Hangfeldolgozás

A feladat megoldásához az általam jobban ismert C illetve C++ programozási nyelvet választottam LabView helyett. A projektek Microsoft Visual Studio 2010-ben készültek, a grafikus megjelenítéshez Simple Directmedia Library-t használtam (továbbiakban SDL).

A forrást mellékeltem, a lefordított binárisok a `release/` könyvtárban találhatók.

DFT és FFT algoritmusok

Saját adatstruktúrát alakítottam ki a komplex számok ábrázolására, és az ezekkel való műveletek elvégzéséhez is saját függvényeket írtam. Saját FFT algoritmus implementálása helyett, egy kész algoritmust módosítottam úgy, hogy a saját adatstruktúrámmal kompatibilis legyen. Azért választottam egy kész algoritmus felhasználását, mert a saját implementációm több szempontból is használhatatlannak bizonyult, és pontatlan is volt. A választásom így a cFFTv2-re esett.

Az algoritmus csak $N = 2^n, n \in Z$ darab minta feldolgozására alkalmas.

$N = 20$	$N' = 32$
$N = 30$	$N' = 32$
$N = 40$	$N' = 64$
$N = 200$	$N' = 255$

A feldolgozandó minták tehát a mellékelt táblázat szerint alakulnak.

A Diszkrét Fourier Transzformáció $D[k] = \frac{1}{N}\sum_{n=0}^{N-1} x[n] \cdot e^{-j \cdot k \cdot \Theta_0 \cdot n}$ - ahol $\Theta = \frac{2\pi}{N}$ - szerint alakul. Azonban ez felbontható páros és páratlan szekvenciák összegére:

$$D[k] = \frac{1}{N}\sum_{n=0}^{\frac{N}{2}-1} x[2n] \cdot e^{-j \cdot k \cdot \Theta_0 \cdot 2n} + \sum_{n=0}^{\frac{N}{2}-1} x[2n+1] \cdot e^{-j \cdot k \cdot \Theta_0 \cdot (2n+1)} =$$

$$= \frac{1}{N}\left(\sum_{n=0}^{\frac{N}{2}-1} x[2n] \cdot e^{-j \cdot k \cdot \frac{4\pi}{N} \cdot n}\right) + \left(\sum_{n=0}^{\frac{N}{2}-1} x[2n+1] \cdot e^{-j \cdot k \cdot \frac{4\pi}{N} \cdot n}\right) \cdot e^{-j \cdot k \cdot \pi} = D_E[k] + D_O[k] \cdot W^n$$

Ez tetszőleges mértékben elvégezhető. Így a DFT-vel szembeni $O(N^2)$ műveletigény $O(N \cdot \log_2 N)$-re csökken.

A CFFTv2 algorutmus két részből áll, egy iteráló, és egy kernel részből. Az iteráló rész adja össze a különböző mélységű páros és páratlan szeleteket a kernel segítségével.

A kernel algoritmusa a köveketőképpen alakul:

```
step(
      N     minták száma
      M     páros-páratlan mélység (n*log2(N))
      a[]   bal oldali bemenő buffer
      b[]   jobb oldali bemenő buffer
      c[]   bal oldali kimenő buffer) – következő bemenet eredménye
      d[]   bal oldali kimenő buffer  – következő bemenet eredménye
```

```
      sgn   előjel (inverz fourier transzformációhoz)
){
```

k megy 0-tól $\frac{2N}{M}$ -ig:

i megy 0-tól M -ig:

$$c\left[k\cdot\frac{1}{2}M+i\right]=a[k\cdot M+i]+b[k\cdot M+i]$$

$$d\left[k\cdot\frac{1}{2}M+i\right]=\left(a[k\cdot M+i]-b[k\cdot M+i]\right)\cdot e^{j\frac{2\pi}{N}\cdot 2M\cdot k}\cdot\left(j\cdot\mathrm{sgn}\right)$$

```
}
```

Az ezt meghívó itáráló függvény pedig meghívja ezt minden n*log2(N) lépsre.

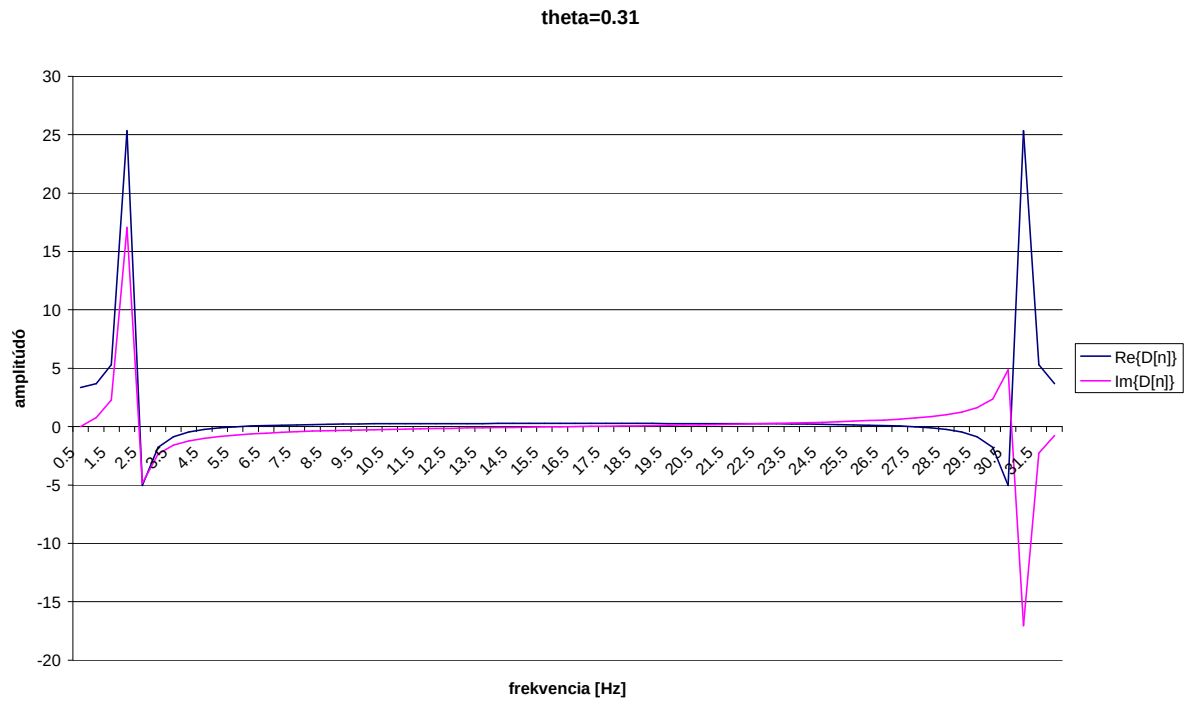
$M=1$

m megy 0-tól $\log_2 N$ -ig

$M=2M$

```
step(N, M, &y[0], &y[n/2], &x[0], &x[M], sgn)
cserél(x[],y[])
```

Az algoritmus nem tartalmazza az $\frac{1}{N}$ való szorzást. Bár az algoritmus bőséges mértékben optimalizálható párhuzamosítható lenne, de ezt nem tettem meg.



Wav file spektrumának ábrázolása idő függvényében

Ez egy olyan program, mely képes egy tetszőleges wav formátumú filet betölteni, lejátszani, illetve megjeleníteni a spektrumát.

A feldolgozás alatt álló buffer tartalmát továbbítja a hangkártya felé. Ennek egyik hibája, hogy Windows operációs rendszeren a CR/LF sortörés miatt nem működik megfelelően.

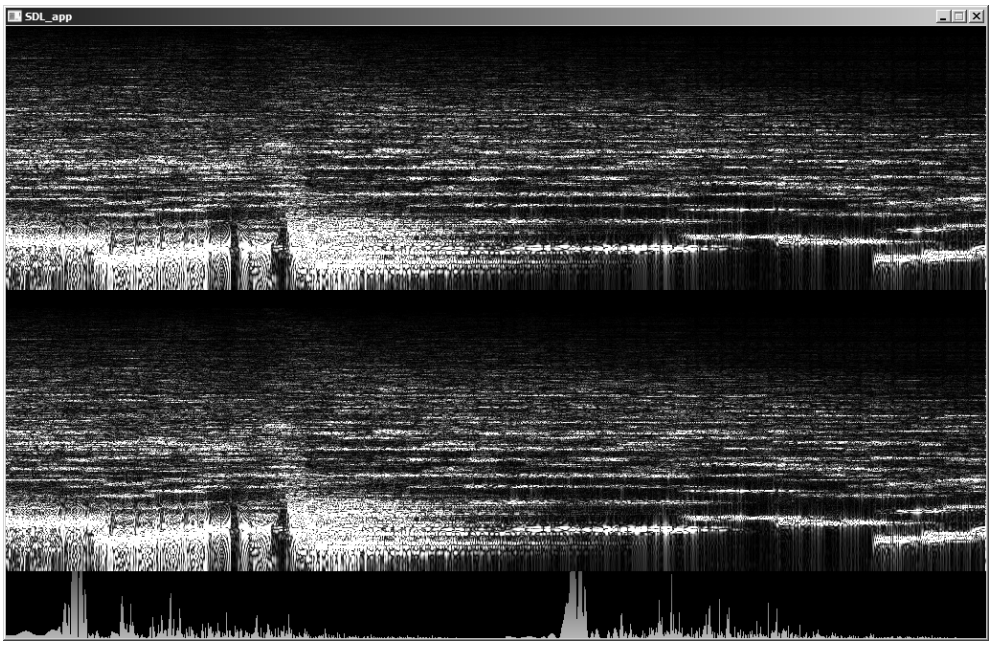
A spektrumot minden csatornára kirajzolja az ablak aljára, illetve fölé spektogrammot az idő függvényében. Mivel frame-alapú, ezért a spektogram vízszintes tengelye a kirajzolt frameket, és nem az időt tartalmazza, bár minden framet általában egységnyi idő alatt számolja ki (AMD Phenom II X6 processzor egyetlen magján ~15 ms 16384 mintán)

A fourier transzformáció szimmetriája miatt a felhasznált mintáknak csak fele jeleníthető meg, a másik fele ennek a konjugáltja. Ezt 20 Hz és 20 kHz közötti tartományon logaritmikusan ábrázoltam. Az interpoláció hiába miatt Moiré-minták jelennek meg, melyeket nem tudtam szűrni megfelelően.

Használat

A program első paraméterében várja a feldolgozandó file nevét/elérési útvonalát. Amennyiben nincs ilyen, akkor standard inputról vár adatot. A program a lejátszás végeztével automatikusan kilép, illetve ESC billentyűvel vagy bezárás gombbal megszakítható.

```
>wavViz.exe <filenév>
```



1. ábra Da Tweekaz - Healing Incentation

Audio Processing Library

A továbbiakban minden olyan osztályt illetve eszközt, amire több projektnek is szüksége van egyetlen közös könyvtárba implementáltam, melyet minden projekt el tud érni.

Ebben foglal helyet az FFT algoritmus, a szűrők, illetve a wav file írásához/olvasásához szükséges függvények, osztályok.

A wav olvasása egy meghatározott méretű bufferbe történik, melynek kiürülése esetén betölti a következőt, egésze addig ameddig el nem fogy. Ennek egy ismert hibája, hogy az utolsó szeletet nem dolgozza fel, tervezési hiba miatt.

Moving Average szűrő készítése

Ez az alkalmazás egy konvolúciós szűrőt alkalmaz egy bemeneti filera. A konvolúciót belépő jelre alkalmazza; mind a súlyfüggvény (impulzusválasz) mind pedig a gerjesztés belépő. Minden kezdeti feltétel nulla.

A szűrés $y[n]=\sum_{k=0}^{M-1} h[k]\cdot x[n-k]=h*x$ alakul. $h[n]=\frac{1}{M}$ ahol M az átlagoló szűrő súlyfüggvényének hossza.

Erre létrehoztam egy szűrőláncot, mely áll egy generátorból, és tetszőleges számú szűrőből. Ez minden bufferen végigfut csatornánkként, majd minden definiált szűrő szuperpozícióját kiírja fileba. Jelen esetben csak egyetlen darab konvolúciós szűrő van a láncban.

Megj.: nagyjából M=20-30 hosszig működik jól.

Használat

A program egy konzolos alkalmazás; paraméterezése a következő:

```
>MA_filter.exe <M> <bemenet> <kimenet>
```

Ahol `<M>` a futó átlagoló hossza, `<bemenet>` a bemenő, illetve `<kimenet>` a kimenő wav file. Minden paraméter megadása kötelező. Amennyiben valamelyik file neve helyett kötőjel karakter szerepel, akkor azt a stadard inputról várja, vagy standard outputra adja vissza.

Sinc szűrő

Az előzőhöz hasonlóan ideális alul áteresztő szűrő megvalósítása konvolúcióval. Az előzőehkez hasonlóan.

A súlyfüggvény $h[n]=2\cdot\sigma\cdot\frac{\sin(2\pi\sigma t)}{2\pi\sigma}=2\cdot\sigma\sin c(2\pi\sigma t)=\frac{\sin(2\pi\sigma t)}{\pi t}$ melyet eltoltam $\frac{1}{2}M$-mel: $h[n]=\left.\frac{\sin(2\pi\sigma t)}{\pi t}\right|_{t=n-\frac{1}{2}M}$

Használat

A program egy konzolos alkalmazás; paraméterezése a következő:

```
>MA_filter.exe <sigma> <M> <bemenet> <kimenet>
```

Ahol `<sigma>` a sinc szűrő határfrekvenciája Hz-ben, `<M>` a sinc szűrő hossza hossza, `<bemenet>` a bemenő, illetve `<kimenet>` a kimenő wav file. Minden paraméter megadása kötelező. Amennyiben valamelyik file neve helyett kötőjel karakter szerepel, akkor azt a stadard inputról várja, vagy standard outputra adja vissza.

Tetszőleges szűrő

A program előállít egy fűrészjelet, majd egy rezonáns alul áteresztő szűrőt alkalmaz rá. Ezzel megközelítve az elektronikus zenében legendássá vált Roland TB-303 szekvencert/szintetizátort.

A szűrést időfüggvényben végzi el.

$$y_A[n] = y_A[n-1] + f \cdot (x[n] - y_A[n-1] + fb \cdot (y_A[n-1] - y_B[n-1]))$$
$$y_B[n] = y_B[n-1] + f \cdot (y_A[n-1] - y_B[n-1])$$

Ahol $f = \frac{f_{cutoff}}{\frac{1}{2} f_{sampling}}$, $f \in R \cap \{0..1\}$ a határfrekvencia $q \in R \cap \{0..1\}$ a rezonancia mértéke, illetve $fb = q \cdot \left(1 + \frac{1}{1-f}\right)$

A program létrehoz egy fűrészjelet, melynek amplitúdóját egy másik fűrésszel modulálja – kezdetleges ADSR (Attack – Decay – Sustain - Release) modulációt szimulálva, illetve automatikusan modulálja f és q értékeket egy LFO (Low Frequency Oscillator) segítségével. Ezzel hasonló hatást kelt egy azonos hangmagasságon működő sequencerrel.

Használat

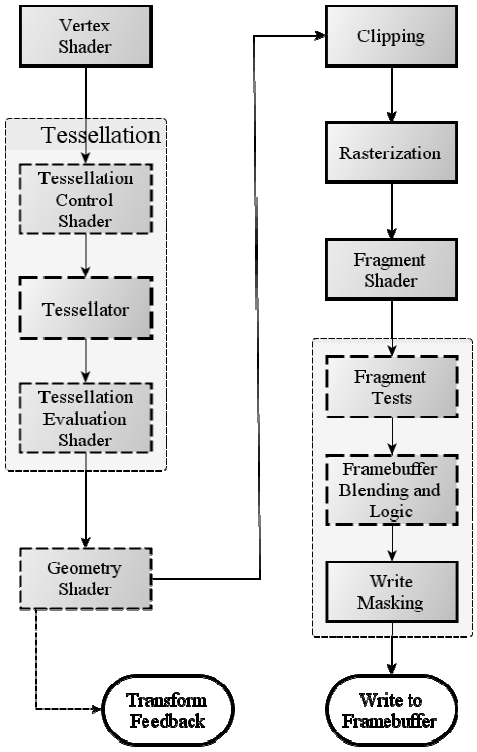
A program egy konzolos alkalmazás, mely létrehozza a munkakönyvtárba a 303.wav filet.

```
>303teszt.exe
```

Képfeldolgozás

Képek feldolgozását WebGL segítségével végeztem el, korábbi tapasztalataim nyomán. A WebGL egy böngészőbe ágyazható, javascriptből programozható 3D grafikus API, mely nagyjából ekvivalens az OpenGL ES-sel.

A megvalósíáshoz WebGL2-t használtam. Ez lehetővé teszi, hogy a redering pipeline vertex, illetve fragment műveleteit saját programkóddal implementáljuk. Ilyenek például a transzformáció, illerve projekció számítása vertexenkként, a háromszögelő számára, illetve a fényforrások szuperponálása raszterelés során. Ez a nyelv a GLSL (OpenGL Shading Language). Bár annak ellenére, hogy rengeteg dolog megvalósítható ezzel, sajnos rengeteg paraméter a renderelés többi szakaszából nem, vagy nehezen befolyásolható. Ennek bemutatása messze túlhaladja ennek a feladatnak a kereteit.



Egy shader program egy vertex, és egy fragment shader párból áll. Mivel 2D képfeldolgozásra volt szükség, ezért a 3D-s tér két olyan polygont tartalmaz, mely lefedi a teljes render felületet. Ennek minden pixelére fut le a fragment shader.

A böngészők rendelkeznek egy ún. **same-orign policyval**, amely egy biztonsági zár, ami azt gátolja meg, hogy bármilyen objektumot javasctiptből csak azonos forrásból (domain, tartomány, port) származó objektumokat lehet elérni. Ez vonatkozik a fileszintű elérésre is. Mivel a képeket más módon nem lehet betölteni, és megkerülni sem lehet, mindenképp szükség van arra, hogy ezek elérhetőek legyenek. Ezt

megkerülni úgy lehet, hogy egy webszerverről kell hosztolni a fileokat. Egy egyszerű webszervert mellékeltem (`inkaServer.exe`) ami a gyökérkönyvtárat hosztolja a localhoston, default 80-as porton.

Minden shader forrását külön mellékeltem.

Fényerő, kontraszt

Ez az alkalmazás egy kiválasztott kép (textúra) fényerejének, kontrasztjának befolyásolására, és gamma-korrekcióra alkalmas.

Ezen paraméterek szerinti képmanipuláció a kép mintáinak egy transzformációs görbe szerinti módosítása szerint történik. GLSL-ben az intenzitásértékek mintánkként nem 0..255 tartományon, hanem fix-pontosan, 0..1 tartományban vannak, illetve ugyanígy kell a háromszögelő felé visszaadni. Ez rengeteg dolgot megkönnyít.

Alkalmazott transzformációs görbék:

$$I_{fényerő} = I_{be} + F, F \in [-1..1]$$

$$I_{kontraszt} = I_{be} \cdot K, K \in [0..1]$$

$$I_{gamma} = I_{be}^{\gamma}$$

Minden transzformációs görbe egymás mögé sorosan van kapcsolva. Ahhoz, hogy megakadályozzam, hogy az intenzitás átforduljon negatívba, a fényerőt és kontrasztot követően elvágtam az intenzitás értéket 0..1 tartományban.

Továbbá hozzáírtam azt, hogy az intenzitás értékeket egy tartományban el lehessen vágni.

Mivel GLSL-ben a mintákhoz tartozó intenzitás értékek három (négy) komponensű vektorként jelennek meg, ezért értelmezhetőek egy vektortérben (színtér – color space), melyekre különböző affin transzformációk elvégezhetők mátrixműveletek segítségével:

$$\bar{y}[n] = \begin{pmatrix} \bar{x}[n].r \\ \bar{x}[n].g \\ \bar{x}[n].b \\ 1 \end{pmatrix} \cdot \begin{pmatrix} c & 0 & 0 & b \\ 0 & c & 0 & b \\ 0 & 0 & c & b \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ahol } \begin{matrix} c = F, F \in [-1..1] \\ b = K, K \in [0..1] \end{matrix}$$

A vektor negyedik komponense az affin transzformációhoz szükséges kiterjesztés, melyet eliminálni kell. (Ha van alpha faktor, akkor azt vissza kell írni.)

Használat

Az alkalmazás egy darab HTML file, illetve az ehhez tartozó JS modulok, és textúrák. Ezeket mindenképp egy web szerveren kell tárolni. Egy ilyent mellékeltem.

```
>http://localhost/01_fenyero_kontraszt.html
```

Az alkalmazás teljesen grafikus.

2. ábra Alkalmazás futás közben

Konvolúciós szűrő

Ez az alkalmazás egy kiválasztott kép (textúra) tetszőleges 3x3-as PSF függvénnyel való konvolúcióját valósítja meg.

OpenGL-lel való képmanipuláció rengeteg dologban eltér a hagyományos értelemben vett képfeldolgozásétól. Számos korlát is jelen van, mely a 3D renderelést kívánja támogatni. Ilyen például, hogy a textúrák oldalaránya 1:1 lehet, és csak kettő valamely hatványa lehet a hosszuk. Erre a textúracímzés egyszerűsítése miatt van szükség. (Bár léteznek olyan bővítmények, amik ezeket feloldják, de nem használják őket) A textúra címezés képernyő koordináta helyett Descartes-koordináta szerint alakul. A textúra felcímzése eszerint 0..1 tartományon lehetséges. Ehhez mérten a konvolúcióhoz ismerni kell a textúra méreteit.

A 2D-s konvolúció $y[u,v] = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} x[u,v] \cdot p[u-i, v-j]$. Mivel GLSL-ből nem tudjuk felcímezni pixelenkként, ezért másmilyen módszerrel kell a választ meghatározni. Ennek értelmében a koordinátákat vektorműveletekkel kell meghatározni:

$\overline{y}\left[\overline{uv}\right] = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} \overline{x}\left[\overline{uv}\right] \cdot p\left[\overline{uv} - \overline{\Delta uv}[m,n]\right]$ ahol $\overline{x}$ és $\overline{y}$ az $\overline{uv}$ mintához tartozó intenzitás értékek vektora (RGB), $\overline{uv}$ az aktuális textúra koordináta $\overline{uv} = (u \quad v)$, p a pont-szétterjedési függvény mátrix, illetve $\overline{\Delta uv}$ a mátrix $[m,n]$ koordinátájához tartozó $\overline{uv}$-tól vett távolság.

Textúrából való mintavételezés egy textúra szűrőn megy keresztül, mely megfelelően a textúra méretétől, a szomszédos pixelek távolságától interpolálja, vagy újra mintavételezi a képet különböző eljárások segítségével. Ennek célja az, hogy egy textúrát egy 3D felületre felhúzva simább és szebb legyen. (aliasing, mip-maping, anisotropic filtering)

Ezért $\overline{\Delta uv}$ mátrix $\overline{\Delta s} = \frac{1}{\overline{ST}}$ egész számú többszöröseit fogja tartalmazni.

Megj.: $\overline{uv}$-val az egységnégyzeten felvett koordinátát, $\overline{ST}$-vel pedig a képernyő koordináta szerintit. Bizonyos irodalmak, szoftverek, API-k ettől eltérnek.

Használat

Az alkalmazás egy darab HTML file, illetve az ehhez tartozó JS modulok, és textúrák. Ezeket mindenképp egy webszerveren kell tárolni. Egy ilyent mellékeltem.

```
>http://localhost/02_konvolucio.html
```

3. ábra Alkalmazás futás közben

Irodalomjegyzék

- Simple Direct Media Library http://www.libsdl.org/
- CFFTv2 ftp://ftp.acer-euro.com/gpl/AS1800/xine-lib/src/libfaad/cfft.c
- Music-DSP source code archive http://www.musicdsp.org/archive.php
- Shadertoy https://www.shadertoy.com/
- Ińigo Quílez (IQ^RGBA) cikkei http://www.iquilezles.org/