Table of Contents


```
function MakeQTMovie(cmd,arg, arg2)
% function MakeQTMovie(cmd, arg, arg2)
% Create a QuickTime movie from a bunch of figures (and an optional
 sound).
%
% Syntax: MakeQTMovie cmd [arg]
% The following commands are supported:
% addfigure - Add snapshot of current figure to movie
%  addaxes - Add snapshot of current axes to movie
% addmatrix data - Add a matrix to movie (convert to jpeg with
 imwrite)
% addmatrixsc data - Add a matrix to movie (convert to jpeg with
 imwrite)
%  (automatically scales image data)
% addsound data [sr] - Add sound to movie (only monaural for now)
%  (third argument is the sound's sample rate.)
% cleanup - Remove the temporary files
% demo - Create a demonstration movie
%  finish - Finish movie, write out QT file
% framerate fps - Set movies frame rate [Default is 10 fps]
% quality # - Set JPEG quality (between 0 and 1)
%  size [# #] - Set plot size to [width height]
%  start filename - Start creating a movie with this name
% The start command must be called first to provide a movie name.
% The finish command must be called last to write out the movie
% data. All other commands can be called in any order.  Only one
% movie can be created at a time.
%
% This code is published as Interval Technical Report #1999-066
% The latest copy can be found at
% http://web.interval.com/papers/1999-066/
% (c) Copyright Malcolm Slaney, Interval Research, March 1999.

% This is experimental software and is being provided to Licensee
% 'AS IS.'  Although the software has been tested on Macintosh, SGI,
% Linux, and Windows machines, Interval makes no warranties relating
% to the software's performance on these or any other platforms.
%
% Disclaimer
% THIS SOFTWARE IS BEING PROVIDED TO YOU 'AS IS.'  INTERVAL MAKES
% NO EXPRESS, IMPLIED OR STATUTORY WARRANTY OF ANY KIND FOR THE
% SOFTWARE INCLUDING, BUT NOT LIMITED TO, ANY WARRANTY OF
```

```
% PERFORMANCE, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.
% IN NO EVENT WILL INTERVAL BE LIABLE TO LICENSEE OR ANY THIRD
% PARTY FOR ANY DAMAGES, INCLUDING LOST PROFITS OR OTHER INCIDENTAL
% OR CONSEQUENTIAL DAMAGES, EVEN IF INTERVAL HAS BEEN ADVISED OF
% THE POSSIBLITY THEREOF.
%
%   This software program is owned by Interval Research
% Corporation, but may be used, reproduced, modified and
% distributed by Licensee.  Licensee agrees that any copies of the
% software program will contain the same proprietary notices and
% warranty disclaimers which appear in this software program.

% This program uses the Matlab imwrite routine to convert each image
% frame into JPEG.  After first reserving 8 bytes for a header that
 points
% to the movie description, all the compressed images and the sound
 are
% added to the movie file.  When the 'finish' method is called then
 the
% first 8 bytes of the header are rewritten to indicate the size of
 the
% movie data, and then the movie header ('moov structure') is written
% to the output file.
%
% This routine creates files according to the QuickTime file format as
% described in the appendix of
% "Quicktime (Inside MacIntosh)," Apple Computer Incorporated,
% Addison-Wesley Pub Co; ISBN: 0201622017, April 1993.
% I appreciate help that I received from Lee Fyock (MathWorks) and
 Aaron
% Hertzmann (Interval) in debugging and testing this work.

% Changes:
% July 5, 1999 - Removed stss atom since it upset PC version of
 QuickTime
% November 11, 1999 - Fixed quality bug in addmatrix.  Added
 addmatrixsc.
% March 7, 2000 - by Jordan Rosenthal (jr@ece.gatech.edu), Added
 truecolor
%    capability when running in Matlab 5.3 changed some help comments,
 fixed
%    some bugs, vectorized some code.
% April 7, 2000 - by Malcolm.  Cleaned up axis/figure code and
 fixed(?) SGI
%    playback problems.  Added user data atom to give version
 information.
%    Fixed sound format problems.
% April 10, 2000 - by Malcolm. Fixed problem with SGI (at least) and
 B&W
%    addmatrix.

if nargin < 1
 fprintf('Syntax: MakeQTMovie cmd [arg]\n')
 fprintf('The following commands are supported:\n');
```

```
 fprintf(' addfigure - Add snapshot of current figure to movie\n')
 fprintf(' addaxes - Add snapshot of current axes to movie\n')
 fprintf(' addmatrix data - Add a matrix to movie ');
   fprintf('(convert to jpeg)\n')
 fprintf(' addmatrixsc data - Add a matrix to movie ');
   fprintf('(scale and convert to jpeg)\n')
 fprintf(' addsound data - Add sound samples ');
   fprintf('(with optional rate)\n')
 fprintf(' demo - Show this program in action\n');
 fprintf(' finish - Finish movie, write out QT file\n');
 fprintf(' framerate # - Set movie frame rate ');
   fprintf('(default is 10fps)\n');
 fprintf(' quality # - Set JPEG quality (between 0 and 1)\n');
 fprintf(' size [# #] - Set plot size to [width height]\n');
 fprintf(' start filename - Start making a movie with ');
   fprintf('this name\n');
 return;
end

global MakeQTMovieStatus
MakeDefaultQTMovieStatus;  % Needed first time, ignored otherwise

switch lower(cmd)
case {'addframe','addplot','addfigure','addaxes'}

 switch lower(cmd)
 case {'addframe','addfigure'}
  hObj = gcf;  % Add the entire figure (with all axes)
 otherwise
  hObj = gca;  % Add what's inside the current axis
 end
 frame = getframe(hObj);
 [I,map] = frame2im(frame);
 if ImageSizeChanged(size(I)) > 0
  return;
 end
 if isempty(map)
     % RGB image
  imwrite(I,MakeQTMovieStatus.imageTmp, 'jpg', 'Quality', ...
   MakeQTMovieStatus.spatialQual*100);
 else
     % Indexed image
  writejpg_map(MakeQTMovieStatus.imageTmp, I, map);
 end
 [pos, len] = AddFileToMovie;
 n = MakeQTMovieStatus.frameNumber + 1;
 MakeQTMovieStatus.frameNumber = n;
 MakeQTMovieStatus.frameStarts(n) = pos;
 MakeQTMovieStatus.frameLengths(n) = len;
```

Allow images to be added by doing:

```
%% MakeQTMovie('addimage', '/path/to/file.jpg');
```

This case adapted from addmatrix. Thanks to

Stephen Eglen stephen@cogsci.ed.ac.uk for this idea.

```
case 'addimage'

 if nargin < 2
  fprintf('MakeQTMovie error: Need to specify a filename with ');
  fprintf('the image command.\n');
  return;
 end
```

Check to see that the image is the correct size. Do

this by reading in the image and then checking its size.

```
 %% tim - temporary image.
        tim = imread(arg); tim_size = size(tim);

 fprintf('Image %s size %d %d\n', arg, tim_size(1), tim_size(2));
  if ImageSizeChanged(tim_size) > 0
   return;
  end
 [pos, len] = AddFileToMovie(arg);
 n = MakeQTMovieStatus.frameNumber + 1;
 MakeQTMovieStatus.frameNumber = n;
 MakeQTMovieStatus.frameStarts(n) = pos;
 MakeQTMovieStatus.frameLengths(n) = len;

case 'addmatrix'
 if nargin < 2
  fprintf('MakeQTMovie error: Need to specify a matrix with ');
  fprintf('the addmatrix command.\n');
  return;
 end
 if ImageSizeChanged(size(arg)) > 0
  return;
 end
     % Work around a bug, at least on the
     % SGIs, which causes JPEGs to be
     % written which can't be read with the
     % SGI QT.  Turn the B&W image into a
     % color matrix.
 if ndims(arg) < 3
  arg(:,:,2) = arg;
```

```
 arg(:,:,3) = arg(:,:,1);
end
imwrite(arg, MakeQTMovieStatus.imageTmp, 'jpg', 'Quality', ...
 MakeQTMovieStatus.spatialQual*100);
[pos, len] = AddFileToMovie;
n = MakeQTMovieStatus.frameNumber + 1;
MakeQTMovieStatus.frameNumber = n;
MakeQTMovieStatus.frameStarts(n) = pos;
MakeQTMovieStatus.frameLengths(n) = len;

case 'addmatrixsc'
if nargin < 2
 fprintf('MakeQTMovie error: Need to specify a matrix with ');
 fprintf('the addmatrix command.\n');
 return;
end
if ImageSizeChanged(size(arg)) > 0
 return;
end
arg = arg - min(min(arg));
arg = arg / max(max(arg));
    % Work around a bug, at least on the
    % SGIs, which causes JPEGs to be
    % written which can't be read with the
    % SGI QT.  Turn the B&W image into a
    % color matrix.
if ndims(arg) < 3
 arg(:,:,2) = arg;
 arg(:,:,3) = arg(:,:,1);
end
imwrite(arg, MakeQTMovieStatus.imageTmp, 'jpg', 'Quality', ...
 MakeQTMovieStatus.spatialQual*100);
[pos, len] = AddFileToMovie;
n = MakeQTMovieStatus.frameNumber + 1;
MakeQTMovieStatus.frameNumber = n;
MakeQTMovieStatus.frameStarts(n) = pos;
MakeQTMovieStatus.frameLengths(n) = len;

case 'addsound'
if nargin < 2
 fprintf('MakeQTMovie error: Need to specify a sound array ');
 fprintf('with the addsound command.\n');
 return;
end
    % Do stereo someday???
OpenMovieFile
MakeQTMovieStatus.soundLength = length(arg);
arg = round(arg/max(max(abs(arg)))*32765);
negs = find(arg<0);
arg(negs) = arg(negs) + 65536;

sound = mb16(arg);
MakeQTMovieStatus.soundStart = ftell(MakeQTMovieStatus.movieFp);
MakeQTMovieStatus.soundLen = length(sound);
```

```
 fwrite(MakeQTMovieStatus.movieFp, sound, 'uchar');
 if nargin < 3
  arg2 = 22050;
 end
 MakeQTMovieStatus.soundRate = arg2;

case 'cleanup'
 if isstruct(MakeQTMovieStatus)
  if ~isempty(MakeQTMovieStatus.movieFp)
   fclose(MakeQTMovieStatus.movieFp);
   MakeQTMovieStatus.movieFp = [];
  end
  if ~isempty(MakeQTMovieStatus.imageTmp) & ...
     exist(MakeQTMovieStatus.imageTmp,'file') > 0
   delete(MakeQTMovieStatus.imageTmp);
   MakeQTMovieStatus.imageTmp = [];
  end
 end
 MakeQTMovieStatus = [];

case 'debug'
 fprintf('Current Movie Data:\n');
 fprintf('    %d frames at %d fps\n',
 MakeQTMovieStatus.frameNumber, ...
     MakeQTMovieStatus.frameRate);
 starts = MakeQTMovieStatus.frameStarts;
 if length(starts) > 10, starts = starts(1:10);, end;
 lens = MakeQTMovieStatus.frameLengths;
 if length(lens) > 10, lens = lens(1:10);, end;
 fprintf('         Start: %6d      Size: %6d\n', [starts; lens]);
 fprintf('    Movie Image Size: %dx%d\n', ...
  MakeQTMovieStatus.imageSize(2), ...);
  MakeQTMovieStatus.imageSize(1));
 if length(MakeQTMovieStatus.soundStart) > 0
  fprintf('    Sound: %d samples at %d Hz sampling rate ', ...
   MakeQTMovieStatus.soundLength, ...
   MakeQTMovieStatus.soundRate);
  fprintf('at %d.\n', MakeQTMovieStatus.soundStart);
 else
  fprintf('    Sound: No sound track\n');
 end
 fprintf('    Temporary files for images: %s\n', ...
  MakeQTMovieStatus.imageTmp);
 fprintf('    Final movie name: %s\n', MakeQTMovieStatus.movieName);
 fprintf('    Compression Quality: %g\n', ...
  MakeQTMovieStatus.spatialQual);


case 'demo'
 clf
 fps = 10;
 movieLength = 10;
 sr = 22050;
 fn = 'test.mov';
```

```
 fprintf('Creating the movie %s.\n', fn);
 MakeQTMovie('start',fn);
 MakeQTMovie('size', [160 120]);
 MakeQTMovie('quality', 1.0);
 theSound = [];
 for i=1:movieLength
  plot(sin((1:100)/4+i));
  MakeQTMovie('addaxes');
  theSound = [theSound sin(440/sr*2*pi*(2^(i/12))*(1:sr/fps))];
 end
 MakeQTMovie('framerate', fps);
 MakeQTMovie('addsound', theSound, sr);
 MakeQTMovie('finish');

case {'finish','close'}
 AddQTHeader;
 MakeQTMovie('cleanup')   % Remove temporary files
 %MakeDefaultQTMovieStatus;

case 'framerate'
 if nargin < 2
  fprintf('MakeQTMovie error: Need to specify the ');
  fprintf('frames/second with the framerate command.\n');
  return;
 end
 MakeQTMovieStatus.frameRate = arg;

case 'help'
 MakeQTMovie    % To get help message.

case 'size'
      % Size is off by one on the
      % Mac.
 if nargin < 2
  fprintf('MakeQTMovie error: Need to specify a vector with ');
  fprintf('the size command.\n');
  return;
 end
 if length(arg) ~= 2
  error('MakeQTMovie: Error, must supply 2 element size.');
 end
 oldUnits = get(gcf,'units');
 set(gcf,'units','pixels');
 cursize = get(gcf, 'position');
 cursize(3) = arg(1);
 cursize(4) = arg(2);
 set(gcf, 'position', cursize);
 set(gcf,'units',oldUnits);

case 'start'
 if nargin < 2
  fprintf('MakeQTMovie error: Need to specify a file name ');
  fprintf('with start command.\n');
  return;
```

```
 end
 MakeQTMovie('cleanup');
 MakeDefaultQTMovieStatus;
 MakeQTMovieStatus.movieName = arg;

case 'test'
 clf
 MakeQTMovieStatus = [];
 MakeQTMovie('start','test.mov');
  MakeQTMovie('size', [320 240]);
 MakeQTMovie('quality', 1.0);
 subplot(2,2,1);
 for i=1:10
  plot(sin((1:100)/4+i));
  MakeQTMovie('addfigure');
 end
 MakeQTMovie('framerate', 10);
 MakeQTMovie('addsound', sin(1:5000), 22050);
 MakeQTMovie('debug');
 MakeQTMovie('finish');

case 'quality'
 if nargin < 2
  fprintf('MakeQTMovie error: Need to specify a quality ');
  fprintf('(between 0-1) with the quality command.\n');
  return;
 end
 MakeQTMovieStatus.spatialQual = arg;

otherwise
 fprintf('MakeQTMovie: Unknown method %s.\n', cmd);
end

%%%%%%%%%%%%%%%%  MakeDefaultQTMovieStatus %%%%%%%%%%%%%%%%%%
% Make the default movie status structure.
function MakeDefaultQTMovieStatus
global MakeQTMovieStatus
if isempty(MakeQTMovieStatus)
   MakeQTMovieStatus = struct(...
      'frameRate', 10, ... % frames per second
      'frameStarts', [], ...  % Starting byte position
      'frameLengths', [], ...
      'timeScale', 10, ... % How much faster does time run?
      'soundRate', 22050, ... % Sound Sample Rate
      'soundStart', [], ... % Starting byte position
      'soundLength', 0, ...
      'soundChannels', 1, ... % Number of channels
      'frameNumber', 0, ...
      'movieFp', [], ...  % File pointer
      'imageTmp', tempname, ...
      'movieName', 'output.mov', ...
      'imageSize', [0 0], ...
      'trackNumber', 0, ...
      'timeScaleExpansion', 100, ...
```

```
      'spatialQual', 1.0); % Between 0.0 and 1.0
end


%%%%%%%%%%%%%%%  ImageSizeChanged %%%%%%%%%%%%%%%%%
% Check to see if the image size has changed.  This m-file can't
% deal with that, so we'll return an error.
function err = ImageSizeChanged(newsize)
global MakeQTMovieStatus

newsize = newsize(1:2);   % Don't care about RGB info, if present
oldsize = MakeQTMovieStatus.imageSize;
err = 0;

if sum(oldsize) == 0
 MakeQTMovieStatus.imageSize = newsize;
else
 if sum(newsize ~= oldsize) > 0
  fprintf('MakeQTMovie Error: New image size');
  fprintf('(%dx%d) doesn''t match old size (%dx%d)\n', ...
   newsize(1), newsize(2), oldsize(1), oldsize(2));
  fprintf('   Can''t add this image to the movie.\n');
  err = 1;
 end
end

%%%%%%%%%%%%%%%  AddFileToMovie %%%%%%%%%%%%%%%%%
% OK, we've saved out an image file.  Now add it to the end of the
 movie
% file we are creating.
% We'll copy the JPEG file in 16kbyte chunks to the end of the movie
 file.
% Keep track of the start and end byte position in the file so we can
 put
% the right information into the QT header.
function [pos, len] = AddFileToMovie(imageTmp)
global MakeQTMovieStatus
OpenMovieFile
if nargin < 1
 imageTmp = MakeQTMovieStatus.imageTmp;
end
fp = fopen(imageTmp, 'rb');
if fp < 0
 error('Could not reopen QT image temporary file.');
end

len = 0;
pos = ftell(MakeQTMovieStatus.movieFp);
while 1
 data = fread(fp, 1024*16, 'uchar');
 if isempty(data)
  break;
 end
 cnt = fwrite(MakeQTMovieStatus.movieFp, data, 'uchar');
```

```
 len = len + cnt;
end
fclose(fp);

%%%%%%%%%%%%%%%  AddQTHeader %%%%%%%%%%%%%%%%
% Go back and write the atom information that allows
% QuickTime to skip the image and sound data and find
% its movie description information.
function AddQTHeader()
global MakeQTMovieStatus

pos = ftell(MakeQTMovieStatus.movieFp);
header = moov_atom;
cnt = fwrite(MakeQTMovieStatus.movieFp, header, 'uchar');
fseek(MakeQTMovieStatus.movieFp, 0, -1);
cnt = fwrite(MakeQTMovieStatus.movieFp, mb32(pos), 'uchar');
fclose(MakeQTMovieStatus.movieFp);
MakeQTMovieStatus.movieFp = [];

%%%%%%%%%%%%%%%  OpenMovieFile %%%%%%%%%%%%%%%%%
% Open a new movie file.  Write out the initial QT header.  We'll fill
 in
% the correct length later.
function OpenMovieFile
global MakeQTMovieStatus
if isempty(MakeQTMovieStatus.movieFp)
 fp = fopen(MakeQTMovieStatus.movieName, 'wb');
 if fp < 0
  error('Could not open QT movie output file.');
 end
 MakeQTMovieStatus.movieFp = fp;
 cnt = fwrite(fp, [mb32(0) mbstring('mdat')], 'uchar');
end

%%%%%%%%%%%%%%%  writejpg_map %%%%%%%%%%%%%%%%%
% Like the imwrite routine, but first pass the image data through the
 indicated
% RGB map.
function writejpg_map(name,I,map)
global MakeQTMovieStatus

[y,x] = size(I);

% Force values to be valid indexes.  This fixes a bug that
 occasionally
% occurs in frame2im in Matlab 5.2 which incorrectly produces values
 of I
% equal to zero.
I = max(1,min(I,size(map,1)));

rgb = zeros(y, x, 3);
t = zeros(y,x);
t(:) = map(I(:),1)*255; rgb(:,:,1) = t;
t(:) = map(I(:),2)*255; rgb(:,:,2) = t;
```

```
t(:) = map(I(:),3)*255; rgb(:,:,3) = t;

imwrite(uint8(rgb),name,'jpeg','Quality',MakeQTMovieStatus.spatialQual*100);

%%%%%%%%%%%%%%%  SetAtomSize %%%%%%%%%%%%%%%%%
% Fill in the size of the atom
function y=SetAtomSize(x)
y = x;
y(1:4) = mb32(length(x));

%%%%%%%%%%%%%%%  mb32 %%%%%%%%%%%%%%%%%
% Make a vector from a 32 bit integer
function y = mb32(x)
if size(x,1) > size(x,2)
 x = x';
end

y = [bitand(bitshift(x,-24),255); ...
     bitand(bitshift(x,-16),255); ...
     bitand(bitshift(x, -8),255); ...
     bitand(x,              255)];
y = y(:)';

%%%%%%%%%%%%%%%  mb16 %%%%%%%%%%%%%%%%%
% Make a vector from a 16 bit integer
function y = mb16(x)
if size(x,1) > size(x,2)
 x = x';
end

y = [bitand(bitshift(x, -8),255); ...
     bitand(x,              255)];
y = y(:)';

%%%%%%%%%%%%%%%  mb8 %%%%%%%%%%%%%%%%%
% Make a vector from a 8 bit integer
function y = mb8(x)
if size(x,1) > size(x,2)
 x = x';
end

y = [bitand(x,              255)];
y = y(:)';

%
% The following routines all create atoms necessary
% to describe a QuickTime Movie. The basic idea is to
% fill in the necessary data, all converted to 8 bit
% characters, then fix it up later with SetAtomSize so
% that it has the correct header.  (This is easier than
% counting by hand.)

%%%%%%%%%%%%%%%  mbstring %%%%%%%%%%%%%%%%%
% Make a vector from a character string
```

```
function y = mbstring(s)
y = double(s);


%%%%%%%%%%%%%%%  dinf_atom %%%%%%%%%%%%%%%%%
function y = dinf_atom()
y = SetAtomSize([mb32(0) mbstring('dinf') dref_atom]);

%%%%%%%%%%%%%%%  dref_atom %%%%%%%%%%%%%%%%%
function y = dref_atom()
y = SetAtomSize([mb32(0) mbstring('dref') mb32(0) mb32(1) ...
  mb32(12) mbstring('alis') mb32(1)]);

%%%%%%%%%%%%%%%  edts_atom %%%%%%%%%%%%%%%%%
function y = edts_atom(add_sound_p)
global MakeQTMovieStatus
fixed1 = bitshift(1,16);   % Fixed point 1
if add_sound_p > 0
 duration = MakeQTMovieStatus.soundLength / ...
   MakeQTMovieStatus.soundRate * ...
   MakeQTMovieStatus.timeScale;
else
 duration = MakeQTMovieStatus.frameNumber / ...
   MakeQTMovieStatus.frameRate * ...
   MakeQTMovieStatus.timeScale;
end
duration = ceil(duration);

y = [mb32(0) ...    % Atom Size
     mbstring('edts') ...   % Atom Name
     SetAtomSize([mb32(0) ...   % Atom Size
    mbstring('elst') ...  % Atom Name
    mb32(0) ...   % Version/Flags
    mb32(1) ...   % Number of entries
    mb32(duration) ...  % Length of this track
    mb32(0) ...   % Time
    mb32(fixed1)])];  % Rate
y = SetAtomSize(y);

%%%%%%%%%%%%%%%  hdlr_atom %%%%%%%%%%%%%%%%%
function y = hdlr_atom(component_type, sub_type)
if strcmp(sub_type, 'vide')
 type_string = 'Apple Video Media Handler';
elseif strcmp(sub_type, 'alis')
 type_string = 'Apple Alias Data Handler';
elseif strcmp(sub_type, 'soun')
 type_string = 'Apple Sound Media Handler';
end

y = [mb32(0) ...    % Atom Size
     mbstring('hdlr') ...   % Atom Name
     mb32(0) ...    % Version and Flags
     mbstring(component_type) ...  % Component Name
     mbstring(sub_type) ...   % Sub Type Name
```

```
     mbstring('appl') ...   % Component manufacturer
     mb32(0) ...    % Component flags
     mb32(0) ...    % Component flag mask
     mb8(length(type_string)) ...  % Type Name byte count
     mbstring(type_string)];   % Type Name
y = SetAtomSize(y);

%%%%%%%%%%%%%%%  mdhd_atom %%%%%%%%%%%%%%%%%
function y = mdhd_atom(add_sound_p)
global MakeQTMovieStatus

if add_sound_p
 data = [mb32(MakeQTMovieStatus.soundRate)  ...
  mb32(MakeQTMovieStatus.soundLength)];
else
 data = [mb32(MakeQTMovieStatus.frameRate * ...
   MakeQTMovieStatus.timeScaleExpansion)  ...
  mb32(MakeQTMovieStatus.frameNumber * ...
   MakeQTMovieStatus.timeScaleExpansion)];
end

y = [mb32(0) mbstring('mdhd') ...  % Atom Header
     mb32(0) ...
     mb32(round(now*3600*24)) ...  % Creation time
     mb32(round(now*3600*24)) ...  % Modification time
     data ...
     mb16(0) mb16(0)];
y = SetAtomSize(y);

%%%%%%%%%%%%%%%  mdia_atom %%%%%%%%%%%%%%%%%
function y = mdia_atom(add_sound_p)
global MakeQTMovieStatus

if add_sound_p
 hdlr = hdlr_atom('mhlr', 'soun');
else
 hdlr = hdlr_atom('mhlr', 'vide');
end

y = [mb32(0) mbstring('mdia') ...  % Atom Header
     mdhd_atom(add_sound_p) ...
     hdlr ...     % Handler Atom
     minf_atom(add_sound_p)];
y = SetAtomSize(y);


%%%%%%%%%%%%%%%  minf_atom %%%%%%%%%%%%%%%%%
function y = minf_atom(add_sound_p)
global MakeQTMovieStatus

if add_sound_p
 data = smhd_atom;
else
 data = vmhd_atom;
```

```
end

y = [mb32(0) mbstring('minf') ...  % Atom Header
     data ...
     hdlr_atom('dhlr','alis') ...
     dinf_atom ...
     stbl_atom(add_sound_p)];
y = SetAtomSize(y);

%%%%%%%%%%%%%%%  moov_atom %%%%%%%%%%%%%%%%%
function y=moov_atom
global MakeQTMovieStatus
MakeQTMovieStatus.timeScale = MakeQTMovieStatus.frameRate * ...
    MakeQTMovieStatus.timeScaleExpansion;

if MakeQTMovieStatus.soundLength > 0
 sound = trak_atom(1);
else
 sound = [];
end

y = [mb32(0) mbstring('moov') ...
     mvhd_atom udat_atom sound trak_atom(0) ];
y = SetAtomSize(y);

%%%%%%%%%%%%%%%  mvhd_atom %%%%%%%%%%%%%%%%%
function y=mvhd_atom
global MakeQTMovieStatus

fixed1 = bitshift(1,16);   % Fixed point 1
frac1 = bitshift(1,30);    % Fractional 1
if length(MakeQTMovieStatus.soundStart) > 0
 NumberOfTracks = 2;
else
 NumberOfTracks = 1;
end

     % Need to make sure its longer
     % of movie and sound lengths
MovieDuration = max(MakeQTMovieStatus.frameNumber / ...
   MakeQTMovieStatus.frameRate, ...
      MakeQTMovieStatus.soundLength / ...
   MakeQTMovieStatus.soundRate);
MovieDuration = ceil(MovieDuration * MakeQTMovieStatus.timeScale);

y = [mb32(0) ...   % Size
     mbstring('mvhd') ...  % Movie Data
     mb32(0) ...   % Version and Flags
     mb32(0) ...   % Creation Time (unknown)
     mb32(0) ...   % Modification Time (unknown)
     mb32(MakeQTMovieStatus.timeScale) ... % Movie's Time Scale
     mb32(MovieDuration) ...  % Movie Duration
     mb32(fixed1) ...   % Preferred Rate
     mb16(255) ...   % Preferred Volume
```

```
     mb16(0) ...   % Fill
     mb32(0) ...   % Fill
     mb32(0) ...   % Fill
     mb32(fixed1) mb32(0) mb32(0) ... % Transformation matrix
 (identity)
     mb32(0) mb32(fixed1) mb32(0) ...
     mb32(0) mb32(0) mb32(frac1) ...
     mb32(0) ...   % Preview Time
     mb32(0) ...   % Preview Duration
     mb32(0) ...   % Poster Time
     mb32(0) ...   % Selection Time
     mb32(0) ...   % Selection Duration
     mb32(0) ...   % Current Time
     mb32(NumberOfTracks)];  % Video and/or Sound?

y = SetAtomSize(y);

%%%%%%%%%%%%%%%  raw_image_description %%%%%%%%%%%%%%%%%
function y = raw_image_description()
global MakeQTMovieStatus

fixed1 = bitshift(1,16);   % Fixed point 1
codec = [12 'Photo - JPEG                     '];

y = [mb32(0) mbstring('jpeg') ...  % Atom Header
     mb32(0) mb16(0) mb16(0) mb16(0) mb16(1) ...
     mbstring('appl') ...
     mb32(1023) ...    % Temporal Quality (perfect)
     mb32(floor(1023*MakeQTMovieStatus.spatialQual)) ...
     mb16(MakeQTMovieStatus.imageSize(2)) ...
     mb16(MakeQTMovieStatus.imageSize(1)) ...
     mb32(fixed1 * 72) mb32(fixed1 * 72) ...
     mb32(0) ...
     mb16(0) ...
     mbstring(codec) ...
     mb16(24) mb16(65535)];
y = SetAtomSize(y);


%%%%%%%%%%%%%%%  raw_sound_description %%%%%%%%%%%%%%%%%
function y = raw_sound_description()
global MakeQTMovieStatus
y = [mb32(0) mbstring('twos') ...  % Atom Header
     mb32(0) mb16(0) mb16(0) mb16(0) mb16(0) ...
     mb32(0) ...
     mb16(MakeQTMovieStatus.soundChannels) ...
     mb16(16) ...    % 16 bits per sample
     mb16(0) mb16(0) ...
     mb32(round(MakeQTMovieStatus.soundRate*65536))];
y = SetAtomSize(y);


%%%%%%%%%%%%%%%  smhd_atom %%%%%%%%%%%%%%%%%
function y = smhd_atom()
```

```
y = SetAtomSize([mb32(0) mbstring('smhd') mb32(0) mb16(0) mb16(0)]);

%%%%%%%%%%%%%%%  stbl_atom %%%%%%%%%%%%%%%%%
% Removed the stss atom since it seems to upset the PC version of QT
% and it is empty so it doesn't add anything.
% Malcolm - July 5, 1999
function y = stbl_atom(add_sound_p)
y = [mb32(0) mbstring('stbl') ...  % Atom Header
     stsd_atom(add_sound_p) ...
     stts_atom(add_sound_p) ...
     stsc_atom(add_sound_p) ...
     stsz_atom(add_sound_p) ...
     stco_atom(add_sound_p)];
y = SetAtomSize(y);

%%%%%%%%%%%%%%%  stco_atom %%%%%%%%%%%%%%%%%
function y = stco_atom(add_sound_p)
global MakeQTMovieStatus
if add_sound_p
 y = [mb32(0) mbstring('stco') mb32(0) mb32(1) ...
      mb32(MakeQTMovieStatus.soundStart)];
else
 y = [mb32(0) mbstring('stco') mb32(0) ...
      mb32(MakeQTMovieStatus.frameNumber) ...
      mb32(MakeQTMovieStatus.frameStarts)];
end
y = SetAtomSize(y);

%%%%%%%%%%%%%%%  stsc_atom %%%%%%%%%%%%%%%%%
function y = stsc_atom(add_sound_p)
global MakeQTMovieStatus
if add_sound_p
 samplesperchunk = MakeQTMovieStatus.soundLength;
else
 samplesperchunk = 1;
end

y = [mb32(0) mbstring('stsc') mb32(0) mb32(1)  ...
     mb32(1) mb32(samplesperchunk) mb32(1)];
y = SetAtomSize(y);

%%%%%%%%%%%%%%%  stsd_atom %%%%%%%%%%%%%%%%%
function y = stsd_atom(add_sound_p)
if add_sound_p
 desc = raw_sound_description;
else
 desc = raw_image_description;
end

y = [mb32(0) mbstring('stsd') mb32(0) mb32(1) desc];
y = SetAtomSize(y);

%%%%%%%%%%%%%%%  stss_atom %%%%%%%%%%%%%%%%%
function y = stss_atom()
```

```
y = SetAtomSize([mb32(0) mbstring('stss') mb32(0) mb32(0)]);

%%%%%%%%%%%%%%%  stsz_atom %%%%%%%%%%%%%%%%%
function y = stsz_atom(add_sound_p)
global MakeQTMovieStatus
if add_sound_p
 y = [mb32(0) mbstring('stsz') mb32(0) mb32(2) ...
      mb32(MakeQTMovieStatus.soundLength)];
else
 y = [mb32(0) mbstring('stsz') mb32(0) mb32(0) ...
      mb32(MakeQTMovieStatus.frameNumber) ...
      mb32(MakeQTMovieStatus.frameLengths)];
end
y = SetAtomSize(y);

%%%%%%%%%%%%%%%  stts_atom %%%%%%%%%%%%%%%%%
function y = stts_atom(add_sound_p)
global MakeQTMovieStatus
if add_sound_p
 count_duration = [mb32(MakeQTMovieStatus.soundLength) mb32(1)];
else
 count_duration = [mb32(MakeQTMovieStatus.frameNumber) ...
  mb32(MakeQTMovieStatus.timeScaleExpansion)];
end

y = SetAtomSize([mb32(0) mbstring('stts') mb32(0) mb32(1)
 count_duration]);

%%%%%%%%%%%%%%%  trak_atom %%%%%%%%%%%%%%%%%
function y = trak_atom(add_sound_p)
global MakeQTMovieStatus

y = [mb32(0) mbstring('trak') ...  % Atom Header
 tkhd_atom(add_sound_p) ...  % Track header
 edts_atom(add_sound_p) ...  % Edit List
 mdia_atom(add_sound_p)];
y = SetAtomSize(y);

%%%%%%%%%%%%%%%  tkhd_atom %%%%%%%%%%%%%%%%%
function y = tkhd_atom(add_sound_p)
global MakeQTMovieStatus

fixed1 = bitshift(1,16);   % Fixed point 1
frac1 = bitshift(1,30);    % Fractional 1 (CHECK THIS)

if add_sound_p > 0
 duration = MakeQTMovieStatus.soundLength / ...
   MakeQTMovieStatus.soundRate * ...
   MakeQTMovieStatus.timeScale;
else
 duration = MakeQTMovieStatus.frameNumber / ...
   MakeQTMovieStatus.frameRate * ...
   MakeQTMovieStatus.timeScale;
end
```

```
duration = ceil(duration);

y = [mb32(0) mbstring('tkhd') ... % Atom Header
     mb32(15) ...   % Version and flags
     mb32(round(now*3600*24)) ... % Creation time
     mb32(round(now*3600*24)) ... % Modification time
     mb32(MakeQTMovieStatus.trackNumber) ...
     mb32(0) ...
     mb32(duration) ...   % Track duration
     mb32(0) mb32(0) ...  % Offset and priority
     mb16(0) mb16(0) mb16(255) mb16(0) ... % Layer, Group, Volume,
 fill
     mb32(fixed1) mb32(0) mb32(0) ... % Transformation matrix
 (identity)
     mb32(0) mb32(fixed1) mb32(0) ...
     mb32(0) mb32(0) mb32(frac1)];

if add_sound_p
 y = [y mb32(0) mb32(0)]; % Zeros for sound
else
 y = [y mb32(fliplr(MakeQTMovieStatus.imageSize)*fixed1)];
end
y= SetAtomSize(y);

MakeQTMovieStatus.trackNumber = MakeQTMovieStatus.trackNumber + 1;

%%%%%%%%%%%%%%%  udat_atom %%%%%%%%%%%%%%%%%
function y = udat_atom()
atfmt = [64 double('fmt')];
atday = [64 double('day')];

VersionString = 'Matlab MakeQTMovie version April 7, 2000';

y = [mb32(0) mbstring('udta') ...
 SetAtomSize([mb32(0) atfmt mbstring(['Created ' VersionString])]) ...
 SetAtomSize([mb32(0) atday '  ' date])];
y = SetAtomSize(y);


%%%%%%%%%%%%%%%  vmhd_atom %%%%%%%%%%%%%%%%%
function y = vmhd_atom()

y = SetAtomSize([mb32(0) mbstring('vmhd') mb32(0) ...
    mb16(64) ...   % Graphics Mode
    mb16(0) mb16(0) mb16(0)]);  % Op Color

Syntax: MakeQTMovie cmd [arg]
The following commands are supported:
 addfigure - Add snapshot of current figure to movie
 addaxes - Add snapshot of current axes to movie
 addmatrix data - Add a matrix to movie (convert to jpeg)
 addmatrixsc data - Add a matrix to movie (scale and convert to jpeg)
 addsound data - Add sound samples (with optional rate)
 demo - Show this program in action
```

```
finish - Finish movie, write out QT file
framerate # - Set movie frame rate (default is 10fps)
quality # - Set JPEG quality (between 0 and 1)
size [# #] - Set plot size to [width height]
start filename - Start making a movie with this name
```

Published with MATLAB® R2018a