Dlib FaceLandmark Detector 2.0.0

iOS & **Android** & **Windows10 UWP** support
Win & **Mac** & **Linux** Standalone support
WebGL support
ChromeOS support
visionOS support(beta)
Support for preview in the **Editor**

System Requirements
Build Win Standalone & Preview Editor : Windows8 or later
Build Mac Standalone & Preview Editor : macOS 10.13 or later
Build Linux Standalone & Preview Editor : Ubuntu18.04 or later
Build Android : API level 21 or later
Build iOS : iOS Version 12.0 or later
Build VisionOS : visionOS 1 or later (beta)

DlibFaceLandmarkDetector can **Object Detection** and **Shape Prediction** using Dlib19.7 C++ Library.

Features :

- You can detect **frontal human faces and face landmark (68 points, 17points, 6points)** in **Texture2D**, **WebCamTexture** and **Image byte array**. In addition, you can detect a different objects by changing trained data file.
- **Object Detector** is made using the now classic Histogram of Oriented Gradients (HOG) feature combined with a linear classifier, an image pyramid, and sliding window detection scheme. You can train your own detector in addition to human faces detector. If you want to train your own detector, please refer to this page.
- **Shape Predictor** is created by using dlib's implementation of the paper (One Millisecond Face Alignment with an Ensemble of Regression Trees by Vahid Kazemi and Josephine Sullivan, CVPR 2014). You can train your own models in addition to human face landmark model using dlib's machine learning tools. If you want to train your own models, please refer to this page.
- Advanced examples using "**OpenCV for Unity**" are Included. (The execution of this examples are required "OpenCV for Unity".)
- By utilizing the **VisualScripting With DlibFaceLandmarkDetector Example**, you can leverage all the methods available in DlibFaceLandmarkDetector within the **Unity's Visual Scripting development environment**.
  VisualScripting With DlibFaceLandmarkDetector Example (GitHub)

Basic Examples :

- Texture2D Example
- WebCamTexture Example
- Benchmark Example

Advanced Examples (require OpenCV for Unity) :

- Texture2DToMat Example
- WebCamTexture2MatHelper Example
- VideoCapture2MatHelper Example
- AR Head WebCamTexture Example
- AR Head VideoCapture Example
- Frame Optimization Example
- NoiseFilter WebCamTexture Example
- NoiseFilter VideoCapture Example

Official Site | ExampleCode | Android Demo | WebGL Demo | Setup Tutorial & Demo Video | Forum | API Reference

DlibFaceLandmarkDetector uses Dlib under Boost Software License ; see Third-Party Notices.txt file in package for details.
The Shape Predictor model files included with this asset are available for commercial use.

Version changes :

2.0.0 [Common]Updated FpsMonitor to version 1.0.4. [visionOS]Updated beta support for the VisionOS platform on Unity 2022.3.18f1 and later. [Android]Updated native library libdlibfacelandmarkdetector.so for Android to be compatible with 16KB page size. [Common]Removed CatDetectionExample. [Common]Changed namespace of the Utils class from UnityUtils to UnityIntegration, split into function-specific classes, and marked old classes as deprecated. [Common]Changed moved the OpenCVForUnityUtils class bundled with DlibFaceLandmarkDetectorWithOpenCVExample to the main Dlib module as DlibOpenCVUtils. [Common]Changed return types of DetectValueTuple and DetectLandmark to List. [Common]Changed removed the Extra folder. [Common]Changed increased the resolution of video files used for examples. [Common]Added overloads of GetDetectResult and GetDetectLandmarkResult methods that accept Span<double>. [Common]Added overloads of the DrawFaceLandmark method that support ReadOnlySpan<Vector2>, ReadOnlySpan<Vec2d>, and ReadOnlySpan<double>. [Common]Added IsDebugMode and IsThrowException methods to DlibDebug. [Common]Added GetFilePathCoroutine, GetFilePathAsync, and GetFilePathTaskAsync methods to DlibEnv.

1.4.2 [Common]Changed The FpsMonitor display system from IMGUI to uGUI. [Common]Added support for the new Input System (UnityEngine.InputSystem).

1.4.1 [Common]Added method overloads in DlibFaceLandmarkDetector class that use ValueTuples. [Common]Add method overloads in OpenCVForUnityUtils class that use structs and ValueTuples corresponding to the data formats of OpenCV base classes. Accordingly, the minimum required version of OpenCVForUnity is now 2.6.4. [Common]Added Utils.getFilePathAsyncTask() and Utils.getMultipleFilePathsAsyncTask() methods. It is an asynchronous Task return version of the existing Utils.getFilePathsAsync() method. It can be seamlessly integrated with other asynchronous code. [iOS]Fixed an issue where AddToEmbeddedBinaries was processed redundantly with each incremental build.

1.4.0 [Common]Changed the minimum supported version to Unity2021.3.35f1. [Common]Separated the examples using the Built-in Render Pipeline and Scriptable Render Pipeline.

1.3.9 [iOS]Added separate plugin files for iOS for devices and simulators. [WebGL]Added plugin files with only simd enabled.

1.3.8 [Common]Changed to use unsafe code by default. [Common]Optimized the amount of memory allocation, in the FaceLandmarkDetector class.

1.3.7 [Common]Changed the minimum supported version to Unity2020.3.48f1. [WebGL]Added support for "WebAssembly 2023". [iOS]Changed “Target minimum iOS Version” to 11.0.

1.3.6 [WebGL]Added a plugin file with threads and simd enabled for the WebGL platform. This update removes support for the WebGL platform in Unity 2021.1 and below. (Select MenuItem[Tools/Dlib FaceLandmark Detector/Open Setup Tools/WebGL Settings])

1.3.5 [Windows]Added Support for ARM64. [WebGL]Added Unity2023.2 or later support. [Lumin]Removed Lumin platform support (for MagicLeapOne). [Common]Add a button to SetupTools to automatically add scenes under the "Examples" folder to "Scenes In Build".

1.3.4 [Common]Changed the setup procedure to use the SetupToolsWindow. [Common]Change the namespase under "DlibFaceLandmarkDetector/Editor" folder from "DlibFaceLandmarkDetector" to "DlibFaceLandmarkDetector.Editor". [Common]Added "DlibFaceLandmarkDetector" folder under "StreamingAssets" folder. [Common]Added function to automatically move the StreamingAssets folder.

1.3.3 [Android]Added Support for ChromeOS (x86 and x86_64 architectures).

1.3.2 [Common]Added Assembly Definitions.
1.3.1 [Common]Fixed a small issue.
1.3.0 [UWP]Added ARM64 Architecture.
1.2.9 [Common]Added optimization code using NativeArray class. (require PlayerSettings.allowUnsafeCode flag, "DLIB_USE_UNSAFE_CODE" ScriptingDefineSymbol and Unity2018.2 or later.) [Common]Added support for Unicode file path (objectDetectorFilePath and shapePredictorFilePath). [Common]Added ImageOptimizationHelper to ARHeadWebCamTextureExample. [Common]Added some converter methods to OpenCVForUnityUtils.cs.
1.2.8 [Lumin]Added the code for MagicLeap.
1.2.7 [WebGL]Added Unity2019.1 or later support.
1.2.6 [Common]Added "sp_human_face_17.dat", "sp_human_face_17_mobile.dat" and "sp_human_face_6.dat". [Common]Changed the training dataset of Shape Predictor model. Since the training dataset consists of flickr CC0 licensed images, the Shape Predictor model files are available for commercial use. [Common]Added BenchmarkExample.
1.2.5 [Common]Re-assined namespace. [Common]Support for OpenCVforUnity2.3.3 or later.
1.2.4 [macOS]Removed 32bit architecture (i386) from dlibfacelandmarkdetector.bundle.
1.2.3 [Android,UWP]Fixed Utils.setDebugMode() method on the IL2CPP backend.
1.2.2 [iOS]Added a function to automatically remove the simulator architecture (i386,x86_64) at build time. [Common] Improved DlibFaceLandmarkDetectorMenuItem.setPluginImportSettings() method. [Common]Updated to WebCamTextureToMatHelper.cs v1.0.9. [Common]Added support for Utils.setDebugMode() method on all platforms.
1.2.1 [Common]Updated to WebCamTextureToMatHelper.cs v1.0.8. [Common]Updated to LowPassPointsFilter v1.0.1. Updated to KFPointsFilter v1.0.2. Updated to OFPointsFilter v1.0.2. [Common] Added updateMipmaps and makeNoLongerReadable flag to DrawDetectResult () and DrawDetectLandmarkResult() method. [Common]Fixed Utils.getFilePathAsync() method.(Changed #if UNITY_2017 && UNITY_2017_1_OR_NEWER to #if UNITY_2017_1_OR_NEWER.)
1.2.0 [Common]Updated to WebCamTextureToMatHelper.cs v1.0.7. [Common]Fixed WebCamTextureExample and OpenCVForUnityUtils. [Common]Added NoiseFilterVideoCaptureExample and NoiseFilterWebCamTextureExample. [Commo]Added useLowPassFilter option to ARHeadVideoCaptureExample and ARHeadWebCamTextureExample. [Common]Added throwException flag to Utils.setDebugMode() method. [Common]Added drawIndexNumbers flag to DrawFaceLandmark() method.
1.1.9 [Android]Added arm64-v8a Architecture.
1.1.8 [Common]Updated WebCamTextureExample. (support Portrait ScreenOrientation) [Common]Updated to WebCamTextureToMatHelper.cs v1.0.4.
1.1.7 [Common]Updated "human_face_68_sp.dat" and "human_face_68_sp_for_mobile.dat".
1.1.6 [Common]Updated to dlib19.7. [Common]Updated to WebCamTextureToMatHelper.cs v1.0.3. [Common]Updated "human_face_68_sp.dat" and "human_face_68_sp_for_mobile.dat".
1.1.5 [Common]Switched to the shape predictor file trained using new datasets.
1.1.4 [Common]Updated WebCamTextureToMatHelper.cs v1.0.2 [Common]Improved Utils.getFilePathAsync().
1.1.3 [Common]Fixed to improve the pose estimation performance. [Common] Changed DetectLandmarkArray (int left, int top, int width, int height) to DetectLandmarkArray

(double left, double top, double width, double height). [WebGL]Fixed Utils.getFilePathAsync() method.

1.1.2 [Common]Updated WebCamTextureToMatHelper.cs and OptimizationWebCamTextureToMatHelper.cs(Changed several method names.). [Common]Changed the Example name.

1.1.1 [Common]Improved Utils.getFilePath() and Utils.getFilePathAsync().

1.1.0 [Win][Mac][Linux][UWP]Added the native plugin file enabled SSE4 or AVX compiler option.

1.0.9 [WebGL]Added WebGL Plugin for Unity5.6.

1.0.8 [Common]Changed the name of asset project. ("Sample" to "Example") [Common]Fixed VideoCaptureARExample and WebCamTextureARExample.

1.0.7 [Common]Fixed WebCamTextureToMatHelper.cs. (flipVertical and flipHorizontal flag)

1.0.6 [Common]Fixed OpenCVForUnityMenuItem.cs.(No valid name for platform: 11 Error) [Common]Added OptimizationWebCamTextureToMatHelper.cs

1.0.5 [Common]Fixed WebCamTextureToMatHelper class. [Common]Added Utils.getVersion(). [Common]Fixed Utils.getFilePathAsync().

1.0.4 [Common]Updated shape_predictor_68_face_landmarks_for_mobile.dat.

1.0.3 [WebGL]Added WebGL (beta) support. (Unity5.3 or later) [Common]Fixed missing script error.(WebCamTextureToMatHelper.cs) [Common]Added shape_predictor_68_face_landmarks_for_mobile.dat.

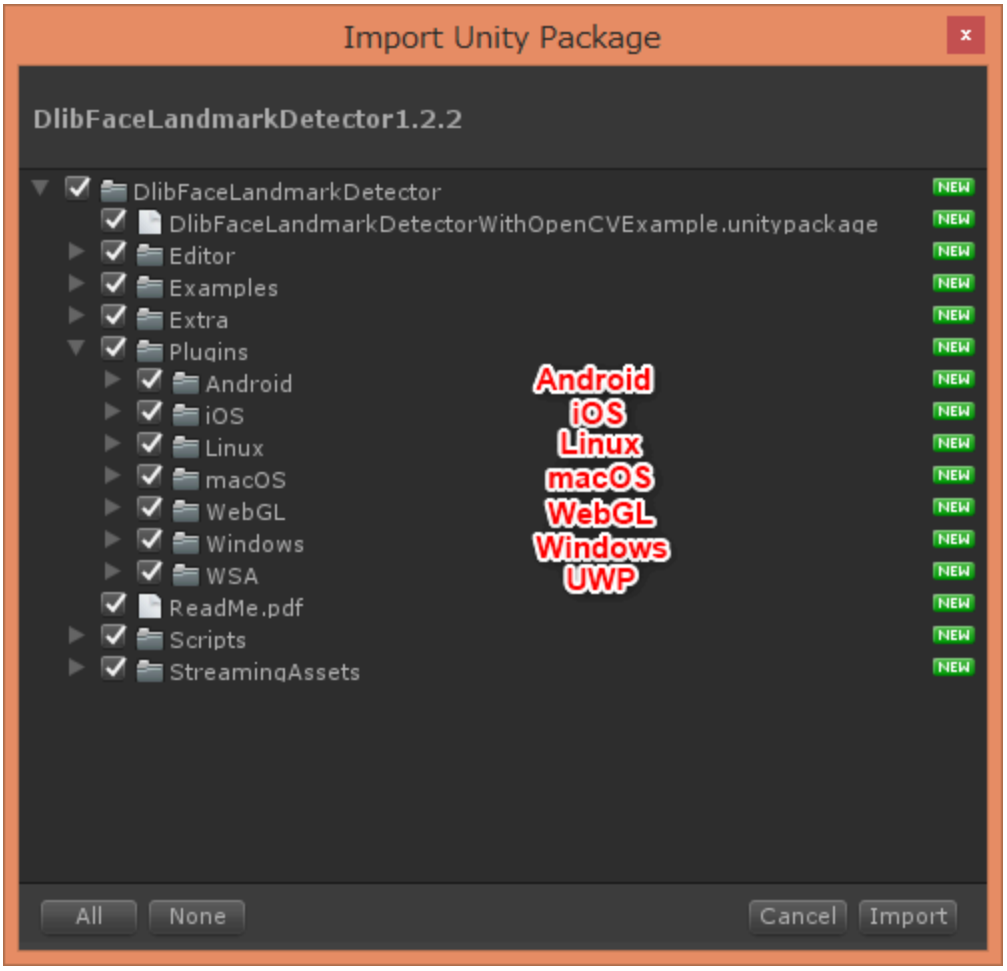
1.0.2 [Common]Improved WebCamTextureHelper class.

1.0.1 [Common]Added OptimizationSample. [Common]Added DetectRectDetection() method.

1.0.0 Initial version

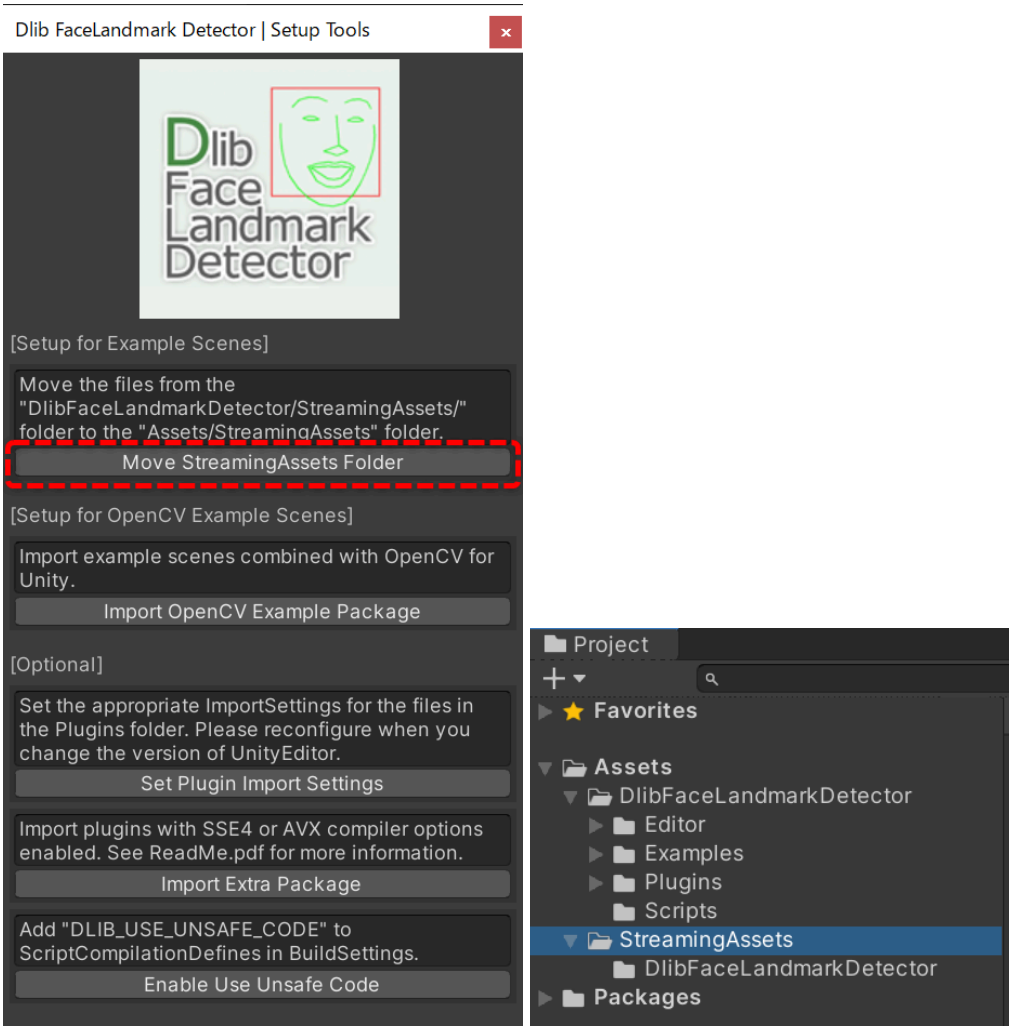
Quick setup procedure to get started with development (Setup Tutorial Video) :

1. Import the DlibFaceLandmarkDetector.package. **You do not need to import plug-in files for platforms not supported by your project.**

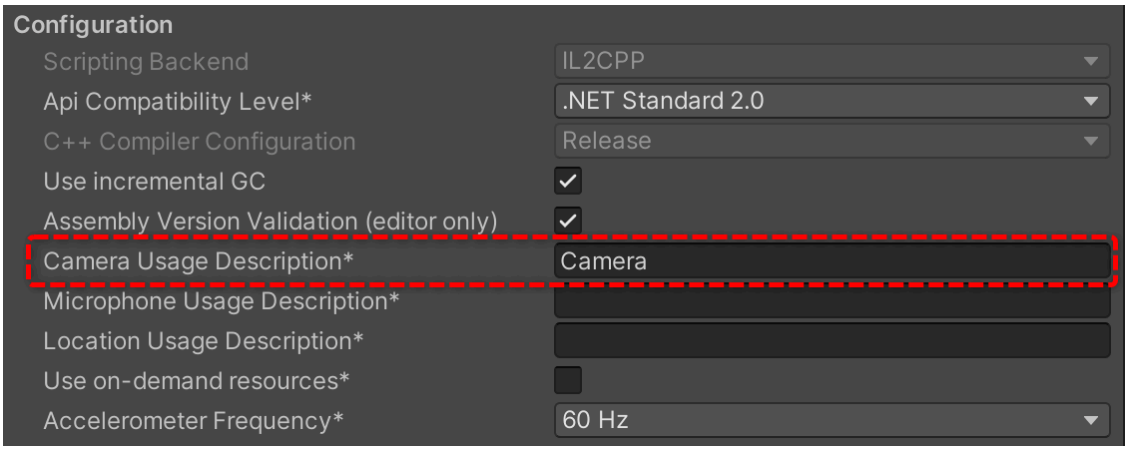


Quick setup procedure to run the example scenes (Setup Tutorial Video) :

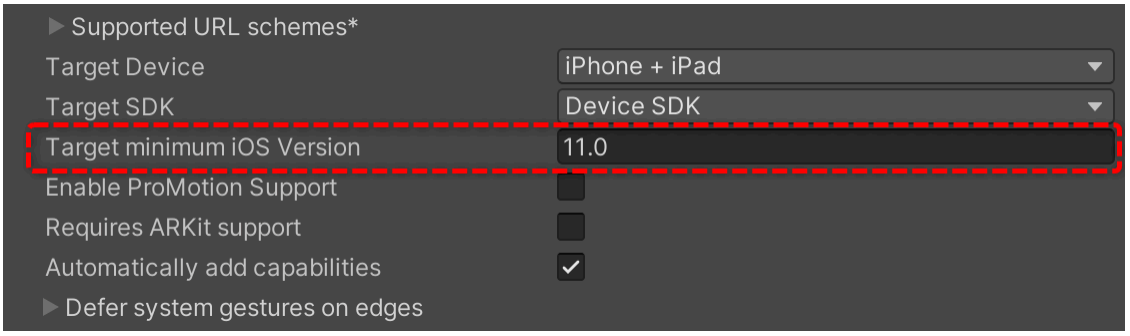
1. Import the DlibFaceLandmarkDetector.package.
2. Select MenuItem **[Tools/Dlib FaceLandmark Detector/Open Setup Tools]**.
3. Click the **[Move StreamingAssets Folder]** button.



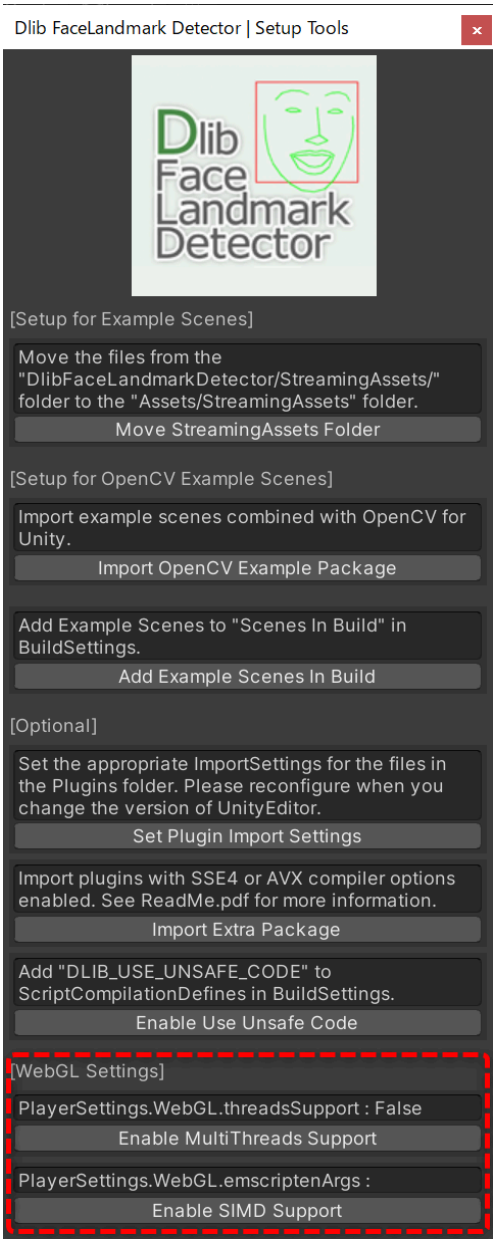
4. **[iOS]** Set [PlayerSettings]-[Other Settings]-[Configuration]-[Camera Usage Description].



Set Target minimum iOS Version to **12.0** or higher.

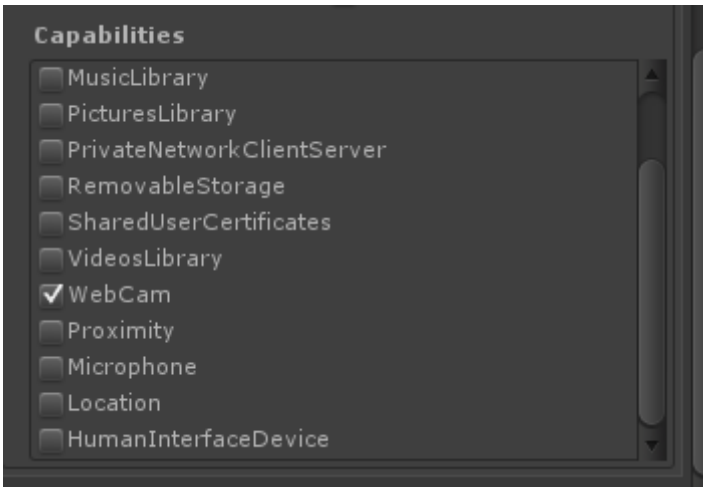


5. **[WebGL]** If you want to enable threads or simd optimization, click the **[Enable MultiThreads Support]** button or the **[Enable SIMD Support]** button. (The executing browser must support threading and SIMD.)

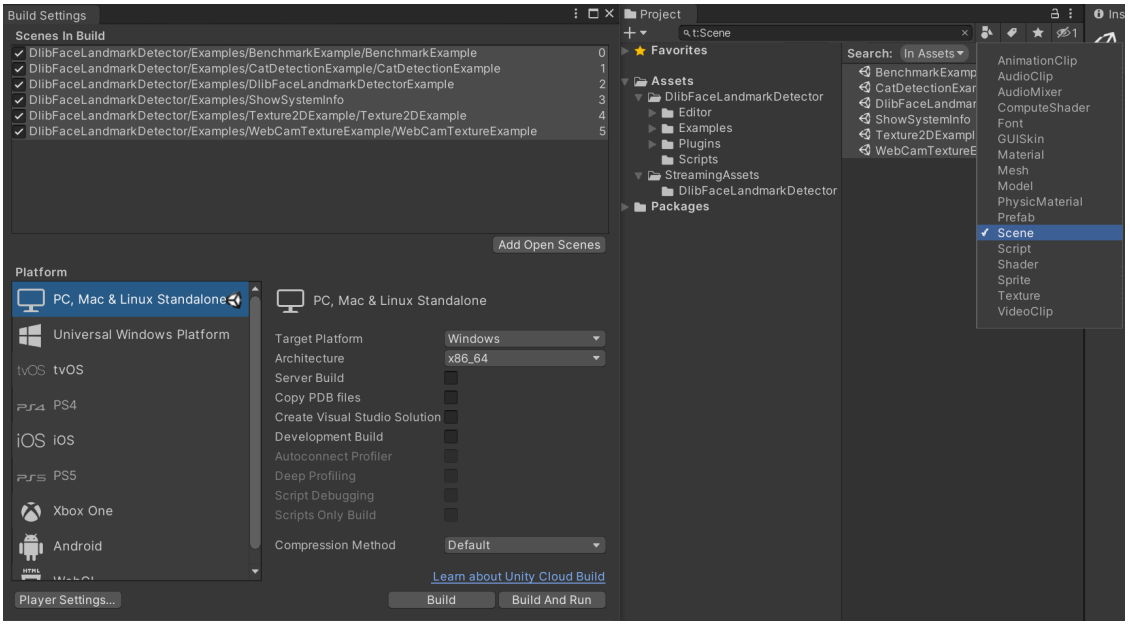


6. [**WindowsStoreApps8.1** & **WindowsPhone8.1** & **Windows10 UWP**] If use

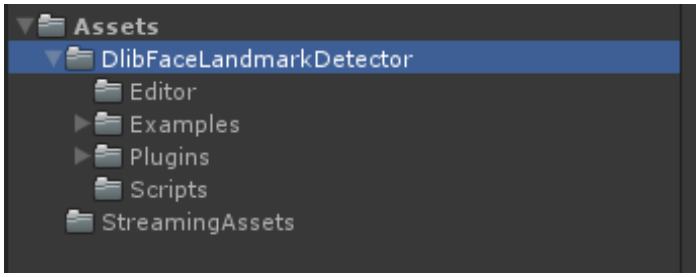
webCamTextue class, Please choose “WebCam” in [PlayerSettings]-[PublishingSettings]-[Capabilities].



7. Add all of the “***.unity” in the “DlibFaceLandmarkDetector” folder to [Build Settings] – [Scene In Build].

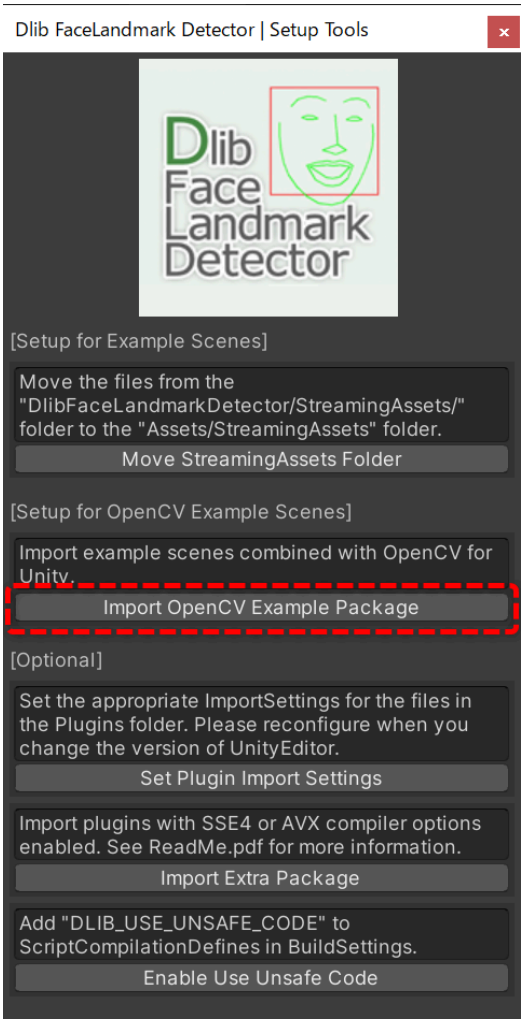


Screenshot after the setup

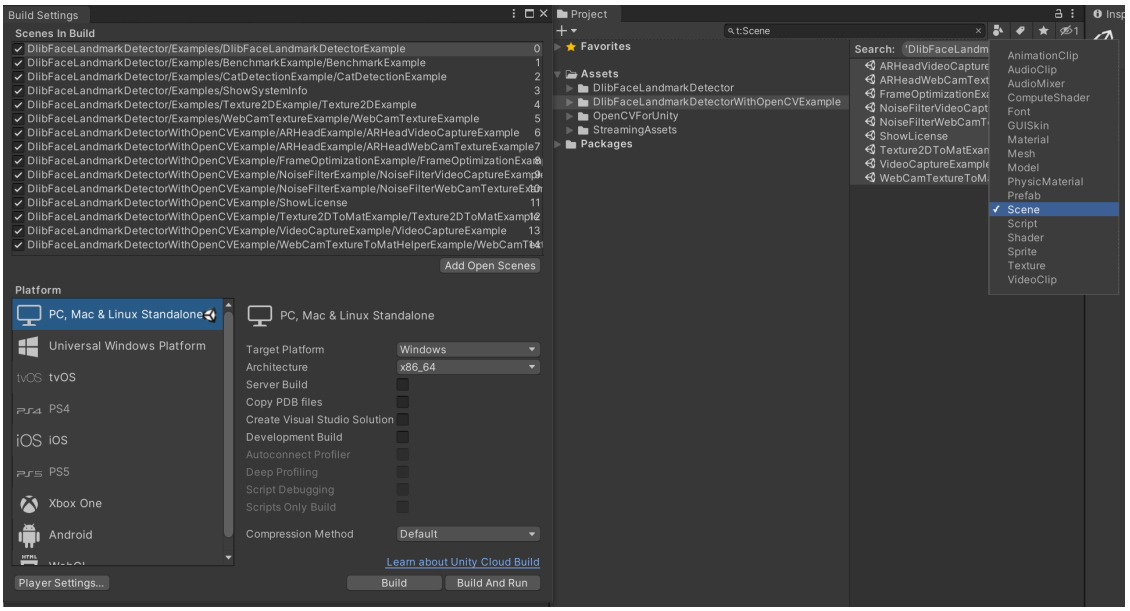


Quick setup procedure to run the Advanced examples using "OpenCV for Unity" scene :

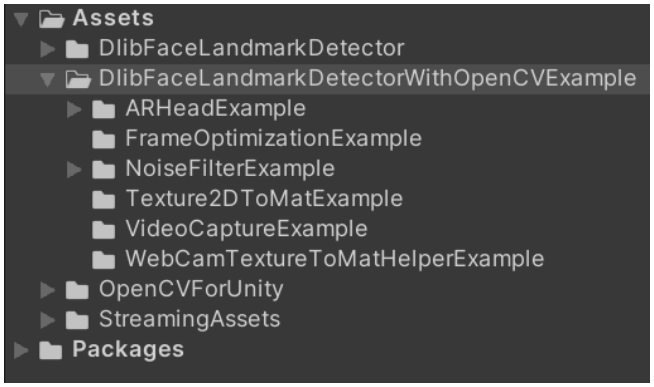
1. Import and Setup the "OpenCV for Unity".
2. Click the **[Import OpenCV Example Package]** button.



3. Add all of the “***.unity” in the “DlibFaceLandmarkDetectorWithOpenCVExample” folder to [Build Settings] – [Scene In Build].



Screenshot after the setup



Q & A

Q1.

“DllNotFoundException: dlibfacelandmarkdetector” is displayed on the console when run the example scene.

A1.

Plugin does not seem to be loaded correctly. Please check the setup procedure.

Q2.

“Level 'Texture2DExample' (-1) could not be loaded because it has not been added to the build settings.” is displayed on the console when run the example scene.

A2.

Please Add all of the “***.unity” in the “DlibFaceLandmarkDetector” folder to [Build Settings] – [Scene In Build].

Q3.

In Texture2DExample, red rectangle is not displayed around a face.

A3.

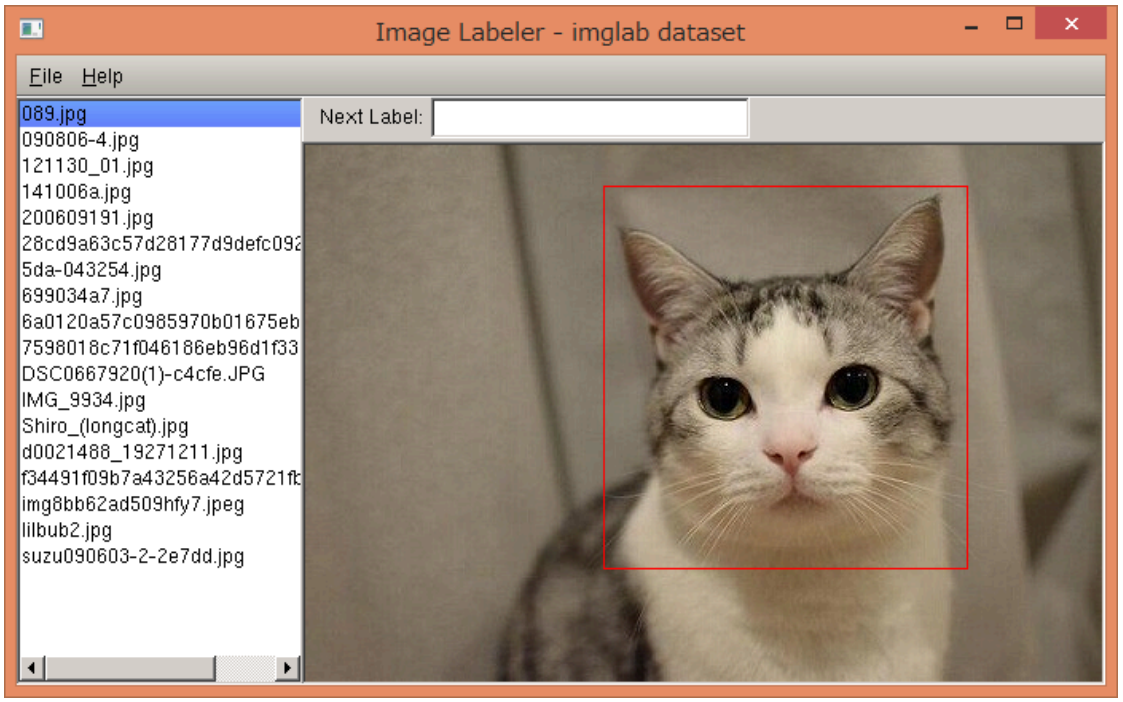
Please move the “DlibFaceLandmarkDetector/StreamingAssets/” folder to the “Assets/” folder.

Q4.

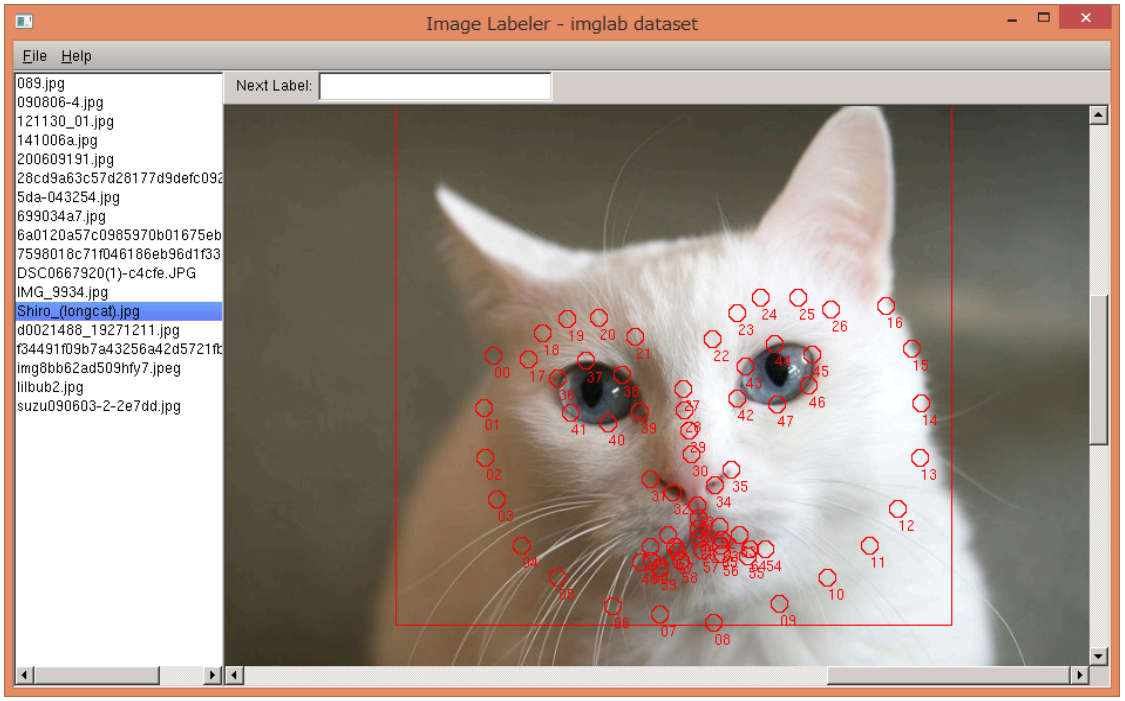
How can I train object detector?

A4.

Please refer to this page.

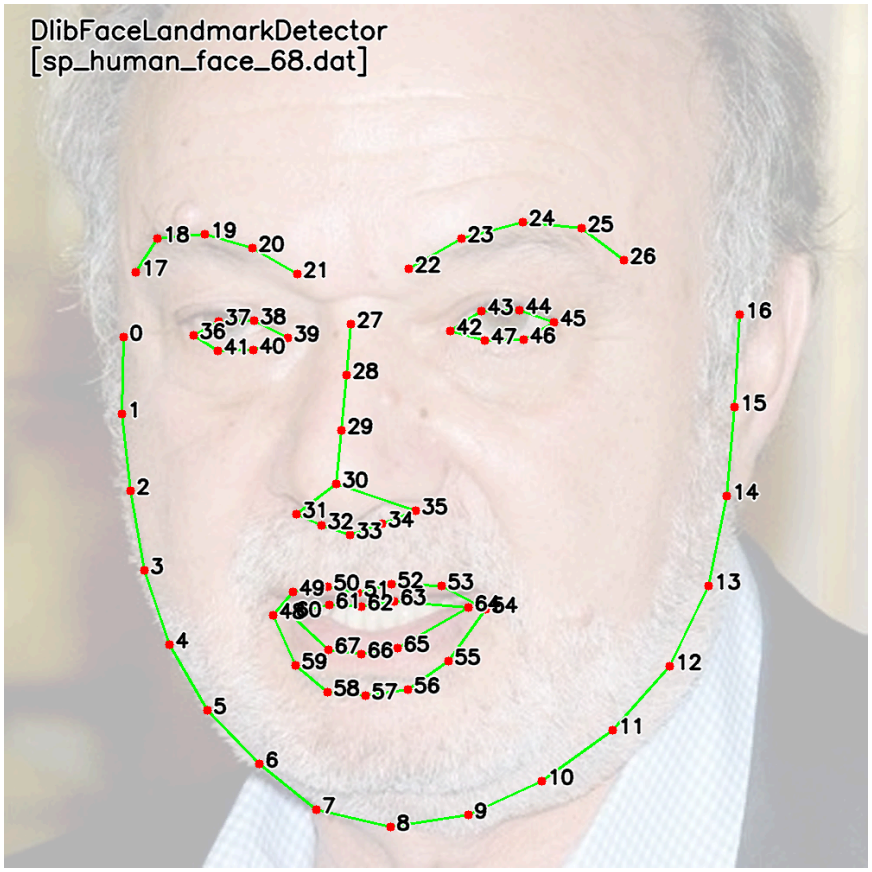


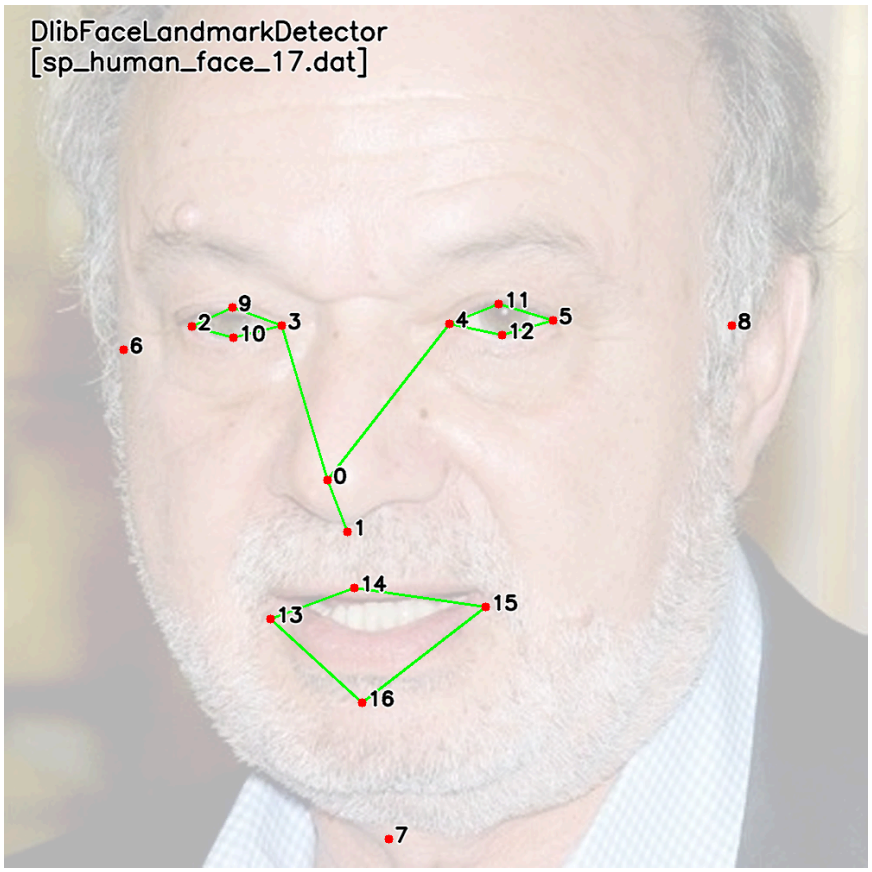
Q4.
How can I train shape predictor?
A4.
Please refer to this page.

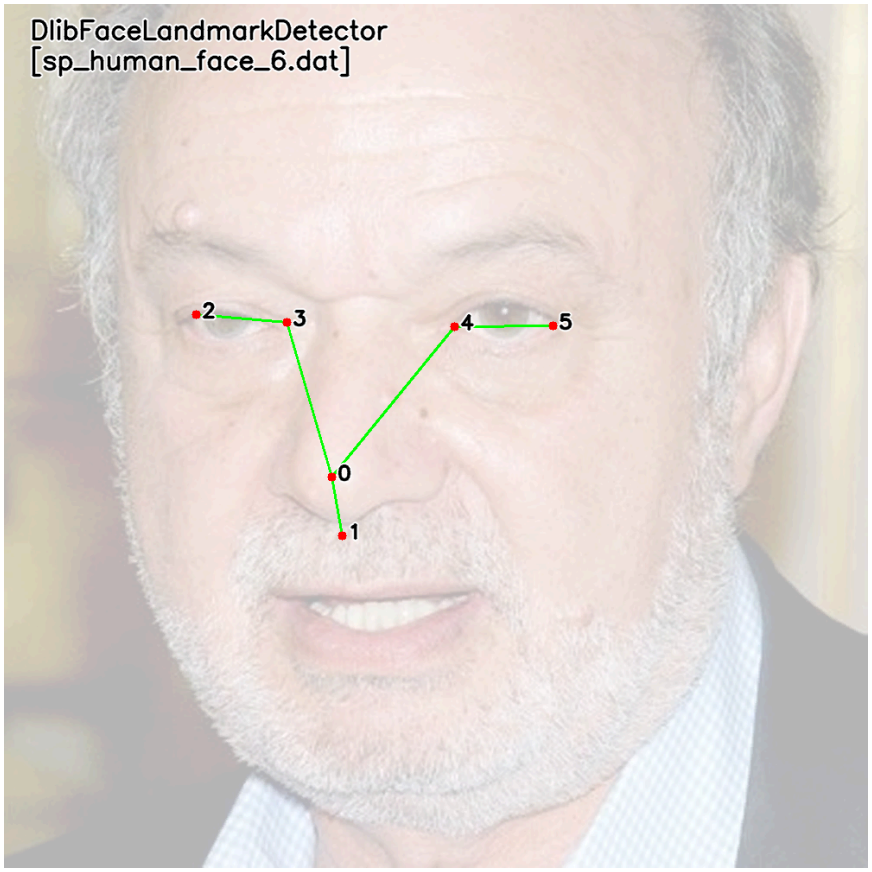


Q5.
The size of the “human_face_68_sp.dat” is too large.
A5.
Please use the “human_face_68_sp_for_mobile.dat”.
the“human_face_68_sp_for_mobile.dat” is less accurate than the “human_face_68_sp.dat”, but it is smaller size.

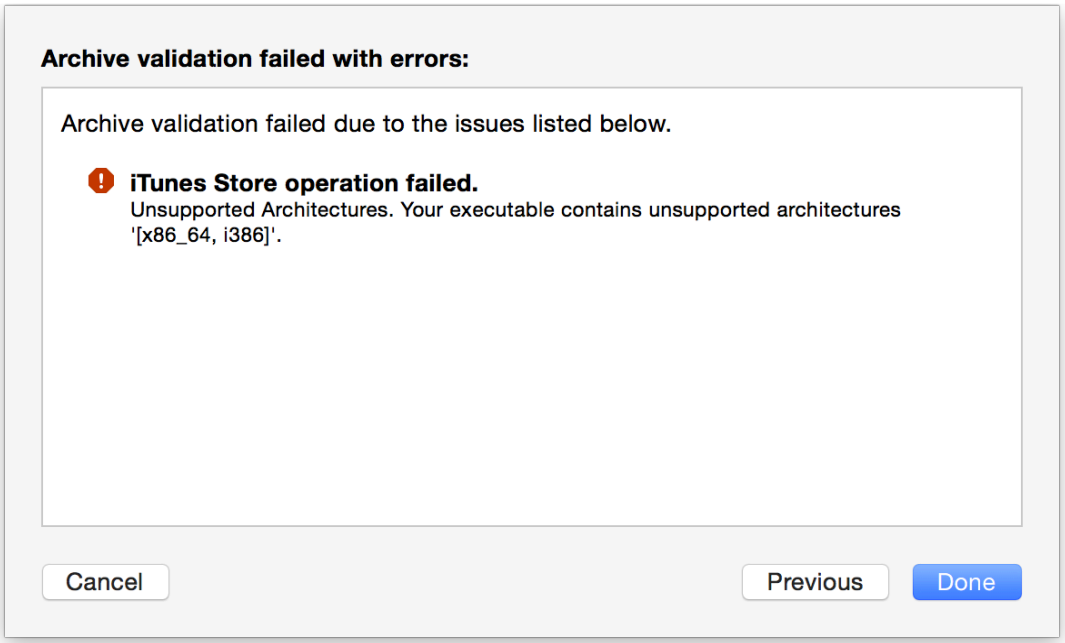
Q6.
The index of face landmark points that can be obtained using the “human_face_68_sp.dat”.
A6.


DlibFaceLandmarkDetector
[sp_human_face_68.dat]


DlibFaceLandmarkDetector
[sp_human_face_17.dat]



Q7.
[iOS]Submit to App Store issues: Unsupported Architecture x86, i386“Unsupported Architecture. Your executable contains unsupported architecture ‘[x86_64, i386]’.”

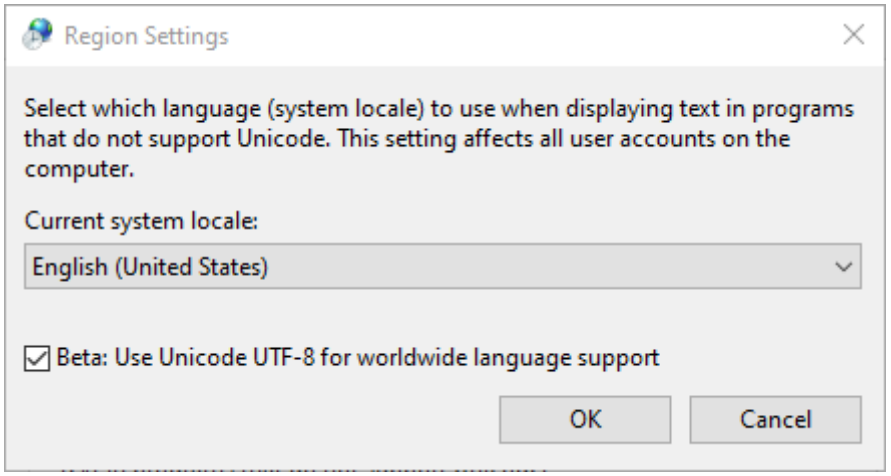


A7.
“The problem is that the Buy framework contains a build for both the simulator (x86_64) and the actual devices (ARM).

Of course, you aren’t allowed to submit to the App Store a binary for an unsupported architecture, so the solution is to “manually” remove the unneeded architectures from the final binary, before submitting it.”
http://ioscake.com/submit-to-app-store-issues-unsupported-architecture-x86.html
http://ikennd.ac/blog/2015/02/stripping-unwanted-architectures-from-dynamic-libraries-in-xcode/

Q8.
Can Shape Predictor model files be used commercially?
A8.
Since the training dataset consists of flickr CC0 licensed images, the Shape Predictor model files are available for commercial use. For details of the training data set, please refer to "DlibFaceLandmarkDetector / DlibFaceLandmarkDetectorTrainingDataset.txt".

Q9.
Using Unicode file path on Windows fails to read shapePredictorFile file.
A9.
Please enable the "Use Unicode UTF-8 for worldwide language support" setting
.All Settings -> Time & Language -> Language -> "Administrative Language Settings"



Q10.
How to catch native Dlib’s errors code (CVException handling)
A10.
In order to display the native Dlib’s error code, please enclose the code in Utils.setDebugMode(true) and Utils.setDebugMode(false).

```
Utils.setDebugMode(true); ---------- Utils.setDebugMode(false);
```

Q11.
What is the minimum file composition required for the assets to work?
A11.
You do not necessarily have to import all the files for the asset to work.
If you do not need to try the example scenes, the minimum file composition required is as follows: