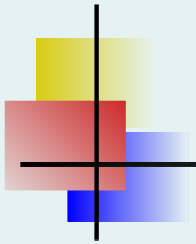


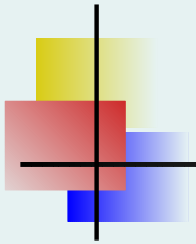
Licence

Image et Son

Pierre Hanna

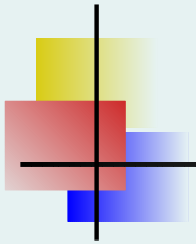
hanna@labri.fr

Université de Bordeaux



Notation musicale

- Manière la plus satisfaisante de transmettre de la musique: **écoute**.
- Plus compliqué : présence de l'interprète nécessaire.
- Notation musicale: transmettre une oeuvre pour qu'elle puisse être jouée/interprétée.
- Langage d'écriture musicale évolué, largement reconnu sur le plan international.
- Répertoire immense disponible sous cette forme dans les librairies musicales.
- Possibilité d'écrire votre musique de manière à la faire jouer par d'autres.
- Partition : une ou plusieurs pages sur lesquelles la musique est écrite.



Formats musicaux

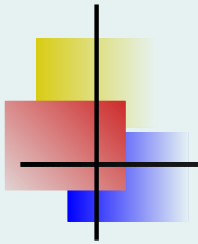
Au lieu de se placer au niveau de l'onde sonore :

- pression acoustique en fonction du temps

niveau musical : évènements musicaux (partition)

- notes
- durée des notes

Format principal : **MIDI** (Musical Instrument Digital Interface)



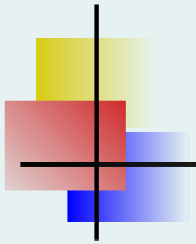
Bibliographie (1/2)

- documents officiels:

  International MIDI Association, 5316 West 57th Street, Los Angeles, CA 90056, USA

  - *International MIDI Association (IMA)*
    MIDI 1.0 Detailed Specification
    Los Angeles, 1988.
  - *International MIDI Association (IMA)*
    General MIDI System
    Los Angeles, 1991.
  - *MIDI Manufacturers Association (MMA)*
    The Complete MIDI 1.0 Detailed Specification – Standard MIDI Files 1.1
    1996.

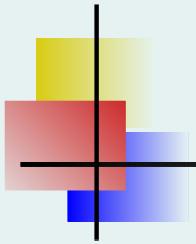
- documents en ligne:
  - `http://www.midi.org` (officiels)
  - `http://www.borg.com/~jglatt/tech/midispec/table.htm`
  - `http://www.harmony-central.com/MIDI`
- très nombreux livres sur le sujet...



Bibliographie (2/2)

quelques critiques:

- F. Richard Moore
  The Dysfunctions of MIDI
  - *Proceedings of the International Computer Music Conference*
    Computer Music Association, San Francisco, 1987;
  - *Computer Music Journal*
    12(1):19–28, *Spring* 1988.
- C. Muir et K. McMillen
  What's Missing in MIDI?
  Guitar Player, *June* 1986.
- . . .
- A. Roberts
  Devices for Increasing the Number of MIDI Channels
  Computer Music Journal
  16(4):101–104, *Winter* 1992.

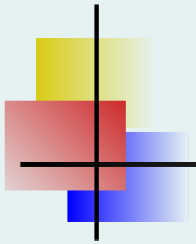


Qu'est-ce que MIDI?

MIDI: *Musical Instrument Digital Interface*

(marque déposée par la MMA: *MIDI Manufacturers Association*)

- protocole de communication entre instruments numériques
- événements musicaux (gestes de l'instrumentiste)
- pas de flux audio-numérique (absence de son)
- pas une véritable norme, mais un standard de marché

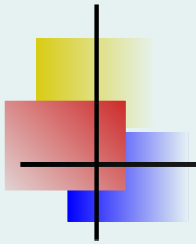


Historique

origine: 1983, *NAMM (North American Music Manufacturers) Show* à Los Angeles (liaison de 2 synthétiseurs de marques différentes pour qu'un instrumentiste puisse jouer le même son avec des timbres différents depuis un unique clavier)

- années 80,
  - première moitié: définition du protocole MIDI;
  - seconde moitié:
    - publication du standard,
    - premières critiques;
- années 90,
  - première moitié: améliorations,
    mais sans toucher au standard original,
  - seconde moitié: grosses critiques;
- depuis: pas de gros changements...

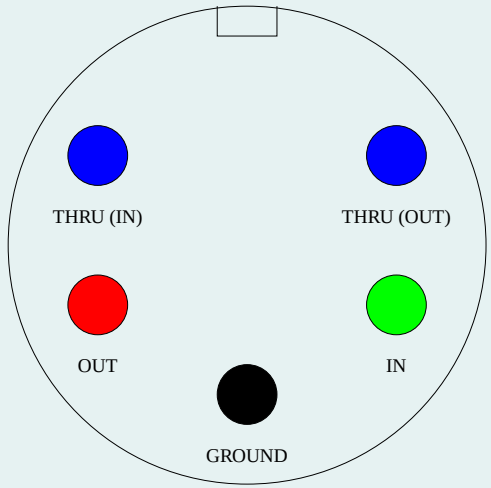
$\rightarrow$ astuces ingénieuses, mais sans réelle remise en question



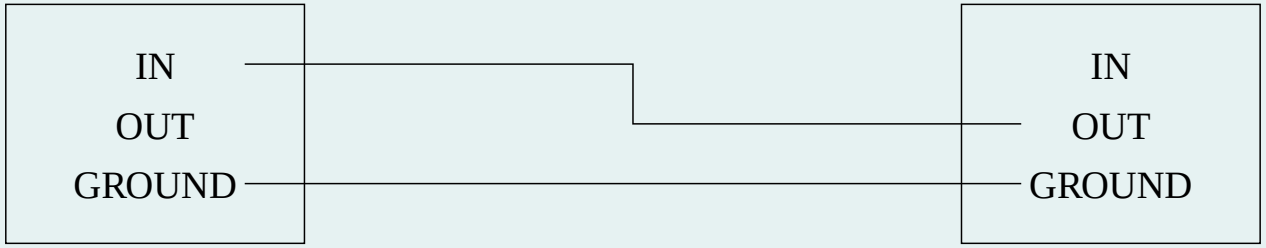
Caractéristiques techniques

- liaison série (type RS 232)
- vitesse de transmission: 31250 bauds
  - 31250 bits par seconde
  - un message MIDI usuel:
    3 octets (1 commande, 2 arguments)
  - un octet $\rightarrow$ 8 bits de données + 2 bits *start* / *stop* (série)
  - $\Rightarrow$ un message MIDI transmis en $(3 \times 10)/31250 \leq 1$ ms
- protocole asynchrone, unidirectionnel
- fonctionnement maître / esclave

Prises MIDI (DIN)

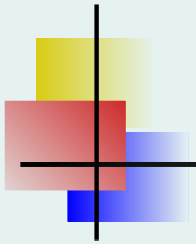


Câble MIDI minimal…



(unidirectionnel: IN $\rightarrow$ OUT ou OUT $\rightarrow$ IN)

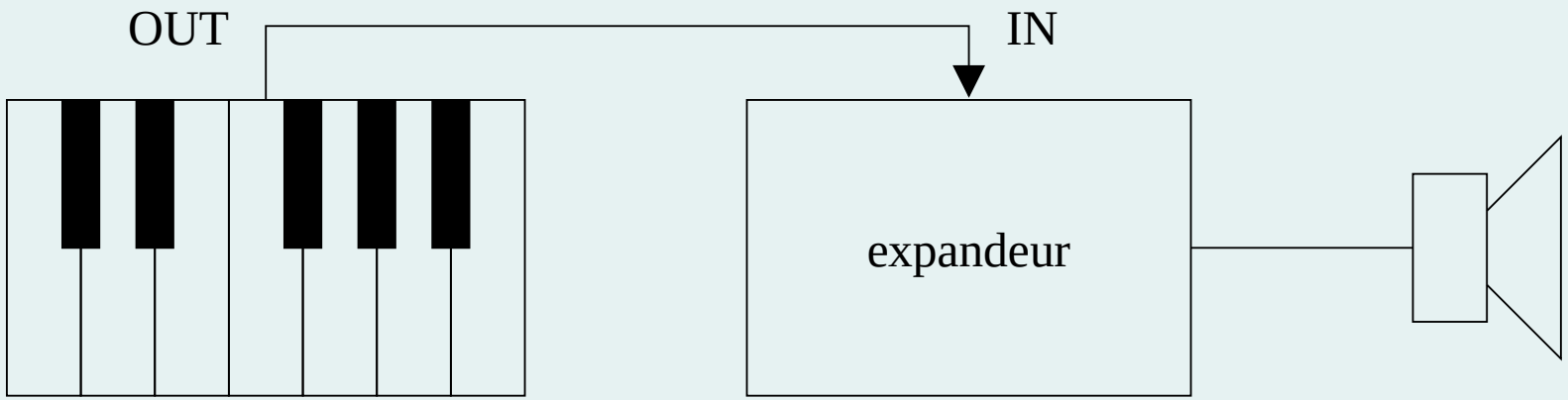
La fonctionnalité THRU (*through*: “à travers”):
renvoie en sortie ce qui provient de l’entrée IN.



Périphériques MIDI

- clavier maître: envoie les événements musicaux
- expandeur: synthétise les sons
- synthétiseur (= clavier maître + expandeur)
- séquenceur: stocke les événements musicaux
- contrôleur matériel: envoi de valeurs
- …

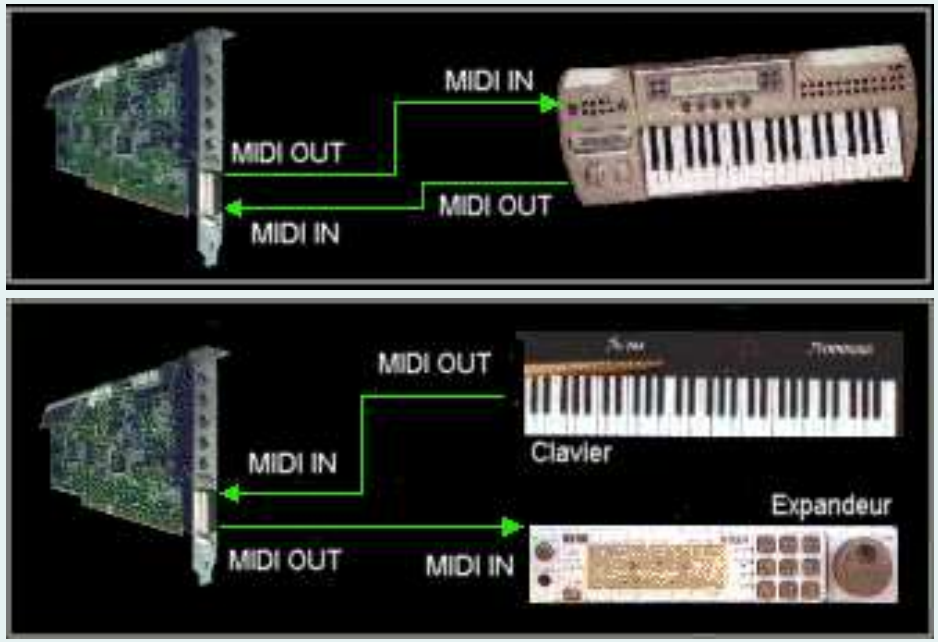
Exemples de réseaux MIDI (1/3)



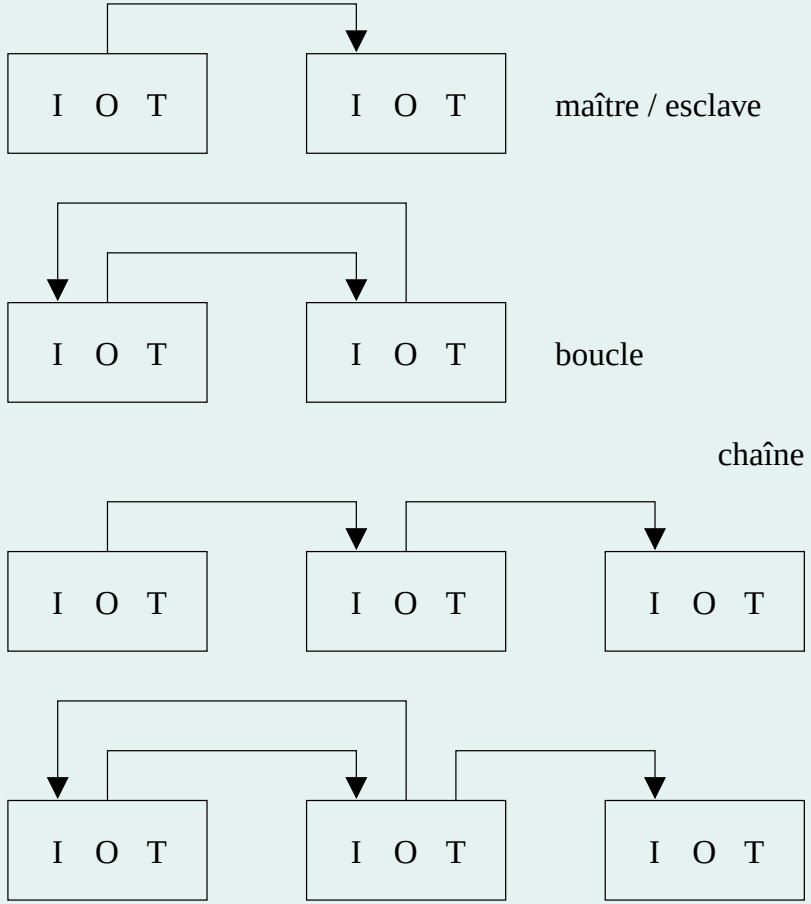
geste (MIDI)

son (non MIDI)

Exemples de réseaux MIDI (2/3)



Exemples de réseaux MIDI (3/3)



Messages normaux (de voix / canal)

channel message: `1xxx cccc`

- `1000cccc 0kkkkkkk 0ppppppp`: note off (touche *k*, vélocité *p*)
- `1001cccc 0kkkkkkk 0ppppppp`: note on (touche *k*, vélocité *p* – note OFF si $p = 0$)
- `1010cccc 0kkkkkkk 0ppppppp`: after touch (touche *k*, pression *p*)
- `1011cccc 0ppppppp 0vvvvvvv`: control change (valeur *v* affectée au paramètre *p*)
- `1100cccc 0nnnnnnn` : program change (numéro *n*)
- `1101cccc 0ppppppp` : channel pressure (pression *p*)
- `1110cccc 0vvvvvvv 0vvvvvvv`: pitch wheel (valeur *v* [14 bits, little-endian])

$$0v_6v_5v_4v_3v_2v_1v_0 \quad 0v_{13}v_{12}v_{11}v_{10}v_9v_8v_7$$

Notes et notations

- notations des 7 notes de la gamme:

français	la	si	do	ré	mi	fa	sol
anglo-saxon	A	B	C	D	E	F	G

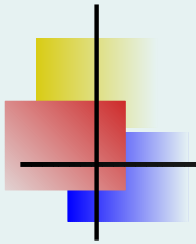
(touches blanches du piano)

- plus les dièses (#),
  au nombre de 5 dans la gamme chromatique

(touches noires du piano)

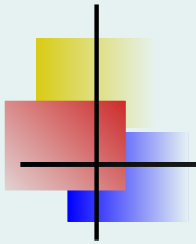
- une octave = 12 demi-tons

(7 + 5 notes)



Code des touches

code	note											
	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
-2 | -1	–	1	2	3	4	5	6	7	8	9	10	11
-1 | 0	12	13	14	15	16	17	18	19	20	21	22	23
0 | 1	24	25	26	27	28	29	30	31	32	33	34	35
1 | 2	36	37	38	39	40	41	42	43	44	45	46	47
2 | 3	48	49	50	51	52	53	54	55	56	57	58	59
3 | 4	60	61	62	63	64	65	66	67	68	69	70	71
4 | 5	72	73	74	75	76	77	78	79	80	81	82	83
5 | 6	84	85	86	87	88	89	90	91	92	93	94	95
6 | 7	96	97	98	99	100	101	102	103	104	105	106	107
7 | 8	108	109	110	111	112	113	114	115	116	117	118	119
8	120	121	122	123	124	125	126	127				
octave												



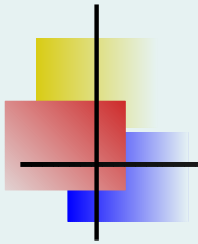
Un exemple. . .

en hexadécimal:

```
0x90 0x3C 0x5F 0x80 0x3C 0x00 0x90 0x3E 0x64 0x80 0x3E

0x00 0x90 0x40 0x63 0x80 0x40 0x00 0x90 0x3C 0x47 0x90

0x3C 0x00 0xFE 0x3E 0x7F 0x3E 0x00
```

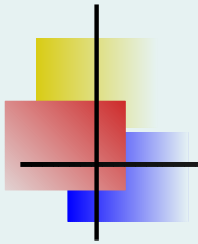


Un exemple. . .

en hexadécimal:

```
0x90 0x3C 0x5F 0x80 0x3C 0x00 0x90 0x3E 0x64 0x80 0x3E
0x00 0x90 0x40 0x63 0x80 0x40 0x00 0x90 0x3C 0x47 0x90
0x3C 0x00 0xFE 0x3E 0x7F 0x3E 0x00
```

`0x90` = `10010000` = *note on* canal 0



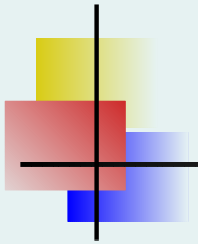
Un exemple. . .

en hexadécimal:

```
0x90 0x3C 0x5F 0x80 0x3C 0x00 0x90 0x3E 0x64 0x80 0x3E
0x00 0x90 0x40 0x63 0x80 0x40 0x00 0x90 0x3C 0x47 0x90
0x3C 0x00 0xFE 0x3E 0x7F 0x3E 0x00
```

`0x90` = `10010000` = *note on* canal 0 $\rightarrow$ prend 2 arguments:

1. numéro de touche: `0x3C` = 60 $\rightarrow$ do
2. vélocité: `0x5F` = 95 $\rightarrow$ forte



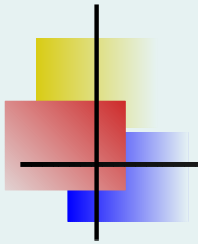
Un exemple. . .

en hexadécimal:

```
0x90 0x3C 0x5F 0x80 0x3C 0x00 0x90 0x3E 0x64 0x80 0x3E
0x00 0x90 0x40 0x63 0x80 0x40 0x00 0x90 0x3C 0x47 0x90
0x3C 0x00 0xFE 0x3E 0x7F 0x3E 0x00
```

- do joué

`0x80` = `10000000` = *note off* canal 0



Un exemple. . .

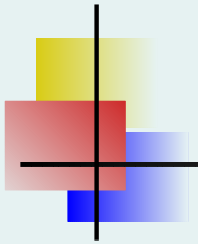
en hexadécimal:

```
0x90 0x3C 0x5F 0x80 0x3C 0x00 0x90 0x3E 0x64 0x80 0x3E
0x00 0x90 0x40 0x63 0x80 0x40 0x00 0x90 0x3C 0x47 0x90
0x3C 0x00 0xFE 0x3E 0x7F 0x3E 0x00
```

- do joué

`0x80` = `10000000` = *note off* canal 0 $\rightarrow$ prend 2 arguments:

1. numéro de touche: `0x3C` = 60 $\rightarrow$ do
2. vélocité: `0x00` = 0 $\rightarrow$ non spécifiée



Un exemple. . .

en hexadécimal:

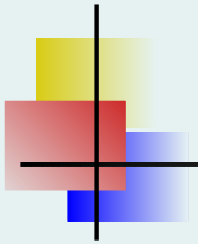
```
0x90 0x3C 0x5F 0x80 0x3C 0x00 0x90 0x3E 0x64 0x80 0x3E
0x00 0x90 0x40 0x63 0x80 0x40 0x00 0x90 0x3C 0x47 0x90
0x3C 0x00 0xFE 0x3E 0x7F 0x3E 0x00
```

- do joué, puis relâché

ensuite:

- ré joué, puis relâché
- mi joué, puis relâché
- do joué, puis relâché*

  ∗ (**astuce**: *note on* avec vélocité 0 $\Rightarrow$ *note off*)



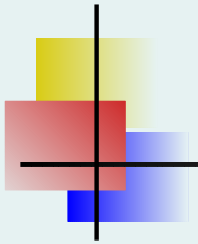
Un exemple. . .

en hexadécimal:

```
0x90 0x3C 0x5F 0x80 0x3C 0x00 0x90 0x3E 0x64 0x80 0x3E
0x00 0x90 0x40 0x63 0x80 0x40 0x00 0x90 0x3C 0x47 0x90
0x3C 0x00 0xFE 0x3E 0x7F 0x3E 0x00
```

- do joué, puis relâché
- ré joué, puis relâché
- mi joué, puis relâché
- do joué, puis relâché

active sensing $\rightarrow$ ne prend pas de paramètre



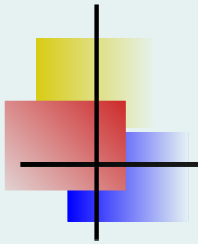
Un exemple. . .

en hexadécimal:

```
0x90 0x3C 0x5F 0x80 0x3C 0x00 0x90 0x3E 0x64 0x80 0x3E
0x00 0x90 0x40 0x63 0x80 0x40 0x00 0x90 0x3C 0x47 0x90
0x3C 0x00 0xFE 0x3E 0x7F 0x3E 0x00
```

- do joué, puis relâché
- ré joué, puis relâché
- mi joué, puis relâché
- do joué, puis relâché

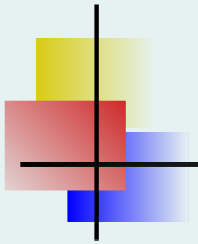
octet de donnée, alors qu'on attend une commande !



État courant: “running status”

- pour économiser l’octet de commande,
- le dernier octet de commande est mémorisé
  (à l’exception des messages système temps-réel,
  qui ne prennent pas de paramètres),
- si un octet de donnée est trouvé alors qu’un octet de commande était attendu, alors le “running status” est utilisé pour la commande.

Dans le cas de l’exemple: *running status* = `0x90`



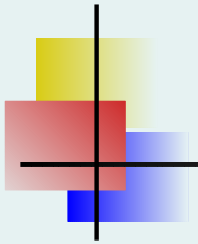
Un exemple. . .

en hexadécimal:

```
0x90 0x3C 0x5F 0x80 0x3C 0x00 0x90 0x3E 0x64 0x80 0x3E
0x00 0x90 0x40 0x63 0x80 0x40 0x00 0x90 0x3C 0x47 0x90
0x3C 0x00 0xFE 0x3E 0x7F 0x3E 0x00
```

- do joué, puis relâché
- ré joué, puis relâché
- mi joué, puis relâché
- do joué, puis relâché

note on (ré, vélocité 127)



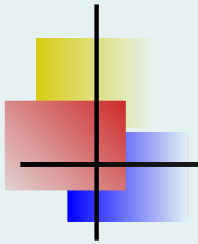
Un exemple. . .

en hexadécimal:

```
0x90 0x3C 0x5F 0x80 0x3C 0x00 0x90 0x3E 0x64 0x80 0x3E
0x00 0x90 0x40 0x63 0x80 0x40 0x00 0x90 0x3C 0x47 0x90
0x3C 0x00 0xFE 0x3E 0x7F 0x3E 0x00
```

- do joué, puis relâché
- ré joué, puis relâché
- mi joué, puis relâché
- do joué, puis relâché
- ré joué

note **off** (ré, vélocité non spécifiée)



Un exemple. . .

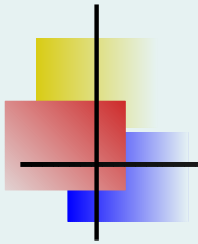
en hexadécimal:

```
0x90 0x3C 0x5F 0x80 0x3C 0x00 0x90 0x3E 0x64 0x80 0x3E
0x00 0x90 0x40 0x63 0x80 0x40 0x00 0x90 0x3C 0x47 0x90
0x3C 0x00 0xFE 0x3E 0x7F 0x3E 0x00
```

interprétation:

- do joué, puis relâché
- ré joué, puis relâché
- mi joué, puis relâché
- do joué, puis relâché
- ré joué, puis relâché

début d'une comptine connue. . .



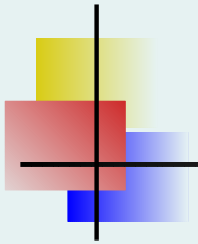
Extension General MIDI (GM)

- publiée en 1991 (*cf.* bibliographie)
- normalise:
  - les numéros des instruments,
  - les numéros des contrôleurs,
  - les identifiants des constructeurs.

Remarque: *General Standard* (GS)

est une extension non standard!

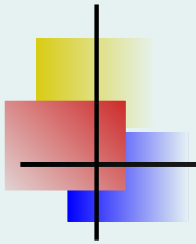
(décision unilatérale d'un constructeur dissident. . .)



Instruments

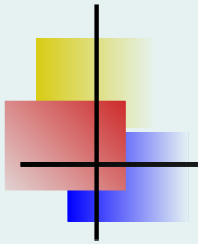
0	acoustic piano	1	brite piano	2	synth piano	3	honky
4	electric piano 1	5	electric piano 2	6	harpischord	7	clavinet
8	celeste	9	glockenspiel	10	music box	11	vibes
12	marimba	13	xylophone	14	tubebell	15	santur
16	home organ	17	percussive organ	18	rock organ	19	church
20	reed organ	21	accordion	22	harmonica	23	concertna
24	nylon guitar	25	acoustic guitar	26	jazz guitar	27	clean guitar
28	mute guitar	29	odd guitar	30	distorted guitar	31	guitar harm
112	carillon	113	agogo	114	steel drum	115	wood block
116	taiko	117	toms	118	synth tom	119	reverse cymbal
120	fx-fret	121	fx-blow	122	seashore	123	jungle
124	telephone	125	helicopter	126	applause	127	pistol

$\Rightarrow$ un canal (numéro 10) reservé à la batterie *(drums)*



Format de fichiers MIDI (Standard MIDI Files)

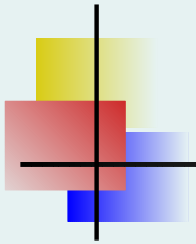
- Différent des formats de fichiers son type WAV
  1min = 10 Mo (WAV) = 1 Mo (MP3)
- Taille importante : limites pour des communications temps-réel entre instruments ou entre utilisateurs.
- Au lieu de se placer au niveau du **son** (microscopique), au niveau de la **musique** (macroscopique).
- format musical: pas la musique elle-même, mais uniquement une description des actions des musiciens



Ecoute de fichiers MIDI

- Pas d'écoute directe d'un fichier MIDI
- Besoin d'un **synthétiseur**
  - interprétation des ordres d'un fichier musical
  - exemple: un synthétiseur génère une onde correspondant à un *do* de violon.
  - la qualité du synthétiseur influence la qualité du son produit.

Synthétiseur MIDI souvent inclus dans la carte son

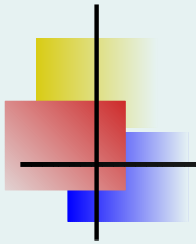


Lecture de fichiers MIDI

Sous Linux, le programme `timidity` [a]

- `timidity -h`
- générer un fichier son (`wav`) à partir d'un fichier MIDI, il suffit d'employer l'option `-Ow`:
  `timidity -Ow` *`fichier`*`.mid`

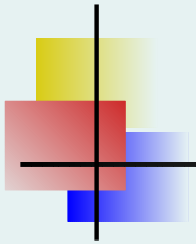
[a] `http://timidity.sourceforge.net/`



Format MIDI: intérêt

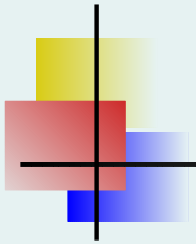
Les fichiers MIDI sont très souvent employés sur internet.

- jouer et transmettre de la musique (site web)
- contrôler un instrument (claviers MIDI, instruments MIDI, . . .)
- représenter un morceau (partition)



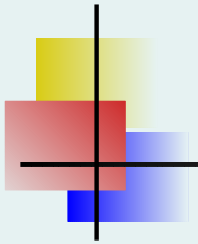
Format MIDI: avantages

- très léger, quelques k-octets pour plusieurs minutes de musique, idéal pour Internet.
- réutilisable et modifiable : chaque note étant clairement identifiée, elle peut être modifiée facilement.
- aucune dégradation lors de la transmission.



Format MIDI: inconvénients

- Réinterprétation du morceau
- Interprétation propre à chaque synthétiseur
- Qualité dépend du synthétiseur
  - Qualité très moyenne
  - bien moins bonne qu'un mp3. . .

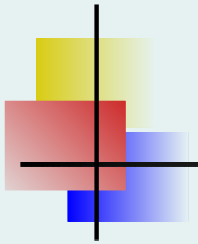


Fichiers MIDI (Standard MIDI Files)

- 3 formats:
  - format 0: piste unique pouvant contenir plusieurs canaux
  - format 1: plusieurs pistes à jouer simultanément
  - format 2: plusieurs pistes à jouer séquentiellement
- structure commune:
  - un en-tête *(header)* `MTHd`
  - des pistes *(tracks)* `MTrk`

Remarque: même principe de base que le format IFF,

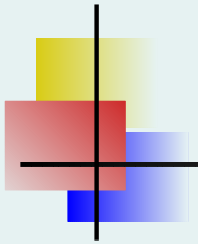
avec des entiers stockés également en *big-endian*.



MThd: *en-tête*

- MThd (4 octets)
- taille (32 bits)
- format (16 bits)
- nombre de pistes (16 bits)
  (+ “*master track*” pour les formats > 0)
- division (16 bits)
  en tops par temps (noire = quart de ronde)
  Exemple:
  - tempo de 125 temps par minute
  - division de 96 tops par temps

  $\Rightarrow$ durée du top d’horloge: 60 / (125 * 96) = 5 ms



`MTrk`: *piste*

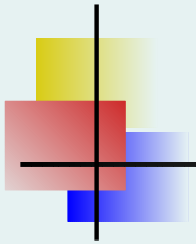
- `MTrk` (4 octets)
- taille (32 bits)
- suite d'événements datés: *date status data . . . data*

Date stockée relativement à la précédente, représentation VLQ *(Variable-Length Quantity)*:

- la représentation binaire est découpée en blocs de 7 bits
- tous les blocs sont placés dans des octets dont le bit de poids fort est positionné à 1, sauf pour le dernier bloc (pour marquer la fin).

Exemples:

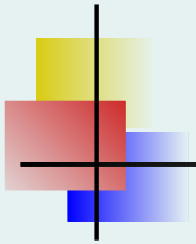
- codage de la valeur 50:
  - codage binaire: `00110010`
  - soit en hexadécimal: `0x32`
- codage de la valeur 32768:
  - codage binaire (blocs de 8 bits): `10000000 00000000`
  - codage binaire (blocs de 7 bits): `0000010 0000000 0000000`
  - rajout des bits de poids fort: `10000010 10000000 00000000`
  - soit en hexadécimal: `0x82 0x80 0x00`
- décodage de la représentation: `0x81 0x02`
  - codage binaire: `10000001 00000010`
  - retrait des bits de poids fort: `(000000)10000010`
  - valeur décimale: $2^7 + 2^1 = 128 + 2 = 130$



Les méta-événements

- meta event: `0xFF`
- séquence complète: `0xFF` code taille $\underbrace{\text{donnée} \cdots \text{donnée}}_{\text{taille octets}}$
- principaux codes (et nombre d'octets de données):

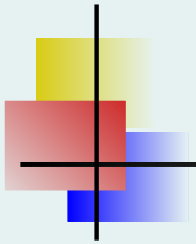
`0x00`	sequence number	(2)
`0x01`	text event	(*)
`0x02`	copyright notice	(*)
`0x03`	sequence / track name	(*)
`0x04`	instrument name	(*)
`0x05`	**lyric**	(*)
`0x06`	marker	(*)
`0x07`	cue point	(*)
...		
`0x20`	MIDI channel prefix	(1)
`0x2F`	**end of track**	(0)
`0x51`	set tempo	(3)
`0x54`	SMPTE offset	(5)
`0x58`	time signature	(4)
`0x59`	key signature	(2)
`0x7F`	sequencer specific	(*)



Un exemple. . .

`test.mid`:

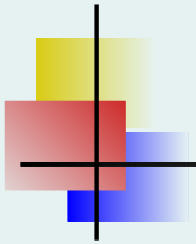
```
0x4D 0x54 0x68 0x64                                  MThd
0x00 0x00 0x00 0x06                                  taille 6
0x00 0x01                                            format 1
0x00 0x03                                            3 pistes
0x00 0x60                                            division: 96
0x4D 0x54 0x72 0x6B                                  MTrk
0x00 0x00 0x00 0x0B                                  taille 11
0x00 0xFF 0x51 0x03 0x07 0xA1 0x20                   0, set tempo
0x00 0xFF 0x2F 0x00                                  0, end of track
0x4D 0x54 0x72 0x6B                                  MTrk
0x00 0x00 0x00 0x48                                  taille 72
0x00 0xB0 0x07 0x0C                                  0, control change: main volume 12
0x00 0xC0 0x01                                       0, program change 1 (piano)
0x00 0x90 0x3C 0x3F                                  note on...
0x60 0x43 0x3F
0x60 0x3C 0x00
00 3E 3F 60 43 00 60 3E 00 00 40 3F 81 40 40 00 00 41 3F 81 40 41
00 00 43 3F 81 40 43 00 00 45 3F 81 40 45 00 00 47 3F 81 40 47 00
00 48 3F 83 00 48 00 00 FF 2F 00
```



Bibliothèque midifile

- Lecture fichier
- Ecriture fichier

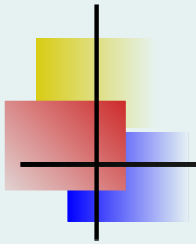
```
#include "midifile.h"
```



Bibliothèque midifile

- Ouverture du fichier MIDI

```
void read_midi(char * midifile)
{
  ...
  MidiFile_t md = MidiFile_load(midifile);
```



Bibliothèque midifile

- Parcours des evenements

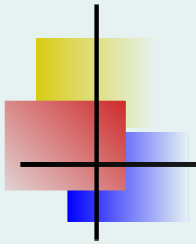
```
event = MidiFile_getFirstEvent(md);

/* boucle principale */
while(event)
  {
    ...

    event = MidiFileEvent_getNextEventInFile(event);
  }
```

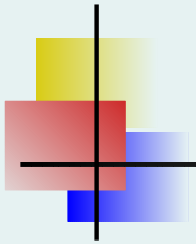
Bibliothèque midifile

```
      if (!MidiFileEvent_isVoiceEvent(event))
{
        }
      /* tempo */
      else if (MidiFileEvent_isTempoEvent(event))
{
        }
      else
{
  if (MidiFileEvent_isNoteStartEvent(event))
    {
          }
        }
```



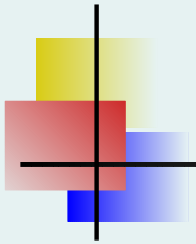
Bibliothèque midifile

```
if (!MidiFileEvent_isVoiceEvent(event))
{
  /* ----- */
  /* key signature */
  if (MidiFileMetaEvent_getNumber(event) == 0x59)
    {
      int key, mode;
      data = MidiFileMetaEvent_getData(event);
      key = (int)(data[0]- '0');
      mode = (int)(data[1] - '0');
      /*
if (mode == 0) "major"
else "minor"
      */
    }
```



Bibliothèque midifile

```
            /* ----- */
  /* time signature */
  if (MidiFileMetaEvent_getNumber(event) == 0x58)
    {
      data = MidiFileMetaEvent_getData(event);
      // Time Signature : (int)(data[0]-'0');
    }
}
```



Bibliothèque midifile

```
if (MidiFileEvent_isNoteStartEvent(event))
{
/* note start */

  // Channel
  c=MidiFileNoteStartEvent_getChannel(event)
  // Time
  t=MidiFileEvent_getTick(event)
  // Pitch
  printf("Note jouee : %d\n", MidiFileNoteStartEvent_getNote(event
  // Duration
  d=MidiFileEvent_getTick(MidiFileNoteStartEvent_getNoteEndEvent(e
  -MidiFileEvent_getTick(event);

  // Attention channel 10 !!
      }
```