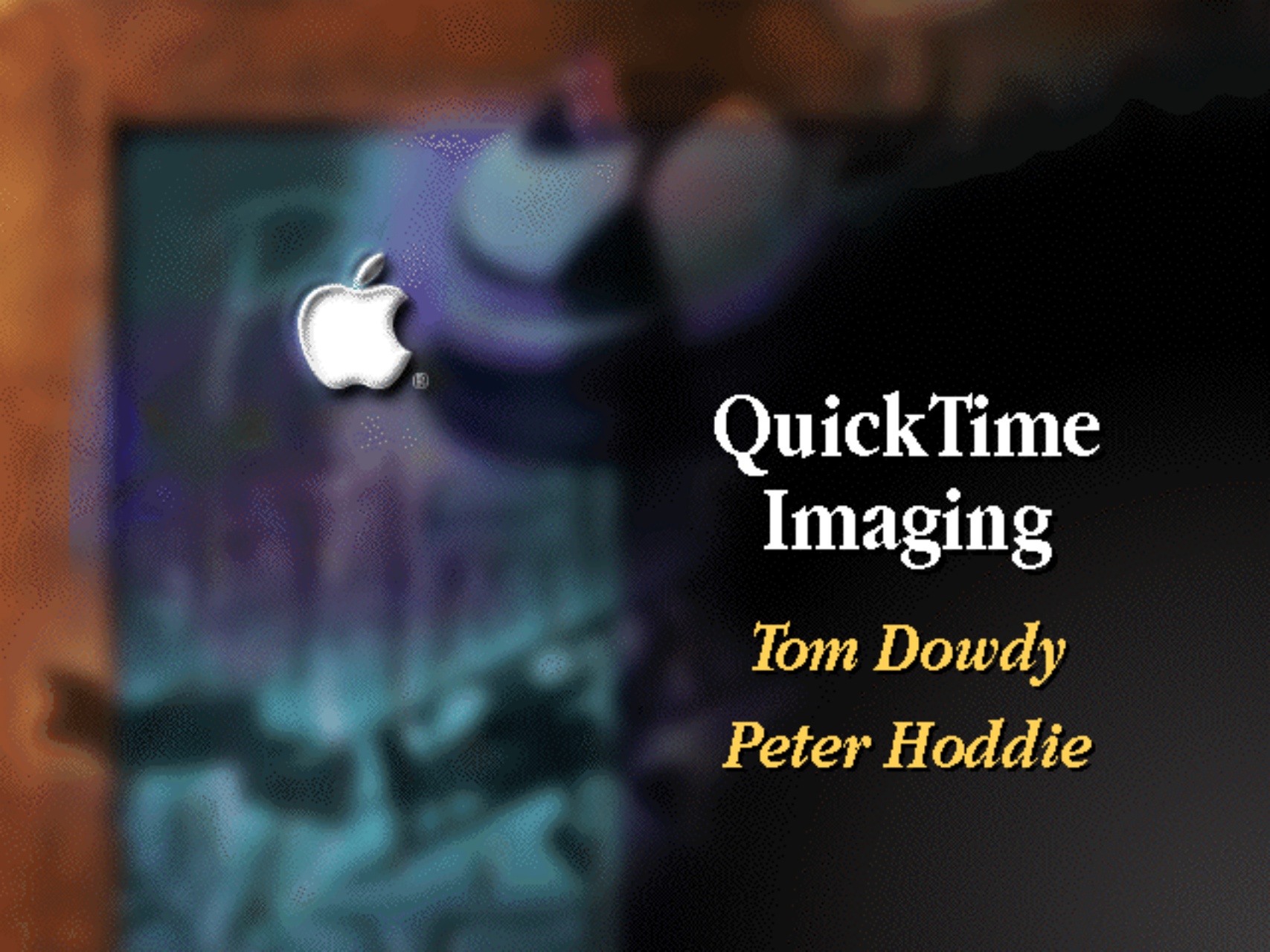

Worldwide
Developers
Conference


QuickTime
Imaging
Tom Dowdy
Peter Hoddie

# Introduction

- Nearly everything drawn by QuickTime is imaged by an image decompressor component (codec)
- Codecs should be considered general purpose blitters, not just decompressors
- Used for many purposes
  - Still images, movies, sprites, renderers, special effects, transitions

# Still Images

- Built-in services for working with image files
- Supports most standard image formats
  - GIF, JPEG/JFIF, Targa, BMP, Photoshop, QuickDraw Picture, MacPaint, SGI
- Extensible
  - Flash Pix
  - PNG support by Sam Bushell


Demo

# Graphics Importer Components

- Still image file support consists of two separate pieces
- Graphics Importer
  - File parsing code for a given file format
  - Typically very simple—3k in size
- Image Decompressor
  - Implements drawing of file format
  - Complexity of implementation depends on image format

# Drawing with Graphics Importers

```
FSSpec imageFile;
GraphicsImportComponent importer;

GetGraphicsImporterForFile(&imageFile,
    &importer);
GraphicsImportSetGWorld(importer,
    myWindow, nil);
GraphicsImportSetBoundsRect(importer,
    &boundsRect);
GraphicsImportDraw(importer);
```

# Determining Image Properties

```
GetGraphicsImporterForFile(&imageFile,
    &importer);
GraphicsImportGetImageDescription
    (importer, &imageDesc);
width = (**imageDesc).width;
height = (**imageDesc).height;
depth = (**imageDesc).depth;
```

# Converting Images into Movies

```
FSSpec theFile;
Movie theMovie;
short movieFileRef;

OpenMovieFile(&theFile, &movieFileRef,
    fsRdPerm);
NewMovieFromFile(&theMovie, movieFileRef,
    nil, nil, newMovieActive, nil);
CloseMovieFile(movieFileRef);
```

# Drawing Compressed Images…

```
ImageDescriptionHandle desc;

desc = NewHandleClear
    (sizeof(ImageDescription);
(**desc).idSize =
    sizeof(ImageDescription);
(**desc).cType = kJPEGCodecType;
(**desc).width = 160;
(**desc).height = 120;
(**desc).hRes = 72L<<16;
(**desc).vRes = 72L<<16;
(**desc).depth = 24;
(**desc).frameCount = 1;
(**desc).clutID = -1;
DecompressImage(jpegDataPtr, desc,
    GetGWorldPixMap(qd.thePort),
    nil, nil, ditherCopy, nil);
```

# Matrices

- QuickTime has always used matrices to describe track location and scaling
  - 3 by 3 fixed point
  - Scale, translate
- QuickTime 3.0 supports all matrix transformations
  - Rotate, skew
  - Perspective


Demo

# Making a Movie Rotate

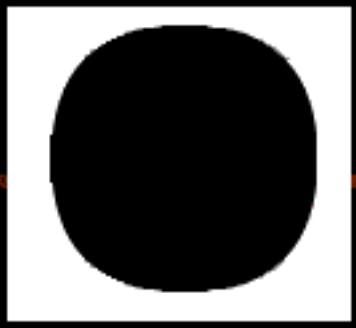
```
MatrixRecord  matrix;
Track theTrack;

GetTrackMatrix(theTrack, &matrix);

RotateMatrix(&matrix,
    IntToFixed(30), 0, 0 );

SetTrackMatrix(theTrack, &matrix);
```

# Curves

- Adds the ability to render vector graphics
  - Small data size
  - Scalable
  - High quality
- Features
  - Quadratic bezier
  - Linear and radial color gradients
  - Anti-aliasing
  - Standard framing and filling types
  - Control over joins, caps, and miter limits


Demo

# Creating a Curve



```
// a rounded square
Fixed pathData[] =
sizeof(long)*5 + sizeof(FixedPoint) * 4,
'path',

1, // one outline
4, // 4 points
0xFFFFFFFF, // all off the curve

rIntToFix(100), rIntToFix(100),
rIntToFix(200), rIntToFix(100),
rIntToFix(200), rIntToFix(200),
rIntToFix(100), rIntToFix(200),

// end of path data
sizeof(long)*2, 'zero'};
```

# Drawing a Curve

```
(**pathDesc).cType = 'path';

DecompressImage(
    pathData, pathDesc,
    GetGWorldPixMap(thePort),
    nil, nil, ditherCopy, nil);
```

# Drawing a Rotated Curve

```
MatrixRecord matrix;

SetIdentityMatrix(&matrix);

RotateMatrix(&matrix,
    IntToFixed(30), 0, 0);

FDecompressImage(
    pathData, pathDesc,
    GetGWorldPixMap(thePort),
    nil, nil,
    &matrix, ditherCopy, nil,
    nil, nil, nil,
    anyCodec, nil,
    0, nil, nil);
```

# Other Curve Atoms

| | |
|---|---|
| 'argb' | ARGBColor |
| 'fill' | fill type |
| 'pent' | Fixed pen thickness |
| 'mitr' | Fixed mitre limit |
| 'join' | join options |
| 'cap ' | cap options |
| 'anti' | anti aliasing control |
| 'grad' | gradient colors |
| 'grdt' | gradient type |
| 'angl' | gradient angle |
| 'radi' | gradient radius |
| 'cent' | gradient center point |

# Importing from GX

```
(**idh).idSize = sizeof(ImageDescription);
(**idh).cType = 'qdgx';

ImageTranscodeSequenceBegin(&ts, idh,
        'path', &pathDesc, nil, 0);

HLock(shapeData);
ImageTranscodeFrame(ts, *flatShapeData,
      GetHandleSize(flatShapeData),
      &pathData, &pathDataSize);
HUnlock(shapeData);

// pathData is now a Vector shape

ImageTranscodeDisposeFrameData(ts,
   pathData);
ImageTranscodeSequenceEnd(ts);
DisposeHandle((Handle) idh);
```

# Renderers/Effects

- Renders generate images from parameters, not image data
- Can be used to enhance or modify other graphical elements
- Useful for synthesizing certain kinds of images
- Ideal for internet delivery, because of extremely small data size


Demo

# Drawing Fire or Clouds…

```
(**pathDesc).cType = 'fire';
      OR
(**pathDesc).cType = 'clou';

while (true) {
    DecompressImage(
        nil, pathDesc,
        GetGWorldPixMap(thePort),
        nil, nil, ditherCopy, nil);
}
```

# Transitions

- Another kind of decompressor
- Takes two or more images and combines them
- Changes over time


Demo

# Specifying Transitions

- QT Atom Container
- Built-in implementations of standard effects
- Industry-wide effort to standardize descriptions of common effects
- Well defined mechanisms for extending existing transitions and adding custom transitions
- Supports hardware acceleration

# QuickTime Imaging

- It's all about codecs…


Worldwide
Developers
Conference