

情報の表現と データ構造

文字・画像・音を、0 と 1 で表す

2026 年 9 月

Olivia Chen Toyama · C2LAB

LECTURE 2

02

1	0	1	1	0	0	1	0
0	1	0	0	0	0	0	1
1	1	1	0	0	0	1	1
1	0	0	0	0	0	1	0

INPUT

CPU

MEMORY

STORAGE

OUTPUT

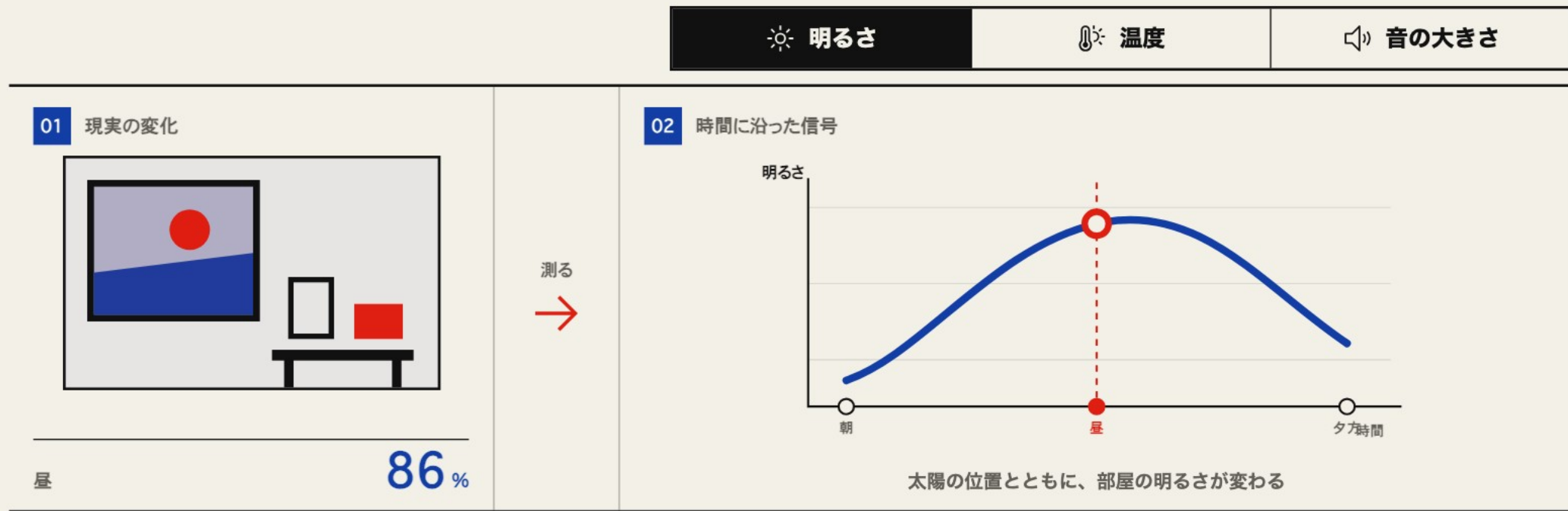
同じ二つの記号が、
数にも、文字にも、音にもなる

コンピュータの中を流れるデータ



部品が違ってても、**受け渡すものはデータである**

現実の情報は、連続して変化する



現実の変化



測定した値

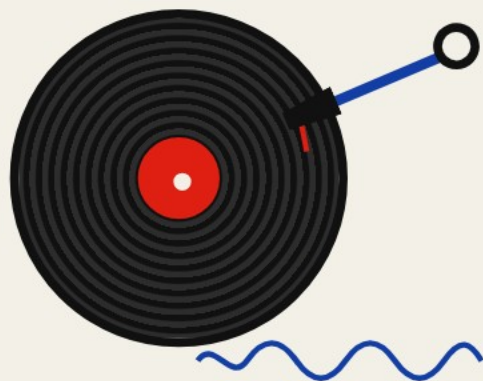


途中の値もすべて存在するため、線はなめらかにつながる

変化の形を、そのまま記録する

01 レコード

音



音の振動 → 溝の起伏

02 カセットテープ

音



音の振動 → 磁化の変化

03 写真フィルム

光



光の強さ → フィルムの濃淡

ANALOG RECORDING

0 / 1 に置き換えず、連続した変化を物理的な形として残す

連続した形を写すと誤差も引き継ぐ



ORIGINAL

■ 原版の連続した形から始める

▶ 1 回目を写す

説明用の模式例。ずれの量や形は、媒体・装置・コピー方法によって異なる。

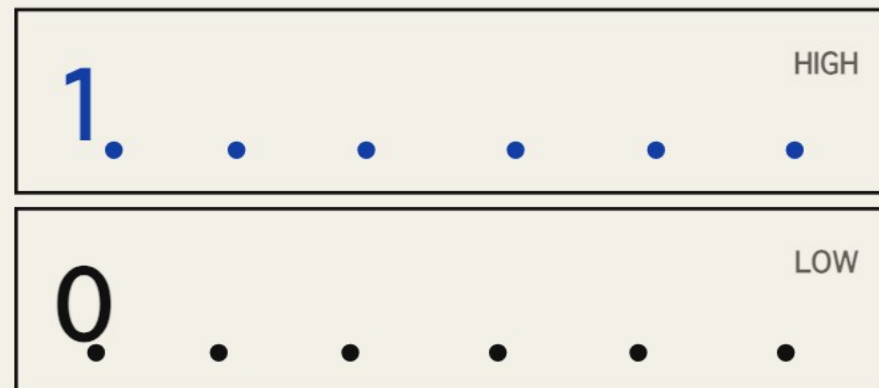
どうすれば、途中の乱れを取り除ける？

現実の変化



値は、なめらかに変わる

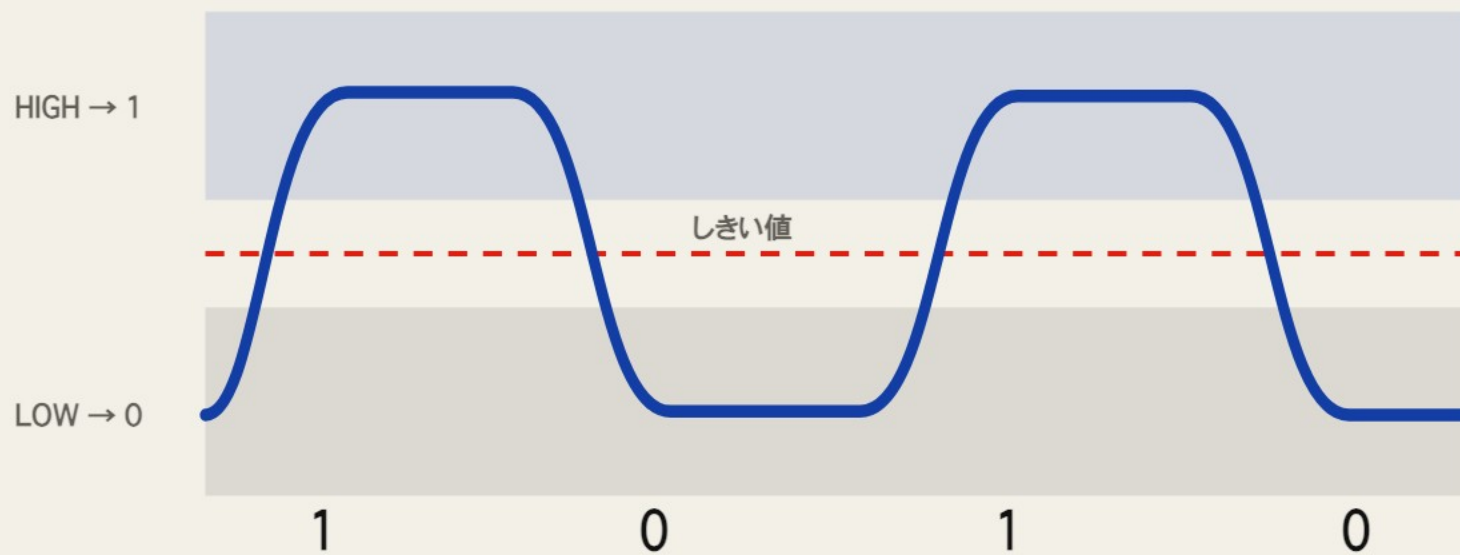
コンピュータの回路



二つの状態に分ける

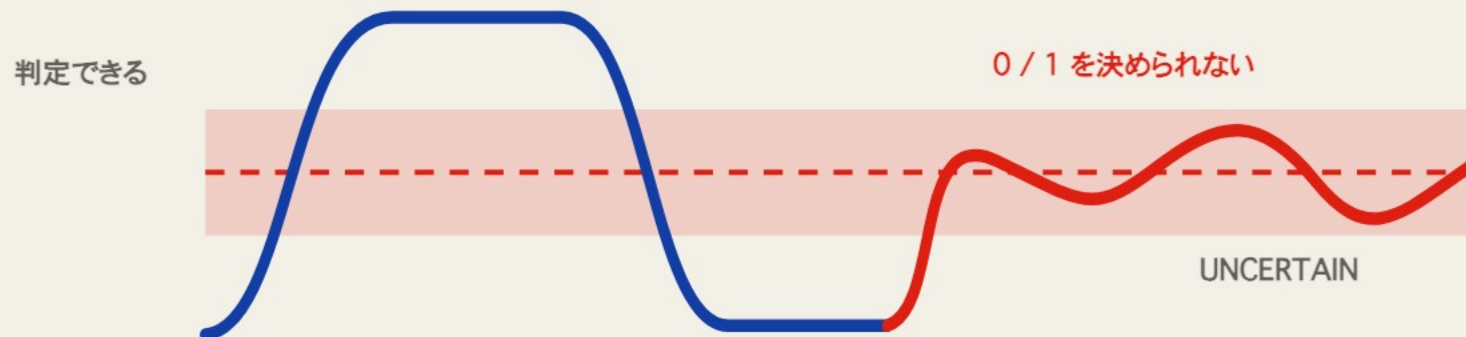
二つの状態なら、小さな乱れを切り離せる？

二つの状態なら信号を作り直せる



波形を写すのではなく、LOW / HIGH を判定して再生する

境界を越える乱れは誤りになる



誤り検出	訂正	再送
------	----	----

LOW と HIGH を、0 と 1 と呼ぶ

情報としてのラベル



0と1は、二つの状態につけた名前

電圧

高い

低い



回路にあるのは、文字の「0」「1」ではなく電気の状態

低い電圧を0、高い電圧を1と呼ぶ → なぜ二つの状態に分ける？

電圧で、電気の通り道を開閉する

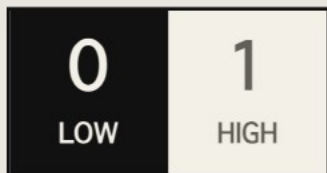
PREVIOUS PAGE

LOW / HIGH を 0 / 1 と呼んだ

では、その電圧で何ができる？

CONTROL VOLTAGE

制御する電圧



ボタンを押して
状態を切り替える

制御電圧

0



電源



消灯

OFF 通り道が切れる

WHAT HAPPENS

1

制御電圧
LOW

2

スイッチ
開く

3

ランプ
消灯

電気で通り道を開閉する、小さなスイッチ = **トランジスタ**

0 と 1 だけで 何を表せる？

見た目が違う情報を、二つの記号だけで区別できるだろうか

Aa 文字

「あ」

 写真

色と位置

 音

波と時間

 動画

画像の連続

0 1

組み合わせで情報を表す

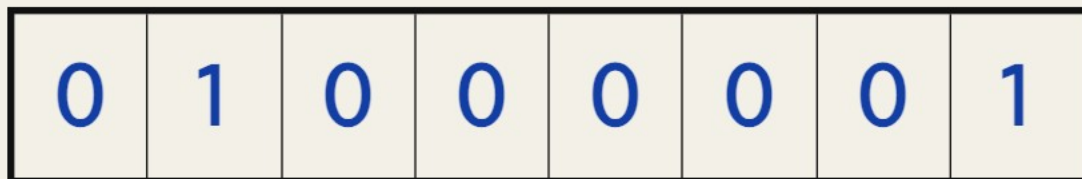
情報を数値に置き換え、0 と 1 で記録する



なぜ、もう一段変換する？ ハードウェアは**二つの状態**を記録するため、数値も**bit 列**にする

同じ 0 と 1 でも、読み方で意味が変わる

保存されている8 bit



INTEGER

65

0から数える整数

数として

文字として

画素として

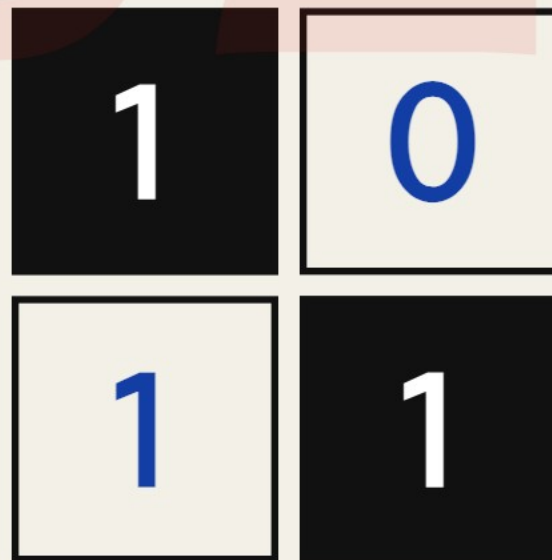
音として

ビット列 + 読み方の規則 = 情報

PART 2 / BITS AS NUMBERS

ビットをまとめて 「数」として読む

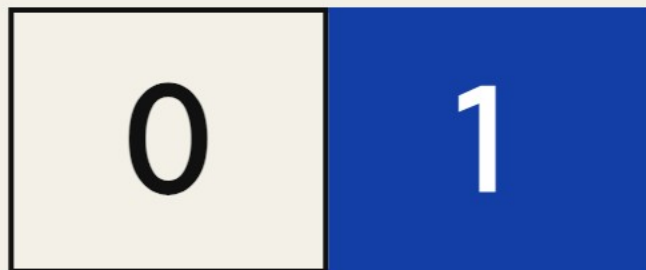
単位を知り、位置と重みを加えます



1 ビットでは、二つの状態を表せる

1 bit

二つから選ぶ



2 通り

1 bit 増えるたびに、通り数は **2 倍**

2 bit

00	01	10	11
----	----	----	----

4 通り

3 bit

000	001	010	011
100	101	110	111

8 通り

n bit → 2^n 通り

8 ビットを、1 バイトと呼ぶ



8 bit = 1 byte

00000000

から

11111111

256 通り

英数字や小さな整数などを、ひとまとまりで扱いやすい

8 bit が必然なのではなく、互換性のための共通ルールとして広まった

WHY EIGHT?

初期 6 / 7 / 8 bit など、機種や用途で長さが違った

1964 IBM System/360 が 8-bit byte を採用

現在 1 B = 8 bit が共通の単位として定着

二進数の桁の重み

8	4	2	1
1	0	1	1

左へ一桁移ると、重みは **2 倍**

二進数 1011_2 を十進数で読む

8	4	2	1
1	0	1	1
1×8	0×4	1×2	1×1

$$8 + 0 + 2 + 1 = 11$$

二進数と十進数を、両方向に変換する

A 01010110_2 $= ?_{10}$ 86	B 178_{10} $= ?_2$ 10110010
---	--

まず一人で考え、次に隣の人と位の重みを確認してください

1 + 1 は、0 を書いて1 を繰り上げる

$0 + 0 = 0$	$0 + 1 = 1$	$1 + 0 = 1$	$1 + 1 = 10$
-------------	-------------	-------------	--------------

$$\begin{array}{r}
 1 \\
 + 1 \\
 \hline
 0
 \end{array}$$

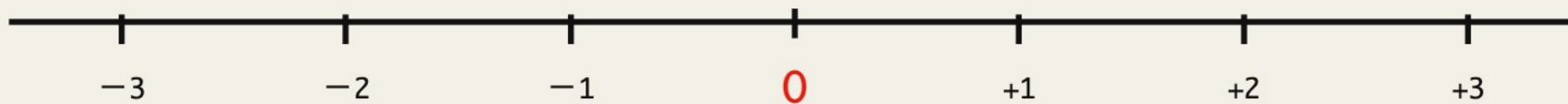
← 1
次の桁へ渡す

繰り上がりを、次の桁へ渡す

$$\begin{array}{rcccc}
 & 1 & 1 & 1 & \\
 & 0 & 1 & 1 & 1 \\
 + & 0 & 0 & 0 & 1 \\
 \hline
 & 1 & 0 & 0 & 0
 \end{array}$$

右端から、一列ずつ同じ規則を繰り返す

0 と 1 だけで、負の数をどう表す？



「-」を保存する場所は、どこにあるのでしょうか

先頭の1 bitを、符号に使うと？

先頭が 0 なら +、1 なら - / 残りの3 bitで大きさを表す

0 0 1 1 +3	1 0 1 1 -3
0 0 0 0 ?	1 0 0 0 ?

下の二つは、それぞれ何を表す？

←

答えを見る →

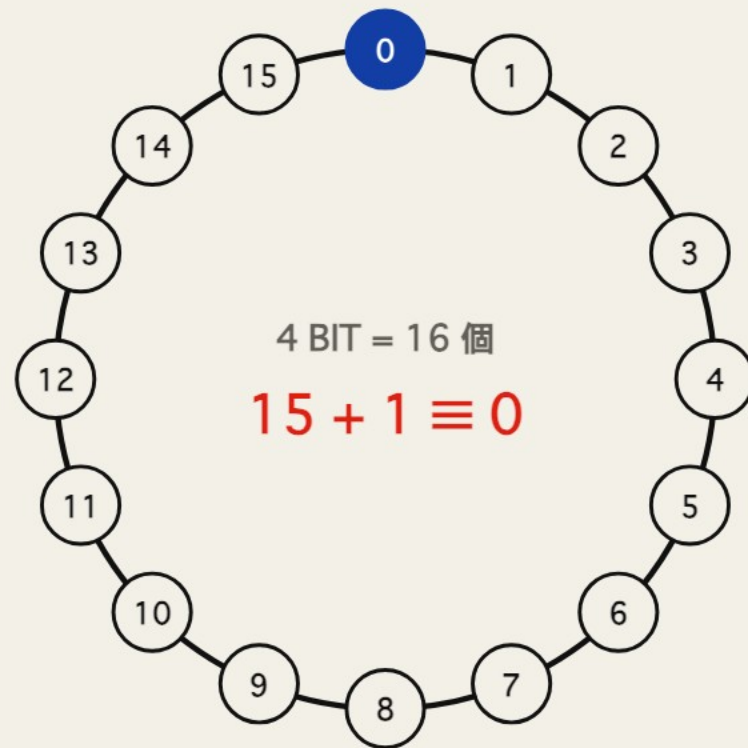
↺

FIXED WIDTH / 16-POSITION CLOCK

4 bit の世界では、 15 の次は 0 に戻る

4 bit で作れる並びは 16 個端まで進むと、最初の位置へ戻ります

$$15 + 1 \equiv 0 \pmod{16}$$

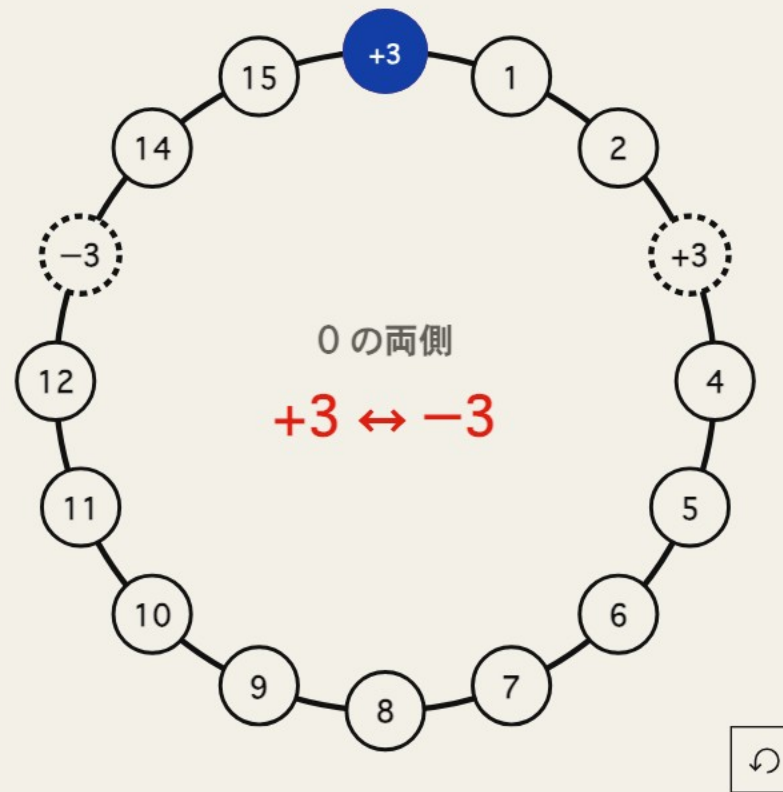


NEGATIVE DIRECTION / AROUND ZERO

数の輪に負の数を置く

時計回りに 3 つ進むと +3 反時計回りに 3 つ進むと -3 です

$$+3 + (-3) \equiv 0 \pmod{16}$$



同じ bit 列を、13 とも -3 とも読める

bit 列	1101	1110	1111	0000	0001	0010	0011
符号なし	13	14	15	0	1	2	3
2 の補数	-3	-2	-1	0	+1	+2	+3

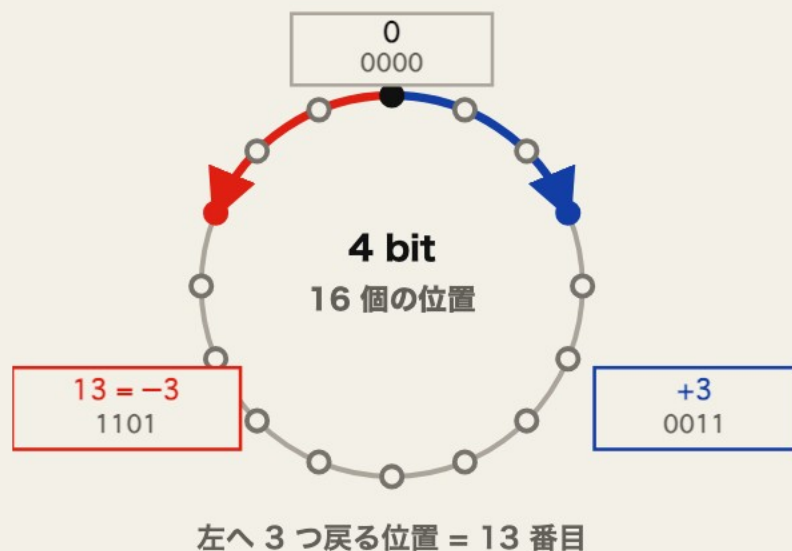
この負数の読み方を「2 の補数」と呼ぶ

4 bit の範囲: -8 ~ +7

-3 は、4 bit のどこにある？

16 POSITIONS / ONE LOOP

0 から逆向きに 3 つ戻る

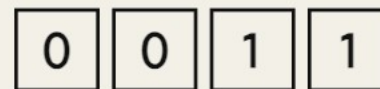


BIT SHORTCUT

その位置を bit 列で見つける

START

+3



まず +3 を 4 bit で書く

↓ 0 ↔ 1 すべて反転

ONE STEP BEFORE

12

まだ -3 の **1 つ手前**

↓ +1 最後に 1 つ進む

ARRIVE

13 = -3



数の輪で -3 と読む位置に着く

だから **反転** → **+1** で、反対の符号の数を作れる

数と桁数を変えて、2の補数を作る

正の数 N から $-N$ を作る

BIT WIDTH

☒ 4 bit☐ 8 bit☐ 16 bit

4 bit の符号付き範囲

 $-8 \sim 7$

01
+3 を bit にする

0 0 1 1

02
0 と 1 を反転

反転すると？

03
1 を足す $\rightarrow -3$

最後に +1

SAME NUMBER / DIFFERENT WIDTH

同じ -3 でも
bit 列は変わる

4 bit

8 bit

16 bit

0 と 1 を反転

ボタンで 2 の補数を一段ずつ作る

4 bit で「7 + 5」を計算すると？

DECIMAL ANSWER $7 + 5 = 12$

01

02

03

04

いまの桁数
4 bit

4 BIT CALCULATION

$$\begin{array}{r} 0111 \quad 7 \\ + \quad 0101 \quad 5 \\ \hline \dots \quad ? \end{array}$$

正しい答えは 12。では、限られた桁数ではどうなる？

SIGNED RANGE

4 bit で表せる範囲



まず、答えが収まるかを見る

bit 数が決まると、表せる数の範囲も決まる

4 bit の範囲を確認

bit 数は、保存できる数の範囲を決める

十六進数でビット列を短く書く



$$1011\ 0010_2 = B2_{16}$$

データは変わらない 人の書き方だけが短くなる

PART 3 / NUMBERS BECOME MEDIA

数字に意味を与える 文字・色・音・動き

何を測り、どの順番に並べるか

03

文字

色

音

動き

数値で表す、文字・画像・音・動画

数を読む規則は分かった

次は、番号を文字として読む

01 文字

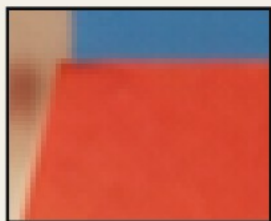
ここから

A 65

文字に番号を割り当てる

A → 65 → 01000001

02 画像



色の値を位置ごとに並べる

位置 + 色の値

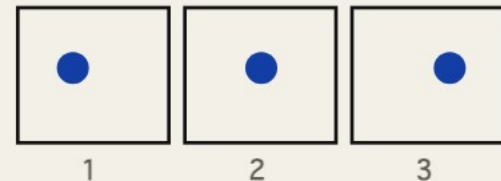
03 音



波の高さを時間順に並べる

時刻 + 振幅の値

04 動画

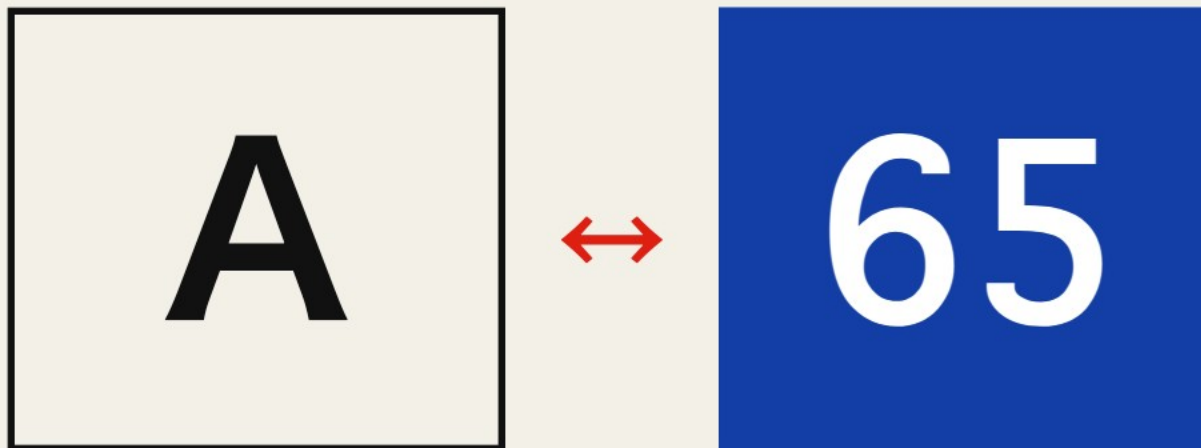


画像を時間順に並べる

画像の列 + 表示時刻

数値を bit 列で記録する 意味を決めるのは、読む規則と並べ方

文字コードは、文字と番号を対応させる規則



番号だけでは、どの対応表を使うか分かりません

ASCII：文字に番号をつける

文字を選ぶ

目録 全128コード



対応する番号



65

十進数

ASCIIは英字・数字・記号などに
7 bit (0~127) の番号を割り当てる

下の図は1 byteにそろえた8桁表記

同じ番号の書き方

十進数 **65** = 十六進数 **41** = 二進数 **01000001**

ASCIIだけでは「あ」を表せない → [世界の文字を扱うUnicodeへ](#)

01000001 が「A」に見えるのは、なぜ？



どの**読み方の規則**を使えば「A」になる？



bit 列そのものではなく、**読み方の規則**が意味を与える

規則を取り違えると → 文字化け

Unicode : 世界の文字に共通の番号をつける

A U+0041	あ U+3042	 U+1F600
--------------------	--------------------	--

コードポイントは、文字につけた共通の番号です

UTF-8 : 文字の番号をバイト列にする

A	あ	😄
41	E3 81 82	F0 9F 98 80
1 byte	3 byte	4 byte

覚えるのは値ではなく、文字によって byte 数が違うこと

文字化けと、フォントの欠字

01 文字化け

番号の読み方を間違える

こんにちは UTF-8

E3 81 93 E3 82 93 ...

保存したバイト列は同じ

Shift_JISとして読む



縛雍 S 縛オ縛。縛ッ

保存時と同じ文字コードで読む

「読めない」が同じでも、直す場所が違う

02 フォントの欠字

描くための形がない

あ U+3042

文字の番号は正しく分かっている

フォント = 文字の形のデータ

字形がないフォント



その文字に対応したフォントを使う

文字 番号 → 文字 → 画像 番号 → 色

数値で表す、文字・画像・音・動画

文字は、番号で区別できた

次は、写真の色と位置を数字にする

01 文字

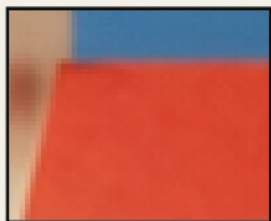
A 65

文字に番号を割り当てる

A → 65 → 01000001

02 画像

ここから



色の値を位置ごとに並べる

位置 + 色の値

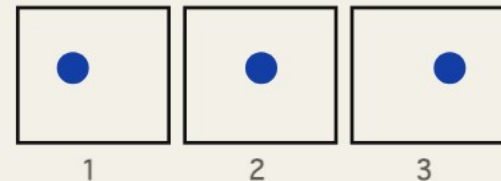
03 音



波の高さを時間順に並べる

時刻 + 振幅の値

04 動画



画像を時間順に並べる

画像の列 + 表示時刻

数値を bit 列で記録する 意味を決めるのは、読む規則と並べ方

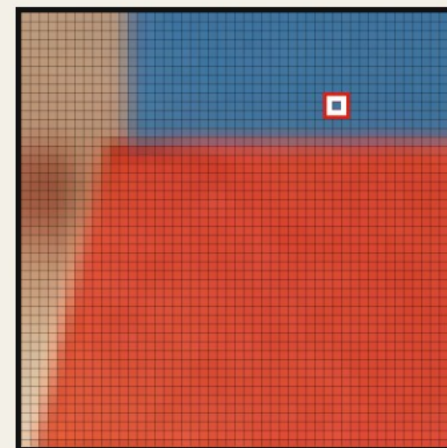
写真を拡大すると、小さなピクセルが見えてくる



CLICK / 50×50 を選ぶ

1672 × 941 pixels

→
× 5



50 × 50 PIXELS

CLICK / 1 pixel を選ぶ

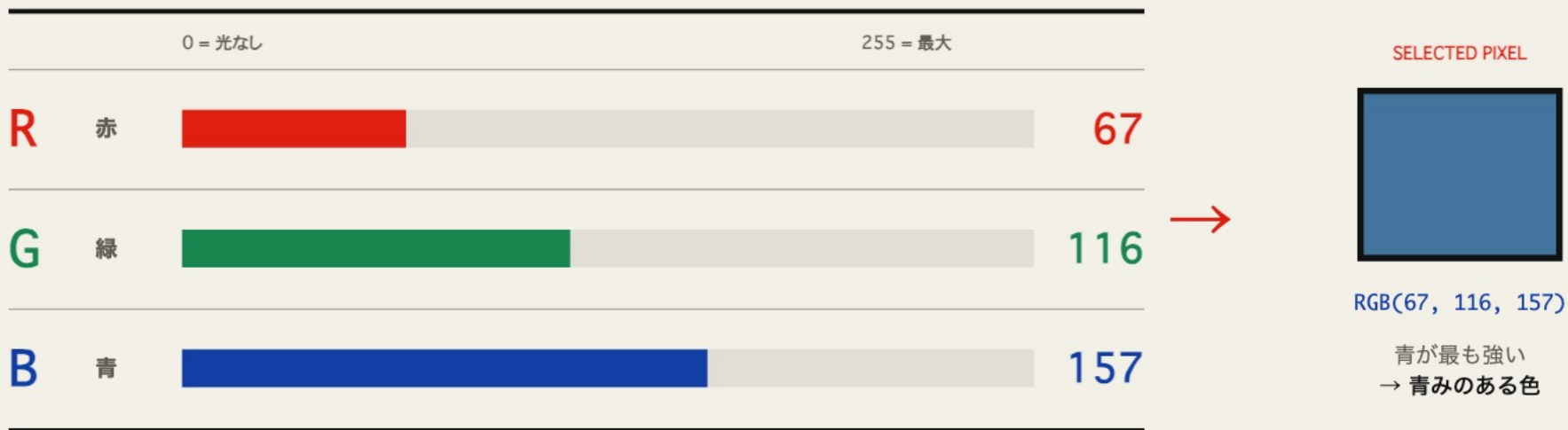


ONE PIXEL / POSITION (795, 375)

R 67 G 116 B 157

写真のどこを選んでも、最後は位置と RGB の数字になる

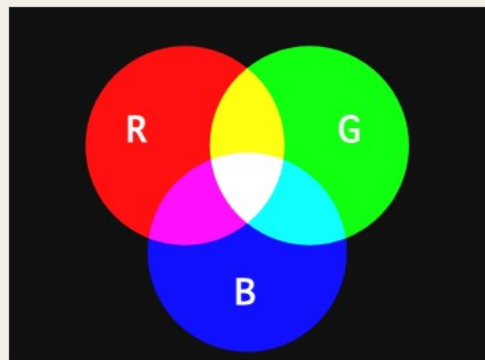
三つの数字は、赤・緑・青の光の強さを表す



1 channel = 8 bit = 256 段階 3 channel で 24 bit 約 1,677 万色

画面は RGB、印刷は CMY(K) : 色の作り方が違う

DISPLAY / LIGHT
加法混色



光を足す

重なるほど
明るくなる

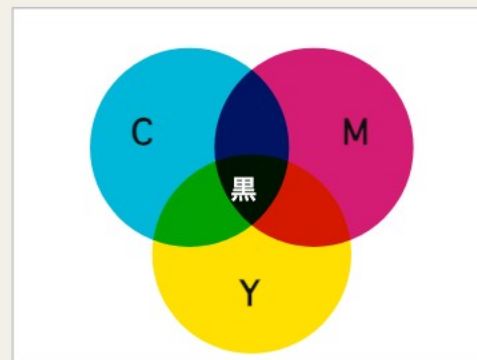
黒 → 白

画面・プロジェクタ

R 赤 G 緑 B 青

RGB

PRINT / INK
減法混色



光を吸収する

重なるほど
暗くなる

白い紙 → 黒

K = 黒インク

C シアン M マゼンタ Y イエロー

CMY + K

COLOR MODEL

RGB / CMYK

色をどう組み立てるか



STANDARD / PROFILE

sRGB / ICC profile

機器の間で、同じ色をどう対応させるか

数値で表す、文字・画像・音・動画

画像は、位置ごとに色を記録した

次は、時間ごとに波の高さを記録する

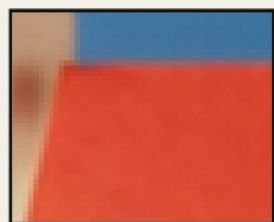
01 文字

A 65

文字に番号を割り当てる

A → 65 → 01000001

02 画像



色の値を位置ごとに並べる

位置 + 色の値

03 音

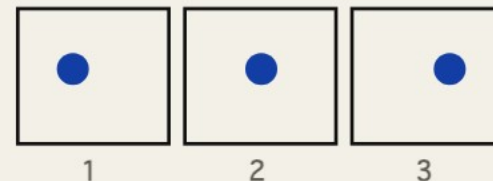
ここから



波の高さを時間順に並べる

時刻 + 振幅の値

04 動画



画像を時間順に並べる

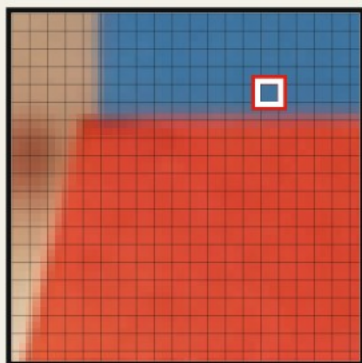
画像の列 + 表示時刻

数値を bit 列で記録する 意味を決めるのは、読む規則と並べ方

画像は空間に「色」を、音は時間に「振幅」を並べる

IMAGE / SPACE

画像

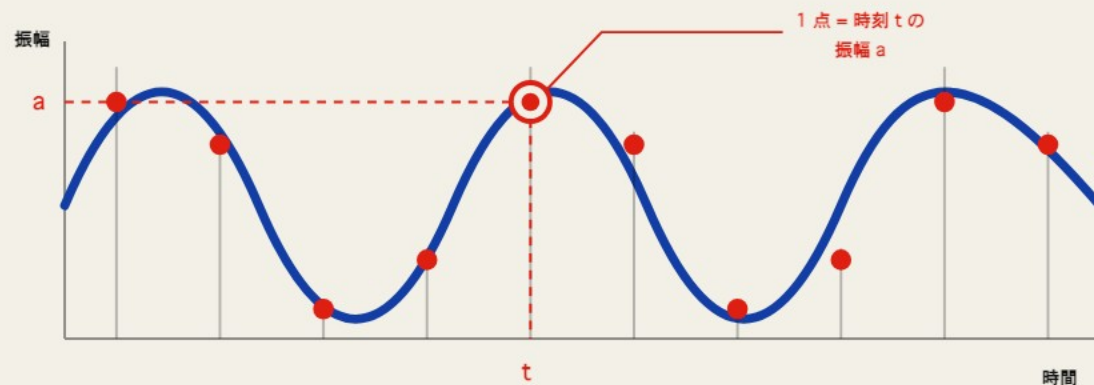
位置 (x, y) + RGB

格子のどこに、何色があるか

並べる軸
空間
→
時間

AUDIO / TIME

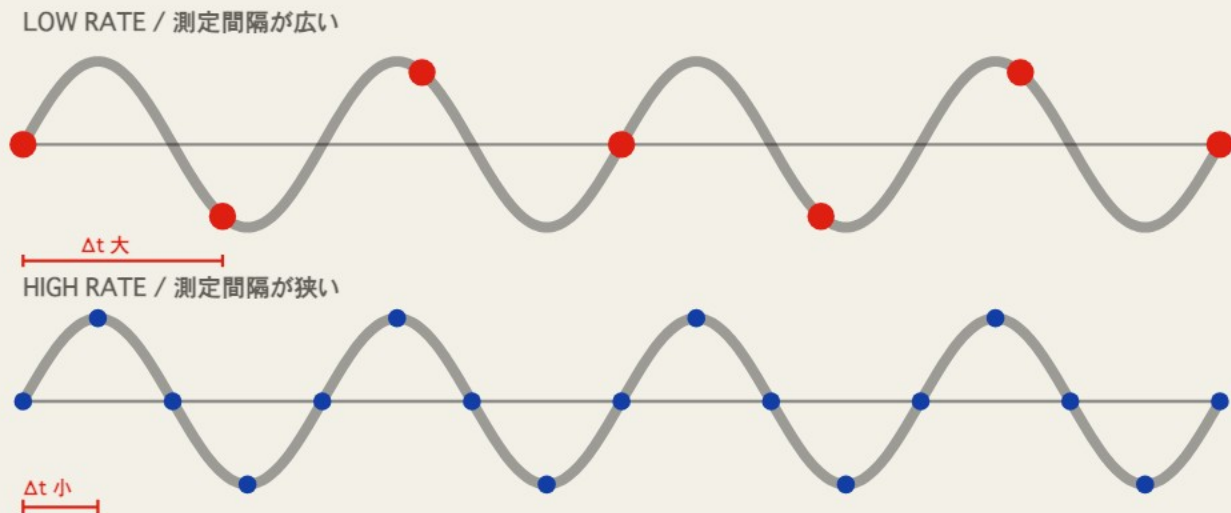
音

時刻 t + 振幅 a

いつ、波がどの高さだったか

値だけでなく、どこ／いつに並ぶかも情報の一部

標本化周波数：1秒あたりの測定回数



RELATION

測定間隔 Δt が小さい $\rightarrow$ 1秒あたりの点が多い

$$f_s = 1 / \Delta t$$

44.1 kHz = 1秒間に 44,100 回

同じ 7 秒でも、測る回数を減らすと何が変わる？

Beethoven, Symphony No. 5 — III. Allegro / recorder ensemble: Papalin

01 / REFERENCE

44.1 kHz

1 秒に 44,100 回

▶ 0:00 / 0:00

7 秒

mono / 16 bit

617 kB

02 / THEN

8 kHz

1 秒に 8,000 回

▶ 0:00 / 0:00

7 秒

mono / 16 bit

112 kB

03 / LAST

4 kHz

1 秒に 4,000 回

▶ 0:00 / 0:00

7 秒

mono / 16 bit

56 kB

高い音

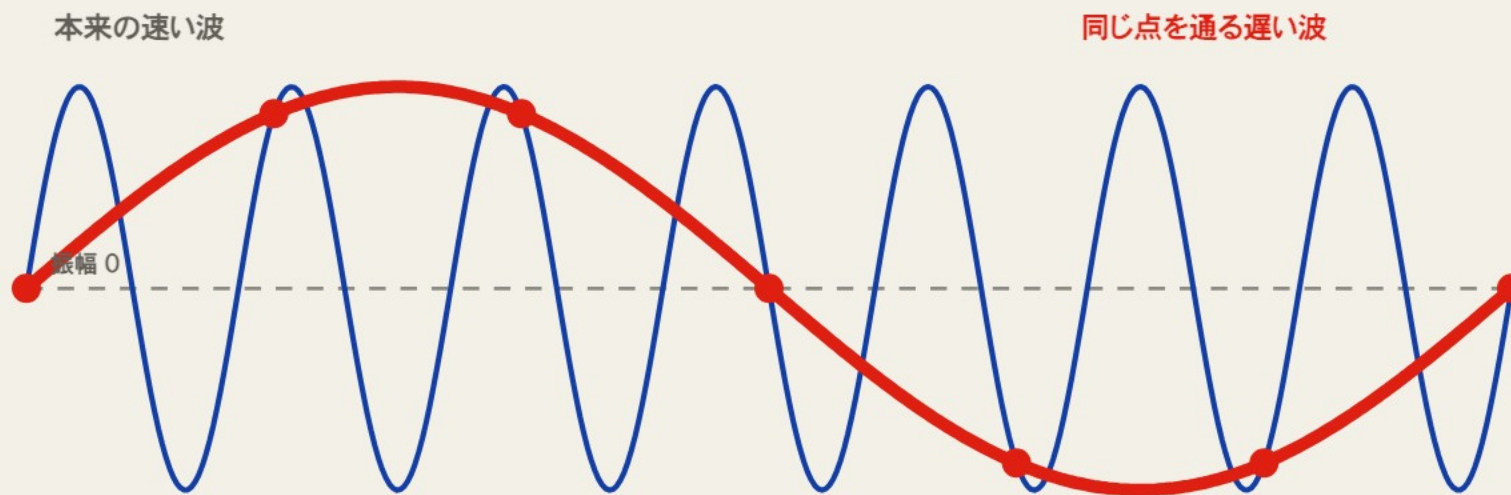
音の輪郭

こもり方

どこが変わった？

標本化周波数 ↓ → 1 秒あたりのデータ量 ↓ / 高い変化を捉えにくくなる

測る間隔と見える波の違い



記録したい最高周波数の 2 倍を超える 標本化周波数が必要

測った波の高さを、bit 列にする

1 回の測定に使う桁数

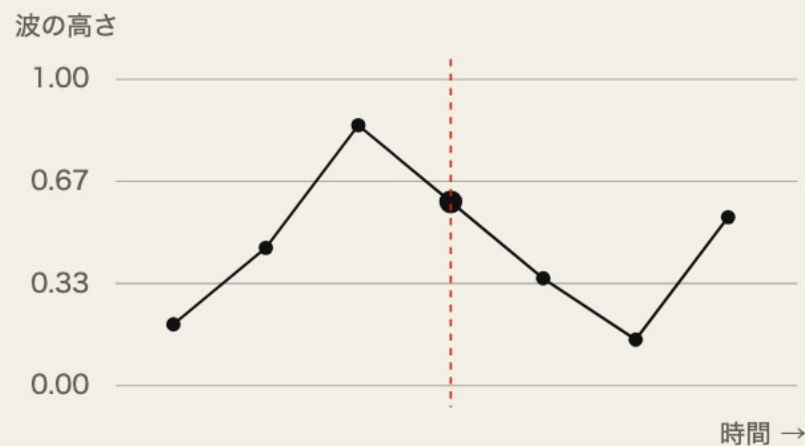
2 bit

3 bit

4 bit

4 段階

● 測った高さ ● 記録する高さ 差 = 誤差



測定 1 2 3 4 5 6 7

測った高さ 0.60

高さは 0~1 に換算した模式値 / 測定時刻は固定



近い目盛りに丸める →

測った値 0.60

4 段階のどこに記録する？

01 最も近い目盛りに丸める

?

02 目盛りの番号を bit 列にする

番号 ? → ..

限られた bit 数では、細かな値をすべて区別できない

標本化：いつ測るか 量子化：何段階で記録するか

数値で表す、文字・画像・音・動画

音は、値を時間順に記録した

次は、画像そのものを時間順に並べる

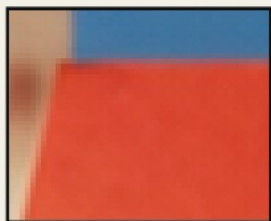
01 文字

A 65

文字に番号を割り当てる

A → 65 → 01000001

02 画像



色の値を位置ごとに並べる

位置 + 色の値

03 音

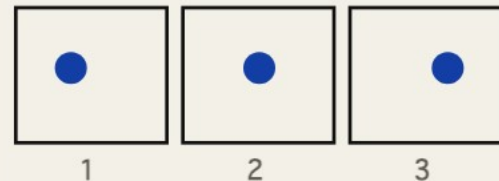


波の高さを時間順に並べる

時刻 + 振幅の値

04 動画

ここから



画像を時間順に並べる

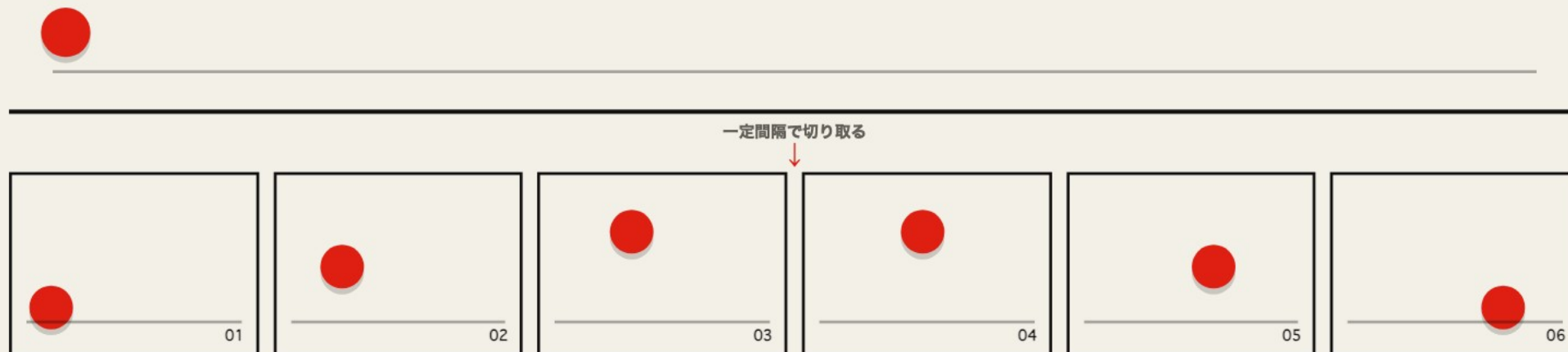
画像の列 + 表示時刻

数値を bit 列で記録する 意味を決めるのは、読む規則と並べ方

動画を分解すると静止画の列になる

CONTINUOUS MOTION

連続して動く



動画 = 画像の列 + 時間

映画の 1 秒には、何枚の画像がある？



12 fps

動きが飛んで見える



1 秒

24 fps

細かくつながる



1 秒

60 fps

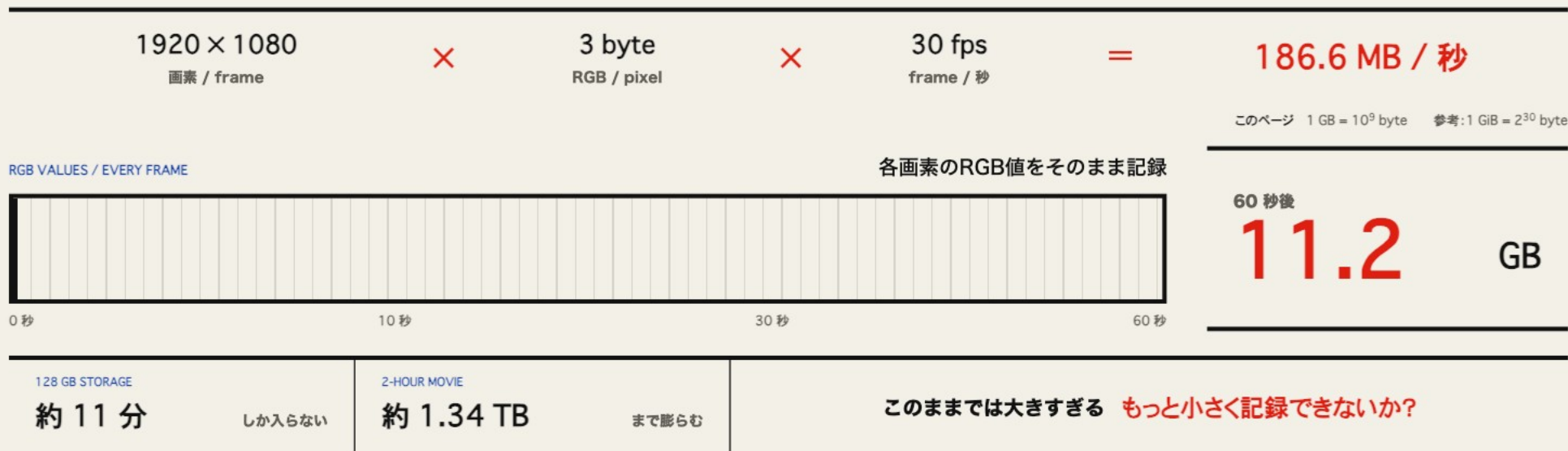
より滑らかに見える



1 秒

fps ↑ → 動きが滑らか / 1 秒あたりの画像数とデータ量も増える

動画を1分記録すると、何GBになる？



大きすぎるデータを、 どう小さくする？

同じ内容を、より短いビット列で表せるでしょうか

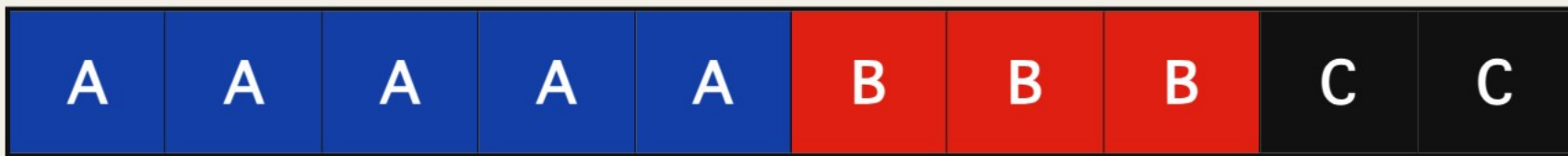
04
101100101011001010
110010



短く表す

11.2 GB / 分 → より短いビット列へ

同じ値の繰り返しを「値＋回数」で表す



A × 5	B × 3	C × 2
-------	-------	-------

連続して同じ値が多いほど効果が大きいABCABC... には効きにくい

圧縮したあと、元に戻せる？

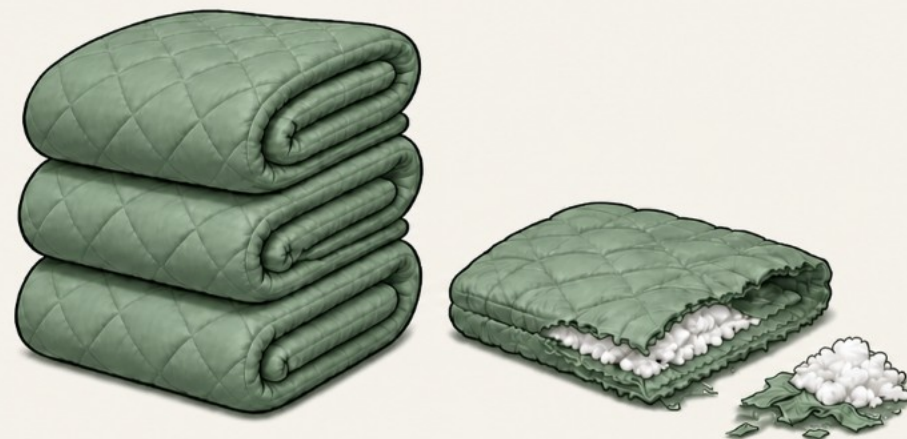


LOSSLESS

可逆圧縮

形を変えて小さくする
元と同じ状態に戻せる

ZIP・PNG・FLAC



LOSSY

非可逆圧縮

一部を捨てて小さくする
元と完全には戻らない

JPEG・MP3・H.264

文書・プログラムは完全復元が必要 写真・音声・動画は目的に応じて情報量を減らせる

画像の非可逆圧縮の代表例：JPEG

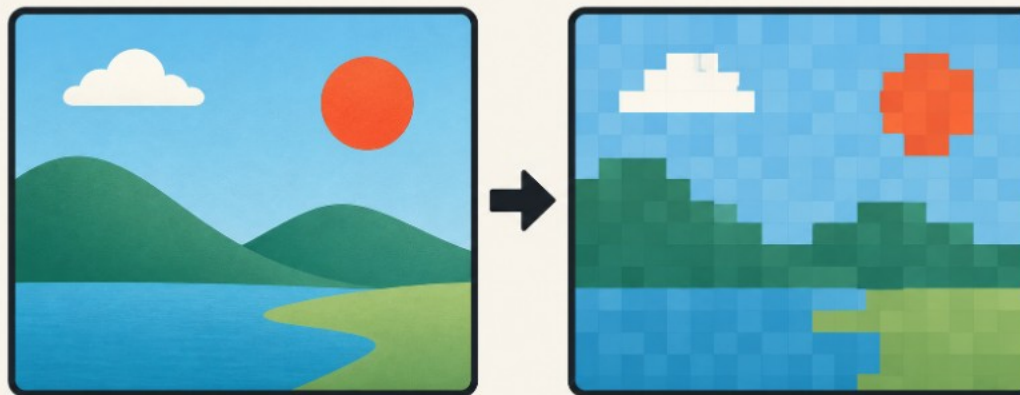
.jpg / .jpeg

写真 file でよく見る拡張子

見た目をなるべく保ちながら、file size を小さくする

JPEG の大まかな考え方

- 明るさの変化は、比較的しっかり残す
- 色の細かな変化は、少し減らす
- 画像を **8×8 block** に分け、細かな変化を粗く表す



細かな変化がある画像

細かな変化をまとめた画像

大きな形は残る圧縮を強くすると、細部や境界の違いが見えてくる

OTHER IMAGE FORMATS

目的に合わせて方式を選ぶ

PNG

図・文字・透明背景

可逆

WebP

Web の画像

可逆／非可逆

GIF

簡単な animation

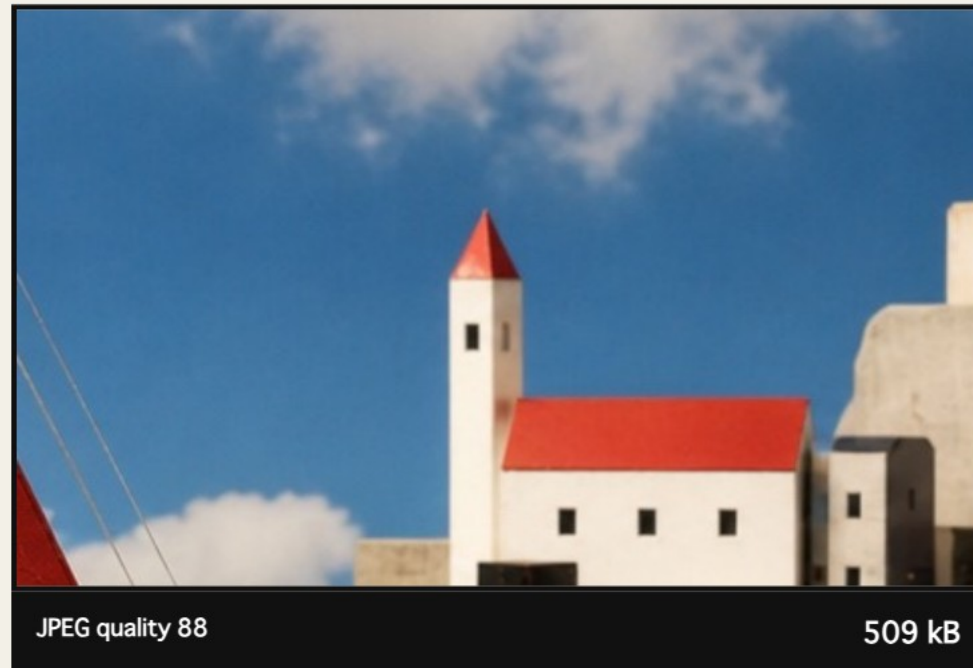
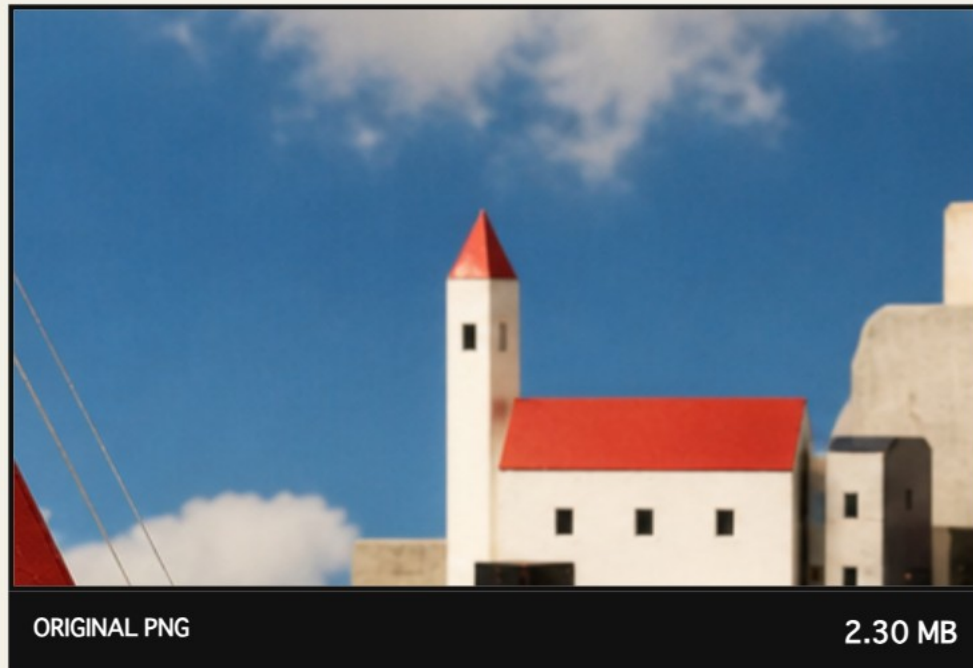
可逆・256 色

JPEG圧縮を同じ写真で比較する



同じ原画像		同じ切り抜き		同じ表示サイズ		変えるのは圧縮の強さだけ
基準 2.30 MB	ORIGINAL PNG	→	まず 見た目と容量は？	JPEG QUALITY 88	→	次に どこが崩れる？ JPEG QUALITY 12

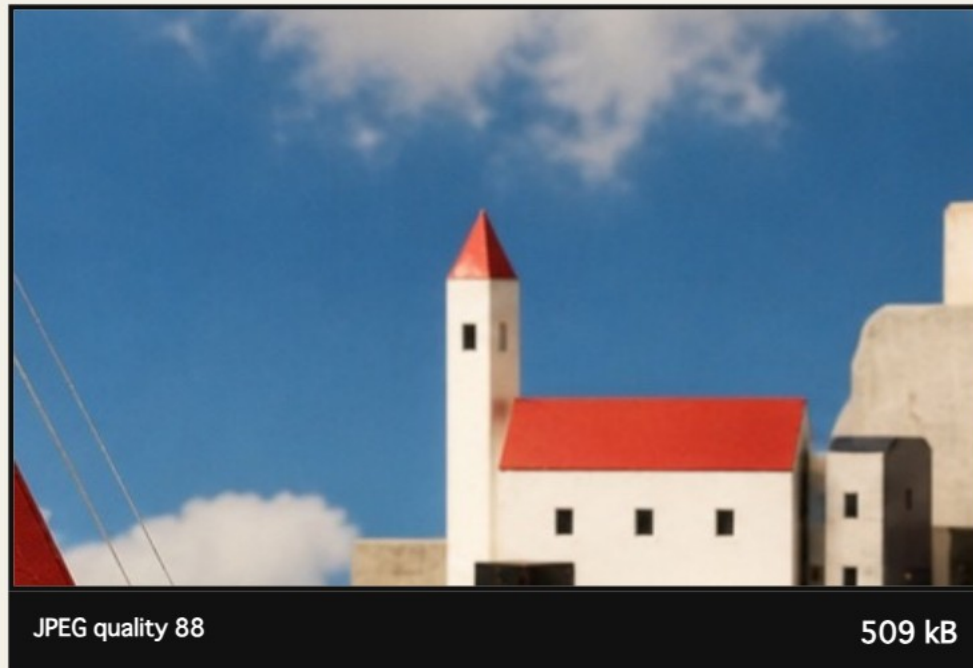
見た目はほぼ同じ、容量は約1/4



約 78% 減

違いがほとんど見えないことが、この結果のポイント

さらに小さくすると細部が崩れる



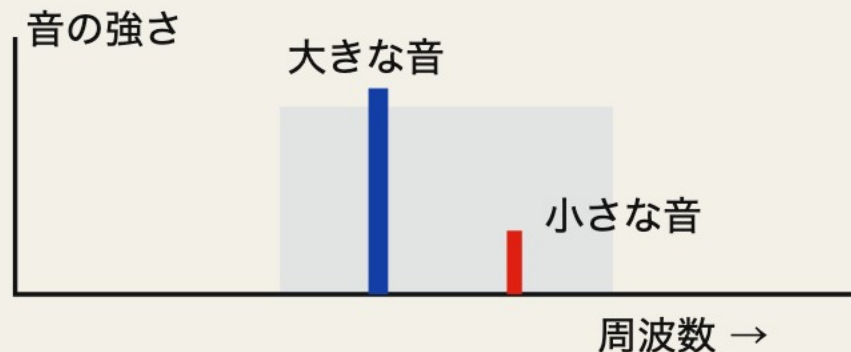
A 空が block 状になる

B 屋根の境界がにじむ

C 細い線が消える

音声圧縮：聞こえ方を利用する

小さな音を、聞き分けられる？



▶ 小さな音だけ

▶ 二つを同時に

☐

模式図・試聴用の合成音 / 聞こえ方は音量・環境で変わる

人の聞こえ方を利用

マスキング

大きな音があると、近い周波数の
小さな音を聞き取りにくい

↓ 聞こえにくい成分を
より少ないbitで表す

MP3・AAC

音声の非可逆圧縮方式

1秒のデータ量の例 **PCM 約1,411 kbit** → **AAC 256 kbit**

標本化周波数を下げることとは、**別の方法**

44.1 kHz・16 bit・ステレオのPCMと比較
AACの容量・音質は設定や内容による

動く映像にも、繰り返し現れる情報がある



実際の動画・6 秒／Lumière「列車の到着」

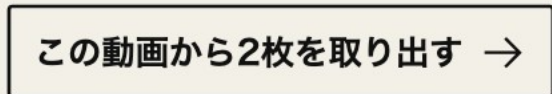
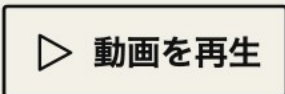
まず、動画を見る

列車が近づき、人が歩く

動きがあるたびに、画面は変わる

でも、建物や線路まで
毎回すべて描き直す必要がある？

この動画から、**0.5 秒離れた2枚**を
取り出して比べる



1 / 3

映像と音声を、一つのファイルにまとめる

映像と音声は別々に圧縮したのに、なぜ **movie.mp4** は一つ？

01 / 別々のデータ

映像・音声・字幕

映像 

音声 

字幕 列車が到着します

02 / 前のページまで

それぞれ圧縮する

H.264 / 映像

AAC / 音声

映像と音声で、使う方式が違う

一緒にまとめる情報

字幕・時間情報

映像

音声

字幕

時刻

03 / 今回のポイント

一つのファイルにまとめる

映像 **H.264**

音声 **AAC**

字幕 **表示する文字**

時刻 **いつ再生・表示するか**

時刻を合わせて、映像と音声を再生できる

movie.mp4

+ 仕組みの名前を確認

「小さくする方式」と「一つにまとめる形式」は別のもの

圧縮方式の選び方

完全に戻す必要がある？

YES

可逆圧縮

プログラム・文書・原本

人が見る／聞かため？

YES

非可逆も選べる

配信用の写真・音声・動画

保存用の原本？

YES

高品質または可逆を優先

後から再編集する素材

「一番よい形式」ではなく、**目的に合う形式**を選びます

「デジタルの自分」を分解してみよう

名前 自分の名前を 0 と 1 に	日本語 1 文字の UTF-8 を確かめる	形 byte 列から生きものをつくる
-----------------------------	---------------------------------	------------------------------

ミニ体験を開く



情報の表現：今回のまとめ

LOW / HIGH

判定して同じ状態を再生できる

→ 0 / 1

1	0	1	1
0	0	1	0

+

位置と重み

文字の対応表

空間と色

時間と測定

=

数

文字

画像

音・動画

データが大きくなれば、目的に合わせて圧縮の規則を選びます

次回：0と1を扱う回路の仕組み

THIS LECTURE

0 と 1 の規則

$$1 + 1 = 10$$

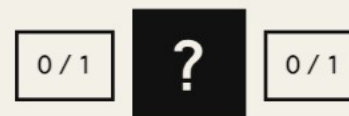
PHYSICAL STATE

電気の ON / OFF



NEXT LECTURE

規則を実行する回路



次回：回路の基本単位「論理ゲート」から始めます