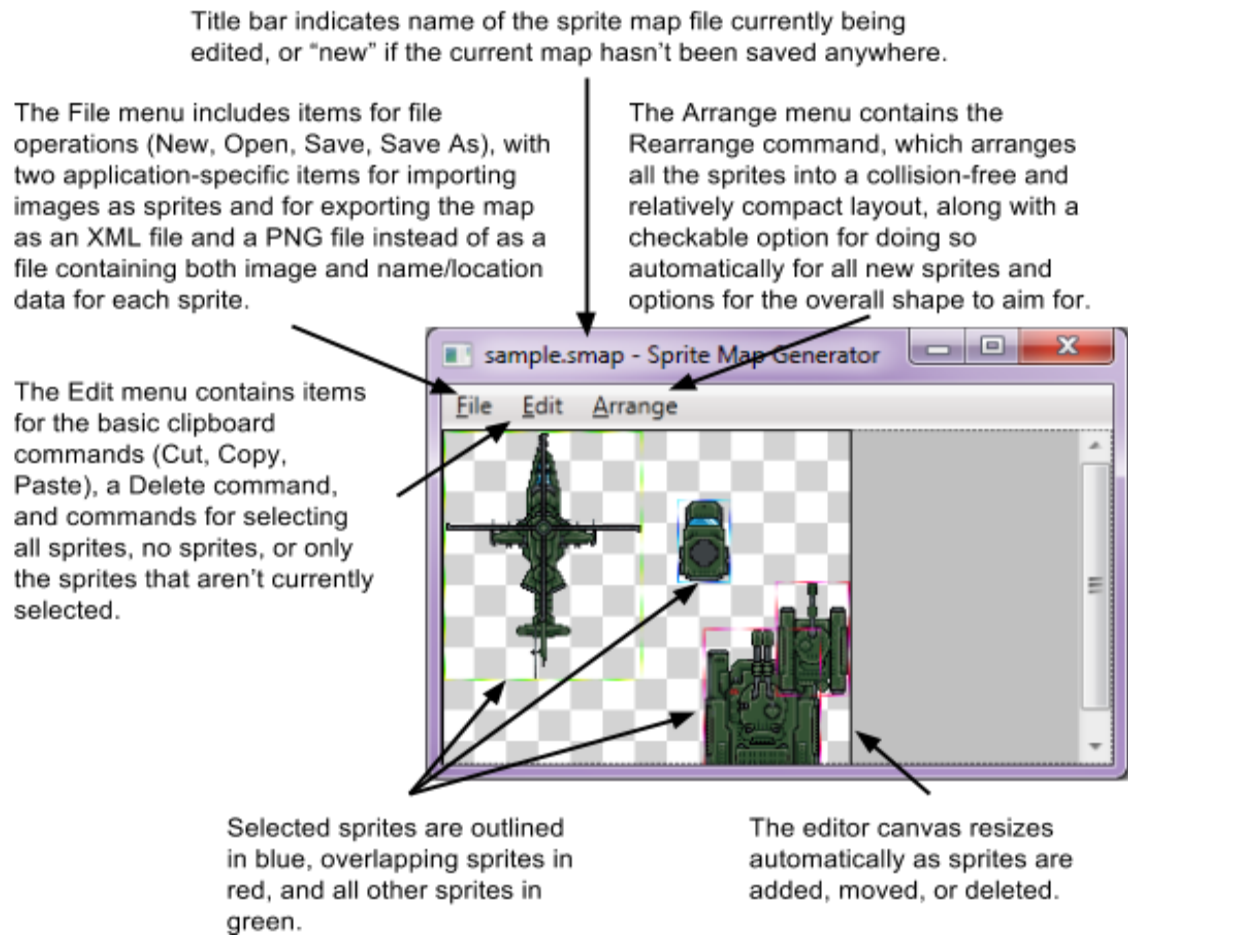
1. Analysis

A barebones sprite map needs to list only three things for each animation frame in an image: the location of the frame, the size of the frame, and an identifier of some kind so that consumers of the sprite sheet can refer to specific frames. The identifier can be as simple as an index number, but a (hopefully descriptive) name string would probably be more useful.

An editor for a barebones sprite map needs to do, at minimum, three things: allow the user to add sprites to the sheet, keep the sprites from overlapping, and keep their identifiers unique. Packing the sprites in a map into a compact area and allowing the user to remove sprites or to move them on their own are also important. Less important, though still desirable, is allowing the user to change the names of sprites (as long as the names remain unique).

An editor that allows a user to move sprites on their own must either adjust sprites after movement to remove collisions or warn the user about collisions so that they can adjust the arrangement themselves.

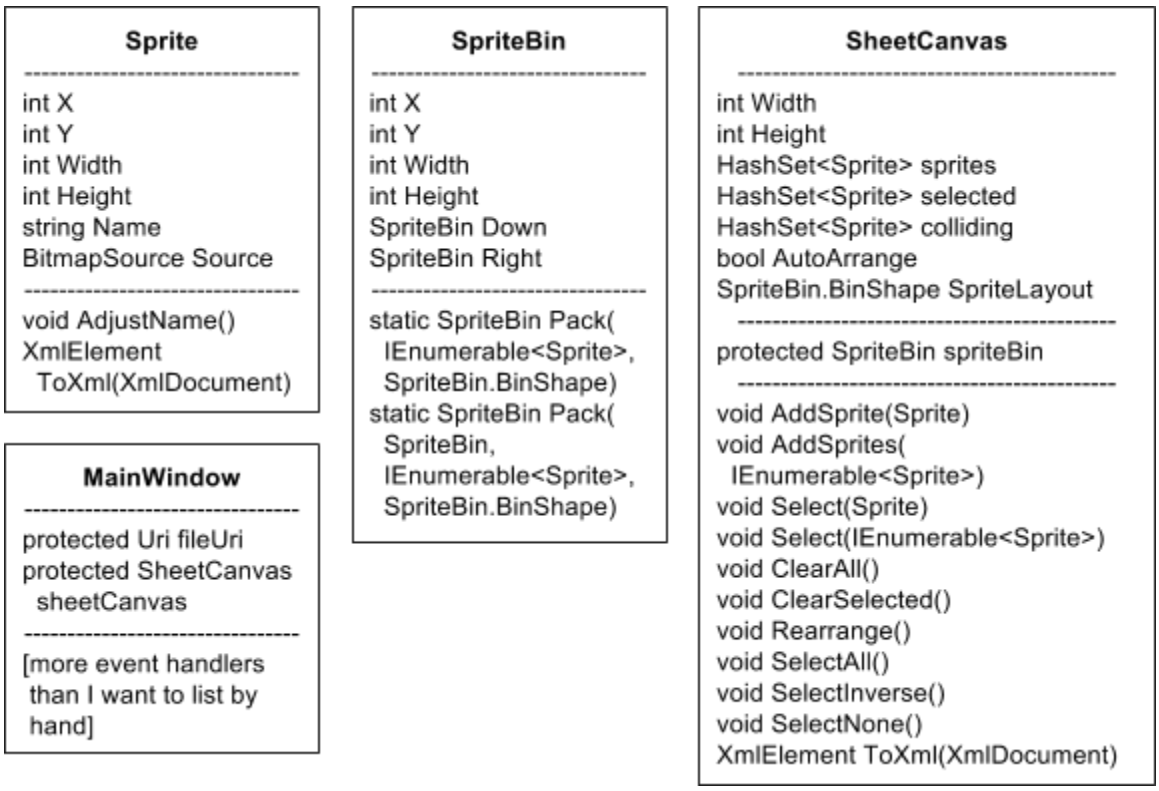
2. GUI Design



3. Application Data Structure and Algorithm Design

In order to recover from overlapping without losing data, a sprite object must store image data. The canvas containing all the sprites must keep track of which sprites are selected and of collisions between sprites (recalculated whenever a sprite is added, moved, or removed). The window as a whole needs to keep track of the current sprite map file, along with providing event handlers for all the menu items.

The SpriteBin class implements a packing algorithm posted by Jake Gordon on May 7, 2011 at http://codeincomplete.com/posts/2011/5/7/bin_packing/ in order to arrange sprites in a collision-free and relatively compact layout. The SpriteBin class includes an enum, BinShape, for indicating whether the layout should grow vertically, horizontally, or in a roughly square shape.



4. XML Design

A sprite map in pure XML, for use with a single image containing all the sprites, needs to show the size, location, and name of each sprite. Thus:

```
<sheet src=”spritesheet.png”>
      <sprite name=”square sprite” x=”0” y=”0” w=”32” h=”32” />
      <sprite name=”tall sprite” x=”0” y=”32” w=”16” h=”32” />
      <sprite name=”wide sprite” x=”32” y=”0” w=”32” h=”16” />
<sheet>
```

A sprite map within the editor, however, should also contain image data for the sprite so that it can be repositioned without complicated pixel editing and so that image data is not lost if sprites temporarily overlap. Thus, XML data for a sprite or group of sprites copied to the clipboard should contain either the location of a file containing the image data or the image data itself for the/each sprite. There are convenient system functions available for encoding PNG data to memory and for translating between raw bytes and base64 strings. Thus:

```
<sheet>
      <sprite name=”sprite in a file” x=”0” y=”0” w=”64” h=”64”
   src=”sprite_image.png” />
      <sprite name="tiny chessboard" x="0" y="64" w="8"
h="8">iVBORw0KGgoAAAANSUhEUgAAAAgAAAAICAIAAABLbSncAAAAAXNSR0IArs4c6QAAAARnQU1B
AACxjwv8YQUAAAAJcEhZcwAADsMAAA7DAcdvqGQAAAAaSURBVBhXY/j//z8DAwMmCRTEIgpSNyh1AA
Bdc1+hN5lbDQAAAABJRU5ErkJggg==</sprite>
<sheet>
```

5. Testing

Testing the application consisted of importing individual sprites from the *example* folder, copying them, pasting them directly back into the application, pasting into *Notepad++* as XML with base64-encoded image data and then copying that text from *Notepad++.exe* and pasting into the SpriteMapGenerator application, and the same for *Paint* and *Gimp* as image data (transparent for *Gimp*, white background for *Paint*), saving the resulting map as a file, creating a new file, loading the previously-saved file, and then exporting it and confirming that the resulting PNG and XML contained what I expected. Testing the different selection commands and the ability to drag and delete sprites happened by hand along the way. Testing the auto-arrange features consisted of importing several sprites, using the Rearrange command, and verifying that there were no indicated collisions and that the layout looked reasonably compact. This was tested for all three layout options (Square, Vertical Strip, and Horizontal Strip) and for all, no, or some of the sprites selected.