# CREATING BASIC ANIMATIONS IN MATLAB

Katrina Palmer
Appalachian State University
346 Walker Hall, Boone, NC 28608
palmerk@appstate.edu

This paper describes how students with little matrix knowledge and no programming or MATLAB experience created short quicktime videos in 5 days (meeting 1.5 hours a day). The intent for creating this short module was to include it in a new computational math course for Applachian State University's new computational math degree [1], but this course has been delayed so this module was tested on summer venture students [4]. Summer Ventures students are some of the top achieving high school math and science students in North Carolina. This module starts with linear transformations, then introduces MATLAB and *for* loops to create short animations, then ends with a sample video.

**Introduction to Matices and Linear Transformations**

Since basic addition, subtraction, and multiplication of matrices is now taught in high school [3], we started the module with, "What is a linear transformation?" In order for a transformation to be linear, two criteria must hold: 1) adding two vectors, then transforming them produces the same result as first transforming two vectors, then adding 2) multiplying a scalar and a vector, then transforming that new vector produces the same result as first transforming the vector, then multiplying by the scalar. For example, consider point A (-2, -1) and point B (3,-4) in Figure 1. We'll use these two points to see if the first property holds for reflection across the *x*-axis. Figure 1 visualizes the algebra cooresponding to the first criteria.

Adding first, then reflecting we get: $(-2,-1)+(3,-4)=(1,-5)\xrightarrow{reflect}(1,5)$

Reflecting, then adding we get: $\begin{cases}(-2,-1)\xrightarrow{reflect}(-2,1)\\(3,-4)\xrightarrow{reflect}(3,4)\\(-2,1)+(3,4)=(1,5)\end{cases}$

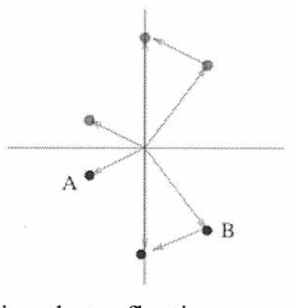

Figure 1: Visual representation that reflection across the x-axis satifies the first criteria of a linear transformation.

To demonstrate the second criteria, consider the point (-2,-1) and the scalar *k*. Multiplying by *k*, then reflecting we get: $k(-2,-1)=(-2k,-k)\xrightarrow{reflect}(-2k,k)$

Reflecting, then multiplying we get: $\begin{cases}(-2,-1)\xrightarrow{reflect}(-2,1)\\ k(-2,1)=(-2k,k)\end{cases}$

Since both criteria hold, it would suggest that reflection across the *x*-axis is a linear transformation. Note that this is not a proof because it is not generalized, but that is not the goal of this module.

If these two citeria hold for a transformation, then the transformation is linear and can be computed by matrix multiplication. The matrix that you multiply by comes from knowing where (1,0) and (0,1) (the standard basis elements for $\Re^2$) get transformed. For reflection across the *x*-axis again, $(1,0)\xrightarrow{reflect}(1,0)$ and $(0,1)\xrightarrow{reflect}(0,-1)$ so the matrix is $\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$. The first column is the vector that (1,0) transforms to and the second column the the vector that (0,1) transforms to. In other words, if you multiply a vector by $\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$, the resulting vector is the reflection across the x-axis. For example, (3,4) reflected across the x-axis is (3,-4) and $\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}\begin{pmatrix}3\\ 4\end{pmatrix}=\begin{pmatrix}3\\ -4\end{pmatrix}$.

After learning what a linear transformation is, students investigated which of rotation about the origin, dilation, reflection (across the *x*-axis, *y*-axis, and $y=x$), and translation are linear transformations, and derived the matrices that represent the linear transformations. For example, rotation around the origin is a linear transformation, and the matrix that rotates a vector $\theta$ degrees about the origin is $\begin{pmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{pmatrix}$. Of the transfomations listed above, all are linear transformation except translations. To see this, consider the example that translates up 2 and right 3. Checking the first criteria:

Adding first, then translating we get: $(1,0)+(0,1)=(1,1)\xrightarrow{translate}(4,3)$

Translating first, then adding we get: $\begin{cases}(1,0)\xrightarrow{translate}(4,2)\\ (0,1)\xrightarrow{translate}(7,5)\\ (4,2)+(7,5)=(11,7)\end{cases}$

Since $(4,3)\neq(11,7)$, criteria (1) does not hold, and thus this translation is not a linear transformation, and therefore translation can't be modeled by matrix multiplication in $\Re^2$. Our goal is to use matrix multiplication for animation, so we need to fix this. The remedy is to extend into 3-space using homogeous coordinates, that is, a point $(x,y)$ is rewritten as $(x,y,1)$. Since each point is now written as a 3x1 vector, the transformation matrix is modified to be a 3x3 matrix instead of a 2x2 matrix by the following extention:

$$\begin{pmatrix} \text{2 by 2} \\ \text{transformation matrix} \end{pmatrix} \text{ becomes } \begin{pmatrix} \begin{pmatrix} \text{2 by 2} \\ \text{transformation matrix} \end{pmatrix} & \begin{matrix} 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 & \qquad 0 \end{matrix} & 1 \end{pmatrix}$$

For example, the transformation matrix to rotate around the origin $\theta$ degrees becomes

$\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$. This allows for translations to be represented by matrix multiplication; that is, to translate a point right $h$ and up $k$, you multiply by $\begin{pmatrix} 1 & 0 & h \\ 0 & 1 & k \\ 0 & 0 & 1 \end{pmatrix}$.

Since rotations around a point (other than the origin) are not linear transformations (check this!), we can combine translations with rotation around the origin. For example, to rotate $\theta$ degrees around the point (2,3), first translate to the origin, then rotate, then translate back by multiplying on the left by the following matrices:

$$\begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -2 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{pmatrix}$$

So using composition, we can combine several different transformations to create animations.

**Introduction to Matlab and *for* Loops**

The students were given a brief introduction to inputing matrices and plotting in MATLAB, then used exerpts from [2] to practice. To gain more experience with plotting and transforming, students also worked problems similar to the ones below. These problems also provide practice with the `hold on` command which allows for multiple plots on the same figure.

*Problem* Write code to: Plot the vertices (0,0), (0,3), (1,4), (4,3), and (4,0), then use matrices to plot (on the same figure) the reflection across the y-axis, then using matrix multiplication rotate the points 30 degrees about the origin and plot them (on the same figure).

*Problem* Write code to: Plot the vertices (0,0), (0,3), (1,4), (4,3), and (4,0), then plot the vertices after shrinking them by a factor of .95 and then rotate them 45 degrees counter clockwise about the origin.

*Problem* Write code to: Plot the circle with radius (3,4). Then translate the circle right 5 and up 3, and plot the new circle on the same figure.

The students were introduced to *for* loops using basic examples and problems such as:

*Problem* Predict the output of the following code, then test it.

```
Total = 0;
for i = 1:10
           Total = Total + i;
end
```

*Problem* Describe the output of the following code, then test it.

```
M=[0 2 2 0 0; 0 0 1 1 0;1 1 1 1 1];
theta = pi/4;
RotM = [cos(theta) -sin(theta);
        sin(theta)  cos(theta)];
for i = 1:8
     M = RotM * M;
     plot(M(1,:),M(2,:),'b*')
end
```

In order to make a quicktime video from their codes, they need two commands: `getframe` and `movie2avi`, both of which are available in the image processing toolbox. `getframe` returns a movie frame, and is a snapshot of the current axes or figure. The command movie2avi converts the series of saved frames into a movie, and saves it as an avi (Audio Video Interleave) file which is a format that quicktime can read. Note that if you are using octave, you could still see the animation in the figures. The advantage to making it into a quicktime video is that they can keep the videos without needing MATLAB. Several of the students even emailed their videos to their parents.

**Sample Project**

Here is the MATLAB code created by one of the students. After running the script below, type

```
>> movie2avi(Mov,'MyMovie.avi')
```

at the command line to convert the frames into a video. The first and last frame of the resulting video named MyMovie is shown in Figure 2. The video can be played at http://www.mathsci.appstate.edu/~kmp/research/SV1Mov.avi.

```
M = [1 2 2 1 1; 1 1 2 2 1; 1 1 1 1 1];
X = [1 2 2 1 1; 1 1 2 2 1; 1 1 1 1 1];
rho = pi/8;
figure, plot(M(1,:), M(2,:),'b-.', M(1,:), M(2,:),'*m'),
axis([-10,10,-10,10]); hold on
```

```
plot(X(1,:), X(2,:),'--g',X(1,:), X(2,:),'xr')
TransOrigin=[1 0 1; 0 1 1; 0 0 1];
TransBack=[1 0 -1; 0 1 -1; 0 0 1];
RotMatrix=[cos(rho) -sin(rho) 0 ;sin(rho) cos(rho) 0; 0 0 1];
DilMatrix=[1.01 0 0;0 1.01 0; 0 0 1];
CompositionMatrix=TransBack*DilMatrix*RotMatrix*TransOrigin;
for i = 1:64
    M = CompositionMatrix*M;
    pause(.08)
    plot(M(1,:),M(2,:),'-.b',M(1,:),M(2,:),'*m'),
    X = CompositionMatrix*X;
    Mov(i) = getframe;
    pause(.08)
    plot(X(1,:),X(2,:),'--g',X(1,:),X(2,:),'xr'), hold on
end
```

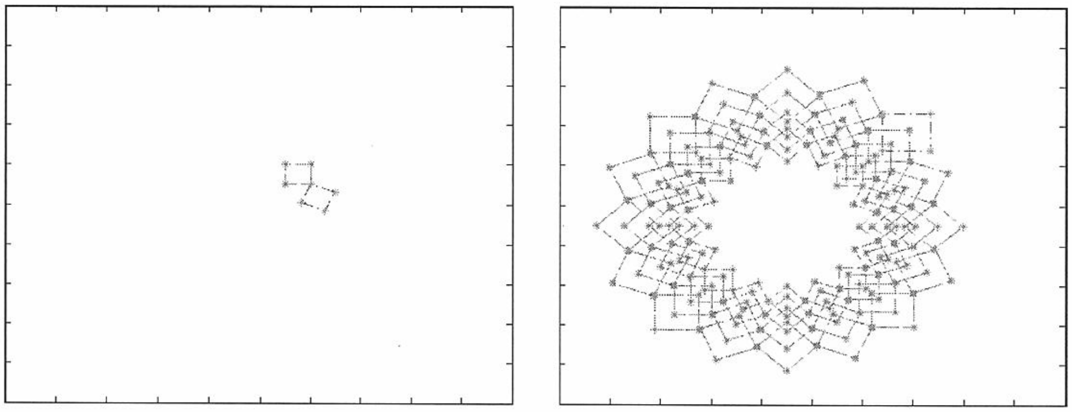
Figure 2: First and last frame from sample student video

## Conclusions

In summary, students learned a great deal about matrices, *for* loops, and MATLAB – all of which they hadn't seen before. Even though this was a very short period of time (and the summer venture students couldn't have homework), the student response was far better than I had imagined it would be. Several students wanted to stay later or have access to the lab to be able to work on their animations. When our computational math class is finally offered, I plan on incorporating this module to teach them loops and MATLAB as they will have already had linear algebra.

References:

[1] Department of Mathematical Sciences, Appalachian State University. May 14, 2008 <http://www.mathsci.appstate.edu/Student>

[2] Griffiths, David. *An Introduction to Matlab,* Version 2.3, University of Dundee, 2005.

[3] National Council for Teachers of Mathematics. May 14, 2008 <http://www.nctm.org>

[4] Summer Ventures. May 14, 2008 <http://www.summerventures.org>