A users guide to Flipper

Mark Bell

December 17, 2013

Flipper is a program for computing the action of mapping classes on curves and laminations on a punctured surface using ideal triangulation coordinates. Flipper is currently under development and this users guide will be based on Flipper 0.1.0. Eventually it will also be able to construct the canonical triangulation of the surface bundle associated to pseudo-Anosov mapping classes.

1 Getting and starting Flipper

You can get the latest copy of flipper from `https://bitbucket.org/Mark_Bell/flipper` or straight from the mercurial repository with the command:

```
> hg clone https://bitbucket.org/Mark_Bell/flipper
```

You can start Flipper by navigating to its folder and running the command:

```
> python Flipper.py
```

Flipper has been tested on Python 2.7 and Python 3.2 on Windows 7 and Ubuntu 12.04. Some of the functions of Flipper require exact arithmetic. This is done using either Sage (tested with Sage 5.12) or SympPy (tested with SympPy 0.7.3.rc1) but you can also create an interface to your own exact arithmetic library.

To run Flipper using Sage you must run it using Sage's Python by using the command:

```
> sage -python Flipper.py
```

Note that Sage if does not recognise your Tkinter install you may initally see an error such as `Error: no module named _tkinter`. You can fix this by installing the tcl/tk development library and then rebuilding Sage's Python. On Ubuntu you can do this using the commands:

```
> sudo apt-get install tk8.5-dev
> sage -f python
```

Flipper also includes a setup file. So if you have cx_Freeze installed as a Python module you can use the command:

```
> python setup.py build
```

to create an executable within the build directory. As with most cx_Freeze distutils scripts, you can also build a Windows installer by doing:

```
> python setup.py bdist_msi
```

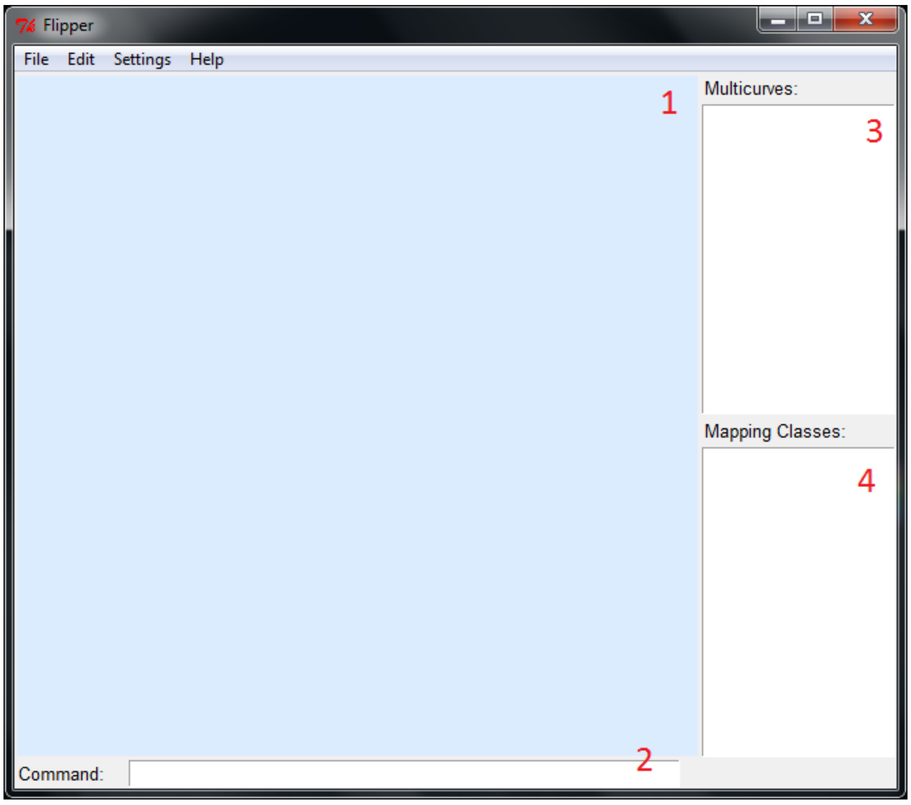
and a Mac disk image by doing:

```
> python setup.py bdist_dmg
```

At some point the BitBucket repository may also include precompiled binaries.

2 Getting Started

This is the main window of Flipper. It has a canvas for drawing on (1), a command bar for entering instructions (2), a list of known curves (3) and a list of known mapping classes (4).



The main work flow for using Flipper is:

1. Create a planar triangulation.
2. Identify pairs of boundary edges together to create a triangulation of a marked closed surface.
3. Create curves on the surface.
4. Specify mapping classes on the surface.
5. Apply some of the known mapping classes to the curve.

We now describe how to perform each of these steps in detail. First note that in each stage the currently selected object is highlighted in red. You can cancel your current selection at any time by pressing `Escape`, right clicking or double clicking.

2.1 Creating a planar triangulation

To create a planar triangulation use the Triangulation Edit mode, which can be selected from the edit menu. This is done by creating vertices in the plane and connecting pairs of them together via edges. With nothing selected, clicking on a vertex or edge will select it and clicking on the canvas will create a new vertex. With a vertex selected, clicking on the canvas or on an existing vertex will create a new vertex, if necessary, and connect the two via an edge. With an edge selected, clicking on the canvas or on an existing vertex will create a new vertex, if necessary, and connect it two the two ends of the edge via new edges. You can delete the currently selected object by pressing either `Delete` or `Backspace`.

Flipper automatically adds triangles between any triple of vertices each of which is pairwise connected via an edge. You cannot add an edge if it would meet the interior of an existing edge.

2.2 Identifying boundary edges

To identify pairs of boundary edges of the current planar triangulation use the Gluing Edit mode, which can be selected from the edit menu. With nothing selected, clicking on a boundary edge of the triangulation will select it. With a boundary edge selected, clicking on another will identify them via an affine transformation and colour them accordingly.

2.3 Creating curves

To draw a curve on the current triangulation use the Curve Edit mode, which can be selected from the edit menu. Note that this can only be accessed if the triangulation is closed. With nothing selected, clicking on the canvas will start a new section of curve. With a section of curve selected, clicking on the canvas will extend the curve to pass through the current point. You can remove the last point of the currently selected section of curve by either pressing `Delete` or `Backspace`. You should make sure to draw the curve transverse to the underlying triangulation.

The currently drawn multicurve can be named and added to the list of known curves by using the `curve <name>` command. Known multicurves can be shown by using the `show <name>` command.

2.4 Specifying mapping classes

There are currently three ways to specify a mapping class. The first is to define the mapping class to be a left Dehn twist about the currently drawn curve, see Section 2.3. This is done by using the `twist` command. This takes a `<name>` and adds this twist to the list of known mapping classes under the name `<name>`. Note that this also adds the curve to the list of known curves under the same name.

The second is, when the currently drawn curve bounds a twice marked disk, to define the mapping class to be a left half twist about the currently drawn curve, see Section 2.3. This is done by using the `half` command. This takes a `<name>` and adds this half twist to the list of known mapping classes under the name `<name>`. Note that this also adds the curve to the list of known curves under the same name.

Alternatively, a mapping class can be defined by specifying how it permutes the underlying triangles of the triangulation. This is done by using the `isometry` command. This takes a `<name>`, `<source triangle>` and `<target triangle>` and adds the mapping class induced by the the isometry which sends `<source triangle>` to `<target triangle>` to the list of known mapping classes under the name `<name>`.

In either case, the inverse of the mapping class is also added to the list of known mapping classes under the swapcase of `<name>`.

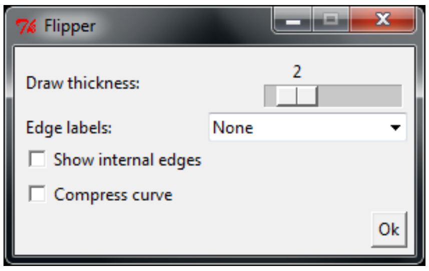
Remark. Currently Flipper is only capable of performing Dehn twists and half twists about curves where every complementary region contains at least one puncture. Hence, for example, it cannot perform a half twist on a twice marked surface.

2.5 Applying a mapping classes

To apply one of the known mapping classes to the currently drawn curve click on its name in the list of known mapping classes. The result will be instantly displayed on the triangulation. To apply the inverse of a known mapping class, right click on its name in the list of known mapping classes. Alternatively use the command `apply <mapping class>` where `<mapping class>` is a string of mapping class names and inverse names each separated by a '.'.

3 Options

The options pannel allows you to change how the triangulation and curves are drawn on the canvas.


Flipper

Draw thickness: 2

Edge labels: None

Show internal edges

Compress curve

Most of the options are self-explanatory. You can change the thickness of the vertices, edges and curves drawn on the canvas along with the text size in the command bar and lists. This can be useful if presenting. You can label the edges of the triangulation with their indices or with the geometric intersection numbers of the currently drawn curve. You can show or hide the internal edges of the triangulation. You can switch to drawing just a single copy of each parallel edge but with a multiplicity labelled on it. This is useful when dealing with very complicated curves as it can take a long time to draw render a curve when it is very complicated.

4 All commands

The main Flipper commands are:

- `curve <name>` - Stores the currently drawn multicurve as `<name>`.
- `show <name>` - Shows the multicuve with name `<name>`.
- `twist <name>` - Stores a left Dehn twist about the currently drawn curve as `<name>`.
- `half <name>` - Stores a left half twist about the currently drawn curve as `<name>`. This curve must bound a twice marked disk.
- `isometry <name> <source triangle> <target triangle>` - Stores the mapping class induced by the isometry of the underlying triangulaton as `<name>`.
- `apply <mapping class>` - Applies `<mapping class>` to the currently drawn multicurve.
- `order <mapping class>` - Computes the order of `<mapping class>`.
- `periodic <mapping class>` - Determines if `<mapping class>` is periodic.
- `reducible <mapping class>` - Determines if `<mapping class>` is reducible.
- `pA <mapping class>` - Determines if `<mapping class>` is pseudo-Anosov.

Flipper also has commands for quickly creating standard triangulations:

- `ngon <specification>` - Creates a regular ngon with either the number of sides or side gluings given by `<specification>`.
- `rngon <specification>` - Creates a regular ngon with either the number of sides or side gluings given by `<specification>` where the underlying triangulation has rotational symmetry.

The following commands are also available:

- `information` - Reports information about the underlying surface such as genus and Euler characteristic.
- `tighten` - Tightens the currently draw curve to the underlying triangulation.
- `vectorise` - Shows the edge vector of the currently drawn curve.
- `render <curve vector>` - Draws the curve with edge vector `<curve vector>`.

- `lamination <mapping class>` - Computes a lamination which is approximately projectively invariant under `<mapping class>` along with its dilatation.
- `lamination_exact <mapping class>` - Computes a lamination which is projectively invariant under `<mapping class>` along with its dilatation. This functionality requires a symbolic computation library.
- `split <mapping class>` - Computes the maximal splitting sequence of `<mapping class>`. This functionality requires a symbolic computation library.

Finally, there are Flipper commands which provide perform the actions available in the various menus:

- `new` - Resets Flipper. Equivalent to `File > New`.
- `save <filename>` - Saves the current configuration to `<filename>`. Equivalent to `File > Save`.
- `open <filename>` - Loads the configuration stored in `<filename>`. Equivalent to `File > Open`.
- `export_script <filename>` - Exports the current surface and mapping classes as a Python script to `<filename>`. Equivalent to `File > Export > Export script`.
- `export_image <filename>` - Exports the current canvas as a postscript file to `<filename>`. Equivalent to `File > Export > Export image`.
- `exit` - Exits Flipper. Equivalent to `File > Exit`.
- `triangulation_mode` - Sets mode to triangulation mode. Equivalent to `Edit > Triangulation`.
- `gluing_mode` - Sets mode to gluing mode. Equivalent to `Edit > Gluing`.
- `curve_mode` - Sets mode to curve mode. Equivalent to `Edit > Curve`.
- `erase` - Erases the current curve. Equivalent to `Edit > Erase curve`.
- `zoom` - Scales the current drawing to fill the canvas. Equivalent to `View > Zoom > Auto zoom`.
- `options` - Shows the options menu. Equivalent to `Settings > Options`.
- `help` - Shows the help information. Equivalent to `Help > Help`.
- `about` - Shows the about information. Equivalent to `Help > About`.

Arguments given to Flipper commands must meet the Conventions:

- `<filename>` must be a valid file name or path.
- `<name>` must be an string of letters, numbers and underscores. It must contain at least one letter.
- `<mapping class>` must be a string of mapping class names and inverse names, given by their swapcase, each separated by a '.'. This is read left to right to agree with the order of composition.
- `<source triangle>` and `<target triangle>` must be a string of three edge indices, each separated by a '.'.
- `<curve vector>` must be a list of edge weights each separated by a '.'.