# Using Works of Visual Art to Teach Matrix Transformations

James Luke Akridge*, Rachel Bowman*, Peter Hamburger, Bruce Kessler
Department of Mathematics
Western Kentucky University
1906 College Heights Blvd.
Bowling Green, KY 42101
E-mail: james.akridge620@wku.edu, rachel.bowman708@wku.edu,
peter.hamburger@wku.edu, bruce.kessler@wku.edu
**Undergraduate students.*

## Abstract

The authors present a modern technique for teaching matrix transformations on $\mathbb{R}^2$ that incorporates works of visual art and computer programming. Two of the authors were undergraduate students in Dr. Hamburger's linear algebra class, where this technique was implemented as a special project for the students. The two students generated the images seen in this paper, and the movies that can be found on the accompanying webpage **www.wku.edu/~bruce.kessler/**.

## 1 Introduction

As we struggle to increase the number of students succeeding in the science, technology, engineering, and mathematics disciplines, it is the opinion of the authors that

- we must find ways to motivate a larger audience as to the importance and relevance of these disciplines, and
- once we have convinced students to invest in a discipline, we must then be innovative in the ways that we engage the students in learning, so that they stay in the discipline.

The second of these ideas was put into practice in Dr. Hamburger's Fall 2008 linear algebra class, in such a way that both the scientifically-minded and artistically-minded students were actively engaged in the course, and eventually learned a great deal about the topic.

Matrix multiplication has a number of applications that can attract the interest of students, but as we move to a more abstract understanding of matrices as generators of transformations, we sometimes lose the interest of the more application-driven students. In general, multiplying an $m \times n$-matrix (that is, a matrix with $m$ rows and $n$ columns) on the right by an $n$-vector (column), generates an $m$-vector, so it can be interpreted as a transformation from $\mathbb{R}^n$ to $\mathbb{R}^m$. We generally illustrate this type of transformation in $\mathbb{R}^2$ with rather dull diagrams, such as the ones shown in Figure 1. If the matrix causes the same change with every vector in $\mathbb{R}^2$, then the effect of the matrix multiplication would be shown on one vector, as on the left in Figure 1, where $R(x,y)$ is a counterclockwise rotation about the origin of 60°. If the matrix has varying effects depending on the choice of the vector, then perhaps two vectors are shown under the transformation, as on the right in Figure 1, where $T(x,y)$ stretches vectors horizontally by a factor of 2 and vertically by a factor of 1.5. (See sample college-level linear algebra texts [1, 2, 3, 4], where [3] probably comes closest to the work illustrated in this paper with its visual interpretation of matrix transformations.)

This method of showing the effects of matrix multiplication is antiquated and goes back to a time when computers were not available. A better understanding of the effect of the matrix multiplication would be

*The authors would like to thank the four anonymous reviewers for their valuable suggestions which improved the manuscript.

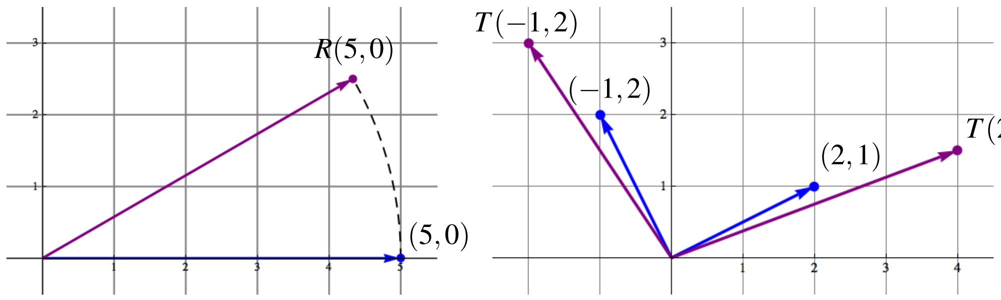

**Figure 1**: Typical diagrams illustrating the effects of matrix multiplication.

gained by applying the matrix to every point in $\mathbb{R}^2$, an impossibility. However, we can look at a large number of regularly-spaced vectors in the plane, but drawing all of the vectors would cause an undecipherable mess. Instead, we start with a digital image that we recognize, so that we can tell the effects of the matrix transformation on each pixel of the image. This could be a digital portrait of anyone, but given that some people might object to having their portrait warped, we decide instead to use famous works of visual art. This gives the observer an original image to which they can compare, and has the added bonus of capturing the interest of students that have an affinity to the visual arts.

The following section provides a brief introduction to matrix transformations in general, and to the specific code that was used to generate the examples in this paper.

## 2 Definitions and Code Used

**Terminology.** A ***transformation on*** $\mathbb{R}^2$ is any mapping $T$ that takes a point in $\mathbb{R}^2$ to a point in $\mathbb{R}^2$. Any transformation on $\mathbb{R}^2$ that can be expressed in the form

$$T\begin{pmatrix}x\\y\end{pmatrix} = \begin{bmatrix}a(x,y) & b(x,y)\\ c(x,y) & d(x,y)\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}, \tag{1}$$

where $a$, $b$, $c$, and $d$ are real-valued functions dependent upon $x$ and $y$, is called a ***matrix*** transformation on $\mathbb{R}^2$. Note that we use the short-hand notation $T\begin{pmatrix}x\\y\end{pmatrix}$ in place of the bulky, but more accurate, notation $T\left(\begin{bmatrix}x\\y\end{bmatrix}\right)$. For example, the transformation $T$ defined by

$$T\begin{pmatrix}x\\y\end{pmatrix} = \begin{bmatrix}x^2+y^2\\ 2xy\end{bmatrix} \text{ can be considered a matrix transformation, since } T\begin{pmatrix}x\\y\end{pmatrix} = \begin{bmatrix}x & y\\ y & x\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}.$$

Not all transformations are matrix transformations. As an example, the transformation $T$ defined by

$$T\begin{pmatrix}x\\y\end{pmatrix} = \begin{bmatrix}a & b\\ c & d\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix} + \begin{bmatrix}e\\f\end{bmatrix}, \quad a,b,c,d,e,f \in \mathbb{R}, \tag{2}$$

called an ***affine*** transformation, is defined for all $(x,y) \in \mathbb{R}^2$ and always maps to a point in $\mathbb{R}^2$, but it does not fit the form of (1) unless $\begin{bmatrix}e\\f\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$. In that case, the transformation is said to be a ***linear*** transformation.

**Programming the Transformations.** The code used for this project was written by the author Kessler for the algebraic manipulation software package *Mathematica*$^{\text{TM}}$, due to the ease in generating graphics and movies, and due to our students' familiarity with the software from their prerequisite coursework. The code is presented here in the syntax of the *Mathematica*™ programming language, which is very readable since the commands are usually full English words. The code generates an animation of the transformations applied recursively, which we obviously can not show here. In our illustrations, we show selected frames from the animations. The full animations are available for download at **www.wku.edu/~bruce.kessler**.

We opted to center the digital images on our coordinate axes, as shown in Figure 2. This allows us the best "viewing window" with which to watch the effects of the matrix transformation on the image when using our choices of matrices. The units on the axes are pixel-widths. The size of the images were not standardized, as our code adjusts for the size of the image.

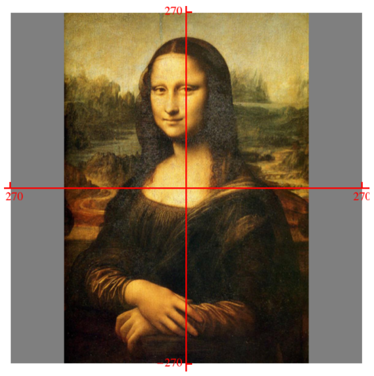

**Figure 2**: Placement of the coordinate axes with respect to the digital images.

While we want to give the appearance that we are moving the pixels to different locations in the plane, we are actually considering each pixel as a fixed location. Thus, instead of changing the location of a pixel $P$ by repeated multiplying by a matrix $A$, we apply the inverse matrix $A^{-1}$ to each pixel $P$, and note the color of its resulting pixel $P'$. We then recolor each pixel $P$ to that color, giving the illusion of applying the transformation to each pixel. The *Mathematica*™ code is given below, with the matrix set to $\begin{bmatrix} \frac{79}{80} & 0 \\ 0 & \frac{39}{40} \end{bmatrix}$.

```
(* The following command inputs the image file.  You supply the
 file name. *)
temp = Import["31PersistenceOfMemory.jpg"];
colors = Reverse[temp[[1, 1]] / 255.];
(* Scales the RGB values for Mathematica. *)
{rows, cols, check} = Dimensions[colors];  (*Sets the dimensions. *)
countermax = Max[rows, cols];
rmax = rows / 2.; cmax = cols / 2.; max = Max[rmax, cmax];
centers = Table[{-max + j - .5, -max + i - .5}, {i, countermax}, {j, countermax}];
(* The following command inputs the matrix that will be used to
 generate the graphics. *)
matrix[x_, y_] := {{79/80, 0}, {0, 39/40}} // N;
imat[x_, y_] = Inverse[matrix[x, y]];
f[{x_, y_}] := imat[x, y].{x, y};
```

```
(* The following two commands put the original picture into the
 list of frames, as the initial frame. *)
newcolors =
  Table[If[1 ≤ Floor[centers[[i, j, 1]] + cmax + .50000001] ≤ cols &&
     1 ≤ Floor[centers[[i, j, 2]] + rmax + .50000001] ≤ rows,
    colors[[Floor[centers[[i, j, 2]] + rmax + .50000001],
     Floor[centers[[i, j, 1]] + cmax + .50000001]]], {.5, .5, .5}],
   {i, countermax}, {j, countermax}];
frames = {Graphics[Raster[Reverse[newcolors]]]};
Print["0"];
(* The following do-loop generates the frames for the animation.  You
  specify how many frames past the original that are generated
  by changing the value after "n," in the last line. *)
Do[(
  centers = Map[f, centers, {2}];
  newcolors =
   Table[If[1 ≤ Floor[centers[[i, j, 1]] + cmax + .50000001] ≤ cols &&
      1 ≤ Floor[centers[[i, j, 2]] + rmax + .50000001] ≤ rows,
     colors[[Floor[centers[[i, j, 2]] + rmax + .50000001],
      Floor[centers[[i, j, 1]] + cmax + .50000001]]], {.5, .5, .5}],
    {i, countermax}, {j, countermax}];
  AppendTo[frames, Graphics[Raster[Reverse[newcolors]]]];
  Print[n];
 ), {n, 100}]
Export["WM_stochastic.avi", frames]
```

## 3 Examples

The students experimented with a number of different types of $2 \times 2$ matrices. The following sections and figures show some of the more interesting results from this experimentation. The still figures show the effects of the various matrix transformations much more effectively than the vector diagrams shown in Figure 1, and the animations are even more illustrative.

**Dilations.** Matrix transformations with matrices of the forms

$$\begin{bmatrix} k & 0 \\ 0 & 1 \end{bmatrix} \qquad \text{and} \qquad \begin{bmatrix} 1 & 0 \\ 0 & k \end{bmatrix}$$

are called ***horizontal*** and ***vertical dilations***, respectively. If $0 < k < 1$, the dilation is a ***contraction***. If $k > 1$, then the dilation is called an ***expansion***. The effects can be combined into one dilation matrix, since

$$\begin{bmatrix} k_1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & k_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & k_2 \end{bmatrix} \begin{bmatrix} k_1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix}.$$

The images shown in Figure 3 were generated by applying the dilation matrix $\begin{bmatrix} \frac{79}{80} & 0 \\ 0 & \frac{39}{40} \end{bmatrix}$ to a 580 pixel wide by 418 pixel tall digital image of Salvador Dali's painting "Persistence of Memory". This causes a horizontal contraction, and a slightly faster vertical contraction.

**Shears.** Matrix transformations with matrices of the forms

$$\begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix} \qquad \text{and} \qquad \begin{bmatrix} 1 & 0 \\ k & 1 \end{bmatrix}$$

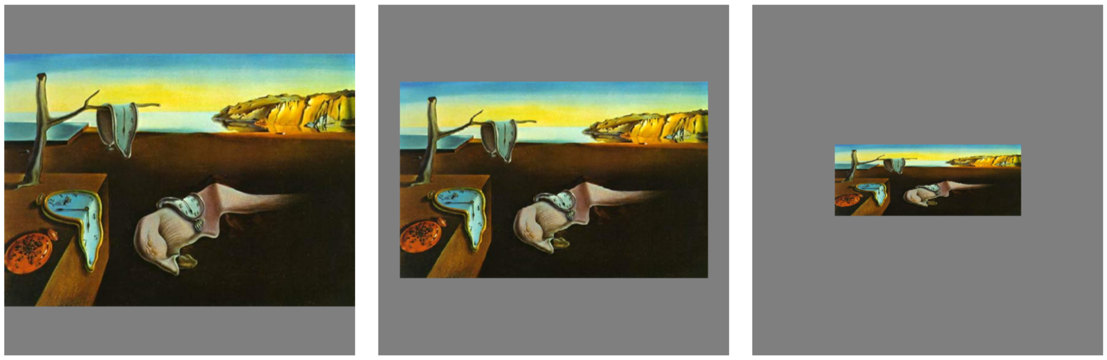
**Figure 3**: The original picture at left. Ten applications of the contraction matrix at center. Fifty applications of the matrix at right.

are called ***horizontal*** and ***vertical shears***, respectively. The sign of the value $k$ determines the direction of the shear.

The images shown in Figure 4 were generated by applying the shear matrix $\begin{bmatrix} 1 & 0.01 \\ 0 & 1 \end{bmatrix}$ to a 563 pixel wide by 705 pixel tall digital image of one of Van Gogh's self-portraits. This causes a horizontal shear, with the top moving to the right and the bottom moving to the left. Figure 5 shows the effects of a vertical shear with the matrix $\begin{bmatrix} 1 & 0 \\ -0.02 & 1 \end{bmatrix}$ applied to the same image.

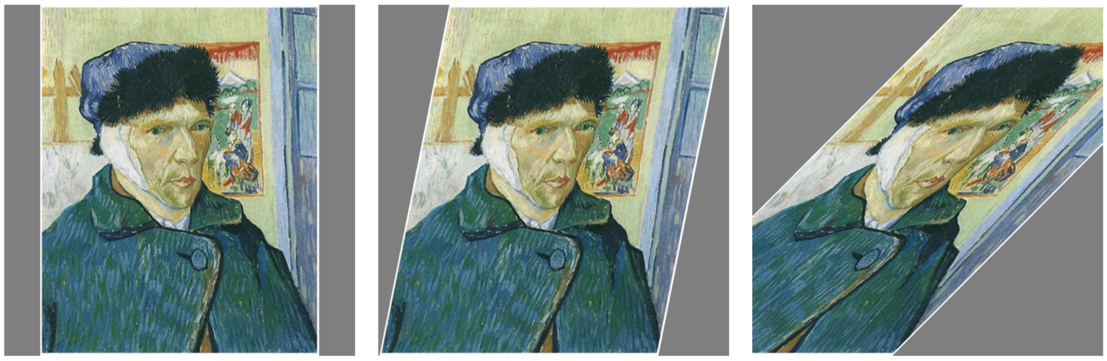
**Figure 4**: The original picture at left. Twenty applications of the horizontal shear matrix at center. One hundred applications of the matrix at right.

**Rotations.** Matrix transformations with a matrix of the form

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

are called ***rotations***. Multiplication by this matrix causes a counterclockwise rotation by $\theta$ degrees about the origin.

The images shown in Figure 6 were generated by repeatedly applying the rotation matrix $\begin{bmatrix} \cos(1^\circ) & -\sin(1^\circ) \\ \sin(1^\circ) & \cos(1^\circ) \end{bmatrix}$ to a 400 pixel wide by 572 pixel tall digital image of Leonardo daVinci's painting "Mona Lisa".

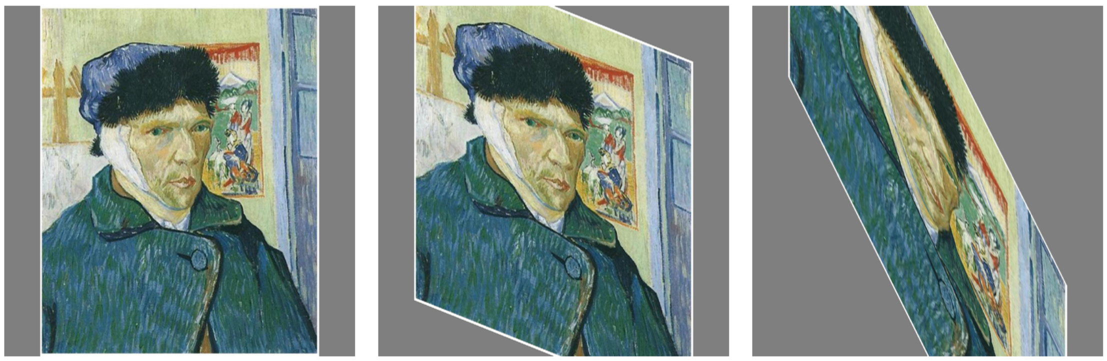

**Figure 5**: The original picture at left. Twenty applications of the vertical shear matrix at center. One hundred applications of the matrix at right.

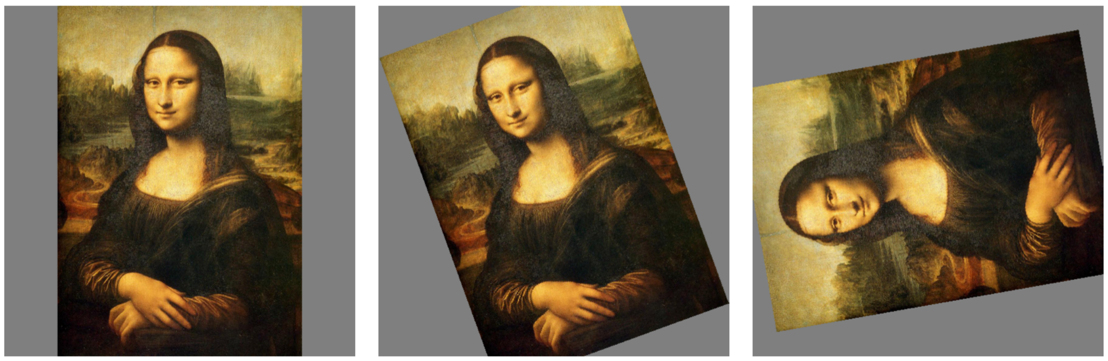

**Figure 6**: The original picture at left. Twenty applications of the rotation matrix at center. One hundred applications of the matrix at right.

**Distance-dependent rotations.** Matrix transformations with a matrix of the form

$$\begin{bmatrix} \cos(f(d)) & -\sin(f(d)) \\ \sin(f(d)) & \cos(f(d)) \end{bmatrix},$$

where $f$ is a function defined on non-negative real numbers and $d$ is the Euclidean distance from the origin $\sqrt{x^2+y^2}$ for the pixel with coordinates $(x,y)$, are called ***distance-dependent rotations***. The images shown in Figure 7 were generated by repeatedly applying the distance-dependent rotation matrix

$$\begin{bmatrix} \cos\left(2\sqrt{\frac{x^2+y^2}{rmax^2+cmax^2}}\,1^\circ\right) & -\sin\left(2\sqrt{\frac{x^2+y^2}{rmax^2+cmax^2}}\,1^\circ\right) \\ \sin\left(2\sqrt{\frac{x^2+y^2}{rmax^2+cmax^2}}\,1^\circ\right) & \cos\left(2\sqrt{\frac{x^2+y^2}{rmax^2+cmax^2}}\,1^\circ\right) \end{bmatrix},$$

where *rmax* and *cmax* are as defined in the *Mathematica*™code, to a 400 pixel wide by 572 pixel tall digital image of Leonardo daVinci's painting "Mona Lisa".

**Stochastics.** Matrices of the form

$$\begin{bmatrix} a & 1-b \\ 1-a & b \end{bmatrix},$$

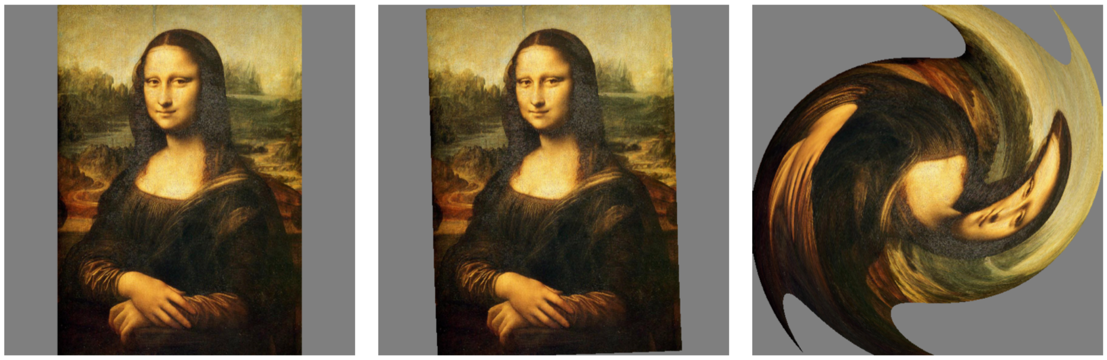

**Figure 7**: The original picture at left. One application of the distance-dependent rotation matrix at center. One hundred eighty applications of the matrix at right.

where $0 \leq a \leq 1$ and $0 \leq b \leq 1$, are called ***stochastic*** matrices. Stochastic matrices are used to calculate probabilities, and are usually thought of in the context of Markov chains or random walks. While not generally thought of in the context of matrix transformations, we can certainly define matrix transformations using these matrices.

The images shown in Figure 8 were generated by applying the stochastic matrix $\begin{bmatrix} \frac{29}{30} & \frac{1}{60} \\ \frac{1}{30} & \frac{59}{60} \end{bmatrix}$ to a 750 pixel wide by 652 pixel tall digital image of James Whistler's painting best known as "Whistler's Mother".

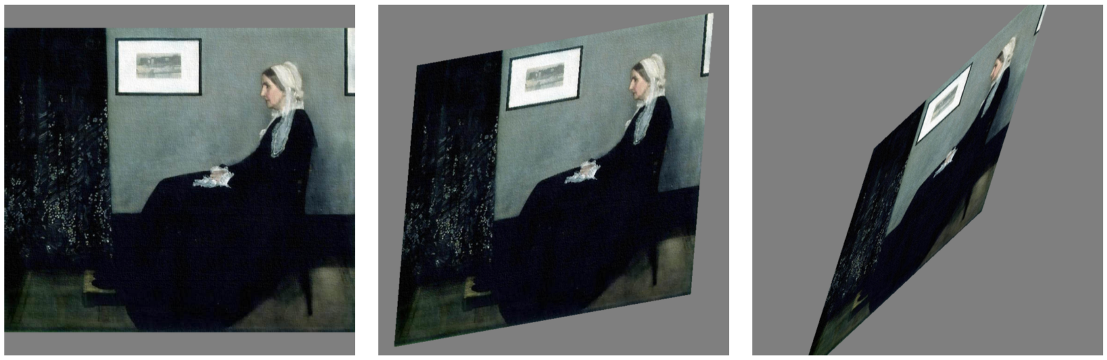

**Figure 8**: The original picture at left. Five applications of the stochastic matrix at center. Twenty-five applications of the matrix at right.

**Other Matrices.** The undergraduate students went through their own type of transformation – from being mathematics students using matrices on art, to "matrix artists" using mathematics to generate their pieces. The students involved in this project were very creative in developing matrices, other than the types described above, that generated beautiful animations, using time and distance dependencies in their matrices. In order to increase computation efficiency, they replaced the pictures with grids of points in the plane, colored according to their initial quadrant as shown at the top left in Figure 9, thereby allowing the students to run experiments more quickly. Figure 9 shows selected frames from animations created by the students, using the transformations

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} 1 - \frac{1}{xy} & -\frac{1}{xy} \\ \frac{1}{xy} & 1 - \frac{1}{xy} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad \text{and} \quad T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} \cos\left(\frac{n\sqrt{x^2+y^2}}{200} 1^\circ\right) & -\sin\left(\frac{n\sqrt{x^2+y^2}}{200} 1^\circ\right) \\ \sin\left(\frac{n\sqrt{x^2+y^2}}{200} 1^\circ\right) & \cos\left(\frac{n\sqrt{x^2+y^2}}{200} 1^\circ\right) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

at top and at bottom, respectively, where $n$ is the iteration number. Again, the still frames do not do the animations justice, and we encourage readers to view the animations on the previously mentioned website.

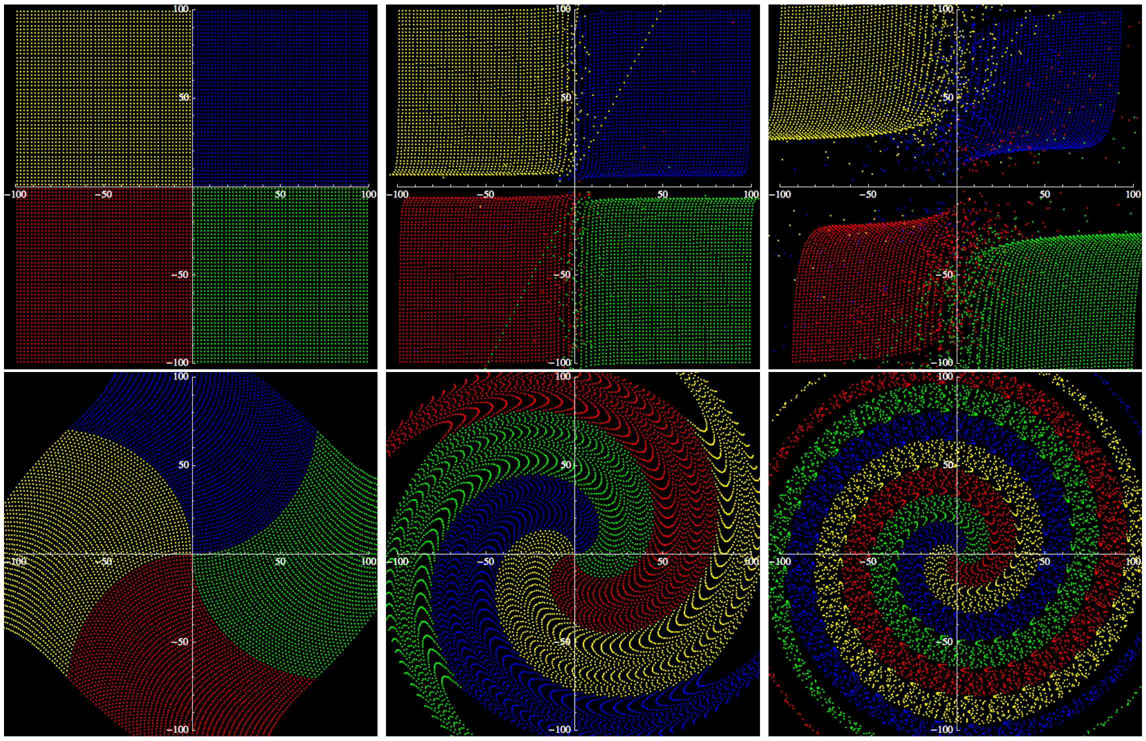

**Figure 9**: The original picture at top left. Progressive frames of one animation, top center and top right. Progressive frames of another animation, bottom row.

## 4 Conclusion

It has often been said that “A picture is worth a thousand words.” In our cases illustrated here and on the fore-mentioned webpage, this idea could be extended to say “An animation is worth a thousand diagrams.” While the idea of using matrix transformations to create art is not new, it was definitely a new approach to learning linear algebra for our students. There can be little doubt that the students who worked on this project now have a better understanding of the effects of these matrices on points in the plane than they would have ever had just looking at vector diagrams. In the course of analyzing the results, they have also touched upon other seemingly unrelated topics, like vector norms, eigenvalues, and matrix limits. In the course of doing so, they have broadened their recognition of art, and have learned how to create their own “masterpieces”.

## References

[1] H. Anton, “Elementary Linear Algebra”, 8th edition, John Wiley & Sons, 2000.

[2] R. Larson, B. Edwards, and D. Falvo, “Elementary Linear Algebra,” 5th edition, Houghton Mifflin Company, 2004.

[3] G. Strang, “Introduction to Linear Algebra”, Wellesley-Cambridge Press, 1998.

[4] D. Wright, “Introduction to Linear Algebra”, The McGraw-Hill Companies, 1999.