Datamoshing

Et pourquoi pas en temps réel ?

Mode 1

Mode 2

Cette fois c'est le fond (les pixels de couleur) qui est continuellement mis-à-jour et le masque de déformation ne change que de temps en temps. Vous pouvez ajuster la fréquence de mise-à-jour du masque de déformation en modifiant la constante REFRESH_INTERVAL.

Mode 2 (cliquer pour afficher le code)

datamoshing_2.pde

```
import processing.video.*;

//
// PARAMETERS
//
float START_DISPLACEMENT = 800.0;
float SPEED = 2.5;
int REFRESH_INTERVAL = 15000;  // in milliseconds
boolean INVERT_COLORS = false;


Capture video;
PVector[] vectorMap;
PImage display;
PImage source_img;
int source_x, source_y;
int index;
float amp;
int last_update;


void setup() {
  size(1024, 768);
  video = new Capture(this, width, height);
  video.start();
  while (!video.available()) {
    delay(100);
  }
  video.read();
  vectorMap = new PVector[video.pixels.length];
  updateDisplacementMap(vectorMap, video);
  display = createImage(width, height, RGB);
  source_img = video.copy();
  amp = START_DISPLACEMENT;
  last_update = millis();
}


void updateDisplacementMap(PVector[] vector_map, PImage map_img) {
  map_img.loadPixels();
  float x_off, y_off;
  for (int j=0; j<height; j++) {
    for (int i=0; i<width; i++) {
      index = i + width*j;
      color displacementPix = map_img.pixels[index];
      // Use red channel for horizontal displacement
      // and green channel for vertical displacement
      x_off = -0.5 + (displacementPix >> 16 & 0xFF) / 255.0;
      y_off = -0.5 + (displacementPix >> 8 & 0xFF) / 255.0;
      vector_map[index] = new PVector(x_off, y_off);
    }
  }
}

void draw() {
  if (video.available()) {
    video.read();
    if (millis() - last_update > REFRESH_INTERVAL) {
      // Update vectorMap
```

```
      updateDisplacementMap(vectorMap, video);
      last_update = millis();
      amp = START_DISPLACEMENT;
    }
    if (INVERT_COLORS) video.filter(INVERT);
  }

  index = 0;
  for (int j=0; j<display.height; j++) {
    for (int i=0; i<display.width; i++) {
      source_x = round(amp * vectorMap[index].x + float(i));
      source_y = round(amp * vectorMap[index].y + float(j));
      while (source_x < 0)
        source_x += display.width;
      while (source_x >= display.width)
        source_x -= display.width;
      while (source_y < 0)
        source_y += display.height;
      while (source_y >= display.height)
        source_y -= display.height;

      display.pixels[index] = video.pixels[display.width*source_y + source_x];

      index++;
    }
  }
  display.updatePixels();
  image(display, 0, 0);

  amp += SPEED;
}

void mouseClicked() {
  saveFrame("pic-###.png");
}
```