UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Stickman Fight

Seminarska naloga pri predmetu Računalniška grafika

Matija Ojo

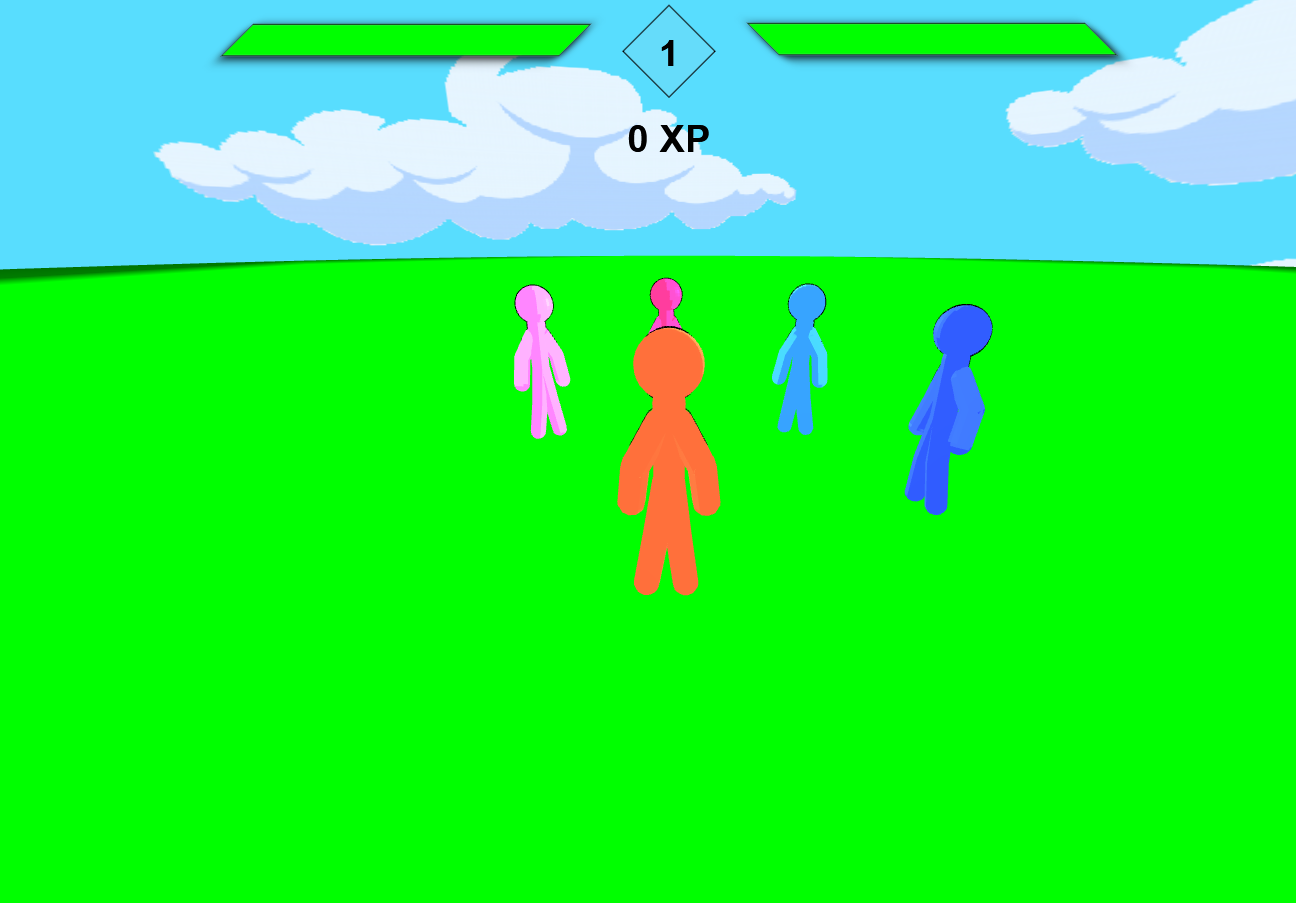
Jan Renar

Ljubljana, januar 2022

# Uvod

V okviru WebGL-a sva naredila igro, v kateri se uporabniki bojujejo s stick figurami.

V igri so implementirane skeletne animacije in skybox – za modela razvita v programu Blender, luči, obroba stick figur ter več igralski način preko omrežja.



# Skeletne animacije

Skeletne animacije temeljijo na »kosteh« in premikanju le teh, kosti namreč določijo kako naj se ustrezne točke 3D modela premikajo, vrtijo ali skalirajo – v nadalje bo za premik, vrtenje in skaliranje uporabljen izraz **transformacija**.

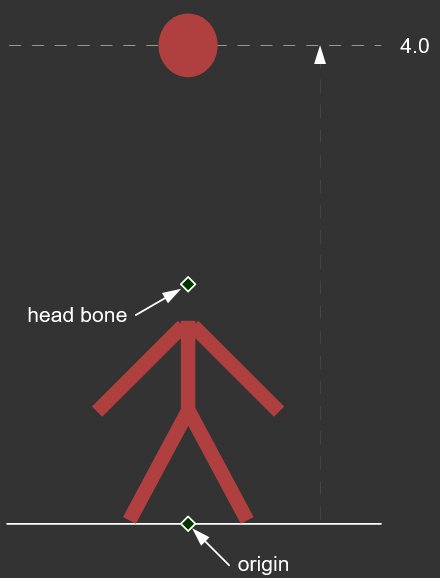
Skeletna animacija je torej le zaporedje transformacij kosti, pri čemer je vsaka označena s keyframe-om. Za koherentno, zvezno animacijo je potrebno med keyframe-i vrednosti ustrezno interpolirati. To je možno storiti ta več načinov, v projektu je uporabljena preprosta linearna interpolacija.

Animacije so shranjene v Javascript objektu, s ključema:

* t: hrani čas določenega keyframe-a
* v: hrani transformacijo določene kosti ob danem času t

## Armatura

Pri skeletnih animacijah je ključnega pomena na kakšen način se shranijo kosti, prebrane iz datoteke formata gltf. Vsaka kost ima poleg imena in transformacije tudi naslednje matrike, ki so praktično obvezne, če želimo upoštevati hierarhijo kosti:

* **Lokalna matrika**: hrani lokalne koordinate kosti in se izračuna med obdelavo animacije (tukaj je shranjena vsaka vrednost kosti med interpoliranjem keyframe-ov, kar je razloženo na prejšnji strani)
* **Svetovna** **(globalna) matrika**: je ključnega pomena za simuliranje hierarhije, izračuna se namreč tako da se starševa svetovna matrika množi s trenutno lokalno matriko. V primeru ko starš ne obstaja (korenska kost) je ta matrika enaka lokalni matriki. Na ta način vsak premik, rotacija, skalacija kosti vpliva tudi na vse otroke te kosti. Gre za zelo eleganten način doseganja hierarhije, saj lahko premik, rotacijo, skalacijo predstavimo z relativno preprosto matriko, za doseganje hierarhije starš-otrok pa matrike preprosto množimo.
* **Inverzno vezavna matrika** (»Inverse bind pose«): Ker imajo tako točke modela kot kosti modela svoje pozicije v prostoru, nastane problem ko želimo točko modela spremeniti (jo množiti) glede na to kje se nahaja kost. Množijo se namreč pozicija točke s pozicijo kosti in ker gre za popolnoma ločene koordinate v prostoru, bi dobili točko, katere pozicija je enaka produktu teh točk. Rezultat bi bil nekaj takega: 

Iz tega razloga je potrebno pozicijo kosti nekako “odšteti”, kar je storjeno tako da se svetovna matrika posamezne kosti (globalna pozicija v prostoru) invertira, ter rezultat shrani v novo matriko.

Ko je ta struktura zagotovljena se v vsakem frame-u določi transformacija vsake kosti glede na trenutno vrednost znotraj interpolacije, svetovna matrika se množi z inverzno vezavno matriko, nakar se rezultat shrani v tabelo, ki se nato shrani kot uniforma znotraj vertex senčilnika.

Senčilnik nato upošteva **uteži** kosti in preko **indeksov** (atributa senčilnika) množi 2 zaporedni kosti, kar pomeni, da na določen vertex vplivata kvečjemu 2 kosti, rezultat vseh množenj pa shrani v matriko, ki se množi s pozicijo veretex-ov.

Na tej točki bi rad izpostavil blog, ki skeletne animacije zelo dobro razloži in je k razvoju le teh neizmerno pripomogel: <https://veeenu.github.io/blog/implementing-skeletal-animation/>

Stick figura ima naslednje animacije: Idle, Run, Tired, Dies, Hit\_Center, Hit\_R, Hit\_L, Punch\_R, Punch\_L, Kick\_R, Kick\_L, Kick\_Spin, Kick\_Jump, Punch\_Uppercut.

Spremenljivka currAnimation znotraj razreda Player določa katera animacija se trenutno izvaja. To omogoča, da preko reference na instancno igralca določamo tudi animacije ob raznih dogodkih kot so kombo napad, figura je pod napadom, figura umre…

# Kamera in kontrole

Tretjeosebno kamero lahko uporabnik premika z miško, z kolescem pa spreminja oddaljenost kamere od igralca (spremenljivka viewDistance). Kamera je implementirana tako, da se v vsakem klicu funkcije »render« pozicija kamere nastavi na pozicijo igralca, nato pa se translira po Z osi za viewDistance enot. Nazadnje se s funkcijo lookAt iz webgl matrix module-a izračuna matrika pogleda, tako da je kamera vedno usmerjena proti igralcu. Izboljšava bi lahko bila ta, da kamera gleda v isto smer kot igralec, torej dodal bi se vektor določene dolžine, ki kaže v smeri igralca, in bi se upošteval pri lookAt matriki. Na podoben načni namreč delujejo ostale tretjeosebne igre.

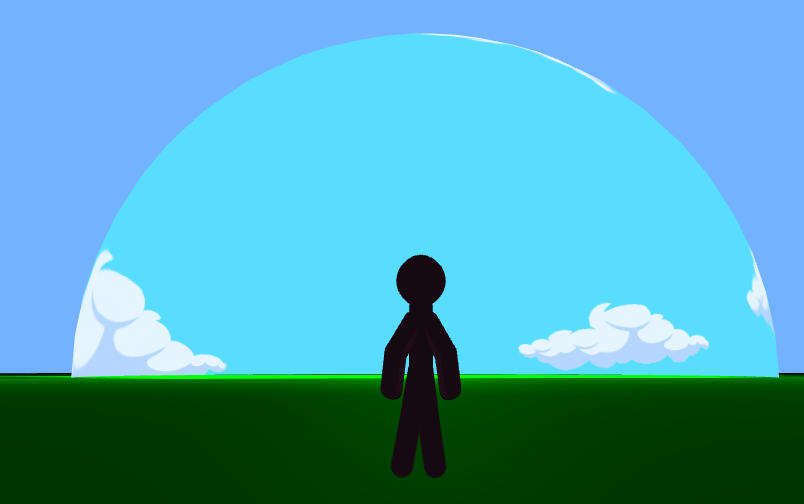
Uporabnik stick figuro premika s tipkami WASD, napade pa izvaja s puščicami (arrow keys). Ob izvedbi več ustreznih napadov, se sproži kombo napad. Kombo napadi preprečujejo to, da bi uporabniki napadali z le eno tipko, kar bi bilo zelo nezanimivo. Dodajo pa tudi dodaten cilj oziroma nivo težavnosti, kar naredi izkušnjo veliko bolj zabavno.

# Skyball

Sprva sem v programu Blender ustvaril kocko, jo »odvil«, ji nastavil teksturo ter invertiral normale. Ker sem uporabljal teksturo, ki ni bila oblike odvitega kvadrata, so bili robovi preveč opazni (slika), zato sem se odločil da bom raje ta isti postopek naredil s kroglo.



Tako sem naredil skyball (vendar mi te seveda ni bilo treba »odviti«).



# Osvetlitev in obroba

TODO: