# Progressive Transmission of 3D Models

## Compression and Progressive Decoding of 3D Models

September 2025

**Project Objective** The aim of this project is to progressively transmit a 3D model represented by an `OBJ` file. We will focus only on geometric data (meshes). An adapted representation will be developed by *inverting* state-of-the-art compression algorithms.

**Project Progress** The project will be carried out in **groups of 4 students**. It will be presented during the second course of Compression, Streaming, Video Interactions **(Friday, September 26)**. Each group will then need to review the papers available on Moodle and choose one out of four categories of papers to implement.

A **Midterm Evaluation** will take place on **Friday, October 24**, during which each group must present: i) *the chosen paper*, ii) *their progress in implementing the algorithm*. It is expected that the paper's content will already be mastered, the implementation planned, and the division of work within the group defined.

The **final evaluation** will be held **between 10 - 21 November (TBD)** , in the form of a presentation before the entire class, covering: i) *the chosen paper*, ii) *the implementation*, iii) *the results obtained*.

**Expected Work** You must develop an algorithm that transforms an `OBJ` file into an `OBJA` file (Augmented OBJ). This format will allow progressive transmission and visualization of the 3D object by a client. The objective is to achieve the best *bitrate vs. distortion tradeoff*. The code must be submitted online via Moodle **no later than the day before the final presentation**. Additionally, an online evaluation of your performance will be available throughout your development to test the efficiency of your encoding: `http://csi-benchmark.mooo.com`

**Provided Code** The provided code is available at here. It includes:

- `decimate.py`: an example script for naïve decimation of a 3D model,
- `/example`: example `obj` and `obja` files generated with `decimate.py`,
- `server.py`: the visualization client,
- `obja.py`: Python parser for OBJ models and OBJA model creation.

# 1 Transmission of a 3D Mesh

A first solution that transmits a naïve (single-resolution) OBJA file is provided (`decimate.py`). The implementation and documentation are available here. The provided code extends the OBJ format into OBJA, allowing a triangular mesh to be represented and transmitted progressively.

The goal is therefore to develop a **progressive and compact** representation. The file will be transmitted progressively in packets, so that the object is displayed as the data is received. You may verify and visualize your algorithms with the client available in the Git repository (`server.py`).

You can evaluate: i) the quality of your simplified model (compression efficiency), ii) its progressiveness, using the online evaluation site: `http://csi-benchmark.mooo.com`.

# 2 Decompression and Progressive Model

To develop a progressive 3D model, you should base your work on existing **3D mesh compression algorithms**, which allows a 3D object to be represented at multiple levels of detail.

Several works are proposed, and each group must choose a different paper (or set of papers). Suggested options include:

- *Progressive meshes*, with decimation criteria to select (4 papers),
- *Squeeze & CPM*, papers by Pajarola and Rossignac (2 papers),
- *Arbitrary triangular meshes*, paper by Cohen-Or and Levin (1 papers),
- *Lossless transmission*, paper by Alliez and Desbrun (1 papers),
- *MAPS, remeshing*, paper by Lee et al. (1 papers).

Please refer to the bibliography details, **form groups of 4**, and **register on the Inscription document**. The papers can also be retrieved online (via N7) in the folder.

By inverting the compression process, you will define **refinement operations** for the model, sent progressively to the server as a sequence of `obja` instructions. The Python library `obja.py` is provided to help invert the compression scheme (Code). It includes: i) a Python OBJ parser, ii) tools for managing indices (vertices and faces), useful for element deletion.