dsDraw Technical Documentation

Canvas

The drawing tools in dsDraw are implemented primarily through the HTML5 <canvas> element. Some features such as editing text are simplified by bringing up other HTML elements, but once editing is done, everything on the canvas is represented by hand-coded classes that maintain their own states. The basic structure that is used to interface between user and canvas is the CanvasState class.

This is the current design for CanvasState:

Fields:

- mouseDown: object with x, y values assigned new coordinates whenever the user depresses the left mouse button
- mouseMove: object with x, y values assigned new coordinates whenever the user moves the mouse to a new coordinate (no click necessarily)
- mouseUp: object with x, y values assigned new coordinates whenever the user release the left mouse button
- drawMode: string representing current drawing tool
- canvas: reference to main HTML canvas element
- ctx: CanvasRenderingContext2D object with actual draw/fill/line methods
- objects: array of type: str, canvasObj: object objects currently on the canvas
- ctx: CanvasRenderingContext2D object with actual draw/fill/line methods
- clickedBare: boolean representing whether last click was on an object or bare canvas
- activeObj: reference to most recently clicked object
- dragOffset: x, y object representing where a drag event began (updated as object is dragged and thus separate information from mouseDown)
- hitCanvas: reference to hidden HTML canvas element for event detection
- hitCtx: CanvasRenderingContext2D object for hitCanvas
- uniqueColors: Set object with unique colors currently on hitCanvas

- colorHash: rgbString: canvasObject object/mapping from colors to current objects
- hotkeys: keyCode: bool object/mapping for keeping track of currently depressed hotkeys/modifiers

Methods:

- setMode(string mode): change drawing mode, update toolbar label
- undo(): remove most recently drawn object from CanvasState.objects
- initToolbars(): initialize toolbars for different drawing modes
- bindKeys(): initialize key bindings for hotkeys
- addCanvasObj(objType, canvasObj): add a new object (and its type as a string) to current list and do color hashing
- getClickedObject(mouseX, mouseY): returns topmost object at given coordinates
- repaint(): clear canvas and redraw each current object (and possibly "creator" elements, such as a hollow box if user is currently creating a new text box, etc.)

Canvas events

A hidden canvas with the same dimensions as the main canvas is used for constant time event detection. When a new element is added to the canvas, it is assigned a unique RGB color that is used to fill the space of the element on the hidden canvas. When the canvas is clicked, the hidden canvas is queried for the color at that coordinate, and the color is used as a key in a map of current canvas objects. Using this method eliminates the need to iterate through the current canvas objects, doing math on each to calculate its bounds and so on.

TODO:

- Implement reordering of canvas objects so the most recently clicked objects are drawn last (these constitute the topmost layer, so their colored hit-detection shapes shouldn't be colored over by other elements).

Canvas Objects

Canvas objects such as text boxes, arcs, arrays, etc. are represented as classes with a reference to the canvas state and several methods such as draw, drag, click, release, etc.. When a click event happens, it is first determined whether it is a drag, release, click (initial press down), and then the corresponding method gets called on the active canvas object.