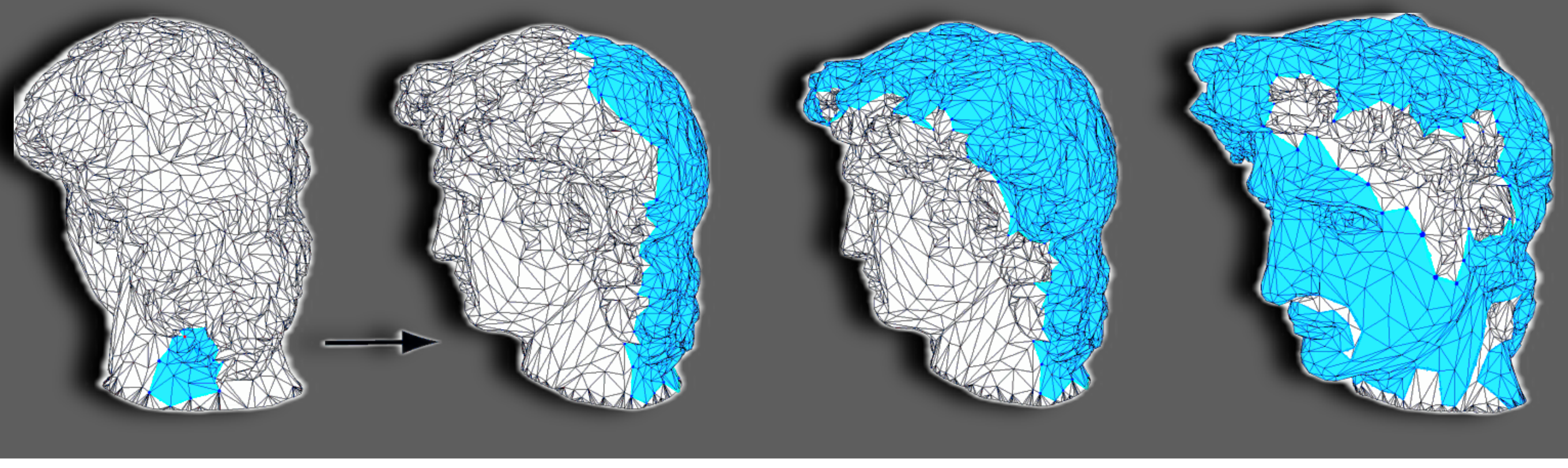
3D Mesh Crypto-Compression

Emery Bourget-Vecchio - Charles Sayamath
Master IMAGINE

3D meshes can be considered the most popular representation of volumes and surfaces and are used on a variety of devices with different performance constraints.

In a need for realism, these are becoming more and more detailed and complex over time. This is accompanied by an increase in the volume of data and can cause problems (memory, speed, performance, etc.) due to the constraints of the different media on which they are displayed.

To solve these problems, 3D mesh compression methods have been developed to reduce the amount of data required to represent them, making them easier to store and transfer.



VALENCE-DRIVEN APPROACH

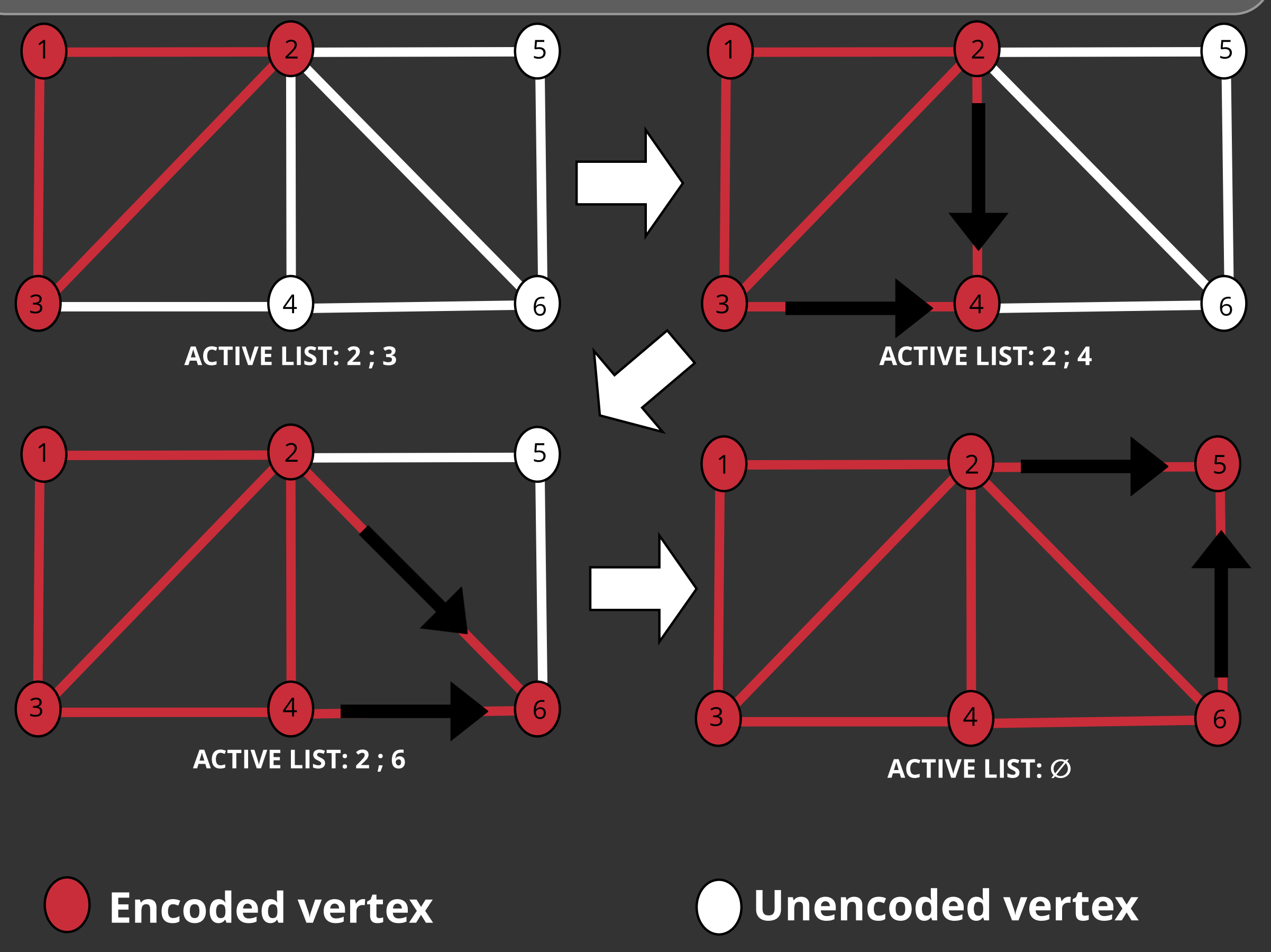
Among the large number of algorithms that exist to encode connectivity, we decided to go with a valence-based approach. It is one of the most optimized algorithms since it encodes our vertex with an average of 3 bits. The principle relies on the conquest of the mesh edges by expanding an initial boundary and encoding the conquered vertices along the edges.

Algorithm:

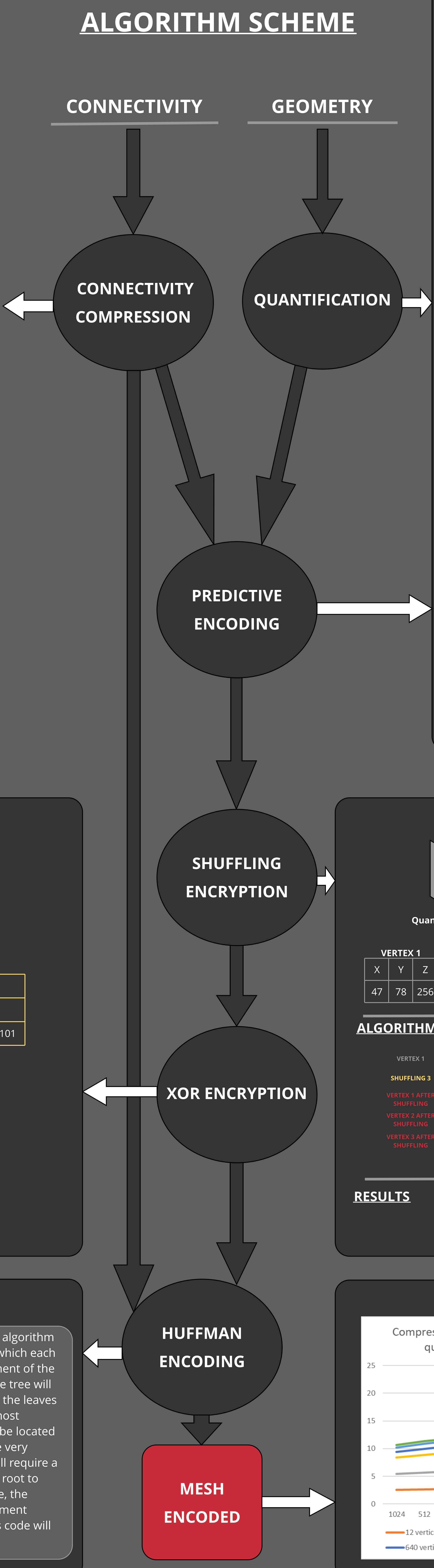
- start from a random triangle whose vertices are added in an active list (AL) representing the boundary to expand
- pick a vertex pivot from the AL
- try to conquer its unencoded edges in counter-clockwise order
- the vertices along each edge are put into the active list and encoded as a symbol that depends on their situation (see below)
- when the vertex has no more unencoded edges, the next vertex in the active list becomes the next pivot and we go on until the AL is empty

3 types of symbols are possible :

- **ADD < vertex valence > :** Occurs when a vertex is added to the active list.
- **SPLIT < valence offset >:** Occurs when the traversed vertex is already present in the active list. The active list is then split in half at the vertex occurrence.
- **MERGE <AL index> <vertex offset>:** Occurs when the traversed vertex is already present in another active list. The two lists are then merged



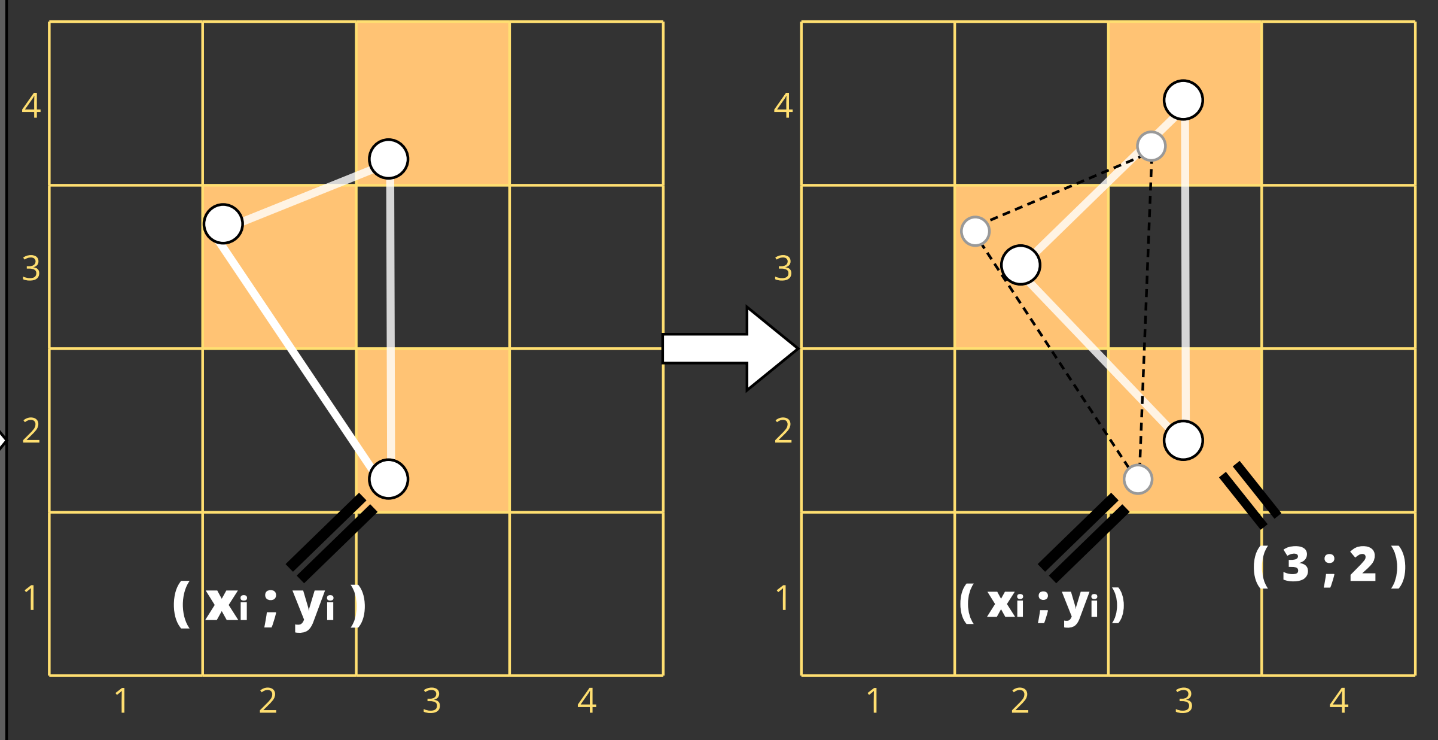
ALGORITHM SCHEME



QUANTIFICATION

First, the coordinates of each vertex are quantified. To do this, we normalize the coordinates between [0;1] and multiply the values by a quantification value.

This allows us to remove the superfluous decimals and thus to encode the vertices on fewer bits.

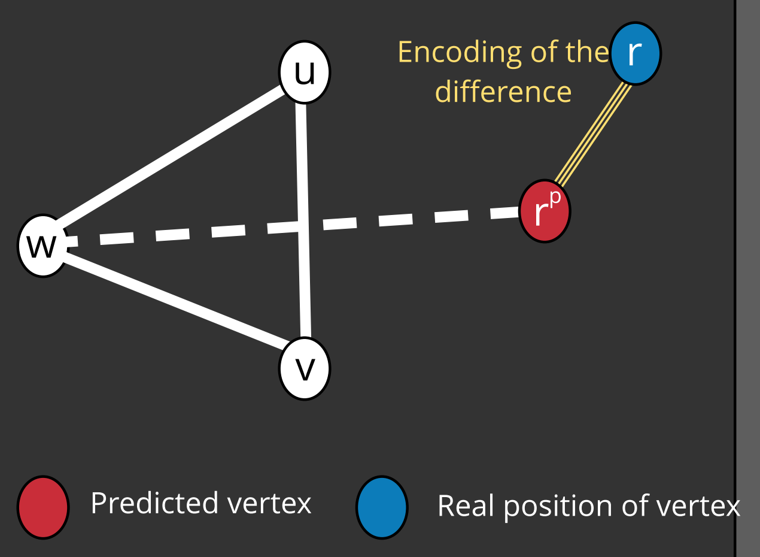


PREDICTION

Parallelogram prediction:

- Predict the position of vertex r using a triangle (u , v , w).
- Compute r^p, the vertex that forms a parallelogram with the triangle.
- Encode the difference between r^p and r

$$r^p = u + v - w$$



XOR ENCRYPTION

MESH

Quantified and predicted vertex

X	Y	Z
330	144	449
0101001010	0010010000	0111000001

KEY

X	Y	Z
729	471	69
1011011001	0111010111	0001000101

XOR

X	Y	Z
915	327	388
1110010011	0101000111	0110000100

SHUFFLING ENCRYPTION

MESH

Quantified and predicted vertex

KEY

	VERTEX 1			VERTEX 2			VERTEX 3	
X	Y	Z	X	Y	Z	X	Y	Z
47	78	256	145	354	142	365	145	2

	SHUFFLING 1			SHUFFLING 2			SHUFFLING 3	
X	Y	Z	X	Y	Z	X	Y	Z
2	3	2	3	2	1	3	1	2

ALGORITHM

	X	Y	Z
VERTEX 1	47	78	256
SHUFFLING 3	2	3	2
VERTEX 1 AFTER SHUFFLING	145	145	142
VERTEX 2 AFTER SHUFFLING	47	354	256
VERTEX 3 AFTER SHUFFLING	365	78	2

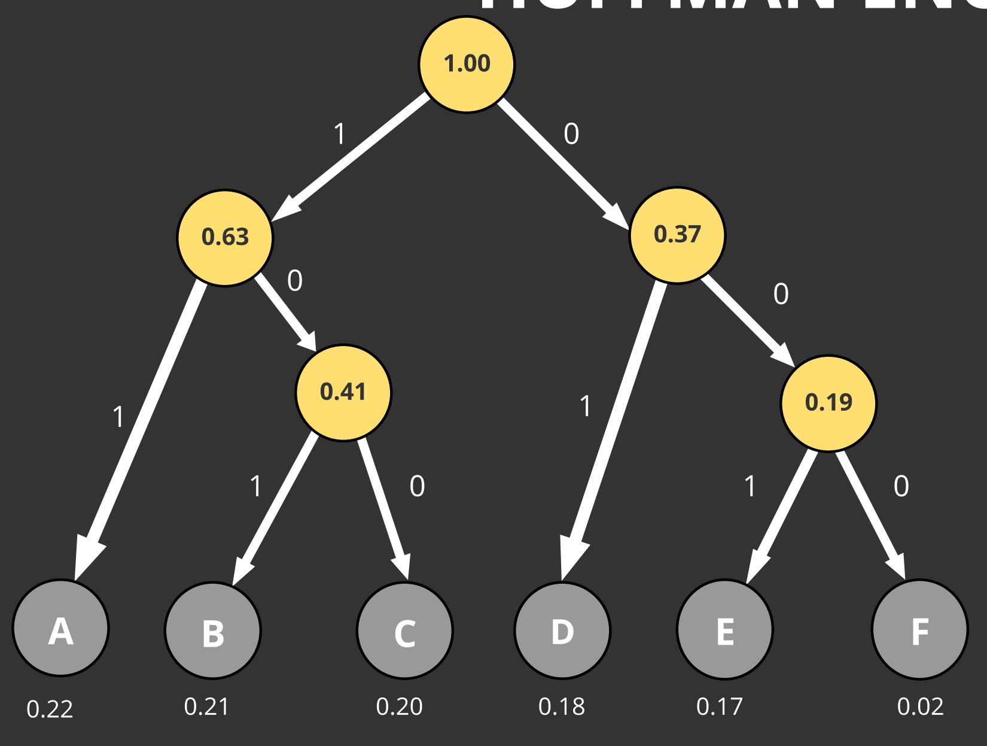
	X	Y	Z
VERTEX 2	47	78	256
SHUFFLING 2	3	2	1
VERTEX 1 AFTER SHUFFLING	145	145	256
VERTEX 2 AFTER SHUFFLING	365	354	142
VERTEX 3 AFTER SHUFFLING	47	78	2

	X	Y	Z
VERTEX 3	47	78	2
SHUFFLING 3	3	1	2
VERTEX 1 AFTER SHUFFLING	145	78	256
VERTEX 2 AFTER SHUFFLING	365	354	2
VERTEX 3 AFTER SHUFFLING	47	145	142

RESULTS

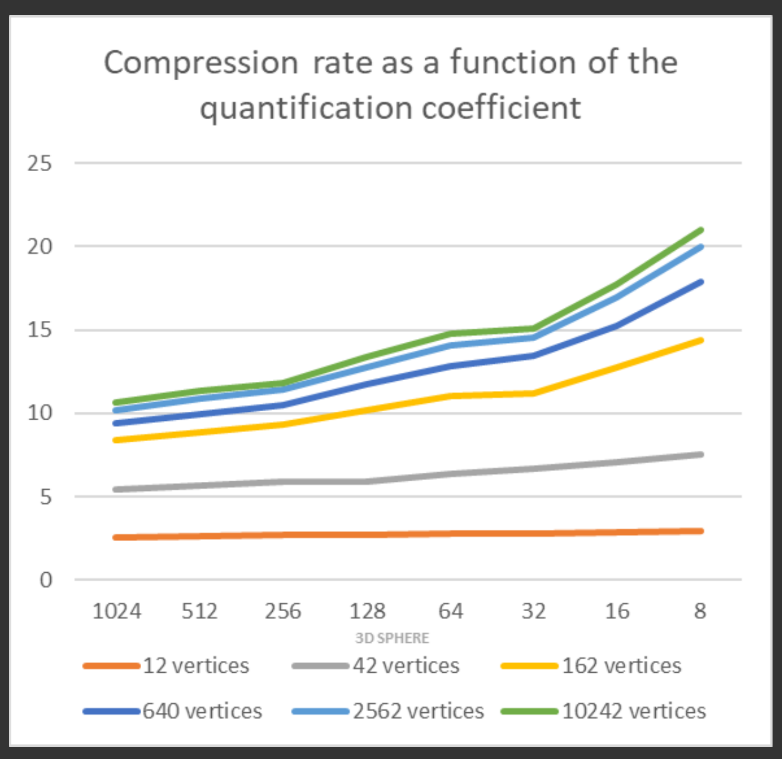
	NEW VERTEX 1			NEW VERTEX 2			NEW VERTEX 3	
X	Y	Z	X	Y	Z	X	Y	Z
145	78	256	365	354	2	47	145	142

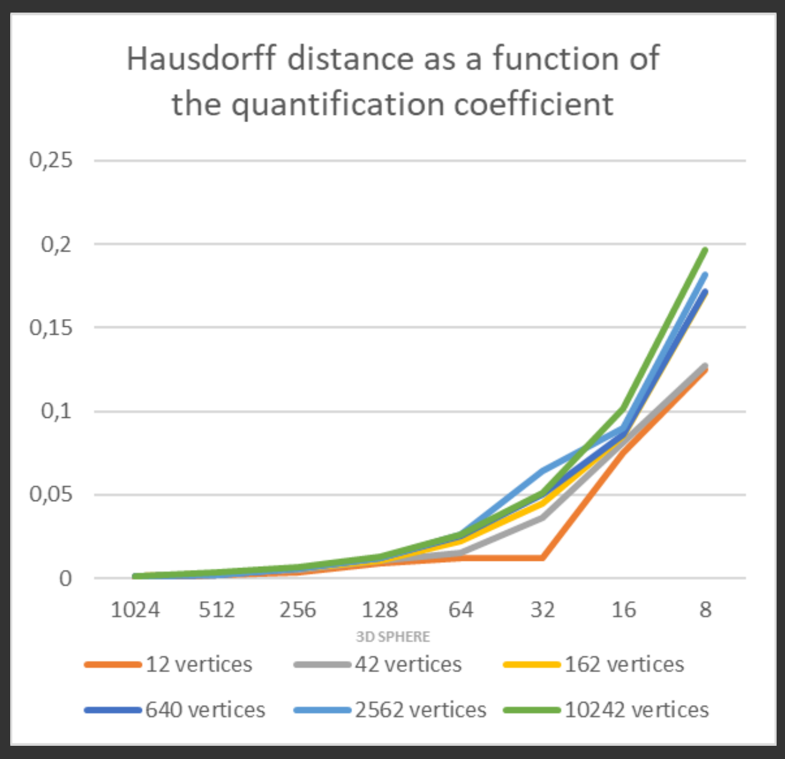
HUFFMAN ENCODING



The Huffman encoding algorithm builds a binary tree in which each leaf represents an element of the message to encode. The tree will have a shape such that the leaves corresponding to the most frequent elements will be located near the root, while the very infrequent elements will require a complex path from the root to access them. Therefore, the more frequently an element appears, the shorter its code will be when encoded.

RESULTS


Compression rate as a function of the quantification coefficient
3D SPHERE
12 vertices
42 vertices
162 vertices
640 vertices
2562 vertices
10242 vertices


Hausdorff distance as a function of the quantification coefficient
3D SPHERE
12 vertices
42 vertices
162 vertices
640 vertices
2562 vertices
10242 vertices