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 IPR2023-00568
U.S. Patent No. 11,182,641
Filing Date: Feb. 26, 2020
Issue Date: Nov. 23, 2021

---

**DECLARATION OF PHILIP GREENSPUN, PH.D.**

## TABLE OF CONTENTS


I. INTRODUCTION AND SCOPE OF WORK .....21

II. EXPERIENCE AND QUALIFICATIONS.....21

III. COMPENSATION .....26

IV. LEGAL CONSIDERATIONS .....27

A. Level of Ordinary Skill in the Art .....27

B. Claim Construction.....28

V. BASIS FOR OPINIONS .....29

VI. TECHNOLOGY BACKGROUND.....29

VII. OVERVIEW OF THE ’641 PATENT .....37

A. Claim Construction.....38

VIII. ANALYSIS OF CLAIMS 1-4, 8-15, AND 19-20 OF THE ’641 PATENT IN VIEW OF THE PRIOR ART .....38

A. Public Availability of AEM, IMU, Okabe, and Li.....38

B. Ground 1: AEM, and Claims 1-4, 8-15, and 19-20.....43

1. Summary of AEM.....43

a. Workspace and Panels .....45

b. Compositions and Layers .....46

c. Extracting a Single Frame from a Video.....47

d. Modifying Transparency of Certain Pixels .....48

e. Puppet Effect .....50

f. Animating with the Puppet Effect .....51

2. Example of Animating in AECS6 .....55

a. Importing a Video into a Composition.....57

b. Extracting a Single Frame from the Video and Importing the Frame as a Layer.....59

c. Placing a Puppet Deform Pin to Create a Starting Keyframe .....63

d. Creating an Ending Keyframe and a Motion Path .....65

e. Using a Mask or the Roto Brush Tool to Select a Set of Pixels to be Animated .....68

i. Creating a Mask Using the "Pen" Tool.....68

ii. Creating a Mask Using the "Auto-Trace" Function.....71

iii. Creating a Matte Using the "Roto Brush" Tool.....73

f. Looping the Animation.....81

g. Final Resulting Animation.....82

h. Animating with Multiple Deform Pins.....87

3. Independent Claim 1 .....98

a. [1pre]: A computer system providing, to a client computing device, software for automating a shifting of pixels within a video file, the computer system comprising: one or more processors; and one or more computer-readable media having stored thereon executable instructions that are transmitted to the client computing device for execution by one or more client processors on the client computing device, the executable instructions comprising instructions that when executed by the one or more client processors configure the client computing device to perform at least the following.....98

b. [1a]: access, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within a frame of the digital image file. .....99

c. [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 a first image frame. .....101

d. [1c]: 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. .....107

e. [1d]: create a first digital link between the first starting point and the first ending point, wherein the first digital 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.....108

f. [1e]: identify a first set of pixels that lie along the first digital link between the first starting point and the first ending point; and.....110

g. [1f]: shift the first set of pixels in the first direction. .....115

4. Claim 2: The computer system of claim 1, wherein the first ending portion comprises a particular portion of the first image frame.....117

5. Claim 3: The computer system of claim 1, wherein the digital image file comprises a video file and the first image frame comprises a first video frame of the video file. .....117

6. Claim 4: The computer system of claim 3, wherein the first ending portion comprises a particular portion of a second video frame within the video file.....117

7. Claim 8: The computer system of claim 1, wherein shifting the first set of pixels comprises rendering in a loop the first set of pixels being shifted within the first image frame. .....118

8. Claim 9 .....119

a. [9a]-[9e]: The computer system of claim 1, wherein the executable instructions include instructions that are executable to configure the computer system to: receive a second starting point through the user interface, wherein the second starting point is received through a user selection of a second beginning portion of the first image frame; receive a second ending point through the user interface, wherein the second ending point is received through a user selection of a second ending portion; create a second digital link between the second starting point and the second ending point, wherein the second digital link comprises: a second direction extending from the second starting point to the second ending point; and a second length between the second starting point and the second ending point; identify a second set of pixels that lie between the second starting point and the second ending point; and shift the second set of pixels in the second direction. ..... 119

9. Claim 10: The computer system of claim 9, wherein the first direction is different from the second direction. ..... 122

10. Claim 11: The computer system of claim 9, wherein a 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. ..... 122

11. Independent Claim 12 ..... 125

a. [12pre]: A computer program product comprising one or more non-transitory computer storage media having stored thereon computer-executable instructions that, when transmitted to a remote computer system for execution at a processor, cause the remote computer system to perform a method for automating a shifting of pixels within an image file, the method comprising. ..... 125

b. [12a]: 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 portion of a first image frame. .....126

c. [12b]: receiving, through the user interface, a first direction associated with the first starting point.....126

d. [12c]: create a first digital link extending in the first direction from the first starting point. .....127

e. [12d]: selecting a first set of pixels that are along the first digital link and extend in the first direction away from the first starting point; and. .....127

f. [12e]: shifting the first set of pixels, in the first image frame, in the first direction. .....128

12. Claim 13 .....128

a. [13a]-[13b]: The computer program product as recited in claim 12, further comprising receiving an indication to generate a first mask over a second portion of the first image frame, wherein pixels under the first mask are prevented from shifting. .....128

13. Claim 14: The computer program product as recited in claim 13, further comprising computer-executable instructions that, when transmitted to the remote computer system for execution at the processor, cause the remote computer system to perform a method for automating the shifting of pixels within the image file, the method comprising receiving through a user interface a selection of the second portion of the first image frame around which the first mask should be generated. .....132

14. Claim 15 .....133

a. [15a]-[15b]: The computer program product of claim 14, further comprising computer-executable instructions that, when transmitted to the remote computer system for execution at the processor, cause the remote computer system to perform a method for automating the shifting of pixels within the image file, the method comprising: identifying one or more edges that form a first boundary around the second portion; and generating the first mask to cover area within the first boundary. .....133

15. Independent Claim 19 .....136

a. [19pre]: A method for transmitting to a client computing device instructions for shifting pixels within a video file, comprising: transmitting computer executable instructions to a client computing device, the computer executable instructions configured to cause the client computing device to.....136

b. [19a]: access, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within a frame of the digital image file. .....137

c. [19b]: 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 a first image frame. .....137

d. [19c]: 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. .....137

e. [19d]: create a first digital link between the first starting point and the first ending point, wherein the first digital 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.....138

f. [19e]: identify a first set of pixels that lie along the first digital link between the first starting point and the first ending point; and. ..... 138

g. [19f]: shift the first set of pixels in the first direction. ..... 138

16. Claim 20: The method of claim 19, wherein the digital image file comprises a video file and the first image frame comprises a frame of the video file. ..... 138

C. Ground 2: IMU and Okabe, and Claims 1-4, 8-14, and 19-20 ..... 138

1. Summary of IMU ..... 138

a. Applying Effects and Creating an Animation ..... 140

b. Extracting a Single Frame of a GIF Animation ..... 143

c. Modifying Transparency of Certain Pixels Using a Matte ..... 145

2. Summary of Okabe ..... 146

3. The IMU-Okabe Combination ..... 147

a. Motivation to Combine IMU with Okabe ..... 147

b. Resulting Combination of IMU with Okabe ..... 151

4. Independent Claim 1 ..... 153

a. [1pre]: A computer system providing, to a client computing device, software for automating a shifting of pixels within a video file, the computer system comprising: one or more processors; and one or more computer-readable media having stored thereon executable instructions that are transmitted to the client computing device for execution by one or more client processors on the client computing device, the executable instructions comprising instructions that when executed by the one or more client processors configure the client computing device 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 a frame of the digital image file. .....154

c. [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 a first image frame. .....155

d. [1c]: 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. .....158

e. [1d]: create a first digital link between the first starting point and the first ending point, wherein the first digital 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. .....159

f. [1e]: identify a first set of pixels that lie along the first digital link between the first starting point and the first ending point; and. .....160

g. [1f]: shift the first set of pixels in the first direction. .....162

5. Claim 2: The computer system of claim 1, wherein the first ending portion comprises a particular portion of the first image frame. .....163

6. Claim 3: The computer system of claim 1, wherein the digital image file comprises a video file and the first image frame comprises a first video frame of the video file. .....163

7. Claim 4: The computer system of claim 3, wherein the first ending portion comprises a particular portion of a second video frame within the video file. .....163

8. Claim 8: The computer system of claim 1, wherein shifting the first set of pixels comprises rendering in a loop the first set of pixels being shifted within the first image frame. .....164

9. Claim 9 .....165

a. [9a]-[9e]: The computer system of claim 1, wherein the executable instructions include instructions that are executable to configure the computer system to: receive a second starting point through the user interface, wherein the second starting point is received through a user selection of a second beginning portion of the first image frame; receive a second ending point through the user interface, wherein the second ending point is received through a user selection of a second ending portion; create a second digital link between the second starting point and the second ending point, wherein the second digital link comprises: a second direction extending from the second starting point to the second ending point; and a second length between the second starting point and the second ending point; identify a second set of pixels that lie between the second starting point and the second ending point; and shift the second set of pixels in the second direction. ..... 165

10. Claim 10: The computer system of claim 9, wherein the first direction is different from the second direction. ..... 168

11. Claim 11: The computer system of claim 9, wherein a 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. ..... 169

12. Independent Claim 12 ..... 171

a. [12pre]: A computer program product comprising one or more non-transitory computer storage media having stored thereon computer-executable instructions that, when transmitted to a remote computer system for execution at a processor, cause the remote computer system to perform a method for automating a shifting of pixels within an image file, the method comprising. ..... 171

b. [12a]: 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 portion of a first image frame. .....172

c. [12b]: receiving, through the user interface, a first direction associated with the first starting point.....172

d. [12c]: creating a first digital link extending in the first direction from the first starting point. .....172

e. [12d]: selecting a first set of pixels that are along the first digital link and extend in the first direction away from the first starting point. .....173

f. [12e]: shifting the first set of pixels, in the first image frame, in the first direction. .....173

13. Claim 13 .....173

a. [13a]-[13b]: The computer program product as recited in claim 12, further comprising receiving an indication to generate a first mask over a second portion of the first image frame, wherein pixels under the first mask are prevented from shifting. .....173

14. Claim 14: The computer program product as recited in claim 13, further comprising computer-executable instructions that, when transmitted to the remote computer system for execution at the processor, cause the remote computer system to perform a method for automating the shifting of pixels within the image file, the method comprising receiving through a user interface a selection of the second portion of the first image frame around which the first mask should be generated. .....175

15. Independent Claim 19 .....175

a. [19pre]: A method for transmitting to a client computing device instructions for shifting pixels within a video file, comprising: transmitting computer executable instructions to a client computing device, the computer executable instructions configured to cause the client computing device to.....175

b. [19a]: access, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within a frame of the digital image file. .....176

c. [19b]: 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 a first image frame. .....176

d. [19c]: 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. .....176

e. [19d]: create a first digital link between the first starting point and the first ending point, wherein the first digital 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

f. [19e]: identify a first set of pixels that lie along the first digital link between the first starting point and the first ending point; and. .....177

g. [19f]: shift the first set of pixels in the first direction. .....177

16. Claim 20: The method of claim 19, wherein the digital image file comprises a video file and the first image frame comprises a frame of the video file. .....177

D. Ground 3: IMU, Okabe, and Li, and Claims 13-15 .....177

1. Summary of Li .....177

2. The IMU-Okabe-Li Combination .....179

a. Motivation to Combine the IMU-Okabe Combination with Li .....179

b. Resulting Combination of the IMU-Okabe Combination and Li .....182

3. Claim 13 .....183

a. [13a]-[13b]: The computer program product as recited in claim 12, further comprising receiving an indication to generate a first mask over a second portion of the first image frame, wherein pixels under the first mask are prevented from shifting. .....183

4. Claim 14: The computer program product as recited in claim 13, further comprising computer-executable instructions that, when transmitted to the remote computer system for execution at the processor, cause the remote computer system to perform a method for automating the shifting of pixels within the image file, the method comprising receiving through a user interface a selection of the second portion of the first image frame around which the first mask should be generated. .....184

5. Claim 15 .....185

a. [15a]-[15b]: The computer program product of claim 14, further comprising computer-executable instructions that, when transmitted to the remote computer system for execution at the processor, cause the remote computer system to perform a method for automating the shifting of pixels within the image file, the method comprising: identifying one or more edges that form a first boundary around the second portion; and generating the first mask to cover area within the first boundary. .....185

IX. CONCLUSION.....186

## LIST OF DOCUMENTS CONSIDERED

| Exhibit | Shorthand | Description |
|---|---|---|
| 1001 | ’641 Patent | U.S. Patent No. 11,182,641 |
| 1003 | AEM | Adobe® After Effects® Help and tutorials, Adobe (2013) |
| 1004 | IMU-Home | Wayback Machine Capture dated Mar. 27, 2012 of Anthony Thyssen, *Examples of ImageMagick Usage (Version 6)*, 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, *ImageMagick v6 Examples -- Distorting Images*, 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, *ImageMagick v6 Examples -- Masks*, 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, *ImageMagick v6 Examples -- Animation Basics*, 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, *ImageMagick v6 Examples -- Usage under Windows*, 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 | ’641 PH | Prosecution history of U.S. Patent No. 11,182,641 |
| 1012 | Archive | Declaration of Nathaniel E Frank-White of the Internet Archive |
| 1022 | | Plaintiffs’ Opposed Motion for Leave to Serve Second Supplemental Disclosures Pursuant to PR. 3-1 and P.R. 3-2, *Plotagraph, Inc. v. Lightricks Ltd.*, Civil Action No. 4:21-cv-03873, Dkt. No. 42 (S.D. Tex. May 21, 2022) |
| 1023 | Hair | U.S. Patent No. 6,014,491 |
| 1024 | Nakagawa | U.S. Patent No. 5,835,911 |
| 1025 | | WayBack Machine Capture dated Sept. 7, 2012 of *Downloads*, Adobe (Sept. 7, 2012), http://www.adobe.com/downloads/ [https://web.archive.org/web/20120907130453/http://www.adobe.com/downloads/] |
| 1026 | | WayBack Machine Capture dated Mar. 28, 2012 of *Download ImageMagick*, ImageMagick (Mar. 28, 2012), http://www.imagemagick.org/script/download.php [https://web.archive.org/web/20120328075001/http://www.imagemagick.org/script/download.php] |

## CLAIM LISTING

| Claim | Limitation No. | Limitation |
|---|---|---|
| 1 | [1pre] | A computer system providing, to a client computing device, software for automating a shifting of pixels within a video file, the computer system comprising:<br><br>one or more processors; and<br><br>one or more computer-readable media having stored thereon executable instructions that are transmitted to the client computing device for execution by one or more client processors on the client computing device, the executable instructions comprising instructions that when executed by the one or more client processors configure the client computing device 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 a frame of the digital image file; |
| | [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 a first image frame; |
| | [1c] | 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; |
| | [1d] | create a first digital link between the first starting point and the first ending point, wherein the first digital link comprises:<br><br>a first direction extending from the first starting point to the first ending point; and<br><br>a first length between the first starting point and the first ending point; |
| | [1e] | identify a first set of pixels that lie along the first digital link between the first starting point and the first ending point; and |

| Claim | Limitation No. | Limitation |
|---|---|---|
| | [1f] | shift the first set of pixels in the first direction. |
| 2 | The computer system of claim 1, wherein the first ending portion comprises a particular portion of the first image frame. | |
| 3 | The computer system of claim 1, wherein the digital image file comprises a video file and the first image frame comprises a first video frame of the video file. | |
| 4 | The computer system of claim 3, wherein the first ending portion comprises a particular portion of a second video frame within the video file. | |
| 8 | The computer system of claim 1, wherein shifting the first set of pixels comprises rendering in a loop the first set of pixels being shifted within the first image frame. | |
| 9 | [9a] | The computer system of claim 1, wherein the executable instructions include instructions that are executable to configure the computer system to:<br><br>receive a second starting point through the user interface, wherein the second starting point is received through a user selection of a second beginning portion of the first image frame; |
| | [9b] | receive a second ending point through the user interface, wherein the second ending point is received through a user selection of a second ending portion; |
| | [9c] | create a second digital link between the second starting point and the second ending point, wherein the second digital link comprises:<br><br>a second direction extending from the second starting point to the second ending point; and<br><br>a second length between the second starting point and the second ending point; |
| | [9d] | identify a second set of pixels that lie between the second starting point and the second ending point; and |
| | [9e] | shift the second set of pixels in the second direction. |

| Claim | Limitation No. | Limitation |
|---|---|---|
| 10 | The computer system of claim 9, wherein the first direction is different from the second direction. | |
| 11 | The computer system of claim 9, wherein a 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. | |
| 12 | [12pre] | A computer program product comprising one or more non-transitory computer storage media having stored thereon computer-executable instructions that, when transmitted to a remote computer system for execution at a processor, cause the remote computer system to perform a method for automating a shifting of pixels within an image file, the method comprising: |
| | [12a] | 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 portion of a first image frame; |
| | [12b] | receiving, through the user interface, a first direction associated with the first starting point; |
| | [12c] | creating a first digital link extending in the first direction from the first starting point; |
| | [12d] | selecting a first set of pixels that are along the first digital link and extend in the first direction away from the first starting point; and |
| | [12e] | shifting the first set of pixels, in the first image frame, in the first direction. |
| 13 | [13a] | The computer program product as recited in claim 12, further comprising receiving an indication to generate a first mask over a second portion of the first image frame, |
| | [13b] | wherein pixels under the first mask are prevented from shifting. |
| 14 | The computer program product as recited in claim 13, further comprising computer-executable instructions that, when transmitted to the remote computer system for execution at the processor, cause the remote computer system to perform a method for automating the shifting of pixels within the image file, the method comprising | |

| Claim | Limitation No. | Limitation |
|---|---|---|
| | receiving through a user interface a selection of the second portion of the first image frame around which the first mask should be generated. | |
| 15 | [15a] | The computer program product of claim 14, further comprising computer-executable instructions that, when transmitted to the remote computer system for execution at the processor, cause the remote computer system to perform a method for automating the shifting of pixels within the image file, the method comprising:<br><br>identifying one or more edges that form a first boundary around the second portion; and |
| | [15b] | generating the first mask to cover area within the first boundary. |
| 19 | [19pre] | A method for transmitting to a client computing device instructions for shifting pixels within a video file, comprising:<br><br>transmitting computer executable instructions to a client computing device, the computer executable instructions configured to cause the client computing device to: |
| | [19a] | access, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within a frame of the digital image file; |
| | [19b] | 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 a first image frame; |
| | [19c] | 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; |
| | [19d] | create a first digital link between the first starting point and the first ending point, wherein the first digital link comprises: |

| Claim | Limitation No. | Limitation |
|---|---|---|
| | | a first direction extending from the first starting point to the first ending point; and<br><br>a first length between the first starting point and the first ending point; |
| | [19e] | identify a first set of pixels that lie along the first digital link between the first starting point and the first ending point; and |
| | [19f] | shift the first set of pixels in the first direction. |
| 20 | The method of claim 19, wherein the digital image file comprises a video file and the first image frame comprises a frame of the video file. | |

## 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. 11,182,641 (the "'641 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 claims 1-4, 8-15, and 19-20 of the '641 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 '641 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 understood how to implement such systems. The '641 Patent and 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 ’641 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 ’641 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:

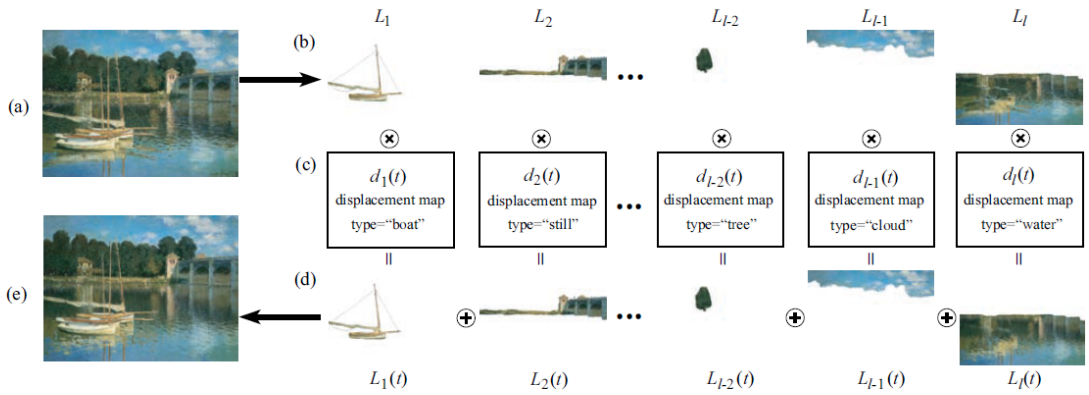



**Figure 2** Overview of our system. The input still image (a) is manually segmented into several layers (b). Each layer $L_i$ is then animated with a different stochastic motion texture $d_i(t)$ (c). Finally, the animated layers $L_i(t)$ (d) are composited back together to produce the final animation $I(t)$ (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. Image processing on moving images (videos) began in the 1960s. For example, "Digital Video-Data Handling" (NASA JPL Technical Report No. 32-877, January 5, 1966) describes an analog television signal that is converted to a digital file, processed, and then converted once again into an analog video. By the 1980s,

such systems were available commercially. "The Coming Revolution in Interactive Digital Video" (Fox, Communications of the ACM, July 1989) describes digital video systems produced for consumers, typically based on optical disks such as CD-ROM and notes that "image processing techniques can enhance individual frames" (page 796).

41. One popular system that I personally observed in the late 1980s was sold to television stations by Avid Technology. *See., e.g.*, U.S. Patent No. 4,970,633 (filed in 1989). This patent describes storing digital video in "a standard PC file system" (2:7-15), compressing, decompressing, and displaying video data in real time (4:13-14), and digital effects, "as in a television newscast when an overlaid image in a corner of the screen expands in size to take up the entire screen" (4:52-55).

42. The capabilities of Avid's proprietary system became available to consumers in the 1990s with programs such as Adobe Premiere (1991).[6]

43. Because, like film, video gives the illustration of motion by presenting individual frames in succession, almost any system that had the capability of digitizing and processing video (multiple frames) also had the capability of

---

[6] *See* "Adobe Premiere brings digital video capabilities to the desktop" (Business Wire, December 13, 1991)

extracting an individual frame. In fact, the hardware for digitizing video was typically marketed as a "frame grabber." This capability is described explicitly in "Picture processor breaks new ground," *Electronics Times* (January 30, 1986): "The frame grabber board digitises a video image from a camera or tape into 256 x 256 pixel image with up to 64 grey scale levels. It has a 64kbyte memory which ***can be used to store, or 'photograph', a particular frame for analysis***" (emphasis added).

44. Our modern world of distributing software over a network in which the client machine checks for updates, and downloads updates from a network is described in Nakagawa (Abstract):

> A number of sets of software may be systematically distributed and maintained via a network connecting many vendors and users of client/server software. A client program in a user computer detects when software subject to maintenance is activated and transmits an inquiry over the network to the software vendor's computer for information on the current version of the software. The server program compares data in the inquiry with data relating to the latest version of the software and returns update instruction information and updated software if appropriate. The client program automatically updates the software to the latest version according to the update instruction information when it is received.

45. Any person of ordinary skill by the ECPD would have been familiar with software being provided by a server to a client computing device, such as a desktop personal computer or a mobile phone.

46. In summary, graphical user interfaces, algorithms, and software for adding motion to a static image had been developed in the 1990s. All of the hardware and software tools necessary for obtaining that static image from an analog video signal or a digital video file were also available in the 1990s.

## VII. OVERVIEW OF THE '641 PATENT

47. The '641 Patent itself is directed to "systems, methods, and computer-readable media that automate the shifting of pixels within a digital video file." '641 Patent, 3:13-16. A user provides a video file and selects a "starting point" on a video frame, as well as an "ending point" on the same or a different frame. '641 Patent, Abst., 6:33-54. The system or user then creates a "digital link" between the starting and ending points and identifies a set of pixels that includes at least "a line of individual pixels extending from the starting point to the ending point" but may, "at another extreme," include "a relatively wide swatch of pixels that are parallel to the link that extends between the starting point and the ending point." '641 Patent, 6:58-67, 7:22-36. The set of pixels is then shifted in the link's direction. '641 Patent, 7:37-40. Figure 9 is representative:

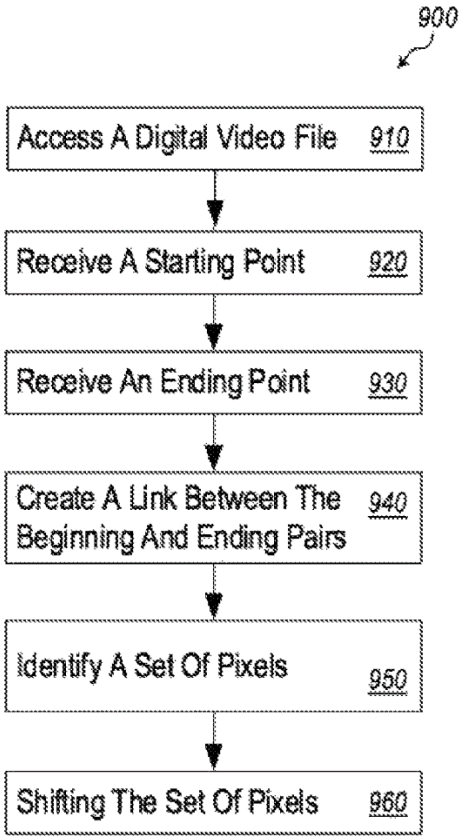



*Fig. 9*

'641 Patent, Fig. 9.

### A. Claim Construction

48. For purposes of this IPR, I apply the plain and ordinary meanings of all claim terms in the '641 Patent.

## VIII. ANALYSIS OF CLAIMS 1-4, 8-15, AND 19-20 OF THE '641 PATENT IN VIEW OF THE PRIOR ART

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

49. AEM is a user manual for the Adobe After Effects CS6 software ("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.

50. IMU contains Wayback Machine captures dated 2012 of the ImageMagick.org 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) 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).[7] 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.org website were each publicly available for viewing by IMV6 users on the ImageMagick.org website by the ECPD. I consulted the Internet Archive and confirmed that this was true through the following URLs of archived Wayback Machine captures of the ImageMagick.org 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.org 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

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

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.org 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.org 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.org 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.org 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.

51. 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.

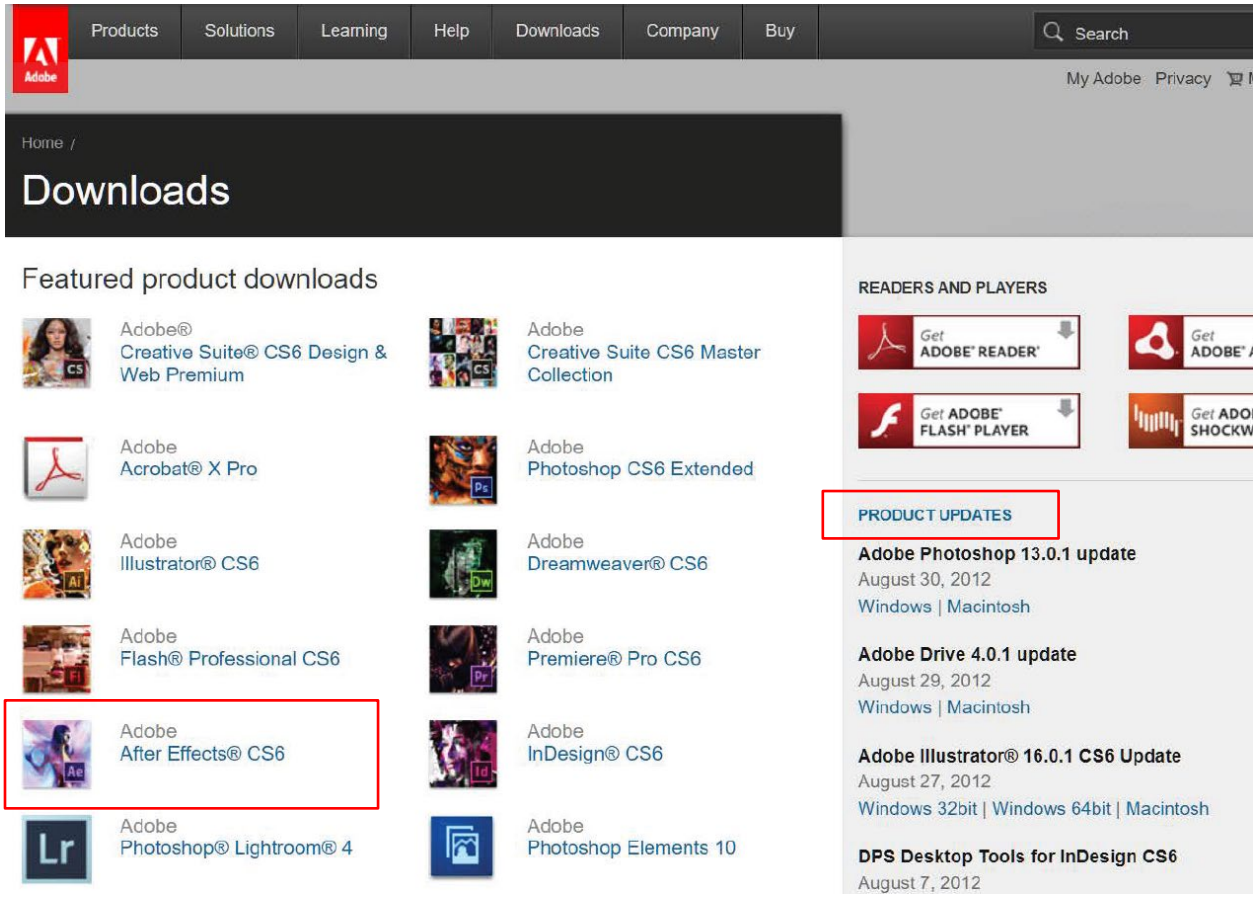
52. Li was published in Volume 23, Issue 3 of ACM Transactions on Graphics in August 2004 and was thus publicly available to those in the art by that date. Li, 1.

## B. Ground 1: AEM, and Claims 1-4, 8-15, and 19-20

### 1. Summary of AEM

53. AEM is a user manual for AECS6—software for performing a variety of animation tasks from, e.g., "animat[ing] a simple title" to "creat[ing] complex motion graphics, or composit[ing] realistic visual effects." AEM, 25.

54. AEM instructs the user to "[m]ake sure that you've installed the current version of [AECS6], including any available updates." AEM, 517. To view such updates, AEM instructs to "go to the Downloads section of the Adobe website," and provides a hyperlink for doing so. AEM, 517. As a POSITA would have known, clicking such a hyperlink would have directed the user to the "Downloads" webpage of the Adobe website shown in EX1025, from which the user could indeed download and install AECS6, including any updates:




EX1025 (annotations added); AEM, 517. In fact, I downloaded and installed Adobe Creative Suite, which included AECS6, onto my desktop computer via the "Downloads" webpage of Adobe's website by the ECPD. Further, I generated EX1025 by entering the hyperlinked "Downloads" webpage's URL into the Wayback Machine, and then selecting and printing out the Wayback Machine capture of the webpage captured on the same day as AEM's capture date of

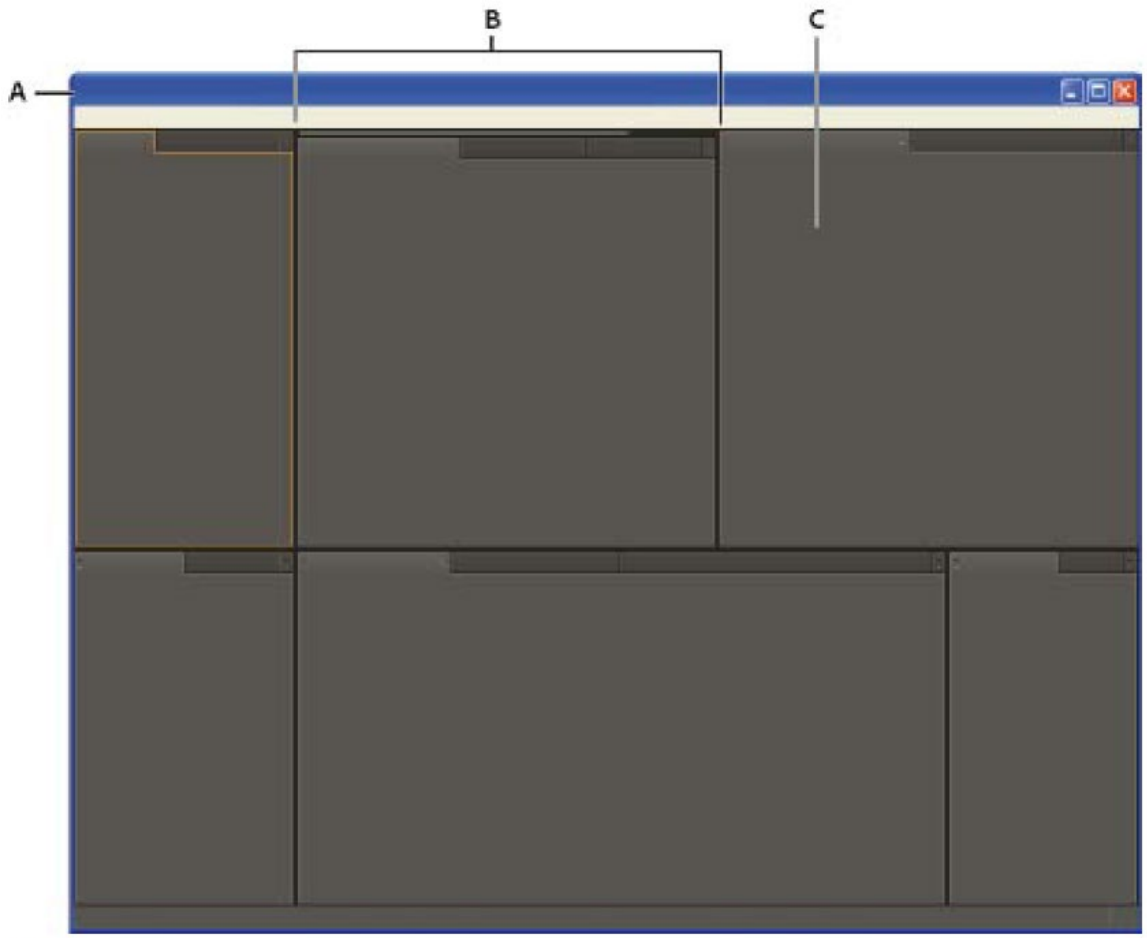
September 7, 2012.[8] *Compare* EX1025, *with* AEM, 517, *and* Archive, p. 98. *See* Archive, ¶5. Because these dates match, EX1025 thus shows the webpage that a user would have been directed to upon clicking the aforementioned hyperlink in AEM on that date. *See* Archive, ¶5.

55. AEM also explains how to navigate AECS6's workspace and panels, define "compositions" and "layers," extract a single frame of a video, apply a "mask" or "matte" to the frame, and animate the frame using the "Puppet" effect. Each is discussed herein.

**a. Workspace and Panels**

56. 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 in AECS6. *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:

[8] I provided counsel with this printout, which I understand is being used as an exhibit in this proceeding.




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

AEM, 39.

**b. Compositions and Layers**

57. AEM explains that a composition is the "framework" for animation in AECS6. AEM, 75. "Footage items," such as images and videos, are imported into a composition. AEM, 75-76; *see also* AEM, 101 ("Imported footage items appear in the Project panel."). AEM lists specific "[s]upported import formats" of such footage items, including for example "Still-image formats" such as JPEG, PNG, and PDF, and "Video and animation formats" such as MOV, MPEG, and animated GIF.

AEM, 98-101. Once imported, a footage item becomes a "layer" in that composition. AEM, 75, 98; *see also* AEM, 121 ("You can create a layer from any footage item in the Project panel…" by "[s]electing one or more footage items and folders in the Project panel" and "[d]rag[ging] the selected footage items to the Composition panel.").

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

59. 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. Extracting a Single Frame from a Video**

60. While AEM teaches creating a composition using a video (AEM, 75, 98-101), AEM also teaches extracting a "single frame" from such a composition and thus the video, which "is useful for," e.g., "exporting an image from a movie for posters or storyboards" (AEM, 590). To do so, AEM instructs to "[g]o to the frame that you want to export so that it is shown in the Composition panel." AEM, 590.

Then, "choose Composition > Save Frame As > File. Adjust settings in the Render Queue panel if necessary, and then click Render." AEM, 590.

61. AEM also teaches that the aforementioned "Render Queue panel" allows the user to manage various "render settings and output module settings" for the frame, including "output format." AEM, 572. "Supported output formats" include many of the same formats listed as "[s]upported import formats" for footage items discussed in Section VIII.B.1.b, such as JPEG, PNG, and other "Still-image formats." *Compare* AEM, 573-74, *with* AEM, 98-101. Indeed, AEM teaches importing the frame as a new footage item "by dragging its output module from the Render Queue panel into the Project panel." AEM, 574-75. Such provides "a convenient way to convert a footage item from one format to another," e.g., to extract a single frame from a video as a footage item for use as a new layer of the composition. AEM, 573-75, 98-101, 590.

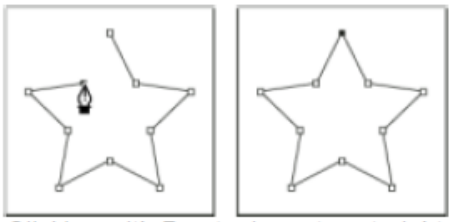
**d. Modifying Transparency of Certain Pixels**

62. AEM describes tools for modifying 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. Masks and mattes modify the layer's "alpha channel," which "determines the transparency of the layer at each pixel." AEM, 318.

63. 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. Masks are created 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 mask's path and vertices. AEM, 264-66. AEM depicts an exemplary star-shaped mask created using the Pen tool:



*Clicking with Pen tool creates straight segments.*

AEM, 265. Masks can also be automatically created using AECS6's "Auto-trace" function, which creates a mask by "searching for edges" across a 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.

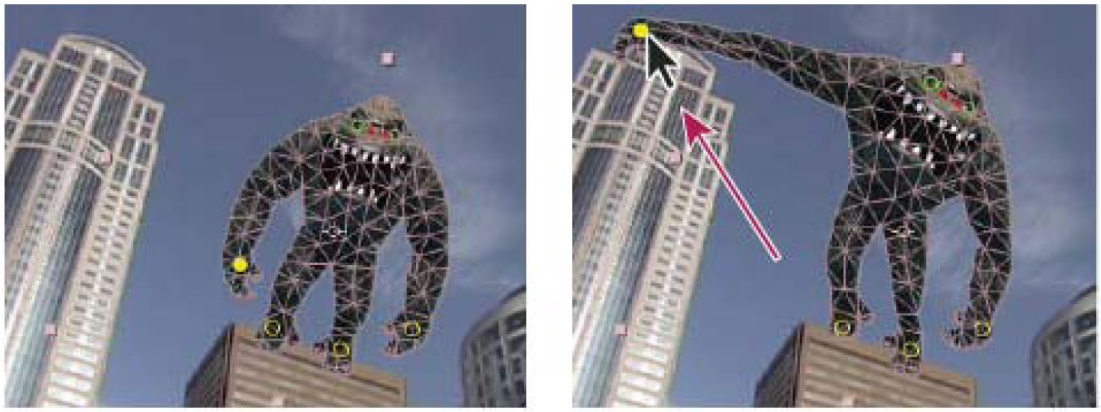
64. Mattes similarly "define[] the transparent areas" of a layer. AEM, 317. Specifically, a matte isolates a layer's "foreground" from its "background," making the latter transparent. AEM, 328. A matte is created by using, e.g., AECS6's "Roto Brush" tool to draw "strokes" in the "Layer panel" over "representative 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.

**e. Puppet Effect**

65. 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.

66. In one example, AEM depicts a foreground layer containing an image of a gorilla 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.").

**f. Animating with the Puppet Effect**

67. 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 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."[9]

68. 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 allows a user to specify the current animation time point). AECS6 then automatically creates the starting keyframe for the Deform pin based

---

[9] *See, e.g.*, *The Illusion Of Life: Disney Animation* (1981), "The Principles of Animation" chapter, available at https://archive.org/details/TheIllusionOfLifeDisneyAnimation/.

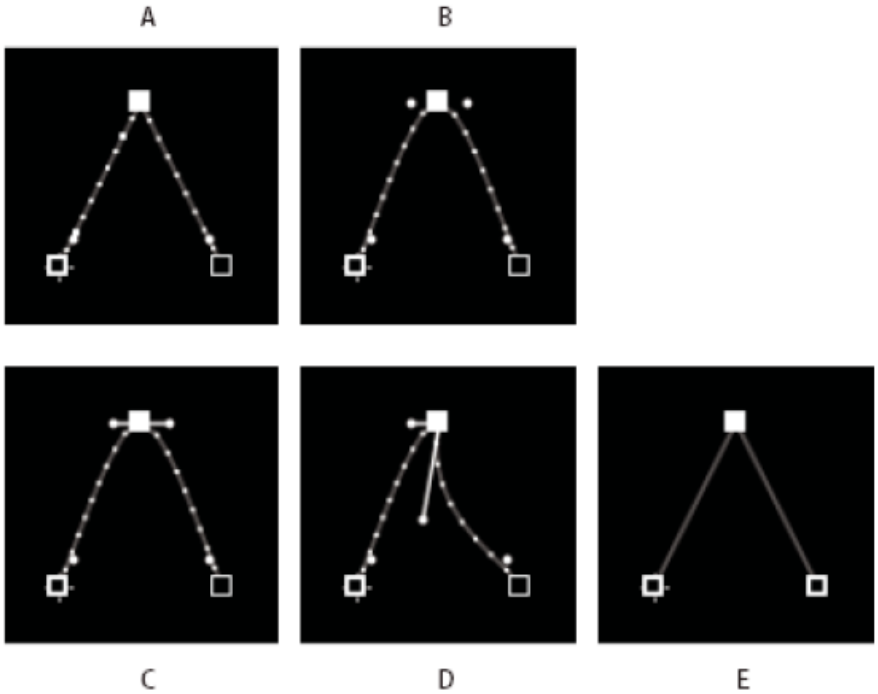
on both the specified time point and the Deform pin's starting position. AEM, 218-19.

69. 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.

70. 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.

71. 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:




*Motion path interpolation*

***A.** Linear **B.** Auto Bezier **C.** Continuous Bezier **D.** Bezier **E.** Hold*

AEM, 193.

72. 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

73. 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 a single frame from a video of a smokestack, where the frame has been animated using the features of AECS6 discussed above in Section VIII.B.1.[10]

---

[10] A nearly identical example of animating smoke in a single, still image using 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

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

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

---

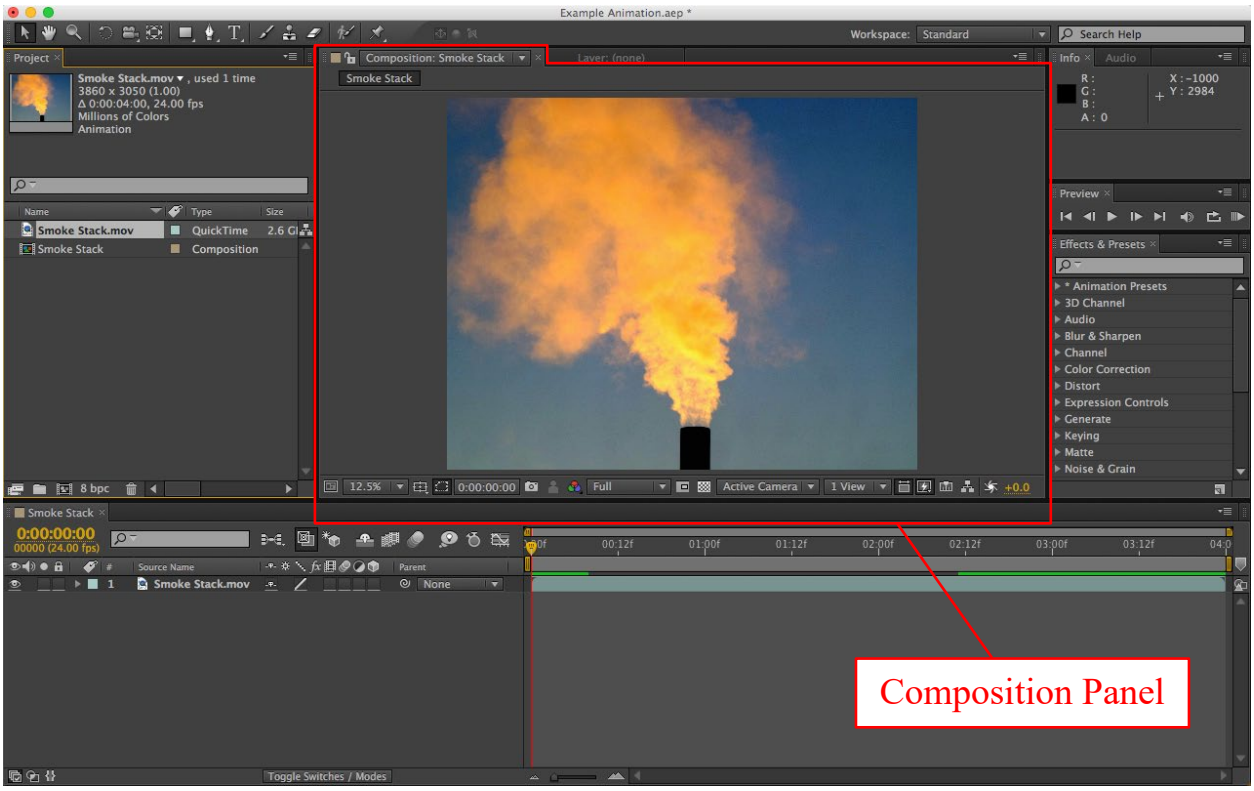
the ECPD) and uses a 2020 version of After Effects to animate the image rather than AECS6, the video may nevertheless be helpful to those unfamiliar with how such animations are created in AECS6, as AECS6 includes all of the same features of After Effects 2020 that are discussed in the video, e.g., the Puppet effect and masks.

76. 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 a Video into a Composition**

77. To begin, the below screenshot of AECS6's user interface shows a 4-second long MOV video of a smokestack billowing smoke that has been imported

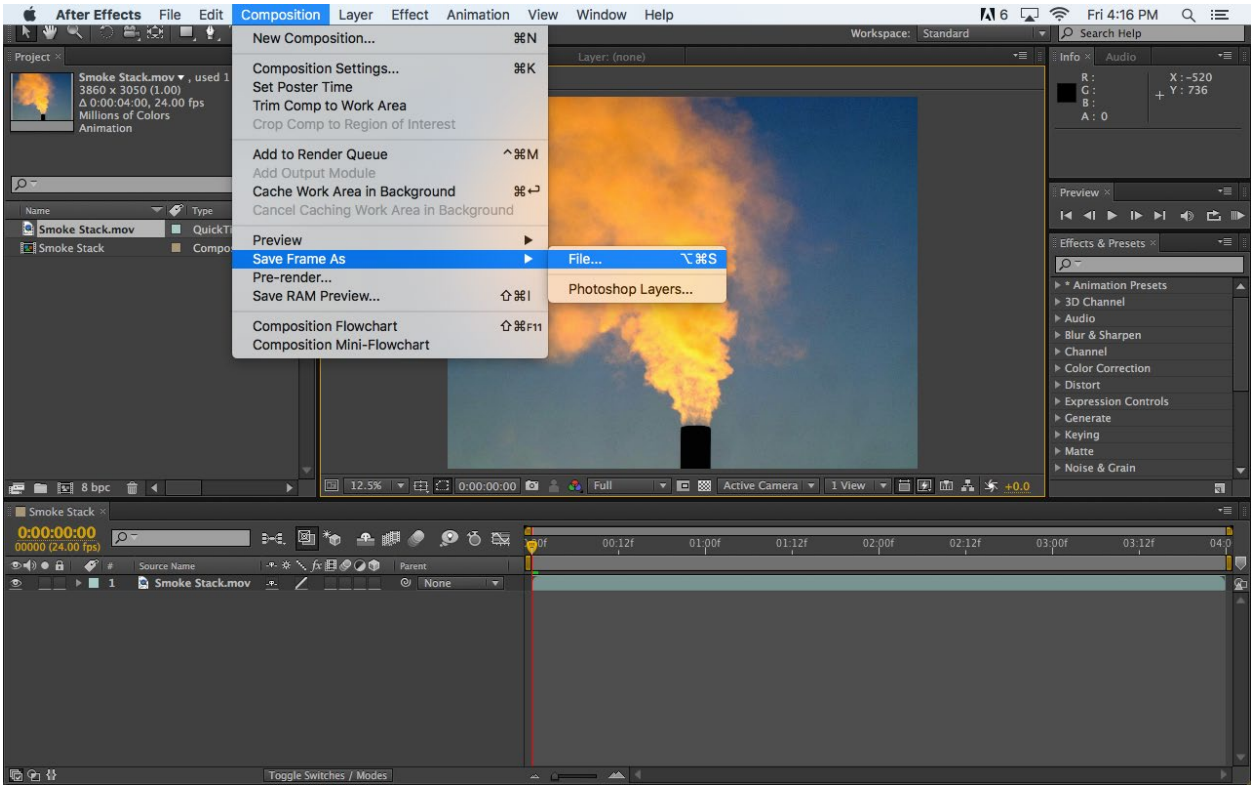
into an exemplary composition to form a layer in that composition.[11] *See* AEM, 75, 99. The center panel within the user interface (indicated in the annotations) is the "Composition Panel," which provides a preview of the layer within the composition and, as later screenshots also show, enables a layer to be manually modified using AECS6's effects. *See* AEM, 75, 120, 335.




[11] Attachment D shows the depicted frames of the MOV video at 1-second time intervals to mimic the playback of the MOV video over time. As shown, the smoke gradually billows upwards from the smoke stack, showing that the MOV video is indeed a video file.

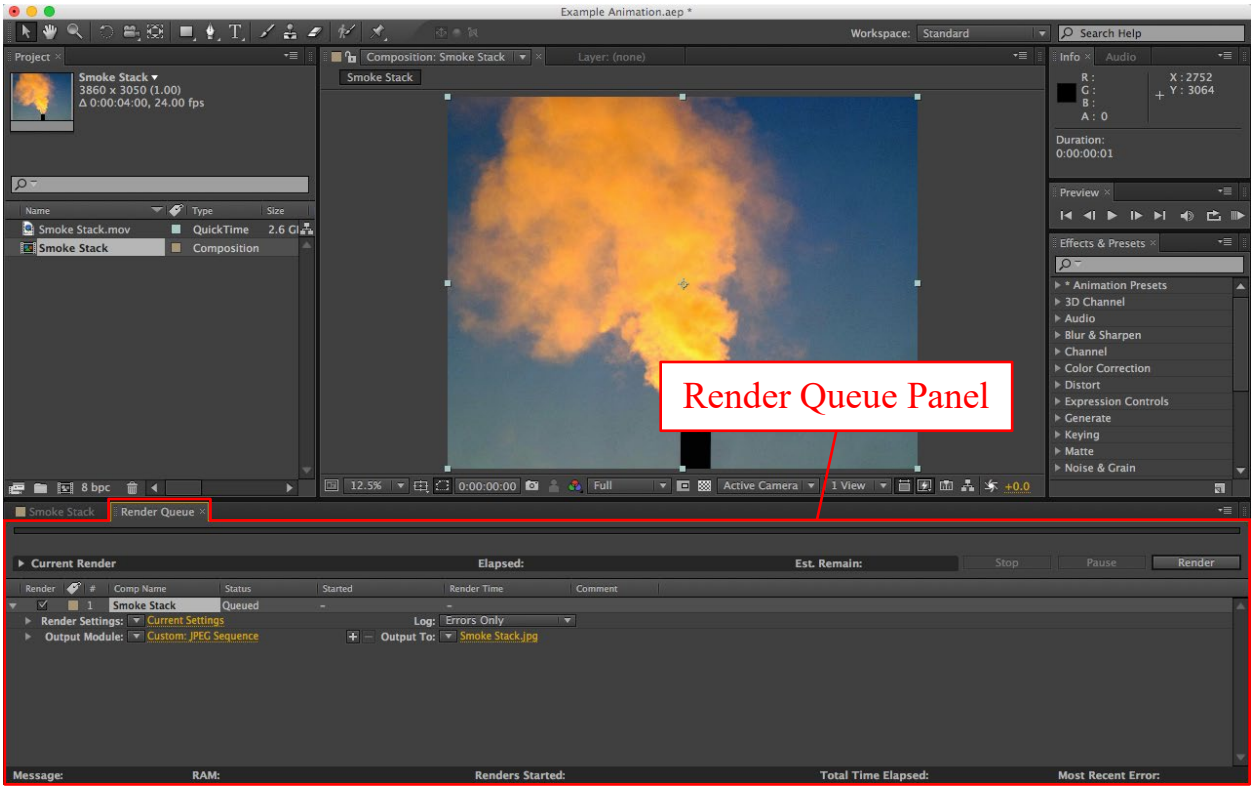
### b. Extracting a Single Frame from the Video and Importing the Frame as a Layer

78. As shown in the following screenshots, a single frame of the video may be extracted and used as a still-image footage item to be imported into the composition as a new layer. AEM, 590, 573-75, 98-101. The first screenshot below shows the selection of the first frame of the video—i.e., the frame at time "0:00:00:00," which is selected using the "current-time indicator (CTI)" in the "Timeline panel"—as the frame to be extracted. *See* AEM, 590, 166. The frame is then extracted by choosing "Composition > Save Frame As > File." AEM, 590.



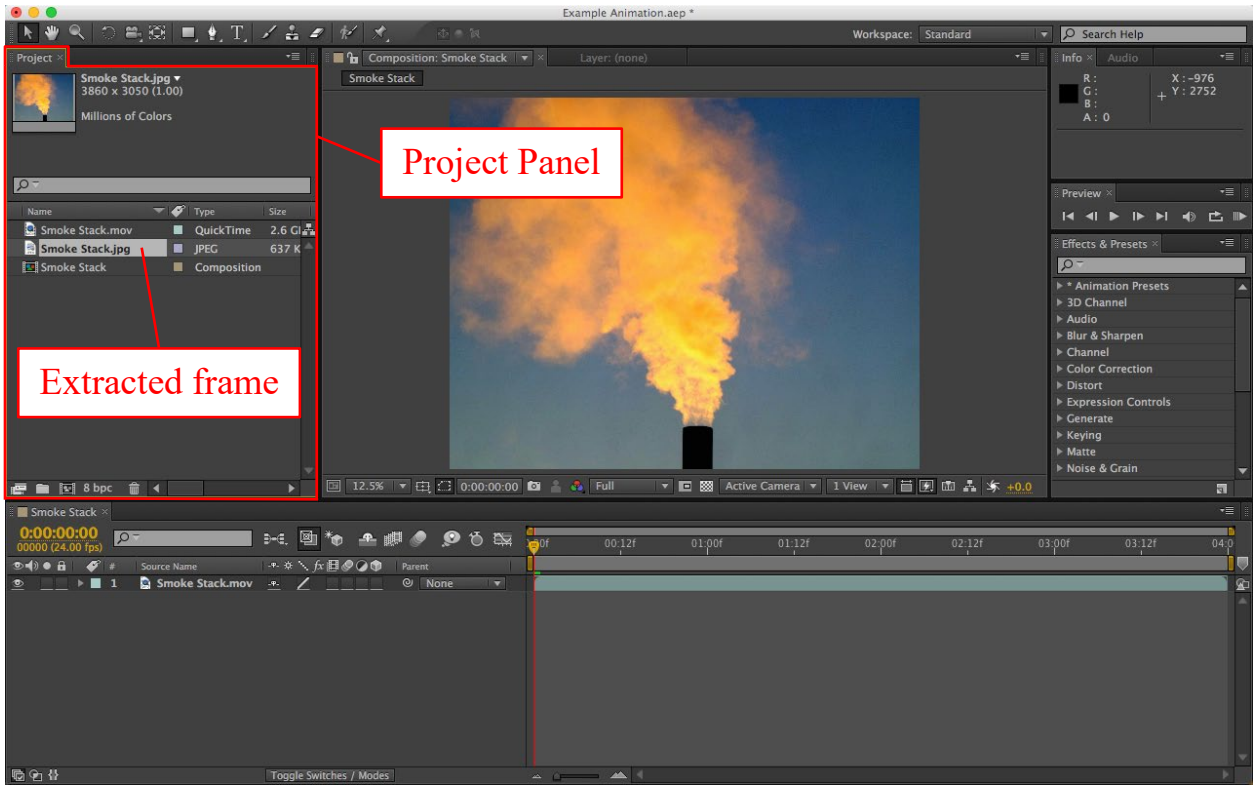

79. The next screenshot below shows the resulting "Render Queue panel" (indicated in the annotations), which allows the render and output module settings

for the frame to be managed. *See* AEM, 590, 572. The settings have been adjusted such that the output format of the frame is a JPEG—i.e., one of the "[s]till-image formats" that AECS6 supports as both an output format and a footage item import format. *See* AEM, 573-74, 98-101.

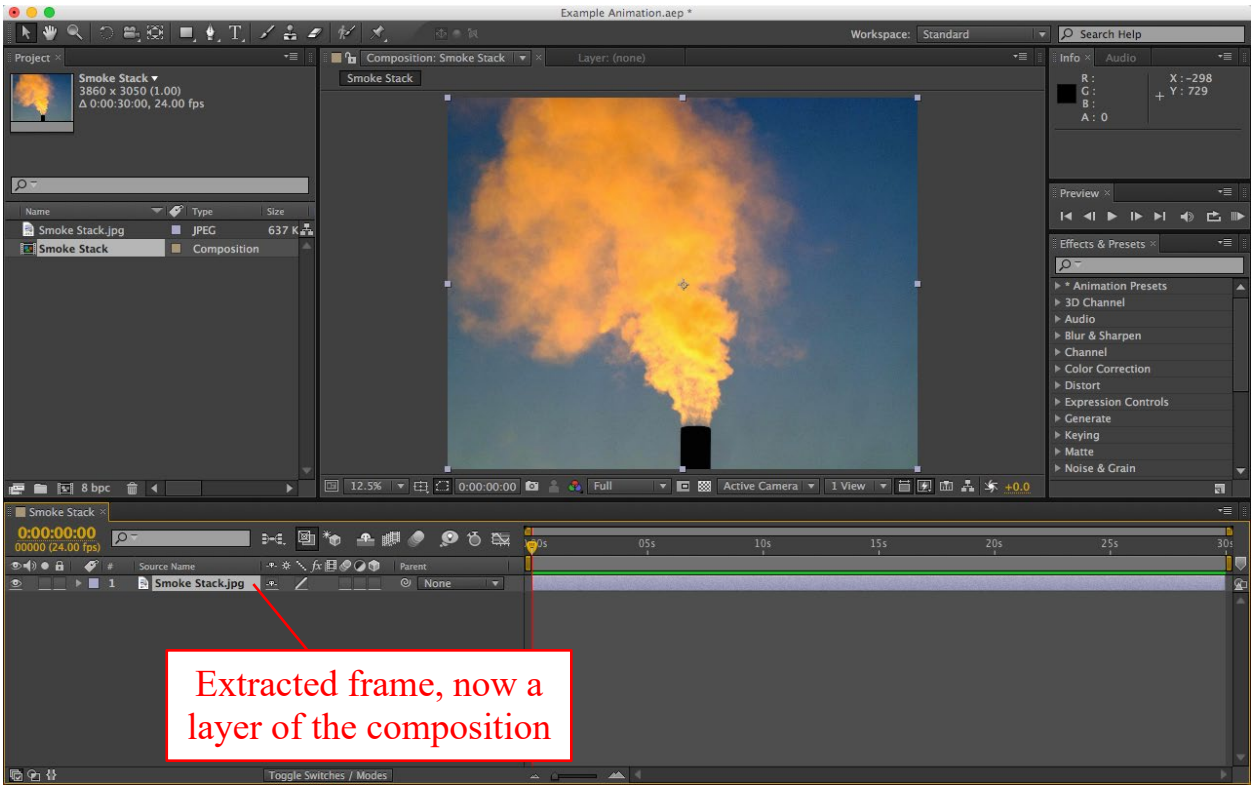



80. The frame is then extracted accordingly by clicking the "Render" button at the top-right of the Render Queue panel, as shown in the screenshot below. *See* AEM, 590.

81. From here, the frame may be imported into the composition by first dragging the frame's "Output Module" to the "Project panel" (indicated in the annotations below), which thus enables the frame to be used as a footage item, as shown in the following annotated screenshot. *See* AEM, 574-75; *see also* AEM, 90. The Render Output panel may also be closed, as it is no longer needed.

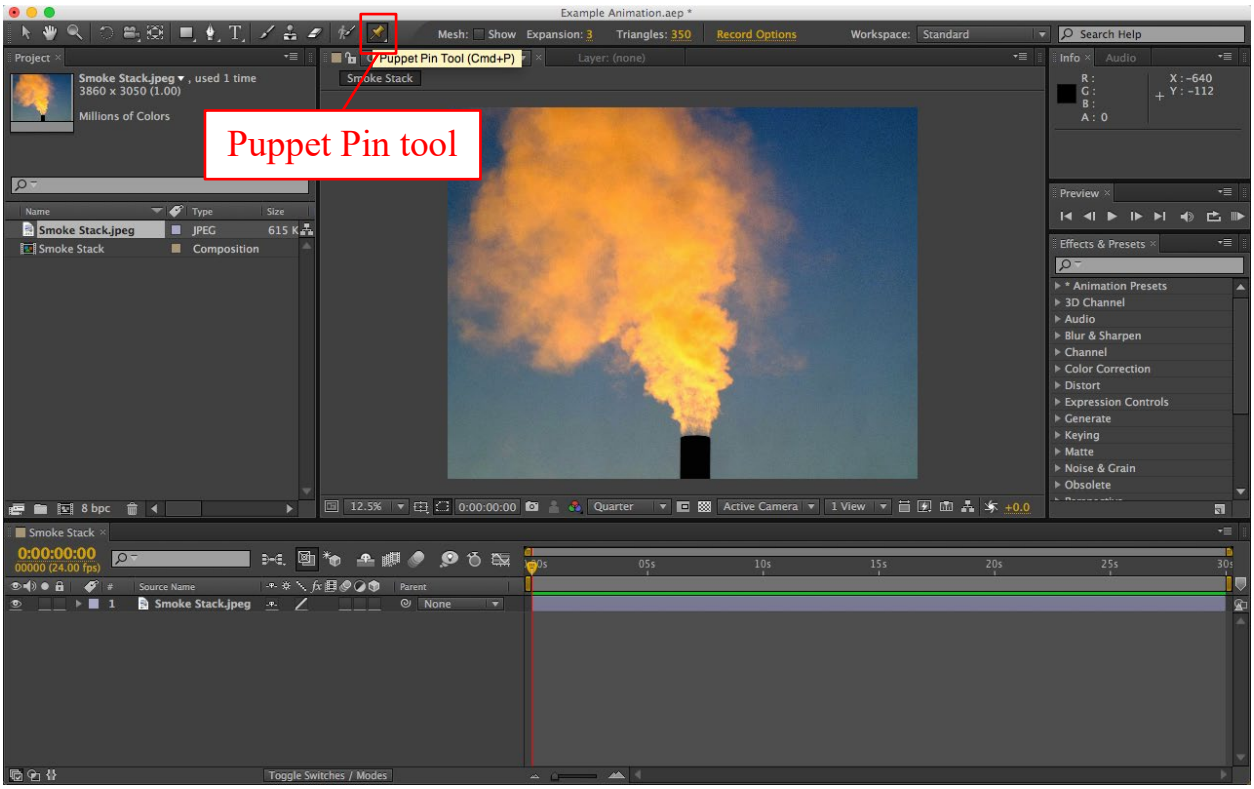


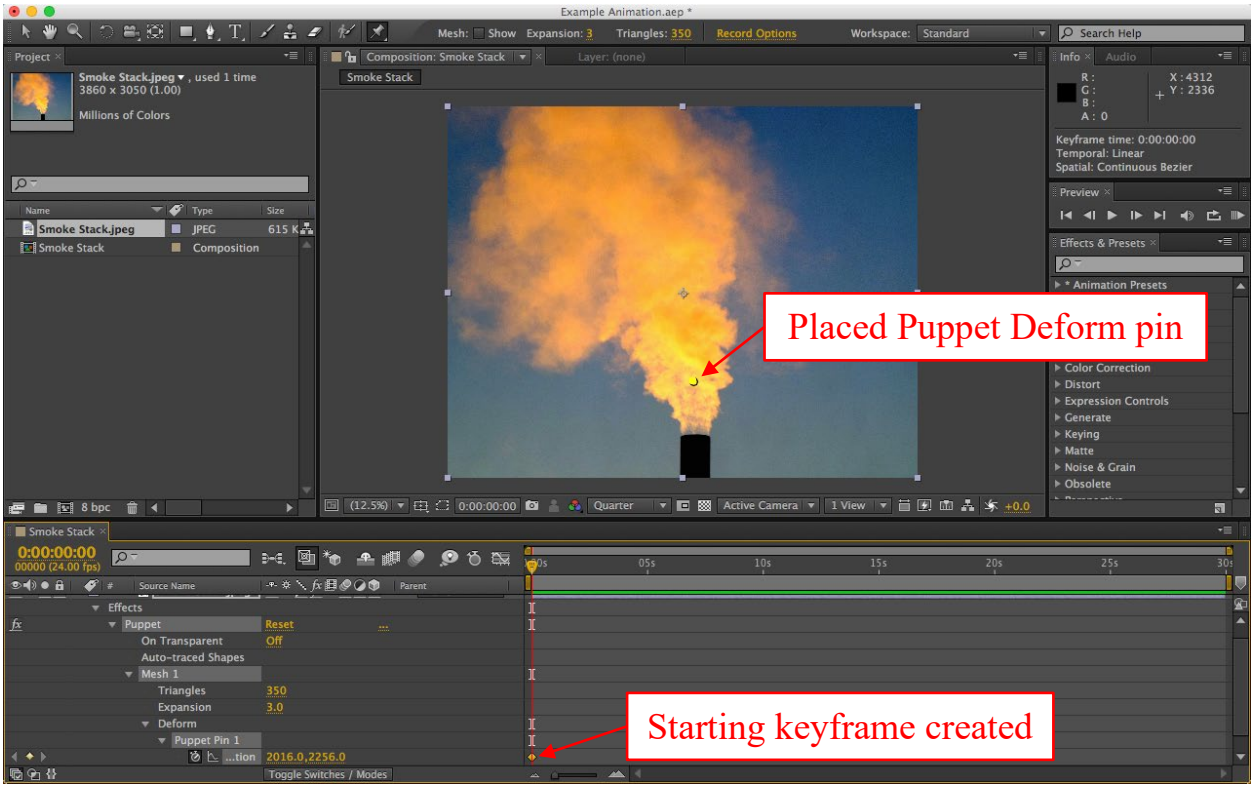

The extracted frame is then selected and dragged to the Composition panel, thus causing the frame to become a new layer of the composition, as shown in the below annotated screenshot. *See* AEM, 121, 75. For purposes of clarity, the original video is also deleted from the composition and Project panel as unnecessary for this example.



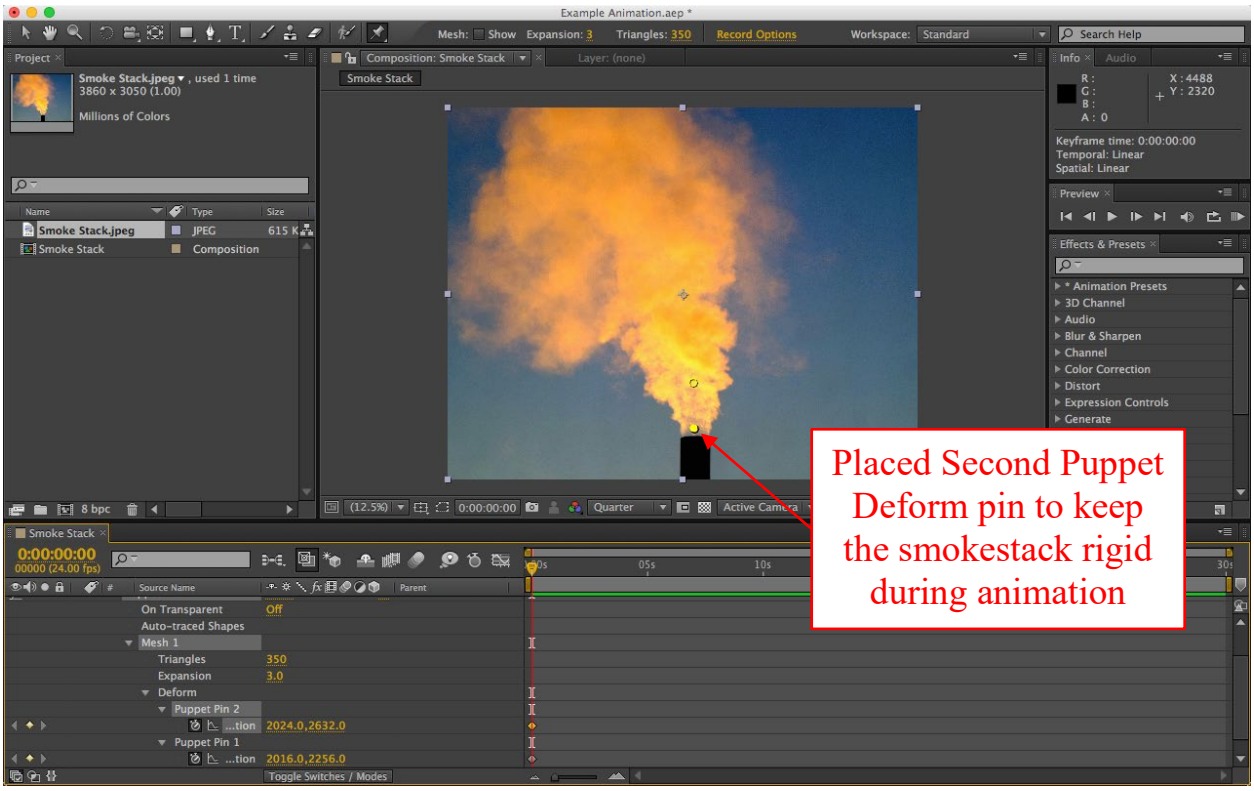

### c. Placing a Puppet Deform Pin to Create a Starting Keyframe

82. 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.


Example Animation.aep *
Puppet Pin Tool (Cmd+P)
Puppet Pin tool
Smoke Stack.jpeg, used 1 time
3860 x 3050 (1.00)
Millions of Colors
Smoke Stack.jpeg
Smoke Stack
Composition
Toggle Switches / Modes


Example Animation.aep *
Smoke Stack.jpeg, used 1 time
3860 x 3050 (1.00)
Millions of Colors
Keyframe time: 0:00:00:00
Temporal: Linear
Spatial: Continuous Bezier
Placed Puppet Deform pin
Starting keyframe created
Puppet
Reset
On Transparent
Off
Auto-traced Shapes
Mesh 1
Triangles
350
Expansion
3.0
Deform
Puppet Pin 1
2016.0,2256.0
Toggle Switches / Modes

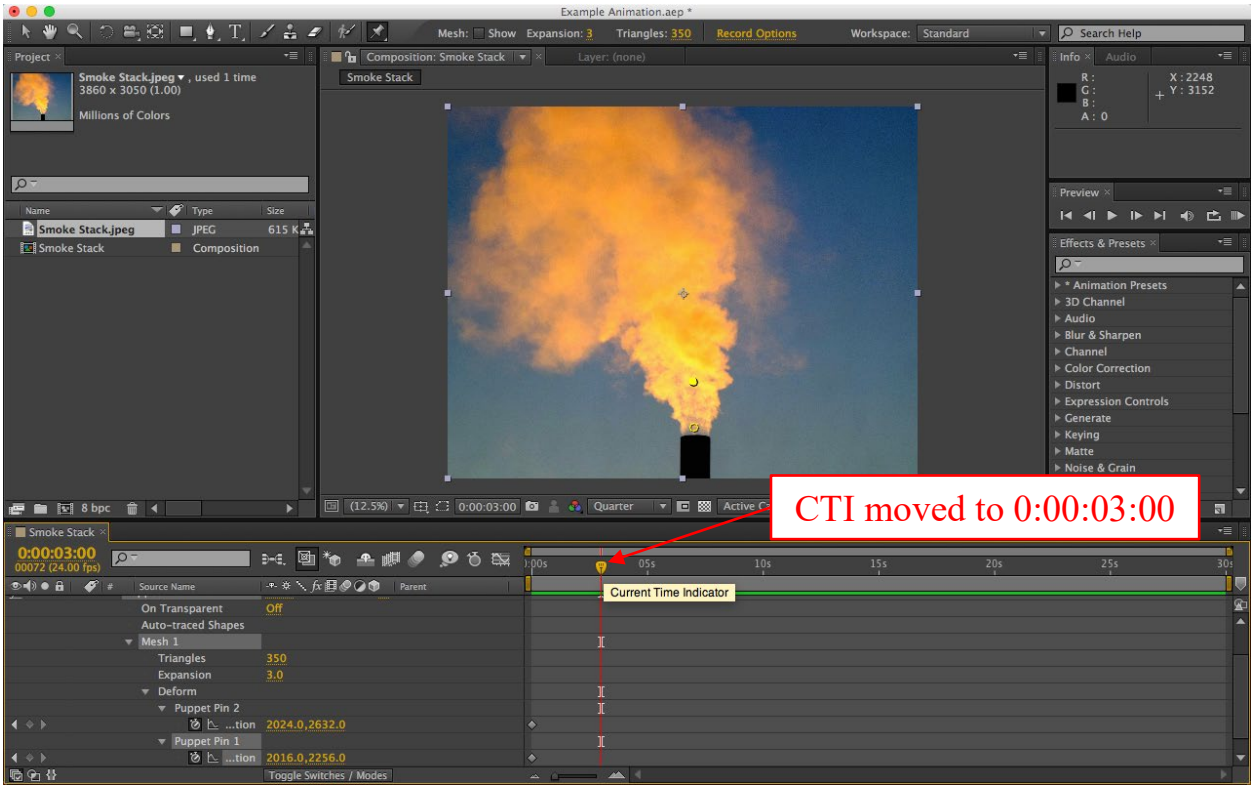
83. A second Deform pin can optionally be placed at the bottom of the smoke near the smokestack itself to keep the smokestack rigid and to prevent it from moving during the animation, as shown below. *See* AEM, 218-19 ("[W]hen animating a person waving, add a pin to each foot to hold them to the ground,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

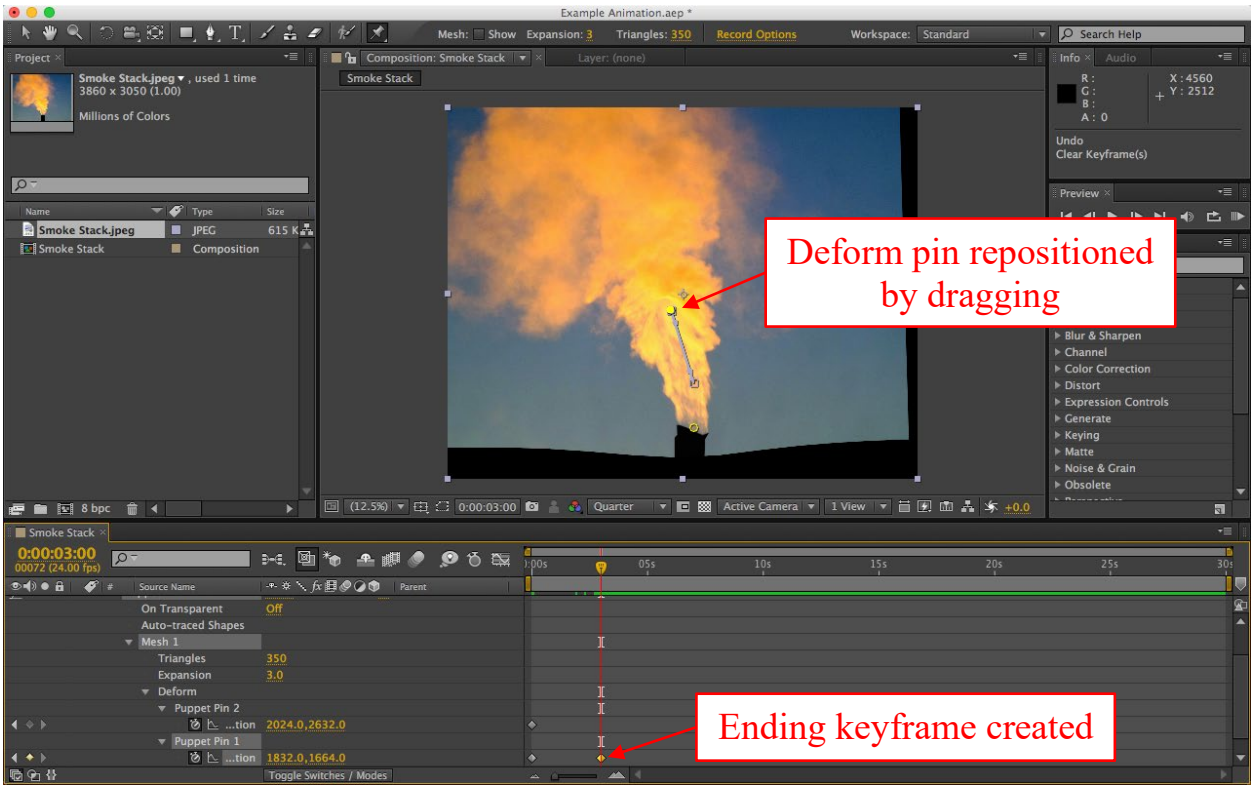


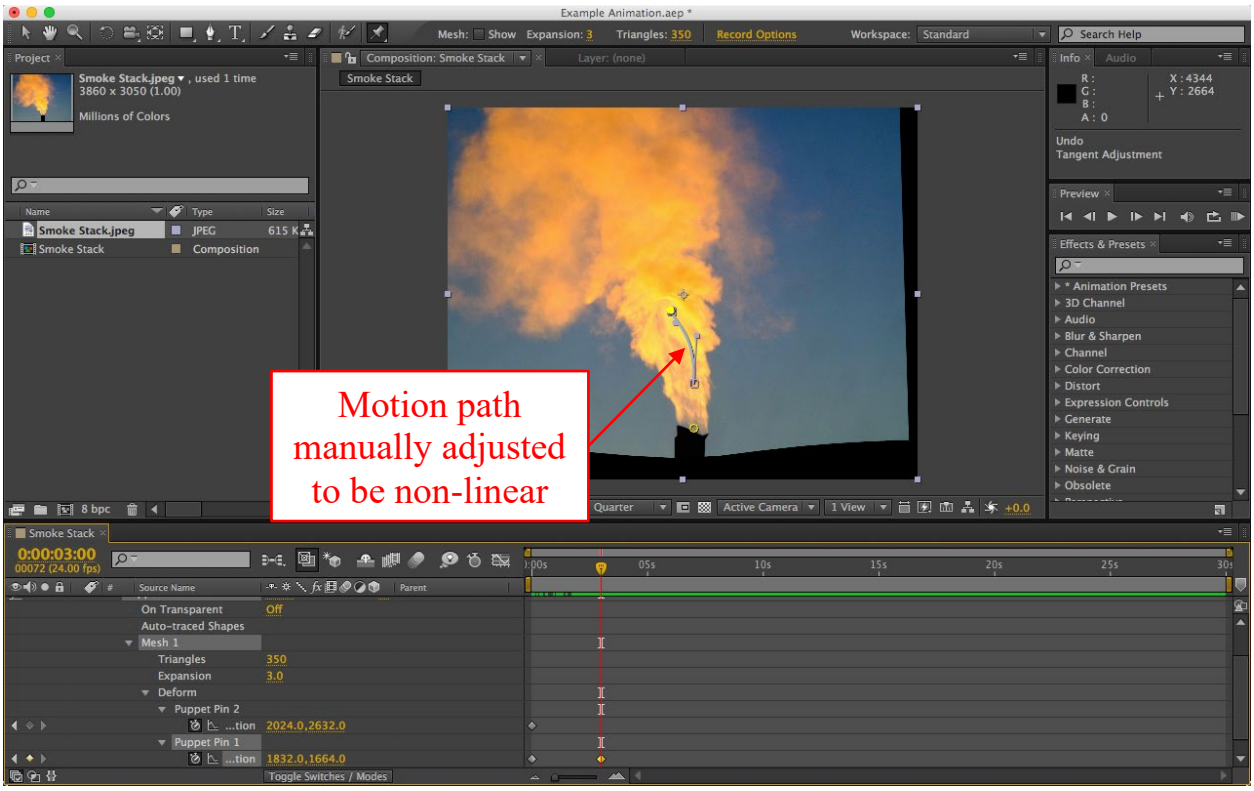

### d. Creating an Ending Keyframe and a Motion Path

84. Next, an ending keyframe for the first Deform pin is created by using the 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 can be made non-linear by manually adjusting the motion path accordingly. *See* AEM, 194-95.




Example Animation.aep *
Deform pin repositioned by dragging
Ending keyframe created

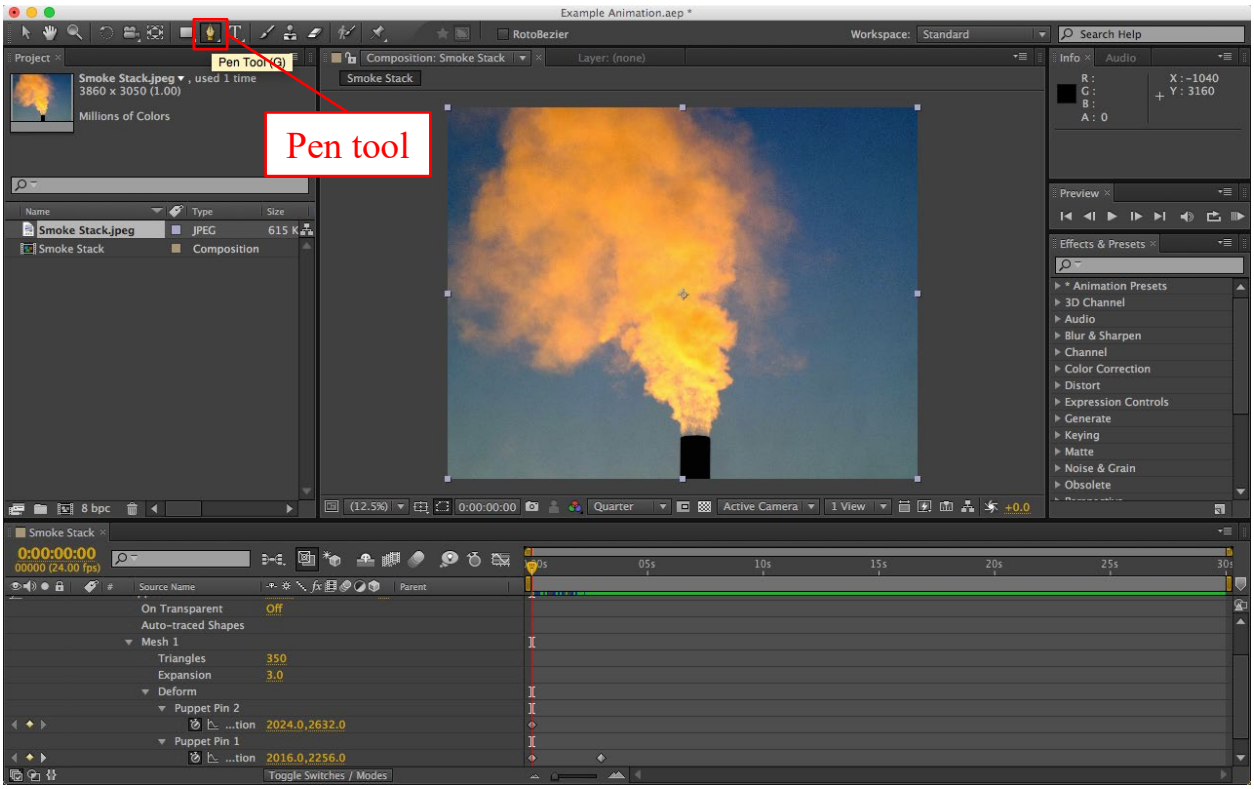

Example Animation.aep *
Motion path manually adjusted to be non-linear

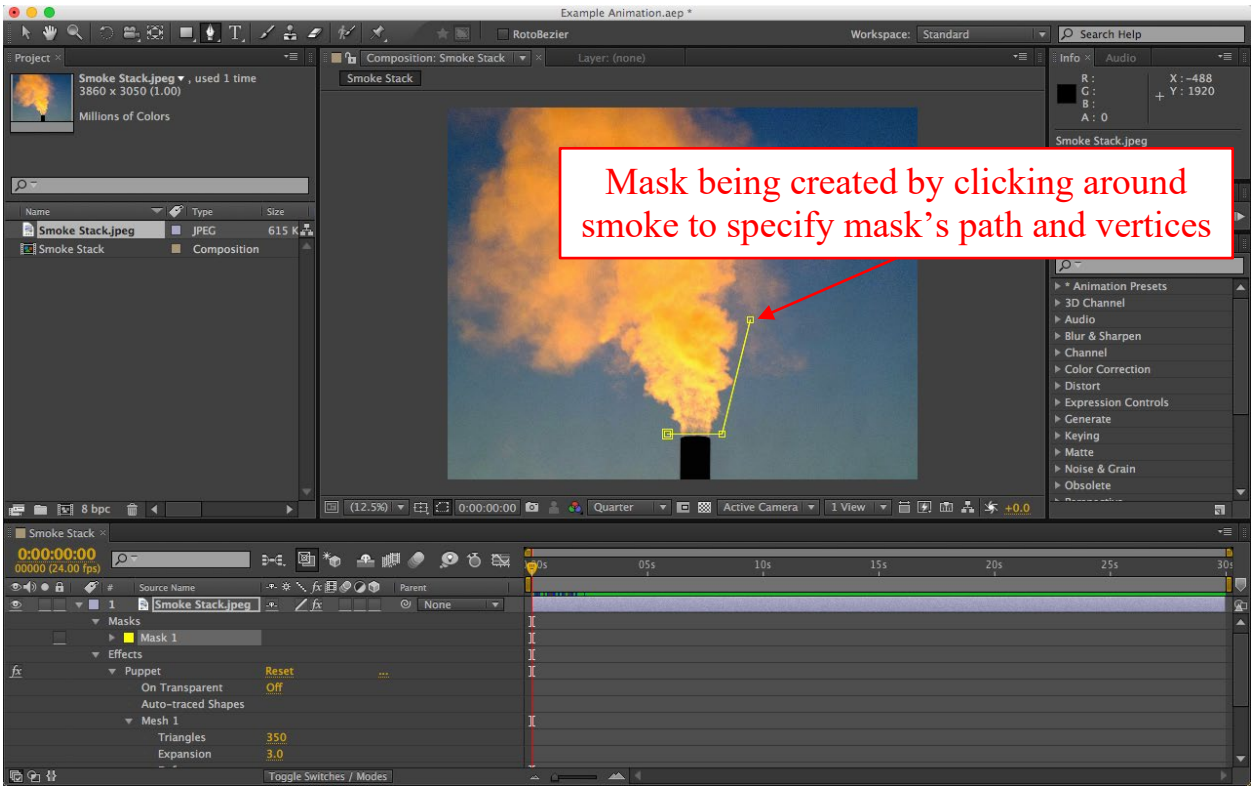
### e. Using a Mask or the Roto Brush Tool to Select a Set of Pixels to be Animated

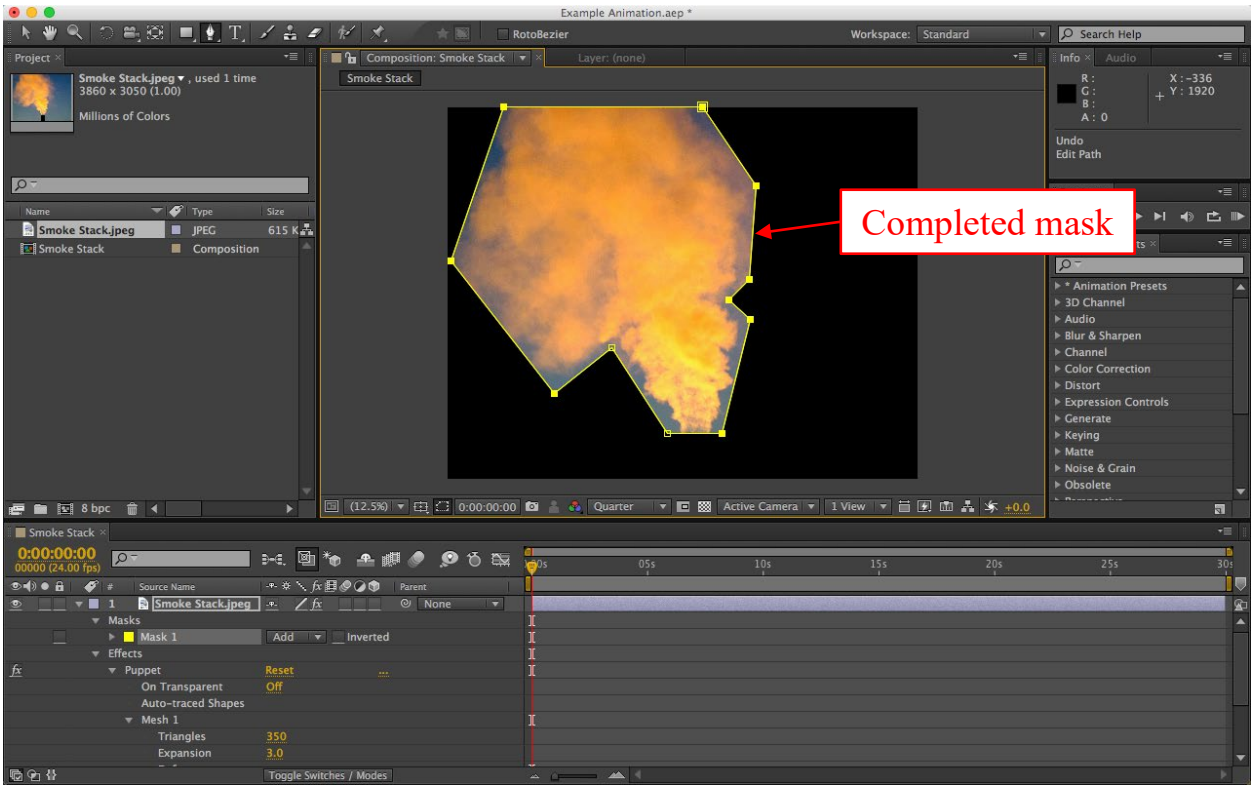
85. The following screenshots show how a mask or matte is used to select only a set of certain pixels to be nontransparent, such as those pixels of the billowing smoke in the exemplary composition, to cause only those pixels to be animated by the Puppet effect. *See* AEM, 315.

#### i. Creating a Mask Using the "Pen" Tool

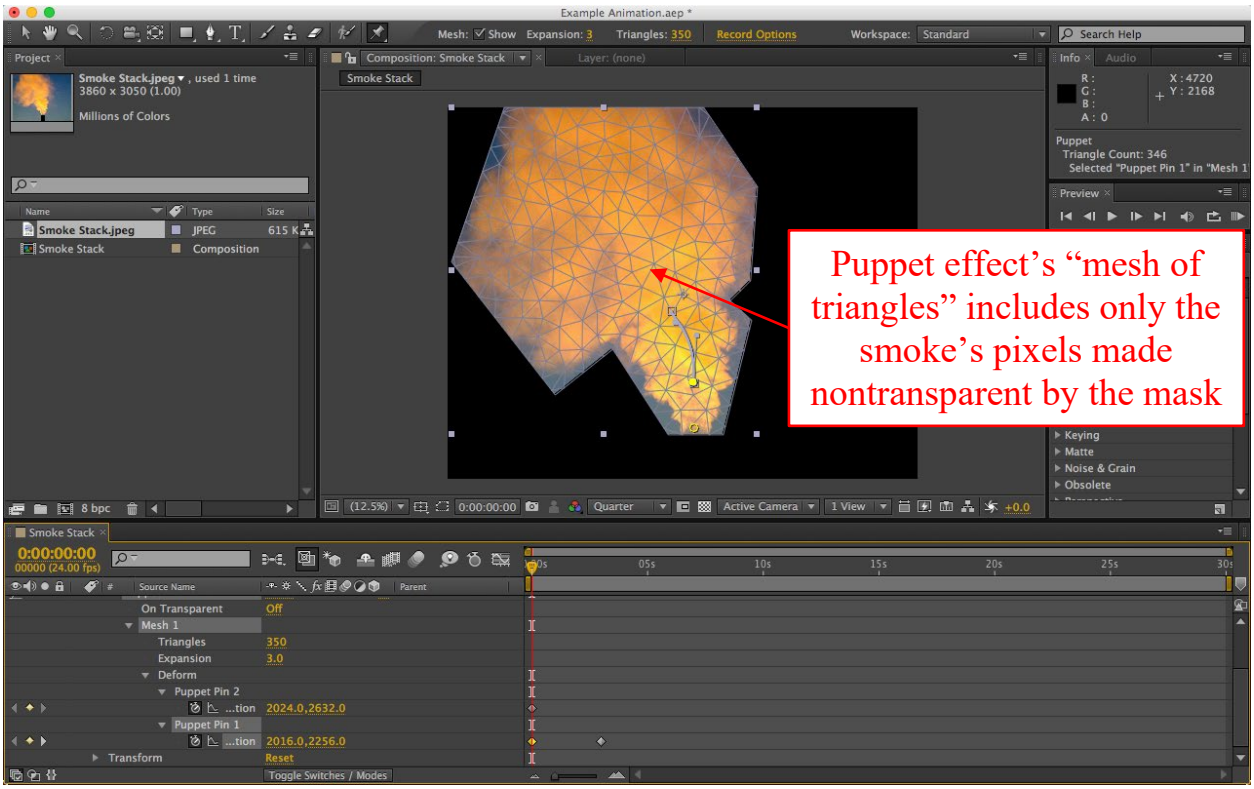
86. As shown in the three screenshots below, a mask can be created around the perimeter of the billowing smoke by using the "Pen" tool. *See* AEM, 318, 264-65. Specifically, the mask is created using the "Pen" tool by clicking on different points around the smoke in the "Composition panel" to specify the shape of the mask correspondingly. *See* AEM, 264-65.


Example Animation.aep *
Pen Tool (G)
Pen tool
Smoke Stack.jpeg, used 1 time
3860 x 3050 (1.00)
Millions of Colors
Smoke Stack.jpeg
Smoke Stack
Composition: Smoke Stack
Layer: (none)
Mesh 1
Triangles 350
Expansion 3.0
Deform
Puppet Pin 2
Puppet Pin 1


Example Animation.aep *
Mask being created by clicking around smoke to specify mask's path and vertices
Smoke Stack.jpeg, used 1 time
3860 x 3050 (1.00)
Millions of Colors
Smoke Stack.jpeg
Smoke Stack
Composition: Smoke Stack
Masks
Mask 1
Effects
Puppet
Reset
Mesh 1
Triangles 350
Expansion 3.0

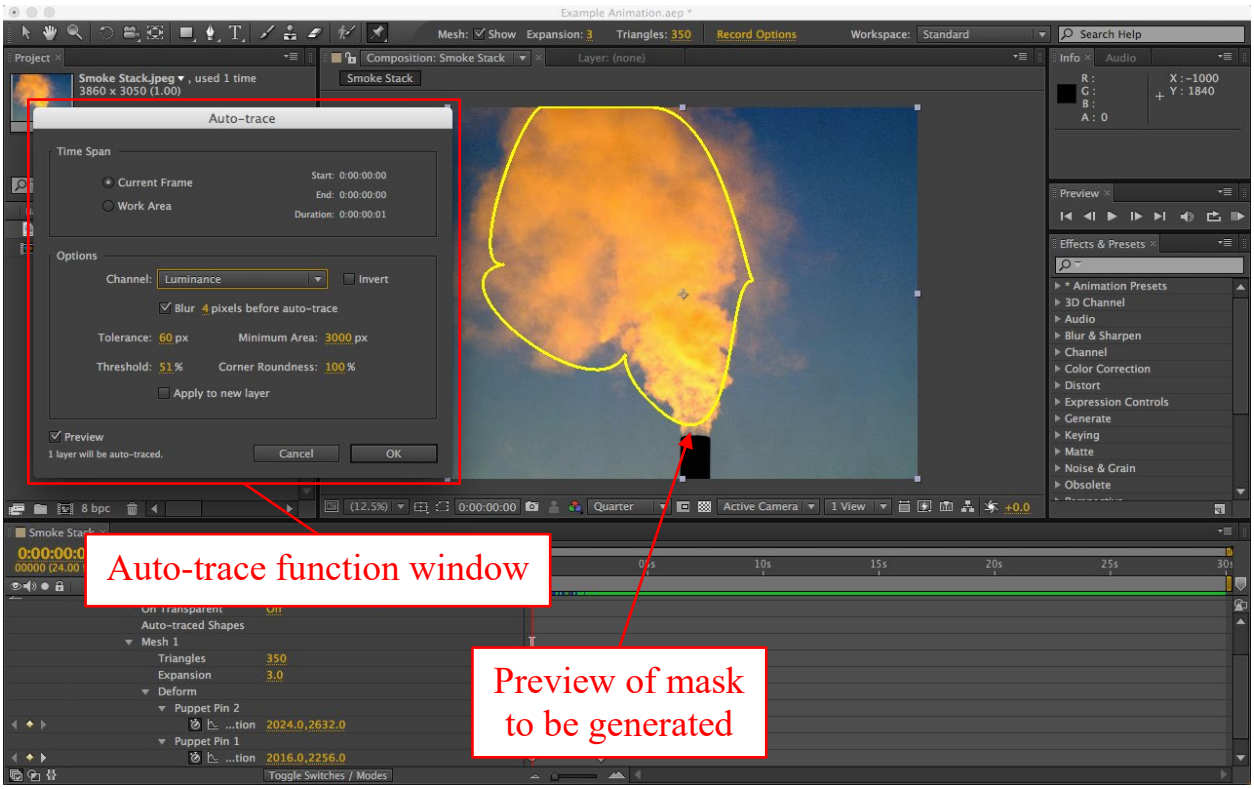


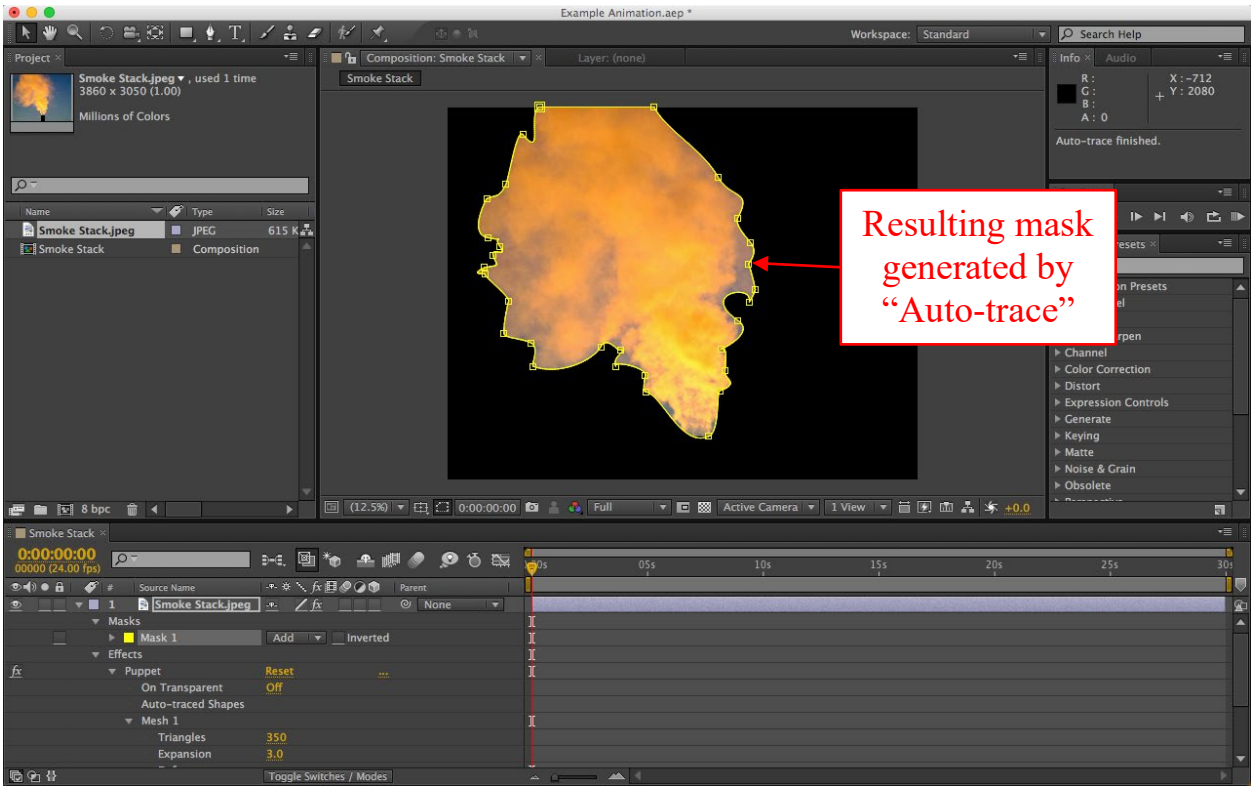

As can be seen, the mask causes only those pixels of the smoke to be nontransparent. *See* AEM, 318. And, as shown in the below screenshot, this causes only these nontransparent pixels to be included in the "mesh of triangles" to be moved during the Puppet effect animation mentioned above. *See* AEM,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



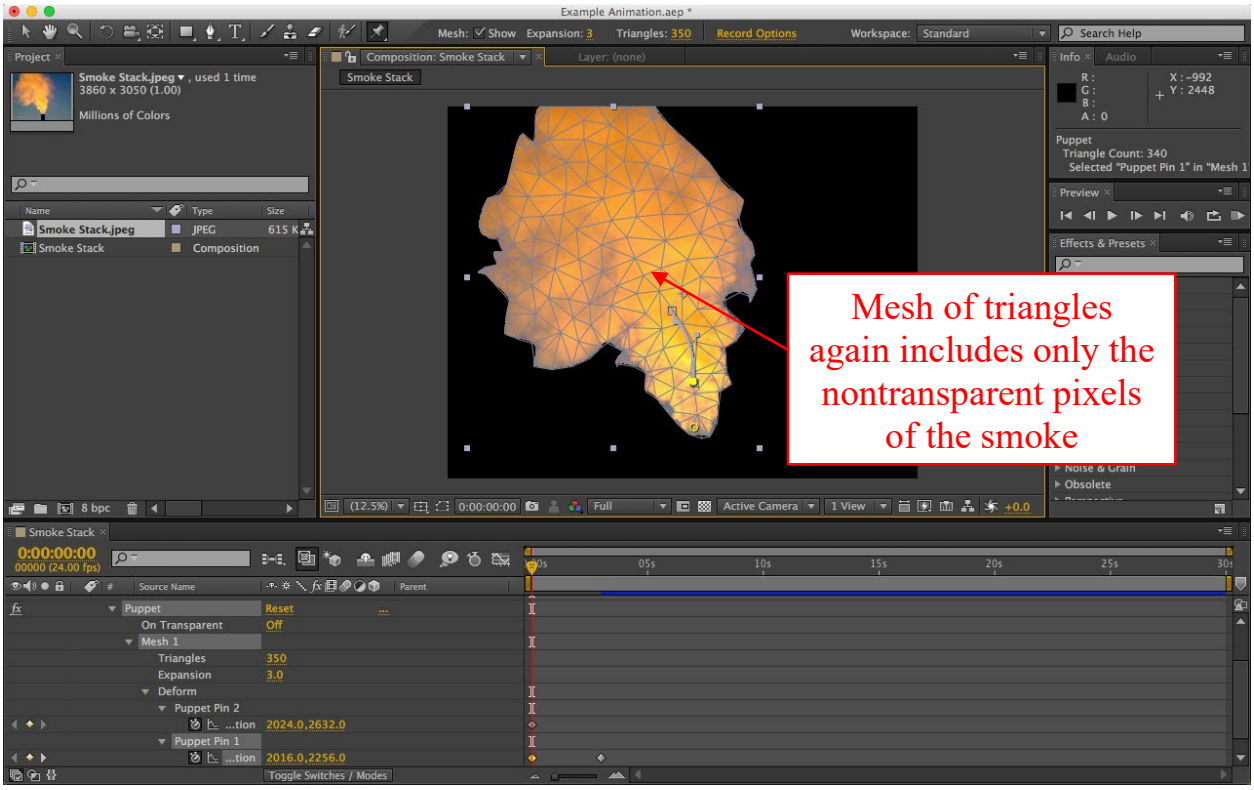

### ii. Creating a Mask Using the "Auto-Trace" Function

87. 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.


Auto-trace function window
Preview of mask to be generated

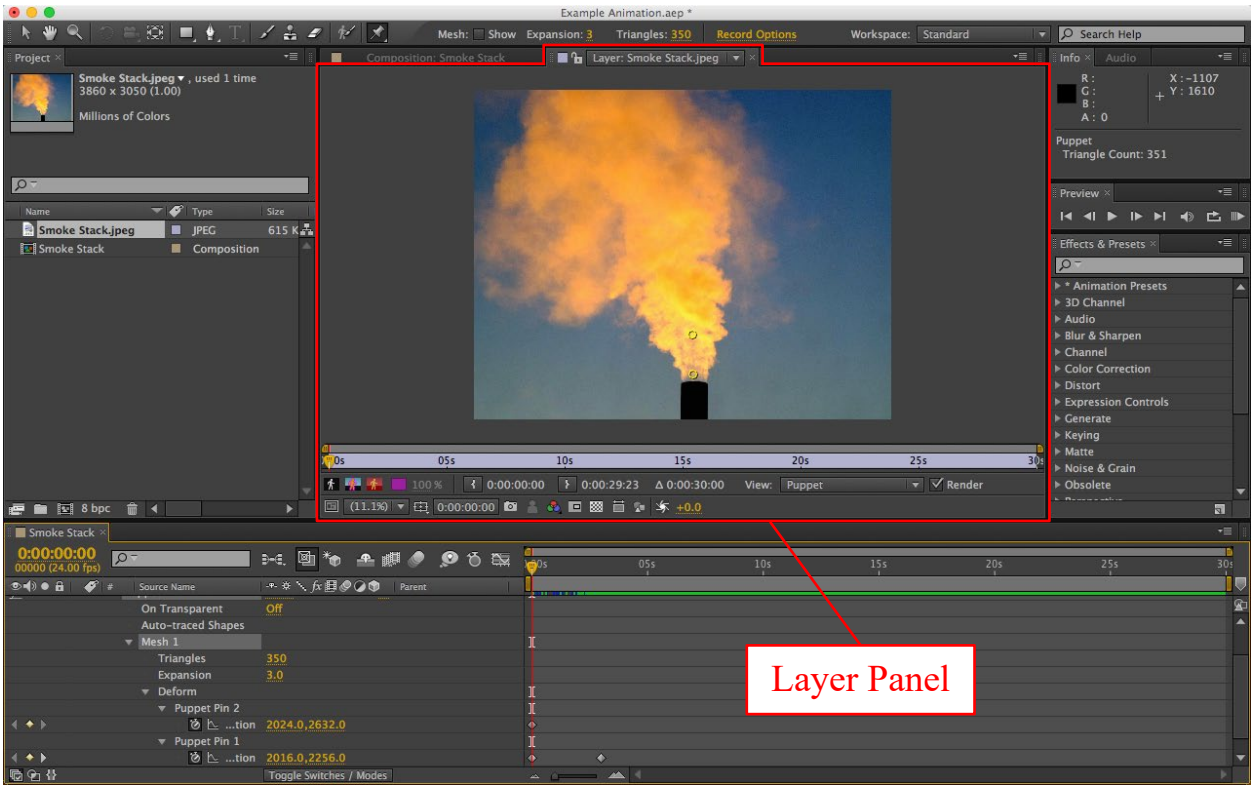

Resulting mask generated by "Auto-trace"

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.

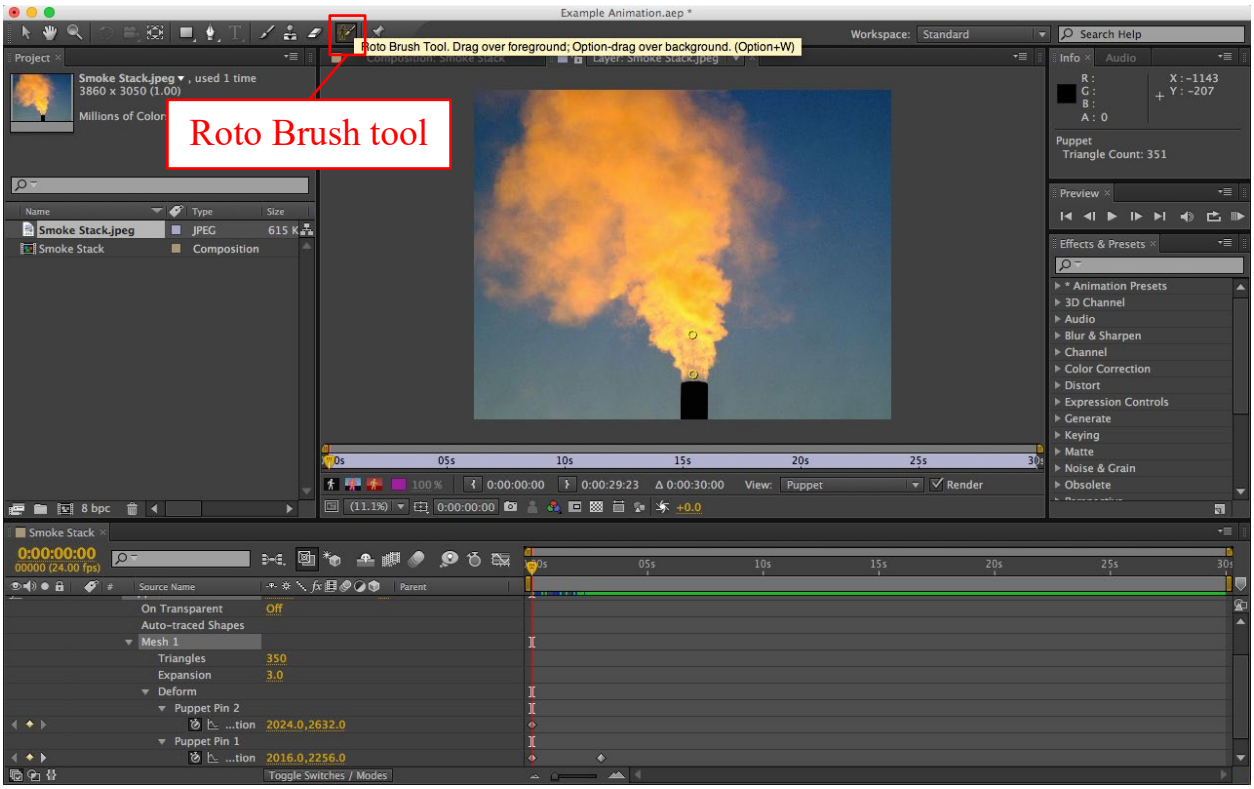


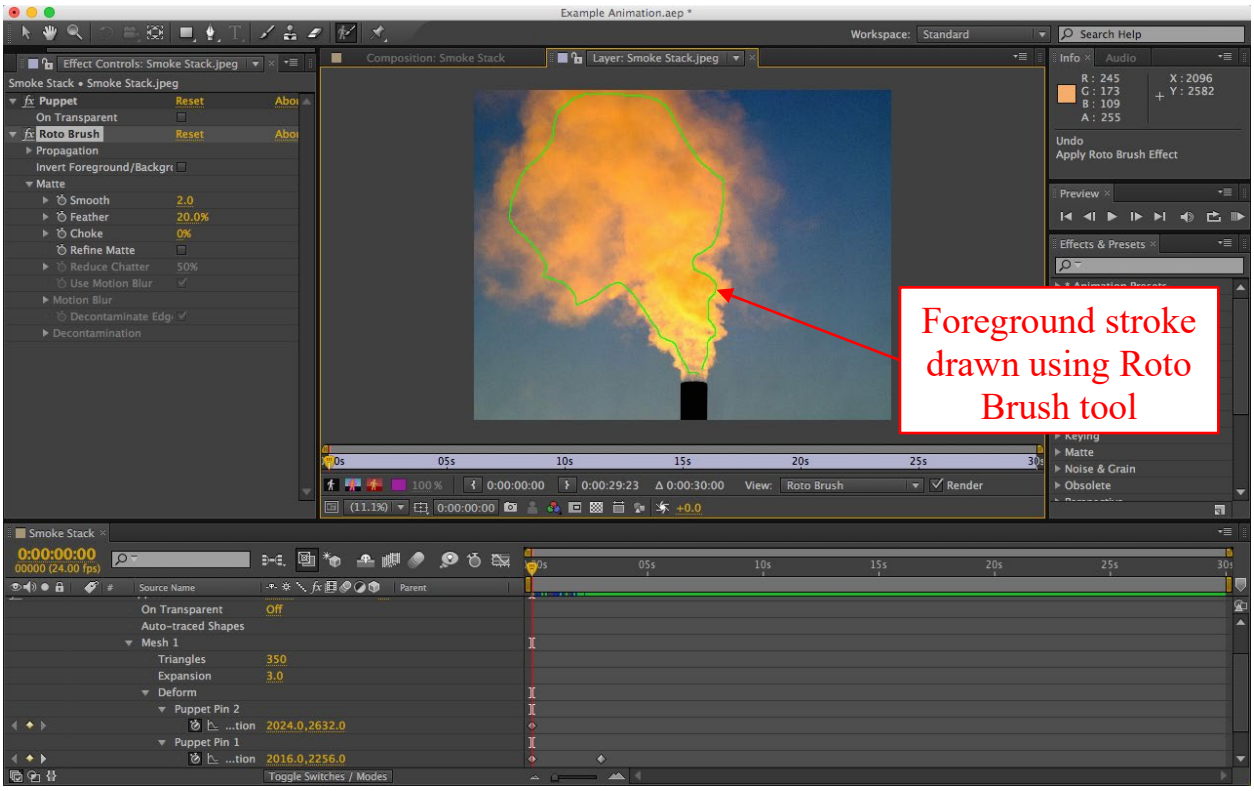

### iii. Creating a Matte Using the "Roto Brush" Tool

88. 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.

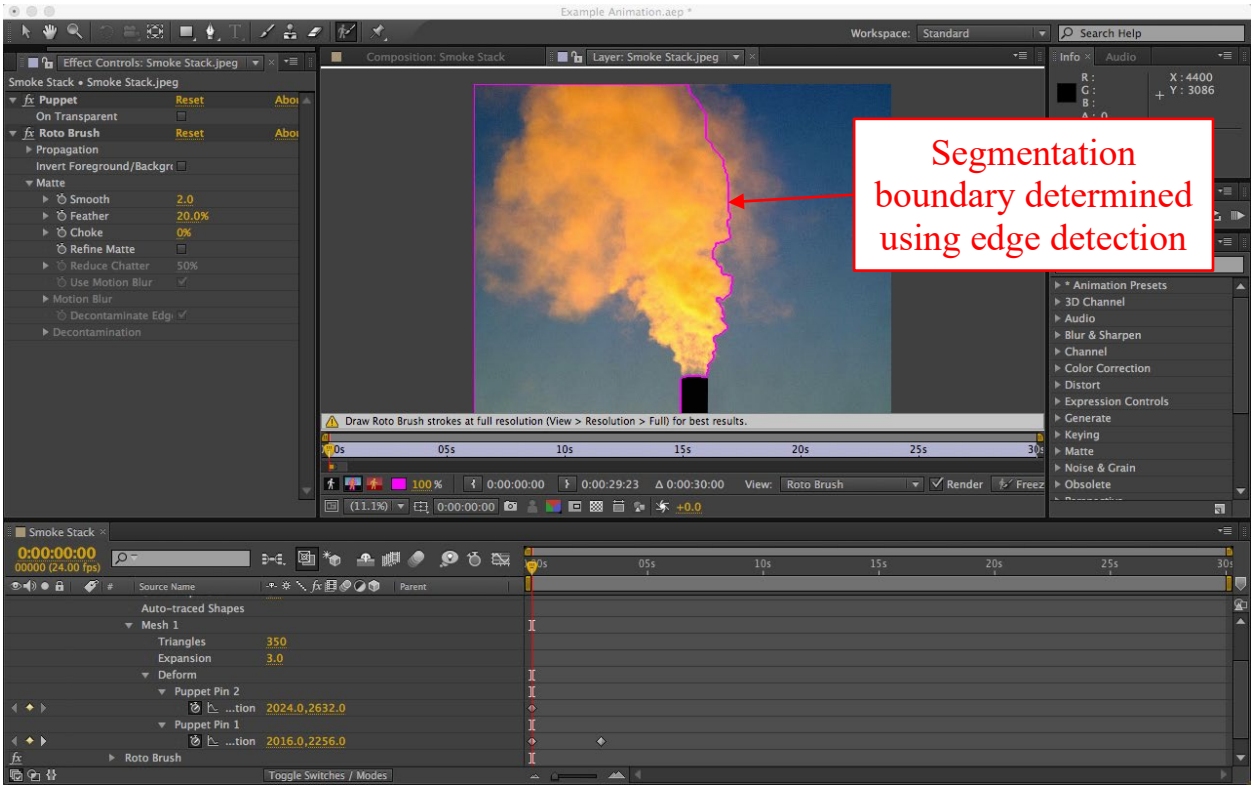



89. Thereafter, as shown in the screenshots below, the "Roto Brush" tool is used to draw a foreground "stroke" (shown in green) that defines the area of the layer to be considered the foreground, which in this exemplary composition is the billowing smoke. *See* AEM, 328.

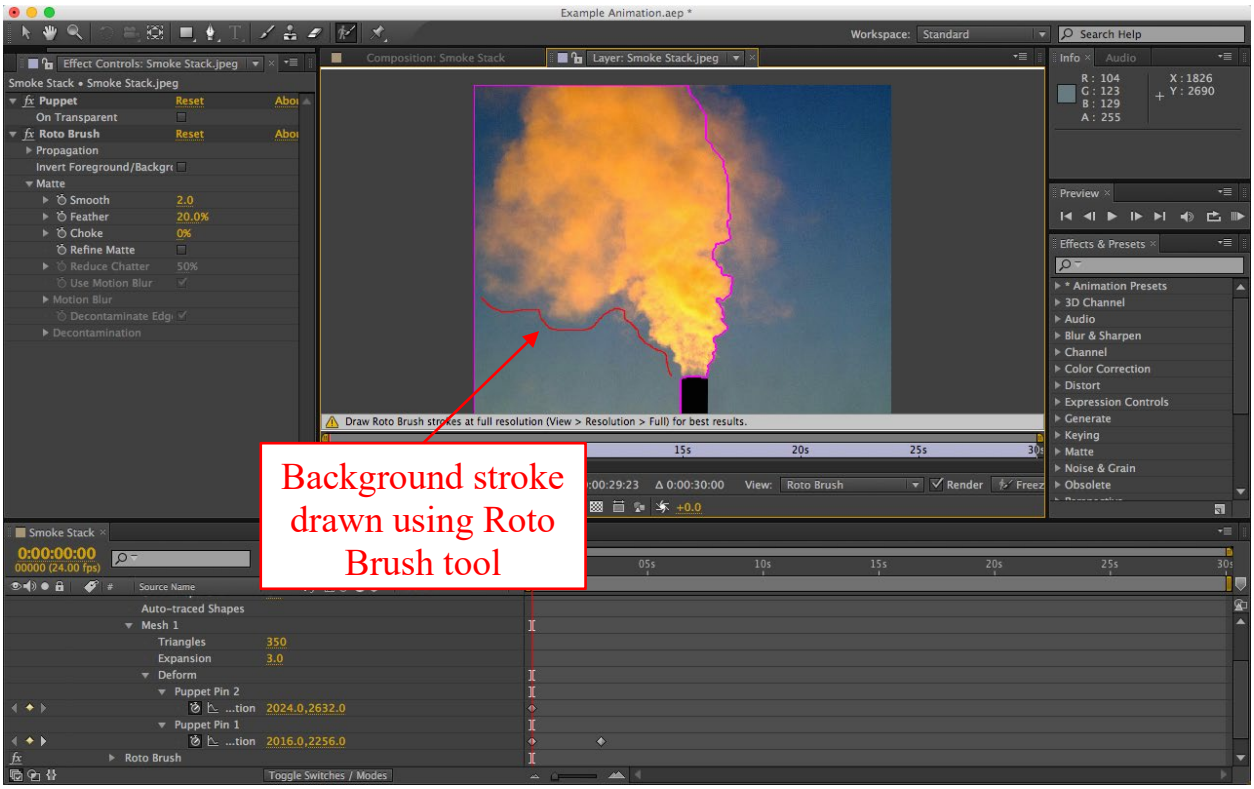

Example Animation.aep *
Roto Brush Tool. Drag over foreground; Option-drag over background. (Option+W)
Roto Brush tool
Project
Smoke Stack.jpeg, used 1 time
3860 x 3050 (1.00)
Millions of Color
Smoke Stack.jpeg
Smoke Stack
Composition
Workspace: Standard
Search Help
Info
Audio
Puppet
Triangle Count: 351
Preview
Effects & Presets
Animation Presets
3D Channel
Audio
Blur & Sharpen
Channel
Color Correction
Distort
Expression Controls
Generate
Keying
Matte
Noise & Grain
Obsolete
View: Puppet
Render
Smoke Stack
On Transparent
Auto-traced Shapes
Mesh 1
Triangles 350
Expansion 3.0
Deform
Puppet Pin 2
Puppet Pin 1
Toggle Switches / Modes


Example Animation.aep *
Effect Controls: Smoke Stack.jpeg
Smoke Stack • Smoke Stack.jpeg
Puppet
Reset
On Transparent
Roto Brush
Propagation
Invert Foreground/Backgr
Matte
Smooth 2.0
Feather 20.0%
Choke 0%
Refine Matte
Reduce Chatter 50%
Use Motion Blur
Motion Blur
Decontamination
Composition: Smoke Stack
Layer: Smoke Stack.jpeg
Foreground stroke drawn using Roto Brush tool
Undo
Apply Roto Brush Effect
View: Roto Brush
Render
Smoke Stack
On Transparent
Auto-traced Shapes
Mesh 1
Triangles 350
Expansion 3.0
Deform
Puppet Pin 2
Puppet Pin 1
Toggle Switches / Modes

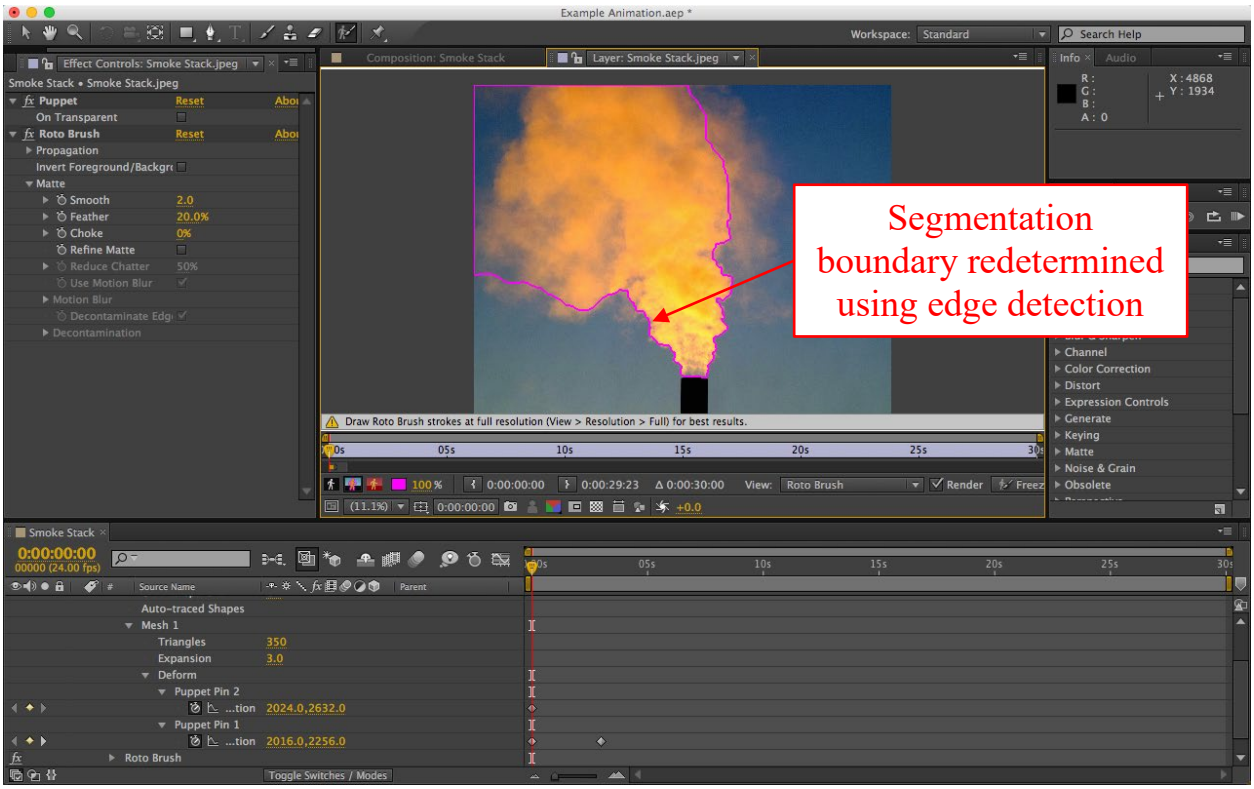
Based on this foreground stroke, AECS6 uses "Edge Detection" to then determine a "segmentation boundary" (shown in pink) that separates the foreground smoke and the background sky, as shown below. *See* AEM, 328-31.



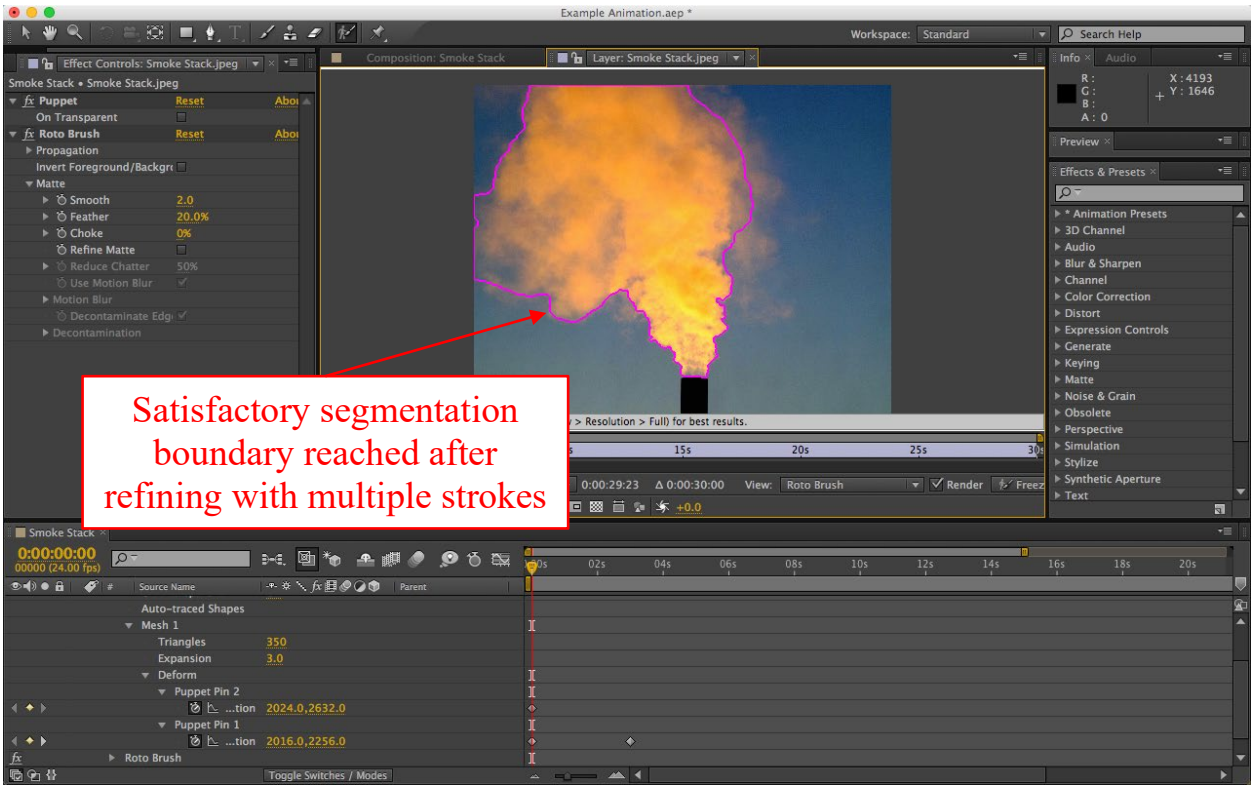

90. Although the segmentation boundary is not perfect, it is refined as shown in the screenshot below by using the "Roto Brush" tool to draw a background stroke (shown in red) that defines the area of the layer to be considered the background, which in this exemplary composition is the sky. *See* AEM,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



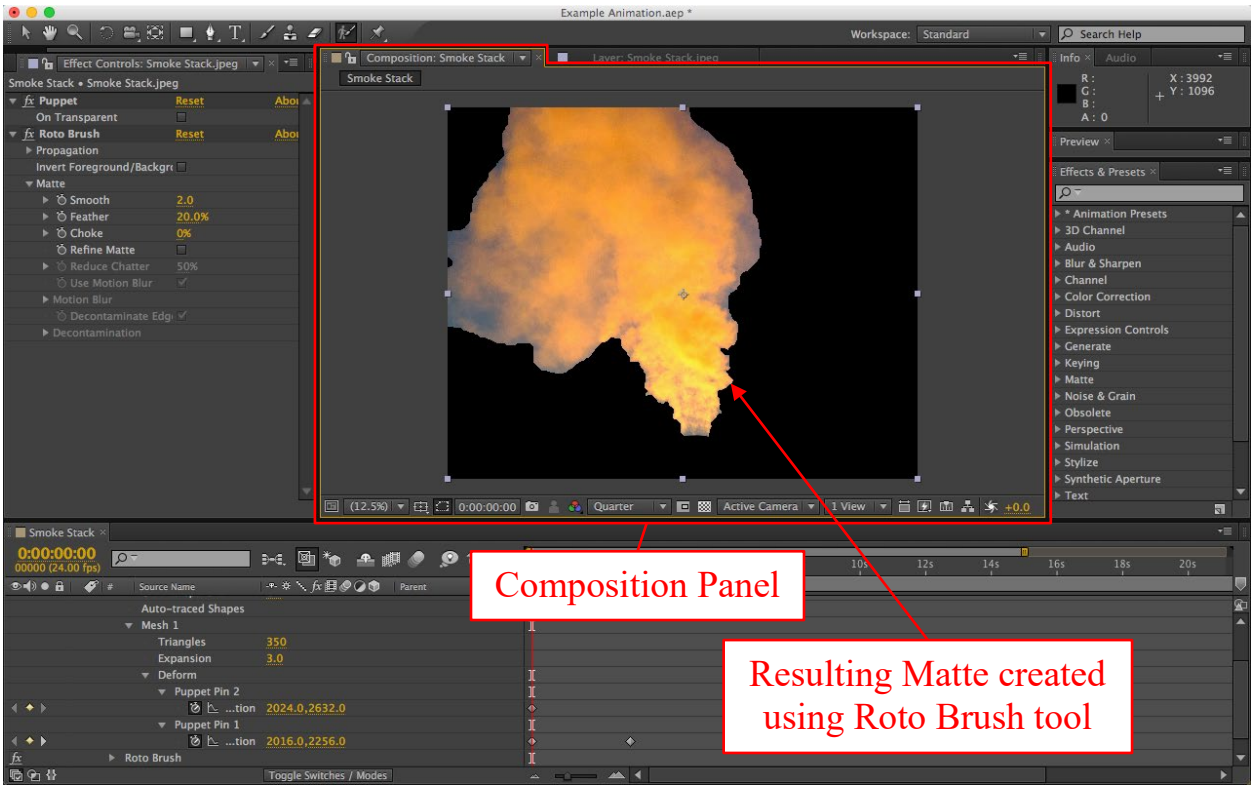

Based on this background stroke and the previous foreground stroke, AECS6 again uses edge detection to then redetermine the segmentation boundary, as shown below. *See* AEM, 328-31.



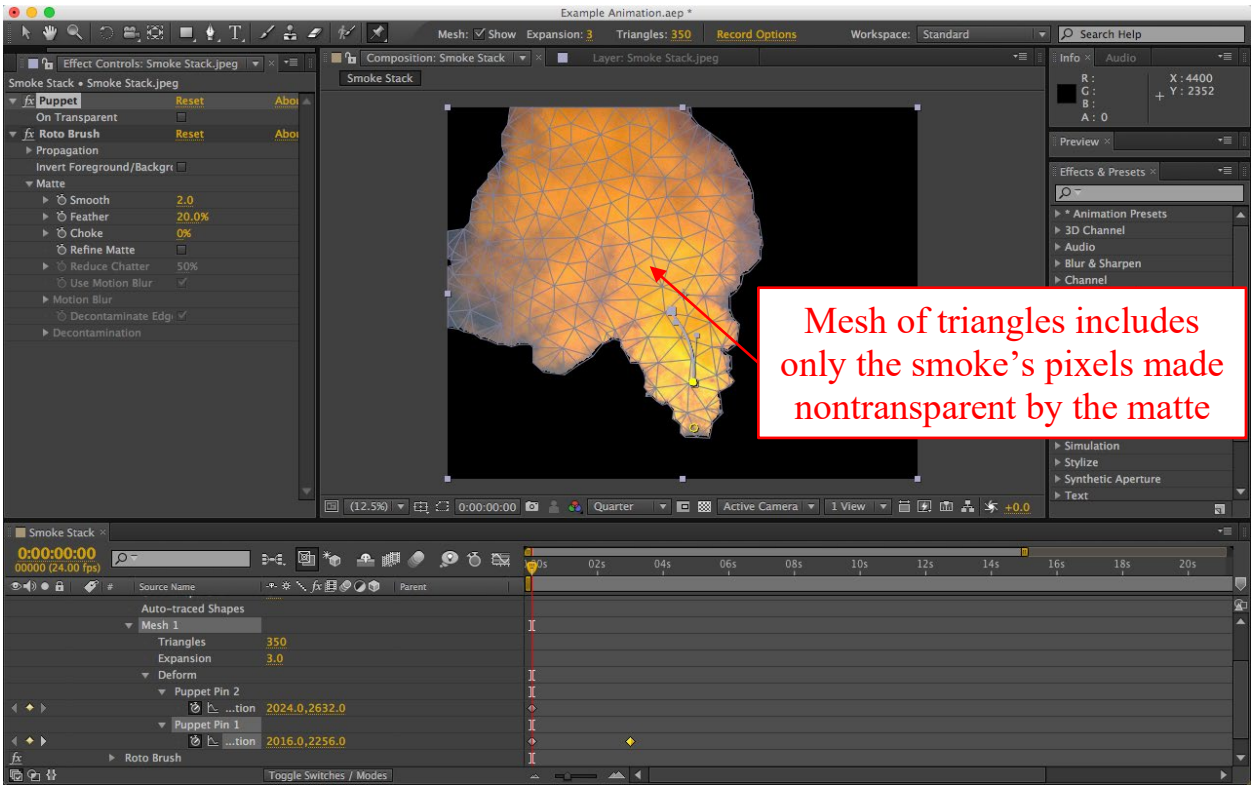

91. As can be seen, the segmentation boundary is again not perfect but is now closer than before to the true boundary between the foreground smoke and background sky. Additional foreground and background strokes are drawn to continue the refining process until a satisfactory segmentation boundary is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




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

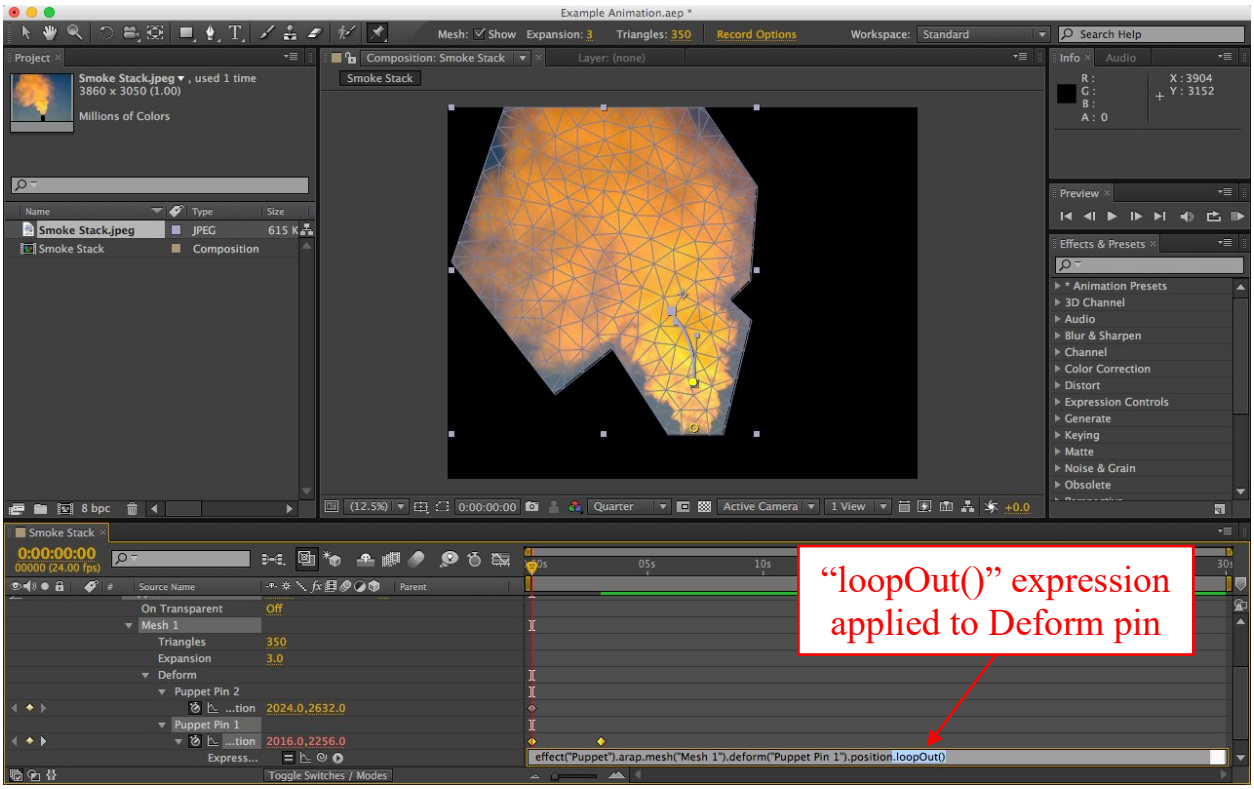



The below screenshot shows the resulting "mesh of triangles" created from the pixels specified by the matte to be part of the foreground, i.e., the pixels of the billowing smoke made nontransparent by the matte. *See* AEM, 219.




### f. Looping the Animation

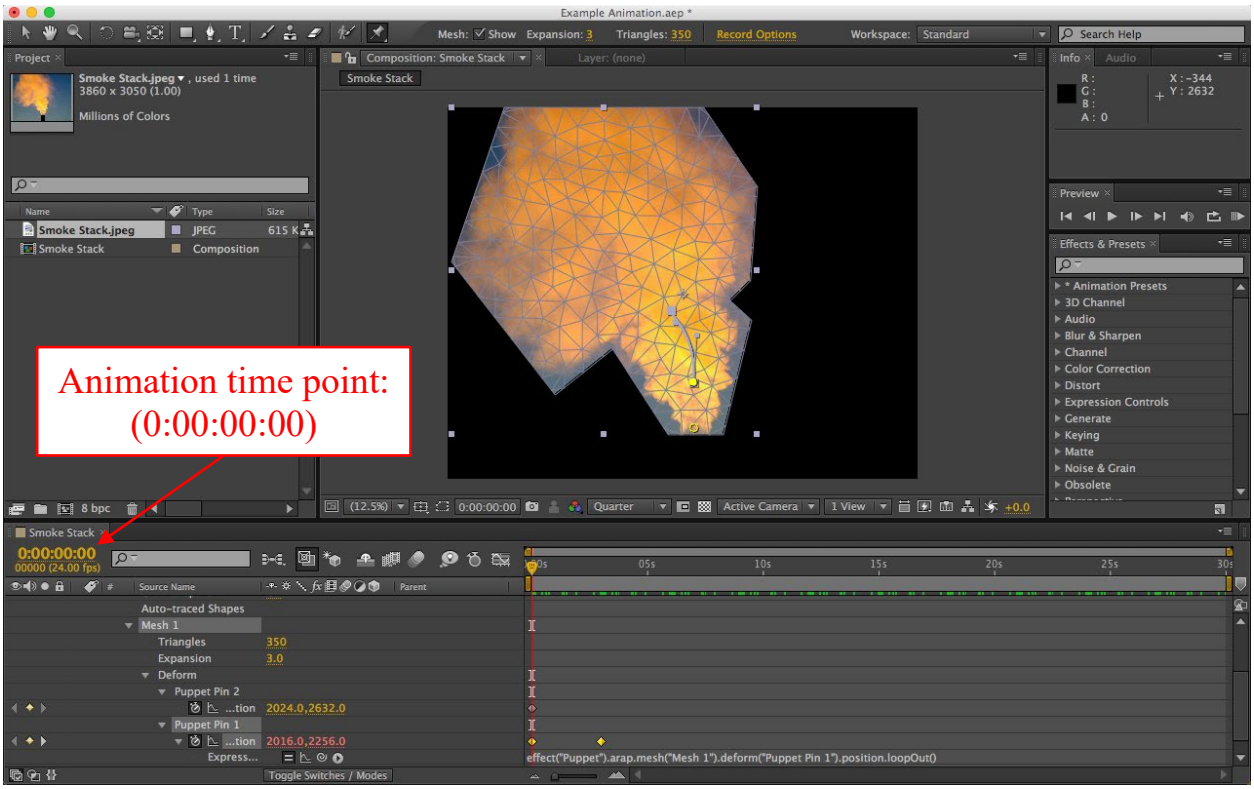
93. After creating the mesh using one of the above tools and functions, a "loopOut()" expression can be applied to the Deform pin to cause the resulting animation to repeatedly loop. *See* AEM, 562, 219. The screenshot below shows how a "loopOut()" expression is applied to the Deform pin in the exemplary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

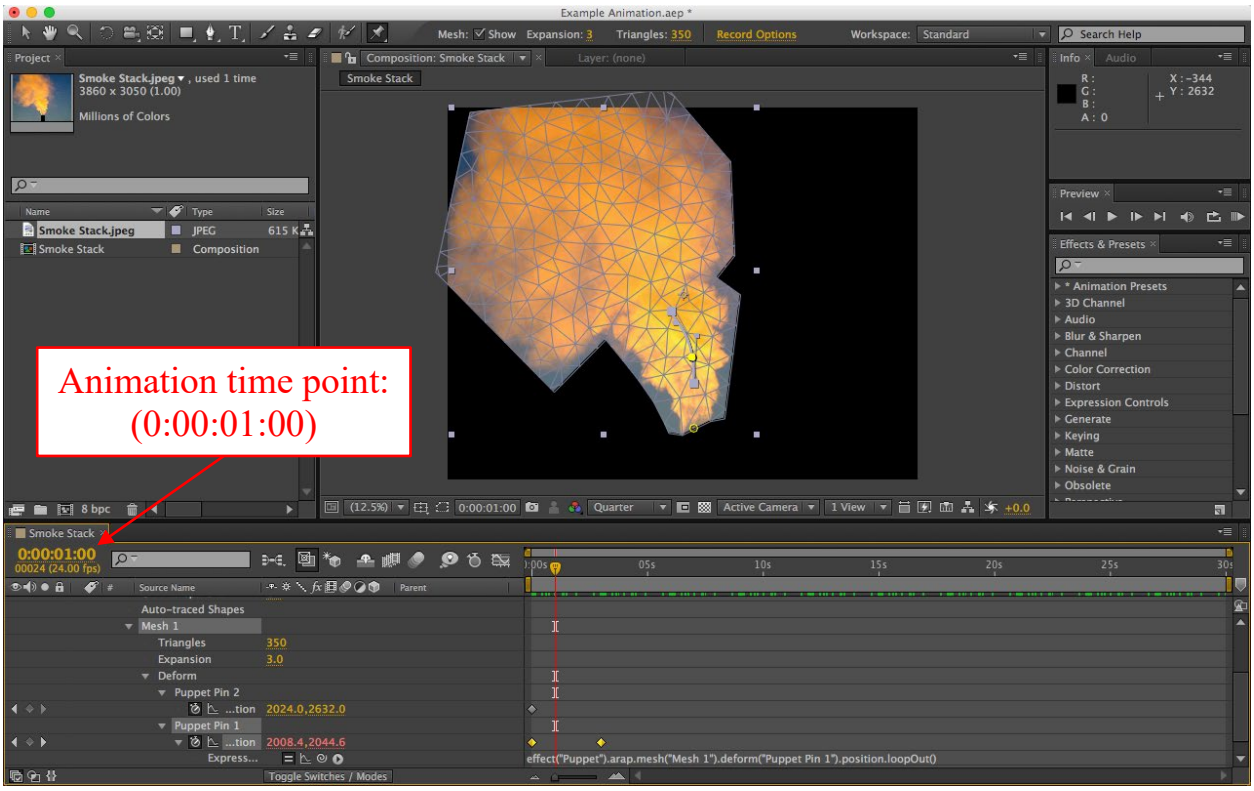


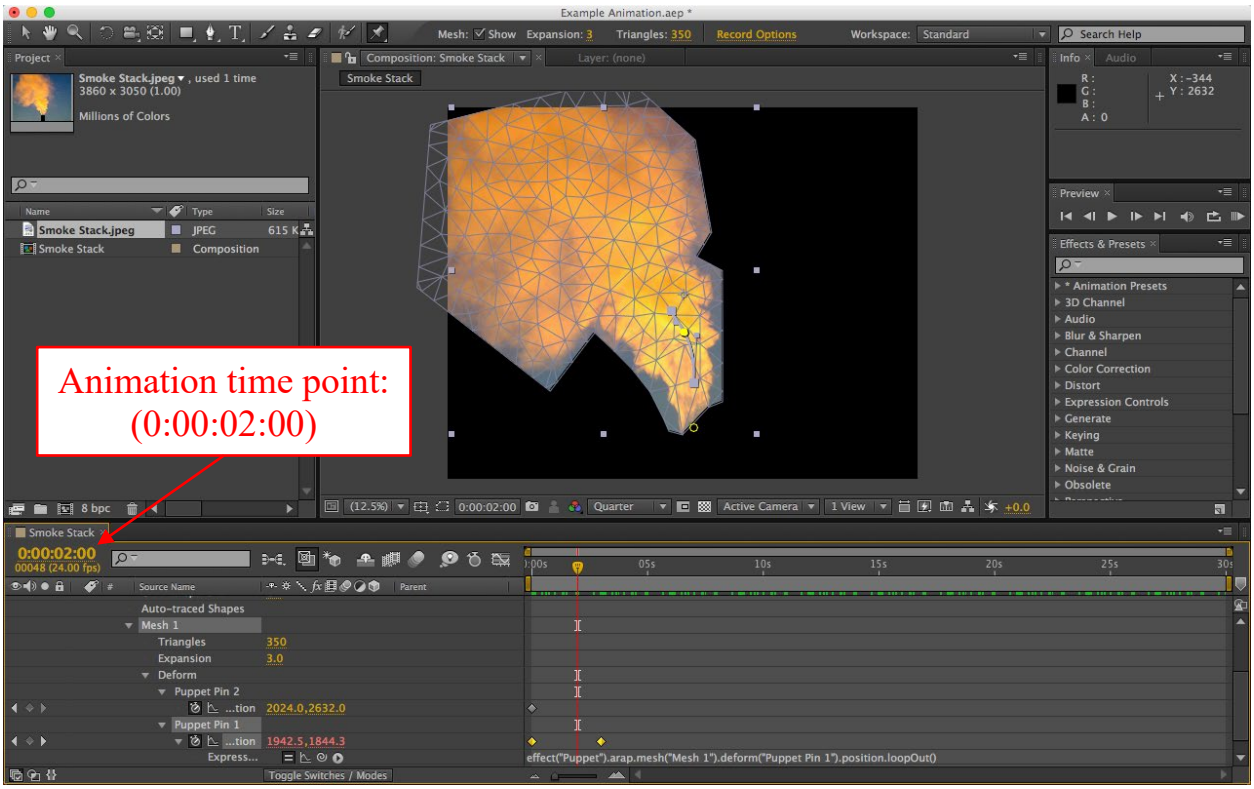

While not shown here, the same can be done to the Deform pins in the exemplary compositions depicted in the final screenshots of Sections VIII.B.2.e.ii and VIII.B.2.e.iii to achieve the same result of a repeatedly looping animation.

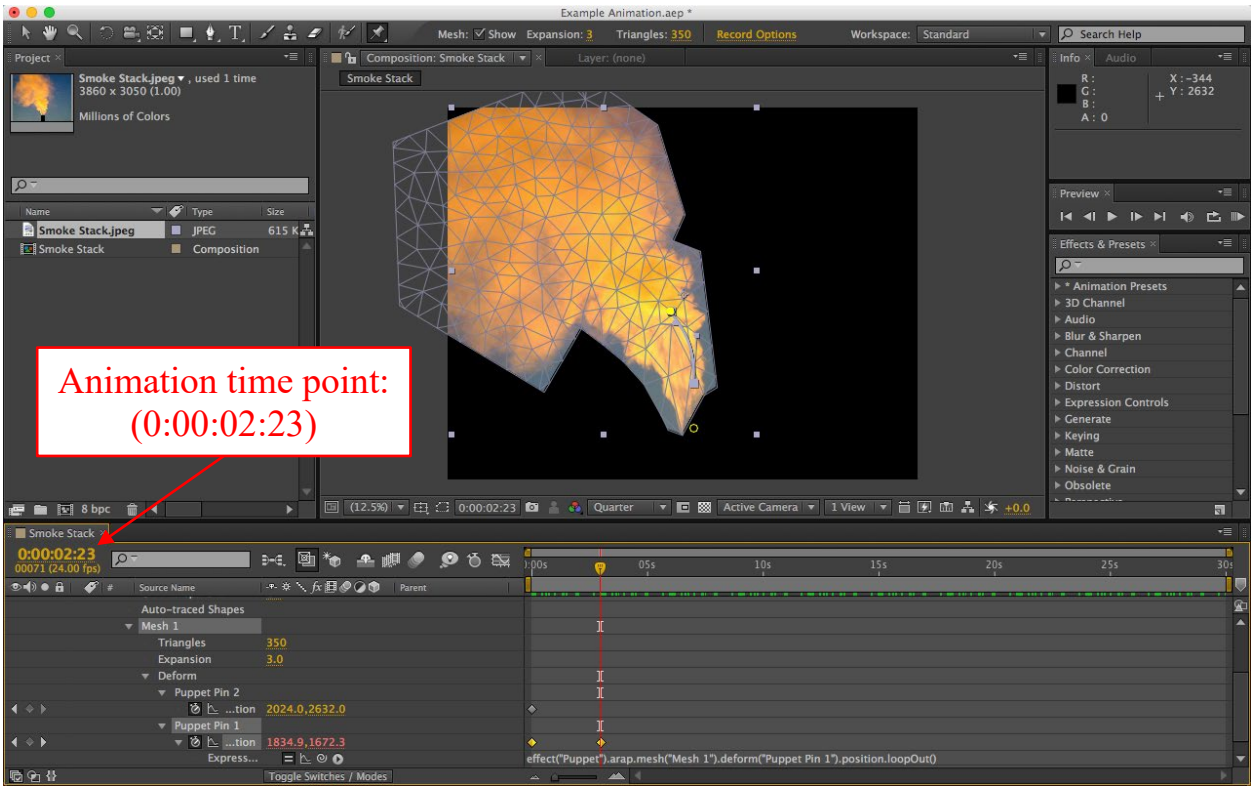
### g. Final Resulting Animation

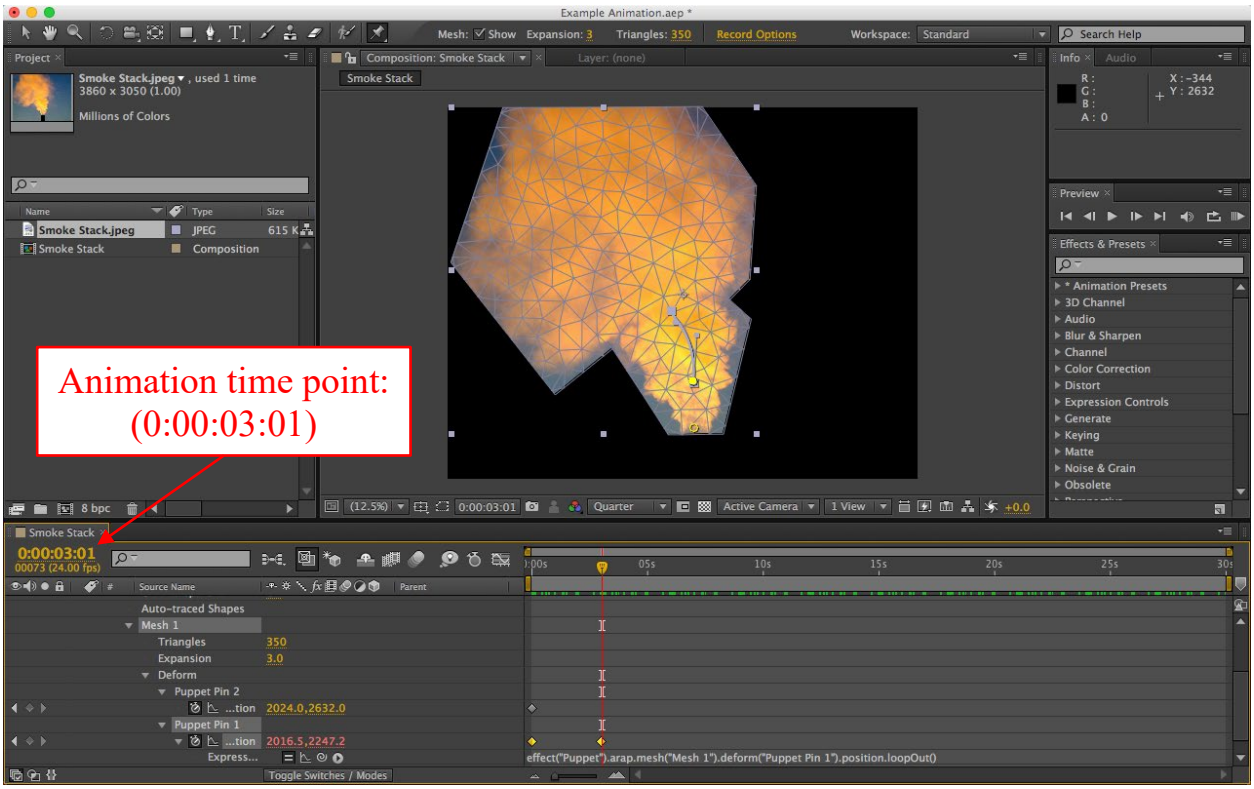
94. The final resulting animation of the exemplary composition in Section VIII.B.2.f is shown in the below screenshots. From top to bottom, the screenshots show the animation at the following time points, respectively, in order to mimic the animation playing in time: (0:00:00:00), (0:00:01:00), (0:00:02:00), (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.

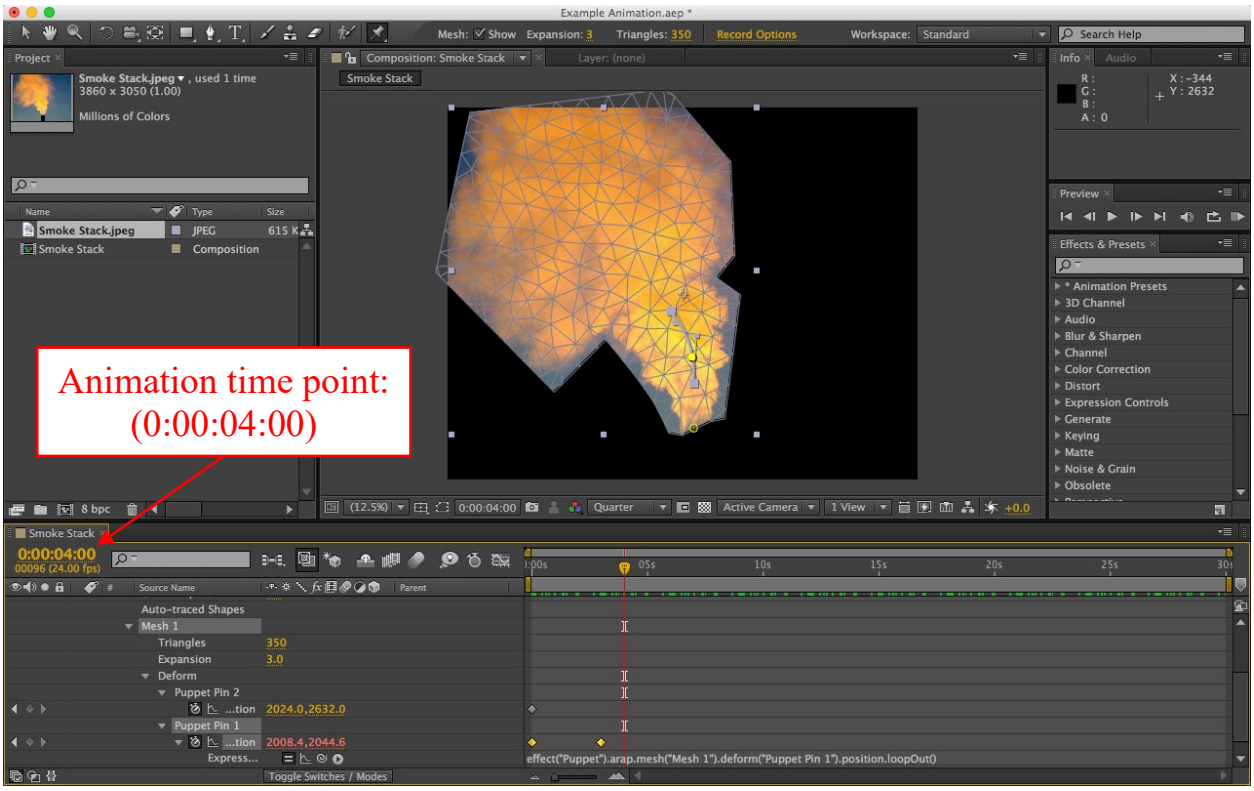

Example Animation.aep *
Mesh: Show Expansion: 3 Triangles: 350 Record Options Workspace: Standard
Smoke Stack.jpeg , used 1 time
3860 x 3050 (1.00)
Millions of Colors
Smoke Stack.jpeg JPEG 615 K
Smoke Stack Composition
Composition: Smoke Stack
Smoke Stack
Animation time point:
(0:00:00:00)
0:00:00:00
00000 (24.00 fps)
Auto-traced Shapes
Mesh 1
Triangles 350
Expansion 3.0
Deform
Puppet Pin 2
...tion 2024.0,2632.0
Puppet Pin 1
...tion 2016.0,2256.0
effect("Puppet").arap.mesh("Mesh 1").deform("Puppet Pin 1").position.loopOut()

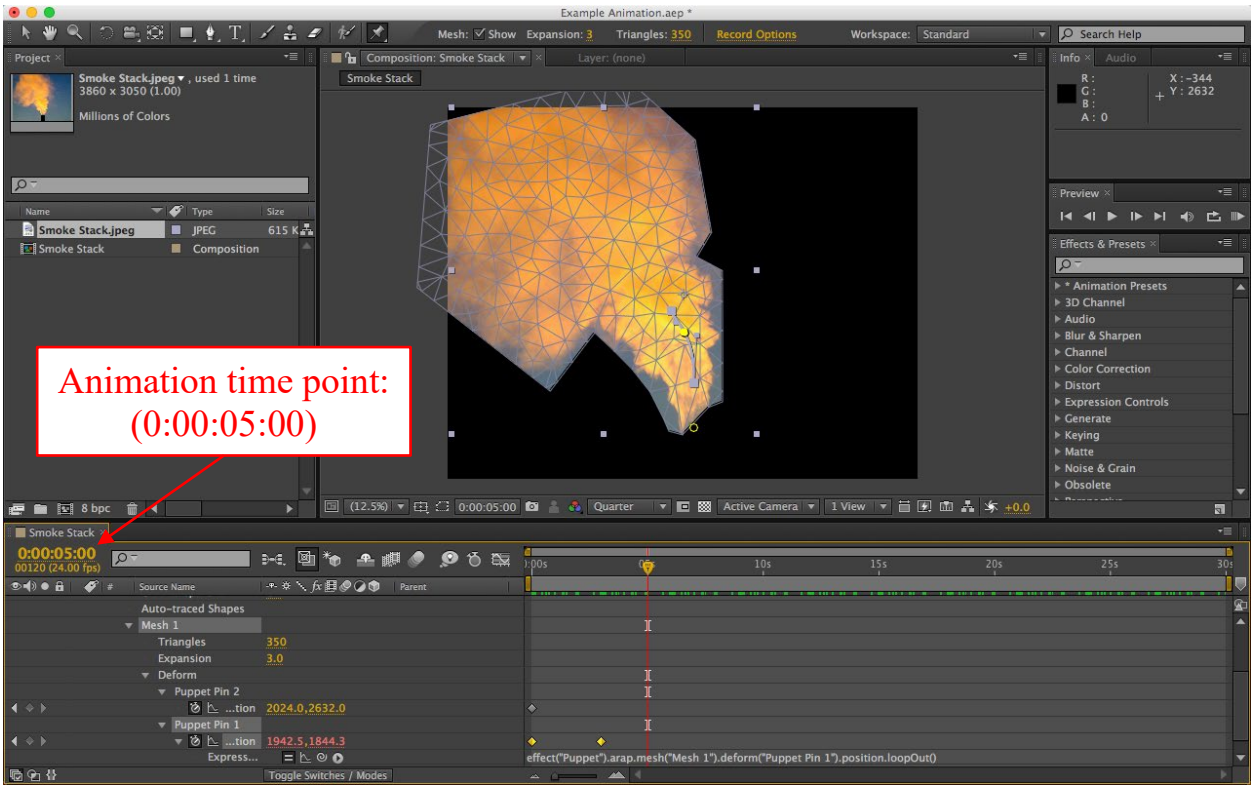

Example Animation.aep *
Mesh: Show Expansion: 3 Triangles: 350 Record Options Workspace: Standard
Smoke Stack.jpeg , used 1 time
3860 x 3050 (1.00)
Millions of Colors
Smoke Stack.jpeg JPEG 615 K
Smoke Stack Composition
Composition: Smoke Stack
Smoke Stack
Animation time point:
(0:00:01:00)
0:00:01:00
00024 (24.00 fps)
Auto-traced Shapes
Mesh 1
Triangles 350
Expansion 3.0
Deform
Puppet Pin 2
...tion 2024.0,2632.0
Puppet Pin 1
...tion 2008.4,2044.6
effect("Puppet").arap.mesh("Mesh 1").deform("Puppet Pin 1").position.loopOut()

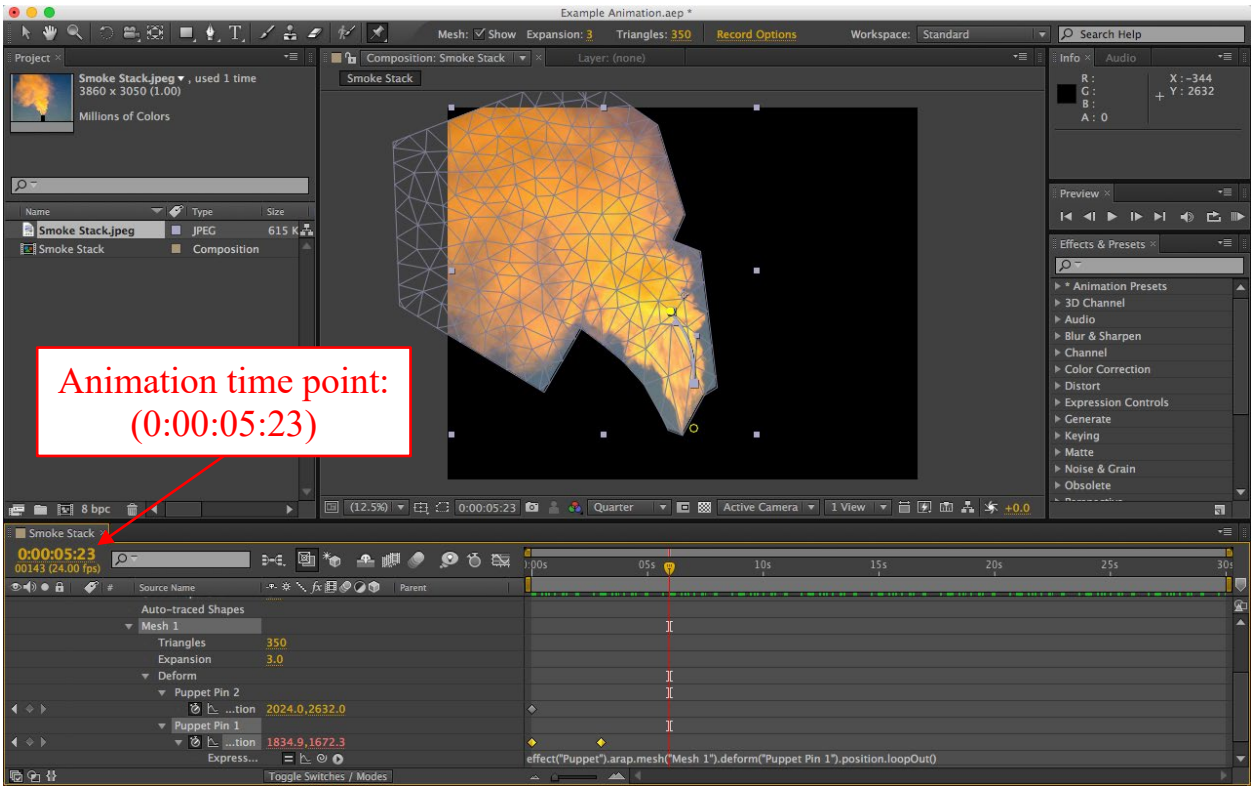

Example Animation.aep *
Mesh: Show Expansion: 3 Triangles: 350 Record Options Workspace: Standard
Search Help
Project
Smoke Stack.jpeg ▾ , used 1 time
3860 x 3050 (1.00)
Millions of Colors
Name Type Size
Smoke Stack.jpeg JPEG 615 K
Smoke Stack Composition
Composition: Smoke Stack
Layer: (none)
Smoke Stack
Info Audio
R :
G :
B :
A : 0
X : -344
Y : 2632
Preview
Effects & Presets
* Animation Presets
3D Channel
Audio
Blur & Sharpen
Channel
Color Correction
Distort
Expression Controls
Generate
Keying
Matte
Noise & Grain
Obsolete
Animation time point:
(0:00:02:00)
8 bpc
(12.5%) 0:00:02:00 Quarter Active Camera 1 View +0.0
Smoke Stack
0:00:02:00
00048 (24.00 fps)
Source Name Parent
Auto-traced Shapes
Mesh 1
Triangles 350
Expansion 3.0
Deform
Puppet Pin 2
...tion 2024.0,2632.0
Puppet Pin 1
...tion 1942.5,1844.3
Express...
effect("Puppet").arap.mesh("Mesh 1").deform("Puppet Pin 1").position.loopOut()
Toggle Switches / Modes

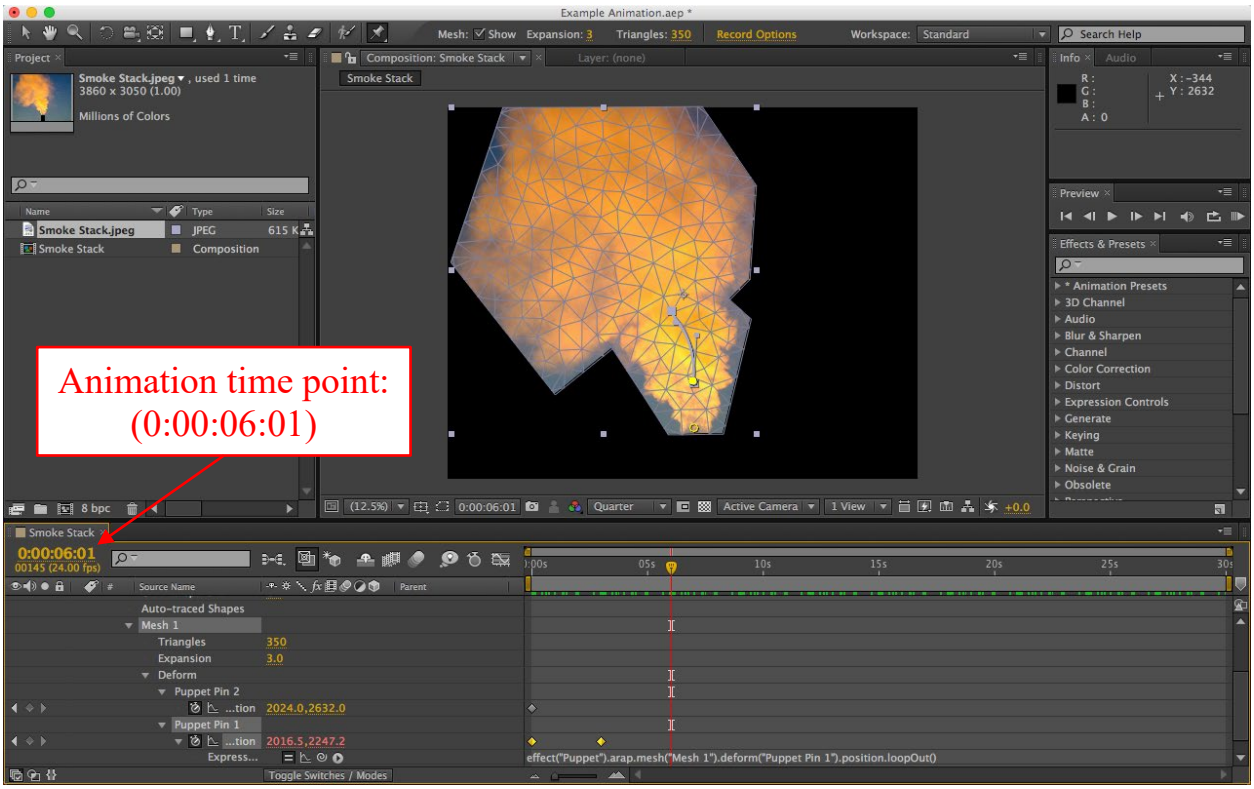

Example Animation.aep *
Mesh: Show Expansion: 3 Triangles: 350 Record Options Workspace: Standard
Search Help
Project
Smoke Stack.jpeg ▾ , used 1 time
3860 x 3050 (1.00)
Millions of Colors
Name Type Size
Smoke Stack.jpeg JPEG 615 K
Smoke Stack Composition
Composition: Smoke Stack
Layer: (none)
Smoke Stack
Info Audio
R :
G :
B :
A : 0
X : -344
Y : 2632
Preview
Effects & Presets
* Animation Presets
3D Channel
Audio
Blur & Sharpen
Channel
Color Correction
Distort
Expression Controls
Generate
Keying
Matte
Noise & Grain
Obsolete
Animation time point:
(0:00:02:23)
8 bpc
(12.5%) 0:00:02:23 Quarter Active Camera 1 View +0.0
Smoke Stack
0:00:02:23
00071 (24.00 fps)
Source Name Parent
Auto-traced Shapes
Mesh 1
Triangles 350
Expansion 3.0
Deform
Puppet Pin 2
...tion 2024.0,2632.0
Puppet Pin 1
...tion 1834.9,1672.3
Express...
effect("Puppet").arap.mesh("Mesh 1").deform("Puppet Pin 1").position.loopOut()
Toggle Switches / Modes


Example Animation.aep *
Mesh: Show Expansion: 3 Triangles: 350 Record Options Workspace: Standard
Smoke Stack.jpeg, used 1 time
3860 x 3050 (1.00)
Millions of Colors
Composition: Smoke Stack
Smoke Stack
Smoke Stack.jpeg JPEG 615 K
Smoke Stack Composition
Animation time point:
(0:00:03:01)
0:00:03:01
00073 (24.00 fps)
Auto-traced Shapes
Mesh 1
Triangles 350
Expansion 3.0
Deform
Puppet Pin 2
...tion 2024.0,2632.0
Puppet Pin 1
...tion 2016.5,2247.2
effect("Puppet").arap.mesh("Mesh 1").deform("Puppet Pin 1").position.loopOut()


Example Animation.aep *
Mesh: Show Expansion: 3 Triangles: 350 Record Options Workspace: Standard
Smoke Stack.jpeg, used 1 time
3860 x 3050 (1.00)
Millions of Colors
Composition: Smoke Stack
Smoke Stack
Smoke Stack.jpeg JPEG 615 K
Smoke Stack Composition
Animation time point:
(0:00:04:00)
0:00:04:00
00096 (24.00 fps)
Auto-traced Shapes
Mesh 1
Triangles 350
Expansion 3.0
Deform
Puppet Pin 2
...tion 2024.0,2632.0
Puppet Pin 1
...tion 2008.4,2044.6
effect("Puppet").arap.mesh("Mesh 1").deform("Puppet Pin 1").position.loopOut()


Example Animation.aep *
Mesh: Show Expansion: 3 Triangles: 350 Record Options Workspace: Standard
Search Help
Project
Smoke Stack.jpeg , used 1 time
3860 x 3050 (1.00)
Millions of Colors
Composition: Smoke Stack
Layer: (none)
Smoke Stack
Info
Audio
R :
G :
B :
A : 0
X : -344
Y : 2632
Preview
Effects & Presets
* Animation Presets
3D Channel
Audio
Blur & Sharpen
Channel
Color Correction
Distort
Expression Controls
Generate
Keying
Matte
Noise & Grain
Obsolete
Name
Type
Size
Smoke Stack.jpeg
JPEG
615 K
Smoke Stack
Composition
Animation time point:
(0:00:05:00)
8 bpc
(12.5%)
0:00:05:00
Quarter
Active Camera
1 View
+0.0
0:00:05:00
00120 (24.00 fps)
Source Name
Parent
Auto-traced Shapes
Mesh 1
Triangles 350
Expansion 3.0
Deform
Puppet Pin 2
...tion 2024.0,2632.0
Puppet Pin 1
...tion 1942.5,1844.3
Express...
effect("Puppet").arap.mesh("Mesh 1").deform("Puppet Pin 1").position.loopOut()
Toggle Switches / Modes

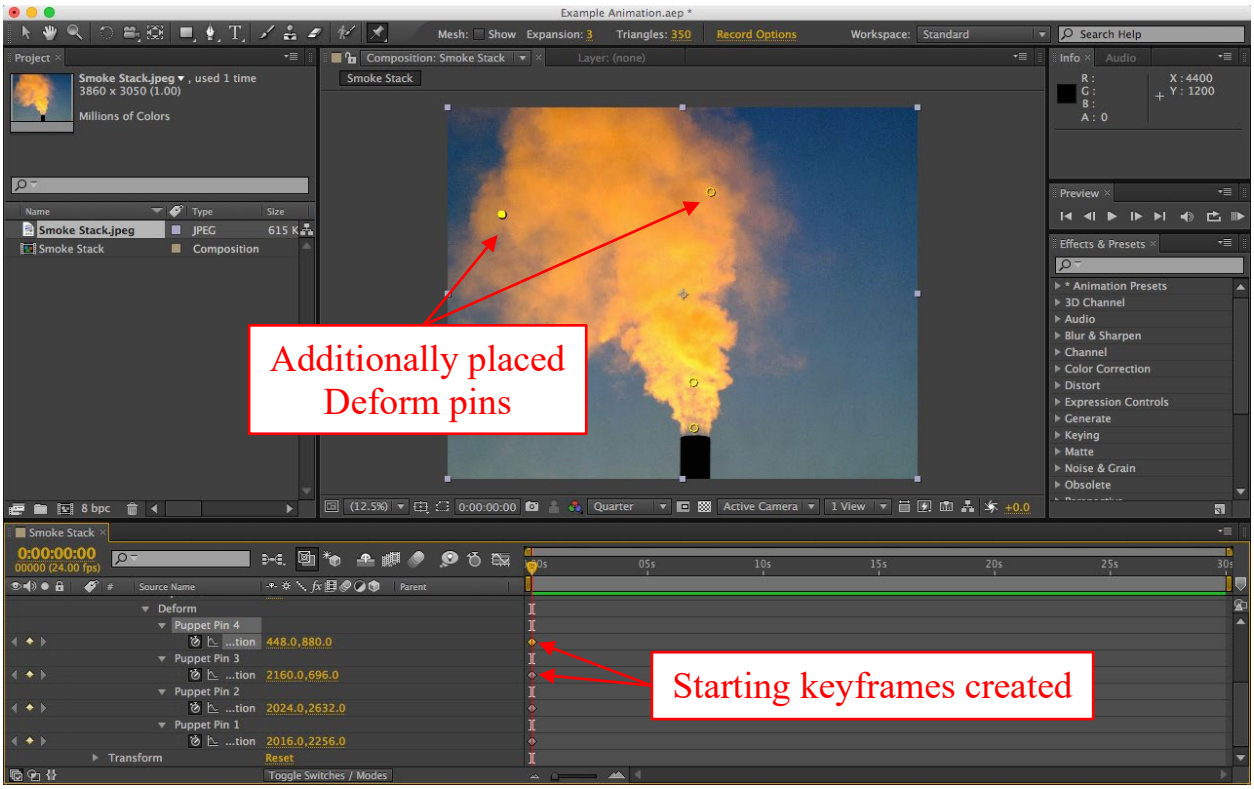

Example Animation.aep *
Mesh: Show Expansion: 3 Triangles: 350 Record Options Workspace: Standard
Search Help
Project
Smoke Stack.jpeg , used 1 time
3860 x 3050 (1.00)
Millions of Colors
Composition: Smoke Stack
Layer: (none)
Smoke Stack
Info
Audio
R :
G :
B :
A : 0
X : -344
Y : 2632
Preview
Effects & Presets
* Animation Presets
3D Channel
Audio
Blur & Sharpen
Channel
Color Correction
Distort
Expression Controls
Generate
Keying
Matte
Noise & Grain
Obsolete
Name
Type
Size
Smoke Stack.jpeg
JPEG
615 K
Smoke Stack
Composition
Animation time point:
(0:00:05:23)
8 bpc
(12.5%)
0:00:05:23
Quarter
Active Camera
1 View
+0.0
0:00:05:23
00143 (24.00 fps)
Source Name
Parent
Auto-traced Shapes
Mesh 1
Triangles 350
Expansion 3.0
Deform
Puppet Pin 2
...tion 2024.0,2632.0
Puppet Pin 1
...tion 1834.9,1672.3
Express...
effect("Puppet").arap.mesh("Mesh 1").deform("Puppet Pin 1").position.loopOut()
Toggle Switches / Modes



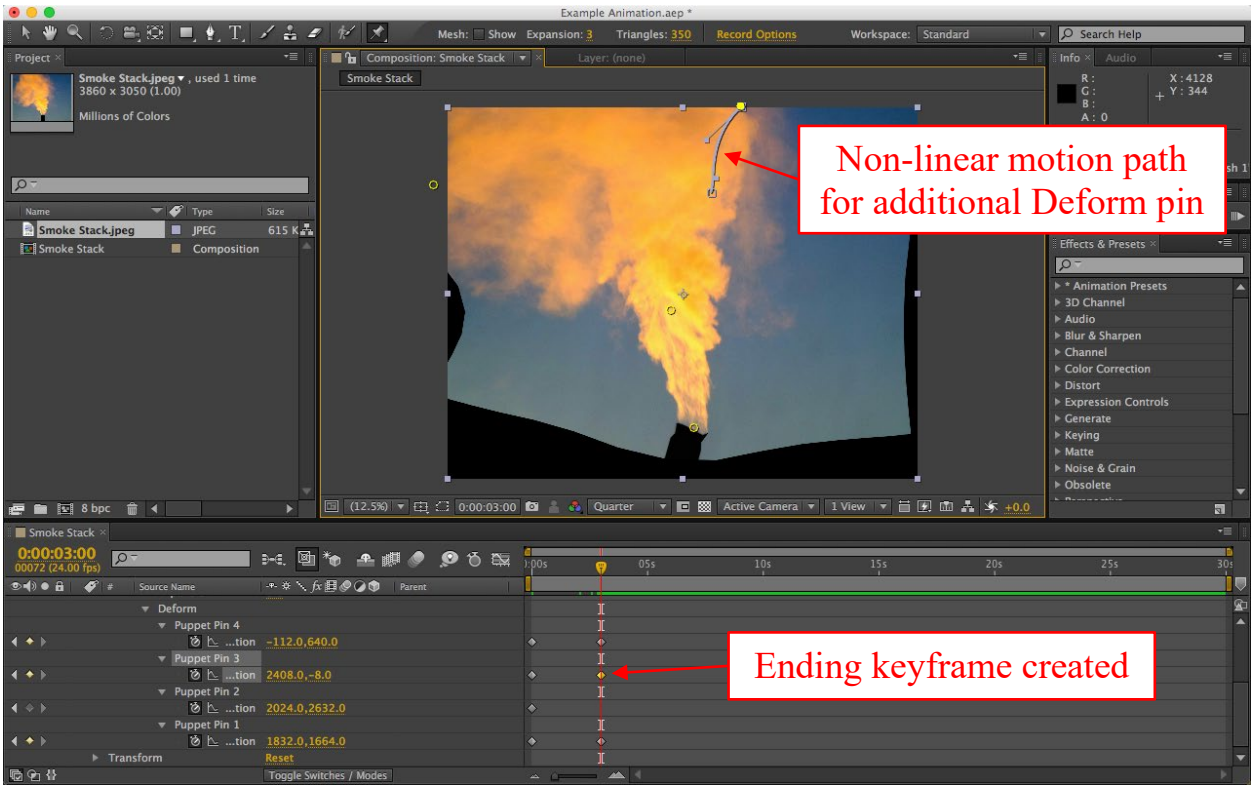

### h. Animating with Multiple Deform Pins

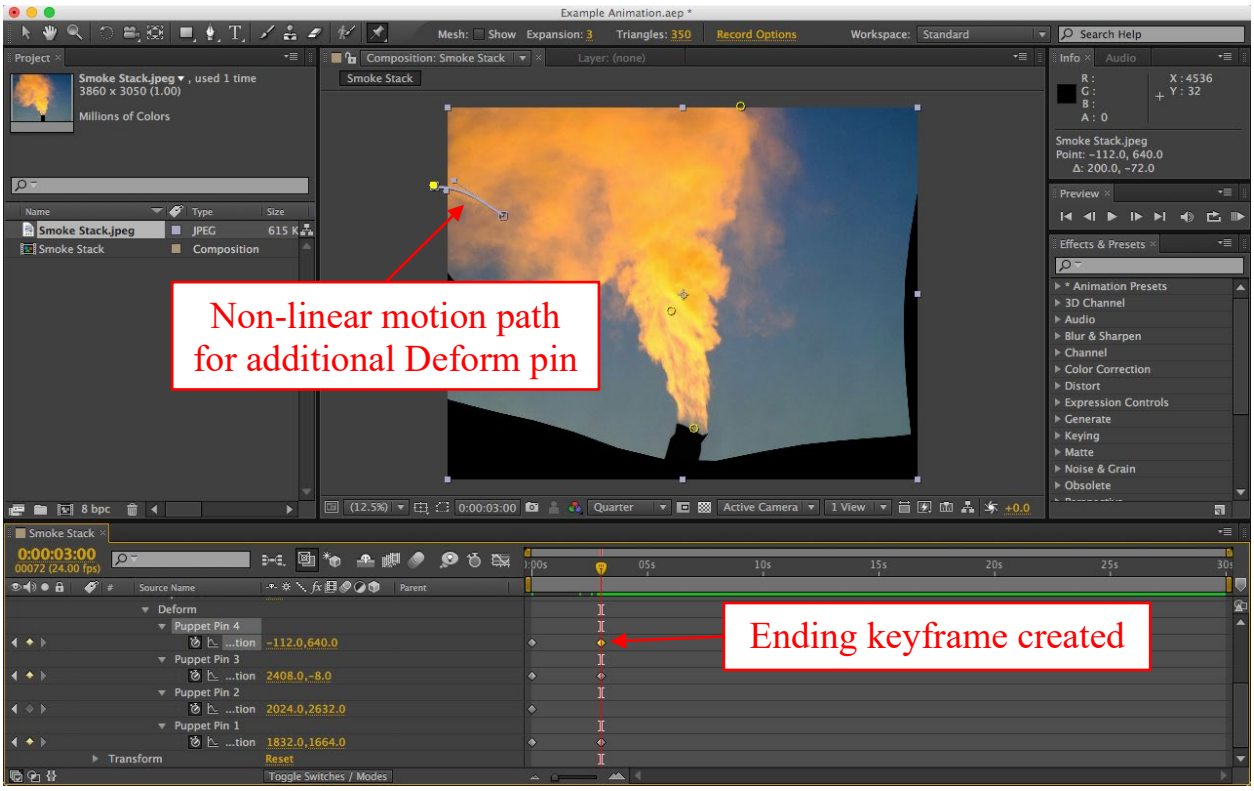
95. Although in the above exemplary composition only one Deform pin is specified to move during the animation, additional Deform pins can also be applied and moved in the same way, as shown in the below screenshots. *See* AEM, 218-19.

96. 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.

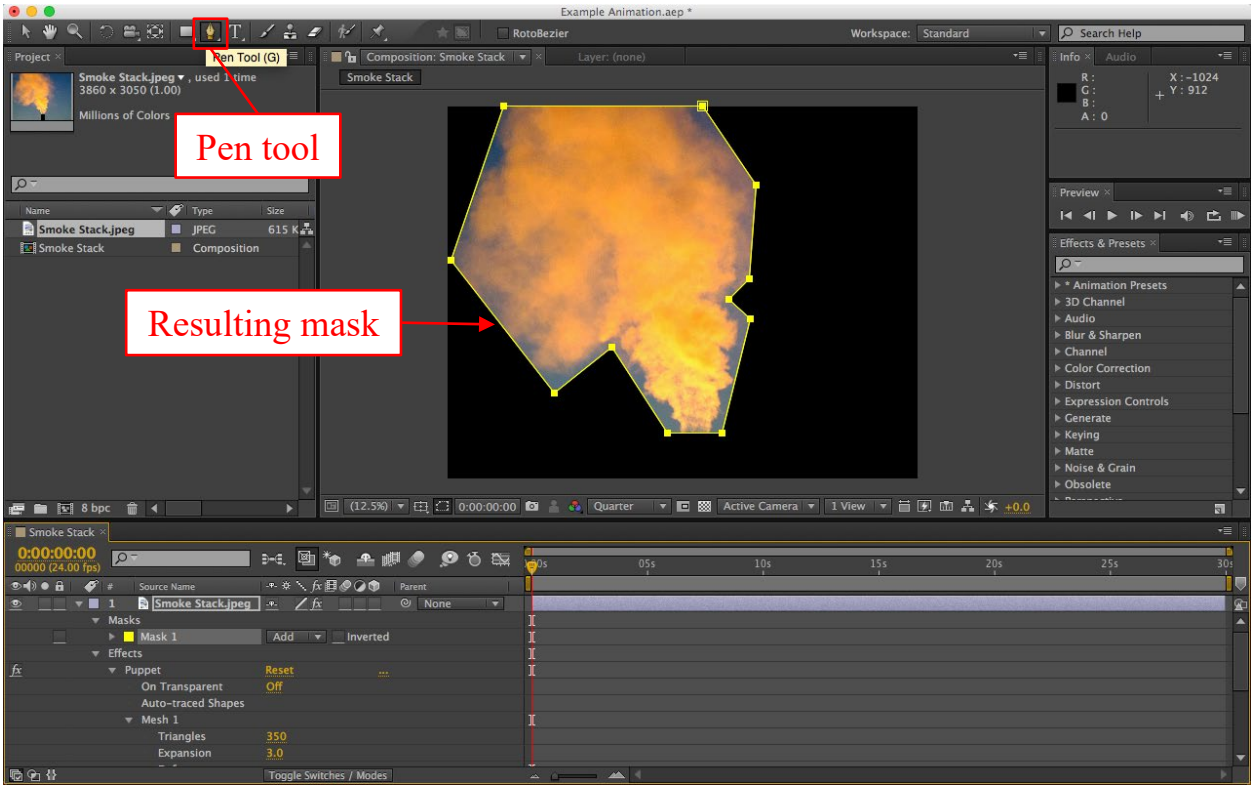



97. Next, ending keyframes and non-linear motion paths for each of the additional Deform pins are created in the same way as discussed in Section VIII.B.2.d, as shown below. *See* AEM, 219, 194-95.

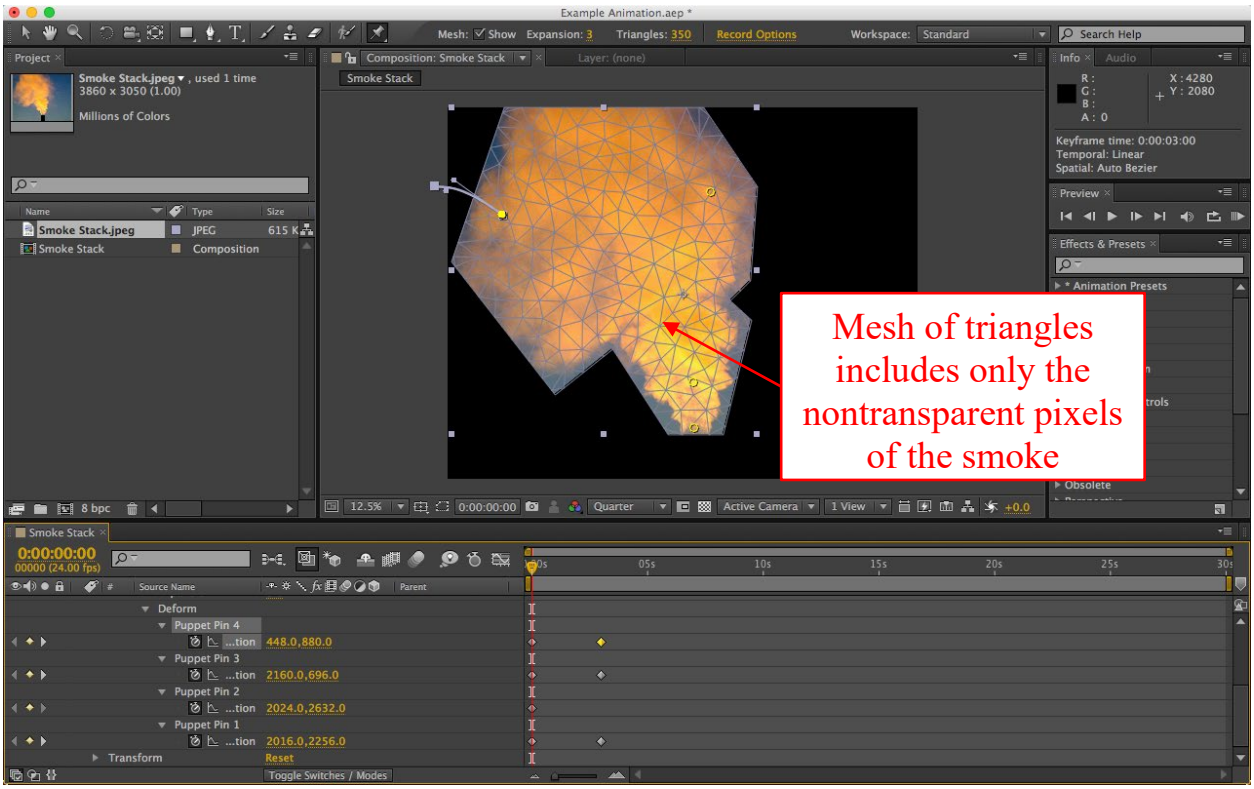

Non-linear motion path
for additional Deform pin
Ending keyframe created


Non-linear motion path
for additional Deform pin
Ending keyframe created

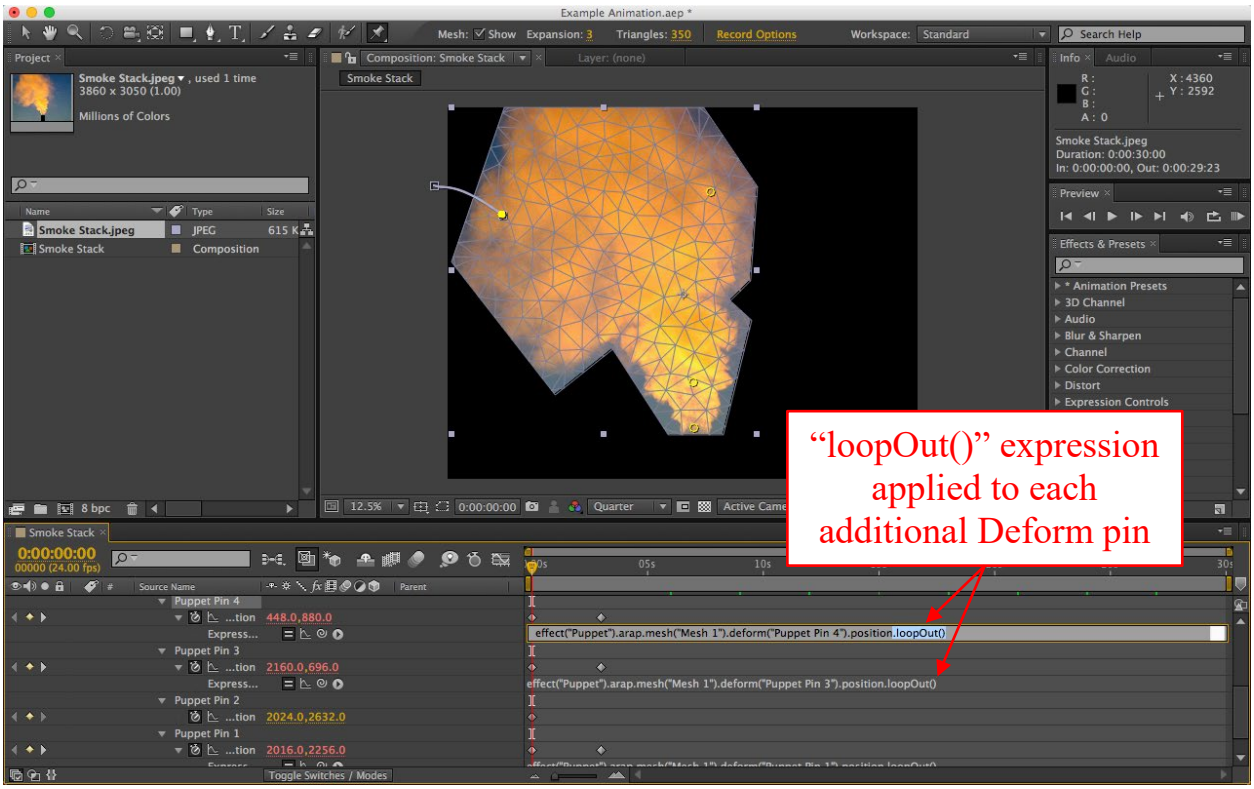
98. The mask from Section VIII.B.2.e.i is then likewise created and applied to the layer in the same way discussed therein to cause only the pixels of the billowing smoke to be nontransparent and thus animated, as shown in the screenshot below. *See* AEM, 318, 264-65.



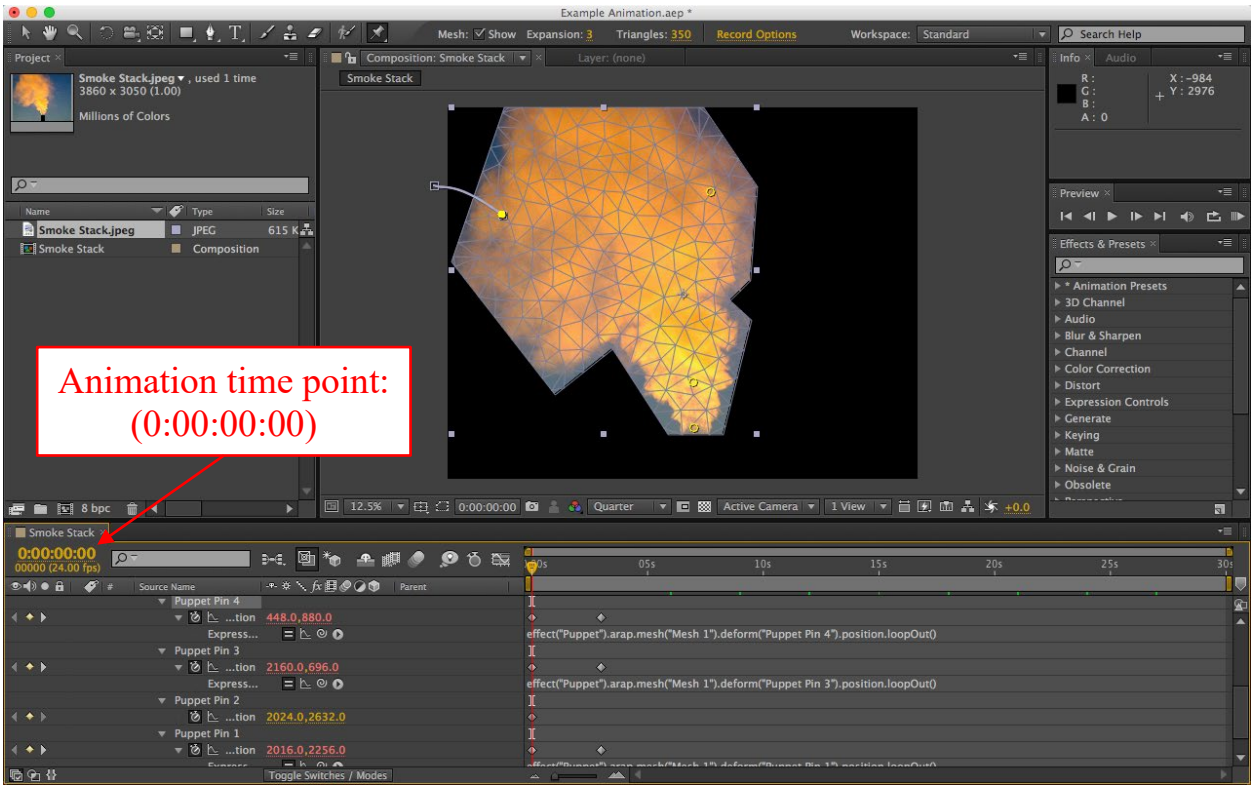

The below screenshot shows the "mesh of triangles" created from the smoke's pixels made nontransparent by this mask. *See* AEM, 219.

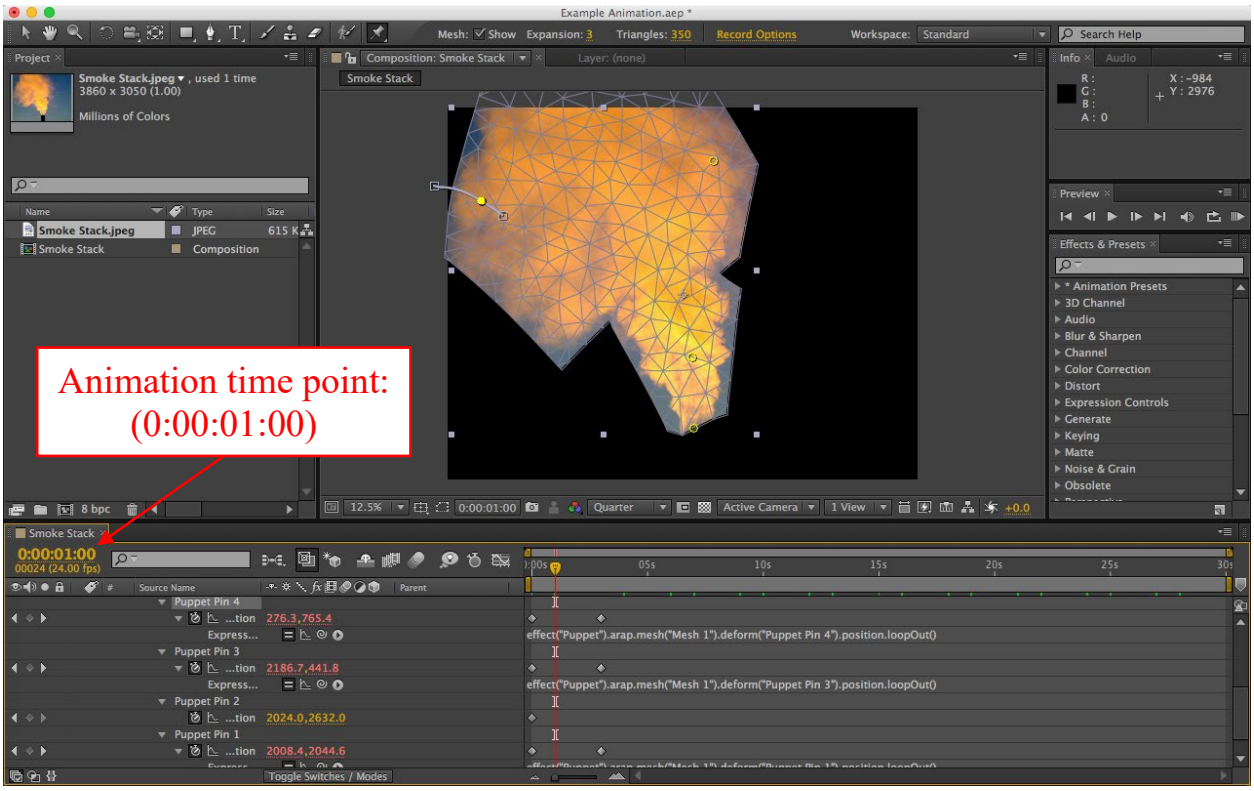


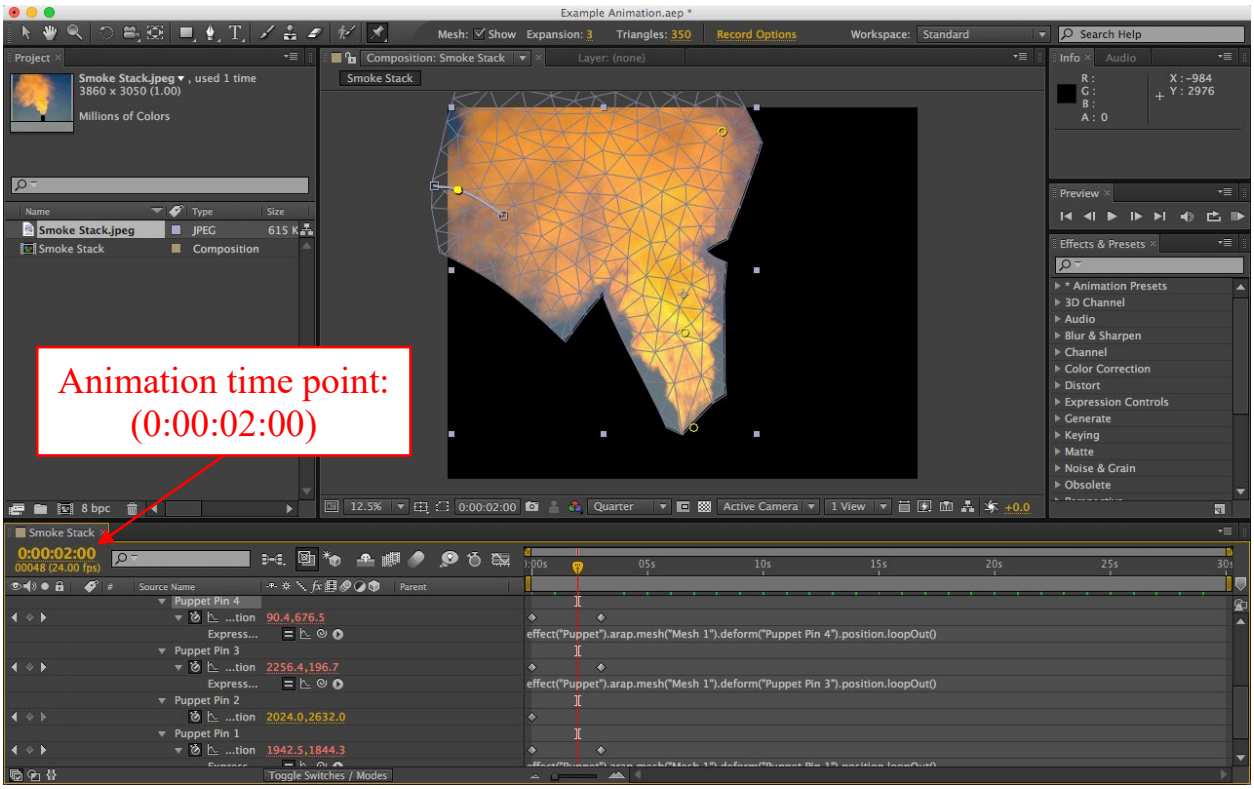

99. Finally, as shown below, a "loopOut()" expression is applied to each of the additional Deform pins as similarly discussed in Section VIII.B.2.f to cause the overall animation to repeatedly loop. *See* AEM, 562, 219.

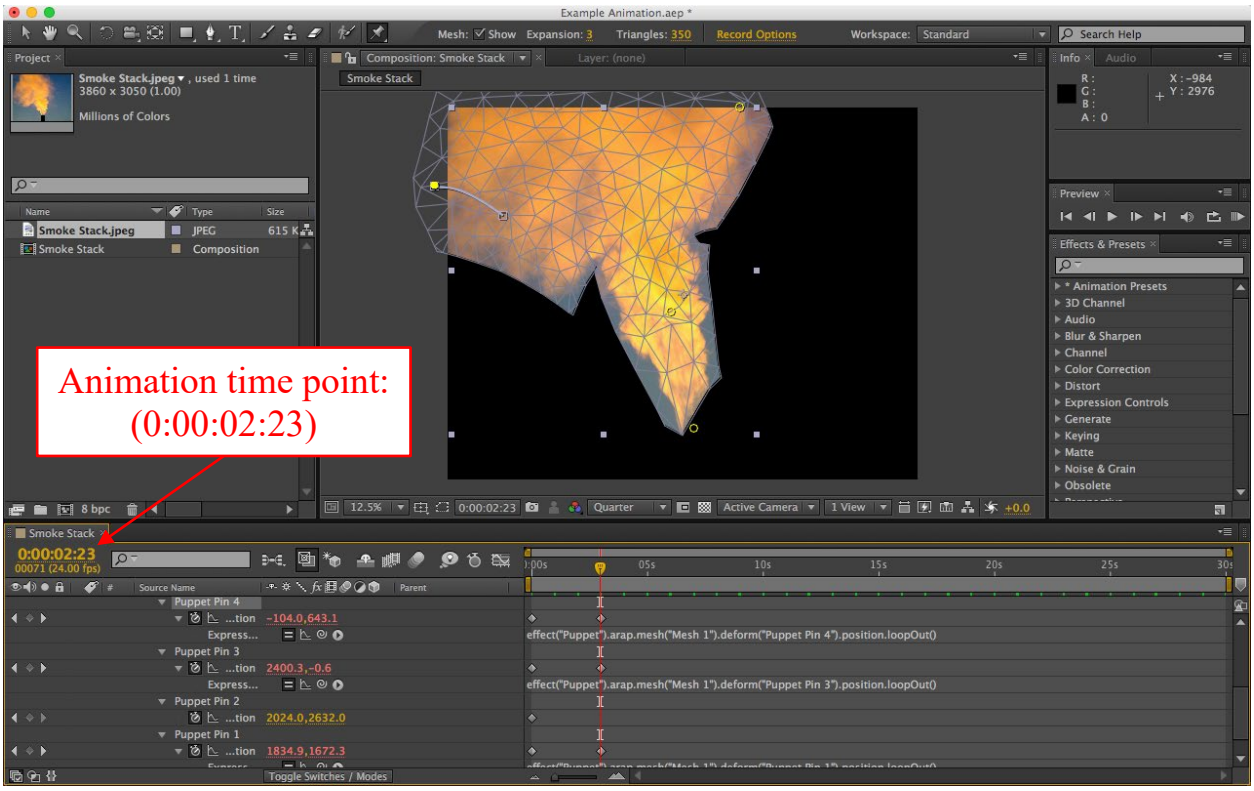


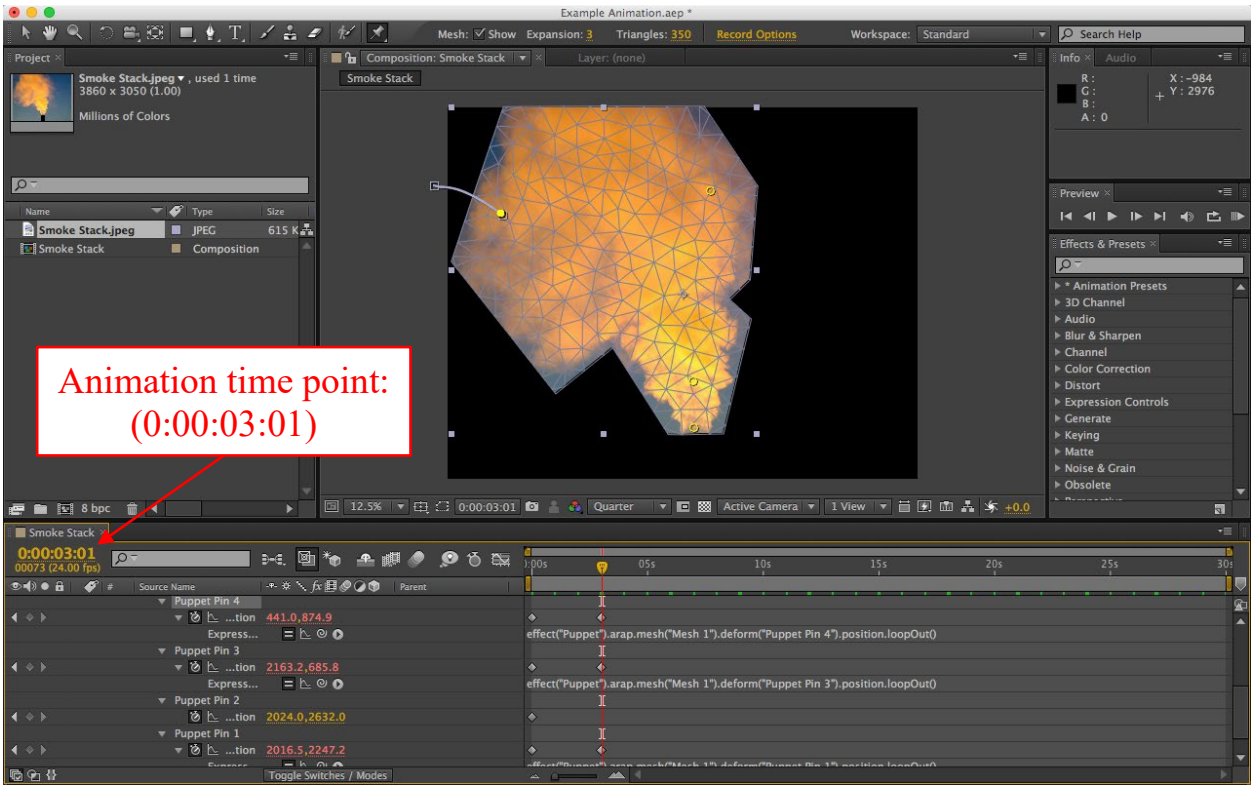

100. The final resulting animation is shown in the below screenshots, which follow the same sequential order as in Section VIII.B.2.g in order to mimic the animation playing in time—i.e., animation time points (0:00:00:00), (0:00:01:00), (0:00:02:00), (0:00:02:23), (0:00:03:01), (0:00:04:00), (0:00:05:00), (0:00:05:23), and (0:00:06:01). *See* Section VIII.B.2.g. 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.

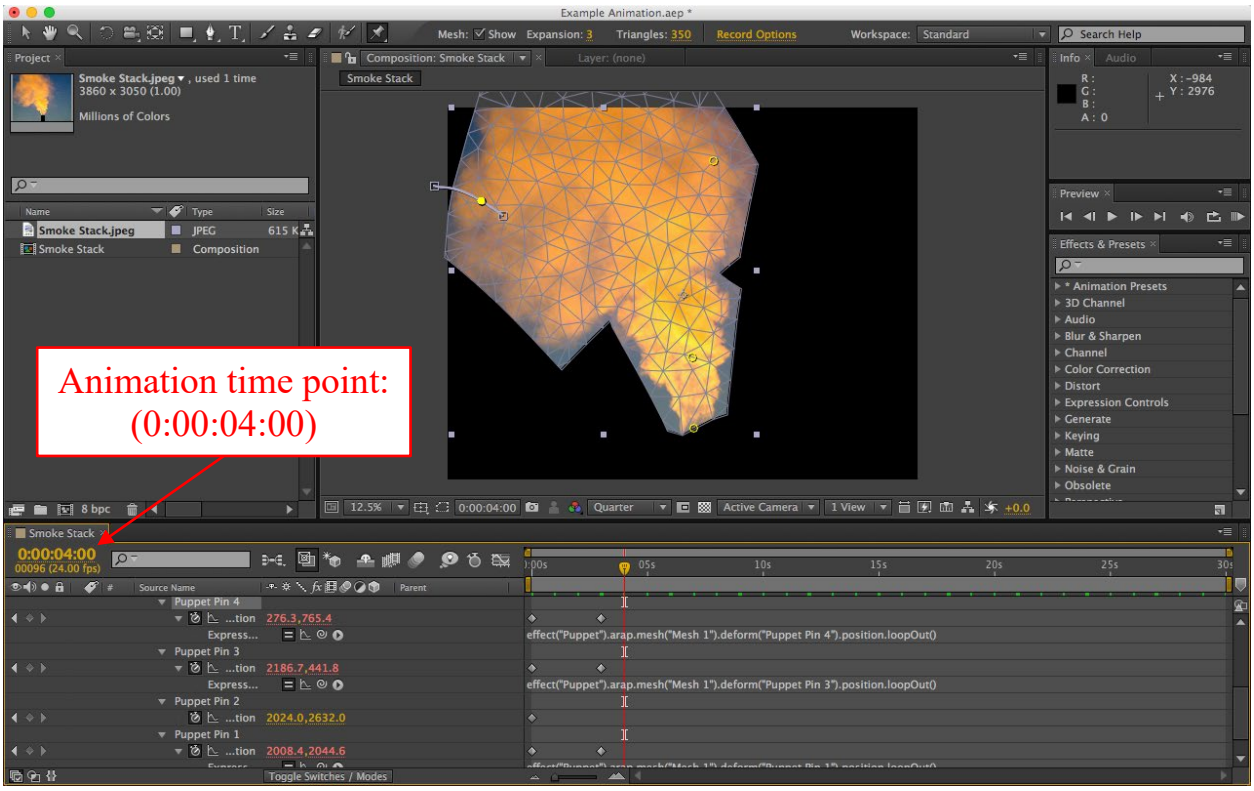

Example Animation.aep *
Mesh: Show Expansion: 3 Triangles: 350 Record Options Workspace: Standard
Search Help
Project
Smoke Stack.jpeg ▾ , used 1 time
3860 x 3050 (1.00)
Millions of Colors
Name Type Size
Smoke Stack.jpeg JPEG 615 K
Smoke Stack Composition
Composition: Smoke Stack
Layer: (none)
Smoke Stack
Info Audio
R :
G :
B :
A : 0
X : -984
Y : 2976
Preview
Effects & Presets
* Animation Presets
3D Channel
Audio
Blur & Sharpen
Channel
Color Correction
Distort
Expression Controls
Generate
Keying
Matte
Noise & Grain
Obsolete
Animation time point:
(0:00:00:00)
8 bpc
12.5% 0:00:00:00 Quarter Active Camera 1 View +0.0
Smoke Stack
0:00:00:00
00000 (24.00 fps)
Source Name Parent
Puppet Pin 4
...tion 448.0,880.0
Express...
effect("Puppet").arap.mesh("Mesh 1").deform("Puppet Pin 4").position.loopOut()
Puppet Pin 3
...tion 2160.0,696.0
Express...
effect("Puppet").arap.mesh("Mesh 1").deform("Puppet Pin 3").position.loopOut()
Puppet Pin 2
...tion 2024.0,2632.0
Puppet Pin 1
...tion 2016.0,2256.0
Toggle Switches / Modes

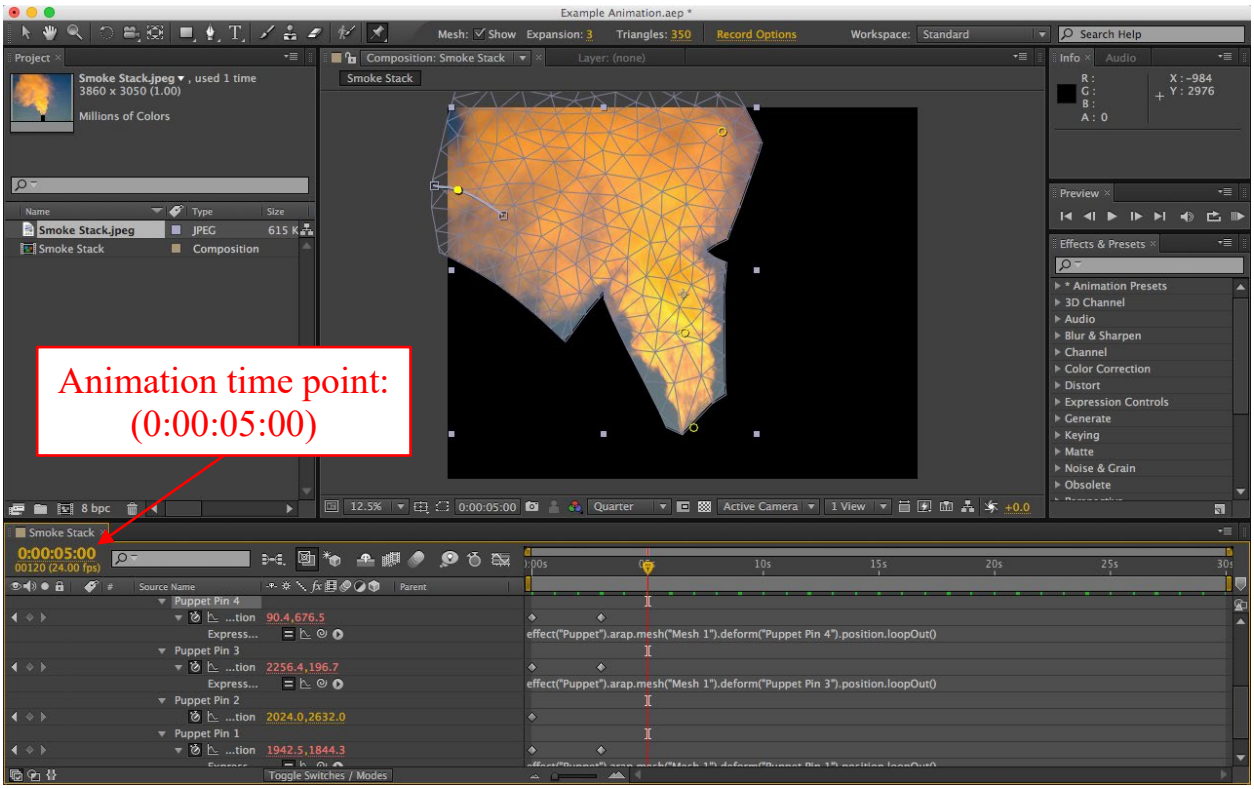

Example Animation.aep *
Mesh: Show Expansion: 3 Triangles: 350 Record Options Workspace: Standard
Search Help
Project
Smoke Stack.jpeg ▾ , used 1 time
3860 x 3050 (1.00)
Millions of Colors
Name Type Size
Smoke Stack.jpeg JPEG 615 K
Smoke Stack Composition
Composition: Smoke Stack
Layer: (none)
Smoke Stack
Info Audio
R :
G :
B :
A : 0
X : -984
Y : 2976
Preview
Effects & Presets
* Animation Presets
3D Channel
Audio
Blur & Sharpen
Channel
Color Correction
Distort
Expression Controls
Generate
Keying
Matte
Noise & Grain
Obsolete
Animation time point:
(0:00:01:00)
8 bpc
12.5% 0:00:01:00 Quarter Active Camera 1 View +0.0
Smoke Stack
0:00:01:00
00024 (24.00 fps)
Source Name Parent
Puppet Pin 4
...tion 276.3,765.4
Express...
effect("Puppet").arap.mesh("Mesh 1").deform("Puppet Pin 4").position.loopOut()
Puppet Pin 3
...tion 2186.7,441.8
Express...
effect("Puppet").arap.mesh("Mesh 1").deform("Puppet Pin 3").position.loopOut()
Puppet Pin 2
...tion 2024.0,2632.0
Puppet Pin 1
...tion 2008.4,2044.6
Toggle Switches / Modes

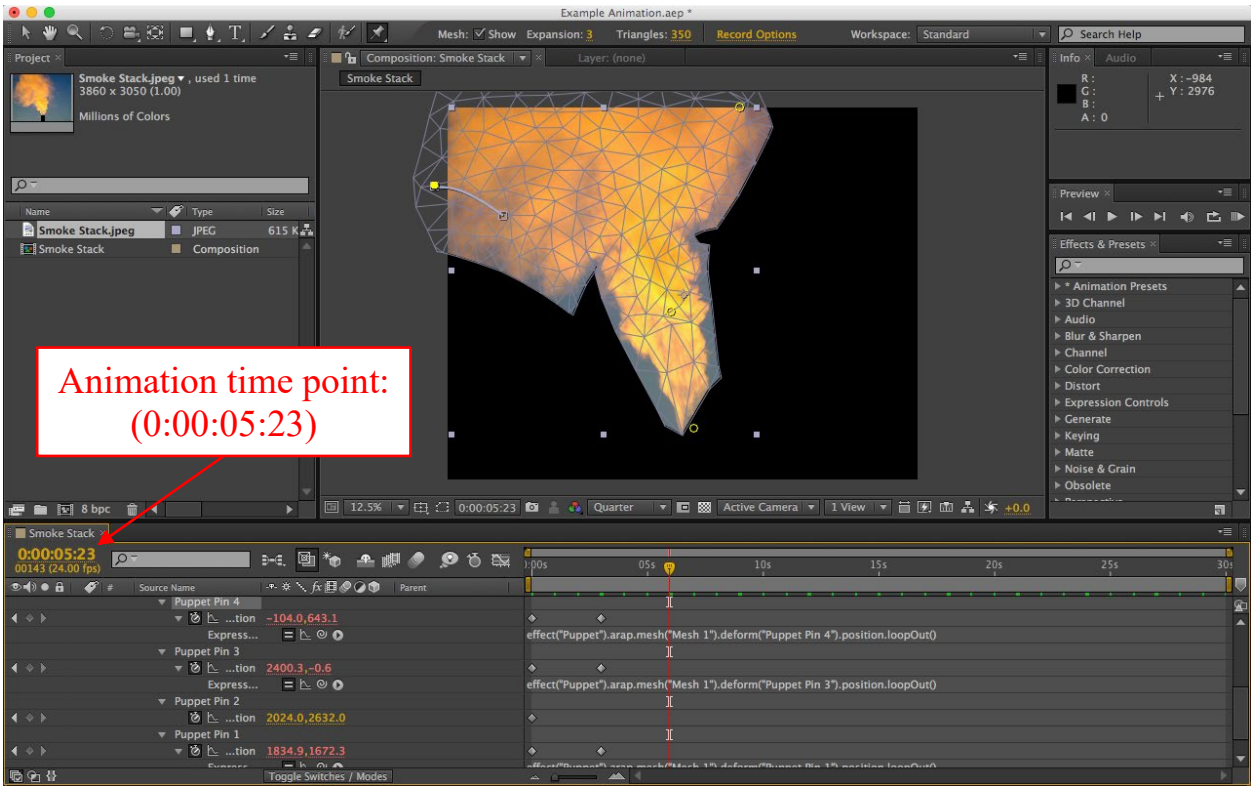

Example Animation.aep *
Mesh: Show Expansion: 3 Triangles: 350 Record Options Workspace: Standard
Search Help
Project
Smoke Stack.jpeg ▾ , used 1 time
3860 x 3050 (1.00)
Millions of Colors
Name Type Size
Smoke Stack.jpeg JPEG 615 K
Smoke Stack Composition
Composition: Smoke Stack
Layer: (none)
Smoke Stack
Info Audio
R :
G :
B :
A : 0
X : -984
Y : 2976
Preview
Effects & Presets
* Animation Presets
3D Channel
Audio
Blur & Sharpen
Channel
Color Correction
Distort
Expression Controls
Generate
Keying
Matte
Noise & Grain
Obsolete
Animation time point:
(0:00:02:00)
8 bpc
12.5% 0:00:02:00 Quarter Active Camera 1 View +0.0
Smoke Stack
0:00:02:00
00048 (24.00 fps)
Source Name Parent
Puppet Pin 4
...tion 90.4,676.5
Express...
effect("Puppet").arap.mesh("Mesh 1").deform("Puppet Pin 4").position.loopOut()
Puppet Pin 3
...tion 2256.4,196.7
Express...
effect("Puppet").arap.mesh("Mesh 1").deform("Puppet Pin 3").position.loopOut()
Puppet Pin 2
...tion 2024.0,2632.0
Puppet Pin 1
...tion 1942.5,1844.3
Toggle Switches / Modes

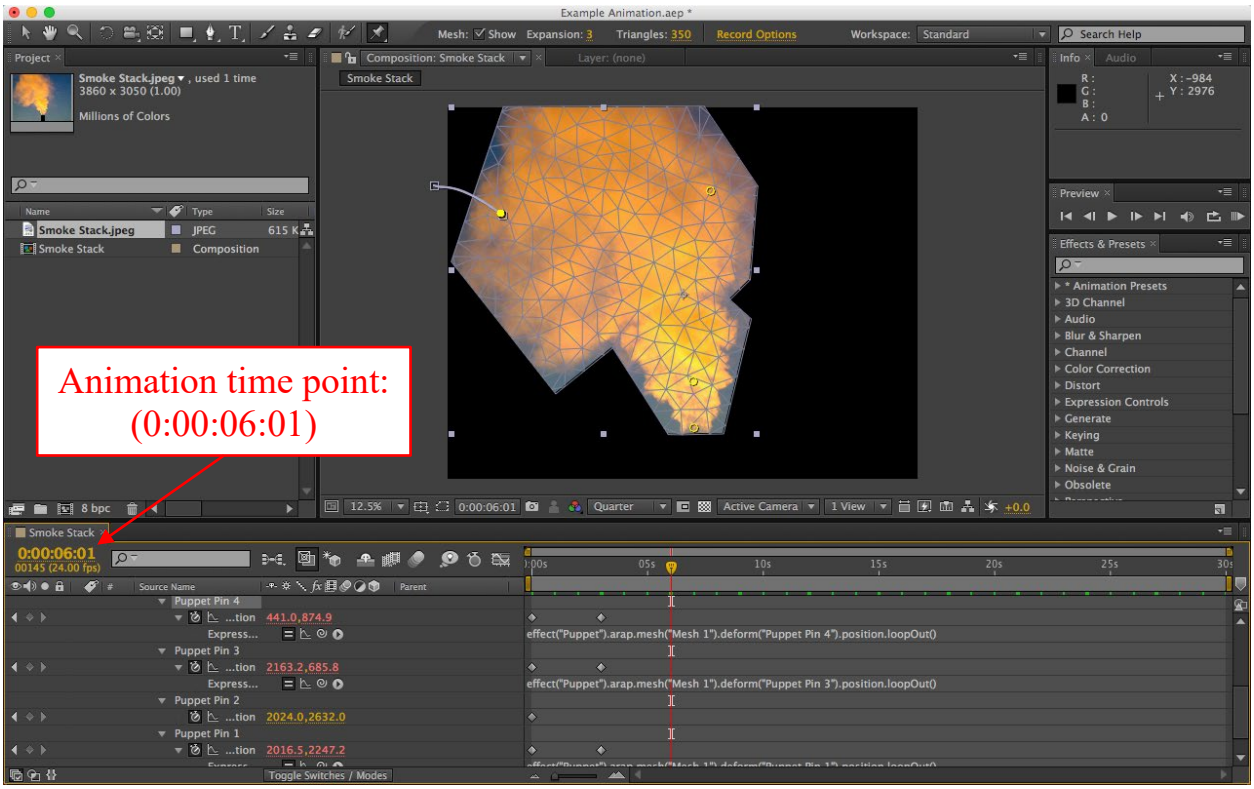

Example Animation.aep *
Mesh: Show Expansion: 3 Triangles: 350 Record Options Workspace: Standard
Search Help
Project
Smoke Stack.jpeg ▾ , used 1 time
3860 x 3050 (1.00)
Millions of Colors
Name Type Size
Smoke Stack.jpeg JPEG 615 K
Smoke Stack Composition
Composition: Smoke Stack
Layer: (none)
Smoke Stack
Info Audio
R :
G :
B :
A : 0
X : -984
Y : 2976
Preview
Effects & Presets
* Animation Presets
3D Channel
Audio
Blur & Sharpen
Channel
Color Correction
Distort
Expression Controls
Generate
Keying
Matte
Noise & Grain
Obsolete
Animation time point:
(0:00:02:23)
8 bpc
12.5% 0:00:02:23 Quarter Active Camera 1 View +0.0
Smoke Stack
0:00:02:23
00071 (24.00 fps)
Source Name Parent
Puppet Pin 4
...tion -104.0,643.1
Express...
effect("Puppet").arap.mesh("Mesh 1").deform("Puppet Pin 4").position.loopOut()
Puppet Pin 3
...tion 2400.3,-0.6
Express...
effect("Puppet").arap.mesh("Mesh 1").deform("Puppet Pin 3").position.loopOut()
Puppet Pin 2
...tion 2024.0,2632.0
Puppet Pin 1
...tion 1834.9,1672.3
Toggle Switches / Modes


Example Animation.aep *
Animation time point:
(0:00:03:01)
0:00:03:01
00073 (24.00 fps)
Puppet Pin 4
441.0,874.9
Puppet Pin 3
2163.2,685.8
Puppet Pin 2
2024.0,2632.0
Puppet Pin 1
2016.5,2247.2


Example Animation.aep *
Animation time point:
(0:00:04:00)
0:00:04:00
00096 (24.00 fps)
Puppet Pin 4
276.3,765.4
Puppet Pin 3
2186.7,441.8
Puppet Pin 2
2024.0,2632.0
Puppet Pin 1
2008.4,2044.6


Example Animation.aep *
Animation time point:
(0:00:05:00)
0:00:05:00
00120 (24.00 fps)
Puppet Pin 4
90.4,676.5
Puppet Pin 3
2256.4,196.7
Puppet Pin 2
2024.0,2632.0
Puppet Pin 1
1942.5,1844.3


Example Animation.aep *
Animation time point:
(0:00:05:23)
0:00:05:23
00143 (24.00 fps)
Puppet Pin 4
-104.0,643.1
Puppet Pin 3
2400.3,-0.6
Puppet Pin 2
2024.0,2632.0
Puppet Pin 1
1834.9,1672.3


Example Animation.aep *
Mesh: Show Expansion: 3 Triangles: 350 Record Options
Workspace: Standard
Search Help
Project
Smoke Stack.jpeg ▾ , used 1 time
3860 x 3050 (1.00)
Millions of Colors
Name Type Size
Smoke Stack.jpeg JPEG 615 K
Smoke Stack Composition
Composition: Smoke Stack
Layer: (none)
Smoke Stack
Info Audio
R :
G :
B :
A : 0
X : -984
Y : 2976
Preview
Effects & Presets
* Animation Presets
3D Channel
Audio
Blur & Sharpen
Channel
Color Correction
Distort
Expression Controls
Generate
Keying
Matte
Noise & Grain
Obsolete
Animation time point:
(0:00:06:01)
8 bpc
12.5% 0:00:06:01 Quarter Active Camera 1 View +0.0
Smoke Stack
0:00:06:01
00145 (24.00 fps)
Source Name Parent
Puppet Pin 4
...tion 441.0,874.9
Express...
effect("Puppet").arap.mesh("Mesh 1").deform("Puppet Pin 4").position.loopOut()
Puppet Pin 3
...tion 2163.2,685.8
Express...
effect("Puppet").arap.mesh("Mesh 1").deform("Puppet Pin 3").position.loopOut()
Puppet Pin 2
...tion 2024.0,2632.0
Puppet Pin 1
...tion 2016.5,2247.2
Toggle Switches / Modes

3. Independent Claim 1

**a. [1pre]: A computer system providing, to a client computing device, software for automating a shifting of pixels within a video file, the computer system comprising:**

**one or more processors; and**

**one or more computer-readable media having stored thereon executable instructions that are transmitted to the client computing device for execution by one or more client processors on the client computing device, the executable instructions comprising instructions that when executed by the one or more client processors configure the client computing device to perform at least the following.**

101. AEM teaches that AECS6 is "software" that is installed on a user's computer and run on "64-bit operating systems." AEM, 19. Such installation of AECS6 on a user's computer discloses the claimed "computer system" and remaining claim language according to Plotagraph's infringement contentions in *Plotagraph, Inc. v. Lightricks Ltd.*, Civil Action No. 4:21-cv-03873, Dkt. No. 42 (S.D. Tex. May 5, 2022), which simply assert that "[c]laim 1 is infringed when the claimed invention is made. The claimed computer system is made when a user downloads [Petitioner's] app or program to his/her smartphone, tablet or computer." *See* EX1022, 134.

102. Alternatively, AEM instructs the user to "[m]ake sure that you've installed the current version of [AECS6], including any available updates." AEM,

517. To view such updates, AEM instructs to "go to the Downloads section of the Adobe website," and provides a hyperlink for doing so. AEM, 517. A user following these instructions would have downloaded and installed the most up-to-date version of AECS6 (i.e., the claimed "software") onto the user's computer (i.e., "client computing device" with "one or more client processors") from Adobe's website, which would have been provided by a server (*see* Nakagawa, Abst.) (i.e., "computer system providing, to a client computing device, software" and comprising "one or more processors" and "one or more computer-readable media having stored thereon executable instructions that are transmitted to the client computing device for execution by one or more client processors").

103. Further, in my opinion, AEM describes using AECS6 to animate the movement of pixels within a single frame of a video (i.e., "automating a shifting of pixels within a video file"). Sections VIII.B.3.b-VIII.B.3.g.

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

104. In my opinion, AEM discloses limitation [1a]. AEM teaches importing video "footage items" from a "local disk drive" into a composition in AECS6—i.e., the claimed "access, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within a frame of the digital image file." AEM, 75, 98-100 (listing "[v]ideo and animation formats"

of footage items supported for importing, including MOV, MPEG, and animated GIF),[12] 17; '641 Patent, claim 3; *see also* AEM, 90 ("Compositions and footage items are listed in the project panel."), 121 ("You can create a layer from any footage item in the Project panel…" by "[s]electing one or more footage items and folders in the Project panel" and "[d]rag[ging] the selected footage items to the Composition panel.").

105. By way of further illustration, the annotated screenshot of AECS6 in Section VIII.B.2.a is provided below (with annotations removed), which shows an exemplary composition where a MOV video of a smokestack has been imported as a layer:

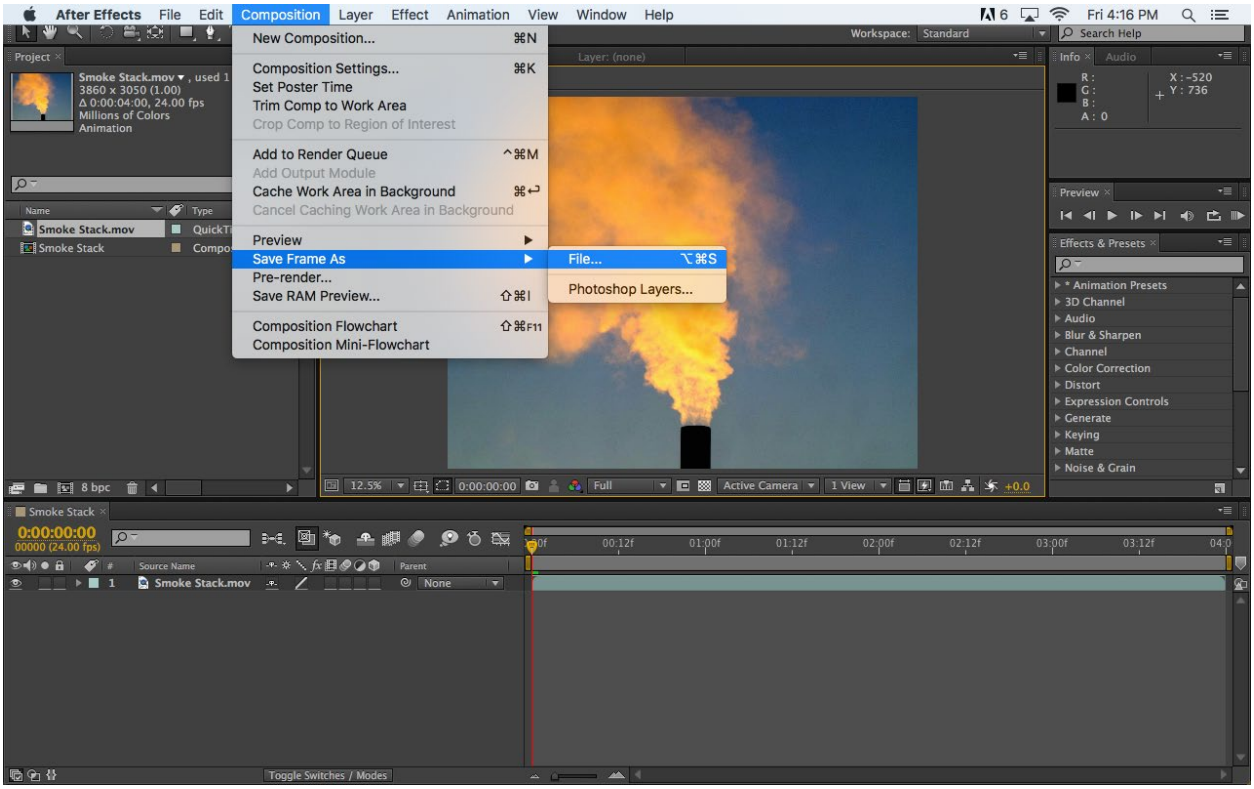
---

[12] A POSITA would have known that such video files "comprise[] information that corresponds to individual pixels within a frame" as claimed. *See* AEM, 104 ("Each motion-footage item in a composition can also have its own frame rate."), 105-08 ("[M]any video formats—including ITU-R 601 (D1) and DV—use non-square rectangular pixels."); Hair, 1:15-21 (stating "computer file formats for… digital video (hereinafter referred to as a 'Dynamic Video File')" include "the MPEG video file format"), 1:45-49 ("[M]otion picture quality Digital Video Files are generally composed of about 30 video frames (images) per second. Each of these video frames are composed of a two dimensional, usually rectangular or square, grid of pixels.").

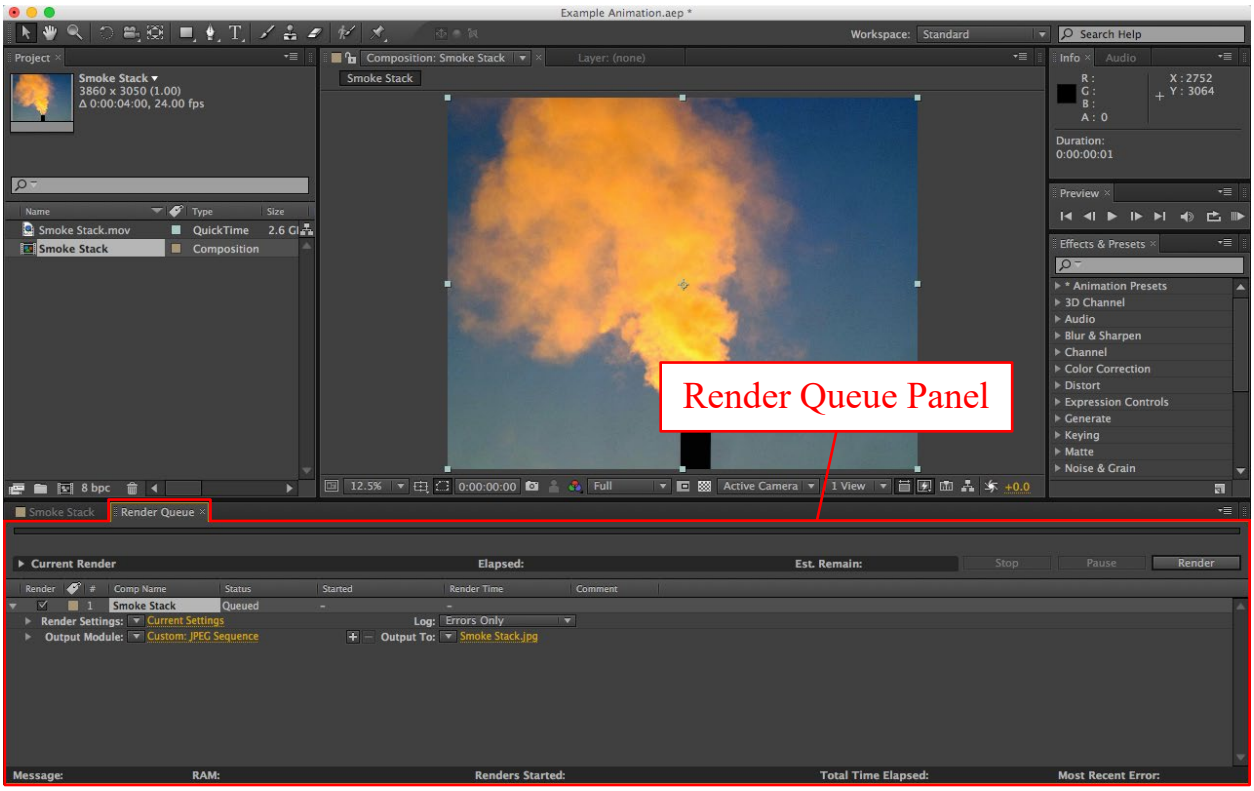
**c. [1b]: receive a first starting point through a user interface, wherein the first starting point is received through a user selection of a first beginning portion of a first image frame.**

106. In my opinion, AEM teaches limitation [1b]. Beginning with the recited "first image frame," AEM teaches, after importing a video into a composition, extracting a single frame from the composition (and thus the video)—i.e., the claimed "first image frame"—to be a new layer in the composition, which "is useful for," e.g., "exporting an image from a movie for posters or storyboards." AEM, 590. To do so, AEM instructs to "[g]o to the frame that you want to export so that it is shown in the Composition panel." AEM, 590. Then, "choose Composition > Save Frame As > File." AEM, 590.

107. To further illustrate, the first screenshot in Section VIII.B.2.b is provided below. The screenshot depicts the exemplary composition from Section VIII.B.3.b, where the first frame of the video (at "0:00:00:00") is now being extracted according to AEM's discussed instructions:

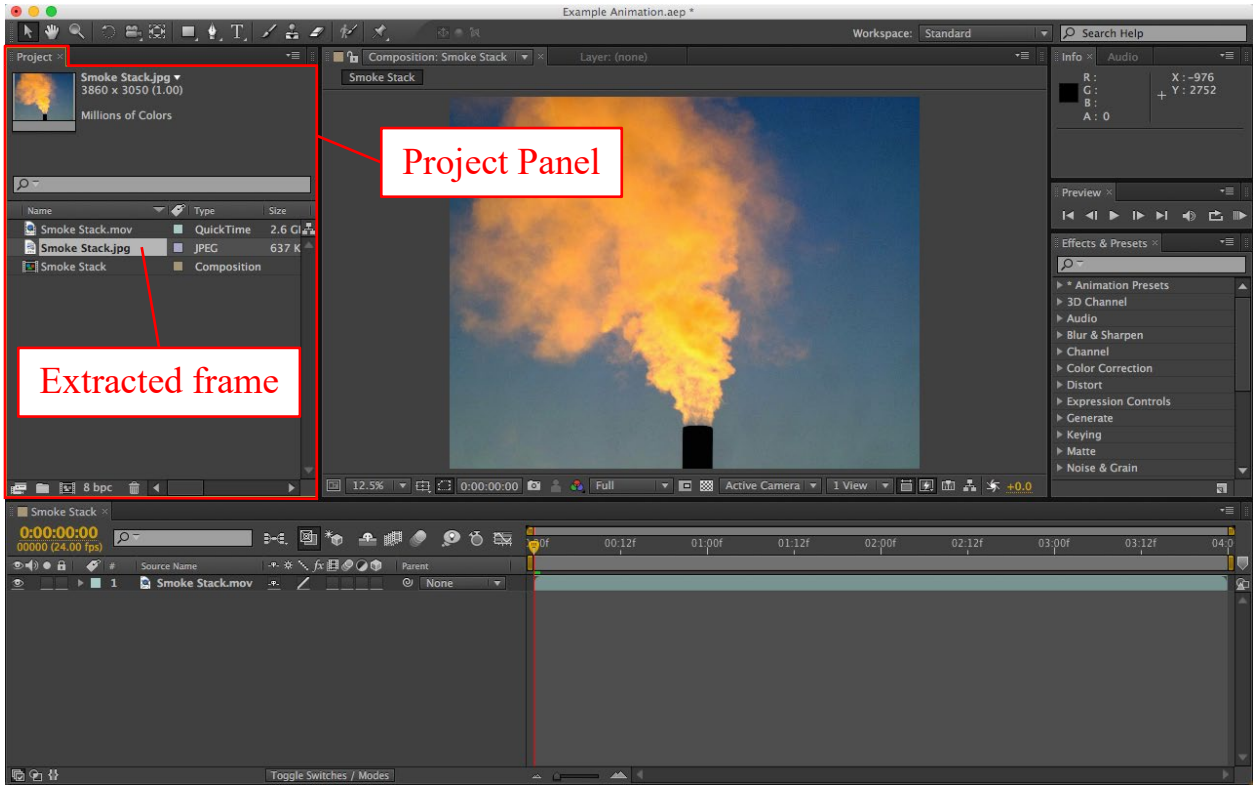



108. AEM also teaches managing the "render" and "output module" settings for the frame using the "Render Queue panel." AEM, 573-74. Section VIII.B.2.b's second screenshot, reproduced below with the added annotations, shows this Render Queue panel, which appears after clicking "Composition > Save Frame As > File" in the above screenshot:



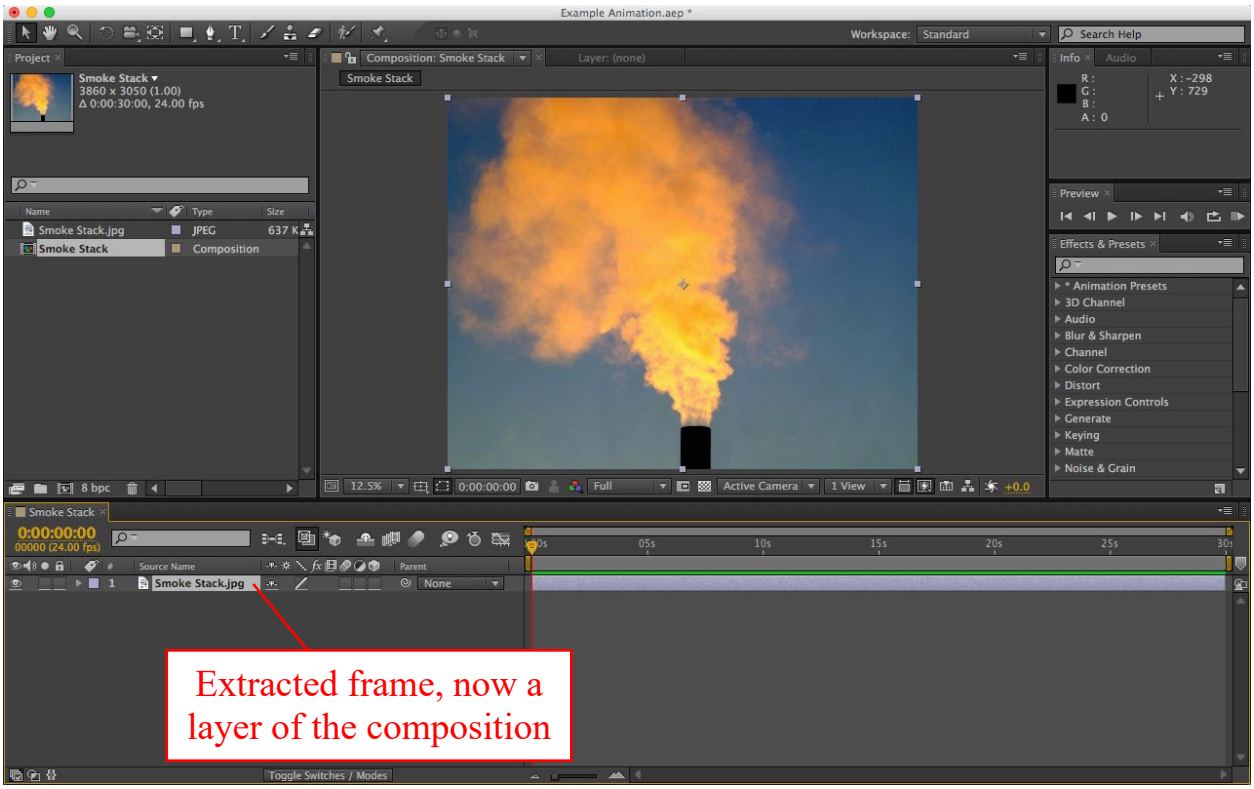

One such manageable setting in the Render Queue panel is "output format," which may be adjusted so the extracted frame can be used later as a footage file, e.g., by setting the output format to be JPEG, which is a "[s]till-image format[]" supported as both an output format and a footage item import format. *Compare* AEM, 573-74, *with* AEM, 98-101. Indeed, AEM teaches importing the extracted frame as a new footage item "by dragging its output module from the Render Queue panel into the Project panel," and then creating a layer from the frame by selecting the frame in the Project panel and "[d]rag[ging] the selected footage item[] to the Composition panel." AEM, 574-75, 121. Such provides "a convenient way to convert a footage item from one format to another," e.g., to extract and import a single frame from a video as a new layer in the composition. AEM, 574-75, 98-101, 590.

109. To further illustrate, the fourth screenshot in Section VIII.B.2.b, including the added annotations, is provided below, which shows the result of extracting the first frame of the above screenshot's video as a JPEG and dragging the frame's output module to the Project panel:




The frame is then selected and dragged to the Composition panel, causing the frame to become a new layer of the composition, as shown in the fifth screenshot of AECS6 in Section VIII.B.2.b, reproduced below with the added annotations[13]:

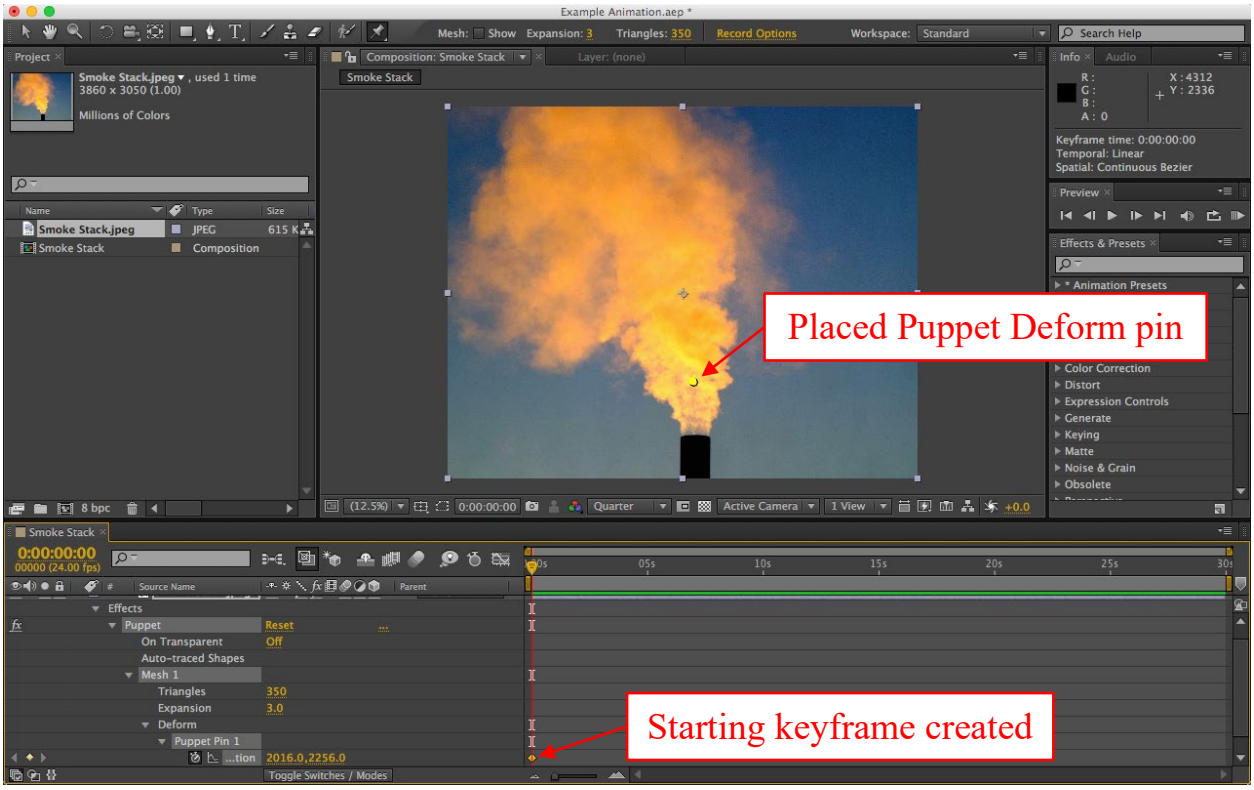
[13] As discussed in Section VIII.B.2.b, for purposes of clarity, the original video is deleted from the composition and Project panel as unnecessary.




110. In addition to the claimed "first image frame," AEM also, in my opinion, teaches the remaining elements of limitation [1b]. Specifically, AEM teaches creating a Puppet effect animation of the frame 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 animation time and the Deform pin's position—i.e., the claimed "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 a first image frame." AEM, 218-19; *see also* '641 Patent, 6:38-40 ("The beginning portion of the video frame comprises a starting ***pixel***, or area, from which the user wishes pixels to shift." (emphasis added)), 6:43-45 ("In at least

one embodiment, a beginning point comprise[s] a particular ***pixel*** that is selected….” (emphasis added)).

111. To further illustrate, the second annotated screenshot in Section VIII.B.2.c is provided below. The screenshot shows a Deform pin placed in the “Composition panel” on the extracted frame of the previous screenshots, and a starting keyframe automatically created as a result:



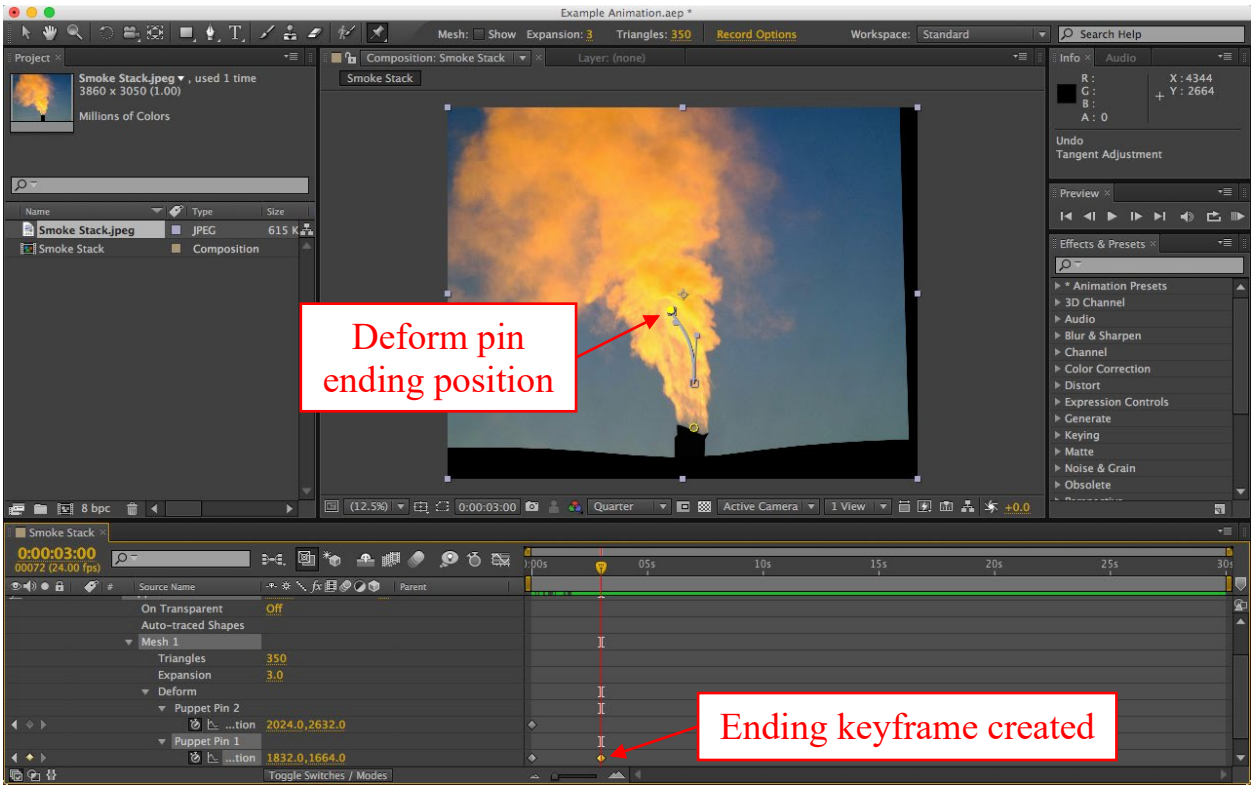

112. Further, should there be any argument that AEM discloses only creating a Puppet effect animation of an image and not specifically the video frame extracted according to the above, a POSITA would have nevertheless been motivated to use such an extracted frame as the digital image to be animated because such a frame would have been easily accessible and, in common instances, particularly desirable

for modifying and animating with AECS6's effects, e.g., the Puppet effect. *See, e.g.*, '641 Patent, 1:36-40 ("The increased ease with which video and images can be captured has led to an explosion in the amount of shared multimedia content."); AEM, 590 (stating that "export[ing] a single frame from a composition" is "useful" for, e.g., "exporting an image from a movie for posters or storyboards").

**d. [1c]: 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.**

113. In my opinion, AEM discloses limitation [1c]. 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 "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." AEM, 218-19; *see also* '641 Patent, 6:48-51 ("The ending portion of the video frame comprises an ending pixel, or area, to which the user wishes pixels to shift.").

114. To further illustrate, the below annotated screenshot depicts creating an ending keyframe for the Deform pin previously discussed and depicted in the fifth screenshot in Section VIII.B.3.c:




**e. [1d]: create a first digital link between the first starting point and the first ending point, wherein the first digital link comprises:**

**a first direction extending from the first starting point to the first ending point; and**

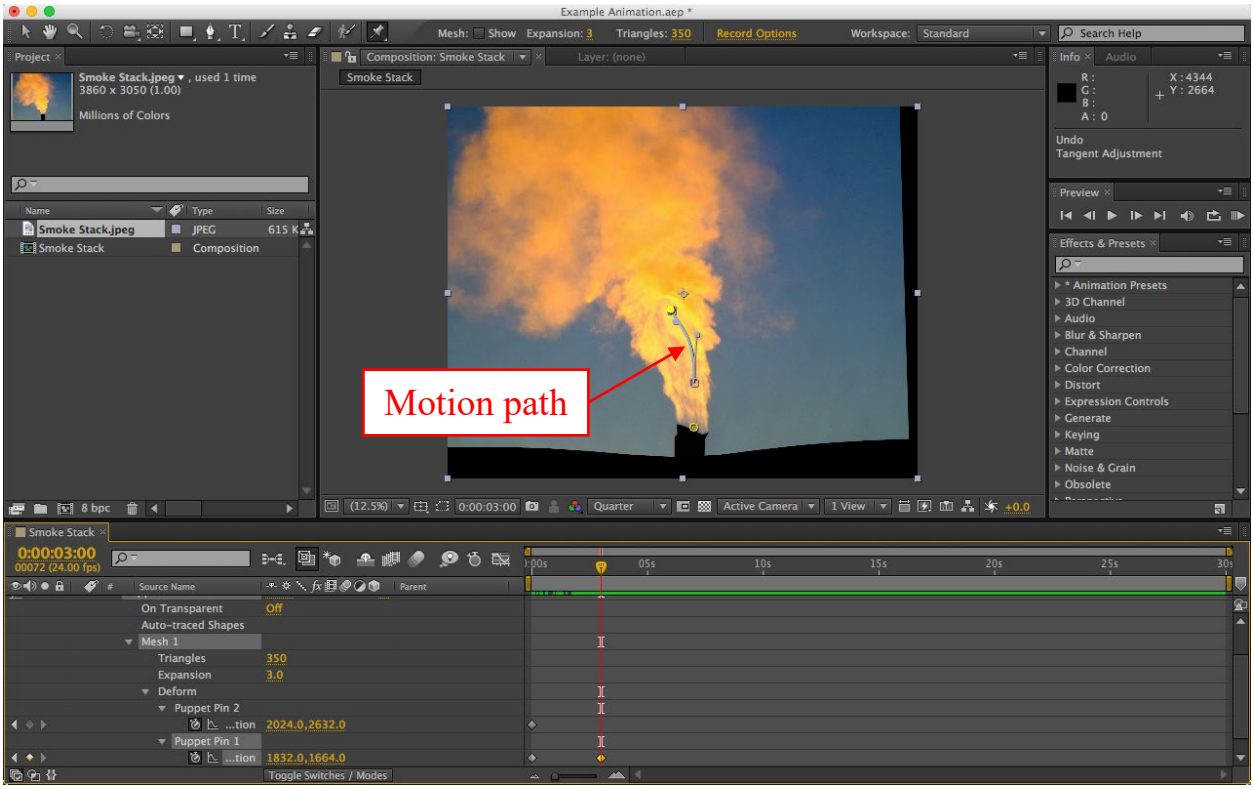
**a first length between the first starting point and the first ending point.**

115. AEM teaches that, when a Deform pin is initially placed and a starting keyframe is created, 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.

116. Further, the ending keyframe's creation prompts AECS6 to perform "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 by, a "motion path," which extends from the starting to ending position/keyframe—i.e., the claimed "create a first digital link between the first starting point and the first ending point, wherein the first digital 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." AEM, 187, 218-19.

117. To further illustrate, the screenshot in Section VIII.B.3.c is reproduced below, but with further annotations highlighting the motion path created between the Deform pin's starting and ending position/keyframe:




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

**f. [1e]: identify a first set of pixels that lie along the first digital link between the first starting point and the first ending point; and.**

118. Regarding the claimed "identify a first set of pixels," the '641 Patent states that "[t]he size of the group of pixels may be user selectable or automatically determined." '641 Patent, 7:28-29. For example, a user can perform the claimed "identify[ing]" of a "first set of pixels" by identifying certain pixels between the

starting and ending points to be "covered by a mask" and excluded from the set of pixels to be shifted. '641 Patent, 7:25-28, 4:24-36. 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." '641 Patent, 7:22-28.

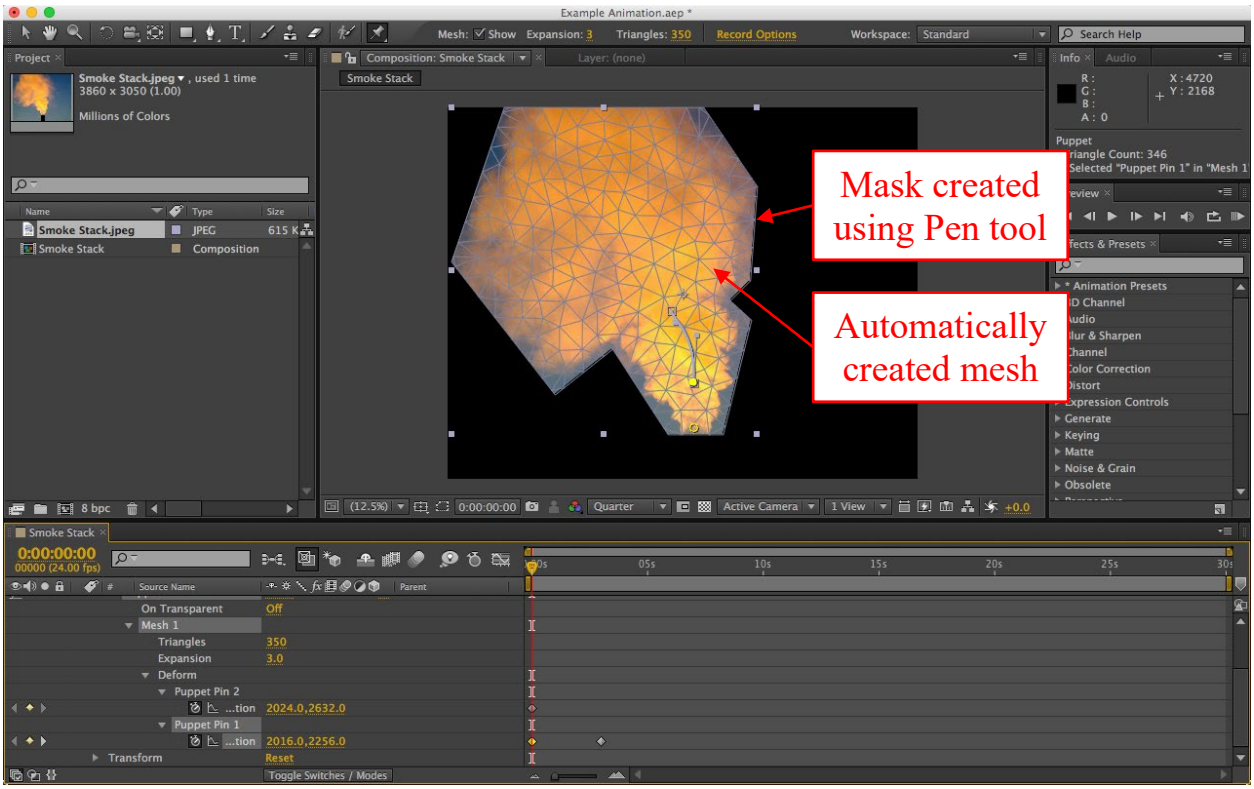
119. In my opinion, AEM discloses limitation [1e]. AEM teaches selecting certain pixels of the layer to be transparent or nontransparent (i.e., modifying the "alpha channel" of the layer) by applying either a "mask" or a "matte," where the former is created using, e.g., the "Pen" tool or "Auto-trace"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

120. Thus, as AEM teaches, AECS6 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 that lie along the Deform pin's motion path between the Deform pin's starting position/keyframe and ending position/keyframe—i.e., the claimed "lie along the first digital link between the first starting point and the first ending point." AEM, 218-19, 315, 194-95; '641 Patent, 7:28-29 ("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 [1e]. AEM, 218-19,

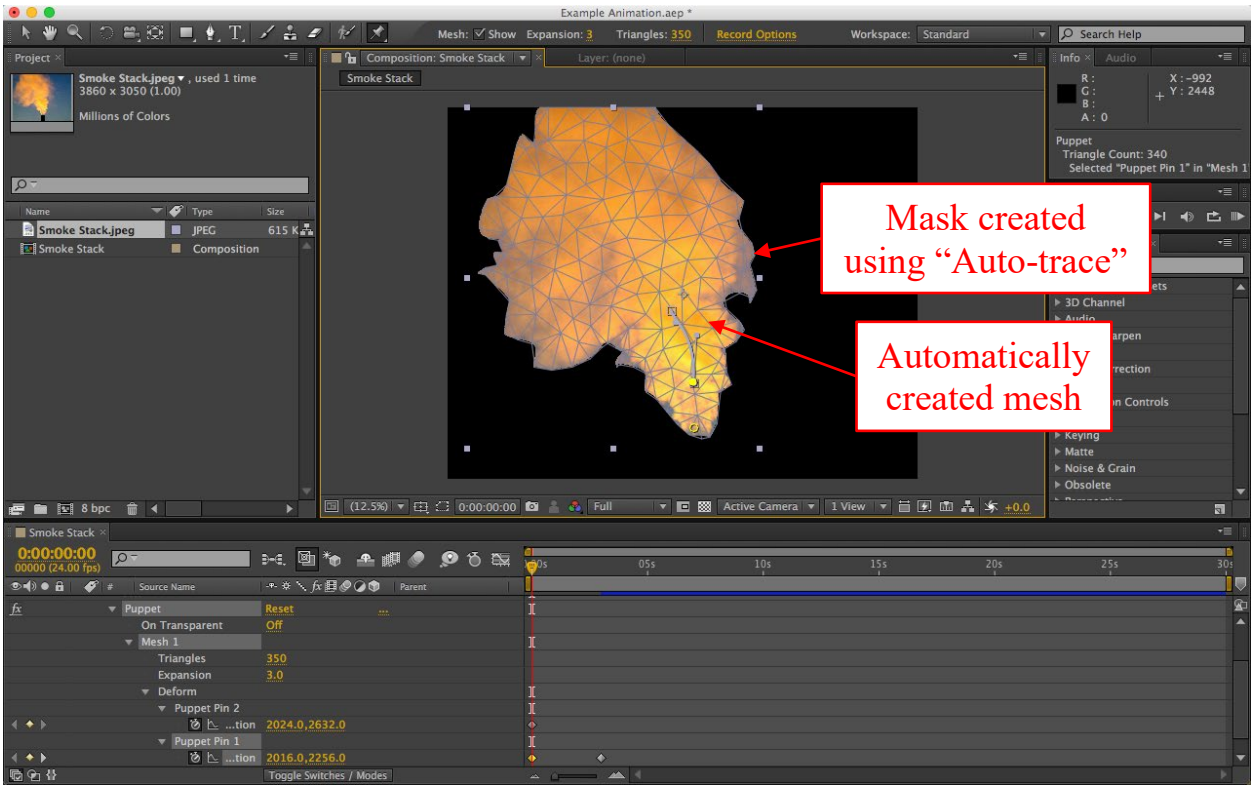
194-95; ’641 Patent, 7:28-29 (“The size of the group of pixels may be… ***automatically determined***.” (emphasis added)).

121. Separately, a POSITA would have been motivated to select and make nontransparent, using a mask or matte, those pixels that lie along the Deform pin’s motion path between the Deform pin’s starting position/keyframe and ending position/keyframe, 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 of the video frame 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 below[14]:

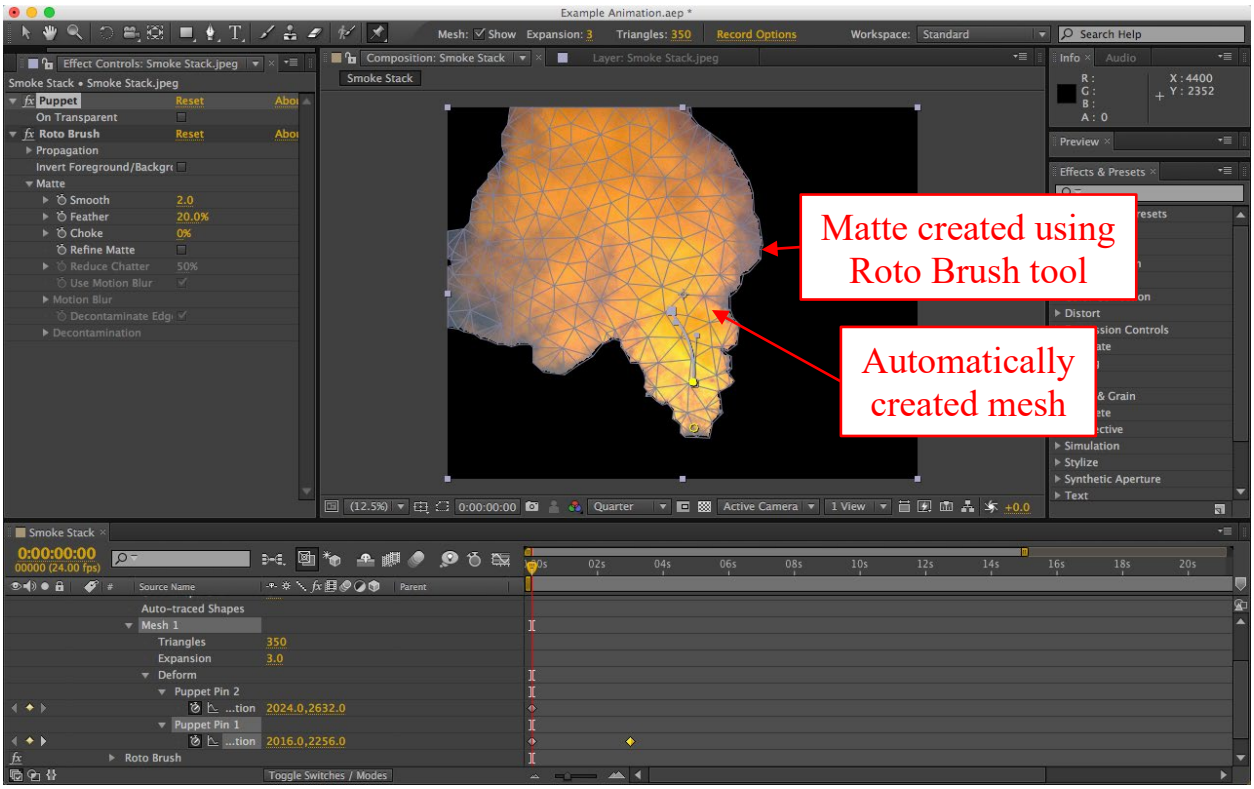
[14] Okabe provides further support by depicting a similar instance in its Figure 1, which depicts user-drawn “strokes” going downwards through a waterfall, and a “matte” specifying a to-be-animated “region of interest” (in white) that includes all pixels of the waterfall, including that lie along each stroke between each stroke’s starting point and ending point. Okabe, 7, 2; *see also* Section VIII.C.4.f.




As shown, the mask makes nontransparent, and thus the mesh includes, pixels that lie along the Deform pin's motion path between the Deform pin's starting position/keyframe and ending position/keyframe. 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.e.i, VIII.B.2.e.ii, and VIII.B.2.e.iii.

### g. [1f]: shift the first set of pixels in the first direction.

122. The ’641 Patent states the claimed pixel “shift[ing]” can use a “mesh algorithm” that “triangulates the mesh using defined points and then calculates an affine transformation for every angle.” ’641 Patent, 9:29-32. The ’641 Patent’s Figure 4 depicts an example, where “a starting mesh and ending mesh [are] overlaid on [a] video frame of [a] house and landscape”:

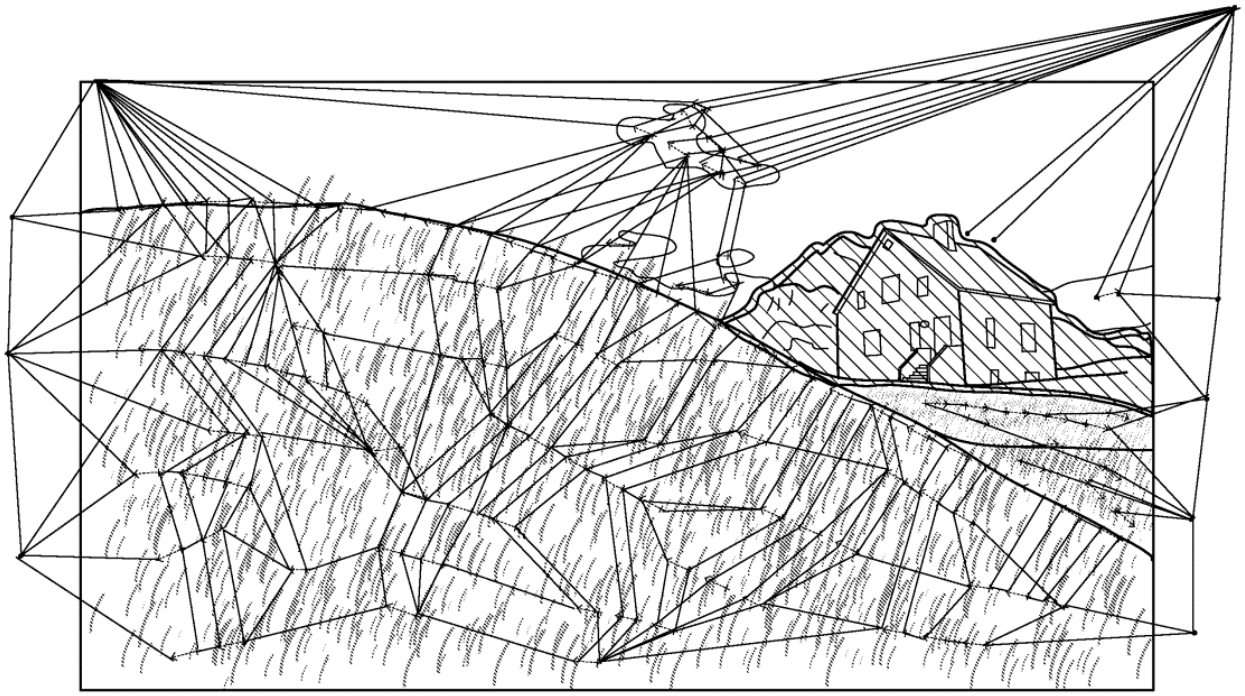


***Fig. 4***

’641 Patent, Fig. 4, 7:11-21.

123. In my opinion, AEM discloses limitation [1f]. AEM similarly teaches that, when a user places a Deform pin, AECS6 creates a mesh from the layer’s nontransparent pixels. AEM, 219, 220-21; Section VIII.B.3.e. And when a motion path for the Deform pin is also created, AECS6 animates the corresponding change to the shape of the layer’s mesh (which includes the claimed “first set of pixels,” *see* Section VIII.B.3.f), and thus the movement of the layer’s nontransparent pixels accommodating the Deform pin’s repositioning—i.e., “shift the first set of pixels in the first direction.” AEM, 218-19, 194-95; ’641 Patent, 9:29-32; Section VIII.B.3.e.

124. To further illustrate, the series of screenshots provided in Section VIII.B.2.g shows the resulting animation of the exemplary composition in Section

VIII.B.3.f's first screenshot at sequential time points to mimic the animation playing in time. There, a "loopOut()" expression (to be discussed further below in Section VIII.B.7) has also been added to the Deform pin to allow the animation to loop, but the first four screenshots in the series show the animation before the looping begins and are thus indicative of what the animation would look like when the "loopOut()" expression is not applied. Cropped versions for these screenshots are also provided in the table in Attachment B to provide a better view of the resulting animation.

4. Claim 2: **The computer system of claim 1, wherein the first ending portion comprises a particular portion of the first image frame.**

125. *See* Sections VIII.B.3.c-VIII.B.3.d. The ending position/keyframe for the Deform pin is created on the same video frame as that which includes the starting position/keyframe—i.e., "wherein the first ending portion comprises a particular portion of the first image frame."

5. Claim 3: **The computer system of claim 1, wherein the digital image file comprises a video file and the first image frame comprises a first video frame of the video file.**

126. *See* Section VIII.B.3.b. The discussed video footage items are "video file[s]" comprising "video frame[s]" as claimed.

6. Claim 4: **The computer system of claim 3, wherein the first ending portion comprises a particular portion of a second video frame within the video file.**

127. *See* Sections VIII.B.3.c-VIII.B.3.d, VIII.B.5. While the ending position/keyframe for the Deform pin is created on the same video frame as that

which includes the starting position/keyframe, such nevertheless discloses claim 4 because the claimed "first video frame" and "second video frame" may be the same frame. '641 Patent, 6:51-54; EX1022, 137.

7. Claim 8: **The computer system of claim 1, wherein shifting the first set of pixels comprises rendering in a loop the first set of pixels being shifted within the first image frame.**

128. Claim 8 is dependent from claim 1. In my opinion, AEM discloses claim 8. AEM teaches applying a "loopOut()" expression to a 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, such as the initially placed Deform pin discussed in Sections VIII.B.3.c-VIII.B.3.g, 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 until otherwise specified—i.e., "rendering in a loop the first set of pixels being shifted within the first image frame." AEM, 562, 219.

129. The screenshot in Section VIII.B.2.f illustrates how a "loopOut()" expression can be applied to the initial Deform pin placed in the exemplary composition therein. The series of screenshots provided in Section VIII.B.2.g shows 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.

8. Claim 9

**a. [9a]-[9e]: The computer system of claim 1, wherein the executable instructions include instructions that are executable to configure the computer system to:**

**receive a second starting point through the user interface, wherein the second starting point is received through a user selection of a second beginning portion of the first image frame;**

**receive a second ending point through the user interface, wherein the second ending point is received through a user selection of a second ending portion;**

**create a second digital link between the second starting point and the second ending point, wherein the second digital link comprises:**

**a second direction extending from the second starting point to the second ending point; and**

**a second length between the second starting point and the second ending point;**

**identify a second set of pixels that lie between the second starting point and the second ending point; and**

**shift the second set of pixels in the second direction.**

130. Claim 9 is dependent from claim 1. In my opinion, AEM discloses limitations [9a]-[9e]. 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 beginning portion of the first image frame" as recited in limitation [9a]. AEM, 219 (emphasis added); '641 Patent, 6:38-40. The user creates ending keyframes and motion paths for these additional Deform pins in the same way as for the first Deform pin—i.e., "receive a second ending point through the user interface, wherein the second ending point is received through a user selection of a second ending portion" as recited in limitation [9b], and "create a second digital link between the second starting point and the second ending point, wherein the second digital link comprises: a second direction extending from the second starting point to the second ending point; and a second length between the second starting point and the second ending point" as recited in limitation [9c]. AEM, 218-19; '641 Patent, 6:48-51; Sections VIII.B.3.d-VIII.B.3.e. Further, AEM teaches, and a POSITA would have been motivated to perform, creating a mask or matte to select any and all pixels of the layer to be nontransparent and thus included in the layer's mesh to be moved and animated, including pixels that lie along each Deform pin's motion path between each Deform pin's starting

position/keyframe and ending position/keyframe[15]—i.e., "identify a second set of pixels that lie between the second starting point and the second ending point" as recited in limitation [9d]. AEM, 218-19, 315, 194-95; Section VIII.B.3.f. Finally, 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—i.e., "shift the second set of pixels in the second direction" as recited in limitation [9e]. AEM, 218-19, 194-95, 562; Section VIII.B.3.g.

131. 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 additional Deform pins. Like the table in Attachment B as discussed in Section VIII.B.3.g, a "loopOut()" expression (*see* Section VIII.B.7) has also been added to each Deform pin to allow the animation to

---

[15] Okabe again provides further support that a POSITA would have been motivated to perform these steps by depicting, in its Figure 1, not 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 that lie along the strokes between each stroke's starting point and ending point. Okabe, 7, 2; *see also* Section VIII.C.6.a.

loop, but the first four screenshots in the table show the animation before the looping begins and are thus indicative of what the animation would look like when the "loopOut()" expression is not applied. Additional screenshots in Section VIII.B.2.h 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.

9. Claim 10: **The computer system of claim 9, wherein the first direction is different from the second direction.**

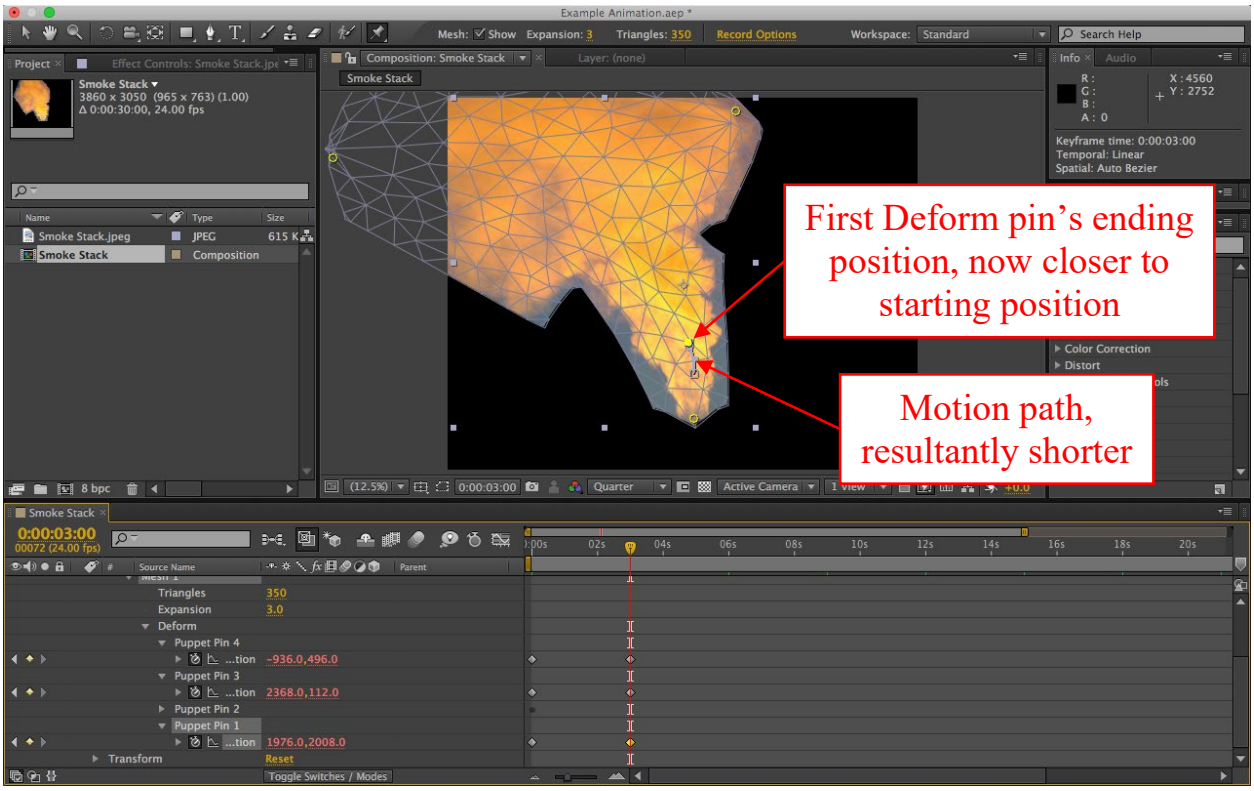
132. Claim 10 is dependent from claim 9. AEM teaches creating different motion paths for each Deform pin placed on a layer, thereby disclosing the claimed "wherein the first direction is different from the second direction." AEM, 218-19; Section VIII.B.8.a. Indeed, the example in Attachment C's table (discussed in Section VIII.B.8.a) shows each Deform pin having a separate motion path that differs from the others. AEM therefore, in my opinion, discloses claim 10.

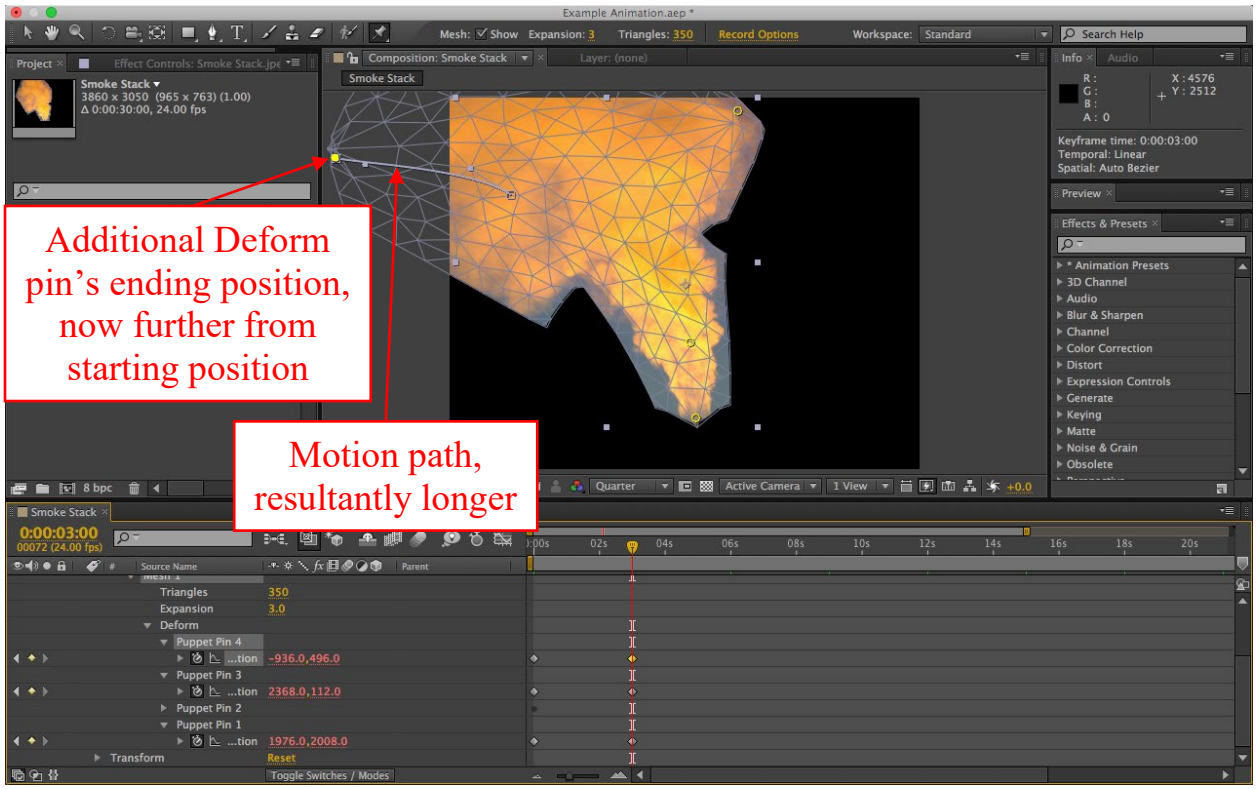
10. Claim 11: **The computer system of claim 9, wherein a magnitude of the shifting of the first set of pixels is proportionally related to the first length and the magnitude of the shifting of the second set of pixels is proportionally related to the second length.**

133. Claim 11 is dependent from claim 9. In my opinion, AEM discloses claim 11. 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.c-VIII.B.3.d. 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 to ending position/keyframe. 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 relatively long distance away from its starting position/keyframe, the corresponding change to the shape of the layer's mesh—and the movement of the layer's nontransparent pixels—will be large 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 a relatively short distance. *See* AEM, 218-19. Such illustrates a proportional relationship between the magnitude of pixel movement and the length of each Deform pin's motion path—i.e., "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 11.

134. To further illustrate, the exemplary composition from Attachment C's table (discussed in Section VIII.B.8.a) 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 a fixed 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.

11. <u>Independent Claim 12</u>

**a. [12pre]: A computer program product comprising one or more non-transitory computer storage media having stored thereon computer-executable instructions that, when transmitted to a remote computer system for execution at a processor, cause the remote computer system to perform a method for automating a shifting of pixels within an image file, the method comprising.**

135. AEM instructs a user to "[m]ake sure that you've installed the current version of [AECS6], including any available updates." AEM, 517. To view such

updates, AEM instructs to "go to the Downloads section of the Adobe website," and provides a hyperlink for doing so. AEM, 517. A user following these instructions would have downloaded and installed the most up-to-date version of AECS6 from Adobe's website—and thus a server (*see* Nakagawa, Abst.) (i.e., the claimed "computer program product comprising one or more non-transitory computer storage media having stored thereon computer-executable instructions")—onto the user's computer (i.e., "when transmitted to a remote computer system for execution at a processor, cause the remote computer system to perform").

136. Further, in my opinion, AEM describes using AECS6 to animate the movement of pixels within a single frame of a video (i.e., "method for automating a shifting of pixels within an image file"). Sections VIII.B.11.b-VIII.B.11.f.

**b. [12a]: 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 portion of a first image frame.**

137. *See* Sections VIII.B.3.b-VIII.B.3.c. Receiving the Deform pin's starting position from the user discloses the claimed "receiving a first indication of a first starting point through a user interface."

**c. [12b]: receiving, through the user interface, a first direction associated with the first starting point.**

138. *See* Sections VIII.B.3.d-VIII.B.3.e. The discussed motion path—which results when the user creates a starting and ending keyframe for the Deform pin via

the Composition or Layer panel, and which extends from the starting to ending position/keyframe—discloses the claimed "receiving, through the user interface, a first direction associated with the first starting point" as recited in limitation [12b]. AEM, 187, 218-19.

**d. [12c]: create a first digital link extending in the first direction from the first starting point.**

139. *See* Section VIII.B.3.e. The created motion path, as discussed, extends from the starting to ending position/keyframe and thus discloses the claimed "create a first digital link extending in the first direction from the first starting point" as recited in limitation [12c]. AEM, 187, 218-19.

**e. [12d]: selecting a first set of pixels that are along the first digital link and extend in the first direction away from the first starting point; and.**

140. For similar reasons discussed in Section VIII.B.3.f, AEM teaches, and a POSITA would have been motivated to perform, applying a mask or matte in AECS to select any or all pixels of a layer to be nontransparent and thus included in the layer's mesh to be moved and animated, including pixels that are along a Deform pin's motion path and extend in the motion path's direction away from the Deform pin's starting position/keyframe—i.e., "selecting a first set of pixels that are along the first digital link and extend in the first direction away from the first starting point" as recited in limitation [12d]. *See* Section VIII.B.11.d.

**f. [12e]: shifting the first set of pixels, in the first image frame, in the first direction.**

141. *See* Section VIII.B.3.g.

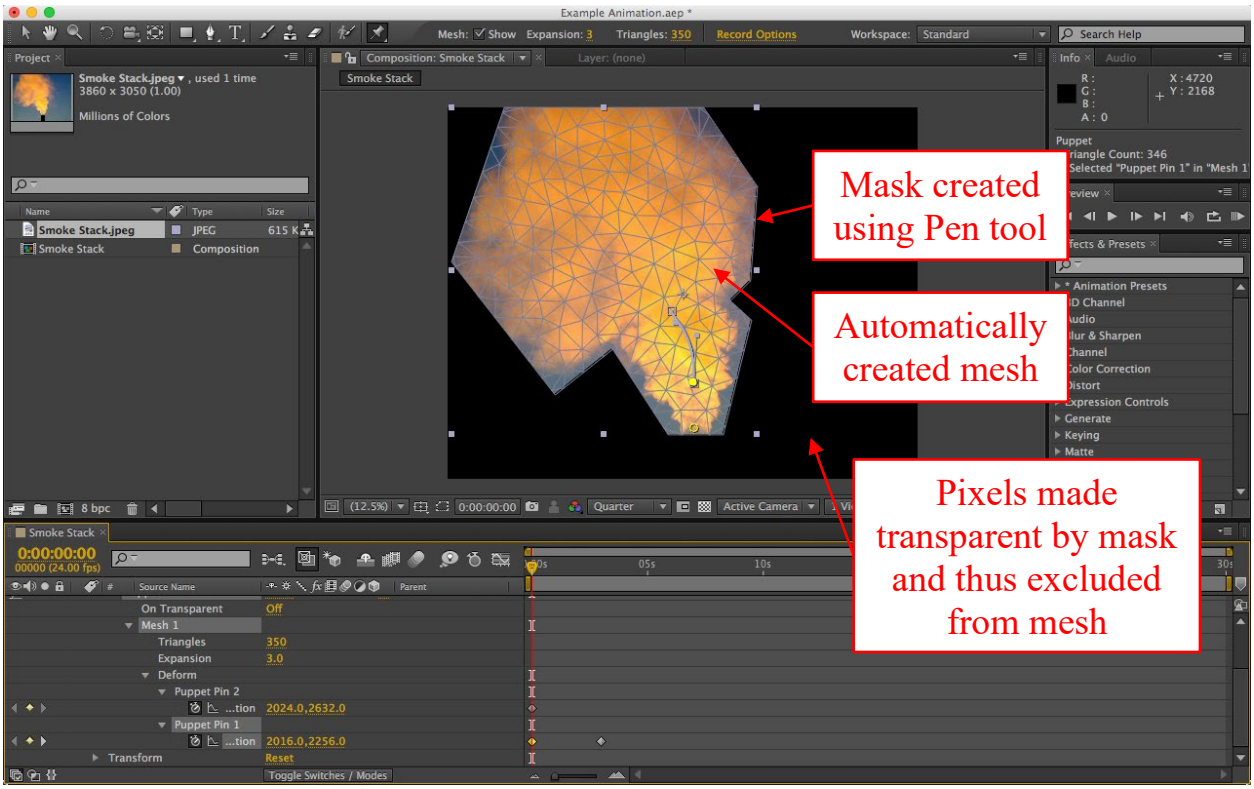
12. Claim 13

**a. [13a]-[13b]: The computer program product as recited in claim 12, further comprising receiving an indication to generate a first mask over a second portion of the first image frame, wherein pixels under the first mask are prevented from shifting.**

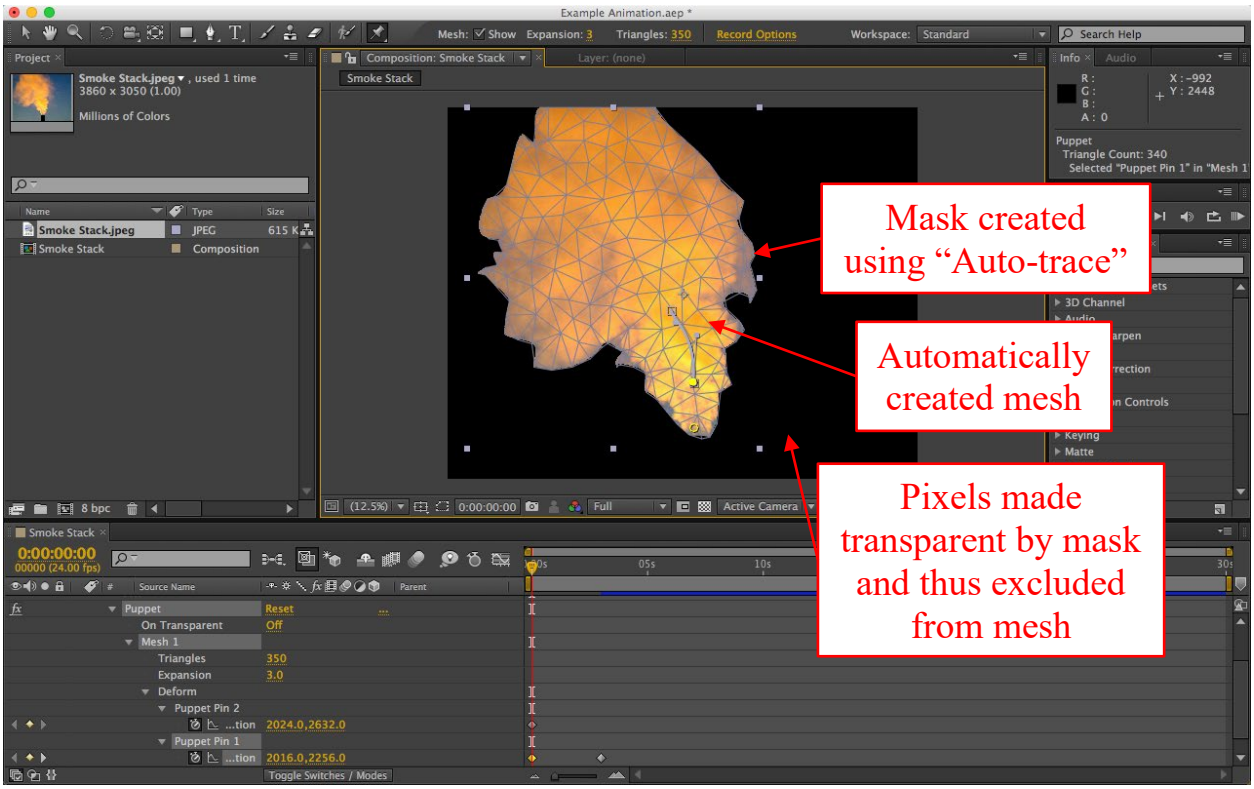
142. Claim 13 is dependent from claim 12. AEM teaches using a mask or matte to select pixels to be nontransparent and thus included in the layer's mesh to be moved and animated by the Puppet effect. AEM, 315, 218-19; Section VIII.B.3.f. Naturally, such a mask or matte is therefore also used to select 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., "a first mask over a second portion of the first image frame" as recited in limitation [13a], "wherein pixels under the first mask are prevented from shifting" as recited in limitation [13b]. 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.").

143. 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 "receiving an indication to generate a first mask" as recited in limitation [13a]. 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 "receiving an indication to generate a first mask." AEM, 328. Thus, in my opinion, AEM discloses limitations [13a]-[13b].

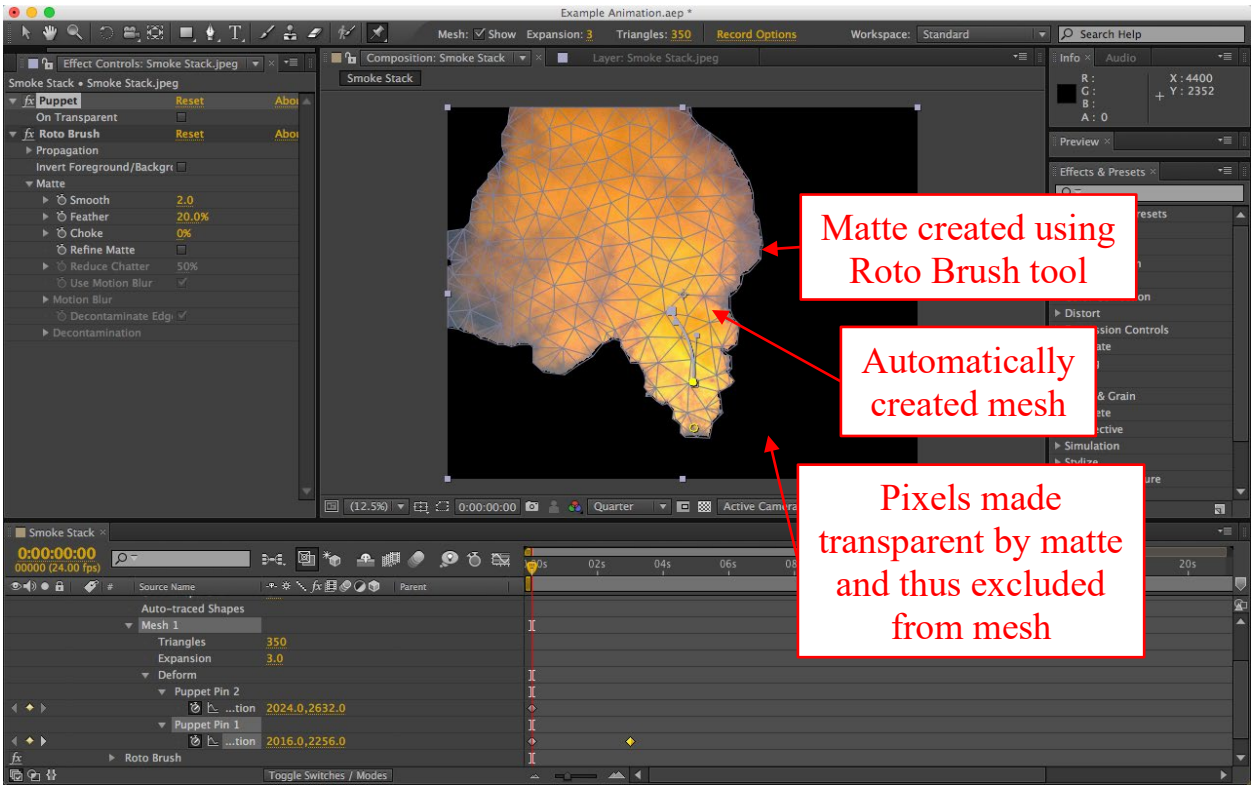
144. Section VIII.B.3.f's first screenshot depicting a mask created using the "Pen" tool is shown below, but the annotations now indicate pixels of the layer made transparent by the mask and thus excluded from the mesh:




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:




13. Claim 14: **The computer program product as recited in claim 13, further comprising computer-executable instructions that, when transmitted to the remote computer system for execution at the processor, cause the remote computer system to perform a method for automating the shifting of pixels within the image file, the method comprising receiving through a user interface a selection of the second portion of the first image frame around which the first mask should be generated.**

145. Claim 14 is dependent from claim 13. As discussed in Section VIII.B.12.a, 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 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, any of which disclose the claimed "receiving through a user interface a selection of the second portion of the first image frame around which the first mask should be generated." AEM, 318, 262, 264-65, 328. Therefore, in my opinion, AEM discloses claim 14.

14. Claim 15

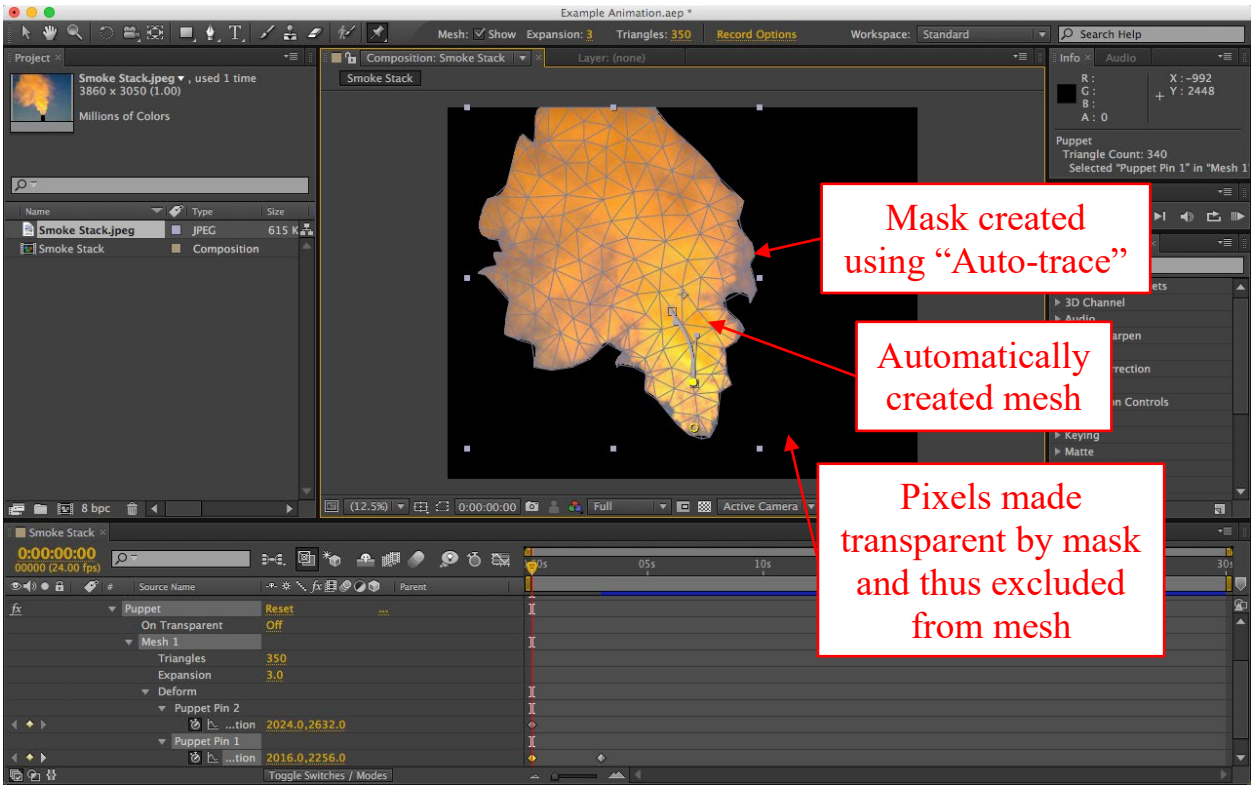
**a. [15a]-[15b]: The computer program product of claim 14, further comprising computer-executable instructions that, when transmitted to the remote computer system for execution at the processor, cause the remote computer system to perform a method for automating the shifting of pixels within the image file, the method comprising:**

**identifying one or more edges that form a first boundary around the second portion; and**

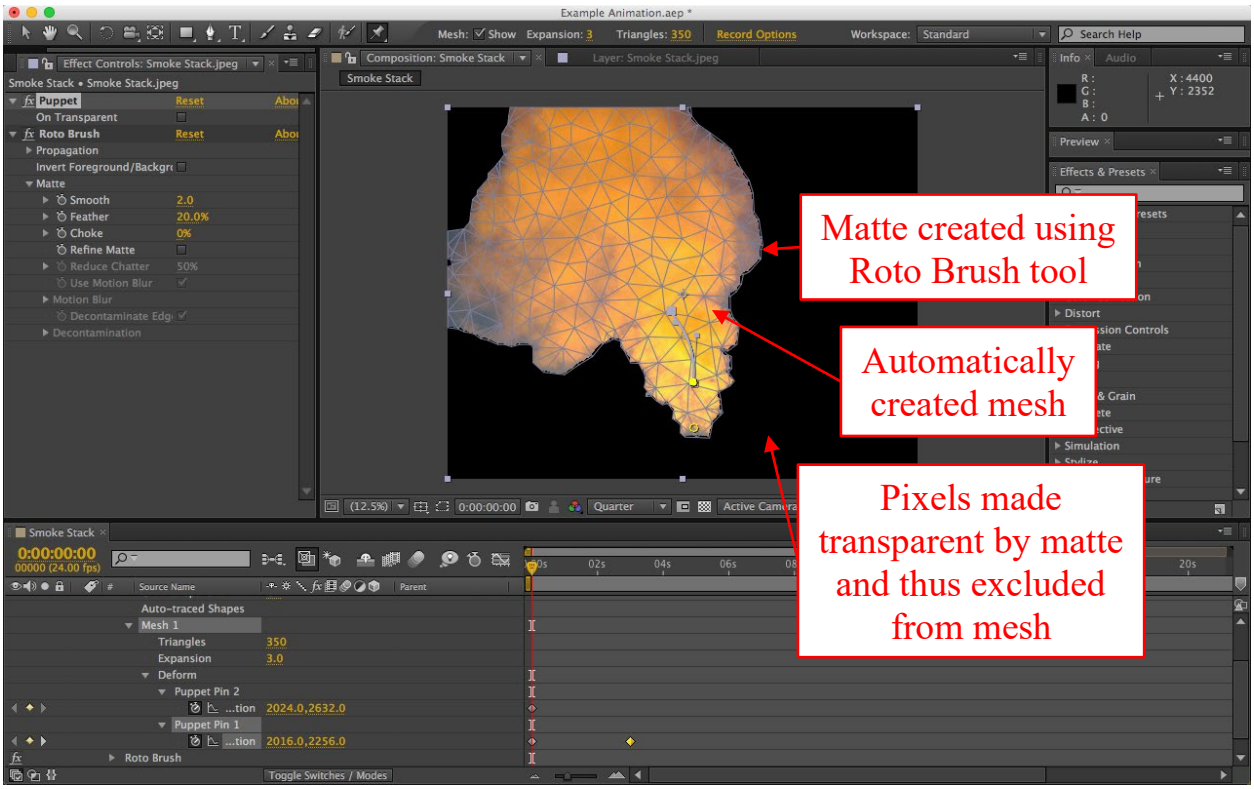
**generating the first mask to cover area within the first boundary.**

146. Claim 15 is dependent from claim 14. AEM teaches that a mask is automatically created for a layer when using AECS6's "Auto-trace" function. 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 first boundary around the second portion" recited in limitation [15a]—and generates a mask that outlines such edges—i.e., "generating the first mask to cover area within the first boundary" as recited in

limitation [15b]. 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.12.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:




147. Further, AEM teaches creating a matte using the "Roto Brush" tool, which, similar to Auto-trace (*see* AEM, 262), uses "Edge Detection" to detect a "segmentation boundary" separating a layer's foreground and background elements based on user-drawn "strokes" over "representative areas of the foreground and background elements." AEM, 328-31. This additionally discloses the claimed "identifying one or more edges that form a first boundary around the second portion." *See* '641 Patent, 4:46-64. 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 first mask to cover area within the first boundary." AEM, 328, 318. Indeed, Section VIII.B.12.a's third screenshot depicts creating a matte using the "Roto Brush" tool:




As shown, the segmentation boundary between the foreground (the billowing smoke) and background (the sky, made transparent by the matte) traces the edge between the two. Therefore, in my opinion, AEM discloses limitations [15a]-[15b].

15. <u>Independent Claim 19</u>

**a. [19pre]: A method for transmitting to a client computing device instructions for shifting pixels within a video file, comprising:**

**transmitting computer executable instructions to a client computing device, the computer executable instructions configured to cause the client computing device to.**

148. AEM instructs a user to "[m]ake sure that you've installed the current version of [AECS6], including any available updates." AEM, 517. To view such

updates, AEM instructs to "go to the Downloads section of the Adobe website," and provides a hyperlink for doing so. AEM, 517. A user following these instructions would have downloaded and installed the most up-to-date version of AECS6 onto the user's computer (i.e., the claimed "client computing device") from Adobe's website—and thus a server (*see* Nakagawa, Abst.) (i.e., "transmitting to a client computing device instructions," and "transmitting computer executable instructions to a client computing device, the computer executable instructions configured to cause the client computing device to").

149. Further, in my opinion, AEM teaches using AECS6 to animate the movement of pixels within a single frame of a video (i.e., "for shifting pixels within a video file"). Sections VIII.B.15.b-VIII.B.15.g.

**b. [19a]: access, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within a frame of the digital image file.**

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

**c. [19b]: 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 a first image frame.**

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

**d. [19c]: 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.**

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

**e. [19d]: create a first digital link between the first starting point and the first ending point, wherein the first digital 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.**

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

**f. [19e]: identify a first set of pixels that lie along the first digital link between the first starting point and the first ending point; and.**

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

**g. [19f]: shift the first set of pixels in the first direction.**

155. *See* Section VIII.B.3.g.

16. Claim 20: **The method of claim 19, wherein the digital image file comprises a video file and the first image frame comprises a frame of the video file.**

156. *See* Section VIII.B.5.

## C. Ground 2: IMU and Okabe, and Claims 1-4, 8-14, and 19-20

### 1. Summary of IMU

157. IMU contains Wayback Machine captures from 2012 of the ImageMagick.org 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.

158. IMU "strongly recommend[s] to use an up-to-date version of IM[V6]" (IMU-Windows, 3; *see also* IMU-Home, 6 ("Also all examples are re-built using the latest beta release for IM…. If you get something different your IM[V6] is probably a much older version, with old bugs, or your IM[V6] is incorrectly installed.")), and hyperlinks the ImageMagick.org website's "Downloads Page" for users to download and install such an new version of IMV6 onto their computers (IMU-Home, 1). This hyperlink, when clicked, directs to the archived "Download ImageMagick" webpage shown in EX1026, from which the user could indeed download and install the "latest release of I[MV6]." EX1026; IMU-Home, 1. In fact, I generated EX1026 by clicking this hyperlink in IMU-Home, which directed me to the archived webpage shown in EX1026, and then printing out that webpage.[16] *Compare* EX1026, *with* IMU-Home, 1. EX1026's capture date of March 28, 2012 is only one day after IMU-Home's capture date of March 27, 2012, and thus EX1026 shows the webpage that a user would have been directed to upon clicking IMU-Home's "Downloads Page" hyperlink around the time of these capture dates. *Compare* EX1026, *with* IMU-Home, 1, *and* Archive, p. 4. *See* Archive, ¶5.

[16] I provided counsel with this printout, which I understand is being used as an exhibit in this proceeding.

159. IMU also teaches that, once installed, IMV6 is operated in the command-line. IMU-Home, 2. Thus, IMV6 "*is not a GUI image editor*." IMU-Home, 2.

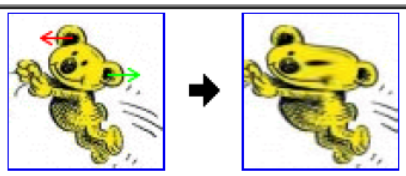
160. Further, IMU teaches how to manually write text commands in IMV6 to apply effects to an image, such as IMV6's "Shepard's Distortion" pixel shifting effect, as well as how to create infinitely looping animations from such effects. IMU also teaches how to extract individual frames from a GIF animation, which can then be used to create a subsequent animation. IMU additionally teaches how to manually write commands to modify the transparency of the pixels of an image (e.g., an extracted frame of a GIF animation)—and thus the visibility of effects that are applied to the image—using a "matte." Each is discussed herein.

**a. Applying Effects and Creating an Animation**

161. 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 that allows a user to place "control points" on pixels located at user-specified coordinates of an image and move such control points to new coordinates "to distort the image in terms of 'local' effects." IMU-Distorting, 59, 19-20. As an example, IMU provides an image of a koala, as well as an exemplary command that places and moves Shepard's Distortion control points 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 command, as well as the 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 within the image:

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

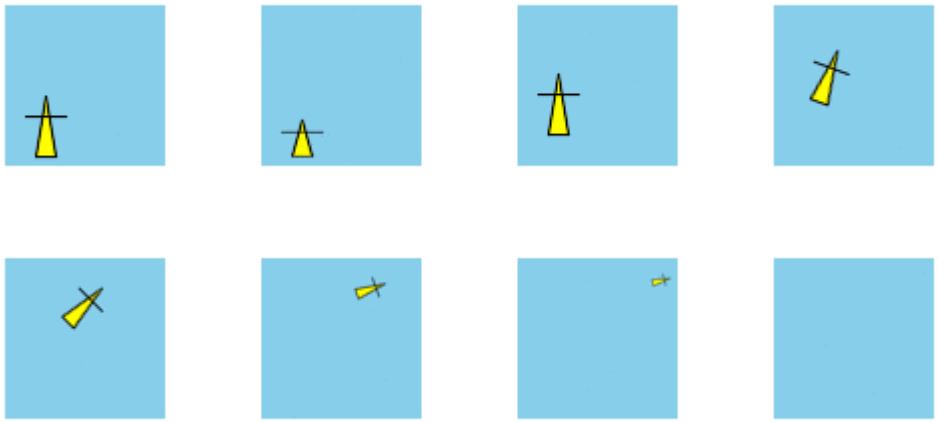


IMU-Distorting, 59.

162. 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, result in an animation. *See* IMU-Distorting, 17-18. IMU provides an example that applies 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 moving a placed SRT Distortion control point in seven increments to create frames that, when played sequentially, animate the space ship launching along a path through the sky. IMU-Distorting, 17-18. This example, as well as 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.[17]

[17] 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

163. After teaching 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. In creating such a GIF, IMU teaches applying a "-loop" operator to specify the "[n]umber of times the GIF animation is to cycle though the image sequence before stopping." IMU-Animating, 2. When "-loop" is applied at its default value of zero, such as in the above example, the GIF will render the animation in an "infinite loop." IMU-Animating, 2; IMU-Distorting, 17-18.

**b. Extracting a Single Frame of a GIF Animation**

164. In addition to animation, IMU also teaches how to use IMV6 to "split[] an animation into frames"—i.e., extract and output an animation's frames as individual images. IM-Animating, 7. Particularly, IMV6's "+adjoin" operator can be used to "read in" a "GIF animation[] and output the individual frame images in the animation sequence." IM-Animating, 7.

165. The "+adjoin" operator, however, extracts "sub-frame[s]"—certain animated GIFs are created by layering sub-frames over one another over time (*see*

---

GIF file, accessible at the preceding URL) in a similar manner as that taught in IM-Animation, 8-9, and discussed *infra* in Section VIII.C.1.b.

IMU-Animating, 16-17), and, for such GIFs, the "+adjoin" operator outputs only those sub-frames (IM-Animating, 7-8). To see instead "a complete view of the animation at each point," IMU teaches to use IMV6's "-coalesce" operator, which works similarly to "+adjoin" but outputs each full frame image rather than sub-frames. IM-Animating, 8-9.

166. IMU provides a first example of applying "+adjoin" to an animated GIF of a script letter K being drawn to illustrate the operator's functionality:

```
convert script_k.gif  +repage  +adjoin  script_k_%02d.gif
```

IMU-Animating, 8. IMU also provides a second example of applying "-coalesce" to the same animated GIF to show the differences in results:

```
montage script_k.gif -coalesce \
        -tile x1 -frame 4 -geometry '+2+2' \
        -background none -bordercolor none coalesce_k_montage.gif
```

IMU-Animating, 8.

167. Further, IMU also teaches numerous ways to use such frame images, including to "study, edit, modify and re-optimize" the original animation. IMU-Animating, 8. The user can also "use the individual frames for other projects," e.g., for creating a subsequent animation. IMU-Animating, 7. For instance, IMU

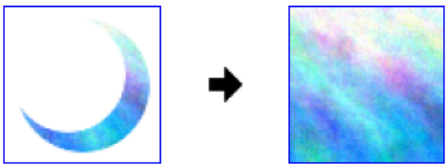
provides an example of applying "+adjoin" to extract the sub-frames of a "canvas_prev.gif" animated GIF, and teaches that these extracted sub-frames may be used for "easil[y] rebuild[ing] the animation" or may be additionally modified. IMU-Animating, 7.

#### c. Modifying Transparency of Certain Pixels Using a Matte

168. IMU also teaches modifying the transparency of the pixels of an image (e.g., a frame image)—and thus the visibility of effects that are applied to the frame image's pixels—by modifying the image's "transparency (alpha) channel," also referred to as a "'matte' channel" or simply a "matte." IMU-Masking, 2-3; IMU-Animating, 7-8. 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.

169. 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 correspondingly change the transparency of an image's pixels:



IMU-Masking, 3-4.

2. Summary of Okabe

170. Okabe describes a "method for synthesizing fluid animation from a single image." Okabe, 1. While Okabe acknowledges 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.

171. 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 target image, "which is a simple task and takes less than 1 min[ute]." Okabe, 3, 7. Okabe's Figure 1 provides an example of these steps, where the discussed "strokes" are "shown as orange arrows":

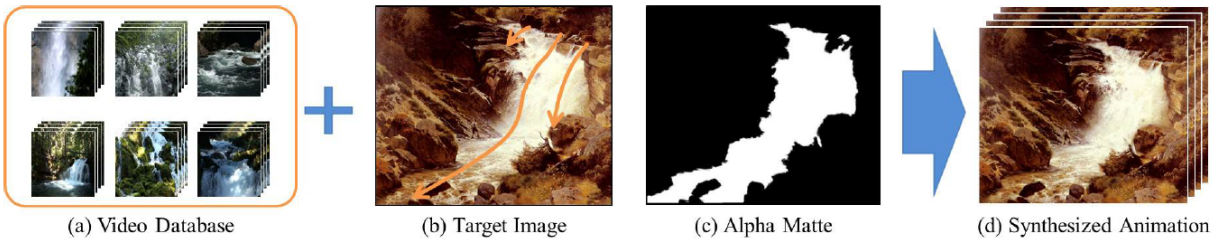



**Figure 1:** *We employ a database of video examples of fluids (a). The user specifies a target image (b) with a few optional suggestions about fluid motion, e.g., sketches of flow direction, shown as orange arrows. The user also provides an alpha matte of the region of interest (c). The system synthesizes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

Okabe, 2. From these inputs, Okabe's algorithm is then able to generate a flow animation in the target image corresponding to the flow direction and speed specified

by the strokes, and permits "infinite repetition" of such an animation. Okabe, 1, 8. According to Okabe, such simple and graphical commands allow a user to animate images "with less effort than with previous methods." Okabe, 1.

3. The IMU-Okabe Combination

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

172. According to IMU, IMV6 is operated in the command-line and therefore "*is 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.

173. In IMV6 in particular, to create an animation from an image (e.g., a frame image extracted from a GIF animation using "+adjoin" or "-coalesce" in IMV6, *see* IMU-Animating, 7-8), 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-Animating, 7-8; IMU-Distorting, 17-18. For example, to animate a frame image using Shepard's Distortion, a POSITA would have recognized to 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 direction and speed. *See* IMU-Distorting, 17-18 (creating an animation in the same way, but using IMV6's "SRT distortion" effect instead, to animate the launch of a space ship along a non-linear path), 59, 19-20.

174. Further, to make distortion and animation visible only in a particular region of the image, the user would write further commands applying 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 "-loop." IMU-Animating, 1-2; *see also, e.g.*, IMU-Distorting, 17-18.

175. But given this laborious, non-graphical process for extracting a frame and creating a subsequent animation of the frame image in IMV6, a POSITA would have been motivated to modify IMV6 (as described in IMU) by at least enabling a user to animate an already-extracted frame 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*," at least with respect to the aspect of animating images. *See* IMU-Home, 2.

176. 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 using effects applied to a frame image (e.g., using the Puppet effect) with simple and graphical user commands for defining such animation (e.g., by placing and moving Deform pins to define keyframes and motion paths, 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.

177. Okabe teaches simple and graphical user commands that a POSITA would have found well-suited for implementing a GUI for at least animating an image (e.g., a frame image) 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" 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 these inputs, an infinitely repeating animation is automatically produced that follows the user-specified direction and speed. Okabe, 1-2, 8. Okabe's simple and graphical user commands thus enable creating looping animations "with less effort than with previous methods" and "markedly reduces the user burden." Okabe, 1-2.

178. Given such advantages, a POSITA would have found Okabe's simple and graphical user commands desirable for animating an image (e.g., a frame image) in IMV6, i.e., obviating at least 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.

179. Further, a POSITA would have been motivated to make such a modified version of IMV6 available for users to download on the ImageMagick.org website via the "Download Page" hyperlinked in IMU-Home. *See* IMU-Home, 1. As IMU teaches, "I[MV6] is under constant development, new versions are released roughly on a monthly basis. It is strongly recommended to use an up-to-date version of IM[V6], especially when IM[V6] doesn't seem to perform a job quite as you expect it to do." IMU-Windows, 3; IMU-Home, 6. And IMU provides the above hyperlink for users to download and install such new versions of IMV6. IMU-

Home, 1. Indeed, as a POSITA would have known, clicking the above hyperlink would have directed the user to the "Download ImageMagick" webpage of the ImageMagick.org website shown in EX1026, which provides a download link for the "latest release of I[MV6]." EX1026; IMU-Home, 1. Given that modifying IMV6 in view of Okabe as discussed above would have likewise created a new version of IMV6, a POSITA would have thus found the hyperlinked "Download Page" ideal for hosting and providing such a new version of IMV6 to users. *See* IMU-Home, 1.

**b. Resulting Combination of IMU with Okabe**

180. So modified, the IMU-Okabe Combination allows a frame image from a GIF animation to be extracted and outputted using, e.g., IMV6's "+adjoin" or "-coalesce" operator. *See* IMU-Animating, 7-8. The IMU-Okabe Combination then allows a subsequent animation of the image to be created using Okabe's simple and graphical user commands, where, rather than requiring complex commands to be written that place and move control points (e.g., for Shepard's Distortion) on the image frame-by-frame and in the user's desired direction and speed across the animation (*see* IMU-Distorting, 17-18, 59, 19-20), the IMU-Okabe Combination enables the same result to be achieved via user-drawn strokes on the image that specify the direction and speed of corresponding Shepard's Distortion control points across the animation (*see* Okabe, 7, 2-3).

181. Also, similar to how IMV6 allows a user to apply a matte to make distortion and animation effects visible in only a portion of the image (IMU-Masking, 2), the IMU-Okabe Combination enables a user to apply 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 saving an animation as an infinitely looping animated GIF using "-loop" (IMU-Animating, 1-2), the IMU-Okabe Combination permits infinite repetition of the resulting animation (*see* Okabe, 1, 8).

182. Lastly, the IMU-Okabe Combination, being a new version of IMV6, would have been made downloadable for installation onto the user's computer from the ImageMagick.org website's "Downloads Page" hyperlinked in IMU-Home. *See* IMU-Home, 1.

4. Independent Claim 1

**a. [1pre]: A computer system providing, to a client computing device, software for automating a shifting of pixels within a video file, the computer system comprising:**

**one or more processors; and**

**one or more computer-readable media having stored thereon executable instructions that are transmitted to the client computing device for execution by one or more client processors on the client computing device, the executable instructions comprising instructions that when executed by the one or more client processors configure the client computing device to perform at least the following.**

183. IMU teaches downloading and installing the most "up-to-date version" of IMV6 onto a user's computer from the ImageMagick.org website. IMU-Windows, 3; IMU-Home, 1, 6. 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 a computer, the IMU-Okabe Combination would thus also be installed to run on a user's computer, which meets the claimed "computer system" and remaining claim language according to Plotagraph's infringement contentions in *Plotagraph*, Dkt. No. 42. *See* EX1022, 134.

184. Alternatively, IMU "strongly recommend[s] to use an up-to-date version of IM[V6]" (IMU-Windows, 3; IMU-Home, 6), and hyperlinks the ImageMagick.org website's "Downloads Page" for users to download and install such a new version of IMV6 onto their computers (IMU-Home, 1). Following IMU's recommendations, a user would have thus downloaded and installed the IMU-Okabe Combination (i.e., the claimed "software") onto the user's computer (i.e., "client computing device" with "one or more client processors") from this webpage, which would have been provided by a server (*see* Nakagawa, Abst.) (i.e., "computer system providing, to a client computing device, software" and comprising "one or more processors" and "one or more computer-readable media having stored thereon executable instructions that are transmitted to the client computing device for execution by one or more client processors"). *See* Section VIII.C.3.b.

185. Further, the IMU-Okabe Combination enables a user to automatically animate an extracted frame of a GIF animation (i.e., "automating a shifting of pixels within a video file"). Sections VIII.C.4.b-VIII.C.4.g.

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

186. According to IMU, when using the "+adjoin" or "-coalesce" operators, IMV6 and thus the IMU-Okabe Combination "read[s] in" a "GIF animation" from the current directory being operated on—i.e., the claimed "access, from memory, a

digital image file, wherein the digital image file comprises information that corresponds to individual pixels within a frame of the digital image file." IMU-Animating, 7-8; IMU-Home, 4; '641 Patent, claim 3; Section VIII.C.3.b; *see also* AEM, 100 (listing "Animated GIF (GIF)" as a "[v]ideo and animation format[]"); '641 Patent, 10:13-24 (stating "animated GIF" is one "format[]" for viewing the video file after pixel shifting is applied); IMU-Animating, 15-19 (explaining GIF animations are "displayed one ***frame*** to another," where each frame includes "***pixels***" that may be "cleared or erased from one frame to the next" (emphasis added)).

**c. [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 a first image frame.**

187. IMU teaches that "+adjoin" and "-coalesce," as used by the IMU-Okabe Combination, "split[] an animation into frames" to allow a user to "use the individual frames for other projects," such as for creating subsequent animations.[18]

[18] Should there be any argument that IMU teaches applying Shepard's Distortion to animate only an image and not specifically the extracted frame image, a POSITA would have nevertheless been motivated to use such an extracted frame as the image to be animated because such a frame would have been easily accessible and, in

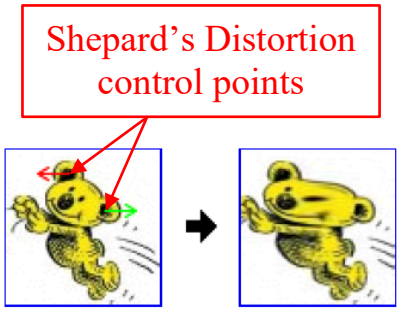
IMU-Animating, 7-8; Section VIII.C.3.b; *see also, e.g.*, IMU-Animating, 7. Any of such extracted frame images meets the claimed "first image frame."

188. Additionally, IMU teaches that IMV6 is an "image-to-image converter" that enables a user apply effects such as distortion effects (e.g., "Shepard's Distortion") to convert one image (e.g., a frame image) into a sequence of frames that animate the image when played sequentially. IMU-Home, 2; IMU-Distorting, 17-18; IMU-Animating, 7-8. To do so, the user places a "control point" on a pixel 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

---

common instances, particularly desirable for modifying and animating with IMV6's effects, e.g., Shepard's Distortion. *See, e.g.*, '641 Patent, 1:36-38; AEM, 590; IMU-Animating, 7-8 (teaching that extracted frames are useful not only "for other projects" but also to "study, edit, modify and re-optimize" the original GIF animation).

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:

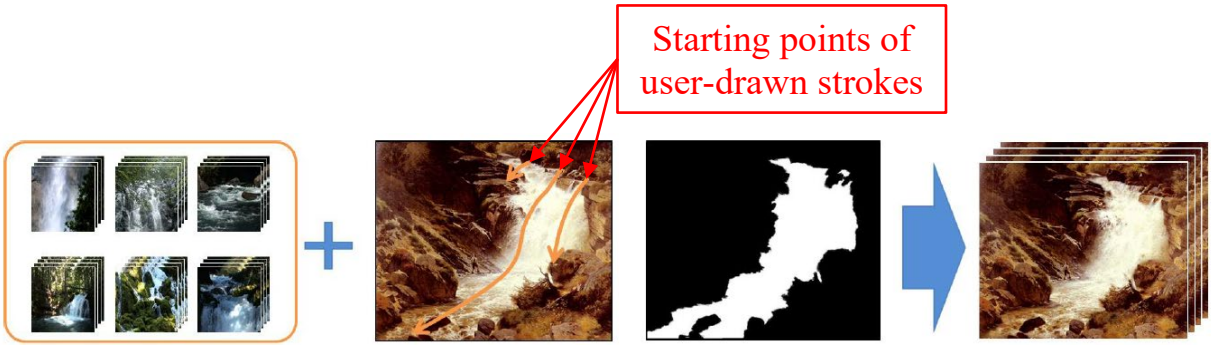



IMU-Distorting, 59 (additional annotations added).

189. Further, in the IMU-Okabe Combination, IMU's placement and movement of a Shepard's Distortion control point corresponds to Okabe's user-drawn "stroke[]" on the frame image. *See* Okabe, 7, 3; Section VIII.C.3.b. That is, Okabe's user-drawn stroke as used in the IM-Okabe Combination includes an indication of a starting point, direction, and speed across the animation, like how a user in IMV6 indicates a control point's starting coordinate, 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. Receiving the starting point of a user-drawn stroke on the frame image thus meets the claimed "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 a first image frame." *See* IMU-Distorting, 59, 19-20; Okabe, 7, 3.

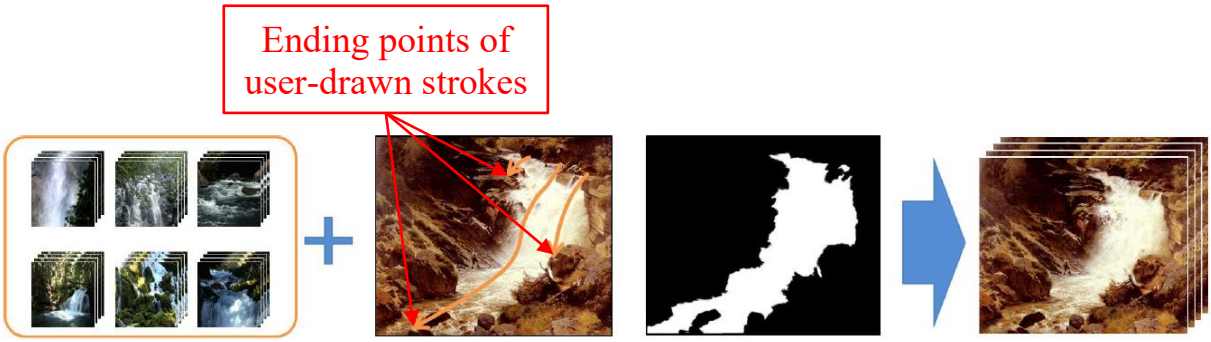
190. Okabe's Figure 1 illustrates the relevant disclosure by depicting several user-drawn strokes (shown in orange) on an image of a waterfall:




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

**d. [1c]: 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.**

191. In my opinion, the IMU-Okabe Combination meets limitation [1c]. In the IMU-Okabe Combination, a user defines a Shepard's Distortion control point's direction and speed across the animation of a frame image by drawing a "stroke[]" on the image. *See* Okabe, 7, 2-3; IMU-Distorting, 59; Section VIII.C.3.b. Receiving this user-drawn stroke on the frame image, specifically the stroke's ending point, meets 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." *See* Okabe, 2 (Figure 1, reproduced below with annotations added).




**e. [1d]: create a first digital link between the first starting point and the first ending point, wherein the first digital link comprises:**

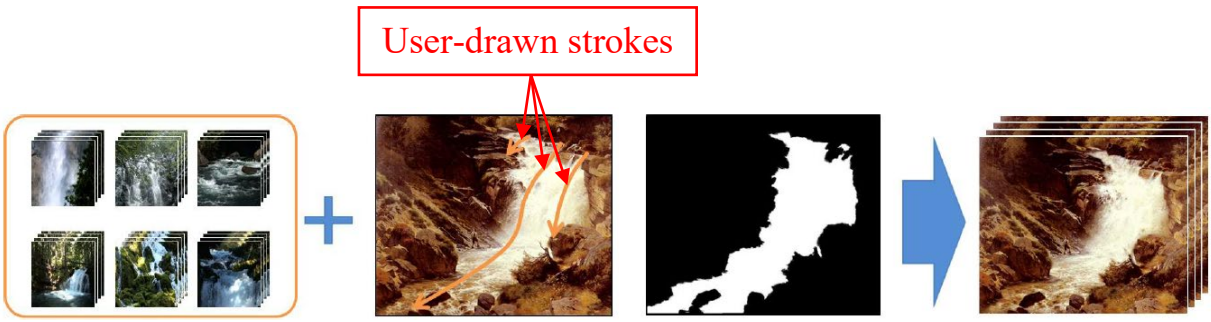
**a first direction extending from the first starting point to the first ending point; and**

**a first length between the first starting point and the first ending point.**

192. IMU teaches animating an image (e.g., a frame image) in IMV6 using distortion effects (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; IMU-Animating, 7-8.

193. 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 "stroke[]" from Okabe on the frame image—i.e., the claimed "create a first digital link between the first starting point and the first ending point, wherein the first digital 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-drawn strokes, each of which comprise a direction (indicated by the arrowhead) and a length:




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

(a) Target Image

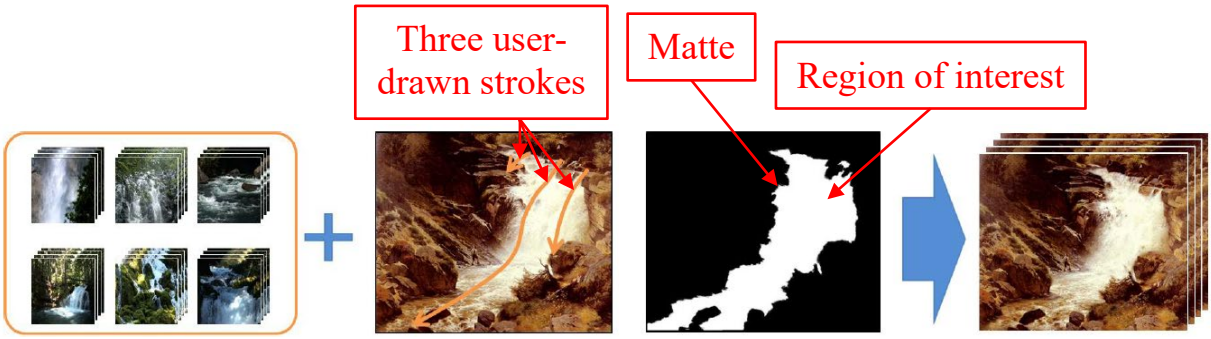
Okabe, 6 (discussing Figure 8-a). Therefore, in my opinion, the IMU-Okabe Combination meets limitation [1d].

**f. [1e]: identify a first set of pixels that lie along the first digital link between the first starting point and the first ending point; and.**

194. IMU teaches modifying the transparency of pixels of an image (e.g., an extracted frame) in IMV6—and thus the visibility of effects applied to the frame

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; IMU-Animating, 7-8.

195. The IMU-Okabe Combination similarly allows a user to apply a matte specifying a region of interest the user desires to animate in the image (*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, a user would have been allowed to select any or all pixels of the image to be included in this region of interest, including pixels that lie along the stroke between the stroke's starting point and ending point—i.e., the claimed "identify a first set of pixels that lie along the first digital link between the first starting point and 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 to be animated (in white) that includes pixels that lie along three user-drawn strokes between each stroke's starting point and ending point:



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

**g. [1f]: shift the first set of pixels in the first direction.**

196. With respect to "shift the first set of pixels," the '641 Patent states the claimed "shift[ing]" can be performed using "a warping function, such as Shepard's distortion." '641 Patent, 9:27-29. 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 a frame image in the user's specified direction and speed. *See* IMU-Distorting, 17-18, 59, 19-20; IMU-Animating, 7-8.

197. Further, in the IMU-Okabe Combination allows, the user specifies a Shepard's Distortion 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 also allows a user to apply a matte to select pixels that lie along the stroke between the stroke's starting point and ending point—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. The IMU-Okabe Combination then automatically generates animation frames by shifting the selected pixels in the

stroke's direction using IMV6's Shepard's Distortion—i.e., "shift the first set of pixels in the first direction." *See* Okabe, 1-2; Section VIII.C.3.b.

5. Claim 2: **The computer system of claim 1, wherein the first ending portion comprises a particular portion of the first image frame.**

198. *See* Sections VIII.C.4.c-VIII.C.4.d. The stroke is drawn on a single frame image, and thus the stroke's ending point is on the same frame image as the stroke's starting point—i.e., "wherein the first ending portion comprises a particular portion of the first image frame."

6. Claim 3: **The computer system of claim 1, wherein the digital image file comprises a video file and the first image frame comprises a first video frame of the video file.**

199. *See* Section VIII.C.4.b. The animated GIF discussed is a "video file" as claimed, and the extracted frame image is "a first video frame of the video file."

7. Claim 4: **The computer system of claim 3, wherein the first ending portion comprises a particular portion of a second video frame within the video file.**

200. *See* Sections VIII.C.4.c-VIII.C.4.d, VIII.C.6. While the stroke is drawn on a single frame image, and thus the stroke's ending point is on the same frame image as the stroke's starting point, such nevertheless meets claim 4 because the claimed "first video frame" and "second video frame" may be the same frame. '641 Patent, 6:51-54; EX1022, 137.

8. Claim 8: **The computer system of claim 1, wherein shifting the first set of pixels comprises rendering in a loop the first set of pixels being shifted within the first image frame.**

201. IMU teaches saving an animation of an image (e.g., a frame image) in IMV6 as an infinitely looping animated GIF using IMV6's "-loop" operator. IMU-Animating, 1-2, 7-8. 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 of the frame image by, e.g., allowing a user to apply "-loop" to an animation created by Shepard's Distortion and the user's stroke—i.e., "rendering in a loop the first set of pixels being shifted within the first video frame." See IMU-Animating, 1-2; Okabe, 1, 8; Section VIII.C.3.b. Therefore, in my opinion, the IMU-Okabe Combination meets claim 8.

9. Claim 9

a. **[9a]-[9e]: The computer system of claim 1, wherein the executable instructions include instructions that are executable to configure the computer system to:**

**receive a second starting point through the user interface, wherein the second starting point is received through a user selection of a second beginning portion of the first image frame;**

**receive a second ending point through the user interface, wherein the second ending point is received through a user selection of a second ending portion;**

**create a second digital link between the second starting point and the second ending point, wherein the second digital link comprises:**

**a second direction extending from the second starting point to the second ending point; and**

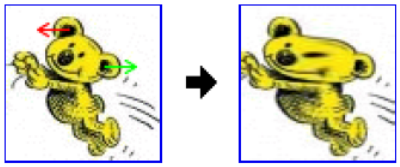
**a second length between the second starting point and the second ending point;**

**identify a second set of pixels that lie between the second starting point and the second ending point; and**

**shift the second set of pixels in the second direction.**

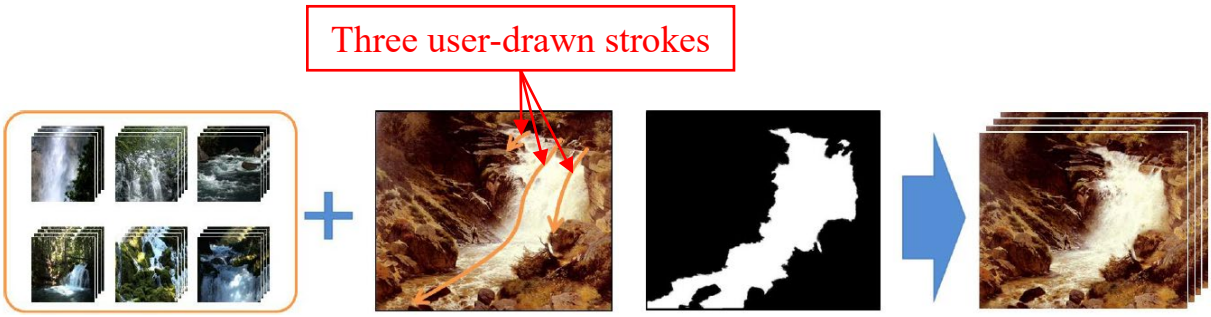
202. In my opinion, the IMU-Okabe Combination meets limitations [9a]-[9e]. IMU teaches applying multiple Shepard's Distortion control points at different coordinates of an image (e.g., a frame image) in IMV6 and incrementally moving the control points to animate the image. *See* IMU-Distorting, 17-18, 59, 19-20;

IMU-Animating, 7-8. IMU provides an example of one such increment using two Shepard's Distortion control points on an image of a koala:



IMU-Distorting, 59.

203. 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 frame image, each specifying a direction and speed of a different control point. *See* Okabe, 7, 2-3; Section VIII.C.3.b. Indeed, Okabe's Figure 1 depicts three user-drawn strokes on an image:

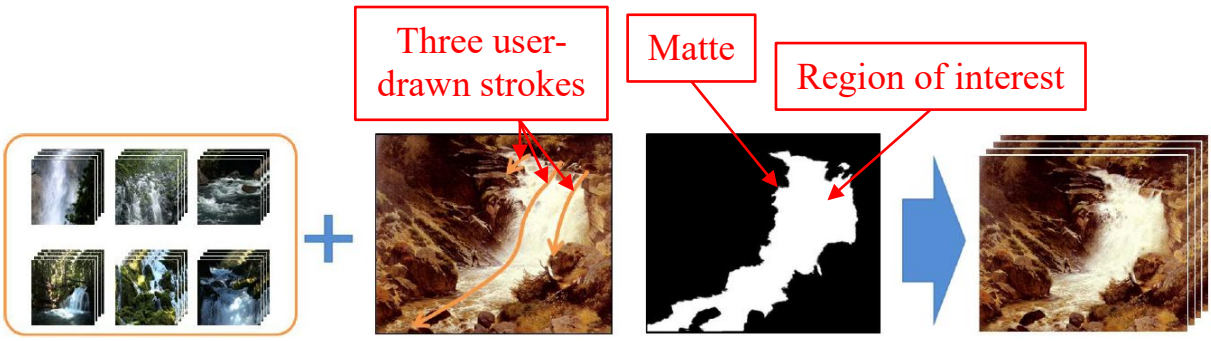



Okabe, 2 (Figure 1, annotations added). Similarly, Okabe's Figure 8-a depicts two user-drawn strokes shown as two green lines:

(a) Target Image

Okabe, 6 (discussing Figure 8-a). Receiving multiple user-drawn strokes—including each stroke's starting and ending points, direction, and speed—meets "receive a second starting point…" as recited in limitation [9a], "receive a second ending point…" as recited in limitation [9b], and "create a second digital link…" as recited in limitation [9c]. *See* Sections VIII.C.4.c-VIII.C.4.e.

204. The IMU-Okabe Combination also allows a user to apply a matte specifying a region of interest the user desires to animate, including pixels that lie along each stroke between each stroke's starting point and ending point—i.e., the claimed identify a second set of pixels that lie between the second starting point and the second ending point" as recited in recited in limitation [9d]. *See* Okabe, 1; IMU-Masking, 2; Section VIII.C.3.b; *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 that lie along three strokes between each stroke's starting point and ending point:

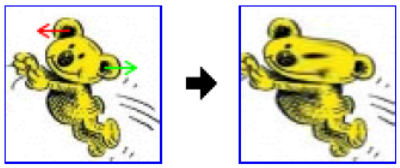



Okabe, 2 (Figure 1, annotations added).

205. After receiving the strokes and matte, the IMU-Okabe Combination automatically generates animation frames using Shepard's Distortion according to the user's strokes—i.e., "shift the second set of pixels in the second direction" as recited in limitation [9e]. *See* Okabe, 1-2; IMU-Animating, 1-2; Section VIII.C.3.b; *see also* Section VIII.C.4.g.

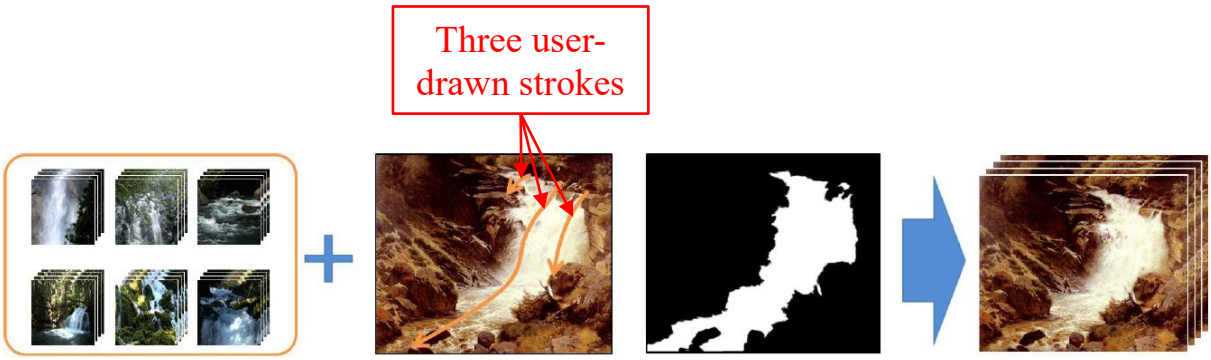
10. Claim 10: **The computer system of claim 9, wherein the first direction is different from the second direction.**

206. IMU teaches applying and incrementally moving control points, such as for Shepard's Distortion, to different coordinates on an image (e.g., a frame image) to generate an animation. *See* IMU-Distorting, 17-18, 59, 19-20; IMU-Animating, 7-8. 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.

207. 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 frame 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 10.

11. Claim 11: **The computer system of claim 9, wherein a 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.**

208. In my opinion, the IMU-Okabe Combination meets claim 11. IMU teaches that distortions using control points in IMV6, 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, "the 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 using Shepard's Distortion to animate, 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.

209. In the IMU-Okabe Combination, the placement and movement of multiple Shepard's Distortion control points is determined by a user-drawn "stroke[]" on the image that specifies 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."

12. Independent Claim 12

**a. [12pre]: A computer program product comprising one or more non-transitory computer storage media having stored thereon computer-executable instructions that, when transmitted to a remote computer system for execution at a processor, cause the remote computer system to perform a method for automating a shifting of pixels within an image file, the method comprising.**

210. IMU "strongly recommend[s] to use an up-to-date version of IM[V6]" (IMU-Windows, 3; IMU-Home, 6), and hyperlinks the ImageMagick.org website's "Downloads Page" for users to download and install such a new version of IMV6 onto their computers (IMU-Home, 1). Following IMU's recommendations, a user would have thus downloaded and installed the IMU-Okabe Combination from this webpage—and thus a server (*see* Nakagawa, Abst.) (i.e., the claimed "computer program product comprising one or more non-transitory computer storage media having stored thereon computer-executable instructions")—onto the user's computer (i.e., "when transmitted to a remote computer system for execution at a processor, cause the remote computer system to perform"). *See* Section VIII.C.3.b.

211. Further, using the IMU-Okabe Combination meets the claimed "method for automating a shifting of pixels within an image file." Sections VIII.C.12.b-VIII.C.12.f.

**b. [12a]: 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 portion of a first image frame.**

212. *See* Sections VIII.C.4.b-VIII.C.4.c. Receiving the stroke from the user, including the stroke's starting point, meets the claimed "receiving a first indication of a first starting point through a user interface."

**c. [12b]: receiving, through the user interface, a first direction associated with the first starting point.**

213. *See* Sections VIII.C.4.d-VIII.C.4.e. Receiving the discussed user-drawn stroke, which indicates the Shepard's Distortion control point's starting and ending coordinates as well as direction and speed across the animation, meets the claimed "receiving, through the user interface, a first direction associated with the first starting point." *See* IMU-Distorting, 17-18, 59, 19-20; Okabe, 7, 2-3; Section VIII.C.3.b.

**d. [12c]: creating a first digital link extending in the first direction from the first starting point.**

214. *See* Section VIII.C.4.e. The discussed user-drawn stroke indicates the Shepard's Distortion control point's starting and ending coordinates as well as direction and speed across the animation, thus meeting the claimed "creating a first digital link extending in the first direction from the first starting point." *See* IMU-Distorting, 17-18, 59, 19-20; Okabe, 7, 2-3; Section VIII.C.3.b.

**e. [12d]: selecting a first set of pixels that are along the first digital link and extend in the first direction away from the first starting point.**

215. As discussed similarly in Section VIII.C.4.f, the IMU-Okabe Combination's user selects any or all pixels of the image to be included in a to-be-animated region of interest, including pixels that are along the stroke and extend in the stroke's direction away from the stroke's starting point—i.e., the claimed "selecting a first set of pixels that are along the first digital link and extend in the first direction away from the first starting point"—by using a matte. Indeed, Okabe's Figure 1 depicts a matte specifying a region of interest to be animated (in white) that includes pixels that are along three user-drawn strokes and extend in each stroke's direction away from each stroke's starting point. Okabe, 2.

**f. [12e]: shifting the first set of pixels, in the first image frame, in the first direction.**

216. *See* Section VIII.C.4.g.

13. Claim 13

**a. [13a]-[13b]: The computer program product as recited in claim 12, further comprising receiving an indication to generate a first mask over a second portion of the first image frame, wherein pixels under the first mask are prevented from shifting.**

217. In my opinion, the IMU-Okabe Combination meets limitations [13a]-[13b]. Similar to how IMV6's user applies a matte to modify the transparency of the pixels of an image (e.g., a frame image) and therefore the visibility of effects

applied to the pixels (IMU-Masking, 2; IMU-Animating, 7-8), the IMU-Okabe Combination's user applies a matte specifying a region of interest the user desires to animate (see Okabe, 1). Section VIII.C.3.b. Naturally, 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 white and black regions for pixels to be or not to be animated:

Okabe, 2 (Figure 1, annotations added). Therefore, the matte meets "a first mask over a second portion of the first image frame" as recited in limitation [13a], "wherein pixels under the first mask are prevented from shifting" as recited in limitation [13b]. Receiving and applying such a matte specified by the user meets "receiving an indication to generate a first mask" as recited in limitation [13a].

14. Claim 14: **The computer program product as recited in claim 13, further comprising computer-executable instructions that, when transmitted to the remote computer system for execution at the processor, cause the remote computer system to perform a method for automating the shifting of pixels within the image file, the method comprising receiving through a user interface a selection of the second portion of the first image frame around which the first mask should be generated.**

218. As discussed in Section VIII.C.13.a, the IMU-Okabe Combination's user applies a matte specifying a region of an image (in black) covering pixels the user does not desire to animate. *See* Okabe, 1-2. As seen in Okabe's Figure 1, such a matte includes a black region covering pixels the user does not desire to animate. Okabe, 2. Receiving such a region meets the claimed "receiving through a user interface a selection of the second portion of the first image frame around which the first mask should be generated."

15. Independent Claim 19

a. **[19pre]: A method for transmitting to a client computing device instructions for shifting pixels within a video file, comprising:**

**transmitting computer executable instructions to a client computing device, the computer executable instructions configured to cause the client computing device to.**

219. IMU "strongly recommend[s] to use an up-to-date version of IM[V6]" (IMU-Windows, 3; IMU-Home, 6), and hyperlinks the ImageMagick.org website's "Downloads Page" for users to download and install such a new version of IMV6

onto their computers (IMU-Home, 1). Following IMU's recommendations, a user would have thus downloaded and installed the IMU-Okabe Combination onto the user's computer (i.e., the claimed "client computing device") from this webpage—and thus a server (*see* Nakagawa, Abst.) (i.e., "transmitting to a client computing device instructions," and "transmitting computer executable instructions to a client computing device, the computer executable instructions configured to cause the client computing device to"). *See* Section VIII.C.3.b.

220. Further, the IMU-Okabe Combination is used "for shifting pixels within a video file." Sections VIII.C.15.b-VIII.C.15.g.

**b. [19a]: access, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within a frame of the digital image file.**

221. *See* Section VIII.C.4.b.

**c. [19b]: 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 a first image frame.**

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

**d. [19c]: 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.**

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

**e. [19d]: create a first digital link between the first starting point and the first ending point, wherein the first digital 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.**

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

**f. [19e]: identify a first set of pixels that lie along the first digital link between the first starting point and the first ending point; and.**

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

**g. [19f]: shift the first set of pixels in the first direction.**

226. *See* Section VIII.C.4.g.

16. Claim 20: **The method of claim 19, wherein the digital image file comprises a video file and the first image frame comprises a frame of the video file.**

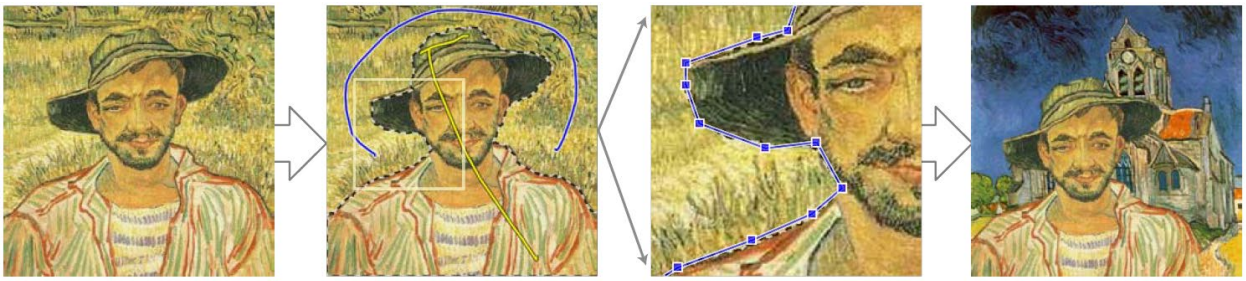
227. *See* Section VIII.C.6.

## D. Ground 3: IMU, Okabe, and Li, and Claims 13-15

### 1. Summary of Li

228. 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.

229. 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. "Lazy Snapping," 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 image cutout is performed in "Lazy Snapping" using these two steps:



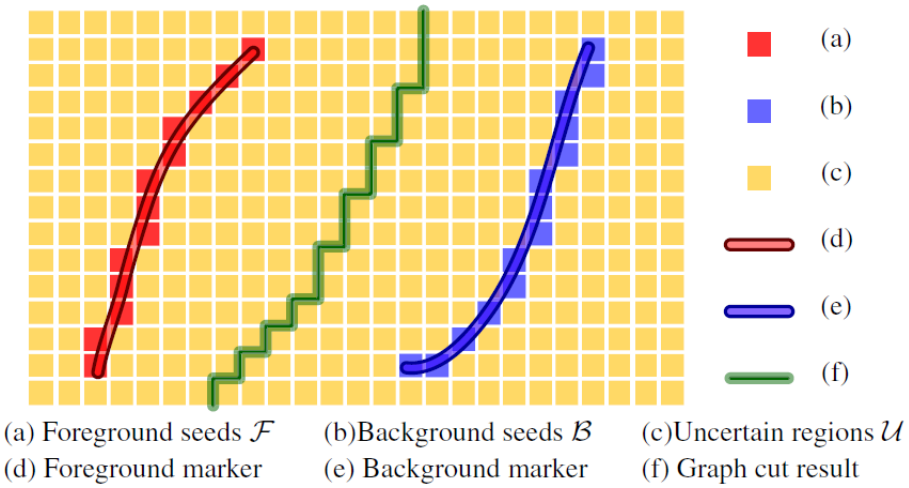
(a) Input image (b) Object Marking (c) Boundary editing (d) Output composition

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.

230. As Li teaches, the first "object marking" step "works at a coarse scale," where the user begins by "mark[ing] 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. After each marker line is drawn, a "segmentation process" is 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 (green) between a foreground line (red) and a background line (blue):




Li, 2-3.

231. 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. <u>The IMU-Okabe-Li Combination</u>

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

232. 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").

233. 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. Similarly, in the IMU-Okabe Combination, a user applies a matte that, as taught in Okabe, specifies a region of interest that the user desires to animate in an extracted frame image (shown in white), as well as a region the user does not desire to animate (shown in black). Okabe, 1-2; Sections VIII.C.3.b, VIII.C.13.a.

234. 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 explicitly teaches using "Lazy Snapping" to create a matte in Okabe and the IMU-Okabe Combination, and thus explicitly teaches combining the IMU-Okabe Combination with Li. Okabe, 7.

235. Separately, Li teaches that "Lazy Snapping" 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.

236. Given such benefits, a POSITA would have been motivated to utilize "Lazy Snapping" 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. Further, given that a user of the IMU-Okabe Combination applies a matte that, as taught in Okabe, specifies regions in an image (e.g., an extracted frame image) to be or not to be animated (*see* Okabe, 1; IMU-Animating, 7-8; 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.

237. 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.a), but also functionality similar to "Lazy Snapping." Particularly, similar to Li's "object marking" step (Li, 2-3), AECS6's "Roto Brush" tool 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

238. 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, e.g., an extracted frame image as discussed in Section VIII.C.3.b, to indicate the image's foreground (i.e., the region of interest that the user desires to animate) and 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 frame image's foreground and background based on the user's marker lines, as well as Li's "boundary editing" step for manual refinement of the boundary. *See* Li, 1-2; Okabe, 7. The resulting matte is then used in the remaining steps for animating the frame image discussed in Section VIII.C.3.b. *See* Okabe, 1, 7.

3. Claim 13

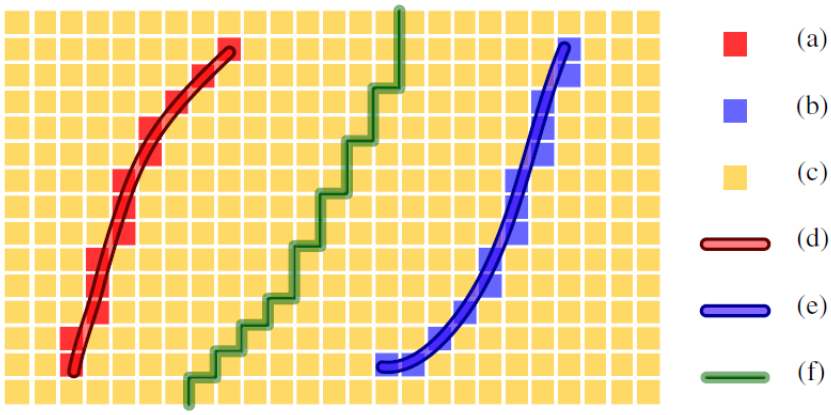
**a. [13a]-[13b]: The computer program product as recited in claim 12, further comprising receiving an indication to generate a first mask over a second portion of the first image frame, wherein pixels under the first mask are prevented from shifting.**

239. In my opinion, the IMU-Okabe-Li Combination meets limitations [13a]-[13b]. The IMU-Okabe-Li Combination allows a user to create a matte that specifies regions of an extracted frame image to animate (foreground) and not to animate (background). *See* Li, 1-2; Section VIII.D.2.b. Such a matte thus meets the claimed "first mask over a second portion of the first image frame, wherein pixels under the first mask are prevented from shifting" as recited in limitations [13a]-[13b].

240. To create this matte, the IMU-Okabe Combination's user performs Li's first "object marking" step of drawing marker lines on the image indicating the foreground and 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 "receiving an indication to generate a first mask over a second portion of the first image frame" recited in limitation [13a].

4. Claim 14: **The computer program product as recited in claim 13, further comprising computer-executable instructions that, when transmitted to the remote computer system for execution at the processor, cause the remote computer system to perform a method for automating the shifting of pixels within the image file, the method comprising receiving through a user interface a selection of the second portion of the first image frame around which the first mask should be generated.**

241. In my opinion, the IMU-Okabe-Li Combination meets claim 14. Li teaches that the "object marking" step of drawing marker lines on the image comprises drawing marker lines over the pixels that form the foreground and background. Li, 2. Specifically, Li teaches, "[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":




(a) Foreground seeds $\mathcal{F}$ (b)Background seeds $\mathcal{B}$ (c)Uncertain regions $\mathcal{U}$
(d) Foreground marker (e) Background marker (f) Graph cut result

Li, 2-3. Such foreground and background "pixels" or "seeds" are then used in Li's segmentation algorithm, as utilized by the IM-Okabe-Li Combination, to detect the boundary between the foreground and background. Li, 2-3; Section VIII.D.2.b. The

"pixels" or "seeds" of the marker lines thus meet the claimed "selection of the second portion of the first image frame around which the first mask should be generated." Li, 2-3. Receiving the foreground and background marker lines, and their intersecting "pixels" or "seeds," meets the claimed "receiving through a user interface a selection." Li, 2-3.

5. Claim 15

**a. [15a]-[15b]: The computer program product of claim 14, further comprising computer-executable instructions that, when transmitted to the remote computer system for execution at the processor, cause the remote computer system to perform a method for automating the shifting of pixels within the image file, the method comprising:**

**identifying one or more edges that form a first boundary around the second portion; and**

**generating the first mask to cover area within the first boundary.**

242. In my opinion, the IMU-Okabe-Li Combination meets limitations [15a]-[15b]. 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. Such "pixels" or "seeds" are used by the IM-Okabe-Li Combination to detect the boundary between the foreground and background for generating the matte, meeting the claimed "identifying one or more edges that form a first boundary around the second portion; and generating the first mask to cover area within the

first boundary" as recited in limitations [15a]-[15b]. Li, 2-3; Section VIII.D.4; *see also* '641 Patent, 4:46-64.

## IX. CONCLUSION

243. 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: Apr. 6, 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 16.687 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

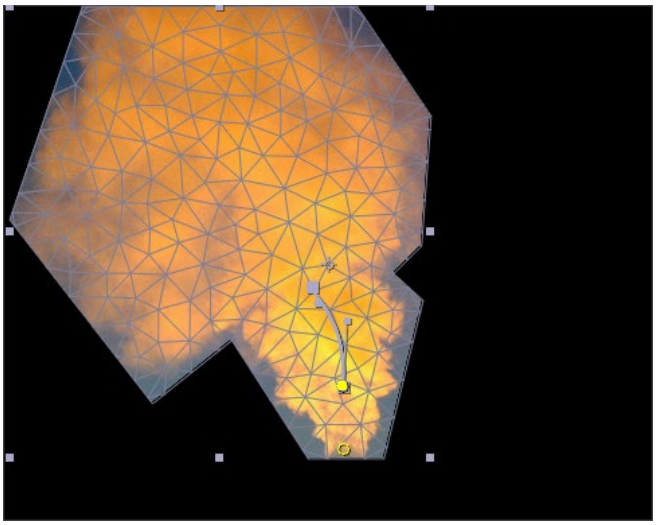
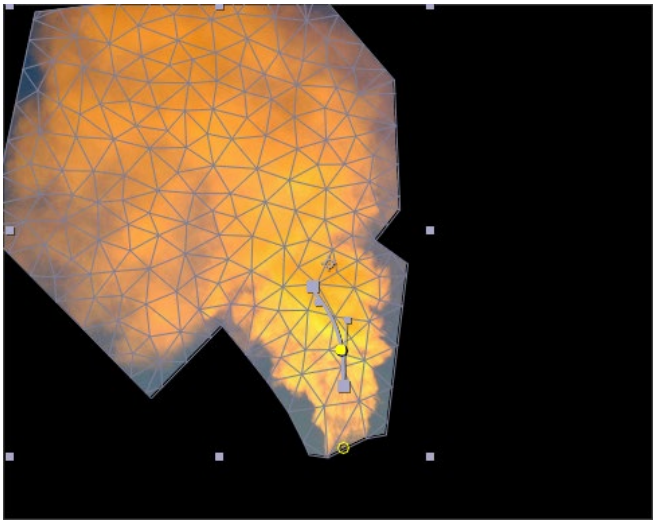
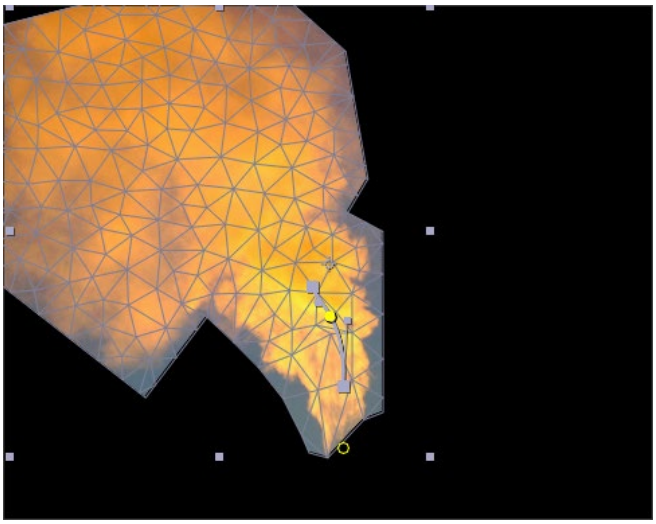
*Software Engineering for Internet Applications* (online and MIT Press 2006), *Philip and Alex's Guide to Web Publishing* (Morgan Kaufmann; 1999), *Database Backed Web Sites* (Ziff Davis Press; 1997), *Travels with Samantha*, a book about North America; SITE CONTROLLER: A system for computer-aided civil 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).

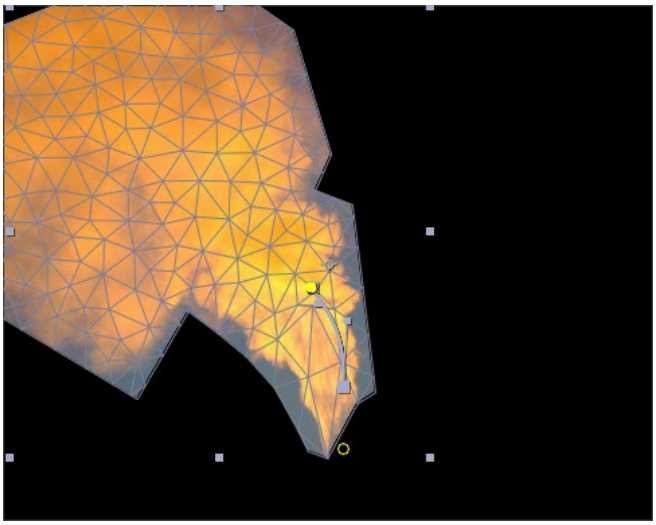
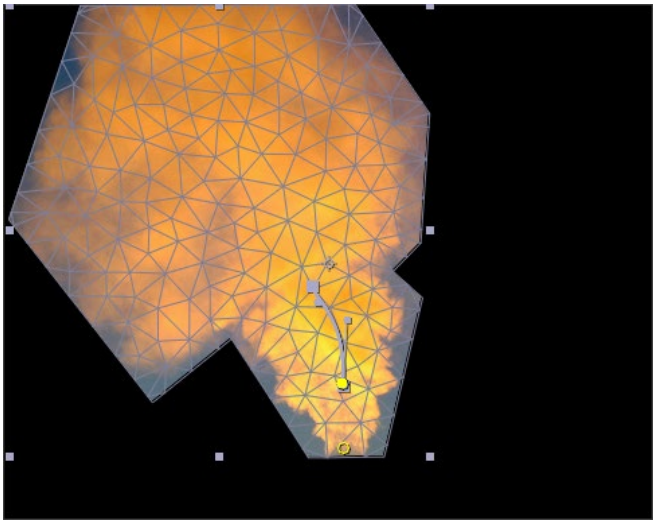
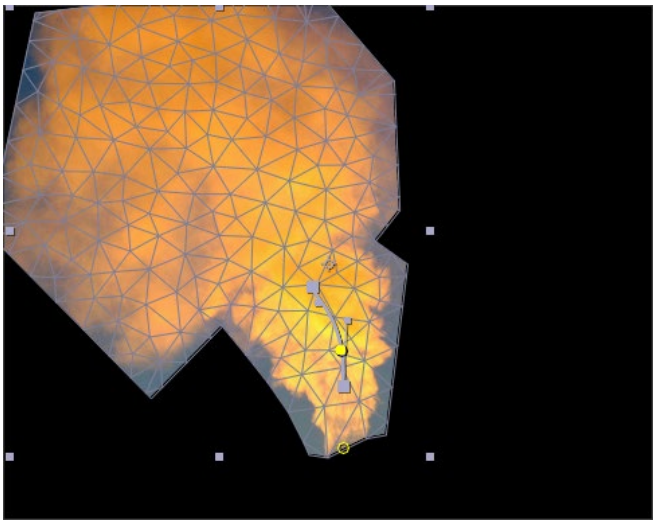
Most of my relevant publications are linked from philip.greenspun.com or fifthchance.com.

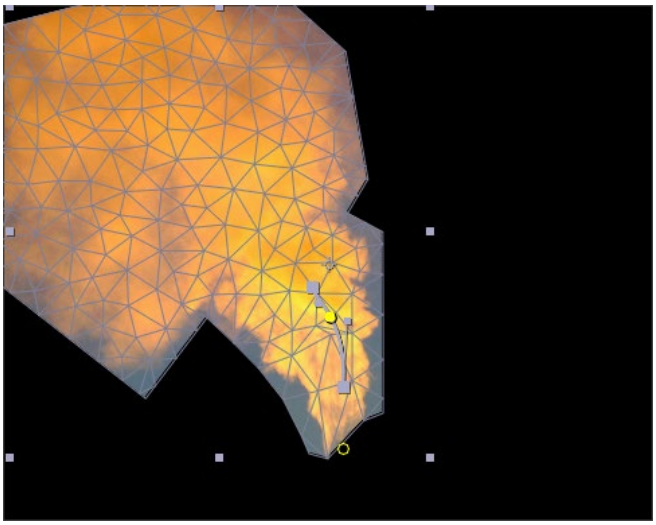
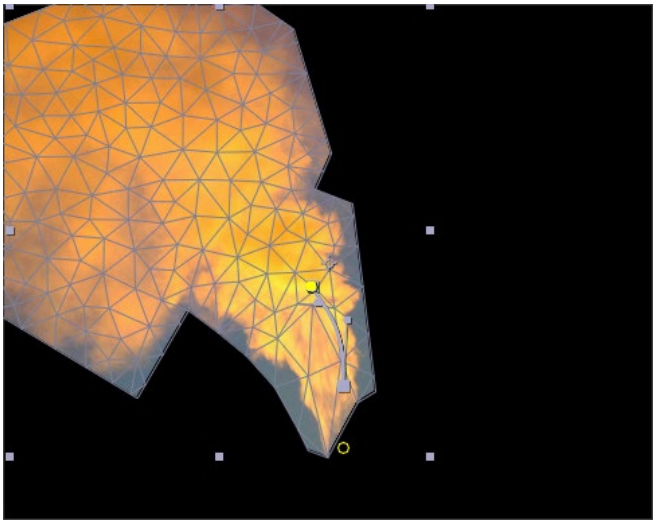
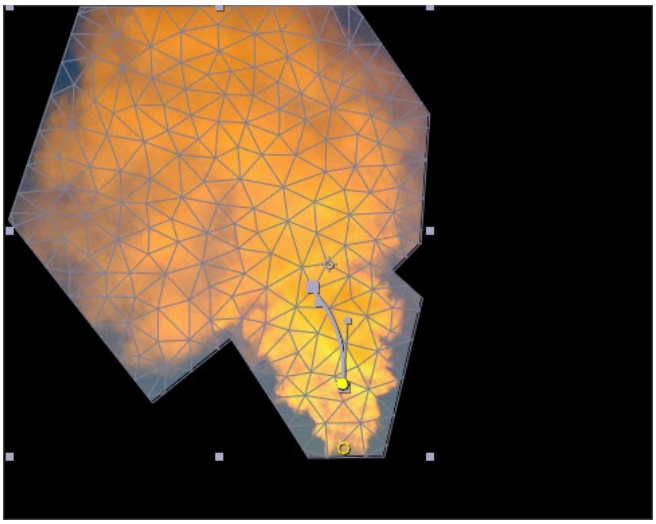
---

*philg@mit.edu*

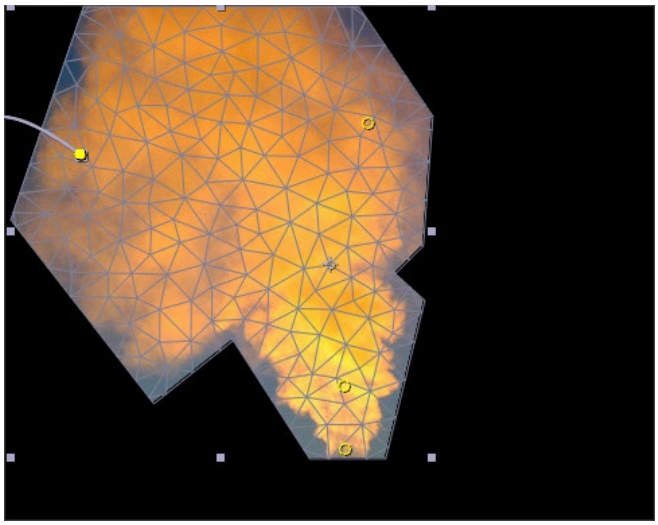
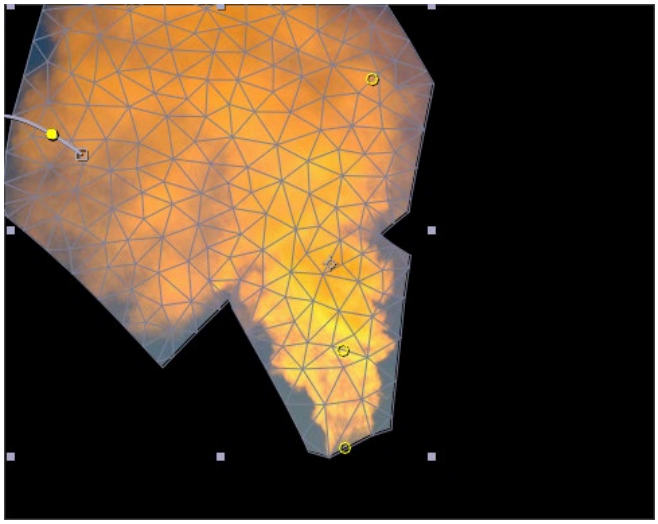
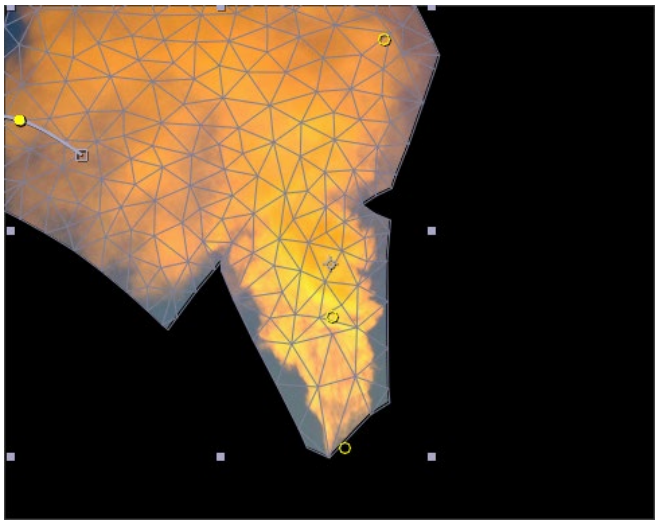
# ATTACHMENT B

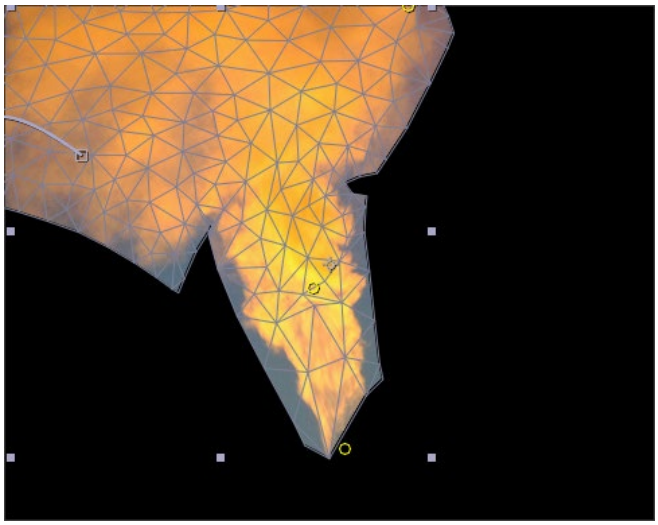
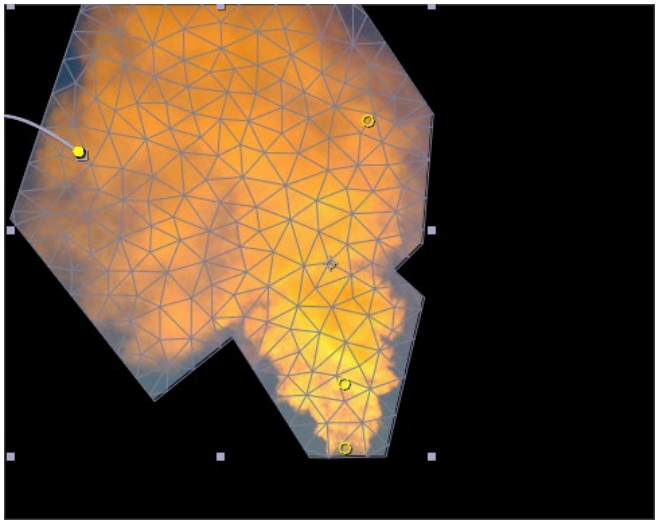
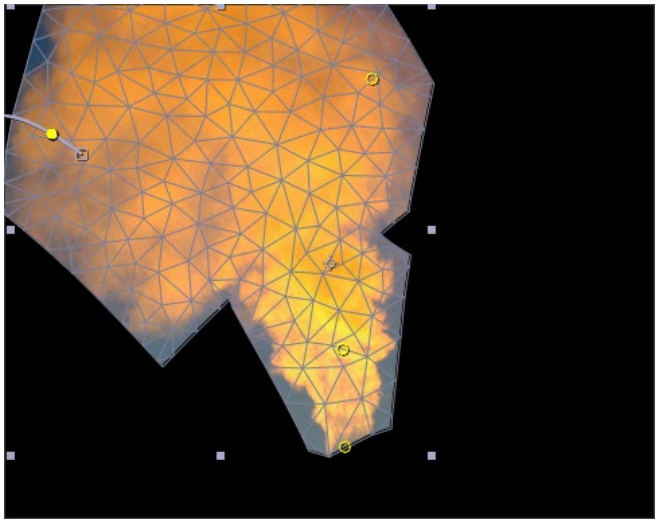
| Animation Time Point | Composition Frame |
|---|---|
| (0:00:00:00) |  |
| (0:00:01:00) |  |
| (0:00:02:00) |  |

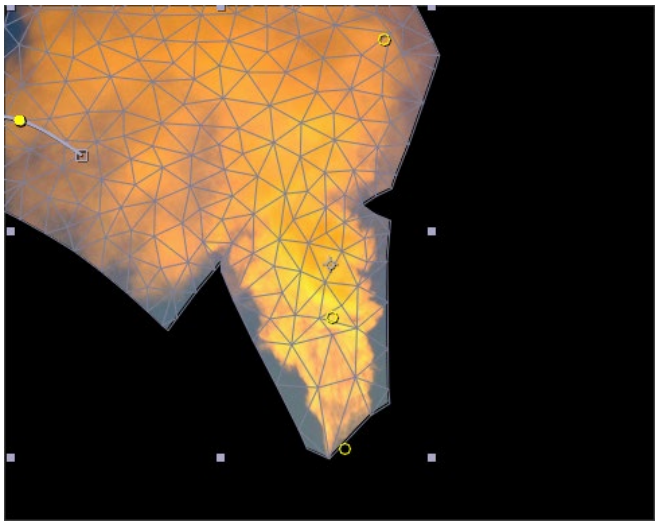
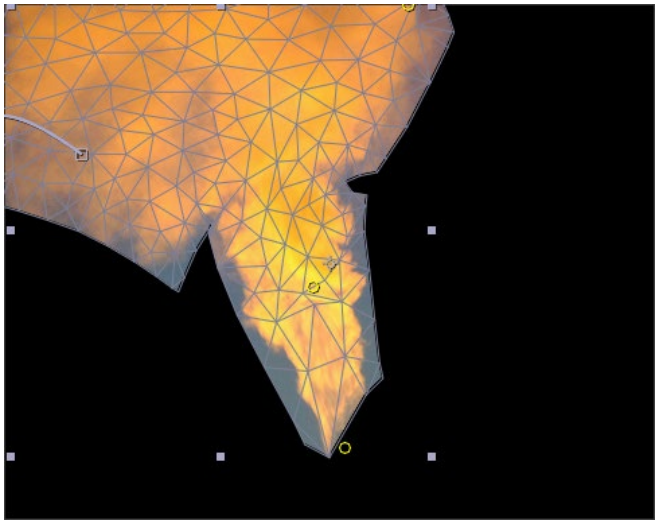
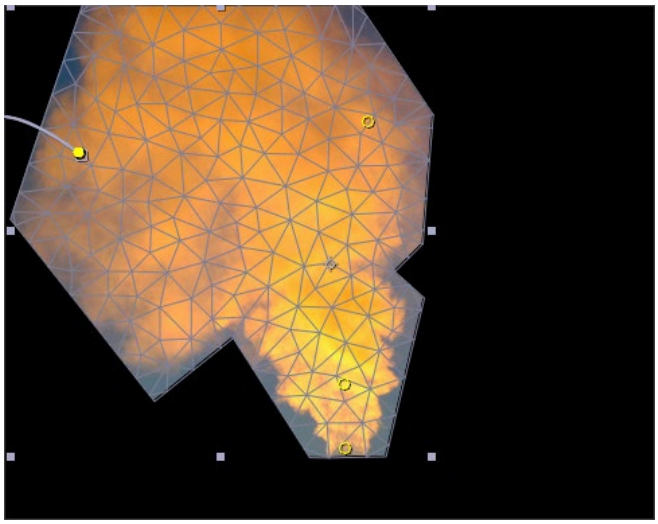
| Animation Time Point | Composition Frame |
|---|---|
| (0:00:02:23) |  |
| (0:00:03:01) |  |
| (0:00:04:00) |  |

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

# ATTACHMENT C

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

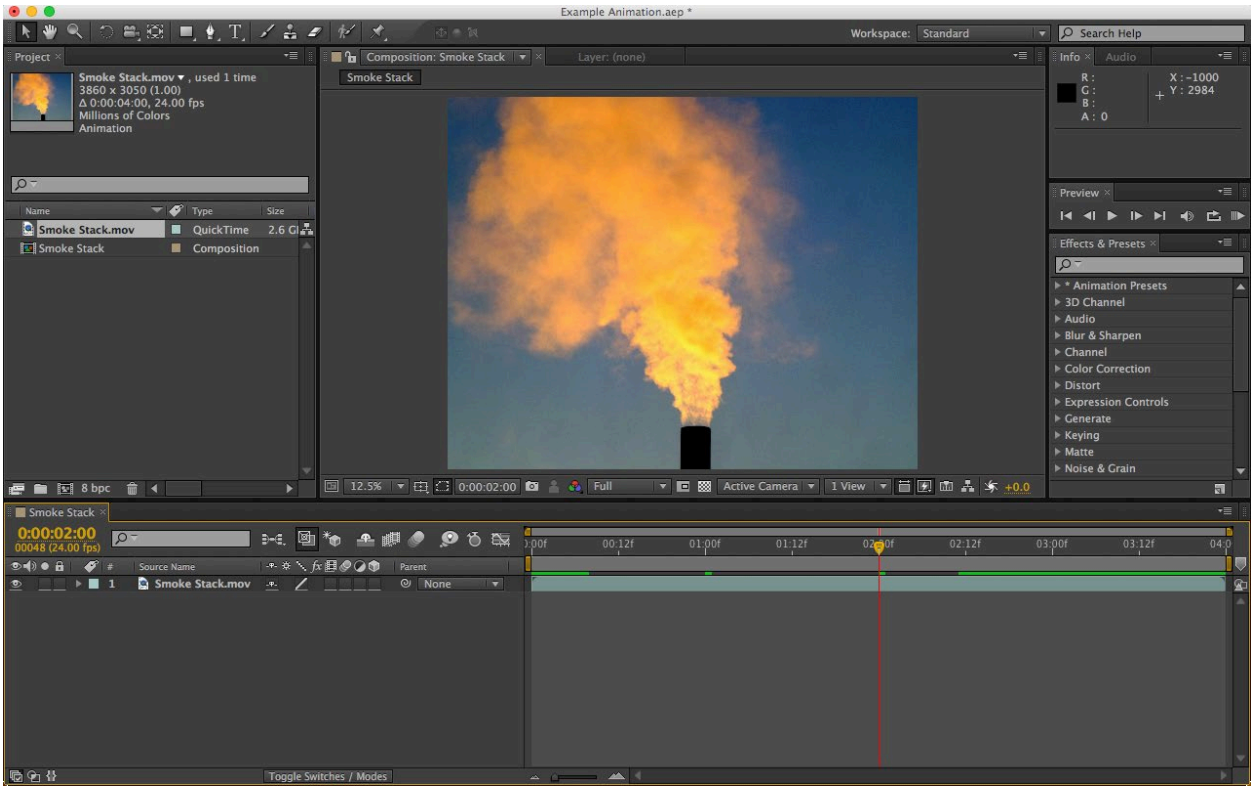
| Animation Time Point | Composition Frame |
|---|---|
| (0:00:02:23) |  |
| (0:00:03:01) |  |
| (0:00:04:00) |  |

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

# ATTACHMENT D

Example Animation.aep *
Workspace: Standard
Search Help
Project
Smoke Stack.mov ▾ , used 1 time
3860 x 3050 (1.00)
Δ 0:00:04:00, 24.00 fps
Millions of Colors
Animation
Composition: Smoke Stack
Layer: (none)
Smoke Stack
Info
Audio
R :
G :
B :
A : 0
X : -1000
Y : 2984
Preview
Effects & Presets
* Animation Presets
3D Channel
Audio
Blur & Sharpen
Channel
Color Correction
Distort
Expression Controls
Generate
Keying
Matte
Noise & Grain
Name
Type
Size
Smoke Stack.mov
QuickTime
2.6 GI
Smoke Stack
Composition
8 bpc
12.5%
0:00:00:00
Full
Active Camera
1 View
+0.0
Smoke Stack
0:00:00:00
00000 (24.00 fps)
00:12f
01:00f
01:12f
02:00f
02:12f
03:00f
03:12f
Source Name
Parent
1
Smoke Stack.mov
None
Toggle Switches / Modes

Example Animation.aep *
Workspace: Standard
Search Help
Project
Smoke Stack.mov ▾ , used 1 time
3860 x 3050 (1.00)
Δ 0:00:04:00, 24.00 fps
Millions of Colors
Animation
Composition: Smoke Stack
Layer: (none)
Smoke Stack
Info
Audio
R :
G :
B :
A : 0
X : -1000
Y : 2984
Preview
Effects & Presets
* Animation Presets
3D Channel
Audio
Blur & Sharpen
Channel
Color Correction
Distort
Expression Controls
Generate
Keying
Matte
Noise & Grain
Name
Type
Size
Smoke Stack.mov
QuickTime
2.6 GI
Smoke Stack
Composition
8 bpc
12.5%
0:00:01:00
Full
Active Camera
1 View
+0.0
Smoke Stack
0:00:01:00
00024 (24.00 fps)
00:12f
01:12f
02:00f
02:12f
03:00f
03:12f
Source Name
Parent
1
Smoke Stack.mov
None
Toggle Switches / Modes


Example Animation.aep *
Workspace: Standard
Search Help
Project
Smoke Stack.mov ▾ , used 1 time
3860 x 3050 (1.00)
Δ 0:00:04:00, 24.00 fps
Millions of Colors
Animation
Composition: Smoke Stack
Layer: (none)
Smoke Stack
Info
Audio
R :
G :
B :
A : 0
X : -1000
Y : 2984
Preview
Effects & Presets
* Animation Presets
3D Channel
Audio
Blur & Sharpen
Channel
Color Correction
Distort
Expression Controls
Generate
Keying
Matte
Noise & Grain
Name
Type
Size
Smoke Stack.mov
QuickTime
2.6 GI
Smoke Stack
Composition
8 bpc
12.5%
0:00:02:00
Full
Active Camera
1 View
+0.0
Smoke Stack
0:00:02:00
00048 (24.00 fps)
Source Name
Parent
1
Smoke Stack.mov
None
Toggle Switches / Modes

Example Animation.aep *
Workspace: Standard
Search Help
Project
Smoke Stack.mov ▾ , used 1 time
3860 x 3050 (1.00)
Δ 0:00:04:00, 24.00 fps
Millions of Colors
Animation
Composition: Smoke Stack
Layer: (none)
Smoke Stack
Info
Audio
R :
G :
B :
A : 0
X : -1000
Y : 2984
Preview
Effects & Presets
* Animation Presets
3D Channel
Audio
Blur & Sharpen
Channel
Color Correction
Distort
Expression Controls
Generate
Keying
Matte
Noise & Grain
Name
Type
Size
Smoke Stack.mov
QuickTime
2.6 GI
Smoke Stack
Composition
8 bpc
12.5%
0:00:03:00
Full
Active Camera
1 View
+0.0
Smoke Stack
0:00:03:00
00072 (24.00 fps)
Source Name
Parent
1
Smoke Stack.mov
None
Toggle Switches / Modes

Example Animation.aep *
Workspace: Standard
Search Help
Project
Smoke Stack.mov ▾ , used 1 time
3860 x 3050 (1.00)
Δ 0:00:04:00, 24.00 fps
Millions of Colors
Animation
Name
Type
Size
Smoke Stack.mov
QuickTime
2.6 GI
Smoke Stack
Composition
8 bpc
Composition: Smoke Stack
Layer: (none)
Smoke Stack
12.5%
0:00:03:23
Full
Active Camera
1 View
+0.0
Info
Audio
R :
G :
B :
A : 0
X : -1000
Y : 2984
Preview
Effects & Presets
* Animation Presets
3D Channel
Audio
Blur & Sharpen
Channel
Color Correction
Distort
Expression Controls
Generate
Keying
Matte
Noise & Grain
Smoke Stack
0:00:03:23
00095 (24.00 fps)
Source Name
Parent
1
Smoke Stack.mov
None
00:12f
01:00f
01:12f
02:00f
02:12f
03:00f
03:12f
Toggle Switches / Modes