FancyZones is the base class that runs the show. It uses hooks to monitor for windows entering and exiting the move/size loop and to listen for key presses for hotkey interception. For every connected display, it creates a ZoneWindow which is used to display the active ZoneSet per monitor for use when editing the layout or displaying the drop targets. A ZoneSet is composed of one or more Zones which are the locations where windows can be easily positioned.

SetWinEventHook

The main driving force behind FancyZones is the accessibility hook used to know when a window enters the move/size loop. It listens for EVENT_SYSTEM_MOVESIZESTART, EVENT_SYSTEM_MOVESIZEEND, and EVENT_OBJECT_LOCATIONCHANGE. For each of these three events, it forwards on to the ZoneWindow associated with the monitor that the window being dragged is currently on.

Keyboard Hook

A low-level keyboard hook is installed in order to, optionally, intercept Window+Arrow hotkeys. Traditionally, Win+Left/Right arrow will move a window between Windows Snap regions. This hook allows FancyZones to use Win+Left/Right arrow to move windows between Zones. The hook also allows using 0-9 to change the active ZoneSet during a drag operation.

Display Changes

During initial standup, FancyZones creates a ZoneWindow for each connected monitor. When it receives a WM_DISPLAYCHANGE, it updates the available ZoneWindows to reflect the state of the system (eg add a new ZoneWindow for newly connected monitor, delete ZoneWindow for disconnected monitor, etc)

Interface

```
interface IFancyZones : public IUnknown
{
    // Returns the main application window
    IFACEMETHOD_(HWND, GetWindow)() = 0;

    // Returns the global HINSTANCE for the process
    IFACEMETHOD_(HINSTANCE, GetHInstance)() = 0;

    // Returns the global Settings object used to look up individual settings
throughout the product
    IFACEMETHOD_(Settings, GetSettings)() = 0;

    // Used in WinMain to initialize FancyZones and enter the message loop
    IFACEMETHOD_(void, Run)() = 0;

    // Toggles the visibility of all ZoneWindows
    IFACEMETHOD_(void, ToggleZoneViewers)() = 0;

    // Shows a single ZoneWindow in editor mode on the provided monitor
    IFACEMETHOD_(void, ShowZoneEditorForMonitor)(_In_ HMONITOR monitor) = 0;

    // Returns true if we are currently detecting a movesize loop
```

```
    IFACEMETHOD_(bool, InMoveSize)() = 0;

    // Called by the event hook in response to EVENT_SYSTEM_MOVESIZESTART
    IFACEMETHOD(MoveSizeEnter)(_In_ HWND window, _In_ HMONITOR monitor, POINT
ptScreen) = 0;

    // Called by the event hook in response to EVENT_SYSTEM_MOVESIZEEND
    IFACEMETHOD(MoveSizeExit)(_In_ HWND window, POINT ptScreen) = 0;

    // Called by the event hook in response to EVENT_OBJECT_LOCATIONCHANGE
    IFACEMETHOD(MoveSizeUpdate)(_In_ HMONITOR monitor, POINT ptScreen) = 0;

    // Called during startup or on WM_DISPLAYCHANGE to add a ZoneWindow to the
collection
    // There will be one ZoneWindow per connected monitor
    IFACEMETHOD(AddZoneWindow)(_In_ IZoneWindow* zoneWindow, _In_ HMONITOR monitor) =
0;

    // Called in response to WM_DISPLAYCHANGE from the main application window
    IFACEMETHOD_(void, OnDisplayChange)(DisplayChangeType changeType) = 0;

    // Used to move the specified HWND into Zone
    // The ZoneSet used is found by looking up the monitor that window is currently on
    // This gets called to keep windows in their current zones after a
WM_DISPLAYCHANGE
    IFACEMETHOD_(void, MoveWindowIntoZoneByIndex)(_In_ HWND window, int index) = 0;

    // Used to filter out windows that the hook should not be processing
    // Currently checks if the window's GWL_STYLE has WS_MAXIMIZEBOX
    IFACEMETHOD_(bool, IsInterestingWindow)(_In_ HWND window) = 0;

    // Called byt the event hook in response to EVENT_OBJECT_NAMECHANGE on the Desktop
window
    // The accessible name of the desktop window changes when the current virtual
desktop changes
    IFACEMETHOD_(void, VirtualDesktopChanged)() = 0;

    // Returns the GUID of the current active virtual desktop
    IFACEMETHOD_(GUID, GetCurrentVirtualDesktopId)() = 0;

    // Called by the LL keyboard hook
    // Used to override snap hotkeys and to change the active ZoneSet during a drag
    IFACEMETHOD_(bool, OnKeyDown)(LPARAM lparam) = 0;

    // Keep windows positioned inside their zones when the active ZoneSet changes
    IFACEMETHOD_(void, MoveWindowsOnActiveZoneSetChange)() = 0;
};
```

ZoneWindow

ZoneWindow is used to display the Zones a user can drop a window in during a drag operation, flash the Zones when the ZoneSet changes, and draw the Zone Editor UI when in edit mode. Basically, when a ZoneSet needs to be visualized, ZoneWindow does it.

Interface

```
interface IZoneWindow : public IUnknown
{
    // Shows the ZoneWindow
    // If activate is true, set foreground to the window otherwise just show
    IFACEMETHOD(ShowZoneWindow)(bool activate) = 0;

    // Hide the ZoneWindow
    IFACEMETHOD(HideZoneWindow)() = 0;

    // Called when the drag enters the monitor this ZoneWindow is assigned to
    IFACEMETHOD(MoveSizeEnter)(_In_ HWND window, POINT ptScreen, DragMode dragMode) =
0;

    // Called when the drag exits the monitor
    IFACEMETHOD(MoveSizeExit)(_In_ HWND window, POINT ptScreen) = 0;

    // Called when the drag updates position on this monitor
    IFACEMETHOD(MoveSizeUpdate)(POINT ptScreen, DragMode dragMode) = 0;

    // Called when a drag ends and the window is not dropped in a Zone
    IFACEMETHOD(MoveSizeCancel)() = 0;

    // Returns the DragMode of the current drag operation
    // DragMode allows for overriding drag behavior via settings or via hotkey
    IFACEMETHOD_(DragMode, GetDragMode)() = 0;

    // Part of the chain to move a window into a specific Zone
    IFACEMETHOD_(void, MoveWindowIntoZoneByIndex)(_In_ HWND window, int index) = 0;

    // Used to cycle a window between zones via the hijacked snap hotkeys
    IFACEMETHOD_(void, MoveWindowIntoZoneByDirection)(_In_ HWND window, DWORD vkCode)
= 0;

    // Called in response to WM_DISPLAYCHANGE
    // Allows cleanup, if necessary, since ZoneWindow will be destroyed shortly
thereafter
    IFACEMETHOD_(void, OnDisplayChange)(DisplayChangeType type) = 0;

    // Allows changing the active ZoneSet via key press either during a drag or while
the ZoneWindow is in foreground
    IFACEMETHOD_(void, CycleActiveZoneSet)(DWORD vkCode) = 0;
};
```

ZoneSet

Collection of one or more Zones. Only one ZoneSet is active at a time per monitor.

Interface

```
interface IZoneSet : public IUnknown
{
    // Gets the unique ID used to identify this ZoneSet
    IFACEMETHOD_(GUID, GetId)() = 0;

    // Adds a Zone to the collection
    IFACEMETHOD(AddZone)(_In_ Microsoft::WRL::ComPtr<IZone> zone, bool front) = 0;

    // Removes a Zone from the collection
    IFACEMETHOD(RemoveZone)(_In_ Microsoft::WRL::ComPtr<IZone> zone) = 0;

    // Returns the topmost Zone at the given point
    IFACEMETHOD_(Microsoft::WRL::ComPtr<IZone>, ZoneFromPoint)(POINT pt) = 0;

    // Returns a Zone that the window is in
    // Will return nullptr if the window is not in a Zone
    IFACEMETHOD_(Microsoft::WRL::ComPtr<IZone>, ZoneFromWindow)(_In_ HWND window) = 0;

    // Gets all the Zones
    IFACEMETHOD_(std::vector<Microsoft::WRL::ComPtr<IZone>>, GetZones)() = 0;

    // ZoneSetLayout
    // * Grid - Pregenerated layout (2x2, 3x3, etc)
    // * Row - Pregenerated layout in a single row
    // * Focus - Pregenerated layout with a central focus Zone and fanned peripheral
Zones
    // * Custom - User generated Zone
    IFACEMETHOD_(ZoneSetLayout, GetLayout)() = 0;

    // The amount of default padding between Zones in a generated layout
    IFACEMETHOD_(int, GetInnerPadding)() = 0;

    // Makes a copy of the IZoneSet and marks it as ZoneSetLayout::Custom
    IFACEMETHOD_(Microsoft::WRL::ComPtr<IZoneSet>, MakeCustomClone)() = 0;

    // Persists ZoneSet data to the registry
    IFACEMETHOD_(void, Save)() = 0;

    // Moves a Zone to the front of the collection
    IFACEMETHOD_(void, MoveZoneToFront)(_In_ Microsoft::WRL::ComPtr<IZone> zone) = 0;

    // Moves a Zone to the back of the collection
    IFACEMETHOD_(void, MoveZoneToBack)(_In_ Microsoft::WRL::ComPtr<IZone> zone) = 0;

    // Part of the chain to move a window into a specific Zone
    IFACEMETHOD_(void, MoveWindowIntoZoneByIndex)(_In_ HWND window, _In_ HWND
zoneWindow, int index) = 0;
```

```
    // Part of the chain to move a window into a specific Zone
    IFACEMETHOD_(void, MoveWindowIntoZoneByDirection)(_In_ HWND window, _In_ HWND
zoneWindow, DWORD vkCode) = 0;

    // Called when a drag ends or leaves the monitor this ZoneWindow is on
    // This will remove the window from its currently assigned Zone and assign it
    // to a different Zone based on the current cursor position
    IFACEMETHOD_(void, MoveSizeExit)(_In_ HWND window, _In_ HWND zoneWindow, _In_
POINT ptClient) = 0;
};
```

Zone

Basically a RECT and a map of HWND->RECT to keep track of where windows can be placed and which windows are currently in the Zone.

Interface

```
interface IZone : public IUnknown
{
    // Returns the RECT that this Zone represents
    IFACEMETHOD_(RECT, GetZoneRect)() = 0;

    // Returns true if the specified window is in this Zone's collection
    IFACEMETHOD_(bool, ContainsWindow)(_In_ HWND window) = 0;

    // Adds the window the collection
    IFACEMETHOD_(void, AddWindowToZone)(_In_ HWND window, _In_ HWND zoneWindow, bool
stampZone) = 0;

    // Removes the window from the collection
    IFACEMETHOD_(void, RemoveWindowFromZone)(_In_ HWND window, bool restoreSize) = 0;

    // Sets an id for this Zone
    // The id will be unique per ZoneSet
    IFACEMETHOD_(void, SetId)(size_t id) = 0;

    // Returns the id given to this Zone
    IFACEMETHOD_(size_t, GetId)() = 0;
};
```