Keyboard alternative to drag and drop

Joop Ringelberg 17-08-20 Version: 1

Problem statement

The InPlace GUI relies heavily on the notion of inserting a role card into a role slot, to create a binding (or to create a new role filled with another). Mouse-mediated drag and drop is a very natural way to perform this action, but is unavailable to some desktop users and obviously not available on mobile or tablet.

General solution

We solve the problem by adding another method of role binding, that relies entirely on the keyboard on the desktop, and on finger gestures on mobile/tablet (from now on I'll just mention the mobile, implicitly understanding the tablet is included too).

The approach consists of steps:

1. Selecting a card;
2. Putting it on a clipboard;
3. Selecting a dropzone;
4. Inserting the card in the clipboard into the dropzone.

Card selection and putting it on the clipboard

On the desktop, we'll make a list of cards tab-navigable. On focussing on such a list, the first card (if any) is automatically selected[1]. Then the user

- can select cards with the up- and down arrow;
- can put the selected card on the clipboard with the space bar or the enter key;
- can leave the card list with the tab key (or the shift-tab key). On leaving, the visual markers on the selected card are removed.

Alternatively, the user can select a card with a mouse click.

On mobile, the user taps the card of his choice with his finger to put it on the clipboard.

Selecting a dropzone and putting the card into it

On the desktop, we'll make the dropzone tab-navigable. When a dropzone receives the focus, nothing happens (otherwise than indicating focus) if no card is on the clipboard. When there is a card on the clipboard, if the (role represented by the) card on the

[1] A CardList may remember a selection from a previous interaction and show that on receiving focus.

clipboard can fill the role represented by the dropzone, the latter will show green; otherwise it will show red. By pressing enter or space, the role is bound.

Alternatively, the user can select a dropzone with a mouse click.

On mobile, the user taps a dropzone to find out if the card is compatible with it or not. By tapping a second time (**not** a double-tap!), the role is bound.

The clipboard

On the desktop, the card on the clipboard is shown on the left side of the screen, in an overlay of the screen itself. It cannot be selected or interacted with, otherwise than described above **and** it can always be removed from the clipboard by pressing the escape key.

On mobile, the card on the clipboard can be swiped in on a panel that moves in from the left. By double-tapping it, it is removed from the clipboard (alternatively we can provide a button on the panel).

Technical approach

We just describe the desktop solution.

CardList

We create a React component CardList that expects a Role component that houses a Card component. These cards must be marked with the style `cardListItem`. This style adds no visible markup.

The CardList is tab-navigable and responsible for

- finding all the cards (we may keep the result in state);
- the key handlers for the up and down arrow, space bar and return key;
- the mouse click handler that selects a card. Note: this handler should give the CardList focus programmatically first, to remove focus from other elements. This may lead to automatic selection of the first element, which then must be overridden by the clicked element.
- keeping the selected card in state;
- visibly marking the selected card;
- selecting the first card on receiving focus (or the remembered selection);
- removing the visible markers on the selected card on losing focus;
- putting the selected card in the clipboard.

The CardList may remember a selected card after it loses focus. On again receiving focus, it may show that card as selected. If the card remembered as selected no longer exists, the first element is selected.

CardDropZone

The CardDropZone is tab-navigable and responsible for

- showing itself as having focus, visibly different in three states:
    - neutral when no card is on the clipboard;
    - green when a compatible card is on the clipboard;
    - red otherwise.
- the keyboard handler for the space bar and the return key to perform the role binding. Obviously, only in the ‘green state’!
- removing the card from the clipboard after a successful role binding;
- the mouse click handler to select itself;

Note. I assume it is impossible to first select a dropzone and then select a card, because that implies the dropzone loses focus first.

CardClipBoard

The CardClipBoard is responsible for

- building the selected card from data provided by the selection handler of the CardList;
- showing it.

We do not make CardClipBoard responsible for keeping a card in its state. Instead, we have the MySystem component do that. CardClipBoard receives the selected card as one of its props.

We build the card to show on the clipboard from a ref to the original card component, then cloning it and returning it as (part of) the result of the CardClipBoard render function.

CardClipBoard uses the Overlay component. In InPlace, we provide the column that houses the screens of the Apps as the target to Overlay and the cloned card as the overlay itself, positioning it to the left of the column.

MySystem: connection between components

Both the CardList and the CardDropZone must be able to address the CardClipBoard. The Electron version of InPlace has just a single window and consequently just a single clipboard. In effect, the `MySystem` React component is the root of the entire application. We should ‘publish’ the CardClipBoard from this component. A straightforward way to do so is by having MySystem provide a Context. Because MySystem is root, its context can be requested anywhere in the component tree.

So we extend MySystem to make it responsible for:

- keeping the selected role in state;
- a function on its context to set a role in state (to be called by the CardDropZone);
- a function on its context to remove the selected role from state (idem)
- the keyboard handler for removing the selected role from state (escape key).

Multiple windows in a browser version

A future browser version of InPlace may run in multiple windows. We then have to solve the issue of providing them with a single shared clipboard state.