure 4.7: Diagrama de classes relacionades amb l'assignació de missions. A cada classe només es mostren aquells mètodes essencials en el procés.

Persistència del joc

Un cop els jugadors recullen figures als *Minigame*, ja sigui de manera continuada en el *GeoCollect* o bé seguint el diccionari d'onades a *GeoTatami / GeoFootball*, on les preguntes de temàtica geomètrica s'han llegit de manera asíncrona, cal actualitzar aquesta informació tant a nivell de *Player* com a nivell de *Designer*. En aquest cas, però, l'usuari actual és un/a estudiant.

En la selecció de missió, el *MissionListPlayerScript* de la vista fa una crida a *MissionListController* (on s'ha aplicat també el patró *Singleton*) per a fer una lectura del llistat de missions i mostrar-les. Un cop seleccionada, aquesta s'assigna com a valor de l'atribut *currentMissionPlayer* del controlador. Quan es recullen figures en els minijocs, s'actualitzen les dades de l'inventari i del diccionari d'onades accedint a aquesta variable i, finalment, se'n fa la serialització. Primer, es copia aquesta missió actual a l'associada a la llista del *currentUser* que encara roman amb les dades antigues, i es torna a fer una escriptura de l'usuari a la base de dades.

Però es necessita l'identificador de l'usuari dissenyador per a actualitzar-ne les dades. Es delega en *UserController* la lectura asíncrona del *Designer* i l'*id*, a partir de l'atribut *email* del professor de la *currentMissionPlayer*. A continuació, el controlador fa una crida a *UpdateMission(player, currentMission, designerID)* de la classe *Designer*, actualitzant així el diccionari de *MissionPlayer* amb l'usuari actual i la *currentMissionPlayer*. Finalment, es fa el procés d'escriptura a la base de dades ajudant-se de l'identificador obtingut abans. S'observa a la Figura 4.8.

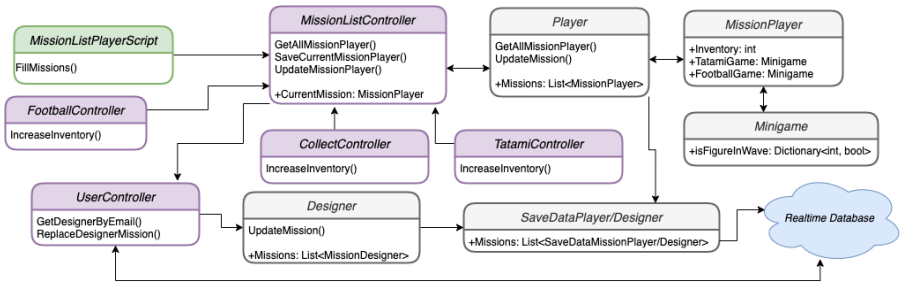

Figure 4.8: Diagrama de classes relacionades amb la persistència del joc. A cada classe només es mostren aquells mètodes essencials en el procés.

4.4 Principals algorismes dels escenaris

Es mostren a continuació els algorismes més rellevants a cada escena principal del joc: l'Editor i el Constructor 3D, i els minijocs de *Collect*, *Tatami* i *Football*. S'exposen els comportaments de la càmera en els diversos escenaris, la creació de la matriu associada a les graelles en el *Sudoku* 3D i, finalment, la tria de desafiaments.

4.4.1 Edició i construcció 3D

La primera dificultat que s'ha trobat en el moment de realitzar una construcció 3D, ja sigui al dissenyar-la o al replicar-la, és saber gestionar la rotació de la càmera i posicionar-la a cada *frame* quan s'afegeixen figures noves. Per tant, la solució passa per aplicar el concepte d'interpolació explicat a l'apartat 2.5.4, tot usant dades de la capsa mínima contenidora que engloba la construcció. Es tenen els següents paràmetres:

- P_{obs}, P_{box}, V_{UP}, $P_{VRP} \in \mathbb{R}^3$ són la posició de la càmera, el punt més pròxim de la capsa mínima, el vector **VUP** i el punt d'observació de la càmera, respectivament,
- $I_{scroll} \in [-1, 1]$ l'*input* de la rodeta del ratolí de l'usuari,
- $R_{obs} \in \mathbb{H}$ la rotació actual de l'observador, detallada a la secció 2.5.1,
- t, r, d_{min}, $d_{max} \in \mathbb{R}_+$ el temps entre l'anterior *frame* i l'actual (*deltaTime*), la distància entre figures, i una cota inferior i superior de distància de posicionament amb la construcció, respectivament.

A cada *frame*, doncs, s'empra l'Algorisme 1 per a posicionar la càmera i suavitzar-ne la rotació.

Algorithm 1: Recàlcul de posició i rotació: Editor/Constructor 3D

Data: P_{obs}, P_{box}, V_{UP}, $P_{VRP} \in \mathbb{R}^3$, $R_{obs} \in \mathbb{H}$, $I_{scroll} \in [-1, 1]$, t, r, d_{min}, $d_{max} \in \mathbb{R}_+$
Result: void
$d \leftarrow \|P_{obs} - P_{box}\|$;
$V_{PN} \leftarrow \overrightarrow{P_{VRP}P_{obs}}$;
$R_{target} \leftarrow Quaternion.LookRotation(-V_{PN}, V_{UP})$;
$R_{obs} \leftarrow Quaternion.Slerp(R_{obs}, R_{target}, t)$;
if $I_{scroll} > 0$ *and* $d \geq d_{min}$ **then**
    $P_{obs} \leftarrow P_{obs} - rV_{PN}$;
else if $I_{scroll} < 0$ *and* $d \leq d_{max}$ **then**
    $P_{obs} \leftarrow P_{obs} + rV_{PN}$;

S'observa de l'algorisme que la rodeta del ratolí gestiona el *zoom* de l'observador dins dels límits d_{min} i d_{max}, amb salts de distància r.

Es veu també l'aplicació de la interpolació esfèrica entre el quaternió que representa la rotació actual i la rotació objectiu, la qual té la direcció de visió cap al centre de la capsa mínima i està generada gràcies a la funció *LookRotation()*, i ve donada per la funció ja implementada *Slerp* explicada a la secció 2.5.4. Ambdós mètodes es troben a la llibreria *Quaternion* [15].

Un cop construïda la figura geomètrica, queden determinades les posicions de cada peça i es poden agrupar en el conjunt $V \subset \mathbb{R}^3$. Aleshores, per a comprovar la validesa de la construcció, es genera el graf G explicat a la secció 2.2.1 considerant $\mathbf{d} = 0.5$, que és la distància on es situen els cubs entre ells dos a dos i, mitjançant els algorismes exposats en l'apartat mencionat, es verifica la connexió o no del graf.

En el projecte actual, les construccions estan formades per cubs situats a distància 0.5 en coordenades globals de món. Per tant, en qualsevol eix es pot comptar el nombre de sòlids platònics que hi ha usant $2(max_{position} - min_{position}) + 1$. Seguint aquesta idea, en l'escenari del Constructor 3D de l'alumne, es planteja un algorisme per generar les matrius de projecció (XY, ZY, XZ) de mateix tamany totes elles. A l'Algorisme 2 es veu el cas per a projectar la construcció en la matriu associada al pla XY, tot indicant a cada posició el nombre de figures que hi ha a cada parell de coordenades (x, y):

Algorithm 2: Matriu de projecció XY: Constructor 3D

Data: V
Result: M
$D \leftarrow \max_{k=x,y,z} \left\{ \max_{0 \leq i \leq \#V} \{V_i.k\} - \min_{0 \leq i \leq \#V} \{V_i.k\} \right\}$;
$N \leftarrow 2D + 1$;
for $y \leftarrow 0$ *to* N **do**
  $m_y \leftarrow \frac{y}{2} + \min_{0 \leq i \leq \#V} \{V_i.y\}$;
  for $x \leftarrow 0$ *to* N **do**
    $S_z \leftarrow 0$;
    $m_x \leftarrow \frac{x}{2} + \min_{0 \leq i \leq \#V} \{V_i.x\}$;
    for $v \in V$ **do**
      if $v.x == m_x$ *and* $v.y == m_y$ **then**
        $S_z \leftarrow S_z + 1$;
      end
    end
    $M_{N-y-1,x} \leftarrow S_z$;
  end
end
return M

Si s'analitza l'algorisme, es pot veure que té complexitat $\mathcal{O}(n^3)$ ja que es fa un recorregut per una matriu $n \times n$, però comprovant per a cada parell (x, y) el nombre de peces que hi ha en la construcció de dimensió n també. De manera anàloga es generen les matrius de projecció ZY i XZ.

4.4.2 *Tatami* i *Football*

Si s'analitza ara els escenaris dels minijocs com el *GeoTatami*/*GeoFootball*, es pot observar que el comportament de la càmera no és el mateix que l'exposat a l'Algorisme 1. Això és degut a que ara es busca afegir una angle determinat a la càmera ja que és el propi usuari l'encarregat de controlar-la. En canvi, en l'anterior escena es volia suavitzar la rotació automàtica de la càmera en el moment d'afegir noves figures en la construcció. Per tant, es tenen els següents paràmetres:

- $P_{player} \in \mathbb{R}^3$ la posició del jugador a l'escena,
- $I_{vertical}$, $I_{horizontal} \in [-1, 1]$ l'*input* al clickar WS o AD del teclat, respectivament,
- $R_{obs} \in \mathbb{H}$ la rotació actual de l'observador,
- t, $m \in \mathbb{R}_+$ el temps entre l'anterior *frame* i l'actual (*deltaTime*), la massa associada a la component *Rigidbody* del jugador, respectivament.

I a cada *frame* s'aplica l'algorisme 3 per a afegir rotació a l'observador.

En aquest cas s'ha aplicat la funció *Rotate()* de Unity on l'*input* horitzontal de l'usuari simbolitza l'angle de gir a cada *frame*, i el vector *forward* del quaternió R_{obs} és la direcció de

Algorithm 3: Recàlcul de posició i rotació: Tatami/Football

Data: P_{player}, $V_{UP} \in \mathbb{R}^3$, $R_{obs} \in \mathbb{H}$, $I_{horizontal}$, $I_{vertical} \in [-1, 1]$, $m, t \in \mathbb{R}_+$
Result: void
if $I_{horizontal} \neq 0$ **then**
    $R_{obs} \leftarrow Rotate(V_{UP}, I_{horizontal})$;
    $V_{PN} \leftarrow -R_{obs}.forward$;
end
if $I_{vertical} \neq 0$ **then**
    $v \leftarrow t\frac{I_{vertical}}{m} V_{PN}$;
    $P_{player} \leftarrow P_{player} + vt$;
end

visió de la càmera. Per a modificar la posició del jugador s'aplica les fórmules de mecànica clàssica, considerant l'*input* vertical com la força en Newtons que s'aplica al personatge.

4.4.3 *Challenges*

S'exposa a la Figura 4.9 com estan estructurats els desafiaments relacionats amb les característiques geomètriques que permet treballar el primer minijoc.

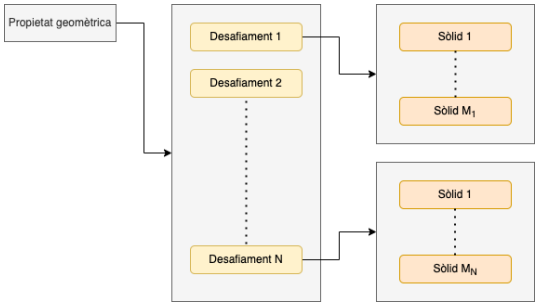

Figure 4.9: Estructura dels desafiaments. Implementats a la classe *Challenges* del *Model* i escollits mitjançant el controlador *ChallengesListController*.

A més, s'ha considerat necessari esmentar l'algorisme de tria de característiques vàlides pel professor a l'hora de generar la missió. En el minijoc de *GeoCollect* es necessita classificar els objectes que apareixen en pantalla segons tres atributs associats a tres característiques geomètriques diferents. Pot passar fàcilment que els sòlids que posseeixen una propietat geomètrica estiguin inclosos en el conjunt de sòlids que en tenen una altra. Això provocaria una partida no resoluble i es requereix la necessitat de desenvolupar l'Algorisme 4 per a comprovar si són compatibles les característiques escollides pel professor. Partint de la base que cal escollir-ne un mínim de tres, els paràmetres d'entrada són:

- $N_{attempt} \in \mathbb{N}$ el nombre d'intents d'execució de l'algorisme,

- S_{prop} la llista de característiques escollides pel professor,
- $S_{challenges}$ el diccionari de desafiaments relacionats amb propietats geomètriques de les figures.

Algorithm 4: Selecció de propietats geomètriques

Data: $N_{attempt} \in \mathbb{N}$, S_{prop}, $S_{challenges}$
Result: bool $found$
$found \leftarrow false$;
$S_{figures} \leftarrow \emptyset$;
while *not found and* $N_{attempt} \neq 0$ **do**
  $S \leftarrow S_{prop}.Random(3)$;
  for $p \in S$ **do**
    $c \leftarrow S_{challenges}(p).Random(1)$;
    $S_{figures} \leftarrow S_{figures} \cup \{c.figures\}$;
  end
  if *not ThereIsSubset* $(S_{figures})$ **then**
    $found \leftarrow true$;
  end
  $N_{attempt} \leftarrow N_{attempt} - 1$;
end
return *found*

D'aquesta manera es limita la ineficiència que provocaria fer una cerca de totes les combinacions de tots els desafiaments de totes les característiques disponibles, comprovant per a cada cas si hi ha subconjunts de figures inclosos l'un amb els altres. Es tracta d'una combinació sense repetició que provocaria una complexitat de $\mathcal{O}(\binom{mn}{3})$, amb m el nombre de propietats i n el nombre de desafiaments associats. La crida a *ThereIsSubset* fa referència a l'Algorisme 5 per a comprovar la no inclusió dels conjunts de figures entre ells.

Algorithm 5: Comparació de subconjunts de figures

Data: $S_{figures}$
Result: bool *subset*
$subset \leftarrow false$;
for $S_i \in S_{figures}$ **do**
  $W \leftarrow \emptyset$;
  for $S_j \in S_{figures} - S_i$ **do**
    $W \leftarrow W \cup S_j$;
  end
  if $S_i \subset W$ **then**
    $subset \leftarrow true$;
    break;
  end
end
return *subset*

Capítol 5

Simulacions i resultats

5.1 Resultats

S'han desenvolupat les pantalles de *Register* i *Log In*, on l'usuari es registra i s'autentica amb un correu electrònic, un nom d'usuari i una contrassenya de mínim 6 caràcters. Es pot veure a la Figura 5.1.

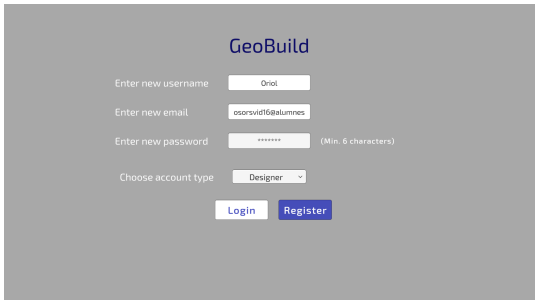

Figure 5.1: Pantalla de registre.

5.1.1 Vista del *Designer* / professor

S'ha realitzat l'escena que mostra la llista de missions dissenyades pel professor. Ho podem veure a la Figura 5.2.

Es poden veure també les estadístiques de cada alumne a la Figura 5.3.

Amb l'editor 3D s'ha assolit satisfactòriament l'objectiu principal, però permetent només la construcció amb hexaedres regulars. Per afegir o eliminar un cub es fa mitjançant click esquerra o dret del ratolí respectivament. Per a girar al voltant de la figura que es construeix s'utilitzen les tecles WASD, i per fer *zoom* s'usa la rodeta del ratolí. El

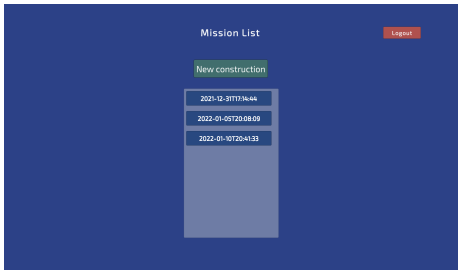

Figure 5.2: Missions del professor.

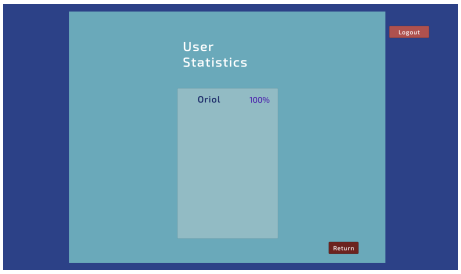

Figure 5.3: Estadístiques dels usuaris.

professor pot fer *Reset* si ho troba convenient, o bé comprovar que l'estructura que ha construït és vàlida i passar a la següent escena. S'observa a les Figures 5.4 i 5.5 respectivament.

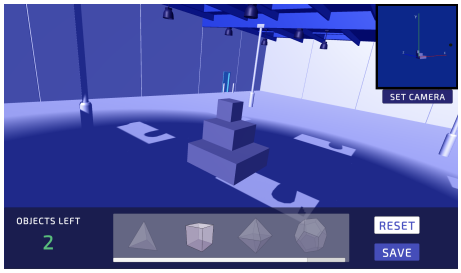

Figure 5.4: Construcció vàlida.

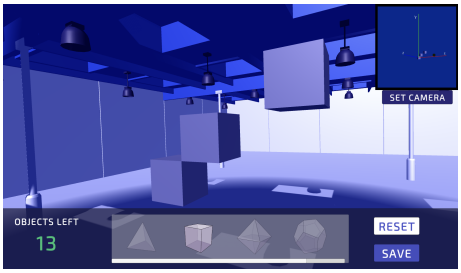

Figure 5.5: Construcció invàlida.

S'ha implementat la pantalla de crear missió com havia estat pensada inicialment. Es mostra aquest menú a la Figura 5.6.

Figure 5.6: Creació de la missió.

5.1.2 Vista del *Player* / estudiant

La llista de missions que té assignades un alumne es poden veure a les Figures 5.7 i 5.8, on si es pressiona la tecla TAB, el jugador pot veure l'inventari de figures que porta.

Figure 5.7: Missions de l'alumne.

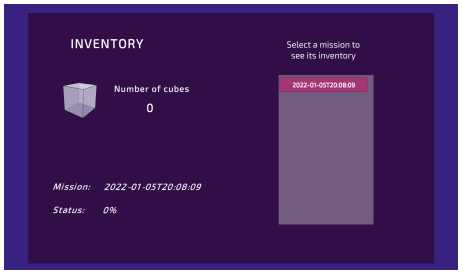

Figure 5.8: Inventari.

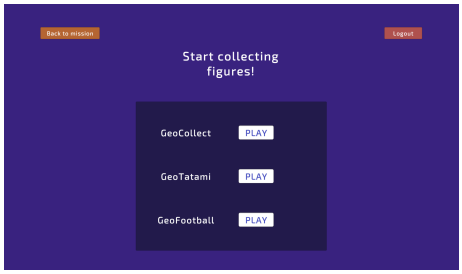

Figure 5.9: Selecció de minijoc.

Si se selecciona una missió, es dirigeix cap a la tria de minijoc. Aquesta pantalla és la mateixa independentment del repte triat anteriorment, però el *farmeig* que s'obtingui dins les 3 opcions anirà enllaçat amb la construcció que hi té assignada. Es pot observar a la Figura 5.9.

S'han assolit els requeriments funcionals de *GeoCollect*, on el *player* ha d'intentar recollir les figures amb el camió, tenint en compte la perspectiva i controlant amb les tecles WASD. Es pot veure a la Figura 5.10.

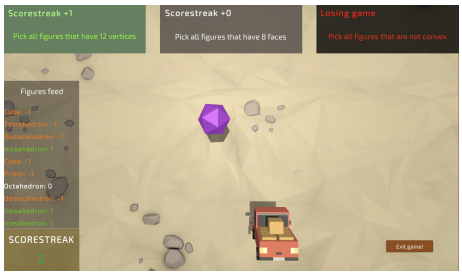

Figure 5.10: *GeoCollect*.

Figure 5.11: Indicador de figura daurada.

L'alumne pot veure en tot moment a l'esquerra de la pantalla la llista de figures que va recollint fins esperar a que caigui el cub daurat, tal i com s'observa a la Figura 5.11.

En el cas de *GeoTatami* els controls són novament amb les tecles WASD, però en aquest cas les tecles A i D estan destinades a rotar la càmera i amb W, S es mou endavant i endarrere en una sola direcció. Es veu a la Figura 5.12.

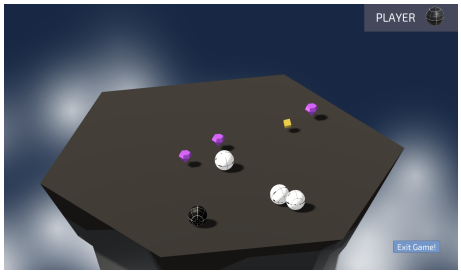

Figure 5.12: *GeoTatami.*

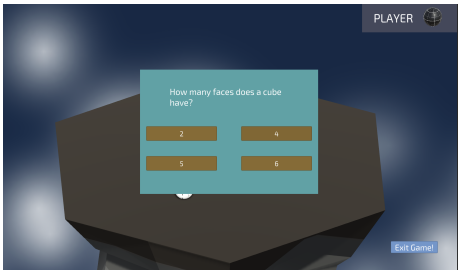

Figure 5.13: Pregunta test de geometria.

S'ha aconseguit també la lectura correcta de les qüestions geomètriques a partir del *bonus* ressaltat de color lila (vegeu la Figura 5.13). A *GeoFootball* els controls són els mateixos i en tot moment l'usuari pot veure el marcador a dalt a l'esquerra (vegeu la Figura 5.14).

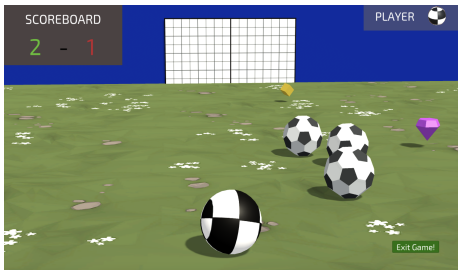

Figure 5.14: *GeoFootball.*

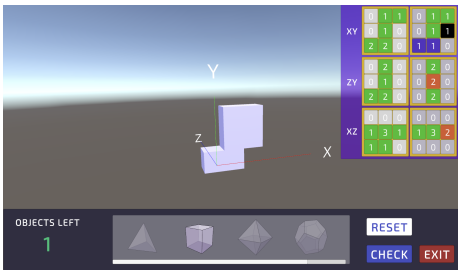

Figure 5.15: GUI de construcció de l'alumne.

En completar l'inventari associat a una missió, en comptes d'aparèixer un percentatge al costat d'aquesta a la pantalla de *Player Mission List* (Figura 5.7), apareix un botó daurat que condueix al *Sudoku* 3D. Aquest s'ha implementat segons l'esperat, mostrant les tres graelles objectiu i les tres graelles de guiatge per l'alumne. Vegeu la figura 5.15.

5.2 Test de competències

S'ha provat el videojoc amb un noi de 4t de la ESO per tal de comprovar la millora que s'assoleix en els conceptes de la geometria i en la percepció espacial de l'estudiant. El test i els resultats es poden veure a l'Apèndix B.

Consisteix en dos apartats: el reconeixement i la construcció de figures. En el primer, es demana que s'ompli la Taula 2.1 amb les característiques exposades a 2.1.1. Després, es realitzen 5 partides a *GeoCollect* i es torna a fer el mateix test. En el segon, es planteja a l'estudiant que intenti construir 2 vegades una figura usant el *GeoSudoku*.

Capítol 6

Conclusions i treball futur

El projecte tenia inicialment dos objectius ben clars: la creació d'un videojoc en 3D que serveixi com a eina en l'aprenentatge de la geometria tridimensional a Secundària, i que el professorat sigui capaç de personalitzar aquesta experiència de joc segons les necessitats de cada alumne, tot visualitzant el progrés de cadascun.

En el primer cas, s'ha assolit satisfactòriament l'objectiu: s'han implementat 3 minijocs que permeten solucionar reptes de reconeixement de sòlids geomètrics, i s'ha desenvolupat un constructor 3D a partir de proporcionar-li a l'alumne 3 projeccions de la figura objectiu.

En el cas de l'edició per part del professor, s'ha aconseguit gairebé de manera completa. S'ha creat un editor 3D on es pot generar la figura objectiu i una interfície de selecció de propietats geomètriques per assignar a cada estudiant de manera individualitzada. Tot i així, l'editor actualment només disposa de la possibilitat de construir mitjançant cubs i no amb tots els sòlids platònics.

La comunicació entre els dos usuaris s'ha assolit correctament, permetent l'autenticació d'aquests i mantenint les dades persistents en remot, de tal manera que l'estudiant rep les missions assignades pels professors i, aquests, al seu torn, reben el percentatge de progrés dels alumnes. Tot i així, s'ha observat que actualment la vinculació de Firebase amb Unity per aplicacions d'escriptori està en fase beta, cosa que dificulta per exemple el seu funcionament en MacOS.

L'exploració dels conceptes matemàtics treballats al capítol 2 han estat imprescindibles per a la consecució del desenvolupament, permetent comprendre el funcionament dels quaternions de Unity i com generar una capsa que recobreixi la construcció.

En quant a treball futur, una direcció a seguir és complementar l'editor 3D amb la resta de sòlids platònics, entre d'altres políedres senzills com els prismes. En un altre sentit, seria bona idea que el professor pogués escriure bancs de preguntes noves en el moment de la creació de la missió i no tan sols seleccionés les propietats geomètriques a treballar. Finalment, es podria adaptar el videojoc a una xarxa local, de tal manera que no es requereixi la connexió a internet i les dades quedin persistents en una mateixa classe.

Apèndix A

Manual d'execució

S'exposen a continuació les necessitats que ha de tenir l'entorn per a fer funcionar el videojoc i el procediment de descàrrega i execució.

A.1 Requisits

- Es requereix de Windows 10 per a poder fer ús de les característiques de la Firebase.
- Es requereix de connexió a internet per a poder registrar-se i iniciar sessió en el joc.
- El dispositiu ha de tenir un *hardware* estàndard sense necessitat de tenir prestacions d'alta gamma.

A.2 Descàrrega i execució

1. Descarregar l'arxiu .zip en la *release* del següent enllaç:
   https://github.com/OriolSors/Geometry-Videogame-Exe/tree/main.
2. Descomprimir-lo: s'obté un arxiu README.txt, l'executable Unity i altres arxius necessaris.
3. Executar el videojoc desde la mateixa carpeta. Si salta l'avís "Aplicació desconeguda", fer click a "Més informació" i permetre l'execució de totes maneres.

El codi font es pot trobar a l'enllaç:

https://github.com/OriolSors/Geometry-Videogame.

Apèndix B

Test de competències

S'explica a l'estudiant les nocions formalitzades a la secció 2.1.1 de manera adaptada, se li mostren imatges dels 5 sòlids platònics i, posteriorment, se li proporciona la graella de la Figura B.1 a l'inici de l'exercici sense cap recurs informatiu disponible.

Sòlid / Propietat	Convexitat	Regularitat	Cares	Arestes	Vèrtexs	Característica d'Euler	Vèrtexs per cara	Cares per vèrtex	Simetria	Uniforme	Poliedre dual
Tetraedre											
Cub											
Octaedre											
Dodecaedre											
Icosaedre											

Figure B.1: Graella de sòlids platònics.

Se li permet un temps de 10 minuts per provar d'omplir-la i s'obtenen els resultats de la Figura B.2.

Sòlid / Propietat	Convexitat	Regularitat	Cares	Arestes	Vèrtexs	Característica d'Euler	Vèrtexs per cara	Cares per vèrtex	Simetria	Uniforme	Poliedre dual
Tetraedre	SI	SI	4	3	4	5	3	3	Axial, Especular	SI	Octaedre
Cub	SI	SI	6	12	8	2	4	3	Axial, Especular	SI	
Octaedre	NO	SI	8	20	8	-4	3	3	Axial, Especular	SI	Tetraedre
Dodecaedre	NO	SI	12		12		5		Axial, Especular	SI	
Icosaedre	NO	SI	20		20		3		Axial, Especular	SI	

Figure B.2: Graella prèvia de sòlids platònics.

Es pot extreure que d'un inici ha entès ràpidament les simetries i la regularitat dels sòlids abans de jugar, però no ha detectat la convexitat de tots els sòlids platònics. Per altra banda, ha entès el concepte de dualitat d'un políedre però els resultats són incorrectes degut a una acumulació d'errors en quant a nombre de vèrtexs i arestes.

Tot seguit es deixa a l'estudiant fer 5 partides a *GeoCollect* amb totes les característiques disponibles (aproximadament 20 minuts). Posteriorment, se li demana omplir des de zero un altre cop la graella i s'obtenen els resultats de la Figura B.3.

S'observa que ha corregit la classificació de convexitat dels sòlids però ha confós els polígons que formen el dodecaedre i l'icosaedre, en contraposició al resultat correcte que

Sòlid / Propietat	Convexitat	Regularitat	Cares	Arestes	Vèrtexs	Característica d'Euler	Vèrtexs per cara	Cares per vèrtex	Simetria	Uniforme	Poliedre dual
Tetraedre	SI	SI	4	6	4	2	3	3	Axial, Especular	SI	Dodecaedre
Cub	SI	SI	6	12	8	2	4	3	Axial, Especular	SI	Octaedre
Octaedre	SI	SI	8	12	6	2	3	4	Axial, Especular	SI	Cub
Dodecaedre	SI	SI	12		12		3	3	Axial, Especular	SI	Tetraedre
Icosaedre	SI	SI	20	30	20	10	5	5	Axial, Especular	SI	

Figure B.3: Graella posterior de sòlids platònics.

havia obtingut prèviament. El concepte de dualitat l'ha mantingut present però novament el resultat és incorrecte degut a errors previs.

La segona activitat consisteix en fer la construcció de la Figura B.4 dues vegades i mesurar el temps que es triga. Com la posició del cub inicial és aleatòria, l'alumne ha de tornar a orientar-se i no és immediata la solució.

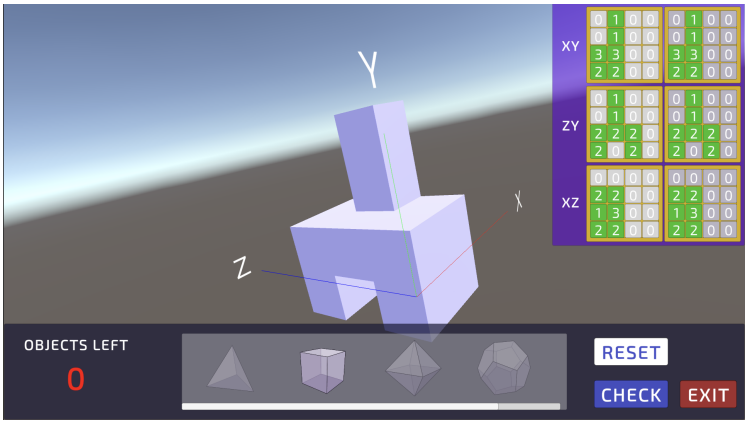

Figure B.4: Figura objectiu a construir.

En el primer intent, s'ha assolit satisfactòriament la construcció en un temps de 5.22 minuts. En el segon, s'ha reduït considerablement fins a 1.47 minuts.

Observacions:

- S'ha hagut de traduir a l'inici de cada partida de *GeoCollect* les característiques bona, neutral i incorrecta ja que aquestes apareixen en anglès, com tota la resta del videojoc. El noi ha tingut problemes amb la perspectiva al recollir les figures (algunes les perdia degut a un mal control del personatge). Les primeres 2 jugades s'han hagut de repetir ja que han durat un temps curt: s'ha perdut la partida per recollir objectes incorrectes al principi.
- La primera construcció realitzada ha trigat força degut a la dificultat d'adaptar-se als controls de la càmera (li costava posicionar-se) i a l'intent de situar cubs fora de l'escenari (l'indicador de "posició incorrecta" ha sortit amb freqüència al llarg del primer intent). A la segona partida ja no ha tingut problemes i senzillament ha anat posant i eliminant els cubs que considerava.

Referències

[1] Lengyel, E., & Smith, E. (2012). Mathematics for 3D game programming and computer graphics / Eric Lengyel; edited by Emi Smith (3rd Ed.). Cengage Learning.

[2] Lengyel, Eric (September 2016). Foundations of Game Engine Development, Volume 1: Mathematics. Terathon Software.

[3] Vince, J. (2021). Quaternions for Computer Graphics. Springer London, Limited.

[4] Baker, M. (2021). Mathematics and Computing - Martin Baker. Retrieved 7 November 2021, from http://www.euclideanspace.com/

[5] Coxeter. (1973). Regular polytopes / H.S.M. Coxeter (3rd ed). Dover.

[6] Hatcher. (2002). Algebraic topology / Allen Hatcher. Cambridge University Press.

[7] Rausell Seibert. (2016). Geopieces: joc seriós per l'aprenentatge de la geometria. Part 2D.

[8] Cebrián Gres. (2016). Geopieces: Joc seriós per l'aprenentatge de la geometria. Part 2D-3D.

[9] Zardaín Rodríguez. (2017). Ampliación del juego serio Geopieces para el aprendizaje de la geometría en 3D.

[10] WisWeb applets. Retrieved October 2021, from http://www.fi.uu.nl/wisweb/applets/mainframe_en.html

[11] Building with blocks. Retrieved October 2021, from http://www.fisme.science.uu.nl/toepassingen/00724/#

[12] Isometric Drawing Tool. Retrieved October 2021, from https://www.nctm.org/Classroom-Resources/Illuminations/Interactives/Isometric-Drawing-Tool/

[13] Unity Learn. Retrieved September 2021, from https://learn.unity.com/

[14] Muriel Ordóñez. (2016). Fracsland: joc seriós per aprendre fraccions.

[15] Unity Documentation. Retrieved September 2021, from https://docs.unity3d.com/Manual/index.html

[16] Firebase Documentation. Retrieved October 2021, from https://firebase.google.com/docs

[17] Requicha, A.A.G. (1980). Representations for Rigid Solids: Theory, Methods, and Systems. ACM Computing Surveys.

[18] Unity. Last access January 2022, from https://unity.com/es

[19] Unreal Engine. Last access January 2022, from https://www.unrealengine.com/en-US/

[20] Android Studio. Last access January 2022, from https://developer.android.com/studio/intro?hl=es-419

[21] Xcode. Last access January 2022, from https://developer.apple.com/xcode/

[22] Firebase. Last access January 2022, from https://firebase.google.com/

[23] Leiserson, Rivest, R. L., Cormen, T. H., & Stein, C. (2001). Introduction to algorithms (2nd ed.). MIT Press.

[24] Mark H. Ashcraft. (2002). Math Anxiety: Personal, Educational, and Cognitive Consequences. Current Directions in Psychological Science.