Canvas Interaction for Krita

Introduction

This proposal discusses a change to the way canvas interaction is handled within Krita. I will discuss the current situation, a set of proposed changes and how I plan to implement these changes.

Current Situation

There are currently several actions within Krita that deal directly with interaction with the canvas, for example panning, rotating and zooming the canvas. Currently, these actions are handled in different ways depending on what the current tool is and in several cases are simply duplicated across tools. For example, when you want to pan the canvas, you either use the pan tool, or you use the panning "embedded" within another tool. Zooming can be similarly performed in different ways while rotating the canvas can be done only by the pan tool or by using keyboard shortcuts.

This scattering of functions across the different tools obviously not ideal. In the long run, it will only serve to confuse new users and hamper the work flow of experienced users. In addition, the code behind it is not really extensible or modular, which means that the implementation of new features may be hampered or even impossible.

Proposed Changes

The first and foremost change I propose is to define a set of universal actions that interact with the canvas and a default set of shortcuts for using these actions, trying to stay as close to current default behaviour as possible. Each of these actions is discussed below, with details on how each action behaves.

Once the universal actions have been implemented, I propose to implement a configuration page to configure these actions, specifically to make it possible to assign several alternatives to these actions. For example, for zooming, it would be great to be able to use at least four different methods of input: mouse button and drag, mouse wheel, keyboard shortcuts and a pinch gesture for tablets that support those.

Proposed Actions

The following is a list of proposed universal actions, with descriptions of each action. They are designed to mostly reflect the existing actions, but in a more universal way. Each action also lists the proposed default shortcut associations. These shortcuts initiate the action. Once an action is started, additional input events will be sent directly to an action. This means that, for example, when you start painting with the Brush tool, the Tool Invocation action will be started. While you continue with that action, simply by holding down the mouse button or pen, all other input will be sent to this action. As soon as you release the mouse button or pen, the action will be stopped and you can start another. This makes it possible for individual

Primary Invocation Alternative Invocation Change Primary Setting

Figure 1 Tool invocation with the Brush tool

actions to change their behaviour once they are started depending on keys, like snapping to a certain angle for the line tool.

Tool Invocation

Tool Invocation is the action of applying the current tool to the canvas. While it may not be considered to be actual canvas interaction, I have included it to make it possible to create a more generalised framework for canvas input.

Tool invocation is divided into three separate actions:

- **'Primary' tool invocation:** Primary tool invocation simply invokes the tool. So in case of the Brush tool it will begin painting, in case of the selection tool it will begin a selection, etc.
  Default Association: Left Mouse Button
- **Alternative tool invocation:** This is a generalisation of the 'colour picker' action currently only available in some tools. Each tool will get access to this action and the precise implementation is tool dependant. The default implementation will be to pick a colour from the canvas, but individual tools can override this.
  Default Association: Control + Left Mouse Button
- **Change tool primary setting:** This is a generalisation of the 'change brush size' action currently only available in the Brush tool. Each tool will have access to this action and the precise implementation is tool dependant. There will be no default implementation since the tools are too different to define a common setting to modify. One implementation that is already defined is to change the brush size of the Brush tool.
  Default Association: Shift + Left Mouse Button

Open Pop-up Palette

Figure 2 The Pop-up Palette

The pop-up palette provides quick access to

a colour selector, colour history and favourite presets. It is designed to provide quick access to an artist's most often used tools. Since there is only a single possible mode for this action, keyboard shortcuts and other methods of input produce the same behaviour, which is to toggle the visibility of the palette at the current mouse position.

Default Association: Right Mouse Button

Pan

Figure 3 The Pan Overlay Widget

The pan action allows you to change the position of the canvas. The pan action has an associated widget, which can be seen in Figure 3, which shows the current position of the viewport relative to the canvas. It can be dragged to re-position the viewport. The pan action has two sets of associated shortcuts. The first is a single button that, when held, shows the widget and then allows panning by dragging the canvas. The second is a set of four keys to pan in discrete steps.

Default Association: Toggle: Space, Middle Mouse. Direct: Arrow Keys

Rotate

Figure 4 The Rotate Overlay Widget

The rotate action allows you to change the rotation of the canvas. The rotate action has an associated widget. It can be seen in Figure 4. The widget shows the current rotation of the canvas. A click on the widget will reset the rotation. The rotate action also defines two sets of shortcuts. The first one is a button that, when held, shows the widget and then allows rotation of the canvas by dragging the canvas. The second is a set of three buttons, one to rotate clockwise by a certain amount, the other to rotate counterclockwise and one to reset the rotation.

Default Association: Toggle: Shift + Space, Shift + Middle Mouse. Direct: Numpad 4 (CCW), Numpad 5 (Reset), Numpad 6 (CW)

Zoom

The zoom action allows you to zoom the canvas. Like the Pan and Rotate actions, it also has an associated widget, which can be seen in Figure 5. The widget shows a track and a

Figure 5 The Zoom Overlay Widget

handle that indicates the current zoom level. The handle can be dragged to change the zoom level. The zoom action defines three sets of shortcuts. The first one is again a button. When held it shows the widget and then allows zooming by dragging the canvas up or down. The second set are two keys, one for zooming in and one for zooming out. Both use fixed steps for the zoom values. The third set are two keys, one for resetting the zoom to 100% and the other for zooming the canvas so it fits the viewport.

Default Association: Toggle: Ctrl + Space, Ctrl + Middle Mouse. Direct: Numpad - , Numpad +. Reset: Numpad 1 (100%), Numpad 0 (Fit to Page)

Other Actions

There are several additional actions related to the canvas that require research whether they should be universal actions or whether a different way of handling would be better.

- **Setup Mirror Axis:** Changes the axis used for mirrored painting. It is not often used and a more direct approach may work better.
- **Modify Assistants:** Currently a tool, but more directly related to the canvas. It may make sense to combine this with the Setup Mirror Axis action and create a single 'Modify Assistants' action.

Changes to Tools

Since some of the tools have now become universal actions, the tools that are related to those actions should be removed, as they only duplicate functionality. The primary tool to remove is the pan tool, since its functionality is now completely covered by the Pan and Rotate actions. The Zoom tool can also be removed, however, if there is a desire to keep the zoom to area functionality, a simplified version of the zoom tool could be kept around.

A third tool that can be removed is the the colour picker tool, once some of the settings it allows are moved to different locations. It has been suggested to use a docker for this, it does seem to be the most accessible location. The docker should contain a setting for which layers to use, whether to use the average of all pixels when doing a drag or simply the last position and the radius of the sampling area.

Once a good solution has been devised for assistant editing, the assistant tool can also be changed to merely create assistants. It might even be possible to move this to its own docker as well, though I am unsure whether this is desirable. Right now creating assistants at least is fairly obvious and simple.

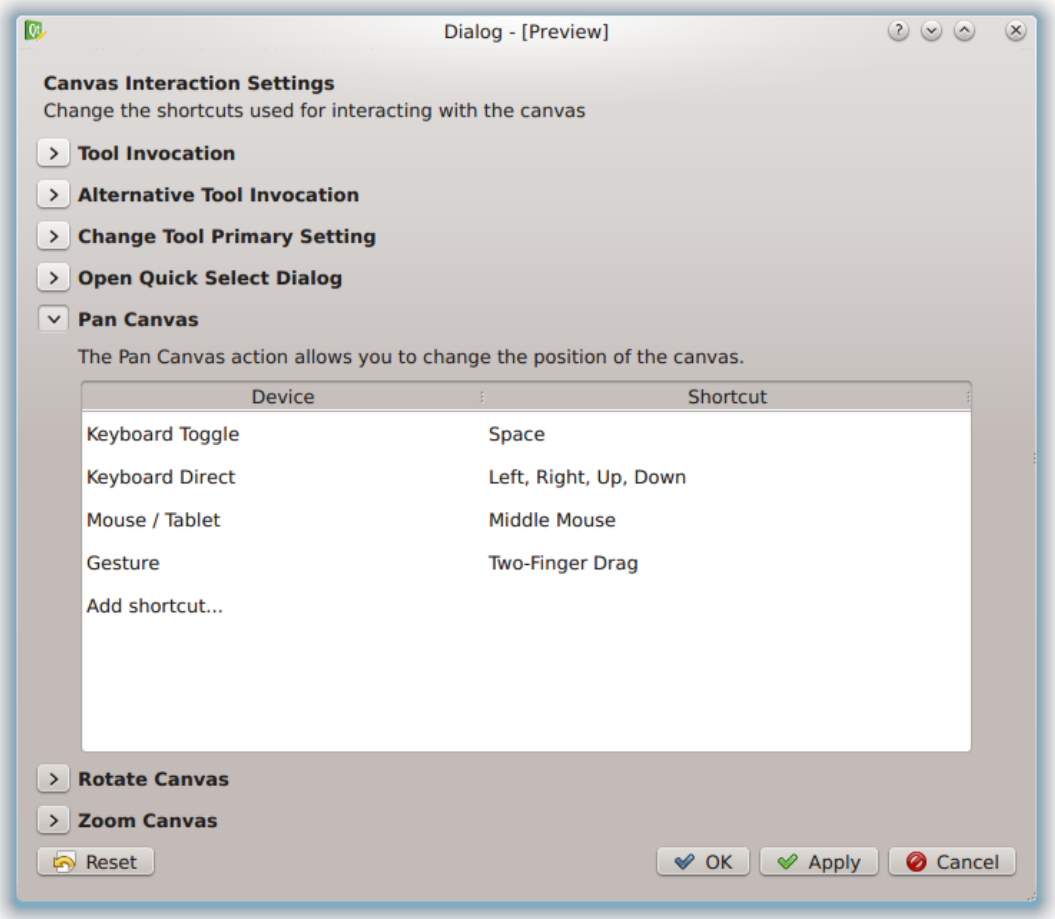

Figure 6 Mockup of the Configuration Dialog

Configuration

The configuration of the actions is meant to be included in the global configuration dialog. It allows configuration of all the shortcuts related to the actions described in *Proposed Actions*.

A mockup of this dialog can be seen in Figure 6. It lists each available action, with its associated shortcuts. The precise sets of shortcuts available are defined by each action. Each action allows multiple of these sets of shortcuts to be added, so it becomes easy to add alternative sets of shortcuts. Later on, it may also be interesting to implement profiles for the shortcuts, primarily to enable different sets of default shortcuts.

Implementation Details

The first step of the implementation will be to create a class that manages the actions and the associated shortcuts. It will install an event filter on the canvas and that way intercept any input events from the canvas. It then uses the input events to decide which action to start. Since the combination of keys and buttons will most likely trigger several input events, several input events may need to be combined before a proper decision can be made on which action to start. However, this should not cause any noticeable input lag. The object is also responsible for loading and saving of shortcut configuration and mapping it to actions.

Each of the actions is an implementation of a more generic universal action interface. Each action needs to provide a list of available shortcuts, a name that can be displayed in the configuration dialog and a trigger method that actually triggers the action. Each action will have access to both the canvas and the current tool.

Once these classes are in place, the second step is to create a configuration page that can be inserted into the configuration dialog. This page uses the action manager to list the available actions and their shortcuts.