Keymapping spec

- author: Mike Griese **migrie**
- created on: 2018-Oct-23

Abstract

It should be possible to configure the terminal so that it doesn't send certain keystrokes as input to the terminal, and instead triggers certain actions. Examples of these actions could be copy/pasting text, opening a new tab, or changing the font size.

This spec describes a mechanism by which we could provide a common implementation of handling keyboard shortcuts like these. This common implementation could then be leveraged and extended by the UX implementation as to handle certain callbacks in the UX layer. For example, The TerminalCore doesn't have a concept of what a tab is, but the keymap abstraction could raise an event such that a WPF app could implement creating a new tab in its idiomatic way, and UWP could implement them in their own way.

Terminology

- **Key Chord**: This is any possible keystroke that a user can input simultaneously, as a combination of a single character and any set of (Ctrl, Alt and Shift). For example, pressing Ctrl and C at the same time is the key chord Ctrl+C. Pressing Ctrl+B, C are two separate key chords. Trying to press them simultaneously (Ctrl+B+C) should generate two separate key chords, with the order determined by the OS.

User Stories

1. The User should be able to press certain key-chords to trigger certain actions in the frontend of the application, such as copying text, pasting, opening new tabs, or switching focus between panes.
2. The user should be able to configure which key chords are bound to which actions.
3. If a key chord is not mapped to an action, it should be sent to the Terminal just as any other keypress.

Details

When the UX frontend is created, it should instantiate a `IKeyBindings` object with the keybindings mapped as it would like.

When it's creating its platform-dependent terminal component, it can pass the `IKeyBindings` object to that component. The component will then be able to pass that object to the terminal instance.

When the terminal component calls `ITerminalInput.SendKeyEvent(uint vkey, KeyModifiers modifiers)`, the terminal will use `IKeyBindings.TryKeyChord` to see if there are any bound actions to that input. If there are, the `IKeyBindings` implementation will either handle the event by interacting with the `ITerminalInput`, or it'll invoke an event that's been registered by the frontend

```
struct KeyChord
{
    KeyModifiers modifiers;
    int vkey;
}

interface IKeyBindings {
    bool TryKeyChord(KeyChord kc);
}
```

Each frontend can implement the `IKeyBindings` interface however it so chooses.

The `ITerminalInput` interface will be extended with the following method:

```
public interface ITerminalInput
{
    ...
    void SetKeyBindings(IKeyBindings bindings);
    ...
}
```

This will set the `IKeyBindings` object that the `ITerminalInput` implementation will use to filter key events.

This method will be implemented by the `Terminal` object:

```
partial class Terminal
{
    public void ITerminalInput.SetKeyBindings(IKeyBindings bindings);
}
```

Project Cascadia Sample

Below is an example of how the Project Cascadia application might implement its keybindings.

```
enum ShortcutAction
{
    CopyText,
    PasteText,
    NewTab,
    NewWindow,
    CloseWindow,
```

```
    CloseTab,
    SwitchToTab,
    NextTab,
    PrevTab,
    IncreaseFontSize,
    DecreaseFontSize,
    ...
}
public delegate bool NewTabEvent(object sender);
public delegate bool CloseTabEvent(object sender);
public delegate bool NewWindowEvent(object sender);
public delegate bool CloseWindowEvent(object sender);
public delegate bool CopyEvent(object sender);
public delegate bool PasteEvent(object sender);

class KeyBindings : IKeyBindings
{
    private Dictionary<ShortcutAction, KeyChord?> keyShortcuts;

    public void SetKeyBinding(ShortcutAction action, KeyChord? chord);
    public bool TryKeyChord(KeyChord chord);
}
```

Copy/Paste

How does Copy/paste play into this?

When Input is written to the terminal, and it tries the copy keybinding, what happens? The Keybindings are global to the frontend, not local to the terminal. Copy/Paste events should also be delegates that get raised, and the frontend can then determine what to do with them. It'll probably query its active/focused Terminal Component, then Get the `ITerminalInput` from that component, and use that to CopyText / PasteText from the Terminal as needed.