



## Compression

---



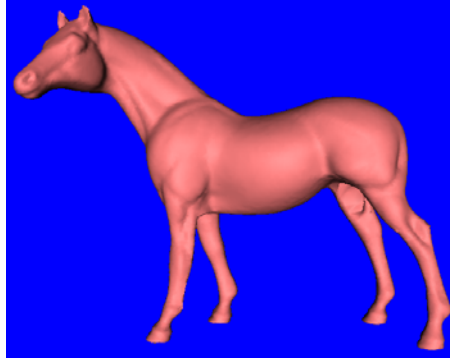
## Motivation

---

- **Bandwidth:** Communicate large complex & highly detailed 3D models through low-bandwidth connection (e.g. VRML over the Internet)
- **Storage:** Store large & complex 3D models (e.g. 3D scanner output)



## Mesh Compression



University of  
British Columbia

VRML = 200K, zipped VRML = 70K, compressed = 15K



## Compression

- Represent data by short (shortest) sequence of bits
- Compression achieved by using data coherence & distribution
  - have **small** set of symbols (bit strings) encoding data
  - use shorter encodings for frequent symbols
    - works for **uneven** distribution

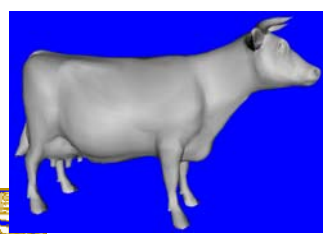


University of  
British Columbia



## Mesh Encoding

- Input: 3D triangle mesh
  - Assumed to be orientable manifold
- Output: bit stream



010011110010101100010101 ... →



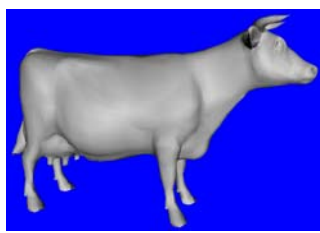
University of British Columbia



## Mesh Decoding

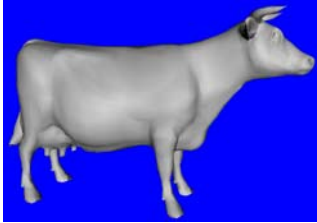
- **Input:** Bit stream
- **Output:** Reconstruction of original 3D triangle mesh

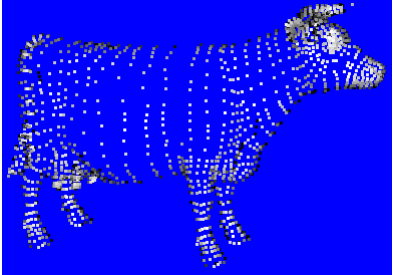
→ ...100111010010101100010101



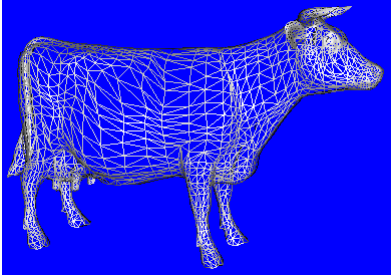
University of British Columbia

## Mesh Compression (Coding)







Geometry



Connectivity

VRML = 278K, zipped VRML = 70K, compressed = 9K






## Definitions

- **Alphabet**: Finite set containing at least one element:  

$$\mathbf{A} = \{a, b, c, d, e\}$$
- **Symbol**: Alphabet element:  $s \in \mathbf{A}$
- **String (over alphabet)**: Sequence of symbols:  
`ccdabdcaad...`
- **Codeword**: Sequence of bits representing coded symbol or string:  
`110101001101010100...`
- $p_i$ : Occurrence probability of symbol  $s_i$  in input string

$$\sum_{v_i \in \mathbf{A}} p_i = 1$$



# Entropy

- Entropy of set of elements  $e_1, \dots, e_n$  with probabilities  $p_1, \dots, p_n$ :

$$H(p_1 \dots p_n) \equiv - \sum_{\forall i} p_i \log_2 p_i$$

- Entropy** = smallest number of bits needed **on average** to represent symbol
  - average on all symbols code lengths
- $\log_2 p_i$  is **uncertainty** in symbol  $e_i$  (or "surprise" when we see this symbol)
- Entropy – average "surprise"
- Assumption: No dependencies between symbols' appearances



University of British Columbia



# Entropy example

Entropy calculation for two symbol alphabet

Example 1:

$A$	$p_A=0.5$
$B$	$p_B=0.5$

$$H(A, B) = -p_A \log_2 p_A - p_B \log_2 p_B =$$

$$= -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

*Requires one bit per symbol on average to represent the data*

Example 2:

$A$	$p_A=0.8$
$B$	$p_B=0.2$

$$H(A, B) = -p_A \log_2 p_A - p_B \log_2 p_B =$$

$$= -0.8 \log_2 0.8 - 0.2 \log_2 0.2 \cong 0.7219$$

*It requires less than one bit per symbol on average to represent data*  
*How can we code this ?*

University of British Columbia



## Entropy examples

- Entropy of  $e_1, \dots, e_n$  is maximized when

$$p_1 = p_2 = \dots = p_n = 1/n \rightarrow H(e_1, \dots, e_n) = \log_2 n$$

- No symbol is “better” than the other or contains more information
  - $2^k$  symbols may be represented by  $k$  bits
- Entropy of  $p_1, \dots, p_n$  is minimized when

$$p_1 = 1, p_2 = \dots = p_n = 0 \rightarrow H(e_1, \dots, e_n) = 0$$



University of  
British Columbia



## Entropy coding

- Entropy is **lower bound** on average number of bits needed to represent symbols = data compression limit
- Entropy coding methods:
  - Aspire to achieve **the entropy** for given alphabet  
BPS  $\rightarrow$  Entropy
  - Code achieving the entropy limit is optimal
- BPS : bits per symbol

$$\text{BPS} = \frac{|\text{encoded message}|}{|\text{original message}|}$$



University of  
British Columbia



## Code types

- **Fixed-length codes** - all codewords have same length (number of bits)

A – 000, B – 001, C – 010, D – 011, E – 100, F – 101

- **Variable-length codes** - may give different lengths to codewords

A – 0, B – 00, C – 110, D – 111, E – 1000, F - 1011



University of  
British Columbia



## Code types (cont.)

- **Prefix code** - No codeword is prefix of any other codeword

A = 0; B = 10; C = 110; D = 111

- **Uniquely decodable code** - Has only one possible source string producing it

- Unambiguously decoded

- Examples:

- Prefix code - end of codeword immediately recognized (without ambiguity) : 010011001110 →  
0 | 10 | 0 | 110 | 0 | 111 | 0

- Fixed-length code



University of  
British Columbia



## Huffman coding

- Each symbol assigned *variable-length* code depending on its frequency
  - Higher frequency = shorter codeword
- Prefix code
- Huffman code is **optimal** prefix and variable-length code **given** symbols' probabilities of occurrence
- Codewords generated by building Huffman Tree



University of  
British Columbia



## Huffman tree example

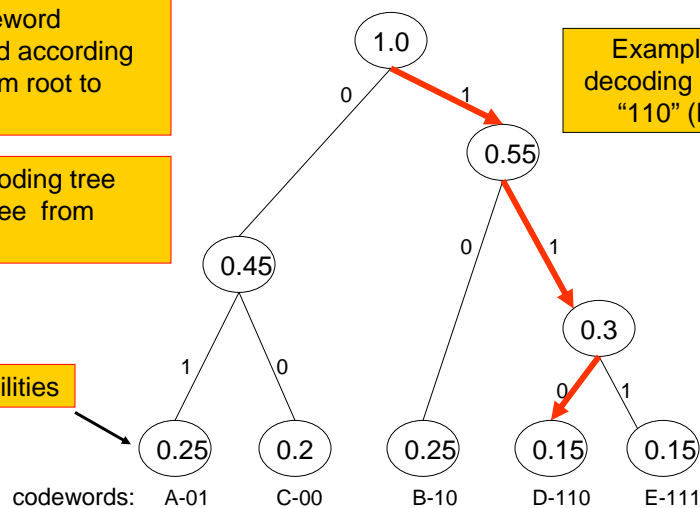
Each codeword determined according to path from root to symbol

When decoding tree traverse tree from root.

Example:  
decoding input  
"110" (D)



University of  
British Columbia







## Huffman encoding

- Use codewords from previous slide to encode the string "BCAE":

String:    B    C    A    E  
Encoded:  10  00  01  111

- Number of bits used: 9
- BPS (9 bits/4 symbols) = **2.25**
- Entropy:  $-0.25\log 0.25 - 0.25\log 0.25 - 0.2\log 0.2 - 0.15\log 0.15 - 0.15\log 0.15 = \mathbf{2.2854}$



University of  
British Columbia

BPS is lower than entropy - **WHY ?**



## Symbol probabilities

- How are probabilities known?
  - Counting symbols in input string
    - Data given in advance
    - Requires extra pass on input string
  - Data source's distribution is known
    - Data not necessarily known in advance, but we know its distribution



University of  
British Columbia



## Example

“Global” English frequencies table:

Letter	Prob.	Letter	Prob.
A	0.0721	N	0.0638
B	0.0240	O	0.0681
C	0.0390	P	0.0290
D	0.0372	Q	0.0023
E	0.1224	R	0.0638
F	0.0272	S	0.0728
G	0.0178	T	0.0908
H	0.0449	U	0.0235
I	0.0779	V	0.0094
J	0.0013	W	0.0130
K	0.0054	X	0.0077
L	0.0426	Y	0.0126
M	0.0282	Z	0.0026
Total: 1.0000			



University of  
British Columbia



## Huffman summary

- Achieve entropy when occurrence probabilities are negative powers of 2
- Alphabet & its distribution must be known in advance
- Given Huffman tree, very easy (and fast) to encode and decode
- **Not unique**
  - arbitrary decisions in tree construction



University of  
British Columbia