Implementacija miješanja animacija (engl. *animation blending*)

Adam Ergotić
Mentor: Željka Mihajlović
Fakultet elektrotehnike i računarstva, Zagreb, Hrvatska
adam.ergotic@outlook.com

Sažetak **– U radu je opisana i prikazana metoda miješanja animacija (engl. *animation blending*). Objašnjen je postupak te implementacija navedene metode korištenjem programskog sučelja OpenGL. Također, opisani su međukoraci koji dovode do krajnjeg rješenja rada, a to je miješanje, odnosno kombinacija dviju animacija. Uvođenjem parametra/faktora miješanja, omogućeno je odrediti intenzitet jedne odnosno druge animacija tijekom kombinacije animacija. Opisani su napravljeni dodaci u programu koji omogućavaju detaljniji pregled animacija te utjecaja zglobova (engl. *joint*) koji transformiraju model tijekom animacija.**

Ključne riječi – animacije; skeletalna animacija; miješanje animacija (engl. animation blending); implementacija; OpenGL

I. UVOD

Animacija je vrlo važna tehnika koja omogućuje prikazivanje pokreta različitih tijela i živih bića. Jedno od složenijih zahtjeva animacija je napraviti animacije koje izgledaju prirodno i realistično, pogotovo kod zahtjevnijih modela (poput čovjeka).

Postoje različite metode koje služe za poboljšanje realističnosti i glatkosti animacija, a među njima je metoda miješanja animacija (engl. *animation blending*). Ova tehnika omogućuje kombinaciju više animacija odjednom unutar određenog vremenskom intervala.

Osim opisa navedene metode, u ovom radu se također prikazuje kôd implementacija metode korištenjem tehnologije OpenGL i ostalih.

II. TEHNOLOGIJE

A. *OpenGL*

OpenGL je programsko sučelje koje služi za prikazivanje 2D odnosno 3D grafike. Unutar ovog rada, OpenGL koristi se za prikazivanje modela i animacija modela. Za pisanje sjenčara (engl. *shader*) koristi se GLSL.

B. GLFW

GLFW jednostavna je biblioteka (engl. *library*) koja služi kao apstraktni sloj koji omogućuje poziv funkcija OpenGL API-ja iz programa. Također, biblioteka je napravljena tako da radi na više platformi (engl. *cross-platform*).

C. *Assimp*

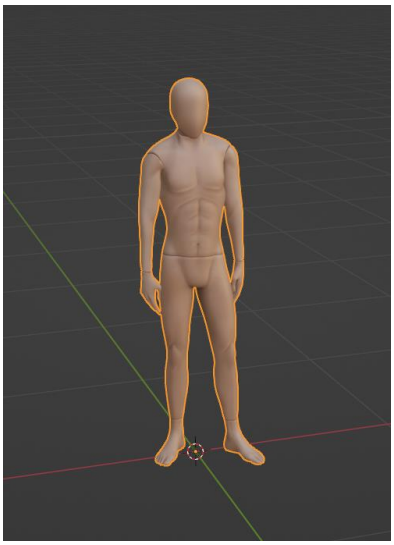
Assimp je C++ biblioteka koja služi za uvoz modela. Unutar ovog rada, Assimp se koristi za uvoz modela i animacija te svih podataka koji su potrebni za implementaciju miješanja animacija.

D. *FBX*

Za zapis modela koristi se .fbx format. FBX format omogućuje zapis modela zajedno s animacijama.

E. *Blender*

Blender se koristi za spajanje svih animacija i modela u jednu datoteku. Korišten model (Slika 1) unutar rada je s web stranice Mixamo[1].



Slika 1: Mixamo model

III. IMPLEMENTACIJA

U nastavku se nalazi postupak i opis postupka kojim se postiže metoda miješanja animacija.

A. *Opis*

Kao prvi korak, potrebno je preuzeti i konfigurirati sve potrebne biblioteke: GLFW, Assimp te GLM (biblioteka za izvođenje matematičkih operacija za grafičke potrebe).

Nakon inicijalnog koraka, prvo je potrebno učitati model s pomoću Assimp biblioteke. Objekt koji se dobije učitavanjem sadrži sve potrebne informacije za

implementaciju animacija. Nakon uvoza modela, potrebno je napisati kôd za izvođenje animacija. Implementacija animacija je obrađena u prethodnom seminaru[2]. Nakon implementacija animacije, potrebno je implementirati miješanje animacija što se opisuje u nastavku rada.

B. Implementacija

Kod uobičajenog izvođenja animacija koristi se samo jedna animacija te se između svakog ključnog okvira (engl. *keyframe*) događa interpolacija između transformacije[1] početnog ključnog okvira i transformacije krajnjeg ključnog okvira. Za miješanje animacija, osim interpolacije između početnog i krajnjeg ključnog okvira, potrebno je omogućiti kombinacija više animacije (u ovom radu omogućeno je kombinacija dvije animacije).

Za početak, potrebno je definirati faktor miješanja (engl. *blend factor*). Faktorom miješanja definiramo koliko će jedna animacija prevladavati nad drugom. Na primjer, ako je faktor jednak nuli, onda će potpuno prevladavati prva animacija. Kad je faktor jednak 0.5, dobivamo miješanje dviju animacija u kojoj obje doprinose jednako (50 % prva i 50 % druga animacija) krajnjoj animaciji koju vidimo na zaslonu. Izračun krajnje animacije možete se prikazati sljedećom formulom:

$$Finalna\ animacija = (1 - \alpha) * A_1 + \alpha * A_2$$
$$(linearna\ interpolacija)$$
$$gdje\ su\ A_1\ i\ A_2\ dvije\ različite\ animacije$$
$$\alpha\ je\ faktor\ miješanja$$

Prilikom uvoza modela pomoću biblioteke Assimp, objekt sadrži informacije o zglobovima te za svaki zglob (engl. *joint*) mapira indekse vrhova modela na koje taj zglob utječe. Naime, budući da je sjenčaru vrhova potrebno proslijediti informacije za svaki vrh zasebno, potrebno je napraviti obrnuto mapiranje (za svaki vrh odrediti indekse zglobova i koliko utječu na taj vrh, odnosno težine). Potom se ti indeksi i težine zglobova šalju u sjenčar vrhova gdje se koristi za izračun nove pozicije vrha. Izračun obrnutog mapiranja potrebno je napraviti samo jednom jer se mapiranje ne mijenja tijekom izvođenja animacija.

Nakon toga, jedino što je preostalo je izračunati krajnje transformacije za svaki zglob. To se radi tako da za svaki zglob odredimo matricu za translaciju, rotaciju i skaliranje. Za to je potrebno linearno interpolirati između dvije transformacije koristeći faktor miješanja. Nakon toga sve matrice transformacija pomnožimo čime dobijemo transformacijsku matricu svakog zgloba. Izračun transformacijskih matrica je potrebno napraviti nakon svakog iscrtavanja budući da se mijenjaju ovisno o tome koliko je vremena prošlo između iscrtavanja. Cijeli postupak je prikazan u nastavku:

[1] Odnosi se na transformacijske matrice

```
// Interpolate scaling
const aiVector3D& Scale0 = StartTransform.mScaling;
const aiVector3D& Scale1 = EndTransform.mScaling;
aiVector3D BlendedScaling = (1.0f - blendFactor) * Scale0 + Scale1 * blendFactor;
aiMatrix4x4 ScalingM;
aiMatrix4x4::Scaling(BlendedScaling, ScalingM);

// Interpolate rotation
const aiQuaternion& Rot0 = StartTransform.mRotation;
const aiQuaternion& Rot1 = EndTransform.mRotation;
aiQuaternion BlendedRot;
aiQuaternion::Interpolate(BlendedRot, Rot0, Rot1, blendFactor);
aiMatrix4x4 RotationM = aiMatrix4x4(BlendedRot.GetMatrix());

// Interpolate translation
const aiVector3D& Pos0 = StartTransform.mTranslation;
const aiVector3D& Pos1 = EndTransform.mTranslation;
aiVector3D BlendedTranslation = (1.0f - blendFactor) * Pos0 + Pos1 * blendFactor;
aiMatrix4x4 TranslationM;
aiMatrix4x4::Translation(BlendedTranslation, TranslationM);

// Combine it all
NodeTransformation = TranslationM * RotationM * ScalingM;
```

Slika 2: Interpolacija skaliranja, rotacije i translacije

Nakon izračuna transformacijski matrica za svaki zglob, iste matrice se prosljeđuju sjenčaru u obliku *uniform* varijabli. Jedino što je preostalo je izračun krajnje transformacije vrha koja se potom primjenjuje na sami vrh (Slika 3).

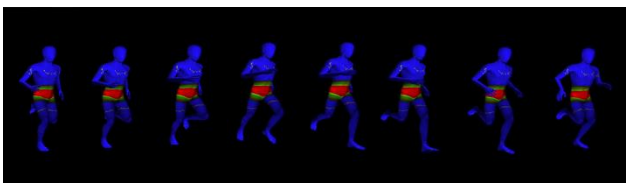
```
mat4 boneTransform = mat4(0.0f);

for (int j = 0; j < MAX_NUM_OF_BONES_PER_VERTEX; j++)
{
    if(j < 4)
    {
        boneTransform += uBones[boneIDs_1[j]] * boneWeights_1[j];
    }
    else
    {
        boneTransform += uBones[boneIDs_2[j-4]] * boneWeights_2[j-4];
    }
}

vFragPos = vec3(model * vec4(position, 1.0));
gl_Position = projection * view * model * boneTransform * vec4(position, 1.0f);
```

Slika 3: Primjena zglobova unutar sjenčara vrhova

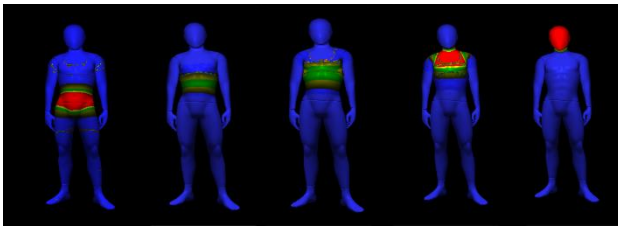
Pokretanjem programa dobije se sljedeća animacija:



Slika 4: Miješanje animacije trčanja i hodanja

U programu je također omogućeno nekoliko opcija za korisnika. S tipkom B moguće je povećati odnosno smanjiti faktor miješanja što se odmah može primijetiti na samoj animaciji na zaslonu. Tipkom N je moguće mijenjati animacije koje su uvezene zajedno s modelom. Tipkom R se rotira model.

Dodatno, omogućen je i prikaz težina svakog zgloba. U početku se prikazuju težine za nulti zglob, a pritiskom tipke *Spacebar* moguće je vidjeti i utjecaj ostalih zglobova na pojedine dijelove modela. Težine različitih zglobova moguće je vidjeti u nastavku:



Slika 5: Prikaz utjecaja različitih zglobova modela

Poveznica na videozapis u kojemu se prikazuju funkcionalnosti aplikacije: https://www.youtube.com/watch?v=oCOYA4nW01s

IV. ZAKLJUČAK

Miješanje animacija je jedna od mnogih tehnika koje omogućuju poboljšanje realističnosti animacija. Pomoću nekoliko parametara i animacija nam omogućuje stvaranje vrlo prirodne tranzicije između dvije animacije.

Osim toga, metoda je vrlo fleksibilna. Umjesto dvije animacije, moguće je napraviti miješanje tri ili više animacija ako je potrebno. Također, umjesto linearne interpolacije, mogu se koristi i druge vrste interpolacije kako bi se dobio drugačiji efekt miješanja animacija.

LITERATURA

[1] Adobe, “Mixamo”, mixamo.com

[2] A. Ergotić, “Implementacija animacija pomoću Vulkan API”, drive.google.com/file/d/1FZekVlj4ppCPoI8bQ3bXlN6yV0I4u34o/view?usp=sharing

[3] E. Meiri, “ogldev”, github.com/emeiri/ogldev

[4] freeCodeCamp.org, “Advanced OpenGL Tutorial – Skeletal Animations with Assimp”, https://www.youtube.com/watch?v=GZQkwx10p-8

[5] Assimp documentation, https://assimp.sourceforge.net/lib_html/annotated.html

[6] Glm documentation, https://glm.g-truc.net/0.9.9/api/index.html

Poveznica na repozitorij: https://github.com/Wuffll/DiplomskiProjekt