Rubikova kocka

Matej Belšak (63210022)

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko

Abstract

Tradiciolna Rubikova kocka je mehanična uganka, ki je sestavljena iz 26 kock. Izumil jo je Ernő Rubik. Vsaka stran kocke ima svojo barvo (bela, rdeča, modra, oranžna, zelena in rumena). Mehanizem kocke omogoča, da se vsaka stran neodvisno obrača, tako da je možno zmešati barve in jo nato rešiti tako, da se vsaka stran vrne v enotno barvo. Za izdelavo Rubikove kocke sem uporabil HTML, CSS, Javascript, WebGL in GLSL.

1 Pregled igre

Zahtevnost Rubikove kocke se lahko razlikuje glede na osebo in njeno izkušnjo z ugankami. Nekateri ljudje jo lahko rešijo zelo hitro, medtem ko drugi potrebujejo več časa in več poskusov. Obstajajo tudi različne stopnje težavnosti, ki se lahko uporabijo pri reševanju Rubikove kocke, tako da lahko vsakdo izbere stopnjo, ki ustreza njegovim sposobnostim.
Namenjena je vsem, ki uživajo v izzivih, želijo izboljšati svoje sposobnosti reševanja ugank in je primerna za ljudi vseh starosti.
Scenarij Rubikove kocke je preprost: najprej se kocke premešajo, nato pa se poskušajo obrniti tako, da se vsaka stran obrne v enotno barvo. To se lahko naredi s tehnikami in strategijami, ki se jih najde na spletu, knjigah ali pa se jih izmisli sam igralec. Ko se kocke obrnejo v enako barvo na vsaki strani je Rubikova kocka rešena.

Izraz kocka se navezuje na Rubikova kocka.
Izraz kockica se navezuje na sestavni del Rubikove kocke.

1.1 Opis sveta

Svet je stiliziran, saj so kocka in njeni obrati zelo poenostavljeni. Kockice se obračajo in premikajo v treh dimenzijah.

1.1.1 Pregled

Uporabnik se po svetu ne more premikati, obrača celotno kocko s pomočjo kazalca miške in ima možnost vklopiti pomešanje barv kocke.

1.1.2 Ozadje

Odločil sem se za sivo ozadje, ker je nevtralno in ne privlači preveč pozornosti (pozornost želiva preusmeriti na kocko). Prav tako se siva barva dobro ujema z večino barvnih shem in se ne meša s svetlimi ali temnimi deli prikaza. Siva barva je tudi enostavna za uporabo, saj ne zahteva veliko spremembe osvetlitve ali kontrasta.

1.1.3 Objekti

Kocka je sama predstavljena kot objekt, ki ima tridimenzionalen seznam objektov pobarvanih kockic, katere se obračajo. Konstruktor kocke kot parametre dobi n (to določa kocko $n \times n \times n$), renderer, očeta (hierarhija), model, teksture, platno in kamero. S pomočjo teh objekt kocke poskrbi za postavitev, teksturiranje, premik in obračanje kockic. Nastavi tudi dogodke miške (angl. mouse events). Razrede, ki določajo objekt kocke in njihove atribute sem izdelal sam.

1.1.4 Čas

Obračanje kockic oz. strani kocke se dogaja s pomočjo časa, kjer ima uporabnik možnost spreminjati hitrost obračanja strani. Metoda za posodabljanje kocke je v neskončni zanki in uporablja produkt spremembe časa in hitrosti za animacijo obrata strani.

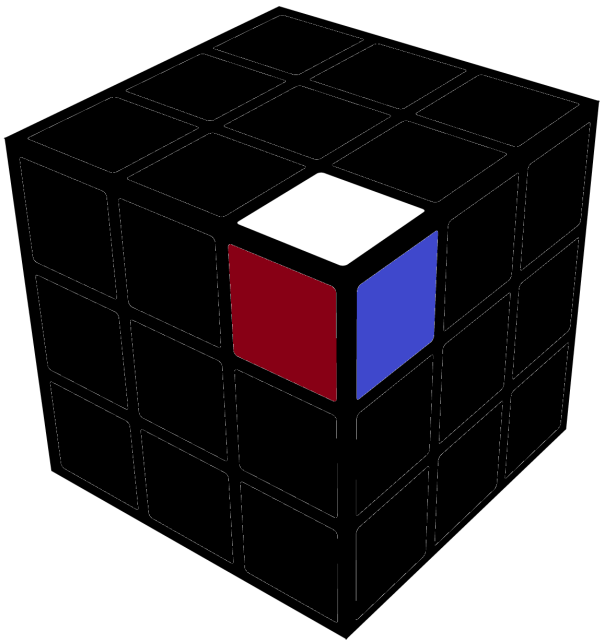
1.2 Igralni pogon in uporabljene tehnologije

HTML in CSS se uporablja za prikaz začetne strani ter prikaz platna. S pomočjo Javascripta, WebGL in GLSL se izriše slika na platno. Za izrisom je logika same kocke, to je izdelano s programskim jezikom Javascript. Za animacije premikov strani, postavitev in ostale izračune se uporablja knjižnica gl-matrix-module.js. S pomočjo te se računa matrike, vektorje in kvaternione. V GLSL je napisan senčilnik, ki s pomočjo vhodnih spremenljivk in uniform v senčilu oglišč (angl. vertex shader) spremeni položaj oglišč z uporabo modelne, pogledne in projekcijske matrike in izračuna položaj ter normale. Nato izhodne spremenljivke prenese v senčilnik fragmentov (angl. fragment shader), ki s pomočjo phongovega in lambertovega zakona izračuna končno barvo pikslov.

2 Kocka

2.1 Teksturiranje in postavitev

Objekt kocka ima seznam kockic. Ta seznam napolni konstruktor kocke. Med polnjenom seznama se izrisujejo kockice z majhnim odmikom ena od druge. Vsaka pridobi svojo teksturo, ki se določi glede na pozicijo. Primer za $x = 0, y = 0, z = 0$:



Slika 1: Teksturirana le ena kockica.

Pozicije kockic omogočajo sledenje le-teh in se razlikujejo od dejanskih koordinat. Koordinate se morajo izračunati tako, da bo sredina celotne kocke v $[0, 0, 0]$. Te se za $n \times n \times n$ izračunajo tako:

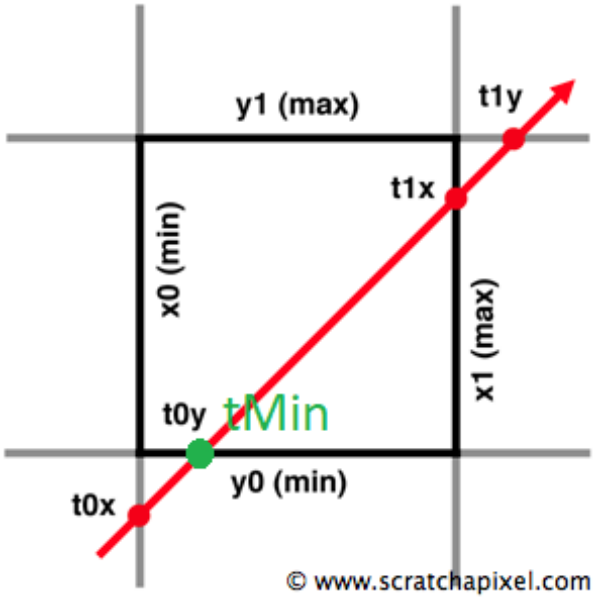
$$f(k) = \begin{cases} (2+\sigma)(\lfloor \frac{n}{2} \rfloor - k), & n \text{ je liho} \\ (2+\sigma)(\lfloor \frac{n}{2} \rfloor - k) - \frac{2+\sigma}{2}, & n \text{ je sodo,} \end{cases}$$

kjer je σ odmik med dvema kockicama in k koordinata. Koordinate se izračunajo tako: $[f(x), f(y), f(z)]$, kjer sta σ in n že globalno določena.

2.2 Izbira kockice

Po uspešno ustvarjeni kocki, se omogočijo obrati strani. To se doseže s klikom kazalca na kockico z metodo raycasting[1] z AABB[2] trki. Ob kliku se odda žarek, ki ima začetne koordinate (koordinate kamere) ter smer. Dolžina smeri žarka določi, kako daleč potuje. Po oddanem žarku se pregleda, katere kockice seka z metodo AABB trki in si zapomni tisto, ki je najbližje kameri.

Algoritem raycasting sem prilagodil, da glede na tMin vrne os strani, katero smo kliknili.



Slika 2: Izbira y strani glede na tMin.

2.3 Obrat strani

Strani se lahko obračajo po oseh x, y in z. Pri obratih sem za animacijo rotacije posamezne kockice uporabljal kvaternione in prišel do zaklepa kardanskega sistema (angl. Gimbal lock). Iz spleta sem probala več različnih rešitev. Ena od teh je tudi rešitev z matrikami.
Rešitev[3] zaklepa, ki je delovala je naslednja:

1. Ob začetku animacije si zapomni stari kvaternion
2. Konstruiranje novega kvaternioniona
3. Rotacija novega kvaterniona za željen kot na željeni osi
4. Rotaciji nastavi produkt starega kvaterniona in novega kvaterniona za določen čas

Stran kocke se obrača po času, kjer je $t = 0$ začetek animacije in $t = 1$ konec animacije.

2.4 Pogled

Uporabil sem perspektivno kamero. Uporabnik se ne more premikati po prostoru in pred sabo vidi le kocko. Uporabnik s pomočjo miškinega kazalca s klikom na ozadje obrača kamero okoli kocke, ki ustvari iluzijo obračanja kocke. Razdalja med kamero in kocko je nastavljena tako, da uporabnik vidi kocko v celoti in nima nadzora nad približevanjem ali oddaljevanjem od kocke. Ker se v svetu kamera obrača okoli kocke in se s tem tudi posodabljajo njene koordinate, se sproti posodabljajo tudi koordinate svetlobe, zato da je vedno osvetljena iz strani kamere.

[1] https://www.youtube.com/watch?v=fQOeEA_8-uk

[2] https://www.scratchapixel.com/lessons/3d-basic-rendering/minimal-ray-tracer-rendering-simple-shapes/ray-box-intersection.html

[3] https://finddiffer.com/why-is-gimbal-lock-a-problem/

3 Osebek

Uporabnik ima nadzor nad vsemi kockicami. Te nadzoruje s pomočjo miškinih dogodkov. To naredi tako, da izbere stran s klikom na kockico in s premikom miške gor, dol, levo ter desno izbere smer obrata. Po dvigu miške (angl. mouse up) pa se izvede animacija obračanja.

4 Uporabniški vmesnik

Začetna stran vsebuje izbiro velikosti kocke. Po izbrani velikosti pritisne na gumb "Play!", ki ga preusmeri na stran igre, kjer pa se za uporabniški vmesnik uporablja GUI iz knjižnice dat.gui.module.js. GUI zagotavlja kontrole za:

- vklop/izklop zvoka
- vklop/izklop premeševanja
- drsnik za spreminjanje hitrosti obračanja strani

5 Glasba in zvok

Zvok sem dodal, zaradi boljše uporabniške izkušnje. Sem mnenja, da je zvok lahko privlačen in spodbudi uporabnika k igranju. Zvok se začne predvajati ob začetku obrata strani do končanega obrata. Zvok se ne prilagaja hitrosti obratom, temveč poda iluzijo hitrih obratov.

6 Gameplay

Na začetku je kocka že sestavljena. Kocko lahko premeša sam uporabnik ali pa vklopi možnost premeševanja. Ko se kocka dobro premeša se lahko loti reševanja. Za reševanje se lahko uporablja več strategij. Uporabnik igro zaključi tako, da kocko v celoti sestavi nazaj v prvotno stanje. Igra nima vgrajenega preverjanja ali je uporabnik sestavil kocko.

7 Zaključki in možne nadgradnje

Naučil sem se delovanje senčilnikov, ugotovil kaj so kvaternioni in zaklep kardanskega sistema ter kako se izbere objekte s pomočjo kazalca. Med izdelavo sem si pogledal tudi OBB (Oriented Bounding Box), ampak se odločil za lažjo implementacijo AABB (Axis-Aligned Bounding Boxes).

Pomankanje je bilo, da nikoli nismo obravnavali, kako delujejo metode raycasting, OBB in AABB. Sem mnenja, da so te pri izdelavi igre pomembne metode saj se uporabljajo pri vseh igrah, kjer se interaktira z miškinim kazalcem (še posebej FPS igre). S predmetom sem na splošno zadovoljen.

Pred začetkom izdelave sem si zastavila cilj premikanja kocke najprej s tipkovnico ali pritiski na gumbe nato pa po možnosti še z miško. Izkazalo se je, da mi je uspelo narediti interakcijo z miško, čeprav sem mislil pustiti brez.

Nadgradnje bi lahko bile:

- OBB metoda za zaznavanje trkov žarka.
- Rotacija kocke in ne kamere okoli kocke.
- Optimizacija kode
- S premikom miške se animira vrtenje strani (zdaj se stran obrne, ko kazalec spustimo)
- Ob izbrani kocki se osvetlijo (angl. bloom) možni obrati strani.
- Senca kocke