MicroProject for Lecture 11-12

Jarek Rossignac
www.gvu.gatech.edu/~jarek

1 - Objective

Learn how to implement an animation of a subdivision surface and how to use it to warp an image.

2 - Deadline

It should be submitted on T-Square before the beginning of Lecture 13. No extensions.

3 - Deliverable

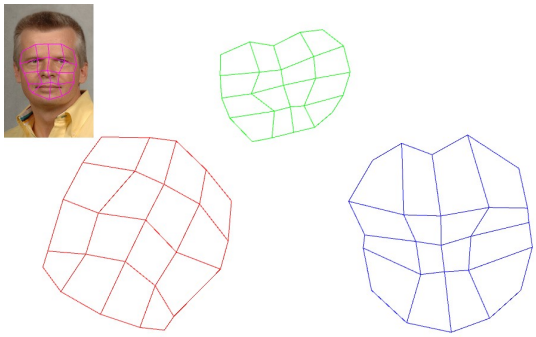
You may work in a team of 2 or individually if you prefer. All members of the team must contribute to the design and implementation of all aspects of the project (no splitting of tasks).

4 - Assignment details

4.1 User's experience

I suggest that you use the photographs of the team members (instead of my picture) for the screen shots and the video. Best would be if you alternate these for teams of two, so that the face of each one appears in some portion of the video.

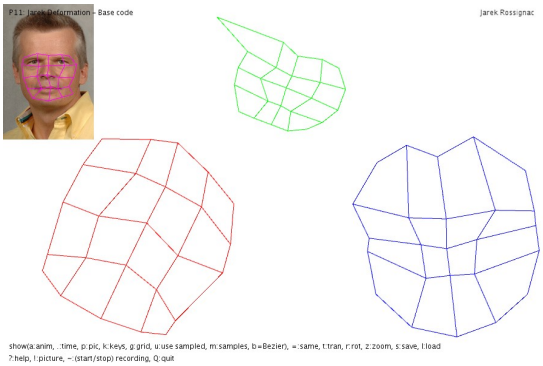
At start up, the system should show a **source image** of your choice as background and four 5x5 **control grids** (drawn in magenta, red, green, and blue) over the image in places where they do not overlap.



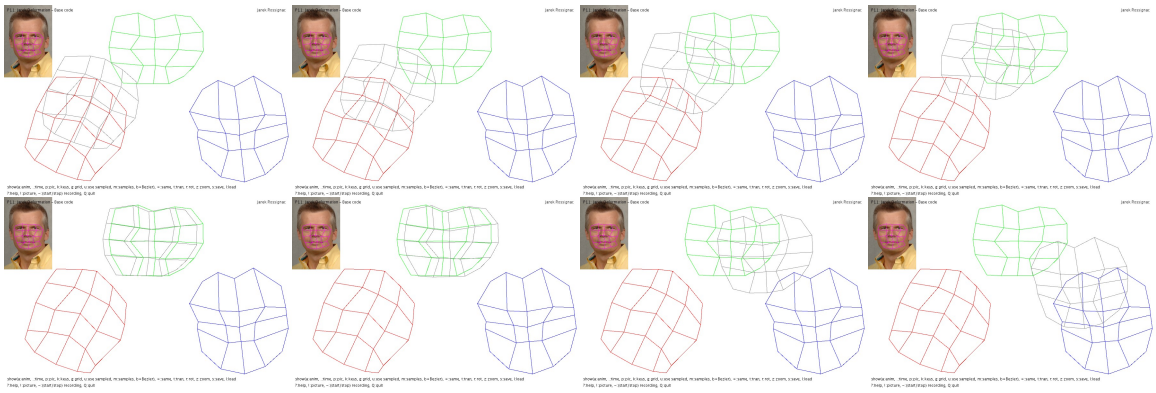
The user should be able to select any one of the four control grids by pressing '0', '1', '2', or '3'.

Pressing '=' when a non-zero grid is selected should make that grid identical to grid zero (magenta).

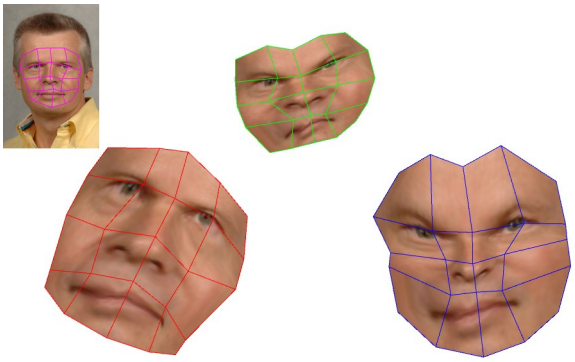
The points of the selected control grid should be editable as usual with the mouse: individually or as a group (rotation, translation, scaling).



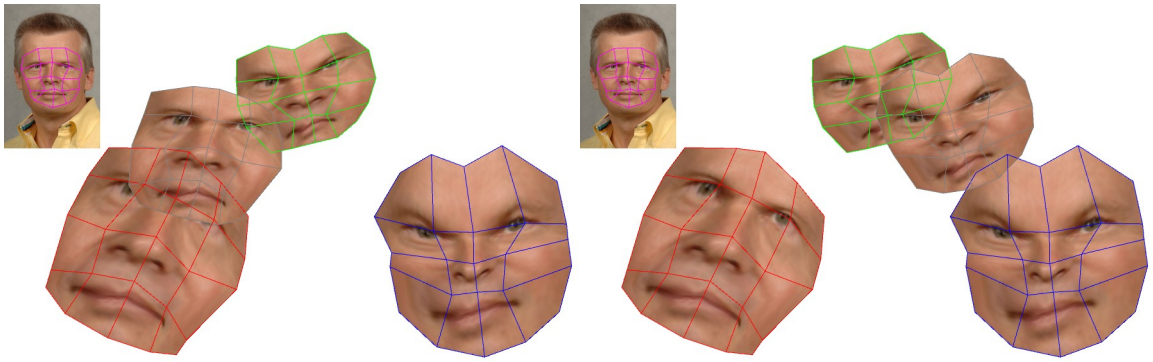
Once the 3 control grids are properly positioned, oriented, scaled, and deformed, we are ready to animate the deformation. Pressing ‘a’ starts the animation, which should last about 4 seconds. The animation shows a moving grid in grey. It starts with exactly as grid 1, then moves and deforms smoothly so that it interpolates grid 2 midcourse and ends exactly as grid 3. During the animation of the grey control grid, the position of its vertices (nodes) must be computed using **Neville’s interpolation** of the corresponding control vertices in the red, green, and blue control grids. Hence, the position of the vertices of the grey grid changes over time during animation, starting at the red grid, passing through the green one, and ending at the blue one. During animation, your code should advance time by a small amount so that the whole animation lasts about 3 or 4 second. For the current value t of time, use the Neville’s algorithm to compute the current positions of the vertices of the grey control grid.



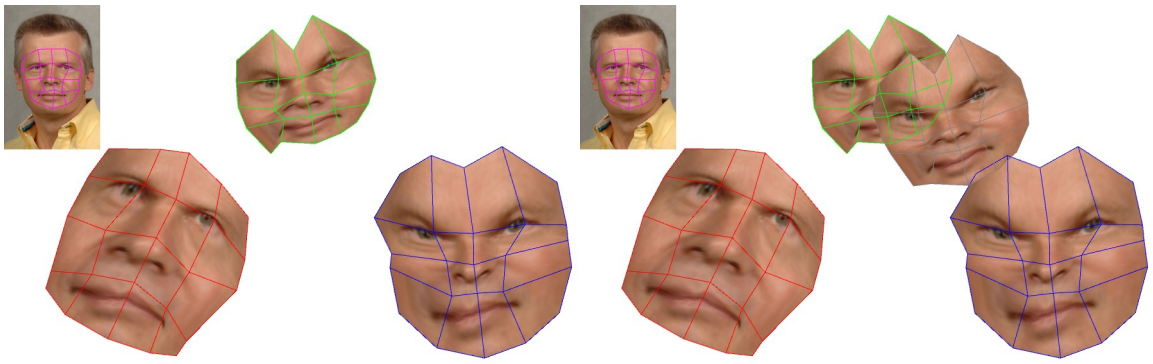
Pressing ‘p’ should toggle on/off the display of the texture. The texture should be the one under grid 0. It’s distorted version should be displayed for the other 3 keyframes. For each quad of the selected grid, use the coordinates of the vertices of the corresponding quad of grid 0 as texture coordinates and display the textured quad. This will produce a distorted copy of the portion of the source image that is covered by the quads of control grid zero. In your video, show the user translating, rotating, scaling, and tweaking the locations of individual vertices of the red grid while the distorted image is shown. Hence, the user is editing the initial warp.



During animation, when the texture is displayed, it should also be displayed for the moving frame.



By pressing ‘1’, ‘2’, or ‘3’, the user should be able select the grid again and modify it to change the key frame and hence the animation.



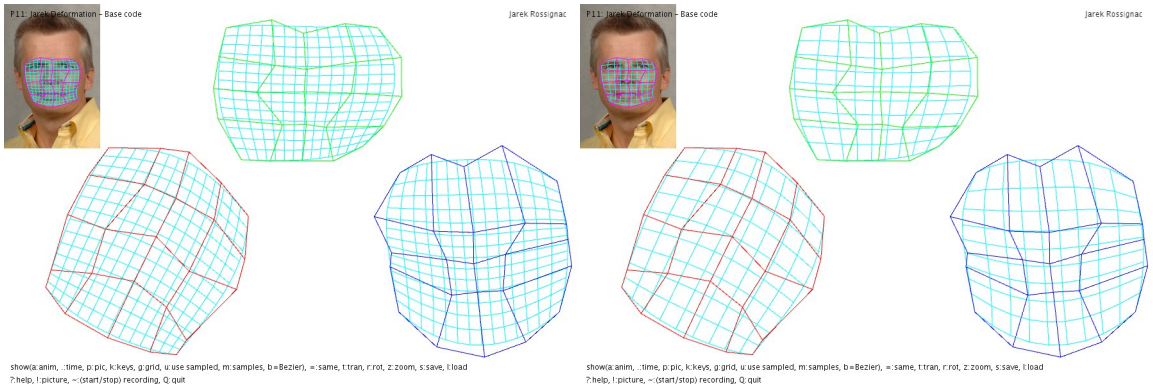
Pressing ‘k’ should toggle the display of the keyframes. Pressing ‘g’ should toggle whether the grids lines are displayed or no (including the grey and magenta ones).

Your video should show the user editing each one of the four grid, using control grid 0 to define the desired portion of the image (**cut-out**) to be used for animation. For example, in a picture of a face, you may use grid 0 to cut out the mouth or an eye or a portion of the face.

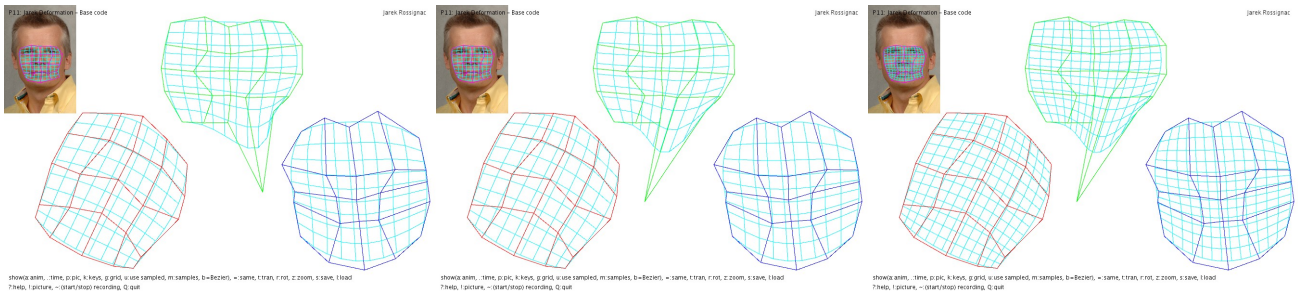
Make sure that you first display the source image, then the textures, and then the grids. This way, if you press ‘=’ to make all grids equal to grid 0 and then select grid 2 and edits some if the interior grid points, then press ‘k’ and ‘g’ to not show the keyframes nor the grids, the animation will display a temporary warp of the original image (try using this to display a smile, frown, or wink).

4.2 Required choices/technologies for the implementation the approach

You should use a **bi-parametric Bezier** formulation of a surface to produce a **finer version of each grid** that contains mxm points. The user should be able to adjust the sampling rate m by pressing ‘+’ and ‘-’. Show the fine grid in cyan and let the user toggle its display by pressing ‘m’.



The row and column of the vertices of the quads of the fine grid follow the parametric lines of the surface. In your video, you should show the fine grid while the user is editing a grid and also refining the sampling rate.

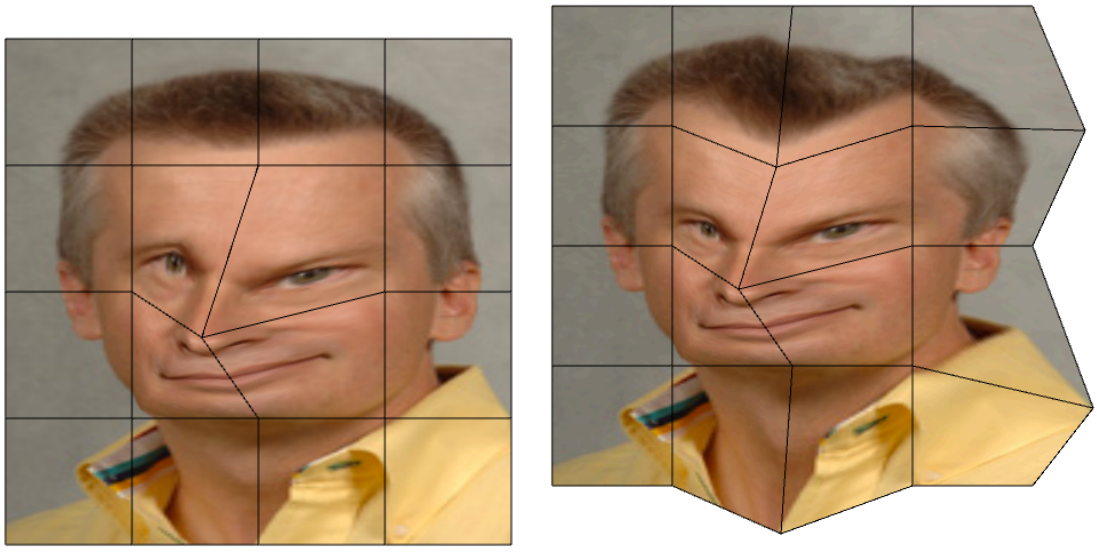


By pressing ‘u’, the user should be able to switch between using the 5x5 control grid and the finer sampled grid for texture mapping. Using the Bezier grid (below right) should produce smoother results.

Your video should show the animation once with the control grids, so that we can verify that the moving grey grid is indeed interpolating the red, green, and blue control grids at the proper times. Then show the animation once without the grids nor the keyframes.

4.3 Suggested approach

The base code provided has support for arranging the points of a pts object into a grid and for displaying that grid. It also supports editing that object as variable pts P. It also display the texture mapped to the quads of a unique (non-subdivided) control grid, **assuming that grid 0 is a regular (axis-aligned) grid over the image**. Hence, it can produce images as the one below, where triangulation artifacts are seen because the quads are too large. Your project solves the artifact problem via Bezier subdivision and also produces an animation.



It took me about less than a hundred lines to implement the required functionality, starting from the base code. Much of the new code deals with user interface functions (Boolean state variables indicating what should be displayed and whether we should be using the coarse grid or the refined one, key presses to toggle these variables and to change the sampling rate).

I declared 11 pts objects, two per color (the coarse grids Gi and the refined ones RGi) and P which points to the selected grid.

```
pts G0 = new pts(), G1 = new pts(), G2 = new pts(), G3 = new pts(), Gt = new pts(); pts P=G0;
pts RG0 = new pts(), RG1 = new pts(), RG2 = new pts(), RG3 = new pts(), RGt = new pts();
int levels=6, grid=5;
float t=0, f=0, s=0;
Boolean animate=false, fill=true, showIDs=false, timing=false;
Boolean showKeyFrames=true, showGrids=true, showResampled=true, useSampled=false; //#09
int samples=15; // number of samples for resampling
```

I create a longer canvas and initialize these pts objects.

```
void setup() {
  size(900, 600,P3D);
  myImage = loadImage("data/pic.jpg");
  G0.declare(); G0.makeGrid(grid);
  G1.declare(); G1.makeGrid(grid);
  G2.declare(); G2.makeGrid(grid);
  G3.declare(); G3.makeGrid(grid);
  Gt.declare(); Gt.makeGrid(grid);
  G0.loadPts(path+"0");
  G1.loadPts(path+"1");
  G2.loadPts(path+"2");
  G3.loadPts(path+"3");
  RG0.declare();
  RG1.declare();
  RG2.declare();
  RG3.declare();
  RGt.declare();
  }
```

I added the loadPts instructions only after I have saved the 4 files, using the ‘s’ key, as shown below:

```
if(key=='s') {G0.savePts(path+"0"); G1.savePts(path+"1"); G2.savePts(path+"2"); G3.savePts(path+"3"); }
if(key=='l') {G0.loadPts(path+"0"); G1.loadPts(path+"1"); G2.loadPts(path+"2"); G3.loadPts(path+"3"); P=G0;}
```

I added several actions to keyPressed()

```
if(key=='0') {P=G0; fill=false;} if(key=='1') {P=G1;} if(key=='2') {P=G2;} if(key=='3') {P=G3;}
if(key=='a') {animate=true; t=0;}
if(key=='u') useSampled=!useSampled;
if(key=='b') useBezier=!useBezier;
if(key=='k') showKeyFrames=!showKeyFrames;
if(key=='g') showGrids=!showGrids;
if(key=='m') showResampled=!showResampled;
if(key=='=') {G0.copyInto(G1); G0.copyInto(G2); G0.copyInto(G3); }
if(key=='p') fill=!fill; // used for timing
if(key=='#') G0.makeGrid(grid);  // resets regular grid
if(key=='+') samples++;  // increment number of samples for resampling
if(key=='-') samples=max(2,samples-1);  // decrement number of samples for resampling
```

In draw(), I added some 30 lines which compute the resampled RGi grids from the corresponding Gi grids.

```
void draw() {      // executed at each frame
  background(white); // clear screen and paints white background
  noFill();
  image(myImage, 0,0);
  G0.resampleTo(RG0,grid,samples);
  G1.resampleTo(RG1,grid,samples);
  G2.resampleTo(RG2,grid,samples);
  G3.resampleTo(RG3,grid,samples);
```

I also wrote the resampleTo() method for the pts class. I did not try to make it general: my implementation only supports situations where the original grid is 5x5. You may do the same, or you may implement a generalization to nxn for extra credit. You need to write your own implementation of the quartic Bezier: pt B(pt A, pt B, pt C, pt D, pt E, float t) {}, but that is one line of code.

I also added code in draw() that decides whether and how the keyframes textures are to be shown:

```
  if(showKeyFrames) {
    if(fill) {noStroke();
      if(useSampled) {
        RG1.paintImage(grid,RG0);
        RG2.paintImage(grid,RG0);
        RG3.paintImage(grid,RG0);
        }
      else {
        G1.paintImage(grid,G0);
        G2.paintImage(grid,G0);
        G3.paintImage(grid,G0);
        }
      }
```

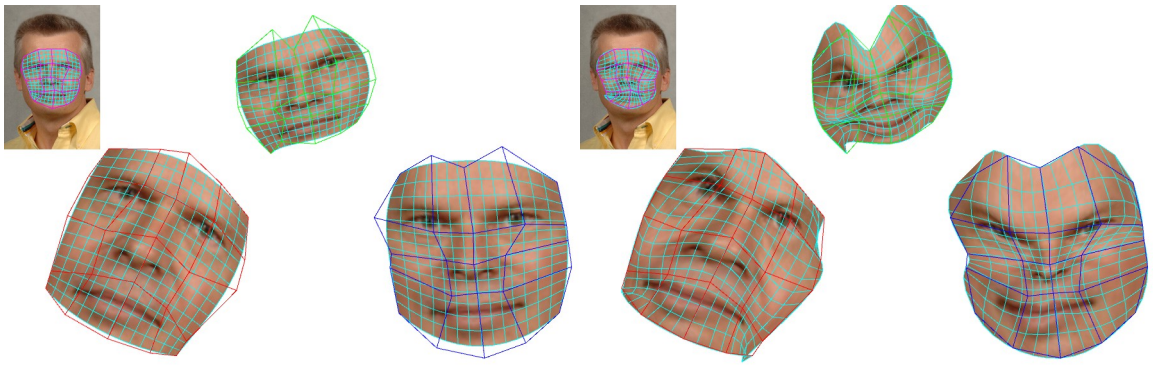
I also added to draw() code for computing the grey grid, for resampling it, and for displaying it with texture and/or grid lines (original or refined).

```
if(animate || (keyPressed && key=='.')) {
  Gt.interpolate(G1,G2,G3,t);
  Gt.resampleTo(RGt,grid,samples);
  noStroke(); if(fill) if(useSampled)  RGt.paintImage(grid,RG0); else  Gt.paintImage(grid,G0);
  pen(cyan,1);  if(showResampled) RGt.showGrid(grid);
  pen(grey,1); if(showGrids) Gt.showGrid(grid);
  }
if(animate) {t+=0.006; if(t>=2) animate=false;}
```

I used Neville for 3 points, assuming that the times associated with them are 0, 1, and 2 (that is why the code above stops animation when t>=2.

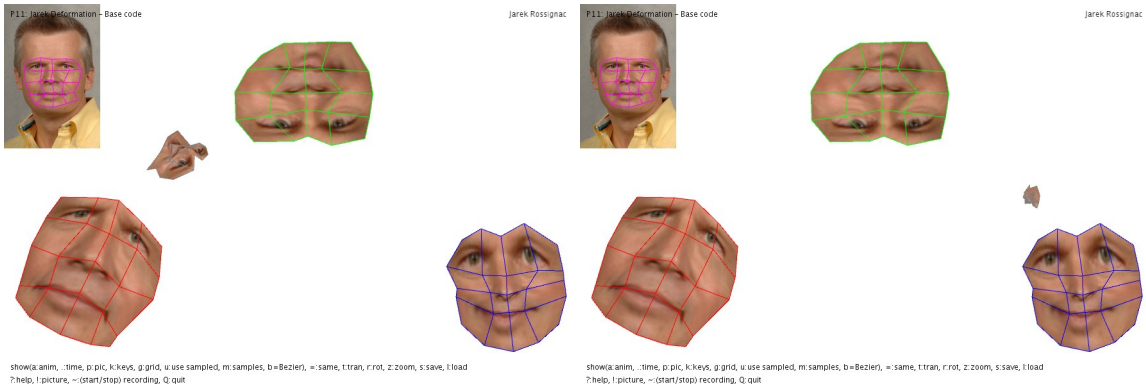
4.4 Extra credit suggestions

Provide the user with a key ('b') to toggle between the Bezier approximation (below, left) and the Neville interpolation (right), when computing the finer grid. I like the Bezier for its smoothness, but it filters out the sharp angles in the mesh and hence may be less expressive.



Use a four-point interpolation (rather than the parabolic trajectories of the Neville's algorithm) to create a continuous (cyclic) animation loop that interpolates the red, green, and blue control grids. Add a fourth control grid.

Extra credit for grad students: When there is a significant change of orientation (relative rotation angle close to PI) between consecutive keyframes, the interpolation produces horrible results. Show this effect, design and implement fix, and show a video of the corrected interpolation.



4.5 Submission

The usual report, zipped code, and short video in a file P11-12_ParticipantsNames.zip.

Make sure that your image file exists in the data folder. Please make sure that the zip file uploaded onto T-square actually contains all materials and that unzipping it produces a sketch that can be executed and a video that works. The TAs will no longer try to fix file problems for you and you may not get any credit if your video or sketch is not working or if you sipped, by mistake, the wrong version of one of these files.

If you implement extra credit, please submit a separate video labeled with "ExtraCredit" followed by the last names of the team members.