Pong

Seminarska naloga pri predmetu Računalniška grafika

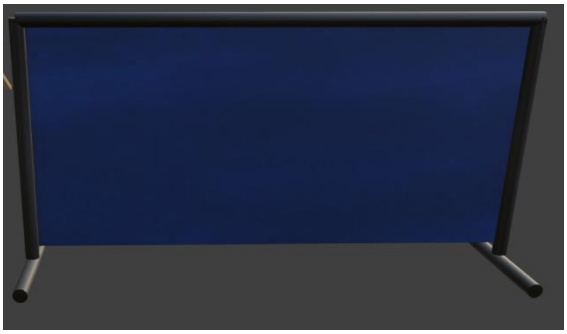
Jernej Rigler (63230280), Tim Hajdinjak (63230099), Žan Petkovšek (63230244)

Ljubljana, 2025

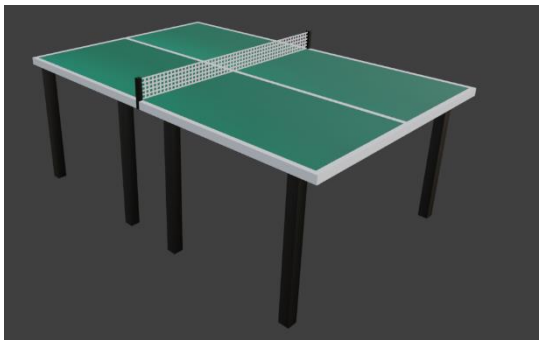
Objekti

Vse objekte (lopar, miza, stoli, glavna luč) smo ustvarili v Blenderju, pri čemer smo uporabili preproste primitivne mesh-e. Vsakemu objektu smo dodali teksture z uporabo slik v formatu .png ter jih ustrezno postavili v svet. Naš cilj je bil ustvariti čim manj objektov in jih narediti čim bolj preproste, da bi zagotovili optimalno delovanje aplikacije in predvsem zmanjšali zahtevnost upodabljanja. Po zaključku modeliranja in teksturiranja smo celoten svet izvozili v format .gltf in ga tako naložili v samo aplikacijo.

Slika 1: Stol



Slika 2: Pregrada



Slika 3: Miza

Slika 4: Končni izgled sveta

Kamera

V igri imamo implementirani dve kameri:

- **Fiksna kamera za mizo**: omogoča premikanje (po krožnici) preko drsanja miške, pri čemer je pogled centriran okoli mize, z rotacijo po radiju s sredino mize kot izhodiščem. S tem se lahko obrača okoli mize, da jo lahko vidi pod kotom.
- **Prosta (prvoosebna) kamera, ko nismo za mizo**: Ta kamera omogoča prosto premikanje po prostoru z uporabo tipk WASD, kar omogoča igralcu, da se svobodno sprehaja po svetu. Pogled se prosto usmerja z miško, kar zagotavlja prvoosebno izkušnjo.

Za prestop iz ene v drugo stisnemo tipko Enter, ko se približamo mizi.

Luči

V igri smo implementirali 11 luči, ena glavna, ki se nahaja na stropu nad mizo (osvetljuje cel prostor z belo barvo), ter 5 modrih ter 5 rdečih stranskih luči na steni (za dodaten nivo ambienta, z veliko manjšo intenziteto). Za senčenje smo uporabili osnovni Phongov osvetlitveni model, za njegov doseg smo razširili medpomnilnik renderpass-a in dopolnili senčilnik.

Slika 5: Svet po osvetlitvi

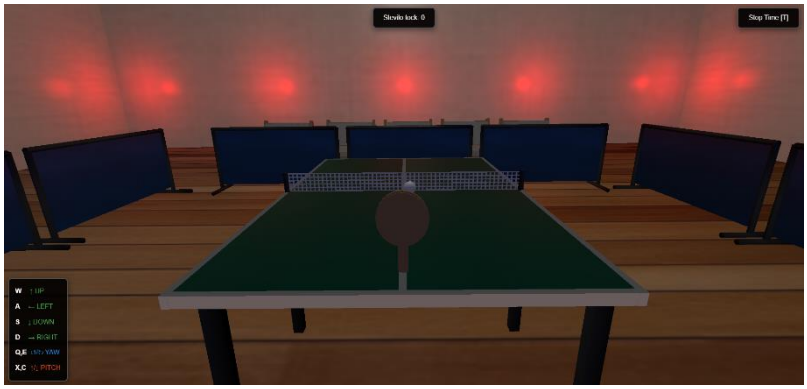
Game loop

Osnovni koncept igre je odboj servisa pri igri pingpong. Na začetku postavimo žogo nekje nad mizo, da uporabnik ve, iz katere smeri bo žoga prispela. Žoga ima določeno začetno hitrost, po nekaj sekundah pa se izstreli iz svojega dela mize proti igralčevemu, ki jo poskuša odbiti. Če mu to uspe, pridobi točko, če pa ne, se mu točke ponastavijo.

Igralec lahko premika lopar z uporabo tipk WASD, horizontalno ga rotira s tipkama E in Q, vertikalno pa s tipkama X in C. Igralec ima tudi možnost ustavitve časa (s tipko T), da se lažje pripravi na odboj.

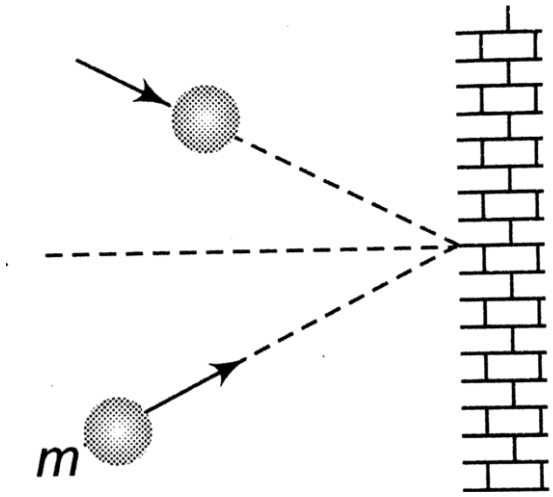
Zvok

Ob vsakem trku žoge se predvaja zvok odboja pingpong žogice. Če igralec uspešno odbije žogo, se zasliši aplavz publike, če pa ne uspe, se predvaja žalosten odziv.



Kolizija

Osnovni problem naše kolizije je določit pravilno smer odboja žoge. Po principu zakona gibanja smo implementirali gibalno količino, zračni upor in izgubo energije ob trku.



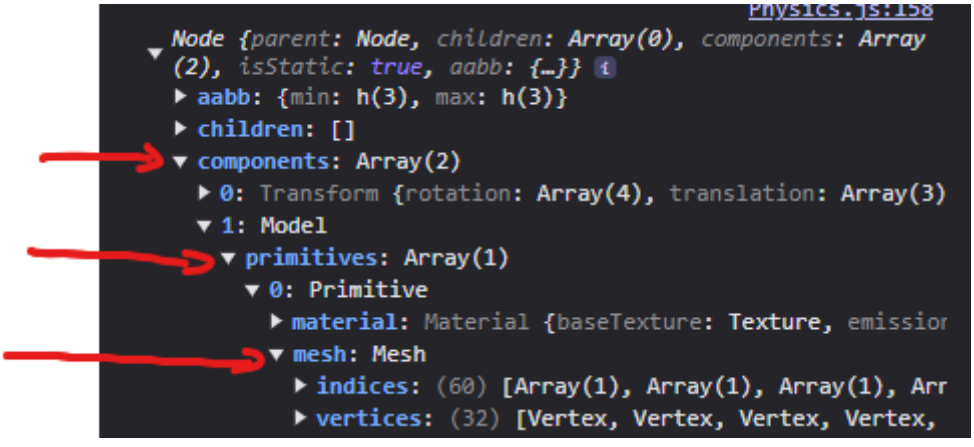
Slika 7: Osnovni problem

Osnovna kolizija

Zavrtimo se čez vse elemente, se za vsak par elementov, če ustrezajo pogojem, preveri če je prišlo do kolizije, in jo obravnavamo.

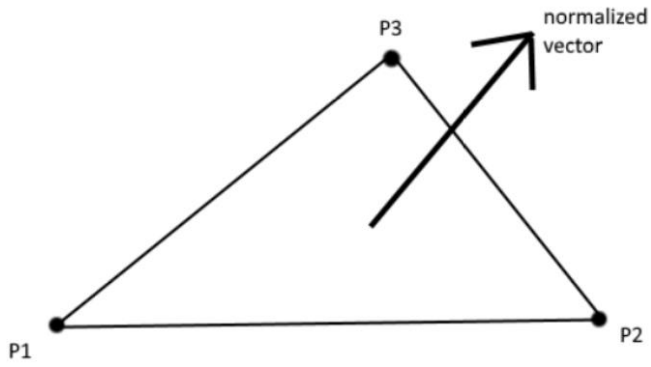
Kako pridobiti pravilno normalo?

1. Najprej pogledamo v kateri element se je zabila žoga.
2. Pridobimo primitive/mesh drugega elementa trka.
3. Ven vzamemo indekse in oglišča.



Slika 8: Objekt v katerega je trčila žoga

4. Da dobimo trikotnike, združimo indekse in oglišča ter zračunamo njihovo normalo. Pri tem upoštevamo pretvorbo iz lokalnega v globalni koordinatni sistem, tako da elemente pomnožimo z globalno matriko.



Slika 9: Trikotnik z normalo

5. Pregledamo vse trikotnike, dokler ne najdemo tistega v katerega smo trčili. Smer žoge smo predstavili z ray direction-om, ki je samo enotski vektor premikanja žoge.

6. Ko smo dobili trikotnik in njegovo normalo, samo še izračunamo skalarni produkt trenutne hitrosti žoge in normiranega normalnega vektorja površine, ki določa, koliko hitrosti je usmerjeno v smeri normale. Nato izračuna zrcaljeno hitrost tako, da od prvotne hitrosti odšteje dvakratnik skalarnega produkta, pomnoženega z normalnim vektorjem. Na koncu posodobimo hitrost žoge z novo zrcaljeno hitrostjo, kar simulira odboj žoge od površine.