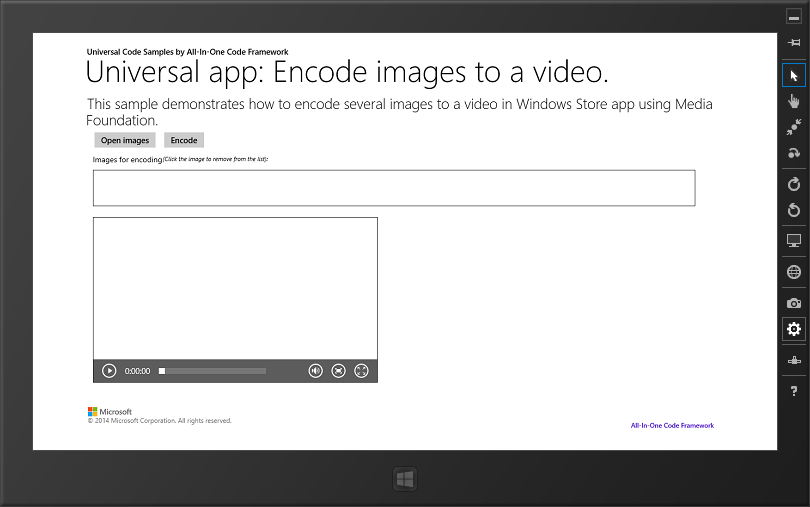
# How to encode several images to a video in Universal apps using Media Foundation

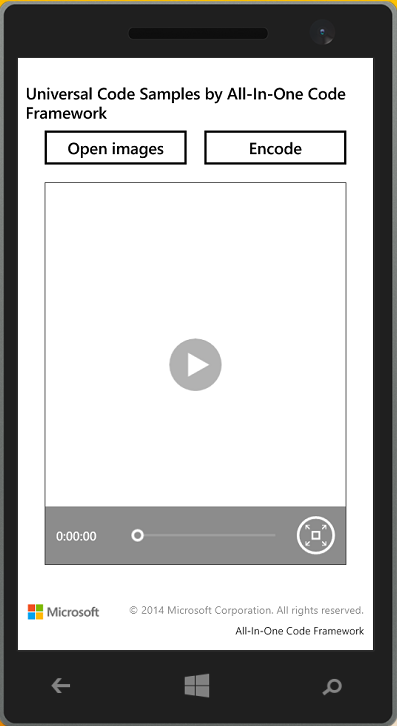
## Introduction

This sample demonstrates how to encode several images to an mp4 video using Media Foundation in a C++/CX component. This is a Universal app which can be built for both Windows 8 and Windows Phone.

## Running the Sample

Build the sample in Visual Studio 2013 and then run it. Click the "Create new video" button to create a video file which will be processed. Click the "Open images" button to pick several images which will be shown below the button in a GridView control. Then click the "Encode" button to create a video file which will be processed and encode the video with the images as frames. The video will play below the GridView in a MediaElement control.





## Using the Code

We implement the basic capabilities in PictureWriter calss in EncodeImage namespace using Media Foundation.

In the code behind MainPage.xaml, we first open images and store the files to List<StorageFile> object.

|  |
| --- |
| -Code block start-  --C# code snippet start--  private async void ImageBtn\_Click(object sender, RoutedEventArgs e)  {  if (m\_images.Count != 0)  {  m\_images.Clear();  }  if (m\_files.Count != 0)  {  m\_files.Clear();  }  statusText.Text = "";  FileOpenPicker openPicker = new FileOpenPicker();  openPicker.SuggestedStartLocation = PickerLocationId.PicturesLibrary;  openPicker.ViewMode = PickerViewMode.Thumbnail;  openPicker.FileTypeFilter.Add(".jpg");  openPicker.FileTypeFilter.Add(".png");  openPicker.FileTypeFilter.Add(".bmp");  IReadOnlyList<StorageFile> files = await openPicker.PickMultipleFilesAsync();  if (files.Count > 0)  {  foreach(StorageFile file in files)  {  m\_files.Add(file);  using( IRandomAccessStream stream = await file.OpenAsync(FileAccessMode.Read))  {  BitmapImage bitmapImage = new BitmapImage();  await bitmapImage.SetSourceAsync(stream);  Image image = new Image();  image.Source = bitmapImage;  m\_images.Add(image);  }    }  ImageGV.DataContext = m\_images;  }  }  --C# code snippet end--  --C++ code snippet start--  void CppUniversalAppImageToVideo::MainPage::ImageBtn\_Click(Platform::Object^ sender, Windows::UI::Xaml::RoutedEventArgs^ e)  {  statusText->Text = "";  if (m\_images->Size != 0)  {  m\_images->Clear();  }  if (m\_files->Size != 0)  {  m\_files->Clear();  }  // Open images.  FileOpenPicker^ picker = ref new FileOpenPicker;  picker->SuggestedStartLocation = PickerLocationId::PicturesLibrary;  picker->ViewMode = PickerViewMode::Thumbnail;  picker->FileTypeFilter->Append(".jpg");  picker->FileTypeFilter->Append(".png");  picker->FileTypeFilter->Append(".bmp");  create\_task(picker->PickMultipleFilesAsync()).then([=](IVectorView<StorageFile^>^ files){  if (files->Size == 0)  {  cancel\_current\_task();  }    auto images = std::make\_shared<Platform::Collections::Vector<Windows::UI::Xaml::Controls::Image^>^>(m\_images);  for (StorageFile^ file : files)  {  create\_task(file->OpenAsync(FileAccessMode::Read)).then([=](Streams::IRandomAccessStream^ stream){    auto bitmapImage = ref new BitmapImage();  bitmapImage->SetSource(stream);  Image^ xamlImage = ref new Image;  xamlImage->Source = bitmapImage;  m\_images->Append(xamlImage);  }).then([=](){  m\_files->Append(file);  }, task\_continuation\_context::use\_arbitrary()).then([=](task<void> t){  try  {  t.get();  }  catch (InvalidArgumentException^ e)  {  statusText->Text = "Some errors occur when openning, please try again";  m\_images->Clear();  m\_files->Clear();  }  });  }  });  }  --C++ code snippet end--  --JavaScript code snippet start--  function openImages()  {  if (items.length != 0)  {  items.splice(0, items.length);  }  if (g\_imageFiles.length != 0)  {  g\_imageFiles.splice(0, g\_imageFiles.length);  }  var openPicker = new Windows.Storage.Pickers.FileOpenPicker();  openPicker.viewMode = Windows.Storage.Pickers.PickerViewMode.thumbnail;  openPicker.suggestedStartLocation = Windows.Storage.Pickers.PickerLocationId.picturesLibrary;  openPicker.fileTypeFilter.append(".jpg");  openPicker.fileTypeFilter.append(".png");  openPicker.fileTypeFilter.append(".bmp");  openPicker.pickMultipleFilesAsync().then(function (files) {  if(files.size > 0)  {  files.forEach(function (file) {  g\_imageFiles.push(file);  file.openAsync(Windows.Storage.FileAccessMode.read).then(function (stream) {  items.push({ picture: URL.createObjectURL(stream) });  });  });  }  });  }  --JavaScript code snippet end--  -Code block end- |

Then we initialize the the PictureWriter object after creating the new video file.

|  |
| --- |
| -Code block start-  --C# code snippet start--  private async void EncodeBtn\_Click(object sender, RoutedEventArgs e)  {  if (m\_files.Count == 0)  {  statusText.Text = "You must select one image at least.";  return;  }  // Create the video file via file picker.  FileSavePicker savePicker = new FileSavePicker();  savePicker.SuggestedStartLocation = PickerLocationId.VideosLibrary;  savePicker.FileTypeChoices.Add("MP4 File", new List<string>() { ".mp4" });  savePicker.SuggestedFileName = "output";  StorageFile videoFile = await savePicker.PickSaveFileAsync();  if(videoFile != null)  {  IRandomAccessStream videoStream = await videoFile.OpenAsync(FileAccessMode.ReadWrite);  m\_picture = new PictureWriter(videoStream, m\_videoWidth, m\_videoHeight);  // Add frames to the video.  ProcessVideoRing.IsActive = true;  statusText.Text = "Encoding...";  foreach (StorageFile file in m\_files)  {  Windows.Storage.FileProperties.ImageProperties properties = await file.Properties.GetImagePropertiesAsync();  float scaleOfWidth = (float)m\_videoWidth / properties.Width;  float scaleOfHeight = (float)m\_videoHeight / properties.Height;  float scale = scaleOfHeight > scaleOfWidth ?  scaleOfWidth : scaleOfHeight;  uint width = (uint)(properties.Width \* scale);  uint height = (uint)(properties.Height \* scale);  using (IRandomAccessStream stream = await file.OpenAsync(FileAccessMode.Read))  {  for (int i = 0; i < 10; ++i)  {  BitmapDecoder decoder = await BitmapDecoder.CreateAsync(stream);  PixelDataProvider dataProvider = await decoder.GetPixelDataAsync(  BitmapPixelFormat.Bgra8,  BitmapAlphaMode.Straight,  new BitmapTransform { ScaledWidth = width, ScaledHeight = height },  ExifOrientationMode.RespectExifOrientation,  ColorManagementMode.ColorManageToSRgb);  m\_picture.AddFrame(dataProvider.DetachPixelData(), (int)width, (int)height);  }  }  }  m\_picture.Finalize();  m\_picture = null;  statusText.Text = "The image files are encoded successfully. You can review the video.";  ProcessVideoRing.IsActive = false;  videoStream.Dispose();  videoStream = null;  videoStream = await videoFile.OpenAsync(FileAccessMode.Read);  VideoElement.SetSource(videoStream, videoFile.ContentType);  }  }  --C# code snippet end--  --C++ code snippet start--  void CppUniversalAppImageToVideo::MainPage::EncodeBtn\_Click(Platform::Object^ sender, Windows::UI::Xaml::RoutedEventArgs^ e)  {  if (m\_files->Size == 0)  {  statusText->Text = "You must select one image at least.";  return;  }  // Create the video via file picker.  statusText->Text = "";  FileSavePicker^ picker = ref new FileSavePicker;  picker->SuggestedStartLocation = PickerLocationId::VideosLibrary;  auto mp4Extensions = ref new Platform::Collections::Vector<Platform::String^>();  mp4Extensions->Append(".mp4");  picker->FileTypeChoices->Insert("MP4 file", mp4Extensions);  picker->DefaultFileExtension = ".mp4";  picker->SuggestedFileName = "output";  picker->SuggestedStartLocation = PickerLocationId::VideosLibrary;    create\_task( picker->PickSaveFileAsync())  .then([=](StorageFile^ file){  if (nullptr == file)  {  cancel\_current\_task();  }  m\_videoFile = file;  return file->OpenAsync(FileAccessMode::ReadWrite);  }).then([=](Streams::IRandomAccessStream^ stream){  m\_picture = ref new PictureWriter(stream, m\_videoWidth, m\_videoHeight);  }).then([this](){  // Add frames to the video.  ProcessVideoRing->IsActive = true;  statusText->Text = "Encoding...";  static int imageWidth, imageHeight, width, height;    create\_task([=](){  for (StorageFile^ file : m\_files)  {  // We set 10 FPS default in the PictureWriter, so we add 10 same frames with each image.  for (int i = 0; i < 10; ++i)  {  create\_task(file->Properties->GetImagePropertiesAsync()).then([&](FileProperties::ImageProperties^ properties){  imageWidth = properties->Width;  imageHeight = properties->Height;  return file->OpenAsync(FileAccessMode::Read);  }).then([=](Streams::IRandomAccessStream^ stream){  return BitmapDecoder::CreateAsync(stream);  }).then([&](BitmapDecoder^ decoder){  float scaleOfWidth = static\_cast<float>(m\_videoWidth) / imageWidth;  float scaleOfHeight = static\_cast<float>(m\_videoHeight) / imageHeight;  float scale = scaleOfHeight > scaleOfWidth ?  scaleOfWidth : scaleOfHeight;  width = static\_cast<int>(imageWidth \* scale);  height = static\_cast<int>(imageHeight \* scale);    BitmapTransform^ transform = ref new BitmapTransform;  transform->ScaledWidth = width;  transform->ScaledHeight = height;  return decoder->GetPixelDataAsync(BitmapPixelFormat::Bgra8,  BitmapAlphaMode::Straight,  transform,  ExifOrientationMode::RespectExifOrientation,  ColorManagementMode::ColorManageToSRgb);  }).then([&](PixelDataProvider^ provider){  m\_picture->AddFrame(provider->DetachPixelData(), width, height);  }).wait();  }  }  }).then([=](){  m\_picture->Finalize();  m\_picture = nullptr;  }).then([=](){  return m\_videoFile->OpenAsync(FileAccessMode::Read);  }).then([=](Streams::IRandomAccessStream^ stream){  VideoElement->SetSource(stream, nullptr);    ProcessVideoRing->IsActive = false;  statusText->Text = "The image files are encoded successfully.";  });  });  }  --C++ code snippet end--  --JavaScript code snippet start--  function encode()  {  if (g\_imageFiles.length == 0)  {  displayInfo("You must select one image at least.");  return;  }  var savePicker = new Windows.Storage.Pickers.FileSavePicker();  savePicker.suggestedStartLocation = Windows.Storage.Pickers.PickerLocationId.videosLibrary;  savePicker.suggestedFileName = "output";  savePicker.defaultFileExtension = ".mp4";  savePicker.fileTypeChoices.insert("MP4 file", [".mp4"]);  var promise = savePicker.pickSaveFileAsync().then(function (videoFile) {  if (videoFile) {  g\_videoFile = videoFile;  return videoFile.openAsync(Windows.Storage.FileAccessMode.readWrite);  }  }).then(function (stream) {  if (stream)  {  g\_picture = new EncodeImages.PictureWriter(stream, g\_videoWidth, g\_videoHeight);  }  });  var imageWidth, imageHeight;  var promiseArray = g\_imageFiles.map(function (file) {  promise = promise.then(function () {    return file.properties.getImagePropertiesAsync().then(function (props) {  if (g\_picture) {  imageWidth = props.width;  imageHeight = props.height;  return file.openAsync(Windows.Storage.FileAccessMode.read);  }    }).then(function (stream) {  if (stream)  {  return Windows.Graphics.Imaging.BitmapDecoder.createAsync(stream);  }    }).then(function (decoder) {  if (decoder)  {  // Transform the image size.  var scaleOfWidth = g\_videoWidth / imageWidth;  var scaleOfHeight = g\_videoHeight / imageHeight;  var scale = scaleOfWidth > scaleOfHeight ? scaleOfHeight : scaleOfWidth;  imageWidth \*= scale;  imageHeight \*= scale;  var transform = new Windows.Graphics.Imaging.BitmapTransform();  transform.scaledWidth = imageWidth;  transform.scaledHeight = imageHeight;  return decoder.getPixelDataAsync(  Windows.Graphics.Imaging.BitmapPixelFormat.bgra8,  Windows.Graphics.Imaging.BitmapAlphaMode.straight,  transform,  Windows.Graphics.Imaging.ExifOrientationMode.respectExifOrientation,  Windows.Graphics.Imaging.ColorManagementMode.colorManageToSRgb);  }    }).then(function (provider) {  if (provider)  {  var data = provider.detachPixelData();  for (var i = 0; i < 10; ++i) {  g\_picture.addFrame(data, imageWidth, imageHeight);  }  }  });  });  return promise;    });  WinJS.Promise.join(promiseArray).then(function () {  if (g\_picture)  {  g\_picture.finalize();  g\_picture = null;  displayInfo("The image files are encoded successfully.");  var myVideo = document.getElementById("videoElement");  myVideo.src = URL.createObjectURL(g\_videoFile);  }    });  }  --JavaScript code snippet end--  -Code block end- |

## More Information

[Developing a WinRT component to create a video file using Media Foundation](http://blogs.msdn.com/b/eternalcoding/archive/2013/03/06/developing-a-winrt-component-to-create-a-video-file-using-media-foundation.aspx)