

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

LIGHTRICKS LTD.,
Petitioner,

v.

PLOTAGRAPH, INC. and SASCHA CONNELLY,
Patent Owner.

Case IPR2023-00153
U.S. Patent No. 10,346,017
Filing Date: Aug. 13, 2018
Issue Date: July 9, 2019

DECLARATION OF PHILIP GREENSPUN, PH.D.

TABLE OF CONTENTS

I.	INTRODUCTION AND SCOPE OF WORK	32
II.	EXPERIENCE AND QUALIFICATIONS.....	32
III.	COMPENSATION	37
IV.	LEGAL CONSIDERATIONS	38
	A. Level of Ordinary Skill in the Art	38
	B. Claim Construction.....	39
V.	BASIS FOR OPINIONS	40
VI.	TECHNOLOGY BACKGROUND.....	40
VII.	OVERVIEW OF THE '017 PATENT	46
	A. Claim Construction.....	47
VIII.	ANALYSIS OF CLAIMS 1-18 OF THE '017 PATENT IN VIEW OF THE PRIOR ART	47
	A. Public Availability of AEM, IMU, Okabe, and Li.....	47
	B. Ground 1: AEM and Claims 1-18	51
	1. Summary of AEM.....	51
	a. Workspace and Panels	52
	b. Compositions and Layers	53
	c. Modifying Transparency of Certain Pixels	54
	d. Puppet Effect	56
	e. Animating with the Puppet Effect	57
	2. Example of Animating in AECS6	61
	a. Importing an image into a Composition.....	63
	b. Placing a Puppet Deform Pin to Create a Starting Keyframe	64
	c. Creating an Ending Keyframe and a Non-Linear Motion Path	67
	d. Using a Mask or the Roto Brush Tool to Select a Set of Pixels to be Animated	70
	i. Creating a Mask Using the “Pen” Tool.....	70

ii.	Creating a Mask Using the “Auto-Trace” Function.....	73
iii.	Creating a Matte Using the “Roto Brush” Tool.....	75
e.	Looping the Animation.....	83
f.	Final Resulting Animation.....	84
g.	Animating with Multiple Deform Pins.....	89
3.	Independent Claim 1	99
a.	[1pre]: A computer system for automating the shifting of pixels within a digital image, comprising: one or more processors; and one or more computer-readable media having stored thereon executable instructions that when executed by the one or more processors configure the computer system to perform at least the following.	99
b.	[1a]: access, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within the digital image.	100
c.	[1b]-[1c]: receive a first starting point through a user interface, wherein the first starting point is received through a user selection of a first beginning portion of the digital image; render a first visual indication of the first starting point on the user interface.....	100
d.	[1d]-[1e]: receive a first ending point through the user interface, wherein the first ending point is received through a user selection of a first ending portion of the digital image; render a second visual indication of the first ending point on the user interface.....	102

- e. [1f]: render, on the user interface, a first rendered link between the first starting point and the first ending point, wherein the first rendered link comprises: a first direction extending from the first starting point to the first ending point; and a first length between the first starting point and the first ending point.....104
- f. [1g]: identify a first set of pixels that are: parallel to the first rendered link, within a user-defined threshold distance from the first rendered link, and extending from the first starting point to the first ending point. 106
- g. [1h]: automatically shift the first set of pixels in the first direction, wherein shifting the first set of pixels comprises rendering and re-rendering in a loop the first set of pixels being shifted.....112

- h. [1i]-[1o]: receive a second starting point through the user interface, wherein the second starting point is received through a user selection of a second starting portion of the digital image, the second starting portion being different than both the first starting portion and the first ending portion; render a third visual indication of the second starting point on the user interface; receive, through the user interface, a second ending point associated with the second starting point, wherein the second ending point is received through a user selection of a second ending portion of the digital image, the second ending portion being different than both the first starting portion and the first ending portion; render a fourth visual indication of the second end point on the user interface; render, on the user interface, a second link between the second starting point and the second ending point; identify a second set of pixels that are: parallel to the second rendered link, within the user-defined threshold distance from the first rendered link, and extending from the second starting point to the second ending point; and automatically shift the second set of pixels between the second starting point and the second ending point, wherein shifting the second set of pixels comprises rendering in a loop the second set of pixels being shifted. 115
- 4. Claim 2.....118
 - a. [2a]-[2b]: The computer system of claim 1, wherein the executable instructions include instructions that are executable to configure the computer system to identify a particular portion of the digital image to mask, wherein the mask prevents pixels covered by the mask from being shifted. 118

5.	Claim 3: The computer system of claim 2, wherein receiving an indication of a particular portion of the digital image to mask comprises receiving through a user interface a selection of a particular pixel within the digital image.....	122
6.	Claim 4.....	123
a.	[4a]-[4c]: The computer system of claim 2, wherein the executable instructions include instructions that are executable to configure the computer system to generate the mask by: identifying one or more edges that form a boundary around the particular pixel; and generating the mask to cover area within the boundary.	123
7.	Claim 5: The computer system of claim 1, wherein the first direction is different from the second direction.....	126
8.	Claim 6: The computer system of claim 1, wherein the magnitude of the shifting of the first set of pixels is proportionally related to the first length and the magnitude of the shifting of the second set of pixels is proportionally related to the second length.....	126
9.	Claim 7.....	129
a.	[7a]: The computer system of claim 1, wherein: the first set of pixels comprise pixels that intersect the first rendered link; and.....	129
b.	[7b]: The first rendered link comprises a non-linear pathway extending between the first starting point and the first ending point.....	130
10.	Independent Claim 8:.....	131
a.	[8pre]: A method for automating the shifting of pixels within a digital image, comprising.....	131
b.	[8a]-[8b]: receiving a first indication of a first starting point through a user interface, wherein the first starting point is received through a user selection of a first starting portion of the digital image; render a first visual indication of the first starting point on the user interface.	132

- c. [8c]-[8d]: receiving, through the user interface, a first ending point associated with the first starting point, wherein the first ending point is received through a user selection of a first ending portion of the digital image; render a second visual indication of the first ending point on the user interface. 132
- d. [8e]: render, on the user interface, a first rendered link between the first starting point and the first ending point. 132
- e. [8f]: selecting a first set of pixels that intersect with the first rendered link.....132
- f. [8g]: automatically shifting the first set of pixels along the first rendered link between the first starting point and the first ending point, wherein shifting the first set of pixels comprises rendering in a loop the first set of pixels being shifted.....133

- g. [8h]-[8n]: receiving a second indication of a second starting point through the user interface, wherein the second starting point is received through a user selection of a second starting portion of the digital image, the second starting portion being different than both the first starting portion and the first ending portion; render a third visual indication of the second starting point on the user interface; receiving, through the user interface, a second ending point associated with the second starting point, wherein the second ending point is received through a user selection of a second ending portion of the digital image, the second ending portion being different than both the first starting portion and the first ending portion; render a fourth visual indication of the second end point on the user interface; render, on the user interface, a second rendered link between the second starting point and the second ending point; selecting a second set of pixels that intersect with the second rendered link; automatically shifting the second set of pixels along the second rendered link between the second starting point and the second ending point, wherein shifting the second set of pixels comprises rendering in a loop the second set of pixels being shifted.....134
- 11. Claim 9: The method as recited in claim 8, further comprising when a pixel selected from the first set of pixels reaches the ending point, rendering and re-rendering in the loop the pixel being shifted from the first starting point to the ending point.....135
- 12. Claim 10.....136
 - a. [10a]-[10b]: The method as recited in claim 8, further comprising receiving an indication to generate a mask over a portion of the digital image, wherein pixels under the mask are prevented from shifting. 136

13.	Claim 11: The method as recited in claim 10, further comprising receiving through a user interface a selection of a particular pixel within the digital image from which the mask should be generated.....	136
14.	Claim 12.....	136
a.	[12a]-[12b]: The method of claim 11, further comprising: identifying one or more edges that form a boundary around the particular pixel; and generating the mask to cover area within the boundary.	136
15.	Claim 13.....	137
a.	[13a]: The method of claim 8, wherein: the first set of pixels further comprise: pixels that are parallel to the first rendered link, within a user-defined threshold distance from the first rendered link, and that extend from the first starting point to the first ending point; and.....	137
b.	[13b]: the first rendered link comprises a non-linear pathway extending between the first starting point and the first ending point.....	137
16.	Independent Claim 14.....	137
a.	[14pre]: A method for automating the shifting of pixels within a digital image, comprising.....	137
b.	[14a]: accessing, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within the digital image.	137
c.	[14b]-[14c]: receiving a first starting point through a user interface, wherein the first starting point is received through a user selection of a first beginning portion of the digital image; rendering a first visual indication of the first starting point on the user interface.....	138

- d. [14d]-[14e]: receiving a first ending point through the user interface, wherein the first ending point is received through a user selection of a first ending portion of the digital image; rendering a second visual indication of the first ending point on the user interface.....138
- e. [14f]: rendering, on the user interface, a first rendered link between the first starting point and the first ending point, wherein the first link comprises: a non-linear pathway extending between the first starting point and the first ending point; and a first length extending along the non-linear pathway.....138
- f. [14g]: identifying a first set of pixels that lie along the first rendered link.....138
- g. [14h]: automatically shifting the first set of pixels along the non-linear pathway, wherein shifting the first set of pixels comprises rendering and re-rendering in a loop the first set of pixels being shifted. 139

h.	[14i]-[14o]: receiving a second starting point through the user interface, wherein the second starting point is received through a user selection of a second starting portion of the digital image, the second starting portion being different than both the first starting portion and the first ending portion; rendering a third visual indication of the second starting point on the user interface; receiving, through the user interface, a second ending point associated with the second starting point, wherein the second ending point is received through a user selection of a second ending portion of the digital image, the second ending portion being different than both the first starting portion and the first ending portion; rendering a fourth visual indication of the second end point on the user interface; rendering, on the user interface, a second link between the second starting point and the second ending point; selecting a second set of pixels along the second link; and automatically shifting the second set of pixels between the second starting point and the second ending point, wherein shifting the second set of pixels comprises rendering in a loop the second set of pixels being shifted.	140
17.	Claim 15	141
a.	[15a]-[15b]: The method of claim 14, further comprising identifying a particular portion of the digital image to mask, wherein the mask prevents pixels covered by the mask from being shifted.....	141
18.	Claim 16: The method of claim 15, wherein receiving an indication of a particular portion of the digital image to mask comprises receiving through a user interface a selection of a particular pixel within the digital image.	141
19.	Claim 17	141

a.	[17a]-[17c]: The method of claim 15, further comprising: generating the mask, wherein generating the mask comprises: identifying one or more edges that form a boundary around the particular pixel; and generating the mask to cover area within the boundary.	141
20.	Claim 18: The method of claim 15, wherein: the first set of pixels further comprise: pixels that are parallel to the first rendered link, within a user-defined threshold distance from the first rendered link, and that extend from the first starting point to the first ending point.....	142
C.	Ground 2: IMU and Okabe, and Claims 1-3, 5-11, 13-16, and 18	142
1.	Summary of IMU	142
a.	Applying Effects and Creating an Animation	143
b.	Modifying Transparency of Certain Pixels Using a Matte	146
2.	Summary of Okabe	147
3.	The IMU-Okabe Combination.....	148
a.	Motivation to Combine IMU with Okabe	148
b.	Resulting Combination of IMU with Okabe	151
4.	Independent Claim 1	153
a.	[1pre]: A computer system for automating the shifting of pixels within a digital image, comprising: one or more processors; and one or more computer-readable media having stored thereon executable instructions that when executed by the one or more processors configure the computer system to perform at least the following.	153
b.	[1a]: access, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within the digital image.	154

- c. [1b]-[1c]: receive a first starting point through a user interface, wherein the first starting point is received through a user selection of a first beginning portion of the digital image; render a first visual indication of the first starting point on the user interface.....155
- d. [1d]-[1e]: receive a first ending point through the user interface, wherein the first ending point is received through a user selection of a first ending portion of the digital image; render a second visual indication of the first ending point on the user interface.....157
- e. [1f]: render, on the user interface, a first rendered link between the first starting point and the first ending point, wherein the first rendered link comprises: a first direction extending from the first starting point to the first ending point; and a first length between the first starting point and the first ending point.....158
- f. [1g]: identify a first set of pixels that are: parallel to the first rendered link, within a user-defined threshold distance from the first rendered link, and extending from the first starting point to the first ending point. 160
- g. [1h]: automatically shift the first set of pixels in the first direction, wherein shifting the first set of pixels comprises rendering and re-rendering in a loop the first set of pixels being shifted.....161

- h. [1i]-[1o]: receive a second starting point through the user interface, wherein the second starting point is received through a user selection of a second starting portion of the digital image, the second starting portion being different than both the first starting portion and the first ending portion; render a third visual indication of the second starting point on the user interface; receive, through the user interface, a second ending point associated with the second starting point, wherein the second ending point is received through a user selection of a second ending portion of the digital image, the second ending portion being different than both the first starting portion and the first ending portion; render a fourth visual indication of the second end point on the user interface; render, on the user interface, a second link between the second starting point and the second ending point; identify a second set of pixels that are: parallel to the second rendered link, within the user-defined threshold distance from the first rendered link, and extending from the second starting point to the second ending point; and automatically shift the second set of pixels between the second starting point and the second ending point, wherein shifting the second set of pixels comprises rendering in a loop the second set of pixels being shifted. 164
- 5. Claim 2.....167
 - a. [2a]-[2b]: The computer system of claim 1, wherein the executable instructions include instructions that are executable to configure the computer system to identify a particular portion of the digital image to mask, wherein the mask prevents pixels covered by the mask from being shifted. 167

6.	Claim 3: The computer system of claim 2, wherein receiving an indication of a particular portion of the digital image to mask comprises receiving through a user interface a selection of a particular pixel within the digital image.....	168
7.	Claim 5: The computer system of claim 1, wherein the first direction is different from the second direction.....	169
8.	Claim 6: The computer system of claim 1, wherein the magnitude of the shifting of the first set of pixels is proportionally related to the first length and the magnitude of the shifting of the second set of pixels is proportionally related to the second length.....	170
9.	Claim 7.....	171
	a. [7a]: The computer system of claim 1, wherein: the first set of pixels comprise pixels that intersect the first rendered link; and.....	171
	b. [7b]: the first rendered link comprises a non-linear pathway extending between the first starting point and the first ending point.....	172
10.	Independent Claim 8:.....	174
	a. [8pre]: A method for automating the shifting of pixels within a digital image, comprising.....	174
	b. [8a]-[8b]: receiving a first indication of a first starting point through a user interface, wherein the first starting point is received through a user selection of a first starting portion of the digital image; render a first visual indication of the first starting point on the user interface.	174
	c. [8c]-[8d]: receiving, through the user interface, a first ending point associated with the first starting point, wherein the first ending point is received through a user selection of a first ending portion of the digital image; render a second visual indication of the first ending point on the user interface.	175
	d. [8e]: render, on the user interface, a first rendered link between the first starting point and the first ending point.	175

- e. [8f]: selecting a first set of pixels that intersect with the first rendered link.....175
 - f. [8g]: automatically shifting the first set of pixels along the first rendered link between the first starting point and the first ending point, wherein shifting the first set of pixels comprises rendering in a loop the first set of pixels being shifted.....175
 - g. [8h]-[8n]: receiving a second indication of a second starting point through the user interface, wherein the second starting point is received through a user selection of a second starting portion of the digital image, the second starting portion being different than both the first starting portion and the first ending portion; render a third visual indication of the second starting point on the user interface; receiving, through the user interface, a second ending point associated with the second starting point, wherein the second ending point is received through a user selection of a second ending portion of the digital image, the second ending portion being different than both the first starting portion and the first ending portion; render a fourth visual indication of the second end point on the user interface; render, on the user interface, a second rendered link between the second starting point and the second ending point; selecting a second set of pixels that intersect with the second rendered link; automatically shifting the second set of pixels along the second rendered link between the second starting point and the second ending point, wherein shifting the second set of pixels comprises rendering in a loop the second set of pixels being shifted.....176
11. Claim 9: The method as recited in claim 8, further comprising when a pixel selected from the first set of pixels reaches the ending point, rendering and re-rendering in the loop the pixel being shifted from the first starting point to the ending point.....177

12.	Claim 10.....	178
a.	[10a]-[10b]: The method as recited in claim 8, further comprising receiving an indication to generate a mask over a portion of the digital image, wherein pixels under the mask are prevented from shifting.	178
13.	Claim 11: The method as recited in claim 10, further comprising receiving through a user interface a selection of a particular pixel within the digital image from which the mask should be generated.....	178
14.	Claim 13.....	178
a.	[13a]: The method of claim 8, wherein: the first set of pixels further comprise: pixels that are parallel to the first rendered link, within a user-defined threshold distance from the first rendered link, and that extend from the first starting point to the first ending point; and.....	178
b.	[13b]: the first rendered link comprises a non-linear pathway extending between the first starting point and the first ending point.....	178
15.	Independent Claim 14:.....	179
a.	[14pre]: A method for automating the shifting of pixels within a digital image, comprising.....	179
b.	[14a]: accessing, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within the digital image.	179
c.	[14b]-[14c]: receiving a first starting point through a user interface, wherein the first starting point is received through a user selection of a first beginning portion of the digital image; rendering a first visual indication of the first starting point on the user interface.....	179

- d. [14d]-[14e]: receiving a first ending point through the user interface, wherein the first ending point is received through a user selection of a first ending portion of the digital image; rendering a second visual indication of the first ending point on the user interface.....179
- e. [14f]: rendering, on the user interface, a first rendered link between the first starting point and the first ending point, wherein the first link comprises: a non-linear pathway extending between the first starting point and the first ending point; and a first length extending along the non-linear pathway.....180
- f. [14g]: identifying a first set of pixels that lie along the first rendered link.....180
- g. [14h]: automatically shifting the first set of pixels along the non-linear pathway, wherein shifting the first set of pixels comprises rendering and re-rendering in a loop the first set of pixels being shifted. 180

- h. [14i]-[14o]: receiving a second starting point through the user interface, wherein the second starting point is received through a user selection of a second starting portion of the digital image, the second starting portion being different than both the first starting portion and the first ending portion; rendering a third visual indication of the second starting point on the user interface; receiving, through the user interface, a second ending point associated with the second starting point, wherein the second ending point is received through a user selection of a second ending portion of the digital image, the second ending portion being different than both the first starting portion and the first ending portion; rendering a fourth visual indication of the second end point on the user interface; rendering, on the user interface, a second link between the second starting point and the second ending point; selecting a second set of pixels along the second link; and automatically shifting the second set of pixels between the second starting point and the second ending point, wherein shifting the second set of pixels comprises rendering in a loop the second set of pixels being shifted. 181
- 16. Claim 15.....182
 - a. [15a]-[15b]: The method of claim 14, further comprising identifying a particular portion of the digital image to mask, wherein the mask prevents pixels covered by the mask from being shifted.....182
- 17. Claim 16: The method of claim 15, wherein receiving an indication of a particular portion of the digital image to mask comprises receiving through a user interface a selection of a particular pixel within the digital image. 182
- 18. Claim 18: The method of claim 15, wherein: the first set of pixels further comprise: pixels that are parallel to the first rendered link, within a user-defined threshold distance from the first rendered link, and that extend from the first starting point to the first ending point.....182

D.	Ground 3: IMU, Okabe, and Li, and Claims 2-4, 10-12, and 15-17	182
1.	Summary of Li	182
2.	The IMU-Okabe-Li Combination	184
a.	Motivation to Combine the IMU-Okabe Combination with Li.....	184
b.	Resulting Combination of the IMU-Okabe Combination and Li	187
3.	Claim 2	188
a.	[2a]-[2b]: The computer system of claim 1, wherein the executable instructions include instructions that are executable to configure the computer system to identify a particular portion of the digital image to mask, wherein the mask prevents pixels covered by the mask from being shifted.	188
4.	Claim 3: The computer system of claim 2, wherein receiving an indication of a particular portion of the digital image to mask comprises receiving through a user interface a selection of a particular pixel within the digital image.....	189
5.	Claim 4.....	190
a.	[4a]-[4c]: The computer system of claim 2, wherein the executable instructions include instructions that are executable to configure the computer system to generate the mask by: identifying one or more edges that form a boundary around the particular pixel; and generating the mask to cover area within the boundary.	190
6.	Claim 10.....	190
a.	[10a]-[10b]: The method as recited in claim 8, further comprising receiving an indication to generate a mask over a portion of the digital image, wherein pixels under the mask are prevented from shifting.	190

7.	Claim 11: The method as recited in claim 10, further comprising receiving through a user interface a selection of a particular pixel within the digital image from which the mask should be generated.....	191
8.	Claim 12	191
a.	[12a]-[12b]: The method of claim 11, further comprising: identifying one or more edges that form a boundary around the particular pixel; and generating the mask to cover area within the boundary.	191
9.	Claim 15	191
a.	[15a]-[15b]: The method of claim 14, further comprising identifying a particular portion of the digital image to mask, wherein the mask prevents pixels covered by the mask from being shifted.....	191
10.	Claim 16: The method of claim 15, wherein receiving an indication of a particular portion of the digital image to mask comprises receiving through a user interface a selection of a particular pixel within the digital image.	191
11.	Claim 17	192
a.	[17a]-[17c]: The method of claim 15, further comprising: generating the mask, wherein generating the mask comprises: identifying one or more edges that form a boundary around the particular pixel; and generating the mask to cover area within the boundary.	192
IX.	CONCLUSION.....	192

LIST OF DOCUMENTS CONSIDERED

Exhibit	Shorthand	Description
1001	'017 Patent	U.S. Patent No. 10,346,017
1003	AEM	Adobe® After Effects® Help and tutorials, Adobe (2013)
1004	IMU-Home	Wayback Machine Capture dated Mar. 27, 2012 of Anthony Thyssen, <i>Examples of ImageMagick Usage (Version 6)</i> , ImageMagick (Mar. 15, 2011), http://www.imagemagick.org/Usage/ [https://web.archive.org/web/20120327064501/http://www.imagemagick.org/Usage/]
1005	IMU-Distorting	WayBack Machine Capture dated Mar. 29, 2012 of Anthony Thyssen, <i>ImageMagick v6 Examples - - Distorting Images</i> , ImageMagick (Mar. 21, 2012), http://www.imagemagick.org/Usage/distorts/ [https://web.archive.org/web/20120329131929/http://www.imagemagick.org/Usage/distorts/]
1006	IMU-Masking	Wayback Machine Capture dated Sept. 28, 2012 of Anthony Thyssen, <i>ImageMagick v6 Examples - - Masks</i> , ImageMagick (Mar. 10, 2011), http://www.imagemagick.org/Usage/masking/ [https://web.archive.org/web/20120928070642/http://www.imagemagick.org/Usage/masking/]
1007	IMU-Animating ¹	Wayback Machine Capture dated Mar. 10, 2012 of Anthony Thyssen, <i>ImageMagick v6 Examples - - Animation Basics</i> , ImageMagick (Feb. 8, 2011), http://www.imagemagick.org/Usage/anim_basics/ [https://web.archive.org/web/20120310193613/http://www.imagemagick.org/Usage/anim_basics/]
1008	IMU-Windows	Wayback Machine Capture dated Apr. 5, 2012 of Anthony Thyssen, <i>ImageMagick v6 Examples -- Usage under Windows</i> , ImageMagick (Mar. 21, 2012),

¹ IMU-Home, IMU-Warping, IMU-Distorting, IMU-Masking, IMU-Animating, and IMU-Windows are collectively referred to hereinafter as “IMU.”

		http://www.imagemagick.org/Usage/windows/ [https://web.archive.org/web/20120405151502/http://www.imagemagick.org/Usage/windows/]
1009	Okabe	Makato Okabe, et al., Creating Fluid Animation from a Single Image using Video Database, 30 Computer Graphics Forum 1973 (Nov. 4, 2011)
1010	Li	Yin Li, et al., Lazy Snapping, 23 ACM Transactions on Graphics 303 (Aug. 1, 2004)
1011	'017 PH	Prosecution history of U.S. Patent No. 10,346,017
1012	Archive	Declaration of Nathaniel E Frank-White of the Internet Archive
1022		Plaintiffs' Unopposed Sworn Motion for Leave to Extend Time to Comply with P.R. 3-1 and P.R. 3-2 Dislcousre [sic] of Asserted Claims and Preliminary Infringement Contentions and Make Document Production, <i>Plotagraph, Inc. v. Lightricks Ltd.</i> , Civil Action No. 4:21-cv-03873, Dkt. No. 38 (S.D. Tex. May 5, 2022)

CLAIM LISTING

Claim	Limitation No.	Limitation
1	[1pre]	A computer system for automating the shifting of pixels within a digital image, comprising: one or more processors; and one or more computer-readable media having stored thereon executable instructions that when executed by the one or more processors configure the computer system to perform at least the following:
	[1a]	access, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within the digital image;
	[1b]	receive a first starting point through a user interface, wherein the first starting point is received through a user selection of a first beginning portion of the digital image;
	[1c]	render a first visual indication of the first starting point on the user interface;
	[1d]	receive a first ending point through the user interface, wherein the first ending point is received through a user selection of a first ending portion of the digital image;
	[1e]	render a second visual indication of the first ending point on the user interface;
	[1f]	render, on the user interface, a first rendered link between the first starting point and the first ending point, wherein the first rendered link comprises: a first direction extending from the first starting point to the first ending point; and a first length between the first starting point and the first ending point;
	[1g]	identify a first set of pixels that are: parallel to the first rendered link,

Claim	Limitation No.	Limitation
		within a user-defined threshold distance from the first rendered link, and extending from the first starting point to the first ending point;
	[1h]	automatically shift the first set of pixels in the first direction, wherein shifting the first set of pixels comprises rendering and re-rendering in a loop the first set of pixels being shifted;
	[1i]	receive a second starting point through the user interface, wherein the second starting point is received through a user selection of a second starting portion of the digital image, the second starting portion being different than both the first starting portion and the first ending portion;
	[1j]	render a third visual indication of the second starting point on the user interface;
	[1k]	receive, through the user interface, a second ending point associated with the second starting point, wherein the second ending point is received through a user selection of a second ending portion of the digital image, the second ending portion being different than both the first starting portion and the first ending portion;
	[1l]	render a fourth visual indication of the second end point on the user interface;
	[1m]	render, on the user interface, a second link between the second starting point and the second ending point;
	[1n]	identify a second set of pixels that are: parallel to the second rendered link, within the user-defined threshold distance from the first rendered link, and

Claim	Limitation No.	Limitation
		extending from the second starting point to the second ending point; and
	[1o]	automatically shift the second set of pixels between the second starting point and the second ending point, wherein shifting the second set of pixels comprises rendering in a loop the second set of pixels being shifted.
2	[2a]	The computer system of claim 1, wherein the executable instructions include instructions that are executable to configure the computer system to identify a particular portion of the digital image to mask,
	[2b]	wherein the mask prevents pixels covered by the mask from being shifted.
3	The computer system of claim 2, wherein receiving an indication of a particular portion of the digital image to mask comprises receiving through a user interface a selection of a particular pixel within the digital image.	
4	[4a]	The computer system of claim 2, wherein the executable instructions include instructions that are executable to configure the computer system to generate the mask by:
	[4b]	identifying one or more edges that form a boundary around the particular pixel; and
	[4c]	generating the mask to cover area within the boundary.
5	The computer system of claim 1, wherein the first direction is different from the second direction.	
6	The computer system of claim 1, wherein the magnitude of the shifting of the first set of pixels is proportionally related to the first length and the magnitude of the shifting of the second set of pixels is proportionally related to the second length.	
7	[7a]	The computer system of claim 1, wherein: the first set of pixels comprise pixels that intersect the first rendered link; and

Claim	Limitation No.	Limitation
	[7b]	the first rendered link comprises a non-linear pathway extending between the first starting point and the first ending point.
8	[8pre]	A method for automating the shifting of pixels within a digital image, comprising:
	[8a]	receiving a first indication of a first starting point through a user interface, wherein the first starting point is received through a user selection of a first starting portion of the digital image;
	[8b]	render a first visual indication of the first starting point on the user interface;
	[8c]	receiving, through the user interface, a first ending point associated with the first starting point, wherein the first ending point is received through a user selection of a first ending portion of the digital image;
	[8d]	render a second visual indication of the first ending point on the user interface;
	[8e]	render, on the user interface, a first rendered link between the first starting point and the first ending point;
	[8f]	selecting a first set of pixels that intersect with the first rendered link;
	[8g]	automatically shifting the first set of pixels along the first rendered link between the first starting point and the first ending point, wherein shifting the first set of pixels comprises rendering in a loop the first set of pixels being shifted;
	[8h]	receiving a second indication of a second starting point through the user interface, wherein the second starting point is received through a user selection of a second starting portion of the digital image, the second starting portion being different than both the first starting portion and the first ending portion;
	[8i]	render a third visual indication of the second starting point on the user interface;
	[8j]	receiving, through the user interface, a second ending point associated with the second starting

Claim	Limitation No.	Limitation
		point, wherein the second ending point is received through a user selection of a second ending portion of the digital image, the second ending portion being different than both the first starting portion and the first ending portion;
	[8k]	render a fourth visual indication of the second end point on the user interface;
	[8l]	render, on the user interface, a second rendered link between the second starting point and the second ending point;
	[8m]	selecting a second set of pixels that intersect with the second rendered link;
	[8n]	automatically shifting the second set of pixels along the second rendered link between the second starting point and the second ending point, wherein shifting the second set of pixels comprises rendering in a loop the second set of pixels being shifted.
9	The method as recited in claim 8, further comprising when a pixel selected from the first set of pixels reaches the ending point, rendering and re-rendering in the loop the pixel being shifted from the first starting point to the ending point.	
10	[10a]	The method as recited in claim 8, further comprising receiving an indication to generate a mask over a portion of the digital image,
	[10b]	wherein pixels under the mask are prevented from shifting.
11	The method as recited in claim 10, further comprising receiving through a user interface a selection of a particular pixel within the digital image from which the mask should be generated.	
12	[12a]	The method of claim 11, further comprising: identifying one or more edges that form a boundary around the particular pixel; and
	[12b]	generating the mask to cover area within the boundary.
13	[13a]	The method of claim 8, wherein: the first set of pixels further comprise:

Claim	Limitation No.	Limitation
		pixels that are parallel to the first rendered link, within a user-defined threshold distance from the first rendered link, and that extend from the first starting point to the first ending point; and
	[13b]	the first rendered link comprises a non-linear pathway extending between the first starting point and the first ending point.
14	[14pre]	A method for automating the shifting of pixels within a digital image, comprising:
	[14a]	accessing, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within the digital image;
	[14b]	receiving a first starting point through a user interface, wherein the first starting point is received through a user selection of a first beginning portion of the digital image;
	[14c]	rendering a first visual indication of the first starting point on the user interface;
	[14d]	receiving a first ending point through the user interface, wherein the first ending point is received through a user selection of a first ending portion of the digital image;
	[14e]	rendering a second visual indication of the first ending point on the user interface;
	[14f]	rendering, on the user interface, a first rendered link between the first starting point and the first ending point, wherein the first link comprises: a non-linear pathway extending between the first starting point and the first ending point; and a first length extending along the non-linear pathway;

Claim	Limitation No.	Limitation
	[14g]	identifying a first set of pixels that lie along the first rendered link;
	[14h]	automatically shifting the first set of pixels along the non-linear pathway, wherein shifting the first set of pixels comprises rendering and re-rendering in a loop the first set of pixels being shifted;
	[14i]	receiving a second starting point through the user interface, wherein the second starting point is received through a user selection of a second starting portion of the digital image, the second starting portion being different than both the first starting portion and the first ending portion;
	[14j]	rendering a third visual indication of the second starting point on the user interface;
	[14k]	receiving, through the user interface, a second ending point associated with the second starting point, wherein the second ending point is received through a user selection of a second ending portion of the digital image, the second ending portion being different than both the first starting portion and the first ending portion;
	[14l]	rendering a fourth visual indication of the second end point on the user interface;
	[14m]	rendering, on the user interface, a second link between the second starting point and the second ending point;
	[14n]	selecting a second set of pixels along the second link; and
	[14o]	automatically shifting the second set of pixels between the second starting point and the second ending point, wherein shifting the second set of pixels comprises rendering in a loop the second set of pixels being shifted.
15	[15a]	The method of claim 14, further comprising identifying a particular portion of the digital image to mask,
	[15b]	wherein the mask prevents pixels covered by the mask from being shifted.

Claim	Limitation No.	Limitation
16		The method of claim 15, wherein receiving an indication of a particular portion of the digital image to mask comprises receiving through a user interface a selection of a particular pixel within the digital image.
17	[17a]	The method of claim 15, further comprising: generating the mask, wherein generating the mask comprises:
	[17b]	identifying one or more edges that form a boundary around the particular pixel; and
	[17c]	generating the mask to cover area within the boundary.
18		The method of claim 15, wherein: the first set of pixels further comprise: pixels that are parallel to the first rendered link, within a user-defined threshold distance from the first rendered link, and that extend from the first starting point to the first ending point.

I. INTRODUCTION AND SCOPE OF WORK

1. My name is Philip Greenspun. I am over the age of twenty-one (21) years, of sound mind, capable of making the statements set forth in this declaration, and competent to testify about the matters set forth below. All the facts and statements contained in this declaration are within my personal knowledge, and they are, in all things, true and correct.

2. I have been retained as an expert witness on behalf of Lightricks Ltd. (“Petitioner”) to provide my opinions and views on the materials I have reviewed related to U.S. Patent No. 10,346,017 (the “’017 Patent”), and the scientific and technical knowledge regarding that subject matter. I understand that Petitioner has filed a Petition for *Inter Partes* Review (“IPR”) arguing that all claims of the ’017 Patent are unpatentable. I have been asked to provide expert opinions on the issues relating to this IPR, which I address below.

II. EXPERIENCE AND QUALIFICATIONS

3. I am a salaried employee of Fifth Chance Media LLC, which I understand is being compensated for my work in this matter. I am not an owner of Fifth Chance Media LLC, and my compensation is not contingent on the outcome of this matter or the specifics of my testimony. Fifth Chance Media LLC is being compensated for my work as an expert on an hourly basis. My compensation is not dependent on the outcome of these proceedings or the content of my opinions.

4. My resumé is attached as Attachment A. In terms of my background and experiences that qualify me as an expert in this case, I earned a Ph.D. in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology (“MIT”) in 1999. I also obtained a Bachelor of Science Degree in Mathematics from MIT in 1982 and a Master of Science Degree in Electrical Engineering and Computer Science from MIT in 1993.

5. My Ph.D. thesis concerned the engineering of large online Internet communities with a Web browser front-end and a relational database management system (RDBMS) containing site content and user data. This work was substantially based on building and operating the photo.net online community, a site where photography enthusiasts reviewed tutorials, many of which I authored, uploaded their own photos for display and discussion, and exchanged questions and answers. The thesis included a chapter on using digital image processing tools, including Adobe Photoshop, as well as batch-processing tools based on ImageMagick (see below) for handling 100 or more photos in an automated pipeline.

6. I have authored five computer science textbooks in total, including Database Backed Web Sites (Macmillan), Software Engineering for Internet Applications (MIT Press), and a SQL language tutorial.

7. I have served as an independent member of various advisory and corporate boards, mostly for technology companies. For example, I joined the

corporate board of an MIT materials science spin-off in late 2005 during a \$550,000 seed capital phase. I stepped down when the company secured \$10 million in venture capital in mid-2007.

8. I have previously served as an expert witness for Amazon.com, Ford Motor Company, IBM, Microsoft, Oracle, Samsung, Canon, and Google, among others, in patent cases. I have been retained as a patent expert by the U.S. Department of Justice in a trade matter and by the Commonwealth of Massachusetts in a criminal matter in which software was at issue.

9. I began working full-time as a computer programmer in 1978, developing a database management system for the Pioneer Venus Orbiter at the National Aeronautics and Space Administration's Goddard Space Flight Center.

10. I began working in the area of computer graphics in 1983, specifically on primitives for the Symbolics Operating System as well as a video game for that computer. I began working with digital video and digital video editing programs in the 1990s and was using Adobe Premiere continuously starting in the 2000s. *See, e.g., Philip Greenspun, Suggestions for video editing computer* (June 6, 2010), <https://philip.greenspun.com/blog/2010/06/06/suggestions-for-video-editing-computer/>.

11. I began working on systems that rendered 2D screens from 3D models in 1984 while developing the ICAD computer-aided mechanical design system for

the Symbolics Lisp Machine. Also for the Lisp Machine, I developed a 2D anti-tank warfare simulator for the U.S. Department of Defense (through Textron). I also developed a computer system for supporting civil engineering, especially earthmoving, that included a digital three-dimensional map. The latter system was the topic of my Master's thesis at MIT and also U.S. Patents 5,150,310 and 5,964,298 ("Integrated civil engineering and earthmoving system").

12. In 1995, I led an effort by Hearst Corporation to set up an infrastructure for Internet applications across all their newspaper, magazine, radio, and television properties. This infrastructure included software for managing users, shopping carts, electronic commerce, advertising, and user tracking. The software that I designed managed images for both editorial and advertising.

13. Between 1995 and 1997, I significantly expanded the photo.net online community that I had started in 1993 to help people teach each other to become better photographers. I began distributing the source code behind photo.net to other programmers as a free open-source toolkit called "ArsDigita Community System."

14. The photo.net site enabled users to upload photos as attachments to discussion forum postings and also, beginning in 1999, included a complete photo-sharing system along the lines of Flickr. The system included some server-based image processing capabilities, e.g., to produce thumbnail images from full-size images.

15. In May 1997, Macmillan published my first textbook on Internet Application development, *Database Backed Web Sites*. This book includes a chapter on processing images, including with Adobe and ImageMagick software, for inclusion within Web sites.

16. In 1997, I started a company, ArsDigita, to provide support and service for the free open-source toolkit based on photo.net. Between 1997 and the middle of 2000, I managed the growth of ArsDigita to 80 people, almost all programmers, and \$20 million per year in annual revenue. This involved supervising dozens of software development projects, nearly all of which were Internet Applications with a Web front-end and an Oracle RDBMS back-end. The typical project also involved handling images and image processing.

17. Between 2000 and the present, I have done software development projects for philip.greenspun.com and photo.net, two online services that are implemented as relational database management applications. In addition, I developed postclipper.com, a database-backed Web application that works in conjunction with Facebook to allow parents to produce electronic baby books based on photographs previously included in Facebook posts.

18. Separately from this commercial and public work, I have been involved, as a part-time teacher within the Department of Electrical Engineering and Computer Science, educating students at MIT in how to develop Internet

Applications with an RDBMS back-end. In the Spring of 1999, I taught 6.916 Software Engineering of Innovative Web Services with Professors Hal Abelson and Michael Dertouzos. In the Spring of 2002, this course was adopted into the standard MIT curriculum as 6.171. I wrote 15 chapters of a new textbook for this class, *Software Engineering for Internet Applications*. This book was published on the Web at <http://philip.greenspun.com/seia/> starting in 2002 and 2003 and also in hardcopy from MIT Press in 2006. I am the sole author of a supplementary textbook for the class, *SQL for Web Nerds*, a succinct SQL programming language tutorial available only on the Web at <http://philip.greenspun.com/sql/>. I am also one of the creators and teachers of a three-day intensive course in developing database applications. We teach this class periodically at MIT.

19. I periodically teach a database programming class at Harvard Medical School. Students have access to a relational database of more than 5 billion insurance claims and write SQL programs to try to identify correlations and trends. I taught this course most recently in March 2021. In the fall of 2021, I taught an Information Security class at Florida Atlantic University. The most recent course that I have taught is an aeronautical engineering class at MIT in January 2023.

III. COMPENSATION

20. My work on this matter is being billed at \$550/hour. Also, Fifth Chance Media is being reimbursed for reasonable and necessary expenses incurred in

relation to my services. Fifth Chance Media's compensation is not dependent on my testimony or the outcome of this or any other proceeding.

IV. LEGAL CONSIDERATIONS

21. I am not a lawyer, and I offer no legal opinions. For the purposes of this declaration, I have been informed by counsel for Petitioner about certain aspects of the law that are relevant to my analysis, as summarized below.

A. Level of Ordinary Skill in the Art

22. I understand from Petitioner's counsel that the claims of the '017 Patent, the teachings from the prior art, and the related issues I address below must be considered from the perspective of a person of ordinary skill in the art ("POSITA") at the time of the earliest claimed priority date ("ECPD"), which I have been asked by counsel for Petitioner to assume, for purposes of this IPR, is July 28, 2016.

23. I have also been advised that a POSITA is a hypothetical person to whom the claimed subject matter pertains with the capability of understanding the scientific and engineering principles applicable to the pertinent art. I understand that the following factors may be considered in determining the level of ordinary skill: type of problems encountered in the art; prior art solutions to those problems; speed with which innovations are made; sophistication of the technology; and educational

level of active workers in the field. I also understand that not every factor may be present and that one or more factors may predominate.

24. In my opinion, a POSITA as of the ECPD would have had at least a bachelor's degree in computer science, electrical engineering, or a related field, and at least 1-2 years of experience in image processing and animation. Less education could have been compensated with more experience, and vice versa. A POSITA would have also been familiar with existing systems for image processing and animation and would have understood how to implement such systems. The '017 Patent and the prior art discussed in Section VIII evidence this level of ordinary skill.

25. Under this definition, I at least possessed ordinary skill in the art at the ECPD. I have applied this definition of a POSITA in rendering my opinions below.

B. Claim Construction

26. I am informed that a claim term is given the meaning that the term would have to a POSITA at the time of the invention, which generally is the ordinary and customary meaning of the term. I further understand that the ordinary and customary meaning of a term may be evidenced by a variety of sources, including the words of the claims themselves, the specification, the prosecution history, and extrinsic evidence concerning relevant scientific principles, the meaning of technical terms, and the state of the art.

V. BASIS FOR OPINIONS

27. My opinions are based on my education, training, and experience as well as items that I reviewed to prepare my opinions, including the '017 Patent and at least the publications listed in the table at the beginning of this declaration.

28. My opinions address what would have been logical to, and within the skill level of, a POSITA at the ECPD, given the state of the relevant art, the knowledge and skill that a POSITA would have, the teachings of the references discussed below, and how a POSITA would have understood those teachings. My opinions also address whether a POSITA would have had a reasonable expectation of the modified systems discussed below successfully functioning in their modified forms as discussed below.

29. I have not been asked to take a position on whether a given claim would have been legally anticipated or otherwise obvious to a POSITA at the ECPD, but I have been told that some or all of my opinions are being used to support the argument that claims of the '017 Patent are anticipated or otherwise obvious.

VI. TECHNOLOGY BACKGROUND

30. Interactive computing with a command-line interface dates to the 1960s and was an improvement on batch computing with decks of punched cards. The command-line interface became widely familiar to consumers in 1978 with Apple

DOS on the Apple II and in 1981 with MS-DOS—the operating system included on the IBM PC.

31. Bit-mapped or “raster” graphics computer displays, which replaced vector graphics terminals, were developed in the 1960s and became more popular as the cost of memory fell. By 1977, the Apple II was available to consumers and included the ability to display color graphics from an area of its memory.² “Computer science curriculum for high school students,” *ACM SIGCSE Bulletin* 12:1 (1980), page 172, describes students in an introductory high school class programming 2D looping animations in the PASCAL language on the Apple II. Packaged animation programs enabled consumers to create animations without programming. Fantavision, an example mid-1980s program, can be seen in operation in 2010 YouTube videos.³

32. Conventional mid-80s hardware and low-level software for displaying computer-generated images is described in “VAXstation: A General-Purpose Raster Graphics Architecture,” *ACM Transactions on Graphics* 3:1 (1984), page 70, which

² https://www.si.edu/object/nmah_334638.

³ E.g., Highretrogamelord, *Fantavision for the Apple II [Part 01 \ 02]*, YouTube (Dec. 5, 2010), <https://youtu.be/k4ysfd8r0fA>; Highretrogamelord, *Fantavision for the Apple II [Part 02 \ 02]*, YouTube (Dec. 5, 2010), https://youtu.be/vSo5_2TB91E.

notes that “high-resolution bit-mapped raster displays, as pioneered on the Xerox PARC Alto computer, have now become standard on many personal computers and workstations.” The paper describes an operation to move a group of pixels as “the fundamental operation of the display system.” “VAXstation: A General-Purpose Raster Graphics Architecture” at 75.

33. The earliest graphical user interfaces predate bitmapped graphics. One may be seen in a 1963 demonstration of Ivan Sutherland’s Sketchpad program on a vector graphics display.⁴ The familiar modern windows, icons, menus, and pointer (“WIMP”) interface is generally dated to the 1973 and the Xerox Alto computer, many of whose ideas were popularized for consumers in the Apple Macintosh (1984) and Microsoft Windows (1985). One of the application programs shipped with the first Macintosh was MacDraw, which enabled users to create and manipulate drawings on the screen.

34. Although professionals had access to film scanners in the 1980s, the typical consumer did not have a library of personal digital photos in that decade. The 1992 Kodak PhotoCD system enabled the bulk conversion of images on film to high-resolution digital files. I myself was a user of the PhotoCD system starting in

⁴ Interactive Chronicles, *Ivan Sutherland Sketchpad Demo 1963*, YouTube (May 30, 2012), https://youtu.be/6orsmFndx_o.

late 1993, processing the files with ImageMagick (first released in 1990). Professional photographers began to originate digital images in the late 1980s, while the first consumer digital cameras were products of the 1990s and the first mobile phones with built-in cameras arrived in the late 1990s. Today's familiar touch-screen smartphone with an included camera was pioneered in 2007 with the first Apple iPhone.

35. Almost as soon as there were digital images, there were digital image editing programs. A system from Bell Labs is described in a 1987 paper, "PICO-A Picture Editor," *AT&T Technical Journal* 66:2 (1987), page 2. PICO includes the capability of transforming images. Introduced originally in 1990, Photoshop version 3.0 was released in 1994 and included layers and the capability of generating animated GIFs.⁵ Consumer-targeted image editing applications appeared in the 1990s as well. A November 18, 1997, press release notes that 5 million copies of Adobe PhotoDeluxe had been shipped to consumers. A November 24, 1997, press

⁵ Rik Fairlie, *A Look Back at 20 Years of Photoshop* (Feb. 18, 2010, 12:23 PM), <https://gadgetwise.blogs.nytimes.com/2010/02/18/a-look-back-at-20-years-of-photoshop/> [<https://archive.nytimes.com/gadgetwise.blogs.nytimes.com/2010/02/18/a-look-back-at-20-years-of-photoshop/>]

release from Microsoft, on the other hand, claimed that the company's Picture It! 2.0 was the market leader in both sales and capability.

36. George Wolberg, *Digital Image Warping* (1990) describes underlying algorithms for transforming portions of digital photographs. Applications of some of these techniques are described in Thaddeus Beier & Shawn Neely, "Feature-Based Image Metamorphosis," *SIGGRAPH '92: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques* 1992), page 35, and Peter Litwinowicz & Lance Williams, "Animating Images with Drawings," *SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (1994), page 409.

37. Adding motion to images was being done with standard desktop software, such as Adobe After Effects, first released in 1993, no later than 1999. "Cycore's Cult Effects Filters To Be Offered Free With Adobe After Effects 4.1," *PR Newswire Europe*, September 10, 1999, describes "CE Noise Turbulent: fractals can be used to describe many real world objects that do not have simple geometric shapes. You can animate all fractals with full control. Great for simulating anything from caustics to clouds, from lava to flowing water or gas."

38. Yung-Yu Chuang et al., "Animating Pictures with Stochastic Motion Textures," *ACM Transactions on Graphics* 24:3 (2005), page 853 ("Chuang"), explains "we explore the problem of enhancing still pictures with subtly animated

motions” and describes a looping two-dimensional displacement. The user assists the software by segmenting “the scene into a set of animatable layers and assigns certain parameters to each one.” Masks are used to divide up the image. From Chuang at page 855:

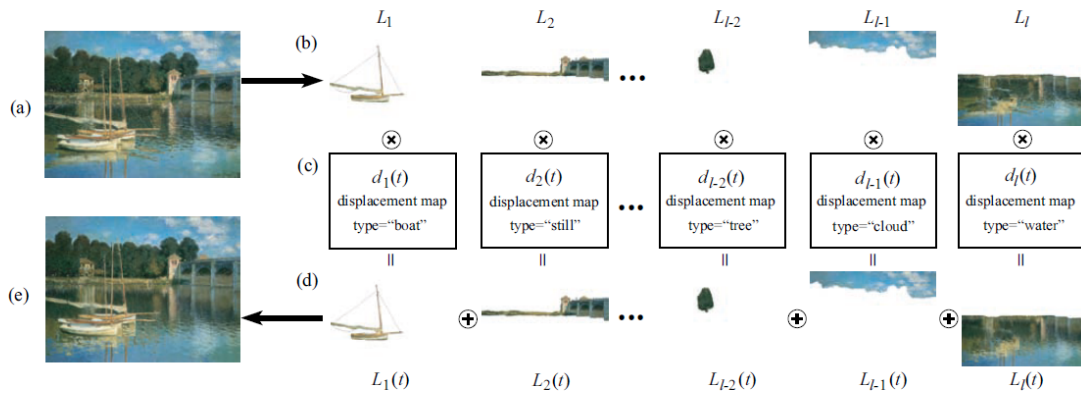


Figure 2 Overview of our system. The input still image (a) is manually segmented into several layers (b). Each layer L_i is then animated with a different stochastic motion texture $d_i(t)$ (c). Finally, the animated layers $L_i(t)$ (d) are composited back together to produce the final animation $I(t)$ (e).

39. Each layer may be annotated with “a line segment,” and the “motion texture” for each layer can be different. Chuang at page 855. Although the system has the capability of implementing complex motions as much of the paper is concerned with, it can also move pixels: “Since clouds often move very slowly and their motion does not attract too much attention, we simply assign a translational motion field to them.” Chuang at page 858.

40. In summary, graphical user interfaces, algorithms, and software for adding motion to a static image had been developed in the 1990s.

VII. OVERVIEW OF THE '017 PATENT

41. The '017 Patent explains that known “digital image software tools” can be used to “edit or otherwise manipulate digital images,” such as “to incorporate movement within a digital image.” '017 Patent, 1:35-40, 1:50-52. The '017 Patent itself is directed to “interfaces and tools for automating the shifting of pixels within a digital image.” '017 Patent, 3:34-36, Abst. To use such interfaces and tools, a user provides a “starting point” on a digital image, as well as “a direction associated with the starting point.” '017 Patent, Abst. A set of pixels that “extend[s] in the direction away from the starting point” is then selected and shifted in the first direction, where the shifting comprises “rendering and re-rendering in a loop the set of pixels being shifted.” '017 Patent, Abst. Figure 7 illustrates the claimed invention:

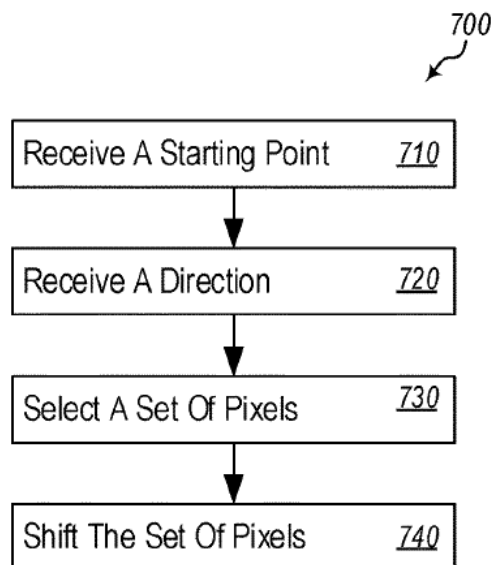


Fig. 7

'017 Patent, Fig. 7.

A. Claim Construction

42. For purposes of this IPR, I apply the plain and ordinary meanings of all claim terms in the '017 Patent.

VIII. ANALYSIS OF CLAIMS 1-18 OF THE '017 PATENT IN VIEW OF THE PRIOR ART

A. Public Availability of AEM, IMU, Okabe, and Li

43. AEM is a user manual for the Adobe After Effects CS6 software program (“AECS6”). *See, e.g.*, AEM, 3. In my personal experience, AECS6 was a publicly available animation software that was popular in the art by the ECPD. It was included in the Adobe Creative Suite bundle, and I had a copy installed on my desktop computer by the ECPD. Its user manual, AEM, was likewise publicly available to AECS6 users by the ECPD, when it was downloadable from Adobe’s website. I consulted the Internet Archive and confirmed that this was true through the following URL of an archived Wayback Machine capture of the Adobe website: https://web.archive.org/web/20120907012238/https://helpx.adobe.com/pdf/after_effects_reference.pdf. This URL shows that AEM was publicly available for download from Adobe’s website by September 7, 2012—I understand how to read the URL from Archive at ¶ 5. I downloaded AEM from this URL and provided counsel with the copy of AEM that I understand is used as an exhibit in this

proceeding. AEM matches the copy at Archive, 99-699. Indeed, the above URL matches that found in Archive, 98.

44. IMU contains Wayback Machine captures dated 2012 of the ImageMagick.com website, specifically the website's section titled "Examples of ImageMagick Usage (Version 6)," which provides guidance and examples on how to use the website's ImageMagick Version 6 ("IMV6") software. *See* IMU-Home, 1. The section's homepage (IMU-Home) contains links to different subpages explaining how to use IMV6's various effects and capabilities, including "Distorting Images" (IMU-Distorting), "Masking and Background Removal" (IMU-Masking), "Animation Basics" (IMU-Animating), and "Usage under Windows" (IMU-Windows).⁶ IMU-Home, 1-2. In my experience, IMV6 was an opensource and publicly available image processing software that was popular in the art by the ECPD. Likewise, the above guiding webpages on IMV6 provided by the ImageMagick.com website were each publicly available for viewing by IMV6 users on the ImageMagick.com website by the ECPD. I consulted the Internet Archive

⁶ IMU is thus a single reference. Separately, a POSITA would have also been motivated to consider the IMU webpages together in order to gain a more complete understanding of IMV6.

and confirmed that this was true through the following URLs of archived Wayback Machine captures of the ImageMagick.com website:

- <https://web.archive.org/web/20120327064501/http://www.imagemagick.org/Usage/>. This URL corresponds to an archived version of the “Examples of ImageMagick Usage (Version 6)” homepage discussed above and shows that this homepage was publicly available for viewing on the ImageMagick.com website by March 27, 2012. I printed the webpage at this URL and provided counsel with the printout that I understand is used as the “IMU-Home” exhibit in this proceeding.
- <https://web.archive.org/web/20120329131929/http://www.imagemagick.org/Usage/distorts/>. This URL corresponds to an archived version of the “Distorting Images” subpage linked on the “Examples of ImageMagick Usage (Version 6)” homepage. *See* IMU-Home, 1. This URL shows that this subpage was publicly available for viewing on the ImageMagick.com website by March 29, 2012. I printed the webpage at this URL and provided counsel with the printout that I understand is used as the “IMU-Distorting” exhibit in this proceeding.
- <https://web.archive.org/web/20120928070642/http://www.imagemagick.org/Usage/masking/>. This URL corresponds to an archived version of the “Masking and Background Removal” subpage linked on the “Examples of

ImageMagick Usage (Version 6)” homepage. *See* IMU-Home, 1. This URL shows that this subpage was publicly available for viewing on the ImageMagick.com website by September 28, 2012. I printed the webpage at this URL and provided counsel with the printout that I understand is used as the “IMU-Masking” exhibit in this proceeding.

- https://web.archive.org/web/20120310193613/http://www.imagemagick.org/Usage/anim_basics/. This URL corresponds to an archived version of the “Animation Basics” subpage linked on the “Examples of ImageMagick Usage (Version 6)” homepage. *See* IMU-Home, 1. This URL shows that this subpage was publicly available for viewing on the ImageMagick.com website by March 10, 2012. I printed the webpage at this URL and provided counsel with the printout that I understand is used as the “IMU-Animating” exhibit in this proceeding.

- <https://web.archive.org/web/20120405151502/http://www.imagemagick.org/Usage/windows/>. This URL corresponds to an archived version of the “Usage under Windows” subpage linked on the “Examples of ImageMagick Usage (Version 6)” homepage. *See* IMU-Home, 2. This URL shows that this subpage was publicly available for viewing on the ImageMagick.com website by April 5, 2012. I printed the webpage at this URL and provided counsel

with the printout that I understand is used as the “IMU-Windows” exhibit in this proceeding.

IMU-Home, IMU-Distorting, IMU-Masking, IMU-Animating, IMU-Windows match, respectively, the copies at Archive, 5-8, 10-44, 46-64, 66-78, and 80-96. Indeed, the URLs for these exhibits match, respectively, those found in Archive, 4, 9, 45, 65, and 79.

45. Okabe was published in Volume 30, Number 7 of Computer Graphics Forum in September 2011 and was thus publicly available to those in the art by that date. Okabe, 1.

46. Li was published in Volume 23, Issue 3 of ACM Transactions on Graphics in August 2004 and was thus publicly available to those in the art by that date. Li, 1.

B. Ground 1: AEM and Claims 1-18

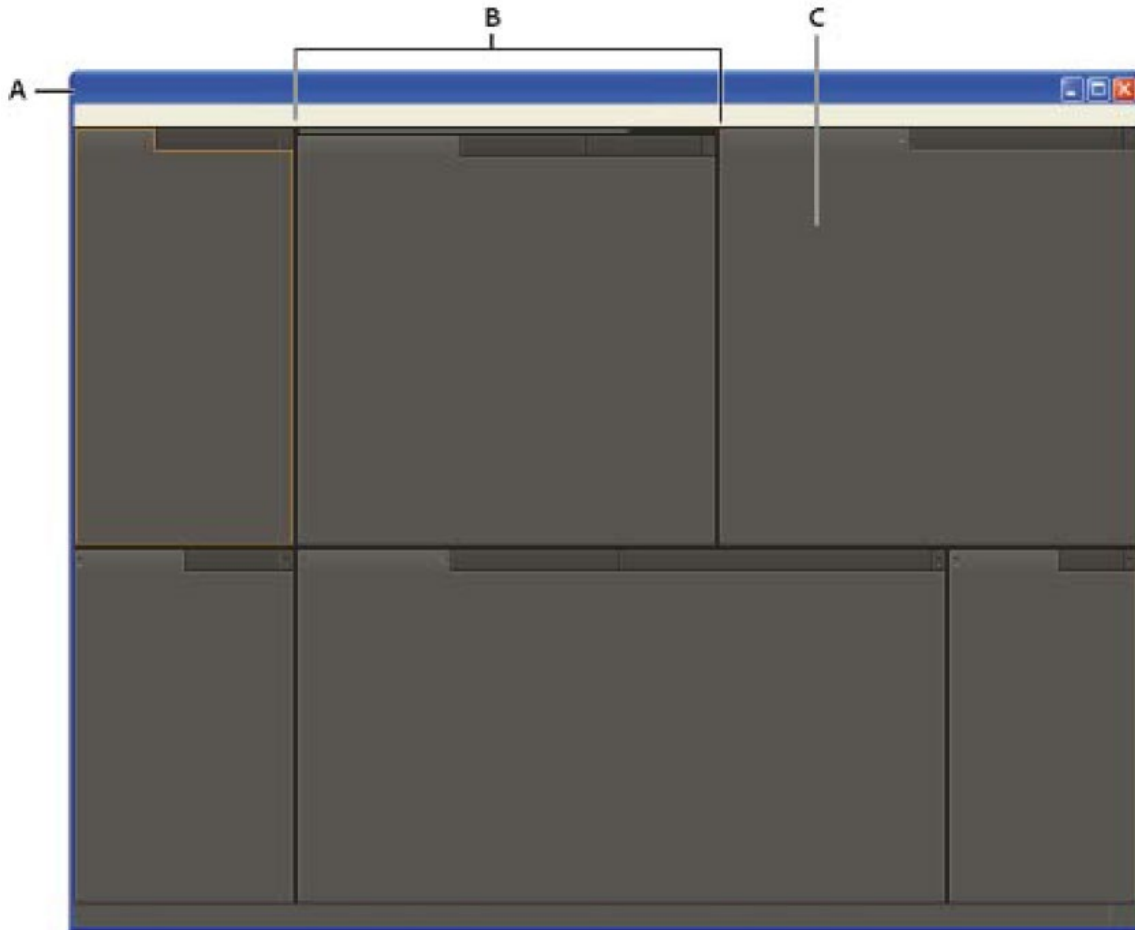
1. Summary of AEM

47. AEM is a user manual for AECS6—software that can perform a variety of digital image animation tasks, e.g., “animat[ing] a simple title” to “creat[ing] complex motion graphics, or composit[ing] realistic visual effects.” AEM, 25. AEM explains how to navigate AECS6’s workspace and panels, define “compositions” and “layers,” apply a “mask” or “matte” to a digital image, and animate the image using the “Puppet” effect. Each is discussed herein. In doing so,

AEM teaches that, using AECS6's user interface, a user can import an image, specify which pixels of the image are nontransparent or transparent using a mask or matte, and apply the Puppet effect to move only the nontransparent pixels of the image along a user-defined path. Further, such movement of the image's nontransparent pixels can be played in a loop.

a. Workspace and Panels

48. According to AEM, AECS6's user interface comprises an application window housing "panels" organized in a "workspace." AEM, 39. Different panels contain different tools and effects for editing and animating images. *See, e.g.*, AEM, 25 (discussing the general workflow using AECS6's "Composition panel" and "Timeline panel"). AEM depicts an example workspace containing several panels:



Example workspace

A. *Application window* **B.** *Grouped panels* **C.** *Individual panel*

AEM, 39.

b. Compositions and Layers

49. AEM explains that a composition is the “framework” for animation in AECS6. AEM, 75. “Footage items,” such as images, are imported into a composition. AEM, 75, 99-101 (listing exemplary footage item file formats, including “Still-image formats” such as JPEG, PNG, and PDF). Once imported, each footage item becomes a “layer” in that composition. AEM, 75, 98.

50. Layers are “the elements that make up a composition.” AEM, 120. Layers, as implied, are stacked in a “vertical arrangement” such that the uppermost layer is visible. AEM, 125.

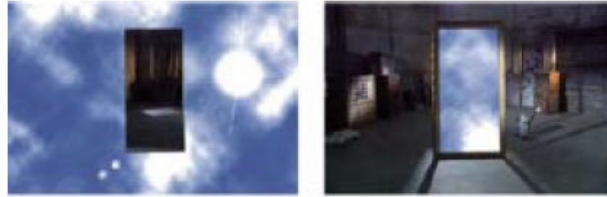
51. Additionally, “[AECS6] includes a variety of effects, which you apply to layers to add or modify characteristics of still images, video, and audio.” AEM, 335. Such effects are previewed and applied to a composition in the “Composition panel,” or to a single layer in the “Layer panel.” AEM, 75, 120, 335. When ready, the user “*render[s]* [the] composition to create the frames of a final output movie.” AEM, 75.

c. Modifying Transparency of Certain Pixels

52. AEM describes tools that modify the transparency of a layer’s pixels—and thus the visibility of effects applied to the layer. “You can make portions of a layer transparent using any of several features in After Effects,” including “masks” and “mattes.” AEM, 315. Such masks and mattes modify the layer’s “alpha channel,” which “determines the transparency of the layer at each pixel.” AEM, 318.

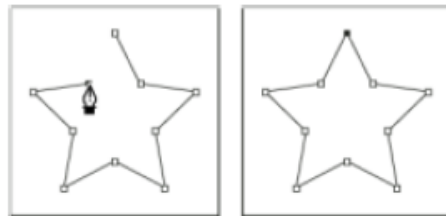
53. Regarding masks, AEM explains that a user applies a mask to all or a portion of a layer so that pixels enclosed within the mask are made nontransparent, while the remaining pixels are made transparent, or vice versa. AEM, 318. AEM

provides an example composition illustrating this functionality using two layers and a rectangular mask:



Default behavior for a drawn mask (left); same mask inverted (right)

AEM, 318. The user creates a mask by using, e.g., AECS6's "Pen" tool to click on different points on the layer in the "Composition" or "Layer panel[s]" to specify the path and vertices of the mask. AEM, 264-66. AEM provides an example of creating a star-shaped mask using the Pen tool:



Clicking with Pen tool creates straight segments.

AEM, 265. Masks can also be automatically created for a layer using AECS6's "Auto-trace" function, which creates a mask by "searching for edges" across the entire layer and tracing such edges. AEM, 262. Such edges are detected based on "the alpha, red, green, blue, or luminance channel of [the] layer." AEM, 262.

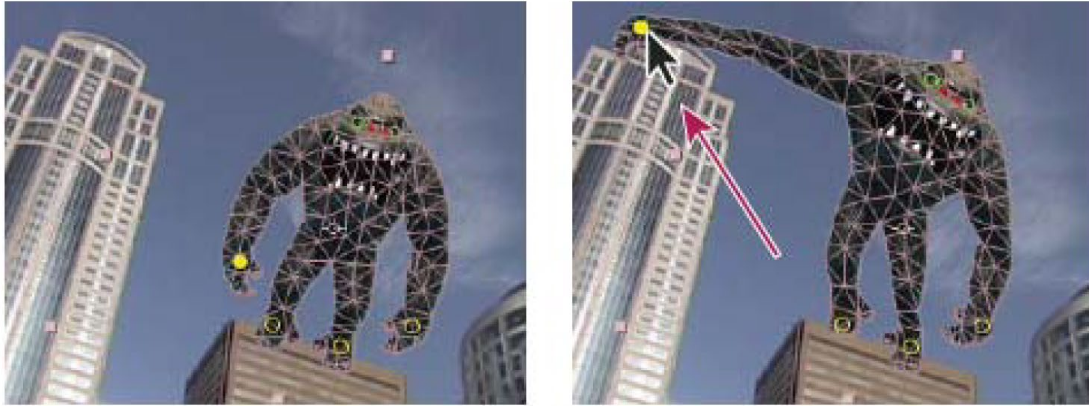
54. Mattes similarly "define[]" the transparent areas" of a layer. AEM, 317. Specifically, a matte isolates a layer's "foreground" from its "background," making the latter transparent. AEM, 328. The user creates a matte by using, e.g., AECS6's

“Roto Brush” tool to draw “strokes” in the “Layer panel” over “representative areas of the foreground and background elements.” AEM, 328. AECS6 then uses “Edge Detection” to determine a “segmentation boundary” separating the foreground and background elements drawn over by the user’s strokes. AEM, 328-31.

d. Puppet Effect

55. One available effect is AECS6’s “Puppet” effect, which “deform[s] part of an image according to the positions of pins that you place and move.” AEM, 218. A user first places a Puppet “Deform” pin on the layer via the “Composition” or “Layer panel[s],” specifically on a “nontransparent pixel” in a portion of the layer to be moved. AEM, 219. AECS6 then creates an “outline” by “auto-tracing the alpha channel of [the] layer”—i.e., outlining the nontransparent pixels of the layer. AEM, 219, 220-21. This outline is “automatically divided into a mesh of triangles,” where “[e]ach part of the mesh is also associated with the pixels of the image, so the pixels move with the mesh.” AEM, 218. Thus, when the user repositions the Deform pin, “the mesh changes shape to accommodate this movement, while keeping the overall mesh as rigid as possible,” resulting in a corresponding movement of the layer’s nontransparent pixels. AEM, 218. The user can add more Deform pins to move other portions of the layer and its mesh. AEM, 218.

56. In one example, AEM depicts a foreground layer containing an image of a gorilla, which is stacked over a background layer containing several buildings:



Mesh created by placing Deform pins (left), and result of dragging a Deform pin

AEM, 218. A Deform pin has been placed on each of the gorilla’s limbs, and a mesh has been created from the nontransparent pixels of the gorilla. AEM, 218. As shown, the Deform pin on the gorilla’s right arm is repositioned, causing that arm to move, while the other three Deform pins are not repositioned, causing the other limbs to remain “as rigid as possible” during the animation. *See* AEM, 218; *see also* AEM, 219 (“For example, when animating a person waving, add a pin to each foot to hold them to the ground, and add a pin to the waving hand.”).

e. Animating with the Puppet Effect

57. A user not only can apply the Puppet effect to move a portion of a layer but can also animate this movement. AEM, 218-19. To do so, AEM instructs the user to define a starting time point and starting position for a Deform pin, and then define an ending time point and ending position for the Deform pin. AEM, 177, 219. This is known as defining starting and ending “keyframes” for the Deform pin, respectively. AEM, 177, 219. AECS6 then interpolates the Deform pin’s position

from the starting time and position (the “starting keyframe”) to the ending time and position (the “ending keyframe”) and generates “in-between” frames based on the interpolation, resulting in an animated movement of the layer. AEM, 177, 193, 219. This is known as “keyframe interpolation.” AEM, 177, 193. Note that the terminology in the hand-drawn days of animation was similarly “key drawing” or simply “key” and “in-between.”⁷

58. To define a starting “keyframe” for a Deform pin, AEM instructs the user to specify a time point within the animation, and then place the Deform pin on a nontransparent pixel of the layer in the “Composition” or “Layer panel[s].” AEM, 218-19; *see also* AEM, 166 (describing the “current-time indicator (CTI)” in the “Timeline panel,” which is used to specify the current animation time point). AECS6 then automatically creates the starting keyframe for the Deform pin based on both the specified time point and the Deform pin’s starting position. AEM, 218-19.

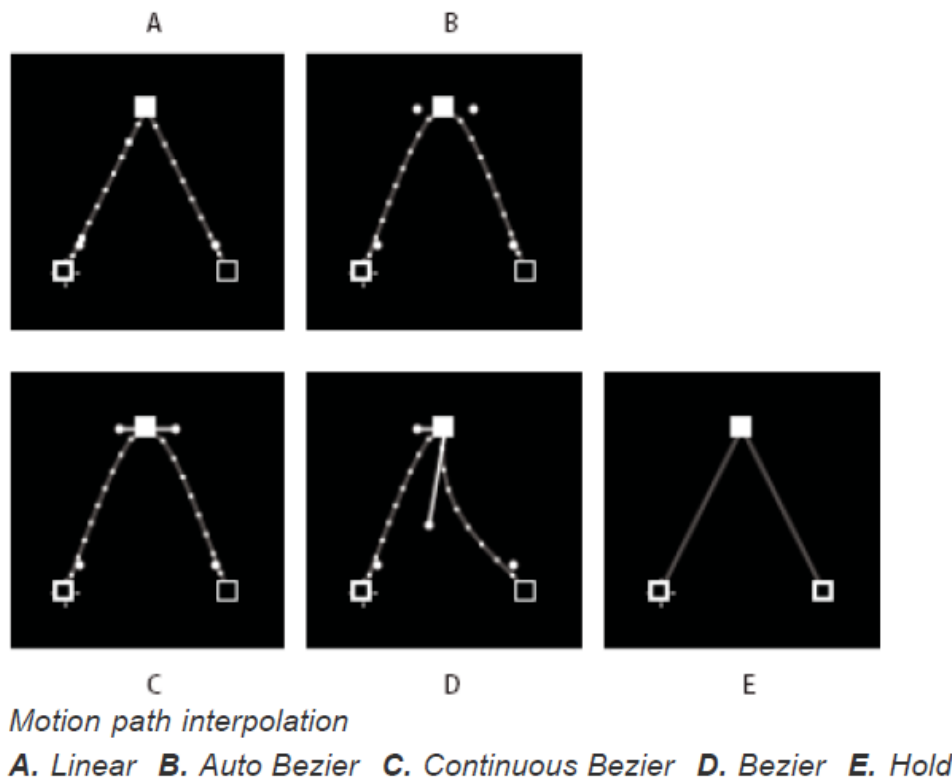
⁷ *See, e.g., The Illusion Of Life: Disney Animation* (1981), “The Principles of Animation” chapter, available at <https://archive.org/details/TheIllusionOfLifeDisneyAnimation/>.

59. AEM then instructs the user to define an ending keyframe by “[g]o[ing] to another time in the composition, and mov[ing] the position of... the Deform pin[] by dragging [it] in the Composition or Layer panel.” AEM, 219.

60. Once the keyframes for a Deform pin have been specified, AECS6 will perform keyframe interpolation by interpolating the position of the Deform pin from the starting to ending keyframe and generating animation frames to animate the layer’s corresponding movement. AEM, 193 (“You set keyframes to specify a property’s values at certain key times. After Effects interpolates values for the property for all times between keyframes.”), 218-19 (“[A] keyframe is set or modified each time that you change the position of a Deform pin.”). AECS6 performs such interpolation for all keyframes of a Deform pin. AEM, 193, 219. The user may place additional Deform pins to animate other parts of the layer in the same way. AEM, 219.

61. Further, keyframe interpolation of a Deform pin’s change in position follows a user-defined path—a “motion path.” AEM, 187, 219. A motion path is visually indicated in both the “Composition” and “Layer panel[s]” as “a sequence of dots, where each dot marks the position of the [Deform pin] at each frame. A box in the path marks the position of a keyframe.” AEM, 187, 219. A user selects between using a linear or non-linear motion path for interpolation. AEM, 194-95. With a linear motion path—i.e., “Linear Interpolation”—the position of the Deform

pin is interpolated linearly from the starting to ending keyframe. AEM, 194. With a non-linear motion path—e.g., “Bezier interpolation”—the user can adjust the motion path to include “any combination of curves and straight lines,” and the position of the Deform pin will follow that path when interpolated from the starting to ending keyframe. AEM, 194-95. AEM provides an example illustrating the available linear and non-linear motion paths:



AEM, 193.

62. Additionally, a Deform pin’s keyframe interpolation can be looped such that a layer’s movement is animated in a repeating loop. This is done by applying a “loopOut()” expression to the Deform pin, which, by default, causes “all keyframes [to] loop.” AEM, 562 (explaining that, when using the default inputs for

the “loopOut()” expression, “all keyframes will loop”), 219 (“You can use expressions to link the positions of Deform pins to motion tracking data, audio amplitude keyframes, or any other properties.”). Thus, after the Deform pin reaches the ending position/keyframe, its position will be reset to the starting position/keyframe, and keyframe interpolation will repeat. AEM, 562, 219. This loop repeats until the user specifies otherwise. AEM, 562, 219.

2. Example of Animating in AECS6

63. To further illustrate the animation capabilities of AECS6, I provide below a number of annotated screenshots of AECS6’s user interface, as well as accompanying explanations, showing the step-by-step creation of an exemplary composition comprising an image of a smokestack that has been animated using the features of AECS6 discussed above in Section VIII.B.1.⁸

⁸ A nearly identical example of animating smoke in an image in Adobe After Effects 2020 can also be viewed on YouTube. See Blackbronx, *How To Animate a Still Photo in After Effects*, YouTube (Apr. 2, 2020), https://youtu.be/L_d1Wo-hmoQ. Although this video was published in 2020 (after the ECPD) and uses a 2020 version of After Effects to animate the image rather than AECS6, the video may nevertheless be helpful to those unfamiliar with how such animations are created in AECS6, as

64. The specific version of AECS6 used to generate the screenshots was AECS6 version 11.0.4.2:



65. The software was installed on a 2012 Mac running macOS High Sierra version 10.13.6:

AECS6 includes all of the same features of After Effects 2020 that are discussed in the video, e.g., the Puppet effect and masks.

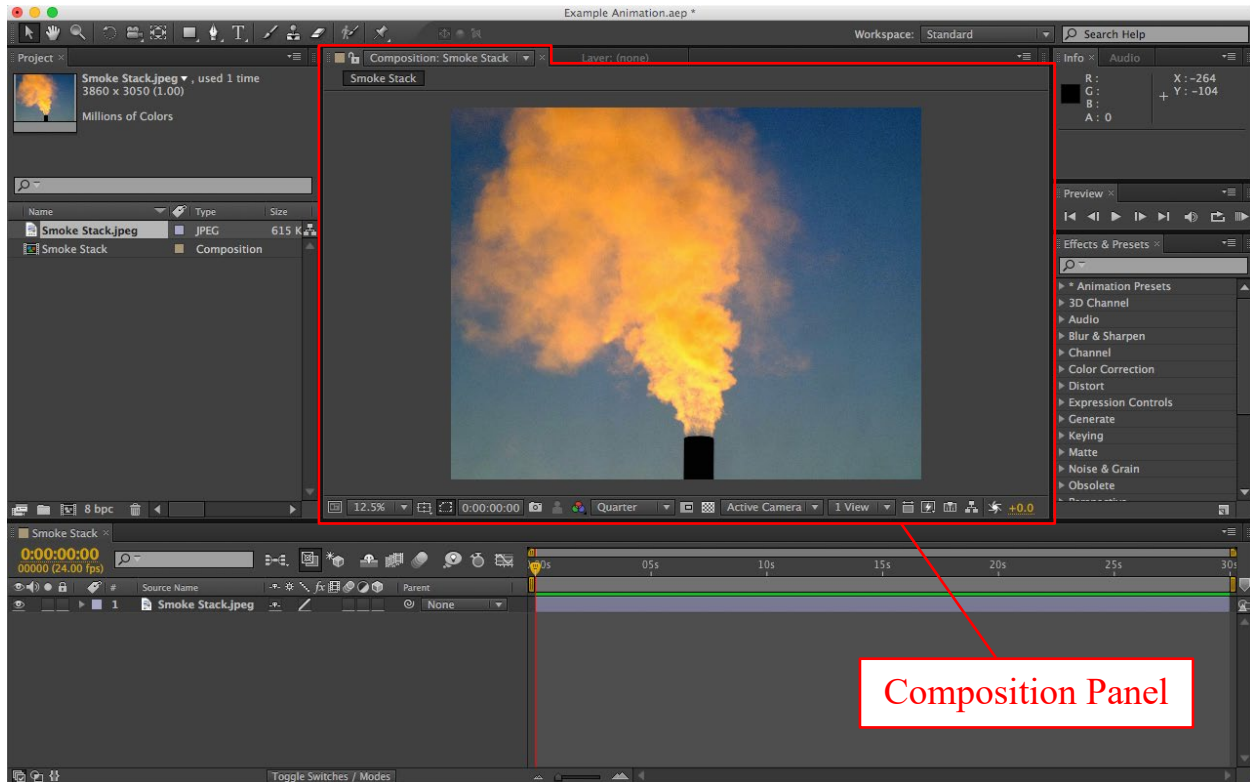


66. I have been informed by counsel that, although AECS6—including screenshots of AECS6—cannot be used as a basis of the grounds in the Petition because AECS6 does not qualify as either a patent or printed publication under 35 U.S.C. §311(b), such screenshots can nevertheless be used for establishing the background knowledge possessed by a POSITA, including demonstrating what a POSITA would have known about AECS6’s animation capabilities, especially in light of AEM’s teachings. The below screenshots and accompanying explanations are thus provided for this specific and limited purpose.

a. Importing an image into a Composition

67. To begin, the below screenshot of AECS6’s user interface shows a still JPEG image of a smokestack that has been imported into an exemplary composition to form a layer in that composition. *See* AEM, 75, 99. The center panel within the

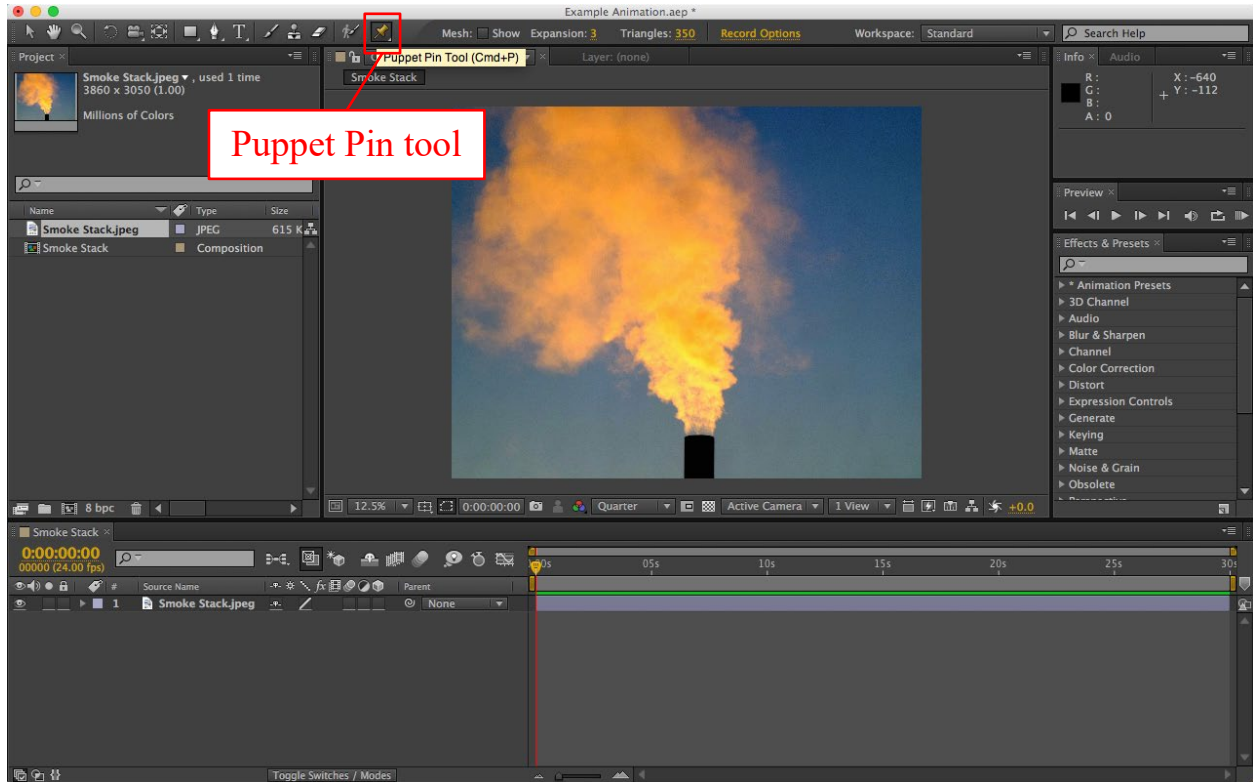
user interface (indicated in the annotations) is the “Composition Panel,” which provides a preview of the layer within the composition and, as later screenshots also show, enables the layer to be manually modified using AECS6’s effects. *See* AEM, 75, 120, 335.

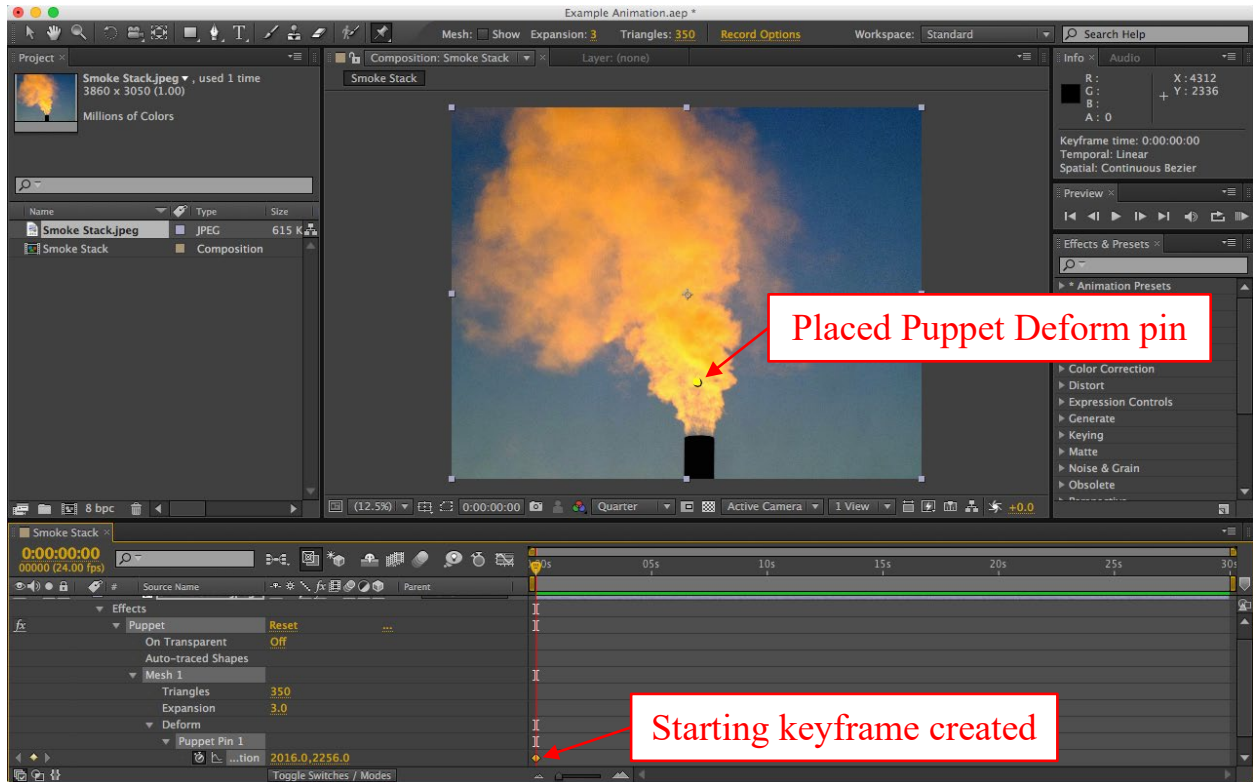


b. Placing a Puppet Deform Pin to Create a Starting Keyframe

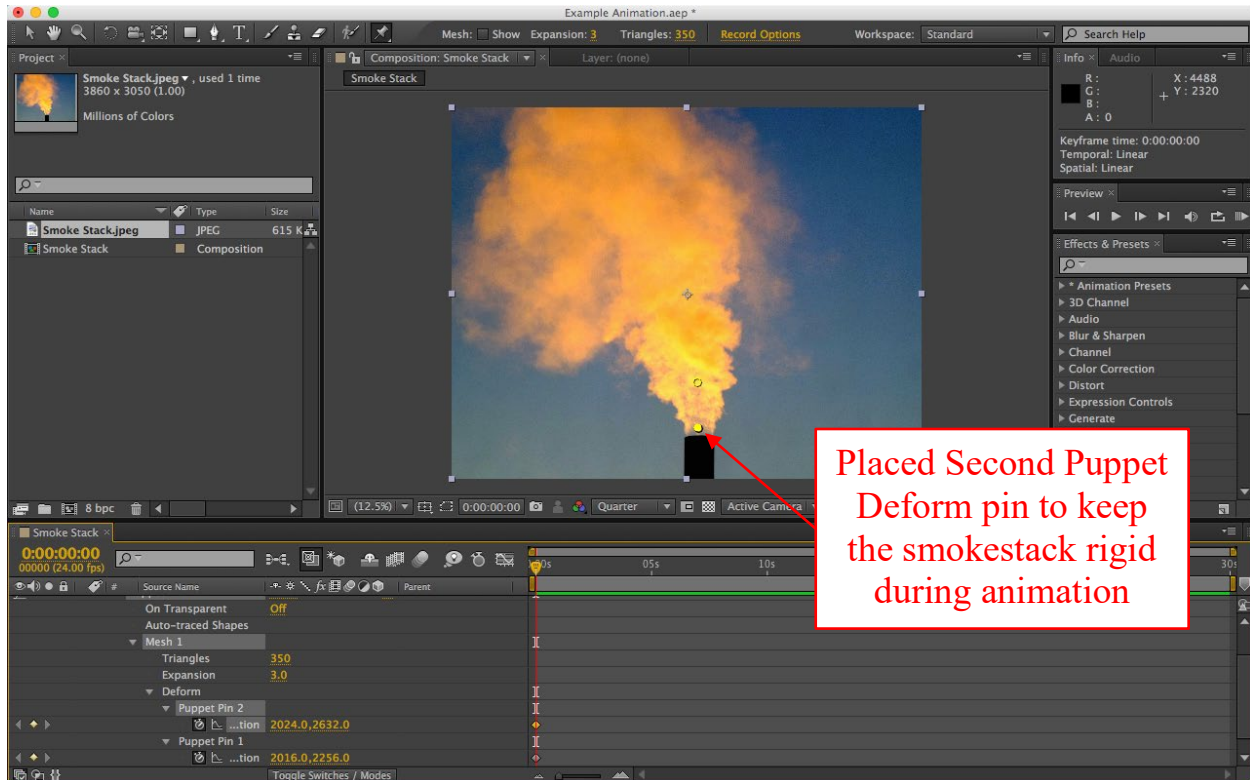
68. As shown in the next two screenshots, AECS6’s Puppet effect is applied to the layer using the Puppet Pin tool, beginning with the placement of a Puppet Deform pin within the “Composition panel” specifically on the portion of the layer to be moved and animated—i.e., the billowing smoke in the exemplary composition. *See* AEM, 218-19. Such placement of the Deform pin automatically

creates a starting keyframe for the Deform pin based on the Deform pin's placed location and the starting time of the animation (0:00:00:00). *See AEM, 218-19.*



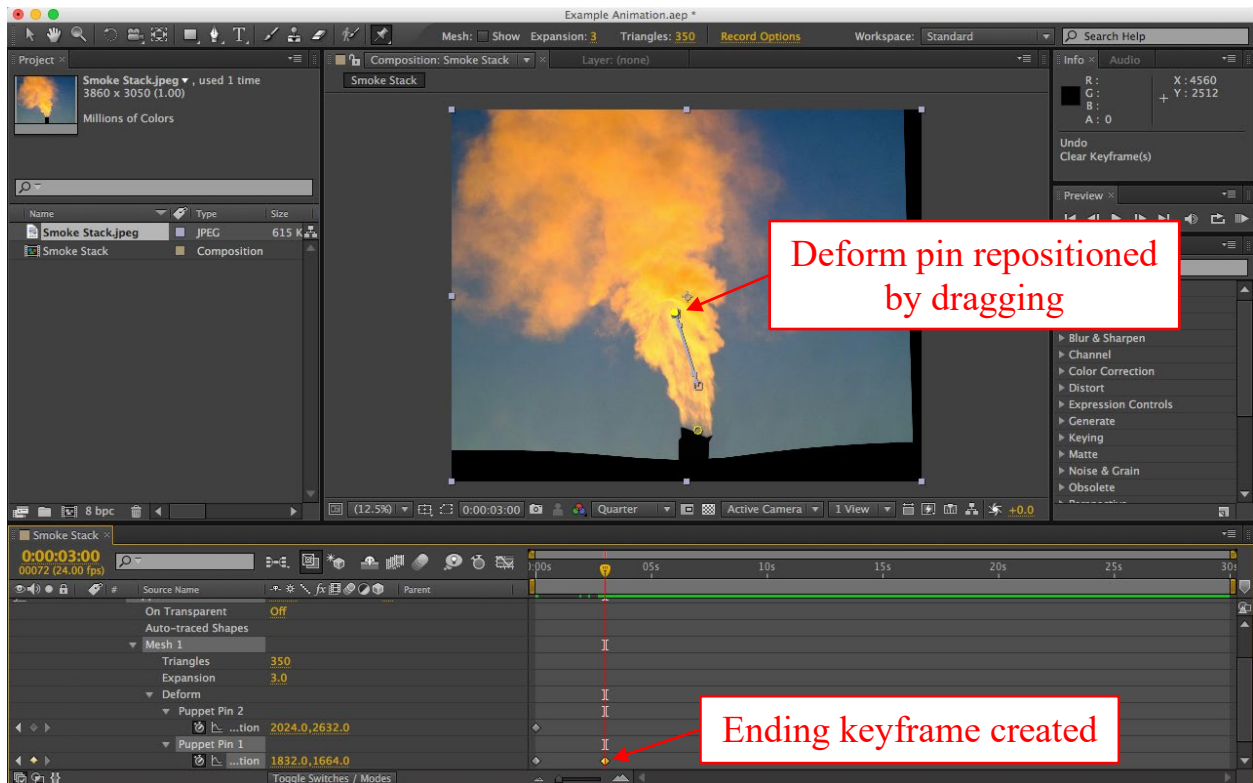
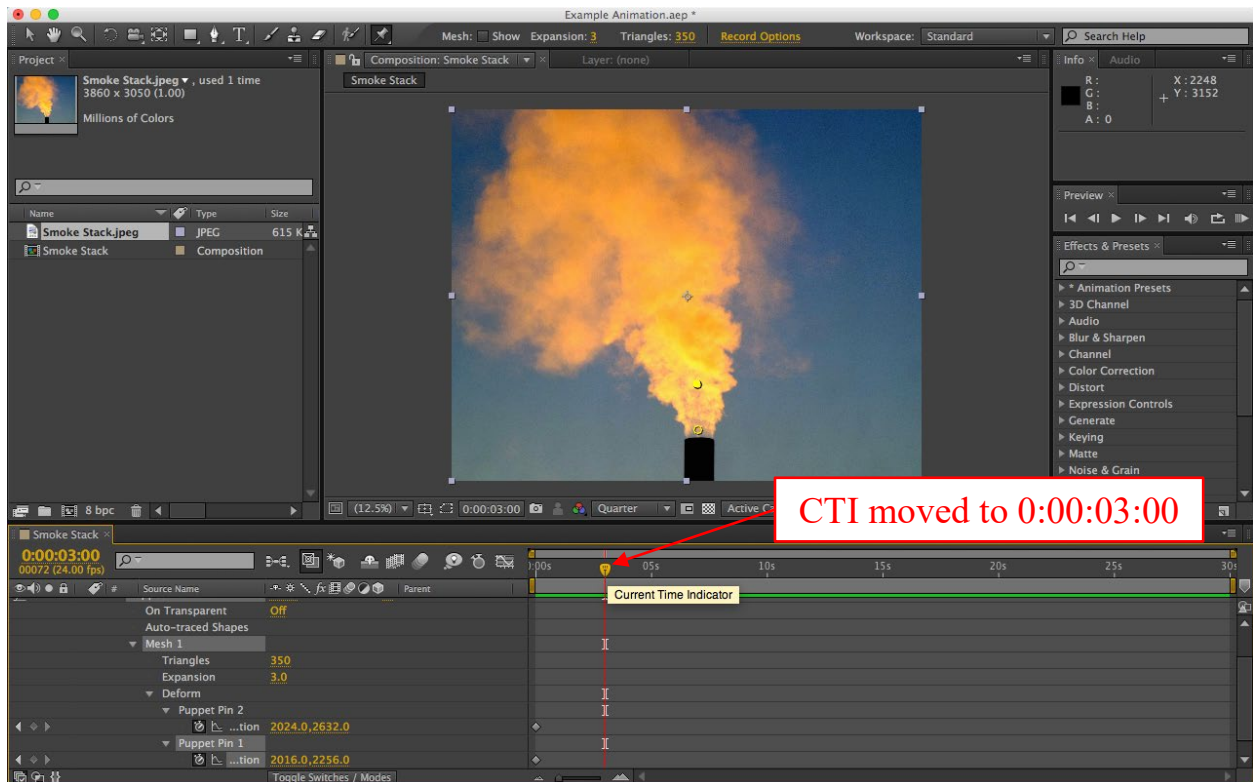


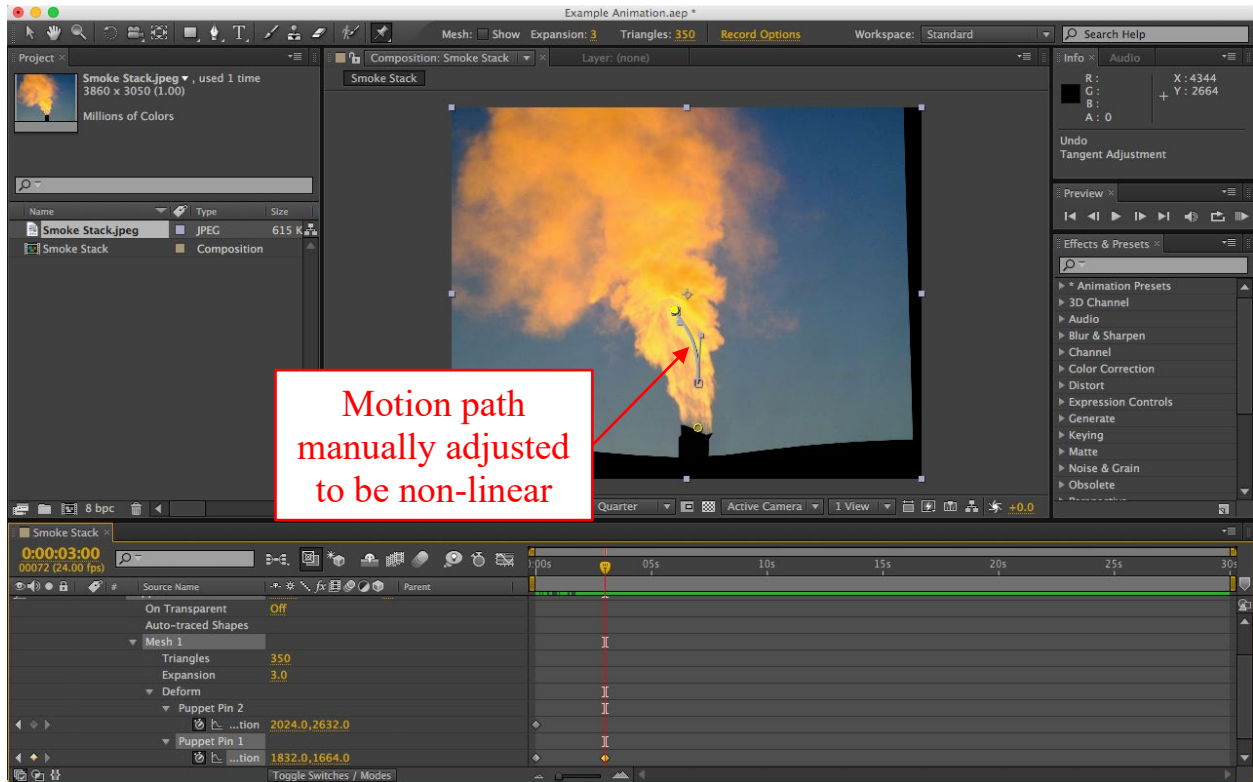
69. A second Deform pin can optionally be placed at the bottom of the smoke near the smokestack itself to keep the smokestack rigid and to prevent it from moving during the animation, as shown below. *See AEM, 218-19 (“[W]hen animating a person waving, add a pin to each foot to hold them to the ground, and add a pin to the waving hand.”).*



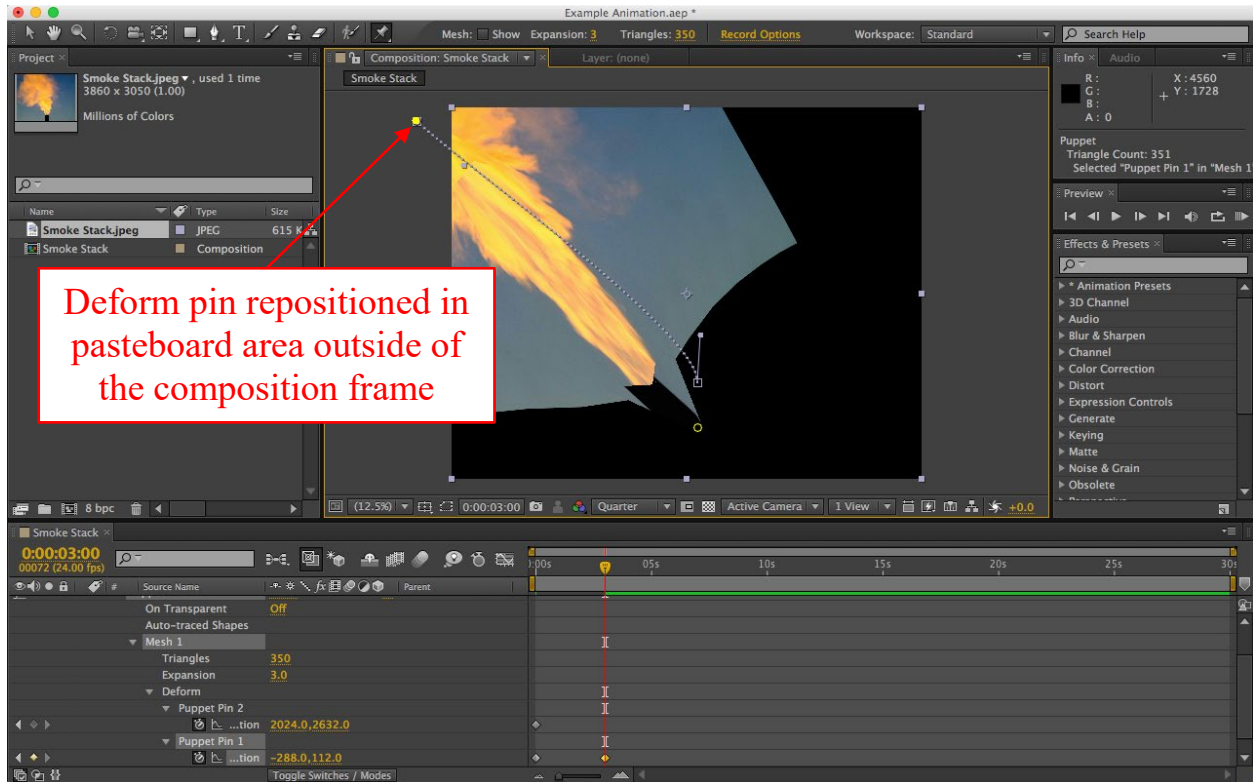
c. Creating an Ending Keyframe and a Non-Linear Motion Path

70. Next, an ending keyframe for the first Deform pin is created by using the “current-time indicator (CTI)” in the “Timeline panel” to go to another time in the animation (0:00:03:00) and then dragging the Deform pin to a new position in the “Composition panel.” *See AEM, 219, 166.* As the below screenshots indicate, the Deform pin’s repositioning creates a linear motion path for the Deform pin by default, but the shape of the motion path is made non-linear by manually adjusting the motion path accordingly. *See AEM, 194-95.*





71. Notably, as shown below, the Deform pin's position for the ending keyframe can be moved into the "pasteboard area" outside of the composition frame. See AEM, 75.



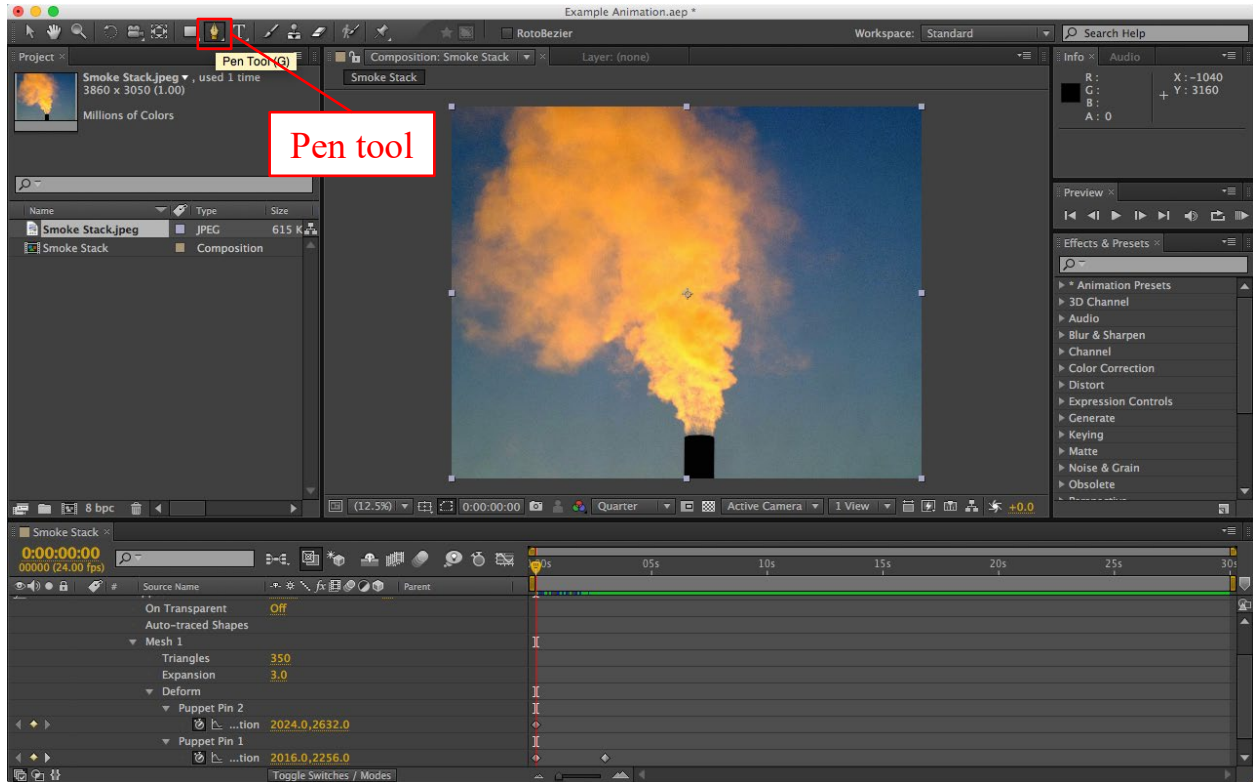
d. Using a Mask or the Roto Brush Tool to Select a Set of Pixels to be Animated

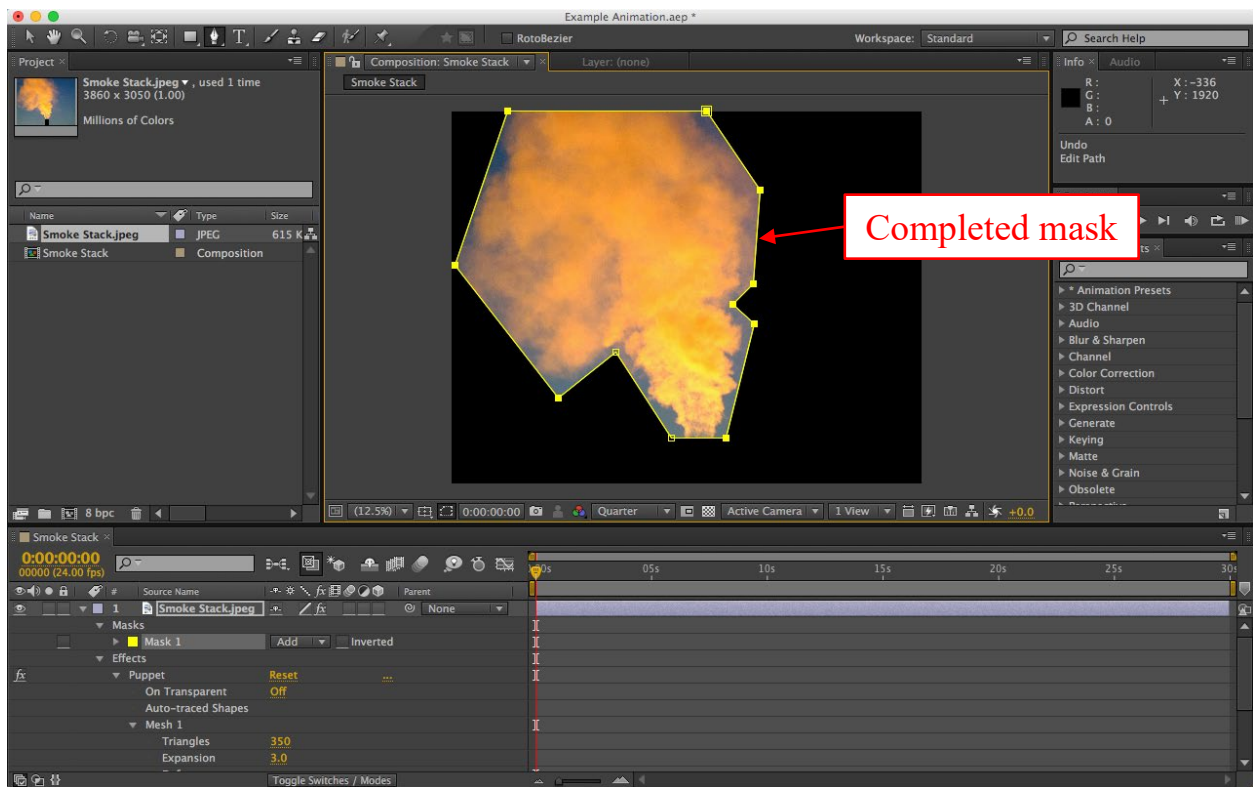
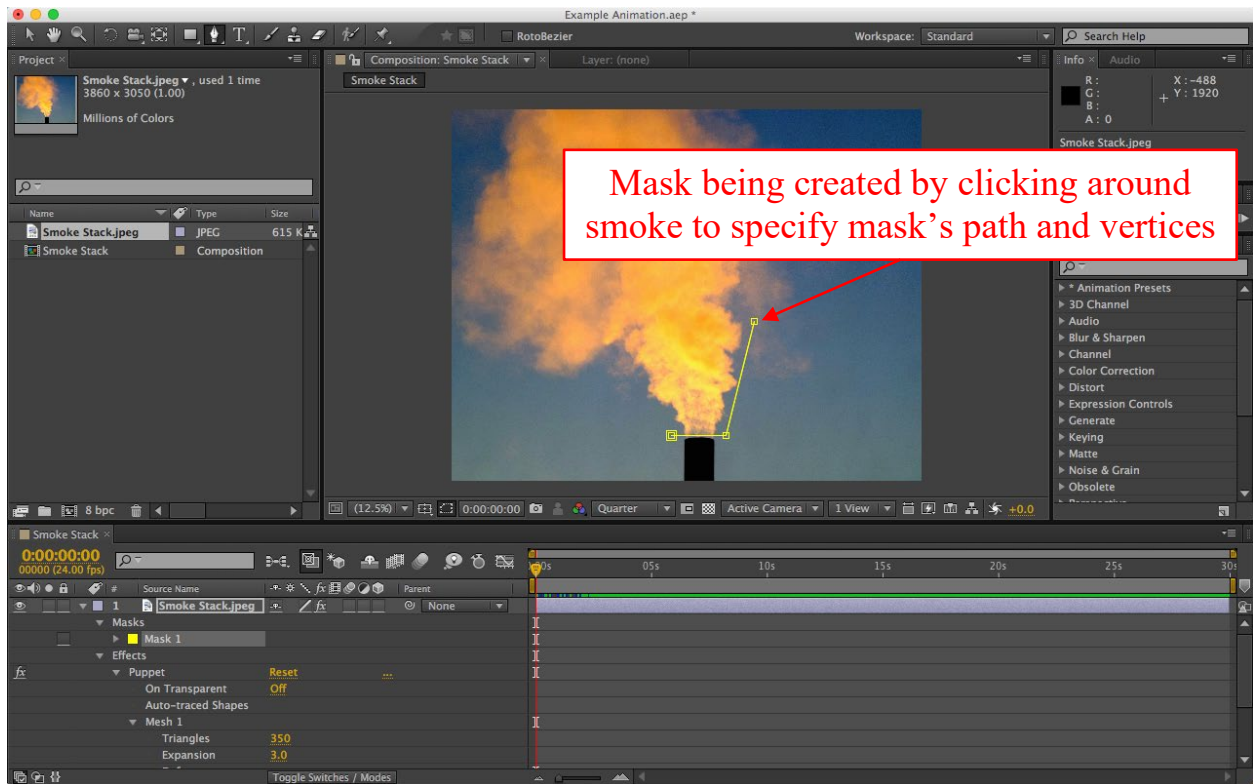
72. The following screenshots show how a mask or matte is used to select only a set of certain pixels to be nontransparent, such as those pixels of the billowing smoke in the exemplary composition, to cause only those pixels to be animated by the Puppet effect. *See AEM, 315.*

i. Creating a Mask Using the “Pen” Tool

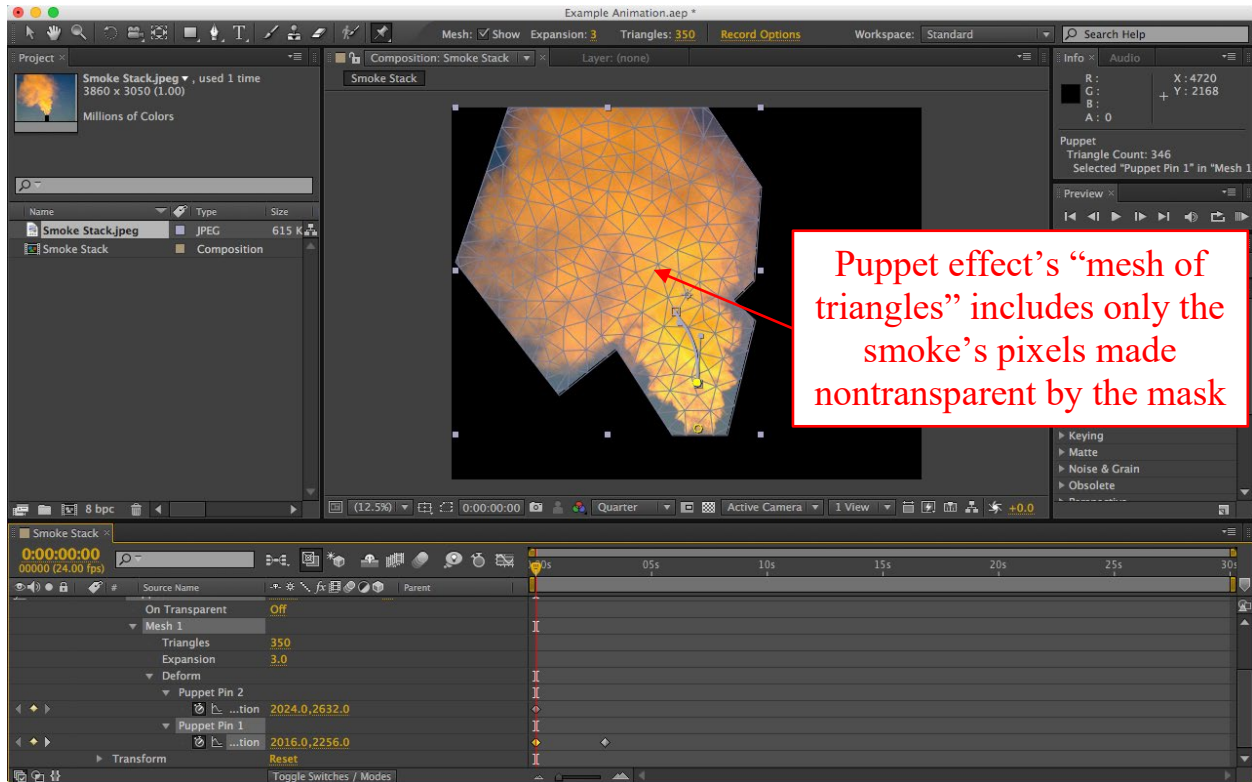
73. As shown in the three screenshots below, a mask can be created around the perimeter of the billowing smoke by using the “Pen” tool. *See AEM, 318, 264-65.* Specifically, the mask is created using the “Pen” tool by clicking on different

points around the smoke in the “Composition panel” to specify the shape of the mask correspondingly. *See AEM, 264-65.*



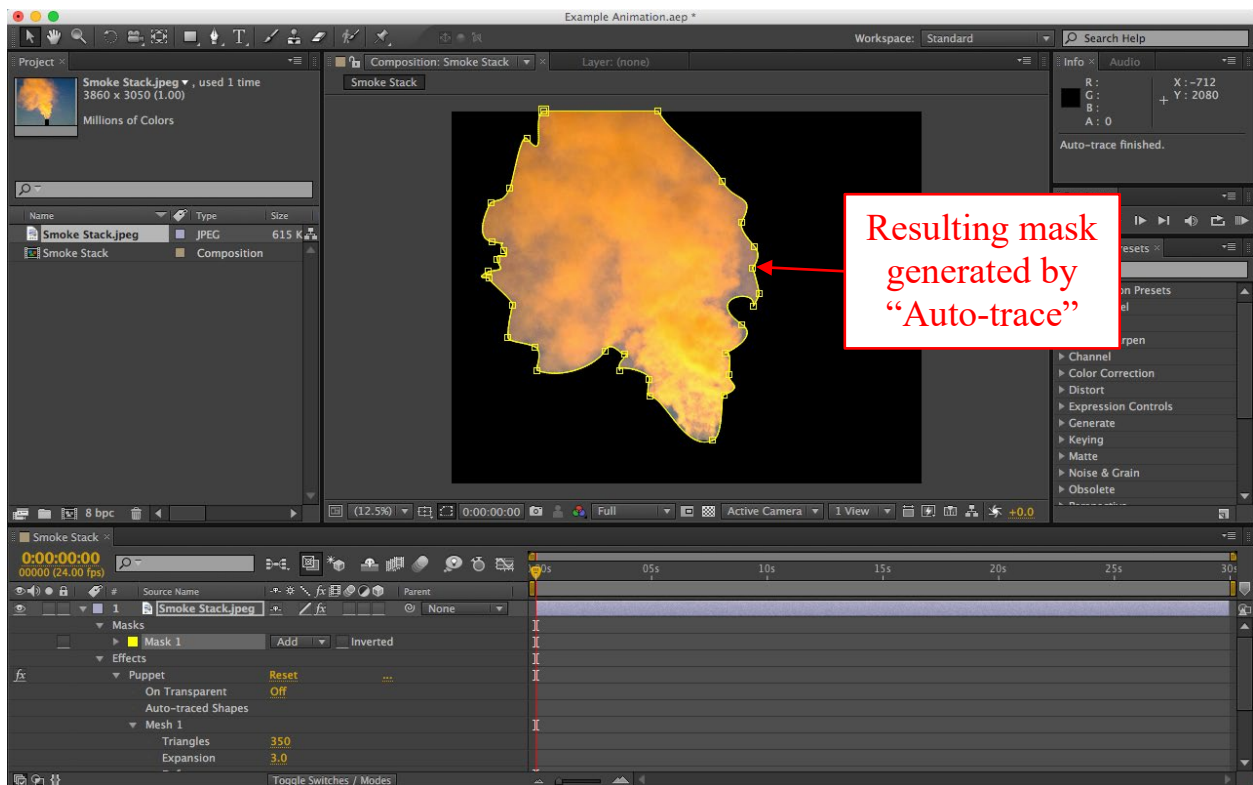
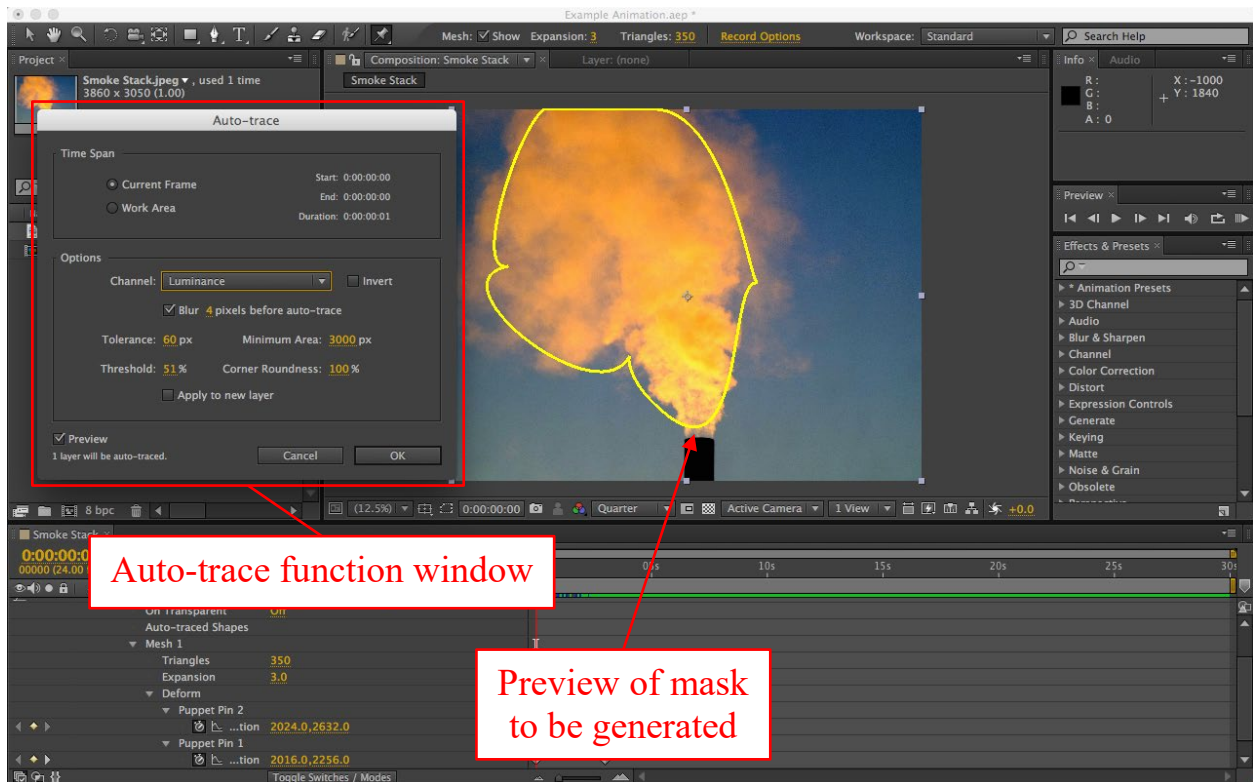


As can be seen, the mask causes only those pixels of the smoke to be nontransparent. *See* AEM, 318. And, as shown in the below screenshot, this causes only these nontransparent pixels to be included in the “mesh of triangles” to be moved during the Puppet effect animation mentioned above. *See* AEM, 219.

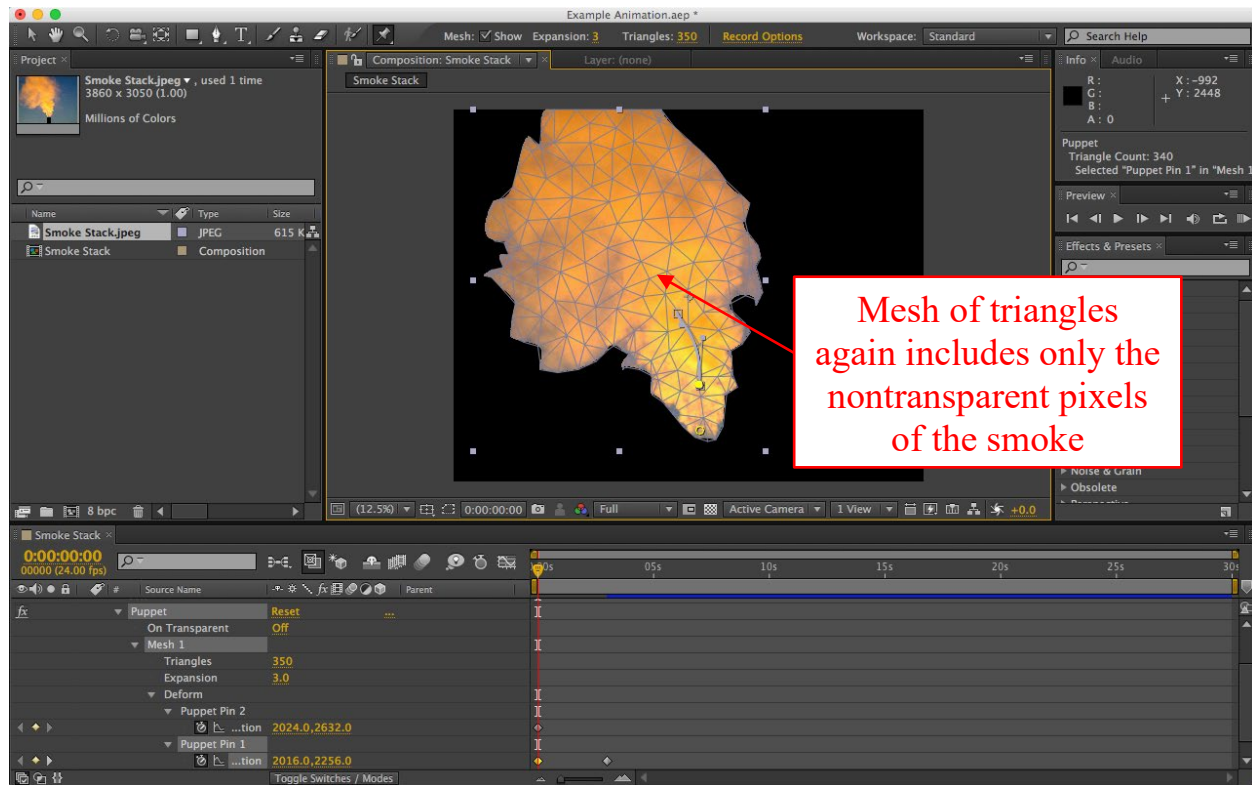


ii. Creating a Mask Using the “Auto-Trace” Function

74. Alternatively, AECS6’s “Auto-trace” function can be used to create a similar mask by “searching for edges” across the entire layer using, for example, the layer’s “luminance channel,” as shown in the below screenshots. *See* AEM, 262.

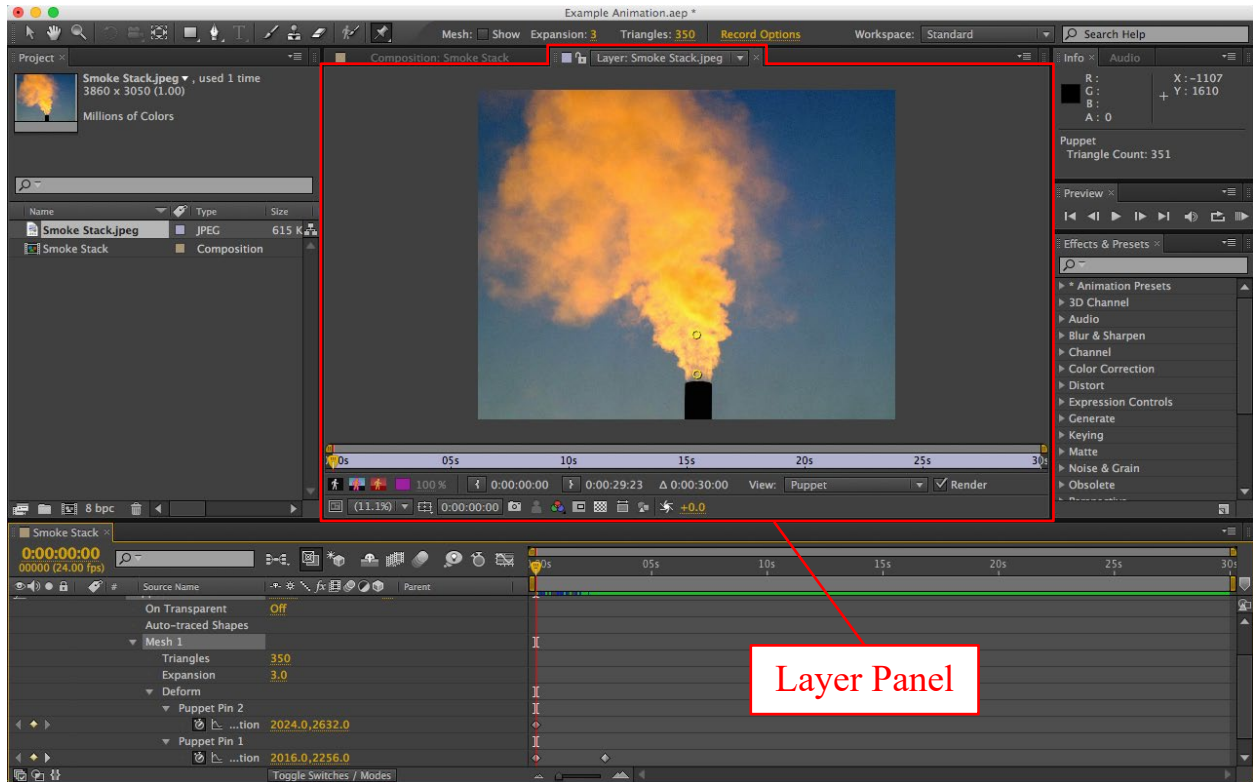


The below screenshot also shows the resulting “mesh of triangles” created from the smoke’s pixels made nontransparent by this mask. *See AEM, 218-19.*

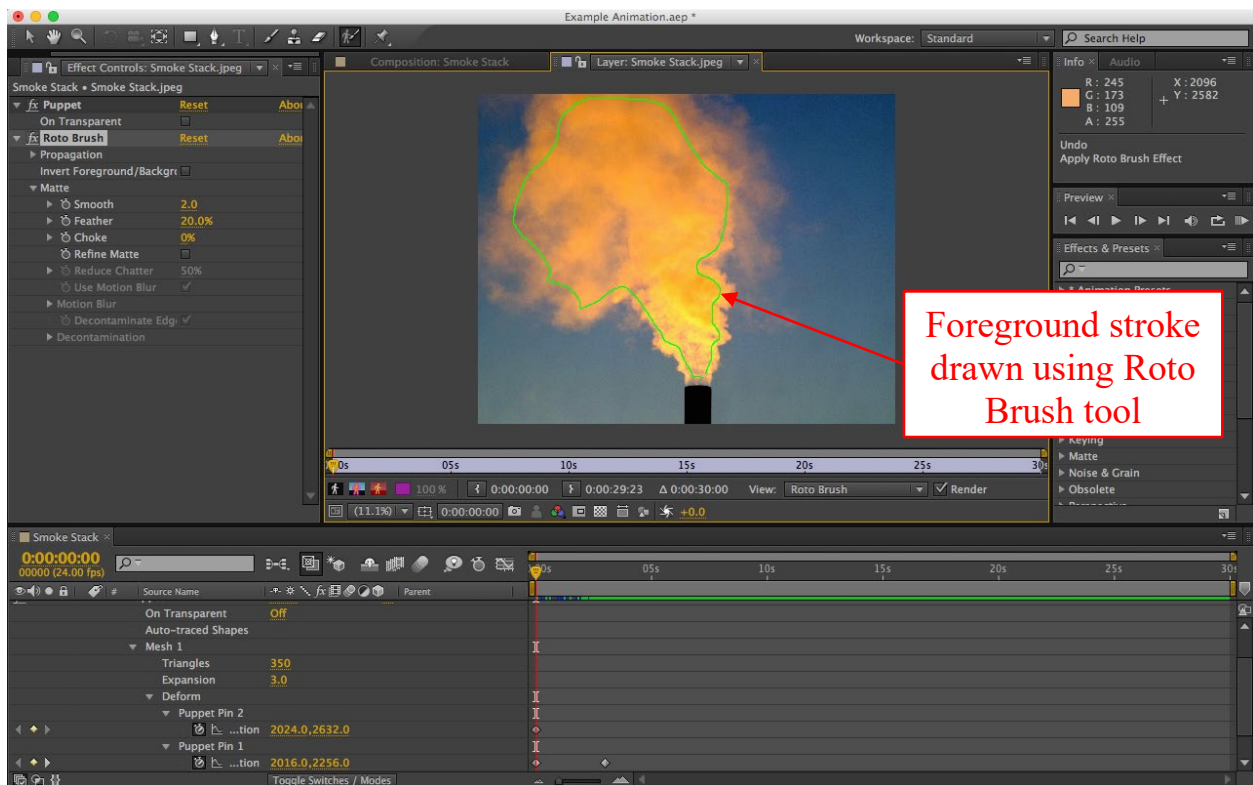
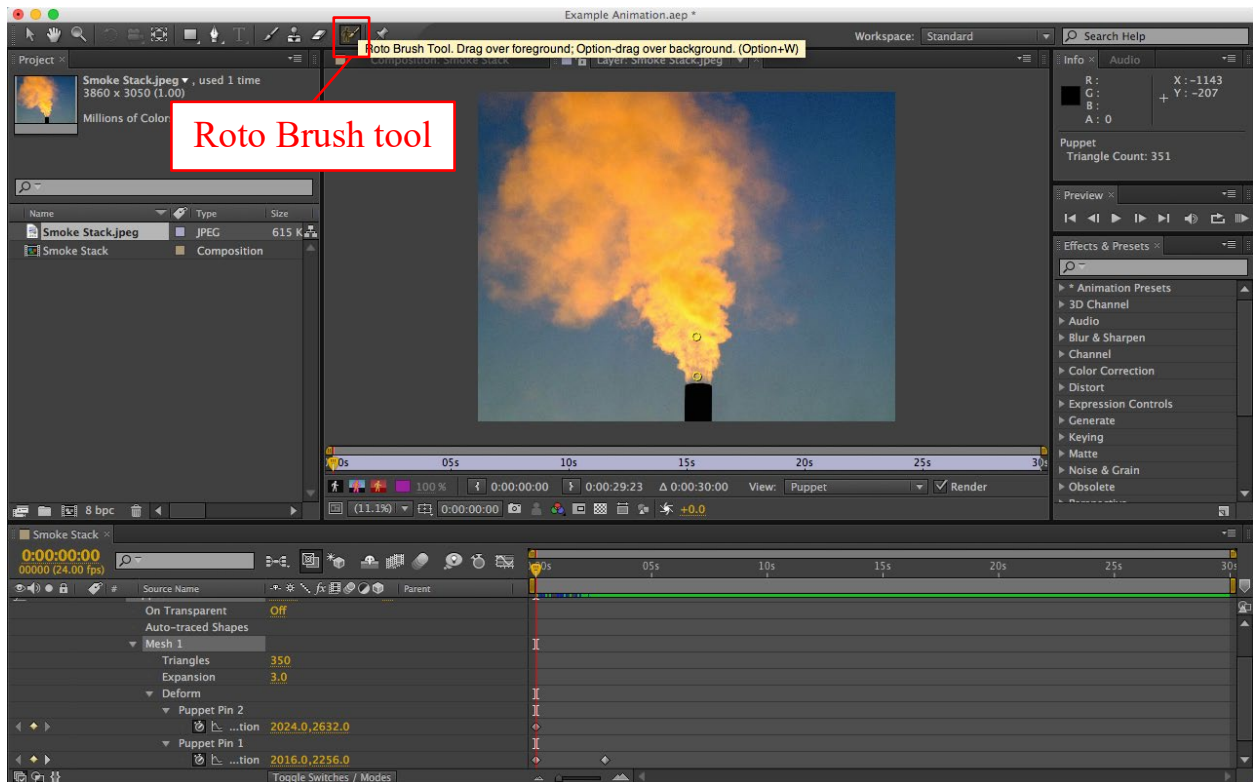


iii. Creating a Matte Using the “Roto Brush” Tool

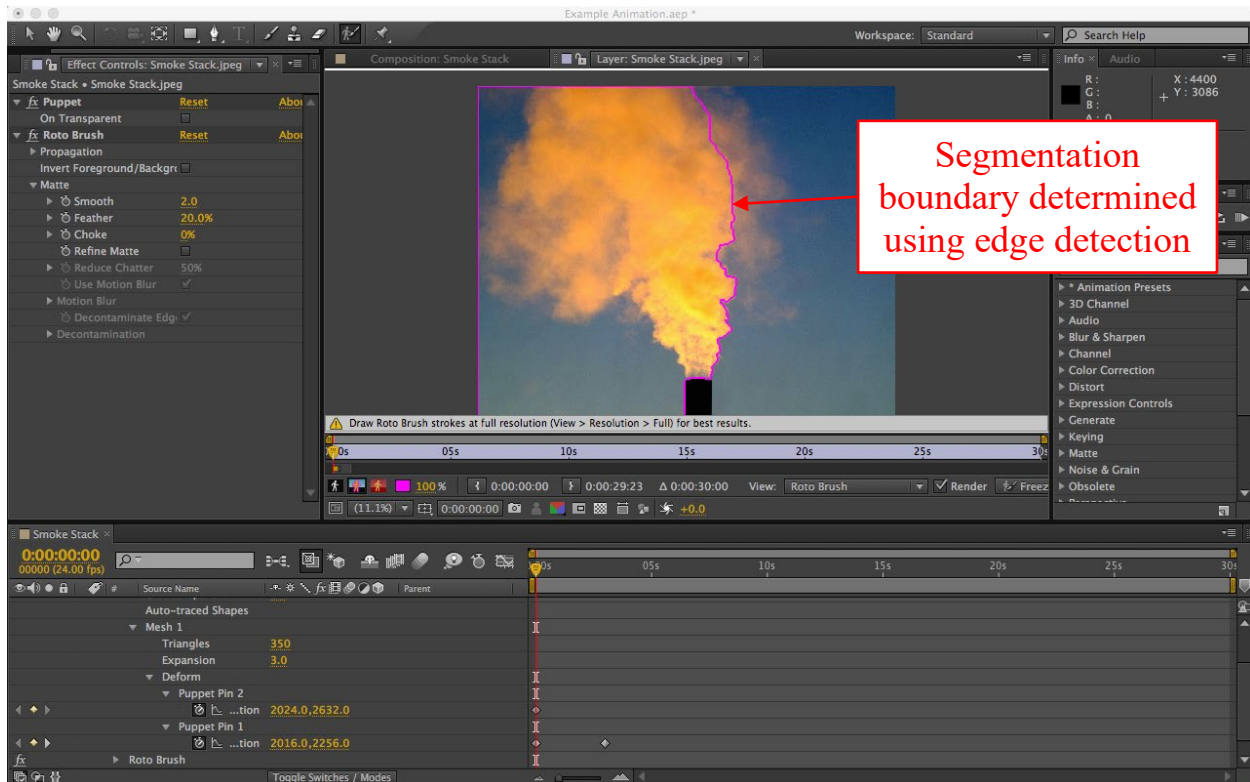
75. As an additional alternative, AECS6’s “Roto Brush” tool can be used to create a matte, which, like a mask, defines which areas of the layer are nontransparent (in this case, the foreground) or transparent (in this case, the background). *See AEM, 328.* To begin using the “Roto Brush” tool, the layer must first be opened in the “Layer panel,” as shown below. *See AEM, 328.*



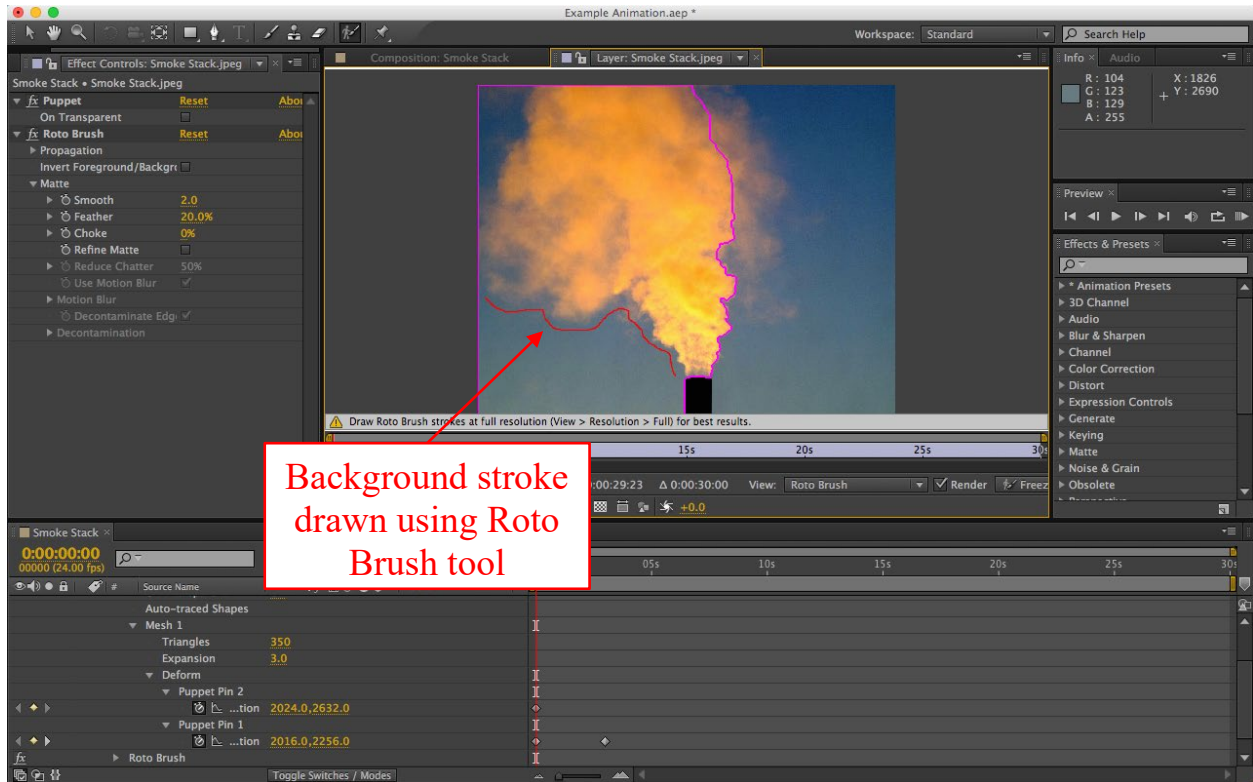
76. Thereafter, as shown in the screenshots below, the “Roto Brush” tool is used to draw a foreground “stroke” (shown in green) that defines the area of the layer to be considered the foreground, which in this exemplary composition is the billowing smoke. *See AEM, 328.*



Based on this foreground stroke, AECS6 uses “Edge Detection” to then determine a “segmentation boundary” (shown in pink) that separates the foreground smoke and the background sky, as shown below. *See AEM, 328-31.*

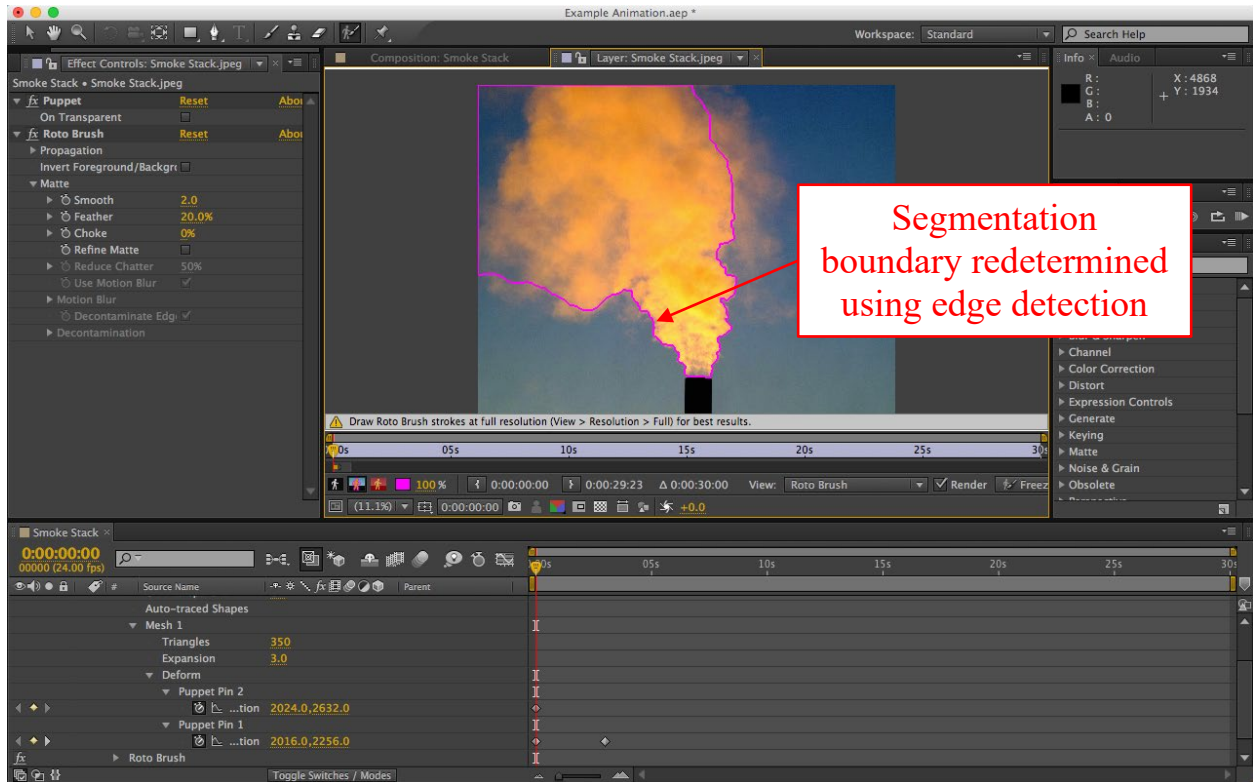


77. Although the segmentation boundary is not perfect, it is refined as shown in the screenshot below by using the “Roto Brush” tool to draw a background stroke (shown in red) that defines the area of the layer to be considered the background, which in this exemplary composition is the sky. *See AEM, 328-29.*

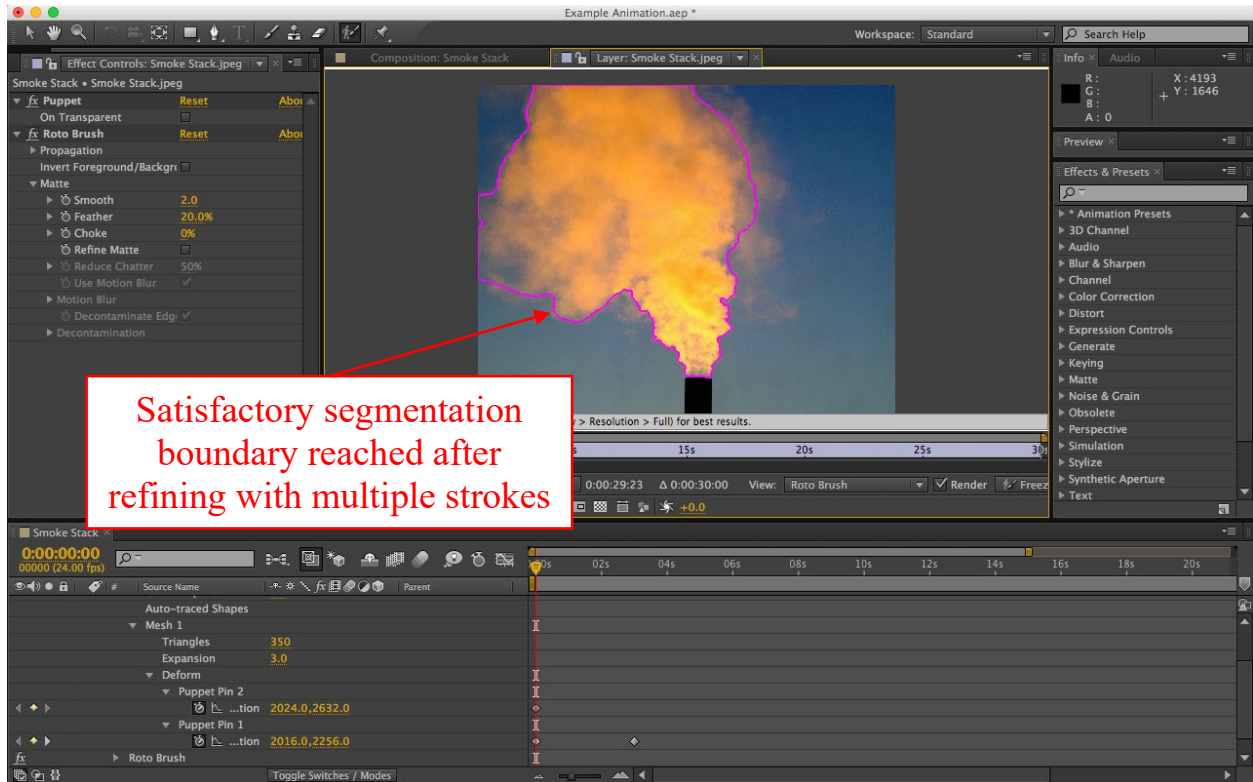


Based on this background stroke and the previous foreground stroke, AECS6 again uses edge detection to then redetermine the segmentation boundary, as shown below.

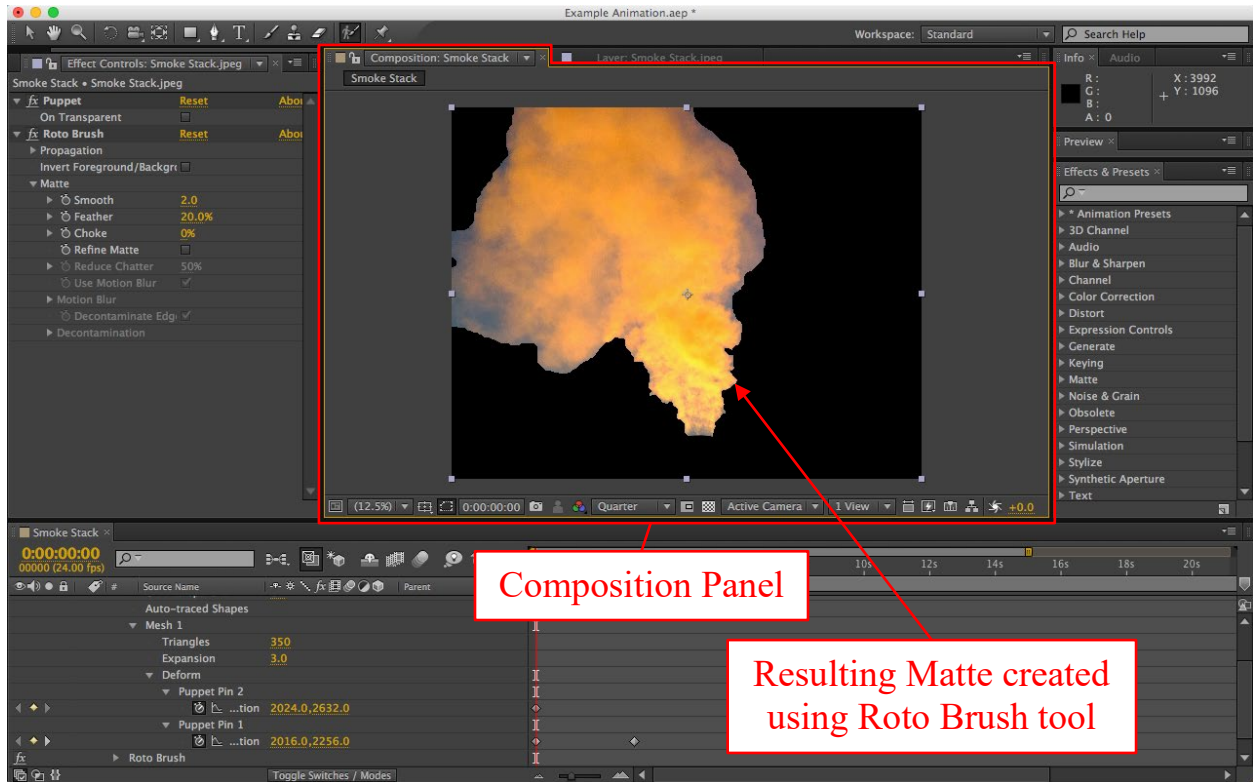
See AEM, 328-31.



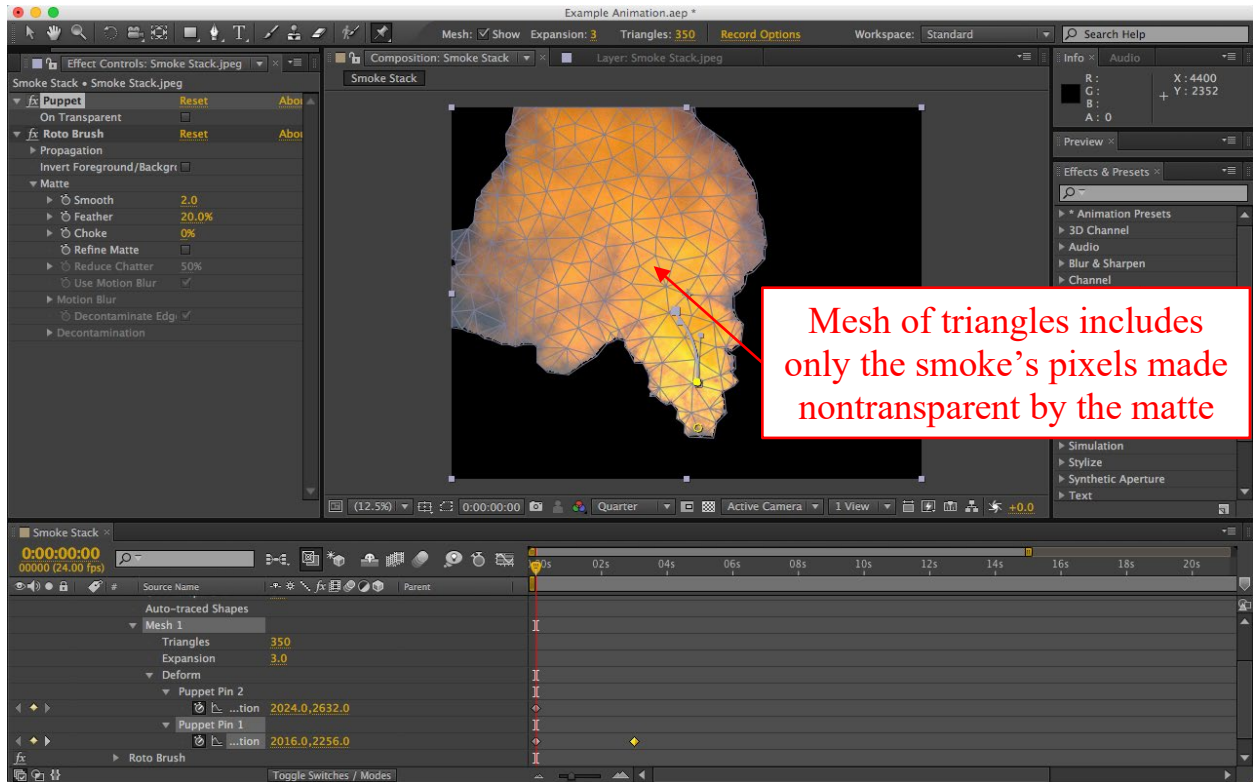
78. As can be seen, the segmentation boundary is again not perfect but is now closer than before to the true boundary between the foreground smoke and background sky. Additional foreground and background strokes are drawn to continue the refining process until a satisfactory segmentation boundary is reached, such as shown below. See AEM, 328-29.



79. Once a satisfactory segmentation boundary is reached, the resulting matte can be viewed in the “Composition Panel,” as shown below. *See AEM, 317.*

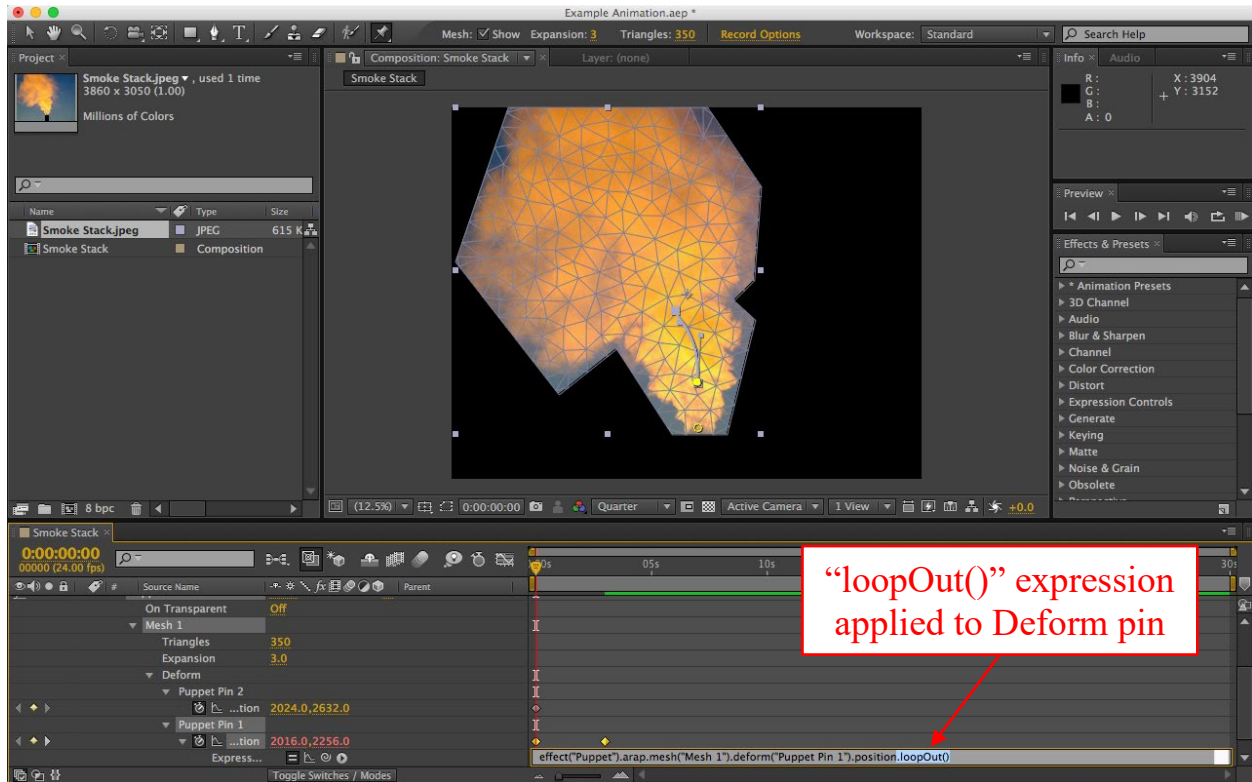


The below screenshot shows the resulting “mesh of triangles” created from the pixels specified by the matte to be part of the foreground, i.e., the pixels of the billowing smoke made nontransparent by the matte. *See* AEM, 219.



e. Looping the Animation

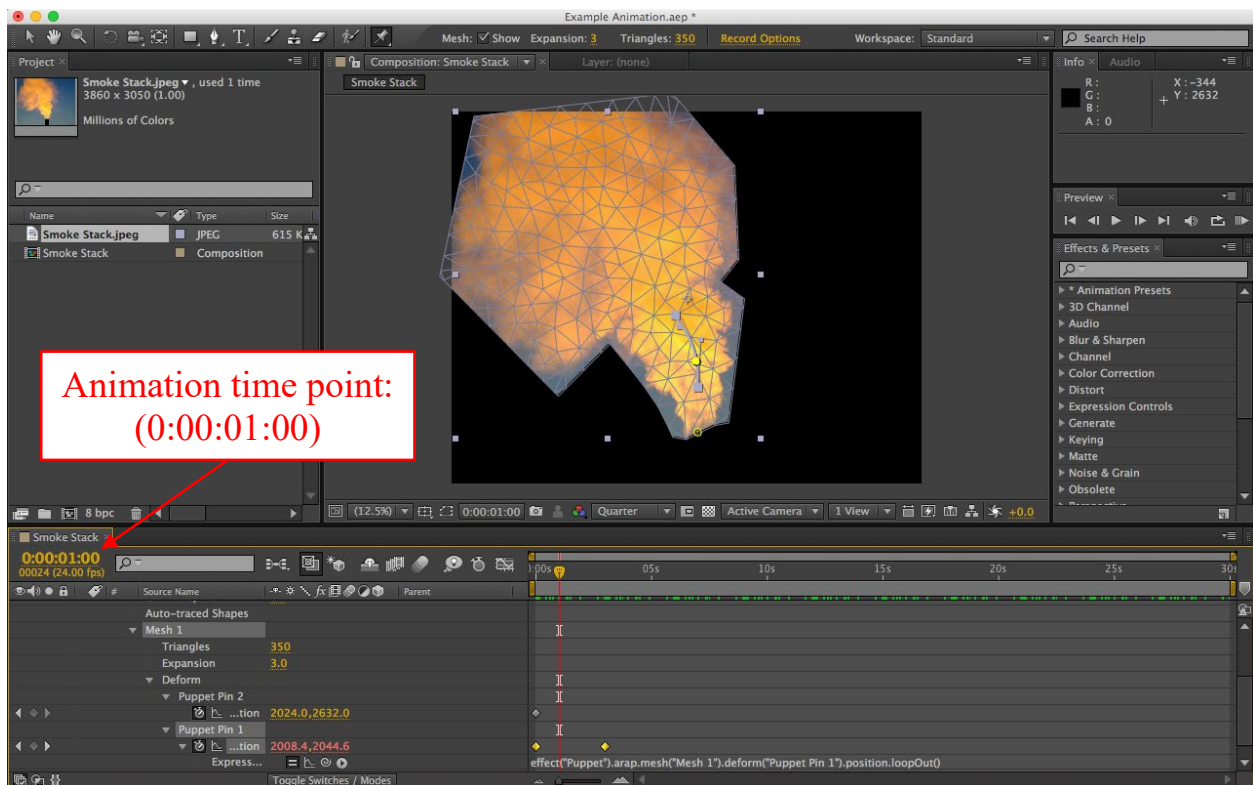
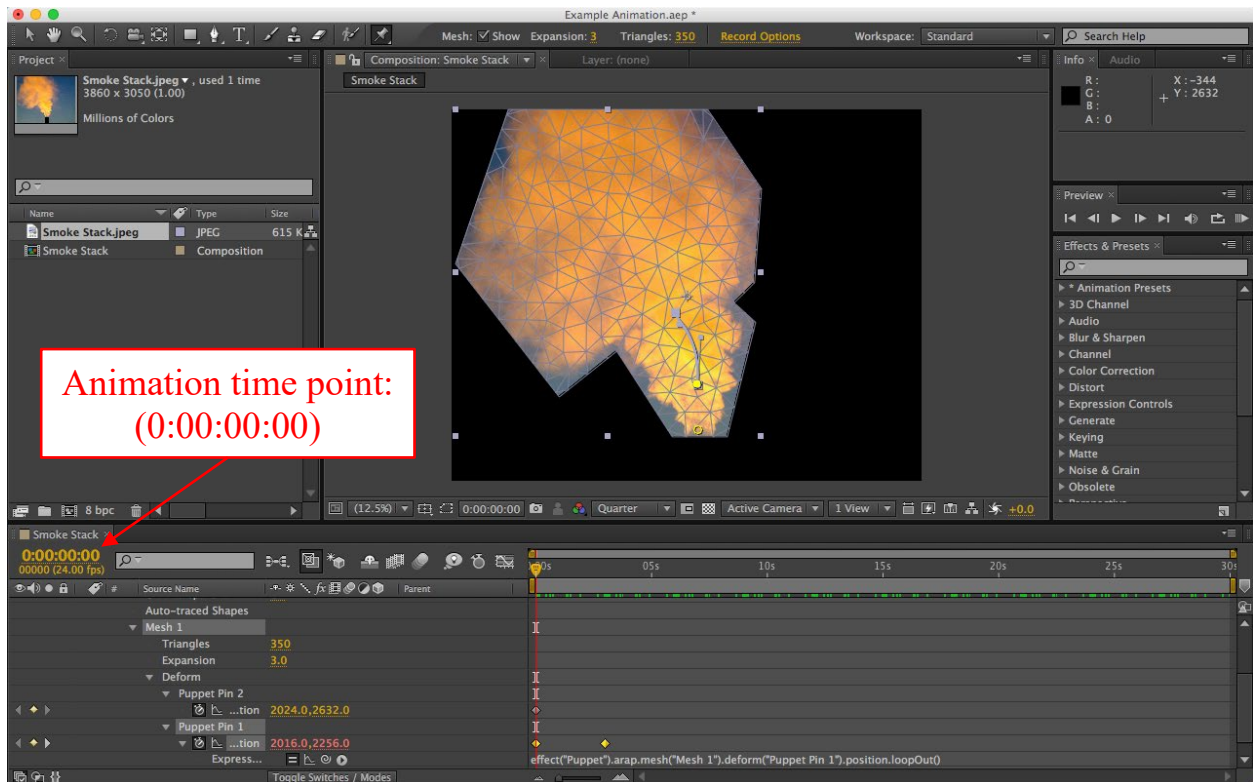
80. After creating the mesh using one of the above tools and functions, a “loopOut()” expression can be applied to the Deform pin to cause the resulting animation to repeatedly loop. See AEM, 562, 219. The screenshot below shows how a “loopOut()” expression is applied to the Deform pin in the exemplary composition depicted in Section VIII.B.2.d.i’s final screenshot:

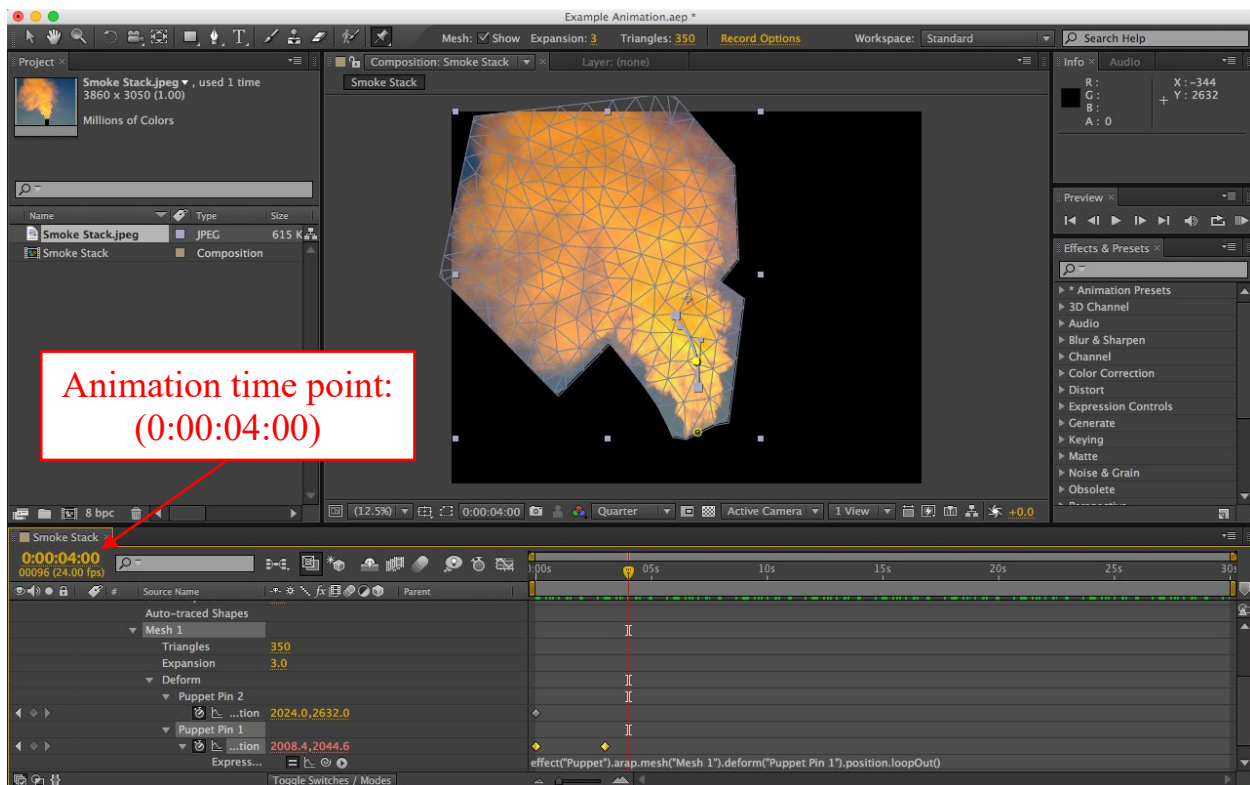
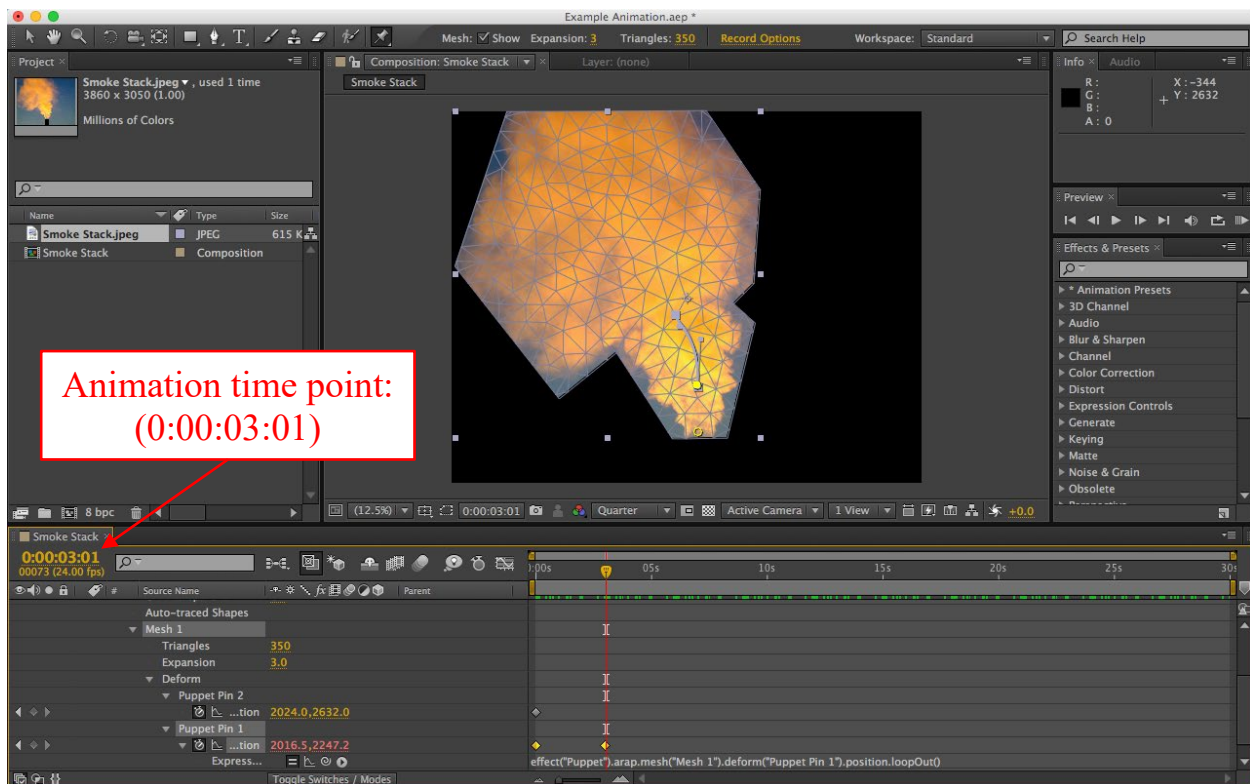


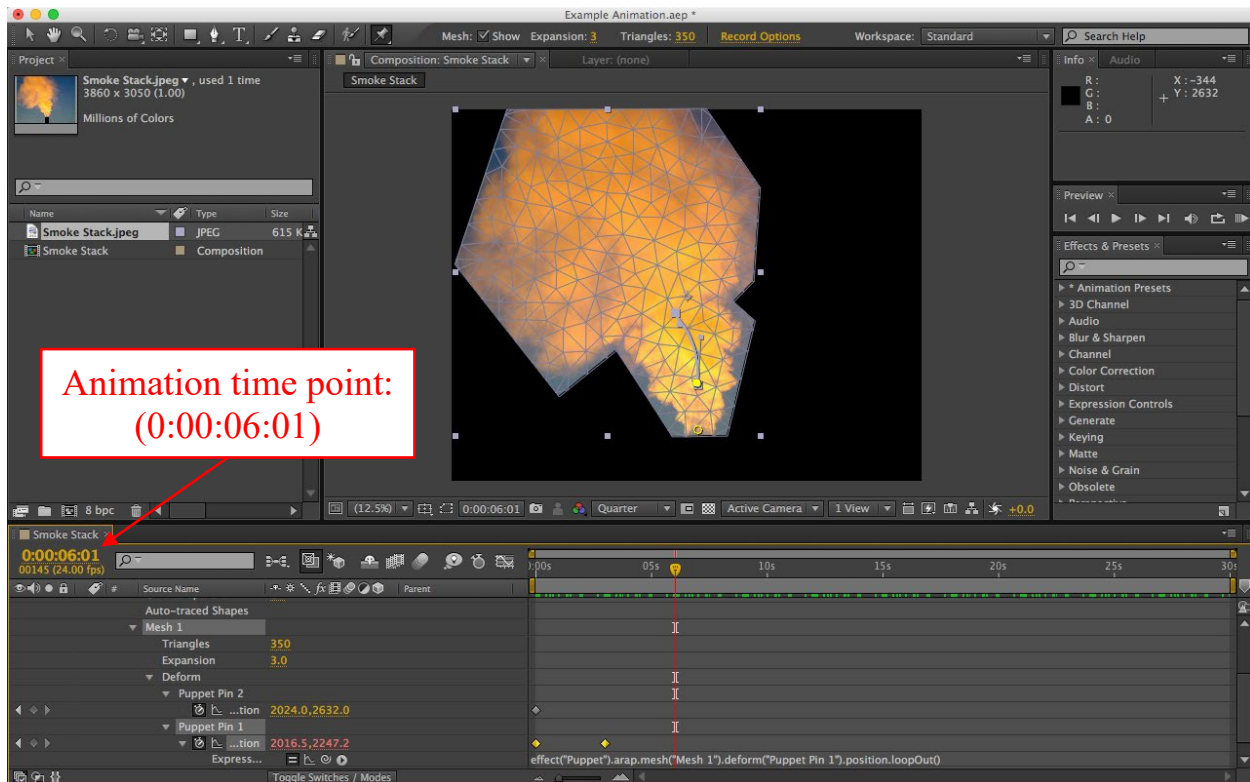
While not shown here, the same can be done to the Deform pins in the exemplary compositions depicted in the final screenshots of Sections VIII.B.2.d.ii and VIII.B.2.d.iii to achieve the same result of a repeatedly looping animation.

f. Final Resulting Animation

81. The final resulting animation of the exemplary composition in Section VIII.B.2.e is shown in the below screenshots. From top to bottom, the screenshots show the animation at the following time points, respectively, in order to mimic the animation playing in time: (0:00:00:00), (0:00:01:00), (0:00:02:00), (0:00:02:23), (0:00:03:01), (0:00:04:00), (0:00:05:00), (0:00:05:23), and (0:00:06:01). I have also included in Attachment B cropped versions of the below screenshots depicting only the “Composition panel” in order to provide a better view of the resulting animation.



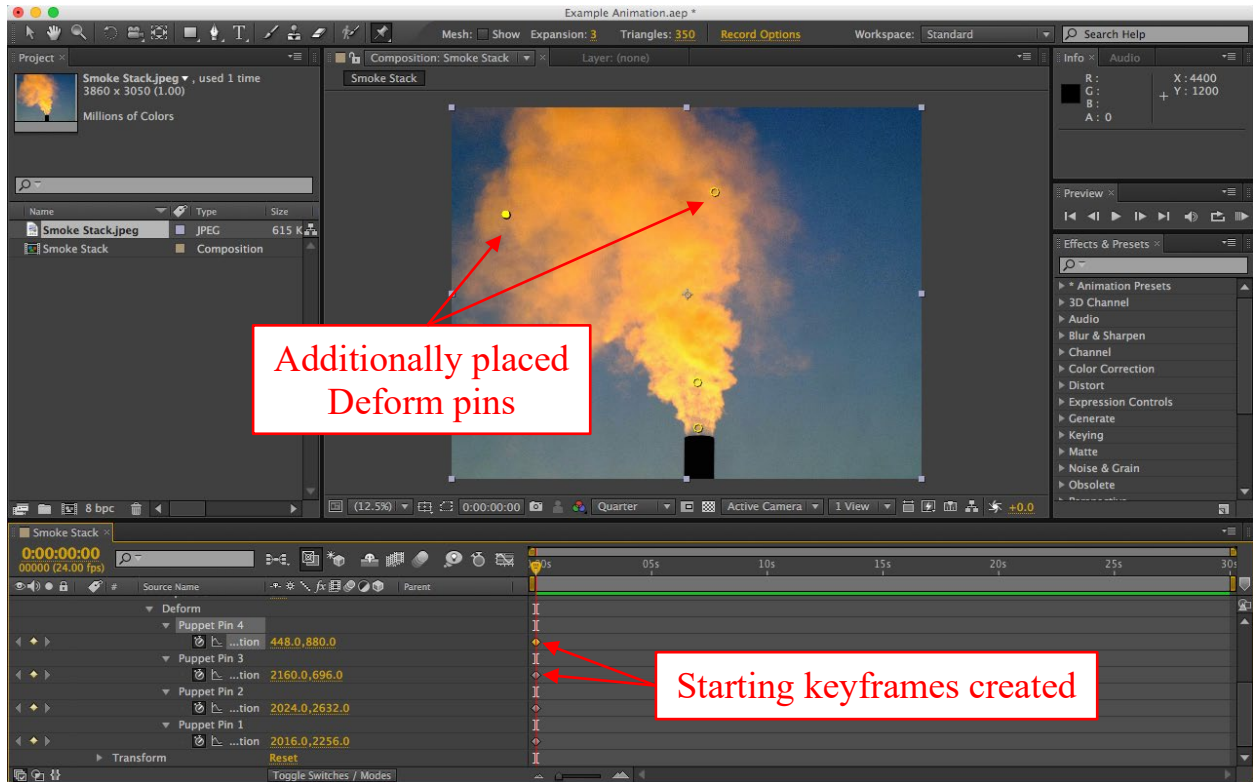




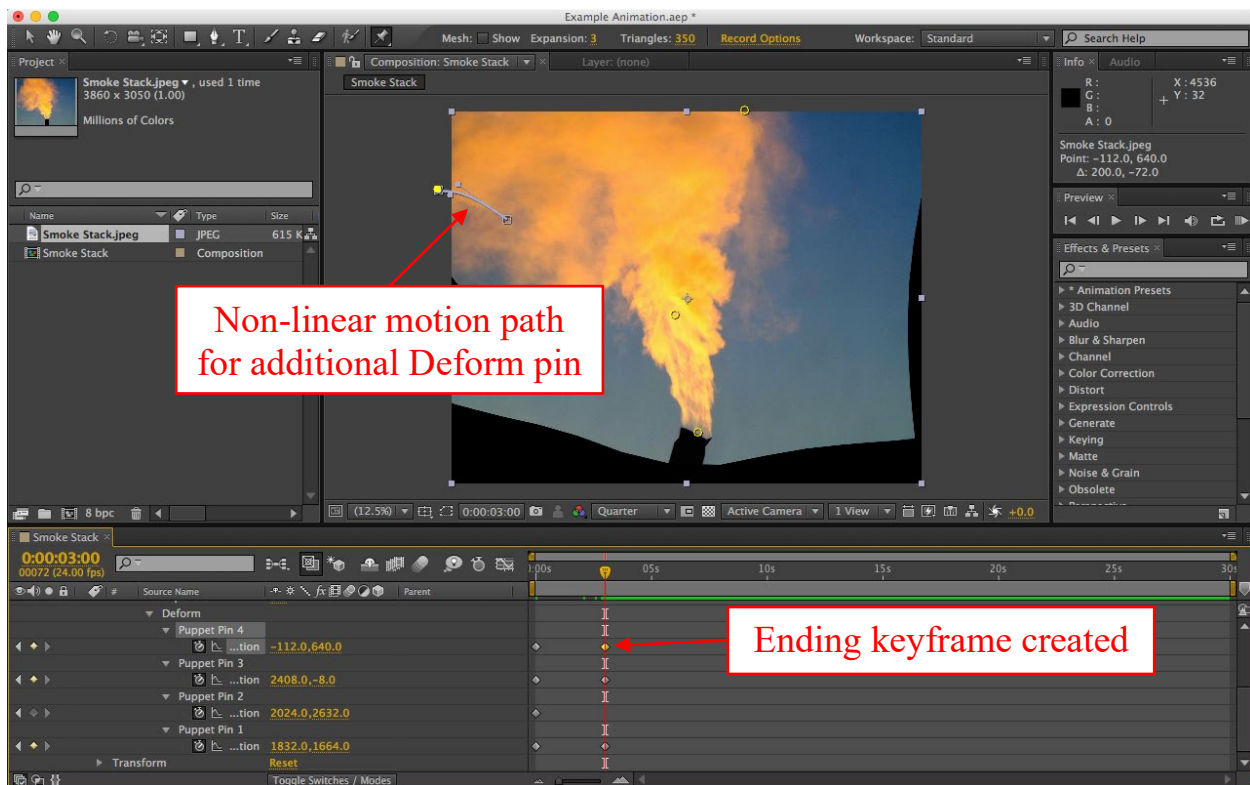
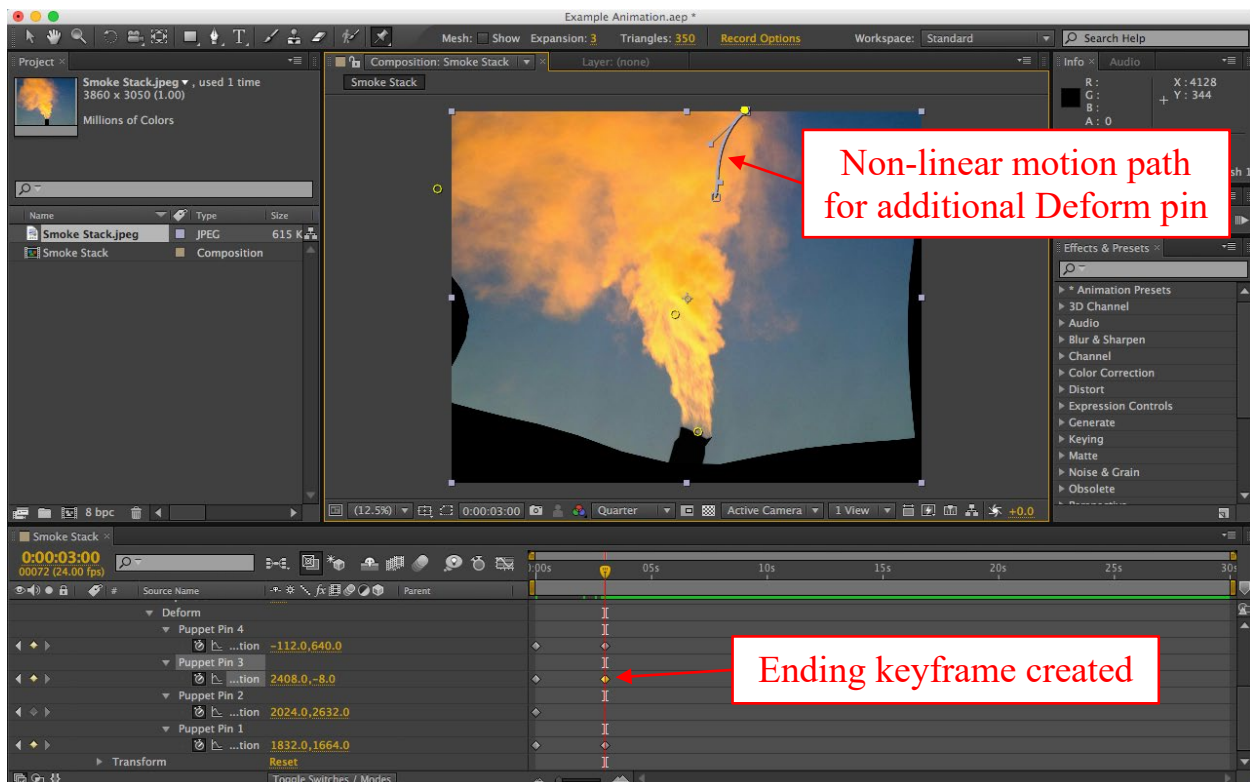
g. Animating with Multiple Deform Pins

82. Although in the above exemplary composition only one Deform pin is specified to move during the animation, additional Deform pins can also be applied and moved in the same way, as shown in the below screenshots. *See* AEM, 218-19.

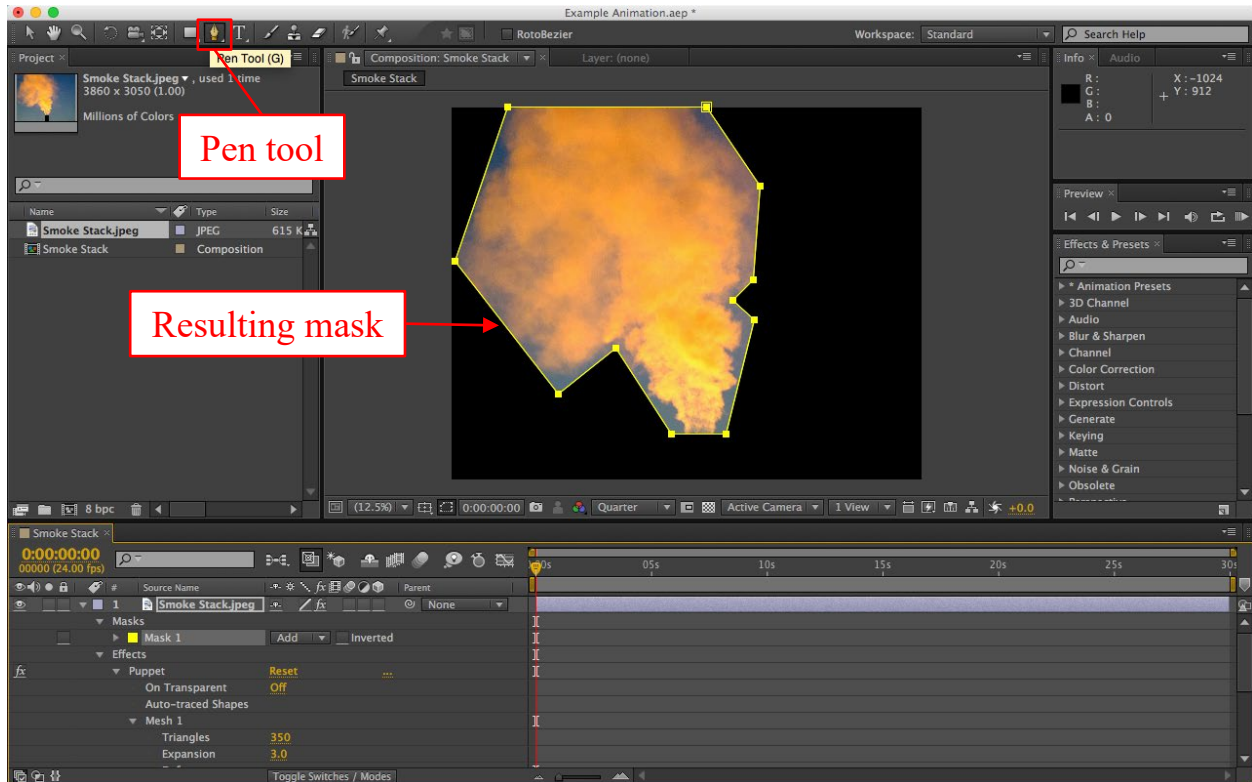
83. First, in addition to placing the first Deform pin as discussed in Section VIII.B.2.b, multiple other Deform pins are placed on other portions of the layer, and starting keyframes will be automatically created accordingly, as shown below. *See* AEM, 218-19.



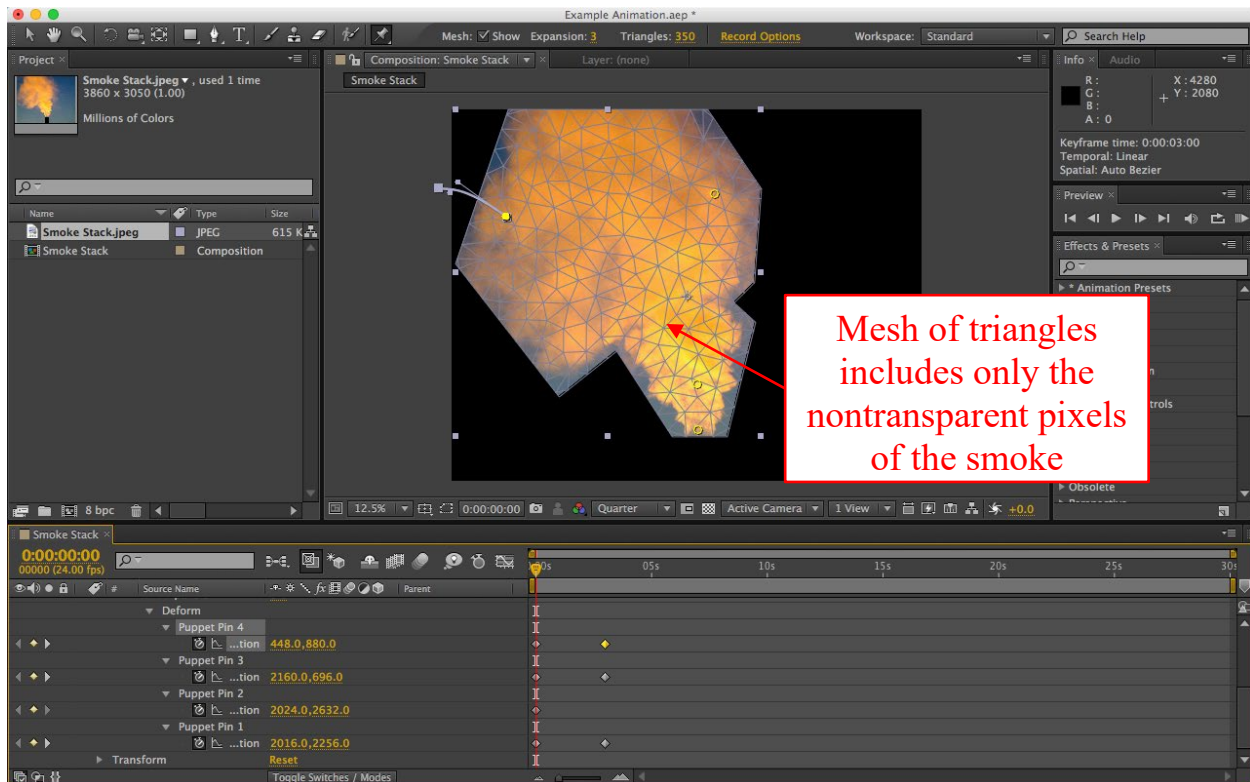
84. Next, ending keyframes and non-linear motion paths for each of the additional Deform pins are created in the same way as discussed in Section VIII.B.2.c, as shown below. *See AEM, 219, 194-95.*



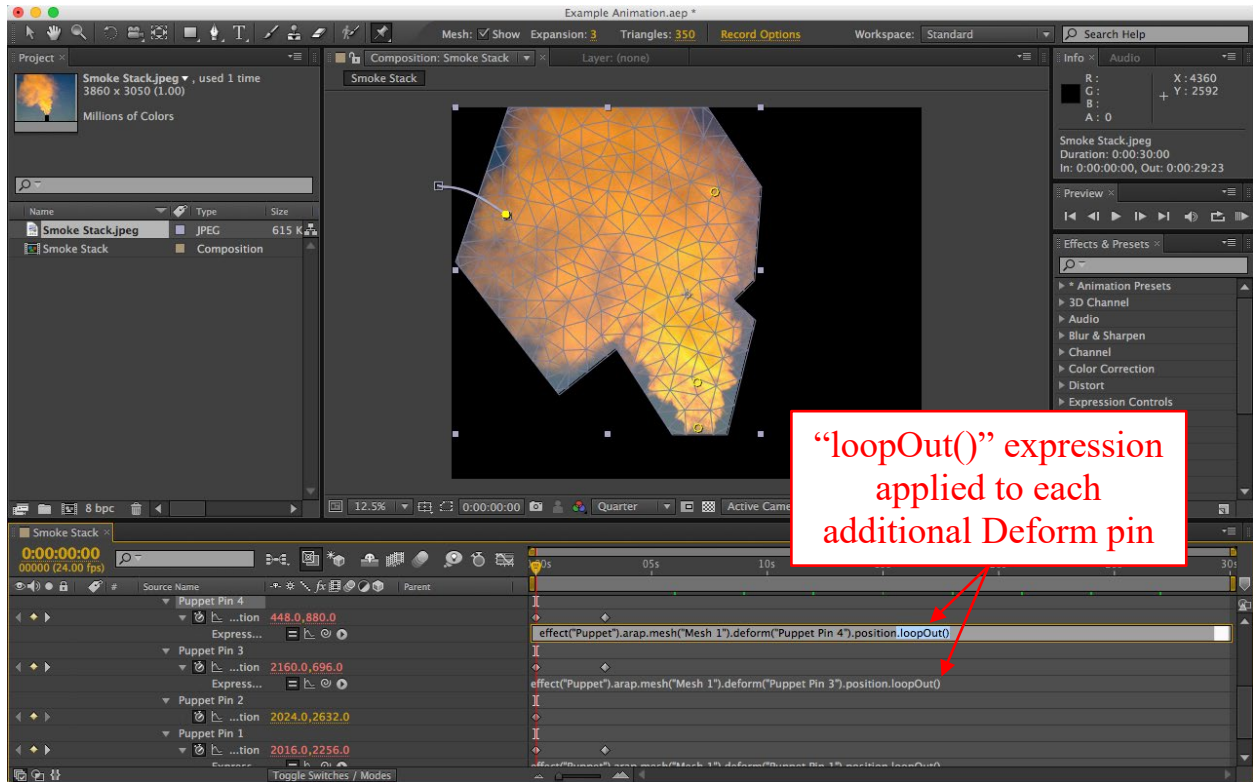
85. The mask from Section VIII.B.2.d.i is then likewise created and applied to the layer in the same way discussed therein to cause only the pixels of the billowing smoke to be nontransparent and thus animated, as shown in the screenshot below. *See AEM, 318, 264-65.*



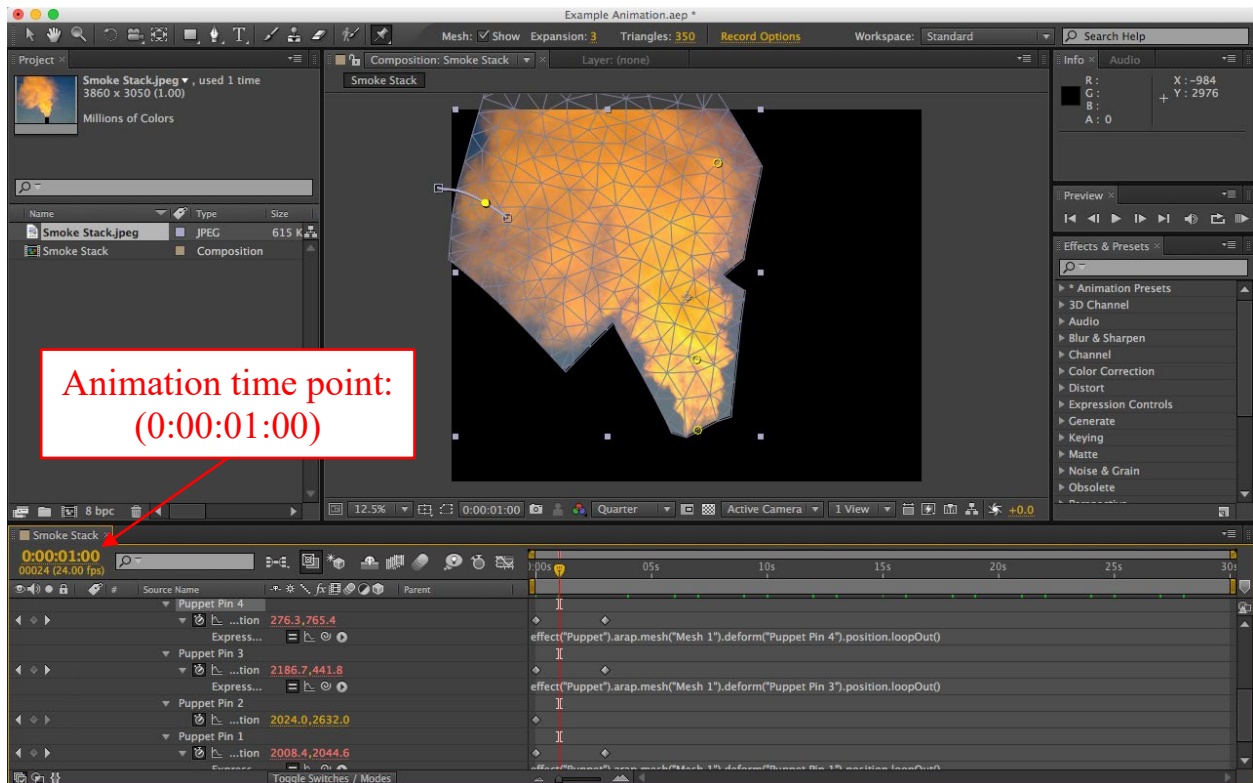
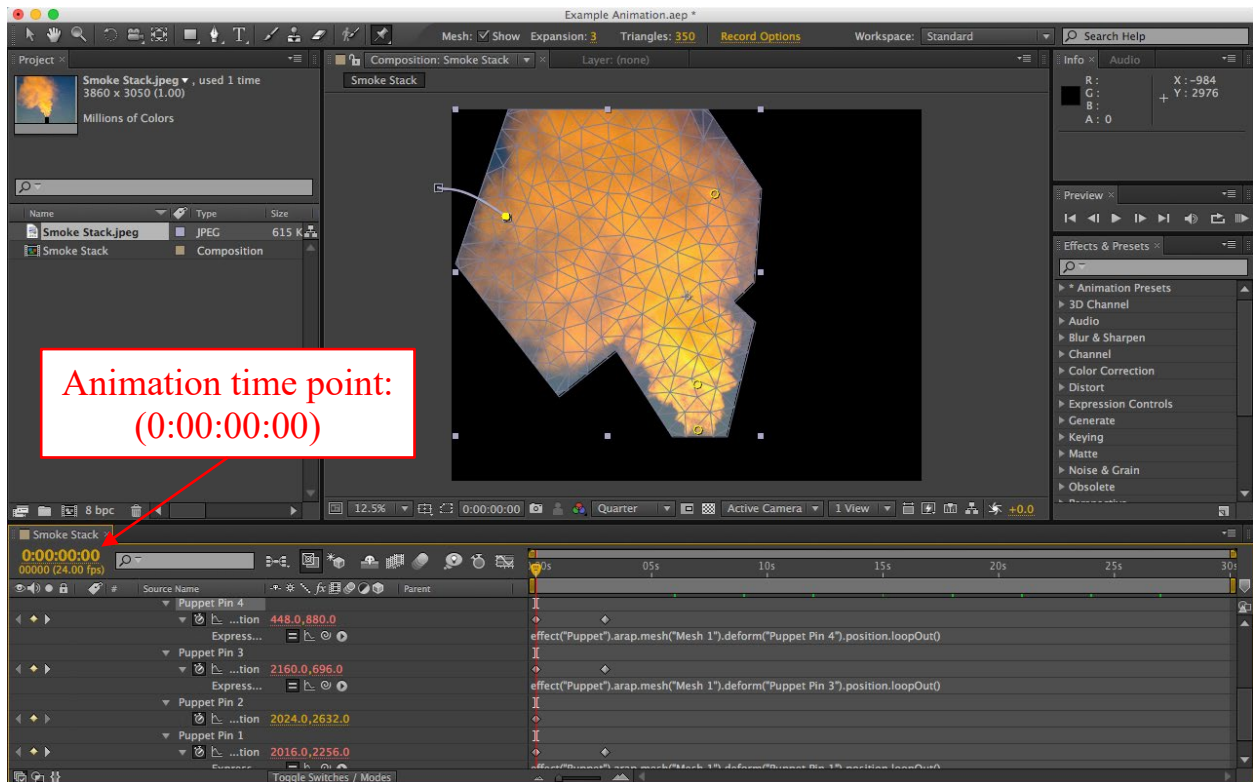
The below screenshot shows the “mesh of triangles” created from the smoke’s pixels made nontransparent by this mask. *See AEM, 219.*

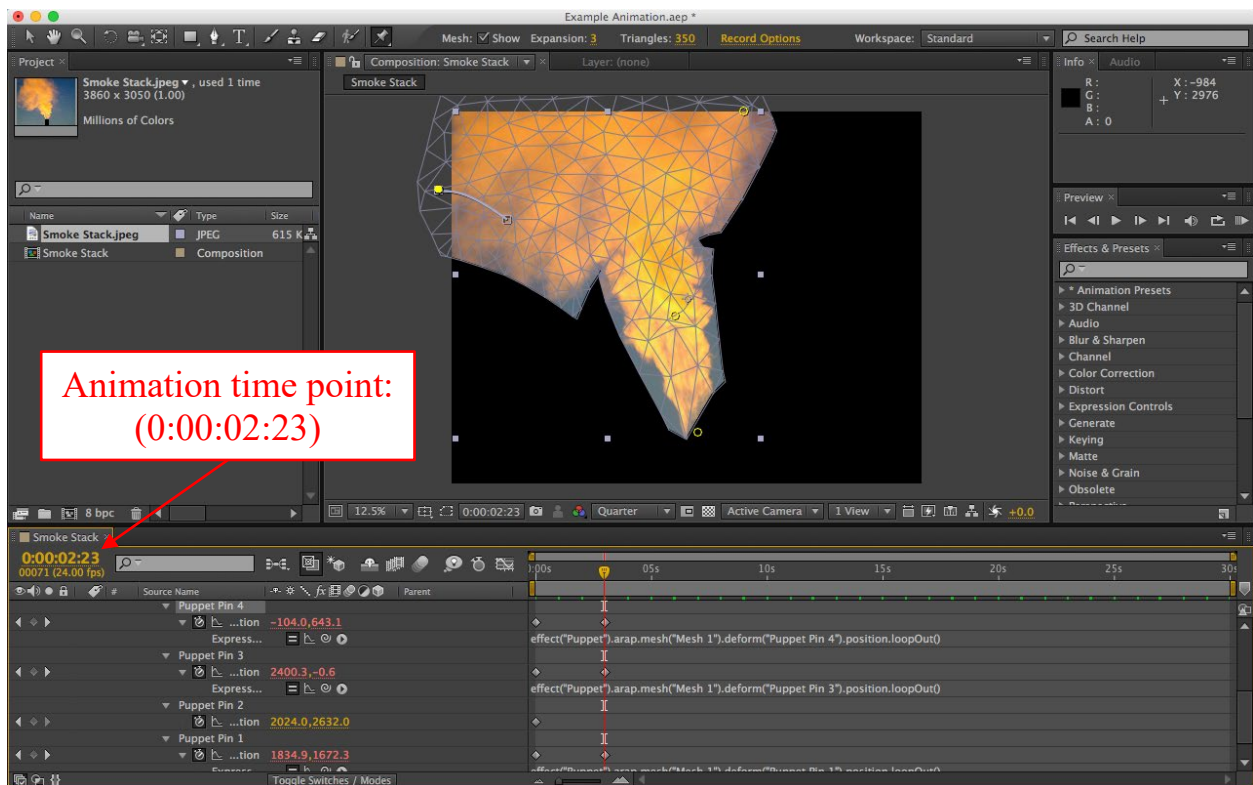
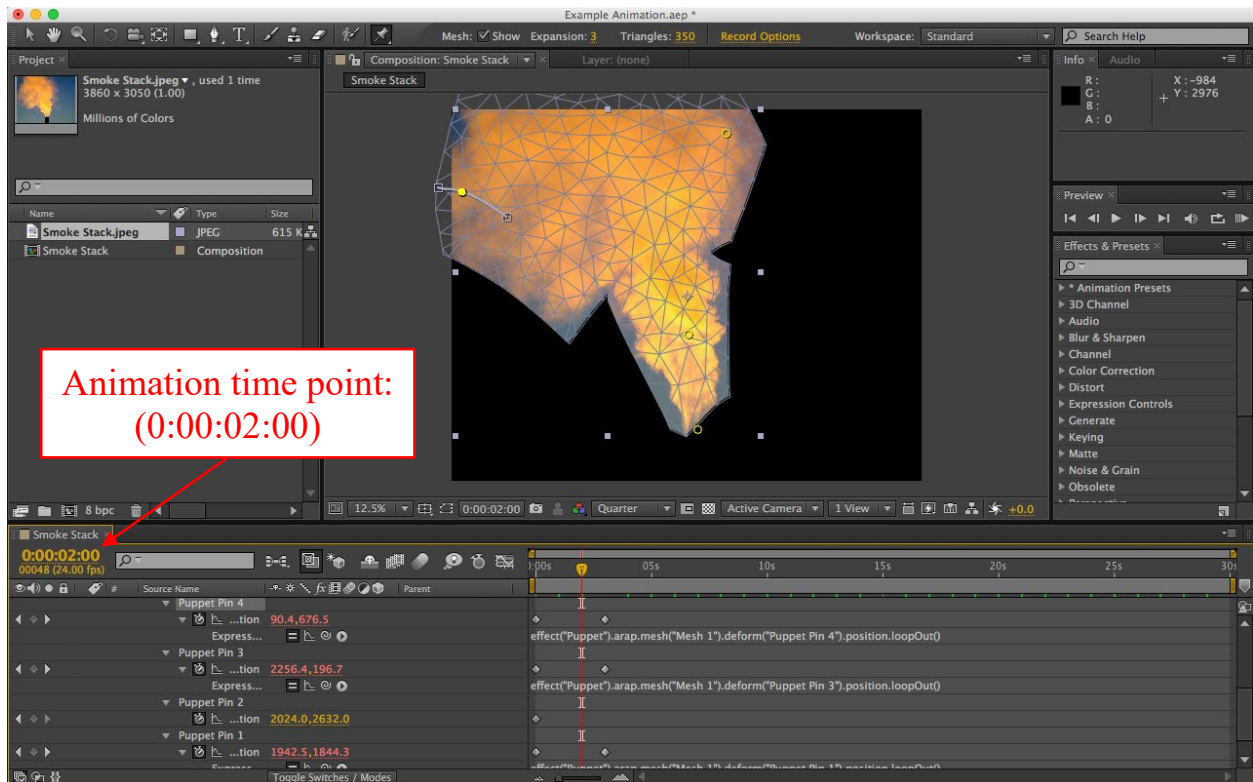


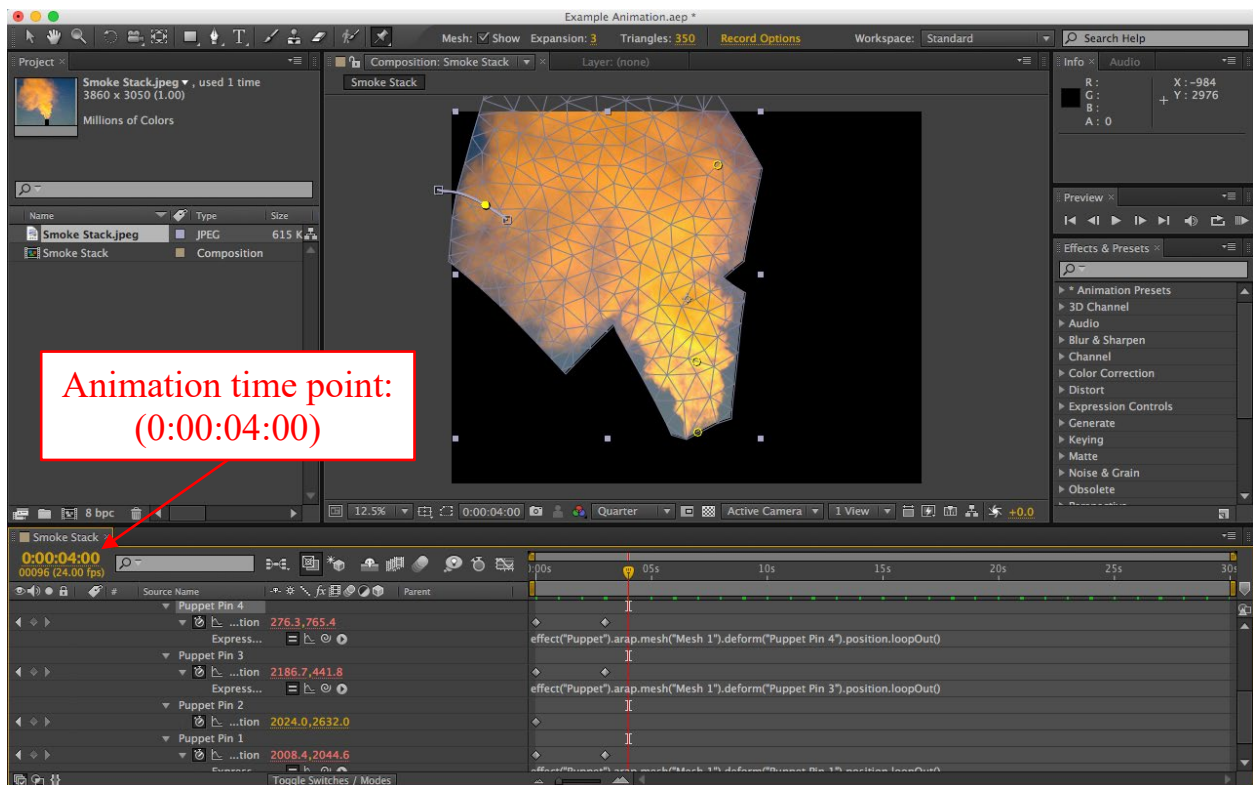
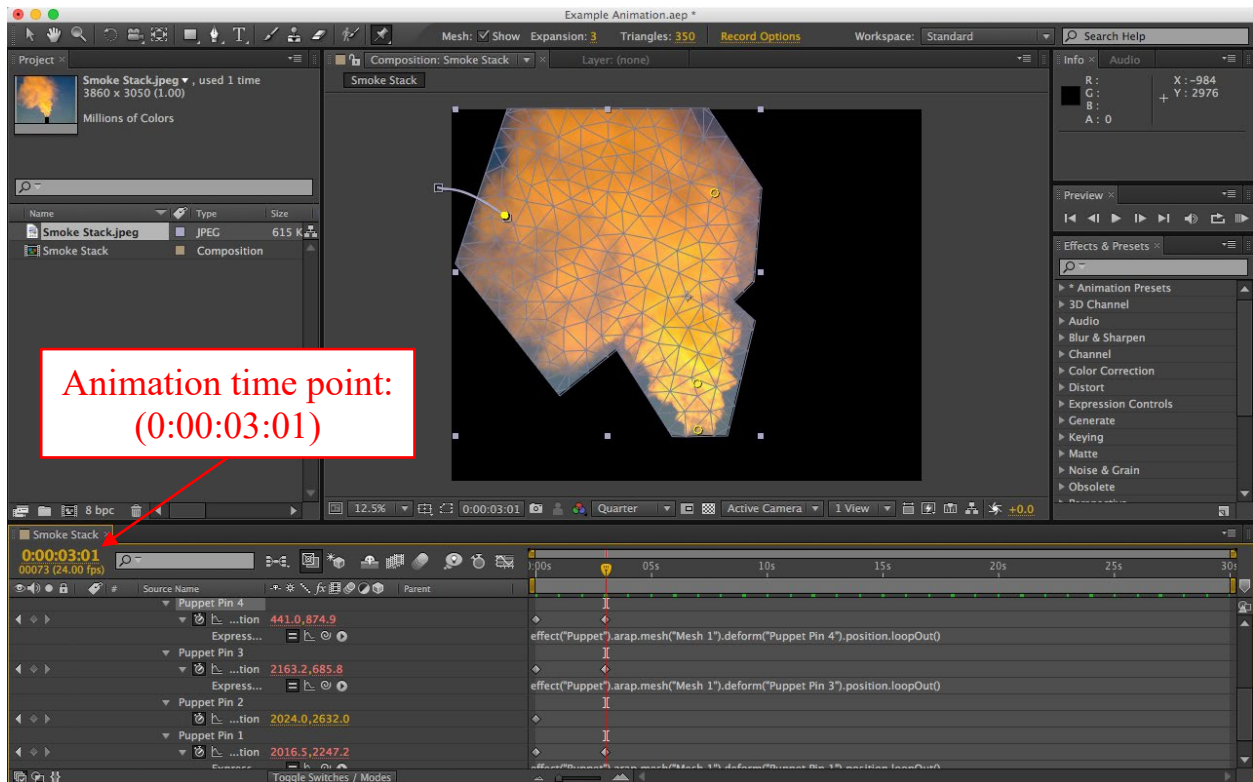
86. Finally, as shown below, a “loopOut()” expression is applied to each of the additional Deform pins as similarly discussed in Section VIII.B.2.e to cause the overall animation to repeatedly loop. See AEM, 562, 219.

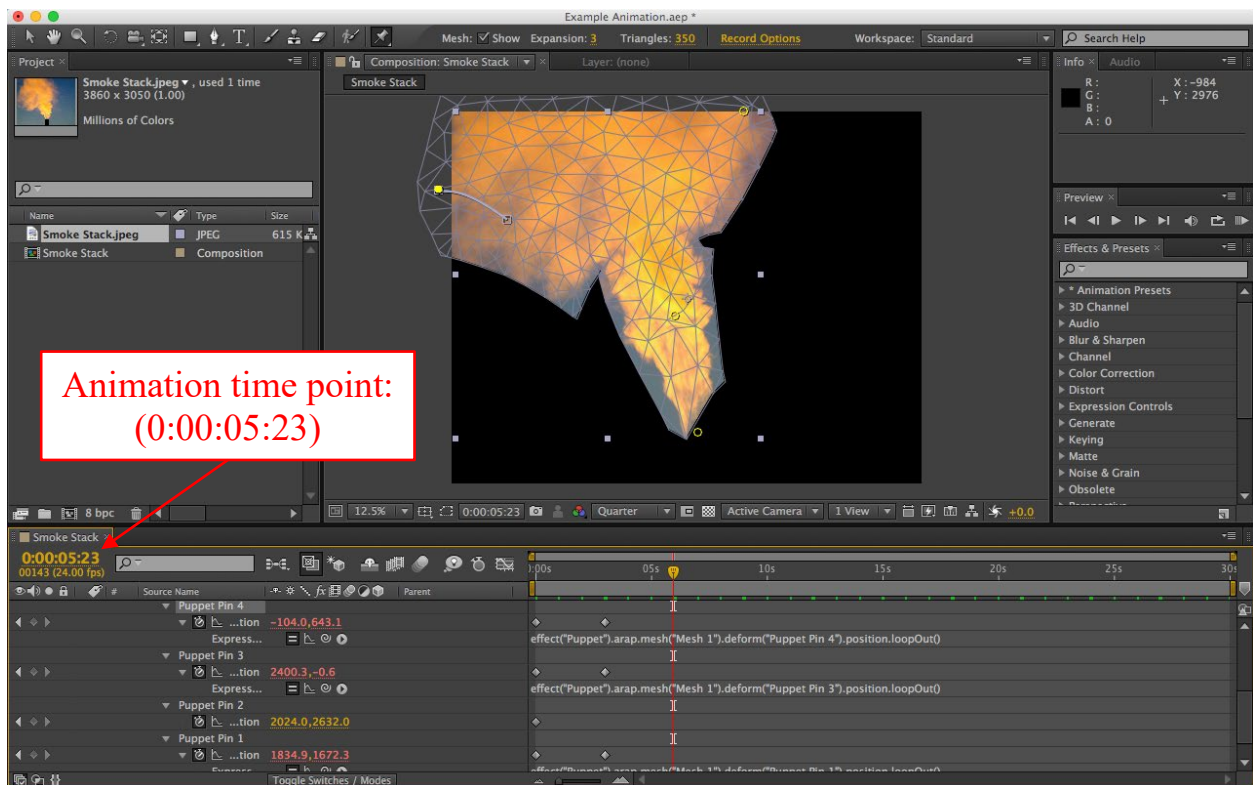
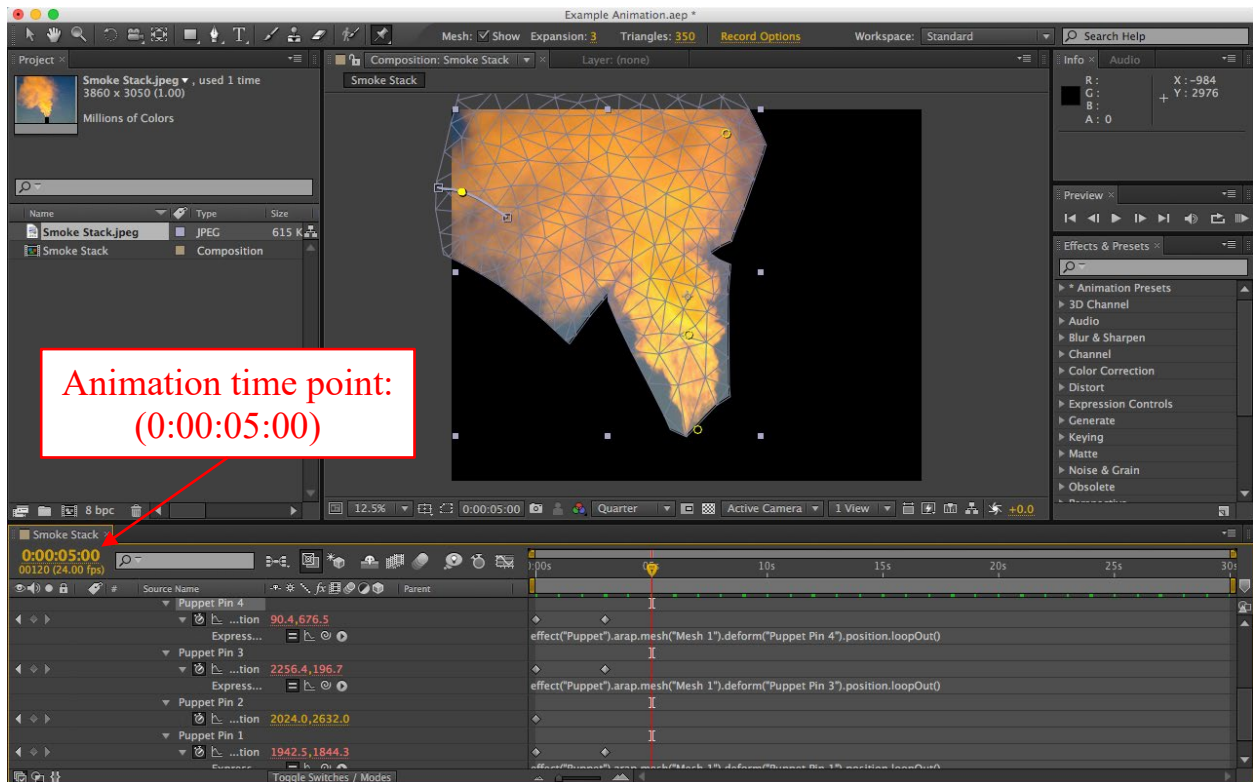


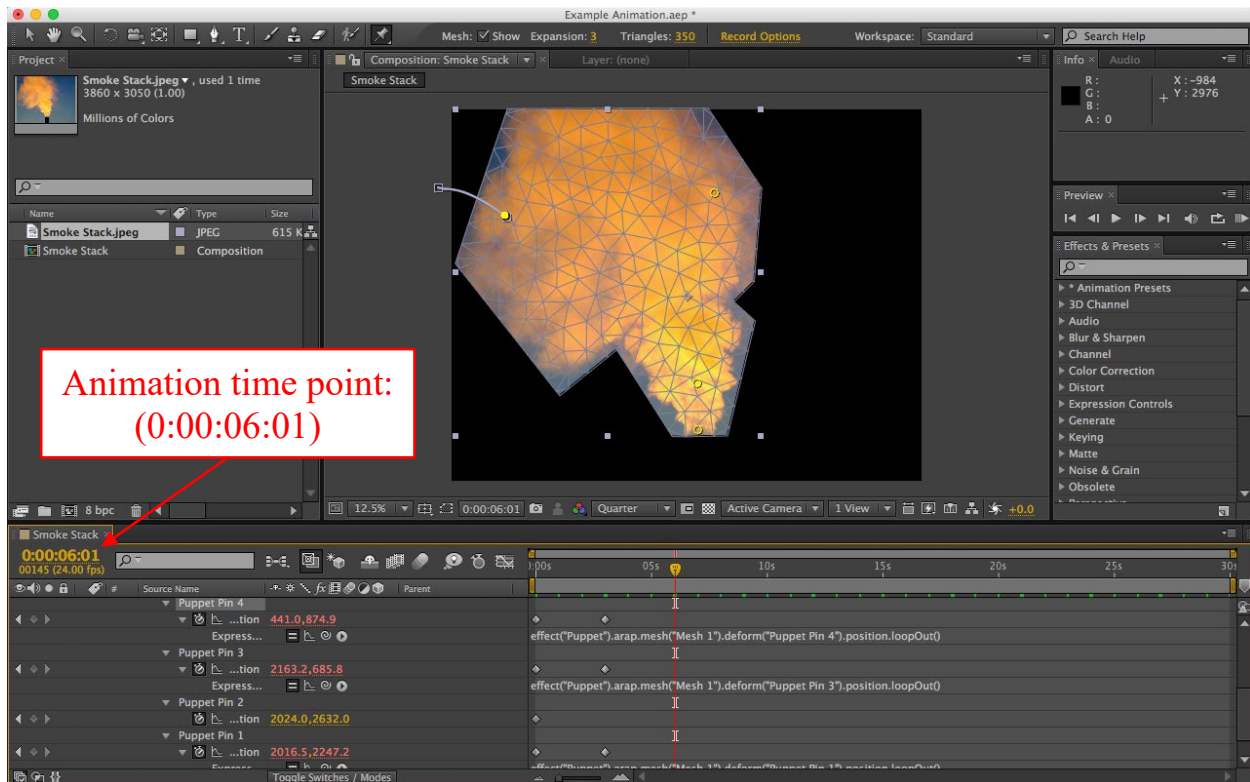
87. The final resulting animation is shown in the below screenshots, which follow the same sequential order as in Section VIII.B.2.f in order to mimic the animation playing in time—i.e., animation time points (0:00:00:00), (0:00:01:00), (0:00:02:00), (0:00:02:23), (0:00:03:01), (0:00:04:00), (0:00:05:00), (0:00:05:23), and (0:00:06:01). *See* Section VIII.B.2.f. I have also included in Attachment C cropped versions of the below screenshots depicting only the “Composition panel” in order to provide a better view of the resulting animation.











3. Independent Claim 1

- a. [1pre]: A computer system for automating the shifting of pixels within a digital image, comprising:

one or more processors; and

one or more computer-readable media having stored thereon executable instructions that when executed by the one or more processors configure the computer system to perform at least the following.

88. AEM teaches that AECS6 is “software” that is installed on computers and run on “64-bit operating systems.” AEM, 19. This necessarily discloses the claimed “computer system,” “processors,” and “computer-readable media.” Indeed, AEM teaches that a user ideally should have three “local disk drives,” i.e., “one for

source footage files, *one from which the application runs*, and one for rendered output.” AEM, 17 (emphasis added). “If you commonly work with complex source compositions, try adding *RAM* or a faster *processor*.” AEM, 38 (emphasis added).

89. Further, in my opinion, AEM describes using AECS6 to animate the movement of pixels within an imported image—i.e., the claimed “automating the shifting of pixels within a digital image.” Sections VIII.B.3.b-VIII.B.3.h.

- b. [1a]: access, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within the digital image.**

90. In my opinion, AEM discloses limitation [1a]. AEM teaches importing images and other footage items into a composition in AECS6—i.e., the claimed “access... a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within the digital image.” AEM, 75, 98. AEM teaches that the user should have a “local disk drive” that is a “source” for such footage files—i.e., the claimed “from memory.” AEM, 17.

- c. [1b]-[1c]: receive a first starting point through a user interface, wherein the first starting point is received through a user selection of a first beginning portion of the digital image;**

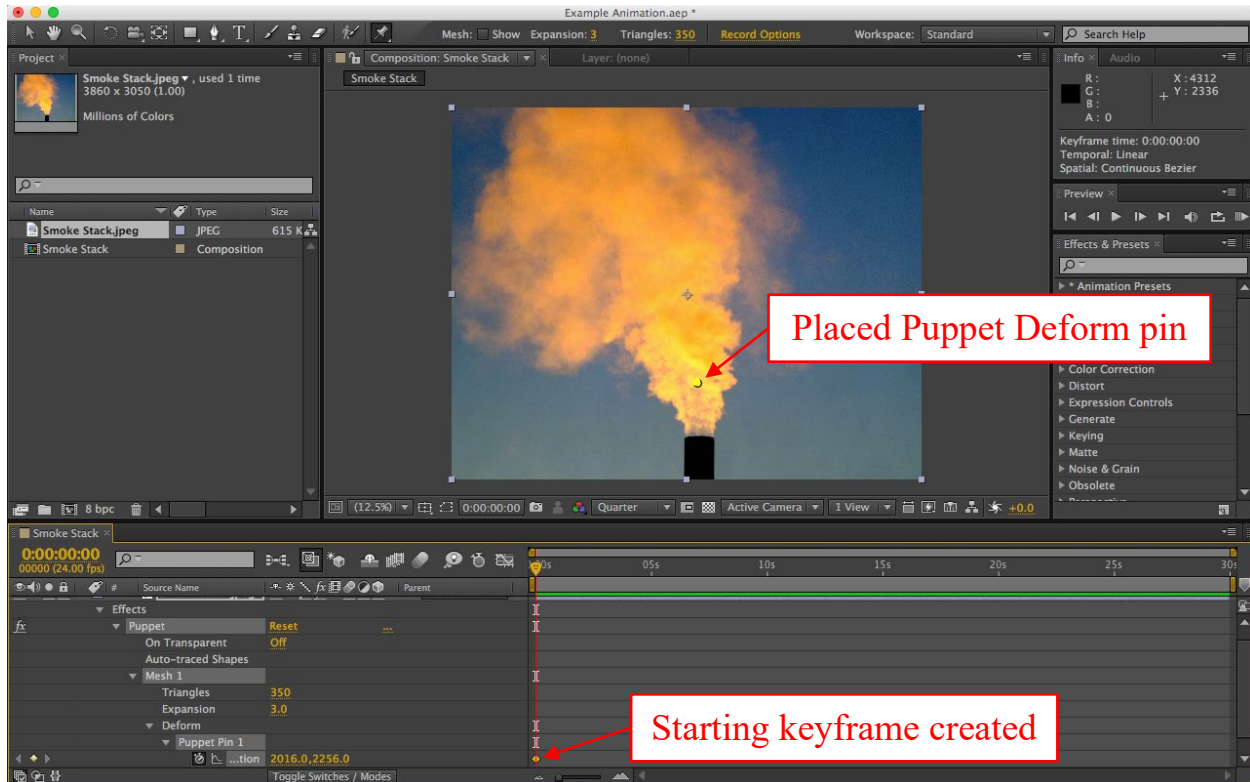
render a first visual indication of the first starting point on the user interface.

91. Regarding the claimed “portion of the digital image,” the ’017 Patent states the claimed “portion” can be a single pixel within the image. ’017 Patent, 6:3-

5 (“The beginning portion of the digital image comprises a starting *pixel*, or area, from which the user wishes pixels to shift.” (emphasis added)), 6:8-10 (“In at least one embodiment, a beginning point comprise[s] a particular *pixel* that is selected by a user....” (emphasis added)), 6:14-16 (“The ending portion of the digital image comprises an ending *pixel*, or area, to which the user wishes pixels to shift.” (emphasis added)).

92. AEM teaches that, when an image or other footage item is imported, the image becomes a layer of a composition—i.e., the claimed “digital image.” AEM, 75, 98. AEM also teaches creating a Puppet effect animation of the layer by placing a Puppet “Deform” pin in the “Composition” or “Layer panel[s]” on a “nontransparent pixel” at a specific part of the layer that the user desires to move, causing AECS6 to create a starting “keyframe” based on both the current time within the animation and the Deform pin’s position—i.e., “receive a first starting point through a user interface, wherein the first starting point is received through a user selection of a first beginning portion of the digital image.” AEM, 218-19; ’017 Patent, 6:3-5, 6:8-1. AECS6 makes this starting keyframe “visible in the Composition panel and Layer panel”—“render a first visual indication of the first starting point on the user interface.” AEM, 218. Thus, in my opinion, AEM discloses limitations [1b]-[1c].

93. By way of further illustration, the annotated screenshot of AECS6 in Section VIII.B.2.b is provided below. The screenshot shows a Deform pin placed in the “Composition panel” of an image of a smokestack, and a starting keyframe that is automatically created by the placement of the Deform pin:



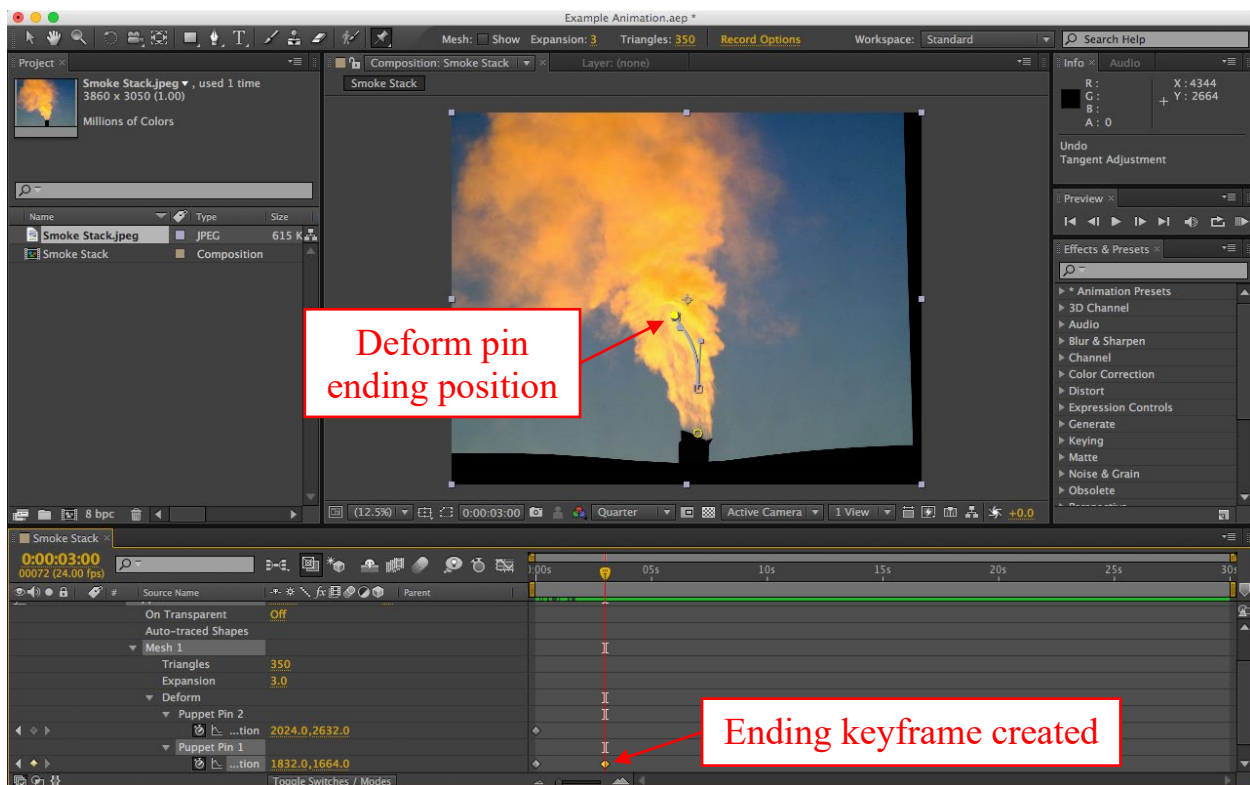
- d. [1d]-[1e]: receive a first ending point through the user interface, wherein the first ending point is received through a user selection of a first ending portion of the digital image;

render a second visual indication of the first ending point on the user interface.

94. Once a starting keyframe for a Deform pin has been defined, AEM instructs the user to create an ending keyframe by “[g]o[ing] to another time in the composition, and mov[ing] the position of... the Deform pin[] by dragging [it] in the

Composition or Layer panel”—i.e., the claimed “receiving an indication of an ending point through a user interface, wherein the ending point is received through a user selection of a different portion of the digital image.” AEM, 218-19; ’017 Patent, 6:14-16. AECS6 makes this ending keyframe “visible in the Composition panel and Layer panel”—“render a second visual indication of the first ending point on the user interface.” AEM, 218. Thus, in my opinion, AEM discloses limitations [1d]-[1e].

95. To further illustrate, the previously discussed (in Section VIII.B.3.c) annotated screenshot of AECS6 is shown again below, but now illustrating the creation of an ending keyframe for the Deform pin:



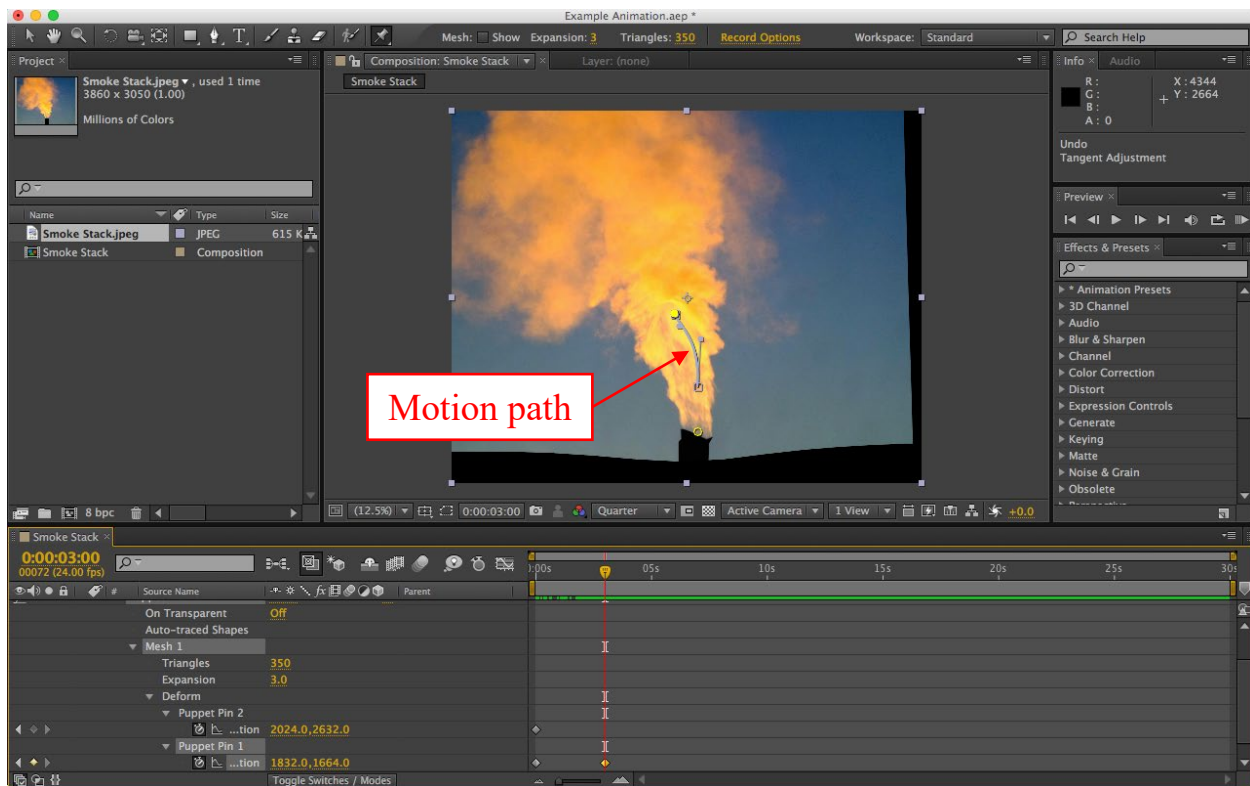
- e. **[1f]: render, on the user interface, a first rendered link between the first starting point and the first ending point, wherein the first rendered link comprises:**
 - a first direction extending from the first starting point to the first ending point; and**
 - a first length between the first starting point and the first ending point.**

96. AEM teaches that, when the user initially places the Deform pin to create a starting keyframe, AECS6 automatically outlines the layer's nontransparent pixels and divides this outline into a "mesh of triangles." AEM, 219 ("Click any nontransparent pixel of a raster layer to apply the Puppet effect and create a mesh for the outline created by auto-tracing the alpha channel of a layer."), 220-21. AEM also teaches that, when the user thereafter creates an ending keyframe for the Deform pin by "[g]o[ing] to another time in the composition, and mov[ing] the position of... the Deform pin[] by dragging [it] in the Composition or Layer panel," the shape of the layer's mesh changes—and thus the layer's nontransparent pixels move—to "accommodate" this repositioning. AEM, 218-19 ("Each part of the mesh is also associated with the pixels of the image, so the pixels move with the mesh.... When you move one or more Deform pins, the mesh changes shape to accommodate this movement, while keeping the overall mesh as rigid as possible."), 177.

97. Further, the creation of an ending keyframe prompts AECS6 to perform animation via "keyframe interpolation," where AECS6 interpolates the Deform pin's

position from the starting to ending keyframe and generates “in-between” frames based on the interpolation, thus animating the movement of the layer’s nontransparent pixels. AEM, 177, 193, 218-19. The interpolation is performed according to, and visually indicated in the “Composition” and “Layer panel[s]” by, a “motion path,” which extends from the starting position/keyframe to the ending position/keyframe. AEM, 187, 218-19. The visual indication of such a motion path discloses “render, on the user interface, a first rendered link between the first starting point and the first ending point, wherein the first rendered link comprises: a first direction extending from the first starting point to the first ending point; and a first length between the first starting point and the first ending point” as recited in limitation [1f].

98. To further illustrate, the previously discussed (in Section VIII.B.3.c) screenshot of AECS6 is shown again below, but with further annotations highlighting the motion path created between the Deform pin’s starting position/keyframe and ending position/keyframe:



Additional screenshots illustrating how this motion path was created can be found in Section VIII.B.2.c.

- f. [1g]: identify a first set of pixels that are:

parallel to the first rendered link,

within a user-defined threshold distance from the first rendered link, and

extending from the first starting point to the first ending point.

99. Regarding the claimed “identify a first set of pixels,” the ’017 Patent provides additional context by stating that “[t]he size of the group of pixels may be user selectable or automatically determined.” ’017 Patent, 6:55-56. For example, a user can perform the claimed “identify[ing]” of a “first set of pixels” by “increas[ing]

the threshold distances that the set of pixels extend beyond the link.” ’017 Patent, 6:57-58. Particularly, the user can identify certain pixels between the starting and ending points to be “covered by a mask” and excluded from the set of pixels to be shifted. ’017 Patent, 6:52-55, 4:48-52. Alternatively, the software can perform the claimed “identify[ing]” automatically by “identif[ying] sets of pixels that lie between the respective starting points and the respective ending points”—e.g., identifying pixels between the starting and ending points that are or are not “covered by a mask.” ’017 Patent, 6:49-56.

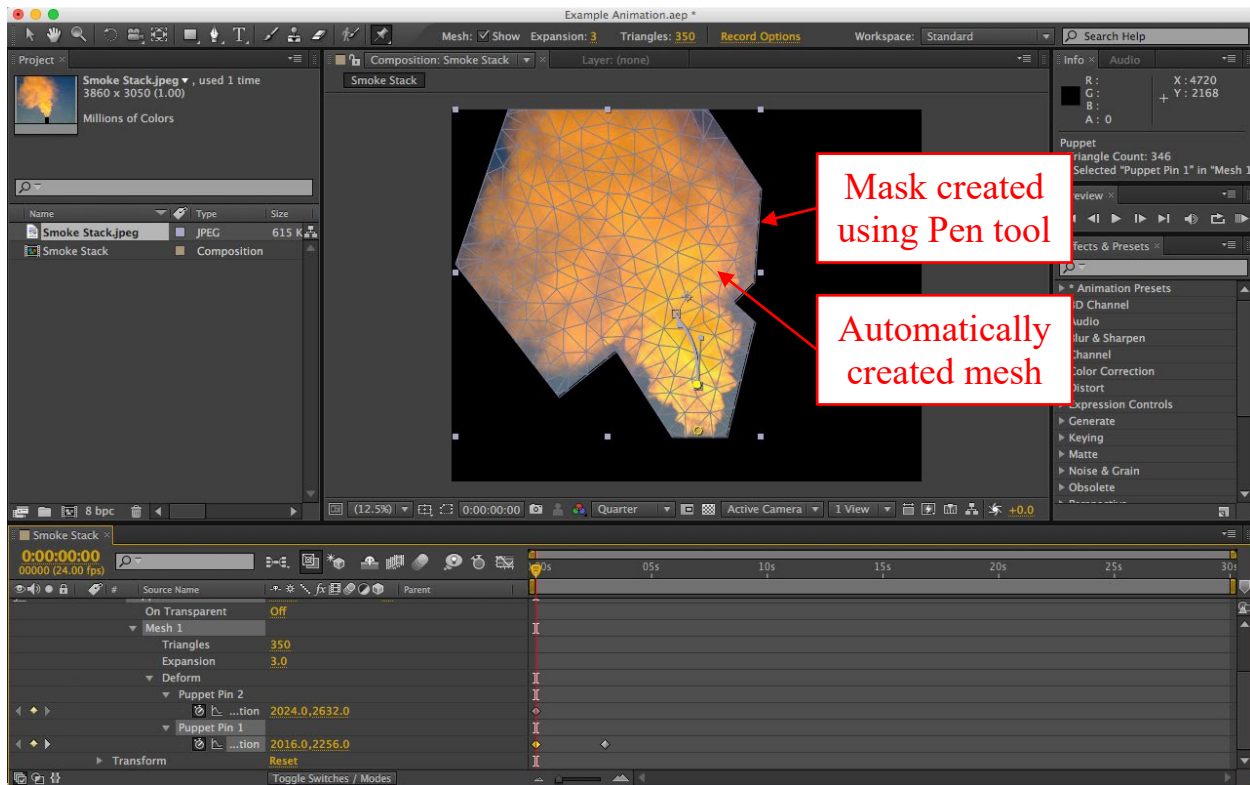
100. In my opinion, AEM meets limitation [1g]. AEM teaches selecting certain pixels of the layer to be transparent or nontransparent (i.e., modifying the “alpha channel” of the layer) by applying either a “mask” or a “matte,” where the former is created using, e.g., the “Pen” tool or “Auto-trace” function, and the latter is created using, e.g., the “Roto Brush” tool. AEM, 315, 318.

101. Thus, as AEM teaches, AECS6 necessarily allows a user to apply either a mask or matte to select any or all pixels of the layer to be nontransparent and thus included in the layer’s mesh to be moved and animated, including pixels parallel to the Deform pin’s motion path—i.e., the claimed “parallel to the first rendered link”—within a user-defined threshold distance from the motion path—i.e., “within a user-defined threshold distance from the first rendered link”—and extending from the Deform pin’s starting position/keyframe to the ending position/keyframe—i.e.,

“extending from the first starting point to the first ending point.” AEM, 218-19, 315, 194-95; ’017 Patent, 6:52-56 (“The size of the group of pixels may be *user selectable*....” (emphasis added)). Further, AECS6 automatically creates the mesh from the layer’s nontransparent pixels, which additionally discloses limitation [1g]. AEM, 218-19, 194-95; ’017 Patent, 6:52-56 (“The size of the group of pixels may be... *automatically determined*.” (emphasis added)).

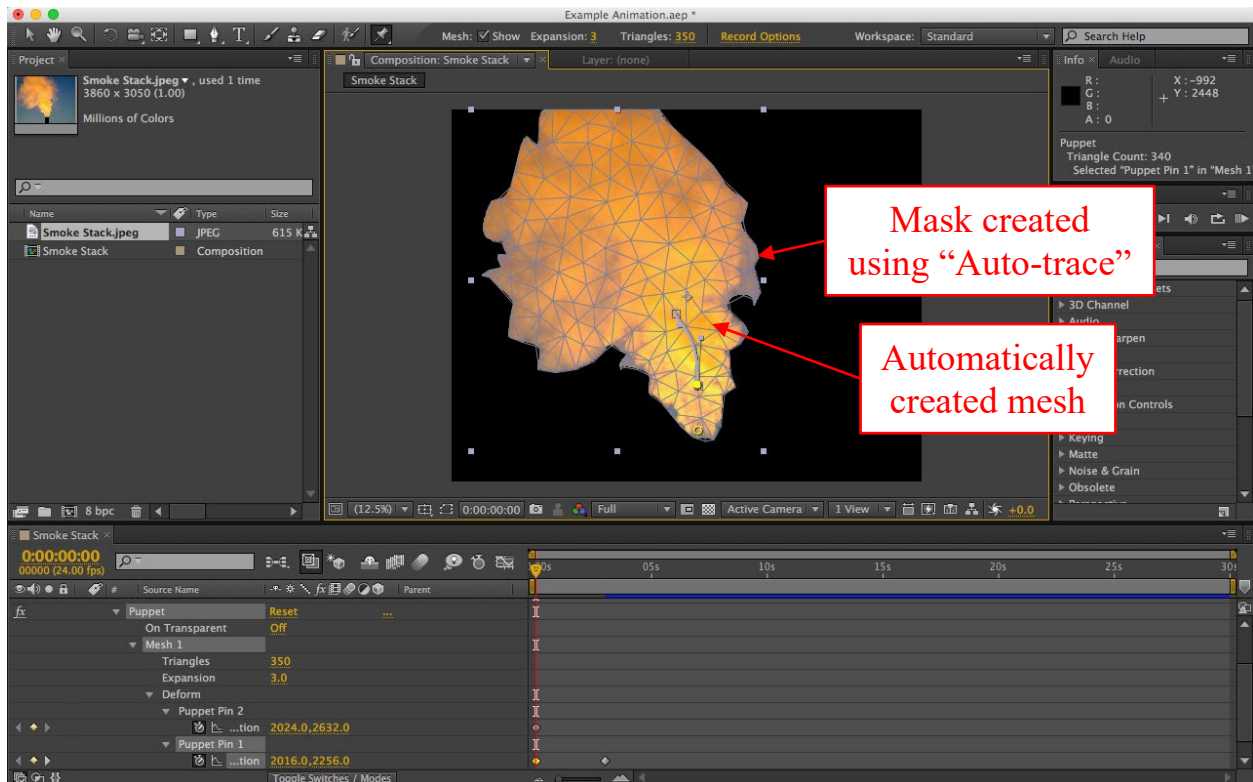
102. Separately, a POSITA would have been motivated to select and make nontransparent, using a mask or matte, those pixels parallel to the Deform pin’s motion path, within a user-defined threshold distance from the motion path, and extending from the Deform pin’s starting to ending position, because such pixels would in common instances be pixels that the user desires to include in the mesh to be moved and animated. For instance, to move and animate the smoke rising from the smokestack depicted in Section VIII.B.3.e’s screenshot using the Puppet effect, a POSITA would have recognized to apply a Deform pin within the smoke, create for the Deform pin a motion path rising through the smoke, and ensure the smoke’s pixels are included in the mesh to be moved by selecting and making nontransparent

all pixels of the smoke using, e.g., a mask created with the Pen tool, as shown below⁹:

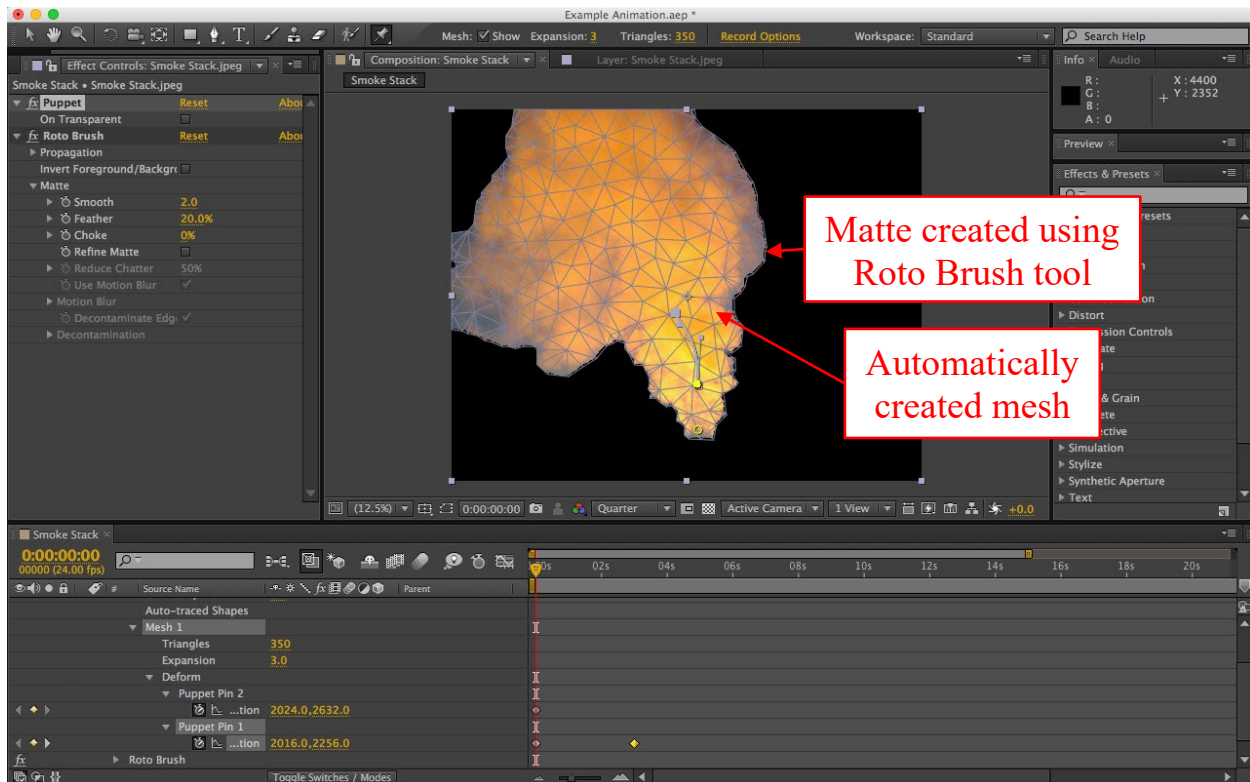


⁹ Okabe provides further support by depicting a similar instance in its Figure 1, which depicts user-drawn “strokes” going downwards through a waterfall, and a “matte” specifying a to-be-animated “region of interest” (in white) that includes all pixels of the waterfall, including those parallel to each stroke, within a user-defined threshold distance from the strokes to the waterfall’s edges, and extending from the starting point to the ending point of each stroke. Okabe, 7, 2; *see also* Section VIII.C.4.f.

As shown, the mask makes nontransparent, and thus the mesh includes, pixels parallel to the Deform pin's motion path, within a user-defined threshold distance from the motion path (i.e., to the smoke's edges), and extending from the Deform pin's starting to ending position. The following additionally illustrates how a similar mask and mesh is created using the “Auto-trace” function:



Further, the following shows how a similar result is achieved by creating a matte using the “Roto Brush” tool:



Additional screenshots illustrating how each of the above meshes were created can be found, respectively, in Sections VIII.B.2.d.i, VIII.B.2.d.ii, and VIII.B.2.d.iii.

103. The above is also consistent with Plotagraph’s infringement contentions in *Plotagraph, Inc. v. Lightricks Ltd.*, Civil Action No. 4:21-cv-03873, Dkt. No. 38 (S.D. Tex. May 5, 2022) (EX1022). Plotagraph states that limitation [1g]¹⁰ is infringed by Petitioner’s “Motionleap” app when the user, after “using the Direction tool” to draw the claimed link, “defines the threshold distance from the

¹⁰ Plotagraph’s assertions are directed to claim 11 of the related U.S. Patent No. 10,621,469, which recites “selecting a first set of pixels” identical to that in limitation [1g]. EX1022, 10-11.

first digital link by using the Select tool to paint an area to be animated,” where the area also includes pixels parallel to the link and extending from the link’s starting to ending point. EX1022,¹¹ 9-11. Thus, according to Plotagraph, a user’s freehand selection of a to-be-animated area that includes pixels surrounding the link meets limitation [1g], where the distance from the link to the area’s boundary meets the specifically recited “user-defined threshold distance.” EX1022, 10-11. And AEM discloses selecting pixels this same way using AECS6’s “Pen” tool, “Auto-trace” function, and “Roto Brush” tool, as discussed above. AEM, 315, 318. Accordingly, AEM discloses limitation [1g].

- g. [1h]: automatically shift the first set of pixels in the first direction, wherein shifting the first set of pixels comprises rendering and re-rendering in a loop the first set of pixels being shifted.**

104. Regarding the claimed “shifting,” the ’017 Patent states the pixel shifting can use a “mesh algorithm” that “triangulates the mesh using defined points and then calculates an affine transformation for every angle.” ’017 Patent, 8:28-37. The ’017 Patent’s Figure 4 depicts an example, where “a starting mesh and ending mesh [are] overlaid on [a] digital image of [a] house and landscape”:

¹¹ Although the contentions are marked “CONFIDENTIAL,” I have been informed by counsel that the contentions are actually public because Plotagraph submitted them as a public exhibit to a publicly filed motion.

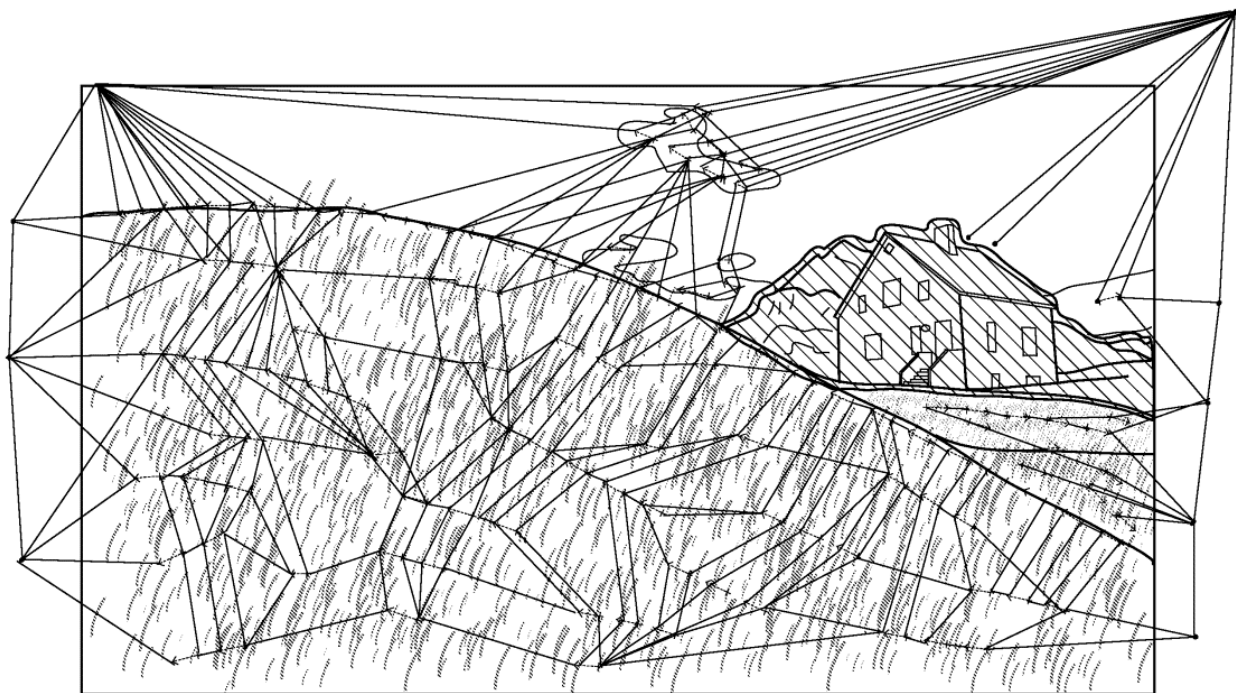


Fig. 4

'017 Patent, Fig. 4, 6:39-48.

105. In my opinion, AEM discloses limitation [1h]. AEM similarly teaches that, when a user places an initial Deform pin, AECS6 creates a mesh from the layer's nontransparent pixels. AEM, 219, 220-21; Section VIII.B.3.e. AEM also teaches, and a POSITA would have been motivated to perform, selecting and making nontransparent, using a mask or matte, pixels parallel to the Deform pin's motion path, within a user-defined threshold distance from the motion path, and extending from the Deform pin's starting position/keyframe to the ending position/keyframe—i.e., the claimed “first set of pixels.” AEM, 218-19, 315, 194-95; Section VIII.B.3.f. When a motion path for the Deform pin is created, AECS6 animates the corresponding change to the shape of the layer's mesh, and thus the movement of

the layer's nontransparent pixels accommodating the Deform pin's repositioning—i.e., “automatically shift the first set of pixels in the first direction.” AEM, 218-19, 194-95; '017 Patent, 8:34-37; Section VIII.B.3.e.

106. Further, AEM teaches applying a “loopOut()” expression to the Deform pin, which, by default, causes “all keyframes [to] loop.” AEM, 562, 219 (“You can use expressions to link the positions of Deform pins to motion tracking data, audio amplitude keyframes, or any other properties.”). Specifically, by applying a default “loopOut()” expression to a Deform pin, the Deform pin's position will be reset to the starting keyframe upon reaching the ending keyframe during keyframe interpolation, and the interpolation will be repeatedly looped then and thereafter until otherwise specified by the user—i.e., “rendering and re-rendering in a loop the first set of pixels being shifted.” AEM, 562, 219.

107. The screenshot in Section VIII.B.2.e illustrates how a “loopOut()” expression can be applied to the Deform pin in the exemplary composition therein. The series of screenshots provided in Section VIII.B.2.f show the resulting animation at different, sequential time points to mimic the animation playing in time. Cropped versions for these screenshots are also provided in the table in Attachment B to provide a better view of the resulting animation.

- h. [1i]-[1o]: receive a second starting point through the user interface, wherein the second starting point is received through a user selection of a second starting portion of the digital image, the second starting portion being different than both the first starting portion and the first ending portion;**

render a third visual indication of the second starting point on the user interface;

receive, through the user interface, a second ending point associated with the second starting point, wherein the second ending point is received through a user selection of a second ending portion of the digital image, the second ending portion being different than both the first starting portion and the first ending portion;

render a fourth visual indication of the second end point on the user interface;

render, on the user interface, a second link between the second starting point and the second ending point;

identify a second set of pixels that are:

parallel to the second rendered link,

within the user-defined threshold distance from the first rendered link, and

extending from the second starting point to the second ending point; and

automatically shift the second set of pixels between the second starting point and the second ending point, wherein shifting the second set of pixels comprises rendering in a loop the second set of pixels being shifted.

108. In my opinion, AEM discloses limitations [1i]-[1o]. Specifically, AEM teaches how to move and animate different parts of a layer through the use of multiple Deform pins. AEM, 218-19. Like when placing a first Deform pin (*see* Section VIII.B.3.c), a user places additional Deform pins and creates for each a starting keyframe visible in the Composition and Layer panels by simply “[c]lick[ing] in one *or more* places within the outline”—i.e., “receive a second starting point through the user interface, wherein the second starting point is received through a user selection of a second starting portion of the digital image, the second starting portion being different than both the first starting portion and the first ending portion” as recited in limitation [1i], and “render a third visual indication of the second starting point on the user interface” as recited in limitation [1j]. AEM, 219 (emphasis added). The user creates different ending keyframes and motion paths for these additional Deform pins in the same way as for the first Deform pin—i.e., “receive, through the user interface, a second ending point associated with the second starting point, wherein the second ending point is received through a user selection of a second ending portion of the digital image, the second ending portion being different than both the first starting portion and the first ending portion” and “render a fourth visual indication of the second end point on the user interface” as recited in limitations [1k]-[1l], and “render, on the user interface, a second link between the second starting point and the second ending point” as recited in

limitation [1m]. AEM, 218-19; Sections VIII.B.3.d-VIII.B.3.e. Further, for similar reasons discussed in Section VIII.B.3.f, AEM teaches, and a POSITA would have been motivated to perform, creating a mask or matte to select any or all pixels of the layer to be nontransparent and thus included in the layer's mesh to be moved and animated, including pixels parallel each corresponding Deform pin's motion path, within the user-defined threshold distance from the first Deform pin's motion path (e.g., to the smoke's edges in Section VIII.B.3.f's exemplary compositions), and extending from the starting position/keyframe to the ending position/keyframe of each Deform Pin¹²—i.e., “identify a second set of pixels that are: parallel to the second rendered link, within the user-defined threshold distance from the first rendered link, and extending from the second starting point to the second ending point” as recited in limitation [1n]. AEM, 218-19, 315, 194-95; *accord* EX1022,

¹² Okabe again provides further support that a POSITA would have been motivated to perform these steps by depicting, in its Figure 1, not just one but three user-drawn “strokes” going downwards through the waterfall, where the to-be-animated “region of interest” includes all pixels of the waterfall, including those parallel to each stroke, within the user-defined threshold distance from the strokes to the waterfall's edges, and extending from the starting point to the ending point of each stroke. Okabe, 7, 2; *see also* Section VIII.C.4.h.

10-11. Finally, as similarly discussed in Section VIII.B.3.g, AEM teaches that AECS6 animates the change to the shape of the layer's mesh, and thus the movement of the layer's nontransparent pixels, that accommodates each Deform pin's repositioning, which is repeatedly looped by applying a "loopOut()" expression to each Deform pin—i.e., "shifting the second set of pixels in the second direction, wherein shifting the second set of pixels comprises rendering and re-rendering in a loop the second set of pixels being shifted" recited in limitation [13d]. AEM, 218-19, 194-95, 562.

109. Attachment C further illustrates the foregoing by including a table that provides a series of cropped screenshots of AECS6 like those in Attachment B, but the exemplary composition now includes multiple Deform pins in addition to the first. Additional screenshots in Section VIII.B.2.g show how the animation depicted in Attachment C was created using the multiple Deform pins, including the uncropped versions of the screenshots in Attachment C's table.

4. Claim 2

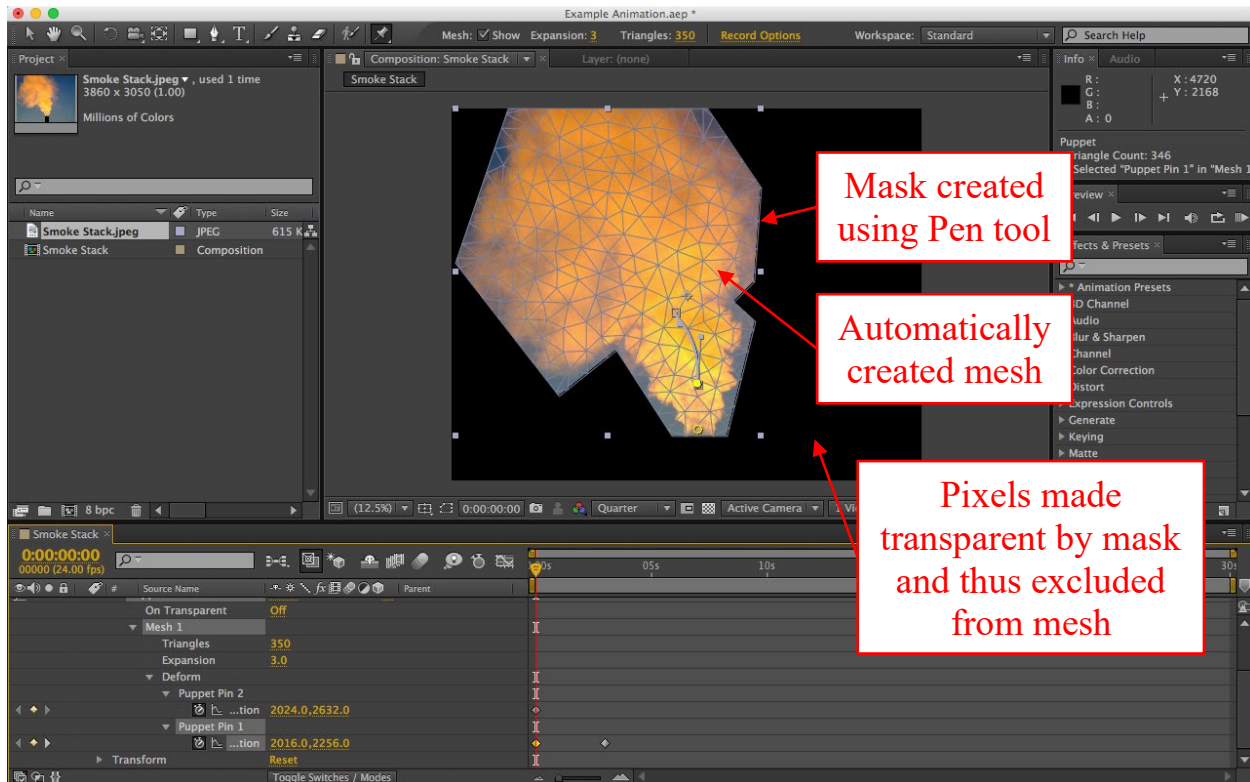
- a. **[2a]-[2b]: The computer system of claim 1, wherein the executable instructions include instructions that are executable to configure the computer system to identify a particular portion of the digital image to mask, wherein the mask prevents pixels covered by the mask from being shifted.**

110. Claim 2 is dependent from claim 1. AEM teaches using a mask or matte to select pixels to be nontransparent and thus included in the layer's mesh to be

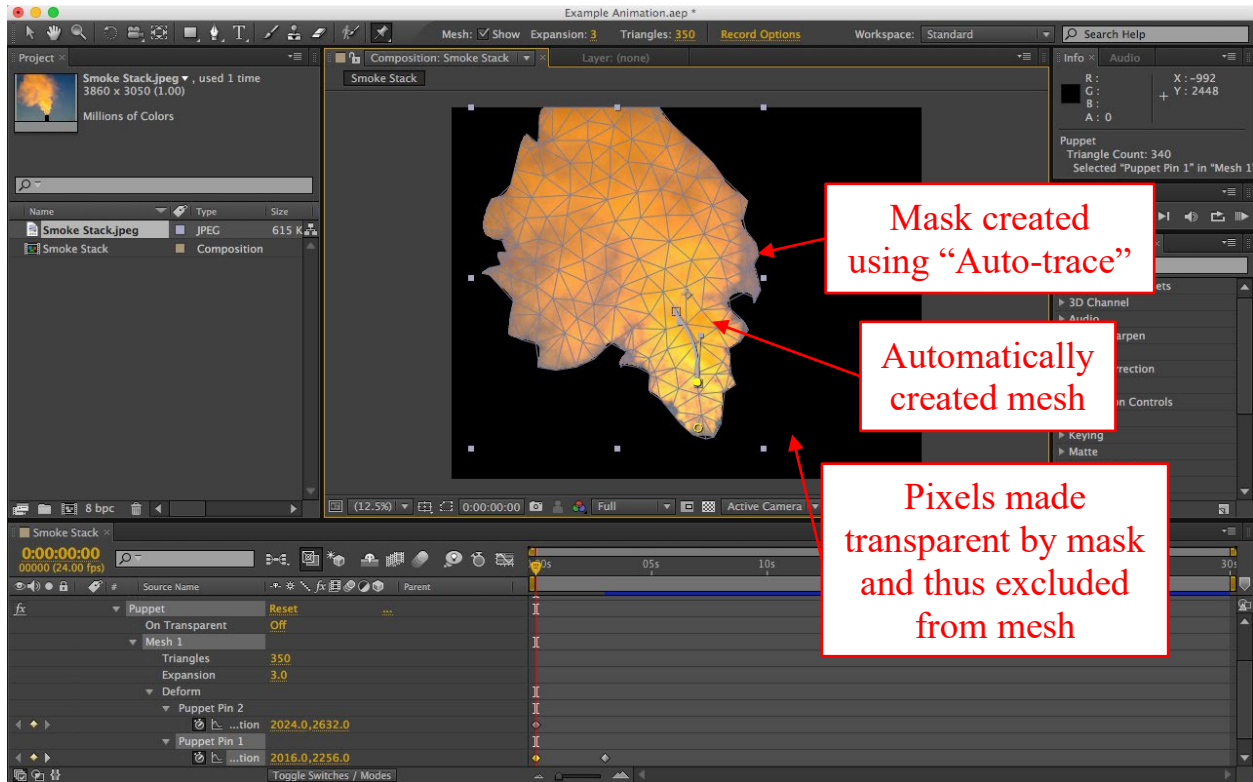
moved and animated by the Puppet effect. AEM, 315, 218-19; Section VIII.B.3.f. Inherently, such a mask or matte is therefore also used to select one or more pixels to be transparent and thus *excluded* from the layer's mesh so as to *not* be moved and animated—i.e., the claimed “mask [that] prevents pixels covered by the mask from being shifted” as recited in [2b]. Indeed, AEM teaches how to toggle between making selected pixels transparent or nontransparent. AEM, 104 (“If you want to switch the opaque and transparent areas of the image, select Invert Alpha.”), 318 (“To invert what is considered inside and what is considered outside for a specific mask, select Invert next to the mask name in the Timeline panel.”), 328 (“[D]raw a background stroke on the area that you want to define as the background.”).

111. AEM also teaches creating a mask using the “Pen” tool by clicking on different points on the layer in the “Composition” or “Layer panel[s]” to specify the path and vertices of the mask, or the “Auto-trace” function to “search[] for edges” across the entire layer and create a mask tracing such edges, both of which disclose the claimed “identify a particular portion of the digital image to mask” as recited in limitation [2a]. AEM, 262, 264-65. As for a matte, AEM teaches creating a matte using the “Roto Brush” tool to draw “strokes” in the “Layer panel” over foreground and background elements to be separated, which likewise discloses the claimed “identify a particular portion of the digital image to mask” recited in limitation [2a]. AEM, 328. Thus, in my opinion, AEM discloses limitations [2a]-[2b].

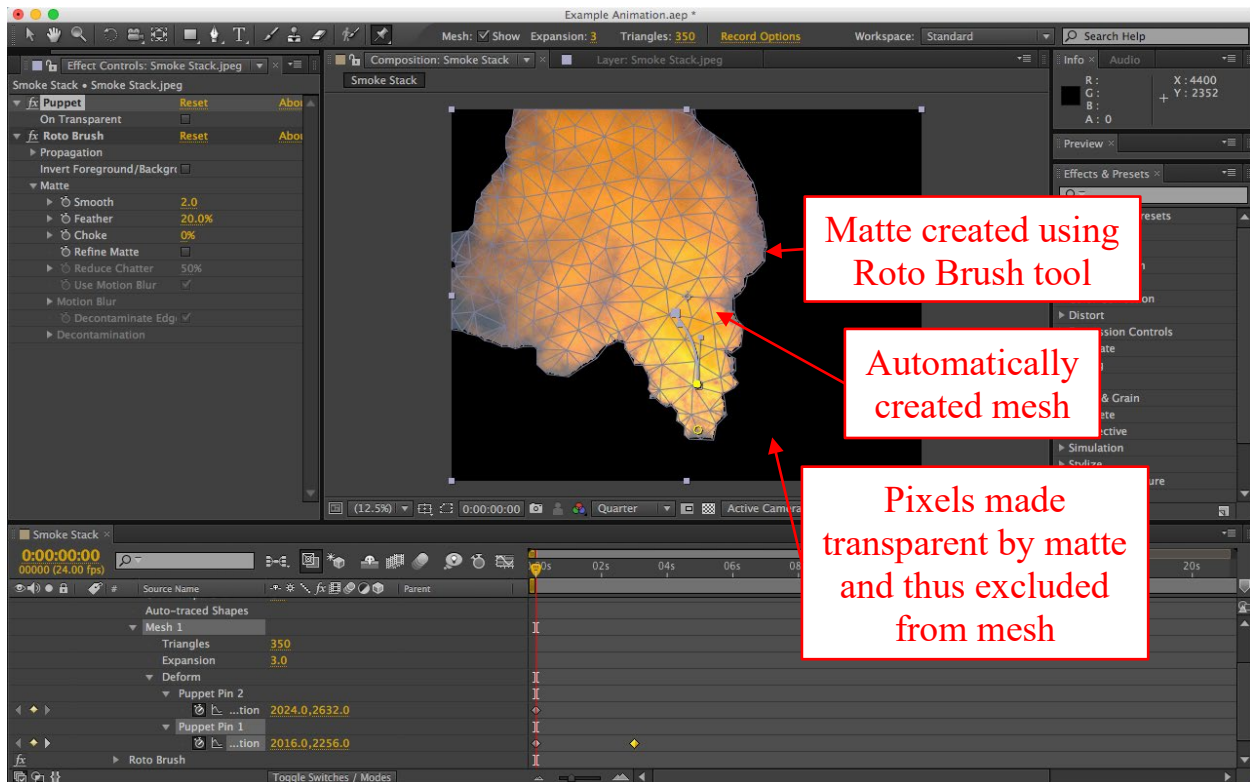
112. Section VIII.B.3.f’s first screenshot depicting a mask created using the “Pen” tool is shown below, but now indicating pixels of the layer made transparent by the mask and thus excluded from the mesh:



Section VIII.B.3.f’s second screenshot, which depicts a mask created using the “Auto-trace” function, is shown below, but now again indicating pixels of the layer excluded from the mesh by the mask:



Section VIII.B.3.f’s third screenshot, which shows a matte created using the “Roto Brush” tool, is provided below, but now indicating pixels of the layer excluded from the mesh by the matte:



5. **Claim 3**: The computer system of claim 2, wherein receiving an indication of a particular portion of the digital image to mask comprises receiving through a user interface a selection of a particular pixel within the digital image.

113. Claim 3 is dependent from claim 2. AEM teaches selecting, in the “Composition” or “Layer panel[s],” one or more pixels to be transparent and thus excluded from a layer’s mesh so as to not be moved and animated with the Puppet effect—i.e., the claimed “wherein receiving an indication of a particular portion of the digital image to mask comprises receiving through a user interface a selection of a particular pixel within the digital image”—by creating a mask using the “Pen” tool to specify the path and vertices of the mask, creating a mask using the “Auto-trace” function to “search[] for edges” across the entire layer and create a mask tracing such

edges, or creating a matte using the Roto Brush tool to draw over foreground and background elements to be separated. AEM, 318, 262, 264-65, 328; Section VIII.B.4.a. Therefore, in my opinion, AEM discloses claim 3.

6. Claim 4

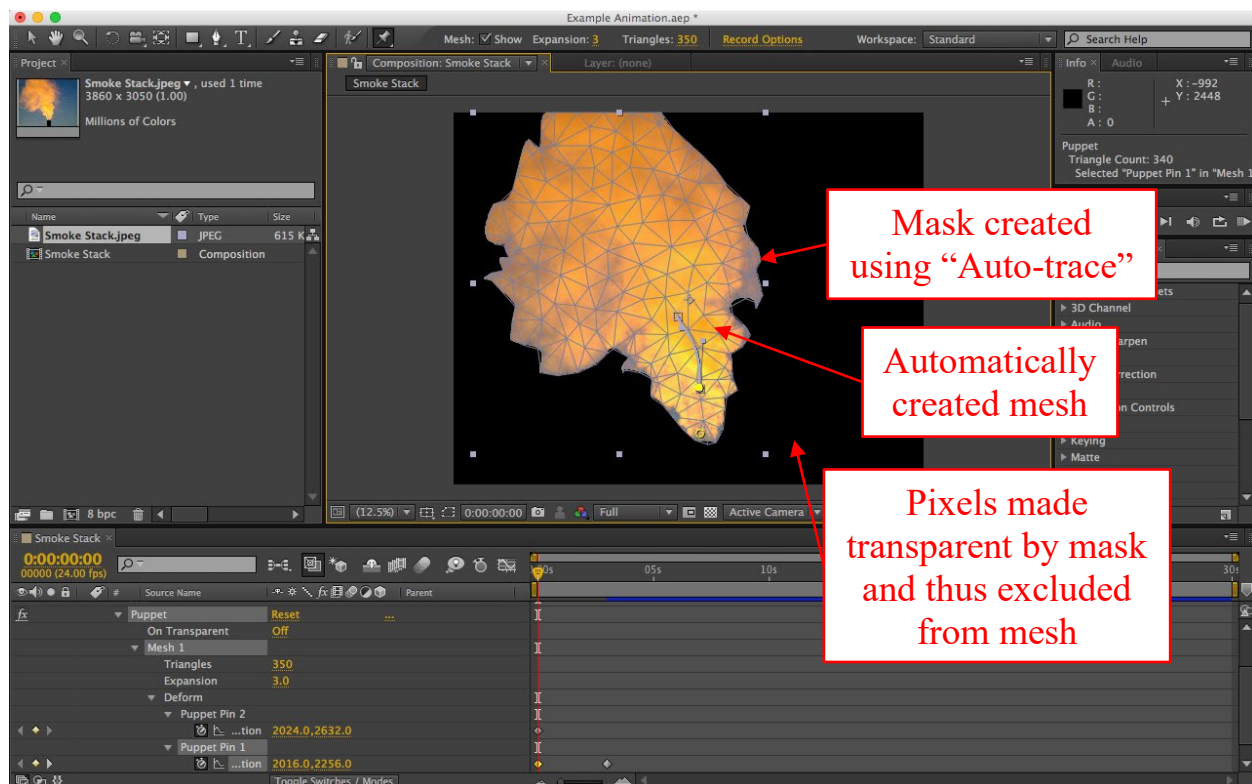
- a. **[4a]-[4c]: The computer system of claim 2, wherein the executable instructions include instructions that are executable to configure the computer system to generate the mask by:**

identifying one or more edges that form a boundary around the particular pixel; and

generating the mask to cover area within the boundary.

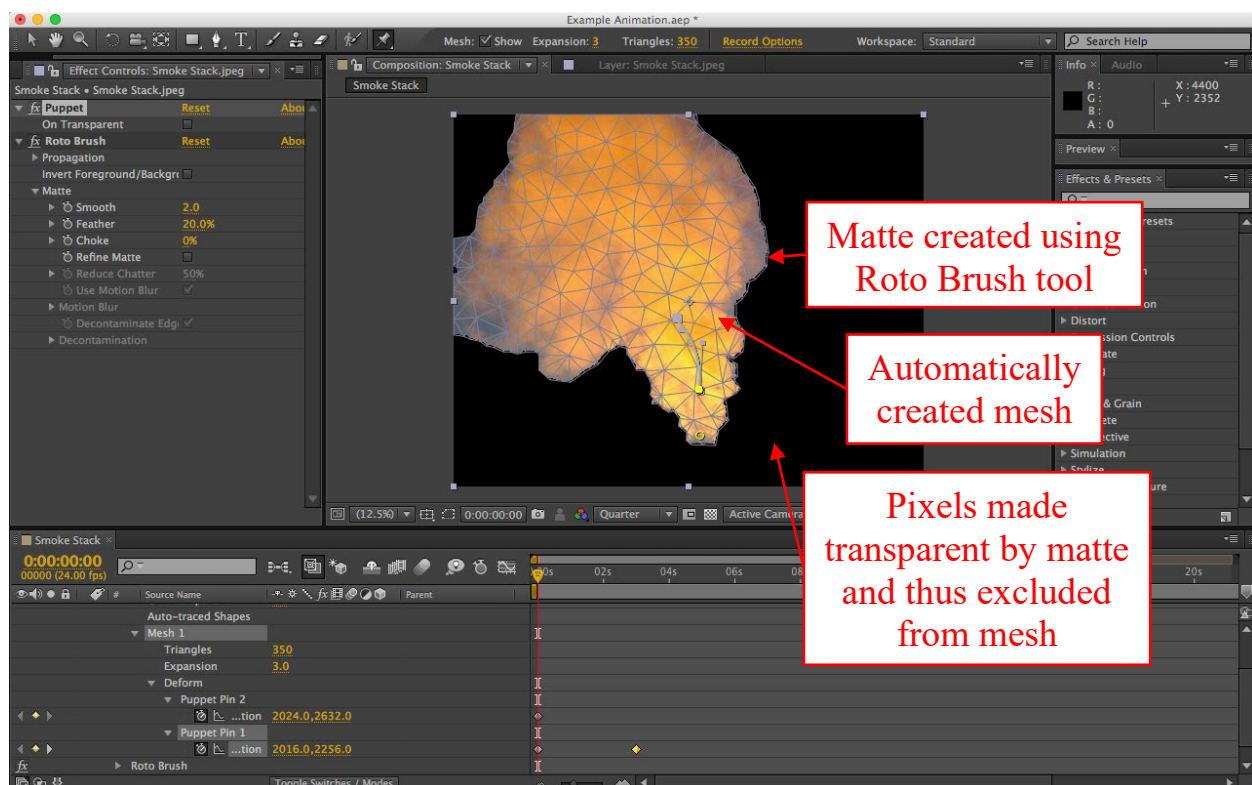
114. Claim 4 is dependent from claim 2. AEM teaches that a mask is automatically created for a layer when using AECS6's "Auto-trace" function—i.e., the claimed "generate the mask" as recited in limitation [4a]. AEM, 262. This "Auto-trace" function "search[es] for edges" across the entire layer based on the value of each pixel's "alpha, red, green, blue, or luminance channel"—i.e., "identifying one or more edges that form a boundary around the particular pixel" recited in limitation [4b]—and generates a mask that outlines such edges—i.e., "generating the mask to cover area within the boundary" as recited in limitation [4c]. AEM, 262. In particular, Auto-trace compares the "alpha, red, green, blue, or luminance channel" of each of the layer's pixels to a user-set "Threshold," which is a "percentage" value. AEM, 262. "Pixels with channel values over the threshold

are mapped to white and are opaque; pixels with values under the threshold are mapped to black and are transparent.” AEM, 262. As such, a mask is created for the layer that traces the edges between pixels with values above and below the selected threshold. *See* AEM, 262. Additionally, “[y]ou can modify a mask created with Auto-trace as you would any other mask” (AEM, 262), such as toggling between whether pixels enclosed within the mask are made transparent or nontransparent (AEM, 104, 318, 328). Indeed, Section VIII.B.4.a’s second screenshot shows that the mask traces the luminance edge between the billowing smoke and the sky, where only the smoke’s pixels are nontransparent:



115. Further, AEM teaches creating a matte using the “Roto Brush” tool, which, similar to the Auto-trace function (*see* AEM, 262), uses “Edge Detection” to

detect a “segmentation boundary” that separates a layer’s foreground and background elements based on “strokes” drawn by the user over “representative areas of the foreground and background elements.” AEM, 328-31. This additionally discloses the claimed “generate the mask” and “identifying one or more edges that form a boundary around the particular pixel.” *See* ’017 Patent, 5:1-22. Further, as AEM teaches, AECS6 applies a matte that isolates the foreground from the background, thereby making the latter transparent, and thus AEM discloses the claimed “generating the mask to cover area within the boundary.” AEM, 328, 318. Indeed, Section VIII.B.4.a’s third screenshot shows the creation of a matte using the “Roto Brush” tool:



As shown, the segmentation boundary between the foreground (the billowing smoke) and background (the sky, which has been made transparent by the matte) traces the edge between the billowing smoke and the sky. Therefore, in my opinion, AEM discloses limitations [4a]-[4c].

7. **Claim 5: The computer system of claim 1, wherein the first direction is different from the second direction.**

116. Claim 5 is dependent from claim 1. AEM teaches creating different motion paths for each Deform pin placed on a layer, thereby disclosing the claimed language. AEM, 218-19; Section VIII.B.3.h. Indeed, as the example in Attachment C's table (discussed in Section VIII.B.3.h) shows, each Deform pin has a separate motion path that differs from the others. AEM therefore, in my opinion, discloses claim 5.

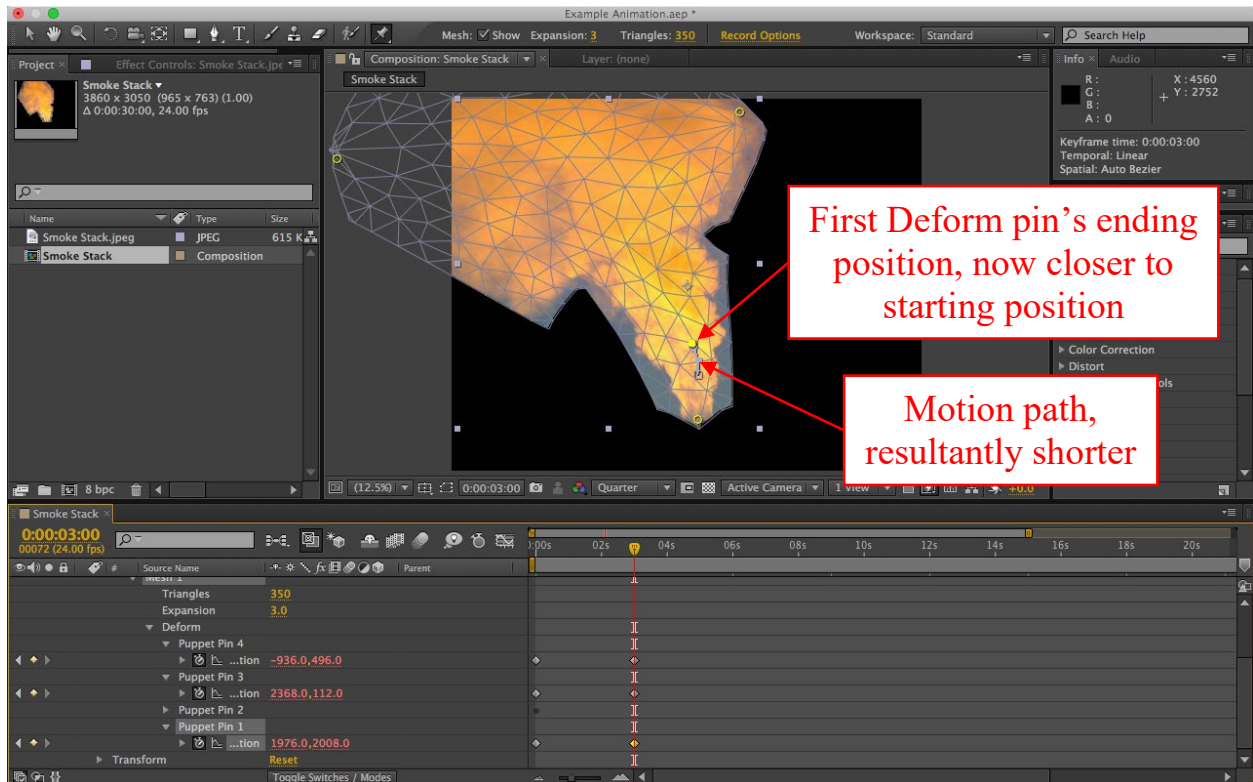
8. **Claim 6: The computer system of claim 1, wherein the magnitude of the shifting of the first set of pixels is proportionally related to the first length and the magnitude of the shifting of the second set of pixels is proportionally related to the second length.**

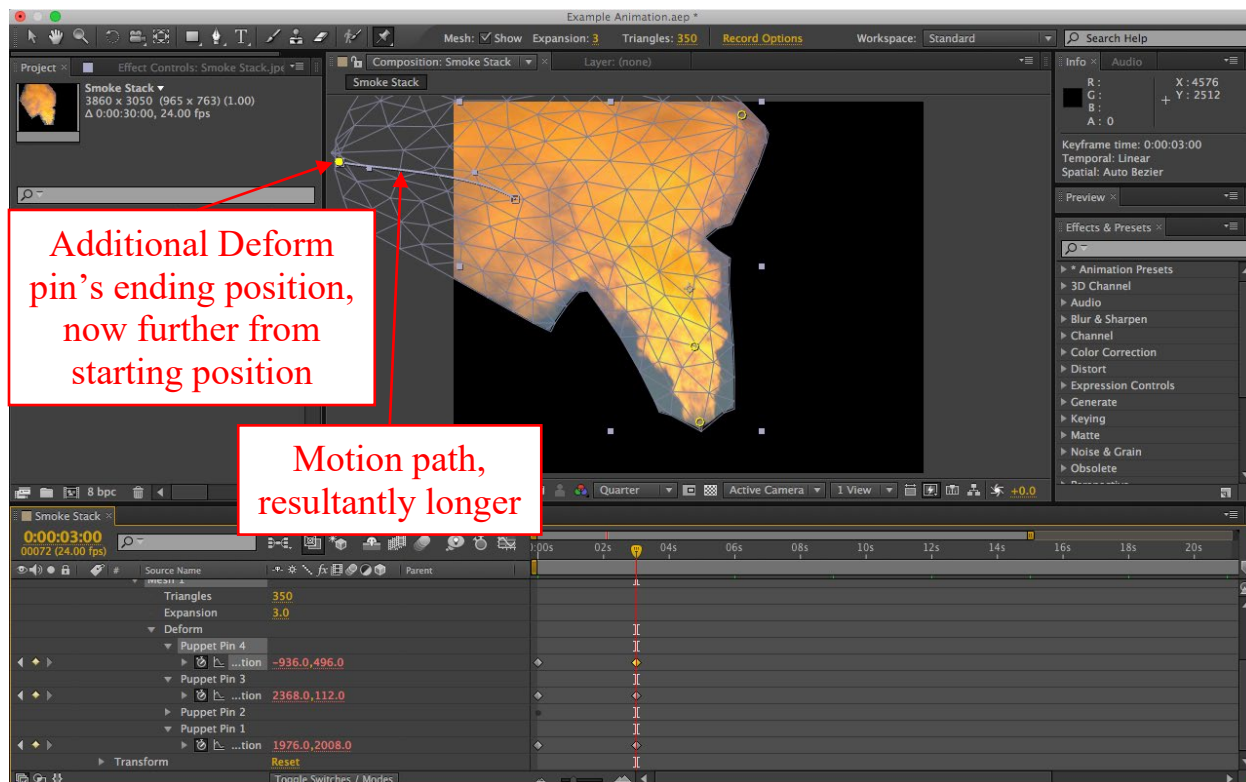
117. Claim 6 is dependent from claim 1. In my opinion, AEM discloses claim 6. Specifically, AEM teaches, after a user defines a starting keyframe for a Deform pin and a mesh of the layer's nontransparent pixels is automatically created, creating an ending keyframe by "[g]o[ing] to another time in the composition, and mov[ing] the position of... the Deform pin[] by dragging [it] in the Composition or Layer panel." AEM, 218-19; Sections VIII.B.3.d-VIII.B.3.e. This repositioning

causes the shape of the layer's mesh to change—and thus the layer's nontransparent pixels to move—to “accommodate” this repositioning. AEM, 218-19; Section VIII.B.3.e. Further, AECS6 performs keyframe interpolation to animate this movement of the layer's nontransparent pixels, which is performed according to, and visually indicated by, a motion path extending from the starting point to the ending point. AEM, 187, 218-19; Section VIII.B.3.e. As such, for a given span of time in the composition, if a Deform pin's ending position/keyframe is repositioned a long distance away from its starting position/keyframe, the corresponding change to the shape of the layer's mesh—and thus the movement of the layer's nontransparent pixels—will be of a large magnitude to “accommodate” such a large repositioning, and the Deform pin's corresponding motion path will be of a longer length. *See* AEM, 218-19. The opposite occurs if the Deform pin repositioned only a short distance. *See* AEM, 218-19. Such illustrates a proportional relationship between the magnitude of pixel movement and the length of the motion path for each corresponding Deform pin—i.e., the claimed “magnitude of the shifting of the first set of pixels is proportionally related to the first length and the magnitude of the shifting of the second set of pixels is proportionally related to the second length” as recited in claim 6.

118. By way of further illustration, the exemplary composition from Attachment C's table (discussed in Section VIII.B.3.h) is depicted in the screenshots

below, but now the first Deform pin is repositioned a lesser amount (shown in the first screenshot) while one additional Deform pin is repositioned a larger amount (shown in the second screenshot) for the same span of time:





As shown, the first Deform pin's motion path is now shorter, and the surrounding pixels have a proportionally lower magnitude of movement, whereas the opposite is true of the additional Deform pin.

9. Claim 7

a. [7a]: The computer system of claim 1, wherein:

the first set of pixels comprise pixels that intersect the first rendered link; and.

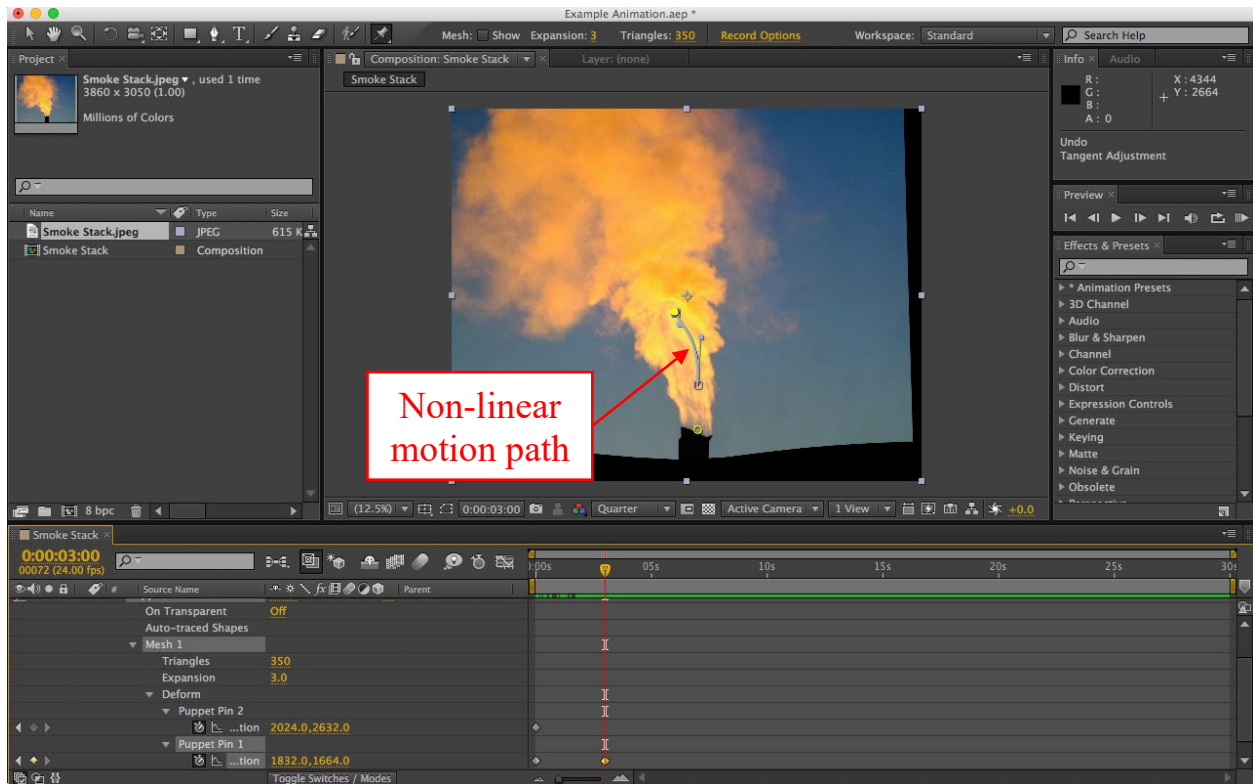
119. Claim 7 is dependent from claim 1. As similarly discussed in Section VIII.B.3.f, AEM teaches that AECS6's user applies a mask or matte to select any or all pixels of the layer to be nontransparent and thus included in the layer's mesh to be moved and animated, including pixels that intersect the first Deform pin's motion

path—i.e., the claimed “first set of pixels comprise pixels that intersect the first rendered link.” AEM, 218-19, 315, 194-95.

120. Separately, for the same reasons discussed in Section VIII.B.3.f, a POSITA would have been motivated to select and make nontransparent, using a mask or matte, not only pixels parallel to the Deform pin’s motion path, within a user-defined threshold distance from the motion path, and extending from the Deform pin’s starting position/keyframe to the ending position/keyframe, but also pixels intersecting the motion path. Indeed, each mask/matte and mesh depicted in the screenshots in Section VIII.B.3.f covers pixels intersecting the first Deform pin’s motion path. Thus, in my opinion, AEM meets limitation [7a].

b. [7b]: The first rendered link comprises a non-linear pathway extending between the first starting point and the first ending point.

121. AEM teaches not only creating a motion path extending between a Deform pin’s starting position/keyframe and ending position/keyframe as discussed in Section VIII.B.3.e, but also making such a motion path non-linear using, e.g., “Bezier interpolation”—i.e., the claimed “first rendered link comprises a non-linear pathway extending between the first starting point and the first ending point.” AEM, 193-95. In fact, the motion path depicted in Section VIII.B.3.e’s screenshot, further annotated below, is non-linear:



AEM therefore, in my opinion, discloses limitation [7b].

10. Independent Claim 8

- a. **[8pre]: A method for automating the shifting of pixels within a digital image, comprising.**

122. In my opinion, AEM teaches how to animate the movement of pixels within a still image using AECS6—i.e., the claimed “method for automating a shifting of pixels within a digital image.” Sections VIII.B.10.b-VIII.B.10.g.

- b. **[8a]-[8b]: receiving a first indication of a first starting point through a user interface, wherein the first starting point is received through a user selection of a first starting portion of the digital image;**

render a first visual indication of the first starting point on the user interface.

123. *See* Section VIII.B.3.c.

- c. **[8c]-[8d]: receiving, through the user interface, a first ending point associated with the first starting point, wherein the first ending point is received through a user selection of a first ending portion of the digital image;**

render a second visual indication of the first ending point on the user interface.

124. *See* Section VIII.B.3.d. Additionally, the Deform pin's ending position/keyframe is, as claimed, "associated with" its starting position/keyframe via a motion path for the Deform pin, which extends between these two positions. AEM, 187, 218-19; Section VIII.B.3.e.

- d. **[8e]: render, on the user interface, a first rendered link between the first starting point and the first ending point.**

125. *See* Section VIII.B.3.e.

- e. **[8f]: selecting a first set of pixels that intersect with the first rendered link.**

126. *See* Section VIII.B.9.a.

- f. [8g]: automatically shifting the first set of pixels along the first rendered link between the first starting point and the first ending point, wherein shifting the first set of pixels comprises rendering in a loop the first set of pixels being shifted.**

127. *See* Sections VIII.B.3.g, VIII.B.9.a.

- g. [8h]-[8n]: receiving a second indication of a second starting point through the user interface, wherein the second starting point is received through a user selection of a second starting portion of the digital image, the second starting portion being different than both the first starting portion and the first ending portion;**

render a third visual indication of the second starting point on the user interface;

receiving, through the user interface, a second ending point associated with the second starting point, wherein the second ending point is received through a user selection of a second ending portion of the digital image, the second ending portion being different than both the first starting portion and the first ending portion;

render a fourth visual indication of the second end point on the user interface;

render, on the user interface, a second rendered link between the second starting point and the second ending point;

selecting a second set of pixels that intersect with the second rendered link;

automatically shifting the second set of pixels along the second rendered link between the second starting point and the second ending point, wherein shifting the second set of pixels comprises rendering in a loop the second set of pixels being shifted.

128. *See* Sections VIII.B.3.h, VIII.B.9.a.

11. Claim 9: **The method as recited in claim 8, further comprising when a pixel selected from the first set of pixels reaches the ending point, rendering and re-rendering in the loop the pixel being shifted from the first starting point to the ending point.**

129. Claim 9 is dependent from claim 8. AEM teaches, and a POSITA would have been motivated to perform, applying a mask or matte in AECS6 to select any or all pixels of a layer to be nontransparent and thereby included in the layer's mesh to be moved and animated, including pixels intersecting the Deform pin's motion path and away from the starting keyframe—i.e., the claimed “first set of pixels.” AEM, 218-19, 315, 194-95; Section VIII.B.10.e. The user creates the starting keyframe—i.e., “first starting point”—by specifying a time point and placing a Deform pin on a “nontransparent pixel” of the layer—i.e., “a pixel selected from the first set of pixels.” AEM, 218-19; Section VIII.B.10.b. The user then repositions the Deform pin to create the ending keyframe—i.e., “ending point.” AEM, 218-19, 177; Section VIII.B.10.c. AEM explains that AECS6 then performs animation by interpolating the position of the Deform pin from the starting to ending keyframe along the user-specified motion path and moving the layer's mesh and thus the layer's nontransparent pixels correspondingly. AEM, 218-19, 194-95; Section VIII.B.10.f. Further, the user applies a “loopOut()” expression to loop and repeat such interpolation and animation—i.e., “when a pixel selected from the first set of pixels reaches the ending point, rendering and re-rendering in the loop the pixel

being shifted from the first starting point to the ending point.” AEM, 219, 562;
Section VIII.B.10.f. Therefore, in my opinion, AEM discloses claim 9.

12. Claim 10

- a. **[10a]-[10b]: The method as recited in claim 8, further comprising receiving an indication to generate a mask over a portion of the digital image, wherein pixels under the mask are prevented from shifting.**

130. *See* Section VIII.B.4.a.

13. Claim 11: **The method as recited in claim 10, further comprising receiving through a user interface a selection of a particular pixel within the digital image from which the mask should be generated.**

131. *See* Section VIII.B.5.

14. Claim 12

- a. **[12a]-[12b]: The method of claim 11, further comprising:**

identifying one or more edges that form a boundary around the particular pixel; and

generating the mask to cover area within the boundary.

132. *See* Section VIII.B.6.a.

15. Claim 13

- a. **[13a]: The method of claim 8, wherein: the first set of pixels further comprise:**

pixels that are parallel to the first rendered link,

within a user-defined threshold distance from the first rendered link, and

that extend from the first starting point to the first ending point; and

133. *See* Section VIII.B.3.f.

- b. **[13b]: the first rendered link comprises a non-linear pathway extending between the first starting point and the first ending point.**

134. *See* Section VIII.B.9.b.

16. Independent Claim 14

- a. **[14pre]: A method for automating the shifting of pixels within a digital image, comprising.**

135. In my opinion, AEM teaches how to animate the movement of pixels within an image using AECS6—i.e., the claimed “method for automating a shifting of pixels within a digital image.” Sections VIII.B.16.b-VIII.B.16.h.

- b. **[14a]: accessing, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within the digital image.**

136. *See* Section VIII.B.3.b.

- c. **[14b]-[14c]: receiving a first starting point through a user interface, wherein the first starting point is received through a user selection of a first beginning portion of the digital image;**

rendering a first visual indication of the first starting point on the user interface.

137. *See* Section VIII.B.3.c.

- d. **[14d]-[14e]: receiving a first ending point through the user interface, wherein the first ending point is received through a user selection of a first ending portion of the digital image;**

rendering a second visual indication of the first ending point on the user interface.

138. *See* Section VIII.B.3.d.

- e. **[14f]: rendering, on the user interface, a first rendered link between the first starting point and the first ending point, wherein the first link comprises:**

a non-linear pathway extending between the first starting point and the first ending point; and

a first length extending along the non-linear pathway.

139. *See* Sections VIII.B.3.e, VIII.B.9.b.

- f. **[14g]: identifying a first set of pixels that lie along the first rendered link.**

140. *See* Sections VIII.B.3.f, VIII.B.9.a-VIII.B.9.b.

- g. [14h]: automatically shifting the first set of pixels along the non-linear pathway, wherein shifting the first set of pixels comprises rendering and re-rendering in a loop the first set of pixels being shifted.**

141. *See* Sections VIII.B.3.g, VIII.B.9.a-VIII.B.9.b.

- h. [14i]-[14o]: receiving a second starting point through the user interface, wherein the second starting point is received through a user selection of a second starting portion of the digital image, the second starting portion being different than both the first starting portion and the first ending portion;**

rendering a third visual indication of the second starting point on the user interface;

receiving, through the user interface, a second ending point associated with the second starting point, wherein the second ending point is received through a user selection of a second ending portion of the digital image, the second ending portion being different than both the first starting portion and the first ending portion;

rendering a fourth visual indication of the second end point on the user interface;

rendering, on the user interface, a second link between the second starting point and the second ending point;

selecting a second set of pixels along the second link; and

automatically shifting the second set of pixels between the second starting point and the second ending point, wherein shifting the second set of pixels comprises rendering in a loop the second set of pixels being shifted.

142. *See* Sections VIII.B.3.h, VIII.B.9.a-VIII.B.9.b.

17. Claim 15

- a. **[15a]-[15b]: The method of claim 14, further comprising identifying a particular portion of the digital image to mask, wherein the mask prevents pixels covered by the mask from being shifted.**

143. *See* Section VIII.B.4.a.

18. Claim 16: **The method of claim 15, wherein receiving an indication of a particular portion of the digital image to mask comprises receiving through a user interface a selection of a particular pixel within the digital image.**

144. *See* Section VIII.B.5.

19. Claim 17

- a. **[17a]-[17c]: The method of claim 15, further comprising:**

generating the mask, wherein generating the mask comprises:

identifying one or more edges that form a boundary around the particular pixel; and

generating the mask to cover area within the boundary.

145. *See* Section VIII.B.6.a.

20. Claim 18: **The method of claim 15, wherein:**

the first set of pixels further comprise:

pixels that are parallel to the first rendered link,

within a user-defined threshold distance from the first rendered link, and that extend from the first starting point to the first ending point.

146. See Sections VIII.B.9.a-VIII.B.9.b, VIII.B.3.f.

C. Ground 2: IMU and Okabe, and Claims 1-3, 5-11, 13-16, and 18

1. Summary of IMU

147. IMU contains Wayback Machine captures from 2012 of the ImageMagick.com website, specifically the website's section titled "Examples of ImageMagick Usage (Version 6)," which provides guidance and examples on how to use IMV6's various effects and capabilities. IMU-Home, 1-2.

148. IMU teaches that IMV6 is operated in the command-line. IMU-Home,

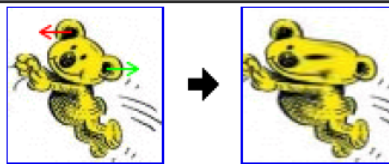
2. As such, IMV6 "*is not a GUI image editor.*" IMU-Home, 2.

149. Further, IMU teaches how to manually write text commands in IMV6 to apply effects to an image, such as IMV6's "Shepard's Distortion" pixel shifting effect, as well as how to create infinitely looping animations from such effects. IMU also teaches how to manually write commands to modify the transparency of an image's pixels—and thus the visibility of effects that are applied to the image—using a "matte."

a. Applying Effects and Creating an Animation

150. IMU teaches that IMV6 is an “image-to-image converter” that utilizes a “library of Image Processing Algorithms” to apply effects and convert one image into another. IMU-Home, 2. One such effect is “Shepard’s Distortion,” a distortion effect applied by placing “control points” on pixels located at user-specified coordinates of an image and moving such control points to new coordinates “to distort the image in terms of ‘local’ effects.” IMU-Distorting, 59, 19-20. IMU provides an example of placing and moving Shepard’s Distortion control points on an image of a koala to “torture the ‘koala’ by pulling on his ears” from coordinates (30,11) and (48,29) to coordinates (20,11) and (59,29), respectively. IMU-Distorting, 59. The example’s command and koala image before and after the “torture,” is reproduced below, with original annotations from IMU indicating the corresponding placement and movement of the two control points:

```
convert koala.gif -virtual-pixel Black \  
-distort Shepards '30,11 20,11 48,29 58,29' \  
koala_ear_pull.png
```

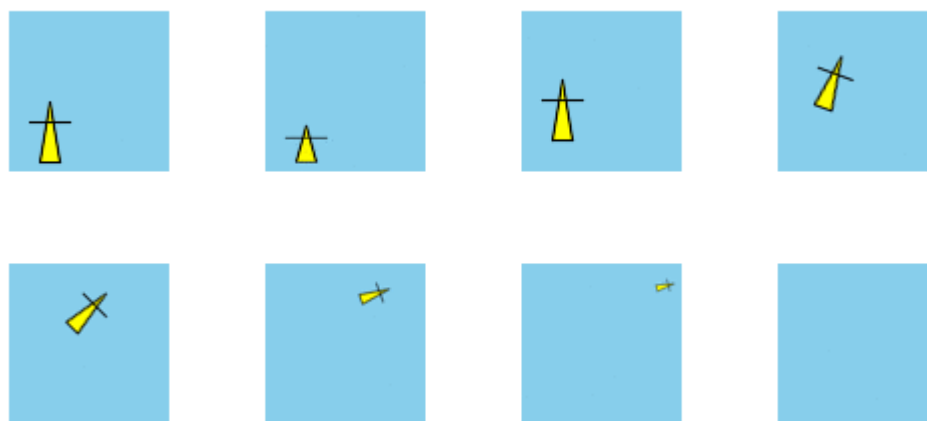


IMU-Distorting, 59.

151. Although IMV6 is an “image-to-image converter,” IMU further teaches how to incrementally apply IMV6’s effects to an image to produce a sequence of

frames that, when played sequentially, resulting in an animation of the original image. See IMU-Distorting, 17-18. IMU provides an example of applying IMV6's "SRT Distortion" effect—another "[d]istortion" effect that, like Shepard's Distortion, uses "control points"—to an image of a "stylized space ship" by placing and moving an SRT Distortion control point in seven increments to create a sequence of eight frames that, when played sequentially, animates the launch of the space ship along a curved path through the sky. IMU-Distorting, 17-18. The example's command and the eight resulting frames in their sequential order are reproduced below:

```
convert -size 80x80 xc:skyblue -fill yellow -stroke black \
  -draw 'path "M 15,75 20,45 25,75 Z M 10,55 30,55" ' \
  spaceship.gif
convert spaceship.gif \
  \( -clone 0 -distort SRT '20,75 1.0,0.6 0' \) \
  \( -clone 0 -distort SRT '20,60 1 0 20,49' \) \
  \( -clone 0 -distort SRT '20,60 0.9 20 27,35' \) \
  \( -clone 0 -distort SRT '20,60 0.8 45 40,23' \) \
  \( -clone 0 -distort SRT '20,60 0.5 70 55,15' \) \
  \( -clone 0 -distort SRT '20,60 0.3 75 72,11' \) \
  \( -clone 0 -distort SRT '20,60 0.1 80 100,8' \) \
  -set delay 50 -loop 0 spaceship_launch.gif
```



IMU-Distorting, 17-18.¹³

152. After creating a sequence of frames, IMU teaches how to save these frames in an animated GIF file that will be rendered as an animation by a variety of standard computer programs, including Web browsers, that show each frame in sequence. IMU-Animating, 1-2; IMU-Distorting, 17-18. For creating such an animated GIF, IMU teaches applying a “-loop” operator to specify the “[n]umber of

¹³ The full animation can be viewed on the IMU-Distorting webpage as archived at <https://web.archive.org/web/20120329131929/http://www.imagemagick.org/Usage/distorts/> (end of the section titled “Scale-Rotate-Translate (SRT) Distortion” and just above a headline “Methods of Rotating Images”). The above sequence of eight frames was created by applying IMV6’s “-coalesce” operator to the animation (a GIF file, accessible at the preceding URL) in a similar manner as that taught in IM-Animation, 8-9.

times the GIF animation is to cycle through the image sequence before stopping.” IMU-Animating, 2. When the “-loop” operator’s value is set to its default value of zero, such as in the above exemplary command, the animated GIF will render the animation in an “infinite loop.” IMU-Animating, 2; IMU-Distorting, 17-18.

b. Modifying Transparency of Certain Pixels Using a Matte

153. IMU also teaches modifying transparency of an image’s pixels—and thus the visibility of effects that are applied to the pixels—by modifying the image’s “transparency (alpha) channel,” also known as a “matte” channel” or simply a “matte.” IMU-Masking, 2-3. This matte “is just a plain grey scale image of values which range from white, for full-transparent (or clear), to black for fully-opaque.” IMU-Masking, 2.

154. IMU provides an example matte shaped like a crescent moon:



IMU-Masking, 2. IMU further illustrates how this matte is applied or not applied to to correspondingly change the transparency of an image’s pixels:



IMU-Masking, 3-4.

2. Summary of Okabe

155. Okabe describes a “method for synthesizing fluid animation from a single image.” Okabe, 1. While acknowledging that “[m]any methods have been proposed for creating an animation from a single image,” Okabe asserts that its method of creating such an animation provides, among other advantages, “markedly reduce[d]... user burden.” Okabe, 2.

156. Okabe teaches that, to animate an image, the user first “inputs a target painting or photograph of a fluid scene along with its alpha matte that extracts the fluid region of interest in the scene.” Okabe, 1. Thereafter, the user “sketch[es] the flow direction and paint[s] a speed map” by drawing “a sparse set of user-drawn strokes” on the image, “which is a simple task and takes less than 1 min[ute].” Okabe, 3, 7. Okabe’s Figure 1 provides an example and shows that the “strokes”—“shown as orange arrows” in Okabe’s interface—may be non-linear:

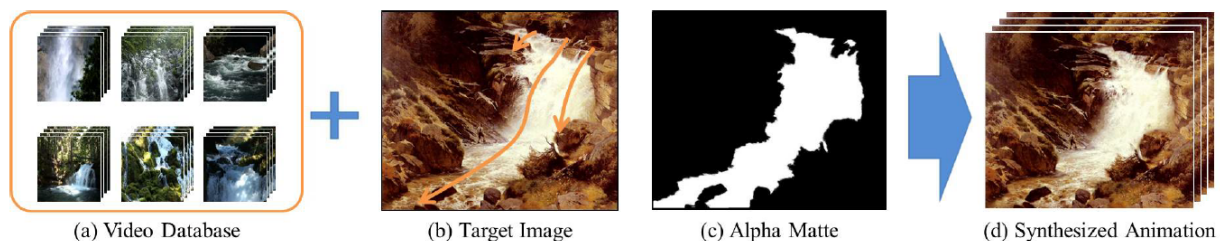


Figure 1: *We employ a database of video examples of fluids (a). The user specifies a target image (b) with a few optional suggestions about fluid motion, e.g., sketches of flow direction, shown as orange arrows. The user also provides an alpha matte of the region of interest (c). The system synthesizes an animation (d).*

Okabe, 2. From these inputs, Okabe’s algorithm is then generates flow animation in the target image corresponding to the flow direction and speed specified by the user’s strokes, and permits “infinite repetition” of such an animation. Okabe, 1, 8.

According to Okabe, such simple and graphical commands allow images to be animated “with less effort than with previous methods.” Okabe, 1.

3. The IMU-Okabe Combination

a. **Motivation to Combine IMU with Okabe**

157. As explained below, a POSITA, in my opinion, would have been motivated to combine IMV6’s animation capabilities described in IMU with the simple and graphical user commands for creating an animation taught by Okabe (hereinafter, the “IMU-Okabe Combination”).

158. According to IMU, IMV6 is an “image-to-image converter” operated in the command-line and therefore “*not a GUI image editor.*” IMU-Home, 2. In general, command-line interfaces, such as that used in the MS-DOS operating system, have grown less and less popular in favor of graphical user interfaces, such as that used in MS-DOS’s successor—Microsoft Windows. A POSITA understood that, when releasing a computer software program or system, a graphical user interface would almost always be more commercially successful and preferred by users than a command-line interface.

159. In IMV6 in particular, to create an animation from an image using IMV6, a user must create each frame of the animation by manually writing DOS-style text commands that apply IMV6’s effects to each frame. *E.g.*, IMU-Distorting, 17-18.

160. Given this, a POSITA would have recognized that, to animate an image in IMV6 using, e.g., Shepard’s Distortion, a user would write commands that place and incrementally move Shepard’s Distortion control points within the image to new coordinates for each successive frame, such that each control point moves across the animation at the user’s desired—and potentially non-linear—direction and speed. *See* IMU-Distorting, 17-18 (creating an animation in the same way, but using IMV6’s “SRT distortion” effect, to animate the launch of a space ship along a non-linear path), 59, 19-20.

161. Further, to make distortion and animation visible only in a particular region of the image, the user would write further commands that apply a corresponding matte to the image such that only the particular region is visible during the distortion and animation. IMU-Masking, 2. Lastly, to save the animation as a looping animated GIF, the user would write further commands to save the animation using IMV6’s “-loop” operator. IMU-Animating, 1-2; *see also, e.g.,* IMU-Distorting, 17-18.

162. But given this laborious, non-graphical, text-based process for creating an animation in IMV6, a POSITA would have been motivated to modify IMV6 by enabling a user to animate an image using simpler and more intuitive graphical user commands. In other words, a POSITA would have been motivated to modify IMV6 such that it *is* a “GUI image editor.” *See* IMU-Home, 2.

163. The motivation is underscored in IMU itself, which acknowledges the disadvantages of IMV6’s lack of a GUI when applying effects to a single image, noting “there are a lot of readily available image manipulation programs, such as Adobe Photoshop, Corel’s Paint Shop Pro, IrfanView (<http://www.irfanview.com/>) and even GIMP (<http://www.gimp.org/>). So why should you bother to perform image processing by IM[V6]’s command line programs and scripts?” IMU-Windows, 1. Indeed, AECS6 was another such “readily available image manipulation program[]” that combined the ability to create a looping animation of effects applied to images (e.g., Puppet effect) with simple and graphical user commands for defining such animation (e.g., placing Deform pins, creating motion paths between keyframes for the Deform pins, and specifying which pixels move using a mask or matte, all within the “Composition” or “Layer panel[s]”), confirming a POSITA would have been—and in fact was—motivated to make such a combination of features. *See* AEM, 218-19, 187-89, 193-95, 317, 562; Section VIII.B.1.

164. Okabe teaches simple and graphical user commands that a POSITA would have found well-suited for implementing a GUI to create animations in IMV6. Specifically, Okabe teaches that a user animates an image by first using an “alpha matte” to specify a “region of interest” the user wants to animate. Okabe, 1. The user also draws on the image a “sparse set of user-drawn strokes,” which may be

non-linear, to specify the user's desired direction and speed of animation, "which is a simple task and takes less than 1 min[ute]." Okabe, 7, 2, 3. From just these inputs, Okabe describes automatically producing an infinitely repeating animation that follows the user-specified direction and speed. Okabe, 1-2, 8. As such, Okabe's simple and graphical user commands enables a user to create looping animations "with less effort than with previous methods" and "markedly reduces the user burden." Okabe, 1-2.

165. Given such advantages, a POSITA would have recognized that Okabe's simple and graphical user commands would have been desirable for animating an image in IMV6, i.e., obviating IMV6's laborious and non-graphical animation procedure. As such, a POSITA would have been motivated to modify IMV6 to include Okabe's simple and graphical user commands to make the animation process easier, graphical, and less time consuming.

b. Resulting Combination of IMU with Okabe

166. So modified, rather than require a user to write complex commands that place and move Shepard's Distortion control points on an image frame-by-frame such that each control point moves across the animation at the user's desired—and potentially non-linear—direction and speed (*see* IMU-Distorting, 17-18, 59, 19-20), the IMU-Okabe Combination enables a user to achieve the same result by drawing

non-linear strokes viewable by the user¹⁴ on the image to specify the direction and speed of corresponding Shepard's Distortion control points across the animation (*see* Okabe, 7, 2-3).

167. Also, similar to how IMV6 allows a user to apply a matte to make distortion and animation effects visible in only a portion of an image (IMU-Masking, 2), the IMU-Okabe Combination's user applies a matte to make the Shepard's Distortion effect from the user-drawn strokes visible in only a portion of the image (*see* Okabe, 1). Further, as IMV6 allows an animation to be saved as an infinitely looping animated GIF using the "-loop" operator (IMU-Animating, 1-2), the IMU-Okabe Combination likewise permits infinite repetition of the resulting animation (*see* Okabe, 1, 8).

¹⁴ Should Plotagraph contend Okabe's strokes are not disclosed as user-viewable, a POSITA would have nevertheless been motivated to make Okabe's strokes user-viewable to aid users in visualizing the direction and speed of their animations, thereby making the animation process simpler, graphical, and more intuitive. *See, e.g.,* AEM, 187 ("Motion paths are simply an alternative visual, spatial way of viewing and working with spatial properties and their keyframes, in addition to the ways that you work with properties in the Timeline panel.").

4. Independent Claim 1

- a. **[1pre]: A computer system for automating the shifting of pixels within a digital image, comprising:**

one or more processors; and

one or more computer-readable media having stored thereon executable instructions that when executed by the one or more processors configure the computer system to perform at least the following.

168. IMU teaches that IMV6 is operated in the command-line and installed and run on a computer. IMU-Home, 2; IMU-Windows, 3-4 (“By default, IM[V6] installs itself in a program directory....”). Okabe teaches that, when animating an image using Okabe’s simple and graphical user commands, “the designer specifies a single target image along with several characteristics regarding motion and *uses a computer* to synthesize animated sequences derived from the input.” Okabe, 1 (emphasis added). Given both IMV6 and Okabe’s commands require the use of a computer, the IMU-Okabe Combination would thus also be installed and run on a computer, which would include a processor—i.e., the claimed “one or more processors”—and storage storing instructions for running the IMU-Okabe Combination—i.e., “one or more computer-readable media having stored thereon executable instructions that when executed by the one or more processors configure the computer system to perform at least the following.” Section VIII.C.3.b.

169. Further, the IMU-Okabe Combination enables a user to automatically animate an image—i.e., “automating the shifting of pixels within a digital image.” Sections VIII.C.4.b-VIII.C.4.h.

- b. **[1a]: access, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within the digital image.**

170. According to IMU, IMV6 is an “image-to-image converter” that enables a user to apply effects to animate an image—i.e., the “digital image file, wherein the digital image file comprises information that corresponds to individual pixels within the digital image.” IMU-Home, 2; IMU-Distorting, 17-18. The image accessed and animated by IMV6 can be “an IM[V6] built-in image (such as ‘logo:’, or ‘rose:’)” or an “external image” from an external source “placed in the ‘images’ and ‘img_photos’ sub-directories”—i.e., “access, from memory, a digital image file.” IMU-Home, 3-4.

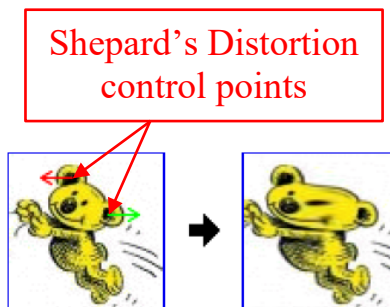
171. Further, Okabe teaches that, to utilize its simple and graphical user commands, “[t]he user *inputs a target image* and an alpha matte that extracts the fluid region of interest, while providing a few optional suggestions about fluid motion (i.e., flow direction and speed),” which likewise meets the claimed “access, from memory, a digital image file.” Okabe, 1-2 (emphasis added). The IMU-Okabe Combination would thus enable a user to input a digital image from a memory, meeting limitation [1a]. Section VIII.C.3.b.

- c. **[1b]-[1c]: receive a first starting point through a user interface, wherein the first starting point is received through a user selection of a first beginning portion of the digital image;**

render a first visual indication of the first starting point on the user interface.

172. IMU teaches that IMV6 is an “image-to-image converter” that enables a user to apply effects, such as distortion effects, to convert one image—i.e., the claimed “digital image”—into a sequence of frames that animate the original image when played sequentially. IMU-Home, 2; IMU-Distorting, 17-18. Distortion effects include “Shepard’s Distortion.” IMU-Distorting, 59. To animate with distortion effects, the user places a “control point” on a pixel located at a user-specified coordinate of the image and then moves the control point to a new coordinate. IMU-Distorting, 19-20; *see also, e.g.*, IMU-Distorting, 17-18 (placing and moving an “SRT distortion” control point on an image of a space ship to animate the launching of the space ship across the sky). This is done incrementally for each successive frame such that the control point moves across the animation at the user’s desired direction and speed. *See* IMU-Distorting, 19-20; *see also, e.g.*, IMU-Distorting, 17-18. IMU provides an example of one such increment, where two Shepard’s Distortion control points are placed on an image of a koala at coordinates (30, 11) and (48, 29), before being moved to coordinates (20,11) and (59,29), respectively.

See IMU-Distorting, 59. IMU also provides a figure showing the koala image before and after the Shepard's Distortion is applied:



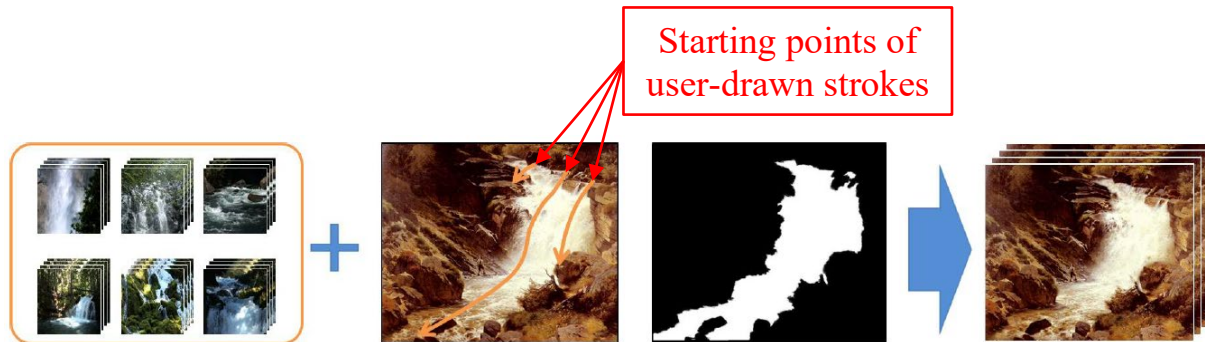
IMU-Distorting, 59 (additional annotations added).

173. Further, in the IMU-Okabe Combination, IMU's placement and movement of a Shepard's Distortion control point corresponds to Okabe's user-drawn, user-viewable "stroke[]" on an image. See Okabe, 7, 3; Section VIII.C.3.b. That is, Okabe's user-drawn stroke includes a user-viewable indication of a starting point, direction, and speed across the animation, much like how a user in IMV6 indicates a control point's starting point, direction, and speed across an animation according to IMU. See Okabe, 7, 3; IMU-Distorting, 17-18, 59, 19-20; Section VIII.C.3.b.

174. The starting point of a user-drawn, user-viewable stroke corresponds to the placement of a Shepard's Distortion control point on a specific coordinate of the image—i.e., "wherein the first starting point is received through a user selection of a portion of the digital image" and "render a first visual indication of the first starting point on the user interface." See IMU-Distorting, 59, 19-20; Okabe, 7, 3. Receiving

the stroke's starting point from the user meets "receiving a first indication of a first starting point through a user interface." *See* Okabe, 7, 3.

175. Okabe's Figure 1 illustrates the relevant disclosure by depicting several user-drawn strokes (shown in orange) on an image of a waterfall:

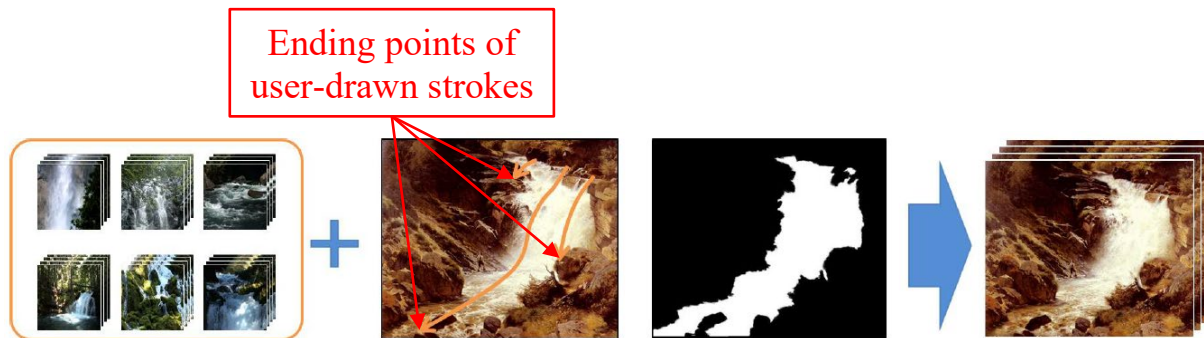


Okabe, 2 (Figure 1, annotations added). In my opinion, the IMU-Okabe Combination therefore meets limitation [7a].

- d. **[1d]-[1e]: receive a first ending point through the user interface, wherein the first ending point is received through a user selection of a first ending portion of the digital image;**
render a second visual indication of the first ending point on the user interface.

176. In my opinion, the IMU-Okabe Combination meets limitations [1d]-[1e]. In the IMU-Okabe Combination, a user defines a Shepard's Distortion control point's direction and speed across the animation of an image by drawing a user-viewable "stroke[]" on the image. *See* Okabe, 7, 2-3; IMU-Distorting, 59; Section VIII.C.3.b. Receiving a user-drawn, user-viewable stroke, specifically the stroke's ending point, corresponds to the claimed "receive a first ending point through the

user interface, wherein the first ending point is received through a user selection of a first ending portion of the digital image” and “render a second visual indication of the first ending point on the user interface.” *See* Okabe, 2 (Figure 1, reproduced below with annotations added).

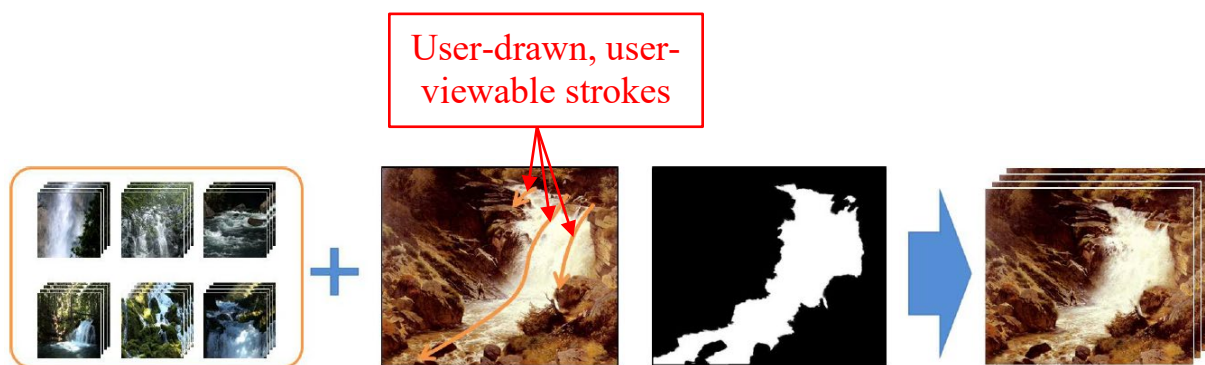


- e. **[1f]: render, on the user interface, a first rendered link between the first starting point and the first ending point, wherein the first rendered link comprises:**
 - a first direction extending from the first starting point to the first ending point; and**
 - a first length between the first starting point and the first ending point.**

177. IMU teaches animating in IMV6 using distortions (e.g., Shepard’s Distortion) by placing and incrementally moving control points frame-by-frame such that each control point moves across the animation at the user’s desired direction and speed. *See* IMU-Distorting, 17-18, 59, 19-20.

178. Further, in the IMU-Okabe Combination, a user specifies such a direction and speed of a Shepard’s Distortion control point by drawing a corresponding user-viewable “stroke[]” from Okabe on the image—i.e., the claimed

“render, on the user interface, a first rendered link between the first starting point and the first ending point, wherein the first rendered link comprises: a first direction extending from the first starting point to the first ending point; and a first length between the first starting point and the first ending point.” *See* Okabe, 7, 2-3; Section VIII.C.3.b. Indeed, Okabe’s Figure 1 illustrates three user-viewable strokes received from the user:



Okabe, 2 (Figure 1, annotations added). Similarly, Okabe’s Figure 8-a depicts two user-drawn, user-viewable strokes, each shown in Okabe’s interface as a green curved line with an arrow specifying direction:



(a) Target Image

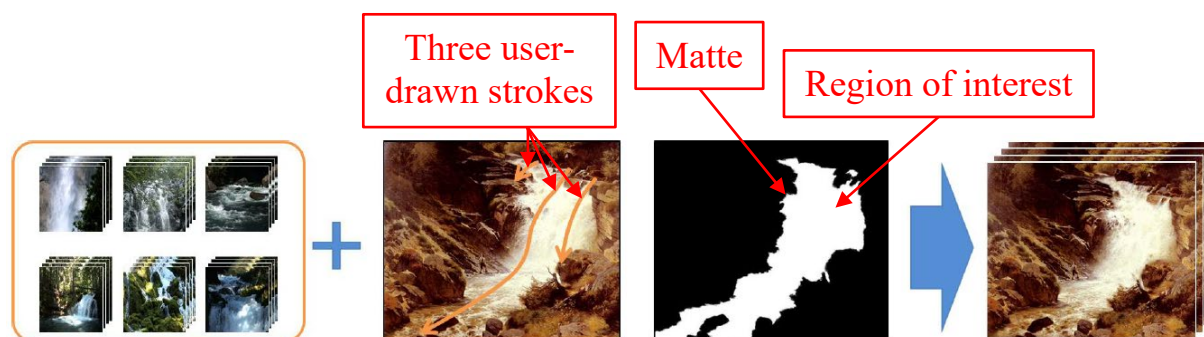
Okabe, 6 (discussing Figure 8-a). Therefore, in my opinion, the IMU-Okabe Combination meets limitation [1f].

- f. [1g]: identify a first set of pixels that are:**
parallel to the first rendered link,
within a user-defined threshold distance from the first
rendered link, and
extending from the first starting point to the first
ending point.

179. IMU teaches modifying the transparency of an image's pixels in IMV6—and thus the visibility of effects that are applied to the image—by applying a “matte,” which “is just a plain grey scale image of values which range from white, for full-transparent (or clear), to black for fully-opaque.” IMU-Masking, 2.

180. Further, the IMU-Okabe Combination's user similarly applies a matte specifying a region of interest in the image the user desires to animate (*see* Okabe, 1), in addition to a “stroke[]” that specifies the direction and speed of a Shepard's Distortion control point across the animation (*see* Okabe, 7, 3). Section VIII.C.3.b. Thus, the IMU-Okabe Combination would have permitted selecting any or all pixels of the image to be included in this region of interest, including pixels parallel to the stroke, within a user-defined threshold distance from the stroke, and extending from the starting point to the ending point of the stroke—i.e., the claimed “identify a first set of pixels that are: parallel to the first rendered link, within a user-defined threshold distance from the first rendered link, and extending from the first starting point to the first ending point”—by using a matte. *See* Okabe, 1. Okabe's Figure 1

confirms this by depicting a matte specifying a region of interest (i.e., a to-be-animated region) in white that includes pixels parallel to each of three user-drawn strokes, within a user-defined threshold distance from the strokes (i.e., to the waterfall's edges), and extending from the starting point to the ending point of each stroke:



Okabe, 2 (Figure 1, annotations added); *accord* EX1022, 10-11. This confirms that the IMU-Okabe Combination meets limitation [1g].

- g. **[1h]: automatically shift the first set of pixels in the first direction, wherein shifting the first set of pixels comprises rendering and re-rendering in a loop the first set of pixels being shifted.**

181. With respect to “shifting the first set of pixels along the non-linear pathway,” the ’017 Patent’s specification states the claimed “shifting” can be performed by using, for example, “a warping function, such as Shepard’s distortion.” ’017 Patent, 8:32-34. IMU explicitly teaches that IMV6 performs “Shepard’s Distortion” and therefore the claimed pixel shifting. IMU-Distorting, 59. That is, IMV6’s user shifts pixels as claimed by using Shepard’s Distortion to place and

incrementally move a “control point” on an image frame-by-frame in the user’s specified direction and speed. *See* IMU-Distorting, 17-18, 59, 19-20.

182. Further, the IMU-Okabe Combination’s user specifies a control point’s direction and speed by drawing a corresponding “stroke[]” from Okabe on the image. *See* Okabe, 7, 3, 2; Sections VIII.C.3.b, VIII.C.4.e. The IMU-Okabe Combination’s user also applies a matte to select pixels parallel to the stroke, within a user-defined threshold distance from the stroke, and extending from the starting point to the ending point of the stroke by using a matte—i.e., “the first set of pixels”—to be in a region of interest and thus animated. *See* Okabe, 1, 2; Section VIII.C.4.f. From these inputs, the IMU-Okabe Combination automatically generates animation frames by shifting the selected pixels in the stroke’s direction using IMV6’s Shepard’s Distortion—i.e., “automatically shift the first set of pixels in the first direction.” *See* Okabe, 1-2; Section VIII.C.3.b.

183. Additionally, IMU teaches that IMV6 allows a user to save an animation as an infinitely looping animated GIF using IMV6’s “-loop” operator. IMU-Animating, 1-2. Similarly, Okabe describes automatically producing an infinitely repeating animation that follows a direction and speed based on a user-drawn stroke. Okabe, 1-2, 8. The IMU-Okabe Combination thus likewise permits infinite repetition of an animation by, e.g., applying the “-loop” operator to an animation created by Shepard’s Distortion and the user’s stroke—i.e., “rendering

and re-rendering in a loop the first set of pixels being shifted.” *See* IMU-Animating, 1-2; Okabe, 1, 8; Section VIII.C.3.b. Therefore, in my opinion, the IMU-Okabe Combination meets limitation [1h].

- h. [1i]-[1o]: receive a second starting point through the user interface, wherein the second starting point is received through a user selection of a second starting portion of the digital image, the second starting portion being different than both the first starting portion and the first ending portion;**

render a third visual indication of the second starting point on the user interface;

receive, through the user interface, a second ending point associated with the second starting point, wherein the second ending point is received through a user selection of a second ending portion of the digital image, the second ending portion being different than both the first starting portion and the first ending portion;

render a fourth visual indication of the second end point on the user interface;

render, on the user interface, a second link between the second starting point and the second ending point;

identify a second set of pixels that are:

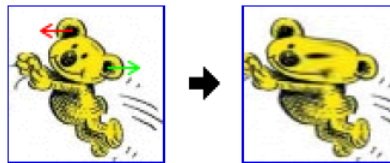
parallel to the second rendered link,

within the user-defined threshold distance from the first rendered link, and

extending from the second starting point to the second ending point; and

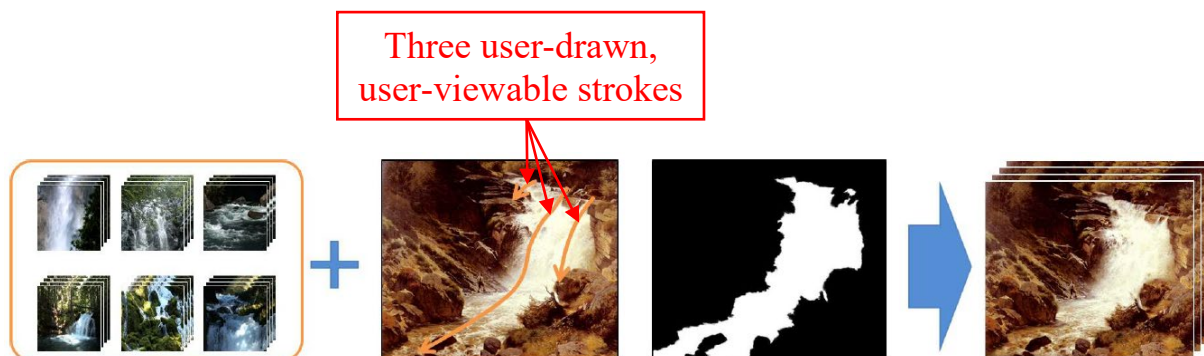
automatically shift the second set of pixels between the second starting point and the second ending point, wherein shifting the second set of pixels comprises rendering in a loop the second set of pixels being shifted.

184. In my opinion, the IMU-Okabe Combination meets limitations [1i]-[1o]. IMU teaches applying multiple Shepard's Distortion control points at different coordinates of an image in IMV6 and incrementally moving the control points to new coordinates to animate the image. *See* IMU-Distorting, 17-18, 59, 19-20. IMU provides an example that places and moves two Shepard's Distortion control points on an image of a koala:



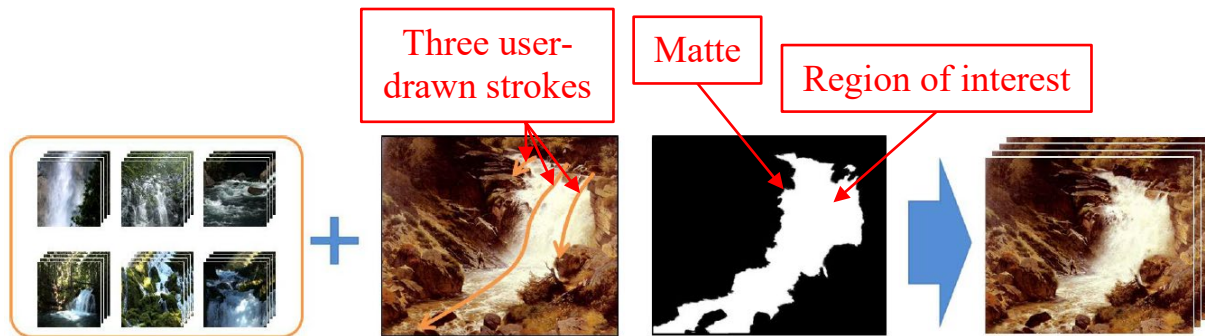
IMU-Distorting, 59.

185. Further, in the IMU-Okabe Combination, this placement and movement of multiple Shepard's Distortion control points is performed by a user drawing multiple “strokes” from Okabe on the image, each of which is user-viewable and specifies a direction and speed of a Shepard's Distortion control points. *See* Okabe, 7, 2-3; Section VIII.C.3.b. Indeed, Okabe's Figure 1 depicts three user-drawn, user-viewable strokes on:



Okabe, 2 (Figure 1, annotations added). Receiving multiple user-drawn strokes—including each stroke’s starting and ending points, direction, and speed—meets “receive” and “render” a “second starting point” as recited in limitations [1i]-[1j], “receive” and “render” a “second ending point” as recited in limitations [1k]-[1l], and “render, on the user interface, a second link between the second starting point and the second ending point” as recited in limitation [1m]. *See also* Sections VIII.C.4.c-VIII.C.4.e.

186. The IMU-Okabe Combination’s user also applies a matte specifying a region of interest in the image the user desires to animate, including pixels parallel to each stroke, within a user-defined threshold distance from the strokes, and extending from the starting point to the ending point of each stroke—i.e., “identify a second set of pixels that are: parallel to the second rendered link, within the user-defined threshold distance from the first rendered link, and extending from the second starting point to the second ending point” as recited in limitation [1n]. *See* Okabe, 1; IMU-Masking, 2; Section VIII.C.3.b; *accord* EX1022, 10-11; *see also* Section VIII.C.4.f. Okabe’s Figure 1 confirms this by depicting a matte specifying a region of interest that includes pixels parallel to each of three strokes, within a user-defined threshold distance from the strokes (i.e., to the waterfall’s edges) and extending from the starting point to the ending point of each stroke:



Okabe, 2 (Figure 1, annotations added).

187. After receiving multiple strokes and the matte from the user, the IMU-Okabe Combination automatically generates animation frames using Shepard’s Distortion and permits infinite repetition of the animation by, e.g., applying IMV6’s “-loop” operator—i.e., “automatically shift the second set of pixels between the second starting point and the second ending point, wherein shifting the second set of pixels comprises rendering in a loop the second set of pixels being shifted” as recited in limitation [1o]. See Okabe, 1-2, 8; IMU-Animating, 1-2; Section VIII.C.3.b; *accord* EX1022, 10-11; *see also* Section VIII.C.4.g.

5. Claim 2

- a. **[2a]-[2b]: The computer system of claim 1, wherein the executable instructions include instructions that are executable to configure the computer system to identify a particular portion of the digital image to mask, wherein the mask prevents pixels covered by the mask from being shifted.**

188. In my opinion, the IMU-Okabe Combination meets limitations [2a]-[2b]. Similar to how IMV6 enables a user to apply a matte to modify the transparency of an image’s pixels and therefore the visibility of effects applied to

the pixels (IMU-Masking, 2), the IMU-Okabe Combination's user applies a matte specifying a region of interest in the image the user desires to animate (*see* Okabe, 1). Section VIII.C.3.b. Inherently, the matte also specifies a region that includes pixels the user does not desire to animate. Indeed, Okabe's Figure 1 depicts a matte that includes a white region (pixels to be animated) and black region (pixels not to be animated):



Okabe, 2 (Figure 1, annotations added). Therefore, the matte includes a “mask [that] prevents pixels covered by the mask from being shifted” as recited in limitation [2b]. Receiving and applying such a matte specified by the IMU-Okabe Combination's user meets “identify a particular portion of the digital image to mask” as recited in limitation [2a].

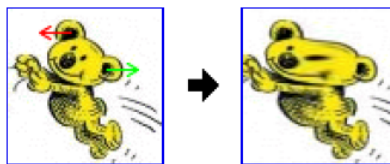
6. **Claim 3: The computer system of claim 2, wherein receiving an indication of a particular portion of the digital image to mask comprises receiving through a user interface a selection of a particular pixel within the digital image.**

189. In my opinion, the IMU-Okabe Combination meets claim 11. The IMU-Okabe Combination's user applies a matte that specifies a region of an image that the user does not desire to animate. *See* Okabe, 1-2; Section VIII.C.5.a. As

seen in Okabe's Figure 1, such a matte includes a black region covering pixels the user does not desire to animate, each of which satisfies a "particular pixel within the digital image." Okabe, 2. Receiving the matte from the user meets the claimed "receiving through a user interface a selection."

7. **Claim 5: The computer system of claim 1, wherein the first direction is different from the second direction.**

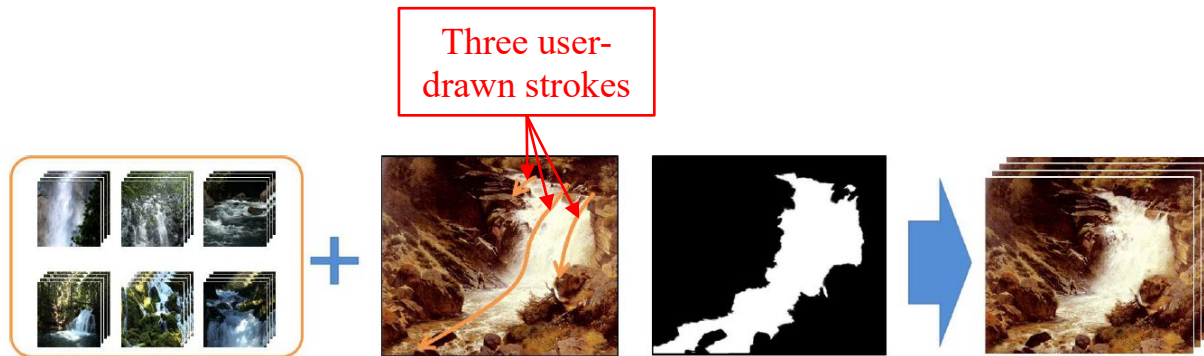
190. IMU teaches applying and incrementally moving control points, such as for Shepard's Distortion, to different coordinates on an image to generate an animation. *See* IMU-Distorting, 17-18, 59, 19-20. IMU provides an example that places two Shepard's Distortion control points on an image of a koala and moves them in different directions:



IMU-Distorting, 59.

191. Further, in the IMU-Okabe Combination, this placement and movement of multiple Shepard's Distortion control points is performed by a user drawing multiple "strokes" from Okabe on the image, which specify different directions and speeds of corresponding Shepard's Distortion control points across the animation—i.e., the claimed "wherein the first direction is different from the second direction."

See Okabe, 7, 3; Section VIII.C.3.b. In fact, Okabe's Figure 1 depicts three user-drawn strokes with different directions:



Okabe, 2 (Figure 1, annotations added). Therefore, in my opinion, the IMU-Okabe Combination meets claim 5.

8. **Claim 6: The computer system of claim 1, wherein the magnitude of the shifting of the first set of pixels is proportionally related to the first length and the magnitude of the shifting of the second set of pixels is proportionally related to the second length.**

192. In my opinion, the IMU-Okabe Combination meets claim 6. IMU teaches that distortions using control points (e.g., Shepard's Distortion) require users to input "2 pairs of coordinates": X_i, Y_i and I_i, J_i . IMU-Distorting, 19, 59. Specifically, "control point $X_i, [Y]_i$ in the source image (relative [to] its virtual canvas), is mapped to I_i, J_i on the distorted destination image." IMU-Distorting, 19. Thus, when generating animation frames using Shepard's Distortion, the magnitude of distortion from one frame to the next is directly related to the distance between the "source" and "destination" coordinates inputted for a corresponding control point. See IMU-Distorting, 19, 59.

193. In the IMU-Okabe Combination, the placement and movement of multiple Shepard's Distortion control points is performed by drawing a "stroke[]" from Okabe on the image to specify the direction and speed of a corresponding Shepard's Distortion control point across the animation. *See* Okabe, 7, 3; Section VIII.C.3.b. Thus, longer strokes correspond to larger frame-by-frame distances of travel for corresponding Shepard's Distortion control points, resulting in larger shifts, and vice versa. *See* IMU-Distorting, 19, 59; Section VIII.C.3.b. Such establishes a proportional relationship between the magnitude of Shepard's Distortion applied and the length of each corresponding stroke—i.e., the claimed "magnitude of the shifting of the first set of pixels is proportionally related to the first length and the magnitude of the shifting of the second set of pixels is proportionally related to the second length."

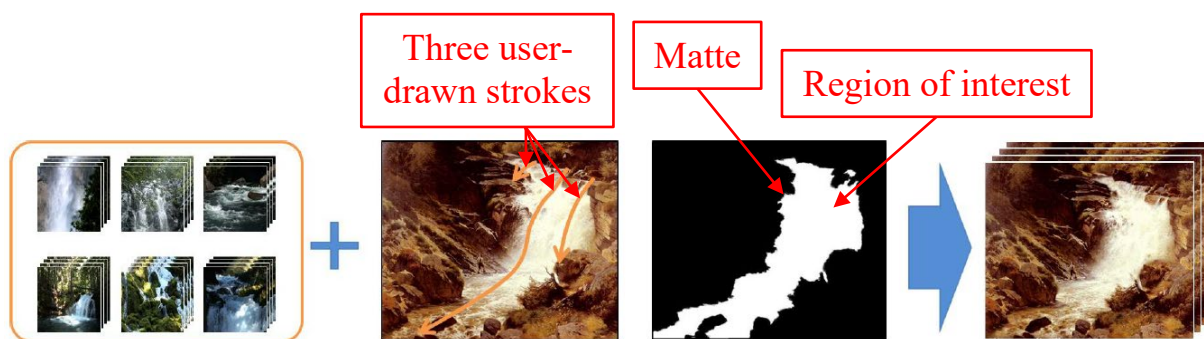
9. Claim 7

a. [7a]: The computer system of claim 1, wherein:

the first set of pixels comprise pixels that intersect the first rendered link; and.

194. The IMU-Okabe Combination's user applies a matte (*see* IMU-Masking, 2) specifying a to-be-animated region of interest in the image the user desires to animate (*see* Okabe, 1), in addition to a "stroke[]" that specifies the direction and speed of a Shepard's Distortion control point across the animation (*see* Okabe, 7, 3). Sections VIII.C.4.f, VIII.C.3.b. Thus, using a matte, a user would

have selected and included in this region of interest not only pixels parallel to the stroke, within a user-defined threshold distance from the stroke, and extending from the starting point to the ending point of the stroke, but also pixels intersecting the stroke—i.e., the claimed “first set of pixels comprise pixels that intersect the first rendered link.” *See* Okabe, 1. Okabe’s Figure 1 confirms this by depicting a matte specifying a region of interest in white that includes pixels intersecting three strokes:



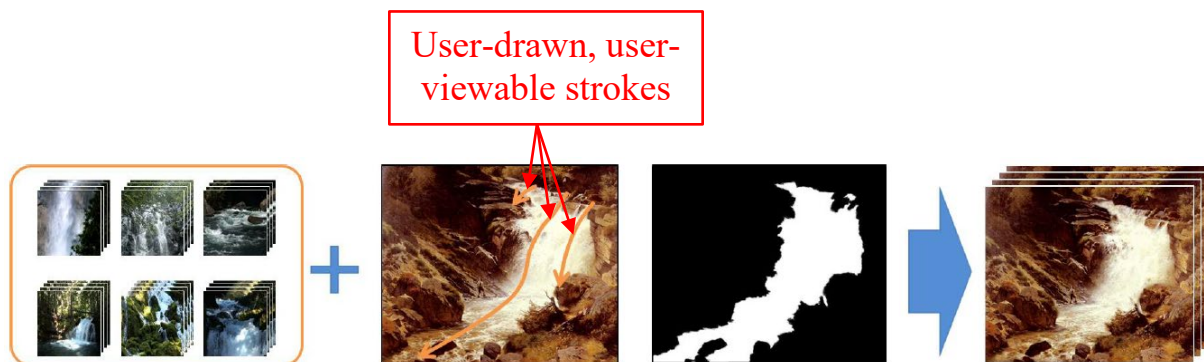
Okabe, 2 (Figure 1, annotations added). This confirms that the IMU-Okabe Combination meets limitation [7a].

- b. **[7b]: the first rendered link comprises a non-linear pathway extending between the first starting point and the first ending point.**

195. IMU teaches that, when animating in IMV6 using distortion effects such as Shepard’s Distortion, the direction and speed of animation desired by the user (*see* Section VIII.C.4.e) may be non-linear as specified by coordinates input by the user. *See, e.g.*, IMU-Distorting, 17-18 (placing and moving an “SRT distortion” control point on an image of a space ship to animate the launching of the space ship

along a non-linear path through the sky); *see also* IMU-Distorting 59, 19-20; Section VIII.C.3.b.

196. Further, in the IMU-Okabe Combination, a user specifies such a non-linear direction and speed of a Shepard's Distortion control point by drawing a corresponding non-linear “stroke[]” from Okabe on the image extending between the stroke's starting and ending point—i.e., the claimed “non-linear pathway extending between the first starting point and the first ending point.” *See* Okabe, 7, 2-3; Section VIII.C.3.b. Indeed, Okabe's Figure 1 illustrates three non-linear strokes received from the user:



Okabe, 2 (Figure 1, annotations added). Similarly, Okabe's Figure 8-a depicts two non-linear strokes, each shown as a green curved line with an arrow specifying direction:



(a) Target Image

Okabe, 6 (discussing Figure 8-a). Therefore, in my opinion, the IMU-Okabe Combination meets limitation [7b].

10. Independent Claim 8

- a. **[8pre]: A method for automating the shifting of pixels within a digital image, comprising.**

197. The IMU-Okabe Combination meets the claimed “method for automating the shifting of pixels within a digital image.” Sections VIII.C.10.b-VIII.C.10.g.

- b. **[8a]-[8b]: receiving a first indication of a first starting point through a user interface, wherein the first starting point is received through a user selection of a first starting portion of the digital image;**

render a first visual indication of the first starting point on the user interface.

198. *See* Section VIII.C.4.c.

- c. **[8c]-[8d]: receiving, through the user interface, a first ending point associated with the first starting point, wherein the first ending point is received through a user selection of a first ending portion of the digital image;**

render a second visual indication of the first ending point on the user interface.

199. *See* Section VIII.C.4.d. Additionally, the stroke's ending and starting positions are, as claimed, "associated with" one another by being positions for the same stroke. Section VIII.C.4.e.

- d. **[8e]: render, on the user interface, a first rendered link between the first starting point and the first ending point.**

200. *See* Section VIII.C.4.e.

- e. **[8f]: selecting a first set of pixels that intersect with the first rendered link.**

201. *See* Section VIII.C.9.a.

- f. **[8g]: automatically shifting the first set of pixels along the first rendered link between the first starting point and the first ending point, wherein shifting the first set of pixels comprises rendering in a loop the first set of pixels being shifted.**

202. *See* Sections VIII.B.3.g, VIII.B.9.a.

- g. [8h]-[8n]: receiving a second indication of a second starting point through the user interface, wherein the second starting point is received through a user selection of a second starting portion of the digital image, the second starting portion being different than both the first starting portion and the first ending portion;**

render a third visual indication of the second starting point on the user interface;

receiving, through the user interface, a second ending point associated with the second starting point, wherein the second ending point is received through a user selection of a second ending portion of the digital image, the second ending portion being different than both the first starting portion and the first ending portion;

render a fourth visual indication of the second end point on the user interface;

render, on the user interface, a second rendered link between the second starting point and the second ending point;

selecting a second set of pixels that intersect with the second rendered link;

automatically shifting the second set of pixels along the second rendered link between the second starting point and the second ending point, wherein shifting the second set of pixels comprises rendering in a loop the second set of pixels being shifted.

203. *See* Sections VIII.C.4.h, VIII.C.9.a.

11. Claim 9: **The method as recited in claim 8, further comprising when a pixel selected from the first set of pixels reaches the ending point, rendering and re-rendering in the loop the pixel being shifted from the first starting point to the ending point.**

204. The IMU-Okabe Combination's user selects pixels in a region of interest of an image using a matte, including pixels extending along a length of a user-drawn "stroke[]" and away from the stroke's starting point—i.e., the claimed "first set of pixels." *See* IMU-Masking, 2; Okabe, 1; Section VIII.C.4.f.

205. Further, the user-drawn stroke specifies the direction and speed of a Shepard's Distortion control point across the animation. *See* Okabe, 7, 3; IMU-Distorting, 59; Section VIII.C.3.b. Specifically, the stroke's starting point represents the placement of the Shepard's Distortion control point at a specified coordinate—i.e., "a pixel selected from the first set of pixels." *See* Okabe, 7, 3; IMU-Distorting, 59; Section VIII.C.4.c. The stroke's ending point meets the "ending point" "reache[d]" at the end of the animation. *See* Okabe, 7, 3; Section VIII.C.4.d.

206. The IMU-Okabe Combination then automatically generates animation frames using Shepard's Distortion and permits infinite repetition of the animation by, e.g., applying IMV6's "-loop" operator—i.e., "when a pixel selected from the first set of pixels reaches the ending point, rendering and re-rendering in the loop the pixel being shifted from the first starting point to the ending point." *See* IMU-

Animating, 1-2; Okabe, 1-2, 8; Section VIII.C.4.g. Therefore, in my opinion, the IMU-Okabe Combination meets claim 9.

12. Claim 10

- a. **[10a]-[10b]: The method as recited in claim 8, further comprising receiving an indication to generate a mask over a portion of the digital image, wherein pixels under the mask are prevented from shifting.**

207. *See* Section VIII.C.5.a.

13. Claim 11: **The method as recited in claim 10, further comprising receiving through a user interface a selection of a particular pixel within the digital image from which the mask should be generated.**

208. *See* Section VIII.C.6.

14. Claim 13

- a. **[13a]: The method of claim 8, wherein: the first set of pixels further comprise:**

pixels that are parallel to the first rendered link,

within a user-defined threshold distance from the first rendered link, and

that extend from the first starting point to the first ending point; and

209. *See* Section VIII.C.4.f.

- b. **[13b]: the first rendered link comprises a non-linear pathway extending between the first starting point and the first ending point.**

210. *See* Section VIII.C.9.b.

15. Independent Claim 14

- a. **[14pre]: A method for automating the shifting of pixels within a digital image, comprising.**

211. The IMU-Okabe Combination meets the claimed “method for automating the shifting of pixels within a digital image.” Sections VIII.C.15.b-VIII.C.15.hVIII.B.16.h.

- b. **[14a]: accessing, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within the digital image.**

212. *See* Section VIII.C.4.c.

- c. **[14b]-[14c]: receiving a first starting point through a user interface, wherein the first starting point is received through a user selection of a first beginning portion of the digital image;**

rendering a first visual indication of the first starting point on the user interface.

213. *See* Section VIII.C.4.c.

- d. **[14d]-[14e]: receiving a first ending point through the user interface, wherein the first ending point is received through a user selection of a first ending portion of the digital image;**

rendering a second visual indication of the first ending point on the user interface.

214. *See* Section VIII.C.4.d.

- e. **[14f]: rendering, on the user interface, a first rendered link between the first starting point and the first ending point, wherein the first link comprises:**

a non-linear pathway extending between the first starting point and the first ending point; and

a first length extending along the non-linear pathway.

215. *See* Sections VIII.C.4.e, VIII.C.9.b.

- f. **[14g]: identifying a first set of pixels that lie along the first rendered link.**

216. *See* Sections VIII.C.4.f, VIII.C.9.a-VIII.C.9.b.

- g. **[14h]: automatically shifting the first set of pixels along the non-linear pathway, wherein shifting the first set of pixels comprises rendering and re-rendering in a loop the first set of pixels being shifted.**

217. *See* Sections VIII.C.4.g, VIII.C.9.a-VIII.C.9.b.

- h. [14i]-[14o]: receiving a second starting point through the user interface, wherein the second starting point is received through a user selection of a second starting portion of the digital image, the second starting portion being different than both the first starting portion and the first ending portion;**

rendering a third visual indication of the second starting point on the user interface;

receiving, through the user interface, a second ending point associated with the second starting point, wherein the second ending point is received through a user selection of a second ending portion of the digital image, the second ending portion being different than both the first starting portion and the first ending portion;

rendering a fourth visual indication of the second end point on the user interface;

rendering, on the user interface, a second link between the second starting point and the second ending point;

**selecting a second set of pixels along the second link;
and**

automatically shifting the second set of pixels between the second starting point and the second ending point, wherein shifting the second set of pixels comprises rendering in a loop the second set of pixels being shifted.

218. *See* Sections VIII.C.4.h, VIII.C.9.a-VIII.C.9.b.

16. Claim 15

- a. **[15a]-[15b]: The method of claim 14, further comprising identifying a particular portion of the digital image to mask, wherein the mask prevents pixels covered by the mask from being shifted.**

219. *See* Section VIII.C.5.a.

17. Claim 16: **The method of claim 15, wherein receiving an indication of a particular portion of the digital image to mask comprises receiving through a user interface a selection of a particular pixel within the digital image.**

220. *See* Section VIII.C.6.

18. Claim 18: **The method of claim 15, wherein:**

the first set of pixels further comprise:

pixels that are parallel to the first rendered link,

within a user-defined threshold distance from the first rendered link, and that extend from the first starting point to the first ending point.

221. *See* Sections VIII.C.9.a-VIII.C.9.b, VIII.C.4.f.

D. Ground 3: IMU, Okabe, and Li, and Claims 2-4, 10-12, and 15-17

1. Summary of Li

222. Li describes “a novel coarse-to-fine UI design for image cutout” named “Lazy Snapping.” Li, 2. “The task in image cutout is specifying which parts of the image are ‘foreground’ (the part you want to cut out) and which belong to the background.” Li, 1.

223. Li teaches that performing image cutout on a computer historically required a user to specify the foreground by “marking every pixel individually”—a “tedious” and “particularly frustrating task for users.” Li, 1. Li’s “Lazy Snapping” tool, however, allows a user to easily perform image cutout in two steps: “a quick *object marking* step (b) and a simple *boundary editing* step (c).” Li, 2. Li’s Figure 1 illustrates how these two steps are performed in “Lazy Snapping”:

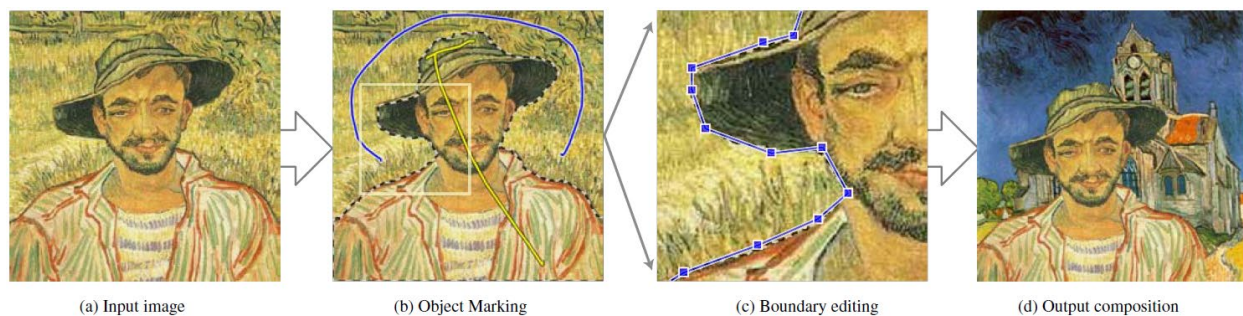
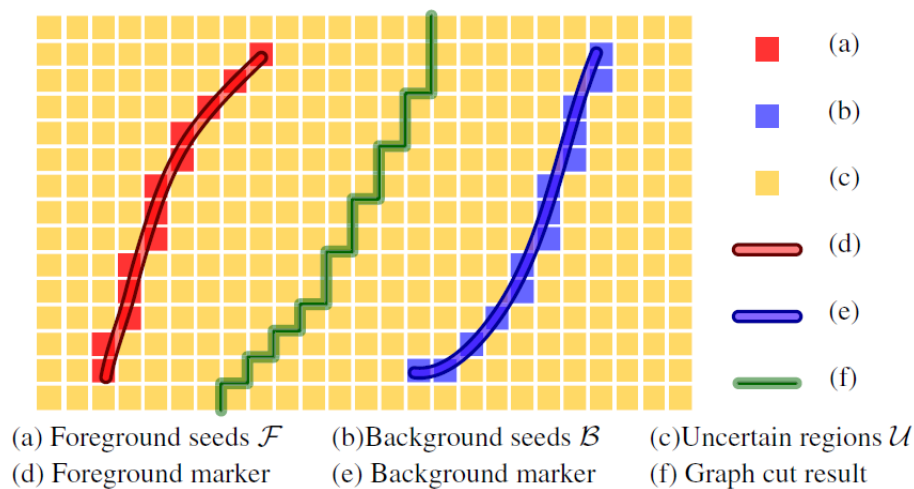


Figure 1: Lazy Snapping is an interactive image cutout system, consisting of two steps: a quick object marking step and a simple boundary editing step. In (b), only 2 (yellow) lines are drawn to indicate the foreground, and another (blue) line to indicate the background. All these lines are far away from the true object boundary. In (c), an accurate boundary can be obtained by simply clicking and dragging a few polygon vertices in the zoomed-in view. In (d), the cut out is composed on another Van Gogh painting.

Li, 1.

224. As illustrated, Li teaches that the first “object marking” step “works at a coarse scale,” where the user “marks a few lines on the image by dragging the mouse cursor while holding a button (left button indicating the foreground, and right button for the background). A yellow line or a blue line is displayed for the foreground marker or background marker respectively.” Li, 2. A “segmentation process” is then triggered that uses a “novel interactive graph cut algorithm” to detect a “cutout boundary” between the foreground and background lines. Li, 2. Li’s

Figure 2 depicts an example of detecting such a boundary (in green) between a foreground line (in red) and a background line (in blue):



Li, 2-3.

225. The second “boundary editing” step allows the user to further refine the boundary. Li, 3. This second step “works at a finer scale” by converting the boundary into “editable polygons” and allowing the user to manually refine the boundary “by simply clicking and dragging polygon vertices.” Li, 3, 2.

2. The IMU-Okabe-Li Combination

a. **Motivation to Combine the IMU-Okabe Combination with Li**

226. The prior art contains express teachings, suggestions, and motivations for combining the IMU-Okabe Combination (Section VIII.C.3.b) with Li’s “Lazy Snapping” tool (hereinafter, the “IMU-Okabe-Li Combination”).

227. IMU teaches that, in IMV6, a user applies a matte to make only a particular region of an image visible when animating the image using Shepard’s

Distortion. IMU-Masking, 2; Section VIII.C.3.a. Similarly, in the IMU-Okabe Combination, a user applies a matte that, as taught in Okabe, specifies regions that the user desires to animate (in white) and not animate (in black). Okabe, 1-2; Sections VIII.C.3.b, VIII.C.5.a.

228. Okabe teaches that its matte can be created “using a scribble-based image segmentation tool,” specifically Li’s “Lazy Snapping” tool. Okabe, 7 (citing “LSTS04”), 10 (indicating “LSTS04” is the shorthand for Li). As such, Okabe contains an explicit teaching, suggestion, and motivation for using Li’s “Lazy Snapping” tool to create a matte in Okabe and the IMU-Okabe Combination, and thus for combining the IMU-Okabe Combination with Li. Okabe, 7.

229. Separately, Li itself teaches that its “Lazy Snapping” tool allows a user to easily perform “image cutout”—i.e., separating and removing an image’s background, leaving only the foreground visible—by using a “novel image segmentation algorithm” to detect a boundary between the foreground and background based on “a quick *object marking* step” and “a simple *boundary editing* step” performed by the user. Li, 1-2. Compared to other existing methods and tools for image cutout, Li’s tool “outperforms in terms of ease of use, efficiency, and quality of results.” Li, 1-2.

230. Given such benefits, a POSITA would have been motivated to utilize Li’s “Lazy Snapping” tool to remove the background of an image and make only the

foreground visible—i.e., to create a matte in IMV6 that makes the former portion of the image transparent and the latter portion visible. *See* IMU-Masking, 2; Section VIII.C.3.a. Further, given that a user of the IMU-Okabe Combination applies a matte that, as taught in Okabe, specifies regions in an image to be or not to be animated (*see* Okabe, 1; Section VIII.C.3.b), a POSITA would have likewise been motivated to utilize Li’s “Lazy Snapping” tool in the IMU-Okabe Combination to create such a matte for the same benefits of “ease of use, efficiency, and quality of results” (*see* Li, 2). Indeed, Okabe describes utilizing Li’s tool in this exact way, and states doing so allows a matte to be created in “less than 5 min[utes],” confirming that a POSITA would have been motivated to modify the IMU-Okabe Combination in view of Li to reach the IMU-Okabe-Li Combination. Okabe, 7.

231. Additionally, AECS6 was an image manipulation program “readily available” by the ECPD that included not only the functionality of the IMU-Okabe Combination (*see* Section VIII.C.3), but also functionality similar to Li’s “Lazy Snapping” tool (*see* Section VIII.D.1). In particular, compared to Li’s “object marking” step (Li, 2-3), AECS6’s “Roto Brush” tool similarly allows a user to create a matte that isolates a layer’s foreground from its background by having the user draw “strokes” on the layer over “representative areas of the foreground and background elements,” which AECS6 then uses to determine a “segmentation boundary” between the foreground and background elements via “Edge Detection”

(AEM, 328-331). Given AECS6 already contained functionality similar to the IMU-Okabe-Li Combination, a POSITA therefore would have been—and in fact was—motivated to further modify the IMU-Okabe Combination in view of Li to reach the IMU-Okabe-Li Combination.

b. Resulting Combination of the IMU-Okabe Combination and Li

232. So modified, the IMU-Okabe-Li Combination allows a user to create a matte by first performing an “object marking” step of drawing marker lines on an image to indicate the foreground—i.e., the region of interest that the user desires to animate—and the background—i.e., the region of the image that the user does not desire to animate. *See* Li, 2; Okabe, 1, 7. The IMU-Okabe-Li Combination utilizes Li’s “novel image segmentation algorithm” to detect the boundary between the image’s foreground and background based on the user’s marker lines. *See* Li, 1-2; Okabe, 7. The IMU-Okabe-Li Combination also allows the user to perform a “boundary editing” step to manually refine the boundary. *See* Li, 2; Okabe, 7. The resulting matte is then used in the remaining animation steps discussed in Section VIII.C.3.b. *See* Okabe, 1, 7.

3. Claim 2

- a. **[2a]-[2b]: The computer system of claim 1, wherein the executable instructions include instructions that are executable to configure the computer system to identify a particular portion of the digital image to mask, wherein the mask prevents pixels covered by the mask from being shifted.**

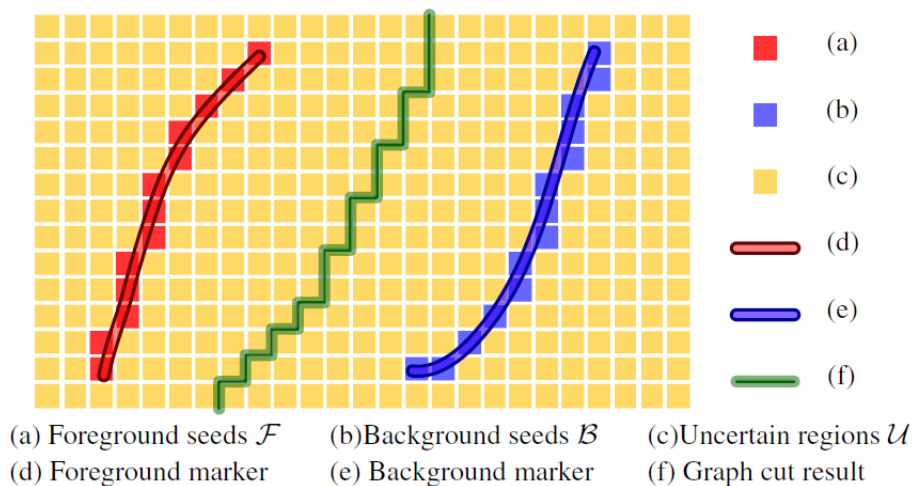
233. In my opinion, the IMU-Okabe-Li Combination meets limitations [2a]-[2b]. The IMU-Okabe-Li Combination allows a user to create a matte that specifies a region to animate (foreground) and a region not to animate (background). *See* Li, 1-2; Section VIII.D.2.b. Such a matte thus meets the claimed “mask [that] prevents pixels covered by the mask from being shifted” as recited in limitation [2b].

234. To create this matte, the IMU-Okabe Combination allows a user to perform Li’s first “object marking” step of drawing marker lines indicating either the foreground or background. *See* Li, 2; Okabe, 1, 7; Section VIII.D.2.b. Receiving such marker lines from the user, from which the matte is created, meets “identify a particular portion of the digital image to mask” recited in limitation [2a].

235. The IMU-Okabe-Li Combination then generates a boundary based on the user’s marker lines and thereafter allows the user to perform Li’s second “boundary editing” step to manually refine the boundary. *See* Li, 2; Okabe, 7; Section VIII.D.2.b. Receiving such refinements from the user to modify the boundary—and the resulting matte—additionally meets “identify a particular portion of the digital image to mask” as recited in limitation [2a].

4. **Claim 3: The computer system of claim 2, wherein receiving an indication of a particular portion of the digital image to mask comprises receiving through a user interface a selection of a particular pixel within the digital image.**

236. In my opinion, the IMU-Okabe-Li Combination meets claim 3. Li teaches that the “object marking” step of drawing marker lines comprises drawing marker lines over the particular pixels forming the foreground and background. Li, 2. Specifically, “[o]nce the user marks the image, two sets of pixels intersecting with the foreground and background markers are defined as *foreground seeds* $\mathcal{F}$ and *background seeds* $\mathcal{B}$ respectively, as shown in Figure 2”:



Li, 2-3. Such foreground and background “pixels” or “seeds” are then used in Li’s segmentation algorithm to detect the boundary between the foreground and background. Li, 2-3. Each “pixel[]” or “seed[]” meets the claimed “particular pixel within the digital image.” Li, 2-3. Receiving the foreground and background marker lines, and their intersecting “pixels” or “seeds,” meets the claimed “receiving an

indication of a particular portion of the digital image to mask comprises receiving through a user interface a selection.” Li, 2-3.

5. Claim 4

- a. **[4a]-[4c]: The computer system of claim 2, wherein the executable instructions include instructions that are executable to configure the computer system to generate the mask by:**

identifying one or more edges that form a boundary around the particular pixel; and

generating the mask to cover area within the boundary.

237. In my opinion, the IMU-Okabe-Li Combination meets limitations [4a]-[4c]. Li teaches that, in the “object marking” step, a user draws marker lines over the particular “pixels” or “seeds” that form the foreground and background. Li, 2-3; Section VIII.D.4. Such “pixels” or “seeds” are used to detect the boundary between the foreground and background for generating the matte, meeting the claimed “generate the mask by: identifying one or more edges that form a boundary around the particular pixel; and generating the mask to cover area within the boundary” as recited in limitations [4a]-[4c]. Li, 2-3.

6. Claim 10

- a. **[10a]-[10b]: The method as recited in claim 8, further comprising receiving an indication to generate a mask over a portion of the digital image, wherein pixels under the mask are prevented from shifting.**

238. *See* Sections VIII.D.3.a, VIII.D.4.

7. Claim 11: The method as recited in claim 10, further comprising receiving through a user interface a selection of a particular pixel within the digital image from which the mask should be generated.
239. See Section VIII.D.4.
8. Claim 12
- a. [12a]-[12b]: The method of claim 11, further comprising:
- identifying one or more edges that form a boundary around the particular pixel; and
- generating the mask to cover area within the boundary.
240. See Section VIII.D.5.a.
9. Claim 15
- a. [15a]-[15b]: The method of claim 14, further comprising identifying a particular portion of the digital image to mask, wherein the mask prevents pixels covered by the mask from being shifted.
241. See Sections VIII.D.3.a, VIII.D.4.
10. Claim 16: The method of claim 15, wherein receiving an indication of a particular portion of the digital image to mask comprises receiving through a user interface a selection of a particular pixel within the digital image.
242. See Section VIII.D.4.

11. Claim 17

a. [17a]-[17c]: The method of claim 15, further comprising:

generating the mask, wherein generating the mask comprises:

identifying one or more edges that form a boundary around the particular pixel; and

generating the mask to cover area within the boundary.

243. See Section VIII.D.5.a.

IX. CONCLUSION

244. All statements made herein of my own knowledge are true, all statements made herein on information and belief are believed to be true, and these statements were made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under 18 U.S.C. §1001.

Dated: Jan. 17, 2023



Philip Greenspun, Ph.D.

ATTACHMENT A

Resume

of [Philip Greenspun](#); updated May 2022

Summary:

- Business experience: started six companies and buried three. As CEO, grew an open-source enterprise software company to \$20 million annual revenue in two years with \$10,000 in capital. Served as corporate board member for venture capital-backed MIT spinoff companies.
- Software product development experience: 20 years. Same email address since 1976: philg@mit.edu. Developing open source software since 1982. List of engineering projects completed is available from <http://philip.greenspun.com/personal/resume-list>
- Pedagogy experience: Co-developed "[Software Engineering for Internet Applications](#)" with Hal Abelson at MIT; it has been a successful course at MIT and is being used by computer science departments at 10 other universities around the world. Re-developed 6.087 for the MIT Department of Aeronautics and Astronautics. Co-developed the RDBMS materials for a Harvard Medical School course on computational medicine.
- Non-profit experience: Started a 501c3 foundation in December 1998. The Foundation operated a prize program for high-school age Web developers and a one-year post-baccalaureate program in computer science; the annual budget was approximately \$1.5 million.
- Political experience: Testified before the U.S. Senate Commerce Committee and the Subcommittee on Patents, Copyrights and Trademarks of the Senate Judiciary Committee
- Writing experience: four computer science textbooks, one book about North America and its people, numerous journal and magazine articles.
- Photography experience: started [photo.net](#) in 1993, an online community for photographers. Work published in dozens of print magazines and books and used for advertising (see [separate photo resume](#)).
- Aviation experience: holder of Airline Transport Pilot certificate with multi-engine, single-engine seaplane, and helicopter ratings; holder of flight instructor certificate with instrument and helicopter ratings; have flown single-engine aircraft to Alaska (twice) and just about everywhere else in North America and the Caribbean; have flown three coast-to-coast trips in Robinson helicopters; flew the Canadair Regional Jet out of JFK for Delta Airlines
- Education: three MIT degrees (including a Ph.D., but you can't call me "Dr. Greenspun" because my brother is a real doctor).

Employment Experience

Fall 2021: Florida Atlantic University

Teach Information Security at this 31,000-student branch of the State University System of Florida.

2018-present: Harvard University

Develop curricular materials for medical students and post-doc researchers learning how to query a 12 TB insurance claims database. Assist student groups with their analytics projects in SQL and R.

1991-present: Massachusetts Institute of Technology

Teach and expand the MIT computer science curriculum, conduct research, and supervise student research. Teach the most popular course in the MIT Department of Aeronautics and Astronautics,

16.687 (via Zoom for 2021, with over 500 students)

2013-present: Fifth Chance Media LLC

I develop software, write, photograph, and create videos for this publishing company whose current products are listed at fifthchance.com. Through Fifth Chance Media LLC I also work as a [software expert witness](#), especially in cases regarding [Internet software patents](#) (e.g., for Amazon, Ford, IBM, Microsoft, and the U.S. Department of Justice). I have also served as an [aviation expert witness](#), testifying in front of a Federal Court jury, and as a [relational database expert witness](#).

1993-2000; 2006-2007: photo.net

Started, programmed, financed, and managed this online learning community as a personal hobby. Spun it off in 2000 to a team of entrepreneurs who attempted to make it a profitable business. Took it back over in mid-2006 to clean up the content, software, and balance sheet (crippled with debt). With 600,000 registered users and 60 million page views per month, sold the company in April 2007 to NameMedia.

1997 through March 2000: ArsDigita Corporation

Started, financed, and managed this company, which developed an open-source toolkit for building collaborative Internet applications. Grew the company profitably from 5 part-time people to 80 full-timers and revenue of \$20 million per year. Between January and March 2000, negotiated and closed a \$38 million venture capital investment from Greylock and General Atlantic Partners. Handed over the reins to a team of professional managers brought in by the venture capitalists.

February 1988 through August 1990: Isosonics Corporation

Founded company to develop a product that stored digital data with consumer video recorders. Co-designed custom digital signal processor. Developed simulation environment, complete simulator for digital audio recorder (1.4 Mbits/second), microcode compiler on the Symbolics Lisp Machine. Used Lisp tools to develop error correction microcode and refine DSP architecture. Co-designed three phase locked loops. With partners, developed system for auditing television broadcasts nationwide by monitoring commercials and compiling reports for advertisers. We designed a single board that tunes a chosen channel, recognizes tagged advertisements and makes a record for each ad of time of broadcast, number of fields, video quality and color burst presence. Served as president of Isosonics from its inception until its dissolution.

April 1986 through November 1989: ConSolve Incorporated

Co-founded this construction automation company. With partner, developed initial product, obtained financing (from PaineWebber Ventures), hired software development, marketing and support staff, established R&D partnership with Tektronix, obtained government contracts and sold software. Was active participant in all important planning, legal, and management activities. Wrote every line of code in the first system shipped to a customer (Caterpillar).

November 1984 through August 1985: ICAD, Inc.

Co-founded company with three partners. With Patrick O'Keefe, developed Lisp software to automate mechanical engineering. The ICAD System was initially primarily intended for large steel structures, e.g., air-cooled heat exchangers, offshore oil rigs, coal-fired power plants, but has been extended to many general ME problems.

Company went public in January 1995 as Concentra with a market valuation of \$50 million and was subsequently acquired by Oracle Corporation.

June 1983 through November 1984: Symbolics, Inc.

Developed VLSI tools, including automatic layout functions and worked on the system architecture for the Ivory microprocessor (the base of all Symbolics products sold in the late 1980s). Wrote parts of the Symbolics operating system.

June 1982 through June 1983: Hewlett-Packard Labs

Wrote packet-switched network simulation software on Symbolics Lisp Machine. Helped architect, simulate and design prototype of HP's Precision Architecture RISC computer. The prototype took two man-years to complete and ran at VAX 11/780 speed in June 1983. This architecture became the basis of HP's computer product line for 15 years and then became the basis for the 64-bit generation of Intel processors.

1978 to 1982

Paid tuition and living expenses through MIT with employment and contract work for Wang Laboratories, Verbex Corporation, National Aeronautics and Space Administration, and other organizations.

Education (Massachusetts Institute of Technology)

Ph.D. 1999 in electrical engineering and computer science. Thesis title: *Architecture and Implementation of Online Communities*.

S.M. 1993 in electrical engineering and computer science. Thesis title: *Site Controller: A system for computer-aided civil engineering and construction*.

S.B. 1982 in mathematics. Completed coursework for electrical engineering S.B. with emphasis on digital systems and signal processing. Took undergraduate and graduate computer science courses, with an emphasis on algorithms. Took graduate courses in microeconomics and neurophysiology.

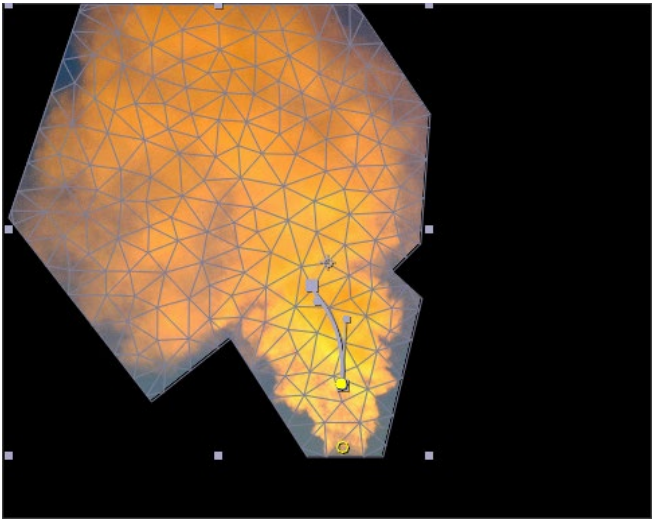
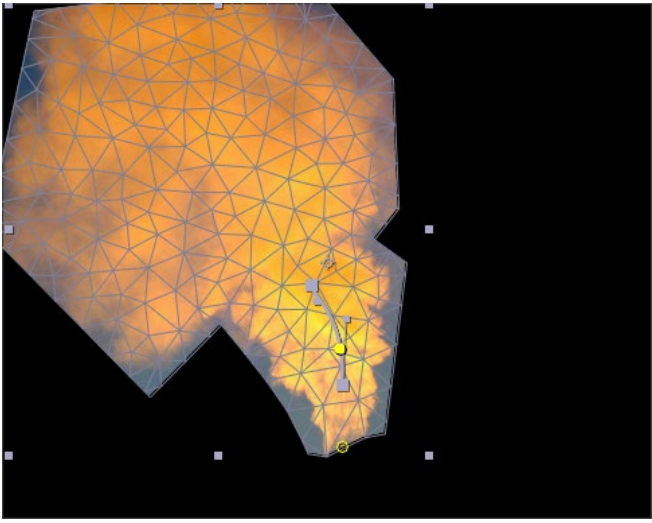
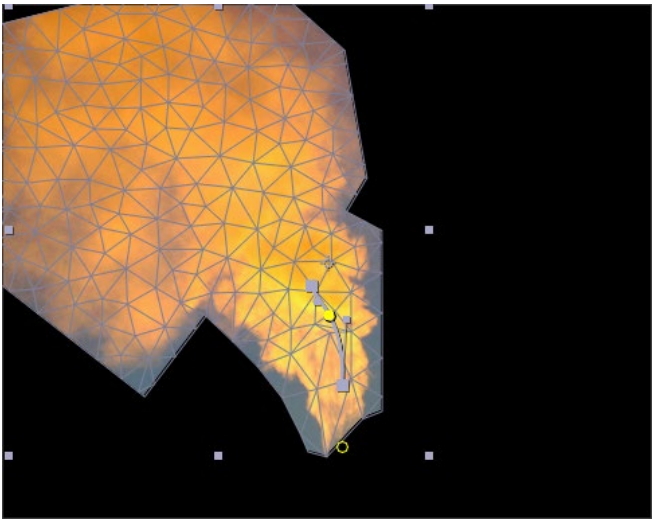
Selected Technical Publications

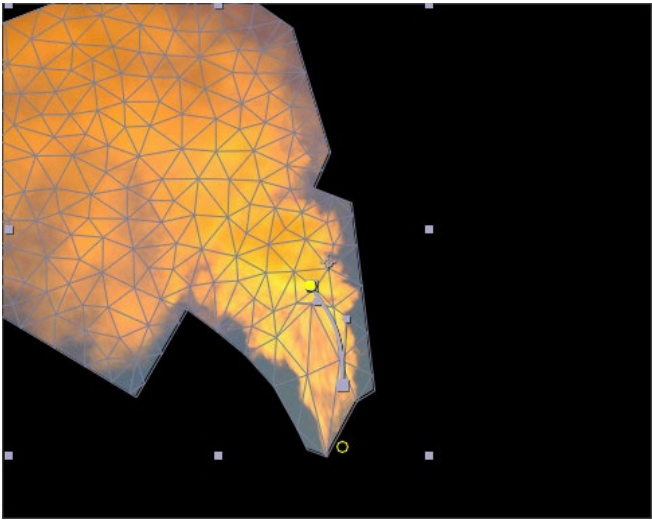
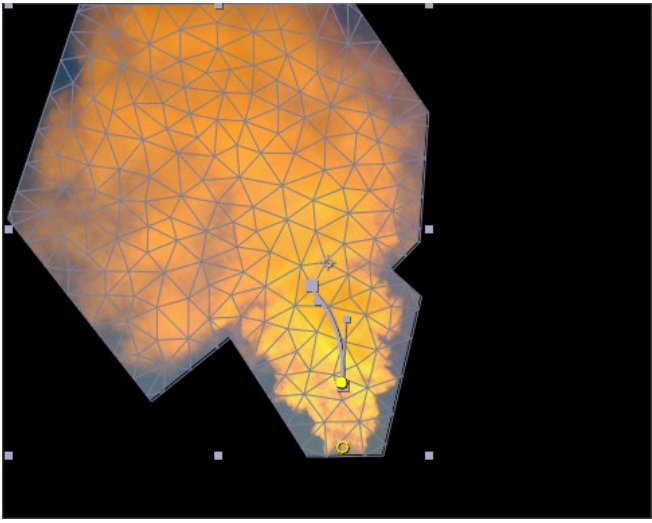
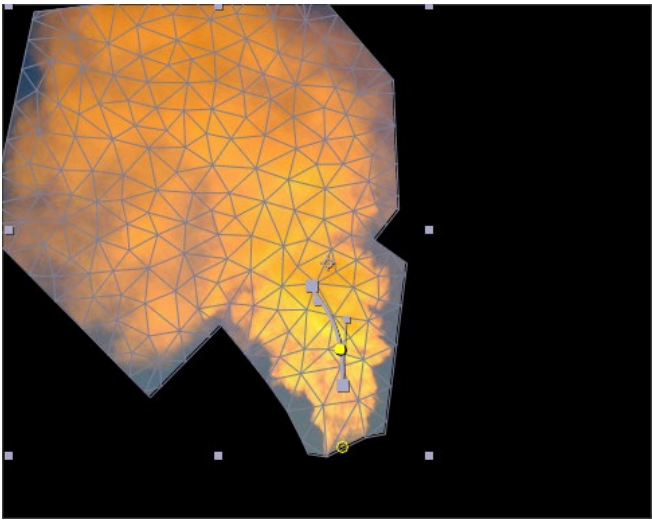
[*Software Engineering for Internet Applications*](#) (online and MIT Press 2006), [*Philip and Alex's Guide to Web Publishing*](#) (Morgan Kaufmann; 1999), [*Database Backed Web Sites*](#) (Ziff Davis Press; 1997), [*Travels with Samantha*](#), a book about North America; [*SITE CONTROLLER: A system for computer-aided civil engineering and construction*](#); various journal articles (most recent: [*"Medication Use in the Management of Comorbidities Among Individuals With Autism Spectrum Disorder From a Large Nationwide Insurance Database,"*](#) JAMA Pediatrics, June 2021); dozens of magazine articles. United States patents [5,172,363](#) (digital audio recorder circuit), [5,150,310](#) (location system), and [5,964,298](#) (computer-aided earthmoving system).

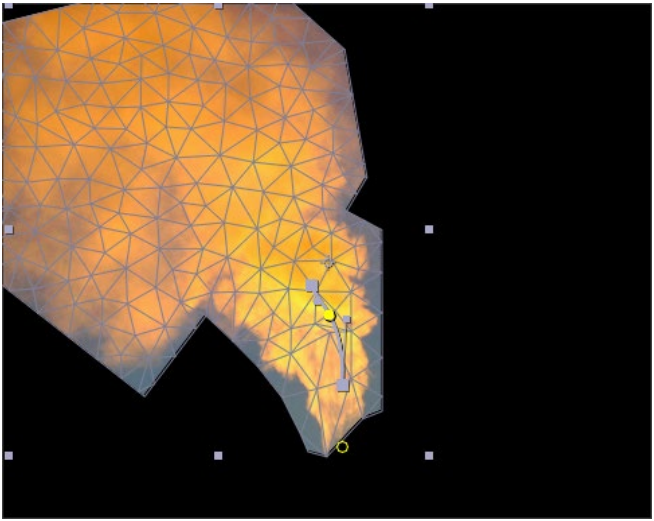
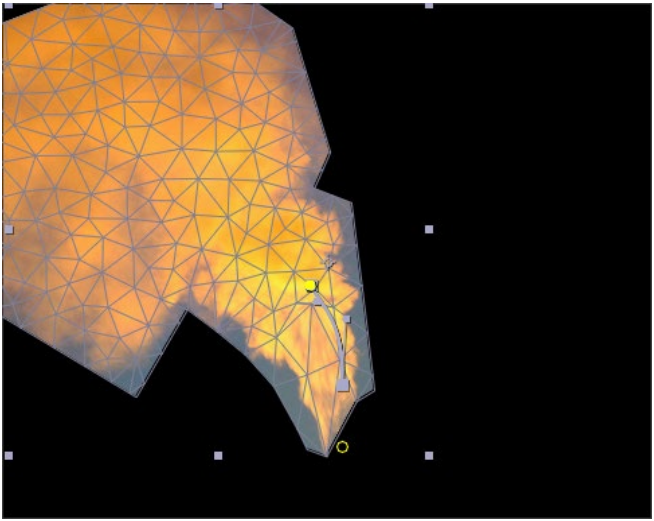
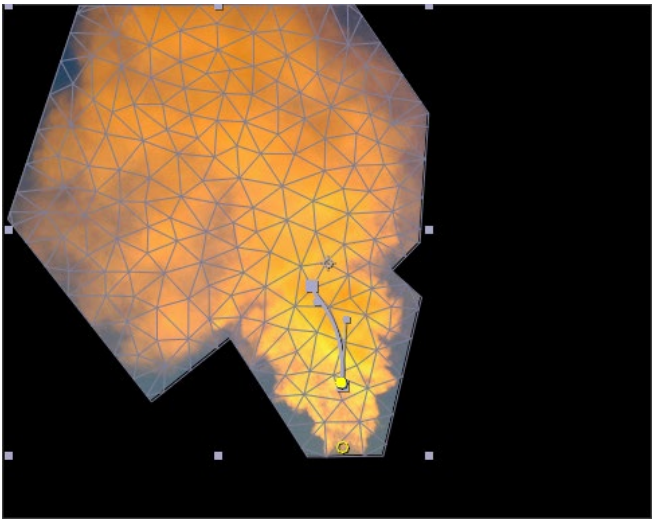
Most of my relevant publications are linked from philip.greenspun.com or fifthchance.com.

philg@mit.edu

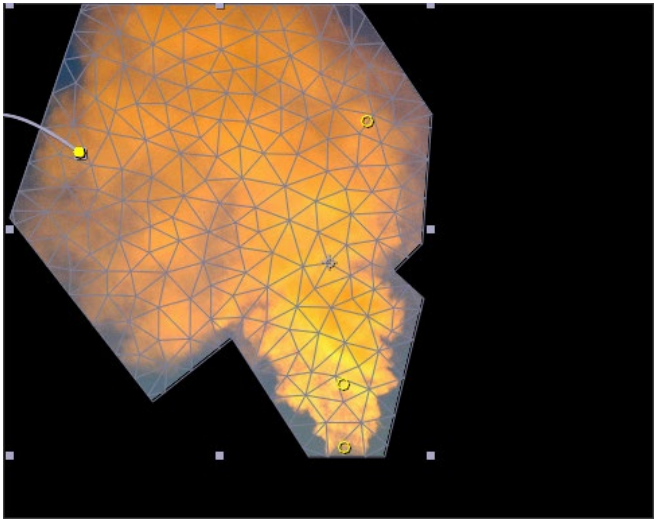
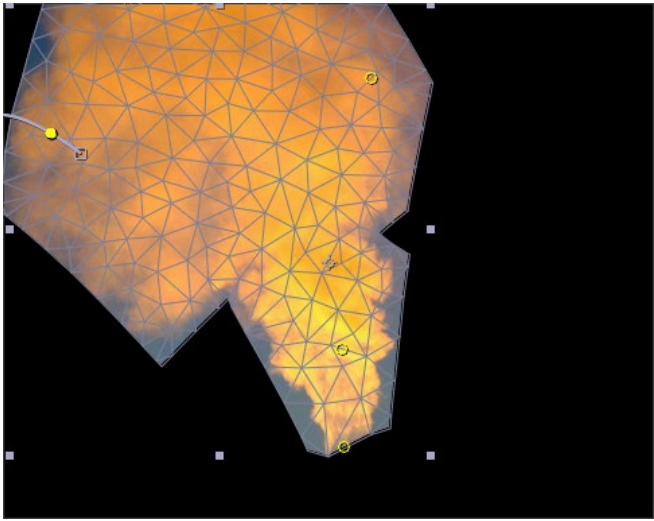
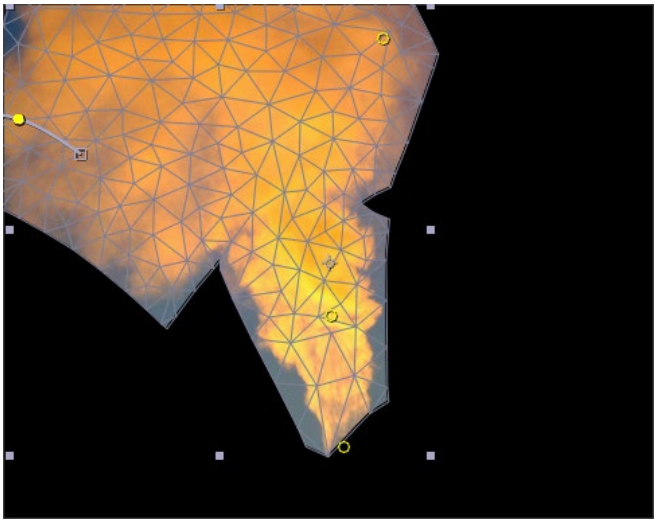
ATTACHMENT B

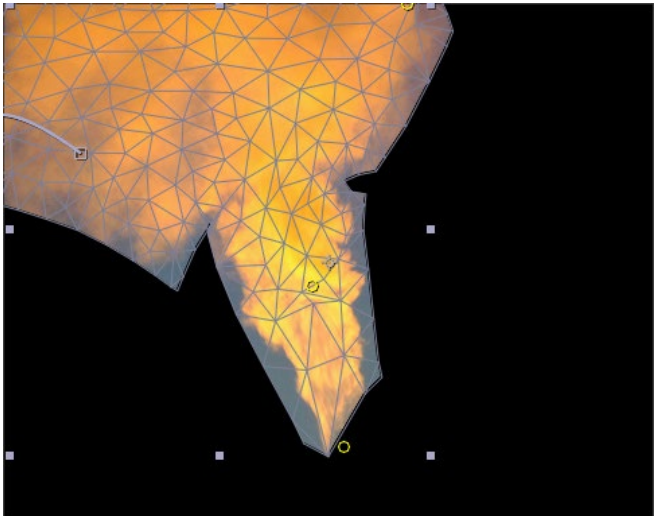
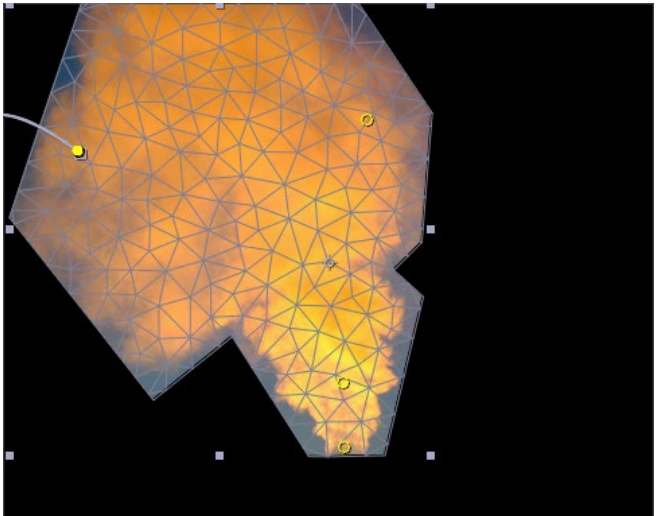
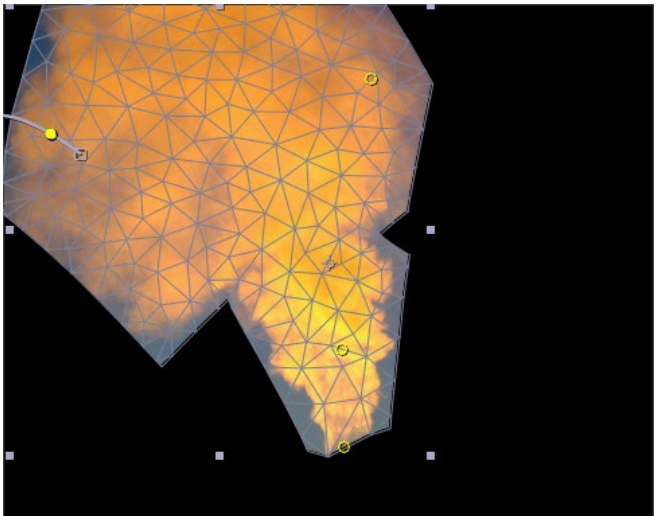
Animation Time Point	Composition Frame
(0:00:00:00)	
(0:00:01:00)	
(0:00:02:00)	

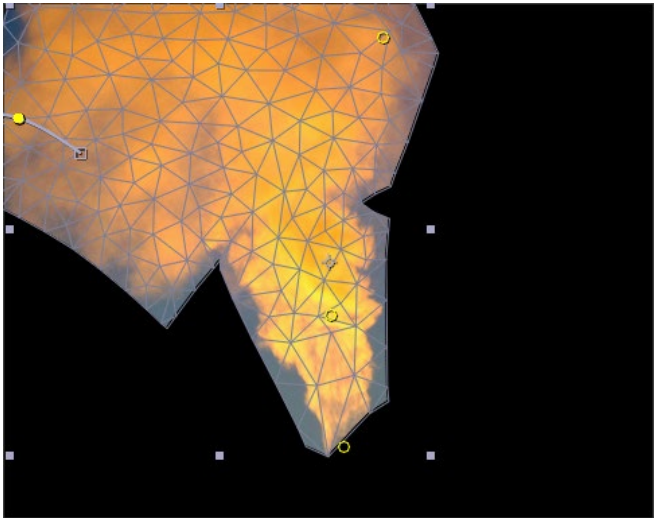
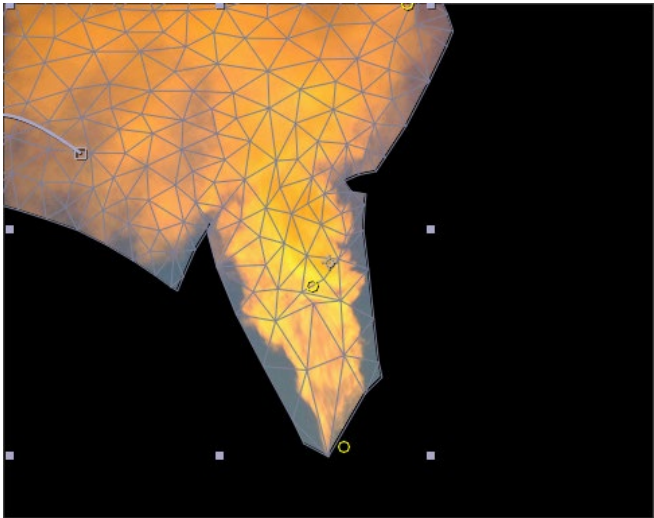
Animation Time Point	Composition Frame
(0:00:02:23)	 A wireframe map of a region, possibly a coastline or a specific area of interest, rendered in a yellow-orange color. The map is overlaid on a black background. A yellow path is visible, starting from a yellow circle at the bottom right and moving towards the top left. The map is composed of a grid of triangles, with the yellow path following a specific route through the grid.
(0:00:03:01)	 A wireframe map of a region, similar to the previous frame, rendered in a yellow-orange color. The map is overlaid on a black background. A yellow path is visible, starting from a yellow circle at the bottom right and moving towards the top left. The map is composed of a grid of triangles, with the yellow path following a specific route through the grid.
(0:00:04:00)	 A wireframe map of a region, similar to the previous frames, rendered in a yellow-orange color. The map is overlaid on a black background. A yellow path is visible, starting from a yellow circle at the bottom right and moving towards the top left. The map is composed of a grid of triangles, with the yellow path following a specific route through the grid.

Animation Time Point	Composition Frame
(0:00:05:00)	
(0:00:05:23)	
(0:00:06:01)	

ATTACHMENT C

Animation Time Point	Composition Frame
(0:00:00:00)	
(0:00:01:00)	
(0:00:02:00)	

Animation Time Point	Composition Frame
(0:00:02:23)	
(0:00:03:01)	
(0:00:04:00)	

Animation Time Point	Composition Frame
(0:00:05:00)	
(0:00:05:23)	
(0:00:06:01)	