CS335 Program 3: "JMorph" Phase I
Due Friday, 16 November

1. Introduction

The goal of this project is to provide a user interface and backend driver that supports the specification and rendering of a piecewise (triangular) image morph between two images. Once specified, the morph will be rendered as a sequence of images that can be wrapped as a video (mp4, for example) and used in standard video editing software. Your work will be divided into two phases. The first phase will develop *graphical support* for specifying and previewing all the parameters (geometry, number of frames, start/end images, etc.) of the morph. The second phase will complete the backend of the framework by applying the specified transitions to the underlying target images and by adding additional features.

2. Features

The morph is defined to be a piecewise linear mapping over time from a start image to an end image. The start and end images function as keyframes; this program is to generate the tween frames of the mapping. The user must specify three primary pieces of information to define the morph: position of control points in starting image; position of control points in ending image; number of tween frames to render.

In this phase you must support a 10x10 evenly spaced grid of control points on both the start and end images. The user must be allowed to move the control points by dragging them with the mouse. You must display this process using rubber-banding. You must further devise a cue that indicates *correspondence* (which control point in the other image corresponds to the one I am dragging in this image?). Overall the user must be enabled to position any/all of the 100 control points in each of the two images in order to define the piecewise mapping between the two images. The final position of the control points is the definition of the warp: corresponding triangles in the two images (from a triangulation of the control points) define a geometric warp of those triangles from the start position to the end position.

In terms of the number of tweens to render, the user must be enabled to choose (on some intuitive scale) how to set the number of frames to render. At 30 frames per second, for example, the user could choose how long the transition should last in seconds, which would then determine the number of tween frames to generate.

Once parameters are selected and control points have been positioned, the user must be allowed to *preview* the geometric warp as an animation. The preview should show how the control points will move over time from the start position to the end position.

It is not necessary in this stage to apply the warp to underlying images, or to render actual frames. This phase should concentrate on the user interface and the specification/localization of the control points and warp parameters, and the preview of the defined piecewise warp.

3. Design Issues

Design your project so that parameters can be changed. For example, the 10x10 grid resolution is arbitrary. It will likely change in phase two (to more points, and the ability to select the grid resolution). If you design the control point management so that it is modular, object-oriented, and scalable, this change will be much easier in the second phase.

The visualization of the animated control structure (the “preview” of the geometric warp) is also important and should be implemented so that finer control of the preview is possible in phase 2 (for example, speeding up or slowing down the animation, etc.)

Finally, consider a way to save and restore projects (positions of points, the settings, the images being used, etc.) We will discuss this as a feature in the final phase of the project.

4. What to Turn In

Submit a complete solution to the Canvas portal for Program 3. If you are working in a team, only one submission is required.