Problem

When ***dissolve*** goes with ***motion51***, dissolve effect is not correctly applied when video driver OpenGL enabled. It shows only a blank frame or a static (frozen) frame without being blended with new frames of the video. This will not happen when OpenGL is disabled or only ***dissolve*** is applied.

Root cause

Video effects (like ***dissolve*** and ***motion51*** plugins) when applied to a video clip will be organized into some kind of "chain". The first plugin calls the second one. This make the second plugin is applied first then the first plugin will be applied the second. For the case of ***dissolve*** and ***motion51***, motion51 plugin calls dissolve at the beginning of its processing cycle (line 437 in motion51.C):

```
int Motion51Main::process_buffer(VFrame **frame, int64_t position, double frame_rate)
{
    ...
    read_frame(out, target_layer, out_position, frame_rate, use_opengl);
    ...
}
```

`read_frame` function call at that line will lead to `process_realtime` function in dissolve.C is called (line 57 in dissolve.C). **out** variable in the above line of code is a video frame (VFrame) and it is expected to be filled with data processed by `process_buffer` in dissolve plugin (that is, line 57 in dissolve.C).

Now, let's have a look at the function process_realtime from line 57 in dissolve.C

```
int DissolveMain::process_realtime(VFrame *incoming, VFrame *outgoing)
{
        fade = (float)PluginClient::get_source_position() /
                        PluginClient::get_total_len();
// Use hardware
        if(get_use_opengl())
        {
                fprintf(stderr, "DissolveMain::process_realtime: use openGL\n");
                run_opengl();
                return 0;
        }
// Use software
        if(!overlayer) overlayer = new OverlayFrame(get_project_smp() + 1);

        overlayer->overlay(outgoing, incoming,
                0, 0, incoming->get_w(), incoming->get_h(),
                0, 0, incoming->get_w(), incoming->get_h(),
                fade, TRANSFER_SRC, NEAREST_NEIGHBOR);

        return 0;
}
```

outgoing is the same as **out** in `read_frame` function call, and it should be filled with data after dissolve process_realtime finishes so that motion51 can have data for it

processing. However if OpenGL is enabled **outgoing** will not be proccessed at all (`run_opengl()` function call have no arguments. Compare it with `opeverlayer->overlay()` function call). OpenGL processing actually uses different datapath related to `handle_opengl` callback and playback command at video device layer. Function `run_opengl()` is just for kickstarting that procedure.

Conclusion

We have to disable OpenGL for transition plugins (e.g. dissolve). If dissolve is running using OpenGL, there is no way to pass data after proccessed by dissolve to the next plugin (motion51). So, instead of disabling OpenGL in Motion51 during transition, I suggest that we just disable openGL for dissolve and let motion51 uses OpenGL all the time. I think there is no way to '*fix*' Cinelerra in this sittuation, or we have to change plugin architecture of Cinelerra for OpenGL processing.

Notes

Setting `use_opengl = 0;` as you did will disable OpenGL for both motion51 and dissolve plugin (that's why it works). Commenting out "Use hardware" branch in dissolve.C as my first patch for this will only disable OpenGL for dissolve.

At this moment, I can provide you a patch that disable OpenGL for all video transition plugins (not only dissolve) and let other effect use openGL all the time.