CAe 2019 P3

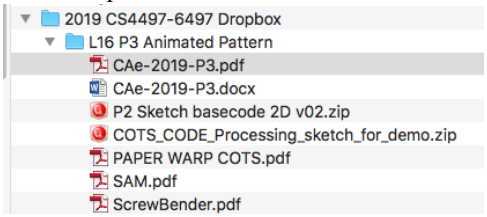
Animated COTS map of animated pattern

ABSTRACT

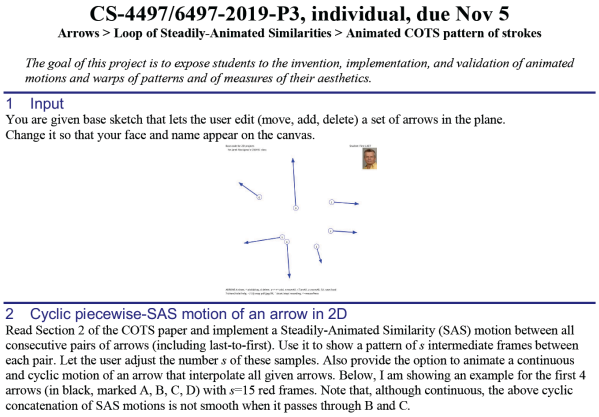
Here are the steps that I followed to implement the first part of the project.

1 Get the project description, plan your work, and decide on your LIVE PATTERN

From Dropbox



The file CAe-2019-P3.pdf Contains the project description.



CS-4497/6497-2019-P3, individual, due Nov 5

Arrows > Loop of Steadily-Animated Similarities > Animated COTS pattern of strokes

The goal of this project is to expose students to the invention, implementation, and validation of animated motions and warps of patterns and of measures of their aesthetics.

1 Input

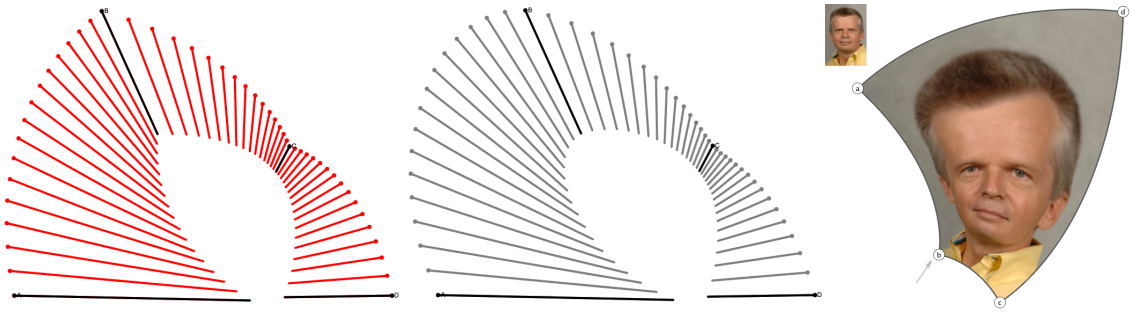
You are given base sketch that lets the user edit (move, add, delete) a set of arrows in the plane. Change it so that your face and name appear on the canvas.

2 Cyclic piecewise-SAS motion of an arrow in 2D

Read Section 2 of the COTS paper and implement a Steadily-Animated Similarity (SAS) motion between all consecutive pairs of arrows (including last-to-first). Use it to show a pattern of s intermediate frames between each pair. Let the user adjust the number s of these samples. Also provide the option to animate a continuous and cyclic motion of an arrow that interpolate all given arrows. Below, I am showing an example for the first 4 arrows (in black, marked A, B, C, D) with s=15 red frames. Note that, although continuous, the above cyclic concatenation of SAS motions is not smooth when it passes through B and C.

…

It explains the input and defines 3 clearly prescribed milestones (Phases A, B, C) for all students:



Consider, in your first attempt, replacing the first two by LERPS between consecutive arrows and replacing the COTS map by a bilinear map. This will be your "trilinear" solution and you will demonstrate the aesthetics benefits of the:

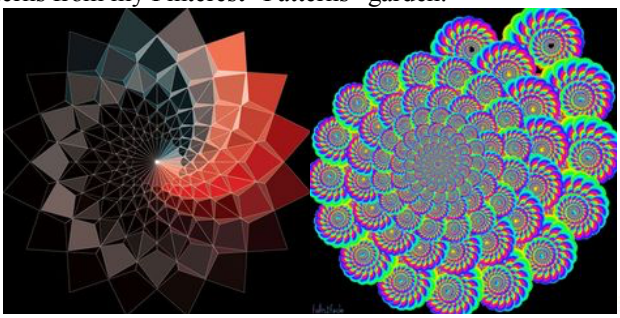
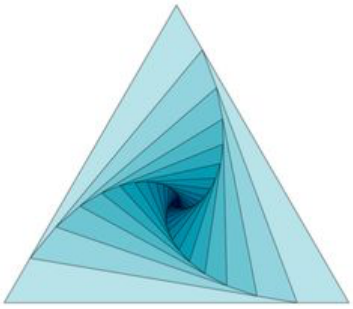
- Phase A: Use Similarity Steady Morphs between consecutive arrows instead of LERPS
- Phase B: Use several steps of quintic B-spline subdivision of the cycle of key-arrows to increase smoothness
- Phase C: Replace the bilinear map by a COTS map to even the texture map distortion

Do not forget to use YOUR face (not mine) as the image for texture mapping.
Do not forget to close the loop (add morph between last and first arrow), so that the image travels and warps in a continuous and infinite loop.

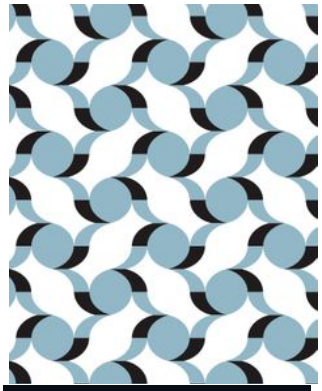
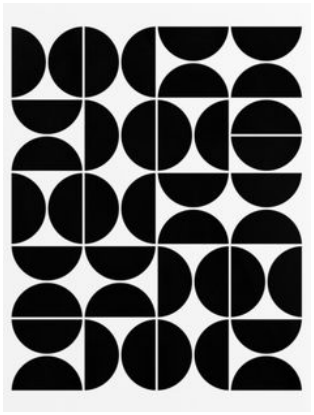
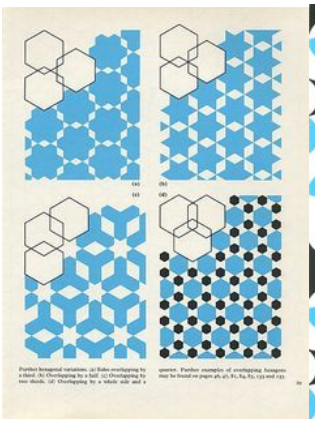
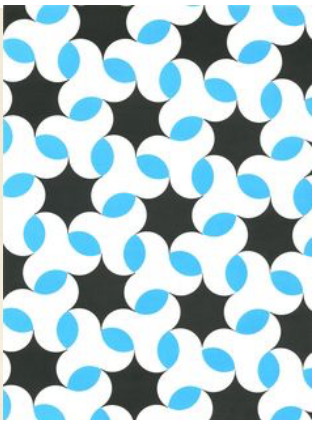
Phase D provides an open ("creativity") milestone for grad students and for extra credit (for those who need it) and a way of showing off your creativity and/or modeling/implementation skills. Here is the idea.

To paint the Bilinear or the COTS warp of your picture, you display the warping of each square tile, one at a time. To do so, you compute the images of the four vertices of that tile by the bilinear or by the COTS map. I have provided you with the code of these two maps. In Phase D, instead of warping the corners of each tile (or of the grid of tiles, if you want to avoid computing the image of each interior tile-corner 4 times), you will warp a set of control-points, which your code will use to produce a cool pattern.

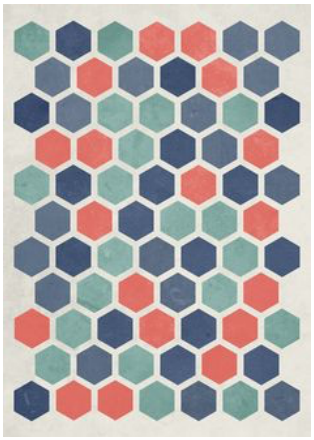
Here are some examples of cool patterns from my Pinterest “Patterns” garden.

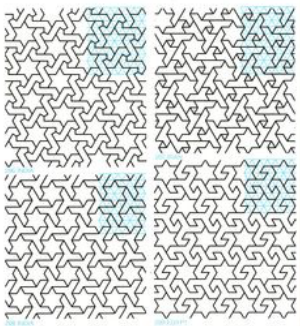
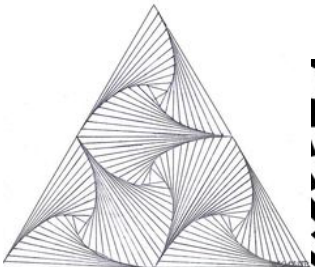
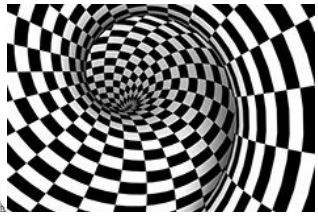




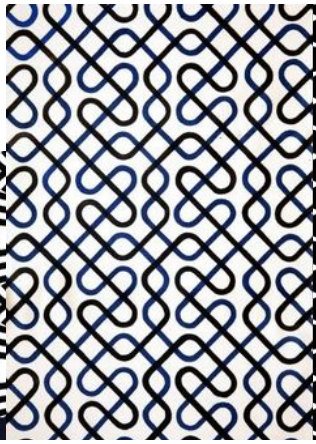


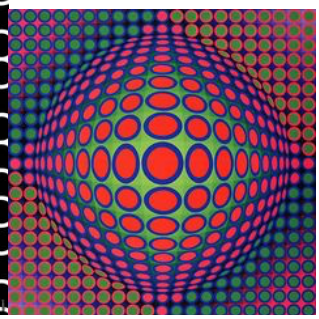
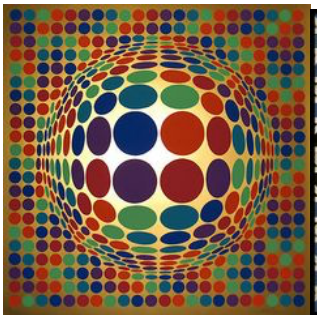






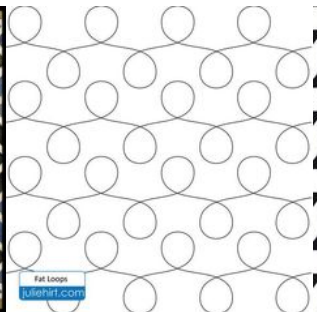


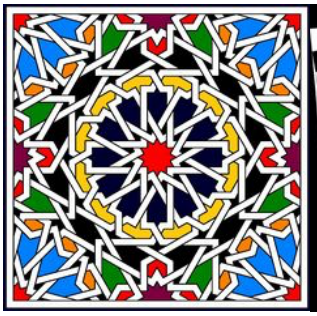
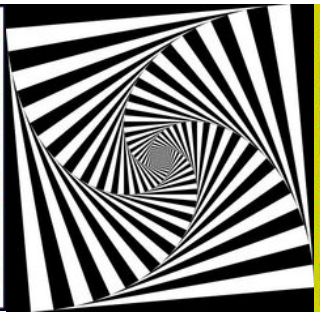
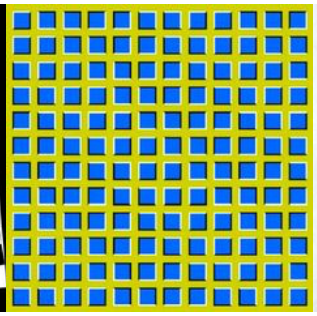
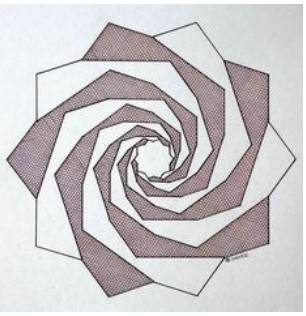
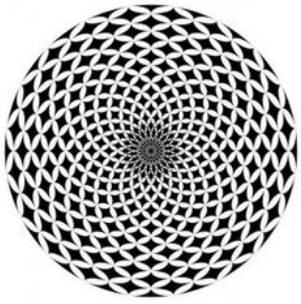
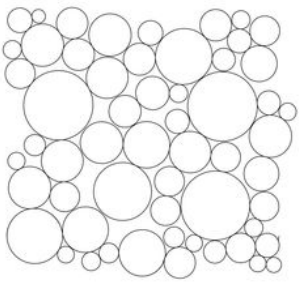


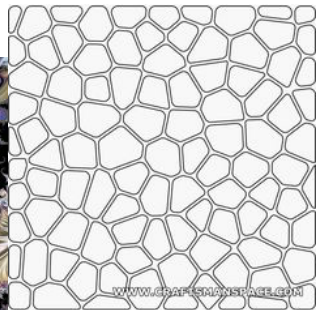




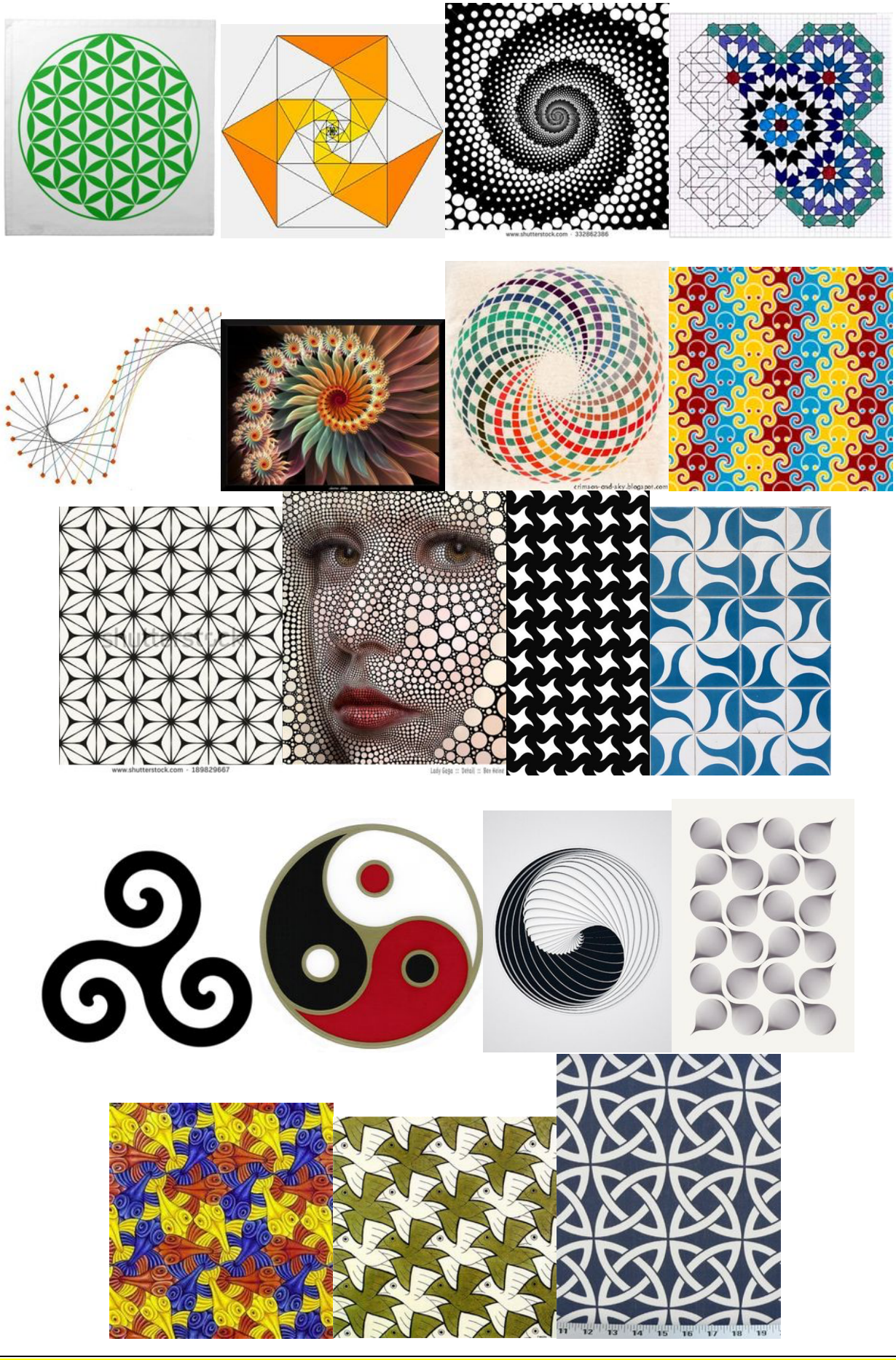

VectorStock


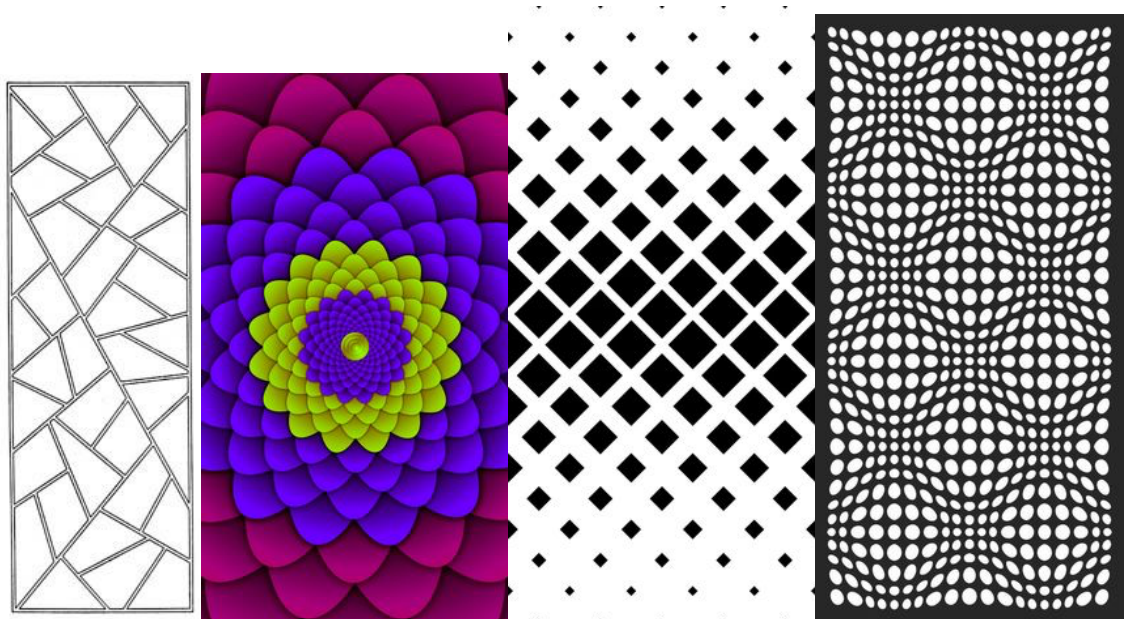



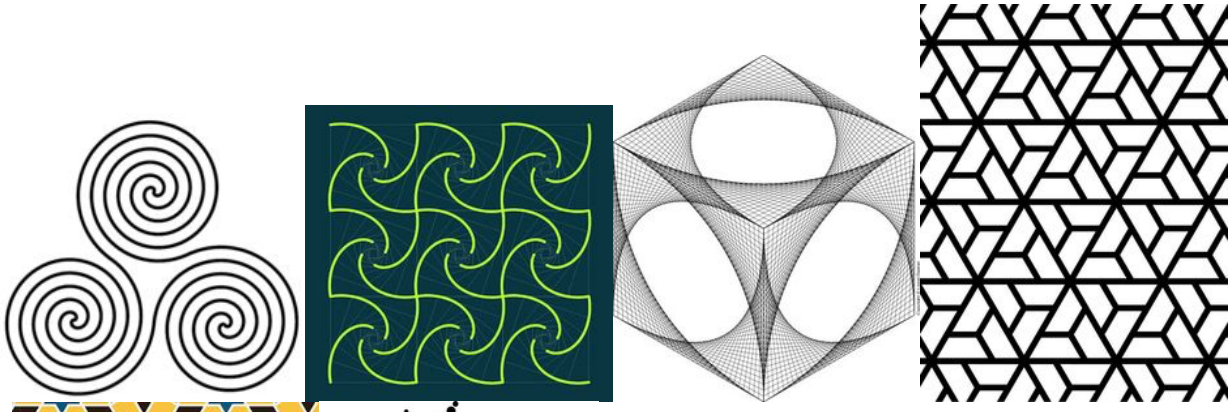

Fat Loops
juliehirt.com

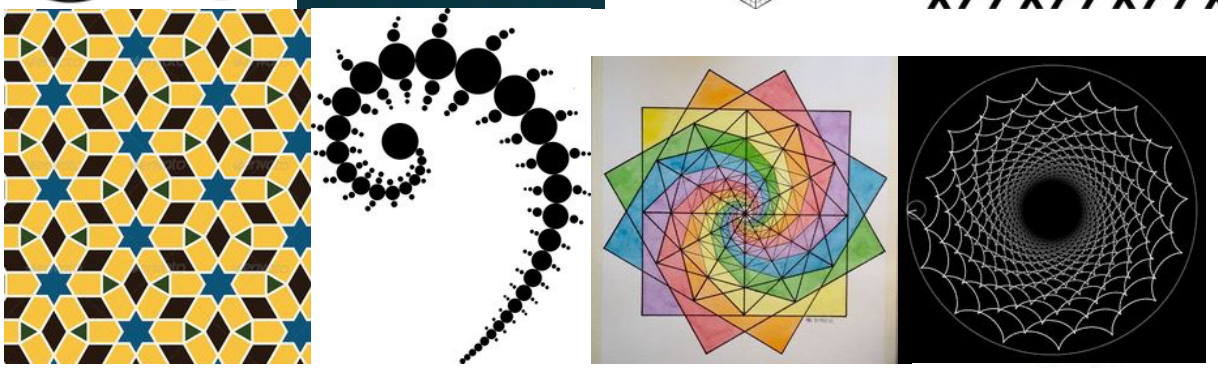







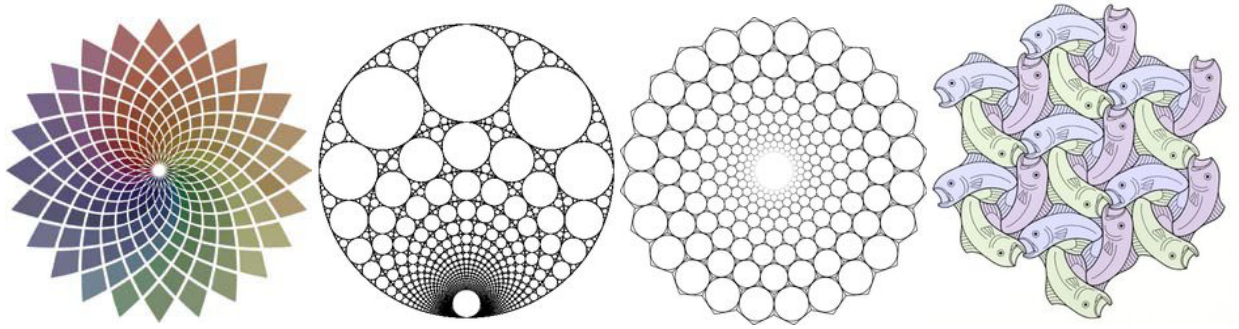

WWW.CRAFTSMANSPACE.COM



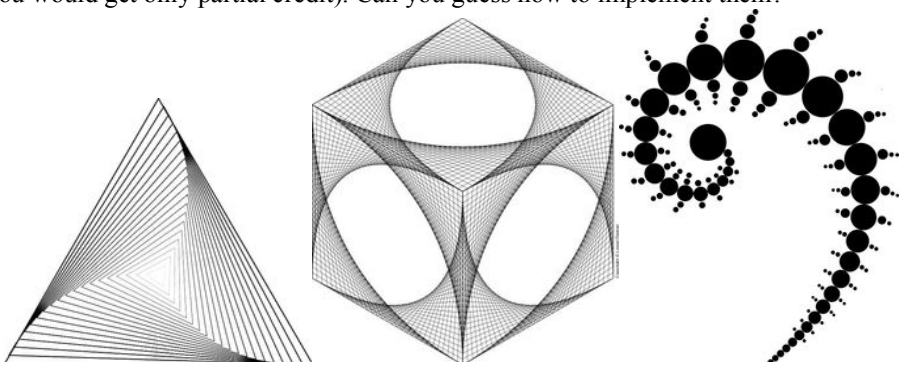




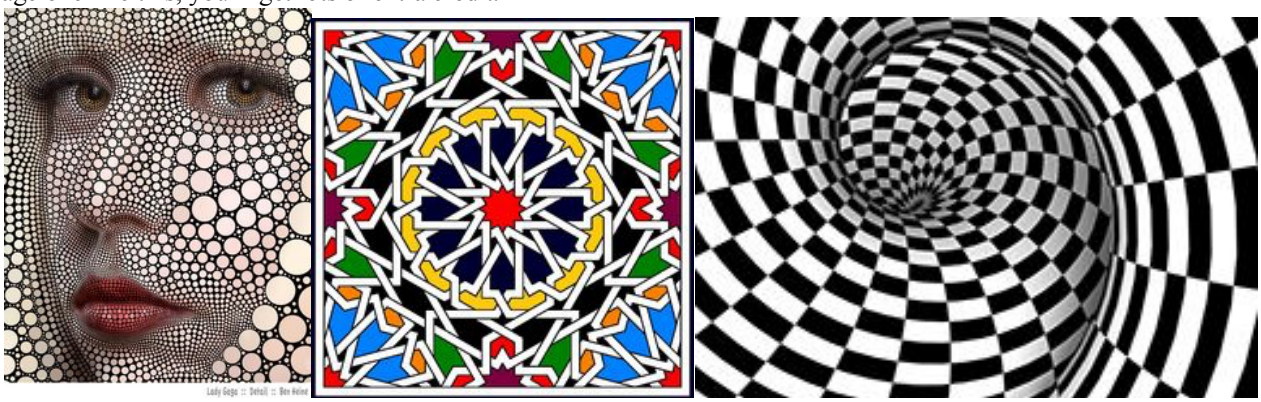




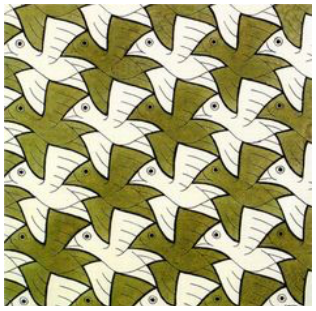
So, which one should you pick. To get full credit, you should be the only one (only team) to implement that pattern. It should be aesthetics and impressive by its complexity. It should not be obvious, at first glance, how to implement it. How to judge this? Suppose that we show it to the class : if 50% of the students guess a simple strategy for implementing it, then it is too easy. For example, these three are simple (hence you would get only partial credit). Can you guess how to implement them?



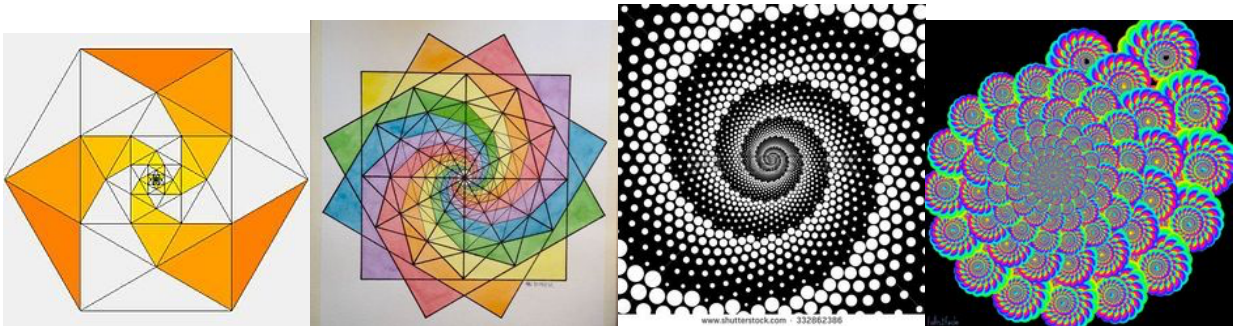
On the other hand, you want something for which you can invent a simple and elegant algorithm. These look really hard to me. So, if you manage one like this, you'll get lots of extra credit.



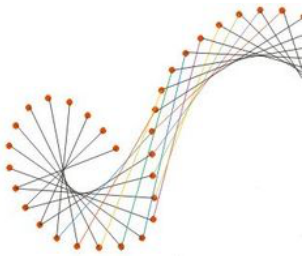
You may also have the user intervene to design your pattern (not all of it has to be "programmed"). Here are some examples. This may appeal to those of you who have artistic talents.



Note that many seem to use log spiral patterns and possible could be easily implemented using some variant of the COTS map. You will get full credit for these.



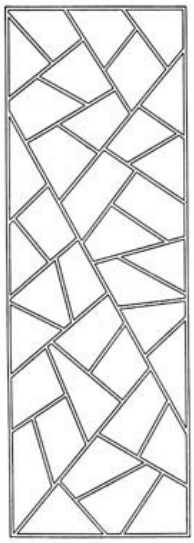
This one is a bit related to the animation produced by Phases A, B, and C. It may inspire a variant, which you may implement to get extra credit.



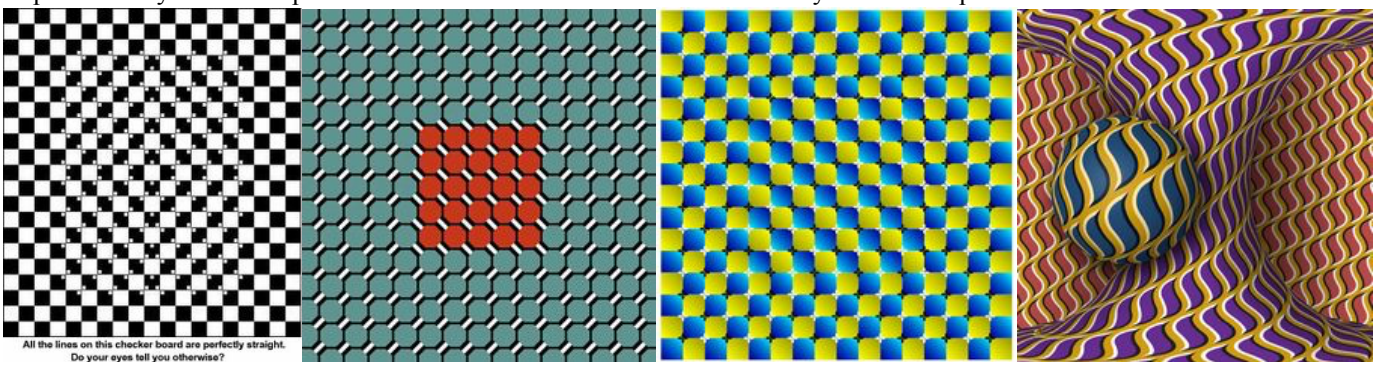
What about using code that you found online or wrote for other classes? We will accept this, provided that:

- You acknowledge clearly the source (with link and other details)
- You include the code inside your implementation and provide profuse and detailed comments
- You explain in your report how and why it works

Here are some examples, one of which is a fractal (I think) and another is a subset of project 2 for CS6491.



Some patterns may create an optical illusion or an illusion of motion. These may be fun to explore.



But, remember, this is your CREATIVITY moment. You do not have to implement any of these. You may do so, and enjoy the PROBLEM—SOLVING CREATIVITY: Invent a simple algorithm that makes one of these. Or you may instead conceive your own procedural pattern, implement it, and beautify it by trial and error.

Try to use an aesthetically pleasing color scheme derived from your project 1.

Here's the catch: Your patterns needs to be LIVE! By that, I mean, smoothly animated. For example, it could "pulse" with a period of 1 second. By pulse, I do not mean oscillate between two or three states, but change in a smooth manner so that consecutive frames of your animation are not too different from one another. Yet, the rate of change may vary through the period to create a pulsing feeling, like in dancing. So, when you select/design your pattern, think about its parameter(s) and how they will vary over time to create a continuous, smooth, pulsing motion.

To recap, this project has 3 complementary components:

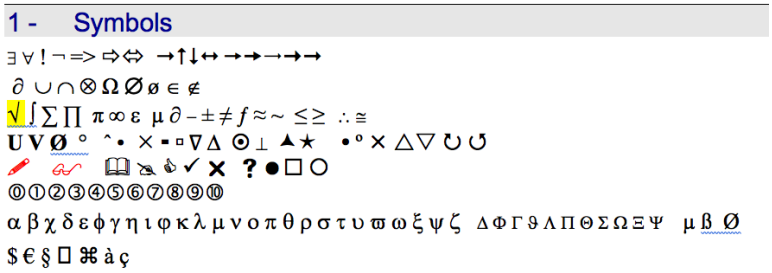
- Design and implement a LIVE pattern in the unit square ($[0,1]^2$).
- Map it to a quad using the bilinear and the COTS map and show that COTS is more aesthetics

- Animate the quad vertices using the quintic B-Spline subdivision of a cyclic list of control-arrows.

If you are not a great programmer, or do not have a strong math foundation, you may consider implementing only one or two of these components, or even implementing simplified versions (like LERP morph and bilinear map) of these, for partial credit.

2 Report

I also include the original, CAe-2019-P3.doc, file, which may be useful for you for formatting your report.



Please keep your report concise:
Title
Course reference
Author
Date

Brief abstract of what is accomplished, spelling clearly extra credit and phases that were not fully implemented.
Provide a separate section for each phase/component. In it, explain what is the input, how it is stored (array of arrows…), what you compute, how you store it, how you compute it (provide math using points and vectors) when not trivial. Show a few images produced by your implementation. Try to provide a brief explanation of why your solution works and, if needed, a disclaimer of when it does not work and why.

I suggest that, for each phase, you write the corresponding section BEFORE you start implementing it. Once you do implement it, you will add the pictures and may have to adjust the text, but writing (explaining) it first may help you anticipate problems or discover mistakes.

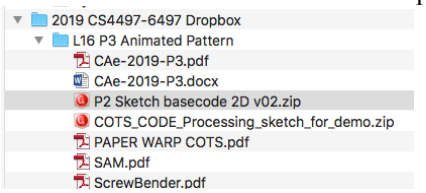
Please include references to prior work!

3 Part A: Similarity Steady Morph between arrows

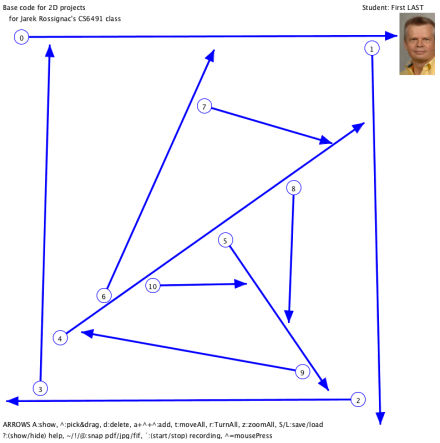
This is how I tackled it. You do not have to do the same, but please read anyway.

3.1 Base code

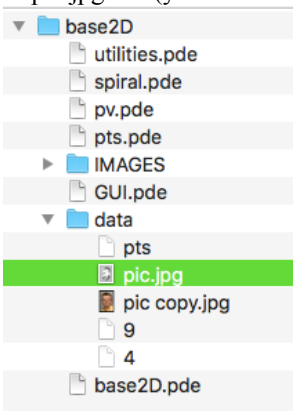
I downloaded the P2 sketch and decompressed it



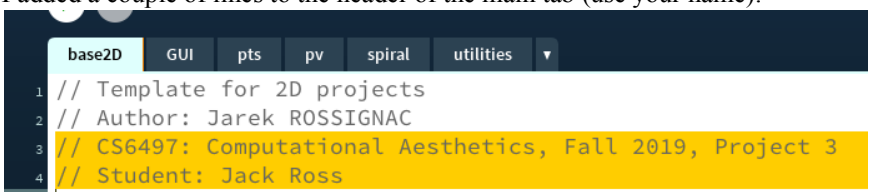
I run it:



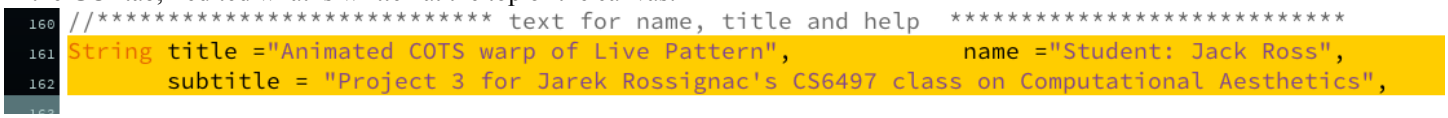
I renamed my picture (to keep a copy) and placed a new pic.jpg file (you should use one your smiley face) in the data folder.



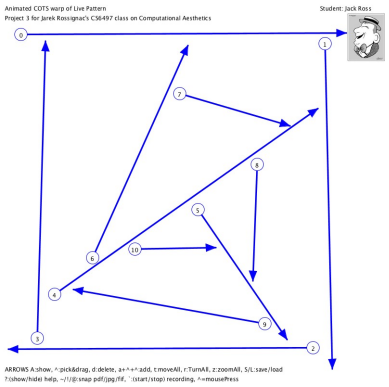
I added a couple of lines to the header of the main tab (use your name):



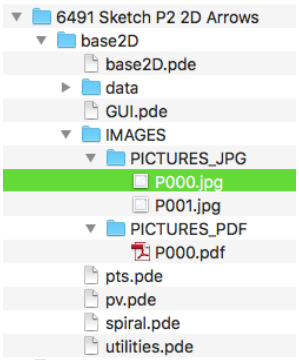
In the GUI tab, I edited what is written at the top of the canvas:



This is what I get when I run it:

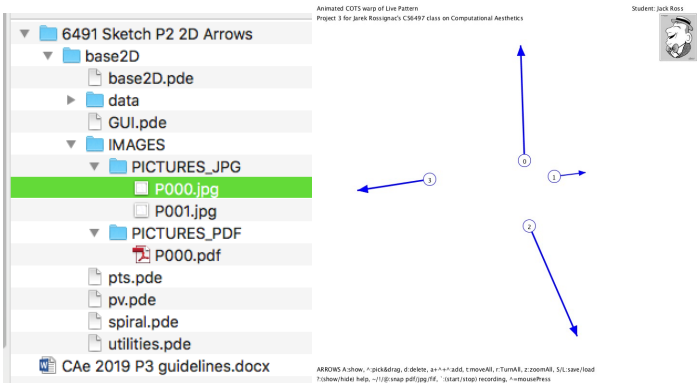


I pressed '!' to capture a .jpg file of the canvas.



I pasted it into this word doc file (see image above). This is how you can produce and use images for your report. If you do not restart your sketch, the file names are incremented, so that you can save several images without having to rename them.

The default set of control arrows has a lot of arrows. I deleted most of them by keeping 'd' pressed and clicking near the start of the arrow that I want to delete. Note that the provided code renumbers the arrows. Then, I arranged them into a simple , but non trivial circular pattern. This is easy. I just click and drag the closest arrow end-point. Keeping 't' pressed, and click-dragging, I can translate all the arrows. Do not forget to press 'S' to save the new configuration on file. Next time you start the sketch, it will be loaded automatically. I made a picture for you (using '!') and dragged it into my doc.



3.2 Class for Arrows

I decided that the code would be simpler if I have a class for Arrows.
So, I created a new tab named Arrows, copied the class declaration from tab pts

```
base2D | GUI | loopOfArrows | pts | pv | spiral | utilities
1  //*************************************************************************
2  // TITLE:         Point sequence for editing polylines and polyloops
3  // AUTHOR:        Prof Jarek Rossignac
4  // DATE CREATED:  September 2012
5  // EDITS:         Last revised August 10, 2017
6  //*************************************************************************
7  class pts
8    {
9    int nv=0;                              // number of vertices in the sequence
10   int pv = 0;                            // picked vertex
11   int iv = 0;                            // insertion index
12   int maxnv = 100*2*2*2*2*2*2*2*2;       //  max number of vertices
13   Boolean loop=true;                     // is a closed loop
14
15   pt[] G = new pt [maxnv];               // geometry table (vertices)
16
17  // CREATE
18
19
20   pts() {}
21
22   void declare() {for (int i=0; i<maxnv; i++) G[i]=P(); }              // creates all poir
```

and modified it as follows

```
base2D | Arrows | GUI | pts | pv | spiral | utilities
1  class ARROW
2    {
3    pt P = P();
4    vec V = V();
5    // CREATE
6    ARROW() {}
7    ARROW(pt Q, vec U) {P=P(Q); V=V(U);}
8    // SHOW
9    void show() {arrow(P,V);}
10   }
11
12 // ARROW FUNCTIONS
13 ARROW Arrow(pt Q, vec U) {return new ARROW(Q,U);}
14 void show(ARROW A) {A.show();}
```

For each arrow, I store a point P and a vector V.
I Initialize these to zero.
I had an error on line 4, so I went into the pv tab and added a function that creates a null vector:

```
base2D | Arrows | GUI | pts | pv | spiral | utilities
153 //*************************************************************
154
155 // create
156 vec V() {return new vec(0,0); };     // make zero vector
157 vec V(vec V) {return new vec(V.x,V.y); };
```

To test it, I added a couple of lines to the draw() in the base2D tab:

```
base2D | Arrows | GUI | pts | pv | spiral | utilities
void draw()      // executed at each frame
 {
 if(recordingPDF) startRecordingPDF(); // starts recording graphics

   background(white); // clear screen and paints white background


   if(showArrow) P.drawArrows(); // draws all control arrows

   ARROW A0 = Arrow(P.G[0],V(P.G[0],P.G[2]));
   stroke(red); fill(yellow); show(A0);
```

I used the start point of the first and second arrow, which are points 0 and 2 of the pt G[] array in the pts class

```
base2D | Arrows | GUI | pts | pv | spiral | utilities
class pts
  {
  int nv=0;                                // number of vertices in the seque
  int pv = 0;                              // picked vertex
  int iv = 0;                              // insertion index
  int maxnv = 100*2*2*2*2*2*2*2*2;         //  max number of vertices
  Boolean loop=true;                       // is a closed loop

  pt[] G = new pt [maxnv];                 // geometry table (vertices)
```

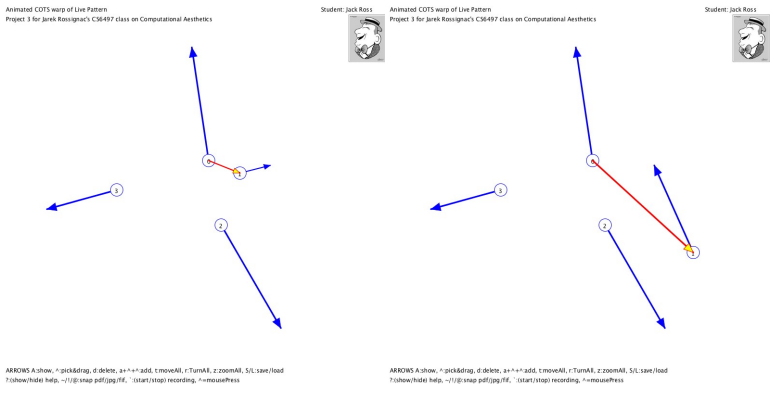
Note that in the base code, I was using the object P of type pts

```
base2D | Arrows | GUI | pts | pv | spiral | utilities
//***************************** global variables *****************************
pts P = new pts(); // class containing array of points, used to standardize GUI
```

to support editing of an array G[] of points and drawing the arrows between consecutive even and odd points of G[]:

```
base2D | Arrows | GUI | pts | pv | spiral | utilities
  void drawArrows()
    {
    stroke(blue);
    for (int a=0; a<nv/2; a++)
      {
      fill(blue); arrow(G[2*a],G[2*a+1]);
      fill(white);
      show(G[2*a],13);
      fill(black);
      if(a<10) label(G[2*a],str(a));
      else label(G[2*a],V(-5,0),str(a));
      }
    noFill();
```

I checked that this works by playing with the control points of



3.3 LERP Average between two arrows

Now I want to try and compute the average of two arrows. I will start with the simple LERP.
Searching for "average" in the pv tab, I found:

```
base2D | Arrows | GUI | pts | pv | spiral | utilities

// average
pt P(pt A, pt B) {return P((A.x+B.x)/2.0,(A.y+B.y)/2.0); };
```

To have the same for vectors, I copied

```
base2D | Arrows | GUI | pts | pv | spiral | utilities
// Interpolation
vec L(vec U, vec V, float s) {return new vec(U.x+s*(V.x-U.x),U.y+s*(V.y-U.y));};
```

And inserted two lines in:

```
base2D | Arrows | GUI | pts | pv | spiral | utilities
// average
vec V(vec U, vec V) {return new vec(U.x+(V.x-U.x),U.y+(V.y-U.y));};

// Interpolation
```

Where I removed the factor s.

In the Arrows tab, I added the skeleton of a LerpAverageOfArrows function

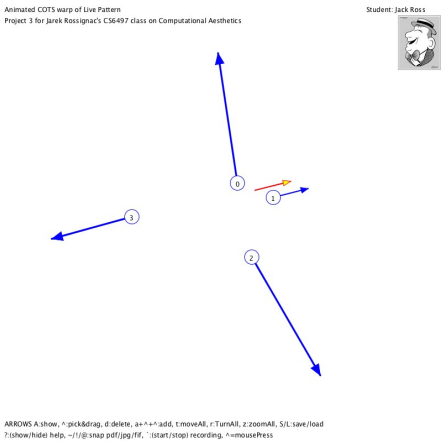
```
ARROW LerpAverageOfArrows(ARROW A0, ARROW A1) {return Arrow(P(),V());}
```

And then filled the missing details:

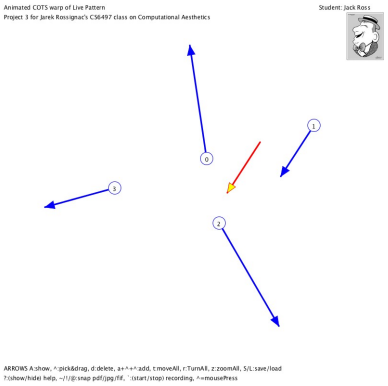
```
ARROW LerpAverageOfArrows(ARROW A0, ARROW A1) {return Arrow(P(A0.P,A1.P),V(A0.V,A1.V));}
```

To try it, I modified, in the base2D tab, the two lines that I had added in draw:

```
    if(showArrow) P.drawArrows(); // draws all control arrows

    ARROW A0 = Arrow(P.G[0],V(P.G[0],P.G[1]));
    ARROW A1 = Arrow(P.G[2],V(P.G[2],P.G[3]));
    ARROW Am = LerpAverageOfArrows(A0,A1);
    stroke(red); fill(yellow); show(Am);
```



That is not what I was expecting. So, I played with the control points of P a bit to make the problem more obvious and saved the configuration (press 'S'), so facilitate debugging.



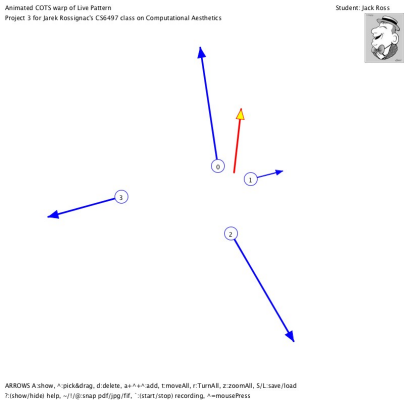
Clearly, the point average works, but the vector average does not.
I looked back in the pv tab at how I computed the vector average:

```
base2D | Arrows | GUI | pts | pv | spiral | utilities
// average
vec V(vec U, vec V) {return new vec(U.x+(V.x-U.x),U.y+(V.y-U.y));};

// Interpolation
```

Oh, I see! Just deleting "s*" was not a good idea. s should be ½ for the average. So, I changed that function to:

```
// average
vec V(vec U, vec V) {return new vec(U.x+(V.x-U.x),U.y+(V.y-U.y));};

// Interpolation
```

Better. Now it works



That incident reminds me to not be overconfident and to pay attention to details.

3.4 Spiral (Similarity-Steady) Average between two arrows

Now, I felt like trying to do the log-spiral average and compare them.
When I was looking for interpolations in the pv tab, I noticed this:

```
vec L(vec U, vec V, float s) {return new vec(U.x+s*(V
vec S(vec U, vec V, float s) // steady interpolation
  {
  float a = angle(U,V);
  vec W = R(U,s*a);
  float u = n(U), v=n(V);
  return W(pow(v/u,s),W);
  }
```

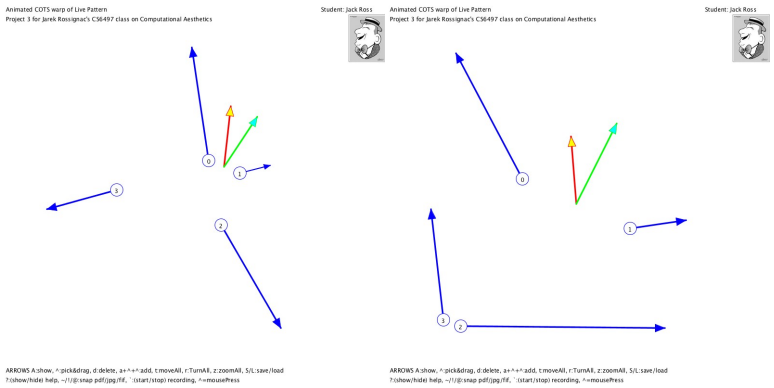
Which is the LERP that I want for the vector. I know that this is not the final LERP for the arrow, but I want to check it first.
So, in the Arrows tab, I added a new function:

```
ARROW LerpAverageOfArrows(ARROW A0, ARROW A1) {return Arrow(P(A0.P,A1.P),V(A0.V,A1.V));}
ARROW SteadyAverageOfArrows(ARROW A0, ARROW A1) {return Arrow(P(A0.P,A1.P),S(A0.V,A1.V,0.5));}
```

And added 2 lines in draw() in the base2D tab:

```
    ARROW Am = LerpAverageOfArrows(A0,A1);
    stroke(red); fill(yellow); show(Am);
    ARROW As = SteadyAverageOfArrows(A0,A1);
    stroke(green); fill(cyan); show(As);
```

Nice. It works. But this is not the final spiral average that I need. So I created a configuration showing that this is not what is needed and saved it.



To build the proper spiral morph, I looked in the spiral tab and found the spiralCenter function:

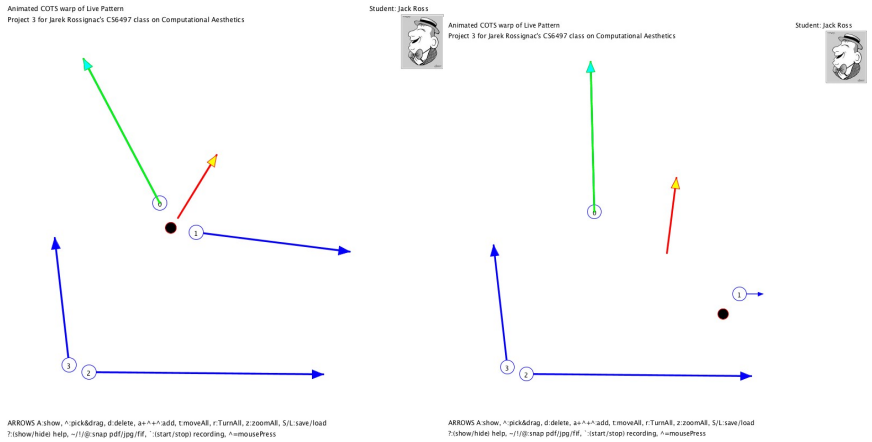
base2D | Arrows | GUI | pts | pv | spiral | utilities

```
pt spiralCenter(float a, float z, pt A, pt C)
  {
  float c=cos(a), s=sin(a);
  float D = sq(c*z-1)+sq(s*z);
  float ex = c*z*A.x - C.x - s*z*A.y;
  float ey = c*z*A.y - C.y + s*z*A.x;
  float x=(ex*(c*z-1) + ey*s*z) / D;
  float y=(ey*(c*z-1) - ex*s*z) / D;
  return P(x,y);
  }
```

I want to use it to replace the point LERP.
I modified mu SteadyAverageOfArrows to compute and display the fixed point F.

```
ARROW SteadyAverageOfArrows(ARROW A0, ARROW A1)
  {
  float w = angle(A0.V,A1.V);
  float m = n(A1.V)/n(A0.V);
  pt F = spiralCenter(w,m,A0.P,A1.P);
  fill(black); show(F,10);
  return(A0);
  }
```

I return A0 for now.
I tested it to ensure that F is where I think it should be: closer to the start of the smaller arrow, so that the angle between the vectors from F to the starts of the two arrows is the angle between the arrow directions.



Seems to work. Now, I need to compute the correct position of the start of the average arrow.
I could use the construction provided in the spiral tab:

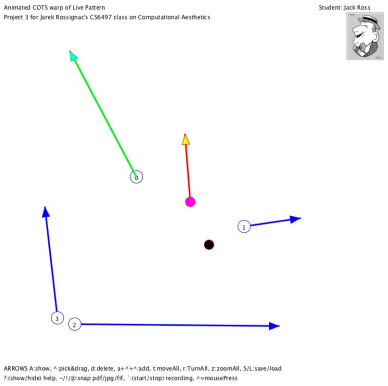
```
// IMAGE OF POINT Q BY SPIRAL MOTION THAT MORPHS EDGE(A,B) AND EDGE(C,D)
pt spiral(pt A, pt B, pt C, pt D, float t, pt Q)
  {
  float a =spiralAngle(A,B,C,D);
  float s =spiralScale(A,B,C,D);
  pt G = spiralCenter(a, s, A, C);
  return L(G,R(Q,t*a,G),pow(s,t));
  }
```

But, since I only want the average, I will try to be clever:

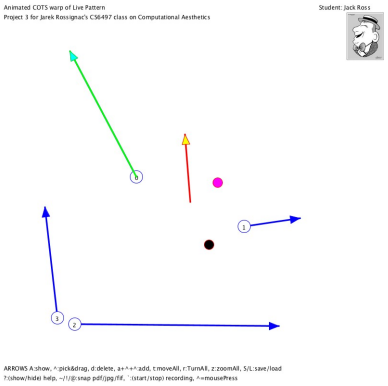
```
ARROW SteadyAverageOfArrows(ARROW A0, ARROW A1)
  {
  float w = angle(A0.V,A1.V);
  float m = n(A1.V)/n(A0.V);
  pt F = spiralCenter(w,m,A0.P,A1.P);
  fill(black); show(F,10);
  vec U = V(V(F,A0.P),V(F,A1.P));
  pt Q = P(F,U);
  fill(magenta); show(Q,10);
  return(A0);
  }
```

Can you figure out what I am trying to do here?
P(F,U) returns F+U
How do I compute vector U?

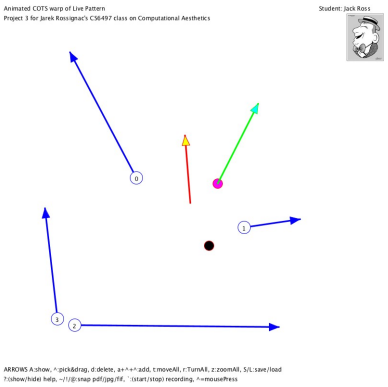
Is this correct?



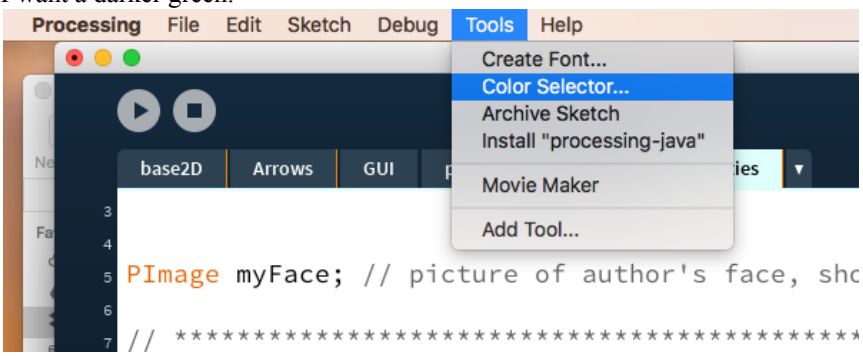
Good answer. This is wrong.
So, how to fix that?
Do it. (If you don't see how, review what we did earlier for interpolating vectors.)

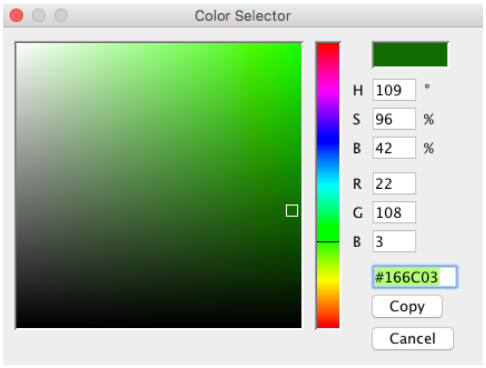


Now I can finish: use the V vector computed as before and compute the average arrow correctly:

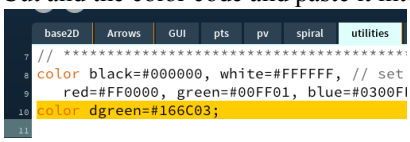


Let me remove the displays of points and adjust the colors.
I want a darker green:



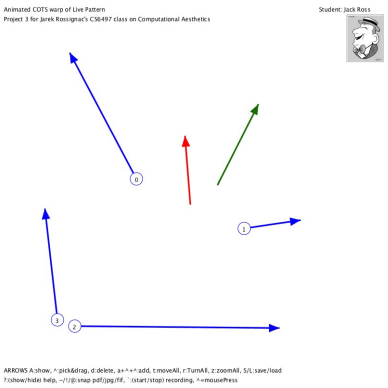


Cut and the color code and paste it into the utilities tab:



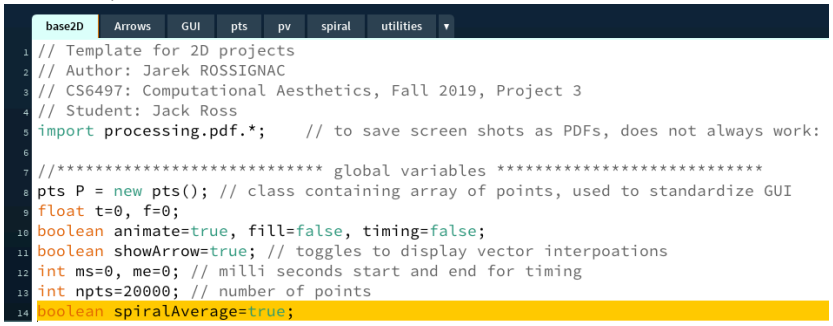
And use that color in draw() in the base2D tab:

```
    ARROW A0 = Arrow(P.G[0],V(P.G[0],P.G[1]));
    ARROW A1 = Arrow(P.G[2],V(P.G[2],P.G[3]));
    ARROW Am = LerpAverageOfArrows(A0,A1);
    stroke(red); fill(red); show(Am);
    ARROW As = SteadyAverageOfArrows(A0,A1);
    stroke(dgreen); fill(dgreen); show(As);
```



3.5 Switching between LERP and Spiral Average

To support each toggle between LERP and Spital Averaging of arrows, I did this:
In base2D, I added a Boolean:



In the GUI tab, I added a key to toggle it:

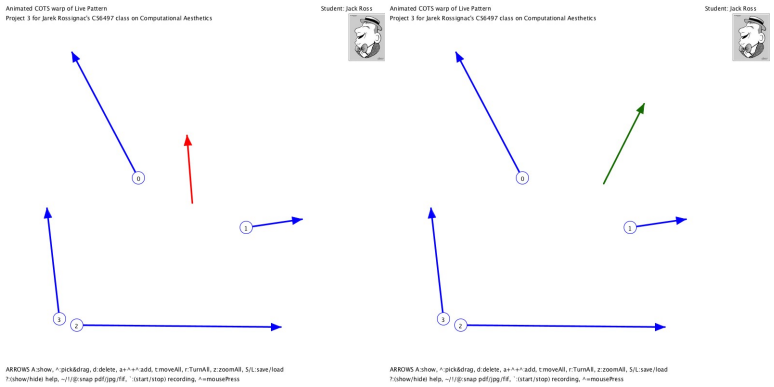
In the base2D tab, I modified the display:

```
39      ARROW A0 = Arrow(P.G[0],V(P.G[0],P.G[1]));
40      ARROW A1 = Arrow(P.G[2],V(P.G[2],P.G[3]));
41      ARROW Am = AverageOfArrows(A0,A1);
42      if(spiralAverage) {stroke(red); fill(red);} else {stroke(dgreen); fill(dgreen);}
43      show(Am);
```

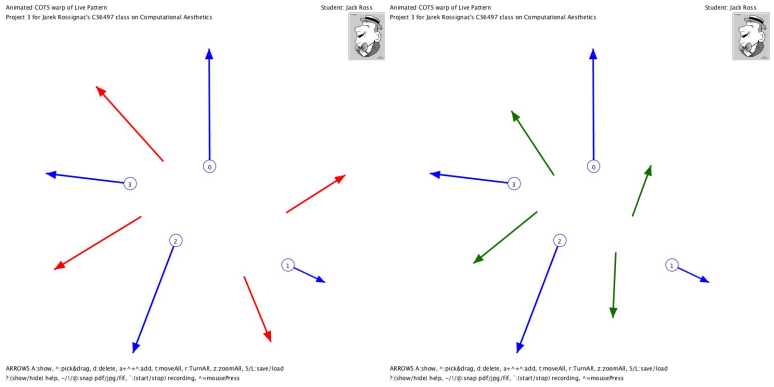
And in the Arrows tab, I added the AverageOfArrow function:

```
28  ARROW AverageOfArrows(ARROW A0, ARROW A1)
29    {
30    if(spiralAverage) return SteadyAverageOfArrows(A0,A1);
31    else return LerpAverageOfArrows(A0,A1);
32    }
```

Now, I can toggle between the default spiralAverage (red) and the LerpAverage (dark green), and this functionality will remain if I only use AverageOfArrows.



To conclude this first session. I decided to average all consecutive pairs of 4 arrows (showing the red and green options):



To do this, I commented out the old code I wrote in draw() in the base2D tab and added a new code.

```
38
39      //ARROW A0 = Arrow(P.G[0],V(P.G[0],P.G[1]));
40      //ARROW A1 = Arrow(P.G[2],V(P.G[2],P.G[3]));
41      //ARROW Am = AverageOfArrows(A0,A1);
42      //if(spiralAverage) {stroke(red); fill(red);} else {stroke(dgreen); fill(dgreen);}
43      //show(Am);
44
45      int i=0;
46      ARROW A0 = Arrow(P.G[i],V(P.G[i++],P.G[i++]));
47      ARROW A1 = Arrow(P.G[i],V(P.G[i++],P.G[i++]));
48      ARROW A2 = Arrow(P.G[i],V(P.G[i++],P.G[i++]));
49      ARROW A3 = Arrow(P.G[i],V(P.G[i++],P.G[i++]));
50
51      ARROW A01 = AverageOfArrows(A0,A1);
52      ARROW A12 = AverageOfArrows(A1,A2);
53      ARROW A23 = AverageOfArrows(A2,A3);
54      ARROW A31 = AverageOfArrows(A3,A0);
55      if(spiralAverage) {stroke(red); fill(red);} else {stroke(dgreen); fill(dgreen);}
56      show(A01); show(A12); show(A23);  show(A31);
57
```

Next session, I will make a class for a loop of arrows and will write and test functions to compute the two arrows (tweaked original and inserted) for the quintic B-Spline subdivision.

3.6 ARROWRING class

I created an ARROWRING class in the Arrows tab:

```
34 class ARROWRING
35   {
36   int nA=0;                                  // number of Arrows
37   int maxnA = 6*2*2*2*2*2*2*2;               //  max number of ARROWS
38   ARROW[] A = new ARROW [maxnA];                  // geometry table (vertices)
39   ARROWRING() {}
40   void declare() {for (int i=0; i<maxnA; i++) A[i]=new ARROW(); }
41   void addArrow(pt Q, vec U) { A[nA].P=P(Q);  A[nA].V=V(U); nA++;  }
42   void empty() {nA=0; }   // empties this object
43   void showArrows() {for(int a=0; a<nA; a++) show(A[a]);}
44   }
```

I declared a global object in the base2D tab:

```
14 boolean spiralAverage=true;
15 ARROWRING Aring = new ARROWRING();
16
17 //************************** initialization **************************
18 void setup()              // executed once at the begining
19   {
```

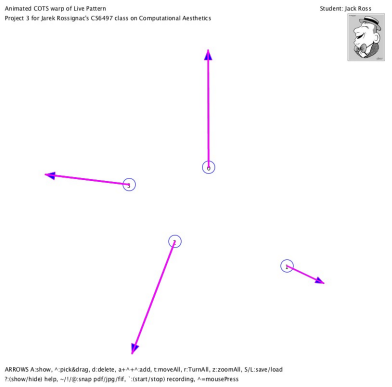
I call the declare method in setup:

```
27   P.loadPts("data/pts");  // loads points form file sa
28   Aring.declare();
29   } // end of setup
30
31 //************************** display current frame *
32 void draw()      // executed at each frame
33   {
```

I replaced part of main() as follows:

```
base2D | Arrows | GUI | pts | pv | spiral | utilities
//*************************** display current frame ***************************
void draw()      // executed at each frame
  {
  if(recordingPDF) startRecordingPDF(); // starts recording graphics to make a PDF

    background(white); // clear screen and paints white background

    if(showArrow) P.drawArrows(); // draws all control arrows

    Aring.empty();
    int i=0;
    Aring.addArrow(P.G[i],V(P.G[i++],P.G[i++]));
    Aring.addArrow(P.G[i],V(P.G[i++],P.G[i++]));
    Aring.addArrow(P.G[i],V(P.G[i++],P.G[i++]));
    Aring.addArrow(P.G[i],V(P.G[i++],P.G[i++]));
    stroke(magenta); Aring.showArrows();
```

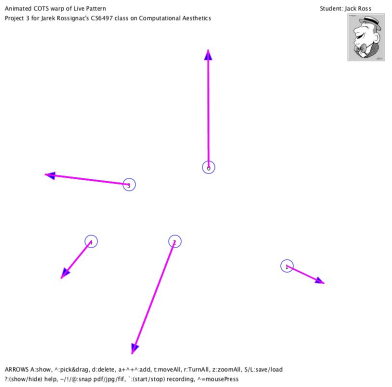
I check that it works



Now, I turn this into a loop to build the proper number of arrows.

```
    Aring.empty();
    for(int i=0; i<P.nv; i+=2) {Aring.addArrow(P.G[i],V(P.G[i],P.G[i+1]));}
```

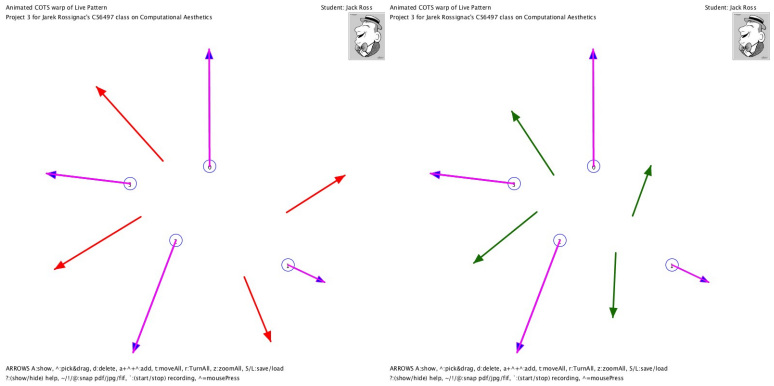
And check that it works:



Now, in the Arrows tab, I add a method to display averages of consecutive arrows. Later, I will transform that method into one that refines the ARROWRING. For this, I will use a previous and next methods.

```
34 class ARROWRING
35   {
36   int nA=0;                                  // number of Arrows
37   int maxnA = 6*2*2*2*2*2*2*2;               //  max number of ARROWS
38   ARROW[] A = new ARROW [maxnA];                  // geometry table (vertices)
39   ARROWRING() {}
40   void declare() {for (int i=0; i<maxnA; i++) A[i]=new ARROW(); }
41   void addArrow(pt Q, vec U) { A[nA].P=P(Q);  A[nA].V=V(U); nA++;  }
42   void empty() {nA=0; }   // empties this object
43   void showArrows() {for(int a=0; a<nA; a++) show(A[a]);}
44   void showAverageArrows()
45     {
46     for(int a=0; a<nA; a++)
47       show(AverageOfArrows(A[a],A[n(a)]));
48     }
49   int n(int a) {return (a+1)%nA;}
50   int p(int a) {return (a+nA-1)%nA;}
51   }
```

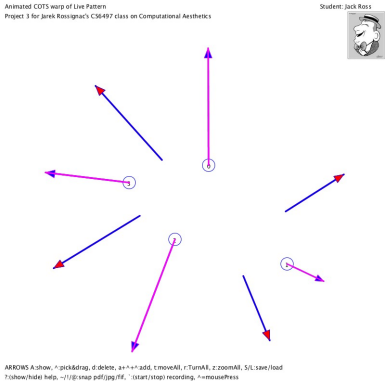
I check that it works for the spiral average (and also for the LERP average, pressing ‘s’)



To make sure that the previous function works, I added a version using it and make sure that they overlap;

```
44   void showAverageArrows()
45     {
46     for(int a=0; a<nA; a++)
47       show(AverageOfArrows(A[a],A[n(a)]));
48     stroke(blue);
49     for(int a=0; a<nA; a++)
50       show(AverageOfArrows(A[a],A[p(a)]));
51     }
52   int n(int a) {return (a+1)%nA;}
53   int p(int a) {return (a+nA-1)%nA;}
```

They do, Yey!



3.7 Refining the ARROWRING

Now I want to compute a new ARROWRING A2 that is the refined version of an ARROWRING A1.
I add a method in ARROWRING to add an arrow, B:

```
41   void addArrow(pt Q, vec U) { A[nA].P=P(Q);  A[nA].V=V(U); nA++;  }
42   void addArrow(ARROW B) { A[nA].P=P(B.P);  A[nA].V=V(B.V); nA++;  }
43   void empty() {nA=0; }   // empties this object
```

I also add a refineInto method that builds a second arrow passed as parameter:

```
  int n(int a) {return (a+1)%nA;}
  int p(int a) {return (a+nA-1)%nA;}
  void refineInto(ARROWRING A2)
    {
    A2.empty();
    for(int a=0; a<nA; a++)
      {
      A2.addArrow(A[a]);
      A2.addArrow(AverageOfArrows(A[a],A[n(a)]));
      }
    }
```

To test it; I create a second (Refined) arrowing in base2D:

```
ARROWRING Aring = new ARROWRING();
ARROWRING RevinedAring = new ARROWRING();
```

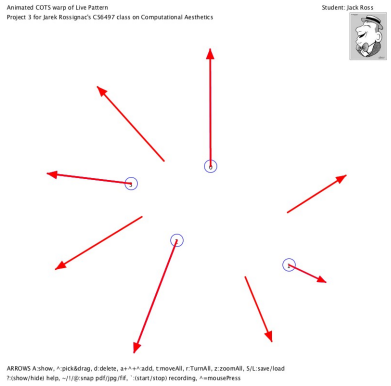
I declare it at the end of setup():

```
  Aring.declare();
  RevinedAring.declare();
  } // end of setup
```

And I compute and display it in draw():

```

    Aring.refineInto(RefinedAring);
    if(spiralAverage) {stroke(red); fill(red);} else {stroke(dgreen); fill(dgreen);}
    RefinedAring.showArrows();
    //Aring.showAverageArrows();

```

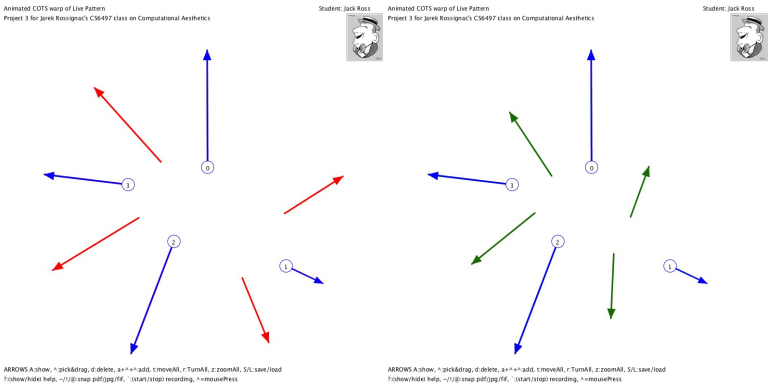
I check that it works



It seems to, but I no longer see the original key arrows, so I will flip the order of display in draw(), to display the arrow from P last, so that I also see the arrow numbers:

```
//***************************** display current frame *****************************
void draw()      // executed at each frame
  {
  if(recordingPDF) startRecordingPDF(); // starts recording graphics to make a PDF
    background(white); // clear screen and paints white background
    Aring.empty();
    for(int i=0; i<P.nv; i+=2) {Aring.addArrow(P.G[i],V(P.G[i],P.G[i+1]));}
    //stroke(magenta); Aring.showArrows();
    Aring.refineInto(RefinedAring);
    if(spiralAverage) {stroke(red); fill(red);} else {stroke(dgreen); fill(dgreen);}
    RefinedAring.showArrows();
    if(showArrow) P.drawArrows(); // draws all control arrows

  if(recordingPDF) endRecordingPDF();  // end saving a .pdf file with the image of the canv
```

I check that the Spiral and LERP variants work:



Nice! Now, I can put that in a loop.
I'll use a ARROWRING copyInto method to make things simpler and clearer:

```
52   void refineInto(ARROWRING A2)
53     {
54     A2.empty();
55     for(int a=0; a<nA; a++)
56       {
57       A2.addArrow(A[a]);
58       A2.addArrow(AverageOfArrows(A[a],A[n(a)]));
59       }
60     }
61   void copyInto(ARROWRING A2)
62     {
63     A2.empty();
64     for(int a=0; a<nA; a++) A2.addArrow(A[a]);
65     }
66
```

Now, I create a TempAring:

```
15 ARROWRING Aring = new ARROWRING();
16 ARROWRING RefinedAring = new ARROWRING();
17 ARROWRING TempAring = new ARROWRING();
18
```

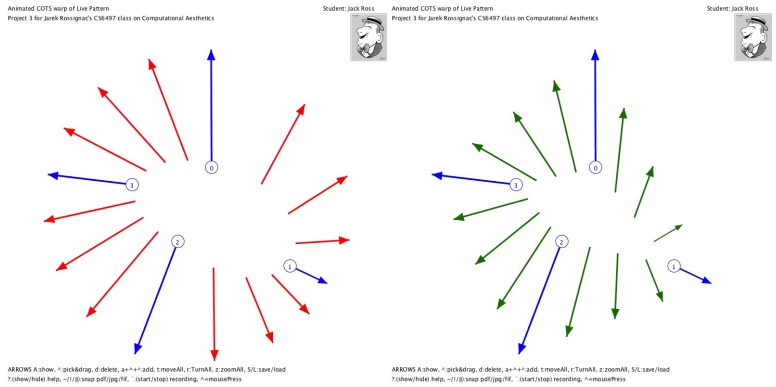
Declare it in setu():

```
31   Aring.declare();
32   RefinedAring.declare();
33   TempAring.declare();
34   } // end of setup
```

And use it in draw() to iterate:

```
44     Aring.copyInto(RefinedAring);
45     for(int i=0; i<2; i++)
46       {
47       RefinedAring.refineInto(TempAring);
48       TempAring.copyInto(RefinedAring);
49       }
50     if(spiralAverage) {stroke(red); fill(red);} else {stroke(dgreen); fill(dgreen);}
51     RefinedAring.showArrows();
52     if(showArrow) P.drawArrows(); // draws all control arrows
```

And test it:



Wow. Incredible. It worked!
Since I feel lucky, I will declare a refinement counter:

```
17 ARROWRING TempAring = new ARROWRING();
18 int refineCounter = 1;
19
```

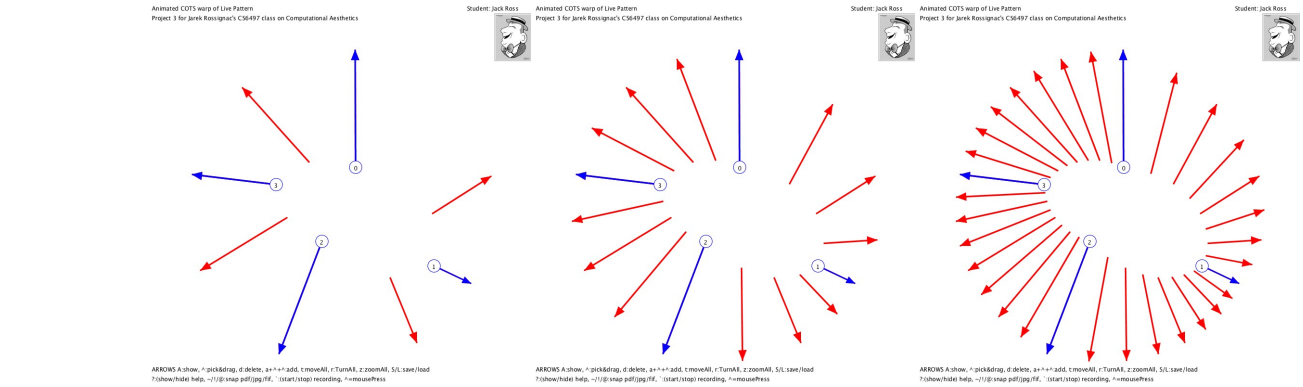
And, in the GUI tab, add keys to control it:

```
17     if(key=='_') ;
18     if(key=='+') refineCounter++;
19
30   if(key=='0') ;
31   if(key=='-') refineCounter=max(1,refineCounter-1);
32   if(key=='=') ;
```

Then, I use it to control the number of refinements

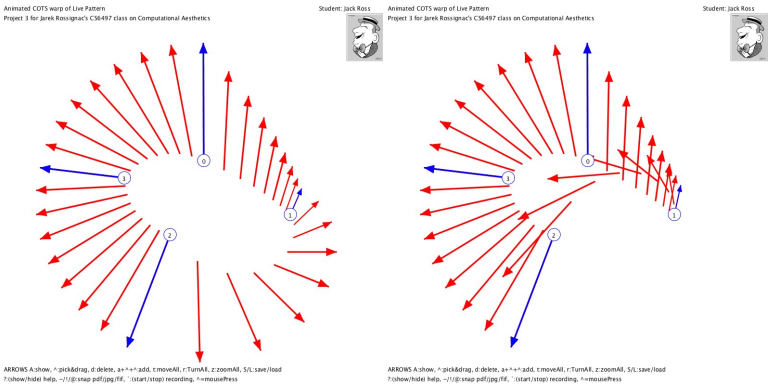
```
45     for(int i=0; i<refineCounter; i++)
46       {
47       RefinedAring.refineInto(TempAring);
48       TempAring.copyInto(RefinedAring);
49       }
```

And try it:



Wow! What a day! I love programming ;-)
Now, let's see whether this works for weird configurations.

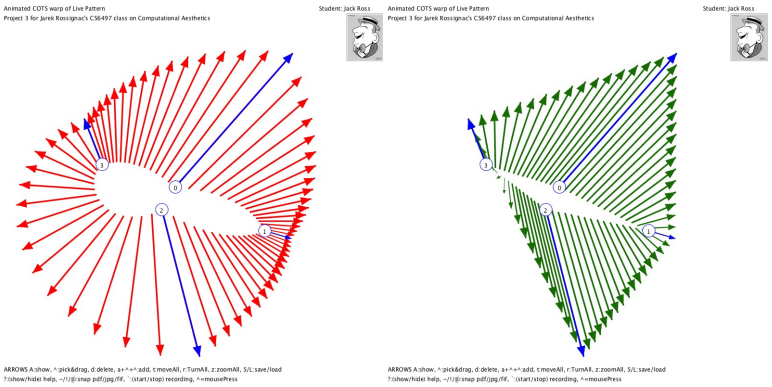
As expected, it was easy to figure a place where changing arrow (1) creates a jump in the average sequence.
I saved the second configuration ('S') in case I come up with ideas on how to fix it.



But I am not yet sure that this needs fixing. Let's discuss (in class) with the artist/customer to figure out what makes sense.

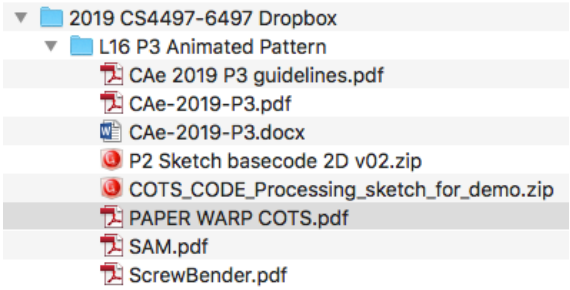
3.8 Quintic B-Spline subdivision of the ARROWRING

You should be able to work this our by yourselves (but I will discuss, in class, the math again).
Make and save a nice configuration for which the previous interpolation has clear velocity discontinuities at the key-arrows.
Show the benefits for both the LERP and the Spiral morphs.

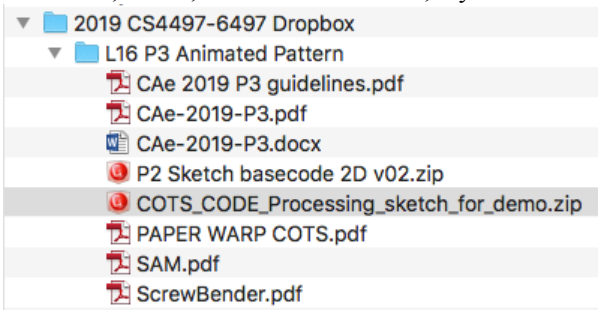


4 Animated Bilinear and COTS warps of your face

Read the COTS paper.



Download, install, and learn how to run, my COTS demo sketch



Play with it and try to understand the code.
Especially, given the four control points (A,B,C,D), how to compute the COTS map, how to use it to display a grid, how to use that grid to display a warped texture of your face.
Use the start & end points of control-arrows (0) and (1) as points (A,B,C,D). Check the paper carefully to get their order right. Show your warped face. Move these two control-arrows by hand to ensure that all is good.
Now, add an animation option (start/stop it by pressing 'a'), which, at each frame, advances these two arrows to the next arrow of the refined or of the subdivided ARROWRING.

5 Animated Bilinear and COTS warps of your pattern

Here, these is no customer or artist to tell you what to do. But I will gladly serve as a consultant.