Animation Studio
CSC490
By Michael Lee and Mohamed Chehab

Goals and Purpose

The main focus of this project is to create an animation tool that allows users to make some convincing animations. Using gesture input, we hope it can streamline the use of the animator and improve the user interface. Our secondary focus is to improve ways of speeding up the animation process to allow users to create quick, professional animation. Later on, another focus was added to explore with Windows Presentation Foundation (WPF).

Originally, the main goal was to create an application that allows users to create fluid animations. We also wanted to allow users to:

- Import images
- Translate objects along a path
- Oscillation between images to form animation
- Rotation of an object with mouse gestures
- Change faces to do facial expressions with mouse gestures.

Short-comings

During production, we encountered a few things that prevented us from achieving everything that we wanted:

- Gestures that controls speed was considered too impractical for users. It's difficult to control the mouse for long periods of dragging to match the speed you want exactly, and the frequent errors during mouse input would give undesirable results.
- Editing exact key-frames. This is linked with using the storyboard class.
- Animating gifs, problem also linked with WPF
- Automatically rendering animation that logically makes sense in the spatial-temporal sense. We just didn't have enough time to implement this function

Accomplishments

Despite our short-comings, we still managed to finish many of the milestones we set out initially:

- Users can import images as animation objects
- Translation of objects along motion paths
- Allows frame by frame animation
- Rotating an object
- Time control using multiple paths
- Oscillation and repetition for the above functions
- Explored different usage of gestures as an user input

Future Changes and Direction

In the months of working on this project, there are many things that we would have done differently. First, we wouldn't have implemented in C#'s WPF. WPF is a very awesome environment for small programs that requires animation. Later on, we started to

experience difficulties with the language as it started to behave in an unexpected pattern as we added many key-frames for path animations which is the reason we did not implement path animations with key-frames. Also, working off of another layer of design makes it harder to debug our code. The API is also poorly written, where many times the purpose of the class is not well stated and the sample code was not in the standard background code we're used to seeing, but rather in an XML format. WPF is also slow when there are too many environments when it can easily be optimized if it was in another language, making it slightly inefficient.

Another thing we would have liked to do is to plan out our program better. Without much foresight or modularity, we ended up changing many lines of code just to include new functions multiple times. This proved to be costly in time and effort. We would have also liked to have more direction in our planning. We spent a lot of time experimenting around with different environments and languages, and many of the languages are not compatible with each other, making that work mostly wasted with respect to the final project.

Other difficulties we witnessed included difficulty associating gestures to animations techniques and functions that we have. Despite hoping that gestures would be able to add in the timing and speed of an object moving around screen, the frequency of errors during input makes this a very hard to implement gesture. Since timing is very important, to let it go error ridden would be very unprofessional, hence the function was removed. We also found it hard to find gestures to simplify frame by frame animation process, but ultimately, we didn't find a solution. There are many more gesture ideas we would have loved to explore but didn't have the time to.

Implementation

The application was programmed in C#, using the WPF Storyboard. The storyboard keeps track of a timeline, and allows events to be applied to it. When the event timer reaches a flag, the storyboard will apply the appropriate changes. We applied this onto animation objects. Animation object is a class we created that keeps track of the image information, different flags and events, and animation data. Associating changes to an image with a timeline is the basis of all animation

WPF's Storyboard also handles matrix animation. We applied the rotation and translation matrix animation via the storyboard onto the images to facilitate rotation and translate function. It was harder to work out because implementing on one timeline implies that one edit will stop the other one from taking effect, creating odd results. Therefore we implemented two timelines, having them edit the same image at the same time. This is also true for frames. Separate timeline has been created to say which image to change the current 'placeholder' image to. The values are then passed to the placeholder so it changes its source to the new image.

During the creation of the translation path, we implemented a simple Laplacian smoothing function so we don't have as many jagged lines that would normally show up if we took all of the inputs directly from the mouse. After the line has been smoothed out, points would be taken from the line and input into the timeline as coordinates.
Several gestures were implemented in the software to handle various functions with objects such as selecting, moving, adding, deleting them. There are also gestures for playing and stopping the storyboard. We used a GestureRecognizer to look at a stroke

and then find the best match for a gesture. For example, if the gesture is recognized as up-right, then the function to add a frame to the selected object is started. Currently in our code there is also a timer attached to the gesture stroke. This was initially used for checking the timing to compare with length of a stroke and creating a duration from this information, but it proved to be inaccurate to work with for users so all durations are edited using input in the left and right panes. Unfortunately we could not find better implementation of using gestures to get useful information about the rotations and translations in the time we had.

How to use the program

Adding a background:

1. Press Background button
2. Select image and press Open

Adding an Image:

1. Press Add Object[1]
2. Select image and press Open

[1] if you're going to make this image into a frame by frame animation, you need to change the value in the duration textbox under "Frames" section to define how long this frame lasts.

Adding frames to an Image:

1. Change the duration value to change how long the frame lasts (in seconds)
2. Press Add Frame
3. Select the next frame and press Open

Focus on an Image Object:

1. Left-click on the image

Moving an image:

1. Tap on an object you wish to move
2. Move the image to desired location
3. Tap again to release the object

Note: Images can only be moved when the storyboard is stopped.

Record New Path

1. Select desired image that you wish to change
2. Change the desired duration that the image will play this path. This symbolizes how long it takes for the image to go from the beginning to the end of the path.
3. Right-click and drag the mouse to make the new path
4. Let go when you're done

To overwrite a path:

1. Select desired path to overwrite
2. Record a new path

To select a desired path:

1. Scroll up/down on the mouse wheel. The colour under the rotation section will change accordingly

Animate Rotation

1. Select image
2. Select how much should it rotate by
3. Select how long it takes to reach the angle

Repeat/Auto-reverse Frames, Translation, and Rotation:

1. Select image
2. Select appropriate radio boxes

Note about behaviour:

Selecting repeat makes the animation cycle. Just by itself, it'll go through the same path, frames, or rotation from start to finish. Auto-reverse by itself will make the image go back the same alterations after it reaches finish to return to original state. Combining the two will make the image oscillate.

Play/Stop/Pause the Animation:

1. Press the respective button to play, stop or pause the animation

Hide/Reveal Path Lines:

1. Press 'Hide paths' button

Hide/Show highlights:

1. Press 'Hide/Show highlight' button

Gestures:

- Down-Left (a L shape): Adds an image object
- Up-Right (an upside-down L): Adds a frame to the object
- Scratch: deletes an image
- Right chevron (>): Play storyboard
- Left chevron (<): Stop storyboard