General Questions

How are frame settings influenced by certain images?

Usually MUI will add a unique frame to each object, unless a specific frame is enforced by the application. The kind of frame MUI chooses depends on several things.

By default all objects get a standard frame which can be configured in the MUI preferences application, i.e. a button frame for buttons. This setting can be overridden by applications for single objects if the developer thinks that this is necessary. However, for image buttons there exists a special feature which removes the frame chosen by MUI if the image comes with its own frame within the imagery and the file has a special name. For example if an image file is named "ArrowUp_framed.mim" instead of "ArrowUp.mim" MUI will set not set the standard image button frame for the created object as this would result in a double frame (one by MUI and the second by the image itself) which definitely is not intended and would look bad. In this case MUI will trust the imagery to provide a proper frame and will not draw the frame defined for image buttons.

How to create bitmap frames?

In addition the the fixed built-in standard frames MUI offers custom bitmap frames. Such frames require a truecolor screen to be displayed and offer full 8bit alpha blending. There are a few rules to be obeyed when designing your own custom bitmap frame:

- the bitmap frame should be saved as PNG. The PNG image format is able to save transparency information (alpha channel) with a depth of 8 bits. In contrast to that GIF and ILBM files can handle a single transparent color only (1 bit depth).
- the bitmap frame image consists of 3 square base images each of which is devided up into 9 subsquares. The first (left) base image represents the normal/unpressed state of the frame. The second (middle) image represents the selected/pressed state of the frame. The third (right) image represents the frame and the content area. The full bitmap frame image must have an aspect ratio of 9:3, the subparts must have an aspect ratio of 3:3. For example, a frame design with 24x24 pixels must have a dimension of 72x24 pixels.
- the "corner" parts will be drawn as they are, the "side" parts will be drawn repeatedly up to the final frame dimension.
- the real frame parts (corner and side) of the normal and the selected images may have arbitrary imagery, including full 8bit alpha blending. However, the third (right) image must be drawn in pure black and white only. The outer white area defines the frame's thickness, while the inner black area defines the usable area for the object's contents, i.e. the label of a button. Do not use other colors than black (RGB 0/0/0) and white (RGB 255/255/255) and use a square shape only.
- imagine the object's final look as a combination of 3 stacked layers:
    - bottom layer: the object's background
    - middle layer: frame
    - top layer: the object's contents (i.e. the button label)

This makes clear that the frame's thickness should be chosen in such a way that the object's contents do not cover the non-transparent parts of the frame imagery.

A very simple ASCII example:

```
111122223333AAAABBBBCCCC............
```

```
111122223333AAAABBBBCCCC............
111122223333AAAABBBBCCCC............
111122223333AAAABBBBCCCC............
4444    5555DDDD    EEEE............
4444    5555DDDD    EEEE.....##.....
4444    5555DDDD    EEEE.....##.....
4444    5555DDDD    EEEE............
666677778888FFFFGGGGHHHH............
666677778888FFFFGGGGHHHH............
666677778888FFFFGGGGHHHH............
666677778888FFFFGGGGHHHH............
```

corner parts: 1, 3, 6, 8, A, C, F, H
side parts: 2, 4, 5, 7, B, D, E, G
The '.' in the right image must be white pixels, the '#' must be black pixels.
The center parts of the left and middle images will be ignored and should be left transparent for consistency.

Note that the right images defines a frame thickness of 5 pixels, whereas the frame imagery might be up to 4 pixels thick only. Thus the frame will have an additional space of one pixel between the frame imagery and the object contents. These proportions are your design decisions. The thickness does not need to be symmetrical. Even something like pixel at the top, 2 pixels on the left, 5 pixels on the right and 4 pixels at the bottom is allowed.

Such a frame design might be drawn like shown below. The additional inner frame indicated by "+—-+" is there to show the additional space between the frame image and the inner contents as described above. It is not visible within the MUI GUI.

```
1111222222222222222222222222222222222223333
1111222222222222222222222222222222222223333
1111222222222222222222222222222222222223333
1111222222222222222222222222222222222223333
4444                                    5555
4444  +----------------------------------+ 5555
4444  |                                  | 5555
4444  |                                  | 5555
4444  |                                  | 5555
4444  |     This is a normal button      | 5555
4444  |                                  | 5555
4444  |                                  | 5555
4444  |                                  | 5555
4444  +----------------------------------+ 5555
4444                                    5555
6666777777777777777777777777777777777778888
6666777777777777777777777777777777777778888
6666777777777777777777777777777777777778888
6666777777777777777777777777777777777778888
```

```
AAAABBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBCCCC
AAAABBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBCCCC
AAAABBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBCCCC
AAAABBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBCCCC
DDDD                                    EEEE
DDDD +----------------------------------+ EEEE
DDDD |                                  | EEEE
DDDD |                                  | EEEE
DDDD |                                  | EEEE
DDDD |      This is a pressed button    | EEEE
DDDD |                                  | EEEE
```

```
DDDD |                                      | EEEE
DDDD |                                      | EEEE
DDDD +--------------------------------------+ EEEE
DDDD                                          EEEE
FFFFGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGHHHH
FFFFGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGHHHH
FFFFGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGHHHH
FFFFGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGHHHH
```

I hope this makes the basics a bit more comprehensible. In case of doubt feel free to ask. Also refer to the included frame images.

How much stack memory is required for MUI based applications?

MUI applications require a certain amount of stack memory during creation of the GUI due to the typical deeply nested code. The default stack size of 8K (8192 bytes) is usually not enough. A too low stack size will cause unpredictable crashes of either the MUI application or random other applications due to a stack overflow and hence trashed memory.

Since AmigaOS has no concept of automatic stack enlargement the only solution is to provide enough stack memory from the beginning. Its size can be defined either in the application's Workbench icon or, if the application is started from a shell or a script, by setting a sufficiently large stack by running C:Stack with an appropriate size.

MUI for AmigaOS has a built-in check whether the stack size of an application is large enough. If the stack size is found to be too small a warning requester will be displayed to inform the user about this nuisance. The current minimum stack size requirement is at least 32K (32768 bytes) to avoid the stack overflows mentioned above. Please note that this size is **NO** guarantee. It is just a compromise between safety and excessive memory usage. There may be applications with a complex GUI which requires more stack memory. MUI cannot know the required amount of stack memory as the application defines the complexity of the GUI. Further the warning requester might be opened too late and the possible harm was done already, because usually the Application object is created **after** the GUI, but only the Application class can do all the necessary checks and build an appropriate warning requester with all necessary informations.

Unfortunately there exist some applications which launch a subprocess which then creates the MUI application and the GUI. Most of the time this subprocess' stack size is fixed and too low and cannot be adjusted by the user. In this situation the only way to increase the stack size to avoid the warning requester and possible crashes is a patch like ?StackAttack, ?StackAttack2 or ?MinStack. Such patches will apply to **all** applications, not just MUI applications.

Some known applications with subprocesses and too low stack sizes are MagicASL (an MUI based ASL requester replacement) and the Poseidon USB stack (its popup task).