// Copyright 2013 The Flutter Authors. All rights reserved.

// Use of this source code is governed by a BSD-style license that can be

// found in the LICENSE file.

#ifndef FLUTTER\_SHELL\_PLATFORM\_COMMON\_PUBLIC\_FLUTTER\_TEXTURE\_REGISTRAR\_H\_

#define FLUTTER\_SHELL\_PLATFORM\_COMMON\_PUBLIC\_FLUTTER\_TEXTURE\_REGISTRAR\_H\_

#include <stddef.h>

#include <stdint.h>

#include "flutter\_export.h"

#if defined(\_\_cplusplus)

extern "C" {

#endif

struct FlutterDesktopTextureRegistrar;

// Opaque reference to a texture registrar.

typedef struct FlutterDesktopTextureRegistrar\*

FlutterDesktopTextureRegistrarRef;

// Possible values for the type specified in FlutterDesktopTextureInfo.

// Additional types may be added in the future.

typedef enum {

// A Pixel buffer-based texture.

kFlutterDesktopPixelBufferTexture,

// A platform-specific GPU surface-backed texture.

kFlutterDesktopGpuSurfaceTexture

} FlutterDesktopTextureType;

// Supported GPU surface types.

typedef enum {

// Uninitialized.

kFlutterDesktopGpuSurfaceTypeNone,

// A DXGI shared texture handle (Windows only).

// See

// https://docs.microsoft.com/en-us/windows/win32/api/dxgi/nf-dxgi-idxgiresource-getsharedhandle

kFlutterDesktopGpuSurfaceTypeDxgiSharedHandle,

// A |ID3D11Texture2D| (Windows only).

kFlutterDesktopGpuSurfaceTypeD3d11Texture2D

} FlutterDesktopGpuSurfaceType;

// Supported pixel formats.

typedef enum {

// Uninitialized.

kFlutterDesktopPixelFormatNone,

// Represents a 32-bit RGBA color format with 8 bits each for red, green, blue

// and alpha.

kFlutterDesktopPixelFormatRGBA8888,

// Represents a 32-bit BGRA color format with 8 bits each for blue, green, red

// and alpha.

kFlutterDesktopPixelFormatBGRA8888

} FlutterDesktopPixelFormat;

// An image buffer object.

typedef struct {

// The pixel data buffer.

const uint8\_t\* buffer;

// Width of the pixel buffer.

size\_t width;

// Height of the pixel buffer.

size\_t height;

// An optional callback that gets invoked when the |buffer| can be released.

void (\*release\_callback)(void\* release\_context);

// Opaque data passed to |release\_callback|.

void\* release\_context;

} FlutterDesktopPixelBuffer;

// A GPU surface descriptor.

typedef struct {

// The size of this struct. Must be

// sizeof(FlutterDesktopGpuSurfaceDescriptor).

size\_t struct\_size;

// The surface handle. The expected type depends on the

// |FlutterDesktopGpuSurfaceType|.

//

// Provide a |ID3D11Texture2D\*| when using

// |kFlutterDesktopGpuSurfaceTypeD3d11Texture2D| or a |HANDLE| when using

// |kFlutterDesktopGpuSurfaceTypeDxgiSharedHandle|.

//

// The referenced resource needs to stay valid until it has been opened by

// Flutter. Consider incrementing the resource's reference count in the

// |FlutterDesktopGpuSurfaceTextureCallback| and registering a

// |release\_callback| for decrementing the reference count once it has been

// opened.

void\* handle;

// The physical width.

size\_t width;

// The physical height.

size\_t height;

// The visible width.

// It might be less or equal to the physical |width|.

size\_t visible\_width;

// The visible height.

// It might be less or equal to the physical |height|.

size\_t visible\_height;

// The pixel format which might be optional depending on the surface type.

FlutterDesktopPixelFormat format;

// An optional callback that gets invoked when the |handle| has been opened.

void (\*release\_callback)(void\* release\_context);

// Opaque data passed to |release\_callback|.

void\* release\_context;

} FlutterDesktopGpuSurfaceDescriptor;

// The pixel buffer copy callback definition provided to

// the Flutter engine to copy the texture.

// It is invoked with the intended surface size specified by |width| and

// |height| and the |user\_data| held by

// |FlutterDesktopPixelBufferTextureConfig|.

//

// As this is usually called from the render thread, the callee must take

// care of proper synchronization. It also needs to be ensured that the

// returned |FlutterDesktopPixelBuffer| isn't released prior to unregistering

// the corresponding texture.

typedef const FlutterDesktopPixelBuffer\* (

\*FlutterDesktopPixelBufferTextureCallback)(size\_t width,

size\_t height,

void\* user\_data);

// The GPU surface callback definition provided to the Flutter engine to obtain

// the surface. It is invoked with the intended surface size specified by

// |width| and |height| and the |user\_data| held by

// |FlutterDesktopGpuSurfaceTextureConfig|.

typedef const FlutterDesktopGpuSurfaceDescriptor\* (

\*FlutterDesktopGpuSurfaceTextureCallback)(size\_t width,

size\_t height,

void\* user\_data);

// An object used to configure pixel buffer textures.

typedef struct {

// The callback used by the engine to copy the pixel buffer object.

FlutterDesktopPixelBufferTextureCallback callback;

// Opaque data that will get passed to the provided |callback|.

void\* user\_data;

} FlutterDesktopPixelBufferTextureConfig;

// An object used to configure GPU-surface textures.

typedef struct {

// The size of this struct. Must be

// sizeof(FlutterDesktopGpuSurfaceTextureConfig).

size\_t struct\_size;

// The concrete surface type (e.g.

// |kFlutterDesktopGpuSurfaceTypeDxgiSharedHandle|)

FlutterDesktopGpuSurfaceType type;

// The callback used by the engine to obtain the surface descriptor.

FlutterDesktopGpuSurfaceTextureCallback callback;

// Opaque data that will get passed to the provided |callback|.

void\* user\_data;

} FlutterDesktopGpuSurfaceTextureConfig;

typedef struct {

FlutterDesktopTextureType type;

union {

FlutterDesktopPixelBufferTextureConfig pixel\_buffer\_config;

FlutterDesktopGpuSurfaceTextureConfig gpu\_surface\_config;

};

} FlutterDesktopTextureInfo;

// Registers a new texture with the Flutter engine and returns the texture ID.

// This function can be called from any thread.

FLUTTER\_EXPORT int64\_t FlutterDesktopTextureRegistrarRegisterExternalTexture(

FlutterDesktopTextureRegistrarRef texture\_registrar,

const FlutterDesktopTextureInfo\* info);

// Asynchronously unregisters the texture identified by |texture\_id| from the

// Flutter engine.

// An optional |callback| gets invoked upon completion.

// This function can be called from any thread.

FLUTTER\_EXPORT void FlutterDesktopTextureRegistrarUnregisterExternalTexture(

FlutterDesktopTextureRegistrarRef texture\_registrar,

int64\_t texture\_id,

void (\*callback)(void\* user\_data),

void\* user\_data);

// Marks that a new texture frame is available for a given |texture\_id|.

// Returns true on success or false if the specified texture doesn't exist.

// This function can be called from any thread.

FLUTTER\_EXPORT bool

FlutterDesktopTextureRegistrarMarkExternalTextureFrameAvailable(

FlutterDesktopTextureRegistrarRef texture\_registrar,

int64\_t texture\_id);

#if defined(\_\_cplusplus)

} // extern "C"

#endif

#endif // FLUTTER\_SHELL\_PLATFORM\_COMMON\_PUBLIC\_FLUTTER\_TEXTURE\_REGISTRAR\_H\_