3.5. Videotömörítési algoritmusok

Korábban már indokoltam a digitális videoadatok tömörítésének szükségességét, most röviden vizsgáljuk meg azokat a szempontokat, amelyeket figyelembe kell venni a videojel tömörítésénél.

- Real-time - Non-real-time: a real-time tulajdonképpen azt jelenti, hogy a rögzítés-tömörítés (enkódolás; röviden kódolás) és a lejátszás-kitömörítés (dekódolás) valós időben zajlik, tehát szinkronban van, nincs késleltetés. A non-real-time ennek az ellenkezője, tehát a lejátszás nem tudja követni a rögzítést, így esetleg képkockák maradhatnak ki lejátszáskor, a kép (video) és a hang (audio) nincs szinkronban.
- Szimmetrikus - Aszimmetrikus: ez a tulajdonság a kódolás és a dekódolás időarányára vonatkozik. Szimmetrikus a tömörítés, ha a kódoláshoz ugyanannyi időre van szükség, mint a dekódoláshoz. Erre élő közvetítéseknél van szükség, ilyen például az Internetes videokonferencia. Az aszimmetrikusnál általában a kódolás időigénye nagyobb, mint a dekódolásé. Ilyen tömörítést alkalmaznak olyan videók esetében, amelyeket egyszer kell rögzíteni, és sokszor lejátszani, mint például házi mozifilmeket.
- Tömörítési arány: fogalmát az 1.3. fejezetben érintettük.
- Veszteséges - Veszteségmentes: fogalmát szintén az 1.3. fejezetben tárgyaltuk. Veszteségmentes tömörítésnél matematikailag kifejezve: DEC(ENC(x)) = x, ahol DEC a dekódoló függvény, ENC az kódoló függvény, x pedig az a jel amit tömörítünk. Ugyanez veszteséges tömörítésnél: DEC(ENC(x)) ≠ x = y, ahol y a módosult jel. A veszteségi arány a 100*y/x [%] képlettel fejezhető ki. Az FLI formátumnak például van veszteségmentes fajtája is, de az MPEG tömörítés mindig veszteséges.
- Intraframe - Interframe: az intraframe kifejezés azt jelenti, hogy egy tömörített képkocka egy, diszkrét képként van tárolva, függetlenül a többi képkockától. Interframe-ként van tömörítve egy képkocka, ha az függ az őt megelőző és/vagy az őt időben követő képkockák tartalmától. Ezek szerint különbségeket képez, és a változatlan részeket nem tárolja.[11]
- Adatátviteli sebesség (KBps, MBps): Megadja, hogy hány bit adat megy át a csatornán másodpercenként. Ez is összefügg a minőséggel, hiszen nagyobb bitarány jobb minőséget eredményez. A fejlesztők fő célja az volt, hogy olyan tömörítési technikákat dolgozzanak ki, amelyekkel alacsonyabb bitarány mellett is megtartható a jó minőség.

A gyakorlatban különféle videotömörítési eljárások állnak a rendelkezésünkre, ilyenek például:

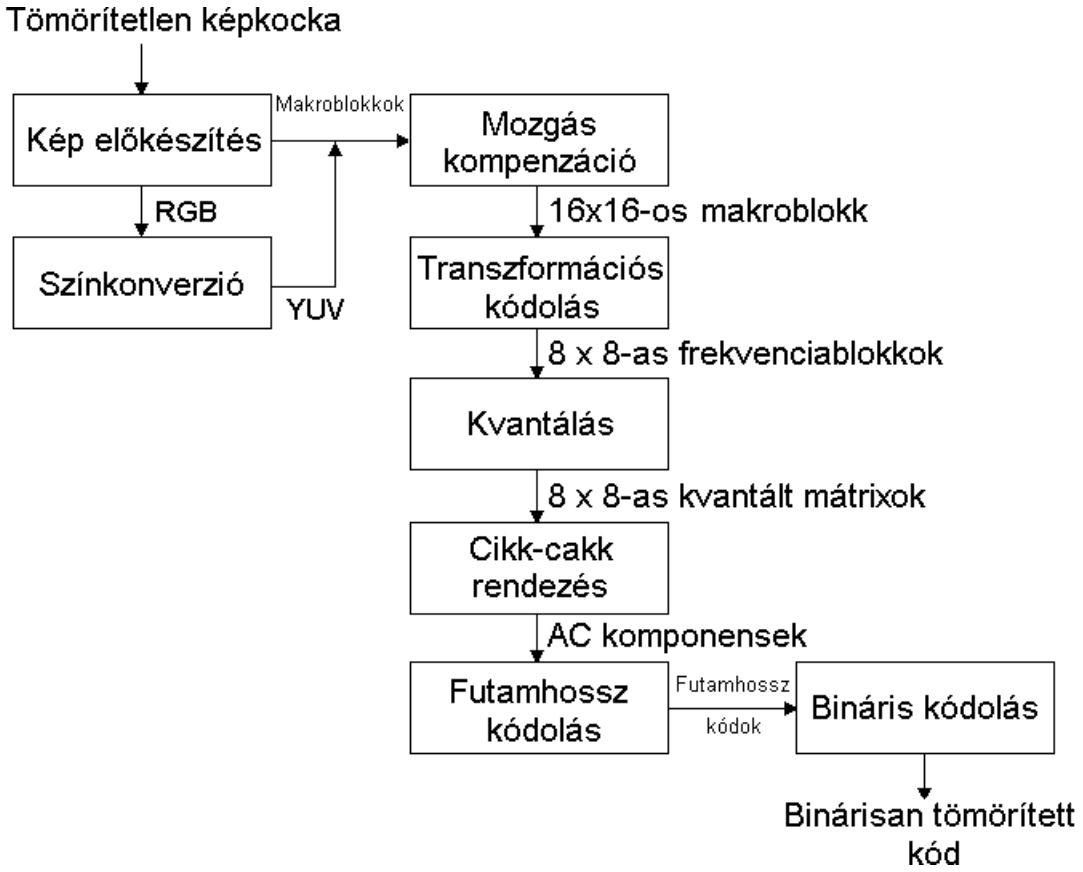
- Motion JPEG
  A JPEG algoritmus egy kibővített továbbfejlesztése állóképek sorozatának tömörítésére. A képeket önálló egységekként kezelik és tömörítik úgy, hogy a képek közötti redundanciákat nem veszik figyelembe.[11]
- PLV
  A Production Level Video, egy aszimmetrikus tömörítőalgoritmus, amelyet az Intel fejlesztett ki. A nyers videót egy ún. DCF (Digital Compression Facility), képkockánként dolgozza fel, Intraframe módon, amely jó minőséget eredményez.[12]
- Intel Indeo
  Az Intel által a Microsoft Windows alkalmazásokhoz (Video for Windows, QuickTime) kifejlesztett codec (coder/decoder), amely többfajta színmélységben (pl. 24 bitesben) is tud tömöríteni.[12]

- RTV
  Az RTV (Real Time Video) egy valósidejű videotömörítést valósít meg egyszerű algoritmussal.[12]
- MPEG
  Az MPEG (Moving Picture Experts Group) mozgóképek nemzetközi szabványává vált, amely nem defacto szabvány - nem eredeti ötletekből hozták létre -, hanem az ISO szabványokból indultak ki és erre építették. A következő fejezetben ezt a széles körben elterjedt algoritmust fogom bemutatni.

3.5.1. Az MPEG eljárás

3.5.1.1. A tömörítés folyamata

Az MPEG videokódolás meglehetősen összetett algoritmust követ. Az MPEG kétféle képkódolást használ, egyet az állóképként való tömörítésre (intraframe), és egyet a képek közötti összefüggések leírására (interframe). Ez a szabványban I ill. P és B képtípusként van deklarálva. Ehhez a három típushoz adtak hozzá még egy negyediket is - a D képeket - , amely az I kép egy speciális változata. Ezekről a későbbiek során még lesz szó. A kódolási folyamatban ezen két módszer természetesen együttműködik, nem lehet éles határvonalat húzni a kettő közé. Az 59. ábra mutatja a képkódolás menetét, amely elemeit külön-külön fogom tárgyalni.



59. ábra

Az első lépés annak megállapítása, hogy a tömörítendő képkocka intra- vagy interframe kódolással kerüljön-e feldolgozásra. Ahhoz, hogy ezt el tudjuk dönteni és a további

feldolgozás érdekében is előnyős, ha legelőször az adott képkockát előkészítjük a kódolásra. Csak ezen előkészítés után következhet maga a döntés, hogy miként kódoljuk a képkocka egyes részeit.

A képkocka azon részei, amelyek jelentősen megváltoznak a JPEG eljárásnál ismertetetett módon kerülnek tömörítésre (intraframe). Vannak az egymást követő képkockák között olyanok is, amelyek egyáltalán nem vagy csak néhány részletükben különböznek egymástól (interframe). Azok a képkockák, amelyek megegyeznek az előzőekkel nem kerülnek kódolásra. Azok pedig, amelyek bizonyos részletei változatlan tartalommal, de egy másik helyen jelennek meg mozgásvektor hozzárendeléssel kerülnek kódolásra. A *mozgásvektor* megadja a képrészlet helyét az új képkockán. Mint azt majd a 3.5.1.2. fejezetben tárgyalni fogjuk több MPEG fejlesztés is történt az elmúlt évek során, az elkövetkező pontokban az MPEG-1 rendszerű videotömörítés folyamatát fogom bemutatni.

a.) Képelőkészítés

A tömörítetlen képkockát először előkészítjük a további kódolásra. Első lépésként a színeket konvertáljuk át egy másik színtérbe. Ennek kiváltó okait és folyamatát a JPEG algoritmusnál már érintettem.

A színkonverzió következő lépése a színkomponensek (Y-U-V) képsíkokra osztása. Egy képsíkon egy adott színkomponenst tárolunk. Ettől kezdve minden sík el lesz választva egymástól, tehát a kódolási műveleteket minden egyes síkra külön-külön el kell végezni. A képelőkészítés során a képet blokkokra osztjuk fel. Az MPEG-1 szabványban 8x8 képpont méretű blokkok vannak előírva, de a későbbi szabványokban ettől eltérő méreteket is használhatunk. A különböző képsíkokon nem azonos méretű blokkokat hozunk létre. Erre azért van szükség, mert a luminencia (fényesség) komponens jellemzően meghatározza a képkocka minőségét, ezért azok értékeit kétszeresen mintavételezzük, azaz kétszerakkora blokkokat definiálunk számára (16x16). A krominancia (színezet) komponenseknek, mivel nem jellemzően határozzák meg a képminőséget, elég egy-egy 8x8 képpont méretű blokk. Ebből a két darab 8x8-as blokkból és a luminencia által használt négy darab 8x8-as blokkból alakul ki egy makroblokk. Az így előkészített makroblokkok kerülnek a mozgáskompenzáció bemenetére.[6]

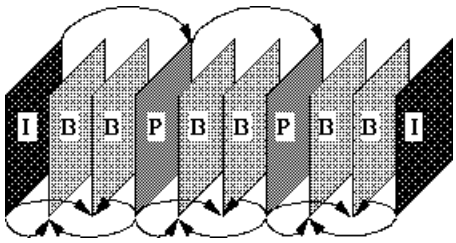
b.) Mozgáskompenzáció

A valóságban az MPEG négyféle képtípust definiál, ezekből állítják össze a képsorozatot:

- I-képek: (Intra Coded Pictures). A többi képtől függetlenül kerül tömörítésre. A JPEG algoritmushoz nyúlik vissza (intraframe), de a kódolás itt valósidejű (real-time).
- P-képek: (Predictive Coded Pictures). A kódoláshoz felhasználják korábbi I és P képek tartalmát (interframe kódolás).
- B-képek: (Bidirectionally Predictive Coded Pictures). A kódolásnál korábbi és időben későbbi I és P képek tartalmát is felhasználják. A leghatékonyabb képtípus. (interframe kódolás).
- D-képek: (DC Coded Pictures). Csak a DCT-ből származó alacsony frekvenciájú DC összetevőket kódolják, amelyet gyors bevezetőknél használnak, mint például a film elején, amikor csak fekete képernyő van.

Az előbb láthattuk, hogy a képelőkészítés során a kódoló az adott képkockát blokkokra osztja, méghozzá a különböző síkokon, különböző méretűekre. Továbbiakban az egyszerűség kedvéért a luminancia blokkokról fogok beszélni, amelyek 16x16 képpont területűek. A kódoló következő feladata, annak megállapítása, hogy milyen típusú kép következik. Amennyiben I képként kell kezelnie a képkockát, nem hajt végre semmilyen mozgásfigyelést és innentől állóképként fogja kezelni a kockát. Ellenkező esetben, ha P vagy B képről van szó, végrehajtódik a mozgáskompenzáció (MCP - Motion Compensation Prediction).

Egy képsorozat felépítését a 60. ábra mutatja. A két I kép közötti képsorozatot *képcsoport*nak (GOP-Group Of Pictures) nevezzük (61. ábra, 62. ábra). Az egész videónk ilyen GOP-ok sorozatából épül fel. Az első I kép a referenciakép, innen indul ki a többi nem I kép. A felső nyilak azokat a mozgásjóslásokat mutatják, amelyek a P képekre vonatkoznak. Megfigyelhetjük, hogy a P képek információi származhatnak korábbi I referenciaképből és P képekből is. Az alsó nyilak mutatják a B képek mozgáselőbecslését. Látszik, hogy mindkét irányba hivatkozhatnak, akár korábbi, akár időben későbbi I és P képekre. Ezt nevezzük *interpoláció*nak. B kép nem hivatkozhat másik B képre.

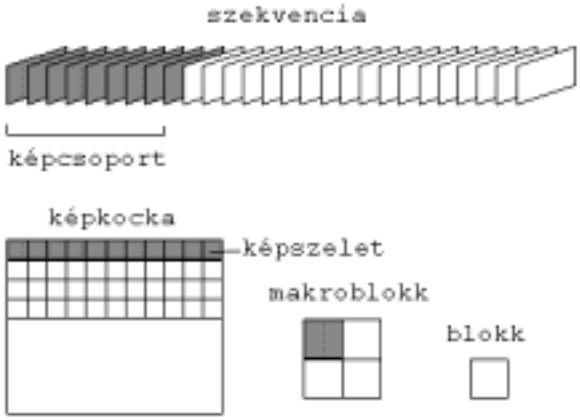


60. ábra

Természetesen mivel a B képek időben előre és hátra is hivatkozhatnak, a kódolt adatok általában nem időrendi sorrendben vannak tárolva, mint mondjuk egy tömörítetlen videó esetében, és a kitömörítés sem sorrendben történik. Először az I képeket dekódolják, majd az ezekre hivatkozó P képeket, végül az előző kettőre is hivatkozható B képeket.

Felmerülhet az a kérdés, hogy mikor, milyen képtípust hozzunk létre, és ezekből mennyit helyezzünk el egy képcsoporton belül, és egyáltalán hány képből álljon egy képcsoport? Ez utóbbi kérdésre válaszolva a gyakorlatban 10-15 képből szokott állni egy GOP, az MPEG alapértelmezésben 12 képkockát ajánl (0,4s). Nyilván a kis sorozatunk két I képet fog tartalmazni - egyet az elején, egyet a végén - ezeket a többi képtől függetlenül kódoljuk (állóképként). Tulajdonképpen az I képek sűrűsége azt határozza meg, hogy a tömörített videónk mennyire lesz szerkeszthető, tehát milyen időközönként lehet beleszólni az adatfolyamba (véletlen elérés - random access). Ha sok I képet teszünk a képsorozatba, könnyen szerkeszthető lesz, igaz a mérete jelentősen megnő. Ha kevés I képet definiálunk, nagyon jó tömörítési arányt érhetünk el, ám kevés beavatkozási pontunk lesz a videó menetébe. Láthatjuk, hogy kódolás során kompromisszumot kell kötnünk az I és P, B képek számarányában. Ez mindig a célalkalmazás függvénye. Az, hogy a P és B képek milyen gyakran követik egymást, előzetes tapasztalat kérdése. Ajánlatos egy P kép után kettő B képet használni. Természetesen minden GOP a képtípusok egy, előre meghatározott sorrendjéből épül fel. Ezt a sorrendet a kódolás során határozhatjuk meg. A B képek előnye tehát elsősorban a nagy hatásfokú tömörítésben rejlik, ám van más előnye is ennek a képtípusnak. Alacsony átviteli arány esetén, mivel a B képek átlagot képeznek

két makroblokk között, a zajt jelentősen csökkenteni tudják. *Zaj*nak nevezzük a hasznos jel mellett jelentkező, nem kívánatos, minőségi romlást okozó jeleket. A B képek hátránya viszont, hogy a makroblokkok átlagképzéséhez nagy tárolási kapacitás kell, így ez csökkenti a sávszélességet (63. ábra).[7][8]



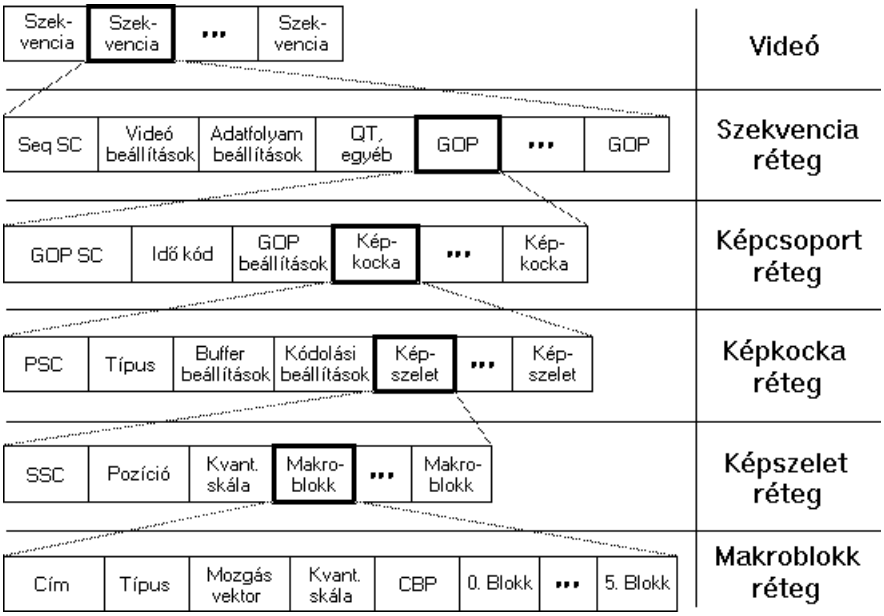
61. ábra

62. ábra

Mint érzékelhettük, az MPEG file felépítése a sok előre-hátra hivatkozás következtében igen bonyolult lehet, ezért az MPEG fejlesztők kidolgoztak egy rétegelt struktúrát, amellyel az MPEG file szerkezete áttekinthetőbbé, egyszerűbbé, és ami legfontosabb, egységessé válik. Ennek eredményeként könnyebbé válik a szerkesztés, a hibakeresés és a szinkronizálás is. A fejlesztők 6 réteget deklaráltak, amely rétegek mindegyike hordoz egy fejlécet, amely az adott réteg paramétereit állítja be. A rétegek hierarchikusan épülnek fel, tehát lefelé haladva egyre részletesebbé tehető az adatfolyam szerkezete. [9] [10]

1. Video-szekvencia réteg: maga a videófilmünk, mint például mozifilm, oktatófilm, stb. Több percnyi, egységes kódolással létrehozott képsorozat. Ez tartalmazza a GOP-okat, azaz a képcsoportokat.
2. Képcsoport réteg: Más néven GOP, amely 10-15 képkockából álló sorozatot jelent. A benne lévő I, P és B képek előre meghatározott sorrendben követik egymást. Ezt a sorrendet a kódolás során állíthatjuk be, amely nagy tapasztalatot igényel. Egy képcsoportban az I képek frekvenciáját - milyen távol van egymástól két I kép - N-nel jelölik (GOP(N)). A GOP-ok további alcsoportokra oszthatók, amelyekben a P és a B képek frekvenciáját, távolságát M-mel jelölik. Példánkban N = 9 és M = 3 (62. ábra). Szélsőséges esetben az N lehet 1 is. Ilyenkor minden kép I kép és ezt hívjuk képkocka-pontosan szerkeszthető MPEG-nek. Ez tulajdonképpen a Motion JPEG (MJPEG) szerkezetével analóg. Az N és M értékek általában előre meghatározottak, alkalmazás függőek.
3. Képkocka réteg: Maga a képkocka, amelynek típusa a korábban már fent említett I, P és B típusú kép lehet.

4. Képszelet réteg: Egy adott képkocka szeletekből épül fel, amelyek folytonosan helyezkednek el a képkocka területén. Ezek a szeletek azonos tulajdonságú makroblokkokból állnak. Célszerű egy képkockasornak beállítani, de más méret is megadható (61. ábra).
5. Makroblokk réteg (Macroblock Layer): Az MPEG 16x16 képpontos blokkokat definiál a luminancia komponensekre és 2 darab 8x8-as tömböt használ a krominancia komponensekhez (U, V). Az így kapott 6 darab 8x8-as blokk alkot egy makroblokkot.
6. Blokk réteg: A makroblokkokat 8x8-as különálló tömbökre osztják fel a további feldolgozás érdekében, mint például a transzformációs kódolás.[10]



63. ábra

Térjünk rá ezek után a mozgáskompenzáció lényegére. A kódoló szisztematikusan vesz egy blokkot, és az előző vagy a következő képkockákban keres egy vele teljesen megegyező vagy csak részben megegyező másik blokkot. Ehhez a kódolónak szüksége van egy olyan kereső modellre, amely képes összehasonlítani két mátrixot információtartalmuk alapján. A gyakorlatban több kereső módszer áll rendelkezésünkre, ilyenek például az abszolút hiba módszere, a teljes keresés módszere vagy a kétdimenziós logaritmikus keresés módszere.

Ezek közül kiemelve az abszolút hiba módszerét, amely a keresett blokkot hasonlítja össze más blokkokkal, keresve a minimális eltérést az információtartalomban (39).

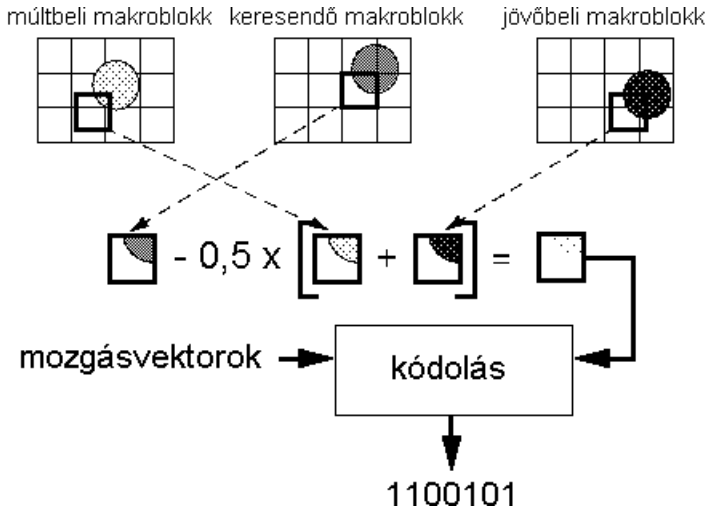
$$AH(d_x, d_y) = \sum_{i=0}^{15} \sum_{j=0}^{15} \left| f(i, j) - g(i - d_x, j - d_y) \right| \qquad (39)$$

A cél, hogy megtaláljuk azt a legkisebb AH értéket, amely a legkevesebb eltérést jelenteni a keresett blokkhoz képest. A (dx, dy) vektor jelöli azt a helyet, ahol ezt a legjobban megfelelő blokkot megtaláltuk. Az x és y a blokk koordinátáit jelölik. Az f(i, j) jelöli a keresendő blokk mátrixát, míg a g(i ,j) mátrix reprezentálja azt a blokkot, amelyet éppen a keresendőhöz hasonlítunk. Természetesen a keresés egy bizonyos [x, y] keresési területen belül zajlik le. A keresés sikeres befejezése után, a keresett blokkot helyettesítjük a (dx, dy) vektorral, amelyet mozgásvektornak nevezünk. A mozgásvektor tartalmazza a mutatott blokk képkockájának az azonosítóját és a blokk számát (x, y koordinátáit) a képkockán

belül. Ha B képről van szó, akkor a mozgásvektor két irányba tartalmaz vektort, egyiket egy előző blokkhoz a másikat pedig egy következő blokkhoz rendeli hozzá.

Ha a mutatott blokk teljesen megegyezik a keresett blokkal, azaz az abszolút hiba egyenlő nullával, akkor a mozgáskompenzáció befejeződik. Ha viszont az AH nem nulla - a két blokk között van különbség - akkor ún. predikciós hibáról beszélünk. A *predikciós hiba*, a keresett blokk és a hozzá leginkább hasonlító blokk közötti eltérés. Ebben az esetben szükség van arra is, hogy az eltérést rögzítsük, így kitömörítésnél ezt a dekódoló figyelembe tudja venni. A mozgásvektor mellé csatoljuk az eltérés mátrixot is, természetesen kódolva, amelynek szerepéről a későbbiekben még lesz szó.

B kép esetén a különbséget úgy tárolják, hogy a két mozgásvektor által mutatott blokk közötti eltérést interpolálják, azaz egy átlagot képeznek egy múltbeli és egy jövőbeli blokk információtartalma között. Erre láthatunk példát a 64. ábrán.



64. ábra

A képen látható, hogy a végső blokk mátrixa az eredeti blokk, továbbá a múltbeli és jövőbeli blokkok számtani közepének különbségéből származik. [12][13][14]

c.) Transzformációs kódolás

Mint azt az 1. ábra mutatja, a képelőkészítés után vagy egyből kódolásra kerül sor (intraframe) vagy pedig a kódolást megelőzi a mozgáskompenzáció (interframe). Mindkét esetben ugyanaz az eljárás fog lezajlani. A kódolás első lépése a transzformációs kódolás, amelynek bemenetei a makroblokkok. Ezek a makroblokkok vagy a tényleges képpont mátrixok, vagy pedig - mozgáskompenzáció esetén – a mozgásvektorokkal egybefűzött predikciós hibamátrixok. A transzformációs kódolás elvégzéséhez a képsíkokat külön kell kezelni. A luminancia síkon 16x16-os, míg a krominancia síkokon 8x8-as blokkokat kell képezni. Ezeken egyesével végre kell hajtani a 2.2.1. fejezetben tárgyalt Fourier-transzformációt (DCT).

A transzformációs kódolás célja, hogy a blokkban szereplő színértékeket olyan módon alakítsuk át és rendezzük el a mátrixon belül, hogy az a leghatékonyabban legyen kódolható. Tulajdonképpen azt használjuk ki, hogy a mátrixokban az egymás mellett lévő képpontok színinformációi összefüggenek egymással. A transzformáció eredménye egy együttható mátrix, amelyben a kép minősége szempontjából fontos összetevők együtthatói a mátrix egy bizonyos részében fognak csoportosulni (bal felső sarok), míg a relatíve kevésbé fontos összetevők együtthatói kis értékeket fognak felvenni, így később könnyű

lesz őket kiszűrni. Ezen jelentéktelen, kis amplitúdójú együtthatók nagy része el is hagyható a további kódolás során. Az MPEG szabvány ennél az eljárásnál a JPEG szabványra támaszkodik, amely transzformációs kódolásként a Diszkrét Koszinuszos Transzformációt (DCT) alkalmazza, amelyet részletesen a 2.3.1. fejezetben tárgyaltam. Itt csak példaként egy 8x8-as blokk 2D-s DCT-ját szeretném bemutatni (65. ábra).

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	162	160	159	159	159
159	160	161	162	162	155	155	155
161	162	161	163	162	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

→ DCT →
transzformálás

235	-1	-12	-5	2	-1	-2	1
-22	-17	-6	-3	-2	0	0	-1
-10	-9	-1	1	0	0	0	0
-7	-1	0	1	0	0	0	0
0	0	1	1	0	0	0	1
1	0	1	0	0	1	1	-1
-1	0	0	-1	0	1	1	0
-2	1	-3	-1	1	1	0	0

65. ábra

Látható, hogy a kiindulási mátrix (a 65. ábra baloldali része) szomszédos elemei nem sokban különböznek egymástól. A képelőkészítés során ezekből az értékekből 128-at le kell vonni, mivel a DCT csak a [-128..127] intervallumon értelmezett. Megfigyelhető, hogy az együttható mátrixban az első elem a legnagyobb - ez a DC komponens - míg a mátrix jobb-alsó sarka felé haladva mindinkább nullák következnek. Természetesen ezek már a kerekített értékek, hiszen a későbbiekben egész számokat fogunk véglegesen lekódolni. Így keletkezik az első veszteség. A DCT egy invertálható függvény, inverze az IDCT (Inverz DCT), ám mivel a kódolók az FDCT-t (Gyors DCT) használják, amelynek nincsen pontos inverze, ezért a dekódoló nem tudja egészen pontosan visszaállítani az eredeti blokkokat. A transzformációs kódolás végeredménye tehát, egy DC és 63 db AC együttható, mátrixos formában megadva.[14][15]

d.) Kvantálás

A transzformációs kódolás után olyan blokkokat kapunk, amelyekben különböző nagyságú számok szerepelnek. Ezeket a számokat kellene különböző nagyságú bithosszúságon tárolni, annak megfelelően, hogy minőségi szempontból mennyire fontosak. A DCT után láthattuk, hogy az alacsonyfrekvenciás - minőséget meghatározó - komponensek, együtthatók a mátrix (blokk) bal-felső sarkában helyezkednek el. A kisebb, nem annyira fontos összetevők pedig a jobb-alsó sarok felé találhatók. A DCT tehát egyfajta szűrést is elvégez a mátrix elemein. Ezt a szűrést finomítjuk tovább a kvantálással, méghozzá úgy, hogy a mátrixpontok helyén található együtthatókat egész osztással elosztjuk egy adott számmal. Még jobb hatásfokot tudunk elérni, ha ez az osztó szám változik (kvantálási mátrixba rendezve), tehát a fontos összetevőket kisebb számmal osztjuk - finomabban kvantáljuk, míg a kevésbé meghatározó összetevőket nagy számokkal osztjuk el - durvábban kvantáljuk. A kvantálás egyfajta értéksúlyozás, kiegyenlítés (normalizálás), melynek során a mátrix elemek saját fontosságukhoz képest kapnak értéket. A kvantálás egyszerűsített képlete tehát a következő:

$$K_{[i,\,j]} = \text{Integer}\left(\frac{B_{[i,\,j]}}{T_{[i,\,j]}}\right) \quad i, j = 0,1,2,...N$$

(40)

A képletben lehet látni, hogy egy N méretű K kvantált blokk [i, j] elemei kiszámíthatók, a B bemeneti (már transzformált) blokk és a T táblázat [i, j] elemeinek egész hányadosát vesszük. Azt a táblázatot, amely az egyes blokkelemekhez tartozó osztószámokat tartalmazza *kvantálási tábl*ának nevezzük. Minden blokkelemet tehát elosztunk a kvantálási tábla (44. ábra, 45. ábra) megfelelő helyén található számmal, majd kerekítjük. A kvantálási táblát külön meghatározzák a luminancia (44. ábra) és külön a krominancia (45. ábra) blokkokra.

Ezeket a táblákat, mivel előzetesen vannak definiálva, az adatfolyam mellé csatolják, mert nélkülük lehetetlen lenne a dekódolás. Az MPEG-nél felmerült az a probléma is, hogy egy képkockában a blokkokat nem biztos, hogy célszerű egyfajta kvantálással kódolni, hanem esetenként egy blokkot durvábban vagy finomabban is lehessen normalizálni. Erre két megoldás létezik: vagy definiálunk több kvantálási táblát, melynek hátránya, hogy helyet foglal, és ezzel rontja a tömörítés hatásfokát; vagy pedig egy adott táblázatot skálázunk ún. faktorokkal. Ez utóbbi lényege, hogyha egy blokk minden elemét szeretnénk kétszer olyan durván kvantálni, mint egy másik blokkot, akkor nem teszünk mást, mint a kvantálási táblázat értékeit kettővel megszorozzuk, így dupla akkora számmal fogunk osztani (normalizálni). Továbbiakban nem kell tárolni a módosított táblázatot, csak magát a *kvantálási faktor*t kell csatolni az adatfolyamhoz.

A következőkben egy példát mutatok be a kvantálásra, mégpedig az előzőekben eltranszformált blokkot (65. ábra) kvantáljuk, a fent bemutatott luminancia kvantálási tábla segítségével, példánkban a kvantálási faktor egyszeres. A 66. ábra jobboldali blokkjának értékeit megkapjuk, ha a 65. ábra jobboldali mátrixának értékeit osztom a 44. ábrán látható kvantálási mátrix elemeivel, majd egészekre kerekítem azokat.

Dekódolás során nincs más dolgunk csak az eltárolt kvantálási táblában lévő elemekkel beszorozni a blokk megfelelő elemeit, így majdnem teljesen visszakapjuk az eredeti blokkot. A veszteség a kódolás során végzett kerekítés miatt keletkezik, amely annál nagyobb lesz, minél durvábban hajtottuk végre a kvantálást. [14]

235	-1	-12	-5	2	-1	-2	1
-22	-17	-6	-3	-2	0	0	-1
-10	-9	-1	1	0	0	0	0
-7	-1	0	1	0	0	0	0
0	0	1	1	0	0	0	1
1	0	1	0	0	1	1	-1
-1	0	0	-1	0	1	1	0
-2	1	-3	-1	1	1	0	0

kvantálás ➔

15	0	-1	0	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

66. ábra

e.) Futamhossz kódolás

A kvantálás után olyan blokkokat kapunk, amelyek bal-felső sarkában behatárolt értékű ill. bitkiosztású egész számok vannak, míg a jobb-alsó sarok felé többnyire 0-ákat találunk. Ezeket a mátrixos formában lévő adatokat kéne valamilyen rendszer szerint adatsorba rendezni, hogy tovább lehessen folytatni a kódolást. Az MPEG erre a JPEG-től vett cikk-cakk rendezési algoritmust használja. A 47. ábra értelmében célszerű különválasztva kezelni a DC és az AC együtthatókat. A mátrix [0, 0] pontjában szereplő DC komponenst tesszük az adatfolyam elejére, megjelölve, hogy ez az egyenáramú összetevő, majd ezt követi (8x8-as blokkméret esetén) a 63 darab AC összetevő, méghozzá a 46. ábrán megadott sorrendben.

Az ábrán látható, hogy az átlókon megy keresztül az algoritmus az [1, 0] elemtől egészen a [7, 7] mátrixelemig. Ennek az eljárásnak köszönhetően az alsó tartományban található nullák egymás mellé fognak kerülni, így az további kódolást tesz lehetővé. Ezután következik a *futamhossz kódolás* (RLE - Run-Length Encoding), melynek során a kapott adatsorban lévő egymás melletti redundanciákat a minimálisra csökkentjük. Ez úgy zajlik, hogy sorban vesszük az adatokat és elempárokat képzünk belőlük a következő módon:

- az elempár első tagja az adott elem előtti nullák száma,
- az elempár második tagja maga az elem értéke (ami nem 0),
- az utolsó értékes (nem 0) adat után EOB (End Of Block - Blokk vége) jelet rakunk.

Ezzel az eljárással az ismétlődő nullákat redukáljuk le a minimumra. Az eljárás végén kételemű adatpárok állnak rendelkezésünkre, amelyeket további kódolásnak vethetünk alá. A DC együtthatókat külön kódoljuk az AC összetevőktől, méghozzá úgy, hogy mindig az előző blokk DC komponenséhez viszonyított különbséget tároljuk el (47. ábra). Az i-edik DC értéket a DC_i - DC_{i-1} képlettel számítjuk ki. Erre azért van szükség, mert az egymás után következő blokkok DC együtthatói nem sokban térnek el egymástól, különbségüket tehát kevés biten tárolhatjuk. Tulajdonképpen DPCM (Differential Pulse Code Modulation - differenciális impulzuskódolású moduláció) kódolást valósítunk meg. Nézzük meg a 3.5.1.5. fejezetben kapott kvantált mátrix futamhossz kódolásának eredményét (67. ábra), tételezzük fel, hogy $DC_{i-1} = 12$.

A kódolás végeredményeként kaptunk egy DC különbséget és 5 darab AC elempárt, a végén egy EOB szimbólummal. Mivel ezek az elempárok (szimbólumok) egy képkocka esetében gyakran előfordulnak, szükség van valamilyen előfordulást figyelő tömörítésre is. Ezt valósítja meg a *bináris kódolás*.[6][13]

15	0	-1	0	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

➔

DC: 15 - 12 = 3
AC: [1, -2];
[0, -1];
[0, -1];
[0, -1];
[2, -1];
EOB.

67. ábra

f.) Bináris (változó hosszú) kódolás

Az MPEG kódolás utolsó lépése a futamhossz szimbólumok változó bithosszú (VLC - Variable-Length Coding) kódolása. Nevezik ezt gyakoriság vizsgáló kódolásnak vagy *entrópia kódolás*nak is (1.3. fejezet), mivel az egyes szimbólumokhoz rendelt tárolási méret (bitszám) függ a szimbólumok előfordulási gyakoriságától. A gyakrabban előforduló elempárokhoz kevesebb bitet a ritkábban fellépőekhez több bitet rendelünk. Két olyan megoldás kínálkozik erre, amelyek a *Huffman-féle bináris kódolás*ra épülnek. Az egyiknél a kódolás közben vizsgáljuk meg a gyakoriságokat, és akkor hozzuk létre a bitkiosztó táblázatot; a másiknál pedig előre meghatározott szimbólumokhoz ún. Huffman kódszavakhoz vannak rendelve, és a kódolás során ebből a táblázatból keressük ki az adott elempárhoz tartozó kódot. Az MPEG ez utóbbit választotta a bináris kódolás elvégzésére. Az előre definiált táblázatokban tároljuk az egyes szimbólumokhoz tartozó kódszavakat, bitmintákat. A kódoláskor megvizsgáljuk a soron következő szimbólumot és kikeressük a hozzá tartozó bitmintát. Ezt hozzáfűzzük a bitfolyamhoz, majd vesszük a következő elempárt. Egy blokk végén a bitfolyamot byte méretű (8 bites) szeletekre osztjuk, majd így kerülnek tárolásra. Nézzük meg az előző fejezetben futamhossz eljárással kódolt elempárok bináris kódolását egy adott Huffman kódszó táblázat felhasználásával (68. ábra).

Kódtáblázat a szimbólumokhoz	
3 (DC)	011
[1, -2]	11011
[0, -1]	00
[2, -1]	11100
EOB	1010

A képzett bináris adatfolyam
01111011100000011100 1010
Byte-okra osztva:
01111011 0000001 11001010
123, 1, 202

68. ábra

Láthatjuk, hogy a bináris kódolás igen eredményes volt, hiszen a blokkunkban szereplő 8x8 = 64 byte adatból, mindössze 3 byte lett. Ez 1:20 tömörítési arányt jelent! Ezeket a byte-okat megfelelő fejléccel ellátva a Huffman kódtáblával együtt az MPEG adatfolyamhoz csatoljuk és következhet a következő blokk kódolása.[6]

3.5.1.2. Az MPEG szabványsorozat

a.) Az MPEG-1 szabvány

1992-ben készült el az MPEG első ajánlása az MPEG-1. A fő cél olyan kódolási technika kifejlesztése volt, amely támogatja az audio és videó jelek számítógépes feldolgozását. A fejlesztők sztereó hangot, kétfajta felbontású képméretet (352*240 NTSC és 352*288 PAL), maximum 30 képkocka/másodperc képváltási sebességet (frame rate) és jó, VHS minőségű lejátszást garantáltak az ajánlásukban. Mivel akkoriban az 1x CD-ROM volt elérhető a piacon, ezért erre a sebességre (1.5 Mbit/s) optimalizálták a lejátszást.

b.) Az MPEG-2 szabvány

Az MPEG fejlesztők 1993 novemberében publikálták az MPEG-2 Bizottsági Tervezetét, amely 1994 óta egy általánosan alkalmazott nemzetközi szabvány. Céljuk a jó minőségű videók (DVD-Digital Video Disc) kódolásának támogatása, mint például a műholdas műsorsugárzás megvalósítása. A sávszélesség 2 és 15 Mbit/s között mozog, a felbontás négyszer nagyobb, mint az MPEG-1-nek, tehát 704x576 PAL rendszerben és 704x480 NTSC rendszerben. A kódolás és dekódolás is bonyolultabb lett, ezért a lejátszáshoz kb. tízszer akkora számítási igényre van szükség. Az MPEG-2 képes kezelni az átlapolt (interlaced) megjelenítésű videókat is. A megjelenítés sebessége továbbra is 25-30 FPS. A hangminőség továbbra is CD minőségű, ám több csatorna használatát is lehetővé teszi, ezáltal támogatja a térhangzást is. A kódolási eljárások és technikák nagy része az MPEG-1-re épül, megegyezik azzal. Az MPEG-2 felülről kompatibilis, tehát az MPEG-2 képes kezelni az MPEG-1 adatfolyamokat is.

c.) Az MPEG-3 szabvány

Az MPEG-3 célja a HDTV (High Definition Television) alkalmazások kiszolgálása, méghozzá 1920x1080 felbontású és 30 FPS sebességű megjelenítés felhasználásával. A bitarányt 20-40 Mbit/s nagyságúra tervezték. Később rájöttek, hogy az átviteli arány finomításával az MPEG-2 is képes ezeket a paramétereket kezelni. A lényeg az, hogy meg kellett találni a kompromisszumot a mintavételezés és az átviteli sebesség között. Ma már a HDTV az MPEG-2 High 1440 szint ill. a High szint része, ezáltal az MPEG-3 az MPEG-2 részévé vált.[7]

d.) Az MPEG-4 szabvány

Az MPEG-4 1998-ban jelent meg és alapjaiban változtatta meg az eddigi mozgókép kódolási technikákat. Ez az új szabvány egyesítette a kommunikációt, a mozit és a multimédiát egy egységes keretbe foglalva. Az MPEG-4 célja az igen alacsony átviteli sebességű, kisfelbontású interaktív mobil kommunikáció, a videotelefónia, a mobil audiovizuális kommunikáció, a távérzékelés, a multimédiás elektronikus levelek, az elektronikus újságok, az interaktív multimédia adatbázisok, a játékok valamint a multimédiás videotex területén működő mozgóképek kódolása volt. Az átviteli arányt 2 kbps és 64 kbps közé optimalizálták, míg a felbontást 176x144 képpontra és a képváltási sebességet 10 FPS értékűre tervezték. Ez az első szabvány, amely átlépi a passzív szemléltetés határait és bevezeti az interaktivitást. Ettől kezdve a képkockákat nem szín- és hangadatok halmazaként fogják fel, hanem audiovizuális objektumok egymáshoz kapcsolódásaként definiálják. A dekóder ezekből az objektumokból állítja össze a képkocka eredeti tartalmát. Az objektumok bizonyos tulajdonságait, mint például a hely, méret, sebesség, meg lehet változtatni, vagy akár törölhetjük és be is szúrhatjuk őket egy adott képkockába. Az interaktivitás megvalósításához új kódolási eljárásokat kellett kidolgozni, amelyek olyan objektumokat írnak le, amelyek az ember gondolkodási sémáihoz igazodnak, hiszen nem elvont adathalmazokat szeretnénk kódolni, hanem számunkra jelentéssel bíró összefüggéseket, kompozíciókat. Ehhez a teljesen új kódolási eljáráshoz a korábbiakhoz képest teljesen új adatstruktúrát kellett kifejleszteni. Az MPEG-4 támogatja a sztereovizuális kódolást, tehát a két szemünkkel látott különböző képeket egy adatfolyamba tudja integrálni, így akár egy jelenet (szekvencia) több különböző nézőpontból felvett képét is képes magába foglalni (virtuális valóság). Másik nagy előnye az MPEG-4-nek, hogy a természetes és a mesterséges képtartalmak tetszőlegesen keverhetőek egymással. [7][16]

e.) További fejlesztési elképzelések

Az MPEG fejlesztőcsoport az ezredforduló környékére tervezi az MPEG család újabb tagjának, az MPEG-7-nek a megjelenését. Az MPEG-5 ill. MPEG-6 kimaradt, nem hozták létre. Ennek oka az volt, mivel az MPEG-3 szabványt menetközben törölték (mert az MPEG-2 megoldotta a funkcióit), a tervezők úgy gondolták, hogy a sorszámok legyenek a 2 hatványai, így az 1, 2 és a 4 után a 8 jön. Később rájöttek, hogy a 8-as elé szeretnének még egy verziót beiktatni, tehát létrejött az MPEG-7.

Az MPEG-7 feladata lenne a keresési képességek kiterjesztése minden információtípusra. A mai technológiák még csak szöveg alapú keresést használnak, mint például az Internetes keresők. Az audiovizuális keresés egyenlőre még nem megoldott probléma, a fejlesztők ma ezen dolgoznak. Az MPEG-7 ezt a korszerű keresési módszert fogja alkalmazni, tehát tartalom alapján is lehet majd keresni benne. Például lehetővé tesz majd olyan keresést, mint például egy jelenet, ahol „bal oldalt egy ház van és egy autó jön jobbról”.

A tervek szerint az MPEG-8 támogatni fogja a tárgyak négydimenziós leírását.[16]

3.5.2. Az AVI eljárás

Az AVI (Audio Video Interleaved) a Video for Vindows standard állomány típusa. Az AVI állományokban az audio- és videoadatok egymásután követik egymást. A hangadatokat WAVE formában, míg a képeket DIB formátumban tárolja. A DIB (Device Independent Bitmap) olyan eszközfüggetlen pontmátrixos képtípus, amelyet a Windows megjelenésekor fejlesztettek ki a multimédia támogatására. A BMP-hez hasonlóan az így előállított kép független a videokártyától és 24 bit színmélységű. Az AVI-ban a képeket nem állóképek sorozataként tároljuk, hanem egy teljes képből kiindulva a következő képkockák (frame) mindig csak az előzőtől mért különbéget tartalmazzák (lásd MPEG). Az ilyen ún. differencia képeket delta-frame-nek nevezzük. Fontos kérdés, hogy mekkora az a legkisebb különbség két egymást követő képkocka között, amelyet még változásnak tekintünk. Ezen küszöbérték hatással van a tömörítési aránytényező nagyságára és a kép minőségére is. Alacsony küszöbérték kismértékű változást is érzékel, ennek következtében a deltakeret kicsi, míg a tömörítési aránytényező nagy lesz. Magas küszöbérték jelentős mértékű változásnál nagy, de ritkán előforduló deltakeretet eredményez jó tömörítési hatékonyság mellett, viszont a kép gyengébb minőségű lesz. A pszichovizuális – emberi látás tulajdonságait figyelembe vevő – elvek figyelembe vételével megállapítható egy helyes küszöbérték. Az AVI állományok előállítása lehetséges külső videoforrásból videodigitalizáló kártya segítségével, vagy néhány animáció készítő, videoszerkesztő programnak is lehet ilyen formátumú kimenete.

A lejátszás során a kitömörített videoadatok a videokártya felé, az audioadatok a hangkártya felé áramolnak. A két részfolyamat időben összehangoltan játszódik le, ezt *szinkronizálás*nak nevezzük.

A digitalizált videoklippek képeinek felbontása 160*120, 192*144, 320*240 képpont, színmélység általában 8 bit.

4. AUDIOFELDOLGOZÁS ÉS TÖMÖRÍTÉS

4.1. Hangtani alapfogalmak

Mielőtt megvizsgálnánk a hangtömörítési eljárásokat, nézzünk meg néhány hangtani alapfogalmat. A hang a levegő molekulái által tovaterjedő mechanikai rezgés. Eklatáns példa erre egy megpendített hangvilla, amelynek rezgése következtében a levegő molekulái hol sűrűbb, hol pedig ritkább tartományokat alkotnak, ahol a nyomás egy pillanatban megnő ill. lecsökken. A rezgés a hangforrásból (példánkban a hangvilla) kiindulva egyre csillapodó rezgéshullámként terjed az akusztikus tér minden irányába. A hangvilla hullámformája a legegyszerűbb: szinuszhullám. A hanghullám jellemzésére az alábbi alapfogalmak szolgálnak:

- frekvencia
  A másodpercenkénti rezgésszám, műszeres mérésre alkalmas érték (ν [Hz]). Az egészséges emberi fül érzékelési tartománya 20 Hz – 20 kHz.
- hangmagasság
  A frekvenciához hasonlóan a rezgések egy adott időn belüli számát adja. Eszerint minél több az adott időn belül a rezgések száma, annál magasabb az adott hang. A hangmagasság tehát inkább a rezgés észleléséről tájékoztat. Ilyen értelemben beszélünk magas és mélyhangokról. A hangrögzítéskor és lejátszáskor az alsó frekvenciatartományt szokás basszusnak, míg a felső frekvenciatartományt szopránnak nevezni.
  - amplitúdó
    Az amplitúdó a maximális kitérés, a hullám erősségének mérőszáma (A [μm]).
  - hangerősség
    A hang erősségét két hang „hangosságának” összehasonlításaként is definiálhatjuk. A „hangosság” mérőszáma a decibel [dB]. Az emberi fül elég érzéketlen, ami azt jeleni, hogy csak a nagy mértékű hangerősség változásokat képes érzékelni, de azt elég nagy tartományban. Az érzékenységi tartományt dinamikatartománynak is nevezik. Mivel az emberi fül a hangerősséget logaritmikusan érzékeli, úgy a dB-t is logaritmikus léptékkel mérjük. Például az 86 dB-es hang erőssége kétszerese az 83 dB-esének. A hangerősségben bekövetkezett 3 dB-es változás az a legkisebb érték, amit még az emberi fül érzékelni képes.

Mindezek után térjünk rá a *hangrögzítés és lejátszás* folyamatára, amely az alábbi műveletekből áll:

- a rögzítendő hang érzékelése (mikrofon),
- a rögzítendő hang átalakítása rögzíthető jellé (moduláció, mintavételezés -hangkártya),
- a rögzíthető jel tárolása (háttértár),
- a rögzített jel visszaolvasása (háttértár),
- a rögzített jel átalakítása hallható hanggá (hangkártya),
- a hallható hang megjelenítése (hangszóró).

Egyre inkább fontos színfoltjai a számítástechnikának a digitális szintetizátorok, amelyek alkalmasak *digitális zene előállítására*. Kidolgozták a MIDI protokollt, amely lehetővé tette külső eszközök hozzákapcsolását a számítógéphez, sőt a hangkártyákra is szereltek ún. FM szintetizátort.

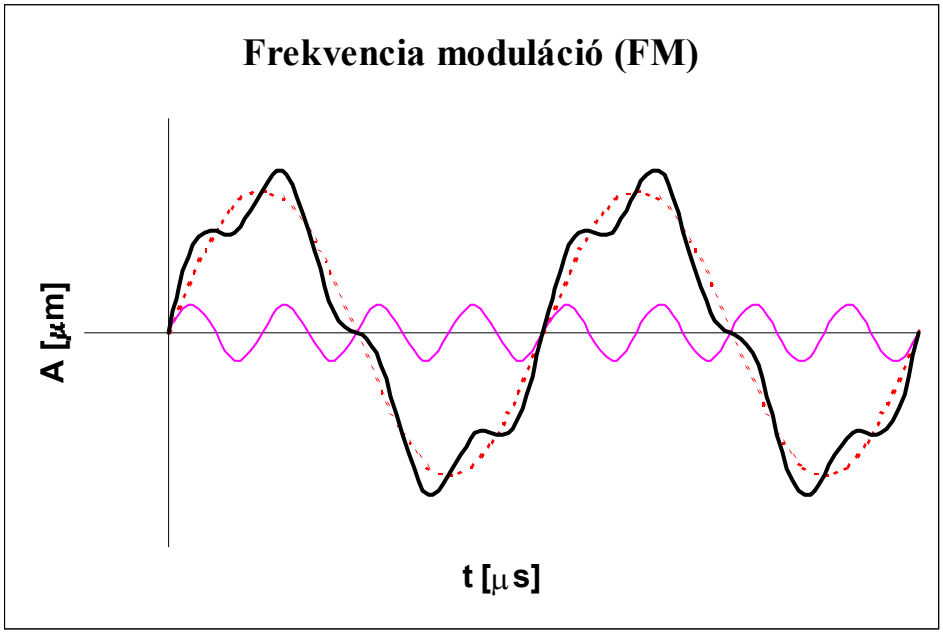
4.2. Hang előállítási módok

A hangok előállítása alapvetően két eljárás szerint történhet: frekvencia moduláció, mintavételezési eljárás.

A zenei hangok szintézisére J. Chowning 1971-ben alkalmazta először a *frekvencia modulációt (FM)*, amelyen egy hullám frekvenciájának egy másik általi kicsiny megváltoztatását értjük. Az így kapott összetett hanghullám tartalmazza a két kiinduló komponenst, összegüket, különbségüket, továbbá felharmonikusaikat is. Ez utóbbiak az eredeti komponensek (alap)frekvenciáinak egészszámú többszörösei és ezek adják mindenféle hang jellegzetes hangszínét. Mára az FM szintézis a legelterjedtebb módszer zenei hangok létrehozására, ennek megvalósítása érdekében a legtöbb hangkártyára beépítettek egy FM szintetizátort is.

Az FM eljárás alapját a Fourier transzformáció (DFT) jelenti. Ennek elméleti hátterét a 2.2.1. fejezetben érintettem. A Fourier elmélet szerint az összetett hullámok egyszerű hullámok sorozatára bonthatók ill. ennek a megfordítottja is igaz. Ezek lehetnek alapfrekvenciák (alapkomponensek) és felharmonikusok. A 69. ábrán szaggatott vonal jelöli az alapfrekvenciát, vékony folytonos vonal a harmadik felharmonikust (frekvenciája négyszerese az alapénak) és vastag folytonos vonal az összetett (itt összeg) hanghullámot.

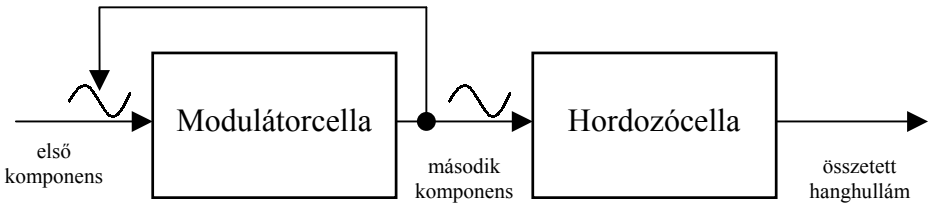
A mai hangkártyák támogatják a két, ill. négyoperátoros FM-szintézist, amelyekkel sztereóhang is előállítható. Ezen célok megvalósítására ún. Yamaha chip-eket építenek a hangkártyára. A kétoperátoros FM-szintézis elvét a 70. ábra mutatja, eszerint a két komponens előállítása az egyes operátor cellák feladata. Az ún. modulátorcella határozza meg a hangszínt a felharmonikusokon keresztül, míg az ún. hordozócella a hang magasságát adja.



69. ábra

A kétoperátoros üzemmódban az FM szintetizátor maximum 16 melodikus és 6 féle ütős hangot tud egy időben létrehozni. A melodikus hangokat, amelyek hangszereket utánoznak, szoftverrel állítjuk elő, majd azokat zenei szerkesztő programokkal átalakíthatunk. Az ütős hangszerek hangját gyárilag programozzák. Hangzás

szempontjából ezek lehetnek mono, ill. sztereo hangzásúak, legfeljebb csak feleannyi hangszert lehet „megszólaltani” egyszerre.

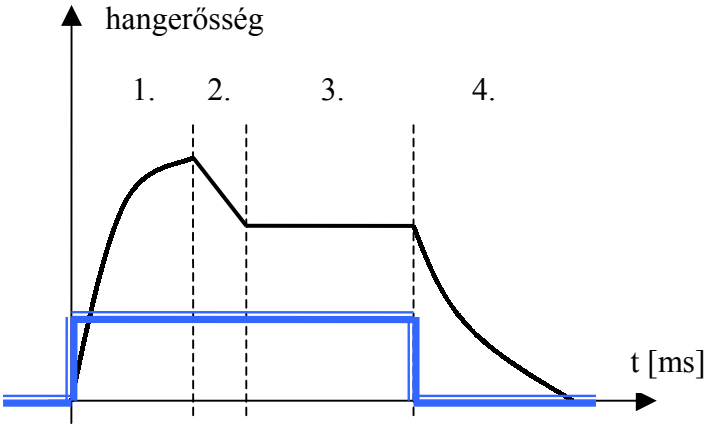


70. ábra

A négyoperátoros üzemmódban két cellapár van sorba kapcsolva, így gazdagabb hangzás állítható elő.

A hangszínt az operátorcellák frekvenciáján kívül még befolyásolja a *burkológörbe* néhány jellemzője is. A burkológörbe egy hang erősségének lefutását adja az idő függvényében. A burkológörbe (71. ábra) négy szakaszának (emelkedés, lecsengés, szinttartás, kicsengés) bármelyikét megváltoztatva további hangzásokat állíthatunk elő.

A *mintavételezési eljárás* tulajdonképpen szűrők alkalmazását jelenti, amelyek típustól függően csak megadott frekvencia tartományt képes átereszti. Azt a frekvenciát, amelyet a mintavétel időpillanatában a hangkártya az adott szűrő kiválasztásához megállapít, *határfrekvenciá*nak nevezzük. Minden szűrőhöz tartozik ilyen határfrekvencia. Ezek alapján lehet kiválasztani az éppen megfelelőt. A határfrekvencia éppen a Nyquist-határ alatt van. A Nyquist-határ az a legnagyobb frekvencia, amelyből még pontosan lehet mintát venni. Ez a határérték a mintavételi frekvenciának éppen a fele. Ezt egy példával illusztrálva: ha a mintavételi frekvencia 44.1 kHz, akkor a Nyquist-határ 22.05 kHz, a határfrekvencia pedig egy kicsivel kevesebb ennél, tehát 20 kHz.

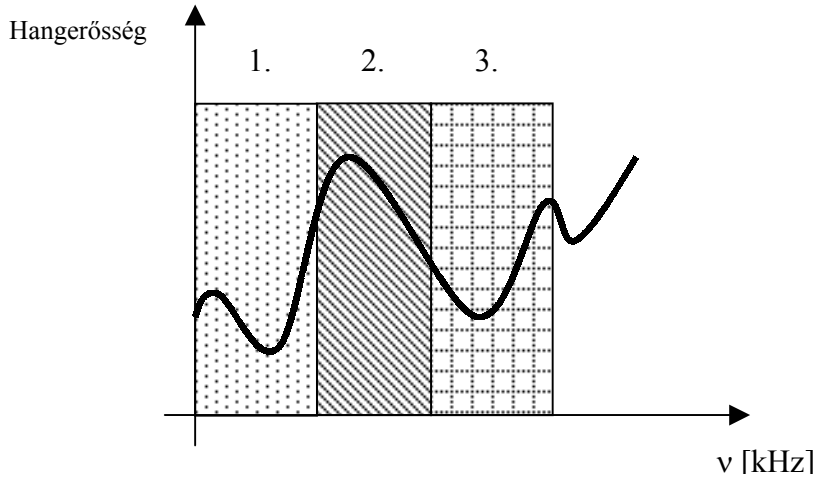


1. Emelkedési fázis
2. Lecsengési fázis
3. Szintentartási, kitartási fázis
4. Kicsengési fázis

A kettős vonal egy billentyű megnyomását ill. felengedését szemlélteti.

71. ábra

Alapvetően háromféle szűrőt definiáltak: mélyáteresztő (2.), magasáteresztő (1.), sávszűrő (3.). A magasáteresztő szűrő csak a magas hangokat, a mélyáteresztő csak a mély hangokat, míg a sávszűrő csak a kettő közötti hangokat engedik át 72. ábra.



72. ábra

Mint az előbbi példában láttuk a szűrőt csak ideális esetben tudjuk beállítani a Nyquist-határra, ezért az alatti értéket kell beállítanunk, amely pedig alulmintavételezést (aliasing) okoz. Ennek eredményeként hamis hangok kerülhetnek a felvételünkbe. Ezért követelmény, hogy a Nyquist-határ és a gyakorlatilag beállított határfrekvencia nagyon közel legyen egymáshoz.

A mintavételezéses vagy más néven hullámtáblázatos szintézis azt jelenti tehát, hogy ROM jellegű memóriában tárolják a hangszerek 16 bites felvételeit. Ezekből a mintákból „manipulálják" az éppen előállítandó hangokat. Az FM eljárásnál tökéletesebb hangzást eredményez, hiszen egy hétpontos interpolációt alkalmaz mintavételezéskor azért, hogy jól közelítse az adott hangszer valódi hangját.

4.3. Audiodigitalizálás

Ahhoz, hogy a számítógéppel hangot tudjunk rögzíteni, szerkeszteni ill. lejátszani előbb az analóg audiojelet digitalizálni kell ill. a digitális jelet analóg akusztikus jellé kell átalakítani. Ez folyamat korántsem fogja az eredeti jelet visszaadni. A hangkártyán van egy DAC-ADC (impulzuskód-modulációs áramkörök) egység, amely az előbb vázolt műveleteket elvégzi. Ez a folyamat két egymással szorosan összefüggő két részre bontható: mintavételezés, kvantálás.

Amennyiben előre megadott szabályos időközönként megmérjük az analóg akusztikus jel nagyságát, úgy azt diszkrét digitális jellé alakítjuk át (68. ábra). Ezt a folyamatot, mint azt korábban már láttuk, *mintavételezés*nek nevezzük. Jó esetben ennek a diszkrét jelsorozatnak az információtartalma megegyezik a folytonos akusztikus jelével. Ennek teljesüléséhez az kell, hogy a mintavételi frekvencia értéke legalább kétszer akkora legyen, mint az eredeti analóg akusztikus jelben előforduló legmagasabb frekvencia (lásd Nyquist-határ). Ahhoz, hogy Hi-Fi (természethű hangvisszaadás) minőségű hanghatást érjünk el a mintavételi frekvenciát 40 kHz körüli értékűre kell választani. A mintavételi frekvencia értékeket rögzítették: 11.025 kHz, 22.05 kHz, 44.1 kHz egy (mono) vagy két (sztereó) csatornára vonatkoztatva, hangkártyától függően. A Sound Blaster 16 típusú hangkártya például 44.1 kHz mintavételi frekvenciát képes produkálni mind a két csatornában. Befolyásolja a mintavételi frekvenciát az egész rendszer sávszélessége. *Sávszélesség*nek nevezzük azt a frekvenciatartomány, ahol az amplitúdó 3 dB-lel csökken. Nincs értelme 44.1 kHz-es mintavételezésnek ott ahol egy 12 kHz-es mikrofonnal kívánunk egy mélyhangú férfi beszédét felvenni, amelyben ritkán találunk 7 kHz feletti komponenseket.

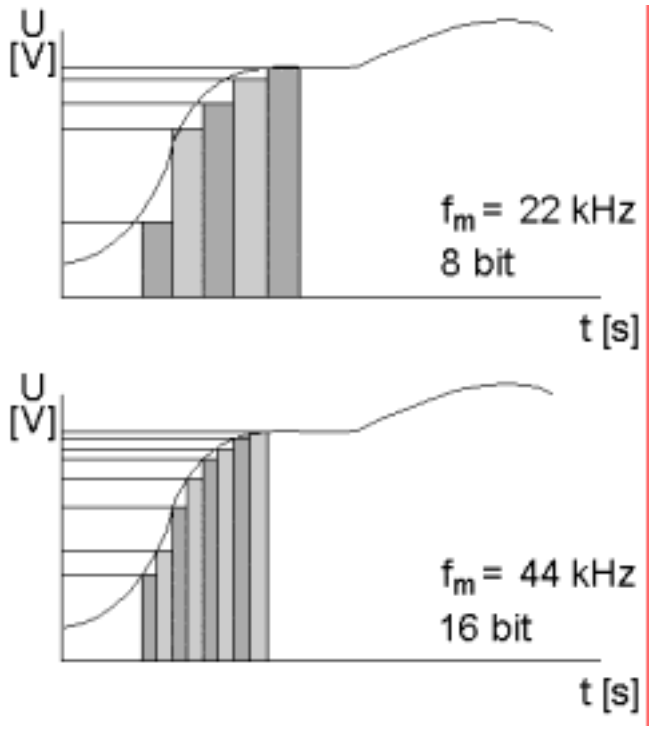
A jelátalakítás következő lépése a *kvantálás.* A folyamat általános bemutatását már a 2.1.3. fejezetben láthattuk. A mintavételezéssel kapott amplitúdóminták még végtelen sok értéket vehetnek fel, tehát bizonyos értelemben analóg jelnek tekinthetők. Ezeket az értékeket egy „mintatáblázattal vetjük össze" és a mintavételezett értékekhez legközelebbi mintaelemet választjuk ki. A minta nagysága lehet 8 bites (256 elemű) és 16 bites (65536 elemű). Az így kapott kvantált értékeket kódoljuk, majd a kódban szereplőnulla és egyes bitértékeket impulzusonként folyamatosan egymásuán írva kapjuk meg a PCM (Pulse Code Modulation) jelsorozatot. Az impulzus-kódmuduláció (PCM) elvét a 73. ábra mutatja be.

Összefoglalva a digitális hangállomány minőségét három tényező együttesen befolyásolja: a mintavételi frekvencia, a kvantálási szóhosszúság, a csatornák száma (mono, sztereó). Minél jobb minőségű digitális hangot szeretnénk kicsiholni hangkártyánkból, annál gyakrabban kell mintavételezni nagyobb kvantálási szóhosszal és több csatornára vonatkoztatni. Ehhez azonban nagy állományméret tartozik.

Az audioadatok tárolásához T tárigényt [MB/s], ν a mintavételi frekvencia [1/s], N a csatornák számát [csatorna]és K a kvantálási szóhossz [bit/csatorna]:

$$T = (\nu * N * K)/(8 * 1024 * 1024) \qquad (41)$$

Legyen a csatornák száma N = 2 (sztereó), a mintavételi frekvencia 44100 Hz, a kvantálási szóhossz 16 bit/csatorna, így a tárkapacitás másodpercenként 0.168 MB/s. Így egy 1 perces sztereóhanganyag tárigénye ≈10 MB. Ebből egyértelműen következik ismételten a tömörítés szükségessége. Amennyiben jó minőségű beszédet akarunk rögzíteni, akkor elégséges 22.05 kHz-en mintavételezni és 8 ill. 16 bites kvantálási hosszt alkalmazni. Hi-Fi minőségű zene felvételéhez már 44.1 kHz-es mintavételi frekvenciára és 16 bit kvantálási szóhosszra van szükségünk.



73. ábra

4.4. Audiotömörítési algoritmusok

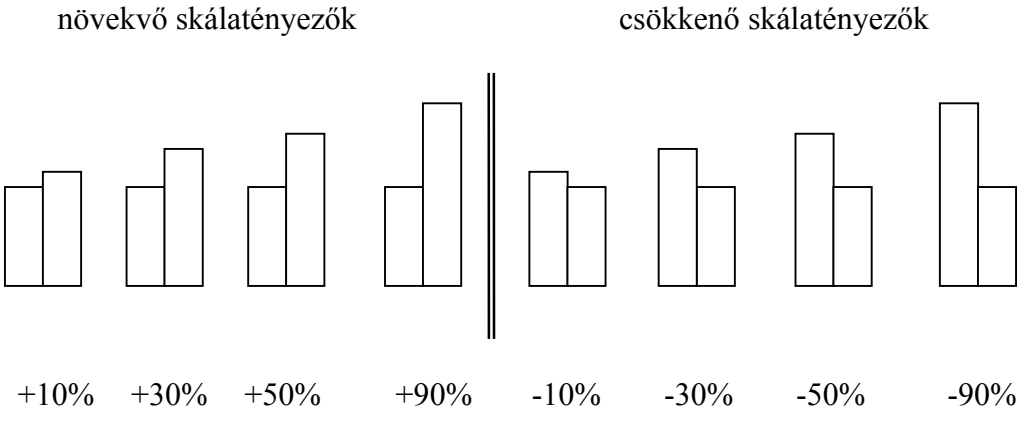
Mint az előző fejezetben láttuk néhány másodpercnyi hanganyag mérete óriási lehet, ezért szükségünk van tömörítésre. Megvizsgálva a 1.3. fejezet elején az adattömörítéssel szemben támasztott követelményeket, a hangtömörítésre az alábbi megállapításokat tehetjük:

- A hanganyagok eléggé különbözőek lehetnek, ezért elég változatos tömörítési aránytényező mellett tömöríthetőek, természetesen veszteségesen. Például az emberi beszéd a gyakori szünetek miatt jobb hatásfokkal tömöríthető, mint a zene.
- A tömörítéssel kapott hang minőségének közel olyannak kell lennie, mint az analóg akusztikus jelnek.
- A tömörítési algoritmusnak az emberi hallás legfontosabb tulajdonságainak figyelembe vételével kell történnie (pszichoakusztika).
- A kódolás-dekódolás ne okozzon időkésleltetést (valósidejűség - real time).

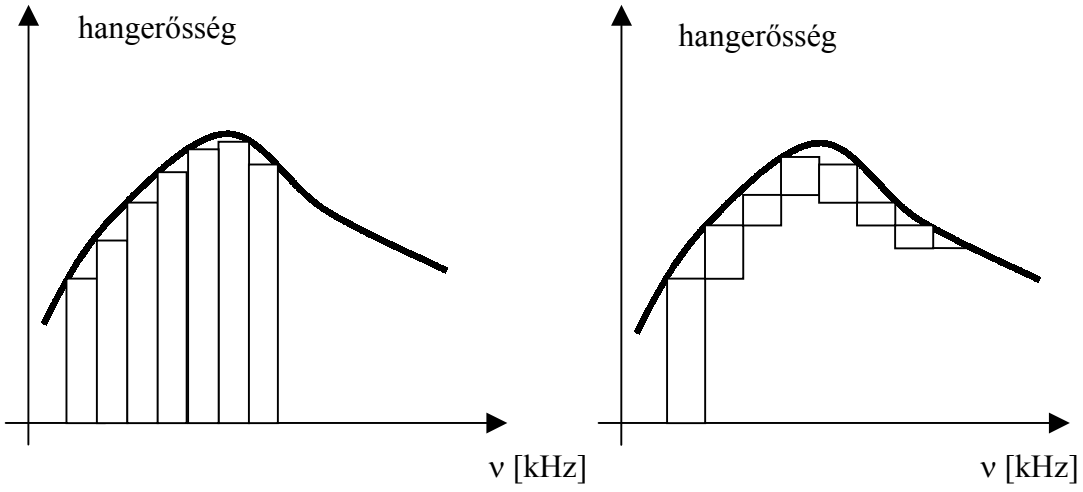
A hangtömörítésekben alkalmazott legfontosabb eljárások:

- veszteséges előrebecsléses eljárások (DPCM, ADPCM)
  A 1.3.2. fejezetben tárgyaltam a veszteséges előrebecsléses tömörítést, mint alapelvet, nézzük ennek alkalmazási lehetőségeit a hangtömörítéseknél. A *differenciális impulzus kód moduláció (DPCM)* a PCM jel mintavételi értékeit használja fel. Ez az eljárás lineáris kvantálási karakterisztikát követ, miszerint elegendő csak az első PCM mintavételi értéket megadni, a többieknél pedig az ehhez viszonyított eltérést kódolják. Így nem kell minden mintavételi értéknél a teljes kvantálási szóhosszal megadni. Az *adaptív differenciális impulzus kód moduláció (ADPCM)* nagyon jól figyelembe veszi a PCM jelben bekövetkező változásokat. Az ADPCM eljárásnál a különbségi értékeket kevesebb bitszám mellett kódolják, mint a DPCM-nél, mégis pontosan tudja követni még az összetett hanghullámot is. Az ADPCM módszer egy 4 bites értéket, ún. skálatényezőt rendel a mintához. Ez az érték nem a mintavételezett amplitúdó, hanem egy olyan skála tényező (φ_n), amellyel az előző minta amplitúdóját (A_i) kell megszorozni ahhoz, hogy az aktuális minta amplitúdóját (A_{i+1}) megkaphassuk ($A_{i+1} = A_i * \varphi_n$). A φ_n skálatényező lehet 4 (16 elemű) ill. 8 bites (256 elemű). Hátránya az eljárásnak, hogy a jelben hirtelen bekövetkező jelentős változásokat nem tudja követni, ami jeltorzítást okoz. Az így elérhető tömörítési aránytényező 0.25-0.5.
  Az 74. ábra a 16 elemű skálatényezők közül mutat nyolcat, míg az 75. ábra összehasonlítja a DPCM (a) és az ADPCM (b) eljárásokat. Az utóbbinál a skálatényezők balról jobbra haladva a következők: +30%, +20%, +20%, -10%, -20%, -10%, stb.
- A-szabály, Ω-szabály
  Az A-szabályt inkább Európában, míg az Ω-szabályt inkább az USA-ban és Japánban használják emberi beszéd tömörítésére. A lineáris kvantálású, egyenletes osztásközű DPCM helyett célszerű alkalmazni a logaritmikus osztásközű skálát, amely egyrészt jól illeszkedik a fül érzékeléséhez, másrészt jól közelíti a zenében bekövetkező hangerősség változásokat is. Az eljárásnak az a lényege, hogy a kisebb amplitúdójú jelek felbontása nagyobb, mint a nagyobb amplitúdójúaké. Ez a két eljárás a 8 bites áramkörével olyan jel/zaj arányt és dinamika tartományt tud elérni mint a DPCM eljárás a 12 bites áramkörével. Ennek az az oka, hogy az analóg jelre szuperponálódó zaj kisebb amplitúdójú mint maga a jel. Ezt a tételt kihasználva a kis amplitúdójú tartományban a zaj hatása jelentős lehet, ezért kis felbontást alkalmaznak; a nagy

amplitúdójú tartományban pedig a zaj már nem hat olyan torzító hatással, ezért megengedhető a nagyobb felbontás is.



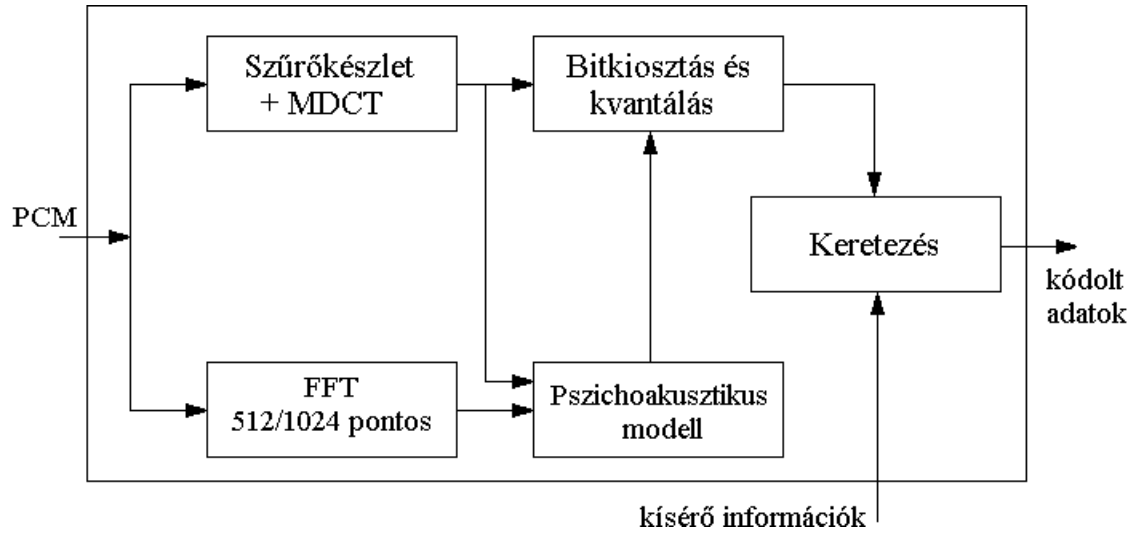
74. ábra



a.) b.)

75. ábra

- MPEG Layer I, Layer II, Layer III
  Az MPEG-1 fejlesztők ajánlásukban három módszert dolgoztak ki a hangtömörítésre. Ezt a három módszert nevezik audio rétegeknek (layer), mivel rétegszerűen, hierarchikusan épülnek egymásra. Megkülönböztetünk MPEG Audio Layer I, Layer II, ill. Layer III kódolási technikákat. Az egymásra épülés úgy valósul meg, hogy a Layer III képes kezelni önmagán kívül az összes alatta lévőt - Layer II-t és Layer I-t -, a Layer II önmagán kívül csak a Layer I-t tudja feldolgozni, míg a Layer I csak önmagát. Mint azt az előzőek is bizonyítják a struktúra felülről kompatibilis.
  A tömörítés előtt rendelkezésünkre állnak a tömörítetlen PCM adatok. Az MPEG meglehetősen bonyolult és igen sok számítást igénylő kódolási eljárással alakítja át ezeket az adatokat tömörített bináris adatokká. A következő ábrán láthatjuk az MPEG Audio-kódolás elvi felépítését (76. ábra).



76. ábra

Első lépésként a bejövő PCM hangokat szűrősorozattal frekvenciák szerint részsávokra (subband) osztjuk, így 32 részsávot hozunk létre. Ezek a részsávok Layer II-nél egyformán 750 Hz szélességűek, míg Layer III-nál a sávok a hangmagasság szerint egyenletesen vannak elosztva. A jobb frekvenciabontás elérése céljából alkalmazhatunk MDCT (Modified Discrete Cosine Transformation - Módosított Diszkrét Cosinus Transzformáció) eljárást, amelyekkel további alsávokra oszjuk a 32 fősávot. Például egy 18 pontos MDCT-t alkalmazva 32*18 azaz 576 frekvencia alsávra tudjuk osztani a spektrumot. Az MDCT előnye, hogy pontosabban szétválaszthatók a lényeges és a lényegtelen hangösszetevők, így ezzel nagyobb hatásfokot érhetünk el a tömörítésben.
A következő lépés a pszichoakusztikus modell alkalmazása, melynek során először mintacsoportokat képzünk, mégpedig úgy, hogy az adatfolyamból veszünk K egymás utáni mintát. A K értéke általában 12 szokott lenni, de Layer III-nál több is lehet (pl.: K=18). Normál esetben tehát 32*12 = 384 minta képez egy adatkeretbe. Ha alsávokat is alkalmazunk, akkor minden alsávban elvégezzük a mintavételt, így 18 pontos MDCT esetén 32*18*12 = 6912 minta fog a rendelkezésünkre állni. Ezek után kiválasztjuk a legnagyobb amplitúdójú mintát (skálafaktor), és finoman kvantáljuk (mintavételezzük). A mintacsoport többi mintáját normalizáljuk (leosztjuk) a maximummal, ennek következtében az így kapott értékek a [-1;+1] tartományba fognak esni, tehát csak a törtrészét kell tárolni, ill. kódolni, az egészrészre nincs szükség. Ezzel az eljárással tehát megtakarítjuk az egészrészek kódolását (a maximális amplitúdójú minta kivételével), ám a hátránya, hogy eközben a kis mintákat nagy torzítással mintavételezzük (a normalizálás miatt). Itt kihasználhatjuk az emberi fül azon a tulajdonságát, hogy bizonyos ideig nem érzékeli az imént említett torzítást. Ezt a „bizonyos időt” nevezzük blokkméretnek. Az előzőekből következik, hogy ha a blokkméretet megfelelően kicsinek választjuk meg, akkor az emberi fül nem fog torzítást érzékelni. A 3. rétegben, mivel három blokk határoz meg egy keretet, lehetőség nyílik a blokkok közötti redundanciák csökkentésére, oly módon, hogy a skálafaktorokat hasonlítjuk össze, és ha nem szükséges, akkor nem tároljuk el mind a három blokkot.

Az időbeli átfedések számításához szükség van a különböző összetevők szétválasztására, amelyet az FFT-vel (Fast Fourier Transform - Gyors Fourier Transzformáció) végzünk el oly módon, hogy a jelet frekvenciaspektrumra transzformáljuk át (2.2.1. fejezet). Az FFT-re azért van szükség, mert az átfedések számolásához szükséges részsávszélesség túl nagy, és az FFT meglehetősen nagy pontossággal tudja szétválasztani a frekvenciákat.
A (11) képletben szereplő exponenciális kifejezés képzetes részt tartalmaz, ezért ezt a formulát a DCT-hez hasonlóan átalakítják valós formátumúra. Az N a transzformáció végrehajtásához felhasznált pontok száma, amely megadja, hogy mennyi egymást követő hangértéken végezzük el műveletet. Ez gyakorlati okokból 2 hatványa szokott lenni. Jellemző pontosság az 512, vagy akár 1024 pontos FFT. A D_n vektorban kapjuk a frekvenciaösszetevőket, méghozzá rendezett formában.
A pszichoakusztikus elemzés után a bitkiosztás következik. Itt vesszük figyelembe a megkívánt kimeneti sebességet, és itt választjuk meg a zajszintet is. Ez utóbbi azt jelenti, hogy addig változtatjuk az összes részsávban a bitszámokat, amíg minden részsávban a kvantálási zaj (bitenként 6 dB), az érzékelhető zajszint alatt marad. A Layer III-nál ezek után következik a Huffman-tömörítés, amelyet előre meghatározott kódtáblákkal végeznek el. A keretezés során az adatfolyamot adatkeretekre bontjuk, így könnyebb lesz a kezelésük a későbbiek során. Végül a keret végére CRC (Cyclic Redundancy Check) ellenőrző összeg kerül, hogy a keletkező hibákat a dekóder kezelni tudja.

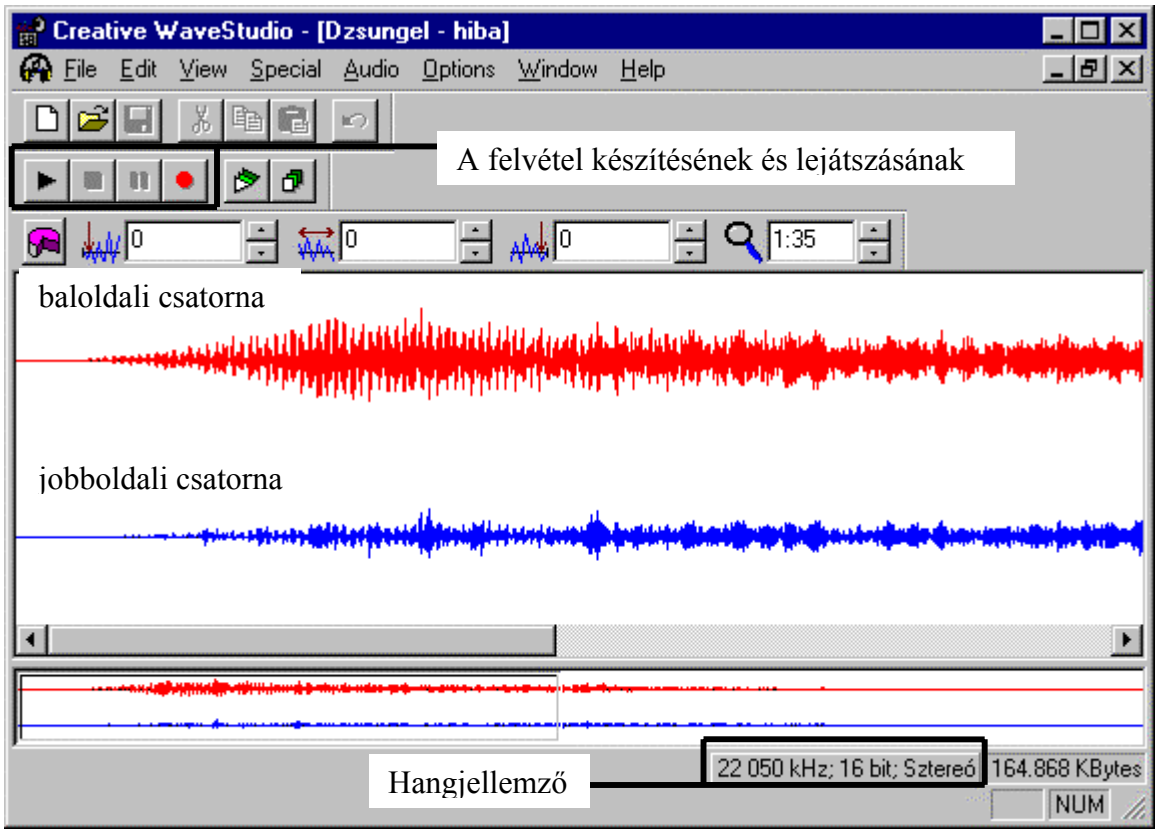
4.5. Digitális audioállományok szerkesztése

*Hangfelvételek készítésé*hez a hangkártyán kívül szükség van egy mikrofonra és szoftverre. Az operációs rendszerhez hozzáadnak egy nagyon egyszerű segédprogramot (Sound Recorder), amellyel a felvételek már elkészíthetőek (77. ábra), és .wav kiterjesztéssel lementhetőek. Ugyanez az eszköz alkalmas a felvételek lejátszására, továbbá a legegyszerűbb szerkesztési műveletek elvégzésére. Ezen kívül minden hangkártyához adnak már olyan segédprogramokat, amelyekkel ezek a műveletek professzionálisan elvégezhetőek. A SoundBlaster kártyákhoz a Soundo'LE (78. ábra) és a Creative WaveStudio (79. ábra) tartozik. Különösen az utóbbit érdemes egy kicsit részletesebben megismernünk, mert mind a három szerkesztési funkcióval felruházták.

77. ábra

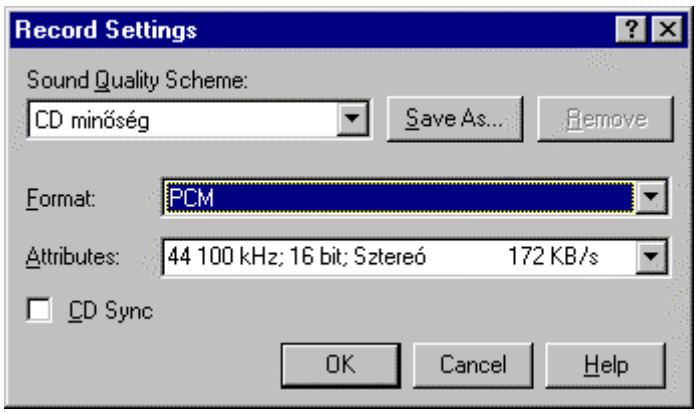
78. ábra

Az ablak alapvetően két részre osztott, az alsó részen látható a teljes hangállomány egy vagy két csatornás hangerősség–idő függvénye, míg a felső részen ennek kinagyított részlete. Ez elősegíti a szerkesztéshez szükséges kijelölést: kijelölés kezdete, hossza és vége.



79. ábra

Digitális hangfelvétel készítéséhez a legfontosabb tulajdonságokat (hangjellemzők [lásd 79. ábra]) a Soundo'LE program Options/recording Settings opciójával állíthatjuk be (80. ábra).



A hangfelvétel minőségének beállítása:

- a hangminőség jellege (CD, rádió, telefonvonal),
- tömörítési eljárások (PCM, ADPCM, A-szabály, Ω-szabály, MPEG Layer-3),
- hangjellemzők (mintavételi frek-vencia, kvantálási szóhossz, csatornaszám,

80. ábra

A *hanghullám szerkesztésé*nek legfontosabb lépései:

- kijelölés (bizonyos részek, minden),

- vágólapra másolás (Copy),
- vágólapra kivágás (Cut),
- kijelölt hangrész törlése, ill. a kijelölt részen kívül minden törlése (Delete, Crop to selection),
- hangrészlet beszúrása a vágólapról (hozzáadás),
- hangrészlet beillesztése a vágólapról az eredetihez történő keveréssel,

Az utolsó két művelet lehetővé teszi egyik hangállományból egy bizonyos részlet átemelése egy másik hangállományba.

A WaveStudio a Special menüpontjában egy csomó hasznos és haszontalan különleges effektust tartalmaz, amelyekkel nagyon sajátos hatásokat adhatunk az eredeti hanghullámhoz:

- a kiválasztott rész beszúrása a kiválasztott résztől jobbra (Rap),
- szünet beszúrása a hangállományba (Insert Silente),
- hangfelvétel lejátszása visszafelé (Reverse),
- visszhang hozzáadása a felvételhez (Echo) [késleltetés – Echo Delay, hangerősség – Magnitude]
- a hangerősség növelése, csökkentése (Fade In/Out),
- a kiválasztott csatorna elnémítása (Mute),
- a kijelölt hangrészlet mozgatása egyik csatornából a másik csatornába (Pan left to right ill. Pan right to left),
- megadott késleltetés beszúrása a kiválasztott csatornába (Phase Shift),
- a hangerősség megváltoztatása csatornánként (Volume).

Befejezésül két témakört érintsünk, amelyeknek még nagy jövője lesz a multimédiában: MIDI, beszédfelismerés.

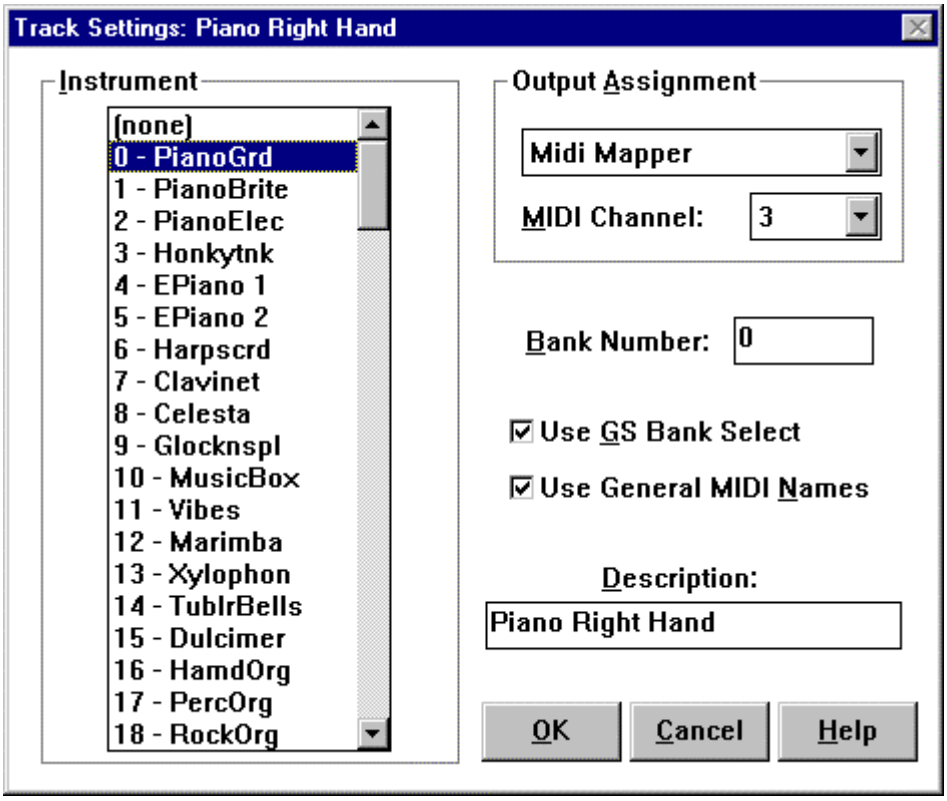
4.6. MIDI

A MIDI (Musical Instrument Digital Interface) egy olyan digitális hangszercsatoló protokoll, amely lehetővé teszi a számítógép és az elektronikus hangszerek közötti szabályozott adatátvitelt. A MIDI 1982 vége óta szabvány, amelynek továbbfejlesztett változata a General MIDI. A MIDI legfontosabb eleme a szintetizátor, amely feladata akusztikus jelek előállítása. Mint láttuk, az lehet egy külső eszköz, de lehet a hangkártyára is integrálva. Minket ez utóbbi eset érdekel, hiszen ennek lehet szerepe zeneállományok készítésében multimédiaalkalmazások részére.

A MIDI 10 oktávon keresztüli kódolást tesz lehetővé, ami 128 féle hangszer megszólaltatását jelenti. Ezek a hangok csatornákhoz rendelhetők, amelyek száma a MIDI szabvány szerint max. 16. Mindegyik hangszerhez egy rájellemző minta (patch) tartozik. A MIDI-ben rögzített hangszerek listáját a 81. ábra tartalmazza.

Ahhoz, hogy a hangkártyán lévő szintetizátor megszólítható legyen, kell egy ún. dallamszerkesztő (sequencer) program. A dallamszerkesztő tulajdonképpen egy elektronikus kottalap. Megadja, hogy az adott csatornához melyik hangszer hangmintája (patch) legyen hozzárendelve, milyen paraméterek mellett. A kottalapon keresztül hangokat szólaltathatunk meg az adott hangszerrel, beállíthatjuk a legfontosabb hangjellemzőket (hangmagasság, ütem, stb.). Ezen paraméterek mindegyike a .mid kiterjesztésű állományban tárolódik. Egy ilyen dallamszerkesztő (Midisoft Recording Session) grafikus felületét látjuk a 82. ábrán. Az ábra alapvetően öt részre osztható:

- a kottalap (1), amely mutatja az egyes sávokat, a hozzájuk rendelt hangszerekkel. Itt lehet szerkeszteni a hangokat, hangjegyeket, megadni az ütemet, stb.,



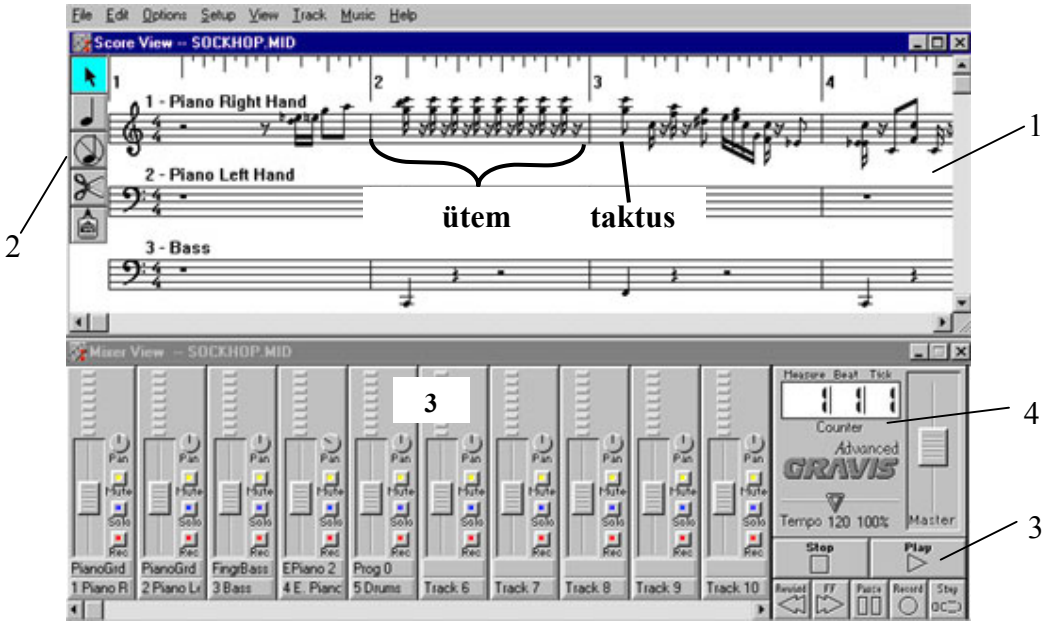
81. ábra

- a kottaszerkesztő eszközök (2), amelyekkel a hangszerfüggő hangjegyek a kottalapon elhelyezhetőek.,
- a csatornaszerkesztő (3), amelyben az egyes sávokhoz hangszerek rendelhetők (81. ábra alapján), továbbá a csatornák tulajdonságai is beállíthatóak,
- a vezérlőblokkban (4) található a lejátszás összes kezelőszerve,
- a mesterblokkban (5) mutatja az az éppen aktuális ütem sorszámát, egy ütemen belüli aktuális taktust, órajelet, a lejátszási sebességet. Egy $\frac{4}{4}$-es ütemet alapul véve a taktus értéke 4 és 1 taktushoz 96 órajel tartozik.

Természetesen nem lehet célom zenei ismereteink felelevenítése, de néhány fogalmat azért érdemes felelevenítenünk. Az ütemmérték azt határozza meg, hogy bizonyos hosszúságú hangokból (egész-, fél-, negyed-, nyolcadhangok) hány van egy ütemen belül. Az ütemek egysége tehát a hangok típusából adódik. A negyedes ütem alapegysége a negyedhang. A $\frac{4}{4}$-es ütemben tehát, 4 db negyedes, vagy 2 db egész, vagy 8 db nyolcados , stb. hang van. Egy $\frac{2}{4}$-es ütemben 2 db negyedes, vagy 1 db egész, vagy 4 db nyolcados, stb. hang van. A Music/Time Signature menüpontban állítható be az ütemérték (83. ábra jobb oldala), míg a Music/Clef menüpontban (83. ábra bal oldala) adható meg a dallam alapja a kottarendszerünkben (violinkulcs, basszuskulcs). Essen néhány szó a félhangokról,

- ezek egyébként a szintetizátor fekete billentyűi – jelölésüket a kottalapon is meg kell tenni:

- félhang emelés (#), C-ből Cisz, D-ből Disz, stb. lesznek,

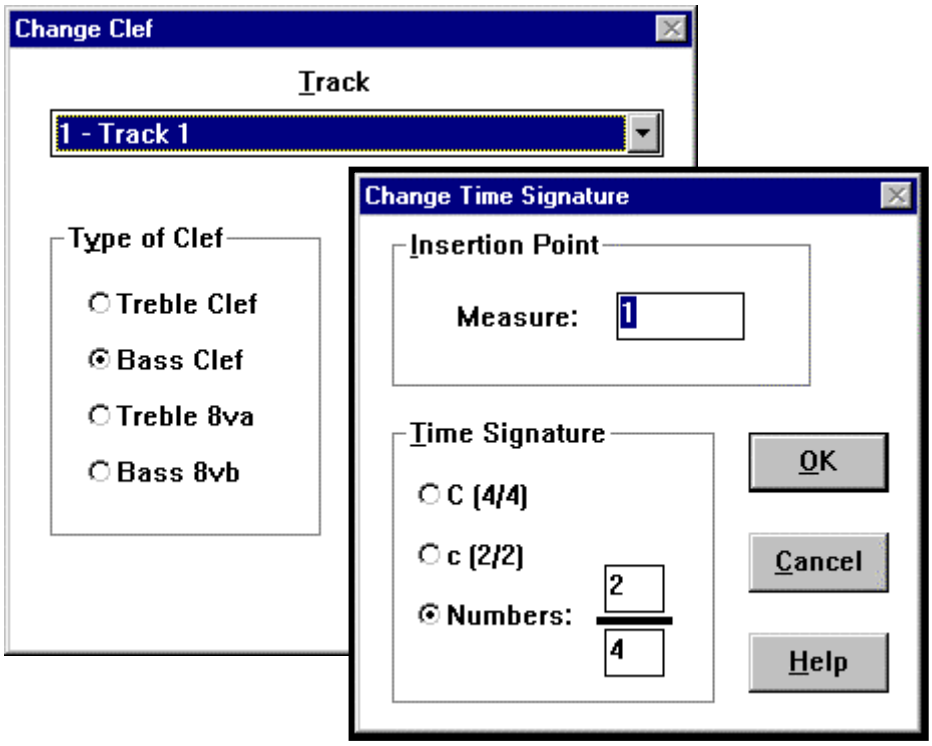


82. ábra

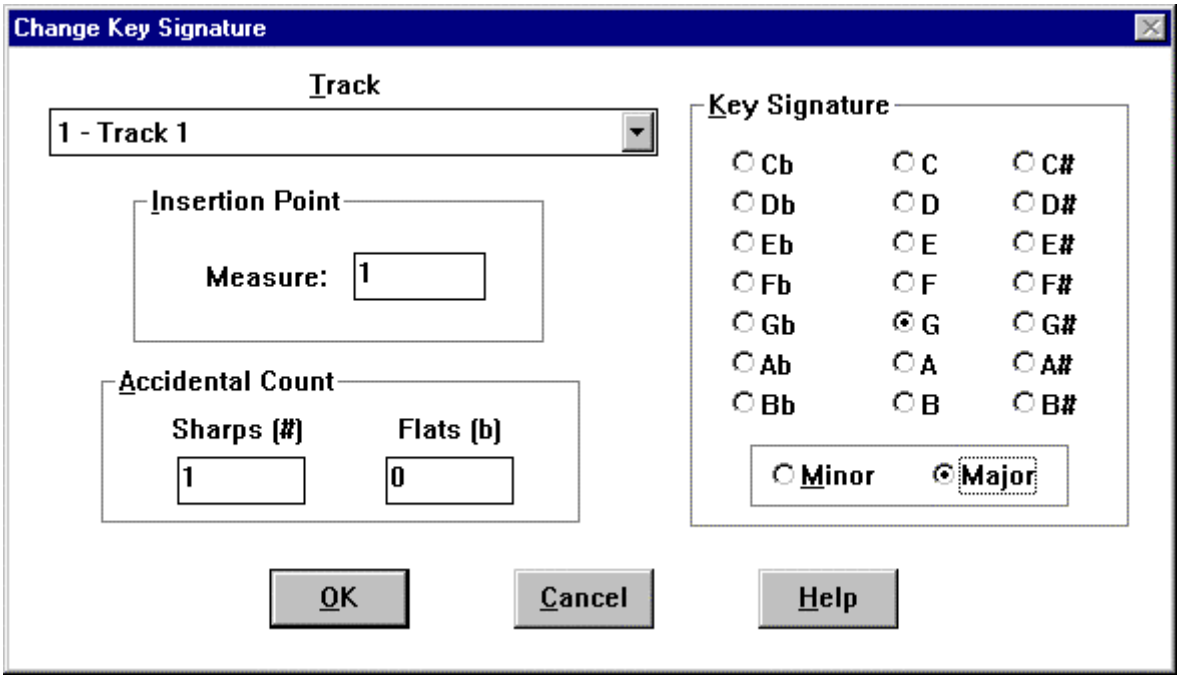
- félhang süllyesztés (♭), A-ból Asz, E-ből Esz, stb. lesznek. (84. ábra)
- A következőkben néhány olyan műveletet sorolnék fel, amelyekkel a csatornák szerkeszthetők (Track menüpontban):
- új csatorna beszúrása,
- csatorna törlése (csak tartalom, tartalom + csatorna),
- csatorna másolása, mozgatása,
- csatornák kombinálása,
- stb.

A 85. ábra azt az ablakot mutatja, amely minden MIDI eseményt feltűntet. Minden eseménynek megvan a saját sora, ezek jelentése a következő (balról jobbra):

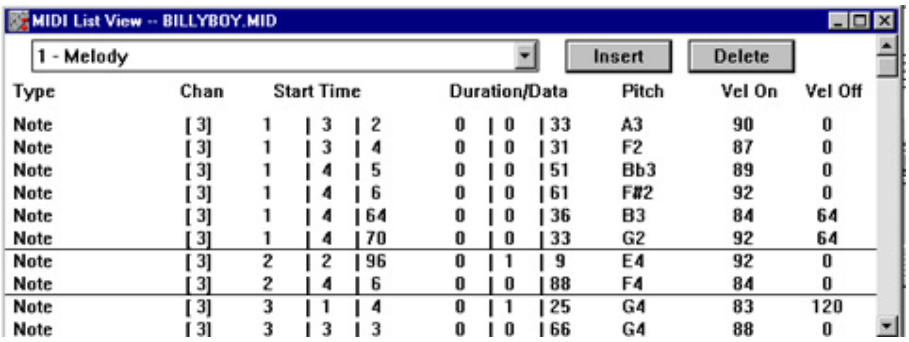
- a bejegyzés típusa (NOTE – hangjegy), ezután a bejegyzések jellemzői következnek, így itt most a hangjegyét említem csak,
- az a csatorna, amelyre a művelet vonatkozik,
- a művelet kezdete és vége (időtartam) [ütem | taktus | órajel], amely megadja, hogy milyen hosszan hallatszik a hang,
- a MIDI billentyűszáma, amely az oktávról és hangjegyről tájékoztat (Pitch),
- sebesség (velocity), amely megadja, hogy milyen gyorsan nyomtunk le egy billentyűt és ezáltal a hang erősségét adhatjuk meg.



83. ábra



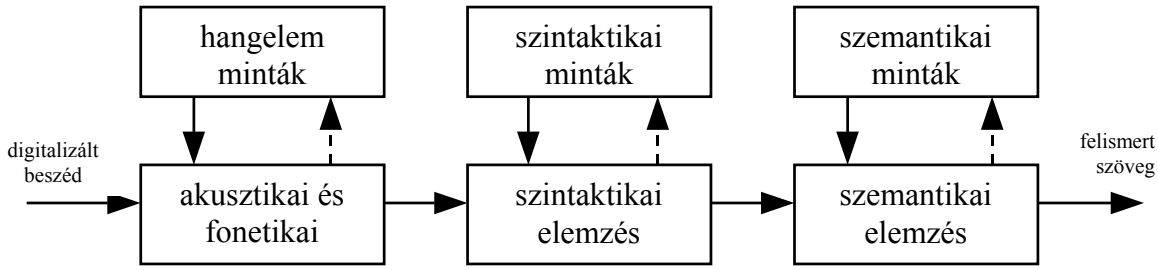
84. ábra



85. ábra

4.7. Beszédfelismerés, beszédszintézis

A részletekbe menő tárgyalás nem lehet célja ennek a fejezetnek, hiszen az a mesterséges intelligencia területére sodornának bennünket. A *beszédfelismerés* egy analóg hangadat digitálisan kódolt adattá történő átalakítását jelenteni. Ezt leírni roppant egyszerű, megvalósítani viszont annál nehezebb, hiszen egy ilyen eljárásnak az emberi gondolkodást, hangképzést kell modelleznie. Gondoljunk csak a hasonló hangzású szavakra (fonjad-fonnyad), vagy a szóban ill. mondatban lévő hangsúlyokra, stb. Egy ilyen eljárásnak elemeznie kell az elhangzott szót, vagy mondatot alkotó beszéd elemeket akusztikai-fonetikai, szintaktikai, szemantikai szempontból, és azt összehasonlítani egy tárolt készlettel. Egy ilyen rendszer elvi felépítését a 86. ábra mutatja. Az ábrán szaggatott vonalak a minták utólagos bővítését jelenti.



86. ábra

Az eljárások egy része a műveleteket az idő-, egy másik része pedig a frekvenciatartományban hajtja végre. Az időtartományban dolgozó eljárások a beszédelemeket időben, mint építőelemeket kapcsolja össze. Ennek az eljárásnak külön problémája az egyes elemek átmeneteinek „elsimítása” (koartikulációja). A frekvenciatartományban dolgozó eljárások a beszédelemeknek megfelelő frekvenciaszűrőket használják az akusztikus adatok feldolgozására. Befolyásolja ezt az eljárást a háttérzaj, amely rászuperponálódik a hanghullámra. Másik probléma, hogy ugyanazok a szavak röviden és elnyújtottan is kiejthetők, ezért ilyenkor célszerű az időbeli vizsgálat. Mindezek alapján, tehát a beszédfelismerés folyamata a következő lépésekből áll:

- analóg akusztikus beszéd digitalizálása,

- idő-, vagy frekvenciatartományba transzformálás (ADPCM, DCT),
- a digitális jel szegmentálása (feldarabolása) beszédelemekre,
- a kapott elemek összehasonlítása – több lépésben – a beszédmintában („szótár") lévő elemekkel (ezt a folyamatot mutatja a 86. ábra),
- esetleg tömörítés.

A beszédfelismerés legneuralgikusabb pontja a beszédelemek megválasztása. Ez természetesen befolyásolja a szegmentálást és a mintával történő összehasonlítását is. A beszédelemek kiválasztása nyelvfüggő. Például az angol nyelvben kevés ún. szóalak (szótő + rag) van, ezek alapján könnyű szótárat készíteni és könnyű a szegmentálás folyamata is. Ezzel ellentétes a magyar nyelv, hiszen nagyon sok féle szóalak létezik. A sok szóalak a sok – egymásra épülő – toldalékból keletkezik. Mit tekintsünk alapelemnek?

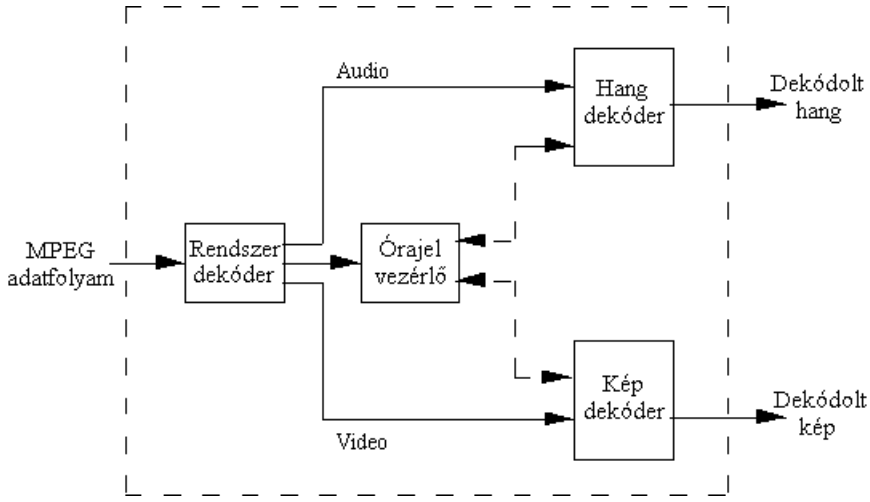
- fonéma (a legkisebb jelentésszerű, de önálló jelentéssel nem bíró beszédegység)
- Viszonylag könnyű felépíteni a „szótárt", de a szomszédos fonémák befolyásolhatják a közöttük lévő fonémát. Például a „multimédia" szó fonémái a következők: „mu ul lt ti im mé éd di ia".
- szó
- Ez a módszer a legegyszerűbb – legalábbis néhány nyelvben -, de a szótő – rag kapcsolat a kiejtésben nehezen jön át. A felismerés folyamatát a 86. ábra mutatja, amely az előzőnél jóval bonyolultabb.
- Markov féle statisztikus módszer
- A beszédfelismeréshez nem a nyelvi egységekből, hanem a matematikai statisztikából indulunk ki. Az akusztikus jelet megadott időközönként (századmásodperces nagyságrend) szegmentáljuk és az így kapott szegmenseket hasonlítjuk össze a „szótárban" lévő szegmensekkel. Ezzel viszonylag pontosan tudjuk leírni az adott beszédet, de a minta akár több milliós is lehet. Mindezek mellett van egy mondatépítő részmodulja is, amely arra ad becslést, hogy az egyes szópárok, szóhármasok milyen gyakorisággal fordulnak elő.
- Chomsky féle generatív grammatika
- N. Chomsky szerint bármilyen nyelv végtelen számú mondata leírható egy véges számú szótár és egy véges számú nyelvtani szabályrendszer segítségével. Ezt az alapelvet felhasználva szófajcsoportokat hoznak létre, amely a beszélőre jellemző és tetszés szerint bővíthető, hiszen az ember szókészlete is bővül. A másik sajátossága a nyelvtani szabályszerűségek felhasználása, például az, hogy főnévből főnévi igenevet képezni ragokkal lehetséges: főnévi igenév = főnév + rag + ni, ahol rag $\in$ [-az, -oz, -ez, -öz, -al, -ol, -el, -öl]. Példák: könyv + el + ni $\rightarrow$ könyvelni; szám + ol + ni $\rightarrow$; vonal + az + ni $\rightarrow$ vonalazni; stb.

A *beszédszintézis* írott szöveg hanggá alakítását jelenti. Alapvetően kétféle eljárással lehet ezt megvalósítani, amelyek elvüket tekintve megegyeznek a beszédfelismerésnél említettek, csak éppen minden fordítva zajlik le:

- fonéma alapú szintézis
  Az így előállított hangállomány kis méretű, a beszédhang hangmagassága könnyen módosítható, de hangzása eléggé „gépiesnek" tűnik.
- Markov féle statisztikus szintézis
  Az így előállított hangállomány nagyobb méretű, a beszédhang hangmagassága nehezen módosítható, de hangzása megegyezik a valódi emberi hangzással.

4.8. Az MPEG-1 rendszerfunkciók

Az MPEG-1 rendszerfunkciók feladata a kódolt videó és audio adatfolyamok egyesítése, összefűzése egyetlen, egységes adatfolyammá és az ezek közötti *szinkronizáció* megvalósítása. A rendszerfunkciók tartalmazzák a szükséges és elégséges információkat a dekódolt kép- és hangrétegek szinkronizációjáról, kezelik és menedzselik a dekódoláshoz szükséges memóriát (adatbufferek), figyelik a túlcsordulást és a kihasználtságot, valamint elvégzik a hibakezeléseket és pozícionálásokat magán az adatfolyamon belül. A rendszer tehát egyfajta multiplexelést végez el a különböző médiatípusok (kép és hang) között. A 87. ábrán megfigyelhetjük, hogyan működik az MPEG rendszer egy dekódolóban. Láthatjuk, hogy a beérkező MPEG adatfolyamot a rendszer dekóder dolgozza fel, itt válogatja szét a hang és kép adatokat valamint küldi a megfelelő dekóderhez. Az adatkeretek szinkronizációját egy egyszerű 90 kHz-es órajelvezérlő végzi, amely elég időt biztosít a különböző feladatok elvégzésére, képes időzíteni az egyéb MPEG lejátszási funkciókat, mint például több adatfolyam egyidejű kezelése, különféle képváltási arányok és mintavételezési frekvenciák, hálózati irányítás, digitális tárolás, stb. Az időzítő nemcsak állandó, hanem változó adatsebességet is tud kezelni, amellyel megvalósítható az optimális sávkihasználás. A rendszer kimenetén jelenik meg a dekódolt hang és kép.[7][17][18]



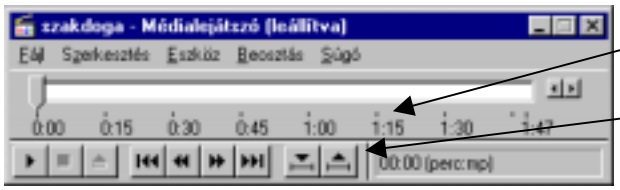
87. ábra

4.9. Digitális video- és audioállományok lejátszása

A video médium lejátszási lehetőségei közül hármat említenék, egyrészt az editáló programba épített lejátszó, másrészt a multimédia keretprogramba épített player, harmadrészt pedig önálló médium lejátszó. Bármelyikről is legyen szó kompatibilisnek kell lennie a videoállomány típusával, ezek szerint az lehet MPEG-payer, AVI-player, ill. hybrid-player, azaz többféle kiterjesztésű állomány lejátszási képessége. Az ActiveMovie a korábban említett AVI video-, MID zene-, FLC, FLI animációállományok lejátszására szolgál (88. ábra), ezen program része a Windows '95 ill. '98 operációs rendszereknek, ezért könnyű hozzáférkőzni.
A QuickTime program alkalmas .mov állományok lejátszására. A 89-90. ábra bemutatja egy animáció két képkockáját, ill. annak kezelőfelületét.

A tömörített audioállományok közül a gyakorlatban az MPEG Layer-3 terjedt el, amely 0.08-0.1 közötti tömörítési arányt valósít meg 56-64 kbit/s –os adatátviteli sebesség mellett. A Layer-3 állományok kiterjesztése .mp3, amelyeket egy speciális programmal

játszhatunk le. Ilyen program (decoder) például a WinPlayer (33. ábra), ill. a WinAmp (34. ábra).



lejátszási idő vagy képkockák száma

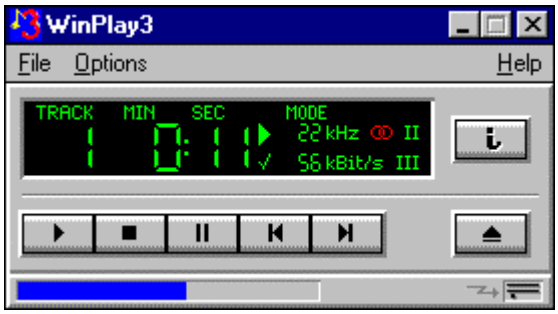
kezelőfelület: balról jobbra haladva - indítás, lejátszás, szünet, az első képkockára ugrás, egy kockával vissza, egy kockával előre, az utolsó képkockára ugrás

88. ábra

kezelőfelület

89. ábra

90. ábra



91. ábra

92. ábra

5. ANIMÁCIÓFELDOLGOZÁS ÉS TÖMÖRÍTÉS

5.1. Digitális animációk jellemzői

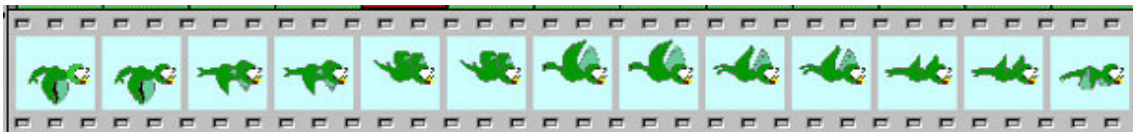
Az első animációk még kézzel készültek és nem kisebb névhez, mint Walt Disney nevéhez kötődnek, a 30-as évek közepe táján. Animációinak fő célja „az életszerűség keltése” volt.

Az animáció átmenetet jelent az állókép és a videó között. A mozgás, a változás érzetét kelthetjük vele, de viszonylag kis tárigényt jelent számítógépünknek. Az állókép szekvencia akkor észlelhető mozgásnak, változásnak, ha a *képismétlési sebesség* meghaladja a 15 képkocka/másodperc értéket [FPS – frame per second]. Az *animáció hosszá*t a (3.34) összefüggés alapján határozzuk meg.

$$\text{animációhossz}[\text{s}] = \frac{\text{képkockaszám}[\text{frame}]}{\text{képismétlésisebesség}[\text{frame}/\text{s}]} \qquad (42)$$

Az animáció lehet két- és háromdimenziós, amelyek alapeleme a képkocka (frame). A képkocka tulajdonképpen egy két- ill. háromdimenziós ábra, amelyek készítésére a 2. fejezetben tanult eszközök szolgálnak. Ezek figyelembevételével a kétdimenziós ábrákat a CorelDraw, míg a háromdimenziósakat a Mechanical Desktop vagy pedig a 3D Studio MAX programokkal készíthetjük el. Az animáció készítésére a Corel Move, ill. Animator Studio (kétdimenziós) és a 3D Studio MAX (háromdimenziós) programok szolgálhatnak. Előbb azonban említsünk meg néhány animáció típust:

- állandó hátterű animáció
  Az egyik legősibb animáció, amelyben úgy keltik a mozgás érzetét, hogy a mozgás különböző fázisait egy-egy képkockára készítik el, majd azokat gyorsan egymásután lejátszák. A mozgás során a frame-ek háttere nem változik (93. ábra).

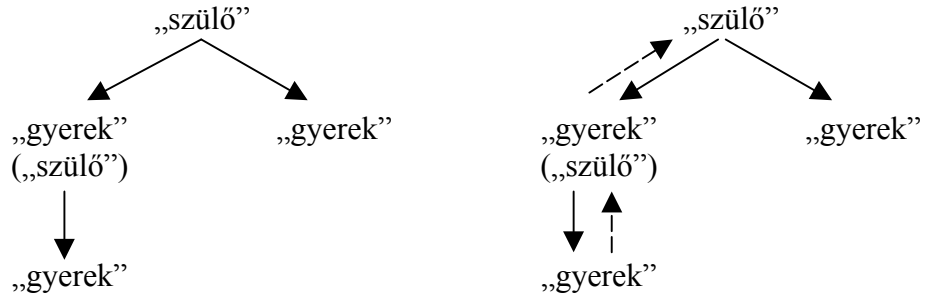


93. ábra

- állandó előterű animáció
  Az előzőnek éppen az volt a hátránya, hogy a mozgás minden fázisát elő kell állítani, amely bizony elég időigényes és komoly szakértelmet igényel. Írhattam volna művészetet is. Itt az előtér állandó és a hátteret változtatva érjük el a mozgás érzetét.
- megadott útvonal menti animáció
  Az objektum egy előre definiált útvonal mentén történő elmozdítása, a program képes a pálya matematikai leírására.
- fázisanimáció
  A definiált két szélső állapot közötti fázisokat a program határozza meg. A két szélső állapotot tartalmazó képkockát kulcsnak (keyframe) nevezzük. A keyframe-ben nem szükséges minden objektum minden tulajdonságát rögzíteni, sokszor csak egy-két tulajdonságát tároljuk, míg a többi paramétert az előző és következő kulcskockákból számított értékek lesznek. Az is előfordulhat, hogy egyik objektum kulcskockája egy másik objektum közbülső kockája is lehet.
- alakváltozás animálása
  A mozgás animáció helyett az objektum alakja változik meg. Ennek szemléletes módja lehet egy emberi grimasz, vagy két golyó ütközése.
- fényhatások, kamera mozgás animálása
  Az objektumokra ható fényviszonyok (lámpa típusok, fényerőssége, színe, stb.) valamint a kameramozgások növelhetik a tárgy animáció hatásait. A kamera vagy a lámpa körbejárhat egy álló objektumot, amely a mozgás látszatát keltheti, hiszen tudjuk

a mozgás relatív, csak a szemlélőtől függ, hogy a fák szaladnak az álló vonathoz képest, vagy pedig a fa mellet állok és a vonat robog el mellettem.

- anyagjellemzők animálása
  Az objektumok anyaga színhatásként jelenik meg, amelyik ha megváltozik, azt a hatást keltik, hogy megváltozik az anyagminőség.
- morfózis
  Az egyik objektumnak a másikba történő átalakulását morfózisnak nevezzük, amely lehet teljes és részleges módosulás.
- bemozdulásos életlenség animálása
  Amikor a kamera előtt nagy sebességgel elmozdul egy objektum, akkor körvonalai nem látszhatnak tökéletesen. Az élvonalak elmosódása, életlensége, átmenete növeli a mozgás dinamikáját.
- dinamikai animáció
  Az animációs paraméterek beállításakor a legnehezebb feladat az egymással kölcsönhatásban lévő objektumok kapcsolatának modellezése, a különböző pályamódosulások, sebességváltozások, stb. meghatározása. Ezt hidalja át ez a lehetőség. A dinamikai animáció nem valós időben történik, hanem a program kiszámítja az értékeket, és a jelenetben résztvevő objektumokhoz egy-egy kulcsot rendel hozzá. Az objektumok deformációit – például ütközéskor – az objektumoknál beállított arányszám alapján lehet meghatározni. Ez az arányszám megadja, hogy az objektum mozgási energiájának hányadrésze marad meg. Az objektumok energiahányadosa mellett megadható a súrlódási tényező (tapadási, csúszási), a tömeg, a sűrűség, stb. valamint külső hatások (pl. külső erők, szél, stb.), erőterek (pl. gravitáció), stb. is.
- kinematikai animáció
  Az egyes objektumok közötti kapcsolat lehet előreható és visszaható (94. ábra), ezen kapcsolatok leírását hierarchikus kapcsolatnak nevezzük. Az előreható kinematikánál (a) a „szülőobjektum” valamilyen animációja (elmozdulás, elfordulás, stb.) maga után vonja a „gyerekobjektum(ok)” animációját is, viszont fordítva ez nem igaz. Az inverzkinematikánál (b) a „gyerek” visszahat a „szülőre”, de a „szülő” korlátozhatja. Egyes „gyerekobjektumok” lehetnek „szülők” is. A 94. ábrán a szaggatott vonal jelöli azokat a kapcsolatokat, amelyek korlátosak.



a) b)

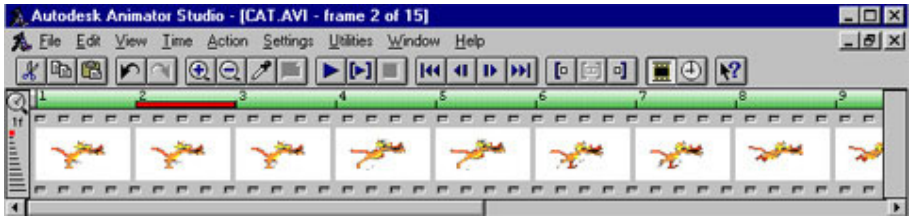
94. ábra

5.2. Kétdimenziós animációk tervezése

A kétdimenziós animációk szerkesztési eszközeit ismerjük meg az Autodesk Animator Studio által. A program négy komponenst tartalmaz:

- az Animátor alkalmas digitális animációk készítésére állóképek, ill. már korábban készített digitális animációk beszúrásával,
- a SoundLab program által lehetőségünk van Windows alapú hangállományok (WAVE) készítésére, szerkesztésére,
- a Scriptor felhasználásával a különböző digitális időfüggő médiumok összeszerkesztésére nyílik mód,
- a Player segítségével lejátszhatjuk a digitális videó és animáció állományokat.

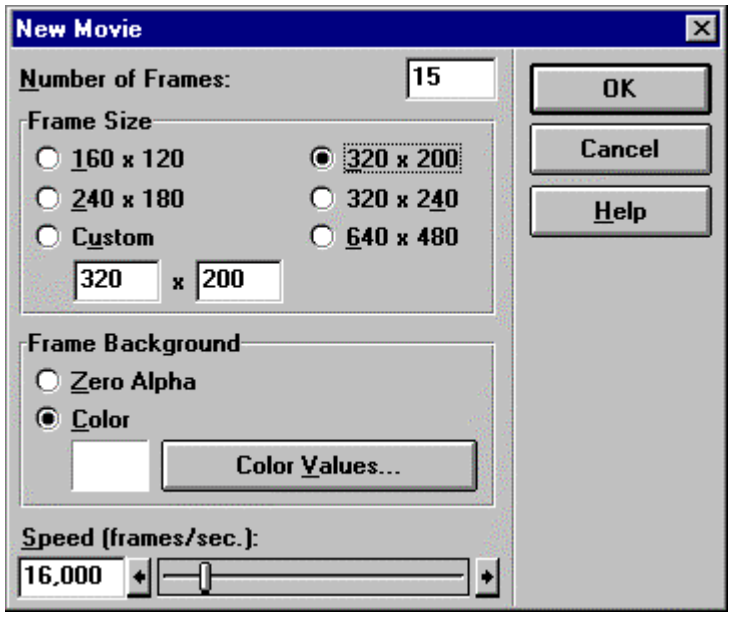
Az Animátor program grafikus felületét a 95. ábra mutatja. Természetesen nem lehet célom a program teljes megismertetése, inkább csak az általános érvényű ismeretekre koncentrálok. A grafikus felület négy részre osztható: menüszerkezet, képkocka sor, ill. az ábrán nem látható szerkesztő felület valamint a 96. ábrán látható eszköztár.



95. ábra

Az új animáció készítésekor (File New) a 97. ábrán látható ablak jelenik meg, ahol beállítható a képkocka mérete képpontban (Frame Size), az animáció hosszát képkockában (Nunber of Frames), a képkockák háttérszíne (Frame Background) és a képváltási sebesség (Speed [frame/sec]).

96. ábra



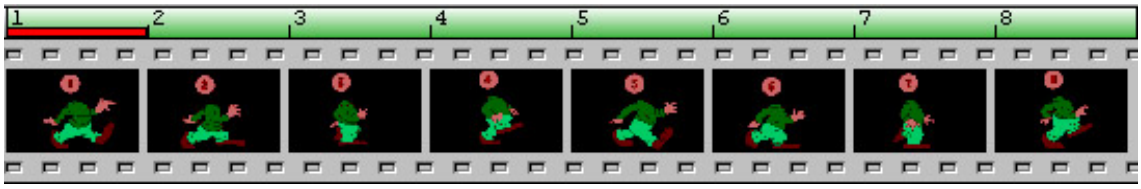
97. ábra

Az eszköztár felhasználásával elkészíthetjük az egyes képkockákat, ezek jórészt megegyeznek az Adobe Photoshop-nál, ill. a Corel Draw-nál megismert képszerkesztő, rajzoló eszközökkel. Egyedül a második sor jobbszélén lévő Sprite ikont emelném csak ki, ui. ez alkalmas a különböző médiumok beszúrására (állóképek-BMP, GIF, JPG; animáció, videó-MPG, FLC, FLI).

A rajzolási és szerkesztési műveleteket végrehajthatjuk Frame vagy pedig Time módban. Az első esetben az egyes műveletek csak az aktuális képkockára, míg a második esetben a teljes időszegmensre lesznek érvényesek. Frame módban az első és a második kulcs-képkockát külön-külön tudom szerkeszteni, addig Time módban ezeken túl egyidejűleg is szerkeszthetőek. *Kulcs-képkocká*t bárhol elhelyezhetünk az animációnkban, persze célszerű olyan pozícióba tennünk, ahol lényeges változások következnek be. A digitális animáció készítés során a programok a kulcskockák közötti képkockákat interpolálással határozzák meg. A kulcs-képkockák képezik a veszteséges tömörítés alapját is, ezt korábban már a deltakeretek fogalmának tisztázásakor is láttuk. Ezek azok a képkockák, amelyeket MPEG algoritmusú tömörítéskor I képeknek tekintünk.

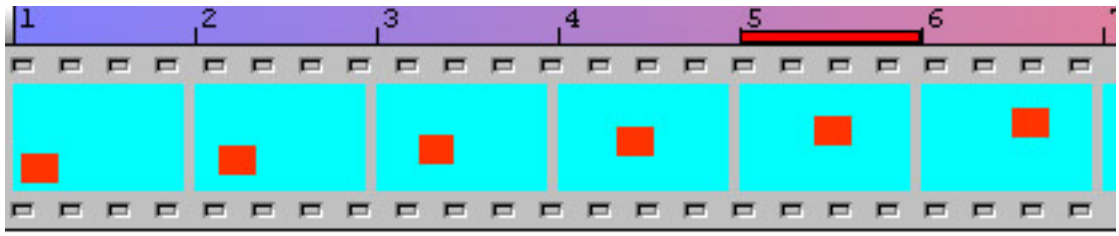
A program háromféle animáció készítést támogat:

- cella típusú animáció
  Ide tartoznak a kockánként elkészített „rajzfilm” animációk. Amennyiben az egymást követő képkockákon a változásokat apró mozzanatonként dolgozzuk ki, úgy azokat lejátszva azt a hatást kelti, mintha a mozgás, ill. a változás folytonos lenne. Erre látunk nyolc kockából álló példát a 98. ábrán. A képkockák mindegyike egy önállóan készített állókép, a program lehetővé teszi azok importálását, animációvá történő összerendelését. Amennyiben jól választom meg a képkockák számát és a képváltási sebességet akkor figura mozgása folyamatosnak tűnik.



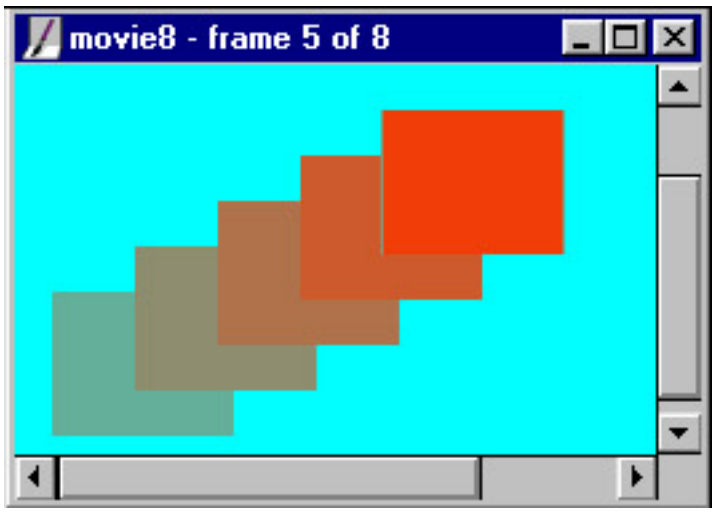
98. ábra

- „automatikus” animáció
  Az automatikus jelző azt jelenti, hogy az animáció készítőjének csak a kulcskockákat kell elkészítenie a közöttük lévő többi képkockát a program automatikusan interpolálja (közelítve határozza meg). A kulcs-képkockák között lehetnek szín-, méret-, helyzet- és alakváltozások. A 99. ábrán lévő téglalap két szélső helyzetét (kulcs-képkockák) adtam meg, ezután már csak az elmozdulás útvonalát kellett definiálni.



99. ábra

Az Action menüpontban lévő szerkesztési műveletekkel az objektum elfordulását, torzulását, stb. lehet megadni. Fontos, hogy ezeket a műveleteket a Time módban hajtsuk végre és a Float opció is be legyen kapcsolva, mert így tudom az egyes kulcs-képkocka tartalmakat egymástól függetlenül szerkeszteni (100. ábra). A közbenső képkockák tartalmát a Render művelet végrehajtásával határozza meg.



100. ábra

- digitális videó

Ez a műveletsor megegyezik az Adobe Premiere-nél megismert eljárásokkal. A szerkesztés végeredménye AVI és FLC kiterjesztésű állomány.

101. ábra

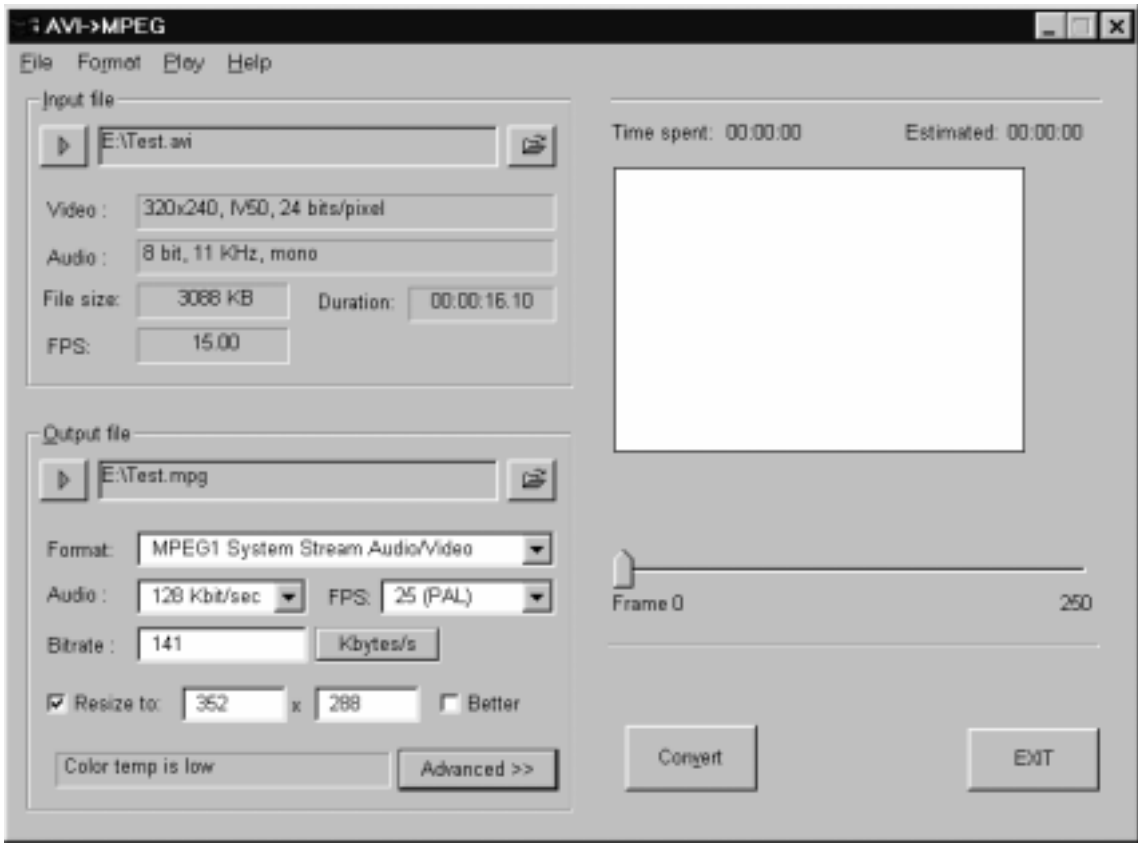
Befejezésül a 101. ábra mutatja a Scriptor grafikus felületét, amely segítségével a digitális animáció- és videóállományok szerkeszthetők össze az audioállományokkal.

5.3. AVI-MPEG konvertálás

A video és animációs állományok kimenete igen gyakran a nem igazán jó tömörítési hatásfokú AVI (Audio-Video Interleave), ezért olyan programot alkalmazunk, amely az AVI állományt átkonvertálja MPEG-gé. Ilyen például a DVMPEG programcsomag AVI2MPEG Conventor-a.

A program bemenete lehet akár tömörített vagy tömörítetlen formátumú AVI videójel, de WAV típusú hangállományt is lehet vele tömöríteni. A program a forrásfile-ból MPEG-1, ill. MPEG-2 formátumú állományt tud készíteni. Természetesen az MPEG kódoláshoz szükséges legfontosabb paramétereket mi magunk is beállíthatjuk.

A program elindítása után megjelenik a főablak (102. ábra), amely a legfontosabb információkat tartalmazza a forrás ill. a cél állományról. Itt tudjuk a kódolásra legjellemzőbb paramétereket beállítani.

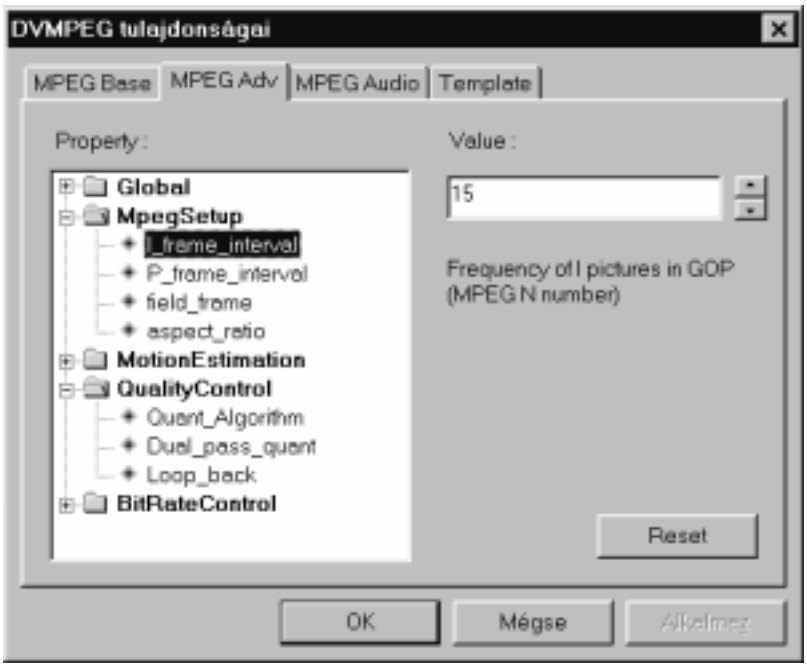


102. ábra

Az ablak bal felső sarka a bemeneti fájl adatait mutatja: az állomány mérete, a teljes hossza, a képváltási sebessége, továbbá az audio és videó adatfolyam paramétereit (képfelbontás, hangfelbontás, mintavételi frekvencia, színmélység, stb.). Természetesen itt lehet megadni a forrásfájl nevét is. A bal alsó részen adhatjuk meg a kimeneti állomány legfontosabb paramétereit: az MPEG fájl neve, formátuma, az audio átviteli sebessége , a

video képváltási sebessége (FPS), a videóadatfolyam átviteli sebessége (Bitrate) és szükség esetén a képkockák átméretezése (Resize to). Az ablak jobb oldalán jeleníthetjük meg a forrásvideót a képkockák számával együtt, míg alul találjuk a [*Convert*] gombot, amelyet lenyomva elindul a beállított paraméterek szerinti tömörítés.

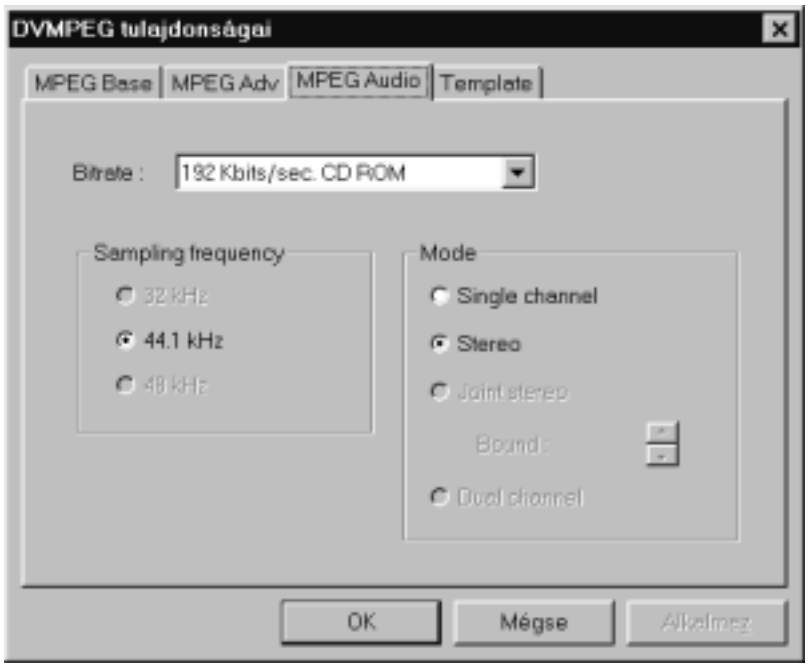
A kimenő fájl beállítási részén, alul található az [*Advanced>>*] gomb, amelyet megnyomva beállíthatjuk az MPEG tömörítés többi paraméterét is. A megjelenő ablaknak több „nyuszi füle" van, az első oldalon az (*MPEG Base*) adatfolyam típusát, a képszabványt (PAL, NTSC) és az átviteli sebességet állíthatjuk be (MPEG-1, MPEG-2, Video Only, stb.).



103. ábra

Az *MPEG Adv*. „nyuszi fülnél" (103. ábra) az első csoportban (Global) az általános beállítási lehetőségek szerepelnek, mint például a minőségi faktor (Quality factor), amely tulajdonképpen a kvantálással van összefüggésben. Értékét a Value sorban lehet beállítani. A második csoportban (MpegSetup) vannak a GOP-okkal kapcsolatos beállítások, mint például az I képek frekvenciája (I frame interval), a P képek frekvenciája (P frame interval) vagy, hogy átlapolt legyenek-e a képkockák avagy sem (Field frame). A harmadik csoport (MotionEstimation) a mozgáskompenzáció paramétereit tartalmazza, mint a vízszintes keresési tartomány (motion width), a függőleges keresési tartomány (motion height) és beállítható a mozgáskereső algoritmus is (motion algorithm). A következő csoportban (QualityControl) a minőséget meghatározó paraméterek találhatók: a kvantálási algoritmus (Quant Algorithm) - egyenletes vagy előre beállított legyen -, a kvantálás fokozatosan finomodó legyen-e vagy sem (Loop back). Az utolsó csoportban (Bitrate Control) a különböző képtípusok átviteli arányát lehet megadni %-ban.

A következő „nyuszi fülön" (*MPEG Audio*) a hangkódolással kapcsolatos információkat tudjuk meghatározni (104. ábra). Itt adhatjuk meg a hangadatok átviteli arányát (Bitrate), a mintavételezési frekvenciát (Sampling frequency) és a csatornák típusát: mono (single channel), sztereo, kapcsolt sztereo (Joint stereo) vagy duál-mono (Dual channel).



104. ábra

A paraméterek megfelelő beállítása után nincs más hátra, csak meg kell nyomni a [*Convert*] gombot és végrehajtódik a tömörítés.

A program egyszerűen kezelhető és előnye, hogy a felhasználó tetszőlegesen adhatja meg a legfontosabb MPEG kódolási paramétereket.

Befejezésül szeretném megköszönni lektoraimnak azt az áldozatos munkát, amelynek hiányában ez a munka nem kerülhetett volna az olvasó kezébe.

IRODALOMJEGYZÉK

[1] Tóth Péter: Multimédia I. Időfüggő médiumok, Ligatura, Budapest, 1999
[2] Ralf Steinmetz: Multimédia Becezetés és alapok, Springer, Budapest, 1995
[3] Deke McClelland: Photoshop 5 Biblia I-II. kötet, Kiskapu Kiadó, Budapest, 1999
[4] Budai Attila: A számítógépes grafika, LSI Oktatóközpont, Budapest, 1999
[5] Barna Tamás: Videotechnika a gyakorlatban, Műszaki Könyvkiadó, Budapest, 1988
[6] Flag '98 Coder-1 előadás, Veszteséges kódolás introkban, 1998
http://rs1.szif.hu/~tomcat/konf/vesztkod/vesztkod.htm
[7] The MPEG Standard, MPEG FAQ, 1998
http://www.crs4.it/~luigi/MPEG/mpegfaq1.html
[8] Gaia Multimédia Stúdió - Fórizs István: MPEG, 1998-2000
http://www.tiszanet.hu/Gaia/mpeg.htm
[9] Video Compression, 1996
http://www.cs.sfu.ca/undergrad/CourseMaterials/CMPT479/material/notes/Chap4/Chap4.2/Chap4.2.html
[10] John Wiseman: An introduction to MPEG video compression,
http://members.aol.com/symbandgr/
[11] Videa - Interactive Video and Multimedia, Digital Video,
http://www.ividea.com/dvmpeg.htm
[12] Intel Support, Glossary of Common Video Terms, 2000.
http://www.intel.hu/support/videocapture/8006.htm
[13] Woobin Lee: Motion compensation, 1995.
http://icsl.ee.washington.edu/~woobin/papers/General/node5.html
[14] Tektronix MBD, Compression in Video, 1997
http://www.tek.com/Measurement/App_Notes/mpegfund/sect2.html
[15] The implementation of the 2D-DCT
http://rnvs.informatik.tu-chemnitz.de/~ja/MPEG/HTML/IDCT.html
[16] Sarkadi Csaba, ATM és MPEG - A jövő alakulása, Új alaplap 1997/6 szám.
[17] Leonardo Chiariglione, Short MPEG-1 description, 1996.
http://drogo.cselt.stet.it/mpeg/standards/mpeg-1/mpeg-1.htm
[18] Rico Dreier - Niels Rump, MPEG Systems FAQ, Version 7.0a, 1998.
http://drogo.cselt.stet.it/mpeg/faq/faq-systems.htm
[19] Nádaskuti Ákos: Adobe Premiere ComputerBooks, Budapest, 1999.
[20] Tóth Dezső: Multimédia mikroszámítógépes környezetben LSI Kiadó, Budapest, 1997
[21] Csuth László: MPEG tömörítés. Szakdolgozat. BDMF szakdolgozat
[22] Csuth László: LEARN MPEG. Multimédia alapú oktatóprogram.