// 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\_WINDOWS\_CLIENT\_WRAPPER\_INCLUDE\_FLUTTER\_FLUTTER\_VIEW\_CONTROLLER\_H\_

#define FLUTTER\_SHELL\_PLATFORM\_WINDOWS\_CLIENT\_WRAPPER\_INCLUDE\_FLUTTER\_FLUTTER\_VIEW\_CONTROLLER\_H\_

#include <flutter\_windows.h>

#include <windows.h>

#include <memory>

#include <optional>

#include "dart\_project.h"

#include "flutter\_engine.h"

#include "flutter\_view.h"

namespace flutter {

// A controller for a view displaying Flutter content.

//

// This is the primary wrapper class for the desktop C API.

// If you use this class, you should not call any of the setup or teardown

// methods in the C API directly, as this class will do that internally.

class FlutterViewController {

public:

// Creates a FlutterView that can be parented into a Windows View hierarchy

// either using HWNDs.

//

// This also creates a new FlutterEngine.

//

// |dart\_project| will be used to configure the engine backing this view.

FlutterViewController(int width, int height, const DartProject& project);

virtual ~FlutterViewController();

// Prevent copying.

FlutterViewController(FlutterViewController const&) = delete;

FlutterViewController& operator=(FlutterViewController const&) = delete;

// Returns the view controller's view ID.

FlutterViewId view\_id() const;

// Returns the engine running Flutter content in this view.

FlutterEngine\* engine() const { return engine\_.get(); }

// Returns the view managed by this controller.

FlutterView\* view() const { return view\_.get(); }

// Requests new frame from the engine and repaints the view.

void ForceRedraw();

// Allows the Flutter engine and any interested plugins an opportunity to

// handle the given message.

//

// If a result is returned, then the message was handled in such a way that

// further handling should not be done.

std::optional<LRESULT> HandleTopLevelWindowProc(HWND hwnd,

UINT message,

WPARAM wparam,

LPARAM lparam);

private:

// Handle for interacting with the C API's view controller, if any.

FlutterDesktopViewControllerRef controller\_ = nullptr;

// The backing engine

std::shared\_ptr<FlutterEngine> engine\_;

// The owned FlutterView.

std::unique\_ptr<FlutterView> view\_;

};

} // namespace flutter

#endif // FLUTTER\_SHELL\_PLATFORM\_WINDOWS\_CLIENT\_WRAPPER\_INCLUDE\_FLUTTER\_FLUTTER\_VIEW\_CONTROLLER\_H\_