README for Homework 4

Rong Kang Chew (rchew)

Integration of Behaviors

- **TopGraphics** had to handle the conversion of events and switching of behaviors
- **Behavior**
  - Decided to separate start(), running() etc. from the implementations in each behavior, common code would check if the event matched etc. while implementation would just handle
  - **(Improved Behavior Handling):** I noticed that stopping required the modifier keys to be held down when releasing (e.g. CONTROL_SHIFT_LEFT_MOUSE_UP) which could lead to issues when a user releases both the modifiers and the mouse button at the same time and the keys are released first. Therefore, I added an additional flag for a **"Loose Stop"** which would only match the key and direction of the event and not modifier keys. This is used in the NewBehavior.
  - The BoundingBox for lines and text has some issues, so it may be problematic to select it.
- **BehaviorEvent**
  - **Direction** was added to check for up/down/press/click/dblclick
  - Mouse-based events are tracked only with a single buttons at a time
  - JavaScript does not expose the function button, command/windows key combined
  - There is an additional looselyMatches() method that only checks id, key, direction

Programming Guide

Behaviors can be added to groups that will apply to its immediate children through .addBehavior(). Multiple behaviors can be added to one group.

Pre-built behaviors include MoveBehavior, NewBehavior and ChoiceBehavior.
Example:

```
objsGroup.addBehvaior(new MoveBehavior(objsGroup,
              new BehaviorEvent("CONTROL_SHIFT_LEFT_MOUSE_DOWN"),
              new BehaviorEvent("CONTROL_SHIFT_LEFT_MOUSE_UP"),
              new BehaviorEvent("ESC")
         ));
```

To create your own behavior, extend the `Behavior` class.
Every behavior is constructed minimally with a reference to the group with the behavior, `BehaviorEvents` describing the events that trigger the start, stop, and abortion of the behavior.

The constructor can be can be overridden, but the base Behavior class will still require those parameters.

An optional flag `allowLooseStop` allows for Behaviors to stop without the modifier keys matching, which is useful for mouse and keyboard combinations.

The new Behavior must implement the `doStart, doRunning, doStop` and `doCancel` methods to describe what happens when the matching input events are encountered for the behavior. Matching is already done by the base Behavior class and does not need to be re-implemented. The `affectedObject` property informs your behavior of the object that was selected when starting, but is not updated for running or other state changes.